IBM®

Programming the IBM 1440 Data Processing System

Student Text

Preface

This book is written for the student who has had some exposure to data processing systems and who is interested in learning to program and operate the IBM 1440 Data Processing System. It is assumed he has some knowledge of computers and programming acquired either by having satisfactorily completed the IBM Basic Computer Systems course or an equivalent course; or through actual but limited experience in programming.

Before studying the material in this book, the student should know the meaning of terms such as: actual codes, symbolic codes, input, output, core storage, address modification, indexing, compiler, block diagramming, source program, object program, record layout, binary-coded decimal – terms defined in the Basic Computer Systems course.

The present book contains Part 1 which through the generous use of examples covers most aspects of programming the 1440 Data Processing System. It is intended as a supplement to classroom instruction in the IBM 1440 Basic Programming Course and its contents therefore closely parallels subjects covered in the 1440 course.

Chiefly, these subjects are the Autocoder (programmer's) language, the techniques of programming, and the system in general. The book will be republished in the future to include Part 2 which will explain those details one must know to operate the computer and to test one's program.

Questions to test comprehension follow each section. Answers to the questions will be found in the Appendix at the back of the book. The student's ability to complete these test questions indicates readiness to proceed to the next section. Upon successfully completing Part 1, the student will be able to write his own programs for the 1440 in Autocoder language.

Each section in this book builds upon the previous section. Therefore the student will find it to his advantage to study the material and to complete the exercises in the order presented. Students with considerable programming experience on other systems may wish to review the questions and problems to determine at which point they will begin their study.

The programming examples represent the author's solutions to the problems. The student may find solutions that differ from the author's. This is possible, for seldom is there *only one* acceptable solution. Look upon the author's solutions simply as a guide.

Additional reference material covering the subjects described in this book may be found in the Systems Reference Library for the IBM *1440 Data Processing System* and in the IBM *1440 Autocoder (Preliminary Specifications) Systems Bulletin.*

Contents

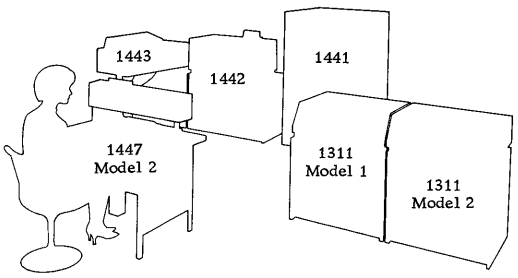


Figure 1. IBM 1440 Data Processing System

Address comments regarding this publication to:
IBM, Product Publications Department, San Jose, California

Section 1

Components of the 1440 Data Processing System

Before directing our attention to programming the IBM 1440 System, we will describe the 1440 components, the characteristics of core storage, and the capabilities and capacities of the system. This background is essential to learning to program the system.

Every data processing system has various components. These are classified as input devices, output devices, and combination input/output or processing devices. The data to be processed by the system is known as input because it is entered into the system through an input device. The data already processed leaves the system through an output device and therefore is known as output.

In the 1440 System, input and output data can be in typewritten form, or in the form of punched cards or magnetized spots on disks. Output in the form of a printed report is further possible.

Processing in the 1440 is performed by the IBM 1441 Processing Unit. The 1440 combination input/output devices are the IBM 1442 Card Read-Punch, the IBM 1311 Disk Storage Drive, and the IBM 1447 Console. The IBM 1443 Printer is an output device only. Figure 1 shows the complete system. To understand the functions of the components and the part they play in the complete data processing operation, let us examine each unit individually.

1441 Processing Unit. The 1441 provides the programmer with *control* over the other components of the system. It contains the logic as described by its stored program and sufficient *storage* to perform its functions among which are the *arithmetic* operations. In summary, then, the three functions of the 1441 unit are: (1) Control, (2) Storage, and (3) Arithmetic.

1442 Card Read-Punch. The 1442 provides the system with input from punched cards and output in the same form.

1443 Printer. The 1443 is used to print high-volume output.

1311 Disk Storage Drive. The 1311 utilizes disk packs to provide storage for large amounts of output. The output stored in the disk packs becomes input to the system.

1447 Console. The 1447 provides external control over the 1440 Data Processing System. It can be equipped with a typewriter for keyboard input to the system and also for use as a low-volume output printer.

The relationship of the 1441 Processing Unit to the other components of the system is shown in Figure 2. The arrows in the figure indicate the direction in which data moves from component to component, as directed by the processing unit. They show that:

1. Data can be read from the card in the read-punch into the processing unit. From this unit the data can be moved as output to disk storage, or to the printer, or to the console printer, or to the card read-punch, or to any combination of these, as required
2. Input data from disk storage can be read into the processing unit and moved from there as output to the printer, or to the console I/O printer, or to the card read-punch, or to disk storage, or to any combination of these, as required
3. Input data keyed in at the typewriter can be read into the processing unit and moved from there as output to the card read-punch, or to disk storage, or to the printer, or to the console I/O printer, or to any combination of these, as required.

IBM 1441 Processing Unit

The IBM 1441 Processing Unit has provision for from 4,000 to 16,000 positions of core storage, in increments of 4,000. Each position is capable of storing an alphabetic, numerical, or special character. Characters are coded in binary-coded-decimal form. When we discuss core storage in a later section, we will have more to say about BCD coding.

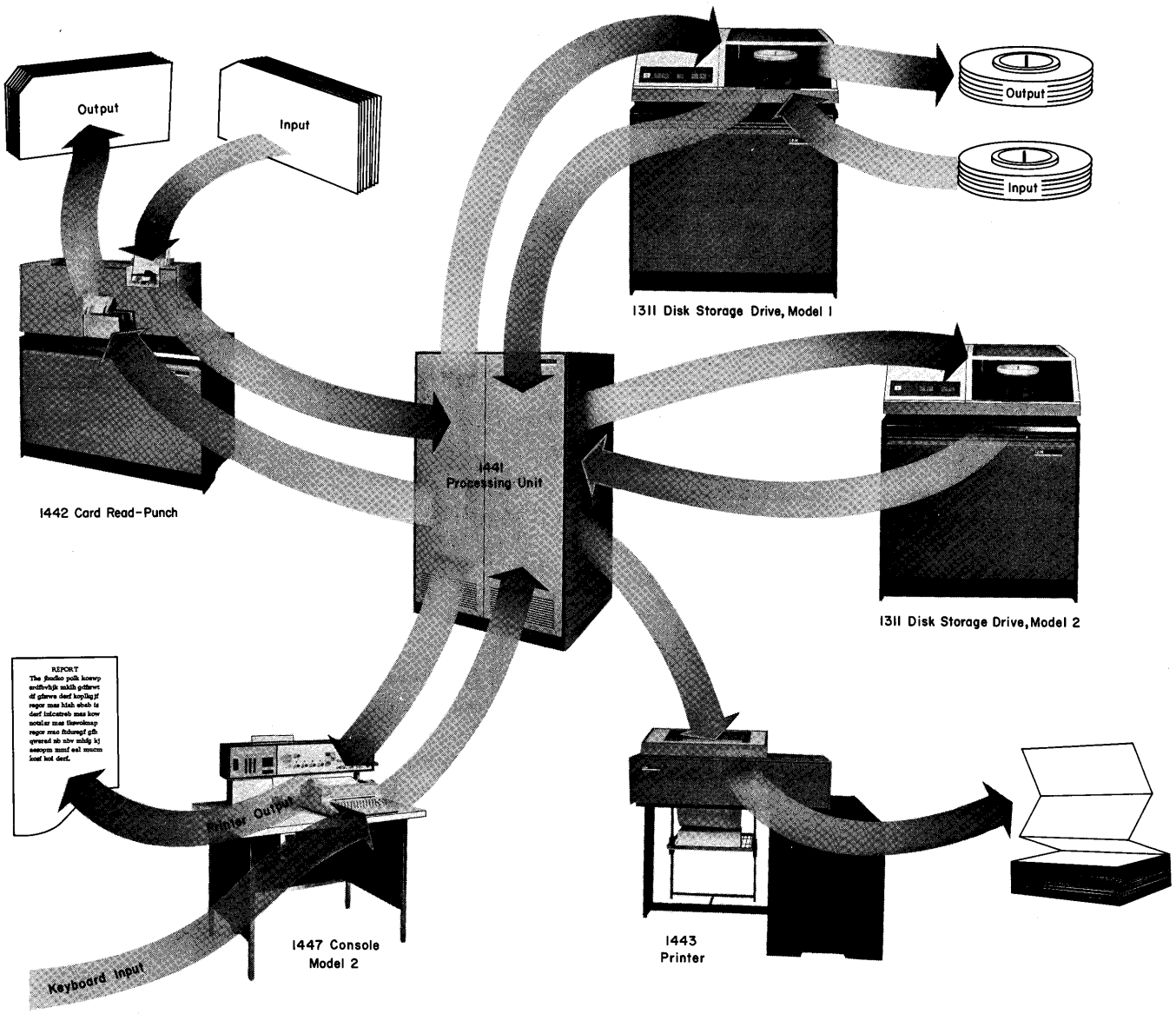


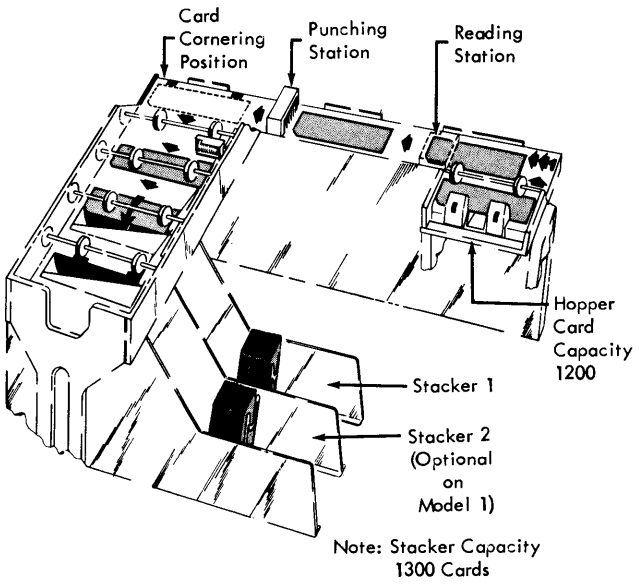
Figure 2. Flow of Data through 1440 System

Each position of core storage is addressable; that is, may be addressed by a program instruction. The data in the location addressed may be used by the computer in performing an arithmetic operation or may be moved to or from the addressed location. Variable length fields as in the 1440 allow greater use and flexibility in storing data and these variable fields can be handled by most program instructions. The program instructions are stored within the unit to perform the operations of input/output, arithmetic, and logic.

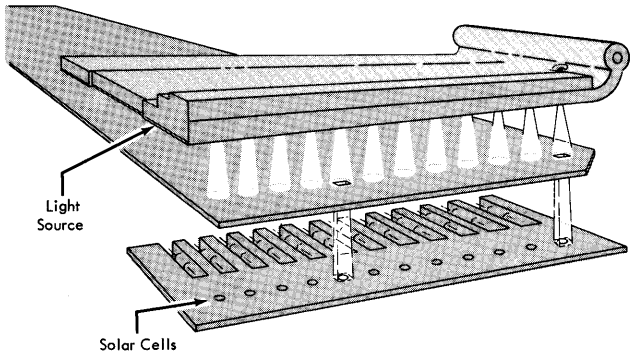
IBM 1442 Card Read-Punch

Card reading and punching functions are provided in one unit and have a single card path. The cards are read or punched, proceeding serially from column 1 to

column 80. This diagram shows the path of the card through the read-punch.



The card *must* pass the reading station before it can be punched. Cards are read at the reading station — column by column — by means of a solar sensing unit, consisting of twelve solar cells, one for each punch position in a card column.



As each column is read, the character is converted from card code to its binary-coded-decimal representation and is stored in core storage. Any area in core storage may be used to store the card data as determined by a stored program instruction.

The 1442 Card Read-Punch is available in two models: Model 1 is equipped with one card stacker and Model 2 with two card stackers to allow card selection. The second stacker is available with the Model 1 as a special feature. A comparison of the card reading and punching speeds for the two models is shown in the following table.

Operation	Model 1	Model 2
Reading only	300 cpm	400 cpm
Reading and then:		
Punching first 10 columns	180 cpm	270 cpm
Punching first 20 columns	130 cpm	210 cpm
Punching first 30 columns	103 cpm	172 cpm
Punching first 40 columns	84 cpm	146 cpm
Punching first 50 columns	62 cpm	112 cpm
Punching first 80 columns	50 cpm	91 cpm

At these card speeds, a minimum of 75 milliseconds of calculating time is allowed for the Model 1 between the reading and punching of each card, and 55 milliseconds for the Model 2.

Two 1442 Card Read-Punches can be attached to the system. This configuration allows the card reading function and card punching function to be performed on separate units, thus providing additional flexibility as the application requires.

IBM 1443 Printer

A single, stored program instruction can cause up to 120 positions of core storage to be read out of storage and printed. Therefore, the instruction or instructions to assemble a complete line of printing within an area in storage should precede the print instruction.

The standard character set for the printer comprises 52 characters, each printable in all 120 printing positions. Two different 52 character sets, designated Type Fonts A and H, are available. A listing of the special characters associated with Type Fonts A and H follows:

16 Special Characters

Card Codes	12	12 3 8	12 4 8	11	11 3 8	11 4 8	0 3 8	0 4 8	0 1	3 8	4 8	12 0	11 0	0 2 8	2 8	5 8
Type Font A	&	.	¤	–	$	*	,	%	/	#	@	?	!	≠	Þ	:
Type Font H	+	.	)	–	$	*	,	(	/	=	'	?	!	≠	Þ	:

Each has the 26 letters of the alphabet, the 10 numerals, and 16 special characters.

NOTE: The numbers at the head of each column are the card codes for the special characters.

Type Font A mounted on a single type bar is the standard character set. The type bar is operator interchangeable, i.e., the operator can remove a type bar of a particular font and replace it with a type bar of a different font.

A character set is repeated several times on a type bar. During printing, the type bar travels on a horizontal plane. When the character on the type bar matches the character desired in the print position, printing occurs. All positions are printed in this manner.

The rated speed of a Model 1 printer equipped with the 52-character set is 150 lines per minute. This speed includes a period of 24 milliseconds for calculating time.

In addition to the 52-character set, a 63-character set, a 39-character set, and a 13-character set are available as special features. These sets also are operator interchangeable. Their rated printing speeds are:

Set	Model 1 (lines per minute)
13 character	430
39 character	190
63 character	120

At these speeds, 24 milliseconds is allowed for calculating.

In addition to the 120-position printing line, a 144-position printing line is available.

Normally, each application prescribes its own format for printing a form. To position a form in its prescribed format, a separate tape must be prepared. This tape, working in conjunction with the stored program, controls the carriage that positions the form for printing.

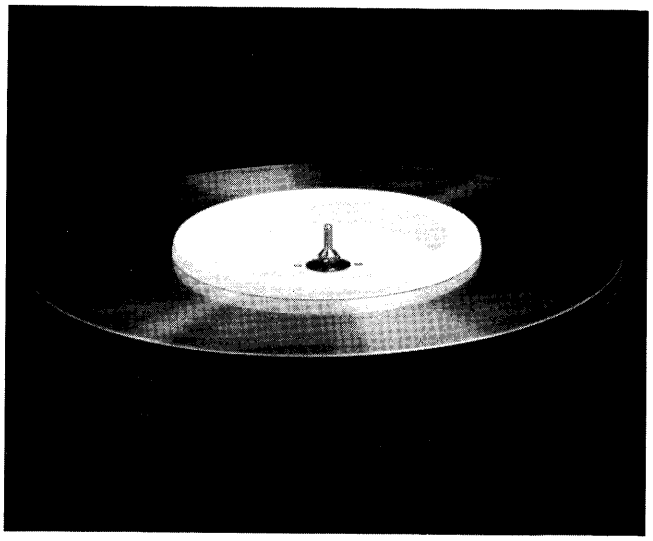
IBM 1311 Disk Storage Drive

As many as five disk storage drives are available to the 1440 Data Processing System. On these drives can be mounted individual disk packs with a storage capacity of 2,000,000 alphameric characters per disk. In this way there can be made available to the system, continually and at any one time, a total of 10,000,000 characters of data. The disk packs can be removed by the operator and replaced with others, in effect providing the system with unlimited storage.

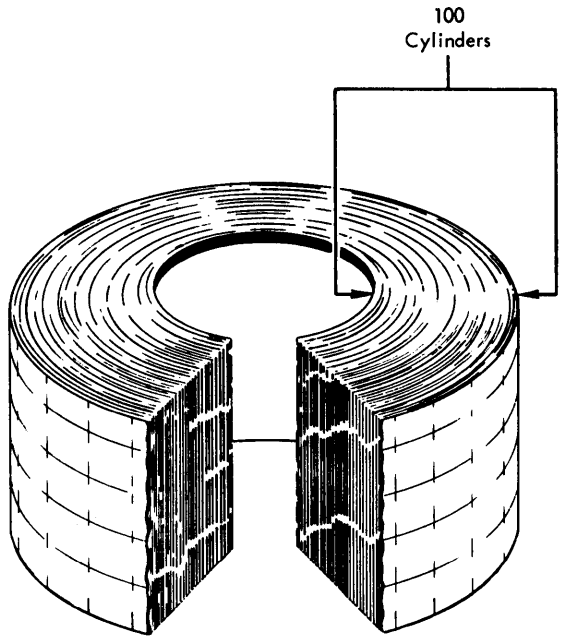
Another feature of the disk packs is their interchangeability. A disk pack from one disk drive can be interchanged with one from another disk drive, either in that same system or in another 1440 system.

The replaceable disk pack offers both unlimited sequential storage and random storage. Records for a particular application can be stored sequentially or randomly or both randomly and sequentially. Retrieval in either manner is also possible. Random records can be retrieved sequentially or sequential records can be retrieved randomly. However, the latter method requires that the programmer provide a table lookup.

The disk pack weighs approximately ten pounds and consists of six 14-inch disks. Each disk surface with the exception of the upper surface of the top disk and the lower surface of the bottom disk is used as a recording surface. Therefore, ten readable surfaces are provided by a disk pack.



Mounted vertically on a disk drive is a comb-type access assembly with five access arms on each of which there are two read-write heads. The access assembly is used to position the read-write heads on the disk surfaces (Figure 3). Each of the access arms, with its two read-write heads, moves between the bottom surface of a disk and the top surface of the next lower disk, and is capable of reading or writing data on both of these surfaces.



As the disk revolves, a read-write head passes over a circular band on the surface of the disk — this is called a *track*. The ten tracks that are exposed simultaneously to the ten read-write heads as the disk pack revolves are referred to as a *cylinder,* because of their cylindrical appearance when represented as in the illustration. A cylinder, in this sense, is a quantitative concept rather than a physical component. It may be considered to be a three-dimensional entity comprising a particular track on each of the ten disk surfaces. There are 100 cylinders, numbered 00 to 99, in a disk pack (Figure 3). The read-write heads can be positioned at any one of these 100 cylinders.

Each track is divided into 20 equal-size sections known as *sectors*. Every sector contains a 5-digit address and provides storage for either 90 characters of data with word marks or 100 characters of data without word marks.

All sectors are addressable. Therefore a disk pack provides 20,000 addresses that may be accessed for reading or writing of data. This figure is arrived at in the following way:

20 addresses per track	×	10 tracks per cylinder	=	200 addresses per cylinder
200 addresses per cylinder	×	100 cylinders	=	20,000 addresses per disk pack

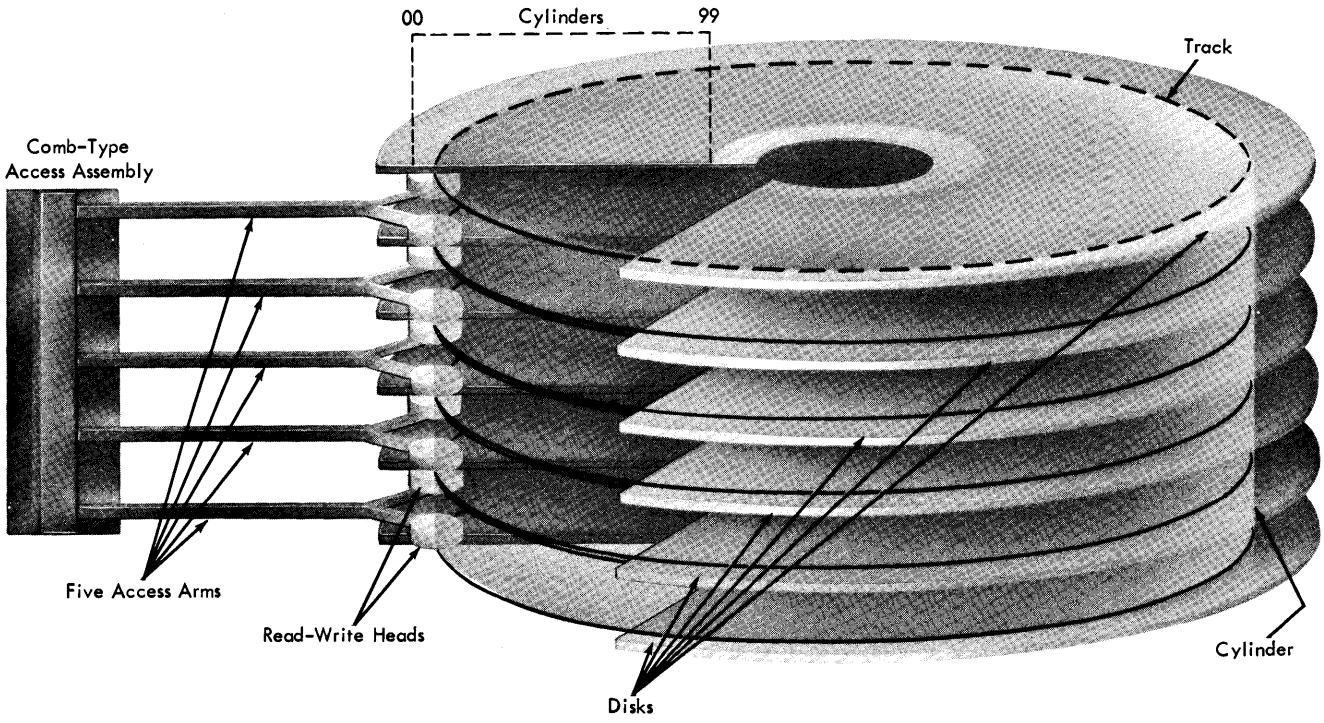


Figure 3. Read-Write Heads Positioned at one of the 100 Cylinders

The 5-digit sector address which precedes each sector in a disk pack is called the *disk sector address.* The sector addresses are in numerical sequence within the track, cylinder, and disk pack. Figure 4 shows the layout of sector addresses for each track of cylinder 00 on the number 0 disk drive. The sector-address ranges for the five disk drives are:

Disk Drive Number	Address Range
0	00000-19999
1	20000-39999
2	40000-59999
3	60000-79999
4	80000-99999

The average time required to seek a sector address is 250 milliseconds; the maximum time, 400 milliseconds. A seek is the operation of positioning the read-write heads (access assembly) at a specific cylinder on a disk drive. For each seek, the access assembly must first retract to its home position (outside cylinder 00) and then move to the selected cylinder. A Direct Seek (special feature) is available which eliminates the return-to-home motion of the access assembly, thus reducing access time. With this special feature, the average access time is 150 milliseconds; the maximum time, 250 milliseconds.

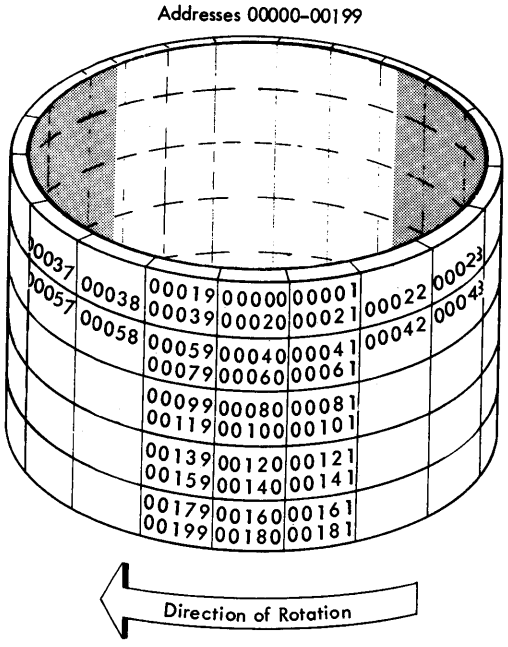


Figure 4. Layout of Sector Addresses by Track for Cylinder 00

IBM 1447 Console

The 1447 Console is offered in two models: the Model 1 is equipped with keys, lights, and switches only. The panel on this console allows the operator to:

1. Start and stop the computer.
2. Initiate the loading of a program into core storage.
3. Display data from core storage.
4. Restart the program after an error condition has caused it to stop.
5. Alter instructions in core storage.

