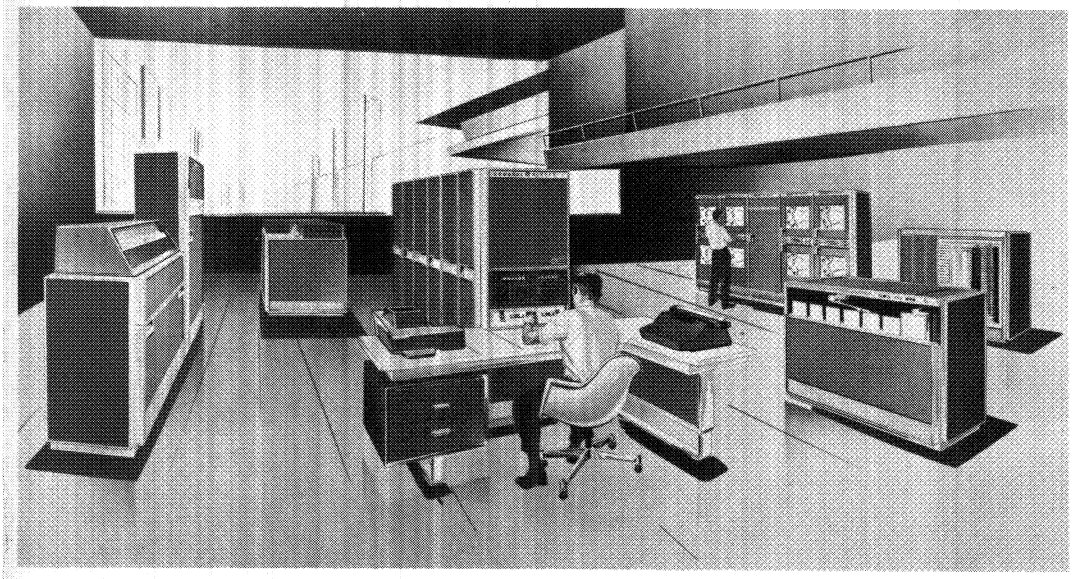
CPB 126A

GE 225



PROGRAMMING MANUAL

(INCLUDING PROGRAMMING NOTES)

GENERAL ELECTRIC

GE 225

GE 225

PROGRAMMING MANUAL

(INCLUDING PROGRAMMING NOTES)

GENERAL ELECTRIC
COMPUTER DEPARTMENT
PHOENIX, ARIZONA

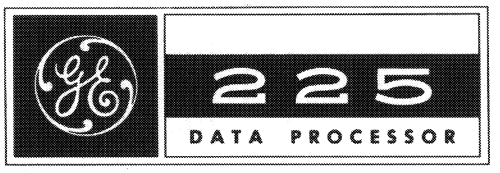


TABLE OF CONTENTS

A. INTRODUCTION 1

B. THE GENERAL ELECTRIC 225 SYSTEM 3
- CENTRAL PROCESSOR 3
- CONTROL CONSOLE 3
- CARD READER — CARD PUNCH 4
- PAPER TAPE READER — PAPER TAPE PUNCH 5
- DATA MATING FUNCTION 5
- HIGH SPEED PRINTER SUB-SYSTEM 6
- MAGNETIC TAPE SUB-SYSTEM 6
- MASS RANDOM ACCESS FILE SUB-SYSTEM 7
- DOCUMENT HANDLER SUB-SYSTEM 9

C. CENTRAL PROCESSOR ORGANIZATION 11
- REPRESENTATION OF INFORMATION 11
  - DATA WORDS 11
  - INSTRUCTION WORDS 12
- THE DATA MATING FUNCTION (CONTROLLER SELECTOR) 12
- ARITHMETIC AND CONTROL REGISTERS 14
- AUTOMATIC ADDRESS MODIFICATION 15
- CYCLE OF OPERATION 17
- PRIORITY INTERRUPT 17

D. INSTRUCTION REPERTOIRE 21
- ARITHMETIC 21
- DATA TRANSFERS 23
- SHIFT OPERATIONS 24
- INTERNAL TEST-AND-BRANCH 26
- CONSOLE OPERATION 28
- PAPER TAPE INPUT-OUTPUT 29
- PUNCHED CARD INPUT-OUTPUT 29
- HIGH SPEED PRINTER SUB-SYSTEM 33
- MAGNETIC TAPE SUB-SYSTEM 36
- MASS RANDOM ACCESS FILE SUB-SYSTEM 40
- DOCUMENT HANDLER SUB-SYSTEM 41

E. THE GENERAL ASSEMBLY PROGRAM 45
- GENERAL DESCRIPTION 45
- PSEUDO-INSTRUCTIONS 45
- THE GE 225 CODING SHEET 46
- RELATIVE ADDRESSING 46
- PSEUDO-INSTRUCTION USAGE 48
- ILLUSTRATIVE PROBLEM 51

F. CONTROL CONSOLE OPERATION 53
- INDICATOR PANEL 53
- CONTROL PANEL 55

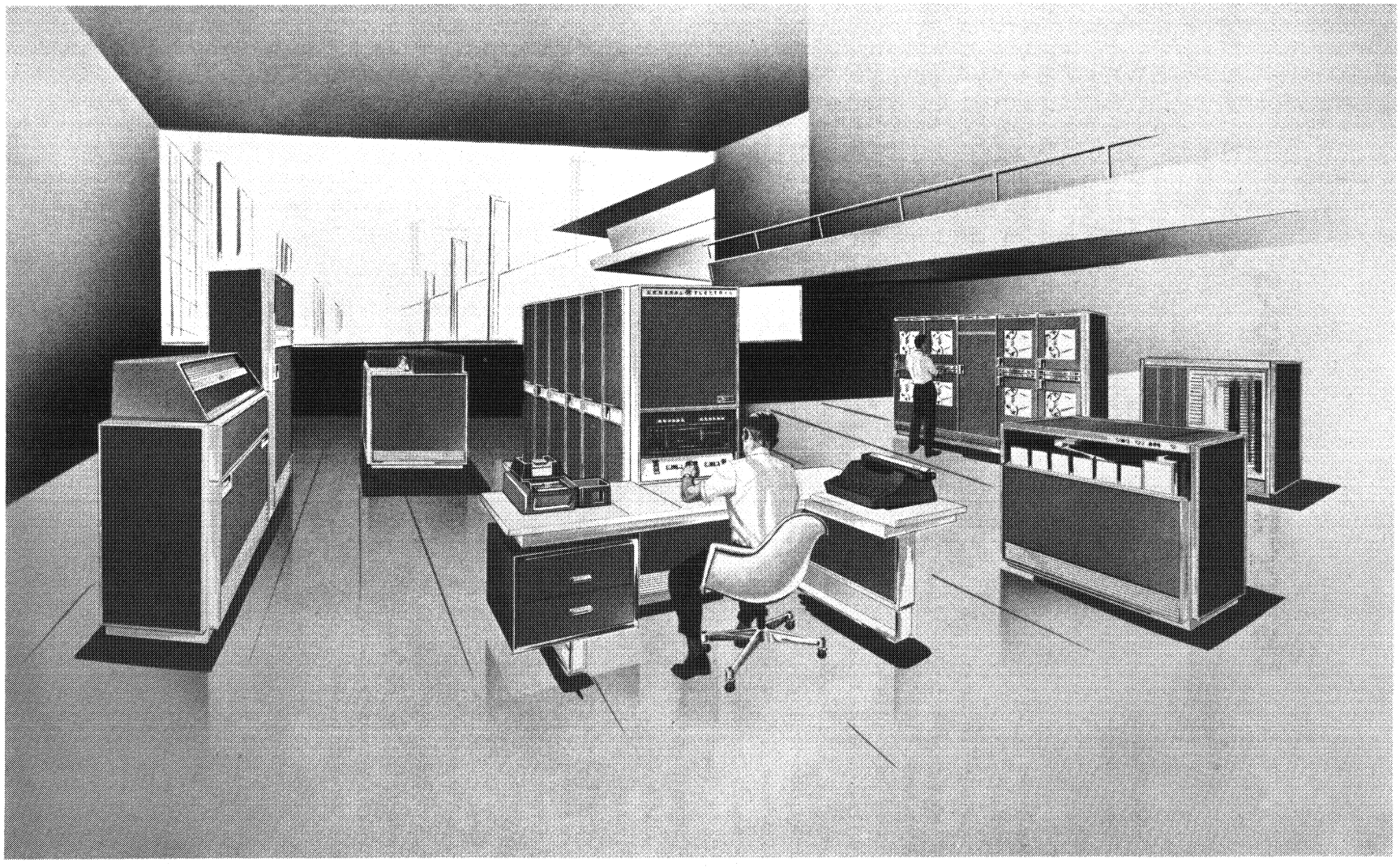
G. SYSTEM ERROR CHECKING AND RECOVERY FEATURES 57

H. PROGRAMMING NOTES 59
PROGRAMMING MACHINE CALCULATIONS 59
PROGRAMMING LOGICAL DECISIONS 73
MODIFICATION WORD PROGRAMMING 83
PROGRAMMING FOR SUBROUTINE USAGE 87
PROGRAMMING FOR CONSOLE CONTROL 91
PUNCHED PAPER TAPE OPERATIONS 97
PUNCHED CARD OPERATIONS 101
PROGRAMMING PRINTED REPORTS 127
MAGNETIC TAPE OPERATIONS 141
MASS RANDOM ACCESS FILE OPERATIONS 149
MAGNETIC DOCUMENT HANDLER OPERATIONS 155

I. APPENDIX 165
NUMBER SYSTEMS 165
BINARY ARITHMETIC 166
BINARY-DECIMAL CONVERSION TABLE 168
REPRESENTATION OF CHARACTERS 169
INSTRUCTION FORMATS 170
OCTAL LIST OF INSTRUCTIONS 172
STANDARD FLOW CHART SYMBOLS 175

LIST OF ILLUSTRATIONS

Figure 1 GE 225 Central Processor 3
Figure 2 GE 225 Control Console 4
Figure 3 Card Reader 4
Figure 4 Card Punch 4
Figure 5 Paper Tape Reader 5
Figure 6 Paper Tape Punch 5
Figure 7 High Speed Printer Sub-System 6
Figure 8 Magnetic Tape Sub-System 7
Figure 9 Mass Random Access File 8
Figure 10 Twelve-pocket Document Handler 8
Figure 11 Two-pocket Document Handler 9
Figure 12 E13B Font 9
Figure 13 Large Configuration 13
Figure 14 Register Relationships 16
Figure 15 Central Processor Operating Cycle 18
Figure 16 GE 225 Coding Sheet 47
Figure 17 GE 225 Coding Sheet Example 50
Figure 18 Control Console 54
Figure 19 Central Processor 59
Figure 20 Control Console 91
Figure 21 Paper Tape Reader 97
Figure 22 Paper Tape Punch 97
Figure 23 Card Reader 101
Figure 24 Card Punch 102
Figure 25 High Speed Printer Sub-System 127
Figure 26 Magnetic Tape Sub-Systems 141
Figure 27 Mass Random Access File Sub-System 149
Figure 28 Twelve-pocket Document Handler 155
Figure 29 Two-pocket Document Handler 155



GE 225 Information Processing System

A. INTRODUCTION

```
READ TIME~CARDS.
COMPUTE FEDERAL~TAX FROM (GROSS~
PAY—(DEPENDENTS * 13.00)) * 0.18.
ADD FICA TO YDT~FICA.
GO TO CALC~STATE~TAX.
IF YTD~FICA GR 120.00, GO TO
REIMBURSE~FICA.
```

```
A (I, J)=SIN (A—B)+(33.33* (Q) (P—1) #3)
—LOG (A—Q).
DO STRESS~LIMIT FOR X, X1, X2, X3,
X4, X5.
MOVE Q~VALUE (L, M) TO MOD~P(J, K).
```

This is the General Electric Common Language - the language understood by all General Electric computers when programs are prepared using the new General Compiler. It is no longer necessary for the systems analyst to concern himself with the intricacies of the machine language of each computer with which he deals. The new General Electric common language, specifically designed to meet the needs and requirements of both business data processing and scientific applications, permits the systems analyst to express his solution to a problem in a language which is advantageous for the problem - the language of narrative statements having the wordage and syntax of English sentences or the language of mathematics employing convenient decimal numbers. Arguments regarding the relative ease or difficulty of programming a particular computer no longer have validity when the General Electric Common Language is used.

Despite the reduction of time consuming computer coding through the use of problem oriented languages, some of the personnel at every computer installation may require a detailed understanding of the actual computer language. It is to these programmers who must learn the art of communicating directly with the GE 225 Information Processing System that the present manual is dedicated. If the need for such programmers seems unnecessary in view of the General Compiler, several good reasons for this need are:

1. The program which is executed by the computer is in its own language; hence, machine testing (debugging) a program requires a knowledge of the actual machine instructions.
2. Making small changes to existing programs can often be more efficiently done by patching with direct machine coding.
3. Each installation should have some expert programmers who are capable of contributing to the extension and improvement of the automatic coding techniques and problem oriented languages which are in use.

A special facility has been provided with the GE 225 to permit the programmer to code for this computer employing symbolic notation in a concise tabular form, yet retaining both the single address format and the general structure of the actual computer instruction words. This symbolic program is read into computer memory along with a General Assembly Program. The General Assembly Program is a basic assembly program with extensive error checking features and provision for effective program modification. Its output is a running program prepared in the absolute binary code of the computer. One instruction written in General Assembly Program language is usually translated into one computer instruction.

Programs produced by use of the General Compiler are also expressed in General Assembly Program language. Therefore, the General Assembly Program will be the basic tool of the GE 225 programmer, and this programming manual will be oriented to its use. Absolute machine bit configurations will be given only for the sake of completeness and for reference in unusual program debugging situations. Complete details of the General Assembly Program are given in a later section.

THE GENERAL ELECTRIC 225 SYSTEM

The GE 225 Information Processing System places emphasis on the total systems concept and flexibility of computer hardware organization as the answer to the increasing complexity of todays's applications. A modern, fully transistorized central computer provides an economical basis for extremely flexible hardware configurations ranging from simple card or paper tape input-output systems, suited to scientific laboratories or small business users, to sophisticated arrangements of high speed printers, magnetic tape units, mass random access file memories and direct information links with communication and data collection networks, suited to large installations employing the latest techniques of completely integrated data processing.

CENTRAL PROCESSOR

The GE 225 Central Processor is a single address, stored program, general purpose digital computer which operates primarily in a straight binary mode but processes both alphanumeric and binary information. The Central Processor performs the computation (arithmetic), fast random access memory storage and control functions for the GE 225 System. The programs to be executed and the data to be immediately operated upon are stored in a magnetic core memory wherein each core, depending on direction of magnetization, represents a binary digit (bit) of an instruction or data word; a word being the basic unit of addressable information in the memory. The memory is thought of as consisting of a number of individual cells, each capable of holding one word; each cell having a unique designation or address. A more complete description of Central Processor characteristics is the subject of the following section.

CONTROL CONSOLE

The control exercised by the console is of a manual nature, in distinction to the control function performed by the Central Processor, and need not be

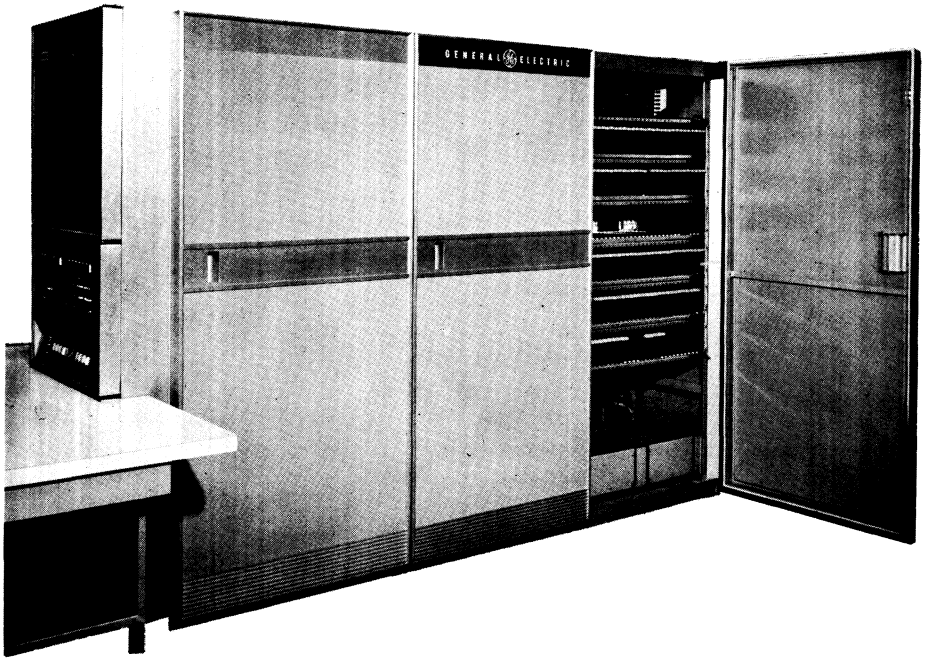


Figure 1 GE 225 Central Processor

Figure 2 GE 225 Control Console

necessarily operative in normal program execution. This manual control is concerned with initially loading the program into memory, starting the execution thereof, monitoring the progress of the program primarily via the console typewriter, and occasionally stopping the program for checking or other purposes. Typing out on the Console Typewriter proceeds at the rate of 10 characters per second. Information may be numeric or alphanumeric. Other indications of Central Processor operating status are available in the form of lights and indicators on the Console panel. In addition to the typewriter serving as an output device, switches on the console allow the manual entry of information into the Central Processor. To this extent, the console thus serves as an input device as well.

CARD READER — CARD PUNCH

Punched cards have been widely used for many years as a file storage medium. These cards are usually prepared in a separate clerical operation to record transaction data in business applications but may well be produced as computer output.

The Card Reader is a basic on-line input device used in the 225 System to read punched card information into the Processor's memory. Reading of standard, 80-column punched cards may proceed at a maximum rate of 400 cards per minute. Information may be recorded on these cards in either binary (column) or standard Hollerith (alphanumeric) code, the interpretation of the data punched in the card being determined by the particular mode of the "read cards" instruction being executed by the processor. Since the card reader is considered a basic unit, it is controlled by the control function of the Central Processor, and external control switches and indicators associated with the card reader appear on the control console.

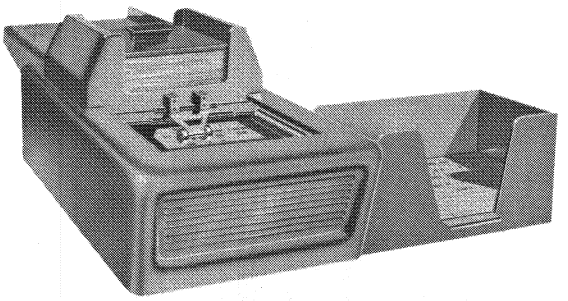


Figure 3 Card Reader

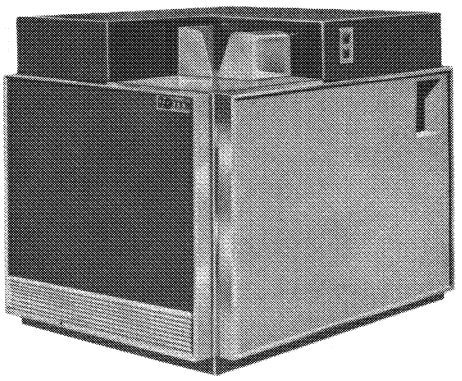


Figure 4 Card Punch

It was mentioned that punched cards may be obtained as a result of computer output during processing operations. The on-line Card Punch associated with the 225 System performs this function. Card punching may also be in either binary or Hollerith code at the rate of 100 cards per minute. As was the case in card reading, the Card Punch is controlled by the control function of the Central Processor.

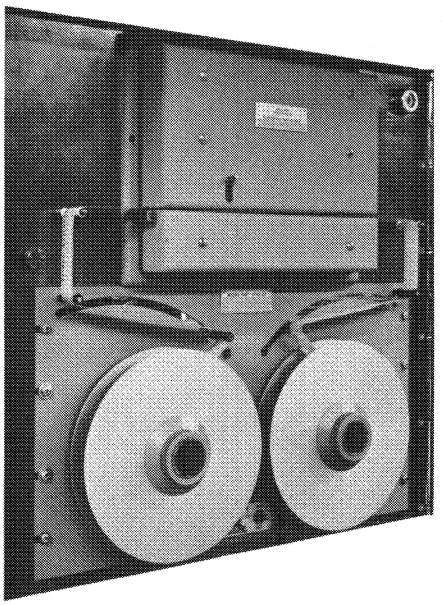


Figure 5 Paper Tape Reader

PAPER TAPE READER — PAPER TAPE PUNCH

Perforated paper tape is in fairly wide use in diversified business and scientific operations. It is commonly produced by a distinct clerical operation from such business equipment as typewriters (flexowriters), billing machines, desk calculators and cash registers. Each of these devices are used to create transaction records in machine sensible form. Punched paper tape is a particularly popular input medium with scientific laboratories. Paper tape is also used as a data communication medium during teletype operations.

The High Speed Paper Tape Reader is an on-line input device of the 225 System that reads punched paper tape under program control at the rate of either 100 characters per second or 1000 characters per second at operator's option. Standard punched paper tape is 8 channel; other formats are optional.

In some applications it may be desirable to record computer output information on punched paper tape. Towards this end a Paper Tape Punch has been included as an on-line output device for 225 System users. Information may be thus punched at a rate of 60 characters per second. Again, standard punched paper tape is 8 channel; other formats are optional.

DATA MATING FUNCTION

The ability of the GE 225 to incorporate a variety of peripheral devices is accomplished by means of a common connecting device, or "data mating function". The data mating component is a common control and transfer point for such peripheral units as the high speed printer, the magnetic tape systems, the mass random access file memories and the direct data links.

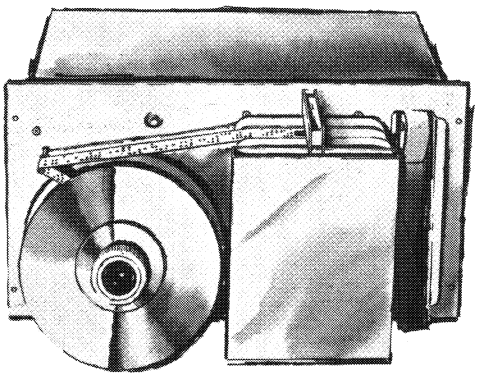


Figure 6 Paper Tape Punch

Through the use of convenient plug-in connectors, associated peripheral units can be connected in varying configurations and interchanged according to the immediate requirements of the system. This function allows for the easy addition of peripheral equipment as the needs of a particular installation grow and for the addition of new or improved input-output devices with little, if any, logic or wiring changes. Information can be transferred through the data mating unit at the rate of 50,000 words per second.

HIGH SPEED PRINTER SUB-SYSTEM

In applications where a vast amount of data is produced by the Central Processor for visual output, the console typewriter will obviously not suffice. To fill this need, an on-line High Speed Printer has been incorporated into the 225 System.

The printer sub-system consists of the Printer Mechanism and a Printer Controller. Information to be printed on one line is transferred from the Processor's memory to the Printer Controller, which stores this information in a "buffer" where it will be scanned many times during the printing process. After this transfer is complete, printing commences completely "off-line" from Central Processor computation and no further interruption of Processor operation is necessary. After the line has been printed, the paper will be slewed (spaced or advanced) for one or more lines to provide vertical format arrangement. Editing functions can be accomplished by a special mode of the print instruction. Horizontal data arrangement and additional editing functions will be performed by the Central Processor prior to printing. 600 lines of alphanumeric information, up to 120 characters per line, may be printed in one minute.

MAGNETIC TAPE SUB-SYSTEM

Magnetic tape is one of the chief items that distinguishes modern electronic data processing systems from earlier card calculating devices. Tapes used in the 225 System consist of thin strips of mylar plastic 1/2 inch wide, 1 mil thick and 3600 feet in length, coated with magnetizable iron oxide. Information is recorded on the tape as a series of tightly packed magnetic "spots". Advantages of magnetic tapes over other types of file storage media include the high speed with which information may be placed on or retrieved from them (provided that the information is processed sequentially) and reusability.

Figure 7 High Speed Printer Sub-System

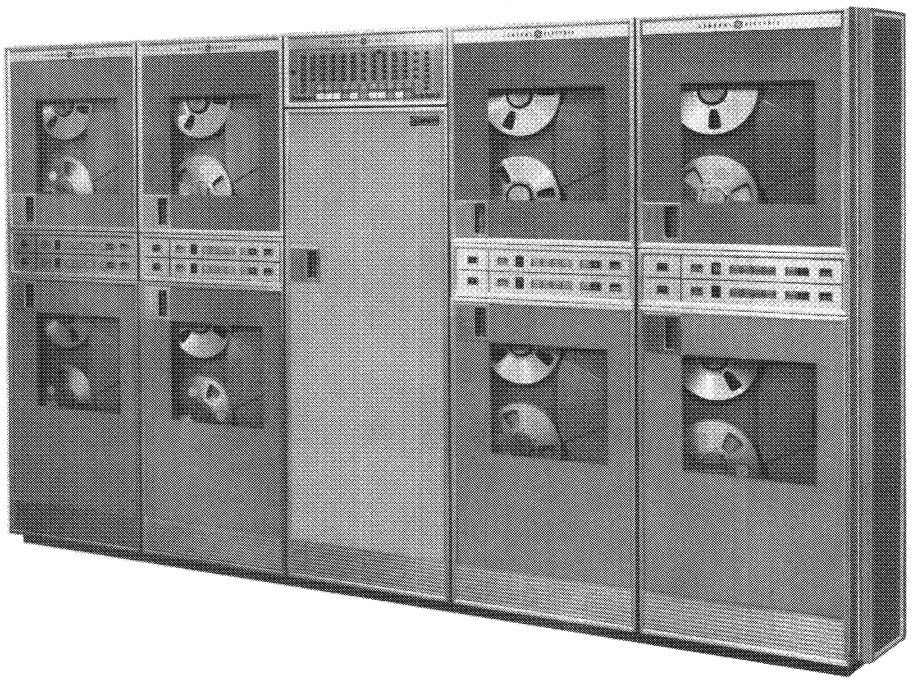


Figure 8 Magnetic Tape Sub-System

Theoretically, up to 8 magnetic tape sub-systems may be included in a 225 System; however, one or two tape sub-systems will be a more common arrangement. Each sub-system will contain a Magnetic Tape Controller and from one to eight Magnetic Tape Handlers. The function of the Controller is to relieve the Central Processor from the constant monitoring necessary during reading and writing operations. The Controller, therefore, will be capable of controlling the operation of the Magnetic Tape Handler simultaneously with Central Processor computations. Of course, these computations will have to be occasionally interrupted while an accumulation of information from tape is transferred to the Processor's memory as during a magnetic tape read operation; this later type of operation being more fully discussed in a later section on the automatic priority interrupt feature.

In addition to the normal functions of reading and writing in either binary or alphanumeric (binary coded decimal) mode, such housekeeping functions as advancing or backing up the tape, detection of the end of tape and the end of file, and the rewinding of tape are performed by this system. It should be mentioned that a given Tape Controller may direct either a reading or writing operation, but not both simultaneously. The basic unit of information on magnetic tape is the "tape record" consisting of from 1 to 16,384 words. This information is transferred to or from tape at a rate of 15,000 characters per second.

MASS RANDOM ACCESS FILE SUB-SYSTEM

To take advantage of the high speed data processing facility offered by magnetic tape systems, all files (both master and transaction) must be sorted into some sequence before processing begins. In certain important business applications this latter operation is either not practical, or significant time savings will result if it can be eliminated. To this end, the GE 225 System includes an on-line, large capacity, random access, file storage device. Basically the Mass Random Access File consists of several large magnetic discs arranged vertically on a rotating shaft. Several million characters, usually representing master files, are randomly accessible with this device.

The Mass Random Access File sub-system consists of a Controller unit and from one to four Mass Random Access File memory units. The function of the Controller is to relieve the Central Processor from the monitoring required for reading and writing operations. The Controller checks the parity of words received from memory and checks for errors when reading or writing on a disc. The Central Processor is free for other computations except for occasional interruptions as when an accumulation of information from the file is transferred to the Processor's memory during a read operation.

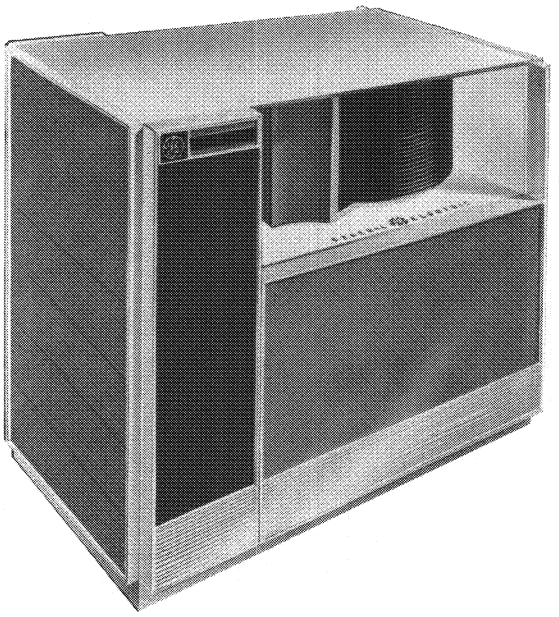


Figure 9 Mass Random Access File

The basic unit of information on a magnetic disc is the "disc record" consisting of 64 words. Each word in this record is recorded as an image of the word as it appears in the Central Processor memory. Thus, both binary and alphanumeric (binary coded decimal) configurations are retained without change. Information is recorded serially in 256 circular tracks on each side of a disc: eight 64-word records in each of 128 inner tracks, and sixteen 64-word records in each of 128 outer tracks. Eight read-write heads on the positioning arm associated with each disc assure a maximum positioning time of 200 milliseconds. Information transfer rates are 250,000 bits per second for the inner tracks and 500,000 bits per second for the outer tracks. Latency time, the time required for information in a track to reach a read head, is a maximum of 100 milliseconds. Mass Random Access File memory units are offered in two sizes: a 16-disc file with a capacity of 98,304 records and a 64-disc file with a capacity of 393,216 records.

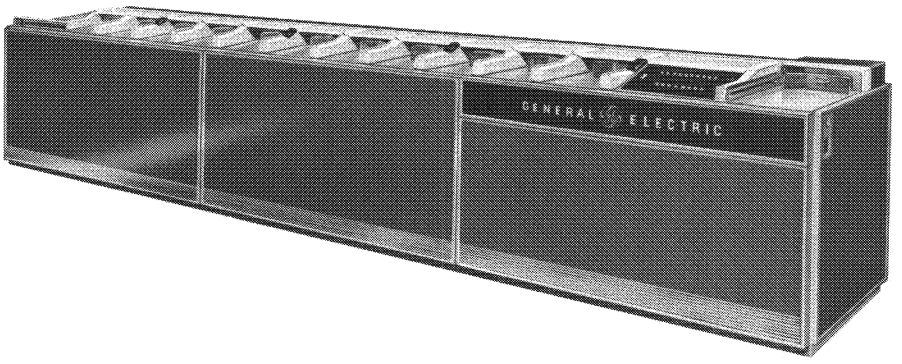


Figure 10 Twelve-pocket Document Handler

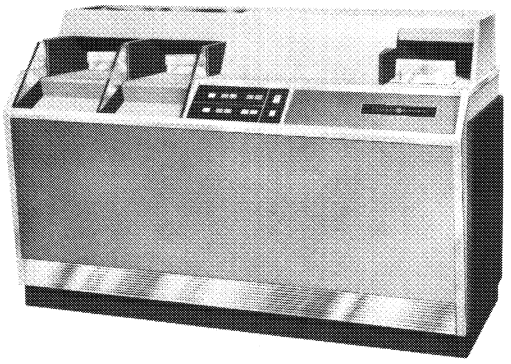


Figure 11 Two-pocket Document Handler

DOCUMENT HANDLER SUB-SYSTEM

A recent breakthrough in the field of data processing has been the development of magnetic ink character recognition by General Electric. The most common application of MICR equipment thus far has been in the area of banking; however, new uses of this powerful tool will soon be seen in many other business data processing applications.

Both the 12-pocket and the 2-pocket GE Document Handlers are available. The Document Handler is an optional on-line input device to the 225 System that is capable of reading paper documents printed with E13B font and magnetic ink, and transferring this information to the Central Processor, at the rate of 1200 items per minute. These documents may vary in quality, size and degree of mutilation. A controller unit permits concurrent operation with other peripherals and the Central Processor. The Controller will accommodate both types of handlers in any combination not to exceed two. The Document Handler may also be used off-line for document sorting operations.

The Document Handler can recognize 14 characters: the ten decimal digits and four special symbols called Cue characters. Cue characters are normally used to separate fields of decimal digits (to identify dollar amount fields and identification fields).

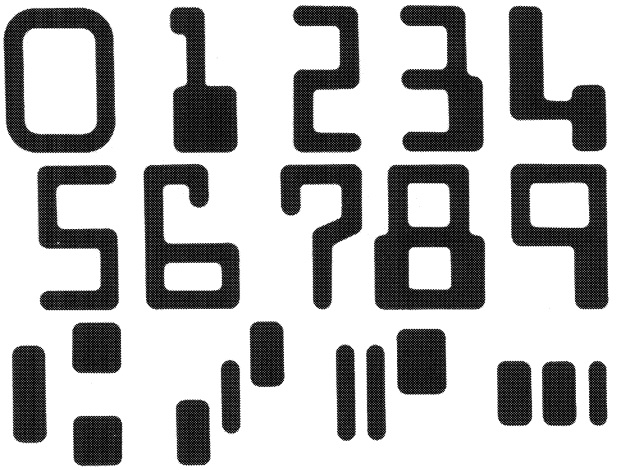


Figure 12 E13B Font

C. CENTRAL PROCESSOR ORGANIZATION

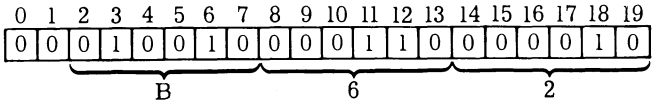
REPRESENTATION OF INFORMATION

The memory of the Central Processor of the GE 225 System is made up of magnetic cores, each core storing one bit of information. All information held within the memory of the Central Processor must exist in the form of words, a word being the basic unit of addressable information in the memory. The memory is organized into a number of individual cells each capable of holding one word of information and each labelled with a unique designation or address. Each word is considered to consist of 20 bits although a word stored in memory actually has 21 bits since a parity check bit is computed and stored as a transfer to memory occurs. Available memory sizes are 2048 words, 4096 words, 8192 words and 16,384 words. The core memory provides storage for both data and computer instructions.

A. Data Words

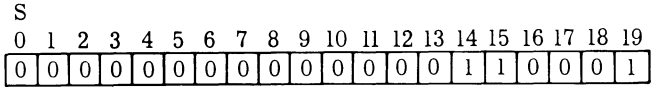
One basic method of entering information into the Central Processor is by reading punched cards. The information may be punched in either 80-column Hollerith (alphanumeric) code or binary (column) form. For this reason, the GE 225 can function as either an alphanumeric machine with binary capabilities or as a normal binary computer; so that, by means of the program which he prepares, the user may switch between modes of operation to take advantage of the particular characteristics of a given application. On those occasions when a conversion from alphanumeric to binary or from binary to alphanumeric representation of data is desirable, subroutines are automatically provided by the General Assembly Program.

If information is punched in Hollerith code, upon reading a card in the alphanumeric (decimal) mode, each character is converted into a six-bit binary coded decimal configuration. Thus, three alphanumeric characters will occupy 18 of the 20 bit positions of a memory word. (See the Appendix for an equivalence table of the Hollerith and binary coded decimal codes.) In this sense, an alphanumeric data word consists of three alphanumeric characters. Double length operations also permit the automatic handling of symbolic codes (account numbers, stock numbers, etc.) of six alphanumeric characters. These convenient word sizes reduce the need for elaborate partial word facility. The 26 alphabetics, 10 numerics and 11 special characters can be expressed by six-bit binary coded decimal configurations. As an example, the symbolic code B62 in binary coded decimal form would appear as follows:

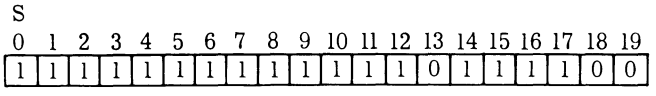


As can be seen in the above illustration, the most significant bit positions 0 and 1 are free to hold other information while bit positions 2 through 19 store the three six-bit binary coded decimal digits.

The GE 225 operates arithmetically in the binary mode in order to capitalize on the advantages of high speed, versatile command structure and ability to handle data in binary as well as decimal and alphabetic forms. A binary data word consists of 19 binary digits plus a sign bit. For example, a pure binary 49 may be represented as follows:



Special operations are provided to handle computations on double length data words (two adjacent memory words); that is, 38 bits plus the sign. The sign of the value stored in straight binary notation will be in bit position zero. A 0 bit represents a plus; a 1 bit represents a minus. In addition to the presence of a 1 bit in the sign position, a negative number is represented by the 2's complement of the corresponding positive number. For example, a pure binary -68 may be represented as follows:



Negative equivalents may be generated by a special instruction which performs this function.

The largest positive numeric value that may be stored in 19 bit positions is $2^{19}-1$, equivalent to the decimal number 524,287. This may be verified by reference to the binary-decimal conversion table in the Appendix. A numeric (decimal) data word may, therefore, be regarded as consisting of approximately 5-1/2 decimal digits. Double length operations permit the handling of numbers of up to 11 decimal digits.

Subroutine packages automatically provided by the General Assembly Program make the necessary conversions when it is desired to work in the decimal (BCD) mode before and after the binary arithmetic operations (add, subtract, multiply, etc.). In particular, a conversion will be necessary from straight binary to binary coded decimal (BCD) before the result of an arithmetic operation may be transferred to the console typewriter, high speed printer or paper tape punch. Information output to punched cards or magnetic tape may be in either binary or BCD notation. It should be noted that when information is output in straight binary form, the intention is to re-enter this information as input to a subsequent process. The necessity of input and output conversions is thereby avoided.

B. Instruction Words

An instruction word in the GE 225 is a single address word consisting of 20 bits. The basic format of the instruction word is as follows:

Bits 0 through 4 designate the operation which is to be performed, bits 5 and 6 determine whether or not the instruction is to be automatically modified, and bits 7 through 19 indicate the operand address.

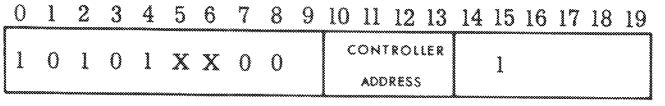
The significance of the instruction bits 7 through 19 will vary depending upon the particular operation specified by the operation code. They will frequently be used to elaborate the exact operation to be performed when no memory address is associated with the execution of the instruction. For example, word transfers between internal registers do not require an operand address. Other instructions, such as shift commands, require only bit positions 15 through 19 to indicate length of shift. Such use of available bit positions of the operand address field extend the instruction repertoire of the GE 225 far beyond the maximum of 32 operations which would be allowed by the five bit positions of the operation code. The type of instruction which makes use of these additional bits to further specify the operation is called a "general" instruction. Complete detailed information on instruction formats is given in the appendix. An explanation of micro-programming capabilities is also included for the advanced programmer. It should be noted, however, that the user does not code his programs by means of binary notation except for unusual debugging situations. Under normal circumstances machine running programs are created by use of the General Assembly Program, wherein operations are expressed by mnemonic codes.

Bits 5 and 6 of an instruction word may be employed for the automatic modification of the instruction before its execution. One of three modification words (also known as X Registers) may be selected and the contents added to the operand address portion of the instruction word. These three words are memory locations 1, 2 and 3. Bit combination 01 selects memory location 1; bit combination 10 selects memory location 2; and bit combination 11 selects memory location 3. A 00 combination indicates no address modification.

THE DATA MATING FUNCTION (CONTROLLER SELECTOR)

The Data Mating Function, or Controller Selector, is the focal point for information transfers between the Central Processor and the input-output peripheral units other than punched card, paper tape and console typewriter. The decision as to which of the associated peripheral equipments is to be granted memory access during any given word time is made by the Controller Selector. The basis for this decision is the priority assigned to each controller connected to the Controller Selector, which in turn is dependent upon the address associated with each controller. The assignment of an address to a controller is the result of a manual modification. There is an inverse relationship between an assigned address and the priority level; thus, the highest priority is associated with the controller which has been assigned address #0, the next highest to the controller addressed #1, etc.

The input-output instructions which relate directly or indirectly to the data mating function fall within the "general instruction" classification. This "general" instruction consists of three words. The first word has the following format:



This word selects the control unit of the particular peripheral that is to perform the desired operation. Bit positions 0 through 4 and bit positions 7, 8 and 15 must be as shown above. Bit positions 10 through 13 hold the binary address of the controller to be selected. At present, only eight peripheral controllers are intended to be on-line through the Controller Selector. Bit position 10 is available for expansion. The bit positions that are not filled in indicate that they are not significant during the execution of the command.

The instruction words in the two following memory locations are sent directly to the selected control unit upon execution of the first word of the general input-output instruction. These two instruction words contain the necessary information to indicate the kind of operation the peripheral is to perform and the starting memory location where information is to be stored or extracted. If, for example, a magnetic tape controller had been addressed and selected, the contents of the two command words might contain such information as:

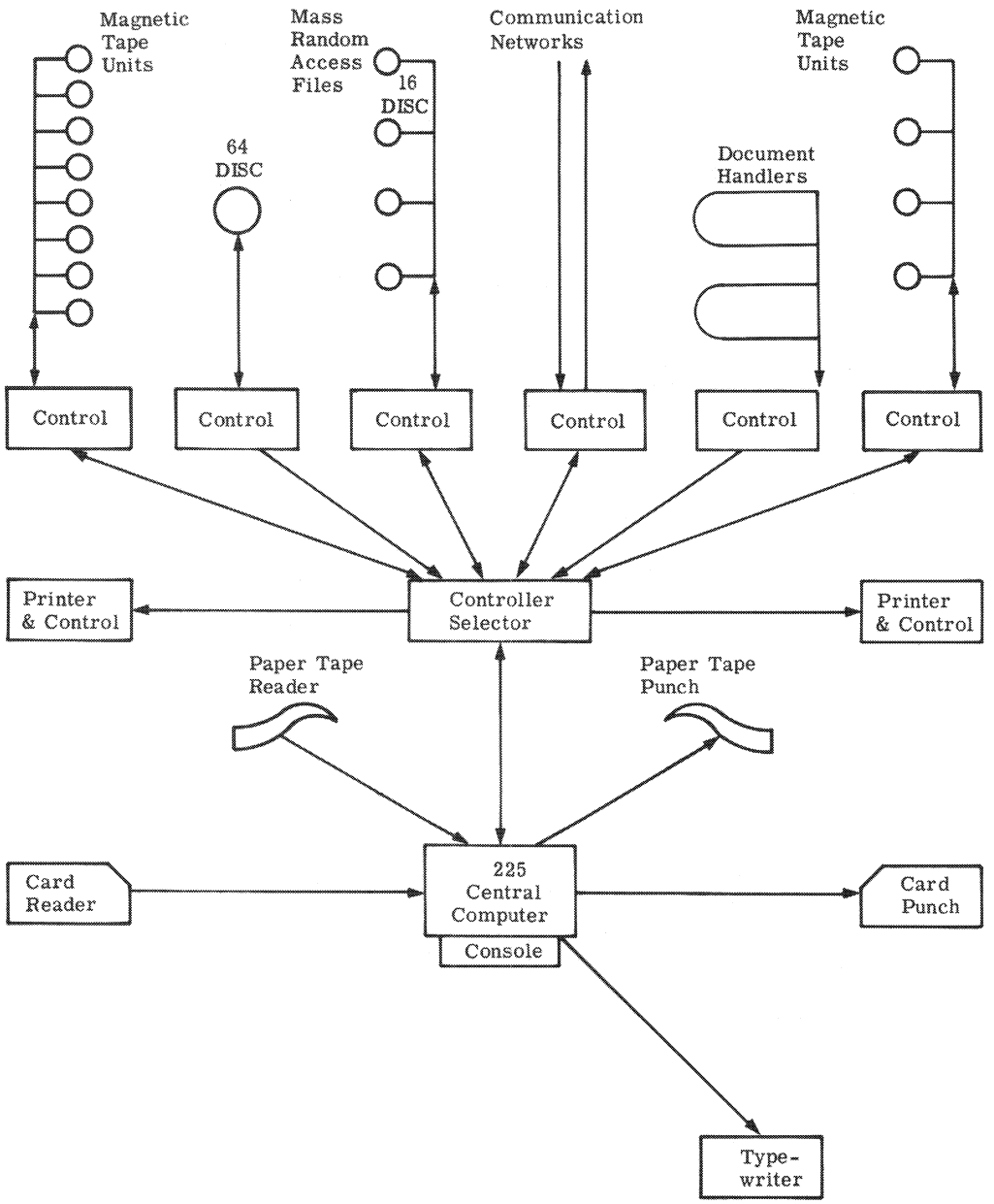


Figure 13 Large Configuration

the operation to be performed (e.g., the controller is to read a magnetic tape),

the designation of the magnetic tape unit,

the number of words to be read, and

the starting address in the memory where the information is to be stored.

