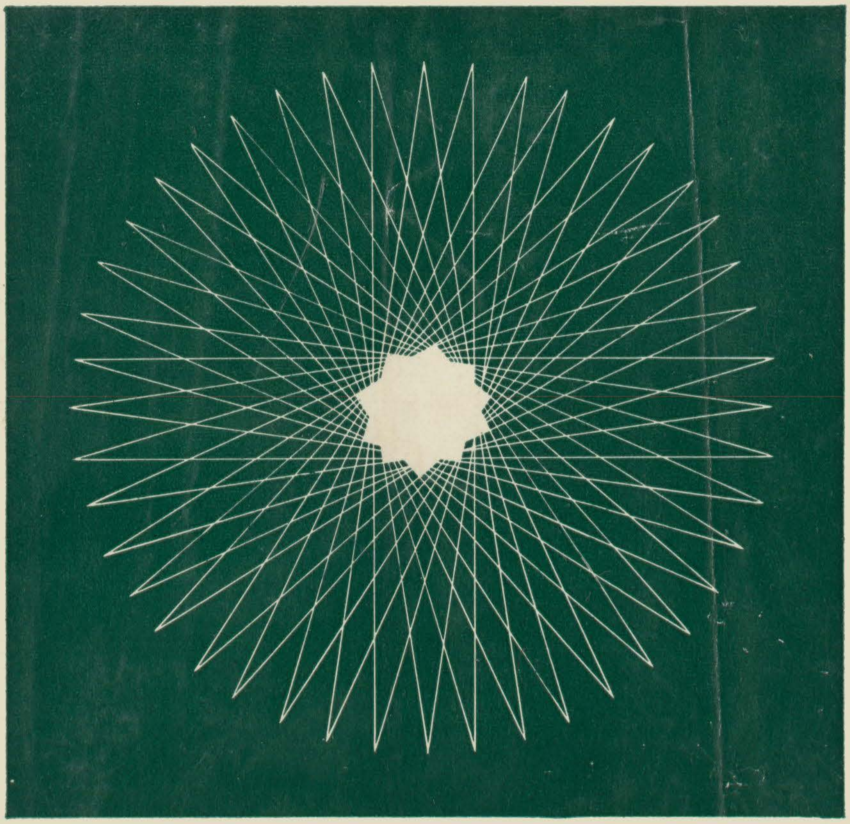
ELECTRONIC DIGITAL COMPUTERS



CHARLES V. L. SMITH
Goddard Space Flight Center
National Aeronautics and Space Administration

McGRAW-HILL BOOK COMPANY

ELECTRONIC DIGITAL COMPUTERS

Electronic Digital Computers

CHARLES V. L. SMITH

Chief, Computing Laboratory
Aberdeen Proving Ground

McGRAW-HILL BOOK COMPANY, INC.

New York Toronto London

1959

ELECTRONIC DIGITAL COMPUTERS

Library of Congress Catalog Card Number 58-10009

THE MAPLE PRESS COMPANY, YORK, PA.

To my wife
Virginia H. Smith

PREFACE

This is not a treatise on digital computer engineering, nor on the other hand an exhaustive treatise on logical design. The reader will find considerable discussion of circuits and components—enough, I hope, to give him a reasonably complete understanding of various ways in which the usual functions of a computer, such as memory, control, the performance of arithmetic, and input and output, can be realized physically. But I have not attempted a treatment sufficiently detailed to provide design information. The reader will also find sufficient information on computer arithmetic and instruction codes to provide him with a basic understanding of these matters, but here again I have not attempted the detailed treatment that a logical designer would demand, and I have for brevity considered only machines using binary arithmetic. My purpose has been to provide the reader with sufficient information to understand how digital computers function.

To this end, the first twelve chapters, which deal with arithmetic and logic and with circuits and components, are followed by four chapters which attempt to explain the organization of a system, specifically the parallel direct-coupled asynchronous machine developed at the Institute for Advanced Study by the group led by the late John von Neumann and H. H. Goldstine. I have chosen this particular system partly because it is the one with which I am most familiar and partly because I consider the work at the Institute to be of the very highest importance for the development of the computer art. The original report "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," written by von Neumann, Burks, and Goldstine under an Army Ordnance contract and published in 1946, is a classic report that has profoundly influenced all subsequent developments. It is very interesting to find that this original logical design, improved by the incorporation of *B* registers and certain additional elements designed to provide for concurrent rather than strictly sequential operation, has recently resulted in the design of a highly flexible superspeed machine by members of the Digital Computer Laboratory of the University of Illinois. Some account of this is given in the concluding chapter.

It should not be forgotten that the Institute for Advanced Study program was carried out under an Army Ordnance contract. The Ordnance Corps had already supported the development of the ENIAC and the EDVAC to meet the needs of the Ballistic Research Laboratories, Aberdeen Proving Ground. The digital-computer industry and all who are interested in the progress of the art of computation thus owe the Ordnance Corps a debt of gratitude for the foresight shown in supporting these programs.

In rather extreme contrast with the Institute for Advanced Study machine and its numerous progeny is the strictly dynamic circuitry developed at the National Bureau of Standards in SEAC, DYSEAC, and a very fast parallel computer now under development. I have given a fair amount of information about these circuits and about components such as counters, shifting registers, and adders based upon them, but have not attempted to describe the system organization of a complete machine.

I regret that I have not had time or space to cover the pioneer work at Harvard and at the Massachusetts Institute of Technology. The Harvard Computation Laboratory under the leadership of Professor Aiken introduced many new components and methods of design: for example, the use of a logical algebra (or algebra of logic) in the design of switching circuits. Fortunately the work at Harvard has been well documented in the volumes of the *Annals of the Computation Laboratory* published by the Harvard University Press. The work at MIT led to the first really high-speed machine (Whirlwind I) to be successfully operated. I have given some detail about the high-speed shifting registers and adder of this machine.

The excellent work of the manufacturers of computing equipment is not much discussed, largely because relatively little detailed information about it is readily available. The principal items covered are the magnetic-core memory built by Telemeter Magnetics under Army Ordnance contract and installed in the ORDVAC at the Aberdeen Proving Ground and a magnetic-drum memory built by Sperry Rand for use with the same machine. Both these memories have seen several years of continuous and highly reliable use on a twenty-four-hour-per-day seven-day-per-week schedule.

I am happy to acknowledge the kind permission given me by Telemeter Magnetics, Inc., and by the Sperry Rand Corporation to include descriptions in Chapter 14 of the above-mentioned equipments manufactured by them. I also wish to record my gratitude to Dover Publications for permission to use material from Prof. P. C. Rosenbloom's "Elements of Mathematical Logic" in Chapter 7, and to Dr. J. A. Rajchman and the *RCA Review* for permission to reproduce Figure 7-15. Finally I wish to record my great debt to my wife, Virginia H. Smith, for her constant encouragement and for the great help she has given me in the preparation of the manuscript and in the reading of the proof.

Charles V. L. Smith

CONTENTS

CHAPTER 1

DIGITAL-COMPUTER ARITHMETIC

1-1. Introduction. The convenient and expeditious performance of the familiar operations of arithmetic would hardly be possible without the use of positional representation of numbers in terms of powers of an integer base. This is because, in a positional system, it is necessary to remember only very simple tables of elementary sums and products, that is, sums and products of all possible pairs of integers each of which is less than the base of the system. From the successive performance of such operations, sums and products of numbers of arbitrary magnitude are built. If b is the base (a positive integer), then

$$N = \sum_{i=-m}^{i=+n} a_i b^i \qquad 0 \leq a_i < b \tag{1-1}$$

and only the sums $a_i + a_j$ and the products $a_i \times a_j$ need be remembered in order to perform all the operations of arithmetic.

Computation performed by successive operations on the digits a_i and a_j of two numbers N and N' is "digital computation," and any device that can be caused to perform such a computation is a "digital computer." Simple devices for carrying out these processes have of course long been known. The significant innovations of (1) retaining or "storing" numerical data until it is needed and (2) causing the computing element to perform automatically an arbitrary sequence or "program" of operations seem first to have occurred to Charles Babbage considerably over a century ago. Any device that fulfills these functions is properly called an "automatically sequenced digital computer." Obviously, it is necessary also to provide a means of inserting into the device both the quantities upon which it is to operate and the instructions governing these operations as well as a means of extracting the result of the operations. This, then, defines the conventional analysis of the units of a digital computer: input, output, memory, arithmetic unit, and control. It is better to regard these components as functions, for the equipment needed to perform them is not necessarily localized into "units," but can be distributed physically throughout the machine. This is particularly true of the equipment used to carry out the control function. Indeed, some machine designs use the same circuits on a time-sharing basis, now as part of the control, now for performing arithmetical operations.[1]

Of course, the functions needed for the construction of a computer can be instrumented in a variety of ways. Present machines rely for the most part upon vacuum tubes and crystal diode circuits, whereas earlier machines (e.g., Harvard Mark I and Mark II) used electromechanical elements, such as counter wheels and relays. Binary magnetic elements are in extensive use, chiefly for the memory function, and the use of transistors is steadily increasing. In a few years' time the standard machine will, presumably, use no vacuum tubes at all but will consist entirely of "solid-state" devices such as magnetic cores, transistors, and crystal diodes. Examples of the use of these elements as well as of the use of vacuum tubes will be given in later chapters.

1-2. Number Systems and Their Machine Representation. The choice of the manner in which numbers are to be represented is basic. The base of the scale of notation must be chosen first. This, however, is not sufficient to describe the number system. The radical point (the general case of what, in the decimal system, is called the "decimal" point) may be fixed or freely movable. If it is movable, the numbers may be represented in the "floating-point" form

$$N = ab^m \tag{1-2}$$

where a = number with fixed radical point
b = base
m = positive or negative integer

The number of significant digits must also be specified.

As yet the form of the coded instructions that cause the computer to execute sequences of operations has not been considered. Important decisions concerning the instructions influence and are influenced by the choice of number representation. Finally, the arithmetical and logical operations to be performed must be selected, and these choices in turn influence strongly the structure of the instruction code.

First, consider the choice of the mode of representation of numbers. Since

$$N = \sum_{i=-m}^{i=+n} a_i b^i \tag{1-3}$$

a device used to store a single digit must be capable of assuming at least b recognizable states; any redundancy introduced by having available more than b recognizable states can be used as a means of detecting errors.

It was pointed out above that, when the arithmetical operations are

[1] W. L. van der Poel, A Simple Electronic Digital Computer, *Appl. Sci. Research*, sec B, vol. 2, no. 5, pp. 367–400, 1952.

performed by human beings, reliance is placed on memorized tables of sums and products of two integers both of which are less than the base of the scale of notation. If $b = 10$ and if the person doing the computing is ignorant of the commutative laws of addition and multiplication, he will need to remember 100 sums and 100 products. Knowledge of each law reduces the pertinent cases from 100 to 55. In general, the number of sums or products needed is b^2 without the commutative law and $b(b + 1)/2$ with it. Clearly, electrical devices for performing addition and multiplication should possess some such built-in table.

Considerations of storage construction and space and of the performance of arithmetical operations, as outlined above, indicate why the binary (or base 2) system of notation is so popular in digital-computer design: for the binary system, the storage element need be capable of assuming only two recognizable states. This condition is easily satisfied in a number of ways. The simple Eccles-Jordan trigger pair, or "flip-flop," possesses two stable states, one of which may be used to designate binary 1 and the other, binary 0.

A magnetized spot on a wire, tape, or surface coated with magnetizable material may be used to designate binary 1, and the absence of such a spot in a specified position may be taken to designate binary 0. A number of storage devices of this kind will be described in the appropriate section. If information is to be stored dynamically in the form of circulating trains of pulses, a pulse may be used to designate binary 1 and its absence at a particular time may designate binary 0.

Binary arithmetic is particularly simple. The addition table consists of

$$\begin{aligned} 0 + 0 &= 0 \\ 1 + 0 &= 1 \\ 0 + 1 &= 1 \\ 1 + 1 &= 10 \end{aligned} \tag{1-4}$$

and the multiplication table is equally short,

$$\begin{aligned} 0 \times 0 = 0 \times 1 &= 1 \times 0 = 0 \\ 1 \times 1 &= 1 \end{aligned} \tag{1-5}$$

Of course, use of the binary system also entails some disadvantages. When a problem is being prepared for machine solution, it is a nuisance to be forced to convert the numerical data into binary notation. Hence, the usual procedure is to insert it in coded-decimal form and to supply at the outset instructions that cause the machine to perform the necessary conversion. Similarly, instructions are also inserted that cause the binary-to-decimal conversion of results that are to be removed from the machine. The disadvantage of this scheme is that the storage of the conversion instructions in the memory absorbs capacity that might otherwise be more usefully employed. A related disadvantage is often alleged, since the operator must sometimes interpret stages in computation (as well as input and output). For purposes of trouble shooting, it is customary to provide indicators that display the contents of various registers (units for temporary storage of information). If the binary system is used, the proper interpretation of this information requires facility in the operations of binary arithmetic.

However, when it is necessary for persons preparing problems for a machine to deal with numbers or instructions in binary notation, the very length of the expressions becomes troublesome, especially from the standpoint of recollection. For compactness, it is possible to use octal ($b = 8$) or sexadecimal ($b = 16$) notation; these systems have the obvious conversion property that each digit corresponds to a triad or a tetrad, respectively, of binary digits. Thus, to convert 10110111 into octal notation, write 010 110 111 and replace each triad of binary digits by the equivalent decimal digit, obtaining 267. To convert a sexadecimal, digits must be invented to represent decimal "ten" to "fifteen." The first six lower-case letters of the alphabet, *a*, *b*, *c*, *d*, *e*, and *f*, may conveniently be used. This being agreed, write 1011 0111 and replace tetrads. The first tetrad 1011 is decimal "eleven," or "*b*," and the second 0111 is decimal "seven," or "7." So the sexadecimal equivalent of binary 10110111 becomes *b*7.

The conversion of a number written in one scale of notation to another is accomplished by a simple algorithm, which is to be found in some of the older textbooks on algebra.[1] Given an integer N; required to write it using base b, that is,

$$N = a_n b^n + a_{n-1} b^{n-1} + \cdots + a_2 b + a_0 \qquad (1\text{-}6)$$

First divide N by b, obtaining as quotient q and remainder r:

$$q = a_n b^{n-1} + a_{n-1} b^{n-2} + \cdots + a_2 b + a_1 \qquad r = a_0 \qquad (1\text{-}7)$$

thus determining a_0. Next divide the quotient by b and obtain a_1 as remainder, and continue in this way until the quotient a_n is reached. For fractions,

$$N = \frac{a_{-1}}{b} + \frac{a_{-2}}{b^2} + \cdots \qquad (1\text{-}8)$$

Observing that a_{-1} is the integral part of bN, subtract a_{-1} from bN, and observe that a_{-2} is the integral part of $b(bN - a_{-1})$, and so on.

If, however, it is desired to use the decimal rather than the binary scale of notation, the simplicity of binary devices is ordinarily exploited to the extent that each decimal digit is replaced by its binary representation or by a binary-coded equivalent. Four bits (binary digits) are required

[1] See, for example, H. S. Hall and S. R. Knight, "Higher Algebra," 4th ed., pp. 59–62, The Macmillan Company, New York, 1904.

to represent each decimal digit. Thus the number 823 can be represented by

$$0100 \quad 0010 \quad 0011 \qquad (1\text{-}9)$$

where in each group of four the bits have their usual meaning.

A number of variants are possible.[1] In a scheme known as the "2*421" system, the four bits in sequence may represent 2, 4, 2, 1, where the first "2" is used in representation of all integers larger than 4.

Decimal	*Coded decimal*
0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

(1-10)

This scheme has the advantage that, in subtracting each digit from 9 to form a 9's complement, it is necessary merely to replace, in the coded representation of each decimal digit, binary 1 by 0 and vice versa. Additional attractive features of this scheme are that (*a*) even numbers have a 0 in the 1's position, odd numbers a 1; (*b*) numbers equal to or larger than 5 have a 1 in the 2's position, those less than 5, a 0; (*c*) three of the four positions (the last three) have the same weights as in the binary system, so many of the simple properties of that system are retained.[2]

Another scheme, known as the "excess-three" system, is to represent each decimal digit by the binary representation of that digit plus 3.

Decimal	*Excess-three*
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

(1-11)

The simple-complementation property of the 2*421 code also obtains here.

[1] Staff, Harvard Computation Laboratory, "Synthesis of Electronic Computing and Control Circuits," chap. 11, Harvard University Press, Cambridge, Mass., 1951.

[2] M. V. Wilkes, Automatic Calculating Machines, *J. Roy. Soc. Arts*, vol. C, no. 4862, Dec. 14, 1951.

In any of these 4-bit schemes (and clearly 4 is the smallest possible number of bits) for encoding the decimal digits, an n-place decimal quantity requires $4n$ bits in its coded representation. On the other hand, if the n-place decimal quantity were converted into binary notation, the number of digits required for its representation would be smaller. This can be shown as follows.

The largest n-place decimal integer is

$$10^n - 1 \tag{1-12}$$

Let
$$10^n \doteqdot 2^{m*} \tag{1-13}$$

so
$$n \doteqdot m \log_{10} 2 \tag{1-14}$$

or
$$m \doteqdot 3.32n \tag{1-15}$$

Hence, at least as far as the number of bits per quantity is concerned, the binary system is more efficient than any of the binary-coded-decimal systems. Of course, machines using any of these schemes of number representation are basically binary, at least as far as circuit elements are concerned, and have, therefore, been dubbed "disguised" binary machines.[1]

It should be mentioned that any binary-coded-decimal representation may lead to complication in the structure of the adding circuits, that is, the circuits used to accomplish addition. For example, consider the addition of two numbers m and n in the excess-three system. The machine representations of these numbers are $m + 0011$ and $n + 0011$, respectively, and the sum of these is $m + n + 0011 + 0011$, whereas the correct representation of the sum would be $m + n + 0011$. Hence, it is necessary to carry out a corrective operation every time two decimal digits are added. It has been stated,[2] however, that this drawback is more than compensated for by the simplicity of complementing referred to above.

Nothing has yet been said concerning the encoding of other than numerical information. It is, clearly, necessary that machines designed for commercial and statistical purposes be able to handle not only decimal digits but also alphabetic characters, punctuation marks, typewriter symbols, and other special characters. The UNIVAC system,[3] which handles 64 characters, employs a 7-bit code, in which one bit is used for checking purposes only. Of the remaining six, if the two bits in the most significant positions, the zone indicators, are 0's, the four least significant represent the excess-three binary-coded form of a decimal digit. If either

* Where $\doteqdot$ means "approximately =."

[1] M. V. Wilkes, *op. cit.*

[2] "Review of Electronic Digital Computers," Joint AIEE-IRE Computer Conf., p. 11, American Institute of Electrical Engineers, February, 1952.

[3] *Ibid.*

or both of the zone indicators are 1's, then an alphabetic character, punctuation mark, or typewriter symbol is indicated. Special circuits allow the nondigital characters to bypass the arithmetic circuits.

Some mention has already been made of the need to define the position of the radical point. The earlier practice was to keep this point fixed and to represent all numbers as proper fractions. This practice requires that scale factors be allowed for in coding the problem for machine computation. The only reason for using a fixed radical point was that it required less equipment than the "floating" radical point. Codes can be written that cause a fixed-radical-point machine to operate as if a floating point were in fact used. However, this is far from an ideal solution, since the floating-point instructions waste memory capacity and since the carrying out of these instructions each time an arithmetical operation is performed effectively decreases the machine's computing speed. The modern practice is to represent numbers in the floating-point form; in many machines the option of using either fixed-point or floating-point representation is provided. A floating-point arithmetic unit must perform operations on the fractional part m and on the exponent n of the numbers of the form $m \times b^n$ presented to it. In multiplication and division the treatment of the exponents is very simple, but in addition and subtraction one of the operands must be suitably shifted to make the exponents agree. The treatment of binary arithmetic below will be based on fixed-point numbers, for in any case it is fixed-point numbers that the arithmetic unit actually adds, subtracts, multiplies, and divides.

In a fixed-radical-point design, the choice of the position of the radical point can be made at will. Certain advantages are obtained if the point is placed just before the most significant position, for then the possibility of overflow (or spilling out of the storage space available) when multiplication is performed is eliminated, although overflow may still occur in addition, subtraction, or division. Some machines are so designed that the fixed position of the point can be arbitrarily chosen for the problem in hand.

The EDVAC makes use of 44-bit quantities; the first 43 bits represent the absolute value of the quantity, and the 44th is its sign.

$$a = \left(\sum_{i=1}^{i=43} 2^{-i} a_i \right) (-1)^{a_{44}} \tag{1-16}$$

In this case, as usual in machines using serial storage and a serial arithmetic unit, numbers are represented within the machine as temporal trains of pulses. At any point in the circuitry, the arrival times of the pulses are in inverse order of the significance of the digits represented. This convention is dictated by the nature of serial adders, which must add the least-significant digits first, obtaining a sum and a carry, the latter being then delayed until the digits in the next higher place arrive, when it is added to their sum, and so on. The sign digit precedes the digits representing the numerical quantity. If a parallel adder is used, the position of the sign digit is a matter of choice, though it usually is the most-significant-digit position.

An alternative scheme for the representation of negative numbers is by complementation. The complement of a number $N = \sum_{-m}^{+n} a_i b^i$ is usually taken to be

$$b^{n+1} - \sum_{i=-m}^{i=+n} b^i a_i \tag{1-17}$$

which is called the complement of N "on b^{n+1}." If $a > 0$, $-a$ is represented by the complement of a. For example, the complement (on 10^0, or 1) of 0.8436 is $0.1564 = 1.0000 - 0.8436$. This is the "10's complement" of the given number. The so-called "9's complement" is found by subtracting the number from $b^{n+1} - b^{-m}$. In the example, this yields $0.9999 - 0.8436 = 0.1563$, which differs from 0.1564 only in the least significant place. If the binary system is used, the complement may be formed by subtracting either from 2^{n+1} or from $2^{n+1} - 2^{-m}$. These analogues of the 10's and 9's complements, respectively, are commonly referred to as the "2's" and "1's" complements.

Of course, the fact that a negative number is being represented must be indicated in some manner. In the binary system, if parallel arithmetic units are to be used, the sign bit may precede the quantity and the radical point may be regarded as lying between the sign bit and the most significant digit. Thus -0.01101 may be represented by 1.10011 or by 1.10010, according as 2's or 1's complements are used. Here, in the first case, for the negative fraction $-a$ the number

$$2 - a = 1 + 1 - a \tag{1-18}$$

is written; this is also the complement of a with respect to the first power 2^1 of the base. This remark has considerable arithmetical significance,[1] as will be shown in detail later in this chapter.

For the present, the question of ease in forming complements will be discussed. Consider first the case of decimal numbers represented in

[1] A full discussion of this mode of representation is given by A. W. Burks, H. H. Goldstine, and J. von Neumann, "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," a report prepared under U.S. Army contract W-36-034-7481, Institute for Advanced Study, Princeton, N.J., 1st ed., June 28, 1946; 2d ed., Sept. 2, 1947.

binary-coded form. If either the 2*421 or the excess-three scheme is used, the 9's complement is formed merely by replacing, in the coded form of each digit, 0's by 1's and vice versa. Again, in the true binary system, the corresponding complement is formed by replacing 0's by 1's and vice versa. In both schemes, the 10's complement and the corresponding 2's complement can be effected by forming the 9's complement or its binary analogue, the 1's complement, and then adding a unit in the least significant place. Thus, in the binary system, let

$$a = \sum_{i=1}^{i=n} 2^{-i} a_i \tag{1-19}$$

Then the complement of a on $2 - 2^{-n}$ is

$$1 + \sum_{i=1}^{i=n} 2^{-i} \bar{a}_i \tag{1-20}$$

where $\bar{a}_i = 1 - a_i$. And the complement on 2 is

$$1 + \sum_{i=1}^{i=n} 2^{-i} \bar{a}_i + 2^{-n} \tag{1-21}$$

There are thus available three ways of representing negative numbers:

1. Minus sign and absolute magnitude
2. Complement on b^{n+1} (10's complement)
3. Complement on $b^{n+1} - b^{-m}$ (9's complement)

Each of these has its advantages and its disadvantages in relation to the design of arithmetic units. If complementation is used to represent negative numbers, an adder is also a subtractor, for a subtrahend has merely to be complemented before it is permitted to enter the adding circuits. Of the two types of complements, the 9's complement is simple to form, but its use in computation requires an "end-around carry," as will be seen.

Multiplication can be instrumented most easily by constructing the product of the absolute values and affixing the appropriate sign; if complementation is used, corrections must be made whenever a negative number is a factor in the product being formed. However, this correction is not difficult to make, and, since complementation permits all arithmetical operations to be performed by a single adder, it has very definite attractions.

1-3. Addition and Subtraction. It is important to examine how the various schemes of number representation affect the carrying out of the processes of arithmetic. At the outset, let it be noted that the ordinary representation of numbers by signs and absolute value, though it seems simple and obvious because it is customary, is not simple in relation to the instrumentation of the addition process. Take an example first from decimal arithmetic: form the sum

$$(+7{,}652) + (-8{,}324) = -672 \tag{1-22}$$

The equipment for this operation must execute the following tasks:

1. Compare the signs of the addends and note that a difference of the absolute values is called for.
2. Compare the absolute values and remember the sign of the larger.
3. Form the difference of the absolute values and affix to this the remembered sign.

The process becomes simpler if complements are used to represent negative numbers. Suppose that only four-place decimal integers are to be used, that positive numbers are to be represented by five decimal digits, the first zero, and that negative numbers are to be represented by their complements on 10^5. Then each negative number will start with a 9 in the leading position: in fact, the 9 can be regarded as a sign indicator followed by the complement on 10^4. Thus $-8{,}324$ is represented as

$$\begin{array}{r} 100{,}000 \\ \underline{-8{,}324} \\ 91{,}676 \end{array} \tag{1-23}$$

If this is added to $+7{,}652$,

$$\begin{array}{r} 07{,}652 \\ \underline{91{,}676} \\ 99{,}328 \end{array} \tag{1-24}$$

Since $99{,}328 = 100{,}000 - 672$, the result represents the correct answer, -672. If this scheme is used, the equipment is not called upon to make comparisons or to remember signs, but need only be able to add.

Now consider the sum

$$(+8{,}324) + (-7{,}652) = +672 \tag{1-25}$$

Using complements,

$$\begin{array}{r} 08{,}324 \\ \underline{92{,}348} \\ 1|\overline{00{,}672} \end{array} \tag{1-26}$$

which will yield the correct result, provided that any attempted carry to the left from the leading position is discarded; the vertical line drawn immediately after the 1 is intended to indicate this. Similarly, the addition of negatives can be readily handled; thus

$$\begin{array}{r} -2{,}432 \\ \underline{-3{,}318} \\ -5{,}750 \end{array} \qquad (1\text{-}27)$$

becomes

$$\begin{array}{r} 97{,}568 \\ 96{,}682 \\ 1|\overline{94{,}250} \end{array} \qquad (1\text{-}28)$$

which is correct.

With this scheme no two numbers can be added if the sum of their absolute values exceeds 9,999. Of course there is some such limitation in any fixed-point system where the length of the number is limited.

Instead of complements on 10^5, it would of course have been possible to use complements on $10^5 - 1$. The first example (1-24) would become

$$\begin{array}{r} 07{,}652 \\ \underline{91{,}675} \\ 99{,}327 \end{array} \qquad (1\text{-}29)$$

which is correct. In the second example (1-26) the result would be

$$\begin{array}{r} 08{,}324 \\ 92{,}347 \\ 1|\overline{00{,}671} \end{array} \qquad (1\text{-}30)$$

which needs further interpretation. If the attempted carry to the left from the leading position is not discarded but added, instead, into the least significant place, the result will be 00,672, which is correct. Again, in the third example (1-28), use of complements on $10^5 - 1$ would give

$$\begin{array}{r} 97{,}567 \\ 96{,}681 \\ 1|\overline{94{,}248} \end{array} \qquad (1\text{-}31)$$

Addition of the carry digit into the least significant place gives 94,249, which is the correct representation of $-5{,}750$. This process is called "end-around carry," and is always required when negatives are represented by 9's complements.

The above examples are given in detail because of the familiarity of the decimal system. An exactly similar situation occurs when the binary system is employed. Thus, if $|x| < 1$ in all cases and if negative x is represented by $2 + x$, then $5/16$ (or 0.0101) may be subtracted from $7/16$ (or 0.0111) by using the 2's complement of 0.0101 for $-5/16$, that is, 1.1011, and adding as follows:

$$\begin{array}{r} 0.0111 \\ 1.1011 \\ 1|\overline{0.0010} \end{array} \tag{1-32}$$

where the carry into the nonexistent 2^1 place is discarded. Using complements on $2 - 2^{-4}$ this becomes

$$\begin{array}{r} 0.0111 \\ 1.1010 \\ 1|\overline{0.0001} \\ \longrightarrow 1 \\ \overline{0.0010} \end{array} \tag{1-33}$$

with the end-around carry as indicated.

Nothing has yet been said about the structure of the arithmetic unit; a full discussion will be deferred to a later chapter. However, as far as the adder is concerned, it can be appreciated that a wide variety of structures is possible, depending upon the physical form in which information is handled in the machine, upon the temporal order in which the addends are presented to the adding circuits, and upon the immediate disposition of the sum.

Information can be presented to the adder (1) as temporal trains of pulses transmitted along a single conductor, (2) as pulses all occurring simultaneously on a number of wires, one for each bit, (3) as static voltage levels of a number of wires, or (4) as combinations of forms 1 and 2, as in most computers that use coded-decimal representation of numbers. Ordinarily the augend and the addend are extracted from the memory in time sequence; so some device for remembering the augend until the addend arrives must be included. Finally, it may be desired to return the sum immediately to the memory or it may be desired to retain it close to the actual adding circuits so that other numbers may be added to it. A device which can hold the augend until the addend has arrived and until the addition has been performed and which then holds the sum is usually called an "accumulator." If only single additions were to be performed this would suffice; but, as will shortly be shown, the processes of multiplication and division require further equipment to hold multipliers and divisors while they are in progress.

One other question should be taken up before proceeding to discuss multiplication and division. This is the treatment of the carries in the addition process. This is simple in serial binary adders, where the pulse trains representing the augend and addend are presented to the adding circuits simultaneously. Such adders begin by forming the sum of the lowest-order digits of two components. The carry, if any, is remembered for one pulse time; then it is added to the sum of the pair of digits in the next-higher position. In this process, the time required by the adder to form a sum is merely the time required for the sequence of pulses representing a number to enter the adder.

In parallel adders, that is, adders in which a number is presented as a set of pulses arriving simultaneously on a set of wires or as a simultaneous set of voltages of these wires, the case is quite different; a variety of forms is possible.

If numbers are presented as the static voltage levels of a set of wires, the adder may be so constructed that the sets of voltages representing two numbers are presented to it simultaneously, and the sum is completely formed, including the addition of all carries, by the time steady state is reached. This will be called a "static" adder.

On the other hand, if numbers are presented to the adder as sets of pulses arriving simultaneously on a set of parallel wires, a counter (or device that keeps count) may be provided for each digit position, and the augend and addend may be presented to the counter one after another. There must also exist devices that can hold any carry from one position to the next. After the two components have been transmitted to the counters, it is necessary to add in the carries, then the carries resulting from this addition, and so on, until all the devices used to hold carries have been returned to their zero state.[1] This is a rather extreme form of "dynamic" adder. Fortunately, it is possible to devise circuits, at least in the binary case, that are able to recognize situations in which carries will be propagated and to effect the results of propagation automatically.

1-4. Multiplication. Consider the multiplication process first in the decimal system. Form, for example, the product 473 × 674:

$$\begin{array}{r} 473 \\ 674 \\ \hline 1892 \\ 3311 \\ 2838 \\ \hline 318802 \end{array} \qquad (1\text{-}34)$$

It is necessary to obtain in some way multiples of the multiplicand by the digits of the multiplier, which then must be shifted left (i.e., multiplied by powers of 10) and added. On the other hand, multiplication in the binary system is very simple indeed, as the following example shows:

[1] Burks, Goldstine, and von Neumann, *op. cit.*, p. 13.

$$\begin{array}{r} 1101 \\ \underline{1011} \\ 1101 \\ 1101 \\ \underline{1101} \\ 10001111 \end{array} \tag{1-35}$$

At each state, the shifted multiplier is added or not added depending upon whether the bit in the multiplier is a 1 or a 0.

Before going on to a more detailed treatment of multiplication, two points of great importance for the physical structure of the arithmetic unit will be brought out. First, it is clear that the multiplicand must be readily available at all times in the course of the process. The multiplier, however, may be thrown away digit by digit, for, after the partial product corresponding to the least-significant digit of the multiplier has been formed, that digit is of no further use and may well be discarded. Indeed, it is simpler thus to discard the digits of the multiplier than not, for, if they are discarded, the register that must be provided to hold the multiplier can be so arranged that only the least-significant digit is examined and the contents of the register are shifted right one place after each partial product has been formed. Thus binary multiplication can be performed if only an accumulator and two registers, one of them with the shifting property, are available.[1] Decimal multiplication requires, as well, the equipment needed to form the products of the multiplicand by the digits of the multiplier. Of course, it would be quite possible in a binary machine to examine successively the bits of the multiplier and to call or not call the multiplicand from the high-speed memory according as the bit examined were a 1 or a 0, but this would result in a ridiculously long time to complete the process.

A second point worth noting is that a product contains more digits than either factor; in fact, if one factor contains m places and the other n,

$$p = \sum_{i=0}^{i=n-1} b^i p_i \qquad q = \sum_{i=0}^{i=m-1} b^i q_i \tag{1-36}$$

then the product contains either $m + n - 1$ or $m + n$ places. This is readily seen if the leading terms of the product are written out:

$$pq = p_{n-1}q_{m-1}b^{m+n-2} + (p_{n-1}q_{m-2} + p_{n-2}q_{m-1})b^{m+n-3} + \cdots \tag{1-37}$$

so the product contains not less than $m + n - 1$ places. Since

$$pq \leq (b^m - 1)(b^n - 1) < b^{m+n} \tag{1-38}$$

[1] *Ibid.*

it is clear that the product contains less than $m + n + 1$ places. If, therefore, it is desired to leave open the possibility of holding the entire product of two n-place numbers, a facility for storing a $2n$-place number must be provided. This is usually accomplished by shifting the partial product one place to the right at every step and allowing the overflow to enter the most significant non-sign position in the shifting register used to hold the multiplier. The desirability of shifting the multiplier one place to the right after each step has already been indicated on other grounds. This provides space free to receive the overflow from the accumulator, so that, at the end of the process, the most significant n digits of the product are contained in the accumulator, and the least-significant n digits of the product are in the register that originally held the multiplier. This convenient and practical arrangement is used in some form or another in most machines.

To illustrate the remarks above, the performance of a multiplication will be described. Suppose the product 0.1101 $\times$ 0.1011 is to be formed, where the 0's in the 2^0 positions indicate that the numbers are both positive. The first step is to examine the least significant bit of the multiplier 0.1011. Since this is a 1, the multiplicand is added into the accumulator, which has previously been set in its zero state, or, in the accepted terminology, "cleared to zero." The multiplier and the contents of the accumulator are now both shifted one place to the right; the least significant bit of the multiplier is lost, and the least significant bit of the accumulator is shifted into the position immediately after the binary point in the register that holds the multiplier; so, at the end of the add-and-shift operation,

$$\begin{array}{rl} \text{Multiplicand} & 0.1101 \\ \text{Multiplier} & \underline{0.1101} \\ \text{Partial product } P_1 & 0.0110 \end{array} \tag{1-39}$$

The least significant bit of the multiplier is again sensed, and, since it is a 1, the multiplicand is added into the accumulator. The sum becomes 1.0011, a carry having been made into the sign (2^0) position. This is not a source of difficulty, however, for the next step is to shift multiplier and partial product one place to the right, which removes the 1 from the sign position. There is no possibility of a carry beyond the sign position, so that no trouble can ever occur. This point will be considered more fully in discussing the multiplication of numbers one of which may be negative. Again, the last bit of the partial product is not lost but shifted into the register containing the multiplier, and

$$\begin{array}{rl} \text{Multiplicand} & 0.1101 \\ \text{Multiplier} & \underline{0.1110} \\ \text{Partial product } P_2 & 0.1001 \end{array} \tag{1-40}$$

The least significant bit of the multiplier is now a 0, so the next operation is to shift multiplier and partial product one place to the right again to give P_3. Since the last bit of the multiplier is 1, the final result is

$$\begin{array}{rl} \text{Multiplicand} & 0.1101 \\ \text{Multiplier} & \underline{0.1111} \\ \text{Partial product } P_4 & 0.1000 \end{array} \qquad (1\text{-}41)$$

All bits of the original multiplier have now been used and discarded, and the place of the multiplier is occupied by the least significant four bits of the product (here, of course, the binary point is ignored), and the most significant four bits of the product stand in the accumulator. The process is now terminated, for the product is completely formed. A convenient method of terminating the process is to count the number of shifts that occur, to compare the contents of the counter that does this with the number of bits in the original multiplier, in this case 4, and to arrange for a termination signal when the two numbers reach agreement.

So far nothing has been said about the multiplication of negative numbers. There are two possible procedures. If representation by sign and absolute value is used, the product of the absolute values can be formed, and to it can be affixed the correct sign, as determined by comparing the signs of the factors.

On the other hand, if negatives are to be represented by complements, it is necessary to proceed otherwise. Consider a binary number system in which the sign bit is at the extreme left end of the number, followed directly by the binary point, and in which negatives are represented by a 1 in the sign place followed by the complement of the number on $2^0 = 1$. It has been pointed out above that this amounts to representing $-1 \leq x < 0$ by $2 + x$. In the following discussion, y will represent the multiplicand and x, the multiplier.

If x and y are both negative, their machine representations are $x + 2$ and $y + 2$, whose product is $xy + 2x + 2y + 4$. The 4 would in any case be lost since there exists no 2^1 place, but corrections of $-2x$ and $-2y$ would have to be made in order to recover xy. Of course y is always available, so the $-2y$ correction can always be made. The x, however, is ordinarily discarded bit by bit in the course of the multiplication and so is not available at the end of the process for use in making the necessary correction. A procedure due to Burks, Goldstine, and von Neumann avoids this difficulty.

First consider a negative multiplier, with a positive multiplicand, where the machine representation of the multiplier is $2 + x$, or $1 + 1 + x$. There is no point in treating the sign bit of the multiplier in the same way as the other bits. Hence, if the numbers are n bits in length, including the sign, only $n - 1$ steps need be performed. This would in any case be sufficient if x were positive. In the present case it amounts to forming the product

$$(1 + x)y = y + xy \tag{1-42}$$

This must be corrected to give $2 + xy$, the correct machine representation of the negative product xy. The correction is easily performed by adding $-y$, which in machine representation is $2 - y$. Thus the rule so far can be stated:

Let the multiplication process proceed until $n - 1$ shifts have been performed. If now the sign bit of x is 0, the product has been obtained; if it is 1, correct the product by subtracting y.

Now consider $y < 0$, assuming $x > 0$, since negative multipliers have already been treated. Consider the number $1 + y$, that is, the number obtained from the machine representation of the negative y by replacing the sign bit by 0. Multiplying it by x gives $xy + x$. As noted above, the obvious correction cannot be made after the multiplication has been performed, for then x is no longer available. Hence it is necessary to make the correction for each bit of x before it is lost. It turns out to be easiest to make the total correction $2 - x$ in two stages: (*a*) $1 - x - 2^{-(n-1)}$, bit by bit; (*b*) $1 + 2^{-(n-1)}$, after the partially corrected product has been formed. The form of $1 - x - 2^{-(n-1)}$, of course, is very simple:

$$x = \sum_{1}^{n-1} 2^{-j}x_j \tag{1-43}$$

$$1 - x - 2^{-(n-1)} = \sum_{1}^{n-1} 2^{-j}(1 - x_j) \tag{1-44}$$

Consider the first step in forming $(1 + y)x$. Supposing the least significant bit of x is a 1, add $1 + y$ into the accumulator. If a correction is to be made for x_{n-1}, make it now, before x_{n-1} is lost by the first right shift. Clearly the correction must be entered into the sign position of the partial product, for, after $n - 1$ shifts, that will appear in the $n - 1$ position.

Hence, before the right shift, add $1 - x_{n-1}$ to the sign position of the partial product standing in the accumulator. Then make the first right shift. After the formation of the second partial product and before the second shift, add $1 - x_{n-2}$ into the sign position of the partial product standing in the accumulator, and so on. Thus,

$$P_1 = \tfrac{1}{2}[(1 + y)x_{n-1} + (1 - x_{n-1})] \tag{1-45}$$

since the right shift is equivalent to division by 2. Similarly,

$$P_2 = \tfrac{1}{2}[P_1 + (1 + y)x_{n-2} + (1 - x_{n-2})] \tag{1-46}$$

and, in general,

$$P_k = \tfrac{1}{2}[P_{k-1} + (1 + y)x_{n-k} + (1 - x_{n-k})] \tag{1-47}$$

Provided $0 < P_{k-1} < 1$, it is clear that the sum

$$P_{k-1} + (1 + y)x_{n-k} + (1 - x_{n-k}) < 2 \tag{1-48}$$

whence, upon division by 2, it follows that $0 \leq P_k < 1$.

Since $0 \leq P_1 < 1$, it follows by induction that this inequality holds for $k = 2, 3, \ldots, n - 1$ and, therefore, that in no case does the addition of

$$(1 + y)x_{n-k} + (1 - x_{n-k}) \tag{1-49}$$

cause a carry beyond the sign (2^0) position.

Noting that

$$(1 + y)x_{n-k} + (1 - x_{n-k}) = \begin{cases} 1 & \text{if } x_{n-k} = 0 \\ 1 + y & \text{if } x_{n-k} = 1 \end{cases} \tag{1-50}$$

P_k is obtained by adding to P_{k-1} either the sign bit of y or the machine representation of y less the sign bit, according as x_{n-k} is 0 or 1, and then shifting this sum one place to the right. Since if $y > 0$, either 0 or y is added, according as $x_{n-k} = 0$ or 1, and then shifted right, so the complete verbal prescription for the formation of P_k is:

Add to P_{k-1} either the sign bit of y or the machine representation of y with the sign bit omitted, and shift the resulting sum one place to the right.

This procedure holds for $k = 1, 2, \ldots, n - 1$, provided P_0 is defined as equal to 0. If $y < 0$, it is then necessary (*a*) to add 2^{n-1} to the contents of the accumulator and (*b*) to place a 1 in the sign position, in order to obtain from P_{n-1} the correct machine representation of the product. If both x and y are negative, the uncorrected product is

$$(1 + x)(1 + y) = y + xy + (1 + x) \tag{1-51}$$

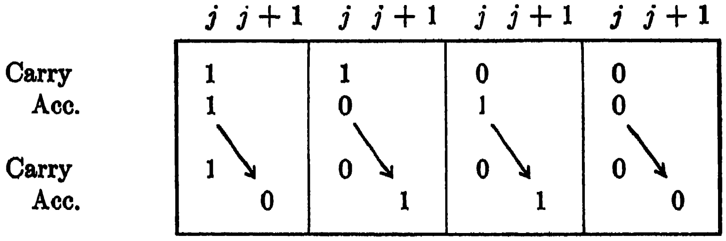
and the above process corrects properly for $1 + x$, for it uses only the non-sign bits in the machine representation of x. Of course, in this case it is still finally necessary to correct for y, that is, to add $-y$.

In the above discussion it has been tacitly assumed that each addition that occurs in building up P_k is completed before the subsequent shift. If a dynamic type of accumulator is used, this means that provision must be made to effect a complete carry at each addition, which is wasteful of time. A scheme has been devised which makes it possible to avoid this and, in fact, to provide for but a simple carry following each addition. The description of this procedure follows.[1]

[1] N. H. Taylor, The Five-digit Multiplier, *Project Whirlwind Repts.* R-134, MIT, Servomechanisms Laboratory, Dec. 3, 1948.

To introduce a simple observation, let a number A_1 stand in an accumulator, and let a number A_2 be added to it. Consider any position in the accumulator, say the jth, and its associated carry into the next-higher place. If the jth place after the addition contains a 0, the carry into the next-higher place can be 1 or 0 but, if it contains a 1, the carry can only be 0. Hence, after a single addition, no place and its associated carry together can contain more than a single 1. Thus, if a sequence of numbers is to be added into an accumulator, it is possible to add in the third to the sum of the first and second before any carries are added in at all. Clearly, it would not be permissible to add in a fourth number before adding in carries. In forming a continued sum, the systematic procedure is: (*a*) add A_1 into the accumulator, which has previously been set to zero; (*b*) add A_2 into the accumulator, and add in a single set of carries; (*c*) repeat steps *a* and *b* as often as necessary; then (*d*) add A_n into the accumulator, and carry out the complete operation of adding in carries.

The application of the above principle to the process of building up a product is obvious. It is found possible to go even further and to combine the addition of the first set of carries with the right shift. Suppose that, at some stage in the multiplication process, the multiplicand has just been added into the accumulator but that the addition of the first set of carries has not yet taken place. Clearly, if desired, both the contents of the accumulator and the carries may be shifted one place to the right before adding the carries. Fix attention upon some definite position. It may hold a 1 or a 0, and the carry into it may be a 1 or a 0. If both are 1's, the effect of shifting and adding is to leave a 1 in the carry and to shift a 0 to the right. If either combination 0 and 1 or 1 and 0 is present, the effect is to make the carry 0 and shift a 1 right. This may be summed up in a table. Note that the framework is that of the binary system with radical point immediately following the sign, so that the next-lower position after the jth is the $(j + 1)$st.



(1-52)

The circuits used to effect this shift-and-carry process will be described in the chapter on adders. Here attention is called merely to the fact that a dynamic parallel accumulator equipped with this shift-and-carry and the high-speed-carry circuits already mentioned can be caused to build up a product very rapidly.

Another scheme for minimizing the time required to build up a product in a parallel arithmetic unit relies upon the reduction of the number of additions that must be effected, since in most designs addition requires more time than shifting. This scheme depends upon the observation that[1,2]

$$2^{-(j+1)} + 2^{-(j+2)} + \cdots + 2^{-(j+m)} = 2^{-j} - 2^{-(j+m)} \qquad (1\text{-}53)$$

which permits every unbroken string of binary 1's to be replaced by a 1 and a -1, if we wish to contemplate a number system in which each position can be 1, 0, or -1. In other terms, any string of additions of the multiplicand followed by the necessary shifts would be replaced by an addition, a subtraction, and the necessary shifts. As an example, consider the conversion of the number 0.00101110011111 to the new notation. First, the string of 1's in positions 10 to 14 is replaced by -1 in position 14, 0's in positions 10 to 13, and 1 in position 9, giving 0.0010111010000(-1). Second, the 1's in positions 5, 6, and 7 are replaced by -1 in position 7, 0's in positions 5 and 6, and 1 in position 4, giving 0.001100(-1)010000(-1). Finally, the new string of 1's that has been created in positions 3 and 4 is replaced, giving as a result 0.010(-1)00(-1)010000(-1). In effecting a multiplication with this number as multiplier, two additions and three subtractions would be required as compared with the nine additions required if the multiplier is left in its conventional binary form. The reduction is not so striking if there are fewer and less lengthy strings of 1's. If, indeed, 0's and 1's alternate, no reduction occurs at all. Obviously, in the converted number, no two consecutive positions both contain non-0 characters. Reitwiesner has shown that, in general, the expected number of non-0 positions in the 1, 0, -1 notation approaches $n/3$ with increasing word length n, which is a reduction of $33\frac{1}{3}$ per cent from $n/2$, the expected number of non-0 bits in an n-place binary number.

As to the actual method of effecting a multiplication using this scheme, it is not necessary to effect the conversion explicitly; indeed, it has been shown that inspection of the bits of the binary multiplier, two at a time, suffices.[3] Let it be supposed that the arithmetic unit of a machine can

[1] Technical Progress Report, Digital Computer Laboratory, University of Illinois, Urbana, October, 1956 [work supported by AEC contract AT(11-1)-415 and ONR contracts N6ori07130 and N6ori07124].

[2] G. W. Reitwiesner, Summary Discussion of Performing Binary Multiplication with the Fewest Possible Additions, *BRL Tech. Note* 113, Ballistic Research Laboratories, Aberdeen, Md., February, 1957.

[3] See the University of Illinois Technical Progress Report for October, 1956, cited above; the table is on p. 6.

be set into two modes, positive and negative; that, in the positive mode, the multiplicand can only be subtracted from the existing partial product; that, in the negative mode, it can only be added; and that one mode persists until it is changed. Let the arithmetic unit be set in the positive mode at the beginning of the multiplication. Consider the following table, where M_{n-1} and M_n stand for the least significant bits of the multiplier; these bits are inspected, and the following operations are performed:

Mode	M_{n-1}	M_n	Operation	
+	0	0	Shift right	
	0	1	Add -icand and shift right	
	1	0	Shift right	(1-54)
	1	1	Subtract -icand, shift right, and switch to negative mode	

The next inspection is made of M_{n-2}, M_{n-1}. If the positive mode is still in effect, the operation is determined by the table (1-54); if it has been switched to the negative, the table becomes:

Mode	M_{n-2}	M_{n-1}	Operation	
−	0	0	Add -icand, shift right, and switch to positive mode	
	0	1	Shift right	(1-55)
	1	0	Subtract -icand and shift right	
	1	1	Shift right	

The inspection proceeds, two bits at a time, until all the bits of the multiplier have been inspected, and operations performed at each inspection in accordance with the two tables just given.

From the two tables just considered, it is obvious that an addition and shift or a subtraction and shift are always followed by a shift. Hence a more sophisticated procedure is always to shift right two places and to go directly from inspection of M_n, M_{n-1} to inspection of M_{n-2}, M_{n-3}, and so on, halving the number of inspections. This requires that at any stage it must be possible to add or subtract either the multiplicand or twice the multiplicand, i.e., the multiplicand shifted one place to the left. Obviously the multiplier must contain an even number of binary places. Let there be 40, let the leftmost be the sign, with the binary point between the sign and the next place, and let the negative be represented by the complement on 2. Since it must be possible to add twice the multiplicand to the contents of the accumulator at any step, it is necessary that this device have extra places, say two, to the left of its sign, and to make provision for the "spreading" of the sign automatically into these two positions whenever the multiplicand is added to the contents of the accumulator. The tables must be modified as shown below[1] (notice that a separate table must be used when one of the bits inspected is the sign bit of the multiplier):

M_{i-1}	M_i	+ mode	− mode
0	0	Shift right 2 places	Add -icand, shift right 2 places, and switch to + mode
0	1	Add -icand and shift right 2 places	Subtract 2 × -icand and shift right 2 places
1	0	Add 2 × -icand and shift right 2 places	Subtract -icand and shift right 2 places
1	1	Subtract -icand, shift right 2 places, and switch to − mode	Shift right 2 places

(1-56)

M_0	M_1	+ mode	− mode
0	0	Shift right 1 place	Add -icand and shift right 1 place
0	1	Add -icand and shift right 1 place	Add 2 × -icand and shift right 1 place
1	0	Subract 2 × -icand and shift right 1 place	Subtract -icand and shift right 1 place
1	1	Subtract -icand and shift right 1 place	Shift right 1 place

(1-57)

As an illustration, consider the formation of the product 0.10101 × 0.11011. The first inspection finds 11 and so subtracts the -icand from the contents of the accumulator, giving

$$111.01011 \tag{1-58}$$

This is shifted right two places, and, since the second inspection reveals 10, with the mode negative, it is necessary to subtract the -icand again:

$$\begin{array}{r} 111.1101011 \\ \underline{111.01011} \\ 111.0010111 \end{array} \tag{1-59}$$

This is shifted right two places, and, since the last inspection finds 01 with the mode still negative, it is necessary to add 2 × -icand, giving

[1] I owe these tables to a private communication from Mr. G. W. Reitwiesner of the Ballistic Research Laboratories, Aberdeen, Md.

$$\begin{array}{r} 111.110010111 \\ 1.0101 \\ \hline 001.000110111 \end{array} \tag{1-60}$$

and, finally, to shift right one place, giving as the product

$$0.1000110111$$

which is readily verified by performing the multiplication in the usual way.

Another approach to the speeding up of the multiplication process by calling for fewer additions depends upon having available a number of multiples of the multiplicand which can be added in as they are needed. For example, suppose m, $2m$, and $3m$ are available. Then the multiplier can be inspected two bits at a time, and m, $2m$, or $3m$ can be added, according as the bits inspected are 01, 10, 11, after which the partial product is shifted two places to the right. If the inspection is to be performed three bits at a time, $m, 2m, 3m, \ldots, 7m$ must be available. These schemes are best adapted to serial arithmetic units, since the additional adders needed for serial units are simple. For example, in the 2-bit inspection, m is always at hand, $2m$ is obtained by a simple left shift, and $3m = m + 2m$; so only a single additional adder need be used. This scheme is in fact used in the Ferranti Mercury computer; an account of the arithmetic unit of this machine is given in Chap. 13. A 4-bit-inspection scheme is used in the faster EDVAC multiplier installed in 1957.

1-5. Division. As division occurs the least frequently of the four elementary arithmetical processes, there has been considerable difference of opinion about the relative desirability of "built-in" division and division as effected by multiplying by a reciprocal. The latter procedure is:

If a process is available which computes b^{-1}, in order to compute a/b, first compute b^{-1} and then form the product ab^{-1}.

It is possible to set up a simple iterative procedure for finding the reciprocal b^{-1} of a number. This procedure is given by the formula

$$x_{n+1} = 2x_n - bx_n^2 \tag{1-61}$$

Let $b > 0$. Starting with a first approximation $x_1 > 0$, which is too small, then

$$x_2 - x_1 = x_1(1 - x_1b) > 0 \tag{1-62}$$

and

$$1 - bx_2 = 1 - 2bx_1 + b^2x_1^2 = (1 - bx_1)^2 \tag{1-63}$$

so x_2 is a better approximation but still too small. Repeated application of these inequalities shows that the sequence x_n is monotonic, increasing, and bounded, and hence $\lim x_n$ exists. Going back to the original formula, this means that

$$\lim x_{n+1} = 2 \lim x_n - b \lim x_n^2 \tag{1-64}$$

Since $\lim x_n = \lim x_{n+1}$ and $\lim x_n^2 = (\lim x_n)^2$ (1-65)

then

$$0 = \lim x_n - b(\lim x_n)^2 \tag{1-66}$$

and finally

$$\lim x_n = \frac{1}{b} \tag{1-67}$$

Under favorable conditions, the convergence of $\{x_n\}$ is very rapid, for the error $1 - bx_n$ in the nth step is the square of the error in the preceding step. Hence, if a fairly coarse table of reciprocals is provided in the computer's memory, only a few iterations are needed to attain whatever precision may be desired. It has been pointed out[1] that, with a binary machine where multiplication by 2 involves merely a left shift, an initial value x_1 of a precision of 2^{-5} can be improved by three iterations to a precision of 2^{-40}, that this can be done in six multiplication times, and that, therefore, division by multiplication by the reciprocal can be performed in seven multiplication times. A built-in division process, to be worthwhile, must be able to carry out the division process in considerably less time—fortunately, it can. In several of the earlier computers, e.g., the Mark III (Harvard) and the EDSAC (Cambridge),[2] division is programmed, but in current practice built-in division is general.

Turn now to the direct performance of the division process. It is desired to form the quotient y/x, where

$$y = \sum_{-k}^{n} 10^i y_i \tag{1-68}$$

and

$$x = \sum_{-l}^{m} 10^i x_i \tag{1-69}$$

The first step is to find the largest multiple of x by $10^{k_1}a_{k_1}$, $0 < a_{k_1} \leq 9$, that does not exceed y. Subtracting this from y, a remainder is obtained:

$$r_1 = y - 10^{k_1}a_{k_1}x = r_1 \tag{1-70}$$

The decimal digit a_{k_1} is entered in the k_1 place of the quotient. The same process is now repeated, with y replaced by r_1, and a digit a_{k_2} is obtained for the k_2 place in the quotient. If $k_2 < k_1 - 1$, the places $k_1 - 1, \ldots, k_2 + 1$ of the quotient are filled with 0's. The process may terminate if at some stage a vanishing remainder is obtained, or it may continue indefinitely, in which case a repeating group of decimal digits is eventually discovered. Clearly, the process for any base integer b is exactly parallel to the process for base 10.

[1] Burks, Goldstine, and von Neumann, *op. cit.*, p. 9.
[2] "Review of Electronic Digital Computers," pp. 50 and 80.

Now consider the problem of mechanizing this process. From the preceding paragraph several points are clear. It is necessary to have dividend and divisor immediately available at all times during the process; this requires two registers in the arithmetic unit. A third register into which the successively determined digits of the quotient can be entered is also desirable (of course it would be possible merely to reserve space for the quotient in the main memory and to transfer each digit there as it is determined). It may or may not be desirable to retain the remainder for further use. The process of determining "the greatest multiple of x of the form $10^{k_1}a_{k_1}x$ that does not exceed y" is simple to carry out working with pencil and paper, but a machine must perform successive subtractions until a negative remainder is reached in order to ascertain that the greatest multiple of x not exceeding y has, in fact, been found. If the latter process is used, the machine must record the value of a_{k_1} found and then add to the negative remainder the number $10^{k_1}x$ to obtain the correct positive remainder. For this reason this procedure is described as "restoring" division. Nonrestoring division will be described below. The arithmetical processes involved in mechanizing division are multiplication, subtraction, and shifting (multiplying by the powers of the base); in the general situation, both left and right shifts are required.

The term "restoring" has just been applied to the type of division in which at each step a positive remainder is always obtained. In the "nonrestoring" variety the partial remainder and the divisor are compared for sign. If both signs are alike, the procedure is continued until a negative remainder is reached; then the value of a_{k_1} giving the multiple of the divisor used is entered in the quotient. If the signs are opposite, the shifted divisor is repeatedly added to the remainder until a new positive remainder is obtained; then the negative of the number of additions is entered in the quotient. The quotient developed in this way is in a rather peculiar form, containing as digits $\pm 1, \pm 2, \ldots, \pm 9$ [or, in general, $\pm(b-1)$], and must be corrected so as to conform to the standard machine notation. The correction consists in subtracting the number formed by the negative digits from that formed by the positive digits, columnar positions being retained. The following example should make the process clear:

$$\begin{array}{rrrrr} & & (+2) & (-3) & (+3) \\ \cline{2-5} \multicolumn{1}{r|}{22} & 3 & 8 & 0 & 6 \\ & -4 & 4 & & \\ \cline{2-5} & & -5 & 9 & 4 \\ & & +6 & 6 & 0 \\ \cline{2-5} & & & +6 & 6 \\ & & & -6 & 6 \\ \cline{4-5} & & & 0 & 0 \end{array} \qquad q = 203 - 30 = 173 \qquad \text{(1-71)}$$

Observe that $2(b - 1)$ digits are used instead of b digits. This redundancy exists in all cases save $b = 2$, where but two digits are used in either scheme. As might be suspected, nonrestoring division is particularly simple in the binary system.

Narrowing the point of view, consider division in machines using a fixed radical point. At once, the possibility of exceeded capacity arises, just as for addition and subtraction. If this is not to be avoided by suitable scaling, it is necessary to provide an "exceeded capacity" signal which either stops the machine or apprises the control unit of overflow so that an alternative course may be taken.

Consider an example of restoring binary division:

$$\begin{array}{r|l} & \underline{0.1101} \\ 0.101 & \overline{0.1000001} \\ & \underline{0.0101} \\ & 0.0011001 \\ & \underline{0.00101} \\ & 0.0000101 \\ & \underline{0.0000101} \\ & 0.0000000 \end{array} \tag{1-72}$$

This can be rearranged as follows:

First, shift the dividend one place left and subtract the divisor; since a positive remainder results although a negative one would be obtained by a second subtraction, enter a 1 in the position following the binary point in the quotient. Repeat this process, with a similar result, and enter a second 1 in the quotient. When this process is attempted a third time, however, a negative remainder is found; so add the divisor to give a positive remainder, enter a 0 in the quotient, and shift the remainder again one place to the left. A final subtraction of the divisor yields a zero remainder, terminating the process and putting a final 1 in the quotient.

It has already been mentioned that nonrestoring division becomes particularly simple in the binary system. A simple and elegant version of this process, due to Burks, von Neumann, and Goldstine, will now be described. The fixed-point binary system is assumed, with all numbers less than 1 in magnitude and with negative x represented by $2 + x$. Let x = dividend and y = divisor; assume $|x| < |y|$, to preclude the possibility of overflow. The successive partial remainders will be designated by r_{k-1}, with $r_0 = x$. The first step is to form

$$r_1 = 2r_0 \pm y \tag{1-73}$$

subtracting if the signs of r_0 and y are the same but adding if they are different. In the former case, record $z'_1 = 1$, in the latter, $z'_1 = 0$. Continue to form in succession

$$r_k = 2r_{k-1} \pm y \tag{1-74}$$

by subtracting if r_{k-1} and y are of the same sign but adding if the signs are different, and by recording $z'_k = 1$ in the first case, $z'_k = 0$ in the second. Since $|r_0| < |y|$, by (1-73), $|r_1| < |y|$. Furthermore, by (1-74), $|r_{k-1}| < |y|$ implies that $|r_k| < |y|$. Hence the latter inequality holds for all k. Trouble might be expected to occur in the left shift of $r_{k-1} < 0$ before the subsequent addition or subtraction is performed, for the machine representation of r_{k-1} contains a 1 in the sign position, which is lost upon performance of the shift. The sign of r_{k-1} must, therefore, be sensed before the shift is performed, so that it can be determined whether the divisor is to be added or subtracted. The loss of the sign bit in the shift causes no difficulty, for the machine representation of $r_{k-1} < 0$ is $2 + r_{k-1}$, and thus

$$2(2 + r_{k-1}) = 2 + 2 + 2r_{k-1} \tag{1-75}$$

and, when the first 2 is lost because of the left shift, there remains the machine representation of $2r_{k-1}$.

In terms of the bits z'_k of the quotient, the formula for r_k may be rewritten as

$$r_k = 2r_{k-1} + (1 - 2z'_k)y \tag{1-76}$$

for, if r_{k-1} and y are of the same sign, then $z'_k = 1$ but, if they are of different sign, $z'_k = 0$. Multiplication by 2^{-k} results in

$$2^{-k}r_k = 2^{-(k-1)}r_{k-1} + (2^{-k} - 2^{-(k-1)}z'_k)y \tag{1-77}$$

Summing over all $k = 1, 2, \ldots, n$, where n is the number of steps performed in carrying out the process, and recalling that $x = r_0$,

$$2^{-n}r_n = x + \left(1 - 2^{-n} - \sum_1^n 2^{-(k-1)}z'_k\right) y \tag{1-78}$$

or

$$x = \left(-1 + \sum_1^n 2^{-(k-1)}z'_k + 2^{-n}\right) y + 2^{-n}r_n \tag{1-79}$$

Comparison with

$$x = qy + r \tag{1-80}$$

shows that the quotient and remainder are given by

$$\begin{aligned} q &= -1 + \sum_1^n 2^{-(k-1)}z'_k + 2^{-n} \\ r &= 2^{-n}r_n \end{aligned} \tag{1-81}$$

where

$$|r| < 2^{-n}\,|y| \leq 2^{-n} \tag{1-82}$$

The -1 modifies only the first term, i.e., the sign bit, of the sum, and the remaining $n - 1$ terms and the 2^{-n} give n places of the quotient. It is convenient to choose $n + 1$ equal to the number of places in the registers, including the sign.

Thus, if an accumulator, a register with the shifting property, and a third nonshifting register are available, the dividend can be placed in the accumulator and the divisor in the nonshifting register. At the first step, z_1' is put in the lowest-order position of the second shifting register, which in turn is shifted left. At the second step, z_2' is inserted in the position vacated by z_1', and so on. After n steps, corrections are made in positions 2^0 and 2^{-n}, and the resulting number in the register is the quotient.

1-6. Round-off Procedures.[1] In multiplication, the full $2n$-digit product is ordinarily replaced by one of the normal number of digits. In division the quotient is retained only to n digits, discarding the remainder. Thus, it is important to set up some criterion for deciding just how these truncations should be accomplished. In other words, it is necessary to determine a method of round-off that is, in some sense, the "best." Since it has not been found practicable to do this with full generality, recourse is usually had to a statistical argument. The assumption is made that the digits to be discarded are uniformly distributed random variables, and two criteria are adopted:

1. The variance of the approximate product or quotient from the true value should be as small as practical.
2. The approximation should be unbiased, in the sense that its mean value should be the true value.

For purposes of discussion, assume that the numbers x and y whose product or quotient is being formed are both n-place proper fractions in binary notation.

Consider two possible round-off procedures:

1. All digits beyond the nth are discarded and the nth is replaced by unity.
2. Unity is added in the $(n + 1)$st place, all carries are performed, and then all digits beyond the nth are discarded.

Scheme 2 is, of course, the familiar one used in everyday arithmetical practice.

Treating the case of division in detail, consider

$$q = \frac{x}{y} = \sum_{i=1}^{\infty} 2^{-i} q_i \qquad (1\text{-}83)$$

[1] Burks, Goldstine, and von Neumann, *op. cit.*, pp. 21–22.

where $1 > y > x > 0$, which is to be replaced by an n-place approximation. Assume that the number

$$\sum_{i=+n}^{\infty} 2^{-i}q_i \tag{1-84}$$

is a random variable, uniformly distributed over the interval from 0 to 2^{-n+1}. Then the difference between q and the n-place approximation having 1 in the nth place is a random variable uniformly distributed in the interval from -2^{-n} to 2^{-n}. The mean of this difference is obviously 0, and its variance is given by

$$\sigma^2 = 2^{n-1} \int_{-2^{-n}}^{+2^{-n}} x^2\,dx = \frac{2^{n-1}}{3}(2^{-3n} + 2^{-3n}) = \frac{2^{-2n}}{3} \tag{1-85}$$

whence

$$\sigma = \frac{2^{-n}}{\sqrt{3}} = 0.578 \times 2^{-n} \tag{1-86}$$

Now consider the second case. The number

$$\sum_{i=+n+1}^{\infty} 2^{-i}q_i \tag{1-87}$$

is treated as a random variable uniformly distributed in the interval from 0 to 2^{-n}. The rule states that the nth digit is to be left unchanged or increased by unity, i.e., that either 0 or 2^{-n} is to be added to the number

$$\sum_{i=+1}^{i=+n} 2^{-i}q_i \tag{1-88}$$

as the random variable (1-87) assumes a value in the left- or right-hand half of its interval. Hence comparison of q with the rounded value yields a random variable uniformly distributed in the intervals from -2^{-n-1} to 0 and from 0 to $+2^{-n-1}$, that is, in the interval $(-2^{-n-1}, 2^{-n-1})$. Again the mean is 0, but the variance becomes

$$\sigma^2 = 2^n \frac{2^{-3n-3} + 2^{-3n-3}}{3} = \frac{2^{-2n}}{12} \tag{1-89}$$

or

$$\sigma \doteq 0.289 \times 2^{-n} \tag{1-90}$$

Thus it has been shown that the first procedure is not markedly worse than the second and that, therefore, the decision as to which to use may as well be made on the basis of ease of mechanization, which strongly favors the first procedure because it requires no carries.

CHAPTER 2

INSTRUCTION CODES

2-1. Introduction. To say that a digital computer is "automatically sequenced" means that it is capable of performing automatically, one after another, a set of instructions presented to it. The idea goes back to Babbage, who sought to adapt the punched cards developed by Jacquard for controlling the passage of a shuttle in a loom in the production of fabrics of complicated pattern.[1] Babbage's cards were to be sensed mechanically, in sequence, and various combinations of holes were to cause the computing mechanism to perform the various desired operations. It is interesting to note that the first automatically sequenced digital computer, the Harvard Mark I, operates in much the same way. The instructions are encoded as sets of holes on successive lines of paper tape, which is advanced line by line. Each line is sensed in turn by a set of small metal pins, there being one pin for each possible hole position across the tape. When a pin encounters a hole it passes through, and a connected rod makes an electrical contact that permits the passage of a pulse of current. Thus the sets of holes in the tape are translated into sets of pulses of current, which are used to actuate the computing mechanisms. This is the pattern generally followed; the instructions to the machine are ultimately translated into sets of voltages or currents, either dynamically in the form of pulses or statically in the form of electrical states of multistable elements. Since elements having two stable states are simpler to realize than elements having many stable states and since it is simpler to recognize the presence or absence of a voltage than to perform accurate amplitude discrimination, it seems natural to use the binary system for the representation of quantities within the machine. Similarly, it seems natural to encode the instructions in the same way, as sets of binary 1's and 0's (or of "yes's" and "no's"). Thus numbers and instructions appear within the machine in the same form: as sets of voltage levels, grouped either simultaneously or in time sequence. For this reason, it is customary to refer to symbols for quantities and for instructions alike as "words." A "word" in the machine language is a sequence of bits of prescribed length, as many bits as are grouped and

[1] M. V. Wilkes, Automatic Calculating Machines, *J. Roy. Soc. Arts*, vol. C, no. 4862, Dec. 14, 1951.

transmitted together in the machine, whatever their meaning. The totality of instructions available in a computer is usually referred to as its "instruction code."

2-2. Operations upon Instructions. The fact that data and instructions are encoded in the same form is of the greatest importance, for it is at once apparent that instructions may be operated upon by circuitry of the same character as that used in processing numerical information. Thus, as the computation progresses, the machine may be caused to modify certain instructions in the code that it is following. Instructions for the performance of arithmetical operations contain numerical specifications of the positions in the memory of the quantities to be operated upon, commonly called the "addresses" of the operands. Frequently, the same cycle of operations must be performed repeatedly, but upon different quantities at each repetition. It is clear that the cycle need be entered in the code of the problem only once if instructions are included to cause the relevant addresses to be modified successively as each cycle is completed. This ability of the machine to modify a set of instructions to refer to various sets of operands makes it possible so to compress the instruction codes of many problems that they can be held, along with partial results and required data, in the limited internal memory of computers that have no relatively fast secondary or external memory. Examples will be given later in this chapter.

2-3. Types of Instructions. Existing machines differ widely in the size and complexity of their instruction codes. They differ not only in the operations made available but in the way in which the machine proceeds from instruction to instruction and in the number of instructions needed to cause the machine to perform a complete arithmetical operation. These choices, of course, determine the necessary length of the instruction words. It is clear that there must be a direct relationship between the length of the instruction words and the length of the data words if the memory of the machine is to be utilized efficiently. Examples of this will be given later in this chapter.

Obviously, there must be instructions that cause the performance of the arithmetical operations. It has been stated earlier that there is considerable latitude possible in selecting these instructions. In principle, of course, it would be possible to build up all arithmetical operations by means of sufficiently long sequences of purely logical operations, but at present this may be dismissed as impractical, in view of the length of the sequences involved. Hence all machine codes contain one or more addition instructions. Subtraction is usually treated as the addition of the negative (ordinarily the complement) of the subtrahend. Multiplication could, of course, be performed under a succession of instructions to shift and add, but, in view of the frequency of this operation, such a procedure would slow down the machine too much. Hence, it is customary to provide at least one multiplication instruction, which controls the required sequence of shifts and additions, with a counter which keeps track of the number of such operations performed and causes the termination of the process at the proper time. The relative infrequency of division as compared with multiplication made it not unreasonable to omit a direct division instruction in the earlier machines; the reciprocal of the divisor was first computed and then multiplied by the dividend. Various expedients were used to relieve the person coding the problem of the nuisance of writing out the detailed set of instructions for calculation of the reciprocal whenever a division was required. These will be discussed later in this chapter. Modern machines are provided with a division instruction which initiates the required sequence of subtractions or additions and shifts and with a counter (as in multiplication) which keeps track of the number of these operations performed and which terminates the process at the proper time. The advantage obtained through the additional circuitry is that the time required to effect the division is considerably shorter than the time needed for determining and multiplying by the reciprocal.

The four fundamental operations of arithmetic are sufficient, since all other calculations can be performed either exactly or to any desired degree of approximation by sufficiently long sequences of these operations.

There are, however, other ways of combining two words than by the operations of arithmetic. Given two binary words, for example, it is possible to generate a third word each bit of which is the logical sum or the logical product or, indeed, any Boolean function[1] of the bits in the corresponding position of the given words. Operations of this sort may be called "logical operations."

Another sort of combination is one that replaces the contents of a specified set of positions of one word by the contents of a specified set of positions of a second word. For example, see the SEAC instructions listed in Sec. 2-5. The term "transplant" has been suggested as descriptive of this type of instruction.

The instruction types mentioned so far all call for the combination of two words to form a third and so could be described as "combinatory," though this term is perhaps too general and insufficiently descriptive to be acceptable. In order to cause any of the combinatory operations to be carried out, it is necessary to specify the operation and the sources from which the words to be combined are to be taken. Thus the execution of these instructions involves the transfer of information. Transfers must be accomplished also in cases where no other operation is involved, for instance, where it is desired to take a word from one location in the

[1] See Secs. 7-3 and 4-4.

memory and place it in another location. Thus another type of instruction is distinguished.

It is frequently necessary to shift a number a specified number of places to the left or to the right, for example, when a scale factor is introduced. This amounts to the multiplication of the number in question by a power of the base of the scale of notation used, a positive power for a left shift and a negative power for a right shift. A "shift" instruction specifies the location in which the number to be shifted is to be found, the direction of the shift, and the number of places it is to be shifted.

It is a great convenience to the person preparing a code to be able to cause the machine to make certain simple routine decisions. Thus, for example, if an iterative process (e.g., the computation of $1/x$) is to be performed, the number of iterations required to attain the desired degree of approximation cannot be determined in advance. The so-called "branching instructions" are introduced to take care of such situations. A general form of such an instruction is:

Calculate the difference of two specified numbers. If this is positive, take the next instruction from a certain source; if it is negative, take the next instruction from another specified source.

In the case of an iterative calculation, for example, the computer can be caused, after each iteration, to compare the estimated error with a specified tolerance. The iteration proceeds until the error becomes less than the tolerance. At this point control is transferred to another set of instructions, and the iteration stops. Another example is the case of exceeded capacity or "overflow," where the computer attempts to add two numbers whose sum is too large to be represented in the machine's system or where an attempt is made to divide by zero. It is quite possible to arrange for an "exceeded capacity" signal that causes the machine to proceed to an alternative course of action. Instructions of this type will be called "transfer-of-control" instructions.[1]

It is also a great convenience to the coder to have available a simple means of changing the part of the instruction that specifies the location in the memory (the address) of an operand, because this makes it possible for a single instruction to be used instead of many, as, e.g., in performing a summation. The earlier machines did not explicitly incorporate special features of this type. Something of this sort could be done in the Harvard Mark III. However, it was at the University of Manchester that the idea of setting aside certain registers to hold address modifiers and of providing for addition to the address information was conceived, and the registers were given the name "*B* boxes" or "*B* registers."[2] The

[1] Also called "jump" instructions, "compare" instructions, etc.

[2] F. C. Williams, T. Kilburn, and G. C. Tootill, Universal High Speed Computers: A Small Scale Experimental Machine, *Proc. IEE*, vol. 96, part 2, p. 13, 1951.

Ferranti Mark I, based upon the Manchester University work, was the first commercial machine to provide these facilities. In this country the name "index register" is generally preferred.[1] In modern machines several of these are generally included. Certain machines do not explicitly contain index registers, yet do provide means of carrying out the basic function of successively and automatically modifying addresses.

So far nothing has been said about inserting information into the computer or removing it from the computer. Instructions needed to cause the performance of these functions will be called "input" instructions and "output" instructions.

Finally, it is necessary to start the machine, to stop it when the given sequence of instructions has been completed, and to stop it when coded checks on the operation are not satisfied.

Thus, the classes of machine instructions that have been distinguished are as follows:

1. Arithmetical
2. Logical
3. Transplant
4. Transfer-of-information
5. Transfer-of-control
6. Shift
7. Index
8. Input-output
9. Start-stop

The terminology has never been fixed, and the reader will encounter numerous variants in the literature.

2-4. Number of Addresses per Instruction. Before discussing the detailed structure of instruction words, it is necessary to consider the memory unit, that is, the part of the computer in which information is held until it is needed in the computing process and in which results of computation may be kept until they are needed in further computation or until it is desired to remove them from the machine as part of the solution of the problem. The memory unit may be static or dynamic in character. In a static unit the binary digits of each word are assigned a set of locations in space, for example, a set of positions in the phosphor of one or more cathode-ray tubes. In a dynamic unit, each word exists as a timed sequence of electrical pulses that is caused to circulate about a closed loop. In a static unit, memory locations are specified simply by numbering them in some arbitrary fashion. In a dynamic unit the memory "location" becomes essentially the time, with reference to a given

[1] D. D. McCracken, "Digital Computer Programming," chap. 8, John Wiley & Sons, Inc., New York, 1957.

reference time, at which the first pulse of the word passes a specified point in the circulation loop. If a number of loops are used, a location must be specified both in space and in time. In both types of unit, the location of each word in the memory can be given by a number, which is called the "address" of that word.

Let us consider an arithmetical instruction, for example, to carry out the addition of two numbers; the machine must be given their addresses and the operation (in this case, "add") and must ordinarily be instructed to return the result to the memory. After this, a new instruction must be obtained before further operations can be performed. If instructions are stored at arbitrary locations in the memory, to complete one operation and to obtain the instruction specifying the next operation require the specification of one operation and four addresses.

Several possibilities are now open. The most obvious is to give all this information in a single instruction word. This is a perfectly feasible scheme and is used in some existing machines. It has the advantage of complete flexibility, the coder being offered the option of locating his instructions at quite arbitrary addresses in the memory.

An example of a code of this type ("four-address" code) is that used in the EDVAC.[1] This binary machine has a capacity of 1,024 (2^{10}) words in its high-speed memory and obeys 11 instructions. The instruction requires, therefore, 44 bits for its specification. These are labeled c_1 to c_{44}, in inverse order of their sequence; the segments c_1 to c_{10}, c_{11} to c_{20}, c_{21} to c_{30}, and c_{31} to c_{40} give the four addresses, and the four bits c_{41} to c_{44} specify the operation to be performed. Four-address codes are used also in RAYDAC and SEAC.

However, even for quite modest memory capacities, four-address instructions tend to be quite lengthy. In the example cited a word length of 44 bits is not unreasonable, but, if the capacity of the memory were to be doubled and the number of available instructions raised to between 32 and 64, a 50-bit word length would result. To meet this situation, it would be necessary either to use a word length of 50 bits or a word length of 25 bits with two words used to specify each instruction. Any length between 25 and 50 would still require the use of two words per instruction and would be wasteful of memory capacity.

The four-address code has the advantage of complete flexibility with respect to the location of instructions in the high-speed memory. It is clear, however, that the simple form used as an example requires modification if the accumulator is to be used efficiently. For, if a sum of more than two quantities is to be found, it is certainly inefficient to add the first two and return their sum to the memory, then to add this sum to the

[1] "The EDVAC: A Preliminary Report on Logic and Design," p. 9, Moore School of Electrical Engineering, University of Pennsylvania, Feb. 16, 1948.

third and return the new sum to the memory, and so on. Since, in any case, some form of accumulator is required, efficiency demands that arithmetical instructions be provided that leave results in the accumulator and that can use the accumulator as a possible source from which to obtain an operand. An elegant method of accomplishing this is to assign addresses to the accumulator and to the other registers of the arithmetic unit. These addresses and the addresses in the internal memory form a single integrated system.

The number of addresses that must be specified in each instruction can be reduced by a very simple expedient: the successive instructions are stored in successively numbered addresses, and the control unit is designed so as to consult them in succession. If this expedient is used, a number of possibilities are open. Perhaps the most obvious way to construct an arithmetical instruction then is to specify the operation, the addresses of the operands, and the address in which the result is to be stored. In this case a "three-address" code results. The remarks on the efficient utilization of the accumulator made in the last paragraph above apply here also. In the SEAC both three-address and four-address modes of operation are available at the pleasure of the person writing the problem code.

Two-address codes are also possible, though they have not proved popular with designers. The earliest example is in the ACE computer in the National Physical Laboratories in England. The nomenclature used by the designers of this machine is quite different from that used in this chapter, which reflects current American usage. The ACE instructions specify a "source" for an operand, a "destination," and a "source" of the next instruction. The "destination" is not the address of a location in the high-speed memory but the coded specification of any one of several registers within the control and arithmetic circuitry; so it actually determines what operation is to be performed. Each arithmetical instruction, then, specifies the address of an operand, an operation, and the address of the next instruction. Thus the carrying out of each arithmetical operation requires the use of two instructions, and a third must be used to specify the memory location to which the result is to be delivered. For example, in an addition, the first instruction clears the accumulator to zero and transmits the augend to it; the second instruction specifies the transmission of the addend to the accumulator and the formation of the sum. If more than two operands are to be added, they are transferred successively to the accumulator. When the desired sum has been formed, a final instruction is needed to transfer it to the high-speed memory.

It is also possible to specify the addresses of the operands; the instructions are located at consecutively numbered addresses and are executed consecutively. A sophisticated and elegant two-address code of this character is used in the Sperry-Rand Corporation's UNIVAC Scientific 1103 and 1103A computers.[1] Sufficient detail will be given later on in this chapter so that the power and flexibility of this code may be appreciated.

Finally, the one-address code is the most widely used type. A recent survey of digital-computing machinery revealed that, of 103 systems investigated, 46 used single-address codes.[2] Here the instructions are stored in successively consulted addresses. Each arithmetical instruction specifies only the address of one operand and the operation to be performed. The sum (product, quotient) of two quantities requires three instructions, two to transfer the operands and initiate the operation, one to transfer the result to memory. Here an accumulator is used efficiently in forming the sum of more than two numbers. Each instruction in the one-address scheme is, of course, fairly short; hence it is quite feasible to state two or more instructions in each instruction word. For example, if the high-speed memory contains 1,024 locations and if less than 32 different operations are to be provided, each complete instruction requires a minimum of 15 bits. Hence, given a word length of 30 bits, it is possible to state two instructions in each instruction word. This scheme (using a word length of 40 bits) is used in the computer developed at the Institute for Advanced Study (IAS) and in machines based upon it. This type of code was first described by Burks, von Neumann, and Goldstine. Various modifications and adaptions are used in a variety of computers, for example, UNIVAC, Whirlwind I, and the IBM 701 and 704.

Thus far, the structure of the arithmetical instructions only has been discussed, since these instructions furnish the most easily understood illustrations of the differences among the various types of codes. Next, several specific computer codes will be described in detail, considering also the nature of the nonarithmetical instructions.

2-5. Example of a Four-address Code, SEAC. The SEAC is a serial binary computer with word length of 45 bits.[3] The conventional rep-

α	β	γ	δ		
10	10	10	10	4	1

FIG. 2-1. Four-address SEAC instruction.

resentation of an instruction word is given in Fig. 2-1. The 10-bit sections α, β, γ, and δ are addresses, the next four bits specify the operation,

[1] An account of the logic of this machine is contained in *J. ACM*, September, 1953.

[2] M. H. Weik, A Second Survey of Domestic Electronic Digital Computing Systems, *BRL Rept.* 1010, Aberdeen, Md., June, 1957. See table, p. 377.

[3] S. Greenwald, R. C. Haueter, and S. N. Alexander, SEAC, *Proc. IRE*, vol. 41, no. 10, pp. 1300–1313, October, 1953.

and the final single bit indicates whether or not the computer is to halt after performing the instruction. Actually, this last bit in Fig. 2-1 is the first of the pulse train to arrive.

In the arithmetical instructions, α and β are the addresses of the operands, γ the address at which the result is to be stored, and δ the address from which the next instruction is to be obtained. When convenient, the well-established convention that (α) means the contents of address α will be followed.

The arithmetical instructions of SEAC are:

1. Add
2. Subtract
3. Multiply, major, unrounded (retaining the most significant part of the product and performing no round-off)
4. Multiply, major, rounded (retaining the most significant part of the product, after rounding off)
5. Multiply, minor (retaining only the least significant part of the product)
6. Divide

The logical instruction is logical multiplication:

Logical "and" [form the bit-by-bit product of (α) and (β), and put the result in γ]

The transfer-of-information instructions are simple:

1. File (write the contents of a special counter register into address α)
2. Clear (replace the word at address γ by zero)

The transfer-of-control instructions will now be stated in somewhat greater detail:

1. Compare, algebraic [if $(\alpha) \geq (\beta)$, take the next instruction from δ; if $(\alpha) < (\beta)$, take the next instruction from γ]
2. Compare, absolute [like algebraic comparison, except that magnitudes are compared: if $|(\alpha)| \geq |(\beta)|$, take the next instruction from δ; if $|(\alpha)| < |(\beta)|$, take the next instruction from γ]

The transplant and index instructions are as follows:

Transplant [in any position in which a 1 occurs in (β), replace the corresponding bit of (γ) by the corresponding bit of (α); in any position in which a 0 occurs in (β), leave the corresponding bit of (γ) unchanged]. Thus (γ) can be modified at will. For example, suppose that (γ) is an instruction and that it is desired to replace the first address in (γ) by another. The desired first address is placed in α, and the first 10 bits of (β) are made 1's, the rest 0's; then a transplant accomplishes the stated purpose.

Index (if the address number α is greater than or equal to the address number β, then put the address number β into a special "base-number-counter register," and take the next instruction from γ). In subsequent instructions, all references to odd-numbered addresses will be interpreted relative to this constant number β. If the address number α is less than β, then the counter register is cleared to 0 and the next instruction is taken from γ.

Finally, the input-output instructions are:

1. Read in (if β is an odd number, read in one word from the input-output mechanism designated by α and write the data at address γ; if β is even, read in eight consecutive words from mechanism α and write the data in eight consecutive locations beginning with address γ)
2. Print out [if β is odd, record (γ) on input-output mechanism α; if β is even, record eight words beginning with (γ) on mechanism α]
3. Reverse (reverse through one word if β is odd, or through eight words if β is even, on input-output mechanism α)

2-6. Example of Three-address Code, SEAC. SEAC can be operated in both three- and four-address modes. In the three-address mode, the instructions are composed as in Fig. 2-2. The 12-bit sections are

12	12	12	*a*	*b*	*c*	*d*	4	1

FIG. 2-2. Three-address SEAC instruction.

addresses, exactly as in the four-address instruction. The final single bit is again a "halt." The 4-bit section preceding the final single bit designates the operation as before. The new feature is the 4-bit section *abcd*, called the "floating-address" section. The control unit of the SEAC in the three-address mode normally executes instructions from consecutively numbered addresses. For this purpose, the control contains a counter which advances by 1 every time an instruction is executed; this is called a "counter 0." A second counter—counter 1—is also provided. Bit "*d*" in the floating-address section designates the counter from which the address of the next instruction is to be taken. At any point in the code, control can be transferred to counter 1 by calling for a comparison (or "jump") and inserting 1 in position *d*. The address γ is inserted in counter 1, which then counts instructions and retains control as long as the instructions contain a 1 in position *d*.

When, finally, an instruction containing a 0 in position d is encountered, control is transferred back to counter 0, in which the count has remained unaltered while counter 1 was in control. This feature facilitates the coding of subroutines, a topic which will be explained later in this chapter.

The three bits a, b, and c designate whether addresses α, β, and γ, respectively, are to be interpreted as absolute or relative. An "absolute address" is just a number specifying a definite location in storage. A "relative address" identifies a location by specifying its displacement relative to the location from which the instruction word itself was obtained. Suppose, for example, that instruction (m) is being executed and that the first address α is to be interpreted as relative (i.e., the bit in a is 1); then the first word selected is $(m + \alpha)$. This amounts to the incorporation of index registers.

As to the actual instructions provided in the three-address SEAC code, it need only be remarked that, except for the arrangement that normally gives the next instruction (unless a comparison takes place), the instructions are the same as those in the four-address code.

In general, three-address codes have not found much favor with computer designers. However, a code of this type is used in a very-high-performance machine, the NORC, built by IBM for the Bureau of Ordnance of the Navy Department.

2-7. Example of a Two-address Code. The UNIVAC Scientific 1103A, produced by the Sperry-Rand Corporation, is the most notable machine to use a two-address code. This is a parallel binary machine. The word length is 36 bits. The internal memory includes a magnetic drum capable of holding 16,384 words and one to three high-speed magnetic-core units holding 4,096 words each. Auxiliary memory is provided by magnetic-tape units, of which up to ten can be included in the system. The basic number system is fixed-point, but floating-point operation can be provided as an optional feature.

The instruction words are divided as follows: 6 bits specify the operation, and two 15-bit sections specify addresses. All locations in both the magnetic-core units and the drum are individually addressable.

The registers of the arithmetic unit are as follows:

1. X, the 36-bit exchange register, is used both in effecting the internal transfer of information between parts of the system and as an arithmetic register to hold the addend, subtrahend, multiplicand, and divisor.

2. Q, a 36-bit shifting register, is used both as a memory register to hold a single word and in arithmetical and logical operations to hold the multiplier, quotient, and logical multiplier.

3. A, a 72-bit accumulator, is used both as a memory register and as an arithmetic register to hold the sum, difference, product, and quotient

with remainder. The left-hand or highest-order 36 positions are called A_L and the remaining 36, A_R. If the word (u) at memory address u is read into A_R, all positions of A_L can be made to hold 0's, or all can be made to hold the leftmost bit of (u); the contents of A in these two circumstances are referred to as $S(u)$ and $D(u)$, respectively. A can shift left circularly, so that the leftmost bit is shifted into the rightmost position. Both A_R and A_L can transmit to X. A and Q are both individually addressable.

Input and output are effected by a variety of devices, of which some are listed here: punched-tape reader, tape punch, punched-card reader and punch, oscilloscope display, line printer, and magnetic-tape units. These communicate with the internal memory via two registers *IOA* and *IOB*, which in turn transmit information to and receive information from the exchange register X.

The code of instructions for the 1103A will not be given in all detail, but attention will be focused on the more interesting features. In addition to the usual arithmetic instructions there is a "multiply-add" *MAuv*, which multiplies (u) by (v) and adds the product to (A).

The logical operations all involve the bit-by-bit logical product. First, one of the factors of this is put into Q, and then the logical product $L(Q)(u)$ of this word and of the contents of address u is formed. The instructions involving this are:

1. *QTuv* (Q-controlled transmit), which forms $L(Q)(u)$ in A and then transfers (A_R) into address v.
2. *QAuv* (Q-controlled add), which adds $L(Q)(u)$ to (A) and then transfers (A_R) into address v.
3. *QSuv* (Q-controlled substitute), which forms in A the sum of $L(Q)(u)$ and $L(Q)'(v)$, where $(Q)'$ is the 1's complement of (Q), and then transfers (A_R) into address v. The sum can be considered to be a bit-by-bit logical sum. This is in effect a "transplant" instruction, transplanting into (v) those bits of (u) that correspond to 1's in (Q).

In addition to *QSuv*, other transplant instructions are available:

1. *TUuv* (transmit U address), which replaces the bits in places 7 to 21 (counted from the left) of (v) with the bits in the corresponding positions of (u)
2. *TVuv* (transmit V address), which similarly replaces the rightmost 15 bits of (v) with the corresponding bits of (u)

The transfer-of-information instructions are straightforward, permitting any (u) to be replaced with (v), $(v)'$, or $|(v)|$. Of course the arithmetical and logical instructions in many cases involve also transfer of the results to a new address.

The transfer-of-control, or jump, instructions are varied and flexible, including both conditional and unconditional transfers, both of the "one-way" type, in which it is decided either to continue with the sequence or to transfer control to an address out of the sequence, and of the "two-way" type, in which it is decided to transfer from the sequence to one of two addresses out of the sequence. An instruction of great utility, since it provides for the return of control to the main sequence of instructions, is *RJuv* (return jump), which adds unity to the address of the current instruction, replaces with this address the rightmost 15 bits of (u), and causes (v) to be taken as the next instruction.

Only left shifting is available in the 1103A. As noted above, however, both A and Q can shift, and the shifting in A is circular. It is worthwhile to note that one of the shifting instructions is designed to find scale factors automatically. Others allow an arbitrary word to be split into two sections of arbitrary length; this is, e.g., accomplished by putting the word into the rightmost 36 positions of A, shifting left, and then transferring (A_L) to the desired address. After this, the remainder can of course be shifted into A_L and transferred as desired.

Although the 1103A contains nothing designated as an index register, a powerful instruction of the index type is included in the code. This is *RPjnw* ("repeat"). In this, the u-address bits are split into two segments, one of 2 bits for j and one of 13 bits for n, and w is an ordinary 15-bit address. *RPjnw* instructs the machine to repeat the next following instruction n times with the following modification of the u and v address:

1. If $j = 0$, both are unmodified.
2. If $j = 1$, u is unchanged, but 1 is added to v after each execution.
3. If $j = 2$, v is unchanged, but 1 is added to u after each execution.
4. If $j = 3$, 1 is added both to u and to v after each execution.

After the n repetitions, the next instruction is to be taken from address w. If the instruction to be repeated is a stop or a transfer-of-control instruction certain modifications are made, but these will not be considered here.

Finally, the 1103A code contains stop and input-output instructions, but there is no need to consider them in detail.

2-8. Examples of One-address Codes. The one-address code is by far the most commonly used type; the following is a partial list of the machines in which it is used: Whirlwind I, UNIVAC I, IAS computer, EDSAC, Ferranti Mark I, IBM 701, and IBM 704. The first machine in which it was incorporated was the computer built at the Institute for Advanced Study in Princeton, and a detailed exposition of the proposed code was given by Burks, von Neumann, and Goldstine in 1946.[1] The code presently used differs but little from their proposed code. Sufficient detail will be given to illustrate the main features of this code, and this will be followed by some account of the UNIVAC I, Whirlwind I, and 704 codes, all of which possess individual features.

It has been pointed out earlier that the arithmetic unit proposed by von Neumann incorporates an accumulator, a second shifting register, which may be called the "arithmetic register," a register to which words are transferred from the memory, and an adder. The adder forms the sum of the number in the accumulator and the number in the memory register (or its negative), and returns the sum to the accumulator. This being understood, the code is easy to describe. The first four instructions call for clearing the accumulator and adding into it either $\pm x$ or $\pm |x|$, where x is the quantity stored at the address given in the instruction. The second four instructions call for adding $\pm x$ or $\pm |x|$ to the existing contents of the accumulator. Multiplication is accomplished by two instructions: the first transfers the multiplier from the memory to the arithmetic register; the second causes the multiplicand, which is transferred by this instruction to the memory register, to be multiplied by the multiplier. As described in Chap. 1, the accumulator holds the high-order bits of the product and the arithmetic register holds the low-order bits. For division, the first instruction puts the dividend into the accumulator, and the second transfers the divisor from the memory to the memory register, causes the quotient to be formed in the arithmetic register, and causes the remainder to be left in the accumulator. These instructions, together with shift left and shift right, comprise the arithmetical instructions. Of course, to complete the arithmetic there must be an instruction that transfers the contents of the accumulator back into the memory, and there must also be orders calling for transfer of the contents of the arithmetic register to the accumulator, and vice versa. Conditional and unconditional transfers of control are provided, and also a transplant instruction, which replaces the address part (10 bits) of an instruction by another 10-bit number. A stop instruction, instructions for loading the machine's memory preparatory to computation, and instructions for removing the contents of the memory after computation are also included. As the word length used is 40 bits, two instructions of 20 bits each are given in each instruction word. This same scheme of two instructions per word is used in UNIVAC I and in the IBM 701. It is evidently mandatory if efficient use of the memory is to be made unless the basic word length is very short, as in Whirlwind I, or unless the mem-

[1] A. W. Burks, H. H. Goldstine, and J. von Neumann, "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," Institute for Advanced Study, Princeton, N. J., June 28, 1946.

ory is very large and there are features like index registers, as in the IBM 704.

The Whirlwind I code is very much the same as the Princeton code in its logical and arithmetical instructions but is quite elaborate in its input-output instructions, since provision is made for the acceptance of data from several sources and for several types of output. The word length is 16 bits; an instruction word consists of an 11-bit address and a 5-bit specification of the operation. In all, 31 instructions are provided.

The arithmetical instructions contain some features not as yet discussed. A "special" add order permits addition with preservation of overflow. A mutual transfer between the accumulator and an arbitrary address in the memory is provided. There is also a shift in the contents of the accumulator in which bits emerging from the high-order end are shifted into the low-order end of a second register.

The input-output instructions are very flexible. One instruction suffices to select the input-output unit from among the following, which are supplied: three cathode-ray oscilloscope displays, two printers, two punches (for punching paper tape), one photoelectric punched-tape reader, one mechanical punched-tape reader, and six magnetic-tape units. Once the appropriate unit has been selected, other instructions must be used to indicate the direction in which information is to be transferred and the number of words to be transferred. The instructions called "block transfer in" and "block transfer out" are used for transfers between the magnetic-tape or punched-tape units and the high-speed memory of blocks of words of lengths specified in the instruction. A "read" instruction causes one word from the input unit selected to be placed directly in the accumulator but not in the memory. A "record/display" instruction causes the word in the accumulator to be transferred to any of the output units.

The IBM 704 is a high-performance machine with a large high-speed-memory capacity; it is possible to incorporate in the system up to eight magnetic-core units holding 4,096 words each. In addition, magnetic drums and tapes provide a large auxiliary memory. The machine is of parallel binary type, and can operate in either the fixed-point or floating-point mode. In the fixed-point mode, number words consist of a sign and 35 bits arranged in decreasing order of significance from left to right, with the sign at the extreme left and the decimal point immediately following it. In the floating-point mode, the sign comes first and is followed by an 8-bit characteristic (exponent $+$ 128, so that exponents can range from -128 to $+127$) and then by a 26-bit fraction. Instruction words contain a single one-address instruction. There are two types of instruction, A and B. In type A, there is a 3-bit prefix, indicating type A, followed by a 15-bit decrement, a 3-bit tag (indicating one of three index registers), and finally a 15-bit address. In type B, the first 11 bits specify the operation (the second and third bits must be 0's to indicate type B), the next six are not used, the next three are again tags, and the last 15 give the address.

The 704 code is extensive and flexible, containing over 80 instructions. All the usual instructions of the kinds considered in connection with the machines discussed above are of type B, and the type A instructions are essentially transfers of control involving the use of the index registers as counters.

Most type B instructions are "indexable." This means that, when any such instruction is executed, the address actually referred to is the address given decreased by the contents of the index register specified by the tag bits; if all tag bits are made 0's, then naturally the address is not modified. The address part of the instruction word itself, as it stands in the memory, is in no case changed.

A specified index register can be loaded either with the address part of an arbitrary instruction word in the memory, with the address part of a word previously placed in the accumulator, with the decrement part of an arbitrary word whether in the memory or in the accumulator, or with the 2's complement of an arbitrary address (this is needed if an address is to be modified by addition). Conversely, the contents of a specified index register can replace the decrement part of a word in the memory or in the accumulator.

A variety of modes of counting and of conditional transfer of control are available also. One type A instruction causes the contents of a specified index register to be increased by the decrement and causes control to be unconditionally transferred to the instruction held in the specified address. The other type A instructions are essentially conditional transfers of control. For example, one instruction causes the number in a specified index register to be compared with the decrement: if the former is the larger of the two, control is transferred to the instruction held in the specified address and the number in the index register is decreased by the decrement; if the number in the index register is less than or equal to the decrement, control passes to the next instruction in the normal sequence. Several other instructions of this general character are provided, but the one just cited will suffice as an illustration.

The UNIVAC I is a serial alpha-numeric machine; the decimal digits are encoded in the excess-three system mentioned in Chap. 1, and the alphabetic characters are represented by an extension of this system. The word length is 12 characters. Each instruction word consists of two instructions of six digits each; the two most significant characters specify the operation and the least significant three, the address. The third digit is not used.

Some details of the structure of the machine must be given as an aid to understanding the instruction code. Four one-word registers, named A, X, L, and F, are provided for the following purposes:

1. Register A for
   a. One-word transfers
   b. Holding the addend (minuend) in addition (subtraction)
   c. Retaining a partial or complete sum
   d. Holding the more significant half of a 22-digit product or a rounded 11-digit product after multiplication
   e. Holding the dividend at the start of a division
   f. Holding the rounded quotient after division
   g. Holding a quantity to be shifted left or right, and performing the shift
   h. Assembling extracted quantities
   i. Holding one component of a comparison
2. Register X for
   a. One-word transfers
   b. Holding the augend (subtrahend) in addition (subtraction)
   c. Holding the multiplier during multiplication
   d. Holding the less significant half of a 22-digit product after multiplication
   e. Holding the unrounded quotient after division
3. Register L for
   a. Holding the multiplicand during multiplication
   b. Holding the divisor during division
   c. Holding one component of a comparison
4. Register F for
   a. One-word transfers
   b. Holding the extractor
   c. Holding 3 times the absolute magnitude of the multiplicand during multiplication

There are also four registers for multiword transfers:

1. Register V for two-word transfers
2. Register Y for ten-word transfers
3. Register I used to assemble one block of 60 words read from an input tape for transfer into memory
4. Register O used to hold one 60-word block taken from memory for transfer to magnetic tape

It can be seen from the list above that great flexibility in transferring information is available. Without attempting to give all the transfer instructions, those affecting register A will be cited as examples:

1. Erase A and X; transfer (m) to both A and X
2. Transfer (A) to m; clear A to decimal zeros
3. Transfer (A) to m; leave (A) unaltered
4. Transfer (A) to L; clear A to decimal zeros

It should be noted that, when a word is transferred to any of the one-word registers or to memory, it automatically replaces the word already there.

Arithmetical instructions are provided for the four elementary operations. For example, to add (m) to (n), we first transfer (n) to A and X and then use the instruction: "transfer (m) to X; add (X) and (A), leaving the sum in A and not clearing X." A subtraction $(n) - (m)$ is coded by first transferring (n) to A and X, then using the instruction: "transfer $-(m)$ to X; add (A) and (X), leaving the sum in A and not clearing X." These examples should make the structure of the other arithmetical instructions clear. An overflow occurring in addition, subtraction, or division automatically transfers control to the instruction stored at address 000. Instructions are also provided for right and left shifts by any desired number of decimal places.

An "extract," or transplant, instruction and both conditional and unconditional transfer-of-control instructions are provided. There is a "skip," instructing the control to pass immediately to the next instruction, and, of course, a stop. Two "break-point" instructions are included. The first is interpreted as "skip" or "stop," according to the position of the break-point switch on the supervisory control unit. The second operates only if certain buttons in the supervisory control have been pushed, in which case the computer stops after having executed the comparison part of a conditional transfer, but before the actual transfer of control has taken place. Another switch makes it possible to make the control ignore conditional transfers or treat conditional transfers as if they were unconditional.

The input-output magnetic-tape unit of the UNIVAC I is called a Uniservo, and as many as ten of them may be used with a single computer. Information is transferred between computer and Uniservo in blocks of 60 words. The 60-word input-output registers I and O, noted above, form the buffer memory used to effect the transfers while the computation is permitted to continue simultaneously. The "read-in" instructions first select the desired Uniservo and then cause a block of 60 words to be transferred to I. A second instruction transfers these to 60 consecutive addresses in the memory beginning with a specified address, clears I, and causes a second block of 60 words to be read from the Uniservo into I. The Uniservo tape may be moved forward or backward in reading, as desired. "Read-out" is the reverse operation, but there are two separate instructions for writing the contents of O on the tape. In one case, the magnetized spots are recorded densely on the tape (100 to the inch) and may be used again only as inputs to the computer, *not* to drive the printing units; in the other, a less dense recording (20 spots to the inch) is made, and tapes so prepared may be used either as future inputs or to drive the Uniprinter. Thus the entire tape capacity may be used as a supplementary memory.[1]

Other examples of very complete and fully developed one-address codes are those of the Ferranti Mark I and the IBM 701 and 704 computers.[2,3] There is no need to consider these in detail.

This section will be concluded with a complete list of the instructions in the Whirlwind I code. The reader should remember the following designations for the registers of the arithmetic unit: AC (accumulator), AR (arithmetic register), BR (B register—this is not an index register). There is also an in-out register IOR. The letters ES refer to the electrostatic storage, which was the type of storage used when this version of the code was recorded (Sept. 1, 1952) although it was later replaced by a magnetic-core memory. For each order, the common designation (as, for example, ca x), the name, the 5-bit binary number used to designate the operation, and the time required to perform it are listed.

0. si *pqr* (Select in-out unit/stop), 00000, 40 μsec.
The letters *pqr* stand for octal digits, or groups of 3 bits. Of these, p designates the class of equipment (e.g., printers, punches, tape readers), and q and r designate, respectively, the number of the unit and the mode of operation (e.g., read, record, forward, reverse, etc.). The instruction si 0 will stop the computer; si 1 will stop the computer only if the conditional stop switch is "on"; si 600 will stop any in-out unit without stopping the computer. An "si" instruction must be followed by other instructions to specify how many words are to be read in or out and what is to be done with them.

[1] "Programming UNIVAC Fac-Tronic Systems," Instruction Manual I, Eckert-Mauchly Division of Remington-Rand, Inc., January, 1953.

[2] F. C. Williams and T. Kilburn, The University of Manchester Computing Machines, in "Review of Electronic Digital Computers," Joint AIEE-IRE Computer Conf., pp. 57–61, American Institute of Electrical Engineers, New York, February, 1952. The Ferranti machine was based upon the one described by Williams and Kilburn.

[3] W. Buchholz, The System Design of the IBM Type 701 Computer, *Proc. IRE*, vol. 41, no. 10, pp. 1262–1275, October, 1953.

1. rs–	(Stop computer), 00001, 30 μsec. Stop the computer unless a certain switch has been set to "on." The address section has no significance.
2. bi *x*	(Block transfer in), 00010. Transfer a block of *n* words from paper or magnetic tape to *ES*, where *x* is the initial address of the block in *ES* and *n* is contained in *AC*. The computer is stopped while the transfer is taking place.
3. rd–	(Read), 00011, 40 μsec. Wait, if necessary, for information to arrive at *IOR* from an in-out unit. Transfer information in *IOR* to *AC*, then clear *IOR*. If the most recent si instruction called for an oscilloscope, display a point on the selected oscilloscope at the most recent horizontal and vertical settings. The address section has no significance. (The normal method of displaying is given in "rc" below.)
4. bo *x*	(Block transfer out), 00100. Transfer a block of *n* words from *ES* to paper tape, where *x* is the initial address of the block in *ES* and where *n* is contained in *AC*.
5. rc–	(Record/display), 00101, 40 μsec. Transfer contents of *AC* via *IOR* to an in-out unit, then clear *IOR*. If the most recent si instruction called for an oscilloscope, set the vertical deflection to a value corresponding to the contents of digits 0 to 10 of *AC*, and display a point. The address section has no significance.
6. ts *x*	(Transfer to storage), 01000, 86 μsec. Transfer contents of *AC* to address *x* in *ES*.
7. td *x*	(Transfer digits), 01001, 86 μsec. Transfer the last 11 digits from *AC* to the last 11 positions at address *x*. (In instructions, the last 11 digits specify an address.)
8. ta *x*	(Transfer address), 01010, 86 μsec. Transfer the last 11 digits from *AR* to the last 11 positions at address *x*.
9. ck *x*	(Check), 01011, 48 μsec. If contents of address *x* are not identical with contents of *AC*, stop the computer and give a "check-register alarm"; otherwise proceed to next instruction.
10. ex *x*	(Exchange), 01101, 86 μsec. Exchange contents of *AC* with contents of address *x*.
11. cp *x*	(Conditional program), 01110, 30 μsec. If the number in *AC* is negative proceed as in sp; if the

number in AC is nonnegative, disregard the cp instruction, and clear AR.

12. sp x (Subprogram), 01111, 30 μsec.
Take the next instruction from address x. If the sp instruction was at address y, store $y + 1$ in the last 11 digit positions of AR.

13. ca x (Clear and add), 10000, 48 μsec.
Clear AC and BR; then put the contents of address x into AC. If necessary, add in the carry from previous sa instruction.

14. cs x (Clear and subtract), 10001, 48 μsec.
Clear AC and BR; then put the complement of the contents of address x into AC. If necessary, add in the carry from previous sa instruction.

15. ad x (Add), 10010, 48 μsec.
Add contents of address x to contents of AC, leaving result in AC.

16. su x (Subtract), 10011, 48 μsec.
Subtract contents of address x from contents of AC, leaving difference in AC.

17. cm x (Clear and add magnitude), 10100, 48 μsec.
Clear AC and BR; then put absolute value of contents of address x into AC. If necessary, add in the carry from previous sa operation.

18. sa x (Special add), 10101, 48 μsec.
Add contents of address x to contents of AC, storing result in AC and retaining any overflow for the next ca, cs, or cm. Only nonarithmetical instructions may be used between sa and the next ca, cs, or cm, for which sa is a preparation.

19. ao x (Add one), 10110, 86 μsec.
Add the number 1×2^{-15} to the contents of address x, and store the result in AC and in address x.

20. mr x (Multiply and round off), 11000, 65 μsec.
Multiply contents of address x by contents of AC, round off result to 15 numerical digits, and store in AC; clear BR.

21. mh x (Multiply and hold), 11001, 65 μsec.
Multiply contents of address x by contents of AC; retain the full product in AC and in the first 15 digit positions of BR; clear the last digit position of BR.

22. dv x (Divide), 11010, 120 μsec.
Divide the contents of AC by the contents of address x,

leaving 16 numerical digits of the quotient in *BR* and ± 0† in *AC* according to the sign of the quotient. The instruction sl 15 will then round off the quotient to 15 numerical digits and store it in *AC*. The following peculiarities should be noted: (*a*) if the magnitude of the dividend exceeds that of the divisor, an alarm is given; (*b*) if the dividend and divisor are equal, the result is sixteen 1's in *BR*; (*c*) if dividend and divisor are both zero, a zero quotient is obtained.

23. sl *n* (Shift left), 11011, 35 μsec.
Multiply the number represented by the contents of *AC* and *BR* by 2^n. Round off the result to 15 numerical digits and store it in *AC*. Disregard overflow caused by the multiplication, but not that caused by the round-off. Clear *BR*.

24. sl* *n* (Shift left without round-off), 11011, 35 μsec.
Multiply contents of *AC* and *BR* by 2^n, but neither round off nor clear *BR*.

25. sr *n* (Shift right), 11100, 35 μsec.
Multiply the number represented by the contents of *AC* and *BR* by 2^{-n}. Round off the result to 15 numerical digits and store it in *AC*. Clear *BR*.

26. sr* *n* (Shift right without round-off), 11100, 35 μsec.
Multiply as in "sr *n*," but neither round off nor clear *BR*.

27. sf *x* (Scale-factor), 11101, 97 μsec.
Multiply the number represented by the contents of *AC* and *BR* by 2 sufficiently often to make the absolute value equal to or greater than ½. Leave the final product in *AC* and *BR*. Store the number of multiplications as the last 11 digits at address *x*, leaving the first five positions undisturbed. Store the number of multiplications also in *AR*.

28. cl *n* (Cycle left), 11110, 41 μsec.
Shift the full contents of *AC* and *BR* (including the sign digit) left *n* places, and carry around into *BR* the digits shifted left out of *AC*, so that no digits are lost. Clear *BR* without round-off.

29. cl* *n* (Cycle left without clearing *BR*), 11110, 41 μsec.
Same as "cl *n*" except that *BR* is not cleared.

NOTE: In the shift and cycle orders, the whole address section of the instruction is available to specify the number of shifts, though 32 is the

† Using the 1's complement representation of negatives, $1 \cdot 11 \cdots 11$ is to be interpreted as -0.

greatest recognizable number of shifts that can be performed. Hence the following method of discriminating between, for example, "sl n" and "sl* n" is used: in the unstarred instruction, the sixth digit of the address section is always 0; in the starred instruction, it is always 1.

2-9. The Use of Subroutines. It has been pointed out that the only arithmetical operation that can be carried out simply and directly by electronic circuits is addition (this includes subtraction as the addition of the negative of the subtrahend). In all digital computers multiplication and division are effected by sequences of additions or subtractions, though they may be, and in fact usually are, initiated by a single instruction. All other operations must be programmed in terms of the elementary operations. It is a convenience to those who code problems for machine computation to have readily available the codes that cause the computer to perform certain calculations which are of frequent occurrence, as, for example, the calculation of the various elementary functions of a given argument ($\log_{10} x$, e^x, $\sin x$, etc.). The set of instructions needed to cause the performance of such a calculation is called a "subroutine." There are, in general, two different ways of making subroutines available.

If the memory capacity is large, a number of frequently used subroutines can be stored permanently. This is a feature of the Mark III calculator, where a portion of the magnetic drum upon which instructions are stored is allocated to subroutines for calculating x^{-1}, $x^{-1/2}$, $\log_{10} x$, 10^x, $\cos x$, and $\arctan x$. There is also a subroutine for performing division by multiplying the dividend by the reciprocal of the divisor. In the design of this machine, great care was taken to make the coding of problems as simple as possible. An "instructional tape-preparation unit" is provided; by means of its keyboard the problem code is transferred to a punched paper tape to be recorded later on a magnetic drum set apart as a memory for instructions only. This unit is so arranged that one need only write by means of the keyboard a single instruction specifying the address of the argument and the function to be calculated. Instructions for the transfer of control to the first instruction of the appropriate subroutine and for the return of control to the main routine at the conclusion of the performance of the subroutine are automatically recorded on the instruction tape by this single instruction.[1]

It is also possible to proceed quite differently and to store the subroutines outside the calculator. Thus a "library" of subroutines can be accumulated and can then be used as desired in the preparation of the input tape by means of which the data and instruction code of a problem

[1] Staff, Harvard Computation Laboratory, "Description of a Magnetic Drum Calculator," Harvard University Press, Cambridge, Mass., 1952. See in particular chaps. 5 and 8.

are inserted into the machine. This method becomes useful if there exists a means of automatically transferring the subroutines to the problem tape. This procedure appears to have been first developed at the University Mathematical Laboratories at Cambridge University, by M. V. Wilkes and others.[1-4]

A great deal of work has also been done in this country (much of it inspired by the pioneering efforts of Wilkes); we may mention in particular the work of C. W. Adams at the Digital Computer Laboratory at MIT. Extensive discussions of this subject have been presented at the Pittsburgh (May, 1952), Toronto (September, 1952), and Cambridge (September, 1953) meetings of the Association for Computing Machinery (ACM), and are contained in the published proceedings of those meetings.

Wilkes distinguishes two main varieties of subroutines. An "open" subroutine is a sequence of instructions that can be incorporated serially in a problem code. A "closed" subroutine is called in by a special instruction or group of instructions in the main problem code; when its mission has been accomplished, control is returned to the instruction in the main code that immediately follows the instructions that called in the subroutine. The use of closed subroutines makes the coding of iterative processes very efficient. Indeed, it is possible to use such a procedure in cases in which it is not immediately obvious that it is appropriate, and thus to effect a great saving in the number of instructions required. A brief example of this will be given in Sec. 2-10.

2-10. Examples of Coding.

The present volume is not concerned with the subtleties of the art of coding. The reader who wishes to pursue the subject is referred to the excellent works of Wilkes, Wheeler, and Gill and to the fundamental treatment by von Neumann and Goldstine.[5] Two simple examples, however, will be presented. Both will be expressed in the Whirlwind I code, which has been given in full above. For simplicity, it is assumed that at no point in the computations do the numbers grow out of the range $-1 \leq x < +1$.

The first example is extremely simple, dealing with the tabulation of a

[1] M. V. Wilkes, Programme Design for a High-speed Automatic Calculating Machine, *J. Sci. Instr.*, vol. 26, p. 217, 1949.

[2] D. J. Wheeler, Programme Organization and Initial Orders for the EDSAC, *Proc. Roy. Soc.*, vol. 202A, p. 573, 1950.

[3] M. V. Wilkes, D. J. Wheeler, and S. Gill, "The Preparation of Programs for an Electronic Digital Computer," Addison-Wesley Publishing Company, Reading, Mass., 1951.

[4] McCracken, *op. cit.*, chap. 9.

[5] J. von Neumann and H. H. Goldstine, "Planning and Coding of Problems for an Electronic Computing Instrument," Institute for Advanced Study, reports written under U.S. Army Ordnance contract W-36-034ord7481, Princeton, N.J., 1947–1948.

quadratic function

$$f(u) = au^2 + bu + c \tag{2-1}$$

for n values of u from u_0 to $u_0 + (n - 1)h$ in increments of h. For ease in computation, this is first rewritten as

$$f(u) = u(au + b) + c \tag{2-2}$$

Memory addresses are allocated for the data:

Value	*Address*
a	200
b	201
c	202
u_0	203
h	204
$-n \times 2^{-15}$	205

The addresses 300, 301, . . . , $300 + n - 1$ are assigned to the storage of the functional values to be computed, and the instructions are placed in a block of addresses beginning with 100. The instructions required to cause the computer to calculate $f(u_0)$ are:

Instruction	*Address*
ca 200	100
mr 203	101
ad 201	102
mr 203	103
ad 202	104
ts 300	105

It is now necessary to increase u_0 by h, which is accomplished by the following:

ca 203	106
ad 204	107
ts 203	108

Since $u_0 + h$ now occupies the location originally occupied by u_0, $f(u_0 + h)$ can be computed by means of the very sequence of instructions used to compute $f(u_0)$. To do this it is necessary only to transfer control back to the first instruction of the sequence. However, before doing this, one instruction must be modified, that which transfers the computed value of the function to the memory. This is done by adding 1 in the lowest-order position to the address section of "ts 300," which is stored in address 105; the instruction to accomplish this is

ao 105	109

This having been done, the transfer of control back to the first instruction of the sequence, which is located at address 100, is arranged. The sequence must be carried out n times, so that the n functional values desired will be computed; this can be arranged in a variety of ways. A simple and obvious procedure is to store $-n \times 2^{-15}$ in the memory (at address 205) and to add 1 in the lowest-order position after each functional value has been computed. Then a conditional-program instruction is inserted which causes control to be transferred back to the beginning of the sequence of instructions, provided that the number at address 205 is negative:

```
ao 205        110
cp 100        111
```

The code as written assures the computation of the n desired values of $f(u)$. After the computation of the nth value, the number at address 205 becomes nonnegative. As a consequence, "cp 100" will be ignored, and control will pass to whatever instruction is stored at address 112. If it is desired that control pass instead to the instruction at address q, it is necessary to place "sp q" in address 112.

Despite its simplicity, the above example clearly shows the great power inherent in the ability of the computer to modify its instructions. This feature, together with the availability of the conditional-transfer instruction, permitted the computation of any desired number of values of $f(u)$ using only 12 instructions.

In conclusion, a subroutine for the extraction of the square root is presented. This is fundamentally more complicated than the simple example just given. It is an iterative process in which it cannot be decided in advance just how many iterations must be made. It is, therefore, necessary so to arrange the code that the computer makes this decision automatically.

The formula from which the successive approximations are calculated is the familiar one:

$$z_{i+1} = \tfrac{1}{2}\left(z_i + \frac{a}{z_i}\right) \tag{2-3}$$

where a is the number whose square root is to be extracted and where it is assumed that $0 \leq a \leq 1 - 2^{-15}$. From Eq. (2-3) the following equations can be deduced by simple algebraic manipulation:

$$z_{i+1}^2 - a = \frac{1}{4z_i^2}(z_i^2 - a)^2 \tag{2-4}$$

$$z_{i+1} - \sqrt{a} = \frac{1}{2z_i}(\sqrt{a} - z_i)^2 \tag{2-5}$$

$$z_{i+1} - z_i = \frac{1}{2z_i}(a - z_i^2) \tag{2-6}$$

It follows that, if the first approximation z_1 exceeds the square root of a, then the successive approximations form a monotonic decreasing sequence bounded below by $\sqrt{a}$. The sequence therefore converges, and it follows at once from Eq. (2-3) that the limit is indeed the desired square root. For computational purposes it turns out to be more convenient to modify Eq. (2-3) slightly and to put it in the equivalent form

$$z_{i+1} = z_i + \tfrac{1}{2}\left(\frac{a}{z_i} - z_i\right) \tag{2-7}$$

In the actual computation of z_{i+1}, because of the division and right-shift operations, an error of 1 in the lowest-order place may be introduced.

It is convenient at this point to introduce a terminology and a notation due to von Neumann and Goldstine.[1] Designate a quantity actually produced by the machine by writing a bar over it: thus $\bar{z}_{i+1}$ is the actual, in this case the 15-bit, quantity calculated by the machine from formula (2-7). Such quantities are to be called "pseudo numbers." It must be recognized that rounded multiplication and division as performed by the machine should be conceptually distinguished from ideal multiplication and division. They will, therefore, be called "pseudo multiplication" and "pseudo division," respectively, and may be conveniently designated by the signs "$\times$" and "$\div$" usually used in elementary arithmetic.

The sequence $\{\bar{z}_i\}$ generated by the machine differs in one striking respect from $\{z_i\}$: each $\bar{z}_i$ is a 15-bit binary fraction; hence the sequence obviously cannot be indefinitely of the strictly decreasing type, for if it were it would diverge to negative infinity. Since it is evident from Eq. (2-6) that, provided $\bar{z}_1^2$ is chosen greater than $\bar{a}$, the sequence must start by decreasing, it is clear that sooner or later an integer i_0 must be reached such that

$$\bar{z}_{i_0+1} = \bar{z}_{i_0} \tag{2-8}$$

It is now necessary to find how good an approximation $\bar{z}_{i_0}$ is to the desired square root. To this end, first observe that

$$\bar{z}_{i+1} - \sqrt{\bar{a}} - r = \frac{1}{2\bar{z}_i}(\bar{z}_i - \sqrt{\bar{a}})^2 > 0 \tag{2-9}$$

where

$$|r| \leq 2^{-15} \tag{2-10}$$

since, as was noted above, an error of 1 may have been introduced in the lowest-order place in the computation of $\bar{z}_{i+1}$. Therefore,

$$\bar{z}_{i+1} > \sqrt{\bar{a}} - 2^{-15} \tag{2-11}$$

[1] J. von Neumann and H. H. Goldstine, Numerical Inverting of Matrices of High Order, *Bull. Am. Math. Soc.*, vol. 53, no. 11, pp. 1021–1099, November, 1947.

Now suppose that

$$\bar{a} < \sqrt{\bar{a}} + 2^{-14} < \bar{z}_i \leq 1 - 2^{-15} \tag{2-12}$$

and consider the formula (2-7) for the computation of $\bar{z}_{i+1}$. The term $\bar{z}_i - \bar{a} \div \bar{z}_i$ can be written

$$\bar{z}_i - \bar{a} \div \bar{z}_i = (\bar{z}_i - \sqrt{\bar{a}}) + \left(\sqrt{\bar{a}} - \frac{\bar{a}}{\bar{z}_i}\right) + \left(\frac{\bar{a}}{\bar{z}_i} - \bar{a} \div \bar{z}_i\right) \tag{2-13}$$

By Eq. (2-12), the first term on the right exceeds 2^{-14}, and the last term must exceed -2^{-15}; so their sum exceeds 2^{-15}. The second term may be written

$$\sqrt{\bar{a}}\left(1 - \frac{\sqrt{\bar{a}}}{\bar{z}_i}\right) > \sqrt{\bar{a}}\left(1 - \frac{\sqrt{\bar{a}}}{\sqrt{\bar{a}} + 2^{-14}}\right) = \frac{2^{-14}\sqrt{\bar{a}}}{\sqrt{\bar{a}} + 2^{-14}} \tag{2-14}$$

where the last term exceeds 2^{-15} if $\bar{a} \geq 2^{-14}$ and has a minimum of zero for $\bar{a} = 0$. Hence it is clear that the left-hand side of Eq. (2-13) must exceed 2^{-15}, and, since it is a 15-bit number, it must be at least equal to 2^{-14}. It follows that

$$(\bar{z}_i - \bar{a} \div \bar{z}_i) \div 2 \geq 2^{-15} \tag{2-15}$$

which shows that, if $\bar{z}_i$ satisfies Eq. (2-12), then i is less than i_0, because Eq. (2-15) implies that $\bar{z}_{i+1} < \bar{z}_i$. It is, therefore, clear that the sequence of approximations continues to decrease until an integer i is reached such that

$$\sqrt{\bar{a}} - 2^{-15} < \bar{z}_i \leq \sqrt{\bar{a}} + 2^{-14} \tag{2-16}$$

Then

$$\bar{z}_i = \sqrt{\bar{a}} + s2^{-15} \qquad -1 \leq s \leq 2 \tag{2-17}$$

from which it follows easily that

$$\frac{\bar{a}}{\bar{z}_i} = \sqrt{\bar{a}} - s2^{-15} + \frac{s^2 2^{-30}}{\sqrt{\bar{a}} + s2^{-15}} \tag{2-18}$$

whence

$$\frac{\bar{a}}{\bar{z}_i} > \sqrt{\bar{a}} - s2^{-15} \tag{2-19}$$

and, therefore,

$$\bar{a} \div \bar{z}_i \geq \sqrt{\bar{a}} - s2^{-15} + t2^{-15} \qquad t = \pm 1 \tag{2-20}$$

where t represents the error due to the rounded division. This yields

$$\bar{a} \div \bar{z}_i - \bar{z}_i \geq (t - 2s)2^{-15} \tag{2-21}$$

The quantity on the left of Eq. (2-21) may be of either sign. If it is nonnegative, then it is ensured that $\bar{z}_{i+1} \geq \bar{z}_i$, and $i = i_0$. On the other hand, if it is positive, then $\bar{z}_{i+1} < \bar{z}_i$, and it is ensured that i_0, with $\bar{z}_{i_0}$ in the range

(2-16), will shortly be reached. Hence, in any case, a finite number of iterations produces a number $\bar{z}_{i_0}$ in the range (2-16) such that $\bar{z}_{i_0+1} \geq \bar{z}_{i_0}$. This number is assumed to be the desired square root.

The argument above has established the fact that the sequence of approximations continues to decrease until a value in the range (2-16) is reached. Thus, although the integer i_0 is not known in advance, the fact that it marks the turning point at which the sequence $\{\bar{z}_i\}$ ceases to be strictly decreasing means that the code can be so arranged that the computer determines when it has gone "far enough," at which point computation ceases and the desired square root has been obtained.

The code for the square-root subroutine can now be written, using again the Whirlwind I code. Two memory addresses must be allocated for the number $\bar{a}$ and for the successive approximations. Also, if $\bar{a} = 1 - 2^{-15}$, there is no point in carrying out the computation, for $\sqrt{\bar{a}} > \bar{a}$ and $1 - 2^{-15}$ is the largest number that can be handled in the machine; in this case, the square root also must be considered to have the value $1 - 2^{-15}$. Hence $1 - 2^{-15}$ must be stored and compared with $\bar{a}$. If the two are equal, then the square root is already determined; if $\bar{a}$ is smaller, the computation can proceed. Addresses are allocated as follows:

Address	*Number*
200	$1 - 2^{-15}$
201	$\bar{a}$
202	$\bar{z}_i$ (initially unfilled)
203	(This is a temporary-storage location used from time to time as the computation progresses; it is initially unfilled)

The instructions start with those needed to compare the contents of 200 and 201:

Address	*Instruction*	*Effect*
100	ca 200	Puts $1 - 2^{-15}$ in AC
101	su 201	Leaves $1 - 2^{-15} - \bar{a}$ in AC

If the result of this subtraction is positive, the computation of the successive approximations must be called for; if it is zero, $1 - 2^{-15}$ is recorded as the desired square root. The cp x instruction will accomplish this; at the moment, however, it is not known how many instructions are to be required for the complete code, and so x cannot be specified. Proceeding,

102	cp x	If $(AC) > 0$, take the next instruction; if the sign of (AC) is negative, then transfer control to the instruction at address x

This last instruction looks rather odd, for clearly the contents of AC cannot be "negative" in the usual sense. To clear up any difficulty on this point, it is observed that subtraction is accomplished by the addition of the 1's complement; so $(1-2^{-15}) - (1-2^{-15})$ is represented as $1-2^{-15} + [2 - (1-2^{-15}) - 2^{-15}]$, which gives $1.111 \cdots 1$ and which may be regarded as "-0."

103	ca 200	Puts $1-2^{-15}$ in AC
104	ts 202	Puts $\bar{z}_1 = 1-2^{-15}$ in 202
105	ca 201	Puts $\bar{a}$ in AC
106	dv 202	106 and 107 leave $\bar{a} \div \bar{z}_1$ rounded to 15 bits in AC
107	sl 15	
108	su 202	Leaves $\bar{a} \div \bar{z}_1 - \bar{z}_1$ in AC

If this last result is negative, it is assured that $\bar{z}_2 < \bar{z}_1$; if it is nonnegative, the process should be terminated. Hence another "cp" instruction is needed but, to make it work properly, the negative contents of AC must be used.

109	ts 203	Puts (AC) in 203
110	cs 203	Puts $-(203)$ in AC
111	cp y	Go to next instruction if (AC) is nonnegative, but to instruction at as-yet-unspecified address y if (AC) is negative
112	ca 203	Puts $(203) = \bar{a} \div \bar{z}_1 - \bar{z}_1$ in AC
113	sr 1	Divides (AC) by 2
114	ad 202	Leaves $\bar{z}_2$ in AC
115	sp 104	This calls for a repetition of the program from 104 on; successive approximations will be computed and recorded in 202 until $\bar{z}_{i_0}$ has been obtained; then the next following 111 transfers control to y and prevents further iteration

So far no instruction has been given to which control is to be transferred if $\bar{a} = 1-2^{-15}$, nor any indication of what is to be done when $\bar{z}_{i_0}$ has been obtained. Now that the number of instructions in the code is known, choose $x = 116$ and add the following:

116	ca 200	These two instructions put $1-2^{-15}$ in 202
117	ts 202	

If, in the main routine, the computation of $\sqrt{\bar{a}}$ was to take place between the instructions at m and at $m + 1$, the transfer of control to the second address [as $2^{-15}(m + 1)$] can be stored in the memory at 204. Then, at address 118, which is assigned as the address y, the following instruction is added:

```
118          sp 204
```

which returns the control from the subroutine ultimately back to the main routine. The completed problem code is included as Table 2-1.

TABLE 2-1. PROGRAM FOR COMPUTATION OF $\sqrt{\bar{a}}$*

Address	Instruction	Effect
100	ca 200	$1-2^{-15} \rightarrow AC$
101	su 201	$1-2^{-15} - \bar{a} \rightarrow AC$
102	cp 116	If $(AC) \geq 0$, continue to next instruction (103) If $(AC) < 0$, jump to instruction (116)
103	ca 200	$1-2^{-15} \rightarrow AC$
104	ts 202	$\bar{z}_1 = 1-2^{-15} \rightarrow 202$
105	ca 201	$\bar{a} \rightarrow AC$
106	dv 202	$\bar{a}/\bar{z}_1$
107	sl 15	$\bar{a}/\bar{z}_1$ rounded to 15 bits $\rightarrow AC$
108	su 202	$(\bar{a}/\bar{z}_1) - \bar{z}_1 \rightarrow AC$
109	ts 203	$(AC) \rightarrow 203$
110	cs 203	$-(203) \rightarrow AC$
111	cp 118	If $(AC) \geq 0$, continue to next instruction (112) If $(AC) < 0$, jump to instruction (118)
112	ca 203	$(203) \rightarrow AC$
113	sr 1	$(AC)/2 \rightarrow AC$
114	ad 202	$\frac{1}{2}(\bar{a}/\bar{z}_1 - \bar{z}_1) + \bar{z}_1 = \bar{z}_2 \rightarrow AC$
115	sp 104	Jump to instruction (104)
116	ca 200	$1-2^{-15} \rightarrow AC$
117	ts 202	$(1-2^{-15}) \rightarrow 202$
118	sp 204	Jump to instruction (204)
	Data	
200	$1-2^{-15}$	
201	$\bar{a}$	
202	$\bar{z}_i$ (initially unfilled, ultimately $\sqrt{\bar{a}}$)	
203	Temporary storage	
204	sp $m+1$	

* No provision has been made for reading in the data and instructions or for reading out the result.

2-11. Automatic Coding.

The writing of the code of instructions for a fairly lengthy computation is a very tedious process in which it is very easy to make mistakes. Similarly, the checking of the code on the machine to assure accuracy before the computation is performed is very time-consuming; indeed, code checking has frequently been estimated to consume from 30 to 50 per cent of the time during which the machine is operable. Hence, it is not surprising that much effort has been devoted to schemes for making coding easier, faster, and more accurate. "Automatic coding" is a term loosely applied to a variety of procedures designed to accomplish these ends by using the computer itself as an aid in establishing the actual code that it is to execute.[1-3] All these schemes rely upon the use of precoded subroutines, and it can reasonably be claimed that the first steps toward automatic coding were taken at the University Mathematical Laboratories in Cambridge, England, as set forth in the works of Wilkes, Wheeler, and Gill cited above. Since then great development has taken place, for example, by the organizations of users of IBM and of Sperry-Rand equipment in the SHARE and USE activities and by many others.

Automatic coding systems ordinarily replace the actual instructions in machine language by pseudo instructions written in a more easily remembered form, for example, in arithmetical symbols and in abbreviated words, which, being more like ordinary language, are easier to use without error. Obviously, at some stage the pseudo code of the computation to be performed must be translated into the machine language. The translation is effected by the computer itself, obeying a set of instructions in the machine language previously inserted into the machine and called an "executive" or "supervisory" routine, and using a dictionary of equivalents. The translation can be one-to-one or one-to-many, and can be done in a variety of ways. If the translation proceeds instruction by instruction as the computation progresses, so that each pseudo instruction is translated just before it is executed, the supervisory routine is called an "interpreter." If, on the other hand, the complete code is generated in machine language from the pseudo code before the computation begins, the supervisory routine is called a "compiler." In the PACT I system,[4] for example, compilation proceeds by several successive passes of the magnetic tapes; at each pass, the information read from a tape is modified and re-recorded, until a tape carrying the desired code in machine language is finally produced. In translating a pseudo code, a compiler can effect a variety of functions, as examples of which we list the six following functions:

1. Replacing pseudo instructions by instructions in machine language
2. Causing the conversion of numerical information (i.e., from binary-coded-decimal to binary notation)

[1] Symposium on Automatic Programming for Digital Computers, Office of Naval Research, Washington, D.C., 1954.

[2] Symposium on Advanced Programming Methods for Digital Computers, Office of Naval Research, Washington, D.C., 1956. The first lecture, by Dr. Grace M. Hopper, contains the set of definitions of terms that has been followed here.

[3] McCracken, *op. cit.*, chap. 18.

[4] O. R. Mock, Logical Organization of the PACT I Compiler, *J. ACM*, vol. 3, no. 4, pp. 279–287, October, 1956. The same issue contains six other papers on the PACT I system.

3. Selecting subroutines previously stored in some part of the machine's memory
4. Causing the production of subroutines (by means of subsidiary supervisory routines called "generators")
5. Allocating addresses in the memory
6. Putting together several sets of instructions, each written with relative addresses, into a single set (by means of a subsidiary supervisory routine called an "assembler")

It appears from the above discussion that automatic coding schemes either require the use of the computer before the computation begins, and thus tie it up when it might be performing other computations, or else use up a good deal of memory space to hold the instructions of an interpretive routine. The general opinion is that the use of computer time in compilation is not wasteful, since it greatly reduces the time required for code checking and so actually saves machine time. The wasting of memory space can obviously be disadvantageous in some situations, although in others it is of no consequence.

The refining and expansion of the art of automatic coding are being actively pursued, and the day is not far distant when pseudo codes will be so simplified that they amount to little more than a simplified version of ordinary language plus a few mathematical symbols. Another development that is sure to come is the development of a common language system in which the same pseudo code can be used in stating problems for several internally dissimilar computers.

