SAXON AND PLETTE

Programming the IBM 1401:

A Self-Instructional Programmed Manual

PROGRAMMING THE IBM 1401:

A Self-Instructional Programmed Manual

PROGRAMMING THE IBM 1401:

A Self-Instructional Programmed Manual

JAMES A. SAXON
Saxon Research Corporation

WILLIAM S. PLETTE
United Research Services

PRENTICE-HALL, INC.
ENGLEWOOD CLIFFS, N.J. 1962

PRENTICE-HALL INTERNATIONAL, INC. - London
PRENTICE-HALL OF AUSTRALIA, PTY., LTD. - Sydney
PRENTICE-HALL OF CANADA, LTD. - Toronto
PRENTICE-HALL FRANCE, S.A.R.L. - Paris
PRENTICE-HALL OF JAPAN, INC. - Tokyo
PRENTICE-HALL DE MEXICO, S.A. - Mexico City

Library of Congress Catalog Card Number: 62-20615

Printed in the United States of America

73031-C

To

Tottie

and

Bernice

INTRODUCTION

This self-instructional work book has been developed to teach the beginner to program for the IBM 1401 computer. Programming simply means the ability to translate into language understandable by the computer, whatever we wish to have accomplished by the computer.

There is no time limit or speed limit imposed on the course. Each student will progress at his own rate of speed. If an area is not completely clear the first time through, it should be reviewed until there are no further questions in the mind of the student.

The correct answer to every problem is to be found on the back of the page containing the problem. There is nothing to keep the student from cheating by looking at the correct answer before attempting to work the problem except the realization that he will not learn to program if he does this.

For best results, it is suggested that this book be studied for not more than two hours at a time and not more than two such (two hour) sessions a day. There is a great deal of material to be assimilated and attempting to push through too fast will cut down retention of material covered.

It must also be understood that completion of this work book will not qualify the student as an expert programmer. It will teach him the fundamentals of programming for the IBM 1401. He will have the basic tools of programming at his finger tips but only practical experience as a working programmer can develop the knowledge and skill required to be considered an expert.

As time goes on, computer manufacturers will continue to make advances and some of the limitations listed in this text will be exceeded, but as long as the 1401 Computer is used, the general information and programming methodology will be applicable.

TABLE OF CONTENTS

GENERAL INFORMATION

Automatic Data Processing is the self-controlled sequence of actions by a computer, each action depending on another and requiring no human intervention to complete the sequence.

A Stored Program System is one that stores its instructions internally. A sequence of instructions to solve a particular problem is called a program. The individual instructions are called program steps. These program steps are converted from writing to punched cards, which are loaded into the computer (placed into the Memory unit of the 1401). When data is fed into the computer, the stored program acts on the data to produce the desired result.

> As a very simple example, let us assume that we wish to process a large number of punched cards and, among other things, every time we find a card with the digit "5" in column one, we will want to print out the contents of this card in a management report.
>
> Our stored program will contain an instruction that says, "Look for a "5" in column one." "If a "5" is found, this card is to be processed to produce a printed report."
>
> As data cards are fed into the 1401 for processing, each card will be examined for a "5" in column one and the contents of the "5" cards will be printed as desired.

Nearly all data processing systems incorporate the following five functions:

1. INPUT - This refers to anything that enters the system.

2. STORAGE - This refers to the data after it is read by the machine. It is held in storage until it is ready for use.

3. CONTROL - This allows the machine to process the various steps specified by the program.

4. ARITHMETIC - This refers to the ability of the machine to perform arithmetic operations (i.e., add, subtract, multiply and divide).

5. OUTPUT - This refers to anything turned out by the machine.

Program planning is one of the most important functions in programming. When it has been determined that a particular function, or job, is to be handled by a computer in preference to manual methods of handling, the following steps must be taken:

1. Analysis of the job -- how it will be handled and what specifically will be involved.

2. Sequencing the steps to be taken to accomplish the basic purpose.

3. Writing the instructions (program steps).

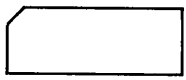
4. Determining which areas of storage will be used for various purposes.

The ability to plan a job for computer application requires a knowledge of the machine components and functions and the instructions which cause the functions to occur. The greater the knowledge, the easier it is to plan and execute the required job.

Flow charting: Before writing machine instructions, it is usual to flow chart the necessary steps to be taken. This has the advantage of proving the logic of the application since a flaw in logic will show up very quickly on a flow chart. It has the additional advantage of providing a chart from which machine coding (writing instructions for the program) may be accomplished with a minimum of error.

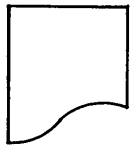
Symbols and signs have been fairly well standardized for flow charting purposes. The primary reason for this standardization is so that others may easily read and interpret a programmer's flow chart. Some of the more commonly used signs and forms are shown on the following page.

COMMONLY USED FLOW CHART SYMBOLS



IBM PUNCH CARD - Denotes input or output data in 80-column cards

MAGNETIC TAPE - Denotes input or output data on magnetic tape

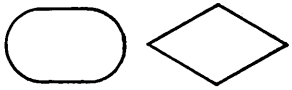


PRINTED OUTPUT - Denotes output in a printed form

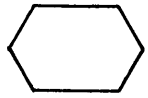
START - Denotes the point at which a program begins

HALT - Denotes the point at which a program ends

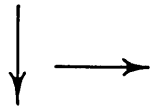
PROCESSING BLOCK - Fill with a brief description of each discrete process



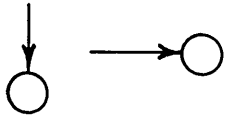
DECISION BLOCKS - Denotes the point at which a program branches due to a decision



SUB-ROUTINE- Represents a "program" within the main program which may be flowcharted in detail elsewhere



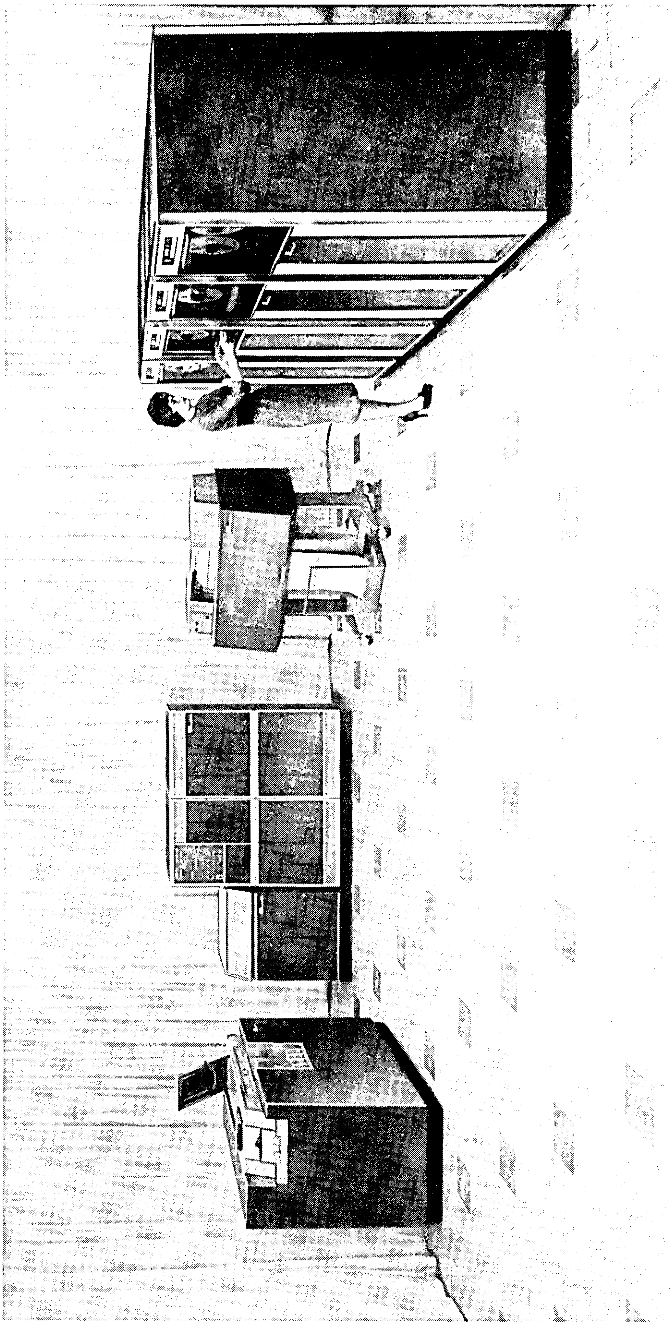
DIRECTION OF FLOW - Used to connect the other symbols



CONNECTORS - Used to show how the sections of the flowchart connect together

This is IBM's solid-state 1401 data processing system. It is available in four basic models: punched card, magnetic tape, RAMAC (random access disk storage), and RAMAC/tape.

The 1402 card read punch (left) can read a maximum of 800 IBM cards a minute. The 1401 processing unit performs all the arithmetic, logical and control functions. The IBM 1403 printer prints numbers and letters at a basic speed of 600 lines a minute. High-speed 729 magnetic tape units (right) are able to read as many as 62,500 characters of information into the system in a second.

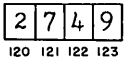


UNIT I

Lesson 1

COMPUTER STORAGE: Storage, or memory, permits the computer to retain information, in a readily accessible form, until it is needed. The IBM 1401 may have 1400, 2000, 4000, 8000, 12000 or 16000 positions or cells of high speed memory. Each memory cell is capable of storing, subject to recall, any one of the decimal numeric characters (0 through 9), any alphabetic character (A through Z), or any one of twenty special characters such as a period (.), comma (,), etc. Each storage position is numbered to simplify reference to it, starting with 000 and continuing to 1399, 1999, 3999, and up, depending on the size of memory of the IBM 1401 being referenced. The number representing a storage position permits the programmer to address any specific storage position he may choose.

EXAMPLE: We wish to place the number 2749 into storage at positions 120 through 123. Assume that each position may be represented as a little box, large enough to hold one character, and there are as many boxes as there are storage positions. After the correct instructions have been given to, and executed by, the computer the result may be represented as follows:



PROBLEMS: Write your answers to the following questions in the space provided, referring as necessary to the text. After you have written and checked your answers, refer to the correct solutions on the following page. Do not proceed to the next lesson until you fully understand the questions and their correct solutions.

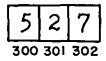
1. Place the digits 527 into storage at positions 300 through 302.	1.
2. Place the letter P into storage at 512.	2.
3. Place the word HOLD into storage at positions 101 through 104.	3.
4. If there are 8000 positions of storage they are numbered from (a) to (b).	4. (a)_____ (b)_____
5. When a programmer calls for a character from a specific cell in storage, he is _____ that memory position.	5. _____

UNIT I

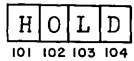
Lesson 1

NOTES AND ANSWERS

ANSWERS

1. 

2. 

3. 

4. (a) 000

   (b) 7999

5. Addressing

REMINDERS

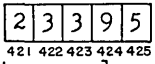
Each digit must be assigned to an individual memory position, each of which has an address number. Thus 5 goes into position 300, 2 into 301, 7 into 302. Alphabetic characters may be stored in individual, addressable memory cells in the same fashion.

The first memory position is always numbered zero; therefore the last position must end in nine -- or one less than the memory size.

Just as a post office box number serves as a simple means of addressing a particular box, the storage number serves as a means of addressing memory cells.

COMPUTER WORD: A computer word, as the term is used in connection with the IBM 1401, refers to a single character or a group of characters that represent a unit of information. An IBM 1401 word is not limited to a specific number of storage positions as in many computers. Computer words may be as long or as short as is actually needed to contain the information in consecutive storage positions. The position to the extreme left is called the high-order position of the word, and the position to the extreme right is called the low-order position of the word. Also, it must be remembered that every storage position in every word is addressable.

EXAMPLES: The number 23395 occupying storage locations 421-425, might be considered a word if it is a single unit of information such as, "total number of employees". It would be pictured in memory as follows:

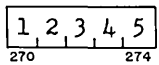


The word STEP occupying storage locations 001-004 would be considered a computer word pictured as follows:

Position 421 in the first example and 001 in the second are the high-order positions.
Position 425 in the first example and 004 in the second are the low-order positions.

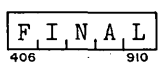
PROBLEMS: Write your answers to the following questions in the space provided, referring as necessary to the text. After you have written and checked your answers, refer to the correct solutions on the following page. Do not proceed to the next lesson until you fully understand the questions and their correct solutions.

6. Write the position numbers of the high-order characters in the following examples:

(a) 

(b) 

7. Write the position numbers of the low-order characters in the same examples.

(c) 

8. A single character or group of characters that represent a unit of information, is called a_____.

6.(a)_____
(b)_____
(c)_____
7.(a)_____
(b)_____
(c)_____
8._____

UNIT I

Lesson 2

NOTES AND ANSWERS

ANSWERS	REMINDERS
6. (a) 270 (b) 102 (c) 906	The high-order position of the word is always the left-most digit or position.
7. (a) 274 (b) 103 (c) 910	The low-order position of the word is always the right-most digit.
8. WORD	A word may occupy as many consecutive storage positions as are necessary to contain a unit of information.

WORD MARKS: The IBM 1401 is a variable word length computer. This means that a computer word may be as long or as short as needed to contain a unit of information. The word mark makes variable word lengths possible. A word mark is not a character in itself, but is associated with the high-order (left-most) position of a word. The word mark may be shown symbolically by underlining the character with which it is associated. The word mark tells the computer that the position so designated is the beginning of a word. When the computer senses another word mark, as it scans each character, it recognizes that a new word is beginning and that the previous word has ended.

EXAMPLE:

5	9	2	5	S	E	N	D	1	5	6	H	O	L	D	T
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116

The storage positions shown above contain five computer words. The word marks in positions 101, 105, 109, 112, and 116 indicate the beginning of each word.

PROBLEMS: Write your answers to the following questions in the space provided, referring as necessary to the text. After you have written and checked your answers, refer to the correct solutions on the following page. Do not proceed to the next lesson until you fully understand the questions and their correct solutions.

9.

1	2	3	4	3	2	1	0	5	H	2	3	5	S	T	U	V	4
200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217

Indicate the high- and low-order positions of each word shown in the storage cells above.

9. High Low

(a)____ ____

(b)____ ____

(c)____ ____

(d)____ ____

(e)____ ____

(f)____ ____

10. Symbolically represent the word PRICE in storage positions 501-505, placing the word mark in the proper position.

10.

11. With respect to computer words, the IBM 1401 is a ___(a)___ ___(b)___ ___(c)___ computer.

11.(a)_____

(b)_____

(c)_____

UNIT I

Lesson 3

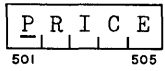
NOTES AND ANSWERS

ANSWERS

	High	Low
9.(a)	200	202
(b)	203	206
(c)	207	210
(d)	211	215
(e)	216	216
(f)	217	217

REMINDERS

The word mark always indicates the high-order position. The last two words are only one character in length, therefore the single character in each case is both the high- and low-order position.

10. 

The word mark should be associated with the P or high-order position.

11.(a) VARIABLE

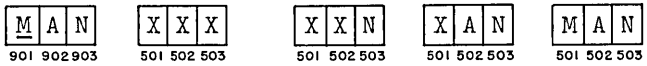
(b) WORD

(c) LENGTH

Since a word may be of any length in the IBM 1401, it is called a variable word-length computer.

DATA WORDS: Data, as we refer to it in this text, means any computer word or group of words which are to be operated-on by the computer in some way. A data word is frequently referred to as a data field, meaning a field of characters which might be descriptively referred to as a NAME, EMPLOYEE NUMBER, PAY-RATE, or TOTAL HOURS. A data field is always addressed by its low-order (right-most) position. Whenever the computer performs an operation on a data field, it begins with the low-order position and proceeds to the left, character-by-character, up to and including the position containing the word mark (W/M).

EXAMPLE: If the word MAN was in storage at positions 901-903 and we wished to move it to positions 501-503, we would instruct the computer to: "MOVE 903 to 503," addressing the low-order positions of both words.



First the N would move from 903 to 503; then the A would move from 902 to 502; finally, the M would move from 901 to 501 and the word mark (W/M) in 901 would stop the move operation.

PROBLEMS: Write your answers to the following questions in the space provided, referring as necessary to the text. After you have written and checked your answers, refer to the correct solutions on the following page. Do not proceed to the next lesson until you fully understand the questions and their correct solutions.

(a) F L I P (225 ... 228) (b) 1 3 5 7 6 (004 ... 008)

(c) 3 T 7 (202 ... 204) (d) S T O P (205 ... 208)

12. Write the storage position number to be addressed in each of the above.

13. Write the storage position in which the word mark (W/M) should be placed in each of the above.

12. (a)_____
(b)_____
(c)_____
(d)_____

13. (a)_____
(b)_____
(c)_____
(d)_____

UNIT I

Lesson 4

NOTES AND ANSWERS

ANSWERS	REMINDERS
12. (a) 228	Data words or fields are addressed by their right-most or low-order positions.
(b) 008	
(c) 204	
(d) 208	
13. (a) 225	W/M's are associated with the left-most or high-order position.
(b) 004	
(c) 202	
(d) 205	

INSTRUCTION WORD: The computer performs operations by following instructions prepared by a programmer. Each instruction specifies a specific operation and is contained in a computer word. The number of characters in an instruction word may vary from one to eight including up to four parts, depending on the nature of the instruction. The four parts are defined as follows:

OPERATION CODE	1 digit	A code defining the operation to be performed. This code must always be present in every instruction and must have a word mark.
A - ADDRESS	3 digits	The address of the low-order character in a data field which is to be operated-on. In a MOVE instruction, for example, the field from which data will move.
B - ADDRESS	3 digits	The address of the low-order character in the second field, if any, to be operated-on. In a MOVE instruction, for example, the field to which data will move.
DIGIT MODIFIER	1 digit	This character, if present, usually modifies the OPERATION CODE, making it possible to use the same OP CODE for several different but similar processes.

EXAMPLES:

OP	A			B			d
M	0	0	6	5	5	6	

An instruction to move data from position 006 to position 556.

M is the OPERATION CODE for MOVE. 006 is the low-order address of the data to be moved. 556 is the low-order address to which the data is to be moved. This instruction does not require a data modifier, so none is present.

PROBLEMS: Write your answers to the following questions in the space provided, referring as necessary to the text. After you have written and checked your answers, refer to the correct solutions on the following page. Do not proceed to the next lesson until you fully understand the questions and their correct solutions.

14. Instructions may have (a) to (b) characters.

15. The W/M is always associated with the (a) (b).

16. There may be up to (a) parts to an instruction, but never less than (b).

14.(a)_____
(b)_____

15.(a)_____
(b)_____

16.(a)_____
(b)_____

UNIT I

Lesson 5

NOTES AND ANSWERS

ANSWERS	REMINDERS
14. (a) 1 (b) 8	The number of characters in an instruction depends upon the kind of instruction and what it is to do.
15. (a) OPERATION (b) CODE	The operation code must always be present and must have a W/M associated with it.
16. (a) 4 (b) 1	

INSTRUCTION USE: All four parts of an instruction need not be used. Six combinations are possible:

1. OPERATION code, only. (OP)
2.
2. OPERATION CODE and DIGIT MODIFIER. (OP, d)

3. OPERATION CODE, A-ADDRESS. (OP, A)

4. OPERATION CODE, A-ADDRESS and DIGIT MODIFIER.(OP,A,d)

5. OP CODE, A-ADD, and B-ADD. (OP, A, B)

6. OP CODE, A-ADD, B-ADD, and DIGIT MODIFIER (OP,A,B,d)

SPECIAL NOTE: A discrepancy may have occurred to you. Since A and B-Address spaces are 3-digit parts of the instruction and storage addresses may be four or five digits long -- how can addresses higher than 999 be placed in the A or B parts of an instruction? This is accomplished by substituting a special code for the first two or three digits of the address. This text will use examples for a 4000 position memory only - therefore a table of codes up to 3999 follows:

CODES FOR ADDRESSES IN STORAGE

Actual Address	3-Digit Address	Actual Address	3-Digit Address
000 to 999	000 to 999	2500 to 2599	N00 to N99
1000 to 1099	‡00 to ‡99	2600 to 2699	Ø00 to Ø99
1100 to 1199	/00 to /99	2700 to 2799	P00 to P99
1200 to 1299	S00 to S99	2800 to 2899	Q00 to Q99
1300 to 1399	T00 to T99	2900 to 2999	R00 to R99
1400 to 1499	U00 to U99		+ +
1500 to 1599	V00 to V99	3000 to 3099	000 to 099
1600 to 1699	W00 to W99	3100 to 3199	A00 to A99
1700 to 1799	X00 to X99	3200 to 3299	B00 to B99
1800 to 1899	Y00 to Y99	3300 to 3399	C00 to C99
1900 to 1999	Z00 to Z99	3400 to 3499	D00 to D99
	- -	3500 to 3599	E00 to E99
2000 to 2099	000 to 099	3600 to 3699	F00 to F99
2100 to 2199	J00 to J99	3700 to 3799	G00 to G99
2200 to 2299	K00 to K99	3800 to 3899	H00 to H99
2300 to 2399	L00 to L99	3900 to 3999	I00 to I99
2400 to 2499	M00 to M99		

Codes for positions higher than 3999 will not be needed for this text but may be obtained from the IBM 1401 Reference Manual.

In the Codes for Addresses in Storage on page 11, you will notice that the third digit to the left contains a special character or alphabetic character. This character is placed in the hundreds position.

EXAMPLE:

‡	9	9
HUNDREDS	TENS	UNITS

= 1099

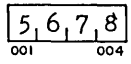
You will have noticed that the conversion from 2600-2699 is Ø00 to Ø99. The Ø symbol is used to designate the alphabetic O. If there is no slash through the O, it is considered to be a zero.

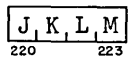
Examples of the use of these special codes:

1. S57 = 1257
2. M39 = 2439
3. Ø00 = 2600
4. /22 = 1122
5. D7‡ = 347‡
6. I99 = 3999
7. ‡015 = 3015
8. R09 = 2909
9. Z11 = 1911
10. 55 = 1055

17. Place the word AREA into storage at positions 901-904.

18. Write the position numbers of the high and low order positions in the following examples:

a. 

b. 

c. 

19. In the above examples, write the position numbers where word marks should be placed.

20. In the above examples, write the storage position numbers to be addressed.

21. If a word is to be moved from one place in storage to another, the (a) (b) stops the move operation.

22. Name the four parts of an instruction.

23. The one part of an instruction that must always be present is the (a) (b).

24. Write the following storage addresses in three digit code:

a. 0049	d. 2401
b. 1357	e. 2900
c. 2299	f. 3750

17.

18. high order
a._____
b._____
c._____
low order
a._____
b._____
c._____

19.a._____
b._____
c._____

20.a._____
b._____
c._____

21.a._____
b._____

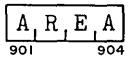
22.a._____
b._____
c._____
d._____

23.a._____
b._____

24.a._____
b._____
c._____
d._____
e._____
f._____

UNIT I QUIZ

ANSWERS

17. 

REMINDERS

Refer to page 1

18. high order low order

a. 001 a. 004

b. 220 b. 223

c. 601 c. 602

Refer to page 3

19.a. 001

b. 220

c. 601

Refer to page 5

20.a. 004

b. 223

c. 602

Refer to page 7

21.a. word

b. mark

Refer to page 7

22.a. Op. Code

b. A-address

c. B-address

d. digit modifier

Refer to page 9

23.a. Op.

b. Code

Refer to page 9

24.a. 049 d. M01

b. T57 e. R00

c. K99 f. G50

Refer to page 11

UNIT II

Lesson 7

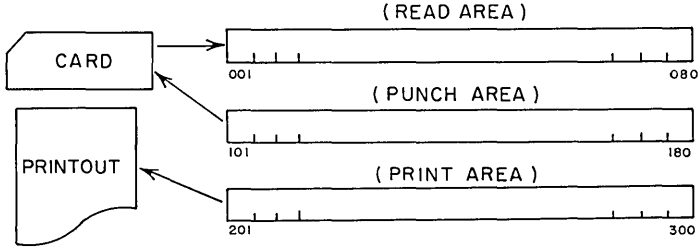
RESERVED AREAS OF STORAGE: We have described the memory as being like a set of post office boxes, each capable of holding a single character. Some of these boxes are reserved for specific purposes. They are:

Read Area: Positions 001 through 080 are reserved for information coming into memory from punched cards. A "READ" instruction directed to the card reader will cause that machine to extract coded information, up to 80 digits, from a standard IBM punched card and place the digits in corresponding memory storage positions. The digit in column one goes to storage position 001; column two goes to storage position 002, etc.

Punch Area: Positions 101 through 180 are reserved for information to be punched into 80 column IBM cards. A "PUNCH" instruction will cause all information stored in positions 101 to 180 to be properly coded and punched into an IBM card.

Print Area: Positions 201 through 300 are reserved for printing. Thus up to 100 characters will form a single-line to be printed on the printer attached to the IBM 1401. A "WRITE" instruction addressed to the printer will cause all the characters in positions 201 through 300 to be printed as a single line of information. (Some models of 1401 may be equipped with 132 reserved positions but the principle is the same. The printer will print all 132 positions on a single line.)

EXAMPLES:



PROBLEMS:

25. Information from a punched card going into memory will always go into positions (a) through (b). — 25.(a)_____ (b)_____

26. Each printed line will normally be _____ characters long. — 26._____

27. If a card is desired as output from the machine, the data must first be moved to positions (a) through (b). — 27.(a)_____ (b)_____

28. Printing is always done from positions (a) through (b) of storage, therefore, data must be moved there before giving the instruction "(c)." — 28.(a)_____ (b)_____ (c)_____

UNIT II

Lesson 7

NOTES AND ANSWERS

ANSWERS	REMINDERS
25.(a) 001 (b) 080	A knowledge of these reserved areas of storage is absolutely essential. They should be thoroughly memorized if they are not already firmly in mind.
26. 100	
27.(a) 101 (b) 180	
28.(a) 201 (b) 300 (c) WRITE	

BASIC INSTRUCTIONS: Each instruction is recognized in the machine by its one-digit operation code. In this and the lessons that follow you will learn these codes and, with the table given in lesson 6, you will develop instruction words which will direct the computer's operation. The following are basic instructions:

Set Word Mark: This instruction may use either the A address, if one W/M is needed, or the A and B addresses if W/M's are needed in two places. The operation code is a comma (,). The A and B addresses should be the specific character positions at which word marks are needed. The d position in the instruction word is not used.

Op Code	A-Address	B-Address	Digit Mod.
,	XXX	XXX	(not used)

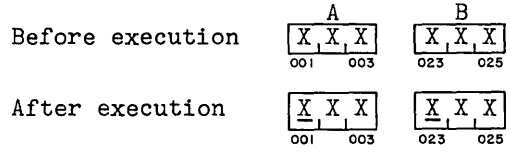
Move Characters: This instruction must use both the A and B fields but the digit modifier is not used. The data stored at the A address is moved to the B address without destroying the A field. The A address must be the address of the low-order digit in the field and the B address the low-order digit of the new field. Data is moved one digit at a time from right to left until a W/M is encountered in either field. W/M's are not moved or destroyed.

Op Code	A-Address	B-Address	Digit Mod.
M	XXX	XXX	(not used)

EXAMPLES:

1. Instruction - "Set word marks in positions 001 and 023"

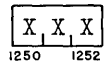
Op	A	B
,	0 0 1	0 2 3



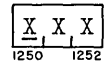
2. Instruction - "Set word mark in position 1250"

Op	A
,	S 5 0

Before execution



After execution

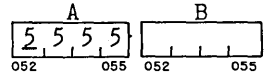


As shown in lesson 6, S is the code for 12nn.

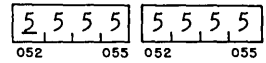
3. Instruction - "Move four characters from positions 052-055 to 252-255"

Op	A	B
M	0 5 5	2 5 5

Before execution



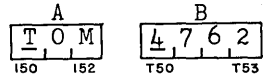
After execution



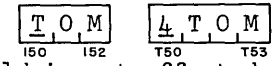
4. Instruction - "Move three characters from 150-152 to 1350-1353"

Op	A	B
M	1 5 2	T 5 3

Before execution



After execution

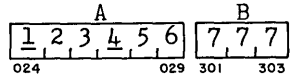


The high-order position of the B-field is not affected since the first W/M encountered stops the operation.

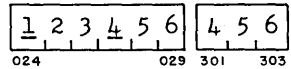
5. Instruction - "Move three characters from positions 027-029 to 301-303"

Op	A	B
M	0 2 9	3 0 3

Before execution



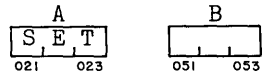
After execution



6. Instruction - "Set word marks in positions 021 and 051"

Op	A	B
,	0 2 1	0 5 1

Before execution



After execution

PROBLEMS: Write your answers to the following questions in the space provided, referring as necessary to the text. After you have written and checked your answers, refer to the correct solutions on the following page. Do not proceed to the next lesson until you fully understand the questions and their correct solutions.

Show how the memory positions would look after execution of each of the following instructions:

Instruction	Before Execution	After Execution
29. M 4 0 0 9 5 2	A: F I E L D (396–400); B: 1 2 3 4 5 6 (947–952)	29. B (947–952)
30. M 0 5 5 4 2 5	3 2 2 3 4 (051–055); 9 8 7 6 (422–425)	30. B (422–425)
31. M 0 0 5 0 2 5	A B C D (002–005); 1 2 3 4 (002–025)	31. B (022–025)
32. ' 0 2 5	X Y Z (025–027)	32. A (025–027)
33. M J 0 9 5 0 1	3 3 3 3 (J06–J09); 4 4 4 4 (498–501)	33. B (498–501)
34. M 0 0 7 0 2 7	P (007); J J J (025–027)	34. B (025–027)
35. M 0 2 2 0 5 1	6 6 (021–022); 8 8 8 8 (049–052)	35. B (049–052)
36. ' 0 0 1 0 2 3	S E T (001–003); H O L D (021–024)	36. A (001–003); B (021–024)

UNIT II

Lesson 8

NOTES AND ANSWERS

ANSWERS	REMINDERS
29. B: 1 F I E L D (947 … 952)	The high-order position in cell 947 is untouched.
30. B: 2 2 3 4 (422 … 425)	W/M in the B field stops the move.
31. B: A B C D (022 … 025)	A W/M in either the A or B field will stop character transmission.
32. A: X Y Z (025 027)	W/M set in position 025.
33. B: 4 4 3 3 (498 … 501)	First W/M stops the move.
34. B: J J P (025 027)	W/M's are not transmitted to the B field.
35. B: 8 6 6 8 (049 … 052)	Low-order position specified in B address was 051. Positions 049 and 052 unaffected.
36. A: S E T (001 003)	W/M's set in the locations specified by the instruction.
B: H O L D (021 … 024)	

INPUT-OUTPUT INSTRUCTIONS: Means for getting data into storage and out of storage requires a set of input-output instructions. The simplest means of getting data into memory is to read a card which has been punched with the coded information. Cards may be punched by the computer with data developed during processing. Printed information may be supplied by directing the printer to print information stored in memory. The following instructions direct these operations:

Read a Card: This instruction causes the card reader to read a card transmitting the 80 columns of data to memory positions 001 through 080. Word marks are not disturbed. A or B addresses or the digit modifier are not needed.

Op Code	A-Address	B-Address	Digit Mod.
1	(not used)	(not used)	(not used)

Punch a Card: This instruction causes the card punch to feed a blank card which is then punched with the 80 characters stored in memory positions 101 to 180. Word marks are not disturbed nor are they punched in the card.

Op Code	A-Address	B-Address	Digit Mod.
4	(not used)	(not used)	(not used)

Print (Write a Line): This instruction directs the printer to print the 100 characters stored in memory positions 201 to 300. Word marks are not disturbed nor are they printed. (132 characters in some models.)

Op Code	A-Address	B-Address	Digit Mod.
2	(not used)	(not used)	(not used)

NOTE: The Read a Card instruction will bring data into the reserved area without affecting word marks in the area. Therefore the programmer can set word marks in this area at the beginning of his program and not change them unless the size of fields in the cards change.

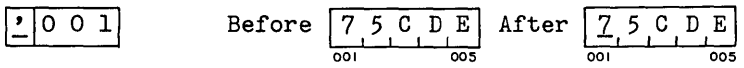
EXAMPLES:

1. A card is read into the read area. The information in Col. 1-5 are stored in positions 001-005. We want to punch a new card with the same information in Col. 11-15.