ARITHMETIC AND CONTROL REGISTERS

1. M Register

All information written into or out of the Central Processor's magnetic core memory must first pass through the 21-bit M Register. The M Register is thus a focal point for information transfers among 225 System units. The 21 bits of the M Register include 20 information bits and a parity check bit. When a word (20 bits) enters the M Register in preparation for writing into memory, the number of 1 bits are counted. If the total is an even number, a one bit is generated for the 21st bit position. If the total is odd, a zero is generated. The full 21 bits are then stored away in memory. When a word is read from memory into the M Register, another bit count is made to determine that there are still an odd number of 1 bits in the 21 bits of the memory word. Should this check reveal an even number of 1 bits, a parity error will be indicated.

2. B Register

All information transferred from memory (via the M Register) to other internal registers and components of the Central Processor must first pass through the 20-bit B Register. In this manner, the B Register serves as a buffer between the arithmetic and control components and the memory and M Register. The memory and M Register are then free to be accessed and utilized simultaneously with Central Processor operations not requiring memory usage. For example, multiply and divide instructions, after initial memory access, consist simply of a series of addition and shifting operations. In arithmetic operations, the B Register holds the addend for addition, the subtrahend for subtraction, the multiplicand for multiplication and the divisor during division. It is also used in the execution of certain data transfer commands.

3. A Register

The A Register is a 20-bit register which serves as the accumulator for the Central Processor. The A Register performs the following functions:

a. Holds the augend during addition.

b. Holds the sum after addition.

c. Holds the minuend during subtraction.

d. Holds the result after subtraction.

e. Holds the most significant half of the product after multiplication.

f. Holds the most significant half of the dividend before division.

g. Holds the quotient after division.

h. Holds the most significant half of the double length word after the execution of all double word length instructions.

i. Holds a word which has been transferred from memory or which is to be transferred to memory.

j. Holds the word on which extraction is performed during the execution of the Extract instruction.

k. Holds the word to be shifted during various shift instructions.

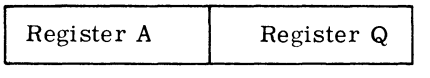
l. Holds a word which is to be transferred to another register or which is to be modified in some way during the execution of various DATA TRANSFER COMMANDS.

m. Holds the word which determines future action during the execution of various branch instructions.

Manual input to the A Register is possible from 20 console switches provided for this purpose. An example of the use of these switches would be in initial program startup when it is necessary to get the first instruction into the machine. Console operation is discussed more fully in a later section.

4. Q Register

The Q register is a 20-bit register which acts with the A Register to form a double word length (38 bits plus sign) accumulator during the execution of double word length instructions. Information is not transferred directly from memory into the Q Register but is transferred through the A Register into the Q Register.



The Q Register performs the following functions:

a. Holds the least significant half of the double length word during the execution of double length load and store instructions.

b. Holds the least significant half of the result after multiplication.

c. Holds the least significant half of the dividend prior to division.

d. Holds the remainder after division.

e. Holds the least significant half of the augend prior to double addition and the least significant half of the sum afterwards.

f. Holds the least significant half of the minuend prior to double subtraction and the least significant half of the result afterwards.

g. Holds the least significant half of the information to be shifted during double shift instructions.

h. Can be shifted right or left along with the N and A Registers in special shift instructions.

5. Arithmetic Unit

The Arithmetic Unit serves two functions. It performs the arithmetic calculations specified by the operation code in the I Register during arithmetic operations. It serves as a transfer bus for data words going between the A Register and memory (via the M Register) and for instruction words going to the I Register.

6. N Register

The N Register is a six-bit register which is used as a single character buffer between the computer and the Console Typewriter, Paper Tape Reader or Paper Tape Punch. This permits the appropriate input-output processes to occur simultaneously with other Central Processor operations. Information is transferred directly between the N Register and the A Register by means of several shift instructions.

7. I Register

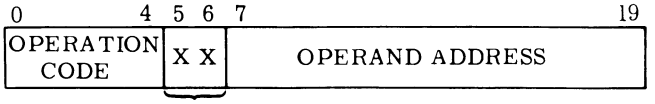
The I Register is the instruction register. It holds the 20 bits of the instruction word during execution of a computer command. In the execution of all instructions, the left-most five bits indicate the operation which is to be performed; the next two bits refer to the automatic address modification words. During the execution of instructions involving reading an operand from memory, the thirteen rightmost bits indicate the memory location of the operand. During the execution of other instructions, these thirteen bits may have various meanings as given in the instruction repertoire.

8. P Counter

The P Counter (Program Address Counter) is the location counter that controls the execution sequence of the instructions; that is, it holds the memory address of the next instruction to be executed. The thirteen bits of the next instruction address are indicated by thirteen display lights on the control panel. The P Counter is incremented by one before the execution of an instruction so that it normally indicates the address of the next instruction in sequence. Under certain conditions, an address in the I Register may be transferred to P.

AUTOMATIC ADDRESS MODIFICATION

Automatic modification of the address portion (bits 7 through 19) of instruction words prior to their execution can be accomplished under program control through the use of three special automatic address modification words. These special words, also referred to as X Registers, are magnetic core memory locations 00001, 00002 and 00003. The 5 and 6 bit positions of an instruction word are used to designate which of the three memory locations is to be used in modifying the address.



ADDRESS MODIFICATION BITS

Bit configuration 01 in bit positions 5 and 6 selects memory location 00001, 10 selects memory location 00002 and 11 selects memory location 00003. Bits 00 indicate that no modification is to be performed.

When an instruction word comes from memory to the I (instruction) Register, the two modification bits (5 and 6) are tested to determine whether or not address modification is required. If bits 5 and 6 are other than 00, an address modification is performed (see reference below to certain exceptional cases). The address portion of the selected modification word is sent through the B Register to the Arithmetic Unit. Bits 7 through 19 of the I Register (operand address bits) are also sent to the Arithmetic Unit where the address portion of the selected modification word is added to it. The changed address is then put in the I Register, and the instruction is executed.

Summing up, the result of an automatic address modification is: the operation code (bits 0 through 4) of the instruction word in the I Register is unchanged; and the operand address (bits 7 through 19) is altered by the amount in the modification word. If automatic address modification is called for, an extra word time (18 microseconds) is required to accomplish this.

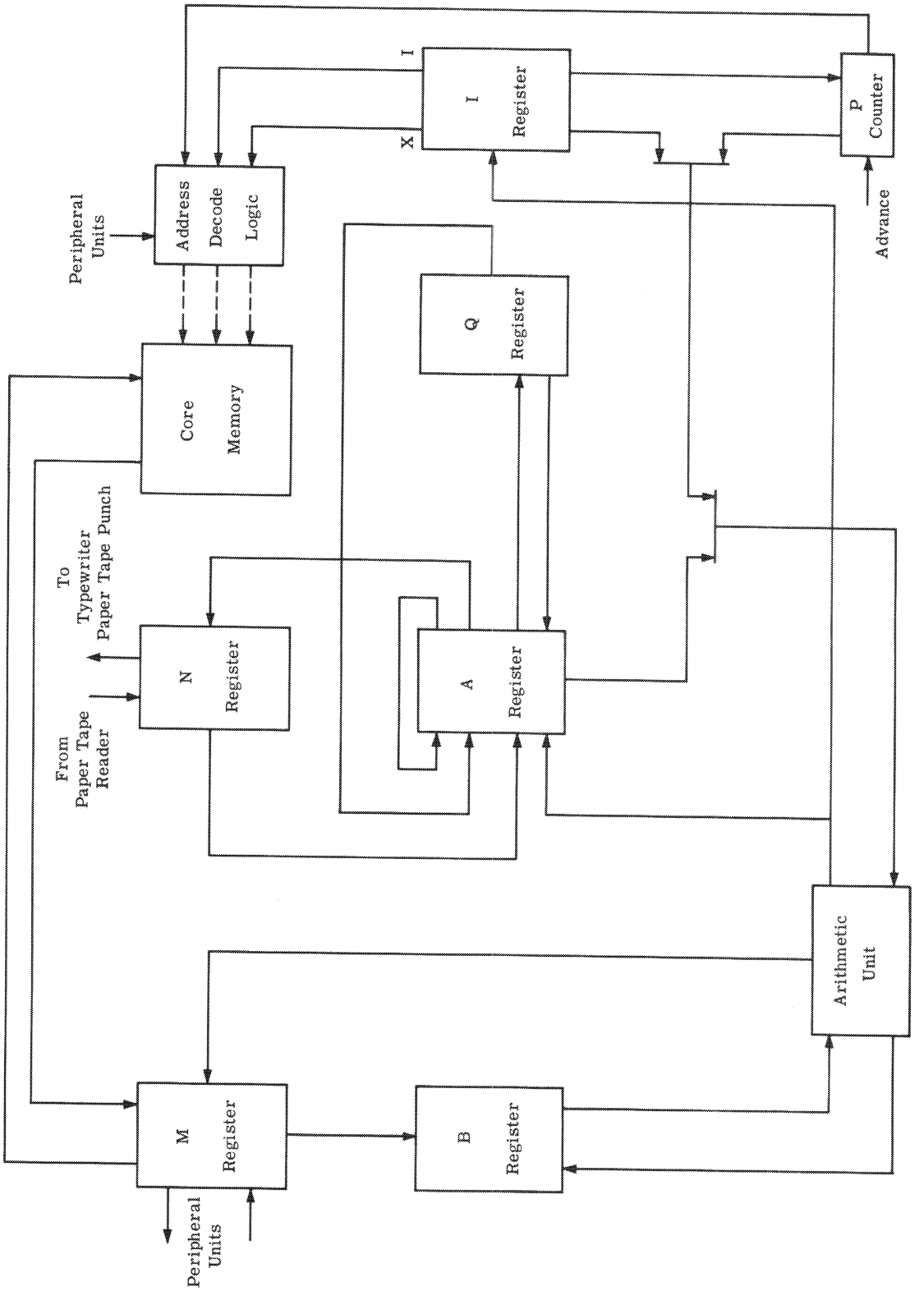


Figure 14 Register Relationships

In the case of certain instructions which affect the address modification words themselves, the 5 and 6 bits are used to designate these modification words as a part of the normal execution of the instruction. The bit positions are, therefore, not available to indicate address modification. The Instruction Repertoire section of this manual indicates which instructions can be automatically modified and which cannot. The programmer must be aware, however, that the modification of many instructions, though possible, will not have a proper meaning. In this category are those instructions which use a portion or all of the operand address field as an elaboration of the operation code.

CYCLE OF OPERATION

With the exception of branching operations, instructions are executed sequentially. The next instruction is read from memory after the execution of the current instruction. A sequence control counter, the P counter, contains the address of the next instruction to be executed. The contents of P are displayed on the control console.

In order for the Central Processor to fetch and execute program instructions in an orderly and sequential manner, it is necessary to provide a definite time base for these operations. The effective pulse frequency for all Central Processor operations is 450 KC. Therefore, the time between these basic pulses is approximately 2.25 microseconds. Eight of these pulses comprise another basic unit of time, known as a "word time", which is of 18 microseconds duration.

A word time is the time required to read one word out of memory, transfer this word to the appropriate register or registers, and re-write the word back into memory (due to destructive readout). Thus, one word time will be required to fetch an instruction out of memory, and one word time will normally be required to look up the operand associated with this instruction and perform all operations necessary to its proper execution. Some instructions will require more than one word time for their complete execution. Examples of these include double word length instructions and multiply and divide. Single word transfers from or to memory, including instruction access time, require a total of 36 microseconds; double word length transfers require a total of 54 microseconds. Execution times for each instruction in the command repertoire of the GE 225 is given in a later section of this manual.

That portion of the control logic necessary to ensure the orderly sequence of (1) fetching an instruction, (2) modifying the data address (if required), and (3) executing the instruction is known as the sequence control. The sequence control also ensures the repetition of these steps in a cyclic manner, thereby permitting program execution. In addition, the sequence control directs the sequence of additional steps required for the execution of multiple-word-time instructions by providing appropriate signals (Operation Enable or OE) for this purpose.

Figure 15 is a flow chart of the operations performed by the Central Processor while executing a program. More exactly, this diagram illustrates the nature of the operations and tests performed during one complete instruction cycle, including the extraction of the instruction from memory (AMP), modification of the data address portion of the instruction if required (AMX), and the subsequent execution thereof (AMI, GIS or AMX). Program execution is accomplished by properly repeating this basic cycle until the program has been completely executed. Program execution may also be stopped from the Operator's Console, in which event the cycle will be stopped immediately following the AMP operation.

The following example illustrates the cycle of computer operation. Assume that the first instruction of a program has somehow been manually entered into the I Register and that it is a LOAD REGISTER A instruction. Further assume that bits 5 and 6 of this instruction word are zeros, indicating that no modification of the data address in I is required. The remainder of the program has been loaded into memory beginning at location 0000 and the P Register has been manually cleared (i.e., set equal to 0000) by the operator. The Auto/Manual Console Switch is placed on Auto and the Start (Step) button is depressed. Since the execution of this particular instruction involves use of the memory (the instruction is not a "general" instruction), the next operation to be performed will be AMI. During the execution of the LOAD A instruction, the contents of the memory cell specified by the data address bits (7-19) of the I Register will be transferred to the A Register (via the M and B Registers and the Arithmetic Unit). Since the LOAD A instruction requires only one word time for its complete execution, an End of Operation signal (EOO) will be generated during AMI time, thereby instructing the sequence control to perform the AMP operation next. Assuming that memory access will be available to the Processor during this following word time, the instruction located in cell 0000 (in this example) will be pulled out of memory, transferred to the I Register, and the cycle repeated.

PRIORITY INTERRUPT

Due to the unique design concept of the 225 System, the core memory serves the dual function of: (1) main memory and (2) input-output buffer. Thus, two or more asynchronous operations may be performed simultaneously; for example, reading cards at a relatively slow rate while computing at the standard 450 KC repetition rate. Processor computation and access to the memory will have to be interrupted occasionally to allow information to be entered into or taken out of the memory at the request of the input or output devices currently in operation. Since several requests for memory access might be made at the same time, a provision is made to grant only one such request for access during a particular word time. The analysis of these various requests for memory access and the determination of whether an input-output device or the Central Processor should have access to the memory is performed by the Priority Interrupt logic.

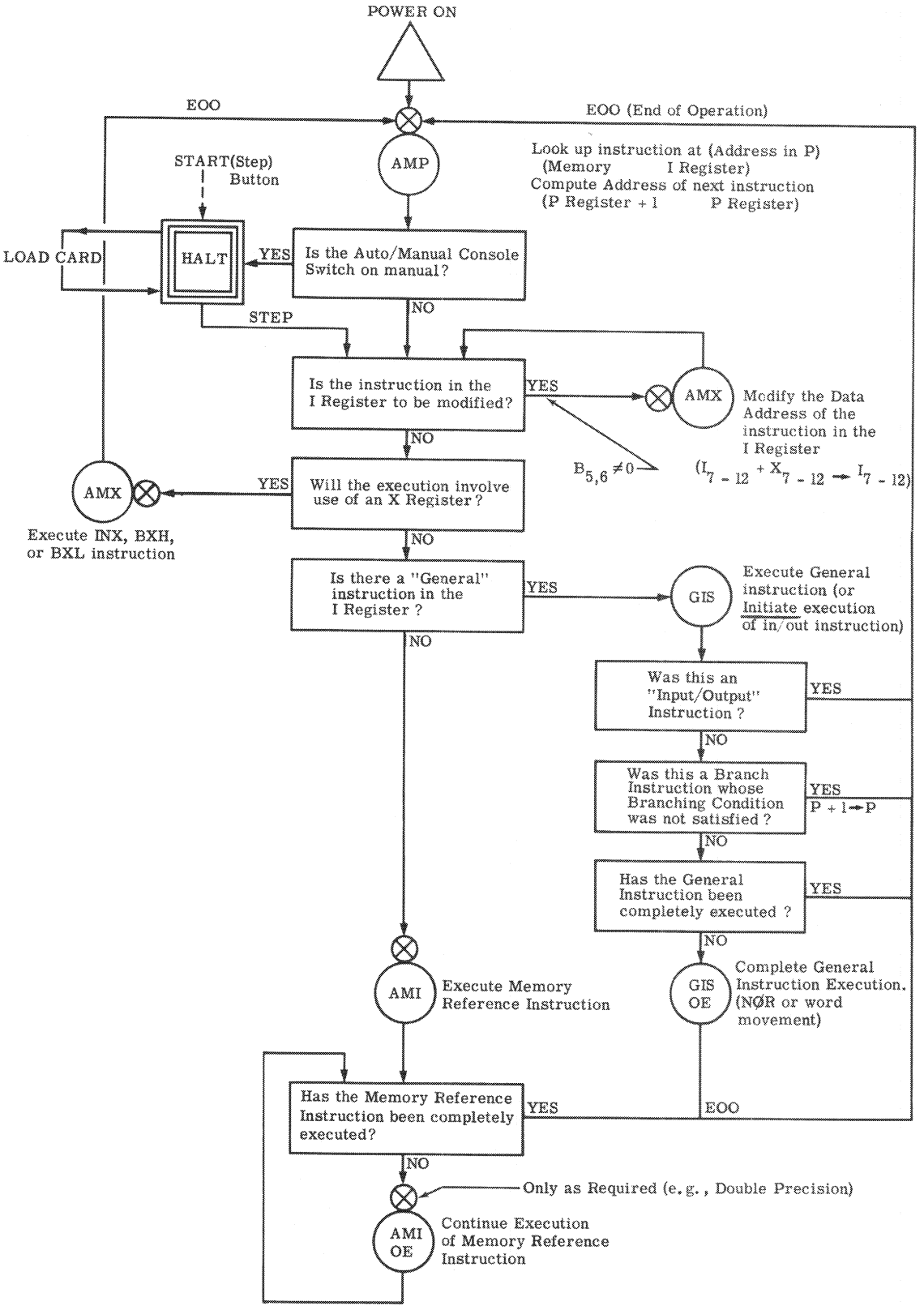
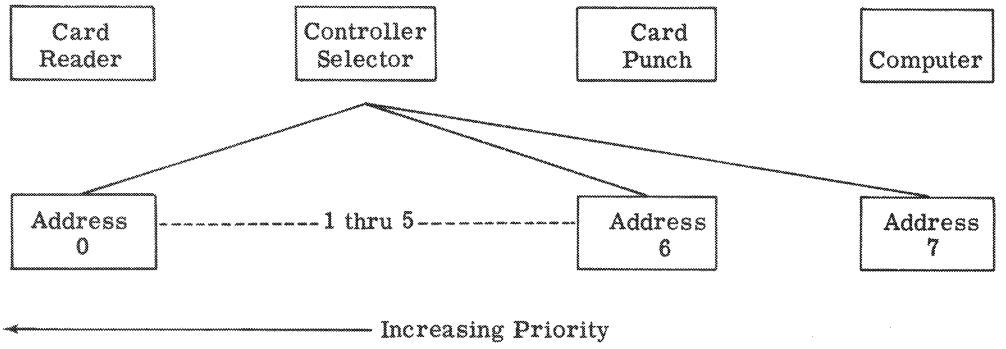


Figure 15 Central Processor Operating Cycle

The granting of a request for memory access is dependent upon the priority assigned (built into the computer) to a particular device. This priority, in turn, is determined by the repetition rate of pulses (information) going to or coming from the input-output device. Thus, if a request for access is received from two input-output devices simultaneously, the one with the higher repetition rate will have top priority and, hence, be the first to be granted access. The other device will wait for one word time. The reasoning behind this basis for priority assignment is that the slower speed unit can afford to wait without danger of loss of information. The faster unit cannot afford to wait since additional information is soon to follow. The Central Processor will always have the lowest priority since no loss of information can result if it is forced to "hang up" or remain in a state of suspended animation awaiting memory access during its normal cyclic operation. The Priority Interrupt logic will consider the Controller Selector (data mating function) as one other input-output device. Thus, two levels of priority are involved in the data mating operation. First, there is the computer priority (Priority Interrupt logic). This priority determines whether memory access is granted to the computer, to the card system, or to the Controller Selector. Second, there is a Controller Selector priority which determines which associated peripheral unit is granted memory access when the Controller Selector, itself, has such access.

Computer Priority Interrupt Control



D. INSTRUCTION REPERTOIRE

The GE 225 operates under the programmed control of over one hundred instructions. These instructions are classified into the following categories:

1. Arithmetic
2. Data Transfers
3. Shift Operations
4. Internal Test-and-Branch
5. Console Operation
6. Punched Card Input-Output
7. Paper Tape Input-Output
8. High Speed Printer Sub-System
9. Magnetic Tape Sub-System
10. Mass Random Access File Sub-System
11. Document Handler Sub-System

The following list of instructions gives the mnemonic code for the command and an indication of whether a memory location (operand address) or a constant is required.

Machine running programs are created by use of the General Assembly Program. Routines are prepared by specifying in tabular form the mnemonics of the instructions, the memory locations or constants involved, and whether or not the instructions are to be modified. For example, if it is desired to store the contents of the A Register in memory and modify the instruction, the following display will accomplish this:

Operation Code	Operand Address	Modification Word
STA	Y	X

The mnemonic STA (Store A) stores the contents of the A Register in the memory location (Y). The operand address (Y) may be a decimal number (for example, memory location 01050) or the operand address may be a symbolic designation (for example, NETPAY) which is intelligible to the GE 225 assembler routines. With the General Assembly Program the programmer may use symbols for operand addresses whenever he so desires, and memory locations will be automatically assigned. The letter "X" (X Register) indicates whether or not the instruction is to be automatically modified. A zero indicates that there is to be no modification, while a 1, 2 or 3 selects memory modification words 00001, 00002 or 00003.

In the following list of instructions, all instructions can be automatically modified unless stated otherwise. In all instructions involving the bringing of a word from memory, the word in memory remains unchanged; in all instructions involving the transfer of information from registers, the condition of the register after execution is unchanged unless otherwise stated. A "Y" indicates an operand address; a "K" indicates the operand itself. Some instructions by their nature do not require an operand. The word times required for execution include the fetching of the instruction.

ARITHMETIC

The capacity of the A Register may be exceeded in execution of Add, Subtract or Multiply commands resulting in a condition known as "overflow". In this event, the overflow indicator is turned on, the high-order (most significant) bit of the result is lost, and the sign of the result is reversed.

ADD Y Octal: 01 Word Time: 2

ADD. The contents of Y (s,1-19) are algebraically added to the contents of Register A (s,1-19). The result is placed in Register A (s,1-19). The contents of Y are not changed.

SUB Y Octal: 02 Word Time: 3

SUBTRACT. The contents of Y (s,1-19) are algebraically subtracted from the contents of Register A (s,1-19). The result is placed in Register A (s,1-19). The contents of Y are not changed.

DAD Y Octal: 11 Word Time: 3

DOUBLE LENGTH ADD. If Y is even, the contents of Y (s,1-19) and Y + 1 (1-19) are algebraically added to the contents of Register A (s,1-19) and Register Q (1-19). If Y is odd, the contents of Y (s,1-19) and Y (1-19) are algebraically added to the contents of Register A (s,1-19) and Register Q (1-19). The result is placed in Register A (s,1-19) and Register Q (1-19). The sign of Register Q is set to agree with the sign of Register A. The contents of Y and Y + 1 are unchanged. If this instruction is automatically modified, the address after modification will determine the result as indicated above.

DSU Y Octal: 12 Word Time: 5

DOUBLE LENGTH SUBTRACT. If Y is even, the contents of Y (s,1-19) and Y + 1 (1-19) are algebraically subtracted from the contents of Register A (s,1-19) and Register Q (1-19). If Y is odd, the contents of Y (s,1-19) and Y (1-19) are algebraically subtracted from the contents of Register A (s,1-19) and Register Q (1-19). The result is placed in Register A (s,1-19) and Register Q (1-19). The sign of Register Q is set to agree with the sign of Register A. The contents of Y and Y + 1 are unchanged. If this instruction is automatically modified, the address after modification will determine the result as indicated above.

MPY Y Octal: 15 Maximum Of: 21

MULTIPLY. The contents of Y (s,1-19) are algebraically multiplied by the contents of Register Q (s,1-19). The result is placed in Register A (s,1-19) and Register Q (1-19); the sign of Register Q is the same as the sign of Register A after multiplication. If the contents of Register A are not set to zero before the MPY command is given, the contents of Register A will be added algebraically to the least significant half of the product. Thus, with proper scaling, it is possible to form the value AB + C.

DVD Y Octal: 16 Maximum Of: 30

DIVIDE. The contents of Register A (s,1-19) and Register Q (1-19) are algebraically divided by the contents of Y (s,1-19). The quotient is placed in Register A (s,1-19); the remainder is placed in Register Q (s,1-19). The sign of the remainder is the sign of the dividend. The magnitude of the divisor must be greater than the magnitude of the contents of Register A. If not, the overflow indicator will be turned ON and control will be immediately transferred to the next instruction in sequence.

INX X, K Octal: 14 Word Time: 3

INCREMENT X BY K. K, positions 7 through 19 of the I Register, is added absolutely to the contents of Register X (7-19), and the result replaces the contents of Register X (7-19). Any carry from position 7 of Register X is lost. This instruction is not automatically modified since bits 5 and 6 are used to identify the particular X Register.

ADO Octal: 2504032 Word Time: 3

ADD ONE. Plus one is added algebraically to Register A (19). If the capacity of Register A is exceeded, the overflow indicator will be turned ON.

SBO Octal: 2504112 Word Time: 3

SUBTRACT ONE. One is subtracted algebraically from Register A (19). If the capacity of Register A is exceeded, the overflow indicator will be turned ON.

LDA Y Octal: 00 Word Time: 2

LOAD A. The contents of Y (s,1-19) replace the contents of Register A (s,1-19). The contents of Y are not changed.

STA Y Octal: 03 Word Time: 2

STORE A. The contents of Register A (s,1-19) replace the contents of Y (s,1-19). The contents of Register A are not changed.

DLD Y Octal: 10 Word Time: 3

DOUBLE LENGTH LOAD. If Y is even the contents of Y (s,1-19) and Y + 1 (s,1-19) replace the contents of Register A (s,1-19) and Register Q (s,1-19). If Y is odd, the contents of Y (s,1-19) replaces the contents of Register A (s,1-19) and Register Q (s,1-19). The contents of Y and Y + 1 are unchanged. If this instruction is automatically modified, the address after modification will determine the result as indicated above.

DST Y Octal: 13 Word Time: 3

DOUBLE LENGTH STORE. If Y is even, the contents of Register A (s,1-19) and Register Q (s,1-19) replace the contents of Y (s,1-19) and Y + 1 (s,1-19). If Y is odd, the contents of Register Q (s,1-19) replace the contents of Y (s,1-19). The contents of Register A and Register Q are unchanged. If this instruction is automatically modified, the address after modification will determine the result as indicated above.

LQA Octal: 2504004 Word Time: 3

LOAD Q FROM A. The contents of Register A (s,1-19) replace the contents of Register Q (s,1-19). The contents of Register A are unchanged.

LAQ Octal: 2504001 Word Time: 3

LOAD A FROM Q. The contents of Register Q (s,1-19) replace the contents of Register A (s,1-19). The contents of Register Q are unchanged.

XAQ Octal: 2504005 Word Time: 3

EXCHANGE A AND Q. The contents of Register A (s,1-19) and Register Q (s,1-19) are interchanged.

MAQ Octal: 2504006 Word Time: 3

MOVE A TO Q. The contents of Register A (s,1-19) replace the contents of Register Q (s,1-19). Zeros replace the contents of Register A (s,1-19).

STO Y Octal: 27 Word Time: 3

STORE OPERAND ADDRESS. The contents of Register A (7-19) replace the contents of Y (7-19). The contents of Register A and Y (s,1-6) are unchanged.

ORY Y Octal: 23 Word Time: 3

OR A INTO Y. Each bit of Register A is examined. If there is a 1 bit in Register A in a given position, a 1 bit is placed in Y in that position. The contents of Register A and the other bit positions of Y are unchanged.

EXT Y Octal: 20 Word Time: 3

EXTRACT. Each bit of Y is examined. If there is a 1 bit in Y in a given position, a zero is placed in the corresponding position of Register A. If there is a zero in a given position of Y, the corresponding position in Register A is left unchanged. The contents of Y are unchanged.

LDZ Octal: 2504002 Word Time: 3

LOAD ZERO INTO A. The contents of Register A (s,1-19) are replaced by 0's.

LDO Octal: 2504022 Word Time: 3

LOAD ONE INTO A. The contents of Register A (s,1-19) are set to 0, and a 1 is placed in Register A (19).

LMO Octal: 2504102 Word Time: 3

LOAD MINUS ONE INTO A. The contents of Register A (s,1-19) are replaced by 1's.

CPL Octal: 2504502 Word Time: 3

COMPLEMENT A. Each bit in Register A (s,1-19) is inverted; that is, each 1 is replaced by a 0 and each 0 is replaced by 1.

NEG Octal: 2504522 Word Time: 3

NEGATE A. The 2's complement (negative value) of the contents of Register A (s,1-19) replaces the contents of Register A (s,1-19). If the capacity of Register A is exceeded, the overflow indicator will be turned ON.

CHS Octal: 2504040 Word Time: 2

CHANGE SIGN OF A. The sign of Register A is changed. Positions 1-19 of Register A are unchanged.

NOP Octal: 2504012 Word Time: 3

NO OPERATION. Zero is added to the contents of Register A (s,1-19).

SHIFT OPERATIONS

The Shift commands shift the contents of the A Register to the right or left serially (bit by bit) either alone or with the contents of the N and/or Q Registers. A maximum of 31 places can be shifted. All shift commands vary between two and nine word times, depending upon the length of the shift. Two word times are required for a shift of four bit positions or less. One additional word time is required for each additional four bit shift or fraction thereof.

SRA **K** Octal: 25100 Word Time: 2+

SHIFT RIGHT A. The contents of Register A (1-19) are shifted right K places. If Register A is plus, 0's are inserted in the vacated positions of Register A. If Register A is minus, 1's are inserted in the vacated positions of Register A. Bits shifted out of position 19 are lost. The sign of Register A is not changed.

SLA **K** Octal: 25120 Word Time: 2+

SHIFT LEFT A. The contents of Register A (1-19) are shifted left K places. Vacated positions of Register A are filled with zeros. If a non-zero bit is shifted out of position 1, the overflow indicator will be turned ON, and the bit is lost. The sign of Register A is unchanged.

SCA **K** Octal: 251004 Word Time: 2+

SHIFT CIRCULAR A. The contents of Register A (1-19) are shifted right K places in a circular fashion; that is, the bit shifted out of position 19 is inserted in position 1, replacing the bit shifted out of position 1. The sign of Register A is not affected.

SRD **K** Octal: 25110 Word Time: 2+

SHIFT RIGHT DOUBLE. The contents of Register A (1-19) and the contents of Register Q (1-19) together are shifted K places to the right. Bits shifted out of Register A (19) shift into Register Q (1). Bits shifted out of Register Q (19) are lost. If the sign of Register A is plus, 0's fill the vacated positions; if the sign of Register A is minus, 1's fill the vacated positions. The sign of Register Q is replaced by the sign of Register A. The sign of Register A is unchanged.

SLD **K** Octal: 25122 Word Time: 2+

SHIFT LEFT DOUBLE. The contents of Register A (1-19) and the contents of Register Q (1-19) together are shifted K places to the left. Bits shifted out of Register Q (1) shift into Register A (19). The vacated positions of Register Q are filled with zeros. If a non-zero bit is shifted out of Register A (1), the overflow indicator is turned ON, and the bit is lost. The sign of Register Q replaces the sign of Register A. The sign of Register Q is unchanged.

SCD **K** Octal: 25112 Word Time: 2+

SHIFT CIRCULAR DOUBLE. The contents of Register A (1-19) and Register Q (1-19) together are shifted K places to the right in a circular fashion. Bits shifted out of Register A (19) shift into Register Q (1) and those from Register Q (19) shift into Register A (1). The sign of Register A replaces the sign of Register Q. The sign of Register A is unchanged.

SAN **K** Octal: 25104 Word Time: 2+

SHIFT A AND N RIGHT. The contents of Register A (1-19) and Register N (1-6) together are shifted K places to the right. Bits shifted out of Register A (19) shift into Register N (1). Bits shifted out of Register N (6) are lost. If the sign of Register A is plus, 0's fill the vacated positions of Register A; if the sign of Register A is minus, 1's fill the vacated positions of Register A. The sign of Register A is unchanged.

SNA K Octal: 25101 Word Time: 2+

SHIFT N AND A RIGHT. The contents of Register N (1-6) and Register A (1-19) together are shifted K places to the right. Bits shifted out of Register N (6) shift into Register A (1). Vacated positions in Register N are filled with 0's. Bits shifted out of Register A (19) are lost. The sign of A is unchanged.

NAQ K Octal: 25111 Word Time: 2+

SHIFT N, A AND Q RIGHT. The contents of Register N (1-6), Register A (1-19) and Register Q (1-19) together are shifted K places to the right. Bits shifted out of Register N (6) shift into Register A (1). Bits shifted out of Register A (19) shift into Register Q (1). Bits shifted out of Register Q (19) are lost. Vacated positions of Register N are filled with 0's. The sign of Register A is unchanged.

ANQ K Octal: 25114 Word Time: 2+

SHIFT A INTO N AND Q. The contents of Register A (1-19) are shifted K places to the right into both Register N and Register Q. Bits shifted out of Register A (19) enter both Register Q (1) and Register N (1). Bits shifted out of Register N (6) and Register Q (19) are lost. If the sign of Register A is plus, the vacated positions of Register A are filled with 0's; if the sign of Register A is minus, 1's fill the vacated positions of Register A. The sign of Register A replaces the sign of Q. The sign of Register A is unchanged.

NOR K Octal: 25130 Word Time: 2+

NORMALIZE A REGISTER. If R, the number of leading zeros of Register A (1-19), is less than K, the contents of Register A (1-19) are shifted left R places, and K-R replaces the contents of location 0000 (15-19). If R is greater than, or equal to, K, the contents of Register A (1-19) are shifted left K places, and a zero replaces the contents of location 0000 (15-19). Positions S, 1-14 of location 0000 are always set to zero. Vacated positions of Register A are filled with zeros. The sign of Register A is unchanged.

DNO K Octal: 25132 Word Time: 2+

DOUBLE LENGTH NORMALIZE. If R (the number of leading zeros of Register A) is less than K, the contents of Register A (1-19) and Register Q (1-19) are shifted left R places, and K-R replaces the contents of location 0000 (15-19). If R is greater than, or equal to, K, the contents of Register A (1-19) and Register Q (1-19) are shifted left K places and a zero replaces the contents of location 0000 (15-19). Positions S, 1-14 of location 0000 are always set to zero. Bits shifted out of Register Q (1) shift into Register A (19). Vacated positions of Register Q are filled with zeros. The sign of Register Q replaces the sign of Register A. The sign of Register Q is unchanged.

INTERNAL TEST-AND-BRANCH

BRU Y Octal: 26 Word Time: 1

BRANCH UNCONDITIONALLY. Control is transferred to the instruction located at Y; i.e., Y becomes the address of the next instruction. (The contents of Register I (7-19) are transferred to Register P (7-19.)

SPB **X, Y** Octal: 07 Word Time: 2

STORE P AND BRANCH. The location of this instruction, Register P (7-19), replaces the contents of Register X (7-19), and control is transferred to the instruction located at Y, i.e., Y becomes the address of the next instruction. This instruction is not automatically modified since bits 5 and 6 are used to identify the particular X Register.

BPL Octal: 2516001 Word Time: 2

BRANCH ON PLUS. If the sign of Register A is plus, the computer takes the next sequential instruction. If the sign of Register A is not plus, the computer skips the next instruction and executes the second sequential instruction. The contents of Register A are unchanged by this instruction. (Note that branching will occur if Register A is all zeros.)

BMI Octal: 2514001 Word Time: 2

BRANCH ON MINUS. If the sign of Register A is minus, the computer takes the next sequential instruction. If the sign of Register A is not minus, the computer skips the next instruction and executes the second sequential instruction. The contents of Register A are unchanged by this instruction.

BZE Octal: 2514002 Word Time: 2

BRANCH ON ZERO. If the contents of Register A (s,1-19) are zero, the computer takes the next sequential instruction. If the contents are not zero, the computer skips the next instruction and executes the second sequential instruction. The contents of Register A are unchanged by this instruction.

BNZ Octal: 2516002 Word Time: 2

BRANCH ON NO ZERO. If the contents of Register A (s,1-19) are not zero, the computer takes the next sequential instruction. If the contents are zero, the computer skips the next instruction and executes the second sequential instruction. The contents of Register A are unchanged by this instruction.

BOD Octal: 2514000 Word Time: 2

BRANCH ON ODD. If the contents of Register A are odd (A (19) contains a 1), the computer takes the next sequential instruction. If the contents of Register A are even (A (19) contains a 0), the computer skips the next instruction and executes the second sequential instruction. The contents of Register A are unchanged by this instruction.

BEV Octal: 2516000 Word Time: 2

BRANCH ON EVEN. If the contents of Register A are even [A (19) contains a 0], the computer takes the next sequential instruction. If the contents of Register A are odd [A (19) contains a 1], the computer skips the next instruction and executes the second sequential instruction. The contents of Register A are unchanged by this instruction.

BOV Octal: 2514003 Word Time: 2

BRANCH ON OVERFLOW. If the overflow indicator is ON, the indicator is turned OFF and the computer takes the next sequential instruction. If the overflow indicator is not ON, the computer skips the next instruction and executes the second sequential instruction.

BNO Octal: 2516003 Word Time: 2

BRANCH ON NO OVERFLOW. If the overflow indicator is not ON, the computer takes the next sequential instruction. If the overflow indicator is ON, the indicator is turned OFF, the computer skips the next instruction and executes the second sequential instruction.

BPE Octal: 2514004 Word Time: 2

BRANCH ON PARITY ERROR. If the parity error indicator is ON, the indicator is turned OFF, and the computer takes the next sequential instruction. If the parity error indicator is not ON, the computer skips the next instruction and executes the second sequential instruction.

BPC Octal: 2516004 Word Time: 2

BRANCH ON PARITY CORRECT. If the parity error indicator is OFF, the computer takes the next sequential instruction. If the parity error indicator is ON, the indicator is turned OFF, the computer skips the next instruction and executes the second sequential instruction.

BXH X, K Octal: 05 Word Time: 3

BRANCH IF X IS HIGH OR EQUAL. If the contents of Register X (7-19) are greater than or equal to K, the computer takes the next sequential instruction; if the contents of Register X (7-19) are less than K, the computer skips the next instruction and executes the second sequential instruction. The contents of Register X are not changed. This instruction is not automatically modified since bits 5 and 6 are used to identify the particular X Register. (Note: K is required to be the 2's complement of the desired test value. This requirement is automatically taken care of by the General Assembly Program.)

BXL X, K Octal: 04 Word Time: 3

BRANCH IF X IS LOW. If the contents of Register X (7-19) are less than K, the computer takes the next sequential instruction; if the contents of Register X (7-19) are greater than or equal to K, the computer skips the next instruction and executes the second sequential instruction. The contents of Register X are not changed. This instruction is not automatically modified since bits 5 and 6 are used to identify the particular X Register. (Note: K is required to be the 2's complement of the desired test value. This requirement is automatically taken care of by the General Assembly Program.)

CONSOLE OPERATION

RCS Octal: 2500011 Word Time: 2

READ CONTROL SWITCHES. Each of the 20 manually set Register A control switches is examined. If a switch is DOWN (ON), a 1 bit is placed in the corresponding position of Register A; otherwise the corresponding position in Register A will not be altered. The A Register should be cleared before this instruction is given.

TON Octal: 2500007 Word Time: 2

TYPEWRITER ON. The power for the typewriter is turned ON. To allow the motor to attain operating speed, a delay of at least 200 milliseconds must be programmed before giving a command to type. However, if the command TON is given within 1 millisecond after turning off the typewriter, no delay is required. (A manual switch on the typewriter must also be turned on.)

BNR Octal: 2514005 Word Time: 2

BRANCH ON N REGISTER READY. If the N Register is available for input-output, (if the last TYPE, READ PAPER TAPE, or WRITE PAPER TAPE instruction has been executed) the computer takes the next sequential instruction; if not, the computer skips the next instruction and executes the second sequential instruction.

BNN Octal: 2516005 Word Time: 2

BRANCH ON N REGISTER NOT READY. If the N Register is not available for input-output (if the last TYPE, READ PAPER TAPE, or WRITE PAPER TAPE instruction has not been executed) the computer takes the next sequential instruction. If it is, the computer skips the next instruction and executes the second sequential instruction.

TYP Octal: 2500006 Word Time: 2

TYPE. The six-bit, coded character in Register N is typed. The contents of Register N are not changed. (No keys will be activated when an attempt is made to type an illegal bit configuration, and the N Register will be placed in a BUSY condition. Clearing of this condition is accomplished by manually typing a character.)

OFF Octal: 2500005 Word Time: 2

TYPEWRITER OFF. Power supply for the typewriter is turned off.

PAPER TAPE INPUT-OUTPUT

RPT Octal: 2500010 Word Time: 2

READ PAPER TAPE. The N Register is cleared, and one six-bit coded character is read into Register N. Other instructions not using Register N may be executed during this time.

WPT Octal: 2500012 Word Time: 2

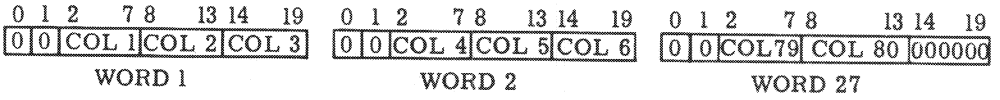
WRITE PAPER TAPE. The six-bit coded character in Register N is punched. The contents of Register N are not changed. Other instructions not using Register N may be executed during this time.

PUNCHED CARD INPUT-OUTPUT

CARD READER

The starting memory address into which information is placed must be a multiple of 128 but less than 2048. Once a card read instruction has been initiated, it will continue reading cards continuously until terminated by execution of a HALT CARD READER instruction, a hopper "empty" condition, or a misfeed. (Card reading will also be stopped if: 1) the Auto-Manual Console Switch is on Manual, 2) a Card Punch Alarm has occurred or 3) if a parity error has occurred and the STOP ON PARITY ALARM console switch is in the STOP position.)

In the decimal (alphanumeric) mode of operation, each card column (one character) is converted into an equivalent 6-bit binary coded decimal form. Each group of three card columns (characters) represents 18 bits which are placed in the 18 least significant (rightmost) bits of a memory word. The 2 most significant (leftmost) bits of the 20-bit word are set equal to zero. With three digits to a memory location, 27 memory locations are required to accommodate an 80-column punched card. The 27th memory location contains only two characters.



Four cards of BCD information may be contained in memory at one time. The fifth card is read into the same memory area as the first card, and so forth.

First card - Read into memory at (starting address) to (starting address + 26)

Second card - Read into memory at (starting address + 32) to (starting address + 58)

Third card - Read into memory at (starting address + 64) to (starting address + 90)

Fourth card - Read into memory at (starting address + 96) to (starting address + 122)

Fifth card - Read into memory at (starting address) to (starting address + 26)

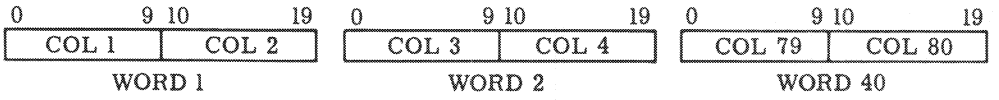
etc.

The word following the last word filled from the card (i.e., starting address + 27, starting address + 59, starting address + 91, and starting address + 123) will automatically receive the following information:

10 110000 110000 111111 when the reading of the card is complete.

11 110000 110000 111111 when the card is the last card of the input deck.

In the binary mode of operation, each row (0 thru 9) in each card column represents a bit position; a blank is a 0 bit, a punch is a 1 bit. Rows 11 and 12 on the card are ignored in the binary mode. Thus, each set of two 10-bit card columns is converted into a 20-bit memory word. The first column is placed in the 10 most significant bits of the memory word, the second column in the 10 least significant bits of the memory word, and so forth. The 80 columns of punched information are stored in forty successive memory locations.



Two cards of binary information may be contained in memory at one time. The third card is read into the same memory area as the first card, and so forth.

First card - Read into memory at (starting address) to (starting address + 39)

Second card - Read into memory at
(starting address + 64) to (starting address + 103)

Third card - Read into memory at
(starting address) to (starting address + 39)

etc.

The second word following the last word filled from the card (i.e., starting address + 41 and starting address + 105) will automatically receive the following information:

1000000000 1111111111 when the reading of the card is complete.

1100000000 1111111111 when the card is the last card of the input deck.

BCR Octal: 2514006 Word Time: 2

BRANCH ON CARD READER READY. If the card reader is ready to read cards and the card hopper is not empty, the computer takes the next sequential instruction; if not, the computer skips the next instruction and executes the second sequential instruction.