The operation of the keys, lights, and switches on the 1440 will be covered in Part 2 of this book.

The Model 2 has, in addition, a Console I/O Printer. The printer provided by this feature can be used to enter data into core storage and to type out data from core storage. In addition, the operator can use the printer to initiate an inquiry, that is, to request that a certain disk record be printed. Because inquiries are under control of the stored program, the programmer must provide the inquiry test instructions and subroutines necessary to get the record.

Any of the 63 characters that are permissible in core storage can be printed out by the Console I/O Printer. Its character rate is 14.8 characters per second. Normally, this printer is used to type error messages, exception notices, or inquiries.

Core Storage

In describing the 1441 Processing Unit, we mentioned that the 1441 uses core storage to store data and instructions. At this point, we will describe some characteristics of core storage such as coding, organization, and the addressing scheme, for a knowledge of these characteristics will strengthen the programmer's foundation.

Character Coding

Characters are represented in core storage in binary-coded-decimal form. Each character is composed of a unique combination of *bi*nary dig*its,* or *bits* as they are called; one character to one position in core storage. A position in core storage has a 7-bit structure, which is designated CBA 8421.

The four rightmost bits (8421) are called numerical bits, because they correspond in a way to the numerical punches in card coding. Similarly, the BA bits correspond to the zone punches in card coding and are called zone bits. The C bit is the check bit that is generated within the system to give each character an odd or even count of bits, depending on whether the system uses odd or even parity.

The 1440 system uses an odd number of bits and the parity check consists of checking for this odd configuration of bits. It is an internal self-checking device

of the computer for providing it with a high degree of assurance that it is operating correctly.

Table 1 shows the card codes and corresponding binary-coded-decimal form of the 64 characters (this includes a blank character) that can be entered into core storage. Note that the letters A-I have zone bits AB, corresponding in a way to zone punch 12. Similarly, the letters J-R have zone bit B, corresponding to zone punch 11. The letters S-Z have zone bit A, corresponding to zone punch 0. The numbers 0-9 are represented by the numerical bits, 8-4-2-1, alone.

In addition to the seven bits which we have discussed, there is an eighth bit, called the word mark (WM), that is associated with each position of core storage. This bit defines the leftmost position of a data field and the first position of an instruction. Usually, the word mark is placed in the leftmost position of a data field. When it is present in a position of storage, the bit combination in that position is adjusted to maintain an odd number of bits. For example, the letter A, when a word mark $\binom{W}{M}$ is present, appears as:

$$C\ BA\ 1\ \begin{matrix}W\\M\end{matrix}$$

or when the word mark is absent, as:

$$BA\ 1$$

The bit configuration of each character shown in Table 1 is adjusted in the same manner. Zone bits or numerical bits of a character do not change: the configuration of the character changes only with respect to the C bit, which is either added or not added.

Table 1. Character Codes in Ascending Sequential Order

Defined Character	Card Code	Binary Code								Defined Character	Card Code	Binary Code							
		C		B	A		8	4	2	1			C		B	A		8	4	2	1
Blank		C									G	12,7			B	A			4	2	1
. Period	12,3,8			B	A		8		2	1	H	12,8			B	A		8			
⌑ Lozenge	12,4,8	C		B	A		8	4			I	12,9	C		B	A		8			1
[Left Bracket	12,5,8			B	A		8	4		1	! Minus-Zero	11,0			B			8		2	
< Less Than	12,6,8			B	A		8	4	2		J	11,1	C		B						1
‡ Group Mark	12,7,8	C		B	A		8	4	2	1	K	11,2	C		B				2		
& Ampersand	12	C		B	A						L	11,3			B				2	1	
$ Dollar Sign	11,3,8	C		B			8		2	1	M	11,4	C		B			4			
* Asterisk	11,4,8			B			8	4			N	11,5			B			4			1
] Right Bracket	11,5,8	C		B			8	4		1	O	11,6			B			4	2		
; Semicolon	11,6,8	C		B			8	4	2		P	11,7	C		B			4	2	1	
Δ Delta	11,7,8			B			8	4	2	1	Q	11,8	C		B			8			
- Minus Sign	11			B							R	11,9			B			8			1
/ Diagonal	0,1	C			A					1	≠ Record Mark	0,2,8				A		8		2	
, Comma	0,3,8	C			A		8		2	1	S	0,2	C			A				2	
% Percent Sign	0,4,8				A		8	4			T	0,3				A				2	1
∨ Word Separator	0,5,8	C			A		8	4		1	U	0,4	C			A			4		
\ Left Oblique	0,6,8	C			A		8	4	2		V	0,5				A			4		1
⧺ Segment Mark	0,7,8				A		8	4	2	1	W	0,6				A			4	2	
ↄ Substitute Blank	2,8				A						X	0,7	C			A			4	2	1
# Number Sign	3,8						8		2	1	Y	0,8	C			A		8			
@ At Sign	4,8	C					8	4			Z	0,9				A		8			1
: Colon	5,8						8	4		1	0 Zero	0	C					8		2	
> Greater Than	6,8						8	4	2		1	1									1
√ Radical	7,8	C					8	4	2	1	2	2								2	
? Plus-Zero	12,0	C		B	A		8		2		3	3	C							2	1
A	12,1			B	A					1	4	4							4		
B	12,2			B	A				2		5	5	C						4		1
C	12,3	C		B	A				2	1	6	6	C						4	2	
D	12,4			B	A			4			7	7							4	2	1
E	12,5	C		B	A			4		1	8	8						8			
F	12,6	C		B	A			4	2		9	9	C					8			1

Organization and Addressing

Instructions must be placed in core storage before they can initiate operations. The data that is to be processed must also be placed in core storage to be operated upon. Therefore, both data and instructions are in core storage during operation of the program. The allocation of areas in core storage for input data, output data, constants, and work areas is determined by the programmer. Figure 5 shows how 1000 positions of storage, numbered 000-999, can be subdivided into areas. None of the areas is fixed in storage; i.e., the print area, for example, is not always assigned positions 001-120; it could just as easily have been assigned positions 801-920.

The Autocoder system of programming provides certain instructions to aid the programmer in defining areas. If these instructions are used, storage assignment is automatically performed by the Autocoder processor while the object program is being assembled. Automatic assignment of storage is described in greater detail in Sections 2 and 3.

Each position of core storage is addressable by a unique 3-digit mixed alphameric address code. For the most part the programmer need not concern himself with these codes when programming in the Autocoder language. The coded addresses and their decimal equivalents for the first 4000 positions of core storage appear in Figure 6. Decimal numbers

0 through 15999

are used when coding a program in Autocoder language. When the object program is assembled, these numbers are converted into 3-digit machine language codes.

The first one thousand 3-character addresses are coded in machine language as numbers between 000 and 999. Addresses of 1000 and over are handled in a special manner to fit into the 3-character format. The numerical bits of the three characters represent the units, tens, and hundreds position of the address. The

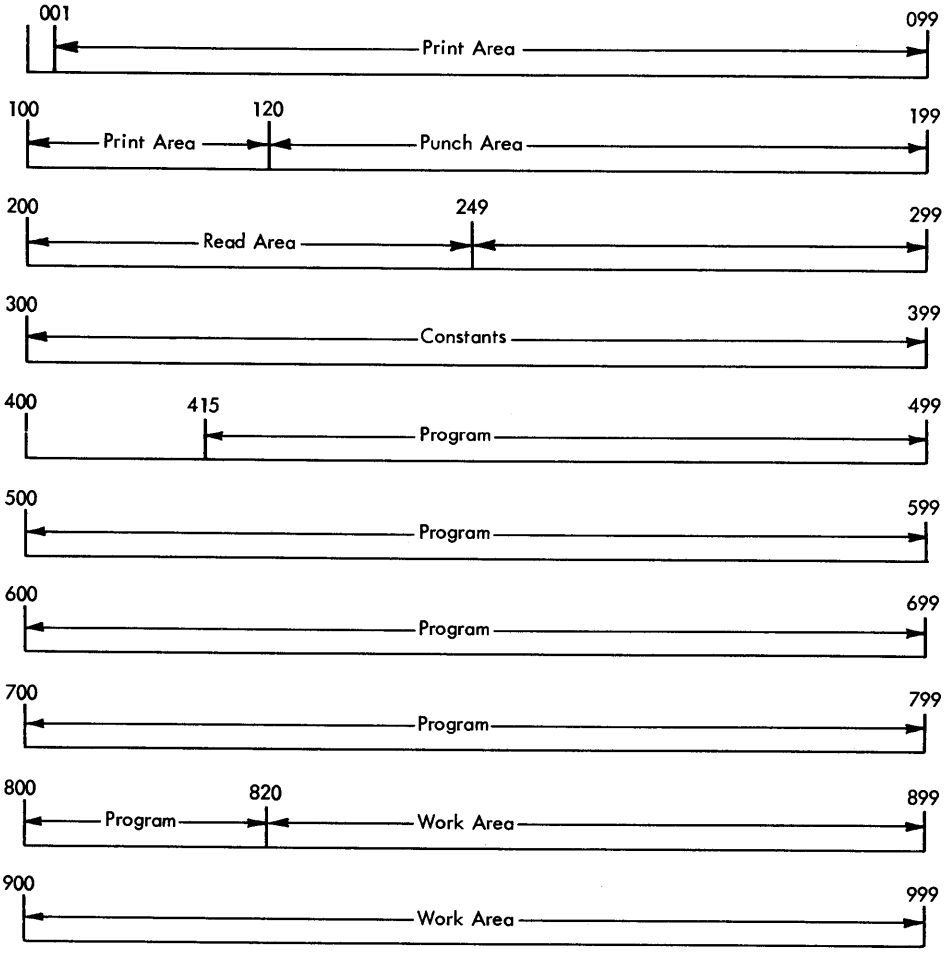


Figure 5. Allocation of Areas in Core Storage

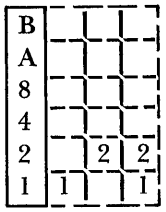
Coded Addresses in Storage		
Equivalent Decimal Addresses		3-Character Addresses
000 to 999	No zone bits	000 to 999
1000 to 1099		≠00 to ≠99
1100 to 1199		/00 to /99
1200 to 1299		S00 to S99
1300 to 1399		T00 to T99
1400 to 1499	A-bit	U00 to U99
1500 to 1599	using 0 zone	V00 to V99
1600 to 1699		W00 to W99
1700 to 1799		X00 to X99
1800 to 1899		Y00 to Y99
1900 to 1999		Z00 to Z99
2000 to 2099		!00 to !99
2100 to 2199		J00 to J99
2200 to 2299		K00 to K99
2300 to 2399		L00 to L99
2400 to 2499	B-bit	M00 to M99
2500 to 2599	using 11 zone	N00 to N99
2600 to 2699		*O00 to O99
2700 to 2799		P00 to P99
2800 to 2899		Q00 to Q99
2900 to 2999		R00 to R99
3000 to 3099		?00 to ?99
3100 to 3199		A00 to A99
3200 to 3299		B00 to B99
3300 to 3399		C00 to C99
3400 to 3499	A-and B-bits	D00 to D99
3500 to 3599	using 12 zone	E00 to E99
3600 to 3699		F00 to F99
3700 to 3799		G00 to G99
3800 to 3899		H00 to H99
3900 to 3999		I00 to I99
* Letter O followed by zero zero		

Figure 6. Core Storage Address Codes

zone bits B and A of the leftmost character are given values of 2 and 1, respectively, and are used to represent the thousands position of the address for addresses up to 3,999. The address

0123

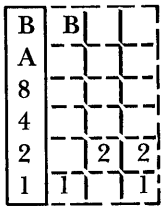
is represented in storage as



whereas the address

2123

is represented as



Parity (C) bits are not shown.

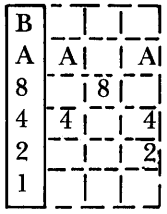
Addresses over 3,999 are handled by using the zone bits of the rightmost character similarly. The B and A bits of this character are given values of 8 and 4, respectively, and are combined with the bits from the leftmost character to represent the thousands and ten thousands positions of the address.

Bit Values	
Leftmost Zone	Rightmost Zone
B = 2	B = 8
A = 1	A = 4

Therefore, the address

5486

is represented in core storage as



The coded address range for addresses 4,000 through 16,000 is shown at the top of the following page.

Equivalent Decimal Address			Zone Bits over Hundreds Character Position	Zone Bits over Units Character Position	3-Character Address Range		
4000	to	4999	No zone bits	A bit (zero zone)	00≠	to	99Z
5000	to	5999	A bit (zero zone)	A bit (zero zone)	≠0≠	to	Z9Z
6000	to	6999	B bit (11 zone)	A bit (zero zone)	!0≠	to	R9Z
7000	to	7999	A, B bits (12 zone)	A bit (zero zone)	?0≠	to	I9Z
8000	to	8999	No zone bits	B bit (11 zone)	00!	to	99R
9000	to	9999	A bit (zero zone)	B bit (11 zone)	≠0!	to	Z9R
10000	to	10999	B bit (11 zone)	B bit (11 zone)	!0!	to	R9R
11000	to	11999	A, B bits (12 zone)	B bit (11 zone)	?0!	to	I9R
12000	to	12999	No zone bits	A, B bits (12 zone)	00?	to	99I
13000	to	13999	A bit (zero zone)	A, B bits (12 zone)	≠0?	to	Z9I
14000	to	14999	B bit (11 zone)	A, B bits (12 zone)	!0?	to	R9I
15000	to	15999	A, B bits (12 zone)	A, B bits (12 zone)	?0?	to	I9I

Section 1. Questions

1. High-volume printed output is normally obtained from
   a. 1443 Printer
   b. 1447 Console I/O Printer
2. If input data is used as output—as, for example, when listing from cards—does this input data go through the processing unit?
3. Can a record which is stored in the processing unit be stored on the disk, be printed out, and be punched into a card?
4. How many alphabetic characters can be stored in 4,000 positions of storage?
5. Which statement is true?
   a. A card must pass the punching station before it can be read.
   b. A card must pass the reading station before it can be punched.
6. What are three of the fundamental functions of the processing unit?
7. How many characters with word marks can be stored in one disk sector? How many characters without word marks?
8. What is the range of sector addresses on cylinder 00 of the second disk drive?
9. What are the five digits that precede each data sector called?
10. Are all positions of core storage addressable?
11. Besides identifying the first position of an instruction, what is another function of the word mark?

Section 2

In Section 1 we described the components of the IBM 1440 Data Processing System and the characteristics of core storage. In this section, we will concern ourselves with the fundamentals of programming and the methods of programming the 1440 itself. This will involve an exploration of the purpose of programming and its functions in data processing. Finally, we will point out the programming details that the programmer must pay attention to, the form that he will use in writing a program, and the method he will employ to describe data.

Purpose of Programming

Data processing by any method requires that an orderly procedure of steps or operations be followed. This procedure must be translated step-by-step into a series of *instructions*, arranged in logical sequence, which the computer can perform. The series of instructions is known as a *program*.

Function of the Programmer

There are two major tasks a programmer must perform to develop a program:

1. To establish the requirements of the problem and the requirements of the solution.
2. To write the program for the computer to follow.

The first task, a systems study, requires that the programmer define the problem in the broad sense: choose the machine or computer configuration; and establish the format or layout of report forms, records, and cards.

The second task requires that, preliminary to the actual writing of the program, the programmer define the problem in a step-by-step analysis and work out the logic of the program in the documented form of block diagrams or the recently developed decision tables or by using both. Either form of documentation is acceptable: both are immeasurable aids.

In the programming examples presented in this book, we have provided both forms of documentation, leaving it up to the student to choose the one he is more familiar with or the one he prefers.

Language of Programming

Problems for the computer can be coded in actual machine language (absolute) if the programmer knows the language of the particular computer; however, even beyond the problems of the machine language itself, the method is often difficult in other respects: in assigning storage locations, in correcting or modifying programs, and in cross referencing within a program. For these reasons, other languages have been developed. We shall use the language called Autocoder for it most nearly eliminates these difficulties. The use of Autocoder language almost completely precludes the programmer's need to know how the computer operates on each and every instruction or how the data is represented in storage. Therefore, we will delay describing machine functions until we have presented the principles of programming.

In summary, the advantages of using Autocoder rather than machine language for our program are:

1. We can use symbolic addresses in the coded program in place of actual addresses.
2. We can replace actual codes in the coded program with mnemonic operation codes that are descriptive and more meaningful.
3. The processor, while it is translating our symbolic source program into an actual (machine language) object program, edits our source program and uncovers coding errors.

Here is a specific set of actual machine language instructions.

```
M % G 1  A 1 2 R
A A 2 5  A 3 5
A A 3 0  A 3 5
M A 3 5  D 3 5
M % G 1  D 1 2 P
```

If we write them in Autocoder language, they will look like this:

Operation	Operands
R	1, RDAREA
A	AFIELD, SUM
A	BFIELD, SUM
MLC	SUM, PCFIELD
P	PCAREA

Each line represents one instruction. These instructions direct the computer to do the following:

No. 1 tells it to — Read a card into the storage read area which is symbolized by "RDAREA."

No. 2 — Add amount of "AFIELD" to amount "SUM."

No. 3 — Add amount "BFIELD" to amount "SUM"

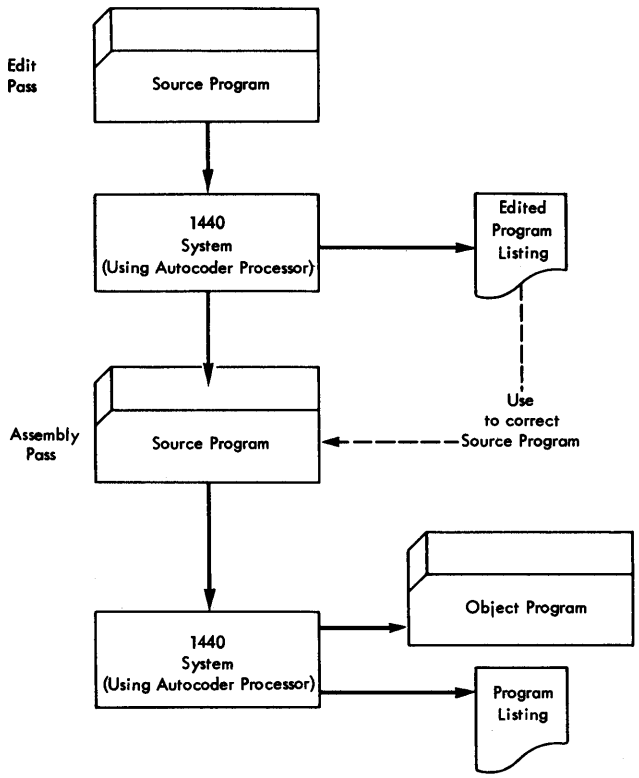
No. 4 — Move the amount "SUM" to its punching location symbolized by "PCFIELD."

No. 5 — Punch a card from the data at the punch area symbolically represented by "PCAREA."

From this example, we can see that operands of an instruction are written immediately adjacent to one another and are separated only by a comma, a requirement of the Autocoder language.

These Autocoder instructions must be transcribed into punched cards, for it is the cards that become the actual input (source program) that the processor will assemble as an object program to be run by the computer. The object program produced by the Autocoder processor contains its own loading instructions; that is, it has instructions for automatically loading itself into its assigned areas of storage when it is entered into the 1440 system.

We may want to edit our source program to uncover coding errors and to be able to correct these errors before proceeding to produce the object program in card form. The edit pass produces an edited program listing. With this listing, we can correct our mistakes, again process the corrected input cards (the source program), and produce the final object program in both card and list form.



The program listing gives the original Autocoder instructions, the assembled instructions, the error messages, and the assignment of storage for instruction and data. This listing is used later to *debug* the program. *Debugging* is testing the program for errors until all errors (bugs) are eliminated. Figure 7 depicts the steps thus far described.

In our description of Autocoder language, we showed some operations and operands and referred to these as program instructions. No mention was made of any particular format in which these instructions should be written because the purpose of the example was to illustrate the difference between machine language and Autocoder language. If we wish to write these Autocoder instructions properly for machine use, we will have to write them in the format prescribed by the Autocoder coding sheet.

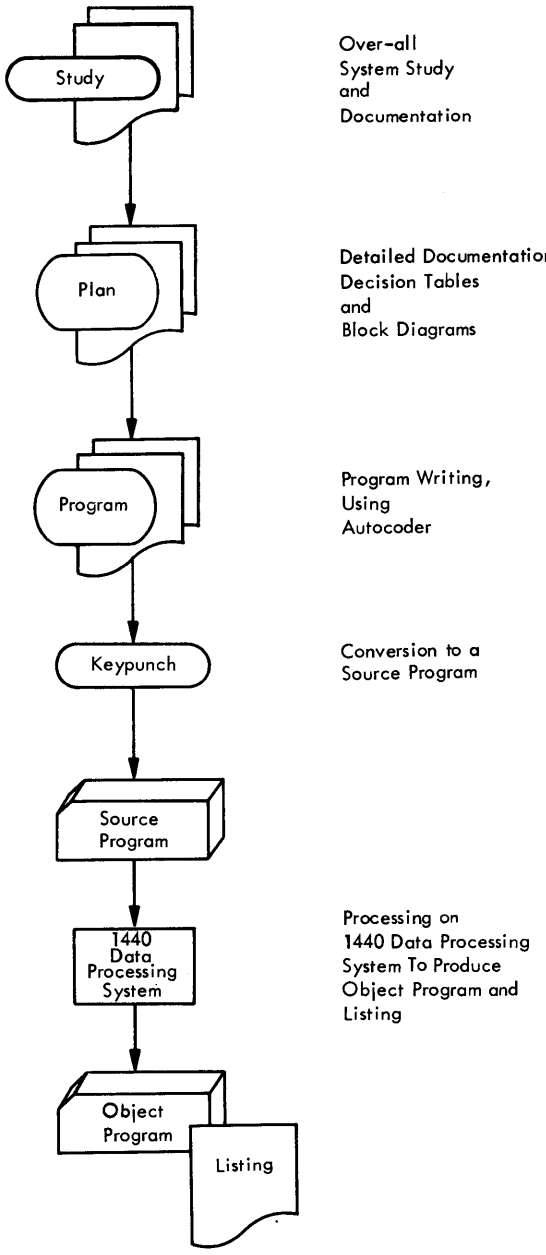


Figure 7. Steps in Preparing an Object Program

Use of Autocoder Coding Sheets

The programmer enters all program entries on the Autocoder coding sheet (Figure 8). The column numbers on the coding sheet indicate the punching format for all input cards in the source deck. Each line of the coding sheet is punched into a separate card. The function of each field of the coding sheet is explained, as follows:

Page Number (Columns 1-2)

This entry provides sequencing for coding sheets.

Line Number (Columns 3-5)

This entry provides sequencing for lines within a coding sheet. The first 25 lines are prenumbered 01-25. The source program cards must be in ascending line number order when they become input to the system.

Label (Columns 6-15)

A symbolic label can have as many as six alphameric characters. The first letter of the label is always written in column 6; the character contained in this position must be alphabetic.

Operation (Columns 16-20)

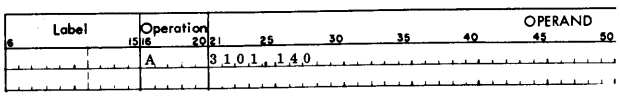
Mnemonic operation codes are written in this field, starting in column 16. Table 2 shows the Autocoder mnemonics for declarative operations, imperative operations, and control operations. *Declarative* operations are described as being those that are used to reserve areas and to store constants. *Imperative* operations are those that are converted, one for one, into instructions or program steps in the object program. *Control* operations are those that enable the programmer to exercise some control over the assembly process.

Operand (Columns 21-72)

The operand field in an imperative instruction can contain:

1. The actual or symbolic address of the data to be acted upon, as directed by the command in the operation field. (NOTE: a label that appears as an operand is considered a *symbol.*)
2. The actual data to be operated upon; data in this form is referred to as a *literal.*
3. The address constant, i.e., the symbol which represents the actual 3-character address that is assigned to the corresponding label. (A plus or minus sign must always precede an address constant so that the processor can distinguish it from a symbolic address.)