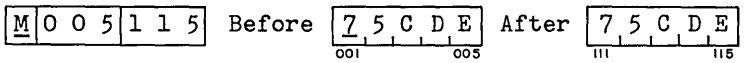
   Storage would look like this:

   To get the data from read area to punched card:

   (a) Set W/M at 001



   (b) Move data from read area to punch area

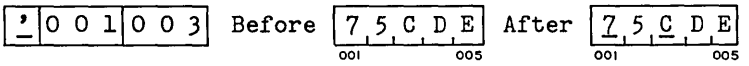


   (c) Punch a card

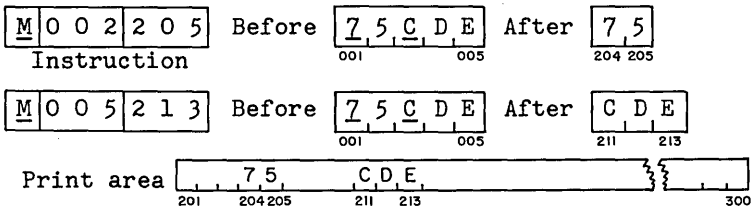
   All memory areas would be unchanged.

2. Let us assume that in the same example we also wish to print the five characters but not together. We want to print the first two digits in print positions 4-5 and the last three digits in print positions 11-13.

   (a) Set W/M's at 001 and at 003



   (b) Move data from read area to print area



(c) Print data

2	All memory areas would be unchanged.

NOTE: When a card is read to memory Col. 1 will be stored in memory position 001, Col. 2 in position 002, etc. To punch in Col. 1 place the data in memory position 101, Col. 2 in position 102, etc. Simply add 100 to the desired column to obtain the memory address. To print in the 1st position of the print line store data in memory position 201, 2nd print position in 202, etc. Simply add 200 to the desired print position to obtain the memory address.

PROBLEMS: Write your answers to the following questions in the space provided, referring as necessary to the text. After you have written and checked your answers, refer to the correct solutions on the following page. Do not proceed to the next lesson until you fully understand the questions and their correct solutions.

In each of the following problems, show the locations in the reserved areas where data must be moved to achieve the desired result.

37. Print in positions 85-90. 37. _____

38. Punch in Columns 5-14. 38. _____

39. Punch in Column 25. 39. _____

40. Print in positions 25-34. 40. _____

41. Punch an entire card exactly as it came in. 41. _____

42. In problem 41 above show the move instruction required to move the data from read area to print or punch area. 42. | | | |

43. In problem 40 show the "Set W/M" instructions required to control the movement of data. 43. | | |

UNIT II

Lesson 9

NOTES AND ANSWERS

ANSWERS

37. 290

38. 114

39. 125

40. 234

41. 180

42. | M | 0 8 0 | 1 8 0 |

43. | ' | 0 2 5 |

REMINDERS

Simply add 100 to any desired punch position 0-80 to get the punch-area address.

Add 200 to any desired print position to get the print-area address.

It is usually best to set word marks in the read-area when the program begins and leave them as long as the fields within the input card are in the same positions.

ERASING INFORMATION: The following instructions erase information from memory:

Clear Word Mark: This instruction, opposite to the set word mark instruction, will erase word marks from the memory positions designated by the A and B addresses. The B address need not be used.

Op Code	A-Address	B-Address	Dig. Mod.
¤	XXX	XXX	(not used)

Clear Storage: This instruction will erase up to 100 positions of memory. Clearing (erasing) starts at the position specified by the A-address and continues leftward down to the nearest 100's position. Word marks are erased when encountered.

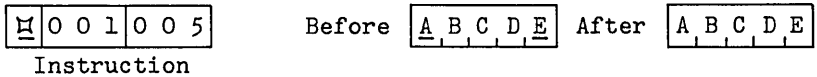
Op Code	A-Address	B-Address	Dig. Mod.
/	XXX	(not used)	(not used)

Clear Storage and Branch: This instruction has the same Op. Code and performs the same as the Clear Storage instruction except that it has a B-address. When the B-address is present erasing begins at the B-address leftward to the nearest 100's position. Upon reaching the nearest 100's position the next instruction is taken from the memory position designated in the A-address.

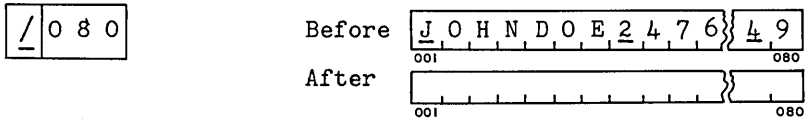
Op Code	A-Address	B-Address	Dig. Mod.
/	XXX	XXX	(not used)

EXAMPLES:

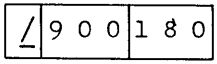
1. Clear word marks from positions 001 and 005



2. Clear read area



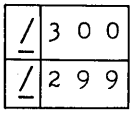
3. Clear punch area and take next instruction from 900



Punch area 101-180 will be erased. The next instruction will be taken from 900 rather than from the next word as would normally be true.

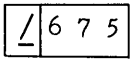
NOTE: The branch instruction always specifies the high-order or Op Code portion of the instruction which is to be next executed.

4. Clear print area



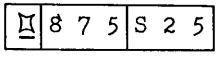
Print area 201-300 will be erased. The first Clear instruction will erase position 300 only and the second one will erase positions 299 through 200.

5. Clear a storage area



Storage area will be cleared from position 675 down through position 600.

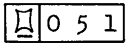
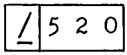
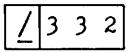
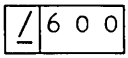
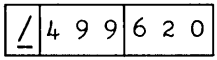
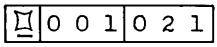
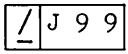
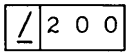
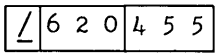
6. Clear word marks from positions 875 and 1225



Word marks will be erased from the two indicated storage positions.

PROBLEMS: Write your answers to the following questions in the space provided, referring as necessary to the text. After you have written and checked your answers, refer to the correct solutions on the following page. Do not proceed to the next lesson until you fully understand the questions and their correct solutions.

What would be the effect of the following instructions:

44.		Would clear what?	44. _____
45.		Would clear which storage areas?	45. _____
46.		Would clear which storage areas?	46. _____
47.		Would clear which storage areas?	47. _____
48.		(a) Would clear what storage?	48.(a) _____
		(b) Would branch to where?	(b) _____
49.		Would clear what?	49. _____
50.		Would clear which storage areas?	50. _____
51.		Would clear which storage areas?	51. _____
52.		(a) Would clear what storage areas?	52.(a) _____
		(b) Would branch to where?	(b) _____

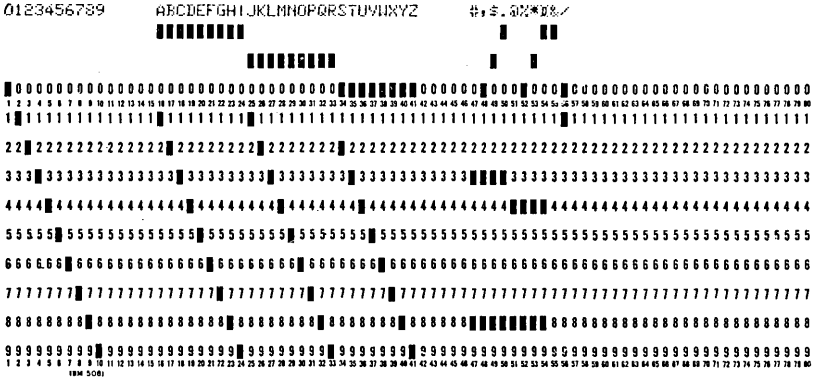
UNIT II

Lesson 10

NOTES AND ANSWERS

ANSWERS	REMINDERS
44. W/M at position 051	The W/M would be erased at 051 without affecting the data.
45. 500-520	Erasing begins at the address specified and continues leftward to the hundreds position which is also erased.
46. 300-332	
47. 600 only	
48.(a) 600-620	The Clear Storage and Branch has the "Clear" address in the B position and the "Branch" address in the A position.
(b) 499	
49. W/M's at positions 001 and 021	
50. 2100-2199	
51. 200 only	
52.(a) 400-455	
(b) 620	

PUNCHED CARDS: The two cards on this page show the holes punched to represent characters and the interpretation at the top of the cards. The first card shows the entire alphabet and the second card shows all numerics, the alphabet and special characters.



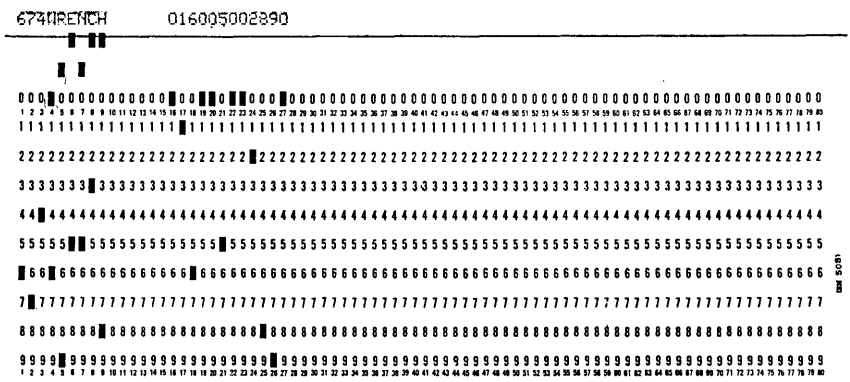
A SAMPLE PROGRAM: A small manufacturing concern makes many different items. Each day the foreman fills out a sheet with information about the day's production. Each line on the sheet represents a different item. The sheet looks like this:

PRODUCTION REPORT				
Part No.	Part Name	Number Produced	Man Hours	Labor Cost
674	WRENCH	16	5	28.90
684	FILE	20	4	24.00
692	HAMMER	41	11	65.60
698	SCREWDRIVER	20	7	41.70

This sheet is sent to a key-punch operator who punches the data into IBM cards; one card for each line of the report. She places the data in the following columns:

Columns	1- 3	Part Number
"	4-15	Part Name
"	16-18	Number Produced
"	19-21	Man Hours
"	22-27	Labor Cost
"	28-80	All left blank

Wherever a number has too few digits to fill the columns she punches zeros in the high-order positions. The card has data that looks like this--



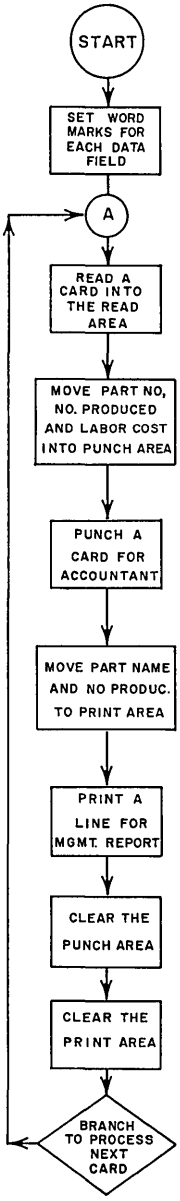
The accountant wants a card punched with the part number, the number produced and the labor cost in columns, as follows:

Columns 4- 6	Part Number
" 10-12	Number Produced
" 15-20	Labor Cost

The manager wants a printed list of all of the parts, by name, and the number produced, as follows:

Print positions 10-20	Part Name
" " 30-32	Number Produced
WRENCH	016
FILE	020
HAMMER	041
SCREWDRIVER	020

We will write a program which will read each card, punch the cards for the accountant and print a list of items produced for the manager. First we will prepare a flow-chart showing each step to be performed in the computer.



The word marks are placed at the beginning of the program and, since they are not destroyed, need not be placed again.

The card must be placed in memory at memory positions 001 to 080.

Each data field must be moved to its proper place in the punch area, memory positions 101 to 180.

When the data is in the punch area a card is punched.

Each data field must be moved to its proper place in the print area, memory positions 201-300.

When the data is in the print area a line of printing is printed.

Clear the digits already punched out of the punch area.

Clear the digits already printed out of the print area.

Branch back to the read instruction to read the next card.

PROGRAM

(The instruction address tells where the instruction word is located in memory)

	Op	A	B	d
5 0 0	,	0 0 1	0 0 4	
5 0 7	,	0 1 6	0 2 2	
5 1 4	1			
5 1 5	M	0 0 3	1 0 6	
5 2 2	M	0 1 8	1 1 2	
5 2 9	M	0 2 7	1 2 0	
5 3 6	4			
5 3 7	M	0 1 5	2 2 0	
5 4 4	M	0 1 8	2 3 2	
5 5 1	2			
5 5 2	/	1 8 0		
5 5 6	/	5 1 4	2 9 9	

Note: The last instruction clears the print area and branches back to the instruction located at 514 which causes a new card to be read and the process is repeated.

This process will continue until the last card is read. when there are no more cards the computer will stop.

PROBLEMS: Write your answers to the following questions in the space provided, referring as necessary to the text. After you have written and checked your answers, refer to the correct solutions on the following page. Do not proceed to the next lesson until you fully understand the questions and their correct solutions.

53. Input a card with the following data:

Card Col.	Description
5- 8	Insurance Deduction
9-12	FICA "
13-16	Withholding Tax
17-21	Total Deductions

INSTRUC. ADDRESS	Op	A	B
5 0 0			

Output a card with the following data:

Card Col.	Description
1- 4	Insurance Deduction
11-14	FICA "
21-24	Withholding Tax
76-80	Total Deductions

Write a program to read each card, move data to punch area, punch a card, clear the punch area and branch back to read again. Start your program at 500--and don't forget to set word marks in the read area before reading.

54. Input a card with the following data:

Card Col.	Description
1-19	Salesman's Name
20-22	Salesman's Number
25-39	City
40-44	Total Sales

INSTRUC. ADDRESS	Op	A	B
5 0 0			

Print a line with the information as follows:

Print Position	Description
20-39	Salesman's Name
50-54	Total Sales

Write a program to read each card, move data to print area, print a line and branch to repeat. Start at 500.

UNIT II

Lesson 11

NOTES AND ANSWERS

53.

INSTRUC. ADDRESS	Op	A	B	
5 0 0	,	0 0 5	0 0 9	Set W/M's
5 0 7	,	0 1 3	0 1 7	" "
5 1 4	1			Read a card
5 1 5	M	0 0 8	1 0 4	Move Ins. to Punch Area
5 2 2	M	0 1 2	1 1 4	Move FICA to Punch Area
5 2 9	M	0 1 6	1 2 4	Move Tax to Punch Area
5 3 6	M	0 2 1	1 8 0	Move Deductions to Punch Area
5 4 3	4			Punch a card
5 4 4	/	5 1 4	1 8 0	Clear Punch Area and Branch

54.

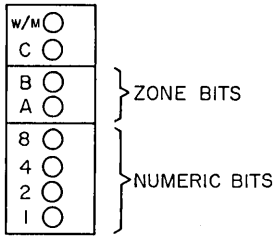
INSTRUC. ADDRESS	Op	A	B	
5 0 0	,	0 0 1	0 2 0	Set W/M's
5 0 7	,	0 2 5	0 4 0	" "
5 1 4	1			Read a card
5 1 5	M	0 1 9	2 3 9	Move Name to Print Area
5 2 2	M	0 4 0	2 5 4	Move Sales to Print Area
5 2 9	2			Print a line
5 3 0	/	5 1 4	2 9 9	*Clear Print Area and Branch to repeat

*Note that the "Clear" instruction will not clear position 300. If it is necessary to clear this position, a "Clear" instruction should precede the final instruction.

/	3 0 0

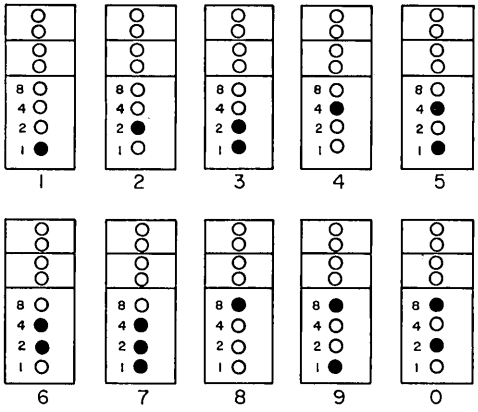
MAGNETIC CORE STORAGE: The IBM 1401 uses magnetic core storage for internally storing data and instruction. A magnetic core is a tiny donut-shaped bit of iron capable of being magnetized or de-magnetized by the direction of flow of an electrical current passing through it. When it is magnetized we may consider it to be "on" like a switch--when de-magnetized it is "off." Each storage position is composed of eight magnetic cores. A code made up of various combinations of these eight cores in "on" or "off" conditions tells the computer which character is stored at that position.

A storage position may be figuratively represented as follows:



The top core, or bit, is "on" if this character has a W/M associated with it. The C bit is used only by the computer as an internal accuracy check. The A and B bits are known as zone bits and the lower four bits are known as numeric bits. The numeric bits, singly or in combination, represent the digits from 0-9. The numeric bits, with one or the other or both zone bits added, make up the alphabet and special characters.

EXAMPLES: When a bit is shaded, it is considered to be "on."



Alphabetic Characters:

A through I - Both the A and B bits are "on," with numbers 1 through 9.

(A = 1, B = 2, C = 3, D = 4, E = 5, F = 6, G = 7, H = 8 and I = 9)

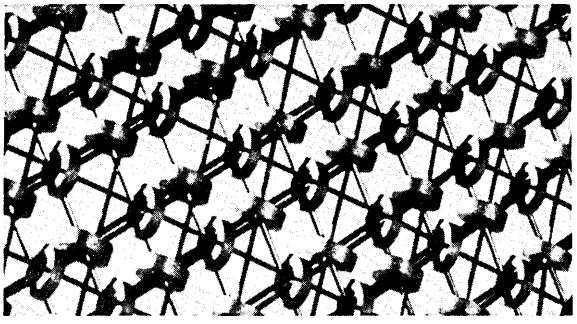
J through R - Just the B bit is "on" using numbers 1 through 9.

(J = 1, K = 2, L = 3, M = 4, N = 5, O = 6, P = 7, Q = 8 and R = 9)

S through Z - Just the A bit is "on" using numbers 2 through 9.

(S = 2, T = 3, U = 4, V = 5, W = 6, X = 7, Y = 8, and Z = 9)

Special characters such as the period (.), the comma (,), the slash (/), etc., are composed of various other combinations of the zone and numeric bits.



MAGNETIC CORE PLANE

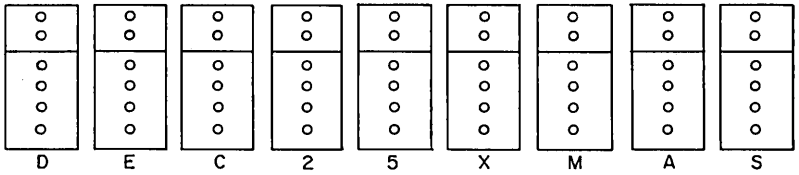
This close-up of a magnetic core plane shows some of the thousands of metallic, doughnut-shaped cores that make up the central memory units of most electronic computers. The cores can be magnetized individually by energizing the tiny electronic wires on which they are threaded. Data is represented in each core by its "on" or "off" (magnetized or unmagnetized" condition.

PROBLEMS: Write your answers to the following questions in the space provided, referring as necessary to the text. After you have written and checked your answers, refer to the correct solutions on the following page. Do not proceed to the next lesson until you fully understand the questions and their correct solutions.

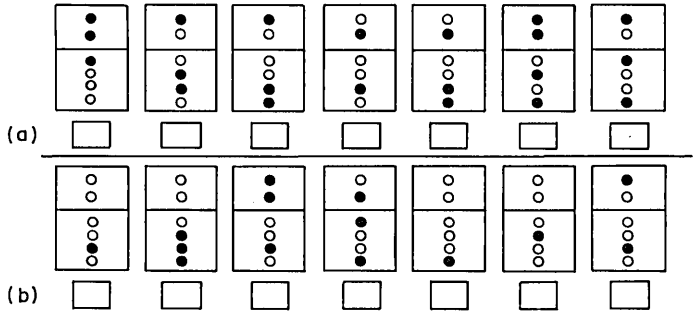
Since only six of the eight bits are used as coded versions of characters, the problems will only show six bits.

55. In the following symbolic representations of storage units, darken the bits to spell out

DEC25XMAS



56. Read the following storage units and place the letter or number each represents in the box below the figure.



57. If the zone bit was removed from the letter "P" the result would be 57._____

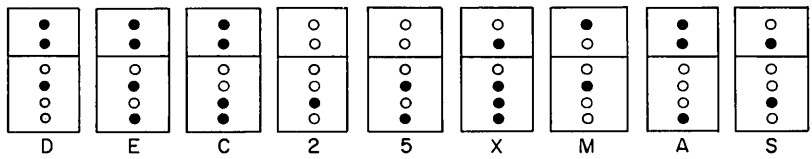
58. If the zone bits were removed from the letter "G" the results would be 58._____

59. If an A-bit was added to the numeric "5" the result would be 59._____

UNIT II

Lesson 12

NOTES AND ANSWERS

55. 

56.(a) H O L S T E R

(b) 2 7 B Z 1 4 K

57. 7

58. 7

59. V

Remember that zone bits are used to indicate addresses higher than 999. Therefore alphabetic and special characters may appear in instruction addresses as a means of designating such addresses. Refer to Lesson 6.

60. What storage positions are reserved for (a) Read, (b) Punch, (c) Print?

60.(a)____ ____

(b)____ ____

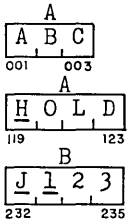
(c)____ ____

61. Show the contents of the indicated field after execution of the following instructions.

(a) | . | 0 0 1 |

(b) | M | 1 2 3 | 2 3 5 |

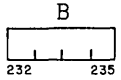
Before:



61. After:

(a)

(b)



62. Data is already in the Read area. Show the instructions that would be used to punch all of this data on a card.

62.

Op	A	B

63. Show the instructions that will cause the computer to execute the following:

(a) Clear storage area 650-600

(b) Clear word mark from position 023.

(c) Clear storage areas 429-400 and branch to location 620.

63.

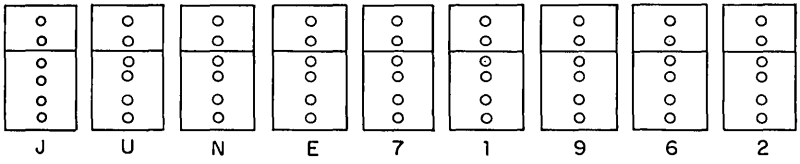
(a)

Op	A	B

(b)

(c)

64. In the following symbolic representations of storage units, darken the bits to spell out: JUNE71962



65. If both A and B zone bits were added to the 7, 1, 9, 6, 2 above, what letters of the alphabet would they represent?

7 = ☐ 1 = ☐ 9 = ☐ 6 = ☐ 2 = ☐

UNIT II QUIZ

ANSWERS

REMINDERS

60.(a) 001 - 080

(b) 101 - 180

(c) 201 - 300

Refer to page 15

61.(a) A: A B C

(b) B: J O L D

Refer to page 17

62. M | 0 8 0 | 1 8 0 | 4

Refer to page 21

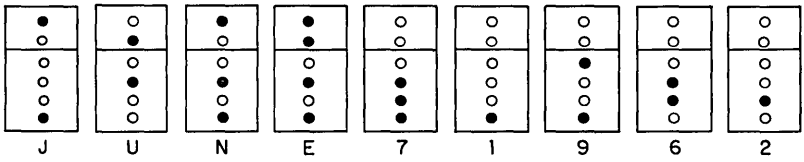
63.(a) / | 6 5 0

(b) ⌸ | 0 2 3

(c) / | 6 2 0 | 4 2 9

Refer to page 25

64.



65. 7 = G 1 = A 9 = I 6 = F 2 = B

Refer to page 35

DATA TRANSMISSION: In addition to the MOVE instruction (already discussed) there are several other methods of moving characters from one location in storage to another. They are:

MOVE & ZERO SUPPRESS:

Op	A	B
Z	X X X	X X X

This instruction operates in a similar fashion to MOVE except that any zeros in the high-order positions (leading zeros) will be replaced by blanks. The data located at the address shown in (A) will move to address (B) starting with the low-order position and stopping with the word mark in the high-order position of field (A).

MOVE DIGIT:

Op	A	B
D	X X X	X X X

This instruction will move a single digit from position (A) to position (B). Only the numeric bits are moved (i. e., bits 1, 2, 4, 8), so this instruction cannot be used for alphabetic or special characters. Since only one digit is moved, the word mark is not used.

MOVE ZONE:

Op	A	B
Y	X X X	X X X

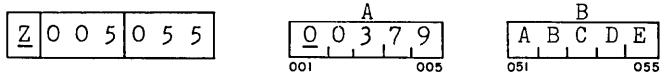
This instruction will move only the zone bits of the single character addressed by (A) to (B). It might be considered opposite to MOVE DIGIT in that it moves only the zone bits (i. e., bits A and B), therefore it will not transmit numeric information. The zone bits of (A) will replace any zoning at (B). The word mark is not used.

LOAD:

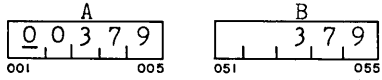
Op	A	B
L	X X X	X X X

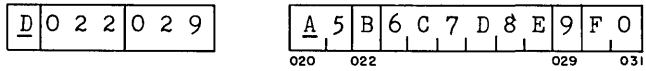
This instruction is like MOVE except that the W/M in the (A) field will be moved to the (B) field along with the data. Any W/M's already present in the (B) field will be erased.

EXAMPLES:

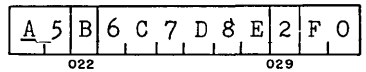
(1) 

Move the five digits at 001-005 to 051-055 and suppress insignificant zeros.

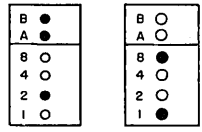
After 

(2) 

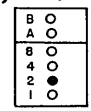
Move the numeric portion of the character at 022 to position 029.

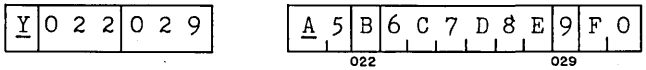
After 

Note that storage at 022 and 029 looks like this to start:

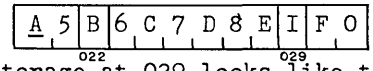


After the instruction is executed, 029 looks like this:

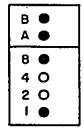


(3) 

Move the zone portion of the character at 022 to position 029.

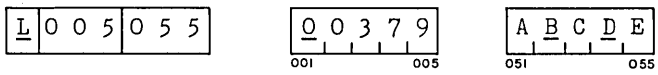
After 

After execution, storage at 029 looks like this:



A (9) with A and B zoning becomes the letter (I).

EXAMPLES cont'd:

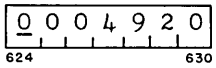
(4) 

Move the five digits and W/M at 001-005 to 051-055 and clear the word marks presently at 052 and 054.

After 

PROBLEMS:

66. After adding several different fields together we have the result:



We want to print the significant digits only (i. e., suppressing the insignificant zeros) in print positions 87-90. Show the MOVE & ZERO SUPPRESS instruction to accomplish this.

66.

Op	A	B

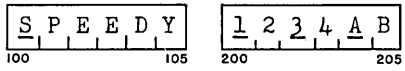
67. The letter "K" is located at position 400 and the letter "V" is located at position 500. Show a MOVE DIGIT instruction that will change the "V" at 500 to an "S".

67.

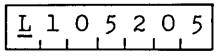
Op	A	B

68. Show a MOVE ZONE instruction that will change the "V" at 500 to an "N".

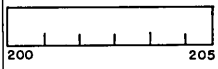
68.

69. 

Show how 200-205 would look after execution of the LOAD instruction:



69.



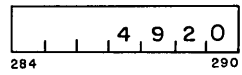
UNIT III

Lesson 13

NOTES AND ANSWERS

66.

Op	A	B
Z	630	290



Note that positions 284-286 will be filled with blanks regardless of what was previously stored there.

67.

D	400	500

Only the numeric portion of "V" is affected--changing it to a numeric 2 with the A-bit which equals "S".

68.

Y	400	500

The B-bit associated with "K" replaces the A-bit associated with "V". The numeric portion of "V" (i. e., 5) remains unchanged.

69.

S	P E E D Y

200 205

All the characters designated by the A-address move to the B-address, including the W/M. W/M's already in B are erased.

LOGICAL DECISIONS: Making logical decisions is an important computer capability. If you were searching for a certain folder in a file cabinet that contained a folder for each employee, by name, and in alphabetic sequence, you would follow a certain procedure. You would memorize the name, then search the proper section of the file name-by-name until you found a folder matching the name--or until you found a name whose alphabetic sequence was higher than the one you were seeking. Thus you would either locate the correct folder, or prove that the folder was not in the file. An IBM 1401 makes comparisons and decisions in a similar fashion. The following computer operations deal with comparisons, tests and decisions:

COMPARE:

Op	A			B		
C	X	X	X	X	X	X

This instruction causes the data in the field designated by the B-address to be compared (starting with the low-order characters), character by character, to the data in the A-field. A word mark in either field stops the comparison. If the B-field is longer than the A-field, an unequal comparison will result. If the A-field is longer than the B-field, only the number of characters in the B-field will be compared, ignoring the high-order digits of the A-field. Normally the programmer will only compare fields of equal length. The result of the comparison will be stored in "triggers" which may be tested by the TEST instruction. These "triggers" may show:

B does not equal A ($\neq$ unequal)

*B does equal A (= equal)

*B is less than A ($<$ less than)

*B is greater than A ($>$ greater than)

*NOTE: Not all IBM 1401 computers have the last three "triggers." These are optional features--not standard equipment. This text will assume that these triggers are available.

BRANCH:

Op	I
B	X X X

The letter I has been substituted for A to indicate that the memory address is the location of an instruction rather than the location of data.

This instruction causes the sequence of the program steps to change. The next instruction to be performed is designated by the I address.

TEST AND BRANCH:

Op	I	d
B	X X X	X

This instruction is like BRANCH except that the next instruction will come from I only if the test referred to by the d-character is satisfied. If the test is negative the next instruction following this one will be taken. Many tests are possible but only the four resulting from COMPARE will be discussed now. If a COMPARE instruction had been performed one or more "triggers" would be "turned-on." To test any of these four triggers you would place one of the following codes in the digit modifier:

d	Branch to I-address if:
/	B is not equal to A
S	B is equal to A
T	B is less than A
U	B is greater than A

Testing the "triggers" does not change them. They remain as they were after COMPARE until they are replaced by the results of a new COMPARE instruction. See other tests in IBM Reference D24-1401-0.

TEST CHARACTER AND BRANCH:

Op	I	B	d
B	X X X	X X X	X

This instruction causes the single character located at the B-address to be compared to the d-character. If they are the same the program branches to I. If the two characters are not alike, the program continues to the next instruction in sequence.

TEST FOR ZONE OR W/M AND BRANCH:

Op	I	B	d
V	X X X	X X X	X

This instruction tests the zone bits only of the character located at the B-address. The type of test to be performed is designated by the digit modifier.

d	Branch to I-address if:
1	W/M bit is present
2	No A or B bits are present
B	A and B bits are present
K	B bit is present - no A bit
S	A bit is present - no B bit

Other tests are possible but are so seldom used that they will not be covered by this text. See the IBM Reference Manual.

EXAMPLES:

(1) (a)

Op	A	B
C	0 0 5	0 5 5

A-Field: X Y Z (003 – 005)

B-Field: X Y Z (053 – 055)

Triggers: ○ ≠ ● = ○ < ○ >

B is equal to A

(b)

Op	A	B
C	0 1 0	5 1 0

A-Field: D O E (007 – 010)

B-Field: R O E (507 – 510)

Triggers: ● ≠ ○ = ○ < ● >

B is not equal to A

B is greater than A

(2)

INSTRUC. ADDRESS	Op	I
5 0 0	B	6 0 4

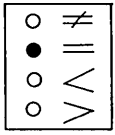
The next instruction to be performed will be the instruction located at 604 rather than the instruction at 504 which would normally follow.

EXAMPLES cont'd:

(3) (a)

INSTRUC. ADDRESS	Op	I	d
5 0 0	B	6 0 4	S

Triggers

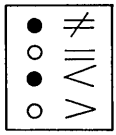


Since the trigger representing "equal to" is "turned-on," the next instruction would be taken from address 604.

(b)

INSTRUC. ADDRESS	Op	I	d
5 0 0	B	6 0 4	U

Triggers



Since the trigger representing "greater than" is off, the next instruction would be taken from the next instruction word in sequence which is 505.

(4)

INSTRUC. ADDRESS	Op	I	B	d
5 0 0	B	6 0 4	9 2 1	7

B
7

921

Since the character located at the memory position specified in B is exactly like the d-character (both are 7), the program will branch to the instruction at 604.

(5) (a)

INSTRUC. ADDRESS	Op	I	B	d
5 0 0	V	6 0 4	9 2 1	2

B
7

921

Since the character located at the memory position specified in B has no zone bits (it has only numeric bits), the program will branch to 604.

