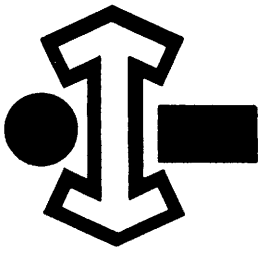
CHARACTERISTICS
OF THE

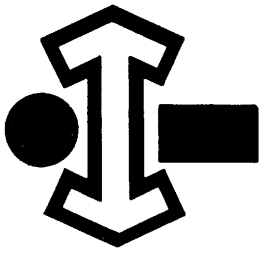
MODEL 1604
COMPUTER



CONTROL DATA CORPORATION

CHARACTERISTICS OF THE

MODEL 1604 COMPUTER



CONTROL DATA CORPORATION

Publication No. 018c

R.– 5/1/61

The Model 1604 Computer

GLOSSARY OF TERMS

The following terms, symbols and abbreviations are defined as they pertain to the discussions of the Model 1604 Computer contained in this manual.

A	Accumulator, or A register
A_n	The binary digit in position n of the A register
Adv	Advance (add 1)
$\longrightarrow$	Transmit to
b	Index designator
B	Index register
(B^b)	Contents of the designated index register
Exit	Proceed to next program step
f	Function code
Half exit	Proceed to lower instruction of a program step
Indirect Addressing	Using the execution address portion of an instruction to specify a storage location which contains the address of the operand
j	The condition designator for jump and stop instructions and the sub-instruction designator for the external function instruction
k	Base execution address for shift and scale instructions
K	Shift count for shift instructions ($k + B^b = K$)
LA	Lower address - the base execution portion of the lower instruction of a program step
LQM	Logical (bit-by-bit) product of (Q) and (M)
m	Base execution address - the 15-bit portion of an instruction as read from storage or the 15-bit portion of an instruction obtained by indirect addressing

M	Execution address - the 15-bit portion of an instruction obtained directly from storage or by indirect addressing which has been modified by the addition of the contents of the designated index register. If b = 0, or $(B^b) = 0$, M = m If b = 1-6, $M = m + (B^b)$
NI	Next instruction
( )	Contents of a register or storage location
( )'	Complement contents of a register or storage location
$( )_i$	Initial contents of a register or storage location
$( )_f$	Final contents of a register or storage location
O^i	Output register i
P	Program address register or P register
Q	Auxiliary arithmetic register or Q register
R	Address buffer register or R register
S^1-S^2	Storage address registers or S registers
Red	Reduce (subtract 1)
Ret Jump	Return jump
UA	Upper address
U^1	Program control register or U register
U^2	Auxiliary program control register
X	Exchange register or X register
y	The base execution address for instructions in which the operand is normally contained in the address portion of the instruction
Y	The operand specified by $y + (B^b)$
Z^1-Z^2	Storage restoration registers or Z registers

TABLE OF CONTENTS

I. INTRODUCTION

Control Data's Model 1604 is an all-transistorized, stored program, general-purpose digital computer. Having a large storage capacity (32,768 48-bit words), exceedingly fast computation and transfer speeds, and special provisions for input-output communications, the 1604 is designed to handle large-volume data processing and to solve large-scale scientific problems.

The following is a summary of the 1604 features:

- Stored-program, general-purpose digital computer
- Parallel mode of operation
- 48-bit word length
- Single-address logic, two instructions per 48-bit word:

  operation code ..... 6 bits

  designator ..... 3 bits

  base execution address ..... 15 bits
- Six index registers
- Indirect addressing feature
- 32,768 48-bit words of magnetic core storage:

  Storage in two independent 16,384 word banks, alternate phased

  4.8 microseconds effective cycle time (representative program)

  6.4 microseconds total cycle time
- Highly versatile input-output facilities:

  Three 48-bit buffer input channels

  Three 48-bit buffer output registers

  One high-speed 48-bit input transfer channel
  (4.8 microseconds, 48-bit parallel word)

  One high-speed 48-bit output transfer channel
  (4.8 microseconds, 48-bit parallel word)

. Program interrupt

. Control console includes:

350 character per second transistorized photo-electric paper tape reader

60 character per second paper tape punch

Input-output electric typewriter

Translated contents of all operational registers displayed as Arabic numerals (octal)

. Flexible repertoire of 62 instructions provides:

Fixed point arithmetic (integer and fractional) . Floating binary point arithmetic . Logical and masking operations . Indexing . Memory searching . Input-output . Sequence control (conditional and unconditional) . Multiple precision capability (accumulator and auxiliary register treated as a single double-length register), etc.

. Binary arithmetic - modulus 2^{48} minus one (one's complement)

parallel addition, 1.2 microseconds basic add time (without access)

. Real-time clock

. Completely solid-state

Diode logic - transistor amplifiers - magnetic core memory

. Small size

Goes in 20' x 20' room

. Low power consumption

In addition to communicating with standard peripheral equipment, such as magnetic tape units, card reader, punch, high-speed printers, and typewriter, the 1604 can also be used for control or communication in radar and sonar systems, real-time instrumentation systems,

digital communication systems, and special display systems.

In the 1604 Computer, input-output operations are carried out independently of the main computer program. When transmission of data is required, the main computer program is used only to initiate an automatic cycle which buffers data to and from the computer memory. The main computer program then continues while the actual buffering of data is carried out independently and automatically.

The input-output section of the 1604 contains the facility for several modes of communication. For normal exchange of data with peripheral equipment, independent control is provided for the transfer of data via three 48-bit buffer input and three 48-bit buffer output channels asynchronously with the main computer program. For high-speed communication one 48-bit input transfer channel and one 48-bit output transfer channel are provided so that two or more 1604's can communicate with each other. Communication control is performed by the external function instruction. In addition, the interrupt feature provides requests from peripheral equipment to the computer.

The storage section of the 1604 is a large-capacity magnetic core storage system providing high-speed, non-volatile, random-access storage for 32,768 48-bit words. One 48-bit word may contain either a 48-bit data word or two 24-bit instructions. The read access time, i.e., the time from request of data to delivery from storage, is 2.2 microseconds.

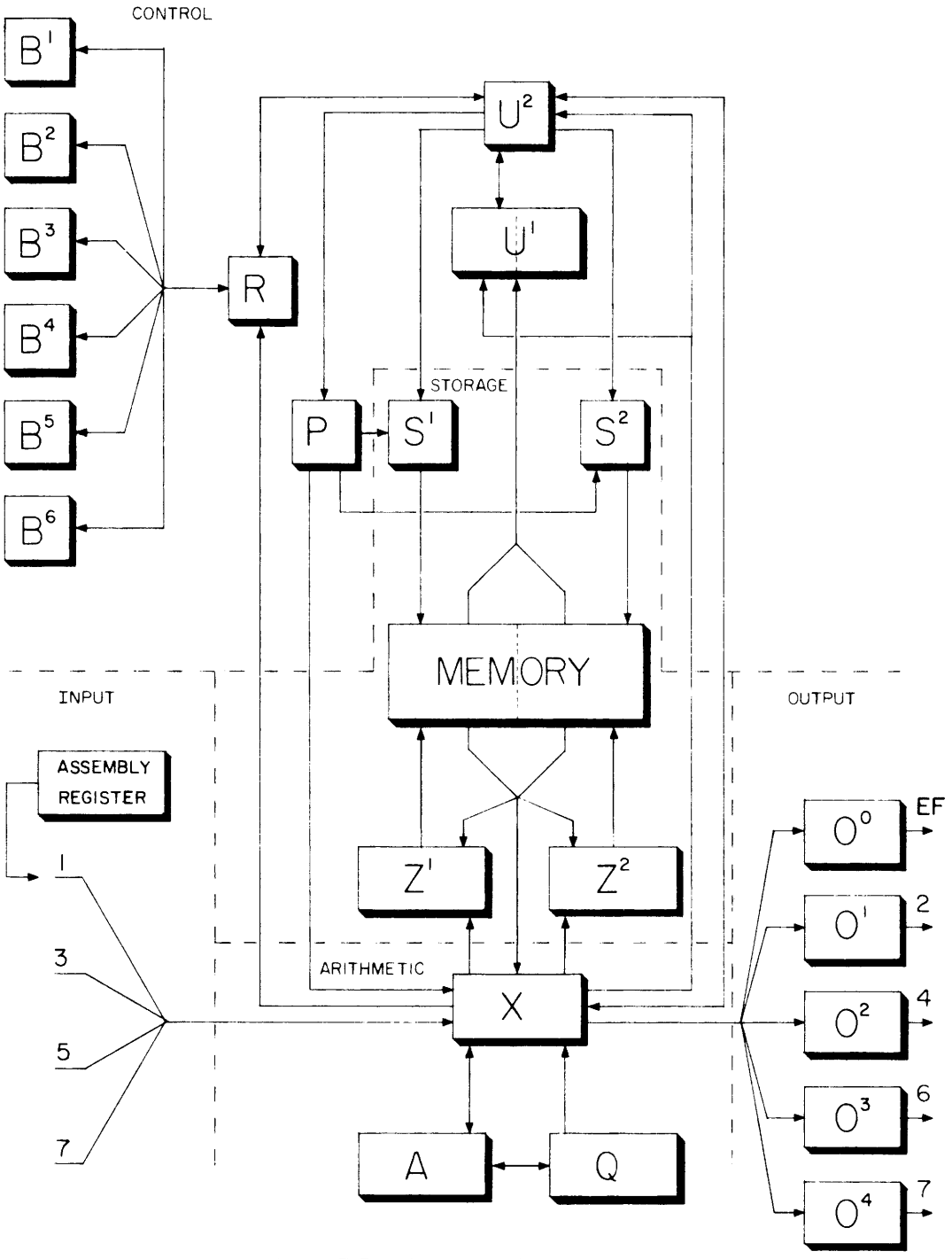
The 1604 instruction repertoire contains a flexible list of 62 instructions which expand into many sub-instructions. These 62 instructions provide fixed binary point arithmetic (integer and fractional), floating binary point arithmetic, logical and masking operations, normal arithmetic operations modulus 2^{48} minus one (one's complement), indexing, memory searching, input-output, sequence control (conditional and unconditional),

multiple precision capability, etc. Some of the special programming features include ease of handling constants, indirect addressing, four search instructions, high-speed input-output transfers, buffering, external function, program interrupt, and a large group of logical commands.

In addition to the standard 1604 console with its display panel (translated contents of all operational registers are displayed in Arabic numerals--octal), typewriter, and paper tape reader and punch, Control Data offers as optional equipment the Model 1607 Magnetic Tape System and the Model 1605 Adaptor. A number of 1607 Magnetic Tape Systems can be attached to a 1604 Computer. Simultaneously among these 1607's three tape handlers can be reading and three can be writing--each at a 30KC character transfer rate. Each 1607 tape system has the facility for simultaneously reading from one tape handler and writing on one tape handler, while the remaining two tape handlers are rewinding. A summary of 1607 features is as follows:

Tape Width	1/2 inch
Tape Recording Density	200 characters per inch
Read-Write Speed	150 inches per second
Rewind Speed	150 inches per second
Character Transfer Rate	30KC
Length of Tape	2,500 feet
Block Length	Variable
Number of Tape Handlers	Four per Model 1607

The Control Data Model 1605 Adaptor permits communication between the 1604 Computer and any of the following IBM peripheral equipment: 714 Card Reader, 727 Magnetic Tape Units, 717 Line Printer, and 722 Card Punch. Each 1605 Adaptor can be



SIMPLIFIED BLOCK DIAGRAM OF THE MODEL 1604 COMPUTER

FIG. I-1

connected to any of the three buffer input channels and three buffer output channels, and each 1605 is independently addressable.

PRINCIPLES OF OPERATION

Physical Description

Standard equipment available with the Model 1604 includes the main computer cabinet and the 1604 console. The console consists of an electric typewriter, paper tape punch, and paper tape photo-electric reader. Optional equipment for the 1604 Computer System includes the 1607 Magnetic Tape System, 1605 Adaptor; the 1606 Printer, the 1608 Control Unit for IBM high-speed tape transports and the 1609 Punched Card Control Unit.

Over-all Analysis of Computer

The computer can be divided functionally into four major sections: (1) Input-output section which provides the means of communication between the computer and the various external equipments, (2) Arithmetic section which performs both the arithmetic and logical operations required for the execution of instructions, (3) Storage section which provides internal storage for both data and instruction, and (4) Control section which successively obtains the instruction from storage, then interprets each instruction to send the required commands to other sections. The control section coordinates and sequences all the operations which carry out the execution of an instruction. A simplified block diagram of the computer is shown in Figure 1-1.

Input-output Section

There are four input channels which bring information into the computer via the X register. Channels 1, 3, and 5 are used for buffer communication; Channel 7 is used in transfer communication -- a very high-speed means of exchanging data. Information from the input-output equipment at the 1604 console is always received via channel 1.

Typically, channel 3 is connected to a 1607 magnetic tape system. Channel 5 provides another means of input which may be used by the 1605 Adaptor, a second 1607, or another equipment.

Output registers O^1, O^2, O^3, and O^4 are used to transmit information from the computer to channels 2, 4, 6, and 7, respectively. Function register O^0 is used to transmit control information to the various external equipments.

Arithmetic Section

The A register, or accumulator, is the principal arithmetic register. Nearly all arithmetic and logical operations make use of A. This register has provisions for the parallel addition of (X) to the contents of A. It can be shifted either separately or in conjuction with the Q register.

The Q register is an auxiliary arithmetic register which assists the accumulator in the performance of the more complicated arithmetic operations. It is used in combination with the X register in the formation of logical products. Q may be shifted either separately or in conjunction with A.

The X or exchange register is used in arithmetic operations as well as in most data transmission between various sections of the computer.

Storage Section

The 32,768 48-bit word magnetic core storage section is controlled by a two-phase timing system, each phase controlling one-half (16,384 48-bit words) of the total storage. All odd storage addresses reference one storage unit; all even addresses reference the other storage unit. The read access time of each section is 2.2 microseconds after which, without delay, the next arithmetic operation is initiated. Each unit has a total cycle time

of 6.4 microseconds. The storage cycles of the two sections overlap one another in the execution of a program, with the result that the effective cycle time is 3.2 microseconds when addresses of alternate memory banks are referenced. The average effective cycle time for random addresses is about 4.8 microseconds for a representative program.

The address register for the even storage unit is S^1; S^2 is the address register for the odd storage unit. In addition, each unit has a storage restoration register (Z^1 and Z^2) which holds the word to be written into a given storage location.

Words to be read out of either storage unit are entered in the X register, and from there transmitted to the appropriate register. Words to be entered or written into a storage unit are transmitted from X to the appropriate Z register, and thence to storage.

Control Section

The control section directs the operations required to execute instructions and to exchange data with external equipment. The major portion of the control section consists of command sequences, static networks for sensing and storing special conditions, and several registers (U, P, R and B^1 through B^6). The control section acquires instructions and initiates the command sequences for executing them. The coordination of operations in the various sections of the computer is maintained by the control section.

II. PROGRAMMING

GENERAL

Operation of the 1604 Computer is sequenced by an internally stored program. The program is contained, along with the data being processed, in a central random-access storage. The input-output equipments, as well as the internal program, have direct access to this common storage unit, thus permitting input-output operations to proceed during computation.

This chapter describes the make-up of the program that orders machine operation. Included in the chapter are a description of the address system, a summary of the repertoire of instructions, and an analysis of each instruction.

1604 ADDRESS SYSTEM

The Control Data 1604 is a single-address computer. As such, one address is explicitly referred to by one instruction. However, the computer is capable of self-modification during the course of a program by the use of the B-boxes or indirect addressing

As already mentioned, the storage system contains 32,768 addresses divided into two units: an odd unit and an even unit. The locations are identified, generally, by the octal range 00000 through 77777 - the even addresses referring to one unit, the odd addresses to the other.

