PROGRAMMING THE
IBM
1401

A SELF-INSTRUCTIONAL PROGRAMMED MANUAL

Presents the basic principles and methods of programming the IBM 1401 Computer

JAMES A. SAXON • WILLIAM S. PLETTE

SAXON
•
PLETTE

PRENTICE
HALL

Programming the IBM 1401: A Self-Instructional Programmed Manual, by James A. Saxon and William S. Plette, answers a major training problem of businesses that use the IBM 1401.

Training programs for present employees and incoming personnel are costly and time consuming. The major advantage of this manual is that the student can train himself, at his own pace, in the basic principles and methods of computer programming. The authors have used typical computer problems to create on the job conditions, giving the student a professional insight into techniques and short cuts normally gained through actual experience.

This manual is primarily concerned with business applications of the 1401 with some attention given to scientific and mathematical uses. The subject matter is presented in a series of concise lessons. Each lesson contains practical problems, examples, and visual aids, all of which assure maximum comprehension. Self-scored tests at the end of each lesson allow the student to check his progress.

Programming the IBM 1401 teaches the language of these computers with a minimum of time, expense, and effort, while greatly increasing an employee's competence and job value.

Another book by JAMES A. SAXON . . .

PROGRAMMING THE IBM 7090: A Self-Instructional Programmed Manual

This manual is a direct step to increasing competency and job value by teaching the language of 7090 computers with a minimum of time, expense and effort.

The subject matter is presented in a series of concise lessons designed to permit the learner to proceed at his own pace. Frequent tests provide reinforcement and measure progress.

Published January, 1963   *210 pages*

Other books of interest . . .

PROGRAMMING THE IBM 1620

by CLARENCE B. GERMAIN, *College of St. Thomas*

A first course in digital computer programming based on the IBM 1620 computer, covering all the special aspects of the machine.

Outstanding features include:

- all 1620 features are covered as well as the SPS, FORTRAN, etc., programming systems
- nine appendices provide a complete summary of 1620 operations, character codes, instructional timings, and a summary of the one-pass and two-pass SPS languages and operation of the corresponding processors
- first programming book based on the IBM 1620
- may be used with either a card or tape 1620

Published 1962   *208 pages*

COMPUTER LOGIC: The Functional Design of Digital Computers

by IVAN FLORES, *Consultant, Norwalk, Connecticut*

This book on computers describes and illustrates with block diagrams exactly how a computer is put together and how it works.

The author emphasizes the organization and functional interrelation of fundamental units and discusses these relationships from the viewpoint of operational necessity rather than mathematical logic, Boolean algebra or circuit theory.

Among the features the reader will find especially helpful are:

- the extensive treatment of two often-neglected topics: (1) input, output and (2) control of computer function
- the use of functional symbols for functional parts of equipment
- the problems that provide a means of acquiring facility with computer terms
- the final chapter that reviews in detail what happens from the time the programmer receives the problem until the answer is ready

Published 1960   *458 pages*

PROGRAMMING THE IBM 1401:

A Self-Instructional Programmed Manual

JAMES A. SAXON

Saxon Research Corporation

WILLIAM S. PLETTE

United Research Services

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

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 instruc-
tions 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 process-
> ing, 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 fol-
lowing 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 <u>turned</u> <u>out</u> by the
   machine.

   <u>Program</u> <u>planning</u> is one of the most important functions
in programming. When it has been determined that a partic-
ular 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 re-
quires 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.

   <u>Flow</u> <u>charting</u>: Before writing machine instructions, it
is usual to <u>flow</u> <u>chart</u> the necessary steps to be taken.
This has the advantage of proving the logic of the applica-
tion 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 <u>coding</u> (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 stand-
ardization 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
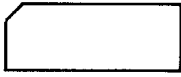
IBM PUNCH CARD - Denotes input or
output data in 80-
column cards

MAGNETIC TAPE - Denotes input or
output data on
magnetic tape
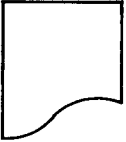
PRINTED OUTPUT - Denotes output in a
printed form

START - Denotes the point at which a
program begins

HALT - Denotes the point at which a
program ends
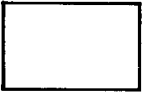
PROCESSING
BLOCK   - Fill with a brief
description of each
discrete process
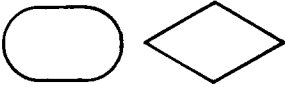
DECISION
BLOCKS   - Denotes the point at
which a program branch-
es due to a decision

SUB-ROUTINE- Represents a "program"
within the main program
which may be flow-
charted in detail
elsewhere
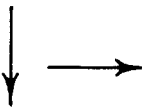
DIRECTION
OF FLOW   - Used to connect the
other symbols

CONNECTORS - Used to show how the
sections of the flow-
chart 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.

U N I T I

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 be-
ing referenced. The number representing a storage position
permits the programmer to address any specific storage posi-
tion 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 char-
acter, 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 repre-
sented as follows:

$$\boxed{2}\boxed{7}\boxed{4}\boxed{9}$$
120 121 122 123

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 charac-
    ter from a specific cell in storage,
    he is _____ that
    memory position.

    5. _____

NOTES AND ANSWERS

ANSWERS   REMINDERS

1. | 5 | 2 | 7 |
   300 301 302

2. | P |
   512

3. | H | O | L | D |
   101 102 103 104

4. (a)_____000_____

   (b)____7999_____

5. __Addressing__

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 char-
acters 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 address-
ing 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 con-
nection 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:

2	3	3	9	5
421 422 423 424 425
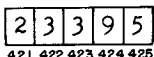
The word STEP occupying storage locations 001-004 would be
considered a computer word pictured as follows:

S	T	E	P
001 002 003 004
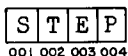
Position 421 in the first example and 001 in the second are
the high-order positions.
Position 425 in the first example and 004 in the second are
the low-order positions.

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   (a) | 1 | 2 | 3 | 4 | 5 | 6.(a)_____
    the high-order characters in   270   274
    the following examples: (b)_____
                                    (b) | 5 | T |
7. Write the position numbers of   102 103   (c)_____
    the low-order characters in
    the same examples.   (c) | F | I | N | A | L | 7.(a)_____
                                         906   910
                                                                   (b)_____
8. A single character or group of characters
    that represent a unit of information, is   (c)_____
    called a_____.
                                                                   8._____

3

Lesson 2

NOTES AND ANSWERS

ANSWERS   REMINDERS

6. (a)____270____   The <u>high-order</u> position of the
                            word is always the <u>left-most</u>
   (b)____102____   digit or position.

   (c)____906____

7. (a)____274____   The <u>low-order</u> position of the
                            word is always the <u>right-most</u>
   (b)____103____   digit.

   (c)____910____

8. _____WORD_____   A <u>word</u> may occupy as many con-
                            secutive storage positions as
                            are necessary to contain a
                            unit of information.

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

EXAMPLE:

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

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

PROBLEMS: Write your answers to the following questions in
the space provided, referring as necessary to the text.
<u>After</u> 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.

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

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

9.   <u>High</u>   <u>Low</u>

(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)_____

5

NOTES AND ANSWERS

ANSWERS		REMINDERS

High   Low

9.(a) 200   202   The word mark always indicates the
                      high-order position. The last two
 (b) 203   206   words are only one character in
                      length, therefore the single char-
 (c) 207   210   acter in each case is both the
                      high- and low-order position.
 (d) 211   215

 (e) 216   216

 (f) 217   217

10.   ┌─────────────┐
      │ P R I C E │   The word mark should be associated
      └─────────────┘   with the P or high-order position.
      501   505

11.(a) VARIABLE   Since a word may be of any length
                        in the IBM 1401, it is called a
 (b) WORD   variable word-length computer.

 (c) LENGTH

6

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 re-
ferred 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 be-
gins 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.

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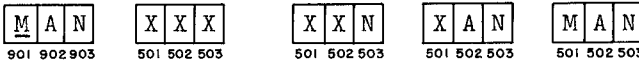
First the N would move from 903 to 503; then the A would
move from 902 to 502; finally, the M would move from 901 to
501 and the word mark (W/M) in 901 would stop the move
operation.

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 | (b) | 1 | 3 | 5 | 7 | 6 |
     225   228   004   008

(c) | 3 | T | 7 | (d) | S | T | O | P |
     202   204   205   208
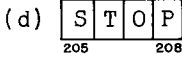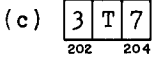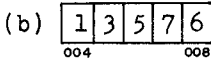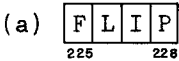
12. Write the storage position number to
    be addressed in each of the above.

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)_____

7

Lesson 4

NOTES AND ANSWERS

ANSWERS   REMINDERS

12. (a) __228__   Data words or fields are
                                 addressed by their right-most
    (b) __008__   or low-order positions.

    (c) __204__

    (d) __208__

13. (a) __225__   W/M's are associated with the
                                 left-most or high-order
    (b) __004__   position.

    (c) __202__

    (d) __205__

INSTRUCTION WORD: The computer performs operations by fol-
lowing instructions prepared by a programmer. Each instruc-
tion 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 in-
                             struction and must have a word
                             mark.
A - ADDRESS   3 digits   The address of the low-order char-
                             acter in a data field which is to
                             be operated-on. In a MOVE instruc-
                             tion, for example, the field from
                             which data will move.
B - ADDRESS   3 digits   The address of the low-order char-
                             acter in the second field, if any,
                             to be operated-on. In a MOVE in-
                             struction, for example, the field
                             to which data will move.
DIGIT MODIFIER   1 digit   This character, if present, usu-
                             ally modifies the OPERATION CODE,
                             making it possible to use the same
                             OP CODE for several different but
                             similar processes.

EXAMPLES:

O P	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)_   14.(a)_____
     characters.   (b)_____

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

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

Lesson 5

NOTES AND ANSWERS

ANSWERS   REMINDERS

14. (a)_____1_____   The number of characters in an
                              instruction depends upon the
    (b)_____8_____   kind of instruction and what
                              it is to do.

15. (a)_OPERATION_   The operation code must always
                              be present and must have a W/M
    (b)___CODE___   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 in-
struction 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 sub-
stituting 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	∅00 to ∅99
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 underline hundreds position.

EXAMPLE: | ‡ | 9 | 9 | = 1099
                  HUNDREDS TENS UNITS

You will have noticed that the conversion from 2600-2699
is ∅00 to ∅99. The ∅ symbol is used to designate the alpha-
betic O. If there is no slash through the O, it is con-
sidered to be a zero.

Examples of the use of these special codes:

                1. S57 = 1257

                2. M39 = 2439

                3. ∅00 = 2600

                4. /22 = 1122

                5. D78 = 3478

                6. I99 = 3999

                7. Ö15 = 3015

                8. R09 = 2909

                9. Z11 = 1911

               10. ‡55 = 1055

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

17. _____

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

    a. | 5 | 6 | 7 | 8 |
         001   004

    b. | J | K | L | M |
         220   223

    c. | 1 | 5 |
         601 602

18. high order

    a._____

    b._____

    c._____

    low order

    a._____

    b._____

    c._____

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

19. a._____

    b._____

    c._____

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

20. a._____

    b._____

    c._____

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

21. a._____

    b._____

22. Name the four parts of an instruction.

22. a._____

    b._____

    c._____

    d._____

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

23. a._____

    b._____

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

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

24. a._____

    b._____

    c._____

    d._____

    e._____

    f._____

13

UNIT I QUIZ

17. | A R E A | Refer to page 1
        901   904

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

14

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

Read Area: Positions 001 through 080 are reserved for in-
formation 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 corres-
ponding 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 in-
formation 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:

(READ AREA)

CARD

001   080

(PUNCH AREA)

101   180

PRINTOUT

(PRINT AREA)

201   300
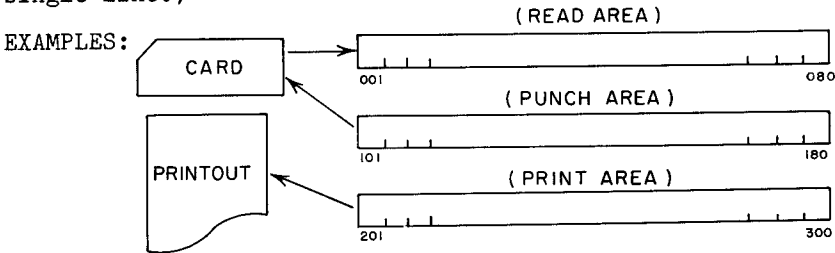
PROBLEMS:

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)_____

.

NOTES AND ANSWERS

ANSWERS   REMINDERS

25.(a)____001____   A knowledge of these reserve
                          areas of storage is absolute
   (b)____080____   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 destroy-
ing 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

Before execution

A
| X | X | X |
001   003

B
| X | X | X |
023   025

After execution

A
| X | X | X |
001   003

B
| X | X | X |
023   025

17
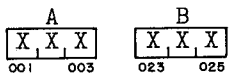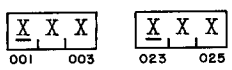
2. Instruction - "Set word mark in position 1250"

Op	A
,	S 5 0

Before execution $\underline{X}_, X_, X$
                   1250   1252

After execution $\underline{X}_, X_, X$
                   1250   1252

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

A: $\underline{5}_, 5_, 5_, 5$ B: [   ]
   052   055   052   055

After execution

A: $\underline{5}_, 5_, 5_, 5$ B: 5, 5, 5, 5
   052   055   052   055

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

Op	A	B
M	1 5 2	T 5 3

Before execution

A: $\underline{T}, O\ M$ B: $\underline{4}, 7, 6\ 2$
   150   152   T50   T53

After execution

A: $\underline{T}, O\ M$ B: $\underline{4}, T, O\ M$
   150   152   T50   T53

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

A: $\underline{1}, 2, 3, \underline{4}, 5\ 6$ B: $\underline{7}, 7, 7$
   024   029   301   303

After execution

A: $\underline{1}, 2, 3, \underline{4}, 5\ 6$ B: $\underline{4}, 5\ 6$
   024   029   301   303

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

Op	A	B
,	0 2 1	0 5 1

Before execution

A: $S, \underline{E}, T$ B: [   ]
   021   023   051   053

After execution

A: $\underline{S}, \underline{E}\ T$ B: $\underline{\ }, $
   021   023   051   053

18

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

A: |3 2 2 3 4| 051 ... 055
B: |9 8 7 6| 422 ... 425

30. B: |_ _ _ _| 422 ... 425

31. M 0 0 5 0 2 5

A: |A B C D| 002 ... 005
B: |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

A: |3 3 3 3| J06 ... J09
B: |4 4 4 4| 498 ... 501

33. B: |_ _ _ _| 498 ... 501

34. M 0 0 7 0 2 7

A: |P| 007
B: |J J J| 025 ... 027

34. B: |_ _ _| 025 ... 027

35. M 0 2 2 0 5 1

A: |6 6| 021 022
B: |8 8 8 8| 049 ... 052

35. B: |_ _ _ _| 049 ... 052

36. ' 0 0 1 0 2 3

A: |S E T| 001 ... 003
B: |H O L D| 021 ... 024

36. A: |_ _ _| 001 ... 003
    B: |_ _ _ _| 021 ... 024

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 | 2 | 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 informa-
tion 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 charac-
ters 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.

21

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:

   Read area   [///////]_____|
              001   005   080

   Punch area |_____[///////]_____|
              101   111 115   180

   To get the data from read area to punched card:

   (a) Set W/M at 001

        | , | 0 0 1 | Before | 7 5 C D E | After | 7 5 C D E |
          ‾   001   005   001   005

   (b) Move data from read area to punch area

        | M | 0 0 5 | 1 1 5 | Before | 7 5 C D E | After | 7 5 C D E |
                                        001   005   111   115
                                        ‾   ‾

   (c) Punch a card

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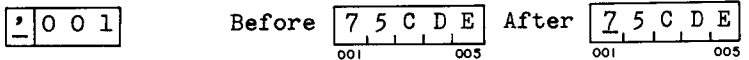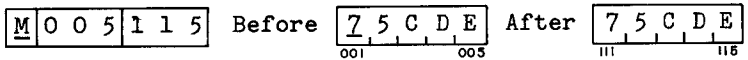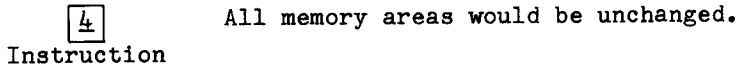
2. Let us assume that in the same example we also wish to
   print the five characters but not together. We want to
   print the first two digits in print positions 4-5 and
   the last three digits in print positions 11-13.