BCN Octal: 2516006 Word Time: 2

BRANCH ON CARD READER NOT READY. If the card reader is not ready to read cards or if the card hopper is empty, the computer takes the next sequential instruction; if not, the computer skips the next instruction and executes the second sequential instruction.

RCD Y Octal: 250YY00 Word Time: 2

READ CARDS DECIMAL. This instruction initiates continuous reading of decimal cards (i.e., information is interpreted by the Processor's Card Reader logic as being in the decimal format) into memory starting at location Y, where Y is a multiple of 128 and less than 2048. The first card will be read into locations Y through Y + 26, the second into Y + 32 through Y + 58, the third into Y + 64 through Y + 90, the fourth into Y + 96 through Y + 122, the fifth into Y + 26, etc. After each card is read in, the sign bit of the word after the last word of the card (Y + 27, Y + 59, Y + 91, or Y + 123) will be set to minus. After the last card of the deck is read in, bit position 1 of the word after the last word of the card (Y + 27, Y + 59, Y + 91, or Y + 123) will be set to a 1. If the card reader is not in ready status when the READ instruction is given, the computer will halt.

RCB Y Octal: 250YY01 Word Time: 2

READ CARDS BINARY. This instruction initiates continuous reading of binary cards (i.e., information is interpreted by the Processor's Card Reader logic as being in the binary format) into memory starting at location Y, where Y is a multiple of 128 and less than 2048. The first card will be read into locations Y through Y + 39, the second into Y + 64 through Y + 103, the third into Y through Y + 39, etc. After each card is read in, the sign bit of the second word following the card image (Y + 41 or Y + 105) will be set minus. After the last card of a deck is read in, position 1 of the second word following this card image (Y + 41 or Y + 105) will be set to 1. If the card reader is not in ready status when the READ instruction is given, the computer will halt.

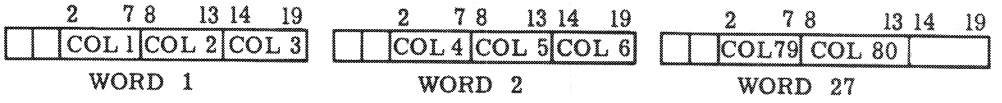
HCR Octal: 2500004 Word Time: 2

HALT CARD READER. This instruction halts the card feed. If the first half of a card is being read at the time this instruction is given, the reading of this card into memory will be completed, after which no further cards will be read until another READ instruction is given. This instruction does not delay the computer until input is complete. The program continues in sequence, therefore a delay must be programmed to insure that the information is in memory before attempting to utilize it.

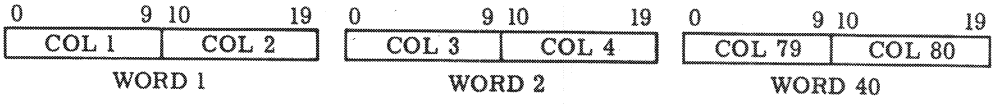
CARD PUNCH

The starting memory address from which information is punched must be a multiple of 128 but less than 2048. A significant difference between card reading and card punching operations is that a write (punch) card instruction causes only a single card to be punched. If a write (punch) card instruction is given during the next 40 milliseconds after completion of the punching of a card, the punching will proceed at the rate of 100 cards per minute. If the next punch command is not given until after the 40 millisecond period, the maximum punching rate is 50 cards per minute.

In the decimal (alphanumeric) mode of operation, bit positions 2 thru 7 of the starting address memory word are converted into the equivalent punched card (Hollerith) character and punched into card column 1; bit positions 8 thru 13 are converted and punched into column 2; and bit positions 14 thru 19 are converted and punched into column 3. Bits in positions 0 and 1 are disregarded. Since there are 80 columns in the card, only 2 characters will be punched from the 27th memory location; these latter 2 occupying bit positions 2 thru 7 and 8 thru 13 respectively.



In the binary mode of operation, each row (0 thru 9) in each card column may hold a bit of information from memory; a blank is a 0 bit, a punch is a 1 bit. Rows 11 and 12 on the card are not punched in the binary mode. Thus, a 20-bit memory word is punched as a set of two 10-bit card columns. Information in bit positions 0 thru 9 of the starting address memory word is punched bit for bit in the first card column, information in bit positions 10 thru 19 is punched in the second card column, and so forth until 40 memory words have been punched.



BPR Octal: 2514007 Word Time: 2

BRANCH ON CARD PUNCH READY. If the card punch is in a ready status, the computer takes the next sequential instruction; if not, the computer skips the next instruction and executes the second sequential instruction.

BPN Octal: 2516007 Word Time: 2

BRANCH ON CARD PUNCH NOT READY. If the card punch is not in a ready status, the computer takes the next sequential instruction; if it is, the computer skips the next instruction and executes the second sequential instruction.

WCD **Y** Octal: 2504402 Word Time: 2

WRITE CARD DECIMAL. The information in memory locations Y through Y + 26 (where Y is a multiple of 128 and less than 2048) is punched into a card in decimal (alphanumeric) format. If the card punch is not in ready status when this instruction is given, the computer will halt.

WCB **Y** Octal: 2504403 Word Time: 2

WRITE CARD BINARY. The information in memory locations Y through Y + 39 (where Y is a multiple of 128 and less than 2048) is punched into a card in binary format. If the card punch is not in ready status when this instruction is given, the computer will halt.

HIGH SPEED PRINTER SUB-SYSTEM

Operation Code	Operand Address	Modification Word
BDM	**C+T/F**	**P**

BRANCH ON DATA MATING FUNCTION INTERROGATED CONDITIONS. P is the address of the high speed printer controller to be interrogated. C is the number of the specific condition to be tested. Both C and P have the range 0 to 7. If C+T is specified and the condition tested (C) is true, the computer takes the next sequential instruction; if it is not true, the computer skips the next instruction and executes the second sequential instruction. If C+F is specified and the condition tested (C) is not true (false), the computer takes the next sequential instruction; if it is true, the computer skips the next instruction and executes the second sequential instruction.

Condition

0 Controller busy
1
2 Out of Paper
3
4
5
6
7 Any Error

Operation Code	Operand Address	Modification Word
SEL		**P**
WPL	**M_2**	**F**
WPL	**M_1**	**N**

WRITE PRINT LINE. P is the address (0 thru 7) of the controller to which the high speed printer is attached. SEL is the mnemonic code for the selection function. WPL is the mnemonic code for Write Print Line. M_1 is the memory address of the first data word in the block of 40 (maximum) data words to be printed. Data words to be printed consist of three BCD characters each. If less than 40 words are to be printed on one line, the sign bit must be a 1 in the last word to be printed. F is the format control indicator. If a blank (space) is written in the F position, the line is to be printed without automatic format control, and M_2 is ignored. If an F is written in the F position, automatic format control words starting at memory address M_2 are used to control the printing of the data words. N is the numeric print indicator. A blank (space) is written in the N position if the data words to be printed are alphanumeric. An N is written in the N position if the data words to be printed consist only of decimal numbers and the 14 special symbols. Both M_2 and M_1 must be in the same half of a 16,000 word memory. The General Assembly Program normally arranges to space the paper one line after printing. Spacing of 0 to 63 lines, or ejecting the paper to the top of the next page may be coded as part of the WPL command by coding lines 2 and 3 in Octal as shown below:

1 = PRINT and slew
0 = Slew only

	S	1	2	3	4	5	6 - 19
2nd line:	1	1=Format 0=No Format	V1	V2	V3	1=Numeric 0=Alpha-numeric	Format Address
3rd line:	V4	V5	V6	C1	C2	Data Address	

If C1=0 and C2=1, ignore V1 thru V6, and slew paper to top of next page.
If C1=1 and C2=1, slew paper the number of lines (0 thru 63) indicated by the binary number in positions V1 thru V6.

AUTOMATIC FORMAT CONTROL

If a line is to be printed under a format control, the format data is stored in the Central Processor memory in a block of words under the same organization as the print line data. The format control data consists of:

a. Any printable character
b. Special control characters

The Printer Controller, in assembling a formatted line, reads in from the Central Processor memory one word of data and one word of format. The first format character is considered initially. If it is a printable character, the character is printed. If it is a special control character, it is treated as described below. Assuming it was a printable character, it is printed, and the first data character is considered. If it is a printable character it is printed. It may be a special control character, in which case it is treated as explained below. In sequence, the second format, then second data characters are considered, followed by the third format and the third data characters. Following the consideration of the third data character another word of data and another word of format are requested from the Central Processor memory. Upon receiving these new words, the procedure described above is again followed. This routine is continued until a one in the sign bit of a data word is encountered; whereupon, after consideration of that data word and its respective format word, the sequence is ended.

There are five special control characters, mentioned above, which are available for controlling the format of the printed line. These characters, their BCD bit representations, and their functions are:

1. Ignore (Octal: 35)

   If a format character is an Ignore, the next data character is immediately considered.

2. Ignore/Skip (Octal: 36)

   If a format character is an Ignore/Skip, a blank is printed and the next data character is considered.

3. Delete (Octal: 37)

   If a format character is a Delete, the next data character is ignored, and the next character considered is the next format character.

4. Delete/Skip (Octal: 56)

   If a format character is a Delete/Skip, a blank is printed, the next data character is ignored, and the next character considered is the next format character.

5. Zero Suppress (Octal: 57)

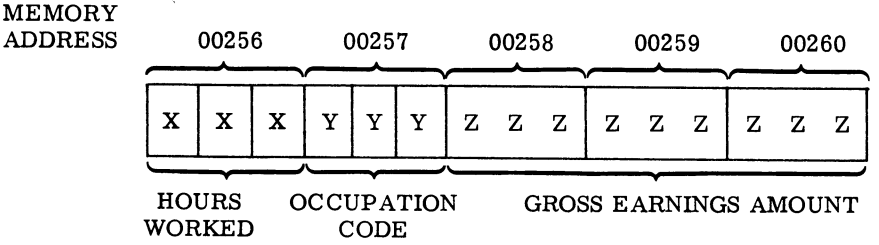
   If a format character is a Zero Suppress, the next data character is ignored; and the next format character is printed if it is a printable character. After considering this last format character, blanks will be inserted in the print line until: (a) a non-zero data character is detected, or (b) a period comes up in the format data. It should be noted that once a Zero Suppress has been put into effect, the print line data is inspected only for a non-zero data character, and the format control data is inspected only for a period. A $ symbol in the format data also initiates the insertion of blanks in the print line in the same manner as Zero Suppress after it has been printed.

It is possible for an Ignore or an Ignore/Skip character to be placed in the print line data (as well as in the format control data). If a data character is an Ignore, the next format character is immediately considered and nothing is printed for that data character. If a data character is an Ignore/Skip, a blank is printed and the next format character is considered.

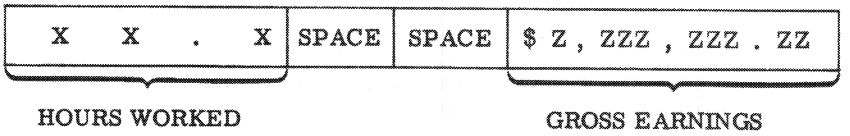
The above procedure makes it possible for a line format to be stored in the Central Processor memory once and to be used as often as needed to print lines of data in that format. The data may, within the limitations imposed by the use of the special control as described above, be stored in sequence in computer memory, the Printer Controller automatically constructing the print line according to the prescribed format.

EXAMPLE

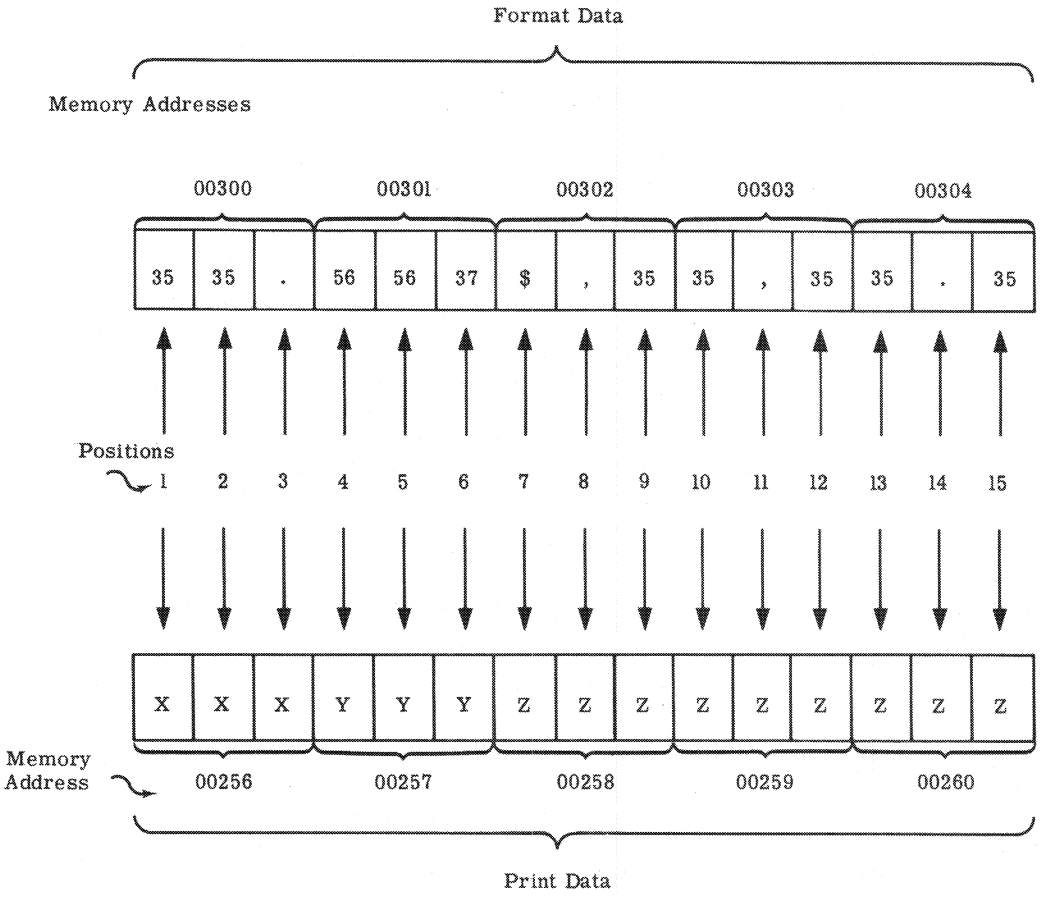
Assume that 5 words of BCD data constitute the information in storage at 00256, 00257, 00258, 00259 and 00260 as follows:



Assume that it is desired to print 4 words of the above data as follows:

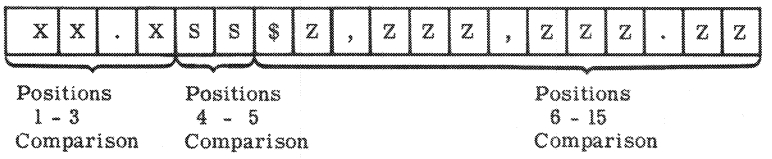


Further assume that insignificant blanks and zeros must be deleted from the gross earnings field, before it is printed. Design the format control data to achieve the desired result. Let the origin of the format control data be at memory location 00300 as follows:



Line Image Transferred And Printed.

Created by comparison of Positions in Format Data and Print Data



KEY

Δ	Any bit configuration permissable	35	Octal 35 for Ignore
S	Space	37	Octal 37 for Delete
X	Any alphanumeric data character	56	Octal 56 for Delete/Skip

Operation Code	Operand Address	Modification Word
SEL **SLW** **SLW**	**N**	**P**

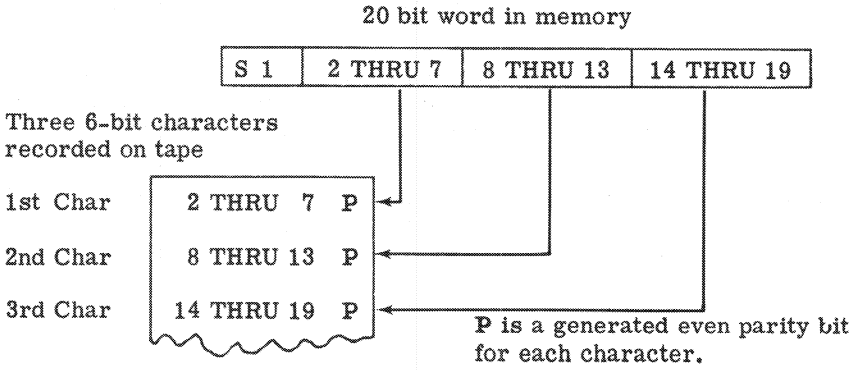
SLEW PAPER N LINES. P is the address (0 thru 7) of the controller to which the high speed printer is attached. SEL is the mnemonic code for the selection function. SLW is the mnemonic code for Slew (Space) Paper. N is the number (0 thru 63) of lines to be slewed (spaced) before printing the next line.

Operation Code	Operand Address	Modification Word
SEL **SLT** **SLT**		**P**

SLEW PAPER TO TOP OF PAGE. P is the address (0 thru 7) of the controller to which the high speed printer is attached. SEL is the mnemonic code for the selection function. SLT is the mnemonic code for Slew Paper to Top of Next Page.

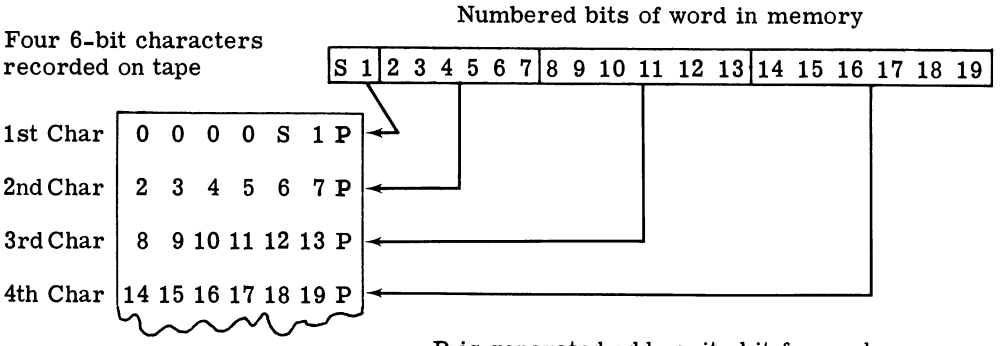
MAGNETIC TAPE SUB-SYSTEM

In the alphanumeric (binary coded decimal) mode of operation, each binary coded decimal character is stored on tape as a corresponding magnetic tape character; that is, a memory word is stored as three magnetic tape characters. Some of the memory bit patterns are altered as they are recorded, making the GE 225 magnetic tape compatible with computer systems now in use. Further information on this point is given in the Appendix. The alteration of the character codes when writing and reading magnetic tape is automatic.



Bits S and 1 are not recorded on tape when writing in (binary coded) decimal mode. Writing mixed binary and BCD words on tape must be done in the binary mode. When reading tapes in the (binary coded) decimal mode, bits S and 1 are set to zero in memory for each word read from tape.

In the binary mode of operation the 20 bits of a memory word are written on magnetic tape as 4 magnetic tape characters. Three of the magnetic tape characters contain 6 bits of data each, while the fourth magnetic tape character contains only 2 bits of data. The four remaining bits in this character will be written as zeros. These 4 zeros will automatically be inserted when recorded and ignored when read back from tape.



P is generated odd parity bit for each character.

The GE 225 tape system has error detection circuits to insure accuracy in transferring information between memory and tape. These error detection circuits are:

1. Controller Input/Output Register Overflow/Exhaust. Checks that capacity of Input/Output Register is not exceeded.

2. Lateral (vertical) Parity. This parity bit checks the accuracy of each character when read from tape.

3. Horizontal Parity. This is a parity check on each of the seven record tracks on tape which is recorded at the end of each record.

4. Modulo Three or Four. When information is read from tape a check is made to determine that the proper number of characters (three in decimal mode and four in binary mode) constitute a word.

5. Write Check. All data written on magnetic tape is checked immediately after it is written by reading back on the physically separate read head and verifying lateral parity and horizontal parity.

Operation Code	Operand Address	Modification Word
BDM	**C+T/F**	**P**

BRANCH ON DATA MATING FUNCTION INTERROGATED CONDITIONS. P is the address of the magnetic tape controller to be interrogated. C is the number of the specific condition to be tested. Both C and P have the range 0 to 7. If C+T is specified and the condition tested (C) is true, the computer takes the next sequential instruction; if it is not true, the computer skips the next instruction and executes the second sequential instruction. If C+F is specified and the condition tested (C) is not true (false), the computer takes the next sequential instruction; if it is true, the computer skips the next instruction and executes the second sequential instruction.

Condition	
0	Controller Busy
1	End of File
2	End of Tape
3	Any Tape Rewinding
4	Parity Error
5	Input/Output Buffer Error
6	Mod 3 or Mod 4 Error
7	Any Error

Operation Code	Operand Address	Modification Word
SEL		**P**
XXX	**M**	
XXX	**N**	**T**

MAGNETIC TAPE CONTROL INSTRUCTIONS. P is the address (0 thru 7) of the magnetic tape controller to be selected for this magnetic tape instruction. For tape read instructions M is the address of the first word of a block of N words in memory which is to receive data from magnetic tape; for tape write instructions M is the address of the first word of a block of N words in memory which is to be written on magnetic tape. N is the maximum number of words to be read or the exact number of words to be written. T is the number of the tape handler unit (0 thru 7) to be used. **XXX** is the mnemonic code for the specific tape movement desired. The mnemonic codes for specific tape movements are:

WTD	Write tape in decimal (alphanumeric) mode
WTB	Write tape in binary mode
RTD	Read tape in decimal (alphanumeric) mode
RTB	Read tape in binary mode
RWD	Rewind tape to leader
BKW	Backspace one record and position WRITE head
BKR	Backspace one record and position READ head
WEF	Write END of FILE character (0001111)

After reading (in either binary or decimal mode) N words from magnetic tape into memory starting at location M, memory location M + N will contain zeros if exactly N words were read from a record on tape containing N words. If the number of words contained in the record currently read is less than N, then only the contents of the record will be stored in memory; and the 2's complement of the residue (N - record length) will be stored in memory cell M + N with a one-bit in the sign position. If the number of words in the record is greater than N, then only N words will be stored in memory and the increment (record length - N) will be stored in memory cell M + N with a zero in the sign position. M is not automatically modified. In order to forward space (skip) one record, the RTD or RTB command is used with N set equal to zero.

MASS RANDOM ACCESS FILE SUB-SYSTEM.

Each Mass Random Access File (MRAF) consists of either 16 or 64 storage discs. From one to four 16-disc MRAFs or one 64-disc MRAF may be attached to a controller unit and, through the data mating function, to the Central Processor. A MRAF record consists of 64 20-bit information words. Each word is an image of the corresponding word in computer memory. In addition, there is a 65th odd-parity check word which insures against loss of information during data transfers. Six numbers are required to address a specific record:

1. Controller Address (0 thru 7)
2. File Number (0 thru 3)
3. Disc Number (0 thru 15) or (0 thru 63)
4. READ-WRITE Head Number (0 thru 7)
5. Track Number (0 thru 63)
6. Record Number { (0 thru 15) on 256 outer tracks; (0 thru 7) on 256 inner tracks

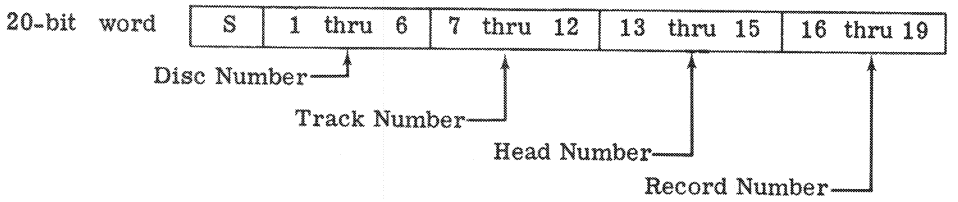
Operation Code	Operand Address	Modification Word
BDM	**C + T/F**	**P**

BRANCH ON DATA MATING FUNCTION INTERROGATED CONDITIONS. P is the address of the mass random access file controller to be interrogated. C is the number of the specific condition to be tested. Both C and P have the range 0 to 7. If C + T is specified and the condition tested (C) is true, the computer takes the next sequential instruction; if it is not true, the computer skips the next instruction and executes the second sequential instruction. If C + F is specified and the condition tested (C) is not true (false), the computer takes the next sequential instruction; if it is true, the computer skips the next instruction and executes the second sequential instruction.

Condition	
0	Controller Busy
1	File #0 Ready
2	File #1 Ready
3	File #2 Ready
4	File #3 Ready
5	Input-Output Error
6	Parity Error
7	Any Error

Operation Code	Operand Address	Modification Word
SEL		**P**
PRF		**R**
OCT	(MRAF address)	

POSITION MRAF. P is the address (0 thru 7) of the controller to which the MRAF is attached. SEL is the mnemonic code for the selection function. PRF is the mnemonic code for Position MRAF to transmit or receive a specific record. R is the number (0 thru 3) of the selected MRAF. The third line contains the actual MRAF address (octal) of the record to be acted upon. The format of this line is:



If S, the sign bit, is 0, the MRAF is positioned to read (or write) the specific record designated by the entire address in bits 1 thru 19. If S is 1, the MRAF is positioned to read (or write) on the track designated by bits 1 thru 15. When a subsequent read (or write) MRAF instruction is given, the first available record from this position will be transmitted to core memory (or written from core memory).

Operation Code	Operand Address	Modification Word
SEL		**P**
RRF	**N**	**R**
RRF	**M**	

READ MRAF. P is the address (0 thru 7) of the controller to which the MRAF is attached. SEL is the mnemonic code for the selection function. RRF is the mnemonic code for Read MRAF. N is the number (1 thru 16) of 64-word records to transmit from the disc storage to core storage. R is the number (0 thru 3) of the selected MRAF. M is the core memory address into which the first word of the first record is to be copied. All following words and records, if any, will be copied into sequentially higher memory locations. The MRAF origination address will be the one at which the MRAF is currently positioned. M must be an even multiple of 64 and is not automatically modified.

Operation Code	Operand Address	Modification Word
SEL		**P**
WRF	**N**	**R**
WRF	**M**	

WRITE MRAF. P is the address (0 thru 7) of the controller to which the MRAF is attached. SEL is the mnemonic code for the selection function. WRF is the mnemonic code for Write MRAF. N is the number (1 thru 16) of 64-word records to transmit from consecutive core storage locations to disc storage. R is the number (0 thru 3) of the selected MRAF. M is the core memory address from which the record(s) will be copied. The MRAF destination address will be the one at which the MRAF is currently positioned. M must be an even multiple of 64 and is not automatically modified.

DOCUMENT HANDLER SUB-SYSTEM

Each of the 14 characters recognized by the Document Handler is converted into a BCD code and stored in the least significant (rightmost) four bits of a designated word in memory; that is, bit positions 16, 17, 18 and 19. All other bits of the word are zeros except for the case of cue characters. If the character read is one of the 4 cue characters, 1 bits are stored in bit positions 0 (sign) and 1 of the word. If a character is invalid (cannot be recognized and translated by the Document Handler), a 1 bit is placed only in bit position 0 (sign). Characters are read from the document in sequence from right to left into successive memory locations until the document is completely read.

Operation Code	Operand Address	Modification Word
BDM	**C+T/F**	**P**

BRANCH ON DATA MATING FUNCTION INTERROGATED CONDITIONS. P is the address of the document handler controller to be interrogated. C is the number of the specific condition to be tested. Both C and P have the range 0 to 7. If C+T is specified and the condition tested (C) is true, the computer takes the next sequential instruction; if it is not true, the computer skips the next instruction and executes the second sequential instruction. If C+F is specified and the condition tested (C) is not true (false), the computer takes the next sequential instruction; if it is true, the computer skips the next instruction and executes the second sequential instruction.

Condition	
0	Handler #1 reading
1	Handler #2 reading
2	
3	
4	Handler #1 feeding
5	Handler #2 feeding
6	Input buffer error (priority)
7	Any error

Operation Code	Operand Address	Modification Word
SEL		**P**
RSD		**N**
RSD	**M**	

READ SINGLE DOCUMENT. P is the address (0 thru 7) of the controller to which the document handler is attached. SEL is the mnemonic code for the selection function. RSD is the mnemonic code for Read Single Document. N is the number (1 or 2) of the selected Document Handler. If N is left blank, the assembly program will assume that Handler #1 is to be used. M is the core memory address into which the first character read from the document will be copied. M is not automatically modified. Single reading of documents can be done at the rate of 600 per minute.

Operation Code	Operand Address	Modification Word
SEL		**P**
RDC		**N**
RDC	**M**	

READ DOCUMENT AND CONTINUE FEEDING NEXT DOCUMENT. P is the address (0 thru 7) of the controller to which the document handler is attached. SEL is the mnemonic code for the selection function. RDC is the mnemonic code for Read Document and Continue. N is the number (1 or 2) of the selected Document Handler. If N is left blank, the assembly program will assume that Handler #1 is to be used. M is the core memory address into which the first character read from the document will be copied. M is not automatically modified. This instruction calls for moving a second document into position for immediate reading after the first document passes the reading head. RDC instructions must be used to achieve the 1200 document per minute reading speed. With each use of the RDC instruction, there are approximately 50 milliseconds (minimum) of processing time available before another RDC or Halt Feeding instruction must be given.

Operation Code	Operand Address	Modification Word
SEL **PKT** **OCT**	(Pocket Address)	**P** **N**

POCKET SELECT. P is the address (0 thru 7) of the controller to which the document handler is attached. SEL is the mnemonic code for the selection function. PKT is the mnemonic code for Pocket Select. N is the number (1 or 2) of the selected Document Handler. If N is left blank, the assembly program will assume that Handler #1 is to be used. The third line contains the address (octal) of the pocket into which the document is to be stacked. The following table gives the octal codes to be used in the pocket selection.

Pocket	Handler #1	Handler #2
SPECIAL	0000001	0000020
0	0000017	0000360
1	0000016	0000340
2	0000015	0000320
3	0000014	0000300
4	0000013	0000260
5	0000007	0000160
6	0000006	0000140
7	0000005	0000120
8	0000004	0000100
9	0000003	0000060
REJECT	0000002	0000040

To be effective, the Pocket Selection command must be given within a maximum of 35 milliseconds after the reading of the document is completed.

Operation Code	Operand Address	Modification Word
SEL **HLT** **HLT**	**M**	**P** **N**

HALT CONTINUOUS FEEDING. P is the address (0 thru 7) of the controller to which the document handler is attached. SEL is the mnemonic code for the selection function. HLT is the mnemonic code for Halt Continuous Feeding. N is the number (1 or 2) of the selected Document Handler. If N is left blank, the assembly program will assume that Handler #1 is to be used. M is the core memory address into which the first character of the document currently approaching the reading head will be copied. This document will be completely read when the HLT command is first used following an RDC instruction. Document feeding may or may not cease. It is therefore necessary to use a second HLT command, and another document may or may not be read. If a document is not read, the handler will remain in a busy condition; that is, the Read Busy signal will be left on. (See ERB instruction.) In no case will a document be only partially read. M is not automatically modified.

Operation Code	Operand Address	Modification Word
SEL		**P**
ERB		**N**
XXX		

END READ BUSY SIGNAL. P is the address (0 thru 7) of the controller to which the document handler is attached. SEL is the mnemonic code for the selection function. ERB is the mnemonic code for End Read Busy. N is the number (1 or 2) of the selected Document Handler. If N is left blank, the assembly program will assume that Handler #1 is to be used. The third line must be present, but it is not used in this instruction. The programmer may use this line as working storage or as constant storage. XXX may be any one of the following pseudo-instruction mnemonics: ALF, DEC, or OCT. When a HLT command is given and a document is not read, the ERB instruction is used to reset the Document Handler to a ready condition for further operation.

E. THE GENERAL ASSEMBLY PROGRAM

GENERAL DESCRIPTION

The General Assembly Program is a basic assembly routine with extensive error checking features and provision for program modification. With the General Assembly Program, the programmer writes his own GE 225 program employing symbolic notation rather than the absolute code of the computer. However, both the single address format and the general structure of a computer instruction word are retained. This symbolic program is read into memory along with the General Assembly Program itself. The output from the computer is the user's original symbolic program, now converted into absolute machine language. One symbolic instruction is usually translated into one computer instruction. This program is now ready to be read into memory for execution.

The symbolic notation selected to designate each instruction is a mnemonic code. These mnemonic codes are carefully chosen to provide maximum significance to the user. For example, the mnemonic code for the addition instruction is ADD, for subtraction the code is SUB, and so forth. The assembly program translates these mnemonic codes into the absolute code of the computer.

Memory addresses may be assigned using decimal notation (location 1500 for example) or using symbolic notation chosen for maximum convenience to the programmer. If an alphabetic (NETPAY) or an alphanumeric (TAX3) is used to designate a memory address, the General Assembly Program automatically assigns memory locations. The programmer need only specify the starting address into which the first instruction of the program is stored.

PSEUDO-INSTRUCTIONS

In addition to the mnemonic codes for the instructions in the normal repertoire of the GE 225, the General Assembly Program uses other mnemonic codes to define a group of terms called pseudo-instructions. A pseudo-instruction is a symbolic representation of information required by the General Assembly Program for the assembly of a program. The pseudo-instruction has the same general form as a computer instruction, and it is listed like a normal instruction in the preparation of a program; however, it is never executed by the computer as an actual instruction. For example, ORG is a pseudo-instruction which may be used to indicate the starting address in the assignment of a program to memory. Thus, ORG 400 may indicate that a program is to enter memory with the first instruction starting at location 400. The General Assembly Program automatically assigns succeeding memory locations to the remaining instructions of the program. ORG never enters memory to become a part of the program as do regular instruction words. In a similar manner, the other pseudo-instructions provide information to the assembly routine but do not actually become part of the final program.

A list of pseudo-instructions available for use with the General Assembly Program is given below. More specific details for their use will be found in a later section.

ALF ALPHANUMERIC. Used for program headings. The first three characters in the operand address field are converted to a binary coded decimal word and assigned a memory location.

BSS BLOCK STARTED BY SYMBOL. Saves a block of memory locations of specified amount. Amount may be decimal or symbolic. If symbolic, number of locations specified by the symbol is saved.

DEC DECIMAL. Decimal numbers are converted to binary. Limited to one word.

DDC DOUBLE DECIMAL. Used for establishing decimal constants larger than 524287 or, in other words, larger than can fit into one word.

END END OF PROGRAM. Punches all assembled instructions to this point and punches control card indicating where to start program. END indicates the end of assembly and should be used only at the end of a program.

EQU EQUAL. Defines a symbol which can be equal to a decimal or another symbol.

OCT OCTAL. Gives the binary representation of up to 7 octal digits, left justified.

ORG ORIGIN. Designates the starting storage location of a program or a portion of a program in memory. Address may be decimal or symbolic.

REM REMARKS. Remarks immediately following this pseudo-instruction are written by programmer for reference only and are not processed by assembly program. Remarks are part of the source program and output listing but not the final object program.

TCD TRANSFER CARD. Like END, TCD punches all instructions to this point and punches a control card; but TCD allows assembly to continue.

THE GE 225 CODING SHEET

The General Assembly Program coding sheet is divided into six fields: Symbol, Operation, Operand, X Register, Remarks, and Sequence. The numbers 1 through 80 in the header information on each sheet correspond to the column numbers of a standard 80-column punched card. When a symbolic program is punched into cards, columns 7, 11 and 21 are not used; these blank columns serve to separate important fields.

1. Symbol Field.

Columns 1 through 6 constitute the symbol field. Symbols may consist of from 1 to 6 digits. One of the digits in the symbol field must be alphabetic. HOPE, CONST3 are legitimate symbols; 345 is not a legitimate symbol. A symbol may be either right or left justified in the symbol field; that is, the symbol AB in columns 1 and 2 is the same symbol as in columns 5 and 6. The plus and minus signs cannot be used in the symbol field because they are used in the operand field for relative addressing. A blank (space) in the symbol field is ignored by the assembly.

2. Operation Field.

Columns 8, 9 and 10 make up the operation field. Any of the mnemonic codes for the normal computer instructions (LDA, BRU, etc.) or for the pseudo-instructions (ORG, DEC, etc.) can be placed in this field.

3. Operand Field.

Columns 12 through 19 constitute the operand field. Operands may be an alphabetic or alphanumeric symbol up to six characters in length or a decimal number and can be positioned anywhere in the operand field. The plus and minus sign are used only in the operand field and only when expressing a relative address. The subject of relative addressing is discussed in a later section. All numbers appearing in the operand field are considered to be decimal except when following the operation OCT and ALF. Numbers following OCT are treated as octal and are converted to their binary equivalent. Digits following ALF are converted to their binary coded decimal equivalents. Blanks (spaces) in the operand field are ignored.

4. X Register.

Column 20 designates the X Register (automatic address modification word). A decimal 1, 2 or 3 in this field designates modification word 00001, 00002 or 00003 respectively in memory. A zero in this column indicates that address modification is not to be performed. A blank (space) is considered a zero.

5. Remarks Field.

Columns 22 through 75 make up the remarks field. Remarks are written in this field for reference by the programmer. These remarks are punched in the assembly program source deck, but the information is not carried through to the final object program. Thus, information in the remarks field is obtained only on a printed listing as a part of the assembly process.

6. Sequence Field.

Columns 76 through 30 constitute the sequence field. Each card is to be given a sequence number so that a deck can be sorted into proper order should the cards get out of sequence.

RELATIVE ADDRESSING

The General Assembly Program provides facility for the assignment of addresses relative to some starting point (relative addressing). Assume, for example, that the programmer has established that the symbol B is equal to memory location 00500. Using the technique of relative addressing, memory location 00510 can now be addressed by simply writing B + 10 in the operand field of the coding sheet. This is illustrated as follows:

GENERAL ELECTRIC

GE 225 CODING SHEET

PROBLEM: _____

WRITTEN BY: _____

PAGE ___ OF ___

SYMBOL	OPR	OPERAND	X	REMARKS	SEQUENCE
1 6	8 10	12 19	20	22 75	76 80

Figure 16 GE225 Coding Sheet

Symbol	Operation	Operand
B	EQU	500
	LDA	B
	↓	↓
	LDA	B+10

The EQU pseudo-instruction establishes that the symbol B is equal to memory location 00500. The instruction LDA (Load Register A) loads the A Register with the contents of memory location 00500. The next LDA instruction, some program steps later, loads register A with the contents of B + 10 (location 00500 + 10 = 00510).

Another illustration of relative addressing is the following example with reference to pseudo-instruction ORG.

Symbol	Operation	Operand
C	EQU	200
	ORG	C
	↓	↓
	ORG	C+1000

The first ORG (Origin) establishes that information is to be placed in memory starting at the location indicated by the symbol C; that is, at memory location 00200. The second ORG, some program steps later, establishes that information is to be placed in memory starting at location 01200.

Because the plus and minus signs are used in relative addressing operations in the operand field, they cannot be used as symbols in the symbol field of the coding sheet.

PSEUDO-INSTRUCTION USAGE

In this section some examples of the use of GAP pseudo-instructions are given. The headings Symbol, Operation, Operand and Remarks correspond to the headings on the GAP coding sheets.

1. **ALF** The psuedo-instruction ALF (alphanumeric) is used for program headings. The first three characters of the operand field are converted to binary coded decimal equivalents and this BCD word is assigned a memory location. Three alphabetics are converted per pseudo-instruction. For example, the heading NAME RATE HOURS could be entered as program constants in the following manner:

Symbol	Operation	Operand
A	ALF	NAM
	ALF	E
	ALF	RA
	ALF	TE
	ALF	HOU
	ALF	RS

This heading can be picked up by addressing the symbol A and entering a program loop to pick up the other locations. Note from the arrangement of information in the operand field that three spaces separate NAME and RATE and that one space separates RATE and HOURS.

2. **BSS** The pseudo-instruction BSS (Block Started by Symbol) is used to reserve a block of memory storage. For example:

Operation	Operand
BSS	50

The assembly program reserves the next 50 memory locations, and the assignment of memory addresses to instructions continues with the 51st memory locations following.

The BSS pseudo-instruction is conveniently used to reserve input memory locations when punched cards are read. For example, when a Read Cards Decimal (RCD) instruction is given, the information from the first card is stored in memory location 00128 (or a multiple thereof) through the next 26 memory locations. Assume that the programmer is at first undecided as to the memory location to be used as the starting address for input when cards are read in the decimal mode.

Therefore, each time an RCD command is used in the program, the operand address is indicated simply by the symbol IN:

Operation	Operand
RCD	IN

Later, the programmer decides to use memory location 0128. The symbol IN must then first be defined in the program as follows:

Symbol	Operation	Operand
IN	ORG	128
	BSS	27

The result is that each time the symbol IN is addressed the assembly assigns memory location 00128, so that information from the card is stored starting at that memory location. The BSS pseudo-instruction reserves the next 26 memory locations for the remainder of the card.

3. **DEC** The pseudo-instruction DEC (Decimal) converts decimal numbers to binary:

Symbol	Operation	Operand
HOPE	DEC	560

The constant 560 can be called for by the programmer using the symbol HOPE. A negative 560 can be created by simply writing -560 in the operand field. The decimal number 524,287 is the largest decimal number that can be associated with DEC. Larger decimal constants can be established by the pseudo-instruction DDC, which is discussed below.

A convenient symbol for the programmer to use to represent a decimal number may often be the alphabetic representation of the decimal number.

Symbol	Operation	Cperand
THREE	DEC	3

4. **DDC** The pseudo-instruction DDC (Double Decimal) is used to establish decimal constants larger than 524,287. For example, the decimal number 576,897 which is referred to symbolically as constant 1 is established as follows:

Symbol	Operation	Operand
CONST1	DDC	576897

The assembly program converts the decimal number 576897 to a double length binary word.

If the decimal constant is larger than the eight digit positions allowed for the operand field, the digits in excess of eight are written on the next line of the operand field.

Symbol	Operation	Operand
CONST2	DDC	-1234567 89

5. **END** The pseudo-instruction END (End of Program) indicates the end of the program to be assembled. END causes all instructions to this point to be output as well as a control record indicating where to start the program.

Operation	Operand
END	400

The memory address 00400 is converted to binary and output in the control record. It indicates the origin of the program.

6. **OCT** The pseudo-instruction OCT (Octal) converts up to seven octal digits into a binary equivalent. These digits are left justified prior to conversion.

Symbol	Operation	Operand
CONST1	OCT	0371652

OCT converts 0371652 from octal into its binary equivalent and stores it so that the symbol CONST1 can be used as the memory address where the binary constant is stored. The programmer may use a negative octal number in which case seven octal digits and the minus sign are written in the operand portion:

Symbol	Operation	Operand
CONST2	OCT	-0371652

As in the previous example, the symbol CONST2 can be used as the memory address where the binary equivalent of the stated octal number is stored. The effect of the minus sign is to place a 1 bit in the zero bit position of the word.

7. **ORG** The pseudo-instruction ORG (Origin) is used to indicate the location of the first instruction of the program when it is stored in memory. ORG can be used in the program as many times as desired. For example, assume that the first ORG directs that the program is to be stored in successive memory locations starting at location 00400. After 200 memory locations are filled with program steps, however, the rest of the program is to be stored starting at location 01000.

Operation	Operand
ORG	400
200 program instructions	
ORG	1000

The memory locations between 00600 and 01000 are not used for storage of the program.

The memory address of ORG may be symbolic as well as decimal. The definition of the symbol must precede this use of the symbol as illustrated below.

Symbol	Operation	Operand
A	EQU	512
	ORG	A

The pseudo-instruction EQU establishes that the symbol A is equal to memory location 00512. The pseudo-instruction ORG sets the value of A (i.e., 00512) as the origin for the storage of the program in memory.

8. **REM** When the pseudo-instruction REM (Remarks) is in the operation field, the programmer's remarks immediately following are not processed by the assembly program:

Operation	Operand	Remarks
REM		Programmer's remarks

GENERAL ELECTRIC — GE 225 CODING SHEET

PROBLEM: _____ PAGE ___ OF ___

WRITTEN BY: _____

SYMBOL	OPR	OPERAND	X	REMARKS	SEQUENCE
1 6	8 10	12 19	20	22 75	76 80
$4800	DEC	4800000			
ZERO	DEC	0			
FICA	DLD	YTD		LOAD YEAR-TO-DATE IN A AND Q	
	DCU	$4800		SUBTRACT $4800	
	BPL			TEST FOR PLUS	
	BRU	MAIN		NO FICA DEDUCTION. GO TO MAIN PROGRAM.	
	DLD	$4800		LOAD $4800	
	DSU	YTD		SUBTRACT YEAR-TO-DATE	
	DSU	CE		SUBTRACT CURRENT EARNINGS	
	BPL			TEST FOR PLUS	
	DLD	ZERO		ALL CE TAXABLE. CLEAR A AND Q	
	DAD	CE		ADD CURRENT EARNINGS	
	FROM THIS POINT ON FICA TAX WILL BE COMPUTED ON THAT PORTION OF CURRENT				
	EARNINGS IN REGISTERS A AND Q.				

Figure 17 GE 225 Coding Sheet Example

The programmer's remarks appear only on a printed list which is produced as part of the assembly process.