Certain specific storage locations in the memory are used for control and reference functions. These storage locations may be addressed as operands as well as being addressed implicitly by certain control functions. The address assignments for these functions are listed below and will later be explained in detail:

Special Address	Function or Purpose
00000	Real Time Clock (60 steps per second)
00001	Channel 1 control
00002	Channel 2 control
00003	Channel 3 control
00004	Channel 4 control
00005	Channel 5 control
00006	Channel 6 control
00007	Interrupt program (exit-entrance)

Program Step

A step in the computer program is composed of the orders provided to the computer by the contents of one 48-bit word. This word is treated by halves, each containing a 1604 instruction expressed in 24 bits. The two instructions are thus described as the "upper" and "lower" instructions of the program step.

The two instructions can be considered as logically separate entities in a program sequence. As a practical matter in program coding, however, the pair is an entity similar to a two-address instruction. The two single-address instructions are not separable in the sense that the lower may not be executed without the upper.

If it is desired to place only one instruction in a program step, the other instruction location should contain a "do nothing" or pass instruction, described in a later paragraph.

Instruction Composition

The 1604 instruction is a 24-bit quantity specifying an operation which the computer is to perform. The composition of the instruction word is shown in the figure below. Shown

in the figure are the general format of the four specific instruction types. These four specific types are obtained by a variance in the interpretation of the designator and base execution address portions of the instruction. The External Function instruction is a special case and is covered in greater detail later in this chapter.

General Format

6 bits	3 bits	15 bits
Operation Code	Designator	Base Execution Address

Indexed Instructions with storage reference for operand

f = 6 bits	b = 3 bits	m = 15 bits
Operation Code	Index	Address of Operand

Indexed Instructions with self contained operand

f = 6 bits	b = 3 bits	y = 15 bits
Operation Code	Index	Operand

Shift Instructions

f = 6 bits	b = 3 bits	k = 15 bits
Operation Code	Index	Shift Count

Jump Instructions and sense external or internal condition sub-instructions

f = 6 bits	j = 3 bits	m = 15 bits
Operation Code	Condition	Address of Next Instruction

The breakdown of the instruction word, as indicated in the figure is:

6 bits - Operation Code, f

3 bits - Designator - Index, b

- Condition - j

15 bits - Base Execution Address - Address of Operand, m

- Operand, y

- Shift Count, k

The operation code specifies the general character of the instruction. There are 62 operation codes, identified 01 through 76 (octal), available for use in programming. Two codes, 00 and 77, though capable of being expressed as operation codes, are interpreted as faults which will stop computation when translated by the computer.

Depending upon the particular instruction, the designator is interpreted in one of two different ways. As an address modifier (b), it causes the base execution address to be changed by the addition of the contents of the index register specified by the designator. When used in certain of the instructions as a jump designator (j), it determines a change in program sequence, depending upon the condition of certain specified registers.

An index designation of 7 will indicate that indirect addressing is to be used. (Indirect addressing is a means for expanding the reference capabilities of the instruction execution address.)

Eight jump conditions are specified by the jump designator, j. These provide for change of sequence depending upon the status of the A register, the Q register, or the position of jump and stop select switches on the console. These conditions are completely delineated in later paragraphs describing the instructions.

The base execution address of the instruction format (lower 15 bits) holds the address which is basic to the particular instruction. This base execution address is generally the designator of the location of the operand for the instruction which is to be performed. In some cases, the base execution address itself can become the operand, or the shift count. The operation code determines the particular role and definition of the base execution address.

INPUT-OUTPUT

Operation	Instruction Code	Execution Time μs		
		Min.	Av.	Max.
Input transfer	62 b m	4.8	4.0+4.8r	6.8+4.8r
Output transfer	63 b m	4.8	4.0+4.8r	6.8+4.8r
External Function	74 j y	6.4	6.4	6.4

ARITHMETIC

Increase Accumulator	11 b m	2.8	3.0	3.2
Add	14 b m	4.8	7.2	9.6
Subtract	15 b m	4.8	7.2	9.6
Multiply Integer	24 b m	25.2	25.2+.8n	66.4
Divide Integer	25 b m	63.6	65.2	66.4
Multiply Fractional	26 b m	25.2	25.2+.8n	66.4
Divide Fractional	27 b m	63.6	65.2	66.4
Floating Add	30 b m	11.2	18.8	26.8
Floating Subtract	31 b m	11.2	18.8	26.8
Floating Multiply	32 b m	3.2	36.0	57.2
Floating Divide	33 b m	3.2	56.0	57.2
Replace Add	70 b m	10.2	13.2	16.0
Replace Subtract	71 b m	10.2	13.2	16.0
Replace Add One	72 b m	10.2	13.2	16.0
Replace Subtract One	73 b m	10.2	13.2	16.0

SHIFT

Operation	Instruction Code	Execution Time μs		
		Min.	Av.	Max.
Accumulator Right Shift	01 b k	2.8	2.8 +.4s	54.4
Q Register Right Shift	02 b k	2.8	2.8 +.4s	54.4
AQ Right Shift	03 b k	2.8	2.8 +.4s	54.4
A Left Shift	05 b k	2.8	2.8 +.4s	54.4
Q Left Shift	06 b k	2.8	2.8 +.4s	54.4
AQ Left Shift	07 b k	2.8	2.8 +.4s	54.4
Scale A	34 b k	2.8	2.8 +.4s	54.4
Scale AQ	35 b k	2.8	2.8 +.4s	54.4
	TRANSMISSIVE			
Enter Q	04 b y	2.8	3.0	3.2
Enter A	10 b y	2.8	3.0	3.2
Load A	12 b m	4.8	7.2	9.6
Load A, Complement	13 b m	4.8	7.2	9.6
Load Q	16 b m	4.8	7.2	9.6
Load Q, Complement	17 b m	4.8	7.2	9.6
Store A	20 b m	4.8	7.2	9.6
Store Q	21 b m	4.8	7.2	9.6
Substitute Address (upper)	60 b m_u	4.8	7.2	9.6
Substitute Address (lower)	61 b m_l	4.8	7.2	9.6

LOGICAL

Operation	Instruction Code	Execution Time μs		
		Min.	Av.	Max.
Storage Skip	36 b m	7.2	8.8	16.0
Storage Shift	37 b m	10.4	12.8	16.0
Selective Set	40 b m	4.8	7.2	9.6
Selective Clear	41 b m	4.8	7.2	9.6
Selective Complement	42 b m	4.8	7.2	9.6
Selective Substitute	43 b m	5.2	7.4	9.6
Load Logical	44 b m	5.2	7.4	9.6
Add Logical	45 b m	5.4	7.4	9.6
Subtract Logical	46 b m	5.4	7.4	9.6
Store Logical	47 b m	4.8	7.2	9.6
	SEARCH			
Equality Search	64 b m	3.6	4.0+3.6r	6.8+3.6r
Threshold Search	65 b m	3.6	4.0+3.6r	6.8+3.6r
Masked Equality	66 b m	3.6	4.0+3.6r	6.8+3.6r
Masked Threshold	67 b m	3.6	4.0+3.6r	6.8+3.6r
	INDEXING			
Enter Index	50 b y	2.8	3.0	3.2
Increase Index	51 b y	2.8	3.0	3.2
Load Index (upper)	52 b m_u	4.8	7.2	9.6
Load Index (lower)	53 b m_1	4.8	7.2	9.6

INDEXING (cont.)

Operation	Instruction Code	Execution μs		
		Min.	Av.	Max.
Index Skip	54 b y	5.6	5.6	5.6
Index Jump	55 b m	4.4	4.4	4.4
Store Index (upper)	56 b m_u	4.8	7.2	9.6
Store Index (lower)	57 b m_l	4.8	7.2	9.6
	JUMPS AND STOPS			
A Jump	22 j m	4.0	7.2	11.6
Q Jump	23 j m	4.0	7.2	11.6
Selective Jump	75 j m	3.0	7.2	11.6
Selective Stop	76 j m	3.0	7.2	11.6

*s = number of positions shifted

*n = number of one's in multiplier

*r = number of repeated executions

The following paragraphs describe the individual instructions. The title line gives the octal code and format, verbal name, mnemonic code in parentheses, and symbolic description of the instruction.

01 b k A RIGHT SHIFT (ARS) Shift (A) Right by K

This instruction shifts the contents of the A register to the right the number of bit positions specified by the shift count, K. The sign bit is extended and the lowest order bits are discarded as the shift is performed. Shift counts greater than 127 (decimal) are shifted in the normal manner, but are considered shift faults, and produce an interrupt (if selected). An indicator is set which may be sensed by an external function instruction.

02 b k Q RIGHT SHIFT (QRS) Shift (Q) Right by K

This instruction shifts the contents of the Q register to the right the number of bit positions specified by the shift count, K. The sign bit is extended and the lowest order bits are discarded as the shift is performed. Shift counts greater than 127 (decimal) are shifted in the normal manner, but are considered shift faults, and produce an interrupt (if selected). An indicator is set which may be sensed by an external function instruction.

03 b k AQ RIGHT SHIFT (LRS) Shift (AQ) Right by K

This instruction shifts the contents of the A and Q registers to the right as one 96-bit register. The A register is considered as the left-most 48 bits and the Q register as the right-most 48 bits. The number of bit positions is specified by the shift count, K. The sign bit of the A register is extended as the shift is performed. The lowest order bits of

the A register replace the highest order bits of the Q register and the lowest order bits of the Q register are discarded as the shift is performed. Shift counts greater than 127 (decimal) are shifted in the normal manner, but are considered shift faults, and produce an interrupt (if selected). An indicator is set which may be sensed by an external function instruction.

04 b y ENTER Q (ENQ) Y → Q, Extend Sign Y

This instruction enters the execution address portion, Y, of the instruction into the Q register. The operand, Y, is entered into the Q register as a 14-bit quantity plus sign. The highest order bit of Y is copied into the remaining higher order bits of the Q register. No operand storage reference is made in this instruction.

05 b k A LEFT SHIFT (ALS) Shift (A) Left by K

This instruction shifts the contents of the A register circularly to the left the number of bit postitions specified by the shift count, K. The lowest order bits are replaced with the higher order bits as the shift is performed. Shift counts greater than 127 (decimal) are shifted in the normal manner, but are considered shift faults, and produce an interrupt (if selected). An indicator is set which may be sensed by an external function instruction.

06 b k Q LEFT SHIFT (QLS) Shift (Q) Left by K

This instruction shifts the contents of the Q register circularly to the left the number of bit positions specified by the shift count, K. The lowest order bits are replaced with the higher order bits as the shift is performed. Shift counts greater than 127 (decimal) are shifted in the normal manner, but are considered shift faults, and produce an interrupt (if selected). An indicator is set which may be sensed by an external function instruction.

07 b k AQ LEFT SHIFT (LLS) Shift (AQ) Left by K

This instruction shifts the contents of the A and Q registers circularly to the left as one 96-bit register. The number of bit positions is specified by the shift count, K. The right-most bits of the A register are replaced with the left-most bits of the Q register as the shift is performed. The right-most bits of the Q register are replaced with the left-most bits of the A register during the shift. Shift counts greater than 127 (decimal) are shifted in the normal manner, but are considered shift faults, and produce an interrupt (if selected). An indicator is set which may be sensed by an external function instruction.

10 b y ENTER A (ENA) $Y \longrightarrow A$, Extend Sign Y

This instruction enters the execution address portion, Y, of the instruction into the A register. The A register is cleared and the operand Y is entered into the cleared A register as a 14-bit quantity plus sign. The highest order bit of Y is copied into the remaining higher order bits of the A register. No operand storage reference is made in this instruction.

11 b y INCREASE A (INA) $\left[Y + (A) \right] \longrightarrow A$, Extend Sign Y

This instruction adds the operand, Y, to the previous contents of the A register. The operand Y is treated as a 14-bit quantity plus sign in this operation. The addition is performed as if Y were a 48-bit quantity with the higher order bits copies of the sign bit. No operand storage reference is made in this instruction. An overflow condition produces an interrupt (if selected) and sets an indicator which may be sensed by an external function instruction.

12 b m LOAD A (LDA) $(M) \rightarrow A$

This instruction replaces the contents of the A register with an operand, contained in the location specified by the execution address. The A register is cleared, and a storage reference is then made to obtain the 48-bit quantity designated. The 48-bit operand is copied into the cleared A register. Negative zero may be loaded into the A register.

13 b m LOAD A, COMPLEMENT (LAC) $(M)' \rightarrow A$

This instruction replaces the contents of the A register with the complement of an operand contained in the location specified by the execution address. The A register is cleared and a storage reference is made to obtain the 48-bit quantity designated. The 48-bit operand is complemented and entered into the cleared A register. Negative zero may be thus loaded into the A register.

14 b m ADD (ADD) $\left[(A) + (M)\right] \rightarrow A$

This instruction adds a 48-bit operand to the previous contents of the A register. A storage reference is made to obtain the 48-bit quantity contained in the location specified by the execution address. The operand is then added to the previous contents of the A register. Occurrence of an overflow condition produces an interrupt (if selected) and sets an indicator which may be sensed by an external function instruction. A negative zero may be produced by this instruction if and only if both operands are initially negative zeros.

15 b m SUBTRACT (SUB) $\left[(A) - (M)\right] \rightarrow A$

This instruction subtracts a 48-bit operand from the previous contents of the A register. A storage reference is made to obtain the 48-bit quantity contained in the location

specified by the execution address. The operand is then subtracted from the previous contents of the A register. An overflow condition produces an interrupt (if selected) and sets an indicator which may be sensed by an external function instruction. A negative zero may be produced by this instruction if the initial content of A is negative zero and the quantity in storage is a positive zero.

16 b m LOAD Q (LDQ) (M) $\longrightarrow$ Q

This instruction replaces the contents of the Q register with an operand contained in the location specified by the execution address. The Q register is cleared and a storage reference is made to obtain the 48-bit quantity designated. The 48-bit operand is then entered into the cleared Q register. Negative zero may be loaded in Q.

17 b m LOAD Q, COMPLEMENT (LQC) (M)' $\longrightarrow$ Q

This instruction replaces the contents of the Q register with the complement of an operand contained in the location specified by the execution address. The Q register is cleared and a storage reference is then made to obtain the 48-bit quantity specified. The 48-bit operand is complemented and entered into the cleared Q register. A negative zero may be thus loaded in Q.

20 b m STORE A (STA) (A) $\longrightarrow$ M

This instruction stores the contents of the A register at the storage location specified by the execution address. The contents of the A register are not modified by this instruction.

21 b m STORE Q (STQ) (Q) $\longrightarrow$ M

This instruction stores the contents of the Q register at the storage location specified

by the execution address. The Q register content is not modified by this instruction.

22 j m A JUMP (AJP) Jump to m

This instruction has eight sub-instructions which cause a change in the prgoram sequence because of a specified condition of the A register. The index registers are not used for address modification in this instruction. The jump designator, j, in the instruction specifies which sub-instruction is to be performed. In the jump conditions both negative and positive zero are treated as zero.

The sub-instructions and the conditions required to cause a jump in the program sequence are as follows:

22 0 m - Jump if the A register content is zero

22 1 m - Jump if the A register content is not zero

22 2 m - Jump if the A register content is positive

22 3 m - Jump if the A register content is negative

22 4 m - Return jump if the A register content is zero

22 5 m - Return jump if the A register content is not zero

22 6 m - Return jump if the A register content is positive

22 7 m - Return jump if the A register content is negative

23 j m Q JUMP (QJP) Jump to m

This instruction has eight sub-instructions which cause a change in program sequence because of a specified condition of the Q register. The index registers are not used for address modification in this instruction. The jump designator, j, in the instruction specifies which sub-instruction is to be performed. In the jump conditions both negative and

positive zero are treated as zero.