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

        | , | 0 0 1 | 0 0 3 | Before | 7 5 C D E | After | 7 5 C D E |
          ‾   001   005   001   005
                                       ‾   ‾   ‾

   (b) Move data from read area to print area

        | M | 0 0 2 | 2 0 5 | Before | 7 5 C D E | After | 7 5 |
              Instruction   001   005   204 205
                                         ‾   ‾   ‾
        | M | 0 0 5 | 2 1 3 | Before | 7 5 C D E | After | C D E |
                                        001   005   211   213
                                        ‾   ‾
        Print area |_____7 5_____C D E_____~~~_____|
                   201   204 205   211 213   300
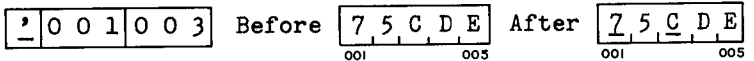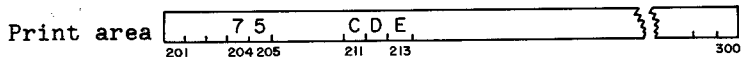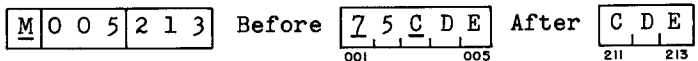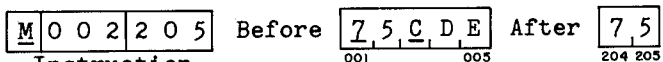
(c) Print data

$$\boxed{\underline{2}}$$ 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 de-
sired 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.

NOTES AND ANSWERS

ANSWERS   REMINDERS

37. ___290___   Simply add 100 to any desired
                       punch position 0-80 to get the
                       punch-area address.

38. ___114___   Add 200 to any desired print
                       position to get the print-area
39. ___125___   address.

40. ___234___

41. ___180___

42. | M̲ | 0 8 0 | 1 8 0 | It is usually best to set word
                              marks in the read-area when
                              the program begins and leave
                              them as long as the fields
43. | ⸴̲ | 0 2 5 | within the input card are in
                              the same positions.

ERASING INFORMATION: The following instructions erase in-
formation 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 posi-
tions 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 pres-
ent 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

| ⌶ | 0 0 1 | 0 0 5 | Before | A B C D E | After | A B C D E |
Instruction

2. Clear read area

| / | 0 8 0 | Before | J O H N D O E 2 4 7 6 } 4 9 |
                                 001   080

After | } |
        001   080

25
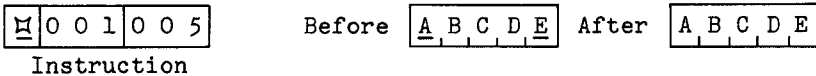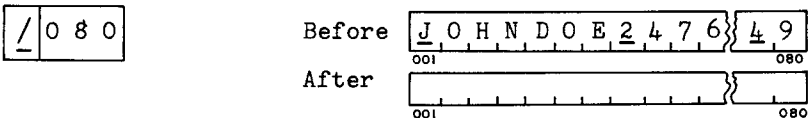
3. Clear punch area and take next instruction from 900

<u>/</u> 9 0 0	1 8 0

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
<u>high</u>-<u>order</u> or <u>Op</u> <u>Code</u>
portion of the instruc-
tion which is to be
next executed.

4. Clear print area

<u>/</u> 3 0 0

<u>/</u> 2 9 9

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

5. Clear a storage area

<u>/</u> 6 7 5

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

6. Clear word marks from positions 875 and 1225

<u>Ⅱ</u> 8 7 5	S 2 5

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

26

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. ☒ 0 5 1   Would clear what?   44._____

45. / 5 2 0   Would clear which
                   storage areas?   45._____

46. / 3 3 2   Would clear which
                   storage areas?   46._____

47. / 6 0 0   Would clear which
                   storage areas?   47.____ _____

48. / 4 9 9 6 2 0
                   (a) Would clear
                       what storage?   48.(a)_____

                   (b) Would branch to
                       where?   (b)_____

49. ☒ 0 0 1 0 2 1 Would clear what?   49._____

50. / J 9 9   Would clear which
                   storage areas?   50._____

51. / 2 0 0   Would clear which
                   storage areas?   51._____

52. / 6 2 0 4 5 5
                   (a) Would clear
                       what storage
                       areas?   52.(a)_____

                   (b) Would branch to
                       where?   (b)_____

27

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 left-
                                 ward to the hundreds position
46.   300-332   which is also erased.

47.   600 only

48.(a) 600-620   The Clear Storage and Branch
                                 has the "Clear" address in the
   (b) 499   B position and the "Branch"
                                 address in the A position.

49. W/M's at positions 001
    and 021

50.   2100-2199

51.   200 only

52.(a) 400-455

   (b) 620

28

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 alpha-
bet and special characters.

29

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--

674WRENCH   016005002890

```
674WRENCH      016005002890
  I II
  I I
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
```

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.

START

SET WORD MARKS FOR EACH DATA FIELD

A

READ A CARD INTO THE READ AREA

MOVE PART NO, NO. PRODUCED AND LABOR COST INTO PUNCH AREA

PUNCH A CARD FOR ACCOUNTANT

MOVE PART NAME AND NO PRODUC. TO PRINT AREA

PRINT A LINE FOR MGMT. REPORT

CLEAR THE PUNCH AREA

CLEAR THE PRINT AREA

BRANCH TO PROCESS NEXT CARD
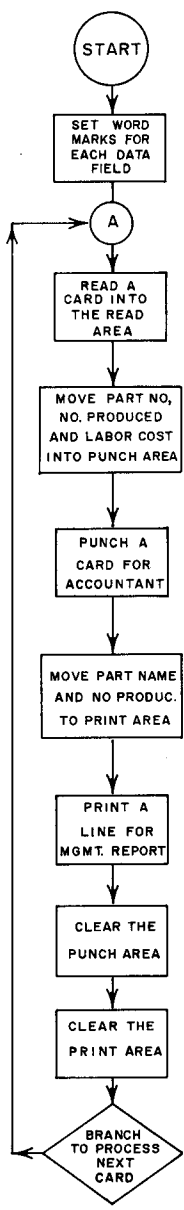
The word marks are placed at the begin-ning of the program and, since they are not destroyed, need not be placed again.

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 ad-dress tells where the instruction word is located in memory)

INSTRUC. ADDRESS	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 in-struction 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	INSTRUC. ADDRESS	Op	A	B
5- 8	Insurance Deduction	5 0 0			
9-12	FICA "				
13-16	Withholding Tax				
17-21	Total Deductions				

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 for-
get to set word marks in the read
area before reading.

54. Input a card with the following data:

Card Col.	Description	INSTRUC. ADDRESS	Op	A	B
1-19	Salesman's Name	5 0 0			
20-22	Salesman's Number				
25-39	City				
40-44	Total Sales				

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.

33

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 4	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 posi-
tion 300. If it is necessary to clear this position,
a "Clear" instruction should precede the final instruc-
tion.

/	3 0 0

MAGNETIC CORE STORAGE: The IBM 1401 uses <u>magnetic</u> <u>core</u>
storage for internally storing data and instruction. A <u>mag-</u>
<u>netic</u> <u>core</u> 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 com-
posed of eight magnetic cores. A code made up of various
combinations of these eight cores in "on" or "off" condi-
tions tells the computer which character is stored at that
position.

A storage position may be figuratively represented as
follows:

```
W/M O
 C  O
 B  O  }
 A  O  } ZONE BITS
 8  O  ]
 4  O  } NUMERIC BITS
 2  O  }
 I  O  ]
```
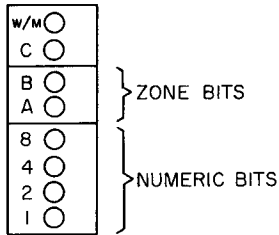
The top <u>core</u>, or <u>bit</u>, is "on" if this character has a W/M
associated with it. The C bit is used only by the computer
as an internal accuracy check. The A and B bits are known
as <u>zone</u> bits and the lower four bits are known as <u>numeric</u>
bits. The numeric bits, singly or in combination, 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."

```
   1        2        3        4        5
 O        O        O        O        O
 O        O        O        O        O
 O        O        O        O        O
 O        O        O        O        O
8 O      8 O      8 O      8 O      8 O
4 O      4 O      4 O      4 ●      4 ●
2 O      2 ●      2 ●      2 O      2 O
I ●      I O      I ●      I O      I ●

   6        7        8        9        0
 O        O        O        O        O
 O        O        O        O        O
 O        O        O        O        O
 O        O        O        O        O
8 O      8 O      8 ●      8 ●      8 ●
4 ●      4 ●      4 O      4 O      4 O
2 ●      2 ●      2 O      2 O      2 ●
I O      I ●      I O      I ●      I O
```

35
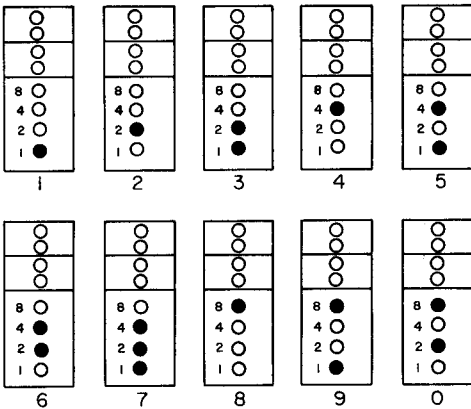
Alphabetic Characters:

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 com-
binations of the zone and numeric bits.
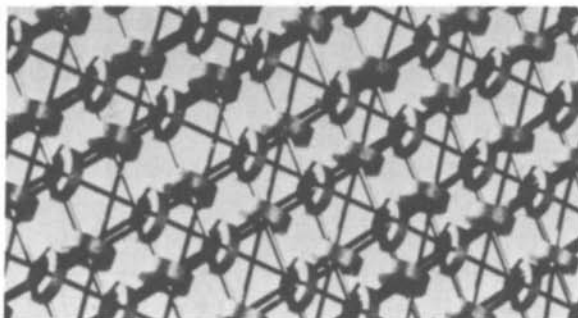
MAGNETIC CORE PLANE

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

D	E	C	2	5	X	M	A	S
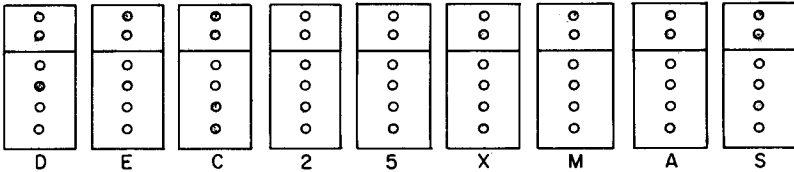
56. Read the following storage units and place the letter
     or number each represents in the box below the figure.
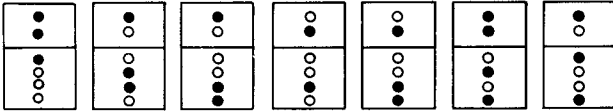
(a) ☐ ☐ ☐ ☐ ☐ ☐ ☐
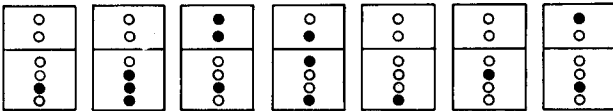
(b) ☐ ☐ ☐ ☐ ☐ ☐ ☐

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._____

37

NOTES AND ANSWERS

55.

D E C 2 5 X M A S
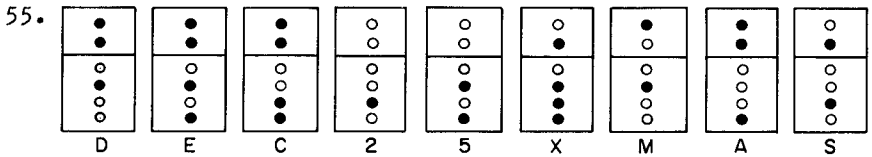
56.(a) H O L S T E R

   (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 charac-
ters may appear in instruction
addresses as a means of desig-
nating 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 | Before: A B C (with A label above, 001 003 below)

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

Before: H O L D (with A label above, 119 123 below)

B
J 1 2 3 (232 235)

61. After:

(a)   A
   [_____] 001 003

(b)   B
   [_____] 232 235
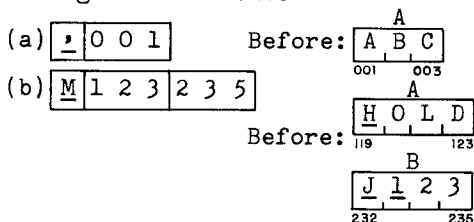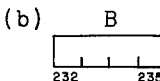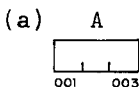
62. Data is already in the Read area. Show the instructions that would be used to punch all of this data on a card.

62.

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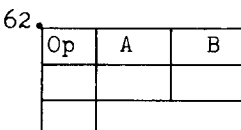
63. Show the instructions that will cause the computer to execute the following:

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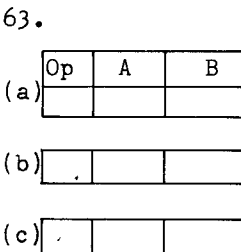
64. In the following symbolic representations of storage units, darken the bits to spell out: JUNE71962

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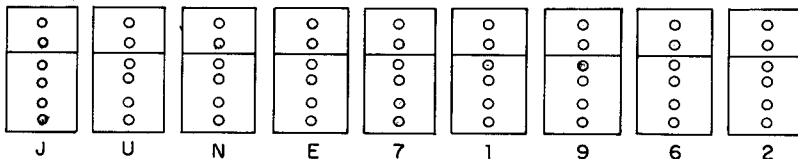
65. If both A and B zone bits were added to the 7, 1, 9, 6, 2 above, what letters of the alphabet would they represent?

7 = [ ]   1 = [ ]   9 = [ ]   6 = [ ]   2 = [ ]

39

ANSWERS   REMINDERS

60.(a) 001 - 080   Refer to page 15

(b) 101 - 180

(c) 201 - 300

61.(a) [A B C]   Refer to page 17
         (A)

(b) [J O L D]
     (B)

62. [M 0 8 0 1 8 0]   Refer to page 21
    [4]

63.(a) [／ 6 5 0]   Refer to page 25

(b) [Д 0 2 3]

(c) [／ 6 2 0 4 2 9]

64.

      J   U   N   E   7   1   9   6   2

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

                    Refer to page 35

40

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
Z̲	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
D̲	X X X	X X X

This instruction will move a <u>single</u> 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
Y̲	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
L̲	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) | Z | 0 0 5 | 0 5 5 |
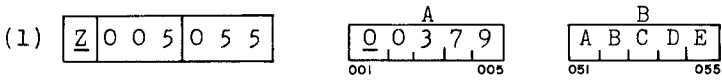
Move the five digits at 001-005 to 051-055 and suppress insignificant zeros.

After

(2) | D | 0 2 2 | 0 2 9 |
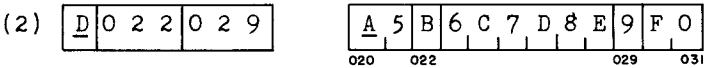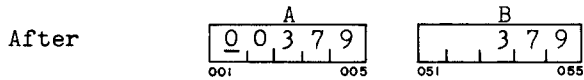
Move the <u>numeric</u> portion of the character at 022 to position 029.
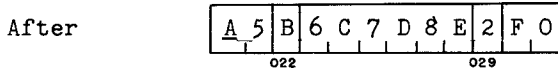
After

Note that storage at 022 and 029 looks like this to start:
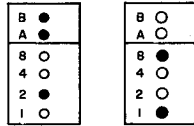
After the instruction is executed, 029 looks like this:

(3) | Y | 0 2 2 | 0 2 9 |
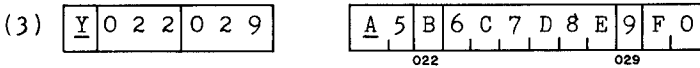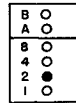
Move the zone portion of the character at 022 to position 029.
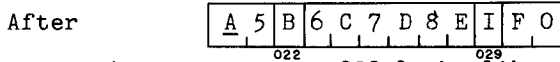
After

After execution, storage at 029 looks like this:
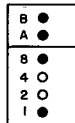
A (9) with A and B zoning becomes the letter (I).

42

EXAMPLES cont'd:

(4) | L | 0 0 5 | 0 5 5 | | 0̲ 0 3 7 9 | | A B̲ C D̲ E |
                               001   005   051   055
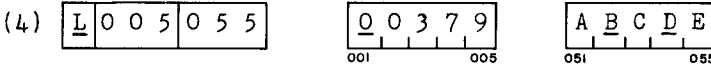
Move the five digits and W/M at 001-005 to 051-055 and
clear the word marks presently at 052 and 054.