9. **TCD** The pseudo-instruction TCD (Transfer Control Data) outputs all instructions to this point in the program and then outputs a control record just as does the pseudo-instruction END. TCD, however, does not indicate end of assembly. TCD allows the table of symbols to be carried over to the next assembly. This permits the assembly of more than one program using the same basic constants. The operand address of TCD indicates the starting address for the program.

Operation	Operand
TCD	500

The 500 is converted to binary and indicates the origin of the program. This location is punched on the control record.

EXAMPLE - (page 50)

This example illustrates the decisions which are required in the calculation of Social Security (FICA) tax during payroll computations. Assume that payroll data (representing an employee master payroll record) is in memory. If Year-to-Date (before current week's earnings) gross pay is equal to or greater than $4800, control is transferred to the main program since no additional FICA will be deducted (this year). Should this first test indicate that either all or a portion of this week's pay is taxable, the new YTD gross pay is computed (old YTD + this week's current earnings) and subtracted from the $4800 limit to determine whether all or just a portion of this week's earnings are taxable. If this second test indicates that the entire week's pay is taxable, the A Register will be cleared, and this week's entire current earnings loaded into A for subsequent tax computation. If this second test indicates that only a portion of this week's pay is taxable, the taxable portion is computed and left in the A Register for further computation. Thus for example, if old YTD = $4750, and this week's CE $125, only $50 of this week's pay is taxable; the remair ing $75 exceeding the $4800 limit.

F. CONTROL CONSOLE OPERATION

The primary function of the Control Console is to provide an indicating control center for the computer operator from which he has visual representation and manual control of operation of the system.

The Indicator Panel, consisting of display lights of the A, I and P Registers and various alarm and ready status indicators, occupies the upper three quarters of the Control Console; the Control Panel occupies the lower quarter.

INDICATOR PANEL

1. Three registers are displayed by lights on the indicator panel:

   a) The thirteen-bit P counter.
   b) The twenty-bit I Register.
   c) The twenty-bit A Register.

2. Switches:

   a) The Save P switch inhibits the normal advance of the P Counter so that the contents of the P Counter are retained and the execution of the addressed instruction is repeated. This switch is used primarily for maintenance purposes.

   b) The Reset A switch clears the A Register; i. e., sets it equal to zero. This switch has no effect when the Auto-Manual switch is in the automatic position.

   c) Twenty switches are provided to set up any one-word bit configuration into the A Register. These switches each have 3 positions which have the following significance:

      UP - If the Auto-Manual switch is in the Manual position, a 1 bit is set into the corresponding A Register position. If the Auto-Manual switch is in the automatic position, there is no effect. The UP position is spring loaded and will return to the CENTER position when released.

      CENTER - No effect.

      DOWN - When placed in the DOWN position, the switch will not return to CENTER when released. In this position the switches may be "read" by the RCS instruction as discussed in the instruction repertoire.

3. Ready Lights (GREEN):

   a) The Card Reader Ready light indicates the card reading equipment is ready to operate; i. e., the card hopper is not empty and a card reading operation is not currently being performed.

   b) The Card Punch Ready light indicates the card punch equipment is ready to operate; i. e., the card hopper is not empty, the stacker is not full, a card is located at the first punch station, and a card is not currently being punched.

   c) The N Register Ready light indicates that the N Register is available for input-output.

4. Alarm Lights (RED):

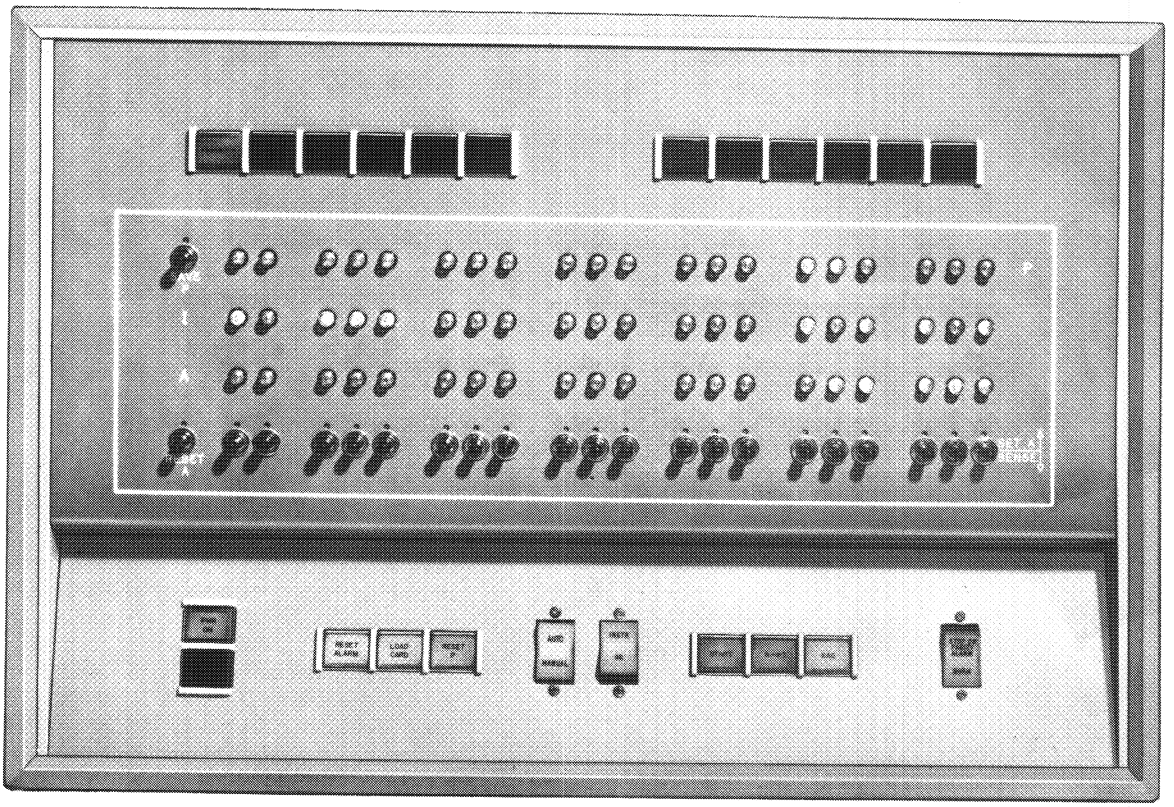
   a) The Priority alarm light indicates that the Central Processor has lost priority (access to the memory). This indicator will also be turned on when the Central Processor is operating in the Manual mode.

   b) The Parity alarm light indicates a parity error.

   c) The Overflow alarm light indicates overflow in the Arithmetic Unit (i. e., its capacity has been exceeded) or overflow in the A Register as the result of a shift left instruction. The computer does not halt.

   d) The Card Punch alarm light indicates an attempt to execute a WCB or WCD instruction when the Card Punch is not in the ready condition. The computer halts.

   e) The Card Reader alarm light indicates an attempt to execute a RCB or RCD instruction when the Card Reader is not in the ready condition. The computer halts.



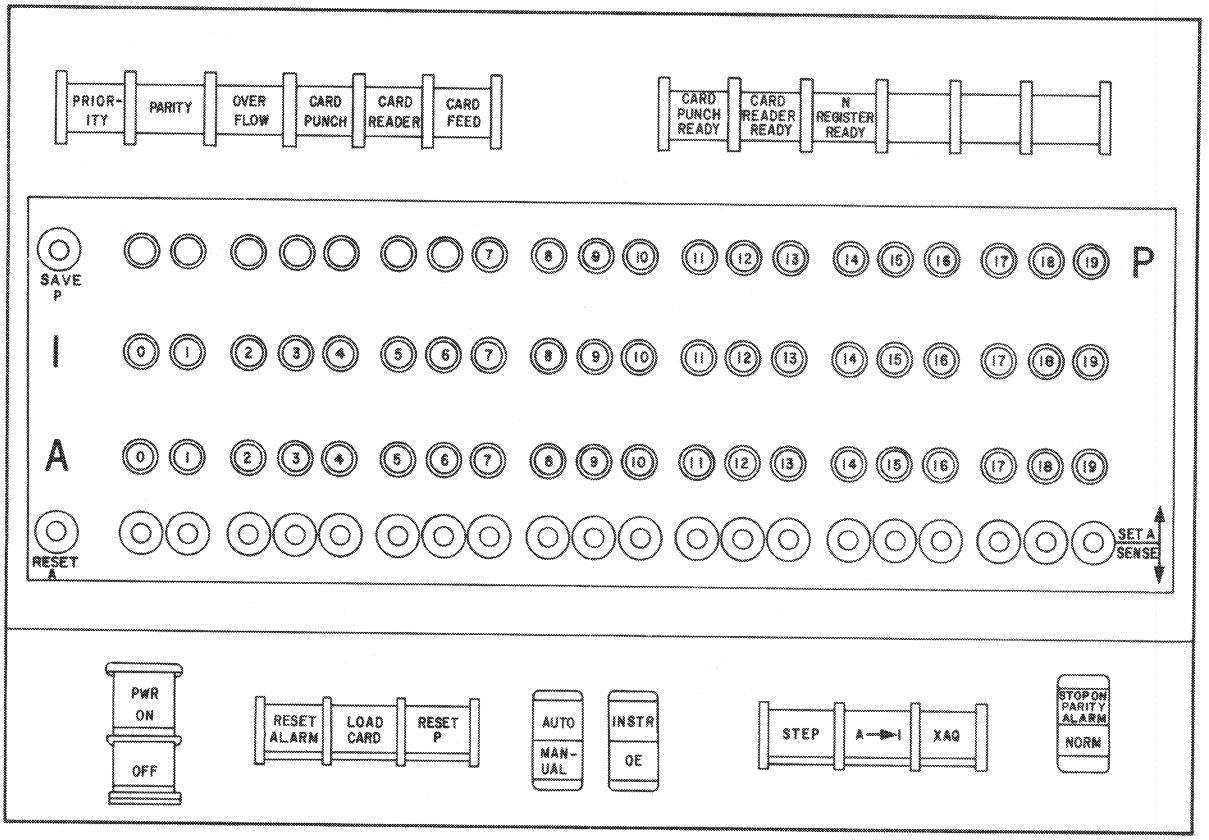


Figure 18 Control Console

f) The Card Feed alarm light indicates an attempt to execute a RCB or RCD instruction when no card has been positioned on the sensing platform and the hopper is not empty, or if a misfeed occurs during card reading. The computer halts.

CONTROL PANEL

1. Push Buttons

a) The two left-most push buttons are the computer Power On and Power Off controls.

b) The Reset Alarm push button resets (clears) all alarms. This switch is only effective when the Auto-Manual switch is in the Manual position.

c) The Load Card push button allows one card to be sent through the card reading mechanism, and its contents are read in binary mode into memory starting at location 0000. The load card button offers a method of initiating the first program card through the card reading mechanism. This switch is only effective when the Auto-Manual switch is in the Manual position.

d) The Reset P push button clears the P Counter, i.e., sets it equal to zero.

e) The Step push button allows step-by-step operation of the computer when the Auto-Manual switch to the left of it is in the manual position. Otherwise it starts the computer into automatic operation.

f) The A→I push button transfers the contents of the A Register into the I Register. An instruction can be set up in the A Register by means of the twenty switches available and then transferred to the I Register. This switch has no effect in the Automatic mode of operation.

g) The XAQ push button allows an exchange of information between the A and Q Registers; that is, the contents of A goes into Q and the contents of Q goes into A. This switch has no effect in the automatic mode of operation.

2. Switches

a) When the Auto-Manual switch is in the automatic position, the computer executes instructions in the normal sequential manner. When the switch is in the manual position, the computer executes instructions in a step-by-step procedure, going from one step to the next each time the Step push button is pushed. The exact step-by-step procedure followed in the manual mode of operation is determined by the position of the switch discussed below.

Placing the Auto-Manual switch in the manual position while the computer is operating will bring it to a halt after completion of the instruction being executed. At this time the I Register lights will display the next instruction to be executed; the P Register lights will display the address of the instruction following the instruction currently displayed in the I Register; and the A Register lights will display the contents of the A Register after the execution of the last instruction. (If the Instruction-OE switch is in the OE position, the computer will be stopped immediately at the end of the current word time).

b) When the Instruction-OE switch is in the Instruction position, one instruction is executed each time the Step push button is pushed. When the switch is in the OE (Operation Enable) position, one word time (cycle) is completed each time the Step push button is pushed. The OE position is intended for maintenance use only.

c) The right-most switch is concerned with parity alarm. The switch has two positions; (1) Stop On Parity Alarm which stops the computer for any parity error and (2) Norm which does not stop the computer when there is a parity error. The switch will be placed in the Norm position when the program being run has been prepared to take remedial action when any parity error occurs. This permits processing to continue without interruption.

G. SYSTEM ERROR CHECKING AND RECOVERY FEATURES

The GE 225 is a fully transistorized system and represents a new standard of reliability in commercial computer design. Accuracy is assured by built-in parity checking circuits that check against the loss of information during transfers to and from peripheral equipment. The design objective is to reduce the occurrence of undetected errors to zero.

Wherever feasible, a detected error will not cause a halt or hang-up of the system, but will initiate a programmed rescue and recovery routine. Entry into such routines is made possible by appropriate branch instructions. For example, if an error is detected during the reading of magnetic tape, an error indicator is set. This indicator may be interrogated by the programmer using a branch instruction which then routes the program to a recovery routine. The recovery routine might be programmed to reposition the tape and attempt to reread any number of times at the programmer's discretion. If the error persists (as, for instance, a media error would), then and only then would an error halt be programmed. The remainder of the system continues in operation and treats the halted unit as if it were busy. An Attention Indicator will be prominently displayed on the appropriate control panel.

The Central Processor Control Panel contains a switch which controls the action upon the occurrence of parity errors. In the Halt position the computer will halt on an error. In the Normal position an error will not cause a halt. The switch will be placed in the Normal position when a recovery routine has been programmed. This permits processing to continue without interruption.

The following types of errors may be encountered:

1. Media Error

   An illegal character configuration occuring on an input or output medium. On an input medium, this error is non-recoverable. On an output medium, the character may be erased and rewritten.

2. Program Error

   A non-existent operation code or incorrect unit address selection.

3. Transmission Error

   An information parity error generated during transmission between units.

4. Unit Operating Error

   Information or control error generated within a unit.

H. PROGRAMMING NOTES

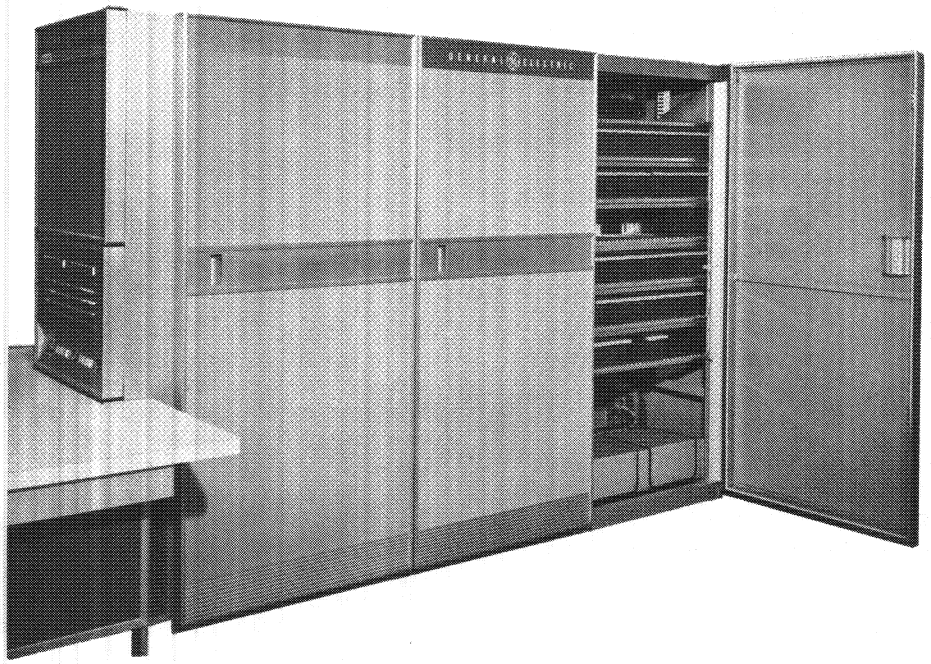


Figure 19 Central Processor

PROGRAMMING MACHINE CALCULATIONS

Conversion Routines

All calculations within the computer deal with numbers represented as binary numbers. Since input data will normally be represented as binary coded decimal (BCD) numbers, it will be necessary to convert binary coded decimal (BCD) numbers to binary numbers before the calculations are performed. Likewise, the results of calculations are expressed as binary numbers. It will be necessary to convert the binary result to a binary coded decimal (BCD) number in order to print or punch the number as a decimal. The programmer will have to provide for this conversion of numbers. Conversion is facilitated, however, by "package routines" which are available with the GE 225. The programmer utilizes the package routines by specification on the "calling sequence". For example, the calling sequence for a BCD to binary conversion assuming punched card input would be specified as follows:

Location	Instruction Word: Operation	Operand	Modification
A	SPB	Sub-routine origin	1
A + 1	DEC	Card image origin	
A + 2	DEC	Card column starting location	
A + 3	DEC	Field size	
A + 4	BRU	Error return	
A + 5	Normal return		

If there is a BCD number starting in card column 19 and ending in card column 25 and the card image origin is at 0128, the calling sequence would be:

Location	Instruction Word: Operation	Operand	Modification
A	SPB	Sub-routine origin	1
A + 1	DEC	0128	
A + 2	DEC	19	
A + 3	DEC	7	
A + 4	BRU	Error sub-routine origin	
A + 5	Normal return		

The converted BCD number will appear in the "A" and "Q" registers as a double precision binary number. The least significant half will be in "Q" and the most significant half will be in "A". The field size of the number to be converted is limited to 11 digits.

The calling sequence for binary to BCD conversion assuming punched card output would be specified as follows:

Location	Instruction Word: Operation	Operand	Modification
A	SPB	Sub-routine origin	1
A + 1	DEC	Card image origin	
A + 2	DEC	Card column ending	
A + 3	DEC	Field size	
A + 4	BRU	Error return	
A + 5	Normal return		

If there is a binary number contained in the A and Q registers which is to be converted into a BCD number starting in card column 18 and ending in card column 23, and the card image origin is at 0128, the calling sequence would be:

Location	Instruction Word Operation	Operand	Modification
A	SPB	Sub-routine origin	1
A + 1	DEC	128	
A + 2	DEC	23	
A + 3	DEC	6	
A + 4	BRU	Error sub-routine origin	
A + 5	Normal return		

If the field size specified is larger than the number of digits produced, the remaining positions of the field will contain leading zeros. If the integer is negative, the 11 punch will be placed over the units position of the card field.

Double Length Operations

The programmer may specify the use of other subroutine packages which will be available with the GE 225. For example, a subroutine will be available to accomplish multiplication of 2 word-length numbers by 2 word-length numbers. The double multiply subroutine is utilized as follows:

DOUBLE MULTIPLY

If the multiplicand is in the A and Q Registers, the least significant half in Q, the most significant half in A, the calling sequence for the double multiply subroutine would be:

Location	Operation	Operand	Modification
A	SPB	Sub-routine origin	1
A + 1	Normal return		

Product will appear at locations 4090, 4091, 4092 and 4093 with least significant half in 4093-4092 and most significant half in 4091-4090. Multiplier must be in location 4094 and 4059; least significant half in 4095, and most significant half in 4094.

If a double length word is being multiplied by a single length word, the single length word should be the multiplier in location 4095 and zeros in location 4094. Result will be a maximum of 3 word lengths: 4091, 4092 and 4093. Also, a sub-routine will be available for the division of 2 word-length numbers by 2 word-length numbers. The double division sub-routine is utilized as follows:

DOUBLE DIVISION

If the dividend is in A and Q with the least significant half in Q and the most significant half in A, the calling sequence for the double divide sub-routine would be:

Location	Operation	Operand	Modification
A	SPB	DDV	1
A + 1	LDA	DIVISOR	
A + 2	LDA	REMAINDER	
A + 3	Normal return		

The quotient will be in A and Q with the least significant half in Q and the most significant half in A. The remainder will be stored in two consecutive locations addressed by the operand of A + 2. The Divisor is in two consecutive locations addressed by the command in A + 1.

The equivalent of the double multiply subroutine and double divide subroutine is provided for addition and subtraction by double length add and double length subtract commands. Because of their more frequent use, these instructions are provided as part of the normal command repertoire of the computer.

The path of information transfer during arithmetic calculations with single word length instructions is from main memory to the A register and from the A register to main memory, one word at a time. Access to the Q register required in multiplication and division is achieved through the A register. The double length add, double length subtract, double length load, and double length store commands provide the programmer with extra facility to transfer two words of data between the main memory storage areas and the A and Q registers. The double length commands must specify even numbered memory locations.

Overflow

A value stored in the core memory may be added to, or subtracted from, the contents of the A register. The capacity of the A register may be exceeded in execution of add, subtract, or multiply commands. In this event, the overflow indicator is turned on, the high-order bit of the result is lost, and the sign of the result is reversed. Overflow may also occur in the divide command; for which, see below. Examples of binary arithmetic may be found in the Appendix.

Multiply Command

It will be noted that the multiply command adds the contents of the A register to the product of the number in the Q register and the number in the specified memory location. Therefore, for normal multiplication it is necessary to replace the contents of the A register with zeros. The MAQ instruction loads zeros into the A register and also moves the multiplicand to the Q register. This instruction would normally be used in the program before a multiplication command.

Divide Command

In order to divide correctly, the absolute value of the divisor must be greater than the absolute value of that portion of the dividend in the A register. If this rule is violated, the overflow indicator will be turned on. Therefore, the programmer should interrogate the overflow indicator and program a procedure to deal with overflow if there is a possibility of violating this rule.

Scaling

The movement of the decimal point to the right or left in order to properly align numbers is called "scaling" or "decimal positioning". Before numbers can be correctly added or subtracted in the computer the number of places to the right of the decimal point of both numbers must be the same. For example, in order to add 3.0 to 4.16 one must first arrange 3.0 to correspond to 3.00 and then add. If the decimal point is moved to the right in order to prepare for calculations, the number is "scaled to the right"; if the decimal point is moved to the left, the number is "scaled to the left".

When two numbers are multiplied, the number of places to the right of the decimal point in the product is the sum of the places to the right of the decimal point in both the multiplier and the multiplicand. If it is desired to scale the product (which is expressed as a binary number), the product must be divided by a constant that is the binary equivalent of the appropriate power of 10.

Rounding

After a calculation has been completed, it is common to round the result to the next highest integer. "Rounding" is accomplished by adding a "5" into the position adjacent to the position to receive any carry. Since, within the GE 225, all calculations are performed with binary numbers, the proper rounding factor of "5" is expressed in binary and is carried as an appropriate constant within memory. After the round factor is added, the positions to the right of the position which receives any carry are deleted through scaling.

CALCULATIONS ON DATA STORED WITHIN THE 225

Load A register	LDA
Store A register	STA
Add	ADD
Subtract	SUB
Multiply	MPY
Divide	DVD
Double Length Add	DAD
Double Length Subtract	DSU
Double Length Load	DLD
Double Length Store	DST
Move A to Q	MAQ
Load Zero	LDZ
Branch on Overflow	BOV
Branch on Zero	BZE

LDA Y LOAD A

The contents of Y (s, 1-19) replace the contents of A (s, 1-19). The contents of Y are not changed.

STA Y STORE A

The contents of A (s, 1-19) replace the contents of Y (s, 1-19). The contents of A are not changed.

ADD Y ADD

The contents of Y (s, 1-19) are algebraically added to the contents of A (s, 1-19). The result is placed in A (s, 1-19). The contents of Y are not changed.

SUB Y SUBTRACT

The contents of Y (s, 1-19) are algebraically subtracted from the contents of A (s, 1-19). The result is placed in A (s, 1-19). The contents of Y are not changed.

MPY Y MULTIPLY

The overflow indicator is turned OFF. The contents of Y (s, 1-19) are algebraically multiplied by the contents of Q (s, 1-19). The result is placed in A (s, 1-19) and Q (s, 1-19), the sign of Q is the same as the sign of A. If the contents of A are not set to zero before the MPY command is given, the contents of A will be added algebraically to the least significant half of the product. Thus, with proper scaling, it is possible to form the value AB plus C. If overflow occurs, the overflow indicator will be turned ON. If no overflow occurs, the overflow indicator will be left OFF after this command is executed.

DVD Y DIVIDE

The contents of A (s, 1-19) and Q (1-19) are algebraically divided by the contents of Y (s, 1-19). The quotient is placed in A (s, 1-19); the remainder will be in Q (s, 1-19). The sign of the remainder is the sign of the dividend. The overflow indicator will be turned OFF when execution of the DVD command is complete The magnitude of the divisor must be greater than the magnitude of the contents of A. If not, the overflow indicator will be turned ON and control will be transferred to the specified next instruction

DAD Y DOUBLE LENGTH ADD

If Y is even, the contents of Y (s, 1-19) and Y + 1 (1-19) are algebraically added to the contents of A (s, 1-19) and Q (1-19). If Y is odd, the contents of Y (s, 1-19)

and Y (1-19) are algebraically added to the contents of A (s, 1-19) and Q (1-19). The result is placed in A (s, 1-19) and Q (1-19). The sign of Q is set to agree with the sign of A. The contents of Y and Y + 1 are unchanged. If this instruction is automatically modified, the address after modification will determine the result as indicated above.

DSU Y DOUBLE LENGTH SUBTRACT

If Y is even, the contents of Y (s, 1-19) and Y + 1 (1-19) are algebraically subtracted from the contents of A (s, 1-19) and Q (1-19). If Y is odd, the contents of Y (s, 1-19) and Y (1-19) are algebraically subtracted from the contents of A (s, 1-19) and Q (1-19). The result is placed in A (s, 1-19) and Q (1-19). The sign of Q is set to agree with the sign of A The contents of Y and Y + 1 are unchanged If this instruction is automatically modified, the address after modification will determine the result as indicated above.

DLD Y DOUBLE LENGTH LOAD

If Y is even, the contents of Y (s, 1-19) and Y + 1 (s, 1-19) replace the contents of A (s, 1-19) and Q (s, 1-19). If Y is odd, the contents of Y (s, 1-19) replaces the contents of A (s, 1-19) and Q (s, 1-19). The contents of Y and Y + 1 are unchanged. If this instruction is automatically modified, the address after modification will determine the result as indicated above.

DST Y DOUBLE LENGTH STORE

If Y is even, the contents of A (s, 1-19) and Q (s, 1-19) replace the contents of Y (s, 1-19) and Y + 1 (s, 1-19). If Y is odd, the contents of Q (s, 1-19) replace the contents of Y (s, 1-19). The contents of A and Q are unchanged. If this instruction is automatically modified, the address after modification will determine the result as indicated above

MAQ MOVE A TO Q

The contents of A (s, 1-19) replace the contents of Q (s, 1-19). Zeros replace the contents of A (s, 1-19)

LDZ LOAD ZERO INTO A

The contents of A (s, 1-19) are replaced by 0's.

BOV BRANCH ON OVERFLOW

If the overflow indicator is ON, the indicator is turned OFF and the computer takes the next sequential instruction. If the overflow indicator is not ON, the computer skips the next instruction and executes the second sequential instruction.

BZE BRANCH ON ZERO

If the contents of A (s, 1-19) are zero, the computer takes the next sequential instruction. If the contents are not zero, the computer skips the next instruction and executes the second sequential instruction. The contents of A are unchanged by this instruction.

EXAMPLES

1. Add the number in storage in memory location 0129 to the number in storage in memory location 0257. Store the result in memory location 0257.

```
1000    LDA    0219
1001    ADD    0257      Add 2 numbers and store the sum.
1002    STA    0257
```

Integers located in memory locations 0129 and 0257 will be represented as binary numbers. Later material on arrangement of data will provide facility for conversion of binary coded decimal data to binary numbers. The load A command loads the number in memory location 0129 into the A register. The command in storage at 1001 will add the number in storage at 0257 to the contents of the A register. The addition of the number in 0129 to the number in 0257 produces exactly the same algebraic result as the addition of the same integers if they had been expressed as decimal numbers. The Store A command will replace the contents of the word in storage at location 0257 with the sum of the numbers in 0129 and 0257. The number stored at 0257 will be represented as a binary number. Later material on arrangement of data will provide facility for conversion of binary numbers to binary coded decimal numbers.

2. In the previous example, test the result of the addition to determine whether the result of the addition of the numbers in 0129 and 0257 exceeds the capacity of the A register. If the capacity of the A register is exceeded, transfer control to the command in storage at 3200.

```
1000    LDA    0129    }
1001    ADD    0257    }   Add 2 numbers
1002    BOV            }
1003    BRU    3200    }   Test for overflow and transfer
1004    STA    0257        Store the sum
```

If the results of the addition exceed the capacity of the A register, the overflow indicator will be turned on. The overflow indicator does not affect machine operation unless it is interrogated by the program. The command in storage at location 1002 will interrogate the overflow indicator. If the indicator is on, control will be transferred to the command in storage at 1003. If the indicator is not on, control will continue with the command in storage at 1004.

3. Write the necessary commands to reconstruct the sum as a double precision binary number, when overflow occurs. Hold the reconstructed double precision number in the A and Q registers and program a "halt".

```
3200    SRD    1       }
3201    CHS            }   Reconstruct as a double precision
3202    SRD    18      }   number in A & Q
3203    BRU    3203    }   HALT
3204    DST    0256    }   Store the double precision result and
3205    BRU    1005    }   continue.
```

The commands in storage at 3200, 3201 and 3202 will reconstruct the sum as a double precision binary number in the A and Q registers. The BRU command at 3203 will execute a continuous branch to itself and further processing will be suspended until there is intervention from the console. When further processing is desired the console operator must put the GE 225 in "manual" and cause a branch to the command at 3204. The command at 3204 will cause the double precision result to be stored and a transfer to the command at 1005 to continue processing.

4. One positive number, X, occupies two memory locations at 0130 and 0131. Another positive number, Y, also occupies two memory locations at 0258 and 0259. Add A to B and store the result at 0258 and 0259.

```
1000    DLD    0130    }
1001    DAD    0258    }   Add 2 numbers and store the sum
1002    DST    0258    }
```

The double length load command loads X into the A and Q registers. The command in storage at 1001 will add Y to X and the sum will remain in the A and Q registers. The double length store A command at 1002 will replace the contents of the word in storage at location 0258 and 0259 with the sum of X and Y.

5. Subtract the number stored in memory location 0257, Z, from the sum of X and Y. Before subtracting, replace the contents of memory location 0256 with zeros. If the capacity of the A and Q registers is exceeded after the addition of X and Y, transfer control to 3200. Assume that Z is a positive number.

```
998     LDZ                 }
999     STA     0256        }   Store zeros in 0256
1000    DLD     0130        }
1001    DAD     0258        }   Add 2 numbers
1002    BOV                 }
1003    BRU     3200        }   Test for overflow and transfer
1004    DSU     0256        }
1005    DST     0258        }   Subtract number from A and
                                store the difference.
```

The double length subtract command in memory location 1004 will subtract Z from the sum of X and Y, and the difference will remain in the A and Q registers.

Since the double length subtract command will "address" memory location 0256, the commands at 998 and 999 will "set" the contents of 0256 fo zero.

The test for overflow condition is programmed after the addition of X and Y, since if Z is a positive number, the capacity of A and Q will not be exceeded as a result of the double length subtract command at 1004. The result of X + Y - Z is stored in memory at locations 0258 and 0259.

6. Multiply the number in memory location 128, by the number in memory location 10. Store the result in memory location 0256 and 0257. Test for the overflow condition before and after the multiply command.

```
688     BOV                 }
689     BRU     3200        }   Test and transfer for overflow
690     LDA     0128        }
691     MAQ                 }   Load number and position
692     MPY     0010        }   multiplicand
693     DST     0256            Store product
```

The commands at location 688 and 689 interrogate the overflow indicator before the multiply command is executed to determine whether the overflow indicator has been previously "set". The number in memory location 0128 is loaded into the A register and moved to the Q register by the commands at locations 690 and 691. Also, the contents of the A register will be replaced with zeros by the command at location 691. The multiply command will multiply the contents of the A and Q registers by the number in memory location 0010. The product of the two numbers can exceed the capacity of the A and Q registers only when both numbers are negative and each represent a maximum negative value (-2^{19}). Therefore, there is not interrogation for the overflow condition after the MPY command. The product is stored at memory location 0256 and 0257 by the double length store command.

7. Divide the number in memory location 0257 by the number in memory location 10. Store the quotient in memory location 0128. Disregard the remainder. Test for the overflow condition after the divide command is executed.

```
785     BOV                 }
786     BRU     3200        }   Test and transfer for overflow
787     LDA     0257        }
788     MAQ                 }   Position quotient and divide
789     DVD     0010        }
790     BOV                 }
791     BRU     3200        }   Test and transfer to overflow
792     STA     0128            Store quotient
```

The dividend is loaded into the A register and moved to the Q register by the commands in memory locations 787 and 788. Also, the contents of the A register will be replaced with zeros by the command at location 788. The divide command will divide the contents of the A and Q registers by the number in memory location 0010. The magnitude of the divisor must be greater than the magnitude of the portion of the dividend in the A register. If not, the overflow indicator will be turned on and control will be transferred to the command in storage at 3200. The dividend is stored in memory location 0128 by the command in memory at 792. The remainder, if any, will appear in the Q register and will be disregarded.

Shift Left A	SLA
Shift Right A	SRA
Load A from Q	LAQ

SLA K SHIFT LEFT A

The contents of A (1-19) are shifted left K places. Vacated positions of A are filled with zeros. If a non-zero bit is shifted out of position 1, the overflow indicator will be turned ON, and the bit is lost. The sign of A is unchanged.

SRA K SHIFT RIGHT A

The contents of A (1-19) are shifted right K places. If A is plux, 0's are inserted in the vacated positions of A. If A is minus, 1's are inserted in the vacated positions of A. Bits shifted out of position 19 are lost. The sign of A is not changed.

LAQ LOAD A FROM Q

The contents of Q (s, 1-19) replace the contents of A (s, 1-19). The contents of Q are unchanged.

EXAMPLES

1. Move the binary point of the number in location 0128 one place to the left. (Multiply the binary number in memory location 0128 by 2.)

```
0600    LDA    0128   }
0601    SLA    0001   }   Load and shift left 1 binary position
0602    BOV           }
0603    BRU    3200   }   Test and transfer for overflow
```

The binary number in location 0128 is loaded into the A register by the command in storage at 0600. The shift of a binary number to the left will increase the magnitude of the number by a power of 2 for every place that the number is shifted to the left. If the number exceeds the capacity of the A register after the number is shifted the overflow indicator will be turned on and control will be transferred to the command in storage at 0603.

2. Move the binary point of the number in location 0129 one place to the right. (Divide the binary number in memory location 0129 by 2.) Assume any remainder is lost.

```
0600    LDA    0129   }
0601    SRA    0001   }   Load and shift right
```

The binary number in location 0129 is loaded into the A register by the command in storage at 0600. The shift of a binary number to the right will decrease the magnitude of the number by a power of 2 for every place that the number is shifted to the right. Division will not increase the magnitude of the number, and therefore, the overflow indicator will not be turned on.

3. Move the decimal point of the number in location 0128 one place to the left. (Multiply the number in location 0128 by 10). Assume that the number in 0128 represents an integer expressed as a binary number.

```
3999    DEC    10     00000  00000  00000  01010

0600    LDA    0128   }
0601    MAQ           }   Multiply by constant of 10
0602    MPY    3999   }
```

The binary number in location 0128 is loaded into the A register by the command in storage at 0600 and moved to the Q register by the command at 0601. The multiplication of a binary number by other than a power of 2 is accomplished by the multiply command and the binary equivalent of the multiplier. The multiply command at location 0602 "addresses" the binary equivalent of 10 at location 3999. The Generalized Assembly Program will provide facility for the creation of binary constants for use in the program. For example the DEC operation at 3999 will create the binary equivalent of 10. The product represents exactly the same result as the shift of a decimal integer one place to the left.

4. Move the decimal point of the number in location 0128 one place to the right. (Divide the number in location 0128 by 10.) Assume that the number in 0128 represents an integer expressed as a binary number. Assume any remainder is lost.

```
3999    DEC    10          00000    00000    00000    01010

0600    LDA    0128    }
0601    MAQ            }   Divide by constant of 10
0602    DIV    03999   }
```

The binary number in location 0128 is loaded into the A register by the command in storage at 0600 and moved to the Q register by the command at 0601. The division of a binary number by other than a power of 2 is accomplished by the divide command and the binary equivalent of the divisor. The divide command at location 0602 "addresses" the binary equivalent of 10 at location 3999. The quotient in the A register represents exactly the same result as the shift of a decimal integer one place to the right.

5. The number in location 0128 is an integer expressed as a binary number with 2 places to the right of the decimal point. The number in 0130 is an integer expressed as a binary number with 3 places to the right of the decimal point. Add the number in location 0128 to the number in location 0130 and store the sum of the numbers in location 0132.

```
3999    DEC    10          00000    00000    00000    01010

0600    LDA    0128    }
0601    MAQ            }   Multiply by constant of 10
0602    MPY    3999    }
0603    LAQ            }   Position number in A register
0604    ADD    0130    }
0605    STA    0132    }   Add 2 numbers and store the sum.
```

The commands in storage at 0600, 0601 and 0602 perform the equivalent function of the commands specified in example 3 above. After the decimal point of the number in location 0128 is moved one place to the left, the number is properly aligned for addition with the number in 0130. The LAQ command will move the number to the A register in preparation for the addition. The commands in storage at 0604 and 0605 will add the number in storage at 0130 to the contents of the A register and store the results in location 0132.

6. Assume that X represents rate and occupies memory location 0130. X is an integer (which has 3 places to the right of the decimal point) represented as a binary number. Assume that Y represents hours and occupies memory location 0131. Y is an integer (which has 1 place to the right of the decimal point) represented as a binary number. Multiply X by Y, round the product to the nearest cent and store the result in memory at location 0258.

```
3996    DEC    0      00000    00000    00000    00000
3997    DEC    50     00000    00000    00001    10010
3998    DEC    100    00000    00000    00011    00100
```

```
0600    LDA     0130    }
0601    MAQ             }   Multiply 2 numbers
0602    MPY     0131    }
0603    DAD     3996        Add round factor
0604    DVD     3998    }
0605    STA     0258    }   Divide by constant of 100 and store
                            the quotient.
```

X is loaded into the A register and moved to the Q register by the commands at 0600 and 0601. X is multiplied by Y by the multiply command at 0602. The product of X and Y, in the A and Q registers, is an integer with four places to the right of the decimal point which is represented as a binary number. The double length add command at location 0603 will add the binary equivalent of the round factor (a 50 expressed as a binary number) in memory storage at 3996 and 3997 to the product of X times Y in the A and Q registers. The Generalized Assembly Program will provide the facility for the creation of binary constants for use in the program. The rounded sum in the A and Q registers is expressed as a binary number, and is exactly equivalent to the algebraic addition of a decimal round factor to a decimal number. The divide command in memory location 0604 address the binary equivalent of 100 at location 3998. The quotient in the A register represents exactly the same result as the shift of the decimal integer two places to the right (reference example 4 above). The command in memory location 0605 will store the product of X times Y rounded to the nearest cent in memory location 0258.

CONVERSION OF NUMERIC BINARY CODED DECIMAL DATA TO BINARY FORM (FROM CARD INPUT)

Assume that a binary coded decimal number starts in card column 20 and ends in card column 26 and the card image has been read in beginning in memory location 0128. Write the "calling sequence" to convert the number to a binary number. Assume that the appropriate conversion routine is in storage beginning at location 3700. Branch to the command at 3000 if an error should be generated during the conversion routine.

0700	SPB	3700	1	Transfer to conversion subroutine
0701	DEC	0128		
0702	DEC	0020		Calling sequence
0703	DEC	0007		
0704	BRU	3000		

The SPB command in storage at 0700 will transfer control to the first command in the conversion subroutine in storage at 3700. The location of the SPB command is preserved in modification word 1. The operation of the DEC command is explained in the section on the General Assembly Program (GAP 225). The commands in storage at 0701, 0702 and 0703 will provide binary equivalents of the decimal integers 0128, 0020 and 0007. The utilization of the integers expressed as binary numbers is illustrated in example 2 in the section on Subroutine Programming.

If there is an error in the calling sequence specification, for example, an attempt to convert a (binary coded) decimal field of more than 11 digits, control will be transferred to the command in storage at 0704. After a number is converted, control will be transferred to the command in storage at 0705.

When the conversion is accomplished, the converted number will appear in the A and Q registers. The least significant word of the 2 word binary number will appear in the Q register; the most significant word will appear in the A register. The conversion routine will properly interrogate the least significant character position of the card field and produce a negative binary number if this position contains the appropriate overpunch (11 punch in Hollerith code).

CONVERSION OF BINARY DATA TO NUMERIC BINARY CODED DECIMAL FORM (FOR CARD OUTPUT)

Assume that an integer is expressed as a binary number contained within the A and Q registers and that it will consist of no more than 9 digits when converted to a (binary coded) decimal number. Write the "calling sequence" to convert the number and store the binary coded decimal number in a 9 digit field ending in card column 25. Assume that the card image begins in memory storage at location 0128 and that the appropriate conversion routine is in memory storage at location 3300. Branch to the command at 3100 if an error should be generated during the conversion routine.

```
0640    SPB    3300        Transfer to conversion subroutine
0641    DEC    0128   }
0642    DEC    0025   }    Calling sequence
0643    DEC    0009   }
0644    BRU    3100   }
```

The SPB command in storage at 0640 will transfer control to the first command in the conversion subroutine in storage at 3300. The location of the SPB command is preserved in modification word 1. The operation of the DEC command is explained in the section on the General Assembly Program (GAP 225). The commands in storage at 0641, 0642 and 0643 will provide binary equivalents of the decimal integers 0128, 0025 and 0009. The utilization of the integers expressed as binary numbers is illustrated in example 2 in the section on Subroutine Programming.

If there is an error in the calling sequence specification, for example, an attempt to store the converted number in a field greater than 11 digits, control will be transferred to the command in storage at 0644. After a number is converted and contained within the card image, control will be transferred to the command in storage at 0645.

When the conversion is accomplished and converted number will appear in columns 17 - 25 within the card image in storage. The conversion routine will properly interrogate the sign position of the binary number (sign of the A register) and, if the number is negative, produce the equivalent of an overpunch (11 punch in Hollerith code) over the least significant character position of the card field (column 26 in this example.

PROGRAMMING LOGICAL DECISIONS

Perhaps the most important feature of a computer is its ability to make logical decisions. Naturally, these decisions can be made only through stored programs which utilize test and branch commands in the proper sequence. Frequently, decisions which determine the paths of processing within a routine are based on the results from tests on the data being processed itself. Thus, input records may be categorized by some identification key or factor and processed accordingly.

When large volumns of records are involved, it is usually more efficient to prearrange them prior to processing. This prearrangement involves the ordering (sorting) of records by some key. A key is a portion of the record which is set aside to identify it uniquely from other records or to categorize it with other records, according to the nature of the data processing problem. A key is made up of a variable number of numeric, alphabetic or alphanumeric characters; keys may range in length from 2 or 3 digits to 20 or more. When data pertaining to a particular subject (account #, employee #, or catalog #, etc.) is split up into several records, each carried in a separate file, it is necessary to "match up" the records on the key before any processing can be done. A common example is the matching of input transaction records against records in a master file prior to the posting and updating of information in the master file.

The sequencing and matching of records in accomplished through the comparison of appropriate keys. In order to match keys or to arrange them in some standard understandable sequence, it is clearly necessary to have some system of ranking of all characters in the "alphabet" of the computer language. Thus, a system must be established similar to that employed by dictionaries and encyclopedias by which the letter A is given lowest rank and the letter Z highest. If the reader will consider the octal (or equivalent binary) value for each computer character in BCD mode (see Character Representation in Appendix), he will see that viewing each character as a six bit binary number accomplishes an automatic ranking. 0 (zero) has the lowest rank and] has the highest. The numbers 0-9 are lower than any other characters; the alphabetics A-Z are arranged in their normal sequence; and the special characters are not ordered in any particular way. Further investigation will show that if any combination of characters is viewed as a pure binary number and compared to any other combination of characters viewed in the same way, the combination with highest numerical value will be listed <u>following</u> the other combination if one proceeds to order them in "dictionary sequence" (examining them character by character from left to right). In this sense one may say that a key (whether numeric, alphabetic or alphanumeric) of a given record is either equal to, higher than or lower than the key of another record.

In the GE 225 keys are compared in the following manner. One key is loaded into the A or A and Q registers utilizing the LDA command or the DLD command. The second key is subtracted from the first key by the SUB command or the DSU command, and the difference between the two keys remains in the A or A and Q registers. If the two keys are exactly equal, the contents of the A register or A and Q registers will be equal to zero. Therefore, the BNZ or BZE commands will be ultized to interrogate the contents of the registers for zero. However, the BNZ and BZE commands interrogate the A register only. If a test of the Q register is required because of a double word length key, the XAQ command or LAQ command will load A with the contents of the Q register. The BZE or BNZ commands can then be repeated to test the status of the second key. Control will either transfer to the next command, or the command after the next command, depending upon whether the contents of the A register is equal to zero.