The sub-instructions and the conditions required to cause a jump in the program sequence are as follows:

23 0 m - Jump if the Q register content is zero

23 1 m - Jump if the Q register content is not zero

23 2 m - Jump if the Q register content is positive

23 3 m - Jump if the Q register content is negative

23 4 m - Return jump if the Q register content is zero

23 5 m - Return jump if the Q register content is not zero

23 6 m - Return jump if the Q register content is positive

23 7 m - Return jump if the Q register content is negative

24 b m MULTIPLY INTEGER (MUI) (A) (M) $\longrightarrow$ QA

This instruction forms a 96-bit product from two 48-bit operands. The multiplier must be loaded into the A register prior to the execution of this instruction. The execution address specifies the location of the multiplicand in storage. The resulting product is contained in the QA register as a 96-bit quantity. If the operands are considered as integers, the product is correctly positioned as an integer in the QA register, i.e., the high order bits in Q and the low order bits in A.

25 b m DIVIDE INTEGER (DVI) (QA) / (M) $\longrightarrow$ A; Remainder = Q_f

This instruction divides a 96-bit integer dividend by a 48-bit integer divisor. The 96-bit dividend must be formed in the QA register prior to the execution of this instruction. The 48-bit divisor is read from the storage specified by the execution address. The

quotient is formed in the A register. The remainder is left in the Q register at the end of the operation. The dividend and remainder bear the same algebraic sign.

A divide overflow produces an interrupt (if selected) and sets an indicator which may be sensed by an external function instruction.

NOTE: In the case of Integer Multiply and Divide, it should be noted that the position of the most significant bits in the product and dividend differ from the usual positioning of bits in the AQ register. Since the most significant digits are found in Q, this combined use of A and Q is referred to as QA (See Glossary of Terms for further definitions).

26 b m MULTIPLY FRACTIONAL (MUF) (A) (M) $\longrightarrow$ AQ

This instruction forms a 96-bit product from two 48-bit operands. All quantities involved in this operation are treated as fractions with the binary point immediately to the right of the sign digit. The multiplier must be loaded into the A register prior to the execution of this instruction. The multiplicand is read from the storage location specified by the execution address. The product is formed in the AQ register and the multiplier is discarded in the multiplication process.

27 b m DIVIDE FRACTIONAL (DVF) (AQ) / (M) $\longrightarrow$ A: Remainder = Q_f

This instruction divides a 96-bit quantity by a 48-bit divisor. All quantities involved in this operation are treated as fractions with the binary point immediately to the right of the sign digit. The 96-bit dividend must be loaded into the AQ register prior to the execution of this instruction. The 48-bit divisor is read from the storage location specified by the execution address. At the end of the operation the quotient is left in the A register. The remainder and the dividend bear the same algebraic sign.

A divide overflow produces an interrupt (if selected) and sets an indicator which may be sensed by an external function instruction.

30 b m FLOATING ADD (FAD) $\left[(A) + (M)\right] \rightarrow A$

This instruction forms the sum of two 48-bit quantities which are packed in floating-point format. An operand is read from the storage location specified by the execution address and is added to the previous contents of the A register. The result is normalized and rounded and left in the A register at the end of the operation. The Q register contains the residue from the rounding operation at the end of the sequence. Floating-point range faults (exponent overflow or underflow) produce an interrupt (if selected) and set an indicato which may be sensed by an external function instruction.

31 b m FLOATING SUBTRACT (FSB) $\left[(A) - (M)\right] \rightarrow A$

This instruction subtracts an operand in floating-point format from the previous contents of the A register, also in floating-point format. The operand is read from the storage location specified by the execution address. The result is normalized and rounded in the A register. The residue from the rounding operation is left in the Q register at the end of the sequence. A floating-point range fault produces an interrupt (if selected) and sets an indicator which may be sensed by an external function instruction.

32 b m FLOATING MULTIPLY (FMU) $(A)\ (M) \rightarrow A$

This instruction forms the product of an operand in floating-point format with the previous contents of the A register, also in floating-point format. The operand is read from the storage location specified by the execution address. The result is rounded and

normalized in the A register. The residue from the rounding operation is left in the Q register at the end of the sequence. A floating-point range fault produces an interrupt (if selected) and sets an indicator which may be sensed by an external function instruction.

33 b m FLOATING DIVIDE (FDV) (A) / (M) → A

This instruction forms the quotient of two 48-bit quantities in floating-point format. The dividend must be loaded into the A register prior to the execution of this instruction. The divisor is read from the storage location specified by the execution address. The quotient is rounded and normalized in the A register at the end of the operation. The residue from the rounding operation is left in the Q register at the end of the operation. A floating-point range fault produces an interrupt (if selected) and sets an indicator which may be sensed by an external function instruction.

34 b k SCALE A (SCA) A left until $(A_{47}) \neq (A_{46})$ or k = 0
Reduce k by one per shift; $k_f \rightarrow B^b$

This instruction shifts the quantity in the A register circularly to the left until the most significant digit is immediately to the right of the sign digit. The shift count k is reduced by the number of bit positions shifted. The shift is terminated if k becomes zero before the normalizing operation is completed. In any event the reduced shift count is then entered into the designated index register. The range of k is 0 through 77777_8. The Shift Fault indicator is not affected by the execution of this instruction.

35 b k SCALE AQ (SCQ) AQ left until $(A_{47}) \neq (A_{46})$ or k = 0
Reduce k by one per shift; $k_f \rightarrow B^b$

This instruction shifts the quantity in the AQ register circularly to the left until the most significant digit is immediately to the right of the sign digit. The shift count k is re-

duced by the number of bit positions shifted. The shift is terminated if k becomes zero before the normalizing operation is completed. In any event the reduced shift count is then entered into the designated index register. The range of k is 0 through 77777_8. The Shift Fault indicator is not affected by the execution of this instruction.

36 b m STORAGE SKIP (SSK) (M_{47}) Neg: Exit
(M_{47}) Pos: Half Exit

This instruction senses the sign digit of the quantity in the storage location designated by the execution address. If the quantity is negative, an exit is performed. If the quantity is positive, a half exit is performed. None of the quantities in the operational registers are modified by this instruction.

37 b m STORAGE SHIFT (SSH) (M_{47}) Neg: Exit, Shift Left One
(M_{47}) Pos: Half Exit, Shift Left One

This instruction senses the sign digit of the quantity in the storage location designated by the execution address. If the quantity is negative, an exit is performed. If the quantity is positive, a half exit is performed. In either case the quantity in storage is then shifted circularly to the left one bit position. None of the quantities in the operational registers are modified by this instruction.

40 b m SELECTIVE SET (SST) SET (A_n) = 1 for (M_n) = 1

This instruction sets individual bits of the A register to one where there are corresponding ones in the quantity in the storage location designated by the execution address. This is a bit-by-bit function and does not involve normal addition. Bits in the Accumulator

corresponding to zeros in the operand are not modified. This is the inclusive "or" function.

41 b m SELECTIVE CLEAR (SCL) Clear (A_n) to zero for $(M_n) = 1$

This instruction clears individual bits of the A register to zero where there are corresponding ones in the quantity in the storage location designated by the execution address. This is a bit-by-bit function and does not involve normal addition. Bits in A corresponding to zeros in the operand are not modified.

42 b m SELECTIVE COMPLEMENT (SCM) Complement (A_n) for $(M_n) = 1$

This instruction complements individual bits of the A register where there are ones in the quantity in the storage location designated by the execution address. This is a bit-by-bit function and does not involve normal addition. Bits in A corresponding to zeros in the operand are not modified. This is the exclusive "or" function.

43 b m SELECTIVE SUBSTITUTE (SSU) $(M_n) \rightarrow A_n$ for $(Q_n) = 1$

This instruction substitutes portions of an operand into the A register using the Q register contents as a mask. This may be considered in two steps. Individual bits of the A register are cleared to zero where there are ones in corresponding bits of the Q register. Then those same individual bits of the A register are replaced with corresponding bit values from the storage location specified by the execution address.

44 b m LOAD LOGICAL (LDL) L (Q) (M) $\rightarrow$ A

This instruction loads the A register with the bit-by-bit logical product of the Q register contents and the quantity in the storage location designated by the execution address.

45 b m ADD LOGICAL (ADL) $\left[(A) + L(Q)\ (M)\right] \rightarrow A$

This instruction adds to the A register contents the logical product of the Q register contents and the quantity in the storage location designated by the execution address. This is a normal addition of the selected portion of the operand with all other bits interpreted as zero. Occurrence of an overflow condition produces an interrupt (if so selected) and sets an indicator which may be sensed by an external function instruction.

46 b m SUBTRACT LOGICAL (SBL) $\left[(A) - L(Q)\ (M)\right] \rightarrow A$

This instruction subtracts from the A register contents the logical product of the Q register contents and the quantity in the storage location designated by the execution address. This is a normal subtraction operation for the selected portion of the operand with all other bits interpreted as zero. Occurrence of an overflow condition produces an interrupt (if selected) and sets an indicator which may be sensed by an external function instruction.

47 b m STORE LOGICAL (STL) $L(Q)\ (A) \rightarrow M$

This instruction stores the logical product of the A register and the Q register contents at the storage location specified by the execution address. Neither the A nor the Q register contents are modified by this instruction.

50 b y ENTER INDEX (ENI) $y \rightarrow B^b$

This instruction replaces the contents of the designated index register with the operand y contained in the instruction itself. No storage reference is made in this instruction. If zero is used as the index designator, this instruction becomes the pass instruction.

51 b y INCREASE INDEX (INI) $\left[y + (B^b)\right] \rightarrow B^b$

This instruction adds the operand y to the contents of the designated index register. The addition is performed modulus 2^{15} minus one. No storage reference is made in this instruction.

52 b m_u LOAD INDEX (upper) (LIU) $(m_{UA}) \rightarrow B^b$

This instruction replaces the contents of the designated index register with the address from the upper instruction at the designated storage location.

53 b m_l LOAD INDEX (lower) (LIL) $(m_{LA}) \rightarrow B^b$

This instruction replaces the contents of the designated index register with the address from the lower instruction at the designated storage location.

54 b y INDEX SKIP (ISK) $(B^b) = y$: Exit, Clear B^b

$(B^b) \neq y$: Adv (B^b), Half Exit

This instruction compares the quantity in the designated index register with the operand, y. If the two quantities are equal, then the designated index register is cleared to zero and an exit is performed. If the quantity in the index register is not equal to y, then the quantity in the index register is increased one count and a half exit is performed.

55 b m INDEX JUMP (IJP) $(B^b) \neq 0$: Reduce (B^b), Jump to m

$(B^b) = 0$: Execute NI

This instruction examines the quantity in the designated index register. If this quantity is not zero, then the quantity is reduced one count and a jump is executed to the base

execution address. If this quantity is zero, then the present program sequence is continued.

56 b m_u STORE INDEX (Upper) (SIU) $(B^b) \rightarrow m_{UA}$

This instruction stores the quantity in the designated index register in the address portion of the upper instruction contained in the storage location specified by the base execution address. The remaining bits at the specified storage location are not modified in this operation. This instruction effectively inserts an address in the first instruction at the specified storage location.

57 b m_1 STORE INDEX (lower) (SIL) $(B^b) \rightarrow m_{LA}$

This instruction stores the quantity in the designated index register in the address portion of the lower instruction contained in the storage location specified by the base execution address. The remaining bits at the specified storage location are not modified in this operation. This instruction effectively inserts an address in the second instruction at the specified storage location.

60 b m_u SUBSTITUTE ADDRESS (upper) (SAU) $(A_{00-14}) \rightarrow M_{UA}$

This instruction replaces the address portion of the upper instruction word in the storage location designated by the execution address with the lowest order 15 bits of the A register contents. The remaining bits of the designated word in storage are not modified by this operation. This instruction effectively inserts an address in the first instruction at the designated storage location. The contents of A are not modified by this instruction.

61 b m_1 SUBSTITUTE ADDRESS (lower) (SAL) $(A_{00-14}) \rightarrow M_{LA}$

This instruction replaces the address portion of the lower instruction word in the storage location designated by the execution address with the lowest order 15 bits of the A register contents. The remaining bits of the designated word in storage are not modified by this operation. This instruction effectively inserts an address in the second instruction at the designated storage location. The contents of A are not modified by this instruction.

62 b m	INPUT TRANSFER	(INT)	Transfer (B^b) words to memory, beginning at the last address

This instruction transfers a block of data from an external equipment into the central computer storage. The number of words to be transferred is specified by the contents of the designated index register, B^b. These words are located in a consecutive list which begins at the location specified by the base execution address, m. The transfer begins by storing the first word in the last position in the list. The content of the designated index register is reduced by one for each word that is transferred. The transfer continues until the contents of the designated index register are reduced to zero.

63 b m	OUTPUT TRANSFER	(OUT)	Transfer (B^b) words from memory, beginning with the last address

This instruction transfers a block of data from computer storage to an external equipment. The number of words to be transferred is specified by the contents of the designated index register, B^b. The words to be transferred are located in a consecutive list which begins at the location specified by the execution address, m. The transfer begins by obtaining the first word from the last position in the list. The content of the designated index register is reduced by one for each word that is transferred. The transfer continues until the contents of the designated index register are reduced to zero.

64 b m EQUALITY SEARCH (EQS) Search (B^b) words, beginning with the last word

(M) = A: Exit

This instruction searches a list of operands to find one that is equal to the content of the A register. The number of items in the list is specified by the content of the designated index register. These items are located in a consecutive list beginning at the location specified by the base execution address. The search begins with the last operand in the list. The content of the designated index register is reduced by one for each operand examined. The search continues until an operand is reached that is equal to the content of the A register or until the contents of the designated index register are reduced to zero. If the search is terminated by finding an operand equal to the value in A, an exit is performed. The address of the operand which satisfied the criterion is given by the sum of the base execution address and the final contents of the index register. If no operand in the list is equal to the value in A, then a half exit is performed. If b = 0, only the word at m is searched. In the equality comparison made here, plus zero (that is, all zeros) and minus zero (that is, all ones) are treated as equal.

65 b m THRESHOLD SEARCH (THS) Search (B^b) words, beginning with the last word

(M) > (A): Exit

This instruction searches a list of operands to find one that is greater than the contents of the A register. The number of items in the list is specified by the contents of the designated index register. These items are located in a consecutive list beginning at the location specified by the base execution address. The search begins with the last operand in the list. The content of the designated index register is reduced by one for each operand

examined. the search continues until an operand is reached that is greater than the content of the A register or until the contents of the designated index register are reduced to zero. If the search is terminated by finding an operand greater than the value in A, an exit is performed. The address of the operand which satisfied the criterion is given by the sum of the base execution address and the final contents of the index register. If no operand in the list is greater than the value in A, then a half exit is performed. If b = 0, only the word at m is searched. In the comparison made here plus zero is considered as greater than minus zero.

66 b m	MASKED EQUALITY	(MEQ)	Search (B^b) words beginning with the last word L(Q) (M) = (A): Exit

This instruction searches a list of operands to find one such that the logical product of the operand and the contents of the Q register (that is, the masked operand) is equal to the contents of the A register. The number of items in the list is specified by the content of the designated index register. These items are located in a consecutive list beginning at the location specified by the base execution address. The search begins with the last operand in the list. The content of the designated index register is reduced by one for each operand examined. The search continues until an operand is reached that, when masked, is equal to the value in the A register, or until the contents of the designated index register are reduced to zero. If the search is terminated by finding a masked operand that is equal to the value in A, an exit is performed. The address of the operand which satisfied the criterion is given by the sum of the base execution address and the final contents of the index register. If no operand in the list satisfies the criterion then a half exit is performed. If b = 0, only the word at m is searched.

67 b m MASKED THRESHOLD (MTH) Search (B^b) words, beginning with the last word

L(Q) (M) > A: Exit

The instruction searches a list of operands to find one such that the logical product of the operand and the contents of the Q register (that is, the masked operand) is greater than the contents of the A register. The number of items in the list is specified by the contents of the designated index register. These items are located in a consecutive list beginning at the location specified by the base execution address. The search begins with the last operand in the list. The content of the designated index register is reduced by one for each operand examined. The search continues until an operand is reached that, when masked, is greater than the value in the A register or until the contents of the designated index register are reduced to zero. If the search is terminated by finding a masked operand that is greater than the value in A, an exit is performed. The address of the operand which satisfied the criterion is given by the sum of the base execution address and the final contents of the index register. If no operand in the list satisfied the criterion then a half exit is performed. If b = 0, only the next word at m is searched.