After | 0̲ 0 3 7 9 | | 0̲ 0 3 7 9 |
                  001   005   051   055
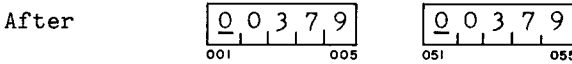
PROBLEMS:

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

    | Q̲ 0 0 4 9 2 0 |
     624   630
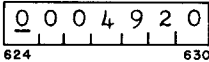
    We want to print the significant
    digits only (i. e., suppressing the
    insignificant zeros) in print posi-
    tions 87-90. Show the MOVE & ZERO
    SUPPRESS instruction to accomplish
    this.

66.
Op	A	B

67. The letter "K" is located at posi-
    tion 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. | S̲ P E E D Y | | 1̲ 2 3̲ 4 A̲ B |
      100   105   200   205
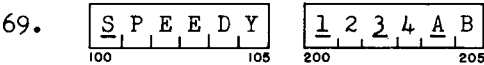
    Show how 200-205 would look after
    execution of the LOAD instruction:

    | L 1 0 5 2 0 5 |

69.
200   205

43

```
        ┌─────────────────┐
        │    ,  , 4 ,9 ,2 ,0│
        └─────────────────┘
         284           290
```
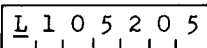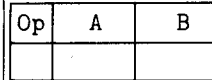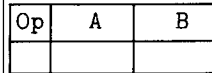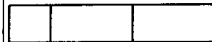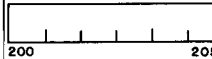
66.

```
┌───┬─────┬─────┐
│Op │  A  │  B  │
├───┼─────┼─────┤
│ Z │6 3 0│2 9 0│
└───┴─────┴─────┘
```
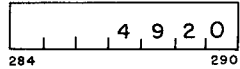
Note that positions 284-286
will be filled with blanks re-
gardless of what was previous-
ly stored there. *

67.

```
┌───┬─────┬─────┐
│ D │4 0 0│5 0 0│
└───┴─────┴─────┘
```

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

68.

```
┌───┬─────┬─────┐
│ Y │4 0 0│5 0 0│
└───┴─────┴─────┘
```

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.

* CAUTION: The MOVE & ZERO SUPPRESS instruction moves only
the <u>numeric</u> portion of the characters in the A-field.
Alphabetic or sign-bits will not be moved. Furthermore,
W/M's in the B-field will be 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 cor-
rect 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 com-
parisons, 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 com-
parison 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 in-
struction 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 in-
struction 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 re-
sults 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.

       <u>d</u>   <u>Branch to I-address if</u>:

       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:

                                                           Triggers

(1) (a)

Op	A	B
C	0 0 5	0 5 5

A-Field   B-Field
[X Y Z]   [X Y Z]
003 005 053 055

    ○ ≠
    ● =
    ○ ≤
    ○ >
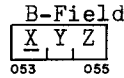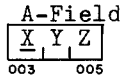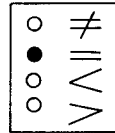
B is equal to A

(b)

Op	A	B
C	0 1 0	5 1 0

A-Field   B-Field
[D O E]   [R O E]
007 010 507 510

    ● ≠
    ○ =
    ○ ≤
    ● >
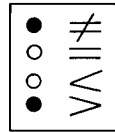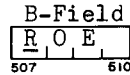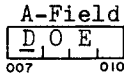
B is not equal to A

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 in-
    struction located at 604 rather than the instruction
    at 504 which would normally follow.

EXAMPLES cont'd:

                                                      Triggers

	INSTRUC. ADDRESS	Op	I	d
(3) (a)	5 0 0	B	6 0 4	S

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

                                                      Triggers

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

         Since the trigger representing "great-
         er than" is off, the next instruction
         would be taken from the next instruc-
         tion word in sequence which is 505.

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

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

	INSTRUC. ADDRESS	Op	I	B	d	B
(b)	5 0 0	V	6 0 4	9 2 1	1	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 posi-
         tion 508--the next word in sequence.

48

PROBLEMS:

                        A-Field   B-Field
70. | C | 0 0 4 | 5 0 4 | | 1 2 6 4 | | 0 2 4 7 | 70.
                              001   004 501   504

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

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

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

     | INSTRUC. ADDRESS | Op | I | B | d |
     | 6 2 0 | B | 7 2 8 | 5 0 9 | X | | 7 |
                                             509

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

     | INSTRUC. ADDRESS | Op | I | B | d |
     | 6 2 0 | V | 7 2 8 | 5 0 9 | B | | A |

74. Write an instruction which would cause
     the program to branch to 455 under the
     following conditions: 74.
                                                        | Op | A | d |
     (a) B > A   (a)

     (b) B ≠ A   (b)

     (c) B < A   (c)

70.

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

71.

Op	A			d
B	7	2	5	S

72.

6	2	8

Since the d-character is <u>not</u> 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:

```
                    ┌────────┐
                   (  START   )
                    └────┬───┘
                         │
                  ┌──────▼──────┐
                  │    CLEAR    │
                  │   STORAGE   │
                  │  SET W/M'S  │
                  └──────┬──────┘
                         │◄─────────────────┐
                  ┌──────▼──────┐            │
                  │    READ     │            │
                  │   A CARD    │            │
                  └──────┬──────┘            │
                         │                   │
                      ┌──▼──┐                │
                    ╱  TOTAL  ╲   NO         │
                   ╱  = OR <   ╲────────────►│
                   ╲   4800?   ╱             │
                    ╲   ┬   ╱                │
                     ╲  │  ╱                 │
                       │ YES                 │
                  ┌────▼────┐                │
                  │  MOVE   │                │
                  │  DATA   │                │
                  └────┬────┘                │
                       │                     │
                  ┌────▼────┐                │
                  │  PUNCH  │                │
                  │ A CARD  │────────────────┘
                  └─────────┘
```

51
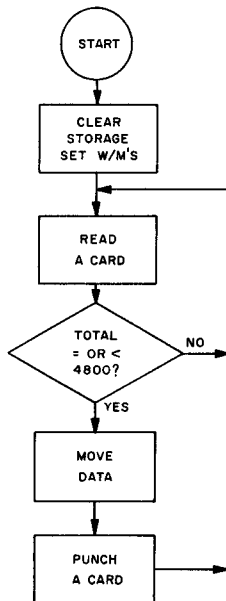
CODING

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

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

52

PROBLEMS:

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

<u>Inventory Card</u>:   <u>Columns</u>

    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.

<u>FLOW CHART</u>   <u>CODING</u>

Instruc. Address	Op	A	B	d

Start
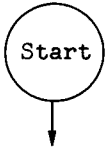
NOTES AND ANSWERS

75.

FLOW CHART

Start

CLEAR
STORAGE
SET W/M'S

READ
A CARD

QUANTITY
=0?

CODE
=X?

NO

NO

YES

YES

MOVE
STOCK NO.

PRINT
A LINE

CLEAR
PRINT
AREA
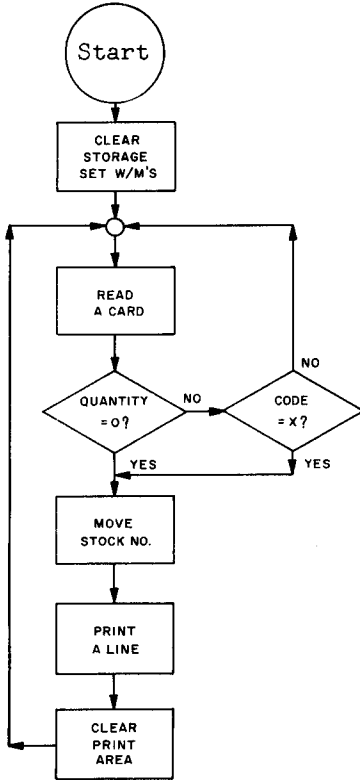
CODING

Instruc. Address	Op	A	B	d
5 0 0	/	0 8 0		
5 0 4	/	2 9 9		
5 0 8	/	3 0 0		
5 1 2	,	0 0 1	0 1 1	
5 1 9	,	0 2 0		
5 2 3	1			
5 2 4	C	0 1 5	5 6 4	
5 3 1	B	5 4 8		S
5 3 6	B	5 4 8	0 2 0	X
5 4 4	B	5 2 3		
5 4 8	M	0 1 0	2 2 0	
5 5 5	2			
5 5 6	B	5 2 3		
5 6 0	0	0 0 0 0		

54

76. Instruction:

| Z | 0 2 4 | 3 6 9 |

Before

A
| Q 3 3 3 3 |
020   024

B
| 6 6 6 6 6 |
365   369

76.   After

B
[diagram]
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
[diagram]
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
[diagram]
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"?

[diagram with symbols: ≠, =, ≤]

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?

_____

55

ANSWERS   REMINDERS

76.   After   Refer to page 41

```
        B
 | 6 | 3 | 3 | 3 | 3 |
  365           369
```

77.   After   Refer to page 41

```
        B
 | G | A | T |
  463     465
```

78.   After   Refer to page 41

```
        B
 | H | O | L | D |
  201       204
```

79.   After   Refer to page 45

```
 | ● ≠ |
 | ○ = |
 | ● < |
 | ○ > |
```

80. Next instruction   Refer to page 46
     in sequence.

81. 600   Refer to page 47

ADDITION: The IBM 1401 performs addition according to alge-
braic rules. According to these rules, every numeric field
is considered to have a <u>sign</u>. 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 differ-
ence 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

<u>Overflow</u> is a special problem in computer arithmetic pro-
cesses. 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 charac-
ters.) If the sum of A + B results in more digits than the
B-field can contain, <u>overflow</u> 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.
<u>Overflow</u> 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: Before Execution | After Execution

Op	A	B
A̲	0 7 0	1 7 0

(1)

A-Field: 0̲ 0 1 4 9 (066 ... 070)
B-Field: 0̲ 0 2 0 0 (166 ... 170)

After: B-Field: 0̲ 0 3 4 9 (166 ... 170)
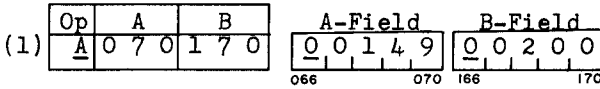
Since no B-bit is present over either
low-order digit, both fields are plus
and the result is plus.

Op	A	B
A̲	0 5 4	3 6 4

(2)

A-Field: 5̲ 6 9 (052 ... 054)
B-Field: 7̲ 0 2 (362 ... 364)

After: B-Field: S̲ 7 1 (362 ... 364)
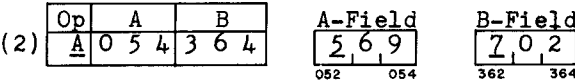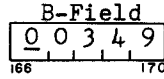
Since the result (1271) contains more
digits than the B-field, overflow
occurs; an "A"-bit is stored over the
high-order position and the overflow
indicator is set. Setting the B-field
to four or more positions would have
avoided overflow.

Overflow
trigger is
on

Op	A	B
A̲	3 0 4	5 0 4

(3)

A-Field: 0̲ 1 4 K (301 ... 304)
B-Field: 0̲ 4 6 7 (501 ... 504)

After: B-Field: 0̲ 3 2 5 (501 ... 504)
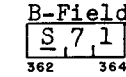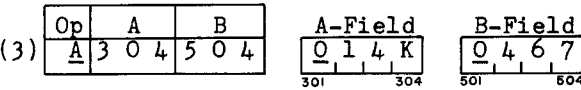
The numeric 2 with a "B"-bit in the
zone bits appears in memory as a K.
Adding the -142 (denoted as minus by
the "B"-bit) to +467 results in a +325.

Op	A	B
?̲	0 7 0	1 7 0

(4)

A-Field: 0̲ 4 9 (068 ... 070)
B-Field: 0̲ 0 2 0 0 (166 ... 170)

After: B-Field: 0̲ 0 0 4 9 (166 ... 170)
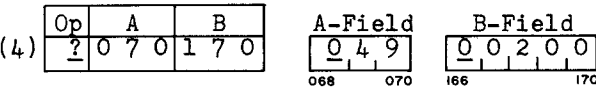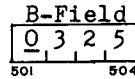
The B-field is filled with zeros be-
fore adding A.

Op	A
A̲	1 4 3

(5)

A-Field: 0̲ 2 2 2 (140 ... 143)

After: A-Field: 0̲ 4 4 4 (140 ... 143)

The A-field is added to itself.

Op	A	B
A̲	0 2 4	2 2 4

(6)

A-Field: 0̲ 2 2 B (021 ... 024)
B-Field: 0̲ 2 2 S (221 ... 224)

After: B-Field: 0̲ 4 4 U (221 ... 224)
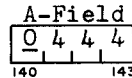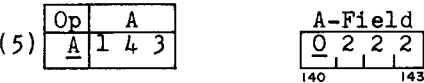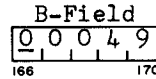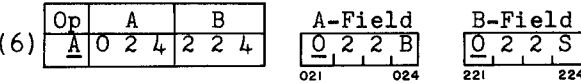
The alphabetic B is a 2 with both A and B bits, there-
fore +. The S is a 2 with just the A bit, therefore
also +. When the two figures are added together, the
result in the B-field will contain the zone-bits of the
original B-field, therefore U.

59

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			Print Positions	
Employee No.	1- 5		Employee No.	11-15
Employee Name	6-24		Employee Name	41-59
Net Earnings	71-77		Total Earnings:	
			Supervisors	75-81

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

Non-
Supervisors   90-96

FLOW CHART
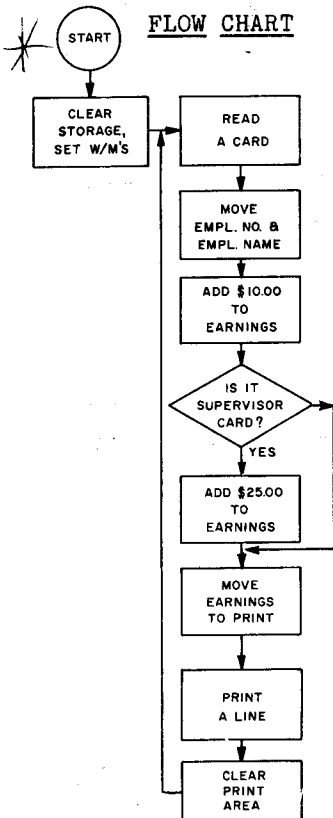
CODING

Instruc. Address	Op	A	B	d
5 0 0	/	0 8 0		
5 0 4	/	2 9 9		
5 0 8	/	3 0 0		
5 1 2	,	0 0 1	0 0 6	
5 1 9	,	0 7 1	0 8 0	
5 2 6	1			
5 2 7	M	0 0 5	2 1 5	
5 3 4	M	0 2 4	2 5 9	
5 4 1	A	5 9 3	0 7 7	
5 4 8	V	5 6 7	0 8 0	K
5 5 6	M	0 7 7	2 9 6	,
5 6 3	B	5 8 1		
5 6 7	A	5 9 7	0 7 7	
5 7 4	M	0 7 7	2 8 1	
5 8 1	2			
5 8 2	/	5 2 6	2 9 9	
5 9 0	1	0 0 0		
5 9 4	2	5 0 0		

*Data stored in 590-593 and
594-597 are constants used
as $10.00 and $25.00 addi-
tives.

60

PROBLEMS:

82. Indicate the result of each of the   82.
    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

	B-Field
(a)	—
(b)	—
(c)	—
(d)	—

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

	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)	

	B-Field
(a)	— (181 184)
(b)	— (441 444)
	A-Field
(c)	— (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		Output Card	
Name	1-19	Name	1-19
Salary	20-24	Tot. Pay	45-49
Commis.	25-29		
O'time	30-34		

FLOW CHART

(START)

↓

CODING

Instruc. Address	Op	A	B	d
5 0 0				

61

ANSWERS

82.(a) [1 7 6]　　　This is a "true add"

(b) [7 1 0]　　　This is a "complement add"--the result
　　　　　　　　　　　is the difference between the two
(c) [4 4 L]　　　numbers and takes the sign of the
　　　　　　　　　　　larger.

(d) [T 7 1]　　　This is an <u>overflow</u> causing an A-bit to
　　　　　　　　　　　be placed in the zone bits of the high-
　　　　　　　　　　　order digit.