CHAPTER 3

SOME GENERAL CONSIDERATIONS ON SYSTEMS

3-1. Introduction. The two preceding chapters have been devoted to the forms in which information is encoded in digital-computing or, better perhaps, data-processing systems and to the types of operations commonly performed by such systems. It is now necessary to turn to the physical realization of the systems; to a number of general considerations pertinent to this subject the present chapter is devoted. There follow a number of chapters containing rather detailed treatments of basic circuits and components, and finally, in Chaps. 13 to 16, it becomes possible to show how these elements can be combined to produce the major functional units of a system and how these units are caused to work together.

3-2. The Basic Functions and Units. It is easy to list a set of basic functions that must be performed if a system is to exist at all. (1) It is obviously necessary to introduce information into the system; this is usually spoken of as "input." (2) It must be possible to hold information until it is needed; this is called "memory." (3) It must be possible to do certain things to the information, e.g., to perform the operations of arithmetic upon items interpreted as numbers; this is frequently called the "arithmetic" function although "data processing" would be better. (4) It must be possible to cause the system to function in accordance with the wishes of the operator, as expressed in the instruction code; this is "control." (5) It must be possible to remove from the system the results of the data processing; this is "output."

At this point it is customary to produce a block diagram, with one block for each of these functions, interconnected by lines representing the flow of information to be processed and of commands that the processing be effected. This is not done here, because it is felt that such diagrams tend to be misleading. It should be clearly understood that to distinguish five major functions does not imply that each of them is performed wholly or exclusively by a single unit of the system. For example, the complex of circuits used for data processing ordinarily contains certain memory elements, and so does the complex used for control. Even more notably, the control function is generally distributed throughout the system. There is usually a portion that can reasonably be called the "central" (or main) control, which exercises a general interpretative and supervisory function, but there are also control circuits that are quite narrowly connected with the data-processing and the memory functions; the degree of centralization and delegation of authority varies considerably among existing systems. If this is kept clearly in mind, it does no harm to use such convenient terms as, e.g., "arithmetic unit," which should be understood to mean the aggregation of circuits and (perhaps) devices used in carrying out and controlling the data-processing function. These terms are in common use, are convenient provided they are properly understood, and will indeed be used. It is hoped that the reader will always keep clearly in mind the general principles just stated; he will find them abundantly illustrated in Chaps. 13 to 16.

3-3. External Characteristics and Inner Structure. One who is interested only in using a data-processing system is naturally interested chiefly in the features that affect him directly. In order to operate at all, he must be informed about the number system and instruction code employed, the memory capacity, the speed of operation, how to insert information into the system, what sort of output devices is available, what sort of reliability he can expect, and what sort of checking is provided. He can afford to be rather indifferent to details of the inner logical structure and the circuitry employed and, presumably, can operate quite effectively if he regards the system as a kind of "black box." The properties of a system that appear on the outside and that are of direct importance to the user may as well be called its "external characteristics." In this chapter the general procedure will be first to deal with a number of important features of the inner structure of digital computers and then to show how they affect the external characteristics. It will be necessary also to deal in a general sort of way with the question of reliability as influenced by circuit design and with the question of checking.

3-4. The Machine Language. The first two chapters have described machine languages in some detail, and so only a few remarks will be made here. Machines are frequently classified by type of number system and of instruction code employed.

With respect to number systems, there are the binary and the decimal machines (other number systems have not been used), each of which may be fixed-point or floating-point. Since binary memory elements are simple and natural, but decimal elements are complicated and in many instances merely combinations of binary ones, and since the logic circuits are essentially the physical means of carrying out the operations of the two-valued Boolean algebra of logic, an understanding of binary systems is fundamental. For this reason binary systems only are treated.

Ideally, as pointed out in Chap. 1, a binary representation of a given number requires fewer bits than a binary-coded-decimal representation and, hence, should be more economical of memory elements. Inasmuch as in binary machines the input and output are commonly in binary-coded-decimal notation, with conversion to and from binary done internally, the theoretical economy is not realized in the registers used for input and output; nor is it realized in the registers used in performing arithmetical operations, for these must be designed to accommodate the coded decimal numbers although, during the normal course of computation, they handle only binary numbers and, hence, are inefficiently used. In the internal memory, however, it is clear that the theoretical economy could in fact be attained by appropriate design, although it is not usually considered expedient to do so because it can be accomplished only at the expense of introducing additional complications into the system. The general conclusion, therefore, is that the theoretical advantage of the binary system in reducing the necessary number of memory elements is llusory as far as present machines are concerned.

Since the arithmetic involved in conversion of binary-coded-decimal $\leftrightarrow$ binary notation is not very involved, it would seem that a special converter unit might be built to carry out the conversions externally. This would permit the theoretical advantage of the binary system as compared with the binary-coded-decimal system to be fully realized in the machine proper. It must be recognized that this scheme also involves difficulties, and certainly in a small system the additional unit would probably use more equipment than was saved in the machine proper, but it is possible that in a large-scale system with a large memory capacity a net saving could be achieved.

An advantage of decimal machines using the binary-coded-decimal representation of numbers, from the point of view of business-data-processing applications, is that the code can be extended (as mentioned in Chap. 1) so that alphabetic as well as numerical information can be encoded.

It has been noted in Chap. 1 that most of the earlier computers operated in a fixed-point system because floating-point arithmetic required more equipment. The trend toward floating-point operation is now marked, and it seems obvious that before long all but the simplest and cheapest machines will be of this character. In Chap. 13 will be found an account of a floating-point arithmetic unit.

As far as instruction codes are concerned, enough has already been said in Chap. 2, to which the reader is referred.

3-5. The Flow of Information. Turn now to consideration of the internal structure of digital computers. This section will deal with the way in which information is transmitted from point to point. The two extremes are serial transmission, in which each word is transmitted along a single conductor as a train of accurately timed pulses synchronized with a master oscillator usually called the "clock," and parallel transmission, in which all the bits of a word are transmitted simultaneously along a set of conductors equal in number to the number of bits in the word, each conductor being allocated to a definite binary position. Between these extremes, it is possible to use various modes of series-parallel transmission. A common example occurs in decimal machines that use binary-coded-decimal representation with 4 bits per decimal digit, where four conductors are used on which the 4 bits that represent a single decimal digit are transmitted simultaneously, but where the digits are transmitted in time sequence. Another situation occurs in the Manchester megacycle computer (MEG), where the 40 bits of a word are removed from the memory in blocks of ten, each block is transmitted in parallel, and the four blocks are transmitted in series.

There is an essential difference between serial and parallel transmission: in serial transmission the information takes the form of a train of pulses, and in parallel transmission it may take the form of a set of quasi-fixed voltages impressed upon the wires, and may be received either in this form or in pulse form when desired, the latter being accomplished by the enabling of a set of gates (see Chap. 4). Thus serial transmission is well adapted to systems of a highly synchronous type, and parallel transmission is equally well adapted to the asynchronous type, which will be discussed below.

In serial transmission the interval occupied by a single word and the space separating it from the word immediately following is usually called a "minor cycle." Thus, in SEAC, the words have 45 pulse positions and are followed by a space of 3 pulse periods; so a minor cycle is 48 pulse periods or 48 μsec in length. This is the time required for a single word to pass a given point in the circuitry. With parallel transmission, a single pulse period suffices to transmit an entire word, and various series-parallel schemes require varying amounts of time depending upon the number of groups of bits transmitted in series.

3-6. The Retention of Information. Facilities that retain information until it is needed in the course of the computation include vacuum-tube (or other) "registers," small temporary-memory units for single words or perhaps a few words, and large-scale memories for many words. Some sort of retention of information is required in connection with all machine functions. Thus, in the arithmetic unit it is necessary to retain the first of two operands until the second has emerged from the memory, in the control it is necessary to retain the bits specifying the operation to be performed until (at least) all the circuits needed have been properly set up, and in the input unit it is usually necessary to provide some kind of register in which the bits read into the machine are assembled before transmission into the high-speed internal memory.

The various memory devices differ very greatly in size, speed of operation, and complexity. Vacuum-tube (or transistor) registers are the least economical in equipment but have the advantages that the information held is instantaneously available for parallel transmission and that serial transmission can also be provided for if the register is so constructed that its contents can be shifted one position at a time. Vacuum-tube (or transistor) registers are commonly encountered in the arithmetic and control circuitry and associated with input and output; they are much too expensive and far too small in capacity to provide the main memory of thousands of words. For this function a great number of devices are in use. Some are intrinsically serial in their natural mode of receiving and transmitting the information, others are intrinsically parallel, and still others can more or less naturally be arranged for either or both types of transmission and reception. These points will be covered in some detail in Chaps. 8, 11, and 12.

One way of classifying memories, then, is as "serial" or "parallel." Another important classification distinguishes between memories in which all words held are equally available with precisely the same time delay and memories in which the information is constantly circulating, so that a given word is available periodically at intervals equal to the time required for a signal to pass around the circulation path. The former are said to be "random-access" memories; examples (see Chap. 14) are Williams memories and magnetic-core-matrix memories. Examples of circulating memories are those that employ acoustic delay lines and magnetic drums. The random-access memories are usually, but not necessarily always, operated with parallel input and output, whereas the circulating memories usually receive and transmit information serially.

In all cases, the waiting time between the demand for a word and its transmission from the memory is called the "access time." In random-access memories this is the same for all words. As a first example, consider the parallel Williams memory of the Institute for Advanced Study (IAS) computer. Because information held in the memory deteriorates rapidly, it must be restored periodically; for this reason, the memory is operated cyclically, regeneration cycles alternating with action cycles (in which access to the memory is permitted). The memory cycles are 25 μsec in length (they could if necessary be reduced by about 50 per cent). Each instruction word contains two instructions. The first of these can be put into execution during the cycle immediately following the cycle in which the instruction word was brought from the memory (this is the exception to the rule that a regeneration cycle must always follow an action cycle); so the access time of the first instruction is one memory cycle, or 25 μsec. For the second instruction of the pair, a regeneration cycle must intervene between the instant when the instruction becomes available to the control and the initiation of the operation called for; so the access time becomes 50 μsec. Observe that the access time depends in effect upon the control circuitry as well as upon the memory device used; in the case cited, nothing at all is done during about half the cycle, and the other half is devoted to the generation of a sequence of pulses which actually cause the reading or writing operation to be carried out.

As a second example consider the magnetic-core memory of Whirlwind I. No periodic regeneration is necessary, but, because of the destructive character of the read-out process, it is necessary to restore each word immediately after reading it; this leads to a basic memory cycle of about 6 μsec (originally about 10 μsec). As each instruction word contains but a single instruction, an instruction obtained in one cycle can begin to be obeyed in the next, and the access time is effectively the length of the cycle, although, if the availability of the address is taken for granted, it becomes only a third or a quarter as long.

In contrast to the true random-access memories just discussed, memories of the circulating type exhibit quite long access time. The mercury-filled acoustic delay lines of the SEAC introduce a delay of 384 μsec, and thus hold eight words of 45 bits plus a 3-μsec word space each, the prf (pulse-repetition frequency) being 1 Mcps (megacycle per second) and the pulse length about 0.5 μsec. The access time varies from word to word, depending upon its position in a delay line when the demand is made; clearly the average access time is one-half the total delay, or 192 μsec. It would be possible to reduce this figure by making the lines shorter, but this would be wasteful of equipment, since more recirculating circuitry would be required.

For a second example, consider magnetic drums. These are ordinarily rotated at about 3,600 rpm (i.e., synchronously with a 60-cps power supply); hence one rotation requires $\frac{1}{60}$ sec, so the average access time is $\frac{1}{120}$ sec, or $8\frac{1}{3}$ msec.

Different types of memory differ also with respect to capacity. Thus each cathode-ray tube of a Williams memory can be made to hold about 1,000 bits, given proper circuit design. This can be doubled or perhaps quadrupled if extreme care is taken. If 40-bit words are considered standard, at least for purposes of comparison, an average Williams memory might be considered to require 40 cathode-ray tubes and associated equipment for a capacity of 1,000 words. If two banks of cathode-ray tubes are used in order to double the capacity, the amount of equipment in the memory is doubled.

In acoustic- (and other) delay-line memories, a compromise must be made between average access time and the number of words per line; the figures for the SEAC have already been quoted. Some machines use lines 1,000 μsec in length with correspondingly greater capacity and access time; in the interest of over-all computing speed this is about the limit. Such a line could be made to hold 1,000 bits; this might even be doubled if the pulse length were halved and the prf doubled, but design problems would become more severe. Thus a capacity of one thousand 40-bit words requires 40 such lines if a prf of 1 Mcps is used; of course, a second bank of lines could be added to double the capacity, but again this would involve doubling the already extensive electronic circuitry.

Magnetic drums are intrinsically large in capacity. Information is recorded in the form of small magnetized spots on circumferential tracks, and on these a density of 100 spots (i.e., bits) per inch is conservative. Thus, a drum of diameter somewhat less than 8 in. can carry 60 words of 40 bits each in every track, and, as the tracks can be packed quite close together, a capacity of, say, 5,000 words can be obtained with a fairly small drum. It is true that reading and writing circuits must be available for each track, but these are fairly simple in structure and, in some applications at least, by means of a switching scheme, can be shared by several tracks. In fact, if the drum is used as an auxiliary to a small low-access-time memory, it is possible to arrange for information always to be transferred between drum and fast memory in blocks of many words each, with rather long times between transfers. In this case, the magnetic heads, which produce the magnetization in writing and detect its presence in reading, can be moved physically from one set of tracks to another.[1]

Finally, arrays of magnetic cores are used to build up the best and at present most popular short-access-time random-access memories. In these, each bit is assigned to one tiny ferrite core, which can be saturated magnetically in two opposite directions, corresponding to binary 1 and 0. These cores are arranged in square or rectangular arrays (see Chaps. 12 and 14), the usual procedure being to let each such array represent a binary position—there being a stack of as many such arrays as there are binary positions in a word—and to let the cores that represent the bits of a single word occupy corresponding physical positions, one in each array. Selection can be accomplished by a variety of methods, all of which reduce to performing a selection of the row and column proper to the desired word. Fortunately, doubling the number of cores in each array does not double the amount of circuitry needed to select and drive the cores; the circuitry required is doubled only when the number of cores in the array is squared. The first working memory of this kind was installed in Whirlwind I to replace the earlier memory composed of specially constructed electrostatic memory tubes; this unit held 1,024 17-bit words

[1] This is done, e.g., in the pilot model of the ACE at the National Physical Laboratory in England and in the DEUCE, which is a production model of this computer built by the English Electric Company, Ltd.

(15 bits + sign + parity check) in 17 square arrays of 32 × 32 cores each. Later work at MIT and elsewhere led to much larger arrays; e.g., in the TX-O built in the Lincoln Laboratories, the arrays are of 256 × 256 cores. Thus core memories can be built that have as great a capacity as a magnetic drum, without the disadvantage of long access time.

Tremendously large capacities, at the expense of access times measured in seconds rather than in microseconds, are offered by a variety of devices, such as those employing magnetic tape, punched paper tape, or photographic film as a memory medium. Another scheme, developed at the National Bureau of Standards (NBS) and discussed in Chap. 16, uses an array of stationary plates coated with magnetic material, which are scanned by movable recording and playback heads. Devices like these are used as auxiliaries to the short-access-time memory and also for input and output.

3-7. The Processing of Information. Let us now consider the actual carrying out of the operations. The title of this section is chosen deliberately to emphasize the possible nonarithmetical character of the operations performed, which can include those of logic as well.

Here again serial, serial-parallel, or parallel operation can be employed. In all cases it is necessary to obtain the operands sequentially from the memory; the data-processing circuits, therefore, must have associated with them some sort of register in which the first operand to emerge from the memory can be held until the other arrives. The total time required to execute the instruction includes both the times required to obtain the operands and the time required by the data-processing circuits to operate upon them.

In addition, for example, the register has associated with it adding circuits, which form the sum of the number already held and the number arriving (the complex of circuits being called an "accumulator"); this is true in both serial and parallel adders, but in a serial-parallel adder the register is split into several sections, as many sections as there are parallel channels. The serial adder is quite simple. It receives in synchronism the two pulse trains representing the numbers to be added, and its output is the sum, which is formed, therefore, as far as the adder itself is concerned, in one minor cycle. In the parallel case there must be an adder for each binary position, with provision for carry from each position to the next more significant one. Since carries must be propagated along this structure, the time required to effect the addition, as far as the adder itself is concerned, is the time required for carrying. Given ordinary circuit techniques, the carry-propagation time may be as high as about 10 μsec; in the IAS machine 15 μsec is allowed for a comfortable margin. This can be reduced by more sophisticated methods that use more equipment, such as the Whirlwind high-speed carry scheme already discussed in Chap. 1.[1]

Another scheme is especially applicable to asynchronous machines. It depends upon the fact, proved by von Neumann,[2] that the average number of carries in the addition of two n-bit numbers does not exceed $\log_2 n$. (For $n = 40$, $\log_2 n$ is only 5.3 and the accurate value of the average number of carries is 4.62.) Thus, instead of allowing for addition the time required for carry propagation in the worst possible case, means can be included of establishing when the carries have actually been completed and then indicating to the appropriate part of the control that the operation has in fact been finished and that the next one may therefore be initiated. A scheme for doing this, developed at IAS, will be described in Chap. 10.

Serial-parallel schemes of various kinds can be devised. As an example, consider that used in MIDSAC.[3] The 32-bit words are read out of the Williams memory in parallel into four shifting registers each capable of holding 8 bits. When these are shifted, they transmit serially to four accumulators, each consisting of an 8-bit register with an adder in its input. Carries are transmitted from the lowest-order column to the next, and so on, and finally from the highest-order column back to the lowest; enough delay is introduced so that this carry is delayed one pulse time. Thus the addition is accomplished by eight parallel additions of 4-bit blocks. The prf is 1 Mcps, and in each parallel addition there is time enough for carry propagation to be completed; so, as far as the adder is concerned, addition of two 32-bit numbers requires 8 μsec.

For comparison, consider the addition times in the three cases, assuming 40-bit numbers and a data rate of 1 Mcps for serial transmission. These work out to (*a*) serial, 40 μsec; (*b*) parallel, about 10 μsec or less, down to about 1 μsec, depending upon the technique used; (*c*) serial-parallel (as in MIDSAC), 10 μsec. Thus the serial-parallel scheme is almost as fast as the parallel and requires less equipment. It might be thought that eight-wire transmission could be used to reduce the addition time by half in MIDSAC. In that case, however, 1 μsec would probably be insufficient time for carry propagation, and so a slower digit rate would

[1] See also Margaret F. Mann, R. R. Rathbone, and J. B. Bennett, Whirlwind I Operation Logic, *MIT Digital Computer Lab. Rept.* R-221, issued under ONR contract N5ori60, May 1, 1954.

[2] A. W. Burks, H. H. Goldstine, and J. von Neumann, "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," report on Army Ordnance contract W-36-034ord7481, p. 10, Institute for Advanced Study, Princeton, N.J., June, 1946.

[3] W. Brown, J. De Turk, H. Garner, and E. Lewis, "The MIDSAC Computer," report on Project MX-1599, pp. 30–37, Engineering Research Institute, University of Michigan, Ann Arbor, April, 1954.

be mandatory (say 500 kcps), and twice as much adding equipment would be needed; nothing would be gained in speed to compensate for the extra complexity.

In multiplication and division, many machines use the straightforward but rather slow methods of successive additions and of additions and subtractions discussed in Chap. 1. With additional equipment it is possible to speed these processes considerably. One method is discussed in the account of the MEG multiplier, given in some detail in Chap. 13. In the serial-parallel MIDSAC, an array of 16 simple serial adders suffices to effect the multiplication of two 32-bit numbers in eight minor cycles (that is, 64 μsec). Many schemes of this sort have been worked out.[1]

Despite the emphasis at the beginning of this section on the importance of nonarithmetical computation, nothing has yet been said about logical operations. Two examples of these are logical multiplication and logical addition: forming from two binary words a new word such that the bit in any position is (1) the logical product (here the same as the ordinary product) or (2) the logical sum of the bits in the same position of the two input words. These are simple operations and can be rapidly executed because no carry is involved.

3-8. Computing Speed. The actual computing speed of a digital computer depends both upon the access time to the memory and upon the circuits used to carry out the computations. From the discussion of the last three sections it is now easy to derive the times required to perform the basic operations of arithmetic for a given computer.

First, consider the IAS computer. The basic memory cycle as the machine is currently operated is 25 μsec. About 15 μsec is allowed for the propagation of carries in the adder, and this is fixed. Hence, a complete addition operation requires one cycle to obtain an instruction pair, one cycle to add the first addend to the (cleared) accumulator, one regeneration cycle, and one cycle to obtain the second addend and add it to the contents of the accumulator, or a subtotal of four cycles, or 100 μsec. The sum now stands in the accumulator register; to transfer it to the memory, it is necessary first to obtain a second instruction pair from the memory. So the following cycles must elapse: one for regeneration, one to obtain the next instruction pair, one to effect the transfer; that is, 75 μsec. Hence the complete operation requires 175 μsec, which is reduced to 150 μsec if the first addition instruction is the second instruction of a pair. In multiplication without round-off, 41 additions take place. As a delay of 15 μsec is introduced in each addition to ensure that carry propagation has been completed, it is evident that, in the worst case (all 1's in the multiplier), the multiplication process itself consumes 615

[1] R. K. Richards, "Arithmetic Operations in Digital Computers," chap. 5, D. Van Nostrand Company, Inc., Princeton, N.J., 1955.

μsec. To this must be added the time to obtain the instruction pair, the time to obtain the multiplier, and a regeneration cycle, the operation being initiated upon the appearance of the multiplicand. All together this requires $615 + 75 = 690$ μsec, and an additional 50 μsec is required to obtain the instruction calling for the transmission of the product to the memory and to carry out this transmission. Evidently, if an improved adder were installed in which only the carry-propagation time required in each step was introduced, these figures could be greatly reduced. The current estimate is not much over 10 μsec plus the access times. Division requires about the same time as multiplication.

In SEAC, the instruction code is of either the three- or the four-address type, the choice being left to the programmer, and so a single instruction word suffices for the whole operation. If the mercury-delay-line memory is used, the time required to obtain a desired word from the memory can be as little as 48 μsec or as much as 384 μsec. Since the performance of a complete addition operation requires that the instruction word and then the two operands be obtained from the memory and, finally, that the sum be put back in the memory, addition can consume from 192 μsec to 1.536 msec, the average being 864 μsec. Multiplication is accomplished by successive additions, each requiring one minor cycle. Since one of the 45 bits of a word is the sign, there must be 44 additions, consuming $44 \times 48 = 2{,}112$ μsec. Four accesses to the memory are required; so the total operation time can be as large as 3.648 msec or as small as 2.304 msec. The time required for division has exactly the same bounds. SEAC has a logical-multiplication instruction; as remarked above, the time required for such operations is generally short, in this case the same as for addition. If the Williams memory is used as the exclusive source of instructions and operands, addition requires but 240 μsec, and multiplication and division each require 2.35 msec.[1]

Without going into details, the times required for MIDSAC operations will be listed: (1) addition, 40 μsec; (2) multiplication, 88 μsec; (3) division, 272 μsec.*

It is quite clear from the above discussion that, given the conventional circuitry, which operates satisfactorily on pulses about half a microsecond wide, occurring at a prf of 1 Mcps, and given a memory with access time of a few tens to a few hundreds of microseconds, both access time and processing time make significant contributions to the total time required to perform the operations. If long-access-time memories are used, such as magnetic drums, the operation time is for all practical purposes determined entirely by the accesses to the memory, unless great care is taken

[1] S. Greenwald, S. N. Alexander, and R. C. Haueter, SEAC, *NBS Circ.* 551, Jan. 25, 1955. See the table on p. 26.

*Brown, De Turk, Garner, and Lewis, *op. cit.*, p. 53.

so to insert the information into the memory that it becomes available just at the instant it is required—in which case ("optimum programming") quite high-speed operation can be obtained.

If machines substantially faster than those now in existence are to become feasible, both processing time and access time must be reduced. In serial machines, processing time can be reduced by operation at higher prf's; in parallel machines, better transient response is required; in both cases this amounts to a requirement for greater bandwidth in the circuits. Within limits, such improvements can be accomplished without too great a departure from present circuit techniques.

Using different techniques, very great increases in speed are obtainable. Thus Fischer and Marshall[1] designed a very fast scaler based upon a bistable circuit (i.e., a binary memory element) of 10 $\text{m}\mu$sec resolving time, using grid-controlled secondary-emission tubes. However, fast circuitry cannot accomplish much in reducing the time required for operations unless access time is also reduced. Furthermore, its use would impose very severe requirements upon the memory. Suppose, for example, that it was desired to build a machine like SEAC in all respects except that it would be ten times as fast in operation. From the discussion above it is evident that the speeding up of addition would require a 10:1 reduction in access time. Hence, the mercury delay lines would have to be reduced to 38.4 μsec in length, so that, if each were to hold as much information as in the original machine, a prf of 10 Mcps would have to be used with, say, 0.05-μsec pulses; this would be exceedingly difficult, if not impossible, to accomplish. It might be thought that a lower prf might be used and a loss in memory capacity per delay line accepted, in the interest of speed. However, with respect to multiplication, it becomes clear that a 10:1 reduction in the length of the minor cycle would be necessary, at least given the present method of forming the product; this in effect reimposes the 10-Mcps prf. A way around this difficulty exists: it consists in using a more elaborate multiplier. If this were done and if a reduction of 2:1 in delay-line memory capacity per line were accepted, it would be possible to use a 5-Mcps prf—presumably feasible, for a rate of 3.77 Mcps has been used successfully in RAYDAC.[2] The moral to be drawn from the above discussion is that a brute-force attack on the speed problem by increasing prf's and the speed of response of circuits is likely to meet with only limited success; more sophisticated logic is required, and it would appear, in fact, that the problem of building superspeed computers involves con-

[1] J. Fischer and J. Marshall, A Ten Millimicrosecond Scaler, paper read at the Natl. Electronics Conf., Chicago, 1953. Equivalent work was done at the George Washington University by Prof. Z. Bay and Dr. N. T. Grismore.

[2] "A Survey of Digital Computers," p. 81, Office of Naval Research, Washington, D.C., 1953.

siderable modification of the conventional logical design as well as improvements in circuitry and memory devices. In a rather modest way, MIDSAC affords a good example of what can be accomplished by logic alone, with no increase in the operating speed of individual circuits and only a small improvement in the Williams-memory access time.

3-9. The Need for Increased Speed. It is important to realize that, for all the successful development of digital computers to date, they are still unsatisfactory for carrying out some of the most important and characteristic computational problems of mathematical physics, and their failure here is due chiefly to lack of speed. It is also true that currently attainable speeds are only barely sufficient to make digital techniques of any use in many control-system applications, where extensive real-time treatment of data derived from a number of sources must be carried out. The failure with respect to problems in physics is the more scandalous, for it was precisely for scientific computation that the first digital machines were designed.

The failure lies in the field of the numerical solution of partial differential equations, which arise in all branches of fluid mechanics.[1] One of the earliest problems successfully reduced to computation was that of the propagation of a spherical shock wave. In this problem, which involved a hyperbolic equation in time and but a single space coordinate, it was necessary to carry out the computation at 100 spatial positions for each time step and to do this for 3,000 time steps. About 4,500 multiplications and divisions are executed at each time step, plus other less time-consuming operations. For the 3,000 time steps, the total amount of computing was equivalent to about 1.7×10^7 multiplications. Assuming a 1-msec multiplication time, this would require nearly 5 hr of machine time; actually, if rather careful checks are included in the problem code, this time might be at least doubled. If it were desired to find the effect of changing a parameter, it might be necessary to repeat the whole computation a considerable number of times (say 50). Thus, even for a comparatively simple problem, a survey computation of this type would consume a tremendous amount of time; if 10 hr is needed for a single computation with programmed checks, the survey involving 50 parameter values would require 500 hr of machine time.

The situation becomes much worse. A hyperbolic equation of much the same character but involving two space coordinates might easily require that values be computed at each point of a spatial lattice of 2,500 points and that this be done for each of 3,000 time steps. A reasonable

[1] Much of this material is taken from the author's notes on a lecture by H. H. Goldstine delivered at the Computer Conference held at Darmstadt, Germany, in October, 1955. Any inaccuracies can properly be attributed to the author, not to Dr. Goldstine.

estimate, including programmed checks, gives a computation equivalent to 5×10^8 multiplications, which, at 1 msec per multiplication, would require nearly 140 hr of machine time. Thus a survey computation involving even a modest number of parameter changes is for all practical purposes out of range for a machine of the speed assumed. Furthermore, in such a computation, it is necessary to hold in the memory a great deal of information; so a memory capacity of about 12,000 words would be required.

For a three-dimensional hyperbolic equation describing shock-wave propagation, even for a relatively coarse mesh (e.g., a cubic lattice with 20 divisions per edge), the equivalent of 1.7×10^9 multiplications would have to be performed and a memory capacity of over 50,000 words provided. Thus a single computation of this sort would require 500 hr of machine time.

Similarly depressing estimates can be made for elliptic and parabolic equations. The general conclusion to which one is forced is that a machine that would be really useful for such computations would have a fast-access memory capacity of from 20,000 to 50,000 words and would be fast enough to complete 10^{10} multiplications in a reasonable time. This means that a multiplication time of 10 μsec or less is needed and an access time of about 0.1 μsec. Of the two, the fast circuitry is the more nearly attainable in the present state of the art; it seems clear that memories of such short access time are at present out of reach and that their physical realization must wait upon new discoveries.

3-10. Timing: Synchronous and Asynchronous Systems. Since the execution of any data-processing program is a sequential process, some sort of timing must be involved. In serial transmission, the pulses representing the bits of a word must follow one another in a definite sequence, since this is the only way in which they can retain their identity; they must also be uniform in shape so that they will all be treated equally well by the electronic circuits. Hence, in serial and serial-parallel machines, all transmission, processing, and transfers to or from the memory are regulated by timing signals derived from a master oscillator called the "clock," which can be as elaborate and precise as a crystal-controlled oscillator or as simple and relatively unprecise as a train of permanently inscribed signals upon a magnetic drum. Since the pulses deteriorate in shape with transmission, they are (at least in many machines) reshaped and retimed with reference to the clock from point to point in the system to guarantee that the timing relations are properly preserved.

In parallel transmission and processing, alternative procedures exist. The first is to regulate all machine operations strictly by signals derived from a clock as in the serial case; this is done, for example, in Whirlwind I. The second alternative for completely parallel systems is to use no standard time reference, but to permit the various operations to be self-timing, so that the circuits concerned are set into operation by a signal from the main control, proceed with the appropriate operation at their own pace, and, upon its completion, generate and transmit back to the main control a signal that causes the initiation of the next operation to be commanded. As an example, consider the arithmetic unit of the IAS computer, which will be described in detail in Chap. 13. This contains a circuit that is held quiescent until an arithmetical operation is to be performed, whereupon it is set into operation. At the same time, the number of cycles that it must execute is presented as one input to a comparison circuit, whose other input is the contents of a counter that counts the cycles executed. When agreement is reached, the operation is terminated, the circuit is brought back to rest, and a signal is sent to the main control signifying that the operation has been completed.

Operation of the kind just described is said to be "asynchronous." It is very well adapted to parallel systems, though it would be difficult to say whether it is better than synchronous operation. It simplifies the system by making it unnecessary to provide the same time reference in all units. It also makes possible enhanced speed of operation, for a fixed period of time need not be allowed for each operation; this is true particularly in multiplication and division.[1] It appears to lead to considerable economy in vacuum tubes—according to a comparison, for example, of the parallel asynchronous IAS computer with the parallel synchronous Whirlwind I, which has a much shorter word length. However, Whirlwind has certain features, like the fast carry and marginal check, that the Institute machine has not; so mere numbers of tubes probably give a rather exaggerated notion of the economy of the asynchronous design.

3-11. General Types of Circuitry. Going back to the manner of representing 1's and 0's electrically, recall that this may be done by pulses (or their absence) occurring at times synchronized with a master clock or by voltages assuming for comparatively long periods one of two possible levels. It has been proposed to call these "a-c" (alternating-current) and "d-c" (direct-current) systems, respectively.[2] Alternatively, the first type can be spoken of as "dynamic" and the second as "static"[3] or, perhaps better, as "quasi-static." In most systems both representations are used; for example, it is quite common in serial machines to "staticize" the instruction word by inserting it into a shifting register built up of vacuum-tube-toggle circuits, so that a set of temporarily fixed voltages

[1] Richards, *op. cit.*, p. 342.

[2] *Ibid.*, pp. 342–343.

[3] Dynamic and static accumulators were discussed by Burks, von Neumann, and Goldstine as early as 1946 in the "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument."

are available to activate the logic circuits that control the flow of information during the operation.

The most thoroughly dynamic computer is probably the serial synchronous SEAC, in which all information is constantly in motion in the form of pulses; even the binary memory cells of which certain registers are built up consist of short loops about which a pulse circulates and is available at a specified point once every cycle of the clock. The other extreme is represented by the IAS machine, in which, however, dynamic operation occurs in the circuits controlling the cycling of the Williams memory and the performance of the operations; evidently there must be some dynamic operation in a system if it is to function at all.

It is evident that thoroughly dynamic operation must also be carefully synchronized. Thus in a serial dynamic system it is the general practice to reshape pulses and to reestablish their time reference as they progress through the circuitry, and it is hard to see how this can be avoided. Furthermore, it is necessary in design to take into careful account the time delays along various paths that converge at a point and to ensure that suitable equalization is introduced.

Clearly, considerations of this sort strongly influence the type of circuitry used. Thus, quasi-static operation leads naturally to direct coupling between individual circuits; for dynamic operation, transformer coupling is natural and popular, or capacitive coupling can be used. Each system naturally has its own advantages and disadvantages. Direct coupling is the simplest but may require the provision of a number of levels of d-c supply voltage. On the other hand, with capacitive or transformer coupling, the use of quasi-static circuits like ordinary flip-flops (or "toggles," as they will be called here) involves the difficulty that the output is transmitted only as a transient. In Whirlwind I, for example, this forced the incorporation of so-called "restoration," in which, by a specially provided pulse train applied through specially provided input circuits, all toggles are periodically caused to switch twice in rapid succession, so that their state is not changed but the charge on the capacitor that couples from the plate of a tube to the grid of the tube controlled by it is periodically restored.

It must be emphasized that direct comparisons between different types of systems are nearly impossible to make in a rational way. Each type has its own merits and its own shortcomings. The important thing is to achieve as natural a design as possible, which must always involve a compromise between general system considerations and considerations of the physical circuits to be used.

3-12. Reliability. Present electronic digital computers contain from a few hundred to many thousands of vacuum tubes and from none to perhaps tens of thousands of crystal diodes. It is quite obvious that systems of this kind cannot be expected to function perfectly all the time. That they function as well as they do is sufficient evidence of the skill of their designers and builders.

The question of the reliability of complex electronic equipment has been widely discussed in recent years. It is not intended to consider in detail the various definitions that have been proposed or to discuss the definition adopted by the Radio-Electronic-Television Manufacturers' Association (RETMA),[1] which equates the reliability of a device with the probability of its performing adequately for a specified period of time under a specified range of operating conditions. The important point to note is that, for purposes of comparison, a numerical measure of reliability has been introduced whereby figures of merit can in principle be assigned to various equipments designed for the same purpose.

It is felt, however, that different figures of merit (rather than the "probability" of the RETMA definition, which is difficult to establish) are appropriate in different situations. For digital data-processing systems it may be well to consider the mean time between failures, that is, the average interval in which the system functions properly. However, this does not take into account the time lost in preventive maintenance or the time lost in correcting troubles. These can be expressed by a ratio of the time during which the system is capable of functioning usefully to the total time during which any operations are performed at all, either for data processing, trouble shooting, or scheduled preventive maintenance. It is true, of course, that the time during which the system is capable of functioning properly will not all be devoted to useful operation, because of errors in programming, incorrect scheduling of work load, and so forth, but these factors measure the efficiency of the organization using the system rather than the excellence of the system. In any case, in comparing the ratios for different equipments, it is necessary to know how the ratios have been computed, for occasionally the scheduled maintenance time is excluded from the total "on" time, and the ratio of total "good" time to total "available" time is computed. This ratio, too, is a reasonable figure of merit, provided that the scheduled maintenance time is also specified.

If the two figures of merit described in the last paragraph are considered, it is evident that they measure the excellence of different aspects of the system. The first (average time between errors) depends upon the excellence of the components and of the circuit design; the second (opera-

[1] This whole question is discussed in detail by Captain W. F. Luebbert, U.S.A. Signal Corps, in Principles and Concepts of Reliability for Electronic Equipment and Systems, *Stanford Univ. Electronics Research Lab. Tech. Rept.* 90, Aug. 18, 1955. (A report on ONR contract N5onr25107, jointly supported by ONR, USAF, and the Signal Corps.)

bility ratio) depends upon the ease with which troubles can be located and corrected and also upon how many troubles there are.

As far as circuit reliability is concerned, a good deal can be accomplished if the design allows for fairly wide safety margins.[1] The designer who demands that too close component tolerances be held, who does not allow for changes in value due to aging and to instability of various kinds (e.g., due to changes in temperature and humidity), or who depends upon excessively good regulation of power-supply voltages, is asking for trouble. Fortunately, the fact that binary (and, for that matter, decimal) computing circuits operate on a two-level or two-state basis makes it fairly easy to avoid the necessity for any very close voltage-amplitude discrimination; this in turn makes the problem of designing for satisfactory operation under rather wide safety margins comparatively easy to solve.

The general procedure is first to design the circuit to give nominal values of output voltage for binary 0 and 1 and then to calculate the variations introduced by changes in the values of the circuit parameter. When the outputs have departed by the permitted tolerance from their nominal values, the maximum permissible parameter variation has been found; failure points can be determined in this way and plotted, e.g., as a function of the permissible tolerance in the output. Thus Taylor[2] has given as an example the tolerance in the crossover resistors of a toggle: under variation of a "marginal-checking parameter" (in fact, the plate-supply voltage) the deviation of these resistors from their nominal design values was increased until the toggle would no longer switch from one state to the other. Thus for each value of plate-supply voltage a failure point was determined, and these were plotted. The plots for two different designs reveal at a glance which is the better from the point of view of safety margin.

At this point it is not inappropriate to cite some of the rules laid down by J. H. Bigelow for the guidance of the engineers who designed the circuits of the IAS computer. As to vacuum-tube ratings, the average plate (or grid) dissipation should not exceed 50 per cent of the rated value, the average cathode current should not exceed 50 per cent of the rated value, and the peak cathode current should never exceed the rating. As far as the use of tubes in circuits is concerned, the design should be such that satisfactory operation continues even if, because of tube deterioration, the currents produced by a given set of voltages fall to as little as 50 per cent of the values specified in the published tube characteristics, and, when a tube is operated in the positive grid region and a specified

[1] An excellent discussion of these points was given by N. H. Taylor, Rudiments of Good Circuit Design, Electronic Components Symposium, Pasadena, Apr. 30, 1953, subsequently published as *MIT Digital Computer Lab. Rept.* R-224, May 19, 1953.

[2] *Ibid.*, pp. 8ff.

grid current is flowing, satisfactory operation should be obtained under variation of plate and screen volt-ampere characteristics from half to twice the average ones. As far as resistors are concerned, satisfactory circuit operation should be obtained under variations of up to ± 10 per cent from the nominal values, and dissipation should never exceed 50 per cent of the rated value. Apart from these rules on components, Bigelow laid down a number of general principles, of which the following are typical: (1) use no balancing of one pulse against another, either to cancel the two out or to override one by the other; (2) have no amplitude discrimination on pulses and, therefore, permit no spurious pulses that are expected to be rejected because of their small amplitude; (3) do not attempt to discriminate between wanted and unwanted signals by means of time constants. These rules proved to be very helpful in the design work, being particularly appropriate to the parallel asynchronous logic that had been decided upon.

Even with the best circuit design and with the best procurable components, failures are bound to occur from time to time, and measures must be taken to forestall them or to minimize their effect upon the operability of the system. Failures are of different types, of which the three following can be distinguished: (1) those due to the cumulative effect of gradual deterioration of components; (2) those due to sudden breakdowns, such as shorts or open circuits; and (3) those that are of an intermittent or transient character.

The question of how best to avoid trouble due to failures has been debated at great length; various measures have been used. These fall into three classes: (1) programmed checks, (2) checks performed during normal operation by special circuits, and (3) marginal checks—but the last may perhaps be more properly considered part of the routine preventive maintenance of the system.[1]

Programmed checks are of several kinds. The simplest consist merely in causing the system to execute from time to time (e.g., once per shift or once per day) a routine designed to cause every instruction to be obeyed once or more and in noting errors or causing the machine to stop when an error occurs. Thus, for example, to test the function of the arithmetic circuits, it is possible to place in the memory, along with each addition or multiplication instruction, the correct sum or product and to cause each computed sum or product to be compared by subtraction with the correct value, the code being so arranged that the machine is stopped if agreement does not occur. Codes of this kind may be so framed as to test the operation of various parts of the machine. Thus it is possible to test all memory addresses to ensure that information can be recorded in

[1] N. L. Daggett and E. S. Rich, Diagnostic Programs and Marginal Checking in the Whirlwind I Computer, *IRE Convention Record*, 1953, pp. 48–55.

and read from them, input and output can be checked, the arithmetic circuits can be checked, and so on. Thus failures can be localized; for this reason, routines of this sort are sometimes called "diagnostic routines." Of course it can be argued that they consume too much time, but this is not necessarily so; in most installations an hour or two of each day is devoted to tests of this sort and to various standardized electrical checks, and good operation is then generally obtained for the rest of the day. A degree of optimism is involved in reliance on checks of this kind, of course, and, clearly, a sudden breakdown in the future can hardly be expected to be foreseen.[1]

It is prudent to include also certain checks in the code of each problem. No general rule can be given with respect to these, but the programmer must be sufficiently astute to exploit the peculiar features of the problem under consideration. Thus, for example, in a problem concerning the behavior of a conservative dynamical system, the total energy may be computed from time to time; if it changes, some error has occurred. In other situations, the same number may be calculated in alternative ways and a comparison made. In many cases, checks of this kind can be included in the code of a problem without unduly lengthening it.

An alternative—or, better, a supplementary—method of checking is to incorporate certain checking circuits in the machine. These can take many forms. The simplest conceptually—and the most expensive method—is to duplicate certain parts of the equipment. For example, in some machines two arithmetic units are included; so each arithmetical operation is carried out twice in parallel, and the results are compared. A very simple method for checking transfers is to add to each word an extra "parity check," which is 0 or 1 according as the total number of 1's in the word proper is even or odd, or the reverse. At certain check points the number of 1's is counted, and a new parity-check bit is determined, which is compared with the one received with the word. If the old and new parity-check bits disagree, the machine is automatically stopped; this must be done both to prevent propagation of the error through the system and to facilitate error location. In UNIVAC I, both duplication of certain critical circuits and parity checking are provided,[2] and the

[1] Daggett and Rich, *op. cit.*

J. P. Eckert, Checking Circuits and Diagnostic Routines.

L. R. Walters, Diagnostic Programming Techniques for the IBM Type 701 EDPM.

G. Estrin, Diagnosis and Prediction of Malfunctions in the Computing Machine at the Institute for Advanced Study.

M. V. Wilkes and S. A. Barton, Experience with Marginal Checking and Routining of the EDSAC.

These papers are all in the Symposium on Diagnostic Programs and Marginal Checking for Large Scale Digital Computers, *IRE Convention Record*, 1953, pt. 7.

[2] Eckert, *op. cit.*, pp. 62–65.

equipment required constitutes about 30 per cent of the equipment in the central computer.

A more elaborate checking system was incorporated in RAYDAC. A certain number, the "weighted count" of the bits of the word in question, was generated for each word and transmitted with it. At certain points a new weighted count was determined and compared with that received with the word. The weighted counts also had certain simple arithmetical properties, which were exploited in checking arithmetical operations.[1]

The methods just described are all intended to detect malfunctions that already exist. Preventive maintenance procedures, on the other hand, are designed to forestall malfunctions. In most installations, an hour or two each day is devoted to various testing operations, including the running of diagnostic routines and also various electrical tests. At longer intervals the whole system is subjected to careful examination, during which adjustments are made and components regarded as suspicious are replaced. A very potent aid to preventive maintenance is "marginal checking," which was first evolved systematically during the development of Whirlwind I and has since been adapted to many machines.[2] The governing concept is very simple: the operating conditions of the circuits to be checked are worsened by varying the d-c supply voltages from their nominal levels, and this is continued until malfunction occurs, so that measures of the safety margins of the individual circuits are obtained directly. This is done in a systematic and automatic manner, the system being divided into about 200 sections which are checked individually in sequence. A day-to-day record is kept of the least satisfactory margins encountered; if an abrupt change in any circuit's margin is noted, it is subjected to careful investigation and the necessary steps (such as the replacement of components) are taken to restore the margin to a satisfactory value. If only a gradually decreasing margin is noted, nothing is done immediately, but during a regularly scheduled weekly maintenance period steps are taken to correct any such margins that are approaching the danger point. Marginal checking is most effective in detecting gradually deteriorating components before deterioration has gone so far as to cause malfunction. It cannot be expected to forestall troubles of an abrupt and catastrophic sort, like the sudden shorting of tube elements, but it can be used, once malfunction has been noted, as a means of diagnosis and fault location.

All the methods discussed have their advantages, of course. Certainly marginal checking in some form is useful, and certainly operating checks using diagnostic routines are of great value. Built-in checks, such as the parity check mentioned above, are very attractive provided they

[1] R. M. Bloch, R. V. D. Campbell, and M. Ellis, The Logical Design of the Raytheon Computer, *MTAC*, vol. 3, no. 24, pp. 286–295, October, 1948.

do not require excessive additional equipment. Well-designed and well-maintained systems that incorporate no special checking circuits at all seem to be giving just as good performance as those that incorporate fairly elaborate checking circuits. It can, however, be argued that, in the absence of checking circuits, it is necessary to use a good deal of programmed checking and that, therefore, the machine is used less efficiently than it would be otherwise.

CHAPTER 4

BASIC LOGIC CIRCUITS AND THEIR REPRESENTATION

4-1. Introduction. The circuits and components used in the physical realization of digital computers will now be discussed. These will be treated in Chaps. 4 to 10; then attention will be directed to their interconnection in building up the major units of the machine. The order to be followed is to a certain extent arbitrary. The exposition will start with the simplest "logic" (or switching) circuits, to which this chapter will be devoted. Chapters 5 and 6 will cover the basic memory elements. The natural course is then to show how the basic logic and memory elements can be used to build up the switching circuits, registers, counters, and adders in terms of which it is most convenient to describe the functioning of the machine. The treatment deals mostly, but not entirely, with circuits and components used in computers in current service. These machines are essentially based logically upon concepts developed in the 1940s, and for the most part use circuits and components either well understood or at least under development in that period. The first electronic computer, the ENIAC, used vacuum-tube circuits to perform all computer functions. Next came nonelectronic memory devices (e.g., acoustic delay lines) and then solid-state elements: crystal diodes, magnetic cores, transistors. Current machines use both vacuum tubes and solid-state elements. Machines of the early 1960s will presumably use solid-state elements exclusively.

4-2. General Remarks on Logic Circuits. It has been remarked in Chap. 1 that, in electronic digital computers, either the binary scale of notation is used for the representation of numbers or, if the decimal scale is used, each digit is represented by a binary-coded equivalent. This is because of the simplicity and engineering convenience obtained by using only bistable elements and circuits that need to discriminate between only two electrical states. Instructions are, of course, also encoded as collections of binary digits. Each binary digit may be represented either by a pulse of definite duration (or the lack of one) in a synchronous machine or by the electrical state of a wire; this has been discussed in Chap. 3, where the various modes of both storage and transmission of information within a machine have been distinguished.

Speaking quite generally, means are required (1) for storing information and (2) for operating upon information. Circuits used for these two functions can conveniently be called "memory" and "switching" (or "logic") circuits, respectively. The basic binary memory devices will be dealt with in Chap. 5. The present chapter will be devoted to the basic circuits that combine two (or more) bits of information to produce a third.

The basic circuits have a number of important direct applications, and from them it is possible to build up elaborate structures for such functions as (1) encoding and decoding, (2) performing arithmetical operations, and (3) controlling the flow of information. These higher-level combinations of the basic elements will be discussed in Chaps. 7 and 9.

4-3. The Basic Operations on Binary Variables. The operations to be considered are those affecting variables that take on but two possible values, 0 and 1, called "binary variables." The results of these operations are to be in every case variables taking on only these values. The

TABLE 4-1*

Input		Output							
x	y	0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	1	1	1
0 1	1 0	0	0	1	0	1	0	1	1
1	1	0	0	0	1	1	1	0	1

* F. C. Williams, A. A. Robinson, and T. Kilburn, Universal High-speed Digital Computers: Serial Computing Circuits, *Proc. IEE*, vol. 99, pt. 2, no. 68, pp. 107–123, April, 1952.

simplest case, of course, is that in which but one binary variable is to be operated upon. Consider the operations to be performed by a "black box" which has the variable as its input and the result as its output. There are but four possible cases: (1) the output is a constant 0 no matter what the input; (2) the output is a constant 1 no matter what the input; (3) the output is identical with the input; (4) the output is 0 if the input is 1, 1 if it is 0. With two inputs, x and y, there is an output for each possible combination of inputs (0,0), (0,1), (1,0), and (1,1). To each input must correspond an output 0 or 1. Since a single black box must give a unique set of four outputs, one for each of the possible inputs, there are, clearly, $2^4 = 16$ possible black boxes. This is the most general situation, which will be dealt with in Chap. 7. For present purposes, it is more convenient and perspicacious to require that in every case (0,1) and (1,0) shall give rise to the same output so that there are but $2^3 = 8$ possible black boxes, as listed in Table 4-1.

Examining this table, note that outputs 0 and 7 are constants, not depending upon the inputs. In column 3, an output of 1 arises only if both x and y are 1. In column 4, an output of 1 arises if either x or y is a 1. In column 5, an output of 1 is obtained if both the inputs are the same, but 0 is obtained if they differ. Such verbal descriptions could be continued, but it would be better to devise a concise symbolic way of describing the operations. Fortunately a mathematical science exists in the form of Boolean algebra,[1] which is ideally suited to the treatment of such ideas.

4-4. The Boolean Algebra. The algebra of logic, first worked out a century ago by George Boole,[2] was originally an attempt to deal symbolically with relations of inclusion and exclusion among classes of objects. Thus it actually was concerned with such ideas as (1) the "union" of two classes A and B (written $A \cup B$), or the class each member of which is a member of A, of B, or of both A and B; (2) the "intersection" of A and B (written $A \cap B$), or the class of objects each of which is a member of both A and B; and (3) the "complement" or "negative" of A (written $\bar{A}$), or the class all of whose members are outside A, but are members of the "universal set." These classes are also written "A or B," "A and B," and "not A," respectively, or "$A + B$," "$A \cdot B$," and "$-A$." Given these definitions, it is possible to develop a great number of theorems, which form the body of the algebra. This will be treated in some detail in Chap. 7; for present purposes only the fundamental ideas are needed.

It turns out also that a logic of propositions can be worked out along similar lines.[3] If p and q represent propositions, then $p \wedge q$ is true if p and q are both true, $p \vee q$ is true if either p or q is true, and $\sim p$ is true if p is false. To every p the value of 1 is assigned if p is true, or the value 0 if p is false. The logic of propositions is then a Boolean algebra having but two elements.

Here a two-element algebra is employed. Clearly, $1 \cap 1 = 1$, $1 \cap 0 = 0 \cap 1 = 0$, $0 \cap 0 = 0$; $1 \cup 1 = 1$, $1 \cup 0 = 0 \cup 1 = 1$, $0 \cup 0 = 0$; $\bar{1} = 0$, $\bar{0} = 1$. Also the "and" and "or" relations can be written: $1 \cdot 1 = 1, 1 \cdot 0 = 0 \cdot 1 = 0 \cdot 0 = 0, 1 + 1 = 1, 1 + 0 = 0 + 1 = 1$, $0 + 0 = 0$. Inspection of Table 4-1 shows that column 3 is $x \cap y$ and that column 4 is $x \cup y$. Column 1 is the negative of column 4, or $\overline{(x \cup y)}$, and column 6 is $\overline{(x \cap y)}$. Column 5 may be written "$x = y$," signifying

[1] See P. C. Rosenbloom, "Elements of Mathematical Logic," chaps. 1 and 2, Dover Publications, New York, 1950.

A useful summary is given by Robert Serrell, Elements of Boolean Algebra for the Study of Information-handling Systems, *Proc. IRE*, vol. 41, no. 10, pp. 1366–1380, October, 1953.

[2] G. Boole, "Investigation of the Laws of Thought," 1854. Dover Publications, New York, brought out a reprint in 1951.

[3] Rosenbloom, *op. cit.*, chap. 2.

that the output is 1 if the variables x and y have the same value but 0 if they do not, and column 2 may be written "x not $= y$."* In passing, it may be remarked that this last can be expressed as $\overline{(\bar{x} \cap \bar{y})} \cap \overline{(x \cap y)}$. In general, the most complicated propositions of Boolean algebra involve only "and," "or," and "not." In fact, two of these operations theoretically suffice; for example, "or" can be defined in terms of "and" and "not." The circuits used to realize these may now be examined.

4-5. Gates and Mixers ("And" and "Or"). Consider circuits having two inputs and a single output. A "logical-and" circuit will be called a "gate"; and a "logical-or" circuit will be called a "mixer." Each of these has a voltage level representing 0 and a voltage level representing 1. Inputs may be pulses of definite duration, or they may be temporarily fixed voltages. A similar statement is true of the output. Circuits actually in use exist in rather considerable variety; hence, a number of examples will be given. Crystal-diode gates and mixers are now standard, but earlier circuits used thermionic pentodes and triodes.

4-6. Pentode Gates. Figure 4-1 shows a pentode circuit.[1] The bias voltages $-E_{CC}$ and $-E_{SS}$ hold the tube safely off in the absence of signals

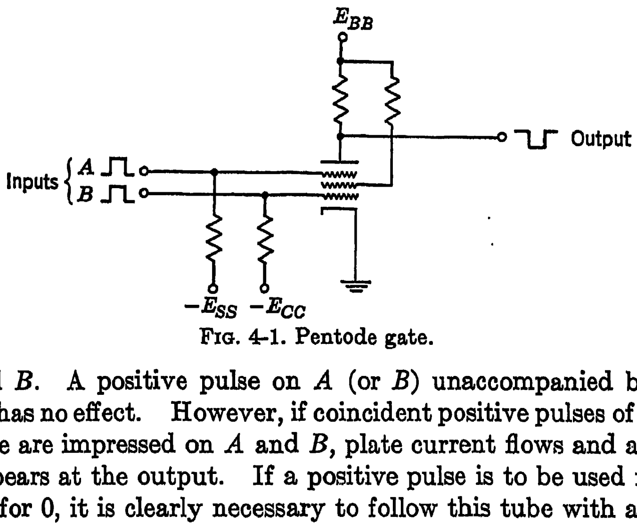


FIG. 4-1. Pentode gate.

at A and B. A positive pulse on A (or B) unaccompanied by one on B (or A) has no effect. However, if coincident positive pulses of sufficient amplitude are impressed on A and B, plate current flows and a negative pulse appears at the output. If a positive pulse is to be used for 1 and no pulse for 0, it is clearly necessary to follow this tube with another in order to invert the output and produce a "gate" or "logical-and" circuit in the usual sense of the term.[2] In the Whirlwind I computer a pulse-

* F. C. Williams, A. A. Robinson, and T. Kilburn, *Proc. IEE*, vol. 99, pt. 2, no. 68, p. 109, April, 1952.

[1] Britton Chance, F. C. Williams, V. W. Hughes, D. Sayre, and E. F. MacNichol, Jr., "Waveforms," p. 379, McGraw-Hill Book Company, Inc., New York, 1949.

[2] The circuit with inverter following the pentode is shown in Staff, Harvard Computation Laboratory, "Description of a Magnetic Drum Calculator," p. 48, Harvard University Press, Cambridge, Mass., 1952. The triode circuits of Sec. 4-7 are shown on the same page.

transformer output is used,[1] obviating the need for a pulse-inverting tube. The circuit is given in Fig. 4-2. The diode and resistor combination across the primary of the pulse transformer damps out any tendency of the transformer to ring and guarantees that only a single positive pulse will appear at the output when the circuit is properly excited. This circuit requires a sharp-cutoff pentode, such as the 7AK7, which was in fact developed for this application.

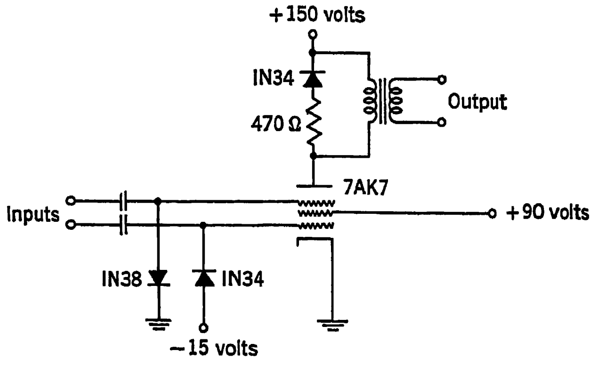


FIG. 4-2. Whirlwind I pentode gate.

4-7. Triode Gates and Mixers. A variety of forms of triode circuits will now be described. The Rossi circuit is shown in Fig. 4-3. In the absence of input pulses, both triodes are conducting and the plate voltage is low. If a sufficiently negative pulse is applied to A, the left triode is cut off, but because of increased conduction in the right triode, the plate voltage does not change much. If sufficiently negative pulses are applied to both A and B, however, both triodes are cut off, and the plate voltage rises to E_{BB} for the duration of the pulse. Again, as in the case of the pentode gate shown in Fig. 4-1, a reversal of polarity occurs, which must be corrected by subsequent inversion if two negative inputs are to result in a negative pulse. It should be remarked, however, that in many cases the pulse of opposite polarity at the output can be used directly without inversion, depending upon the circuit that follows the gate.

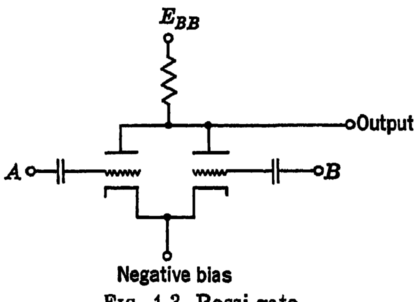


FIG. 4-3. Rossi gate.

By a simple change of bias, this circuit can be changed into a "logical-or" circuit or a "mixer." Suppose the inputs are positive pulses for 1's.

[1] C. A. Roland, Jr., An Investigation of the Possibilities for Improving Pentode Gate Tube Circuits, *Project Whirlwind Rept.* R-186, submitted under ONR contract N5ori60, MIT, Servomechanisms Laboratory, Apr. 26, 1950.

The negative cathode bias is replaced by a positive bias sufficient to hold both triodes cut off in the absence of inputs at A and B. In the quiescent state, the output voltage is E_{BB}. If a sufficiently positive pulse is applied at A (or B) the left (right) triode conducts, and the output voltage falls; positive pulses at both A and B have the same result. Thus the circuit is transformed into a mixer. Here also pulse polarity is reversed and must be corrected by a subsequent stage. The circuit of Fig. 4-3 is shown as having pulse inputs. This is not necessary. A direct-coupled version, with grounded cathode and input levels that are positive for 1 and negative for 0, is also in use.[1]

A very simple triode gate is used in the IAS computer;[2] it is shown in Fig. 4-4. This gate transmits information by drawing plate current so

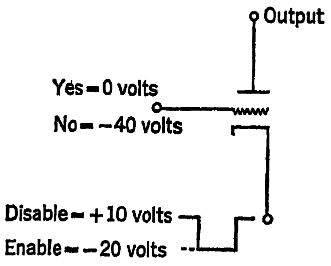


FIG. 4-4. IAS triode gate.

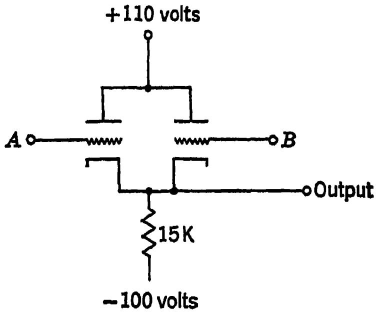


FIG. 4-5. IAS cathode-coupled triode gate.

that there must exist a resistive path between the plate and a source of positive voltage. The input to the grid is either 0 volts or −40 volts (1 or 0). The quiescent voltage of the cathode is 10 volts, which is ample to prevent conduction for either grid voltage. The gate is "enabled" by pulsing the cathode down to −20 volts, which may be considered as the 1 level as far as this element is concerned. During the enabling pulse, current is drawn if the grid voltage is 0 volts, but is not drawn if the grid voltage is −40 volts. It should be noted that this type of gate was designed specifically for use in direct-coupled circuits.

Another triode circuit,[3–5] also used in the IAS computer, is shown in Fig. 4-5. It consists of two cathode followers with a common cathode

[1] Staff, Harvard Computation Laboratory, "Synthesis of Electronic Computing and Control Circuits," p. 12, Harvard University Press, Cambridge, Mass., 1951.

[2] C. V. L. Smith, "Sixth Interim Progress Report on the Physical Realization of an Electronic Computing Instrument," p. 4, Institute for Advanced Study, Princeton, N.J., September, 1951.

[3] *Ibid.*, p. 6.

[4] Engineering Research Associates, Inc. (W. W. Stifler, ed.), "High-speed Computing Devices," p. 39, McGraw-Hill Book Company, Inc., 1950.

[5] Chance et al., *op. cit.*, p. 378.

resistor. As used there, the input levels are ± 10 volts. As the cathode follows the higher grid, it is clear that, if either A, B, or both are $+10$ volts, the output is also $+10$ volts (plus a slight cathode rise, which can be neglected). If both A and B fall to -10 volts, so does the output. If $+10$ volts represents 1, and -10 volts represents 0, then the circuit acts as a mixer. If these polarities are reversed, then the circuit is a gate. In any case, the circuit preserves polarity.

In relation to triode gates, another circuit should be considered. It consists of two triodes in series. The first has its cathode grounded and its plate connected to the cathode of the second. The plate of the second is connected through a resistor to the plate supply. If both grids are kept sufficiently low, the plate voltage of the second tube is the supply voltage as both tubes are cut off. For current to flow through the resistor, it is necessary for both grid voltages to be raised; that of the first tube must be brought up toward ground and that of the second must be made considerably positive. If this is done, the plate voltage of the second tube falls. If it is desired to pass positive pulses, this circuit must be followed by an inverter. This circuit was used successfully in the ACE computer.[1]

4-8. Diode Gates and Mixers. It has been mentioned above that most designers today use solid-state diodes for their gate and mixer circuits. Such elements are very attractive, of course, because they consume no cathode power and generate negligible heat. Furthermore, the circuits are very simple. There are, however, certain difficulties that can plague the unwary user. One of these is the unfortunate transient effect: immediately after a pulse of current is transmitted, the back resistance falls rather spectacularly for a short time, which may be as short as a fraction of a microsecond but may also be much longer. In any case, this should be taken into consideration. It has made unusable diodes which were extremely good in other respects such as low forward and high back resistance, and stability (except for the transient effect!) of back-to-front resistance ratio. It is also a mistake to design circuits that rely heavily on an optimistic view of the maintenance of high back resistance or that permit the rated inverse voltage to be applied to the diode. Conservatism and caution pay large dividends, and properly designed solid-state-diode logic circuits are very satisfactory.

The first example is a gate for positive signals, originally devised for thermionic diodes.[2] Suppose that 0 is represented at the input by 0 volts and 1 by some positive voltage. If A is positive but B is 0 volts, then conduction by D_1 holds P to a small positive voltage, D_2 does not conduct,

[1] A. D. Booth and K. H. V. Booth, "Automatic Digital Calculators," p. 85, Butterworth & Co. (Publishers) Ltd., London, 1953.

[2] Chance et al., *op. cit.*, p. 370.

and the output is 0 volts. Finally, if both A and B are positive, a positive voltage must appear at the output.

The circuit of Fig. 4-6 can give an output equal to B only if A is larger than B. The circuit of Fig. 4-7[1-3] avoids this objection. Suppose 1 is indicated by V_1 and 0 by V_0, where $V_0 < V_1 < E_{BB}$. If at least one of A and B is V_0, then the output voltage is V_0 plus the small drop in the diode or diodes. If both A and B are V_1, then the output voltage is V_1 plus the small drop in the diodes. Thus the circuit is a gate for positive signals, either pulses or temporarily fixed voltages. The difference in voltage between the 1 and 0 outputs is essentially the same as that between the 0 and 1 inputs.

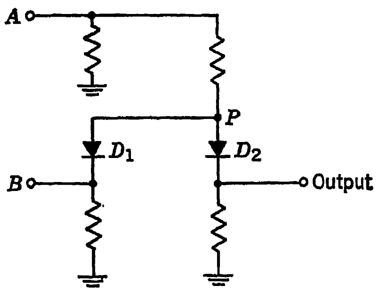


FIG. 4-6. Diode gate (positive signals).

The circuit of Fig. 4-7 also serves as a mixer for negative-going signals. Suppose that $V_1 < V_0 < E_{BB}$; then if V_1 is impressed on A, B, or both, the output level is slightly above V_1. Only if both inputs are V_0 does the output level rise to slightly above V_0.

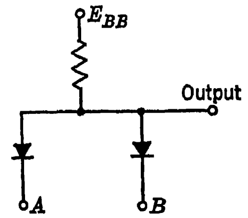


FIG. 4-7. Diode gate (positive signals).

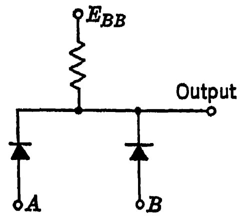


FIG. 4-8. Diode gate (negative signals).

A gate for negative-going signals or a mixer for positive-going signals is obtained simply by turning the diodes around and making the supply voltage low, as in Fig. 4-8. If V_1 and V_0 represent 1 and 0, respectively, then if $E_{BB} < V_1 < V_0$, a gate is obtained, and if $E_{BB} < V_0 < V_1$, a mixer.

4-9. The Negative. The treatment of the negative in the logical sense depends upon the manner in which information is represented within the machine. First consider the case of asynchronous direct-coupled com-

[1] Tung Chang Chen, Diode Coincidence and Mixing Circuits in Digital Computers, *Proc. IRE*, vol. 38, no. 5, pp. 511–514, May, 1950.

[2] C. H Page, Digital Computer Switching Circuits, *Electronics*, vol. 21, pp. 110–118, September, 1948.

[3] "Synthesis of Electronic Computing and Control Circuits," chap. 10. The simple circuits shown here are given on p. 130.

puters. In the IAS computer, for example, information that is to be operated upon is stored in toggles; the output is taken from a grid, 0 volts for 0, -40 volts for 1, and the voltage at the opposite grid gives the logical complement or negative of this. In other situations it may be necessary to use an inverter, but then the inversion takes place at a higher voltage level and some means must be employed of translating the voltages so obtained to the desired levels.

In a serial synchronous computer, some such scheme as that shown in Fig. 4-9 can be used. The stage labeled "I" is a simple inverter. It is assumed that information and clock pulses start from 0 volts and rise to $+E$ volts. In the absence of an input pulse, the clock pulses appear at the output. However, if a positive pulse appears at the input, the voltage at point P is driven down to 0 volts, and a clock pulse received at that time does not appear at the output. The pair of diodes D_1 and D_2 constitute a gate for positive pulses, but D_3 merely "bumps" the negative

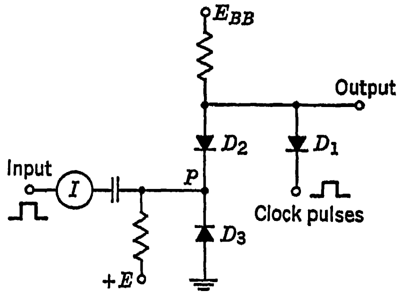


FIG. 4-9. "Not" circuit.

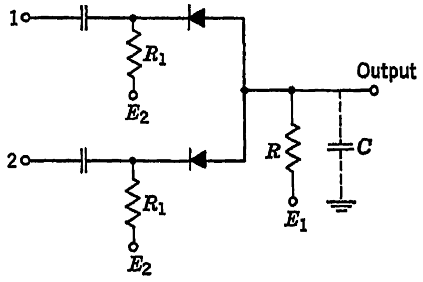


FIG. 4-10. Two-input gate, showing distributed capacitance.

swing of point P at ground. Thus a 1 at the input can be regarded as preventing, or "inhibiting," the passing of the clock pulse by the gate circuit.

4-10. Design of Diode Gates. So far, crystal gating structures have been considered only from an ideal point of view, and their actual operation has been neglected. In view of the practical importance of circuitry of this type, it is important for the reader to realize the need for careful electrical design. In the present section, a simple circuit will be discussed with some degree of realism.

Consider a simple crystal gate for two inputs; suppose that information is conveyed by positive rectangular pulses, and that, as frequently occurs, it is necessary to provide d-c isolation. The circuit then assumes the form shown in Fig. 4-10. The presence of the coupling capacitors at the input makes it necessary to add the resistors R_1 and the source $E_2 < E_1$. The dashed capacitor C represents the capacitance from the output line to ground. If no inputs appear at 1 and 2, the output voltage remains at very nearly

$$\frac{R_1E_1 + 2RE_2}{R_1 + 2R}$$

but if one input is pulsed high enough to keep its diode cut off, the output changes exponentially toward

$$\frac{R_1E_1 + RE_2}{R_1 + R}$$

for back resistance of the cutoff diode is assumed to be infinite. Finally, if both inputs are pulsed positive to a value $E \leq E_1$, then the output changes exponentially toward this value.[1] If there were n inputs instead of two, there would be $n - 1$ output levels

$$\frac{R_1E_1 + kRE_2}{R_1 + kR} \qquad k = 1, 2, \ldots, n - 1$$

corresponding to 1, 2, . . . , $n - 1$ positive inputs, and for n inputs the value would be E, supposing as before that $E \leq E_1$. It is evident that the degree of discrimination between coincident and noncoincident inputs is not very good. This situation can be remedied by a very simple expedient, shown in Fig. 4-11, where an extra diode has been added, and a source E_0 such that $E_2 < E_0 < E_1$ and $E_0 < E$. The output voltage now remains at E_0 volts (minus a small diode drop) if zero or one positive pulse signal appears at the input. If both inputs are pulsed simultaneously, then the output condenser starts to charge toward E_1 volts and the output voltage becomes

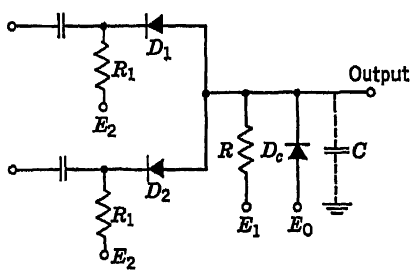


FIG. 4-11. Gate with clamping diode on output. (*Tung Chang Chen, Proc. IRE, vol.* 38, *no.* 5, *pp.* 511–514, *May,* 1950.)

$$E_1 - (E_1 - E_0)e^{-t/RC}$$

if the capacitance of the diodes D_1 and D_2 is neglected and infinite back resistance is assumed for D_c. This mode of variation persists until the value E is reached, provided of course that the pulses last long enough. In this case, the time taken by the output in rising from E_0 to E is given by

$$RC \ln \frac{E_1 - E_0}{E_1 - E}$$

where the effect of decay of the voltages applied to D_1 and D_2 is neglected

[1] It should be clear that if $E > E_1$ and if both inputs are energized, the output will rise exponentially to E_1.

because of the finite values of input time constants. The output pulse now has an exponentially rising leading edge in place of a vertical one. This will not be serious if the pulse duration is several times the length of the rise time. On the other hand, a short pulse would be badly distorted in passing through the gate.

Fall time can be calculated by assuming the output capacitance C charged to E volts and then removing the exciting pulses. The calculation will not be given, but it should be observed that the time constant of fall must be less than the time constant of rise, since the resistance involved is now essentially the parallel combination of R, R_1, and R_1.

So far nothing has been said about the capacitances of the input diodes. If these (each of which will be called C_d) are comparable to C, then the rise time of the output is given by

$$(nC_d + C)R \ln \frac{E_1 - E_0 - nC_d\,(E - E_0)/(C + nC_d)}{E_1 - E}$$

if it is supposed that the gate circuit has n inputs. There is no point in considering the influence of C_d upon fall time, for, when all the inputs are removed, the input diodes all conduct and their small forward resistances effectively short-circuit their capacitances.

No consideration has been given to the effect of the transient lowering of back resistances of crystal diodes that have been in a conducting state when the polarity of the voltage across them is suddenly reversed. Clearly, the intensity and duration of this effect should be known, since if input pulse durations are very short, this effect causes the output of the circuit to degenerate still further.

4-11. More than Two Inputs: Inhibition. So far, mixers and gates having but two inputs have been considered. Most of the circuits are

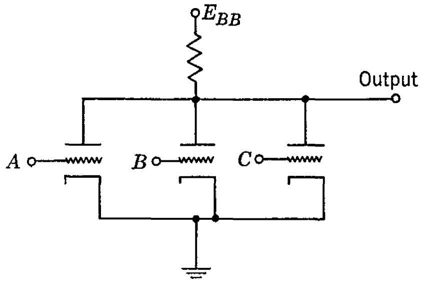


FIG. 4-12. Three-input triode gate (negative-going signals).

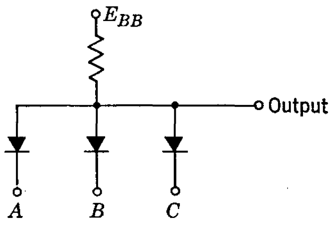


FIG. 4-13. Three-input diode gate (positive-going signals).

simply extended in the obvious way to form $A \cap B \cap C \cap \cdots \cap N$ and $A \cup B \cup C \cup \cdots \cup N$. Two examples will be given. If 1 is represented by a negative signal and 0 by 0, the direct-coupled version of the Rossi circuit becomes the circuit shown in Fig. 4-12. This is also a mixer for positive-going signals. In many applications a subsequent inverter is required to restore polarity. The cathode-coupled gate of Fig. 4-5 is similarly extended to more than two inputs. For diodes, a gate for positive-going signals is given in Fig. 4-13; polarity is of course preserved. A combination of a gate and a not-A circuit (see, for example, Fig. 4-9) gives a circuit that forms $B \cap C$ if $A = 0$, but 0 if $A = 1$. The input A is then said to inhibit the gate.

4-12. The Harvard Vacuum-tube Operators. Now that the usual types of gate and mixer circuits have been reviewed and the utility of Boolean algebra in representing symbolically the operations performed by such circuits has been pointed out, it is worthwhile to direct the reader's attention to the equivalent algebra devised at the Harvard Computation Laboratory by Professor H. H. Aiken and his associates.[1] This algebra possesses a certain advantage in that it is developed entirely in terms of arithmetical operations on variables whose values are binary numbers and that it is not necessary to introduce the notions of logic. Also, the development is carried out in terms of "vacuum-tube operators" and "switching functions," so that the learner retains contact with familiar language and, presumably, finds it easier to pick up the new ideas.

Consider direct-coupled circuits whose inputs can assume only two values, a high and a low, and whose outputs are necessarily similarly restricted. At both the input and the output, assign the numerical value 1 to the high value and 0 to the low value. This is not to be construed as implying that the high value of voltage at the input is the same as the high value at the output; no such assumption is made.

Consider first the simple case of an inverter, a triode with resistive load. Let e_1 be a binary variable expressing the input, and let the output be designated by the functional expression $T_1(e_1)$. Since this must take on values (1, 0) according as the values of e_1 are (0, 1),

$$T_1(e_1) = 1 - e_1 \qquad (4\text{-}1)$$

This function is the "vacuum-tube operator" describing such a stage.

Now consider the pentode circuit of Fig. 4-1, and consider the direct-coupled case where the capacitors have been eliminated. If the corresponding operator is called $P_2(e_1,e_2)$, it is clear that the following expression is valid:

$$P_2(e_1,e_2) = 1 - e_1e_2 \qquad (4\text{-}2)$$

This emphasizes the fact that the circuit as it stands is not a gate, even though it detects the coincidence of the high values of e_1 and e_2.

The direct-coupled version of the Rossi circuit (a mixer for positive signals) is described by the operator

$$T_2(e_1,e_2) = (1 - e_1)(1 - e_2) = e_1'e_2' \qquad (4\text{-}3)$$

[1] "Synthesis of Electronic Computing and Control Circuits."

where $e' = 1 - e$; and, if n triodes are used,

$$T_n(e_1,e_2, \ldots ,e_n) = (1 - e_1)(1 - e_2) \cdots (1 - e_n) = e_1'e_2' \cdots e_n' \tag{4-4}$$

The cathode-coupled circuit of Fig. 4-5 is described by

$$C_2(e_1,e_2) = 1 - e_1'e_2' \tag{4-5}$$

Finally, in the diode circuits, the circuit of Fig. 4-7 is described by the operator

$$F_2(e_1,e_2) = e_1e_2 \tag{4-6}$$

4-13. Transistor Logic Circuits. Transistor logic circuits exist in considerable variety, depending upon the type of transistor employed and the mode of operation.[1] As a number of these are merely combinations of several emitter followers, the emitter-follower circuit will be treated first. In Fig. 4-14 are shown circuits of this type using a *pnp* and *npn*

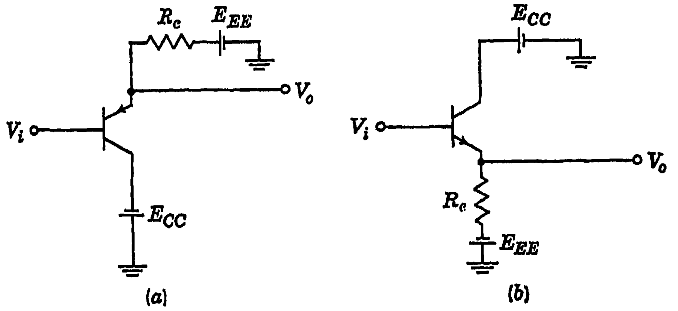


Fig. 4-14. Emitter followers. (*a*) *pnp*, (*b*) *npn*.

junction transistor. Suppose, in the *pnp* case, that the quiescent value of V_i is 0 volts, so the emitter-to-base junction is biased in the forward direction. Suppose also that the total excursion of V_i and the supply voltages E_{EE} and E_{CC} are so chosen that the transistor always functions in its active region except perhaps for very brief periods under transient conditions. Finally, suppose that E_{CC} is somewhat (about 1.5 volts) more negative than the negative excursion of V_i, to minimize the storage of minority carriers. If, now, V_i is made positive, the amount of forward bias is decreased and the emitter current is decreased so that V_0 increases. On the other hand, if V_i is made negative, then the amount of forward bias is increased and V_0 swings in the negative direction. Indeed, it can be shown that V_0 is a linear function of V_i and E_{EE}; the coefficient of E_{EE} is in general negligible if the resistor R_e is of the order of a few thousand kilohms, and the coefficient of V_i is of the order of unity. The term pro-

[1] J. C. Logue, Transistor Switching Circuits, in Lloyd P. Hunter, "Handbook of Semiconductor Electronics," sec. 15, pp. 15–34, McGraw-Hill Book Company, Inc., New York, 1956.

portional to E_{BB} represents a shift in reference level of the output as compared to the input, and can be compensated for by inserting a resistor between V_i and the base and then connecting the base through another resistor to a negative voltage source, the resistors being chosen to form a voltage divider which, under quiescent conditions, holds the base sufficiently negative to keep V_0 at 0 volts. The transient response of the circuit is complicated by the existence of a relatively large emitter-to-base capacitance, which necessitates some care in choosing the resistor in series with the base so that the output does not rise too slowly. The paralleling of this resistor by a capacitor also improves response by speeding up the charging of the emitter-to-base capacitance at the beginning of any sudden change in V_i. In Fig. 4-14*b* is shown the emitter-follower circuit using an *npn* transistor. There is no need for a separate discussion of this circuit.

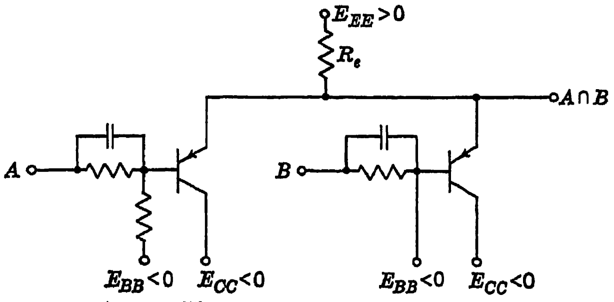


FIG. 4-15. *pnp* gate circuit.

Using the *pnp* emitter follower as a building block, a "logical-and" circuit, or gate, is easily constructed. The circuit shown in Fig. 4-15 detects the coexistence of positive-going excursions of A and B. Suppose that 0 levels of both A and B are, say, 0 volts, and the 1 levels are a few volts positive. If A and B are both 0, then the output must be 0 volts and, therefore, represents 0. If $A = 1$ and $B = 0$, the output is held down to 0 volts by B and, hence, represents 0; and the same reasoning shows that if $A = 0$ and $B = 1$ the output must be 0. Finally, if both $A = 1$ and $B = 1$, the output must rise to a positive value approximately equal to that of A and B, and the output represents 1. Hence the circuit does in fact generate $A \cap B$.

The *npn* emitter follower similarly serves as a building block from which a "logical-or" circuit, or mixer, can readily be constructed, as shown in Fig. 4-16. Notice that if A, say, is raised above B, the bias of the emitter-to-base junction is increased, more current flows through R_e, and the output follows A. The same is true if B rises above A, or if both rise together. Hence, as stated, the circuit is a mixer for positive-going signals.

If the A and B signals are negative-going to represent 1, then the circuits shown become a mixer and a gate, respectively.

To complete a stock of logical building blocks, a "not" circuit or inverter is needed, which exchanges signal levels and, hence, logically replaces 0 by 1 and 1 by 0. Simple *pnp* and *npn* configurations are shown in Fig. 4-17; these are both grounded-emitter amplifiers. Minority-carrier storage can be eliminated by preventing saturation, which can be accomplished, for example, by clamping the collector by means

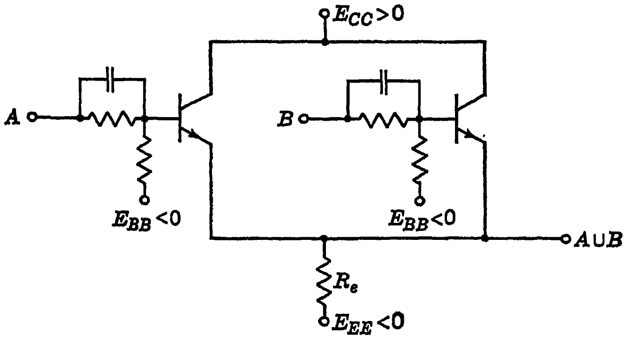


Fig. 4-16. *npn* mixer (positive-going signals).

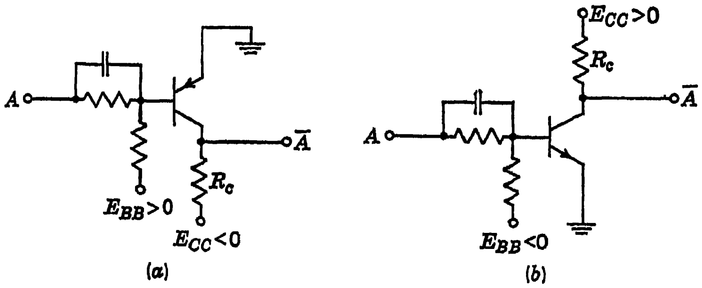


Fig. 4-17. Inverters. (*a*) *pnp*, (*b*) *npn*.

of a diode to a suitable voltage; in the circuit of Fig. 4-17*a* this would be a few volts negative.

Some very simple logic circuits based upon the grounded-emitter inverter employ surface-barrier transistors (e.g., the Philco SB100).[1] These are junction transistors of a special type, in which the base layer is very thin. Consequences of this are (1) that they are intrinsically low-voltage devices, using supply voltages of from -1.5 to -3 volts, and (2) that they are very fast in operation, owing to high alpha-cutoff frequency

[1] R. H. Beter, W. E. Bradley, R. B. Brown, and M. Rubinoff, Surface Barrier Transistor Switching Circuits, *IRE Convention Record*, 1955, pt. 4, pp. 145–148.

and small minority-carrier storage time. In the grounded-emitter circuit small variations in base voltage (say from ground to a few tenths of a volt negative) causing proportional variations in base current (say from a few to a few hundred microamperes) cause rather large variations in collector current (say from a few microamperes to more than 1 ma) accompanied by but small variations in emitter-to-collector voltage. For this reason, the grounded-emitter circuit of this type has been said[1] to operate "in the current mode," and is best thought of as a current amplifier, as distinguished from the circuits described above, which use ordinary junction transistors and which can be described as operating "in the voltage mode." In the present case, then, one can regard the current as the means of conveying information. Circuits of this type are obviously very suitable if direct coupling is to be used. They have the

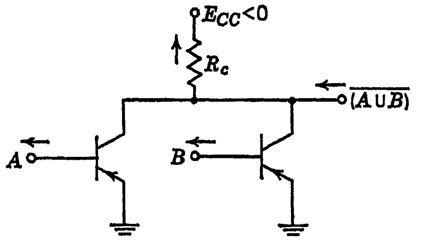


FIG. 4-18. Inverting mixer using surface-barrier transistors.

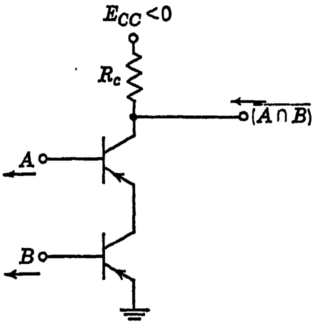


FIG. 4-19. Inverting gate using surface-barrier transistors.

advantage of simplicity, for few circuit elements besides the transistors are required and but a single power supply (-1.5 volts for the Philco SB100). They are also quite fast in transient response, as noted above; rise time is less than 0.1 μsec and fall time about 0.1 μsec. They possess the disadvantage that the outputs are inverted.

Consider now the circuit of Fig. 4-18. Suppose that the positive directions of current flow at inputs and output are as given by the arrows and that positive current flow is taken to correspond to $A = 1$, zero current flow to $A = 0$. If (*a*) current is drawn from either or both of the bases, then current flows up through R_c to E_{CC}, which, as noted above, may be taken as -1.5 volts; but if (*b*) no current is drawn from either base, no current flows from the collectors through R_c. Hence, if the output terminal is connected to the base of a following grounded-emitter circuit and if R_c is so chosen that, when either of the transistors conducts, the common-collector voltage rises to ground (that is, $R_c = 1{,}000$ ohms if the

[1] Logue, *op. cit.*

transistors are Philco SB100's), then it follows that in case a no current flows into the output terminal, but in case b it does. Thus, logically the current entering the output terminal represents $\bar{A} \cap \bar{B}$ or $\overline{(A \cup B)}$. In terms of voltage inputs, with 0 volts corresponding to 0 and a small negative voltage (three or four tenths of a volt) corresponding to 1, then the output level is high if one or both inputs are negative, but low if both are at ground.

An inverting gate is shown in Fig. 4-19. If current is drawn from both bases, if E_{CC} and R_c are properly chosen ($E_{CC} = -1.5$ volts and $R_c = 1{,}000$ ohms for SB100's), and if the output terminal is connected to the base of a grounded-emitter transistor, then no current flows into the output terminal, whereas if no current is drawn from at least one base, then no collector current flows through R_c and the source draws current in through the output terminal. Thus the current flowing into this terminal logically represents $\overline{(A \cap B)}$.

4-14. Magnetic-core Logic Circuits. Certain ferromagnetic materials have nearly rectangular hysteresis loops, characterized by fairly low coercive force and high remanence. These properties were exploited by Aiken and his associates at the Harvard Computation Laboratory to produce simple, rugged, and reliable bistable devices. This work was concerned first with memory devices such as shifting registers, which will be described in Chap. 5. The same materials can be used to produce logic circuits, and methods of doing this will be described in this section. Later on it will be shown how the same basic ideas have led to the principal type of large-capacity low-access-time random-access memory used in modern high-performance computing machines.

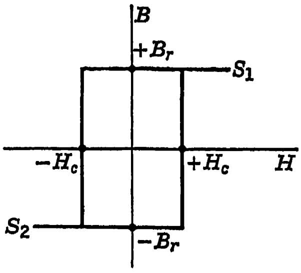


FIG. 4-20. Ideal hysteresis loop.

In Fig. 4-20 is shown an idealized hysteresis loop, taken as rectangular and symmetrical with respect to the origin. The positive value H_c for which $B = 0$ is called the "coercive force," and the value $+B_r$ for which $H = 0$ is called the "remanence," or "residual induction." S_1 and S_2 indicate positive and negative magnetic saturation. If a magnetizing force of magnitude greater than H_c is impressed upon a core of the material by the flow of current through a suitable winding on that core, then the residual induction after the removal of the unidirectional current is either $+B_r$ or $-B_r$, depending upon the direction of the impressed magnetizing force and the sense of circulation about the core taken as positive. This makes it possible to store binary information and also to use combinations of the cores to generate logical functions.

It must, of course, be understood that actual magnetic materials do not possess ideal hysteresis loops but loops exhibiting nonvertical sides and rounded corners, pretty much like the one exhibited in Fig. 4-21. Certain oriented silicon steels and a variety of iron-nickel alloys possess such hysteresis curves, more or less well approximating the ideal, and certain ferrites, which are materials having a crystal structure like that of spinel ($MgAl_2O_4$), an example being $MnFe_2O_4$, also approach it fairly well.[1] Cores of the steels are usually formed by wrapping an ultrathin ribbon of the material on a ceramic bobbin. The ribbon may be (e.g.) of molybdenum Permalloy, as thin as 1/8 mil,* with the outside diameter of the completed core perhaps 1/4 in. The ferrite cores are pressed in the shape of tiny doughnuts of outer diameter 1/10 in. or thereabouts.

The simplest imaginable magnetic-core logic circuit consists of four windings upon a single core, two being inputs, one an output, and the fourth used to prepare the core by driving it to negative saturation before information pulses arrive at the inputs. When the core is so prepared, if either or both inputs are excited by current pulses strong enough to drive the material of the core to positive saturation, then the large change of flux through the output winding will cause a voltage pulse to appear across its terminals. Hence this circuit functions as a mixer. For the case in which the two input pulses follow one another a gate can be devised if the windings are so arranged that, when the preparatory pulse has set the core to negative saturation, the first pulse, if it arrives, drives the core to positive saturation; here it would be necessary to sample the first output during the second information pulse. A modification of this idea, using two cores connected in such a way as to compensate for less-than-ideal hysteresis loops, has been suggested.[2] An undesirable feature of the simple gate described is that several voltage pulses appear across the output winding before the desired one, and steps have to be taken to suppress or ignore them.

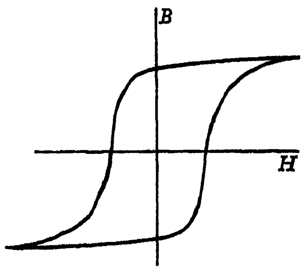


FIG. 4-21. Actual hysteresis loop.

A single three-legged core carrying one winding upon the center leg and two other windings and operated as a saturable-core transformer can be used as a gate, provided the core material possesses high incremental permeability at remanence and, hence, possesses a far-from-ideal hystere-

[1] R. M. Bozorth, "Ferromagnetism," chaps. 5 and 7, D. Van Nostrand Company, Inc., Princeton, N.J., 1951.

* D. Loev, W. Miehle, J. Paivinen, and J. Wylen, Magnetic Core Circuits for Digital Data Processing Systems, *Proc. IRE*, vol. 44, no. 2, pp. 154–162, February, 1956.

[2] Booth and Booth, *op. cit.*, p. 88.

sis loop.[1] This arrangement also requires that the two inputs be received in sequence. A saturable reactor can be used in series with the load if the pulses applied to it are derived from a low-impedance source;[2] this arrangement will work if the two information signals are simultaneous, although the enabling signal applied to the control winding should be of longer duration than the pulse that is to be granted or denied transmission.

Of more interest in view of current applications are multicore logic circuits. These exist in a variety of forms, of which several will be presented here.[3–5] These arise because of the essentially sequential mode of operation imposed by the fact that a core, once driven to saturation in one sense, must remain in that condition until it is driven to saturation in the other sense; the manner in which the designer accommodates himself to this fact depends upon the modes in which information is delivered to the circuit and in which the output is obtained.

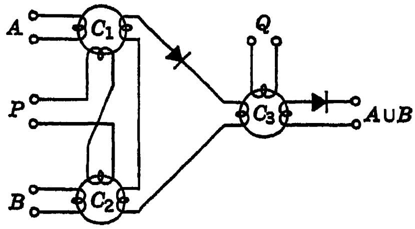


Fig. 4-22. Mixer operated by two "advance" pulse sequences.

Consider Fig. 4-22, which shows one form of mixer. Assume that, before the receipt of any information, the three cores are all in the 0 (e.g., negative-saturation) state. The windings driven by the inputs A and B are so arranged that standard (e.g., positive) pulses arriving on these drive the cores to their 1 state, and the series diode prevents the flow of current when the transition from 0 to 1 is made. After A and B pulse times, a pulse P, called in the common terminology an "advance" pulse, drives both cores back to the 0 state; if either or both hold a 1, the transition produces an output of such a sense that current passes freely through the diode and through the input winding of C_3, and the magnitude of this current is sufficient to drive C_3 to the 1 state. Of course, if neither A nor B were energized, no such energizing of C_3 would take place. The diode in the output of C_3 is so arranged that it passes current only when C_3 switches from 1 to 0; hence no output occurs when and if C_3 is switched to 1. To obtain an output, a pulse is applied to Q, of such amplitude and sign that C_3 is driven to the 0 state. Hence, if an input is received at

[1] *Computer Components Fellowship Quarterly Rept.* 10, sec. 3, The Mellon Institute of Industrial Research, Pittsburgh, 1953. See also *Quarterly Rept.* 12, sec. 7, pp. 7–10.

[2] *Computer Components Fellowship Quarterly Rept.* 11, sec. 4, p. 3.

[3] S. Guterman, R. D. Kodis, and S. Ruhman, Circuits to Perform Logical and Control Functions with Magnetic Cores, *IRE Convention Record, 1954*, pt. 4, pp. 124–132.

[4] A. J. Lincoln, Ferromagnetic Core Logical Circuity and Its Application to Digital Computers, *BRL Memo Rept.* 911, Ballistic Research Laboratories, Aberdeen, Md., August, 1955.

[5] Loev, Miehle, Paivinen, and Wylen, *op. cit.*

either A or B, then an output is transmitted when C_3 is switched back to 0 by the second advance pulse, and the circuit functions as a mixer, as stated. Notice also that all cores are finally in the 0 state and that the circuit is thus prepared for another cycle of operation. The analysis shows that there must be two distinct sequences of advance pulses to operate a logic circuit of this kind.

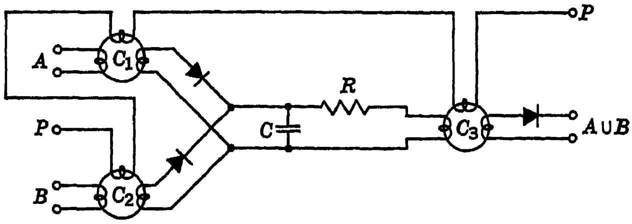


FIG. 4-23. Mixer operated by a single "advance" pulse sequence. (*S. Guterman, R. D. Kodis, and S. Ruhman, IRE Convention Record, 1954, pt.* 4, *pp.* 124–132.)

It is also possible to use but a single advance pulse sequence; a circuit by which this can be accomplished is shown in Fig. 4-23. Assume that all three cores are in the 0 state and that inputs A and B cause one at least of C_1 and C_2 to assume the 1 state. Now let a pulse drive the line P in such a way that all cores are driven to negative saturation, or 0. The output pulse or pulses charge C rapidly through the low forward diode resistance, and C then discharges more slowly through R and the input winding of C_3, driving C_3 to the 1 state. The next advance pulse finds C_1 and C_2 already in the 0 state and so does not affect them, but drives C_3 back to 0, producing an output pulse, which this analysis shows must represent $A \cup B$.

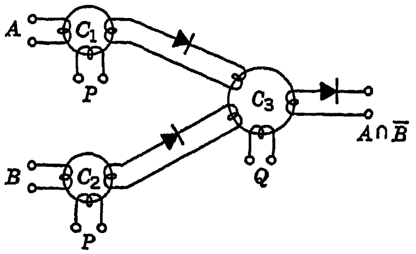


FIG. 4-24. Inhibitor. (*A. J. Lincoln, BRL Memo Rept.* 911, *p.* 46, *Ballistic Research Laboratories, Aberdeen, Md., August,* 1955.)

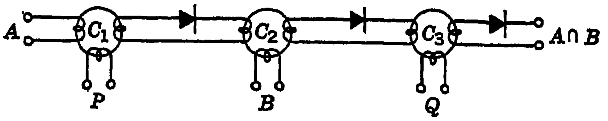


FIG. 4-25. Gate. (*A. J. Lincoln, BRL Memo Rept.* 911, *p.* 48, *Ballistic Research Laboratories, Aberdeen, Md., August,* 1955.)

Two other examples of circuits using the two sequences of advance pulses are shown in Figs. 4-24 and 4-25. The first of these produces $A \cap B'$, and hence illustrates inhibition. Observe that, if A is a continuous train of pulses, so that C_1 is always driven to 1 after the advance pulse has set it to 0, the circuit is a complementer or inverter, generating

$\bar{B}$ from B. The means of accomplishing this is the additional winding on C_3, so arranged that the signals from C_1 and C_2 produce opposing magnetizing forces, which cancel out.

Finally, in Fig. 4-25 appears a gate. Its principle of operation is simple. Suppose $A = 1$. The output from C_1 so drives C_2 that, when C_1 is switched from 0 to 1 by a P pulse, C_2 is driven to 0. Hence, if $B = 0$ there is no output from C_2, but if $B = 1$, such an output exists and drives C_3 to 1, and the subsequent advance pulse Q drives C_3 back to 0 and produces an output. Clearly such an output is generated if and only if $A = B = 1$, and the circuit is a gate, as described.

4-15. Other Logic Elements. It should also be possible to use the ferroelectric materials in constructing gates. No use of these in computer circuits is known to the author except in some experimental matrix memories. Discussion of these materials is accordingly deferred to Chap. 5, where binary memory elements are treated.

If the diodes in a simple mixer circuit are replaced by ordinary resistors,[1] as in Fig. 4-26, and if the inputs have the possible levels 0 volts (for 0) and E volts (for 1), then the output voltage E_0 can assume the four possible levels:

$$0 \qquad \frac{RE}{R_1 + 3R} \qquad \frac{2RE}{R_1 + 3R} \qquad \frac{3RE}{R_1 + 3R} \tag{4-7}$$

If $R_1 = R$, these become

$$0 \qquad 0.25E \qquad 0.5E \qquad 0.75E \tag{4-8}$$

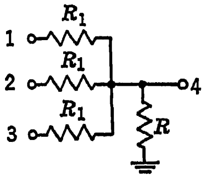


FIG. 4-26. Circuit using nonlinear resistors.

For the general case of n inputs and for $R_1 = R$, the possible levels of E_0 become

$$0 \qquad \frac{E}{n+1} \qquad \frac{2E}{n+1} \qquad \cdots \qquad \frac{nE}{n+1}. \tag{4-9}$$

and the discrimination becomes worse with increasing n. If the resistors are replaced by nonlinear resistors having a common volt-ampere characteristic curve given by

$$i = kv^n \tag{4-10}$$

the situation is quite different. A very simple analysis shows that, in the three-input case under consideration, the possible output voltages are

$$0 \qquad \frac{E}{1 + \sqrt[n]{3}} \qquad \frac{E}{2} \qquad \frac{\sqrt[n]{3}\,E}{1 + \sqrt[n]{3}} \tag{4-11}$$

For thyrite, for example, $4 \leq n \leq 7$. Hence, taking $n = 4$, the levels become

$$0 \qquad 0.43E \qquad 0.5E \qquad 0.57E \tag{4-12}$$

[1] *Computer Components Fellowship Quarterly Rept.* 9, pt. 2, 1953.

and, for $n = 7$, they are

$$0 \qquad 0.46E \qquad 0.5E \qquad 0.54E \tag{4-13}$$

In general, if there are M inputs and N are energized, then the output voltage is

$$\frac{E}{1 + A_n} \tag{4-14}$$

where
$$A_n = \left(\frac{1 + M - N}{N}\right)^{1/n} \tag{4-15}$$

The three-input case just cited, using nonlinear resistors with $n = 4$, would serve as a reasonably good mixer, but would have to be followed by an amplifier to restore the signal level.

Granular aggregates of silicon carbide show the type of nonlinearity discussed above. This material is easy to handle. It can be formed in a sheet after being mixed with some suitable bonding material such as rubber, and electrodes can be sprayed on where desired. As long as the electrodes are kept apart by about three times the thickness of the sheet, there is substantially no leakage between them. This technique makes it possible to make large arrays of logic elements in very compact form. Unfortunately, tests appear to show that waveforms become badly distorted when repetition rates of the order of 100 kcps are reached. Rather considerable capacitance across each element may be expected to be the reason for this.

4-16. Symbols for Logic Elements. Simple symbols for mixers and gates are shown in Fig. 4-27. These are much like the symbols published by Hartree,[1] and are said to be due, at least as far as computer applications are concerned, to von Neumann. The modification used minimizes the possibility of confusion. Inhibition is simply shown by the

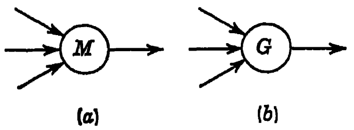


FIG. 4-27. Symbols for (a) mixers and (b) gates.

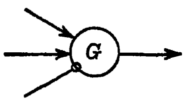


FIG. 4-28. Gate with inhibition.

symbol of Fig. 4-28. Circles with letters inside them are used to designate several kinds of circuits: thus, "*CF*" for cathode follower, "*I*" for inverter, "*P*" for puller. A simple circle carrying no letter and having a single input and a single output designates a vacuum tube or transistor. Other symbols will be introduced as they are needed.

[1] D. R. Hartree, "Calculating Instruments and Machines," p. 98, University of Illinois Press, Urbana, Ill., 1949.

CHAPTER 5

STATIC MEMORY CELLS

5-1. Introduction. A basic function that must be performed in digital computers is the retention of information until it is to be used; this is referred to as "memory" or "storage." In addition to the large-scale memory, in which the data of the problem and the code to be executed are held, other machine units must also incorporate devices that retain information. Thus, the arithmetic unit must contain registers that receive information from the large-scale memory and retain it until the arithmetical operations have been completed; and the control must be able to hold each instruction of the code until it has been executed. The chief devices for large-scale memory will be described in a later chapter; here those used in building up the high-speed registers used throughout the computer will be discussed. The present chapter will be limited to circuits and devices that have two distinct stable physical states, leaving to Chap. 6 those that retain information by virtue of periodic regeneration or recirculation. The present chapter covers the traditional Eccles-Jordan flip-flop, similar circuits using transistors, saturable magnetic cores, and ferroelectric elements.

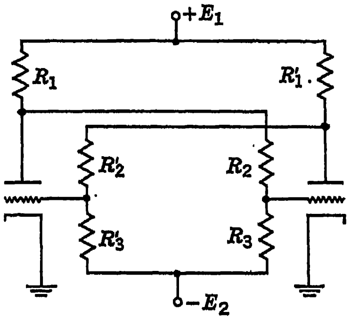


FIG. 5-1. Triode toggle circuit.

5-2. The Vacuum-tube Toggle. The most commonly used circuit for 1-bit storage derives from the trigger pair first described by Eccles and Jordan.[1] This circuit is frequently referred to as a "flip-flop," and sometimes as a "bistable multivibrator"; the term "toggle" will be used here (following J. H. Bigelow), for this term is more descriptive of its action.

A commonly used circuit schematic is shown in Fig. 5-1. A common variant of this circuit is the cathode-biased type, in which the points shown here connected to the negative supply voltage are grounded and the cathodes are connected to ground through a common resistor.

[1] W. H. Eccles and F. W. Jordan, A Trigger Relay Utilizing Three-electrode Thermionic Vacuum Tubes, *Radio Rev., Dublin*, vol. 1, pp. 143–146, 1919.

So-called "speed-up" capacitors are often connected across the crossover resistors R_2 and R'_2. Most often the circuit is symmetric, with $R_i = R'_i$. Another variant uses pentodes, with the cathodes, control grids, and screen grids connected as in Fig. 5-1, the plates connected through resistors to positive supply voltages, and the suppressors connected together; provision is made to set the voltage of the suppressors at ground or sufficiently below ground to inhibit plate current. So the circuit functions as a toggle plus read-out gates.[1]

The most complete analysis of the bistable nature of the circuit has been given by Starr.[2] For simplicity, Starr assumes that the design is such that no grid current ever flows. Let i_{p1} be the plate current in T_1 and let i_1 be the current flowing through R_2 and R_3. Let R_{p1} be the d-c plate resistance of T_1. The same letters with right-hand subscript "2" will refer to T_2, T_1 and T_2 being the left and right triodes. Then

$$E_1 = (i_{p1} + i_1)R_1 + i_{p1}R_{p1} \tag{5-1}$$
$$E_1 + E_2 = (i_{p1} + i_1)R_1 + i_1(R_2 + R_3) \tag{5-2}$$
$$e_{g2} = i_1R_3 - E_2 \tag{5-3}$$

From these equations the following are obtained by eliminating i_1:

$$i_{p1}(R_{p1} + R_{01}) = E_{01} \tag{5-4}$$
$$e_{g2} = E_{02} - R_{02}i_{p1} \tag{5-5}$$

where

$$\left.\begin{aligned}
&(a) \qquad R_{01} = \frac{R_1(R_2 + R_3)}{R_1 + R_2 + R_3}\\
&(b) \qquad R_{02} = \frac{R_1R_3}{R_1 + R_2 + R_3}\\
&(c) \qquad E_{01} = E_1\frac{R_{01}}{R_1} - E_2\frac{R_{02}}{R_3}\\
&(d) \qquad E_{02} = E_1\frac{R_{02}}{R_1} - E_2\frac{R_{01}}{R_1}
\end{aligned}\right\} \tag{5-6}$$

Similarly,

$$i_{p2}(R_{p2} + R'_{01}) = E'_{01} \tag{5-7}$$
$$e_{g1} = E'_{02} - R'_{02}i_{p2} \tag{5-8}$$

where the primed letters stand for the expressions obtained by priming the R_i in Eqs. (5-6).

The dynamic-transfer characteristic of T_1 is of the form A in Fig. 5-2, where the dashed portion emphasizes the restriction to the nonpositive

[1] Staff, Harvard Computation Laboratory, "Description of a Magnetic Drum Calculator," Harvard University Press, Cambridge, Mass., 1952. The schematic of the pentode toggle is on p. 52.

[2] A. T. Starr, A Trigger Peak Voltmeter Using "Hard" Valves, *Wireless Engineer*, vol. 12, no. 146, pp. 601–606, November, 1935.

grid region. From Fig. 5-2 and Eq. (5-5) a plot of e_{g2} versus e_{g1} is obtained. This is shown in Fig. 5-3 for two possible cases: I, $E_{02} > 0$, and II, $E_{02} < 0$.

The curve giving the variation of i_{p2} with e_{g1} is plotted on the axes of Fig. 5-2. This is shown in Fig. 5-4, together with three straight lines

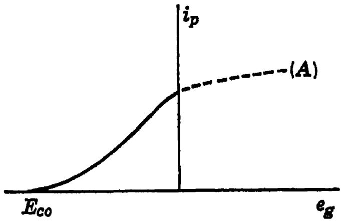


FIG. 5-2. Dynamic-transfer characteristic.

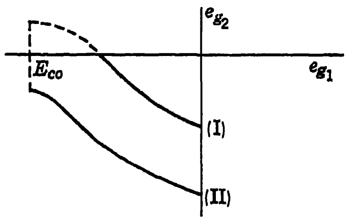


FIG. 5-3. Relationship of the grid voltages.

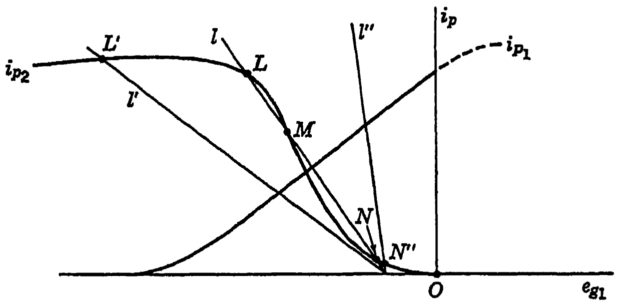


FIG. 5-4. Possible equilibrium states of the toggle.

representing Eq. (5-8) for three different values of the slope. These lines can intersect the i_{p2} curve in at most three points. The drawing shows two intersecting the curve in one point and one in three points; these points represent possible equilibrium positions of the circuit. Points such as M, however, cannot represent stable equilibrium positions. For suppose the circuit is in the condition represented by point M. It is clear from Eq. (5-8) that, if for any reason i_{p2} suffers a slight decrease, e_{g1} will become more positive, which will lead to a further decrease in i_{p2}. This process is cumulative until point N is reached.

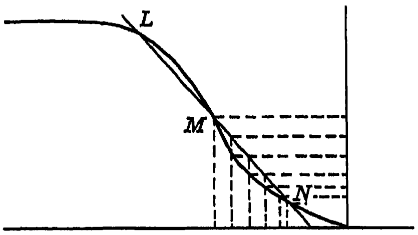


FIG. 5-5. Instability of point M.

Similarly, a slight increase in i_{p2} from the value at M automatically builds up until point L is reached. An illustration is given in Fig. 5-5. A similar argument shows that points L and N are stable: a slight variation from either tends to reduce itself and to bring the circuit back to the original condition. It is, therefore, clear that in Fig. 5-4 only the line l represents a bistable circuit; lines l' and l'' both represent circuits that have only one stable state. It is also evident that line l represents a circuit having two stable states in neither of which is either tube cut off, a situation generally to be avoided for it increases the chances of spurious triggering.

Given that the circuit has actually been designed to possess two stable states, it must now be shown how it can be caused to change from one to the other. In Fig. 5-6, let line 3 be the normal position of the line representing Eq. (5-8). The other lines represent the effect of adding various fixed voltages in series with the grid of T_1, the effect being to change E_{02}. Suppose the circuit is in the state corresponding to the lower stability point on line 3, which, following Fig. 5-4, will be called N_3, and suppose that it is desired to switch the circuit to the state corresponding to the upper stability point L_3. First let E_{02} be decreased, shifting line 3 to position 1. Here but one stable state exists, and the circuit must assume it; the operating point is now L_1. Now let E_{02} return to its original value, so that line 3 returns to its original position, carrying L_1 into position L_3. Similarly, switching the circuit from the state represented by L_3 into that represented by N_3 can be accomplished by shifting line 3 to position 5 and back again. The two cases considered exemplify triggering by negative pulses and by positive pulses, respectively. The transition of the circuit can also be caused by momentarily dropping the supply voltage E_1 to one of the plates, leaving the supply voltage to the other plate unaltered; in this case separate supply buses are required. Later on it will be seen that this method is used (in a number of machines) to "clear" the toggles, that is, to set them all in one of their two states.

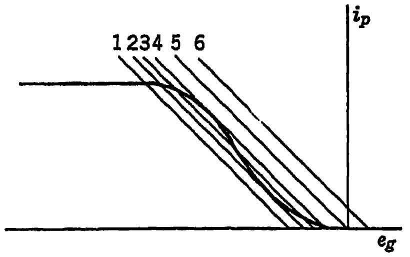


FIG. 5-6. Triggering of a toggle.

5-3. The Design of Triode Toggles. In digital computers, particularly of parallel type, large numbers of toggles are used. In the interest of reliability, it is necessary to choose a design that guarantees the desired operation despite variations in circuit parameters. Thus the effects of aging on resistors and tubes, of fluctuations in supply voltages, and of the usual tolerances in commercial resistors and tube characteristics should as far as possible be taken into account. Design methods that recognize these factors have been worked out and presented in the literature in readily applicable form. Here an account will be given only of the method of Johnston and Ratz,[1] since this is the least readily accessible.

[1] R. F. Johnston and A. G. Ratz, A Graphical Method of Flip-Flop Design, *AIEE Tech. Paper* 53–60, December, 1952.

The reader is referred also to the papers of Rubinoff[1] and of Paivinen and Auerbach.[2]

Postulate a symmetrical circuit. Two tolerances are introduced: (1) t_e, the tolerance in the supply voltages, such that the plate supply always lies in the range $(1 - t_e)E_1$ to $(1 + t_e)E_1$, where E_1 is the nominal value, and similarly for the negative supply of nominal value $-E_2$, (2) t_r, the tolerance in the resistors, such that the actual value of any R lies in the range $(1 - t_r)R$ to $(1 + t_r)R$, where R is the nominal value. Specify also the maximum d-c plate resistance of the tube at zero grid voltage acceptable (written R_t) and the maximum grid voltage for cutoff at a plate voltage of E_1 (written E_c). In this way the divergence of tube characteristics from the average values published by the manufacturer and the effects of aging upon tube performance can be taken into account.

The next step is to derive inequalities that guarantee (1) that the grid of the "on" tube is positive and (2) that the grid of the "off" tube is below cutoff—an obvious means of minimizing the possibility of spurious triggering by bursts of noise. For (1), the worst possible condition is that the plate-supply voltage should be $(1 - t_e)E_1$ and the bias voltage $-(1 + t_e)E_2$, with the values of the resistors $(1 + t_r)R_1$, $(1 + t_r)R_2$, and $(1 - t_r)R_3$. If no grid current flows in the "on" tube, the grid voltage is given by

$$e_g = (1 - t_r)R_3 \frac{E_1 + E_2 + (E_2 - E_1)t_e}{(1 + t_r)(R_1 + R_2) + (1 - t_r)R_3} - (1 + t_r)E_2 \qquad (5\text{-}9)$$

If e_g as calculated by Eq. (5-9) turns out to be positive, then grid current flows, and the grid voltage is reduced by a term proportional to the grid current, so in practice it can be only slightly positive. The necessary and sufficient condition for the grid voltage to be positive is that the right-hand member of Eq. (5-9) should be positive; this condition, after some simplifying, is seen to be equivalent to the inequality

$$\frac{E_1}{E_2}\frac{1 - t_e}{1 + t_e} > \frac{R_1 + R_2}{R_3}\frac{1 + t_r}{1 - t_r} \qquad (5\text{-}10)$$

which can be rewritten in the form

$$R_2 < g\frac{E_1}{E_2}R_3 - R_1 \qquad (5\text{-}11)$$

where $$g = \frac{1 - t_e}{1 + t_e}\frac{1 - t_r}{1 + t_r}$$

The requirement that the grid voltage of the other tube be below cutoff

[1] M. Rubinoff, Further Data on the Design of Eccles-Jordan Flip-Flops, *Elec. Eng.*, vol. 71, no. 10, pp. 905–910, October, 1952.

[2] J. O. Paivinen and I. L. Auerbach, Design of Triode Flip-Flops for Long Term Stability, *IRE Trans. on Electronic Computers*, vol. EC-2, no. 2, pp. 14–26, June, 1953.

leads similarly to a second inequality:

$$R_2 > \frac{1}{g} \frac{s + e_c}{E_2/E_1 - e_c} R_3 - sR_1 \tag{5-12}$$

where $s = R_t/(R_1 + R_t)$ and $e_c = E_c/E_1$.

Replace the inequalities (5-11) and (5-12) by equations and suppose that all quantities except R_1, R_2, and R_3 are given. Choosing a value of R_1, plot the equations as straight lines in the (R_3,R_2) plane; these are the lines L_1 and L_2, respectively, of Fig. 5-7. Any point lying below L_1 and above L_2 satisfies the inequalities (5-11) and (5-12). Hence R_1 must be chosen such that the slope of L_2 is less than the slope of L_1. Setting the two slopes equal gives a value of R_1 that must be exceeded if the circuit is to perform at all. Similarly, it can easily be shown by comparing the expressions for the slopes of L_1 and L_2 that, if R_1 is fixed but E_1 and E_2 are subject to choice, a minimum ratio E_1/E_2 exists which must be

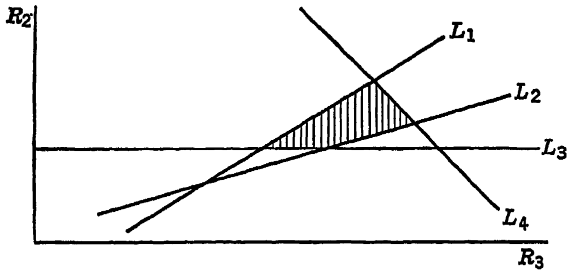


FIG. 5-7. Permissible area for R_2 and R_3, according to Johnston and Ratz.

exceeded if a design is to be possible. Supposing that R_1, E_1, and E_2 have been chosen such that the slope of L_1 exceeds the slope of L_2, the two lines appear as shown in Fig. 5-7.

The region of possible values of R_2 and R_3 can be further delimited if other conditions are imposed. As an example, suppose that the plate voltage of the "on" tube is not to exceed a value E_D and that of the "off" tube is to exceed a value E_U. These conditions lead to two more inequalities:

$$R_2 > \frac{[R_1/(1 + t_e)](E_U/E_1)}{g - (E_U/E_1)[(1 - t_r)/(1 + t_e)(1 + t_r)]} \tag{5-13}$$

$$R_2 < -R_3 + \frac{(E_D/E_1 + E_2/E_1)gsR_1}{s(1 - t_e) - g(E_D/E_1)[(1 + t_r)/(1 - t_r)]} \tag{5-14}$$

By replacing these inequalities with equations and plotting, two more lines are obtained: L_3 (horizontal) and L_4 (of slope equal to -1). Points above L_3 and below L_4 satisfy the inequalities. These lines are plotted along with L_1 and L_2 in Fig. 5-7, and the small shaded area is the region from which values of R_2 and R_3 compatible with the choice of the other parameters and the requirements may be chosen.

Of course, inequalities arising from other requirements can be used in place of (5-13) and (5-14), along with the basic stability inequalities (5-11) and (5-12).

Consider next the dynamics of the circuit. Switching from one stable state to the other is not instantaneous, because of the interelectrode capacitances of the tubes and the stray capacitances of the circuit. When switching takes place, the charges held by these capacitances must be changed; so the time required for the transition depends upon the time constants involved.

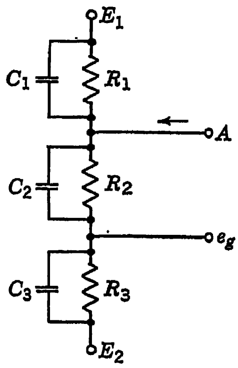


FIG. 5-8. Circuit used in analysis of settling time.

First consider triggering by a positive pulse applied to the grid of the "off" tube. The total time required to switch the circuit from one stable state to the other can be divided into three parts: (1) the time required to bring the circuit to the point at which positive feedback becomes effective; (2) the actual transition time, which begins when positive feedback becomes effective and ends when it ceases to be effective; and (3) the time required after positive feedback ceases for the voltages and currents to reach their final stable values. From an analysis[1] certain important points become clear: (1) a critical trigger duration exists, such that shorter triggers are unable to initiate the switching action; (2) circuits designed for very fast switching are not necessarily the best from the point of view of minimum tolerable time between triggers.

Tillman studied a toggle using pentodes and concluded that the critical trigger duration is a small multiple (say 2) of the time constant R_1C_1, where R_1 is the plate-load resistance and C_1 the total capacitance shunting it. Therefore, in the case studied, this duration does not exceed a few hundredths of a microsecond. It is hardly desirable to attempt to make this smaller, because success in this undertaking merely intensifies trouble from spurious triggering by noise.

The transition speed and settling-down time of a triode toggle have been studied by Rubinoff,[2] who bases his analysis upon the behavior of the circuit of Fig. 5-8. Here E_1, E_2, R_1, R_2, and R_3 have the meanings used before in this chapter. It is assumed that point A is connected to the plate of the "on" tube, which is abruptly cut off, and the effect of this sudden action is assumed equivalent to the application of a step of cur-

[1] This analysis is due to Tillman. J. R. Tillman, Transition of an Eccles-Jordan Circuit, *Wireless Engineer*, vol. 28, no. 331, pp. 101–113, April, 1951.

[2] *Op. cit.*, pp. 909–910.

rent injected into the circuit at point A. Simple circuit analysis shows that the rise of e_g is dependent upon a term

$$(1 + b)e^{-\alpha t} - be^{-\gamma t}$$

where α and γ are rather complicated functions of the R's and C's and of b; in general $\gamma \geq \alpha$, with $\gamma = \alpha$ only if the three time constants $\tau_i = R_iC_i$ are equal, and b is given by the expression

$$\frac{\alpha(1 - \gamma\tau_2)}{\gamma - \alpha}$$

It is possible to deduce from the analysis a number of conclusions (and the reader should find no difficulty in doing this if he possesses sufficient patience). Perhaps the most interesting of these conclusions is that, if τ_2 is made sufficiently large (larger than either τ_1 or τ_3), the grid voltage attempts to overshoot its final value. In this way the time required to switch the circuit from one stable state to the other can be made small. The rising grid does not actually overshoot, since it is clamped at cathode potential by the flow of grid current. The falling grid, however, overshoots in the negative direction and must subsequently rise to its final value; this means that, if triggering takes place before the quiescent value is reached, the low grid has further to rise and hence requires a longer time. Thus it must be concluded that it is impossible to minimize both the switching time and the time between successive triggering actions, which is called the "resolving time."

The present section will be closed by citing a design that has been used in a number of machines.[1] The circuit is that of Fig. 5-1; it is symmetrical, and no crossover capacitors are used. The tube is a 6J6 (or equivalent); $R_1 = 15$, $R_2 = 33$, and $R_3 = 120$ kilohms; $E_1 = 150$, $E_2 = -300$ volts. The "on" tube's grid voltage is about 0.5 volt positive, and its plate voltage is about 35 volts. For the "off" tube these values are about -40 and 100 volts, respectively. The resolving time is about 0.5 μsec. No diodes are used to clamp grid or plate voltages, for the design is such that rather large variations in electrode voltages are tolerable.

5-4. Communications with Toggles: Symbols. This section takes up the ways in which information can be inserted into or extracted from toggle circuits. First, however, the symbols used to represent these circuits must be illustrated. Figure 5-9*a* represents the common grounded-cathode toggle; the symbol is simple and obvious, and merely omits the passive circuit elements. This is modified in Fig. 5-9*b* to represent a toggle provided with two separate plate-voltage-supply connections that are used for clearing operations. Here it is convenient to include the plate resistors in the diagram. If the toggle is of the self-biased type, the

[1] In the IAS computer and in those whose design is based upon it.

diagram of Fig. 5-9*c* is used, in which an additional input terminal is shown. In all the circuits, the plates and grids are available as input or output terminals. These diagrams are used whether the tubes are triodes, tetrodes, or pentodes—if the last are used in the conventional way. The Harvard Mark III circuit, which uses pentodes in which the suppressors and plates are essentially output gates, would require that all tube elements be shown.

Finally, the simple block shown in Fig. 5-9*d* is used in logical diagrams. The arrows entering the base at the left and right of the vertical line are inputs which, upon being affirmed, set the toggle to 1 or to 0, respectively, while the arrow entering at the center of the base is a "complement" input, which causes the toggle to change state. This arrow will be shown only in cases in which such an input is actually used. The outputs ordinarily go to gates (or mixers) and signify that the affirmative, or

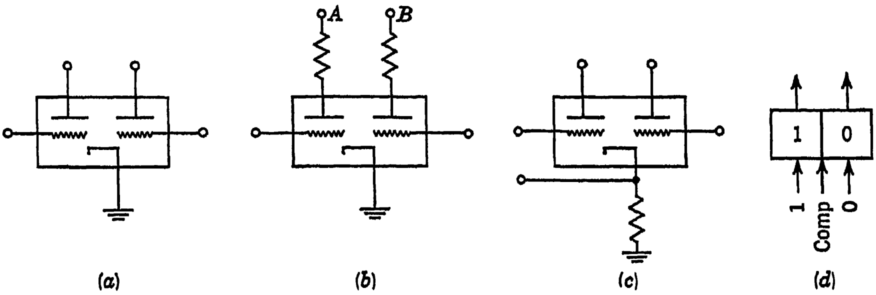


FIG. 5-9. Toggle symbols.

enabling, signal exists at the left or at the right output when the toggle holds a 1 or a 0, respectively.

There are two different approaches to the triggering of toggles: (1) the toggle may be instructed to change state or not to change state, assuming that the lumped and distributed capacitances can be relied upon to remember the previous state; or (2) the triggering voltage may be so applied that the circuit is guaranteed to assume a definite state. The latter will be referred to as "positive-action" triggering. In many computers both these schemes are used; in others the second scheme is used consistently.

First, consider the problem of "clearing," i.e., putting the circuit into one of its two stable states, without regard to the previous state. This is necessary, for example, when, at the beginning of an operation, a register must be caused to hold all 0's. In existing machines this is done in several ways. One of these is to provide means of applying triggering pulses (usually via *RC* networks) to each of the grids. Thus, if, in the circuit of Fig. 5-9*a*, the 0 state is that in which the left tube conducts, the clear-to-0 pulse could be a negative pulse applied to all right grids only. If a circuit is already in the 0 state, nothing will happen; if it is in the 1 state a change will occur. Note that one of two things may be done: (1) the triggering pulse may be made short, intended merely to initiate action and relying upon the remembered previous state of the circuit; or (2) the triggering pulse may be made so long that the circuit assumes its new state before the pulse is removed. The latter is true "positive action." A very simple way of accomplishing it is used in the computer at the Institute for Advanced Study (IAS). In this computer, all the toggles of a register are supplied by two separate buses (*A* and *B* in Fig. 5-9*b*) and clearing is accomplished by dropping the appropriate bus from its normal value of 150 to 50 volts for a time long enough to guarantee that all the toggles assume the desired state. In the example cited above, clearance to 0 is accomplished by dropping the voltage of the *A* bus.

Not only is it frequently necessary to clear toggles, but even more often it is necessary to change the information held. Thus, in the circuit of Fig. 5-9*c*, it is possible merely to apply a sufficiently positive pulse to the cathode to cut off the "on" tube and to rely upon the redistribution of charge on the lumped and distributed capacitances to cause the other tube to assume the "on" state after the removal of the trigger, thus essentially depending upon the circuit's remembering its former state. A similar scheme is to apply a negative pulse simultaneously to both grids. In such schemes, the shape of the triggering pulse may well be important.

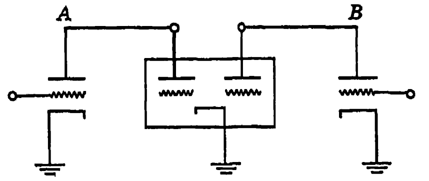


FIG. 5-10. Clearing and read-in by means of puller tubes.

Positive-action triggering schemes are more reliable, and, at the expense of some additional equipment, they avoid dependence upon pulse shape. Consider the circuit of Fig. 5-10, which is used in the Harvard Mark III.[1] The two triodes (puller tubes) are used to read in a 0 or a 1, respectively, depending upon the convention used. Suppose that 0 is signified by the left tube of the toggle being on; 0 can be read in (or the circuit may be cleared to 0) by raising the grid of puller tube *A*, which is normally cut off, above the cathode, and, similarly, puller tube *B* can be used to read in a 1. Normally, it would be desirable to deliver information via a single path only, e.g., via puller tube *B*. As this tube can insert only a 1, it is necessary to break the process down into two steps: (*a*) use *A* to clear the toggle to 0; (*b*) use *B* either to read in a 1 or to leave the circuit unaltered; that is, if 1 is to be transmitted it arrives via *B* and sets the toggle to 1, whereas if 0 is to be transmitted no pulse comes via *B*, and the toggle continues to hold the 0 to which it was previously cleared.

[1] "Description of a Magnetic Drum Calculator," p. 51.

The toggles of the registers of the IAS computer use a similar scheme: (*a*) clearance to 0 (or 1) by dropping the appropriate bus voltage, (*b*) read-in by triodes which are in fact combined pullers and gates.

As observed above, the plate and grid voltages of a toggle are available as outputs. As the plate swing is apt to be greater than the grid swing and ordinarily occurs between positive levels (unless the cathodes of the toggle tubes are returned to a negative voltage), it is frequently necessary or desirable to provide additional voltage-divider networks from plate to $-E_2$. This complication is not particularly desirable.[1] Another scheme is to use the grid voltages directly as outputs, as in the Harvard Mark III and the IAS computer; in the latter, for example, 0 and -40 volts are available to represent 1 and 0. In both computers, the toggle grid voltage is isolated by a cathode follower from the stage it is to drive in all cases where undue loading would otherwise result. Although neither of the computers does so, it is possible, if desired, to use clamping diodes on both grid and plate circuits to define the "on" and "off" grid and plate voltages precisely and thus to avoid variation from tube to tube.[2]

5-5. A Single-pentode Bistable Circuit. A bistable circuit using a single pentode has been described by Reich.[3] It has not, to the author's knowledge, found any application in computers. It depends upon the possibility of causing, by interconnecting the suppressor and screen of a pentode by means of a resistor, the screen-current–screen-voltage characteristic to contain a portion of negative slope.

5-6. Transistor Toggle Circuits. A variety of transistor toggle circuits have been described in the literature. The earlier ones used but a single point-contact transistor.[4] The possibility of such a circuit depends upon

[1] Paivinen and Auerbach, *op. cit.*, pp. 21–22.

[2] See *ibid.*, for example, p. 15, for clamping in the grid circuit.

[3] Herbert J. Reich, "Theory and Applications of Electron Tubes," pp. 208–209, McGraw-Hill Book Company, Inc., New York, 1944.

[4] A great number of papers were written on point-contact-transistor toggle circuits; as examples, the following will be cited:

E. Eberhard, R. O. Endres, and R. P. Moore, Counter Circuits Using Transistors, *RCA Rev.*, vol. 10, pp. 459–476, December, 1949.

A. W. Lo, Transistor Trigger Circuits, *Proc. IRE*, vol. 40, no. 11, pp. 1531–1541, November, 1952.

The following books will also be cited:

R. F. Shea et al., "Principles of Transistor Circuits," John Wiley & Sons, Inc., New York, 1953. See chap. 19 for point-contact circuits.

Lloyd P. Hunter (ed.), "Handbook of Semiconductor Electronics," sec. 15, McGraw-Hill Book Company, Inc., New York, 1956. Both point-contact and junction-type transistor circuits are covered.

A. W. Lo, R. O. Endres, J. Zawels, F. D. Waldhauer, and Chung-Chih Cheng, "Transistor Electronics," chap. 12, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1955. Both point-contact and junction-type toggle circuits are covered.

the negative incremental resistance presented by point-contact transistors under certain conditions. Circuits of this type, which are now chiefly of historical importance, will be considered briefly; then circuits appropriate to junction transistors will be described

Consider a point-contact transistor with load resistor and collector power supply as shown in Fig. 5-11. The plot of V_e versus I_e is given in Fig. 5-12. The curve falls into three sections: the cutoff region I, the transition region II, and the saturation region III. The negative slope in the transition region is characteristic of point-contact transistors; it can occur only when the current-amplification factor exceeds unity. If a source and a resistor R_B are placed in series across the input terminal of Fig. 5-11, possible operating points may be found by drawing upon the same axes the load line A of slope R_B. In the drawing, R_B is so chosen that the load line intersects the curve in three points P, Q, R, each of which represents a possible equilibrium condition of the circuit. Q, however, is a point of unstable equilibrium; so only two stable positions exist.

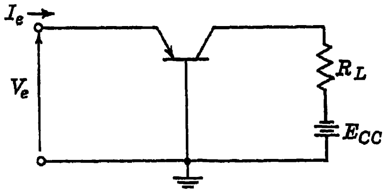


FIG. 5-11. Basic point-contact-transistor bistable circuit.

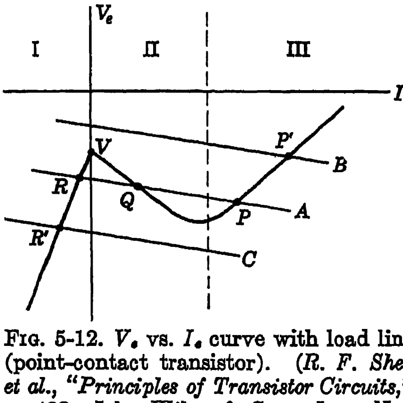


FIG. 5-12. V_e vs. I_e curve with load line (point-contact transistor). (*R. F. Shea et al., "Principles of Transistor Circuits," p. 428, John Wiley & Sons, Inc., New York,* 1953.)

Suppose that the circuit of Fig. 5-13 is in the condition represented by point R; in this circuit the resistor R_B is added to increase the negative slope in region II. A positive triggering pulse applied to the emitter momentarily shifts the load line up above the apex V of the curve in which position only the equilibrium point given by P' exists. The circuit accordingly assumes this equilibrium state, and, when the triggering pulse is removed, P' slides down the curve to point P. Unfortunately, to trigger the circuit from P to R requires the use of a negative pulse to shift the load line down to position C.

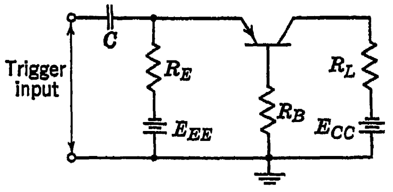


FIG. 5-13. Point-contact-transistor toggle. (*R. F. Shea et al., "Principles of Transistor Circuits,"* p. 433, *John Wiley & Sons, Inc., New York,* 1953.)

One possible way of avoiding the need for triggering pulses of opposite polarity is to make the time constant of the input circuit small compared to the trigger pulse duration.[1] Under these circumstances, provided that the triggering pulse has steep leading and trailing edges, the differentiating action of the input circuit provides a pair of pulses of opposite polarities, one or the other of which triggers the circuit. This expedient makes the action strongly dependent upon pulse shape and provides triggering pulses of very small width. It has little to recommend it from the point of view of reliability.

Alternate triggering by pulses of opposite polarity has been exploited by Felker,[2] in a circuit which he calls a "regenerative amplifier." The purpose of this circuit is to recreate pulses in a standard form. The circuit is triggered first from point *R* to point *P* (in Fig. 5-12) by the pulse to be regenerated, and it is then triggered back to *R* by a standard clock

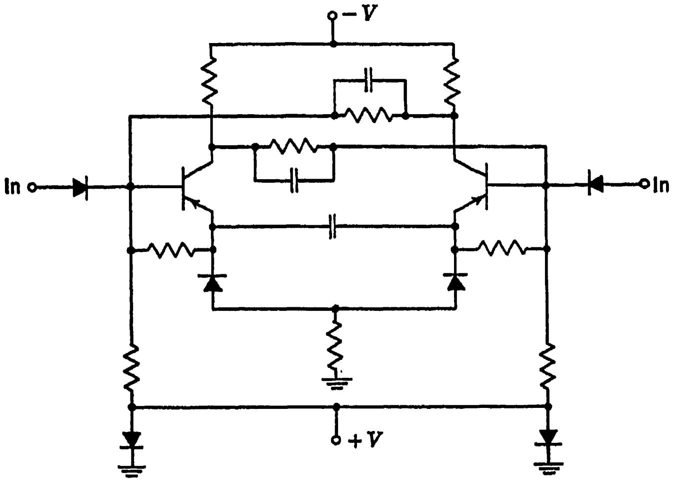


FIG. 5-14. Bistable circuit using two transistors. (*R. L. Trent, Proc. IRE, vol.* 40, *no.* 11, *p.* 1562, *November,* 1952.)

pulse, the output voltage being taken from the collector. The output is thus independent of input amplitude and duration.