Operands in Autocoder language are always separated by commas. The statement that follows demonstrates the use of commas. This instruction causes data specified by the A operand (actual address 3101) to be added to the data specified by the B operand (actual address 140).



IBM

Program _____

Programmed by _____

Date _____

INTERNATIONAL BUSINESS MACHINES CORPORATION

IBM 1401, 1410, AND 1440 DATA PROCESSING SYSTEMS

AUTOCODER CODING SHEET

Identification _____ (76 80)

Page No. ___ (1 2) of _____

Line	Label	Operation	OPERAND
3 5	6 15	16 20	21 25 30 35 40 45 50 55 60 65 70
0 1			
0 2			
0 3			
0 4			
0 5			
0 6			
0 7			
0 8			
0 9			
1 0			
1 1			
1 2			
1 3			
1 4			
1 5			
1 6			
1 7			
1 8			
1 9			
2 0			
2 1			
2 2			
2 3			
2 4			
2 5			

Figure 8. Autocoder Coding Sheet

Note that the first and second operands of an imperative statement are referred to as the A and B operands. A third operand is called the d-modifier. The next statement is an example of a statement that contains all three operands.

Label	Operation	OPERAND
6 15	16 20	21 25 30 35 40 45 50
	BCE	ENTRYA,,SWITCH,2

This instruction, when assembled, tests the storage location equivalent to SWITCH for a digit 2 and branches to the location equivalent to ENTRYA for the next instruction, provided SWITCH contains a 2. The digit 2 is the d-modifier. Only certain Autocoder operations require a d-modifier. They are indicated by ‡ in Table 2.

The programmer can include a remark or comment anywhere in the operand field, if he leaves at least two blank (nonsignificant) spaces between the remark and the operand. If he added a comment to the previous statement, it would appear as follows:

Label	Operation	OPERAND
6 15	16 20	21 25 30 35 40 45 50 55 60 65 70
	BCE	ENTRYA,,SWITCH,,2, ,TEST SW FOR A DIGIT 2 AND BRANCH

Table 2. Autocoder Mnemonic Operation Codes

Mnemonic Operation Code	Description
DECLARATIVE OPERATIONS	
DA	Define Area
DC	Define Constant
DCW	Define Constant with Word Mark
DS	Define Symbol
DSA	Define Symbolic Address
EQU	Equate
IMPERATIVE OPERATIONS	
ARITHMETIC TYPE	
A	Add
D	Divide*
M	Multiply*
S	Subtract
ZA	Zero and Add
ZS	Zero and Subtract
DATA CONTROL TYPE	
MCE	Move Characters and Edit
MCS	Move Characters and Suppress Zeros
MLC	Move Characters to A or B Word Mark
MLCWA	Move Characters and Word Mark from A-field
MLNS	Move Numerical Portion of Single Character
MLZS	Move Zone Portion of Single Character
MRCM	Move Characters to Record Mark or Group Mark with a Word Mark
LOGIC TYPE	
B	Branch Unconditional
BAV	Branch on Arithmetic Overflow
BC9	Branch on Carriage Channel 9
BCV	Branch on Carriage Channel 12
BE	Branch on Equal Compare (B = A)
BH	Branch on High Compare (B > A)
BL	Branch on Low Compare (B < A)
BU	Branch on Unequal Compare (B ≠ A)
BLC	Branch on Last Card (Sense Switch A)
BPB	Branch Printer Busy
BW	Branch on Word Mark
BWZ	Branch on Word Mark or Zone ‡
BCE	Branch If Character Equal ‡
BSS	Branch If Sense Switch On ‡
C	Compare
BIN	Branch If Indicator On ‡
BBE	Branch If Bit Equal* ‡
INPUT/OUTPUT TYPE	
W	Write a Line
WS	Write a Line and Suppress Spacing
WCP	Write Console I/O Printer
RCP	Read Console I/O Printer
R	Read to Group Mark with a Word Mark
P	Punch and Feed
PS	Punch and Stop
LU	Load Unit*
MU	Move Unit*
WDC	Write Disk Check
WDCW	Write Disk Check with Word Marks
SDL	Scan Disk Low, Equal*
SDLW	Scan Disk Low Equal with Word Marks*
SDE	Scan Disk Equal*
SDEW	Scan Disk Equal with Word Marks*
SDH	Scan Disk High, Equal*
SDHW	Scan Disk High, Equal with Word Marks*
SD	Seek Disk
RD	Read Disk Sector(s)
RDW	Read Disk Sector(s) with Word Marks
RDT	Read Disk Track Sectors with Addresses
RDTW	Read Disk Track Sectors with Addresses and Word Marks
WD	Write Disk Sector(s)
WDW	Write Disk Sector(s) with Word Marks
WDT	Write Disk Track Sectors with Addresses
WDTW	Write Disk Track Sectors with Addresses and Word Marks
RDCO	Read Disk with Sector Count Overlay
RDCOW	Read Disk with Sector Count Overlay and Word Marks
WDCO	Write Disk with Sector Count Overlay
WDCOW	Write Disk with Sector Count Overlay and Word Marks
RDTR	Read Disk Track Record*
RDTRW	Read Disk Track Record with Word Marks*
RDTA	Read Disk Track Record and Address*
RDTAW	Read Disk Track Record and Address with Word Marks*
WDTR	Write Disk Track Record*
WDTRW	Write Disk Track Record with Word Marks*
WDTA	Write Disk Track Record and Address*
WDTAW	Write Disk Track Record and Address with Word Marks*
MISCELLANEOUS TYPE	
CC	Control Carriage ‡
CS	Clear Storage
CW	Clear Word Mark
H	Halt
MA	Modify Address*
NOP	No Operation
SAR	Store A Address*
SBR	Store B Address*
SS	Select Stacker*
SW	Set Word Mark
CONTROL OPERATIONS	
CTL	Control
END	End
EX	Execute
LTORG	Literal Origin
XFR	Transfer
JOB	Job
ORG	Origin
SFX	Suffix

* Indicates a Special Feature

‡ Indicates that the programmer must supply d-modifier in Autocoder Statement

A comment in no way affects the assembled machine language instruction, however it does appear on the program listing opposite the assembled instruction.

In preparing a program, it is necessary to determine the constants and storage areas that are needed, and to define them. Usually this step is completed before the program is written; however, it may be accomplished as the program is being written.

Symbolic Assignment of Core Storage

In Autocoder language, assignment of symbolic names (labels) to data fields, constants, or instructions relieves the programmer of the problem of keeping track of actual storage locations. The programmer should assign names that are meaningful. For example, he might label a 6-position field reserved for withholding tax as WHTAX. A word of caution—once he has assigned this label, he must be careful not to assign it to two different fields.

The Autocoder processor program assigns the actual storage locations as it produces the object program. The Processor has a *location assignment counter* for this purpose. The counter is set at 210 at the start. Each time the processor assigns an area, the counter is incremented by the number of positions in that area. The last number in the assignment counter is the address of the area.

A 4-position constant (for example, ABCD) is assigned the address 213, provided the assignment counter was set at 210 when the statement defining the constant was encountered. Subsequent statements will be assigned storage in the order in which they are encountered. The Autocoder language contains control instruction that allow the programmer to preset the counter to a particular address or to change the setting of the counter during assembly. Thus, the programmer can start the assignment of storage at any address he specifies.

There are no permanently assigned areas within core storage except for storage positions 0087-0099 which are used by the Indexing feature (a special feature). Areas for input and output are assigned by the processor to areas which it selects (unless specified otherwise) and with one restriction. The leftmost position of an area reserved for printing on the 1443 Printer must be an address that ends with 01 (XX01).

The Autocoder instructions used to create constants and to reserve areas in core storage are known as Declarative operations. Here we describe some of the ways to write these instructions and show how the processor operates upon them.

Storage Assignment with Declarative Operations

DCW Define Constant with Word Mark

This operation is used to create alphameric or numerical constants, address constants, or blank areas, in core storage as part of the object program. A word mark is automatically added to the leftmost position of the created field.

A numerical constant can be given a plus or minus value by preceding it with a + or − sign in the statement in which it appears. Numerical constants take the following form

Label (6–15)	Operation (16–20)	OPERAND (21–50)
CONST1	DCW	+1234
CONST2	DCW	-5632
CONST3	DCW	1234567

where the constant itself may be any number of characters. The address that is associated with the label of the constant will be that of the rightmost position of the constant itself. An alphameric constant or a blank constant can also be defined; these constants must be preceded and followed by an "at" (@) sign.

Label (6–15)	Operation (16–20)	OPERAND (21–45)
CONST4	DCW	@ERROR MESSAGE 1@
CONST5	DCW	@JANUARY 28, 1962@
CONST6	DCW	@ @

CONST6, which is a 9-position blank field, can also be defined by the following statement;

Label (6–15)	Operation (16–20)	OPERAND (21–50)
CONST6	DCW	#9

where the # symbol precedes the number that tells how many blank storage positions are to be defined.

DC Define Constant

This operation is the same as the DCW operation except that a word mark is *not* added by the processor to the leftmost position of a field being defined. The statements are written in the same manner as DCW statements.

DS Define Symbol

This operation is used to reserve an area of core storage and to associate a label with the address of that area. Unlike the DCW or DC operation, the DS operation does not load any data as part of the object program.

Label	Operation	OPERAND
ACCUM	DS	10

In the example given, the number of positions to be reserved is ten, as specified by the value of the number in the operand field.

DA Define Area

The DA operation is used to define an area for input or output and to place a group mark with a word mark in a position immediately to the right of the area. This operation is unlike the declarative operations described so far, where the address associated with the label is the rightmost position of the area. The address of a label associated with a DA statement is the *leftmost* position of the area. Entries following the DA entry are used to define fields and subfields that fall within the limits of the area. For example, the statements

Line	Label	Operation	OPERAND
01	ITEM	DA	1X53,G LAYOUT OF ITEM CARD
02	CODE		1,1
03	DATE		2,5 FIELD
04	MO		3 SUBFIELD
05	DAY		5 SUBFIELD
06	SLMNO		6,8
07	CLASS		9,13
08	UP		14,19
09	QTYS		20,22
10	QTYBO		23,25
11	AMT		26,31
12	SUBS		32,37
13	ITMNO		38,45
14	QTY		46,48
15	CUSNO		49,53
16			

reserve a 53-position storage area followed by a group mark with a word mark. The first position of the area becomes the first field, the second through fifth positions become the second field, etc.

A field within a field is called a subfield. MO and DAY are subfields of the DATE field. In the example, the rightmost position of the subfields is written in column 21. Word marks are not generated by the processor for subfields.

A word mark is automatically generated in the leftmost position of each field, provided there are two numbers in the operand field and these are separated by a comma. The data generated by the statement appears in the object program as:

$$\underline{b}\,\underline{b}\,b\,\underline{b}\,\underline{b}\,b\,\underline{b}\,\underline{b}\,b\,\underline{b}\,\underline{b}\,b\,\underline{b}\,\underline{b}\,b\,\underline{b}\,\underline{b}\,b\,\underline{b}\,\underline{b}\,b\,\underline{b}\,\underline{b}\,b\,\underline{b}\,\underline{b}\,b\,\underline{b}\,\underline{b}\,b\,\underline{b}\,\underline{b}\,b\,\underline{b}\,\underline{b}\,b\,\underline{b}\,\underline{b}\,b\,\overset{G}{\underline{M}}$$

where $\underline{b}$ represents a blank position with a word mark, b represents a blank position, and $\overset{G}{\underline{M}}$ represents a position which contains both a group mark and word mark. The address of the rightmost position of each field is associated with its corresponding label.

The following statements show how three similar areas, each followed by a *record mark*, can be defined.

Label	Operation	OPERAND
RECBLK	DA	3X5,G
NAME		1,2
DATE		3,5

These statements generate

$$\underline{b}\,\underline{b}\,\underline{b}\,b\,b = \underline{b}\,\underline{b}\,\underline{b}\,b\,b = \underline{b}\,\underline{b}\,\underline{b}\,b\,b = \overset{G}{\underline{M}}$$

as a part of the object program. The number in column 21 in the operand field specifies the number of times the fields should be repeated.

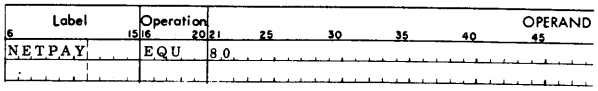
EQU Equate

This operation is used to assign a label to an actual or a symbolic address. It allows the programmer to assign different labels to the same storage location and to refer to the storage location by either name. No storage is reserved and no data is loaded as part of the object program when this operation is used.

Label	Operation	OPERAND
INDIV	EQU	MANNO

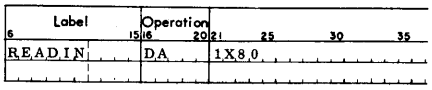
Suppose, for example, that the symbol MANNO is assigned the address 1234 by the processor. When the EQU operation is encountered, the label INDIV will be assigned the same address (1234).

The following statement equates the label NETPAY to an actual address.



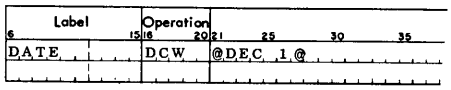
Section 2. Questions

1. What are the two major tasks of a programmer?
2. What is the name of the programming language we use in writing instructions for the 1440 system?
3. Area definition statements are used for which of these?
   a. to instruct the machine to perform arithmetic.
   b. to assign storage areas for data.
   c. to control the processor function.
   d. to write instructions for the computer.
4.

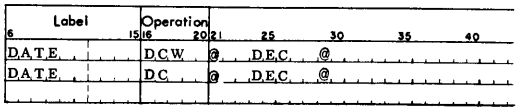


If the statement shown above is entered in a program, will subsequent statements using the symbol READIN be addressing:
   a. the leftmost position of the area reserved?
   b. the rightmost position of the area reserved?

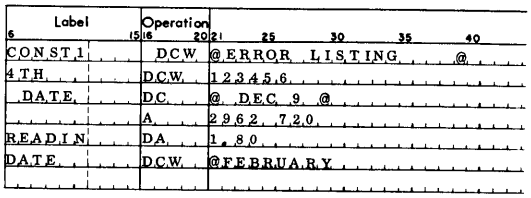
5.



In this example the address of the constant associated with the label DATE will be:
   a. the address of the leftmost position of the constant.
   b. the address of the rightmost position of the constant.
7. What is the difference in the result of these two statements?



8. Correct the following statements.



Section 3

Since we are now familiar with the programming tools that are used in planning and coding a job, we will proceed to program a basic job. In this job, two factors are to be read from a card, added together, and the total punched into the same card.

Figure 9 shows the format of the input cards. Field B is to be added to field A and punched into field C. Whenever possible, the field or fields that are to be punched should be located near the column-1 end of the card. This arrangement minimizes the time required to punch each card and thereby maximizes card throughput. The term *card throughput* refers to the number of cards per minute that can be processed by the system for any given job.

Block diagram, Figure 10A, and decision table, Figure 10B, initiate the program with the reading of a card.

The second step is finding the total of A plus B and punching the result. The third step is a test for last card. The fourth step is a branch back to the first action (read a card). These four steps are repeated for each card until all cards are processed.

Twelve Autocoder statements (see Figure 11) constitute the coded program. The first three statements define the core storage area into which the card is read; the next two statements define the area from which the card is punched; and the remaining seven statements are the imperative statements that make up the program instructions. This ordering of the statements is only one possibility; the imperative statements could have been written or could have appeared first and been followed by the five declarative statements.

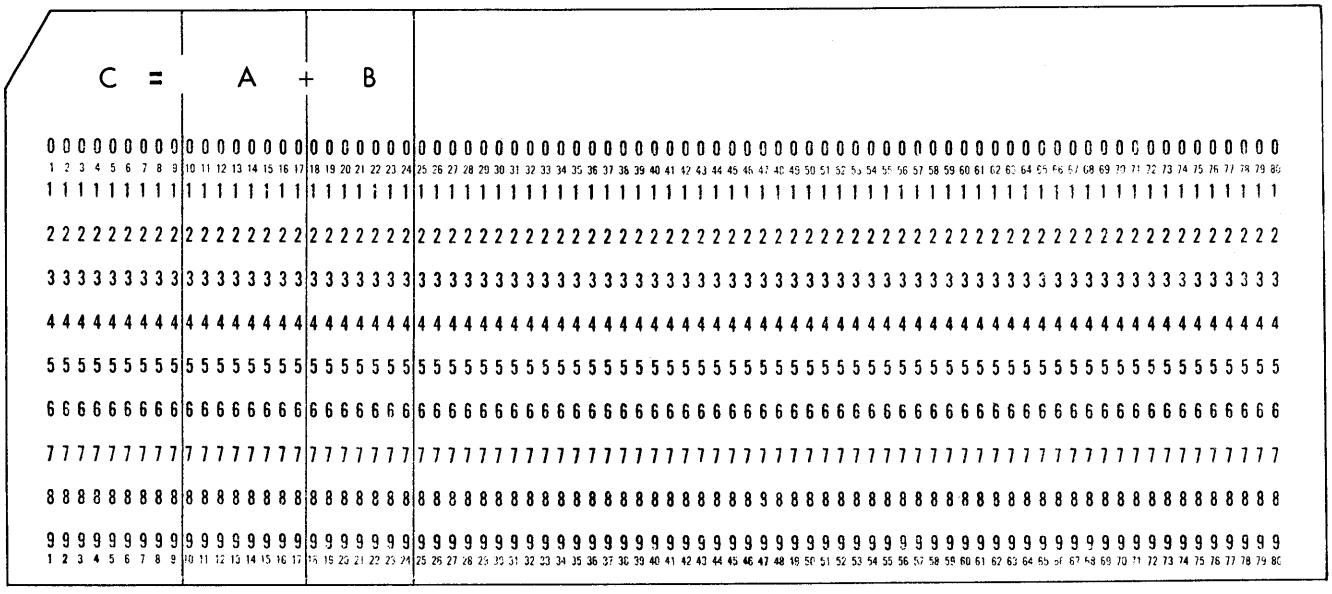


Figure 9. Card Format for A + B = C

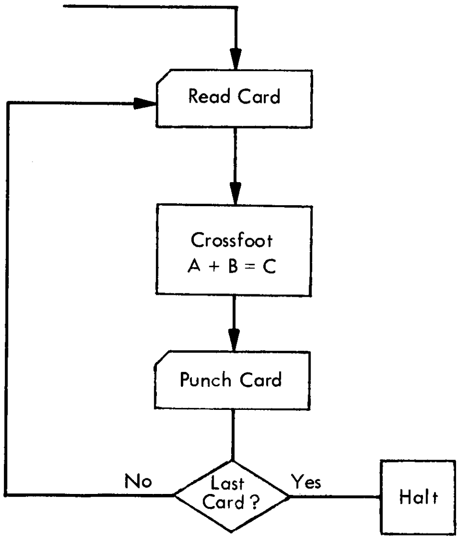


Figure 10A. Block Diagram for A + B = C

** START												
1 HAS LAST CARD BEEN PROCESSED ?	N	Y										
2 READ CARD	X											
3 SET C = A	X											
+ B	X											
4 PUNCH CARD	X											
5 STOP		X										

Figure 10B. Decision Table for A + B = C

A detailed analysis of the coding sheet is given in Figure 11. *Line 01* of the coding sheet reserves 24 positions in storage and one additional position for a group mark with a word mark.

Reserving an Area for Card Reading. The number of positions the programmer reserves must equal the highest-numbered card column to be read. In this example, the highest-numbered card column is 24 (see Figure 9). The label RDAREA will be assigned the address equivalent to the leftmost position of the read area when the program is assembled.

Lines 02 and *03* assign the labels AFIELD and BFIELD to the addresses of the A and B factors, respectively. In effect, the two fields within the read area are defined and word marks are placed in the leftmost positions of the fields to identify their limits. Because the processor initiates assignment of storage, the read area will start in storage position 210 and will extend into higher-numbered positions, as shown:

```
                  A Field      B Field
bbbbbbbbbbbbbbbbbbbbbbbbGM
↑
210
```

Line 04 reserves a 9-position punch area labeled PCHARA and sets a group mark with a word mark in the position to the immediate right of the area.

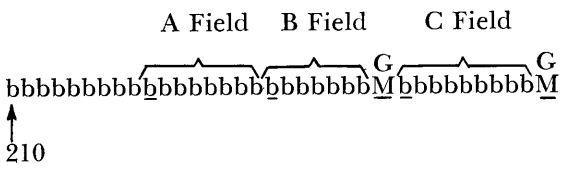
Reserving an Area for Card Punching. An area reserved for punching must contain the number of positions equal to the highest-numbered card column to be punched. The position to the right of this area must contain a group mark with a word mark.

Line (3–5)	Label (6–15)	Operation (16–20)	OPERAND (21–65)
01	RDAREA	DA	1X24,G DEFINE READ AREA
02	AFIELD		10,17
03	BFIELD		18,24
04	PCHARA	DA	1X9,G DEFINE PUNCH AREA
05	CFIELD		1,9
06	START	R	1,RDAREA READ A CARD INTO READ AREA
07		ZA	AFIELD,CFIELD CROSSFOOT
08		A	BFIELD,CFIELD
09		PS	1,PCHARA PUNCH FROM PUNCH AREA
10		BLC	HALT TEST FOR LAST CARD
11		B	START
12	HALT	H	END
13			

Figure 11. Coding Sheet for A + B = C

Line 05 assigns the label CFIELD to the address equivalent to the rightmost position of the field to be punched. A word mark is set in the leftmost position of this field, thus identifying its limit.

After assembly, the combined read and punch areas reserved by the first five statements will appear as:



Line 06 initiates a card reading operation that causes card columns 1-24 to be read into core storage, starting at position 210. The first group mark with a word mark encountered terminates the transfer of data to storage.

> ***Card Reading Operation.*** In a card reading operation, the A operand (1) is a constant that specifies card reading is to be performed by the 1442 Read-Punch, Unit Number 1. If the constant (A operand) contained a 2, the card reading operation would be performed instead by Unit Number 2 (special feature). The symbol RDAREA specifies the address in core storage where the storing of data starts (position 210), i.e., where the data from card column one is stored. We will want to branch back (return) to this instruction from another point in the program, therefore we must give it a label. The label START causes the address of the instruction to be saved for further use during the assembly of the object program.

Line 07 is a ZA statement which resets to zeros the field in the punch area containing the C factor and then adds the A factor into the cleared area.

> ***Zero and Add Operation.*** In a ZA statement, the A operand specifies the field to be added whereas the B operand specifies the field to be cleared (reset to zeros). Data to be added is transmitted, starting with the rightmost position. A word mark terminates the operation.

Word Marks in Arithmetic Operations. The A-field word mark stops data transfer from the A field. The B-field word mark stops the operation and data transfer. Therefore, if both fields are of equal length, only the B-field word mark must be present. If the A field is shorter than the B field, a word mark must be present in both fields.

We will find that this use of word marks is the same for all other arithmetic operations with the exception of multiply and divide (special features). In our example, word marks have been set in the leftmost position of the fields by the DA statements. These word marks are not erased or affected by an arithmetic operation.

Line 08 adds the B factor to the A factor to produce the total *C*.

> ***Add Operation.*** In an Add (A) operation, the data addressed by the A operand is added to the data addressed by the B operand, digit by digit, starting in the rightmost position. The total that is developed replaces the data previously stored at the B address.
>
> The use of word marks in an Add operation is the same as in the ZA operation just described.

Line 09 initiates a card punching operation. The A operand indicates that the punching is to be performed by the Read-Punch, Unit Number 1. The B operand PCHARA is the address of the leftmost position of the core storage area to be punched.

> ***Card Punching Operation.*** The A operand specifies which Card Punch unit is to be used: code 1 for Punch unit, Number 1 and code 2 for Punch unit, Number 2. Punching starts with the addressed position and continues through higher-numbered positions until a group mark with a word mark terminates the operation. Either the PS operation code (Punch and Stop) or the P operation code (Punch and Feed) can be used to start the punching operation.
>
> If the PS operation code is used, the card remains at the punching station after punching is completed until the next card is read or until a P operation code is given. If the P operation code is used, the card after being punched is ejected from the punching station into the stacker and another card is fed past the reading station to the punching station.

Line 10 tests the last card indicator by use of a BLC operation code.

> ***Branch on Last Card Operation.*** The A operand (HALT) directs a branch to the instruction labeled HALT, if the Last Card indicator is on. If it is not on, the next sequential instruction is executed. The BLC statement provides the logical means for ending the program after the final card has been processed.