If the contents of the A register is positive (plus), the first key loaded must be higher than the second key. A test for the zero condition should precede the test for a plus condition since a word of zeros would also be positive. The A register may be interrogated for the plus condition by the BPL command. Control will either transfer to the next command, or the command after the next command, depending upon whether the contents of the A register is positive (plus) or negative (minus). If the contents of the registers is neither positive nor equal to zero, the contents must be negative. If control does not transfer as a result of the zero or plus test, the absolute value of the second key must be greater than the absolute value of the first key. The registers may be interrogated directly for "minus" by the BMI command, or the minus condition may be assumed if control does not transfer after a BZE and BPL command.

It is important to note that the keys for matching do <u>not</u> have to be converted from BCD to binary numbers before the LDA, SUB, BZE, BPL, BMI sequence is executed. Even though the correct algebraic result will not be obtained from the subtraction, the correct relative conditions of equality (zero), high (plus) or low (minus) will be properly generated. Therefore, keys do not have to be converted before logical decisions are made.

One of the distinguishing features which take stored program computers out of the class of desk calculators and punched card equipment is the power to perform calculations on the instructions which govern the machine's functions. If there is some relationship between memory addresses and their contents, it is thus possible to compute the operand addresses of instructions in a program so that data may be moved about, to and from memory storage, without the use of the standard test and branch commands described

in the previous paragraphs for routing of the information. This is a particularly powerful method for table lookup and table posting operations. If a relationship does exist between memory addresses and their contents, one can use mathematical terminology and say that the address is a "function" of the contents. For this reason the term "function table technique" is often applied to such programming usage.

The STO command is the chief tool used for this purpose. The command permits data to be inserted directly into the operand address portion of an instruction so that the data value itself will determine the address selected by the instruction. Since there is no restriction on the type of instruction that may be thus modified, the data may determine memory locations from which other data is extracted or into which other data is stored, or it may determine the addresses of other parts of the program to which control is transferred.

TRANSFER OF CONTROL BASED UPON LOGICAL COMPARISONS OF DATA WITHIN THE 225

Branch Unconditionally	BRU
Branch on Plus	BPL
Branch on Minus	BMI
Subtract One	SBO
Branch on No Zero	BNZ

BRU Y BRANCH UNCONDITIONALLY

Control is transferred to the instruction located at Y.

BPL BRANCH ON PLUS

If the sign of A is plus, the computer takes the next sequential instruction. If the sign of A is not plus, the computer skips the next instruction and executes the second sequential instruction. The contents of A are unchanged by this instruction.

BMI BRANCH ON MINUS

If the sign of A is minus, the computer takes the next sequential instruction. If the sign of A is not minus, the computer skips the next instruction and executes the second sequential instruction. The contents of A are unchanged by this instruction.

SBO SUBTRACT ONE

One is subtracted algebraically from A (19). If the capacity of A is exceeded, the overflow indicator will be turned ON.

BNZ BRANCH ON NO ZERO

If the contents of A (s, 1-19) are not zero, the computer takes the next sequential instruction. If the contents are zero, the computer skips the next instruction and executes the second sequential instruction.

EXAMPLES

1. Compare the word in memory location 0128 with the word in memory location 0256. If both words are equal transfer control to memory location 1500.

```
1000    LDA    0128  }
1001    SUB    0256  }   Subtract in A register.
1002    BZE          }
1003    BRU    1500  }   Test A register and transfer for equal.
```

If the word in memory at location 0128 is equal to the word in memory at location 0256, the contents of the A register would be equal to zero, and computer control would continue with the command in memory at location 1003. If the contents of 0128 is not equal to the contents of 0256, control would transfer to the command in memory at location 1004.

2. In the previous example, if the word in memory at location 0128 is higher than the word in memory at 0256, transfer control to the command at location 3000. If the word in 0128 is lower, transfer control to the command at location 3500.

```
1000    LDA    0128  }
1001    SUB    0256  }   Subtract in A register.
1002    BZE          }
1003    BRU    1500  }   Test and transfer for equal.
1004    BPL          }
1005    BRU    3000  }   Test and transfer for high.
1006    BRU    3500  }   Transfer for low.
```

If the word in memory at location 0128 is higher than the word in memory at location 0256, the contents of the A register would be plus and control would continue with the command in memory at location 1005. If the word in memory at location 0128 is lower than the word in memory at location 0256 the contents of the A register would be minus and control would be transferred to the command in memory at location 1006.

3. If the punch in column 1 of an input card is a 1, transfer control to a routine starting at memory location 3000; if the punch is a 2, transfer control to memory location 3050; if it is a 3, transfer to 3075. If the punch in card column 1 is neither a 1, 2 or 3, continue control with the next command in sequence. Assume the card has been read into memory beginning with the word at 0128.

```
1500    LDA    0128  }
1501    SRA    0012  }   Position first card column in A register.
1502    SBO          }
1503    BZE          }   Test and transfer for 1.
1504    BRU    3000  }
1505    SBO          }
1506    BZE          }   Test and transfer for 2.
1507    BRU    3050  }
1508    SBO          }
1509    BZE          }   Test and transfer for 3.
1510    BRU    3075  }
```

The contents of memory location 0128 contain equivalent binary coded decimal representation of the Hollerith code punching contained in columns 1, 2 and 3 of the 80-column card. The command in memory at location 1500 loads the A register with the contents of memory location 0128. The shift right A command deletes the data from card columns 2 and 3 from the A register and shifts the binany coded decimal data from card column 1 into the "low order positions" of the A register. (Bit positions 14-19). Computer control will transfer during the commands which follow if reduction of the contents of the A register by 1 causes a zero condition for any of 3 tests. If control does not transfer, then the digit in card column 1 is neither a 1, 2 or 3, and computer control will continue with the command in memory at location 1511.

4. One master file record consisting of 12 words of data is in storage beginning at location 0256. One input file transaction record consisting of 8 words of data is in storage beginning at location 0128. The first 3 words of each record contain the identification, or key, of the record, the "keys" in both files are in ascending sequence. If all three words of each key are identical, transfer control to the next available storage location to update the master file record. If the input transaction item key is higher than the master file item key, transfer control to the command in storage at 3100 to output the master file record onto the new updated master file. If the key on the transaction record is lower than the master file key, transfer control to the command at 3200 to create and insert a new master file record into the file.

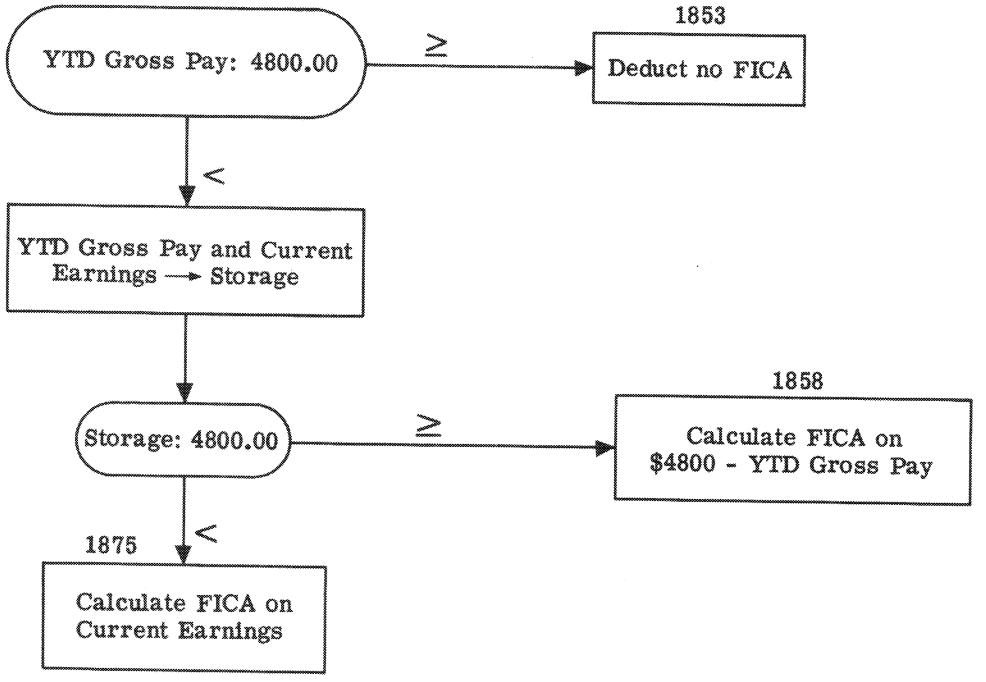
```
0598    LDA    0128  }
0599    SUB    0256  }
0600    BZE          }
0601    BRU    0605  }   Test and transfer for first word.
0602    BMI          }
0603    BRU    3200  }
0604    BRU    3100  }
0605    LDA    0129  }
0606    SUB    0257  }   Test and transfer for second word.
0607    BZE          }
0608    BRU    0610  }
0609    BRU    0602  }   Transfer for not equal.
```

```
0610    LDA    0130
0611    SUB    0258
0612    BNZ
0613    BRU    0602
```

Test and transfer for third word.

The commands in storage at locations 0598 through 0601 will test the first word of the master file and transaction keys for equality. If the first words are equal, the contents of the A register will be zero; and control will be transferred to 0605 where successive comparisons are made on the remaining words of the key, progressing from left to right. At any point where the keys are discovered to be unequal, control will be transferred to 0602. The commands in storage at 0602 through 0604 determine which key is larger and route control of the program accordingly. If all three words of the keys check for equality (i.e., the keys are identical) control passes to location 0614 where proper posting of the transaction occurs.

5. This example involves the decisions which are required in the calculation of Social Security (FICA) tax for a payroll. Assume that payroll data is in storage beginning at location 0128. Year-to-year gross pay occupies locations 0130 and 0131. Current weeks' earnings occupy locations 0132 and 0133. If year-to-date gross pay is equal to or greater than $4800, transfer control to the command in storage at 1853 where no additional FICA will be deducted. If year-to-date gross pay is less than $4800, determine whether year-to-date gross pay plus current earnings is equal to or greater than $4800. If year-to-date plus current earnings is equal to or greater than $4800, transfer control to 1858 where the FICA tax will be calculated on $4800 minus year-to-date earnings (not including the current week). If year-to-date plus current earnings is less than $4800, transfer control to 1875 where the FICA tax will be calculated for the total amount of current weeks' earnings.




```
YTD Gross Pay                          0130-0131
Current (Week's) Earnings              0132-0133

3998        DEC      0
3999        DEC      480000
```

(The section of the General Assembly Program in the Programming Manual will show how these two pseudo-commands will result in the storage of a binary zero in 3998 and the binary equivalent of 480000 in 3999.)

```
0431    DLD    0130 |
0432    DSU    3998 |   Load YTD gross pay and subtract 4800.00
0433    BPL         |
0434    BRU    1853 |   Test and transfer for no FICA.
0435    DAD    0132 |   Add current earnings.
0436    BPL         |
0437    BRU    1858 |   Test and transfer for all earnings taxable.
0438    BRU    1875 |   Transfer for part earnings table.
```

The commands in storage at 0431, 0432, 0433 and 0434 will compare the YTD earnings with the constant 480000 and transfer control to 1853 if YTD earnings are greater than or equal to this amount. The DAD command will increment the amount in registers A and Q by current weeks' earnings. Testing register A at this point for a positive sign is the logical equivalent of comparing YTD gross pay plus current weeks' earnings against the constant 480000 to see if total earnings (including this week) are equal to or larger than $4800.00. If so, control transfers to 1858; if not, control transfers to 1875 where the FICA tax is calculated on all of this weeks' earnings.

	Store Operand Address		STO
STO	Y	STORE OPERAND ADDRESS	

The contents of A (7-19) replace the contents of Y (7-19). The contents of A and Y (s, 1-6) are unchanged.

EXAMPLES

1. As successive employee input records are read into memory, the current week's gross earnings for each employee record will appear in storage at 0132 and 0133. The employee departmental charge number will appear in each employee input record in storage at 0134. It is desired that current week's gross earnings be summarized by departmental charge number in a table which originates at memory location 0900 and extends to 1099. Departmental charge numbers are represented by the numbers 100-199. Thus, charges to number 100 should be accumulated in location 900-901, etc., allowing two locations in the table for each departmental charge number. (This may be accomplished by increasing the departmental charge number to the appropriate value in the 0900 to 1099 range and then storing this "calculated address" in the operand portion of the instructions which perform the calculations.) Write the necessary commands to accumulate and distribute the gross earnings for each employee input record to the correct departmental charge number at the appropriate location.

```
3998    DEC   (  2      00000   00000   00000   00010)
3999    DEC   ( 700      00000   00000   10010   11000)
```

(Explained in General Assembly Program section; results in storage of binary equivalents of 2 and 700.)

```
0474    LDA   0134  }  Load departmental charge number into A.
0475    MAQ         |
0476    MPY   3998  |  Multiply charge number by 2.
0477    LAQ         |
0478    ADD   3999  |  Add 700 to product.
0479    STO   0482  |
0480    STO   0483  |  Insert address into commands at 482 and 483.
0481    DLD   0132  }  Load current gross earnings.
0482    DAD   0000  |
0483    DST   0000  |  Add previous gross and store.
```

Since the location of the departmental accumulated charges is a function of the departmental charge number, the location is calculated by the direct use of the departmental charge number itself. The departmental charge number is loaded into register A by the command at 0474. The departmental charge number is then multiplied by 2 since the departmental charge accumulations will each occupy 2 locations of storage. After multiplying by 2, it is necessary to increase the departmental charge number by 700 to get the proper value in the 0900 to 1099 range. This is accomplished by the instructions at 0477 and 0488. The STO commands is storage at 0479 and 0480 will insert the calculated departmental charge number location into the operand portion of the commands at 0482 and 0483. The commands in storage at 0481, 0482 and 0483 will now add current week's gross earnings to appropriate departmental accumulated charges.

PROGRAMMING SWITCHES

	Exchange A and Q	XAQ
XAQ	EXCHANGE A AND Q	

The contents of A (s, 1-19) and Q (s, 1-19) are interchanged.

EXAMPLES

1. Transaction cards are being read into memory storage and processed. The first card is to be read into 0128-0154, the second card into 0256-0282, the third into 0128-0154, the fourth into 0256-0282, etc. Provide for this alternation of input area by means of a programmed "flip-flop".

```
3998    RCD    0256  }
3999    RCD    0128  }   Constant for RCD command.

0150    DLD    3998  }   Load both commands
0151    XAQ          }
0152    DST    3998  }   Exchange and store for next use.
0153    STA    0156  }   Store appropriate RCD command.
0154    BCN          }
0155    BRU    0154  }   Delay for reader ready.
0156    RCD    0000  }   Appropriate RCD command.
0157    HCR
0158    Continue
        processing
  ↓       ↓
0367    BRU    0150
```

The DLD command at 0150 will load both RCD commands into the A and Q registers. The XAQ command will exchange the contents of the A and Q registers, and the DST command will store the now exchanged RCD commands back into position for the next alternation. The STA command at 0153 will store the appropriate RCD command in position to be executed. The chosen RCD command will be executed when the card reader is ready. (See material on Card Input-Output Programming.) Control will be returned to read the next card by the command in storage at 0367.

2. In the above example assume that alternate cards require entirely different processing and, hence, entirely different programs. Write a control routine utilizing a "program switch".

```
0050    BRU    0400  }   BRU command "constants".
0051    BRU    0300  }

0100    DLD    0050  }   Load both commands.
0101    XAQ          }
0102    DST    0050  }   Exchange and store for next use.
0103    STA    0104  }   Store appropriate BRU command.
0104    BRU    0000  }   Appropriate BRU command.

0300    BCN          }
0301    BRU    0300  }
0302    RCD    0128  }
0303    HCR          }   Routine for card read into 0128.
0304    process      }
  ↓                  }
0350    BRU    0100  }
```

```
0400      BCN
0401      BRU     0400
0402      RCD     0256
0403      HCR
0404      process
  ↓
0496      BRU     0100
```

Routine for card read into 0256.

The commands in storage at 0100, 0101, 0102 and 0103 are equivalent to the commands in storage at 0105, 0151, 0152 and 0153 in the previous example. In this example the "flip-flop" consists of alternation of branch commands.

MODIFICATION WORD PROGRAMMING

The contents of a modification word may be programmed to automatically increment the "operand address" portion of an instruction each time the instruction is executed. The words in memory storage at 0001, 0002 and 0003 perform the functions of modification words. These functions include testing and tallying for control of iterative program loops as well as simple address modification.

When using these modification words, it is necessary to "initialize" their setting, which usually means replacing their contents with zeros, and to periodically increment the contents of the modification words themselves. The contents of a modification word may be most conveniently replaced with zeros by the LDZ command followed by the STA command. The operand address of the STA command will specify the modification word to be cleared to zeros. The modification words can be incremented directly by the INX command. The INX command will increment a modification word by the amount contained in the "operand address" portion of this instruction. The amount may by specified as a negative number to permit decrements.

It will often by necessary to test the contents of a modification word to determine the path for program control to follow. The contents of a modification word may be tested most directly by the BXH and BXL commands. Control will be transferred to the next instruction in sequence if the contents of the modification word are equal to or higher than the test amount in the case of the BXH command or lower than the test amount in the case of the BXL command. The BXH commands are utilized for the testing of modification words only.

Certain commands are not address modified. An example is the INX command. The operand address portion of this command is used to contain the amount by which the contents of a modification word is to be incremented. The programmer designates the modification word he wishes to increment by entering a 1, 2 or 3 into that portion of the instruction normally used to indicate automatic address modification. For similar reasons, the BXH and BXL commands are not automatically address modified.

The shift commands may be automatically address modified. However, the "operand address" of the shift commands consists of a binary number (length of shift) equivalent to 31 or less. Therefore, the programmer should be aware that any "address modification" must not produce a length of shift in excess of 31 positions. Certain data transfer commands (MAQ, LQA, XAQ, etc.) should not be address modified because the "operand address" of these commands are not to be utilized by the programmer. Commands for testing (BPL, BMI, BOV, etc.) should not be automatically address modified for similar reasons.

USE OF THE 225 MODIFICATION WORDS FOR TALLIES, TESTS AND AUTOMATIC ADDRESS MODIFICATION

Increment Modification Word by K	INX
Branch if Modification Word is Low	BXL
Branch on No Zero	BNZ

INX K,X INCREMENT X BY K

K, positions 7 through 19 of the I register are added absolutely to the contents of X (7-19), and the result replaces the contents of X (7-19). A carry from position 7 of X is lost. This instruction is not automatically modified.

BXL K,X BRANCH IF X IS LOW

If the contents of X (7-19) are less than K, the computer takes the next sequential instruction; if the contents of X (7-19) are greater than or equal to K, the computer skips the next instruction and executes the second sequential instruction. The contents of X are not changed. This instruction is not automatically modified.

BNZ BRANCH ON NO ZERO

If the contents of A (s,1-19) are not zero, the computer takes the next sequential instruction. If the contents are zero, the computer skips the next instruction and executes the second sequential instruction. The contents of A are unchanged by this instruction.

EXAMPLES

1. Data punched into a card has been read in BCD mode into memory storage locations 0128-0154. Transfer the information from the card input area to memory storage locations 0256-0282. Do not use address modification.

```
0700    DLD    0128
0701    DST    0256
0702    DLD    0130
0703    DST    0258
0704    DLD    0132
0705    DST    0260
0706    DLD    0134
0707    DST    0262
0708    DLD    0136
0709    DST    0264
0710    DLD    0138
0711    DST    0266
0712    DLD    0140
0713    DST    0268
0714    DLD    0142
0715    DST    0270
0716    DLD    0144
0717    DST    0272
0718    DLD    0146
0719    DST    0274
0720    DLD    0148
0721    DST    0276
0722    DLD    0150
0723    DST    0278
0724    DLD    0152
0725    DST    0280
0726    LDA    0154
0727    STA    0282
```

Straight-line Programming

The 13 double length load and double length store commands in memory storage locations 0700-0725 will transfer 26 words of information to the output storage area; the 27th word is transferred by a single length load and store command.

2. In the previous example, write the program steps required using programmed address modification.

```
3996    DEC    0     }   00000   00000   00000   00000
3997    DEC    2     }   00000   00000   00000   00010
3998    DEC    1     }   00000   00000   00000   00001
3999    DEC    13    }   00000   00000   00000   01101
```

(Explained in General Assembly Program Section; results in storage of binary equivalents of 0, 2, 1, and 13.)

```
0700    DLD    0128   }
0701    DST    0256   }   Move 2 Words of Data
0702    LDA    0700   }
0703    ADD    3997   }   Modify Location 0700
0704    STA    0700   }
0705    LDA    0701   }
0706    ADD    3997   }   Modify Location 0701
0707    STA    0701   }
0708    LDA    3996   }
0709    ADD    3998   }   Increment Tally
0710    STA    3996   }
0711    SUB    3999   }
0712    BNZ           }   Test the Tally
0713    BRU    0700       Go Back to Move 2 Words
0714    LDA    0154
0715    STA    0282
```

Two words of data are transferred by the commands in storage at 0700 and 0701. The commands in storage at 0702-0704 will increment the operand or "address" of the double length load command at 0700. The double length load command will now "address" the next two words to be loaded. The commands in storage at 0705-0707 will increment the operand "address" of the double length store command at 0701. The double length store command will now "address" the next two words of storage. The commands in storage at 0708-0710 will increment a "count" in storage at 3996. When the "count" in storage at 3996 is not equal to 13, control is transferred back to location 0700. Thus, two more words will be transferred, and the programmed address modification will be continued. When 26 words have been transferred, the count will be equal to 13, and control will be continued with the command in memory at 0714.

3. Write the program steps required in examples 1 and 2 above using the 225 modification words for automatic address modification, tallying and testing.

```
0700    LDZ
0701    STA    0001
0702    DLD    0128    1   }   Move 2 Words of Data and Automatic
0703    DST    0256    1   }   Address Modification
0704    INX    2       1       Increment Tally
0705    BXL    26      1       Test Tally
0706    BRU    0702            Go Back to Move 2 Words
0707    LDA    0154
0708    STA    0282
```

The commands in storage at 0700 and 0701 replace the contents of the word at 0001 with zeros. The commands in storage at 0702 and 0703 will transfer two words of data. Before either the command at 0702 or 0703 is executed the contents of the

modification word will automatically increment the command itself. There are 3 memory locations which are available in the 225 for automatic address modification. The 3 memory locations available are 0001, 0002 and 0003.

When a command in storage indicates a 1, 2 or 3 in the proper position of the format, the contents of either word 0001, 0002 or 0003 will automatically increment the command before it is executed. Automatic address modification is performed when the contents of memory words 0001, 0002 or 0003 increment the command. The contents of memory location 0001 automatically increment the commands at locations 0702 and 0703 because a "1" is inserted into the instruction format in the proper position. A tally or "count" is performed when the command in storage at 0704 adds directly to the modification word in storage at 0001. The "1" in the proper position indicates addition to the modification word at memory location 0001. The "2" indicates that a 2 is to be added to the modification word in memory location 0001 each time the INX instruction is executed. Testing is performed when the contents of the modification word at 0001 are compared to a "26". When the contents of the modification word are equal to 26, 26 words have been transferred, and control will be transferred to the command in storage at 0707. As long as the contents of the modification word at 0001 are not equal to 26, 26 words have not been transferred, and control will be transferred back to the command in storage at 0702.

4. Code example 4 of the Programming Logical Decisions section, Transfer of Control Based upon Logical Comparisons of Data within the 225, making use of address modification. A master file record consisting of 12 words of data is in storage beginning at location 0256. An input transaction record consisting of 8 words of data is in storage beginning at location 0128. The first 3 words of each record contain the identification, or key, of the record, and both files are in ascending sequence. If all three words of each key are identical, transfer control to the next available storage location to update the master file record. If the input transaction item key is higher than the master file item key, transfer control to the command in storage at 3100 to output the master file record onto the new updated master file. If the key on the transaction record is lower than the master file key, transfer control to the command at 3200 to create and insert a new master file record into the file.

```
0598    LDZ                          }
0599    STA     0001                 }   Store Zeros in Word 1
0600    LDA     0128    1            }
0601    SUB     0256    1            }   Test for Equality
0602    BZE                          }
0603    BRU     0607                 }
0604    BMI                          }   Test and Transfer for Transaction Key
0605    BRU     3200                 }   Lower.
0606    BRU     3100                     Transfer for transaction key higher.
0607    INX     1       1            }
0608    BXL     3       1            }   Test for all 3 words equal.
0609    BRU     0600                     Transfer to repeat again.
```

The commands in storage at 0598 and 0599 will replace the contents of modification word 1 with zeros The commands in storage at 0600 through 0603 will test each word of the master file and transaction keys, starting with the major key and ending with the minor key, for equality. If the keys are equal, the contents of the A register will be zero and control will be transferred to the command at 0607 to continue the comparison. If all 3 words of the key are equal, control will be transferred to the command at the next available storage location at 0610 by the INX and BXL commands at 0607 and 0608 If the transaction key is higher than the master file key, control will be transferred to the command in storage at 3100. If the transaction key is lower than the master file key, control will be transferred to the command in storage at 3200.

PROGRAMMING FOR SUBROUTINE USAGE

In writing a program, it is often necessary to use on several different occasions a particular set of instructions which perform a specific function. Considerable saving of memory space and programming time will result if it is possible to transfer control from any point in the routine to execute this set of instructions whenever they are required and then jump back to the correct place in the main routine. "Subroutine" is the term applied to such a series of commands designed to perform a repetitive function for the main program.

Programming for the use of subroutines presents the programmer with the opportunity to employ the "building block principle" for the construction of programs. All frequently used data processing functions at an installation are prepared in subroutine form. It is then only necessary for the programmer to prepare the skeletal structure of the main program which provides the mortar for these building blocks. Subroutines for the conversion of binary coded decimal data to binary form and for the conversion of binary data to binary coded decimal form have already been encountered. These subroutines will be combined with the programmer's own coding by the General Assembly Program. Any time a data conversion is required in his own routine, the programmer will simply transfer control to the desired subroutine in the prescribed manner.

It has already been observed that it is necessary to be able to jump to a subroutine from any point in memory and to return there. Thus, along with the transfer of control information must also be retained so the subroutine will know where to go when it is finished. This idea of informing the subroutine how to get back has led to the name "linkage". The SPB command is designed to provide the "link" for the return of control back to the main program after the subroutine function is performed. The memory location of the SPB command will be saved within a modification word. The subroutine will execute a proper return to the main program by reference to the contents of this modification word. The INX command can be used to increment the contents of the modification word; and a BRU command, automatically address modified by the incremented modification word, will transfer control back to the proper next instruction in the main program. Through such "linkage" it is possible to utilize the subroutine many times without repetition of the subroutine each time it is required within the main program.

In addition to linkage, it is also necessary to specify the parameters which define the problem to the subroutine; that is, subroutines are usually written in a form for general applicability and must be self-specializing to the particular problem at hand. For example, the programmer must indicate to one of the general conversion subroutines the length of field information and the location of the data in memory by means of DEC commands following the SPB command. Since the location of the SPB command is saved within a modification word, the following DEC command information by be "called in" by the subroutine program. The "calling sequence" technique is to write a linkage followed by a few words which contain the parameter information.

In summary, it should be clear that subroutines make possible considerable saving of memory space and programming time at the very slight expense of the space and complexity of linkages and calling sequences.

		STORE P AND BRANCH	SPB
SPB	Y,X	STORE P AND BRANCH	

The location of this instruction replaces the contents of x (7-19), and control is transferred to the instruction located at Y. This instruction is not automatically modified.

EXAMPLES

1. Assume that a subroutine is in memory beginning at location 2000. This subroutine will transfer 28 words of data from memory storage beginning at 0128 to memory storage beginning at 0256. (See example 3 under section "Modification Word Programming".) Transfer control to the instruction at location 2000 to execute this subroutine and at the same time preserve the location of the command which initiates the transfer; that is, transfer control to 2000 and establish a link back to the main program.

```
0100    SPB     2000    1
```

The SPB command will transfer control to the instruction in storage at 2000. In addition, the memory location of the SPB command (location 0100) will be stored within modification word 0001.

2. In the previous example include the necessary instruction steps to return to the main program after the subroutine is executed.

```
0100    SPB     2000    1       Transfer to subroutine

2000    LDZ
2001    STA     0002
2002    DLD     0128    2
2003    DST     0256    2       Subroutine to move 28 words
2004    INX     2       2       in storage.
2005    BXL     28      2
2006    BRU     2002
2007    BRU     0001    1       Transfer back to main program at 0101.
```

Modification word 1 will contain the location in memory of the SPB instruction (0100) after this instruction is executed. The subroutine program steps in memory at locations 2000 through 2006 accomplish the transfer of the 28 words. The last instruction of the subroutine, at memory location 2007, will transfer control back to the main program at location 0101. This occurs because the contents of modification word 0001 will automatically increment the operand of the BRU command before it is executed.

3. In the example above, assume that the subroutine in memory beginning at location 2000 is a "skeleton" routine. Provide the necessary "calling sequences" in the main program to adapt the subroutine to the desired job.

```
0100    SPB     2000    1       Branch to main program.
0101    DEC     0128
0102    DEC     0256            Calling Sequence.
0103    DEC     28
```

```
2000    INX     1       1
2001    LDA     0000    1
2002    STO     2011
2003    INX     1       1
2004    LDA     0000    1
2005    STO     2012
2006    INX     1       1
2007    LDA     0000    1
2008    STO     2014
2009    LDZ
2010    STA     0002
2011    DLD     0000    2
2012    DST     0000    2
2013    INX     2       2
2014    BXL     0000    2
2015    BRU     2011
2016    BRU     0001    1
```

Commands to "call in" the necessary parameters before the subroutine is executed.

Subroutine to move 28 words of storage.

Transfer back to main program.

The operation of the DEC command is explained in the section on General Assembly Program. The commands in storage at 0101, 0102 and 0103 will provide binary equivalents of the decimal integers 0128, 0256 and 0028. The instructions in the subroutine at locations 2000 through 2008 will store the binary equivalents of 0128, 0256 and 0028 in the operand portions of the instructions at 2011, 2012 and 2014. The INX commands facilitate the operation by picking up the successive memory locations (where the constants 0128, 0256 and 0028 are stored) using the address modification word technique. Thus, for example, the contents of modification word 1 will become the operand of the LDA command at 2001. (See the material on Modification Word Programming.) After the LDA command is executed the binary equivalent of 0128 will be in the A register. The STO command will replace the operand of the instruction at 2011 with the binary equivalent of 0128. The net result after the command at 2008 is executed is to produce a subroutine identical to that in example 2, above.

4. In the example above, design the "call in" commands without using the INX commands.

```
0100    SPB     2000    1
0101    DEC     0128
0102    DEC     0256
0103    DEC     28

2000    LDA     0001    1
2001    STO     2011
2002    LDA     0002    1
2003    STO     2012
2004    LDA     0003    1
2005    STO     2014
2006    LDZ
2007    STA     0002
2008    DLD     0000    2
2009    DST     0000    2
2010    INX     2       2
2011    BXL     0000    2
2012    BRU     2008
2013    BRU     0004    2
```

Branch to main program.

Calling sequence.

Commands to "call in" the necessary parameters before the subroutine is executed.

Subroutine to move 28 words of storage.

Transfer back to main program.

This example is similar to example 3; however, the example above is more efficient and requires less time to execute. In the example above, the INX commands are eliminated because the LDA commands contain the necessary increment in their operand addresses.

Figure 20 Control Console

PROGRAMMING FOR CONSOLE CONTROL

During the running of a program it will often be desirable to provide information to the console operator. For example, if an overflow condition occurs during the processing of an input transaction, the programmer may desire that the console typewriter type the message "Overflow condition for record XXXX". The console operator then may take remedial action from the console.

There are 20 switches on the console which can be used to enter information directly into register A. These switches correspond to the 20 bit positions of a word. By proper programming, the switches may also have the effect of 20 program control switches. For instance, in the above example the program might have been prepared to transfer control to a portion of the routine that would skip the record which generated the overflow condition and continue processing if the operator had moved console option switch 10 to the "down" position. Another option switch might be sensed in the program to have the effect of rewinding all magnetic tapes and closing out the run. Thus, by using this switch, the operator could suspend all further processing of the file and prepare for the next job.

The external influence of the console operator on a running program would normally be reserved for exceptional situations. The console operator is advised on the exceptional situation by means of a message on the output typewriter. The console typewriter prints the contents of the N register each time the TYPE command is executed. The character to be typed out from the N register must be set up as a particular six-bit configuration which in the case of alphabetics may be slightly different from the BCD configuration for the same character within the computer. The difference in the bit configurations for these characters occurs chiefly in the representation of zone bits.

The alternation of these zone bits are shown below:

In memory	N register
00	00
01	11
10	10
11	01

The above holds true for the numbers 1-9, the letters of the alphabet A-Z, and the special symbols &, $, - and ,. The configuration for 0 (zero) is 010000; the configuration for / (slash) is 001011. In addition there are five command characters which cause the typewriter to take an action. These characters are:

000000	Space
011110	Tab
111111	Carriage return
011010	Print Red
011101	Print Black

The programmer may assume that the program steps

to accomplish any required conversion of bit configurations will be in "package" form. It will not be necessary to write a conversion routine. The conversion will be a part of a subroutine provided to transfer data for typing. The programmer will simply indicate the "calling sequence" for the message to be typed, and the information will be properly converted and transferred to the output typewriter.

CONSOLE PROGRAM CONTROL (OPTION) SWITCHES

EXTRACT — EXT
READ CONTROL SWITCHES — RCS

EXT Y EXTRACT

Each bit of Y is examined. If there is a 1 in Y in a given position, a zero is placed in the corresponding position of A. If there is a zero in a given position of Y, the corresponding position in A is left unchanged. The contents of Y are unchanged.

RCS READ CONTROL SWITCHES

Each of the 20 manually set control switches is examined. If a switch is DOWN (ON), a 1 is placed in the corresponding position of A. If a switch is UP (OFF), the corresponding position in A will not be altered. The A register should be cleared before this command is given.

EXAMPLES

1. Write the necessary commands to interrogate console option switch 11. If console option switch 11 is ON (in the "down" position), branch to the command in memory at 2550 to print a message. If console option switch 11 is OFF (in the "up" position), continue with the normal sequential execution of the program. Assume that no other console option switches may be ON.

```
0531   LDZ          }
0532   RCS          }   "Zero" A and read console switches.
0533   SRA     8        Shift right (19-11) places.
0534   BOD          }
0535   BRU  2550    }   Test and branch for console option switch 11 "on".
```

The command in memory at 0531 will replace the contents of the A register with zeros. The command at 0532 will insert a 1 bit in position 11 in the A register if the corresponding console control switch is down. The shift right command at 0533 will put the "11 bit" into bit position 19. Bit position 19 is tested by the BOD command at 0534. If bit position 19 contains a "1", console option switch 11 is on and control is transferred to 2550 by the commands at 0534 and 0535.

2. In the example above assume that other console option switches may be on.

```
3999   OCT  3777377      11111  11111  10111  11111
0531   LDZ          }
0532   RCS          }   Clear A register and read console control switches.
0533   EXT  3999        Mask out other switches.
0534   BNZ          }
0535   BRU  2550    }   Test and branch.
```

The command in memory at 0532 will insert 1 bits into the bit position of register A corresponding to any of the 20 console switches that are in the "down" position. The EXT command at 0533 will replace all bits in the A register with zeros except the bit position which corresponds to console option switch 11. If console option switch 11 is on, control will be transferred to the command in memory at 2550. The commands above are also appropriate for testing whether or not combinations of switches are "on", by variations of the bit configuration at 3999.

3. Transfer control in a continuous loop if console option switch 15 is on. Assume other console option switches may be on. If console option switch 15 is not on, continue the normal sequential execution of the program.

```
3999    OCT     3777757         (1111111111111101111)
0620    LDZ                 }
0621    RCS                 }       "Zero" and read console switches.
0622    EXT     3999        Mask out other switches.
0623    BNZ                 }
0624    BRU     0621        }   Test and transfer for console switch 15 "ON".
```

The commands at 0620 through 0623 are exactly equivalent to the commands at 0531 through 0534 in the previous example. The command at 0624 will transfer control back to the command at 0620 until console option switch 15 is in the "off" position.

4. Write the commands to "Halt" the GE 225 until further action is initiated from the console.

```
620     BRU     620     Transfer "Loop".
```

Control will be transferred in a continuous loop by the command at 0620. The loop will be identifiable by the console operator since the command will be displayed at the console. The console operator may intervene by depressing the "Manual" switch.

PRINTING OF TYPEWRITER MESSAGES

Shift A and N Right	SAN
Branch on N Register Not Ready	BNN
Type	TYP

SAN K SHIFT A AND N RIGHT

The contents of A (1-19) and N (1-6) together are shifted K places to the right. Bits shifted out of A (19) shift into N (1). Bits shifted out of N (6) are lost. If the sign of A is plus, O's fill the vacated positions of A; if the sign of A is minus, 1's fill the vacated position of A. The sign of A is unchanged.

BNN BRANCH ON N-REGISTER NOT READY

If the N-register is not available for input-output (if the last TYPE instruction has not been executed), the computer takes the next sequential instruction. If it is, the computer skips the next instruction and executes the second sequential instruction.

TYP TYPE

The six-bit, coded character in N is typed. The contents of N are not changed.

EXAMPLES

1. Assume that the message "End of Job" is in storage at locations 3995, 3996, 3997 and 3998 in the "proper configuration for typing". Write the commands to print the message on the output typewriter. Assume that the power for the output typewriter has been turned on. When all characters have been typed, transfer control to the next command in memory.

```
0999    LDZ
1000    STA     0001                    } "Zero" words 1 and 2.
1001    LDZ
1002    STA     0002
1003    LDA     3995    1               Load first word in A.
1004    BNN                             } Delay for N register busy.
1005    BRU     1004
1006    SAN       6                     } Send character to N register and type.
1007    TYP
1008    INX       6     2               }
1009    BXL      18     2               } Test for transfer for 1 word typed.
1010    BRU     1004                    }
1011    INX       1     1               }
1012    BXL       4     1               } Test and transfer for 4 words typed.
1013    BRU     1001                    }
```

The commands in memory locations 0999 through 1002 will replace the contents of modification words 1 and 2 with zeros. The first word to be typed is loaded into the A register by the command at 1003. All characters for typing must be transferred through the N register. The commands at 1004 and 1005 will interrogate the status of the N register. If the N register is occupied from the previous operation, control will be transferred in a "continuous loop" by the commands at 1004 and 1005. When the N register is ready, the shift command at 1006 will transfer one character to the N register from the A register for typing. The TYP command will type the character represented by the contents of the N register on the output typewriter. The commands in storage at 1008, 1009 and 1010 will increment and test the contents of modification word 2 for 18. If all 3 of the digits in the word have been shifted to the output typewriter and typed, the contents of modification word 2 will be equal to 18 and control will be transferred to the command at 1011. If all 3 characters in the word have not been typed, control is transferred back to the command at 1004 to set up the next character, etc.

After the 3 characters in each word have been typed, the commands in storage at 1011, 1012 and 1013 will increment modification word 1 by 1. When modification word 1 is less than 4, control will be transferred back to the command at 1001 to begin typing the contents of the second word. When all words have been typed, the contents of modification word 1 will be equal to 4 and control will be transferred to the command at 1014.

2. Assume that there is a 50-word message in storage beginning at location 0500, and that the message must be converted to the proper configuration for typing. Write the necessary commands to return the typewriter carriage before printing is to begin.

```
0100    SPB     3500    1
0101    LDA     0500
0102    DEC       50
```

} Transfer to subroutine calling sequence.

The above instructions provide the necessary calling sequence information for a subroutine in storage beginning at location 3500. This subroutine for typing messages is part of the GAP subroutine library. In the present case it is assumed that the programmer has previously specified that the subroutine be placed in memory at the given location. The SPB command at 0100 will transfer control to the subroutine for typing messages. The location of the SPB command itself will be preserved in modification word 1. The subroutine will then call in the required parameter information regarding the beginning storage location of the message and the number of words to be typed. Normally, calling sequence information is indicated by the programmer utilizing the DEC command. The LDA command at 0101 will in this case not only provide the necessary calling sequence information, but will also indicate that the typewriter carriage is to be returned before typing is to begin. For subsequent carriage returns, the programmer must include his own carriage returns in the material to be typed out.

3. In the above example write the necessary commands to type the message without initially returning the typewriter carriage.

```
0100    SPB     3500    1
0101    EXT     0500
0102    DEC       50
```

} Transfer to subroutine calling sequence.

The calling sequence information will be utilized as previously explained. However, the EXT command at 0101 will cause the desired message to be typed without a return of the typewriter carriage before typing begins.

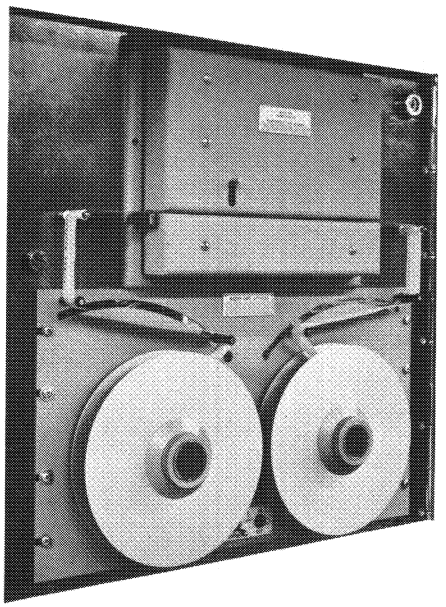


Figure 21 Paper Tape Reader

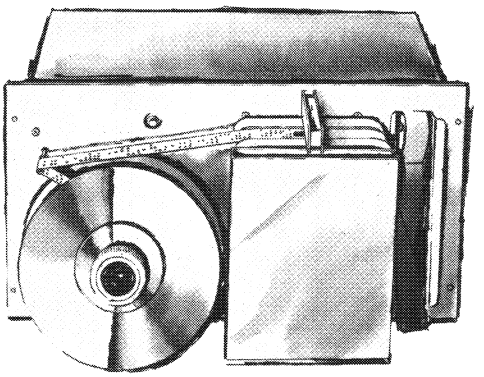


Figure 22 Paper Tape Punch

PUNCHED PAPER TAPE OPERATIONS

Each character transferred to or from punched paper tape must pass through the N register. In order to transmit or receive a character to or from the N register the character must be shifted - either from the A register to the N register or from the N register to the A register.

Configurations of paper tape input and output words will normally be converted to or from the configurations of words in FE 225 internal storage. The conversion of paper tape input and output will normally be assigned to appropriate subroutine aids as described in Programming Aids, GE 225 Programming Library.

TRANSFERS OF DATA TO AND FROM PUNCHED PAPER TAPE

Read Paper Tape	RPT
Write Paper Tape	WPT
Shift N and A Right	SNA

RPT — READ PAPER TAPE

The N register is cleared, and one six-bit coded character is read into N. Other insturctions not using N may be executed during this time.

WPT — WRITE PAPER TAPE

The six-bit coded character in N is punched. The contents of N are not changed. Other instructions not using N may be executed during this time.

SNA K — SHIFT N AND A RIGHT

The contents of N (1-6) and A (1-19) together are shifted K places to the right. Bits shifted out of N (6) shift into A (1). Vacated positions in N are filled with O's. Bits shifted out of A (19) are lost. The sign of A is unchanged.

EXAMPLES

1. Write the commands to transfer the next 3 characters from paper tape to the A register.

```
1998    BNN
1999    BRU     1998        Delay for "N" register busy.
2000    RPT                 Read paper tape.
2001    BNN     2000
2002    BRU     2001        Delay for "N" register busy.
2003    SNA     6
2004    RPT
2005    BNN
2006    BRU     2005
2007    SNA     6           Read next 2 characters from paper tape to
2008    RPT                 A register.
2009    BNN
2010    BRU     2009
2011    SNA     6
2012    SRA     1           Position 3 characters as a BCD word.
```

The next 3-characters will be read in from the paper tape reader and positioned as 3 characters in a BCD word. Appropriate subroutines from the GE 225 Programming Library may be utilized for paper tape input. These subroutines will not only accomplish the functions of reading paper tape, but will also convert the 6 bit paper tape code to the "proper configuration" for use in further processing.

2. Write the commands to transfer 3 characters from the A register to the paper tape punch. Assume the 3 characters are represented in the "proper configuration" for punching.

```
3000    BNN
3001    BRU     3000
3002    SAN     6
3003    WPT
3004    BNN                 Write next 3 characters from A register to
3005    BRU     3004        paper tape.
3006    SAN     6
3007    WPT
3010    BNN
3011    BRU     3010
3012    SAN     6
3013    WPT
```

The next 3 characters will be punched out on the paper tape punch. Appropriate subroutines from the GE 225 Programming Library may be utilized for paper tape output. These subroutines will not only accomplish the function of punching paper tape but will also convert the 6 bit character code of data within storage to the "proper configuration" for punching.