(b)

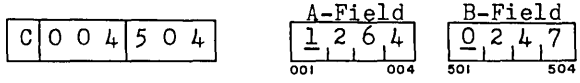
INSTRUC. ADDRESS	Op	I	B	d
5 0 0	V	6 0 4	9 2 1	1

B
7

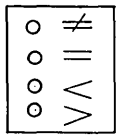
921

Since the character located at 921 does not have a W/M associated with it, the program will not branch, but will take its next instruction from memory position 508--the next word in sequence.

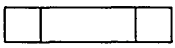
PROBLEMS:

70. 

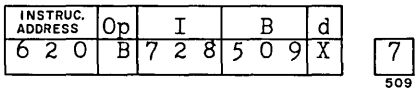
Black-out the circle(s) showing which trigger(s) would be "turned-on" after the above instruction.

70. 

71. Write an instruction which would cause the program to branch to 725 if the "equal to" trigger was on.

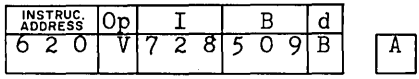
71. 

72. Where would the next instruction be taken from in the following case?



72. 

73. Where would the next instruction be taken from in the following case?



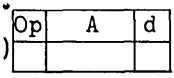
73. 

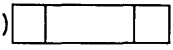
74. Write an instruction which would cause the program to branch to 455 under the following conditions:

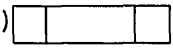
(a) B > A

(b) B ≠ A

(c) B < A

74. (a) 

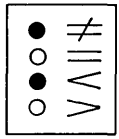
(b) 

(c) 

UNIT III

Lesson 14

NOTES AND ANSWERS

70. 

The data designated by the B-address is not equal to A and is less than A.

71.

Op	A	d
B	7 2 5	S

72. 6, 2, 8

Since the d-character is not 7 no branch will occur.

73. 7, 2, 8

Since the letter "A" located at 509 has both A and B zone bits the program would branch.

74. (a)

Op	A	d
B	4 5 5	U

(b)

B	4 5 5	/

(c)

B	4 5 5	T

PROGRAM EXAMPLE:

PROBLEM:

Write a partial program to read salary cards and punch FICA cards.

Salary Cards

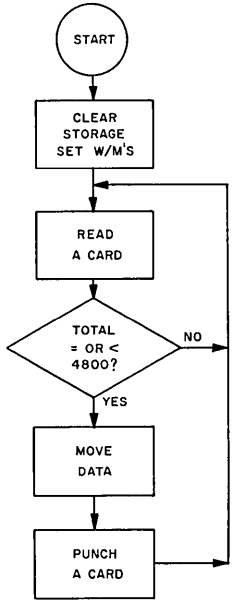
Employee Number	1- 3
Employee Name	4-25
FICA	26-30
Weekly Salary	31-36
Total Salary This Year	37-43

Punch a FICA card if the total salary this year is equal-to or less-than $4800.00 (0480000).

FICA Cards

Employee Number	1- 3
FICA	5- 9

Flow Chart:



CODING

	Instruc. Address	Op	A	B	d
	500	/	080		
	504	/	180		
	508	,	001	026	
	515	,	037		
	519	1			
	520	C	043	566	
	527	B	541		S
	532	B	541		U
	537	B	519		
	541	M	003	103	
	548	M	030	109	
	555	4			
	556	B	519		
*	560	0	480000		

* The data stored at 560-566 is not an instruction. It is a constant provided for the comparison performed at 520.

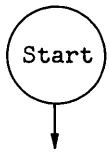
PROBLEMS:

75. The annual inventory is punched in cards like this:

Inventory Card:	Columns
Stock Number	1-10
Quantity Counted	11-15
Order Code	20

If the quantity counted = 0, or if the order code is "X," print the stock number in positions 11-20. Otherwise, branch to read the next card.

FLOW CHART



CODING

Instruc. Address	Op	A	B	d

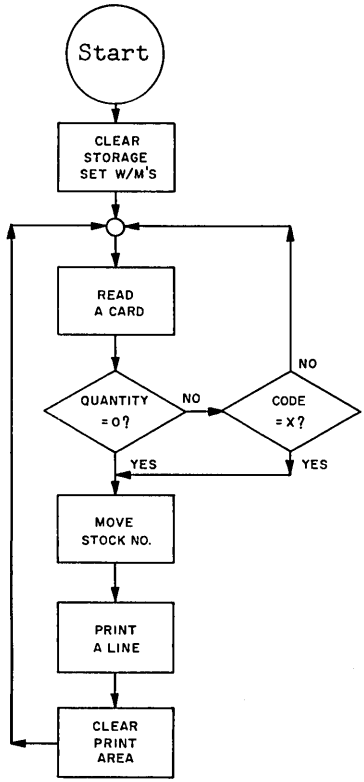
UNIT III

Lesson 14

NOTES AND ANSWERS

75.

FLOW CHART



CODING

Instruc. Address	Op	A	B	d
500	/	080		
504	/	299		
508	/	300		
512	,	001	011	
519	,	020		
523	1			
524	C	015	564	
531	B	548		S
536	B	548	020	X
544	B	523		
548	M	010	120	
555	2			
556	B	523		
560	0	00000		

76. Instruction:

Z	0 2 4	3 6 9

Before

A

3 3 3 3
021 024

B

6 6 6 6 6
365 369

76. After

B

365				369

77. Instruction:

D	4 6 1	4 6 4

Before

A

3 1 3
460 462

B

G E T
463 465

77. After

B

463		465

78. Instruction:

L	0 0 4	2 0 4

Before

A

H O L D
001 004

B

B E A T
201 204

78. After

B

201		204

79. Instruction:

C	0 2 5	1 0 5

Before

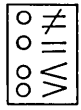
A

B R E A 2
021 025

B

A R E A 2
101 105

79. After
Which trigger or triggers will be "on"?



80. Instruction:

B	8 0 2	0 2 6	2

Before

B

2 3 4 5
025 028

80. After
Where will the program branch?

_____

81. Instruction:

V	6 0 0	1 2 3	K

Before

H I J K
121 124

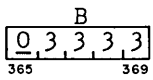
81. After
Where will the program branch?

_____

ANSWERS

REMINDERS

76. After

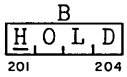


Refer to page 41

77. After

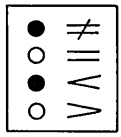
Refer to page 41

78. After



Refer to page 41

79. After



Refer to page 45

80. Next instruction in sequence.

Refer to page 46

81. 600

Refer to page 47

UNIT IV

Lesson 15

ADDITION: The IBM 1401 performs addition according to algebraic rules. According to these rules, every numeric field is considered to have a sign. The sign, associated with the low-order digit, may be plus (positive number) or minus (negative number).

Adding two plus fields results in a sum of the two fields with a sign of plus. Adding two minus fields results in a sum of the two fields with a sign of minus. When a plus field is added to a minus field the result is the difference between the two fields and carries the sign of the larger.

+2	-2	+2	+3	+2	+0
+2	-2	-2	-1	-6	-4
+4	-4	0	+2	-4	-4

Summing more than two fields is accomplished by successive addition of pairs:

$$(+4)+(-6)+(+7)+(-9)+(+7)+(-34)= \ ?$$

+4	-2	+5	-4	+3
-6	+7	-9	+7	-34
-2	+5	-4	+3	-31

Overflow is a special problem in computer arithmetic processes. Overflow occurs when the number of digits in the sum is greater than the number of digits in the original field.

+5	(1 digit)
+9	(1 digit)
+14	(2 digits)

We normally deal with overflow by simply adding digits to the left or high-order end of the field, but since memory positions are fully utilized for data such a step might result in destruction of an adjacent field. The IBM 1401 does not permit overflow to adjacent fields. When overflow occurs a special "trigger" is set that may be tested with a TEST AND BRANCH instruction:

Op	I			d
B	X	X	X	Z

If the overflow "trigger" is on, the program will branch to I.

Furthermore, the zone bits of the high-order digit of the result will be changed as follows:

1st	Overflow	"A" bit, no "B" bit
2nd	"	"B" bit, no "A" bit
3rd	"	Both "A" and "B" bits
4th	"	No "A" bit and no "B" bit
5th	"	Same as 1st overflow, etc.

Overflow can be easily avoided by making the B-field large enough to contain all possible digits formed in the sum.

ADD:

Op	A			B		
A	X	X	X	X	X	X

The data stored at the A-address is added to the data stored at the B-address. The result is stored at the B-address. The data stored at A is unaffected. Negative numbers are denoted by the presence of a "B" bit without an "A" bit over the low-order digit. Any other zoning or no-zoning denotes a positive sign. (Remember that numeric characters with zone bits appear in memory as alphabetic or special characters.) If the sum of A + B results in more digits than the B-field can contain, overflow will occur.

ADD:

Op	A		
A	X	X	X

This instruction causes the data stored at the A-address to be added to itself. The result is stored in the A-field. Overflow will occur if the number of digits in the result is greater than the number of digits in the A-field.

ZERO AND ADD:

Op	A			B		
?	X	X	X	X	X	X

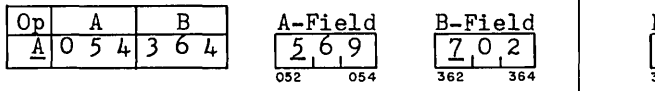
This instruction operates like the ADD instruction except that the B-field is set to zero (filled with zeros) before the A-field is added to it. If the A-field contains zone bits, they will be stripped-off of all but the low-order, or "sign", position.

EXAMPLES:

(1)

Since no B-bit is present over either low-order digit, both fields are plus and the result is plus.

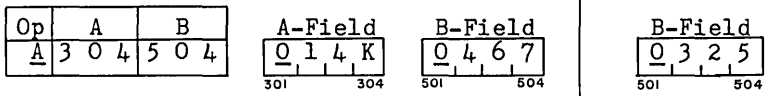
(2)



Since the result (1271) contains more digits than the B-field, overflow occurs; an "A"-bit is stored over the high-order position and the overflow indicator is set. Setting the B-field to four or more positions would have avoided overflow.

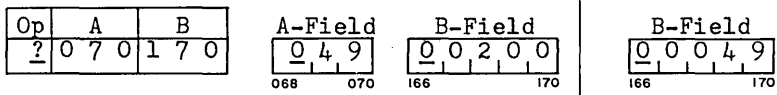
Overflow trigger is on

(3)



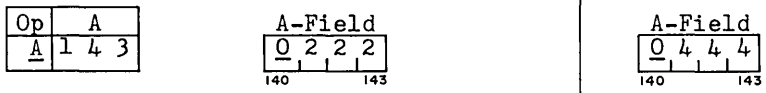
The numeric 2 with a "B"-bit in the zone bits appears in memory as a K. Adding the -142 (denoted as minus by the "B"-bit) to +467 results in a +325.

(4)



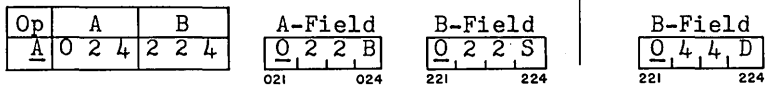
The B-field is filled with zeros before adding A.

(5)



The A-field is added to itself.

(6)



The alphabetic B is a 2 with both A and B bits, therefore +. The S is a 2 with just the A bit, therefore also +. When the two figures are added together, the result in the B-field will contain both A and B bits in the low-order position, therefore D.

PROGRAM EXAMPLE:

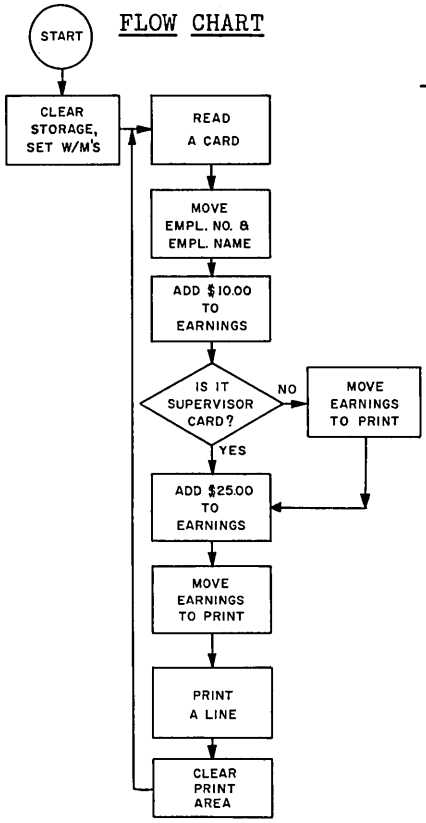
PROBLEM: A group of cards are to be processed. Add a $10.00 bonus to each employee's earnings and an additional $25.00 if the employee is a supervisor. Print out a report to management.

Input Cards	
Employee No.	1- 5
Employee Name	6-24
Net Earnings	71-77

Print Positions	
Employee No.	11-15
Employee Name	41-59
Total Earnings:	
Supervisors	75-81
Non-Supervisors	90-96

NOTE: Sup'v. card has an 11 punch in column 80 (11 punch on a card is equivalent to a B-bit in storage)

FLOW CHART



CODING

Instruc. Address	Op	A	B	d
500	/	080		
504	/	299		
508	/	300		
512	,	001	006	
519	,	071	080	
526	1			
527	M	005	215	
534	M	024	259	
541	A	593	077	
548	V	567	080	K
556	M	077	296	
563	B	581		
567	A	597	077	
574	M	077	281	
581	2			
582	/	526	299	
590	1	000		
594	2	500		

*Data stored in 590-593 and 594-596 are constants used as $10.00 and $25.00 additives.

PROBLEMS:

82. Indicate the result of each of the following additions:

	A-Field		B-Field
(a)	0 1 2	+	1 6 4
(b)	0 1 M	+	7 2 4
(c)	2 4 6	+	6 8 R
(d)	4 7 9	+	8 9 2

82.

	B-Field
(a)	
(b)	
(c)	
(d)	

83. Show the contents of the B-field after execution of the following instructions:

	Op	A	B
(a)	A	0 7 4	1 8 4
(b)	?	1 2 2	4 4 4
(c)	A	1 5 2	

A-Field	B-Field
1 2 2 (072–074)	0 9 8 9 (181–184)
0 1 6 (120–122)	0 7 9 4 (441–444)
1 4 8 (150–152)	

83.

(a) B-Field (181–184)

(b) B-Field (441–444)

(c) A-Field (150–152)

84. Program the following problem. Punch one card for one input card. Add salary, commission and overtime. Start coding at 500.

Input Card	
Name	1-19
Salary	20-24
Commis.	25-29
O'time	30-34

Output Card	
Name	1-19
Tot. Pay	45-49

FLOW CHART

CODING

Instruc. Address	Op	A	B	d
5 0 0				

UNIT IV

Lesson 15

NOTES AND ANSWERS

ANSWERS

82.(a) 176 — This is a "true add"

(b) 710 — This is a "complement add"--the result is the difference between the two numbers and takes the sign of the larger.

(c) 44L

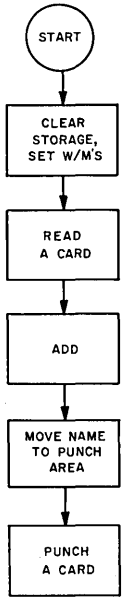
(d) T71 — This is an overflow causing an A-bit to be placed in the zone bits of the high-order digit.

83.(a) 1111 (181–184)

(b) 0016 (441–444) — The zero and add instruction can be used to move a number to a larger field and insure that the high-order digits are set to zero.

(c) 296 (150–152)

84. FLOW CHART



CODING

	Instruc. Address	Op	A	B	d
	500	/	080		
	504	/	180		
	508	,	001	020	
	515	,	025	030	
	522	,	145		
	525	1			
*	526	?	024	149	
	533	A	029	149	
	540	A	034	149	
	547	M	019	119	
	554	4			

*The zero and add (?) is used to set the output area to zero and to start the addition.

SUBTRACTION: As in addition the IBM 1401 follows algebraic rules for subtraction. In fact, the process is identical except that the sign of the subtrahend is reversed and then added to the minuend.

Minuend	+2	=	+2	+2	=	+2	-4	=	-4	+7	=	+7
Subtrahend	-(+2)	=	-2	-(-2)	=	+2	-(+6)	=	-6	-(+2)	=	-2
Difference			0			+4			-10			+5

Overflow may occur as in addition. Subtracting more than two fields is accomplished by successive subtraction of pairs.

SUBTRACT:

Op	A	B
S	X X X	X X X

The data stored in the A-field is subtracted from the data stored in the B-field. The result is stored in the B-field. Effectively, the data stored in the A-field reverses its sign and is added to the B-field algebraically. If the result contains more digits than the B-field, overflow will occur.

SUBTRACT:

Op	A
S	X X X

The data stored in the A-field is subtracted from itself. This instruction invariably results in zeroing the A-field.

ZERO AND SUBTRACT:

Op	A	B
!	X X X	X X X

This instruction operates like the subtract instruction except that the B-field is set to zero before subtraction occurs. The result is that data is moved from the A-field to the B-field with a reversal of the sign.

EXAMPLES:

Before Execution | After Execution

(1)

Op	A	B
S	054	364

A-Field: 569 (052–054)

B-Field: 702 (362–364)

B-Field: 133 (362–364)

The result is the same as if the A-field had the sign of minus and addition occurred.

(2)

Op	A	B
S	304	504

A-Field: 014K (301–304)

B-Field: 0467 (501–504)

B-Field: 0609 (501–504)

Reversing the sign of the A-field (-142) (-142 becomes +142) and adding, results in the sum of the two fields.

(3)

Op	A
S	140

A-Field: 0220 (137–140)

A-Field: 0000 (137–140)

When only an A-address is given the A-field is subtracted from itself.

(4)

Op	A	B
!	070	170

A-Field: 049 (068–070)

B-Field: 00200 (166–170)

B-Field: 0004R (166–170)

The B-field is set to zero and the data in the A-field is moved to B, with the sign reversed (the minus sign is indicated by a "B"-bit over the nine--or the letter R).

(5)

Op	A
!	070

A-Field: 049 (068–070)

A-Field: 04R (068–070)

If only an A-field is given the A-field reverses its sign.

PROBLEMS:

85. Indicate the result of each of the following subtractions:

	A-Field		B-Field
(a)	0 1 7	from	4 5 9
(b)	4 2 K	from	2 6 2
(c)	1 1 L	from	0 9 K
(d)	0 1 9	from	0 3 Q

85.

	B-Field
(a)	
(b)	
(c)	
(d)	

86. Show the contents of the B-field after execution of the following instructions:

	Op	A	B
(a)	S	1 6 5	8 1 7
(b)	!	0 2 1	3 5 7
(c)	S	1 5 2	
(d)	!	1 5 2	

(a) 0 0 2 2 (162–165) 0 8 6 Q (814–817)

(b) 1 M (020–021) 9 6 2 4 (354–357)

(c) 1 4 8 (150–152)

(d) 1 4 8 (150–152)

86.

	B-Field
(a)	(814–817)
(b)	(354–357)
(c)	A-Field (150–152)
(d)	A-Field (150–152)

87. Read a card containing three numbers (A, B and C). Subtract A-B-C and store the result in storage locations 1261-1265. Start coding at location 500.

Input Card	
A	21-25
B	41-45
C	61-65

FLOW CHART

CODING

Instruc. Address	Op	A	B	d
5 0 0				

UNIT IV

Lesson 16

NOTES AND ANSWERS

85.(a) 4 4 2 — The usual subtraction.

(b) 6 8 4 — The minus field is changed to plus and added in both (b) and (c).

(c) 0 2 1

(d) 0 5 P — The letter P is the numeric 7 with B-bit zoning.

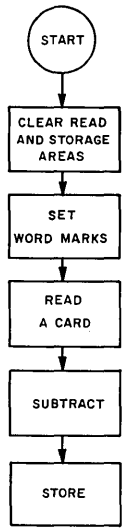
86.(a) 0 8 9 $\bar{0}$ — A minus zero is shown as a zero (0) with a minus sign drawn over it ($\bar{0}$).

(b) 0 0 1 4 — The A-field sign is reversed and is moved to the B-field after the B-field is set to zero.

(c) 0 0 0

(d) 1 4 Q — The sign of the A-field is reversed.

87. FLOW CHART



CODING

Instruc. Address	Op	A	B	d
5 0 0	/	0 8 0		
5 0 4	/	S 6 5		
5 0 8	,	0 2 1	0 4 1	
5 1 5	,	0 6 1		
5 1 9	1			
5 2 0	S	0 4 5	0 2 5	
5 2 7	S	0 6 5	0 2 5	
5 3 4	L	0 2 5	S 6 5	

MISCELLANEOUS INSTRUCTIONS:

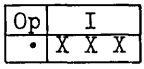
NO OPERATION:

This instruction requires no A, B or d part since it performs no operation. If other parts are present they have no effect. The primary purpose of this instruction code is to substitute for other operation codes when the programmer wishes to make them ineffective.

STOP:

This instruction causes the program to stop and the stop-key light on the console to turn on. When the start-key is depressed, the program resumes with the next instruction.

STOP AND BRANCH:

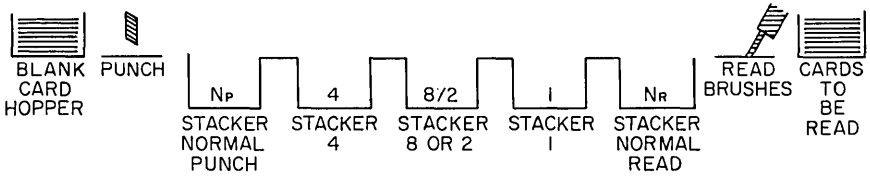


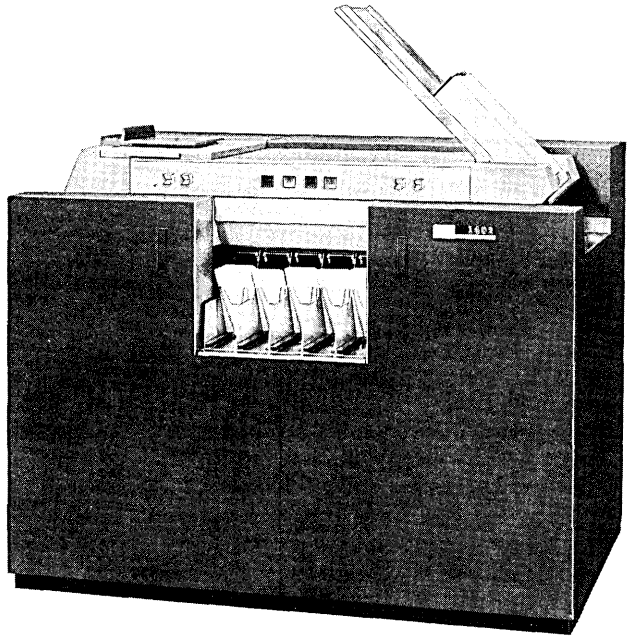
This instruction acts just like the stop instruction, except that when the start-key is depressed the program branches to the address in I.

STACKER SELECT:

As cards are read or punched they fall into a pocket (stacker) of the read punch unit. The IBM 1402 Card Read-Punch Unit has five stackers and permits the programmer some freedom in the selection of the stacker into which cards will fall.

Stackers are numbered as follows:





IBM 1402 Card Read Punch

All card reading and punching occurs in this unit. The reading occurs in the right-hand side at a maximum of 800 cards per minute. Punching occurs in the left-hand side at a maximum of 250 cards per minute.

Cards to be read pass two sets of reading brushes. The first set of reading brushes is used for a checking function; the second set completes the check and allows the data to enter storage.

On the punch side, cards first pass a set of punches where punching occurs and then a checking station where the punching is checked.

Both feeds have a "pretest" station for sensing misfeeds. All input characters are checked for validity. Invalid characters cause the machine to stop.

If no special selection is made the cards go into either the Normal Punch stacker or the Normal Read stacker. Using the STACKER SELECT instruction the programmer may designate the desired stacker by inserting a d-modifier according to the following rules:

d-Modifier	Feed	Stacker Pocket
1	Read	1
2	Read	8/2
4	Punch	4
8	Punch	8/2

The instruction must be given within 40 instructions after the READ or PUNCH instruction, or the card will have already passed the point where it can be directed.

EXAMPLES:

(1)

500	N	540	
504	M	536	500
511			
536	B	241	
540	.		

In this example, the first instruction is a NO OPERATION with an I-field. The first time through, the program would step to 504 which is a MOVE instruction causing the N to be replaced by a B operation code. The next group of instructions would be performed and the program would branch back to instructions located at 241. If the program should again come to 500 there would be a branch to 540, which is a STOP. Thus the NO-OP instruction acts as a "gate" which is open the first time through but is closed against a second entry.

(2)

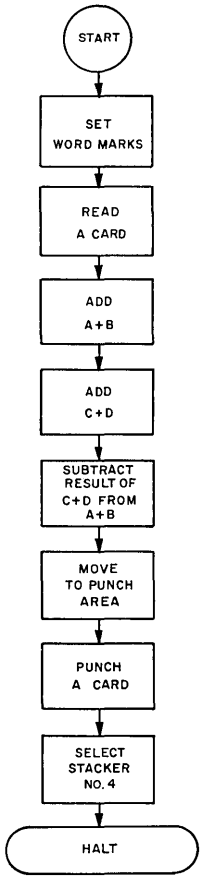
1		Read a card to 1-80
K	1	Stack card in stacker #1
4		Punch a card from 101-180
K	4	Stack card in stacker #4

PROGRAM EXAMPLE:

PROBLEM: Read a card into storage. Solve the following equation and punch a card with the result, placing it into stacker #4. Start the program at location 500.

Equation	Input Card		Output Card	
$(A+B)-(C+D) = T$	A	1- 5	T	51-57
	B	6-10		
	C	11-15		
	D	16-20		

FLOW CHART



CODING

Instruc. Address	Op	A	B	d
500	,	001	006	
507	,	011	016	
514	,	151		
518	1			
519	A	005	010	
526	A	015	020	
533	S	020	010	
540	M	010	157	
547	4			
548	K			4
550	.			

PROBLEMS:

Write instructions to perform each of the following:

88. Cause the program to stop but provide for branching to 644 when the start button is pressed.

88.

Op	A	B	d

89. Provide an instruction that will not operate but can be changed easily to move data at 200 to 280.

89.

90. Read a card and place it in stacker #8/2.

90.

91. Punch a card and place it in stacker Np.

91.

92. Read a card and place it in stacker Nr.

92.

93. Read a card and place it in stacker #1. Add two numbers (stored in loc. 005 and 010). Move to punch area and punch a card, placing it in stacker #4.

93.

UNIT IV

Lesson 17

NOTES AND ANSWERS

88.

Op	A	B	d
.	6 4 4		

89.

N	2 0 0	2 8 0	

Simply by moving an M to the operation code this instruction will change from NO OPERATION to MOVE.

90.

1			
K			2

By following the read instruction with a Stacker Select coded 2--the card will fall in stacker #8/2.

91.

4			

No Stacker Select is needed here since cards normally go to Np after punching.

92.

1			

No Stacker Select needed (as in Punch above).

93.

1			
K			1
A	0 0 5	0 1 0	
M	0 1 0	1 1 0	
4			
K			4

SAMPLE PROGRAM: The following program will illustrate the practical use of many of the instructions we have discussed:

A small insurance company has several field offices, each of which has a group of salesmen. One of the salesmen in each office is also the manager and receives extra pay for this duty.

Each week the offices send a report, in punched card form, to the home office in the following format:

Type of Information	Input Cd. Col.
Office Number	1- 3
* Salesman Number	4- 5
Salesman Name	7-30
Gross Sales ($)	41-46
Commission ($)	51-56

* If the salesman is a manager, his card has an 11-zone punch over the 1st digit of his number (Col. 4).

Each office places the manager's card last in the deck of cards sent in.

The home office puts all the decks together and places a special card at the back which is all blank except for the 3 letters END punched in Columns 1-3.

The big deck is loaded in the card read-punch machine and is processed with a program that does the following:

(1) When a manager's card is read it is placed in stacker #1--others go in stacker Nr.

(2) Managers get $100 added to their commission.

(3) A card is punched for each salesman as follows:

Type of Information	Output Cd. Col.
* Salesman's Number	1- 2
Salesman's Name	5-28
Commission	31-36

* The manager's number does not have an 11-zone punch in the output card.

(4) When a manager's card is punched it goes in stacker #4--others go in stacker Np.

(5) A line is composed and printed whenever a manager's card signals the end of a particular office report. The print line is composed as follows:

Type of Information	Print Line Position
Office Number	10-12
Total Sales	20-25
* Net Sales	30-35

* Net Sales = (Total Sales)-(Total Commissions)

(6) In the output cards and printed line, the insignificant or high-order zeros should not appear.

(7) Stop the program when the END card is encountered.

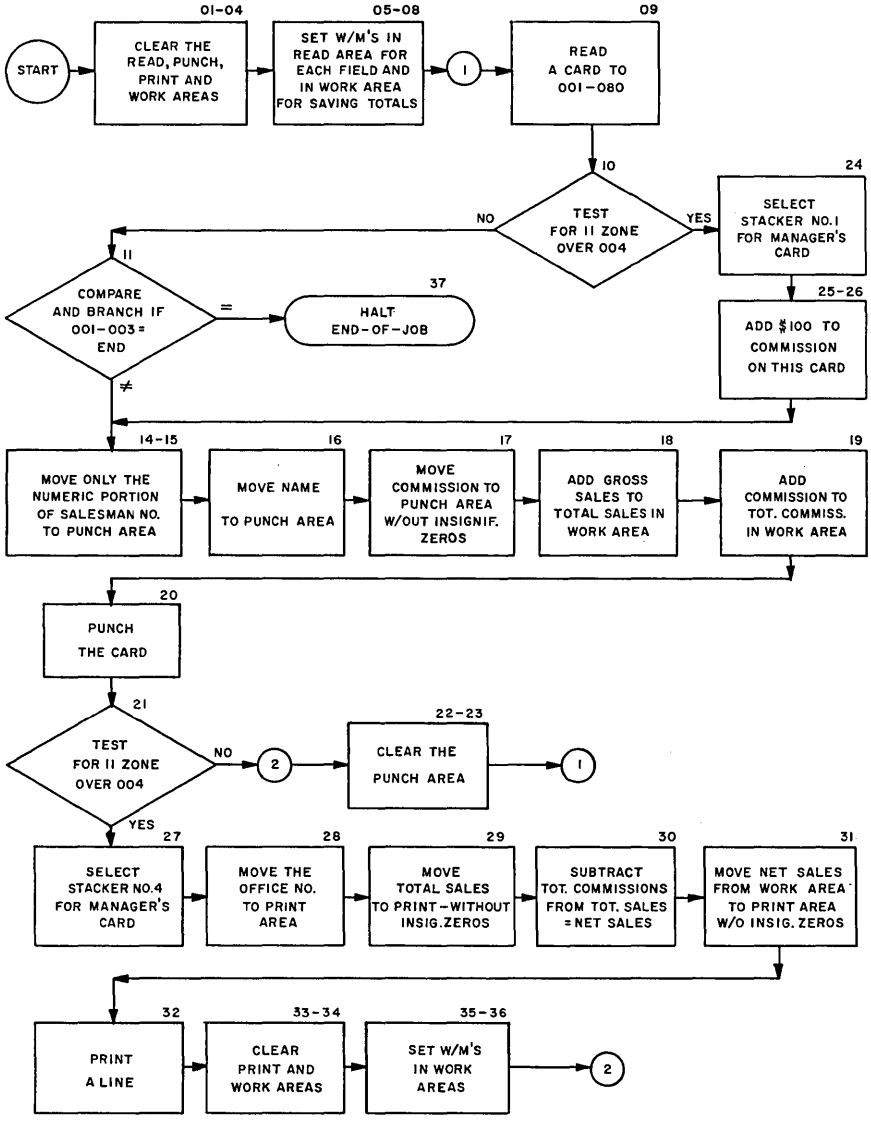
The program is shown as a flow chart and then is coded, on the following pages. The student should thoroughly understand what the program is to do and how it does it. Do not proceed further in the text without this understanding as subsequent lessons refer to this example.

QUICK REFERENCE - OPERATION CODES:

All operation codes covered so far may be found on this page along with the title of the code and the page where it was explained.