83.(a) [1 1 1 1]
　　　　181　　　184

(b) [0 0 1 6]　　The zero and add instruction can be used
　　　441　　　444　　to move a number to a larger field and in-
(c) [2 9 6]　　　sure that the high-order digits are set
　　　150　　152　　　to zero.
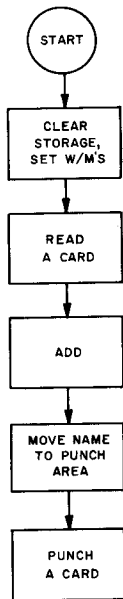
84.　　　FLOW CHART

START

CLEAR
STORAGE,
SET W/M'S

READ
A CARD

ADD

MOVE NAME
TO PUNCH
AREA

PUNCH
A CARD

CODING

Instruc. Address	Op	A	B	d
5 0 0	/	0 8 0		
5 0 4	/	1 8 0		
5 0 8	,	0 0 1	0 2 0	
5 1 5	,	0 2 5	0 3 0	
5 2 2	,	1 4 5		
5 2 5	1			
* 5 2 6	?	0 2 4	1 4 9	
5 3 3	A	0 2 9	1 4 9	
5 4 0	A	0 3 4	1 4 9	
5 4 7	M	0 1 9	1 1 9	
5 5 4	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 re-
sult contains more digits than the B-field, <u>overflow</u> 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 ex-
cept 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

```
     ┌──┬─────┬─────┐    A-Field    B-Field   |   B-Field
(1)  │Op│  A  │  B  │   ┌─┬─┬─┐    ┌─┬─┬─┐    |  ┌─┬─┬─┐
     │S̲│0 5 4│3 6 4│   │5̲│6│9│    │7̲│0│2│    |  │1̲│3│3│
     └──┴─────┴─────┘   └─┴─┴─┘    └─┴─┴─┘    |  └─┴─┴─┘
                        052   054  362   364  |  362   364
```

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

```
     ┌──┬─────┬─────┐    A-Field    B-Field   |   B-Field
(2)  │Op│  A  │  B  │   ┌─┬─┬─┬─┐  ┌─┬─┬─┬─┐  |  ┌─┬─┬─┬─┐
     │S̲│3 0 4│5 0 4│   │0̲│1│4│K│  │0̲│4│6│7│  |  │0̲│6│0│9│
     └──┴─────┴─────┘   └─┴─┴─┴─┘  └─┴─┴─┴─┘  |  └─┴─┴─┴─┘
                        301    304 501    504 |  501    504
```

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

```
     ┌──┬─────┐          A-Field               |   A-Field
(3)  │Op│  A  │         ┌─┬─┬─┬─┐               |  ┌─┬─┬─┬─┐
     │S̲│1 4 0│         │0̲│2│2│0│               |  │0̲│0│0│0│
     └──┴─────┘         └─┴─┴─┴─┘               |  └─┴─┴─┴─┘
                        137    140             |  137    140
```

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

```
     ┌──┬─────┬─────┐    A-Field    B-Field     |   B-Field
(4)  │Op│  A  │  B  │   ┌─┬─┬─┐  ┌─┬─┬─┬─┬─┐    |  ┌─┬─┬─┬─┬─┐
     │⌶̲│0 7 0│1 7 0│   │0̲│4│9│  │0̲│0│2│0│0│    |  │0̲│0│0│4│R│
     └──┴─────┴─────┘   └─┴─┴─┘  └─┴─┴─┴─┴─┘    |  └─┴─┴─┴─┴─┘
                        068  070 166      170   |  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).

```
     ┌──┬─────┐          A-Field                |   A-Field
(5)  │Op│  A  │         ┌─┬─┬─┐                 |  ┌─┬─┬─┐
     │⌶̲│0 7 0│         │0̲│4│9│                 |  │0̲│4│R│
     └──┴─────┘         └─┴─┴─┘                 |  └─┴─┴─┘
                        068  070                |  068  070
```

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

64

PROBLEMS:

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

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

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

      Op   A   B   B-Field
(a) | S 1 6 5 | 8 1 7 | | 0 0 2 2 | | 0 8 6 Q | (a) | — |
                162   165 814   817   814   817
(b) | ! 0 2 1 | 3 5 7 | | 1 M | | 9 6 2 4 | (b) | — |
                020 021   354   357   354   357
                                    A-Field
(c) | S 1 5 2 | | 1 4 8 | (c) | — |
                  150   152   150   152
                                    A-Field
(d) | ! 1 5 2 | | 1 4 8 | (d) | — |
                  150   152   150   152

87. Read a card containing three numbers (A, B and C). Sub-
    tract 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				

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 0̄ | A minus zero is shown as a zero (0) with a minus sign drawn <u>over</u> it (0̄).

(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

START

CLEAR READ
AND STORAGE
AREAS

SET
WORD MARKS

READ
A CARD

SUBTRACT

STORE
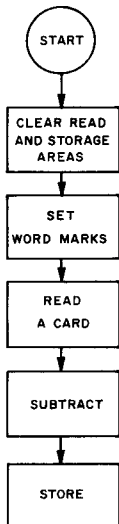
CODING

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	

66

MISCELLANEOUS INSTRUCTIONS:

NO OPERATION:

Op
N

This instruction requires no A, B or d part since it per-
forms 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:

Op
.

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

STOP AND BRANCH:

Op	I
.	X X X

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:

Op		d
K		X

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

Stackers are numbered as follows:

BLANK	PUNCH						READ	CARDS
CARD		Np	4	8/2	I	Nr	BRUSHES	TO
HOPPER		STACKER	STACKER	STACKER	STACKER	STACKER		BE
		NORMAL	4	8 OR 2	I	NORMAL		READ
		PUNCH				READ		

67
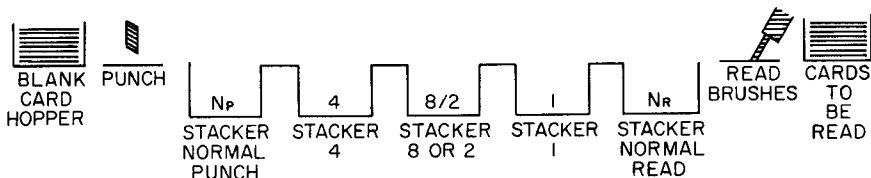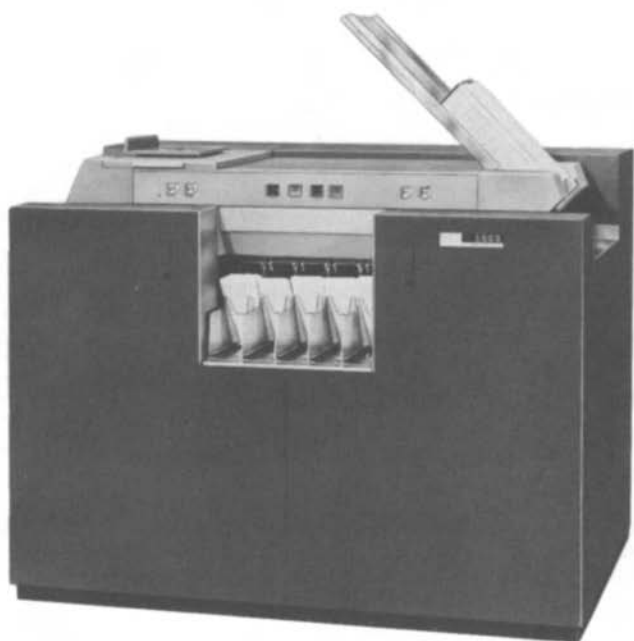
IBM 1402 Card Read Punch

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 <u>after</u>
the READ or PUNCH instruction, or the card will have already
passed the point where it can be directed.

EXAMPLES:

(1)

5 0 0	N	5 4 0	
5 0 4	M	5 3 6	5 0 0
5 1 1			
5 3 6	B	2 4 1	
5 4 0	•		

In this example, the first instruction is a NO OPERA-
TION 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

69

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
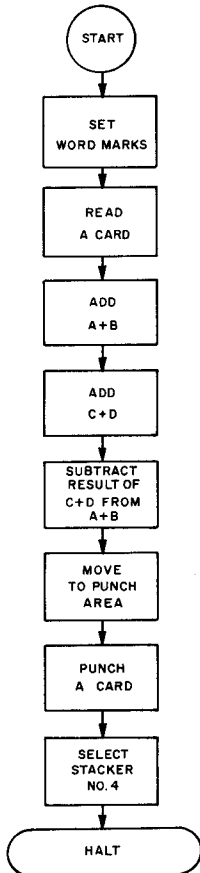
CODING

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

70

PROBLEMS:

Write instructions to perform each of
the following:

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

Op	A	B	d
88.			

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. | | | |
    | | | |
    | | | |
    | | | |
    | | | |
    | | | |

71

NOTES AND ANSWERS

88.

Op	A	B	d
•̲	6 4 4		

89.

N̲	2 0 0	2 8 0	

Simply by moving an M̲ to the opera-
tion code this instruction will
change from NO OPERATION to MOVE.

90.

1̲			
K̲			2

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

91.

4̲			

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

92.

1̲			

No Stacker Select needed (as in
Punch above).

93.

1̲			
K̲			1
A̲	0 0 5	0 1 0	
M̲	0 1 0	1 1 0	
4̲			
K̲			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 <u>last</u> 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 <u>END</u> 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 <u>not</u> 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:

      <u>Type of Information</u>   <u>Print Line Position</u>

         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 insignif-
      icant 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 under-
stand <u>what</u> the program is to do and <u>how</u> 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
	ⴲ	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.

START

01-04	CLEAR THE READ, PUNCH, PRINT AND WORK AREAS
05-08	SET W/M'S IN READ AREA FOR EACH FIELD AND IN WORK AREA FOR SAVING TOTALS
09	READ A CARD TO 001-080

10 — TEST FOR II ZONE OVER 004 — NO / YES

24 — SELECT STACKER NO.1 FOR MANAGER'S CARD

25-26 — ADD $100 TO COMMISSION ON THIS CARD

11 — COMPARE AND BRANCH IF 001-003 = END — = — HALT END-OF-JOB (37) — ≠

14-15 — MOVE ONLY THE NUMERIC PORTION OF SALESMAN NO. TO PUNCH AREA

16 — MOVE NAME TO PUNCH AREA

17 — MOVE COMMISSION TO PUNCH AREA W/OUT INSIGNIF. ZEROS

18 — ADD GROSS SALES TO TOTAL SALES IN WORK AREA

19 — ADD COMMISSION TO TOT. COMMISS. IN WORK AREA

20 — PUNCH THE CARD

21 — TEST FOR II ZONE OVER 004 — NO → (2) → 22-23 CLEAR THE PUNCH AREA → (1)

YES

27 — SELECT STACKER NO.4 FOR MANAGER'S CARD

28 — MOVE THE OFFICE NO. TO PRINT AREA

29 — MOVE TOTAL SALES TO PRINT-WITHOUT INSIG.ZEROS

30 — SUBTRACT TOT. COMMISSIONS FROM TOT. SALES = NET SALES

31 — MOVE NET SALES FROM WORK AREA TO PRINT AREA W/O INSIG. ZEROS

32 — PRINT A LINE

33-34 — CLEAR PRINT AND WORK AREAS

35-36 — SET W/M'S IN WORK AREAS → (2)

76

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.

77

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 (——►(I)—►) 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 be-
cause 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 ac-
complished 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 process-
ing. 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 <u>next</u>
<u>data character</u> to be operated-on by an A-address. The <u>B-</u>
register keeps track of the <u>next</u> <u>data character</u> 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 pro-
cessing 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 eliminat-
ing 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:

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

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

<u>M</u> 015 115

          This is a total of 9 characters, or
<u>M</u>

          a saving of 12 characters of storage.

<u>M</u>

EXAMPLES: In the sample program in Lesson 18, it was nec-
essary to clear the Read, Punch, Print and work areas.
Effectively we cleared from 000-349, using four CLEAR in-
structions 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
—				
—				
—				

94.

Op	A	B	d
/̲	5 2 3		
/̲			
/̲			
/̲			
/̲			

523-500   Start at the highest ad-
dress in the area to be
499-400   cleared. Enter the Op
Code once for each 100
399-300   digits to be cleared.

299-200

199-100

95.

Op	A	B	d
M̲	0 3 0	2 3 0	
M̲			
M̲			

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

96.

Op	A	B	d
A̲	0 1 0	2 5 5	
A̲	0 1 0		
A̲	0 1 0		

251-255   In this case we must re-
peat the A-address each
246-250   time since we want the
same number--not consecu-
241-245   tive numbers.

97.

Op	A	B	d
M̲	0 5 9	2 3 0	
M̲	0 4 9		
M̲	0 2 9		

221-230   This is also partial
chaining as in Problem 96.
211-220   The A-fields were not in
sequence and could not be
201-210   chained.

82

98. Instruction   Before

|A̲|0 0 5|4 2 5|

A
|0̲ 0 2 2 S|
001   005

B
|0̲ 1 2 2 L|
421   425

98.   After

B
| | | |
421   425

99. Instruction   Before

|S̲|0 0 5|4 2 5|

A
|0̲ 0 2 2 S|
001   005

B
|0̲ 1 2 2 L|
421   425

99.   After

B
| | | |
421   425

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

83

ANSWERS   REMINDERS

98.   After   Refer to page 58

B
| <u>O</u> | 1 | O | O | J |
421   425

99.   After   Refer to page 63

B
| <u>O</u> | 1 | 4 | 4 | N |
421   425

100.   Refer to page 67

Op	A	B	d
<u>/</u>	O 8 O		
<u>•</u>	O O 1	1 O 3	
<u>1</u>			
<u>K</u>			2
<u>M</u>	O 7 8	1 8 O	
<u>4</u>			
<u>K</u>			4

101.   <u>24</u>

102.   Refer to page 79

Op	A	B
<u>M</u>	6 3 0	2 1 8
<u>M</u>		
<u>M</u>	6 1 7	

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 pro-
cess 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 con-
verted to machine language by a special assembly program
before it can perform the steps for which it was written:
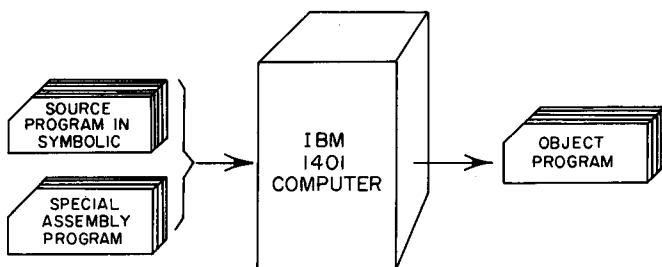
The object program is the program deck with actual machine
language instructions required to do the task of processing
data.

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		M̄	590	690	
PUNCH A CARD	P				4			
NO-OPERATION	NOP				N̄			
ADD*	A	1324	2222		Ā	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		L̄	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	T̄
BRANCH	B	B̄
BRANCH IF WORD MARK OR ZONE	BWZ	V̄
NO-OPERATION	NOP	N̄
COMPARE	C	C̄
HALT	H	•
CLEAR STORAGE	CS	∠
SET WORD MARK	SW	,
CLEAR WORD MARK	CW	H̄
MOVE CHARACTERS TO WORD MARK	MCW	M̄
MOVE CHARACTERS AND SUPPRESS ZEROS	MCS	Z̄
MOVE NUMERICAL	MN	D̄
MOVE ZONE	MZ	Ȳ
LOAD CHARACTERS TO WORD MARK	LCA	L̄
READ A CARD	R	1
PUNCH A CARD	P	4
PRINT A LINE (WRITE)	W	2
STACKER SELECT	SS	K̄

86

PROBLEMS:

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

Mnemonic	A	B	d

104. Write an instruction to <u>ZERO</u> <u>AND</u> <u>ADD</u> a number from
     0160 into 0960.

Mnemonic	A	B	d

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

Mnemonic	A	B	d

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

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 dig-
its. The assembly
program will convert the
2420 to M20.

104.

	A	B	
Z A	0 1 6 0	0 9 6 0	

105.

	A	B	
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 (ob-
ject 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	COUNT	LABEL	OPERATION	(A) OPERAND		CHAR. ADJ.	IND	(B) OPERAND		CHAR. ADJ.	IND	d	COMMENTS
				ADDRESS	±			ADDRESS	±				
3	6 7	8	13 14 16	17	23	27	28		34	36 38	40			88
0 1 0													
0 2 0													
0 3 0													
0 4 0													

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 des-
    criptive 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 in-
    struction 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	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	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	--	--		

91

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 opera-
                       tion codes, but others are abbrevi-
                       ations 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.

LINE	COUNT	LABEL	OPERATION	(A) OPERAND			(B) OPERAND			d	COMMENTS
				ADDRESS	±	CHAR. ADJ.	ADDRESS	±	CHAR. ADJ.	IND.	
0 1 0	0 3	S A L A R Y	D C W	0 6 9 1			1 0 0				A D D F O R M G R S L R Y
0 2 0											

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, <u>provided</u> the count columns indicate the
    correct number of digits.
A word mark will automatically be placed in position 689.

        After assembly: <u>1</u> 0 0
                                  689 690 691

2.

0 4 0	0 3	S A L A R Y	D C W	*			1 0 0				A D D F O R M G R S L R Y

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.

0 7 0	0 3	S A L A R Y	D C	*			1 0 0				A D D F O R M G R S L R Y

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 in-
struction, either the label or actual address may be used.

LINE	COUNT	LABEL	OPERATION	(A) OPERAND			(B) OPERAND			d	COMMENTS
				ADDRESS	±	CHAR. ADJ.	ADDRESS	±	CHAR. ADJ.	IND.	
0 1 0			A	S A L A R Y			C O M M				A D D 1 0 0 T O C O M M
0 2 0											
0 3 0			A	0 6 9 1			C O M M				A D D 1 0 0 T O C O M M

PROBLEMS:

116. Define a constant with a word mark date SEPb20b1962
      (b=blank). Give it a label <u>DATE</u>, and let the program
      assign the address.

COL.
```
6     8          13 14  16 17              24
┌──┬──────────┬──────┬──────────────┬───────────────────┐
│  │          │      │              │                   │
└──┴──────────┴──────┴──────────────┴───────────────────┘
```
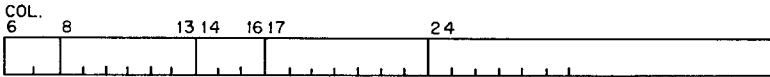
117. Write a mnemonic instruction to move the DATE to 1620.

Mnemonic	A	B	d

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

```
6     8          13 14  16 17              24
┌──┬──────────┬──────┬──────────────┬───────────────────┐
│  │          │      │              │                   │
└──┴──────────┴──────┴──────────────┴───────────────────┘
```
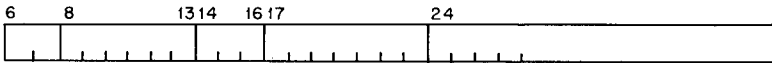
119. Set up a 3-position constant, labelled HOLD, consist-
      ing of 001.