Test for Last Card. In the 1440 system the Last Card indicator turns on, when the last card passes the reading station, if Sense Switch A is in the ON position. In most programs the last card test normally follows the series of instructions which process the data read from a card. In our job, we want to direct the program to halt or end when the last card indicator is found to be on. The fact that it is on indicates that the last card of the deck has been processed and punched.

Line 11 causes an unconditional branch to the instruction labeled START, i.e., it causes the program to return to the card reading instruction to begin the sequence of steps again.

> ***Branch Unconditional Operation.*** The B operation is used at the end of a series of instructions. It always causes a branch to the address of the instruction specified by the A operand.

Line 12 provides the Halt instruction that causes the program to end. This statement is labeled so that it may be branched to from another point in the program. When the last card condition is sensed (see line 10 of the coding sheet), the program branches to the Halt instruction.

> ***Halt Operation.*** In every program, the programmer must provide a Halt (H) instruction to stop the computer after it has processed the data. No operands are required for this operation.
>
> If the Halt instruction is written with an operand, the program will branch, when the Start key is depressed, to the address specified by the operand.

Arithmetic Sign Control and Overflow

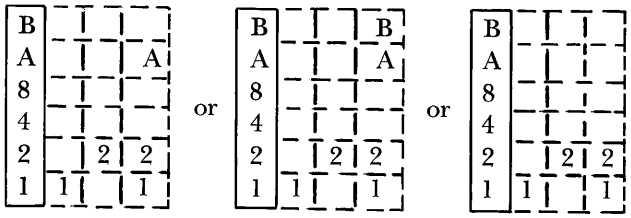
Sign Control. Since two of the arithmetic instructions, Zero and Add (ZA), and Add (A), have been described, it is appropriate at this time for us to consider arithmetic sign control and overflow.

Algebraic signs of fields within the computer are indicated by the zone bits in the rightmost position of the fields.

A positive field may be represented in any one of three ways:

by an A bit only,
by both A and B bits,
by no A and B bits.

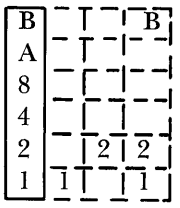
For example, a positive 123 may appear in core storage as:



A negative field is indicated in one way only:

by a B bit.

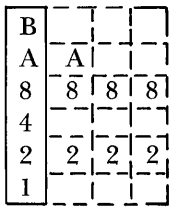
A negative 123 will appear as:



A field containing all zeros can be positive or negative. If a positive zero field and a negative zero field are compared by the computer, the computer will recognize them as being unequal because the zone bits in the units position of the two fields are not alike. The programmer, therefore, must take into account the bit configuration of positive and negative fields when comparing fields that have been added or subtracted. We will delve deeper into this subject in the next section when we describe the Compare operation.

Overflow. When an arithmetic overflow occurs, the zone bits in the leftmost position of the field (specified by the B operand) are altered to signify the overflow.

For example, a field containing 999 to which a 1 is added, thereby causing an overflow, will appear in storage as:



The overflow has caused an A bit to be set in the leftmost position. The zone bit configurations associated with the first overflow and subsequent overflows are:

Overflow	Zone Bits
First	A bit
Second	B bit
Third	A and B bits
Fourth	No A or B bits
Fifth	A bit
Sixth	B bit
Seventh	A and B bits
Eighth	No A or B bits

Additional overflows repeat the zone bit configuration of the first four overflows, as shown. In addition to the manipulation of the zone bits for overflows, an Overflow indicator is turned on when the first overflow occurs. Testing the indicator with a Branch instruction turns it off.

Section 3. Questions

1. When the last card indicator turns on, the last card
   a. is at the reading station ready to be read.
   b. is sitting in the card hopper.
   c. is beyond the punching station or in the card stacker.
   d. is passing the reading station.

2. Which Autocoder operation code is used to define a punch area in core storage?
   a. BCE
   b. DC
   c. DA
   d. ZA

3. How many positions of core storage must be reserved for reading card columns 30-48?

4. Which of these numerical digits stored in core storage are positive?

   B
   a. A
      2

   b. A
      2

   c. B
      2

5. In an Add operation where the A field is shorter than the B field,
   a. only the leftmost position of the B field must contain a word mark.
   b. only the leftmost position of the A field must contain a word mark.
   c. both fields must contain a word mark in their leftmost positions.
   d. neither field requires a word mark.
   e. both fields must contain a word mark in their rightmost positions.

6.

Line	Label	Operation	
01	BEGIN	R	1,RDAREA
02		ZA	A,B
03		A	D,B
04		A	E,B
05		PS	1,PCHARA
06		BLC	STOP
07		B	BEGIN
08	STOP	H	
09			

In this program a label is given to line 01 so that:

   a. the address of the read area will be assigned to the label BEGIN.
   b. the program will have a starting point.
   c. a later program step can branch to the Read instruction.
   d. the start of the program can be identified.

Section 4

In Section 3 we saw an example of a program written in Autocoder language for a basic job of reading two factors from a card and punching the result into the same card. In this section we will describe a series of Autocoder programs that involve operations with the Read-Punch, the Printer and, of course, the Processor. These problems illustrate many programming principles and are designed to broaden the student's programming knowledge and ability.

In the last section, the A operand was used to specify the address of a particular field. Likewise, the B operand was used to specify the address of a second field. Henceforth, we will refer to the fields which are associated with the A and B operands as the A and B fields, respectively.

Programming Example 1. Detail Printing from Cards

Detail printing from cards is printing from each card as it passes through the Card Read-Punch. All 80 columns of the card can be read and printed or only certain fields may be printed.

Figure 12 shows the format of the cards to be read in our example; those fields marked with a ✓ are to be printed. Figure 13 shows the order in which the fields will appear on the printed line in the listing. This listing contains a column headed "invoice amount." The invoice amount is not read from the card: it is the sum of the discount allowed plus the amount paid. The sum (invoice amount) and the decimal points and commas that appear in the amount columns are generated by the program.

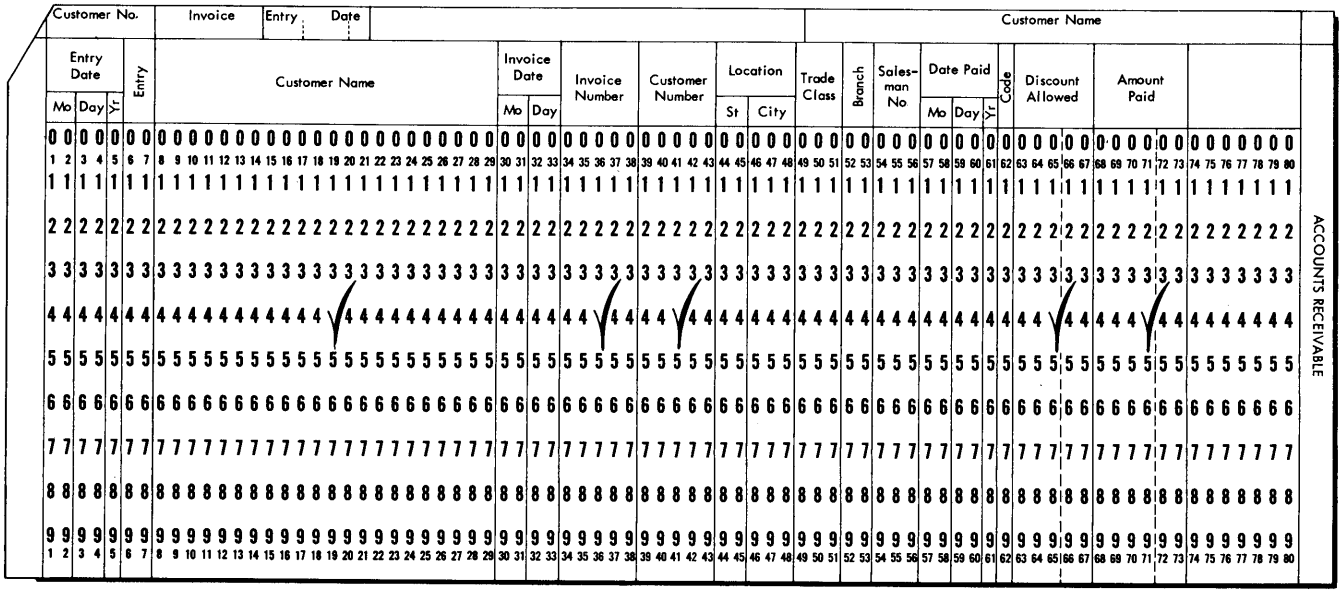


Figure 12. Card Format for Detail Printing

ACCOUNTS RECEIVABLE REGISTER

	CUSTOMER NUMBER	CUSTOMER NAME	INVOICE NUMBER	DISCOUNT ALLOWED	AMOUNT PAID	INVOICE AMOUNT
	09285	COVENTRY OIL	11509	5.93	587.02	592.95
	09285	COVENTRY OIL	12292	9.51	941.46	950.97
	21217	CRESCENT BELT CO	11511	3.36	332.27	335.63
	21217	CRESCENT BELT CO	12263	1.22	120.53	121.75
	29031	DENNIS MFG CO	11615	4.40	435.72	440.12
	29031	DENNIS MFG CO	11676	7.22	715.00	722.22
	29134	DEPOSIT DIST CO	09689	6.45	638.95	645.40
	29134	DEPOSIT DIST CO	11605	2.72	268.97	271.69
	29134	DEPOSIT DIST CO	12234	5.59	553.74	559.33
	30327	GENERAL MFG CO	09352	11.21	1,116.03	1,127.24
Print Positions	7-11	16-37	45-49	55-60	63-70	73-80

Figure 13. Form Layout for Detail Printing

Figure 14A, block diagram, and Figure 14B, decision table, give a general description of the program. In the description, the program proceeds to assemble a line of information for printing, after reading a card. The assembled line is printed and a test is made to determine if the end of the form (assuming a continuous form is used) has been reached. If an end-of-form condition is indicated, a skip to the first line position of the next form must be initiated. After printing of one card is completed, the test for last card is performed. If the Last Card indicator is on, the program halts; if it is off, the next card is read and the steps of the diagram are repeated.

The Coding sheets (Figure 15) contain the statements that make up the program for detail printing this example. Line 01 defines the starting address of storage assignment. Lines 02-08 define the print area, lines 09-14 define the card read area, line 15 reserves a work area, line 16 establishes a constant to be used by the program, and lines 17-25 and 01-11 (second coding sheet) constitute the program itself.

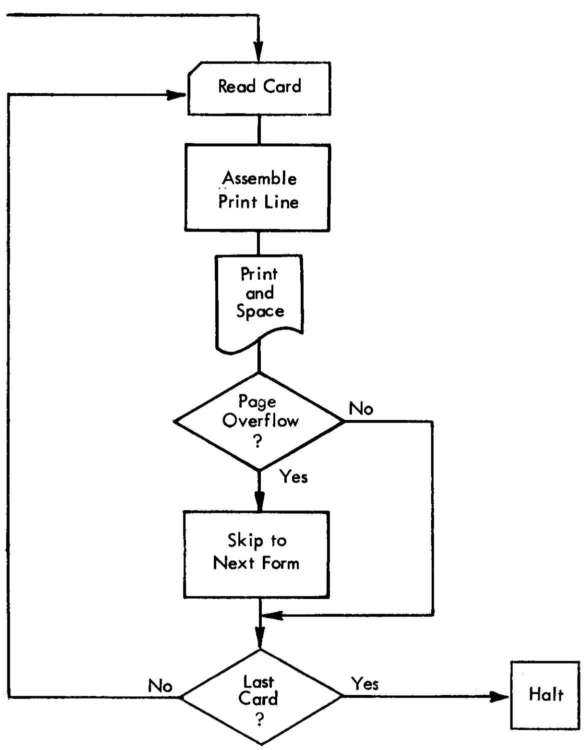


Figure 14A. Block Diagram for Detail Printing

	** START			
1	HAS LAST CARD BEEN PROCESSED?	N	N	Y
2	CARRIAGE OVERFLOW INDICATOR ON?	N	Y	•
3	READ CARD	X	X	
4	PRINT	X	X	
5	SKIP TO NEXT FORM		X	
6	GO TO START	X	X	
7	HALT			X

Figure 14B. Decision Table for Detail Printing

Page No. 01 of 02

Line	Label	Operation	OPERAND
01		ORG	401 BEGIN THE ASSIGNMENT OF STORAGE WITH POS 401
02	PRLINE	DA	2x80,G DEFINE PRINTING AREA
03	CUSNOL		11 SUBFIELDS: CUSTOMER NUMBER
04	CUSNAL		37 CUSTOMER NAME
05	INVNOL		49 INVOICE NUMBER
06	DISCL		60 DISCOUNT ALLOWED
07	AMTPDL		70 AMOUNT PAID
08	INAMTL		80 INVOICE AMOUNT
09	RDAREA	DA	1x73,G DEFINE CARD READING AREA
10	CUSNAC		8,29 FIELDS: CUSTOMER NAME
11	INVNOC		34,38 INVOICE NUMBER
12	CUSNOC		39,43 CUSTOMER NUMBER
13	DISCC		63,67 DISCOUNT NUMBER
14	AMTPDC		68,73 AMOUNT PAID
15	ACCUM	DCW	#6
16	CTRLWD	DCW	@bbb0bb@
17	START	R	1,RDAREA READ A CARD
18		MLC	CUSNOC,CUSNOL ASSEMBLE PRINT LINE
19		MLC	CUSNAC,CUSNAL
20		MLC	INVNOC,INVNOL
21		MLCWA	CTRLWD,DISCL
22		MCE	DISCC,DISCL
23		MLCWA	CTRLWD,AMTPDL
24		MCE	AMTPDC,AMTPDL
25		ZA	DISCC,ACCUM

Page No. 02 of 02

Line	Label	Operation	OPERAND
01		A	AMTPDC,ACCUM
02		MLCWA	CTRLWD,INAMTL
03		MCE	ACCUM,INAMTL
04		CC	S
05		W	PRLINE PRINT A LINE
06		BCV	OFLO TEST FOR CHANNEL TWELVE
07		B	LSTCD BRANCH TO LSTCD
08	OFLO	CC	A SKIP TO FIRST LINE NEXT FORM
09	LSTCD	BLC	HALT TEST FOR LAST CARD
10		B	START BRANCH TO START
11	HALT	H	END PROGRAM

Figure 15. Coding Sheets for Detail Printing

Line 01 starts the assignment of storage at position 401 with an origin statement.

Origin Operation. The ORG statement provides the programmer with a means of controlling storage assignment. The address specified in the operand field is the starting address. If this statement were not present at the beginning of the program, the processor would automatically start assignment with position 210. An ORG statement can be inserted at any point in a program if the programmer desires to assign subsequent addresses beginning at the address specified in the operand field.

Suppose a field such as customer number appears in the read area and the print area. We could use CUSNO as the label for the read area field and CUSN for the label for the print area field, but we would have to remember which of these we assigned to which and this is difficult to do in a large program. Of course, we could continually check ourselves by referring to our last use of the label but this procedure is time-consuming and subject to error. For this reason we use a technique that involves a mnemonic suffix to the label.

Unique Labels with Suffixes. Where the names of two or more fields are similar but must be designated by unique labels, it is only natural to expect that the label associated with each will be similar. For example: the field "customer number" which is located in both the read and print areas may be expected to have the label CUSNO. However, no two fields can be designated by one label.

The customer number field in the read area must be distinguishable from the customer number field in the print area. By using the technique of adding a c suffix for *c*ard read and an *l* suffix for print *l*ine to the base CUSNO, we develop

CUSNOC for customer number in the *read* area
CUSNOL for customer number in the *print* area

This technique of suffixing can be extended to yield other advantages; for once we have established a suffix for an area, we can give it to all fields in that area. For example,

CUSNOL for *customer number* in the print area
CUSNAL for *customer name* in the print area
INVNOL for *invoice number* in the print area
DISCL for *discount allowed* in the print area

Besides distinguishing one label from the other, the common suffix provides an easy clue to, and identification of, an entire group of labels. A single-digit suffix has been used in our examples. Two-digit suffixes can be used if they are more meaningful or prove more descriptive.

Line 15 causes a 6-position storage area to be reserved. A word mark is set in the leftmost position of this field. The # symbol precedes the number that indicates how many blank positions are to be defined. The field established by this statement is used in the program to accumulate "invoice amount."

Line 16 defines the 8-position alphameric constant that will be used by the program to perform its print edit function. The print edit function includes zero suppression and the insertion of decimals and commas in the amount fields. The alphameric constant must be preceded by and followed by an @ symbol. The character b is used to indicate blank positions within the constant.

Line 17 causes card columns 1-73 to be read into core storage.

Line 18 is an MLC operation which moves the A-field data to the B field. The "customer number" is moved from the read area to the print area; the A-field word mark governs the move since no word mark was set in the B field (print area).

> ***Move Characters to A or B Word Mark Operation.*** The A and B operands in an MLC operation represent the addresses of the respective A and B fields. The data is moved, starting with the rightmost digit of the field and continuing to the left until a word mark is encountered in either field (A or B). Word marks in either field remain unchanged. The A-field data remains unchanged by the move operation.

Line 19 causes "customer name" to be moved by an MLC operation from the read area to the print area. *Line 20* causes the "invoice number" field to be moved from the read area to the print area.

Line 21 is an MLCWA operation that moves the data from the A field to the B field. The constant (symbolized in our example by CTRLWD) is set up in the print area preliminary to performing an edit instruction.

> ***Move Characters and Word Mark from A Field Operation.*** In an MLCWA operation, the A-field word mark is used to terminate the movement of data from the A field to the B field. Unlike the MLC operation in which either word mark terminates the move, in the MLCWA only the A-field word mark can stop the movement of data. In addition, the A-field word mark is transmitted to the B field and word marks contained in the B field prior to the move are erased.

Line 22 is an MCE operation for moving data from the A field (discount allowed) to the B field and for modifying the data by the contents of the B field (control word). This control word was previously set up in the print area by line 21.

Move Characters and Edit Operation. The MCE operation, often referred to as the "editing" operation, is provided so that amount fields can be punctuated for readability. Dollar signs, decimals, and commas which are not punched in the card may be produced by this instruction. In addition, insignificant leading zeros can be suppressed and special characters inserted.

In an MCE operation word marks are required in the leftmost position of both fields (A and B). When the instruction is executed, the B-field word mark is removed, the data from the A field is inserted into the character positions occupied by blanks or zeros in the B field, and the high-order zeros in the data are replaced with blanks.

To demonstrate the use of the control word with an edit instruction, let us assume that the B field contains

b̲,bb0.bb (control word)

and that the A field contains:

0̲0003

After the MCE instruction is executed, the field will contain:

bbbbb.03

Blanks in the control word are replaced with characters located in corresponding positions of the A field. The zero in the control word is used for zero suppression and is replaced by a corresponding character from the A field. The rightmost zero in the control word is the rightmost limit of zero suppression.

The comma in the control word remains undisturbed unless zero suppression takes place and no significant numerical characters are to the left of the comma. The period is retained in the edited data in the same position it occupied in the control word. The characters from the word mark to the rightmost blank in the example shown are considered to be the body of the control word.

If the control word looked like this:

$̲b,bb0.bbCR**

the characters from the word mark to the rightmost blank would be the *body* of the control word and the remaining characters would be the *status* portion of the control word. The dollar sign in the body portion is left in the same position in the edited data.

The CR in the status portion remains at the completion of editing if the data edited was negative. If the data was positive, the CR is deleted. If the status portion contained a minus sign in place of the CR, the minus sign would be treated in the same way. Asterisks are usually used to indicate a class of totals and they remain after editing.

A few examples of MCE operations follow.

A Field	B Field (before)	B Field (after)
0̲0123	b̲,bb0.bb	bbb1.23
0̲00123	b̲,bb0.bb	bbb1.23
1̲2345	b̲,bb0.bb	b123.45
1̲23456	b̲,bb0.bb	1,234,56
1̲2345	$̲b,bb0.bb	$bb123.45
1̲23456	$̲b,bb0.bb	$1,234.56
0̲0123 (negative)	b̲,bb0.bb–	bbbb1.23–
0̲0123	b̲,bb0.bb–	bbbb1.23b
0̲0123 (negative)	b̲,bb0.bbCR	bbbb1.23CR
0̲0123	b̲,bb0.bbCR	bbbb1.23bb
0̲0123	b̲,bb0.bb**	bbbb1.23**

Note that the dollar sign and asterisks are left undisturbed in their positions in the B field. However, the CR or – (minus) sign is deleted when the field is positive.

Lines 23-24 move the control word to the "amount paid" print area; amount paid is then moved and edited in the same area. This editing operation is similar to the one performed by lines 21-22.

Line 25 resets to zeros the field labeled ACCUM before adding the "discount allowed" to that field.

Line 01 (page 02) adds the "amount paid" to the ACCUM field.

Lines 02-03 set up the control word and move and edit the "invoice amount" from the ACCUM field to the print area.

Line 04 is a control carriage operation which conditions the carriage to double space after the next print operation.

Control Carriage Operation (Forms Skipping). The CC operation is used to control carriage spacing and line skipping on a report form. In this type of statement, only the operation code (CC) and a d-modifier are required (no addresses need be specified).

Any of the following six d-modifiers can be used to control vertical spacing of the form

d-modifier	Number of Spaces	
J	1	Immediate
K	2	
L	3	
/	1	Delayed
S	2	
T	3	

Immediate means that the carriage will be spaced at the time the d-modifier is encountered in the program; *delayed* means the space will be taken after the next print instruction is encountered.

It is essential when using either type of spacing to remember that normally the printer automatically skips one space after printing. If an immediate space is used, the carriage spaces as many lines as are called for by the d-modifier. Thus, if a J is written, one space is skipped. After spacing, the line is printed and then the form automatically skips one space (when the W operation code is used to initiate printing).

If delayed spacing is called for by a diagonal (/), S, or T, the total number of spaces skipped after printing will be the number of spaces prescribed including the one that normally occurs. Thus, a control carriage (CC) operation with a d-modifier of diagonal (/) has no net effect, since only one space is specified and the carriage automatically skips one space. The d-modifiers S and T call for two and three spaces, and have the net effect of increasing the skipping by one or two additional spaces.

Whenever a control carriage operation is used for forms skipping, a control tape must be prepared for that particular form. The control tape has twelve columns of positions (channels) indicated by lines (see Figure 16). Holes are punched in each channel throughout the length of the tape, which is ordinarily the same length as one form of a continuous form. These holes indicate the end of the form or places in the form where skipping is to start or stop.

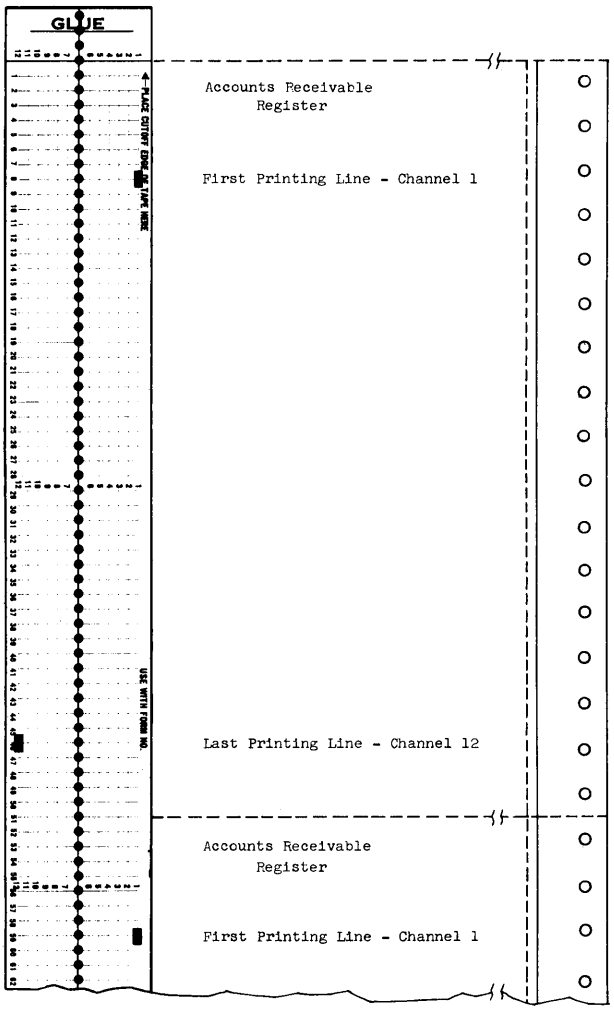


Figure 16. Continuous Form and Control Tape for Detail Printing

The d-modifier is used in the CC statement to control vertical skipping of the form; i.e., to specify which channel of the tape is to stop skipping.

d	Immediate Skip to	d	After Printing Skip to
1	Channel 1	A	Channel 1
2	Channel 2	B	Channel 2
3	Channel 3	C	Channel 3
4	Channel 4	D	Channel 4
5	Channel 5	E	Channel 5
6	Channel 6	F	Channel 6
7	Channel 7	G	Channel 7
8	Channel 8	H	Channel 8
9	Channel 9	I	Channel 9
0	Channel 10	?	Channel 10
#	Channel 11	.	Channel 11
@	Channel 12	¤	Channel 12

The punched tape is looped into a continuous band and mounted on the tape reading mechanism, as shown in Figure 17. A pin feed associated with this mechanism advances the tape through the carriage in synchronism with the printed form. The effect is exactly the same as if the control holes (channel punches) were punched along the edge of each form.