PUNCHED CARD OPERATIONS

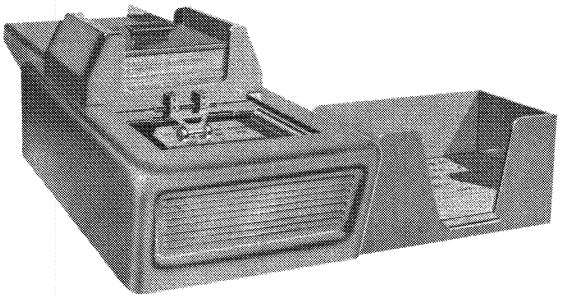


Figure 23 Card Reader

CARD READING

During the alphanumeric mode, each card column (one character) is converted into an equivalent binary-coded decimal form consisting of 6 bits and forms one memory word out of each group of three card columns (characters). Three card characters are represented by 18 bits while one memory word consists of 20 bits. The three characters making up the 18 bits are placed in the 18 least significant bits of the memory word. Thus, with three digits to a memory location 27 memory locations must be used per card with the 27th memory location containing only two characters.

The starting memory address into which information is placed is designated by the program and must be a multiple of 128 but less than 2048. An address counter in the card input logic circuitry is preset to the starting address at the start of each card input operation. Only four cards of information are contained in memory at any one time unless transferred to other areas. After four cards have been read into memory, the fifth card is read into the same location as the first card and so forth.

Data from first card - Read into memory at (starting address) to (starting address +26)

Data from second card - Read into memory at (starting address +32) to (starting address +58)

Data from third card - Read into memory at (starting address +64) to (starting address +90)

Data from fourth card - Read into memory at (starting address +96) to (starting address +122)

Data from fifth card - Read into memory at (starting address) to (starting address +26)

etc.

The word following the last word filled from the card (starting address +27, starting address +59, starting address +91, starting address 123) will automatically receive the following information:

Bit position 0 - "On" when the reading of card is complete.

Bit position 1 - "On" when the card is the last card in the input deck.

Bit positions 14 through 19 - "On" to indicate proper synchronization of the card reader during the reading of the card.

During the binary mode of operation, the circuitry forms one 20-bit memory word out of each set of two ten-bit card columns. The first column is placed in the 10 most significant bits of the memory word, the second column in the 10 least significant bits of the memory word and so forth. The 80 columns of punched information must, during the binary mode, therefore, be stored in forth memory locations. The starting address into which information is placed is designated by the program and must be a multiple of 128 but less then 2048. An address counter in the card input circuitry is preset to the starting address at the start of each card input operation. Only two cards of binary information are contained in memory at any one time. After two cards have been read into memory, the third card is read into memory at the same location as the first card and so forth.

Data from first card - Read into memory at (starting address) to (starting address +39)

Data from second card - Read into memory at (starting address +64) to (starting address +103)

Data from third card - Read into memory at (starting address) to (starting address +30)

etc.

The word after the word following the last word filled from the card (starting address +41, starting address +105) will automatically receive the following information:

Bit Position 0 - "On" when the reading of the card is complete.

Bit position 1 - "On" when the card is the last card in the input deck.

Bit positions 10 through 19 - "On" to indicate proper synchronization of the card reader during the reading of the card.

CARD PUNCHING

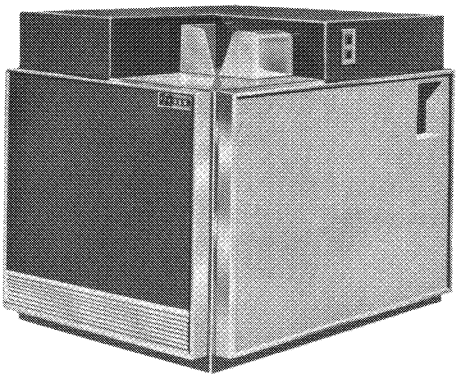


Figure 24 Card Punch

In the alphanumeric mode, a card punching command initiates the necessary steps to energize the card punch and read into the address counter the memory location of the first word to be punched. An 80-bit register is filled with the 80 bits which will constitute row 12 of the card, the first row to be punched. This is accomplished by sequentially reading out of memory, 27 words of output data, converting the three, 6-bit alphanumeric characters contained in each word (there are only two characters in the 27th word) to the 12-bit Hollerith code and reading what will become the 12th row into the 80 bit register. The twelfth row is punched, followed in sequence by the other 11 rows.

In punching a card in the binary mode, the output logic functions in the same manner as for the alphanumeric mode with the following exceptions. A 20-bit GE 225 word in memory will be punched on the card as two ten-bit columns. (Rows 11 and 12 on the card are not used in the binary mode.) The output information for one card, therefore, is contained in 40 memory words; the ten most significant bit positions of the first binary word occupying the first card column, the ten least significant bit positions of the first binary word occupying the second card column, etc. To punch the zero row on the card, therefore, the output logic must read the corresponding 2 bits from each of the 40 words in memory into the 80-bit register.

TRANSFERS OF BINARY CODED DECIMAL DATA TO AND FROM PUNCHED CARDS

Read Cards Decimal	RCD
Halt Card Reader	HCR
Write Card Decimal	WCD

RCD Y READ CARDS DECIMAL

This command initiates continuous reading of decimal cards into memory starting at location Y, where Y, is a multiple of 128 and less than 2048. The first card will be read into locations Y through Y + 26, the second into Y + 32 through Y + 58, the third into Y + 64 through Y + 90, the fourth into Y + 96 through Y + 122, the fifth into Y + 26, etc. After each card is read in the sign bit of the word after the last word of the card (Y + 27, Y + 59, Y + 91, or Y + 123) will be set minus. After the last card of the deck is read in, bit position 1 of the word after the last word of the card (Y + 27, Y + 59, Y + 91, or Y + 123) will be set to a 1. If the card reader is not in ready status when the READ instruction is given, the computer will halt.

HCR HALT CARD READER

This instruction halts the card feed. If a card is being read at the time this instruction is given, the reading of this card into memory will be completed, after which no further cards will be read until another READ instruction is given. This instruction does not delay the computer until input is complete. The program continues in sequence, therefore a delay must be programmed to insure that the information is in memory before attempting to utilize it.

WCD Y WRITE CARD DECIMAL

This instruction causes the information in memory locations Y through Y + 26 (where Y is a multiple of 128) to be punched into a card in alphanumeric format. If the card punch is not in ready status when the WRITE instruction is given, the computer will halt.

EXAMPLES

1. Read one card into magnetic core storage beginning with memory location 0128. Halt the card reader.

```
1000    RCD    0128   } Read card and halt.
1001    HCR           }
```

One card will be read, and the reader will be halted. Memory locations 0128 through 0154 will receive equivalent binary coded decimal representation of information punched in columns 1-80 of the card.

2. Read cards into magnetic core storage, but do not halt the card reader.

```
1000    RCD    0128    Read cards continuously.
```

The card reader will continue to read cards into succeeding memory locations: 160-186, 192-218, 224-250. The fifth card will be read into 0128-0154, the next into 160-186, etc.

3. Punch one card from magnetic core storage beginning with memory location 0256.

```
1000    WCD    0256    Punch card and halt.
```

One card will be punched and the punch will be halted. The contents of memory locations 256-282 will be punched into columns 1-80 of the card in Hollerith code.

PROGRAM REQUIREMENTS RELATED TO TRANSFERS OF BINARY CODED DECIMAL DATA TO AND FROM PUNCHED CARDS

Branch on Card Reader Ready	BCR
Branch on Card Reader Not Ready	BCN
Branch on Card Punch Ready	BPR
Branch on Card Punch Not Ready	BPN

BCR BRANCH ON CARD READER READY

If the card reader is ready to read cards and the card hopper is not empty, the computer takes the next sequential instructions, if not, the computer skips the next instruction and executes the second sequential instruction.

BCN BRANCH ON CARD READER NOT READY

If the card reader is not ready to read cards, or if the card hopper is empty, the computer rakes the next sequential instruction. If the reader is ready and the card hopper is not empty, the computer skips the next instruction and executes the second sequential instruction.

BPR BRANCH ON CARD PUNCH READY

If the card punch is in a ready status, the computer takes the next sequential instruction; if not, the computer skips the next instruction and executes the second sequential instruction.

BPN BRANCH ON CARD PUNCH NOT READY

If the card punch is not in a ready status, the computer takes the next sequential instruction; if it is, the computer skips the next instruction and executes the second sequential instruction.

EXAMPLES

1. Read a single card. Delay further processing until the card has been completely read.

```
1000    RCD    0128   }
1001    HCR           }
1002    BCN           }  Read card and delay processing until completed.
1003    BRU    1002   }
```

The branch on card reader not ready command will transfer control to the next command until the card has been completely read. After reading has been completed, control will transfer to the command located at 1004.

2. Read cards continuously. Delay processing until each card has been completely read.

```
1502    RCD    0128   }
1503    LDA    0155   }
1504    BPL           }  Read card and delay processing until completed.
1505    BRU    1504   }
```

The load A command loads the word after the last word to be filled from the card into the A register. The sign bit of the last word will be set to 1 when the card is completely read. The branch on plus command will transfer control to the next command until the card has been completely read. After reading has been completed, control will transfer to the command located at 1506. After processing of the first card has been completed, it will be necessary to determine in the program whether the second card has been completely read, etc.

3. Determine whether or not the card just read is the last card of the deck.

```
1501    RCD    0128   }
1502    HCR           }
1503    LDA    0155   }  Read card and delay until completed.
1504    BPL           }
1505    BRU    1504   }
1506    SLA    0001   }
1507    BOV           }  Test and branch for last card in deck.
1508    BRU    3000   }
```

The word after the last word filled has been loaded into the A register by the load A command in memory storage at 1503. Bit position 1 of the last word filled will be set to a 1 after the last card of the deck has been completely read. The shift left A command will cause the 1 to be shifted out of bit position 1 and the overflow indicator will be turned on. The branch on overflow command will transfer control to the command in storage at 3000 when the last card of the deck has been read. For all other cards control will be transferred to the command in storage at 1509.

4. Determine whether or not the card just read is the last card of the deck (alternate method).

```
3999    DEC    2777777  10111  11111  11111  11111
1501    RCD    0128   }
1502    HCR           }
1503    LDA    0155   }  Read card and test for comparison.
1504    BPL           }
1505    BRU    1504   }
1506    SRD       6   }
1507    SLA       6   }  Store bits 14-19 in the Q register for later re-
1508    EXT    3999   }    quirements.
1509    BNZ           }
1510    BRU    3000   }  Test for last card in deck and branch.
```

The constant located in storage at 3999 will cause every bit position of the word in the A register to be set to 0 for every card except the last card of the deck when the extract command is executed. The branch of non-zero command will transfer control to the command in storage at 3000 when the last card of the deck has been read. For all other cards control will be transferred to the command in storage at 1511. The commands at 1506 and 1507 will store bits 14-19 in the Q register for later requirements.

5. In the above example determine whether there was proper synchronization during the reading of the card.

```
3998    OCT    0000077
1511    LDZ           }
1512    SLD       6   }  Clear A and recover bits.
1513    SUB    3998   }
1514    BNZ           }
1515    BRU    4000   }  Test and branch for synchronization.
```

The commands at 1511 and 1512 will recover bits 14-19 from the Q register. The contents of the A register should be zero after the constant at 3998 is subtracted if there was proper synchronization during the reading of the card.

6. Punch a single card. Delay further processing until the card has been completely punched.

```
1000    WCD    0256   }
1001    BPN           }  Punch and test for completion
1002    BRU    1001   }
```

The branch on punch not ready command will transfer control to the command in storage at 1002 until punching of the card is completed. When card punching is completed, computer control will be transferred to the command in storage at 1003.

TRANSFERS OF BINARY DATA TO AND FROM PUNCHED CARDS

Read Cards Binary	RCB
Write Cards Binary	WCB

RCB Y READ CARDS BINARY

This command initiates continuous reading of binary cards into memory starting at location Y, where Y is a multiple of 128 and less than 2048. The first card will be read into locations Y through Y + 39, the second into Y + 64 through Y + 103, the third into Y through Y + 39, etc. After each card is read in, the SIGN position of the second word following the card image (Y + 41 or Y + 105) will be set to a 1. After the last card of a deck is read in, position 1 of the second word following this card image (Y + 41 or Y + 105) will be set to 1. If the card reader is not in ready status when the READ instruction is given, the computer will halt.

WCB Y WRITE CARDS BINARY

This instruction causes the information in memory locations Y through Y + 39 (where Y is a multiple of 128) to be punched into a card in binary format. If the card punch is not in ready status when the WRITE instruction is given, the computer will halt.

EXAMPLES

1. Read one card into magnetic core storage beginning with memory location 0128. Halt the card reader.

```
1000    RCB    0128   }
1001    HCR           } Read card and halt.
```

One card will be read, and the reader will halted. Memory locations 0128-0167 will receive equivalent binary representation of information punched in columns 1-80 of the card.

2. Read cards into magnetic core storage, but do not halt the card reader.

```
1000    RCB    0128   Read cards continuously.
```

The card reader will continue to read cards into a succeeding memory location at 0192-231. The third card will be read into 0128-0167, the next into 0192, 231, etc.

3. Punch one card from magnetic core storage beginning with memory location 0256.

```
1000    WCB    0256   Punch card and halt.
```

One card will be punched and the punch will be halted. The contents of memory locations 0256-0295 will be punched into columns 1-80 of the card as follows:

Word	Columns
0256	1 and 2
0257	3 and 4
↓	↓
0295	79 and 80

Each word is distributed, left to right in punching positions 0 - 9 of two columns of the card.

PROGRAM REQUIREMENTS RELATED TO TRANSFERS OF BINARY DATA TO AND FROM PUNCHED CARDS

The program requirements for transfers of binary data to and from punched cards are idential to those discussed previously regarding binary coded decimal data except:

1. Bits 0 and 1, for indication of card reading completion and last card in the deck, will appear in the second word following the card image (starting address +41 and starting address +105).

2. Bit Positions 10 through 19 will be "on" to indicate proper synchronization of the card reader during the reading of the card.

The programmer may program requirements as illustrated in the preceding examples for binary coded decimal data. However, for binary data transfers:

a. The LDA command at 1503 within the previous example 2 will now reference memory location 0169 instead of 155.

b. The commands at 1506 and 1507 within previous example 4 will now SRD 10 and SLA 10 instead of SRD 6 and SLA 6 so as to store bits 10-19 in the Q register for later requirements.

c. When recovering bits 10 through 19 in example 5 the SLD command at 1512 should be SLD 10.

d. When testing the condition of bits 10 through 19 in previous example 5 for "on", the constant at 3998 should be changed to:

```
3998        OCT         0001777
```

CASE PROBLEMS ILLUSTRATING REQUISITION COST AND LABOR PRICING

Problem: (1) Price labor vouchers
(2) Obtain requisition cost

INPUT: Weekly labor voucher cards merged with requisition cost summary cards by requisition number.

OUPUT: (1) Priced labor vouchers
(2) New requisition cost summary cards

INPUT CARD FORMAT:

Labor Voucher:	Badge	1-5
	Requisition	7-13
	Area	28-30
	Hours	31-33 (1 Decimal)
	Card Code	80 (0)
Requisition Cost Card	Requisition	7-13
	Labor	27-33
	O/H	36-42
	Card Code	80 (1)

OUTPUT CARD FORMAT:

Priced Labor Voucher:	Badge	1-5
	Requisition	7-13
	Area	28-30
	Hours	31-33
	Labor	34-39 (Hours X Area Rate)
	Card Code	80 (0)
Requisition Cost Card:	Requisition	7-13
	Labor	27-33
	O/H	36-42 (Labor X Requisition O/H %)
	Card Code	80 (1)

REQN O/H percentage based on 1st digit of requisition

1st Digit of Reqn.	O/H %
0, 1, 2	100
3, 4	150
5, 6, 7, 8	250
9	300

VOUCHER Hourly rate based on area (4 decimals)

Area No.	Area Rate
050	2.5623
125	3.1275
220	2.0000
360	3.2500
420	1.7500
500	2.2500
600	2.3750
640	2.8715
703	2.4571
800	2.7550

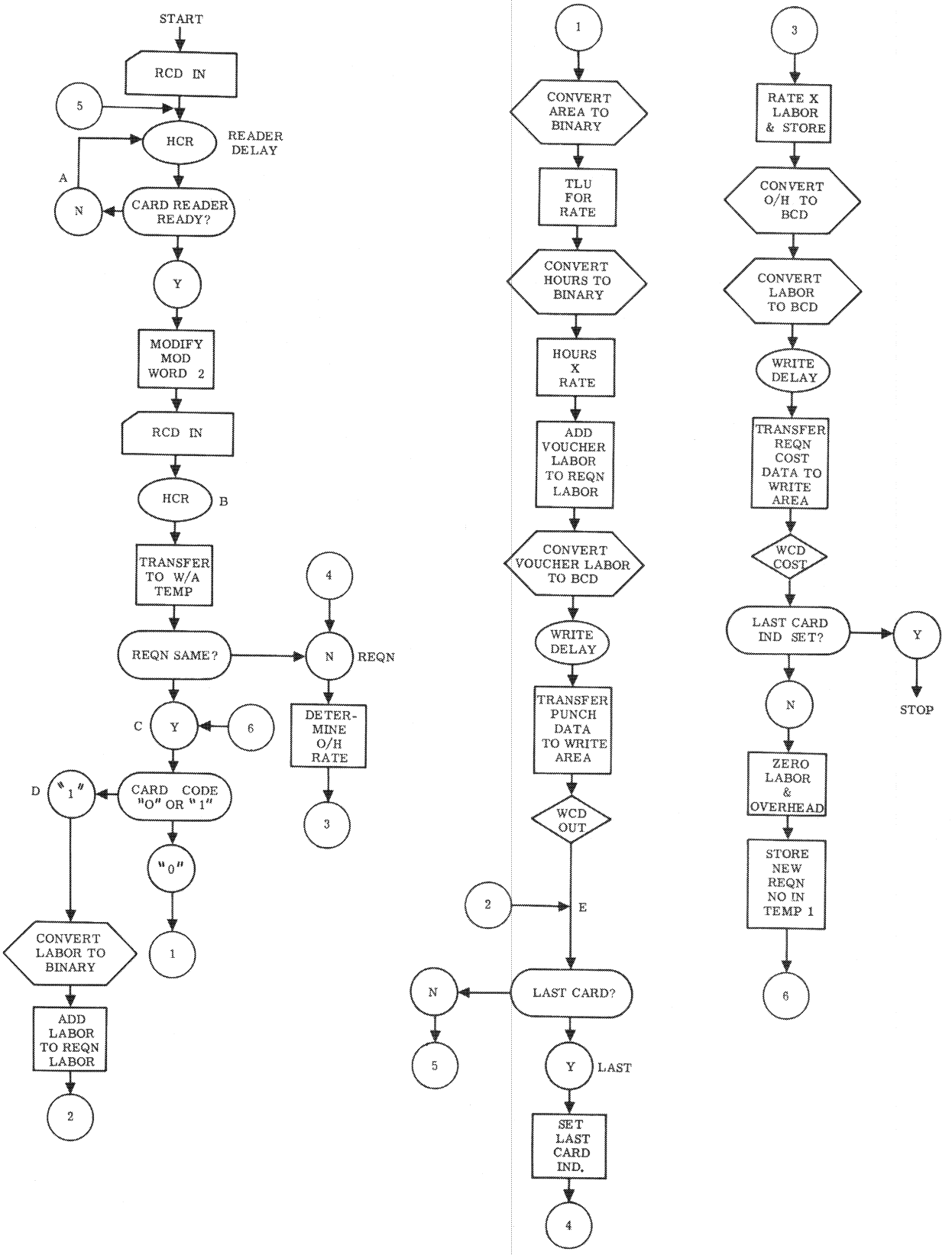
CASE PROBLEM
MEMORY ALLOCATION LAYOUT SHEET
INPUT

	Labor Voucher				Requisition Cost Card			
LOC	0 1	2 7	8 13	14 19	0 1	2 7	8 13	14 19
128/256	00	Badge	Badge	Badge	00	----	----	----
129/257	00	Badge	Badge	----	00	----	----	----
130/258	00	Reqn	Reqn	Reqn	00	Reqn	Reqn	Reqn
131/259	00	Reqn	Reqn	Reqn	00	Reqn	Reqn	Reqn
132/260	00	Reqn	----	----	00	Reqn	----	----
133/261	00	----	----	----	00	----	----	----
134/262	00	----	----	----	00	----	----	----
135/263	00	----	----	----	00	----	----	----
136/264	00	----	----	----	00	----	----	Labor
137/265	00	Area	Area	Area	00	Labor	Labor	Labor
138/266	00	Hours	Hours	Hours	00	Labor	Labor	Labor
139/267	00	----	----	----	00	----	----	O/H
140/268	00	----	----	----	00	O/H	O/H	O/H
141/269	00	----	----	----	00	O/H	O/H	O/H
142/270	00	----	----	----	00	----	----	----
143/271	00	----	----	----	00	----	----	----
144/272	00	----	----	----	00	----	----	----
145/273	00	----	----	----	00	----	----	----
146/274	00	----	----	----	00	----	----	----
147/275	00	----	----	----	00	----	----	----
148/276	00	----	----	----	00	----	----	----
149/277	00	----	----	----	00	----	----	----
150/278	00	----	----	----	00	----	----	----
151/279	00	----	----	----	00	----	----	----
152/280	00	----	----	----	00	----	----	----
153/281	00	----	----	----	00	----	----	----
154/282	10	----	Card Code	----	10	----	Card Code	----

CASE PROBLEM
MEMORY ALLOCATION LAYOUT SHEET
OUTPUT

		Priced Labor Voucher				Requisition Cost Card				
LOC	0 1	2 7	8 13	14 19		LOC	0 1	2 7	8 13	14 19
384	00	Badge	Badge	Badge		512	00	----	----	----
385	00	Badge	Badge	----		513	00	----	----	----
386	00	Reqn	Reqn	Reqn		514	00	Reqn	Reqn	Reqn
387	00	Reqn	Reqn	Reqn		515	00	Reqn	Reqn	Reqn
388	00	Reqn	----	----		516	00	Reqn	----	----
389	00	----	----	----		517	00	----	----	----
390	00	----	----	----		518	00	----	----	----
391	00	----	----	----		519	00	----	----	----
392	00	----	----	----		520	00	----	----	Labor
393	00	Area	Area	Area		521	00	Labor	Labor	Labor
394	00	Hours	Hours	Hours		522	00	Labor	Labor	Labor
395	00	Amt	Amt	Amt		523	00	----	----	O/H
396	00	Amt	Amt	Amt		524	00	O/H	O/H	O/H
397	00	----	----	----		525	00	O/H	O/H	O/H
398	00	----	----	----		526	00	----	----	----
399	00	----	----	----		527	00	----	----	----
400	00	----	----	----		528	00	----	----	----
401	00	----	----	----		529	00	----	----	----
402	00	----	----	----		530	00	----	----	----
403	00	----	----	----		531	00	----	----	----
404	00	----	----	----		532	00	----	----	----
405	00	----	----	----		533	00	----	----	----
406	00	----	----	----		534	00	----	----	----
407	00	----	----	----		535	00	----	----	----
408	00	----	----	----		536	00	----	----	----
409	00	----	----	----		537	00	----	----	----
410	00	----	-0- Card Code			538	00	----	-1- Card Code	----

CASE PROBLEM
FLOW CHART



225 CODING SHEET

PROBLEM: Reqn Cost & Labor Pricing

WRITTEN BY:

Symbol	Opr	Operand	X	Remarks	Sequence
1 6	8 10	12 19	20	22 75	76 80
	ORG	4			
TEMP	BSS	27		Work area for input cards	
	ORG	40			
TEMPI	BSS	14		Work area for cost data	
	ORG	60		Area table start	
TAB	DEC	50		Area 050	
	DEC	75		Area 125	
	DEC	95		Area 220	
	DEC	140		Area 360	
	DEC	60		Area 420	
	DEC	80		Area 500	
	DEC	100		Area 600	
	DEC	40		Area 640	
	DEC	63		Area 703	
	DEC	97		Area 800	
	REM			Rate Table Start	
TABL	DEC	25623		Rate 2.5623	
	DEC	31275		Rate 3.1275	
	DEC	20000		Rate 2.0000	

225 CODING SHEET

PROBLEM: Reqn Cost & Labor Pricing

WRITTEN BY: _____

Symbol (1–6)	Opr (8–10)	Operand (12–19)	X (20)	Remarks (22–75)	Sequence (76–80)
	DEC	32500		Rate 3.2500	
	DEC	17500		Rate 1.7500	
	DEC	22500		Rate 2.2500	
	DEC	23750		Rate 2.3750	
	DEC	28715		Rate 2.8715	
	DEC	24571		Rate 2.4571	
	DEC	27550		Rate 2.7550	
	DEC	0		Rate 0	
MODIFY	DEC	128		Double length word	
	DEC	0			
MASK	OCT	-0777777		Last card mask	
ZERO	OCT	-1770077		Use to mask for card code	
F	DEC	0		W/A for transfer control to "temp"	
TEMP2	DEC	0		W/A for rounding	
TEMP3	DEC	0		Last card indicator	
TEMP4	BSS	2		Temp. storage for labor voucher in binary	
TABL2	DEC	100		Reqn % O/H Rate - 1st digit 0	
	DEC	100		Reqn % O/H Rate - 1st digit 1	
	DEC	100		Reqn % O/H Rate - 1st digit 2	

225 CODING SHEET

PROBLEM: Reqn Cost & Labor Pricing

WRITTEN BY: _____

Symbol	Opr	Operand	X	Remarks	Sequence
1 6	8 10	12 19	20	22 75	76 80
	DEC	150		Reqn % O/H Rate - 1st digit 3	
	DEC	150		Reqn % O/H Rate - 1st digit 4	
	DEC	250		Reqn % O/H Rate - 1st digit 5	
	DEC	250		Reqn % O/H Rate - 1st digit 6	
	DEC	250		Reqn % O/H Rate - 1st digit 7	
	DEC	250		Reqn % O/H Rate - 1st digit 8	
	DEC	300		Reqn % O/H Rate - 1st digit 9	
OVERHD	BSS	2		Used to store reqn overhead	
LABOR	BSS	2		Used to store Reqn labor	
FIFTY	DEC	50			
5HUND	DEC	0		Rounding constants	
	DEC	500			
ROUND	DEC	100			
ROUNDA	DEC	1000			
G	DEC	128		Constant	
	ORG	128		Input area	
IN	BSS	27			
	ORG	256		Input area	
	BSS	27			

225 CODING SHEET

PROBLEM: Reqn Cost & Labor Pricing

WRITTEN BY: _____

Symbol	Opr	Operand	X	Remarks	Sequence
1 6	8 10	12 19	20	22 75	76 80
	ORG	384		Priced voucher output area	
OUT	BSS	26		Established card code "0" in col. 80	
	ALF	⊙0⊙			
	ORG	512		Reqn cost area output area	
COST	BSS	26		Establish card code "1" in col. 80	
	ALF	⊙1⊙			
	ORG	600		Program start	
START	RCD	IN			
	HCR				
A	BCN				
	BRU	A			
	DLD	MODIFY			
	STA	2			
	XAQ				
	DST	MODIFY			
	ADD	G			
	STO	B			
	RCD	IN	2		
	HCR				

225 CODING SHEET

PROBLEM: Reqn Cost & Labor Pricing

WRITTEN BY:

Symbol	Opr	Operand	X	Remarks	Sequence
1 6	8 10	12 19	20	22 75	76 80
	LDZ			Zero modification word 3	
	STA	3			
B	DLD	0	3		
	DST	TEMP	3	Transfer input data into W/A "temp"	
	INX	2	3		
	BXL	28	3		
	BRU	B			
	LDA	TEMP1+4			
	SUB	TEMP+4			
	BNZ				
	BRU	REQN			
	DLD	TEMP1+2		Compare reqn number - temp; temp 1	
	DSU	TEMP+2			
	BNZ				
	ERU	REQN			
	XAQ				
	BNZ				
	BRU	REQN			
C	LDA	TEMP+26			

225 CODING SHEET

PROBLEM: Reqn Cost & Labor Pricing

WRITTEN BY: _____

Symbol	Opr	Operand	X	Remarks	Sequence
1 6	8 10	12 19	20	22 75	76 80
	EXT	ZERO		CARD CODE ZERO OR ONE?	
	BNZ				
	BRU	D		CARD CODE 1	
	INX	-28	3	ZERO MODIFICATION WORD 3	
	SPB	BCD BIN	1		
	DEC	4			
	DEC	28		CONVERT AREA TO BINARY	
	DEC	3			
	BRU	ERROR			
	LAQ			MOVE Q TO A	
H	SUB	TAB	3		
	BNZ				
	BRU	TEST		TLU FOR CORRECT AREA	
	INX	1	3	RELATIVE RATE ADDRESS MAINTAINED IN	
	BXL	11	3	MODIFICATION WORD 3	
	BRU	H			
	BRU	ERROR			
TEST	SPB	BCD BIN	1		
	DEC	4			

225 CODING SHEET

PROBLEM: Reqn Cost & Labor Pricing

WRITTEN BY:

Symbol	Opr	Operand	X	Remarks	Sequence
1 6	8 10	12 19	20	22 75	76 80
	DEC	31		Convert hours to binary	
	DEC	3			
	BRU	ERROR			
	MPY	TABL	3	Rate table modified by mod. word 3	
	DAD	5HUND		Multiply and round 500	
	DVD	ROUNDA		1000	
	MAQ			Move A to Q	
	DST	TEMP4			
	DAD	LABOR		Add voucher labor to reqn labor & store	
	DST	LABOR			
	DLD	TEMP4		Load "A" with voucher labor	
	SPB	BINBCD	1		
	DEC	4		Convert vou. labor to BCD	
	DEC	39			
	DEC	6			
	BRU	ERROR			
	LDZ			Zero modification word 3	
	STA	3			
I	BPN			Checking card punch ready	

225 CODING SHEET

PROBLEM: Reqn Cost & Labor Pricing
WRITTEN BY:

Symbol	Opr	Operand	X	Remarks	Sequence
1 6	8 10	12 19	20	22 75	76 80
	BRU	I			
TRANS	DLD	TEMP	3		
	DST	OUT	3	Transfer priced voucher data to punch	
	INX	2	3		
	BXL	14	3		
	BRU	TRANS			
	WCD	OUT		Write priced voucher	
	BRU	E			
D	SPB	BCDBIN	1		
	DEC	4		Convert reqn labor to binary	
	DEC	27			
	DEC	7			
	BRU	ERROR			
	DAD	LABOR		Add reqn labor to reqn labor	
	DST	LABOR			
E	LDA	TEMP+26			
	EXT	MASK		Check for last card	
	BZE				
	BRU	A			

225 CODING SHEET

PROBLEM: Reqn Cost & Labor Pricing

WRITTEN BY:

Symbol	Opr	Operand	X	Remarks	Sequence
1 6	8 10	12 19	20	22 75	76 80
	STA	TEMP3		Set last card indicator	
REQN	LDA	TEMP1+2			
	SRA	12		Determine O/H % by 1st digit of reqn.	
	STA	1			
	LDA	TABL2	1		
	STA	F			
	DLD	LABOR			
	MPY	F			
	XAQ				
	ADD	FIFTY			
	STA	TEMP2			
	LDZ			MPY labor X O/H % round	
	DVD	ROUND			
	STA	OVERHD			
	LDA	TEMP2			
	XAQ				
	DVD	ROUND			
	STA	OVERHD+1			
	DLD	OVERHD			

CODING SCHEME FOR SIMULTANEOUS READ-PROCESS-PUNCH

WRITTEN BY: _____ PAGE 1 OF 3

Symbol (1–6)	Opr (8–10)	Operand (12–19)	(20)	Remarks (22–75)	Sequence (76–80)
NOREAD	BCN			See if Card Reader is Ready	00001
	BRU	NOREAD		Wait for Card Reader	00002
	RCD	384		Read a Decimal Card into 384 (Area C)	00003
	HCR			Stop after Single Card	00004
BACK	BCN				00005
	BRU	BACK		Wait for Card Reader	00006
READ	RCD	256		Read a Decimal Card into 256 (Area B)	00007
	HCR			Read Single Card Only	00008
	SPB	BCD BIN	1	Branch to BCD to BIN Conversion	00009
CALC1	DEC	0384		**Constant = To Area C Read In	00010
	DEC	0055		Column 5 of Card	00011
	DEC	0003		3 Digit Field	00012
	BRU	ERROR 1		Error Routine as Required	00013
	DST	RATE		Temporary Storage of Binary Rate	00014
	SPB	BCD BIN	1	Branch to BCD to BIN Conversion (Pieces to Binary)	00015
CALC2	DEC	0384		**Constant = to Area C Read In	00016
	DEC	0032		Column 32 of Card	00017
	DEC	0002		2 Digit Field	00018
	BRU	ERROR 2		Error Routine as Required	00019

225 CODING SHEET

PROBLEM: Reqn Cost & Labor Pricing

WRITTEN BY:

Symbol	Opr	Operand	X	Remarks	Sequence
1 6	8 10	12 19	20	22 75	76 80
	STA	TEMP 3		Set last card indicator	
REQN	LDA	TEMP 1+2			
	SRA	12		Determine O/H % by 1st digit of reqn.	
	STA	1			
	LDA	TABLE 2	1		
	STA	F			
	DLD	LABOR			
	MPY	F			
	XAQ				
	ADD	FIFTY			
	STA	TEMP 2			
	LDZ			MPY labor X O/H % round	
	DVD	ROUND			
	STA	OVERHD			
	LDA	TEMP 2			
	XAQ				
	DVD	ROUND			
	STA	OVERHD+1			
	DLD	OVERHD			

CODING SCHEME FOR SIMULTANEOUS READ-PROCESS-PUNCH

WRITTEN BY: _____

Symbol (1–6)	Opr (8–10)	Operand (12–19)	(20)	Remarks (22–75)	Sequence (76–80)
NOREAD	BCN			See if Card Reader is Ready	00001
	BRU	NOREAD		Wait for Card Reader	2
	RCD	384		Read a Decimal Card into 384 (Area C)	3
	HCR			Stop after Single Card	4
BACK	BCN				5
	BRU	BACK		Wait for Card Reader	6
READ	RCD	256		Read a Decimal Card into 256 (Area B)	7
	HCR			Read Single Card Only	8
	SPB	BCD BIN	1	Branch to BCD to BIN Conversion	9
CALC1	DEC	0384		**Constant = To Area C Read In	10
	DEC	0005		Column 5 of Card	11
	DEC	0003		3 Digit Field	12
	BRU	ERROR 1		Error Routine as Required	13
	DST	RATE		Temporary Storage of Binary Rate	14
	SPB	BCD BIN	1	Branch to BCD to BIN Conversion (Pieces to Binary)	15
CALC2	DEC	0384		**Constant = to Area C Read In	16
	DEC	0032		Column 32 of Card	17
	DEC	0002		2 Digit Field	18
	BRU	ERROR 2		Error Routine as Required	19

CODING SCHEME FOR SIMULTANEOUS READ-PROCESS-PUNCH

WRITTEN BY: _____

Symbol	Opr	Operand	X	Remarks	Sequence
1 6	8 10	12 19	20	22 75	76 80
	MPY	RATE+1		Since Rate Only 1 Word Binary Value	00020
	SPB	BINBCD	1	Branch to Binary to BCD Extended Am't to	21
CALC3	DEC	0384		**Constant = to Area C	22
	DEC	0045		Column 32 of Card	23
	DEC	0005		5 Digit Field	24
	RBU	ERROR 3		Error Routine as Required	25
	LDA	PUNCH		Bit Pattern of Command at Punch	26
	XAO			Put Bit Pattern of Punch in 0	27
	LDA	CALC 1			28
	ADD	CONST 2		Binary Const = Decimal 2	29
	STO	PUNCH		Change Punch Instruction to Punch Calc. Area	30
	LDA	READ		Bit Pattern of Command of Read	31
	STO	CALC 1			32
	STO	CALC 2		Change Constants to Refer to Read Area	33
	STO	CALC 3			34
	LAO			Put Punch Area Back in A	35
	SUB	CONST 2		Deduct 2 from Binary Punch Inst. (Changes it to Read)	36
	STO	READ		Read Now into Area Just Punched	37
NOPUN	BPN				38
	BRU	NOPUN		Wait for Punch Cycle	39

CODING SCHEME FOR SIMULTANEOUS READ-PROCESS-PUNCH

WRITTEN BY: _____

Symbol (1–6)	Opr (8–10)	Operand (12–19)	20	Remarks (22–75)	Sequence (76–80)
PUNCH	WCD	128		**Punch Area Just Extended by Calc.	00040
	BRU	BACK		Go Back Read New Card & Calc. Other	41
CONT2	DEC	2		Constant Needed to Increase or Decrease by 2	42
				Notes:	
				Area A = 128	
				Area B = 256	
				Area C = 384	
				Actually: We Read into A	
				Calculate B	
				Punch C	
				Then:	
				Read into C	
				Calculate A	
				Punch B	
				Then:	
				Read into B	
				Calculate C	
				Punch A and then Repeat	

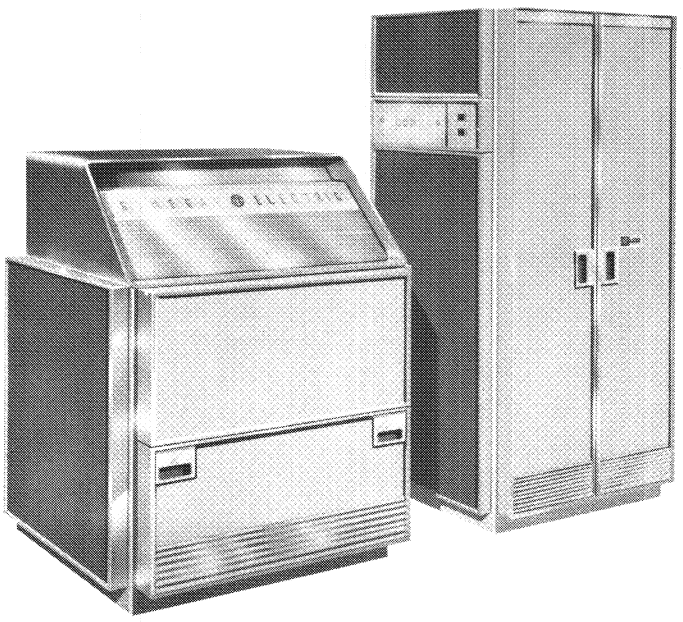


Figure 25 High Speed Printer Sub-System

The Printer receives information directly from the main memory of the computer, through the Data Mating Function, when the PRINT command is specified. The Printer will print the "hard type" equivalent to 10 numeric, 26 alphabetic, and 10 special characters when correctly represented in binary coded decimal form. A single PRINT command will cause printing of up to 120 characters on one line from a block of 40 words of BCD data in memory.

Printer operations may be programmed to overlap or share time with other processing. The printer can be interrogated for "ready" status (completion of previous PRINT commands) by the BDM command. The programmer may transfer control directly back to the BDM command in a "loop", for repeated interrogation, until the printer is ready; or he may transfer control to another part of his program to execute other operations and return at a later time. The printer is not considered as "busy during a slewing operation.

If a number previously calculated within the computer is to be printed, the binary number must first be converted to the binary coded decimal equivalent using the appropriate conversion subroutine from the General Assembly Program library. If the number is a negative number, the sign of the number will be reflected by the insertion of a binary coded decimal hyphen or "-" in the high order position (or low order position at the discretion of the programmer) of the number field so that the number will print as "-xxxxx" (or xxxxx-). An alternate symbol might be chosen as the symbol to represent a negative number in lieu of the (-) hyphen. Alphabetic and alphanumeric data would be carried in BCD from throughout the processing so no conversions are necessary.

Data to be printed is transferred to the printer by either of two modes of the available print commands. Important considerations will arise regarding the appropriate edit of the print data before or during the transfer of the data and the skipping or "slewing" of lines. Several alternatives exist to perform editing and slewing. Editing may be accomplished by any combination of the following methods:

1. Design of records to conform exactly to the design of the print line.

2. Rearrangement of the record and insertion of editing constants to meet the requirements of the print line by programmed changes.

3. Use of automatic format control to provide automatic editing of data as the line is printed.

Slewing or skipping may be accomplished by:

1. Inserting the appropriate control and slew data within the print command.

2. Writing separate slew commands to cause slewing before the line is printed.

3. Wiring a separate slew command to cause slewing after the line is printed.

PRINTING OF DATA ON THE HIGH SPEED PRINTER

BRANCH ON DATA MATING FUNCTION INTERROGATED CONDITIONS

Operation	Operand	Modification Record
BDM	C + T / C / C + F	P

P is the plug number or controller number to be interrogated. C is the number of the specific condition to be tested. Both C and P have a range 0 to 7. C + T calls for branching if the condition tested (C) is true. C + F calls for branching if the condition tested if false.

Condition Number	Condition Tested
0	Printer Controller Busy
1	
2	Out of Paper
3	
4	
5	
6	
7	Any Error in Printer Sub-System

HIGH SPEED PRINTER CONTROL INSTRUCTIONS

All HSP action instructions require three lines of GAP - 225 coding.

Write Print Line

	Operation	Operand	Modification Word
1st line	SEL		P
2nd line	WPL	M_2	F
3rd line	WPL	M_1	N

P is the plug number (0 thru 7) to which the on line printer is attached. WPL is the mnemonic code for Write Print Line. M_1 is the memory "address" of the first data word in the line of 40 (maximum) data words to be printed. Data words to be printed consist of three BCD characters each. If less than 40 words are to be printed on one line, the sign bit must be "on" in the last word to be printed. F is the format control indicator. If a blank is written in the F position, the line is to be printed without horizontal format control and M_2 is ignored. If an F is written in the F position, horizontal format control words starting at memory address M_2 are used to control the printing of the data words. N is the numeric print indicator. If a blank is written in the N position, the data words to be printed are alphanumeric. If an N is written in the N position the data words to be printed consist only of decimal numbers and the 14 special symbols. Both M_2 and M_1 must be in the same half of a 16K memory. After printing the paper is automatically spaced one line. Spacing of 0 to 63 lines, or ejecting the paper to the top of the next page may be coded as part of the WPL command by coding lines 2 and 3 in Octal.

1 = PRINT and slew
0 = Slew only

	S	1	2	3	4	5	6 - 19
2nd line:	1	1 = Format 0 = No Format	V1	V2	V3	1 = Numeric 0 = Alpha-numeric	Format Address
3rd line:	V4	V5	V6	C1	C2	Data Address	

bit 5 is also part of the format address and is assumed to be the same as data address bit 5.

If C1 = 0 and C2 = 1, ignore V1 thru V6, and slew paper to top of next page.

If C1=1 and C2=1, slew paper the number of lines (0 thru 63) indicated by the binary number in positions V1 thru V6.

Slew Paper a Fixed Number of Lines

	Operation	Operand	Modification Word
1st line:	SEL		P
2nd line:	SLW	N	
3rd line:	SLW		

P is the plug number (0 thru 7) to which the on line printer is attached. SLW is the mnemonic code for SLEW PAPER. N is the number (0 thru 63) of lines to be spaced or slewed before printing the next line.

Slew Paper to Top of Next Page

	Operation	Operand	Modification Word
1st line:	SEL		P
2nd line:	SLT		
3rd line:	SLT		

P is the plug number (0 thru 7) to which the on line printer is attached. SLT is the mnemonic code for slew paper to top of next page.

EXAMPLES

1. Assume that the data to form one line of print occupies 23 consecutive storage locations at 2025 - 2047. Assume that the printer is plugged into data mating hub # 6. Write the commands to print only 23 words of data on the high speed printer.

Assume the data to be printed is already in the proper format. Assume the data to be printed is alphanumeric data.

```
3000    LDA     2047
3001    BMI
3002    BRU     3004        } Set sign of 2047 to -
3003    CHS
3002    STA     2047
3004    BDM     O + T   6   } Delay until ready.
3005    BRU     3004
3006    SEL             6
3007    WPL                 } Print alphanumeric data
3008    WPL     2025
```

The commands at 3000 - 3003 will "set" the sign bit of the 24th word to a 1. The commands at 3004 and 3005 will delay processing until the printer is ready. The commands at 3006 - 3008 will cause 23 words of alphanumeric data to be printed on the high speed printer. After printing, the paper is automatically spaced one line.

2. In problem 1 above assume that all of the data to be printed consists of either numeric data or any of the 14 special symbols.