```
6     8          13 14  16 17              24
┌──┬──────────┬──────┬──────────────┬───────────────────┐
│  │          │      │              │                   │
└──┴──────────┴──────┴──────────────┴───────────────────┘
```
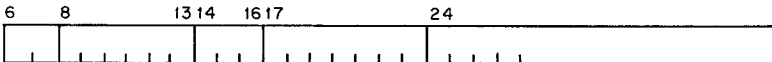
120. Set up a 7-position constant (with W/M) in the follow-
      ing format: $bbb.bb. Call it CONST.

```
6     8          13 14  16 17              24
┌──┬──────────┬──────┬──────────────┬───────────────────┐
│  │          │      │              │                   │
└──┴──────────┴──────┴──────────────┴───────────────────┘
```
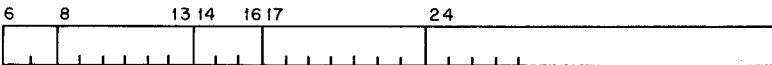
Lesson 22

NOTES AND ANSWERS

116.

	6 8	14	17	24
1 1	D A T E	D C W	*	S E P 2 0 1 9 6 2

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.

	6 8	14	17	24
1 0		D C	0 6 0 9	0 0 0 0 0 0 0 0 0 0

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

119.

	6 8	14	17	24
0 3	H O L D	D C	*	0 0 1

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

120

	6 8	14	17	24
0 7	C O N S T	D C W	*	$.

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 char-
     acters in each case to insure that memory is systemat-
     ically filled.

2.   Sets up a dictionary (table) of labels used with cor-
     responding 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 diction-
     ary 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 precede
     the rest of the program.   If the (ORG) mnemonic is not
     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 must 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 compilation will take place.

97

END (END) This mnemonic operation code must be coded as the
<u>last</u> card in the program. It tells the assembly pro-
gram 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
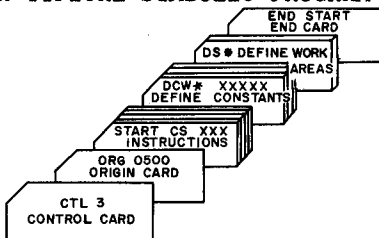
EXAMPLES:

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

Same without giving actual, low-order addresses:

A program starting at storage position O5OO for a computer
with 4OOO memory positions:

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

98

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 loca-
      tion 601 and 4000 storage positions are available.

Mnemonic	A

99

121.   C T L   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
122.   O 3 3 3   origin card will permit the program-
                          mer 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 in-
struction without a label by using Columns 23-26 ($\pm$ 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				(B) OPERAND				d	Object Program			
				ADDRESS	±	CHAR. ADJ.	IND.	ADDRESS	±	CHAR. ADJ.	IND.					
0 1 0		START	CS	0080									333	/	080	
0 2 0			SW	0001				0020					337	,	001	020
0 3 0			R										344	R		
0 4 0			MCW	0010				0210					345	M	010	210
0 5 0			MCW	0024				0244					352	M	024	244
0 6 0			W										359	W		
0 7 0			B	START	+ 011								360	B	344	
0 8 0																

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 charac-
ters 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				(B) OPERAND				d
			ADDRESS	+/−	CHAR. ADJ.	IND	ADDRESS	+/−	CHAR. ADJ.	IND	
	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 <u>COMPARE</u> instruction   START + _____

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

128. Refer to the <u>LAST</u> <u>MOVE</u> instruction   START + _____

129. Refer to the <u>FIRST</u> <u>CLEAR</u> <u>STORAGE</u>
     instruction   START - _____

130. Refer to the <u>BRANCH</u> instruction   START + _____

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 1 9

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			ORG	0500									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			BWZ	STKIN				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			MCW	0030				0128					MOVE SLSMN NAME
180			MCS	0056				0136					COMMISSION
190			A	0046				0349					GROSS SLS TO TOT
200			A	0056				0324					COM TO TOTAL
210			P										PUNCH A CARD
220			BWZ	STKOUT				0004				K	STACK MGRS CARD
230			CS	0180									CLEAR PUNCH AREA
240			B	READ									GO TO READ
250		STKIN	K									1	STACK MGRS CARD
260			A	CONS2				0046					ADD 100 TO MGR
270			B	READ		+009							GO BACK TO PROC
280		STKOUT	K									4	STACK MGRS CARD
290			MCW	0003				0212					MOVE OFF NR
300			MCS	0349				0225					MOVE TOT SALES
310			S	0324				0349					SUB COMMISSION
320			MCS	0349				0235					MOVE NET SALES

330		W					PRINT A LINE
340		CS	0349				CLEAR WORK AREA
350		CS					CHAIN CLEARING
360		SW	0325		0310 .		SET WM IN WA
370		B	STKIN -008				GO BACK CLR PCH
380	STOP	H	STOP				END OF PROGRAM
390	003 CONS1	DCW *		END			CONSTANT
400	003 CONS2	DCW *		100			CONSTANT
410		ENDS TART					END CARD

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.

Lesson 25

NOTES AND ANSWERS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	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.

UNIT V QUIZ

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.

_____

_____

_____

_____

109

UNIT V QUIZ

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

110

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 permit-
ting 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 read-
ing 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 opera-
tions.   As soon as printing is completed, the punch will
punch a card.

WRITE, READ AND PUNCH:

<u>Op</u>   <u>Mnemonic</u>

| 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	O O 8 O			O 2 8 O		
	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	O O 8 O			O 2 8 O		
	W R						
	B	S T A R T	+	O O 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 proc-
esses. 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

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	O O 1 O			O 1 1 O			
	M C W	O O 1 O			O 2 2 O			
	M C W	O O 1 5			O 1 2 5			
	M C W	O O 1 5			O 2 3 O			
	W R P							
	S S	S T A R T	+	O O 1				4

EDITING AND ARRANGING PRINTOUTS: Amounts are usually punch-
ed 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 in-
struction.

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:

$ b , b b 0 . b b & C R

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 abbre-
viation 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.

115

EXAMPLES:

Control Field | $ b , b b 0 . 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 | $ 2 4 . 9 5 | (B-Address after edit)

Or:

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

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

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

Control Word: | $ b b b , b b 0 . b b & * * |

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

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

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

Notice in the example above that the <u>data word</u> is moved on
top of the <u>control word</u>. 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 con-
trol word.

PROBLEMS:

139. Assume the following control field and data field:

         Data Field:
         Control Field: | $ b b b , b 0 . b b 0 & * |
         Data Field: | 0 0 0 0 0 0 0 5 |

            Printed Result: | _ _ _ _ _ _ . _ _ _ |

140. Assume the following control field and data field:

         Data Field:
         Control Field: | * b b , b b 0 & & A S S E T S |
         Data Field: | 0 1 4 7 0 |

            Printed Result: |_ _ _ _ _ _ _ _ _ _ _ |

141. Assume the following control field and data field:
         Data Field:
         Control Field: | $ b b 0 . b b & & W H & T A X |
         Data Field | 0 1 6 7 0 |

            Printed Result: |_ _ _ _ _ _ _ _ _ _ _ |

142. Assume the following control field and data field:

         Control Field: | * * & J U N E & b b & b b |
         Data Field: | 0 7 6 2 |

            Printed Result: |_ _ _ _ _ _ _ _ _ _ |

NOTES AND ANSWERS

139. | $. 0 5   * | By inserting zeros in
                                          place of b's in the con-
                                          trol field--leading zeros
                                          may be retained.

140. | * 1 , 4 7 0   A S S E T S |

141. | $ 1 6 . 7 0   W H T A X |

142. | * * J U N E 0 7 6 2 |

PRINTER CONTROLS: When printing out reports, it is neces-
sary 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 con-
trolled.
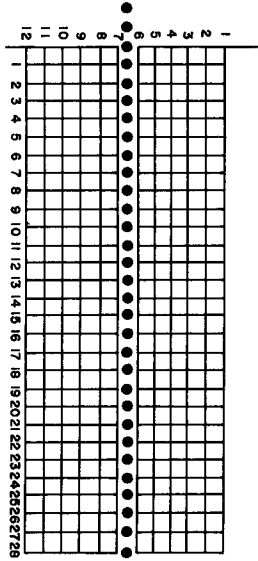
The tape has 12 _channels_ or columns marked across the top,
and usually 140 _line_ _numbers_, corresponding to print lines on
a page, marked lengthwise. Punching a rectangular hole in
a particular channel, at a specific line number, permits the
programmer to direct a control instruction to the printer
referencing the _channel_ _punch_.

CONTROL CARRIAGE:

Op		Mnemonic		d
F		C C		X

Only the operation and digit modifier is used.

119

By choosing a d-modifier from the following table, the car-
riage 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</u> <u>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) <u>immediately</u>

           = K   2

           = L   3

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

           = S   2

           = T   3

EXAMPLES:

1. Skip to Channel 4 immediately

Label	Oper.	A-Operand				B-Operand				d
		Address	±	Ch.Adj	N/B	Address	±	Ch.Adj	N/B	
	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 car-
riage 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

GL●UE

143.(a)

(b)

(c)

(d)

PLACE CUTOFF EDGE OF TAPE HERE

Mnemonic	d

144.(a)

(b)

(c)

(d)

144. Write a control carriage
      instruction to skip to each of
      the four punches immediately.

145. Write a control carriage in-
      struction to skip 3 lines after
      printing and branch to 0540.

Mnemonic	A-Address	d

145.

	Channel	Line
143.(a)	1	4
(b)	5	1 0
(c)	7	1 9
(d)	1 2	2 7

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

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

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

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

The control carriage instruc-
tion 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
    W R P 0 4 0 0   meaning of
                     this instruc-
                     tion?

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:

125

ANSWERS   REMINDERS

146.

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

Refer to pages 111 and
121

147. Write, read, punch and   Refer to page 112
      branch to location 400.

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 repet-
itive 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   or   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 pro-
vide 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 |X₁X₁X₁0₁0₁0₁0₁0|
           Multiplier
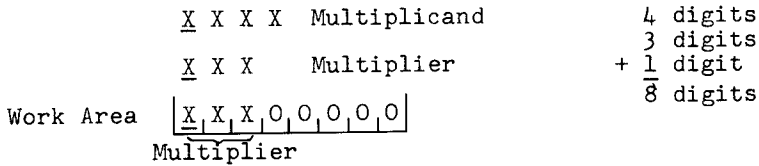
The multiplier in the work area is destroyed as the product
is formed.

If either field is negative, the minus sign must be indi-
cated 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.

| 1,2,2,2 | | 4,7,4 |
  045   048   157 159

              | 0,0,0,0,0,0,0,0 | Work Area
                201   208

| Mnemonic | A-Address | B-Address |
| L C A | 0 1 5 9 | 0 2 0 3 | Move multiplier to high-order
                                        digits of work area

              | 4,7,4,0,0,0,0,0 | Work Area
                201   208

| M | 0 0 4 8 | 0 2 0 8 | Multiply--addressing the
                                        whole work area

              | 0,0,5,7,9,2,2,8 | Work Area with product
                201   208
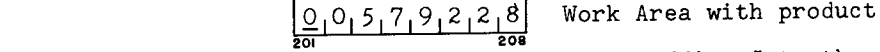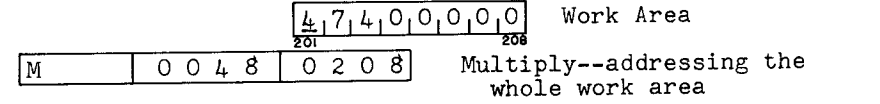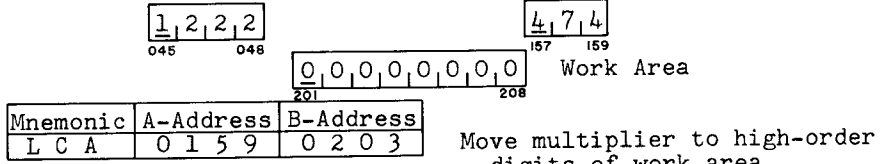
Half-adjusting, or rounding, may be done by adding 5 to the
highest digit to be cut-off. To reduce an 8-digit product
to 6 digits and round requires the following steps:
                                        ┌HIGHEST DIGIT
| 5 | | 0,5,6,7,8,6,5,4 | Product
                         201   207 208

| A | 0 2 0 0 | 0 2 0 7 | Add 5 to Position 0207

              | 0,5,6,7,8,7,0,4 | After rounding
                201   206 208

| L C A | 0 2 0 6 | X X X X | Move 6 digits to where wanted
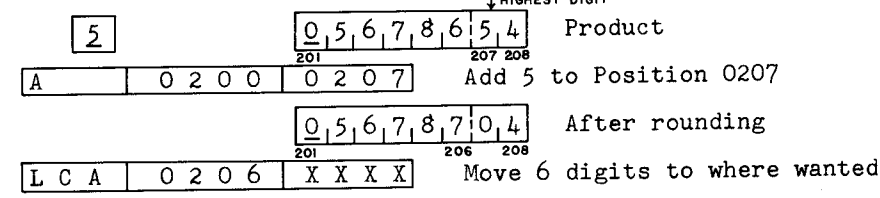
To adjust for decimal points, count off from the right digit
a number of digits equal to the total number of decimal
digits in the multiplier and multiplicand.

128

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
0 9	P R O D	D C W	*
0 9	P R O D	D C W	0 5 0 1

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 $\underline{X}_|X_|X_|X_|X$

          Multiplier in location MULT $\underline{X}_|X_|X$

          Product area set up in location
                                   PROD $\underline{-}_|\ _|\ _|\ _|\ _|\ _|\ _|\ _|$

     Instruction
  Op   A   B   Product Area
| Z A | M U L T | P R O D - 6 | $\underline{X}_|X_|X_|\ _|\ _|\ _|\ _|\ _|$ | Multiplier