Two-transistor toggles using point-contact transistors avoid the undesirable necessity of triggering by pulses of alternate polarities. Carlson[3] devised a circuit consisting essentially of two of the circuits of Fig. 5-11 connected emitter to emitter and with resistors inserted between the bases and ground; Trent (Fig. 5-14) has described a circuit which bears a superficial resemblance to the familiar Eccles-Jordan toggle.

[1] Eberhard, Endres, and Moore, *op. cit.*, p. 467.

[2] J. H. Felker, Regenerative Amplifier for Digital Computer Applications, *Proc. IRE*, vol. 40, no. 11, pp. 1584–1596, November, 1952.

[3] A. W. Carlson, High Speed Transistor Flipflops, *Air Force Cambridge Research Center Tech. Rept.* 53-16, June, 1953.

All point-contact circuits, however, suffer from the variability and drift of transistor parameters and require some sort of stabilization if reliable operation is to be obtained. For example, in Fig. 5-13,[1] the wire from the lower end of R_B to ground can be replaced by two paths, one to ground via a diode, the other to a positive voltage source via a resistor, R_B itself being fairly small and the other resistor much larger (for example, 1 kilohm and 10 kilohms, respectively). For small positive or negative values of I_e, the diode conducts, with the result that the resistance from base to ground is small, and this effectively prevents variations in the collector resistance from having much effect in the region near point V (Fig. 5-12). When the current I_e increases in the positive direction, current flows from the source upward to the base, the diode no longer conducts, and the effective base resistance becomes large, making the curve in region II quite steep.

The various stabilization schemes have the common drawback that they severely limit the operating speed of the circuit (by a factor of 5, according to Poppelbaum[2]). Hence, although a switching time of as little as 0.2 μsec between stable states has been reported,[3] adequately stabilized circuits generally can be operated with good reliability only at rates considerably below 1 Mcps. As an example will be cited one of the earliest transistor computers, built by Cooke-Yarborough at Harwell, in which it was found advisable to limit the repetition rate to 60 kcps.

The earlier junction transistors were rather poor for applications of the type considered here, for they were very limited in frequency response, and, therefore, toggles made of them switched rather slowly. However, as improvements in design have been made, this drawback has been overcome. Present transistors of, e.g., the "drift" type (IBM) or the surface-barrier and silicon-surface-alloy types (Philco) put 10-Mcps transistor-digital-computer circuitry within the designer's grasp. Naturally, circuit design must be appropriate to the type used if high speeds are to be realized; thus, the hole-or-electron-storage effect can limit operating speed severely if saturation is permitted to occur.[4]

The existence of both *npn* and *pnp* transistors and the consequent possibility of obtaining amplification with either a rise or a drop in the d-c output level make it possible to devise a number of bistable circuits that are fairly tolerant of component parameter variations and that avoid the slowing-down effects due to saturation.

[1] Shea et al., *op. cit.*, p. 433.

[2] W. Poppelbaum, A Fast Junction Transistor Flipflop, with Stabilized Output Levels (report on ONR contract N6ori71, TO XXIV), Digital Computer Laboratory, University of Illinois, Jan. 24, 1955.

[3] B. G. Farley, Dynamics of Transistor Negative Resistance Circuits, *Proc. IRE*, vol. 40, no. 11, pp. 1497–1508, November, 1952.

[4] Poppelbaum, *op. cit.*, p. 1.

A circuit due to Wier[1] depends upon the bistable character of the circuit shown in Fig. 5-15, which consists essentially of a *pnp* and an *npn* amplifier in series. The two stable states are (1) both transistors conducting, and (2) both transistors cut off. The conditions that must be fulfilled are (1) that, given $I_e = 0$ in both transistors and $I_c = I_{co}$, the voltage drop across R_C must be smaller than the drop across R_E to guarantee cutoff, and (2) that, when both transistors are conducting, the diode must draw enough current to reduce the a-c loop gain to less than 1. Wier has given a design procedure that leads to the correct values of resistors and voltage to guarantee the bistable character of the circuit and the avoidance of saturation in the conducting state. The measured switching time from one state to the other given by Wier is between 0.3 and 0.5 μsec.

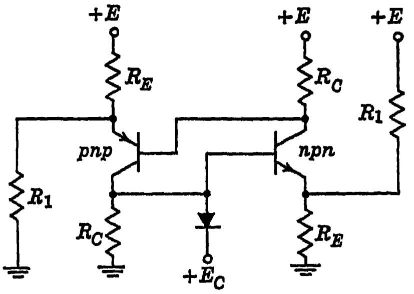


FIG. 5-15. *pnp-npn* bistable circuit.

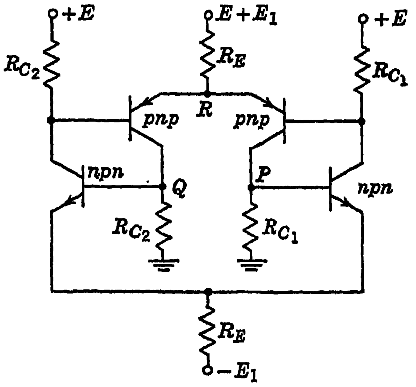


FIG. 5-16. Symmetrical bistable circuit based upon Fig. 5-15.

The circuit of Fig. 5-15 is not itself well suited to computer applications because of its lack of symmetry, which makes it impossible to find two points from which complementary signals can be taken. The four-transistor circuit of Fig. 5-16 consists essentially of two of these circuits and is completely symmetrical. To see that it is in fact bistable, consider the case where the two right transistors are conducting, and assume the two left ones cut off. Then the voltages at points P and Q are

$$\alpha I_e R_{c_1}$$

and

$$2I_{co}R_{c_2}$$

If the collector resistors have the common value R_C, then the voltage at

[1] J. M. Wier, Some Direct-coupled Computer Circuits Utilizing NPN and PNP Transistors in Combination (report on ONR contract N6ori71, TO XXIV), Digital Computer Laboratory, University of Illinois, Aug. 31, 1955.

P minus the voltage at Q is

$$R_C(\alpha I_e - 2I_{co})$$

which is positive by a fairly large amount. Also, since the common emitter connection R must follow the base of the conducting transistor, it is easy to see that this is in fact a stable condition.

The circuit of Fig. 5-16 is very tolerant of component drifts. It can be designed to operate with small voltage swings and in such a way that no transistor ever saturates. The switching time is approximately the same as for the circuit of Fig. 5-15. P and Q are convenient points from

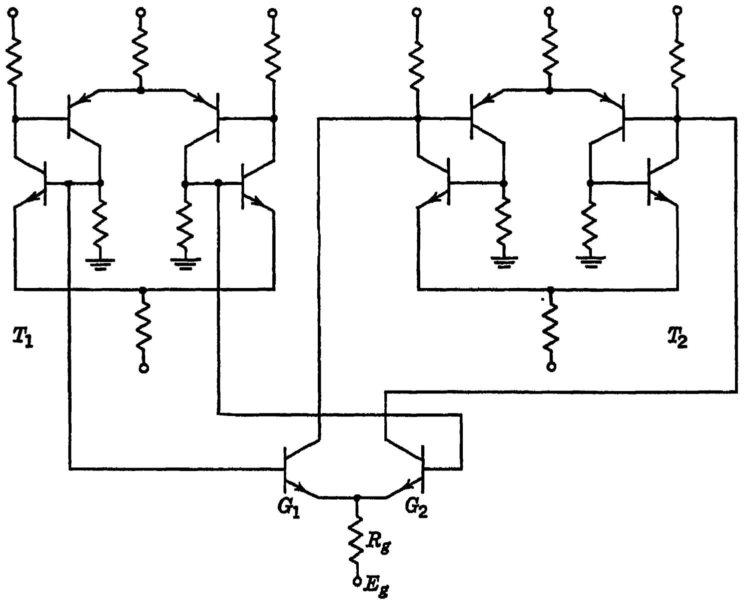


FIG. 5-17. Transfer of information between transistor toggles.

which to take complementary outputs. A circuit for transferring information from one such circuit to another is shown in Fig. 5-17, where E_g is normally high enough to exceed the positive voltages on the bases of the transistors G_1 and G_2 by an amount sufficient to ensure that they are both cut off. To accomplish the transfer, E_g is made negative with respect to the more positive possible level of base voltage, so that either G_1 or G_2 draws collector current and forces toggle T_2 to assume the same state as T_1.

Another symmetrical junction-transistor circuit, due to Poppelbaum,[1] is shown in Fig. 5-18, where the two transistors, a *pnp* and an *npn*, have a common emitter resistor and a common base connection, and thus form

[1] *Op. cit.*

an emitter follower, with one transistor working for positive and the other for negative base voltages. Between points P and Q is connected a grounded-base d-c amplifier, giving at Q an amplified and inphase version of the voltage appearing at P. To see the bistable nature of the circuit, imagine a positive voltage at Q less than $+u_0$. This voltage tends to make P more positive, which in turn forces Q up, and this continues until $+u_0$ is reached, whereupon D_1 conducts, and further increase of Q is stopped. On the other hand, a negative excursion of the point Q continues until the voltage at Q is caught at $-u_0$ by D_2. Clearly, one of these actions must take place if Q is initially at any voltage between $+u_0$ and $-u_0$.

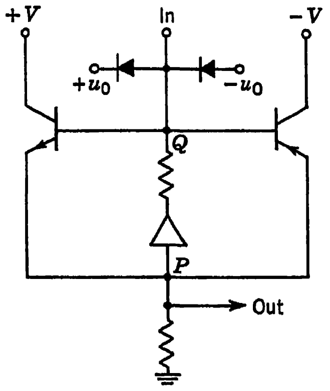


Fig. 5-18. Another form of *npn-pnp* transistor toggle.

The circuit appears to be fairly tolerant of parameter variations and can be so designed that no transistor is ever saturated. A complete schematic, including the amplifier, is given in Fig. 5-19, where the *pnp* transistor was a GE 2N44 and the *npn* a Texas Instrument 200. For a circuit of this type tests have shown a switching time of less than 0.5 μsec; a triggering pulse as short as 0.2 μsec can be used.

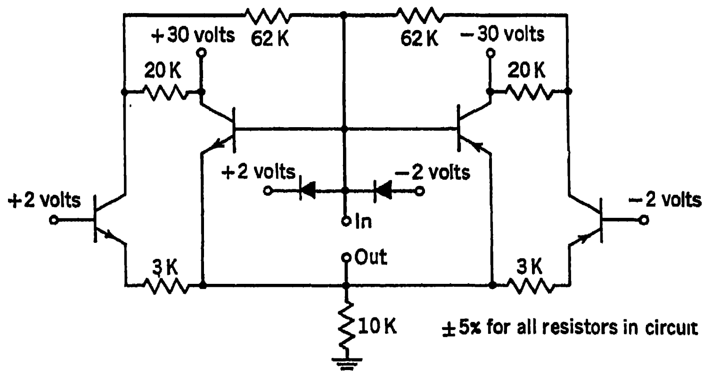


FIG. 5-19. Schematic of Fig. 5-18, showing amplifier.

With the improvement in the frequency response of transistors, it is obviously possible to obtain toggles that switch faster. As an example, consider the Philco surface-barrier transistors already mentioned in connection with logic circuits in Sec. 4-13. A very simple circuit[1] using

[1] R. H. Beter, W. E. Bradley, R. B. Brown, and M. Rubinoff, Surface Barrier Transistor Switching Circuits, *IRE Convention Record, 1955*, pt. 4, pp. 145–148.

these is shown in Fig. 5-20. If T_1 is conducting, its base must be slightly negative and its collector is very nearly at ground. This is consistent with T_2 cut off; so the state is stable, and so is the case of T_1 cut off and T_2 on. In the given circuit the collector of the "off" transistor is at -0.35 volt, and that of the "on" transistor is at -0.03 volt. Switching from one state to another can be accomplished by (e.g.) pulsing the base of the "off" transistor in the negative direction, thus turning it on and cutting off the other. This simple circuit can be switched from one state to another in ¼ μsec. If separate collector supply buses are provided, the toggle can be caused to switch by momentarily causing the bus that supplies the "on" transistor to assume a more negative voltage. A similar circuit using an experimental Philco *pnp* junction transistor[1] (of the silicon-surface-alloy type) has been reported to switch in a few tenths of a microsecond. The transistors employed have the additional advantage that they can be operated at temperatures as high as 140°C.

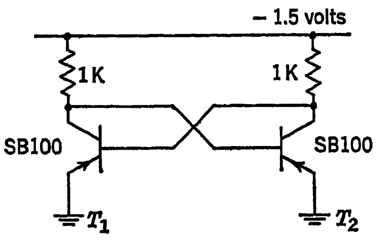


FIG. 5-20. Direct-coupled surface-barrier transistor toggle.

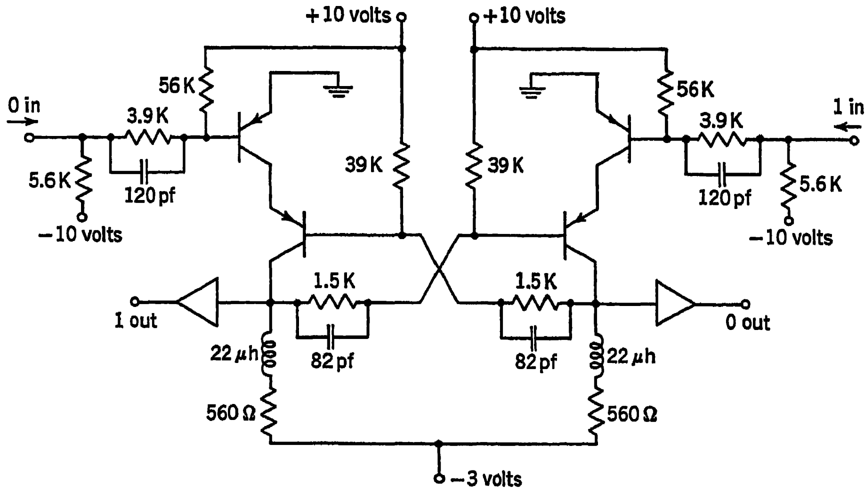


FIG. 5-21. TX-0 toggle. (*J. L. Mitchell and K. H. Olsen, Proc. Eastern Joint Computer Conf., 1956, p.* 93, *American Institute of Electrical Engineers, New York,* 1957.)

A more elaborate circuit, developed in the Lincoln Laboratories for the TX-0 computer, is shown in Fig. 5-21. This circuit also uses Philco

[1] J. B. Angell, High Temperature Silicon Transistor Computer Circuits, *Proc. Eastern Joint Computer Conf., 1956,* pp. 54–57, American Institute of Electrical Engineers, New York, 1957.

surface-barrier transistors. The grounded-emitter inverter circuits in series with the emitters of the toggle transistors are used for read-in. This is accomplished by pulsing the appropriate input terminal positive, which cuts off the inverter concerned and hence cuts off in turn the toggle transistor if it is in conduction; a positive pulse of a little less than 1.5 volts suffices to cause these events. The 120-pf (picofarad) capacitor is sufficiently large to take all the "holes" out of the base in a very short time. The output amplifiers serve both to isolate the toggle from its loads and to provide output voltages of 0 and −3 volts corresponding to the two stable states in place of the collector voltages, which differ by only a few tenths of a volt. This circuit has been caused to switch back and forth satisfactorily at rates up to 10 Mcps.

So far the circuits considered have been of the direct-coupled and resistance-coupled types. Coupling by means of emitter followers[1] results in higher-speed operation. A nonsaturating emitter-follower-coupled toggle using surface-barrier transistors is shown in Fig. 5-22. The transistors in the emitter-follower circuits do not suffer from hole-storage effects and they serve as a low-impedance source to supply relatively large currents to charge stray and internal capacitances, and thus expedite the transition from one state to the other. It has been stated that the circuit shown in Fig. 5-22 can perform the transition from one state to another in 22 mμsec; to this very small time must of course be added a short time, perhaps 10 mμsec, which elapses between the application of the driving signal and the initiation of the transition. Thus, the toggle should be capable of being driven at rates up to about 30 Mcps.

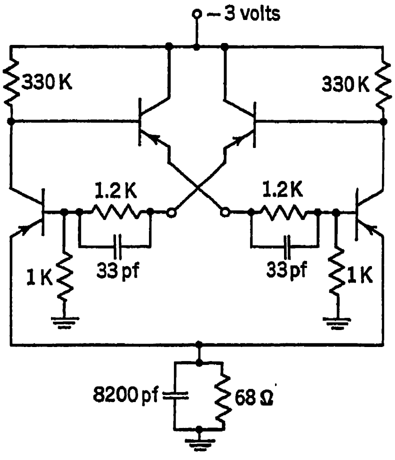


FIG. 5-22. Nonsaturating emitter-follower-coupled toggle.

5-7. Magnetic Binary Elements. A way of obtaining greater reliability in digital computers is to replace vacuum-tube circuits by magnetic circuits wherever possible. This has already been touched upon in Sec. 4-14, where the use of magnetic elements in logic circuits was described. Attention is now turned to the use of such elements for the storage of binary information; this is indeed their primary function, for their use in logic circuits depends upon it.

[1] A. K. Rapp and S. Y. Wong, Transistor Flip-Flops Have High Speed, *Electronics*, vol. 29, no. 12, pp. 180–181, December, 1956.

The pioneer work in this field was done at the Harvard Computation Laboratory. The original idea was due to H. H. Aiken, who saw early that the properties of certain ferromagnetic materials, which are characterized by nearly rectangular hysteresis loops, low coercive force, and high remanence, might be exploited to obtain simple, rugged, and reliable binary-storage elements.[1–3] In the present chapter, a brief discussion of the principles of such devices will be given; accounts of the Harvard magnetic delay lines and shift registers and of the multicore memory arrays developed at the MIT Digital Computer Laboratory, RCA Laboratories, and elsewhere will be deferred.

At this point the reader would be well advised to reread the material given in Sec. 4-14, particularly the discussion of the kinds of materials available, both metallic and ceramic, and of their magnetic behavior.

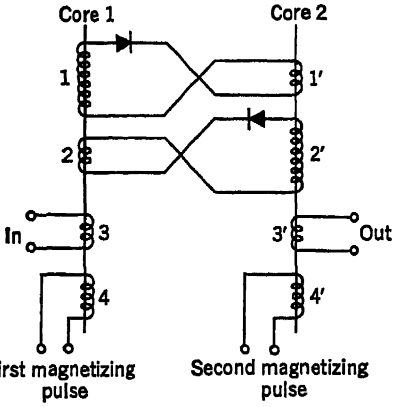


FIG. 5-23. Magnetic "trigger pair."

It is easy to see that a core of such material, provided with three windings for read-in, sensing, and read-out, can be used to store information until it is needed. However, in such a simple device, sensing is destructive of the information held, and some means must be employed to regenerate it. Later on it will be seen how this problem is solved in the high-capacity core memories, but the present treatment will show how the problem was first solved at the Harvard Computation Laboratory.[4] The magnetic trigger pair described by An Wang is shown in Fig. 5-23. Each core carries four windings: two for transfer, one for read-in or read-out, and one for sensing. Sensing is done by the magnetizing pulses, so chosen that they drive the coil to which they are applied to negative magnetic saturation. The diodes in the connecting paths between 1 and 1′ and 2 and 2′ are so arranged that current can flow from 1 to 1′ only when core 1 makes a transition from positive to negative saturation and from 2′ to 2 only when core 2 makes a similar transition.

Suppose both cores originally are in the state of negative saturation, which is taken to represent 0. It is clear that the two successive magne-

[1] *Harvard Computation Laboratory Progress Rept.* 2, under USAF contract W19-122-AC-24, Sec. 4, Nov. 10, 1948.

[2] An Wang, Magnetic Triggers, *Proc. IRE*, vol. 38, no. 6, pp. 626–629, June, 1950.

[3] An Wang and Way Dong Woo, Static Magnetic Storage and Delay Line, *J. Appl. Phys.*, vol. 21, pp. 49–54, January, 1950.

[4] An Wang, *loc. cit.*

tizing pulses make no change in this situation and that no pulse appears across the output winding. Now suppose that an input pulse drives core 1 to positive saturation. The first magnetizing pulse returns this to 0, but the large flux change causes a considerable voltage to appear across coil 1, which causes current to flow through coil 1′ and drive core 2 to positive saturation. The diode prevents flow of current when core 1 is driven from negative to positive saturation. The second magnetizing pulse now drives core 2 from positive to negative saturation, and the coupling by means of coils 2 and 2′ drives core 1 to positive saturation again. Thus the net effect of the two magnetizing pulses is to return the circuit to its initial condition. Note that a pulse is available across the read-out winding during both first and second magnetizing pulses. Information can be introduced by means of the read-in winding in several ways; one way would be to use positive pulses for 1's, negative for 0's, and to drive the read-in winding in synchronism with the second magnetizing pulses.

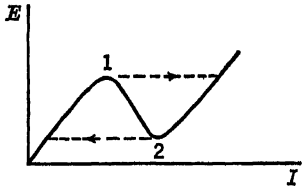


FIG. 5-24. Voltage-current relationship for nonlinear *RLC* circuit.

Wang has described also a single-core circuit[1] in which the core carries read-in, read-out, and triggering windings, the triggering windings shunted by a resistor and capacitor. A single magnetizing pulse is used, and the discharge of the capacitor reestablishes the magnetic state of the core after it has been destructively sensed by the magnetizing pulse. The original Harvard trigger pairs could be switched back and forth at rates up to about 50 kcps. Improved core materials of course permit this rate to be considerably increased, and access times to magnetic-core memories are now of the order of a few microseconds.

Memories employing magnetic cores as elements will be treated at some length in the chapters on registers and high-speed memories. This section concludes with a brief account of a magnetic memory element depending upon an effect known as "ferroresonance." The basic principle is quite simple, and has been known for a long time. It is observed that, if the amplitude of the a-c voltage impressed across a series *RLC* circuit, in which the inductor is of the iron-core type, is gradually increased, the current also increases gradually up to a certain point; at this point the current increases abruptly to a higher value without any increase in impressed voltage, after which it again increases in continuous fashion. If, now, the impressed voltage is gradually decreased, the current also decreases gradually until a certain value is reached, at which it decreases abruptly to a lower value. The situation is represented in Fig. 5-24, where the jumps occur at points 1 and 2, in the senses shown; here the

[1] *Ibid.*, pp. 628–629.

variables are supposed to be a-c voltage and current amplitudes, and the fact that harmonics must be present at the higher values of current because of the nonlinearity of the magnetization curve of the core is conveniently ignored. In a rough way, the iron-cored inductor can be considered to possess two values of "inductance," a large one for small currents, the core not being saturated at any time during the cycle, and a small one for currents of such amplitude that the core is saturated during part of each cycle.[1]

A bistable circuit based upon the effect just described has been devised by Isborn.[2] A pair of iron-cored inductors are connected as in Fig. 5-25, the extra windings being provided as a means of triggering. The capacitor C_c is such that, for the given a-c supply voltage, it is impossible for both LC circuits to be in the high-current (or resonant) condition, for then the drop across C_c would be so great, and the voltage across the cir-

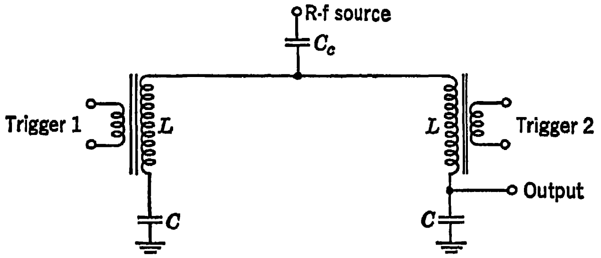


FIG. 5-25. Basic ferroresonant bistable circuit.

cuits would be so low, that both would attempt to make the transition from point 2 to the left (Fig. 5-24). Similarly, if both LC circuits were in the low-current condition, the voltage across C_c would be so small and the voltage across the LC circuits would be so large that both would attempt to make the transition from point 1 to the right. Under these circumstances, one of the LC circuits must operate in the saturated (high-current) condition and the other in the unsaturated (low-current) condition. To trigger from one state to the other is simple. For suppose that the left LC circuit is in the high-current condition, and apply a pulse to the triggering winding on the right core of sufficient amplitude to cause saturation of the right core; the inductance of the right core falls to its low value, the current through it increases, and the voltage across the left circuit is so far reduced that it is forced to shift to its low-current state.

[1] Of the large number of papers on ferroresonance in the literature we content ourselves with citing one: P. H. Odessey and E. Weber, Critical Conditions in Ferroresonance, *Trans. AIEE*, vol. 57, pp. 444–452, 1938. This is the clearest exposition that we have found, the argument being given in terms of a simple and elegant graphical construction.

[2] C. Isborn, Ferroresonant Flip-Flops, *Electronics*, vol. 25, no. 4, pp. 121–123, April, 1952.

Upon the removal of the trigger, the right core remains in the high-current state, for the voltage impressed across it and its series capacitor is sufficient so to maintain it. The trigger pulse must be of duration equal to several times a cycle of the a-c supply (say five). Output can be taken across either of the capacitors in series with the iron-cored inductors, the a-c voltage being rectified to provide d-c levels corresponding to the two states.

A circuit using but a single-trigger pulse and providing the rectified output is shown in Fig. 5-26. Triggering is accomplished by negative pulses. It is stated that, using cores of ⅛-mil 4-79 molybdenum Permalloy, it is possible to use r-f source frequencies up to 2 Mcps and to trigger at a rate of 100 kcps. Triggering voltage pulses are fairly large, of the order of 30 volts, and output (as, e.g., at point 0 of Fig. 5-26)

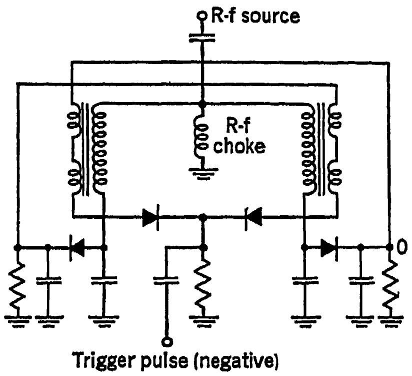


FIG. 5-26. Ferroresonant bistable circuit. (*R. W. Rutishauser, Electronics, vol. 27, no. 5, p. 152, May, 1954.*)

ratios of up to 16:1 between the two states can be obtained, with 8:1 being readily designed for; the specific circuit published by Isborn gave outputs of 3 and 25 volts, respectively.

The circuit is open to the objection that (like circuits using vacuum tubes) it continually consumes power and is thus at a disadvantage as compared to memory cells of the static magnetic type described earlier. It is, however, possible to design for very low power drain, as low, for example, as 0.05 va. A definite advantage of this circuit is that its stability with respect to variation in the frequency and amplitude of the r-f power supply is very good.

5-8. Ferroelectric Binary-memory Elements. Quite recently much interest has been shown in the possible exploitation of the so-called "ferroelectric" materials in the construction of binary-memory cells. The designation "ferroelectric" is applied to a number of dielectric materials such as rochelle salt, potassium niobate, potassium dihydrogen phosphate, and barium titanate, in which domains of permanent electric dipoles exist and which exhibit residual electric polarization after having been subjected to an electric field.[1] A plot of electric displacement or polarization versus electric field (D or P vs. E) for any of these materials has much the same appearance as the B-H curve of a ferromagnetic material, whence the name.

Of the materials mentioned above, rochelle salt exhibits the effect only in a limited temperature range (−15 to +20°C), and the phosphates exhibit it only at low temperatures. On the other hand, barium titanate ($BaTiO_3$) retains the property up to 120°C; it may be employed as a rugged ceramic material or in the form of single crystals.[2–4] The curve of Fig. 5-27 plots polarization versus applied voltage for a single crystal 0.0045 in. thick used as the dielectric of a capacitor, to the plates of which a 60-cps sinusoidal voltage of amplitude 100 volts was applied. If the

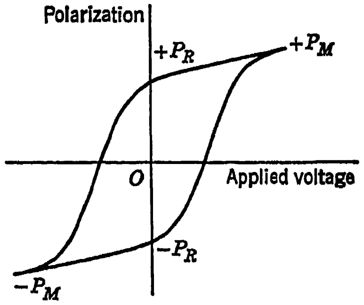


FIG. 5-27. Hysteresis curve of "ferroelectric" material.

amplitude were smaller, a similar curve of smaller size would be obtained, whereas, if the exciting frequency were raised, a flattened curve would result.[5]

For use in memory devices a loop as nearly rectangular as possible is desired, with low coercive force and high residual polarization. This excludes barium titanate ceramic in the form prepared for use as the dielectric of capacitors, since this exhibits a badly nonrectangular hysteresis loop and its coercive force is high. Single crystals have good characteristics, but growing them is a problem and they are not commercially

[1] A. von Hippel, Ferroelectricity, Domain Structure, and Phase Transitions in Barium Titanate, *Revs. Modern Phys.*, vol. 22, no. 3, pp. 221–237, July, 1950.

[2] D. A. Buck, Ferroelectrics for Digital Information Storage and Switching, *MIT Digital Computer Lab. Rept.* R-212, June 5, 1952.

[3] J. R. Anderson, Ferroelectric Elements for Digital Computers and Switching Systems, *Elec. Eng.*, vol. 71, pp. 916–922, October, 1952.

[4] C. F. Pulvari, An Electrostatically Induced Permanent Memory, *J. Appl. Phys.*, vol. 22, no. 8, pp. 1039–1044, August, 1951.

[5] Anderson, *op. cit.*, gives the curve reproduced in Fig. 5-27.

available in quantity. However, if the ceramic material is formed into a thin sheet (about 0.2 mm thick) and is subjected to crystallization conditions, apparently some minute crystals tend to grow through the thickness dimension of the material, and the resulting highly oriented material has quite good characteristics.[1]

A basic binary-memory element is shown in Fig. 5-28. Let positive residual polarization represent binary 0; let negative represent binary 1. Hence, to insert a 1 into the cell, a negative voltage pulse is applied at the read-in terminal; a 0 is inserted by a positive pulse. As in the case of the static magnetic memory cells, sensing is destructive, and in this case is performed by a positive read-out pulse. In this chapter means of regenerating the information after destructive read-out will not be discussed. If a 1 is held, the read-out pulse causes a large change in polarization, and the curve of Fig. 5-27 is traversed from $-P_R$ upward and to the right to P_M. Since the slope of the curve is very steep during part of this traversal the capacitance is large; if it is large compared with C, as it

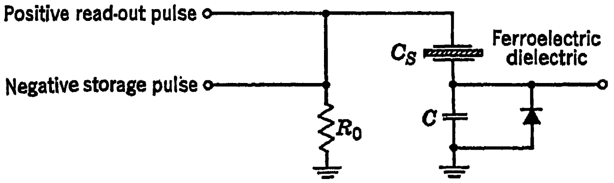


FIG. 5-28. Basic ferroelectric memory cell. (*J. R. Anderson, Elec. Eng., vol.* 71, *p.* 918, *October,* 1952.)

will be if C has been correctly chosen, most of the pulse voltage must appear across C. On the other hand, if the cell holds 0, the read-out pulse causes the curve to be traversed only from $+P_R$ to P_M; so the crystal exhibits a capacitance small compared with C (if C has been correctly chosen) and, as a result, most of the voltage appears across the crystal and little across the capacitor C. It is not hard to show that, if C_S'' is the minimum value of capacitance exhibited by the $BaTiO_3$ element, if X is the ratio of the maximum value of this capacitance to C_S'', and if R is the desired ratio of output voltage when reading 1 to output voltage when reading 0, then C should have the value

$$C_S'' \frac{R-1}{X-R}$$

Cells of the type discussed have been tested by Anderson, who used a single crystal about 0.1 sq in. in area and about 0.004 in. thick, with small

[1] C. F. Pulvari, The Snapping Dipoles of Ferroelectrics as a Memory Element for Digital Computers, *Proc. Western Computer Conf., Feb. 4–6, 1953*, pp. 140–159, Institute of Radio Engineers, New York, 1953.

silver-paste electrodes 0.02 in. in diameter on opposite faces. The value of C was 0.01 μf, and read-in and read-out pulses of 30 volts amplitude and 5 μsec duration were used to give output pulses of 25 volts amplitude for 1 and 0.6 volt amplitude for 0. If a shorter (0.75 μsec) pulse of smaller amplitude (10 volts) was used, this excellent discrimination between 1 and 0, amounting to a value of R greater than 40, fell to about 3. As to the permanence of the information, Anderson found that, if the material is actually driven to saturation, no deterioration is noted even after several days, but that, if one operates on a low-voltage hysteresis loop, deterioration sets in after a few hours. Switching in a microsecond or less has been obtained by other experimenters[1] also and may be regarded as an established fact.

To date, ferroelectric memories have not emerged from the experimental stage. This is due to the lack of commercial quantities of suitable materials. If sufficient interest is shown in this means of memory, no doubt the manufacturers will in time respond with greatly improved materials, just as they have over the years greatly improved the commercially available ferrite cores. A defect of ferroelectric memory elements in comparison with magnetic elements is the comparatively high dissipation of energy during switching. This dissipation is of the order of one hundred or more times the value for ferrites, for example, 5,000 joules per cu m for $BaTiO_3$ single crystals as against 4 to 90 joules per cu m for ferrites. This could lead to a severe cooling problem, but, fortunately, $BaTiO_3$ ceramic can be formed into thin sheets that are easily cooled.

[1] Pulvari and Buck, for example.

CHAPTER 6

DYNAMIC MEMORY CELLS

6-1. Introduction. The memory elements described in Chap. 5 were all such that information once inserted is retained without deterioration until it is deliberately disturbed. In this chapter devices will be considered in which the information deteriorates rapidly and must, therefore, be regenerated periodically. Two of these are schemes for maintaining the charge on a capacitor at either of two values. In the third the information is caused to circulate about a loop in which a delay line is inserted. Under these circumstances the pulse shape deteriorates very rapidly and must be periodically restored; hence the memory cell is essentially a pulse restoration or regeneration circuit with feedback from output to input through a delay line. The thoroughly dynamic character of such memory cells must be emphasized. The best-known system is that developed at the National Bureau of Standards (NBS) and used in the SEAC computer and those machines whose designs evolved from SEAC. As these all use transformer coupling between stages, a discussion of pulse transformers is included; electrical delay lines also will be treated briefly in this chapter.

6-2. IBM Microsecond-delay Unit. The memory cells of which the registers of the IBM 701 electronic data-processing machine were built up are known as "microsecond-delay units."[1] The basic principle used is that of periodic restoration of the charge on a capacitor every microsecond for as long as it is desired that the information be held. Of course, provision is also made for altering the charge on the capacitor so that the information held can be changed.

In the actual circuit, shown in simplified form in Fig. 6-1, binary 0 is represented by -30 volts and binary 1 by $+10$ volts. The "peaker" is a triode with a parallel *RL* (and, of course, distributed *C*) circuit in the plate lead, driving a cathode follower. A positive pulse input causes an output corresponding to one cycle of a damped sine wave, which starts by swinging negative. The "synch" voltage remains at -30 volts for six-tenths of each microsecond, then swings up to $+10$ volts for the

[1] H. D. Ross, The Arithmetic Element of the IBM Type 701 Computer, *Proc. IRE*, vol. 41, no. 10, pp. 1287–1294, October, 1953.

remaining four-tenths. When "synch" falls, "clamp" falls from +10 to −40 volts for about 0.2 μsec, and then swings back to +10 volts.

Suppose the memory capacitor C_2 is charged to −30 volts. If −30 volts is impressed at the input, nothing happens when the synch pulse is applied to the gate. The fall of the clamp, however, to −40 volts permits C_2 to regain any charge that may have leaked off and, at its expiration, −30 volts appears across C_2. If, on the other hand, +10 volts is

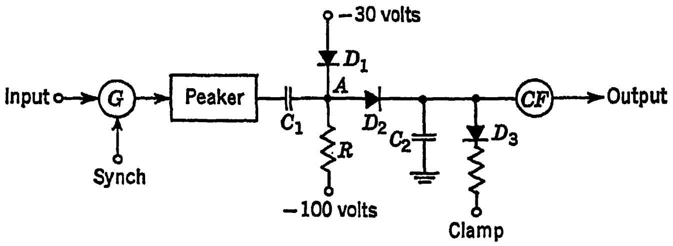


FIG. 6-1. Microsecond-delay unit.

impressed at the input, the gate passes the synch pulse and the peaker is activated. The negative swing of output voltage cannot pull point A down below −30 volts, and C_1 charges during this period. When the peaker output swings positive, point A rises and C_2 charges positively. When the clamp voltage returns to +10 volts, any excess charge is drained off by D_3, and the voltage across C_2 is established at +10 volts. It has been shown (a) how 0 is restored and (b) how 0 is replaced by 1. The reader will have no trouble in treating the two remaining cases.

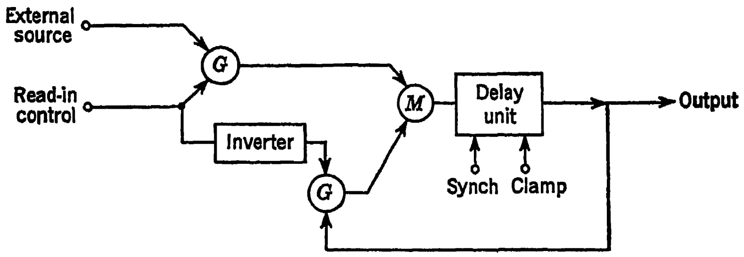


FIG. 6-2. Input and regeneration paths of the delay unit.

The regeneration and read-in paths to a delay unit are shown in Fig. 6-2. The read-in control is normally held at −30 volts, which (1) prevents information from entering from the external source, and (2) produces a +10-volt output from the inverter, thus enabling the gate in the feedback path. When new information is to be inserted, the read-in line is pulsed up to +10 volts, which permits information to flow in from the external source and which disables the gate in the feedback path.

6-3. NBS Diode-Capacitor Memory Cell. A development of the NBS, the memory cell to be described (the basic circuit of which is shown in

Fig. 6-3) uses a capacitor as the memory element and provides means of periodically restoring the charge of the capacitor.[1] In the circuit shown, the levels chosen were +2 volts from point *B* to point *D* to represent 1, and −2 volts to represent 0.

Before the action of the circuit is described, the block labeled "delayed gate" in Fig. 6-3 should be noted. This unit consists of an amplifier with output gates and delay elements so arranged that a slightly delayed negative signal is sent out on the line marked " − " if the input is negative and a positive signal on the line marked " + " if the input is positive; for inputs of 1 volt or more these outputs exceed 2 volts in magnitude.

Suppose that points *A* and *C* are held at ground and that a negative signal arrives at the input. The net effect is to pull the voltage of point *D* down after a slight delay, but the diode D_1 permits *D* to fall only to −2 volts. As *D* falls, the capacitor C_1 starts to charge through the low

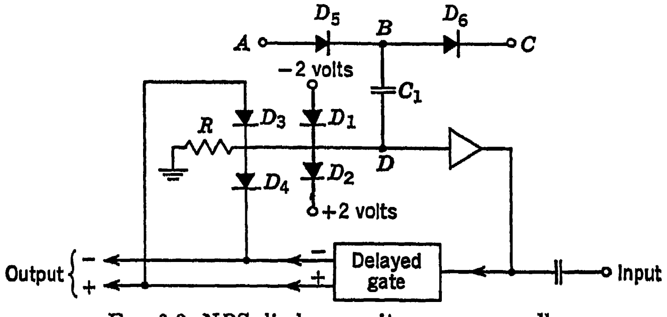


FIG. 6-3. NBS diode-capacitor memory cell.

forward resistance of the diode D_5; suppose that the input lasts long enough to permit C_1 to charge essentially to 2 volts. The next step is to remove the input and to establish points *A* and *C* at −4 and +4 volts, respectively. The capacitor C_1 then proceeds to discharge slowly through the high back resistances of D_5 and D_6. Note that, if the input had been positive instead of negative, the voltage across C_1 would have ended up at −2 volts.

The storage capacitor is permitted to discharge to ±1 volt, and thereupon its charge is restored. Suppose that it is positively charged and that, at the time the voltage reaches +1 volt, the points *A* and *C* are both suddenly grounded, thus also grounding point *B*. The voltage at *D* drops abruptly to −1 volt, and this signal is amplified, delayed, and gated out on the line marked " −"; point *D* is forced down to −2 volts, and the storage capacitor rapidly charges to 2 volts via D_5. When *A* and

[1] Holt, A. W., A Preliminary Report on the Prototype Diode-Capacitor Memory, *Proc. Argonne Conf. on Digital Computers*, August, 1953. See also *NBS Rept.* 2713, June, 1953, and *NBS Rept.* 3374, May 1, 1954.

C are once more set at -4 and $+4$ volts, respectively, the storage capacitor's voltage has been restored. This restoration process makes available a negative pulse which can be used as an output. Thus a negative pulse input results in a positively charged capacitor but in a negative output pulse that is available every time the regeneration process is carried out; a similar remark naturally also applies to a positive pulse input.

In the actual circuit, a value of 2 kilohms was used for R, and the storage capacitor was 10,000 pf if ordinary germanium diodes were used. Regeneration had to be performed only at intervals of several hundred microseconds. In practice, however, since regeneration must take place in order to read, this operation is carried out much more frequently. In an experimental memory at NBS, regeneration was accomplished every 6 μsec. Reading required $\frac{1}{4}$ μsec and writing $\frac{1}{5}$ μsec. With higher-quality diodes, leakage is reduced and the requirement on regeneration is relaxed. Recent work using silicon junction diodes has indicated that operation on a 1-μsec cycle is obtainable.

The circuit of Fig. 6-3 as it stands is obviously rather wasteful of equipment. It is possible to arrange in a matrix a great number of storage capacitors, each with two diodes (the D_5 and D_6 of Fig. 6-3), each column served by a single amplifier and delayed gate, so that a reasonably economical memory results.

6-4. Pulse Restoration or Regeneration. Inasmuch as physical circuits approximate only more or less satisfactorily to the ideal characteristics for transmission of pulses, pulses are bound to suffer a good deal of distortion and attenuation as they move about the machine. The attenuation is of course a reduction in amplitude, and the distortion takes the form of a "smearing out" or widening accompanied by rounding of corners and decrease of slope of edges. It is, therefore, necessary in many cases to provide means of reshaping pulses at various places in the circuitry. This may be done in a variety of ways. Some of these will be described in order to lay the foundation for a discussion of a dynamic memory device which consists of a pulse-regeneration circuit with a delayed feedback.

A very simple method of pulse restoration is to use a monostable ("one-shot") multivibrator. If, however, the pulse used to drive it is badly smeared, the time at which triggering takes place is rather uncertain. If such a slight uncertainty in the timing of the pulse can be tolerated, this method should be satisfactory. However, in serial computers, precise timing is commonly required.

Another simple scheme is shown in Fig. 6-4. The pulses to be restored or standardized are originally positive, but have been amplified and inverted before arriving at the input. In the absence of a pulse input, the pentode conducts strongly. The negative input pulse cuts the pentode off, thus setting up a transient in the circuit that consists of the inductor, diode, 270-ohm resistor, and the distributed capacitance of the inductor. The end of the inductor connected to the plate of the tube starts by swinging through the positive half cycle, the diode not conducting. When the negative half cycle begins, the diode conducts, and the resistor rapidly dissipates the remaining stored energy; so the output is essentially the positive arch of a sine curve, of period determined by the inductance and the distributed capacitances of the coil.

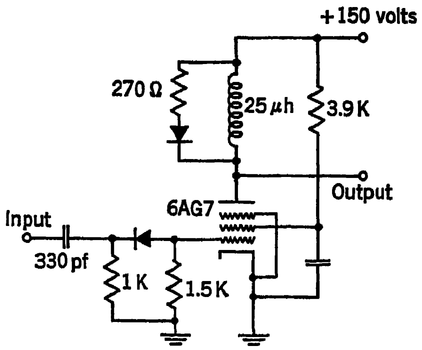


FIG. 6-4. Pulse standardizer. (*R. R. Rathbone, Radio and Television News, November*, 1951, *p.* 6.)

A transistor circuit for pulse standardization has been discussed in Sec. 5-6. A transistor one-shot multivibrator is, of course, equally applicable,[1] and numerous circuits of this type have been published.[2-5] Typical circuits for both point-contact and junction transistors are shown in Fig. 6-5.

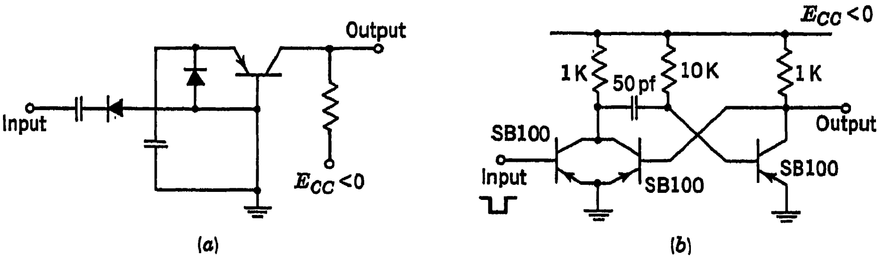


FIG. 6-5. Transistor one-shot multivibrators. (*a*) Point contact; (*b*) junction (surface barrier).

An important scheme, which forms the basis of the dynamic memory cell to be described in Sec. 6-5, will now be described. This was developed for use in serial machines. In principle it is very simple; the restoring circuit accepts as inputs both information pulses and the clock wave-

[1] A. W. Lo, Transistor Trigger Circuits, *Proc. IRE*, vol. 40, no. 11, p. 1531, 1952.

[2] H. G. Follingstad, J. N. Shive, and R. E. Yeager, An Optical Position Encoder and Digit Resolver, *Proc. IRE*, vol. 40, no. 11, pp. 1573–1583, 1952.

[3] J. R. Harris, A Transistor Shift Register and Serial Adder, *Proc. IRE*, vol. 40, no. 11, pp. 1597–1602, 1952.

[4] R. H. Beter, W. E. Bradley, R. B. Brown, and M. Rubinoff, Surface Barrier Transistor Circuits, *IRE Convention Record, 1955*, pt. 4, pp. 139–145.

[5] Lloyd P. Hunter (ed.), "Handbook of Semiconductor Electronics," McGraw-Hill Book Company, Inc., New York, 1956. See sec. 14 by E. Eberhard and sec. 15 by J. C. Logue.

form. Whenever an information pulse is received, a standard pulse derived from the timing waveform is transmitted. A simplified block diagram is shown in Fig. 6-6. As developed at NBS, the timing waveform is a pure sine wave, and the information signals are positive pulses for binary 1's, nothing at all for 0's.[1] A positive information pulse arriving at the input passes through the mixer, and is amplified and limited and fed back to open the gate for the standard clock signal. The clock signal is now mixed with the information pulse, amplified and limited, and fed back to the gate. Thus the output pulse lasts until the termination of the positive portion of the timing waveform. This scheme, which was suggested by earlier work on the EDVAC and UNIVAC, has received intensive development and exploitation at the NBS, and is aptly described as "regenerative reclocking." Section 6-5 is devoted entirely to circuits based upon this idea, in particular to the so-called "dynamic flip-flop."

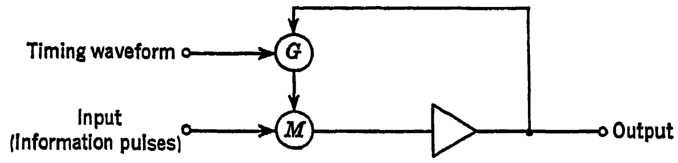


FIG. 6-6. Regenerative reclocking.

6-5. Basic NBS Circuit. It has been observed in Chap. 3 that the circuit philosophy that dominated the development of both SEAC and DYSEAC at the NBS was simply to perform all logical functions by means of crystal-diode structures and to use vacuum tubes solely as power amplifiers.[2,3] Thus the mixer and gate of Fig. 6-6 are of the diode type described in Chap. 4, but the amplifier is simply a miniature beam-power tube 6AN5 with a pulse transformer in the plate circuit. If a slight delay is inserted in the feedback path, so chosen that the transmission time about the loop is one pulse-repetition period, a binary storage cell results. A longer delay permits the storage of several bits and can be utilized in the building up of one-word storage devices, or registers, which will be described in a later chapter. Because of the versatility of the gate-mixer-amplifier combination, it was found possible, in the design of DYSEAC, to use a single "package" of this type, which, with different types of feedback, can be used as pulse repeater, binary storage cell, or part of a register.

[1] R. D. Elbourn and R. P. Witt, Dynamic Circuit Techniques Used in SEAC and DYSEAC, *Proc. IRE*, vol. 41, no. 10, pp. 1380–1387, October, 1953.

[2] *Ibid.*

[3] S. Greenwald, R. C. Haueter, and S. N. Alexander, SEAC, *Proc. IRE*, vol. 41, no. 10, pp. 1300–1313, October, 1953.

The basic package is shown schematically in Fig. 6-7. Five gates are provided at the input; their outputs are fed to a mixer. The two diodes in the grid circuit of the tube limit grid excursion; the negative level is so chosen that the 6AN5 conducts slightly when there is no pulse input. This has two effects: (1) it prevents the formation of cathode interface; and (2) it protects the grid from noise as well as holding the tube in the cut-off state would, and does not require as large pulse inputs, much of

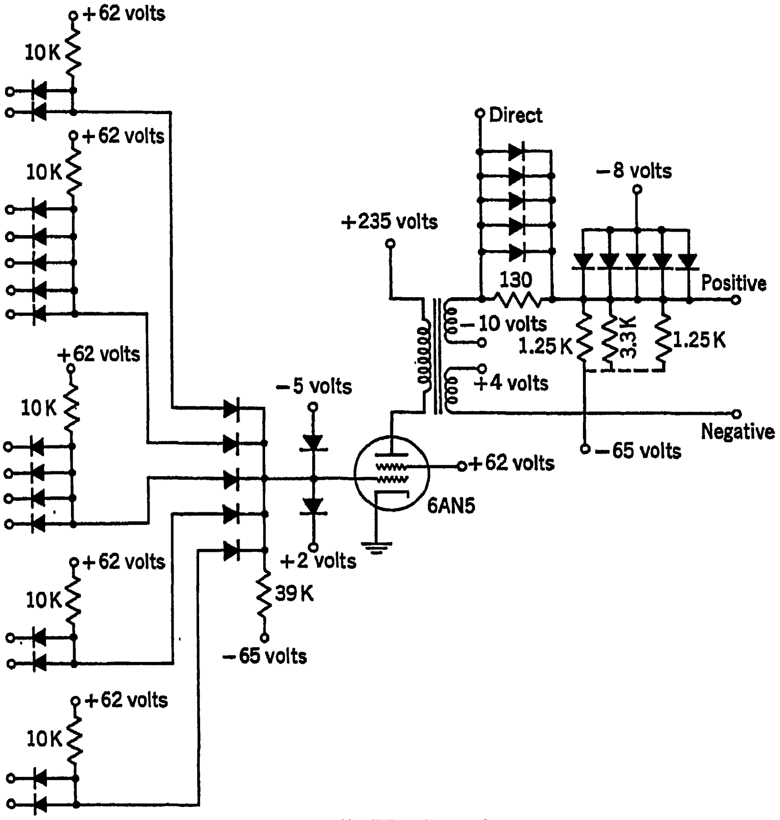


FIG. 6-7. NBS basic package.

which would be wasted in charging the grid capacitance through a region of low transconductance. The 10- and 39-kilohm resistors in the gates and mixers, respectively, are so chosen that the grid-circuit parasitic capacitances are charged and discharged at a rate of at least 75 volts per μsec at the beginning and end of each pulse.

The pulse transformer (see Sec. 6-6) has a 60-turn primary and two 6-turn secondaries, thus yielding both positive-going and negative-going output pulses, which are about 20 volts in amplitude. The quiescent levels of the positive- and negative-going outputs are a little below -8 and $+4$ volts, respectively. The former level is established by the clamping diodes. No current flows in the transformer secondary under these circumstances, and the current needed to hold down input gates of other packages driven by the positive output lead is supplied by the -65-volt source. One or more of the three resistors (1.25, 1.25, and 3.3 kilohms) are connected, according to the number of gates that must be held down, up to five each for the 1.25-kilohm resistors and two for the 3.3-kilohm resistor. The total rated driving capacity of the package is 14.5 gates, of which 12 can be driven by the positive output and the others by the "direct" or negative outputs. During the pulse, the positive lead swings up to about $+12$ volts, and the diodes shunting the 130-ohm resistor conduct, so the resistor does not contribute to the load. At the termination of a pulse, any negative overshoot is damped (somewhat less than critically) by the 130-ohm resistor. The necessity of damping out this back transient limits the maximum duty cycle of the circuit to about 50 per cent.

With the stepdown transformer in the output circuit, the tube sees a very small load. The load line actually intersects the plate characteristic for $e_g = +2$ volts far below the "knee." This type of operation, known as "bottoming," results in almost uniform voltage output from tubes exhibiting a 2 :1 variation in current capability. As the minimum plate-current swing between $e_g = -5$ and $e_g = +2$ volts is at least 42 ma, large secondary current is available.

It is for this reason that the circuit is able to drive heavy loads. It has already been mentioned that the maximum resistive load is equivalent to 14.5 gates; in addition to this, up to 300 pf capacitance may be present. If only nine gates are driven, up to 1,000 pf capacitance may be tolerated. The low effective impedance of this circuit is such that it is possible to use long (e.g., 50-ft) unshielded wires between stages without running into trouble from noise.

6-6. NBS Dynamic Flip-Flop. It is now possible to describe the dynamic binary memory circuit based upon the NBS pulse-amplifier package described in the last section. Both SEAC and DYSEAC are synchronous serial computers.[1,2] The timing waveform is a 1-megacycle sine wave (from 30 to 45 volts peak to peak in DYSEAC); all pulses are derived from it and are ½ μsec in duration. The transmission delay caused by a single pulse-amplifier package is very nearly ¼ μsec. For this reason, the timing signal is distributed as four phases of a 1-megacycle sine wave, spaced 90° apart, and successive packages are timed by successive phases of the clock. To clarify the process, refer to Fig. 6-8,

[1] *Ibid.*

[2] Elbourn and Witt, *op. cit.*

which shows phases 1 and 2 of the clock, with the grid clipping levels +2 and −5 volts and with the pulse output from the first stage, which is given as timed by phase 1. The outputs of stage 1 and phase 2 are used as inputs to one of the gates of stage 2. To a second gate are delivered phase 2 and the "direct" output of stage 2. Hence (referring to Fig. 6-8) although the input pulse crosses the −5-volt level at point *a*, it is clear that the output of the gate to which it is fed does not rise until point *b*. The gate output passes through the mixer and is clipped at +2 volts by the grid-circuit diodes, amplified, and fed back to the second gate—to which phase 2 is also an input—and returned via the mixer to the grid circuit of the amplifier tube. Thus, although at point *e* the first gate is disabled, the second gate continues to pass a signal to the mixer and hence to the grid until the phase 2 signal falls below −5 volts (point *f*). The grid input of stage 2 is thus the precisely timed trapezoidal pulse *bcdf*. This scheme allows a precise determination of the time at which a given information pulse will arrive at the grid of any stage in the computer.

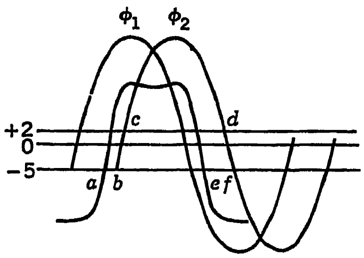


FIG. 6-8. Regenerative reclocking.

To obtain a dynamic memory circuit, a second feedback path is added in which a delay of ¾ μsec is inserted by means of a delay line (see Sec. 6-8); so the total delay around the loop is 1 μsec. A block diagram is

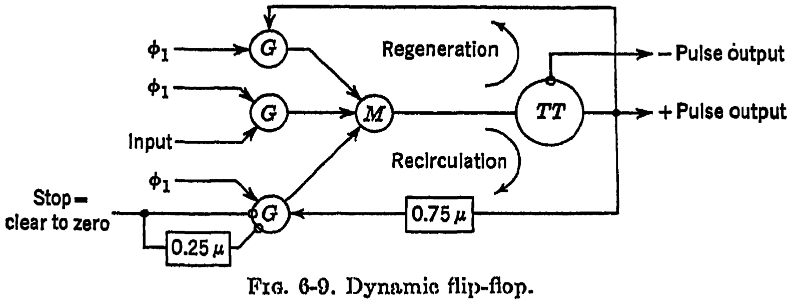


FIG. 6-9. Dynamic flip-flop.

shown in Fig. 6-9, where the tube-transformer combination is represented by a circle labeled *TT*. From this are taken a positive output and also a negative output, indicated by the small loop drawn about the circle. The upper loop (regenerative) is as described in the preceding paragraph, and in the lower loop is inserted the delay line. It is assumed that phase 1 is the appropriate clock phase. To the gate in the lower or recirculation loop, one input is the stop signal, which inhibits the gate and hence prevents the delayed pulse from being fed back to the mixer; it may as well be called a "clear-to-0" input. If the circuit holds 0, nothing whatever happens; if it holds 1, a pulse constantly recirculates and is available at the output at intervals of 1 μsec.

6-7. Pulse Transformers. The use of transformer coupling in the SEAC circuitry has been described above. It is used extensively also in machines whose design principles are quite different, notably in Whirlwind I. An obvious advantage is that reversal of polarity is avoided in a natural and evident way; if capacitive coupling were used, some means of correcting such reversal would have to be provided, either by inserting extra tubes or by operating alternate stages in a quiescent conducting condition, with attendant waste of power. An additional benefit is that impedance matching is facilitated. Thus, for example, in Whirlwind I, low-impedance coaxial line is used to transmit pulses to remote points, and matching at both ends is accomplished by pulse transformers.[1] The use of pulse transformers in SEAC to provide such large output currents that there is no need in the design to take excessive pains to minimize wiring capacitance between stages has already been considered. In a work such as this it would be impractical to enter into a discussion of the theory and design of pulse transformers; hence only a few general observations are included.

To provide good reproduction of pulse shape,[2] a pulse transformer should be designed to meet the following requirements:

1. The decay of output pulse amplitude during the pulse should be small (assuming that the input pulse is flat-topped).
2. The rise and fall times should be small.
3. Ringing and general distortion should be minimized.

It can be shown that requirement 1 can be met by making the inductance of the primary large; this can be done by using many turns in the primary, by using a high-permeability core, and by using a core of large cross-sectional area. Requirement 2 can be met by making the product of leakage inductance by interwinding capacitance small; this can be done by using a small number of turns on the primary and a small winding circumfereuce. It is, therefore, necessary to make a compromise between these conflicting requirements. Finally, requirement 3 can be met by making leakage inductance and interwinding capacitance with reference to the primary small, but in such a way that a certain ratio is preserved.

[1] C. A. Rowland, Pulse Transformers and Interstage Coupling in Whirlwind I, *MIT Servomechanisms Laboratory Eng. Note* E-328, Jan. 31, 1950.

[2] T. F. Wimett, Low Power Pulse Transformers, *Project Whirlwind Tech. Rept.* R-122, submitted under ONR contract N5ori60, December, 1947. The present discussion is summarized from this report.

From the requirements stated in the above paragraph it is possible to deduce some general design principles:

1. The core material should have as high a value of permeability as possible and should produce the lowest possible eddy current and hysteresis losses. For this reason, in the earlier pulse transformers, cores were of thin laminations of such materials as Hipersil; recently ferrites have become feasible and widely used.
2. The core itself should present the shortest possible magnetic path length consistent with its being able to hold the windings. Cross-sectional area must be the best possible compromise consistent with the need to hold down the length of wire in the windings.
3. The number of turns in the primary (and therefore in the other coils) must be determined as the most acceptable compromise between pulse decay and rise and fall times.
4. The winding with the greatest number of turns should be of the smallest-size wire that can be used without danger of breaking.
5. Yet another compromise involves the characteristic impedance, or the load into which the transformer will deliver the least distorted output. This is proportional to the spacing between primary and secondary, which also directly affects leakage inductance and interwinding capacitance.

The original Whirlwind I pulse transformers were designed to pass pulses of the shape of one arch of a sine wave and of 0.1 μsec duration. They use laminated metallic cores. A variety of experimental designs are described in the report by Wimett, cited above. As an example of ferrite-core transformers, a design used at the NBS may be cited. This uses a core of sintered manganese-zinc ferrite. The core is very small, shaped like a flat cylinder $\frac{5}{16}$ in. in height and $\frac{9}{16}$ in. in diameter, and actually composed of two parts, each the general shape of an angel-cake tin, which are held together by a screw. Thus the windings are completely enclosed. These were designed to pass rectangular pulses of 0.5 μsec duration. Descriptions of these have been published.[1,2]

6-8. Electrical Delay Lines. Electrical delay lines were first encountered in the account of the NBS dynamic flip-flop. They are also used in the arithmetic registers of SEAC and the machines influenced by SEAC design, and in a variety of other machines. Therefore, a brief account will be included here.

The desideratum in all applications is to introduce delay without pulse distortion. This would require the use of a suitable length of ideal lossless

[1] Elbourn and Witt, *op. cit.*

[2] *NBS Tech. News Bull.*, vol. 36, no. 5, pp. 78–79, May, 1952.

transmission line or of distortionless lossy line. A moment's reflection is sufficient to determine that this would lead to the use of impractically long pieces of line, since delays from 1 up to perhaps 50 μsec are required. Since ordinary coaxial line is thus ruled out, the natural thing is to have recourse to artificial lines of various sorts, and it is these that will be considered.

Delay lines first became important during World War II, and both of the principal types were used: (1) the lumped-constant lines and (2) the distributed-constant lines.[1] The theory of the lumped-constant line was well known,[2] and the theory of the distributed-constant type is essentially that of ordinary transmission lines.

Since lumped-constant artificial lines are essentially cascades of low-pass filter sections, considerable departure from ideal characteristics must be tolerated, such as distortion in shape and attenuation. The following important characteristics of delay lines have been listed: (1) total delay, (2) rise time of the output pulse (e.g., the time required to rise from 10 to 90 per cent of peak amplitude), (3) the density of pulse spacing that may be achieved without disastrous interference between adjacent pulses, (4) insertion loss for pulse transmission, (5) volume, (6) pulse distortion due to phase distortion and cross talk, (7) stability of delay as a function of temperature, (8) characteristic impedance, (9) adjustability of delay time, (10) cost, (11) maximum voltage that can be tolerated across the line.[3]

Lumped-constant delay lines are probably less popular than distributed-constant lines. A useful form of the latter may be regarded as the limiting form obtained if, starting with a cascade of T sections, each having inductors in the series arms and a capacitor in the shunt arm, one proceeds to the limit, letting the inductors merge into a continuous coil and replacing the capacitors by the distributed capacitance between the coil and a cylindrical metallic core.[4] A typical delay line of this type is described as producing a delay of 0.55 μsec per ft, with an attenuation of 1.3 db per μsec, and as presenting a characteristic impedance of 1,100 ohms. The version of this line used in SEAC and DYSEAC[5] is built up

[1] Britton Chance, F. C. Williams, V. W. Hughes, D. Sayre, and E. F. MacNichol, Jr., "Waveforms," McGraw-Hill Book Company, Inc., New York, 1949. Chapter 22 is devoted to the theory of delay lines. Design is treated in detail in John F. Blackburn (ed.), "Components Handbook," chap. 6, McGraw-Hill Book Company, Inc., New York, 1949.

[2] E. A. Guillemin, "Communication Networks," vol. 2, chap. 7, John Wiley & Sons, Inc., New York, 1935.

[3] J. R. Anderson, Electrical Delay Lines for Digital Computer Applications, *IRE Trans. on Electronic Computers*, vol. EC-2, no. 2, pp. 5–13, June, 1953.

[4] J. P. Blewett and J. H. Rubel, Video Delay Lines, *Proc. IRE*, vol. 35, no. 12, pp. 1580–1584, December, 1947.

[5] Elbourn and Witt, *op. cit.*

of (1) an inner core of plastic tubing, (2) a close-wound helix of very fine wire upon the tubing, (3) a sprayed layer of aluminum paint, (4) two layers of Teflon tape, (5) a grounded braid of insulated wires, and (6) an outside protective cotton covering. This is the delay line used in the dynamic flip-flops and in the registers, and it transmits $\frac{1}{2}$-μsec pulses without intolerable distortion. It has proved very satisfactory.

In the SEAC circuits, about 5 μsec of delay line is the maximum length driven by any one tube. In many applications it would be desirable to use a single tube to drive a length of line sufficient to hold an entire word, but attenuation and distortion restrict the length that can be used without some form of pulse reshaping. Anderson has carried out an experimental investigation from which he concluded that, for existing delay lines of both types, the maximum number of pulses seems to be about 23, irrespective of line length. He has adduced a theoretical argument to show that the number is indeed limited by a quantity proportional to the resultant quality factor Q' of the inductance and capacitance per unit length.[1] Thus, if

$$\frac{1}{Q'} = \frac{1}{Q_L} + \frac{1}{Q_C}$$

where Q_L and Q_C have the usual meanings of $\omega L/R$ and $\omega C/G$, rise time is given by the expression

$$\frac{3.078T}{Q'}$$

where T is the total delay of the line; and the total number of pulses that can be present on the line, assuming that intolerable distortion will result if the pulse length is less than twice the rise time, is

$$\frac{Q'}{6.156}$$

As generally $Q_L \ll Q_C$, the total number of pulses can, presumably, be increased most easily by increasing Q_L. Anderson has proposed, therefore, a lumped-constant line using high-quality inductors of inductance equal to about 1 μh and having a Q of about 200, consisting of a short length (about 1 in.) of fine wire passing through a small hole drilled in a block of nickel-zinc ferrite. Assuming that Q_C is of the order of ten times Q_L, this should produce a line capable of storing 32 pulses.

Continuously wound delay lines suffer from the inconvenience of such considerable dispersion that the storage of more than a few pulses in a single piece of line is impracticable and the designer is forced to break up the total desired delay into a number of short sections with pulse amplifiers and reshapers interposed between them. A lattice-type delay struc-

[1] Anderson, *op. cit.*

ture has rather better characteristics and can be used to hold up to 40 pulses at a 500-kcps repetition frequency.[1] The lattice impedances are of the forms given in Fig. 6-10.

Let f be a frequency, the significance of which will become apparent later; in A (Fig. 6-10), let $L_j = [(2j - 1)^2 4\pi^2 f^2 C]^{-1}$; and, in B, let

$$L_j = [(2j)^2 4\pi^2 f^2 C]^{-1}$$

when all the capacitors are of the same capacitance C. Then, in A, the jth resonant circuit is tuned to $(2j - 1)f$ and, in B, to $2jf$. If both are

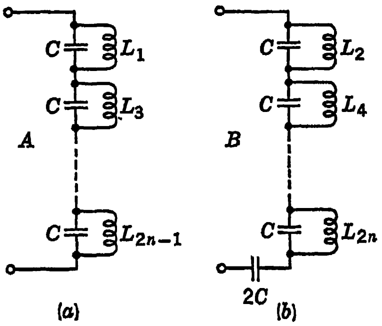


FIG. 6-10. Impedances in delay lattice.

excited by a unit current impulse or delta function, then the voltages appearing across the input terminals are given by

$$V_A = \frac{1}{C}\,[\cos 2\pi ft + \cos (3 \times 2\pi ft) + \cdots + \cos (2n - 1)2\pi ft]$$

$$V_B = \frac{1}{C}\,[\tfrac{1}{2} + \cos (2 \times 2\pi ft) + \cos (4 \times 2\pi ft) + \cdots + \cos (2n - 2)2\pi ft + \tfrac{1}{2} \cos 2n2\pi ft]$$

or, using known formulas,

$$V_A = \frac{n}{C}\,\frac{\sin (2n)2\pi ft}{2n \sin 2\pi ft}$$

$$V_B = \frac{n}{C}\,\frac{\sin (2n)2\pi ft}{2n \tan 2\pi ft}$$

In both cases, the voltage at $t = 0$ is n/C and, thereafter, oscillates. Then the amplitude varies inversely with sin $2\pi ft$ and tan $2\pi ft$, reaching extreme values of $\pm n/C$ when these functions both vanish, that is, at $t = 0, 1/2f, 2/2f, 3/2f, \ldots$ —the peaks at $1/2f, 3/2f, \ldots$ being of opposite sign and those at $0, 2/2f, 4/2f, \ldots$ being of the same sign. The difference

[1] G. G. Scarrott, W. J. Harwood, and K. C. Johnson, Electromagnetic Delay Networks for Digital Storage, paper delivered at IEE Convention on Digital Computer Techniques, London, 1956.

of these two voltages, which can be written in the form

$$V_o = V_B - V_A = \frac{n}{C}\frac{\sin 4n\pi ft}{2n \sin 2\pi ft}(\cos 2\pi ft - 1)$$

is zero at $t = 0$, possesses a peak value of $2n/C$ at $t = 1/2f, 3/2f, \ldots$, and clearly vanishes at $1/2f \pm 1/4nf$. Thus any circuit that develops this voltage can be thought of as introducing a delay of $1/2f$ sec, and the input delta function of current emerges as a voltage pulse of width $1/2nf$ and of shape something like a half sine wave. Because V_o vanishes at $t = 0, 1/4nf, 2/4nf, \ldots$, it appears that input pulses spaced $1/4nf$ apart should give distinguishable outputs. Ideally, the pulse pattern would repeat, but, given the usual resistances associated with the coils, the second pulse, which occurs with a delay of $3/2f$, is so attenuated as to be negligible. The delay of $1/2f$ can thus, ideally, be used to store

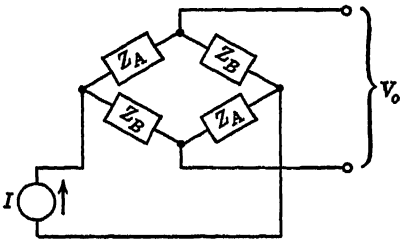


FIG. 6-11. Delay lattice.

$2n$ pulses, or one pulse per coil. However, better pulse shape can be obtained by a slightly more lavish use of elements. The design considerations will not be discussed here, but an example of a practical design will be cited. In this, the number of coils required is $4n/3$ instead of n to store n pulses, or 36 coils for 27 pulses, at a prf of 500 kcps. Thus each network contains 18 resonant circuits. Assume that 20 $\mu\mu$f of distributed capacitance exists across the terminals to which each network is to be connected. It can be shown that each C can be reasonably chosen equal to n times the distributed capacitance, which leads to a value of at least 360 $\mu\mu$f for C. Assuming that the input pulses are 1 μsec in length, the total delay introduced by the network must be $27 \times 2 - 1 = 53$ μsec. As this must equal $1/2f$, the value $f = 9.434$ kcps is determined, and the inductances of the coils are determined as $L_1 = 0.77$ henry (greatest) to $L_{36} = 0.594$ mh (least).

An obvious arrangement of the networks A and B to produce the desired delay is in a lattice or bridge circuit, as shown in Fig. 6-11. More efficient practical arrangements require the use of only one Z_A and one Z_B.

CHAPTER 7

HIGHER-ORDER LOGIC CIRCUITS

7-1. Introduction. In Chap. 4 elementary circuits for expressing logical "and," "or," and "not" have been described. This chapter will proceed to a discussion of what may be called "higher-order" logic circuits. One broad class of these consists of circuits built up of combinations of the elementary circuits in such a way that there are a number of inputs and but a single output. A simple example has already been shown in Fig. 6-9. Generally speaking, especially in the control circuits of digital computers, the situation is much more complicated, and the circuit is designed to make a binary decision based on a rather complicated combination of items of information. On the other hand, there are also circuits with several outputs as well as several inputs. These are of the nature of multiposition switches.

7-2. Switching Algebra. The first class of logic circuits consists of those that realize a binary function of several binary variables. These are logically similar to the two-terminal switching networks that were shown by Shannon to be amenable to treatment by a Boolean algebra.[1,2] These have already been mentioned and will be discussed in Sec. 7-4.

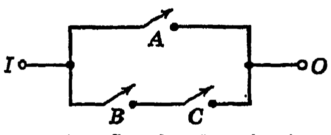


FIG. 7-1. Simple relay circuit.

As an introduction to the use of Boolean algebra in dealing with switching functions, consider the circuit of Fig. 7-1, in which A, B, and C are the contacts of three relays. The conducting path from input to output can be completed either (1) by closing A or (2) by closing both B and C. Now let the letter a be a variable that takes on the value 0 when contacts A are closed and the value 1 when they are open; it is assumed in all cases that the contacts are closed when and only when the relay is energized. The variables b and c are similarly defined. The primed letter a' is to have the value 1 when A is closed, 0 when it is open; so $a' = 1 - a$. Thus a is a binary variable: it can take on only the values

[1] C. E. Shannon, A Symbolic Analysis of Relay and Switching Circuits, *Trans. IEE*, vol. 57, p. 713, 1938.

[2] W. Keister, A. E. Ritchie, and S. H. Washburn, "The Design of Switching Circuits," D. Van Nostrand Company, Inc., Princeton, N.J., 1951.

0 and 1 as described above. Since the product ab has the value 0 if either a or b or both are 0, it follows that the product represents two relays in parallel. Since the sum $a + b$ is 0 only if both a and b are 0 and since it is 1 if a or b is 1, the sum can be used to represent two relays in series—provided we agree that $1 + 1 = 1$, which we forthwith do. Having agreed to these simple rules (see also Sec. 4-4), it becomes easy to write down an algebraic expression that assumes the values 0 or 1 according as there exists or does not exist a conducting path from input to output; in our example this path is clearly given by $a(b + c)$. An expression of this kind is called by Shannon the "hindrance" of the network.

If the hindrance of a network is written down and then transformed by such rules as $a' = 1 - a$ and $aa' = 0$, a variety of equivalent networks can be found. These will frequently be found to be simpler than the original network. A related situation is that in which it is desired to realize a given function of binary variables. This function can be transformed into a variety of forms, and an attempt can be made to find the simplest. The mathematical science useful for dealing with problems of this nature is Boolean algebra, which has already been sketchily introduced in Chap. 4.

7-3. Boolean Algebra. Boolean algebra is an abstract deductive science, in which objects of a class, in general undefined, are subjected to certain operations in accordance with certain postulates. From these are deduced the theorems of the science.

Let there be given a nonempty set of objects C, and two operations $\cap$ and $'$ obeying the postulates:[1]

P_1: if a and b are in C, then a' and $a \cap b$ are uniquely determined members of C.

P_2: if a and b are in C, then $a \cap b = b \cap a$.

P_3: if a, b, and c are in C, then $(a \cap b) \cap c = a \cap (b \cap c)$.

P_4: if a, b, and c are in C and if $a \cap b' = c \cap c'$, then $a \cap b = a$.

P_5: if a, b, and c are in C and if $a \cap b = a$, then $a \cap b' = c \cap c'$.

P_6: if a and b are in C and if $a = b$, then $a' = b'$.

P_7: if a, b, and c are in C and if $a = b$, then $a \cap c = b \cap c$ and $c \cap a = c \cap b$.

Here the equality sign is not an operation of the system, but is taken as a part of the ordinary language in which we discuss the system (the tech-

[1] The postulates and the statements of theorems T_1 to T_{22} below are quoted from "Elements of Mathematical Logic" by P. C. Rosenbloom, through permission granted by Dover Publications, Inc., New York 10, New York. This is the clearest rigorous treatment of the subject that the author has been able to find, and he thanks Dover for their kind permission to make the quotations. It must be added that the author has added some proofs of his own; so any errors that the critical reader may find should be charged to the author and not to Professor Rosenbloom.

nical logical term is the "syntax language"), and is taken to mean that the object to its left is identical with the object on its right.

Let us deduce two simple theorems:

T_1: $a \cap a = a$. For certainly $a \cap a' = a \cap a'$, whence the result follows by P_4.

T_2: $a \cap a' = c \cap c'$. For in P_5 set $b = a$.

From T_2 it is clearly appropriate to introduce a symbol for $a \cap a'$; we choose to call this "0." We also introduce a new relation $\subset$ by defining "$a \subset b$" to mean that $a \cap b = a$. We also define a new symbol "1" by setting $1 = 0'$. We now prove:

T_3: $a \subset b$ if and only if $a \cap b' = 0$. For if $a \subset b$, then by definition $a \cap b = a$. But then by P_5, for any c, we have $a \cap b' = c \cap c' = 0$. Conversely, if $a \cap b' = 0$, then for any c we have $a \cap b' = c \cap c'$, and by P_4 it follows that $a \cap b = a$, or $a \subset b$.

T_4: $a \subset a$. For $a \cap a' = 0$, and the result follows by T_3.

T_5: If $a \subset b$ and $b \subset c$, then $a \subset c$. For since $a \cap b = a$ and $b \cap c = b$, we have, by P_3, $a \cap c = (a \cap b) \cap c = a \cap (b \cap c) = a \cap b = a$.

T_6: $a \cap b \subset a$. For $a \cap b = b \cap a$ by P_2, and $(b \cap a) \cap a = b \cap (a \cap a)$ by P_3, whence, by T_1, we have $(b \cap a) \cap a = (b \cap a)$, or $(a \cap b) \cap a = a \cap b$, which proves the theorem.

T_7: If $a \subset b$ and $b \subset a$, then $a = b$. For, by hypothesis, $a \cap b = a$ and $b \cap a = b$. But by P_2 $a \cap b = b \cap a$, so $a = b$.

T_8: $a \cap 0 = 0$. For, by definition, $0 = a \cap a'$, and by T_6 setting $b = a'$, we have $a \cap a' \subset a$, which proves that $0 \subset a$. But by definition this means that $0 \cap a = a \cap 0 = 0$.

T_9: $a'' = a$. By P_2 $a'' \cap a' = a' \cap a''$, and $a'' \cap a' = 0$. Hence, by P_4 $a'' \cap a = a''$, for in P_4 replace a by a'' and b' by a'. Therefore $a'' \subset a$. Hence it follows that $a''' \subset a'$ and $a'''' \subset a''$, and by T_5 we have $a'''' \subset a$. From this, by T_3, we have $a'''' \cap a' = 0$, and by P_2 $a' \cap a'''' = 0$. Therefore by T_3, $a' \subset a'''$, which, combined with $a''' \subset a'$ proved above, shows, by T_7, that $a' = a'''$. But $a \cap a' = 0$, so also $a \cap a''' = 0$, and by T_3 $a \subset a''$. But we have shown above that $a'' \subset a$, so by T_7 we have $a'' = a$.

A new operation $\cup$ is now defined as follows: $a \cup b = (a' \cap b')'$.

T_{10}: $a \cap b = (a' \cup b')'$. For, by the definition of $\cup$ and T_9, we can write $a' \cup b' = (a \cap b)'$. Hence $(a' \cup b')' = (a \cap b)'' = a \cap b$, again by T_9.

T_{11}: $a \subset b$ if and only if $b' \subset a'$. For if $a \subset b$, then, by T_3, $a \cap b' = 0$. But by T_9 $a'' = a$, and hence by P_2 $b' \cap a'' = 0$, so by T_3 $b' \subset a'$. Conversely, if $b' \subset a'$, then $a'' \subset b''$ by the case just proved, and hence by T_9 $a \subset b$.

T_{12}: $a \subset b$ if and only if $a \cup b = b$. For, by T_{11}, $a \subset b$ if and only if $b' \subset a'$, and hence $a \subset b$ if and only if $b' \cap a' = b'$. Hence, by T_9, $(b' \cap a')' = b'' = b$. By P_2, $b = (a' \cap b')'$, and by the definition of $\cup$ we have $b = a \cup b$.

T_{13}: $a \cup b = b \cup a$, and $(a \cup b) \cup c = a \cup (b \cup c)$. The first part is a direct consequence of P_2 and the definition of $\cup$. To prove the second, we have by definition $(a \cup b) \cup c = ((a \cup b)' \cap c')' = ((a' \cap b') \cap c')'$. Hence by P_3 $(a \cup b) \cup c = (a' \cap (b' \cap c'))'$. By T_9, this becomes $(a \cup b) \cup c = (a' \cap (b' \cap c')'')'$, which, by the definition of $\cup$, gives us $(a \cup b) \cup c = a \cup (b' \cap c')' = a \cup (b \cup c)$.

T_{14}: $a \cup a = a$. For by the definition of $\cup$, we have $a \cup a = (a' \cap a')'$, whence by T_1 $a \cup a = (a')' = a''$, and the theorem follows by T_9.

T_{15}: $a \cup a' = 1$. For $a' \cap a = a \cap a' = 0$, whence $(a \cap a')' = 0' = 1$, and the theorem follows by the definition of $\cup$.

T_{16}: $a \subset a \cup b$. For $(a \cup b)' = (a' \cap b')'' = a' \cap b'$ by T_9 and the definition of $\cup$. By T_6, $a' \cap b' \subset a'$, or $(a \cup b)' \subset a'$. Hence by T_{11} the theorem follows.

T_{17}: $a \cup (a \cap b) = a \cap (a \cup b) = a$. That $a \cup (a \cap b) = a$ is an immediate consequence of T_6 and T_{12}, and that $a \cap (a \cup b) = a$ follows from T_{16} and the definition of $\subset$.

T_{18}: If $a \subset b$, then $a \cap c \subset b \cap c$ and $a \cup c \subset b \cup c$ for every c. For by P_2 and P_3 we have $(a \cap c) \cap (b \cap c) = (a \cap c) \cap (c \cap b) = a \cap (c \cap (c \cap b)) = a \cap ((c \cap c) \cap b)$. By T_1 this last expression reduces to $a \cap (c \cap b) = a \cap (b \cap c) = (a \cap b) \cap c$. This last is equal to $a \cap c$ by the definition of $a \subset b$, so $(a \cap c) \cap (b \cap c) = a \cap c$, and by the definition of $\subset$ the first part of the theorem follows. To prove the second part, we have $(a \cup c) \cup (b \cup c) = (a \cup c) \cup (c \cup b)$ by T_{13}; $a \cup (c \cup (c \cup b))$ by T_{13}; $a \cup ((c \cup c) \cup b)$ by T_{13}; $a \cup (c \cup b)$ by T_{14}; $a \cup (b \cup c)$ by T_{13}; $(a \cup b) \cup c$ by T_{13}; $b \cup c$ by T_{12}. But, by T_{12}, $(a \cup c) \cup (b \cup c) = b \cup c$ implies that $a \cup c \subset b \cup c$.

T_{19}: If $a \subset c$ and $b \subset c$, then $a \cup b \subset c$, and, if $c \subset a$ and $c \subset b$, then $c \subset a \cap b$. To prove the first part, note that, by T_{12}, $a \cup c = c$ and $b \cup c = c$. But $(a \cup b) \cup c = a \cup (b \cup c)$ by T_{13}, which at once reduces to $a \cup c$ and hence to c, so $(a \cup b) \cup c = c$, and so $a \cup b \subset c$ by T_{12}. The second part is established by a similar argument: noting that, by hypothesis, $c \cap a = c$ and $c \cap b = c$; applying the hypotheses and P_3, $c \cap (a \cap b) = (c \cap a) \cap b = c \cap b = c$; and this implies that $c \subset a \cap b$, by the definition of $\subset$.

T_{20}: $a \cap (a' \cup b) = a \cap b$. By the definition of 0 we have $(a \cap b') \cap (a \cap b')' = 0$. By P_2 and P_3 this becomes $b' \cap (a \cap (a \cap b')') = 0$, whence, by T_3, $a \cap (a \cap b')' \subset b$, or, by the definition of $\subset$, $(a \cap (a \cap b')') \cap b = a \cap (a \cap b')'$. But by the definition of $\cup$ and by T_9, $a \cap (a' \cup b) = a \cap (a'' \cap b')' = a \cap (a \cap b')'$. Hence $(a \cap (a' \cup b)) \cap b = a \cap (a' \cup b)$. By P_3 this becomes $a \cap ((a' \cup b) \cap b) = a$

$\cap (a' \cup b)$, and by P_2, $a \cap (b \cap (b \cup a')) = a \cap (a' \cup b)$. Now, applying T_{17}, this becomes $a \cap b = a \cap (a' \cup b)$, which was to be proved.

T_{21}: $a \cap (b \cup c) = (a \cap b) \cup (a \cap c)$. Since $b \subset (b \cup c)$, $c \subset (b \cup c)$ by T_{13} and T_{16}, we have, by T_{18}, $a \cap b \subset a \cap (b \cup c)$, $a \cap c \subset a \cap (b \cup c)$. Hence, by T_{19}, $(a \cap b) \cup (a \cap c) \subset a \cap (b \cup c)$. Now consider the product $(a \cap (b \cup c)) \cap ((a \cap b) \cup (a \cap c))'$. By T_{10} and T_9, this can be written $(a \cap (b \cup c)) \cap ((a' \cup b') \cap (a' \cup c'))$. By P_3, we can write this in the form $(b \cup c) \cap ((a \cap (a' \cup b')) \cap (a' \cup c'))$. By T_{20}, $a \cap (a' \cup b') = a \cap b'$, so the product becomes $(b \cup c) \cap ((a \cap b') \cap (a' \cup c'))$. Again by P_2 and P_3, this becomes $(b \cup c) \cap (b' \cap (a \cap (a' \cup c')))$, and, again by T_{20}, this reduces to $(b \cup c) \cap (b' \cap (a \cap c'))$. By application of P_2 and P_3 we have $(b' \cap (b \cup c)) \cap (a \cap c')$, and, again by T_{20}, this reduces to $(b' \cap c) \cap (a \cap c')$, which, by P_2 and P_3, can be written as $b' \cap ((c \cap c') \cap a)$. By the definition of 0 this is $b' \cap (0 \cap a)$, and two applications of T_8 show that this all reduces to 0. Hence, by T_3, $a \cap (b \cup c) \subset ((a \cap b) \cup (a \cap c))$, and this, together with the previously demonstrated relation $(a \cap b) \cup (a \cap c) \subset a \cap (b \cup c)$, shows, by T_7, that the theorem is true.

T_{22}: $a \cup 0 = a$, $a \cup 1 = 1$, $a \cap 1 = a$. By T_8, $0 \subset a$, whence by T_{12} $0 \cup a = a$. Also $0 \subset a'$, and by T_{11}, $a'' \subset 0'$, or $a \subset 1$, whence by T_{12} $a \cup 1 = 1$. Finally, since we have seen that $a \subset 1$, we appeal to the definition of $\subset$ to show that $a \cap 1 = a$.

We are now in a position to prove a remarkable theorem which is basic to the application of the algebra of logic to the study of switching circuits. It deals with the expansion of a Boolean function. We attempt to explain this notion before proceeding to the proof of T_{23}.

First, let us recall the idea of a function of a complex variable. Essentially it is that a function is a rule that allows us, given one complex number, to determine by computation a second; we denote such a function of a complex variable by the symbol $f(z)$. Similarly, if we have a rule by which, given any pair of complex numbers (z,w), we can calculate a third value, we then have a function of two complex variables $f(z,w)$, and so on. In every case the simplest function is a constant: $f(z) = c$.

In our algebra of logic we shall use the same symbols $f(x)$, $f(x,y)$, etc., where now $x, y, \ldots$ can take as their values any of the objects of the class C with which our algebra concerns itself. The value of the function is also to be a member of C. We shall begin by admitting any object of C as a possible value of a constant function: $f(x) = a$, $f(x,y) = a$, etc. The next step is to admit identity: $f(x) = x$. We have available three operations $\cap$, $\cup$, $'$, and with these we can form a variety of expressions $x \cup y$, $x \cap y$, $x \cup y'$, $(x' \cap y')'$, etc. In general, by a Boolean function we agree to understand any expression that can be built up by starting with constants and the identity function and then applying our three operations a finite number of times.

The fundamental expansion theorem will first be stated for one variable and for two variables:

(1) $f(x) = (f(1) \cap x) \cup (f(0) \cap x')$

(2) $f(x,y) = (f(1,1) \cap (x \cap y)) \cup (f(1,0) \cap (x \cap y')) \cup (f(0,1) \cap (x' \cap y)) \cup (f(0,0) \cap (x' \cap y'))$

Now let $\sum_{0}^{n} x_i$ stand for $x_0 \cup x_1 \cup \cdots \cup x_n$ and let x_j^i signify x_j if $i = 0$ but x_j' if $i = 1$. Then the two forms of the expansion theorem can be written

(1) $$f(x) = \sum_{i=0}^{1} (f(1^i) \cap x^i)$$

(2) $$f(x,y) = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} (f(1^{i_1}, 1^{i_2}) \cap (x^{i_1} \cap y^{i_2}))$$

The general statement is T_{23}:

$$f(x_1,x_2, \ldots ,x_k) = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_k=0}^{1} (f(1^{i_1},1^{i_2}, \ldots ,1^{i_k}) \cap (x_1^{i_1} \cap x_2^{i_2} \cap \cdots \cap x_k^{i_k})) \quad (7\text{-}1)$$

Note that, since, by P_3, $a \cap (b \cap c) = (a \cap b) \cap c$, either expression can be written without parentheses as $a \cap b \cap c$; this justifies the above notation.

The scheme of proof here is to demonstrate the theorem for $k = 1$ and then to proceed by induction. So let $k = 1$ and suppose that $f(x) = c$, a constant. Then, by T_{21}, $(c \cap x) \cup (c \cap x') = c \cap (x \cup x') = c \cap 1 = c$; so the alleged expanded form does in fact always give $f(x) = c$. Thus the theorem is true for $f(x) = c$. Now consider the case $f(x) = x$, so that $f(0) = 0, f(1) = 1$. The theorem then states that $f(x) = (1 \cap x) \cup (0 \cap x') = x \cup 0 = x$, which is obviously correct. Now let $g(x) = f(x)'$, and suppose that the theorem is true for $f(x)$. The expression for $g(x)$ should then be given by $((f(1) \cap x) \cup (f(0) \cap x'))'$, which can be transformed by T_9 and T_{10} into $(f(1)' \cup x') \cap (f(0)' \cup x)$. From this, by application of T_{21} (using P_2 and P_3 twice), there are obtained first $((f(1)' \cup x') \cap f(0)') \cup ((f(1)' \cup x') \cap x)$ and then $((f(0)' \cap f(1)') \cup (f(0)' \cap x')) \cup ((f(1)' \cap x) \cup (x' \cap x))$, which, by T_{22} and the fact that $x' \cap x = 0$, reduces to $((f(0)' \cap f(1)') \cup (f(0)' \cap x')) \cup (f(1)' \cap x)$, and this last expression can, by P_2 and T_{13}, be written in the form

$(f(1)' \cap f(0)') \cup ((f(1)' \cap x) \cup (f(0)' \cap x'))$. Since $x \cup x' = 1$ and since, for any a, $a \cap 1 = a$, this can be written in the form $((f(1)' \cap f(0)') \cap (x \cup x')) \cup ((f(1)' \cap x) \cup (f(0)' \cap x'))$, which, by T_{21}, becomes $(((f(1)' \cap f(0)') \cap x) \cup ((f(1)' \cap f(0)') \cap x')) \cup ((f(1)' \cap x) \cup (f(0)' \cap x'))$. The first and third and the second and fourth of these expressions (joined by "$\cup$") can be brought together and rearranged, by T_{13}, P_2, and P_3, to give $((f(1)' \cap x) \cup ((f(1)' \cap x) \cap f(0)')) \cup ((f(0)' \cap x') \cup ((f(0)' \cap x') \cap f(1)'))$. If T_{17} is then applied twice, the result is $(f(1)' \cap x) \cup (f(0)' \cap x')$, which proves that, if $f(x)$ can be expanded, so can $f(x)'$.

The next step is to show that, if any two functions $f(x)$ and $g(x)$ possess the expansion, then so does $h(x) = f(x) \cup g(x)$. To begin with, it is obviously true that $f(x) \cup g(x) = ((f(1) \cap x) \cup (f(0) \cap x')) \cup ((g(1) \cap x) \cup (g(0) \cap x'))$. By an application of T_{13}, this becomes $((f(1) \cap x) \cup (g(1) \cap x)) \cup ((f(0) \cap x') \cup (g(0) \cap x'))$, which reduces to $((f(1) \cup g(1)) \cap x) \cup ((f(0) \cup g(0)) \cap x')$ by P_2, T_{13}, and T_{21}. This proves that $h(x) = f(x) \cup g(x)$ also possesses the expansion. In a similar fashion it can easily be established that $f(x) \cap g(x)$ possesses the expansion.

The above arguments show that any $f(x)$ that is built up from constants and the variable x by a finite number of applications of $\cup$, $\cap$, and $'$ must possess the alleged expansion, and so the theorem has been proved to hold for all such functions, usually referred to as "Boolean functions."

The proof that the theorem holds for functions of any number of variables now proceeds by induction. Suppose that Eq. (7-1) holds for functions of $k - 1$ variables. Then it is surely true that

$$f(x_1,x_2, \ldots ,x_k) = \sum_{i_2=0}^{1} \sum_{i_3=0}^{1} \cdots \sum_{i_k=0}^{1} (f(x_1,1^{i_2}, \ldots ,1^{i_k}) \cap (x_2^{i_2} \cap x_3^{i_3} \cap \cdots \cap x_k^{i_k})) \qquad (7\text{-}2)$$

but, since the theorem is true for $k = 1$, it is certainly true that

$$f(x_1,1^{i_2},1^{i_3}, \ldots ,1^{i_k}) = (f(1^0,1^{i_2},1^{i_3}, \ldots ,1^{i_k}) \cap x_1^0) \cup (f(1^1,1^{i_2},1^{i_3}, \ldots ,1^{i_k}) \cap x_1^1) \qquad (7\text{-}3)$$

and, if substitution is now made from (7-3) into (7-2), the truth of the theorem for k variables follows immediately. Hence, since it has been established that the theorem is true for one variable, it follows that it is true for any finite number of variables.

There exists a second expansion in which the roles of $\cup$ and $\cap$ are interchanged. It will be stated here, and proved only for the case of a single variable. As a preliminary, the following theorem will be stated and proved:

T_{24}: $a \cup (b \cap c) = (a \cup b) \cap (a \cup c)$. For, by T_{21}, it is true that $a' \cap (b' \cup c') = (a' \cap b') \cup (a' \cap c')$ and, by application of P_6, the definition of $\cup$, and T_{10}, the truth of T_{24} follows.

The expansion theorem T_{25} will now be stated:

$$f(x_1,x_2, \ldots ,x_k) = \prod_{i_1=0}^{1} \prod_{i_2=0}^{1} \cdots \prod_{i_k=0}^{1} (f(1^{i_1},1^{i_2}, \ldots ,1^{i_k}) \cup (x_1^{1-i_1} \cup x_2^{1-i_2} \cup \cdots \cup x_k^{1-i_k})) \qquad (7\text{-}4)$$

The product symbol Π is taken to mean that the intersection or logical product of the terms following it is to be taken. A proof will be given only for $k = 1$, in which case the theorem becomes $f(x) = (f(1) \cup x') \cap (f(0) \cup x)$. Start with the expansion of T_{23}, $f(x) = (f(1) \cap x) \cup (f(0) \cap x')$, and apply P_6, T_9, T_{10}, and the definition of $\cup$, to obtain $f(x)' = (f(1)' \cup x') \cap (f(0)' \cup x)$. Upon application of T_{21}, this can be written $f(x)' = ((f(1)' \cup x') \cap f(0)') \cup ((f(1)' \cup x') \cap x)$. By P_2 and T_{21}, this can be written $f(x)' = ((f(1)' \cup x') \cap f(0)') \cup ((x \cap f(1)') \cup (x \cap x'))$ and, by the definition of 0 and by T_{22}, this easily reduces to $f(x)' = ((f(1)' \cup x') \cap f(0)') \cup (f(1)' \cap x)$, and by P_2 and T_{21} to $f(x)' = (f(0)' \cap f(1)') \cup (f(0)' \cap x') \cup (f(1)' \cap x)$. Now apply T_{15} and T_{22} to replace $f(0)' \cap f(1)'$ by $(f(0)' \cap f(1)') \cap (x \cup x')$, which, by P_3 and T_{21} can be written in the form $(f(0)' \cap (f(1)' \cap x)) \cup (f(1)' \cap (f(0)' \cap x'))$. Thus $f(x)' = (f(0)' \cap (f(1)' \cap x)) \cup (f(1)' \cap (f(0)' \cap x')) \cup (f(0)' \cap x') \cup (f(1)' \cap x)$, which reduces, by T_{17}, to $f(x)' = (f(1)' \cap x) \cup (f(0)' \cap x')$. We now apply P_6, T_9, T_{10}, and the definition of $\cup$ to obtain $f(x) = (f(1) \cup x') \cap (f(0) \cup x)$.

Several interpretations of the abstract algebra developed above can be made. First, the elements a, b, . . . can be taken to be the subsets of a "universal set," which is represented by 1 (sometimes written "I"), and the "empty set," which contains no elements and is represented by 0. The operations are interpreted as follows: (1) $a \cup b$ is the set of objects that are members of set a, set b, or both; it is frequently written $a + b$, and the operation is referred to as "logical sum" or merely "or." (2) $a \cap b$ is the set of objects that are simultaneously members of both set a and set b; it is frequently written $a \times b$ or simply ab and is referred to as "logical product" or "and." (3) a' is the set of objects that are included in the universal class but not in a; it is sometimes written $\bar{a}$ or Ca and is referred to as "not a" or "the complement of a." (4) The derived relation $a \subset b$ is understood to mean that set a is a subset of set b. The drawing of a few simple diagrams suffices to convince one that the postulates P_1 to P_7 do indeed hold with this interpretation and, therefore, that all the theorems must automatically be true when so interpreted.

As a second interpretation, consider the calculus of propositions.[1] The elements, usually written p, q, r, . . . , are interpreted as propositions, at least as propositions that can be asserted to be true or false, so that each proposition has a "truth value" t or 1 if true, f or 0 if false. The operations are: (1) $\vee$ is written for $\cup$; $p \vee q$ signifies the assertion of p or q, and the operation is called "alternation" or "disjunction." (2) $\wedge$ is written for $\cap$; $p \wedge q$ signifies the joint assertion of p and q and is called "conjunction." (3) For $'$ we substitute $\sim$ or $^-$; $\sim p$ or $\bar{p}$ signifies the denial of p, and the operation is called "negation." (4) Whereas in our abstract algebra "$\subset$" stands essentially for a statement in terms of the equivalence sign "$=$," which is itself exterior to the algebra, in the calculus of propositions the symbol "$\supset$" is introduced by the definition that $p \supset q$ is equivalent to the proposition $\bar{p} \vee q$ ("p is false or q is true"); $p \supset q$ is read as "p implies q." A short consideration of the Boolean postulates suffices to convince one that they all hold under the propositional interpretation, and so all the theorems can also be so interpreted; of course theorems whose statement involves "$\subset$" will not be used, and new theorems will appear involving "$\supset$." Note that it is always possible to check a theorem of the calculus of propositions by showing that the propositions asserted to be equivalent do indeed always have the same "truth value." As an example of the computation of truth values, consider the possible values of $\bar{p} \vee q$: p can be 0 or 1 (f or t) and so can q. We apply the rules $0 \vee 0 = 0$, $0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1$. The results can be displayed in tabular form:

p	q	$\bar{p} \vee q$
1	0	0
1	1	1
0	0	1
0	1	1

The table reveals that $p \supset q$ is true if p and q are both false or both true and also when p is false and q is true. Thus $p \supset q$ is not quite what we mean when we say "p implies q" in ordinary language; for this reason the operation is frequently designated "material implication" as distinguished from "strict implication," a notion introduced by C. I. Lewis.

A third interpretation is Shannon's switching algebra. The elements are variables that refer to the state of the contacts of relays; they assume but two values, 0 (closed) and 1 (open). We have the operations (1) $a + b$, which signifies the state of two relays in series and is formally equivalent to $a \vee b$, since $0 + 0 = 0$, $0 + 1 = 1 + 0 = 1$, and $1 + 1 = 1$; (2) ab, which signifies the state of two relays in parallel and is formally equivalent

[1] *Ibid.*, chap. 2.

to $a \wedge b$; (3) a', which signifies the opposite of a, i.e., if $a = 1$, $a' = 0$; if $a = 0$, $a' = 1$. With these interpretations the postulates P_1 to P_7 are satisfied, and so all the theorems of Boolean algebra are available.

As a fourth example, recall the remarks made in Sec. 4-12 on the Harvard switching algebra. This is also a two-valued algebra; its addition and multiplication rules are, however, those of ordinary arithmetic. Precautions are taken that no sum ever exceed 1, and the resulting algebra is equivalent to a system founded strictly on Boolean algebra; for example, the expansion theorem T_{25} is valid.

A fifth example is Lewis's switching algebra,[1] in which the variables refer to the states of pairs of relay contacts: $x = 0$ and $x = 1$ signify open and closed contacts, respectively. Ordinary arithmetical addition and multiplication are used. Thus, if relays x and y are in series, the state of the combination is described by xy. The negation operator is introduced by the definition $N(x) = 1 - x$. It is then possible to describe the combined state of two relays whose pairs of contacts are in parallel by the expression $N(Nx \cdot Ny)$, which turns out to have the value $x + y - xy$, so that, as in the Harvard switching algebra, numerical values are restricted to 0 and 1. The algebra possesses all the customary features, such as an expansion theorem. A variety of other algebras, some strictly arithmetical and some possessing features of Boolean algebra, have been published.[2-4]

As a sixth and final example, there is the system used by D. R. Hartree.[5] This is a two-valued system, 1 standing for the presence of a signal, 0 for its absence. Thus 1 can stand for an electrical pulse and 0 for its absence, or 1 and 0 can stand for the two possible voltage levels at some specified point (e.g., the voltage at the plate of one tube of a toggle), as in the Harvard system. The system is strictly Boolean. The negation and alternation ("or") symbols are those of the calculus of propositions. For conjunction ("and") Hartree uses "&," reserving the symbol "$\wedge$" to signify "exclusive or," defined as follows: $p \wedge q = (p \mathbin{\&} (\sim q)) \vee ((\sim p) \mathbin{\&} q)$. Thus $p \wedge q$ has the value 1 if one and only one of p and q has the value 1. It can readily be shown that $q \wedge p = p \wedge q$ and that $p \wedge (q \wedge r) = (p \wedge q) \wedge r$, this combination having the value

[1] I. A. D. Lewis, A Symbolic Method for the Solution of Some Switching and Relay-circuit Problems, *Proc. IEE*, vol. 98, pt. 2, no. 111, pp. 181–191, May, 1951.

[2] O. Plechl and A. Duschek, Grundzüge eines Algebras der elektrischen Schaltungen, *Österr. Ing.-Arch.*, vol. 1, p. 203, 1946.

[3] G. A. Montgomerie, Sketch for an Algebra of Relay and Contracter Circuits, *Proc. IEE*, vol. 95, pt. 2, p. 355, 1948.

[4] G. H. Buffery, A Contribution to the Algebra of Relay and Switch Contacts, *Proc. IEE*, vol. 97, pt. 2, p. 351, 1950.

[5] D. R. Hartree, "Calculating Instruments and Machines," chap. 8, secs. 8.4 and 8.5, The University of Illinois Press, Urbana, Ill., 1949.

1 if one or three of p, q, r have the value 1, but the value 0 if just two of the three have the value 1.

Enough has been said to alert the reader to the fact—perhaps regrettable—that a number of different ways of writing the theorems of the Boolean algebra exist and that he must be prepared to meet these in the literature.

7-4. On the Simplification of Switching Functions. If the Boolean algebra afforded merely a convenient symbolic means of writing down complicated switching functions, it would hardly be worthwhile to devote so much space to it. But, beyond this, it affords a systematic method for simplifying complicated combinations of logical elements (mixers, gates, inhibitors) by application of the theorems proved above. It was for purposes of systematic simplification that Shannon originally applied methods of this kind. Consider a simple example given by Shannon.[1]

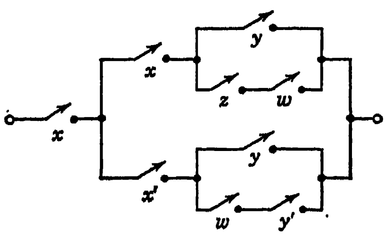


FIG. 7-2. Relay circuit to be simplified.

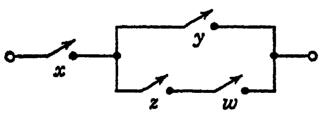


FIG. 7-3. Simplified form of Fig. 7-2.

The hindrance of this circuit (see Sec. 7-2) is given by

$$H(x,y,z,w) = x + [(x + (y \cdot (z + w))) \cdot (x' + (y \cdot (w + y')))] \quad (7\text{-}5)$$

By T_{24}, the right member (bracketed) of Eq. (7-5) can be written in the form

$$(x + (x + (y \cdot (z + w)))) \cdot (x + (x' + (y \cdot (w + y'))))$$

By T_{13}, T_{14}, and T_{15}, this becomes

$$(x + (y \cdot (z + w))) \cdot (1 + (y \cdot (w + y')))$$

and, by T_{22} applied twice, this reduces to $H = x + (y \cdot (z + w))$, which is the hindrance of the much simpler circuit shown in Fig. 7-3. Admittedly, the reduction is obvious upon inspection of Fig. 7-2; however, it is useful as an illustration.

The whole object of such a procedure is to obtain a switching function of the simplest possible form. This term obviously needs to be defined, since it is not at all clear in what sense "simplest" should be taken. Ordinarily, the form of a given switching function considered to be simplest is the form that involves the least number of occurrences of the variables. It should be noted, however, that the function that is simplest

[1] C. E. Shannon, The Synthesis of Two-terminal Switching Networks, *Bell System Tech. J.*, vol. 28, no. 1, pp. 59–98, January, 1949.

in this sense is not necessarily the most desirable from the point of view of practical engineering, since it may well lead to a circuit that is very difficult to design satisfactorily. The results of logical simplification must always be examined critically from the circuit designer's point of view.

By generalization from the example discussed above, the problem of simplifying a switching function can be defined as the problem of finding a function of simpler form (e.g., in the sense of involving fewer occurrences of the variables) that assumes the same values as the given function. In some cases the new function may also be a function of fewer variables, which happens if the functional value is not changed when some variable is replaced by its negative; for example, $f(p,q) = (p \wedge q) \vee (p \wedge q')$ reduces simply to p.

It is possible to proceed to the simplification of a given function by deriving all functions that assume the same values (or have the same truth table) and to select from these the one of simplest form; this is simplification by exhaustive comparison.[1] The only drawback to this is that, presumably, a great deal of labor is required; reflect that 2^{2^n} switching functions of n variables exist (this is readily seen since in the disjunctive normal form there are 2^n terms and the coefficient of each can have one of two possible values). However, there is no reason why the problem of generating the functions equivalent to a given function cannot be handled successfully by means of a digital computer.

A second type of approach is to operate on the given function by means of logical theorems. This procedure goes back to Shannon's original work, and a considerable amount of effort has been expended upon it. Perhaps the most famous result of this has been the creation of the Harvard minimizing charts.[2] No derivation is given in the publication in which their use is expounded, but Burkhart[3] has stated the logical basis for their construction and has shown that they do not necessarily result in the simplest equivalent function. The charts do, however, ordinarily give useful results and have the advantage of being simple to use. An example of a reduction obtained by this method will be given. Another procedure has been presented by W. V. Quine, and this also will be given below.

Reductions can be effected also by other methods worked out at the Harvard Computation Laboratory. The central idea in these is the "minimum generating set" of all the switching functions of n variables. Such a set of functions has the property that, if the functions belonging to it are transformed in all possible ways by permuting and negating the

[1] W. H. Burkhart, Theorem Minimization, *J. ACM*, May, 1952, pp. 259–263.
[2] Staff, Harvard Computation Laboratory, "Synthesis of Electronic Computing and Control Circuits," chap. 5, Harvard University Press, Cambridge, Mass., 1951.
[3] *Op. cit.*

variables, all the possible 2^{2^n} functions of n variables are obtained. An essentially arithmetical approach to this problem, depending on the properties of a suitably defined characteristic function corresponding to a given switching function, has been worked out.[1] Another approach depends upon certain combinatorial theorems connected with the theory of groups, originally due to George Pólya.[2,3]

To apply the Harvard minimizing charts, it is necessary first to write the given function in the disjunctive normal form of Theorem 23 [Eq. (7-1)]. The result of the operations is a simplified equivalent (in the sense of fewer occurrences of the variables) expressed, however, in the conjunctive form of Theorem 25 [Eq. (7-4)]. Before the method is described it is necessary to introduce a simplified notation for the coefficients. Each coefficient denotes the value of the given function for a set of values of the variables, in which each variable assumes one of the values 0 and 1. These are written down in the order in which they occur, and the result is interpreted as an n-bit binary integer; this in turn is converted to a decimal integer and then the functional value in question is designated by the letter f with the equivalent decimal integer as subscript. For example, $f(0,0,0)$ is written f_0, $f(0,0,1)$ is written f_1, and so on, up to $f(1,1,1)$, which is written f_7. For the three-variable case the chart is written as follows:[4]

f_0	p'	q'	r'	$p' \wedge q'$	$p' \wedge r'$	$q' \wedge r'$	$p' \wedge q' \wedge r'$
f_1	p'	q'	r	$p' \wedge q'$	$p' \wedge r$	$q' \wedge r$	$p' \wedge q' \wedge r$
f_2	p'	q	r'	$p' \wedge q$	$p' \wedge r'$	$q \wedge r'$	$p' \wedge q \wedge r'$
f_3	p'	q	r	$p' \wedge q$	$p' \wedge r$	$q \wedge r$	$p' \wedge q \wedge r$
f_4	p	q'	r'	$p \wedge q'$	$p \wedge r'$	$q' \wedge r'$	$p \wedge q' \wedge r'$
f_5	p	q'	r	$p \wedge q'$	$p \wedge r$	$q' \wedge r$	$p \wedge q' \wedge r$
f_6	p	q	r'	$p \wedge q$	$p \wedge r'$	$q \wedge r'$	$p \wedge q \wedge r'$
f_7	p	q	r	$p \wedge q$	$p \wedge r$	$q \wedge r$	$p \wedge q \wedge r$

The first step is to note, in the disjunctive normal form of f, the f_i that are equal to 1, and to draw a horizontal line across the chart through each such f_i. Next, note each variable or combination of variables through which each line passes, and cross off these combinations wherever else they occur in the same vertical column. Now inspect each row for which $f_i = 0$. Some rows will be found in which there is only one unmarked combination of a minimum number of variables. These combinations

[1] W. L. Semon, Characteristic Numbers and Their Use in the Decomposition of Switching Functions, *J. ACM*, May, 1952, pp. 273–280.

[2] T. Singer, The Theory of Counting Techniques, *J. ACM*, May, 1952, pp. 287–291.

[3] R. L. Ashenhurst, The Application of Counting Techniques, *J. ACM*, May, 1952, pp. 293–305.

[4] This description of the construction and use of the minimizing chart is taken directly from "Synthesis of Electronic Computing and Control Circuits," pp. 56–57.

are called "essential" combinations. Each such combination and all repetitions of it in the same column are now to have circles drawn about them. The rows that contain no encircled unmarked combinations are then inspected; each will be found to contain at least two combinations of a minimum number of variables, which are called "optional" combinations. Encircle at least one "optional" combination in each row, and encircle all occurrences of such a combination in its column. This should be done in such a way as to minimize the number of distinct optional combinations encircled. The product of the negatives of the essential and the optional combinations is the desired minimal form of the original function.

As an example, consider the function

$$(q \wedge p') \vee (q \wedge r') \vee (q' \wedge p) \vee (q' \wedge r)$$

Substitution of the values shows that $f_0 = f(0,0,0) = 0$, $f_1 = f(0,0,1) = 1$, $f_2 = f(0,1,0) = 1$, $f_3 = f(0,1,1) = 1$, $f_4 = f(1,0,0) = 1$, $f_5 = f(1,0,1) = 1$, $f_6 = f(1,1,0) = 1$, $f_7 = f(1,1,1) = 0$. The table becomes

f_0	~~p'~~	~~q'~~	~~r'~~	~~$p' \wedge q'$~~	~~$p' \wedge r'$~~	~~$q' \wedge r'$~~	$p' \wedge q' \wedge r'$
~~f_1~~	~~p'~~	~~q'~~	~~r~~	~~$p' \wedge q'$~~	~~$p' \wedge r$~~	~~$q' \wedge r$~~	~~$p' \wedge q' \wedge r$~~
~~f_2~~	~~p'~~	~~q~~	~~r'~~	~~$p' \wedge q$~~	~~$p' \wedge r'$~~	~~$q \wedge r'$~~	~~$p' \wedge q \wedge r'$~~
~~f_3~~	~~p'~~	~~q~~	~~r~~	~~$p' \wedge q$~~	~~$p' \wedge r$~~	~~$q \wedge r$~~	~~$p' \wedge q \wedge r$~~
~~f_4~~	~~p~~	~~q'~~	~~r'~~	~~$p \wedge q'$~~	~~$p \wedge r'$~~	~~$q' \wedge r'$~~	~~$p \wedge q' \wedge r'$~~
~~f_5~~	~~p~~	~~q'~~	~~r~~	~~$p \wedge q'$~~	~~$p \wedge r$~~	~~$q' \wedge r$~~	~~$p \wedge q' \wedge r$~~
~~f_6~~	~~p~~	~~q~~	~~r'~~	~~$p \wedge q$~~	~~$p \wedge r'$~~	~~$q \wedge r'$~~	~~$p \wedge q \wedge r'$~~
f_7	~~p~~	~~q~~	~~r~~	~~$p \wedge q$~~	~~$p \wedge r$~~	~~$q \wedge r$~~	$p \wedge q \wedge r$

so the function reduces to

$$(p' \wedge q' \wedge r')' \wedge (p \wedge q \wedge r)'$$

which involves six occurrences of the variables instead of eight. A moderate amount of manipulation puts this into the form

$$(p \wedge q') \vee (q \wedge r') \vee (p' \wedge r)$$

which is the same kind of expression as the one from which we started but which is a disjunction of only three terms instead of four.

There is a great deal more to the Harvard algebra than appears above; for example, a great deal has been done toward expressing functions in terms of the vacuum-tube operators which were defined in Chap. 4. Nothing will be said here about this; the reader is referred to "Synthesis of Electronic Computing and Control Circuits." Note also that an equivalent chart method has been published by Veitch.[1]

[1] E. W. Veitch, A Chart Method for Simplifying Truth Functions, *J. ACM*, May, 1952, pp. 127–133.

Another simplification process is originally due to Quine[1] and has been developed by Samson and Mills[2] and by Quine.[3] In order to present it, it is necessary first to define some terms:

1. A *fundamental formula* is a conjunction of letters p, q, r, etc., in which no letter appears twice; fundamental formulas will be designated by Greek lower-case letters ϕ, ψ, etc.
2. A *normal formula* is an alternation or disjunction of fundamental formulas, which are said to be its "clauses." Normal formulas will be designated by Greek capitals Φ, Ψ, etc.
3. A fundamental formula is said to *subsume* another fundamental formula if and only if each letter contained in ψ is also contained in ϕ; for example, if $\phi = p \wedge q \wedge r$, $\psi = p \wedge q$, then ϕ subsumes ψ.
4. The *consensus* of fundamental formulas $p \wedge \phi$ and $p' \wedge \psi$ is found by forming $\phi \wedge \psi$ and deleting any duplicates, provided that in the result no proposition is both affirmed and denied.
5. ϕ is a *prime implicant* of Φ if and only if (*a*) $\phi \supset \Phi$ and (*b*) ϕ does not subsume any shorter fundamental formula ψ such that $\psi \supset \Phi$.

The objective of the simplification process is to find the shortest normal formula equivalent to a given Φ. Quine has proved[4] that any shortest alternational normal equivalent of Φ is an alternation of prime implicants of Φ. Hence the process is essentially a systematic method of finding the prime implicants or, rather, of reducing Φ to an alternation of prime implicants; after this has been done, it is still necessary to find the shortest such alternation, as will be explained below.

The reduction of Φ to an alternation of all its prime implicants depends upon the repeated application of two rules:

1. If any clause ϕ of Φ subsumes another clause ψ, then drop ϕ. Also replace $p \vee (p' \wedge \phi)$ by $p \vee \phi$ and $p' \vee (p \wedge \phi)$ by $p' \vee \phi$, which can be done only if p itself (or p') is a clause.
2. Adjoin to Φ as an additional clause the consensus of two clauses, but do this only if the consensus does not subsume any clause already present.

These two rules are applied as often as possible, rule 1 being applied as often as possible before and after each application of rule 2. Quine

[1] W. V. Quine, The Problem of Simplifying Truth Functions, *Am. Math. Monthly*, vol. 59, no. 8, pp. 521–531, October, 1952 (Quine I).

[2] E. W. Samson and B. E. Mills, Circuit Minimization: Algebra and Algorithms for New Boolean Canonical Expressions, *Air Force Cambridge Research Center Tech. Rept.* 54-21.

[3] W. V. Quine, A Way to Simplify Truth Functions, *Am. Math. Monthly*, vol. 62, no. 9, pp. 627–631, November, 1955 (Quine II).

[4] Quine I.

has proved[1] that, when no further applications are possible, the remaining formula must be an alternation of all and only prime implicants.

To illustrate the application of these rules, consider the example given by Quine:

$$\Phi = (p \wedge s) \vee (p' \wedge s') \vee (q' \wedge t) \vee (p \wedge r \wedge s) \vee (q \wedge r \wedge s') \vee (p \wedge q \wedge r \wedge t)$$

Here $p \wedge r \wedge s$ subsumes $p \wedge s$ and is therefore dropped; this is the only possible application of rule 1. Then look for the possibility of finding consensuses; this can be done for $p \wedge s$ and $q \wedge r \wedge s'$, yielding $p \wedge q \wedge r$, which is adjoined since it subsumes no clause. The result is

$$(p \wedge s) \vee (p' \wedge s') \vee (q' \wedge t) \vee (q \wedge r \wedge s') \vee (p \wedge q \wedge r \wedge t) \vee (p \wedge q \wedge r)$$

Now apply rule 1 again to drop $p \wedge q \wedge r \wedge t$, which subsumes $(p \wedge q \wedge r)$. The consensuses $r \wedge s' \wedge t$ of $q' \wedge t$ and $q \wedge r \wedge s'$, and $p \wedge r \wedge t$ of $q' \wedge t$ and $p \wedge q \wedge r$ are adjoined, giving

$$(p \wedge s) \vee (p' \wedge s') \vee (q' \wedge t) \vee (q \wedge r \wedge s') \vee (p \wedge q \wedge r) \vee (r \wedge s' \wedge t) \vee (p \wedge r \wedge t)$$

in which no clause subsumes any other, and no further consensuses exist; so these clauses must be the prime implicants of Φ.

Now, to find the shortest equivalent, it is necessary to eliminate as many clauses as possible. The test of dispensability is very simple: find whether any clauses imply the remainder of the formula, and drop any that do. It must, however, be observed that, if all potentially dispensable clauses are found, it cannot be concluded that all are in fact dispensable, for the dropping of one may make another indispensable. The following procedure has been found convenient. First of all, the clauses that were added by rule 2 after the last application of rule 1 are obvious candidates for dispensability, for they added nothing new to the truth table, being consensuses; they are bracketed. Now the alternation of unbracketed clauses is considered, and the dispensability test applied individually to each. In the example, it turns out that $p \wedge q \wedge r$ and $q \wedge r \wedge s'$ each imply the alternation of the four clauses that remain when it has been deleted; these candidates too are bracketed, giving

$$(p \wedge s) \vee (p' \wedge s') \vee (q' \wedge t) \vee [(q \wedge r \wedge s')] \vee [(p \wedge q \wedge r)] \vee [(r \wedge s' \wedge t)] \vee [(p \wedge r \wedge t)]$$

The unbracketed remainder

$$(p \wedge s) \vee (p' \wedge s') \vee (q' \wedge t)$$

is called the "core" of Φ, and must either be itself or be contained in the

[1] Quine II.

shortest normal equivalent. Next, test each bracketed clause to see whether it implies the core; if it does, drop it entirely. In the example, this is not the case for any of the bracketed clauses, and a further testing program must be carried out. In doing this, it is sensible to start by testing as large as possible an alternation of bracketed clauses to see whether this implies the rest, because it is desired to drop as many clauses as possible. In the example, this means starting with alternations of three of the bracketed clauses, for it is known that the core is not implied by any of the four individually and, therefore, is not implied by their alternation. It turns out, as the reader can verify, that

$$(q \wedge r \wedge s') \vee (r \wedge s' \wedge t) \vee (p \wedge r \wedge t)$$

and $$(p \wedge q \wedge r) \vee (r \wedge s' \wedge t) \vee (p \wedge r \wedge t)$$

imply $$(p \wedge s) \vee (p' \wedge s') \vee (q' \wedge t) \vee (p \wedge q \wedge r)$$

and $$(p \wedge s) \vee (p' \wedge s') \vee (q' \wedge t) \vee (q \wedge r \wedge s')$$

respectively, and so the last two expressions are alternative simplest normal equivalents of Φ.

It becomes clear this process too depends upon cut-and-try methods for its completion. No process has yet been devised that avoids this unfortunate eventuality. However, if the testing appears to be too formidable, it can be programmed for execution by a digital computer.

7-5. Multiple-output Switching Circuits. Circuits of this type exist in great variety and are used for many purposes: as multiposition switches,

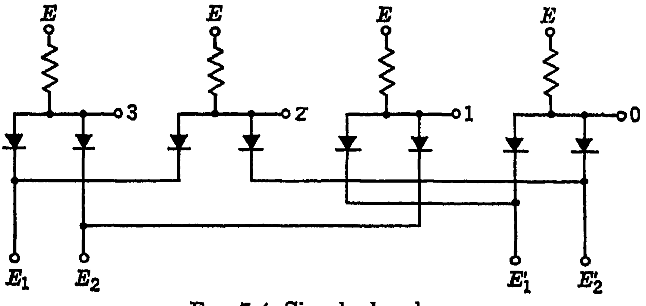


FIG. 7-4. Simple decoder.

as encoders, as decoders, and as arithmetical circuits. A very common type is constructed to accept n binary inputs and to yield 2^n outputs. Clearly, 2^n gates each having n inputs suffice to accomplish this, but it will be seen that more economical arrangements exist. Today, circuits of this kind are constructed almost exclusively of crystal diodes. Circuits built up of triodes and pentodes will not be discussed; a full treatment of them can be found in "Synthesis of Electronic Computing and Control Circuits," chap. 6. Some circuits built up of magnetic elements will be presented; these have the advantage of greater dependability.

Consider first the simplest case, $n = 2$. Let p_1 and p_2 be the inputs, the value 1 signifying the higher of two voltage levels and 0 the lower. Four diode gates having inputs (p_1,p_2), (p_1',p_2), (p_1,p_2'), and (p_1',p_2') generate the $2^2 = 4$ outputs $p_1 \wedge p_2$, $p_1' \wedge p_2$, $p_1 \wedge p_2'$, $p_1' \wedge p_2'$. In Fig. 7-4, the input voltages E_1 and E_2 can assume the levels, for example, E and 0, where E is positive; if $E_1 = E$, then $E_1' = 0$, and so on. The voltage supplied to the anodes of the crystals is any value equal to or greater than E. If E_1 and E_2 are both low, then point 0 is at E volts and points 1, 2, and 3 are at 0 volts. If E_2 is high and E_1 is low, then point 1 is at E volts and the rest at 0 volts, and so on; the outputs 0, 1, 2, 3 correspond to $p_1' \wedge p_2'$, $p_1' \wedge p_2$, $p_1 \wedge p_2'$, and $p_1 \wedge p_2$. A neater way of drawing the schematic is shown in Fig. 7-5; this is the conventional diode matrix and has the advantages of being easy to remember and to draw

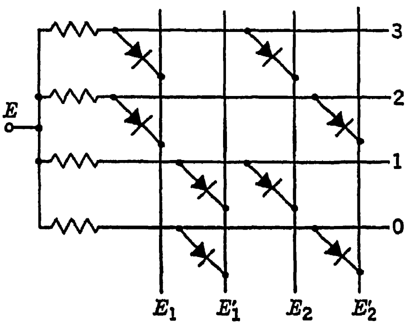


FIG. 7-5. Simple decoder drawn in form of a matrix.

for any n. Very commonly, in practice, the pair of leads to E_1 and E_1' are connected to the anodes of the tubes of a toggle.

For any integer n, the number of diodes required to form the matrix is $n \times 2^n$, since the circuit consists of 2^n gate circuits each having n inputs. More economical circuits will now be derived, i.e., the "pyramid" and the "rectangle."[1]

Consider, for example, the case $n = 4$. The $2^4 = 16$ outputs are

$$
\begin{array}{ll}
P_0 = p_1' \wedge p_2' \wedge p_3' \wedge p_4' & P_1 = p_1' \wedge p_2' \wedge p_3' \wedge p_4 \\
P_2 = p_1' \wedge p_2' \wedge p_3 \wedge p_4' & P_3 = p_1' \wedge p_2' \wedge p_3 \wedge p_4 \\
P_4 = p_1' \wedge p_2 \wedge p_3' \wedge p_4' & P_5 = p_1' \wedge p_2 \wedge p_3' \wedge p_4 \\
P_6 = p_1' \wedge p_2 \wedge p_3 \wedge p_4' & P_7 = p_1' \wedge p_2 \wedge p_3 \wedge p_4 \\
P_8 = p_1 \wedge p_2' \wedge p_3' \wedge p_4' & P_9 = p_1 \wedge p_2' \wedge p_3' \wedge p_4 \\
P_{10} = p_1 \wedge p_2' \wedge p_3 \wedge p_4' & P_{11} = p_1 \wedge p_2' \wedge p_3 \wedge p_4 \\
P_{12} = p_1 \wedge p_2 \wedge p_3' \wedge p_4' & P_{13} = p_1 \wedge p_2 \wedge p_3' \wedge p_4 \\
P_{14} = p_1 \wedge p_2 \wedge p_3 \wedge p_4' & P_{15} = p_1 \wedge p_2 \wedge p_3 \wedge p_4
\end{array}
\qquad (7\text{-}6)
$$

[1] We follow the terminology of "Synthesis of Electronic Computing and Control Circuits." The rectangle is called the most economical circuit in D. R. Brown and N. Rochester, Rectifier Networks for Multiposition Switching, *Proc. IRE*, vol. 37, no. 2, pp. 139–147, February, 1949.

Evidently, if we set $a = p'_1 \wedge p'_2$, $b = p'_1 \wedge p_2$, $c = p_1 \wedge p'_2$, $d = p_1 \wedge p_2$, this table can be written

$$\begin{array}{llll}
P_0 = a \wedge p'_3 \wedge p'_4 & P_1 = a \wedge p'_3 \wedge p_4 & P_2 = a \wedge p_3 \wedge p'_4 & \\
P_3 = a \wedge p_3 \wedge p_4 & P_4 = b \wedge p'_3 \wedge p'_4 & P_5 = b \wedge p'_3 \wedge p_4 & \\
P_6 = b \wedge p_3 \wedge p'_4 & P_7 = b \wedge p_3 \wedge p_4 & P_8 = c \wedge p'_3 \wedge p'_4 & (7\text{-}7) \\
P_9 = c \wedge p'_3 \wedge p_4 & P_{10} = c \wedge p_3 \wedge p'_4 & P_{11} = c \wedge p_3 \wedge p_4 & \\
P_{12} = d \wedge p'_3 \wedge p'_4 & P_{13} = d \wedge p'_3 \wedge p_4 & P_{14} = d \wedge p_3 \wedge p'_4 & \\
P_{15} = d \wedge p_3 \wedge p_4 & & &
\end{array}$$

Now, setting $e = a \wedge p'_3$, $f = a \wedge p_3$, $g = b \wedge p'_3$, $h = b \wedge p_3$, $i = c \wedge p'_3$, $j = c \wedge p_3$, $k = d \wedge p'_3$, and $l = d \wedge p_3$, we obtain

$$\begin{array}{lllll}
P_0 = e \wedge p'_4 & P_1 = e \wedge p_4 & P_2 = f \wedge p'_4 & P_3 = f \wedge p_4 & \\
P_4 = g \wedge p'_4 & P_5 = g \wedge p_4 & P_6 = h \wedge p'_4 & P_7 = h \wedge p_4 & (7\text{-}8) \\
P_8 = i \wedge p'_4 & P_9 = i \wedge p_4 & P_{10} = j \wedge p'_4 & P_{11} = j \wedge p_4 & \\
P_{12} = k \wedge p'_4 & P_{13} = k \wedge p_4 & P_{14} = l \wedge p'_4 & P_{15} = l \wedge p_4 &
\end{array}$$

or, setting $q = p'_3 \wedge p'_4$, $r = p'_3 \wedge p_4$, $s = p_3 \wedge p'_4$, $t = p_3 \wedge p_4$, we may write

$$\begin{array}{lllll}
P_0 = a \wedge q & P_1 = a \wedge r & P_2 = a \wedge s & P_3 = a \wedge t & \\
P_4 = b \wedge q & P_5 = b \wedge r & P_6 = b \wedge s & P_7 = b \wedge t & (7\text{-}9) \\
P_8 = c \wedge q & P_9 = c \wedge r & P_{10} = c \wedge s & P_{11} = c \wedge t & \\
P_{12} = d \wedge q & P_{13} = d \wedge r & P_{14} = d \wedge s & P_{15} = d \wedge t &
\end{array}$$

Consider first the generation of a, b, c, d; this requires four diode gates and can be accomplished by the circuit shown in logical form in Fig. 7-6. Now consider the expressions e, f, g, h, i, j, k, and l. We require for their generation eight diode gates, and, finally, from Eqs. (7-8), we must have 16 more diode gates to generate P_0 to P_{15}. The complete circuit, which is shown in logical form in Fig. 7-7, therefore requires

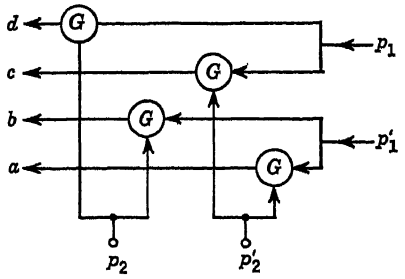


FIG. 7-6. First stage in deriving the pyramid.

$$(4 + 8 + 16)2 = 56$$

diodes, whereas the straightforward matrix required 64. It can be shown that an n-input circuit of this type incorporates $2^{n+2} - 8$ diodes. From Fig. 7-7 it is apparent why this configuration is called a pyramid.

Now consider the form (7-9) of the equations. Here we have but eight factors a to t to deal with, and it is clear from their form that the first four can be generated by a simple matrix consisting of four gates, and the second four by a similar circuit; each of a, b, c, d is then used as one input to a gate of which the other input is one of q, r, s, t. All told there are then 24 gates using 48 diodes. In general, it can be shown that the number R_n of diodes required for n inputs $p_1, p_2, \ldots, p_n$ satisfies recursion formulas

$$\begin{aligned} R_{2n} &= 2R_n + 2^{2n+1} \\ R_{2n+1} &= R_{n+1} + R_n + 2^{2n+2} \end{aligned} \tag{7-10}$$

The logical diagram of a rectangle for $n = 4$ is shown in Fig. 7-8. A table showing the actual number of diodes required for $2 \leq n \leq 8$ is given in the "Synthesis of Electronic Computing and Control Circuits."[1]

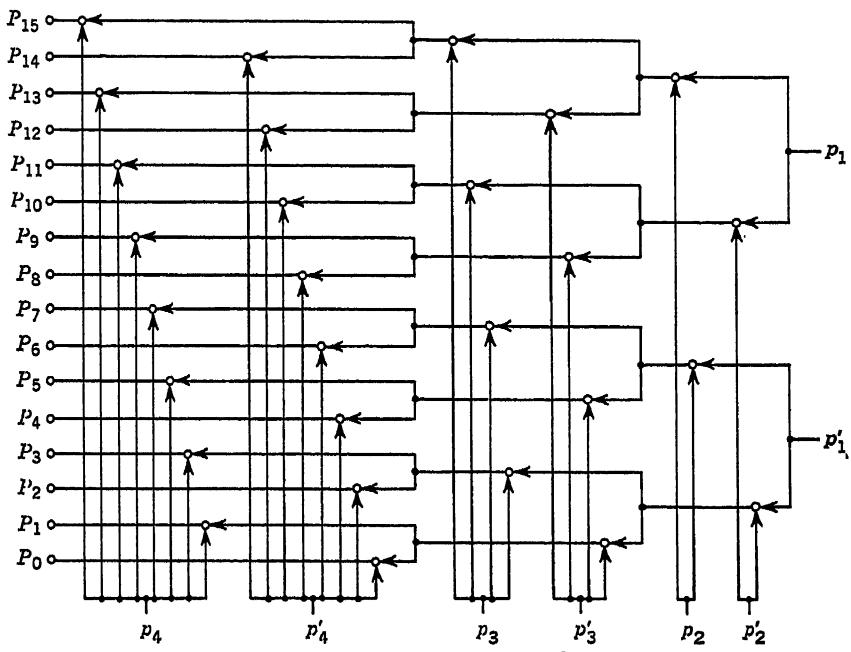


FIG. 7-7. Complete pyramid.

Nothing has been said concerning general procedures for the minimization of multiple-output switching circuits. This has been treated recently by Muller for the case in which the outputs can be expressed as a set of Boolean polynomials in the input variables. He has shown that, under these circumstances, the problem of simultaneously minimizing q polynomials in p variables can be reduced to the problem of minimizing a single polynomial in $p + q$ variables. His argument[2] will not be presented here.

[1] See "Synthesis of Electronic Computing and Control Circuits," pp. 133, 135, 137, 139 for the formulas for the number of diodes in each configuration and the table.

[2] D. E. Muller, Application of Boolean Algebra to Switching Circuit Design, *IRE Trans. on Computers*, vol. EC-3, no. 3, pp. 6–12, September, 1954.

7-6. Crystal-diode Circuits. So far this chapter has been devoted exclusively to questions of the logical design of switching networks. Once a logical structure has been settled upon, the task of working out an adequate physical realization remains, and this involves purely practical considerations of circuit design. It is necessary to take into account such things as the necessary voltage levels at input and output, the types of source from which inputs are derived, the circuits which the output is

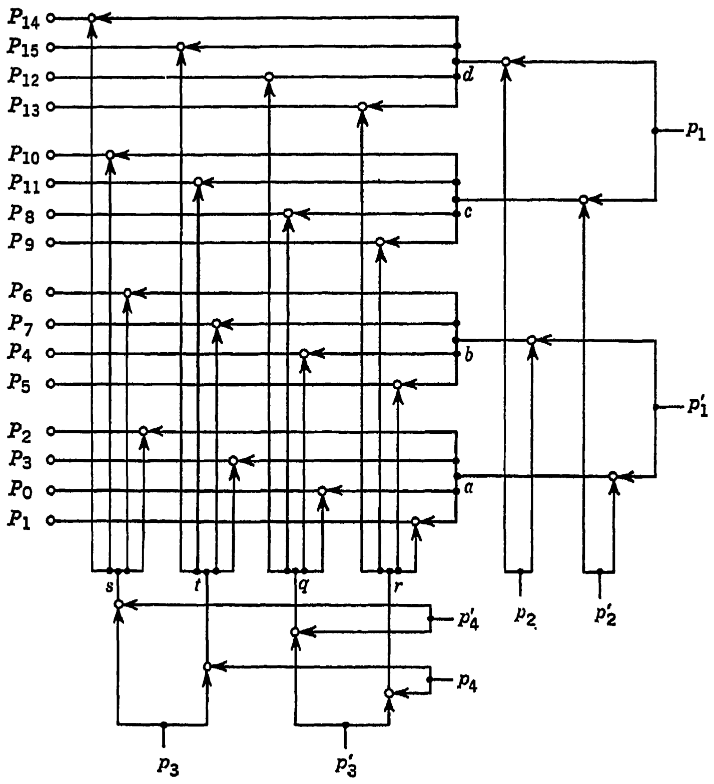


FIG. 7-8. Rectangle.

to drive, distributed capacitances as they affect rise-and-fall time, and the fact that the elements employed, e.g., crystal diodes, are not ideal.