	OP	TITLE	REFER TO PAGE
W/M, MOVE AND LOAD CODES:	,	SET WORD MARK	17
	¤	CLEAR WORD MARK	25
	M	MOVE	17
	L	LOAD	41
	Z	MOVE AND ZERO SUPPRESS	41
	D	MOVE DIGIT	41
	Y	MOVE ZONE	41
INPUT/OUTPUT CODES:	1	READ	21
	4	PUNCH	21
	2	PRINT (WRITE)	21
ARITHMETIC CODES:	A	ADD	58
	S	SUBTRACT	63
	?	ZERO AND ADD	58
	!	ZERO AND SUBTRACT	63
LOGICAL CODES:	C	COMPARE	45
	B	BRANCH	46
	B	TEST AND BRANCH	46
	B	TEST CHARACTER AND BRANCH	46
	V	TEST FOR ZONE OR W/M	47
MISCELLANEOUS CODES:	/	CLEAR STORAGE	25
	N	NO OPERATION	67
	.	STOP (HALT)	67
	.	STOP AND BRANCH	67
	K	STACKER SELECT	67

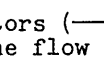
The small numbers above each box show the line number(s) in the coded program which contain the instructions to accomplish each step.



CODED PROGRAM

Line No.	Instr. Add.	Op	A	B	d	Remarks
01	500	≤	080			Clear card read area
02	504	≤	180			Clear card punch area
03	508	≤	299			Clear 99 positions of print area
04	512	≤	349			Clear "work area" & position 300 of print area
05	516	,	001	004		Set W/Ms f/office no. & SM's no.
06	523	,	007	041		Set W/Ms f/SM's name & gross sales
07	530	,	051			Set W/Ms for commission
08	534	,	325	310		Set W/Ms for special work areas for saving totals
09	541	1				Read a card into 001-080
10	542	V	625	004	K	Test for an 11-zone bit over Col.4 --manager's card
11	550	C	560	003		Compare 1st 3 dig. to letters END
12	557	N	END			No-Op. is used to store letters END for compare
13	561	B	688		S	Branch if 1st 3 characters = END
14	566	D	004	101		Move only numeric pt. of SM's no.
15	573	D	005	102		If mgr's card, 11-zone is removed
16	580	M	030	128		Move salesman's name to punch area
17	587	Z	056	136		Move commission to punch area-- suppress insignificant zeros
18	594	A	046	349		Add gross sales to work area to save total sales
19	601	A	056	324		Add commission to 2nd work area to save total commission
20	608	4				Punch the card
21	609	V	638	004	K	Test f/mgr's card--11-zone still to read area
22	617	≤	180			Clear the punch area
23	621	B	541			Branch back to read another card
24	625	K			1	Select stacker #1 for mgr's card
25	627	A	691	046		Add $100 to manager's commission
26	634	B	566			Branch back to punch a card
27	638	K			4	Select stacker #4 for mgr's card
28	640	M	003	212		Move office number to print area
29	647	Z	349	225		Move total sales from work area to print area
30	654	S	324	349		Subtract total commissions from total sales = net sales
31	661	Z	349	235		Move net sales to print area
32	668	2				Print a line
33	669	≤	299			Clear print area
34	673	≤	349			Clear work area
35	677	,	325	310		Reset W/Ms for work area fields
36	684	B	617			Branch back to clear punch area & read a card
37	688	•				Stop--this is the end of program
38	689	1	00			This is not an instruction--it is a "constant" of 100 to be used for adding to manager's commission.

One of the first things accomplished in a program is the clearing of the Read, Punch and Print areas. This is done because there is no way of knowing what data may be in storage at those locations before the program starts. If the program does not use the Print or Punch areas, then they need not be cleared.

Notice the use of connectors () in the flow chart on page 76. Also that the flow of information is from top to bottom and from left to right. Connectors are handy because they eliminate the need for crossing over lines and they tie flow charts together if they cover more than one page. They also offer a simple way of returning from one page of a flow chart to another. There is no hard rule that says flow charting from bottom to top or from right to left may not be done. It is just more usual and accepted to do it the other way.

In a very simple program, all of the processing may be accomplished in the reserved areas, but in actual practice this rarely happens. Usually, work areas must be assigned and used by the program to accomplish the necessary processing. Note the work areas in the sample program on page 77.

The instruction on line 23 takes the program back to read another card. This continues, bringing in and processing one card after another until all cards are processed. The process of going around and around, through a program (or a portion of a program), is called a LOOP.

CHAINING: The IBM 1401 has a unique method of reducing the number of characters required for instructions. This is accomplished by use of "counters" contained in the circuitry of the computer. The A-register keeps track of the next data character to be operated-on by an A-address. The B-register keeps track of the next data character to be operated-on by the B-address. Whenever a word mark stops operation on a data field addressed by A, the A-register contains the address of the character to the left of the W/M. Thus if the next instruction normally required an A-address, but didn't have it, the computer would start processing the field to the left of the one just completed. The same is true of the B-register. This process is called "chaining" instructions.

It must be remembered that the fields to be chained must be in sequence and the chaining operation always moves from the highest storage position to the lowest.

If the A-fields are not in sequence, but the B-fields are in sequence, partial chaining may be accomplished by eliminating all but the first B-field address.

One of the major advantages of chaining is that storage space is saved by this technique. For example:

MOVE		Instructions
From	To	
001-005	101-105	M 005 105
006-010	106-110	M 010 110
011-015	111-115	M 015 115

These three instructions take up 21 characters of storage space. With chaining, the same moves may be accomplished as follows:

M 015 115
M
M

This is a total of 9 characters, or a saving of 12 characters of storage.

EXAMPLES: In the sample program in Lesson 18, it was necessary to clear the Read, Punch, Print and work areas. Effectively we cleared from 000-349, using four CLEAR instructions which occupied 16 characters. We could have cleared 000-349 using four CLEAR instructions which occupied 7 characters, as follows:

(1)

Line Number	Instruct. Address	Op	A	B	d
01	500	/	349		
02	504	/			
03	505	/			
04	506	/			

The first instruction would clear leftward 349-300, the second 299-200, then 199-100 and 099-000.

(2) MOVE:

From	To	Chained Instructions
208-229	320-341	M 240 352
230-232	342-344	M
233-240	345-352	M

Both the A and B-registers are stepped leftward and as each W/M is encountered they contain the A and B address for the next MOVE. The first MOVE would place 240-233 into 352-345. The second MOVE would place 232-230 into 344-342, and the third MOVE would place 229-208 into 341-320.

(3) MOVE:

From	To	Chained Instructions
421-425	601-605	M 495 615
460-464	606-610	M 464
491-495	611-615	M 425

Since the B-fields are in sequence, they may be chained. Each of the A-field locations must be specified as these addresses are not in sequence.

PROBLEMS:

94. Show the instructions required to clear storage:

100-523

94.

Op	A	B	d
–			
–			
–			
–			
–			

95. Show the instructions required to move several data fields:

From	To
010-020	210-220
021-024	221-224
025-030	225-230

95.

Op	A	B	d
–			
–			
–			

96. Show the instructions required to ADD the same number to the following data fields:

Add	To
010	251-255
	246-250
	241-245

96.

Op	A	B	d
–			
–			
–			

97. Show the instructions required to move the following data fields:

From	To
020-029	201-210
040-049	211-220
050-059	221-230

97.

Op	A	B	d
–			
–			
–			

UNIT IV

Lesson 19

NOTES AND ANSWERS

94.

Op	A	B	d	
/	5 2 3			523-500
/				499-400
/				399-300
/				299-200
/				199-100

Start at the highest address in the area to be cleared. Enter the Op Code once for each 100 digits to be cleared.

95.

Op	A	B	d
M	0 3 0	2 3 0	
M			
M			

All three fields in the input area are moved, without change to the print area.

96.

Op	A	B	d	
A	0 1 0	2 5 5		251-255
A	0 1 0			246-250
A	0 1 0			241-245

In this case we must repeat the A-address each time since we want the same number--not consecutive numbers.

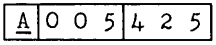
97.

Op	A	B	d	
M	0 5 9	2 3 0		221-230
M	0 4 9			211-220
M	0 2 9			201-210

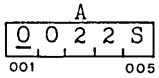
This is also partial chaining as in Problem 96. The A-fields were not in sequence and could not be chained.

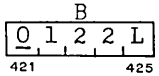
UNIT IV QUIZ

98. Instruction

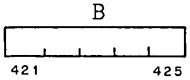


Before

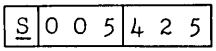




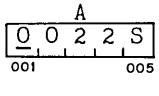
98. After

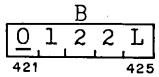


99. Instruction

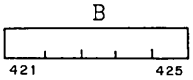


Before





99. After



100. Code the following problem. Read a card, select into pocket #8/2. Input card contains data from Column 1-78. Move the data two spaces to the right and output a card, selecting into pocket #4.

100.

Op	A	B	d

101. How many storage positions are occupied by the instructions used to accomplish problem 100?

101. _____

102. Show the chained instructions for the following:

MOVE:

From	To
612-617	201-206
619-624	207-212
625-630	213-218

102.

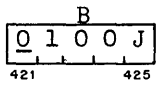
Op	A	B

UNIT IV QUIZ

ANSWERS

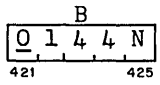
REMINDERS

98. After



Refer to page 58

99. After



Refer to page 63

100.

Op	A	B	d
/	0 8 0		
,	0 0 1	1 0 3	
1			
K			2
M	0 7 8	1 8 0	
4			
K			4

Refer to page 67

101. 24

102.

Op	A	B
M	6 3 0	2 1 8
M		
M	6 1 7	

Refer to page 79

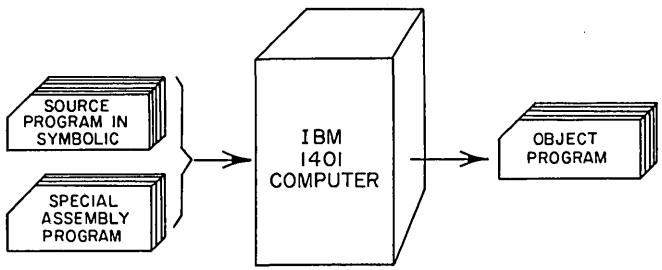
UNIT V

Lesson 20

SYMBOLIC PROGRAMMING: The computer follows a set of one-digit operation codes; other codes are used to express memory addresses higher than 999; still others denote plus or minus signs. Such codes are hard to remember and lead to mistakes. Therefore, a system of symbolic coding has been developed. This system permits the programmer to write his program with symbolic abbreviations in place of codes. The symbolic abbreviations, called mnemonics, are then converted to operation codes by a special program. Although the process takes more time, the mnemonics are easier to work with than operation codes.

When the programmer wishes to address a memory position higher than 999, he must convert the first two or three digits of the address to a letter or special character. In the symbolic coding system he can write addresses with four or five digits; the assembly program converts them to three digit addresses, with the proper code in the high-order position.

A program written in symbolic language must first be converted to machine language by a special assembly program before it can perform the steps for which it was written:



The object program is the program deck with actual machine language instructions required to do the task of processing data.

EXAMPLES:

Processing Step	Symbolic Code				Converts to Machine Language			
	Mnemonic	A-Add.	B-Add.	d	Op	A	B	d
CLEAR STORAGE	CS	0479			⟨	479		
SET WORD MARKS	SW	0055	0089		,	055	089	
MOVE CHARACTERS	MCW	0590	0690		M	590	690	
PUNCH A CARD	P				4			
NO-OPERATION	NOP				N			
ADD*	A	1324	2222		A	T24	K22	
ZERO AND ADD*	ZA	0974	1529		?	974	V29	
BRANCH IF W/M OR ZONE	BWZ	0644	0354	1	V	644	354	1
LOAD CHARACTERS	LCA	0590	0690		L	590	690	

* Note that the 4-digit addresses were converted to 3-digit addresses by changing the two high-order digits to the proper code (see Lesson 6).

TABLE OF MNEMONIC CODES

Operation	Programmer Writes Mnemonic Code	Converts to Op Code
ADD	A	A
ZERO AND ADD	ZA	?
SUBTRACT	S	S
ZERO AND SUBTRACT	ZS	I
BRANCH	B	B
BRANCH IF WORD MARK OR ZONE	BWZ	V
NO-OPERATION	NOP	N
COMPARE	C	C
HALT	H	.
CLEAR STORAGE	CS	⟨
SET WORD MARK	SW	,
CLEAR WORD MARK	CW	¤
MOVE CHARACTERS TO WORD MARK	MCW	M
MOVE CHARACTERS AND SUPPRESS ZEROS	MCS	Z
MOVE NUMERICAL	MN	D
MOVE ZONE	MZ	Y
LOAD CHARACTERS TO WORD MARK	LCA	L
READ A CARD	R	I
PUNCH A CARD	P	4
PRINT A LINE (WRITE)	W	2
STACKER SELECT	SS	K

PROBLEMS:

103. Write an instruction using symbolic coding to MOVE CHARACTERS AND SUPPRESS ZEROS from 0080 to 2420.

Mnemonic	A	B	d

104. Write an instruction to ZERO AND ADD a number from 0160 into 0960.

Mnemonic	A	B	d

105. Write two instructions to COMPARE 1040 to 0010; and BRANCH, if equal, to 0560.

Mnemonic	A	B	d

106. The program which is produced in machine language from an assembly program is called an _____ program.

107. Write a short partial program in symbolic coding to read a card; compare field A with field B; if field A B, punch a card containing both fields A and B; otherwise produce a print-out listing field B only.

Input Card		Output Card		Print Positions	
A	1- 5	A	1- 5	B	4-8
B	21-25	B	76-80		

Label	Mnemonic	A	B	d

UNIT V

Lesson 20

NOTES AND ANSWERS

103.

Mnemonic	A	B	d
M C S	0 0 8 0	2 4 2 0	

Four digit addresses may be entered as four digits. The assembly program will convert the 2420 to M20.

104.

Z A	0 1 6 0	0 9 6 0	

105.

C	1 0 4 0	0 0 1 0	
B	0 5 6 0		S

The d-modifier must be entered the same way as in actual machine language.

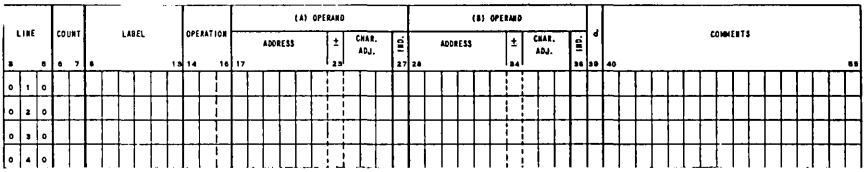
106. OBJECT

The program produced by the assembly program (object program) is like the one the programmer might write in machine language.

107.

Label	Mnemonic	A	B	d
	S W	0 0 0 1	0 0 2 1	
START	R			
	C	0 0 0 5	0 0 2 5	
	B	PUNCH		T
	M C W	0 0 2 5	0 2 0 8	
	W			
	B	START		
PUNCH	M C W	0 0 0 5	0 1 0 5	
	M C W	0 0 2 5	0 1 8 0	
	P			
	B	START		

SYMBOLIC PROGRAM CODING SHEET: To simplify punching the source program cards, a sheet is used which has columns corresponding to the card columns to be punched. Each line represents a single instruction and will be punched in a single card.



LINE (Col. 3-5) This space is used to number each line on the page. Start a new set of numbers on each page. (Columns 1-2 will contain the page number.)

COUNT (Col. 6-7) This space is used to indicate the number of characters following the operation. (Used only for constants or work areas to be described later.)

LABEL (Col. 8-13) This space may be filled in with a descriptive word to identify the particular instruction or data word on this line. The assembly program will set up a table of labels with their equivalent memory addresses. The programmer may then use that label in other instructions as an A or B-address. The assembly program will replace the label with its corresponding address in the object program. The label will then be associated with the location of the op code of an instruction word and the right-most location of a data word.

(A)OPERAND (Col. 17-27) This is A-address information

ADDRESS (Col. 17-22) The programmer may use a four digit actual address, or he may use a label, if the label appears elsewhere in the label column. Always start in Column 17 and use zeros if the memory address occupies less than 4 digits. (Position 54 would be written 0054.)

± (Col. 23) This column will be explained in a later lesson.

CHAR. ADJ. (Col. 24-26) Character adjustment will be explained later.

IND. (Col. 27) Indexing will be explained later.

(B)OPERAND (Col. 28-38) Same as (A) OPERAND except that it carries the B-address part of the instruction.

d (Col. 39) This space is used for the digit modifier.

COMMENTS (Col. 40-55) This field may be used to write any explanatory comment the programmer desires to use. When the assembly program converts the data to machine coding and punches the program cards, it also prints a listing showing the instructions and comments. Comments are helpful, both to the programmer and others who may work with the program.

EXAMPLES:

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	(A) ±	(A) CHAR. ADJ.	(A) IND.	(B) OPERAND ADDRESS	(B) ±	(B) CHAR. ADJ.	(B) IND.	d	COMMENTS
010			SW	0001				0004					WM FOR PTNR,NAME
020			SW	0016				0022					WM FOR NR, COST
030		START	R										READ A CARD
040			MCW	0003				0106					MOVE PTNR TO PCH
050			MCW	0018				0112					MOVE NR TO PUNCH
060			MCW	0027				0120					MOVE COST TO PCH
070			P										PUNCH A CARD
080			MCW	0015				0220					MOVE NAME TO PRT
090			MCW	0018				0232					MOVE NR TO PRINT
100			W										PRINT A LINE
110			CS	0300									CLEAR 300 TO 200
120			CS										
130			CS	START				0199					CLEAR TO 100&RPT
140													
150													

This program may be recognized as the same one used as an example in Lesson 11.

PROBLEMS:

108. The word used as a substitute for an instruction address is called a _____.

109. The A-operand will convert to an A-_____.

110. A mnemonic operation code may contain up to _____ alphabetic characters.

111. The page number will be punched in Columns _____ and _____.

112. A label in the A-address field is written starting in Column _____ and may contain no more than _____ characters.

113. A label may be written as many times as necessary in the A or B-address fields, but it may be written only _____ in the Label field.

114. Columns _____ through _____ contain the Label.

115. What is wrong with the following? (Write answer here)

LABEL	OP	A-Address	B-Address	
	--	HOLD	--	_____
	--	--	--	_____
	--	--	END 2	_____
	--	END	--	_____
	--	--	HOLD	_____
HOLD	--	--		_____
END	--	--		_____.
END2	--	--		

UNIT V

Lesson 21

NOTES AND ANSWERS

108. LABEL — The assembly will assign a 3-digit address to each label.

109. ADDRESS — All of the fields of the A or B operand are used to generate an A or B address.

110. 3 — Some mnemonics are like the operation codes, but others are abbreviations of the operation.

111. 1 and 2

112. 17 6

113. once — If a label is used in one of the Address fields and is not used or defined in the Label field, the program will not assemble properly.

114. 8 13 — A label may be no more than six characters in length and the first character must be alphabetic.

115. END 2 in the B-address is not identical to END2 in the Label. To be correct, they must be identical in every way.

CONSTANTS AND WORK AREAS: In our sample program (Lesson 18) we were required to add $100 to the manager's commission. To accomplish this we set-up a fixed data field in memory positions 689-691, containing the number 100. When numbers or other data are needed, and are not supplied on the input, they must be supplied by the programmer. Such fixed data fields are called constants.

In the same program we needed a temporary storage field in which to summarize totals. Such fields are called work areas. These are usually blank, or zero fields with word marks in the proper position or positions. They may be used in many ways by clearing and resetting word marks after each use.

Constants and work areas must be considered when setting-up the program. A good flow chart will indicate where work areas and constants are needed. When coding, these fields should be listed on a separate coding sheet. Additional fields can be noted on the same sheet if the need becomes evident during coding of the program.

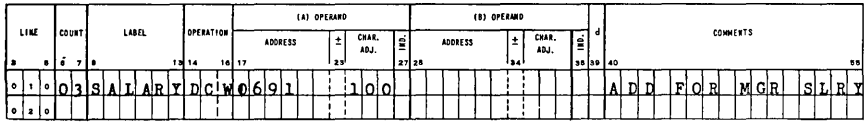
Under the Symbolic Programming System, such areas are specified by a special set of mnemonics which have no equivalent operation code:

DEFINE CONSTANT WITH W/M (DCW) This mnemonic indicates that the data following it is to be set-up in memory as a constant with a W/M in the high-order position.

DEFINE CONSTANT (DC) This mnemonic is like DCW except that no W/M will be set.

EXAMPLES:

1.

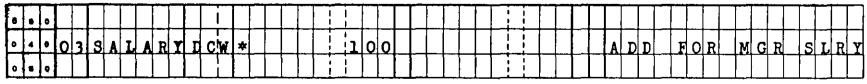


The "Count" columns (6-7) indicate the number of digits.
The actual, low-order address is in Columns 17-20.
Constants may begin in Column 24 and continue to Column 55, if necessary, provided the count columns indicate the correct number of digits.
A word mark will automatically be placed in position 689.

After assembly: 1 0 0

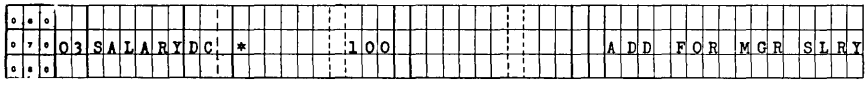
689 690 691

2.



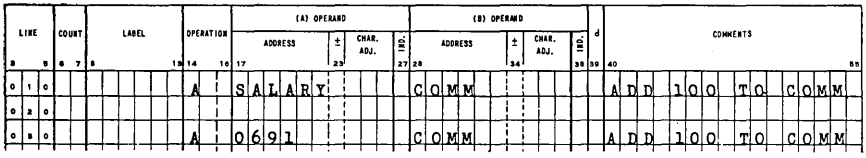
When an asterisk (*) is placed in Column 17, the assembly program will assign the memory address. As the program is assembled, the program will assign the next three available memory positions for the constant.

3.



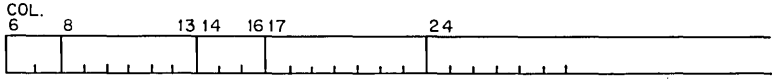
The symbolic code (DC) operates exactly like (DCW) except that no W/M is placed in the high-order position.

4. When it is necessary to refer to the constant in an instruction, either the label or actual address may be used.



PROBLEMS:

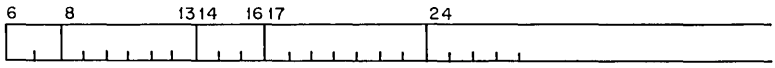
116. Define a constant with a word mark date SEPb20bl962 (b=blank). Give it a label DATE, and let the program assign the address.



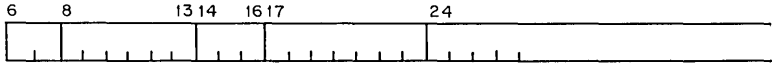
117. Write a mnemonic instruction to move the DATE to 1620.

Mnemonic	A	B	d

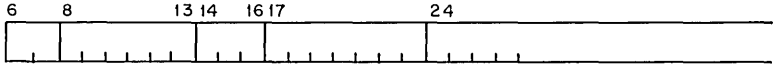
118. Set up a field of 10 zeros at 0600-0609 without a word mark.



119. Set up a 3-position constant, labelled HOLD, consisting of 001.



120. Set up a 7-position constant (with W/M) in the following format: $bbb.bb. Call it CONST.

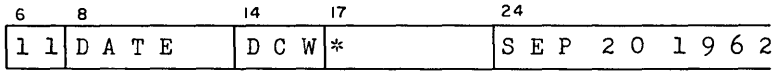


UNIT V

Lesson 22

NOTES AND ANSWERS

116.



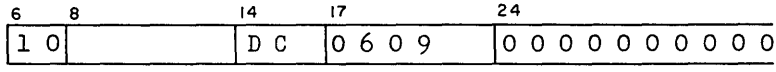
An asterisk tells the assembly program to assign the next 11 memory positions available.

117.

Mnemonic	A	B	d
M C W	D A T E	1 6 2 0	

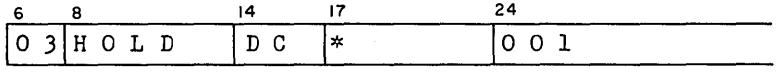
The label is used in place of the address since the address will not be known until after assembly.

118.



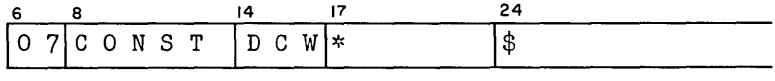
No label was specified and is not needed since the address of the zeros is known.

119.



There was no mention of word mark in the problem, although usually such constants are set up with word marks.

120



ASSEMBLING THE PROGRAM: The cards punched from the Symbolic Programming Sheet are read into the computer following the assembly program. The assembly program does the following:

1. Assigns a memory address to each constant, work area and instruction. It keeps track of the number of characters in each case to insure that memory is systematically filled.

2. Sets up a dictionary (table) of labels used with corresponding memory addresses.

3. Converts mnemonic codes to one-digit operation codes.

4. Converts four-digit addresses to three-digit addresses.

Some additional mnemonic codes are used to give the assembly program necessary information:

DEFINE SYMBOL (DS) This mnemonic operation code is used to define symbolic addresses used in the program. If the program wishes to associate a label with a particular machine address, this mnemonic will permit the assembly program to enter the label and address in the dictionary of labels. Used to define work areas.

ORIGIN (ORG) This mnemonic tells the assembly program to begin assigning memory addresses at the address given in the (A) Operand. This line of coding should <u>precede</u> the rest of the program. If the (ORG) mnemonic is <u>not</u> used, the assembly will begin assigning addresses at memory position 333.

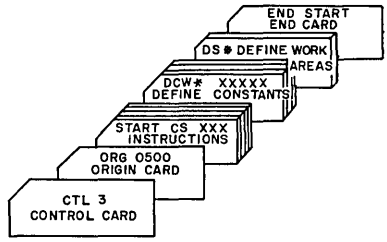
CONTROL (CTL) This mnemonic code indicates how many memory positions are available in the computer. This <u>must</u> be the first card in the deck.

1400	positions	of	storage	-	Col.	17	must	contain 1
2000	"	"	"		"	"	"	" 2
4000	"	"	"		"	"	"	" 3
8000	"	"	"		"	"	"	" 4
12000	"	"	"		"	"	"	" 5
16000	"	"	"		"	"	"	" 6

Column 18 uses the same code to describe the machine on which <u>compilation</u> will take place.

END (END) This mnemonic operation code must be coded as the last card in the program. It tells the assembly program that all symbolic program instructions have been processed. The (A) Operand should contain the address (or label) of the first instruction.

A TYPICAL SYMBOLIC PROGRAM DECK



EXAMPLES:

Assign work areas for accumulating total sales and total commissions at 325-349 and 310-324 respectively:

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
010	25	TOTSLS	DS	0349									WA FOR TOTAL SLS
020	15	TOTCOM	DS	0324									WA FOR TOTAL COM

Same without giving actual, low-order addresses:

140													
050	25	TOTSLS	DS	*									WA FOR TOTAL SLS
060	15	TOTCOM	DS	*									WA FOR TOTAL COM

A program starting at storage position 0500 for a computer with 4000 memory positions:

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
010			CTL	3									
020			ORG	0500									

The END card for a program that has the first instruction labelled "START":

080													
090			END	START									
100													

PROBLEMS:

121. The first card in every symbolic deck must have a mnemonic code _____.

122. If the origin (ORG) code is not used in the program, the assembly program will start assigning addresses at position _____.

123. The last card in the symbolic deck must have a mnemonic code _____.

124.

Mnemonic	A
C T L	5
O R G	2 0 5 0

(a) The computer contains _____ memory positions.

(b) Assembly program starts assigning addresses starting at location _____.

125. Write the Op Codes and A-addresses for the first two cards in the program if assembly is to start at location 601 and 4000 storage positions are available.

Mnemonic	A

UNIT V

Lesson 23

NOTES AND ANSWERS

121. C T L

122. 0 3 3 3

The control card is necessary to tell the assembly program how large is the memory capacity of IBM 1401 in which the object program will be used. The origin card will permit the programmer to start his program where he wishes.

123. E N D

This card tells the assembly program to stop assembling and the A-operand indicates which instruction should be the first executed.

124. (a) 12000

(b) 2050

125.

Mnemonic	A
C T L	3
O R G	0 6 0 1

CHARACTER ADJUSTING: The programmer may branch to an instruction without a label by using Columns 23-26 (± Char. Adj.). By this technique the programmer can indicate a branch to a label ahead of or behind the instruction being sought and adding or subtracting the number of characters separating the label from the instruction.

Characters are counted as they will appear in the object program (machine language). Each operation code equals 1, each operand (A or B) equals 3, and the digit modifier equals 1.

EXAMPLE:

Assume "START" = 333 Object Program

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d
010		START	CS	0080								
020			SW	0001				0020				
030			R									
040			MCW	0010				0210				
050			MCW	0024				0244				
060			W									
070			B	START	+	011						
080												

333	/	080										
337	,	001	020									
344	R											
345	M	010	210									
352	M	024	244									
359	W											
360	B	344										

The branch instruction operand of START + 011 tells the assembly program to substitute an A-address of 333 + 11 or 344.

The possibility of error in computing the number of characters to adjust is considerable. The programmer should use this technique sparingly.

The following symbolic program illustrates the use of character adjustment to specify addresses for which no label has been given.

	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	(A) OPERAND +/-	(A) OPERAND CHAR. ADJ.	(A) OPERAND IND	(B) OPERAND ADDRESS	(B) OPERAND +/-	(B) OPERAND CHAR. ADJ.	(B) OPERAND IND	d
		START	CS	0300								
			CS									
			CS									
			SW	0001				0009				
			SW	0014				0020				
			R									
			C	0003				CONEND				
			B	EXIT								S
(1)			BWZ	EXIT	-	008		0001				K
			MCW	0008				0219				
			MCW	0013				0230				
			MCW	0019				0255				
			W									
			CS	0299								
(2)			B	START	+	020						
*		EXIT	MCW	CONEND				0252				
			W									
(3)			H	EXIT	+	008						
	03	CONEND	DCW	END								

*EXIT SUB-ROUTINE: The term sub-routine is frequently used to specify a block of instructions which operate apart from the main "loop" or program.

(1), (2), and (3) are examples of character adjusting.

PROBLEMS:

LABEL	MNEMONIC	A	B	d
	C S	0 0 8 0		
	C S	0 2 9 9		
	S W	0 0 0 1	0 0 1 4	
	S W	0 0 2 0		
S T A R T	R			
	M C W	0 0 1 3	0 2 1 3	
	M C W	D A T E	0 2 9 9	
	C	D A T E	1 0 4 4	
	B	E X I T		S
	M C W	0 0 1 9	0 2 4 0	

In the partial program shown above, compute the number for character adjustment as follows:

126. Refer to the COMPARE instruction START +_____

127. Refer to the FIRST SET W/M instruction START - _____

128. Refer to the LAST MOVE instruction START + _____

129. Refer to the FIRST CLEAR STORAGE instruction START - _____

130. Refer to the BRANCH instruction START + _____

UNIT V

Lesson 24

NOTES AND ANSWERS

126. + 0 1 5

Each mnemonic is counted as 1, each operand as 3, and each digit modifier as 1.

127. - 0 1 1

For minus adjustment count backwards, starting with the digit modifier and work back to each mnemonic in turn.

128. + 0 2 7

129. - 0 2 0

130. + 0 2 2

A SAMPLE PROGRAM: In Lesson 18 we flow-charted and coded a program for an insurance company with several field offices. In this lesson we will code the same program using Symbolic Programming and chaining.

Pages 73-74 contain the statement of the problem and pages 76-77 contain the flow chart and machine language coding of the problem. Review the problem and the flow chart before turning to the next page for the symbolic coding of the same problem.

The notes below should aid you in interpreting the symbolic program.

NOTE:

1. Lines are numbered by 10's to enable the programmer to insert instructions, if necessary.

2. No indication of W/M's is needed.

3. Instructions on Lines 270 and 370 utilize character adjustment.

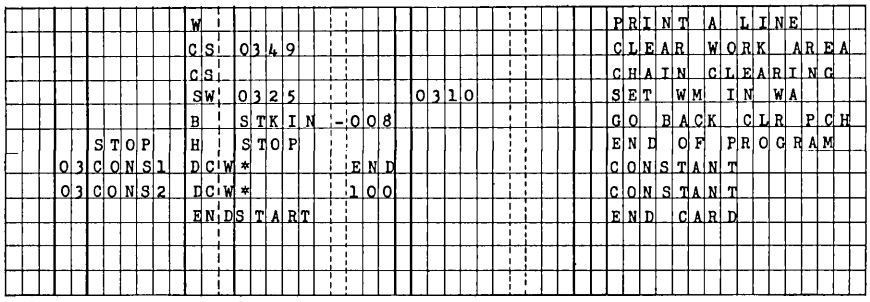
4. The HALT instruction is given an A-address equal to its own label so that if someone presses the "START" key after the HALT the program will immediately return to the HALT.

5. Lines 030 through 060 show a chaining sequence.

6. Labels used in the program, such as STKIN, STKOUT are made up by the programmer and should have some meaning to him. In this case, the meaning is STACK IN and STACK OUT. Any other labels would have been just as acceptable to the program.