70 b m REPLACE ADD (RAD) $\left[(M) + (A)\right] \rightarrow M \& A$

This instruction replaces the quantity specified by the execution address with its original value plus the value in the A register. The resultant sum is left in the A register at the end of the operation. An overflow condition produces an interrupt (if selected) and sets an indicator which may be sensed by an external function instruction.

71 b m REPLACE SUBTRACT (RSB) $\left[(M) - (A)\right] \rightarrow M \& A$

This instruction replaces the quantity specified by the execution address with its original value minus the value in the A register. The resultant difference is left in the A register at the end of the operation. An overflow condition produces an interrupt (if selected) and sets an indicator which may be sensed by an external function instruction.

72 b m REPLACE ADD ONE (RAO) $\left[(M) + 1\right] \rightarrow M \& A$

This instruction replaces the quantity specified by the execution address with its original value plus one. The resultant quantity is left in the A register at the end of the operation. The original A register contents are destroyed by this operation. An overflow condition produces an interrupt (if selected) and sets an indicator which may be sensed by an external function instruction.

73 b m REPLACE SUBTRACT ONE (RSO) $\left[(M) - 1\right] \rightarrow M \& A$

This instruction replaces the quantity specified by the execution address with its original value minus one. The resultant quantity is left in the A register at the end of the operation. The original A register contents are destroyed by this operation. An overflow condition produces an interrupt (if selected) and sets an indicator which may be sensed by an external function instruction.

74 j y EXTERNAL FUNCTION (EXF)

j = 1-6:	activate channel j
j = 0:	select condition y
j = 7:	on condition y exit or half exit

This instruction has eight sub-instructions which are used to control the transfer of

information between the computer and peripheral equipments. The index registers are not used for address modification in this instruction. The designator is used to specify one of eight external functions to be performed.

The sub-instructions and the operation performed for each are as follows:

74 0 y - Select external equipment or internal condition, y

74 1 y - Activate communication channel one

74 2 y - Activate communication channel two

74 3 y - Activate communication channel three

74 4 y - Activate communication channel four

74 5 y - Activate communication channel five

74 6 y - Activate communication channel six

74 7 y - Sense external or internal condition, y

Sub-instructions 74 1 y through 74 6 y are used to begin buffering a block of data between the computer and a previously selected peripheral equipment. The base execution address is used to designate the starting address in the computer central storage. This address is automatically recorded in the upper address position of the appropriate special storage location (00001-00006). The terminal address (plus one) of the block of data must have been previously recorded, by the program, in the lower address position of the appropriate special storage location prior to the execution of this external function instruction.

Sub-instructions 74 0 y and 74 7 y are used to provide the selection (74 0 y) and sensing (74 7 y) of a multiplicity of internal and external conditions or modes of operations. The 24 bit instruction is interpreted as follows:

6 bits	3 bits	3 bits	3 bits	9 bits
OP Code 74	0 = Select 7 = Sense	0 = Internal 1-6 = Channel	Equipment 0-7	Condition or Mode 000-777

Sub-instructions 74 0 y and 74 7 y may be used as either upper or lower instructions. The Select sub-instruction (74 0 y) will yield the same result in either case. However, the Sense sub-instruction (74 7 y) causes a skip or a wait depending upon its position in the 1604 Computer word.

When used in the upper position, a 74 7 y is a skip instruction. That is, the lower instruction is skipped if the condition given by the EF code is present, but the lower instruction is executed if the condition given by the EF code is not present. In the first case, the 74 7 y "exits" to the next pair of instructions. In the second case the 74 7 y "half exits" to the lower instruction.

When the 74 7 y is used in the lower position, it is not a skip instruction. Instead the sense is executed repeatedly until the condition given by the EF code occurs. At this time an exit is performed to the next pair of instructions. Until the condition given by the EF code is present, the instruction simply half exits to repeat itself. A 74 7 y in the lower position is therefore a means of awaiting the occurrence of a specified condition.

Throughout the list of sense codes below, the term "exit" and "half exit" applies as indicated above. An exit is performed if the stated condition is present; if not present, a half exit is performed.

Internal Condition

Internal Select

74 0 00010 - Select interrupt on channel 1 inactive

74 0 00011 - Remove selection

74 0 00020 - Select interrupt on channel 2 inactive

74 0 00021 - Remove selection

74 0 00030 - Select interrupt on channel 3 inactive

Internal Select (cont.)

74 0 00031 - Remove selection

74 0 00040 - Select interrupt on channel 4 inactive

74 0 00041 - Remove selection

74 0 00050 - Select interrupt on channel 5 inactive

74 0 00051 - Remove selection

74 0 00060 - Select interrupt on channel 6 inactive

74 0 00061 - Remove selection

74 0 00070 - Clear arithmetic faults

74 0 00100 - Select interrupt on arithmetic fault

74 0 00101 - Remove selection

74 0 01000 - Start clock

74 0 02000 - Stop clock

Internal Sense

74 7 00010 - Exit on channel 1 active

74 7 00011 - Exit on channel 1 inactive

74 7 00020 - Exit on channel 2 active

74 7 00021 - Exit on channel 2 inactive

74 7 00030 - Exit on channel 3 active

74 7 00031 - Exit on channel 3 inactive

74 7 00040 - Exit on channel 4 active

74 7 00041 - Exit on channel 4 inactive

74 7 00050 - Exit on channel 5 active

74 7 00051 - Exit on channel 5 inactive

74 7 00060 - Exit on channel 6 active

74 7 00061 - Exit on channel 6 inactive

74 7 00110 - Exit on divide fault

74 7 00111 - Exit on no divide fault

74 7 00120 - Exit on shift fault

74 7 00121 - Exit on no shift fault

74 7 00130 - Exit on overflow fault

74 7 00131 - Exit on no overflow fault

74 7 00140 - Exit on exponent fault

74 7 00141 - Exit on no exponent fault

External Equipment

External Clear

74 0 10000 - Clear all channel 1 selections

74 0 20000 - Clear all channel 2 selections

74 0 30000 - Clear all channel 3 selections

74 0 40000 - Clear all channel 4 selections

74 0 50000 - Clear all channel 5 selections

74 0 60000 - Clear all channel 6 selections

Console Select

74 0 11100 - Keyboard entry and no interrupt on carriage return

74 0 11140 - Keyboard entry and interrupt on carriage return

74 0 11200 - PT reader and no interrupt on end-of-tape

74 0 11210 - PT reader and end-of-tape indicator

74 0 11220 - PT reader and interrupt on end-of-tape

74 0 21100 - Print assembly mode

74 0 21110 - Print character mode

74 0 21200 - Punch assembly mode

74 0 21210 - Punch character mode

74 0 21240 - Turn punch motor off

Console Sense

74 7 11100 - Exit on keyboard carriage return

74 7 11101 - Exit on no keyboard carriage return

74 7 11140 - Exit on keyboard lower case

74 7 11141 - Exit on keyboard upper case

Console Sense (cont.)

74 7 11200 - Exit on PT reader, end-of-tape

74 7 11201 - Exit on PT reader, no end-of-tape

74 7 11210 - Exit on PT reader, assembly mode

74 7 11211 - Exit on PT reader, character mode

74 7 21200 - Exit on punch, end-of-tape

74 7 21201 - Exit on punch, no end-of-tape

1607 Magnetic Tape System - Select

Typically, the 1607 Magnetic Tape System is connected to channels 3 and 4. Thus the channel selection code is 3 or 4. The variable "n" is used to specify one of the four tape units by being assigned a value 1 through 4. The codes are so arranged that tape unit n is selected for an input or output channel operation on the initial selection only; subsequent selector sense operations on the unit do not require the n designator. The n designator is required when changing operations from one tape unit to another. Various conditions, including errors arising in the operation of the tape system, may be sensed by the external sense codes.

74 0 32001 - Read selected tape, binary

74 0 32002 - Read selected tape, coded

74 0 32004 - Interrupt when selected tape ready

74 0 32005 - Rewind selected tape

74 0 32006 - Backspace selected tape

74 0 32007 - Rewind selected tape with interlock

74 0 320n1 - Select read tape n, binary

74 0 320n2 - Select read tape n, coded

74 0 42001 - Write selected tape, binary

74 0 42002 - Write selected tape, coded

74 0 42003 - Write end-of-file mark on selected tape

74 0 42004 - Interrupt when selected tape ready

74 0 42005 - Rewind selected tape

74 0 42006 - Backspace selected tape

74 0 42007 - Rewind selected tape with interlock

74 0 420n1 - Select write tape n, binary

74 0 420n2 - Select write tape n, coded

1607 Magnetic Tape System - Sense

74 7 32000 - Exit on ready to read

74 7 32001 - Exit on not ready to read

74 7 32002 - Exit on read parity error

74 7 32003 - Exit on no read parity error

74 7 32004 - Exit on read length error

74 7 32005 - Exit on no read length error

74 7 32006 - Exit on end of file mark

74 7 32007 - Exit on no end of file mark

74 7 42000 - Exit on ready to write

74 7 42001 - Exit on not ready to write

74 7 42002 - Exit on write reply parity error

74 7 42003 - Exit on no write reply parity error

74 7 42004 - Exit on write reply length error

74 7 42005 - Exit on no write reply length error

1607 Magnetic Tape System - Sense (cont.)

74 7 42006 - Exit on end-of-tape marker

74 7 42007 - Exit on no end-of-tape marker

1605 Adaptor - Select

The condition interpretation codes used with the 1605 Adaptor, select and control the operation of the various IBM equipments connected to the Adaptor. The channel selection code is 5 or 6. The variable "n" is used to specify one of a number of similar units of a single class.

74 0 54000 - Begin cycle or read selected tape, binary

74 0 54001 - Read selected tape, coded

74 0 54002 - Select read, binary, for read-while-write

74 0 54003 - Select read, coded, for read-while-write

74 0 54005 - Rewind selected tape

74 0 54006 - Backspace selected tape

74 0 54007 - Interrupt when selected unit ready

74 0 54100 - Turn on indicator, selected unit

74 0 54101 - Turn off indicator, selected unit

74 0 54200 - Clear Interrupt Selection

74 0 544n0 - Read tape n, binary

74 0 544n1 - Read tape n, coded

74 0 544n2 - Select read tape n, binary, for read-while-write

74 0 544n3 - Select read tape n, coded, for read-while-write

74 0 54500 - Begin cycle card reader, binary

74 0 54501 - Begin cycle card reader, coded

74 0 64000 - Begin cycle or write selected tape, binary

74 0 64001 - Write selected tape, coded

1605 Adaptor - Select (cont.)

74 0 64004 - Write end-of-file, selected tape

74 0 64005 - Rewind selected tape

74 0 64006 - Backspace selected tape

74 0 64007 - Interrupt when selected unit ready

74 0 64200 - Clear interrupt selection

74 0 64100 - Turn on indicator, selected unit

74 0 64101 - Turn off indicator, selected unit

74 0 644n0 - Write tape n, binary

74 0 644n1 - Write tape n, coded

74 0 64600 - Begin cycle card punch, coded

74 0 64601 - Begin cycle card punch, binary

74 0 64700 - Begin cycle line printer

1605 Adaptor - Sense

The following sense codes are used with the 1605 Adaptor. The indicator is always received from the selected IBM unit. An indicator from the card reader indicates that the card feed is empty. From a magnetic tape unit, it indicates either end-of-tape (for a write operation) or end-of-file (for a read operation). An indicator from the line printer indicates the end-of-page. No indicator is received from the card punch.

74 7 54000 - Exit on ready to read

74 7 54001 - Exit on not ready to read

74 7 54002 - Exit on parity error

74 7 54003 - Exit on no parity error

74 7 54004 - Exit on indicator from selected unit

74 7 54005 - Exit on no indicator from selected unit

74 7 54006 - Exit on read length error

1605 Adaptor - Sense (cont.)

74 7 54007 - Exit on no read length error

74 7 64000 - Exit on ready to write

74 7 64001 - Exit on no ready to write

74 7 64002 - Exit on parity error

74 7 64003 - Exit on no parity error

74 7 64004 - Exit on indicator from selected unit

74 7 64005 - Exit on no indicator from selected unit

75 j m SELECTIVE JUMP (SLJ) Jump to m

This instruction has eight sub-instructions which cause a jump in program sequence on specified conditions of operator lever keys. The index registers are not used for address modification in this instruction. The index designator in the instruction specifies which lever key is sampled in determining the jump decision.

The sub-instruction and the operation performed by each when a positive result is obtained from sampling the appropriate lever key are as follows:

75 0 m - Jump unconditionally

75 1 m - Jump if lever key one is set

75 2 m - Jump if lever key two is set

75 3 m - Jump if lever key three is set

75 4 m - Return jump unconditionally

75 5 m - Return jump if lever key one is set

75 6 m - Return jump if lever key two is set

75 7 m - Return jump if lever key three is set

76 j m SELECTIVE STOP (SLS) Stop, jump to m

This instruction has eight sub-instructions which cause the program to stop on specified conditions of operator lever keys. The index registers are not used for address modification in this instruction. The index code in the instruction specifies which lever key is sampled in determining the stop decision. A jump to the base execution address occurs regardless of the stop decision.

The sub-instructions and the operation performed by each when a positive result is obtained from sampling the appropriate lever key are as follows:

76 0 m - Stop unconditionally (normal jump)

76 1 m - Stop if lever key one is set (normal jump)

76 2 m - Stop if lever key two is set (normal jump)

76 3 m - Stop if lever key three is set (normal jump)

76 4 m - Stop unconditionally (return jump)

76 5 m - Stop if lever key one is set (return jump)

76 6 m - Stop if lever key two is set (return jump)

76 7 m - Stop if lever key three is set (return jump)

SPECIAL PROGRAMMING FEATURES

Certain of the instructions are more versatile and have greater flexibility than is indicated in the foregoing analysis of the instructions. The following paragraphs, therefore, elaborate on some of the capabilities of the machine, and present data that will be helpful in obtaining a more complete understanding of the instructions.

Skip Instructions

Several 1604 instructions, when used in the upper position of a program step, provide a facility for exits or half exits. These instructions are:

36 b m	Storage Skip
37 b m	Storage Shift
54 b y	Index Skip
64 b m	Equality Search
65 b m	Threshold Search
66 b m	Masked Equality
67 b m	Masked Threshold
74 7 y	External Function (Sense external or Internal)

In each case, the instruction exits to the next pair of instructions if a specified condition exists. If the condition does not exist, a half exit to itself is performed to the lower instruction.

When the 36, 37, 54, (74 7 y) instructions are used in the lower position of a program step, they provide a means of awaiting the occurrence of a specified condition. The instruction half exits to repeat itself until the specified condition occurs, at which time it exits to the next program step. The search instructions do not half exit when placed in the lower position of a program step.

Jump Instructions

A jump instruction causes the termination of a current program sequence and the initiation of a new sequence at a different location in storage. Continuity between individual steps in a sequence is provided by a program address register. This register always contains the storage location of the program step currently being executed. The address in this register is normally increased by one count at the end of each program step to specify the location of the next step. In the case of jump instructions the program address register is cleared, and a new address is entered from the jump instruction. In all jump instructions the base execution address specifies the beginning address of the new program sequence.

Most changes in sequence are conditional upon a register value or an operator key position. If the criterion specified by the individual instruction is satisfied, the jump is taken and the program sequence altered. If the criterion is not met, the program proceeds to the next sequential instruction, and the jump address is discarded.

A jump instruction may appear in either the upper or the lower positions in a program step. If a jump instruction appears in the upper position in a program step, and the jump is taken, the lower instruction in that step is never executed. If a jump instruction appears in the lower position in a program step, the upper instruction in that step is executed in a normal manner.

A normal jump initiates a new program sequence with the upper instruction in the specified storage location. The location of the program sequence which contained the jump instruction is not retained.