Consider the circuit of Fig. 7-9, where the n inputs to the m mixers can assume the levels E_0 and E_1 ($E_0 > E_1$) and where E_0 and E_1 represent 1 and 0, respectively. If the diodes were ideal, possessing zero forward resistance and infinite back resistance, the output levels also would be E_0 and E_1. However, R_b is not infinite and R_f is not zero; so, for any given combination of input voltages, it is necessary to replace the diodes by the appropriate values of R_b and R_f; it is then possible, for any given combination of input voltages, to compute the output voltage. For this voltage to be high, the outputs of all m mixers must be high, and any one of these is high if at least one of its inputs is high; clearly, the greater the number of high inputs, the higher the output will be. Hence the most pessimistic estimate for the high output voltage is obtained when just one input to each is high. If the high value of the output is designated by E_1, then this value is $\bar{E}_{\min}$. Similarly, a low value $\underline{E}$ of output voltage is obtained if at least one mixer output is low; if more are low, then the output voltage is lower. The most pessimistic estimate $\underline{E}_{\max}$ occurs when the output of one is low and those of the others are as high as possible. Analysis of the equivalent circuits for these two cases, under the assumption that $R_f \ll R_1$, $R_2 \ll R_b$, shows that the difference $\bar{E}_{\min} - \underline{E}_{\max}$ is given by[1]

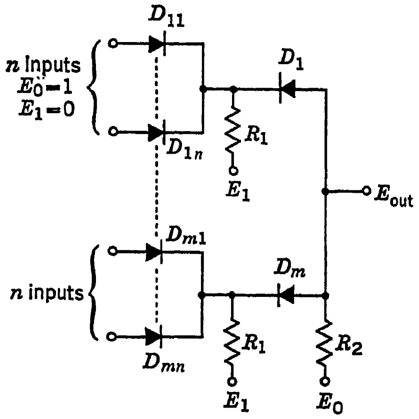


FIG. 7-9. Two-level circuit. (*An Wang, Proc. IRE, vol.* 40, *no.* 8, *p.* 931, *August,* 1952.)

$$(E_0 - E_1)\left\{1 - \frac{R_f}{R_f + R_1R_b/[R_b + (n-1)R_1]} - \frac{R_f + R_1}{R_f + R_1 + R_2R_b/[R_b + (m-1)R_2]}\right\}$$

The factor by which $E_0 - E_1$ is multiplied can easily become 0.5 or less. It is possible to compensate for this attenuation of the signal through the circuit by replacing the d-c supply voltage to the gate by a value greater than E_0.

Similarly, consider the output voltages of a matrix. Consider an n-input matrix of the form of Fig. 7-5, and suppose that

$$E_1 = E_2 = \cdots = E_n = 0 \qquad E'_1 = E'_2 = \cdots = E'_n = E$$

The diodes are then replaced by R_f or by R_b, as appropriate; for example, in the case $n = 2$, the circuit is shown in Fig. 7-10, the selected terminal being terminal 4. It is evident that terminals 1, 2, 3 are at positive voltages, whereas ideally they should be grounded. For the general situation, the equivalent circuit of Fig. 7-10 has been derived,[2] on the assumption that all unselected terminals are at the same voltage. On this basis, the

[1] An Wang, Miniature Rectifier Computing and Controlling Circuits, *Proc. IRE,* vol. 40, no. 8, pp. 931–936, August, 1952.

[2] Brown and Rochester, *op. cit.,* p. 143.

unselected-terminal voltages are given by

$$\frac{E}{1 + \dfrac{n2^{n-1}RR_b}{R_f[n(2^{n-1} - 1)R + (2^n - 1)R_b]}}$$

and in ordinary cases are small compared to the selected-terminal voltage E; in fact, as n becomes infinite the value of this expression approaches $(R_f/R_b)E$ as limit. For the case $n = 6$, $R = 10R_f = 0.1R_b$, the value is very nearly $E/24$. Thus a circuit of this type, operated as assumed, can be depended upon to give good discrimination between selected- and unselected-terminal voltages. For other modes of operation,[1] however, the discrimination can deteriorate seriously for large values of n.

The discussion so far has concerned only the effect of the nonideality of crystal diodes upon the d-c voltage levels of the outputs of switching circuits. The transient behavior, which will next be discussed, must be taken into account because it limits the speed at which the circuits can be operated.

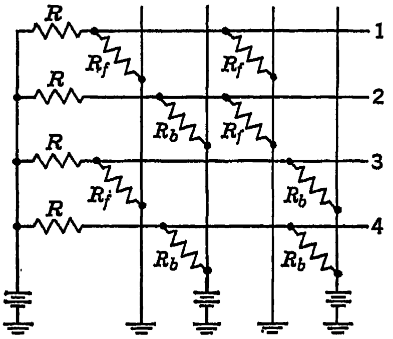


FIG. 7-10. Circuit used for analysis of matrix.

Crystal diodes have associated with them a capacitance. For germanium, this is so small that it is usually negligible compared with various wiring capacitances that are present. For selenium, on the other hand, it amounts to up to 70 pf (1 picofarad = 10^{-12} farad) for each diode,[2] enough to limit speed of response considerably. Thus, although selenium-diode circuits operate successfully in the 100-kcps range, circuits using germanium diodes normally operate at ten times this rate and may be operated at 100 times this rate. This unfortunate defect has limited the use of selenium diodes, which are otherwise attractive in that they are cheap and possess large back-to-front resistance ratios, e.g., up to 200 megohms back resistance and about 1,000 ohms forward.[3]

The effect of the fairly large capacitance of selenium diodes upon the response of circuits consisting of mixers followed by gates has been treated by An Wang in the paper already mentioned; as selenium diodes have not found wide acceptance, An Wang's discussion will not be presented here.

[1] *Ibid.*

[2] An Wang, *op. cit.*, p. 935.

[3] *Proc. Electronic Computer Symposium*, UCLA, Apr. 30–May 2, 1952. See panel report on Utilization of Germanium Diodes, p. VII-19.

In Sec. 4-10 the effect of distributed capacitance upon the rise time of the output of simple gates and mixers has been discussed. It is worthwhile at this point to see how this factor affects the design of some practical circuits. In the SEAC and DYSEAC the common circuit used throughout the machine is a regenerative amplifier which has already been described in Sec. 6-5; at its input are a number of gates feeding a mixer, the output of which is fed to the grid of the amplifier tube.[1–3] The design of such circuits has been discussed in detail in the literature and is of sufficient interest and importance to include here.

The circuit to be treated is typified by that shown in Fig. 7-11, where, for simplicity, only two gates with two inputs each are shown. The two gates are composed of diodes D_{11}, D_{12}, and D_{21}, D_{22}, and the mixer consists of D_1, D_2; the other diodes are used for isolation and bumping. In SEAC and DYSEAC, the dashed diode is also included; this diode and resistor R_1 hold the grid of the amplifier tube, which is connected to point 0, slightly above cutoff when the circuit is quiescent and also a few volts above the voltage at the anodes of D_1 and D_2, thus protecting the amplifier from responding to spurious signals.

The simplest and most obvious way to start the design is by working back from the output. The quiescent output value desired determines the value of e_1. Assume that the upper excursion of point 0 is also fixed, as by the (dashed) clamping diode in Fig. 7-11. The total capacitance from point 0 to ground (this includes input capacitance to the amplifier tube following point 0) is measured and designated by C_1. Suppose that a pulse causes one of the diodes D_1, D_2 to conduct and that, therefore, the point 0 is at its highest positive voltage V^+, the capacitance C_1 being charged to this value before the termination of the pulse. Then, at the termination of the pulse, the output voltage must decay toward its lower extreme V^- with time constant R_1C_1; the actual formula for the time of fall is (see Fig. 7-12)

$$t_f = R_1C_1 \ln \frac{V^+ - E_1}{V^- - E_1}$$

Having chosen V^+, V^-, and t_f, it is necessary to choose E_1 in order to calculate R_1 or, rather, to choose a suitable pair of values such that the equation is satisfied. E_1 must, of course, be more negative than e_1 in

[1] R. D. Elbourn and R. P. Witt, Dynamic Circuit Techniques Used in SEAC and DYSEAC, *Proc. IRE*, vol. 41, pp. 1380–1387, October, 1953.

[2] S. Greenwald, R. C. Haueter, and S. N. Alexander, SEAC, *Proc. IRE*, vol. 41, no. 10, pp. 1300–1313, October, 1953.

[3] S. E. Gluck, H. J. Gray, Jr., C. T. Leondes, and M. Rubinoff, The Design of Logical OR-AND-OR Pyramids for Digital Computers, *Proc. IRE*, vol. 41, no. 10, pp. 1388–1392, October, 1953.

order for D_a to conduct and so establish the quiescent value of output voltage. Normally it should be fairly large so that the current through R_1 will not change very much as the output voltage changes from V^- to V^+; on the other hand, if it is too large, an excessive amount of power is dissipated.

To treat rise time, suppose that all the inputs to one of the gates, for example, G_1, are simultaneously pulsed positive. The rise of voltage of point P_1 must depend upon the charging of the distributed capacitance

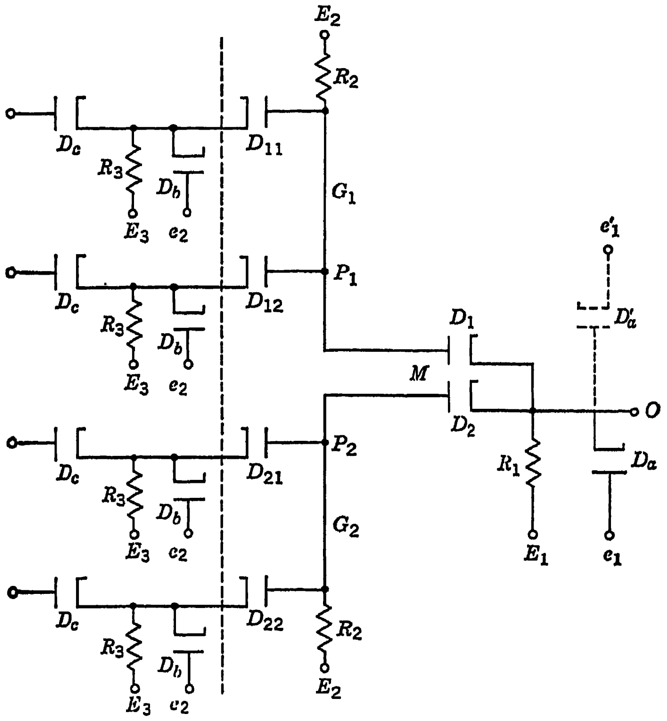


FIG. 7-11. Simple logic circuit discussed in the text.

C_2 between point P_1 and ground. The voltage initially across C_2 will be called $E_0(C_2)$; clearly, it is very nearly equal to the (negative) e_2, which should be more negative than e_1 but more positive than the quiescent value of input voltage in order to provide adequate filtering against noise pulses. The diode D_1 does not conduct until $E(C_2)$ has risen slightly above V^-, but after that time (generally very short) the rise can be determined from the circuit of Fig. 7-13, in which $(n - 1)R_b$ represents the back resistance of the nonconducting diodes of the mixer. It is evident that the time constant involved is that of $C_1 + C_2$ and of the parallel combination of R_1, R_2, and $(n - 1)R_b$. The actual expression for rise time involves this time constant multiplied by a rather complicated expression;[1] E_2 (positive) and R_2 must be chosen to be consistent with this. Again, E_2 should be large enough so that the total current through R_2 does not vary greatly, and due regard must be paid to the question of power dissipation.

Finally, it is necessary to choose values for E_3 and R_3. There is generally no reason why E_3 (negative) should not be the same as E_1. R_3 can then be determined, e.g., on the basis of allowable fall time at P_1 when one of the inputs falls from its high to its low value. In calculating the fall time at P_1, it is necessary to take account of C_3, the distributed capacitance from the cathode of D_c to ground, as well as the capacitance C_2 already mentioned above. There is no point in reproducing here the rather formidable equations that give fall time as a function of the other parameters.

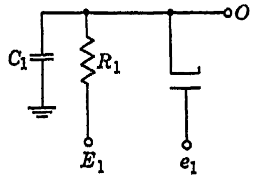


FIG. 7-12. Circuit for the calculation of fall time.

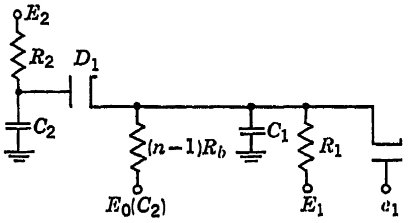


FIG. 7-13. Circuit for the calculation of rise time.

It is of some interest to cite the values of the parameters used in a practical circuit of this type (DYSEAC).[2] There are five gates ($n = 5$) with two, five, four, two, and two inputs, respectively. All the elements to the left of the dashed line are eliminated, adequate clamping of the quiescent voltage level being accomplished at the outputs of the circuits that drive the gates; the quiescent level is -8 volts, and information pulses are positive-going and 20 volts or more in amplitude. Both D_a and D'_a are used, with $e_1 = -5$, $e'_1 = 2$ volts. The other parameters are $E_1 = -65$, $E_2 = +62$ volts; $R_1 = 39$, $R_2 = 10$ kilohms; the measured capacitance from P_1 to ground (including the grid-cathode capacitance of the 6AN5, whose grid is connected to point 0) is 10 pf. With these values, and assuming 5 per cent tolerances in the most unfavorable direction in the resistor values, the rise and fall of the output voltage take place at 75 volts per sec, corresponding to rise and fall times of about 0.1 sec.

If smaller pulse amplitudes are used, it is possible to design for lower voltage levels and lower power consumption. The tube driven from point

[1] Gluck, Gray, Leondes, and Rubinoff, *op. cit.*, p. 1389.
[2] Elbourn and Witt, *op. cit.*, p. 1381.

0 can be of subminiature type, and the resistors can be miniature also, so that much less power need be dissipated. Since distributed capacitances are considerably reduced, short rise and fall times can readily be achieved. An experimental design, in which all d-c voltages were reduced by a factor of 5 (and, therefore, power dissipation by a factor of 25), is reported to give reliable operation with improved rise and fall times.[1]

7-7. Magnetic Matrices. In Sec. 4-14 it has been shown how transformers having two inputs and one output, wound on cores of material having a more or less rectangular hysteresis loop, can be made to function as gates. By an obvious application of this principle, multiposition switches can be built up. These first became of interest when magnetic-core memory arrays were being developed.[2,3] The attractive feature of

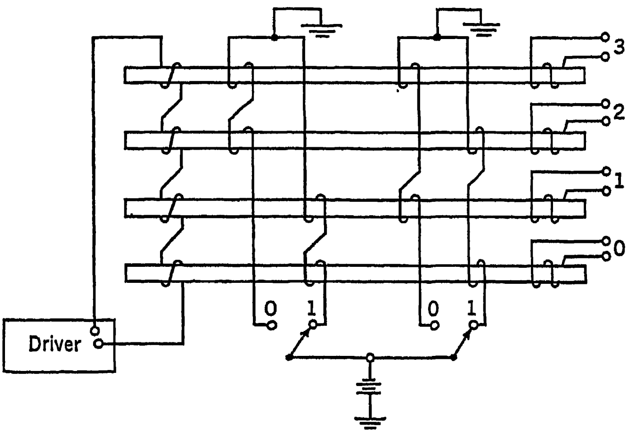


FIG. 7-14. Four-position magnetic-matrix switch.

magnetic switches in these applications is that they are capable of delivering the pulses of current, of amplitude of the order of an ampere or more, that are required to select the cores of the memory array. In this chapter only the switches will be discussed; an account of the memory arrays themselves will be found in Chap. 12.

For a very simple form of magnetic multiposition switch, one which amounts to a set of transformers with cores saturable by control currents, consider Fig. 7-14. The four cores—which have been represented as bars rather than as toroids, for simplicity—each carry four windings, two for selection, one for driving, and one for output. Current from the battery flowing through a selected winding is supposed to be of sufficient

[1] Gluck, Gray, Leondes, and Rubinoff, *op. cit.*, p. 1392.

[2] K. H. Olsen, A Magnetic Matrix Switch and Its Incorporation into a Coincident-current Memory, *MIT Digital Computer Lab. Rept.* R-211, 1952.

[3] J. A. Rajchman, Static Magnetic Matrix Memory and Switching Circuits, *RCA Rev.*, vol. 8, pp. 183–201, June, 1952 (Rajchman I).

strength to saturate the core, and the driver is supposed to put out a current pulse of sufficient amplitude to drive the cores from negative to positive saturation. Suppose that, initially, all the cores are saturated in the negative direction (field pointing to the left). When the input switches are set in the positions shown, the three lower cores all become saturated in the positive direction. When, subsequently, a pulse is put out by the driver, only the upper core suffers a substantial change of magnetic flux, and so a pulse output appears only at 3. To use a circuit like this repeatedly, it would be necessary after each operation to disconnect the inputs and drive all the cores to negative saturation. One winding per core can be dispensed with if the driver replaces the battery as the excitation of the first two (or the third and fourth) windings. The original experimental switches at MIT were of these two types.[1] The cores were ferramic with 20 turns per winding. The input switches ($n = 4$) were sets of flip-flops controlling hard-tube drivers.

A different scheme,[2] better adapted to pulse inputs, is shown in Fig. 7-15. Each core carries two windings for each binary position in the input, and these are wound in opposite senses, one producing magnetization in the positive (counterclockwise) and the other in the negative (clockwise) sense; the N (negative) winding has $K - 1$ as many turns as the P (positive) winding, where K is the number of binary places in the input (three in Fig. 7-15). Their distribution can be obtained by first drawing a circuit like that of Fig. 7-14 with only the P windings and then in each column inserting an N winding in each position left unfilled. The N windings at the extreme left are connected to a driver labeled "restore N." To see how the circuit works, suppose that all cores have been driven to N, and let positive pulses be applied to the grids of the left triode of each pair of input drivers, the right triode remaining cut off. The top core (000) is then driven to positive saturation by the K (three in the drawing) windings of n turns each or Kn turns. The second core (001) is driven in the positive direction by the $(K - 1)n$ turns of $K - 1$ windings and in the negative direction by the $(K - 1)n$ turns of a single N winding; so, as identical currents flow in all columns, it experiences no change. All other cores are driven in the negative direction by at least one winding of $(K - 1)n$ turns and in the positive direction by at most $K - 1$ windings of n turns each. Hence, all cores but the selected one [here the first (000)] either experience no net excitation or are driven strongly in the negative direction. Since all cores were negatively saturated to begin with, an output pulse is obtained only from the selected core. After the input pulses have terminated and before the next set is received, the restore-N driver is pulsed; this restores the selected core to

[1] Olsen, *op. cit.*, pp. 46–51, 56–62.

[2] Rajchman, *op. cit.*, pp. 187–189. (Reproduced by permission.)

negative magnetic saturation and does not change the others, which are already in that condition.

The switch just described, which has been called a magnetic "commutator" switch, becomes difficult to construct when the number of

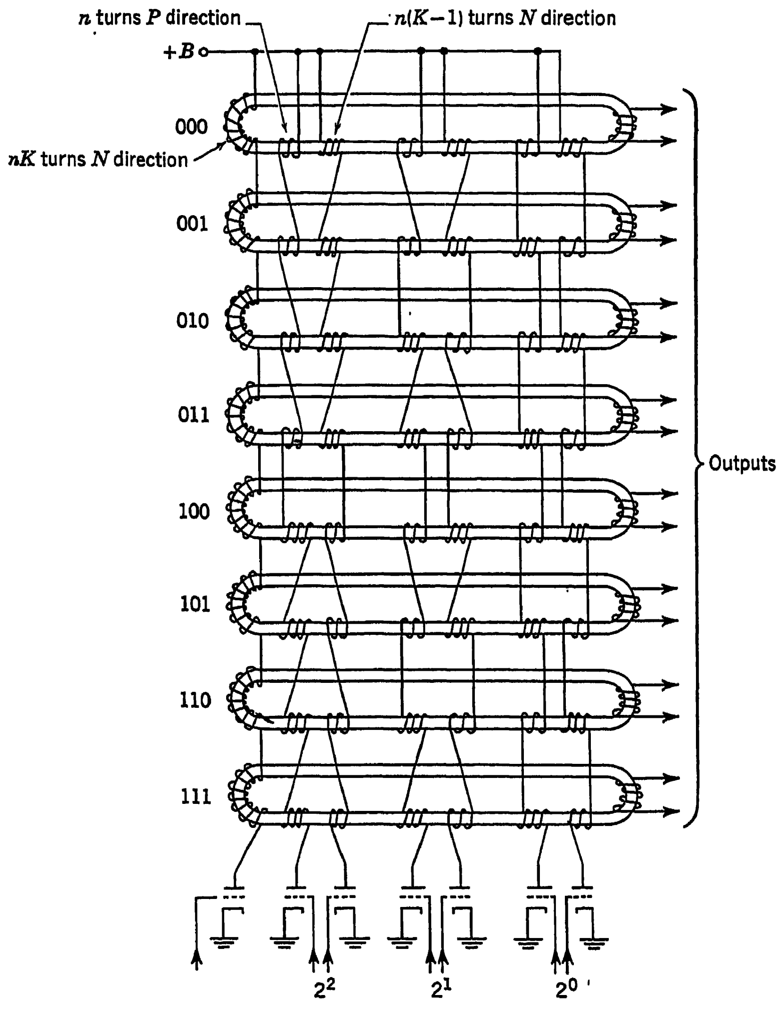


FIG. 7-15. Magnetic commutator switch.

cores is large, because of the number of windings that must be carried by each core; furthermore, the distributed capacitances between the windings may become so great as to produce such undesirable effects as delays and ringing. These troubles are avoided by the double-coincidence biased switch, which is shown in a simple form in Fig. 7-16.[1] The drawing shows a four-position switch, with one core per output. Each core carries four windings: two for selection, one for bias, and one for output; larger switches are constructed in exactly the same way. Inputs drive one horizontal and one vertical selection line, and the selected core is at the intersection of the excited lines. The drivers are biased so as to be normally nonconducting, and the inputs to the desired positions appear as positive pulses. The cores should have fairly rectangular hysteresis loops. The bias windings carry a d-c current of sufficient magnitude to produce a magnetizing force in the negative saturation

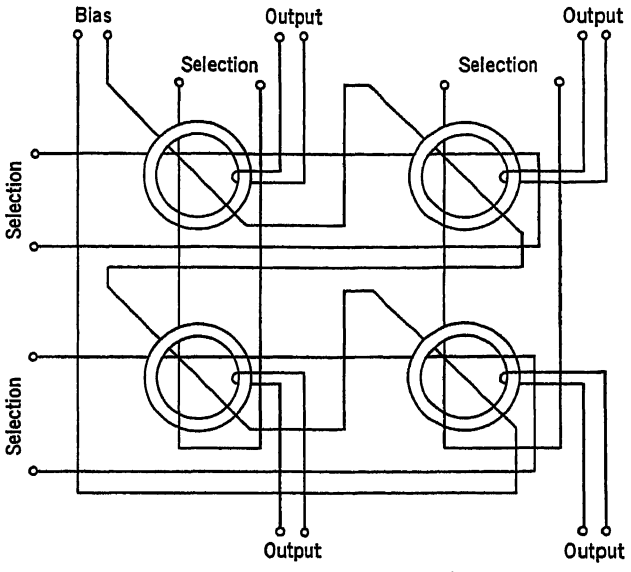


FIG. 7-16. Double-coincidence biased switch.

region, which is shown as H_B in Fig. 7-17; this magnitude is not critical. The selection windings are such that, when excited, they each produce a magnetizing force of magnitude H_B but in the positive direction; so, if one winding of a core is excited, no output (or at least very little) results, but if two are excited simultaneously a large output results, the core being driven to positive saturation during the pulse. In order to keep the output for a single excitation low, it is necessary to use cores whose *B-H* curves are flat from the remanence points out to saturation, as in Fig. 7-17. The magnitude of H_B can be chosen arbitrarily, provided it is sufficient to cause saturation; the turnover time of the cores is, roughly, inversely proportional to H_B. It is clear that the bias current always

[1] J. Rajchman, A Myriabit Magnetic-core Matrix Memory, *Proc. IRE*, vol. 41, no. 10, pp. 1407–1421, October, 1953 (Rajchman II).

returns the selected core to negative saturation at the termination of the input pulses.

The switch described by Rajchman was built up of cores made of molybdenum Permalloy tape wound on ceramic bobbins of about $\frac{5}{16}$ in. outer diameter and $\frac{3}{16}$ in. inner diameter. These were used because they yielded a rather good hysteresis curve, more nearly rectangular than that obtained with ferrites. The selection windings were of 16 turns each, a single winding linking all the cores of each row and a single winding linking all the cores of each column to reduce interturn capacitance. Bias

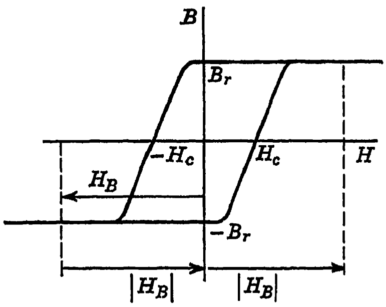


FIG. 7-17. Hysteresis loop.

and output windings were of one turn each, giving 600-ma output-current pulses to the load. These consisted, as they must in any such device, of a positive pulse followed by a negative pulse as the selected core moved back to negative saturation.

7-8. Other Circuits. The use of nonlinear resistors has been considered briefly in Sec. 4-15; clearly they could be used to replace diodes in matrix switches. It should also be possible to exploit a variety of nonlinear circuit elements for these and like purposes. The treatment here, however, will be limited to the diode and magnetic switches, for these are of chief importance in current practice.

CHAPTER 8

SHIFTING REGISTERS

8-1. Introduction. A variety of circuits for the storage of binary information and for the performance of logical operations have been presented. Chapter 7 showed how the elementary logic circuits could be combined into rather complex structures. Similarly, the present chapter will show how the basic storage circuits can be combined into complex structures which serve as repositories (usually temporary) of single words, in which words can not only be inserted and withdrawn as desired but also shifted to the right or left. The shifting feature makes certain of these structures suitable also for converting the transmission of information from serial to parallel and vice versa.

Generally speaking, a device that serves as the repository of a single word is called a "register," whether it consists merely of a set of locations in a large-scale memory device or of a set of toggles; the devices to be treated here are called "shifting registers." They are of great importance as elements of arithmetic circuitry, for multiplication and division are carried out as sequences of additions or subtractions and shifts. Great diversity of form is possible, according to the peculiar requirements of the system; it will be necessary to consider circuits well adapted some to serial and some to parallel transmission.

This chapter will describe some shifting registers built up of vacuum-tube toggles and magnetic binary elements and also registers of a circulating character, where the output is fed back to the input through a delay line. Instead of vacuum-tube toggles, it is possible, of course, to use transistor toggles, NBS dynamic binary elements, NBS diode-capacitor elements, or any other building blocks of like capabilities. It should be apparent that the circulating register is a natural type to use when information transmission is serial but is not adaptable to parallel transmission, whereas the other types can be used in both cases but are at their best in parallel, for then the access time of the register is the same as that of any one of its binary elements. If information is entered in parallel form, it can be removed serially by causing a number of shifts equal to the number of bits in the word; the register is then called a "dynamicizer." On the other hand, if information enters the element at one end 1 bit at a time and if the contents are shifted away from that end one position at a time as each bit is received until the register is full, then the circuit functions as a "staticizer," and, if so desired, the information from it can be transmitted in parallel.

A shifting register built up of vacuum-tube toggles and associated gates for the entrance and removal of information uses a considerable number of vacuum tubes. A nonshifting register similarly built up is not so expensive in tubes, but requires quite a few. In certain computers it has been found convenient to dispense with at least some such registers. Thus, in serial machines using magnetic-drum memories, it is fairly common practice to reserve for temporary storage a few tracks on the drum so arranged that a word written on one of them is read beginning one word time later and the output fed back to the writing head and rewritten, so that this particular word is always available with a maximum of one-word-time access time; in this way the first of a pair of addends to be removed from the main drum memory is kept on hand until the second addend has had time to emerge. Tracks so arranged are frequently referred to as "recirculating tracks."

8-2. A Positive-action Shifting Register. This section will be devoted to the shifting registers of the Institute for Advanced Study (IAS) computer,[1] or, properly speaking, to an archetype of these, for the shifting registers in the actual machine differ from the circuit to be described in various ways noted below. In Chap. 13, the interconnections of the shifting registers with the rest of the computer will be dealt with in detail.

The archetype consists of two ranks of 40 toggles each with associated gates. In Fig. 8-1 are shown two consecutive toggles of each rank, those corresponding to the binary positions 2^{-k} and $2^{-(k+1)}$; 2^0 (the sign position) is at the extreme left end of the register and 2^{-39} is at the extreme right end. The upper and lower ranks of the nth shifting register are called R^n and R_n as shown; thus RI stands for the first shifting register, and R^1 and R_1 are its upper and lower ranks. A number normally resides in the lower rank; when shifting is to take place, the number is transferred to the upper rank and then down to the left or right as desired. Thus the information is never in transit only, but is always securely held in at least one set of toggles, as will appear in the discussion below.

Consider now the detailed structure of the circuit. The convention adopted is that the toggles hold 0's when the left tube conducts, so that the left grid is at higher voltage (0 volts) than the right (−40 volts). There are two ways in which information can be introduced into the

[1] H. H. Goldstine, J. H. Pomerene, and Charles V. L. Smith, Final Progress Report on the Physical Realization of an Electronic Computing Instrument, written under contracts W-36-034ord7481 and DA-36-034ord19, Institute for Advanced Study, Princeton, N.J., January, 1954.

toggles. One is from a toggle of one rank to a toggle of another rank via one of the gates U_1, U_0, D_R, D_L. These are gates of the type shown in Fig. 4-4. In each case one input is taken from a grid of a toggle and, hence, can assume only the levels 0 and -40 volts; the other input is the disable-enable voltage of one of the driving buses G_1 to G_4, which is $+10$ volts to disable or -20 volts to enable. Thus, if the input from the toggle grid is high, then, when the bus voltage falls to -20 volts, current is drawn and this pulls down the plate voltage of the toggle to which transmission is to take place and causes it to switch in case the driven plate was originally high. It remains in the condition to which switched

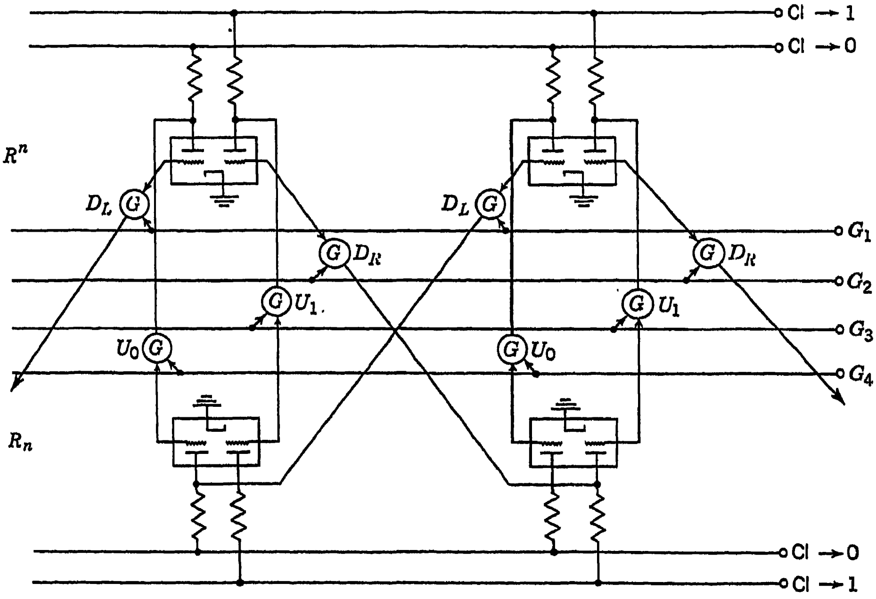


FIG. 8-1. Section of IAS shifting register.

after the bus resumes its higher voltage level. If the toggle grid driving the gate tube is at its low voltage, nothing whatever takes place when the bus falls to its enabling level. This means that each gate can transmit only a 0 or a 1, not both. U_0 can transmit only a 0, U_1 a 1, D_R a 1, and D_L a 0. For this reason the transfer of information always requires that the recipient rank be properly prepared. This is accomplished by the four "clear" buses (labeled "cl → 0," "cl → 1") shown in Fig. 8-1. These supply the $B+$ voltages to the toggles and can be set at one of two levels, $+150$ and $+50$ volts, the high for normal operation, the low for clearing. Thus, suppose it is desired to clear R^n to 0's. This is easily accomplished by maintaining the cl → 1 bus at $+150$ volts and dropping momentarily (for about 1 μsec) the cl → 0 bus to $+50$ volts.

With the gating and clearing capabilities firmly in mind, it is very easy to understand the method of generating shifts. Suppose that a left shift of the contents of R_n by one position is required. There are two possible ways in which the shift up to R^n can be performed. Suppose that it is desired to do it by using the U_1 gates. Since these can transmit only 1's, R^n is first cleared to 0's; after this has been accomplished and the cl $\rightarrow$ 0 bus has returned to +150 volts, the G_3 bus is dropped from +10 to −20 volts. If, e.g., the toggle in the 2^{-k} position in R_n holds 1, the conduction of current by the U_1-gate tube causes the toggle in the corresponding position in R^n to switch to 1. On the other hand, if the toggle in the 2^{-k} position in R_n holds 0, nothing happens. But the upper toggle has just been cleared to 0; so, in any case, the upper toggle finally holds the correct information. Next, the word now held in R^n must be moved back down to R_n via the D_L gates; as these transmit only 0's, R_n must first be cleared to 1's before the D_L gates are enabled, and the result is seen to be the desired one.

Similarly, a right shift can be accomplished by the sequence of operations: clear $R^n \rightarrow 0$, enable U_1 gates, clear $R_n \rightarrow 0$, enable D_R gates.

In any shift, certain information must be lost unless special precautions are taken to provide it with a place to go. In a left shift of one place, the original 2^0 bit would be lost if an extra toggle were not provided to receive it. Similarly, in a right shift, the 2^{-39} bit would be lost; in the case of the RI register of the IAS machine a path is provided by which this bit can be inserted in the high-order end of another register RII; this is done, in fact, during multiplication.

Although two "up" gates are shown in Fig. 8-1, in the actual registers of the IAS machine only the U_1 gates are connected as shown; the U_0 gates are driven by an information source other than the toggles of the lower rank and thus serve as the means by which information can be inserted (in parallel) into the registers. A detailed account of this is given in Chap. 13.

8-3. The Whirlwind I Shifting Registers. As an example of a quite different type of circuit, consider the shifting registers of Whirlwind I. A section of the "B register," which, e.g., holds the multiplier at the beginning of a multiplication, is shown in Fig. 8-2. Note at once that but a single rank of toggles is used; when a shift is ordered, the information is passed on via the delay-line elements shown in the drawing.

In Fig. 8-2, the convention is that a toggle holds 0 if the left tube is conducting. Read-in is accomplished by pulsing the left grids negative; so only 1's can be inserted. Before a read-in, the register is cleared to 0's by pulsing the right grids negative. The cathode connections of the toggles are used only for restoration, which is irrelevant to the logic; restoration is necessary only because the coupling from the toggle plates to the grids of the gate tubes is capacitive, and is accomplished by causing each toggle to switch twice at definite times determined by pulses at a submultiple of the clock frequency. Read-out is accomplished by pulsing the read-out-command line positive. The gates throughout the register are of the pentode type discussed in Chap. 4; so the coincidence of positive inputs to the control and suppressor grids yields a negative voltage swing at the plate. Hence, during the read-out-command pulse, negative pulses from the toggles that hold 1's appear at the outputs of the

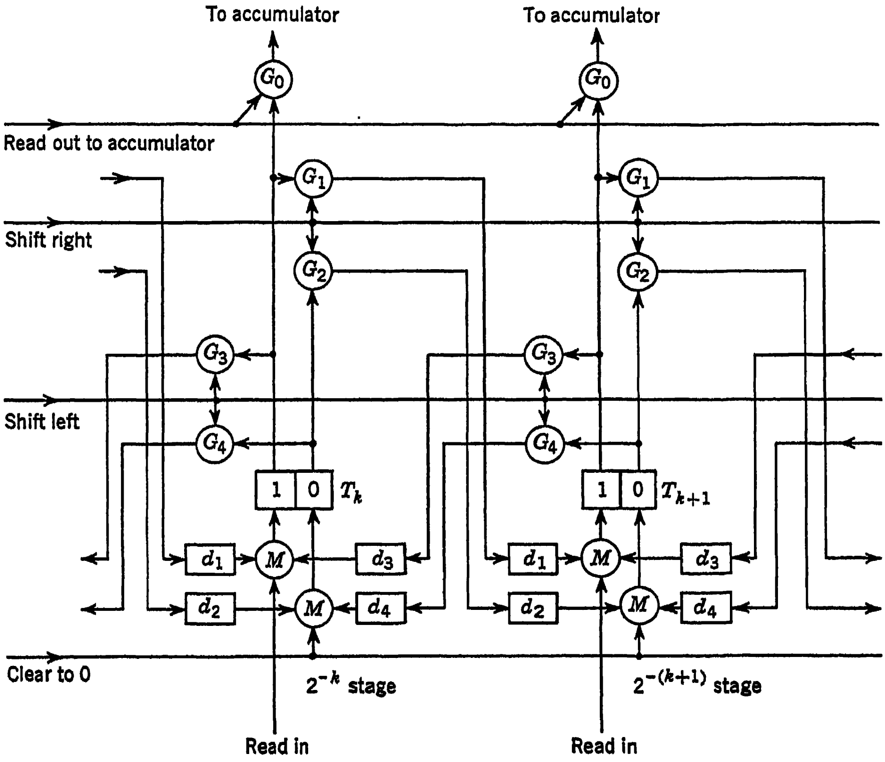


FIG. 8-2. Shifting in Whirlwind I.

read-out gates. The gates G_1, G_2 and the delay lines d_1, d_2 serve to carry out the right shift; similarly, the gates G_3, G_4 and the delay lines d_3, d_4 carry out the left shift. The delays are actually 0.1 μsec, the length of the standard Whirlwind pulse.

To see how the shifts are carried out, consider first the case of a right shift, and let the 2^{-k} stage hold 1. The left plate of T_k is at high voltage; so, when the right-shift-command line is pulsed positive, G_1 gives a negative pulse output, which is delayed by d_1 and finally arrives at the left grid of T_{k+1}, switching it to 1 if it does not already hold a 1. On the other hand, if the 2^{-k} stage initially holds 0, then G_2 responds during the right-shift-command pulse, and the negative pulse output, after delay by d_2, arrives at the right grid of T_{k+1} and switches it to 0. Hence, in both cases, it is guaranteed that T_{k+1} will hold the information held by T_k before the arrival of the shift-right command. The left shift is carried out in exactly the same way.

So far only the interior stages of the register have been considered; clearly some special arrangements must be made at the ends. (1) Output connections are taken from G_3 and G_4 in the 2^0 stage into the lowest-order (2^{-15}) stage of the accumulator. (2) Inputs are brought from outside to d_1 and d_2 of 2^0, so that 0's or 1's can be introduced into T_0. (3) In the 2^{-15} stage, G_1 and G_2 are used to sense the contents of T_{15}; they are enabled by a "digit-sense" pulse rather than by the shift-right-command pulse. The outputs go to the arithmetic control and are used in multiplication to determine whether or not the multiplicand is to be added to the partial product. (4) There is a special arrangement for inserting a 1 into T_{15} above. (5) The output of G_3 in the 2^{-15} stage is fed back, so that, if T_{15} holds 1, it is switched to 0 upon left shift, thus guaranteeing that the shifted number has 0's in all positions to the right of the originally least significant bit. (6) A special provision (used in division) is made so that the contents of all but the 2^0 stage can be shifted left. (7) An extra gate is attached to the left plate of T_0; the other input of this gate is a round-off pulse from the control, and its output is a round-off carry to the lowest-order stage of the accumulator (this is used in single-length rounded multiplication).

8-4. Other Types. In Fig. 8-3 are shown two other methods of effecting right shift. In Fig. 8-3*a*, the capacitors are charged up to the voltages of the plates to which they are connected; the right-shift-command line is normally held at ground and is pulsed negative by the shift command. Using the convention that a toggle holds 0 if the left tube is conducting, it is clear that if, say, T_k holds 0 and if a shift-command pulse of amplitude greater than the voltage of the conducting anode is applied, the grid of the right tube of T_{k+1} is driven below ground—with proper design, far enough to cut off the tube and so switch T_{k+1} to 0 if it holds 1. The situation if T_k holds 1 is similarly treated. The shift command also causes T_{k+2} to assume the state of T_{k+1}. Notice that the switching of T_{k+1} causes no confusion because its original state is "remembered" by the charges on the capacitors C_1^{k+1}, C_2^{k+2}, which, given a proper choice of R, do not change appreciably while the right-shift-command pulse is being applied. The time constants must, of course, be chosen such that the charges on the capacitors have sufficient time to readjust themselves in the interval between successive right shifts.

Still another scheme is shown in Fig. 8-3*b*. If the T_k switches from 1 to 0, the sharp drop in voltage of the plate of the left tube is propagated along the delay line, finally pulling down the grid of the left tube in T_{k+1} if that stage holds 0. Hence, to cause the transfer of the bit in T_k to T_{k+1} it is necessary only to clear all the toggles to 0. This (*a*) causes a negative step to be transmitted along the delay lines by those and only those stages originally holding 1's, and (*b*) prepares each toggle to receive a 1 if this is transmitted to it by its neighbor on the left. This automatically transfers 0's also. The length δ of the delay must be greater than the length of the "clear" pulse.

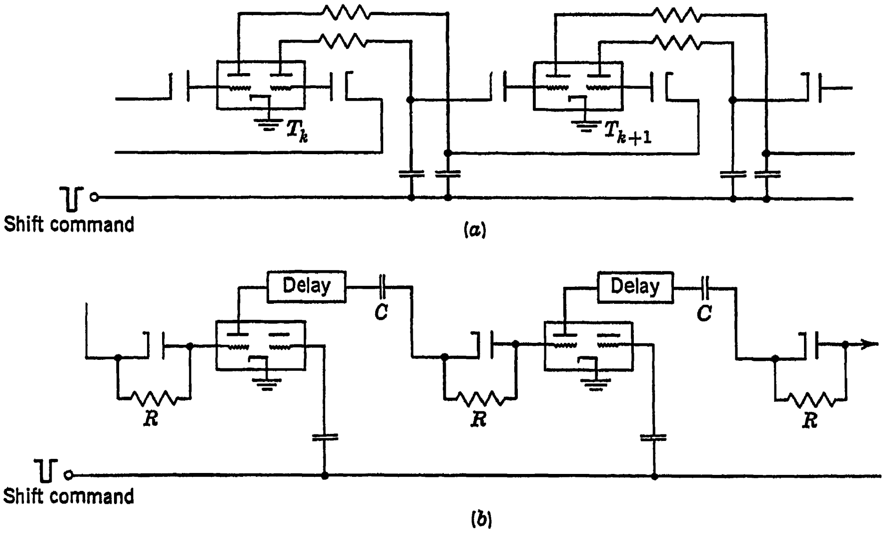


FIG. 8-3. Two types of shifting register. [*A. D. Booth and K. H. V. Booth, "Automatic Digital Calculators," p.* 104, *Butterworth & Co. (Publishers) Ltd., London,* 1953.]

8-5. Transistor Shifting Registers. There is no need to discuss in any detail the question of shifting registers that employ transistors as binary memory elements, since logically these resemble vacuum-tube registers.[1] Naturally, because of the nature of the input signals required and the outputs available, the read-in and read-out circuits must generally be rather different from those used in vacuum-tube registers; examples will be found in Chap. 5.

8-6. Other Binary Elements. NBS diode-capacitor memory cells or the IBM microsecond-delay units described in Chap. 6 can also be used as the memory elements of shifting registers.

[1] For example, a two-rank shifting register logically resembling the IAS design has been described in R. A. Kudlich and Hwa-Nien Yu, A Study of Some Direct-coupled Asynchronous Computer Circuits Using Transistors, report under ONR contract N6ori07124, Digital Computer Laboratory, The University of Illinois, October, 1954.

The shifting registers of the IBM 701 computer were built up of the delay units. The way in which the shifting is accomplished is shown in simplified logical form in Fig. 8-4.[1] To understand this, it is necessary to recall that, in the absence of an input, the contents of each delay unit are regenerated every microsecond by a positive-going synch pulse (from -30 to $+10$ volts); on the other hand, if new information is to be entered, the regeneration path is interrupted by a read-in-control signal, and the new information enters the unit at the time of the synch pulse. In Fig. 8-4, the input to each delay unit is provided by a mixer whose inputs are received from three gates G_1, G_2, and G_3; G_1 is enabled for right shift, G_3

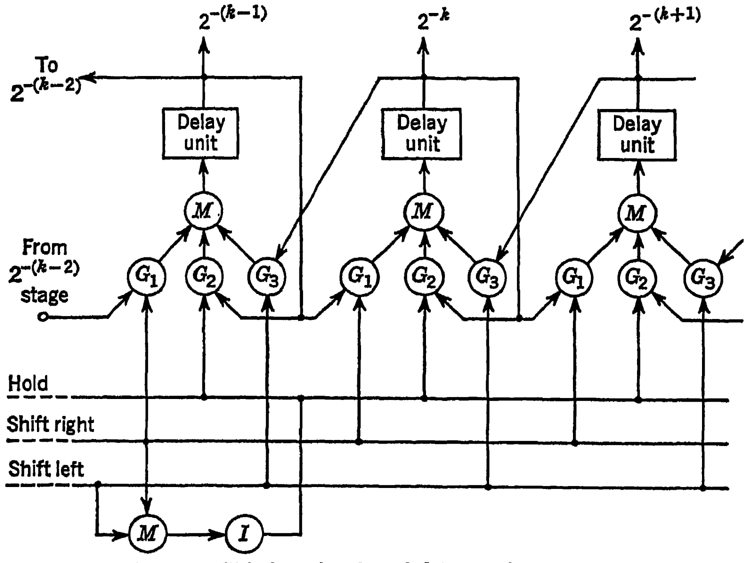


FIG. 8-4. Shifting circuits of delay-unit register.

for left shift, and G_2 to provide for normal regeneration of the contents of the delay unit. The gates are controlled by three lines for shift left, shift right, and hold. The shift lines are held at -30 volts if no shift is ordered, but are pulsed up to $+10$ volts by the shift command; -30 volts disables the gates and $+10$ volts enables. The two shift lines drive a mixer, the output of which is inverted, so that (1) if both shift lines are at -30 volts, then the inverter output is $+10$ volts, and (2) if a shift line rises to $+10$ volts, the inverter falls to -30 volts. Hence, if no shift is ordered, G_2 is enabled, as noted above, but if (1) a left shift is called for, G_1 and G_2 are disabled, but G_3 is enabled, and if (2) a right shift is ordered, G_2 and G_3 are disabled, but G_1 is enabled. The synch and

[1] H. D. Ross, The Arithmetic Element of the IBM Type 701 Computer, *Proc. IRE*, vol. 41, no. 10, pp. 1287–1294, October, 1953.

clamp signals then ensure that the transfers of information from delay unit to delay unit actually take place. As synch and clamp pulses occur every microsecond, the number of places shifted is determined by the length of time the shift line is raised to +10 volts.

It must be emphasized that Fig. 8-4 is incomplete in that it omits the synch, clamp, input, and output circuits, showing only the shifting circuits, but this is sufficient for present purposes.

8-7. Magnetic-core Registers. Magnetic-core shifting registers were first developed in the Harvard Computation Laboratory.[1] The earlier

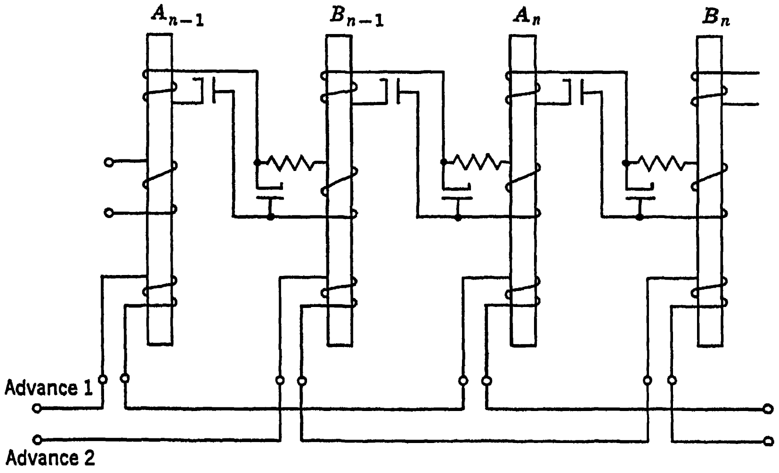


FIG. 8-5. Magnetic shifting register using two cores per bit.

form of this circuit is shown in Fig. 8-5; it requires two cores per bit, each core carrying three windings.[2-6] As in Chap. 4, the negatively saturated state of magnetization will be taken to represent 0, the positive state 1. Suppose that initially all cores are in the 0 state. Suppose that a pulse appears at the input, of correct polarity and sufficient amplitude to drive the first core to positive saturation. The upper winding is supposed to be such that the induced voltage is prevented by the series

[1] See *Harvard Computation Lab. Progress Repts.* 1 to 10 under Air Force contract W-19-122ac24.

[2] An Wang and Way Dong Woo, Static Magnetic Storage and Delay Line, *J. Appl. Phys.*, vol. 21, no. 1, pp. 49–54, January, 1950.

[3] M. Kincaid, J. M. Allen, and R. B. Hanna, Static Magnetic Memory for Low Cost Computers, *Electronics*, vol. 24, no. 1, pp. 108–111, January, 1951.

[4] An Wang, Magnetic Delay Line Storage, *Proc. IRE*, vol. 39, no. 4, pp. 401–407, April, 1951.

[5] A. D. Booth, The Physical Realization of an Electronic Digital Computer, *Electronic Eng.*, vol. 22, no. 274, pp. 492–498, December, 1950.

[6] E. A. Sands, An Analysis of Magnetic Shift Register Operation, *Proc. IRE*, vol. 41, no. 8, pp. 993–999, August, 1953, gives a treatment of the design problem.

diode from driving any current through the input winding of the second core, a 1 is safely registered in the first core, and no disturbance is propagated beyond it. Certainly, before the first core can acquire another bit of information, it must be cleared to 0; this action destroys the bit just registered there, so some place must be provided in which it can be held until needed. It is for this reason that the second core is provided. As soon as the first bit has been registered in the first core, advance pulse 1 is applied to the lower winding, of the correct polarity and amplitude to switch the core back to negative saturation, i.e., clear it to 0. During this operation, however, the voltage induced across the upper winding is opposite in polarity to that induced by the input pulse, and so the series diode allows current to be driven through the input winding of the second core, switching it to positive saturation, while the voltage induced in the top winding of the second core is prevented by the series diode from driving current through the input winding of the third core. The next step is to apply to the second core advance pulse 2, which clears the core to 0 and drives current through the input winding of the third core, switching it to 1. A voltage is also induced across the input winding of the second core, but this is prevented by the series diode from driving current through the output winding of the first core; on the other hand, the shunt diode now conducts, and the induced power is dissipated in the resistor. Thus the net effect of an input pulse representing 1 and followed by two advance pulses is to put the 1 into the third core and to clear the first two cores to 0's. Furthermore, it is clear that, if there is no input to the first core, the net effect of the advance pulses is nil and that, after they have been applied, all three cores still hold 0's. Hence the register is filled 1 bit at a time, two advance pulses occurring after each pulse time in which an information pulse can occur. Once the register has been filled, a right shift of one position can be caused by a pair of advance pulses; this causes a pulse output to appear at the extreme right-hand end of the register if that position holds a 1. In this way, by causing n right shifts, an n-bit word is made to emerge serially from a register consisting of $2n$ cores.

It is possible to use such a register also as a dynamic memory, by feeding the output back to the input and by causing a word once inserted to circulate continually about the loop. Used in this way, the circuit is properly referred to as a "magnetic delay line"; indeed, the original design worked out at the Harvard Computation Laboratory contemplated this type of operation.

The registers just described can fairly be considered to be of the positive-action type, for elements exactly like the main memory elements are used as intermediate memory elements. A dynamic type of register is obtained if the intermediate memory function is performed by capacitors, which are charged when the advance pulse clears all cores and which are then permitted to discharge through the input windings of the succeeding cores. The circuit of such a register,[1] described as "single-line" as distinguished from the "double-line" type already described, is shown in Fig. 8-6. This circuit is, admittedly, simpler than that of Fig. 8-5, for only half as many cores and less than half as many crystal diodes are used. Although but a single advance pulse is required, time must be allowed for the capacitor to lose most of its charge between advance pulses; so the speed of operation depends upon the time constant RC, and R cannot be made too small because it is necessary to ensure that, during the advance pulse, a large fraction of the output current is devoted to charging the capacitor. Nevertheless, a shifting speed of 100,000 pulses per second was reported for an early model of this register.

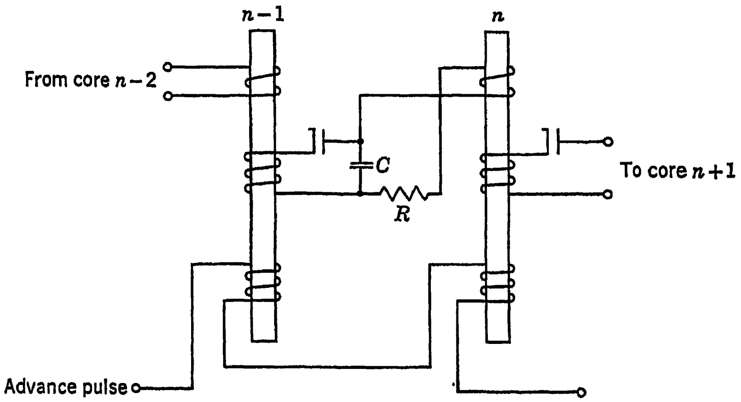


FIG. 8-6. Magnetic shifting register using one core per bit.

In the vacuum-tube shifting registers described earlier in this chapter, it is always possible to insert or extract information either in parallel (without shifting) or serially. Presumably, in a magnetic register, the parallel insertion could be accomplished by equipping each core with an extra winding and by clearing all cores to 0 before the parallel read-in operation. As the registers stand, this last step would entail a number of right shifts equal to the number of binary positions, the input at the left end being held at the 0 level, which would be wasteful of time. A more elaborate system of regulating the flow of information from core to core could be worked out, so that transmission could be enabled or disabled at will; this would permit the simultaneous clearing of all cores. Parallel read-out would involve problems too, for, in the schemes discussed, it would be necessary to cause a switching of the magnetic state of each core

[1] R. D. Kodis, S. Ruhman, and Way Dong Woo, Magnetic Shift Register Using One Core per Bit, *IRE Convention Record*, 1953, pt. 7, pp. 38–42.

in order to obtain output signals. A practicable method of nondestructive reading would make it possible to extract the information in parallel without having to cause a shift of the contents of the register.

8-8. Dynamic Shifting Registers. So far, in every case, some binary memory cell has been the basic element in every shifting register described. With magnetic elements, information is retained indefinitely until it is deliberately disturbed; with vacuum-tube toggles, retention is also indefinite—unless, of course, circuit elements or the power supply fails; in the capacitor memory cells, more or less gradual degradation of the information occurs, so periodic restoration is necessary. Still another type of memory element is thoroughly dynamic, the information being kept in circulation about a loop, so that it may be recovered at some

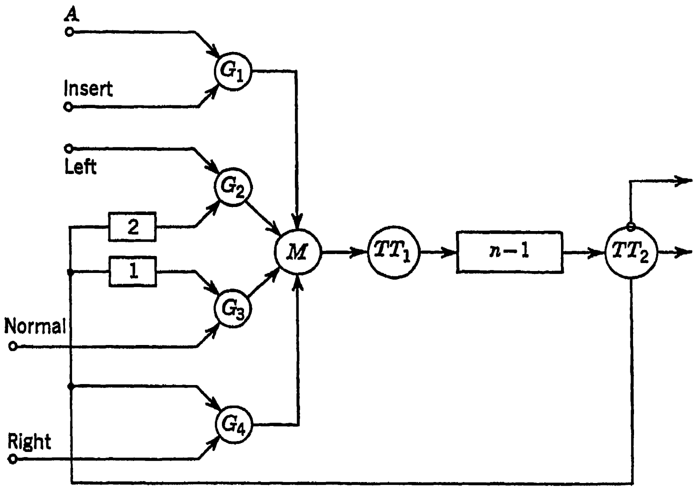


FIG. 8-7. NBS dynamic shifting register.

specific point at intervals equal to the loop transmission time. In Chap. 6 an account has been given of a binary element of this character, the NBS dynamic flip-flop; it remains to show how shifting registers can be devised by adopting this technique.

The shifting register of Fig. 8-7 is a direct extension of the single-bit memory circuit.[1] Information enters at input A, provided that "insert" pulses are present to enable gate G_1; a train of n insert pulses must be applied in synchronism with an n-bit word. The word that passes through G_1 then passes through the mixer M and is amplified by the tube-transformer circuit TT_1. Reshaping and reclocking are performed here also. The circuits are not shown, but can readily be supplied by the reader by referring to Fig. 6-9. The output pulses then travel down a

[1] A. L. Leiner, W. A. Notz, J. L. Smith, and A. Weinberger, System Design of the SEAC and DYSEAC, *NBS Circ.* 551, pp. 73–92, 1955.

delay line $n - 1$ pulse periods in length and upon emerging are amplified (and reshaped) by the tube-transformer circuit TT_2. A direct output (i.e., one that gives positive pulses for 1's) is fed back to gates G_2, G_3, G_4. If the word is merely to be held, the "normal" input is enabled, and the pulses proceed through the one-period delay line and G_3 back to M and then again to TT_1, and so on. Thus when the normal input is excited, the total delay around the loop is n pulse periods (n μsec in SEAC). The word circulating about the loop is available at the output every n microseconds.

For right shift, no enabling pulses appear at "normal," but a train of $n - 1$ pulses is applied to "right," beginning at the instant the second bit of the word reaches the output. Thus the one-period delay line is eliminated, and the pulse in it is lost; the pulse that was formerly the second to arrive at the output, representing the next-to-the-lowest-order bit, becomes the first to arrive at the output and, therefore, now represents the lowest-order bit. Thus the effect of eliminating the one-pulse-period delay for $n - 1$ pulse periods is that the former lowest-order bit of the word is lost and that all other bits are shifted one place to the right.

Similarly, for left shift, a train of enabling pulses is applied to the terminal marked "left" while right and normal are held at the disabling level; the first pulse of this train must arrive two pulse periods after the pulse representing the lowest-order bit emerges from TT_2. Thus all pulses of the word are delayed one pulse period longer than they are in normal recirculation; hence the next time that the pulse representing the lowest-order bit arrives at the output is one pulse period later than it would be normally. The net effect is to displace all bits of the word one place to the left. Notice that in left shift it is not necessary to lose the highest-order bit, since the total delay is made longer than normal. However, if this bit is not suppressed, it is shifted into the lowest-order position.

Although Fig. 8-7 shows a delay line $n - 1$ pulse periods in length, this length must be somewhat restricted since the pulses deteriorate intolerably if it is too great. This difficulty can be avoided by using several short delays, at the end of which amplification and reshaping of the pulses take place. An example is given in Fig. 8-8, which shows the 30-bit nonshifting registers used in the Manchester megacycle computer (MEG). The blocks labeled "*R* and *R*" are the reading-and-retiming amplifiers. The delay lines are each 9.35 μsec in length, 1.95 μsec delay being introduced by the other circuits. Information pulses are negative, so erasure is easily accomplished merely by making the erase terminal sufficiently positive, and writing is accomplished by applying the pulses of the word to be held to the write input. Four read outputs are provided so that blocks of 10 bits can be removed concurrently if desired.

It would be simple enough to convert the MEG register into a shifting register by decreasing the delay along the three lines by one pulse period (1 μsec) and by introducing a scheme like that shown in the diagram of the NBS shifting register. As a matter of fact, this is not required in the MEG arithmetic unit, as will be shown in Chap. 13. A variable delay circuit is incorporated in the adder, which is used to introduce any shifts needed in standardizing the floating point, and the shifts necessary in multiplication are provided by the delays in the multiplier loop; more will be said about these matters in Chap. 13.

Note also that acoustic lines or magnetostrictive lines (e.g., of nickel) can be used and in fact have been used as the delay elements of registers. A scheme like that shown in Fig. 8-7 can be used to introduce shifting.

The NBS dynamic binary elements (dynamic flip-flops) can also be used to construct a register analogous to the toggle registers discussed

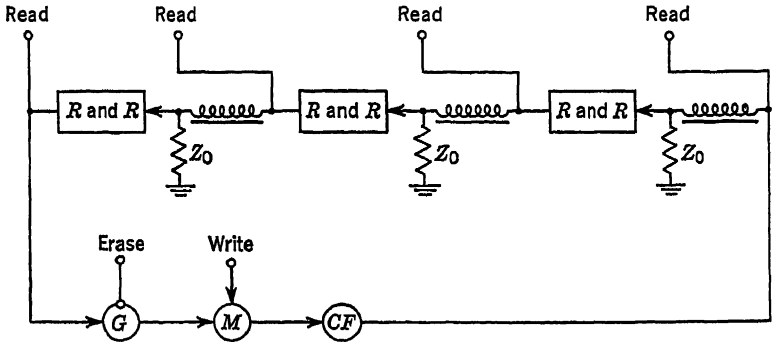


FIG. 8-8. MEG 30-bit register. (*D. B. G. Edwards, Ph.D. thesis, University of Manchester, England,* 1954.)

above, in which each bit of the word held has a definite repository. A very flexible circuit of this sort is shown in Fig. 8-9, in which any pulse-reshaping circuits have been omitted in the interest of clarity.[1] The circuit consists of a chain of n dynamic binary memory cells, each of which can transmit to its neighbors to the left or right, except that the left cell 1 can transmit to the next cell 2 and to n, whereas the rightmost cell n can transmit to the next left cell $n - 1$ and to 1. Thus circular shifts without loss of any information can be carried out if desired. Inputs are (a) serial at terminal B, or (b) parallel at terminals $A_1, A_2, A_3, \ldots, A_n$. Outputs are ($a$) serial at C_n, or (b) parallel at $C_1, C_2, \ldots, C_n$. For serial input to stage 1, the input gate must be enabled by a positive insert pulse I, whereas, for parallel input, it is necessary to enable the parallel-insertion gates by pulsing line P positive. To hold information, line H is pulsed positive once every pulse period (1 μsec) to enable the recircula-

[1] *Ibid.*

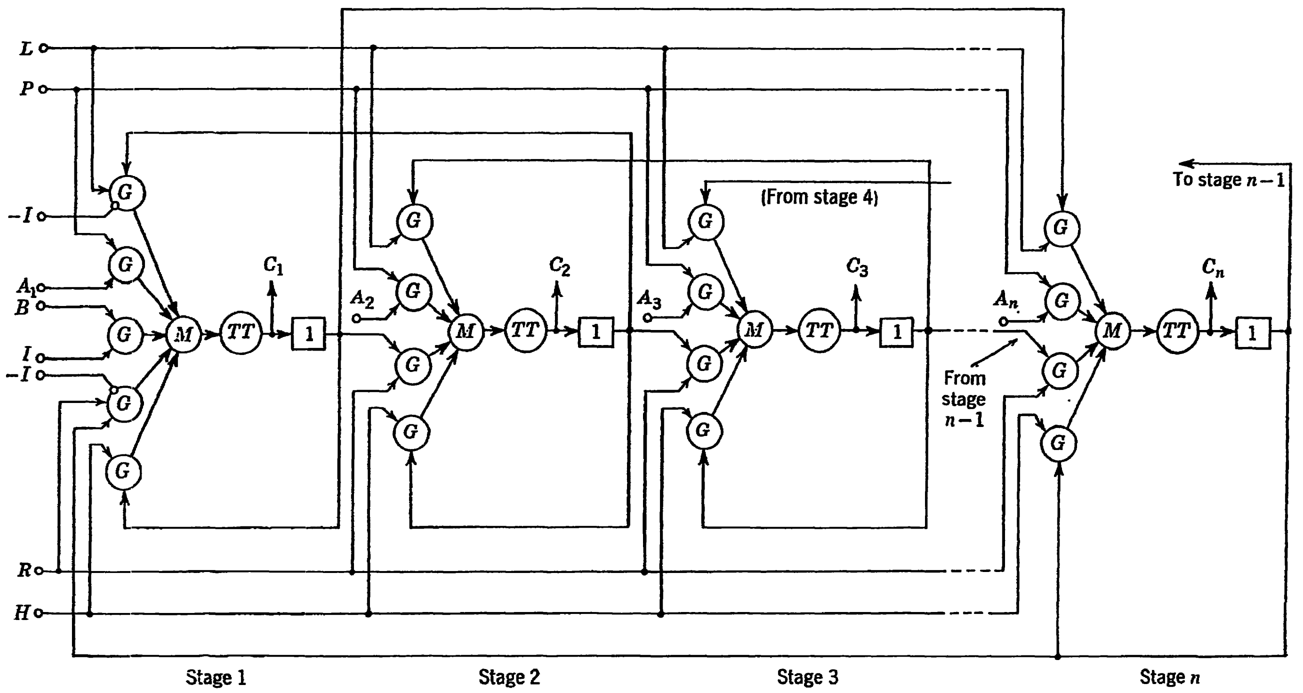


FIG. 8-9. NBS shifting register–dynamicizer–staticizer.

tion gates. To shift right (or left) one pulse, R (or L) is pulsed positive while H is not; so transmission to the right (or left) takes place while recirculation is inhibited. Note also that "insert" prevents the first stage from being influenced by either a right or a left shift.

The register of Fig. 8-9 can be used as a shifting register, as a staticizer, and as a dynamicizer. The last two terms can easily be misinterpreted in this context. A word once inserted serially is available thereafter for parallel transmission in the sense that, in the kth stage, a pulse appears at C_k once each microsecond if the stage holds a 1, and that this is true simultaneously for all stages. If it is desired to transmit the contents serially, it is of course possible to shift right n times in n successive microseconds, while the bits emerge successively from C_n.

8-9. Serial ↔ Parallel Conversion. In several computers it is necessary to convert from serial to parallel transmission of information and vice versa. Two typical situations in which this must be done will be cited.

First, in some machines it has been found advantageous to use a parallel memory but a serial arithmetic unit and generally serial data transmission. In the present state of the art, the most reliable high-speed memories are built up of rectangular matrices of magnetic cores. These were originally developed with the needs of parallel systems in mind, and parallel read-in and read-out are both natural and readily accomplished. On the other hand, many designers prefer serial to parallel circuitry, as more economical of circuit components, particularly of vacuum tubes. Serial ↔ parallel conversion makes it possible to achieve a reasonable design combining a parallel memory and serial arithmetic unit.

It may be objected that a large-scale memory composed entirely of magnetic shifting registers acting as delay lines after the original Harvard fashion would be a more reasonable memory for a serial machine than a parallel memory of the type mentioned above. Apart from considerations of equipment required, there is a very real question of access time. Suppose, for example, that the prf of the pulses of the machine words is to be 1 Mcps. The prf of the pulses emerging from a magnetic shifting register must be the same as the shifting rate, which is limited (given present core materials, at least) to considerably less than 1 Mcps. On the other hand, with parallel read-out, the output pulses can be made substantially less than 1 μsec in duration, either by shaping and differentiating them or by strobing them with a short pulse. The latter technique can also be applied to the parallel output of a toggle register to which the memory output is first delivered. If these short output pulses are applied to the points between 1-μsec delay lines arranged in a chain and if this is arranged so that transmission can take place only in one direction, the set of pulses simultaneously transmitted to the chain on the parallel lines must emerge as a train of pulses at a prf of 1 Mcps. Such a system, used in MEG for conversion of the output of a parallel Williams memory, is shown in Fig. 8-10, where the strobing is done in the gates G_0. The conversion from serial to parallel at the input is accomplished by opening the gates G_i for 1 μsec beginning at the instant the leading bit of the word to be recorded in the memory begins to emerge from the last 1-μsec delay line of the input chain.

A second important application of serial-to-parallel conversion occurs in the central control of serial computers. The bits of the instruction word that specify the operation to be performed must in some manner be decoded in order to derive the signals needed to actuate the circuits that perform the operation. This means that the individual bits of the

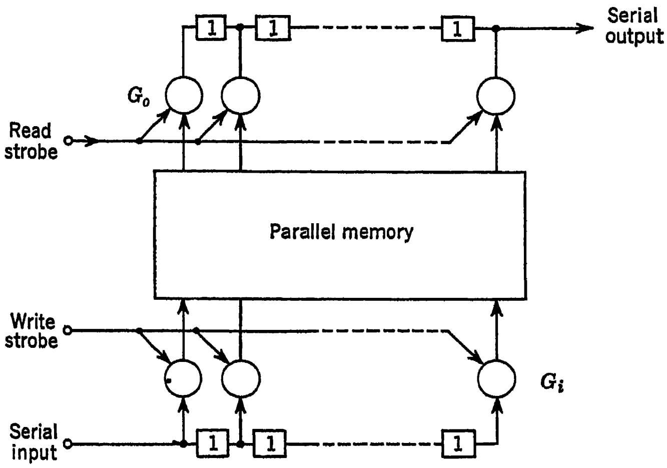


FIG. 8-10. Serial ↔ parallel conversion in MEG. (*D. B. G. Edwards, Ph.D. thesis, University of Manchester, England,* 1954.)

instruction must be simultaneously available. One way of doing this is to staticize them in a toggle register that transmits them in parallel to a crystal-diode matrix.

From the above discussion it is evident that the serial ↔ parallel conversions can be effected in different ways. The parallel-to-serial conversion can be effected either by the method of "flashing" into a delay-line chain, as shown in Fig. 8-10, or by reading first in parallel into a toggle shifting register and then shifting (e.g., right) a number of times equal to the word length, an output being taken from the endmost (e.g., rightmost) stage at each shift. A register that can receive in parallel and transmit in series is called a "dynamicizer." Serial-to-parallel conversion can be effected by the flashing technique from a chain of delay lines, as shown in Fig. 8-10, or by inserting the serial bits one by one into the endmost (e.g., leftmost) toggle of a shifting register and carrying out a shift (e.g., right) after each insertion. The flashing techniques can, of course, be used to insert information into and recover information from toggle registers that cannot shift. Any register that receives serial information to be held thereafter and kept available for parallel transmission is called a "staticizer."

CHAPTER 9

COUNTERS

9-1. The Toggle as Counter. Consider the simple circuit of Fig. 9-1 which consists of a toggle circuit with a common input to both grids. Let the toggle be set in the condition where the left section is conducting. If a negative pulse is applied at I, the left section is turned off. The input pulse tends to hold the right section off, too, but the rise in voltage of the left plate is communicated to the right grid and, if the design has been properly made, overrides the input pulse and causes the right section to conduct. A second input pulse returns the circuit to its original condition. The circuit is, therefore, a simple counter modulo 2,* or a "scale-of-2" counter.[1] Its chief defect lies in the necessity of solving the "pulse-dodging" problem,[1] that is, of ensuring that the plate rise is sufficient to override the negative input signal to the grid of the originally "off" section. Usually this requires the use of capacitors in parallel with the crossover resistors to secure the instantaneous transmission to the grid of the variations in plate voltage. The circuit is, thus, sensitive to variations in the amplitude and shape of the input pulses, which means that in many applications a pulse-standardizing circuit must be provided to form suitable input pulses. This drawback is inherent in all counters that depend upon elements of the form of Fig. 9-1 in which inputs are applied simultaneously to both grids. Later on counters will be presented that are built on the positive-action principle and that avoid this type of input.

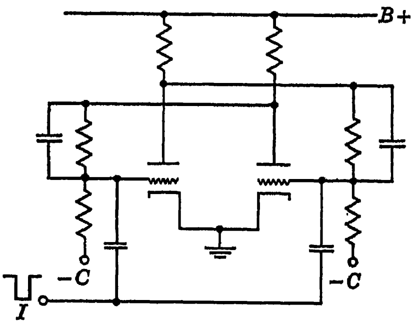


FIG. 9-1. A toggle as counter.

* "High-speed Computing Devices" by the Staff, Engineering Research Associates, McGraw-Hill Book Company, Inc., 1950, contains an interesting section on counters, chap. 3, pp. 12–31.

[1] T. K. Sharpless, High Speed N-scale Counters, *Electronics*, vol. 21, pp. 122–125, March, 1948.

Observe also that the circuit of Fig. 9-1 can be thought of as functioning as a modulo 2 adder; this fact is extensively exploited in parallel arithmetic units. Observe that, if the 0 state is taken to be that in which the left section conducts, the return of the circuit to this condition upon the reception of the second pulse can be indicated to other circuits by the sharp rise in voltage of the right plate or by the sharp fall in voltage of the left plate. These can be used as indications of carry when the circuit is used as an adder.

9-2. Counters Modulo *n*. Counting to an arbitrary number *n* can, obviously, be performed by any device that possesses *n* recognizably

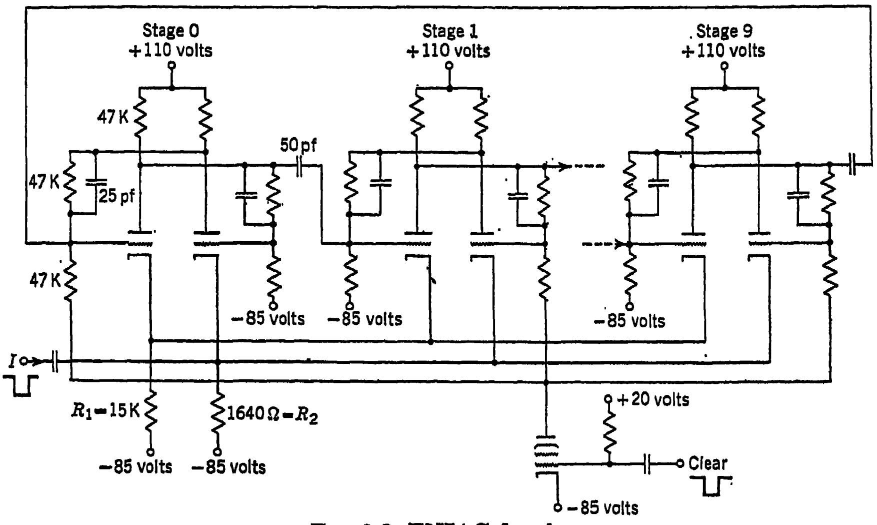


FIG. 9-2. ENIAC decade.

distinct stable states which it can be caused to assume in a definite sequence. A device of this character can be used as a memory element for the scale of notation based on *n*, just as a simple toggle is so used for binary notation. Thus the basic memory element of the first electronic digital computer, the ENIAC, was the decimal ring counter shown in Fig. 9-2.[1-3] This circuit requires $2n$ triodes if *n* stable states are to be obtained; it has been characterized as a "$2n$-triode counter modulo *n*."[1] Before describing this circuit, it is apposite to remark that a multistage binary scaler with feedback can also be used to realize a circuit having *n* stable states, with a considerable saving in the number of

[1] "High-speed Computing Devices," pp. 23, 24.

[2] A. W. Burks, Electronic Computing Circuits of the ENIAC, *Proc. IRE*, vol. 35, no. 7, pp. 756–767, August, 1947.

[3] Sharpless, *op. cit.*, p. 125.

vacuum tubes required; the precise arrangement will be described later on in this chapter.

The toggles in Fig. 9-2 use two triodes with separate cathodes (6SN7's in the original circuit) and have capacitors paralleling the crossover resistors. The 1 state of each toggle is taken to be the state in which the left triode conducts, so that the right plate voltage is high. Input and "clear" are both negative pulses. In the cleared state, $T_0 = 1$, $T_1 = T_2 = \cdots = T_9 = 0$. That this is indeed a stable state depends upon the fact that $R_1 \doteqdot 9R_2$ [$\doteqdot (n-1)R_2$ in an n-stage circuit], so that the left and right cathode voltages are identical. If an attempt were made to make $T_1 = 1$, the voltage of the right cathode would fall by one part in nine and, also, the left tubes of T_0 and T_1 could draw only about half as much current as the left tube of T_0 draws when T_0 alone holds 1. Consequently, the voltages of the left plates of T_0 and T_1 would rise, and the voltages of the right grids; and, since the tube characteristics cannot be expected to be identical, the right tube of one of the toggles would begin to conduct, and the usual action would turn the other off. Notice also that, if an attempt were made to turn all the toggles off, the left cathode would fall toward -85 volts, and one left triode at least would start to conduct when its cathode voltage fell within cutoff voltage of the grid. Thus we can accept the statement that the cleared state is stable. Similarly, any state in which one toggle holds 1 while all the others hold 0's is also stable.

Assume the circuit to be in its cleared state, and let a negative pulse appear at the input. This depresses the voltage of the right cathodes momentarily; since T_1 to T_9 have their right tubes conducting, no change takes place in them, but, if the pulse is of sufficient magnitude to depress the cathode voltage to within cutoff of the voltage of the right grid of T_0, the toggle T_0 is switched to 0. The abrupt rise of the voltage of the left-hand plate is communicated by the coupling capacitor to T_1, switching T_1 to 1. Clearly, the fall of the left plate voltage of T_1 cannot affect T_2; so the circuit is now in the stable state $T_0 = T_2 = T_3 = \cdots = T_9 = 0$, $T_1 = 1$. The next input pulse sets T_2 to 1 and all the rest to 0, and so on. The feedback arrangement from T_9 to T_0 guarantees that the tenth pulse will return the circuit to its original state.

Outputs can obviously be taken in a great variety of ways, e.g., from the right plates (only one right plate at a time is at its upper level of voltage), and visual indication can be given by a set of neon bulbs connected, e.g., from these plates to ground. An m-place scaler working in base n notation can be built of m of these circuits so arranged that the input to any one is generated by switching from 1 to 0 the last toggle T_{n-1} of the preceding circuit.

It has been stated that the ENIAC ten-stage circuit just described can operate satisfactorily at rates up to 180,000 pulses per second.[1,2]

The reader probably considers the counter just described to be rather wasteful of vacuum tubes. Successful modulo n counters using n tubes have been devised by Lewis.[3,4] However, the circuit becomes complicated if n is not fairly small; it is forced to discriminate between voltages that differ by smaller and smaller amounts as n is increased. The original circuit was devised for $n = 5$, and it does not seem possible to increase this beyond $n = 8$ without losing the necessary degree of reliability. Of course it is possible to cascade a circuit for $n = 5$ and a binary scaler to produce a decimal counter.

9-3. Scalers. It has been shown how circuits having n stable states can be devised and how they can be caused to step from one such state to the next upon the receipt of a pulse. These can be used only if n is fairly small, because the number of tubes used is proportional to n. If n is not small, recourse is had to the familiar scaling type of counter, which has been in use for many years in the instruments of nuclear and

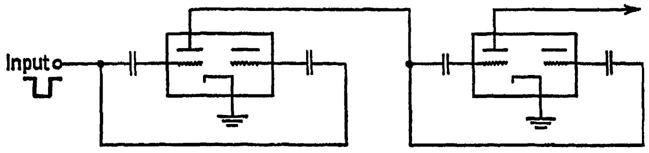


FIG. 9-3. Binary scaler.

cosmic-ray physics. This type of counter depends upon the fact, noted above, that, when a modulo n circuit steps from state $n - 1$ back to state 0, an indication is available that can be used to step a second such circuit. Thus, at every nth pulse received at the input of the first modulo n counting circuit, the second such circuit is caused to step once, and, at the n^2th pulse, a third such circuit is caused to step once, and so on. Apart from intercircuit coupling, a circuit having 2^n stable states requires $2n$ triodes. Note that the second, third, . . . modulo n counters in the chain need be designed to count at speeds only $1/n$, $1/n^2$, . . . of the speed of the first counter.

The simplest possible form of scaling is shown in Fig. 9-3. Assume that negative pulses are to be counted and that, initially, both toggles are in the 0 state with, say, the left triodes conducting. The first pulse switches the 2^0 stage to 1, transmitting a positive pulse to the grids of the second toggle; assume that the design is such that this pulse does not

[1] *Ibid.*

[2] Burks, *op. cit.*

[3] W. B. Lewis, "Electrical Counting," Cambridge University Press, New York, 1942. The circuit is given on p. 91.

[4] "High-speed Computing Devices," p. 20.

affect the second stage. The second pulse switches the 2^0 stage back to 0 and transmits a negative pulse to the 2^1 stage, switching it to 1. A third pulse now switches 2^0 to 1 and does not affect 2^1, and a fourth pulse switches 2^0 to 0, transmitting a negative pulse to 2^1 and switching this also to 0.

A scaling counter in the decimal system can be constructed similarly, if the two toggles of Fig. 9-3 are replaced, for example, by rings of ten of the form of Fig. 9-2 and if the coupling is so arranged that the transition of the lowest-order stage from 9 back to 0 causes the counter in the next-higher stage to advance by unity. In like fashion, scaling counters can be devised to operate in any desired scale of notation.

9-4. Binary-type Counters in an Arbitrary Scale. Binary scaling counters of the type of Fig. 9-3 possess 2^n stable states, which are successively assumed as the count progresses. By the use of feedback, the circuit can be caused to ignore certain of these states;[1] it thus becomes a counter in a different scale. For example, a four-stage binary counter can be caused to ignore six of its possible states and thus to possess in effect only ten distinguishable states, becoming a "decade"; from decades of this type it is possible to construct a decimal scaling counter. Considerable economy in vacuum tubes is achieved by using decades of this type, as compared with 20-triode rings, and even as compared with the Lewis-type ring mentioned above.

There are several possible methods of introducing feedback to achieve the desired result.[2–4] The circuit may be allowed to count initially to $2^n - 1$ (that is, 15) and, upon the next count, may go back not to 0 but to some different state (say 6); or it may be so arranged that it progresses normally through the successive stages up to $N - 1$ (where $N < 2^n$), returning to 0 upon the next count; or it may be caused to skip certain of its normally possible states. As an example of the first type of circuit,[5] consider Fig. 9-4. Initially, T_1, T_2, T_3, and T_4 are to have their right triodes drawing current. Let negative pulses be applied to the input. The counting proceeds as if no feedback connections were present until in every stage the left triode is conducting; this condition obtains after the reception of the 15th pulse. The 16th pulse now returns every stage to its 0 condition; this is a progressive process, first

[1] Harold Lifshutz, New Vacuum Tube Scaling Circuits of Arbitrary Integral or Fractional Scaling Rates, *Phys. Rev.*, p. 243, February, 1940.

[2] Richard J. Blume, Predetermined Counter for Process Control, *Electronics*, vol. 21, pp. 88–93, February, 1948.

[3] I. E. Grosdoff, Electronic Counters, *RCA Rev.*, vol. 7, pp. 438–447, September, 1946.

[4] John T. Potter, A Four Tube Counter Decade, *Electronics*, vol. 17, pp. 110–113, 358–360, June, 1944.

[5] Blume, *op. cit.*, p. 89.

T_1 switching to 0, then T_2, and so on. However, when T_4 switches to 0, the sharp rise in voltage of the plate of the left triode is transmitted to the grids of the left triodes of T_2 and T_3, turning these tubes on and leaving the circuit in the state that would normally signify 6, but which in this case is used to signify 0. From this point on, the circuit steps forward from "normal" 6 to "normal" 15, then back to "normal" 6 and so on, passing through only ten stable states. These may be used to represent digits 0 to 9.

As an example of the second type of circuit,[1] modify Fig. 9-4 as follows: (1) in place of the feedback connections shown, let there be one connection from the plate of the left tube in T_4 to the grid of the right tube in T_2; and (2) add a capacitive connection from the plate of the right tube in T_1 to the grid of the left tube in T_4. Thus, switching T_1 from 1 to 0

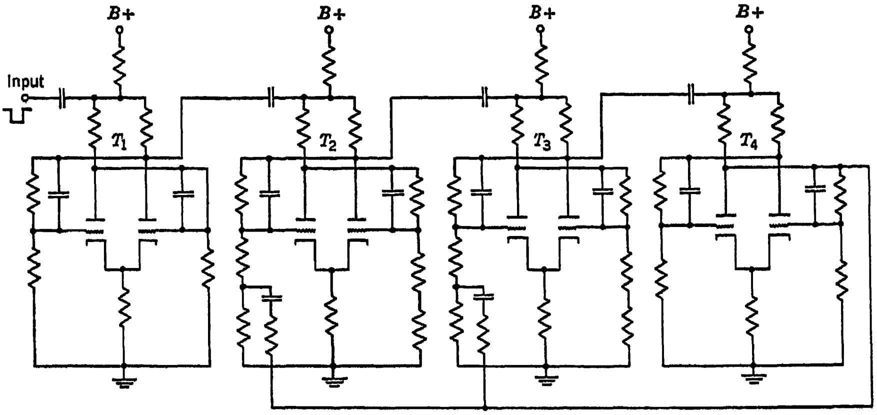


FIG. 9-4. Decade obtained by feedback.

causes T_4 to switch from 1 to 0 but does not otherwise affect T_4, and switching T_4 from 1 to 0 causes T_2 to switch from 1 to 0 but otherwise has no effect. With this understood, let a train of negative pulses be applied at the input. The count proceeds normally up to 9, at which point T_1 and T_4 hold 1's, and T_2 and T_3 hold 0's. The next pulse now switches T_1 to 0, and this does two things simultaneously: (1) it switches T_4 from 1 to 0, and (2) it causes the positive pulse fed back from T_4 to T_2 to override the negative switching pulse from T_1 to T_2, and so prevents T_2 from switching to 1, as it would if the feedback connection were not present. Thus the net effect of the tenth input pulse is to return the whole circuit to its initial, or 0, state.

9-5. Positive-action Counters. Ordinary scaling circuits suffer from dependence on pulse shape. Since the same input is fed to both grids, the voltage fed back from the plate of the tube that has its state of con-

[1] Potter, *op. cit.*, pp. 110–113.

duction changed first must override the input at the grid of the other tube. Or, if the circuit is of the cathode-coupled type, the initial state of the circuit must be remembered by the voltages of the crossover and distributed capacitances. In any case, therefore, the circuit is essentially called upon to "remember" its previous state so that it will "know" to what state to change.

The dependence of the ordinary scaling counter upon the character of the input pulse makes it unsatisfactory for use in computers of the asynchronous type, such as the machine at the Institute for Advanced Study (IAS) and those built under the guidance of the design philosophy evolved there.[1] Another consideration that in some cases makes it necessary to go beyond the simple scaler is that, even in a synchronous machine, the pulses to be counted may not be precisely defined in time and so it may be necessary to reestablish the timing with reference to the standard clock pulses.[2] In these and in other cases where it is necessary to avoid dependence upon input-pulse characteristics, a counter of the positive-action type provides a satisfactory solution. The general principle in circuits of this type has been stated as follows: "With each primary storage element there is associated a secondary element whose only function is to remember the state of the primary element during any interval in which it is changing or is in a state of change."[3]

Consider now two illustrations of counters embodying the positive-action principle. A typical stage of the first is shown in Fig. 9-5.[2] The "up" and "down" gates U_1, U_2, D_1, D_2 are pentode gates, and O_1 is a double-triode common-plate-resistor circuit and so functions as a gate for negative swings. The inverter I occurs only in the first stage, supplying all higher stages with a negative pulse to enable their output gates; the A' and B' outputs serve only to enable the D and U gates of the following stage.

Suppose both toggles initially hold 0, the left tubes conducting. Pulse A causes conduction by D_2 and sets the lower toggle to 1, and pulse B in turn causes conduction by U_1 and sets the upper toggle to 1. Note that, if B does not occur, the next A pulse has no effect. Output is taken from the right plate of the upper toggle. If the upper toggle is in the 0 state, one of the grids of O_1 is high and the output voltage is not affected by a negative pulse at the other input, whereas, if the upper toggle holds 1, the gate is enabled and the arrival of an A pulse causes the

[1] W. H. Ware, The Logical Principles of a New Kind of Binary Counter, *Proc. IRE*, vol. 41, no. 10, pp. 1429–1437, October, 1953.

[2] "Synthesis of Electronic Computing and Control Circuits," chap. 8.

[3] Ware, *op. cit.*, p. 1429. The positive-action principle was systematically applied by J. H. Bigelow in the design of the IAS computer. A description of his positive-action shifting register is given elsewhere in the present volume.

output to the next stage to be a positive pulse. This is the means chosen for providing the scaling action of the circuit. The second pair of pulses sets both toggles of the first stage to 0 and both toggles of the second stage to 1.

Note the complexity of the circuit. This is the price paid to achieve the type of action desired and to eliminate dependence upon pulse shape.

The second illustration is shown in Fig. 9-6.[1] This arrangement is like that of Fig. 9-5 in the up-and-down gating between toggles of the same column, but different in the gating between columns. In Ware's terminology, the upper rank of toggles is called the "true" rank and the lower the "false" rank; the reason for this will become evident shortly.

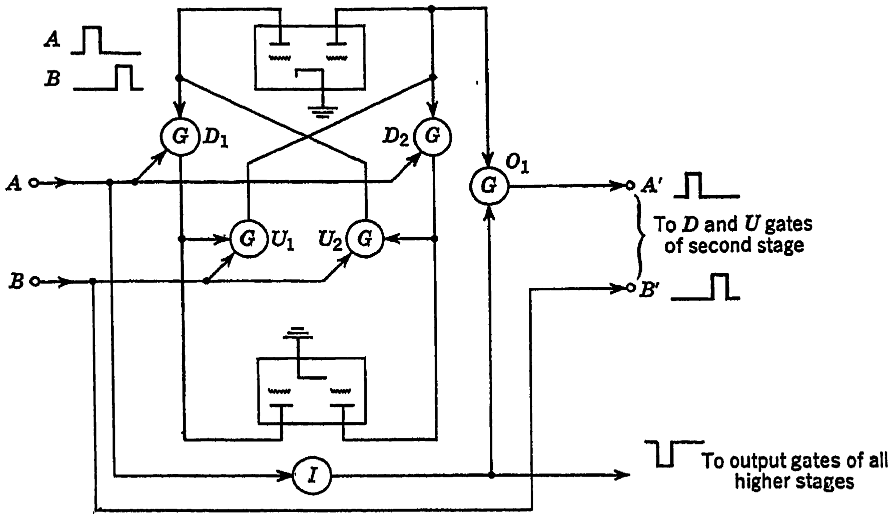

A
B
G
D_1
D_2
O_1
U_1
U_2
A
B
A'
To D and U gates of second stage
B'
I
To output gates of all higher stages

FIG. 9-5. One form of positive-action counter.

The convention is retained that a 0 is represented by a toggle with the left tube conducting.

The input to the lowest-order stage consists of a pair of pulses in sequence. The first opens the "down" gate and thus causes the F toggle to assume the same state as the T toggle above it. The second pulse opens the "up" gates and thus causes the T toggle to assume the state opposite to that of the F toggle (or the 1's complement of the information in the F toggle). These rules can be summarized by the symbols $T_0 \rightarrow F_0$, $F_0 \xrightarrow{c} T_0$. Thus, if both toggles of stage 0 are cleared to 0's before the

[1] *Ibid.*, p. 1431, and also Ware's unpublished Princeton thesis. A detailed discussion is given in Charles V. L. Smith, Sixth Interim Progress Report on the Physical Realization of an Electronic Computing Instrument, pp. 73–82, Institute for Advanced Study, Princeton, N.J., September, 1951. The same report also contains a detailed schematic of the shift counter designed by Ware for the Institute computer.

reception of the first pair of pulses, the net effect of this pair is to leave $T_0 = 1$, $F_0 = 0$, whereas the effect of the second pair is to leave $T_0 = 0$, $F_0 = 1$; this shows why the two ranks are named as they are. Thus the 0th column certainly behaves as it should.

Now examine the behavior of the next stage. The following requirements are evident: (1) if stage 0 changes from 0 to 1, there is no carry to stage 1, and stage 1 must therefore be unaffected; (2) if stage 0 changes from 1 to 0, there is a carry into stage 1, and the state of stage 1 must change. The following rules for transfer between T_1 and F_1 satisfy these requirements: (1) if $T_0 = 1$, then $T_1 \rightarrow F_1$; (2) if $T_0 = 0$, then $F_1 \xrightarrow{c} T_1$. A glance at Fig. 9-6 shows that these rules are in fact obeyed and that, in

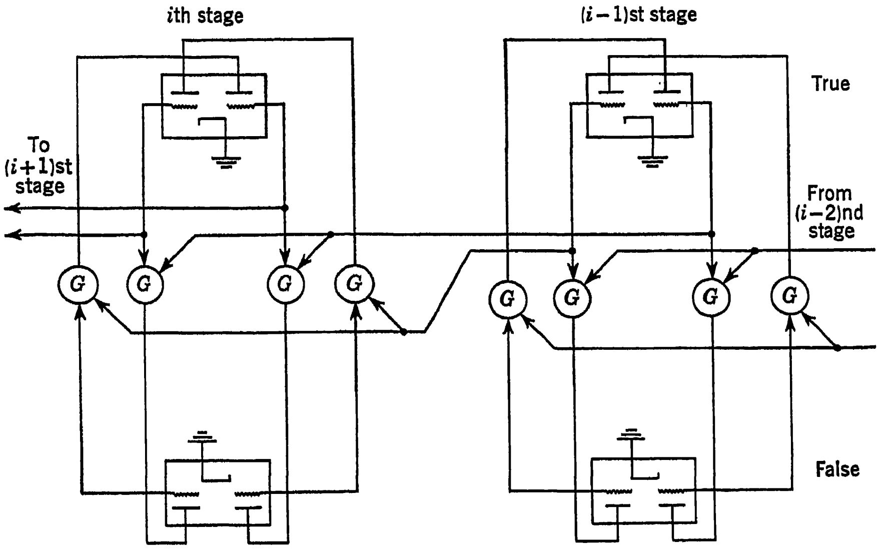


FIG. 9-6. IAS positive-action counter.

general, (1) if $T_{i-1} = 1$, then $T_i \rightarrow F_i$; (2) if $T_{i-1} = 0$, then $F_i \xrightarrow{c} T_i$; so it is guaranteed that the circuit is in fact a binary counter.

The numbers in the false rank corresponding to numbers given in the true rank are readily determined. Thus consider a five-stage counter, initially holding the number 00000 in the true rank. It is evident from the transfer rules that all the stages of the false rank beyond stage 0 must hold 1's. Furthermore, since the last operation of the stage 0 must have been to turn from 1 to 0, F_0 also holds a 1. The first few counts result in false numbers as follows:

True	*False*
00000	11111
00001	11100
00010	11001
00011	11010

There is no need to extend this table. Note only that to each true number corresponds a well-defined and unique false number. Since, in many computer uses, it is necessary to insert a number in a counter and to require it to begin counting from this value, it is clear that, in the present circuit, the false number corresponding to this desired initial value must also be inserted into the false rank. Thus F_0 must be the complement of T_0, and, for higher stages, if $T_{i-1} = 1$, then $T_i = F_i$; if $T_{i-1} = 0$, then F_i must be the complement of T_i. To ensure this requires merely the addition of another set of gates from the true to the false rank, which must be enabled only when an arbitrary number is being inserted into the counter and which are disabled during the ordinary counting process. These gates must be dynamic in the sense that they are enabled only upon command. It is possible also to make a subtracting counter by changing the gates to conform to the following transfer rules: (1) $F_0 \xrightarrow{c} T_0$, and, if $T_{i-1} = 1$, then $F_i \rightarrow T_i$; (2) $T_0 \rightarrow F_0$, and, if $T_{i-1} = 0$, then $T_i \xrightarrow{c} F_i$.

It has already been mentioned that an extra set of dynamic gates can be used to insert into the false rank the dual of a number inserted into the true rank. It is also quite possible to replace by dynamic gates all the $T \rightarrow F$ gates in the stages above the 0th, for a $T \rightarrow F$ transfer makes no change in any T_i and so affects no other $T \rightarrow F$ transfer; hence it suffices to open these gates only for the time required to change the state of a toggle. On the other hand, in the $F \rightarrow T$ transfer, a change in any T_i affects the transfer in the next higher stage, which means that, if dynamic gating were to be used, the enabling pulse would have to last longer than the longest carry-propagation time. One would expect that, if dynamic gating were used in the $T \rightarrow F$ transfer, the false numbers dual to the true ones would not be the same as those found with static (or self-instructed) gating. This is in fact the case. The interested reader will find no trouble in constructing a table of these numbers.

It has been pointed out that, when an arbitrary number is inserted into the true rank, the corresponding false number (its dual in this sense) must be inserted into the false rank and that this insertion requires a set of gates enabled only for this purpose. The rules given were: (1) $T_0 \xrightarrow{c} F_0$; (2) if $T_{i-1} = 0$, then $T_i \xrightarrow{c} F_i$; and (3) if $T_{i-1} = 1$, then $T_i \rightarrow F_i$. If rule 1 is replaced by (1′) $T_0 \rightarrow F_0$, then the rules 1′ and 2 are identical with two of the rules used in counting. If these new rules are used for constructing a false number dual to a number N read into the true rank, it turns out (as the reader can see by constructing a few examples) that, if the true rank holds N, then the false rank holds the number dual to $N + 1$. Hence, with dynamic gating, to start counting from a number N, simply insert N into the true rank and do no more. The $T \rightarrow F$ gates are all arranged to open in synchronism with the first of the two pulses representing the next count. This places in the false rank the dual of $N + 1$, which is exactly the number that should be in the false rank after the reception of the first of the two activating pulses anyway. Thus the use of dynamic gating permits counting and "dual-instructing" gates to be combined.

A column of the shift counter in the IAS computer is shown in Fig. 9-7; this circuit was devised in accordance with the principles outlined above. Only static gates are used. The stage shown is the lowest-order one. All the other stages except the last are identical with this; the last embodies certain simplifications made possible by the fact that the counter is required to count only up to 40. Input 1 is ordinarily maintained at ground potential, the 1-μsec pulse swinging down to -20 volts. Input 2 is normally held at $+60$ volts, the 1-μsec pulse, which occurs 1 μsec after the termination of the pulse at input 1, swinging up to 110 volts.

Let the tubes of the column be designated T_1 to T_7, reading down. T_1 is the false toggle, and the right section of T_7, all of T_6, and the left section of T_5 together form the true toggle. The up (or $F \rightarrow T$) gate consists of T_2. The down ($T \rightarrow F$) gating structure consists of T_3, T_4, the left section of T_7, and the right section of T_5. The voltage dividers between the grids of T_6 and -300 volts are so designed that, when the voltage at, say, the left grid of T_6 is 0, the voltage at the left grid of T_2 is -10, and, when the left grid of T_6 drops to -40 volts, the left grid of T_2 drops to nearly -50 volts.

Now consider the effect of a -20-volt pulse on input 1. It is clear that, before the pulse, the cathode of T_2 was held so far above either grid that neither section of T_2 was conducting. With the pulse, the cathode of T_2 falls to -20 volts, and current is drawn by the section of T_2 the grid of which is at -10 volts. If, for example, this is the left section, then T_1 is already forced to the state where the left section is conducting. Clearly, T_2 forces the state of the false toggle to imitate the state of the true one.

Now consider the effect of the second pulse, and assume that T_1 is in the 0 state, so that the left and right grids of T_4 are held at 0 and -40 volts, respectively. Before the pulse, the plate supply of T_4 is $+60$ volts, so the left plate of T_4 is substantially below 60 volts, and the right plate is $+60$ volts. As the plate voltages of T_6 are $+60$ or $+110$, it is clear that T_6 can be in either state. Now, when the pulse comes, the plate supply of T_4 suddenly increases to $+110$ volts; so the right plate of T_4 also rises to 110, and the left plate rises to about $+60$ volts. Then the right grid of T_5 rises to $+110$ volts, and is followed by the cathode. Hence, the right grid of T_6 is brought up above ground and so T_6 is forced to assume the 1 condition, no matter what its previous condition. The

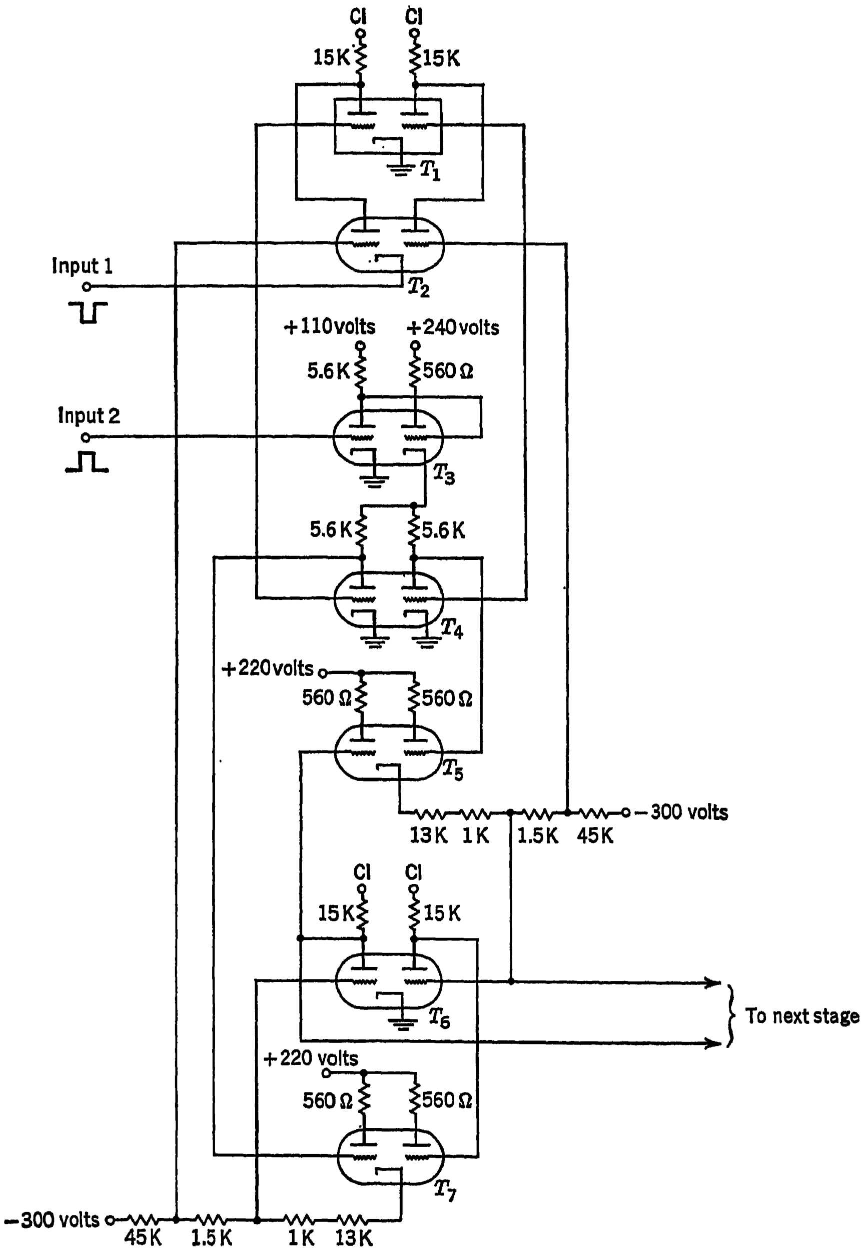


FIG. 9-7. One column of IAS shift counter.

down gates force the contents of the false toggle to become the complement of the contents of the true toggle.

9-6. A Double-ranked Positive-action Counter. Some types of high-speed memories are such that the information rapidly deteriorates and means must be provided of periodically restoring or regenerating the contents of the memory. The chief example is the type of electrostatic memory first proposed by F. C. Williams, which uses conventional cathode-ray tubes. In parallel memories of this type, it is convenient to alternate cycles of restoration and action. A counter is needed to count through all the addresses of the memory, and, in a machine of the single-address type, a counter must also be provided for counting through the successive addresses occupied by the instructions of the code being executed. It is, therefore, possible to effect a considerable economy in equipment by combining these two counters.[1] The resulting circuit consists of a single false rank, a true rank for restoration addresses, and a

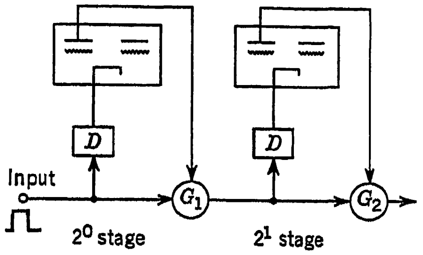


FIG. 9-8. Whirlwind I counter.

true rank for instruction addresses; this may be termed a "dispatch" or "central-exchange" counter. The arrangement becomes especially attractive when it is noted that, since the false numbers are unique and bear a one-to-one relation to the true numbers, the false numbers can be used within the memory to specify addresses and that, therefore, a single output from the false rank to the deflection circuits of the memory suffices.

Another type of dispatch counter will be described in Sec. 9-8.

9-7. Adding Counters. It is also possible to construct counters that are essentially simple adders, in which counting from N to $N + 1$ is accomplished by adding 1 to the contents of the lowest-order stage and permitting the carry to propagate throughout the circuit. Probably the simplest binary counter of this type is that used in Whirlwind I. It is shown in Fig. 9-8.

The toggles are cathode-coupled with capacitors paralleling the cross-over resistors, the gates are of the pentode type (see Chap. 4), and the delay elements are short sections of artificial line. As usual, the convention is that a toggle holds 0 when the left section is conducting.

[1] Ware, *op. cit.*, pp. 1436–1437.

Assume the counter cleared to 0. The first pulse received is stopped by the closed gate G_1 but, after a short delay (this must just exceed the duration of the pulse), switches T_1 to 1. The next pulse finds G_1 open and G_2 closed. It therefore switches both T_1 and T_2, so that now T_1 holds 0 and T_2 holds 1, as they should. It is now clear how the circuit operates. Observe that this circuit is really just as simple as an ordinary scaling circuit but may be expected to be somewhat faster, for the propagation time of the carry down the structure is only the transmission time along the carry line and through the carry gates.

9-8. An Adder-type Dispatch Counter. One form of double counter, the so-called "dispatch" or "central-exchange" counter has been described in Sec. 9-6. A brief description will be given here of the counter used in the IAS computer.

Physically this circuit consists of three ranks of ten toggles each, called the "restore" (T_R), the "dispatch" (T_D), and the "order" (T_O) toggles. The rank T_D may be associated with either of the other ranks to form a simple adding-type counter. Leads are taken from T_D to the deflection circuits in the electrostatic memory, and leads into T_D make it possible to insert an arbitrary number there when a shift or jump instruction is obeyed.

The general procedure is as follows: Supposing that the number in T_R is to be increased by unity, first clear T_D to 0's, then open the gates that force the contents of T_D to duplicate the contents of T_R. Then clear T_R and open the gates from T_D into T_R—these are so arranged that T_R receives a number greater by unity than the number held in T_D. Note that this counter is essentially of the positive-action type, for all information is held intact in one set of toggles until the transfer to another set has been accomplished.

A schematic of one column of the circuit is shown in Fig. 9-9. The convention here is that a toggle holds 1 if the left section of its tube is conducting (just the opposite of the convention followed earlier). The inputs are those labeled B_R, A_R, B_O, A_O; these are normally held at $+10$ volts but are dropped to -10 volts to enable the gates. The voltages $B\ \mathrm{Cl}_A$, $B\ \mathrm{Cl}_R$, $A\ \mathrm{Cl}_A$, $A\ \mathrm{Cl}_R$ are normally $+150$ volts, but are dropped to $+50$ volts when a toggle is to be cleared.

To increase the count in the restore rank (of which T_R is a representative toggle), we first clear the dispatch rank to 0's by pulsing $B\ \mathrm{Cl}_R$ down to $+50$ volts. The next step is to pulse B_R down to -10 volts, so that, if T_R holds 1, this is transmitted to T_D but, if T_R holds 0, then T_D continues to hold 0; the state of T_D must duplicate that of T_R.

Consider now the carry gate V_5 and its associated diodes V_6, V_7. The carry voltage from the preceding stage is positive (normally $+21$ volts) if the carry is 0, but negative (normally -40 volts) if the carry is 1; these

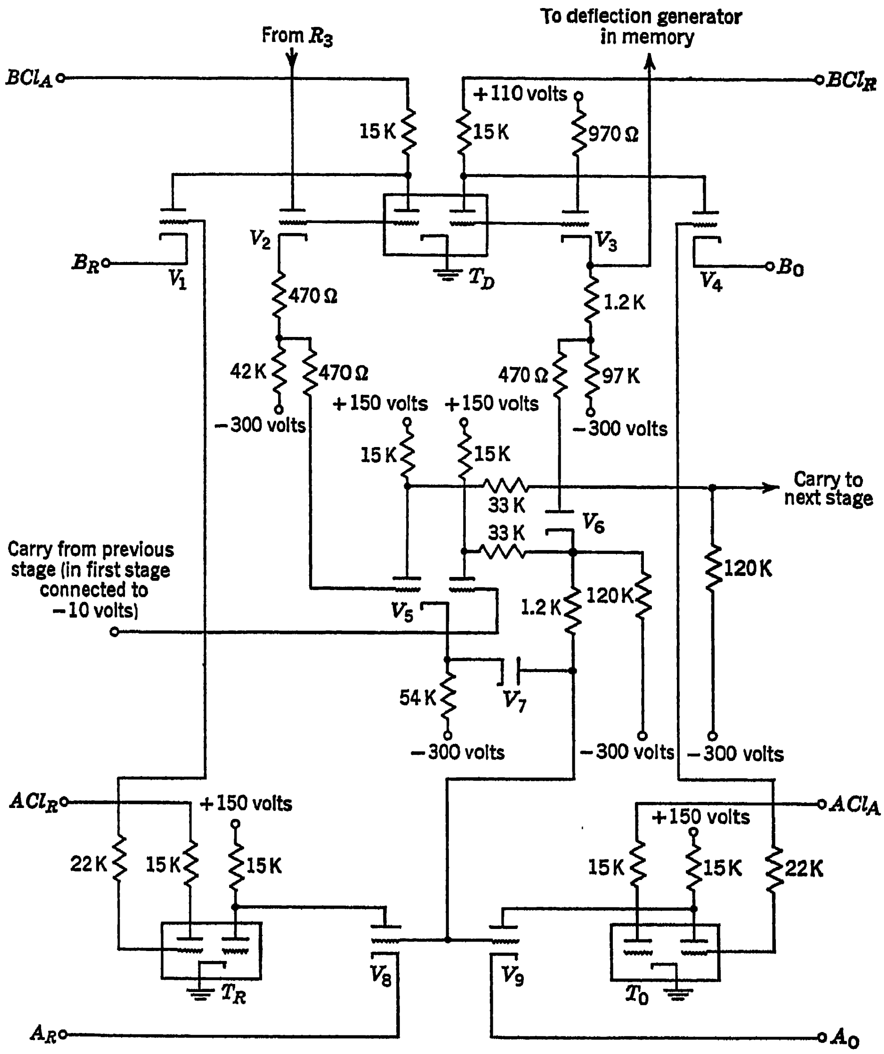


FIG. 9-9. Double-ranked adding counter (one column).

values will be derived shortly. The voltage representing the contents of T_D is 0 volts for 1, -40 for 0. There are, therefore, four cases to discuss.

Case I: The "carry in" is 0; T_D holds 0. The right grid of V_5 is positive, and the left is held at -40 volts. The cathode follows the right grid, and the left section is cut off; so the carry voltage to the next stage is set by the resistance network (between $+150$ and -300 volts) at $+21.4$ volts if all the resistors actually have their nominal values. The action of the diode V_6 causes the grids of gates V_8 and V_9 to be very slightly negative. Hence, when T_R is cleared to 1 and A_R is pulsed from +10 to −10 volts, the gate V_8 conducts and switches T_R back to 0. Thus 0 is left in T_R, and the carry to the next stage is 0, as it should be.

Case II: The carry in is 0; T_D holds 1. Again the left section of V_5 is cut off, and a +21.4-volt carry voltage is transmitted to the next stage. However, since the voltage of the right grid of T_D is −40, the diode V_6 is inoperative, and the grid voltage of V_8 is far negative (below −40 volts); so, when T_R is cleared to 1 and A_R is pulsed down to −10 volts, nothing happens. The net results are that T_R holds 1 and that no carry is propagated to the next stage.

Case III: The carry in is 1; T_D holds 0. Both grids of V_5 are negative. The diode network operates to hold the cathode at such a level that both sections of V_5 are cut off (except perhaps in the lowest-order stage, where the right half may draw some current, the left half being safely cut off), and so a 0 carry is transmitted to the next stage. The voltage of the grid of V_8 is low enough so that finally, when A_R is pulsed down to −10 volts, insufficient current is drawn to change the state of T_R, which continues to hold the 1 inserted into it by the clearing operation.

Case IV: The carry in is 1; T_D holds 1. As the left grid of V_5 is held at 0 volts and the right is negative (−10 volts in the first stage, about −40 in all others), the left half of V_5 conducts and the right half is cut off. Conduction through V_7 holds the grid of V_8 very close to 0 volts. The carry voltage is very nearly −40 volts (this justifies the value already used). Finally, when A_R is pulsed down to −10 volts, the gate V_8 causes T_R to switch to the 0 state.

All possible cases have been covered, and it has been shown that the circuit performs all the necessary functions correctly. As yet the input has not been discussed. Because of the special circumstance that this is always 1, its introduction is conveniently brought about by wiring the carry-in connection in the lowest-order stage permanently to a negative voltage. It is clear from the above analysis that the inputs to the second stage swing between the levels (0, −40 volts) and (+60, +110 volts). The second swing is exactly the same as the swing of the input, but the first is greater. However, −40 volts is enough above −50 volts so that there is considerable margin against malfunction.

9-9. Serial Counters. It is also possible to build counters very well adapted to use in serial computers essentially by using the recirculating register mentioned in Chap. 8 and making provision for the addition of 1 to its contents at suitable times.

A good example is given in Fig. 9-10, which illustrates the techniques developed at the National Bureau of Standards (NBS).[1] First suppose

[1] A. L. Leiner, W. A. Notz, J. L. Smith, and A. Weinberger, System Design of the SEAC and DYSEAC, *NBS Circ.* 551, Jan. 25, 1955.

that the circuit is quiescent. At time $t = 0$, let a pulse about 0.5 μsec in duration appear at I. This passes through M_1 and the tube-and-transformer combination, and enters loop 2 via gate G_2' and mixer M_2. Because G_1 receives no pulse fed back from O until $t = n + 1$, it is clear that no pulse circulates in loop 1, and the pulse just introduced circulates in loop 2, since, in the absence of further input pulses, gate G_2 is not inhibited. The pulse that circulates about loop 2 is available at O at $t = n, 2n, \ldots$. Now suppose that, at $t = n$, a second pulse arrives at I. As far as loop 1 is concerned, this pulse appears at G_1 at $t = n + 1$, and, as the pulse fed back from O appears there at the same time, a pulse is fed to M_1 at that time and passes on to appear at G_2'. The input pulse appears at G_2 and G_2' at $t = n$; it inhibits G_2 and finds G_2' inhibited by the pulse already resident in loop 2; so nothing enters, and the resident pulse is erased. Therefore, the pulse arriving at A at $t = n + 1$ finds the way

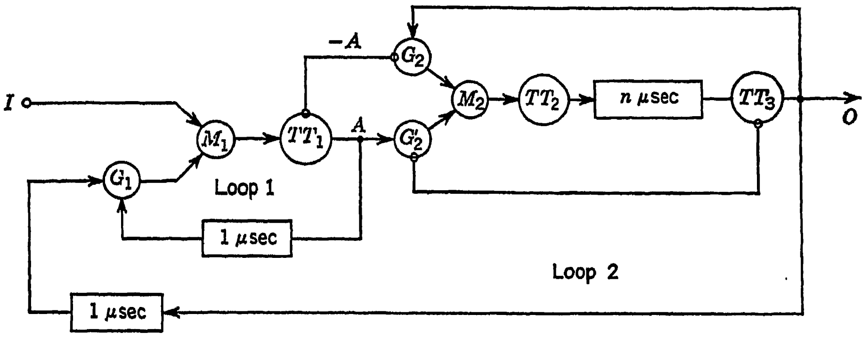


FIG. 9-10. NBS dynamic counter.

into loop 2 open. Hence the effect of the second input pulse at $t = n$ is that a pulse circulates in loop 2 which is available at O at $t = 2n + 1$, $3n + 1, \ldots$. As the time intervals $(kn, kn + 1)$, $(kn + 1, kn + 2)$, $\ldots$, $[kn + n - 1, (k + 1)n]$ can be interpreted as the $2^0, 2^1, \ldots ,$ 2^{n-1} positions of a binary integer, it is clear that the output after a single input represents 1 and that the output after the second input represents 2. Similar considerations show that, if a pulse appears at I at $t = 2n$, it is barred from loop 1 but enters loop 2, so that at 0 there are pulses in the intervals $(3n, 3n + 1)$ and $(3n + 1, 3n + 2)$, or in positions 2^0, 2^1, giving the binary representation of 3. Clearly, therefore, the circuit counts pulses that appear at I at $t = 0, n, 2n, \ldots$. The circuit is actually more versatile than has been seen as yet, for an argument similar to the above shows that, if the second pulse had been in the 2^1 position, the count would have advanced immediately to 3, and, in general, that, if the input pulse is in the 2^k position, the number in loop 2 is increased by 2^k.

Figure 9-11 shows an application of serial counting used in the University of Manchester megacycle computer (MEG). The circuit shown is called the "address-control loop," and is analogous to the IAS dispatch counter, being used both to regulate the regeneration of a Williams memory and to present to it the addresses to or from which information is to be transferred. The addresses are 10 bits in length, and the total delay around the loop is 20 μsec, since the action and regenerator cycles are

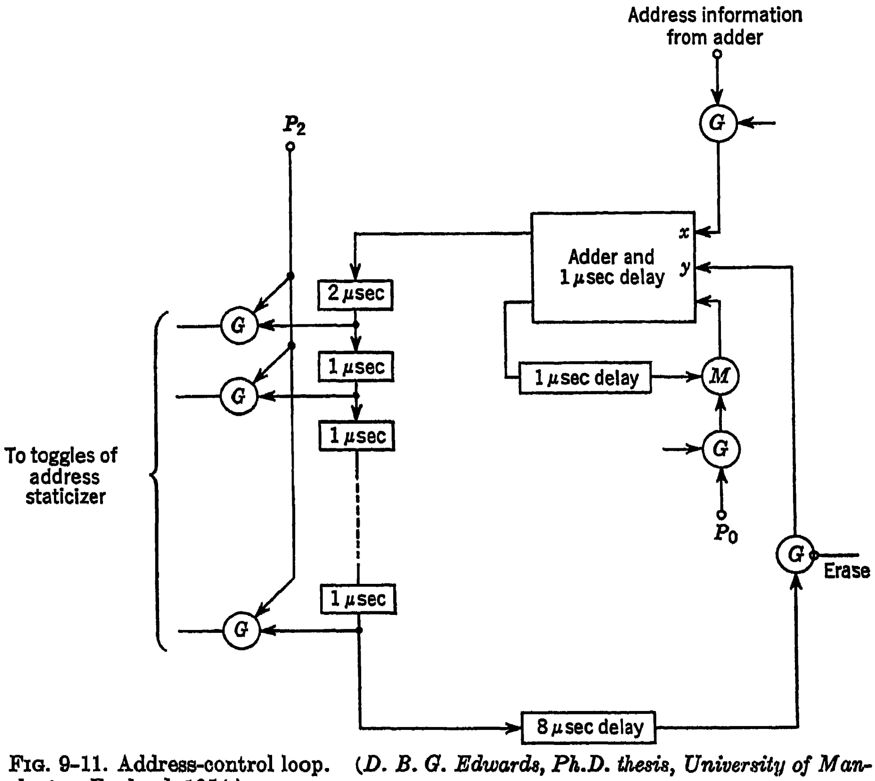


FIG. 9-11. Address-control loop. (*D. B. G. Edwards, Ph.D. thesis, University of Manchester, England,* 1954.)

each 10 μsec in duration. Besides the elements shown, the loop contains retiming and reshaping amplifiers, one for each 10 μsec of delay; these are omitted from the diagram as irrelevant to the logic.

There are two inputs to the adder, x and y; an action address is introduced at x, and the circulating number enters through y provided the erase gate is enabled. The carry is, as usual in serial adders, delayed 1 μsec and fed back; also, an artificial carry can be injected via a gate; this can be done only by the P_0 pulse, which is the first of the ten pulses of the machine's fundamental minor cycle, called a "beat."

Let us suppose that, initially, the loop contains no information and that a P_0 pulse is injected in the first beat. It emerges from the adder 1 μsec later and proceeds about the loop. Notice that it emerges from the last 1-μsec delay element during the P_2 time of the next (second) beat, so that, if the gates leading to the address staticizer are opened during that time, the staticizer records 1 in its lowest-order position. The pulse arrives back at y during the P_0 time of the third beat, so that, if the P_0 pulse of that beat is also injected as an artificial carry, the adder output is advanced to 2; that is, a pulse emerges from the next-to-the-last 1-μsec delay element during the P_2 time of the fourth beat and can be staticized as 2. Clearly, if this process is continued, the addresses 1, 2, 3, to 1023 are successively generated and are available to be staticized at 20-μsec intervals, i.e., every other beat, or, using the numbering introduced above, during beats 2, 4, 6, ... When 1023 has been generated, the regenerate address is reset to 0 simply by disabling the erase gate for the appropriate 10-μsec time interval.

An action address can be introduced via x beginning, e.g., with the P_0 of beat 2, and is available to the address staticizer during the P_2 time of beat 3. If there is no further need for this address, it can forthwith be erased by disabling the erase gate for the appropriate 10 μsec, or it can be permitted to circulate and be increased by unity by the injection of a P_0 pulse into the carry input of the adder at the beginning of beat 4. Actually, the 40-bit words are held in blocks of 10 bits in successive memory locations; so the normal procedure is to add 1 to the action address three times before erasing it.

9-10. Counting Tubes. There exist multielement tubes designed for counting in the decimal system. Although nothing more will be said about them in later chapters, it is worthwhile at least to give a brief introduction to the subject here. Little if any application has been made of these tubes in digital equipment except in one or two slow machines in England, where they have been used in building up decimal registers and decimal adders. The first tubes of this kind to come on the market were of the cold-cathode variety, which could count only at low rates, e.g., a few kilocycles per second. Later there appeared various thermionic tubes capable of much faster operation, up to the megacycle range. It should also be noted that the speed of the cold-cathode tubes has been increased and that in some developmental models a speed in the range from 50 to 100 kcps has been reached.

As cold-cathode tubes are cheap, rugged, reliable, and long-lasting, there seems to be no reason why, if fast-acting commercial models become available, they should not find very useful applications in decimal machines and perhaps also in special equipment associated with binary machines, e.g., in decimal-to-coded-decimal converters and in analogue-to-digital converters. Existing tubes are used a good deal in decimal scalers for counting nuclear events and for various counting applications in industrial control. The thermionic tubes are more expensive than the cold-cathode tubes but inherently capable of faster operation, which in many cases is of overriding importance.[1]

Cold-cathode tubes will now be considered in some detail. In the Dekatron (Ericsson Telephones Ltd.), there are the following tube elements:[2,3] (1) a central anode, (2) ten counter cathodes arranged in a ring around the anode, and (3) two sets of guide electrodes so arranged that one element of each set lies between successive counter cathodes. The operation of the tube depends upon the fact that the ionization of the gas near the glowing cathode lowers the striking potential to the adjacent guide elements. It is, therefore, possible to transfer conduction from a cathode to an adjacent guide electrode, from that guide to the next, and finally from the second guide to the next counting cathode. The counting rate in the earlier tubes was as low as 2,000 pulses per second, but this has been greatly improved. More recent Ericsson tubes are about ten times as fast.

Several other cold-cathode counting tubes have been developed in various places, for example, in the Bell Telephone laboratories[4] and in the laboratories of Sperry-Rand.[5] It was claimed that the original model of the latter (Remtron) was capable of a counting rate of 16,000 pulses per second.

The thermionic decade counter tubes are, as one might expect, inherently capable of faster counting rates. These tubes exist in considerable variety; consider first a tube developed in the Philips laboratories in Holland.[6] This contains as essential elements: (1) an electron gun which produces a ribbon-shaped electron beam, (2) a pair of deflecting plates, (3) a screen electrode containing ten slots through which electrons can pass, (4) an anode containing ten slots through which some of the electrons can pass to strike (5) a fluorescent screen, where they illuminate numbers 0 to 9 corresponding to the slots.

[1] For a review of both cold-cathode and thermionic tubes and of numerous applications of them, see K. Kandiah and D. W. Chambers, Multielectrode Counting Tubes, *J. Brit. IRE*, vol. 15, no. 4, pp. 221–232, April, 1955.

[2] J. H. L. McAuslen and K. J. Brimley, Polycathode Counter Tube Applications, *Electronics*, vol. 26, no. 11, pp. 138–141, November, 1953.

[3] R. C. Bacon and J. R. Pollard, The Dekatron: A New Cold Cathode Counting Tube, *Electronic Eng.*, vol. 22, p. 173, 1950.

[4] M. A. Townsend, Construction of Cold-cathode Counting or Stepping Tubes, *Elec. Eng.*, vol. 69, no. 9, pp. 810–813, September, 1950.

[5] J. J. Lamb and J. A. Brustman, Polycathode Glow Tube for Counters and Calculators, *Electronics*, vol. 22, no. 11, pp. 92–96, November, 1949.

[6] J. L. H. Jonker, A. J. W. M. van Overbeek, and P. H. de Beurs, A Decade Counter Value for High Counting Rates, *Philips Research Repts.*, vol. 7, pp. 81–111, April, 1952.

In operation the anode is connected directly to one of the deflection plates, the effect being that there exist just ten stable beam positions corresponding to the ten slots in the screen and that the beam can be stepped from one position to the next merely by applying a voltage pulse to the other deflection plate. Once the beam has been stepped across the screen, it is necessary, upon the tenth count, to switch it back to the 0 position. This requires some complication in the tube structure which will not be discussed here; it will suffice to remark that the "flyback time" is an essential limitation upon the counting rate, amounting to as much as 100 μsec if the simplest circuits are used but susceptible of considerable reduction with more complicated external circuits. Another limitation on the counting rate is the dual requirement—critical for reliability of operation—that the input stepping pulses have a rather slowly falling trailing edge and a steeply rising leading edge.

Other examples of decimal counter tubes are the trochotron[1] and a tube developed at the University of Sidney, Australia.[2]

Finally a number of beam-switching and counting tubes have been developed in the Burroughs laboratories.[3,4] These include, besides counter tubes, a selector and a coding tube. They are of the ribbon-beam type, some employing electrostatic deflection, others depending upon crossed electrical and magnetic fields. The selector and coder together produce the binary-coded equivalent of the decimal digit presented to the selector. The counters operate in the decimal system and are quite fast; the stepping pulses, although of considerable amplitude (100 volts in the earlier tubes), need be only a fraction (about 0.3) of a microsecond long, so that counting rates of a megacycle per second or more are readily obtainable.

[1] H. Alfven and L. Lindberg, Theory and Applications of Trochotrons, *Acta Polytech., Elec. Eng. Ser.*, vol. 2, no. 2, p. 106, 1949.

[2] D. L. Holloway, An Electron-beam Decimal Counter Tube, *Nature*, vol. 165, no. 4204, pp. 856–857, May 27, 1950.

[3] A. G. Fitzpatrick, A New Coding System for Pulse Code Modulation, paper presented at IRE annual convention, 1953.

[4] S. Kuchinsky, Multi-output Beam Switching Tubes for Computer and General Purpose Use, paper presented at IRE annual convention, 1953.

CHAPTER 10

ADDERS AND ACCUMULATORS

10-1. Introduction. The basic operation for all digital computation is addition. It is the only operation that is performed directly. Subtraction is conveniently performed merely by adding the complement of the subtrahend, products are built up by successive additions, and quotients are formed by successive additions and subtractions.[1-3]

This chapter will be concerned only with the circuits used in effecting addition; in Chap. 13 it will be shown how multiplication and division can be carried out. The basic circuit is the adder, which receives two inputs (the addends or summands) and yields one output (the sum). By combining an adder and a suitable register, it is possible to construct an accumulator, a device which forms the sum of the number already held ("resident number") and a new input ("incident number") and which then holds the sum until it either receives a new input or is deliberately cleared.

Adders and accumulators can be either serial or parallel in nature. A serial adder is a very simple circuit (see Sec. 10-2 below), but a parallel adder, although no more complicated logically, requires much more equipment, since separate and identical simple adders must be provided for all binary positions.

As far as the adders themselves are concerned, serial addition requires one minor cycle. For parallel addition the time is variable; for it depends upon the completion of all necessary carries. There is also serial-parallel addition, as mentioned in Secs. 3-5 and 3-7; the time required for this process is intermediate. However, in terms of actual machine operating time, it is necessary to take into account also the time required to procure the addends from the memory. In both serial and parallel adders, the addends are procured in sequence and added to the number residing in the accumulator, which is cleared to 0 before the reception of the first

[1] Staff, Harvard Computation Laboratory, "Synthesis of Electronic Computing and Control Circuits," chap. 12, Harvard University Press, Cambridge, Mass., 1951.

[2] Engineering Research Associates, "High-speed Computing Devices," chap. 13, McGraw-Hill Book Company, Inc., New York, 1950.

[3] R. K. Richards, "Arithmetic Operations in Digital Computers," D. Van Nostrand Company, Inc., Princeton, N.J., 1955.

addend. In single-address serial machines, for example, the addition of a number to the contents of the accumulator is completed one minor cycle after the moment that the first (or lowest-order) bit begins to emerge from the memory; if the memory is not of the random-access type, the variable access time must also be included. In the parallel single-address case, the carry-propagation time must be included as well as the access time.

At this point some numerical information is of interest; since access time can vary widely, this will be excluded from the discussion, which will be limited to the time required to add a newly appearing addend to the contents of an accumulator. Let the word length be 40 bits; then consider the following cases: (1) serial, with 1-Mcps pulse-repetition frequency (prf); (2) serial-parallel, ten serial groups of four bits, with 1-Mcps prf, as in MIDSAC; (3) parallel, e.g., as used in the Institute for Advanced Study (IAS) computer. The addition times in the first two cases are, clearly, 40 μsec and 10 μsec, respectively. In the existing IAS machine, a delay of 15 μsec is introduced to allow time for all possible carries to be performed and to provide a substantial factor of safety. This arrangement makes the process unnecessarily slow, since the average number of carries[1] in adding two 40-bit binary numbers is only 4.62. Recently a circuit has been devised[2] which determines when all carries are complete and, at that point, terminates the delay and permits the new sum to be recorded. This cuts the average time for adding a new number to the contents of the accumulator down to about 0.36 μsec—exclusive, of course, of access time.

With respect to addition alone this would be small advantage, for it would not greatly reduce the total time required for the addition process, which depends strongly on the access time to the memory, but it would lead to an average improvement of almost an order of magnitude in the speed of multiplication and division.

The fast-carry circuit just mentioned is described at the end of this chapter. It is somewhat complicated, requiring several tubes per stage. Another fast-carry circuit appropriate to the pulsed parallel accumulator of Whirlwind I will also be described in detail.[3] Both are examples of improvement in speed bought at the price of more sophisticated logic and more complicated circuits.

[1] A. W. Burks, H. H. Goldstine, and J. von Neumann, Preliminary Discussion of the Logical Design of an Electronic Computing Instrument, report on Army Ordnance contract W-36-034ord7481, Institute for Advanced Study, Princeton, N.J., 1946.

[2] B. Gilchrist, J. H. Pomerene, and S. Y. Wong, Fast Carry Logic for Digital Computers, report on Army Ordnance contract DA-36-034ord1646, Institute for Advanced Study, Princeton, N.J., 1956.

[3] Several schemes are described in Richards, *op. cit.*, pp. 100ff.

10-2. Half-adders and Adders Based on Them. The so-called "half-adder" is the simplest structure to understand, and it is possible to synthesize both serial and parallel adders from half-adders, as will be shown below.

Consider first the addition of two bits A and B. These give a sum S and a carry C, according to the following table:

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Using the notation of the algebra of logic, the following formulas are seen to be valid:

$$\begin{aligned} S &= (A \vee B) \wedge (A \wedge B)' \\ C &= A \wedge B \end{aligned} \tag{10-1}$$

These can be realized by the simple arrangement of gates and mixers shown in Fig. 10-1*a*, where an inhibitory input to one gate is used, or by the equivalent arrangement of Fig. 10-1*b*, where a separate inverter is

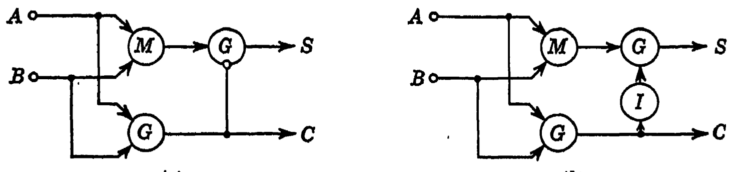


FIG. 10-1. Equivalent forms of half-adder I.

used. Other forms of the half-adder depend upon transformations of Eqs. (10-1) by the theorems of the algebra of logic given in Chap. 7. Thus, by Theorem 10, these can be thrown into the forms

$$\begin{aligned} S &= (A \vee B) \wedge (A' \vee B') \\ C &= (A' \vee B')' \end{aligned} \tag{10-2}$$

By two applications of Theorem 21, still another form can be found for S. One application yields

$$S = [(A \vee B) \wedge A'] \vee [(A \vee B) \wedge B'] \tag{10-3}$$

A second application (to both bracketed expressions) gives

$$S = [(A' \wedge A) \vee (A' \wedge B)] \vee [(B' \wedge B) \vee (B' \wedge A)] \tag{10-4}$$

Finally, since $A' \wedge A = 0$, this becomes

$$S = (A' \wedge B) \vee (A \wedge B') \tag{10-5}$$

A logical diagram appropriate to Eqs. (10-2) is shown in Fig. 10-2, and a diagram appropriate to Eq. (10-5) with the carry given by Eq. (10-1) is shown in Fig. 10-3.