7. Lines 390 and 400 show the use of constants in a program.

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
010			CTL3										CONTROL CARD
020			ORG0500										START AT 500
030		START	CS 0349										CLEAR WA
040			CS										CHAIN CLEARING
050			CS										
060			CS										
070			SW 0001					0004					WM OFF NR-SLS NR
080			SW 0007					0041					WM NAME AND SALE
090			SW 0051										WM COMMISSION
100			SW 0325					0310					WM WORK AREAS
110		READ	R										READ A CARD
120			BWZSTKIN					0004				K	TEST FOR MGR CRD
130			C CONS1					0003					TEST FOR END
140			B STOP									S	GO TO END IF EQU
150			MN 0005					0102					MOVE SLS NR WO Z
160			MN										CHAIN 2ND DIGIT
170			MCW0030					0128					MOVE SLSMN NAME
180			MCS0056					0136					COMMISSION
190			A 0046					0349					GROSS SLS TO TOT
200			A 0056					0324					COM TO TOTAL
			P										PUNCH A CARD
			BWZSTKOUT					0004				K	STACK MGRS CARD
			CS 0180										CLEAR PUNCH AREA
			B READ										GO BACK TO READ
		STKIN	K									1	STACK MGRS CARD
			A CONS2					0046					ADD 100 TO MGR
			B READ	+009									GO BACK TO PROC
		STKOUT	K									4	STACK MGRS CARD
			MCW0003					0212					MOVE OFF NR
			MCS0349					0225					MOVE TOT SALES
			S 0324					0349					SUB COMMISSION
			MCS0349					0235					MOVE NET SALES



PROBLEM:

131. Using symbolic language, write a program to do the following:

Use an 8000 position memory

Start at memory position 1000

Read cards containing the following information:

Col.	1-10	Stock Numbers
"	12-20	Price
"	22-30	Quantity Sold
"	32-40	Quantity On Hand

Punch a card for each card read with the following information:

Col.	1-10	Stock Number
"	22-30	Price

Print a line for each card showing:

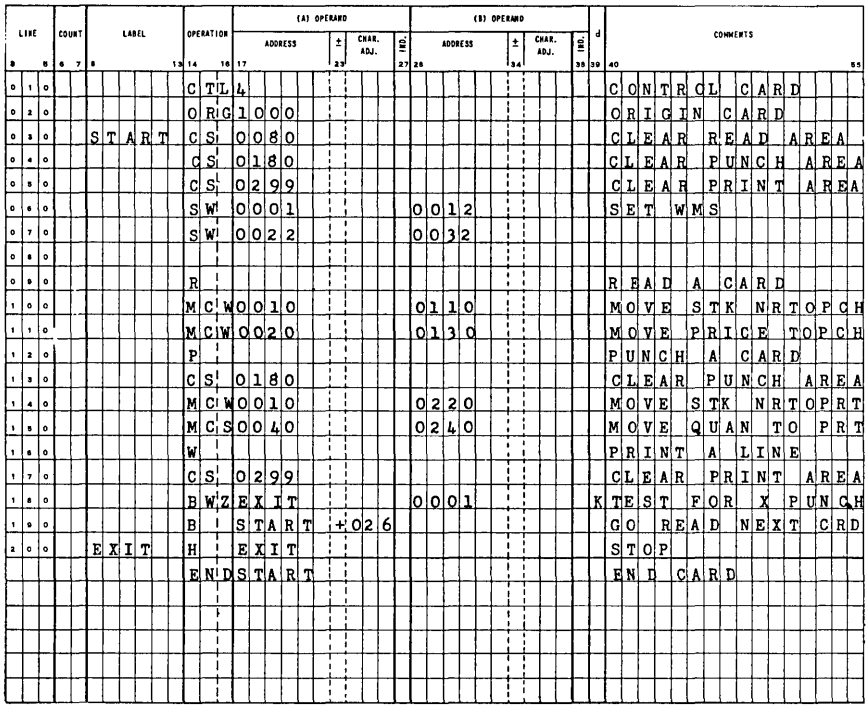
Print Positions	11-20	Stock Number
" "	32-40	Quantity On Hand (without leading zeros)

Stop the program when you finish the last card which has an "eleven" or "X" zone punch in Col. 1 over the first digit of the Stock Number.

UNIT V

Lesson 25

NOTES AND ANSWERS



LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	COMMENTS
010			CTL	4								CONTROL CARD
020			ORG	1000								ORIGIN CARD
030		START	CS	0080								CLEAR READ AREA
040			CS	0180								CLEAR PUNCH AREA
050			CS	0299								CLEAR PRINT AREA
060			SW	0001				0012				SET WMS
070			SW	0022				0032				
080												
090			R									READ A CARD
100			MCW	0010				0110				MOVE STK NR TO PCH
110			MCW	0020				0130				MOVE PRICE TO PCH
120			P									PUNCH A CARD
130			CS	0180								CLEAR PUNCH AREA
140			MCW	0010				0220				MOVE STK NR TO PRT
150			MCS	0040				0240				MOVE QUAN TO PRT
160			W									PRINT A LINE
170			CS	0299								CLEAR PRINT AREA
180			BWZ	EXIT				0001				K TEST FOR X PUNCH
190			B	START	+	026						GO READ NEXT CRD
200		EXIT	H	EXIT								STOP
			END	START								END CARD

Your program may differ slightly from this, but it must do what this program does to meet specifications. Do not proceed to the next unit until you understand all of the steps in this program.

132. Write an instruction (in symbolic) to load data from location HOLD to AREA:

Mnemonic	A	B	d

133. Set up a 10-position constant, labelled CHECK, in the following format (W/M required): $bb,bbb.bb

Count	Label	Op	17 24

134. Set up a one-position constant of 3, labelled CTR.

Count	Label	Op	17 24

135. Set up a 99-position work area, labelled WORK, starting in storage location 501.

Count	Label	Op	17 24

136. Set up a 26-position constant (with W/M), labelled HEAD, in the following format: bbSTK.NO.bbONbHANDbb LOANED

Count	Label	Op	17 24

137.

Op	A
C T L	3
O R G	0 5 0 0

Explain the meaning of these two instructions.

ANSWERS

REMINDERS

132.

Mnemonic	A	B	d
L C A	H O L D	A R E A	

Refer to page 86

133.

Count	Label	Op	17 24 27 31
1 0	C H E C K	D C W	* $, .

Refer to page 93

134.

Count	Label	Op	17 24
0 1	C T R	D C	* 3

Refer to page 93

135.

Count	Label	Op	17 24
9 9	W O R K	D S	0 5 9 9

Refer to page 97

136.

Count	Label	Op	17 24 32
2 6	H E A D	D C W	* S T K . N O .

O N H A N D L O A N E D

35 41 44 49

Refer to page 93

137. C T L 3 means that the 1401 to be used has 4000 memory positions.

O R G 0 5 0 0 means that the assembly program will begin assigning memory addresses starting at location 500.

Refer to page 97

UNIT VI

Lesson 26

VARIATIONS OF READ, PRINT AND PUNCH: The IBM 1401 has several features which increase the efficiency of the card read-punch and the printer. They increase speed by permitting operations to occur simultaneously. These features are utilized through the following instructions:

WRITE AND READ:

Op	Mnemonic
2	W R

This instruction combines the operation of reading a card and printing a line in a single command. Printing and reading are not performed simultaneously. A line is printed first and the card reader is signalled to begin. When the printer finishes, the card is read.

READ AND PUNCH:

Op	Mnemonic
5	R P

This instruction permits reading a card and punching a card to occur at the same time. Time is saved, since the reading is completed during the time the punch is operating.

WRITE AND PUNCH:

Op	Mnemonic
6	W P

This instruction combines the printing and punching operations. As soon as printing is completed, the punch will punch a card.

WRITE, READ AND PUNCH:

Op	Mnemonic
7	W R P

This instruction directs the computer to print a line from the print area, read a card to the input area, and punch a card from the punch area. Printing occurs first, followed by simultaneous reading and punching.

NOTE: None of the above instructions requires an A or B-address or a d-modifier. If the programmer supplies an A-address, the program will branch to the address given.

EXAMPLES:

Follow a simple program to read a card, move data to the print area and print a line.

Label	Oper.	A-Address	±	Char. Adj.	B-Address	±	Char. Adj.
S T A R T	R						
	M C W	0 0 8 0			0 2 8 0		
	W						
	B	S T A R T					

Using the write and read instruction we would write:

S T A R T	R						
	M C W	0 0 8 0			0 2 8 0		
	W R						
	B	S T A R T	+	0 0 1			

We must still read the first card with a read instruction in order to have data to work on, but we save time and program steps by using multiple instructions for subsequent processes. The return to START + 001 takes the program into a LOOP.

PROBLEM:

138. Write a small program, using symbolic language, to read a card, move data field from columns 1-10 into punch area 1-10, and into print area 11-20. Move data field from columns 11-15 into punch area 21-25 and print area 26-30. Punch a card and select stacker #4. Print a line. Go back to read another card. (Do not clear storage, set word marks or provide for a program stop.) The program should require seven steps. (The mnemonic for SELECT STACKER is (SS) and its A-address may be used for the branch address.)

Label	Oper.	A-Address	+ -	Char. Adj.	B-Address	+ -	Char. Adj.	d

UNIT VI

Lesson 26

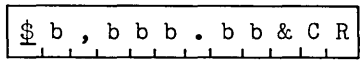
NOTES AND ANSWERS

PROBLEM 138:

Label	Oper.	A-Address	±	Char. Adj.	B-Address	±	Char. Adj.	d
S T A R T	R							
	M C W	0 0 1 0			0 1 1 0			
	M C W	0 0 1 0			0 2 2 0			
	M C W	0 0 1 5			0 1 2 5			
	M C W	0 0 1 5			0 2 3 0			
	W R P							
	S S	S T A R T	+	0 0 1				4

EDITING AND ARRANGING PRINTOUTS: Amounts are usually punched in data cards without dollar signs or decimal points since the computer does not need these symbols. However, when data is printed it is desirable to have symbols and spacing to assist the reader. The IBM 1401 has a powerful aid to the programmer in the MOVE CHARACTER AND EDIT instruction.

Two fields are referenced by the instruction. They are the data field and the control field. The control field is a constant, entered in the program via a DCW instruction. This field indicates how the data is to appear in the edited print-line. It must show commas, decimal points, dollar signs and other symbols. The position of each data digit is indicated by a blank (b) as follows:



This Control Word will permit printing a dollar amount of up to 9,999.99 and will cause any zeros to the left of any valid data to be replaced by blanks. The ampersand (&) will be replaced by a blank and the CR will only be printed if the field is negative. (CR represents the accounting abbreviation for CREDIT.)

First the Control Word is moved to the print area by an MCW instruction and then the MOVE CHARACTER AND EDIT is coded.

MOVE CHARACTER AND EDIT:

Op	Mnemonic	A-Addr.	B-Addr.
E	M C E	X X X X	X X X X

The A-address indicates the data to be printed; the B-address indicates the address of the Control Word in the print area. Both fields are addressed in the low-order position and the B-field W/M stops the operation. The W/M is then erased.

EXAMPLES:

Control Field | $ b , b b b . b b & C R | (B-Address)

Data Field | 0 0 2 4 9 5 | (A-Address)

Instructions

Op	A	B	
M C W	0 5 1 1	0 2 2 1	Move control word to print area
M C E	0 0 2 5	0 2 2 1	Move and edit the data

Printed Result | $ b b 2 4 . 9 5 | (B-Address after edit)

Or:

Data Field | 0 0 2 4 9 N | Data field is negative

Printed Result | $ b b 2 4 . 9 5 C R |

Other numeric, alphabetic or special characters may be used as follows:

Control Word: | $ b b b , b b b . b b & * * |

Data Field: | 0 0 1 6 5 9 7 2 |

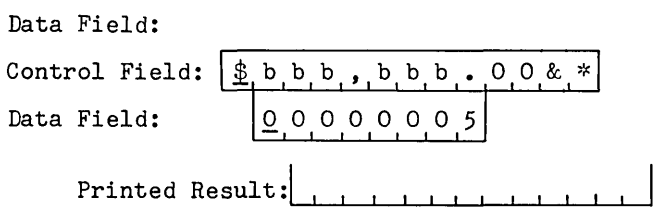
Printed Result: | $ b 1 , 6 5 9 . 7 2 b * * |

NOTE: An optional feature of the IBM 1401, called EXPANDED PRINT, permits still greater edit capability. This text will not cover expanded print but the student may refer to IBM 1401 Reference Manual.

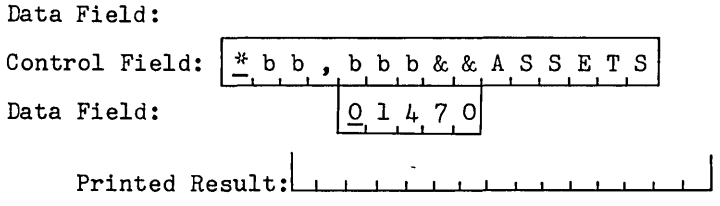
Notice in the example above that the data word is moved on top of the control word. The internal editing operation will place the individual parts of the data word into the blank areas designated by the control word. Also note that an asterisk (*) prints exactly as it is placed in the control word.

PROBLEMS:

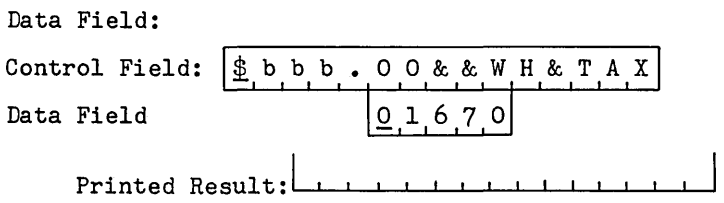
139. Assume the following control field and data field:



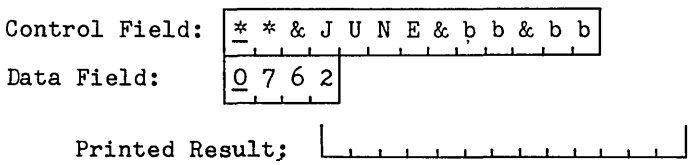
140. Assume the following control field and data field:



141. Assume the following control field and data field:



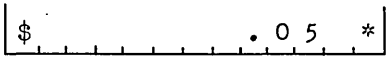
142. Assume the following control field and data field:



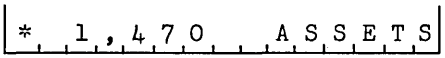
UNIT VI

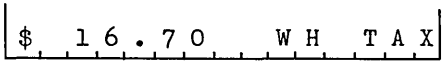
Lesson 27

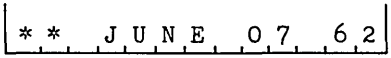
NOTES AND ANSWERS

139. 

By inserting zeros in place of b's in the control field--leading zeros may be retained.

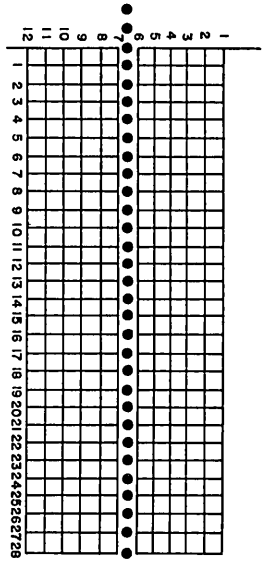
140. 

141. 

142. 

PRINTER CONTROLS: When printing out reports, it is necessary to space the lines of print for a more attractive and readable copy and to permit changing pages. Control of the movement of paper is handled in two ways.

1. A carriage tape is mounted on sprocket wheels on the printer. By punching holes in the tape, spacing may be controlled.



The tape has 12 channels or columns marked across the top, and usually 140 line numbers, corresponding to print lines on a page, marked lengthwise. Punching a rectangular hole in a particular channel, at a specific line number, permits the programmer to direct a control instruction to the printer referencing the channel punch.

CONTROL CARRIAGE:

Op	Mnemonic	d
F	C C	X

Only the operation and digit modifier is used.

By choosing a d-modifier from the following table, the carriage tape may be used to direct the spacing of reports:

If d =	1	skip <u>immediately</u> to Channel 1
	2	2
	3	3
	4	4
	5	5
	6	6
	7	7
	8	8
	9	9
	0	10
	#	11
	@	12
If d =	A	skip <u>after printing</u> to Channel 1
	B	2
	C	3
	D	4
	E	5
	F	6
	G	7
	H	8
	I	9
	?	10
	.	11
	¤	12

Use of any of these d-modifiers requires that holes be punched in the channel being referenced. The printer will skip to the line in which the channel is punched.

2. A second form of control may be programmed, in which the carriage tape is not referenced. One, two, or three lines may be skipped by the use of the following d-modifiers:

If d = J skip 1 space(s) immediately

= K 2

= L 3

If d = / skip 1 space(s) after printing

= S 2

= T 3

EXAMPLES:

1. Skip to Channel 4 immediately

	A-Operand				B-Operand				
Label	Oper. Address	±	Ch.Adj	I/D	Address	±	Ch.Adj	I/D	d
	C C								4

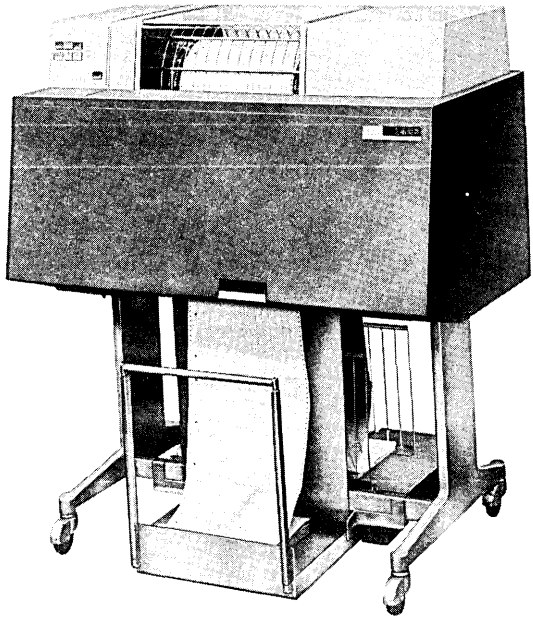
2. Skip to Channel 4 after printing

C C		D

3. Skip 2 spaces immediately and then branch to 0590

C C	0 5 9 0	K

The control carriage instruction may convert to control carriage and branch, by adding an A-address.



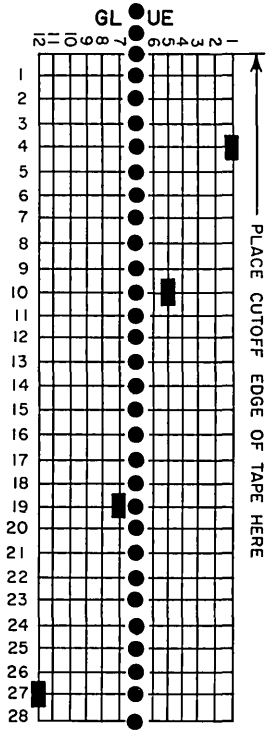
IBM 1403 Printer

The 1403 printer prints at a maximum of 600 lines per minute and has a print span of 100 positions or 132 positions. Horizontal spacing is 10 characters per inch. Any one of 48 characters -- 26 alphabetic, 10 numerical and 12 special -- can print in each print position. The twelfth special character is the + symbol or the record mark, depending upon the 1401 model. Vertical spacing can be six or eight lines to the inch and is under operator control.

The printer can be equipped with optional features to print numbers alone at a rate of 1,285 lines a minute. It has the unique faculty of being able to skip over blank lines on printed forms at the rate of seventy-five inches per second.

PROBLEMS:

143. The tape illustrated below has four punches in it. Indicate the channel number and line number for each punch from top to bottom.



	Channel	Line
143.(a)		
(b)		
(c)		
(d)		

144. Write a control carriage instruction to skip to each of the four punches <u>immediately</u>.

	Mnemonic	d
144.(a)		
(b)		
(c)		
(d)		

145. Write a control carriage instruction to skip 3 lines <u>after printing</u> and branch to 0540.

	Mnemonic	A-Address	d
145.			

UNIT VI

Lesson 28

NOTES AND ANSWERS

143.

	Channel	Line
(a)	1	4
(b)	5	10
(c)	7	19
(d)	12	27

Referencing Channel 1 in a control carriage instruction will cause a skip to Line 4. Channel 5 causes a skip to Line 10, etc.

144.

	Mnemonic	d
(a)	C C	1
(b)	C C	5
(c)	C C	7
(d)	C C	@

The control carriage instruction and the move character and edit instruction may be used to provide attractive and readable print formats.

145.

Mnemonic	A-Address	d
C C	0 5 4 0	T

The control carriage instruction may be used to branch after completing the skip by adding an A-address.

146. Write instructions to accomplish the following actions:

(a) Delayed double space

(b) Write a line and punch a card

(c) Unconditional branch to START

146.

Op	A	B	d	
				(a)
				(b)
				(c)

147. Instruction

Op	A	What is the meaning of this instruction?
W R P	0 4 0 0	

147. _____

_____

_____

148. Write instructions to accomplish the following actions:

(a) Single space and return to START

(b) Write a line, read a card, triple space

(c) Write, read, punch and delayed double space

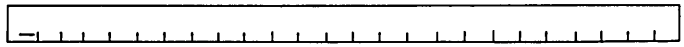
148.

Op	A	B	d	
				(a)
				(b)
				(c)

149. Set up a control field containing the following:

SALARY $XX,XXX.XX TOTAL

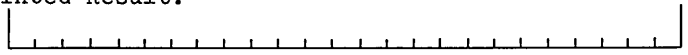
Skip two spaces between salary, amount, and total.



150. Using the control field in Problem 149 and the following data field, show the printed result:

Data Field: 1,2,8,7,4,5

Printed Result:



ANSWERS

146.

Op	A	B	d
C C			S
W P			
B	S T A R T		

REMINDERS

Refer to pages 111 and 121

147. Write, read, punch and branch to location 400.

Refer to page 112

148.

	Op	A	B	d
(a)	C C	S T A R T		J
(b)	C C			T
	W R			
(c)	C C			S
	W R P			

Delayed not used since machine automatically single spaces.

The CC could have been either before or after the WR, but the delayed instruction is preferred.

Refer to pages 111 and 121

149. S A L A R Y & & $, . & & T O T A L

Refer to page 115

150. S A L A R Y $ 1 , 2 8 7 . 4 5 T O T A L

Refer to page 115

MULTIPLICATION: Multiplication is actually a form of repetitive addition. Thus if we wish to multiply 5 x 4 we may accomplish it in either of the following ways:

$$0+5+5+5+5 = 20 \quad \text{or} \quad 0+4+4+4+4+4 = 20$$

We start with zero and add the multiplicand to it the number of times that is equal to the multiplier.

Since the IBM 1401 may not be equipped with the optional MULTIPLY feature, the programmer may resort to a special sub-routine which provides for repetitive addition to obtain multiplication products. Such routines have been prepared by IBM and are available from that company or in Manuals D24-1401 or A24-1403 published at IBM.

The programmer must remember that the product formed by this process will contain more digits than either the multiplier or multiplicand. He should therefore set up a work area large enough to contain the maximum possible product. The product work area should be equal to: the number of digits in the multiplicand + the number of digits in the multiplier + 1.

Thus: (Multiplicand) (Multiplier) (Product)
1592 x 64 = ?
(4 digits) + (2 digits) + 1 = 7 digits

= product work area

The product work area should be set to zero before beginning to multiply.

For IBM 1401 Computers equipped with the optional Multiply-Divide feature, the following rules apply.

MULTIPLY:

Op	Mnemonic	A	B
@	M	X X X X	X X X X

The A-address designates the multiplicand. The B-address designates a work area containing the multiplier in its high-order digits and sufficient additional spaces to provide for the product. The work area should have as many positions as the total of digits in the multiplier and multiplicand plus one.

To multiply a 4-digit field by a 3-digit field the work area should contain 8 digits.

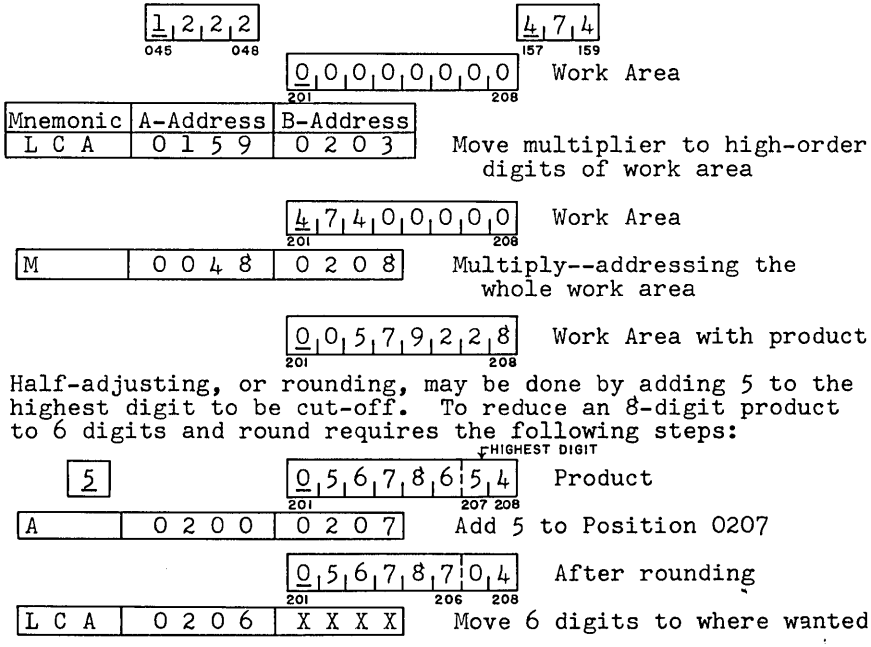
X X X X	Multiplicand	4 digits
		3 digits
X X X	Multiplier	+ 1 digit
		8 digits

Work Area [X X X 0 0 0 0 0]
Multiplier

The multiplier in the work area is destroyed as the product is formed.

If either field is negative, the minus sign must be indicated by a B-bit, without an A-bit, on the low-order digit. Multiplying two fields of the same sign will result in a plus product. Unlike fields yield a minus product.

EXAMPLE: Multiply the field located at memory position 0048 by the field located at 0159, using a work area located at 0208.



Half-adjusting, or rounding, may be done by adding 5 to the highest digit to be cut-off. To reduce an 8-digit product to 6 digits and round requires the following steps:

To adjust for decimal points, count off from the right digit a number of digits equal to the total number of decimal digits in the multiplier and multiplicand.

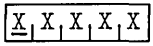
To multiply, using symbolic coding, the work area must be labelled and set up by DCW.

The product area may be set up by DCW in either of two ways:

Count	Label	Op	A
09	PROD	DCW	*
09	PROD	DCW	0501

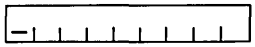
If it is set up with an asterisk (*) in the A-field, the storage location of the field is not known, so the method of placing the multiplier, shown below, is used.

EXAMPLE: Multiplicand in location HOLD



Multiplier in location MULT

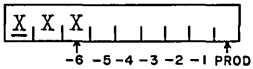
Product area set up in location PROD



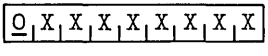
Instruction

Op	A	B
ZA	MULT	PROD-6
M	HOLD	PROD

Product Area



Multiplier set up in high-order positions of product area



Product result

No space is provided in memory for decimal points. It is up to the programmer to keep track of its position.

EXAMPLE: Multiplicand X X X . X X

Multiplier X X . X X

Product will have four decimal places.

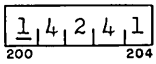
Rounding, using symbolic coding, follows the same rules that are used for machine language coding. The digit 5, which is added to the first position to be dropped, may be set up by DCW and called upon when needed. Whenever rounding is accomplished, it is extremely important to remember that the zone bits must be moved to the new last digit location.

EXAMPLE: Multiply Hours Worked (11.1) by Rate (1.25) to get Gross Pay. Round to two decimal places. Edit by placing a dollar sign in front of the first significant digit, suppressing high-order zeros and placing the decimal point in its proper place. Print out the result to start in low-order print position 75.

Ct	Label	Op	A	B	d	
07	PROD	DCW	*			Set up product area
01	FIVE	DCW	* 5			Set up const. of 5
10	EDIT	DCW	* $bbb.bb			Set up edit field
		LCA	RATE	PROD-4		
		M	HRWK	PROD		
		A	FIVE	PROD		
		MZ	PROD	PROD-1		
		LCA	EDIT	0275		
		MCE	PROD-1	0275		
		W				

PROBLEMS:

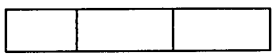
Assume multiplication of the following amounts:

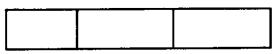
$142.41 shown in memory 

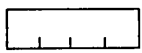
by

$ 3.50 shown in memory as 

151. How many digits should the work area contain? 151. _____

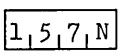
152. If the work area starts at 0400, show the LCA instruction to set up the work area. 152. 

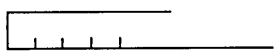
153. Show the multiply instruction. 153. 

154. To round-off two digits show the address to which 5 should be added. 154. 

155. How many digits would be located to the right of the decimal point after rounding? 155. _____

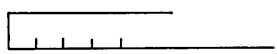
156. A = 

B = 

Indicate the result of A x B, showing sign and size of product. 156. 

157. A = 

B = 

Indicate the result of A x B, showing sign and size of product. 157. 

UNIT VII

Lesson 29

NOTES AND ANSWERS

151. 9

5 digit multiplicand
3 digit multiplier
+ 1
9 digit work area

152. | L C A | 0 2 0 7 | 0 4 0 2 |

Place the multiplier in the high-order positions.

153. | M | 0 2 0 4 | 0 4 0 8 |

The B-address should be the low-order digit address.

154. | 0,4,0,7 |

This references the left-most digit of the two digits being rounded-off.

155. 2

The multiplicand and multiplier each have 2 digits to the right of the decimal for a total of 4 digits in the product. Rounding-off 2 digits left 2 to the right of the decimal.

156. | 0,0,0,7,7,1,7,N |

Multiplication of unlike signs yields a negative product.

157. | 0,0,1,2,3,I |

Multiplication of like signs yields a positive product.

DIVISION: Division may be considered repetitive subtraction. The number of times the divisor can be subtracted from the dividend is the quotient.

Thus: $24 \div 8 = ?$

$24 - 8 = 16$ $16 - 8 = 8$ $8 - 8 = 0$
(1 time) (2 times) (3 times)

3 is the quotient

or: $25 \div 8 = ?$

$25 - 8 = 17$ $17 - 8 = 9$ $9 - 8 = 1$ $1 - 8 =$ negative
(1 time) (2 times) (3 times) (1 = remainder)

Quotient = 3 with remainder = 1

Thus we subtract the divisor from the dividend in a work area and test for minus, adding 1 to a counter each time it does not go minus. When we transfer on minus we do not add to the counter, but we add back the divisor to restore the work area to plus. The counter will contain the quotient--the work area will contain the remainder.

The IBM Company has developed special sub-routines to perform division on computers not equipped with the optional Multiply-Divide feature. Consult IBM Manuals D24-1401 or A24-1403.

The work area in which the repeated subtraction occurs should be equal to: the number of digits in the divisor + the number of digits in the dividend + 1.

Thus: (Dividend) (Divisor)
1592 64 = ?

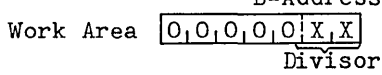
(4 digits) + (2-digits) + 1 = 7 digits

For IBM 1401 computers equipped with the optional Multiply-Divide feature, the following rules apply.

DIVIDE:

Op	Mnemonic	A	B
%	D	X X X X	X X X X

The A-address designates the divisor. The B-address refers to the high-order digit of the dividend. It is stored as the low-order digits of the work area. Length of work area is computed as shown above.



After execution of the division the quotient will be found in the high-order digits of the work area--the remainder will be found in the low-order digits. The number of digits in the quotient will be the same as the number of digits in the dividend. The number of digits in the remainder will equal the number in the divisor.

Work Area [X X X X X X]
Quotient Remainder

The sign of both fields must be designated by zoning. If the field is plus it must have both A and B bits over the low-order digit. If the field is minus it must have a B-bit without an A-bit over the low-order digit.