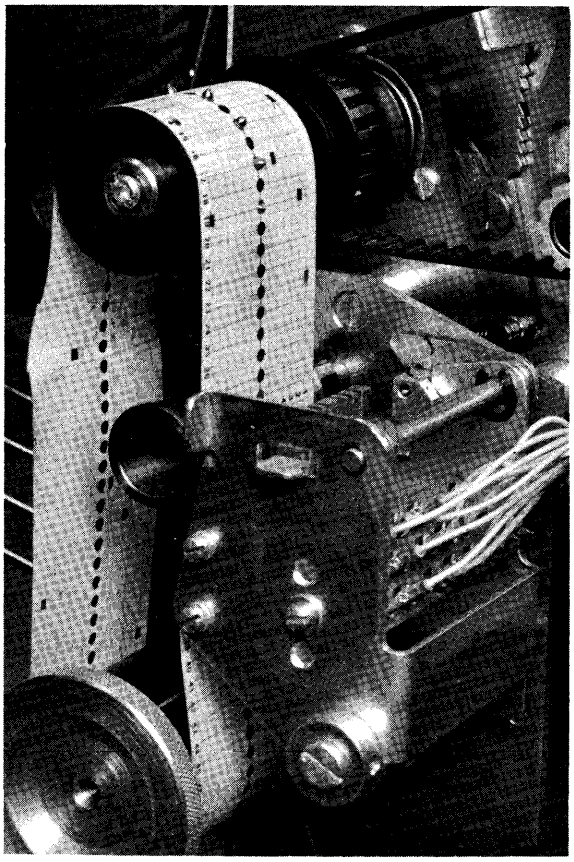
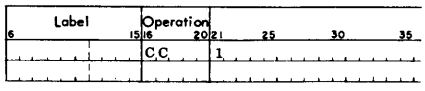
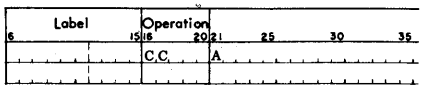


Figure 17. Tape Reading Mechanism of the Tape-Controlled Carriage

If the statement given is



an immediate skip to channel 1 of the control tape is initiated, however, the statement



causes a skip to channel 1 to be initiated after the next line is printed.

Line 05 prints the assembled print data starting with position 401, which was assigned by the processor as a result of the Origin statement that appears in the beginning of our program.

Write a Line Operation. The code for a Write a Line operation is W. The operand specifies the leftmost position of the print area, which must have an address ending in 01. The addressed core storage position is printed in print position 1; subsequent higher-numbered core storage positions are printed in print position 2; etc. The position immediately to the left of the group mark with a word mark is the highest-numbered storage position that is printed. After a line is printed, the carriage automatically skips a space. The WS operation code ***(Write a Line and Suppress Spacing)*** can be used in place of the W operation code if it is desired to suppress the automatic spacing after printing.

Line 06 is a BCV operation which is used to sense the end of a form and to branch to the instruction that initiates a forms skip.

Branch on Carriage Channel 12 Operation. The BCV operation tests the carriage channel 12 indicator. This indicator is turned on when a hole is sensed in channel 12 of the carriage control tape and is turned off when any other channel hole is sensed. If the indicator is on when tested, the program branches to the address of the instruction specified by the A operand. If it is off, the program goes to the next instruction.

Line 07 causes an unconditional branch to the address of the instruction labeled LSTCD.

Line 08 causes the form to skip to a new page. The end of the previous page was sensed by line 06. The

continuous form and carriage control tape are shown in Figure 16. In this problem we have chosen channel 1 to represent the first printing line of a page. Therefore, a skip to channel 1 is initiated when channel 12 is sensed.

Line 09 tests for last card and initiates a branch to Halt after the last card is processed.

Line 10 causes an unconditional branch to the address of the instruction labeled START.

Line 11, labeled HALT, causes the program to end after the last card is processed and the last line has been printed.

Programming Example 2. Detail Printing with Three Classes of Totals

Detail printing with three classes of totals is similar in many respects to the detail printing program explained previously. For that reason, we will explain only those concepts that are different or new.

Figure 18 shows the fields, indicated by check marks (✓), that are to be detail-printed on the report form. The order in which the fields appear on the report form is shown in Figure 19. Note that "item amount" is printed in the amount column for each detail line, "total amount by sub ledger" is printed each time the sub ledger account number changes, "total amount by general ledger" is printed each time the general ledger account number changes, and "total amount by department" is printed each time the department number changes. Therefore, we have three classes of totals.

1. Total by sub ledger
2. Total by general ledger
3. Total by department

After each total the form advances one extra space. Class one, two, and three totals are identified by one, two, and three asterisks, respectively.

Figure 20A, block diagram, and Figure 20B, decision table, describe this program in general. The program starts with the resetting of the accumulators to zeros, the turning off of switch 2, and the reading of a card. The accumulators referred to are not true accumulators. They are merely areas in core storage

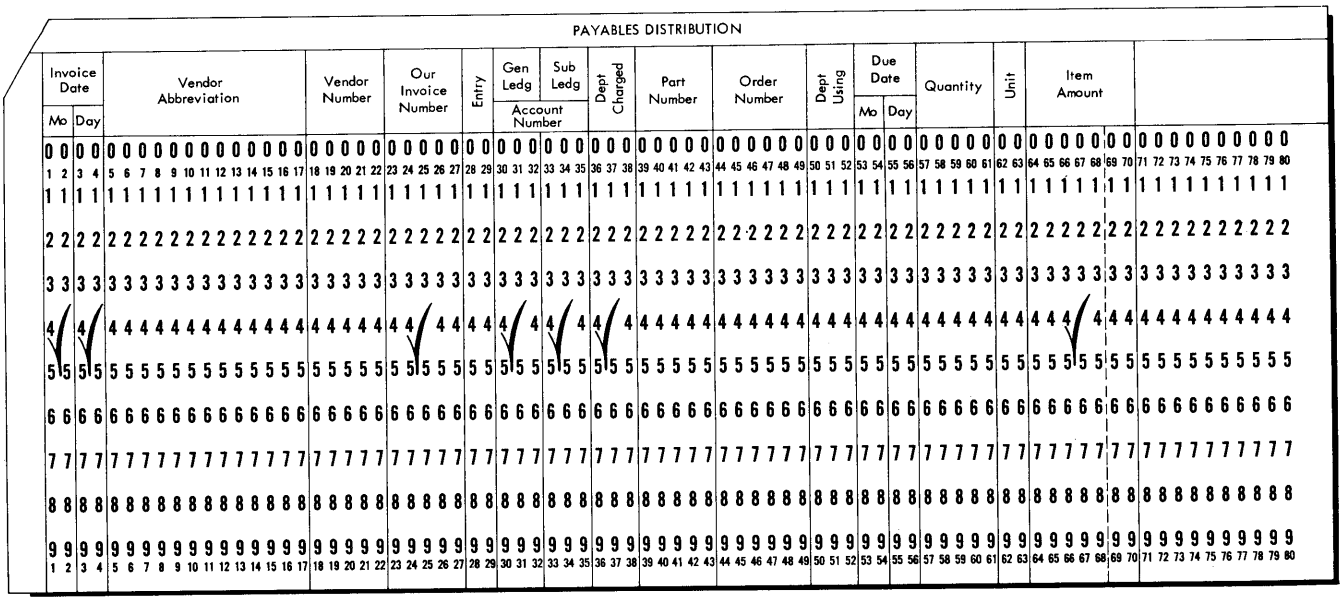


Figure 18. Card Format for Detail Printing with Three Classes of Totals

EXPENSE DISTRIBUTION
By Department or Branch

Dept	Account No		Our Invoice Number	Date		Amount
	Gen Ledg	Sub Ledg		Mo	Day	
82	431	112	12066	12	10	300.00
82	431	112	12153	12	28	300.00
						600.00*
82	431	113	12066	12	10	150.00
82	431	113	12066	12	10	150.00
82	431	113	12066	12	10	125.00
82	431	113	12153	12	28	150.00
82	431	113	12153	12	28	150.00
82	431	113	12153	12	28	125.00
						850.00*
82	431	114	12066	12	10	50.00
82	431	114	12066	12	10	75.00
82	431	114	12066	12	10	50.00
82	431	114	12153	12	28	50.00
82	431	114	12153	12	28	50.00
82	431	114	12153	12	28	75.00
						350.00*
82	431	520	12149	12	28	360.43
						360.43*
82	431	700	12082	12	14	2.25
						2.25*
82	431	750	12003	12	01	100.00
						100.00*
82	431	810	12112	12	18	70.20
						70.20*
82	431	850	12043	12	07	24.75
						24.75*
						2357.63**
82	432	841	12151	12	28	1792.86
						1792.86*
						1792.86**
						4150.49***

Print Positions	1-3	5-7	9-11	13-17	19-20	22-23	25-36

Figure 19. Form Layout for Detail Printing with Three Classes of Totals

designated for accumulating totals. The steps performed thus far by the program may be termed *initializing steps,* so called because they are performed only once during the program and are used to get the program started.

As indicated in the next square in the block diagram, the control fields (department number, general ledger number, and sub ledger number) are read from the first card of the deck and from each card that follows a total and are stored in a work area. These numbers in the work area will be used by the program to recognize changes in control field numbers. Switches 1 and 2 will be used to remember the classes of totals taken. After a total is taken, the particular switch will be reset to its off status.

The next step is to assemble a detail line and print data. The next step is to test for last card. If the final card has been processed, three classes of totals will be required. Before the totals are taken, Switch 3 is turned on to remember the last card condition. After the totals are taken, Switch 3 is tested to end the program. The item amount will be added into each of the three accumulators. After the test for last card, another card is read. When this is done, the three control fields in the card and the three control fields in work storage are compared one at a time. If the fields are found to be equal, no totals are required and the program can loop back to assemble and print another detail line. If any control field is found to be unequal, a total or totals will be required. To properly set the switches, it is important to start comparing the control fields with the most significant control field (department number) and to end with the least significant control field (sub ledger number). The first unequal comparison does not turn on a switch, whereas the second and third unequal comparisons turn switches 2 and 1 on, in that order.

Before a class one total can be printed, the print area must be cleared to blanks. This is required because the area from which the detail line is printed is the same area from which the total lines are printed. If this area is not cleared, portions of the detail line data will be printed erroneously along with the totals.

The class one total is assembled in the print area, is printed, and its associated accumulator is reset to zeros. If switch 1 is found to be on, this means that only a class one total is required, and the program will go back to the process of assembling another detail line. If switch 1 is off, a class two total is assembled, printed, and its associated accumulator reset to zeros. Switch 2 will be tested. If it is found to be on, which means that class one and class two totals are required, the program returns to assemble another detail line. If switch 2 is found to be off, a class three total is printed and its associated accumulator reset to zeros. The program will return to assemble the next detail line, unless switch 3 is on.

In our example you will find that a switch is merely a position of core storage designated SW; when a word mark is placed in that position, the switch is on. When the word mark is removed, it has the effect of turning it off.

The four program coding sheets for detail printing with three classes of total are shown in Figure 21. These are used to define storage areas, work areas, constants, and the program instructions.

Line 01 starts the assignment of storage with position 401.

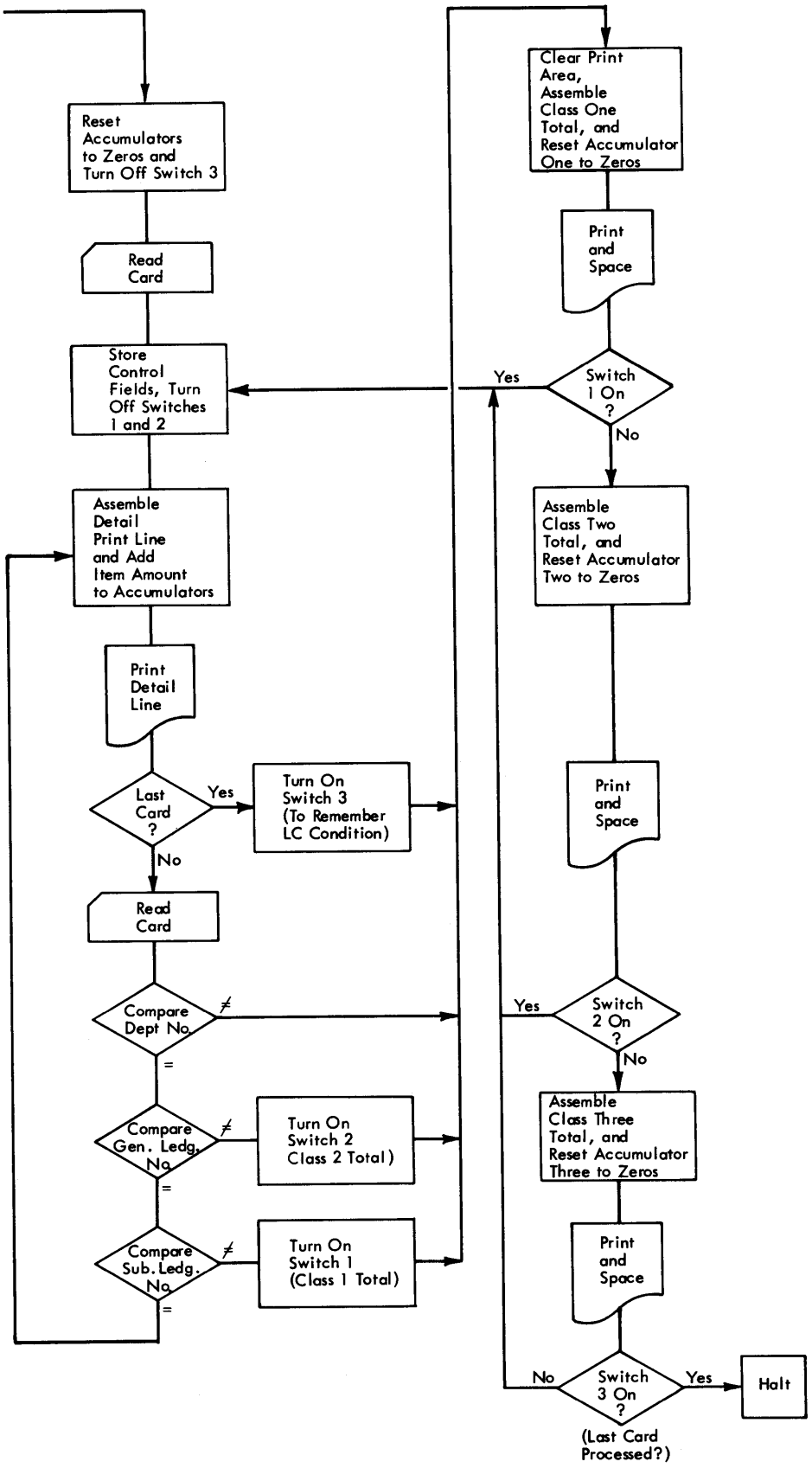


Figure 20A. Block Diagram for Detail Printing with Three Classes of Totals

	** TEST CONTROL CHANGE											
1	DEPT. NO. READ IN VS DEPT. NO.											
	PREVIOUS CARD	≠	=	=								
2	GEN LEDG NO. READ IN VS GEN LEDG											
	NO. PREVIOUS CARD	·	≠	=								
3	SUB LEDG NO. READ IN VS SUB LEDG											
	NO. PREVIOUS CARD	·	·	≠								
4	PRINT DEPT TOTAL	X										
5	PRINT GEN LEDGER TOTAL	X	X									
6	PRINT SUB LEDGER TOTAL	X	X	X								

Figure 20B. Decision Table for Detail Printing with Three Classes of Totals

Lines 02-09 describe the print area and *lines 10-17* describe the card read area. *Line 18* defines an edit control word. *Lines 19-21* reserve three 3-position work fields for the control field numbers. *Lines 22-24* set up three 8-position accumulators. *Line 25* (page 1) and *lines 01-02* (page 2) reserve three positions of storage to designate the three switches to be used by the program.

Lines 03-05 are Subtract (S) operations which reset the three accumulators by subtracting each from itself. The result is the same as if we had performed the following arithmetic:

```
  123
 -123
 ---
  000
```

Subtract Operation. In a subtract operation, the data addressed by the A operand is subtracted from the data addressed by the B operand, digit by digit, starting in the rightmost position. The result that is developed replaces the data previously stored at the B address.

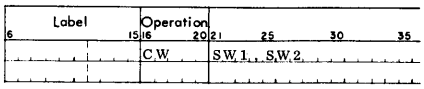
The subtract operation is terminated by the word mark in the leftmost position of the B field. If the A field is smaller than the B field, both fields require a word mark.

Line 06, a CW statement, turns off switch 3 by removing the word mark.

Clear Word Mark Operation. The CW statement clears the word mark at the address specified by the A operand or the word marks specified by the A and B operands without disturbing the data.

Line 07 reads a card. *Lines 08-10* are MLC statements which move the three control fields to working storage.

Lines 11-12 turn off switches 1 and 2 by removing word marks. The CW statements on lines 11 and 12 could have been combined and written as a single statement, thus:



In this case, the word marks at both addresses (SW1 and SW2) would be cleared by a single statement. In effect, removing these word marks is equivalent to turning off these switches.

Lines 13, 14, and 15 are MCS statements which move the three control fields – department number, general ledger number, and sub ledger number – to the print area and eliminate the high-order zeros from each field.

Move Characters and Suppress Zeros. The MCS statement is similar to the MLC statement in certain respects, i.e., it is designed to move data. However, zeros to the left of the leftmost significant digit of the data are eliminated and replaced by blanks. Removal of nonsignificant zeros is considered an editing function. The word mark in the leftmost position of the A field defines the size of the field; B field word marks are erased.

Lines 16-18 are MLC statements which move the invoice number, month, and day to the print area. *Line 19* moves the edit control word to the amount field (print area). Any asterisks in the amount field from a previous total will be eliminated by the move.

Page No. 01 of 04

Line	Label	Operation	OPERAND
01		ORG	401 START STORAGE ASSIGNMENT AT POS. 100
02	PRLINE	DA	1,36,G DEFINE PRINT LINE
03	DEPTL		3 FIELDS: DEPARTMENT NO
04	GENL		7 GENERAL LEDGER NO
05	SUBL		11 SUB LEDGER NO
06	INVNOL		17 INVOICE NO
07	MOL		20 MONTH
08	DAYL		23 DAY
09	AMNTL		36 AMOUNT
10	RDAREA	DA	1,70,G DEFINE CARD READING AREA
11	MOC		1,2 FIELDS: MONTH
12	DAYC		3,4 DAY
13	INVNOC		23,27 INVOICE NO
14	GENC		30,32 GENERAL LEDGER NO
15	SUBC		33,35 SUB LEDGER NO
16	DEPTC		36,38 DEPARTMENT NO
17	AMNTC		64,70 AMOUNT
18	CTRLWD	DCW	@bbbbbb0.bbbbbb@ EDIT CONTROL WORD
19	DEPTW	DCW	#3 DEFINE WORK AREAS
20	GENW	DCW	#3
21	SUBW	DCW	#3
22	ACCUM1	DCW	#8 DEFINE ACCUMULATORS
23	ACCUM2	DCW	#8
24	ACCUM3	DCW	#8
25	SW1	DC	#1 DEFINE SWITCHES

Page No. 02 of 04

Line	Label	Operation	OPERAND
01	SW2	DC	#1
02	SW3	DC	#1
03	START	S	ACCUM1,ACCUM1 RESET ACCUMULATORS TO ZEROS
04		S	ACCUM2,ACCUM2
05		S	ACCUM3,ACCUM3
06		CW	SW3 TURN SWITCH 3 OFF (LC CONDITION)
07		R	1,RDAREA READ A CARD
08	STORE	MLC	DEPTC,DEPTW STORE CONTROL FIELDS
09		MLC	GENC,GENW
10		MLC	SUBC,SUBW
11		CW	SW1 TURN SWITCHES ONE AND TWO OFF
12		CW	SW2
13	ASSEM	MCS	DEPTC,DEPTL ASSEMBLE DETAIL PRINT LINE
14		MCS	GENC,GENL
15		MCS	SUBC,SUBL
16		MLC	INVNOC,INVNOL
17		MLC	MOC,MOL
18		MLC	DAYC,MOC
19		MLCW	ACTRLWD,AMNTL
20		MCE	AMNTC,AMNTL-3
21		A	AMNTC,ACCUM1 ADD ITEM AMOUNT TO ACCUMULATORS
22		A	AMNTC,ACCUM2
23		A	AMNTC,ACCUM3
24		W	PRLINE PRINT DETAIL LINE
25		BLC	SETSW3 TEST FOR LAST CARD

Figure 21. Coding Sheets for Detail Printing with Three Classes of Totals

Page No. 03 of 04

Line	Label	Operation	OPERAND
01		R	1,RDAREA READ A CARD
02		C	DEPTC,DEPTW COMPARE DEPARTMENT NO
03		BU	TOTAL
04		C	GENC,GENW COMPARE GENERAL LEDGER NO
05		BU	SETSW2
06		C	SUBC,SUBW COMPARE SUB LEDGER NO
07		BE	ASSEM
08		SW	SW1 TURN SWITCH ONE ON
09		B	TOTAL
10	SETSW2	SW	SW2 TURN SWITCH TWO ON
11	TOTAL	CS	AMNTL CLEAR PRINT AREA
12		MLCWA	CTRLWD,AMNTL ASSEMBLE TOTAL CLASS ONE
13		MLC	@,*,b,b,@,AMNTL
14		MCE	ACCUM1,,AMNTL-3
15		S	ACCUM1,,ACCUM1 RESET ACCUMULATOR ONE TO ZEROS
16		CC	2
17		W	PRLINE
18		BW	STORE,,SW2
19		MLCWA	CTRLWD,AMNTL ASSEMBLE TOTAL CLASS TWO
20		MLC	@,*,*,b,@,AMNTL
21		MCE	ACCUM2,,AMNTL-3
22		S	ACCUM2,,ACCUM2 RESET ACCUMULATOR TWO TO ZEROS
23		CC	2
24		W	PRLINE
25		BW	STORE,,SW2

Page No. 04 of 04

Line	Label	Operation	OPERAND
01		MLCWA	CTRLWD,AMNTL ASSEMBLE TOTAL CLASS THREE
02		MLC	@,*,*,@,AMNTL
03		MCE	ACCUM3,,AMNTL-3
04		S	ACCUM3,,ACCUM3 RESET ACCUMULATOR THREE TO ZEROS
05		CC	2
06		W	PRLINE
07		BW	HALT,,SW3 LAST CARD PROCESSED ?
08		B	STORE
09	SETSW3	SW	SW3 TURN SWITCH 3 ON
10		B	TOTAL
11	HALT	H	START

Figure 21. Coding Sheets for Detail Printing with Three Classes of Totals (contd.)

Line 20 moves the amount to the print area. The B operand is address-adjusted so that the position immediately to the left of the printing positions which print the asterisks can be addressed. In our example, if the position addressed by AMNTL, which is the location of the rightmost asterisk, had been assigned address 688 by the processor, then AMNTL minus 3 would be 685, the position to the left of the asterisks.

Address Adjustment of an Operand. Address adjustment is used by the programmer to tell the processor to arithmetically adjust the address of an operand. The programmer indicates address adjustment should be performed by writing a plus (+) or minus (–) sign after the symbol. The processor then assigns an address that is a given number of positions away from the specific address, the specific address being the address associated with the label.

If the storage position 1000 has been assigned to the symbol ALPHA and the programmer writes two operands

ALPHA + 40
ALPHA − 30

the processor then assigns to these operands the addresses

1040
970.

Lines 21-23 add the item amount to each of the three accumulators. *Line 24,* a Write a Line operation (W), prints a detail line from the detail data in the print area. *Line 25* tests for last card. *Line 01* (page 3) reads a card into the read area.