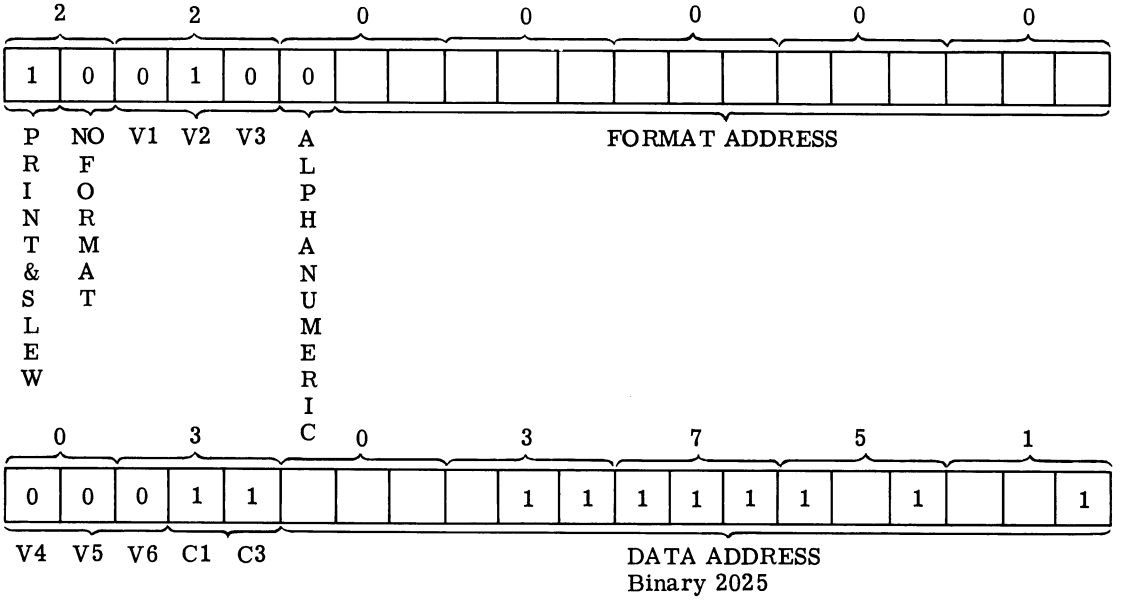
```
3007     WPL     2025     N.
```

All commands will be identical except the command at 3007 which will include an N. in the position normally reserved for the modification word.

3. Assume in the above example that the paper is to be "Slewed" 16 lines after the line is printed. Write the command to "slew" paper as part of the command to print.

```
3006     SEL              6
3007     OCT     22 0 00 00      } BINARY CONSTANT
3008     OCT     03 0 37 51
```

All commands will be identical except the programmer will substitute the two constants above at 3007 and 3008 as indicated below



The binary constant at V1 V2 V3 V4 V5 and V6 will represent the number of lines to be slewed(16), after the line is printed. In the above examples the address 2025 is considered as the "actual" or absolute address in the "object" program of the line to be printed. This address may be specified for the first word to be printed by means of an "ORG" operation.

4. Assume that the paper is to be "slewed" 16 lines after the line is printed. Write the command to "slew" paper as a separate command.

```
3009     SEL     6
3010     SLW     16      } Slew paper 16 lines.
3011     SLW
```

The commands above will cause the paper to "skip" 16 lines after the line is printed. The commands above could cause slewing before the line is printed when they precede the commands to print (WPL) and follow the commands to interrogate the plug (BDM) for "BUSY".

5. Write the commands to cause the paper to skip to the first line on the next page

```
3009    SEL     2   }
3010    SLT         }   SLEW TO FIRST PRINTING LINE
3011    SLT         }
```

The commands above will cause the paper to skip to the first printing line of the next page.

PROGRAMMED EDITING OF DATA FOR OUTPUT

Or A into Y	ORY
Normalize A register	NOR
Branch if Modification Word is High	BXH
No operation	NOP

ORY Y OR A INTO Y

Each bit of A is examined. If there is a 1 in A in a given position, a 1 is placed in Y in that position. The contents of A are unchanged.

NOR K NORMALIZE A REGISTER

If R, the number of leading zeros of A (1-19), is less than K, the contents of A (1-19) are shifted left R places, and K-R replaces the contents of location 0000 (15-19). If R is greater than, or equal to, K, the contents of A (1-19) are shifted left K places, and a zero replaces the contents of location 0000 (15-19). Positions S, 1-14 of location 0000 are always set to zero. Vacated positions of A are filled with zeros. The sign of A is unchanged.

BXH K,X BRANCH IF X IS HIGH

If the contents of X (7-19) are greater than or equal to K, the computer takes the next sequental instruction if the contents of X (7-19) are less than K, the computer skips the next instruction and executes the second sequential instruction. The contents of X are not changed. This instruction is not automatically modified.

NOP NO OPERATION

Zero is added to the contents of A (s, 1-19).

EXAMPLES

1. Assume an 11 digit binary coded decimal number representing dollars and cents is in memory storage beginning in word 0256 and ending in word 0259. Assume that a decimal point is actually represented by the proper BCD configuration within the 12 digit field. Write the necessary commands to delete insignificant zeros to the left of the decimal point. The number in memory is represented as:

XXX	XXX	XXX	.XX
0256	0257	0258	0259

```
3996   DEC   19  }   00000000   000000   010011
3997   ALF       }   00110000   110000   110000
3998   DEC   6   }   00000      00000    00000    00110
3999   DEC   7   }   00000      00000    00000    00111

3148   LDZ              }
3149   STA   0001       }  Clear Modification Word 1
3150   LDA   0256   1   }
3151   NOR   19         }  Set up A Register for Zero Test
3152   LDA   0000       }
3153   BZE              }
3154   BRU   3164       }  Test and Branch for 3 Zeros
3155   LDA   3996       }
3156   SUB   0000       }  Convert from 19 Complement
3157   SUB   3999       }
3158   BMI              }  Test and Branch for No Zeros.
3159   BRU   3179       }
```

```
3160    SUB     3998
3161    BPL                     } Test and Branch for 2 Zeros
3162    BRU     3170
3163    BRU     3175            Branch for 1 Zero
3164    LDA     3997
3165    ORY     0256    1
3166    INX     1       1       } Insert 3 Blanks in Word and Test for Return
3167    BXL     3       1
3168    BRU     3150
3169    BRU     3179            Continue
3170    LDA     3997
3171    SRA     6
3172    SLA     6               } Insert 2 Blanks in Word
3173    ORY     0256    1
3174    BRU     3179
3175    LDA     3997
3176    SRA     12
3177    SLA     12              } Insert 1 Bland
3178    ORY     0256    1
```

The commands in storage at 3148 and 3149 will replace the contents of modification word 1 with zeros. The commands in storage at 3150 and 3151 will insert a number within location 0000, as follows. The value inserted within 0000 will be the binary equivalent of 19 minus the number of high order zero bits in the A register: The A register is then loaded with the number in storage at 0000 and tested for zero by the commands at 3152, 3153 and 3154.

If the contents of the A register are equal to zero, the number of positions shifted by the NOR command would be equivalent to 19; and, therefore, control is transferred to the command at 3164 to insert 3 spaces (blanks) within the word to be edited. The command at 3164 will load 3 spaces (blanks) into the A register. The ORY in storage at 3165 will cause the contents of 0256 to be replaced with binary coded decimal spaces (blanks). Control will be returned to "edit" the next word or continue in sequence if all 4 words have been "edited" by the commands in storage at 3166 through 3169.

If the word in storage does not contain the equivalent of all zeros, control will continue with the command in 3155. The commands in 3155 and 3156 will load a "binary 19" into the A register, subtract the contents of 0000 from the "binary 19", and produce a number within the A register which is equivalent to the number of 0 bits shifted out of the high order positions of the word by the NOR command at 3151. The commands in storage at 3157, 3158 and 3159 will test the contents of the A register to determine whether the NOR command executed 7 or more shifts. If the NOR command does not complete 7 or more shifts, there are no blanks for insertion in the high order positions; and control will be transferred to the command at 3179 to continue the program.

The commands in storage at 3160 through 3162 will further test the number of shifts executed by the NOR command. If the NOR command executed 13 or more shifts, control will be transferred to the command at 3170. The commands at 3170 through 3173 will replace the contents of the word to be edited with 2 blanks in the high order positions of the word and transfer control to continue processing. The instructions for the insertion of spaces (blanks) utilize the LDA command to load a word consisting of 3 BCD blanks, the SRA and SLA commands to delete an appropriate number of blanks, and the ORY command to insert the blanks to replace the insignificant BCD zeros.

If the word to be edited does not contain 3 BCD zeros, no zeros, or 2 BCD zeros, it must contain 1 BCD zero in the high order position. Control in this case will be transferred to the commands at 3175 through 3178 for the insertion of 1 blank.

The above routine for insertion of blanks to the left of the decimal point in place

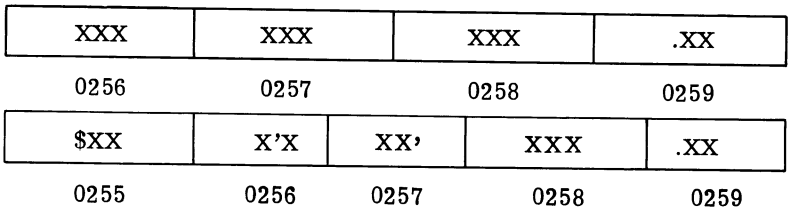
of insignificant BCD zeros may be utilized as a subroutine package. Review material on Programming for Subroutine Usage.

2. In the previous example, after zero suppression takes place, complete editing by the insertion of appropriate commas and a dollar sign. The number in storage after the insertion of a dollar sign and commas may occupy 5 words of memory. Let these memory locations be 0255, 0256, 0257, 0258 and 0259.

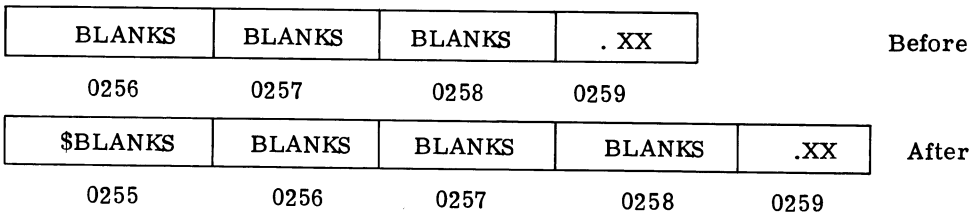
```
3994    ALF     00,     0       000000  000000  111011  Additional
3995    ALF     $▲▲     0       101011  110000  110000  Constants

3179    NOP
3180    BXH     3       1   }   Edit for 3 H O. Words of Blanks
3181    BRU     3211
3182    BXH     2       1   }
3183    BRU     3211            Edit for 2 H.O. Words of Blanks
3184    BXH     1       1   }
3185    BRU     3214            Edit for 1 H.O. Words of Blanks

                                Control continues to 3186 when no H.O. words
                                of Blanks

3186    LDA     3995    }
3187    SRA     12      }
3188    SLA     12      }   Load $ Symbol and insert in left hand
3189    ORY     0255    }   position of 0255.
3190    DLD     0256    }
3191    SRD     6       }   Insert 2 H.O.P. of 0256 into L.O.P. of 0255.
3192    ORY     0255    }
3193    LDZ             }
3194    STA     0256    }   Zero out 0256
3195    SLD     6       }
3196    SLA     12      }   Insert 1 L.O.P. of 0256 into 1 H.O.P. of 0256.
3197    ORY     0256    }
3198    LDA     3994    }
3199    SLD     6       }   Insert comma into middle position 0256 and
3200    ORY     0256    }   H.O. position of 0257 into L.O. 0256.
3201    LDA     0257    }
3202    MAQ             }   Load 0257 and preserve in Q.
3203    STA     0257    }
3204    LAQ             }   Zero out 0257. Reload A with contents of 0257.
3205    SLA     7       }
3206    SRA     1       }   Insert 2 L.O.P. of 0257 into 2 H.O.P. of 0257.
3207    ORY     0257    }
3208    LDA     3994    }
3209    ORY     0257    }   Insert comma into L.O.P. of 0257.
3210    BRU     3219        Transfer to continue processing.
3211    LDA     3995    }
3212    STA     0255    }   Load $ sign and 2 blanks into 0255.
3213    BRU     3219        Transfer control to continue processing.
3214    LDA     3995    }
3215    STA     0255    }   Load $ symbol and insert in left hand
3216    LDA     0257    }   position of 0255.
3217    MAQ             }
3218    BRU     3199        Load 0257 and preserve in Q.
3219
```

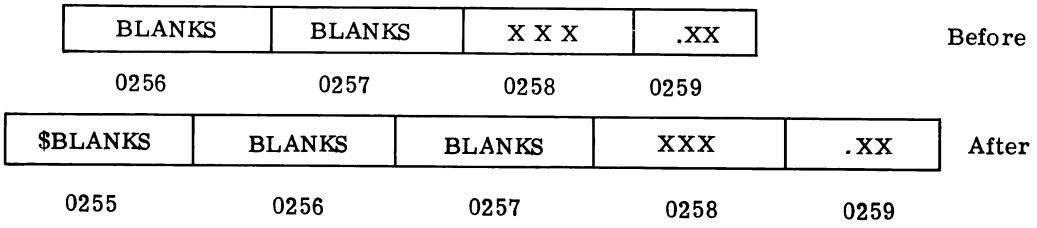
A dollar sign with 2 blanks in the low order positions (L.O.P.) will be in storage at 3995. A comma with zeros in the high order positions (H.O.P.) will be in storage at 3994. The commands at 3180 through 3185 will test modification word 1 to determine the number of high order words in the number which contain all blanks after the edit routine in problem 1 is executed. If no high order words in the number contain all blanks, control will transfer to the command at 3186

The commands in storage at 3186 through 3210 will insert a dollar sign into the high order position of location 0255 and appropriately shift the number, with insertion of commas where necessary, as indicated below.

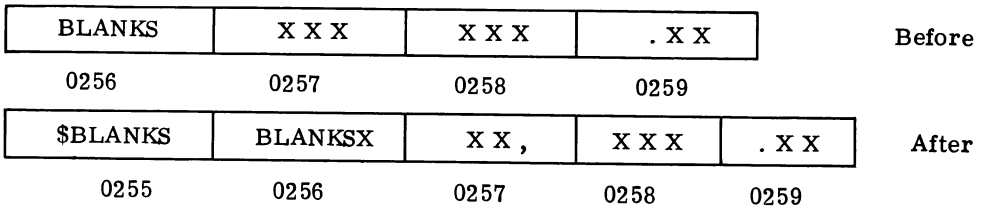


The commands in storage at 3211 through 3213 will insert a dollar sign into the high order position of location 0255 as indicated below.





The commands in storage at 3214 through 3218 and 3199 through 3210 will insert a dollar sign into the high order position of location 0255 and appropriately shift the number, with insertion of commas where necessary, as indicated below:



As mentioned in the previous example, this coding might be used as part of an editing subroutine package. The use of the NOP (No Operation) command is of particular interest here. At the time this example was prepared another instruction was written in 3179, which is the natural continuation of example 1. A check of the coding revealed that this instruction was not correctly placed and would have to be deleted. Something therefore had to be done with memory location 3179. To have the coding in example 1 jump to 3180 to continue or to have moved the coding for example 2 up on memory location would have necessitated several address changes with attendant possibilities for clerical error. The NOP command is a handy way to fill in such gaps in a program.

EDITING OF DATA FOR OUTPUT WITH AUTOMATIC FORMAT CONTROL

If a line is to be printed under a format control, the format data is stored in the Central Computer Memory under the same organization as the print line data. The format control data consists of:

a. Any printable character.
b. Special control characters.

The Printer Controller, in assembling a formatted line, reads in from the Central Computer Memory one word of data and one word of format. The first format character is considered initially. If it is a printable character the character is printed. If it is a special control character it is treated as described below. Assuming it was a printable character, it is printed, and the first data character is considered. If it is a printable character it is printed. It may be a special control character, in which case it is treated as explained below. In sequence, the second format, then second data characters are considered, followed by the third format and the third data characters. Following the consideration of the third data character another word of data and another word of format are requested from the Central Computer Memory. Upon receiving these new words, the procedure described above is again followed. This routine is continued until a one is the sign bit of a data word is encountered, whereupon, after consideration of that data word and its respective format word, the sequence is ended.

There are five special control characters, mentioned above, which are available for controlling the format of the printed line. These characters, and their BCD bit representations are:

Ignore	Octal 35
Ignore/Skip	Octal 36
Delete	Octal 37
Delete/Skip	Octal 56
Zero Supress*	Octal 57

*Note: Zero suppression is also incorporated when a printable $ symbol appears within the format data.

If a format character is an Ignore, the next data character is immediately considered.

If a format character is an Ignore/Skip, a blank is printed and the next data character is considered.

If a format character is a Delete, the next data character is ignored, and the next character considered is the next format character.

If a format character is a Delete/Skip, a blank is printed, the next data character is ignored, and the next character considered is the next format character.

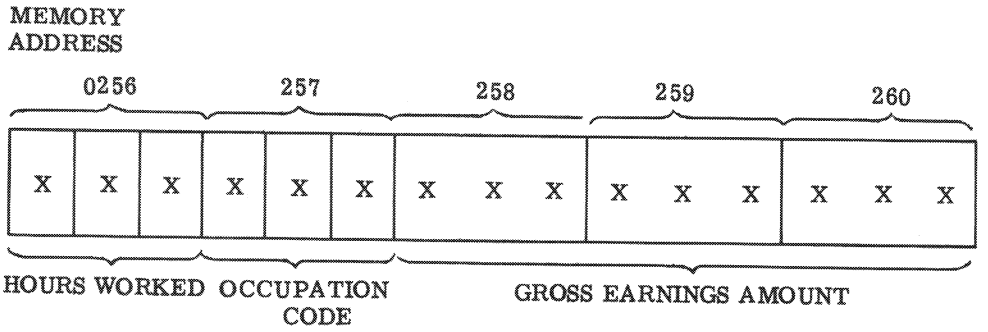
If a format character is a Zero Suppress, the next data character is ignored and the next format character is printed if it is a printable character. After considering this last format character blanks will be inserted in the print line until; (a) a non-zero data character is detected, or (b) a period comes up in the format data. It should be noted that once a Zero Suppress has been put into effect the print line data is inspected only for a non-zero character. The format data is inspected only for a period. Zero suppression is also incorporated when a printable dollar sign appears within the format data.

It is possible for an Ignore or an Ignore/Skip character to be placed in the print line data (as well as the format control data). If a data character is an Ignore, the next format character is immediately considered and nothing is printed for that data character. If a data character is an Ignore/Skip a blank is printed and the next format character is considered.

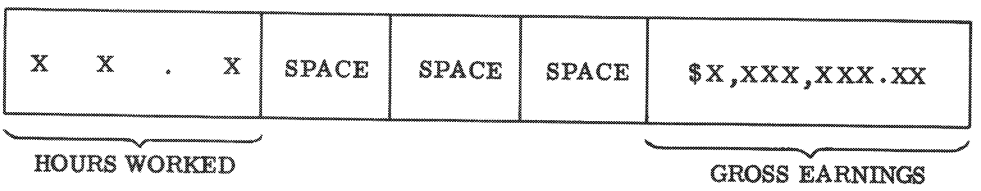
The above procedure makes it possible for a line format to be stored in the Central Computer Memory once, and be used as often as needed to print lines of data in that format. The data may, within the limitations imposed by the use of the special control as described above, be stored in sequence in computer memory, the Printer Controller automatically constructing the print line according to the prescribed format.

EXAMPLES

1. Assume that 4 words of data represent the following information in storage at 0256, 0257, 0258, 0259 and 0260 represented as BCD data.

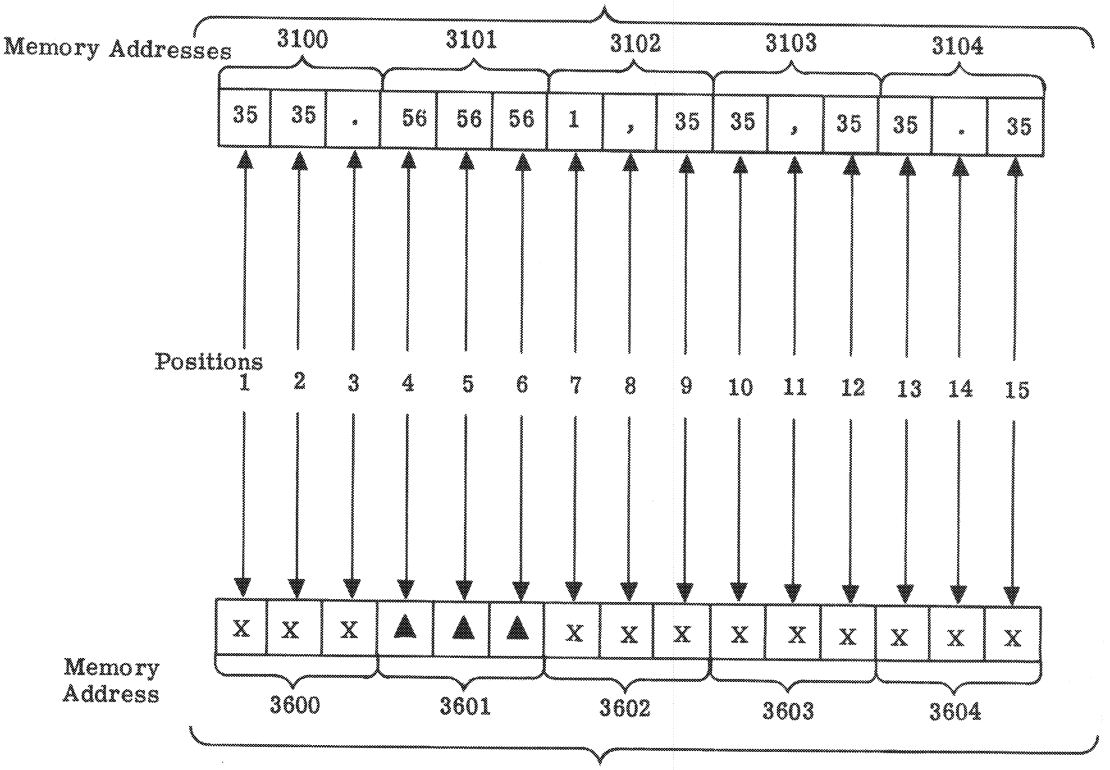


Assume that it is desired to print 4 words of the above data as follows:

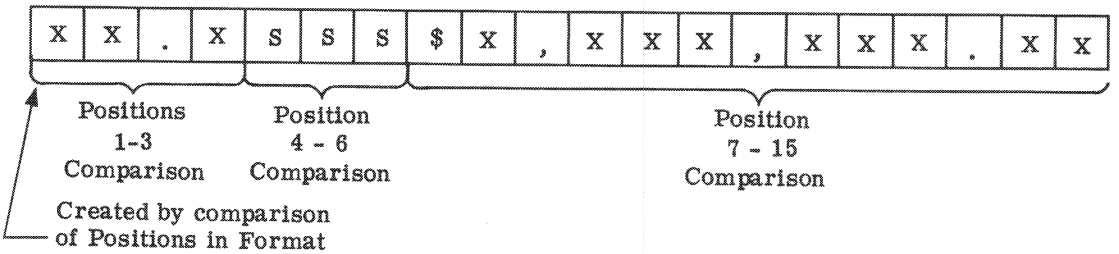


Assume that insignificant blanks and zeros must be deleted from the gross earnings field, before it is printed. Arrange the data, design the format control data, and write the commands to write the data under automatic format control. Assume that the origin of the format data will appear in storage at 3100. Assume that the origin of the data before it is printed will appear in storage at 3600.

FORMAT DATA



PRINT DATA
LINE IMAGE TRANSFERRED AND PRINTED.



KEY

▲	ANY BIT CONFIGURATION PERMISSABLE
S	SPACE
X	ANY ALPHANUMERIC DATA CHARACTER
35	OCTAL 35 FOR IGNORE
56	OCTAL 56 FOR DELETE/SKIP
57	OCTAL 57 FOR ZERO SUPPRESS

NOTE: Zero suppression is also incorporated when a printable dollar sign appears within the format data.

```
2099    ORG     3100                    Origin of Format Data
3100    OCT     035 35 33   }
3101    OCT     056 56 56   |
3102    OCT     053 73 35   }           FORMAT DATA
3103    OCT     035 73 35   |
3104    OCT     035 33 35   }

3501    DLD     0256        }           Insert words 0256 and 0257 in 3600 and 3601.
3502    DST     3600        }
3503    DLD     258         }           Insert 5 characters in 3602 and 3603
3504    DST     3602        }
3505    LDA     0260        }           Insert 4 characters 3604 and 3605
3506    STA     3604        }
3507    BDM     O + T    6  }
3508    BRU     35       7  |
3509    SEL              6  }           Print line under automatic format control.
3510    WPL     3100     F  |
3511    WPL     3600        }
```

The pseudo commands at 2099 through 3104 cause the necessary format data to be established at 3100. The commands at 3501 through 3506 "set" up data words 3600, 3601, 3602, 3603, and 3604, for printing. The commands at 3507 through 3511 will print the line under automatic format control. Automatic format control will cause the line to be printed as described above by comparison of each format data position and each print data position sequentially. Each comparison causes the transfer of the desired character to build the line image as described above.

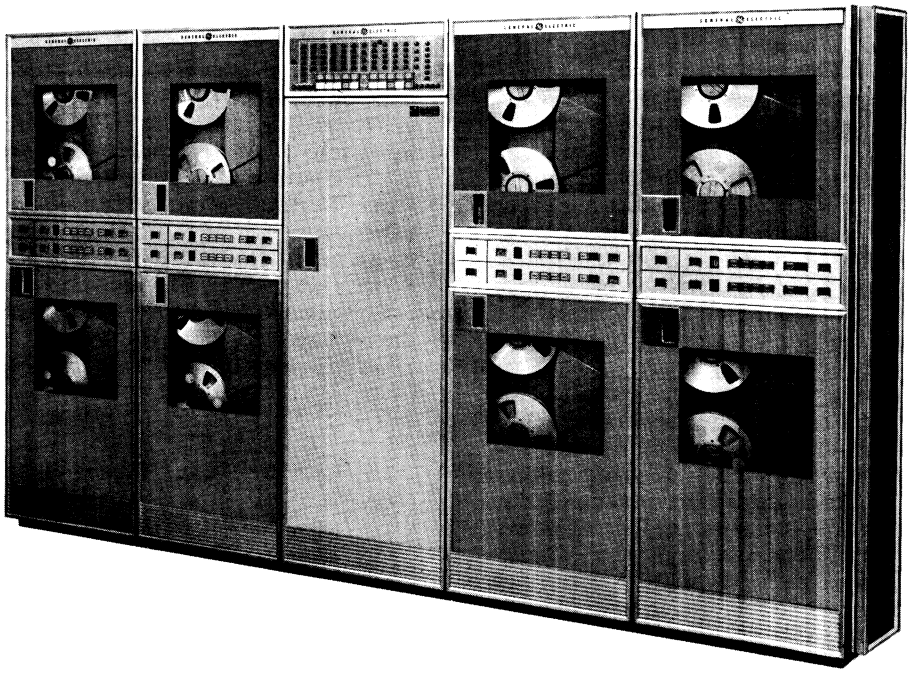


Figure 26 Magnetic Tape Sub-Systems

MAGNETIC TAPE OPERATIONS

Before beginning this discussion it would be well to consider a few fundamental terms normally used in any type of data processing. These are:

Record

A collection of facts peculiar to an identifiable item. Examples are: an employee's pay record; a customer's account containing the date and amount of payments made relative to an obligation; and, an inventory card containing a balance on hand for a specific stock number.

File

A file may be referred to as an accumulation of related records. For instance, in any payroll system a record is maintained for each employee in the company. The record will contain various data such as the individual's name, pay number, rate of pay, type of employee (direct or indirect), the number of withholding exemptions, and other types of deductions to be made from the employee's gross pay at each pay period. An accumulation of all payroll records for a company is termed a "payroll file".

Files are comprised of many forms of information. The example given above may also be termed a master file. The accumulation of all stock withdrawal cards for a specified period of time to be processed against a stores master file for the purpose of updating balances on hand is also referred to as a file - a stores transaction file.

Key

In order to process a transaction file against a master file, it is necessary to have some identification common to the master data and the related transaction. This may be accomplished through the use of a "key" which produces a zero when the transaction key is subtracted from the master key. Examples of keys are: stock number, pay number, cost center, and job level.

The basic use of magnetic tape is to serve as a file storage medium. The prime benefit derived from this type of storage is the rapidity with which information can be placed on the retrieved from the tape surface. Storage may be "on-line" where the tape is utilized as auxiliary memory to the Central Processor, but more commonly it will be "off-line" where one or more reels are used to hold complete master files and transaction files. Master file and transaction tapes usually are sequentially ordered by the record key (e.g., job number, pay number) in order to take advantage of the fast sequential record access associated with tape reading and to facilitate processing within the computer. An exception to the sequential requirement is the use of tapes in conjunction with a random access memory device. In this latter case the magnetic tape is used chiefly as a compact storage medium for vast amounts of data.

Sorting of transaction data may be done prior to conversion to magnetic tape. For instance, punched cards may be sorted in ascending sequence before

placing their contents on tape. Under other circumstances it may be more desirable to sort transactions within the computer. (This is an example of the use of "on-line" temporary tape storage).

Information is placed on magnetic tape in groups of words referred to as records or blocks. A record may consist of only one word, or it may be as large as the entire computer memory. Each record or block is separated by a 3/4 inch gap of erased tape which permits starting and stopping between records. It is extremely important for the programmer to realize that this use of the term "record" is not the same as that previously defined. It is true that the size of a magnetic tape record may be varied at the option of the programmer, but a "magnetic tape record" would not normally be made the same size as the "data record". To do so would unnecessarily waste tape space with an excessive number of inter-record gaps (3/4 inch of tape each). In normal data processing several data records will be contained in each magnetic tape record. In order to avoid this confusion of terms, magnetic tape records might be consistently referred to as "blocks", or data records might be consistently referred to as "items". Unfortunately, "record" is a very popular and traditional term for both usages.

The use of magnetic tapes implies files of great size. Therefore, in many programs a file will consist of more than one reel of tape. When the end of a tape or the end of a file is reached, the tape reel must be rewound, removed and replaced. To maintain the continuity of the running program it is very worthwhile to program an immediate switch, or alternation, from the just completed tape to the succeeding tape (already mounted on another handler). This technique permits the mechanical rewind and manual removal of the completed tape to proceed independently of further processing. If sufficient tape units are not available to permit convenient switching for all files, the most extensive files should be given priority in the allotment of tape handlers.

Tapes used with the GE 225 contain a silver spot to signal the physical end of the tape. When detected by a photo-electric cell within the tape unit, an indicator is set. The condition of the indicator should be tested by programmed instructions after reading or writing each record. If the indicator is not set, normal processing will continue; if set, an end of tape branch will jump into program specified subroutines - normally the rewinding of the current reel and alternation to a new reel. The end of file sentinel is the magnetic representation of the binary code 0001111 preceded by an erased section of the tape 3 3/4 inches long. During magnetic tape operations several other exceptional conditions may occur which are secondary to the main processing job. A list of all such conditions is as follows:

1. Controller Busy
2. End of File
3. End of Tape
4. Tape Rewinding
5. Parity Error
6. Input-output Buffer Error
7. Modulo 3 or Modulo 4 Error
8. Any Error

Handling of these exceptional conditions may be conveniently assigned to "executive routines". The executive routines will be provided to magnetic tape users of the GE 225 as part of standard input-output packages that both detect and correct the above conditions, when they occur, according to standard operating conventions.

In order to process a computer run we must know the data contained in each tape. Is it the master file? Is it the proper transaction file? To tell us this, exterior and interior labels are used. Exterior labels are notations placed on the container relative to the tape data. Interior labels are recorded magnetically on the tape for positive identification and are tested by the computer program. Interior labels are the most important. They are permanent, and reliably establish the tape content. Each input or output tape should contain this identification. It is properly placed in the first record of the tape and should contain the date, identification, and reel number. These data should be checked by the computer program before any further processing ensues.

TRANSFERS OF DATA BETWEEN THE MAGNETIC TAPE UNITS AND THE CENTRAL PROCESSOR

BRANCH ON DATA MATING FUNCTION INTERROGATED CONDITIONS

$$\text{BDM } C + \begin{matrix} T \\ P \end{matrix} \quad P.$$

P is the plug number or controller number to be interrogated. C is the number of the specific condition to be tested. Both C and P have the range 0 to 7. C + T calls for branching if the condition tested (C) is true. C + F calls for branching if the condition tested is false.

Condition Number	Magnetic Tapes
0	Controller Busy
1	End of File
2	End of Tape
3	Any tape rewinding
4	Parity Error
5	Input/output Buffer Error
6	Mod 3 or Mod 4 Error
7	Any Error

MAGNETIC TAPE CONTROL INSTRUCTIONS

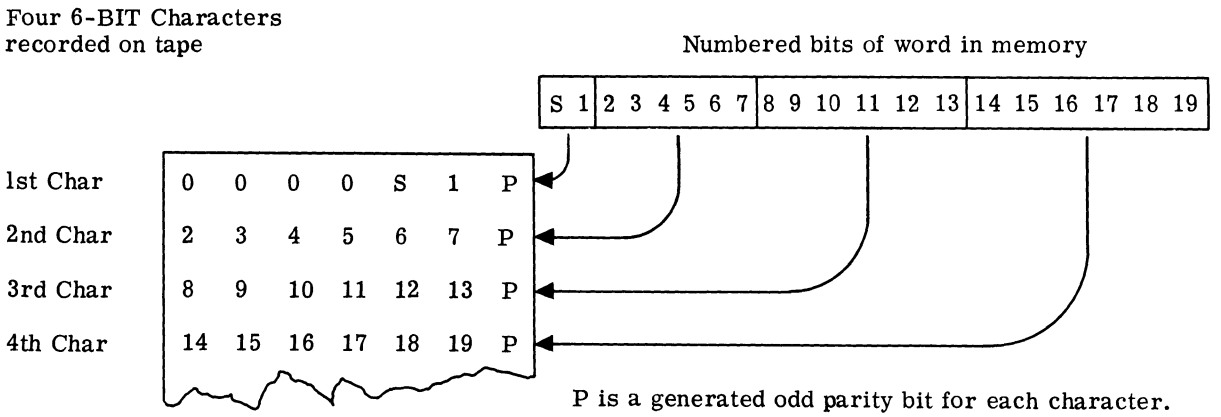
All magnetic tape movement instructions require three lines of GAP-225 coding.

Operation	Operand	Modification
1st line: SEL		P
2nd line: XXX	M	
3rd line: XXX	N	T

P is the plug number (0 thru 7) of the magnetic tape controller to be selected for this magnetic tape instruction. M is the memory address (decimal number of alphabetic symbol) of the first word of a block of N + 1 words to receive data from magmetic tape on tape read instructions. On tape write instructions, M is the address of the first word of a block of N words to be written on magnetic tape. N is the maximum number of words to be read or the exact number of words to be written. T is the number of the Tape Unit (0 thru 7) to be activated on plug P. XXX is the mnemonic code for the specific tape movement desired. The mnemonic codes for specific tape movements are:

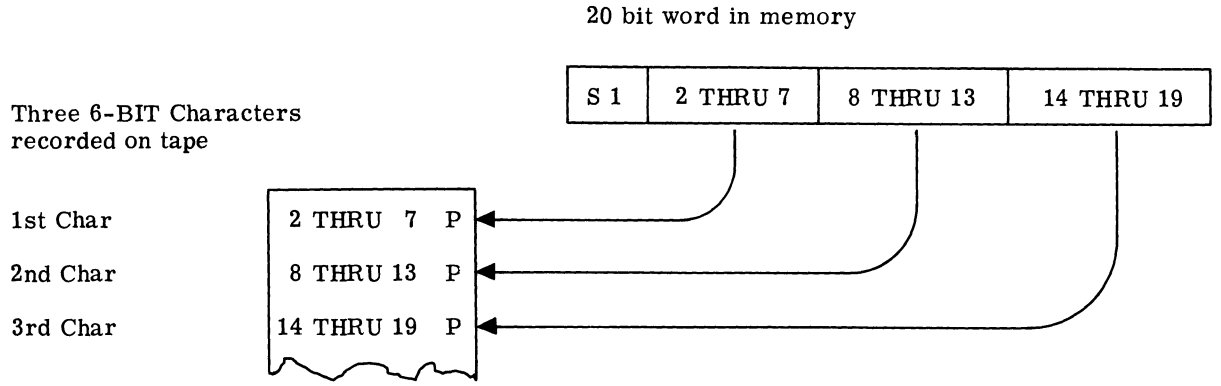
WTD	Write tape in decimal mode
WTB	Write tape in binary mode
RTD	Read tape in decimal mode
RTB	Read tape in binary mode
RWD	Rewind to tape to leader
BKW	Backspace one record and position WRITE Head
BKR	Backspace one record and position READ Head
WEF	Write END of FILE

When writing on magnetic tape in binary mode, one word of memory becomes four 6-bit characters on tape as shown below:



P is a generated odd parity bit for each character.

When writing on magnetic tape in decimal mode, one word in memory becomes three 7 - bit characters on tape as shown below, but some of the 6 - bit patterns are altered to conform to the IBM 727 tape binary codes for BCD and alphebetic characters. The alteration of the character codes when writing and reading magnetic tape is automatic.



P is a generated even parity bit for each character.

Bits S and 1 are not recorded on tape in decimal mode. Writing mixed binary and BCD words on tape must be done in the binary mode When reading tapes in the decimal mode, bits S and 1 are set to zero in memory for each word read from tape.

After reading N words from magnetic tape into memory starting at location M, (in either binary or decimal mode) memory location M + N will contain certain zeros indicating that exactly N words were read from a record on tape containing N words. If the number of words contained in the record currently read is less than N, then only the contents of the record will be stored in memory and the 2's complement of the residue (N-record length) will be stored in memory cell M + N with a one-bit in the sign position. If the number of words in the record is greater than N, then only N words will be stored in memory and the increment (record length-N) will be stored in memory call M + N with a zero in the sign position. M is not indexable.

EXAMPLES

1. Read a record from tape unit 1, assume the controller is plugged into Data Mating hub #4. Assume that records on magnetic tape are variable in length from 0 to 50 words. However, all data needed from processing is contained within the first 10 words of any record. The record is to be read into storage beginning at 1000. Assume mixed binary and binary coded decimal data has previously been written on the tape in binary mode.

```
0600    BDM     0 + T    4   }
0601    BRU     0600         }   Delay until controller ready
0602    SEL              4   }
0603    RTB     1000         }   Read up to 10 words in Binary mode
0604    RTB     10       1   }
0605    BDM     0 + T    4   }
0606    BRU     0605         }   Delay until ready
```

The commands at 0600 and 0601 will delay processing until the controller is not busy. The commands at 0602, 0603 and 0604 will read up to 10 words from the next record on magnetic tape. The commands at 0605 and 0606 will delay processing until the record is completely read. Mixed binary and binary coded decimal data may appear in a magnetic tape record when the record is both written and read in the binary mode. Therefore, binary data may be calculated freely without a need to convert before the calculation, and binary coded decimal data will also retain its identity as binary coded decimal data.

2. After the record in problem 1, above, is read, interrogate the word at 1010 to determine whether the record contained more than 50 words. If the record contains more than 50 words, continue further processing at 3000.

```
3998    DEC     40

0607    LDA     1010     }   Load 1010
0608    BMI              }   Transfer if less than 10 read
0609    BRU     0613     }
0610    SUB     3998     }   Test and Transfer for more
0611    BPL              }   than 50 words
0612    BRU     3000     }
0613                     }
```

The word after the last word read from the block will contain a 1 in the sign bit position if the record read contained less than 10 words. Therefore, the commands at 0608 and 0609 will cause further processing to continue at 0613 if the record just read contained less than 10 words. If the record just read contained more than 50 words of data, the commands at 0610, 0611 and 0612 will cause further processing to continue at 3000.

3. When the record in problem 2 above contains more than 50 words, write the commands to backspace the tape unit and read up to 100 words from the record. Load the number of words in the block into the A register and "halt" the computer at "BRU 3012".

```
3998    DEC     100

3000    BDM     0 + T    4   }   Delay until controller ready
3001    BRU     3000         }
3002    SEL              4   }
3003    BKR                  }   Backspace and position to read
3004    BKR              1   }
3005    SEL              4   }
3006    RTB     1000         }   Read up to 100 words into storage
3007    RTB     100      1   }
3008    BDM     0 + T    4   }
3009    BRU     3008         }
3010    LDA     1100         }   Load and convert to no. of words in
3011    ADD     3998         }   block Programmer halt no. of words
3012    BRU     3012         }   in block in A
```

The commands at 3000 and 3001 will delay the controller until it is ready. The commands at 3002, 3003 and 3004 will cause the tape unit to backspace 1 block and position to read the block. The commands at 3005, 3006 and 3007 will cause up to 100 words of the record to be read. The commands at 3010, 3011 and 3012

will cause the number of words in the block to appear in the A register and a halt due to a programmed loop. Further processing can be initiated from the console.

4. Write a command to rewind the tape to the leader.

```
3001    BDM     0 + T    4   }
3002    BRU     3011         } Delay until controller ready
3013    SEL              4   }
3014    RWD                  } Rewind tape to leader
3015    RWD              1   }
```

The commands above will rewind the tape to the leader.

5 Assume that data concerning one item has been recorded in 2 records on tape unit 2, on a controller on plug 4. Also, assume that pertinent data is contained within the second record. Write the commands to "Skip" the first record and read the second record into 1000.

```
2500    BDM     0 + T    4   }
2501    BRU     2500         } Delay until controller ready
2502    SEL              4   }
2503    RTB     0            } Skip Record
2504    RTB     0        2   }
2505    BDM     0 + T    4   }
2506    BRU     2505         } Delay until ready
2507    SEL              4   }
2508    RTD     1000         } Read second record
2509    RTD     50       2   }
```

The commands in 2500 and 2501 will delay processing until the controller is ready. The commands at 2502, 2503 and 2504 will "skip" a record on magnetic tape. The commands at 2505 and 2506 will delay processing until the controller is ready. The commands at 2507, 2508 and 2509 will read the next record from magnetic tape into storage.

6. Assume the input tape is tape unit 1 on a controller on plug 4. Assume the output tape is tape unit 1 on a controller on plug 5. Write the commands to read 50 words into storage from the input tape beginning at 1000 and to write 50 words on the output tape. Interrogate the "End-of-File" indicator on the input tape. If set, write the end-of-file on the output tape and rewind all tapes.

```
2600    BDM     0 + T    4   }
2601    BRU     2613         } Interrogate and transfer for input
                             } end-of-file
2602    BDM     0 + T    4   }
2603    BRU     2602         } Delay for input tape busy
2604    SEL              4   }
2605    RTD     1000         } Read record from input tape
2606    RTD     50       1   }
2607    BDM     0 + T    4   }
2608    BRU     2607         } Delay until record is read
2609    SEL              5   }
2610    WTD     1000     1   } Write record on output tape
2611    WTD     50       1   }
2612    BRU     2600         Return and repeat
2613    BDM     0 + T    5   }
2614    BRU     2613         } Delay until output ready
2615    SEL              5   }
2616    WEF                  } Write end of file
2617    WEF              1   }
2618    BDM     0 + T    5   }
2619    BRU     2618         } Delay until end of file written
```

2620	SEL		5	Rewind output
2621	RWD			
2622	RWD		1	
2623	BDM	0 + T	4	Delay for input tape busy
2624	BRU	2623		
2625	SEL		4	Rewind input
2626	RWD			
2627	RWD		1	
2628	BRU	2628		Halt

The commands at 2600 and 2601 will interrogate the input tape file to determine whether the last record on the input tape has been previously processed. If the last record has been previously processed, further processing will be continued with the command at 2613. The commands beginning at 2613 will initiate the necessary steps to "close" the files and terminate processing i.e. 2613-2617 will write the end-of-file record on the output tape, 2618-2622 will rewind the output tape, 2623-2627 will rewind the input and the command at 2628 will "Halt" further processing until further action is initiated from the control console. It is important to consider the necessity for writing the delay commands when initiating action on the individual tape units. When information is transferred from or to a tape unit, the controller or data mating plug is "Busy" during the transfer. All operations affecting any tape unit on the same controller must be delayed until the transfer is completed (by use of the appropriate BDM and BRU commands). When rewinding tapes on the same controller the "rewind" controller is "busy" for 250 us, therefore "rewind" commands in a series should be preceded by appropriate BDM and BRU commands. As a matter of policy it is appropriate to precede all operations which affect magnetic tape with appropriate BDM and BRU commands. Therefore, the commands to rewind the "input" magnetic tape (2625-2627) are preceded with the commands to "delay" even though the delay should have been included in the previous processing sequences.

The commands at 2602 and 2612 will read records of 50 words in length from the "input" magnetic tape and write the records on the output magnetic tape with appropriate delay commands as discussed above.

MASS RANDOM ACCESS FILE OPERATIONS

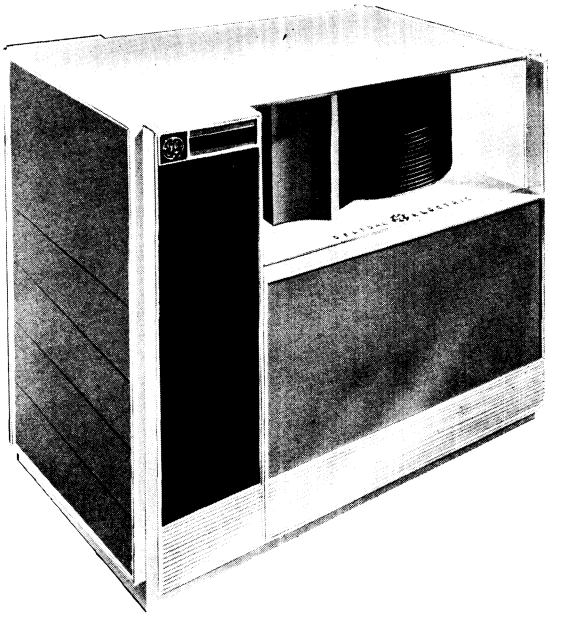


Figure 27 Mass Random Access File Sub-System

The Mass Random Access File Sub-System provides the ability to store vast quantities of data and to access that data randomly rather than sequentially. Random access processing, as opposed to sequential processing, is somewhat self-explanatory. That is, data to be processed may be accumulated on either punched cards, paper tape, or magnetic tape in the non-sequential order in which source documents are received. For example, checks in payment of insurance premiums are received daily, and the account must be credited with the amount paid. Naturally, these payments will not be received in a strict numerical succession by policy number. If payments were received this way, we would have a Utopian example of sequential input - we would not have to sort in numerical sequence prior to magnetic tape processing within a computer. Sequential updating is normally a deinite requisite when using magnetic tape as a storage medium. Since data within the Mass Random Access File is directly addressable, sequential input is not required.