Both the quotient and the remainder will be signed after division according to the following rules:

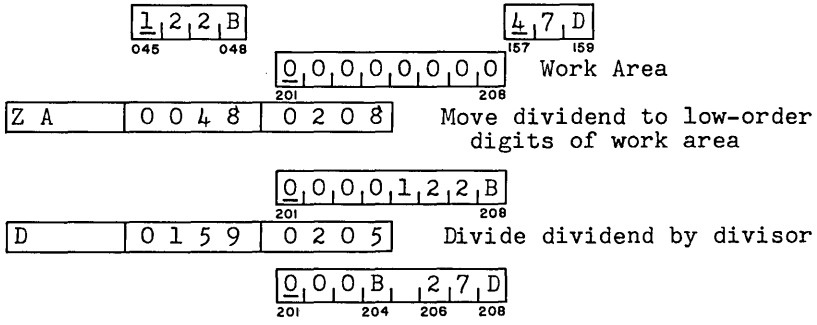
If both fields are plus, both the quotient and remainder will be signed plus.

If both fields are minus, the quotient will be plus but the remainder will be minus.

If the fields have different signs, the quotient will be minus but the remainder will have the sign of the dividend.

EXAMPLES:

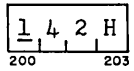
Divide the field located at memory position 0048 by the field located at 0159, using a work area located at 0208.



Note that all fields have a plus sign as shown by presence of the A and B bits over the low-order digits.

PROBLEMS:

Assume division of the following fields:

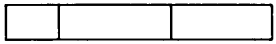
1428 as 

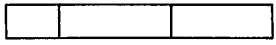
Dividend

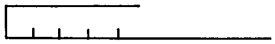
31 as 

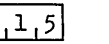
Divisor

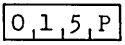
158. How many digits should the work area contain? 158. _____

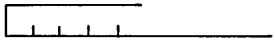
159. If the work area starts at 0400 show the ZA instruction to set up the work area. 159. 

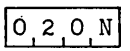
160. Show the divide instruction. 160. 

161. Show the work area after division. 161. 

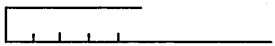
162. A = 

B = 

Indicate the result of B A, showing the sign and size of both quotient and remainder. 162. 

163. A = 

B = 

Indicate the result of A B, showing the sign and size of both quotient and remainder. 163. 

UNIT VII

Lesson 30

NOTES AND ANSWERS

158. 7

4 digit dividend
2 digit divisor
+ 1
7 digit work area

159. Z A | 0 2 0 3 | 0 4 0 6

Place the dividend in the low-order positions of the work area.

160. D | 0 2 0 5 | 0 4 0 3

The B-address must refer to the high-order digit of the dividend.

161. 0 0 4 6 0 2

162. 0 0 1 ! 0 0 P

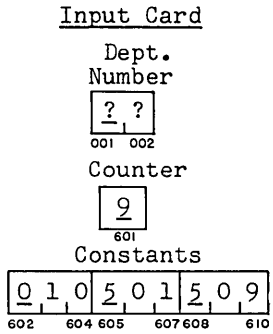
Both the quotient (high-order group) and the remainder (low-order group) would be signed minus (B-bit, no A-bit).

163. 0 0 1 G 0 0 J

The quotient would carry a plus sign (which must be shown as A, B-bits), but the remainder would have the sign of the dividend.

ADDRESS MODIFICATION: It is frequently desirable to modify some part of an instruction to permit re-use of the same instruction. To search a table in storage, for example, it might take many COMPARE instructions to compare each table entry to the item for which you are searching. If, however, you modify the COMPARE instruction address, you can step through the table, using the same COMPARE instruction over and over.

EXAMPLE: Assume you are writing a program that has a 2-digit Dept. Number in each card, ranging from 01-09. This Dept. Number should match a table entry in a table of departments in core storage. When you find the matching number, move the Department Name into the print area and branch back to the main program.



TABLE

Memory Address	Dept. Nr.	Dept. Name
500	01	ACCOUNTING
510	02	FOUNDRY
520	03	MILLING
530	04	PATTERN
540	05	PERSONNEL
550	06	PURCHASE
560	07	SALES
570	08	STORAGE
580	09	TRAFFIC

SUB-ROUTINE:

	Instruc. Addr.	Mnemonic	A	B	d	
*	1000	C	0002	0501		Compare Dept. Nr. to table
	1007	B	1045		S	Branch, if equal
	1012	A	0604	1006		Add 10 to B-addr. of compare
	1019	A	0604	1048		Add 10 to A-addr. of MCW
	1026	S	0603	0601		Subtract 1 from counter
	1033	B	1077	0601	0	Branch if counter equals zero
	1041	B	1000			Go back to COMPARE next entry
*	1045	MCW	0509	0226		Move Dept. Name to print area
	1052	MCW	0607	1006		Initialize COMPARE to 0501
	1059	MCW	0610	1048		Initialize MCW to 0509
	1066	MN	0610	0601		Initialize Counter to 9
	1073	B	XXXX			Branch back to main program
	1077	H	1077			Halt because Dept. Number not in table

* These instructions undergo address modification.

OPTIONAL FEATURE: The IBM 1401 may be equipped with a feature which minimizes the coding required to modify addresses. This feature is called "INDEXING." Three index registers are located at storage addresses as follows:

Index Reg.	#1	at	087-089	Addressed by	089
"	#2	"	092-094	"	094
"	#3	"	097-099	"	099

These registers may be used to store the modifier. When an A or B-address is indexed to a particular register, the contents of that register are added to the address before it is executed. Thus if IR1 contains 010 and an address of 0501 is indexed to IR1, the address will be modified before execution to 0511. Indexing an A or B-address is done by changing the zoning on the middle digit of the 3-digit address.

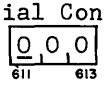
Index Reg.	Address Middle Digit
IR1	A-bit, no B-bit
IR2	B-bit, no A-bit
IR3	A-bit and B-bit

Under the symbolic assembly system the programmer need not worry about zoning. He can indicate the IR being referenced by placing a 1, 2, or 3 in Column 27 to modify the A-address or in Column 38 to modify the B-address.

EXAMPLE: Using IR1 to modify the COMPARE and MCW addresses in the previous example, the program would look like this:

	Instruc. Addr.	Mnemonic	A	Ind	B	Ind	d	
	1000	LCA	0613		0089			Set IR1 to zero
*	1007	C	0002		0501	1		Compare & index B-address
	1014	B	1044				S	Branch if equal
	1019	A	0604		0089			Increase IR1 by 10
	1026	S	0603		0601			Reduce counter by 1
	1033	B	1062		0601		0	Branch if counter = zero
	1041	B	1007					Go back to compare again
*	1044	MCW	0509	1	0226			Move name with A-add. + IR1
	1051	MN	0610		0601			Initialize counter to 9
	1058	B	XXXX					Go back to main program
	1062	H	1062					Halt--no table entry

Special Constant

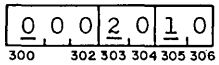


* These instructions are Indexed.

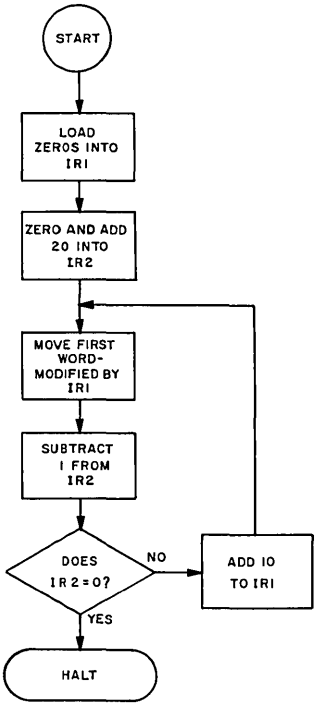
There is no need to initialize the COMPARE and MCW instructions. They were not affected in memory--the indexing occurred while the instructions were being executed.

PROBLEM:

164. Twenty 10-digit fields are located in memory between 0400 and 0599. The first field is at 0409, the second at 0419, etc. We wish to move those twenty fields to 0800 to 0999. We could use 20 MCW instructions, but address modification would require fewer instructions. Both the A-address and the B-address must be modified each time. We must also keep track of the number of times we move to insure moving only the twenty 10-digit fields. Write a program, using IR1 to modify the addresses and IR2 to keep track of the number moved. The following flow chart will assist:



Constants



Label	Mnemonic	A-Addr.	Ind	B-Addr.	Ind	d

UNIT VII

Lesson 31

NOTES AND ANSWERS

164.

Label	Mnemonic	A-Addr.	Ind	B-Addr.	Ind	d	
	LCA	0302		0089			Load zeros in IR1
	ZA	0304		0094			Load 20 in IR2
MOVE	MCW	0409	1	0809	1		Move a word--mod. IR1
	S	0305		0094			Subtract 1 from IR2
	C	0094		0302			Compare IR2 to zero
	B	STOP				S	If equal branch to stop
	A	0306		0089			Otherwise add 10 to IR1
	B	MOVE					Branch to move again
STOP	H	STOP					Stop the program

165. In each of the following multiplication problems, show the zone bits that will appear in the product.

(a) 2 2 2 2 S x 3 3 C

(b) 2 2 2 2 K x 3 3 L

(c) 2 2 2 2 2 x 3 3 L

165.

(a) ☐

(b) ☐

(c) ☐

166. Multiply: X X X X X by X X X and round to two decimal places.

(a) The multiplier should be placed in PROD - ?

(b) A constant of 5 is added to PROD - ?

166.

(a) PROD - ☐

(b) PROD - ☐

167. Divide: X X X X X X by X X X

(a) What is the length of the Quotient area?

(b) Show the location of the Divisor in work area.

(c) Show the location of the Quotient and the Remainder in the work area.

167.

(a) ☐ digits

(b) ⌊ ⌋

(c) ⌊ ⌋

168. In each of the following division problems, show the zone bits that will appear in the quotient and the remainder.

(a) 1 2 3 B ÷ 1 K

(b) 1 2 3 B ÷ 1 B

(c) 1 2 3 K ÷ 1 K

168.

	Quo.	Rem.
(a)	☐	☐
(b)	☐	☐
(c)	☐	☐

169. Name the storage locations of the following:

(a) Index Register 1

(b) Index Register 3

169.

(a) _____ _____

(b) _____ _____

UNIT VII QUIZ

ANSWERS

REMINDERS

165. (a) F

(b) F

(c) Ø

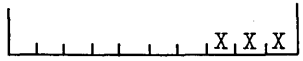
Refer to page 127

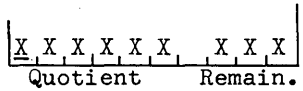
166. (a) PROD - 6

(b) PROD - 1

Refer to page 129

167. (a) 10 digits

(b) 

(c) 

Refer to page 134

168.

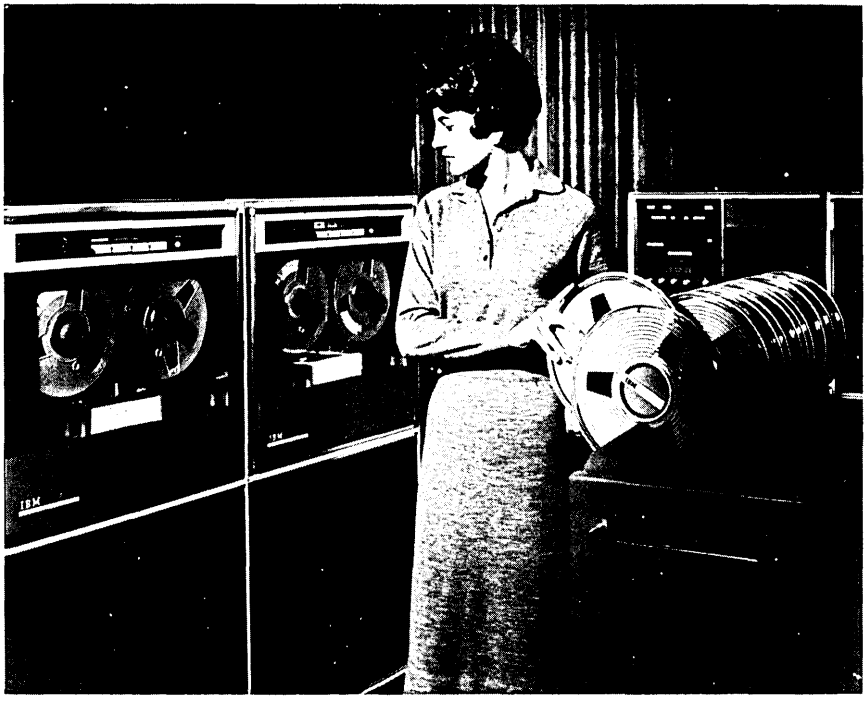
	Quo.	Rem.	
(a)	K	H	One field was plus and one was minus.
(b)	B	H	Both fields were plus.
(c)	B	Q	Both fields were minus.

Refer to page 134

169. (a) 087 - 089

(b) 097 - 099

Refer to page 138

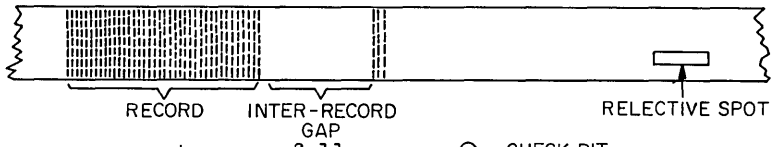


IBM 7330 Magnetic Tape Unit

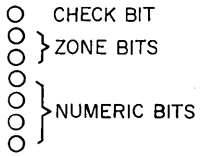
The IBM 7330 magnetic tape unit provides economical, compact record storage for the IBM 1401 data processing system. High speed (out-of-column) and low- speed (in-column) rewinding functions are available with the 7330. Tapes prepared on this unit are completely compatible with the faster IBM 729 II and IV magnetic tape units.

MAGNETIC TAPE: This text has, until now, used punched cards for input and output. Now we must examine a second form of input-output that is frequently used with the IBM 1401.

Magnetic tape is made of thin plastic material in a strip ½ inch wide and 2400 feet long. Data is stored on the tape with the same bit grouping (except W/M's do not go on tape) as in memory. Records are separated from each other by an inter-record gap, 3/4 inch long.



Bits appear on tape as follows:



The check-bit has no bearing on data in tape storage so may be ignored. The bits are counted as "on" if they appear on the tape; "off" if the space is not magnetized.

Three different tape units (machines for reading and writing tapes) may be used with the IBM 1401. They are IBM 729II, IBM 729IV, or the IBM 7330. They differ, primarily, in the speed with which they perform. Since this text will not concern itself with speeds, we will treat all units alike.

Characters may be stored on tape very close together (high density) or further apart (low density). The high density mode has 556 to an inch; low density has 200 to an inch. The IBM 1401 may have up to six tape drives attached to it.

Data is placed on tape as records. A record is a group of one or more characters which normally are needed in the computer at the same time. Up to now we have considered a record as being in one 80-column card. With tape, a record may be as long or as short as necessary.

Each collection of records, of the same type, stored on tape is called a file. A file may be stored on one tape reel (2400 feet) or more than one if it is a very large file.

Every tape reel should have three labels. One label is a sticker attached to the reel showing the name of the file. When we refer to a label in this text we will refer to the second type. A header label should be written on the front of the tape in magnetic characters, just as records are written. This special "record" should provide identifying information about the file contained on the tape reel. Such information may include: file name, date tape written, reel number, etc. The programmer must provide for writing this data on each reel before he begins to write data records.

Each tape reel should also include a trailer label. The trailer label is written at the end of the tape after all data records are on the tape. It should include such information as: number of records on tape, and whether the file is continued on another tape.

The first 20 feet of each tape reel is a "leader" used to thread the tape. Writing begins at the 20 foot point where a piece of reflective aluminum foil is attached to the face of the tape. This foil spot is called a reflective spot or load point. A second reflective spot is located near the end of the tape to signal the approaching end. The tape unit is equipped to "see" the reflective spot and signal the computer.

When all the data is recorded on a tape, it may occupy only a small part of the reel. To avoid reading beyond the end of the data, the programmer must cause a tape mark to be written after the trailer. A tape mark is a one-character record, written as any other record, by the program. As the tape is read, the program continually tests for the tape mark.

A group mark is a special character that must be placed at the end of a record in memory. The group mark causes the tape unit to stop writing and space forward to create an inter-record gap. Its symbol is: ≢ The group mark is read back into memory at the end of the record.

A record mark is a similar symbol (≠) which may be used to separate blocks of information within a record. Its use is optional and it does not affect the tape unit.

When a W/M is to be recorded on tape it is converted to a special character called a word separator, and written immediately ahead of the character with which the W/M was associated in memory.

All characters are checked by the tape unit for correctness during reading or writing. Errors are signalled to the computer but reading or writing continues to the end of the record.

PROBLEMS:

170.	How long is an inter-record gap?	170._____
171.	What character shows the end of a record?	171._____ _____
172.	Who provides for writing a header label?	172._____
173.	How many characters in a tape mark?	173._____
174.	The record count and reel-in-file number at the end of data on a tape is called a _____.	174._____
175.	What keeps tape from running off the end of the reel when writing?	175._____ _____
176.	A collection of records of a similar type on a reel or group of reels is called a _____.	176._____
177.	How many tape units may be attached to the IBM 1401?	177._____
178.	The word separator on tape represents a _____ in memory.	178._____
179.	Show the symbol for a group mark.	179._____
180.	Show the symbol for a record mark.	180._____
181.	Does the program place reflective spots on the tape?	181._____

182. What is the function of the tape mark?

_____.

183. What is the function of a group mark?

_____.

UNIT VIII

Lesson 32

NOTES AND ANSWERS

170. 3/4 inch — This space contains no bits.

171. group mark — This symbol (‡) is recognized by the computer as the end of data to be written.

172. programmer

173. 1 — This character is a special character which causes a signal to be sent to the computer when it is read.

174. trailer — A record count is written when writing. Records may be counted when read and checked against the record count.

175. reflective spot — These are small foil "stickers" placed on the tape at the factory. They may be moved or replaced.

176. file

177. 6

178. word mark

179. ‡

180. ‡

181. no — Reflective spots are applied to the tape by the manufacturer.

182. Signals that the end of the tape is approaching.

183. Separates two records on tape by causing an inter-record gap to be formed.

WRITING ON MAGNETIC TAPE: Data is written on tape, character-by-character, from the address given by the program until a group mark is sensed. The tape unit reads each character as it is written and compares what is read against what was sent from the computer to be written. Errors are signalled by the tape unit. This signal enables the program to backspace (reverse the tape movement back to the last IR gap) and re-write the record. The following instructions are used to write:

WRITE TAPE:

Mnemonic	A-Address	B-Address	d
M C W	% U n	X X X X	W

This is the MOVE CHARACTER TO WORD MARK instruction which we have discussed. The A-address indicates which tape unit is used. The %U indicates a tape operation; the n indicates the number (1-6) of the particular tape unit. The B-address indicates the address of the left-most digit of the record to be written. The d-modifier (W) indicates a WRITE operation. Word marks are ignored.

WRITE TAPE WITH WORD MARKS:

Mnemonic	A-Address	B-Address	d
L C A	% U n	X X X X	W

This is the LOAD CHARACTER TO A WORD MARK previously discussed. This instruction operates like WRITE TAPE except that when a W/M is encountered it is converted to a word-separator character and placed ahead of its associated character.

WRITE TAPE MARK:

Op	Mnemonic	A-Address	d
U	C U	% U n	M

This instruction causes the tape unit to record a tape mark on the tape. Inter-record gaps precede and follow the tape mark.

SKIP AND BLANK TAPE:

Op	Mnemonic	A-Address	d
U	C U	% U n	E

When repeated errors are signalled during writing and re-writing does not correct the trouble, the cause may be a damaged spot on the tape. This instruction may be used to skip about 3½ inches, erasing any data that might be in that space. The data is then written on the new area of tape.

BRANCH IF END OF REEL:

Mnemonic	A-Address	d
B	X X X X	K

If the reflective spot is sensed while writing, or a tape mark while reading, an indicator (EOR) is turned on. After each record is written this instruction should be coded to permit a branch to an "END OF REEL ROUTINE" if the indicator is on. The EOR routine should provide notification to the operator to load a new reel.

BRANCH IF TAPE ERROR:

Mnemonic	A-Address	d
B	X X X X	L

If an error is detected while reading or writing, an indicator is turned on. This instruction should be coded after each read or write to test the error indicator. If it is "on" the program will branch to the A-address where an "error routine" should be coded. The error routine should cause a backspace and a re-read or re-write.

EXAMPLES:

Memory:

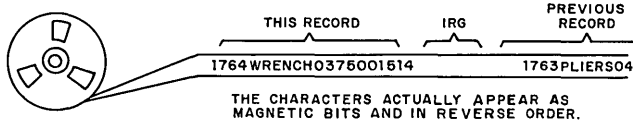
Stock Nr.	Item Name	Price	Quan.	Bin Nr.	Grp Mk
1764	WRENCH	03750015	14		‡

0500 0520

1. Write tape

M C W	% U 1	0 5 0 0	W

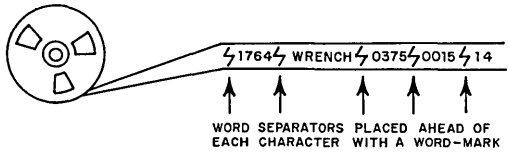
(Write a record on tape unit #1, starting at 0500)



2. Same record

Write tape with W/M's

L C A	% U 1	0 5 0 0	W



PROBLEMS:

184. Show an instruction to write on Tape Unit #4, without W/M's, from 300.

184.

MNE— MONIC	A	B	d

185. Show an instruction to write on Tape Unit #6, with W/M's, from 300.

185.

186. Show an instruction to branch to a routine labelled ERROR if the error indicator is on.

186.

187. Show an instruction to branch to a routine labelled EOR if the end-of-reel indicator is on.

187.

188. Show an instruction to write a tape mark on Tape Unit #5.

188.

189. Show an instruction to SKIP AND BLANK $3\frac{1}{2}$ inches of tape on Unit #2.

189.

UNIT VIII

Lesson 33

NOTES AND ANSWERS

184.

Mnemonic	A-Address	B-Address	d
M C W	% U 4	0 3 0 0	W

This is the most common method of writing on tape.

185.

L C A	% U 6	0 3 0 0	W

This method is used primarily to store programs on tape.

186.

B	E R R O R		L

187.

B	E O R		K

The d-modifier tells which indicator to test. If the indicator is not "on" no branch will occur.

188.

C U	% U 5		M

Write a tape mark and the skip and blank tape are called "control" instructions.

189.

C U	% U 2		E

READING FROM MAGNETIC TAPE: Data is read from tape character-by-character, placing each character in memory starting at the address given in the instruction. If word-separators are encountered, a W/M is placed with the next character read. Reading stops when an inter-record gap (IRG) is encountered. All characters are actually read twice and compared. Errors are signalled but the reading does not stop.

READ TAPE:

Mnemonic	A-Address	B-Address	d
M C W	% U n	X X X X	R

The A-address specifies the tape unit from which data is to be read. The B-address indicates the memory position of the left-most character. The d-modifier is an R, which indicates a read operation. Data must be on tape without word-separators, i. e., it must have been written with a MCW instruction.

READ TAPE WITH WORD MARKS:

Mnemonic	A-Address	B-Address	d
L C A	% U n	X X X X	R

This instruction operates like READ TAPE except that word-separators are converted to W/M's.

BACKSPACE TAPE RECORD:

Mnemonic	A-Address	d
C U	% U n	B

Op Code
U

This control instruction causes the tape unit designated by the A-address to backspace to the last IRG. By this means, the same section of tape can be re-read or re-written.

REWIND TAPE:

Mnemonic	A-Address	d
C U	% U n	R

Op Code
U

This control instruction causes the tape unit to rewind the tape on its original reel. Tapes must always be rewound before taking the reels off the tape unit. The tape stops at the LOAD POINT, properly positioned to read or write.

REWIND TAPE AND UNLOAD:

Mnemonic	A-Address	d	Op Code
C U	% U n	U	U

This instruction is like REWIND TAPE, except that the tape is ready for removal from the tape unit. The tape unit cannot be addressed by the computer again until the operator resets it by positioning the tape.

EXAMPLES:

All read and write instructions should be followed by BRANCH ON EOR and BRANCH ON ERROR instructions to detect these conditions.

Label	Mnemonic	A-Address	B-Address	d	
R E A D	M C W	% U 2	0 5 0 0	R	Read a tape
	B	E R R O R		L	Branch, if error indicator "on"
	B	E O R		K	Branch, if EOR indicator "on"
	B	X X X X			Otherwise branch to main program

Label	Mnemonic	A-Address	B-Address	d	
W R I T E	M C W	% U 1	0 5 0 0	W	Write a tape
	B	E R R O R		L	Branch, if error indicator "on"
	B	E O R		K	Branch, if EOR indicator "on"
	B	X X X X			Otherwise branch to main program

PROBLEMS:

190. Write an instruction to read from Tape Unit #2 to position 0300, without word marks.

190.

MNE MONIC	A	B	d

191. Write an instruction to read from Tape Unit #3 to position 0300, with word marks.

191.

192. Write a control instruction to backspace Tape Unit #6.

192.

193. Write a control instruction to rewind Tape Unit #4.

193.

194. Write a control instruction to rewind and unload Tape Unit #5.

194.

UNIT VIII

Lesson 34

NOTES AND ANSWERS

190.

Mnemonic	A-Address	B-Address	d
M C W	% U 2	0 3 0 0	R

The MCW instruction is normally used for reading.

191.

L C A	% U 3	0 3 0 0	R

The LCA is used when the data on tape was written with an LCA.

192.

C U	% U 6		B

All control instructions are directed to a specific tape unit and differ only in the d-modifier used.

193.

C U	% U 4		R

194.

C U	% U 5		U

SPECIAL FEATURES OF TAPE PROCESSING: Although an IBM 1401 can have as many as six tape units, it may have fewer. Whatever the number, it is possible to waste processing time by using them improperly. If, for example, the whole process must stop while tapes are being loaded on the tape unit, valuable time is lost. If sufficient tape units are available, the "flip-flop" technique may be used.

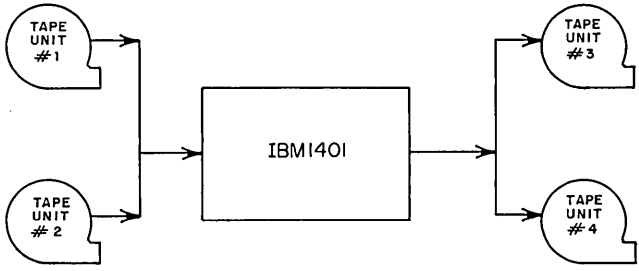
TAPE FLIP-FLOP: If you have a large, multi-reel file to read--and to write, and you have four tape units available, you can assign two to the input file and two to the output file. Assume a 4-reel file:

Tape Unit #1 - Read 1st reel and 3rd reel

Tape Unit #2 - Read 2nd reel and 4th reel

Tape Unit #3 - Write 1st reel and 3rd reel

Tape Unit #4 - Write 2nd reel and 4th reel



When Reel #1 is finished on Tape Unit #1, modify the A-address of the read instruction to %U2 on which Reel #2 is loaded. Processing can continue while the operator takes Reel #1 off Tape Unit #1 and mounts Reel #3. When Reel #2 is exhausted, the A-address is changed back to %U1. The same can be done with the output tapes on Tape Units #3 and #4.

EXAMPLE:

One way to code a flip-flop is as follows:

Set up a special two-digit register in storage.
Start with 02 in the register.

Register

Label	Mnemonic	A-Address	B-Address	d	
READ	MCW	%U1	INPUT	R	Read a record
	B	ERROR		L	Branch if error indicator "on"
	B	EOR		K	Branch if EOR indic. "on"
	B	PROG			Branch bk. to main program
ERROR	CU	%U1		B	Backspace tape one record
	MCW	%U1	INPUT		Read again
	B	HALT		L	Branch on 2nd error to halt
	B	PROG			Branch bk. to main program
HALT	H	HALT			Stop f/operator to correct error
EOR	CU	%U1		U	Rewind and unload
	MN	READ + 3	0181		Move (n) portion of A-add. to 181
	MN	0182	READ + 3		Move the number in the
	MN	0182	ERROR + 3		register to each (n)
	MN	0182	ERROR + 8		portion of the read and
	MN	0182	EOR + 3		control instructions
	MN	0181	0182		Move the "saved" (n) into the register
	B	READ			Go back to read new tape

Now the register looks like this:

The next time through the EOR routine the %U2 will be set back to %U1.

Actually, a routine like this would require more coding to work properly, but this illustrates one way to code a FLIP-FLOP. It is customary to re-read more than once for errors and we would need a counter to count records, a routine to read the label on a new tape, and a counter to tell us when all reels have been read.

PROBLEMS:

195. How many digits in a read instruction must be modified for flip-flop?

195. _____

196. How many tape units are required to flip-flop the read operation?

196. _____

197. Which branch instructions must always follow a read or write?

197.
(a) _____

(b) _____

198. Write a control instruction to skip and erase $3\frac{1}{2}$ inches of tape on Tape Unit 1.

198.

UNIT VIII

Lesson 35

NOTES AND ANSWERS

195. 1 — Only the (n) part of the A-address needs to be modified.

196. 2 — Two units are alternately used.

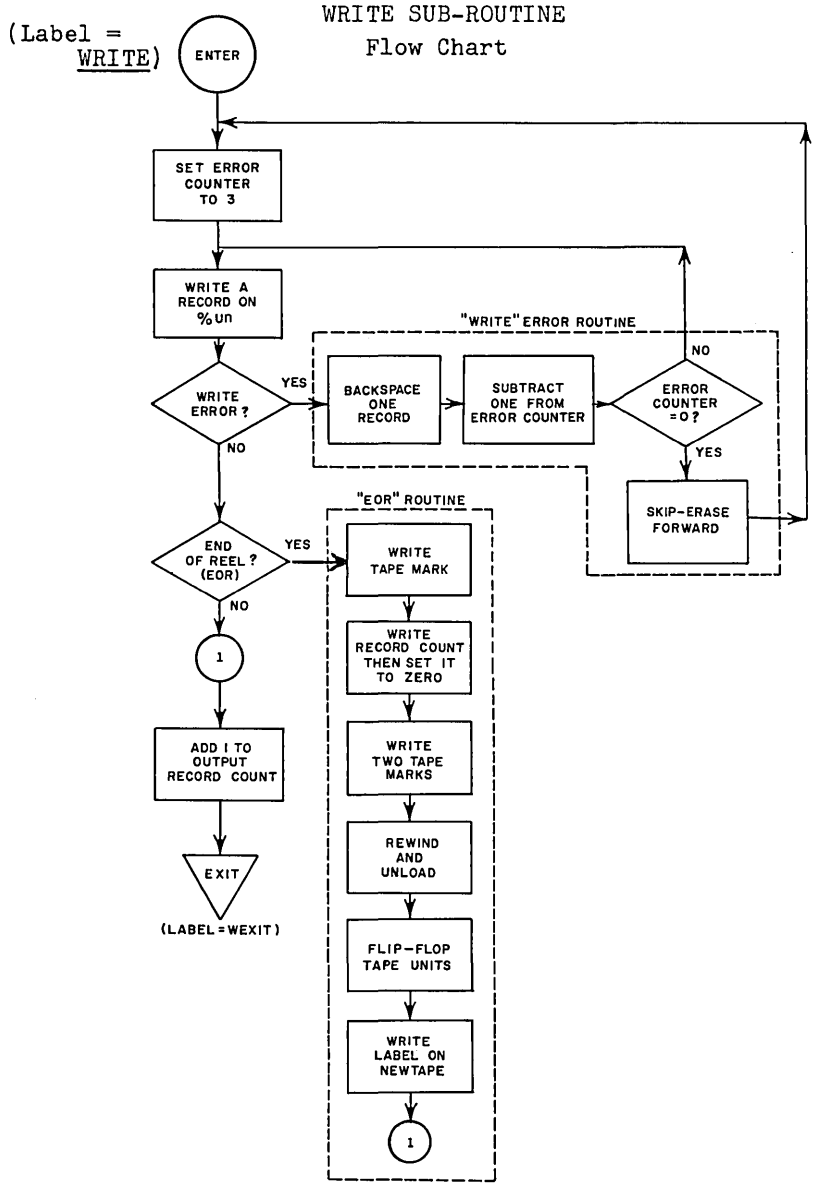
197. (a) Branch if tape error — These two branch instructions must be coded following every read and write instruction.

(b) Branch if end of reel

198.