It has been pointed out by Richards[1] that the half-adders just considered cannot conveniently be realized by circuits employing multielement vacuum-tube logic circuits, in which inputs are applied to control grids and outputs are taken from plates, because of the inversion automatically introduced by each stage. Richards points out that, if the sum and

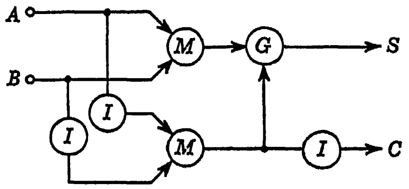


FIG. 10-2. Half-adder II.

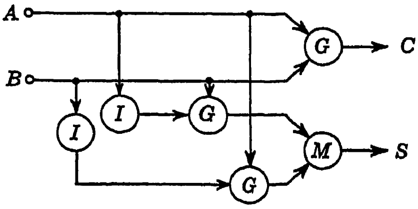


FIG. 10-3. Half-adder III.

carry formulas are thrown into what seems at first glance to be a rather eccentric form, that is,

$$\begin{aligned} S &= (A' \wedge B')' \wedge [(A')' \wedge (B')']' \\ C &= (A')' \wedge (B')' \end{aligned} \tag{10-6}$$

they can be realized by a circuit using eight triodes. Other circuits can be worked out employing triodes and tetrodes or pentodes.[2,3] Diode circuits however, are simpler and more convenient; so no examples will be given of the circuits just mentioned. The diode circuits of SEAC will be discussed fully below.

To make a complete serial adder, it is necessary to preserve in some way the carry arising from the addition of each pair of input bits presented to the circuit in pulse form and to add this carry to the sum of the next pair. Such a circuit can be built up of two half-adders and a delay element which delays the carry by one pulse period and thus provides the necessary memory. Let S_1 and C_1, respectively, signify the sum and carry that arise from the addition of two incoming bits before C_2, the delayed carry from the last pair, is added; and let S and C signify the sum and carry taking account of C_2. C becomes the C_2 for the next pair of bits to arrive

[1] *Op. cit.*, p. 87.
[2] *Ibid.*, pp. 87–88.
[3] "High-speed Computing Devices," p. 272.

at the input. The following table shows the possible cases, excluding $S_1 = C_1 = 1$, which is obviously impossible.

C_2	S_1	C_1	S	C
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
1	0	0	1	0
1	0	1	1	1
1	1	0	0	1

This table is conveniently summarized by the formulas

$$\begin{aligned} S &= (S_1 \vee C_2) \wedge (S_1 \wedge C_2)' \\ C &= C_1 \vee (S_1 \wedge C_2) \end{aligned} \qquad (10\text{-}7)$$

The first of these equations is formally identical with the first of the pair (10-1). To understand them properly, it is necessary to recall that S is formed by adding to S_1 the delayed carry C_2 from the addition of the preceding pair of bits and that C is the carry that is to be delayed and added

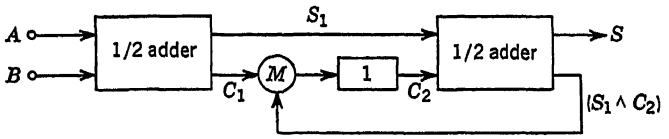


FIG. 10-4. Serial adder.

to the S_1 determined by the first half-adder for the next pair of bits; this C is 1 either if $C_1 = 1$ or if the carry $S_1 \wedge C_2$ out of the second half-adder is 1. With this understood, it is easy to draw the logical diagram of a serial adder, which appears in Fig. 10-4.[1,2]

A parallel adder can also be synthesized from half-adders; this, of course, requires two half-adders for each binary position, and the carry out of any binary position must be added to the sum of the bits in the position of next higher significance. Such an arrangement for one binary position is shown in Fig. 10-5.[3] The two numbers to be added must be presented simultaneously to the circuit not as pulses but as voltages that remain fixed for a time longer than the time required for all carries to be completed, and the output must not be used until this time has elapsed.

10-3. The Full Adder. A different approach begins by considering the addition of two bits A and B and the carry C from the addition of the

[1] Richards, *op. cit.*, p. 130.
[2] "High-speed Computing Devices," p. 274.
[3] Richards, *op. cit.*, p. 84.

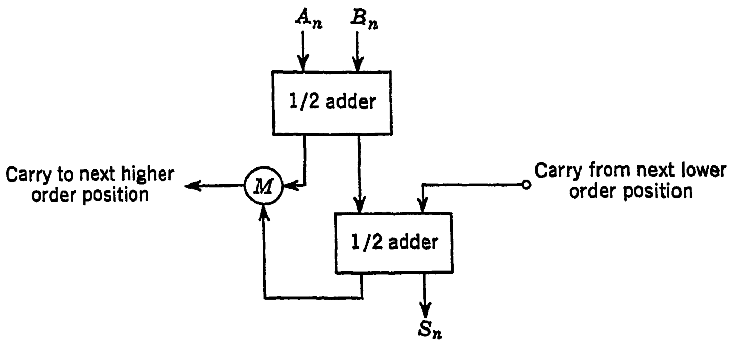

A_n
B_n
1/2 adder
Carry to next higher order position
M
Carry from next lower order position
1/2 adder
S_n

FIG. 10-5. Parallel adder.

pair of bits in the position of next lower significance. This is governed by the accompanying table, where C_1 is the carry to the position of next-higher significance.

	A	B	C	S	C_1
(1)	0	0	0	0	0
(2)	1	0	0	1	0
(3)	0	1	0	1	0
(4)	1	1	0	0	1
(5)	0	0	1	1	0
(6)	1	0	1	0	1
(7)	0	1	1	0	1
(8)	1	1	1	1	1

The tabular information is readily summarized in the logical formulas

$$S = (A \wedge B' \wedge C') \vee (A' \wedge B \wedge C') \vee (A' \wedge B' \wedge C) \vee (A \wedge B \wedge C) \qquad (10\text{-}8)$$

$$C_1 = (A \wedge B \wedge C') \vee (A \wedge B' \wedge C) \vee (A' \wedge B \wedge C) \vee (A \wedge B \wedge C) \qquad (10\text{-}9)$$

In the last expression, the union of the first and last terms yields

$$(A \wedge B) \wedge (C' \vee C) = A \wedge B \qquad (10\text{-}10)$$

Similarly, the union of the second and last terms yields $A \wedge C$, and the union of the third and last terms yields $B \wedge C$. Hence, evidently, C_1 can be written in the simplified form

$$C_1 = (A \wedge B) \vee (A \wedge C) \vee (B \wedge C) \qquad (10\text{-}11)$$

which can be realized without the use of any inversions. The expression for S can be thrown into a variety of forms, which do not have the appearance of being markedly simpler; indeed, some look much more complicated, as, for example,

$$S = ((A \vee B \vee C) \wedge ((A \wedge B) \vee (A \wedge C) \vee (B \wedge C))') \vee (A \wedge B \wedge C) \qquad (10\text{-}12)$$

which, however, like (10-11), possesses the advantage that no inversion of inputs is required.[1]

A device that realizes these formulas is called a "full adder." A number of examples of such structures could be given, but attention will be concentrated on the full adder used in SEAC. Observe first that full adders can be used for both serial and parallel addition. In the serial case, it is necessary to delay the carry C_1 by one pulse period and to reinject it as C; in the parallel case there must be one full adder per binary position with carry from column to column, and the simultaneously applied inputs must be of sufficiently long duration to permit all carries to be completed.

10-4. NBS Full Adder. The National Bureau of Standards (NBS) circuit[2] is given in Fig. 10-6, where the reclocking has been omitted as irrelevant to the logic. The mixers and gates are of the simple diode type, and only two tube-transformer combinations are required. The inputs $-A$ and $-B$ are not inverted in the logical sense (that is, they do not signify A' and B'), but only in the electrical sense. Recall from the discussion of the basic SEAC circuits in Chap. 6 that the pulses representing 1's are positive, but that, since the output from each tube is taken from a transformer whose primary is in the plate circuit and which carries two secondaries, a negative pulse is also available every time a 1 is transmitted. The availability of these pulses makes logical inversion unnecessary, for they can be used very simply to inhibit a gate. For example, the gate at the top of the drawing yields an output if $A = 1$, $B = 0$, $C = 0$, but is inhibited if either $B = 1$ or $C = 1$, or, of course, if $B = C = 1$; it therefore represents $A \wedge B' \wedge C'$.

The logical elements M_1, M_2, G_1, G_2 generate the carry from A, B and the carry resulting from the addition of the previous (lower-order) pair of bits. From G_1 is obtained $A \wedge B$; from M_1, $A \vee B$; from G_2,

$$(A \vee B) \wedge C = (A \wedge C) \vee (B \wedge C)$$

so from M_2 results $(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$, which is the new carry as given in Eq. (10-11). Each carry is delayed 1 μsec (i.e., one pulse period) by the delay lines shown in the drawing, so that it may be used in the addition of the next pair of bits.

[1] This is treated fully by Richards, *op. cit.*, pp. 89–92, who gives several other forms of S.

[2] A. L. Leiner, W. A. Notz, J. L. Smith, and A. Weinberger, System Design of SEAC and DYSEAC, *NBS Circ.* 551, p. 77, Jan. 25, 1955.

The gates G_3, G_4, G_5, G_6 generate $(A \wedge B' \wedge C')$, $(A' \wedge B \wedge C')$, $(A' \wedge B' \wedge C)$, and $(A \wedge B \wedge C)$, which are combined in M_3 to give the sum as written in Eq. (10-8), and this output is amplified by a tube-transformer combination. At S and $-S$ positive and negative pulses, respectively, appear for 1, and no pulse for 0.

The simplicity of the SEAC full adder is striking in comparison with full adders using only multielement tubes. None of these will be described here, but several are given by Richards.[1] The smallest number of tubes in the circuits cited is eight, of which the majority are tetrodes or pentodes (seven tetrodes to one triode in one case, four pentodes to

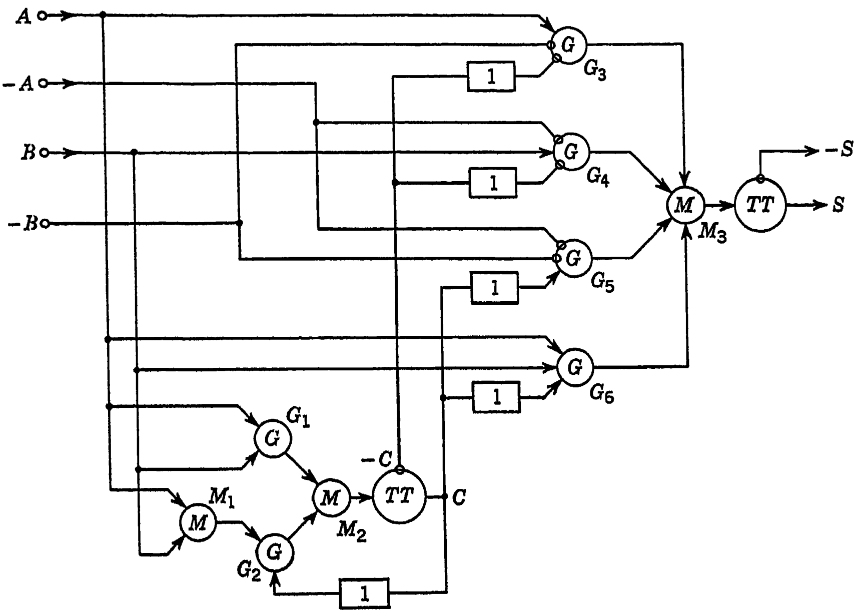


FIG. 10-6. NBS full adder.

three triodes in the other), but those particular circuits require the availability of logically inverted inputs (that is, A' as well as A). The circuits that do not require inverted inputs employ 11 tubes each. Adders synthesized from multielement-tube half-adders are even more wasteful of tubes.

10-5. A Serial Accumulator. It is easy to see how to make a serial accumulator using the adder of Fig. 10-6. A detailed drawing will not be given, but in principle all that must be done is to feed the output S back through a delay of one minor cycle (one word length) and reinject it at one input, say B (and, of course, $-B$). Evidently any number

[1] Richards, *op. cit.*, pp. 94–95.

appearing at A (and $-A$) at the proper time is then added to the number already circulating about the loop. A simple gate must also be inserted in the loop; this gate, when disabled or inhibited, interrupts the path and, therefore, causes the accumulator to be cleared to 0.

10-6. Parallel Binary Adders and Accumulators. There are two main classes of binary parallel adders, distinguished according to the way in which the addends are presented to the circuit.

One has a single set of parallel leads over which the set of pulses representing the first addend is transmitted; these pulses are followed by the set of pulses representing the second addend. Evidently this adder must in itself be an accumulator. It is appropriate to synchronous operation.

The other class has two sets of parallel input leads by means of which the two sets of signals representing the two numbers to be added must be presented to the circuit simultaneously. The adder output must of course be transmitted to a register if it is to be preserved after the inputs have been removed. If means are provided of transmitting this sum back to the register that holds one addend, an accumulator results.

The adjectives "sequential" and "simultaneous" describe the types of input in the two cases well enough. Both types will be considered.

Consider the sequential type. It consists essentially of a set of bistable elements, one for each digit column plus auxiliary ones. First these elements are cleared; then the bits of the first summand (addend) are read into them. The next step is to read in the second summand. The bistable elements in the columns are modulo 2 counters. Hence if in, say, the kth column only 0's were received, 0 would be held; if a 0 and a 1 (in either order) had been received, 1 would be held, but if two 1's had been received, 0 would be held. In the last case the bistable element has been switched twice, and this fact must be recorded, for it indicates a carry into the next-higher-order column. A set of auxiliary bistable elements, therefore, is needed to remember the carries. The naive way to proceed would be to add in the set of carries, producing a new sum and a new carry, and so on. Various schemes exist, however, for performing all the carries essentially simultaneously, in both sequential and simultaneous types of parallel adder. These will be described below.

A sequential parallel adder will be described in Sec. 10-7, a simultaneous-input parallel adder in Sec. 10-8.

10-7. The Whirlwind Accumulator. The Whirlwind accumulator is a shifting register furnished with the equipment needed to perform addition. The general scheme by which shifting is accomplished in the Whirlwind registers has already been described. The shifting circuits may, then, be ignored (and, of course, also the pulse-restorer circuits, which are of no logical importance), in order to describe the structure and operation of the accumulator as simply as possible.

Figure 10-7 shows the basic structure: two rows of toggles interconnected with gates. The delay denoted by the box marked "delay" (D) is in fact actually provided by the response time of the input toggle into which D is shown inserted; it is not a separate and identifiable element. Of course, in the actual accumulator there exist a great number of other gates to take care of shifting, "shift and carry," etc., but these need not be considered until later on; similarly, "clear" and pulse-restorer lines need not be shown here.

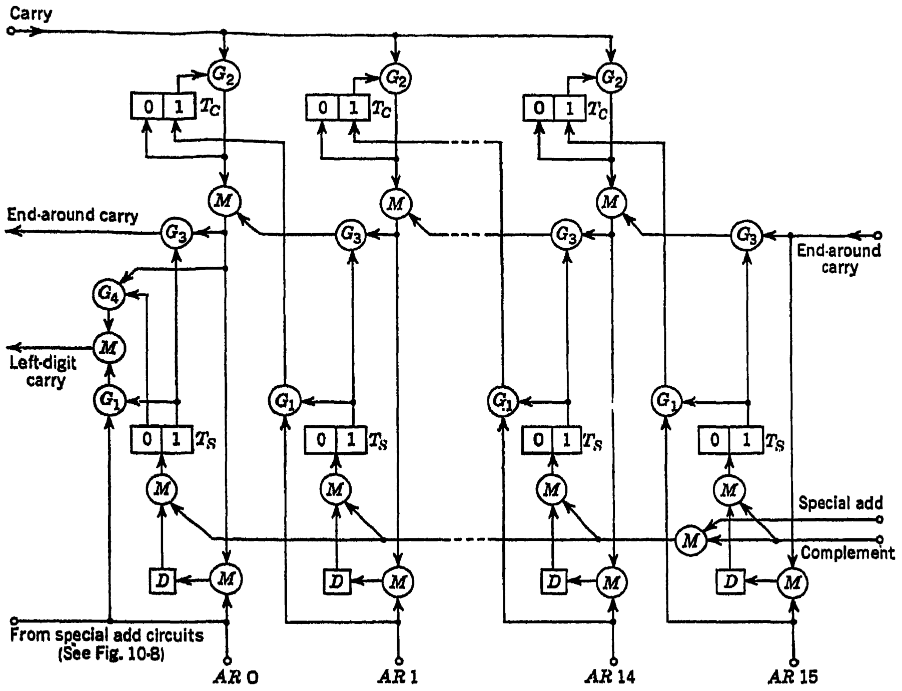


FIG. 10-7. High-speed carry in the Whirlwind I accumulator.

Suppose that, initially, all toggles have been cleared to 0, which is taken conventionally as the state in which the left tube is conducting. The problem is to form the sum of two numbers both less than unity, assuming that the sum also is less than unity. The first step is to read in the first summand from the arithmetic register (the input leads being shown as AR 0, etc.). The second step is to read in the second summand. Observe that the inputs to the "sum" toggles T_S are delayed, and so the input pulses arrive at gates G_1 before the toggles T_S change state. Hence, if a toggle holds a 0 as the result of the first read-in, a second 0 will do nothing at all, whereas a 1 will change the state of the toggle, but will not be transmitted through G_1; G_1 is disabled because the plate of the tube in T_S to which it is connected is at low voltage since it is the plate of the conducting section. Similarly, note that G_1 is enabled if T_S receives a 1 in the first read-in operation. If this is followed by a 0, of course nothing happens, but, if a second 1 is read in, two events occur: (1) the pulse representing the 1 is transmitted (reversed in polarity) by G_1, and switches the carry toggle of the stage to the left to the 1 state; and (2) T_S is switched back to 0. Finally, a positive carry pulse is applied to the gates G_2. Whenever a gate is enabled because its controlling T_C toggle holds a 1, the carry pulse is transmitted by the gate. The carry pulse is then applied (1) as a pulse to clear T_C to 0, (2) as a pulse to change the state of T_S, and (3) as an input to the high-speed-carry gate G_3. Of course, it is

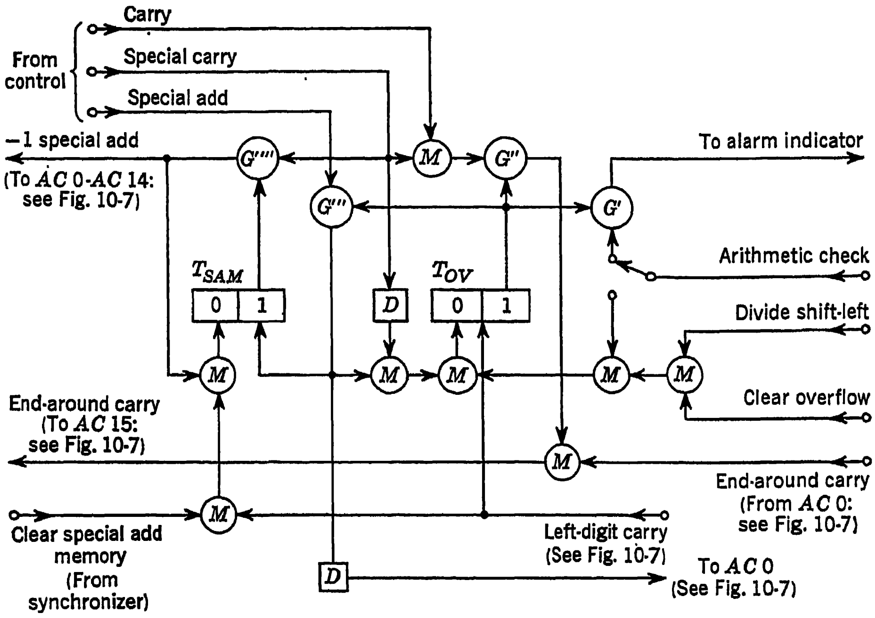

From control
Carry
Special carry
Special add
−1 special add
(To AC 0-AC 14: see Fig. 10-7)
G''''
G'''
M
G''
G'
To alarm indicator
Arithmetic check
T_SAM
0
1
D
T_OV
0
1
Divide shift-left
M
Clear overflow
End-around carry (To AC 15: see Fig. 10-7)
End-around carry (From AC 0: see Fig. 10-7)
Clear special add memory (From synchronizer)
Left-digit carry (See Fig. 10-7)
D
To AC 0 (See Fig. 10-7)

FIG. 10-8. Overflow and special-add features of Whirlwind I accumulator.

clear that a pulse inverter must be inserted between the output of G_2 and the input of G_3, for positive pulses are required both as inputs to gates and as complement inputs to the cathode circuit of the cathode-coupled toggle, but these need not be shown in the logical diagram. The switching of T_S is delayed somewhat. Meanwhile, if T_S holds a 1, G_3 is enabled, and the carry pulse is transmitted to the next stage to the left. It is clear that a gate G_3 can receive an input from G_2 or from the G_3 of the next column to the right but not from both, for if T_C holds a 1 when the carry line is pulsed, this can only be because, in forming the partial sum, the T_S of the next stage to the right was switched back to 0 after the pulse that switched T_C to 1 passed the G_3 gate. Hence if, in the partial sum, there exists a string of 1's in the toggles T_S, but, in the rightmost of these columns, T_C holds a 1, then a carry is propagated to the left, all the 1's turn to 0, and the first 0 encountered is turned to 1. In this manner, all carries are added in to the partial sum in one operation.

Now turn to Fig. 10-8, which shows certain auxiliary equipment: two toggles, the "special-add memory" and the "overflow," together with a number of gates. The functions performed here are as follows: to remember an overflow from AC 0 and either (1) give an alarm, or (2) use this overflow to adjust the sum of the more significant parts, if double-length arithmetic is being performed. These two cases will be treated in order.

Start by noting that T_{OV} acts, as far as carry from the 0th accumulator column is concerned, like a carry toggle in a next-left column. Of course, T_{OV} and T_{SA} are both cleared to zero before the addition takes place. First assume that an ordinary addition of single-length words is to occur; typical situations to illustrate the operation of the circuit will be enumerated.

1. Both summands are positive, and their sum is less than unity; no pulse is transmitted to the "left-digit-carry" line, and the circuits of Fig. 10-8 remain quiescent.

2. The summands are of opposite sign, the positive one larger in magnitude (negatives are represented by 1's complements).

$$\begin{array}{ll} A & 0.1010 \\ B & \underline{1.1100} \\ C & 1 \\ S & \underline{1.0110} \\ & 0.0111 \end{array} \qquad (10\text{-}13)$$

It is clear that, if A is added into the accumulator first and is followed by B, the 1 in the first column of B encounters a closed gate G_1 and so does not get through. When, however, the carry pulse comes down, it finds G_4 closed. But G_3 is open, and the carry pulse is forced to perform the end-around-carry function. A similar argument holds if the order of the transmission of the summands to the accumulator is reversed.

3. A and B are of opposite sign; the negative one is larger in magnitude. Again there is no transmission to the circuits of Fig. 10-8, and, of course, no end-around carry either.

4. Both summands are negative, and the magnitude of the sum is less than unity. Clearly, when the second summand is read into the accumulator, the 1 transmitted to the 0 position passes through G_1, switches the overflow toggle T_{OV} to 1, switches T_S in AR 0 to 0, and thus opens G_4. Now A and B are negative and $-A - B < 1$. Using 1's complements, negative A is represented by

$$2 + A - 2^{-15} = 1 + 1 + A - 2^{-15} \qquad (10\text{-}14)$$

In forming the sum, the 1's in the 0 position set T_{OV} to 1 and clear AC 0 back to 0, as noted. Consider the sum of the remaining places; it is

$$S' = 1 + A - 2^{-15} + 1 + B - 2^{-15} = 2 + A + B - 2^{-15} - 2^{-15} \tag{10-15}$$

But, by hypothesis,

$$-A - B < 1 \tag{10-16}$$

which means that

$$-A - B \leq 1 - 2^{-15} \tag{10-17}$$

whence

$$S' \geq 1 - 2^{-15} \tag{10-18}$$

In only one case is S' less than unity, and in that case the T_S toggles in stages 1 to 15 all hold unity before the arrival of the carry pulse. The carry pulse passes through the open gate G'' of Fig. 10-8; becomes the end-around-carry pulse; encountering open gates G_3 in stages 1 to 15 and G_4 in stage 0, sets stages 1 to 15 to 0 and stage 0 to 1; and proceeds also to switch T_{OV} back to zero, thus precluding any alarm. In general, S' exceeds unity; so, since T_S of stage 0 must hold 0 before the carry pulse arrives, it is clear that a carry pulse must arrive by some route to switch T_S of stage 0 to 1. It also passes G_4 and switches T_{OV} back to 0, precluding an alarm. T_{OV} has meanwhile held G'' open long enough to pass the carry pulse and so to provide the required end-around-carry pulse. .

The examples given in the preceding paragraphs show that no alarm is ever given in any case in which the sum is less than unity in magnitude. Consider now the cases in which the contrary holds.

5. Let both summands be positive (of course both are less than unity) and let their sum exceed unity. It is clear that the reading in of the summands causes no action in AC 0, except that the carry toggle must be switched to 1. When the carry pulse arrives, it is transmitted down the 0 column, finds G_4 open, hence is transmitted to switch T_{OV} to 1, and, of course, switches T_S to 1. Thus the process ends with the alarm gate G' open, and the subsequent check pulse is transmitted to the alarm indicator.

6. Finally, consider the case of two negative summands whose sum is less than -1. Clearly, successive reading in of the summands must result in switching T_{OV} to 1, for both summands have 1's in the 0 position. The sum of the portions of the summands in columns 1 to 15 is

$$\begin{aligned} S' &= (1 + A - 2^{-15}) + (1 + B - 2^{-15}) \\ &= 2 + (A + B) - 2^{-14} < 1 - 2^{-14} \end{aligned} \tag{10-19}$$

Hence it is assured that the carry gate of column 0 is not open and that no pulse is transmitted to G_4 by the high-speed-carry line, for that would result in the switching of T_S in column 0 to 1, which has been shown in Eq. (10-19) to be impossible. Hence in this case, too, the alarm gate is left open to transmit the alarm pulse to the alarm indicator.

It has been shown that the overflow alarm in fact operates as it should. Consider then the function of the "special-add-memory" toggle T_{SAM}. This is used only when double-precision addition is performed. Of course, in a double-precision representation, it would be preferable always to consider both parts of the number as having the same sign. Suppose our word is but 3 bits long. Then the number 0.0111 could be represented by

$$\begin{array}{cc} \textit{Register } 1 & \textit{Register } 2 \\ 0.01 & 0.11 \end{array} \tag{10-20}$$

where a scale factor of 2^{-2} is associated with the second register, so that its value is interpreted as 0.0011. On the other hand, 0.0111 could also be represented as follows:

$$\begin{array}{cc} \textit{Register } 1 & \textit{Register } 2 \\ 0.10 & 1.10 \end{array} \tag{10-21}$$

This may appear at first glance to be a peculiar way to represent the given number, but it is easy to see that it arises quite naturally. For example, suppose it is desired to form the sum

$$\begin{array}{r} 0.1010 \\ \underline{1.1100} \\ 0.0111 \end{array} \tag{10-22}$$

The negative number would be represented as

$$\begin{array}{cc} \textit{Register } 1 & \textit{Register } 2 \\ 1.11 & 1.00 \end{array} \tag{10-23}$$

if both parts were to have the same sign. The pair of sums

$$\begin{array}{rr} 0.10 & 0.10 \\ \underline{1.11} & \underline{1.00} \\ 0.10 & 1.10 \end{array} \tag{10-24}$$

would then be formed, corresponding to

$$0.1000 - 0.0001 = 0.1000 + 1.1110 = 0.0111 \tag{10-25}$$

Hence it is clear that the signs of the two sections of a double-precision number found as a result of computation cannot be expected to agree.

Consider also the case

Most significant part	*Least significant part*	
0.00	0.11	
0.00	0.11	
0.00	1.10	(10-26)

From the analysis of the action of T_{OV}, it is clear that, in forming the sum of the least significant parts, T_{OV} is set to 1. The correct value of the sum is 0.0110, which is not given by (10-26); the overflow must be taken into account to obtain the correct result.

To take care of situations of the kind just described, a "special-add" operation is provided for the addition of the least significant parts of double-precision numbers. This operation precedes the addition of the most significant parts, and the special-add order assumes that overflow is properly used to adjust the sum of the most significant parts.

The logical diagram of the equipment used to effect the special addition is shown in Fig. 10-8. The action of the overflow toggle has already been explained. The other toggle, T_{SAM}, is called the "special-add-memory" toggle. Note also the special-add input, which is a pulse replacing the arithmetic-check pulse of ordinary addition, and the special-carry input, which occurs at the beginning of each clear-and-add instruction.

Recalling the six cases discussed in describing the operation of the alarm circuit, notice that, in the first four, when "proper" addition occurs, the circuit ends up with the special-add gate G''' closed, so nothing further happens. Consider the fifth case, that of positive overflow. It has been shown that this results in T_{OV} holding 1 and in T_S of stage 0 also holding 1. Thus the special-add gate G''' is open, and the subsequent pulse transmitted by it (*a*) switches T_{SAM} to 1, (*b*) switches T_{OV} to 0, (*c*) after a delay (set at 0.5 μsec) switches T_S in stage 1 to 0 but first passes through G_1 of *AC* 0, switches T_{OV} back to 1, and switches T_{SAM} back to 0. The contents of the accumulator may now be sent to the memory. The next arithmetic order must be a "clear and add." One of the control pulses of this instruction is the special carry, which passes G'' and thus comes to the end-around-carry line, increasing the contents of *AC* 15 by unity. Thus the carry from the addition of the least significant parts to that of the most significant parts is accomplished. Note that after a delay (set at 0.1 μsec in Whirlwind I) the special-carry pulse also switches T_{OV} to 0.

The case of negative overflow is similarly analyzed. It has been shown that the result of the addition is 1 in T_{OV} and 0 in T_S of *AC* 0. The special-add pulse sets T_{SAM} to 1 and T_{OV} to 0, and, after a delay, T_S of *AC* 0 to 1; of course in this case it does not get through G_1 of *AC* 0. Thus the sum receives the correct sign. The special-carry pulse of the next clear-and-add (or subtract) operation passes through G'''' and complements the sum toggles of AC 0 to AC 14. As the special-carry pulse occurs right after the clearing of the accumulator, this means that -2^{-15} is placed in it before the read-in of the first of the two most significant summands, so it properly corrects their sum.

In Chap. 1 it was seen that the process of building up a product was speeded considerably if the shift-and-carry feature was provided. The structure needed to carry out this operation (for right shifts only) and the ordinary left shifts needed in forming a quotient is shown in Fig. 10-9, where, for the sake of clarity, the carry gates shown in Fig. 10-7 have been omitted.

First consider the left shift, which involves gates G_{11} and G_{12} and which is ordered by a positive pulse on either the shift-left or the divide-shift-left line. Note that the first of these shifts left the contents of AC 2 to AC 15. Thus the sign bit (in AC 0) is preserved, but the bit in AC 1 is lost. In division, on the other hand, the entire contents of the accumulator must be shifted left (see the discussion of division in Chap. 1). Consider the complete left shift, and let, for example, T_S of AC 15 hold 1. Then G_{11} is open; the divide-shift-left pulse passes through it and, after a slight delay, sets T_S of AC 14 to 1; G_{12} is involved instead if 0 is shifted left. The delay in switching T_S of AC 14 is necessary in order to permit the shift-left pulse to pass through the gates in column 14 before T_C is switched. The connections into AC 15 permit the contents of a second register to be shifted into the accumulator, and the outputs leading to the left from the first column are used in the control of the division process.

The shift-and-carry operation has been discussed in Chap. 1; for details presented in tabular form, the reader may refer to Table 1-52. Suppose first that T_C of, say, AC 1 holds 1. Then G_6 and G_8 of column 1 are open. Hence, if T_S holds 1, the shift-and-carry pulse passes up through G_9 and then through G_6 and so switches T_S of AC 2 to 0, whereas, if T_S of AC 1 holds 0, the shift pulse passes through G_{10} and G_8 to switch T_S of AC 2 to 1. The cases in which T_C holds 0 can be similarly discussed; it will be seen that the requirements set forth in Table 1-52 are all satisfied. Note that ordinary right shift follows the rules for the 0-carry cases. Observe that 0 is always shifted into AC 1 in this process. This is because, in Whirlwind I, multiplication is always performed on positive quantities, the correct sign being separately formed and subsequently affixed. The outputs to the right go to the B register, which ultimately holds the least significant part of the product.

10-8. A Simultaneous-input Parallel Adder. In the Whirlwind I accumulator just described in some detail, the successive numbers are read into the device sequentially: first the set of pulses representing the

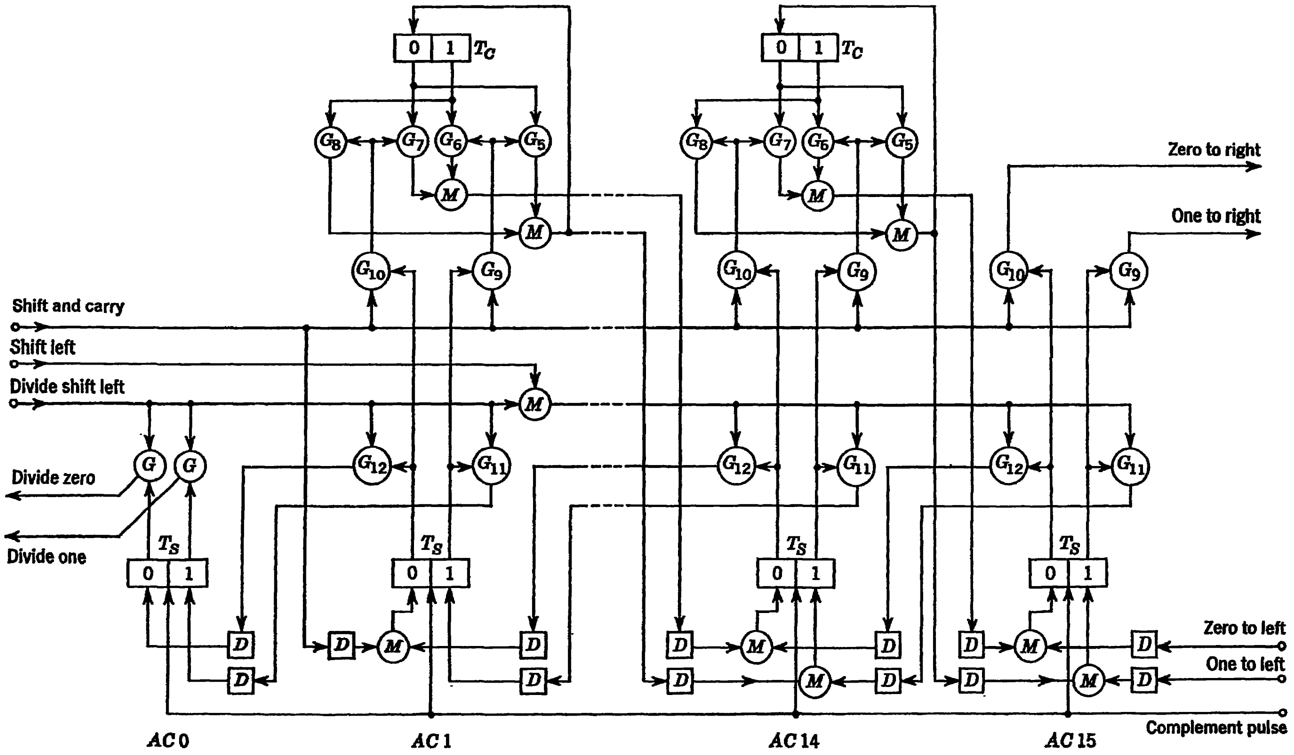


FIG. 10-9. Shifting features of Whirlwind I accumulator.

first summand, then the set representing the second, and so on. This section will deal with an adder to which two inputs are simultaneously presented as the voltages of two sets of wires and which is, therefore, well suited to asynchronous operation. The adder continuously forms the sum of the two numbers represented by the two sets of voltages, carries are automatically taken care of, and the sum appears as the set of voltages of a set of output wires. The scheme presented here was first used in a practical computer in the IAS machine, and, although it is now rather out of date, it is still interesting, if only for historical reasons. It will be seen that the circuit depends upon the addition of standard units of current in a resistor, the voltage developed across it representing the sum. Actually, this voltage assumes one of four possible levels, and a circuit called the "digit resolver" is required to interpret it in terms of the standard representation of numbers in the machine.

In Fig. 10-10 are shown two adjacent columns of the adder.[1] The least significant stage is to be thought of as at the extreme left. The nth stage consists of tubes $T_{1,n}$, $T_{2,n}$, the left sections of $T_{3,n}$ and $T_{3,n-1}$, and $T_{4,n}$. To this stage there are three inputs: (1) the carry from the preceding column; (2) the resident digit A_n from the accumulator register; and (3) the incident digit B_n from the memory via a set of complement gates, which allow either the number from the memory or its complement to be read in to the adder. These wires assume a potential of 0 volts to represent binary 0, and -40 volts to represent binary 1. The outputs are two: to the digit resolver (and thence to the accumulator register RI, as will be seen below) and to the memory.

Start by assuming that the carry input to column n has been disconnected, so that the left grid of $T_{4,n}$ is held at $+211$ volts. The cathode, because of the flow of current through the cathode resistor, rises to a slightly more positive voltage than this, and the diode-connected right section of the tube is nonconducting. Thus, by cathode-follower action, the cathode voltage of the left section of $T_{3,n}$ is established at slightly above $+211$ volts—actually $+215$, allowing for cathode rise.

Now consider the resident-digit gate $T_{2,n}$ and the incident-digit gate (left section of $T_{3,n-1}$). If both these digits are 0, then the voltages of wires A_n and B_n are both 0, and neither the right section of $T_{2,n}$ nor the left section of $T_{3,n-1}$ draws any current; no current flows through the summing resistor R_S, and the voltage at the summing point S is $+215$.

[1] H. H. Goldstine, J. H. Pomerene, and C. V. L. Smith, Final Progress Report on the Physical Realization of an Electronic Computing Instrument, pp. 62a–69, Princeton, The Institute for Advanced Study, January, 1954, a report prepared under U.S. Army Ordnance Contracts W-36-034-ORD-7481 and DA-36-034-ORD-19. Relevant resistor values are R_S = 10.5 kilohms, R_C = 18 kilohms, cathode resistor of $T_{1,n}$ = 90 kilohms, cathode resistor of $T'_{2,n}$ = 32 kilohms, cathode resistors of $T_{3,n-1}$ = 32 kilohms.

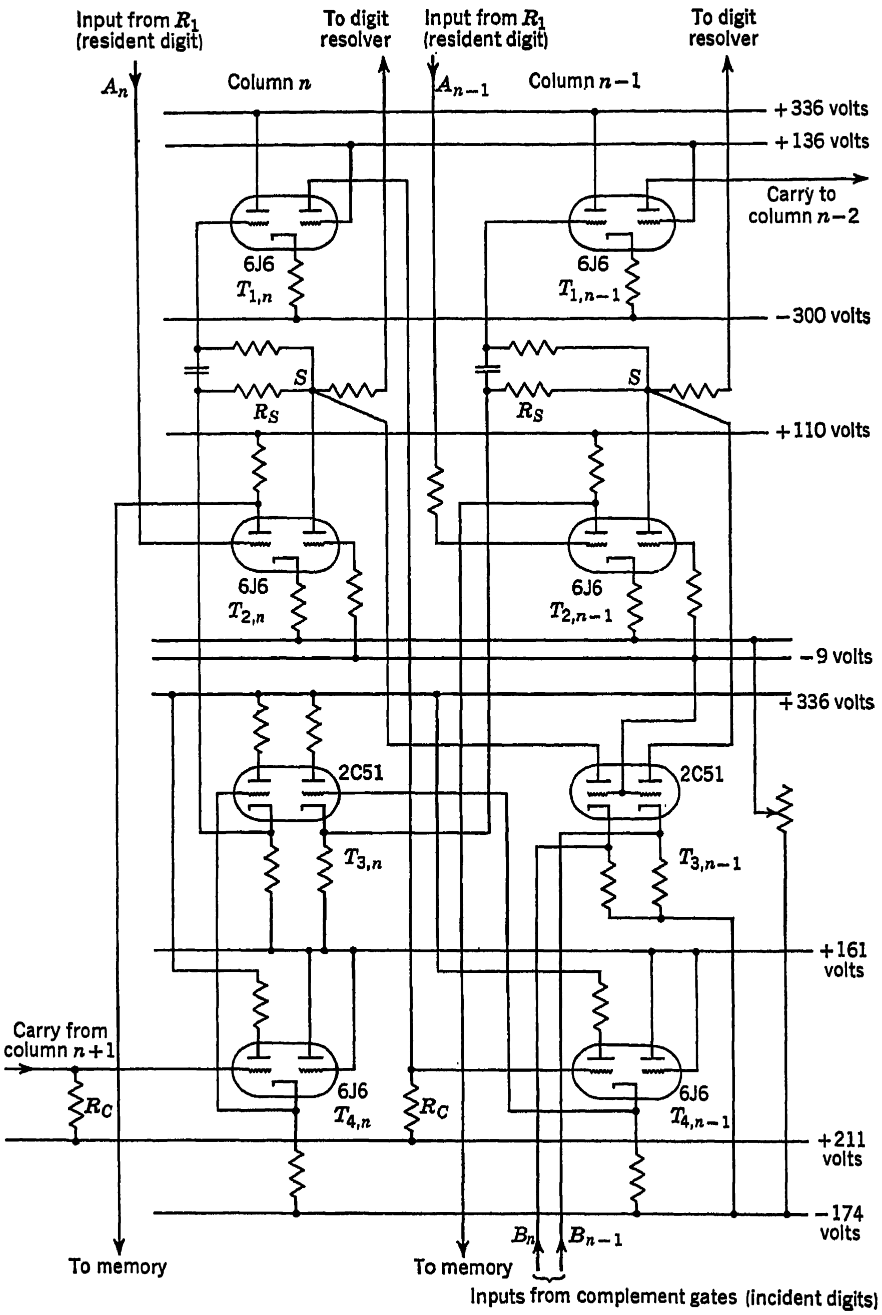


FIG. 10-10. Two columns of Kirchhoff adder.

Clearly, the cathode of $T_{1,n}$ is held so far above the right grid that the right section draws no current through the carry resistor R_C of the next stage; the original assumption, then, was equivalent to the assumption that there was no carry into the nth stage.

If the incident digit is now changed to 1, A_n drops to -40 volts, the left section of $T_{2,n}$ is off, and its right section is on, drawing a plate current through R_S of $(174 - 9)/(32 \times 10^3)$ amp $= 5.15$ ma. Hence V_S drops to $215 - 5.15 \times 10.5 \doteqdot +161$ volts.

The same effect is obtained if the resident digit is 0 but the incident digit is 1; for then, by cathode-follower action, the left section of $T_{3,n-1}$ draws 5.15 ma through R_S. Note that, with V_S at $+161$ volts, the right section of $T_{1,n}$ is still held nonconducting.

Finally, if both incident and resident digits are 1's, a total of 10.3 ma is drawn through R_S, V_S drops to $+107$ volts, and now the right half of $T_{1,n}$ conducts, drawing a current of $436/(90 \times 10^3)$ amp $= 4.85$ ma through R_C ($= 18$ kilohms) and causing the left grid of $T_{4,n-1}$ to fall to approximately $+124$ volts.

It has now been shown that, with no carry into column n, $0 + 0$ is indicated by $V_S = +215$ volts, $0 + 1 = 1 + 0$ by $+161$ volts, and $1 + 1$ by $+107$ volts, and that 0 carry is indicated by $+211$ volts, and a carry of 1 by $+124$ volts.

The effect of a carry can now be shown. Let the left grid of $T_{4,n}$ be at $+124$ volts; as this is far below $+161$ volts, conduction in the diode-connected right section holds the cathode at slightly below 161 volts. Thus the left cathode of $T_{3,n}$ is held at very nearly $+161$ volts instead of at $+211$ volts as in the 0-carry case. Consequently, V_S can assume the following nominal values:

Resident digit	Incident digit	V_S, volts
0	0	+161
0	1	+107
1	0	+107
1	1	+53

Observe that, in all cases but the first, a carry into the next column is generated.

The above analysis can be most easily summarized in a table showing the possible combinations of inputs, outputs, and carries. It must, of course, be appreciated that the four voltage levels shown are only approximate. They are, however, so widely different that fluctuations of a few volts one way or another cause no trouble. The average values in the Princeton computer have been given as $+215.3$, $+161.0$, $+108.4$, and

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Resident digit	0	0	0	0	1	1	1	1
Incident digit	0	0	1	1	0	0	1	1
Carry from next-lower stage	0	1	0	1	0	1	0	1
Sum	0	1	1	0	1	0	0	1
Carry to next-higher stage	0	0	0	1	0	1	1	1
V_S, volts	+215	+161	+161	+107	+161	+107	+107	+53

+55.9 volts, respectively. The potentiometer shown in Fig. 10-10 between the −174-volt bus and the cathode resistors of $T_{2,n}$ and $T_{2,n-1}$ was incorporated as a means of making the actual values as close as possible to the desired values.

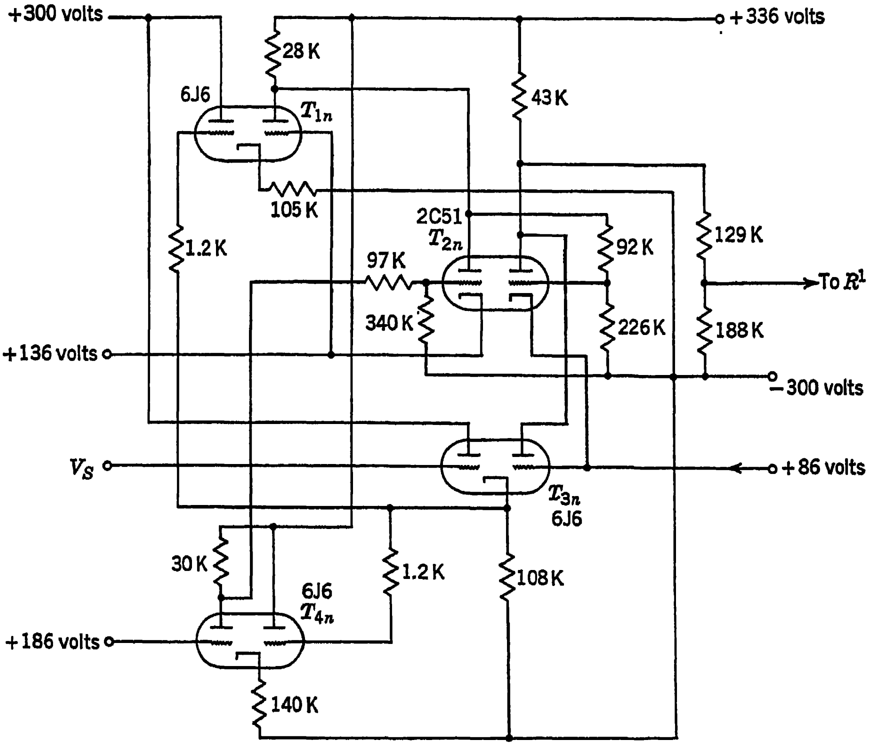


FIG. 10-11. One column of digit resolver.

It is evident that some means is needed to reduce the four levels of V_S essentially to two, since both +215 and +107 volts signify 0 and both +161 and +53 volts signify 1. This is accomplished by a circuit called a "digit resolver." The circuit involves four dual triodes per column, and is shown in Fig. 10-11. It is clear that, for the three higher values of

V_S, the cathode of $T_{3,n}$ follows the input voltage, but, for the fourth value of +53 volts, the +86-volt input to the right grid takes control. In $T_{4,n}$, the right section conducts for $V_S = +215$ volts, the left section being cut off. In all other cases the reverse takes place, and a plate current of approximately 3.47 ma flows in the left section.

In the first case ($V_S = +215$ volts), the left section of $T_{1,n}$ conducts and the right is cut off; the left section of $T_{2,n}$ conducts, drawing a plate current of about 4 ma. This is sufficient to drive the grid of the right section far below cutoff, and the output voltage is determined by the voltage divider to be +32 volts.

In the second case $V_S = +161$ volts. As noted, the left section of $V_{3,n}$ conducts, the right being cut off, and the left section of $T_{4,n}$ conducts, drawing a plate current of very nearly 3.47 ma. The left section of $T_{1,n}$ conducts, the right section being cut off. Simple calculation shows that the left grid of $T_{2,n}$ is held far below cutoff and that the right section is driven into grid current. The output voltage now falls to about −58 volts.

In the third case $V_S = +107$ volts. The same reasoning shows that the left section of $T_{2,n}$ is cut off. Here, however, the voltage of the left grid of $T_{1,n}$ is too low, and the right grid takes control, the right section drawing a plate current of 4.15 ma. This is sufficient to drive the right grid of $T_{2,n}$ far below cutoff, and the output voltage is determined by the voltage divider to be +32 volts.

Finally, let $V_S = +53$ volts. The only change is that now the right section of $T_{3,n}$ conducts. The plate current of 3.58 ma flows through the 43-kilohm plate resistor of the right section of $T_{2,n}$, and the output voltage falls to about −55 volts.

Thus, in the first and third cases, the output voltage of +32 volts signifies 0, and, in the second and fourth, voltages between −50 and −60 volts signify 1. This excessively wide discrimination is not needed; in the Princeton machine, diode "bumpers" clamp the positive excursion to ground, but the negative excursion is not limited. This is because this output is fed to the grid of a triode gate (see Chap. 4) which is disabled by holding the cathode at +10 volts but enabled by dropping it to −20. As this gate transmits only 0's, it is necessary for the voltage representing 0 to lie between these levels; the voltage representing 1 need only lie safely below −20 volts.

The accumulator as a whole consists of the double-ranked register *R*I, the adder, and the digit resolver. The block diagram, Fig. 10-12, shows how these all fit together.

10-9. A High-speed Adder for Asynchronous Machines. It has been noted above that Burks, von Neumann, and Goldstine showed that, in the addition of two binary numbers each 40 bits in length, the expected value of the length of the maximum carry sequence is only 4.6 places. This fact was not exploited in the design of the adder for the IAS machine; indeed, the output of the digit resolver is gated into R^1 only after a delay somewhat longer than the time required for a carry to propagate through the entire 40 stages. Gilchrist, Pomerene, and Wong[1] have devised an adder for parallel asynchronous machines (i.e., of the type of the IAS machine) in which the completion of all carries is detected automatically and this information is available to terminate the delay and to cause the enabling of the gates from the adder into R^1. Using their scheme, they have determined experimentally the expected value of the length of the maximum carry sequence to be 5.6 stages. Thus, their scheme comes fairly close to attaining the theoretical minimum.

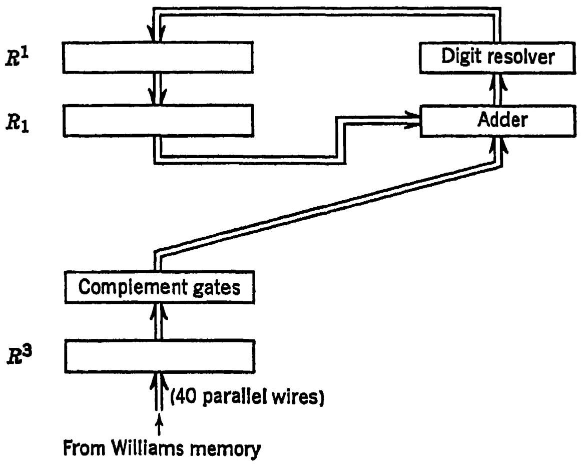


FIG. 10-12. IAS parallel accumulator.

Consider a logical adder to which the addends are presented simultaneously. The carry C_k generated in the kth stage has been shown [Eq. (10-11)] to be given by the formula

$$C_k = (A_k \wedge B_k) \vee (A_k \wedge C_{k-1}) \vee (B_k \wedge C_{k-1}) \qquad (10\text{-}27)$$

which can be rewritten in the form

$$C_k = (A_k \wedge B_k) \vee (C_{k-1} \wedge (A_k \vee B_k)) \qquad (10\text{-}28)$$

The negative, or complement, of C_k is easily found by the rules of Boolean algebra to be

$$C'_k = (A_k \vee B_k)' \vee (C'_{k-1} \wedge (A_k \wedge B_k)') \qquad (10\text{-}29)$$

[1] B. Gilchrist, J. H. Pomerene, and S. Y. Wong, Fast Carry Logic for Digital Computers, *IRE Trans. on Electronic Computers*, vol. EC-4, no. 4, pp. 133–136, December, 1956.

If $C'_k = 1$, no carry, or a "0 carry," is indicated. It is evident from inspection of these formulas that a carry of 1 is generated in any stage that contains $A_k = B_k = 1$, irrespective of the value of C_{k-1}, whereas a carry of 0 is generated in any stage containing $A_k = B_k = 0$, again irrespective of the value of C_{k-1}.

Given the above considerations, the scheme illustrated in Fig. 10-13*a* is easy to understand. Two carry lines are provided, for the 1 and the 0 carries, respectively. A carry of 1 is designated by $C_k{}^1$, and a carry of 0 by $C_k{}^0$; it is understood that $C_k{}^1$ is 0 for "off" and 1 for a carry of 1, and that $C_k{}^0$ is 0 for "off" and 1 for a carry of 0. The "off" levels may be thought of as being enforced by an "inhibit" signal which is not released until the addends have been presented to the adder. After the removal of

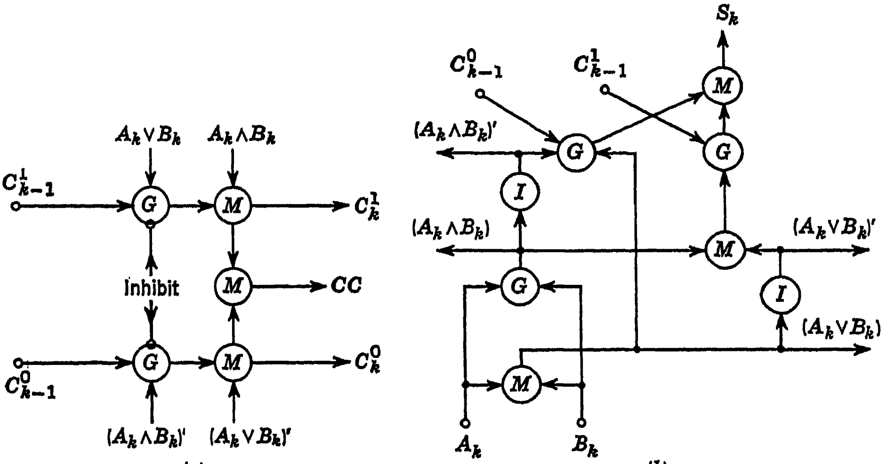


FIG. 10-13. Fast-carry parallel adder. (*a*) Fast-carry logic; (*b*) possible adder configuration to use with fast-carry logic.

the inhibit signal the gates can function normally. In the 1-carry line, $A_k = B_k = 1$ immediately starts a 1-carry sequence, and, if a "1 carry" enters the stage, it is propagated, provided that A_k and B_k are not both 0. On the other hand, $A_k = B_k = 0$ immediately causes a 0 carry to be transmitted forward, and, if a 0 carry enters a stage, it is propagated, provided that A_k and B_k are not both 1. As soon as either $C_k{}^0$ or $C_k{}^1$ becomes 1, the carry-completion signal *CC* from the *k*th stage becomes 1. The carry-completion signals from all stages are taken as inputs to a multi-input gate circuit. The output of this circuit becomes 1 as soon as all carries are complete, and this signal can, of course, be used as an indication that the addition has been completed and that the time has arrived to gate the sum into R^1.

For completeness, the logical elements needed to form S_k are shown in Fig. 10-13*b*, where

$$S_k = ((A_k \vee B_k) \wedge (A_k \wedge B_k)' \wedge C^0_{k-1}) \vee (((A_k \wedge B_k) \vee (A_k \vee B_k)') \wedge C^1_{k-1}) \qquad (10\text{-}30)$$

which can be obtained from (10-8) by (*a*) rearranging the terms, (*b*) noting that

$$(A_k \wedge B'_k) \vee (A'_k \wedge B_k) = (A_k \vee B_k) \wedge (A'_k \vee B'_k) \qquad (10\text{-}31)$$

and (*c*) recalling the definition of logical sum and the theorem T_{10} of Chap. 7.

The actual speed of an adder embodying the logic just expounded of course depends upon the circuits and circuit components used to realize it. Gilchrist, Pomerene, and Wong used simple direct-coupled vacuum-tube circuits of the type used in the IAS computer. Their measured values for the times required for the completion of carries extending over four and six stages, respectively, were 0.18 μsec and 0.22 μsec, leading to an average value of 0.21 μsec for the expected value of the maximum carry sequence, 5.6 stages in a 40-stage adder. They calculated that the essentially parallel portion of the addition required about 0.15 μsec. Hence, using this logic and quite conventional circuits, an addition time of only 0.36 μsec should be attainable.

10-10. NBS Simultaneous-carry Adder. A very fast parallel adder has been proposed by Weinberger and Smith,[1] who observed that C_k, as given by the equation

$$C_k = (A_k \wedge B_k) \vee [(A_k \vee B_k) \wedge C_{k-1}] \qquad (10\text{-}32)$$

depends only upon A_k, B_k, and C_{k-1} and that, for this reason, it is possible by successive substitutions to express C_k in terms of $A_k, A_{k-1}, \ldots, A_1$; $B_k, B_{k-1}, \ldots, B_1$; and, if desired, C_0, the "carry into the lowest-order position," which must be injected when subtraction is performed by addition of the 1's complement. Thus, since in principle it is possible to form all the carries simultaneously, it should be possible to build a very fast adder exploiting this fact. Actually, as will be seen below, it is not practicable to do quite this much, but the idea still leads to a very fast adder.

To illustrate the recursive definition of the carries, consider first

$$C_1 = (A_1 \wedge B_1) \vee [(A_1 \vee B_1) \wedge C_0] \qquad (10\text{-}33)$$

[1] A. Weinberger and J. L. Smith, A One Microsecond Adder Using One Microsecond Circuitry, *IRE Trans. on Electronic Computers*, vol. EC-5, no. 2, pp. 65–73, June, 1956.

and then

$$\begin{aligned} C_2 &= (A_2 \wedge B_2) \vee [(A_2 \vee B_2) \wedge C_1] \\ &= (A_2 \wedge B_2) \vee [(A_2 \vee B_2) \wedge (A_1 \wedge B_1)] \\ &\qquad \vee [(A_2 \vee B_2) \wedge (A_1 \vee B_1) \wedge C_0] \end{aligned} \tag{10-34}$$

For convenience, set

$$\begin{aligned} R_k &= A_k \vee B_k \\ D_k &= A_k \wedge B_k \end{aligned} \tag{10-35}$$

so that C_2 can be written in the form

$$C_2 = D_2 \vee (R_2 \wedge D_1) \vee (R_2 \wedge R_1 \wedge C_0) \tag{10-36}$$

Similarly,

$$\begin{aligned} C_3 = D_3 \vee (R_3 \wedge D_2) \vee (R_3 \wedge R_2 \wedge D_1) \\ \vee (R_3 \wedge R_2 \wedge R_1 \wedge C_0) \end{aligned} \tag{10-37}$$

and

$$\begin{aligned} C_4 = D_4 \vee (R_4 \wedge D_3) \vee (R_4 \wedge R_3 \wedge D_2) \vee (R_4 \wedge R_3 \wedge R_2 \wedge D_1) \\ \vee (R_4 \wedge R_3 \wedge R_2 \wedge R_1 \wedge C_0) \end{aligned} \tag{10-38}$$

which can also be written in the form

$$\begin{aligned} C_4 = D_4 \vee (R_4 \wedge R_3 \wedge (D_3 \vee D_2)) \vee [R_4 \wedge R_3 \wedge R_2 \wedge R_1 \\ \wedge (A_1 \vee C_0) \wedge (B_1 \vee C_0)] \end{aligned} \tag{10-39}$$

The expressions just obtained for the first few carries show the practical difficulties in the way of a straightforward application of the basic idea, since the logic circuits required even to form C_4 involve either a five-input mixer fed by gates having up to five inputs [Eq. (10-38)] or a three-input mixer fed by gates one of which has six inputs [Eq. (10-39)]. Weinberger and Smith were interested in using the DYSEAC circuit packages,[1] which contain four gates having up to six inputs each; so it was not possible for them to compute more than four carries simultaneously. As the R_k and the D_k would be generated during one clock phase, clearly C_1 to C_4 could be generated during the second clock phase. If C_5 to C_8 can all be expressed in terms of C_4, just as C_1 to C_4 were expressed in terms of C_0, then it follows that this second block of carries can be generated during the third clock phase, and so on. Hence, if a five-phase clock is used, this scheme should permit the addition of two 16-place binary numbers in 1 μsec.

Weinberger and Smith have shown how the scheme just described can be extended by the use of certain auxiliary carry functions. Inspection of Eq. (10-38) reveals that the formula for C_4 can be rewritten in the form

$$C_4 = X_4 \vee (Y_4 \wedge C_0) \tag{10-40}$$

[1] See Sec. 6-5 for an account of the basic SEAC and DYSEAC packages.

where X_4 and Y_4 are independent of C_0, being functions of the R_k and D_k only. Since they cannot be formed until the R_k and D_k have first been generated, it is clear that they cannot be formed until the second clock phase, and the same is true of X_k and Y_k for $k = 5$ to 8. However, it can be shown that it is possible to express C_9 to C_{20} all as Boolean functions of C_4 and six auxiliary functions $X_9, Y_9, X_{14}, Y_{14}, X_{18}, Y_{18}$ in such a way that (*a*) each X_k, Y_k depends only upon the D_i and R_i for $i = 1, 2, \ldots, k$; and (*b*) in the generation of each X_k, Y_k, C_k it is never necessary to use more than four gates or more than six inputs per gate. Hence, during the third clock phase, it is possible to generate 16 carries C_5 to C_{20}. By using more auxiliary functions it is possible to generate more carries simultaneously, for example, 25 carries with five pairs of auxiliary functions. This is as far as it is possible to go within the limitations of DYSEAC packages.

A further extension can be made by using a second level of auxiliary functions Z_k and W_k, which are Boolean functions of the X_i and Y_i. Using these, Weinberger and Smith have designed a 53-stage adder. In this, C_1 to C_4 are generated during the second clock phase, C_5 to C_{20} during the third, and C_{21} to C_{52} during the fourth. The sum is completed during the fifth; so, if a five-phase clock is used, the whole process requires 1 μsec for its completion. Three pairs of auxiliary carry functions, computed during the second clock phase, are required for the generation of C_5 to C_{20}. Six more pairs of first-level functions, generated during the second clock phase, and four pairs of second-level functions, generated during the third clock phase, are needed for the generation of C_{21} to C_{52}.

10-11. MIDSAC Serial-Parallel Accumulator. Some discussion of the serial-parallel data transmission and arithmetic of MIDSAC has already been given in Chap. 3. A brief account of the accumulator will be given here. Nothing will be said about the circuits, which are based upon those of SEAC, but attention will be focused upon the logic, which is illustrated in Fig. 10-14.[1]

MIDSAC words are 32 bits in length, and all numbers are proper fractions with fixed decimal point. The highest-order position (in number words) is used only in connection with input and magnetic-drum (i.e., auxiliary-memory) operations, and does not in fact appear at all in the data processing. The next bit is the sign, and the remaining 30 represent the values in the binary positions 2^{-1} to 2^{-30}. The bits will be referred to as a_{-30} to a_{-1} in ascending order of significance; a_0 is the sign, and a_1 is not used.

[1] W. Brown, J. De Turk, H. Garner, and E. Lewis, The MIDSAC Computer, report on Project MX-1599, Engineering Research Institute, University of Michigan, Ann Arbor, April, 1954.

When a number is brought from the parallel Williams memory into the shifting registers *SR*-1 to *SR*-4, the bits are in the following order from left to right: (a_{-2}, a_{-6}, a_{-10}, a_{-14}, a_{-18}, a_{-22}, a_{-26}, a_{-30}) in *SR*-1, (a_{-1}, a_{-5}, a_{-9}, a_{-13}, a_{-17}, a_{-21}, a_{-25}, a_{-29}) in *SR*-2, (a_0, a_{-4}, a_{-8}, a_{-12}, a_{-16}, a_{-20}, a_{-24}, a_{-28}) in *SR*-3, and (—, a_{-3}, a_{-7}, a_{-11}, a_{-15}, a_{-19}, a_{-23}, a_{-27}) in *SR*-4. The contents of these registers are transmitted serially to the arithmetic circuits by right shifts occurring in eight successive microseconds, and hence each shifting register transmits a train of eight pulses at a prf of 1 Mcps. These terms are slightly displaced in time; if the term from *SR*-1 begins at $t = 0$, then those from *SR*-2, *SR*-3, and *SR*-4 begin at $t = 0.25$ μsec, 0.5 μsec, and 0.75 μsec, respectively.

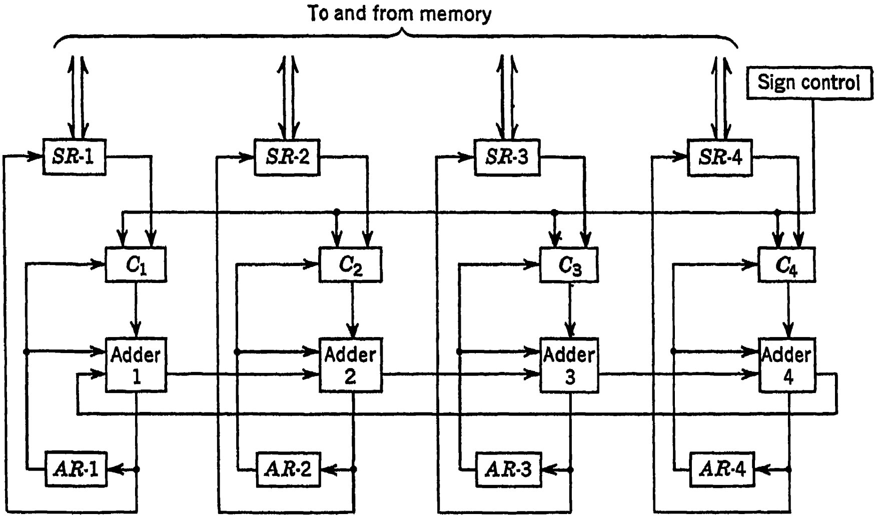


FIG. 10-14. Addition in MIDSAC.

The first step in an addition is to clear the accumulator registers and to transmit to them the first addend via the adders A_1 to A_4, which add this addend to the 0's held in the accumulator. The complementers permit the number itself or its complement to pass on to the adders. The second step is to obtain the second addend from the memory and transmit this also to the adders. The four lowest-order bits a_{-30} to a_{-27} arrive at A_1 to A_4, where they are added, respectively, to the four lowest-order bits of the first addend. The carry from the left or lowest-order column is transmitted to the next-higher-order column, and so on; the carry from the highest-order or rightmost column is transmitted back to the leftmost, where it is added when the second set of four bits a_{-26} to a_{-23} arrives from the shifting registers. As there is a delay of very nearly 0.25 μsec in each adder circuit, it is clear that all carries arrive at their destinations at just the right times. The sum so generated is put back into the accumulator registers, where it replaces the first addend. When the sum is to be transmitted back to the memory, it passes through the adders (being added to zero) and the paths shown to the left ends of the shifting registers are enabled. Eight right shifts then accept the four blocks of 8 bits into the shifting registers, from which they are transmitted in parallel to the memory. The operation of addition (or subtraction) requires five minor cycles of 8 μsec each, or 40 μsec in all. This includes the times required to obtain the three-address instruction, to obtain both addends, to perform the addition, and to transmit the sum back to the memory.

CHAPTER 11

LARGE-SCALE MEMORY DEVICES I

11-1. Introduction. This chapter will be concerned with devices for the storage of large amounts of information. Chapter 5 dealt with devices for holding a single bit of information, and Chap. 8 showed how these could be put together to form registers capable of holding single words, to which very rapid access might be had. The use of vacuum-tube registers for holding large quantities of information is so expensive as to be impracticable. It is, therefore, necessary to adopt other methods.

A considerable variety of memory devices are in use. These vary in capacity, access time (that is, the time required to recover information from the device), reliability, and cost. They can be classified in a variety of ways: as static or dynamic, as erasable or nonerasable, as volatile or nonvolatile, and so on.[1] It is appropriate at this point to define these terms. In a "static" device, the data remain fixed with respect to the memory medium until they are transferred or deliberately changed; in a "dynamic" memory, the data are in motion, usually as trains of electrical or mechanical pulses. An "erasable" memory has the property that information recorded in it can be changed at will; in a "nonerasable" memory, information once recorded can never be changed. In a "volatile" memory, information either naturally decays or is lost when certain electrical troubles occur in the associated circuitry; in a "nonvolatile" memory, information once recorded remains without deterioration until deliberately changed. The chief value of a rehearsal of these terms is to impress upon the reader the wide variety of the devices used.

Of principal concern are memory devices that permit access to an arbitrary word in times of the order of a few milliseconds or less. In the present state of the art, these are as follows: (1) magnetic drums (or disks), (2) ultrasonic delay lines, (3) electrostatic memory tubes, (4) arrays of magnetic cores, (5) arrays of ferroelectric memory cells, and (6) arrays of capacitors with associated diode circuits. The first two types depend upon mechanical motion and the motion of pulses, respectively. A random item of information is, therefore, available only periodically, and it is necessary to wait, on the average, one-half the period. The remain-

[1] Engineering Research Associates, Inc. (W. W. Stifler, Jr., ed.), "High-speed Computing Devices," chap. 4, McGraw-Hill Book Company, Inc., New York, 1950.

ing types are true random-access memories, in the sense that all items of information are equally available.

The access time of the magnetic drum (type 1) depends upon the speed of rotation and the number of playback heads; a rough figure to remember is 10 msec. The access time of the ultrasonic delay line (type 2) depends upon the material and the length; in most machines it is between ½ and 1 msec. Finally, for the random-access memories (types 3 to 6, used as parallel memories) access times run from 1 to about 25 μsec.

The capacity of the memory is a very important parameter. (1) For drums, this can be very large, depending upon the size and the packing density of the information and running up to 1 million bits or higher. (2) For ultrasonic lines, capacity depends upon length and is ordinarily 1,000 bits or less per channel, and it is possible to use a large tank with a number of parallel channels. It may ultimately turn out, if the center frequency and the bandwidth of delay lines can be sufficiently increased, that frequency multiplexing may become practicable, with a consequent multiplication of the capacity by the number of channels. As far as the author knows, no work of this kind has been attempted; certainly none has been published. (3) For electrostatic memory tubes, a good average is an array of 32 rows by 32 columns, so that each tube holds 1,024 bits. Finally, in cases 4 to 6, each bit has its own repository; limitations are imposed, however, by the complexity of the switching circuitry. The first large-scale array of magnetic cores was used to replace an electrostatic memory capable of holding 1,024 16-bit words in Whirlwind I.

Finally, as to reliability, this is best for the magnetic devices (types 1 and 4) and poorest for the electrostatic memory tubes (type 3). Presumably, arrays of ferroelectric elements (type 5) will prove to be as reliable as arrays of magnetic cores. The diode-capacitor system (type 6) appears to be very good, too, but it is natural to expect that a volatile system, in which the information must be periodically regenerated, will not equal a nonvolatile system in performance. In general, it is to be expected that magnetic drums will remain popular for low-speed machines and for use as auxiliary memories in high-speed machines, whereas arrays of magnetic cores and, perhaps, arrays of ferroelectric elements or of capacitors will tend to replace other memories in high-speed machines. Of course, it is quite possible that, by the time these words reach the reader, some new discovery will have made all the types described here appear obsolescent!

The present chapter will deal with magnetic drums, ultrasonic delay lines, and magnetostrictive delay lines. Electrostatic memory tubes and arrays of magnetic cores will be covered in Chap. 12. Ferroelectric arrays are not yet practical and will receive no more attention.

11-2. Relative-motion Magnetic Memories: General. Consider a ring-shaped core A of magnetic material in which there is a small air gap and which carries a winding, as shown in Fig. 11-1. Let the core be in contact at the gap with a piece of magnetizable material B, initially in a completely unmagnetized state. If, now, a unidirectional current is caused to flow in the winding, the magnetic flux induced in the core takes the low-reluctance path from A into B and back into A. In flowing through B, the flux may be considered to orient a great number of elementary magnets, so that, when the current is interrupted, a portion of B is left permanently magnetized. The polarity of the dipole left on B depends on the sense of winding of the coil and the sense of current flow. If the current sense is such that the magnetic polarity of the coil during the excitation is as shown in Fig. 11-1*a* the residual dipole has the polarity of Fig. 11-1*b*. The same effect is obtained, although the residual magnetization is naturally less intense, if B, instead of being in contact with A during the excitation, is slightly removed from it; the residual

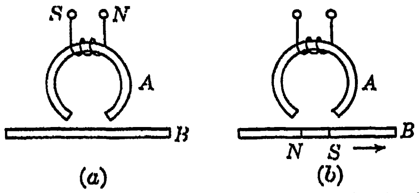


FIG. 11-1. Magnetic recording and playback.

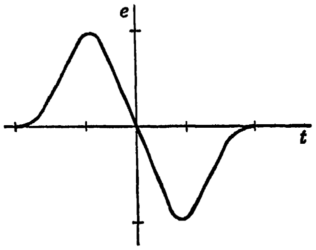


FIG. 11-2. Playback signal.

intensity falls off sharply with increasing distance. As this effect depends on the fringing flux, core design attempts to increase this flux and usually results in a core that is more or less sharply pointed at the gap. The core and winding together are commonly called a "head," with the qualifier "reading" or "writing" according to function.

Now suppose that B, bearing the residual dipole, is displaced longitudinally so that the dipole is no longer under the gap and is then moved longitudinally so that the dipole passes the gap. During this motion, some of the flux emanating from the dipole enters A on one side of the gap and emerges from it on the other. Thus, during the motion, the flux linking the winding on A varies, and an emf proportional to the rate of change of flux is developed across the terminals of the winding. This emf has the form shown in Fig. 11-2, or its negative, depending upon the sense of motion. The amplitude of the voltage developed across the winding of the head depends strongly upon the distance of the material B from the gap during the motion. For example, in the case of a typical head, if the spacing between the head and the material upon which the information is recorded is increased from 0.001 to 0.002 in., the signal amplitude decreases by about 17 db, and, if the spacing is further increased from 0.002 to 0.003 in., the amplitude decreases further by about 15 db.* In the case of drums and disks, where the relative motion is too fast to permit the head and recording material to be in contact, this variation can be seen to be of great importance: it determines tolerances on the drum's eccentricity that must be carefully observed. This point will recur in the discussion of drums in Sec. 11-3.

If the material B had been in motion (in the plane of the paper) during the excitation, the magnetization would have been distributed over a greater distance. This means that, during the reading of the information, the flux linking the winding of the head would have built up to a maximum and stayed there for a time, depending upon the speed of motion of the material during both recording and reading (or "playback"). Consequently the voltage developed across the head winding would have had the form shown in Fig. 11-3 (or its negative), to use a rather extreme illustration.[1] Ordinarily, the length of the exciting pulse is so chosen that the motion of the material during one pulse time is negligible, and the response shown in Fig. 11-2 is obtained.[2]

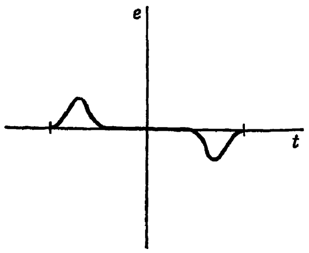


FIG. 11-3. Playback signal if motion is slow.

Information can be recorded on and recovered from a magnetizable medium; for all practical purposes, the information, once recorded, is retained indefinitely. It can be altered at will; any events that would accidentally alter it would be sufficiently devastating in their effects to damage some part of the equipment permanently. Suppose that only two possible magnetic states of the medium are permitted: magnetic saturation in the direction of motion and saturation in the opposite direction. This is quite in the spirit of the binary encoding of information. To write, drive the medium to saturation in one direction; to alter this (or to "erase"), merely drive it to saturation in the opposite direction. If it is desired to return the medium to its unmagnetized state, a-c erasure can be used. This consists in applying an a-c signal, the amplitude of which decreases to zero over a number of cycles while the gap in the head is centered over the spot to be erased.[3] This method is not particularly good for altering information during the use of the memory,

* R. L. Perkins, Magnetic Drum Storage Devices, *Prod. Eng.*, vol. 24, no. 8, pp. 192–195, August, 1953.

[1] A. D. Booth and K. H. V. Booth, "Automatic Digital Calculators," p. 116, Butterworth and Co. (Publishers) Ltd., London, 1953.

[2] "High-speed Computing Devices," pp. 327–328.

[3] *Ibid.*, p. 326.

but could be used in preparing the medium before any information is recorded on it, to ensure that it starts in a demagnetized state.

The effects just outlined make possible a considerable variety of magnetic memory devices, depending upon the physical form of the magnetizable medium. For reasonably rapid-access internal memory, the ordinary form is the drum, although disks are also in use. For input and output and also for auxiliary memory, wires and tapes are the common forms. Finally, a very-large-capacity memory with rather long access time has been built up of a number of stationary disks so arranged that they can all be consulted by a single movable head.

In all cases, assume a uniform relative motion of head and medium. Assuming also that the information pulses arrive at the recording head at a uniform pulse-repetition rate, note that the material can be regarded as divided along the direction of motion into a number of memory cells, each of which can hold a 0 or a 1. The recording of 0's and 1's can be arranged in a number of ways.[1]

In the first, or "two-level return-to-zero" scheme, one level of magnetization is assigned to represent 0's, and the medium is prepared by being put into that state. This is ordinarily chosen as saturation in one direction, so that all along the direction of motion the elementary magnets are lined up in the same direction. To write 1, the head is pulsed with sufficient current to cause saturation in the opposite direction; to erase 1, a pulse of equal amplitude but opposite sign is used. In reading, signals of the form of Fig. 11-2 are obtained for 1's; 0's ideally give rise to no signal at all. In practice, however, the erasure of 1's does not quite restore the original uniform saturation in one direction, and in reading 0's where erasure has been incomplete, small signals are obtained; these cause no trouble if adequate precautions are taken. The reading-head output for the sequence 101101 in this case is shown in Fig. 11-4.

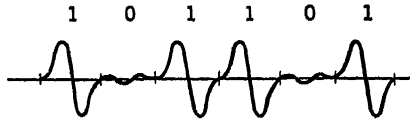


FIG. 11-4. Output signals in two-level return-to-zero system.

It would also be possible to choose the demagnetized state to represent 0. With this system of representation, either a-c erasure must be used or the pulse used to erase 1 must have exactly the correct amplitude to drive the medium from saturation to demagnetization. Neither choice seems to have much to recommend it.[2]

A second scheme has been called the "three-level return-to-zero"[3]

[1] *Ibid.*, pp. 328–329.
[2] J. H. McGuigan, Combined Reading and Writing on a Magnetic Drum, *Proc. IRE*, vol. 41, no. 10, pp. 1438–1444, October, 1953.
[3] "High-speed Computing Devices," p. 329.

system. The medium is first caused to assume the completely unmagnetized state, for example, by means of a-c erasure. To record 0's and 1's, pulses of equal amplitude and opposite sign are used, so that these bits are physically represented on the medium by dipoles of equal strength but opposite orientation. Every cell holding information gives a definite

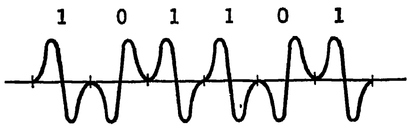


FIG. 11-5. Output signals in three-level system.

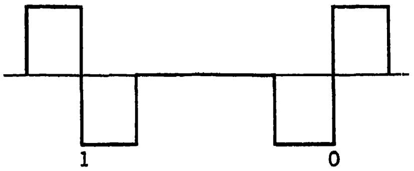


FIG. 11-6. Writing signals in Manchester system.

output; only spaces purposely left blank give no signal. The reading-head output for 101101 appears in Fig. 11-5.

A third scheme is the phase-modulation scheme due to F. C. Williams and his associates.[1] Every information cell is driven to saturation, half in one direction, half in the other. If these two directions are called "positive" and "negative," then 1 is represented by positive saturation followed by negative, and 0 by the exact opposite; the pulses to the recording heads used to accomplish this are as shown in Fig. 11-6. The resulting magnetization of the medium for a 1 is shown in Fig. 11-7. As a reading head moves past the two dipoles in the arrow direction, the rate of change of the flux linking the head winding is first large with one sign (at point 1), then zero at point 2, then large of the opposite sign at 3, then zero again at 4, and large again at 5. The

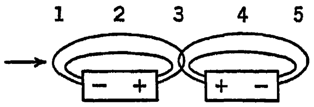


FIG. 11-7. Distribution of magnetization for 1.

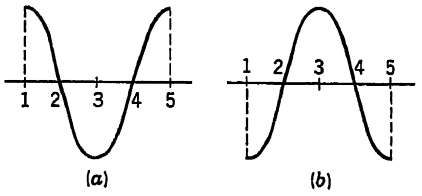


FIG. 11-8. Output signals (a) for 1, (b) for 0.

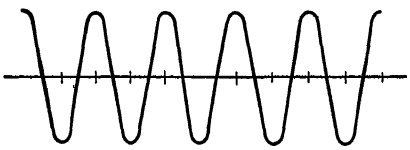


FIG. 11-9. Output for a succession of 1's.

result is an output voltage as shown in Fig. 11-8*a*. By the same argument, Fig. 11-8*b* is recognized as the form of the voltage obtained upon reading 0. The important thing is that waveforms *a* and *b* are negatives of each other; the actual polarity of both can be reversed if desired. If

[1] F. C. Williams, T. Kilburn, and G. E. Thomas, Universal High Speed Digital Computers: A Magnetic Store, *Proc. IEE*, vol. 99, pt. 2, no. 68, pp. 94–106, April, 1952.

a sequence of 1's or of 0's is recorded, the reading waveform is a sequence of waveforms *a* or *b* (Fig. 11-9). However, if a 1 follows a 0, the situation is somewhat different. The magnetization of the medium can be thought of as shown in Fig. 11-10. The two dipoles in the middle essentially combine into one, with the result that the zero rates of change of

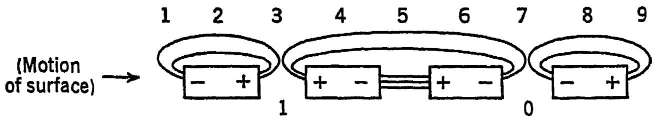


FIG. 11-10. Distribution of magnetization for 1 and 0.

flux are found now at points 2, 5, and 8 and that the reading waveform is as shown in Fig. 11-11, provided that 1 is preceded by 1 and 0 followed by 0. Consider immediately a more complicated case: the sequence 11010100. The output is shown in Fig. 11-12. Assume that either zero magnetization or a 1 precedes the first bit and that zero magnetization or a 0 follows the last 0. Notice that a negative peak must always occur in the middle of reading a 1 but that the other negative peaks occur between 0's. Hence, if the waveform, which resembles very closely a frequency-modulated sinusoid, is squared (for example, by putting it through an overdriven amplifier) and if these negative peaks are used to open a gate whose other input is a sequence of narrow pulses occurring in the middle of the reading time of each digit, then the gate output

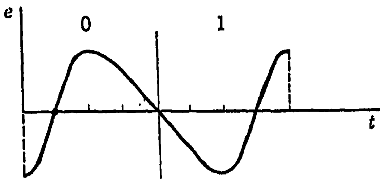


FIG. 11-11. Output for 0 followed by 1.

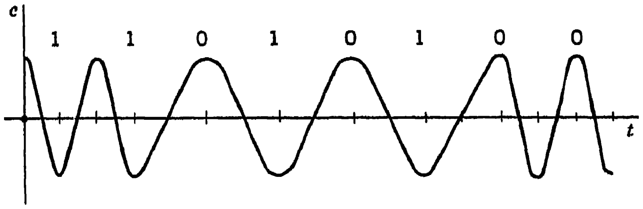


FIG. 11-12. Output for typical sequence of 0's and 1's.

is a narrow pulse for each 1 and nothing for each 0. Since the original information pulses occurred at the beginning of the digit periods, it is necessary to correct for the half-period delay in the reading. An advantage of this method is that the reading head can be transformer-coupled to a simple tuned amplifier, so low-impedance heads can be used.

Still a fourth method of recording is called the "non-return-to-zero" system.[1,2] This will be illustrated for the case of two levels of magnetiza-

[1] "High-speed Computing Devices," pp. 330–331.

[2] Booth and Booth, *op. cit.*, p. 117.

tion. Suppose that the recording head is driven by a toggle in such a way that the medium is saturated in the plus direction when the toggle holds a 1, but in the minus direction when it holds a zero. The first 1 in the information to be recorded initiates the saturation in the plus direction. Since the toggle is unchanged if a succession of 1's is received, "plus" magnetization is recorded until the first 0 changes the state of the toggle and initiates the recording of negative magnetization. In the reading process, voltage pulses appear at the terminals of the head winding only at points of transition from saturation in one sense to saturation in the other. Obviously, polarities may be chosen such that a positive pulse marks the beginning of a string of 1's and a negative pulse, the beginning of a string of 0's. These pulses, respectively, can be used to set a toggle in its 1 and 0 states, and, depending upon the type of gate used, one or another of the available voltages can be used to enable a gate while the toggle is in the 1 state. The other gate input is taken from the clock; so the gate output is a pulse for each 1 stored in the medium and no pulse for each 0; the information is correctly recovered.

The non-return-to-zero scheme permits the most efficient use of the magnetic medium, for it is evident that, if the cell repetition rate is the same in both a return and a nonreturn system, in the latter the maximum frequency that must be handled by the heads and associated circuits is, on the average, one-half the frequency that must be handled in the former. Thus in a nonreturn system it should be possible to double the density of information recorded on the medium with no deterioration of performance. The need for certain additional switching may prevent the attainment of the theoretical doubling; for example, if there are a number of parallel tracks in the memory, as in the case of a drum, it is necessary to provide pulses to set the output toggle to 0 whenever the output circuits are switched from one reading head to another. In the Logistics computer, this switching limited the density of information to 140 bits per in., where doubling would have given 160.*

Physically, there is a disadvantage in the scheme as described here: since current sufficient to provide saturation flux must at all times flow through the recording head, it is implied that a high-impedance head must be used.[1]

A fifth method [2,3] is nonreturn in effect, though the recording is done as in a return-to-zero method. The essential feature here is that the recording pulses (of opposite polarity for 0 and 1) are delivered to the recording

* R. S. Erickson, The Logistics Computer, *Proc. IRE*, vol. 41, no. 10, pp. 1325–1332, October, 1953.

[1] Booth and Booth, *op. cit.*, p. 117.

[2] D. Eadie, EDVAC Drum Memory Phase System of Magnetic Recording, *Elec. Eng.*, vol. 72, no. 7, pp. 590–595, July, 1953.

[3] Booth and Booth, *op. cit.*, pp. 331–332.

heads at so fast a rate that the magnetization remains in one sense for a string of like bits and reverses only when 1 follows 0 or 0 follows 1. This is essentially a nonreturn system, but it is clear that complete saturation is not maintained throughout a sequence of like bits, for a slight ripple occurs. The theoretical doubling of information density in the medium is obtainable in this case, as in the fourth method.

Still other methods of reading are possible. It is possible to use a pulse-modulated carrier signal or a frequency-modulated carrier. No advantages and certain definite disadvantages are inherent in these methods; they will not be discussed here.[1]

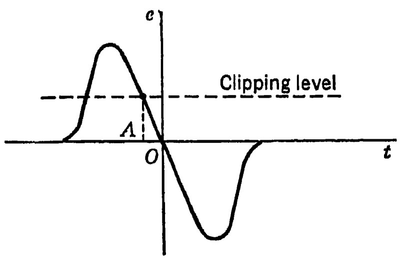


FIG. 11-13. Illustration of DIAD system.

In all the systems of recording treated above, the contents of a memory cell can be altered merely by driving it to the opposite state of saturation while it is under the head. We ordinarily think of reading and writing as completely separate operations; that is, information is either being recorded in a cell or recovered from it. In some cases, however, it is desirable to read information from a cell and then to alter it while it is still under the head. Of course, it is possible to accomplish the same thing by having the cell pass first under a reading head and then, one digit time later, under a recording head. But both operations can be performed by a dual-purpose reading-recording head.[2] This scheme was incorporated in the Bell Telephone Laboratories' Drum Information Assembler and Dispatcher (DIAD).[3] To understand the theoretical basis of this, return to the output of a reading head that uses a two-level return-to-zero system. The reading-head voltage for a 1 is shown again in Fig. 11-13. It has been mentioned above that spots on which 0 has been rewritten over 1 do not give zero output, but do give an output fairly small compared with the 1 output. Therefore, there can be established on the amplified reading-head output a clipping level that gives a pulse for a 1 but none for a 0. Observe (Fig. 11-13) that this implies that reading has been essentially accomplished before the cell is centered under the head. The time *AO* between reading and centering is, therefore, available for the setting up of the writing circuits. In the DIAD system, which operates at a cell density of 30 cells per inch, this is 2 μsec, and it

[1] *Ibid.*, pp. 117–118.

[2] J. H. McGuigan, Combined Reading and Writing on a Magnetic Drum, *Proc. IRE*, vol. 41, no. 10, pp. 1438–1444, October, 1953.

[3] W. A. Malthaner and H. E. Vaughan, An Automatic Telephone System Employing Magnetic Drum Memory, *Proc. IRE*, vol. 41, no. 10, pp. 1341–1347, October, 1953.

suffices for all the required operations. The designers of the system feel that, with additional refinements, this "single-pass" reading-and-writing technique can be extended to operate with cell densities comparable to those used in the ordinary computer memory systems, say 100 cells per inch. The single-pass system poses rather severe circuit-design problems which have been solved successfully.[1]

The treatment given so far in this section applies to magnetic drums and disks and to magnetic tapes and wires, that is, to the whole class of magnetic memories that depend upon relative motion of the medium and the recording and reading devices. The drums and disks are used for internal memories in some machines and as auxiliary memories in others. They will be treated in Sec. 11-3.

11-3. Magnetic Drum and Disk. Magnetic drums exist in a considerable variety of sizes, shapes, and structures. There are also considerable differences in the density with which information is recorded, in the structure and mounting of heads, and various other details. In this section information will be given about some currently used designs.

A pioneer in the construction and use of magnetic drums has been the Engineering Research Associates division of Remington-Rand, Inc.[2] (ERA), which has built more drums than any other commercial supplier. The current ERA practice[3] is to use a precisely milled aluminum casting coated with a magnetic oxide of iron. The cast drum is shrunk or taper-fitted to its shaft. In Sec. 11-2 it is stated that the spacing between drum surface and reading head has a profound effect upon reading-signal amplitude and that this may drop by as much as 17 db when the spacing varies from 0.001 to 0.002 in. and by 15 db when the spacing increases from 0.002 to 0.004 in. This shows the necessity of holding the precision of the peripheral surface to close tolerances. The ERA practice is to carry out the final machining, coating, and balancing operations with the drum mounted in the bearings in which it is to turn and to require that the variation in radius be less than 0.0002 in. The finished drum turns inside a cast-aluminum case, with the rotor bearings supported in pillow blocks machined in the case. The case is drilled with holes in which are inserted the dual-purpose read-write heads. As there are 16 information "tracks" per inch along the drum surface and the heads are single units, it is necessary to stagger the mountings about the drum. The packing density of information along any track is 80 bits per in. if return-to-zero recording is used and about 140 bits per in. if non-return-to-zero recording is used.

[1] The interested reader will find a detailed account of the problems and the solutions in McGuigan, *op. cit.*

[2] Now the Remington-Rand Univac Division of Sperry-Rand, Inc.

[3] Perkins, *op. cit.*; see pp. 192 and 193 for the details given in this paragraph.

The ERA drums come in a variety of sizes, up to some that hold 4,500,000 bits in the non-return system. The smaller drums are driven directly by three-phase induction motors, and the larger drums are driven by axial air-gap motors through centrifugal clutches. The speed of rotation is generally chosen so that the linear speed of the drum surfaces is about 1,600 in. per sec; this is 60 rps for the 8½-in. drum. The drum assembly is designed for mounting with horizontal axis.

A considerable variety of other mechanical designs of drums are in use, but these have not been described in much detail in the literature. The drums of the Harvard Mark III computer are similarly machined from solid aluminum blocks and coated with a film of magnetic oxide of iron suspended in a plastic lacquer. The heads are staggered by being mounted on helical brackets.[1]

Turning to hollow drums, one such, built of drawn brass tubing coated with appropriate material, was designed and built at the Institute for Advanced Study, and has been fully described in IAS reports.[2] This was also mounted horizontally.

Vertical mounting appears to be more popular than horizontal; it is used, e.g., in the Harvard Mark IV, in the computer built by Ferranti, Ltd., and in a great number of small machines. This type of mounting avoids one problem with horizontal mounting. If a fairly long, horizontally mounted drum is allowed to stand stationary for a time (over the weekend, for example), it tends to deform plastically, and this deformation may be so great that, when the drum is first put in rotation, it actually comes into contact with the heads, causing serious damage. The deformation can, of course, be avoided by providing for rotation of the drum while the rest of the machine is shut down. The bearing problem also is somewhat simplified by vertical mounting.

The Ferranti drum,[3] at least the one used in the FERUT computer, is a hollow cylinder 6 in. in diameter and 8.5 in. in length, with eccentricity less than 0.0003 in. and driven by an internally mounted squirrel-cage induction motor. An eddy-current brake is also mounted internally. The normal running speed is 1,950 rpm. The digit-packing density is such that each track holds 2,560 bits. There are 256 information tracks

[1] Harvard Computation Laboratory, "Description of a Magnetic Drum Calculator," chaps. 1 and 3, Harvard University Press, Cambridge, Mass., 1952.

[2] J. H. Bigelow, H. H. Goldstine, R. W. Melville, P. Panagos, J. H. Pomerene, J. Rosenberg, M. Rubinoff, and W. H. Ware, Fifth Interim Progress Report on the Physical Realization of an Electronic Computing Instrument, Institute for Advanced Study, Princeton, N.J., 1949. The account of the drum-and-head design is contained in the appendix.

[3] B. W. Pollard, The Design, Construction and Performance of a Large-scale General-purpose Digital Computer, in "Review of Electronic Digital Computers," pp. 62–69, American Institute of Electrical Engineers, New York, 1952.

on the drum, each holding 64 words of 40 bits each. The heads are mounted in eight stacks of 32 each. The magnetizable surface is obtained by nickel-plating the drum; this does not, however, give as good signal strength as the oxide coating to which a change has been made in later models.

The brake mentioned in the description of the Ferranti drum is part of the servo system designed to synchronize the drum to the clock. Actually, the motor could cause the drum to rotate considerably faster than is desired; so the brake is always more or less on. A waveform derived from spots recorded on one track of the drum is compared with a pulse train derived from the master clock, and the brake is either relaxed or put on harder according as the pulse train is drifting ahead of or lagging behind the clock. This is quite different from the usual procedure, which is to let the drum be the clock, recording on one track a 1 in every digit cell and using the output of the reading head of that reference track as the clock signal. It can be argued that, if several drums are to be used in the same computer, it is necessary to synchronize them and so it is just as well to synchronize all drums with the master clock. In the Harvard Mark III, however, a set of four drums is driven through a gearbox by a single motor; and if, in other parts of the computer, a faster clock is needed than that derived from one track of the drum, it is quite possible to use familiar frequency-multiplying techniques.

Drum-rotation rates are frequently a nominal 3,600 rpm obtained from a synchronous motor, though on many machines they are lower and on some higher, for example, 6,900 rpm on Remington Rand (ERA) 1102 and on the Harvard Mark III, approximately 12,000 on the IBM 650, and 15,000 in the machine at the Technische Hochschule, Munich. One reason for using higher speeds of rotation of the drum is, of course, to shorten the average access time, which is one-half the time required for a single revolution.

Shortened access time can also be obtained by providing more than one head per track, but this is an expensive expedient. In some designs, a certain number of tracks on the drum are set aside for the high-speed memory and provided with several heads, although most of the tracks have but a single head; provision is made for the transfer of the information in blocks from the lower to the higher-speed part of the memory. In one design,[1] the high-speed part of the drum is provided with four pairs of heads per track, each pair being disposed as shown in Fig. 11-14, where the arrow signifies the sense of rotation. Information recorded on the

[1] "A Symposium on Commercially Available General Purpose Electronic Digital Computers of Moderate Cost," Office of Technical Services, Washington, D.C., 1952. See pp. 31–36 on the Consolidated Engineering Corporation's model 30-201 computer.

surface by the record head is picked up by the read head, amplified and reshaped and again sent to the record head to be recorded on the surface. This is essentially a delay-line scheme. In the computer cited, the high-speed tracks contain only two-fifths as much information as the low-speed tracks, but the access time is reduced by a factor of 10. A similar scheme, used for holding a single word between the time it is read out of the main drum memory and the time it is needed in the arithmetic unit, is used in the Harvard Mark III computer.[1] There are three of these so-called "transfer channels" in the machine.

The only disk memory about which information is available is that of the Elliott-NRDC 401 computer.[2] The disk used is 9 in. in diameter, of forged brass, with an annular recording surface 1½ in. wide, ground true within ±0.00015 in. Using a magnetic iron oxide (Fe_2O_3) coating 0.0015 in. thick and a speed of rotation of 4,500 rpm, it was found possible to operate at a density of 160 bits per in. along each track with a track

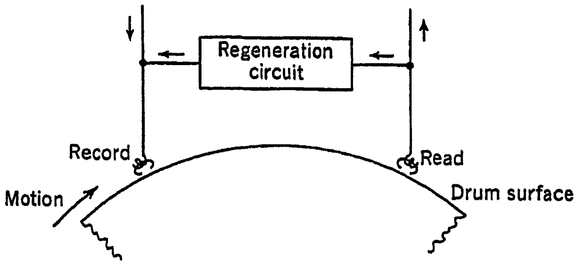


FIG. 11-14. Drum track used as delay line.

spacing of 0.01 in. The head is spaced 0.001 in. from the recording surface; using a 50-turn head winding at this spacing, a reading signal of about ½ mv is obtained.

11-4. Heads. In this section will be given some information on recording and reading-head designs now in use. As stated above, in most cases the same head is used for both functions. In some applications, however, for example, in the transfer channels of the Harvard Mark III, it is necessary to assign one head to the recording function and one to reading.

In drum and disk recording, the speed of motion of the medium makes it impossible to allow the medium and the head to be in contact, and the problem of head design is to get as much of the flux into the recording medium as possible. A variety of core designs are in use, and sizes of windings also vary considerably, according to the impedance levels at which they must operate.

The pole pieces of the heads used in the Harvard Mark III computer[1]

[1] "Description of a Magnetic Drum Calculator," chap. 1, pp. 3–4.
[2] *Proc. ACM*, Pittsburgh, May 2–3, 1952, p. 202.

are split Mumetal cores, each half built up of 12 laminations of Mumetal 0.006 in. thick, stacked and cemented together to a total thickness of about 0.1 in. The active tips of the two cores are carefully lapped and polished. On the reading head a single winding of 250 turns is used, and on the recording head, two 300-turn coils wound in opposite senses with the common connection grounded; one of the coils is excited to record 1, the other to record 0. The general appearance of the core and one coil is shown in Fig. 11-15*a*.

The heads used in ERA drums are dual-purpose heads, carrying a read coil and a center-tapped write coil. The general view is given in Fig. 11-15*b*.[1]

The head developed for the phase-modulation system of recording is shown in Fig. 11-15*c*. A single-turn winding is used on each part of the head. This makes it necessary to drive the recording head by a step-down transformer; the recording current during the pulse is 4 amp.

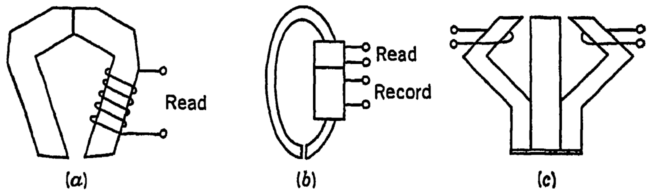


FIG. 11-15. Examples of heads. (*a*) Harvard Mark III, (*b*) ERA (Remington-Rand record-read, (*c*) Manchester record-read.

The whole assembly is very compact, being less than 0.1 in. thick; so a large number of assemblies can be stacked side by side. The gaps in both the read and the record heads are 0.001 in. wide, which is a fairly representative figure for heads in general.

A single-turn head without any core has been studied in some detail.[2] The attractive feature of such a head is its very low impedance; this is conducive to fast operation. However, (1) it is difficult to drive the head hard enough to saturate the medium; (2) information density on the medium is certainly no better than with heads of the type discussed above; (3) the output voltage is naturally very low: in the experiments cited never in excess of 150 μv peak to peak (with iron cores and multiturn windings more like 50 mv); and (4) very fine wire is required for good results; a diameter of 0.001 in. is satisfactory. To avoid the difficulties involved in mounting single loops of very fine wire with no support, Booth constructed heads of one or very few turns, using as a core a single thin lamination of high-permeability metal; he found rather considerable

[1] Perkins, *op. cit.*, pp. 193–194.
[2] Bigelow et al., *op. cit.*, appendix, sec. 4.

improvement in output signal.[1] It has been pointed out above that at least one commercial design of a single-turn head is now in use.

11-5. Circuits. This section will deal only with the read-in and read-out circuits and will not deal with switching, which is variously accomplished, by means of electronic function switches (e.g., matrices) or by equivalent relay circuits.

A simple and straightforward recording circuit used in the Harvard Mark III is shown in logical form in Fig. 11-16. The signal is a positive pulse if 1 is to be recorded, but nothing at all if 0 is to be recorded; the "record" pulses are positive and are delivered to the circuit if either 1 or 0 is to be recorded.[2] If no signal pulse is received, G_2 is not enabled and G_1 is not inhibited; so the positive record pulse passes through G_1, is inverted, and drives into conduction the power amplifier, which is so biased as to be normally cut off. The negative output pulse is delivered to the recording head either by way of a pulse transformer or through a

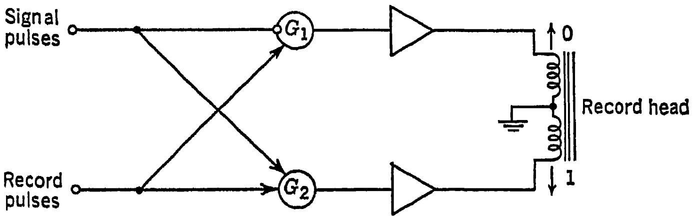


FIG. 11-16. Mark III record circuit.

simple resistive network, and current is drawn upward in the direction designated by 0 in the drawing. On the other hand, if a 1 is to be recorded, the signal inhibits G_1 and enables G_2, the net effect being that current is drawn downward in the 1 direction.

A variety of other circuits are in use. Some use blocking oscillators to produce a sharp pulse of current through the head winding.[3,4] If low-impedance heads (e.g., of a single turn or a very few turns) are used, it is necessary to use current pulses of very great amplitude, 10 amp or more, and very brief duration, e.g., a microsecond or so.[4] In some experimental work, these have been developed by means of a blocking oscillator followed by a cathode follower using tubes capable of carrying rather large currents.[4] A more attractive scheme is to insert between driving circuits and head a pulse transformer of fairly high turns ratio to effect the neces-

[1] A. D. Booth, A Magnetic Digital Storage System, *Electronic Eng.*, vol. 21, no. 257, pp. 234–238, July, 1949.

[2] "Description of a Magnetic Drum Calculator," pp. 60–61.

[3] McGuigan, *op. cit.*, pp. 1440–1442.

[4] Bigelow et al., *op. cit.*, appendix, sec. 7 for the circuit; sec. 6 for correlation of experimental results, pulse amplitude, packing density, and playback amplitude.

sary current step-up.[1] A disadvantage of being obliged to transmit high driving current to the heads is that it makes electronic switching between driver and heads difficult. It is obviously wasteful to use a separate driver for each head.

A reading, or playback, circuit is shown in logical form in Fig. 11-17; this is a circuit developed at Harvard for the Mark III.[2]

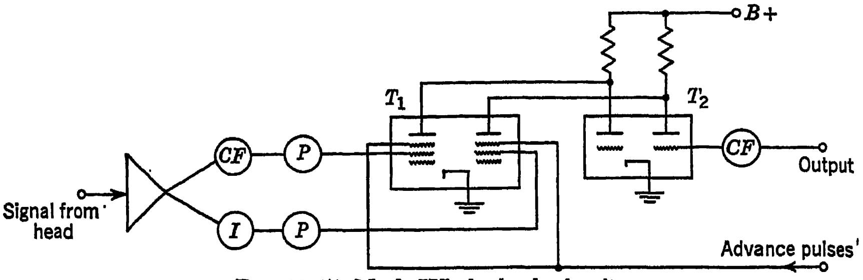


FIG. 11-17. Mark III playback circuit.

In analyzing operation of this circuit, it must be remembered that the reading-head output for 0 is much like the curve of Fig. 11-2, that the curve for 1 is the negative of the curve for 0, and that its peak-to-peak amplitude is of the order of 50 mv. This signal first encounters a two-stage video amplifier; the passband must extend to 150 kcps or more. The amplifier output branches to an inverter and a cathode follower, each followed by a puller triode (see Chap. 5) connected to the screen of one of the pentodes of a toggle that is made up of the cathodes, control, and screen grids of a pair of pentodes. The suppressors of these pentodes are driven by the "advance" pulses derived from one of the timing tracks of the drum, and the plates are connected to the plates of a conventional triode toggle, the grid voltage of one grid of which drives a cathode follower to produce the output.

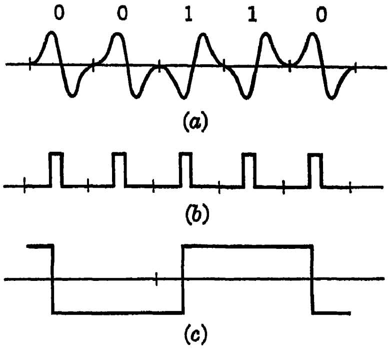


FIG. 11-18. Mark III playback-circuit waveforms. (a) Input, (b) advance pulses, (c) output.

Suppose that the input pulse train represents 00110 as shown in Fig. 11-18a; the advance pulses are shown in Fig. 11-18b, occurring at about the same time as the crossovers of the input signals. The first 0 causes the flow of screen current in the left section of T_1 and cuts it off

[1] Booth and Booth, *op. cit.*, pp. 119–120.

[2] "Description of a Magnetic Drum Calculator," pp. 57–59.

in the right section. Before the waveform swings negative enough to switch T_1 again, the advance pulse raises the suppressor voltage sufficiently to allow plate current to flow in the left tube of T_1, and this in turn guarantees that the left triode of T_2 will assume the conducting state. As a result, the grid voltage of the right triode of T_2 assumes its lower value, and the output is low. During the negative swing of the 0 voltage, T_1 is switched again, but this cannot affect T_2 because no plate current flows in either tube of T_1. When the second 0 arrives, T_1 is switched exactly as it was for the first 0. However, T_2 already holds the same information as T_1 and so the advance pulse causes no change; the output voltage continues low. When the 1 signal arrives, it switches T_1 to the state in which screen current flows in the right tube. The advance pulse causes T_2 to switch, and the output voltage rises to the higher of its two possible values. Continuing to trace the effect of successive signals, it is possible to trace the waveform of Fig. 11-18*c*. If waveform *c* is taken as one input to a gate (the positive level being the enabling signal) and if the other input is a train of pulses synchronized with the advance pulses but lagging behind them by about one-quarter of a pulse-repetition period, the gate output becomes a pulse for each 1 read from the drum and no pulse for each 0; so the information has been converted to the desired form.

Drum memories necessarily contain a number of "timing tracks"; for example, a track may contain a 1 for every digit cell about the drum, another a 1 at the beginning of each revolution. The reading circuit associated with a track containing only 1's can be considerably simpler than the circuit needed when both 1's and 0's are to be read. Thus, in Mark III,[1] the negative swing at the beginning of a 1 is amplified and used to turn on a triode that has a parallel *RLC* circuit in series with its plate. The resulting signal drives an amplifier stage that is operated with the control grid normally at ground or slightly above; the negative excursion of the voltage derived from the *RLC* circuit drives this tube to cutoff and produces a nearly rectangular positive pulse. The final stage is a cathode follower so biased that it is normally nonconducting. Only the positive rectangular pulse succeeds in passing through to the output. For timing tracks containing 0's as well as 1's, it is necessary to use more elaborate circuits designed to make the required discrimination.

Between the reading winding of the head and the input to the circuit of Fig. 11-17, there is a simple gating arrangement and a pulse transformer, as shown in Fig. 11-19. A gating voltage of -2 volts disables the gate, and about $+12$ volts is used to enable it.

With high-impedance heads it is not necessary to use pulse transformers, as shown in Fig. 11-19. If, on the other hand, low-impedance

[1] *Ibid.*, pp. 59–60.

heads of the single-turn variety are used, the output voltage is about one-thousandth the value obtained with heads like those used in Mark III.[1] The voltage step-up provided by a suitable transformer helps to reduce the gain that is required of the reading amplifier.

An even simpler reading system, usable when the 0 and 1 outputs from the heads are as shown in Fig. 11-18*a*, consists of a three-stage video amplifier followed by a fourth stage biased far below cutoff.[2] A strobe pulse reduces the bias briefly at about the time of the first peak of the head output, or about one-quarter of a period ahead of the advance pulse of Fig. 11-18*b*. If the grid of the fourth stage is being driven in the positive direction at the time of the strobe (as when a 1 has been read), a negative pulse output is obtained but, if the grid is then being driven negative, the bias reduction is insufficient to cause conduction, and no output results. An inverter following the output is necessary if 1's are to be represented by positive pulses.

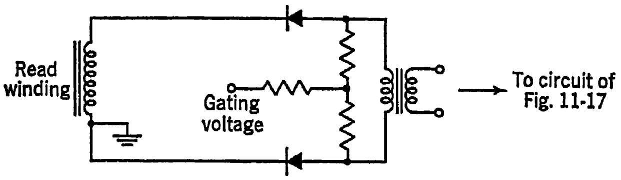


FIG. 11-19. Mark III playback-head gating.

11-6. Ultrasonic Delay Lines: Introduction. Information given as a temporal succession of electrical pulses can be preserved if it is possible to cause it to circulate about a closed path so long that the transmission time around it is greater than the duration of the succession of pulses, and provided that some means exists for restoring the pulses in amplitude and shape and thus undoing the effects of attenuation and finite bandwidth. This principle has been illustrated in Chaps. 6 and 8 as it is used in the SEAC to make the dynamic flip-flop and the registers used in the arithmetic unit. The attenuation and distortion caused by the electrical delay line are such that amplification and reclocking are required after every delay of about 5 μsec. Such a scheme obviously becomes impracticable if the delay line is to hold, say, 20 words instead of one. The introduction of a suitable delay can be accomplished by introducing into the closed loop a piece of material in which the velocity of propagation of an ultrasonic vibration assumes a reasonable value, by setting this in vibration by means of a transducer, and by using a second transducer to convert the mechanical vibration into a voltage pulse. Several devices for doing this are in use: (1) a tube filled with mercury with quartz-crystal transducers at the ends, (2) a solid rod of fused

[1] Bigelow et al., *op. cit.*, appendix, sec. 6, tables 1 to 7.
[2] McGuigan, *op. cit.*, pp. 1441–1442.

quartz, similarly excited, (3) a strip of nickel in which a magnetostrictive vibration is set up and received by small coils at the ends.

A great deal of work on liquid delay lines was done at the Radiation Laboratory at MIT and at the Bell Telephone Laboratories during World War II; so it was natural that the use of these lines as memory devices suggested itself when several computer-development projects were initiated during the war (EDVAC) and shortly afterward (SEAC, EDSAC, UNIVAC). In every case, practical memories resulted. The first fully operational machine to use such a memory was EDSAC, completed in the middle of 1949 at the Mathematical Laboratory of Cambridge University, England, by M. V. Wilkes and his associates. This was followed later in 1949 by BINAC (Eckert-Mauchly Computer Corp., now a division of Remington-Rand), in 1950 by SEAC (National Bureau of Standards), and in 1951 by the first UNIVAC (Remington-Rand). Other machines using this type of memory are the FLAC, MIDAC, DYSEAC (all advanced versions of SEAC), and the RAYDAC (Raytheon Manufacturing Company) in this country, the LEO, ACE, and MOSAIC in England, and the CSIRO Mk I in Australia. More recently, several computers have been built in England in which a section of the memory is built up of magnetostrictive delay lines. Fused quartz and other solid delay lines, though possessing excellent characteristics, have not proved popular with designers.

11-7. Mercury Delay Lines.[1] It is easy to list a number of characteristics[2,3] that should be possessed by a delay line if it is to be used as a memory device. For example,

1. The velocity of propagation should be such that the required delay can be obtained by using a line of reasonable length.
2. The bandwidth should be great enough to permit pulses of the desired width to be passed at the desired repetition rate.

[1] We content ourselves with the following general references on the theory of propagation of ultrasonic waves in mercury-filled tubes and on the structure of such devices:

H. J. McSkimin, Theoretical Analysis of the Mercury Delay Line, *J. Acoust. Soc. Am.*, vol. 20, no. 4, pp. 418–424, July, 1948.

H. B. Huntington, On Ultrasonic Propagation thru Mercury in Tubes, *J. Acoust. Soc. Am.*, vol. 20, no. 4, pp. 424–432, July, 1948.

H. B. Huntington, A. G. Emslie, and V. W. Hughes, Ultrasonic Delay Line I, *J. Franklin Inst.*, vol. 245, no. 1, pp. 1–23, January, 1948.

H. B. Huntington, A. G. Emslie, H. Shapiro, and A. E. Benfield, Ultrasonic Delay Lines II, *J. Franklin Inst.*, vol. 245, no. 2, pp. 101–115, February, 1948.

John F. Blackburn (ed.), "Components Handbook," vol. 17, MIT Radiation Laboratory Series, chap. 7, McGraw-Hill Book Company, Inc., New York, 1949.

[2] "High-speed Computing Devices," p. 342.

[3] T. K. Sharpless, Design of Mercury Delay Lines, *Electronics*, vol. 20, no. 11, pp. 134–138, November, 1949.

3. The over-all insertion loss must be small enough so that the output signal is reliably discernible amid the noise.
4. It should be possible so to terminate the line that echoes are suppressed.
5. It should be possible to compensate for the effect of ambient conditions such as temperature.
6. The lines should be cheap, rugged, reliable, and easy to manufacture.

From the standpoint of velocity of propagation, a number of liquids are satisfactory; for example, at 20°C the velocity of propagation of an ultrasonic wave in mercury is 1,450 m per sec, and in distilled water it is 1,480 m per sec.* Thus, for a mercury line, the delay time is 17.52 μsec per in., and a 1-msec delay is obtainable with a line 4 ft 9.1 in. in length. The temperature dependence of mercury in the neighborhood of 20°C is 0.000297 μsec/(μsec)(°C). That of distilled water is much greater; it falls with rising temperature, however, and vanishes at 72.5°C. Mixture with any of a variety of liquids, such as ethanol, methanol, and ethylene glycol, causes the temperature of maximum velocity to be lowered. The compensation for the effect of temperature, then, consists merely in operating the line at the temperature of maximum velocity of propagation, and this has been done, though not with lines used for computer memories.[1]

Turn now to the acoustic characteristic impedance of the media. Because of the relatively high density of mercury, its characteristic impedance at 20°C is 19.8×10^5 g/(sec)(cm^2); that of distilled water is 1.45×10^5; of quartz, 14.9×10^5. This means that the impedance match between quartz and mercury is fairly good, but that between quartz and water is very poor. The consequences are that (1) the transfer of energy from a quartz transducer to mercury and back again is much more efficient than the transfer would be if water were used as the fluid; (2) troubles from reflections are easier to handle with mercury than with water, for the reflections are much reduced if mercury is used; and (3) the large characteristic impedance of the mercury and the good impedance match load down the quartz-crystal transducer and decrease its effective Q, thus giving a much wider bandwidth than would be obtained if water was used. As quartz transducers appear to be most suitable for effecting the transfer of electrical to mechanical energy in liquid delay lines, the above considerations are rather decisively in favor of mercury vis-à-vis distilled water as a medium, and, in fact, mercury is the only liquid used in delay lines for computer memories.

Since mercury is a liquid, it can transmit only a compressional vibration. Hence it is convenient to use X-cut quartz crystals as the trans-

* Blackburn, *op. cit.*, p. 219.

[1] The numerical values given are to be found in Blackburn, *op. cit.*, chap. 7.

ducers at the two ends of the line. The thickness of the crystals is determined by the carrier frequency to be used. As the duration of the pulses to be delayed is usually $\frac{1}{2}$ μsec or less, it is clear that the carrier frequency must be of the order of 10 Mcps* so that there may be several cycles per pulse.

The attenuation due to transmission through the mercury consists of two terms:† (1) the "free-space" attenuation, proportional to the square of the frequency, and (2) the attenuation due to the finite diameter of the tube that holds the mercury, proportional to the square root of the frequency. Both these terms, however, are much smaller than the loss due to mismatch between transducer and transmitting medium. Considering all these contributions to the over-all loss, Huntington calculates, as an example, that for a 1-msec delay line with a 15-Mcps carrier the over-all insertion loss is 51.5 db, of which 36 db is due to mismatch, 12.8 db to free-space attenuation, and 2.7 db to tube effects.[1] This loss, though considerable, still permits a sufficiently large signal to be obtained at the output, given a reasonable excitation of the input.

Another frequency-dependent effect is the directivity of the radiation. If it is assumed that the mercury fills all the space to one side of an infinite-plane baffle and that the excitation is furnished by a circular piston of diameter d moving back and forth in a hole in the baffle, the formula for the half-angle θ of the cone subtended by the first minimum in the free-space diffraction pattern at large distances is[2]

$$\sin \theta = 1.22 \frac{\lambda}{d} \tag{11-1}$$

Since, for a frequency of 10 Mcps and an effective crystal-radiating diameter of 2 cm, the quantity in the right member of Eq. (11-1) assumes the value 8.85×10^{-3}, the directivity is very good indeed. This opens other possibilities than delay lines, consisting simply of narrow tubes (say, 1 in. or less in diameter). The UNIVAC mercury memory consists of seven tanks in each of which there are 18 parallel channels.[3] The RAYDAC memory consists of a number of small tanks, and the required path length is obtained by multiple reflections.[4] The chief drawback of the very good directivity is that the transmitting and receiving crystal

* For example, in EDSAC, 13.5 Mcps; in SEAC, 8 Mcps.

† Blackburn, *op. cit.*, chap. 7, pp. 220 and 221.

[1] *Ibid.*, chap. 7, pp. 235–236.

[2] *Ibid.*, chap. 7, p. 220.

[3] J. P. Eckert, A Survey of Digital Computer Memory Systems, *Proc. IRE*, vol. 41, no. 10, pp. 1393–1406, October, 1953.

[4] Huntington, Emslie, Shapiro, and Benfield, *op. cit.;* on p. 109 the possibility of such tanks is briefly discussed. Apparently the first successful tanks of this kind were built by Raytheon.

surfaces must be very well aligned; thus, for the EDSAC lines, in which $f = 13.5$ Mcps and $d = \frac{5}{8}$ in., it is stated that the tolerance is but 3 minutes of arc.[1]

It has been stated above that the temperature coefficient of the delay time of mercury is very nearly 0.0003 μsec/(μsec)(°C), and that the delay increases with temperature. This means that if, for example, it is necessary to maintain a delay of 1,000 μsec accurate to within ± 0.1 μsec, the temperature variation from the correct value must not exceed ± 0.33°C. Therefore, the temperature must be carefully controlled, particularly if a fixed-frequency clock is used. Probably the simplest method is to operate the mercury lines at a temperature considerably above ambient. Thus, in the SEAC,[2] the clock is a crystal-controlled oscillator, the lines are 384 μsec long, and the mercury lines are carefully bonded thermally so that the whole memory system forms, for all practical purposes, an isothermal unit maintained by thermostatic control at a temperature of 50 ± 0.25°C. An alternative approach[3,4] is not to attempt to control the absolute temperature too precisely but to ensure only that all the lines are at the same temperature within the required tolerance and to control the clock frequency so that the ratio of line delay to pulse-repetition period remains constant. In one scheme for accomplishing this,[4] one line is assigned to the frequency-control system. A frequency divider produces one pulse per minor cycle. The total delay time is to be 16 minor cycles. The minor-cycle pulses follow two paths; that is, (1) they are used as inputs to the delay line, from which they pass to a comparison circuit, and (2) they are delivered directly to the comparison circuit. The output from the comparison circuit is a pulse whose width is proportional to the error in delay time; this is used (after some processing which need not be discussed here) to drive a reactance tube that regulates the clock frequency. In UNIVAC I, a somewhat similar system is used to regulate the current through the heating coils, and the clock frequency is held constant.

There are in use a number of different physical forms of delay line. In the EDSAC the mercury is held in steel tubes, in the SEAC in glass tubes, in UNIVAC I in multichannel steel tanks, and in the RAYDAC in box-shaped steel tanks. Of course, crystal mountings and other details vary widely also.

The matter of crystal mounting is intimately connected with the sup-

[1] M. V. Wilkes and W. Renwick, An Ultrasonic Memory Unit for the EDSAC, *Electronic Eng.*, vol. 20, no. 245, pp. 208–213, July, 1948.

[2] S. Greenwald, R. C. Haueter, and S. N. Alexander, SEAC, *Proc. IRE*, vol. 41, no. 10, pp. 1300–1313, October, 1953.

[3] Wilkes and Renwick, *op. cit.*, p. 209.

[4] J. M. M. Pinkerton, Automatic Frequency Control, *Electronic Eng.*, vol. 23, no. 278, pp. 142–143, April, 1951.

pression of reflections. Since a crystal at resonance acts as a half-wave section, no reflection will occur if it has mercury on both sides of it.[1] A possible method of mounting (now superseded) is shown in Fig. 11-20. The slanting end plate is of glass and has the function of breaking up and dissipating the beam so that no energy is reflected back to the crystal. In the prototype of the UNIVAC memory tank,[2] a steatite backing was used for the crystals. The steatite was silvered, one surface of the crystal was silvered, and the two were soldered together. The other end of the steatite support was not parallel to the one soldered to the crystal, presumably so that reflection would not be directed back to the crystal. The match of crystal to steatite is very good, and reflections were effectively eliminated.

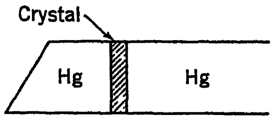


FIG. 11-20. Possible manner of mounting crystal.

The electronic circuits associated with a mercury delay line are given in block-diagram form in Fig. 11-21,[3] which shows the EDSAC memory loop. The clock pulses are 0.9 μsec long, occurring at a pulse-repetition rate of 514 kcps. The oscillator frequency is 13.5 Mcps; the circuit is

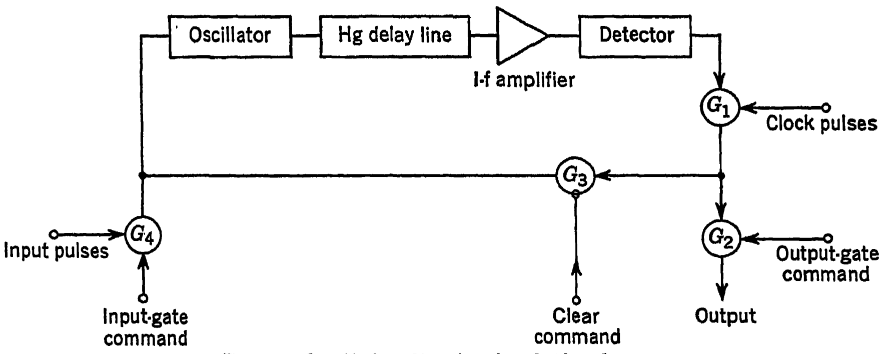


FIG. 11-21. Delay line in circulation loop.

so arranged that oscillation occurs only while a positive pulse is applied to the input. The straightforward i-f amplifier amplifies the attenuated and distorted wave packets that come from the delay line. The pulse envelopes are recovered by the detector and used to gate standard clock pulses into the loop. These can be read out via G_2 when an output-gate

[1] Blackburn, *op. cit.*, p. 224.

[2] I. L. Auerbach, J. P. Eckert, R. F. Shaw, and C. B. Sheppard, Mercury Delay Line Memory Using a Pulse Rate of Several Megacycles, *Proc. IRE*, vol. 37, no. 8, pp. 855–861, August, 1949. A similar scheme was recommended by T. K. Sharpless, Design of Mercury Delay Lines, *Electronics*, vol. 20, no. 11, pp. 134–138, November, 1947.

[3] Wilkes and Renwick, *op. cit.*, p. 209; Auerbach, Eckert, Shaw, and Sheppard, *op. cit.*, p. 857.

command is given. The gate G_3 ordinarily transmits pulses unless it is inhibited by a "clear" command; it is by this means that information is "erased" from the memory. The gate G_4 is used to introduce new information into the memory while G_3 is inhibited. Notice that the scheme of using the distorted pulses from the delay line to gate standard pulses into the loop reduces the bandwidth requirements on the delay line. This procedure is standard in mercury memories.

The delay-line lengths, pulse-repetition frequencies, and pulse lengths of existing mercury-memory machines will be of some interest to the reader. Several of these machines, for example, ACE, EDSAC I, and

Computer	Delay, μsec	Prf	Pulse length, μsec
EDSAC I, LEO	1,100	514 kcps	0.9
SEAC, EDVAC	384	1 Mcps	0.5
UNIVAC I	404	2.25 Mcps	Not given, but presumably one-half the pulse-repetition period or less
RAYDAC	305	3.77 Mcps	
MOSAIC	1,024	570 kcps	
CSIRO Mk I	1,024	333 kcps	
ACE (pilot model)	1,024	1 Mcps	

UNIVAC, also use delay lines of one word length as registers in the arithmetic and control units.

11-8. Solid Delay Lines. During World War II a great deal of work was done on solid delay lines.[1] Unfortunately, progress was neither so rapid nor so satisfactory as in the work with liquid delay lines, and solid lines were, therefore, not available for use when the computer-development programs that led to such machines as EDVAC, UNIVAC, and SEAC were initiated. More recently, successful solid lines have been constructed, but they have not yet found general favor with computer designers, though they have been used in some experimental equipment. A variety of materials have been suggested and have been the subject of experiment: vitreous silica, magnesium alloys, aluminum, and various glasses.[2] Vitreous silica appears at the moment to be the best material, though work on some glasses seems to be very promising.

In a liquid-filled tube or tank it is possible to propagate only a single mode of vibration: the longitudinal compressional vibration. In solids,

[1] This work is described, the theory is reviewed, and a considerable bibliography is given in D. L. Arenberg, Ultrasonic Solid Delay Lines, *J. Acoust. Soc. Am.*, vol. 20, no. 1, pp. 1–26, January, 1948.

[2] A list of these materials, together with their densities and velocities of propagation of both longitudinal and shear waves, is given by F. A. Metz and W. M. A. Andersen, Improved Ultrasonic Delay Lines, *Electronics*, vol. 22, no. 7, pp. 96–100, July, 1949.

the situation is quite different. Transverse (i.e., shear) vibrations can be propagated and, in fact, offer considerable advantages. Whereas compression vibrations are subject to mode conversion upon reflection, which may result in the reception of a main pulse trailed by several spurious pulses, shear vibrations are not.[1] Furthermore, the shear-mode beam spreads less in the course of transmission and results in an improved signal-to-noise ratio, since only a portion of the noise originating in the medium affects the crystal transducer at the receiving end.[2] For these reasons, the shear mode is the mode chiefly used.

Of all the possible materials for the construction of solid delay lines, vitreous silica is the one upon which the most experimental work has been done and which alone has resulted in commercial packaged lines. Extruded magnesium rods have given promising results in experimental work. However, it proved to be difficult to cement the crystal transducers to the ends of the rods properly and the effect of improper attachment was a serious reduction in bandwidth.[3] Pressure mounts for the crystals have been used experimentally[3] but do not appear to be practically satisfactory.

In more recent work at the National Bureau of Standards,[4,5] short lines were made of certain isoelastic alloys, that is, alloys that exhibit a constant modulus of elasticity over a wide temperature range. It was found that certain alloys of iron, nickel, and chromium with various minor constituents possessed a temperature coefficient of delay time of only 8 ppm per °C over the range from -50 to $+200$°C, as compared to 300 ppm per °C for mercury. A satisfactory method of cementing quartz transducers to the ends was also found. Unfortunately for many computer applications, the attenuation in the alloys studied is rather high; lines about 6 in. long giving a delay of 50 μsec cause an insertion loss of 50 db or more. This in effect limits these lines to such uses as in one-word registers in control and arithmetic circuitry. There remains the possibility of finding materials with the same good properties and less losses or of reducing attenuation by using barium titanate transducers, which will be discussed later on in this section. As metal lines in general have excellent bandwidth characteristics, a low-loss metal line of this type should prove very useful indeed.

Vitreous silica has been studied very extensively, and, as mentioned

[1] M. D. Fagen, Performance of Ultrasonic Vitreous Silica Delay Lines, *Tele-Tech.*, March, 1952, pp. 43ff.

[2] Metz and Andersen, *op. cit.*, p. 98.

[3] *Ibid.*, p. 99.

[4] Ultrasonic Metal Delay Lines, *Tele-Tech.*, vol. 13, no. 3, pp. 78 and 178–179, March, 1954 (no author).

[5] A. Voznak, A Temperature Invariant Solid Ultrasonic Delay Line, *NBS Rept.* 2785 (*OED Rept.* 17-277), Sept. 25, 1953.

above, delay lines made of this material are commercially available from several manufacturers. Vitreous silica is a clear, glasslike product of silicon dioxide that results from the fusion of crystal quartz, though it can also be produced by the reduction of other silicon compounds.[1] It is available in the form of rods, blocks, or plates. It possesses the lowest attenuation of all the solid materials suitable for high-frequency work. The attenuation is proportional to the frequency, the factor being 0.08 db/(ft)(megacycle) in the shear mode and 0.05 db/(ft)(megacycle) in the longitudinal-compression mode. The velocity of propagation, measured at 25°C using a 15-Mcps shear wave, is very nearly 3.776×10^5 cm per sec; measurement at the same temperature, using a 60-Mcps compressional wave, gave a value of 5.957×10^5 cm per sec. The temperature coefficient of velocity for shear waves of frequency 15 Mcps has been measured in a variety of specimens and runs to about 70 ppm per °C in the range from 25 to 70°C. Specimens produced by different processes tend to show differences both in velocity of propagation and in temperature coefficient.

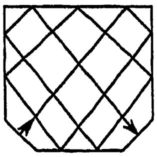


FIG. 11-22. Multiple-reflection delay line.

From the velocity of propagation quoted above, it appears that a 100-μsec delay line in the form of a rod must be 37.76 cm in length. Rods, therefore, become too long if delays of several hundred microseconds are required, and economy of space dictates some scheme relying upon multiple reflections of the ultrasonic beam within a block of the material. Fortunately, shear waves polarized perpendicular to the plane of incidence are reflected at an angle equal to the angle of incidence and do not suffer mode conversion upon reflection. Using a plate of the general form shown in Fig. 11-22, which can be packaged in a container 1 by 6 by 6 in., it is possible to obtain a delay of 1,000 μsec.[2,3] The transducers, Y-cut quartz crystals, are mounted at the two cut-off corners. Polygonal plates are also in use.

[1] E. S. Pennell, Vitreous Silica for Ultrasonic Delay Line Applications, *Proc. Natl. Electronics Conf.*, vol. 8, pp. 799–810, 1952. All the constants of the material given in this paragraph are taken from this source.

[2] Ultrasonic Delay Lines, in "Electrons at Work," *Electronics*, vol. 26, no. 7, pp. 210–216, July, 1953.

[3] T. F. Rogers and W. A. Andersen, Some Recent Researches on Ultrasonic Propagation in Solid Media, *Proc. ACM*, Pittsburgh, May 2–3, 1952, pp. 203–205.

For both rods and multiple-reflection lines, the transducers are usually quartz crystals if the shear mode is to be used. It has been shown[1] that, if the longitudinal mode can be used, transducers made of barium titanate ($BaTiO_3$) ceramic offer a considerable improvement in over-all attenuation; this is true for solid lines in general.

Experimental results on vitreous silica lines have been published by May[1] and by Fagen.[2] Both investigators gave results on a 670-μsec line, the transmitted signal being a 10-Mcps pulsed carrier. The transducers were Y-cut quartz crystals, the capacitance of the output crystal tuned out by an appropriate coil. Working into a load of 75 ohms, a bandwidth of 3.2 Mcps with an insertion loss of 52 db was obtained. When the load was increased to 1,000 ohms, the bandwidth fell to 2 Mcps and the insertion loss to 33 db. May also gives comparative figures on multiple-reflection lines of the same length using Y-cut quartz-crystal and barium titanate transducers, both transmitting a 15-Mcps pulsed carrier. With the crystal, using a termination of 464 ohms, a bandwidth of 4 Mcps and an insertion loss of 54 db were obtained; with the barium titanate, using a

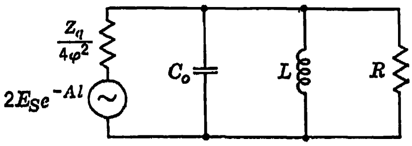


FIG. 11-23. Mid-band equivalent circuit for line with quartz-crystal transducers. (*J. E. May, Proc. Natl. Electronics Conf., vol.* 9, *p.* 265, 1953.)

termination of 50 ohms, a bandwidth of 3 Mcps was obtained with an insertion loss of only 28 db.

Note that, in general, the smaller the termination, the broader the bandwidth and the greater the loss. This is very easy to see from the mid-band equivalent circuits; Fig. 11-23 applies in the case of the line with quartz-crystal transducers. Here E_S is the driving voltage, A an attenuation factor, Z_q the mechanical impedance of the vitreous silica, ϕ a constant proportional to the piezoelectric constant of the transducers, C_0 the shunt capacitance of the output transducer, L the inductance of the coil used to tune out C_0 at resonance, and R the load. An equivalent circuit has also been developed for use with the barium titanate transducer. These equivalent circuits are all based upon the fundamental researches of W. P. Mason of the Bell Telephone Laboratories. The interested reader will find a list of Mason's papers and books on the subject in May's paper.

[1] J. E. May, Characteristics of Ultrasonic Delay Lines Using Quartz and Barium Titanate Ceramic Transducers, *Proc. Natl. Electronics Conf.*, vol. 9, pp. 264–277, 1953.

[2] M. D. Fagen, Performance of Ultrasonic Vitreous Silica Delay Lines, *Proc. Natl. Electronics Conf.*, vol. 7, p. 380, 1951; also *Tele-Tech* article already cited.