A return jump initiates a new program sequence with the lower instruction contained in the specified storage location. The base execution address of the upper instruction at this specified storage location is replaced with the address of the program step following the jump instruction in the original program. This allows the new program sequence to return to the proper point in the original program at a later time.

A program sequence which is entered by a return jump is called a return jump subroutine. The upper instruction in the first step of such a sequence is a normal jump instruction which is provided with the exit address when the sequence is entered.

Two Address Jumps

A number of the instructions of the Model 1604 Computer are intended for application as two-address jump instructions. When used for this purpose, these instructions (36, 37, 54, 64, 65, 66, 67 and 74.7) are located in the upper position of program steps. The lower

position in such a program step is usually occupied by an unconditional jump instruction, providing for conditional branching.

Various combinations of these instructions can be used to provide two address logic where convenient. For example, if a single instruction is to be executed, or not executed, as a result of a sensing operation, that instruction should be used in the lower position of a program step with the appropriate sense instruction in the upper position. As another example, a conditional jump instruction may be used in the lower position to allow a jump decision based on a second criterion.

Search Instructions

The search instructions (64, 65, 66 and 67) represent an extension of the two-address jump concept. If these instructions are used with index designator zero, the search is performed on only one operand, located as specified by the base execution address.

If the search instructions (64, 65, 66 and 67) are used with an index designator not zero or seven, a list of operands is searched for one satisfying the specified criterion. The search is begun with the last operand in the list and the search address reduced one count at a time until the search criterion is met or the base execution address reached. If the criterion is met the designated index register contains the proper increment to be added to the initial address of the list to obtain the operand which satisfied the criterion. The search is resumed by returning to the search instruction with the index value thus reduced. If the index register designated by b contains positive zero, no words will be searched, and hence the criterion will not be met.

Masking Instructions

Instructions 44, 45, 46, 47, 66 and 67 have provisions for performing operations on only a selected portion of an operand by masking the portion that is not to be used. Masking

is accomplished by formation of the logical product of (Q) and the operand. Typically the mask quantity is entered in Q. The bits of the operand which correspond to the zero bits of Q are entered as zeros in A. The selected or unmasked portion of the operand which is entered in A corresponds to the bits of Q that are ones. For instructions 44, 45 and 46 it is sometimes convenient to use the operand obtained from storage as the mask and (Q) as the quantity to be masked. By using an operand containing all ones in this way, the contents of Q may be 1) loaded in A, 2) added to the contents of A, or 3) subtracted from the contents of A.

Shift Instructions

The contents of the arithmetic registers A, Q, or AQ can be shifted either to the right or to the left. All right shifts are "end-off" in fashion; that is, as the quantity in the register is shifted one place, the least significant bit is discarded. The sign bit is duplicated as the most significant bit position is vacated by the right shift. When the double length register AQ is right shifted, the lower order bits in A are shifted into the upper bit position of Q.

The left shifts are circular, or "end-around" in fashion. As the left shift is performed the content of the sign bit position of the word is transferred into the least significant bit position. Thus all of the bits in the original word are retained. When the double-length register AQ is left shifted, the higher order bits in Q are shifted into the lower bit positions in A; and the content of the sign bit position of A is produced in the least significant bit position in Q.

The shift count, or number of places to be shifted, is determined by the sum of the base execution address and the contents of the index register designated by the instruction. Shift counts greater than 127_{10} cause the shift fault indicator to be turned on, but do not otherwise affect the execution of the shift instructions.

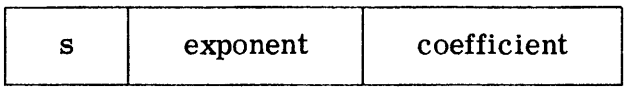
Floating Point

Words packed in floating-point format take advantage of the ability to express a quantity as the product of a fraction and a base number with an integer exponent. The 1604 format includes only the fractional coefficient and the exponent, and their respective signs. The assignment of the factors within the 48-bit word, in order from the most significant bit to the least significant bit is as follows:

1 bit - the sign of the coefficient

11 bits - the signed exponent

36 bits - the magnitude of the coefficient



The 10-bit exponent enables the encoding of quantities with the exponent range $-2^{10}-1)$ to $+2^{10}-1)$, or -1023 to +1023. For precision the coefficient is always written as a fraction (f_n) of the order $1/2 \leq f_n < 1$. The most significant bit will be one for a positive number and zero for a negative number.

In order to make floating-point operation more versatile, operands are included in such a manner that they may be compared with each other in the fixed-point mode. Negative numbers are expressed in floating-point format by one's complement notation. To enable comparison, the most-significant bit of the signed exponent distinguishes the polarity of the exponent by being a "1" when positive and a "0" when negative, as opposed to the usual connotation. The following rules should be observed in packing the exponent into a floating-point word.

If the exponent is positive add the bias quantity 10 000 000 000 to the magnitude of the exponent.

If the exponent is negative add the bias quantity 10 000 000 000 to the magnitude of the exponent and then complement.

Arithmetic performed by the computer with operands packed in the floating-point format follows the normal algebraic rules. In addition and subtraction the coefficients are aligned to have like exponents; the coefficients are then added or subtracted to obtain the coefficient of the answer, its exponent being the common exponent of the aligned coefficients.

The product of two floating-point operands is a quantity whose coefficient is the product of the coefficients of the terms, and whose exponent is the sum of the operand exponents. Similarly, a floating-point quotient is a quantity whose coefficient is the quotient of the dividend coefficient divided by the divisor coefficient, with an exponent which is the difference of the exponents of the terms.

The computer performs floating-point arithmetic by separating the exponents from the coefficients and performing the necessary operations on the two portions independently. The results of the two independent actions are assembled to obtain the final answer.

Scaling

Two instructions, 34 and 35, are used to reposition the contents of the A register (34), or AQ (35), so that the most-significant bit will appear to the immediate right of the sign bit. Scaling is accomplished as a left shift. After each bit position shift A_{47} is compared with A_{46}. If the bits are not alike, a significant bit has moved into A_{46} in which case the operation terminates. The base execution address, k, is used to count the number of shifts in the scaling process. The base execution address, k, is reduced one for each shift. If k becomes zero before a significant bit appears next to the sign bit, the instruction concludes. The final value (k-r) is stored in the designated index register as an indication of the number of shifts (r) needed to satisfy the scale requirements. No shifting occurs if A (or AQ) is initially positive or negative zero. The original value of k is then placed in the designated index register.

Fault Condition

Certain fault conditions may occur in the execution of a computer program which may be sensed later by means of the external function sub-instruction 74 7 y. The sub-instruction 74 0 y is used to clear the fault indicators. These fault conditions occur as a result of arithmetic operations and produce interrupts (if so selected) and set indicators for sensing rather than halting computer operation.

In addition to providing programming indications of fault conditions, the occurrence of a fault produces a display of the fault condition at the operator console. This display consists of a red light on one of the 16 stages of the octal representation of the A register. The displays are organized as follows:

A - Right

1. Floating point range fault - A_0 - A_2
2. Overflow fault - A_3 - A_5
3. Shift fault - A_6 - A_8
4. Divide fault - A_9 - A_{11}
5. Even storage fault - A_{12} - A_{14}
6. Odd storage fault - A_{15} - A_{17}
7. Paper tape punch - A_{18} - A_{20}
8. Reader ready - A_{21} - A_{23}

A - Left

9. Channel one active : A_{24} - A_{26}
10. Channel two active : A_{27} - A_{29}
11. Channel three active : A_{30} - A_{32}
12. Channel four active : A_{33} - A_{35}

A - Left (cont.)

13. Channel five active : A_{36} - A_{38}
14. Channel six active : A_{39} - A_{41}
15. Interrupt request : A_{42} - A_{44}
16. Interrupt lockout : A_{45} - A_{47}

An overflow fault results from the execution of addition and subtraction instructions (11, 14, 15, 45, 46, 70, 71, 72, 73) when the sum or difference of two quantities exceeds the capacity of A.

A divide fault occurs in the execution of fixed-point divide instructions (25 and 27 either when the divisor is zero or when the required quotient exceeds the lower 47 bit capacity of the quotient register, Q.

A floating-point range fault arises in multiply and divide instructions (32 and 33) when the sum of the exponents is 2^{10} or greater, or-2^{10} or smaller, after normalization and rounding, or if the divisor is zero.

A shift fault results from execution of a shift instruction (01, 02, 03, 05, 06, 07) when the shift count is greater than 127 (decimal).

Real Time Clock

Address 00000 in the central storage system is a continually operating time record when selected. The 48-bit quantity is advanced one count every 1/60 of a second. Accuracy is maintained by the 60 cycle power source. The clock is started or stopped by the execution of an external function instruction.

By selecting interrupt on arithmetic overflow, and presetting the contents of address 00000, the real time clock may be used to provide periodic interrupts of the main computer program. The periodicity is variable in increments of 1/60 of a second. Overflow of the

clock is sensed by a negative test on location 00000.

Program Interrupt

A program may be interrupted by 1.) a peripheral equipment initiating a signal on a common interrupt line which links all peripheral equipments, or 2.) the occurrence of a specific internal condition. The main program is interrupted with a return jump instruction to storage address (00007), upon completion of the instruction currently in process. The return jump subroutine analyses the cause of the interruption by interrogating the peripheral equipments or internal condition indicators. If an external equipment has generated the interrupts, the indicated operation is performed. The jump subroutine then returns operation to the main program which proceeds simultaneously with the input or output data transfer, subject only to lockout during single specific memory cycles demanded on a priority basis by the input or output data transfer. By programming means, a program interrupt can be disabled to give priority to main program conditions.

The main program may be interrupted upon the completion of the instruction in either the upper or lower position of an instruction pair. If the interrupt occurs upon completion of the upper instruction, a flip-flop is set to so indicate. The address of the current instruction is retained for re-entry at the lower instruction. If the interrupt occurs after completion of the lower instruction of a program step, the address of the next instruction is retained.

Input-Output Communication

The input-output section of the Model 1604 contains the facility for several modes of communication. For normal exchange of data with peripheral equipment, independent control is provided for the transfer of data via three 48-bit input and three 48-bit output channels asynchronously with the main computer program. For high-speed communication, one 48-bit input channel and one 48-bit output block transfer channel are provided. Communica-

tion is performed by the external function instruction. In addition, the interrupt feature provides requests from peripheral equipment to the computer.

Summary of Buffer Operation. The Model 1604 buffer control continually interrogates all communication channels to determine if a peripheral equipment is ready to send or receive information.

If a peripheral equipment has data ready for transfer, interrogation waits momentarily while a word is being buffered. The buffer control then resumes interrogating the communication channels.

Buffering initiates communication between computer memory, the three buffer input channels, and the three buffer output channels. These buffer information in and out asynchronously with the main computer program.

The three buffered input channels and the three buffered output channels, the interrupt line, and the real-time clock are rapidly scanned by a scanner which looks for action requests from all channels. These action requests are initiated by the peripheral equipment by indicator "flags." A complete scan is made in 3.2 microseconds, which corresponds to the phase rate of magnetic core memory.

When a request is detected by the scanner, the main computer program is halted momentarily to move the data between memory and the requesting channel. The main program proceeds immediately after this action unless the scanner detects that another channel has requested servicing. For example, if the system includes six 1607 magnetic tape systems, all three buffered input channels and all three buffered output channels of the 1604 can operate in the buffer mode simultaneously servicing at full tape-rate three 1607 magnetic tape units operating in the read mode and three 1607 magnetic tape units operating in the write mode.

Summary of High-Speed Transfer Operation. The main computer program performs the high-speed input-output transfer of information between 1604's or between one 1604 and peripheral equipment having comparable speed.

Only one instruction is required for a block of input or output data. A 48-bit word is transferred in or out in 4.8 microseconds.

All transfer operations are carried out via channel 7.

Indirect Addressing

The indirect-addressing feature of the 1604 provides an additional flexibility for programs involving a great deal of address modification.

In indirect addressing, the execution address portion of an instruction is used as the address of a memory location that holds the operand address, whereas direct addressing the operand location is obtained immediately from the execution address (modified by the contents of an index register when desired). An additional memory reference is required in indirect addressing to obtain the operand address.

All instructions except 22, 23, 74, 75, and 76 may be used with either direct or indirect addressing. Indirect addressing occurs when the index code is seven; otherwise direct addressing results. It should be noted that the above statements also apply to instructions which do not normally use the execution address as an address but rather as an operand; (e.g. the shift instructions.)

Consideration of some examples of indirect addressing will aid in understanding it. Suppose that the two instructions in address 05012 are to be executed as part of a sequence (see Table 1A). Because b = 3 in the upper instruction direct addressing is used in its execution. Thus (B^3) is added to 71331 to produce the address of the operand. In the lower instruction b = 7 and therefore m is used as an address for obtaining a new operand address

Now the lower 18 bits are read out of address 00367 (see Table 1B), while the remaining upper bits are ignored. These 18 bits are substituted in the program control register for the original 18-bit quantity comprised of b and m. As a consequence the current instruction has been altered so that it is 14 2 11135.

The index designator b, is examined again, and since it is not 7, the address of the operand is 11135 + (B^2). But, if the new value of b had been 7, a second indirect addressing operation would have resulted. This situation is illustrated by the upper instruction at address 05013. Since b = 7 in this instruction, the lower 18 bits at address 04006 are substituted in the program control register, which then holds 01 7 11466. Since b = 7 again, the lower 18 bits in address 11466 are entered in the program control register. Now, because b = zero, 00012 is used as the execution address.

Table 1. Examples of Indirect Addressing

A. PROGRAM

Address	Upper Instruction	Lower Instruction
	f b m	f b m
05012	36 3 71331	14 7 00367
05013	01 7 04006	12 6 71331

B. OTHER MEMORY LOCATIONS

Address	Content	
00367	01 4 36675	42 2 11135
04006		7 11466
11466		0 00012

III. THEORY OF OPERATION

GENERAL

In order to gain a more thorough understanding and appreciation of the 1604 characteristics, a brief description of the theory of computer operation is presented herein. Although the four logical sections of the computer are primarily contained in the main computer cabinet, they are described as separate entities for discussion purposes. The four sections consist of:

. Input-Output Section

. Storage Section

. Arithmetic Section

. Control Section

It should be noted that the four logical sections are carefully integrated, both physically and electronically, in order that the capabilities of each may be preserved in the total configuration.

INPUT-OUTPUT SECTION

The Input-Output Section of the 1604 Computer provides the means for transmission and proper control of the transfer of information between the computer and the various external equipments. Normally, the timing of this communication is under the control of the external equipment. Input-Output transmissions are of two types: transfer and buffer. Transfer transmissions are under direct program control. Buffer transmissions employ an independent access to computer storage and are asynchronous to the main computer program. Once the buffer channel is activated, the computer is free to work on any program sequence while the exchange of input or output data proceeds to completion without program reference.

A total of six buffer channels, three input and three output, as well as two input-output transfer channels, are provided. Each of these channels contain forty-eight parallel lines. The six buffer channels are grouped into three pairs, each containing one input and one output channel. One transfer channel is used for input and one for output at high-speed information transfer rates. The odd-numbered channel of each pair of buffer channels is designated as the input channel while the even-numbered channel is the output channel. Each channel contains Ready-Resume circuitry for handling the exchange of control signals between the computer and external equipment. In addition there is a scanner which sequentially examines each buffer channel to determine whether a word is to be exchanged. The scanner then examines the next channel. It is possible that all of the buffer channels may require processing simultaneously. In this event the last channel is processed within a maximum elapsed time of 200 microseconds and the communication rate is limited to a 5 kc word rate.

Input information for the computer is taken from the data registers located within the external equipment and is passed directly to the X register; no intermediate registers are used. Output from the computer is passed to the Output Register O^i, associated with the particular output buffer channel or the transfer channel chosen. Information held by the O register passes directly to the output channel involved. These communication channels form the avenues of information passage between the computer and the external equipments.