MNE-MONIC	A	d
C U	% U 1	E

READ AND WRITE SUB-ROUTINES: It is customary to program tape read and write process as sub-routines. This may be done several ways, but the following will serve:

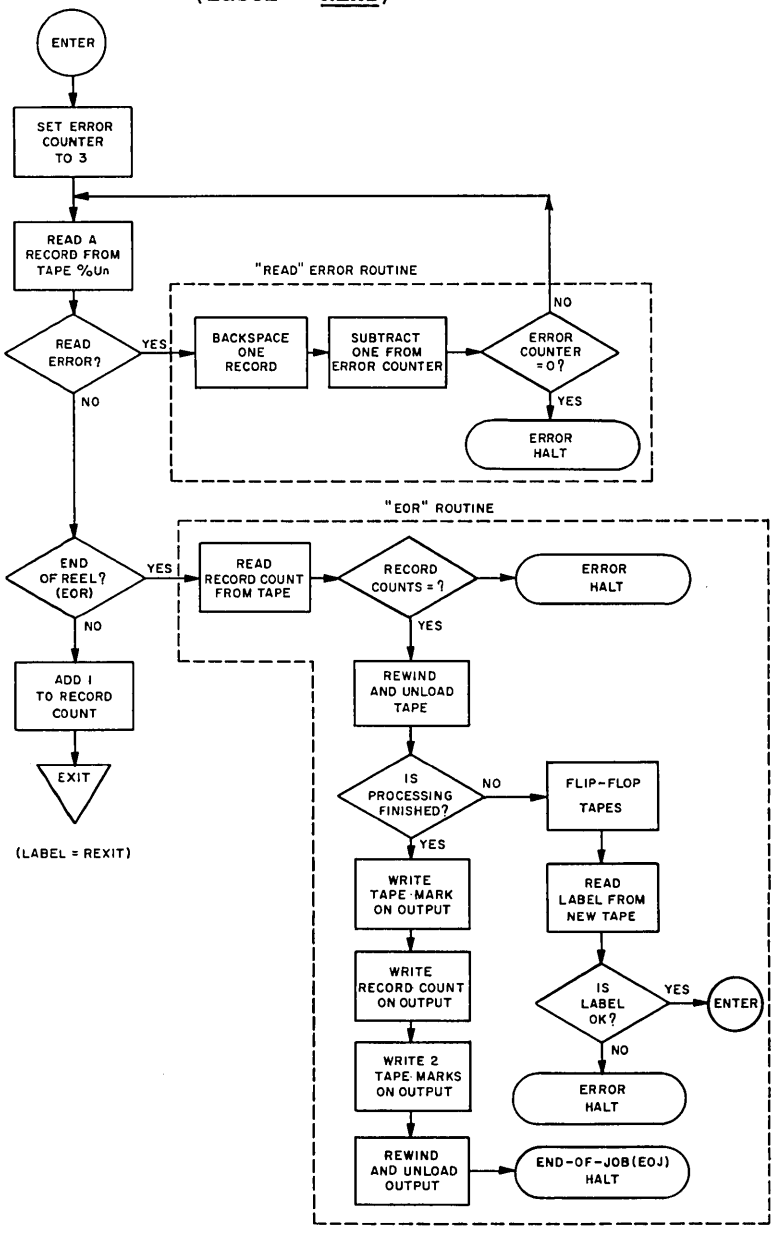


WRITE SUB-ROUTINE
CODING

COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	(A) OPERAND ±	(A) OPERAND CHAR. ADJ.	(A) OPERAND IND.	(B) OPERAND ADDRESS	(B) OPERAND ±	(B) OPERAND CHAR. ADJ.	(B) OPERAND IND.	d	COMMENTS
	WRITE	ZA	CON3				WCTR					SET COUNTER TO 3
		MCW	%U3				0400				W	WRITE A RECORD
		B	WERR								L	BRANCH IF ERROR
		B	WEOR								K	BRANCH IF EOR
		A	CON1				WRCDCT	-	001			ADD 1 TO REC CT
	WEXIT	B	XXXX									BRANCH BK TO PGM
	WERR	CU	%U3								B	BACKSPC A RECORD
		S	CON1				WCTR					REDUCE COUNTER
		C	WCTR				CONZER					COUNTER = 0?
		B	WSKIP								S	TO SKP ERASE IF=
		B	WRITE	+	007							TO REPEAT WRITE
	WEOR	CU	%U3								M	WRITE TAPE MARK
		MCW	%U3				WRCDCT	-	006		W	WRITE RECORD CT
		ZA	CONZER				WRCDCT	-	001			ZERO THE REC CT
		CU	%U3								M	WRITE TAPE MARK
		GU	%U3								M	WRITE TAPE MARK
		CU	%U3								U	REWIND & UNLOAD
		MN	WRITE	+	010		WREG	-	001			CHANGE ALL TAPE
		MN	WREG				WRITE	+	010			REFERENCES TO
		MN	WREG				WERR	+	003			ALTERNATE TAPE
		MN	WREG				WEOR	+	003			AS SHOWN IN THE
		MN	WREG				WEOR	+	008			REGISTER (WREG)
		MN	WREG				WEOR	+	023			
		MN	WREG				WEOR	+	028			
		MN	WREG				WEOR	+	033			
		MN	WREG				WLABEL	+	003			
		MN	WREG				WSKIP	+	003			
		MN	WREG	-	001		WREG					RESET REGISTER
	WLABEL	MCW	&U3				LABEL					WR LAB ON NEW TP
		B	WEXIT	-	007							BR TO RECORD CT
	WSKIP	CU	%U3								E	
		B	WRITE									
1	CON3	DCW	*		3							CONSTANT OF 3
1	WCTR	DCW	*		0							ERROR COUNTER
1	CON1	DCW	*		1							CONSTANT OF 1
7	WRCDCT	DCW	*		000		000					RECORD COUNT
1	CONZER	DCW	*		0							CONSTANT OF 0
2	WREG	DC	*		04							REGISTER OF TAPE

READ SUB-ROUTINE
Flow Chart

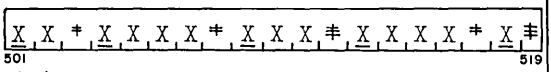
(Label = READ)



READ SUB-ROUTINE
CODING

COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
	READ	ZA	CON3				RCTR					SET COUNTER TO 3
		MCW	%U1				0400					RREAD A RECORD
		B	RERR									LBRANCH IF ERROR
		B	REOR									KBRANCH IF EOR
		A	CON1				RRCDCT	-	001			ADD 1 TO REC CT
	REXIT	B	XXXX									BRANCH BK TO PGM
	RERR	CU	%U1									BBACKSPC A RECORD
		S	CON1				RCTR					REDUCE COUNTER
		C	RCTR				CONZER					COUNTER = 0?
		B	EHLTI									SBR TO HALT IF =
		B	READ	+	007							GO TO REPEAT RD
	REOR	MCW	%U1				0400					RREAD RECORD CT
		C	0405				RRCDCT	-	001			RECORD COUNTS =
		B	RCOK									SIF = GO AHEAD
	EHLTZ	H										IF NOT HALT
	RCOK	CU	%U1								U	
		B	EOJ				INDIC					1BRANCH IF EOJ
		MN	READ	+	010		RREG	-	001			IF NOT EOJ FLPTP
		MN	RREG				READ	+	010			
		MN	RREG				RERR	+	003			
		MN	RREG				REOR	+	003			
		MN	RREG				RCOK	+	003			
		MN	RREG				RLBL	+	003			
	RLBL	MCW	%U1				0400				R	
		C	0406				CONLAB	-	013			COMP WITH LBLCON
		B	READ									SIF = GO AHEAD
	EHLT3	H										IF NOT HALT
	EHLT1	H										
	EOJ	MN	WRITE	+	010		WROUT	+	003			SET OUTPUT TAPE
		MN	WRITE	+	010		WROUT	+	008			ACCORDING TO
		MN	WRITE	+	010		WROUT	+	016			WHAT IS IN WRITE
		MN	WRITE	+	010		WROUT	+	021			SUB-ROUTINE
		MN	WRITE	+	010		WROUT	+	026			
	WROUT	CU	%U3									MWR OTPT TAPE MRK
		MCW	%U3				WRCDCT	-	006			WWRITE RECORD CT
		CU	%U3									MWR TWO TAPE MRKS
		CU	%U3								M	
		CU	%U3									UREWIND & UNLOAD
		H										END OF JOB
1	RCTR	DCW	*		0							
7	RRCDCT	DCW	*		000		000					
1	INDIC	DCW	*		0							
2	RREG	DC	*		02							
20	CONLAB	DCW	*		INVbCTLblbOFblbYDDD							

199.



(a) How many tape records will this make?

(b) Which storage positions will cause IRG?

200. With core storage as above and the following instruction:

Op	A	B	d
L C A	% U 1	0 5 0 1	W

(a) How many word-separators will be placed on tape?

(b) List the storage positions that will cause this to happen.

201. Write the two instructions that should immediately follow a Read instruction.

202. Show an instruction to write on Tape Unit 6 (without W/M's) from storage location 1260.

203. Tape Unit 1 (mentioned in Problem 200) is now to be read into storage into loc. 501. Write the instruction that will do this.

199.

(a) ☐

(b) _____

200.

(a) ☐

(b) _____

_____

201.

Op	A	B	d

202.

Op	A	B	d

203.

Op	A	B	d

UNIT VIII QUIZ

ANSWERS

REMINDERS

199.(a) [2]

(b) 512, 519

The group mark (‡) separates records and causes IRG on tape.

200.(a) [5]

(b) 501, 504, 509, 513, 518

The W/M causes word-separators on tape if the Load instruction is given.

201.

Op	A	B	d
B	R E R R		L
B	R E O R		K

1. Branch if Read Error
2. Branch if End-of-Reel

202.

Op	A	B	d
M C W	% U 6	1 2 6 0	W

The Move instruction writes without placing word-separators on tape.

203.

Op	A	B	d
L C A	% U 1	0 5 0 1	R

The tape contains word-separators, therefore the Load instruction is used to convert them to W/M's in storage.

UNIT IX

Lesson 37

SPECIAL FEATURES: The IBM 1401 has many special features and additional equipment available on an optional basis. Some of these have been discussed, others mentioned, and a few have not been referred to. Also, there are many factors which have bearing on the operation of the computer, but are not considered essential to an elementary text. All features are fully covered in IBM Manual A24-1403.

IBM RAMAC 1401 SYSTEM: This is an optional form of storage which permits records to be filed in a readily accessible magnetic disc file. Like memory storage, sections of this file can be addressed directly, without reading through many records as on tape. This device has a special set of instructions.

MODIFY ADDRESS: This is an optional instruction available on computers with memory storage larger than 4000 characters. In order to express memory addresses higher than 3999 in 3 digits, the computer uses zoning over the units position of the 3-digit address. A special arithmetic is needed to modify such addresses. The (MA) instruction provides a simple means of modifying such addresses.

READ AND PUNCH RELEASE: This extra cost option saves time when reading or punching large numbers of cards. It permits card feeding to occur more rapidly than on the normal equipment.

PUNCH FEED READ: This optional attachment permits punching data in the same card from which data was read.

PRINT STORAGE FEATURE: This option permits less delay in processing while printing. The write instruction transfers data to a temporary storage area instead of directly to the printer. Then the program continues while the printer operates to print out of the temporary storage.

EXPANDED PRINT EDIT: This option adds several additional print features to the basic MOVE CHARACTERS AND EDIT instruction. For example, asterisks, CR and blank-spacing can be placed to the left of the printed data as well as to the right. The dollar sign can be made to "float" so that it is always printed next to the high-order digit.

STORE ADDRESS REGISTER: When this option is available, the programmer can "save" the A and B addresses, at any given point in the program, and refer to them later.

MOVE RECORD: This is an optional instruction that permits moving entire records, without regard to W/M's, from one place in memory to another.

SENSE-SWITCHES: A group of six toggle switches on the console may be turned on or off by the operator. Any one switch can be tested by a BRANCH IF INDICATOR ON instruction. These switches permit the introduction of alternate processes in the program. The operator can select the proper process without changing the program. The sense switches are labelled B through G.

TIMING: Although timing is important in the writing of efficient programs, this text will not cover this detail. Following the rules for coding discussed in this text should result in average efficiency. As the programmer gains experience, further improvement may be realized.

PROGRAMMING AIDS: The IBM company maintains a large force of trained people to write general routines and to assist customers in their use. These aids can save many programming hours and reduce errors.

OPERATING STEPS: A computer is a very precise device. Every character in the program, and there may be thousands, must be exactly correct. Using a symbolic assembly program is one way to minimize errors. We have seen that this system automatically places the proper zoning for 3-digit memory addresses and always places the correct code for the operation.

Practice and experience will aid the programmer to use correct logic. The flow chart is an invaluable aid to insure against logic errors. It is not uncommon for a programmer to set up a COMPARE followed by a BRANCH ON EQUAL--but fail to take care of situations that are not equal. A carefully prepared flow chart will show such errors.

The people who key-punch program cards are careful; but they are people, and people make mistakes. Cards should be checked by the programmer before entering them in the computer for assembly. Many keypunch errors are caused by the programmer when he fails to form letters and numerals clearly. Program sheets should be written in block letters. Some letters and numerals are similar enough to be easily confused. The following alphabet will reduce similarities:

ABCDEFGHIJKLMNØPQRSTUVWXYZ

NUMERALS: 1234567890

Note that letters I, O, S, and Z, which closely resemble numerals 1, 0, 5, and 2, have distinctive form, which must be followed to avoid errors. Similarly, the letters U and V, which may be formed to resemble each other, are also distinctive. Close adherence to these forms will greatly reduce keypunch errors.

The programmer should always write precise directions for the operators who will operate the computer. An inadvertent error, failure to enter the right input cards, mounting the wrong tape, or failure to set the right sense switches might result in an expensive rerun.

Testing, or "debugging," is a necessary step. After a program is assembled the programmer must test the program with sample data precisely like the data the program was designed to process. If minor errors are discovered they may be corrected by changing the machine-language program deck. If these are numerous, or if major errors occur, it may be best to correct the symbolic deck and re-assemble the whole program. In any event, the program should not be used to process "live" data until all errors are corrected.

The programmer will want to make use of several "generalized" programs provided by the IBM Company to its customers. One such program is the "LOAD PROGRAM" used to replace the instructions from cards into their correct memory positions. This program is punched in two or three cards and is placed before the regular program deck. The LOAD PROGRAM may also include a CLEAR MEMORY program which sets all memory positions to blank before loading the program.

Another standard IBM program is the MEMORY PRINT or MEMORY DUMP program. This program provides a printed output showing the contents of every position of memory. When a program stops, due to an error in the program, a MEMORY PRINT will show the exact "picture" of the contents of the memory positions at the time of stoppage. From this the programmer can see whether the core positions appear as they should. With the MEMORY PRINT, the operator should supply a CONSOLE PICTURE, which is a schematic drawing showing all of the lights and switches. The operator must indicate which lights were on, and how the switches were positioned at the time of stoppage. These facts will tell the programmer the type of error and the memory address of the instruction being performed, etc.

The program itself should be organized into three main parts:

HOUSEKEEPING: This part of the program is usually performed only once, each time the program is run. It usually is the first part and provides all of the preparatory

steps that must be taken before processing can begin. It should clear storage in the input, output and work areas. Word marks should be placed wherever needed. The housekeeping routine should read and check labels from input tapes and write labels on output tapes. Frequently, programmers provide for reading the first record from the input tapes and the first card from the card read-punch, in this routine. This enables them to code a READ AND PUNCH or READ AND PRINT instruction near the end of the main program.

MAIN PROGRAM: This is the part of the program where most of the processing actually occurs. This usually takes the form of a large "loop" which is repeated each time a new record is processed. The loop itself may have many branches, but sooner or later the process must return to the beginning of the main program to repeat the cycle.

INPUT-OUTPUT: The reading and writing of data on the input-output devices is really part of the main program. Because, however, such processes have special problems involving error conditions and end-of-reel or end-of-file situations, they are frequently treated separately. Having decided what he will read or write, and where, the programmer should give careful consideration to the coding required to perform these steps. The internal speed of the computer is much greater than the input-output speeds. Therefore he should choose an input-output method that will offer the greatest saving in time. Use of the tape read and write sub-routines will also help to insure proper checking of error and EOR indicators.

The use of sub-routines is strongly recommended, wherever possible. These are like small programs, linked to the main

program by a branch instruction or by a calling sequence. A calling sequence is a set of instructions which may be coded anywhere in the main program, and which provide for modifying the final branch from the sub-routine to always return to the right place.

EXAMPLE:

One method of branching to a sub-routine might be as follows:

Label	Op	A	B
	MCW	ADCON+3	EXIT
	B	SUBRT	
ADCON	DCW	ADCON+4	
	XXX	XXXXX	Continue program after sub-routine
	XXX	XXXXX	
SUBRT	XXX	XXXXX	Sub-routine
	XXX	XXXXX	
EXIT	B	0000	

The MOVE instruction moves the address of the next instruction in sequence to the final branch of the sub-routine.

QUICK REFERENCE
of All Instructions Covered in the Text

Refer to Pg.	Op Code	Mnemonic	A	B	d	Descriptive Title
I. W/M, MOVE and LOAD OPERATIONS:						
25	/	CS	XXX			Clear Storage
25	/	CS	XXX	XXX		Clear Storage and Branch
17	,	SW	XXX	XXX		Set W/M (one or both addresses)
25	⌘	CW	XXX	XXX		Clear W/M (one or both addresses)
17	M	MCW	XXX	XXX		Move
41	L	LCA	XXX	XXX		Load
41	Z	MCS	XXX	XXX		Move & Suppress Zeros
41	D	MN	XXX	XXX		Move Numeric (only)
41	Y	MZ	XXX	XXX		Move Zone (only)
II. ARITHMETIC OPERATIONS:						
58	A	A	XXX	XXX		Add (one or both addresses)
63	S	S	XXX	XXX		Subtract (one or both add.)
127	@	M	XXX	XXX		Multiply
131	%	D	XXX	XXX		Divide
58	?	ZA	XXX	XXX		Zero and Add
63	!	ZS	XXX	XXX		Zero & Subtract (one or both addresses)
III. LOGICAL OPERATIONS:						
45	C	C	XXX	XXX		Compare
46	B	B	XXX			Unconditional Branch
46	B	B	XXX	XXX	X	Branch if d-Char. Equal
46	B	B	XXX		X	Test d-Char. and Branch
47	V	BWZ	XXX	XXX	X	Test for Zone or W/M and Branch

Refer to Pg.	Op Code	Mne-monic	A	B	d	Descriptive Title
IV. MISCELLANEOUS OPERATIONS:						
67	N	NOP				No Operation
67	•	H				Halt
67	•	H	XXX			Halt and Branch
67	K	SS			X	Stacker Select
119	F	CC			X	Control Carriage
119	F	CC	XXX		X	Control Carriage & Branch
115	E	MCE	XXX	XXX		Move Characters and Edit
93		DCW	XXX			Define Constant with W/M
93		DC	XXX			Define Constant
97		DS	XXX			Define Symbol
97		ORG				Origin
97		CTL				Control
98		END				End
V. INPUT/OUTPUT OPERATIONS:						
21	1	R				Read a Card
21	1	R	XXX			Read and Branch
21	2	W				Write a Line
21	2	W	XXX			Write and Branch
21	4	P				Punch a Card
21	4	P	XXX			Punch and Branch
111	3	WR	(XXX)			Write & Read (Br. if A-add)
111	5	RP	(XXX)			Read & Punch (Br. if A-add)
111	6	WP	(XXX)			Write & Punch (Branch if A-address)
112	7	WRP	(XXX)			Write, Read, Punch (Branch if A-address)

Refer to Pg.	Op Code	Mne-monic	A	B	d	Descriptive Title
VI. MAGNETIC TAPE OPERATIONS:						
149	M	MCW	%UX	XXX	W	Write Tape (without W/M's)
149	L	LCA	%UX	XXX	W	Write Tape (with W/M's)
153	M	MCW	%UX	XXX	R	Read Tape (without W/M's)
153	L	LCA	%UX	XXX	R	Read Tape (with W/M's)
150	B	B	XXX		L	Branch if Tape Error
150	B	B	XXX		K	Branch if End-of-Reel
149	U	CU	%UX		M	Write Tape Mark
149	U	CU	%UX		E	Skip and Blank Tape
153	U	CU	%UX		B	Backspace Record
153	U	CU	%UX		R	Rewind Tape
154	U	CU	%UX		U	Rewind and Unload Tape

UNIT IX QUIZ

The quiz on the following pages is somewhat different from the previous ones in that this quiz will attempt to cover the highlights of the entire course.

There will be 25 problems and the correct answers will not be given until all problems have been completed. Work the problems without referring to the text, and when you have finished, score yourself by allowing four points for each problem answered correctly.

The quiz should take between one and two hours to complete and you should score at least 70. The correct answers are referenced to pages in the text. Check out all problems that were done incorrectly.

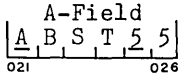
In problems 204 through 213 show the result of combining the A-field with the B-field, using the specified operation code.

	Mnemonic	Contents of A-Field	Contents of B-Field	Resultant B-Field
204.	A	1000	100B	
205.	A	100S	200M	
206.	A	100	11666	
207.	S	1000B	1000K	
208.	S	1111A	1111C	
209.	S	1111C	1111A	
210.	ZA	222	33333	
				Resultant A-Field
211.	S	0111M	No B-Field	
212.	ZA	0111D	No B-Field	
213.	ZS	0111M	No B-Field	

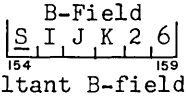
214. After executing the following two steps,

Mnemonic	A	B	d
M N	0 0 2 6	0 1 5 6	
M Z	0 0 2 3	0 1 5 8	

A-Field



B-Field

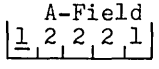


show the contents of the resultant B-field:

215. After executing the following two steps,

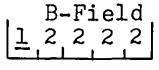
Mnemonic	A	B	d
C	H O L D	A R E A	
B	0 7 5 0		/

A-Field



Hold

B-Field



Area

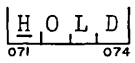
where will the program go for its next instruction?

Answer: _____.

216. After executing the following step,

Mnemonic	A	B	d
B	S U B R	0 0 7 2	H

B-Field



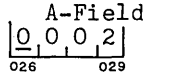
where will the program go for its next instruction?

Answer: _____.

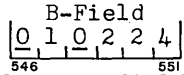
217. After executing the following instruction,

Mnemonic	A	B	d
M C S	0 0 2 9	0 5 5 1	

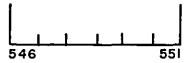
A-Field



B-Field



show the contents of the resultant B-field:



218.

Instr. Addr.	Op	A	B	d
9 0 1	N	0 9 9 1		
9 0 5	M	0 9 8 7	0 9 0 1	
	-			
	-			
9 8 7	B	0 2 2 1		
9 9 1	.			

After execution of the above program, the instruction in 901 will be:

9 0 1		

219. Write the instructions to accomplish the following actions:

Add	to
521-524	807-811
526-529	812-815
530-533	816-819
535-538	820-823

Op	A	B	d

220. Set up a constant, labelled CONST, in the following format: T O T b C O S T b b $ b , b b b . b b

Count	Label	Op	A 24

221.

Label	Op	A	B	d
	C S	0 0 8 0		
	S W	0 3 1 0	0 3 2 5	
R E A D	R			
	B W Z	STKIN I	0 0 0 4	K
	C	CONS1	0 0 0 3	
	B	S T O P		/

(a) READ + 8 refers to: ☐

(b) READ - 7 refers to: ☐

(c) READ + 16 refers to: ☐

(d) READ - 11 refers to: ☐

222. Using the constant set up in Problem 220 and data field as follows: 1 2 7 6 0 4
420 425

(a) Write the instructions to edit the data and to print starting in print position 39.

Op	A	B	d

(b) Show the printed line that would result.

Print Area

220 230 240

223. Write a program, in symbolic, to solve the following problem:

Multiply QTY (XXX.X) by COST (X.XX). Half adjust the product to the nearest cent (PROD = XXXX.XX).

Count	Label	Op	A	B	d	
						Set up constants
						Program to multiply

224. Fill in the missing instructions that will result in a triple-spaced report with the least amount of program interlock.

	Op	A	B	d
	R			
	L C A	H O L D	A R E	
?				
?				

225. Tape has been written with the following instruction:

Op	A	B	d
L C A	% U 6	1 2 0 0	W

Show the instruction to be used to read this tape into storage.

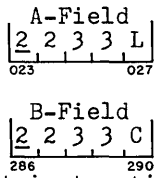
Op	A	B	d

226. Write the two instructions that should immediately follow a WRITE instruction.

Op	A	B	d

227. On a 1401 equipped with the High-Low-Equal compare feature, after executing the following two steps,

Op	A	B	d
C	0 0 2 7	0 2 9 0	
B	S T O P		U

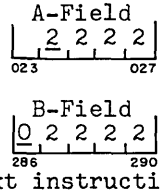


where will the program go for its next instruction?

Answer: _____.

228. On a 1401 equipped with the High-Low-Equal compare feature, after execution of the following two steps,

Op	A	B	d
C	0 0 2 7	0 2 9 0	
B	S T O P		T



where will the program go for its next instruction?

Answer: _____.

ANSWERS TO QUIZ PROBLEMS

204. 200B (Pg. 58)

205. 100K (Pg. 58)

206. 11766 (Pg. 58)

207. 2000M (Pg. 63)

208. 0000B (Pg. 63)

209. 0000K (Pg. 63)

210. 0022B (Pg. 58)

211. 0000Ō (Pg. 63)

212. 0111D (Pg. 58)

213. 0111D (Pg. 63)

214. SINKS6 (Pg. 41)
154 159

215. location 750 (Pg. 46

216. next instruction in sequence
(Pg. 46)

217. 01 2 (Pg. 41)
546 551

218. 901 B 991 (Pg. 69)

219.

Op	A	B	d
A	0538	0823	
A	0533		
A			
A	0524		

(Pg. 79)

220.

Ct	Label	Op	A
19	CONST	DCW	* TOT&COST&&$

24 34
35 42
(Pg. 93)

221. (a) K

(b) SW

(c) B

(d) CS

(Pg. 103)

ANSWERS TO QUIZ cont'd

222. (a)

Op	A	B	d
M C W	CONST	0 2 3 9	
M C E	0 4 2 5	0 2 3 9	
W			

(Pg. 115)

(b)
Print Area: T O T C O S T $ 1 , 2 7 6 . 0 4
220 230 240

223.

Ct	Label	Op	A	B	d
0 8	P R O D	D C W	*		
0 1	K 5	D C W	* 5		
		L C A	C O S T	P R O D - 5	
		M	Q T Y	P R O D	
		A	K 5	P R O D	
		M Z	P R O D	P R O D - 1	

(Pg. 127)

224.

Op	A	B	d
C C			T
W			

(Pg. 119)

225.

Op	A	B	d
L C A	% U 6	1 2 0 0	R

(Pg. 149)

226.

Op	A	B	d
B	W E R R		L
B	W E O R		K

(Pg. 150)

227. S T O P (B > A) (Pg. 46)

228. Next instruction in sequence (B < A). (Pg. 46)

UNIT X

Lesson 40

SYSTEM ANALYSIS: We will set-up an imaginary company which has a central warehouse and 15 branch stores, each ordering its wholesale stocks from the central warehouse. Orders for stock are phoned-in to the inventory control section located at the warehouse. These orders are written on slips, one slip for each type of item. The slips are checked against a ledger showing stocks in the warehouse. If stock is available a "picking-ticket" is made showing the branch's name and address, item stock number (a different number for each type of item), warehouse location, quantity ordered, and total weight. The quantity on the picking ticket is deducted from the ledger sheet and a new balance written-in.

When the ledger record shows no stock-on-hand for an item, a back-order slip is written and placed where it can be checked in the next day's processing before new orders are processed. If partial stocks are available, the partial "picking ticket" is created and the remainder is back-ordered.

When a new shipment of stock is received by the warehouse, the warehouse manager sends a slip called a "notice of receipt," showing the quantity received. This amount is added to the ledger record.

When the orders are filled by the warehouse they are all sent to the shipping department, which sorts the items according to branch destination, packages and ships them. A slip is sent from the shipping department to accounting to show items shipped, branch number, price each, and total price.

A system analyst is asked to design a computer system to perform the work of the inventory control office. After viewing the whole operation, he decides the following records are necessary:

1. Inventory Ledger File: This file will contain one record for each type of item. The records will be sequenced by stock number. The record will keep track of total quantities of each ordered by each branch for a report to be prepared in the future. The following record will be needed:

Data Field	Nr. of Digits
Stock Number	10
Item Name	20
Price	6
Shipping Weight (each item)	4
Warehouse Location	2
Quantity in Stock	8
Quantity on Back-order	8

Data Field (cont'd)		Nr. of Digits
Branch Number-4 digits Total Ordered-6 digits	Once for each branch	150

2. Order Card: When a sales order is received a card will be punched for each type of item ordered:

Data Field	Card Columns
Branch Number	1- 4
Stock Number	5-14
Item Name	15-34
Quantity	35-40

3. Receipt Card: WHen new stock is received at the warehouse a card will be punched for each type of item:

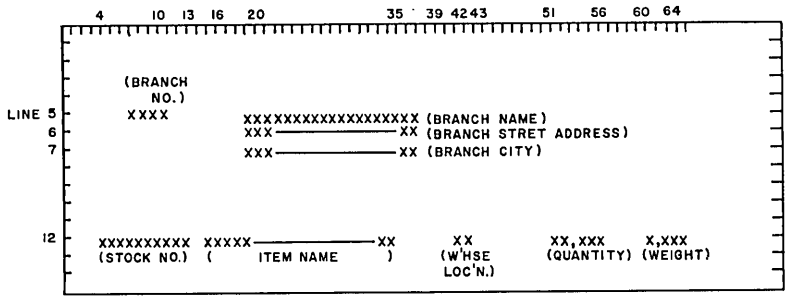
Data Field	Card Columns
Stock Number	5-14
Item Name	15-34
Quantity Received	35-40

(Column 40 will be punched with an 11-punch)

4. Back-Order Card: When stock is not available a card is punched exactly like the order card except for an 11-punch over Column 1. The quantity is the amount <u>not</u> shipped.

5. Picking Ticket: A printed slip will be prepared for each order to be shipped. The slip will be used to select the stock. It will be attached to the container when it is sent to the shipping department.

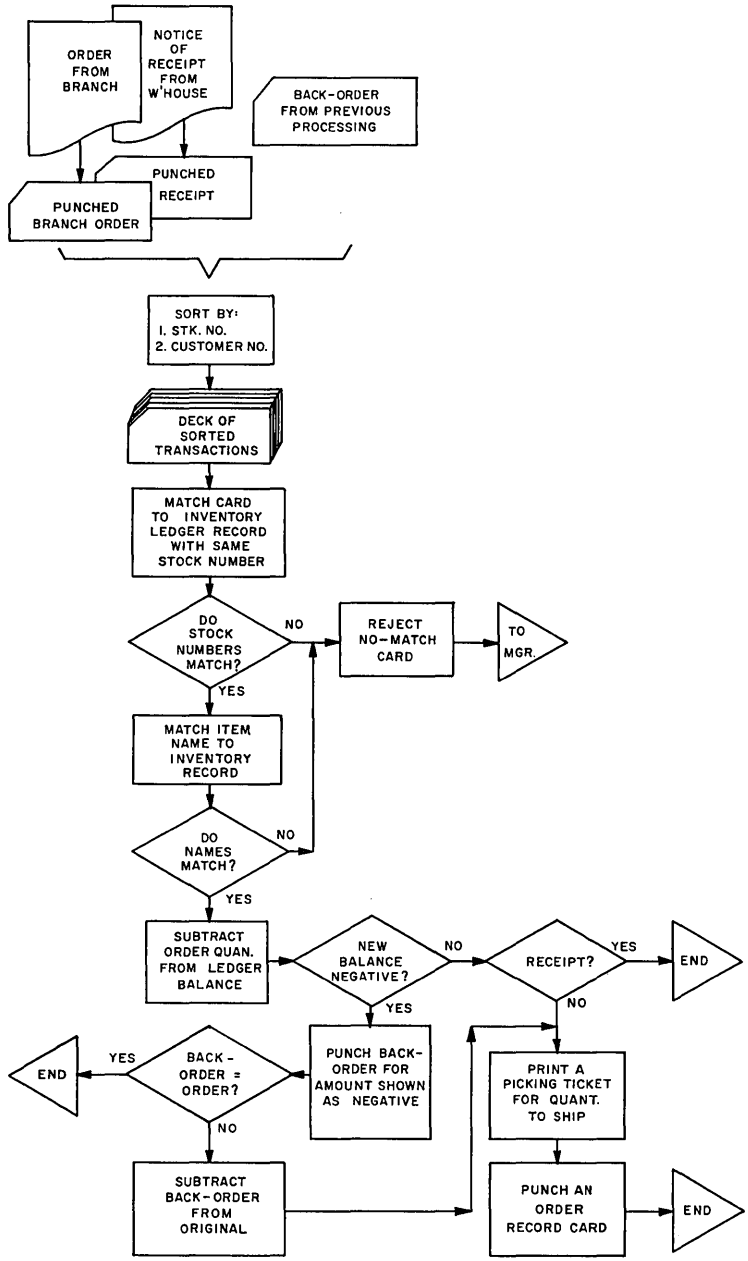
Line 5, Positions	10-13	Branch Number
Line 5, "	20-39	Branch Name
Line 6, "	20-39	Branch Street Number
Line 7, "	20-39	Branch City
Line 12, "	4-13	Stock Number
Line 12, "	16-35	Item Name
Line 12, "	42-43	Warehouse Location
Line 12, "	48-56	Quantity (bb,bbb)
Line 12, "	60-64	Total Wt. (b,bbb)



6. Order Record Card: Each time a picking-ticket is printed a card is punched for use by the shipping department to check-out shipments and then is sent to the accounting department.