Line 02, a Compare operation (C), causes the contents of the A and B fields (department numbers) to be compared.

Compare Operation. In a Compare statement, the rightmost position of the A and B fields is addressed by the A and B operands, respectively. The data in the fields, starting with the rightmost position, is compared until a word mark is encountered in either field. If the B field is longer than the A field, in which case a word mark will be encountered in the A field first, an unequal compare results. If the data in the A and B fields is the same, the equal indicator is turned on. If they are not the same, either the high or low indicator is turned on as well as the unequal indicator.

When the value of the B field data is less than the value of the A field data (B < A), the low indicator turns on. When the value of the B field data is greater than the value of the A field data (B > A), the high indicator turns on.

Comparing is not restricted to numerical fields. Alphabetic and alphameric fields can also be compared. The fields compared must have the same bit configuration to be equal. For this reason, comparison of a plus zero and a minus zero results in the unequal indicator being turned on.

To determine high or low comparisons, the programmer should remember that the character rank (collating sequence) from lowest to highest is

1. blank
2. special characters
3. letters (A-Z)
4. numbers (0-9)

The following comparisons of A and B fields show the indicators that are set as a result.

		Indicators		
A Field	B Field	Unequal	High	Low
01	02	X	X	
02	01	X		X
J	K	X	X	
1J	1K	X	X	
K	J	X		X
1K	1J	X		X
ACA	ABA	X		X
123	124	X	X	

Line 03, a BU operation, tests the Branch Unequal indicator. If the department numbers are unequal, the program will branch to the instruction labeled TOTAL.

Branch on Unequal Compare Operation. If the Branch Unequal indicator is on when tested by this instruction, the program branches to the address specified by the operand.

Lines 04-05 function similarly to the previous two lines. In this case, however, the fields compared contain general ledger numbers and the branch is to the instructions labeled SETSW2.

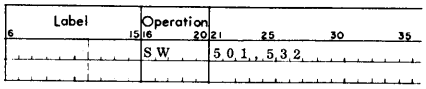
Line 06 compares the sub ledger numbers, and *line 07,* a BE operation, causes the program to branch to the instruction labeled ASSEM, if the sub ledger numbers are equal.

Branch on Equal Compare Operation. If the Branch Equal indicator is on when tested by this instruction, the program branches to the address specified by the operand.

Line 08 turns switch 1 on by setting a word mark in a position of storage. The effect of line 08 is to set a switch that can be tested by another program step at a later time.

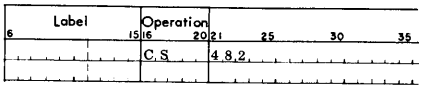
Set Word Mark Operation. This operation causes a word mark to be set in the position of core storage designated by the A operand.

The character is unchanged in a position where a word mark is set. Two word marks can be set by a single SW operation, provided both the A operand and B operand contain an address. The following statement will cause word marks to be set in storage positions 501 and 532.

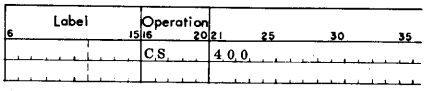


Line 09 causes an unconditional branch to the instruction labeled TOTAL (start of total routine). *Line 10* turns switch 2 on by setting a word mark at the address symbolized by SW2. *Line 11,* a CS operation, clears the print area of all data and resets the area to blanks.

Clear Storage Operation. The CS operation causes storage to be set to blanks and word marks to be removed, starting at the address specified by the operand and continuing through lower-numbered positions until an XX00 address is encountered, i.e., an address that ends in zero-zero. The following statement



results in clearing storage positions 482 through 400, whereas the statement



would clear storage position 400 only. As many as 100 positions can be cleared by a single CS statement.

Line 12 moves the edit control word to the amount field in the print area. *Line 13* moves a single asterisk to the print area to identify the total as a class one total. Since the A operand is a literal (@*bb@), it is preceded and followed by an at (@) sign. *Line 14* moves the class one total to the print area and edits it. *Line 15* resets accumulator 1 to zeros. *Line 16* conditions the carriage so that it skips two spaces after the next print instruction. *Line 17* causes the class one total to be printed. *Line 18,* a BW operation, tests switch 1 which is represented by a word mark and branches to the instruction labeled STORE if the word mark is present. In effect, this is the instruction that tests switch 1.

Branch on Word Mark Operation. The BW operation tests the position specified by the B operand for a work mark and, if the word mark is present, branches to the address of the instruction specified by the A operand.

Lines 19-25 are similar to lines 12-18. They assemble and print a class-two-total line identified by two asterisks.

Lines 01-06 (page 4) assemble and print a class three total identified by three asterisks.

Line 07 tests switch 3 for a word mark. If the word mark is present (meaning that the last card has been processed), the program branches to the instruction labeled HALT.

Line 08, a B operation, is used to initiate an unconditional branch to the instruction labeled STORE.

Lines 09-10 turns on switch 3 to remember the last card condition and branches to the start of the total routine which is labeled TOTAL.

Line 11 ends the program after the last card has been processed.

Programming Example 3. Reading/Punching and Multiplication by Repetitive Addition

This programming example introduces two new operations. In this example an amount field is read from each card. The discount is calculated, by repetitive addition, at a certain discount rate (.07) and the discount is punched. The card format is as follows:

Card columns	1-4 discount (XX.XX)
	5-9 amount (XXX.XX)

When multiplication is performed using programmed repetitive addition, the problem

$$\begin{array}{r} 123.45 \\ \times\ .03 \\ \hline 3.7035 \end{array}$$

is accomplished by adding the number 123.45 to a specific area of storage three times, thus

```
  00000
+12345
  12345
+12345
  24690
+12345
  3.7035  (result)
```

Likewise, the multiplication:

```
  123
×.23
28.29
```

is done by adding the number 123 to an area three times. The multiplicand is then shifted left one position, and added two more times.

```
   0000
+  123
   0123
+  123
   0246
+  123
   0369
+ 123
   1599
+ 123
   2829  (result)
```

Programmed division is performed in a similar manner, except that repetitive subtraction is used in place of repetitive addition.

Figure 22A, block diagram, and Figure 22B, decision table, describe this program. In them the first functions shown are reading a card and testing for last card. The next is the setting of the repetitive-addition count to 7 and the resetting of the accumulator to zeros. The accumulator is an area of storage used to accumulate discount amount.

The discount amount is calculated by adding the amount to the accumulator the number of times designated by the count (in this case, 7). Each time the

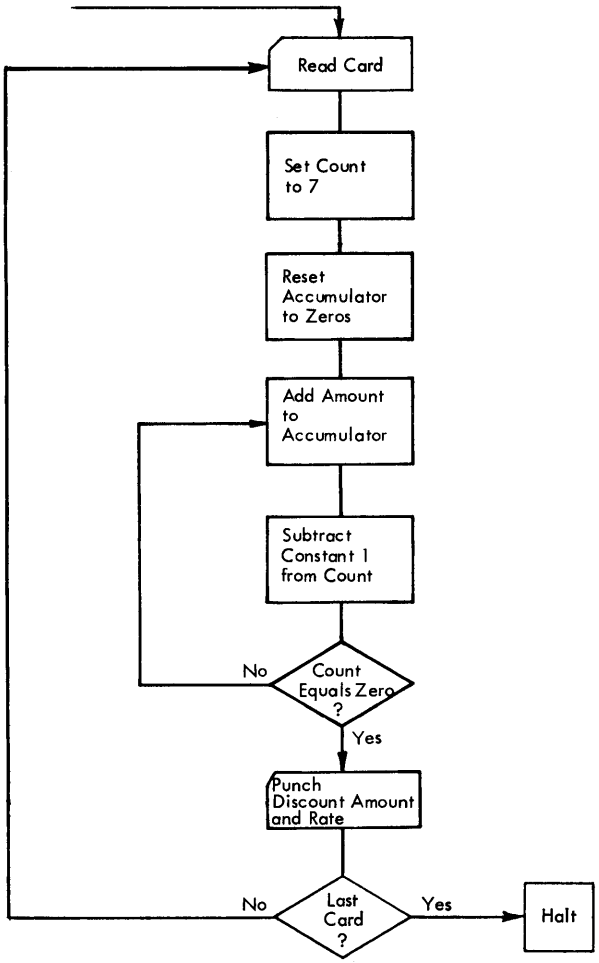


Figure 22A. Block Diagram for Reading/Punching and Multiplication by Repetitive Addition

**	START		
1	HAS LAST CARD BEEN PROCESSED?	N	Y
2	SET ITERATION COUNT = 7	X	
3	SET ACCUMULATOR = ZERO	X	
4	GO TO ITERATION	X	
5	GO TO STOP		X
**	ITERATION		
1	SET ACCUMULATOR + AMOUNT	X	
2	SET ITERATION COUNT – 1	X	
3	GO TO TEST ITERATION COUNT	X	
**	TEST ITERATION COUNT		
1	ITERATION COUNT = ZERO?	N	Y
2	GO TO ITERATION	X	
3	MOVE ACCUMULATOR TO DISCOUNT		X
4	PUNCH DISCOUNT		X
5	GO TO START		X

Figure 22B. Decision Table for Reading/Punching and Multiplication by Repetitive Addition

amount is added, the count is reduced by one. When the count reaches zero, the discount amount is moved from the accumulator to the punch area and the card is punched. Then, the next card is read and the program is repeated.

Figure 23, the coding sheet for this example, contains the 18 statements required to code this program. Lines 01-04 define the card reading and punching areas of storage. Lines 05-06 reserve positions in storage, and lines 07-18 constitute the program itself.

Lines 01-02 define the read area. *Lines 03-04* define the punch area. *Line 05* reserves one position of storage labeled COUNT, and *line 06* reserves six positions of storage labeled ACCUM. *Line 07* reads a card.

Line 08 moves the literal "7" to the position symbolized by COUNT. The plus sign in front of the 7 designates it is a positive numerical literal.

Move Numerical Portion of Single Character. The MLNS operation moves the numerical bits (8421 bits) from the position of storage addressed by the A operand to the position of storage addressed by the B operand. No word marks are needed in either position because only a single digit is transmitted. The A field character is unchanged by this operation. Also, the zone bits (AB bits) of the B field character remain unchanged.

Line 09 resets the accumulator to zeros. *Line 10* adds the amount to the accumulator. *Line 11* subtracts a literal "one" from the count.

Line 12, a BCE operation, tests the position symbolized by COUNT to determine if the count has reached zero. If it has, the program branches to the instruction labeled PCH. If not, it proceeds to the next instruction.

Branch If Characters Equal Operation. The BCE operation is used to test a specific position of storage for a particular character and to branch if that character is present. If that character is not present, the program proceeds to execute the next instruction.

The position to be tested is indicated by the B operand and the particular character is the d-modifier. The address to which the program branches is indicated by the A operand. Note that the d-modifier character and the character to which it is compared must have the same combination of bits to be alike.

Line 13 branches unconditionally to the instruction labeled MULT. *Line 14* moves the product, the "discount amount," to the punch area. Note that the two rightmost digits of the product are dropped because of the A operand being address-adjusted. *Line 15* punches a card. *Line 16* tests for last card. *Line 17* branches unconditionally to the instruction labeled START. *Line 18* ends the program after the BLC operation (line 16) senses the last card.

Page No. 01 of 1

Line	Label	Operation	OPERAND
01	RDAREA	DA	1,x,9,,G DEFINE READ AREA
02	AMOUNT		5,,9
03	PCHARA	DA	1,x,4,,G DEFINE PUNCH AREA
04	DISC		1,,4
05	COUNT	DCW	#1
06	ACCUM	DCW	#6
07	START	R	1,,RDAREA READ CARD
08		MLNS	+7,,COUNT SET COUNT TO SEVEN
09		S	ACCUM,ACCUM RESET ACCUMULATOR TO ZEROS
10	MULT	A	AMOUNT,ACCUM ADD AMOUNT TO ACCUMULATOR
11		S	+1,,COUNT SUBTTRACT CONSTANT ONE FROM COUNT
12		BCE	PCH,COUNT,? COUNT EQUAL PLUS ZERO ?
13		B	MULT
14	PCH	MLC	ACCUM-2,,DISC
15		PS	1,,PCHARA PUNCH CARD
16		BLC	HALT TEST FOR LAST CARD
17		B	START
18	HALT	H	

Figure 23. Coding Sheet for Reading/Punching and Multiplication by Repetitive Addition

Programming Example 4. Multiple-Field Crossfooting with Control Register

This payroll application has as its objectives to:

1. Crossfoot each current earnings card and to punch the resulting gross earnings.
2. Accumulate a total of gross earnings by department and to print that total.
3. Sequence-check the card deck by department number and to stop processing if a sequence error is detected.

The card format of the current earnings card is as follows:

Card columns	14-16	department number (XXX)
	35-39	gross earnings (XXX.XX)
	40-43	miscellaneous earnings (XX.XX)
	44-48	regular earnings (XXX.XX)
	49-52	extra shift earnings (XX.XX)
	53-56	overtime premium earnings (XX.XX)

Miscellaneous, regular, extra shift, and overtime earnings are added together (crossfooted) to obtain gross earnings.

This application combines card reading, card punching, and printing functions in one program. No new operations are described or used. A listing of the operations thus far used follows:

DA	Define Area
DC	Define Constant
DCW	Define Constant with Word Mark
DS	Define Symbol
EQU	Equate
A	Add
S	Subtract
ZA	Zero and Add
MCE	Move Characters and Edit
MCS	Move Characters and Suppress Zeros
MLC	Move Characters to A or B Word Mark
MLCWA	Move Characters and Word Mark from A Field
MLNS	Move Numerical Portion of Single Character
B	Branch Unconditional
BCV	Branch on Carriage Channel 12
BE	Branch on Equal Compare
BH	Branch on High Compare
BL	Branch on Low Compare
BU	Branch on Unequal Compare
BLC	Branch on Last Card
BW	Branch on Word Mark
BCE	Branch If Character Equal
C	Compare
W	Write a Line
WS	Write a Line and Suppress Spacing
R	Read Card to Group Mark with Word Mark
P	Punch and Feed
PS	Punch and Stop
CC	Control Carriage
CS	Clear Storage
CW	Clear Word Mark
H	Halt
SW	Set Word Mark
ORG	Origin

Figure 24 shows the format of the control register. The leading zeros in the department number column are suppressed.

Figure 25A, block diagram, and Figure 25B, decision table, describe Programming Example 4 in a general way. The initializing steps called for in this program are to read a card and to turn off the switch. The next steps are to store department number in a work area, and to reset the accumulator. The earnings are crossfooted to gross earnings and the gross earnings amount is punched. The gross earnings amount is added to the accumulator to accumulate the total department earnings.

The last card indicator is tested. If the indicator is on, the switch is turned on, a line is assembled and printed, the switch is tested, and the program ends. If the last card indicator is off when first tested, the next operations performed are to read another card and to check department sequence.

To check sequence, the program compares the department number in the read area and the department number in the work storage area. If the numbers are equal, signifying there is no change in the control number, the program branches back to repeat the crossfoot operation. If the result of the compare is high,

Dept. No.	Dept. Total
3	1098.67
5	457.83
1 8	785.45
1 1 4	1945.35
Printing Positions 5-7	13-20

Figure 24. Control Register for Multiple-Field Crossfooting

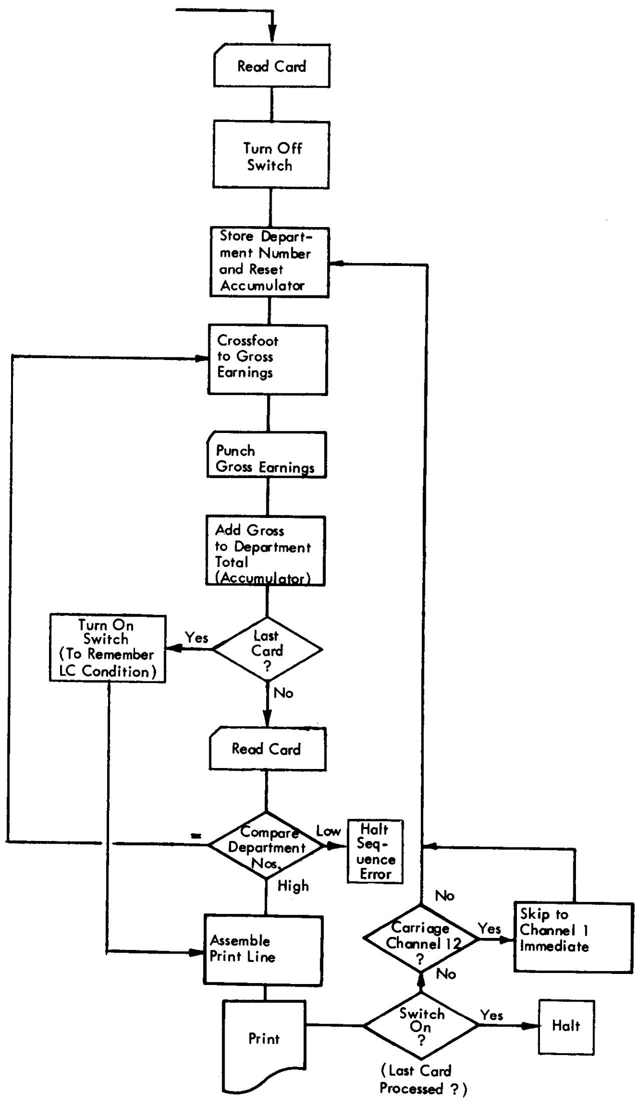


Figure 25A. Block Diagram for Multiple-Field Crossfooting with Control Register

**	START				
1	READ CARD	X			
2	GO TO INITIALIZE	X			
**	INITIALIZE				
1	SET DEPT TOTAL = ZERO	X			
2	MOVE DEPT NO READ IN TO DEPT NO WORK	X			
3	GO TO CROSSFOOT	X			
**	CROSSFOOT				
1	SET GROSS = MISC	X			
	+ REG	X			
	+ EXTRA	X			
	+ O.T.	X			
2	PUNCH GROSS	X			
3	READ CARD	X			
4	SET DEPT TOTAL + GROSS	X			
5	GO TO TEST DEPT CONTROL CHANGE	X			
**	TEST DEPT CONTROL CHANGE				
1	HAS LAST CARD BEEN PROCESSED ?	N	N	N	Y
2	DEPT NO READ IN VS DEPT NO WORK	=	<	>	·
3	GO TO CROSSFOOT	X			
4	GO TO SEQUENCE CHECK HALT		X		
5	GO TO CONTROL REGISTER			X	X
**	CONTROL REGISTER				
1	HAS LAST CARD BEEN PROCESSED ?	N	N	Y	
2	HAS END OF FORM BEEN REACHED ?	N	Y	·	
3	SKIP TO NEXT FORM		X		
4	WRITE PRINT LINE	X	X	X	
5	GO TO INITIALIZE	X			
6	GO TO STOP			X	

Figure 25B. Decision Table for Multiple-Field Crossfooting with Control Register

this condition means there is a change in control number (department change) and the cards are in ascending order. If this is the case, the program proceeds to assemble a printing line and to print. After printing, the switch is tested. If it is on, the last card has been processed and the program is directed to halt. If it is off, the program proceeds to test for carriage channel 12. A skip is initiated to channel 1 of the carriage tape, provided channel 12 was sensed. This is to allow for page overflow. If the result of the compare is low, it signifies the cards are in descending order, therefore out of sequence, and the program is stopped.

After a line is printed, the program branches back to the beginning of the program loop where department number is again stored and the accumulator is reset to zeros.

Figure 26 consists of the coding sheets containing the forty-three statements used to code this application. Line 01 starts the assignment of storage with the address 401. Lines 02-15 define areas and constants, and the remaining lines compose the program.

Line 01 starts assignment of storage.

Lines 02-04 define the printing area.

Lines 05-10 define the card reading area.

Lines 11-12 define the card punching area.

Line 13 reserves a 7-position accumulator labeled ACCUM.

Line 14 defines an edit control word constant.

Page No. 01 of 02

Line	Label	Operation	OPERAND
01		ORG	401
02	PRLINE	DA	1x20,G DEFINE PRINTING AREA
03	DEPTL		7 DEPARTMENT
04	TOTALL		20 TOTAL
05	RDAREA	DA	1x56,G DEFINE CARD READING AREA
06	DEPTC		14,16 DEPARTMENT
07	MISCC		40,43 MISCELLANEOUS EARNINGS
08	REGC		44,48 REGULAR EARNINGS
09	EXTRAC		49,52 EXTRA SHIFT EARNINGS
10	OTC		53,56 OVERTIME EARNINGS
11	PCAREA	DA	1x39,G DEFINE CARD PUNCHING AREA
12	GROSSC		35,39 GROSS EARNINGS
13	ACCUM	DCW	#7
14	CTRLWD	DCW	@bbbbb0.bb@
15	DEPTW	DCW	#3
16	SWITCH	DC	#1 DEFINE SWITCH
17	START	R	1,RDAREA READ CARD
18		CW	SWITCH TURN SWITCH OFF
19	STORE	MLC	DEPTC,DEPTW STORE DEPT NUMBER
20		S	ACCUM,ACCUM RESET ACCUMULATOR TO ZEROS
21	CROSSF	ZA	MISCC,GROSSC CROSSFOOT TO GROSS
22		A	REGC,GROSSC
23		A	EXTRAC,GROSSC
24		A	OTC,GROSSC
25		PS	1,PCAREA PUNCH GROSS EARNINGS

Page No. 02 of 02

Line	Label	Operation	OPERAND
01		A	GROSSC,ACCUM ADD GROSS EARNINGS TO DEPT TOTAL
02		BLC	SETSW TEST FOR LAST CARD
03		R	1,RDAREA READ A CARD
04		C	DEPTW,DEPTC COMPARE DEPARTMENT NUMBERS
05		BE	CROSSF BRANCH IF EQUAL
06		BL	SEQERR BRANCH IF LOW
07	ASSEM	MCS	DEPTW,DEPTL ASSEMBLE PRINT LINE
08		MLCWA	CTRLWD,TOTALL
09		MCE	ACCUM,TOTALL
10		W	PRLINE PRINT
11		BW	END,SWITCH TEST SWITCH
12		BCV	OFLO CARRIAGE CHANNEL 12 ?
13		B	STORE
14	OFLO	CC	1 SKIP TO CHANNEL 1 IMMEDIATE
15		B	STORE
16	SETSW	SW	SWITCH TURN SWITCH ON
17		B	ASSEM
18	END	H	START END PROGRAM
19	SEQERR	H	START

Figure 26. Coding Sheets for Multiple-Field Crossfooting with Control Register

Line 15 reserves a 3-position work area for department number storage.

Line 16 reserves a 1-position work area as a switch.

Line 17 reads a card.

Line 18 turns the switch to its off status by removing the word mark at the position labeled SWITCH.

Line 19 stores the department number in a work area.

Line 20 resets the accumulator to zeros.

Line 21 zero-and-adds miscellaneous earnings to the gross earnings field in the punch area.

Lines 22 -24 add the regular earnings, extra shift earnings, and the overtime earnings to the gross earnings field (punch area).

Line 25 punches the gross earnings amount into the card.

Line 01 (page 2) adds gross earnings to the accumulator (field symbolized by ACCUM).

Line 02 tests the last card indicator and branches to the instruction labeled SETSW.

Line 03 reads a card.

Line 04 compares the department number in the work area with the department number read from the card.

Line 05 branches back to the crossfoot routine (labeled CROSSF), provided the result of the compare operation showed the department numbers to be equal.

Line 06 branches to the halt instruction labeled SEQERR, provided the department number in the read area proved to be lower (of lesser value) than the department number in the work area. This condition (A field < B field) represents a step down in card sequence.

Line 07 moves the department number from the work area to the print area and suppresses leading zeros.

Lines 08 -09 set up the control word in the "department total" print area, and move and edit this amount from the accumulator to that area.

Line 10 prints a department total line.

Line 11 tests the position labeled SWITCH for a word mark and branches to the halt instruction labeled END, provided the word mark is present.

Line 12 branches to the instruction labeled OFLO if the channel 12 indicator is on.

Line 13 branches unconditionally to the instruction labeled STORE.

Line 14 skips the form immediately to channel 1 of the carriage control tape.

Line 15 branches unconditionally to the instruction labeled STORE.

Lines 16 -17 turn the switch on by setting a word mark in the position labeled SWITCH and branch to the instruction labeled ASSEM, which starts the print routine.

Line 18 ends the program.

Line 19 halts the program for a sequence error. Note that separate Halt instructions are provided for the program-end halt and the sequence-error halt although a single halt instruction could have been used. Separate halt instructions allow the computer operator to observe which halt instruction is being displayed at the console and make it easier for him to find the reason for the halt. If only a single halt were provided, the reason for the halt would not be isolated.