Information transmission into and out of the computer is performed as a result of the execution of computer instructions which activate a communication path. This path may be activated by the main computer program, or by a request from the external equipment. In the second case, the computer initially generates a specified condition within the external equipment. Then, either the computer periodically interrogates the equipment to determine

if the external equipment has responded or the external equipment may interrupt the computer program when the external equipment has completed its operation.

The Input-Output section can be conveniently divided into three parts, each associated with a control sequence. These are the External Function (EF), Auxiliary (AUX), and Search and Transfer (ST) sequences. The purpose of the EF sequence is to execute instruction 74, External Function. Thus, this sequence handles 1) the sending of control information to I/O equipment when a 74.0, the EF Select instruction, occurs, 2) the activation of a buffer channel when a 74.1, 74.2, through 74.6 instruction occurs, and 3) the sensing of the status of an I/O equipment when a 74.7, EF sense instruction, occurs.

The primary role of the Auxiliary sequence is to perform the exchange of a word of data via one of the six buffer channels. In addition, however, this sequence handles the advancing of the real-time clock and the initial processing of interrupt signals received from an I/O equipment. There are thus three operations performed by the AUX sequence and they are called "AUX operations." The signal indicating the need for performing any of these three types of operation is called an "Auxiliary request."

The Search and Transfer sequence executes the Search instructions (64-67) and the Transfer instructions (62 and 63). The Transfer instructions effect very high-speed data transmission to and from another equipment. The Search instructions do not pertain to input-output functions, but since they are executed by the same sequence as the Transfer instructions, are considered to be in the I/O section of the computer.

Console Input-Output Equipment

Console input-output equipment furnished with the computer utilizes punched paper tape as the media of presenting and receiving information. Each of these equipments is mounted on the computer console.

The IBM electric typewriter modified by Soroban functions as an input as well as an output device. The typewriter functions as an input device by its direct keyboard entry. As an output device, the typewriter generates typewritten copy of information received from the computer.

The Ferranti Photoelectric Paper Tape Reader is an input device which senses information punched in paper tape and presents it to the computer at the rate of 350 characters per second.

The high-speed Teletype BRPE paper tape punch is an output device which produces punched paper tape output at the rate of 60 characters per second.

Optional External Equipment

Optional external equipments which may be used with the Model 1604 Computer include the 1607 Magnetic Tape System, the 1605 Adaptor, and various IBM equipments which are adaptable to the 1604 Computer through use of the 1605 Adaptor.

1607 Magnetic Tape System

Each 1607 Magnetic Tape System contains four magnetic tape units as well as the associated data-handling and control circuitry. The system is contained in a single cabinet. A set of buffer cables containing input lines, output lines, and function lines connects the system to an input channel, an output channel, and a function channel at the computer. Up to six magnetic tape systems can be used simultaneously. More than six magnetic tape systems may be used with certain limitations. If more than one magnetic tape system is used, each system may be connected to any of the three buffer cables since each system has its own complete controls, assembly registers and disassembly registers.

In the standard magnetic tape system, change-on-one's type of recording is used compatible with that used by IBM 727 magnetic tape units. Reflective spots indicate beginning

and end of tape. Thus a reel of tape generated by a 1607 tape unit may be used on an IBM 727 tape unit and vice versa. Forward, reverse and rewind tape speed is 150 inches per second. Recording density is 200 characters per inch, 6 information bits and one parity bit per character; thus the character rate is 30 KC. Tape width is 1/2 inch. Data is recorded in variable-length blocks, with no limit on the number of 48-bit words in a block. The only limitation is that recording must be by whole 48-bit words (8 characters). Length of interblock spacing is approximately one inch.

Data transmissions to and from the tape system are in the form of 48-bit words. The magnetic tape is read in the forward direction. For writing, the tape control section disassembly register disassembles a 48-bit word into eight 6-bit characters and generates a parity bit for each. Reading follows the reverse procedure, that is, eight 7-bit characters are combined in an assembly register into a 48-bit word by taking only the lower six bits of each character. Parity checks are made on reading, and on writing by a read-head mounted .4 inch following the write-head. Parity errors are registered on a flip-flop for subsequent sensing by the computer (a 74.7 instruction is used for this). A parity error does not immediately halt operation, unless a program stop is specified.

Each tape unit has appropriate switches and indicators which provide for unit designation, positioning tape, rewinding and related functions.

The reading and recording heads are electrically isolated on these tape units. This feature allows the tape to be read back during recording for a positive check on both the recording circuits and the magnetic tape quality.

The magnetic tapes produced by the system are compatible with those produced on IBM 727 Magnetic Tape Handlers.

1605 Adaptor

The 1605 Adaptor is a signal converting unit that permits direct communication between the 1604 Computer and any of the IBM input-output equipments listed below.

The communication between the computer and the 1605 Adaptor always consists of a 48-bit word transmitted in parallel over a buffer channel. The adaptor disassembles the 48-bit words into eight 6-bit characters, generates a parity bit, and presents the 7-bit character on the write buses as requested by the particular storage device involved. On input, the adaptor receives 7-bit characters from the input device, assembles the lower six bits of each character into 48-bit words and presents the assembled word for transmission to the computer. The adaptor also checks the parity of each character and records the error condition for sensing by the computer.

The following input-output equipments may be used with the 1604 by means of the 1605 Adaptor:

- 714 Card Reader (via 759 Control Unit)
- 727 Magnetic Tape Units (via 754 Synchronizer)
- 722 Card Punch (via 758 Control Unit)
- 717 Line Printer (via 757 Control Unit)

Information to the line printer and card punch and from the card reader is in the IBM coded format. Even parity is used with this format. Information to and from the magnetic tape units may be in either the coded format (with even parity) or in a pure binary format (with odd parity). The reading of information in the format other than that recorded will cause no change in the information but will cause a read parity error.

A 17 code is used to indicate the end of magnetic tape file. This is always recorded in an even parity. When a 17 code is recorded in the binary format, it is not recognized

as an end of file mark because of the odd parity.

The 1605 Adaptor is treated as an input-output device by the central computer. It uses one input channel and one output channel which may be shared with other input-output equipments. The lower 12 bits of an external function code are used to select the adaptor and one of the IBM equipments. In the case of a magnetic tape unit the external function code also specifies the unit (1 of 8). The channel-ready signal selects the mode (read or write) and initiates the motion of the selected unit. In the case of an input, the motion of the unit will continue to the end of record. If the computer input block transfer length does not agree with the length of the unit record, an error is recorded for later sensing by the central computer.

The write function continues until information is no longer sent from the computer. The block length for writing is, therefore, variable in eight-character increments as determined by the length of the buffered data. For reading from a tape unit a 48-bit word is always assembled and sent to the computer. When eight characters are not available for forming this word, it is filled out with zeros.

Selecting the 1605 Adaptor locks out any further reference to the IBM units for input or output until the first operation is completed. The lockout may be sensed by the external sense instructions or an interrupt may be initiated at the end of the operation. The programmer has the option of either sensing or interrupting to organize the data flow.

Buffer Control

The processes of data input or output are controlled by the computer Input-Output section. The buffer control circuitry produces the various signals which process each buffer channel sequentially, responds to the interrupt signals, performs the operation advance clock, and generates the enables which select special memory addresses.

Buffer Scanner. The Buffer Scanner (BS) is provided to insure that no single input or output channel monopolizes the input-output function at the expense of the remaining five channels. The scanner sequentially samples each of the three input and three output channels. The Buffer Scanner also inaugurates the interrupt and advance clock routines should either of these be required. Each channel is treated with equal priority by the scanner at the rate of once each 3.2 microseconds.

When a channel requires processing, the scanner stops and the word is buffered into or out of the computer. After the buffer is completed, the scanner resumes its sequential scanning operation with the next channel.

In the event that all six channels request action simultaneously, the last channel is processed within a maximum elapsed time of 200 microseconds. Thus the communication rate is limited to a 5 kc word rate under maximum conditions if all channels are used simultaneously. However, under ordinary conditions, the word rate can be expected to be much higher, up to as high as 50 kc under optimum conditions.

Input-Output Specifications

The input-output specifications apply to all devices which connect with the computer. These specifications are written to allow a minimum data exchange time consistent with accepted engineering practices and moderate hardware requirements. Specifications are contained in the Input-Output Specifications of the 1604 Computer and are partially reproduced here.

Communication Lines. The communication lines used between the various external devices and the computer within the input-output system are divided into data lines and control lines. Physically, these lines are grouped into cables. Each cable group handles one input channel, one output channel and corresponding functions and control lines. There are four such cable groups. The group containing the transfer channel is different in detail operation, but identical in logical organization to the three buffer cable groups. The arrangement of the four cable groups is as follows:

Channel 1 - Buffer input
Channel 2 - Buffer output } Cable group 1

Channel 3 - Buffer input
Channel 4 - Buffer output } Cable group 2

Channel 5 - Buffer input
Channel 6 - Buffer output } Cable group 3

Channel 7 - Transfer input and output - Cable group 4

The channels of each of the above cable groups are inseparable. Each channel set is an entity and an external device can use only the two channels of one cable group.

These four cable groups are used for all communication with the 1604. Each cable group contains the data and control lines appropriate to the group. The input buffer channels as well as input on the transfer channel communicate with the 1604 directly via the Exchange Register. The output buffer channels communicate with the external equipment via one of the Output Registers, O^1, O^2, or O^3. Output to the transfer channel is taken from O^4.

Cable groups 1, 2, and 3 are designated for use by low speed external equipment. The transfer channel, cable group 4, is reserved for use by other high speed equipments. Forty-eight input data lines and forty-eight output data lines are contained in each cable group. The various control lines which are a part of each cable group are defined as follows:

Input Information Ready -- The input information ready line originates in the external equipment and terminates in the computer. A static "1" signal is produced on this line by the external equipment when information is present in the output register of the external equipment in a state which the computer may sample. This signal is dropped by the reception of the input information resume signal from the computer.

Input Information Resume --The input information resume line originates in the computer and terminates in the external equipment. A static "1" signal is produced on this line by the computer when the computer has accepted the input word. The input information resume signal from the computer requests the external equipment to turn off the input information ready signal. The end of the input information ready signal from the external equipment turns off the input information resume signal at the computer, thus completing the cycle.

Output Information Ready--The output information ready line originates in the computer and terminates in the external equipment. A static "1" signal on this line accompanies each word of output information. This signal is turned off by the output information resume signal from the external equipment.

Output Information Resume--The output information resume line originates in the external equipment and terminates in the computer. A static "1" signal on this line indicates that the external equipment has accepted the word of information. This signal is dropped by the external equipment when the output information ready signal is dropped.

Input Buffer Active--The input buffer active line originates in the computer and terminates in the external equipment. A static "1" signal is produced on this line whenever the input buffer channel of the cable group is activated. The signal remains on until the final word of the block is entered into the computer Storage section and a resume signal has been sent.

Output Buffer Active--The output buffer active line originates in the computer and terminates in the external equipment. A static "1" signal is produced on this line whenever the output buffer channel of the cable group is activated. This signal remains on until the final word of the block is buffered to the external equipment.

External Master Clear--The external master clear line originates in the computer and terminates in the external equipment. A static "1" signal appears on this line whenever the clear switch at the 1604 console is placed in the up position. This signal clears all external equipment attached to the cable group.

Interrupt--The interrupt line originates in the external equipment and terminates in the computer. A static "1" signal is produced on this line whenever the external equipment has reached the end of an operation or assumed a certain interrupt condition as previously selected by the computer. When a static "1" appears on this line the computer interrupts the main program and enters a special subroutine which determines the cause of the interruption, takes appropriate action and then returns to the main program. The interrupt line remains energized until the computer removes the interrupt selection or the interrupt condition.

Input Function Ready--The input function ready line originates in the computer and terminates

in the external equipment. A static "1" signal is produced on this line when an external function code is present on the external function lines for translation by the external equipment. A signal on this line is used to select input conditions within the external equipments. This signal is automatically dropped after 4.8 microseconds.

Input Sense Ready--The input sense ready line originates in the computer and terminates in the external equipment. A static "1" signal is produced on this line whenever the computer desires to sense the existence of an input condition within the external equipment. This signal is automatically dropped after 4.8 microseconds.

Output Function Ready--The output function ready line originates in the computer and terminates in the external equipment. A static "1" signal is produced on this line whenever an external function code is present on the external function lines for translation by the external equipment. This line is used to select output conditions within the external equipment. This signal is automatically dropped after 4.8 microseconds.

Output Sense Ready--The output sense ready line originates in the computer and terminates in the external equipment. A static "1" signal is produced on this line whenever an external function code is present on the line to sense the existence of an output condition within the external equipment. This signal is automatically dropped after 4.8 microseconds.

Sense Response--The sense response line originates within the external equipment and terminates in the computer. A static "1" signal on this line indicates to the computer the presence of the condition specified by the 12-bit code sent to the equipment via the external function lines. This signal remains on for the 4.8 microseconds of the input sense ready or output sense ready.

External Function Lines--These lines originate in the computer and are continously monitored by the external equipment. Only the presence of the proper input or output ready function (or sense) signal enables the sampling of the content of these lines by the external equipment as an external function code.

The following control wires of the above described lines are not used in the Transfer Channel set although the wires are physically present in the channel set.

Input Buffer Active

Output Buffer Active

Input Function Ready

Input Sense Ready

The transfer channel contains two sets of forty-eight data lines. Since input and output cannot be simultaneous, only one set of control lines is included.

The exchange of control signals which are required in the transmission of the word with the external equipment during an input buffer operation is as follows:

1. The computer, through the proper combination of external select codes, establishes the external equipment from which it is to take information.
2. The computer activates the input buffer active line. This line remains up until the final word of the block is transmitted.
3. The external equipment produces input information ready.
4. The computer, when it has accepted a word of information, produces an input information Resume signal.
5. The input information Resume signal causes the external equipment to turn off its input information ready signal and to prepare another word of input data infor-

mation for the computer. The input data is placed on the data lines in parallel.

6. The removal of input information Ready signal, upon entrance into the computer, turns off the input information Resume signal.
7. Steps 3 through 6 are repeated until the entire block of information is exchanged. When the final word of the block is transmitted and steps 3 through 6 have been completed, the input buffer active signal is dropped.

The transmission of control signals which are required in the exchange of a word of output buffer information after an output buffer channel is established is as follows:

1. The computer, through the proper combination of external select (sense) code, establishes the external equipment to which it is to send information.
2. The computer activates the output buffer active line. This line remains on until the final word of this block is transmitted.
3. The computer places a word in the proper Output register. All data lines are energized in parallel.
4. When all of the data lines are stable, the computer generates an output information Ready signal which indicates to the external equipment that output data is available on the lines in a stable steady-state condition.
5. The external equipment accepts the output information Ready signal and the information at its own rate. When the external equipment accepts the output information Ready signal it produces an output information Resume signal which it sends to the computer.
6. The computer accepts the output information Resume signal and turns off the output information Ready signal.
7. Termination of output information Ready signal in the computer results in termin-

ation of the output information Resume signal from the external equipment.

8. Steps 3 through 7 are repeated until the entire block of information is transmitted.

When steps 3 through 7 are completed, the output buffer active signal is dropped.

The computer may sense the condition of an external equipment at any time by transmission of the proper external function code and accompanying Input or Output Ready signal. The status of the Sense Response line then indicates the presence or absence of the condition.

STORAGE SECTION

The storage section of the Control Data 1604 Computer is a large-capacity magnetic core storage system, providing non-volatile, random-access storage for 32,768 48-bit words. A word is transferred to or from a storage location by a single instruction. The operation code of the instruction specifies the type of reference (read or write) and the register which serves as the source or destination. The execution address of the instruction identifies the storage location involved. A read reference is performed by transferring the word at a selected storage address to a specified destination via the X register, and restoring the word at the address. A write reference is performed by clearing the selected storage address, then transferring the word at a specified source to the address via the X register. The cycle time, or time for a complete storage reference, is 6.4 microseconds. The access time, or time from request to delivery of data from storage, is 2.2 microseconds.