11-9. Magnetostrictive Delay Lines. Certain metals (e.g., nickel and Monel metal) possess the property of changing dimensions when immersed in a magnetic field. Conversely, if a piece of the metal is immersed in a magnetic field and is subjected to a strain, the flux through the metal is changed. These effects are known, respectively, as the Joule and Villari magnetostrictive effects.[1]

Consider a wire ribbon or tube of magnetostrictive material provided with two coils mounted near the ends and with a permanent magnet that produces a flux through the portion of the material within the second coil, as shown in Fig. 11-24.[2-4] If a pulse of current flows through the transmitting coil, the portion of material lying within it suffers a momentary expansion or contraction, depending upon the material used. This deformation is transmitted as a compression wave along the wire or tube in both directions. The "direct" wave (that starting toward B), when it passes through the receiving transducer, causes a change in the flux linking the coil and, hence, a voltage that appears across its terminals. The

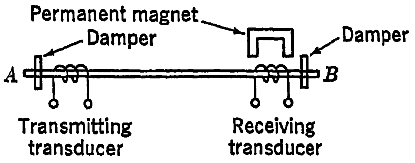


FIG. 11-24. Magnetostrictive delay line.

wave that starts toward A is reflected there and then proceeds toward B; it also causes a voltage pulse to appear across the receiving transducer. A usable memory device can be obtained if some means of suppressing the reflections can be found. Fortunately, the necessary suppression can be obtained in a variety of ways, by coating the ends of the line with a viscous mixture, e.g., a beeswax compound[5] or grease,[6] or by clamping them between damping pads.

The velocity of propagation of the wave in nickel is somewhat more than three times as great as the velocity in mercury; it is 1.58×10^4 ft per

[1] A complete theoretical account of magnetostriction is given by W. P. Mason, "Electromechanical Transducers and Wave Filters," D. Van Nostrand Company, Inc., Princeton, N.J., 1942.

[2] E. M. Bradburd, Magnetostrictive Delay Line, *Elec. Commun.*, vol. 28, no. 1, pp. 46–53, March, 1951.

[3] T. B. Thompson and J. A. M. Lyon, Analysis and Application of Magnetostriction Delay Lines, *IRE Trans. on Ultrasonics Eng.*, vol. PGUE-4, pp. 8–22, August, 1956.

[4] J. W. Fairclough, A Sonic Delay-line Storage Unit for a Digital Computer, *Proc. Conv. on Digital Computer Techniques*, Institution of Electrical Engineers, London, Apr. 9–13, 1956.

[5] Bradburd, *op. cit.*, pp. 50-51.

[6] H. Epstein and O. Strom, Magnetostrictive Sonic-delay Line, *Rev. Sci. Instr.*, vol. 24, no. 3, pp. 231–232, March, 1953.

sec at 60°C.* Hence one might expect that a delay line of 1,000 μsec would be inordinately long. A thin ribbon or wire, however, is quite adequate and, as it can be coiled up in a fairly small space without deterioration in performance, it can be made sufficiently compact.[1]

Ordinarily, the input to the transmitting transducer is a video pulse. If the wavelength of the ultrasonic wave in the line is long compared with the length of the transducer coils, the waveforms shown in Fig. 11-25 are obtained; each is proportional to the derivative of the one above.[2] It is still possible to obtain a usable output if the wavelength is as small as twice the length of the coil.[3] Hence operation with microsecond pulses at a repetition rate of about 500 kcps can be obtained using coils that are not inconveniently small.

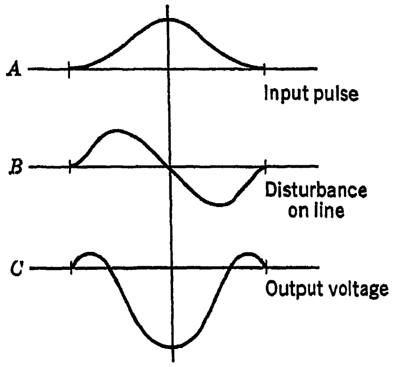


FIG. 11-25. Magnetostrictive-delay-line waveforms.

Given an output as shown in Fig. 11-25 or worse, it is clear that reshaping is necessary before the output is fed back to the input. This is accomplished (as in the case of mercury lines) by using the amplified output to gate a standard pulse into the input. The recirculation loop is logically the same as that used with a mercury line except that no detector or oscillator is included; the oscillator is replaced by an amplifier that has in its plate circuit a transformer which drives the transducer.

The transducers are coils of several hundred turns of fine wire (for example, No. 40). The usual practice is to enclose the transducer coil in a magnetic shield to reduce stray magnetic field and to ensure that the field is confined as well as possible to the metal of the line.

The temperature coefficient of delay in nickel is 0.00013 sec/(msec)(°C),

* Bradburd, *op. cit.*, p. 48.

[1] A. E. De Barr, R. Millership, P. F. Dorey, R. C. Robbins, and P. D. Atkinson, Digital Storage Using Ferromagnetic Materials, *Proc. Assoc. Computing Machinery* (Pittsburgh meeting, May 2-3, 1952), pp. 197–202.

[2] Bradburd, *op. cit.*, p. 68.

[3] Epstein and Strom, *op. cit.*, p. 231.

which means that the temperature-regulation problem is not so severe as it is with mercury lines. Temperature variation can be almost completely eliminated by replacing the section of the line used only for transmission (that is, the part between the transducers) by a piece of material chosen to have a suitably low temperature coefficient of delay and yet to present a fairly good acoustic-impedance match with nickel. Apparently this can best be done if a torsional mode of vibration is used instead of the longitudinal mode considered here; this naturally demands a considerably more complicated transducer.

Attenuation is not especially severe and is due almost entirely to losses in the transfer of energy between the transducers and the line. A signal-amplitude insertion loss of 40 db has been measured[1] for a 100-μsec line, and lines of between 1 and 2 msec have shown insertion losses between 40 and 60 db.

Magnetostrictive delay lines are rugged, cheap, and compact. An additional virtue (at least in some applications) is that the delay can be varied continuously merely by changing the position of the receiving transducer. Their chief drawback is their frequency response, which is not so good as that of, e.g., mercury lines. However, a leading manufacturer[2] offers lines of delay up to 500 μsec that can be operated at pulse-repetition frequencies up to 1 Mcps, and longer lines that can be operated at rates up to 500 kcps.

Magnetostriction lines have not been used to any extent in computers built in this country but have been used quite extensively in England, in machines built by the Elliott Brothers Research Laboratories[3,4] and by the Ferranti Electric Company.

11-10. Conclusion. This concludes the treatment of memory devices of the circulating type. Electromagnetic delay lines, which have been treated in the chapter on registers, will not be considered here, for they are essentially sufficiently cheap only for use in holding a single word. The next chapter will be concerned with true random-address memories, in which all addresses are equally available with the same access time.

[1] *Ibid.*, p. 232.

[2] Ferranti Electric Ltd.

[3] De Barr et al., *op. cit.*

[4] R. Millerslip, R. C. Robbins, and A. E. De Barr, Magnetostriction Storage System for a High Speed Computer, *Brit. J. Appl. Phys.*, vol. 2, p. 304, 1951.

CHAPTER 12

LARGE-SCALE MEMORY DEVICES II

12-1. Random-access Memories. In Chap. 11 two types of memory device were discussed: (1) the magnetic type that requires relative motion between the medium and the reading and writing heads, and (2) the delay-line type. In both of these, the information held is continually in transit, and a specific word can be recovered only periodically. In magnetic drums and disks, the period is determined by the speed of rotation and the number of reading heads per channel; in delay lines it is determined by the length and material of the line. On the average, there is a delay of one-half the period between the instant when a demand is made upon the memory device for a specific word and the instant when it is delivered. This fact imposes a limitation upon the speed of operation of the whole machine. It can be argued that "minimum-access" or "optimum" coding, by properly arranging the information in the memory so that each word becomes available when required, can in fact vastly reduce effective access time, but this imposes an additional burden upon the already sufficiently harassed writer of codes. If speed is important, it is better to use a memory of truly short access time, such that all words are equally available at all times, the delay between demand and delivery being essentially the same for all words. Memories of this type are true "random-access" memories, and only these. Several types will be described in this chapter; in all cases the access time is 20 μsec or less. These are electrostatic memory tubes, arrays of magnetic cores, arrays of ferroelectric cells, and arrays of ordinary capacitors. The first and last of these are volatile and, hence, imply the complication of circuitry required for periodic regeneration of the contents of the memory; the second and third are not volatile, although, in both, the simplest method of reading destroys the information held and this must be immediately restored.

12-2. Electrostatic Memory Tubes: General. Early in the history of the development of modern electronic digital computers, the advantages of short-access-time random-access memories were clearly realized. A scheme that had considerable appeal was to store information as minute spots of electrostatic charge upon a dielectric surface within a vacuum tube. Reading and writing could be accomplished by means of an electron beam provided by a conventional electron gun. The basic process of secondary emission had long been known, and tubes for various types of information storage, although not for the storage of coded information to which access could be had at random, had already been under development before and during the war.[1,2] Hence it seemed likely that adequate computer-memory tubes would shortly be forthcoming. This did not turn out to be the case. None of the development programs had yet resulted in satisfactory production models when, in 1949, F. C. Williams and T. Kilburn published a method of using a conventional cathode-ray tube (such as is commonly used in oscilloscopes) as a memory tube.[3] A parallel and independent development in this country was published about a year later.[4] Williams's cathode-ray-tube memory was the first to be incorporated in a complete computer.

In the scheme first used by Williams, information was inserted and recovered serially. This mode of operation imposes the least severe requirements upon the capabilities of the tube, but it was quite clear from the beginning that the principles were equally adaptable to the construction of a true random-access memory, and this was at once undertaken by J. H. Bigelow at the Institute for Advanced Study (IAS) in Princeton, N.J. This development was successful, and before the end of the year 1951 the IAS machine and a version of it built at the University of Illinois were successfully operating with random-access or "parallel" memories, and the Ferranti Mk I, based upon Williams's original work at the University of Manchester, was successfully using a serial memory. Since then a number of computers based upon the original IAS work have been successfully completed, and also the BESK in Sweden, a computer at the Radar Research Establishment in England, and, last but not least, the IBM 701, of which about twenty were produced before it was superseded by the 704. In all these memories electrostatic deflection tubes were used.

Because of the widespread successful application of the Williams scheme some space will be devoted to it here. This type of memory, however, is now obsolete as a result of the development of magnetic-core arrays for random-access memory. No treatment of the other electrostatic memory tubes, such as the Selectron or the tube developed in the

[1] M. Knoll and G. Kazan, "Storage Tubes and Their Principles," John Wiley & Sons, Inc., New York, 1952.

[2] Britton Chance, F. C. Williams, V. W. Hughes, D. Sayre, and E. F. MacNichol, Jr., "Waveforms," chap. 9, McGraw-Hill Book Company, Inc., New York, 1949.

[3] F. C. Williams and T. Kilburn, A Storage System for Use with Binary Digital Computers, *Proc. IEE*, vol. 96, pt. 2, no. 81, pp. 183–200, March, 1949.

[4] J. P. Eckert, H. Lukoff, and G. Smoliar, A Dynamically Regenerated Electrostatic Memory System, *Proc. IRE*, vol. 38, no. 5, pp. 498–510, May, 1950.

Digital Computer Laboratory at MIT, will be given, for these seem to be of historical interest only. The reader who wishes to inform himself about these developments will find an excellent bibliography in the book by Knoll and Kazan to which reference has already been made.

12-3. Secondary Emission.[1] Consider a well-focused beam of electrons normally incident upon a target of dielectric material (in a Williams tube this will be the phosphor) in an evacuated tube, and suppose for simplicity that all the electrons in the beam are of the same energy, which is determined by the potential rise from the cathode to the last anode. Some of the electrons are reflected by collision with the atoms of the material; others penetrate the material, and some of them sufficiently excite atoms so that they in turn emit electrons, some of which succeed in escaping from the surface. All electrons leaving the target, including those merely reflected, are to be considered "secondary electrons," and the ratio of the total electron current leaving the target to the current of the incident beam is designated as the "secondary-emission ratio."

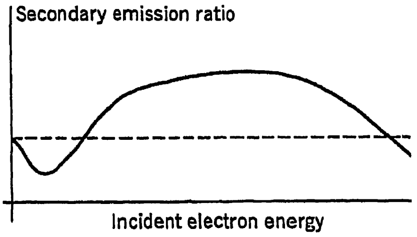


FIG. 12-1. Typical secondary-emission curve.

For a given target material, the secondary-emission ratio is a function of the energy of the incident electrons and, for all materials, it has the general form shown in Fig. 12-1. At very low energies practically all the electrons are reflected. For slightly higher energies, some penetrate the material but very few succeed in causing secondary emission by sufficiently exciting atoms, and the ratio falls to a minimum. As the energy of the incident electrons is increased beyond this point, more and more atoms become sufficiently excited to emit secondaries, and the ratio rises until a maximum in excess of unity (e.g., perhaps about 2) is reached. The maximum is fairly flat. As the electron energies are increased still further, the penetration becomes deeper and deeper, with the result that more and more secondaries are reabsorbed before they succeed in escaping from the surface, and the secondary-emission ratio steadily falls to unity and then below. For the phosphors generally employed, energies in the range 1,000 to 2,000 volts correspond to values of the ratio near the maximum.

The energy distribution of the secondaries is shown in Fig. 12-2. The abscissa is the energy, and the ordinate is the number of secondaries emitted per unit increment of emission energy. The first maximum

[1] A vast literature on this subject exists. A convenient and easily understood account is contained in Knoll and Kazan, *op. cit.*, pp. 1–18. A lengthy bibliography is given in pt. 3, pp. 131–140.

occurs at an energy of 1 volt or so; the second maximum is at V_i, the energy of the incident electrons, and is due to reemission resulting from collision and elastic reflection. As V_i is decreased, the first maximum decreases in height, but does not shift left or right.

12-4. Theory of the Williams Memory Process. The original theory due to Williams and Kilburn has been modified to take account of the work of other investigators.[1–4] Williams's revised theory[5,6] and the account given by Holt and Davis[7] are in substantial agreement and form the basis for the presentation to follow. Actually, the main features of the original theory are preserved in the revision.

Consider a cathode-ray tube with the accelerating anodes maintained at a potential sufficiently high above the cathode so that the electron beam striking the phosphor produces a secondary-emission ratio well above unity and in the neighborhood of the maximum value; this requires that the potential difference be in the range 1,000 to 2,000 volts. It is assumed that there is a third anode consisting of the inner coating toward the front of the tube; this is at the highest potential, and the second anode is held somewhat below it. A common arrangement is to ground the highest-potential electrode and operate with the cathode negative; this will be assumed for the sake of fixing the ideas. The tube is provided with an exterior pick-up plate, usually of fine wire gauze, mounted so that it is in contact with the front surface of the tube and connected to a resistor the other end of which is grounded. It is assumed that initially the control grid is held below cutoff, that the phosphor surface is at ground potential, that some definite set of voltages is applied to the deflecting

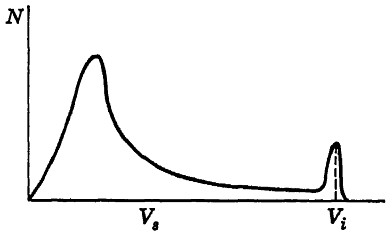


FIG. 12-2. Energy distribution of secondary electrons.

[1] L. Brillouin, Theory of the Williams Tube, *11th Ann. Conf. on Physical Electronics*, MIT, Mar. 29–31, 1951.

[2] W. B. Nottingham, Bombardment of Insulating Surfaces by Electron Beams, *11th Ann. Conf. on Physical Electronics*, MIT, Mar. 29–31, 1951.

[3] J. Kates, Space-charge Effects in Cathode-ray Storage Tubes, Ph.D. thesis, University of Toronto, 1951.

[4] A. W. Holt and J. H. Wright, Progress Report on Electrostatic Storage Research, *NBS Rept.* 1082, July 25, 1951.

[5] F. C. Williams, T. Kilburn, C. N. W. Litting, D. B. G. Edwards, and G. R. Hoffman, Recent Advances in Cathode-ray-tube Storage, *Proc. IEE*, pt. 2, vol. 100, no. 77, pp. 523–539, October, 1953.

[6] C. N. W. Litting, The Physics of Cathode Ray Storage Tubes, *J. Sci. Instr.*, vol. 31, no. 10, pp. 351–356, October, 1954.

[7] A. W. Holt and W. W. Davis, Computer Memory Uses Conventional C-R Tubes, *Electronics*, vol. 26, no. 12, pp. 178–182, December, 1953.

plates (assume an electrostatic-deflection tube), and that appropriate focusing voltages are applied to produce a well-focused beam.

Now suppose that the control-grid voltage is suddenly raised above cutoff. The electron beam strikes the phosphor and causes secondary emission with a ratio above unity; more electrons leave the spot than arrive, and the spot begins to charge positively.

It has been seen in connection with Fig. 12-2 that a large proportion of the emitted secondaries are of quite low energy. Initially there is no field urging them toward the third anode (or "collector"). On the other hand, the electrons in the main beam, with the secondaries that are still in transit in the space before the phosphor, constitute a space charge that produces a field urging newly emitted secondaries toward the surface of the phosphor. In addition, as the bombarded spot begins to charge positively with respect to the surrounding area and, hence, to rise in potential, it exerts an attractive force. Hence, of the secondary electrons, (1) some reach the collector, (2) some are pulled back to the spot, thus reducing the effective secondary-emission ratio, and (3) some are driven to the surface of the phosphor, thus lowering its potential; these electrons are of such low energy that it is assumed that they produce negligible secondary emission. Because of the attraction of the spot, many of those driven to the surface are deposited in a ring about the spot.

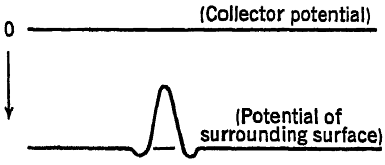


FIG. 12-3. Potential distribution of bombarded spot and surroundings (not to scale).

This whole process tends toward equilibrium; for (1) as the spot potential rises relative to its surroundings, more and more secondaries are pulled back, and the effective secondary-emission ratio is reduced toward unity; and (2) as the surface falls in potential below the collector, a field is created which counteracts the field of the space charge, and so more secondaries reach the collector and fewer reach the phosphor surface. In the equilibrium condition finally reached, (1) the surface of the phosphor is at a potential below that of the collector (which has been taken as ground), (2) the spot potential is above that of the phosphor surface, and (3) a space-charge-limited current equal to the beam current is flowing from the spot to the collector (the situation is like that in a triode whose cathode, anode, and control grid are the spot, the collector, and the phosphor surface, respectively). All this is summed up in Fig. 12-3. As for actual values, the spot equilibrium potential is generally agreed to be 2 or 3 volts positive with respect to the surrounding area, and the surface potential depends quite strongly on tube type, accelerating potentials, and beam current. Values in the range -5 to -10 volts have been measured at Manchester[1] for a beam current of 0.2 μa and accelerating potentials of 1,000 to 1,400 volts; considerably larger values have been found at the National Bureau of Standards (NBS), using a beam current of 1 μa.* The actual time required to establish the total equilibrium is not known with precision; it is of the order of a second at most and may possibly be a great deal less. Once the surface potential has been established, the establishment of equilibrium spot potential at a new location is accomplished in a time of the order of one-half a microsecond.

In the light of the discussion just given, it is simple to deduce the nature of the pick-up-plate-to-ground voltage during the process of initial beam turn-on and turnoff. Most obvious is the fact that the raising of the bombarded spot to the positive equilibrium potential is in effect the charging of a small capacitance, during which negative charge moves from ground to the pick-up plate and causes a pulse positive with respect to ground to appear at this plate. Another effect is due to the introduction of the negative space charge at beam turn-on and its disappearance at turnoff. These events, respectively, cause electrons to depart from and to return to the pick-up plate and, therefore, cause a small negative pulse and a small positive pulse to appear at the pick-up plate.[2,3] The sum of these signals is the total pick-up-plate signal, as shown in Fig. 12-4.

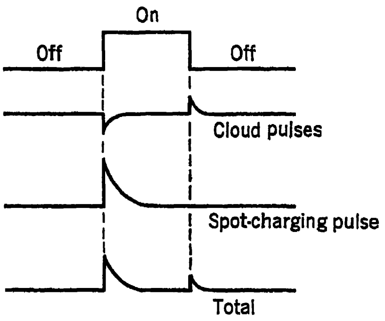


FIG. 12-4. Pick-up-plate pulses at first beam turn-on.

The pulse due to spot charging is very slightly delayed in its leading edge behind the cloud pulse because of the finite velocity of the incident electrons. There has been a good deal of argument about the existence of the cloud pulses, but they were finally directly observed in an ingenious experiment.[4]

If the beam is turned off, the charge on the spot tends to leak off; but, if nothing else were taking place within the tube, several seconds would elapse before any serious diminution of the charge took place. If, at some time before leakage has had much influence, the beam is turned on again and directed to the same spot, it is found that only a very small spot-charging pulse appears at the pick-up plate, and the total signal

[1] Williams, Kilburn, Litting, Edwards, and Hoffman, *op. cit.*, p. 536.

* Holt and Wright, in the NBS report already cited, give a family of curves showing the variation of this value with beam current and accelerating potential.

[2] Williams and Kilburn, *op. cit.*, p. 187.

[3] Eckert, Lukoff, and Smoliar, *op. cit.*, p. 501.

[4] Williams, Kilburn, Litting, Edwards, and Hoffman, *op. cit.*, pp. 528–529.

there is essentially the signal due to the cloud effect; at the end of this second beam turn-on, the original charge of the spot is restored.

In practice, the beam is being continually directed to different spots and continually switched on and off. The secondaries returned to the screen by the field of the space charge cause the charge patterns created at the various points to deteriorate much more rapidly than they would because of mere leakage, and, for this reason, the whole pattern must be regenerated fairly often, certainly several times a second. The details of this process will be discussed below.

So far it has been shown how to produce one sort of charge (and potential) distribution. To record binary information a second distribution is required. There are a great many possible ways of producing this; several are to be discussed, the first being the "double dot" as contrasted with the "dot" or single charged spot treated above.

Suppose that the beam has been on long enough for equilibrium to be established, and consider a single charged spot, which exhibits the potential distribution of Fig. 12-3. Let the deflecting voltages be varied slightly so that the beam is directed to a spot whose center is one spot diameter to the right of the center of the existing spot, and then let the beam be turned on. The process of spot charging certainly takes place; one effect of it is that some of the secondaries that normally return to the phosphor in a ring about the newly charged spot must alight upon the first spot, partly destroy its positive charge, and hence make its potential more negative. If the beam were left on for a long time, the charge of the first spot might be completely neutralized; suppose instead that the beam is left on only a short time, a microsecond or so, so that the resulting potential distribution is that shown in Fig. 12-5. Actually there exists a "critical separation" for any given tube and accelerating voltages; if PQ (the separation of the spot centers) exceeds the critical value, the first spot suffers little deterioration upon the charging of the second. The critical value is quite small; in Williams and Kilburn's original experiments it turned out to be $1.33d$, where d is the spot diameter.[1]

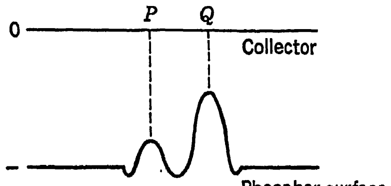


FIG. 12-5. Double-dot potential distribution.

Now suppose the deflection voltages again adjusted to direct the beam to position P in Fig. 12-5. When the beam is turned on, the output signal is the sum of three components: (1) the cloud signal, (2) a positive spot-charging signal caused by the reestablishment of the equilibrium charge in spot P, and (3) a negative-going signal due to secondaries from spot P

[1] Williams and Kilburn, *op. cit.*, p. 187.

that fall on spot Q and neutralize its positive charge (since this is a slower process than the reestablishment of the charge at spot P and since it continues even after P has reached its equilibrium potential, because only a small proportion of the secondaries are available, the negative pulse is rather broad). In Fig. 12-6 the three pulses and their sum are illustrated.

The discussion just concluded has shown that it is easy to distinguish the single- and double-spot distributions from each other. All that is necessary is to direct the electron beam to the first dot position and turn it on briefly (say for 1 to 2 μsec), for it has been seen that the pick-up-plate signals are quite different in character—initially negative-going for the single-dot distribution, but positive-going for the double-dot distribution. It is assumed in this argument that the inspection is performed before leakage and redistribution of electrons have permitted the single-dot pattern to deteriorate substantially; for, if this deterioration were allowed to occur, a positive spot-charging pulse would be obtained in either case, and no distinction would be possible. The actual determination of the character of the distribution being inspected is made by strobing the output signal after it has been greatly amplified. For example, in the IAS computer, the initial negative swing of the 0 signal is approximately 0.25 mv, and the initial positive swing of the 1 signal is about 1 mv. The gain of the read-out amplifier is about 20,000. This is merely a question of enabling a gate for a part (e.g., about the first quarter or third) of the duration of the inspection, for in that interval the dot and double-dot signals are, respectively, negative and positive (see the strobe pulse in Fig. 12-6); the other gate input is the amplified signal, which is passed if it is positive but not if it is negative. The gate output can then be used to read into a toggle, so that the information is made available as long as desired, both for transmission to other parts of the computer and for regenerating the information in the inspected spot, in case this was a double-dot distribution and was destroyed by the inspection.

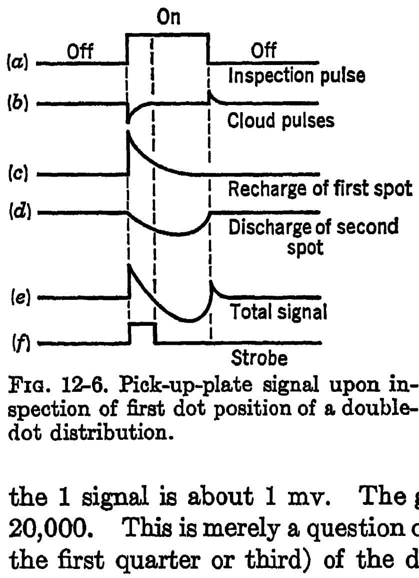


FIG. 12-6. Pick-up-plate signal upon inspection of first dot position of a double-dot distribution.

The signals obtained from the amplifier differ from those shown in Fig. 12-6, showing finite rise time and rounded peaks, so that positive and negative maximum excursions occur after the beginning of the inspection pulse. Hence it has been found advantageous to delay the strobe somewhat,[1] and Thorensen has devised a modified dot-dash system in which strobing occurs just before the end of the inspection.[2]

12-5. Redistribution Effect: Read-around Ratio. No matter what the mode of operation, in a practical memory the electron beams of the tubes are turned on tens of thousands of times per second. During each of these operations, many of the secondary electrons emitted end their flights by falling on adjacent memory locations and cause deterioration of their potential distributions. For example, in the dot-double-dot system, reduction of the potential maxima has the effect that, when a 0 is read, a positive spot-charging signal is added to the cloud pulses and, if deterioration has gone far enough, the initial swing of the output signal may be in the positive direction, so that the 0 is read as a 1. Deterioration of a 1 causes no trouble as far as the initial positive signal is concerned; it does, however, tend to reduce the negative swing that follows the initial positive swing because it reduces the potential of the second dot. If no countermeasures were taken against the "redistribution" or "splash" effect, all the information in the tube would be lost. The usual practice, therefore, is to alternate periods of "action" (reading or writing) with periods of "regeneration." Thus, in the serial mode, during every other scan the contents of an entire line are regenerated, and a counter ensures the successive regeneration of all lines; in the parallel mode a similar program is carried out one location at a time.

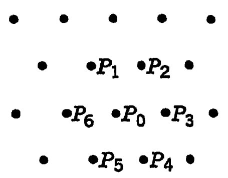


FIG. 12-7. Array of memory locations in ILLIAC Williams tubes.

A further complication arises from the fact that the successive references to the memory in action periods are not random. For example, it is easy to imagine the possibility (see Fig. 12-7) that six instructions of a short subroutine forming part of an iterative process might be located at P_1 to P_6 and that, therefore, these locations would be read a considerable number of times between regenerations of P_0. Under these circumstances, the information at P_0 is bound to deteriorate much more rapidly than it would if references to the memory were purely random. Alternatively, if P_0 is repeatedly consulted, the information at the adjacent P_1 to P_6 must be expected to deteriorate rapidly. A customary figure of merit for the retention of information under these circumstances is the "read-around ratio," which is defined as the maximum number of times a particular location can be consulted without loss of

[1] This will be found both in Eckert, Lukoff, and Smoliar, *op. cit.*, and in Williams and Kilburn, *op. cit.*

[2] R. Thorensen, An Improved Cathode Ray Tube Storage System, *Proc. Western Computer Conf.*, *1953*, pp. 167–173, Institute of Radio Engineers, New York.

information in an adjacent spot.[1] Read-around ratio varies from location to location in a given tube and depends also on the type of tube, the scheme used for representing 0's and 1's, the time during the inspection when the simplifier output is strobed, and the type of output amplifier, to list the more obvious factors.

It is quite possible to live with low read-around ratio, but it will be appreciated that this puts an additional burden on the coder. It is actually difficult to say at what point it ceases to be a cause of concern; it may be assumed that most exactness would be satisfied if every spot in the memory had a read-around ratio of not less than 200.* In the early stages of development such ratios were not achieved, and a great deal of detailed engineering effort was required to raise the performance to a satisfactory level. However, by the time a good read-around ratio was achieved, the Williams memory was already becoming obsolete, so there is no point in recounting the details here.

12-6. Magnetic-core Arrays. Introduction. A treatment of magnetic cores having "rectangular" hysteresis loops has been given in Sec. 4-14, in Sec. 5-7 the use of such cores as binary memory elements has been discussed, and in Chap. 8 it has been shown how shifting registers can be built up of them. In this chapter it will be shown how large-capacity low-access-time parallel memories can be built up of such elements. These were first suggested in 1947 by both J. W. Forrester[2] and A. D. Booth; most of the pioneering development work was done at the Digital Computer Laboratory at MIT under the direction of Forrester and at the RCA Laboratories by J. A. Rajchman. In the summer of 1953 the MIT work resulted in an operable full-scale memory which replaced the electrostatic memory in Whirlwind I. Since then larger and improved memories have been built at MIT and elsewhere; because of their inherent reliability and durability, these have become very popular and seem to be the best answer (at the moment!) to the requirements of large capacity, short access time, and reliability in parallel memories. In the following sections such memories will be discussed in some detail.

12-7. A Simple "Ideal" Array. Consider the array of four cores shown in Fig. 12-8. Each core carries two windings for the reading and writing

[1] See, e.g., *ibid.*, p. 167.

* The design specification of ORACLE required a value of 250, which is said to have been attained; see R. J. Klein, The Oracle Memory System, *Proc. Symposium on Large Scale Digital Computing Machinery*, Argonne National Laboratory, 1953, pp. 47–58.

[2] J. W. Forrester, Data Storage in Three Dimensions, *Project Whirlwind Rept.* M-70, Massachusetts Institute of Technology, Cambridge, Mass., April, 1947.

Booth proposed a similar scheme in lectures in England at the same time. The first publication in a scientific journal was by Forrester, Digital Information Storage in Three Dimensions Using Magnetic Cores, *J. Appl. Phys.*, vol. 22, no. 1, pp. 44–48, January, 1951.

currents and a third sensing winding from which the output is obtained in reading. In actual practice, it is found that, using small cores of outer diameter 1/10 in. or less, each winding can be replaced by a wire thrust through the hole in the core, thus forming in effect a single-turn winding.

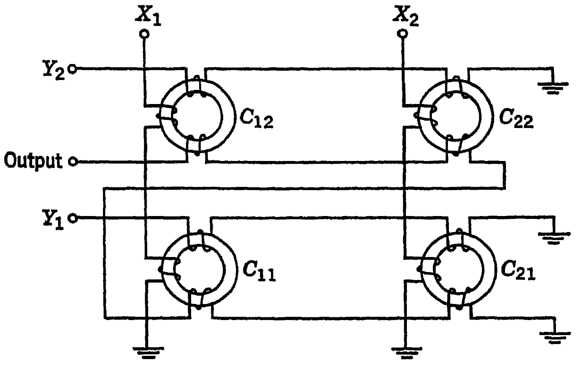


FIG. 12-8. Simple memory array.

Magnetization in the clockwise sense is taken as negative and is used to represent binary 0. The positive sense of current flow is taken as out of the terminals X_1, Y_1, X_2, Y_2, and output. The cores are assumed to possess identical ideal hysteresis loops of the kind shown in Fig. 12-9, in which the magnetizing force H_m required to reach the maximum value of B is less than twice the value H_1 at the knee of the curve.

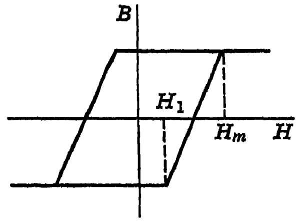


FIG. 12-9. Ideal hysteresis loop for coincident-current memory core.

To begin with, suppose that all the cores are in the state of negative saturation, interpreted as 0, and that it is desired to switch only core C_{11} to the 1 state of positive saturation. To accomplish this, it is necessary to apply to C_{11} a magnetizing force H_m, while at the same time ensuring that no other core experiences a force greater than H_1. Assuming that the reading and writing windings on all cores are identical, and letting I_1 be the current required to produce H_1, there are two obvious alternatives: (*a*) drive Y_1 with a positive current pulse of amplitude I_m and X_2 with a pulse of amplitude $-I_1$, or (*b*) drive both X_1 and Y_1 (the selected lines) with pulses I_1. In the first case, C_{11} is switched to the positive state, but C_{21} is subject to a force less than H_1 and does not switch; in the second case the currents I_1 in the two windings of C_{11} produce a force $2H_1 > H_m$ and cause switching, but no other core experiences an excitation greater than H_1; in both cases, only the "selected core" C_{11} is switched, and all others remain in their original states.

Destruction (or erasure) of the information inserted in C_{11} by the procedure just described can be accomplished by either procedure *a* or *b* with the senses of all currents reversed. This can also be used to determine (or read) the information held by the selected core, since only if it holds a 1 will it be switched to 0, the resulting large change in flux causing a signal to appear at the terminal labeled "output." This mode of "destructive reading," which so far is the only practicable one to be found, requires that the reading of the contents of any core be followed at once by regeneration or rewriting if the information read is also to be retained in the memory.

The extension to a larger square or rectangular array is immediate. If writing scheme *a* is used, the selected Y line is driven by a pulse of current I_m and the unselected X lines are driven by pulses $-I_1$; in writing scheme *b*, the selected X and Y lines are both driven by current pulses I_1. Reversed currents, of course, are used to read or erase any selected core. Scheme *b* was first proposed by Forrester and has been used in the Whirlwind I memory;[1,2] it is said to work by "coincident current." Both *a* and *b* and a variety of other methods have been proposed, but it is clear that, given ideal cores with identical hysteresis loops, one should be as good as another; the differences become significant only when the effects of nonideality in the hysteresis loops have been analyzed. Before passing to a discussion of these, it is worthwhile to give some discussion of the possibility of nondestructive read-out.

The fact that the reading process in the usual forms of magnetic-core arrays destroys the information read, which, therefore, must be restored as soon as the reading has been accomplished, is admittedly a nuisance. Various attempts have been made to devise nondestructive read-out schemes; the first was due to D. A. Buck[3] of the Digital Computer Laboratory at MIT. Only the rudiments of Buck's ideas will be given here; the scheme has not yet been put into large-scale practice, one reason being that the structure of the cores presents a rather severe manufacturing problem. A mathematical analysis of the process has been published by Papoulis.[4]

[1] W. N. Papian, A Coincident Current Magnetic Memory Cell for the Storage of Digital Information, *Proc. IRE*, vol. 40, no. 4, pp. 475–478, April, 1952.

[2] D. R. Brown and E. Albers-Schoenberg, Ferrites Speed Digital Computers, *Electronics*, vol. 26, no. 4, pp. 146–149, April, 1953.

[3] D. A. Buck and W. J. Frank, Nondestructive Sensing of Magnetic Cores, *AIEE Tech. Paper* 53-409, October, 1953. Buck's original proposal was made in an MIT Digital Computer Laboratory engineering note in 1951.

[4] A. Papoulis, The Nondestructive Read-out of Magnetic Cores, *Proc. IRE*, vol. 42, no. 8, pp. 1283–1288, August, 1954.

Buck's idea came from considering a single ferromagnetic crystal of cubic shape in which preferred, or "easy," directions of magnetization lie along the edges and "hard" directions may be imagined to lie between them. If the crystal is magnetically saturated, with the remanent magnetization vector lying along an edge, and if a field is applied at right angles to it, the total magnetization vector tends to align itself with the field and so rotates away from the easy and toward the hard direction of magnetization. If the applied field is not sufficiently strong to rotate the vector past the hard direction, its removal permits the vector to snap back to its original position.

Now consider a core of rectangular-hysteresis-loop material. The core can be thought of as made up of a large number of crystals, all aligned so that an easy direction of magnetization is at every point tangential to the core and the remanent flux B_R is at every point in the tangential direction. If H_1 is applied perpendicular to B_R, the resultant B vector is rotated away from the tangential direction. The length of the resultant cannot differ much from that of B_R, which is assumed to have resulted from prior saturation of the core. The change in flux linking the sensing windings is proportional to $1 - \cos \theta$; this flux decreases when H_1 is applied and increases to its initial value again when H_1 is removed; so an induced voltage appears across the sensing winding. It is clear that, if the same direction is used for H_1, the polarities of the signals for 0 and 1 will be opposite.

There are several ways in which H_1 can be applied, none of them satisfactory from the point of view of simplicity of manufacture: (1) the memory core can be inserted in the gap of a large magnetic core (not of rectangular-hysteresis-loop material); (2) the core can be a hollow toroid, with a conductor placed within the space; and (3) the core can be wound of a ribbon of metallic magnetic material (e.g., Deltamax) in a tight spiral, and the quadrature field set up by passing a current through the core material itself. All these systems have been tried by Buck and others. In all cases fast read-out (well below 1 μsec) was achieved with outputs as high as 1 volt per turn, with sensing currents of the order of 0.5 amp. Information was not observed to deteriorate after numbers of sensings running up to 10^9. The "internal-field" type of excitation (cases 2 and 3 above) seems to be best in that it is relatively compact and the results are independent of air-gap length, position, and other factors exterior to the core itself.[1] The chief drawbacks seem to be the large back voltages developed across the quadrature windings and the poor power efficiency, but it has been suggested that these can be obviated by working with lower excitations; in fact, it has been found possible

[1] D. A. Buck and W. J. Frank, Nondestructive Sensing of Magnetic Cores, *MIT Digital Computer Lab. Eng. Note* E-454-1, Mar. 24, 1953.

to obtain reasonable output signals with quadrature excitation currents of 20 ma or even less.[1]

It can be taken that the principle involved in nondestructive read-out has been well established, but there does not as yet appear to be a simple and practical way to make a large-scale application. However, the sensing speed obtainable would appear to justify a considerable amount of development effort.

12-8. Effects of Nonrectangularity of the Hysteresis Loop. Hysteresis loops encountered in practical cores have been illustrated in Fig. 4-21. The loop of Fig. 4-21 may be taken as fairly representative, though others show a more pronounced corner at the knee;[2] all, however, exhibit positive slopes at the remanence points. The values of H_1 and H_m are not well defined; H_1 must be chosen so that $2H_1 \geq H_m$ if the MIT coincident-current procedure is to work at all. Consider a core having a loop of the form of Fig. 4-21 (reproduced in Fig. 12-10), which has been driven to the 1 state; try to predict the effect of the next excitation by $-H_1$, which occurs during the next reading operation in which the core, though not itself selected, lies in a selected row or a selected column. During the $-H_1$ pulse, the point representing the state of the core moves along the loop to point A_1; upon the removal of the pulse it returns to a point P_1. A second excitation by $-H_1$ drives the core first to some point A_2 below A_1, whence it returns to P_2 as shown, and so on. If a great number of negative excitations $-H_1$ occur with no positive excitation intervening, the sequence of points P_1, P_2, . . . may converge quite rapidly to a point not far below P_2; this has, in fact, been observed in the case of a core with a reasonably rectangular loop.[3] On the other hand, if the loop is of a less

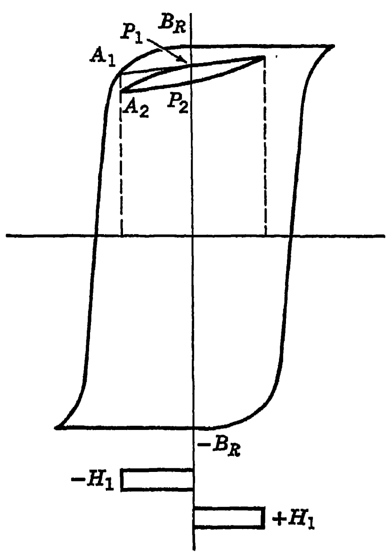


FIG. 12-10. Compensation of disturbed 1.

[1] D. A. Buck, Further Work on Nondestructive Read System, *MIT Digital Computer Lab. Memo.* M-2195, May 27, 1953.

[2] See, e.g., the loop for General Ceramics MF-1118 ferrite, given by Brown and Albers-Schoenberg, *op. cit.*

[3] J. A. Rajchman, A Myriabit Magnetic-core Matrix Memory, *Proc. IRE*, vol. 41, no. 10, pp. 1407–1421, October, 1953 (Rajchman II).

See also J. A. Rajchman, Static Magnetic Matrix Memory and Switching Circuits, *RCA Rev.* vol. 13, no. 2, pp. 183–201, June, 1952 (Rajchman I).

favorable shape, the core may ultimately lose its positive magnetization entirely. A core that has been subjected to the treatment just described is said to hold a "disturbed 1"; when it is selected and read, it certainly yields a weaker output signal than a core holding an "undisturbed 1." The ratio of the peak value of the disturbed-1 signal to the undisturbed-1 signal is a figure of merit called the "1-retention ratio." Similarly, it is evident from Fig. 12-10 that a core holding a 0 gives a small output signal when selected for reading. Excitation by positive pulses H_1 decreases the negative magnetization and causes the disturbed core to yield a larger output signal; the ratio of the disturbed-0 signal to the undisturbed-0 signal has been called the "0-retention ratio." Finally, the ratio of the magnitude of the peak output from a core holding a disturbed 1 to the output magnitude from a core holding a disturbed 0 is a figure of merit called the "disturbed-signal ratio."[1]

From the above discussion it is evident that it is wise to adopt measures to prevent the deterioration of information stored in magnetic cores. In an experiment of Rajchman, a core was first set to $+B_R$ and then brought down to P_2 by two negative excitations $-H_1$, as in Fig. 12-10; then pulses $+H_1$ and $-H_1$ were applied, and the net effect was to cause the representative point to pass around a closed loop and to return to P_2.* Thus, if selection for reading is accomplished by the coincidence of two pulses $-H_1$ and $-H_1$, deterioration of 1's can be prevented by exciting both the selected row and the selected column by $+H_1$ after each reading operation. If these compensation pulses were applied simultaneously, they would write a 1 in the selected core, no matter what information it had previously held; to prevent this, the compensation pulses can be staggered in time. In Rajchman's experiments, the rows and columns were selected by magnetic switches of the d-c biased type described in Chap. 7. It will be recalled that, in this type of switch, the selected core switches from $-B_R$ to $+B_R$ and finally back to $-B_R$; hence the voltage across its output winding is a pair of pulses of opposite polarities. If the polarities are so chosen that the first is used for reading, then the second is available for compensation, and, if the pulses used to excite the X and Y switches begin at the same instant but are of different lengths, compensation is accomplished without automatically writing 1 in the selected core.* Various versions of compensation are incorporated in practical magnetic-core memories.

Another approach to the problem of minimizing the disturbance of information has been suggested:[2] for writing 1, in terms of the scheme of

[1] Papian, *op. cit.*, p. 476.

* Rajchman II.

[2] J. M. Wier, A Dynamic Magnetic Memory, *Digital Computer Lab. Internal Rept.* 54 (report issued under ONR contract N6ori71, Task 24), The University of Illinois, Jan. 7, 1954.

Fig. 12-8, the selected row (column) is excited by a pulse of amplitude $\frac{4}{3}I_1$, and the other rows (columns) are unexcited; the selected column (row) is excited by a pulse of amplitude $+\frac{2}{3}I_1$, and the unselected columns (rows) are excited by a pulse of amplitude $-\frac{2}{3}I_1$. Thus the selected core experiences a total magnetizing current equal to $+2I_1$ and is driven to the 1 state, but other cores of the matrix experience magnetizing currents equal to $\pm\frac{2}{3}I_1$ and, hence, are not so much disturbed as in the method previously described. However, it is not at all clear that the disturbance of information would be so small that compensation could be dispensed with. Other combinatorial schemes have been discussed by Rajchman.[1]

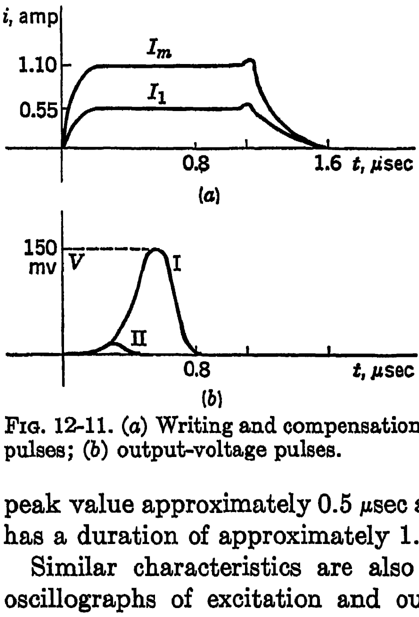


FIG. 12-11. (*a*) Writing and compensation pulses; (*b*) output-voltage pulses.

As an example of commercially available cores consider those used in the Whirlwind I memory,[2] which were made by the General Ceramics and Steatite Corp. These are of a ferrite (General Ceramics type MF-1326B) of outer diameter 80 mils. The total current in a single-turn winding required for switching is 850 ma. The peak output voltage across the one-turn sensing winding during switching is 0.1 volt; the output voltage pulse reaches its peak value approximately 0.5 μsec after the switching pulse is applied and has a duration of approximately 1.2 μsec.

Similar characteristics are also reported by Rajchman,[3] who gives oscillographs of excitation and output pulses about as shown in Fig. 12-11*a*. In Fig. 12-11*a* the pulse I_1 is the one that would be used, given the arrangement of Fig. 12-8, to excite both the selected row and the selected column when writing 1, and the pulse I_m gives the total current required to write 1, in this case 1.1 amp. In Fig. 12-11*b* are shown the output voltage pulses obtained in two cases: I from reading a 1 that has been modified (or disturbed) by two negative pulses $-I_1$ and compensated by a single pulse I_1 following them, and II from a half-selected 1. The ratio of the peak voltage is between 10:1 and 20:1; the peak in I occurs at about 0.6 μsec after the beginning of the reading operation, that in II somewhat earlier.

[1] Rajchman I, p. 186.

[2] M. F. Mann, R. R. Rathbone, and J. B. Bennett, Whirlwind I Operation Logic, MIT Digital Computer Laboratory, May 1, 1954.

[3] Rajchman II, p. 1411.

It should be understood that, since the sensing windings are connected in series, the contributions from the unselected cores can cause trouble by adding up and obscuring the signal from the selected core (and clearly this trouble grows with the size of the array); this can be in part avoided by changing the direction of linkage from core to core but, even then, in the worst possible cases the undesired signal may become more than twice the magnitude of the signal from a 1, so that it is not possible to distinguish 0's from 1's. One possible method of avoiding the difficulty consists in strobing the amplifier output at an instant beyond the peak; since the peak of the signals from the unselected cores occurs before the peak of the signal from the selected core (see Fig. 12-11*b*), confusion should in this way be eliminated, but unfortunately the short times available, given fast cores like the one whose action is shown in Fig. 12-11, make the practical realization of the idea difficult.[1]

In his 10,000-bit array Rajchman[2] has employed successfully a method of reading that consists in first driving the selected core in one direction and then in the other and integrating the output voltage. In terms of the polarities that have been employed here, the selected row and column are both excited by pulses of amplitudes $+I_1$ and $-I_1$ in succession. If the selected core holds 1, it experiences a large change in flux and, hence, puts out a large signal of a single polarity. If it holds 0, it puts out in succession large signals of opposite polarities, which integrate to 0, and returns to its original state, the total change of flux being zero. As to the unselected cores, they put out very nearly equal signals of opposite polarity, which for each core integrate very nearly to zero, and their total contribution is small, in the experimental 10,000-bit array about one twenty-fifth of the signal from a 1. This ratio is independent of the size of the array and of the uniformity of core characteristics, and is essentially determined by the properties of the integrating circuit.

12-9. Practical Memories. So far only the single square (or rectangular) array of cores has been considered. Clearly, given the type of selection by coincident currents and the series connection of the sensing windings, it is possible to enter or remove information only 1 bit at a time. A parallel memory is easily built up of a set of arrays of this type, using as many planes as there are bits in the typical computer word and allocating one plane to each binary position in the word. Thus, if the words are n-place binary fractions, array 1 holds all the sign bits, array 2 all the bits in the 2^{-1} position, array 3 all the bits in the 2^{-2} position, . . . , array $n + 1$ all the bits in the 2^{-n} position; and normally these are all stored in cores with identical coordinates in their several planes. Thus a truly parallel memory is built up, where the access time of a word

[1] Rajchman II, p. 1417.
[2] Rajchman II, pp. 1417–1418.

is the same as that of a single bit; this scheme was proposed by Forrester in 1947,[1] and is today in use in almost all magnetic-core memories.

In place of an overly elaborate diagram, the simplified version given in Fig. 12-12 will suffice to illustrate the main features. Where two arrays (plane I and plane II) of nine cores each are shown, the reader must imagine 17 arrays of 1,024 cores each (16 bits per word with the 17th for a parity check), and where the arrays are shown in vertical planes, the reader must imagine them in horizontal planes, stacked one above

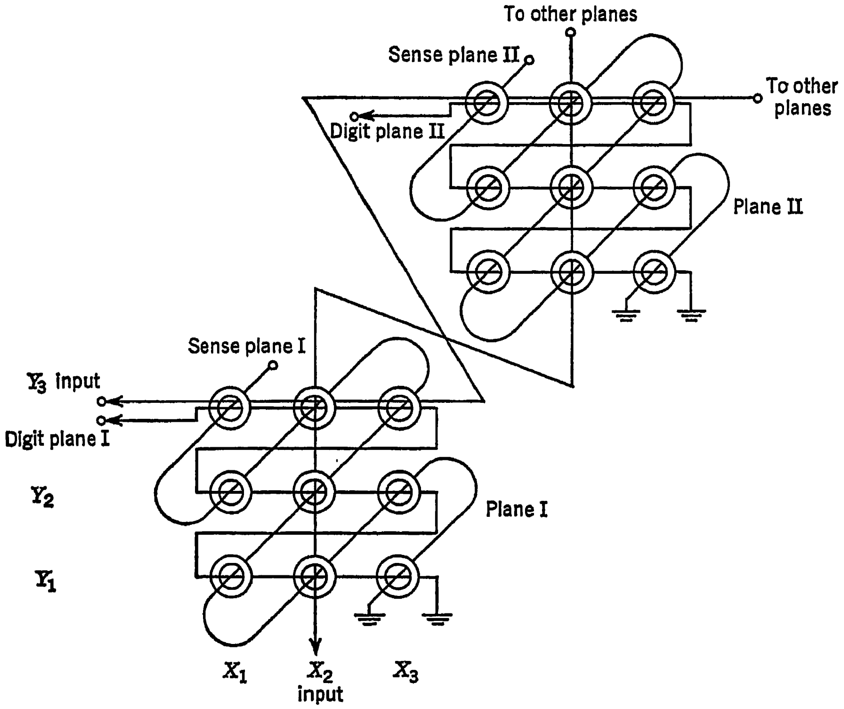


FIG. 12-12. Wiring of a three-dimensional memory.

the other, with each array, or plane, held in a 9½- by 9½-in. frame.[2] The properties and size of the cores have been discussed briefly in Sec. 12-8. The single-turn windings consist merely of 32-gauge magnet wire carrying quadrupole Formex insulation thrust through the cores, and each core carries four windings: one for row selection, one for column selection, one for sensing, and the so-called "digit-plane" winding used in the writing process in a manner to be described. The sensing windings of all the cores in a plane are connected in series, and so are the digit-plane windings. The column-selecting windings of corresponding rows in

[1] Forrester, *op. cit.*, pp. 46–47.

[2] Specifically, this refers to the original Whirlwind I memory.

the several planes are connected in series, and so are other row-selecting ones. In Fig. 12-12, to avoid confusion, the only such windings shown are those selecting row 3 and column 2 (Y_3 and X_2); the others, of course, are treated identically.

Using the same convention as before, negative magnetization is taken in the clockwise sense; so the positive senses of the currents in the row-selecting, column-selecting, and digit-plane windings are as indicated by the arrows.

The read-write cycle of the memory takes 9 μsec; the timing chart is given in Fig. 12-13. The address is first set up by two 32-position diode-matrix switches; the inputs to the toggles that drive them are the 5-bit binary numbers specifying the coordinates X and Y, and each of the 32 outputs of each switch is connected to a gate. The two gates (one for a

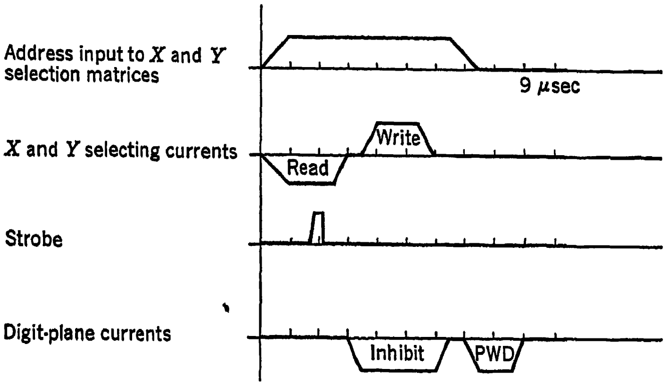


FIG. 12-13. Timing chart of Whirlwind I magnetic memory.

row and one for a column) that are enabled then pass "read" pulses that excite drivers whose plate currents (I_1 is about 0.4 amp) flow through the appropriate selection windings (say Y_3 and X_2) in the negative sense, so that the appropriate cores in all the digit planes are driven to the 0 state if they were not there already. The outputs from the terminals labeled "sense plane I," etc., are each amplified from the level of about 0.1 volt up to that of 30 volts and used to enable a gate; in each case the other input to the gate is a short (0.1-μsec) strobe pulse occurring when the signal reaches its peak value, about 1 μsec after the full excitation is applied to the selected cores, and the outputs are connected to a set of toggles called the "buffer register." The effect of these operations is (*a*) to set all the selected cores to 0, and (*b*) to transfer the information previously held by them to the buffer register, from which it can be transmitted to other parts of the machine.

It is now necessary to restore the selected cores to their original conditions. To do this, both the selected lines are pulsed with a positive current of 0.4 amp. This would drive all the selected cores to the 1 state. To inhibit the writing of 1's in planes that hold 0's, a negative current ($-I_1 = -0.4$ amp) is caused to flow in the appropriate digit-plane windings; this is done for each plane by a gate-driver arrangement controlled by the appropriate stage of the buffer register, 0 enabling the gate. As the chart shows, the inhibiting pulse slightly overlaps the write pulse, for safety.

It is quite clear from the above discussion that writing can be accomplished by first placing the word to be written in the selected position in the buffer register and then going through the routine described but omitting to strobe the output. The buffer register can, therefore, be used as the means of communication between the memory and the rest of the machine.

At the conclusion of the read-rewrite operations, a compensation, or "post-write-disturb" (*PWD*), pulse is applied to all digit-plane windings.

It will be appreciated that a variety of different modes of construction and operation of magnetic-core memories are possible. Thus, for example, the EDSAC II memory[1] holds 1,024 words of 41 bits each (40, and one for parity check), and is built up of 32 rectangular planes each of 32 × 41 cores, so that each plane holds all the cores in a single column, or 32 complete words. Another example in which rectangular arrays are used is the International Telemeter Corporation's[2] Mnemotron, the first of which was built for the Rand Corporation.[3] As this is the first core memory commercially available, it is of sufficient importance to justify a fairly full discussion, which follows. A later chapter contains a full account of a memory of this kind built for the ORDVAC to replace the original Williams memory.

The timing of the selection currents of the Mnemotron is illustrated in Fig. 12-14; actually in the first Mnemotron the entire cycle took 15 μsec instead of 12 μsec, but the waveforms and relative timing were precisely as shown. The drawing given in the reference[3] has been changed to conform to the conventions adopted here. In all cases, the *X*-selecting-current waveform is asymmetrical as shown. Also, in all cases, *Y*-selecting current during the first (or reading) half of a memory cycle is the same; observe that the *Y*-reading pulse does not begin to rise for over 2 μsec after the initiation of the *X* pulse. Thus the reading amplifier need not be gated on until the beginning of the *Y* pulse, and transient effects due to the rise of the *X* pulse, such as signals from half-selected cores, have

[1] In the University Mathematical Laboratory, Cambridge, England. See Sec. 12-10.

[2] Now Telemeter Magnetics, Inc.

[3] R. Stuart-Williams, M. Rosenberg, and M. A. Alexander, "Recent Advances in Coincident-current Magnetic Memory Techniques," International Telemeter Corp., Los Angeles, 1954. This paper was presented at the annual meeting of the ACM, Ann Arbor, June, 1954.

time to subside and, hence, do not affect the amplifier output. In a square matrix this would cut the unwanted signal in half. The second half cycle of the Y-selecting current is variable. If a 0 has been read, it is as shown in Fig. 12-14*a*; if a 1 has been read, it is as shown in Fig. 12-14*b*. In the first case, the write signal never exceeds I_1 (slightly less than 0.5 amp); in the second, the X and Y signals add and give the required $2I_1$.

Other advantages accrue from this type of cycle. Thus, since the transients due to the X excitation subside before the reading amplifier is turned on, it is not necessary to worry about coupling between the X and the sensing windings. The sensing windings consist of a wire parallel to the X windings instead of the diagonal arrangement shown in Fig. 12-12. This makes the assembly of the memory planes a good deal simpler and cheaper. It is also possible to add several spare columns of cores, so that, if a core becomes damaged or if a poor core is inadvertently

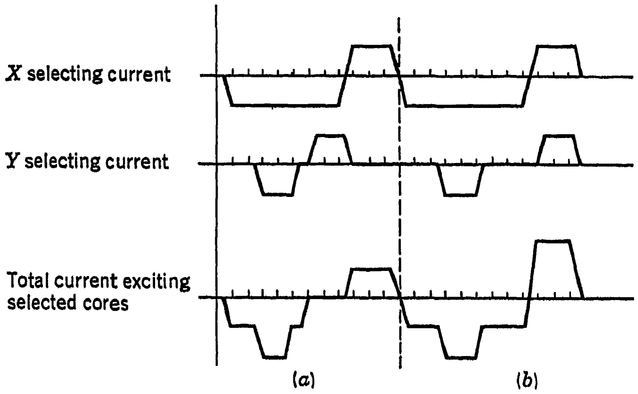


FIG. 12-14. Mnemotron read-write cycle. (*a*) Read-write 0, (*b*) read-write 1.

included, the column containing it can be replaced in operation by one of the spare columns. Finally, since the sensing winding crosses the Y (column) windings at right angles, direct pickup of a signal induced in the sensing winding by a Y winding is minimized.

To realize the type of cycle shown in Fig. 12-14, it is convenient to use a vacuum-tube driver for the creation of the X excitation, because of the length of the initial pulse (about 8 μsec) and also because of the lack of symmetry. A single driver's output is routed to the various X lines by vacuum-tube switches. As the Y excitation is short and the pulses symmetrical, a magnetic switch is appropriate, and a separate switch is provided for each digit plane.

The separate digit planes are not square but rectangular, in the first model 32×128 cores. As the cores along the selected X lines do not contribute disturbed signals to the output amplifier, the long dimension is chosen for the X direction; with this arrangement, the unwanted signal is no worse than that obtained from a 16×16 array under simultaneous

X and Y excitations. In the actual structure, the 32×128 array is folded along the center of its long side to produce a double-sided array of 32×64 cores; a metal sheet forms the physical center of the frame and also provides a common ground return for the switch-driven Y lines.

The Mnemotron has one memory array for each digit position; the first model contained 40 planes, for the word length of the machine in which it was to function was 40 bits. As with Whirlwind, some sort of buffer register is needed to receive the output and hold it (1) until it is needed elsewhere in the machine and (2) until it has served the purpose of determining which of the two possible Y write pulses is to drive each plane.

The last example to be cited will be the IBM 738 magnetic-core memory, built for use in the IBM 704 and 709 data-processing machines and designed to hold 32,768 binary words of 36 bits each.[1] The memory cores are arranged in planes of 64×128; there are four stacks, each consisting of 36 of these planes, rather than a single stack of larger (128×256) planes, because it was felt that difficulties might be encountered in producing the larger planes in quantity.

The memory is of the coincident-current type, using a 12-μsec cycle much like that of the Whirlwind memory described above. The 128 X lines and 256 Y lines are driven by magnetic-matrix switches, each having 64 cores, and these are excited by vacuum-tube drives using low-gain twin power triodes (type 5998). The sensing amplifiers use junction transistors and are assembled on etched-wiring cards. In order to avoid disturbance of information in "half-selected" cores, an ingenious scheme of "address scattering" is used. This is, at the moment, the largest successful magnetic-core memory in commercial production.

12-10. Direct-selection Core Memories. It has been shown above that the coincident-current scheme of organization of magnetic-core memories suffers from the disadvantage that the unwanted signals from the half-selected cores are troublesome and that various inconvenient measures must be adopted to minimize their effect. This difficulty can, however, be eliminated by employing another mode of organization, originally conceived at NBS and variously called the "word-arrangement" or "direct-selection" scheme. This conception was also independently arrived at in the University Mathematical Laboratory in Cambridge, England, where the first sizable (1,024-word) memory based upon it was built, and is incorporated in the EDSAC II.* A small

[1] E. Foss and R. S. Partridge, A 32,000 Word Magnetic Core Memory, *IBM J. Research and Development*, vol. 1, no. 2, pp. 103–109, April, 1957.

* W. Renwick, A Magnetic Core Matrix Store with Direct Selection Using a Magnetic Core Switch Matrix, *Proc. IEE*, vol. 104, pt. B, supplement no. 7, pp. 436–444, 1957.

(64-word) memory of this type is also in use in the TX-2 computer built in the Lincoln Laboratories.[1]

In the direct-selection scheme, only the cores holding the bits of the selected word are disturbed during the reading operation. There are no "half-selected" cores, and the unwanted signals are nonexistent. A price must be paid for this desirable feature, in the increased complexity of the method of selection. However, an additional benefit is obtained in that it becomes possible to drive the selected cores harder and, hence, to cause them to switch more rapidly, so that a shorter memory cycle is feasible than with the conventional coincident-current scheme.

In the EDSAC II memory, the word length is 40 bits plus a parity-check bit. The cores are mounted in planes, and the 41 bits of each (say) row or "word line" composes one word. The cores of each word line are threaded by a single wire, and the cores of each column are threaded by two wires. When a word is to be read, a read pulse of amplitude 1 amp is caused to flow in the appropriate word line. This is about 50 per cent greater than the excitation required to drive the cores to the 0 state, so switching is quite rapid. One of the two lines threading each column is connected to an amplifier, the output of which can set a toggle via an appropriate gate. The signal received by each amplifier depends only upon whether the corresponding core of the selected word line is caused to switch or not, and thus there is no unwanted-signal problem. In writing, the second wire threading the corresponding cores of all word lines comes into play. If a 1 is to be written, this wire is excited with a pulse of amplitude -0.36 amp; if a 0 is to be written, the wire receives no excitation. Simultaneously, a pulse of amplitude -0.36 amp excites the chosen word line, and writing is accomplished in the familiar coincident-current mode.

The selection of the word lines is accomplished by a 40×32 matrix of ferrite switch cores, each of which is a small three-winding transformer (the extra 256 cores in the matrix are used to select certain permanently recorded items of information, such as constants or the instructions of frequently used subroutines). These cores are larger than the memory cores, which are of the type described earlier; they are 0.58 cm in mean radius and 0.1 cm^2 in cross section. The three windings, designated "drive," "bias," and "output," are of 37, 37, and 4 turns, respectively. The bias windings of all the cores in a row are connected in series; so are the drive windings of all the cores in a column, and the rows and columns are driven by pentodes which are driven in turn by two-address decoders. The output windings drive the word lines.

Under quiescent conditions the drive windings are not excited, but all the bias windings carry a current of sufficient strength to hold all the

[1] W. N. Papian, High Speed Computer Stores 2.5 Megabits, *Electronics*, vol. 30, no. 10, pp. 162–167, October, 1957.

switch cores in negative saturation. To cause a switch core to drive its word line, the bias current in its row is removed, and a drive pulse of sufficient strength to drive it to positive saturation is applied in its column. The signal induced in the output winding as the core switches from negative to positive saturation is the read pulse. The write pulse is produced after the ending of the drive pulse by restoring the bias and hence returning the selected switch core to negative saturation.

Reading is obviously destructive. During the write part of the cycle, information obtained during the reading part of the cycle can be rewritten, or new information can be recorded.

The total length of the memory cycle is 7.5 μsec. Recent experiments conducted at the University of Illinois have indicated that a cycle of as little as 1.5 μsec should be attainable.

12-11. Novel Magnetic Elements. Using the familiar coincident-current method of selection described in detail above, the time required

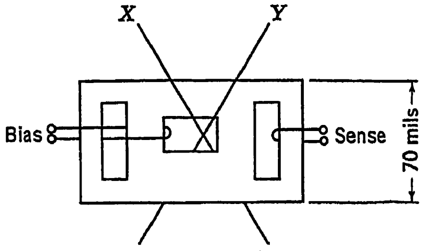


FIG. 12-15. IBM multipath core.

to switch conventional ferrite cores turns out, as noted before, to be somewhere between 1 and 2 μsec. This could be reduced if the cores could be driven harder, that is, by a larger applied magnetomotive force, but this is impossible as long as 2:1 coincident-current selection is employed. A way out of this dilemma has been devised in the laboratories of IBM;[1] this method incorporates features of the biased magnetic switch described above. The switch cores cannot function as memory units, since they are all returned to one state after the switching currents have been removed by the bias current.

It is possible, however, to provide a memory feature by using a more elaborate core: to be specific, a rectangular core with three rectangular holes, as shown in Fig. 12-15. The *X*- and *Y*-selecting windings pass through the center hole, and the sense and bias windings about one end and one center leg, as indicated. The bias current is sufficiently great to produce saturation far in the negative direction; the *X*- and *Y*-selecting currents both are equal but opposite in effect to the bias current. Thus,

[1] W. W. Lawrence, Jr., Recent Developments in Very High Speed Magnetic Storage Techniques, *Proc. Eastern Joint Computer Conf., 1956*, American Institute of Electrical Engineers, New York, 1957.

when selection occurs, the core is driven far in the positive-saturation direction.

Assume that the selecting currents for writing produce a field counterclockwise about the center hole and that the bias current produces a field counterclockwise about the left and clockwise about the center hole. If all three are flowing simultaneously and then the selecting currents are turned off, it is clear that a residual field is left which is counterclockwise about the right hole. If, now, currents equal but opposite to those used for writing are passed through the two selecting windings, the field in the right (i.e., "sense") leg will be reversed, and a voltage will appear across the sense winding during reversal. After the removal of the excitation, the residual field about the right hole must be in the clockwise sense. Thus there are two magnetic states corresponding to 1 and 0.

With these IBM multipath cores, using bias and selecting currents each sufficient to produce a magnetizing force of 5 amp-turns and fast-rising (that is, in 50 mμsec) selecting pulses, a switching time of 0.1 μsec has been obtained. With the same cores, the ratio of the amplitudes of the output voltage pulses corresponding to 1's and 0's was found to be 9:1.

Another rapidly switching element is produced by depositing by evaporation a thin film of metallic ferromagnetic material on glass. The film is deposited in small spots (for example, 1 cm in diameter) a few thousand angstroms thick. The switching process in such elements is, apparently, quite different from that in bulk material and can be accomplished in a fraction of a microsecond. An experimental memory using these elements has been built in the laboratories of the Remington-Rand UNIVAC Division of Sperry-Rand.[1] The selecting and sense windings were thin, flat conductors etched on over the glass plate upon which the cores had been evaporated. The experimental memory operated on a cycle of 2 μsec, but this could, presumably, be reduced. It appears that this type of magnetic-memory plane may be very well suited to large-scale production and could be produced in quantity at very low cost.

Still another novel magnetic element, which should simplify the manufacturing problems in producing memory planes, has been devised in the Burroughs Research Center. The scheme is very simple: first the grid wires of the plane are put in place, and then a small piece of thin tape made of an appropriate material is wrapped about the appropriate places.[2]

[1] A. V. Pohm and S. M. Rubens, A Compact Coincident Current Memory, *Proc. Eastern Joint Computer Conf., 1956*, pp. 120–123, American Institute of Electrical Engineers, New York, 1957.

[2] R. A. Tracy, Megabit Memory, *Proc. Eastern Joint Computer Conf., 1956*, American Institute of Electrical Engineers, New York, 1957.

The final example to be considered is the ferrite apertured plate, developed by J. A. Rajchman of the RCA laboratories.[1] This provides a complete magnetic-memory plane in a single element; it is a small square plate (for example, 1 by 1 in.) in which holes are drilled in a matrix array. On this plate is etched a winding linking all cores, which is conveniently used as the sense winding. The X- and Y-selecting windings must then be threaded through the holes. The individual "cores" require a fairly small (330 ma) current to drive them to saturation, and the switching time is about the same as for ordinary toroidal cores. This again is a form of matrix that could presumably be mass-produced quite cheaply.

12-12. Other Array-type Memories. In principle, an array of binary elements of any type can be used in the construction of a parallel memory, the relevant considerations being reliability, access time, ease of construction, complexity of the circuitry needed for selection, reading, and writing, and cost. As the arrays of magnetic cores satisfy these criteria reasonably well and have reached an advanced stage of development, a good deal of space has been devoted to them. The present chapter, in conclusion, calls attention to two other types of array.

In Sec. 5-8 the ferroelectric memory cell has been discussed. Pulvari has built up arrays of these cells by using a thin sheet of the ceramic material, with sets of parallel electrodes on the two sides, those on one side perpendicular to those on the other, so that the material lying between electrodes at each point of the grid forms a separate memory cell. An experimental 10×10-element array has been formed and successfully operated, but no full-scale memory has yet been constructed on this principle.

In Sec. 6-3 the diode-capacitor memory cell developed at NBS by A. W. Holt has been described.[2,3] Fairly substantial arrays of these elements (for example, 256 words of 45 bits each) have been built at NBS and operated successfully. The operating characteristics are good, such as large output voltage (about 1 volt) and short memory cycle (down to 1 μsec), but the array suffers from the disadvantages of using a very large number of crystal diodes (two diodes per bit of memory) and consequent high cost.

[1] J. A. Rajchman, Ferrite Apertured Plate for Random Access Memory, *Proc. Eastern Joint Computer Conf., 1956*, pp. 107–115, American Institute of Electrical Engineers, New York, 1957.

[2] R. J. Slutz, Memory Studies at the National Bureau of Standards, *NBS Rept.* 2362, March, 1953.

[3] A. W. Holt, An Experimental Rapid Access Memory Using Diodes and Capacitors, Proceedings of the ACM, Toronto meeting, 1952.

CHAPTER 13

THE ARITHMETIC UNIT

13-1. Introduction. The preceding 12 chapters have been devoted to showing how the various functions needed for the performance of digital computation can be realized physically. It will now be shown how the main units of a computing machine can be built up of the elements that have been described. These units will be called the memory, the arithmetic unit, the control, and the input-output unit. They are interconnected in complicated fashion, with a great abundance of both forward-transmission and feedback paths of information flow. It is, therefore, difficult to decide with what unit to begin, and the choice must be somewhat arbitrary. It might appear that the control would be the most reasonable starting point, but the arithmetic unit has been chosen because it is the easiest to understand.

In Chap. 10 both parallel and serial adders have been described. Generally speaking, the adder[1] is the heart of the arithmetic unit. Subtraction is performed conveniently by the addition of the complement of the subtrahend, and multiplication and division are carried out by causing the adder (or adders) to execute certain programs of additions and subtractions (cf. Chap. 1). Little is to be gained by attempting to deal with such processes with any degree of generality. The course that has been chosen is to describe in some detail the arithmetic units of a parallel and of a serial machine; it is felt that the reader who succeeds in understanding these will be able to understand other machines without great difficulty.

13-2. The IAS Arithmetic Unit: Registers and Adder.[2] The principles of the double-rank shifting registers and of the parallel adder and digit resolver of the Institute for Advanced Study (IAS) computer have been discussed in Chaps. 8 and 10, to which the reader is referred. The arithmetic unit consists of two shifting registers RI and RII, a third register $RIII$ which does not have the shifting property, the adder, the

[1] Or, of course, the "subtractor," as, e.g., in EDSAC II, University Mathematical Laboratory, Cambridge, England.

[2] H. H. Goldstine, J. H. Pomerene, and C. V. L. Smith, "Final Progress Report on the Physical Realization of an Electronic Computing Instrument," chaps. 1, 4, and 6, Institute for Advanced Study, Princeton, N.J., 1954.

digit resolver, the control circuits, and various gates (e.g., the "complementing" gates).

From Fig. 8-1 and the accompanying text, it is seen that, in principle, the double-rank registers consist of two ranks of 40 toggles each with gates by which information can be transferred (*a*) "up" from any toggle of the lower rank to the corresponding toggle of the upper rank, and (*b*) "down" from any toggle of the upper rank to the one immediately to the left or to the right of it and in the lower rank. To effect any of the possible transfers of information, the toggles of the recipient rank must first be cleared to 0 or to 1 by pulsing the appropriate bus B_i from 150 volts down to 50 volts for 1 μsec.

The registers RI and RII retain the two sets of "down" gates D_R and D_L and the "up" gates U_1. The gates U_0, however, are driven differently in the two registers: (*a*) in RI they receive one of their inputs from the digit resolver instead of from the toggles of the lower rank R_1, and (*b*) in RII they receive their inputs from the left-hand grids of the toggles of the upper rank R^3 of RIII instead of from those of the lower rank R_2 of RII. From RI and RII, also, certain outputs are taken: (*a*) from the right-hand grids of the toggles of the lower rank R_1 to the adder, and (*b*) from the right-hand grids of R_2 to gates which, when opened, transfer information into the toggles of R^3.

As to RIII, it consists of two nonshifting registers, which are merely sets of 40 toggles. Associated with these are several sets of gates, of which four are located much as the U and D gates are in RI and RII. Three of these are connected as follows: (*a*) U_1 from the Williams memory to the right-hand plates of R^3; (*b*) U_0 from the right-hand grids of R_2 to the left-hand plates of R^3; (*c*) D_R from the Williams memory to the left-hand plates of R_3.

The fourth set, the D_L (or "dispatch") gates, are used to convey orders to the control from R_3. Thus, (*a*) from stages 0 to 9 and 20 to 29 of R_3, address information is transmitted to the address generator (a part of the dispatch counter which is physically located in the Williams memory), and (*b*), from stages 10 to 19 and 30 to 39, the parts of orders that specify the operation to be performed are transmitted to the instruction-synthesizer circuit of the main control. The address information (*a*) is transmitted via the dispatch gates, which are associated closely with the shift counter, for the first 6 bits of address information (i.e., bits 0 to 5 or 20 to 25) are in some cases inputs to the recognition circuit, where they are compared with the number in the shift counter.

The fifth set of gates constitute the path through which information is transmitted from R^3 to the adder. By means of these, either the number in R^3 or its complement can be transmitted to the adder. These gates can also be "nulled," so that 0 is transmitted to the adder no matter what number is in R^3. They are descriptively known as "complement gates."

There is no communication between R^3 and R_3. R_3 is not a part of the arithmetic unit, but belongs rather to the main control. In it are placed the order words as they emerge from the Williams memory; from it address information is transmitted to the address generator, and instruction information is made available to the instruction synthesizer, as explained above. R^3, on the other hand, receives number words from the Williams memory and transmits them to the arithmetic unit by way of the complement gates. Because of their functions, RI, RII, and RIII are called, respectively, the "accumulator register," the "arithmetic register," and the "memory register."

Certain other paths exist for the transfer of information between registers: (*a*) from 2^0R^1 to $2^{-39}R_2$,* (*b*) from $2^{-39}R^1$ to 2^0R_2. The necessity of providing such paths becomes obvious in connection with such operations as shifting, multiplying, and retaining the full-length product. RI is provided also with an extra toggle 2^1RI in the 2^1 position; this can, e.g., be used to save the sign (2^0) bit if a shift of one position to the left is performed, the paths being $2^0R^1 \rightarrow 2^1R\text{I} \rightarrow 2^0R_1$. Sign information can also be communicated to this toggle under certain circumstances (in multiplication) from the sign-position toggle of the discriminator in the Williams memory (information is read from the Williams tubes to a set of 40 toggles, whence it is transmitted to RIII). Also, sensing circuits transmit to the arithmetic control the information held in certain toggles such as those in sign positions and in $2^{-39}R_2$; the position $2^{-39}R_2$ plays a central role in the performance of the multiplication process.

In Chap. 8 it has been shown how shifting in RI and RII is accomplished by sequences of clearing and gating operations. For each register there are clear drivers and clear-driver drivers, one for each of the possible operations Cl $R^i \rightarrow 0$, Cl $R_i \rightarrow 0$, Cl $R^i \rightarrow 1$, Cl $R_i \rightarrow 1$. Which of these is to be performed is determined by a clear selector on the basis of information received from control circuits such as the left/right selector, which ultimately receives its instructions from the main control. Each clear selector basically consists of two 6J6's and some diodes, one 6J6 to determine whether the lower rank of the register is to be cleared to 0's or to 1's and the other 6J6 for the top rank. Under quiescent conditions the control grids of the 6J6 are held at -40 volts and the cathode at $+10$ volts, so that both sections of the tube are cut off, and the plate voltage is clamped at $+150$ volts by 6AL5 diodes, the grid voltages being supplied from a so-called "right/left" circuit in the control. When a shift is called for, the right/left circuit raises the appropriate grid voltage to 0 volts and holds the other at -40 volts; then the cathode of the 6J6 is dropped from $+10$ to -10 volts, and the section of the 6J6 with grid at

* These expressions denote bit positions in the ranks of the registers. 2^0R^1, for example, is the 2^0 place in the upper rank of register RI. Note the special symbol "2^1RI" for the extra toggle.

0 volts conducts, the plate voltage falling to about 50 volts. Each plate drives a clear driver, which is a cathode follower consisting of five 5687's connected in parallel with their common cathode connection supplying the clear bus with plate current for all the left (or right) plates of the toggles of one rank. Merely to clear a rank to 0, in case no shift is to be called for, an alternative connection is made by a gate which is able to draw sufficient current from one plate resistor of the 6J6 to pull the plate voltage down to 50 volts; the signal for this is the coincidence of two pulses (synch and clear) from the control.

The gates are operated by voltages supplied by cathode followers called "gate drivers"; the voltage is +10 volts to disable and −20 volts to enable. The gate drivers are themselves driven by other cathode followers (gate-driver drivers). Which set of gates is to be enabled is in each case determined by the gate-selector circuits, acting upon instructions received from the arithmetic control.

Besides the U and D gates, there are complement gates by which information is transmitted from R^3 to the adder. These are forty 6J6's with appropriate drivers. Each 6J6 has its left and right grids, respectively, connected to the corresponding grids of a toggle of R^3, its cathode connected to the cathode of the tube in the corresponding stage of the adder and thence through a resistor to a negative voltage (−174 volts), and its plate connected to a cathode of a 5687. The 5687 determines whether the number or the complement is to be transmitted; its grids are at either +90 or −30 volts and are driven by the complement-gate selector and magnitude/number circuit, which can be considered part of the arithmetic control. Suppose that the toggle of R^3 under consideration holds a 0, so that the left and right grids are at 0 volts and −40 volts, respectively. If the left grid of the 5687 is at +90 volts and the right at −30 volts, the cathode of the 6J6 is held up to 0 volts, and 0 is transmitted to the adder; if the grid voltages are interchanged, the 6J6 cathode falls to −40 volts, and the adder input is 1. If both grids of the 5687 are held at +90 volts, then certainly the 6J6 cathode is held under all circumstances at 0 volts, and the adder input is 0 irrespective of the contents of the R^3 toggle. Which of these possible combinations of grid voltages is to be applied to the 5687 is determined, as mentioned above, by the complement-gate selector, acting upon instructions from the main control and other elements of the arithmetic control. The complement formed by the method outlined is a 1's complement; to compensate for this the complement-gate selector injects a carry into the lowest-order stage of the adder whenever it causes a complement to be transmitted.

The above discussion gives a fairly complete account of the way in which the registers of the arithmetic unit are connected. There is no need to say anything here about the adder and digit resolver, for they

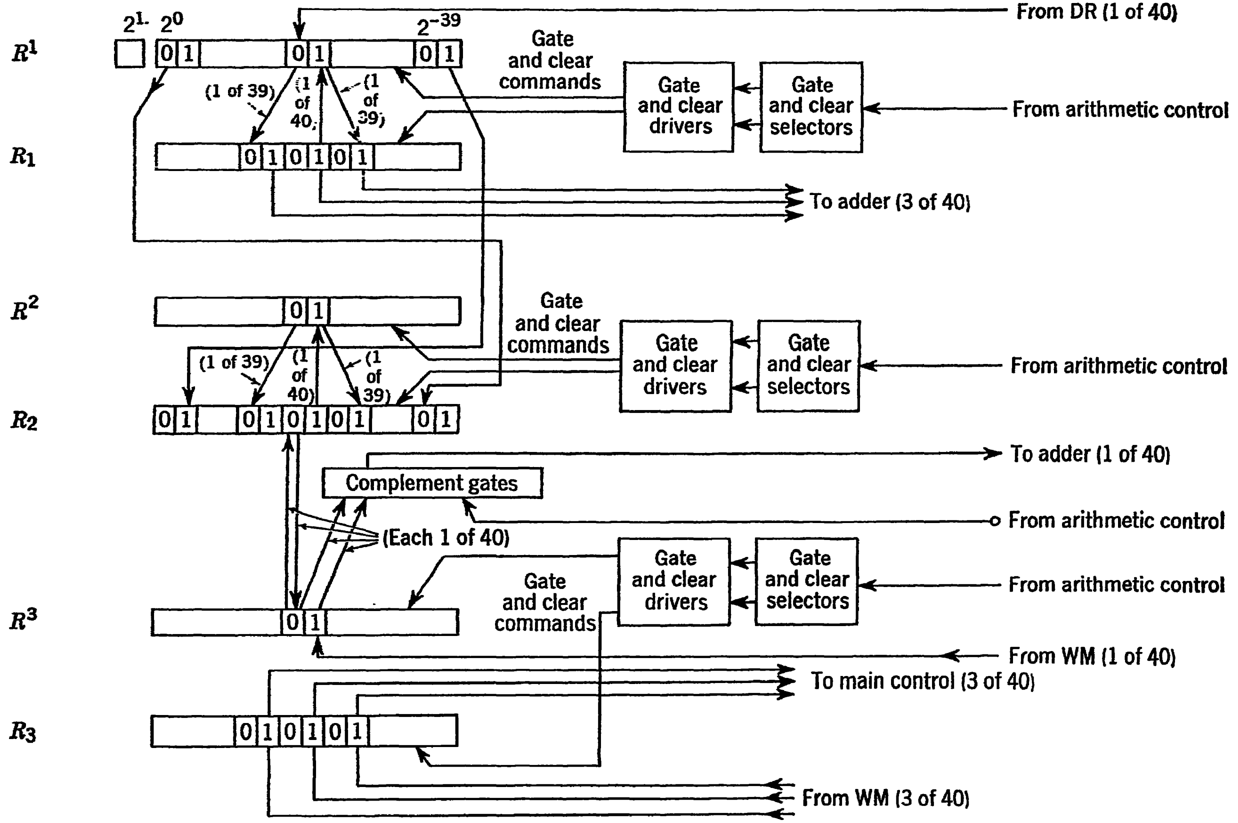


FIG. 13-1. Interconnections of registers of arithmetic unit of the IAS computer.

have been treated rather fully in Chap. 10. Figure 13-1 summarizes in block-diagram form the discussion of the arithmetic unit up to this point.

13-3. Arithmetic Control. Although it is customary, in drawing the over-all block diagram of a digital computer, to insert one block labeled "control," it is a mistake to think of all the control functions as centralized in a single unit. Actually, many are so very closely associated with the arithmetic unit and with the memory that they may reasonably be considered parts of these units. Other functions, concerned chiefly with the interpretation of instructions and the consequent transmission of signals to stimulate the arithmetic unit, the memory, and the input-output unit, are central to the functioning of the whole machine and are best thought of as constituting a unit which may be described as the main or central control. The circuits intimately associated with the arithmetic unit will be regarded here as part of that unit and spoken of as constituting the arithmetic control.

It has been shown in Chap. 10 that the adder forms the sum of the number in R_1 (resident number) and the number in R^3 or its complement (incident number). The adder continuously forms the sum of its inputs, and the digit-resolver output can be gated into R^1 as desired. Multiplication and division are performed as sequences of adding or subtracting and shifting operations. As the adder and digit resolver need no stimulus but automatically and continuously form the sum of their inputs, it is clear that the main features of the arithmetic control must be (1) the circuits that generate the gate and clear voltages that cause the shifts to be performed, and (2) the circuits that keep count of the numbers of shifts performed and cause the process to terminate when the number of shifts appropriate to it have been executed. These circuits comprise (1) the shift counter and recognition circuits and (2) the gate-and-clear sequencing chain.

13-4. The Shift Counter and Recognition Circuit. The shift counter has already been described in Chap. 9. It consists of two ranks of six toggles with the necessary gates connecting them, and each count requires two inputs on separate lines in a definite time sequence. The output to the recognition circuit consists of six wires, one from the lower ("true") toggle of each stage, where 0 is signified by -40 volts and 1 by 0 volts. The number with which this output is compared is derived either from R_3 or from the main control and signifies the number of shifts that must be performed; it is presented to the recognition circuit in the form of six voltages, where 0 is signified by 0 volts and 1 by approximately -20 volts (of these the 0 volts is quite definitely determined, but the -20 volts could be as large as about -5 volts without prejudicing the operation of the circuit).

A logical diagram of the recognition circuit is shown in Fig. 13-2.

Neglecting for the moment the inputs from the main control, it is seen that the inputs consist of a 6-bit number to be recognized NR_0 to NR_5 and the contents of the shift counter SC_0 to SC_5. In addition to the familiar mixers and gates, there are three inverters (or complementers). The output can be written down very easily in the following form:

$$\begin{aligned}\text{Output} = [(SC_0' \wedge NR_0) \vee (SC_1' \wedge NR_1)]' \\ \wedge [(SC_2' \wedge NR_2) \vee (SC_3' \wedge NR_3)]' \\ \wedge [(SC_4' \wedge NR_4) \vee (SC_5' \wedge NR_5)]'\end{aligned}$$

Before the shift counter is used it is naturally cleared to 0; so initially

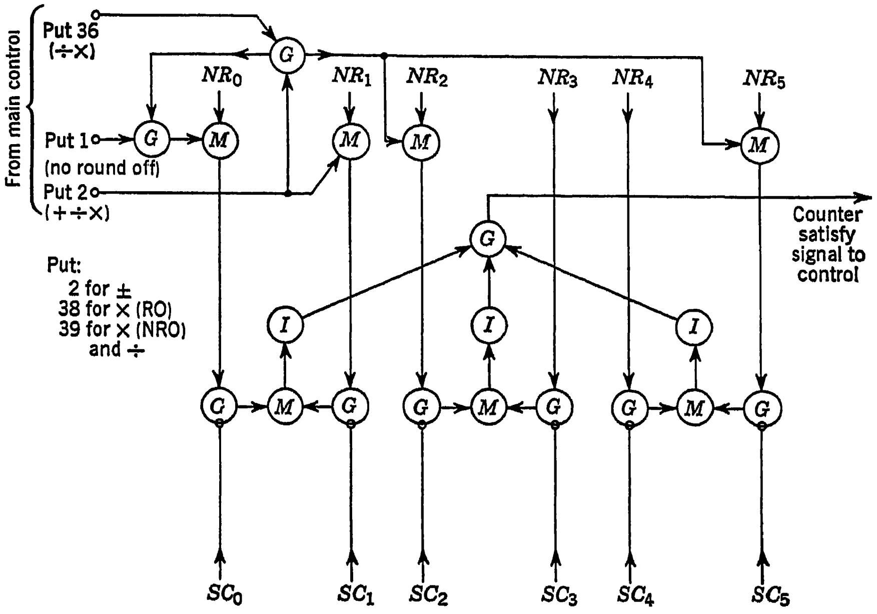


FIG. 13-2. Recognition circuit.

$SC_i = 0$ for $i = 0$ to 5 and, before the counting begins, some number to be recognized appears at the input. Evidently, since the number to be recognized is greater than 0, at least one of the $NR_i = 1$; so at least one of the $SC_i' \wedge NR_i = 1$. Therefore, at least one bracketed term

$$[(SC_i' \wedge NR_i) \vee (SC_{i+1}' \wedge NR_{i+1})] = 1$$

and its complement is 0. Since the product contains at least one factor equal to 0, evidently the output is 0. Now suppose that the counting has started and that the number in the counter has not yet attained equality with the number to be recognized. In the number to be recognized, consider the highest-order column that contains a 1. The corresponding column of the smaller number can contain either a 1 or a 0. If it contains a 1, examine the column of next lower order; in this column, either both numbers contain 0's, both contain 1's, or the larger contains a 1 and the smaller a 0. Since this argument can be repeated as often as necessary, the conclusion is reached that, if corresponding bits are compared beginning with those of highest order, the first disagreement must always be due to a 1 in the larger number and a 0 in the smaller; in positions of order lower than that in which the first disagreement occurs, obviously anything can happen, but this is, clearly, of no concern whatever. Hence it is evident that, before coincidence is reached, (*a*) at least one $SC'_i \wedge NR_i = 1$, and, therefore, (*b*) at least one bracketed term

$$[(SC'_i \wedge NR_i) \vee (SC'_{i+1} \wedge NR_{i+1})]' = 0$$

so (*c*) the output = 0. Now, when coincidence is reached, in every case $NR_i = SC_i$; so each product $SC'_i \wedge NR_i = 0$, and the output is given by

$$(0 \vee 0)' \wedge (0 \vee 0)' \wedge (0 \vee 0)' = 1 \wedge 1 \wedge 1 = 1$$

Hence the "counter-satisfy" signal to the main control is 0 until coincidence is achieved, when it becomes 1. As this signal stops the process whose successive steps are being counted, the question of what the output would be if the number to be recognized were less than the number in the shift counter is irrelevant. The output-signal level is +10 volts until coincidence is reached; then it falls to about −27 volts with a rather wide tolerance.

Now consider the inputs from the main control. These are used when arithmetic processes are to be performed. In these cases each $NR_i = 0$, and it is possible to insert 2, 38, or 39 according as the process is addition/subtraction or multiplication/division; each count corresponds to a transfer first into R^1 and then from R^1 down to R_1.

13-5. The Sequencing Chain. The gate-clear sequencing chain is a circuit that generates clearing and gating signals in the proper sequence to effect transfer of information into the register RI and shifting of information already held there. From the previous discussion, for example, it appears that the U_0 gates in RI are used to insert information from the digit resolver into R^1. Referring back to the discussion in Chap. 8, it is seen that these gates can insert only 0's in R^1, so that before they are opened it is necessary to clear R^1 to 1's. Suppose now that this transfer has been accomplished and that it is desired to effect the transfer from R^1 to R_1. This can be accomplished by a transfer down to the right (clear R_1 to 0's, followed by shift down right of 1's) which records the number in R_1 shifted one place to the right; this transfer must be followed by a left shift of one place if the contents of R_1 are to be an exact copy of the contents of R^1. In multiplication the combined transfer and shift are exactly what is desired.

The possible sequences of clears and gates are listed for convenience:

Record (accept): Cl $R^i \rightarrow 1$, U_0 (transfer up 0)
Shift up (reject): Cl $R^i \rightarrow 0$, U_1 (transfer up 1)
Shift down left: Cl $R_i \rightarrow 1$, D_L (transfer down left 0)
Shift down right: Cl $R_i \rightarrow 0$, D_R (transfer down right 1)

Which of these sequences is to be performed is dictated by certain inputs from the control acting in conjunction with signals fed back from the registers being controlled. The operation of the whole circuit is enabled when the recognition output described above is at its high value of +10 volts, and it is inhibited when this output falls to its low value of −27 volts. Thus, generally speaking, once the circuit is set in operation, it generates clearing and gating signals, the results are fed back to it, and the action once initiated is self-perpetuating until the number of shifts performed coincides with the number-to-be-recognized input to the recognition circuit, at which point the fall of the recognition output inhibits further action.

In terms of circuits, the sequencing control is rather complicated. Nothing would be gained by presenting the detailed schematic, but the operation can be understood from the block diagram, Fig. 13-3. There are three toggles T_1 to T_3 of which T_1 and T_2 are exactly as shown in Fig. 5-1, whereas T_3 is somewhat more elaborate—but the details are not important for understanding the operation of the circuit and so will not be discussed. The plates of T_1 and T_2 obtain their d-c supply from the "clear" buses, as shown, and their states can be changed by the fall of the gating voltages from their high (+10-volt) level to their low (−10-volt) level; these voltages become effective when they fall below a reference level and permit current to be drawn from the plate in question by the puller tubes, as shown. The outputs are taken from the grids of T_1 and T_2; the grid of the conducting tube is at 0 volts and that of the nonconducting tube at −40 volts. However, these voltages drive other tubes to provide both inversion and voltage translation; so the "direct" output is +110 volts when the toggle grid is at 0 volts, and +60 volts when it is at −40 volts, whereas the corresponding "inverted" outputs are −29 and +7 volts, respectively. The circuits used to provide these outputs are simple and straightforward and are not worth discussing, for the reader can easily deduce their form. The tubes labeled G_1 to G_{12} are dual triodes of the common-cathode type, connected as cathode followers (see, e.g., Fig. 4-5). In G_3 to G_6 the outputs are taken from a tap between the cathode and the negative supply voltage. Hence G_1 and G_2 give output voltages +7 or −29 volts, depending of course on the higher grid; but G_3 to G_6, whose grid voltages are either +60 or +110 volts, give −18 or +8 volts; and, finally, G_7 to G_{12} give either +8 or −18 volts. The reader must understand that these are merely nominal values which need not be maintained with precision; thus in the last case cited there is no need to worry about the possibility that +7 volts might be obtained under certain circumstances instead of +8 volts. In fact, the circuits operating can be described quite simply by logical expressions in which "low" is designated by 1 and "high" by 0. As to the remaining circuit elements, M is a diode mixer with the common anode so connected that, when at least one input is low, the left grid of T_3 is pulled down so far that conduction is inhibited; and D is a single diode which, similarly, can

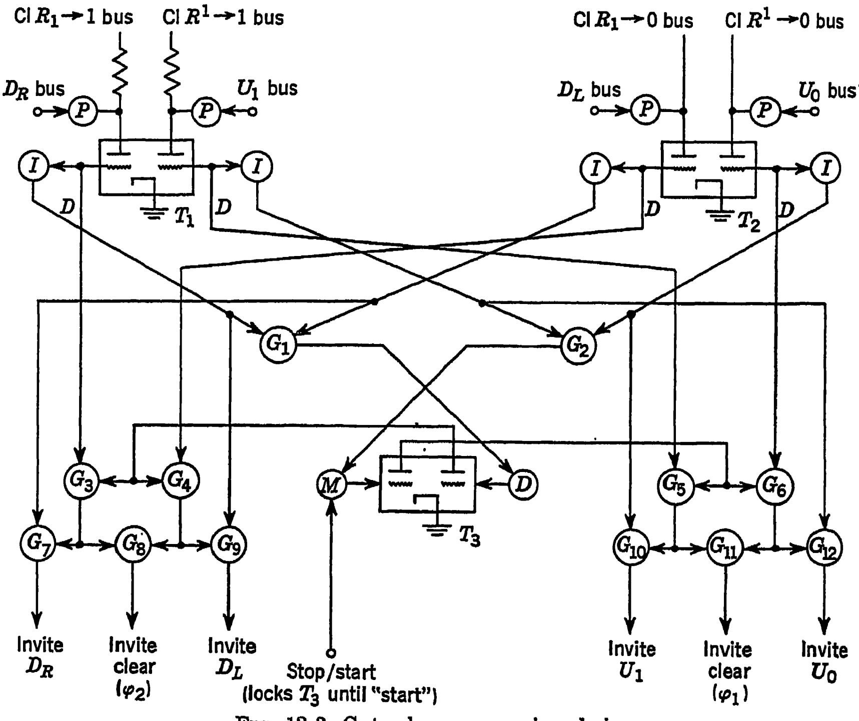


FIG. 13-3. Gate-clear sequencing chain.

inhibit conduction in the right tube (actually a section of a dual triode); thus, if both M and D are properly stimulated, both sections of T_3 are cut off.

Let t_1 and t_2 represent the states of the toggles T_1 and T_2, with the convention that the toggles hold 1 when the right grids are low. Clearly the direct and inverted outputs from the right grid of T_1 are the t_1 and t_1', and those from the left grid are t_1' and t_1; and similarly for T_2. For T_3, let the 1 state be that in which the left section is conducting, and let t_3 represent the information taken from the left plate, so that $t_3 = 1$ (low) in the 1 state and 0 (high) in the 0 state. For the tubes G_1 to G_{12}, let the outputs be represented by g_i. The outputs from the six tubes in the lowest row can be represented as follows:

$$\begin{array}{lll} g_7 = t_1' \wedge t_2 \wedge t_3' & g_8 = t_1' \wedge t_2' \wedge t_3' & g_9 = t_1 \wedge t_2' \wedge t_3' \\ g_{10} = t_1 \wedge t_2' \wedge t_3 & g_{11} = t_1 \wedge t_2 \wedge t_3 & g_{12} = t_1' \wedge t_2 \wedge t_3 \end{array} \qquad (13\text{-}1)$$

provided T_3 is free to behave as a toggle, which it is not initially, as will be shown below.

Assume that initially the following conditions hold: (*a*) $t_1 = 1$, (*b*) $t_2 = 1$, (*c*) the start/stop voltage is sufficiently negative to pull the left grid of T_3 so far below ground that the left section is forced into the cutoff condition. Clearly, $g_1 = t_1 \wedge t_2 = 1$ and $g_2 = t_1' \wedge t_2' = 0$; so the other input to M is high, and the negative output from G_1 applied to diode D holds the right grid of T_3 also down below cutoff. Thus, initially, T_3 is not acting as a normal toggle, and both its plate voltages are high. In this condition information can still be obtained from the expressions g_7 to g_{12} if it is agreed that t_3 and t_3' are both to be replaced by 0. If this is done, all the expressions (13-1) are seen to have the value 0; therefore, all the outputs are high, i.e., about +8 volts.

The start command consists in raising the start/stop voltage to 0. As the other input to the mixer has already been seen to be high ($g_2 = 0$), the left grid of T_3 rises to 0 volts, and the left section of T_3 conducts; t_3 now becomes 1, and t_3' is properly interpreted as 0. By inspection of (13-1) it follows that $g_{11} = 1$ and that all other $g_i = 0$. Hence the consequence of starting the chain is to drop one output, which is labeled "ϕ_1 clear invite," to −18 volts; the ϕ_1 signifies the first of the two phases, or sequences of clear and gate, that constitute a single shift. The clear-invite output is connected first to a simple delay unit, of which one output is identical with the input but the other is somewhat delayed. These in turn go to the accept/reject selector, which transmits either the undelayed step voltage to the clear driver that causes R^1 to be cleared to 0's or else the delayed step voltage to the clear drivers that cause R^1 to be cleared to 1's. The latter occurs when the adder output is to be entered into RI (or accepted); the delay has been introduced to give the adder adequate time for carry propagation and thus to ensure that the correct values of the adder outputs have been attained before its contents are entered into RI. More will be said about the accept/reject selector later on. The ϕ_1-clear-invite voltage also goes to the RII clear selector. The signals that determine the use to be made of the ϕ_1-clear-invite voltage are derived from the main control and are determined by the type of operation called for by the instruction that is being obeyed.

Suppose that R^1 is cleared to 1's as it is when a simple addition is performed. The Cl $R^1 \rightarrow 1$ bus supplies the plate voltage to the right section of the tube in T_1; so the right section is turned on and the left off; the right grid now assumes its high value, and $t_1 = 0$. Both g_1 and g_2 are now 0; T_3 is not switched, but the six outputs become

$$g_7 = 0 \quad g_8 = 0 \quad g_9 = 0 \quad g_{10} = 0 \quad g_{11} = 0 \quad g_{12} = 1$$

So the clearing operation is terminated, and the invite U_0 signal is sent out; this stimulates the gate drivers and in turn causes the U_0 gates in RI and RII to be enabled. (Actually the RI/RII-gate selector can modify the action in RII but not in RI; its function is described later on.) This completes the first phase ϕ_1 of the shift.

The U_0-enabling voltage is fed back to pull down the plate voltage of the right section of the tube in T_2 and so to switch T_2 to 0; g_1 remains 0 but g_2 becomes 1; the left grid of T_3 is pulled down, and, as the right grid is not constrained, T_3 switches to 0. Hence, the outputs become

$$g_7 = 0 \quad g_8 = 1 \quad g_9 = 0 \quad g_{10} = 0 \quad g_{11} = 0 \quad g_{12} = 0$$

and the ϕ_2-clear-invite signal is signified to the clear selectors (both for RI and for RII) by the fall of this voltage from its positive level (about +8 volts) to its negative level (about −18 volts). Whether this causes R_1 and R_2 to be cleared to 0's or to 1's depends upon signals received by the clear selectors from the main control; assume that clearing to 1's is called for. The fall of the Cl $R_1 \rightarrow 1$ bus switches T_1 to 1; so, since now $t_1 = 1$ and $t_2 = 0$, g_1 and g_2 are both 0, T_3 remains in its 0 state, and the six output voltages become

$$g_7 = 0 \quad g_8 = 0 \quad g_9 = 1 \quad g_{10} = 0 \quad g_{11} = 0 \quad g_{12} = 0$$

The clear is terminated and the invite-D_L signal is sent to the gate drivers. The fall of the D_L-enabling voltage is fed back to switch T_2 to 1, which makes $g_1 = 1$ and $g_2 = 0$, and so T_3 is set to 1. The process now recommences unless the stop/start voltage is simultaneously dropped to its negative value; in this case the circuit returns to its quiescent condition and is locked there until the next start command is received. It will be recalled (see Chap. 9) that the shift-counter input is a pair of pulses per count. The first of each pair is taken from the ϕ_1-clear-invite lead; the second is obtained from the RI/RII-gate selector and coincides in time with invite U_0. As the lengths of the negative pulse outputs of the sequencing chain are about 1 μsec and as they are separated by about 1 μsec, the shift counter is given time to settle down during phase 2 of the shift. Apart from the deliberately inserted delay, the timing of the chain depends upon the charging and discharging of the various distributed capacitances associated with the circuit and the circuits driven by it.

13-6. Arithmetic Processes and the Arithmetic Control. Before discussing the remaining circuits of the arithmetic unit, it will be useful to summarize the processes by which the arithmetical operations are effected.

Addition is the basic operation. The adder-and-digit-resolver output is the sum of the contents of R_1 and of the number in R^3, its negative, or 0, depending upon the way in which the complement gates are set up. This sum must be put back into R_1 via R^1. As the "down" gates shift down left and right but not down directly, this is accomplished by the "zigzag": down left, up, down right. Thus the following sequence of clear and gate operations must take place: Cl $R^1 \rightarrow 1$, U_0; Cl $R_1 \rightarrow 1$, D_L; Cl $R^1 \rightarrow 0$, U_1; Cl $R_1 \rightarrow 0$, D_R. Each complete cycle of the gate-clear sequencing chain produces a sequence clear, gate, clear, gate; so two such cycles are needed and the number to be recognized by the shift counter is 2 (it has been seen how this is presented to the shift counter, having originated in the main control).

Multiplication consists of successive additions, one for each non-sign bit of the multiplier. The multiplier is first brought from the memory to R^3 and is then transferred to R_2; subsequently the multiplicand is brought from the memory to R^3. Where both factors are positive, the process consists in first examining the lowest-order (2^{-39}) bit of the multiplier, then adding the multiplicand to the contents of R_1 (initially 0) if this bit is 1 or doing nothing if it is 0, and shifting the contents of R_1 right one place; the contents of R_2 are shifted right one place at the same time. This process is then repeated for the new occupant of $2^{-39}R_2$, and so on, until all 39 non-sign bits of the multiplier have been examined. The bits shifted out of the right end of R_1 are transferred to the 2^0 position of R_2, and so, after 39 steps, the complete product fills R_1 and R_2; the multiplier has been lost but the multiplicand remains intact in R^3. If only a rounded product is desired, the process continues for 38 steps, then 1 is added to the contents of R_1, and a final right shift is performed. At each stage the sum is automatically shifted one place while being transferred from the digit-resolver output to R_1, for the clear-gate sequence is Cl $R^1 \rightarrow 1$, U_0, Cl $R_1 \rightarrow 0$, D_R. Hence the number to be recognized that is presented to the recognition circuit associated with the shift counter is 38 for a rounded product or 39 for an unrounded one.

For nonpositive factors, recall from Chap. 1 that certain corrections are needed because of the representation of negative x by $2 + x$. In fact, if the multiplier x is negative but the multiplicand y is positive, the procedure is to form $(x + 1)y = xy + y$, ignoring the sign bit of the multiplier, and then to correct by subtracting y, which remains available in R^3. On the other hand, if $x > 0$ but $y < 0$, the product formed is

$$x(y + 1) = xy + x$$

and the correction cannot wait until the conclusion of the process, by which time x has been discarded. Fortunately, as shown in Chap. 1, the correction can be effected bit by bit, the successive partial products being formed according to the formula

$$2p_i = p_{i-1} + \bar{y}_i$$

where
$$\bar{y}_i = \begin{cases} 1 & \text{if } x_{40-i} = 0 \\ y + 1 & \text{if } x_{40-i} = 1 \end{cases}$$

which leads to the product

$$xy + 1 - 2^{-39}$$

which can be corrected by adding $1 + 2^{-39}$, since $1 + 1$ gives a 0 in the 2^0 place and the carry is discarded. Finally, if both x and y are negative, the -icand correction is applied bit by bit, adding $1 + 2^{-39}$ at the end, and the -ier correction is made by subtracting y as before. Formulated as a rule covering all cases (see page 18), this reads:

At each stage, if $2^{-39}R_2 = 1$, add to the contents of R_1 the contents of R_3 with the sign bit replaced by 0 if it is a 1; if $2^{-39}R_2 = 0$, add the sign bit of the -icand to the contents of R_1; and, in both cases, transfer the sum so obtained into R_1 shifted one place to the right. Finally, if the -ier is negative, subtract the -icand; and, if the -icand is negative, add $1 + 2^{-39}$ to the contents of R_1.

As will be seen below, when $y < 0$, instead of adding $y + 1$ whenever $x_{40-i} = 1$, the actual procedure used is to add $2 + y$ (the normal machine representation of negative y) and to inject the sign bit into the 2^0 adder stage via 2^0R_1; this causes a carry out of the 2^0 stage, which does not matter, and in effect replaces the sign bit by 1 as far as the adder is concerned. Thus, in general, at every step in the multiplication the sign bit of the -icand is injected via 2^0R_1; if the -icand is positive, this does no harm and, if it is negative, the effect is either to compensate for this fact ($x_{40-i} = 0$) or to cause $y + 1$ to be added instead of $2 + y$ ($x_{40-i} = 1$). During the multiplication the sign bit of the -icand is held in 2^1RI, from which it is easily injected into 2^0R_1 before each addition.

Finally, consider the division of x by y, where $|x| < |y|$. The procedure given in Chap. 1 calls for the successive calculation of r_i and p_i, where $r_0 = x/2$ and $p_0 = 0$, by the following rules:[1]

$$r_i = 2r_{i-1} - (\text{sign } xy)yp_{i-1}$$
$$p_{i-1} = \begin{cases} 0 & \text{if sign } r_{i-1} \neq \text{sign } [2r_{i-1} - (\text{sign } xy)y] \\ 1 & \text{if sign } r_{i-1} = \text{sign } [2r_{i-1} - (\text{sign } xy)y] \end{cases}$$

where sign $A = 1$ if $A > 0$, but $= -1$ if $A < 0$. From the p_i the q_i are formed by

$$q_i = \begin{cases} p_i & \text{if sign } xy = 1 \\ 1 - p_i & \text{if sign } xy = -1 \end{cases}$$

[1] Goldstine, Pomerene, and Smith, *op. cit.*, pp. 17–21.

and the q_i are the bits of the quotient. This is carried out for q_0 to q_{38}, and q_{39} is always made 1.

Basically, the process consists of left shifts and additions or subtractions. The quotient is formed bit by bit, and these bits are fed into the low-order end $(2^{-39}R_2)$ of R_2 as they are generated. RII and RI are shifted left together, so that, after 39 bits q_i have been formed and 1 has been inserted into $2^{-39}R_2$, the rounded quotient is held in R_2, and twice the remainder is held in R_1. Initially the dividend x is placed in R_1 and the divisor y in R^3. The sign comparisons are facilitated by the fact that x and r_i must always have the same sign.

The rules expressed by these equations can readily be put into words. First, if the signs of dividend and divisor are the same, subtract at every step but, if they differ, always add. Second, if the sign of the tentative partial remainder $2r_{i-1} - (\text{sign } xy)y$ is the same as the sign of the dividend, accept the tentative partial remainder, transmitting it from the digit resolver to R^1 and then shifting it down left to become $2r_i$; but, if the signs differ, reject the tentative partial remainder, and shift the existing partial remainder left one place to become $2r_i$. Finally, if the signs of the tentative partial remainder and the divisor are the same, insert a 1 as a quotient bit into $2^{-39}R_2$, but, if they are different, insert a 0; then shift R_2 left one place. Generate the first 39 bits of the quotient in this way and make the last bit always 1.

In the existing computer, the sign comparison is effected somewhat differently. Let, for example, $1 > y > x > 0$. In the initial subtraction, clearly, $x - y < 0$; so the tentative partial remainder is rejected and 0 is entered as the sign bit q_0 of the quotient. As the number actually formed is $x + 2 - y = 2 + (x - y) < 2$, it is clear that the adder does not overflow. For the next comparison, $2x - y$ is formed; suppose that this is positive. The adder actually forms

$$2x + 2 - y = 2 + (2x - y) > 2$$

so there must be an overflow, for a carry out of the 2^0 adder position is indicated. Therefore, a simple circuit is included to detect the carry out of this position, and the accept/reject rule is modified to read: accept if this carry is not the same as the sign bit of the dividend, and reject if it is the same. Similarly, the rule determining the quotient bit becomes: put 0 into $2^{-39}R_2$ if the carry is the same as the sign (2^0R^3) of the divisor, but 1 if they are different.

Consider now how these rules are made operative by the arithmetic-control circuits. These consist of the sequencing chain, shift counter, and recognition circuits already described, and the left/right selector, accept/reject selector, and multiplication-terminate circuit, together with some minor units such as the carry-delay unit, the gate-and-clear selectors, the complement gates, and the magnitude/number circuit. The arithmetic control receives from the main control inputs specifying the operation (arithmetic or merely left or right shift) to be performed and an instruction which sets the unit in operation at the appropriate time. In Chap. 15 it will be shown how these signals are generated, but for the present it will suffice to note the fact of their existence. The initiation of the arithmetic operation is called for immediately after the transfer of an operand from the memory to R^3; the $M \rightarrow R^3GT$ pulse turns a toggle T_{block} in the main control to its on or 1 position, which produces the enablement mentioned; the arithmetic unit proceeds with the operations; at the conclusion of the operations, a signal fed back to main control turns T_{block} off, inhibiting further action by the chain.

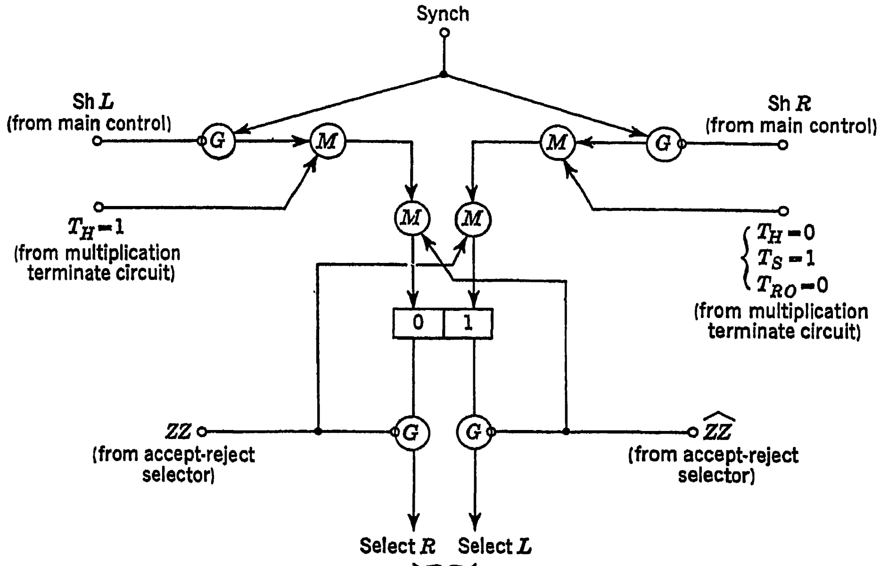
Before the main units of the arithmetic control are described, attention should be called to the carry-delay unit, which is a simple circuit that receives from the sequencing chain the phase-1–clear–invite signal (an abrupt fall in voltage from $+8$ to -18 volts) and transmits it in two forms: (1) an undelayed replica Δ' and (2) a replica Δ delayed by 15 μsec. The delayed clear invite is used to clear R^1 to 1's and so prepare it to receive the digit-resolver output if this output is to be accepted into R^1 (the delay being introduced to allow ample time for the propagation of carries and thus to ensure that the adder output has reached its correct value before it is transmitted into R^1). The length of the delay could be reduced if the carry-propagation time were reduced. It should be recalled that the φ_1-clear-invite voltage remains at its lower (enabling) value until the clear actually generated has been fed back to the sequencing chain and has caused it to change its state.

In the discussion that follows, the reader is referred to the block diagram of the arithmetic control in Fig. 13-7 and to the logical diagrams of the left/right selector, the multiplication-terminate circuit, and the accept/reject selector shown in Figs. 13-4 to 13-7. The accept/reject-selector diagram contains certain elements of other components, e.g., the toggles 2^0R^3, $2^{-39}R_2$, 2^0 shift counter, and the associated gates that generate the zigzag command ZZ and its negative, which are included because they make the operation easier to follow. Various simple minor circuits such as the gate-and-clear selectors are omitted because their functions can easily be understood without recourse to diagrams. It will be appreciated that some details are omitted in order to keep the diagrams reasonably simple, but enough is given so that the essentials of the operations can be understood.[1]

[1] The signals labeled "synch" and $\widehat{WO}$ are obtained from the memory local control and the main control. The affirmation of "synch" indicates that the memory has received instructions to obtain the required operand and deliver it to R^3. The affirmation of WO ("work order") indicates that the memory is to produce an operand;

The mode of presentation will be to follow through each of the arithmetical operations and to show how the elements of the arithmetic control function together to cause them to be accomplished; the order chosen is addition (subtraction), division, and last of all multiplication, because this involves the multiplication-terminate unit, which is inoperative in the others.

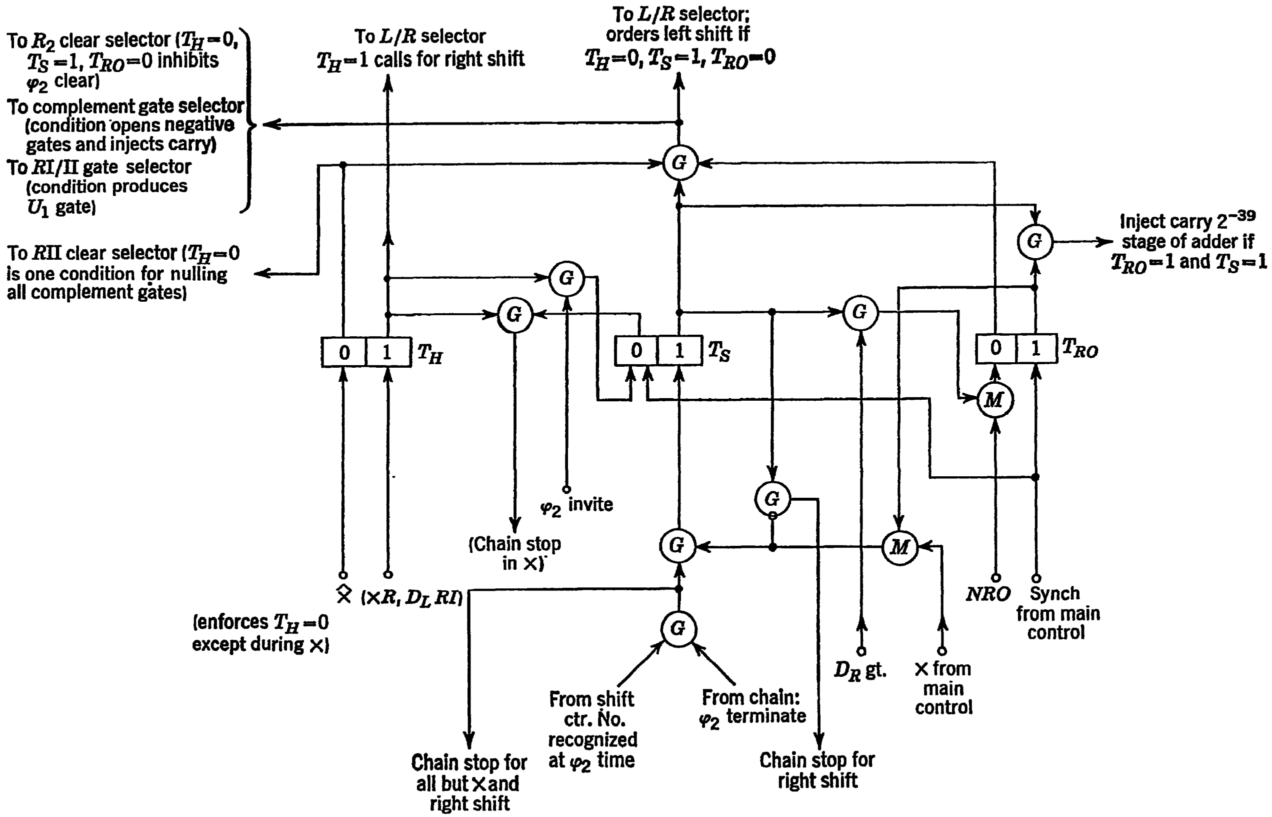
The main control specifies addition or subtraction as opposed to multiplication or division by an output called "$\Sigma/\div\ \times$." This signal is physically negative for Σ but positive for $\div$ or $\times$; in the accept/reject-selector drawing, the affirmation of Σ enables gates out of the 2^0 stage of



L-R selector
FIG. 13-4. Left/right selector.

the shift counter (as a matter of notation $\hat{\Sigma}$ means the denial of Σ, that is, $\div$ or $\times$). The main control also causes 2 to be presented to the recognition gates as the number to be recognized. As between addition and subtraction, the distinction is made by the denial or affirmation of a main-control output called "$-L/+R$"; $-L$ causes the negative complement gates to be opened ($R^3 \rightarrow$ adder), and $+R$ causes the positive (or direct) gates to be opened. In multiplication or division, $-L$ selects division and causes a left-shift command to be transmitted to the left/right selector, and $+R$ selects multiplication and causes the L/R selector to

the negation ($\widehat{WO}$) indicates either that the operation is one in which no reference is made to the memory or that the memory is to transmit a fresh instruction word to R_3.



To R_2 clear selector ($T_H=0$, $T_S=1$, $T_{RO}=0$ inhibits φ_2 clear)
To complement gate selector (condition opens negative gates and injects carry)
To RI/II gate selector (condition produces U_1 gate)
To RII clear selector ($T_H=0$ is one condition for nulling all complement gates)
To L/R selector $T_H=1$ calls for right shift
To L/R selector; orders left shift if $T_H=0$, $T_S=1$, $T_{RO}=0$
Inject carry 2^{-39} stage of adder if $T_{RO}=1$ and $T_S=1$
T_H
T_S
T_{RO}
G
M
0
1
φ_2 invite
(Chain stop in ×)
$\hat{\times}$ (×R, D_L RI)
(enforces $T_H=0$ except during ×)
Chain stop for all but × and right shift
From shift ctr. No. recognized at φ_2 time
From chain: φ_2 terminate
Chain stop for right shift
D_R gt.
× from main control
NRO
Synch from main control

FIG. 13-5. Multiplication-terminate circuit.

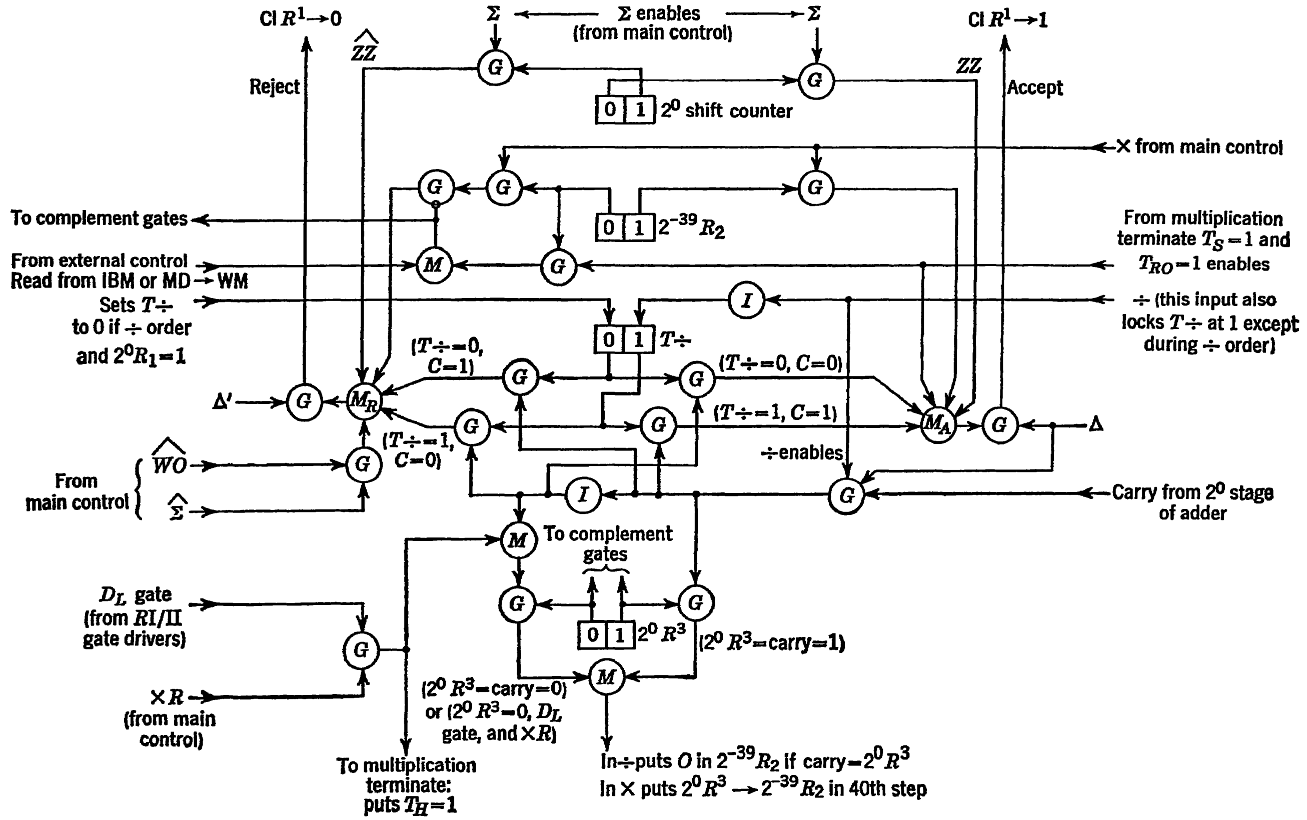

Cl $R^1 \rightarrow 0$
$\widehat{ZZ}$
Reject
Σ
Σ enables (from main control)
Σ
Cl $R^1 \rightarrow 1$
ZZ
Accept
2^0 shift counter
× from main control
$2^{-39}R_2$
To complement gates
From external control Read from IBM or MD → WM
From multiplication terminate $T_S=1$ and $T_{RO}=1$ enables
Sets $T\div$ to 0 if ÷ order and $2^0R_1=1$
÷ (this input also locks $T\div$ at 1 except during ÷ order)
$T\div$
$(T\div=0, C=1)$
$(T\div=0, C=0)$
$(T\div=1, C=1)$
$(T\div=1, C=0)$
Δ'
Δ
M_R
M_A
÷enables
From main control $\widehat{WO}$ $\hat{\Sigma}$
Carry from 2^0 stage of adder
To complement gates
D_L gate (from RI/II gate drivers)
$\times R$ (from main control)
2^0R^3
$(2^0R^3=\text{carry}=1)$
$(2^0R^3=\text{carry}=0)$ or $(2^0R^3=0, D_L$ gate, and $\times R)$
To multiplication terminate: puts $T_H=1$
In ÷ puts O in $2^{-39}R_2$ if carry $=2^0R^3$
In × puts $2^0R^3 \rightarrow 2^{-39}R_2$ in 40th step

FIG. 13-6. Accept/reject selector.

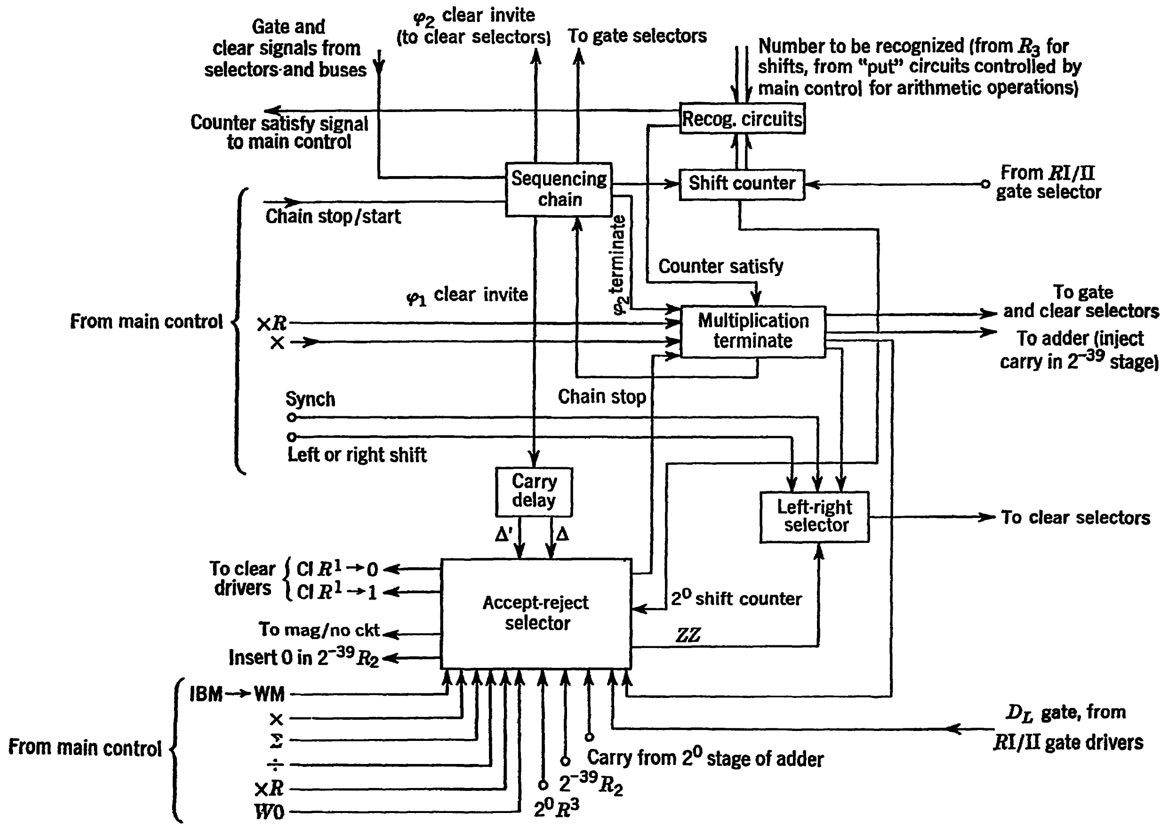


FIG. 13-7. Arithmetic control.

be set to "right" (0). In addition or subtraction, both the left- and right-shift signals are denied, and the L/R selector is governed entirely by the ZZ and $\widehat{ZZ}$ signals. These are generated as shown in the accept/reject-selector diagram; ZZ is affirmed if Σ is called for and if the 2^0 stage of the shift counter holds 0; otherwise ZZ is denied.

Having disposed of these preliminaries, suppose addition is called for. When T_{block} is turned on, the chain starts and generates first the φ_1 clear. The ZZ signal, being affirmed, passes through the mixer M_A and opens the gate for Δ; since $\widehat{ZZ}$ is denied (as are $\times$ and $\div$), no signal passes through the mixer M_R, and so the undelayed Δ' is blocked. Hence the Cl $R^1 \rightarrow 0$ is transmitted to the RI clear-driver drivers, and R^1 is prepared to receive the output of the digit resolver. The clear signal is fed back to the chain, which then completes phase 1 by generating a U_0 gate. Next, the φ_2 clear invite is generated and transmitted to the clear selector for RI, which, under the influence of the signal from the L/R selector, causes R_1 to be cleared to 1; then the chain produces a D_L gate. The chain cycle also transmits two pulses that cause the count in the shift counter to advance from 0 to 1. This causes ZZ to be denied, and so, in the next cycle of the chain, the gate is opened for Δ' and closed for Δ, and R^1 is cleared to 0's and thus prepared to receive the contents of R_1 by a U_1 gate, which is subsequently generated by the chain. In φ_2, the RI-clear selector is directed by the L/R selector, where $T_{L/R}$ has been switched to 0 by $\widehat{ZZ}$, to transmit the φ_2 clear invite as a Cl $R_1 \rightarrow 0$ command, which the chain follows with a D_R gate, thus putting the digit-resolver output in R_1 in the correct position. During this cycle the counter advances to 2, and the recognition signal is given, so that T_{block} is turned off and further action by the chain is inhibited.

Now consider division,[1] which is called for by the denial of $\Sigma/\div\times$ and the affirmation of $-L/+R$. The toggle T_+ in the accept/reject selector is locked in the 1 state at all times except when division is ordered; then it is released and permitted to respond to an input from the 2^0 stage of R_1 (sign of dividend/remainder); it is caused to assume the state opposite to that of 2^0R_1. Other circuits not shown compare the signs of the dividend and the divisor; if these agree, the negative complement gates are enabled throughout the division process; if they disagree, the positive gates are enabled instead. It is, of course, assumed that the dividend has already been placed in R_1 by first clearing R_1 to 0's and then causing the dividend to be added to its contents.

The sequencing chain is set in operation with the turning on of T_{block}, and the φ_1 clear invite is generated. The decision must now be made

[1] Cf. Sec. 1-5. The process given below is a modification of the one described in Sec. 1-5 and has the merit of automatically generating the correct sign bit of the quotient, so that no final correction is needed.

whether to accept into R^1 or to reject the first tentative partial remainder, which is formed by the adder and the digit resolver. This decision is made on the basis of a comparison of the carry from the 2^0 stage of the adder with the sign bit (2^0R_1) of the dividend, and, obviously, cannot be safely made until the carry-propagation time has elapsed. For this reason, the gate by which the carry enters the accept/reject selector is enabled by Δ, whereupon the carry is compared with the contents of T_+, the complement of the sign of the dividend. If these two do not agree [i.e., if the carry and (2^0R_1) are the same], then an affirmative signal is sent to M_R, and a Cl $R^1 \rightarrow 0$ is executed, whereas, if they agree, R^1 is cleared instead to 1's (as a matter of fact, for $|x| < |y| < 1$, which must be the case, the first step must always reject). The chain next produces the correct gate, either U_1 if rejection is indicated or U_0 for acceptance. In φ_2, in either case, the sequence must be Cl $R_1 \rightarrow 1$, D_L, as dictated by the clear-and-gate selector under the influence of the L/R selector. It is also necessary to enter the 2^0 bit of the quotient in $2^{-39}R_2$. During division, the clear-gate sequence in RII is Cl $R^2 \rightarrow 0$, U_1, Cl $R_2 \rightarrow 1$, D_L, but no D_L gate reads into $2^{-39}R_2$; instead, the accept/reject selector gates in a 0 if the adder carry is the same as the sign (2^0R^3) of the divisor but does nothing in the contrary case, leaving the 1 undisturbed. As the insertion takes place as soon as Δ is affirmed, it is clear that, at the end of the first cycle of the chain, the 2^0 bit of the quotient is held in $2^{-38}R_2$ and that $2^{-39}R_2$ holds a 1 and is ready to receive the next bit from the accept/reject selector. Before the process begins, R_2 is cleared to 1's and so prepared to receive the first bit of the quotient. Thus the division rules are quite easily put into force. The chain goes on cycle after cycle, with the gates, clears, and insertions of 0's in $2^{-39}R_2$ generated in the fashion just outlined. This goes on until 39 cycles have been completed; during the 39th cycle, the shift counter advances to 39, and a satisfy signal is generated which, at the end of φ_2 of the 39th cycle, is permitted to stop the chain (see drawing of multiplication-terminate circuit). At this point the sign and the first 38 bits of the quotient are in their correct positions in R_2, and $2^{-39}R_2$ is 1; this is the actual quotient "rounded" by the arbitrary insertion of a 1 in the lowest-order position.

Multiplication is the most complicated of the arithmetic operations, because it is necessary to perform "cleanup" operations to compensate for negative factors and, therefore, to introduce the multiplication-terminate circuit. First of all, the multiplier must be inserted into R_2, and then the multiplicand into R^3. The sign of R^3 is inserted into the extra 2^1RI toggle. The important commands from the main control to the arithmetic control are $\times$ and R. The command $\times$ does a number of things: notably, it enables the gates out of $2^{-39}R_2$ to the mixers M_A and M_R and unlocks the T_H toggle in the multiplication-terminate circuit.

The command R, among other things, sets up the L/R selector in such a way as to demand right shifts in φ_2 of each chain cycle.

Observe that the accept/reject selector now causes a partial product to be accepted into R^1 whenever $2^{-39}R_2 = 1$ or to be rejected if $2^{-39}R_2 = 0$; then the contents of R^1 are shifted down right into R_1. At the same time the bit in the special toggle $2^1R\mathrm{I}$ is shifted down right into 2^0R_1; this is how it is added to the number in the adder or just to the number in R_1 during the next cycle. The D_R gate that transfers the contents of toggles $2^1R\mathrm{I}$ and 2^0R^1 to $2^{-38}R^1$ into 2^0R_1 to $2^{-39}R_1$ also, by the end-around circuits, transfers the contents of $2^{-39}R^1$ into 2^0R_2 and $R\mathrm{II}$ shifts right with $R\mathrm{I}$.