Section 4. Questions

1. Which of the following operations is best suited to move an edit control word to a field prior to editing?
   a. MCE
   b. MCS
   c. MLC
   d. MLCWA
2. What is the result of editing the two fields shown, using edit control word [$b,bb0.bb–]?

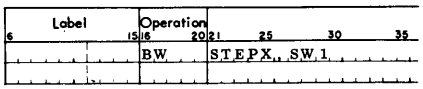
Fields Before Edit	Fields After Edit
000001 (negative)	
643625 (positive)	

3. The instruction to cause skipping after printing a line
   a. must appear before the printing instruction.
   b. must appear after the printing instruction.
   c. can appear before or after the printing instruction.
   d. must appear before and after the printing instruction.
   e. none of the above.
4. Is it true that a hole in channel 1 of the carriage control tape can be used to initiate a carriage skip?

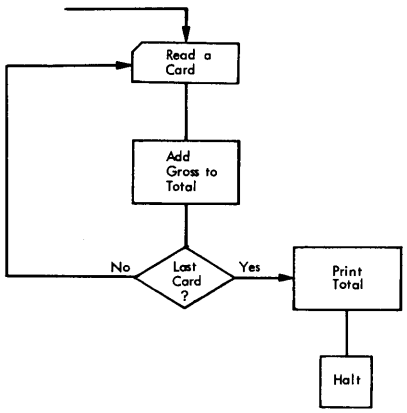
5. Which of the following statements is not true of the print operation?
   a. The position immediately to the left of the area to be printed must contain a group mark with a word mark.
   b. The addressed core storage position is printed in print position 1.
   c. The operand specifies the leftmost position of the print area.
   d. A group mark with a word mark must follow the highest-numbered core storage position to be printed.
   e. The Autocoder operation code for Write a Line is W.
6. Does the carriage skip automatically take place when the carriage channel 12 indicator is found to be on by the BCV operation?
7. Using address adjustment, write a statement that will move only the 123 portion of the field labeled SUM to a 3-position field labeled TOT.

   1234567 SUM is the address of the rightmost position

8. If the contents of the position of storage designated SW1 is a digit 1 with a word mark and the following Branch on Word Mark instruction is executed



   which is the location the program advances to?
   a. the next sequential step.
   b. STEPX.
   c. SW1.
9. Write an instruction that will test the storage position labeled CODE for the letter R, and will branch to the instruction labeled START if the R is present.
10. Given the following block diagram, write a program on an Autocoder program sheet that will read gross earnings from the cards, accumulate a total of gross earnings for all cards, and print this total.



   Use these labels in the area definition statements.

   RDAREA - for read area
   GROSS - for gross earnings field
   PRAREA - for print area
   TOTAL - for total field

   Gross earnings amount is to be read from card columns 1-5, and the total is to be printed in print positions 1-7.
11. A Compare operation compares two fields and
   a. if they are equal, branches to the instruction specified by the A operand.
   b. sets an indicator, which may be tested by subsequent instructions.
   c. if they are unequal, branches to the address of the instruction specified by the B operand.
12. After comparing field A to field B, does the comparison result in an equal or unequal condition?

	Field A	Field B
a.	0000 (unsigned)	0000 (positive)
b.	0000 (unsigned)	0000 (unsigned)
c.	0000 (unsigned)	00bb (unsigned)
d.	123J (unsigned)	1231 (unsigned)

13. Write instructions to clear to blanks the following storage positions:

    800-999
    600-625
    500-501
    200 only

Programming with Disk Storage

Section 5

The sample programs we developed in previous sections were based on a system composed of a Card Read-Punch, a Processing Unit, and a Printer. In this section we will study ways to use the supplemental storage provided by the 1311 Disk Storage Drive, and we will program some basic disk storage operations in Autocoder language.

The 1311 Disk Storage Drive supplements the core storage area at a reasonable and acceptable cost. Like core storage, disk storage can be used to store program instructions, tables, and data records. It can also serve as a working storage area; i.e., an area for storing intermediate results, input data, and output data.

It is advisable in large programs to store the most infrequently used subroutines in disk storage, because it takes a certain amount of time to read these subroutines into core storage where they can be operated on. Using disk storage in this way permits the program to execute its functions in the shortest amount of time.

Some applications require large rate tables. If the tables are stored in disk storage rather than core storage, more core storage is made available for the program and for the input and output areas.

Data records may be stored in disk storage in either a sequential or nonsequential (random) manner.

Sequential Method

For sequential storing, virtually unlimited storage is available. As soon as one disk pack is filled to capacity, it can be removed and replaced by another.

Data records that are stored sequentially usually contain some control field, for example, account number, that is in the same sequence as the sequence in which the records are stored. This control field makes it easy for the computer to scan the sequence of records and update or post changes to the records.

Sequential storing of data is usually used when:

1. Transactions to be processed are in a given sequence (same order as records).
2. Most of the records require processing for any given application.
3. Unlimited storage is needed for the records.
4. Sequential reports are required by the user.

The term "sequential searching" implies that each record, starting with the first, must be scanned to locate a specific record. However, indexing may be used to eliminate the scanning of every record. Indexing is a method of cross-referencing a control field with a disk storage address through the use of a table. By scanning the table, the computer can obtain the address of the record and thus locate the record without sequential searching.

Nonsequential (Random) Method

For nonsequential storing, data records have no particular sequence so far as control information or contents is concerned. However, if these data records contain a control field that can be used as the actual disk address, they can be stored by direct addressing and retrieved in the same way. If direct addressing is not possible, a form of indirect addressing may be used. In indirect addressing, some technique must be employed for converting the control field to the disk address. A number of such techniques known as randomizing techniques are available.

The nonsequential (random) method of data storage is usually used when:

1. Sorting of transactions in a given sequence is undesirable, or not possible before processing.
2. A low percentage of the records is used during processing (low activity for many records).
3. The maximum number of disk records to be made available at one time for processing will fit within an established disk storage area.
4. Nonsequential reports are required by the user.

1311 Disk Storage Operations

Disk storage instructions consist of five basic operations:

Seek
Read
Write
Write Disk Check
Scan Disk (special feature)

The Read, Write, and Scan Disk operations have numerous variations that increase their effectiveness. A general description of the various Disk Operations follows.

Seek Operation

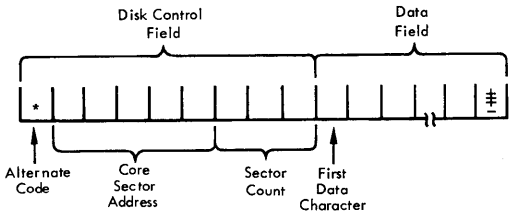
The Seek operation consists of an instruction that directs the read/write heads to the proper cylinder on the disk pack. Data on the disk records is not acted upon by this instruction. The seek instruction merely positions the access arms at the proper cylinder.

The Seek instruction is followed by a read instruction or a write instruction. The operand of this instruction specifies the core storage address of the Disk Control Field (described under READ DISK OPERATION).

Read Disk Operation

The Read Disk operation transfers data from disk storage to the core storage area of the processing unit.

The core storage area where the data is to be placed must be defined by an instruction that precedes the seek or read instruction. This area is defined thus:



The Disk Control Field determines the location in disk storage of the data to be used and determines the number of disk sectors in the data record. The number of positions in core storage reserved for the data field must be large enough to contain all the data read from the disk. The group mark with a word mark at the end of the data field defines the total length of the area reserved.

The Sector Count part of the disk control field specifies the number of disk sectors that are to be read or written by the instruction.

The Core Sector Address part of the Disk Control Field contains the disk address of the data to be read or written. If more than one sector is read or written, the address of the first sector is specified.

The disk address of the data indicates which storage drive is to be used by the system for an instruction, as shown in the examples below.

Address Block	Selects Disk Drive
00000-19999	0
20000-39999	1
40000-59999	2
60000-79999	3
80000-99999	4

If the disk address on a disk pack does not correspond to the number of the disk storage drive it is located on, then a Disk Alternate code is used.

This coding provides the flexibility to operate upon any block of disk addresses, regardless of which disk drive the pack is located on, and, when the occasion requires, to have more than one Disk Pack containing the same disk addresses on-line at once.

The selection of a particular Disk Storage Drive is made by using the following Alternate codes:

Alternate Code	Selects Disk Drive
0	0
2	1
4	2
6	3
8	4

For example, if address block 60000-79999 is located on Disk Drive 4 (instead of Disk Drive 3), the Alternate code in the Disk Control Field will be 8.

When this flexibility is not required, the drive is selected directly from the core sector address and an asterisk (*) is placed in the Alternate Code part of the Disk Control Field.

Write Disk Operation

The Write Disk operation transfers data from core storage to disk storage. An area in core storage is set aside for the Disk Control Field, which specifies the disk storage drive to be used, the disk address where the data is to be written, and the number of sectors to be written on.

The data to be transferred to disk storage is located immediately to the right of the Disk Control Field. If, for example, 100 character records are to be written into core storage, 110 core storage positions will be reserved; 9 positions for the Disk Control Field, 100 positions for the data, and one position for a group mark with a word mark.

Write Disk Check Operation

A Write Disk Check instruction *must* immediately follow each Write Disk instruction. The Write Disk Check operation causes the data just written in *disk* storage to be read and compared with the original source data in *core* storage. When the disk data does not compare, bit-by-bit and character-by-character, with the core storage data from which it was written, a Disk Error indicator is set.

Scan Disk Operation (Special Feature)

This feature provides the system with the ability to make a rapid search of a disk pack for a specific code or condition (such as data or account number) that is *stored within the data itself*. Only one seek and one scan disk instruction are required to cause the program to search through an entire cylinder (200 sectors). The scan can be made to compare the search argument with the data, on the basis of the argument being low or equal to the data, equal to the data, or high or equal to the data.

The program can be directed to scan (search) all records in a file, or all records in a cylinder, or any specific number of records. The Disk Control Field contains the address of the sector where the scan is to begin and the maximum number of sectors to be scanned.

Four Modes of Read or Write Disk Operations

Read instructions and Write instructions have four modes:

Address Mode
Sector Mode
Sector Overlay Mode
Track Record Mode

It is possible to transfer data, with or without word marks, in any of these modes.

Address Mode

This mode of operation allows sector addresses recorded in disk storage to be changed. It transfers *both* data and disk sector addresses to and from the file, one complete track at a time. The sector count must be set to 020 (sectors are transferred in blocks of twenty).

The operation requires that the Disk Control Field contain an address of one of the sectors within the track. This address must be satisfactorily compared with its counterpart in disk storage before the transfer can take place.

To write in the Address Mode, the Write Address key-light on the number 0 Disk Storage Drive unit must be set to the ON position.

Sector Mode

The Sector Mode is the normal mode of operation. Read and Write operations in the Sector Mode transfer data, but do *not* transfer disk sector addresses. The number of sectors to be handled within one operation is designated in the Sector Count part of the Disk Control Field.

Each sector is transferred only upon absolute comparison of sector addresses in Disk Storage with those in the Disk Control Field. The Core Sector Address in the Disk Control Field is automatically increased by one for each sector transferred and thereby supports the comparison of successive sector addresses. Similarly, the Sector Count is reduced by one and indicates by a 000 setting that the required operation has been completed.

If a group mark with a word mark is sensed in the Record field before the sector count reaches zero, the operation is terminated. Likewise, if the sector count reaches zero before the group mark with a word mark is sensed, the operation is terminated. Either of these events will result in the Wrong Length Record indicator being turned on.

The Wrong Length Record indicator is tested by a Branch If Indicator On instruction. It is reset by the next Disk operation.

Sector Count Overlay Mode

The Sector Count Overlay mode of operation allows a record to indicate the number of sectors it contains by modifying the Sector Count portion of the Disk Control Field. This technique permits better utilization of disk storage in sequential-type applications having variable size records. In all cases, the variable size records must be in increments of 100.

When such a record is read into the processing unit in the Sector Count Overlay Mode, the first data character transferred from disk storage is stored in the leftmost position of the Sector Count part of the Disk Control Field *instead* of the position immediately to the right of the Disk Control Field. When a record is stored in disk storage from core storage, the transmission of data starts with the leftmost position of the sector count field.

In the normal operation of read and write instructions, the Sector Count in the Disk Control Field is decremented by one each time a sector is read from or written into the file. At the same time the Sector Address in the Disk Control Field is incremented by one. The operation is stopped when the sector count reaches zero. Incrementing and decrementing are automatic functions of the processing unit.

Initially, the Sector count must be set to a number greater than 001 before a read operation in the Sector Count Overlay mode is executed. The Sector Count Overlay mode cannot process records containing only one sector.

Track Record Mode (Special Feature)

Normally one 100-character record is written in or read from each of the 20 sectors of a track. The Track Record mode makes it possible to read or write *one* record in place of the twenty sectors of a track. This record consists of a 5-position address and 2980 positions of data. The Track Record mode can also be used in the address mode to transfer the 5-digit address with the data.

Reading and Writing with Word Marks

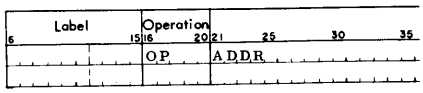
In all reading and writing operations, the data together with word marks may be read from, and written into, disk storage. When word marks are written on the file, the data is written in 8-bit BCD coding. The use of 8-bit coding reduces the number of characters that can be stored on a sector from 100 to 90, and reduces the total number of characters on a disk pack from 2,000,000 to 1,800,000. In the Track Record Mode, the maximum number of characters on one track is reduced from 2980 characters to 2682 characters.

Fixed and Variable Length Records

Fixed length and variable length records can be stored in disk storage. The sector mode is used to process fixed length records. The sector count of the Disk Control Field in this mode is *constant* and indicates the number of sectors contained in the record. The Sector Count Overlay mode is used to process variable length records. In this mode, the record itself contains the sector count. Because each record can contain a different sector count, the length of each record can be variable.

Format of Disk Storage Statement

The disk storage statement takes the following form,



where OP is one of the disk mnemonic operation codes and ADDR is the address of the leftmost position of the Disk Control Field used in the operation.

The mnemonic operation code to be used for all seek operations is:

SK (Seek Disk)

Read, Write, and Write Disk Check mnemonic operation codes are shown by mode in Table 3.

The Scan Disk (special feature) mnemonic operation codes are given in Table 4.

Table 3. Mnemonic Operation Codes Used for Read, Write, and Write Disk Check Operations by Modes

Operation	Modes				
	SECTOR (data only)	ADDRESS (data and addresses)	SECTOR OVERLAY (data and sector count)	TRACK RECORD (special feature)	
				data only	data and addresses
READ DISK					
without WM*	RD	RDT	RDCO	RDTR	RDTA
with WM	RDW	RDTW	RDCOW	RDTRW	RDTAW
WRITE DISK					
without WM	WD	WDT	WDCO	WDTR	WDTA
with WM	WDW	WDTW	WDCOW	WDTRW	WDTAW
WRITE DISK CHECK					
without WM	WDC	WDC	WDC	WDC	WDC
with WM	WDCW	WDCW	WDCW	WDCW	WDCW

*WM = Word Mark

Table 4. Mnemonic Operation Codes for the Scan Disk Operation

This operation compares the Argument to the Data for one of the conditions shown in this table.

Operation	Condition		
	Low or Equal	Equal	High or Equal
SCAN DISK (Special Feature)			
without WM	SDL	SDE	SDH
with WM	SDLW	SDEW	SDHW

Disk Check Indicators

Several disk check indicators provided on the 1440 system are automatically turned on to indicate disk storage conditions or errors. It is up to the programmer to test the indicators in his program to identify the condition or error and to specify the course of action to be followed by his program.

A description of the six disk check indicators follows.

Any Disk Condition

This indicator is set in unison with any other disk check indicator. It is reset by the next programmed disk storage operation. It can be tested by a Branch If Indicator On instruction.

Access Busy

This indicator is set when reference is made to a Disk Storage Drive that is in a Busy status. The programmed disk operation is terminated. The indicator is tested by a Branch If Indicator On instruction. It is reset upon completion of the next programmed disk storage operation.

Access Inoperable

This indicator is set when reference is made by a disk operation to a Disk Storage Drive that is in a Not Ready status. Because the access is in a Not Ready status, the disk operation is terminated.

The Access Inoperable indicator is tested by a Branch If Indicator On instruction. It is reset by the next programmed disk storage operation.

Disk Error

This indicator is set when a parity error is detected during a Read or Write Disk operation, or when an Unequal Compare or parity error occurs during a Write Disk Check operation. The indicator is tested by a Branch If Indicator On instruction. It is reset by the next programmed disk storage operation.

Wrong Length Record Check

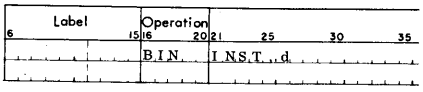
This indicator is set when a 000 Sector Count and the group mark with a word mark fail to correspond at the end of a Read or Write or Disk Check operation. This indicator is also turned on if data records read from disk storage or written to disk storage are not in even increments of 100 characters. The indicator is tested by a Branch If Indicator On instruction. It is reset by the next programmed disk storage operation.

Unequal Address Compare

This indicator is set (1) if a disk data transfer is initiated and the sector address does not compare and (2) after the data transfer process begins, if the next sector address in physical sequence fails to compare, i.e., the sector address of the Disk Control Field and the next disk address are unequal. The indicator is tested by a Branch If Indicator On instruction. It is reset by the next programmed disk storage operation.

Format of Branch If Indicator On Instruction

The Branch If Indicator On instruction can be used to test any of the indicators contained in the system. Included in these indicators are the six Disk Check indicators. The instruction used to test these indicators takes the following form



where BIN is the actual mnemonic operation code, INST is the address of the instruction which the program branches to if the indicator is on, and d is the d-modifier character which identifies the indicator to be tested. A list of the d-modifier characters that can be used with this instruction is given in Table 5.

Table 5. d-modifier Characters Used with Branch If Indicator On Operation

d-modifier character	Indicator	Reset by
b1	Unconditional	Branch
9	Carriage Channel 9	Branch Test
@	Carriage Channel 12	Branch Test
A	"Last Card" Switch (Sense Switch A)	Manual
B	* Sense Switch B	
C	* Sense Switch C	
D	* Sense Switch D	
E	* Sense Switch E	
F	* Sense Switch F	
G	* Sense Switch G	
N	Access Inoperable	Next Disk Storage operation
\	Access Busy	Next Disk Storage operation
P	Printer Busy	
/	Unequal Compare (B ≠ A)	Next Compare or Disk Storage operation
S	Equal Compare (B = A)	
T	Low Compare (B < A)	
U	High Compare (B > A)	
V	Disk Error	Next Disk Storage Operation
W	Wrong Length Record	
X	Unequal-Address Compare	
Y	Any Disk Condition	
Z	Overflow	Branch Test
%	Processing Check with Process Check Switch off	Branch Test
?	Read Error (If I/O Check Stop Switch is off)	Reset by Test; must be reset before next operation
!	Punch Error (If I/O Check Stop Switch is off)	
≠	Printer Error (If I/O Check Stop Switch is off)	

* Special Feature

Testing the Disk Check Indicators

The programmer should *always* test the Access Busy indicator after the first disk (seek, read, write, etc.) instruction following a seek. If this test is omitted, the disk instruction which follows the seek can be negated (not acted upon) without being detected. If this indicator is on when tested, it means that the disk instruction was negated. The program can then be instructed to branch back to repeat the negated disk operation. If the indicator is off when tested, the program may proceed with the next instruction, for the test indicates that the disk operation has been performed.

The Any Disk Condition indicator makes it possible to make one check for any disk errors, and if the indicator is off, to proceed with the normal program. If the indicator is on, only then is it necessary to test the other indicators to find out which error condition is present. A good practice in coding a disk operation is to follow it always by a test of the Any Disk Condition indicator.

The programmer can write one subroutine to test the indicators and use this common subroutine after every disk operation, and thus save program storage space. In such a subroutine, it is desirable to have a program loop which will cause the disk operation to be repeated (up to three times) if the Disk Error indicator is on. By repeating the operation, the error in some instances may be corrected automatically by the computer without operator intervention, thus saving valuable computer time.

For the other indicators (Access Inoperable, Wrong Length Record, or Unequal Address Compare) it may be desirable to halt the program if any of these is on. The program should then try to re-execute the disk storage operation when the Start key is depressed. It should be noted that an ON condition of the Unequal Address indicator can be caused by a cylinder overflow; i.e., by attempting to read or write beyond the capacity of a cylinder. In some applications, the programmer may want to write beyond the capacity of a cylinder. If he does he must give another seek instruction to get to the next cylinder; and then, a second read or write instruction to continue the operation which initially caused the cylinder overflow.

Programming Example 5. Storing Records in Disk Storage

In Programming Example 5, we will read name and address data from cards, two cards for each address, and store the name and address record in disk storage. We will use a control field from the card for direct addressing, that is, use the control field as the sector address. Figure 27 shows the format of input cards. Customer number, which will also become the direct address, is the same number in both cards. Codes 1 and 2 in column six identify the first and second cards of a name and address record.

In Figure 28A, block diagram, and Figure 28B, decision table, we can see that the card code determines the course of action to be followed in the program. Card code 1, the customer name and address part of the record, is assembled and a seek instruction is initiated. Card code 2, the city and state part of the record, is assembled and the disk record is written into disk storage from core storage. Figure 29 is a record layout which corresponds to the 100 characters of a sector.

The Write Disk operation is followed by a Write Disk Check operation, as it must be. Note that certain parts of the disk control field are changed when the write command is given; the sector address is incremented by one and the sector count is decremented by one. Therefore, the disk control field must be restored to its original value before the Write Disk Check can be given. The Any Disk Condition indicator is tested following the Write Disk Check operation. If it is on, the program halts. The program would not halt at this point if we had inserted in our program a subroutine for testing the individual indicators, as described earlier.