The storage section consists of two independent magnetic core storage units each with a capacity of 16,384 48-bit words. All odd storage addresses reference one storage unit; all even addresses reference the other. The storage cycles of the two sections overlap one another to a considerable extent in the execution of a program and result in an effective

cycle time of about 4.8 microseconds.

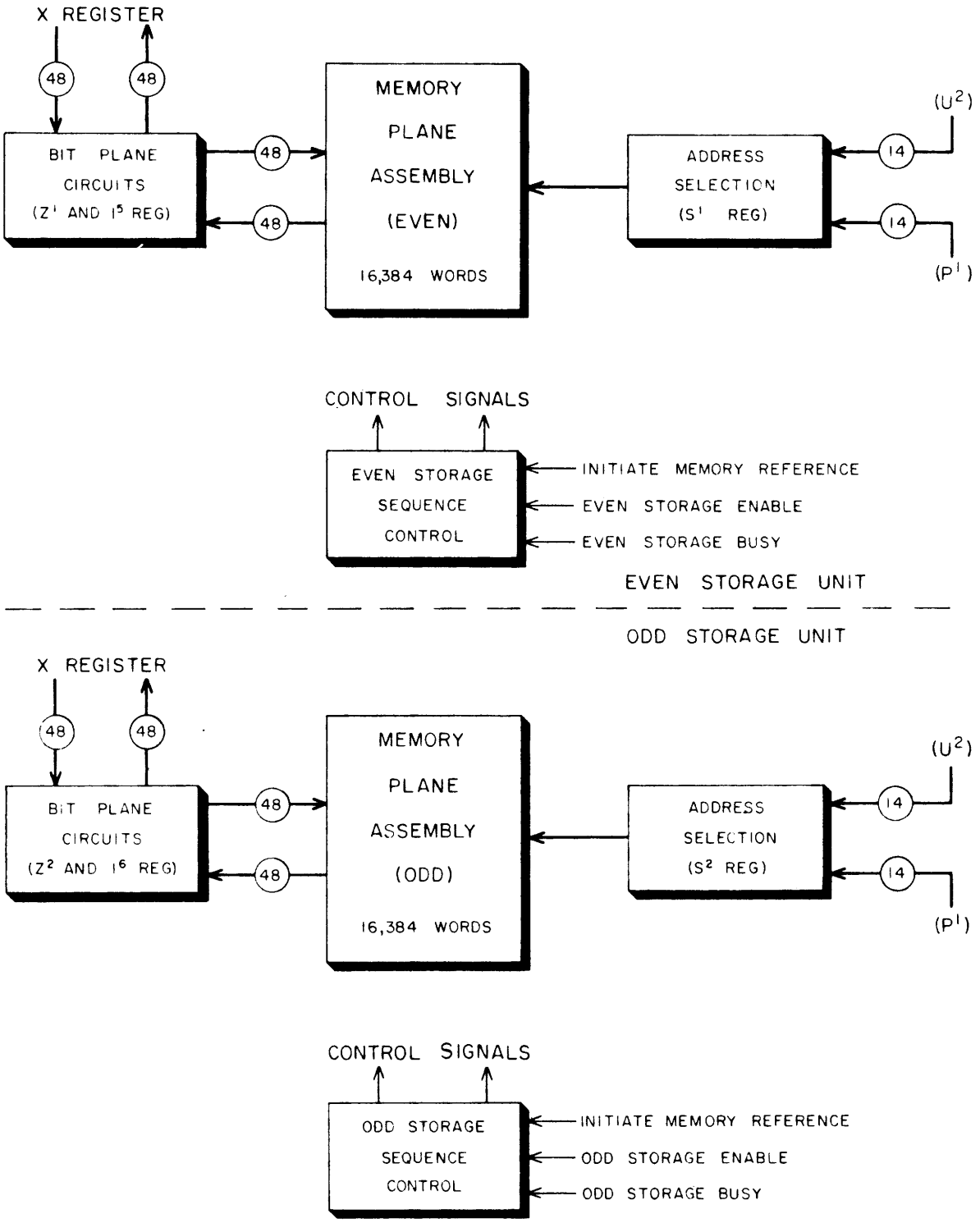
The magnetic core assemblies of the storage section, along with their associated addressing, driving, and control circuitry, are evenly distributed among eight page frame assemblies occupying approximately one-half of each page frame assembly. Data transmissions into the memory registers are channeled through Z^1 and Z^2, the storage restoration registers. The storage address registers, S^1 and S^2, hold the address of the storage location involved in a given cycle of operation. The input-output, arithmetic, and control sections of the computer have independent access to the memory registers by utilizing the appropriate Z and S registers.

General Description

The basic logical divisions of the storage section are shown in Figure III-1. The odd and even storage units are identical, each consisting of four principal parts:

1. Memory Plane Assembly, containing the magnetic core storage elements of the system.
2. Address Selection, which interprets the address from the Control section and selects the specified storage location.
3. Bit-Plane Circuits, which handle the transfer of information between the Memory Plane Assembly and the X register.
4. Storage Sequence Control, which generates the signals that control the storage references.

Memory Plane Assembly. The matrix arrangement of magnetic cores which is the basic storage unit of the storage system is called a memory plane. A memory plane consists of 16,384 cores, in a 128 x 128 array. To employ the coincident-current storage technique,



Logical Divisions of the Storage Section

FIG. III-1

each bit of a word must be stored on a separate memory plane. Thus, the 48-bit words used in the computer are stored by 48 memory planes. This array is called a memory plane assembly.

There are two memory plane assemblies in the storage section, one associated with the odd storage unit and the other with the even storage unit. Both are evenly divided among the page frame assemblies of the computer. Thus, a stack of six "even" and six "odd" memory planes are mounted on each page frame assembly.

Twelve memory planes, stacked one behind another, make up a stack. The memory planes are held together by bolts passing through the four corners of each plane, and each is separated by aluminum spacers. The stack is shielded by an aluminum plate at the back and a plexi-glass plate at the front. A memory plane assembly is made up of six memory planes from each of eight such stacks, or a total of 48 planes. Since each plane stores one bit of a word, and since there are 16,384 cores on a plane the memory plane assembly provides storage for 16,384 48-bit words. Thus, the total storage capacity is 32,768 words.

The stack of chassis 1 stores bits 0 through 5 of each word, the stack of chassis 2 stores bits 6 through 11 of each word, etc. In each case, the memory planes associated with the even memory unit are on the card side of the chassis; the memory planes associated with the odd memory unit are on the wiring side.

Address Selection. Exactly 15 bits are required to identify each of the 32,768 storage locations. The identifications, called addresses, range from 00000 to 77777 inclusive, expressed in octal notation. Two address selection systems, one associated with the odd storage unit and the other with the even storage unit, are provided to interpret the address involved in a storage reference. These two systems select the appropriate odd or even storage location. The lowest order bit of the 15-bit address selects either the odd or even

address selection system. The remaining 14 bits are placed in the S register of the selected system. They are then translated to select a single H wire and a single V wire of each matrix plane of the memory plane assembly.

A storage location is a row of cores at the intersection of a selected pair of H and V wires of a memory plane assembly. It is the function of the address selection circuit to select a discrete storage location for each different address in S^1 or S^2.

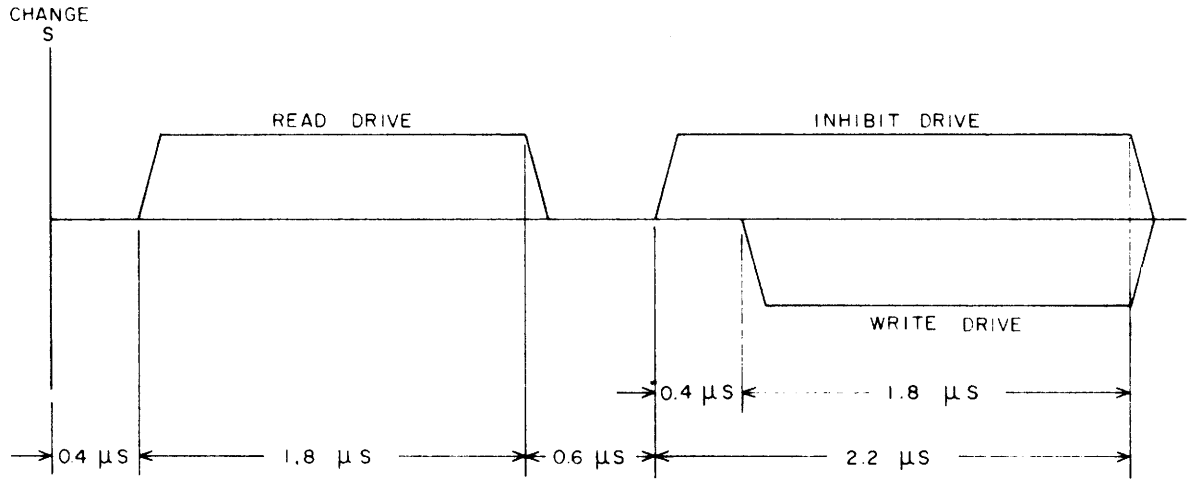
The address selection circuits for the odd and even storage units are identical. Each is composed of four logical systems:

1. The Vertical Drive System, consisting of a translator and eight drivers for each of the storage units on a page frame assembly. This system selects a group of 16 V wires, and supplies the necessary drive current for the desired V wire.
2. The Horizontal Drive System, consisting of a translator and eight drivers for each of the storage units on a page frame assembly. This system selects a group of 16 H wires and supplies the necessary drive current for the desired H wire.
3. The Vertical Diversion System, consisting of a translator and 16 diverters for each of the storage units on a page frame assembly. This system selects the desired V wire from the group of V wires selected by the vertical drive system.
4. The Horizontal Diversion System, consisting of a translator and 16 diverters for each of the storage units on a page frame assembly. This system selects the desired H wire from the group of H wires selected by the horizontal drive system.

The address selection system logically divides each stack into eight vertical and eight horizontal sections. Each horizontal section consists of 16 H wires; each vertical section consists of 16 V wires.

Bit-Plane Circuits. The circuits which handle the transfer of bits between the X register and the memory planes are called the bit-plane circuits. A bit-plane circuit is associated with each of the 48 memory planes of the odd and even memory plane assemblies. Each circuit consists of a stage of the Z register, an input channel from X, four sense channels, and four inhibit-current generators.

Storage Sequence Control. The storage sequence control, in response to initiating signals from the control section of the computer, executes reading and writing operations by generating the signals which control the address selection and bit-plane circuits. The basic pulse sequence for reading and writing is shown in Figure III-2. The first pulse, called the Read Drive, drives the selected core to its "0" state of magnetization. The second pulse, called the Inhibit Drive, allows a "0" bit to be retained in the core by inhibiting (cancelling) the effect of the Write Drive pulse. The third pulse, called the Write Drive, drives the core to its "1" state of magnetization if the Inhibit Drive is absent.



Basic Pulse Sequence for Storage Reference

FIG. III-2

Access Control. The access control, is response to initiating signals from the Control Section of the computer, executes memory references by issuing all the necessary control signals. The access control consists of the Initiate Memory Reference circuit and identical Storage Sequence Controls for the odd and even memory units. The Initiate Memory Reference circuit initiates the reference by entering the address in the appropriate S register and selecting the appropriate odd or even sequence control. The selected sequence control then generates a fixed sequence of control pulses which executes the reference.

Initiate Memory Reference. A sequence chain of the Control Section orders a storage reference by generating the signal Initiate Memory Reference. This signal performs two functions. First, it controls the transmission of the storage address involved in the reference to the appropriate S register. Second, it initiates the basic storage sequence within the memory unit.

The storage references differ from each other with respect to the source of the address involved in the reference. The address may originate from any of three sources, depending upon the sequence which initiates the reference.

Auxiliary Registers

Z Registers. The Z^1 and Z^2 registers provide temporary storage for words during the period they are being written into memory. Each register consists of 48 flip-flop stages.

S Registers. The S^1 and S^2 registers, shown in Figure I-1, hold the storage address during a memory reference. Each register consists of 14 single-rank flip-flop stages and has no properties other than storage. An address is entered into one of the registers by either of three commands. The command $U^2 \rightarrow S^1$ (or S^2) enters the quantity stored in U^2

into S; the command $P^1 \rightarrow S^1$ (or S^2) enters the quantity in P^1 into S; the command Set S^1 (or S^2) inserts one of eight addresses (00000 through 00007) into the appropriate S register during buffer operations.

ARITHMETIC SECTION

Introduction. The arithmetic section of the 1604 computer consists of three registers A, Q, and X. It also includes the logic circuitry which provides for the entry of the registers into the operation of the computer, or which tests the condition in the contents of registers.

Description of the Registers

Accumulator. The Accumulator, or A register, is the principle arithmetic register of the computer. It is used to form the results of logical operations by the process of accumulation; hence its name. The results of most arithmetic operations appear in the storage, or register portion of A, which is closely associated with Q. Together these registers form a double-length accumulator.

In addition to the accumulation function, A can be shifted right or left. The left shift is circular, the highest order bits replacing the lowest order bits as the shift is performed. The right shift extends the sign bit the number of positions shifted, the lowest order bits being discarded. In certain instructions the double-length register, AQ, is shifted, the same shifting properties being used as in the case of A.

The A register initially holds one of the operands for the performance of the arithmetic instruction, and it stores the result from most of these operations.

The control for various conditional instructions is provided by a word stored in A. In the case of the Search instructions, words from memory are compared with (A). The proper relationship causes the instruction to interrupt the normal sequence and proceed in

a pre-planned manner. The A Jump (22) instruction causes the routine to alter its normal sequence by jumping to a different place in the program if (A) satisfies some pre-determined condition.

Shifting a register consists of transmitting the contents of each stage a number of places to the right or left, preserving the initial sequence. The number of positions to be shifted is specified either by the instruction or by a command generated during the execution of an instruction (e.g., shift AQ left one place is required during multiplication). The left shift is a circular shift; the highest bit appears in the least significant register position after each one-stage shift.

Q Register. The Q register is the auxiliary arithmetic register in the computer.

The Q Register consists of 48 stages of double-rank flip-flop storage and associated circuitry. The two ranks operate independently from each other, i.e., the transfer signals causing intercommunication are generated conditionally. Q provides temporary storage for (A). It can be connected directly to A, thus creating a 96-bit register; it holds the multiplier during the multiplication and logical product orders; and it contains a control operand or "mask" for the Masked Equality (66) and Masked Treshold (67) instructions.

Communication to Q is provided only from A. The forced transmission $A^2 \rightarrow Q^1$ causes (A) to be duplicated in Q. The Shift AQ Right command causes the least-significant bit in A^1 to appear in the most significant bit position in Q^2. Shift AQ Left connects A_{47} upper with Q_{00} lower as a path for transmission of the bit in A_{47}. In both of the shift cases mentioned subsequent commands transfer bits passed to Q^2 to Q^1. Operands are placed in Q by the Enter Q (04), the Load Q (16), and Load Q, Complement (17) instructions.

The Store Q (21) instruction transfers (Q) to memory. The address is specified by the instruction.

The particular routing for this transmission is: $Q^1 \rightarrow Q^2 \rightarrow A^1 \rightarrow X^1 \rightarrow Z \rightarrow$ Storage. The contents of Q and A are not destroyed during this transfer.

As an arithmetic register, Q performs the following function: 1) it holds the multiplier during the execution of the multiply instructions; 2) it accumulates the quotient in divide instructions; 3) it contains the least significant 48 bits of the 96-bit products; 4) it holds the residue (the excess bits) following the floating-point routines; and 5) it holds one of the operands for the logical (bit-by-bit) product operations.

Q contains the "mask" or control quantity for the Selective Substitute (43) instruction. This particular instruction substitutes portions of (M_n) into (A) which corresponds to stages containing a "1" in Q.

X Register. The X register is the exchange register or the communication central of the computer. All internal transmissions between the arithmetic section or the input-output section and the rest of the computer are made through X.