                                                                              set up in
                               −6 −5 −4 −3 −2 −1 PROD   high-order
                                                                              positions of
| M | H O L D | P R O D | product area

                               $\underline{0}_|X_|X_|X_|X_|X_|X_|X_|X$ Product
                                                                       result
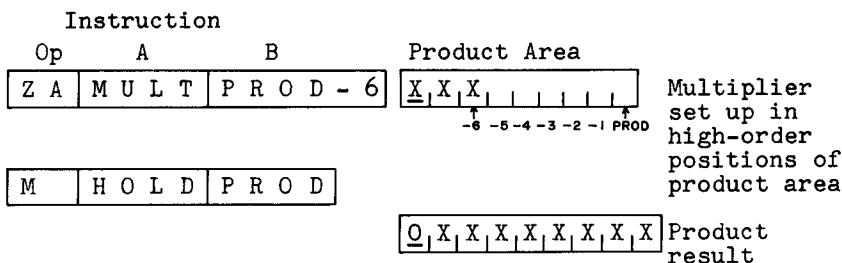
No space is provided in memory for decimal points. It is up
to the programmer to keep track of its position.

EXAMPLE: Multiplicand   X X X . X X

                      Multiplier   X X . X X

Product will have four decimal places.

129

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 ac-
complished, 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, sup-
pressing 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
0 7	P R O D	D C W	*		Set up product area
0 1	F I V E	D C W	* 5		Set up const. of 5
1 0	E D I T	D C W	* $ b b b . b b		Set up edit field
		L C A	R A T E	P R O D -4	
		M	H R W K	P R O D	
		A	F I V E	P R O D	
		M Z	P R O D	P R O D -1	
		L C A	E D I T	0 2 7 5	
		M C E	P R O D -1	0 2 7 5	
		W			

PROBLEMS:

Assume multiplication of the following amounts:

$142.41   shown in memory | 1 | 4 | 2 | 4 | 1 |
                                200   204
        by

$ 3.50   shown in memory as | 3 | 5 | 0 |
                                  205   207

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 = | 0 | 4 | 9 | B = | 1 | 5 | 7 | N |
      Indicate the result of A x B,
      showing sign and size of
      product.   156. | | | | |

157. A = | 2 | J | B = | 0 | 5 | R |
      Indicate the result of A x B,
      showing sign and size of
      product.   157. | | | | |

131

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. Round-
                               ing-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 subtrac-
tion. The number of times the divisor can be subtracted
from the dividend is the quotient.

Thus: 24 ÷ 8 = ?

     24 - 8 = 16   16 - 8 = 8   8 - 8 = 0

     (1 time)   (2 times)   (3 times)

                3 is the quotient

or: 25 ÷ 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 per-
form 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.

                            B-Address

       Work Area | 0 | 0 | 0 | 0 | 0 | X | X |
                            Divisor
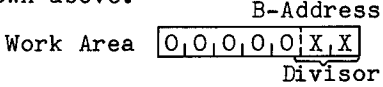
After execution of the division the quotient will be found
in the <u>high-order</u> digits of the work area--the remainder
will be found in the low-order digits. The number of digits
in the quotient will be the same as the number of digits in
the dividend. The number of digits in the remainder will
equal the number in the divisor.

Work Area $\boxed{X_{|}X_{|}X_{|}X_{|} \quad _{|}X_{|}X}$
Quotient   Remainder
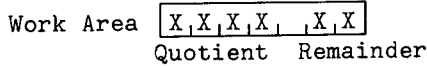
The sign of both fields must be designated by zoning. If
the field is plus it <u>must</u> have both A and B bits over the
low-order digit. If the field is minus it <u>must</u> have a B-bit
without an A-bit over the low-order digit.

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 remain-
> der will be signed plus.

> If both fields are minus, the quotient will be <u>plus</u>
> 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.
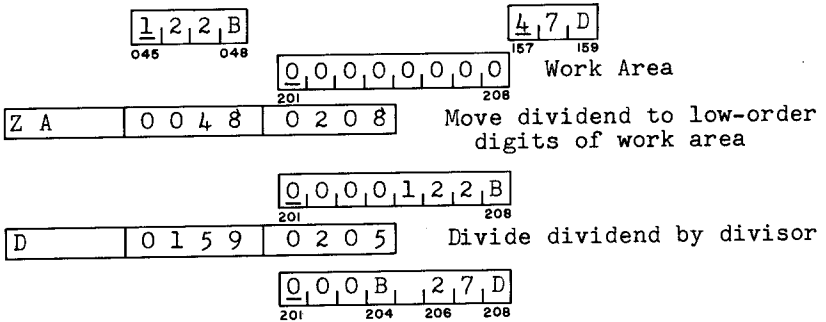
Note that all fields have a plus sign as shown by presence
of the A and B bits over the low-order digits.

134

PROBLEMS:

Assume division of the following fields:

     1428 as | 1 , 4 , 2 , H | Dividend
              200   203

       31 as | 3 , A | Divisor
                204 205

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. $A = | 0 , 1 , 5 |$ $B = | 0 , 1 , 5 , P |$

     Indicate the result of $B \div A$,
     showing the sign and size of
     both quotient and remainder. 162. [, , , ,]

163. $A = | 0 , 2 , 0 , N |$ $B = | 0 , 1 , K |$

     Indicate the result of $A \div B$,
     showing the sign and size of
     both quotient and remainder. 163. [, , , ,]

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. | 0̲ 0 4 6   0 2 |

162. | 0̲ 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. | 0̲ 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 <u>Name</u> into the print area and branch back to the main program.

TABLE

Memory Address	Dept. Nr.	Dept. Name
5 0 0	0 1	A C C O U N T G
5 1 0	0 2	F O U N D R Y
5 2 0	0 3	M I L L I N G
5 3 0	0 4	P A T T E R N
5 4 0	0 5	P E R S O N N L
5 5 0	0 6	P U R C H A S G
5 6 0	0 7	S A L E S
5 7 0	0 8	S T O R A G E
5 8 0	0 9	T R A F F I C

Input Card

Dept.
Number

| ? ? |
| 001 002 |

Counter

| 9 |
| 601 |

Constants

| 0 1 0 | 5 0 1 | 5 0 9 |
| 602   604 605   607 608   610 |

SUB-ROUTINE:

Instruc. Addr.	Mne-monic	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	O	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.

137

OPTIONAL FEATURE: The IBM 1401 may be equipped with a fea-
ture 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 con-
tents 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 <u>middle</u> 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:

Special Constant

Instruc. Addr.	Mne-monic	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

* These instructions are Indexed.

There is no need to initialize the COMPARE and MCW instruc-
tions. 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 modifi-
cation 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:

| 0 | 0 | 0 | 2 | 0 | 1 | 0 | Constants
|---|---|---|---|---|---|---|
| 300 | | 302 | 303 | 304 | 305 | 306 |

Label	Mne-monic	A-Addr.	Ind	B-Addr.	Ind	d

Flow chart:

START

LOAD ZEROS INTO IR1

ZERO AND ADD 20 INTO IR2

MOVE FIRST WORD- MODIFIED BY IR1

SUBTRACT 1 FROM IR2

DOES IR2=0? — NO → ADD 10 TO IR1

YES

HALT
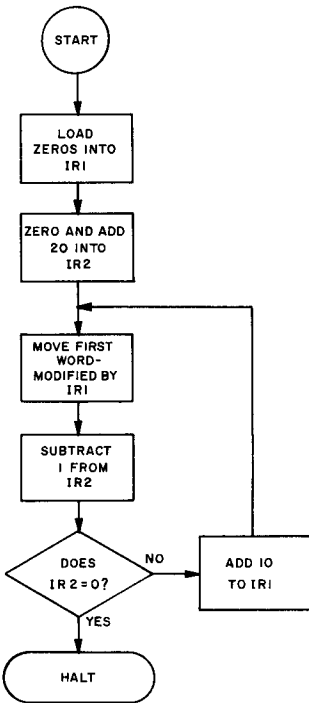
NOTES AND ANSWERS

164.

Label	Mne-monic	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 *dividend* Divisor in work area.

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

167.

(a) | _____ digits

(b) |_I_I_I_I _____

(c) |_I_I_I_I _____

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) _____ _____

141

ANSWERS   REMINDERS

165. (a) | F | Refer to page 127
     (b) | F |
     (c) | Ø |

166. (a) PROD - | 6 | Refer to page 129
     (b) PROD - | 1 |

167. (a) | 10 digits | Refer to page 134

     (b) |_____X_X_X |

     (c) | X_X_X_X_X_X   X X X |
             Quotient   Remain.

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_   Refer to page 138

     (b) _097 - 099_

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
$\frac{1}{2}$ 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.
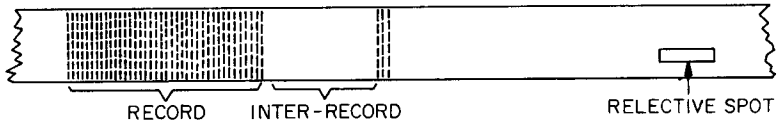
RECORD   INTER-RECORD   RELECTIVE SPOT
                 GAP
Bits appear on tape as follows: O   CHECK BIT
                                       O } ZONE BITS
                                       O
                                       O
                                       O } NUMERIC BITS
                                       O
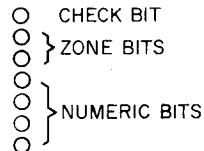
The check-bit has no bearing on data in tape storage so may
be ignored. The bits are counted as "on" if they appear on
the tape; "off" if the space is not magnetized.

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 com-
puter 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 <u>file</u>. A <u>file</u> may be stored on one <u>tape reel</u>
(2400 feet) or more than one if it is a very large file.

Every tape reel should have three <u>labels</u>. One label is a
sticker attached to the reel showing the name of the file.
When we refer to a <u>label</u> in this text we will refer to the
second type. A <u>header label</u> 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 <u>before</u> he begins to write data records.

Each tape reel should also include a <u>trailer</u> label. The
<u>trailer</u> label is written at the end of the tape after all
data records are on the tape. It should include such in-
formation 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 <u>reflective spot</u> or
<u>load point</u>. A second <u>reflective spot</u> 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 <u>tape mark</u> to be
written after the trailer. A <u>tape mark</u> is a one-character
record, written as any other record, by the program. As the
tape is read, the program continually tests for the <u>tape
mark</u>.

                            145

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 head-
      er 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 repre-
      sents 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 reflec-
      tive spots on the tape?   181._____

182. What is the function of the tape mark?

      _____.

183. What is the function of a group mark?

      _____.

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 charac-
                       ter 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 recorded data has been reached.

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

WRITING ON MAGNETIC TAPE: Data is written on tape, charac-
ter-by-character, from the address given by the program
until a <u>group</u> <u>mark</u> 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 <u>backspace</u> (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 <u>left-most</u> digit of the record
to be written. The d-modifier (W) indicates a WRITE opera-
tion. 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 dis-
cussed. This instruction operates like <u>WRITE TAPE</u> except
that when a W/M is encountered it is converted to a word-
separator character and placed ahead of its associated char-
acter.

WRITE TAPE MARK:

Op	Mnemonic	A-Address	d
<u>U</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>U</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 $3\frac{1}{2}$ 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 indi-
cator 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

1.   1 7 6 4 W R E N C H 0 3 7 5 0 0 1 5 1 4
        0500   0520

Write tape | M C W | % U 1 | 0 5 0 0 | W |

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

PREVIOUS
THIS RECORD   IRG   RECORD

1764WRENCH0376001514   1763PLIERS04

THE CHARACTERS ACTUALLY APPEAR AS
MAGNETIC BITS AND IN REVERSE ORDER.
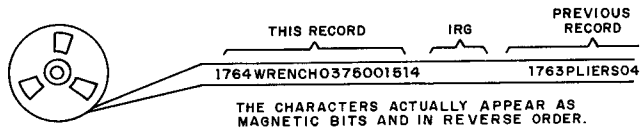
2. Same record

Write tape
with W/M's | L C A | % U 1 | 0 5 0 0 | W |

⚡1764⚡ WRENCH ⚡0375⚡0015⚡14

WORD SEPARATORS PLACED AHEAD OF
EACH CHARACTER WITH A WORD-MARK
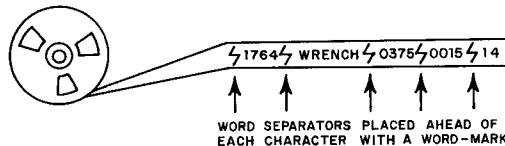
PROBLEMS:

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

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

186. Show an instruction to
     branch to a routine
     labelled <u>ERROR</u> if the
     error indicator is on.

187. Show an instruction to
     branch to a routine
     labelled <u>EOR</u> if the end-
     of-reel indicator is on.

188. Show an instruction to
     write a tape mark on
     Tape Unit #5.

189. Show an instruction to
     SKIP AND BLANK $3\frac{1}{2}$ inches
     of tape on Unit #2.

184.

MNE—MONIC	A	B	d

185.

186.

187.

188.

189.

NOTES AND ANSWERS

	Mnemonic	A-Address	B-Address	d
184.	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

The d-modifier tells which indicator to test. If the indicator is not "on" no branch

187.	B	E O R		K

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 charac-
ter-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 en-
countered. All characters are actually read twice and com-
pared. 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 indi-
cates 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		Op Code
C U	% U n	B		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		Op Code
C U	% U n	R		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:
                                           Op
                  Mnemonic A-Address d   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 can-
not 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 con-
ditions.

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 instruc-
     tion to backspace Tape
     Unit #6.

192.

193. Write a control instruc-
     tion to rewind Tape Unit
     #4.

193.

194. Write a control instruc-
     tion to rewind and unload
     Tape Unit #5.

194.

NOTES AND ANSWERS

	Mnemonic	A-Address	B-Address	d
190.	M C W	% U 2	O 3 O O	R

The MCW instruction is normally used for reading.

191.	L C A	% U 3	O 3 O O	R

The LCA is used when the data on tape was written with an LCA.

192.	C U	% U 6		B

All control instruc-tions 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. What-
ever 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 avail-
able, 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