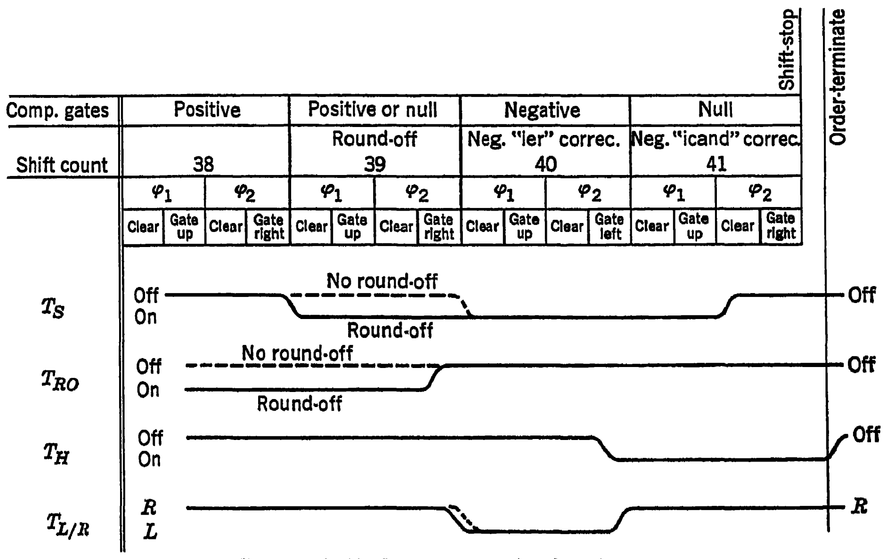


FIG. 13-8. End steps of multiplication.

This process continues for 38 steps (RO, round-off) or 39 steps (NRO, no round-off), 38 and 39, respectively, being the numbers to be recognized presented to the recognition circuit by the "put" circuits in the two cases. Until the counter-satisfy signal is generated by the recognition circuit, the states of the toggles in the multiplication-terminate circuit are (*a*) $T_S = 0$, (*b*) $T_H = 0$, (*c*) $T_{RO} = 0$ (NRO) or $T_{RO} = 1$ (RO). Consider first the NRO case and refer to Fig. 13-8, "End steps of multiplication." At the conclusion of the 39th chain cycle, T_S is turned on. The immediate consequences are: (1) $T_{L/R}$ is switched to 1, calling for a left shift; (2) the positive complement gates are disabled, and the negative gates are enabled; (3) the $R\mathrm{II}$-gate selector is inhibited, so that the contents of $R\mathrm{II}$, i.e., the lowest-order 39 bits of the product, may be left undisturbed during the ensuing cycle. That there must in fact be a 40th cycle is ensured since $T_H = 0$ inhibits the issuance of a chain-stop command. During the 40th chain cycle, the negative -ier correction is made by subtracting the contents of R^3 from the contents of R_1 if the sign bit of the multiplier, now in $2^{-39}R_2$, is 1. Otherwise, there is a shift up $R_1 \rightarrow R^1$, and in both cases a shift down left $R^1 \rightarrow R_1$, and the D_L-gate command, being fed to the multiplication-terminate circuit, turns T_H on, which switches $T_{L/R}$ to 0 and calls for a right shift in the next (41st) cycle.

Now examine the number residing in R_1. If both factors are positive, this is merely the product shifted one place left, its sign, 0, being in 2^1RI and an extra 1 appearing in $2^{-39}R_1$ due to the Cl $R_1 \rightarrow 1$ that preceded the D_L gate. If x (-ier) < 0 but y (-icand) > 0, the 40th step added $2 - y$ to $y + xy$ giving $2 + xy$, the correct machine representation of the negative product; hence, this number appears in R_1, shifted one place left and with an extra 1 in the 2^{-39} position. If $x > 0$, $y < 0$, then during the 40th cycle all that took place was a left shift; the sign of y, however, was injected into 2^0R_1 from 2^1RI during the down-right shift from $R^1 \rightarrow R_1$ at the end of the 39th chain cycle, and so the number shifted during the 40th is actually $2 + xy - 2^{-39}$, the last place being filled by a 1 during the left shift. Hence 2^1RI and R_1 together hold $(2 + xy$, shifted left$) - 2^{-38} + 2^{-39}$. Finally, if both $x < 0$ and $y < 0$, at the end of φ_1 of the 40th cycle R^1 again holds $2 + xy - 2^{-39}$, or just $xy - 2^{-39}$ because $xy > 0$. This is shifted down left, so that R_1 holds $(xy$, shifted left$) - 2^{-38} + 2^{-39}$. Finally, the D_L gate used in performing the left shift makes a transfer from 2^0R^3 into $2^{-39}R_2$; since it is specially cleared to 1 in preparation for this, $2^{-39}R_2$ ends up by holding the sign bit of the -icand, since the D_L gate can transfer only a 0. At this time T_H is switched to 1.

The final cleanup is accomplished during the 41st step, which follows the 40th because the condition for the generation of the chain-stop command ($T_S = 0$, $T_H = 1$) has not yet been satisfied. During this cycle, the complement gates are nulled and a carry is injected into the 2^{-39} stage of the adder; if $2^{-39}R_2$ is 1, acceptance takes place, followed by a D_R gate $R^1 \rightarrow R_1$. If $2^{-39}R_2$ is 0, the sum is rejected, and a right shift takes place in RI. Hence, in all cases, the correct machine representation of the product is obtained: the sign and highest-order 39 bits are in R_1, and the lowest-order 39 bits are in R_2. A sufficiently acute reader will deduce that a 0 multiplier and a negative multiplicand would give (uncorrected) the result $1.00 \cdots$. However, a simple circuit, which need not be presented here, compensates for this. The φ_2 clear/invite signal now, since $T_H = 1$, sets T_S at 0 and causes the "chain-stop" command to be sent to the clear-gate chain.

Now consider the operation with round-off. From Fig. 13-5, it will be seen that $T_S = 0$, $T_H = 0$, $T_{RO} = 1$ until the end of the 38th chain cycle, when the number 38 presented to the recognition circuit by the "put" circuits agrees with the number in the shift counter, and the counter-satisfy signal is generated, switching T_S to 1. The chain-stop command cannot be issued until $T_S = 0$, $T_H = 1$; so the chain continues to operate. During the 39th cycle, the complement gates are nulled, a carry is injected into the 2^{-39} stage of the adder ($T_S = 1$, $T_{RO} = 1$), and the result is accepted into R^1 and shifted down right into R_1. The D_R gate of this cycle turns T_{RO} off, and the remaining cleanup operations are accomplished as in the *NRO* case.

Finally, in the right-shift and left-shift operations, the number of places which the contents of R_1 are to be shifted is specified in the first 10 bits (i.e., the address part) of the instruction. The accept/reject selector always enforces a reject by the combination of $\hat{\Sigma}$ (that is, $\div$ or $\times$) and $\widehat{WO}$ ("not work order"; i.e., no reference is to be made to the Williams memory). The $-L/+R$ signal from main control also sets up $T_{L/R}$ to cause the shift to be in the desired direction until the counter-satisfy signal and the φ_2 termination produce a counter-stop command; as shown in the multiplication-terminate drawing, Fig. 13-5, this does not depend upon the state of T_H. There is a slight subtlety in the case of the right shift, which requires $T_S = 1$, $T_{RO} = 0$. Normally $T_{RO} = 0$, so there is no trouble, but the possibility of a "rounded" right shift is left open and such a shift might possibly be included if desired.

13-7. A Serial Arithmetic Unit (MEG). As an example of a serial arithmetic unit, that of the Manchester megacycle computer (MEG) has been chosen because it is of the floating-point type.[1] In MEG, numbers are represented as $m \times 2^n$, where m is a fraction and n is a positive or negative integer. The fraction m is always carried in "standard form": $\frac{1}{2} > m \geq \frac{1}{4}$, $-\frac{1}{4} > m \geq -\frac{1}{2}$ occupying 29 binary places plus a sign in the highest-order place, the whole being treated as a 30-bit fraction; and n consists of a sign and nine integral places such that $-512 \leq n < 512$. Each number is removed from the memory 10 bits at a time. The 10 bits of the exponent n are the first to emerge; then, after a delay of 10 μsec, come 10 bits of m; after another 10-μsec delay, 10 more bits of m, and so on. It is evident that the chief difficulties in a floating-point arithmetic unit must be connected with addition, for it is necessary to shift one addend so that its exponent agrees with that of the other before the addition can be performed. Multiplication, on the other hand, is quite simple.

The arithmetic unit has as its heart the floating-point accumulator; in addition there are the various registers needed to hold the operands, and the multiplier. Control is effected by the opening and closing of gates to regulate the flow of the pulse trains carrying the information.

[1] D. B. G. Edwards, The Design and Construction of an Experimental High Speed Digital Computer, Ph.D. thesis, University of Manchester, 1954.

To see what must be accomplished by the circuits, suppose that one addend $m_A 2^{n_A}$ has already been inserted in the accumulator and that a second $m_s 2^{n_s}$ is to be added to it. First of all, n_s emerges from the memory, and the difference $\delta = n_A - n_s$ is formed, which immediately reveals which of the addends is the larger. This is first tested to see if it is greater than or equal to 31 in magnitude; if it is, the larger of the numbers is left in the accumulator and the process is terminated. If it is found that the magnitude of δ is less than 31, then the larger exponent is recorded in the exponent part of the accumulator and variable-delay circuits (of which more will be said below) are set up to cause the appropriate shift. After this, addition can be carried out, and, finally, it may be necessary to do some more shifting to put the sum (or difference) in standard form. In order to prevent overflow, each m is automatically "extended" two places to the left upon entering the computing circuits; in the final reduction to standard form, if an overflow has occurred it is corrected by shifts to the right, and the "extended" places are dropped before the sum is transmitted from the accumulator to the memory. The extensions consist of two copies of the highest-order (that is, the sign) bit and are considered to be in positions 2^1, 2^0; the sign bit is in 2^{-1} and the numerical information in positions 2^{-2} to 2^{-30}. As an illustration, consider the sum of $\frac{1}{4} \times 2^n + \frac{1}{4} \times 2^n$. As each m is represented by .010 · · · , the "extended" versions are each 00.010. The sum of these is 00.10; if this were left alone and if the extended places were merely dropped, this would be recorded as a negative number. However, it is noted that in the extended numbers the last change from 0 to 1 or 1 to 0 must occur between the 29th and 30th bit (counted from the lowest-order position) and the fact that in the sum this change occurs between the 30th and 31st indicates an overflow of one place. Hence the sum is shifted one place to the right and the exponent n is increased by unity; so the sum as finally transmitted from the accumulator to the memory is $.01 \times 2^{n+1}$ or $+\frac{1}{4} \times 2^{n+1}$.

In the multiplier, the normal procedure is to obtain the multiplier and the multiplicand from the memory, multiply them together, and add their product to the number already contained in the accumulator. The multiplier forms products of positive m's; negative numbers are complemented and their signs recorded, so that the correct sign of the product is remembered and the product complemented if it is negative. The exponents are merely added. It will be appreciated that it may be necessary to standardize the product; that is,

$$(.01 \times 2^0) \times (.01 \times 2^0) = .0001 \times 2^0 = .01 \times 2^{-2}$$

13-8. Structure of the MEG Arithmetic Unit. The arithmetic unit contains (*a*) a register in which each number is assembled from the 10-bit blocks that emerge from the memory—during multiplication this register holds one factor and, after multiplication, the product; (*b*) a register for holding the second factor in a multiplication; (*c*) the accumulator proper, which consists of a register, an adder, a complementer, an extender, a variable-delay circuit, and a counter (the "*z* counter"); (*d*) the multiplier; and (*e*) local control elements, such as toggles set up at certain stages of the computation and used to control later stages, and gates which control the flow of information and which are enabled either by pulses derived locally or by commands from the central control. Each register is in fact two, one for the 30 bits of *m* and one for the 10 bits of *n*; they are of the circulating type described in Chap. 10, consisting of delay line, amplifier, and the necessary read-in, read-out, and erase gates.

The input register of the arithmetic unit, at least the part of it that holds the 30 bits of *m*, is shown in Fig. 13-9. It has been stated in the last section that numbers emerge from the memory in 10-bit blocks, each

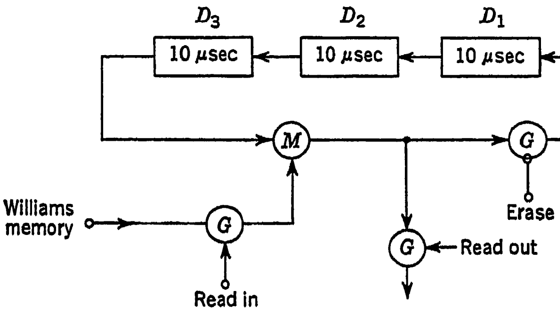


FIG. 13-9. Input from memory to arithmetic unit (MEG), the "*R* register."

10 μsec in duration. The 10-μsec blocks are separated by 10-μsec intervals, during which the contents of the memory are regenerated. If *m* is considered to consist of three blocks of 10 bits each, m_1, m_2, m_3, where m_1 consists of the highest-order bits, then the sequence in which a number emerges from the memory is n_1, m_2, m_1, m_3. This may seem eccentric until Fig. 13-9 is considered (the amplifier has been omitted as irrelevant to the logic). When the read-in and erase gates are enabled, m_2 enters and, by the time its entry period of 10 μsec is complete, it occupies delay line D_1. At the end of the next 10-μsec period, during which no more information enters, m_2 is occupying D_2. Now m_1 enters and, at the end of its 10-μsec entry period, it occupies D_1 while m_2 occupies D_3. Now another 10-μsec period without entry of information ensues; at the end of this m_1 is in D_2 and m_2 in D_1. By the time that m_3 has entered, m_1 is in D_3, m_2 in D_2, and m_3 in D_1—all, therefore, in the correct order. The sign bit (the highest-order bit of m_1) is sensed in the memory and stored in a toggle and is, therefore, available to cause complementation later on when the number is transmitted to the accumulator.

The output register, from which information flows back from the arithmetic unit to the memory, is actually the register in the accumulator; the output is so taken that m_2, m_1, m_3 emerge in their proper order after n has been removed.

The essentials of the structure of the accumulator are shown in Fig. 13-10. A number to be added to the contents of the accumulator register enters on bus A. The elaborate system of gates, which determines whether the input or the number resident in the accumulator register is to be transmitted to the adder by way of the variable-delay circuit, is necessary for floating-point operation. The accumulator functions differently during the processing of the exponents and during the processing of the 30-bit numbers m. For this reason, a section will be devoted to each of these modes of operation, and no more will be said here. The unusual features of the circuit are the variable delay and its control, and the z counter. The variable delay consists of a sequence of delay units incorporating delay lines of lengths 1, 2, 4, 8, and 16 μsec, so that, by connecting or bypassing the individual lines, any delay from 0 to 31 μsec can be introduced in the direction of transmission. Each unit is actually quite simple; it provides a simple alternative of transmitting to the output either via a piece of delay line or directly under the control of a toggle in the variable-delay control. These toggles are set up by opening gates, driven by the points between the 1-μsec delay lines directly below them, when the number specifying the length of the delay is correctly positioned in them; the "clamp" signal ensures that all delays are bypassed when this input is appropriately stimulated. The number used to set up the variable-delay circuit is $31 - |\delta|$, where δ is the difference between the exponent of the number in the accumulator register and the exponent of the number being added to it.

The z counter's purpose is to keep track of the last change from 0 to 1 or 1 to 0 in the adder output during the number arithmetic. The idea is quite simple. A five-stage counter is set to start at 31 and count down to 0 in synchronism with the adder output. The change-overs are detected, and each is caused to open a set of gates from the counter stages to a set of memory elements, replacing whatever information was held there before by the current counter contents. Hence, when the counter has reached 0, the five-stage register has as its contents the position of the last change-over.

The delay circuit and the z counter can be regarded as local control elements. There are also several toggles which are set up in the course of the computation and used to enable or disable gates regulating its subsequent course. Thus, one toggle remembers the sign of δ, another whether or not $|\delta| > 31$, and another whether or not the magnitude of the number in the accumulator register is 0. Various gates are controlled both by these toggles and by signals from the main control. For example, the

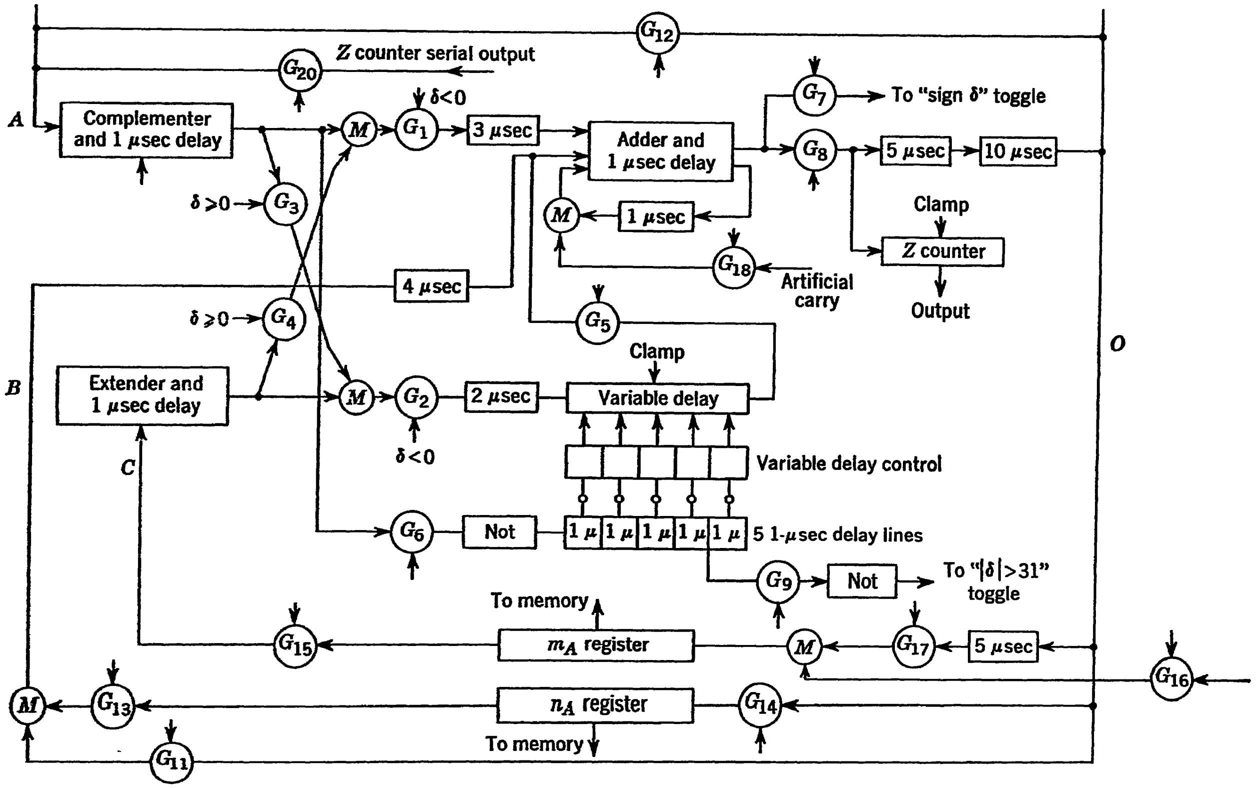


FIG. 13-10. Simplified block diagram of MEG accumulator.

"sign δ" toggle is reset or cleared before the operations begin and is switched only if the sign of δ is positive, so that the cleared condition can be thought of as the output for $\delta < 0$. This toggle controls the very important gates G_1 to G_4; G_1 and G_2 are enabled when sign δ is in its cleared condition, and G_3 and G_4 when it has been switched. It controls the complements to the extent that, when $\delta < 0$, it is fed back to bus A and complemented before being allowed to pass through G_6 into the variable-delay control. The "test $|\delta| > 31$" toggle controls gate G_5 leading to one adder input. This gate remains open as long as the toggle is in its reset, or cleared, condition, but if $|\delta| > 31$, the toggle switches and disables G_5.

Now that the general description of the accumulator has been carried this far, it must next be shown how it functions in performing an addition; this will be done in the next section. Discussion of the multiplier will also be deferred; note only that it consists essentially of a loop containing delay lines and a sequence of adders, into which one factor is gated repeatedly under the control of the successively examined bits of the other factor.

13-9. Exponent Arithmetic. Suppose that a number $m_A 2^{n_A}$ is already contained in the accumulator register and that a second number $m_s 2^{n_s}$ is to be added to the contents of the accumulator. It is first necessary to compare n_s and n_A; actually this is done while m_s is still in the process of emerging from the memory and of being assembled in register R. First the difference $\delta = n_A - n_s$ is formed; since the sign δ toggle is in its cleared (negative) condition, the complementer is stimulated and converts n_s into $-n_s$. Gates G_1 and G_2 are enabled, G_6 is disabled, and the variable-delay control holds all 0's; so the adder output is exactly $n_A - n_s$.

The gate G_7 is enabled as the sign bit of $n_A - n_s$ leaves the adder; if the sign is negative, nothing happens; if it is positive, the sign δ toggle is switched. When the difference reaches bus 0, it finds G_{11} and G_{14} closed but G_{12} open, and so is fed back to the input to the complementer.

Suppose first that $\delta \geq 0$, so that the complementer is disabled and the circuit acts as a simple 1-μsec delay. The gate G_1 is disabled, but G_3 and G_6 are enabled. The clamp signal to the variable-delay line keeps this element short-circuited; so δ is lost. However, δ also passes through G_6, through the "not" circuit (which forms essentially a 1's complement), and then into the five 1-μsec delay lines. Just at the instant when the lowest-order bit is emerging from the delay line, the five gates shown above it are opened, and the five lowest-order bits of $31 - \delta$ are recorded in five toggles, thus setting the delay length once the clamp is removed. Later, when the five highest-order bits begin to emerge from the next-to-the-last 1-μsec delay line, G_9 is opened to pass them; they are then passed through a "not" circuit; and, if any of the resulting bits is 1 (indicating that $\delta > 31$), the $|\delta| > 31$ toggle is switched, disabling G_5. The next event is to open G_{13} and let n_A pass into the adder, where it is added to 0 except for the injection of an artificial carry to the lowest-order position by way of G_{18}. The n_A register is erased, and the adder output $n_A + 1$ is put into it by way of G_{14}.

Now consider the case $\delta < 0$. As before, δ is fed back to bus A. The complementer is enabled because the sign δ toggle has not been switched, so $-\delta = |\delta|$ emerges from the complementer and passes through G_1 and G_6. However, G_{13} is also open; so the adder (with the aid of a single artificial carry) forms $n_A + |\delta| + 1 = n_s + 1$, and this number is put into the n_A register by way of G_{14}, replacing n_A. The formation of "not $|\delta|$," the setting up of the variable-delay circuit, and the test to see if $|\delta| > 31$ are also carried out as in the first case.

To recapitulate, (*a*) if $\delta \geq 0$, the exponent arithmetic ends with G_1 and G_2 closed and G_3 and G_4 open, the variable-delay control set to correspond to $31 - \delta$, $n_A + 1$ in the accumulator exponent register, and the $|\delta| > 31$ toggle switched or not according to the magnitude of $|\delta|$; (*b*) if $\delta < 0$, the exponent arithmetic ends with G_1 and G_2 open and G_3 and G_4 closed, the variable-delay control set to $31 - |\delta|$, $n_s + 1$ in the accumulator exponent register, and the $|\delta| > 31$ toggle set or not according to the magnitude of $|\delta|$. The circuit is now correctly set up to shift m_A or m_s as may be appropriate and to form the sum of the correctly adjusted numbers.

13-10. Number Arithmetic. First suppose that $\delta \geq 0$. The trivial case is $\delta > 31$; it will be disposed of first. As G_5 is disabled, m_s is lost, and m_A goes through the adder (no artificial carries this time!) and is fed back to the accumulator number register. In order to understand the important cases, it is necessary to appreciate the timing of the operation. The addition takes place in four periods of 30 μsec each, which will be designated T_1 to T_4. Actually, it is in T_1 that the final 10 bits of m_s emerge from the memory, but it is still possible to transmit m_s from the R register to the accumulator if necessary; for the final 10 bits emerge from the memory during the last 10 μsec of the period and, hence, follow the other 20 bits into the accumulator in the natural order. The circuit already "knows" which of the two addends has the larger exponent; this information is used to gate the number with the smaller exponent into the accumulator first; if the exponents are identical ($\delta = 0$) the number from the R register enters first. Thus, in any case, the number with the higher exponent (or the number from the R register) enters in T_1, and the other in T_2 (it should be added at this point that the sign δ toggle is not permitted to control the complementer during the number arithmetic; its function is now determined by the instruction being executed). The number with the smaller exponent passes through the variable-delay line, which was unclamped at the termination of the exponent arithmetic.

It is clear that sending in the smaller number first is equivalent to shifting it to the right 30 places and that delaying it $31 - \delta + 2$ μsec is equivalent to shifting it to the left; all together, this amounts to a right shift of $\delta - 3$ places. On the other hand, the larger addend is delayed (i.e., shifted left) three places or, if we choose, right -3 places. Thus the relative shift of the two numbers is δ places to the right for the smaller; this, of course, is just what is required.

It is also clear that the lowest-order bit of the sum reaches bus 0 at $50 - \delta$ μsec after the beginning of T_1 and the highest-order bit 82 μsec after the beginning of T_1, where the extra 2 μsec is due to extension. The z counter is unclamped for 32 μsec at the beginning of T_2. When the 32 bits of the sum have been obtained, the variable-delay circuit is clamped, and the sign δ and $|\delta| > 31$ toggles are cleared. The sum itself passes down bus 0 and through G_{17}; the accumulator number register serves merely as a 30-μsec delay; and the sum passes through G_{15} and back through the 1-μsec delay to G_2 and then to the variable-delay circuit. The variable-delay circuit has meanwhile been set up by the contents of the z counter.

It will be recalled that the larger of the two addends was presented to the accumulator at the beginning of T_2. The pulse in the sum that corresponds to the lowest-order position of this addend must arrive at bus 0 20 μsec later, leave the accumulator register 55 μsec later, and be presented to the input of the variable-delay line 58 μsec later, or 2 μsec before the beginning of T_4. If the sum is actually in standard form, the z-counter final reading must have been 2 (that is, 1 more than the number of the digit position in which the last change from 0 to 1 or 1 to 0 was detected). In this case the bit under consideration reaches gate G_{18} at the beginning of T_4. This gate is opened in all cases at the beginning of T_4 and closed at its termination; for a number in standard form, this automatically discards the two extensions, which are of no use anyway, and also the δ lowest-order places; therefore, the correct 30-place sum is read into the accumulator number register.

There still remains the question of the exponent, for it will be recalled that the accumulator exponent register holds either $n_A + 1$ or $n_s + 1$. Just after the sum has left the adder and the control toggles have been cleared, the number in the z counter is put into serial form and fed to bus A; concurrently, the number in the accumulator exponent register is fed to bus B. As the sign δ toggle has been cleared, the number z is complemented, and, in the adder, an artificial carry is injected into the lowest-order position of the sum; so the result is, for example, $n_A + 2 - z$; in case $z = 2$, this is seen to be n_A, which is correct. This is fed back to the accumulator exponent register.

Now consider the case of an "understandardized" sum, which can occur when the addends are of opposite sign. Suppose, e.g., that the sum is 00.000110 · · · . Both the extensions here are 0's, for the number is positive. The z counter records 4, since the last digit position (coming from the right) whose contents differ from the place immediately to the right is the third, and the counter must read $3 + 1 = 4$, for it started from 31 instead of 30. Hence the variable delay is set to 4 μsec, and the four highest-order places of the sum are lost by the action of G_{18}; so the sum as delivered to the accumulator number register is .01 · · · . It is now necessary to decrease the exponent by 2 to compensate for the shift in the decimal point. But, as we have seen, the sum of $n_A + 1$ (or $n_s + 1$) and δ plus the artificial carry gives $n_A + 2 - z = n_A + 2 - 4 = n_A - 2$, which is correct. In the case of an understandardized sum, i.e., a sum in which overflow has occurred, $z < 2$. For example, suppose the sum is 01.101 · · · , so that $z = 0$. Then the two lowest-order (i.e., first-to-arrive) bits are lost because they reach G_{18} too soon, and the number .01101 · · · is put into the accumulator exponent register as the sum. The final exponent arithmetic now gives $n_A + 2 - z = n_A + 2$, as it should. Hence it is seen that the standardization of the sum is performed correctly in all cases.

13-11. MEG Multiplier. In multiplication a variety of procedures are possible. The simplest from the point of view of equipment requires that one factor be examined bit by bit, beginning with the one in the lowest-order position, and that the other factor be added to the shifted partial product or not according as the bit examined is 1 or 0. On the other hand, it would be possible to examine all the bits of one factor simultaneously and to use each one to control the entry of the other factor; in this scheme, there must be one adder per binary place, and there must be a 1-bit-period delay between adders in the chain. Multiplication would be very fast, performed essentially in twice the time taken by the pulse train representing one factor, but a great deal of equipment would be required.

The MEG system is a compromise between these two extremes, giving reasonably fast multiplication without demanding an exorbitant amount of equipment. One factor m_D is inspected 10 bits at a time, but, because of the gating scheme used, only five adders are required in the multiplier proper (plus an extra one used in preparing the inputs). The 10 bits are used in groups of 2. These can be only 00, 01, 10, 11, and call for additions to the partial product of nothing, the other factor m_R, $2m_R$, or $3m_R$, respectively. From m_R it is simple to derive $2m_R$; this is done by shifting left one place (i.e., delaying 1 μsec, which is the time between the beginnings of successive pulses) and by putting the result in a separate register. Now a simple adder forms $3m_R = m_R + 2m_R$ and puts this sum also in a register. By a gating arrangement these three numbers are made available every 60 μsec. For each pair of bits there are three gates which permit the entry of m_R, $2m_R$, or $3m_R$ into the loop; these are so arranged as to be enabled by the combinations 01, 10, and 11, respectively.

The block diagram of the loop is shown in Fig. 13-11. (Amplifiers are omitted as irrelevant to the logic.) Each adder introduces a delay of 1 μsec, and 1 μsec of delay is introduced between adders. The total delay around the loop is 50 μsec. Read-out can, of course, be taken at any point. The erase gate is disabled when it is desired to clear the loop.

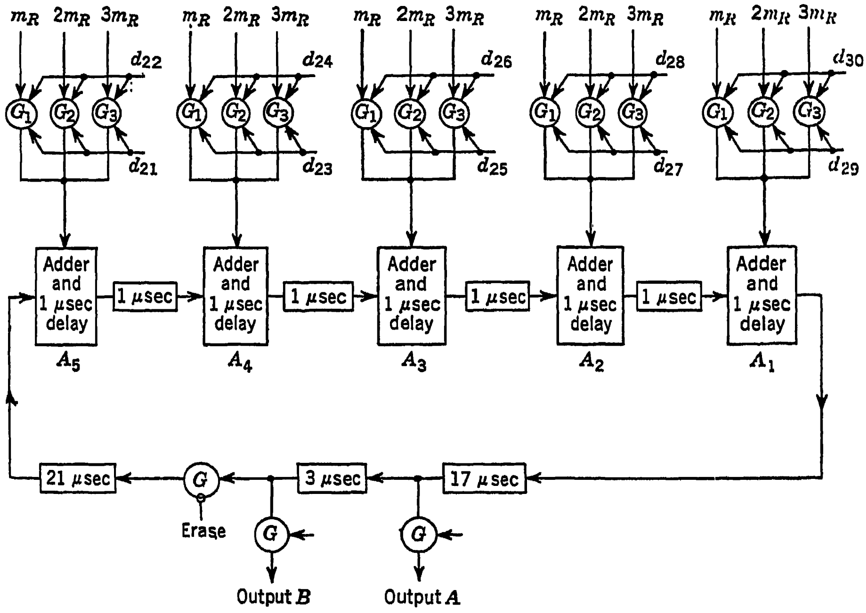

m_R $2m_R$ $3m_R$
d_{22} d_{21} d_{24} d_{23} d_{26} d_{25} d_{28} d_{27} d_{30} d_{29}
G_1 G_2 G_3
Adder and 1 μsec delay
A_5 A_4 A_3 A_2 A_1
1 μsec
21 μsec
G
3 μsec
17 μsec
Erase
Output B
Output A

FIG. 13-11. MEG multiplier loop.

The gates G_1, G_2, G_3 are enabled by the combinations 01, 10, 11, respectively. Before the numbers are presented to the multiplier, their signs are inspected. If either or both are negative, they are complemented, and the sign bits are used to switch a toggle, so that, if one sign is negative, the toggle is switched once and its output is read as a 1, but, if the toggle is switched twice and so returned to the original state, it is read as 0. After the multiplication has been carried out, the sign information remembered by the toggle is used to cause the complementing of the product if necessary.

Calling the contents of the D register $m_d \times 2^{n_d}$, suppose that the lowest-order 10 bits of m_d (written d_{21} to d_{30}) have been placed in a staticizer (e.g., a set of 10 toggles), so that voltages corresponding to them are applied to all the gates G_1 to G_3. Now suppose that m_R, $2m_R$, and $3m_R$ appear at the other inputs to the gates and that, in each case, none or only one of these is permitted to enter the adder to which the output of the gates is connected. The 2 lowest-order bits in channel 1 evidently get through the adder unaltered, but the next bit arrives at the adder at the same time as the lowest-order bit in channel 2. Tracing the inputs through all the adders, it is evident that a sum is formed on the following plan:

```
                <──────30 to 32 bits──────────────>
                XXX· · · · · · · · ·XXXXXXXXXX
            XXX· · · · · · · · ·XXXXXXXXXX
        XXX· · · · · · · · ·XXXXXXXXXX
    XXX· · · · · · · · ·XXXXXXXXXX
XXX· · · · · · · · ·XXXXXXXXXX
──────────────────────────────────────────────
XXX· · · · · · · · · · · · · · · · · · ·XXXXXXXXXX
<────────────────39 or 40 bits────────────────>
```

Thus the train of pulses leaving A_1 represents the product of m_R by the 10 lowest-order bits of m_d. If the inputs arrive at the adders at $t = 0$, then the lowest-order bit of the partial product begins to emerge from A_1 at $t = 1$ μsec, and the highest-order bit at $t = 39$ or $t = 40$ μsec. Therefore, the pulse train begins to enter A_5 at $t = 41$ μsec and begins to emerge from A_1 again at $t = 50$ μsec; at $t = 60$ μsec, when the additions under the control of d_{11} to d_{20} begin, the 9 lowest-order bits of the partial product have emerged from A_1, the 10th is beginning to emerge, and the 11th is beginning to enter and, hence, will have added to it the lowest-order bit coming down channel 1. Thus the product of m_R by the middle 10 bits of m_d is added to the existing partial sum correctly, and, clearly, the same will happen to the product by the highest-order 10 bits of m_d. As the full product occupies 60 (or 59) μsec but the delay around the loop is only 50 μsec, before the complete product has been formed the lowest-order 10 bits must be erased by disabling the erase gate for 10 μsec. If this information is to be retained, the gate in the B output is opened for the same time, and arrangements are made to receive and record the pulses elsewhere.

13-12. The Varieties of Multiplication. The normal procedure is to form a product and add it to the contents of the accumulator. When this is done, the exponents n_d and n_R are presented in succession to the A bus. First n_d is subtracted from n_A, then n_R from the difference, thus giving δ. Then m_Rm_d is added to or subtracted from m_A, exactly as it is in forming any sum or difference, and the result standardized. This process takes 360 μsec.

This multiplication can be either unrounded or else rounded in the sense of making the 30th place 1 in all cases; in any event, the 30 least significant places are dropped. Clearly, if the accumulator initially holds 0, it ends up holding the correct product.

Another form called "long multiplication" is also available in the instruction code. In this form, the product $m_R m_d$ is formed, the 30 least-significant bits are read into the D register, and the 30 most significant bits are read into the accumulator number register. The exponents are added, but no attempt is made to standardize the result. This process requires 300 μsec.

CHAPTER 14

THE MEMORY

14-1. Introduction. In most current digital computers, the memory is split into two or more sections. Of these, one (the smallest and most expensive) is the high-speed section, which is backed up by larger-capacity and cheaper devices, such as magnetic drums and magnetic-tape-handling units. The typical large machine uses magnetic-core arrays for the high-speed section, with a typical access time of 15 μsec, compared with a drum access time about 1,000 times as long and a tape access time of the order of tens of seconds. Various large-scale random-access devices with access times of one second or less have been invented to replace tape units; these will not be considered in the present chapter, which will be devoted to a description of the ORDVAC system. This system has been chosen because it is straightforward and easy to understand and because ORDVAC is a member of the family of machines modeled on the Institute for Advanced Study (IAS) machine, the arithmetic unit of which has been studied in Chap. 13.

However, before proceeding to the description of the magnetic-core memory of the ORDVAC, it will first be necessary to give a very brief account of the Williams memory of the original IAS computer in order to make the discussion of the main control and the arithmetic unit more readily understood. The memory consists of 40 cathode-ray tubes, each holding 1,024 bits, one tube being allocated to each binary place of the 40-bit words, together with the necessary control circuits. It has been shown in Chap. 12 that in a Williams memory it is necessary to regenerate the contents periodically, and so the memory is operated synchronously, the fundamental timing being provided by a multivibrator, since, the maintenance of a precise clock frequency not being important, so simple a clock suffices. The clock pulses drive a sequence of "pulsers" each of which emits, when stimulated, a pulse of appropriate length followed by a terminate pulse which triggers the next pulser in the chain. These pulses and those formed from them by a "pulse routine generator" which acts upon instructions from the memory's local control provide the bright-up, clear, and gate signals which cause the various types of memory cycle ("regenerate," "action," "fetch") to be executed. The term "regenerate cycle" is self-explanatory. An action cycle is one in which information is inserted in the memory or extracted from it and delivered to R^3. A fetch cycle is one in which the memory is required to deliver an instruction word to R_3. Each action cycle is followed by a regenerate cycle, a regenerate cycle by an action or a fetch cycle, and a fetch cycle by an action cycle. The main control signifies to the memory local control that the memory is to be used during its next following cycle by affirming a signal called "yes/no," that the cycle will not be a fetch by affirming "work order" ($WO/\widehat{WO}$), and that reading or writing is to take place by affirming or denying a signal called "read/write." These, in conjunction with pulses from the pulsers, set up toggles in the memory local control, named "yes," "synch," "action," and "store." These toggles control a number of gates which, for example, determine the source from which the address that is to be inserted in the address generator (i.e., the center rank of the dispatch counter) is to be obtained, and cause the appropriate gating and clearing operations needed for the transfer of information from the memory to R^3 or R_3 to be executed. The memory local control sends various signals back to the main control: (1) "synch," which is affirmed from the setting of the synch toggle until the appropriate address has been inserted in the address generator, (2) the signal used to open the gates from the discriminator (a toggle register in the memory which receives each word as it is read from the Williams tubes) into RIII, and (3) a signal signifying that the action toggle has been set. In Chap. 15 it will be shown how the main control generates its instructions to the memory and what uses it makes of the signals received back from it.

The ORDVAC was built originally with a 1,024-word Williams memory, which was later augmented by a 10,032-word drum. Later still, the Williams memory was replaced by a 4,096-word magnetic-core memory. Only in the latter are individual words addressable. On the drum, tracks only can be addressed. The unit of information in all transfer operations is one trackful, or 48 words of 40 bits each, usually called a "block."

A block diagram showing the paths of communication between the memories and the arithmetic unit is given in Fig. 14-1. The ORDVAC possesses the registers RI, RII, RIII, essentially the same in character and function as those of the IAS machine. RIII serves to receive words drawn from the magnetic-core memory; words representing numbers are delivered to R^3, and words representing instructions to R_3. From R^3 a transfer can be made to RII, or the contents of R^3 can be presented to the adder (these transfers are, of course, all parallel). Since the drum functions as a serial device, information must be presented to it in the form of a temporal train of pulses. This is accomplished by first placing each word to be transferred in RII and then shifting left 40 places. In the transfer from the drum to the core memory, the serially emerging bits of a word are inserted in the rightmost, or lowest-order, stage of R_1, and after each bit is inserted the contents of the register are shifted left one place, so that, after the reception of 40 bits alternated with 39 left shifts, the word is properly positioned in R_1 and is ready for parallel transmission to the magnetic-core memory.

The core memory cycle is very nearly 15 μsec, the information being in fact available about one-quarter of the way through the cycle. The

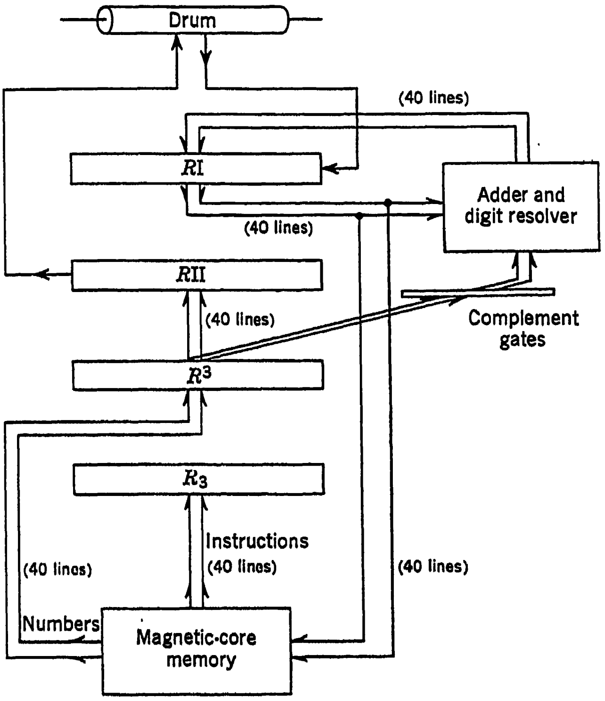


FIG. 14-1. ORDVAC memory: connections with the arithmetic unit.

drum rotates at 3,525 rpm, or once in 17 msec; so the average access time is 8.5 msec. The time needed to set up the relays that perform the track selection is about 35 msec, and three complete drum revolutions are required to read out or record one trackful of information; so the average time required to transfer 48 words from the drum to the magnetic-core memory is 94.5 msec. This is characteristic of transfers between high- and low-speed memories, at least in present-day computers. Several ways of overcoming the problem of transfers exist. One is to have only a high-speed memory; this is not at all impracticable, for several machines now on the market can be equipped with core memories of up to 32,000 words, and at least one design contemplates 100,000 words. Another possibility is to arrange things so that transfers can be made at the same time that the machine is computing. This scheme is very attractive, and will probably be incorporated in production-type machines in the very near future.

14-2. ORDVAC Magnetic-core Memory. The magnetic-core memory used in ORDVAC holds 4,096 ORDVAC words of 40 bits each.[1] The general form of the digit planes has already been described in Chap. 12. The complete memory actually has 42 of these, of which the extra two are spares in case of trouble.

In Fig. 14-2 is shown a block diagram, where, for convenience, only one digit plane is represented. Each digit plane is driven by a 32-position electronic switch for the 32 X lines and by a 128-way magnetic-matrix switch for the 128 Y lines. The X and Y switches get their information from the X and Y decoders; there is a single X-address decoder for the whole memory, but the Y address is split and decoded in two sections of 3 and 4 bits. The 128-way magnetic switch is also driven by a Y_N-switch driver, which is activated by the gates. These in turn receive as inputs the information coming from the ORDVAC (R_1) and from the partial-substitution unit, which determines, on instructions received from the ORDVAC, whether reading or writing is to be accomplished, either in all the planes or in blocks of 8, 12, 8, 12 planes, 8 bits being the part of each instruction used to designate the operation, the remaining 12 bits specifying the address.

Each digit-plane output drives a reading amplifier, the output of this amplifier is gated into a toggle (the complete set of such toggles forms the digit register), and from the toggle an output is taken to the ORDVAC (R^3 or R_3).

The timing of the memory cycle is very simply determined by seven monostable circuits called Univibrators (one-shot multivibrators). Each of these puts out a "main" pulse when stimulated (these are of different lengths in the several circuits), followed by an "end" pulse of 1 μsec duration; these are designated by P_1, P_{1E} for U_1; P_2, P_{2E} for U_2; and so on. As the drawing shows, P_{1E} stimulates U_2; P_{2E} stimulates U_3; P_3, U_4; P_{3E}, U_5; and finally P_{5E}, U_6. U_7 and U_1 are stimulated by the memory-request pulse. The X_P, X_N, and cycle mixers are merely toggles set by one input and reset by the other—X_P, for example, by P_1 and P_{3E}.

The boxes labeled "current control," "level set," and "stabilizer" perform electrical but not logical functions, and so will not be discussed here.

[1] This was built for the Ballistic Research Laboratories (Aberdeen Proving Ground) by Telemeter Magnetics, Inc., under an Army Ordnance contract. The author gratefully acknowledges the company's kind permission to describe the equipment.

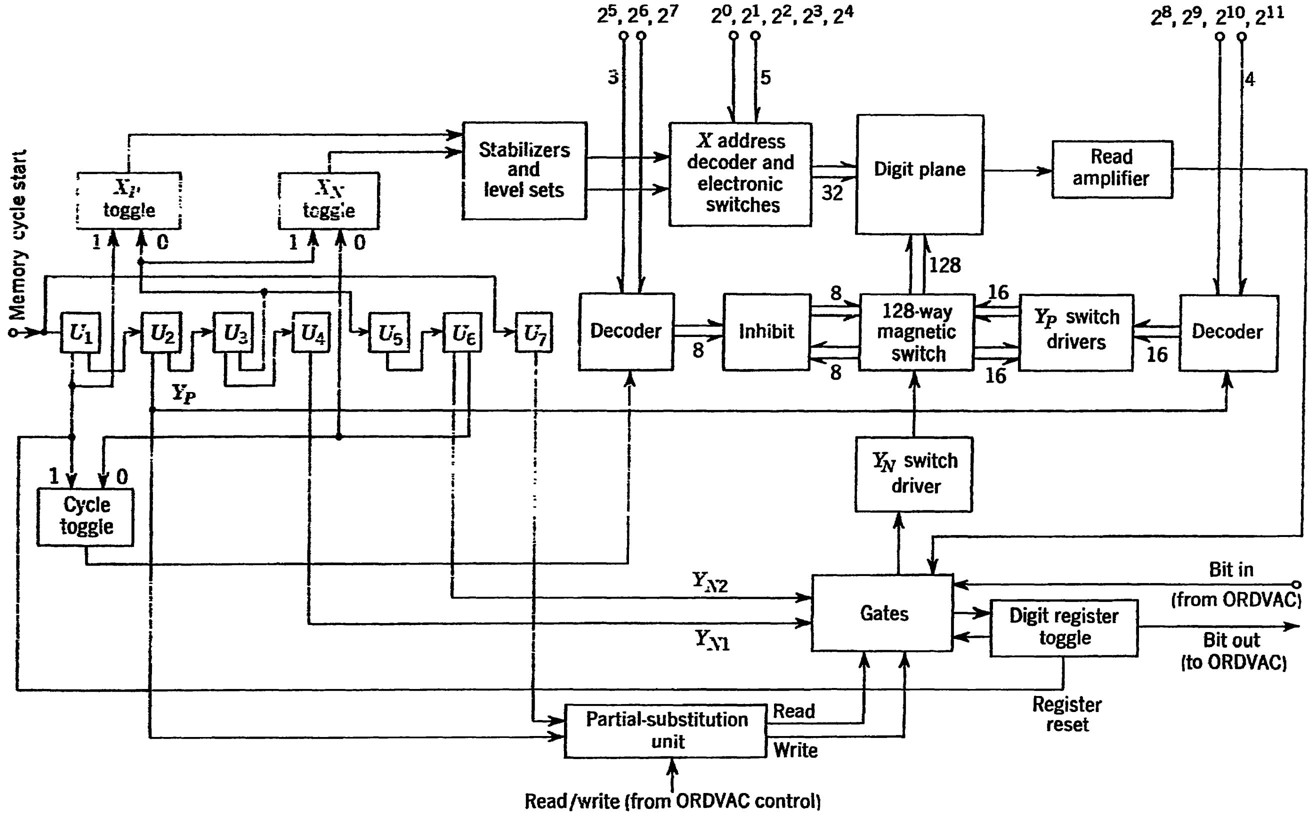


FIG. 14-2. Block diagram of ORDVAC magnetic-core memory (Mnemotron).

14-3. The Digit Planes. There are 42 of these, each an array of 4,096 ferrite cores wired on 32 X and 128 Y lines. Four spare Y lines are provided in each plane in case of damage. Each array is double-sided with 32×64 cores on each side; one may consider it to have been formed by folding a 32×128 array in two. A brass plate is mounted between the two sides to give a common ground return for the Y lines and also to improve the compactness and rigidity of the array. The individual ferrite cores are 0.1 and 0.07 in. in outer and inner diameter, respectively; they are switched from positive to negative saturation in 1 μsec.

Each X line passes through all the cores of its row three times; first it passes through all the cores of its row on one side of the matrix, then it passes over to the other side and returns through all the cores of the row, then it comes back to the first side and passes again through all the cores of the row, and so on. After the third pass, it goes on to be connected in series with the corresponding X line of the next matrix. Finally, the X line is terminated in its characteristic impedance as a countermeasure against reflections.

In each matrix, each Y line passes once through all 32 cores of its column and is grounded to the brass shield.

The read line of each matrix must link each core once and must be carefully arranged to minimize the total disturbed signal. As will be seen in relation to the memory cycle, the X selection is performed first, and no output signal is gated out of it until the full selection has been completed by energizing the proper Y line. However, if the transient due to X selection were permitted to be large, it might block the read amplifier, with disastrous results. The disturbed signal caused by turning on the Y-selection pulse is never very large, coming as it does from only 32 cores; this was one reason for the choice of the rectangular array.

The actual arrangement of the read line is as follows. On one side of each matrix it passes through the cores of row 0, e.g., to the left, and then comes back to the right through the cores of row 1, skips to row 3, where it goes left, and returns to the right through row 4, and so on; finally, from row 29 it passes to row 31 and goes left, then right in row 30, then skips to row 27, and so on. The read line of the other side of the matrix is opposite to this, and the two are so connected to the output-pulse transformers that only the difference between the signals appears in the output.

14-4. The Magnetic Switches. On the frame that holds each array of memory cores is mounted also a 128-way magnetic switch, consisting of ferrite cores arranged in a 16×8 array. These cores are 0.260 and 0.375 in. in inner and outer diameter, respectively, and switch in 5 μsec. Each core carries four windings: (1) inhibit, (2) Y_P, (3) Y_N, (4) output, of eight, eight, ten, and four turns, respectively. Inputs are from the 8 inhibit and the 16 Y_P lines, by means of which selection of a core is accomplished. The Y_N windings are connected in series, and a separate output is taken from each output winding. Figure 14-3 shows a simple 2 × 4 version of this switch.

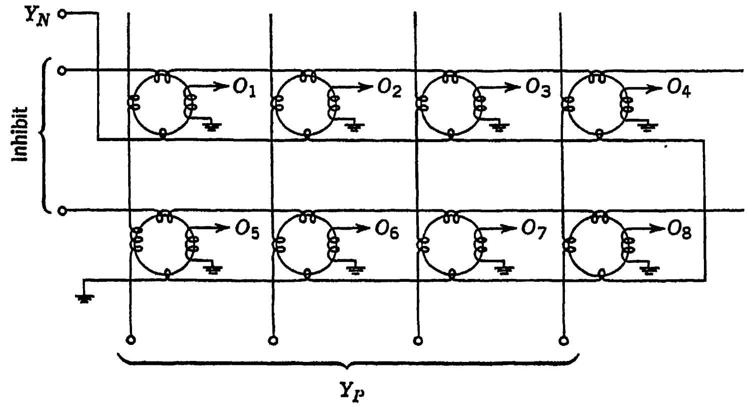


FIG. 14-3. Magnetic switch used in Mnemotron.

Under static conditions, the inhibit windings are driven by a current of 500 ma, and the Y_P inputs are not stimulated. This suffices to hold all cores in a saturated condition, as, e.g., at point P in Fig. 14-4. When switching is to occur, one core is selected by first deenergizing the row in which it is located and then driving the Y_P line in its column with a current pulse of 500 ma peak value. Thus the selected core during the pulse is driven nearly to positive saturation, to some such point as R in Fig. 14-4. The other cores in the column will be driven up only as far as perhaps Q. The selected core, in the process of rising to R, must give an output signal, because of the large change of flux, whereas the unselected cores of its column, in which the flux change is negligible, give a negligible output. At the end of the Y_P pulse, the selected core returns to some such point as S, all others in its column return to P, and all others in its row remain at Q. Later on, the Y_N input is stimulated by a pulse of 250 ma peak value. The cores of the unselected lines are driven further toward negative saturation, with negligible effect upon the output windings since the change in flux linking them is very small. The unselected cores of the selected line

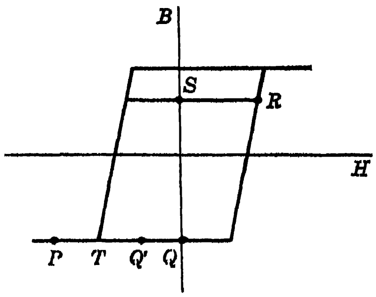


FIG. 14-4. Magnetic-switch operation.

are driven from Q down toward P, again with no effect of importance upon the output windings. The selected core, however, is driven from S down to T or perhaps further; hence, as the change in flux linking the output winding is large, a considerable output signal is developed. Both the Y_P and Y_N output signals are, in fact, 0.5 amp.

The amplitudes of the driving currents are carefully controlled. Troubles due to coupling and loading are avoided as far as possible by (in effect) totally disconnecting the inhibit, Y_P, and Y_N lines when they are deenergized.

14-5. The Y-address Decoding Circuits. The address part of each instruction consists of 12 bits, of which the first 5 (2^0 to 2^4) specify one of the 32 rows X of the digit planes, and the last 7 (2^5 to 2^{11}) refer to the columns Y. As noted above, these columns are divided into groups of 3 (2^5 to 2^7) and 4 (2^8 to 2^{11}) and decoded to give the 8 and 16 inputs to the magnetic switches. Thus the decoding is very simple. It is accomplished in two stages: (1) in the Y-address partial decoder (shown in Fig. 14-5), and (2) in the Y_P switch drivers and the inhibit circuits.

The partial decoder consists simply of inverters and gates. The received voltages are +10 for 1 and −35 volts for 0, and the inverters interchange these. The gates are very simple, consisting of three triode cathode followers with the cathodes connected together, the cathode voltage being the output, so that a coincidence of negative-going signals is detected. The outputs of the partial decoder are fed to the 8 Y inhibitors and to the 16 Y_P switch drivers. Each of these consists of a gate followed by an amplifier to drive a line of a magnetic switch. Thus, in the Y_P driver, one of the outputs 0 to 7 of Fig. 14-5, $\pm 2^{11}$, and the Y_P pulse are tested for coincidence, and the line driven if and only if all are negative. In the inhibitors, $\pm 2^5$, $\pm 2^6$, $\pm 2^7$ are tested for coincidence in just the same way as $\pm 2^8$, $\pm 2^9$, $\pm 2^{10}$ in Fig. 14-5, and the triple that coincides cuts off the amplifier, driving one of the "inhibit" lines in the magnetic switch. Every "inhibit" and Y_P line is provided with circuits at its far end which completely disconnect it from all other circuits when it is not being driven.

14-6. The X Partial Decoder and Switches. The selection and the driving of the X lines of the memory arrays are carried out entirely by vacuum-tube circuits. The signals representing 2^0 to 2^4 are first translated in voltage level and inverted, so that, for each 2^i, there are available on separate leads two signals $+2^i$ and -2^i. In the first, −213 and −140 volts represent 0 and 1; for the second the values are interchanged. These signals are then fed to a number of gates that detect coincidence of negative voltages, just as in the Y-address partial decoder. There are 12 of these gates, eight with three inputs ($\pm 2^2$, $\pm 2^3$, $\pm 2^4$) and four with two inputs ($\pm 2^0$, $\pm 2^1$).

The 12 outputs are then taken to the X_P/X_N switches, of which there are 32; a representative switch is shown in Fig. 14-6. In the switch one voltage of the group of eight and one of the group of four are examined for coincidence, and the gate output is fed to two gates whose other inputs are the negative-going X_P and X_N request pulses, which follow one

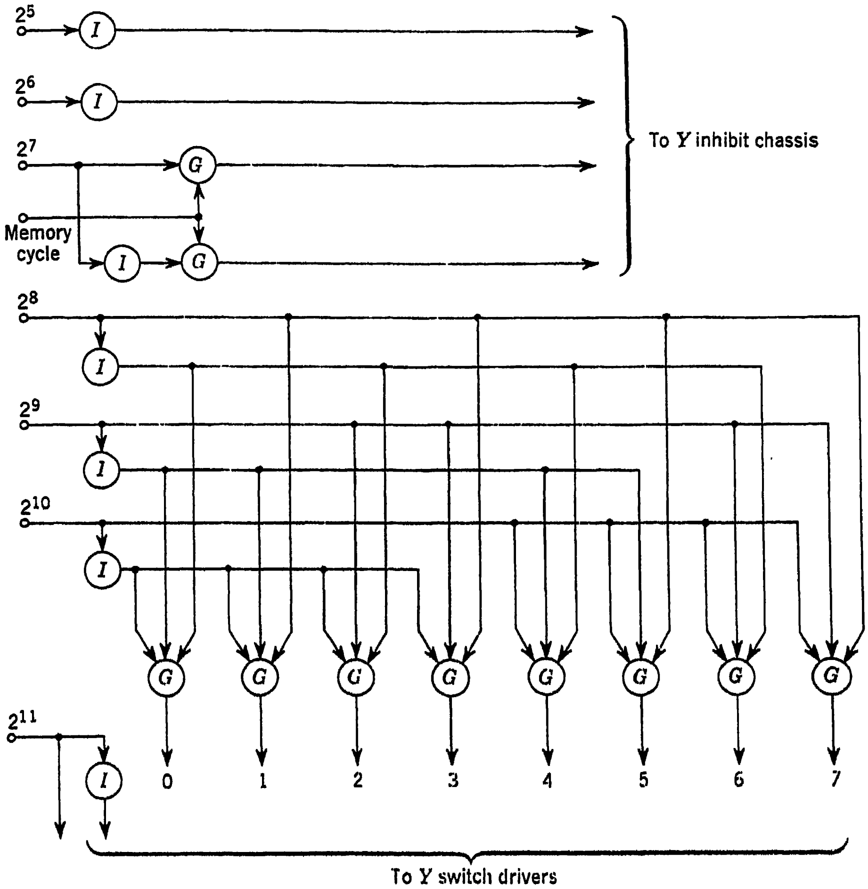


FIG. 14-5. Y-address partial decoder.

another in sequence (X_P, then X_N). The outputs of these gates, if coincidence does not occur, inhibit the flow of current to or from an X line of the memory arrays. If the inputs 1 and 2 coincide, then, during X_P, inhibition does not occur, and current is drawn from the X line; during the following X_N pulse, current is supplied to the line. In both cases, a stabilizer carefully controls the magnitude of the current, holding it to within ± 1 per cent for a ± 20 per cent variation in the load impedance presented by the array (the magnitude of this impedance varies, of course, with the variation in the relative numbers of cores holding 1's and 0's). The fact that the Y-pulse amplitudes are not accurately controllable means that accurate regulation of the total driving current in the selected cores depends upon the accuracy of regulation of the X-line current.

14-7. The Memory Cycle. Having seen how cores in the memory planes are selected, turn now to the insertion of information in the selected cores and its recovery from them. The general procedure is quite simple: the selected cores are read destructively; then either (1) given a read instruction, the information is restored, or (2) given a write instruction, the new information is recorded. In both cases, the cycle is "read" followed by "write"; the only difference is in the source of the information inserted during the portion of the cycle devoted to writing. The read part of the cycle lasts 9 μsec, and the write part 6 μsec, as determined by

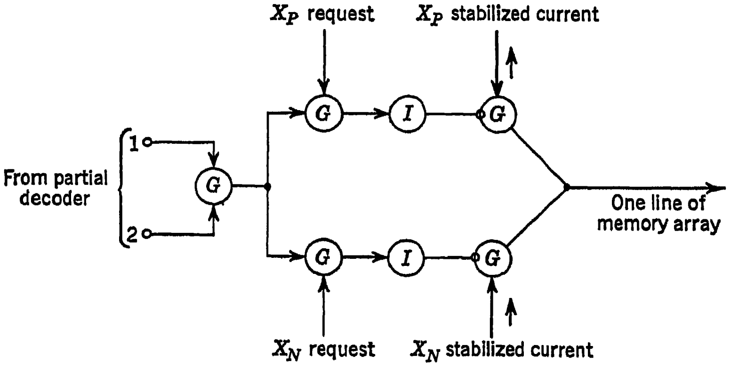


FIG. 14-6. X_P/X_N switch.

the X_P and X_N pulses. The origin of these pulses will be discussed later; the way in which they are permitted to affect the selected X line is shown in Fig. 14-7.

The Y_P pulse does not occur until 2.2 μsec after the beginning of X_P and it lasts but 3.2 μsec; it is during Y_P that the selected memory cores are read and in the process driven to negative saturation (i.e., the 0 state). If a 0 is to be restored (or written) in any selected core, then no further action is taken, but if a 1 is to be restored (or written), the selected core is driven to positive saturation, during the X_N pulse, by the magnetizing currents provided by the X_N pulse and by Y_{N_1}, which starts at 11 μsec and lasts nearly until 15 μsec.

Since the Y_N pulse restores cores of the magnetic switches to negative saturation, it must in every case exist, for this restoration must be accomplished. Hence, as will be shown later on, the circuits are so arranged that, if 0 is to be written, Y_N occurs shortly after Y_P and, although it overlaps X_N, the restore pulse due to this restoration of the magnetic-switch cores is over by the time X_N starts, in which case it is called Y_{N_1}; but, if 1 is to be written, Y_N is the Y_{N_2} already referred to. These pulses are shown in Fig. 14-7; in the next few sections it will be seen how they are generated and applied.

14-8. The Univibrators. The pulses needed to produce the memory cycle are generated by a chain of monostable circuits (i.e., of the character of a one-shot multivibrator) and a few toggles. Each of the monostable circuits, or Univibrators, as they are called, produces two pulses: one that begins shortly after the stimulation of the circuit (the lag is a few tenths of a microsecond) and lasts for a time determined by a time constant of the circuit, and another, or "end" pulse, which starts when the circuit returns to equilibrium and lasts in most cases for about 1 μsec.

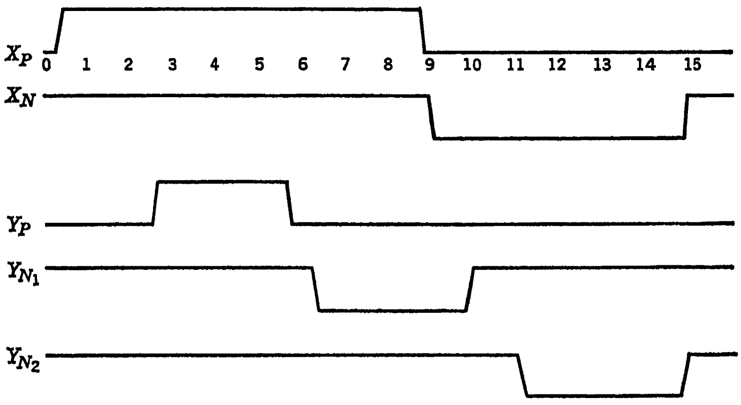


FIG. 14-7. X and Y pulses.

In Fig. 14-8, the main and end pulses of the Univibrator n are called P_n and P_{nE}.

The action is initiated by a pulse coming from the ORDVAC control circuits, which is called the "memory-request" pulse. This pulse stimulates U_1 and U_7, as shown; then U_2 is stimulated by P_{1E}, U_3 by P_{2E}, U_4 by P_3, U_5 by P_{3E}, and U_6 by P_{5E}. If the beginning of the memory request is taken as 0 μsec, then P_1 starts at about 0.3 μsec and ends at 2.3 μsec. P_2 starts at 2.5 μsec and ends at 5.7 μsec; P_3 and P_4 start at 6.1 and 6.2 μsec and last 2.8 and 3.8 μsec, respectively. P_5 starts at 9.0 μsec and lasts 1.8 μsec; P_6 starts at 11.0 μsec and lasts for 3.6 μsec. P_{6E}, which signifies the termination of the memory cycle, then begins at 14.8 μsec and ends at 15.9 μsec. P_7 starts with P_1 and ends at 4.1 μsec, and P_{7E} starts at 4.1 μsec and ends at 5.9 μsec, being somewhat longer than the other end pulses.

In some cases these pulses are used directly, and in others they are used to switch toggles whose outputs are used to time the cycle. Figure 14-8 shows that P_1 turns on the cycle-mixer toggle, which is finally turned off by P_{6E}. This toggle, while it is on, permits the Y-inhibit decoder to operate. The X_P and X_N toggles are turned on and off by P_1 and P_{3E}, and by P_{3E} and P_{6E}, respectively; the "on" time of each represents the duration of the X_P and X_N pulses. P_1 is used to clear the "digit" toggles, which are used to hold information inserted in the cores and to receive information read from them. P_2 is the Y_P pulse and, hence, defines the time at which reading is accomplished. P_4 and P_6 are the Y_{N_1} and Y_{N_2} pulses. The various pulses provided by the Univibrator chain are also

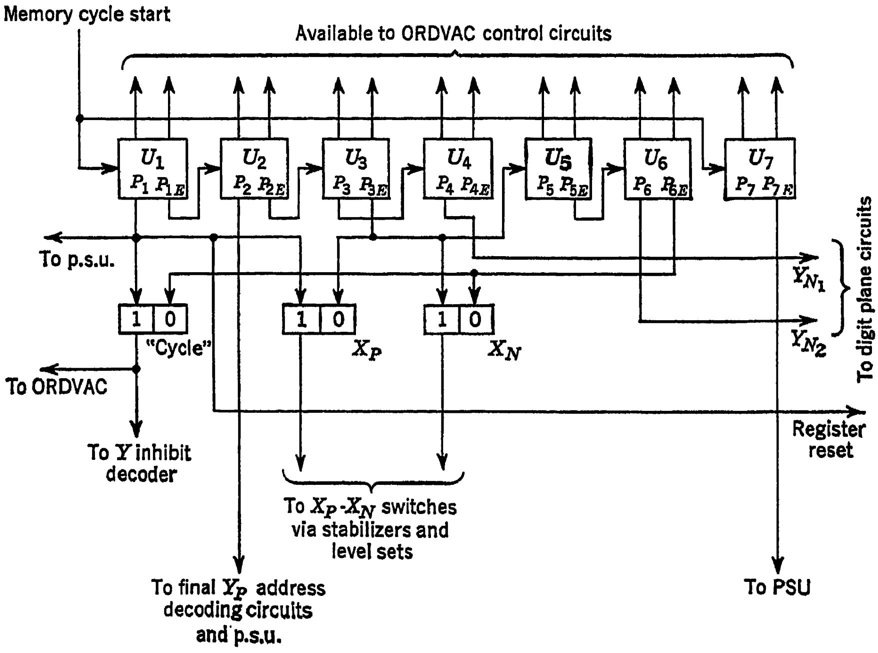


FIG. 14-8. Univibrator chain.

made available to the computer to accomplish the transfer of information between the registers RIII and RI and the digit register of the memory. Furthermore, as ORDVAC is an asynchronous machine, it is necessary that the completion of the memory cycle be signified to the control circuits, so that they can cause the computer to proceed to the execution of the next instruction. Figure 14-9 shows in detail the timing described above.

14-9. The Circuits Associated with the Digit Planes. Associated with each digit plane are circuits used in the actual transfer of information: (1) a reading amplifier, (2) a toggle, (3) a set of gates. There are also circuits used to control the action of these in accordance with (1) information to be inserted in the memory, (2) the memory-cycle pulse. These circuits are the partial-substitution unit and the register-reset circuits.

14-10. The Partial-substitution Units. These units receive from the control of ORDVAC signals which determine whether reading or writing is to be performed. The digit planes are divided into four groups of 12 and 8 each, corresponding to the division of each 20-bit instruction

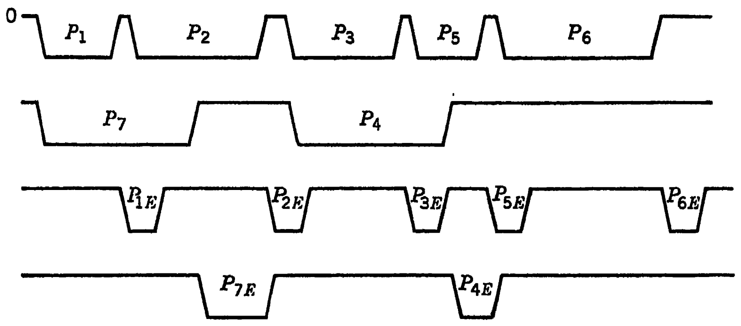


FIG. 14-9. Univibrator-chain outputs.

into an address and an operation portion, and each of the four groups is controlled by a separate partial-substitution unit (PSU). Hence, either the whole 40-bit word can be inserted or extracted, or either 20-bit instruction half, or only the address or the operation part of either of the two instructions in one complete word.

Each PSU receives, as shown in Fig. 14-10, the pulse P_{7E} from the timing chain, the Y_P pulse, and the read/write signal from the ORDVAC control. The two first are negative-going pulses, but the read/write signal is 0 volts for read and -35 volts for write. The diagram shows that, if read is specified, a read pulse is emitted in coincidence with the P_{7E} and that, if write is specified, the write pulse is emitted in coincidence with Y_P. The read/write signal is at the read level unless the particular segment of the word governed by an individual PSU is to have information inserted in it, in which case the ORDVAC control causes the signal to assume the write level.

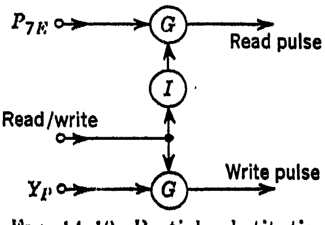


FIG. 14-10. Partial-substitution unit.

14-11. The Amplifier, Toggle, and Gates. The toggle receives information from the ORDVAC or from the selected core of the digit plane with which it is associated by way of certain gates. Its contents are available to the ORDVAC and are also used to control gates which determine whether Y_{N_1} or Y_{N_2} is permitted to clear the selected core of the magnetic switch to 0 and, hence, whether 0 or 1 is left in the selected core of its digit plane at the end of the cycle.

The amplifier, the toggle, and the gates are interconnected as shown in Fig. 14-11. The amplifier is a direct-coupled push-pull circuit of four stages with considerable negative feedback, having a gain of 400. Its output is positive if there is no input (corresponding to the reading of 0), but quite negative (−20 volts) if 1 is read. As shown in the drawing, the coincidence of a 1 with the read pulse from the PSU causes the toggle to be set to 1. There is no such input for 0, but P_1 always resets the toggle to 0 at the beginning of the cycle; hence the toggle always holds the correct information at the expiration of the read pulse.

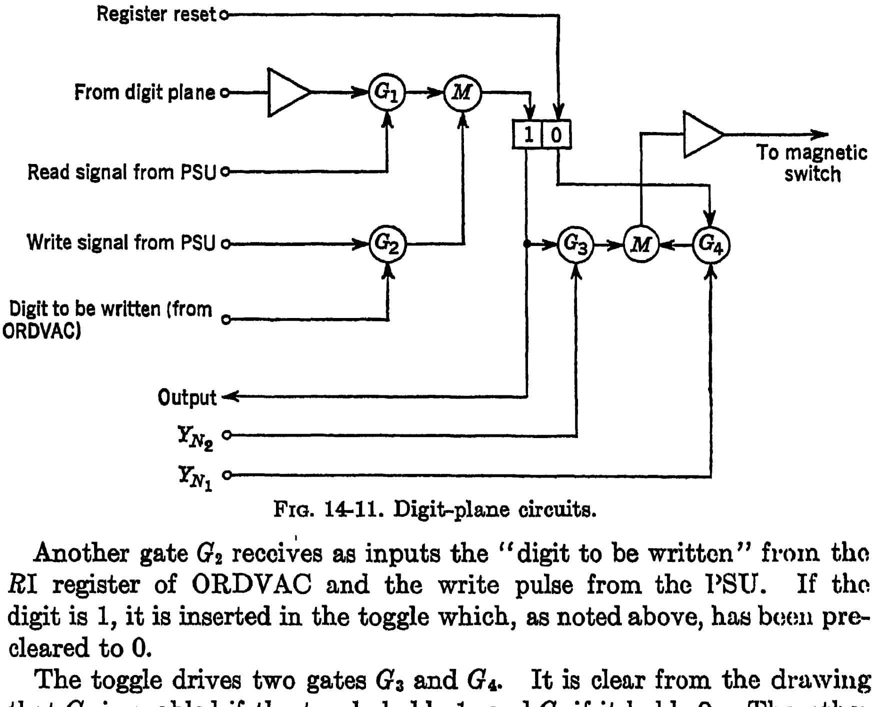


FIG. 14-11. Digit-plane circuits.

Another gate G_2 receives as inputs the "digit to be written" from the *R*I register of ORDVAC and the write pulse from the PSU. If the digit is 1, it is inserted in the toggle which, as noted above, has been precleared to 0.

The toggle drives two gates G_3 and G_4. It is clear from the drawing that G_3 is enabled if the toggle holds 1, and G_4 if it holds 0. The other inputs to these gates are Y_{N_2} and Y_{N_1}; so Y_{N_2} is transmitted to the magnetic switch if the toggle holds 1, but Y_{N_1} is transmitted if the toggle holds 0.

14-12. ORDVAC Drum Memory. In addition to the 4,096-word magnetic-core memory just described, ORDVAC is provided with a magnetic-drum memory[1] capable of holding 10,032 words. The drum itself is

[1] This memory was built for the Ballistic Research Laboratories by the (then) ERA division of Remington-Rand, Inc., now the Remington-Rand Univac Division of Sperry Rand, Inc., on an Army Ordnance contract. The permission of the company to include this description is gratefully acknowledged.

about 8.5 in. in diameter and 15 in. long. It rotates at 3,525 rpm. Information is held in 209 "data tracks," each of which holds 48 of the 40-bit ORDVAC words. On each track, the information density is fairly low (76 bits per in.). Recording is the simple return-to-zero method, using (say) positive magnetization to represent 1 and negative magnetization to represent 0.

Besides the data tracks, there are tracks that hold permanently recorded magnetic patterns that generate timing signals. These are (1) the reference-mark track, which holds a single mark giving the arbitrarily chosen reference position or "fiducial mark" from which all other positions about the drum are reckoned; (2) the word-mark track, which holds groups of marks corresponding to the beginning and end of each of the 48 word positions; and finally (3) the timing track, which holds 2,017 equally spaced marks corresponding to all possible bit positions and to the interword blanks.

Clearly, the selecting of any word position on the drum involves selecting the correct track and then determining the correct interval from the fiducial mark to the beginning of the word. It will be necessary to go into the question of address selection and timing fully below, after first disposing of the reading and writing circuits.

14-13. The Reading and Writing Circuits. Each bit of information is presented to the writing circuits as a pair of voltages on two wires; these are very simply derived from the two sides of the toggle that is the temporary repository of each bit of information to be written on the drum, and are, for example, + and − for 1, − and + for 0; they are labeled *A* and *B* in Fig. 14-12.

The gates G_0 and G_1 are both of the pentode type. Clearly, one and only one of these is enabled at a time, G_0 if the input bit is 0, G_1 if it is 1. The other input to G_0 and G_1 is the write pulse, 0.5 μsec in duration; before the write pulse arrives, the *A* and *B* voltages must be properly established for the bit to be written. When the write pulse gets through the proper gate, one of the two amplifiers A_0 and A_1 will be stimulated and will drive half of the primary of the step-down transformer *T*. The secondary of *T* drives lines connected to the 209 diode head switches, of which two are shown. Just one of these must be selected before the writing is to be accomplished; by way of the selected switch the proper head is driven by the amplified write pulse. The actual selection is accomplished by closing the relay contacts K_1, K_2, etc. Note that, with these contacts open, both diodes in the switch are nonconducting and, in fact, are heavily biased, so that they present a high impedance to any signal at all, whether applied by the transformer secondary or by the head winding. With the contact closed, the diodes are biased in the conducting direction, carrying a current of about 50 ma. In order to drive a greater current through the head winding it is clearly necessary that one of the diodes conduct in the reverse direction; the diodes used (1N91) are of the diffused-junction type, which possess this property of being able momentarily to pass a high current in the reverse direction just after having passed current in the forward direction.

The amplifiers A_0 and A_1 are simple, each consisting of a pentode (6AN5). The amplified write pulse applied to the primary of T is 75 volts (more or less) in amplitude, and it suffices to cause a current pulse of about 325 ma amplitude to flow through the selected head winding if a

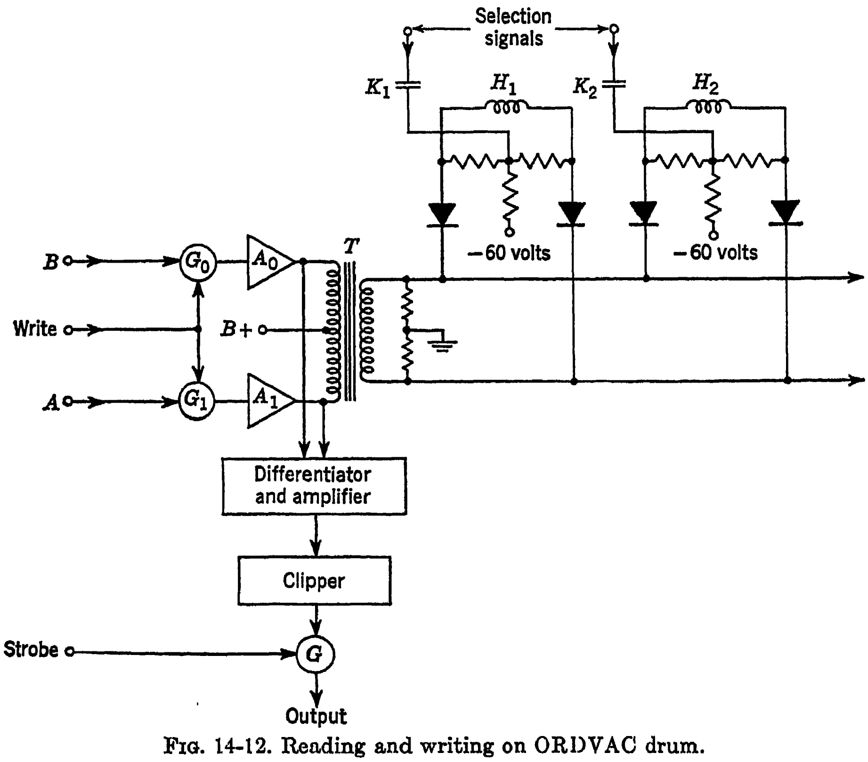


FIG. 14-12. Reading and writing on ORDVAC drum.

1 is written. Of course, if 0 is written, a current pulse of equal amplitude but opposite direction flows through this head winding.

The reading circuits shown in Fig. 14-12 are no more complicated than those just discussed. The head selection is accomplished in exactly the same way. When a 1 is read, a signal exactly like that shown in Fig. 11-2 appears across the head winding. This is stepped up by the transformer T and is then differentiated and amplified. Differentiation is very convenient here, because the falling trace occupies somewhat more than half of the signal's duration and is fairly steep, so that the differentiated signal consists mostly of a negative-going pulse occurring in the middle of the signal's duration. This pulse is amplified and clipped and, finally, strobed by a 0.5-μsec pulse, to give an output of standardized duration occurring at a precisely determined time so chosen as to compensate for delays due to transmission through the circuits.

In the case of the various timing tracks, some differences occur. Thus, the output of the timing track consists of 0.5-μsec pulses used to time the operation of the drum memory. These are produced ultimately by a tube normally conducting and having a parallel *LC* circuit in the plate lead. When the tube is cut off by the amplified pulse, the *LC* circuit starts to oscillate at its natural frequency but is permitted to go through only one half cycle, the opposite swing being damped out by a parallel diode and resistor, so that a pulsed standardized shape is obtained. This is done twice in sequence, so that the final result is a pulse occurring in the middle of the duration of the signal supplied by the head to the reading circuits. The reference-mark signal, occurring once in each revolution, is not narrowed in any way and emerges from the reference-mark-track amplifier as a pulse nearly 5 μsec long; it is used, as will be seen later, only to enable a gate to pass one of the timing-track output pulses.

14-14. Arrangement of Information on the Drum. The discussion so far has shown how a single bit of information is recorded or recovered. The system is so designed that an entire trackful of information, that is, 48 words of 40 bits each, is written or read in one operation; the trackful is, thus, the unit of information for purposes of transfer from and to the drum. In order to effect either transfer, it is obviously necessary to specify the number of the track involved; in addition to this, the address-decoding circuits must be informed that the address information is in fact ready for use, and a signal that causes the initiation of the process must be provided.

A detailed study of the timing and control operations will be given below. Before embarking upon this, it is necessary to show how the information is actually arranged in each information track. If the 48 words were arranged in purely sequential order and, hence, were read or written consecutively, the information rate would turn out to be nearly 120 kcps (actually 118.5 kcps). Since there is no real necessity to use such a high rate and since all the timing problems become simpler if a lower one is used, a three-phase interlocked pattern of recording is utilized. In this scheme three consecutive bit positions hold the first bits of three numbers, the next three positions the second bits, and so on. In reading or writing, one of these three numbers is dealt with in each of three consecutive revolutions of the drum, so that 16 numbers are dealt with in each revolution, and the information rate is only about 40 kcps (actually very nearly 39.5 kcps). The first three words occupy the 120 bit positions following the reference mark. They are followed by a space of a length corresponding to six bit positions. Then follow the next three words and a space, then three more words, and so on. After the last group of three words, a space equal to that occupied by 6 bits occurs just before the reference mark. Thus the total number of bit positions about the drum is $120 \times 16 + 6 \times 16 + 1 = 2{,}017$, the number of marks in the timing track.

The word-mark track has been mentioned above in Sec. 14-12. It holds a number of sets of three marks used to indicate the endings and beginnings of words. If the reference mark is called position 0, then the first set of three marks in the word-mark track occurs in positions 118, 119, 120, which are the positions of the last bits of the first three words. The next three marks are in positions 124, 125, 126, and indicate that the first bits of the next three words must occur in the three following positions 127, 128, 129. This pattern is repeated, the last set of three marks occurring in positions 2008, 2009, 2010, which are occupied in the information tracks by the last bits of the last three words. Thus, between two words in the order in which they are read or written there occur eight bit positions corresponding to a time interval of 67.6 μsec.

Each of the 48 words recorded on a single drum track is transmitted to the drum from *R*II of ORDVAC (the notation used is the same as in the description of the arithmetic unit of the IAS machine) by performing in sequence 40 left shifts and taking the bits as they emerge from the 2^0 end. The process is: (1) take a word from a specified address in the magnetic-core memory and put it into *R*II; (2) transmit it to the drum, as just described; (3) take the word in the next address in the magnetic-core memory and put it in *R*II; (4) transmit it to the drum; and then repeat steps 3 and 4 until 48 words located at consecutive addresses have been transmitted. In reading, a word from the drum enters the right (2^{-39}) end of *R*I one bit at a time, and a left shift is performed after each bit has been recorded. Thirty-nine left shifts cause the register to be properly filled; then the word is transmitted in parallel to the core memory. The instruction that controls the process specifies the address in the core memory to which the first word is to be transmitted, and the following words are transmitted to the 47 following addresses. Clearly, the order in which these consecutive 48 words are located on the drum is as follows: words 33, 17, 1 in the first block of 120 bit positions, words 34, 18, 2 in the next block, . . . , and, finally, words 48, 32, 16 in the last block. It should also be clear from the above discussion that, in both reading and writing, the highest-order (2^0) bit is dealt with first, then the next lower (2^{-1}), and so on down to that of lowest order (2^{-39}).

14-15. The Track-selecting Circuits. The track selection is performed simply and cheaply by relays, at the cost, naturally, of a somewhat longer time to effect the transfer operation than would be required if a faster type of switch had been employed.

The inputs from the main control of ORDVAC are the 8 bits 2^0 to 2^7, which specify the address of the track to be selected, and an "initiate-track-selection" signal. The signals specifying the bits of the address are slightly positive for 1's and negative for 0's; the initiate signal is negative as received from the main control but is inverted before it is applied as an enabling signal to the gates shown in Fig. 14-13. The

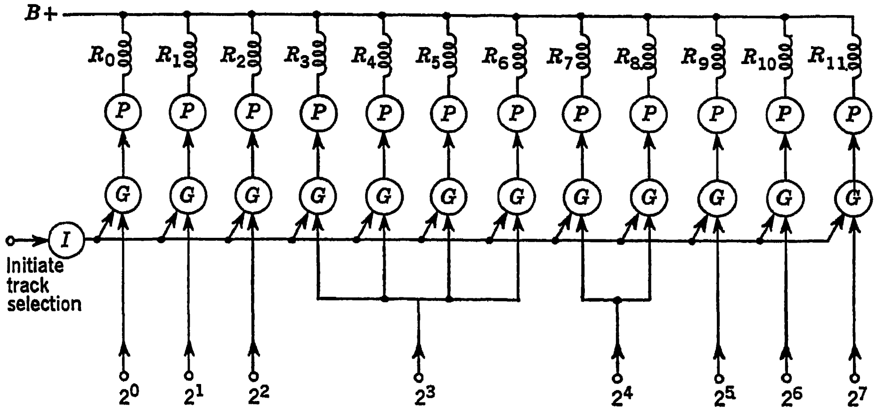


FIG. 14-13. Energizing the first rank of track-selecting relays.

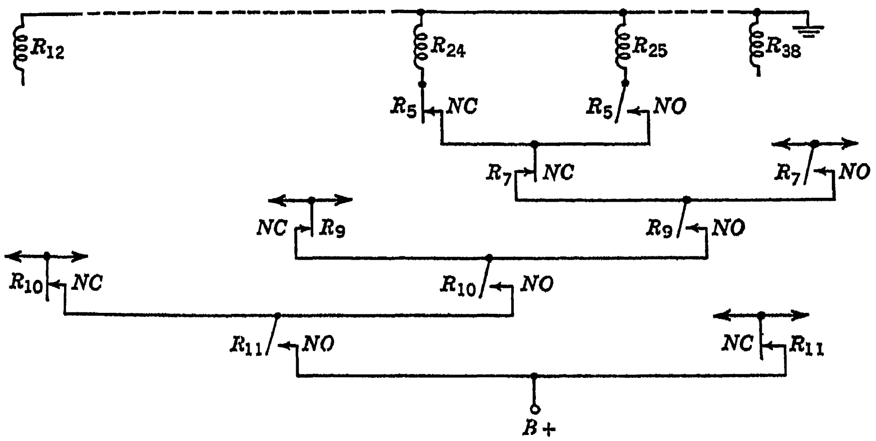


FIG. 14-14. Example of relay-selecting pyramid.

coincidence of two positive signals in any gate diode causes the grid of the puller tube P driven by it to be raised above cutoff and plate current to be drawn through the corresponding relay winding.

The 8 bits of the track address are actually decoded in two sets of 3 and 5 each, as shown in Fig. 14-14 and Fig. 14-15, and the outputs of these two pyramids (see, e.g., Sec. 7-5) are combined, as shown in Fig. 14-15, to effect the final selection. In Fig. 14-14, for example, NO and NC indicate that the contacts in question are open or closed in the unenergized condition of the relay and the opposite after it has been energized. Thus, when R_9 to R_{11} are energized and R_5 and R_7 are not, current flows through the winding of R_{24}, which is therefore energized when the last 5 bits of the track-selecting address are 00111. Of course, these 5 bits could be made to provide a selection of one of 32 relays, instead of the 27 shown in Fig. 14-14, if it were indeed necessary to provide for 256 track addresses instead of the 209 actually used. The pyramid for the decoding of 2^0, 2^1, 2^2 is shown in Fig. 14-15. Each of the eight outputs is connected to one contact on each of R_{12} to R_{38}; so these relays must be provided with eight sets of contacts. Hence every 8-bit track

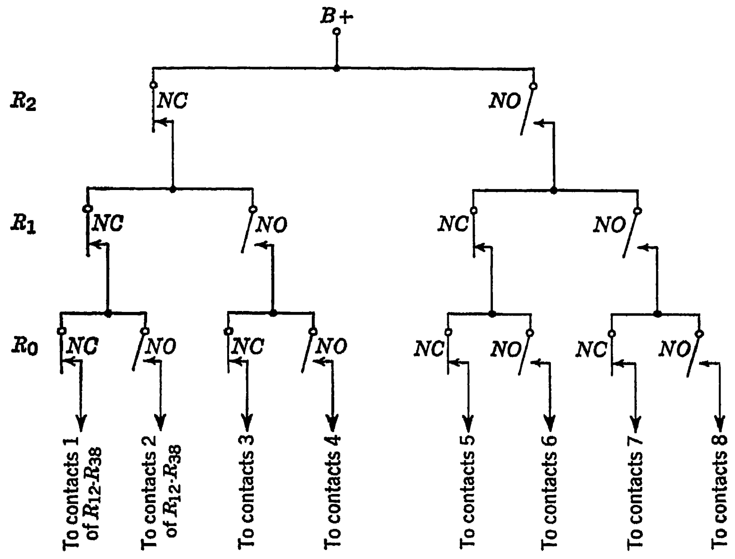


FIG. 14-15. Completion of the track selection.

address causes current to flow through just one of $8 \times 27 = 216$ pairs of relay contacts, of which two pairs K_1 and K_2 are shown in Fig. 14-12.

14-16. The Timing Circuits. Figure 14-16 shows a block diagram of the complete drum-control system. The blocks marked "reading," "writing," "track selection," and "head switching" have been discussed in some detail above; it is now necessary to consider the circuits that provide the properly timed stimuli required to control the operation of the other circuits.

The inputs to the drum memory from the rest of the ORDVAC system are the track address, the initiate-track-selection signal, the information to be recorded ("bits to drum" in the drawing), and the initiate-read and initiate-write signals. Outputs from the drum memory are the information read ("bits from drum"), a shift command that causes *RI* to be

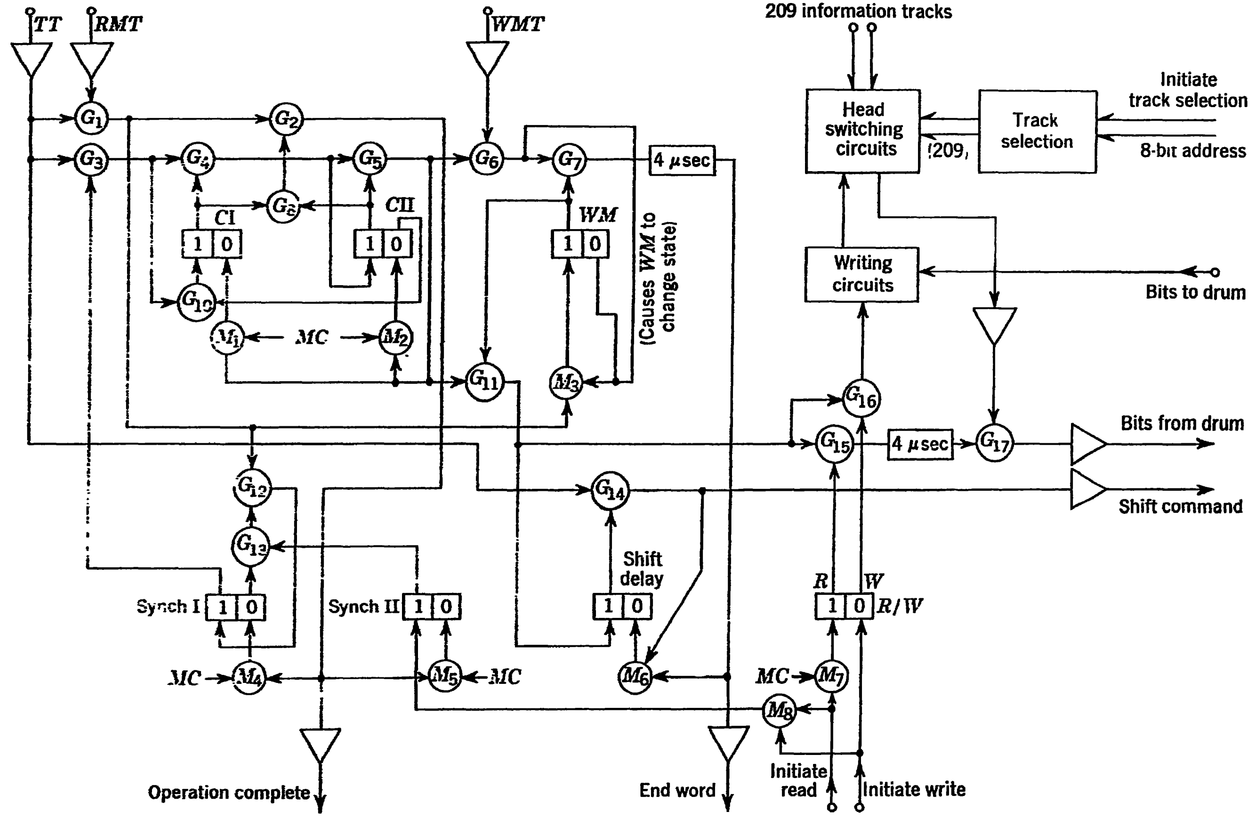


Fig. 14-16. Drum control.

shifted left one place after each bit from the drum has been recorded in its lowest-order stage, and signals indicating the end of each word and the completion of the entire transfer operation. These inputs and outputs are clearly indicated in Fig. 14-16. The only other inputs are those labeled "*MC*" for "manual control," used in testing. Notice also the channels of information flow between the drum control and the drum: information can be transferred in either direction to and from the 209 information tracks, and information flows into the control circuits from the timing (*TT*), reference-mark (*RM*), and word-mark (*WM*) tracks, from which all timing signals are derived by the rather simple circuits that receive this information.

In order to see how the timing circuits operate, suppose that the toggles *C*I, *C*II, *WM*, synch I, synch II, and shift delay have all been set to 0. Clearly the only thing that can happen is that, upon the next occurrence of the reference mark, the *WM* toggle is set to 1, and, after that, nothing at all can take place. Now suppose that either an initiate-read or initiate-write command is received. By the action of mixer M_8, synch II must be set to 1 [of course the *R/W* (read/write) toggle is set to 1 (read) or 0 (write)]. This now causes the enabling of G_{12} (via G_{13}), so that, when the next *RM* pulse occurs, the timing pulse P_0 that occurs within its duration sets synch I to 1, disabling gates G_{12} and G_{13} and enabling G_3. Clearly, also, the *WM* toggle is set to 1.

Before the next *RM* pulse arrives, there intervene 2,016 of the *TT* pulses, designated P_1 to P_{2016}. These find G_3 open and, therefore, pass through to gates G_4 and G_{10}. The two toggles *C*I and *C*II form a counter. P_1 finds G_{10} open and G_4 closed and, hence, sets *C*I to 1. P_2 then finds G_4 open (G_{10} is too, but this has no effect) and so passes through to set *C*II to 1. P_3 finds G_4 and G_5 both open and so passes through to reset *C*I and *C*II both to 0, to set the shift-delay toggle to 1, and to pass through either G_{15} or G_{16}, one of which must be open, to enter the writing circuits or to gate out 1 bit from the reading circuits (the 4-μsec delay called the "reading delay" is introduced to compensate for delays in the reading and writing circuits). P_3 also arrives at G_6 but finds it closed, for no *WM* (word-mark) pulses occur at the beginning of the first word after the reference mark.

Notice that the counter formed by *C*I and *C*II has only three possible states, 00, 10, 11; it is not possible to reset *C*I to 0 without doing the same to *C*II. For this reason, it is clear that every third *TT* pulse gets through G_4 and G_5 and resets the counter; it also gets through G_6 if a *WM* pulse occurs simultaneously with it, and through G_{11} if the *WM* toggle holds 1, and thence to the read or write circuits.

Now, to continue with the effects produced by the *TT* pulses, P_4 sets *C*I to 1 and passes through G_{14} to cause the bit transmitted to the lowest-order stage of RI by the delayed P_3 pulse to be shifted one place left, and so the lowest-order stage is prepared to receive the next bit from the drum. The transmitted shift command is also fed back to clear the shift-delay toggle to 0; it will again be set to 1 by P_6 and will transmit P_7 as a shift command, and so on.

This sequence continues until a pulse transmitted by G_5 finds G_6 enabled by a *WM* pulse. From the earlier discussion it is clear that the first *WM* pulses to be encountered occur simultaneously with P_{118}, P_{119}, P_{120}. Now, of these three *TT* pulses, only P_{120} gets through the gates G_4 and G_5, for P_{120} corresponds to the last or lowest-order position of the word whose first bit occurred simultaneously with P_3. Hence P_{120}, besides (1) resetting the counter, (2) setting the shift-delay toggle to 1, and (3) going to the read or write circuits, also (4) passes through G_6 and G_7 and the 4-μsec delay element to become the end-word pulse and to reset the shift-delay toggle in time to prevent a shift command from being issued, and finally (5) resets the *WM* toggle to 0. Notice that the mechanism for doing the last is to apply the signal to the toggle in such a way that the toggle is commanded to change its state rather than to assume a specified state. It should be noted here that the end-word pulse is transmitted to ORDVAC, where it causes the word just read out of the drum and now resident in R_1 to be transferred to the magnetic-core memory and also causes unity to be added to the memory address to which the next word is to be sent.

After P_{120}, the next three *TT* pulses, P_{121} to P_{123}, advance the counter through its usual cycle, but P_{123} is not transmitted to the read and write circuits nor does it set the shift-delay toggle, because G_{11} is held disabled by the *WM* toggle. The last of the next three pulses, P_{124} to P_{126}, however, coincides with a *WM* pulse, passes through G_6, and sets the *WM* toggle to 1. Hence P_{126} will pass through G_{11} to set the shift-delay toggle and to cause the appropriate gating in the read or write circuits, effecting the transfer of the first bit of the second word. The whole process then goes on just as already described.

Consider the situation at the end of the transfer of the 16th word, effected by P_{2010}. As no *WM* pulses coincide with any of the next 6 *TT* pulses P_{2011} to P_{2016}, these latter do nothing but advance the counter through two complete cycles, leaving it cleared to 00. P_0 then sets the counter to 10 and, because of the presence of the *RM* pulse, also sets the *WM* toggle to 1, just as at the beginning of the previous cycle. G_8 and G_2 are not yet enabled, and so the second revolution proceeds as the first; but notice that, in this revolution, P_2 instead of P_3 resets the counter, since P_0 sets it to 10. Hence the first bit transfer is effected by P_2, the next by P_5, and so on; the last bit of the word (the 17th to be transferred) coincides with P_{119}, the first bit of the 18th occurs at P_{128}, and so on. The last bit of the 32d word occurs simultaneously with P_{2009}, and the last bit of the 48th at P_{2008} of the third revolution. Hence the counter reaches 11 at P_{2016}, and the subsequent P_0, coinciding with the *RM* pulse, gets through G_2 and is transmitted back to main control to announce the completion of the operation, incidentally resetting both synch toggles to 0; of course P_0 also resets the counter to 0. Nothing more can happen until a fresh initiate-read or initiate-write pulse arrives to set the synch II toggle.

CHAPTER 15

THE CENTRAL CONTROL

15-1. Introduction. The central control of a digital computer is the most important part and also the most difficult to describe intelligibly. As in the last two chapters, no extensive general discussion will be attempted, but, after a brief general account of the unit's function, a rather detailed description of one notable unit—that in the Institute for Advanced Study (IAS) machine—will be presented.[1]

The central control is the brain, as it were, of a digital computer. Put briefly, at each step of the program it finds out what must be done next and then causes the machine to do it. Thus, it receives from the memory instructions that specify operations to be performed and the locations of operands. It decodes these and (1) sets up paths of information flow throughout the machine, (2) causes information to be obtained from the memory, and (3) initiates (and perhaps times) the performance of operations by the arithmetic and input-output units. Certain control functions, such as the detailed regulation of the arithmetic processes and the functions intimately connected with reading from or writing in the memory, are usually performed by subsidiary control units whose operations are initiated and supervised (perhaps even timed in a detailed way) by central control.

It is clear, then, that two types of output can be expected from the central control: (1) quasi-static voltages, set in the desired state and maintained there throughout all or part of the execution of an instruction, and (2) dynamic signals, usually voltage pulses, used for triggering other units or for timing their operation. There can be very wide diversity in the actual structure of the control. For example, in serial machines generally and in certain parallel ones (e.g., Whirlwind I), a central feature of the central control is a fairly precise oscillator which generates timing signals that are distributed to all parts of the machine; and, in asynchronous parallel machines like the IAS (and also BESK, EDSAC II, etc.), the dynamic signals do little more than set other units into opera-

[1] H. H. Goldstine, J. H. Pomerene, and C. V. L. Smith, Final Progress Report on the Physical Realization of an Electronic Computing Instrument, The Institute for Advanced Study, Princeton, N.J., January, 1954 (final report on United States Army Ordnance contract DA-36-034-ORD-19). See pp. 113–166.

tion, after which the central control does nothing until there are fed back to it reports that the tasks of other units have been completed. In fact, in the IAS machine, the central control does not really contain a master "clock" at all, but itself receives timing signals from the memory control.

15-2. Brief Outline of Whirlwind I Central Control. Before passing to the main topic of this chapter, it will not be amiss to give a brief outline of the Whirlwind I central control, since this makes a good and reasonably intelligible introduction to the subject and also because, though the Whirlwind I and IAS instruction codes are much alike, the contrast between the highly synchronized Whirlwind system and the asynchronous IAS system is so striking.

Before an instruction can be obtained from the memory, its location must be known. Whirlwind I is a single-address machine, and successive instructions are normally placed in successively numbered memory addresses. A program counter starts off at the beginning of the program holding the address of the first instruction of the program and increases its count by 1 as each instruction is obeyed; of course, jumps are also possible, both conditional and unconditional, but in any case the program counter always holds the address of the next instruction to be executed. Observe that, in the IAS machine, this is precisely the "order counter."

The address of the instruction to be obtained is transferred from the program counter to the memory switch, a toggle register that specifies the address to the memory and is, thus, equivalent to the center or "dispatch" rank of toggles in the IAS dispatch counter. The desired instruction is read out from the memory and placed in a program register, exactly analogous to R_3 of the IAS machine. From this, the 5 bits specifying the operation are transferred to five toggles driving a crystal matrix with 32 outputs, and the 11 bits specifying the address of the operand are transferred to the memory switch. The same general division is made in the IAS machine; the transfer of address information is made to the center rank of the dispatch counter (the decoding of the operation specification will be described in detail below). Various combinations of the 32 outputs of the crystal matrix and of the eight outputs of the time-pulse distributor (see below) provide appropriate enabling voltages to a great number (121) of pentode gate tubes, which thus establish the paths of information flow. The operand located at the address specified by the memory switch emerges from the memory and is delivered to its destination—generally, at least for arithmetic operations, the arithmetic register, which thus plays the same role as R^3. After this the operation can begin, and, finally, 1 is added to the contents of the program counter.

Besides these elements, the central control contains a group of timing circuits, known collectively as the "clock." These comprise a 2-Mcps oscillator, whose sinusoidal output is shaped to give a 2-Mcps pulse train; frequency dividers, which produce 1-Mcps and 62.5-kcps pulse trains; a restorer-pulse generator, which has no bearing upon the logic and so will be ignored hereafter; a time-pulse distributor, which is a matrix controlled by a three-place register and which distributes the 62.5-kcps pulses in blocks of eight in a fixed sequence; and a clock-pulse control, another matrix driven by four toggles, which causes pulses to be transmitted to the memory and the arithmetic unit for stopping and starting operations.

The basic cycle of the control is determined by eight pulses of the 62.5-kcps pulse train; these eight pulses are provided by the time-pulse distributor and they produce the following sequence of operations:

Pulse 1. Clear memory switch.

Pulse 2. Read out of program counter and into memory switch; clear the program register.

Pulse 3. Check transfer from program counter to memory switch.

Pulse 4. Read out of memory and into program register; clear the memory switch and the control switch (this brings the instruction to be executed into the program register). Preset step counter (in the arithmetic unit).

Pulse 5. Read from the program register into the control switch and the memory switch; read in to step counter (i.e., set it to the number of shifts to be performed, if a shift order).

Pulse 6. Check transfers to control switch and memory switch.

Pulse 7. Add 1 to contents of program counter.

Pulse 8. Check transfers; check circuits used for checking.

Notice that, right after pulse 5, the memory switch holds the address of the operand. The operand can forthwith be obtained from the memory and the performance of the operation initiated, without waiting for the end of the eight-pulse cycle of the central control; the necessary gates have been set up during the sixth step upon the transfer of the 5-bit coded operation into the control switch, and the only delay is the memory access time, which is known.

No further account will be given of the Whirlwind I central control; the above was included because it is clear and easy to understand. The IAS main control accomplishes all the same functions but is very different in internal structure, and the timing is rather more difficult to describe without going into detail. This should be kept in mind during the following detailed discussion.

15-3. The IAS Instruction Words. As noted more than once before, the IAS code is of the single-address type. It was designed for an internal memory of 1,024 words, and so 10 bits were allocated to each address. This means that each 40-bit word can contain two instructions comfortably; three would be too many, for then only 3 bits would be left for the operation part of each instruction unless the word length were increased. As 40-bit numbers seem long enough in general and as a single-address code seems to have considerable attractions, the existing scheme is reasonable and practical. Notice that it leaves 10 bits for encoding the instructions, as against only 5 for Whirlwind. This circumstance has been used to advantage in making the decoding extremely simple (see next section).

The allocation of space in each instruction word is as follows:

Phase 1		*Phase* 2	
Address	Operation	Address	Operation
2^{0} — 2^{-9}	2^{-10} — 2^{-19}	2^{-20} — 2^{-29}	2^{-30} — 2^{-39}

Each bit of the operation part of the instruction specifies some important feature of the operation to be performed. Each can be 1 or 0, referring to mutually exclusive alternatives. For example, 2^{-10} (and 2^{-30}) are written "step/no step," meaning that the machine, after executing the instruction under consideration, either proceeds to the next instruction automatically or stops; the first alternative is specified by 1, the second by 0. An inversion gives (step/no step)′, where 0 now means step and 1, no step. In general, at the output of the instruction synthesizer, which combines the individual bits of the operation to form the signals that actually control the flow of information in the machine, a 1 is signified by a negative voltage (−35 volts) and a 0 by a positive voltage (+5 volts). It must be understood that these voltages are quasi-static in the sense that they persist as long as the particular instruction is under consideration by control, that is, for the duration of a certain enabling signal (G_2 or G_4; see Fig. 15-2); at all other times the synthesizer's outputs are electrically in the 0 state.

The list of the functions specified by the various bits follows:

2^{-10} and 2^{-30}	Step/no step.
2^{-11} and 2^{-31}	Internal/external (int/ext). All orders are internal except those involving the input-output and magnetic-drum equipment.
2^{-12} and 2^{-32}	Williams/arithmetic (Wms/arith). "Williams" signifies that reference is to be made to the Williams memory, "arithmetic" that no such reference is to be made. The "arith" instruction bit was originally devised to specify left and right shifts that involve only the arithmetic unit.

2^{-13} and 2^{-33}	Not arithmetically trivial/arithmetically trivial (*NAT*/*AT*). The *AT* category refers to instructions in which no action of the arithmetic unit is involved.
2^{-14} and 2^{-34}	Absolute value/number (mag/no.). In the various addition instructions, the first alternative specifies that the absolute value of the number in R^3 is to be admitted to the adder.
2^{-15} and 2^{-35}	$-$Left/$+$right ($-L/+R$). In an addition order $-$left signifies subtraction and, elsewhere, left shift; $+$right signifies addition or right shift.
2^{-16} and 2^{-36}	$\Sigma/\div\times$. The "Σ" means that an addition or subtraction is to take place; "$\div\times$" means a multiplication or division (or a right or left shift).
2^{-17} and 2^{-37}	Round-off/no round-off (*RO*/*NRO*). This is too obvious to require comment.
2^{-18} and 2^{-38}	Clear/hold. "Clear" calls for clearing R_1 before the rest of the instruction is carried out.
2^{-19} and 2^{-39}	Quick sum/no quick sum (*QS*/*NQS*). *QS* calls for a special addition operation, in which the numbers at all addresses from the address specified in the instruction up to 1023 are added together.

The names given in the above list must not be taken too seriously. In some operations several of them are meaningless and the corresponding bits are used as engineering convenience dictates. The names are, however, sufficiently suggestive to be of very considerable aid in understanding the circuits.

The actual composition of the several instructions is given in Table 15-1. In the step/no step position, the arithmetic and the shift instructions may have 1 or 0 as desired; in the remaining instructions, this position must hold 1 or 0 as specified.

There are eight summation instructions. In the first, R_1 is first cleared to 0's, and the number coming from the memory is thus unchanged in passing through the adder; it is subsequently zigzagged down to R_1, for this is the standard method of transferring from the memory to R_1. In the second instruction, the number in R_1 is held and added to the number coming from the memory; then the sum is zigzagged into R_1 to replace the number originally there. The remaining six summation instructions provide for similar procedures using the negative, the absolute value, or the negative of the absolute value of the number coming from the memory. Besides these eight instructions, there is a quick-sum instruction, which causes the sum of the numbers at all memory addresses from a specific address up to 1023 to be formed. By putting 1 or 0 in the mag/no. and $-L/+R$ positions, various sums Σa_i, $\Sigma(-a_i)$, $\Sigma|a_i|$, $\Sigma(-|a_i|)$ can be formed. A quick-sum instruction must always be the first of the two instructions in a word.

The multiplication and division instructions do not require much comment after the detailed exposition of the arithmetic control's operation in Chap. 13. Note that, in multiplication, it is first necessary to put

TABLE 15-1. IAS OPERATIONS

Digital representation of operation bits 1.........	Step	Int	Wms	*NAT*	Mag	$-L$	Σ	*RO*	Clear	*QS*
0.........	No step	Ext	Arith	*AT*	No.	$+R$	$\times\div$	*NRO*	Hold	*NQS*
Operation	Code form									
Summation:										
1. Clear and add	1/0	1	1	1	0	0	1	0	1	0
2. Hold and add	1/0	1	1	1	0	0	1	0	0	0
3. Clear and subtract	1/0	1	1	1	0	1	1	0	1	0
4. Hold and subtract	1/0	1	1	1	0	1	1	0	0	0
5. Clear and add mag	1/0	1	1	1	1	0	1	0	1	0
6. Hold and add mag	1/0	1	1	1	1	0	1	0	0	0
7. Clear and subtract mag	1/0	1	1	1	1	1	1	0	1	0
8. Hold and subtract mag	1/0	1	1	1	1	1	1	0	0	0
Other NAT orders:										
9. Multiply (*NRO*)	1/0	1	1	1	0	0	0	0	1/0	0
10. Multiply (*RO*)	1/0	1	1	1	0	0	0	1	1/0	0
11. Divide	1/0	1	1	1	1	1	0	0	0	0
12. Load R_2	1/0	1	1	1	0	0	1	1	0	0
Arith. trivial orders (*AT*):										
13. Transfer to memory	1	1	1	0	1	0	1	0	0	0
14. Cl R_1 and transfer to memory (puts 0's in specified address)	1	1	1	0	1	0	1	0	1	0
15. Unconditional *TC* (no step)	0	1	1	0	1	0	0	0	0	0
16. Unconditional *TC* (step)	1	1	1	0	1	0	0	0	0	0
17. Conditional *TC* (no step)	0	1	1	0	1	1	0	0	0	0
18. Conditional *TC* (step)	1	1	1	0	1	1	0	0	0	0
Special order:										
19. Quick sum	1	1	1	1	1/0	1/0	1	0	1/0	1
Nonmemory orders:										
20. Shift right	1/0	1	0	1	0	0	0	0	1/0	0
21. Shift left	1/0	1	0	1	0	1	0	0	1/0	0
22. $R_2 \rightarrow R_1$	1/0	1	0	1	1/0	1/0	1	0	1/0	0
Input-output orders:										
23. IBM priming	1	0	1	1/0	1/0	1/0	1/0	1/0	1/0	0
24. Drum priming	1	0	1	1/0	1/0	1/0	1/0	1/0	1/0	0
25. IBM load	1	0	0	0	0	0	1	0	0	0
26. IBM punch	1	0	0	1	0	0	1	1	1	0
27. Drum $\rightarrow$ memory	1	0	1	0	0	1/0	1	0	0	0
28. Memory $\rightarrow$ drum	1	0	1	1	0	1/0	1	1	1	0

the -ier in its correct position by a load-R_2 instruction. In the subsequent multiplication instruction, it is possible to add to the product in the lowest 39 positions a number already residing in R_1 or not; which of these two courses of action is followed depends upon the bit in the clear/hold position of the instruction. As to division, the dividend must first be inserted in R_1 by a summation instruction before the division instruction can be executed.

The arithmetically trivial instructions call for no action by the arithmetic unit. They are six in number: two call for transfer from R_1 to the Williams memory, two for conditional transfer of control, and two for unconditional transfer of control. These are all specified quite unequivocally by the instruction code; the transfer-of-control (*TC*) instructions will be explained in detail below.

The nonmemory instructions call for either left or right shifts or for transfer from R_2 to R_1. These are not arithmetically trivial, because in all of them the arithmetic control must generate appropriate clears and gates. In the shift instructions, RI and RII are shifted together; the overflow from RI is fed into RII. Accordingly, let clear/hold be 0, and let a right shift of one place be called for; suppose that, before the execution of the shift, the contents of R_1, R_2, R^3 are

$$\begin{aligned}&R_1\colon a_0, a_1, \ldots\ldots, a_{39}\\&R_2\colon b_0, b_1, \ldots\ldots, b_{39}\\&R^3\colon \text{(irrelevant)}\end{aligned}$$

After the shift, they are

$$\begin{aligned}&R_1\colon a_0, a_0, a_1, \ldots, a_{38}\\&R_2\colon a_{39}, b_0, \ldots\ldots, b_{38}\\&R^3\colon b_0, b_1, \ldots\ldots, b_{39}\end{aligned}$$

for the contents of R_2 have been transferred to R^3 before the shift takes place. If a shift of more than one place is called for, this process is merely iterated. If, however, clear/hold is 1 and a right shift of one place is called for, then R_1 is first cleared to 0's; but it must be remembered that a_0 is also resident in the extra 2^1RI toggle, so that it appears in the 2^0 position after the shift, and the result is

$$\begin{aligned}&R_1\colon a_0, 0, 0, \ldots\ldots, 0\\&R_2\colon 0, b_0, b_1, \ldots\ldots, b_{38}\\&R^3\colon b_0, b_1, \ldots\ldots, b_{39}\end{aligned}$$

In the left shift of one place, with "hold" specified, the result is

$$\begin{aligned}&R_1\colon a_1, a_2, \ldots\ldots, a_{39}, 0\\&R_2\colon b_1, b_2, \ldots\ldots, b_{39}, a_0\\&R^3\colon b_0, b_1, \ldots\ldots, b_{39}\end{aligned}$$

but, if "clear" is called for, it is

$$\begin{aligned}&R_1\colon 0, 0, \ldots\ldots\ldots, 0\\&R_2\colon b_1, b_2, \ldots\ldots, b_{38}, 0\\&R^3\colon b_0, b_1, \ldots\ldots, b_{39}\end{aligned}$$

Iterations of these processes generate left shifts of more than one place. In all the shift instructions, the number of places to be shifted is specified in the section normally used for the address.

The $R_2 \rightarrow R_1$ instruction is actually any one of eight instructions, depending upon the bits in mag/no., $-L/+R$, and clear/hold; essentially these are the same as the summation instructions except that the source of the number is R_2 rather than the Williams memory. For these, the address part consists of 0's. If the last six positions of the address section do not all hold 0's, the number so represented, considered as an integer, causes the contents of R_2 to be shifted before being added to the contents of R_1; this will not be considered further.

Finally, consider the input-output and magnetic-drum instructions. A "priming" instruction, which is the first of the pair of instructions needed to execute operations of this kind, must always be in the second phase of a word. The address portion is transferred into a special 10-bit register in the unit that controls the input-output and the transfers between the Williams memory and the magnetic drum. The memory local control also obtains the number in the corresponding Williams-memory address, which is acted upon by the arithmetic unit; the result of such action is, of course, irrelevant. The only harm that can be done is that information held in R_1 (or R_2) can be lost; it is possible to pick all the other bits (except QS/NQS, which must be 0) in such a way that the irrelevant instruction actually executed does no damage. The IBM instructions are all completely specified and so need no comment; a good deal will be said about their execution in Chap. 16. As regards the drum $\leftrightarrow$ Williams instructions, the 80 tracks on the drum are divided into two sets of 40 each, which are designated A and B. The $-L/+R$ bit (the only one in either instruction that can be either 1 or 0) specifies which set of tracks is to be consulted, 0 signifying set A, 1 set B.

15-4. The Synthesizer. Consider now the actual structure of the main control.[1] As good a place to start as any is with the "synthesizer," that is, the circuit which receives from R_3 (the program register) the 10 bits of the instruction that specify the operation to be performed and which produces from these 10 bits the voltage signals that set up the various control elements of the memory, arithmetic unit, IBM unit, and magnetic drum. A logical diagram of this circuit is shown in Fig. 15-1. The toggles are those of stages 2^{-10} to 2^{-19} of R_3. Read-out from them is accomplished by 10 gates, which are enabled by a signal labeled G_2; a similar set of gates reads out from the toggles of stages 2^{-30} to 2^{-39}

[1] The arithmetic control has been discussed in detail in Chap. 13, and some mention has been made of the memory local control in Chap. 14. The dispatch counter described in Chap. 9 is also a part of this control, and its action should be reviewed before reading what follows.

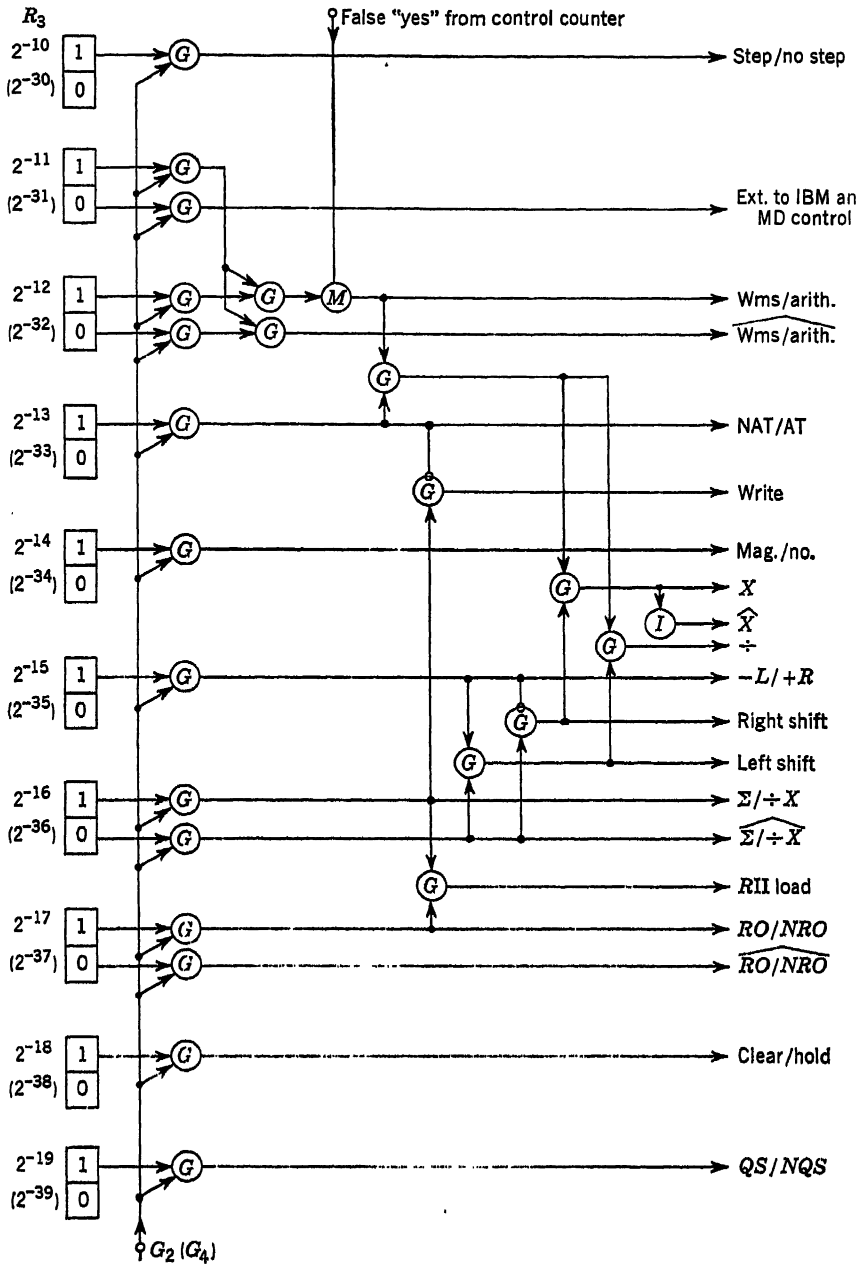


FIG. 15-1. The synthesizer.

when the gates are enabled by a signal G_4; the two sets of gate outputs are applied to exactly the same points of the circuit. Both G_2 and G_4 are generated elsewhere in the main control, as will be shown below, and are voltage steps falling from $+10$ volts (disable) to -2 volts (enable); only one can assume the enabling level at a time, depending upon which of the two instructions of the word in R_3 is to be obeyed.

The synthesizer is seen at a glance to be extremely simple in structure. Many of its outputs merely pass on bits of R_3 directly; others are formed by very simple combinations of two or more bits. The negative voltage (-35 volts) of an output is represented by 1, the positive ($+5$ volts) by 0. Note that the write output is affirmed by the 0 state ($+5$ volts) as a matter of convenience in actuating the memory local control. There are no subtleties to explain in the circuit, so the more complicated parts of the control may now be discussed.

15-5. The Control Counter. The heart of the main control is a counter that determines the sequence in which the two instructions (left and right) standing in R_3 are to be obeyed and then instructs the memory control to obtain the next instruction word and deliver it to R_3. The signals used to step the counter are a certain voltage, which can assume two states "up" and "down," and the $M \rightarrow R_3$ gate. More will be said later concerning the precise method by which "up" and "down" are generated, but for the moment it will suffice to observe that "up" rises with the synch signal fed back from the memory control or with an artificial synch signal generated in the part of the main control that supervises the execution of nonmemory instructions. "Down" occurs essentially when the instruction under consideration has been executed, and so the duration of the "up" signal depends entirely upon the instruction under consideration.

The actual operation of the counter can be understood only by referring to Fig. 15-2, which shows its logical structure. To begin with, assume that all toggles hold 0's. The arrival of the "up" signal switches T_T to 1. Subsequently "down" switches $T_{0/1}$ to 1. The next "up" switches T_T to 0, and the next "down" switches T_F to 1. The third "up" switches T_L to 1, and the third "down" switches $T_{0/1}$ to 0 and T_F to 0 (it must be remembered that both "up" and "down" last for times that are long compared to the time required to flip a toggle). The condition $T_L = 1$ permits the transfer of a new instruction from the memory to R_3 by enabling the gate through which the $M \rightarrow R_3$ gate command pulse must pass. The actual gating voltage, signifying the execution of the "fetch" operation, is fed back to switch T_L to 0. The circuit has now been returned to its initial state. This is summarized in Table 15-2. This discussion assumes that "step" is affirmed in both instructions in the word in R_3, so that the gates into T_T are enabled; otherwise "up"

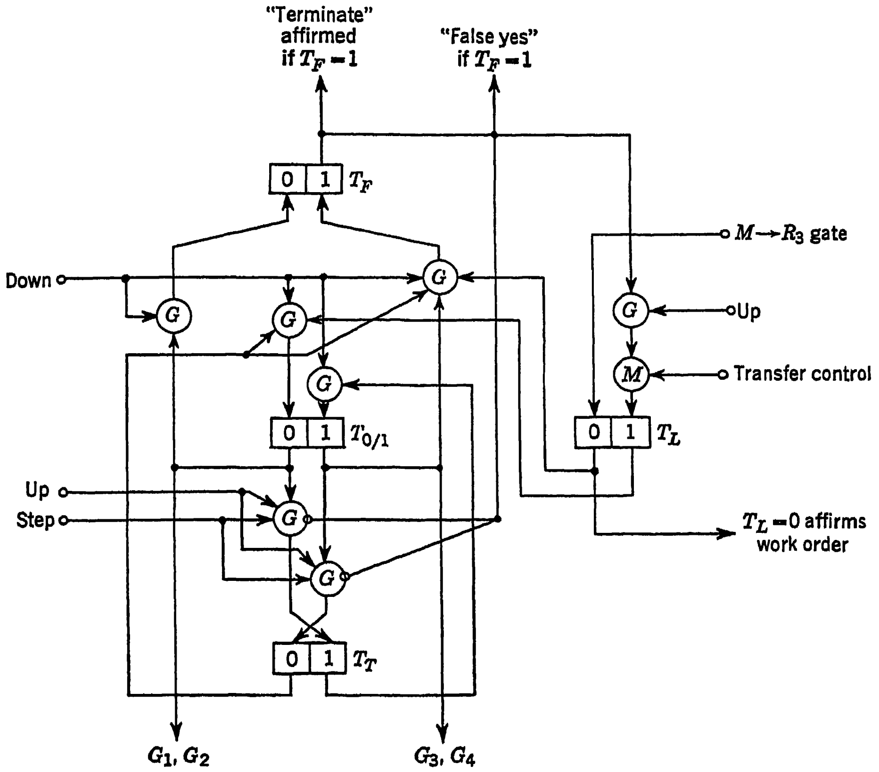


FIG. 15-2. The control counter.

TABLE 15-2

	$T_{0/1}$	T_T	T_F	T_L	
Initial state	0	0	0	0	
1st up	0	1	0	0	Left instruction
1st down	1	1	0	0	
2d up	1	0	0	0	Right instruction
2d down	1	0	1	0	
3d up	1	0	1	1	"Fetch"
3d down	0	0	0	1	
$M \to R_3$ gate	0	0	0	0	

could not switch T_T. Notice also the transfer-control input, which can switch T_L to 1. This is affirmed only in transfer-of-control instructions; the functioning of the control in these will be explained after the reader has become reasonably familiar with the main features.

There are several outputs from the counter. From $T_{0/1}$ are taken voltages which, under circumstances to be explained in the next section, make the contents of 2^0 to $2^{-9}R_3$ and of 2^{-10} to $2^{-19}R_3$ available to the address generator and the synthesizer, respectively, if $T_{0/1} = 0$, but which use instead the contents of 2^{-20} to $2^{-29}R_3$ and of 2^{-30} to $2^{-39}R_3$ if $T_{0/1} = 1$. It is, therefore, the state of $T_{0/1}$ that determines which of the two instructions standing in R_3 is to be executed.

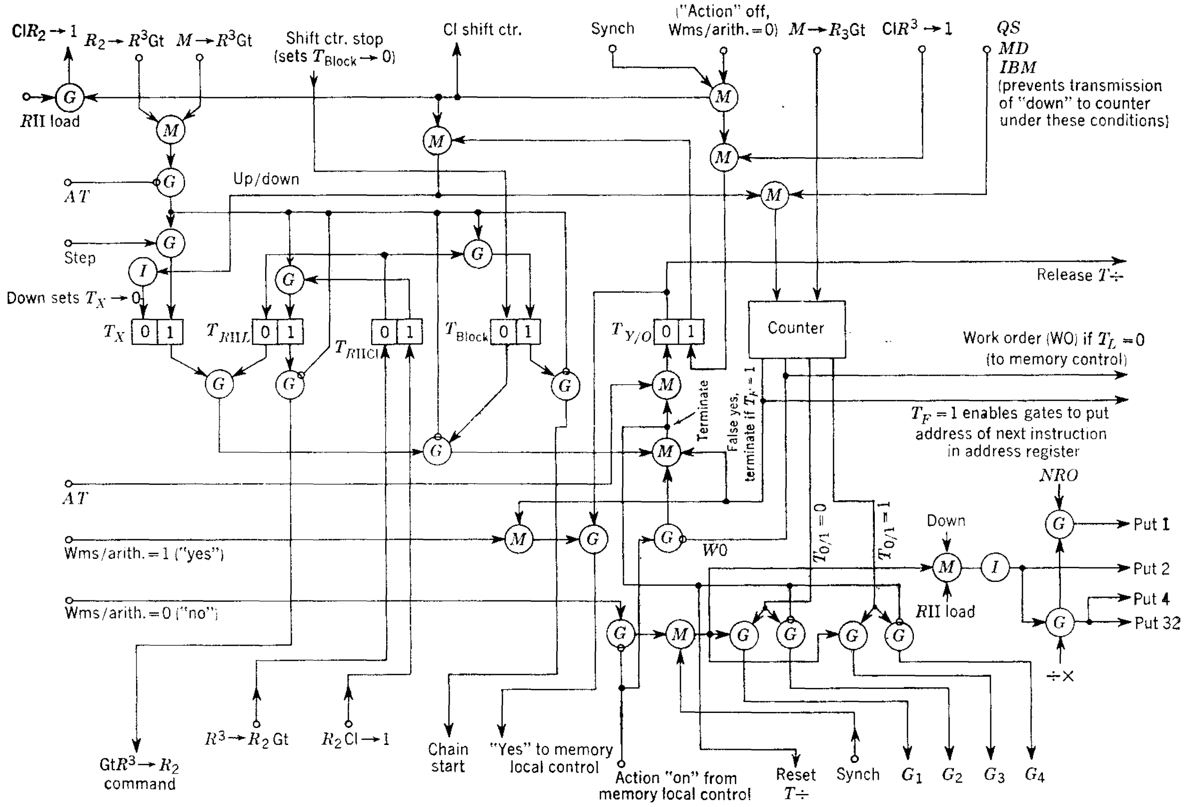
Outputs from T_F are (1) a "false yes," which is the request to use the memory in a fetch cycle, (2) a signal that mixes with others to produce the "terminate" signal, which resets $T_{Y/O}$ to 0, and (3) a signal to the B_0 gates, enabling them if $T_F = 1$ and thus causing the address of the next instruction word to be read into the address generator. Finally, from T_L is taken the work order (WO) signal to the memory control, which is affirmed if $T_L = 0$.

15-6. Structure of the Main Control (Arithmetic Instructions). Now that the counter has been dealt with, it is fairly easy to draw a logical diagram of the main control. To avoid an overly complicated diagram, this has been split up into several. The first diagram, which covers the features of importance in the arithmetic operations, is Fig. 15-3. Figures 15-4 and 15-5 show those features used in the $R_2 \rightarrow R^3$ transfer and in the transfer-of-control instructions.

In Fig. 15-3 the counter is represented, for simplicity, by a single block. The remaining elements are five toggles and associated logic circuits. The toggles T_{RIICl} and T_{RIIL} function only in the execution of the RII-load instructions. T_X exists only to prevent "terminate" from being issued at the wrong time, $T_{Y/O}$ is turned on to terminate the yes signal to the memory control, and T_{block} starts the gate-clear sequencing chain in the arithmetic control. Two other toggles associated with the $R_2 \rightarrow R^3$ transfer will be shown in Fig. 15-4.

The inputs to the main control are (1) from the synthesizer, (2) from the arithmetic control, and (3) from the memory local control, and outputs are transmitted back to these circuits. The outputs to the synthesizer are G_2 and G_4, which determine whether the operation part of the first (left) or second (right) instruction is to be decoded. The outputs to the memory control indicate whether the Williams memory is to be used ("yes") or not in the instruction under consideration and set up the gates to transfer the address from which information is to be obtained into the address-generator toggles of the dispatch counter. The output to the arithmetic control is the chain-start command. These units also, of course, receive information from the synthesizer, which is part of the main control. Such information is of the "static" type; it does not change during the execution of the instruction.

In the discussion of the counter, the importance of the up/down signals was noted. The derivation of these is clearly shown in Fig. 15-3. For simplicity, consider here only arithmetic instructions; the other



ClR₂ → 1
R₂ → R³Gt
M → R³Gt
Shift ctr. stop (sets T_Block → 0)
Cl shift ctr.
Synch
("Action" off, Wms/arith. = 0)
M → R₃Gt
ClR³ → 1
QS
MD
IBM
(prevents transmission of "down" to counter under these conditions)
RII load
AT
Up/down
Step
Down sets T_X → 0
T_X
T_RIIL
T_RIICl
T_Block
T_Y/O
Counter
Release T÷
Work order (WO) if T_L = 0 (to memory control)
T_F = 1 enables gates to put address of next instruction in address register
Terminate
False yes, terminate if T_F = 1
NRO
AT
Wms/arith. = 1 ("yes")
WO
T_0/1 = 0
T_0/1 = 1
Down
Put 1
Put 2
Put 4
Put 32
RII load
Wms/arith. = 0 ("no")
÷×
R³ → R₂ Gt
R₂ Cl → 1
GtR³ → R₂ command
Chain start
"Yes" to memory local control
Action "on" from memory local control
Reset T÷
Synch
G₁
G₂
G₃
G₄

FIG. 15-3. The main control.

instructions will be treated later. The synch signal from the memory control turns $T_{Y/O}$ on, thus terminating the signal that requests the use of the memory, and $T_{Y/O}$ continues to supply the up signal after the termination of synch. "Down" is the negation of "up" and, hence, occurs when $T_{Y/O}$ is turned off. The signal that accomplishes this is seen to be generated under any of the following circumstances:

1. $T_X = 1$, $T_{RIIL} = 0$, $T_{\text{block}} = 0$, and neither the $R_2 \rightarrow R^3$ nor the $M \rightarrow R^3$ gate is enabled.
2. $T_F = 1$.
3. $T_L = 1$ and T_{action} (in the memory control) $= 1$.

It is clear from Fig. 15-3 that case 1 obtains right after the shift counter in the arithmetic control has been stopped and T_{block} has, consequently, been turned off. Cases 2 and 3 occur during a fetch operation, as can be seen from Table 15-2, which shows the successive stages of the counter. At the conclusion of synch, the fact that $T_F = 1$ causes $T_{Y/O}$ to be turned off, thus generating "down" and turning T_F off. T_L and T_{action}, which are both on, keep the outputs G_1 to G_4 at their nonenabling levels until the $M \rightarrow R_3$ gate signal announces that the fetch has been completed and turns T_L off, resetting the counter to zero. Thus, since $T_{0/1} = 0$ and the terminate signal has been removed, G_2 immediately assumes its enabling value, and the synthesizer output becomes available to set up the control to cause the execution of the new left instruction in the memory cycle immediately following the fetch cycle.

In the arithmetic operations, it is always necessary to present to the recognition circuit the number of pairs of clear and gate operations that must be carried out. This is done by the "put" circuits shown in the lower right-hand corner of Fig. 15-3. The main input to these circuits is somewhat complicated. The inverter interchanges 0 and 1 or, physically, a high and a low voltage. Thus, the "put 2" output is affirmed by the following combination: up, (*R*II load)′, (synch)′, and either "action on" or arith/Wms = 0 (i.e., "yes"), for example, in an arithmetic instruction that is not an *R*II load, from the termination of synch until down. Under these circumstances, if the instruction is for addition, the put 4 and put 32 outputs are negated, and, therefore, so is put 1. On the other hand, if the instruction calls for either multiplication or division, then put 2, put 4, and put 32 are all affirmed together, and put 1 is affirmed if there is to be no round-off (*NRO*). In Fig. 13-2 these are shown in the upper left-hand corner as inputs to the recognition circuit (put 4 and put 32 have been combined as put 36, for convenience).