Data Field	Card Columns
Customer Number	1- 4
Stock Number	5-14
Quantity Shipped	15-20
Price Each	21-26
Total Price	27-35
Blanks	36-39
Date	40-46
Day (01-31)	Col. 40-41
Month (Jan,Feb,etc.)	42-44
Year (62,63,64,etc.)	45-46

The systems analyst usually draws a system flow chart which shows the order in which actions will take place. Each point at which a decision must be made is shown and how the decision is to be handled.


ORDER FROM BRANCH
NOTICE OF RECEIPT FROM W'HOUSE
BACK-ORDER FROM PREVIOUS PROCESSING
PUNCHED BRANCH ORDER
PUNCHED RECEIPT
SORT BY: I. STK. NO. 2. CUSTOMER NO.
DECK OF SORTED TRANSACTIONS
MATCH CARD TO INVENTORY LEDGER RECORD WITH SAME STOCK NUMBER
DO STOCK NUMBERS MATCH?
NO
REJECT NO-MATCH CARD
TO MGR.
YES
MATCH ITEM NAME TO INVENTORY RECORD
DO NAMES MATCH?
NO
YES
SUBTRACT ORDER QUAN. FROM LEDGER BALANCE
NEW BALANCE NEGATIVE?
NO
RECEIPT?
YES
END
YES
NO
END
YES
BACK-ORDER = ORDER?
PUNCH BACK-ORDER FOR AMOUNT SHOWN AS NEGATIVE
PRINT A PICKING TICKET FOR QUANT. TO SHIP
NO
SUBTRACT BACK-ORDER FROM ORIGINAL
PUNCH AN ORDER RECORD CARD
END

SYSTEM PROGRAMMING: When the programmer discusses the inventory control problem (Lesson 40) with the analyst, he notes certain important points:

1. The inventory ledger records have 208 characters. The analyst tells him there are about 5000 items in the inventory. He decides to put the ledger file on magnetic tape, low-density (200 characters per inch). Each record will use approximately 1 inch of tape, with an IRG of 3/4 inch.

   5000 x 1 3/4 = 8750 inches = 729 feet

   He concludes that a single tape reel will hold all the ledger records. Thus two tape units are needed--one for input and one for outputting the updated file.

2. The three types of input cards (order, receipt, back-order) all have the same data fields in the same columns. (The receipt card has blanks in the Branch Nr.)

3. Both order record cards and back-order cards will be output. He decides to put back-orders in Stacker #4; order record cards in Stacker Np.

4. The order and back-order cards have plus quantities, but the receipt card has a minus quantity (due to the 11-punch in Col. 40). If all transaction quantities are subtracted, the minus quantity (receipt) will be added to the balance--others actually subtracted.

5. The order and back-order cards have only the branch number--not the name and address as required for the "picking ticket." He decides to place a table in memory, in branch number sequence, cross-referencing the number to the name, street address, and city.

Cust. Nr.	Branch Name	Street Address	Branch City
4 char	20 characters	20 characters	20 characters

   Starting at memory address 0501 to 1460 he will read in a table of addresses for 15 branches, each entry 64 characters long. These must be punched in 15 cards (Col. 1-63) and read-in before the data cards are read.

6. The date will be required for the tape label and for the order record card. A card must be punched before each run and read into memory after the program deck, but before the table of addresses.

Data Field	Card Columns
Day of Month	1-2
Month (3 char. abbreviation)	3-5
Year (last 2 digits)	6-7

7. He decides to use the standard tape read and write routines (see Lesson 36). Although the flip-flop is not needed, he concludes that he will leave that feature in the routine since the inventory may grow to more tapes.

8. The 11-punch in Column 1 of the back-order card will make it sort ahead of other orders for the same stock number.

9. A tape label will be required and he chooses a simple form of header label:

Date
INVENTORYbLEDGERblbOFblbbDDMMYY

These 33 digits are entered as a constant. The date is inserted after the date card is read. The input label will be read from tape, and the first 16 characters will be compared to the constant to insure the correct tape is used. The whole label will be written on the output tape.

10. The trailer label will consist of a 6-digit record count only. Six characters must be reserved in memory to keep count of records as they are read. The input trailer label will be compared to this count to insure accuracy. A second 6-character field, followed by a group-mark, must be reserved to tally each record as it is written. After the last record has been written, the record count will be written as the trailer label.

The programmer then prepares a detail flow chart showing each step of computer processing. Practicing programmers differ on the amount of detail to be included, but beginners should probably show each distinct computer operation, using the symbols shown in the introduction. The tape read and write details can be left out since they are shown as separate sub-routines.

The programmer makes the following memory reservations:

0001-0080	Input Card Area	0501-1460	Tables of Names and Addresses
0101-0180	Output Card Area		
0201-0300	Print Area	1501-1708	Tape Read Area
0301-0364	Work Area for Table Data	1500	Reserved for "Last Card" Indicator
		2000	Start Program (ORG)

The computer used has 4000 memory positions, the COMPARE option, and the MULTIPLY-DIVIDE feature.

FINAL PROBLEM: Using the data shown in Lessons 40 and 41, and the special rules shown here, code a program. This program will require approximately 155 individual instruction words.

1. Housekeeping should clear storage, put the date into constant called LABEL which contains the label data.

2. Constants are designated as:

CON15 = Constant of 15

CON1 = " " 1

CONZER = " " 00000000

etc.

3. The read tape and write tape routines are coded in Lesson 36. These may be copied, except that changes must be made to insert the correct read and write addresses, and "last card" check.

4. To get to sub-routines such as the read and write routines--use the following linkage:

M C W	*	+ 0 1 1	R E X I T	+	0 0 3
B	R E A D				
B	*	+ 0 0 1			

This set of instructions will cause the second branch instruction to be inserted at the end of the READ subroutine. The second branch instruction says to branch to the next instruction in sequence.

5. The last card of the DATA deck will be punched with END in columns 1-3. When this card is detected, move an A to 1500. At end-of-reel on the input tape test for an A in 1500--this will indicate that all cards have been processed.

6. Use a sub-routine called "QUANUP" to update totals for each branch.

7. Use labels for constants and for program entry points, but use actual addresses for data fields within the input-output areas and the table work area.

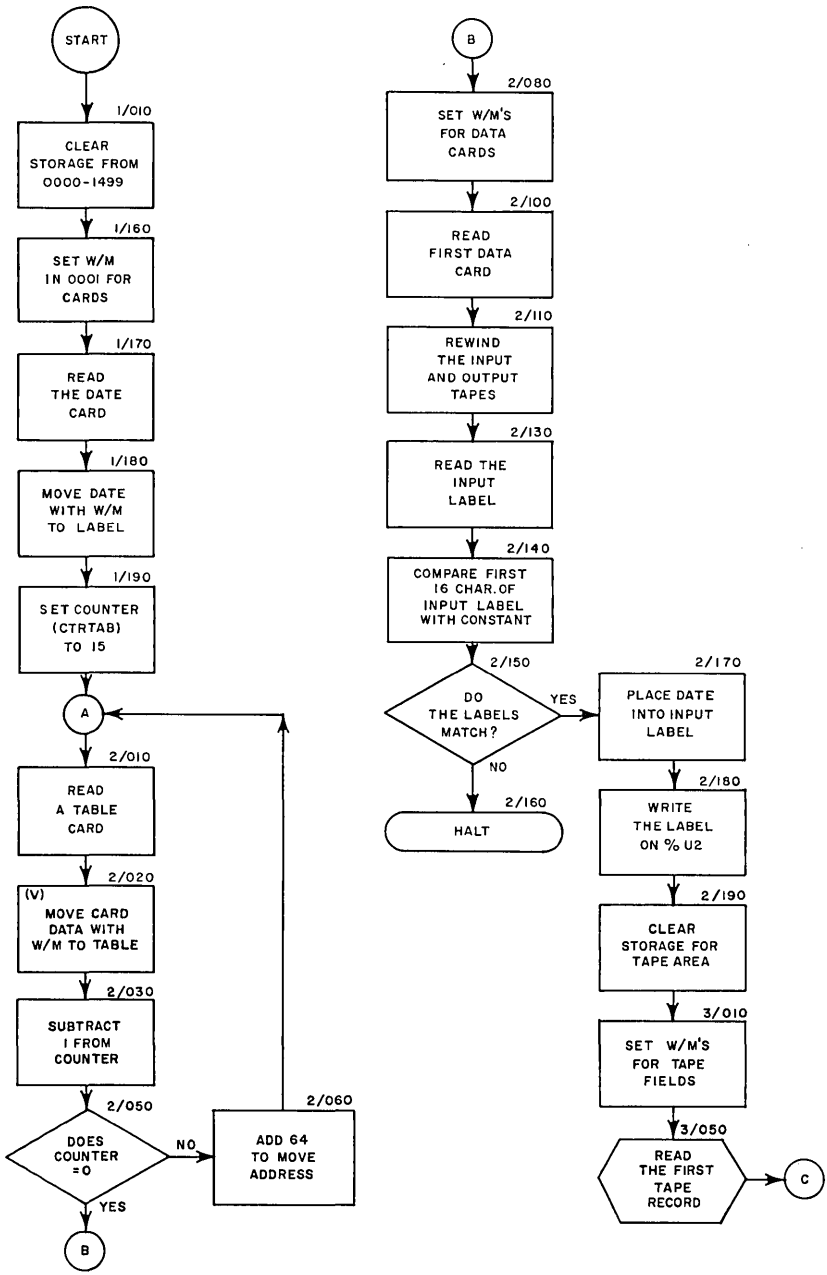
8. The programmer's detailed flow chart is shown on the following pages. Code the entire problem to the best of your ability. The correct answer may be found in the sealed envelope at the back of the book. Do not refer to the correct answer until you have completed your coding--then check it very carefully and compare it to your work. The numbers above the flow chart blocks refer to the page and line number of the correct answer.

CONCLUDING REMARKS

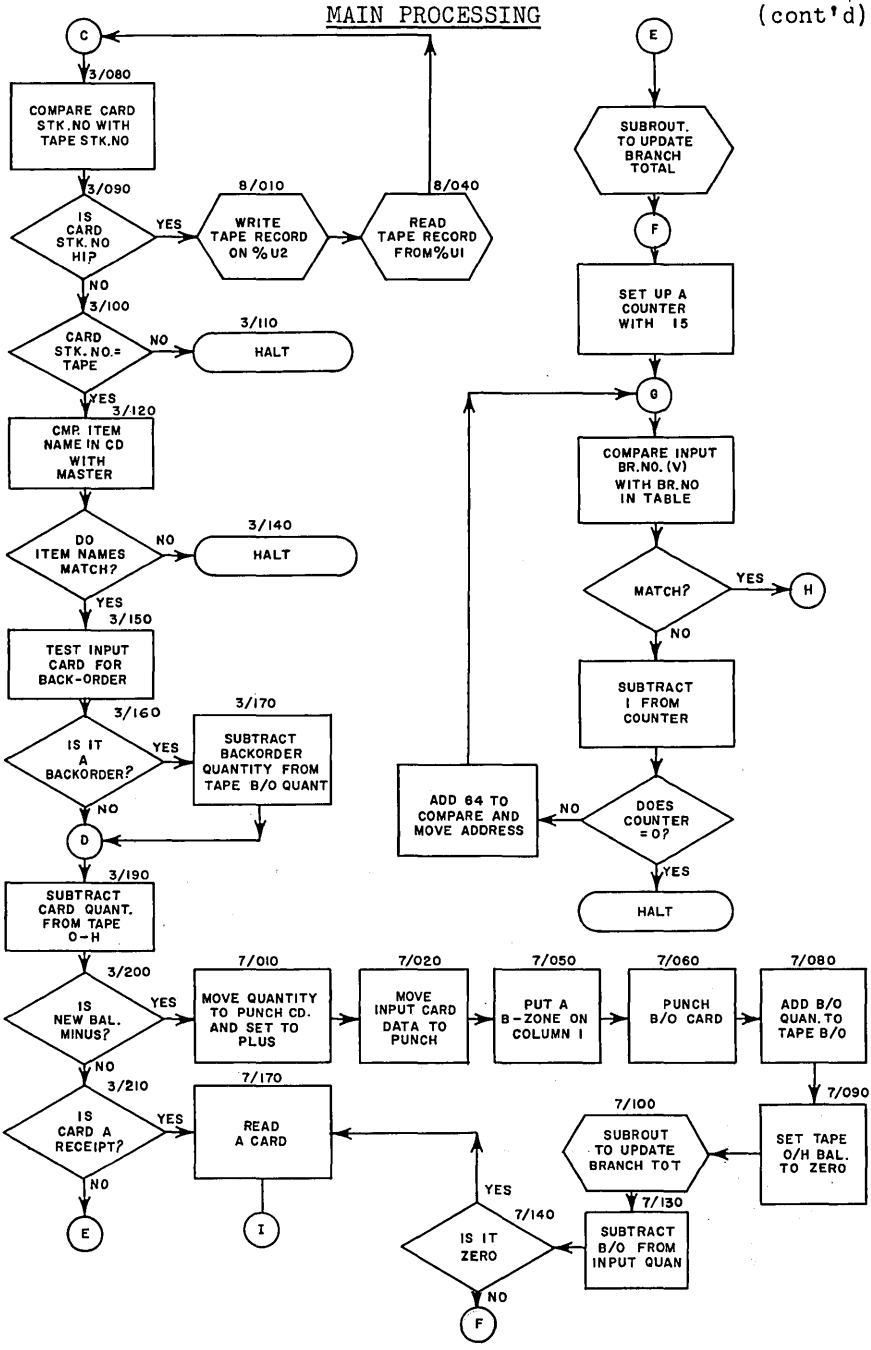
It must be understood that what you have learned is only the beginning of the learning process. To become an accomplished programmer, you must work with the machine and with the problems to be solved by the machine. Nothing can be substituted for experience.

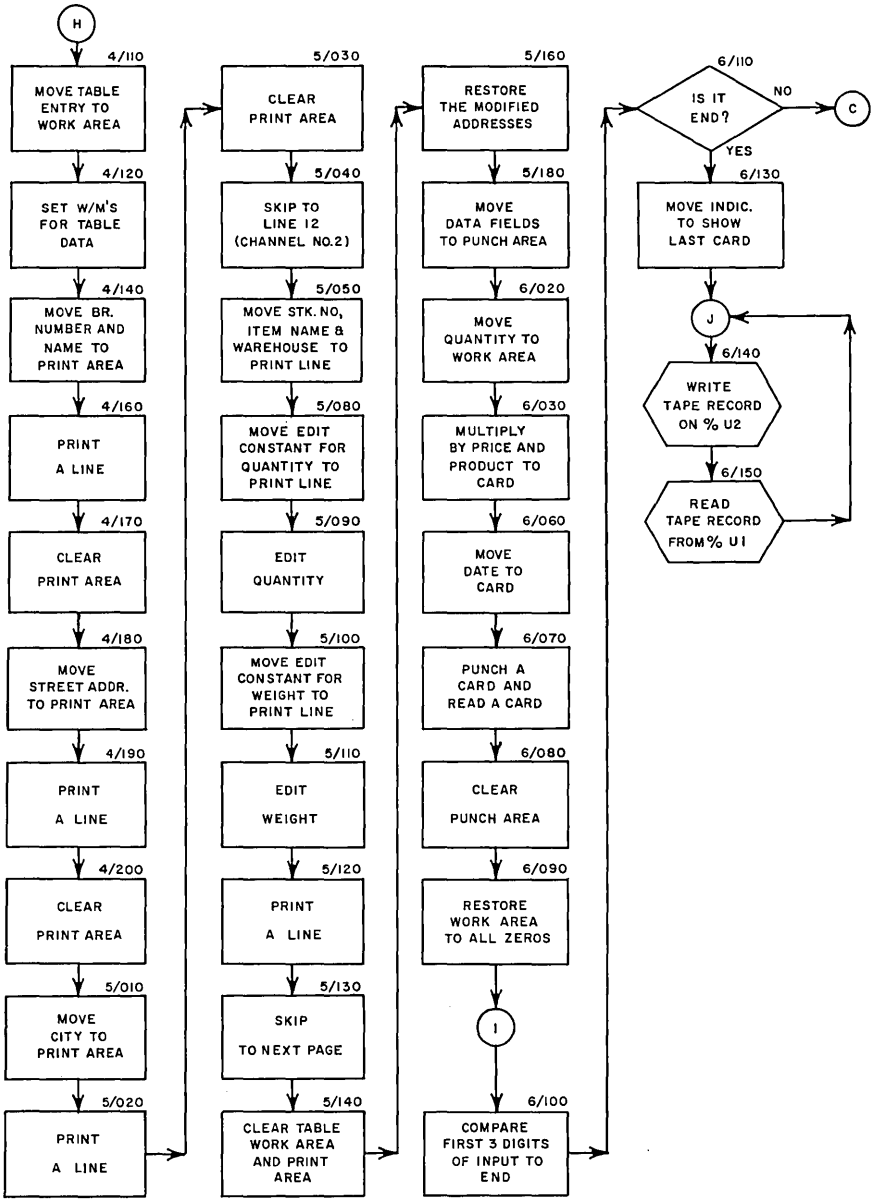
Many of the areas covered in the book only give you a basic idea that such a method exists. No more is possible in a book of this nature (or in a short lecture course, for that matter). Constant use of the concepts and instructions will do more than anything else to implant them firmly in your mind. Good luck in your new profession.

HOUSEKEEPING

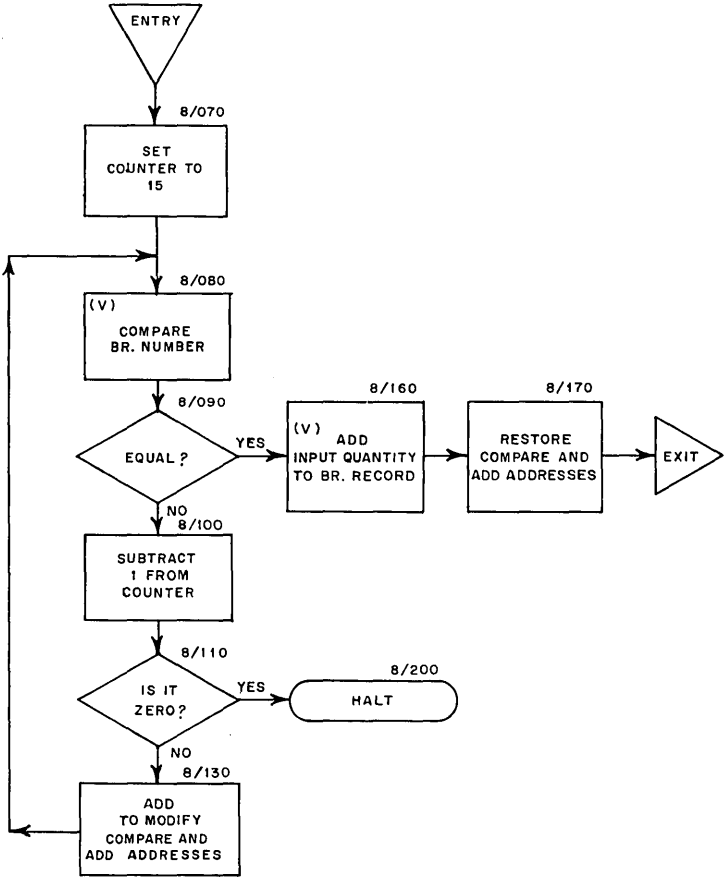


MAIN PROCESSING




H
4/110
MOVE TABLE ENTRY TO WORK AREA
4/120
SET W/M'S FOR TABLE DATA
4/140
MOVE BR. NUMBER AND NAME TO PRINT AREA
4/160
PRINT A LINE
4/170
CLEAR PRINT AREA
4/180
MOVE STREET ADDR. TO PRINT AREA
4/190
PRINT A LINE
4/200
CLEAR PRINT AREA
5/010
MOVE CITY TO PRINT AREA
5/020
PRINT A LINE
5/030
CLEAR PRINT AREA
5/040
SKIP TO LINE 12 (CHANNEL NO.2)
5/050
MOVE STK. NO, ITEM NAME & WAREHOUSE TO PRINT LINE
5/080
MOVE EDIT CONSTANT FOR QUANTITY TO PRINT LINE
5/090
EDIT QUANTITY
5/100
MOVE EDIT CONSTANT FOR WEIGHT TO PRINT LINE
5/110
EDIT WEIGHT
5/120
PRINT A LINE
5/130
SKIP TO NEXT PAGE
5/140
CLEAR TABLE WORK AREA AND PRINT AREA
5/160
RESTORE THE MODIFIED ADDRESSES
5/180
MOVE DATA FIELDS TO PUNCH AREA
6/020
MOVE QUANTITY TO WORK AREA
6/030
MULTIPLY BY PRICE AND PRODUCT TO CARD
6/060
MOVE DATE TO CARD
6/070
PUNCH A CARD AND READ A CARD
6/080
CLEAR PUNCH AREA
6/090
RESTORE WORK AREA TO ALL ZEROS
I
6/100
COMPARE FIRST 3 DIGITS OF INPUT TO END
6/110
IS IT END?
NO
C
YES
6/130
MOVE INDIC. TO SHOW LAST CARD
J
6/140
WRITE TAPE RECORD ON % U2
6/150
READ TAPE RECORD FROM % U1

Special QUANUP Sub-routine



(Variable addresses are indicated by V)

INDEX

TABLES

ILLUSTRATIONS

ANSWERS

LINE (3–5)	COUNT (6–7)	LABEL (8–13)	OPERATION (14–16)	(A) OPERAND ADDRESS (17–23)	(A) OPERAND ±	(A) OPERAND CHAR. ADJ.	(A) OPERAND IND. (27)	(B) OPERAND ADDRESS (28–34)	(B) OPERAND ±	(B) OPERAND CHAR. ADJ.	(B) OPERAND IND. (38)	d (39)	COMMENTS (40–55)
010		START	CS	1499									CLEAR STORAGE BY
020			CS										CHAINING
030			CS										
040			CS										
050			CS										
060			CS										
070			CS										
080			CS										
090			CS										
100			CS										
110			CS										
120			CS										
130			CS										
140			CS										
150			CS										
160			SW	0001									SET WM FOR CARD
170			R										READ DATE CARD
180			LCA	0007				LABEL	-	001			ST DATE IN LABEL
190			ZA	CON15				CTRTAB					SET CTR TO 15
200													

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
3 5	6 7	8 13	14 16	17	23		27	28	34		38	39	40 55
010		BLDTAB	R										READ TABLE CARDS
020			LCA	0064				0564					STORE IN TABLE
030			S	CON1				CTRTAB					SUB FROM CTR
040			C	CTRTAB				CONZER					CTR EQUAL ZERO
050			B	TABEND								S	IF SO, END OF TAB
060			A	CON64				BLDTAB	+	007			IF NOT MOD ADDR
070			B	BLDTAB									REPEAT CYCLE
080		TABEND	SW	0001				0005					SET WM FOR CARDS
090			SW	0015				0035					
100			R										READ 1ST DATA CD
110			CU	%U1								R	REWIND INPUT
120			CU	%U2								R	REWIND INPUT
130			MCW	%U1				1501				R	READ INPUT LABEL
140			C	0416				LABEL	-	017			CHECK LABEL
150			B	*	+	002						S	GO AHEAD IF EQU
160			H										IF NOT - HALT
170			MCW	LABEL	-	001		1526					UPDATE LABEL
180			MCW	%U2				1501					PUT LABEL ON OUT
190			CS	1708									CLEAR TAPE AREA
200			CS										
210			CS										

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
010			SW	1501				1511					SET WMS FOR TAPE
020			SW	1531				1537					RECORDS
030			SW	1541				1543					
040			SW	1551				1559					
050			MCW	*	+	011		REXIT	+	003			SET READ EXIT
060			B	READ									BRANCH TO READ
070			B	MAIN									RETURN ADDRESS
080		MAIN	C	1510				0014					COMP STOCK NRS
090			B	WRITA								U	CARD S/N IS HIGH
100			B	*	+	002						S	CARD S/N EQ TAPE
110			H										NO MATCH-STOP
120			C	0034				1530					COMP ITEM NAME
130			B	*	+	002						S	MATCH-GO AHEAD
140			H										NO MATCH-STOP
150			BZW	*	+	009		0001				K	IS INPUT BACK-Ø
160			B	SUBQTY									IF NOT GO AHEAD
170			S	0040				0458					SUB
180			B	SUBQTY									
190		SUBQTY	S	0040				1550					SUB CARD QUAN
200			BWZ	BACKØ				1550				K	IS NEW BAL MINUS
210			BWZ	READA				0040				K	IS CD A RECEIPT?
220			MCW	*	+	011		QUEXIT	+	003			SET SR EXIT
230			B	QUANUP									BR TO SUB-ROUT
240			B	PRINT									RETURN ADDRESS

LINE (3–5)	COUNT (6–7)	LABEL (8–13)	OPERATION (14–16)	(A) OPERAND ADDRESS (17)	(A) ± (23)	(A) CHAR. ADJ.	(A) IND. (27)	(B) OPERAND ADDRESS (28)	(B) ± (34)	(B) CHAR. ADJ.	(B) IND. (38)	d (39)	COMMENTS (40–55)
010		PRINT	ZA	CON15				CTRTAB					SET CTR TO 15
020			C	0004				0504					COMPR NR-TABLE
030			B	FIND								S	OK-NR IS MATCH
040			S	CON1				CTRTAB					SUB 1 FROM CTR
050			C	CTRTAB				CONZER					COUNTER EQ ZERO?
060			B	ERRTAB								S	IF ZERO-ERROR
070			A	CON64				PRINT	+	013			MODIFY COMPARE
080			A	CON64				FIND	+	003			MODIFY MOVE
090			B	PRINT	+	007							CONTINUE SEARCH
100		ERRTAB	H										ERROR-HALT
110		FIND	MCW	0564				0364					MOVE ENTRY TO WA
120			SW	0301				0305					SET WMS FOR WORK
130			SW	0325				0345					AREA FOR TABLE
140			MCW	0304				0213					MOVE BR NR TO PR
150			MCW	0324				0239					MOVE BR NAME
160			W										PRINT A LINE
170			CS	0299									CLEAR PRINT AREA
180			MCW	0344				0239					MOVE BR ADDRESS
190			W										PRINT A LINE
200			CS	0299									CLEAR PRINT AREA

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
3 5	6 7	8 13	14 16	17	23		27	28	34		38	39	40 55
010			MCW	0364				0239					MOVE BR CITY
020			W										PRINT A LINE
030			CS	0299									CLEAR PRINT AREA
040			CC									2	SKIP TO LINE 12
050			MCW	1510				0213					MOVE STK NR
060			MCW	1530				0235					MOVE ITEM NAME
070			MCW	1542				0243					MOVE WAREHSE LOC
080			MCW	EDQUAN				0256					SET EDIT CONST
090			MCE	0040				0256					EDIT QUANTITY
100			MCW	EDWGT				0264					SET EDIT CONST
110			MCE	1540				0264					EDIT WEIGHT
120			W										PRINT A LINE
130			CC									1	SKIP TO NEXT PGE
140			CS	0364									CLEAR TABLE WA
150			CS										CLEAR PRINT AREA
160			MCW	CONAD1				PRINT	+	013			RESTORE TABLE
170			MCW	CONAD2				FIND	+	003			RESTORE MOVE
180			MCW	0004				0104					MOVE BR NR-CARD
190			MCW	1510				0114					MOVE STK NR
200			MCW	0040				0120					QUANTY TO CARD

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
010			MCW	1536				0126					PRICE EACH
020			MCW	1536				MPYWA	-	007			PUT PR IN WORK A
030			MPY	0040				MPYWA					MULTIPLY
040			SW	MPYWA	-	009							SET WM FOR PROD
050			MCW	MPYWA				0135					MOVE PROD TO CD
060			MCW	LABEL	-	001		0146					MOVE DATE TO CD
070			RP										PUNCH CD AND RD
080			CS	0180									CLEAR PUNCH AREA
090			LCA	MPYCON				MPYWA					RESTORE WORK AR
100		ENDA	C	CONEND				0003					CHECK CD FOR END
110			B	*	+	006						S	IF END-SKIP
120			B	MAIN									IF NOT REPEAT
130			MCW	CONA				1500					MOVE IND TO RD
140		WRITX	MCW	*	+	011		WEXIT	+	003			WRITE RECORD
150			B	WRITE									
160			B	READX									
170		READX	MCW	*	+	011		REXIT	+	003			READ RECORD
180			B	READ									
190			B	WRITX									REPEAT UNTIL EOF
200													

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
010		BACKØ	ZS	1550				0140					SET QUAN TO PLUS
020			MCW	0034				0134					MOVE INPUT CARD
030			MCW										TO OUTPUT FOR
040			MCW										BACK-ORDER
050			ZS	0101									PUT B-ZONE ON 1
060			P										PUNCH BACK-Ø CD
070			SS									4	SEL STACKER NR 4
080			A	0140				1558					ADD QUAN TO B/Ø
090			ZA	CONZER				1550					SET Ø/H TO ZERO
100			MCW	*	+	011		QUEXIT	+	003			GO TO SUB-ROUT
110			B	QUANUP									
120			B	SUBIN									RETURN ADDRESS
130		SUBIN	S	0140				0040					SUB B/Ø FROM INP
140			C	0040				CONZER					SEE IF QUAN=ZERO
150			B	READA								S	IF SO-READ NEXT
160			B	PRINT									IF NOT ORDER
170		READA	R										READ NEXT CARD
180			B	ENDA									GO BACK
190													
200													

LINE (3–5)	COUNT (6–7)	LABEL (8–13)	OPERATION (14–16)	(A) OPERAND ADDRESS (17)	(A) OPERAND ± (23)	(A) OPERAND CHAR. ADJ.	(A) OPERAND IND. (27)	(B) OPERAND ADDRESS (28)	(B) OPERAND ± (34)	(B) OPERAND CHAR. ADJ.	(B) OPERAND IND. (38)	d (39)	COMMENTS (40–55)
010		WRITA	MCW	*	+	011		WEXIT	+	003			WRITE RECORD
020			B	WRITE									
030			B	WRITA	+	015							
040			MCW	*	+	011		REXIT	+	003			READ NEXT
050			B	READ									
060			B	MAIN									GO BACK
070		QUANUP	ZA	CONN15				CTRTAB					SET-UP COUNTER
080			C	0004				1562					COMPARE BR NR
090			B	QFIND								S	IF EQUAL-FOUND
100			S	CONN1				CTRTAB					SUB 1 FROM CTR
110			C	CTRTAB				CONZER					COUNTER EQ ZERO
120			B	QERROR								S	IF SO-ERROR
130			A	CON10				QUANUP	+	013			MODIFY COMPARE
140			A	CON10				QFIND	+	006			MODIFY ADD
150			B	QUANUP	+	007							REPEAT LOOP
160		QFIND	A	0040				1568					UPDATE QUAN
170			MCW	CONAD3				QUANUP	+	013			RESTORE ADDRESS
180			MCW	CONAD4				QFIND	+	006			
190		QUEXIT	B	QUEXIT									RETURN FROM S-R
200		QERROR	H										ERROR-HALT

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
3 5	6 7	8 13	14 16	17	23		27	28	34		38	39	40 55
010	33	LABEL	DCW	*		INVEN		TORY		LEDGER			1 OF 1 DDMMMYY #
020	02	CON15	DCW	*		15							
030	02	CTRTAB	DCW	*		00							
040	02	CON64	DCW	*		64							
050	02	CON1	DCW	*		01							
060	08	CONZER	DCW	*		0000	0	000					
070	06	EDQUAN	DCW	*		,		0					
080	05	EDWGT	DCW	*		,	0						
090	13	MPYWA	DCW	*		0000	0	0000000		0			
100	13	MPYCON	DCW	*		0000	0	0000000		0			
110	03	CONEND	DCW	*		END							
120	01	CONA	DCW	*		A							
130	02	CON10	DCW	*		10							
140	03	CONAD1	DCW	*		504							
150	03	CONAD2	DCW	*		564							
160	03	CONAD3	DCW	*		V62							
170	03	CONAD4	DCW	*		V68							
180													
190													
200													