```
  ┌──────┐                                      ┌──────┐
  │ TAPE │                                      │ TAPE │
  │ UNIT │─────┐              ┌────────►        │ UNIT │
  │  #1  │     │     ┌──────────────┐           │  #3  │
  └──────┘     └────►│              │           └──────┘
                     │   IBM 1401   │
  ┌──────┐     ┌────►│              │────┐      ┌──────┐
  │ TAPE │     │     └──────────────┘    │      │ TAPE │
  │ UNIT │─────┘                         └─────►│ UNIT │
  │  #2  │                                      │  #4  │
  └──────┘                                      └──────┘
```
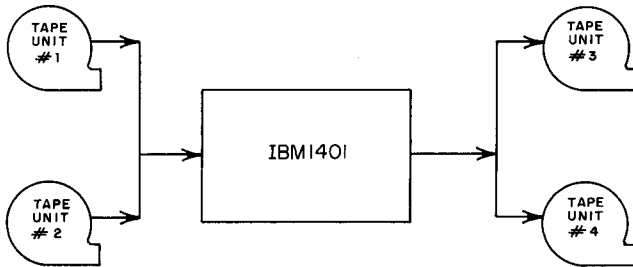
When Reel #1 is finished on Tape Unit #1, modify the A-
address of the read instruction to %U2 on which Reel #2 is
loaded. Processing can continue while the operator takes
Reel #1 off Tape Unit #1 and mounts Reel #3. When Reel #2
is exhausted, the A-address is changed back to %U1. The
same can be done 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	Mne-monic	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	R	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   195. _____
      must be modified for flip-flop?

196. How many tape units are required to   196. _____
      flip-flop the <u>read</u> operation?

197. Which branch instructions must always   197.
      follow a read or write?   (a) _____

                                                    (b) _____

198. Write a control instruction to skip   198.
      and erase $3\frac{1}{2}$ inches of tape on Tape

      Unit 1.

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_

MNE-MONIC	A	d
C U	% U 1	E

198.

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

(Label = WRITE) (ENTER)   Flow Chart

SET ERROR
COUNTER
TO 3

WRITE A
RECORD ON
%un

"WRITE" ERROR ROUTINE

WRITE
ERROR?   YES   BACKSPACE
ONE
RECORD   SUBTRACT
ONE FROM
ERROR COUNTER   ERROR
COUNTER
= 0 ?   NO

NO   YES

SKIP–ERASE
FORWARD

"EOR" ROUTINE

END
OF REEL ?
(EOR)   YES   WRITE
TAPE MARK

NO

1

WRITE
RECORD COUNT
THEN SET IT
TO ZERO

ADD I TO
OUTPUT
RECORD COUNT

WRITE
TWO TAPE
MARKS

EXIT

(LABEL = WEXIT)

REWIND
AND
UNLOAD

FLIP–FLOP
TAPE UNITS

WRITE
LABEL ON
NEW TAPE

1

WRITE SUB-ROUTINE
CODING

COUNT	LABEL	OPERATION	(A) ADDRESS	±	CHAR. ADJ.	IND.	(B) ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
	WRITE	ZA	CON3				WCTR					SET COUNTER TO 3
		MCW	%U3				0400					WRITE A RECORD
		B	WERR									BRANCH IF ERROR
		B	WEOR									BRANCH IF EOR
		A	CON1				WRCDCT	-	001			ADD 1 TO REC CT
	WEXIT	B	XXXX									BRANCH BK TO PGM
	WERR	CU	%U3									BACKSPC A RECORD
		S	CON1				WCTR					REDUCE COUNTER
		C	WCTR				CONZER					COUNTER = 0?
		B	WSKIP									STO SKP ERASE IF=
		B	WRITE	+	007							TO REPEAT WRITE
	WEOR	CU	%U3									WRITE TAPE MARK
		MCW	%U3				WRCDCT	-	006			WRITE RECORD CT
		ZA	CONZER				WRCDCT	-	001			ZERO THE REC CT
		CU	%U3									WRITE TAPE MARK
		GU	%U3									WRITE TAPE MARK
		CU	%U3									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

(Label = <u>READ</u>)

163

READ SUB-ROUTINE
CODING

COUNT	LABEL	OPERATION	(A) ADDRESS	±	CHAR. ADJ.	IND.	(B) ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
	READ	ZA	CON3				RCTR					SET COUNTER TO 3
		MCW	%U1				0400					R READ A RECORD
		B	RERR									L BRANCH IF ERROR
		B	REOR									K BRANCH IF EOR
		A	CON1				RRCDCT	-	001			ADD 1 TO REC CT
	REXIT	B	XXXX									BRANCH BK TO PGM
	RERR	CU	%U1									B BACKSPC A RECORD
		S	CON1				RCTR					REDUCE COUNTER
		C	RCTR				CONZER					COUNTER = 0?
		B	EHLTI									S BR TO HALT IF =
		B	READ	+	007							GO TO REPEAT RD
	REOR	MCW	%U1				0400					R READ RECORD CT
		C	0405				RRCDCT	-	001			RECORD COUNTS =
		B	RCOK									S IF = GO AHEAD
	EHLTZ	H										IF NOT HALT
	RCOK	CU	%U1									U
		B	EOJ				INDIC					1 BRANCH IF EOJ
		MN	READ	+	010		RREG	-	001			IF NOT EOJ FLP TP
		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									S IF = 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									M WR OTPT TAPE MRK
		MCW	%U3				WRCDCT	-	006			W WRITE RECORD CT
		CU	%U3									M WR TWO TAPE MRKS
		CU	%U3									M
		CU	%U3									U REWIND & UN-LOAD
		H										END OF JOB
1	RCTR	DCW	*		0							
7	RRCDCT	DCW	*		000		000					
1	INDIC	DCW	*		0							
2	RREG	DC	*		02							
20	CONLAB	DCW	*		INVbCTLblb	OFblbY DDD						

164

199.

X X ⁺ X X X X ⁺ X X X ≢ X X X X ⁺ X ≢
501   519

(a) How many tape records will this make?

(b) Which storage positions will cause IRG?

199.

(a) [ ]

(b) _____

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.

200.

(a) [ ]

(b) _____

_____

201. Write the two instructions that should immediately follow a Read instruction.

201.

Op	A	B	d

202. Show an instruction to write on Tape Unit 6 (without W/M's) from storage location 1260.

202.

Op	A	B	d

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.

203.

Op	A	B	d

165

UNIT VIII QUIZ

ANSWERS   REMINDERS

199.(a)   [2]   The group mark (‡) separ-
                           ates records and causes IRG
    (b) 512, 519   on tape.

200.(a)   [3]   The W/M causes word-separa-
                           tors on tape if the Load
                           instruction is given.
    (b) 501, 504, 509.   The writing would stop at
                           the first Group Mark (512).

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-separa-
tors 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.

U N I T   I X

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 fea-
tures 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 access-
    ible 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 char-
    acters.   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 tempo-
    rary 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 <u>left</u> 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 con-
     sole may be turned on or off by the operator. Any one
     switch can be tested by a BRANCH IF INDICATOR ON in-
     struction. 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 de-
     tail. Following the rules for coding discussed in this
     text should result in average efficiency. As the pro-
     grammer 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.

UNIT IX   Lesson 38

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 sys-
tem 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 cor-
rect 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 com-
puter 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.

169

Testing, or "debugging," is a necessary step. After a pro-
gram 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 cor-
rected 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 pro-
gram. In any event, the program should not be used to proc-
ess "live" data until all errors are corrected.

The programmer will want to make use of several "general-
ized" 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 posi-
tions to blank before loading the program.

Another standard IBM program is the MEMORY PRINT or MEMORY
DUMP program. This program provides a printed output show-
ing the contents of every position of memory. When a pro-
gram 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

171

program by a branch instruction or by a <u>calling sequence</u>.
A <u>calling sequence</u> 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 <u>MOVE</u> instruction moves the address of the next instruc-
tion in sequence to the final branch of the sub-routine.

QUICK REFERENCE
of All Instructions Covered in the Text

Refer to Pg.	Op Code	Mne- monic	A	B	d	Descriptive Title

I.W/M, MOVE and LOAD OPERATIONS:

Refer to Pg.	Op Code	Mne- monic	A	B	d	Descriptive Title
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:

Refer to Pg.	Op Code	Mne- monic	A	B	d	Descriptive Title
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:

Refer to Pg.	Op Code	Mne- monic	A	B	d	Descriptive Title
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)

174

Refer to Pg.	Op Code	Mne- monic	A	B	d	Descriptive Title

VI.**MAGNETIC TAPE OPERATIONS**:

Refer to Pg.	Op Code	Mne- monic	A	B	d	Descriptive Title
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.

175

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	1 0 0 0	1 0 0 B	
205.	A	1 0 0 S	2 0 0 M	
206.	A	1 0 0	1 1 6 6 6	
207.	S	1 0 0 0 B	1 0 0 0 K	
208.	S	1 1 1 1 A	1 1 1 1 C	
209.	S	1 1 1 1 C	1 1 1 1 A	
210.	ZA	2 2 2	3 3 3 3 3	

	Mnemonic	Contents of A-Field	Contents of B-Field	Resultant A-Field
211.	S	0 1 1 1 M	No B-Field	
212.	ZA	0 1 1 1 D	No B-Field	
213.	ZS	0 1 1 1 M	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

A	B	S	T	5	5
021   026

B-Field

S	I	J	K	2	6
154   159
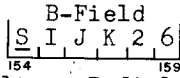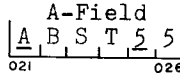
show the contents of the resultant B-field:

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

| 1 | 2 | 2 | 2 | 1 | Hold
|---|---|---|---|---|

B-Field

| 1 | 2 | 2 | 2 | 2 | Area
|---|---|---|---|---|
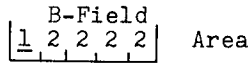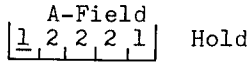
where will the program go for its next instruction?

Answer:_____.

216. After executing the following step,

Mnemonic	A	B	d
B	S U B R	0 0 7 2	H

B-Field

H	O	L	D
071   074
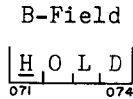
where will the program go for its next instruction?

Answer:_____.

217. After executing the following instruction,

Mnemonic	A	B	d
M C S	0 0 2 9	0 5 5 1	

A-Field

0	0	0	2
026   029

B-Field

0	1	0	2	2	4
546   551

show the contents of the resultant B-field:

546   551

177
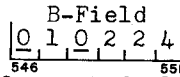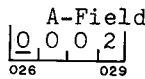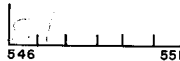
218.

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
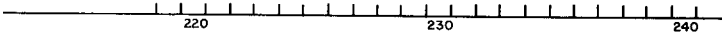
223. Write a program, in symbolic, to solve the following
     problem:

     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 pro-
     gram interlock.

	Op	A	B	d
R				
	L C A	H O L D	A R E	
?				
?				

179

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

A-Field
2 2 3 3 L
023   027

B-Field
2 2 3 3 C
286   290

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

A-Field
2 2 2 2
023   027

B-Field
0 2 2 2
286   290

where will the program go for its next instruction?

Answer:_____.

ANSWERS TO QUIZ PROBLEMS

204. $\boxed{\underline{2}\,_0\,0\,_0\,B}$ (Pg. 58)

205. $\boxed{\underline{1}\,_0\,0\,_0\,K}$ (Pg. 58)

206. $\boxed{\underline{1}\,_1\,7\,_6\,6}$ (Pg. 58)

207. $\boxed{\underline{2}\,_0\,0\,_0\,M}$ (Pg. 63)

208. $\boxed{\underline{0}\,_0\,0\,_0\,B}$ (Pg. 63)

209. $\boxed{\underline{0}\,_0\,0\,_0\,K}$ (Pg. 63)

210. $\boxed{\underline{0}\,_0\,2\,_2\,B}$ (Pg. 58)

211. $\boxed{\underline{0}\,_0\,0\,_0\,\bar{0}}$ (Pg. 63)

212. $\boxed{\underline{0}\,1\,_1\,1\,D}$ (Pg. 58)

213. $\boxed{\underline{0}\,1\,_1\,1\,D}$ (Pg. 63)

214. $\boxed{\underline{S}\,I\,N\,K\,S\,6}$ (Pg. 41)
      154   159

215. location 750   (Pg. 46

216. next instruction in sequence
           (Pg. 46)

217. $\boxed{\underline{0}\ 1 \qquad\quad 2}$ (Pg. 41)
    546   551

218. $\boxed{9\ 0\ 1}\boxed{B}\boxed{9\ 9\ 1}$ (Pg. 69)

219.

Op	A	B	d
$\underline{A}$	0 5 3 8	0 8 2 3	
$\underline{A}$	0 5 3 3		
$\underline{A}$			
$\underline{A}$	0 5 2 4		

                     (Pg. 79)

220.

Ct	Label	Op	A
19	CONST	DCW	* TOT&COST&&$

                          24   34
                     (Pg. 93)
  35   42

221. (a) $\boxed{K}$

     (b) $\boxed{SW}$

     (c) $\boxed{B}$

     (d) $\boxed{CS}$

                  (Pg. 103)

ANSWERS TO QUIZ cont'd

222. (a)

Op	A	B	d
L C A	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)

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 avail-
able 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 deduct-
ed 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 check-
ed in the next day's processing before new orders are proc-
essed. If partial stocks are available, the partial "pick-
ing 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 re-
ceipt," 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 rec-
ords are necessary:

1. Inventory Ledger File: This file will contain one rec-
    ord for each type of item. The records will be se-
    quenced 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   Once for
                         each   150
Total Ordered-6 digits   branch

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 ware-
    house 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 not
    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)