After these preliminaries have been understood, the best way to gain an understanding of the functioning of the main control is to follow it as it guides the other units through the steps necessary to execute some specific instruction. The example chosen here is the first of the two instructions needed to effect an addition: "clear R_1 and put into it the number at address X."

Referring to Table 15-1 of this chapter, note that the instruction is composed as follows (assuming that it is to be followed automatically by the next instruction):

step/no step	= 1	$-L/+R$	= 0
int/ext	= 1	$\Sigma/\div\times$	= 1
Wms/arith	= 1	RO/NRO	= 0
NAT/AT	= 1	clear/hold	= 1
mag/no.	= 0	QS/NQS	= 0

Assume further that this is the first (left) of the two instructions in R_3, which have just been fetched from the memory, so that the control counter is at zero and the five toggles shown in Fig. 15-3 all hold 0. Hence G_2 is at its enabling level, and the gates from 2^{-10} to $2^{-19}R_3$ into the synthesizer are open. Reference to Fig. 15-1 shows that the combined outputs $\times$, $\div$, RII load, right shift, left shift, and write are all denied.

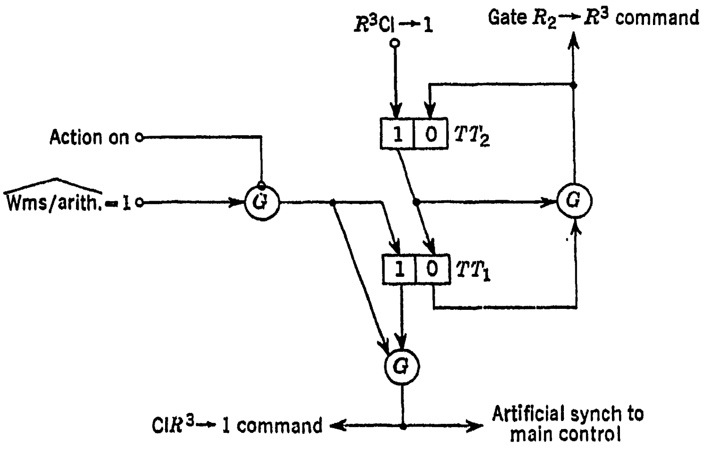
From Fig. 15-3 it is evident that the "yes" output is affirmed. This turns on T_{yes} in the memory local control (see Sec. 14-1), which in turn permits T_{synch} and T_{action} to be turned on by appropriate pulser outputs of the regenerate cycle immediately preceding the action cycle in which the number at address X is to be read, and to set up the local control in such a way as to guarantee the execution of an action cycle. That this is to be a read cycle is assured since writing can take place only if T_{store} is turned on by the affirmation of the write output of the synthesizer, which is certainly not the case (see Fig. 15-1), since NAT is affirmed. The synch signal which is fed back to the main control does several things: it initiates "up" and turns $T_{Y/O}$ on; it inhibits the put outputs; it enables the gate whose output is G_1; and it causes the shift counter to be cleared. The consequences of these actions are: (1) the yes signal to the memory control is denied, and so, at the beginning of the action (read) cycle, T_{yes} can be turned off, guaranteeing that a regenerate cycle will follow; (2) T_T is turned on; and (3) the gates from the address part (2^0 to $2^{-9}R_3$) of the instruction being obeyed are enabled by G_1, so that just before the termination of synch, the address is transmitted to the address generator. Hence, upon the termination of synch, there is no need for G_1 to remain at its enabling level; accordingly, G_1 terminates with synch, as Fig. 15-3 clearly shows.

Thus, at the end of the regenerate cycle preceding the read cycle, the address of the operand is in the address generator, the deflection voltages are set up accordingly, and the cathode-ray tubes are prepared for the read-out to R^3, which was cleared simultaneously with the transmission of the address from R_3 to the address generator; this operation is accomplished by the $M \rightarrow R^3$ gate right at the beginning of the read cycle. This gate pulse is fed back to the main control. It has no effect upon the counter, as can be readily seen from Fig. 15-3. Since the instruction is not arithmetically trivial (*NAT*) and since it contains "step," the $M \rightarrow R^3$ gate pulse turns T_X and T_{block} on.

The arithmetic phase of the operation now begins. Already at the termination of synch, all conditions were satisfied for enabling put 2; the other put circuits are disabled because of the nature of the instruction (Σ and *NRO*). Immediately at the termination of the $M \rightarrow R^3$ gate, the chain-start command is issued by T_{block}. The arithmetic control takes charge and proceeds to zigzag the digit-resolver output into R_1 (after, of course, the appropriate carry delay). The instruction has now been executed; the shift-counter-stop signal is fed back to the main control and turns T_{block} off. Since all the necessary conditions are now satisfied, the terminate signal is generated, turns $T_{Y/O}$ off, and thus generates "down," which turns T_X off and $T_{0/1}$ on. Terminate, which causes G_2 to assume its disabling level, lasts only until T_X has been turned off. By this time $T_{0/1}$ is on; therefore, at the end of terminate, G_4 assumes its enabling level and the gates from 2^{-30} to $2^{-39}R_3$ are enabled, making the operation part of the right instruction available to the synthesizer.

The main control passes through exactly the same cycle for any of the $+, -, \times, \div$ operations; the only differences are in the durations of "up" and in the "put" outputs. The *R*II-load instruction, by which the multiplier is transferred from the memory into R_2 and which is, thus, the first of a pair of instructions specifying a multiplication, requires two special toggles, labeled T_{RIIL} and T_{RIICl} in Fig. 15-3. The composition of this instruction is given in Table 15-1; although the arithmetic unit does not function at all, the instruction is treated as *NAT*. As far as the memory is concerned, there is no difference. Note, however, that the shift counter is not started, nor is any "put" circuit activated; this is because the counter is not used at all. Note also that synch and *R*II load produce, by means of a gate in the extreme upper left corner of Fig. 15-3, a Cl $R_2 \rightarrow 1$ command. The resulting "clear" is fed back and turns T_{RIICl} on; this enables a gate by which $M \rightarrow R^3$ gate turns T_{RIIL} on, and disables the gate by which $M \rightarrow R^3$ gate normally turns T_{block} on in an arithmetic instruction. The consequences of these actions are that (1) immediately after $M \rightarrow R^3$ gate, T_{RIIL} transmits a command to the gate-driver drivers to open the gates from R^3 into R_2, and (2) no chain-start signal is issued. The states of the toggles immediately after $M \rightarrow R^3$ gate are, therefore, $T_X = 1$, $T_{RIIL} = 1$, $T_{RIICl} = 1$, $T_{\text{block}} = 0$. Evidently, terminate will be affirmed as soon as T_{RIIL} is turned off again, which is done by the fed-back $R^3 \rightarrow R_2$ gate signal. The instruction has now been completed. Terminate is affirmed, $T_{Y/O}$ is turned off, and "down" is generated to advance the control counter and so prepare for the execution of the next instruction.

15-7. Structure of the Main Control (Nonmemory Operations). The nonmemory operations are the left and right shifts and the $R_2 \rightarrow R_1$ transfer. In all of these, the operand has already been obtained from the memory and resides in R_1 or R_2; hence the memory is not used, and this is specified by (Wms/arith)′ = 1. These instructions use the arithmetic unit; so $NAT/AT = 1$. For the shifts, which are essentially divisions or multiplications, $(\Sigma/\div\times)' = 1$; but $\Sigma/\div\times = 1$ for the $R_2 \rightarrow R_1$ transfer, which is accomplished by first transferring the number in R_2 to



R^3Cl → 1
Gate $R_2 \rightarrow R^3$ command
Action on
$\widehat{\text{Wms/arith.}}$ = 1
G
1 0 TT_2
G
1 0 TT_1
G
ClR^3 → 1 command
Artificial synch to main control

FIG. 15-4. Nonmemory control.

R^3 and then adding it to the contents of R_1. As stated in Sec. 15-3, $R_2 \rightarrow R_1$ is any one of the eight add/subtract instructions, except that the source of the number added is R_2 instead of the memory. It is also possible to cause the number in R_2 to be shifted left or right before being added. In this and in the shift instructions, the number of places shifted is specified in the last six positions of the address part of the instruction.

The additional circuits in the main control needed to carry out these nonmemory NAT instructions are shown in Fig. 15-4. This unit is exceedingly simple, consisting of two toggles and a few gates. It has two functions: (1) to supply an artificial synch in place of the synch fed back from the memory control, and (2) to supply the clear and gate commands needed to effect the transfer from $R_2 \rightarrow R^3$. In all these instructions this transfer takes place; it is actually needed only in $R_2 \rightarrow R_1$ but can do no damage in the other cases; so there is no need to provide the circuits that would be required to suppress it.

Assume a right shift is the left of the pair of instructions in R_3. This is specified by the following:

step/no step	= 1	$-L/+R$	= 0
int/ext	= 1	$\Sigma/\div\times$	= 0
Wms/arith	= 0	RO/NRO	= 0
NAT/AT	= 1	clear/hold	= 0
mag/no.	= 0	QS/NQS	= 0

The number of places to be shifted is specified in the last six places of the address part of the instruction.

This instruction is read into R_3 by the $M \rightarrow R_3$ gate pulse of the fetch cycle, which is then fed back to main control, where it sets the counter to zero. Once T_L has been turned off, the fetch terminate is denied, permitting G_2 to assume its enabling level and thus to make the operation bits of the instruction available to the main control. As the request to use the memory is denied to the yes and synch toggles, these are not turned on. From Fig. 15-4 it is seen how the turning off of T_{action} at the end of the fetch cycle is simply exploited to produce an artificial synch, which is also fed back to turn $T_{Y/O}$ on. Synch is also used to cause G_1 to assume its enabling value. In the present case, the combination of "no" and "action off" does the same thing at the same time and also causes the "put" circuits to be disabled during the rest of the execution of the instruction—essential because the number to be recognized in this case is given by the last 6 bits of the address, special gates being provided to transmit them to the recognition circuit.

From Fig. 15-4 it is also seen that, when the action toggle is turned off, a clear $R^3 \rightarrow 1$ command is issued. The resulting "clear" is fed back to turn TT_2 on, which in turn turns TT_1 off again and causes the gate $R_2 \rightarrow R^3$ command to be issued. The ensuing gating operation is irrelevant in the shift, because no use of the number transmitted to R^3 is made, but the $R_2 \rightarrow R^3$ gate pulse is fed back to the main control (see the upper left-hand corner of Fig. 15-3), where it passes through a gate that is open because the operation is NAT and proceeds to turn T_X and T_{block} on (T_{RIIL} and T_{RIIOI}, of course, remain off throughout). Thus the chain-start signal is generated. The clear and gate invitations generated by the chain are so directed by the left/right selector and the clear-and-gate selectors as to cause the number in R_1 to be shifted right place by place until the shift-counter output satisfies the recognition circuit, whereupon the shift counter is stopped, and the signal announcing this to the main control turns off T_{block}. This causes terminate to be affirmed, turns $T_{Y/O}$ off, and generates "down," which steps the control counter and turns off T_X. Thus the operation is completed, and the main control is ready to cause the execution of the next (right) instruction.

The left shift is executed in the same manner. In the $R_2 \rightarrow R_1$ transfer, the $\Sigma/\div\times$ bit is 1, which causes the digit-resolver output to be zigzagged into R_1. As explained before, if clear/hold = 0, this places back into R_1 the sum of the number originally in R_1 and either the number in R_2 or plus or minus its absolute value. The case in which a shift is also performed will not be considered.

15-8. The Structure of the Main Control (Arithmetically Trivial Instructions). The arithmetically trivial (AT) instructions are those that do not call for any activity on the part of the arithmetic unit. They are (1) write, (2) unconditional transfer of control, (3) conditional transfer of control, (4) IBM load, and (5) drum $\rightarrow$ Williams memory. In all of these the NAT/AT bit is 0. NAT/AT may be 0 also in the IBM and drum-priming instructions. The treatment in the present section will be devoted to the first three instructions only; the IBM instructions will be dealt with in Chap. 16.

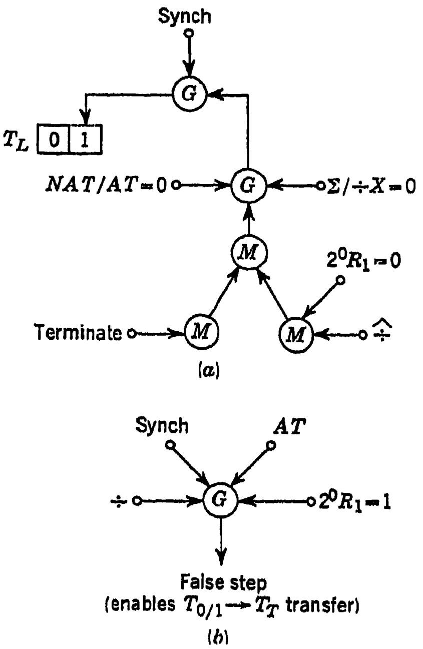


FIG. 15-5. Logic for arithmetically trivial instructions.

Certain extra circuitry is required for the (internal) AT instructions; this circuitry is shown in logical form in Fig. 15-5. In Fig. 15-2 it has been roughly indicated that, in transfer-control (TC) instructions, special means exist for turning T_L on even if T_F is off. These are shown in detail in Fig. 15-5a. Recall that, in the normal process, $T_F = 1$ causes the gating outputs G_2 and G_4 to assume their disabling levels, so that communication from R_3 to the synthesizer is cut off; this causes all inputs to the synthesizer to assume their 0 (high) levels; and thus, clearly, when $T_F = 1$, synch turns T_L on. On the other hand, if $T_F = 0$, as it must be while the control is causing the execution of an instruction, then synch can turn T_L on only if $NAT/AT = 0$ and $\Sigma/\div\times = 0$ and if at least one of the following conditions is fulfilled: (1) $2^0R_1 = 0$ (conditional transfer), (2) $\div$ is denied (unconditional transfer), (3) terminate is affirmed. In the TC instructions the condition $NAT/AT = 0$ and $\Sigma/\div\times = 0$ is always fulfilled, but not in the write instructions, and in them the circuitry shown plays no part. Figure 15-5 shows a gate which generates a "false step" during synch in case $\div$ is affirmed, $NAT/AT = 0$, and

$2^0R_1 = 1$; this functions only for unconditional transfer-of-control instructions, as will be shown below.

The simplest of the AT instructions is "write," that is, transfer the number standing in R_1 to the address in the memory specified by the address bits of the instruction. "Write" can take two forms, with clear/hold either 1 or 0, depending upon whether it is desired to clear R_1 or not; this is one of the instructions in which step/no step must in every case be 1. The composition is given in Table 15-1 and is repeated here for the hold case, as an example:

step/no step	= 1	$-L/+R$	= 0
int/ext	= 1	$\Sigma/\div\times$	= 1
Wms/arith	= 1	RO/NRO	= 0
NAT/AT	= 0	clear/hold	= 0
mag/no.	= 1	QS/NQS	= 0

From Fig. 15-1 it is evident that "write" is asserted. This signal plays no part in the main control; its chief function is to turn on the store toggle in the memory local control at the appropriate time and, thereby, to send a signal to the routine generators, which causes them to generate the voltages that cause the transfer from R_1 into the Williams memory to be carried out.

The role of the main control in this case is exceedingly simple. It sends on to the memory local control the yes signal, which causes the yes, synch, and action toggles to be turned on in sequence. The synch signal is fed back and produces one step in the counter. Because of the presence of AT, $T_{Y/O}$ resumes its 0 state immediately at the termination of synch. The termination of synch, therefore, also initiates "down" and causes the second necessary step in the counter, which is now in position to cause the control to proceed to the execution of the next instruction after the intervening regenerate cycle of the memory has been completed.

In the unconditional transfer of control, the address part of the instruction specifies the address from which the next instruction word is to be taken. The step/no step bit specifies the phase of the new instruction word to which control is to be transferred; 0 signifies the same phase (left or right) as that of the instruction under consideration, and 1 signifies the opposite phase. The other bits are as follows:

int/ext	= 1	$\Sigma/\div\times$	= 0
Wms/arith	= 1	RO/NRO	= 0
NAT/AT	= 0	clear/hold	= 0
mag/no.	= 1	QS/NQS	= 0
$-L/+R$	= 0		

Of course, in this case "write" is not asserted, which means that the memory control must execute a read cycle. The action begins with the transmission of "yes" and the consequent turning on of the yes, synch, and action toggles. "Up" is asserted only for the duration of synch, because $NAT/AT = 0$ guarantees that $T_{Y/O}$ is off after the expiration of synch. From Fig. 15-1 it is clear that $\div$ is denied, i.e., that $\widehat{\div}$ is asserted, because $\Sigma/\div\times = 0$ and $-L/+R = 0$; also $NAT/AT = 0$, so the gate by which synch must pass to turn T_L on is enabled. From Fig. 15-2 it follows that $T_L = 1$ denies WO. Synch also causes G_1 (or G_3) to assume its enabling level, with the result that the address is transmitted into the address generator. Notice also that, as soon as WO is denied, G_2 (or G_4) assumes its disabling value. Hence, by the time of "down," which coincides with the expiration of synch, the address from which the new instruction is to be obtained is in the address register, and the main control has completed its work.

Now consider the counter. First suppose that step/no step = 0. Then no transfer from $T_{0/1} \rightarrow T_T$ takes place during "up" (i.e., during synch), and, because T_L is on, "down" turns $T_{0/1}$ off. Hence, if $T_{0/1} = T_T = 0$ before "up," they are in the same states after "down." Similarly, if $T_{0/1} = T_T = 1$ before "up," then at "down" the same state of affairs obtains. After the $M \rightarrow R_3$ gate, which must be generated by the memory control because WO is denied, switches T_L off, the combination of "down" and $T_T = 1$ holds $T_{0/1} = 1$, and again no change has taken place. This means that, if the transfer instruction was the left (right) of the pair in R_3, then the control will next execute the left (right) of the pair in the new word brought into R_3. Suppose, on the other hand, that step/no step = 1. If at first $T_{0/1} = T_T = 0$, then $T_{0/1} = T_T = 1$ after $M \rightarrow R_3$ gate has turned T_L off, but if at first $T_{0/1} = T_1 = 1$, then $T_{0/1} = T_1 = 0$ afterward. That is, control is transferred from a left (right) instruction to the new right (left) instruction.

Finally, consider the conditional transfer of control, in which the transfer depends upon the sign of the number in R_1; transfer is accomplished if this sign is 0 (+), but, if it is 1 (−), the next instruction is obeyed. The step/no step bit has the same function as before, and the only change is that $-L/+R = 1$, so that in this case $\div$ is affirmed. Everything starts as in the unconditional case. According to Fig. 15-5a it is clear that synch is permitted to turn T_L on only if $2^0R_1 = 0$, since $\widehat{\div}$ is denied. If in fact $2^0R_1 = 0$, then everything proceeds exactly as in the unconditional case. If, on the other hand, $2^0R_1 = 1$, then no transfer must be made. Certainly T_L is not turned on. It must now be shown how the control is forced to the consideration of the next instruction in the ordinary sequence. If the step/no step bit is 1, no problem exists, for, clearly, up and down advance the counter in the desired way. If, however, the step/no step bit is 0, a false step is created by the gate of Fig. 15-5*b*. Since the conditions of $\div$, $NAT/AT = 0$, $2^0R_1 = 1$ are satisfied, synch causes the false step to be generated, and this is fed to the step input in Fig. 15-2; so the counter advances as desired and the control automatically passes on to the next instruction at the appropriate time.

15-9. Concluding Remarks on the IAS Main Control. Nothing has been said about the details of the quick sum or of the IBM instructions. The latter will be treated fully in the next chapter. The quick-sum instruction causes the sum of all the numbers from the address specified in the instruction up to 1023 to be formed (or the sum of their negatives, or of $\pm$ their absolute values); it requires a simple quick-sum control (which will not be shown) with, of course, certain interconnections with the main control. All details of this sort are omitted, and it will suffice to show merely how these circuits enforce the execution of the operation. QS control consists of two toggles and a few simple logic circuits. The QS instruction differs from an add/subtract instruction only in that $QS/NQS = 1$. It can stand only in the first (left) phase of a word; the second (right) phase must be an unconditional transfer of control. One toggle of the QS control is turned on during synch, the other during the subsequent $M \rightarrow R^3$ gate. These together supply signals to the memory and the main control, which prevent the advance of the control counter and cause the address of the first operand to be transferred to the order register of the dispatch counter (with, of course, 1 added to it).

Hence, after the regenerate cycle following the execution of the instruction, the counter is forced to cause it to be executed again, but this time the address of the operand is obtained from the order register of the dispatch counter and is, therefore, a number greater by 1 than the initial address. This process continues until the address reaches 1023; when 1 is added to this, the order register is reset to 0, and the overflow carry causes the QS control toggles to resume their 0 values. The control counter is also advanced; so that the next instruction obeyed is that in the right section of R_3. As the order register has been reset to 0, it must be supplied with address information, and this is done conveniently by making the right instruction an unconditional transfer of control.

This completes the account of the main control of the IAS machine. As such circuits go, it is simple and uncomplicated. It lacks certain features that tend to make coding simpler and more flexible, such as simple methods of modifying instructions. These can all be had at the price of more complicated and extensive circuitry. It will now be shown how these things can be accomplished.

15-10. The B Register. In the first University of Manchester digital computer, a special Williams tube called the B tube was provided[1] which

[1] T. Kilburn, G. C. Tootill, D. B. G. Edwards, and B. W. Pollard, Digital Com-

held but a single number, and provisions were made for adding this number to the address part of an instruction before it was executed and then causing the instruction (so modified) to be executed again; the coder could specify whether or not this was to be done by assigning a value to the so-called "*B* bit" of the instruction. When the second machine was built, additional facilities of this type were provided in a *B* tube holding eight numbers of 20 bits each. These can be used to modify instructions, as in the earlier machine, but they also have the function of counting: a

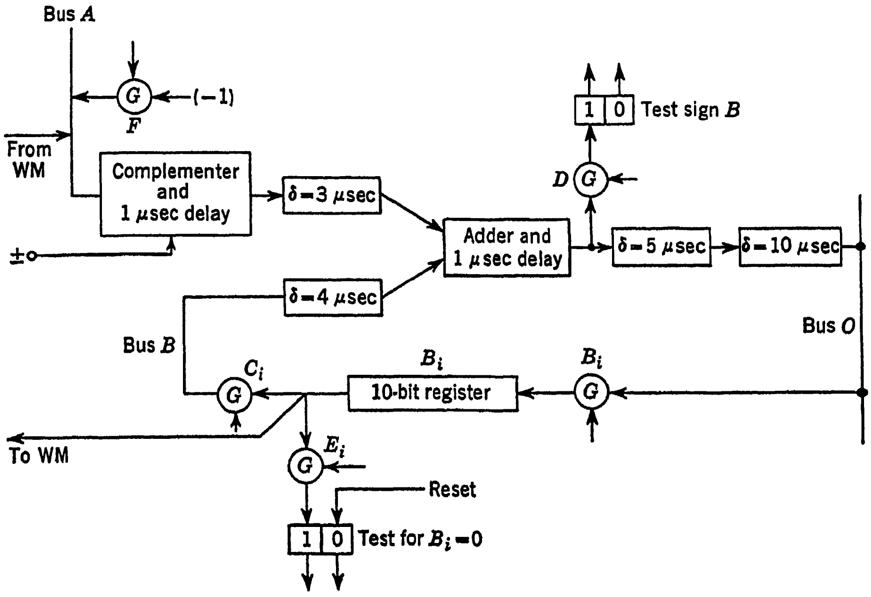


FIG. 15-6. MEG *B* registers. (*D. B. G. Edwards, The Design and Construction of an Experimental High Speed Digital Computer, Ph.D. thesis, University of Manchester,* 1954.)

number can be stored in an arbitrary line of the tube, representing the number of times an instruction is to be obeyed, and 1 can be subtracted from it at every step. The sign of this number is sensed at each step; so a means is provided for ascertaining when the desired number of iterations has been reached and for sending this information back to the control. Thus, transfers of control can be provided for in certain cases without using the accumulator. For example, if it is desired to add the numbers at consecutive memory addresses from n to $n + p$, 1 can be stored in one line of the tube and successively added to the address part of the sum instruction, which is successively reexecuted. In another line the integer p can be stored; 1 can be subtracted from p every time a single

puters at Manchester University, *Proc. IEE,* vol. 100, pt. 2, no. 77, pp. 487–500, October, 1953.

addition takes place. When the sign of the difference changes from + to −, that is, when the $p + 1$ summands have all been added together, this information is used to terminate the process and to transfer control to the next instruction.

In the most recent Manchester computer, the MEG, the same scheme has been retained, but the cathode-ray memory tube is replaced by eight circulating registers, each being 10 bits long. The general arrangement is shown in Fig. 15-6. The number in any register B_i can be read out by enabling the gate C_i and can be introduced as one input to the adder. To this may be added $\pm$ any number that appears at A, or from it 1 may be subtracted via gate F. The result may be read out via bus 0, or put back in B_i via gate B_i. The two toggles permit the determination of sign, for the gates D and E_i are opened only as the sign bit emerges from the adder or from B_i. As stated above, this sign-testing capability is used for counting. It is also used in programming the transfer of information between the high-speed memory and the magnetic drum.

15-11. A Magnetic-core-matrix Sequencing Chain.[1,2] All the control circuits described so far have been of the conventional vacuum-tube type. In closing this chapter, it is desired to call attention to the use of magnetic-core matrices in the generation of sequences of signals, such as, for example, the clear-gate sequences needed to operate a parallel arithmetic unit. Such a matrix is incorporated in the EDSAC II computer currently under development in the University Mathematical Laboratories at Cambridge, England.

The ultimate EDSAC II control matrix will be a 32×32 array of cores. To establish the principles of construction, the machine has been first operated with an 8×6 control matrix, which, naturally, restricted the number of operations that could be called for. For purposes of illustration, a simple 2×2 matrix will be shown in Fig. 15-7. Rather large ferrite cores (radius = 0.8 cm) are used, carrying as many as five windings each: two "drive," one "bias," one "output," and one "sequencing" winding. These are of 40, 40, 25, 3, and 3 turns, respectively. The "drive" currents are 150 ma, and the "bias" current is 300 ma. The bias flows at all times and is sufficient to hold the cores at negative saturation. The coincidence of two drive-current pulses overcomes the bias and drives the selected core into positive saturation, where it remains for the duration of the drive pulses, returning to negative saturation at their

[1] M. V. Wilkes and J. B. Stringer, Micro-programming and the Design of the Control Circuits in an Electronic Digital Computer, *Proc. Cambridge Phil. Soc.*, vol. 49, pt. 2, p. 230, 1953.

[2] M. V. Wilkes, W. Renwick, and D. J. Wheeler, The Design of the Control Unit of an Electronic Digital Computer, *IEE Tech. Paper* M2365, June, 1957 (to appear in *Proc. IEE*, vol. 105, pt. A).

expiration. The time interval between the switching of successive cores is 1.5 μsec.

The drive windings of the cores are connected in rows and columns, as shown. They are driven by driver tubes, which are driven in turn by those drive pulses which find gates enabled for them by the toggles T_x, T_y. Of the toggles T_x, one must hold a 1 and all others must hold 0's;

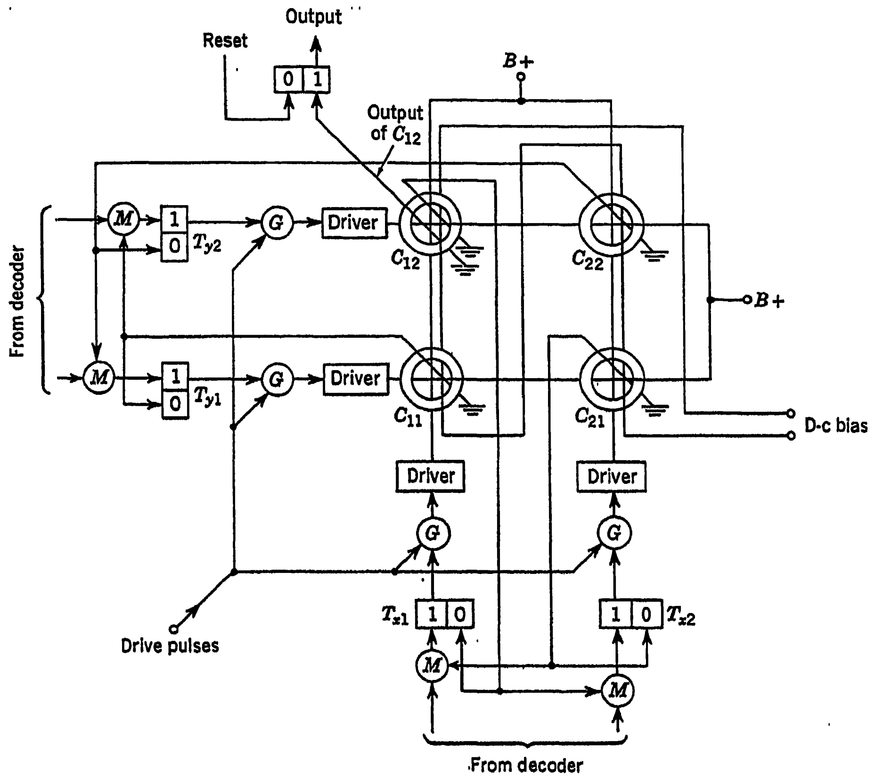


FIG. 15-7. Magnetic-core sequencing circuit (EDSAC II). (*M. V. Wilkes and J. B. Stringer, Micro-programming and the Design of the Control Circuits in an Electronic Digital Computer, Proc. Cambridge Phil. Soc., vol.* 49, *pt.* 2, *p.* 230, 1953.)

the same is true of the T_y. These toggles are set up by two decoders which operate on two sections of the operation bits of the instruction to be executed.

The outputs of the sequencing windings are used to switch the toggles T_x, T_y. The outputs of the output windings are the control pulses that cause the execution of the instruction, such as clear and gate commands, etc. Of these, the drawing shows the output of core C_{12} only; this sets up a toggle which can in turn control certain gating or clearing operations. The output could, of course, be amplified and clipped if only a short pulse were desired; the toggle can be used to lengthen and staticize it. It must be understood that the voltage that appears across a sequencing or output winding when its core is switched is of the correct polarity to cause the switching of the toggle only when the transition from positive to negative saturation is made.

With these principles in mind, suppose that T_{x1} and T_{y1} have been set to 1 and that T_{x2} and T_{y2} hold 0's. When a "drive" pulse enables the gates, the gates driven by T_{x1} and T_{y1} draw sufficient current to override the bias and to switch C_{11} to positive saturation. Upon the expiration of the "drive" pulse, the bias switches C_{11} back to negative saturation, thus producing across the sequencing winding a pulse of the proper magnitude and polarity to set T_{y1} to 0 and T_{y2} to 1. The second "drive" pulse,

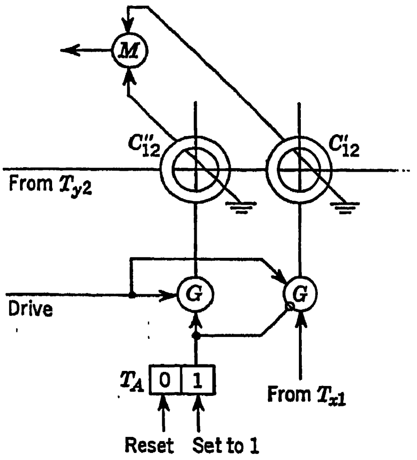


FIG. 15-8. Example of double-core arrangement for alternatives. (*M. V. Wilkes and J. B. Stringer, Micro-programming and the Design of the Control Circuits in an Electronic Digital Computer, Proc. Cambridge Phil. Soc., vol.* 49, *pt.* 2, *p.* 230, 1953.)

therefore, causes the switching of C_{12}, and, upon the expiration of the pulse, C_{12} returns to negative saturation and produces a sequencing pulse, which switches T_{x1} to 0 and T_{x2} to 1. The third "drive" pulse, therefore, causes the switching of C_{22} and the consequent switching of T_{y1} to 1 and T_{y2} to 0. Finally, at the expiration of the fourth "drive" pulse, T_{x1} is switched to 1 and T_{x2} to 0; so the whole circuit has returned to its initial state, and a sequence of outputs from the cores C_{11}, C_{12}, C_{22}, C_{21} has been produced.

In this simple illustration the sequence is inflexible, but measures can be introduced to produce alternatives. Recall that, in the IAS sequencing chain, the circuit does not issue a direct "clear" command, but rather a "clear invite," the particular clear actually performed being dictated by other parts of the arithmetic control. Information as to which clear is actually performed is fed back to the chain and determines which gate command is generated. One way of doing this with the core array is shown in Fig. 15-8. The single core is replaced by two cores C'_{12} and C''_{12}; both of these carry y-selecting windings driven by T_{y2}; and C'_{12} carries an x-selecting winding driven by T_{x1}, but the x-selecting winding of C''_{12} is driven by a separate toggle T_A, which either remains at 0 or is switched to 1 depending upon the use to which the output from C_{11} was put. If $T_A = 0$, the second "drive" pulse affects C'_{12}, and, if $T_A = 1$, then it affects C''_{12}. In both cases, given the mixing of sequencing outputs shown, T_{x1} is switched to 0 and T_{x2} to 1, so that C_{22} is the next core selected. If C_{21} is also replaced by a double-core arrangement of this type, the resulting matrix can clearly be used to generate a sequence: (1) clear invite, (2) either of two gates, (3) clear invite, (4) either of two gates.

Another possibility offered by the double-core arrangement is that of transferring control from one segment of a large matrix to another; obviously, a great variety of arrangements for using the sequencing outputs to set up the selecting toggles are available. The first core to be selected is specified in every case by the location of the single 1 in each of the two sets of selecting toggles. This information is supplied by the operation part of the instruction. From then on the sequence of pulses generated depends upon the way in which the sequencing outputs are fed back to the selecting toggles and upon information supplied by toggles such as T_A in Fig. 15-8.

An arrangement which accomplishes the same thing as the Cambridge matrix has been devised in the Max Planck Institut für Physik in Göttingen.[1] In this, the cores are arranged in sequences with branching and merging provided for so that alternative paths can be taken.

[1] H. Billing and W. Hopmann, Mikroprogramm-Steuerwerk, *Elektron. Rundschau*, vol. 10, pp. 349–353, October, 1955.

CHAPTER 16

INPUT AND OUTPUT

16-1. Introduction. The input-output equipment is the means by which the information to be processed and the appropriate instructions are introduced into the digital computer and by which the results of the processing are removed from it. A very wide range of devices is available to perform these functions. Which devices are used depends largely upon the uses to which the system is to be put. Thus, if the intended use is for mass data processing, as in various statistical operations such as the processing and tabulation of census data, and in various business applications, the input-output equipment must be capable of handling large quantities of data rapidly and efficiently and is, in a sense, the most important part of the data-processing system. The other extreme is represented by a computer to be used exclusively for mathematical computations, in which the amount of input data may be small (though the instruction code may be rather involved) and may be introduced once and for all at the beginning of the computation. The actual computation may then be exceedingly complicated, but, in the end, the amount of information required may be given by only a comparatively few numbers. In applications of this kind little is lost if the input-output equipment is rather primitive.

As the earlier digital computers were designed primarily for scientific calculations, it is not surprising that the principal effort was devoted to memory, control, and arithmetic functions and that existing Teletype equipment was adapted to supply the input and output. In the development by the computer industry of extensive systems for statistical- and business-data processing, however, the problem of devising high-performance input and output equipment was one of the most important and demanding that had to be faced, and many excellent solutions have been found. This chapter will not attempt to deal exhaustively with these proprietary devices, for a complete and detailed account of them would require a book rather than a chapter; rather, examples will be given of the types of equipment in use and their characteristics.

Important as are the special devices required for input and output, the question of how they are interconnected with the rest of the system is still more important. Certain general principles will be discussed, and some details of the methods used in the SEAC and IAS computers will be given.

It is important to realize at the start that many input-output devices can be utilized also as auxiliary memories. Already in Chap. 14, the use of a magnetic drum as auxiliary to a short-access-time magnetic-core memory has been considered in some detail. Magnetic-tape-handling devices are also frequently used for this purpose.

Information to be introduced into a computer is ordinarily first recorded upon punched paper tape, punched cards, magnetic wire, or magnetic tape. This can and, indeed, must be recorded originally at the speed at which a person can operate a typewriter keyboard. The medium is then read by an appropriate sensing device at a much higher speed, and the output is fed into the computer memory. Many machines are provided also with a keyboard by which one word at a time can be read into the machine; this adjustment is useful for testing and checking purposes.

As to output, this can be direct to some sort of electrically driven printing device or indirect, in the sense that the information is first recorded upon the same sort of medium used to carry the input information, then read from this medium by a separate device, which in turn drives a printer. It is also possible, with high-speed machines, to display the output visually on a cathode-ray tube.

16-2. Preparation and Verification of the Input Medium. Once the detailed code of a problem has been worked out, all numerical data have been assembled, and memory addresses have been assigned to all items, the next step is to record this information in order on the input medium. This operation is performed by a human operator who must read the information and operate some sort of keyboard that actuates the recording mechanism, and is, accordingly, subject to all manner of errors. Consequently, it is very important that the recorded information be verified before it is finally introduced into the computer.

Brief mention will be made of how the verification is performed in the numerical-tape-preparation unit of the Harvard Mark III computer.[1] This unit incorporates a keyboard having one column of keys 0 to 9 for each of the 16 decimal places of the numbers to be transcribed plus a column of two keys for the sign; relay circuits for temporary memory; electronic logic circuits for making comparisons; amplifiers to drive the magnetic writing heads; and two reels with associated pulleys and driving motors to carry the magnetic tape. How the information is written on

[1] By the Staff, Harvard Computation Laboratory, "Description of a Magnetic Drum Calculator," Harvard University Press, Cambridge, Mass., 1952. See pp. 142–151 for a complete description of this unit.

the magnetic tape is unimportant for this discussion; each decimal digit is, in fact, recorded as 4 bits in the 2*421 code. When the operator has punched in a number, he pushes a "record" push button, which energizes circuits that check whether some key has been pushed in every column. If the check is not satisfied, an indicator lamp is lighted. If it is satisfied, then the number standing in the keyboard is stored in a relay register, and the keys are reset. It is then necessary for the operator to punch the same number into the keyboard a second time and then to depress the record push button again. If the number now standing in the keyboard is identical with that in the relay register, it is recorded upon the magnetic tape and the keys are reset. If the check fails, a second indicator lamp is lighted, and the operator must take corrective action. In principle this is a very simple and practical method of eliminating error; of course, if the operator makes the same mistake twice in entering the number on the keyboard, it will pass unnoticed, but errors of this kind could be eliminated only if the human operator were replaced by a device that read the written information and encoded it automatically. The details of the circuit of the unit will not be given here, for they are available in the reference cited.

16-3. Punched Tape and Cards. Punched paper tape is the simplest input medium and has been (and is) used quite extensively. When the early digital computers were under development, Teletype tape-perforating and reading equipment had long been in existence and had been developed to a high degree of perfection. Besides cheapness and availability, punched paper tape also offered the advantages of reliability under a rather wide range of operating conditions (it is not, for example, troubled by the presence of moderate amounts of dust, nor by reasonable variations in temperature) and of relatively simple synchronization, since it is normally driven by means of a sprocket that engages small holes punched as data are entered. The disadvantages of punched paper tape are that it is subject to wear, that its information-packing density is low, and that in many respects it is not well adapted to high-speed operation. In cases where these adverse considerations are unimportant, punched paper tape is a cheap, simple, and reliable input medium.

The preparation of a paper tape is, of course, a slow manual process. The process of reading for transmission to the computer can be made fairly rapid and wear to the tape reduced, if reading is accomplished photoelectrically. A reader developed by Ferranti Ltd. has been very popular; the remarks made here apply to the Mark II model. This device reads either five- or seven-hole standard Teletype tape. No sprocket is used; the tape is fed through the reader by a friction drive. The driving roller is part of a differential gear, the cage of which is driven continuously by an induction motor. The roller is started or stopped by applying electromagnetically operated brakes to one or the other of the two output shafts of the differential. In the control circuit usually used, the brake coils are connected each in one of the output circuits of a toggle, so that at all times one brake is on and the other off. The brake shoes are always lightly held against the brake drum and applied very rapidly.

As the inertia of the moving parts of the system is small, the tape can be accelerated quite rapidly; within 5 msec it can be brought from rest to a maximum speed of 20 in. per sec, or a reading rate of 200 characters per second.

Reading is accomplished by a set of photocells, one photocell for each track; the light source is a single 36-watt bulb. No lens system is needed. Each photocell drives a simple single-tube amplifier, whose output is more negative than -20 volts when the light is interrupted but rises to 0 volts when the photocell is excited by light shining through a hole.

The feeding of the tape is also controlled by signals derived from a photocell activated by light shining through the sprocket holes. The output of an amplifier driven by this photocell is squared and thus formed into a train of timing pulses that can be used to synchronize the operation of the reading circuits.

The Ferranti reader is a quite satisfactory input device. No output device of comparable speed exists; in speed for output, punched paper tape is at a serious disadvantage.

Punched-card equipment (IBM, Remington-Rand, etc.) is also very useful for input and output; it also offers the advantages of development to a high state of perfection and ready availability. No attempt will be made to describe these card-preparation devices, which should be well known, but attention will be called to the IBM 514 reproducing punch, which is used as the input-output mechanism of the Institute for Advanced Study (IAS) computer. This machine can both read and punch cards, and can combine these operations to produce a duplicate of a deck of cards fed in. The card-handling rate is 100 per minute. As each card has $12 \times 80 = 960$ punching positions, each of which can be used to represent 1 bit of information, the information rate is 1,600 bits per sec, or, if 40-bit words are used, 40 words per second. As will be seen, as a matter of convenience the IAS system uses only half this capacity, which still gives an information rate comparable, for example, to that of the Ferranti tape reader. The flexibility of the 514, in that it serves as both input and output device, together with its reasonably high speed, makes it very useful for these purposes.

16-4. Magnetic Wire and Tape. The media considered so far lack the property of erasability and, hence, are of rather limited usefulness for auxiliary memory. Magnetic wire and tape are, of course, erasable. This means that, though their initial cost is relatively high, they can be used over and over again and, also, that they are suitable media both for input-output and for auxiliary-memory devices. Another desirable feature is the high packing density of information, of which more will be said below. For all these reasons, a great deal of effort has been expended upon the development of magnetic-tape-handling devices, the chief problems being in the design of the tape-driving mechanism. If the device is to be useful as an auxiliary memory, the drive must be capable of starting and stopping rapidly and of running in both directions. On the other hand, a device used only to load the computer does not require a very elaborate drive, provided it is reasonably fast.

Magnetic-wire devices have not been used a great deal, presumably because they are inherently capable of carrying but a single information channel. A simple magnetic-wire input has been successfully used at the National Bureau of Standards (NBS) with both the SEAC and DYSEAC computers.[1] An office dictating machine was modified by increasing the speed of wire drive and by replacing the ordinary stainless-steel wire by plated wire more satisfactory in packing density. In this equipment, the wire is contained in cartridges which can, of course, be changed as desired; a useful feature in the original office equipment is an indicator on each cartridge that shows how much of the wire has been transferred from one of the two spools to the other. The packing density of information is not particularly great, but the 1,800 ft of wire carries 14,000 words, and, as the wire moves at 8 ft per sec and the SEAC words are 45 bits in length, the information rate is about 70 words per second. Starting is not fast; 1 sec is required to bring the wire up to full speed. For this reason, there must be adequate gaps in the recorded information to allow for starting and stopping. The unit is most efficiently used when there are to be few gaps in information; it is very efficient for loading the computer, for there the uninterrupted runs of information are long.

In the SEAC system, first a paper tape is prepared and verified, then this is read and the information is transcribed to a wire, and then the wire is read into the memory of the machine. Results of computation are removed from SEAC also by means of the wire-handling device. The wire is then read at low speed (1 in. per sec) and the information is transcribed onto a punched paper tape by a device called the "outscriber." It may be wondered why the paper tape is used at all; this is done because the slow drive does not start and stop quickly and, therefore, if it were driving a printer directly, large gaps between words would have to be left on the wire to permit carriage return or else a large memory unit would be required. Both of these undesirable features are very easily avoided by the simple expedient of transferring the information to the paper tape and using this to drive the standard Teletype printing equipment.

[1] J. L. Pike and E. F. Ainsworth, Input-Output Device for NBS Computer, *NBS Circ.* 551, pp. 109–118, Jan. 25, 1955.

Compared with wire, magnetic tape offers the very considerable advantage that it can carry a number of parallel channels, of which one or more can be reserved for timing purposes. Magnetic-tape-handling units for input, output, and auxiliary memory have received intensive development, and a number of good ones are now on the market. One of the first successful input-output units was built in the Harvard Computation Laboratory for use with the Mark III calculator, which in fact uses four of these units. A very complete description is given in the "Description of a Magnetic Drum Calculator," to which the reader is referred.

Another early successful unit[1] was developed by J. L. Pike at NBS for use as an auxiliary memory for SEAC. It is probably the simplest device of the kind ever made, involving no tape reels or servomechanisms. The driving mechanism consists of two capstans continuously rotating in opposite senses, over which the tape hangs loosely. Between these are located the heads. Each capstan is provided with a jam roller consisting of a ball bearing fitted with a nylon tire, which can be moved into contact with the capstan by means of a solenoid, thus pressing the tape against the capstan and causing it to be moved. The two ends of the tape fall in loose folds into narrow glass tanks, each formed by spacing two glass plates just a little farther apart than the width of the tape.[2] The tape has no tendency to turn over or to become tangled. Some difficulty was encountered because the tape tended to pick up electrostatic charge as it passed through the drive mechanism, but this was eliminated by mounting near the drive unit strips of alpha-emitting polonium which cause the air to be ionized and the charge, therefore, to be canceled; another method is to use a tape that has an evaporated aluminum film on its back surface, so that the charge cannot collect.

Tapes up to 3,600 ft long can be accommodated by the tanks (19 in. by 3 ft). The packing density is 110 bits per in. The tape speed generally used is 5 ft per sec; greater speeds could be used if desired.

The absence of reels, which makes for simplicity in the design, prevents the unit from being convenient as an input or output device, for changing tapes is, obviously, a rather inconvenient operation.

The Harvard unit, on the other hand, was a versatile input-output device in which the tape was held on easily changed reels. The original units were rather slow, but a greatly improved model was soon devised in which quite high speed was attained by using an ingenious pneumatic clutch in place of the magnetic clutch previously employed.[3] The principle is very simple. The tape passes over a continuously running capstan in which are drilled holes that communicate through the shaft

[1] This is described in *NBS Tech. News Bull.*, November, 1951.

[2] Hence units of this type are said to work on the "wastepaper-basket" principle.

[3] For details of the system, see *Harvard Computation Lab. Progress Repts.* 8, sec. 6, 1949, and 10, sec. 13, 1950, written under contracts W19-122ac24 and AF-33(038)-9401.

via a valve to a chamber in which a moderate vacuum is maintained. When the valve is opened, air is sucked through the holes in the capstan and the atmospheric pressure applied to the outward surface of the tape firmly engages it to the capstan, so that it is driven in the direction of rotation. To stop the tape, the valve is closed and a pneumatic brake is applied. There are also two tape reels, which merely pay out and take up the tape, and have nothing whatever to do with starting or stopping its motion. This simple and elegant mechanism gives very good stopping and starting characteristics: to bring the tape from rest to a velocity of 100 in. per sec requires about 3 msec from the instant that the initiating pulse is received, and a similar period suffices to bring it from full speed to rest. It is possible to stop or start the tape as frequently as 150 times per second.

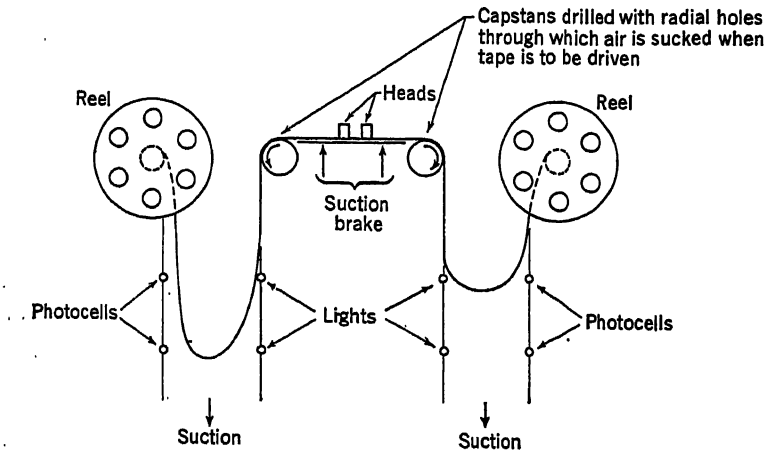

Capstans drilled with radial holes through which air is sucked when tape is to be driven
Reel
Heads
Reel
Suction brake
Photocells
Lights
Photocells
Suction
Suction

FIG. 16-1. Essentials of Cambridge modification of Harvard tape unit.

Directly based upon the Harvard design, though modified in a few details, is the magnetic-tape unit developed in the University Mathematical Laboratory at Cambridge, England, which is shown in simplified form in the drawing of Fig. 16-1. In the developmental model, iron oxide–coated plastic tape ½ in. wide is used which can carry six tracks if necessary. In EDSAC I, where it functions as an auxiliary memory, only two tracks are used, one for information and one for timing pulses, but when EDSAC II is completed it is planned to use all available tracks. The total length of the tape is 1,200 ft, the packing density is 100 bits per in., and the speed is 100 in. per sec, as in the Harvard unit.

As in most tape units, the information is split into a number of blocks of words. This is done very simply when a tape is prepared, by scraping off the coating in narrow bands running across the tape; the distance between block markers can be anything from the distance required for one word on up. When the scraping has been done, the tape is run through the device, and the computer writes immediately after each block marker the number of the block (this, of course, implies that a forward sense has been assigned). When this operation has been completed the tape is ready to receive information from the computer.

To read a block of information from the tape unit into the main high-speed memory, first the tape is driven forward until a block marker is encountered, and the block number immediately following it is read. Then this is compared with the number of the desired block, and the tape is driven forward or back as required. The number of block markers passing the photoelectric cell is counted, and this number is added to or subtracted from the starting-block number to indicate when the desired block is reached. When this has been found, it must of course be read with the tape moving in the forward direction. In the EDSAC I installation, the block-number arithmetic is performed in the accumulator, but special facilities for this could be provided.

Returning to the mechanical details, the capstan and pneumatic braking schemes and the suction applied to the bins in which the tape loops hang are essentially the same as in the Harvard unit. There is a difference in the photoelectric sensing, which here uses two light sources and photocells in each bin. When the capstan starts to drive the tape forward, the take-up reel remains at rest until the light to both photocells is cut off. The brake on the take-up motor is then released, and the tape is reeled in until outputs are obtained from both photocells; this causes the brake to be applied. If both photocells remain excited for over 2 sec, this is taken as an indication that the brake has failed to operate, and power is removed from the unit. Similarly, the pay-out reel is driven forward if both photocells are excited and is stopped when neither is. The sense of rotation of both reels is the same as that of the capstans.

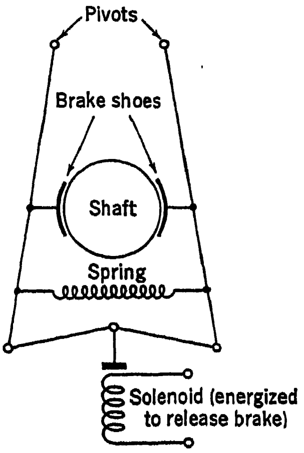


FIG. 16-2. Cambridge braking mechanism.

The capstan and reels are driven by split-field servomotors which develop a torque of 56 oz-in. under static conditions and half that at 5,000 rpm, the field excitation being 100 ma. The tape reaches a full speed of 100 in. per sec in about 15 msec and can be brought to rest in approximately the same time, as compared with about 5 msec for the original Harvard units.

The braking system is very simple; it is shown in principle in Fig. 16-2. The brake shoes are mounted on two pivoted arms, which are drawn together by the spring and forced apart by the linkage when the iron plunger is drawn down by the solenoid. Thus, when the solenoid is energized, the shoes are removed from contact with the drum mounted on the motor shaft, and deenergization permits the spring to force the shoes into contact with the drum. This is an exceedingly simple device that has proved to be quite satisfactory in operation.

Good magnetic-tape units have been developed by several manufacturers. They differ in many ways: in the method of driving the tape, in the number of information tracks, in the packing density of information, in starting and stopping, and so forth. No attempt will be made to describe these devices,[1] which are used variously for input, output, and auxiliary memory.

Magnetic tape is not an ideal material, for certain surface defects prevent the satisfactory recording and recovery of information. Small bumps occur, which may be due either to nodules of magnetic material or to foreign matter that gets into the magnetic material before or during the coating process. These have been measured and found to average about 0.012 in. in length and about 0.0005 in. in height.[2] At a density of 100 pulses per inch, such a defect might be expected to affect only one or two pulse positions. However, as the tape is in contact with the heads, the bump forces it away from the head, and the separation causes deterioration of the signals obtained from adjacent positions. Some success has been reported in attempts to remove such bumps by scrubbing with steel wool[3] or by drawing the tape over a knife edge.[4] Trouble is also caused by small pinholes and tiny cracks in the magnetic coating. It should be appreciated that such defects cause no trouble in ordinary audio work, for which magnetic tape was originally intended, but became quite troublesome when the first attempts were made to use the medium for high-density digital recording.

The manufacturers have greatly improved the quality of their product in response to the more rigid requirements, but it will be appreciated that it is next to impossible to guarantee complete freedom from defects in a reel of a thousand feet or more of tape. For this reason it is necessary to examine each reel of tape carefully before information is recorded on it and, in some way, to mark the bad spots so that they can be avoided.

This can be done in a variety of ways; as an illustration a method found to be useful at NBS will be cited.[4] In principle this is very simple: the

[1] Descriptions of several of the earlier devices were given in Review of Input and Output Equipment Used in Computing Systems, *Proc. Joint ACM-IRE-AIEE Computer Conf., 1952*, American Institute of Electrical Engineers, New York, March, 1953.

[2] K. M. Rehler, The RAYDAC System and Its External Memory, in *ibid.*, pp. 63–70.

[3] *Ibid.*

[4] Pike and Ainsworth, *op. cit.*

timing or synchronizing track is permitted to contain a bit only if it is possible to write and read successfully in the corresponding locations in all the other tracks. One way to accomplish this is as follows. First attempt to record at full density on one track. Then read from that track and write on the adjacent track only if the signal read exceeds some minimum acceptable level. This guarantees that the second track contains bits only at locations corresponding to good locations in the first track. The next step is to read from the second track and to write on the third track only if the signal read exceeds the minimum acceptable level. Continue this process until the last track has been written on. Then erase the first track, read the last track, and write on the first track only in those locations where the signal is acceptable. This guarantees that the first track finally contains bits only in those locations where successful operation is obtained all the way across the tape.

In the examples of tape-handling units cited above, the packing density is 100 bits per in. That this can be considerably increased has been demonstrated in experiments[1] conducted by J. R. Sorrells of NBS, who succeeded in obtaining densities up to 700 bits per in. The advantages of reliable operation at such high densities are so obvious that there is no need to elaborate upon them.

Information is recorded in a modified non-return-to-zero system. The recording currents are actually pulses, both polarities being used in the following way: if 1 is to be recorded, the polarity is opposite to that used for the last bit; if 0 is to be recorded, the polarity is the same as that used for the last bit. Given the high density and the fact that the region of magnetization is longer than the gap in the writing head, it is easy to see that successive 0's overlap and produce a region of saturation just as if a constant-current non-return-to-zero scheme were used and that from this no signals are recovered upon playback. On the other hand, each 1 is represented magnetically by a single transition from saturation in one direction to saturation in the opposite and, hence, produces a single output pulse upon playback. The only disadvantage is that successive 1's yield outputs of opposite polarities, but this is easily enough handled by the reading circuits.

In this system, one channel is (as usual) reserved for the timing (or sprocket) information; if the words are n bits long, the sprocket channel must provide n timing pulses plus an additional pulse used to set up a reference condition at the beginning of the word. The sprocket information is followed by a string of 0's, only a few between words within a block, but sufficient after the last word of a block to permit starting or stopping before the next word arrives under the heads.

In recording, the sprocket channel is read, and the output is used to

[1] *NBS Tech. News Bull.*, vol. 39, no. 9, pp. 121–124, September, 1955.

gate standard 2-μsec pulses into the writing drivers; a logical diagram of the circuit is shown in Fig. 16-3. The coincidence of such a standardized sprocket pulse and a 1 causes the toggle, which is connected as a scaler, to switch, and thus guarantees that the polarity of the magnetization produced upon the tape is the opposite of that produced by the preceding (delayed) sprocket pulse. On the other hand, if no 1 is received, the toggle is not switched, and the sprocket pulse produces a spot of magnetization of the same polarity as the one before.

Simultaneous reading of the sprocket channel and writing on the information channels present a serious problem in that, since the heads are mounted in close proximity, the writing head induces a large signal in the reading head. This cross talk was reduced by several countermeasures: (1) carefully shielding the sprocket-channel reading head; (2) mounting the information-track heads so that they produced fields in opposite senses, which largely canceled in their net effect; (3) exploiting the fact that the sprocket-channel output for high-density recording is

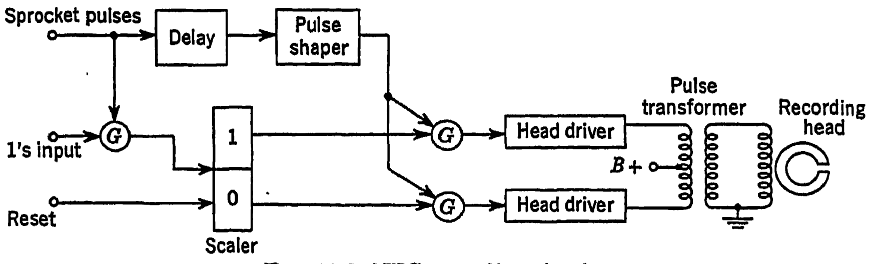


FIG. 16-3. NBS recording circuit.

nearly sinusoidal and of low frequency, whereas the writing signal, being a 2-μsec pulse, contains mostly high-frequency components, by using a low-pass sprocket-channel reading amplifier.

The results of the NBS experiments were very gratifying. As the number of bits per inch was increased, the reading-amplifier output increased slowly, reaching a maximum for a density of 440 bits per in., and thereafter decreased slowly, still being of usable magnitude at 735 bits per in. Thus, if the tape were driven at 100 in. per sec, a high-speed internal memory capable of holding one thousand 40-bit words could be loaded or emptied in less than 1 sec.

16-5. Various Auxiliary Memory Devices. In the performance of large-scale computations, almost inevitably, the high-speed memory of the computer has insufficient capacity to hold all the relevant information. The computation may be such that the numerical data is required only in small blocks; in this case it suffices to bring these blocks in from the input unit as they are required—this is certainly adequate with many applications of the data-processing type. On the other hand, in other applications, a large volume of intermediate results may be generated from time to time which must be used in a later stage of the computation; take as an example the process of computing the inverse of a large matrix by an iterative process. In such applications it is necessary to have an auxiliary memory to which information beyond the capacity of the high-speed memory can be transferred and where it can be held until needed.

In the description of the memory of the ORDVAC, it was shown how a magnetic drum is used as an auxiliary to a short-access-time magnetic-core memory, and some such arrangement is, in fact, used in most high-speed computers. Another possibility is to use a magnetic-tape unit for this purpose. This has the advantage of large information capacity: a 1,200-ft single-channel tape with a packing density of 100 bits per in. holds 28,800 words, assuming that 50 bits suffices for each word and the space following it—this is much more than the capacity of most magnetic drums; five- and six-channel tapes hold proportionately more, and wide tapes capable of carrying, for example, 40 channels have a huge capacity.[1] However, a serious disadvantage is inherent in all tape devices: in order to locate an item or a block of items for reintroduction into the high-speed memory, it may be necessary to run through the entire tape, which, even at a speed of 100 in. per sec, requires over 2 min for a 1,200-ft tape.

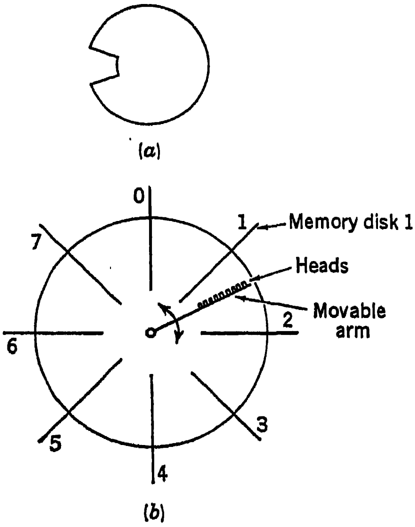


FIG. 16-4. (*a*) NBS notched memory disk; (*b*) mounting of disks and heads.

One way of achieving very large capacity with an access time of the order of 0.5 sec was devised at NBS by J. Rabinow.[2] The memory medium consists of a large number of large aluminum disks (20 in. in diameter) coated on both sides with magnetic material, each of which carries a deep notch, as shown in Fig. 16-4*a*. These are mounted upon a circular shaft, so that the whole memory array is toroidal in shape; Fig. 16-4*b* shows enough of the array so that the principle of arrangement can

[1] L. Merrill, 40 Channel Magnetic Tape Auxiliary Memory, *Proc. Symposium on Large Scale Digital Computing Machinery*, pp. 219–227, Argonne National Laboratory, Aug. 3–5, 1953.

[2] J. Rabinow, The Notched-disk Memory, *Elec. Eng.*, vol. 71, no. 8, pp. 745–749, August, 1952.

be understood. The circular shaft is mounted, e.g., in a horizontal plane, in which another shaft rotates an arm carrying a double array of thin magnetic heads (thickness 0.073 in.), as shown in Fig. 16-4*b*. For reading from or writing on any disk, the arm is first rotated until the plane of the arm and that of the disk coincide—this is possible because of the slots in the disks—and then the disk is rapidly rotated, its surface in contact with the heads as it passes through the slot.

An idea of the capacity of such a memory can be formed by a quick calculation. Assume that the usable part of the surface of each disk extends from 5 in. from the center to the periphery, that there are 12 channels per inch, and that the packing density on each track is 100 bits per in. This results in a capacity of about 566,000 bits per disk, for both sides are used, and it is seen that a toroidal array of 500 disks would have a capacity of some 7 million 40-bit words.

Space does not suffice to go into the mechanical details of this device; it is cited to show that really enormous memory capacities are in fact possible, though a long access time may perhaps be the price.

Other large memories are of a fixed character, with the information recorded upon photographic media and, therefore, not alterable. It has been pointed out[1] that the resolving power of the best photographic emulsions is greater than 10^3 marks per millimeter and that, therefore, a packing density of 10^6 bits per square millimeter of film surface could be attained if a scanning device of comparable resolution were available.

Actual flying-spot scanners (such as the kinescope) can be made to produce a spot of diameter as small as 0.025 mm,* which sets a packing-density upper limit of 1,600 bits per sq mm, which is large, but still very small in comparison with the limit imposed by the resolving power of the medium. Even with this limitation, a rather small surface would be capable of holding 10 million bits or more; it is of course to be expected that a practical device would succeed in realizing only a fraction of the limiting density but, even then, the density would be far in excess of that obtainable on, e.g., magnetic media.

Information might be recorded on the medium by preparing large-scale drawings of the array of coded information, photographing them, and then effecting a reduction in size. This is a tedious process, which would be worthwhile only if the recorded information were to be useful without change for a fairly long time.

The information could be recorded as marks on a transparent plate, and reading could be accomplished by scanning with a flying spot and

[1] G. W. King, G. W. Brown, and L. N. Ridenour, Photographic Techniques for Information Storage, *Proc. IRE*, vol. 41, no. 10, pp. 1421–1428, October, 1953.

* F. Roberts and J. Z. Young, The Flying Spot Microscope, *Proc. IEE*, ser. IIIA, vol. 99, no. 20, p. 750, 1952.

picking up the transmitted light by means of a photomultiplier tube. The medium might be in the form of a rotating disk,[1] which has the advantage that, if a radial scan is used, it is not necessary to move the scanning beam, or in the form of a rectangular slide.[2] In the latter case, beam deflection by the simple means used in Williams memories would not suffice, for the step from line to line is so small, but it has been proposed that a coordinate grid be recorded along with the information and used as a means of positioning the beam. Another possibility is to use a cylindrical storage surface, scanned as it rotates by a flying-spot scanner.

A variety of commercial large-scale auxiliary memories are on the market. Generally speaking, the objective is to provide a memory of the scale provided by tape devices but having the additional merit of shorter access time. One way of doing this is exemplified in the Burroughs Datafile,[3] which consists essentially of a large number of pieces of tape, each 250 ft long. These are mounted in parallel and are all brought out over a guiding system and driving rollers. Each individual tape works on the "wastepaper-basket" principle, the individual bins being provided by partitions in a long box. When any individual tape is selected, it alone is driven and the others remain stationary. A single head assembly suffices; it is mounted on a carriage accurately positioned by a servomechanism to read the selected tape. The carriage also carries pinch rollers which press the selected tape against the drive rollers. An early version of the Datafile incorporated 50 tapes and was capable of holding 22×10^6 digits with an average access time of 15 sec.

Another scheme,[4] developed by the Clevite Corporation and called Tape-Drum, is to use a very wide magnetic tape which passes over a drum. The drum carries the heads, and, during any reading or writing operation, the tape is motionless. Friction between tape and drum is prevented by a thin boundary layer of air about 0.0002 in. thick. The relative speed of the magnetic material and the heads is thus faster than in conventional tape units, for the drum, which is 12 in. in diameter, turns at 1,200 rpm. The tape is divided into "pages," one page being the length of tape that can be scanned at a time. In the model described, each page held 250,000 bits, the length could be up to 400 pages, and several pages could be scanned per second.

[1] King, Brown, and Ridenour, *op. cit.*

[2] R. D. Ryan, A Permanent High Speed Store for Use with Digital Computers, *IRE Trans. on Electronic Computers*, vol. EC-3, no. 3, pp. 2–5, September, 1954.

[3] D. N. MacDonald, Datafile: A New Tool for Extensive File Storage, *Proc. Eastern Joint Computer Conf., 1956*, pp. 124–127, American Institute of Electrical Engineers, New York, 1957.

[4] G. L. Hollander, Quasi-static Access Memory Systems, *Proc. Eastern Joint Computer Conf., 1956*, pp. 128–135.

The IBM Corporation has developed the type 350 magnetic-disk random-access file,[1] in principle a large number of magnetic disks mounted on one drive shaft and read by a movable head assembly. An attractive feature of this arrangement is its low access time, varying from 0.15 sec minimum to 0.80 sec maximum.

Finally, a more conventional large auxiliary memory devised in the Sperry Rand Corporation consists of one or more magnetic drums, each one provided with a movable head assembly. The technique of movable heads has already been applied successfully in England at the National Physical Laboratory (in the ACE computer) and by the English Electric Company (in the DEUCE computer). The Sperry Rand drum is of greater capacity than these earlier ones, however; it is capable of holding 250,000 words of 12 decimal digits each. The maximum access time, which is, of course, determined largely by the time required to move the head, is 2.6 sec, and the average is 1 sec.*

The subject of large-scale auxiliary memories will not be pursued further. It is worth noting that the chief field of application of these devices is not scientific calculation but mass data processing and that, in fact, operations of this character really require memories of very large capacity. This is especially true of operations in which all the data presently kept in a large file must be made easily accessible to the data-processing system.

16-6. Displaying the Results. The final step in data processing is to put the results in a form easily understood by human beings. Something of this kind must be done in all cases; even when the computing device is an element in an automatic control system, some sort of monitoring by a human operator is presumably necessary. The most common procedure is to print the results, but it is possible also to display them visually in various ways.

A great variety of printing devices exist.[2] The slowest and simplest are the single-action printers, which are essentially typewriters controlled by solenoids. These are limited in speed to about ten characters per second. Line-at-a-time mechanical printers are faster; these operate at one to two lines per second. The more recent on-the-fly printers are about ten times faster. All these devices use ordinary type faces and are limited in speed by the difficulty of imparting motion to the relatively heavy mechanical elements and by the time required to move out of the

[1] M. L. Lesser and J. W. Haanstra, The RAMAC Data-processing Machine: System Organization of the IBM 305, *Proc. Eastern Joint Computer Conf., 1956*, pp. 139–146.

* H. F. Welsh and V. J. Porter, A Large Capacity Drum-file Memory System, *Proc. Eastern Joint Computer Conf., 1956*, pp. 136–139.

[2] J. C. Hosken, Survey of Mechanical Printers, in "Review of Input and Output Equipment Used in Computing Systems," pp. 106–112, American Institute of Electrical Engineers, New York, March, 1953.

way the type slug needed to print one letter before the next slug can be positioned.

To overcome these difficulties, matrix printers were developed. These are still mechanical, but the inertia of the moving element is less. The central idea is very simple. The marks are produced by a selected set of small pins from an array of, say, 35, driven by small hammers. Thus the characters are arrays of small dots chosen from a small 7 × 5 matrix and are perfectly intelligible, though of lower quality than those produced by standard type. Matrix printers can be made to operate at several hundred characters per second, which is about the same speed as that of the on-the-fly printers.

To get to much higher speeds it seems obvious that means other than mechanical must be employed.[1] A considerable variety of methods have been developed. Characters may be formed directly on Western Union type L Teledeltos paper by sparking properly selected small electrodes from a matrix through the paper (Atomic Instrument Company). Characters may be formed as patterns of magnetization on a magnetizable medium (e.g., tape), then made visible by dusting with a powdered magnetic material, and printed finally by bringing the tape in contact with a paper tape treated with an adhesive material. This process (called "ferromagnetography") was developed by the General Electric Corporation. Other methods depend upon displaying the characters on a cathode-ray tube (CRT) and recording the picture photographically or by some other process. The earliest method of forming characters in this way was developed in the Harvard Computation Laboratory long ago; it depends upon the generation of appropriate voltages which were applied to the vertical and horizontal plates of a CRT, and the resulting device was called the "numeroscope." The same name has been used for a rather different device, developed by the Engineering Research Associates disivion of the Sperry Rand Corporation (ERA), in which these sets of voltages are worked out in advance and recorded on magnetic drums. For a permanent record the characters displayed on the CRT are photographed. Another approach is represented by the magnetic-matrix printer of the Laboratory for Electronics, Inc., which builds up the characters from arrays of dots on the CRT, the selection of the dots being accomplished by a magnetic-core matrix. Photographing the displayed characters gives a permanent record. Finally, the Charactron is a special tube in which the electron beam passes through a set of character-selecting plates, a beam-shaping stencil plate, and a final set of deflection plates before reaching the fluorescent screen. In accordance with the coded input information, the character-selecting plates direct the beam to the appropriate position in the stencil plate, from which it emerges

[1] R. J. Rossheim, Non-mechanical High Speed Printers, in *ibid.*, pp. 113–117.

properly shaped to write the selected character, and the final set of deflection plates direct the shaped beam to the proper position on the screen. Some sort of recording, either by conventional photography or by xerography, is provided to obtain a permanent record. The Stromberg Carlson SC-5000 High Speed Electronic Printer uses a Charactron and a xerographic printer developed by the Haloid Corp. It can print 10,000 characters per second, with up to 150 characters per line.

In all the devices mentioned above, the results are finally recorded in ordinary decimal-digital form. Other devices are available that prepare a plot of one output quantity versus another, for example, by printing a sequence of dots on a moving piece of coordinate paper. Such a curve can be generated also by displaying the successive points on a CRT and recording them all on a single photographic plate.

Just what kind of printer should be used in a computing or data-processing system depends to a large extent upon the type of work it is chiefly intended to do. In scientific computation, the total volume of results can be expected to be fairly small, except, perhaps, in such matters as the computation of tables. In various operations of the office-arithmetic variety, the situation is quite different; consider, for example, the problem of preparing premium notices in a large insurance company.

Generally speaking, it is not necessary or even desirable to feed the results directly from the computer to the printer. The usual procedure is to feed the results first to an output unit, where they are recorded on whatever medium is used, and, finally, to read this at lower speed, if necessary, to drive the printer. As an example, consider the SEAC scheme mentioned in Sec. 16-4; the output from the machine is recorded on magnetic wire, this is then read and transferred to punched paper tape, and this, finally, is read and used to drive a rather slow printer. With an output unit as intermediary between machine and printer, printing can, of course, be done while the machine is out of service or occupied on other problems.

If only fairly slow printers are available, and if the volume of results to be printed is large, it is possible to distribute the information read from the output medium to a number of printers; this would not be satisfactory for scientific calculation, but would be quite feasible in such operations as the preparation of insurance-premium notices alluded to above. Unfortunately, the high-speed printers developed so far either do not yield as clear and elegant copy as lower-speed mechanical printers or are not capable of providing multiple copies with the same facility. For example, mechanical printers can produce a number of good carbon copies along with the original, but a device that depends upon photographing numbers produced on a CRT must make separate prints from the negative.

16-7. Buffering, Especially in SEAC. After the lengthy though somewhat sketchy review of input and output devices just completed, attention must be turned to the practical problem of interconnecting these devices with the computer proper, that is, to the means by which information is exchanged between input-output unit and internal memory. Naturally, this is a very broad subject, and the means adopted in any case depend upon the type and speed of the computer and the type and speed of the input-output equipment. The expressive term "buffering" has been applied to this transfer process.[1]

One of the things that must be accomplished is to effect some sort of speed changing, for, as noted, input-output equipment is generally much slower in operation than internal memory or computing circuits. Furthermore, the input-output devices all involve mechanical motion, and, clearly, the effects of inertia and friction must make it difficult to synchronize them with the internal circuitry. This situation is worst in serial synchronous machines and least troublesome in parallel asynchronous machines, but it must always be considered carefully.

From the analysis of the last paragraph, it follows that the buffering process involves both conversion of the information read from the input medium into a form suitable for introduction into the high-speed memory and synchronization of this information with the internal timing of the machine. Another operation that must usually be performed is the assembly of the information received from the input unit into machine words. Consider, for example, a serial machine with a four-channel input unit; 4 bits are read simultaneously, then nothing happens until the next 4 bits arrive under the reading heads, these are read simultaneously, and so on. Finally, after the words have been assembled, they must be transmitted to the high-speed memory. Some of these operations can be avoided if the input system is in a sense well-matched to the rest of the computer; later in this chapter, in the account of the input-output unit of the IAS computer, it will be seen how a very simple solution was found in which buffering was not required.

So far, input alone has been considered, but it is obvious that it is similarly necessary to change the speed, electrical form, and timing of words in the transfer from the computer memory to the output units.

Since very many ways of buffering can be devised, it is not worthwhile to indulge in a protracted general discussion; consider instead some concrete illustrations. Figure 16-5 shows the method used in SEAC to obtain a synchronized standard pulse for each nonsynchronized nonstandard pulse appearing at the input. The nonstandard pulses occur

[1] A. L. Leiner, Buffering between Input-Output and the Computer, *NBS Rept.* 2154, December, 1952.

less frequently than the standard ones and are of greater length; the standard SEAC pulses occur at a prf of 10^6 per second and are about 0.5 μsec long. Suppose that a pulse then arrives at the input to the gate. The other input is the continuous train of 1-megacycle clock pulses; so the gate output is a packet of such pulses, of which perhaps the first and last may be of less than standard amplitude and width because of finite rise-and-fall time of the long information pulse. These stimulate a pulse repeater, from which two outputs are taken, one a copy of the input, the other a copy with polarity reversed (this is easy with SEAC circuitry, because of the transformer output from all units). The second train of pulses is delayed two pulse periods (2 μsec) and is applied as an "inhibit" input to a gate; the other inputs are the direct train of pulses

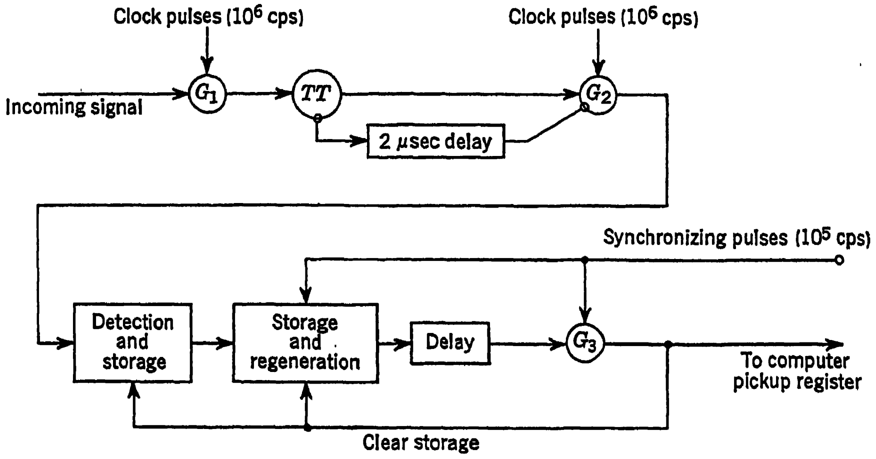


FIG. 16-5. NBS system for deriving a synchronized standard pulse from each non-standard, nonsynchronous input pulse.

and the 1-megacycle clock pulses. The gate passes the first two pulses of the direct train, but the third is either completely inhibited or perhaps transmitted with diminished amplitude if the first pulse is, as explained above, substandard; all following pulses of this packet are completely suppressed. In any case, at least one pulse of standard amplitude and length gets through, but there is uncertainty about its timing.

This uncertainty is removed by the pulse-synchronizing scheme shown in the second line of Fig. 16-5. The synchronizing pulses are of 10^5 per second prf and are derived by a counting procedure that selects every tenth pulse from the clock pulses. The first block in the diagram is a one-pulse dynamic memory unit, which, once stimulated, puts out a continuous train of pulses. These are fed to a second such unit, but cannot stimulate it until the unit is enabled by one of the synchronizing pulses. The output of the second unit is, therefore, a continuous train of

1-megacycle pulses of which the first exactly coincides in time with a synchronizing pulse. This is subjected to a delay of 1 or more μsec but shorter than the interval between synchronizing pulses, which prevents the train of pulses from reaching the output gate while the synchronizing pulse that initiated it is still present. Hence, the gate output is a pulse coinciding in time with the next following synchronizing pulse; this is also fed back to inhibit the dynamic-memory circuits and, hence, to return them to quiescence. It is clear that the initial input pulse can be of any length and that every such pulse gives rise to just one synchronized short pulse of the type used in the computer. Obviously the prf of the incoming signal must be less than 10^5 cps. If the incoming signal consists of pulses of length less than 10 μsec occurring at a prf less than 10^5 cps, it is possible to dispense with the first line of Fig. 16-5.

The standardized and synchronized pulses must now be assembled into words. One way to do this in a serial machine would be to use a processing-loop pick-up register, that is, a register of the recirculating type in which the total delay can be either the synchronizing pulse interval (10 μsec) or 1 μsec greater. Every time a new information pulse is read in, the 10-μsec delay is increased to 11 μsec; so the new pulse falls into position right behind the last one received. By this method, a word can be assembled in 10-bit segments. Another method is to use a shifting register, where the information pulses are read into the left end and the contents are then shifted one place right. This has the advantage that it can be used with either parallel or serial memories, for the register can transmit the assembled word either broadside or by shifting right, emitting 1 bit per shift (which is the scheme used in SEAC).

It is necessary also to provide some way of indicating that the pick-up register has been filled and that it is ready to transmit the assembled word. A very simple expedient is to clear the register to 0's after every such transfer and to insert a 1 in the left end; when this 1 emerges from the right end, it is interpreted as a signal that the register holds a word and is ready to transmit.

If a shifting pick-up register is used as intermediary between a serial input unit and a parallel memory, the broadside transmission of the word into the memory is accomplished very rapidly; then the register is cleared immediately and made ready to receive the next word. With a serial memory, it is convenient to have a second buffer register into which the pick-up register shifts its contents as soon as it has been filled, for, if this is not done, the pick-up register must stand inactive until the memory address to which the picked-up word is to be sent becomes available. In SEAC, the adder register is used for this purpose; it is free, since no provision is made for carrying out an input operation and computing concurrently.

Each word received from the input unit must be sent to some definite memory address. In the case of a single word this is a simple matter, for the input instruction specifies the address completely. If a block of words is to be received into the memory at consecutively numbered addresses, the input instruction must specify only the address of the first and the number of words in the block, and provision must be made for increasing the address by unity after each transfer and for counting the words transferred, comparing the counter's contents with the number of words to be transferred, and generating a signal to terminate the process when coincidence is reached.

Clearly, if the computer is to be able to perform calculations and input or output operations simultaneously, it is necessary both to avoid the use of any arithmetic-unit registers in the buffering process and also to provide a separate register and adder to take care of the address information while the input or output operation is in progress, so that the adder register may be freed for other uses.

16-8. Input-Output Unit of the IAS Computer. The input-output device used at one time with the IAS computer[1] was the IBM 514 reproducing punch, which has facilities for both reading and punching cards. With this was associated the external control, which received commands from the computer and in turn controlled the operation of the punch. As an IBM card has 12 rows and 80 columns of punching positions and as the 514 can handle cards at the rate of 100 per minute, an information rate of 1,600 bits per sec could be attained. Actually only one word was recorded in each row; the columns 1 to 4, 6 to 9, . . . , 46 to 49 were used, and the space at the end of the row (columns 66 to 80) was reserved to hold identifying information useful to the operators of the equipment. In each row in each position it was possible to punch a hole or not and, hence, to record a 1 or a 0; so the words had to be reduced to some form of binary representation. Instruction words were in any case written as strings of 1's and 0's, and number words were so written by replacing each decimal digit by its 4-bit binary equivalent (conversion to binary was accomplished within the machine). Similarly, numbers to be read out of the machine's memory were converted into binary-coded decimals before being transmitted to the punch. All positions in a row were sensed or punched simultaneously. Transmission of information took place over 40 parallel wires, one for each binary position. This means that the word-assembly problem did not exist and that, in fact, the whole buffering problem was trivial.

The method used for communicating between the 514 and the computer is shown in Fig. 16-6, in which one of the 40 parallel channels is repre-

[1] Staff, Electronic Computer Project, "Report on Contract No. N-7onr388, T. O. I.," Institute for Advanced Study, Princeton, N.J., April, 1954.

sented. A word to be transferred from the Williams memory to the punch comes in to R^3 and then passes through the adder, R_1 having previously been cleared to 0's, and appears at the digit-resolver output, where in each position 0 is represented by 0 volts, 1 by a negative voltage. The relay in the plate circuit of the triode is set in the W (write) position; so the tube functions as a cathode follower. The gate G_2 is enabled by the IBM_P signal supplied by the external control, so that, if a 1 is to be written, a second gate (actually a relay) is enabled, providing a path by which the pulse output of the emitter of the 514 can pass to the punch magnets. In the transfer from the 514 to the Williams memory

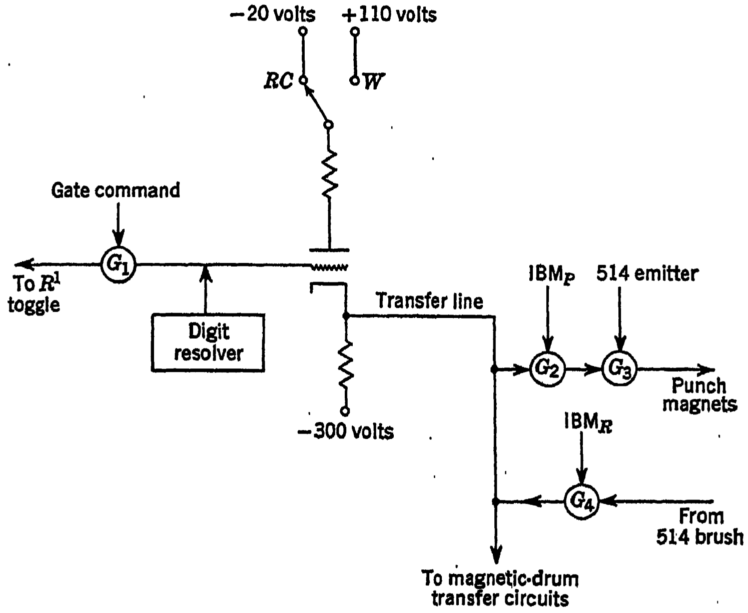


FIG. 16-6. Digit transfer between IAS computer and input-output unit (one line of 40).

the relay in the plate circuit of the triode is switched to the R-C (read-compute) position, the gate G_2 is disabled and the gate G_4 enabled by the IBM_R signal, and the digit-resolver output is made 0 volts. Electrically, the pulse output from the brushes of the 514 is a positive-going signal which is inverted, so that a 1 appears as a negative-going pulse at the output of G_4. The triode acts as a diode, since its plate circuit is disconnected for the signal levels used. Hence, a 1 read in from the 514 pulls the grid down as it falls, and a negative signal is presented to the gate G_1, disabling it, so that the gate-enable signal, which arrives during the pulse from the 514, has no effect. However, R^1 has been previously cleared to 1's; so the net effect is that the corresponding stage of R^1 ends by holding a 1. If a 0 is read by the 514, no pulse is received, and the digit-resolver output is undisturbed at 0 volts, which holds G_1 open; so the gate-enable pulse causes the toggle in the corresponding stage of R^1 to switch to 0. Hence, at the conclusion of the gate-enable signal, the toggles of R^1 hold the entire word read from the punched card.

It is now necessary to consider how the transfers are controlled. Two instructions are needed to effect any of the possible transfer operations. The first of these is the "priming" instruction, which must always be in phase 2, or the second half, of an instruction word. The address part, in bits 23 to 29, specifies the number of IBM cards to be processed. In the magnetic-drum case, bits 20 to 24 specify the number of the starting block, and bits 25 to 29 specify the number of blocks. The first three bits of the operation part of the word must be step/no step = 1, int/ext = 0, Wms/arith = 1; in Fig. 16-7*a* this is represented symbolically. As far as the physical realization is concerned, recall that, since the two instructions in a word occupy separate and distinct parts of R_3, it is a simple matter to provide that the operations be synthesized in different ways in the two parts. Basically, the priming request is used to enable a set of 10 gates, which cause the address part of the priming instruction to be transmitted to a 10-bit register in the external control. Apart from this, the main control also executes whatever operation is specified by the 10 bits in positions 30 to 39 and so brings, finally, an irrelevant number into R_1. If the second of the two transfer instructions is to be that in the first phase of the next instruction word, this is of no consequence; if other instructions are to intervene and if it is desired to preserve the contents of R_1, this can be effected, e.g., by making the remaining 7 bits of the priming instruction the same as those of an *R*II-load instruction.

The second of the pair of transfer instructions must always be in phase 1. That this is an external instruction is signified by step/no step = 1, int/ext = 0. The Wms/arith bit specifies whether the magnetic drum (Wms/arith = 1) or the IBM 514 (Wms/arith = 0) is involved, and the *RO*/*NRO* bit specifies whether the transfer is to be from (*RO*/*NRO* = 1) or to (*RO*/*NRO* = 0) the Williams memory. The synthesis of the relevant bits of the phase 1 instruction is shown logically in Fig. 16-7*b*. The address part of the instruction gives the first address in the Williams memory to be read from or written into. The IBM request causes the 514 to be activated via appropriate relays, and the Wms $\rightarrow$ IBM or IBM $\rightarrow$ Wms request causes the appropriate relay IBM_P or IBM_R to be energized.

The external control contains the 10-bit register already mentioned, a 7-stage binary counter with coincidence circuits by which its contents can be compared with the number in the register (or in either half of it, in the case of magnetic-drum operations), and various toggles and logic circuits. No attempt will be made to present a detailed logical diagram;

an over-all account of its operation will suffice. The external control is supplied with the commands shown in Fig. 16-7*b*, which determine which course of action it is to pursue. It transmits commands to the IBM 514 and receives back certain synchronizing pulses: synch 1, supplied by a cam after each card has been completely read, and synch 2, supplied by the emitter every time a single line is read. It also communicates with the main control, as noted below. The synch 1 and 2 pulses are used to advance the 7-stage binary counter by 1 for each card read or punched.

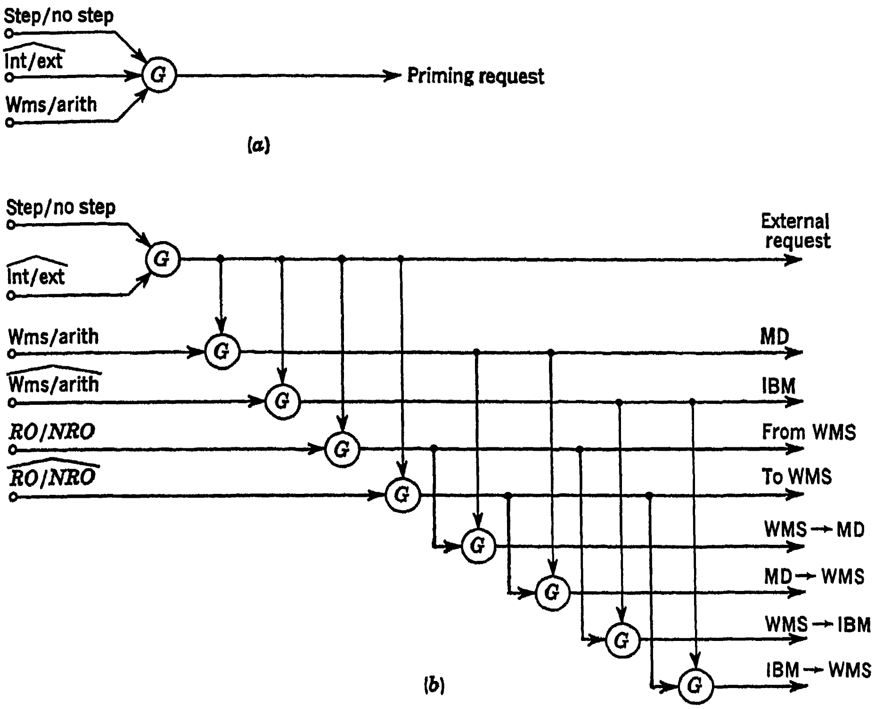


FIG. 16-7. Synthesis of external instructions. (*a*) Phase 2, (*b*) phase 1.

In the IBM case, at the expiration of the delay due to the time required for certain relays to pick up, two toggles (called T_{STT} and T_{block}, not to be confused with the similarly named toggle of the main control) are turned on and the 7-stage counter is cleared.

In the meantime the address part of the instruction has been inserted in the address generator. This fact is signaled back to the external control, where it turns on a toggle $T_{DO/3}$, which ensures that, after each word transfer, the address in the address generator is transferred to the order register of the dispatch counter and then back into the address generator, having 1 added to it in the course of the first transfer.

When a word is read by the 514, the synch 2 pulse is transmitted to the main control, where it turns T_{block} on. The IBM-load instruction must have $\Sigma/\div\times = 1$, so that, when T_{block} is turned on, an addition is performed, and, since precautions have been taken to clear R_1 to 0's and to null the complement gates, the net effect is to accept the word read by the 514 and to zigzag it down to R_1. The counter-satisfy signal is transmitted to the external control, where it turns on a toggle T_{yes}, which in turn issues a Williams request to the Williams-memory local control, which, therefore, in its next action cycle causes the word in R_1 to be recorded at the address in the address generator. The synch signal from the Williams-memory local control in turn turns off T_{yes}. Thus a separate Williams request is generated for each word transmitted to the Williams memory.

In a read-out from the Williams memory to the 514, the word first goes to R^3, and R_1 is cleared to 0's; then, when the positive complement gates are enabled, the word appears at the output of the digit resolver and is automatically punched into the card in the next word position. In this case, the Williams request is initiated by turning on T_{block} in the external control.

When, finally, all the cards required have been processed, the coincidence of the number in the counter and the number in the last seven stages of the register is sensed by the recognition circuit. This generates a signal that turns T_{block} (of the external control) off. The coincidence of $T_{B0/3}$ on and T_{block} off causes the G_2 or G_4 gates in the main control to be disabled. This causes the "external" request to be disaffirmed, and this effect in turn turns off T_{STT} and $T_{B0/3}$ of the external control. The last event removes the disabling signal to the G_2 and G_4 gates, and the main control is now free to execute the phase 2 instruction, which must always be an unconditional transfer of control, since the address of the next instruction was lost when the starting address for the transfer was transmitted from the address generator to the order register.

CHAPTER 17

SUPERSPEED COMPUTERS

17-1. Introduction. The history of the development of digital computing equipment has from the beginning been characterized by a quest for greater speeds and greater memory capacities. In Chap. 3 brief mention was made of a discussion by H. H. Goldstine showing why, from the point of view of scientific computation, it would be desirable to have faster machines. It is obvious also that greater speed, if coupled with high reliability, is important in nonmathematical applications of computing equipment, for example, in certain automatic-control situations.

The present chapter will be devoted to an account of programs that are presently (1958) directed toward the creation of "superspeed" computers. This term is of course not well defined; it really means a machine about ten times as fast (in some sense) as commercial equipment currently available. Thus, to qualify now for consideration here, a design would have to contemplate a multiplication time of about 20 μsec and a memory cycle of, say, 2 μsec or less. It is interesting to notice that a machine (the NORC) only slightly slower than this in arithmetic speed was built some time ago by IBM for the U.S. Naval Proving Ground at Dahlgren, Virginia, and has been in operation there since early in 1955.[1] The development programs that will be discussed below are being conducted by (1) the Data Processing Systems Division of the National Bureau of Standards, (2) the Sperry Rand Corporation (LARC), (3) the IBM Corporation (STRETCH), and (4) the Digital Computer Laboratory of the University of Illinois. The reader will notice that in all these developments a certain degree of concurrency in the operations has been introduced in the interest of speed.

17-2. NBS High-speed Data-processor Development. The first system chosen for discussion is one under development in the Data Processing Systems Division of the National Bureau of Standards. Although it is somewhat slower than the machines to be discussed later in the

[1] M. H. Weik, A Second Survey of Domestic Electronic Computing Systems, *BRL Report* 1010, pp. 280-281, Ballistic Research Laboratories, Aberdeen Proving Ground, Md., June, 1957.

chapter, it exhibits many novel and interesting features of organization.[1] The system actually comprises two complete computers, called the primary and secondary, external (or auxiliary) memory, input-output equipment, input-output control, and a manual-monitor unit designed to facilitate the intervention of the operator in machine operation.

The primary computer has an arithmetic unit, a program control unit, and a large-scale high-speed memory and thus essentially forms a complete computer. The arithmetic unit executes all the normal machine operations under instructions from the program control unit, but such red-tape operations as address modification and counting (e.g., of the steps in an iterative process) are assigned to the secondary computer. The word length is 68 bits. In number words, 64 of these convey numerical information, and the remaining 4 are used for the sign and the "tag" which indicates whether the number is to be interpreted as binary or decimal, fixed-point or floating-point, all these number systems being usable at the option of the coder. The instructions interpreted by the program control unit are of the three-address type, containing (for example) addresses of the operands and of the location to which the result is to be delivered, an operation code, and codes specifying which *B* registers are to be used. *B* modification of any or all of the three addresses is provided for, and it is also necessary to specify in which of the 16 *B* registers the address of the next instruction is to be found. Provision is made for addressing up to 32,768 locations in the high-speed memory. This will be of the conventional coincident-current magnetic-core type, but there will be a small diode-capacitor memory of 256 words at 1 μsec access time between the large memory and the arithmetic unit. Provision is made in the instruction code to call for the transfer of an arbitrary block of words from one of these memories to the other. Once initiated, the transfer runs concurrently with the operation of the arithmetic unit, although of course there are interlocks which can temporarily inhibit operations to ensure that words needed by the arithmetic unit have indeed been placed in the short-access memory. The word length in the memory is 72 bits, the extra 4 bits being used for parity checking.

The secondary computer consists of an arithmetic unit, a program control unit, and a small memory which holds 60 words of 16 bits each. Sixteen of the memory locations function as *B* registers. The others can be used as counters or to hold information to be used in certain special tasks which can be assigned to the secondary computer. As it is actually a general-purpose machine in its own right, it can be used for a

[1] A. L. Leiner, W. A. Notz, J. L. Smith, and A. Weinberger, Organizing a Network of Computers to Meet Deadlines, *NBS Rept.* 5647, November, 1957. Reprinted in *Proc. Joint Computer Conf., Dec. 9–13, 1957,* pp. 115–128, Institute of Radio Engineers, New York, 1958.

variety of operations auxiliary to those of the primary computer, e.g., in sorting, in error analysis, and in general in any operations on short numbers not over 16 bits long. The memory is to be of the diode-capacitor type, with an access time of 1 μsec. The program control unit of the secondary computer uses a two-address code. The primary program control unit can insert an instruction into the secondary program control unit and can cause the transfer of words between primary and secondary memories, while the secondary program control unit can cause the transfer of the contents of the primary instruction register to the secondary memory and the transfer of a word from the primary memory to the primary instruction register. In general, the secondary computer operates concurrently with and subordinate to the primary.

The external memory is to consist of magnetic-tape units, though other devices, such as drums or disks, could also be used. Communications between it and the primary memory will be regulated by the input-output trunks. Transfers in either direction can be initiated by a single instruction, the blocks transferred being specified at will. Once begun, the transfer goes on concurrently with the operations of the primary and secondary computers, subject to certain interlocks to prevent interference with the progress of the computation. Words passing through the trunks can be subjected to a certain amount of processing by the format controller, which is a small plugboard-controlled computer. It can do such things as search words coming from the external memory for various tags, convert words from one coded form to another, and so on.

Provision is made for great flexibility in the use of input and output equipment, including picture-tube display of real-time data held in the high-speed memory. Input and output equipment is to communicate with the external memory and with the primary arithmetic unit.

Control of the machine from without is through the manual-monitor unit. By this the operator can intervene directly in the operation of the machine, or he can set up—by means of switches—various conditions which, being satisfied, will cause the machine to pursue courses of action specified by other switch settings. This amounts to a method of causing the machine to monitor its own operations. By these means the operator can, when certain conditions arise, insert new data and new instructions without the use of special codes prepared in advance. He can also easily extract intermediate computational results. It is obvious that a flexible unit of this sort is a great convenience in code checking and in trouble shooting. It will also be possible to specify and initiate operations through the manual-monitor unit by electrical signals received from remote-control points.

The machine code includes the usual arithmetic, transfer of information, and transfer of control instructions. The logical operations are transplant and a very flexible Boolean operation permiting the generation of any of the Boolean functions of two variables (on, of course, a bit-by-bit basis). Naturally there are instructions to cause the transfer of blocks of words between the external memory and the memory of the primary computer and between the higher- and lower-speed parts of the memory of the primary computer. Finally there are instructions which coordinate the operations of the two computers to ensure that one does not get too far ahead of or behind the other and to cause one to wait until the other catches up under certain specified conditions.

The machine is to use vacuum-tube circuitry based upon that successfully used in the SEAC and DYSEAC. The basic package will bear a strong likeness to that discussed in Chap. 6. A five-phase clocking system replaces the earlier three- and four-phase systems. High-speed arithmetic is obtained by the use of the Weinberger-Smith adder described in Chap. 10. Including all accesses (in obtaining the instruction, the two operands, and in putting the result into the memory), addition and multiplication are to require average times of 7.5 and 31 μsec, respectively. These are very short compared with those obtained with current machines, although longer than those planned for the machines to be discussed below. This development program has the very considerable merit that essentially all the improvements are to be obtained by better organization of the system and that no new and untried circuits or components are required.

Besides being a general-purpose computer and data processor of advanced design, this system was conceived as a step toward a solution of the problem of organizing a network of data-processing machines to work cooperatively on a common problem. Such networks would be of obvious utility in business-data processing, in simulation, and in real-time control operations.

17-3. LARC.[1,2] Some idea of the external characteristics of the LARC system will be given first, to be followed by an account of some of the novel features of the design that permit these characteristics to be realized physically. Briefly, the LARC is a parallel, synchronous machine, handling either fixed- or floating-point decimal numbers (each of 11 decimal digits plus sign) and using a single-address code. The times required to perform the arithmetical operations are to be as follows: (1) addition or subtraction, 4 μsec; (2) multiplication, 8 to 12 μsec; and (3) division, 28 to 32 μsec. Although the first LARC will probably have a memory of 20,000 words, provision is made to address 97,500 separate locations in

[1] Weik, *op. cit.*, pp. 194–195.

[2] J. P. Eckert, Univac-Larc, the Next Step in Computer Design, *Proc. Eastern Joint Computer Conf., 1956*, pp. 16–20, American Institute of Electrical Engineers, New York, 1957.

the main memory. This is to use magnetic cores, organized in blocks of 2,500 words. Each block is to operate on a 4-μsec cycle and is to be addressable independently of and concurrently with any other block. It will also be possible to address a number of high-speed registers (to which access can be had in 1 μsec): the A registers (for holding operands and intermediate results) and the B registers or index registers. The main magnetic-core memory is to be backed up by a very-large-capacity magnetic-drum memory, consisting of up to 24 drums holding 250,000 words each. Provision is made for great flexibility in the use of input and output devices, among which are to be fast magnetic-tape units, a high-speed printer, and a Charactron printing and plotting unit.

Interesting features of the logical design of LARC are the organization of the high-speed memory into independent compartments (alluded to above) and the provision of separate computing units that can operate concurrently to carry out different functions. One of these is called the "processor," and is used to regulate communications among the magnetic-core memory, the drum memory, and the input-output units and also to edit input and output data. The processor can carry out these functions while the main computing unit is busy performing arithmetical operations. The design allows for incorporation of a second main computing unit if this should turn out to be desirable.

As observed above, in all the designs to be treated in this chapter some degree of concurrency of operation is introduced in the interest of speed. This can be done in a variety of ways, as will become apparent, but a general tendency is to separate the functions of high-speed arithmetic and of "bookkeeping," to assign them to separate complexes of equipment, and to make the operations of these complexes of equipment as independent as possible, so that they can, at least in many situations, operate concurrently.

Electronically, the LARC is to be an entirely solid-state machine: all logical, control, and arithmetical functions are to be performed by transistor and crystal-diode circuits, the high-speed memory function by magnetic cores. Circuits must be capable of response at a rate of 2 Mcps, which is surely not high, considering the over-all speed of the machine.

17-4. STRETCH. This system is under development by the International Business Machines Corporation.[1] It is to be a very fast machine: the high-speed parallel arithmetic unit is to effect floating-point addition in 0.6 μsec (i.e., the sum of incident and resident numbers is completely formed 0.6 μsec after the incident number has been presented to the accumulator) and multiplication in 1.2 μsec. To these times must be

[1] S. W. Dunwell, Design Objectives of the IBM Stretch Computer, *Proc. Eastern Joint Computer Conf., 1956*, pp. 20–22, American Institute of Electrical Engineers, New York, 1957.

added the access times to the memory for each operand, or 0.2 μsec for each. The memory is to include a very-high-speed section operating on a 0.5-μsec cycle, with output information available 0.2 μsec after the beginning of the cycle, and a high-speed section operating on a 2.0-μsec cycle, with output information available 0.8 μsec after the beginning of the cycle. The very-high-speed memory is to be organized in sections of 512 words, the high-speed memory in sections of 8,192 words. The very-high-speed unit will consist of arrays of multipath cores, described in Chap. 12 above.[1] These will be backed up by very-large-capacity magnetic-disk memories, each disk being capable of holding 1 million words. It is planned to provide in the design for the possibility of addressing up to 1 million individual memory locations.

The great arithmetic speed quoted above is to be obtained partly by using very fast components and partly by novelties in the logical structure of the system. High-speed transistor toggles and logic circuits using IBM's "drift" transistors have been designed to respond at rates up to 10 million pulses per second.[2]

Although the logical organization has not been described in any more detail than that of the LARC, it should be evident that the basic design philosophy is much the same; here, too, considerable effort has been made to permit diverse functions to be performed concurrently. Thus the various sections of the high-speed and very-high-speed memories can operate concurrently. Separate computers are provided for handling data transfers and for performing high-speed arithmetic. The former functions, including also editing and conversion between binary and decimal representations, are given to a serial computer, which thus takes care of much of the bookkeeping, leaving the high-speed parallel arithmetic unit free to work on the actual computations demanded for the solution of the problem in hand. These two computers operate independently and concurrently. It should be mentioned here that the system design contemplates the use of a wide variety of input and output devices including, in addition to conventional ones, large-capacity magnetic disks, which can communicate with the rest of the system at a rate of one word every 4 μsec, electronic printers and plotters, display units, and a variety of other devices.

17-5. The Illinois Very-high-speed Computer—Generalities. A very interesting machine is under development in the Digital Computer

[1] W. W. Lawrence, Jr., Recent Developments in Very-high-speed Magnetic Storage Techniques, *Proc. Eastern Joint Computer Conf., 1956*, pp. 101–103, American Institute of Electrical Engineers, New York, 1957.

[2] R. A. Henle, High-speed Transistor Computer Circuit Design, *Proc. Eastern Joint Computer Conf., 1956*, pp. 64–66, American Institute of Electrical Engineers, New York, 1957.

Laboratory of the University of Illinois. As much more detailed reports about this have been published than about either of the other machines so far considered, more space will be devoted here to this machine than to the others; it will be possible to show quite specifically how the high operating speed is obtained.[1]

Like LARC and STRETCH, this machine demands high-speed circuitry, a short-access-time memory, and a considerable amount of concurrency in operation. Concurrency is handled somewhat differently than in the other two machines. It will be observed that several features of the design are imposed by the fact that the electronic circuitry is too fast for memories with access times of 1 μsec or more, so it is necessary to minimize reference to the "high-speed" memory. If a higher-speed memory becomes available before the machine is completed, it will be possible to use it to advantage. The memory to be used will probably be made of magnetic cores organized on the "word-arrangement" principle discussed briefly in Chap. 12, with a capacity of 8,192 words 52 bits long, operating on a 1.5-μsec cycle.

This design is a natural development of the single-address asynchronous logic due to von Neumann, Goldstine, and their associates of the Institute for Advanced Study (IAS), which has been treated in detail above. It is interesting to note the vitality of this approach as shown by this latest extension of it.

In this design, the concurrency sought is that of arithmetic and "red-tape" instructions, which comprise such things as address modification, counting, and the transfer of information. There are separate controls to regulate these functions, as will be described below. There are also multiple instruction registers, two extra registers into which operands are inserted as they emerge from the memory, and extra registers, associated with the arithmetic unit, which are used as temporary repositories of the results of arithmetical instructions. In addition, there are 12 B registers and an associated adder, so that address modification and counting can be performed independently of and concurrently with the operation of the arithmetic unit.

17-6. The Operand Registers. When a word is read from the high-speed memory, it is transmitted to a register Z which determines what is to be written into the memory during the second, or "write," portion of the memory's cycle and, hence, causes regeneration of the destructively read information. It may as well be noted here that a word to be written

[1] See On the Design of a Very High Speed Computer, *Univ. Illinois Digital Computer Lab. Rept.* 80, October, 1957. This report, the effort of a committee consisting of D. B. Gillies, R. E. Meagher, D. E. Muller, R. W. McKay, J. P. Nash, J. E. Robertson, and A. H. Taub (chairman), describes work supported in part by the AEC and ONR under AEC contract AT(11-1)-415.

into the memory is first inserted into Z. In the subsequent memory cycle, the address to be written into is read, thus clearing it for the reception of the new word, but the path to Z is disabled, for otherwise the word to be written would be lost. There is of course a path from the arithmetic unit into Z. Now if the word read represents a number to be operated upon, it is received also by one of two registers X and Y. These are identical, and which one receives a given word is determined by the "advanced control," as will be shown below.

Both the registers can transmit to the number register M of the arithmetic unit. Besides M the arithmetic unit contains a quotient register Q, an accumulator A, and auxiliary registers R, R_2, R_3, which serve as temporary repositories of information.

All the registers considered above are 52 bits in length, which is the basic word length. A, Q, and M have the shifting property, being of essentially the same logical structure as the shifting registers of the IAS computer; X, Y, Z, R, R_2, R_3 are simple nonshifting registers. A, Q, R_2, and R_3 can be addressed directly by the programmer, but the others cannot. R has a certain special function: when a number is read into A, the number held in A is first transmitted into R; R is used also in the accumulation of double-length products. Access can be had to the operand registers in 0.05 μsec.

17-7. The Instruction Registers. There are two instruction registers, O_1 and O_2, both nonaddressable. As instructions are of two types, "short," consisting of 13 bits, and "long," consisting of 26 bits, as many as 8 orders can be held in O_1 and O_2. These are scanned by both advanced and arithmetic controls; when all the information in one register has been used for the last time, a fresh instruction word is brought in from the high-speed memory. The registers O_1 and O_2 are both 56 bits long. The extra 4 bits are associated one with each control group of the instruction word held and are used to indicate which of registers X and Y actually holds the operand. It is clear that the availability of the two instruction registers and of the B registers makes it possible to execute many subroutines very rapidly without the expenditure of time to bring new orders in from the memory.

17-8. The B Registers. The design provides for 12 B registers, each 13 bits in length. As mentioned above, an arithmetic element is provided so that the contents of a given B register can be increased or decreased by the address part of an arbitrary instruction word, and it is also possible to increase or decrease the contents by unity for counting.

17-9. Number Words and Instruction Words. Before describing the principles of action of the advanced and arithmetic controls, it is necessary to consider the structure of the instruction words. The 52-bit words are numbers or instructions. The number words can be either fixed-point binary numbers (using the 2's-complement representation for negative numbers) or floating-point numbers where 42 bits form a fixed-point binary numerical portion and the remaining 10 bits represent the exponent.

The instruction words consist of four "control groups" of 13 bits each. A short instruction consists of a single control group, a long instruction of two groups. A short instruction is broken down into the following parts: (1) F, 7 "function bits," specifying the operation to be performed; (2) B, 4 bits used in addressing A, Q, R_2, R_3, and the 12 B registers; (3) C, 2 bits called C_1 and C_2. A long instruction consists of a group F, B, C followed by a second group N that specifies a 13-bit address. Whether the instruction is short or long is indicated by the value of C_1: 0 for short and 1 for long. The 4 B bits represent an address, which is the number B itself, if B is 0, 1, 2, or 3, and the contents of one of the registers B_4 to B_{15} if B is one of 4 to 15. In short instructions ($C_1 = 0$) $B = 0, 1, 2, 3$ refer to A, Q, R_2, R_3, respectively. In long instructions ($C_1 = 1$) the address actually consulted is found by adding the contents of the designated B register to N. Finally, the number in the designated B register is increased by 1 after address formation is complete, provided $C_2 = 1$, but not if $C_2 = 0$.

17-10. The Advanced and Arithmetic Controls. The instruction words held in the registers O_1 and O_2 are used by both the controls. The advanced control must first examine an instruction to determine whether its execution requires the consultation of the memory; if so, it takes steps to acquire the needed word. If the instruction calls for writing into the memory, the advanced control obviously must wait until the word to be written is made available by the arithmetic unit. The arithmetic control examines the instruction after the advanced control and causes the execution of the instruction on the word the advanced control has brought from the memory. Thus the two controls scan the contents of O_1 and O_2 in succession and operate independently and simultaneously to the fullest extent practicable. Obviously such operation becomes impossible under certain circumstances: when both controls would have to use the same equipment at the same time, when the operation of one depends upon the availability of a result not yet obtained by the other, or when the operand registers are full and there is nowhere for the advanced control to place a fresh word brought from the memory. Under any of these circumstances, one of the controls must be caused to wait until the situation has cleared up; it is, accordingly, necessary to provide equipment that detects these circumstances and appropriately inhibits action. Advanced control must keep ahead of arithmetic control but, when it gets too far ahead, it must be halted until arithmetic control has time to digest all the information given it and to catch up.

In order to perform the above functions, the two controls contain certain counters and registers that keep track of their activities and hold information relevant to access to the high-speed memory. Two 3-bit counters D_1 and D_2 hold numbers indicating which control groups of O_1 and O_2 are currently being executed by the arithmetic and by the advanced control, respectively. The locations of the control groups in the two registers are numbered as follows: 0, 1, 2, 3 in O_1; 4, 5, 6, 7 in O_2; so that binary 0 in the highest-order stage of the counter indicates O_1 and binary 1, O_2; and the other two bits indicate the location within the register. There are also two 2-bit reversible counters K_x and K_y. These keep track of the use of the registers X and Y; K_x is advanced by unity whenever advanced control uses X, and K_y functions similarly for uses of Y. There are also three 13-bit registers D_3, A_x, A_y and one 14-bit register A_z. Of these D_3 is the control counter, holding the address of the next instruction word to be brought from the memory and placed in O_1 or O_2; A_x and A_y hold the addresses from which the words currently in X and Y were brought; and, finally, the advanced control places in A_z the address portion of each write instruction it encounters, simultaneously setting the extra bit to 1 to indicate that writing has not yet occurred; when writing actually takes place, which means that the arithmetic control has executed the instruction, the extra bit is set to 0.

17-11. The Sequencing of the Two Controls. It has already been stated that the two controls cannot be permitted completely independent operation; the necessary restrictions or interlocks are introduced by constantly comparing the contents of D_1 and D_2. It must be understood that these counters are reset to 0 when they go beyond the count of 7; that is, they count modulo 8.

The general rule is that arithmetic control is never permitted to get ahead of advanced control, for otherwise it would find itself attempting to cause an operation before the operand was available. The specific rules will now be explained.

I. Let the instruction being executed by advanced control be one that does not call for transfer of control. If it is multiply or divide and if the extra bit of A_z is 1, indicating that an unexecuted write instruction is still to be done, then advanced control waits until the write instruction is completed; at that time the switching of the extra bit of A_z to 0 permits (D_2) (that is, the contents of D_2) to be increased by unity. In all other cases (D_2) is increased by unity as soon as the instruction has been executed. If, however, this makes $(D_2) = (D_1)$, then the next instruction is already being executed by arithmetic control, and advanced control waits until $(D_2) \neq (D_1)$. Notice that the condition $(D_2) = (D_1)$ means essentially that advanced control has run a lap ahead of arithmetic control, which could happen in case a complete loop of instructions were held in O_1 and O_2.

II. If the addition of unity to (D_2) causes the most significant bit to change either from 0 to 1 or from 1 to 0, this indicates that one of two things has happened: (*a*) advanced control has exhausted all the instructions in O_1 and O_2; or (*b*) a loop of instructions containing between five and eight control groups is held in O_1 and O_2, and advanced control has jumped from the last back to the first control group (this is done in obedience to a "short-loop" instruction, which will be explained below).

In case *a*, it is clearly time for advanced control to bring in a fresh instruction word from the memory. Before doing this, advanced control first compares the most significant bits of (D_2) and (D_1). If these are unequal, advanced control causes the instruction word located at the address (D_3) to be read into the register O_1 or O_2 specified by the most significant bit of (D_2), causes (D_3) to be increased by unity, and proceeds to execute the instruction given by the first control group of the new word. If, on the other hand, the most significant bits of (D_2) and (D_1) are equal, the arithmetic control has not yet finished using the contents of the register specified by the most significant bit of (D_2); so in this case the advanced control first causes the word at address (D_3) to be read into Z, from which it is delivered to O_1 or O_2 as soon as the most significant bits of (D_2) and (D_1) disagree, and then proceeds to execute the first control group of the new word. This case occurs if, for example, advanced control has run through all the control groups of O_2 while arithmetic control is still busy with instructions in O_1.

In case *b*, where a short loop of instructions is held in O_1 and O_2, a toggle is provided which is set to 1 when the short-loop instruction calls for transfer of control back to the first control group of the loop, and this state of the toggle prevents the calling in of a new instruction word. Advanced control proceeds instead to execute the first control group of the loop and causes the toggle to be reset. The discrepancy between the most significant bits of (D_2) and (D_1) can cause a new word to be brought in only when the discrepancy is sensed immediately after the increase in the contents of (D_2); so the toggle has fulfilled its function by preventing the read-in operation and, thereafter, can be reset as soon as convenient.

III. Once the arithmetic control has caused the required operand to be transmitted to the arithmetic unit from X or Y and has set the arithmetic unit in action, the count of D_1 is increased by unity. Then (D_1) and (D_2) are compared. If $(D_1) = (D_2)$, the arithmetic control is caused to wait until $(D_1) \neq (D_2)$, for otherwise the arithmetic control would be attempting to use an operand not yet obtained from the memory by the advanced control. If $(D_1) \neq (D_2)$, then the arithmetic control proceeds to execute the control group specified by (D_1).

IV. There are special rules that apply to transfer of control instructions. In the case of an unconditional transfer of control, the advanced control, upon encountering it, first waits until it and the arithmetic control are operating upon the contents of the same instruction register (O_1 or O_2). This is indicated by coincidence of the most significant bits of (D_1) and (D_2); it guarantees that the other instruction register is free to receive a fresh instruction word from the memory.

Once this coincidence has been established, the advanced control performs four steps in sequence: (*a*) it complements the most significant bit of (D_2); (*b*) it replaces the other two bits of (D_2) by C_1 and C_2; (*c*) it replaces (D_3) by the address from the current (i.e., the unconditional-transfer) instruction (this may be given by control group N of the instruction or by the contents of B_{15}); and (*d*) it reads out from the memory the word at the address specified by (D_3) and delivers it to O_1 or O_2 depending upon the most significant bit of (D_2); that is, it delivers it to the instruction register that is free to receive new information.

The bits C_1 and C_2 of an unconditional-transfer-of-control instruction are used to specify to which control group of the new instruction word the control is to be transferred. It is, of course, necessary also to change (D_1) so that the arithmetic control also will jump to the new instruction. Hence the leading bit of (D_1) is also complemented, and the other two bits replaced by C_1 and C_2, whereafter the arithmetic control waits until (D_1) $\not\leq$ (D_2), and then executes the control group specified by (D_1).

V. A conditional transfer of control may depend upon the contents of a B register or upon the result of an arithmetic operation held in the accumulator. Examples of the former are: (*a*) add unity to (B_i); if this sum is not equal to 0, transfer control to the leftmost control group of (1) the word under consideration if $C_2 = 0$, or (2) the preceding word if $C_2 = 1$ (this is the "short-loop" instruction); (*b*) add unity to (B_i); if this sum is not 0, transfer control to the control group specified by C_1C_2 of the word located at address N; (*c*) test the most significant bit of B_i; if this is 0, transfer control to the control group specified by C_1C_2 in the word located at address N.

The second kind of transfer is called "A-conditional." A great deal of flexibility is provided. There are two types, depending upon whether the programmer considers it probable that the transfer will be executed or not. In the first of these, advanced control obtains the word specified by address N and puts it in the Z register before the test of (A) is completed; in the second it does not. In both cases, the B bits of the transfer instruction specify the condition to be satisfied. Eight different possibilities are offered to the programmer: e.g., if $B = 0$, transfer if $(A) = 0$; if $B = 1$, transfer if $(A) < 0$; if $B = 4$, transfer if an overflow from A is indicated, etc. In all cases, if transfer is made, it is made to the control group specified by C_1C_2 of the word located at address N.

Two toggles F_1 and F_2 are used to indicate the status of the execution of the transfer instruction. $F_1 = 1$ indicates that the advanced control has reached a conditional-transfer instruction but that the arithmetic control has not, and F_1 is reset to 0 when the arithmetic control reaches the instruction. If the advanced control reaches a conditional-transfer instruction and if $F_1 = 0$, it executes the instruction and sets $F_1 = 1$ if the instruction is of the B-conditional type, while if it is of the A-conditional type, advanced control waits until $(D_1) = (D_2)$ before executing the instruction. If $F_1 = 1$ it must wait until $F_1 = 0$, that is, until the arithmetic control has executed a previous conditional transfer. As to F_2, when the advanced control finds the specified condition satisfied and executes the transfer, it indicates this to the arithmetic control by setting $F_2 = 1$, leaving $F_2 = 0$ if the condition is not satisfied and no transfer is made. F_2 is reset to 0 by the arithmetic control when it reaches the conditional-transfer instruction.

Thus, when the advanced control reaches a conditional-transfer instruction, it first either brings the word at the specified address into Z or not, depending upon the anticipatory or nonanticipatory nature of the instruction, then tests F_1, and if this is 1, waits until it switches to 0. If the instruction is A-conditional, it waits until $(D_1) = (D_2)$, which guarantees that the relevant word is in the accumulator; but if the instruction is B-conditional, it proceeds to perform the specified operation and make the specified test upon the contents of the specified B register. Having made the test, the advanced control either causes transfer or not as determined by the result of the test, sets $F_1 = 1$, and sets $F_2 = 1$ if the transfer is in fact made.

In all cases except that of the short-loop instruction mentioned above, the actual transfer procedure is the same as that described for the unconditional transfer, except that in the case of an anticipatory transfer the new word is brought into O_1 or O_2 from Z. In the short-loop instruction, the most significant bit of D_2 is complemented or not accordingly as $C_1C_2 = 01$ or 00, and then the two other bits of D_2 are set to 0. The arithmetic control plays no part in the transfer beyond providing the number to be tested in the A-conditional case. When it reaches the actual conditional-transfer instruction, it resets F_1 and increases (D_1) by unity if $F_2 = 0$, or changes (D_1) to conform to the new value of (D_2) if $F_2 = 1$, and then resets F_2.

VI. In the loading of a B register from the arithmetic unit, the advanced control is caused to wait until $(D_1) = (D_2)$, to guarantee that the desired information is transferred to the B register.

17-12. Memory Operations. The reading of words from the memory into the registers X, Y, Z is effected by the advanced control. Consider any instruction that does not demand writing into the memory. The advanced control, upon reading it, determines (1) whether a transfer is to be made from the accumulator to a B register (case VI of the last section), and (2) whether (in all other cases) an operand is (presumably) to be obtained from the memory. In case 1, as stated above (case VI, Sec. 17-11), the advanced control waits until $(D_1) = (D_2)$ and then executes the transfer. In all other cases, the advanced control first compares the address given in the instruction with the contents of A_z, then with those of A_x and A_y. If $(A_z) = N$ and if the extra bit of A_z is 1, then Z has not yet received the new value of the desired operand from the arithmetic unit, and the advanced control waits until this extra bit becomes 0, and then reads the word in Z. If the extra bit is 0 or if $(A_z) \neq N$, then the advanced control proceeds to compare N with (A_x) and (A_y), to determine whether the desired operand is already available in X or Y.

If neither $(A_x) = N$ nor $(A_y) = N$, then the advanced control chooses X or Y to receive the operand, by a simple rule: (1) if one of these registers was most recently used in a "copy-previous-operand" instruction, which merely transfers the contents of X (or Y) into R_2, R_3, or Q, then that register is chosen; (2) otherwise, the register that was not most recently used is chosen, this being indicated by a toggle which is set at each use of X and reset at each use of Y. If X is chosen, the advanced control examines the 2-bit counter K_x. If $(K_x) = 0$, then the advanced control causes (N) to be copied into X, sets $(K_x) = 1$, and sets the extra (14th) bit of the control group currently being obeyed to 0, which indicates to the arithmetic control that the operand has been transferred to X. [If Y had instead been chosen, the transfer $(N) \rightarrow Y$ would be made, (K_y) set to 1 (provided it was initially 0), and a 1 inserted in the extra position of the control group.] If $(K_x) = 1$, this means that X still holds a word that has not been used by the arithmetic control; so the advanced control in this case waits until the arithmetic control has acted and reset (K_x) to 0, and in the meantime causes (N) to be read into Z, whence it is transferred to X when $(K_x) = 0$. It will be recalled that when the arithmetic control causes the contents of X to be transferred into the arithmetic unit (e.g., into A, Q, etc.) it also causes the subtraction of unity from the contents of K_x.

Finally, it is possible that $N = (A_x)$ [or (A_y)], indicating that the desired operand is already in X (or Y); in this case the advanced control increases K_x (or K_y) by unity, sets the extra bit of the control group being executed to 0 (or 1) to show the arithmetic control that the desired operand is in X (or Y), and then goes on. Notice that this feature also makes for increased operating speed. To see this, consider the evaluation of a polynomial in x: once x has been obtained from the memory, it can be held in X, thus saving a memory reference in the formation of each of the successive multiplications by which the polynomial is evaluated.

Now consider write instructions. When the advanced control comes upon such an instruction, it first waits until the extra bit of A_z is 0 (for if this is 1, it indicates that the preceding write instruction has not yet been executed), then puts into A_z the address to be written into and sets the extra bit of A_z to 1. At this point, the advanced control has done its part and goes on to the next control group. When, following this, the arithmetic control reaches the write instruction, it puts the word to be written into Z, and compares (A_z) with (A_x) and (A_y). If one of these [say (A_x)] agrees with (A_z), then the arithmetic control first causes the contents of Z to be transmitted into (say) X, then causes the memory to write the contents of Z into address given by (A_z), and then proceeds to the next control group. While writing is in progress, it is of course necessary to lock out the advanced control.

17-13. The Memory. In the present state of computer technology, the problem of devising large-capacity memories with low access times has found the least satisfactory solutions. Conventional magnetic-core memories in which selection is accomplished by coincident-current excitation have been built to operate at a memory cycle of 6 μsec or less, but it seems hardly possible that this can be reduced below about 3 μsec. Magnetic films and multiapertured cores, described in Chap. 12, give promise of cycles of 1 μsec or less, but practical memories using these elements are still in the developmental stage. An alternative to using these elements is to attempt to use conventional ferrite cores in a manner that permits them to be driven much harder than is possible in the coincident-current scheme. Some discussion of the so-called "word-arrangement" or "word-selection" principle of organizing a core memory has been given in Chap. 12. Experiments at the University of Illinois have indicated that it should be possible to realize a cycle of 1.5 μsec in a memory of this character. The design contemplates a capacity of 8,192 words. This will be backed up by magnetic drums and tapes as necessary, to handle problems involving up to perhaps 100,000 words of numerical information. As standard commercial equipment will be used, no further discussion of the auxiliary-memory equipment will be given here.

17-14. The Arithmetic Unit. The arithmetic unit is a chief source of the machine's speed. The structure is a natural development from that of the IAS machine, modified by incorporating several features that save time in the performance of the arithmetic operations. These features have been individually described above, but this is the first arithmetic unit designed to incorporate them all. Two of these features have been mentioned in Chap. 1. They are: (a) recoding of the multiplier in terms of three "bits" -1, 0, 1, in such a way as to minimize the number of nonzeros and, hence, to minimize the number of additions performed in building up a product;[1] and (*b*) carry storage and only partial assimilation of carries during multiplication, with complete assimilation left to the end of the process, as developed at MIT for use in Whirlwind I. Another feature, which in an asynchronous machine minimizes the addition time, is the provision of circuitry that detects the completion of the carries and at that point terminates the addition process; the scheme used is that developed at IAS and described rather fully in Chap. 10.

The principle of the partial assimilation of carries is applied further. In general, assimilation is always postponed until the last possible moment. Thus, in a multiplication, the product is left in unassimilated form until it is to be transferred to the memory. If a number is first to be added to the product, only a partial assimilation takes place during the addition, and the sum is left in unassimilated form; in this process, the time required for one complete assimilation is saved. Complete assimilation is of course required, as noted above, just before the result is transferred to the memory. The time required for complete assimilation is minimized by the incorporation of carry-completion sensing.

Of course, there are specific circumstances in which partial assimilation of carries is required. Examples are: (1) when it is necessary to determine the sign of a result, as in conditional transfers of control; (2) during a multiplication, in the least significant stage of the accumulator, just before a right shift puts the least significant bits into the Q register. Incidentally, it turns out that the latter operation can be carried out conveniently only if the 2's-complement representation of negative binary fractions is employed, though considerations of space preclude the presentation of the argument here. It might be thought that keeping the arithmetic results in unassimilated form in the accumulator might make impossible the detection of overflows—which is necessary, e.g., in fixed-point arithmetic and in carrying out automatic scaling in floating-point arithmetic—but it turns out that unassimilated overflow detection is quite practicable.

The arithmetic unit consists essentially of the number register M, three shifting registers A, Q, and C (the carry register), a quasi-adder, and a carry assimilator. A, Q, and C are double-ranked shifting registers, structurally like those of the IAS computer. Equipment is also provided for recoding the multiplier before it is inserted into Q. The quasi-adder

[1] This scheme is apparently due originally to D. J. Wheeler of Cambridge University, who, however, has not published a treatment of it. It has been recently worked out in detail by G. W. Reitwiesner of the Ballistic Research Laboratories, J. E. Robertson of the University of Illinois, and M. Lehman in a Ph.D. thesis at the University of London, all working independently.

receives inputs from M via a complementing circuit (which makes subtraction possible), from the lower rank of A (as from R_1 in the IAS machine), and from the lower rank of C and delivers outputs to the upper ranks of A and C. From the upper ranks of A and C inputs to the carry assimilator are taken, and this circuit delivers its output back to the lower rank of A. As noted above, Q receives the least significant bits from A at the successive right shifts in a multiplication, so that a double-length product is built up. No separate carry register is associated with Q, for the lowest-order carry in A is assimilated before the shift. Thus a double-length product consists of an assimilated low-order part in Q and an unassimilated high-order part in A and C.

There are a considerable number of subtleties in the arithmetic unit that cannot be discussed without going into great detail, but the chief sources of the high speed of the unit are those mentioned above.

Nothing has yet been said about floating-point operations. These naturally require certain additional features. The unit as outlined above can handle the arithmetical operations on the fractional parts of the numbers, but an auxiliary unit must be provided for addition and subtraction of exponents and for holding the exponents. Additional shifting facilities and equipment for their control are needed; in this connection, it has seemed simplest to provide M with shifting facilities. Also, equipment is needed to carry out such operations as standardization of results, and so on.

Finally, the times required to perform arithmetic are expected to work out to:

Operation	*Time, μsec,* ±10%
Addition (without assimilation)	0.19
Multiplication (without assimilation):	
Average	3.8
Maximum	4.8
Division (without assimilation)	7–20
Carry assimilation:	
Average	0.13
Maximum	0.67

These are, in every case, times required by the arithmetic unit alone, exclusive of access to the memory. However, from what has been said above about the control of the machine, it is clear that, at least in cases where a high degree of concurrent operation is being realized, the addition of access time to the time used by the arithmetic unit would be largely meaningless.

17-15. Input-Output. This will be handled by commercially available magnetic-tape units. It is felt that the best and fastest units currently on the market will be barely adequate to match the high internal speed of the machine. However, current developments point to much faster tape units in the not too far distant future.

17-16. Circuitry. The circuitry follows that of the original IAS machine, ORDVAC, ILLIAC, etc., in being of the direct-coupled type, capable of asynchronous operation. The high speed of operation is made possible by the use of transistors with high α cutoff. The circuit design was based upon a Western Electric transistor (GA-53233); it seems probable that an even better transistor (GF-45011, also developed by Western Electric) with an α-cutoff frequency of 500 Mcps will be used in the construction of the machine. Using these elements, toggles have been designed with operating times of 30 to 40 mμsec (the operating time is defined as the time elapsing from the instant the input begins to change to its new value to the instant the output goes far enough toward its new value to indicate the existence of the new state to circuits that receive information from it). Simpler and faster toggles using the same transistor can be designed, but these suffer from certain disadvantages, such as the need for both positive-going and negative-going inputs, depending upon the initial state of the toggle.

The logic circuits are quite simple. The "not" circuit, or inverter, is essentially a grounded-emitter circuit followed by an emitter follower to provide power gain and hence to enable the inverter to drive several other circuits. Gain in the inverter as a whole gives it a desirable level-restoring feature. The gate is a diode gate, but each diode is driven by an emitter follower to provide power gain. The mixer is again of the simple diode type, but each diode is again driven by a transistor. It was found worthwhile to design a separate exclusive-or circuit [to realize the function $(p \wedge q') \vee (p' \wedge q)$, rather frequently required] instead of constructing it of the gate and mixer building blocks; there is also a simple complement-gate circuit, instead of one constructed from gates and mixers.

At the outputs taken from the toggles, 1 is represented by -2 volts and 0 by $+2$ volts. As these signals pass through successive logic circuits they of course deteriorate, but they can rise and fall, respectively, to -0.55 and $+0.55$ volt before the information carried is lost, these limits being set by the restoring capability of the inverter mentioned above. This means that up to five mixers can be cascaded, or up to nine gates, before level restoration is required.

As the transmission of information from one part of the machine to another requires about 3.3 mμsec per meter of path length it appears that delays thus introduced are comparable to the operation times of the circuits. Under these circumstances asynchronous operation possesses the advantage that the design of the circuits does not require an exact knowledge of transmission delay, which would have to be taken carefully into account in designing a synchronous machine in order to ensure correct operation.

17-17. The Instruction Code. The design contemplates over 100 instructions, divided into the following classes: (1) arithmetic, both fixed- and floating-point (these include read instructions, which cause words to be brought from the memory and inserted in A and Q), (2) write, (3) logic, (4) B register, (5) unconditional transfers of control, (6) conditional transfers of control (both B-conditional and A-conditional), (7) input-output and magnetic-drum, (8) a few miscellaneous instructions, about which nothing further will be said.

In the interest of brevity, nothing will be said about the arithmetic instructions, except that the repertory is quite complete and allows for very considerable flexibility. The logic instructions include the bit-by-bit operations of "not" (complement), "and" (logical product), and "exclusive or."

4. The B instructions provide for loading the B registers in a variety of ways, for modifying their contents, and for writing their contents in the memory. A designated B register can be loaded with (*a*) the number N, (*b*) the contents of another B register, (*c*) a selected control group (designated by C_1 and C_2) of (N), (*d*) the 13 lowest-order bits of (A) or of (Q). It is also possible to load four consecutively numbered B registers, beginning with B_4, B_8, or B_{12}, with the four 13-bit segments of (N). The contents of a designated B register can be (*a*) decreased by unity, (*b*) increased or decreased by N, (*c*) increased or decreased by the contents of another B register. The contents of four consecutively numbered B registers, beginning with B_4, B_8, or B_{12}, can be written into the memory address N. Finally, considering as always the contents of a B register as a 13-bit binary integer, it is possible to add $(B_n) \times 2^{-51}$ to (A) or subtract it from (A).

5. The unconditional transfers of control are (*a*) the conventional transfer to the instruction at a specified address (*b*) "enter subroutine," which in addition sets B_{15} to the address of the next instruction of the main routine, and (*c*) "leave subroutine," which causes the control to be transferred to the address held in B_{15}.

6. As stated in the discussion of the control, there are two kinds of conditional-transfer-of-control instructions, depending upon the state of the accumulator or of the designated B register. The B-conditional instructions are of two types. The short-loop instruction has already been described in the section on the control. There is also available a similar instruction in which unity is subtracted from (B_n) before the test is made. The remaining B-conditional transfer instructions are all of the long (two-control-group) type and call for transfer of control to the group

designated by C_1C_2 of the word at address N if (*a*) $(B_n) + 1 \neq 0$, (*b*) $(B_n) - 1 \neq 0$, (*c*) $(B_n) \neq 0$, (*d*) $(B_n) = 0$, (*e*) the highest-order bit of B_n is zero, (*f*) the highest-order bit of B_n is unity. The A-conditional instructions have already been described above.

Several experienced coders who have experimented with this code have been struck by its elegance, flexibility, and economy, and in particular have found the use of two instruction registers and the short-loop instruction to be very convenient.

17-18. Conclusion. This completes the survey of these superspeed computers. The chief respect in which they differ from the machines that reached successful development in the middle 1950s is the introduction of concurrent instead of sequential operation. It may be expected that this principle will be extended in the future. It may be expected also that other novelties in the organization of computing systems will be discovered. Indeed, the advance of the computer art depends more strongly upon the creation of new concepts of logical organization than it does upon the development of better physical components.

NAME INDEX

SUBJECT INDEX