Figure 30 is the program coding sheet. First, a card reading area (66 positions) and disk record area (100 positions), each followed by a group mark with a word mark, are defined in core storage. The disk control field, defined by a DCW statement, immediately precedes the

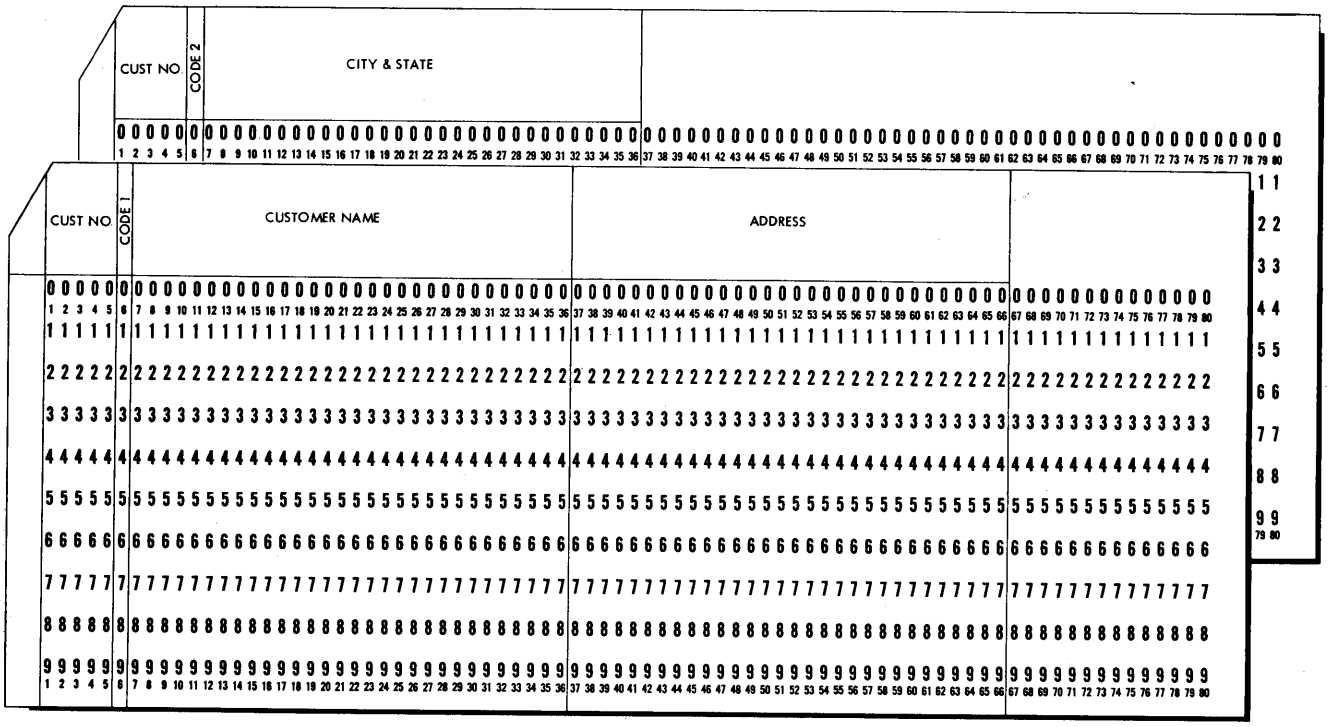


Figure 27. Name and Address Cards for Storing Records in Disk Storage

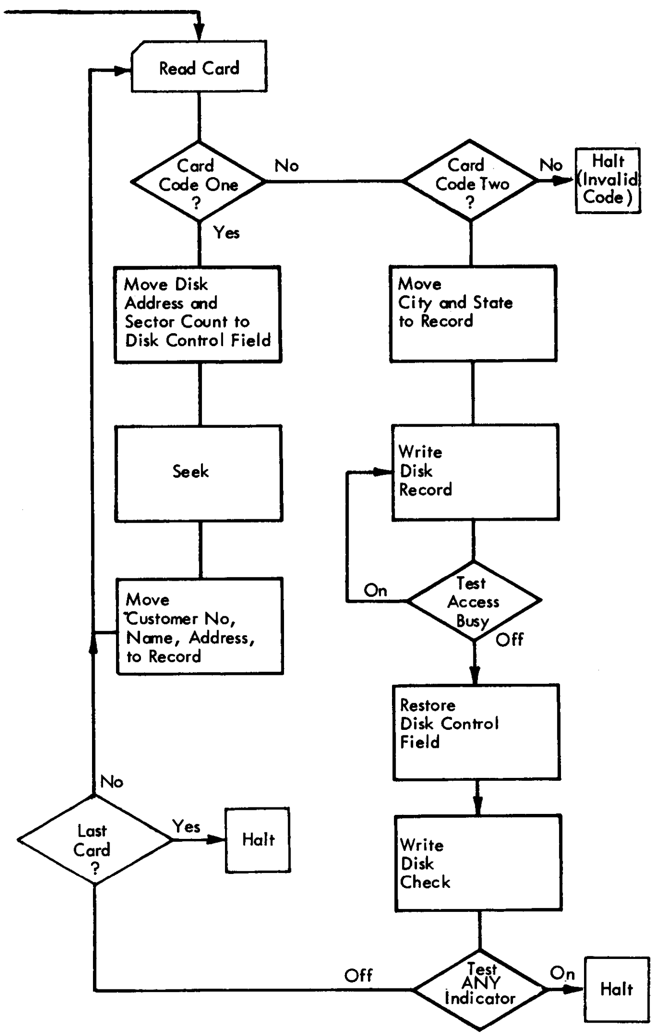


Figure 28A. Block Diagram for Storing Records in Disk Storage

**	START			
1	HAS LAST CARD BEEN PROCESSED ?	N	Y	
2	READ CARD	X		
3	GO TO TEST CARD CODE	X		
4	GO TO END OF JOB HALT		X	
**	TEST CARD CODE			
1	CARD CODE EQ 1	Y	N	N
2	CARD CODE EQ 2	N	Y	N
3	SET CORE SECTOR ADDR EQ CUST.NO.	X		
4	SET SECTOR COUNT EQ 001	X		
5	MOVE CUST.NO., NAME, ADDR TO RECORD: CORE	X		
6	MOVE CITY, STATE TO RECORD;CORE		X	
7	WRITE RECORD;CORE TO DISK		X	
8	DO TEST DISK INDICATORS		X	
9	SET SECTOR COUNT EQ 001		X	
10	WRITE CHECK RECORD: DISK		X	
11	DO TEST DISK INDICATORS		X	
12	GO TO START	X	X	
13	GO TO ERROR HALT			X
**	TEST DISK INDICATORS			
1	ARE ANY DISK INDICATORS ON ?	N	Y	Y
2	IS ACCESS BUSY INDICATOR ON ?		Y	N
3	WRITE RECORD TO DISK AGAIN		X	
4	HALT			X
5	RETURN	X	X	

Figure 28B. Decision Table for Storing Records in Disk Storage

record-defining statements. This arrangement of statements is necessary to assign the disk control field the nine positions to the immediate left of the 100-position record area. Note that records are written in the sector mode; therefore, the sector mode operation code mnemonics are used.

Programming Example 6. Printing Name and Address Labels from Disk Storage

This problem may be considered an extension of the previous one where we loaded name and address records into disk storage. Here, we will use the finder cards which contain the customer number (direct disk address) in columns 1-5 to select the disk records to be printed. The format of the disk record will be the same as it was in the last example (Figure 29).

Figure 31 shows the format for the three name and address labels that are to be prepared simultaneously from one record. The channel 1 punches in the carriage control tape are used for carriage skipping to the first line of the addresses.

In Figure 32A, block diagram, and Figure 32B, decision table, note that the address (customer number) in each card is used to select the record. After the record is read into core storage, the customer number in the record and the customer number in the finder card are compared to see if they are equal.

1 5	6 35	36 65	66 95	100
Cust No	Customer Name	Address	City and State	

Figure 29. Record Layout for Storing Records in Disk Storage

Page No. 001 of 002

Line	Label	Operation	OPERAND
01	RDAREA	DA	1x66,G DEFINE CARD READING AREA
02	CUSNOC		1,5 CUSTOMER NUMBER
03	CODEC		6 CODE
04	NAMEC		7,36 NAME OR CITY & STATE
05	ADDRC		37,66 ADDRESS
06	DSKCTF	DCW	@*bbbbbb,001@ DISK CONTROL FIELD CONSTANT
07		DA	1x100,G DEFINE DISK RECORD
08	CUSNOR		5 CUSTOMER NUMBER
09	NAMER		35 NAME
10	ADDRR		65 ADDRESS
11	CITYSR		95 CITY & STATE
12	START	R	1,RDAREA READ CARD
13		BCE	CARD1,CODEC,1 CARD CODE 1 ?
14		BCE	CARD2,CODEC,2 CARD CODE 2 ?
15		H	START HALT FOR INVALID CODE
16	CARD1	MLC	CUSNOC,DSKCTF-3 MOVE DISK ADDRESS TO DISK CTRL FILD
17		MLNS	@1@,DSKCTF MOVE SECTOR COUNT TO DISK CTRL FILD
18		SD	DSKCTF-8 SEEK
19		MLC	NAMEC,NAMER MOVE CUSTOMER NAME AND
20		MLC	ADDRC,ADDRR ADDRESS, TO RECORD
21		B	START BRANCH UNCONDITIONALLY TO START
22	CARD2	MLC	NAMEC,CITYSR MOVE CITY & STATE TO RECORD
23	WRITE	WD	DSKCTF-8 WRITE DISK RECORD
24		BIN	WRITE,\ BRANCH IF ACCESS BUSY
25		MLNS	@1@,DSKCTF RESTORE DISK CONTROL FIELD

Page No. 002 of 002

Line	Label	Operation	OPERAND
01		A	@1@,DSKCTF-3
02		WDC	DSKCTF-8 CHECK DISK WRITING
03		BIN	DSKERR,Y BRANCH IF ANY ERROR
04		BLC	LSTCD TEST FOR LAST CARD
05		B	START BRANCH UNCONDITIONALLY TO START
06	DSKERR	H	START HALT FOR DISK ERROR
07	LSTCD	H	START HALT FOR LAST CARD

Figure 30. Coding Sheets for Storing Records in Disk Storage

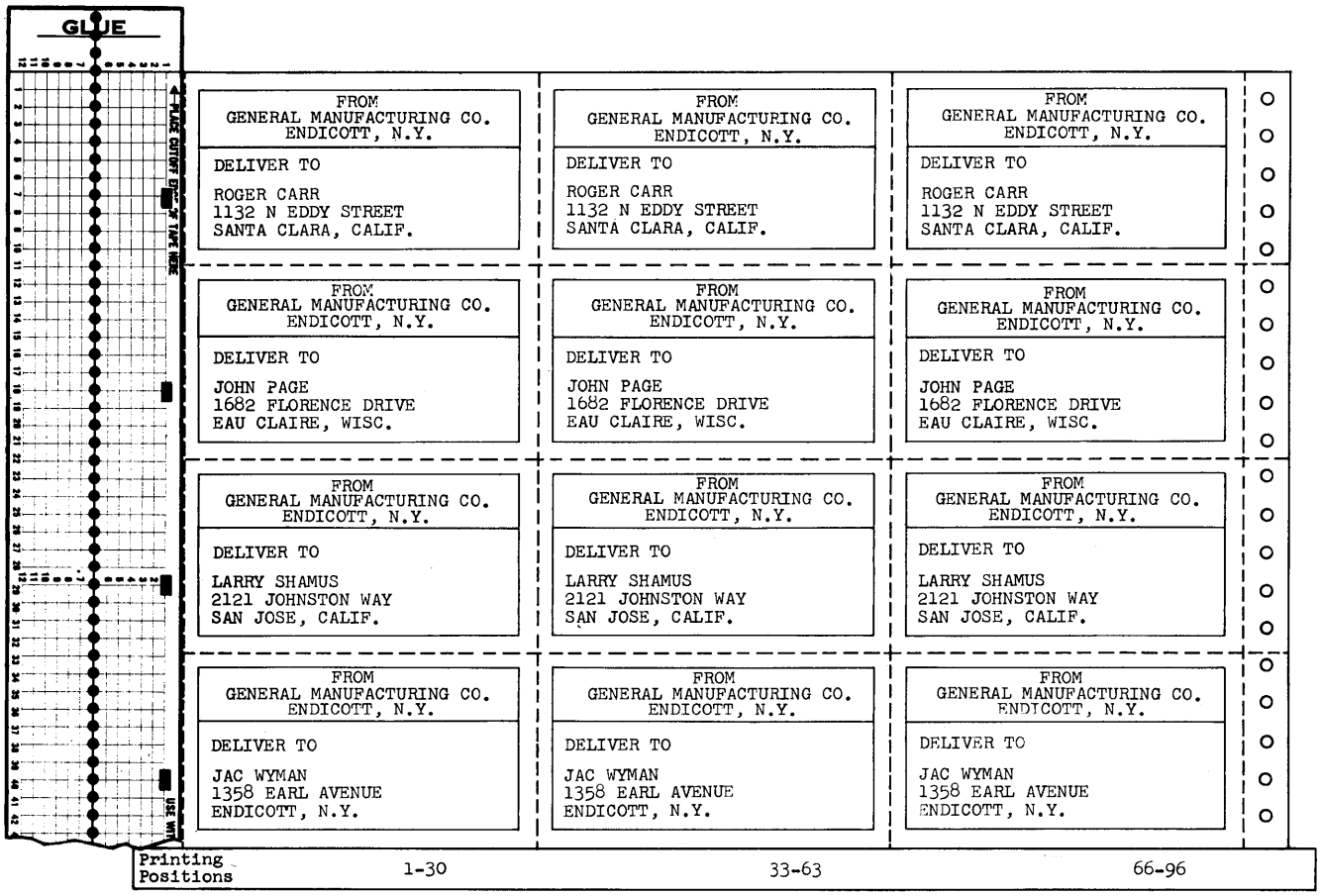


Figure 31. Form Layout for Printing Name and Address Labels from Disk Storage

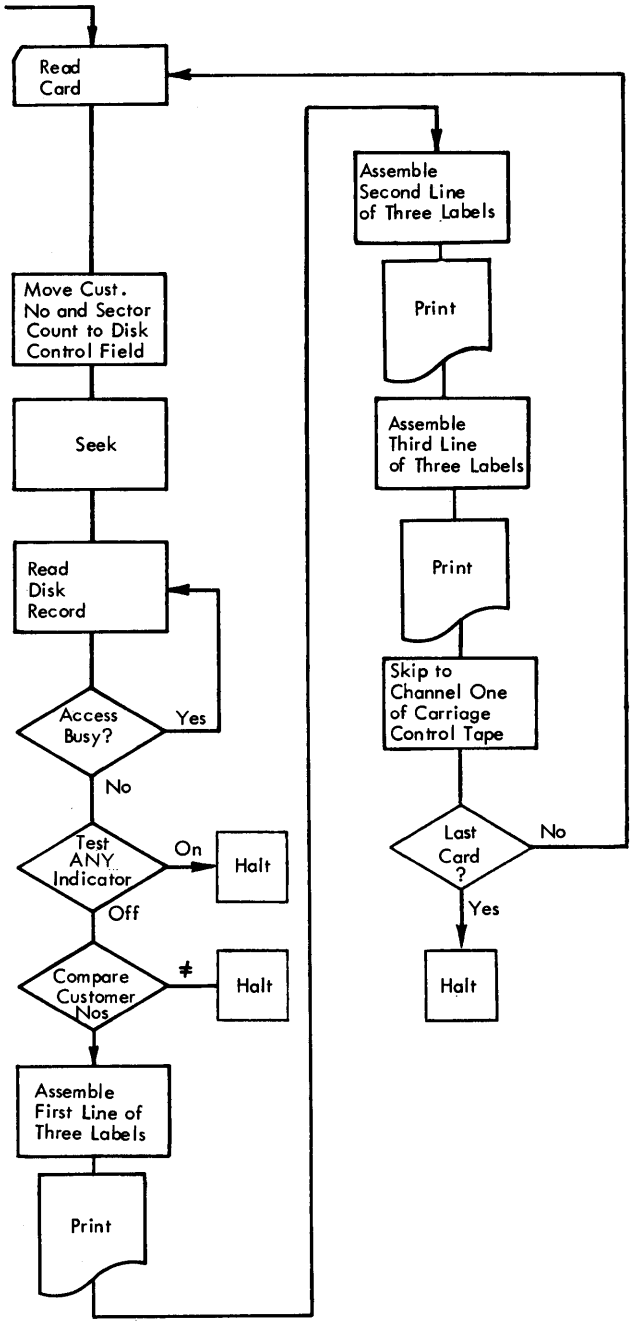


Figure 32A. Block Diagram for Printing Name and Address Labels from Disk Storage

**	START												
1	HAS LAST CARD BEEN PROCESSED?	N	Y										
2	READ CARD	X											
3	SET CORE SECTOR ADDR EQ CUST. NO.	X											
4	SET SECTOR COUNT EQ 001	X											
5	GET CUSTOMER RECORD: DISK	X											
6	DO TEST DISK INDICATORS	X											
7	DO COMPARE CUSTOMER NO.	X											
8	PRINT LABELS	X											
9	GO TO START	X											
10	GO TO HALT		X										
**	TEST DISK INDICATORS												
1	ARE ANY DISK INDICATORS ON?	N	Y	Y									
2	IS ACCESS BUSY ON?	N	Y	N									
3	GET CUSTOMER RECORD: DISK AGAIN		X										
4	HALT			X									
5	RETURN	X	X										
**	COMPARE CUSTOMER NO.												
1	CUST. NO: CARD VS CUST. NO. DISK	=	≠										
2	HALT		X										

Figure 32B. Decision Table for Printing Name and Address Labels from Disk Storage

This check also verifies that the right record was read. In all disk read operations it is good programming procedure to have some field of data in the record such as part number or customer number, which can be compared to a given field to verify that the correct record was read.

In this programming example, we could have printed the name and address labels for each customer, one after the other, rather than printing the three at one time. However, more computer time would have been used to do it this way. Often, it is advantageous to print copies side by side when the printing capacity allows such operation. Figure 33 is the coding sheet for this program.

Programming is the process of preparing a logical set of instructions for the computer to follow. There are many programming systems in existence; Autocoder is but one. Since the 1440 uses Autocoder, we have explored this particular programming system in this book.

Not all functions of the 1440 Data Processing System or of the Autocoder programming system are explored nor are the programming solutions in this book the only acceptable ones. But we have presented the fundamentals which will allow the student programmer to go on to attempt to program a greater variety of jobs and more difficult jobs.

If possible, the student programmer should work with an experienced programmer in developing the first few programs. If not, he should get the help of an experienced programmer and get him to check the accuracy of his programs. In the end, though, it is practice that counts.

Page No. 01 of 02

Line	Label	Operation	OPERAND
01		ORG	401 START S THE ASSIGN OF STORAGE WITH ADDR 401
02	PRAREA	DA	1 X96 G DEFINE PRINTING AREA
03	LAB1		30 FIRST NAME AND ADDRESS LABEL
04	LAB2		63 SECOND NAME AND ADDRESS LABEL
05	LAB3		96 THIRD NAME AND ADDRESS LABEL
06	DSKCTF	DCW	@*bbbbbb001@ DISK CONTROL FIELD CONSTANT
07		DA	1 x 100 G DEFINE DISK RECORD
08	CUSNOR		1 5 CUSTOMER NUMBER
09	NAMER		6 35 NAME
10	ADDRR		36 65 ADDRESS
11	CITYSR		66 95 CITY & STATE
12	RDAREA	DA	1 x 5 G DEFINE CARD READING AREA
13	CUSNOC		1 5 CUSTOMER NUMBER
14	START	R	1 RDAREA READ A CARD
15		MLC	CUSNOR DSKCTF-3 MOVE CUST NO TO DISK CONTROL FIELD
16		MLNS	@1@ DSKCTF MOVE SECTOR COUNT (1) TO DSKCTF
17		SD	DSKCTF-8 SEEK
18	READ	RD	DSKCTF-8 READ DISK RECORD
19		BIN	READ\ TEST ACCESS BUSY INDICATOR
20		BIN	DSKERR Y TEST ANY DISK CONDITION INDICATOR
21		C	CUSNOR CUSNOC COMPARE CUST NOS RECORD AND CARD
22		BU	WRONGR BRANCH IF UNEQUAL
23		MLC	NAMER LAB1 MOVE NAME TO FIRST LABEL
24		MLC	NAMER LAB2 MOVE NAME TO SECOND LABEL
25		MLC	NAMER LAB3 MOVE NAME TO THIRD LABEL

Page No. 02 of 02

Line	Label	Operation	OPERAND
01		W	PRAREA PRINT FIRST LINE
02		MLC	ADDRR LAB1 MOVE ADDRESS TO FIRST LABEL
03		MLC	ADDRR LAB2 MOVE ADDRESS TO SECOND LABEL
04		MLC	ADDRR LAB3 MOVE ADDRESS TO THIRD LABEL
05		W	PRAREA PRINT SECOND LINE
06		MLC	CITYSR LAB1 MOVE CITY & STATE TO FIRST LABEL
07		MLC	CITYSR LAB2 MOVE CITY & STATE TO SECOND LABEL
08		MLC	CITYSR LAB3 MOVE CITY & STATE TO THIRD LABEL
09		W	PRAREA PRINT THIRD LINE
10		CC	1 SKIP TO FIRST LINE OF NEXT LABEL, CHN1
11		BLC	LSTCD TEST FOR LAST CARD
12		B	START BRANCH UNCONDITIONALLY TO START
13	LSTCD	H	START HALT FOR LAST CARD
14	WRONGR	H	START HALT FOR WRONG RECORD
15	DSKERR	H	START HALT FOR DISK ERROR

Figure 33. Coding Sheets for Printing Name and Address Labels from Disk Storage

Section 5. Questions

1. List two other items, besides data records, that could be stored in disk storage.
2. Is it true that the sequential storage method of storing records offers the advantage of unlimited storage capacity?
3. List the three parts of the disk control field.
4. An area of core storage reserved for reading a disk record must be followed by a group mark with a word mark and immediately preceded by
   a. an asterisk.
   b. the disk control field.
   c. a group mark with a word mark.
5. After a disk record is read into an area of core storage from disk storage, what are the two changes that must be made to the disk control field before the record can be written back to its original location in disk storage?
6. If a disk pack containing address block 00000-19999 is located on disk drive 2 instead of disk drive 0, the disk alternate code in the disk control field should be
   a. zero
   b. eight
   c. twelve
   d. four
   e. seven
7. Write an instruction which will test the Unequal Address Compare indicator and, if it is on, branch to the instruction labeled TEST.
8. The Access Busy indicator must be tested
   a. only once during a program.
   b. after every disk operation.
   c. after a disk read or write operation.
   d. after the first disk (seek, read, write, etc.) instruction following a seek instruction.
9. The number of tracks within a disk cylinder is
   a. 100
   b. 10
   c. 1000
   d. 99

Appendix

Section 1. Answers and Solutions

1. 1443 Printer
2. Yes. All data goes through the processing unit.
3. Yes. Data from the processing unit may be moved as output to any one output device or any combination of output devices.
4. 4000 characters
5. A card must pass the reading station before it can be punched.
6. a. Control b. Storage c. Arithmetic
7. 90 characters with word marks
   100 characters without word marks
8. 20,000-20,199 (address range)
9. Disk Sector address
10. Yes. Each position of core storage has a unique address.
11. To identify the leftmost position of a data field.

Section 2. Answers and Solutions

1. a. Establish the requirements of the problem and the requirements of the solution.
   b. Write the program for the computer to follow.
2. Autocoder.
3. b. To assign storage areas for data.
4. a. The leftmost position of the area reserved.
5. b. The address of the rightmost position of the constant.
6. To specify that the constant is alphameric.
7. The leftmost position of the first constant will be given a word mark by the processor, whereas, the second constant will contain no word mark.
8.

Label	Operation	6 15 16 20 21 25 30 35 40
CONST1	DCW	@ERROR LISTING @
FOURTH	DCW	1,2,3,4,5,6
DATE	DC	@ DEC 9 @
	A	2,9,6,2,,7,2,0
READIN	DA	1X,8,0,
DATE	DCW	@FEBRUARY@

Section 3. Answers and Solutions

1. d. is passing the reading station.
2. c. DA (Define Area) operation code.
3. 49 positions of core storage are required: 48 positions are used for data and a 49th position contains a group mark with a word mark.
4. Answers a and b are both positive.

   a. $\overset{B}{A}_{2}$ equals plus 2

   b. $\overset{A}{}_{2}$ equals plus 2

5. c. both fields must contain a word mark in their leftmost position.
6. c. a branch to the address of the instruction specified by line 01 can be performed by a later program step.

Section 4. Answers and Solutions

1. d. MLCWA (Move Characters and Word Mark from A Field). This operation moves the word mark along with the edit control word to the print field. The word mark is required in the control word for editing.
2. Fields after edit

   $bbbbb.01–
   $6,436.25

3. c. can appear before or after the printing instruction. If the CC instruction appears before the print instruction, a skip-after-printing d-modifier is used. If the CC instruction appears after the print instruction, an immediate-skip d-modifier is used.
4. No.
5. a. is not true.
6. No; a CC instruction must be executed to provide the skip.

7.

Label	Operation	OPERAND
	MLC	SUM-4,,TOT

8. b. STEPX.

9.

Label	Operation	OPERAND
	BCE	START,CODE,,R

10.

Line	Label	Operation	OPERAND
01		ORG	4,0,1
02	PRAREA	DA	1X,7,,G DEFINE THE PRINT AREA.
03	TOTAL		1,,7, TOTAL FIELD
04	RDAREA	DA	1X,5,,G DEFINE THE CARD READ AREA
05	GROSS		1,,5 GROSS EARNINGS FIELD
06	START	R	1,,RDAREA READ A CARD
07		A	GROSS,TOTAL ADD GROSS TO TOTAL
08		BLC	PRINT HAS THE LAST CARD BEEN PROCESSED?
09		B	START BRANCH UNCONDITIONALLY TO START
10	PRINT	W	PRAREA PRINT THE TOTAL
11		H	HALT (END OF PROGRAM).
12			

11. b. sets an indicator which may be tested by subsequent instructions.

12. a. Unequal
    b. Equal
    c. Unequal
    d. Unequal

13.

Label	Operation	OPERAND
	CS	999 CLEARS 999-900
	CS	899 CLEARS 899-800
	CS	625 CLEARS 625-600
	CS	501 CLEARS 501-500
	CS	200 CLEARS 200 ONLY

Section 5. Answers and Solutions

1. a. program instructions
   b. tables
2. Yes
3. a. access code
   b. core sector address
   c. sector count
4. b. the disk control field
5. On the read operation the core sector address was incremented by the number of sectors read and the sector count was decremented to zero. The core sector address and sector count must be restored to their original values before the record can be written back to disk storage.
6. d. four
7.

Label	Operation	
	BIN	TEST,X

8. d. after the first disk (seek, read, write, etc.) instruction following a seek instruction.
9. b. 10 tracks

Index

IBM 1440 Printed in U.S.A. C26-5607-0

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York