```
       4      10  13  16   20               35   39 4243      51    56   60  64
      ┌─────────────────────────────────────────────────────────────────────┐
      ┤                                                                       ├
      ┤     (BRANCH                                                           ├
      ┤        NO.)                                                           ├
LINE 5┤     XXXX           XXXXXXXXXXXXXXXXXXXX (BRANCH NAME)                  ├
    6 ┤                    XXX——————————XX (BRANCH STRET ADDRESS)             ├
    7 ┤                    XXX—————————XX (BRANCH CITY)                        ├
      ┤                                                                       ├
      ┤                                                                       ├
   12 ┤  XXXXXXXXXX XXXXX————————XX      XX      XX,XXX    X,XXX               ├
      ┤  (STOCK NO.) (   ITEM NAME   )  (W'HSE  (QUANTITY)(WEIGHT)            ├
      ┤                                 LOC'N.)                               ├
      └─────────────────────────────────────────────────────────────────────┘
```
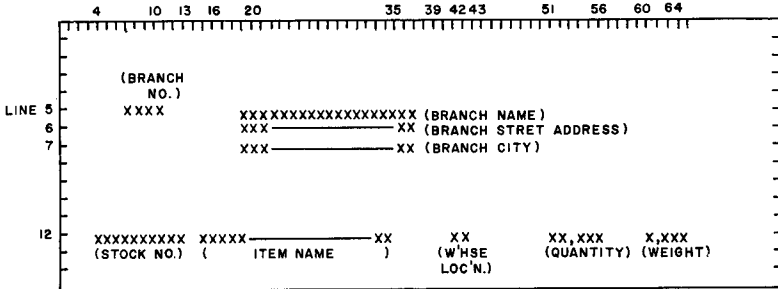
6. Order Record Card: Each time a picking-ticket is print-
   ed a card is punched for use by the shipping department
   to check-out shipments and then is sent to the account-
   ing department.

       Data Field   Card Columns

       Customer Number   .   1- 4
       Stock Number   5-14
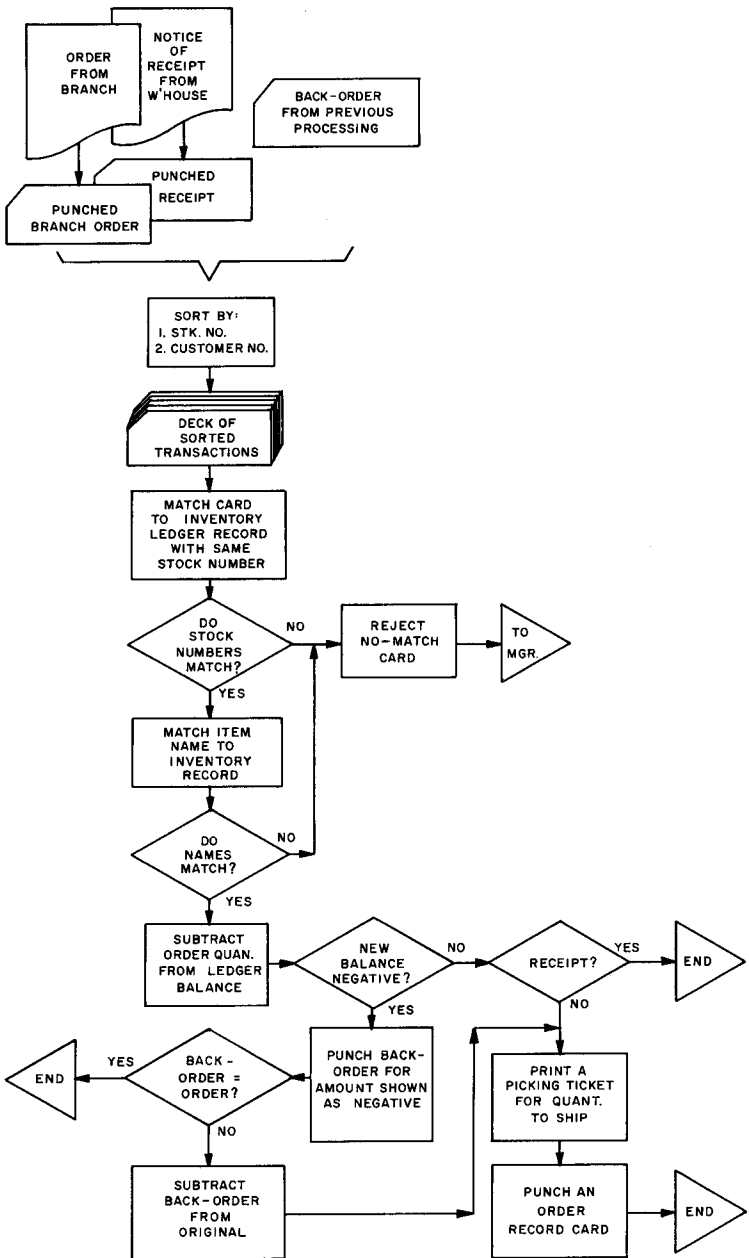   Quantity Shipped   15-20
       Price Each   21-26
       Total Price   27-35
       Blanks   36-39
       Date   40-46

                    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.

185

SYSTEM PROGRAMMING: When the programmer discusses the in-
ventory 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 mag-
    netic 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 col-
    umns. (The receipt card has blanks in the Branch Nr.)

3. Both <u>order</u> <u>record</u> <u>cards</u> and <u>back-order</u> <u>cards</u> 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
    <u>added</u> 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 <u>after</u> the program deck,
    but <u>before</u> 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 rou-
    tines (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 <u>ahead</u> 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
              INVENTORYbLEDGERb1bOFb1bbDDMMYY

    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 <u>detail flow chart</u> 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
0101-0180	Output Card Area		Addresses
0201-0300	Print Area	1501-1708	Tape Read Area
0301-0364	Work Area for	1500	Reserved for "Last
	Table Data		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 <u>LABEL</u> 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:

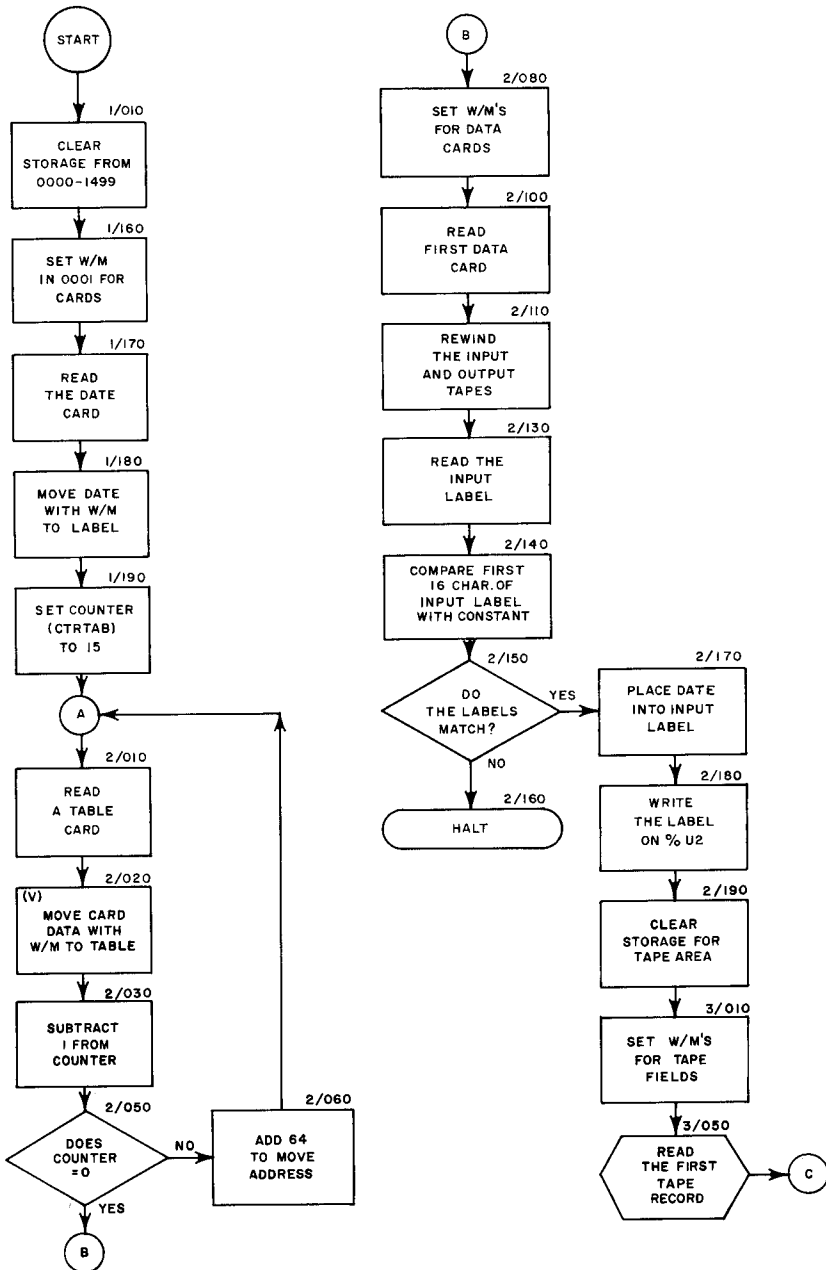| M C W	*	+ 0 1 1	R E X I T	+ 0 0 3
B	R E A D			
B	*	+ 0 0 1		

    This set of instructions will cause the second branch instruction to be inserted at the end of the READ sub-routine. The second branch instruction says to branch to the next instruction in sequence.

189

5.   The last card of the <u>DATA</u> deck will be punched with <u>END</u>
      in columns 1-3. When this card is detected, move an <u>A</u>
      to 1500. At end-of-reel on the input tape test for an
      <u>A</u> 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 on
      page 199. Do <u>not</u> 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 accomplish-
ed programmer, you must work with the machine and with the
problems to be solved by the machine. Nothing can be sub-
stituted 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

START

1/010
CLEAR STORAGE FROM 0000-1499

1/160
SET W/M IN 0001 FOR CARDS

1/170
READ THE DATE CARD

1/180
MOVE DATE WITH W/M TO LABEL

1/190
SET COUNTER (CTRTAB) TO 15

A

2/010
READ A TABLE CARD

2/020
(V) MOVE CARD DATA WITH W/M TO TABLE

2/030
SUBTRACT I FROM COUNTER

2/050
DOES COUNTER = 0

NO → 2/060 ADD 64 TO MOVE ADDRESS

YES → **B**

B

2/080
SET W/M'S FOR DATA CARDS

2/100
READ FIRST DATA CARD

2/110
REWIND THE INPUT AND OUTPUT TAPES

2/130
READ THE INPUT LABEL

2/140
COMPARE FIRST 16 CHAR. OF INPUT LABEL WITH CONSTANT

2/150
DO THE LABELS MATCH?

YES → 2/170 PLACE DATE INTO INPUT LABEL

NO → 2/160 HALT

2/180
WRITE THE LABEL ON % U2

2/190
CLEAR STORAGE FOR TAPE AREA

3/010
SET W/M'S FOR TAPE FIELDS

3/050
READ THE FIRST TAPE RECORD → **C**

191

MAIN PROCESSING

C

3/080 COMPARE CARD STK.NO WITH TAPE STK.NO

3/090 IS CARD STK. NO. HI?

YES → **8/010** WRITE TAPE RECORD ON %U2 → **8/040** READ TAPE RECORD FROM %UI

NO

3/100 CARD STK. NO.= TAPE — **NO** → **3/110** HALT

YES

3/120 CMP. ITEM NAME IN CD WITH MASTER

3/140 DO ITEM NAMES MATCH? — **NO** → HALT

YES

3/150 TEST INPUT CARD FOR BACK-ORDER

3/160 IS IT A BACKORDER? — **YES** → **3/170** SUBTRACT BACKORDER QUANTITY FROM TAPE B/O QUANT

NO

D

3/190 SUBTRACT CARD QUANT. FROM TAPE O-H

3/200 IS NEW BAL. MINUS? — **YES** → **7/010** MOVE QUANTITY TO PUNCH CD. AND SET TO PLUS → **7/020** MOVE INPUT CARD DATA TO PUNCH → **7/050** PUT A B-ZONE ON COLUMN I → **7/060** PUNCH B/O CARD → **7/080** ADD B/O QUAN.TO TAPE B/O

NO

3/210 IS CARD A RECEIPT? — **YES** → **7/170** READ A CARD

NO

E   **I**

7/090 SET TAPE O/H BAL. TO ZERO

7/100 SUBROUT TO UPDATE BRANCH TOT

7/130 SUBTRACT B/O FROM INPUT QUAN

7/140 IS IT ZERO — **YES** (up) / **NO** → **F**

E

SUBROUT. TO UPDATE BRANCH TOTAL

F

SET UP A COUNTER WITH 15

G

COMPARE INPUT BR.NO.(V) WITH BR.NO IN TABLE

MATCH? — **YES** → **H**

NO

SUBTRACT I FROM COUNTER

DOES COUNTER = 0? — **NO** → ADD 64 TO COMPARE AND MOVE ADDRESS

YES

HALT

192

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

193

Special QUANUP Sub-routine

ENTRY

8/070
SET
COUNTER TO
15

8/080
(V)
COMPARE
BR. NUMBER

8/090
EQUAL ?   YES

8/160
(V)   ADD
INPUT QUANTITY
TO BR. RECORD

8/170
RESTORE
COMPARE AND
ADD ADDRESSES

EXIT

NO
8/100
SUBTRACT
1 FROM
COUNTER

8/110
IS IT
ZERO?   YES

8/200
HALT

NO
8/130
ADD
TO MODIFY
COMPARE AND
ADD ADDRESSES

(Variable addresses are indicated by V)

194

Quick Reference - Operation Codes, 75, 173

Read area, 15
Read sub-routine, 163, 164
Record mark, 146
Records, 144
Reflective spot, 144, 145
Reserved storage areas, 15
Rounding, 128, 130

Sign, 57, 63
Source program, 85
Stacker, 67, 69
Storage function, A
Storage position, 1, 3
Stored Program System, A
Sub-routine, C, 102, 171, 172
Symbolic coding, 85
Symbolic coding sheet, 89
Symbolic program, 85, 98, 169
Symbols for flow charting, C
System analysis, 183-185
System flow chart, 186
System programming, 187, 188

Tape, 144, 157
Tape flip-flop, 157, 158
Tape labels, 145
Tape mark, 145
Tape reel, 145
Testing, 170
Trailer label, 145

Variable word length, 5, 6

Word, 3, 4
Word mark, 5, 6
Word separator, 146, 153
Work areas, 78, 93
Write sub-routine, 161, 162

Zone bits, 35, 38

TABLES

ILLUSTRATIONS

ANSWERS

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	+/-	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	+/-	CHAR. ADJ.	IND.	d	COMMENTS
010		START	C S	1 4 9 9									CLEAR STORAGE BY
020			C S										CHAINING
030			C S										
040			C S										
050			C S										
060			C S										
070			C S										
080			C S										
090			C S										
100			C S										
110			C S										
120			C S										
130			C S										
140			C S										
150			C S										
160			S W										SET WM FOR CARD
170			R										READ DATE CARD
180			L C A	0 0 0 7				L A B E L	-	0 0 1			ST DATE IN LABEL
190			Z A	C O N 1 5				C T R T A B					SET CTR TO 15
200													

200

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
010		BLDTAB	LCA	A0064				0564					READ TABLE CARDS
020			S	CON1				CTRTAB					STORE IN TABLE
030			C	CTRTAB				CONZER					SUB FROM CTR
040			B	TABEND									CTR EQUAL ZERO
050			A	CON64				BLDTAB	+	007		S	IF SO, END OF TAB
060			B	BLDTAB									IF NOT MOD ADDR
070		TABEND	SW	0001				0005					REPEAT CYCLE
080			SW	0015				0035					SET WM FOR CARDS
090			R										
100			CU	%U1				1501				R	READ LST DATA CD
110			CU	%U2								R	REWIND INPUT
120			MCW	%U1								R	REWIND INPUT
130			C	0416				LABEL	-	017		R	READ INPUT LABEL
140			B	*	+	002							CHECK LABEL
150			H									S	GO AHEAD IF EQU
160			MCW	LABEL	-	001		1526					IF NOT-HALT
170			MCW	%U2				1501					UPDATE LABEL
180			CS	1708									PUT LABEL ON OUT
190			CS										CLEAR TAPE AREA
200			CS										
210													

LINE	COUNT	LABEL	OPERATION	(A) ADDRESS	±	CHAR. ADJ.	ON	(B) ADDRESS	±	CHAR. ADJ.	ON	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									UCARD S/N IS HIGH
100			B	*	+	002							SCARD S/N EQ TAPE
110			H										NO MATCH-STOP
120			C	0034				1530					COMP ITEM NAME
130			B	*	+	002							MATCH-GO AHEAD
140			H										NO MATCH-STOP
150			BZW	*	+	009		0001					KIS INPUT BACK-Ø
160			B	SUBQTY				0458					IF NOT GO AHEAD
170			S	0040									SUB
180			B	SUBQTY				1550					SUB CARD QUAN
190		SUBQTY	S	0040				1550					KIS NEW BAL MINUS
200			BWZ	BACKØ				0040					KIS CD A RECEIPT?
210			MCW	READA	+	011		QUEXIT	+	003			SET S.R EXIT
220			B	*									BR TO SUB-ROUT
230			B	QUANUP									RETURN ADDRESS
240			B	PRINT									

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
010		PRINT	ZA	CON15									SET CTR TO 15
020			C	0004									COMP BR NR -TABLE
030			B	FIND								S	OK-NR IS MATCH
040			S	CON1				CTRTAB					SUB 1 FROM CTR
050			C	CTRTAB				CONZER				S	COUNTER EQ ZERO?
060			B	ERRTAB									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
010		FIND	MCW	0564				0364					MOVE ENTRY TO WA
020			SW	0301				0305					SET WMS FOR WORK
030			SW	0325				0345					AREA FOR TABLE
040			MCW	0304				0213					MOVE BR NR TO PR
050			MCW	0324				0239					MOVE BR NAME
060			W										PRINT A LINE
070			CS	0299									CLEAR PRINT AREA
080			MCW	0344				0239					MOVE BR ADDRESS
090			W										PRINT A LINE
200			CS	0299									CLEAR PRINT AREA

203

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
010			M C W	0 3 6 4				0 2 3 9					MOVE BR CITY
020			W										PRINT A LINE
030			C S	0 2 9 9									CLEAR PRINT AREA
040			C C									2	SKIP TO LINE 12
050			M C W	1 5 1 0				0 2 1 3					MOVE STK NR
060			M C W	1 5 3 0				0 2 3 5					MOVE ITEM NAME
070			M C W	1 5 4 2				0 2 4 3					MOVE WAREHSE LOC
080			M C W	E D Q U A N				0 2 5 6					SET EDIT CONST
090			M C E	0 0 4 0				0 2 5 6					EDIT QUANTITY
100			M C E	E D W G T				0 2 6 4					SET EDIT CONST
110			M C E	1 5 4 0				0 2 6 4					EDIT WEIGHT
120			W										PRINT A LINE
130			C C									1	SKIP TO NEXT PGE
140			C S	0 3 6 4									CLEAR TABLE WA
150			C S										CLEAR PRINT AREA
160			M C W	C O N A D 1				P R I N T	+	0 1 3			RESTORE TABLE
170			M C W	C O N A D 2				F I N D	+	0 0 3			RESTORE MOVE
180			M C W	0 0 0 4				0 1 0 4					MOVE BR NR-CARD
190			M C W	1 5 1 0				0 1 1 4					MOVE STK NR
200			M C W	0 0 4 0				0 1 2 0					QUANTY TO CARD

204

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
010			M C W	1 5 3 6				0 1 2 6					PRICE EACH
020			M C W	1 5 3 6				M P Y W A	−	0 0 7			PUT PR IN WORK A
030			M P Y	0 0 4 0				M P Y W A					MULTIPLY
040			S W	M P Y W A	−	0 0 9							SET WM FOR PROD
050			M C W	M P Y W A				0 1 3 5					MOVE PROD TO CD
060			M C W	L A B E L	−	0 0 1		0 1 4 6					MOVE DATE TO CD
070			R P										PUNCH CD AND RD
080			C S	0 1 8 0									CLEAR PUNCH AREA
090			L C A	M P Y C O N				M P Y W A					RESTORE WORK AR
100		E N D A	C	C O N E N D				0 0 0 3					CHECK CD FOR END
110			B	*	+	0 0 6						S	IF END-SKIP
120			B	M A I N									IF NOT REPEAT
130			M C W	C O N A				1 5 0 0					MOVE IND TO RD
140		W R I T X	M C W	*	+	0 1 1		W E X I T	+	0 0 3			WRITE RECORD
150			B	W R I T E									
160			B	R E A D X									
170		R E A D X	M C W	*	+	0 1 1		R E X I T	+	0 0 3			READ RECORD
180			B	R E A D									
190			B	W R I T X									REPEAT UNTIL EOF
200													

205

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
010		BACK Ø	Z S	1 5 5 0				0 1 4 0					SET QUAN TO PLUS
020			M C W	0 0 3 4				0 1 3 4					MOVE INPUT CARD
030			M C W										TO OUTPUT FOR
040			M C W										BACK-ORDER
050			Z S	0 1 0 1									PUT B-ZONE ON 1
060			P										PUNCH BACK-Ø CD
070			S S									4	SEL STACKER NR
080			A	0 1 4 0				1 5 5 8					ADD QUAN TO B/Ø
090			Z A	C O N Z E R				1 5 5 0		'			SET Ø/H TO ZERO
100			M C W	*	+	0 1 1		Q U E X I T	+	0 0 3			GO TO SUB-ROUT
110			B	Q U A N U P									RETURN ADDRESS
120			B	S U B I N									SUB B/Ø FROM INP
130		S U B I N	S	0 1 4 0				0 0 4 0					SEE IF QUAN=ZERO
140			C	0 0 4 0				C O N Z E R					IF SO-READ NEXT
150			B	R E A D A									IF NOT ORDER
160			B	P R I N T									READ NEXT CARD
170		R E A D A	R	E N D A									GO BACK
180			B										
190													
200													

206

LINE	COUNT	LABEL	OPERATION	(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
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	CON15				CTRTAB					SET-UP COUNTER
080			C	0004				1562					COMPARE BR NR
090		QFIND	B	QFIND								S	IF EQUAL-FOUND
100			S	CON1				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

207

LINE	COUNT	LABEL	OPERATION		(A) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	(B) OPERAND ADDRESS	±	CHAR. ADJ.	IND.	d	COMMENTS
010	33	LABEL	DCW	*			INVEN	N	TORY		LEDGER	R		1 OF 1 DDMMYY,#
020	02	CON15	DCW	*			15							
030	02	CTRTAB	DCW	*			00							
040	02	CON64	DCW	*			64							
050	02	CON1	DCW	*			01							
060	08	CONZER	DCW	*			000000		000000000					
070	06	EDQUAN	DCW	*			0		0					
080	05	EDWGT	DCW	*			.							
090	13	MPYWA	DCW	*			000		000000					
100	13	MPYCON	DCW	*			000		000000					
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	0													
190	0													
200	0													

208