All transmissions to X^1 are cleared transmissions, i.e., all stages in X^1 are initially set to "0"; and only the "1's" of the word being sent to X are transferred. Register X^1 is unconditionally cleared during the RNI sequence. X^1 receives full words from A^1, storage, and the input devices. Partial words are sent to X^1 during the execution of the floating-point instructions (11 bits are sent to X^{36} - X^{46} during the assembly of the floating-point result). Partial words are also sent to X^1 when "m" is to be entered into the Arithmetic Section (the 15-bit content of U^2 is sent to X^1 lower during the 04, 10, and 11 instructions).

Transmission paths which accommodate a full word are provided from X^1 to Z^1 and Z^2 (and thence to storage) and to the output registers. Partial words are transferred from X^1 to U^2 (the 11-bit exponent X^{36} - X^{46} of the floating-point operand in X^1 is sent to U^2

during the execution of the floating-point instruction). Partial words are also transferred from X^1 to R^1 (15 bits received by X^1 from storage are sent to R^1 during the Buffer Input-Output sequence). X cannot be addressed directly, i.e., there is no instruction to load X.

External communication of X^2 with other sections of the computer are: 1) the forced transmission $P^1 \rightarrow X^2$ transfers the current instruction address plus one in P^1 to X^2 for storage in memory during the Return Jump routines, 2) X^2 provides the augend inputs of the accumulator for the Add X^2 to A^1 command; and 3) X^2 supplies the word to the Output registers or Function register in the Output section of the computer.

As an arithmetic register, X enters into all computations. It holds the augend, subtrahend, multiplicand, and divisor in the corresponding operations. It stores the result of the Logical Product Q and X routine.

As a central exchange register, X is used in most transmissions to and from memory. Transmissions to storage are carried out by first setting up the word in X^1 and then performing the transfer.

Control Registers used in Arithmetic Operations

Certain arithmetic operations "borrow" elements of the Control Section to effect their completion. In the case of the multiply, divide, and shift instructions a control quantity is placed in R to govern the operation. In the case of the floating-point instructions the arithmetic qualities of the U^2 register are used.

The R register holds the step control quantity during all multiplications and divisions. These operations proceed as a series of additions (or subtractions) and shifts. The control word, which is the number of such additions and shifts to be performed, is placed in R and is then reduced by one each time the partial product or quotient is shifted. Thus when R = 0 the operation is concluded.

Floating-point operation separates the individual operands involved into two portions, arithmetic being performed with the corresponding portions independently. The arithmetic involving one of these groups (the exponents) is performed in the U^2 accumulator with the U^2 and R registers providing initial storage for the exponents. The floating-point exponent is eleven bits (including its sign bit) in length.

CONTROL SECTION

The control section of the computer directs the operations required to execute instructions and to exchange data with external equipment. In addition, it establishes the timing relations required to perform the operations in the proper sequence. The control section can be considered as including the following specialized control areas: main or overall control, storage control, arithmetic control, and input-output control. Main control, in addition to performing many operations itself, initiates action in the other specialized control areas. These produce, on a somewhat independent basis, the commands to carry out a special operation.

The major elements of main control include the sequences, several networks of flip-flops and single inverters which sense and store static conditions, and the registers U, P, R, and B^1 through B^6.

The execution of instructions and the exchange of data with external equipment are accomplished by many unit operations called commands. A command, which is issued by a sequence, causes one action to occur, e.g., the transmission of the contents of a register to another register or the setting of a control flip-flop. In the execution of instructions the issuing of commands is controlled directly or indirectly by the instruction word in the program control register. Commands may also be conditioned by the presence or absence of specified conditions in some part of the computer.

The function of the control section may be thought of as sensing or determining the operations that are required and producing commands in an order suitable for the performance of such operations.

Program Control Register

The program control register called the U Register holds the 48-bit instruction word during the execution of the two 24-bit instructions. All operations that are necessary to execute an instruction are governed by the contents of this register.

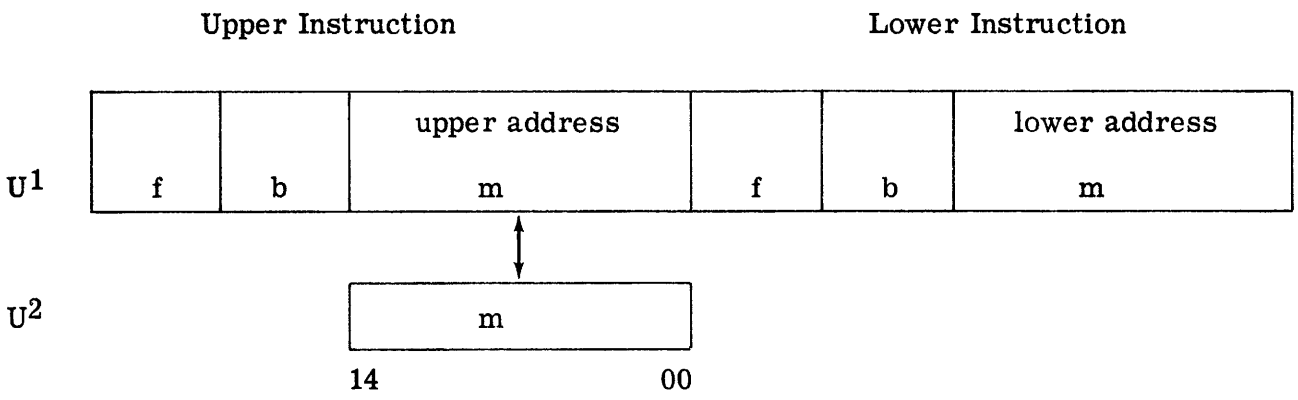
An instruction is a 24-bit quantity consisting of three designators which are arranged as follows:

Operation Code 6 bits	Designator 3 bits	Execution Address 15 bits
f	b	m

Each of the 62 instructions has a unique six-bit operation code, f, which designates the instruction. The translation of f establishes the condition required within the control section for the execution of the instruction. The designator portion of an instruction usually designates the index register, B, the contents of which are to be added to the execution address, m. For some instructions the designator is used to select jump conditions or input-output channels. The execution address of some instructions is a base address quantity that is modified by the addition of (B^b) to yield the actual address of the instruction operand. Other instructions use the execution address as the operand, or shift count.

The U register consists of two ranks of flip-flops. Rank U^1, 48 bits in length, stores an instruction word during the execution of the two instructions. Rank U^2, 15 bits in length, is a small subtractive accumulator. As shown below, transmission paths connect U^2 to the

execution address portion of the upper half of U^1.



A 48-bit instruction word is read from memory and entered in U^1. Execution of the upper instruction occurs first. Following this, the lower instruction is then transmitted to the upper half of U^1 so that it can be executed. Thus the instruction currently being executed is always located in the upper half of U^1.

The primary function of U^2 is to modify the execution address, m, of the instruction in the upper half of U^1. This modification consists of the addition of (B^b) to m. In preparation for modification, the b designator of the upper instruction is translated. The translation specifies the B register, the contents of which are to be added in the modification. (B^b) is then transmitted to R. After transmitting m from U^1 to U^2, (R) is added to (U^2). For most instructions, the modified execution address in U^2 specifies the location of the operand in memory.

Operation Code. This 6-bit code specifies an instruction and controls the operation of the computer during the execution of the instruction. Of the 64 possible values of this code, 62 specify instructions. Codes 00 and 77 are interpreted as fault conditions that halt computation.

Prior to the actual execution of the instruction designated by the value of f, this 6-bit code is translated. The results of the translation go to the various sections of the machine to condition the occurrence of the commands which will actually carry out the required operations. The operation code translator is a network of single inverters which samples the upper six bits of U^1. Outputs from the translator are of two kinds: those that indicate a unique code and those which indicate a combination or group of codes. The latter type of translation is used to condition commands which must occur in the execution of several instructions.

Designator. The primary function of the designator portion of an instruction is the specification of the B register, the contents of which are added to the execution address, m. However, for several instructions the designator specifies one of several conditions or ways of executing the instruction.

The designator is translated completely, and the translation of a specific value is used to gate operations that are appropriate for that value and the current function code.

Execution Address. The lower 15 bits of the instruction usually functions in the specification of the address of the operand, i.e., the address is $m + (B^b)$. However, the execution address may be finally interpreted in the following ways depending upon the operation code and the designator interpretation:

- When the designator is translated as b and b = 0 or $(B^b) = 0$, the execution address is unmodified and 1) specifies the location of the operand in storage, or 2) is interpreted as the operand, or 3) is the shift count.

- When b = 1-6 and $(B^b) \neq 0$, the execution address is modified by the contents of B^b,

and this sum 1) specifies the location of the operand in storage, or 2) is interpreted as the operand, or 3) is the shift count.

- When b = 7, the execution address specifies the storage location of the address of the operand.

- When the designator is translated as j, the conditional jump designator, the execution address specifies the storage location of the next instruction to be executed.

- For External Select and Sense Functions, the execution address is translated for channel selection (3 bits), equipment selection (3 bits), and mode of operation (9 bits).

The transmission of the execution address instruction in U^1 to U^2 occurs in the execution of all instructions. If it is to be modified, (B^b) is transmitted to R and then (R) is added to U^2. This addition makes use of the fact that U^2 is a 15-bit accumulator. When the operand is to be procured from or sent to memory, (U^2) is transmitted to S^1 or S^2. When (U^2) itself is to be used as the operand it is transmitted to R or X.

The U^2 Accumulator

In addition to the usual features of a flip-flop register, U^2 is a 15-bit subtractive accumulator that provides for the addition of (R) to the content of U^2. Except for the reduction in length, this accumulator is similar in structure to the 48-bit accumulator, the A register. In addition to the use of the U^2 accumulator in modifying the execution address, it plays an important part in several other operations. Arithmetic operations on exponents in floating-point instructions are performed by the use of U^2. Buffer operations and transfer instructions employ U^2 in handling address words.

Index Registers

Each of the six index registers B^1 through B^6 provides storage for quantities which are used in a variety of ways, depending upon the instruction. In the majority of instructions the B registers hold quantities used in the modification of the execution address. In this case (B^b) is added to m to form M, the actual execution address. For search instructions (64-67), (B^b) is used instead to indicate the number of items to be searched.

The B registers have no provision for arithmetic operations. But when such an operation is required on an index quantity, (B^b) is entered in R or U^2 and the operation performed there. Subsequently the result is returned to B^b.

Address Buffer Register

The 15-bit address buffer register, R, has provisions for counting and complementing as well as storage. As a counter it operates subtractively. R is suitably connected to the U^2 accumulator for the addition of its contents to U^2.

All transmissions to and from the B registers are via the R register. Modification of m, the base execution address of the current instruction word, is accomplished by adding the contents of R to the contents of U^2. In the execution of shift instructions, R acts as the shift counter. The integer and fractional multiply and divide instructions employ R to keep a record of the number of partial multiplications and divisions which remain to be performed.

Floating-point instructions use R to perform arithmetic operations on one of the two exponents. During buffer operations R is used for incrementing the current buffer address and for comparing it with the terminal address.

Program Address Register

The program address register, or P register, holds the address from which each

instruction word is obtained. After the execution of both the upper and lower instructions of this word (or, if the decision is made to skip the lower instruction) the quantity in P is advanced by one to generate the address from which the next sequential instruction word is obtained. Thus the P register is a counter.

The initial address of a sequence of instruction words may be entered into P normally at the console, or it may be entered during the execution of a program by jump instructions. Such instructions always transmit a new program-address quantity to P. If the jump instruction is a return jump, that is, when j = 4-7, then the previous contents of P are stored, thereby permitting the return to the sequence of instructions from which the jump was made.

Control Sequences

The execution of an instruction or the exchange of data with external equipment is divided into lesser operations called "sequences." These sequences control the initiation and timing of commands which actually execute the instructions, buffers, etc.

Instructions are executed by the performance of the read next instruction (RNI) sequence and one other. For example, instruction 14, ADD, is executed by the RNI sequence followed by the read operand (RO) sequence. The RNI sequence enters the 24-bit instruction word in the upper half of U^1. The read operand sequence obtains the operand from storage and performs the actual addition.

The auxiliary sequence used for input-output control is made up of three sub-sequences. One handles the execution of buffer operations, i.e., the exchange of data with an external equipment. A second provides for the operation of advancing the real-time clock. The third handles the recognition of an interrupt signal and initiation of the routine that responds to the interrupt.

IV. INSTALLATION AND OPERATION

OPERATING CONTROLS

The 1604 control console contains all of the indicators and switches necessary to operate the central computer. The contents of the operational registers are indicated with a projection display for each octal digit in the register. The three bits of information in each octal digit are translated in the console and a numeral (0 through 7) is projected on the control panel. Three push-button switches are located immediately below each octal display to provide the operator with a manual input to the registers. A single clear button is located at the end of each register panel to clear the register contents to zero.

A panel of key levers switches is located below the indicator panel on the operator console. These switches are used to control the various modes of computer operation. The function of each key lever switch is described in the following section beginning with the rightmost key position.

Start Step Key

This key lever switch controls the operation of the central computer. It has a momentary contact in both the up and the down position. Normal high-speed operation is initiated by the up (start) position. The key must be returned to the neutral position before a second start is possible. A single instruction may be executed by the down (step) position. The computer program executes one instruction and stops after the next instruction has been read into the program control register. Repeated "stepping" in this manner allows a step by step examination of the program.

A force stop may be made at any time during the normal high-speed operation of the computer by depressing the start-step key to the step position. This action causes the control to revert to a step mode of operation, and the program stops at the end of the current instruction.

Master Clear Key

This lever switch clears all registers and control mechanism prior to the beginning of a new program. It is a momentary switch in both positions. The up position of the key clears all peripheral equipment control systems. The down position clears all central computer registers and controls.

Selective Stop Keys

Three lever switches are used to specify selective stop conditions. These switches are sensed by the Selective Stop instruction (76) in the computer program. The key switches lock in the up position and are momentary in the down position. The same function is performed in either position. The program stops as the appropriate (76) instruction is read into the program control register. The jump is not executed in the 76 instruction unless the program is restarted.

Selective Jump Keys

Three key lever switches are used to specify selective jump conditions. These switches are sensed by the Selective Jump instruction (75) in the computer program. The key switches lock in the up position and are momentary in the down position. The same function is performed in either position.

Break Point Address

A switch is provided on the operation control panel for setting break-point addresses. This five-octal-digit quantity is continually compared with the program address during operation of the computer and causes the program to stop if the addresses are equal. Stopping occurs only during the instruction acquisition. This provision permits the program to be executed at high speed to a preset address, of interest in debugging a program.

The break-point address is entered on the digit switch in octal code. With the break-point set, the computer stops just before executing the upper instruction at the storage location specified by the break-point address.

Storage Test Keys

Two keys switches are provided on the control panel for test of the central computer storage system. One key provides a marginal check of the storage sense amplifiers. The up position of this switch raises the signal threshold level in the sense amplifiers. The down position lowers the signal threshold in the storage sense amplifiers.

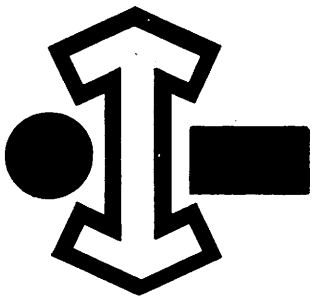
The second test key provides test modes of storage operation. The up position of this key blocks the advance of the program address register during the execution of a program. This condition causes a program step to be endlessly repeated. The down position of this test key causes the storage to sweep continuously through all 32,768 storage locations.

Power On-Off Switch

A pair of push-button switches are provided on the control console for power on-off. This is a remote control of the motor-alternator which provides the main power for the computer.

INSTALLATION

The Control Data Corporation provides installation and checkout of the 1604 Computer System at the customer's premises Information which will aid the customer in selecting and preparing a suitable location for the computer and in providing the necessary facilities --as well as complete information concerning the electrical cables and their interconnections will be found in the 1604 Installation Manual.



CONTROL DATA CORPORATION
501 Park Avenue, Minneapolis 15, Minnesota