Using the insurance premium receipts as an example, we are now in a position to process these transactions against a master file contained in a Mass Random Access File device. However, through the key, it is not necessary to consecutively read every master record until an equal key is found to determine whether it is the one we desire to update. In random access processing we directly address the record using the key; and the time taken in matching one piece of information to another is not dependent upon the location of the last data processed, as it is in sequential processing. The matching time is relatively negligible since it is possible to go directly to the data we need in the random access memory. Two or more files, whose sequence is divergent, may therefore be updated at practically the same time.

We must, however, have a method of using the key to locate the master data; for in order to utilize the random access capabilities of the file most fully, very careful consideration must be given to the actual addressing of records. This may be achieved through arithmetically modifying the key, using the result as a random access memory address; or in some applications the key is synonomous with the address, and no modification is necessary. Thus, the record address will most often be either carried as the key within the record itself or calculated by means of a special routine. When the record address is given by the key contained within the record itself, the key which corresponds to the address is simply inserted by the program into the address portion of the appropriate command words. When the record address is calculated by means of a special routine, it will be convenient to construct and utilize this routine as a subroutine package. At times the modification of two or more keys may result in the same address. In such cases, a system of "linking" is utilized to store and to address the proper record.

As we have seen, random access processing permits us to store master records, and to process input files non-sequentially. Some systems applications, however, may easily use a composite of random and sequential processing. An inventory system may be used as an illustration in which it is desired to update a master file containing stock number, balance on hand, unit cost, total cost, and order point; and, also to develop raw material statistics by job number. For the purposes of this example, assume that:

1. the master file is randomly contained in random access memory,
2. the job number file is on a tape in numerical succession, and
3. the transaction file is sorted in numerical sequence by job number.

We may now read a transaction, and using the job number as a key, update the proper job number master tape record; and at the same time, the stock number key, may be utilized to find and update the master inventory record within random access memory. Study and experience will disclose the most convenient or economical methods for using sequential or random processing, or a combination of both.

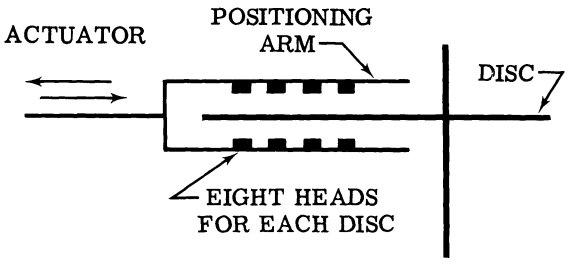


Figure I

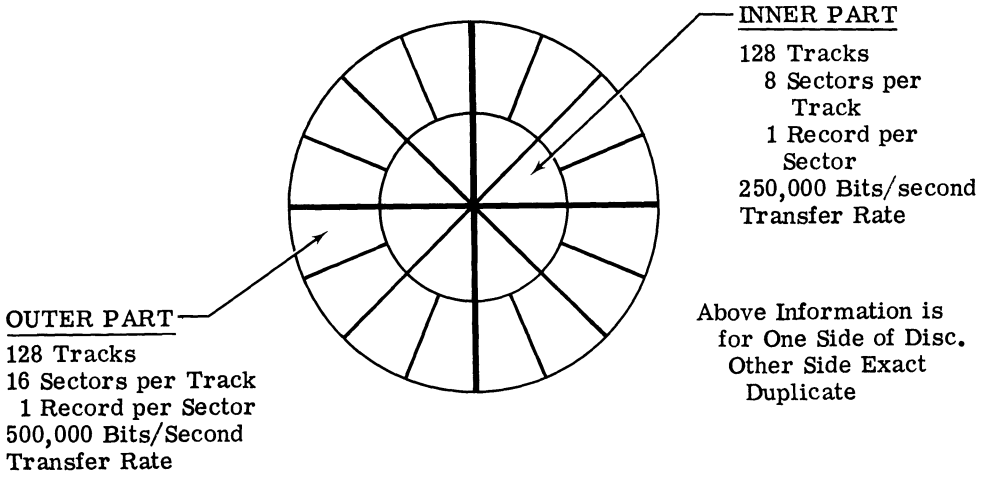


Figure II

TRANSFERS OF DATA BETWEEN THE MASS RANDOM ACCESS FILE AND THE CENTRAL PROCESSOR

BRANCH ON DATA MATING FUNCTION INTERROGATED CONDITIONS

BDM C + T/F P

P is the plug number or controller number to be interrogated. C is the number of the specific condition to be tested. Both C and P have the range 0 to 7. C + T calls for branching if the condition tested (C) is true. C + F calls for branching if the condition tested is false.

CONDITION NUMBER	MASS RANDOM ACCESS FILES
0	Controller Busy
1	File #0 Ready
2	File #1 Ready
3	File #2 Ready
4	File #3 Ready
5	Input-Output Error
6	Parity Error
7	Any Error

MASS RANDOM ACCESS FILE CONTROL INSTRUCTIONS

Each Mass Random Access File (MRAF) consists of either 16 or 64 storage discs. From one to four 16 disc MRAF's or one 64 disc MRAF may be attahced to the GE 225 core memory through a single plug on the DATA MATING FUNCTION. The 16 disc MRAF stores 98,304 records, each record consisting of 64 20-bit information words. Six numbers are required to address a specific 64 word record on a MRAF:

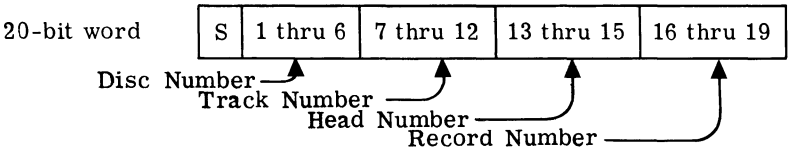
1. DATA MATING PLUG Number	(0 thru 7)	Reference Schematic Figure 1
2. File Number	(0 thru 3)	
3. Disc Number	(0 thru 15) or (0 thru 63)	
4. Head Number	(0 thru 7)	
5. Track Number	(0 thru 63)	
6. Record Number	(0 thru 15) on 256 outer tracks (0 thru 7) on 256 inner tracks	

All MRAF instructions (except Branch on MRAF conditions) require three lines of GAP 225 coding.

POSITION MRAF

	OPERATION	OPERAND	MODIFICATION
1st line:	SEL		P
2nd line:	PRF		R
3rd line:	OCT	(MRAF Address)	

P is the plug number (0 thru 7) to which the MRAF is attached. PRF is the mnemonic code for Position MRAF to transmit or receive a specific record. R is the number (0 thru 3) of the selected MRAF. The third line contains the actual MRAF address of the record to be acted upon. The format of this line is:



If S, the sign bit, is 0, the MRAF is positioned to read (or write) the specific record designated by the entire address in bits 1 thru 19.

Is S is 1, the MRAF is positioned to read any one of the 8 or 16 records designated by bits 1 thru 15. When a subsequent read MRAF instruction is given, the first available record from this position will be transmitted to core memory.

READ MRAF

	OPERATION	OPERAND	MODIFICATION
1st line:	SEL		P
2nd line:	RRF	N	R
3rd line:	RRF	M	

P is the plug number (0 thru 7) to which the MRAF is attached. RRF is the mnemonic code for Read MRAF. N is the number (1 thru 16) of 64-word records to transmit from the disc storage to core storage. R is the number (0 thru 3) of the selected MRAF. M is the core memory address (decimal number of alphabetics symbol) in to which the first word of the first record is to be copied. All following words and records, if any, will be copied into sequentially higher memory locations. M must be an even multiple of 64 words. M is not indexable.

WRITE MRAF

	OPERATION	OPERAND	MODIFICATION
1st line:	SEL		P
2nd line:	WRF	N	R
3rd line:	WRF	M	

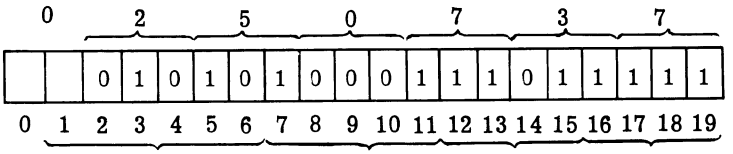
P is the plug number (0 thru 7) to which the MRAF is attached. WRF is the mnemonic code for Write MRAF. N is the number (1 thru 16) of 64-word records to transmit from consecutive core storage locations to disc storage. R is the number (0 thru 3) of the selected MRAF. M is the core memory address (decimal number or alphabetic symbol) from which the record(s) will be copied. The MRAF destination address will be the one at which the MRAF is currently positioned. M must be an even multiple of 64 words. M is not indexable.

EXAMPLES

1. Read a record from mass random access file #1. Mass random access file #1 is plugged into Data Mating plug #2. The record is to be read into storage beginning at 640. The record is located at disc # 10, head #5, track #35, and record # 15.

```
0300    BDM     O + T    2  }
0301    BRU     0300       }  Delay until controller ready
0302    SEL              2  }
0303    PRF              1  }  Position MRAF
0304    OCT     0250737     }
0305    BDM     1 + F    2  }
0306    BRU     0305       }  Delay until ready
0307    SEL              2  }
0308    RRF     1        1  }  Read Record from MRAF
0309    RRF     0604        }
0310    BDM     O + T    2  }
0311    BRU     0310       }  Delay until Read
```

The commands at 0300 and 0301 will transfer control in a continuous loop until the controller is not busy. The commands at 0302, 0303 and 0304 will position the MRAF to read the desired record as defined below:



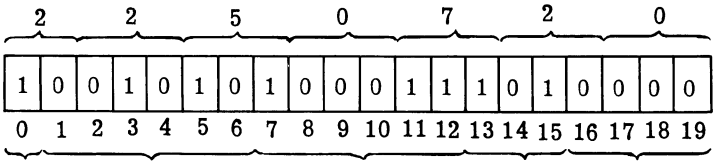
Disc # Binary 10 | Track # Binary 35 | Head # Binary 5 | Record # Binary 15

The command at 0305 and 0306 will delay further processing until the MRAF device is positioned to read the desired record. The commands at 0307, 0308 and 0309 will transfer the desired record into storage beginning at 0640. The commands in storage at 0310 and 0311 will delay further processing until the record has been fully read into storage at 0640 - 703.

2. In problem 1, above, assume that all address data is known except the record number. Read any one of the records within range of access on the same file, disc, head, and track. Assume that the record to be read will furnish further information for the "search" and further processing will interrogate the data within the record itself.

```
0400    BDM     0 + T    2   }
0401    BRU     0400         }  Delay until Controller Ready
0402    SEL              2   }
0403    PRF              1   }  Position MRAF
0404    OCT     2250720      }
0405    BMD     1 + F    2   }
0406    BRU     0405         }  Delay Until Ready
0407    SEL              2   }
0408    RRF     1        1   }
0409    RRF     640          }  Read Record from MRAF
0410    BDM     0 + T    2   }
0411    BRU     0410         }  Delay until Read
```

The commands above are identical to the commands in problem 1, above except for the address at 0404. The address at 0404 corresponds to the following:



Causes First Available Record to be Transmitted | Disc# Binary 10 | Track # Binary 35 | Head # Binary 5 | Record # Binary 0

3. In problem 1, above, assume the actual address of a record to be read from the MRAF is contained within columns 1, 2 and 3 of a card. Assume that the card has been read into storage previously, in binary coded decimal mode, beginning in storage at 0128. Therefore, 0128 contains the appropriate address of the record in the MRAF. Insert the address into the appropriate command programatically.

```
0298    LDA    0128  }
0299    STA    0304  }      Load and Store Address
```

Same and Problem 1, above.

The commands at 0298 and 0299 will insert the "address" from the card into the command at 0304, in problem 1, above. The commands at 0302, 0303 and 0304 in problem 1, above, will cause the MRAF device to be positioned to either read or write the record at the address contained in columns 1, 2 and 3 of the card.

4. In problem 1, above, assume that card columns 1, 2, 3, 4, 5 and 6 of the card contain an address from which the actual address may be derived. Assume that a subroutine is in storage at 2000 which will appropriately derive the actual address from the contents of word 0128 and 0129. Calculate the actual address and insert it within the appropriate command.

```
0297    DLD    0128          Load columns 1 - 6
0298    SPB    2000    1     Transfer control to Subroutine at 2000.
0299    STA    0304          Store calculated address from A register.
```

The numbers to be calculated are loaded into the A and Q registers by the command at 0297. The command at 0298 will transfer control to the subroutine beginning at 2000. The subroutine will "derive" or calculate the actual or "absolute" address and return control to the main program at location 0299. The command at 0299 will store the actual address from the A register into the appropriate command which addresses the record.

5. In problem 1, above, write the commands to write the record into the MRAF file at the same location. Assume the MRAF is positioned at the desired "address".

```
0505    BDM    1 + F    2  }
0506    BRU    0505       }   Delay until Controller Ready.
0507    SEL               2  }
0508    WRF    1          1  }   Write Record from Storage to File.
0509    WRF    0640          }
```

The commands at 0505 and 0506 will delay processing until the controller is ready. The commands at 0507, 0508 and 0509 will transfer the record from storage to the MRAF.

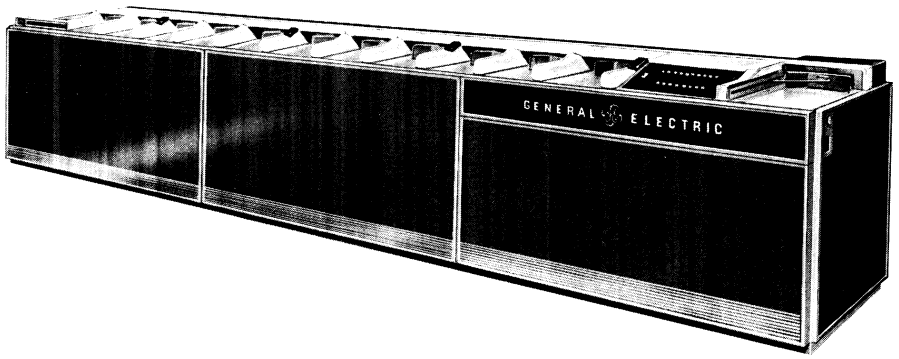


Figure 28 Twelve-pocket Document Handler

MAGNETIC DOCUMENT HANDLER OPERATIONS

Magnetic Documents may be read continuously at speeds of 1200 documents per minute, processed, and directed to appropriate pockets all under control of the central processor of the GE 225. Because of the speeds of document travel, and data transfer, the programmer must direct his attention to timing requirements in order to achieve maximum efficiency, i.e. simultaneous reading at maximum speeds and processing of each document.

Timing requirements will be met if the program instructions are arranged to concide with the following sequence of operations.

1. Read Document
2. Delay until Document Read
3. Direct Document to appropriate Pocket
4. Read next document

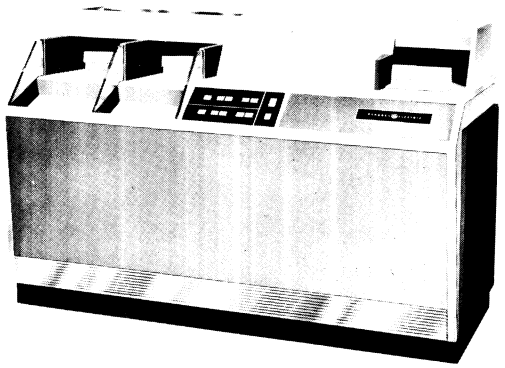


Figure 29 Two-pocket Document Handler

Since the "delay" in step 2 is really a programmed function, the Programmer may readily execute an alternative system of processing which utilizes the simultaneous Read, Process (and Print) abilities of the GE 225.

1. Read Document 1
2. Read Document 2
3. Direct Document 1 to Pocket
4. Process Document 1
5. Read Document 3
6. Direct Document 2 to Pocket
7. Process Document 2
8. etc.

Such a pattern is illustrated in the examples which follow.

The Document Handler can recognize 14 characters: the ten decimal digits and four special symbols called Cue Characters. Cue Characters are normally used to separate fields of decimal digits, to identify dollar amount fields and identification fields. Each character is converted into the BCD code given below and stored in the least significant four bits of a word in memory. Except for the case of Cue Characters or invalid characters, all other bits of the word are zeros. If a character is invalid (cannot be recognized and translated by the Reader), the sign bit of the word in memory for that character is set to 1. If the character read is one of the four Cue Characters, both the sign bit and the most significant bit (bit 1) are set to 1.

The following table gives the BCD codes for each of the 14 recognizable characters.

Magnetic Document Character	GE-225 BCD Code (Bits 16, 17, 18, 19)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
Cue 1	1100
Cue 2	1010
Cue 3	1101
Cue 4	1011
"INVALID CHARACTER"	1111

BASIC PROGRAMMING OF THE MAGNETIC DOCUMENT HANDLER

BRANCH ON DOCUMENT HANDLER INTERROGATED CONDITIONS

OPR	OPERAND	MODIFICATION
BDM	C + T / F	P

BDM is the mnemonic code for Branch on Data Mating Function. P is the plug number (0 thru 7) to which the Handler is attached. C is the number (0 thru 7) of the specific condition to be interrogated. C + T calls for branching if the condition tested is TRUE. C + F calls for branching if the condition tested is FALSE.

CONDITION NUMBER	CONDITION TESTED
0	Reader Number 1 reading
1	Reader Number 2 reading
3	
4	Reader Number 1 feeding
5	Reader Number 2 feeding
6	Input Buffer Error (Priority)
7	Any error or parity error

MAGNETIC DOCUMENT HANDLER INSTRUCTIONS

One or two 1200 document per minute magnetic document handlers may be connected on line with the GE-225 through a single Data Mating Function plug. Under programmed control a document handler can be commanded to perform any of the following actions:

1. Read a single document into a specified address of core memory. (Single Feed)

2. Read the current input document into a specified address of core memory and position the next input document for immediate reading. (Continuous Feed)

3. Deposit the document last read into a specified pocket. The document handler has 12 pockets that may be used to stack the paper documents in separate groups according to identification codes on the documents.

4. Halt the continuous feed action.

Three lines of mnemonic coding are required to define a magnetic document handler action.

READ SINGLE DOCUMENTS

	OPR.	OPERAND	MODIFICATION
1st line:	SEL		P
2nd line:	RSD		N
3rd line:	RSD	M	

P is the Data Mating Function plug number (0 thru 7) to which the Document Handler is attached. RSD is the mnemonic code for Read Single Document. N is the Handler number (1 or 2). If N is blank, the assembly program will assume "1" is the Handler number. M is the memory address into which the first character read from the document will be copied. M may be a symbolic address or a decimal integer fixed address. M is not automatically modified. Single reading of

documents can be done at the rate of 600 per minute. There are no restrictions on the amount of processing time that can be used on each document when the documents are read with RSD type instructions.

READ DOCUMENTS AND CONTINUE FEEDING NEXT DOCUMENT

	OPR	OPERAND	MODIFICATION
1st line:	SEL		P
2nd line:	RDC		N
3rd line:	RDC	M	

This set of instructions is the same as the RSD instructions, except that they call for moving a second document into position for immediate reading after the first document passes the reading head. RDC type instructions must be used to achieve the 1200 document per minute reading speed. With each document read by RDC type instructions, there are approximately 50 milliseconds of processing time available before another RDC instructions must be given to the Document Handler.

POCKET SELECTION

	OPR	OPERAND	MODIFICATION
1st line:	SEL		P
2nd line:	PKT		N
3rd line:	OCT	(Pocket Address)	

P is the Data Mating Function plug number (0 thru 7) to which the Document Handler is attached. PKT is the mnemonic code for Pocket Select. N is the Handler number (1 or 2). If N is blank, the assembly program will assume "1" is the Handler number. The third line contains the binary code for the specific pocket into which the document last read is to be stacked. To be effective, the Pocket Selection command must be given within a maximum of 45 milliseconds after the reading of the document is complete.

The following table gives the codes (in octal) to be used in the third line of a Pocket Selection command.

POCKET IDENTIFICATION	OCTAL CODE FOR HANDLER NO. 1	OCTAL CODE FOR HANDLER NO. 2
SPECIAL	01	020
0	17	360
1	16	340
2	15	320
3	14	300
4	13	260
5	07	160
6	06	140
7	05	120
8	04	100
9	03	060
REJECTS	02	040

The above octal codes are stored as the least significant (rightmost) digits of the operand field of the third line of a POCKET SELECT instruction set, and zeros fill out the field to the left.

HALT THE CONTINUOUS FEEDING ACTION

	OPR	OPERAND	MODIFICATION
1st line:	SEL		P
2nd line:	HLT		N
3rd line:	HLT	M	

P and N have the same meaning as they do in the other Document Handler commands. HLT is the mnemonic code for HALT the continuous feeding of documents. M is the memory address (symbolic or decimal) into which the first character of the document currently approaching the reading head will be copied.

END READ BUSY SIGNAL

	OPR	OPERAND	MODIFICATION
1st line:	SEL		P
2nd line:	ERB		N
3rd line:	XXX		

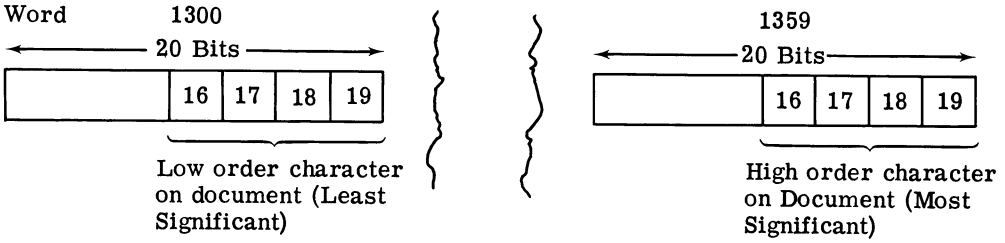
P is the plug number (0 thru 7) to which the Document Handler is attached. ERB is the mnemonic code for End Read Busy. N is the Handler number (blank, 1, or 2). If N is blank, the assembly program will assume "1" is the Handler number. The third line must be present, but it is not used in this instruction. The programmer may use this line as working storage or as constant storage. XXX may be any one of the following mnemonics: DEC, OCT, ALF.

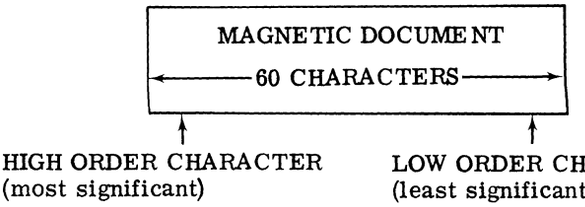
EXAMPLES

1. Document sorter #2 is plugged into Data Mating Plug #5. Read a single document from the document sorter into storage beginning at location 1300. Assume 60 characters of data will be read into storage from the document.

```
900     SEL             5   } READ SINGLE DOCUMENT
901     RSD             2   }
902     RSD     1300
903     BDM     1 + T   5   } DELAY PROCESSING UNTIL DOCUMENT
904     BRU     903         } IS READ
          |       |
          ▼       ▼
Further Processing of Document
```

A single document will be read from the document sorter into storage by the commands at 900 through 902. The commands at 903 and 904 will delay further processing until reading of the document has been completed. Characters are read from the document in sequence from right to left. However, characters are placed in storage, within the G.E. 225 from left to right one character per word, placed in bit positions 16, 17, 18 and 19 of each word for example:





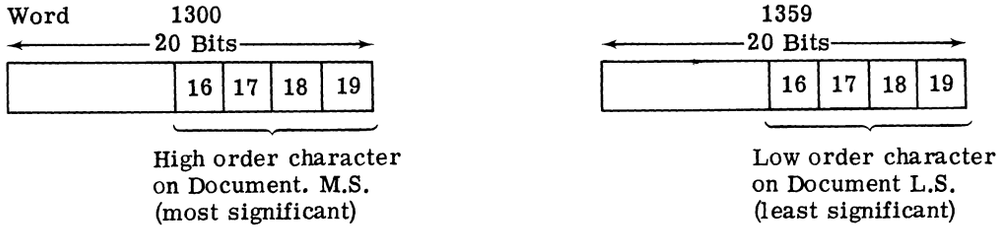
NOTE

Therefore, at this point, the sequence of characters in storage is reversed from the sequence of characters on the document itself.

2. In problem 1, above, write the program commands to "ring shift" the characters from the magnetic document to conform to the sequence of the characters as they appear on the document itself. Also, include the necessary operations in the "ring shift" to "validity check" each character read. If any character is invalid, branch to an error correction routine in storage at 3000. Assume that "cue" characters within the data are designated for control functions on the plugboard of the document handler and will not enter into storage.

905	LDZ			
906	STA	1		Set modification word 1 to 0, modification
907	STA	2		word 2 to 59.
908	INX	59	2	
909	LDA	1300	1	
910	BMI			Load L.S. into Q Register and Test for
911	BRU	3000		Validity.
912	XAQ			
913	LDA	1300	2	
914	BMI			Load M.S. Digit into A Register and Test
915	BRU	3000		for Validity.
916	STA	1300	1	Store M.S. Digit
917	XAQ			Store L.S. Digit
918	STA	1300	2	
919	INX	1	1	Increment modification word 1 and test
920	BXH	30	1	for completion
921	BRU	Continue		
922	INX	-1	2	Decrement modification word 2 Transfer
923	BRU	909		to Repeat.

The above commands are designated to place the characters in storage in sequence as they appear on the document itself, and also test each character for validity. Thus the characters in storage will appear as follows after the above commands:



NOTE: After the characters are arranged in sequence, as above, they may be conveniently listed on the high speed printer with automatic format control. (Explained in another section of these materials).

The commands at 905 through 908 will prepare the modification words for initial processing. The commands at 909 through 918 will reverse the sequence of a set of 2 characters. The sequence is repeated for each of 30 sets of 2 characters.

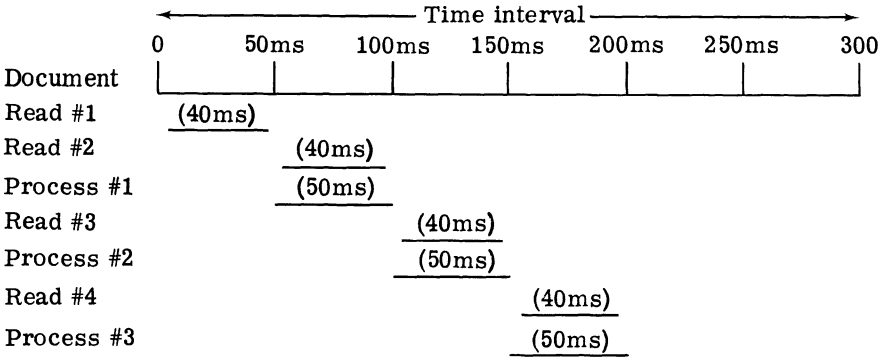
The commands at 919 through 922 will prepare the modification words for processing of the next set of 2 characters, and test for completion of processing. If processing is not complete, the command at 923 will continue processing for the next set of 2 characters to be reversed.

3. Document sorter #1 is plugged into Data Mating Plug #5. Read documents and continue feeding the next document, so as to read documents at the maximum speed of 1200 documents per minute. Continue until the hopper is empty.

Read the first document into storage beginning at location 3000. Read the second document into storage beginning at 4000. Read the third document into storage at 3,000, the fourth at 4,000, etc. Process the first document while reading the second, process the second while reading the first, etc. Direct the first document to pocket 1, the second to pocket 2, the third to pocket 1, the fourth to pocket 2, etc.

```
2000    SEL              5
2001    RDC                    Read Document Continuous Feed (Sorter
                               1 assumed).
2002    RDC     3000
2003    BDM     0 + T          Delay until First Document Read (Approx.
2004    BRU     2003           40 ms).
2005    SEL              5
2006    RDC                    Initiate Reading of Second Document
2007    RDC     4000
2008    SEL              5
2009    PKT                    Direct First Document to Pocket 1.
2010    OCT     0000016
  |       |       |
  ▼       ▼       ▼
Process first document (approx. 50ms) possible.
  |       |       |
  ▼       ▼       ▼
2048    BDM     4 + F    5
2049    BRU     4500           Interrogate and Transfer for Hopper Empty.
2050    BDM     0 + T
2051    BRU     2050           Delay until second document read.
2052    SEL              5
2053    RDC                    Initiate reading of third Document.
2054    RDC     3000
2055    SEL              5
2056    PKT                    Direct second document to pocket 2.
2057    OCT     0000015
  |       |       |
  ▼       ▼       ▼
Process second document (approx. 50 ms).
  |       |       |
  ▼       ▼       ▼
2062    BDM     4 + F    5
2063    BRU     4500           Interrogate and Transfer for Hopper Empty.
2064    BRU     2003           Transfer to continue.
```

The above commands will pattern processing and document reading to occur simultaneously according to the pattern described below:



*Processing time for any one document may continue for 50ms before a new document is read into the processing area.

*Reading time for any one document may continue for approximately 40 ms or less.

The "N" position in the commands at 2001, 2006, 2009, 2053, 2056, is not provided since the General Assembly Program will assume that sorter 1 is intended when a blank appears in the "N" position. The commands at 2048 - 2049 and 2062 - 2064 will interrogate for the "hopper empty" condition since feeding will stop when the hopper is empty. The read document continuous feed next document command (RDC) must be repeated for each document read in all cases - even if all documents are to be read beginning in storage at the name location. The document just read must be directed to an appropriate pocket within 30 ms. after it is read. A possible sequence of processing may consist of:

	Read Documents Continuous (First Document)
May both be considered as processing of first document. *	Delay for Read Completion
	Read Document Continuous (Second Document)
	Direct Document to pocket (First Document)
	Process Document (First Document)
	Interrogate for Hopper Empty
May both be considered as processing of second document. *	Delay for Read completion (Second Document)
	Read Document Continuous (Third Document)
	Direct Document to Pocket (Second Document)
	Process Document (Second Document)
	Interrogate for Hopper Empty

*NOTE:
Since direction of a document to the appropriate pocket will most likely depend upon data read from the document itself, both steps should really be considered as a function of processing.

4. Assume that a RDC command has been issued, also assume that reading of the previous document is complete. Write the commands to Halt the reading and feeding of further documents.

```
2052    SEL          5        Begin halt of Reader.
2053    HLT
2054    HLT     3000
2055    SEL          5        Direct Previous Document to Pocket 2.
2056    PKT
2057    OCT     0000015
```

Process Previous Document

```
2062    BDM     0 + T   5       Delay for Read Busy
2063    BRU     2062
2064    SEL             5
2065    HLT
2066    HLT     4000            Halt the Reader
2067    SEL             5
2068    PKT
2069    OCT     0000016         Direct first Document after Halt to Pocket 1.
```

Process First Document after "Halt"

```
2074    LDZ                     Set x word 1 to zero
2075    STA     1
2076    BDM     0 + F   5
2077    BRU     2085            Delay for Read Busy
2078    INX     -1      1
2079    BXL     300     1       Continue Delay for approx. 50ms.
2080    BRU     2076
2081    SEL             5       Turn off Read Busy Signal
2082    ERB
2083    DEC     0
2084    BRU     2084            Halt
2085    SEL             5
2086    PKT                     Direct Second Document After Halt to Pocket 2.
2087    OCT     0000015
```

Process Second Document after Halt

```
2090    BRU     2090
```

When a Halt is desired after a RDC command, there is a possibility that two more documents may be read before the Halt is completed. Therefore, the programmer most make adequate provision in his program, as above, for:

1. Direction of the next two documents to appropriate locations in storage (Halt commands).

2. Direction of the next two documents to appropriate pockets.

3. Appropriate processing of the next two documents, as necessary.

One or two documents may be read after a "Halt" command. It is possible that the second document after the first "Halt" command will not be read. If the second document is not read, the read busy signal will not be turned off. Therefore, a "time count" is built into the second read busy interrogation "loop" by the commands in storage at 2078 through 2080. If the loop continues for 50ms. it is assumed that the second document after the "Halt" command was not read. The commands at 2081 through 2083 will turn the "read busy signal" off and processing is halted by the command at 2084.

I. APPENDIX

NUMBER SYSTEMS

Number Representation

The following formula defines a pattern for the representation of numbers:

$$N = A_m r^m + \ldots + A_2 r^2 + A_1 r^1 + A_0 r^0 + A_{-1} r^{-1} + A_{-2} r^{-2} + \ldots + A_{-m} r^{-m}$$

where: N is the number,

A is a permissible symbol in the number system,

r is the radix (the total number of permissible symbols in the number system),

m is the position of the symbol (m = 0 is in the first position to the left of the decimal point with increasing integral values of m in positions moving to the left and decreasing integral values of m in positions moving to the right).

(Note: Mathematically, $r^0 = 1$ regardless of values

Decimal System

In demonstration of the validity of the above formula, note that the formation of the numbers 5126 and 32.425 in the decimal system may be written as:

$$\begin{aligned} 5126 &= 5000 + 100 + 20 + 6 \\ &= 5 \times 1000 + 1 \times 100 + 2 \times 10 + 6 \times 1 \\ &= 5 \times 10^3 + 1 \times 10^2 + 2 \times 10^1 + 6 \times 10^0 \end{aligned}$$

In this case, $A_3 = 5$, $A_2 = 1$, $A_1 = 2$, $A_0 = 6$.

$$\begin{aligned} 32.425 &= 30 + 2 + 4/10 + 2/100 + 5/1000 \\ &= 3 \times 10 + 2 \times 1 + 4 \times 1/10 + 2 \times 1/100 + 5 \times 1/1000 \\ &= 3 \times 10^1 + 2 \times 10^0 + 4 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3} \end{aligned}$$

In this case, $A_1 = 3$, $A_0 = 2$, $A_{-1} = 4$, $A_{-2} = 2$, $A_{-3} = 5$.

Thus, decimal numbers are formed by stating the coefficients (symbols) of the powers (position) of 10. Most importantly, since there is nothing special about r = 10, the same rules must apply to number systems using other values for the radix.

Binary System

The binary system consists of two admissible symbols: 0 and 1. Therefore, the radix is two. For example, the decimal number 21 may be represented in binary notation as follows:

$$\begin{aligned} 21 &= 16 + 4 + 1 \\ &= 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^0 \\ &= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 10101 \text{ (binary)} \end{aligned}$$

Octal System

The octal system consists of eight admissible symbols: 0, 1, 2, 3, 4, 5, 6, 7. Therefore, the radix is eight. For example, the decimal number 301 may be represented in octal notation as follows:

$$\begin{aligned} 301 &= 256 + 40 + 5 \\ &= 4 \times 64 + 5 \times 8 + 5 \times 1 \\ &= 4 \times 8^2 + 5 \times 8^1 + 5 \times 8^0 \\ &= 455 \text{ (octal)} \end{aligned}$$

The chief use of the octal system is as a shorthand for binary representation since conversion between systems can be done mentally. To demonstrate the validity of this statement, consider the octal number 455 (decimal 301) used in the example above.

$$\begin{aligned} 455 &= 100\ 101\ 101 \\ &= 100101101 \\ &= 1 \times 2^8 + 0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 1 \times 2^8 + 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 \\ &= 256 + 32 + 8 + 4 + 1 \\ &= 301 \text{ (decimal)} \end{aligned}$$

Clearly, 455 is a much more convenient notation then 100101101, yet the conversion is trivial when required.

BINARY ARITHMETIC

Binary Addition

The binary, single digit, addition table for A + B is:

	+	B = 0	B = 1
A	0	0	1
	1	1	0*

*A "1" binary digit is carried to the next higher order binary digit to the left. Note that this is equivalent to the same rule in the decimal system that yields 9 + 1 = 10, which is actually 9 + 1 = 0 with a carry into the next higher order position. (In the binary system 1 is the highest permissible symbol just as in the decimal system 9 is the highest permissible symbol.) The following examples illustrate binary addition:

Carry	1 1	111	111
	100110	100111	0111
	+ 110101	+001110	+ 1
Sum	1011011	110101	1000

Binary Subtraction

The binary, single digit, subtraction table for A - B is:

	-	B = 0	B = 1
A	0	0	1*
	1	1	0

*A "1" binary digit is borrowed from the next higher order digit to the left. Keeping track of borrowed digits in this method of subtraction is significantly more difficult than keeping track of carries in addition. It is therefore desirable to develop an alternate method of subtraction which will employ an addition table. It is easy to demonstrate, using the more familiar decimal system, that subtraction can be accomplished by addition of the 10's complement of the number to be subtracted and the dropping of the carry. Thus

$$\begin{array}{r} 7 \\ -4 \\ \hline 3 \end{array} \qquad \text{or} \qquad \begin{array}{r} 7 \\ 6 \\ \hline {}^{1}3 \end{array}$$

The mathematical validity of this rule is clearly shown as follows:

$$7 - 4 = 7 - (10 - 6) = 7 + 6 - 10 = 3$$

noting that the carry to be dropped is equal to 10. In the same fashion, binary subtraction may be accomplished by the binary addition of the 2's complement of the number and the dropping of the final carry. (A 2's complement is extremely easy to form: 1 bits are changed to zero bits, 0 bits are changed to 1 bits, and a 1 bit is added to the result.) Examples of binary subtraction are:

```
 39     100111                240    011110000
- 9    +110111               -112   +110010000
      --------                     ---------
      011110 = 30                  010000000 = 128
```

For convenience, the GE 225 has a special instruction which forms the 2's complement of a number.

Binary Multiplication

The binary, single digit, multiplication table for A x B is:

	X	B = 0	B = 1
A	0	0	0
	1	0	1

For example:

```
 35      100011
x 13      1101
        100011
       1000110
     100011
    ---------
    111000111 = 455
```

Thus, multiplication in binary reduces to a series of additions.

Binary Division

Division in binary can be carried out by the same process as in decimal; that is, by repeated subtractions. As in binary subtraction, the 2's complement of the divisor is added repetitively.

```
    3                  11 = 3
9 ) 27          1001 ) 11011
                       0111
                       1001
                       0111
                       0000
```

BINARY-DECIMAL CONVERSION TABLE

2^n	n	2-n
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

REPRESENTATION OF CHARACTERS

CHARACTER	HOLLERITH CODE (PUNCH IN ROWS)	BCD MEMORY (OCTAL)	BCD MAGNETIC TAPE (OCTAL)	HIGH SPEED PRINTER SYMBOLS	CONSOLE TYPEWRITER CHARACTER OR ACTION	PAPER TAPE CHARACTER (8 CHANNEL)
0	0	00	12	0	SPACE	SPACE
1	1	01	01	1	1	1
2	2	02	02	2	2	2
3	3	03	03	3	3	3
4	4	04	04	4	4	4
5	5	05	05	5	5	5
6	6	06	06	6	6	6
7	7	07	07	7	7	7
8	8	10	10	8	8	8
9	9	11	11	9	9	9
A	12-1	21	61	A	⊠	/
B	12-2	22	62	B	S	S
C	12-3	23	63	C	T	T
D	12-4	24	64	D	U	U
E	12-5	25	65	E	V	V
F	12-6	26	66	F	W	W
G	12-7	27	67	G	X	X
H	12-8	30	70	H	Y	Y
I	12-9	31	71	I	Z	Z
J	11-1	41	41	J	J	J
K	11-2	42	42	K	K	K
L	11-3	43	43	L	L	L
M	11-4	44	44	M	M	M
N	11-5	45	45	N	N	N
O	11-6	46	46	O	O	O
P	11-7	47	47	P	P	P
Q	11-8	50	50	Q	Q	Q
R	11-9	51	51	R	R	R
S	0-2	62	22	S	B	S
T	0-3	63	23	T	C	C
U	0-4	64	24	U	D	D
V	0-5	65	25	V	E	E
W	0-6	66	26	W	F	F
X	0-7	67	27	X	G	G
Y	0-8	70	30	Y	H	H
Z	0-9	71	31	Z	I	I
+	12	20	60	+	0	0
-	11	40	40	-	-	-
Δ	BLANK COLUMN	60	20	BLANK	&	&
/	0-1	61	21	/	A	A
	2-8	12			⊠	
#	3-8	13	13	#	/	STOP
@	4-8	14	14	@	⊠	
	5-8	15		-	⊠	
	6-8	16		PENDING DECISION	⊠	
	7-8	17			⊠	
	12-2-8	32	72		PRINT RED	
.	12-3-8	33	73	.	.	,
□	12-4-8	34	74		⊠	
	12-5-8	35			PRINT BLACK	
	12-6-8	36			TAB	TAB
	12-7-8	37			⊠	
	11-2-8	52	52		⊠	
$	11-3-8	53	53	$	$	$
*	11-4-8	54	54	*	⊠	
	11-5-8	55			⊠	
	11-6-8	56			⊠	
	11-7-8	57			⊠	
	0-2-8	72	32		⊠	
,	0-3-8	73	33	,	⊠	
%	0-4-8	74	34	%	⊠	
[	0-5-8	75		[	⊠	
]	0-6-8	76		]	⊠	
	0-7-8	77			CARRIAGE RETURN	DELETE

⊠ INDICATES TYPEWRITER HANG UP. OPERATION HALTS.

MAGNETIC DOCUMENT CHARACTER	BCD CODE	MAGNETIC DOCUMENT CHARACTER	BCD CODE
0	0000	8	1000
1	0001	9	1001
2	0010	Cue 1	1100
3	0011	Cue 2	1010
4	0100	Cue 3	1101
5	0101	Cue 4	1011
6	0110		
7	0111	"INVALID CHARACTER"	1111

NUMERIC CODES FOR STANDARD
225 INSTRUCTIONS

Most Significant Bits (S, 1)

Least Significant Bits (2,3,4) OCTAL	0	1	2	3
0	LDS	DLD	EXT	FLD
1	ADD	DAD		FAD
2	SUB	DSU		FSU
3	STA	DST	ORY	FST
4	BXL	INX		
5	BXH	MPY	General	FMP
6		DVD	BRV	FDV
7	SPB		STO	External Linkage

The extension of the GE 225 instruction repertoire beyond the "standard" commands designated by bit positions 0 through 4 is accomplished through the use of bit positions in the operand address field for those instructions which require only a limited portion of the field (e.g., shift commands require only bit positions 15 through 19 to indicate length of shift) and for those instructions which do not have an operand address (e.g., word transfers between registers). The flexibility of the instruction repertoire is still further enhanced by the addition of a feature known as micro-programming. Micro-programming is the building of a computer instruction under programmer control by the specification of a series of elementary operations. In the chart given below, a 1 bit in any of the labelled bit positions will result in the elemental action described therein when the instruction is executed. In addition, if the contents of more than one register are to shift into the A register, the bits will be added logically.

For example, a 1 in bit positions 10 and 11 of the "general" Shift Right instruction instructs the computer to take the actions $A_{19} \rightarrow Q_1$ and $A_{19} \rightarrow N_1$. The octal operation code for this specific command is 25114. Reference to the octal listing in the Appendix shows this to be an ANQ (Shift A into N and Q) command. The instruction repertoire describes ANQ as follows:

> The contents of Register A (1-19) are shifted K places to the right into both Register N and Register Q. Bits shifted out of Register A (19) enter both Register Q (1) and Register N (1). Bits shifted out of Register N (6) and Register Q (19) are lost.

This information on micro-programming is included only for the use of the advanced programmer who desires to create his own special instructions. Normal programming will employ only the mnemonic or octal codes that have been assigned to the most common combinations of "micro" operations.

S 1 2 3 4	5 6	7 8 9	10	11	12	13	14	15 16 17 18 19
COMMAND	X	SUB-COMMANDS & ADDRESSES						
1 0 1 0 1								
SHIFT RIGHT		1 0 0	$Q_1 \rightarrow A_{19}$	$N_1 \rightarrow A_{19}$	$A_1 \rightarrow Q_{19}$	$A_1 \rightarrow N_6$	$A_1 \rightarrow A_{19}$	LENGTH OF SHIFT
SHIFT LEFT		1 0 1	NORMALIZE		$A_{19} \rightarrow Q_1$			LENGTH OF SHIFT
WORD MOVEMENT (REGISTER TRANSFER)		0 1		FIRST PART A → B		SECOND PART $\overline{B}$ → B	CHS	ADD ONE / A → A.U. / A → Q / A.U. → A / Q → A
INPUT/OUTPUT		0 0	DATA ORIGIN MULTIPLE OF 128					DECODE
DATA MATING FUNCTION		0 0		CONTROLLER ADDRESS				1
BRANCH TRUE		1 1	0					DECODE
BRANCH FALSE		1 1	1					DECODE
DATA MATING BRANCH		1 1	0 or 1	CONTROLLER ADDRESS				1 DECODE

OCTAL LIST OF INSTRUCTIONS

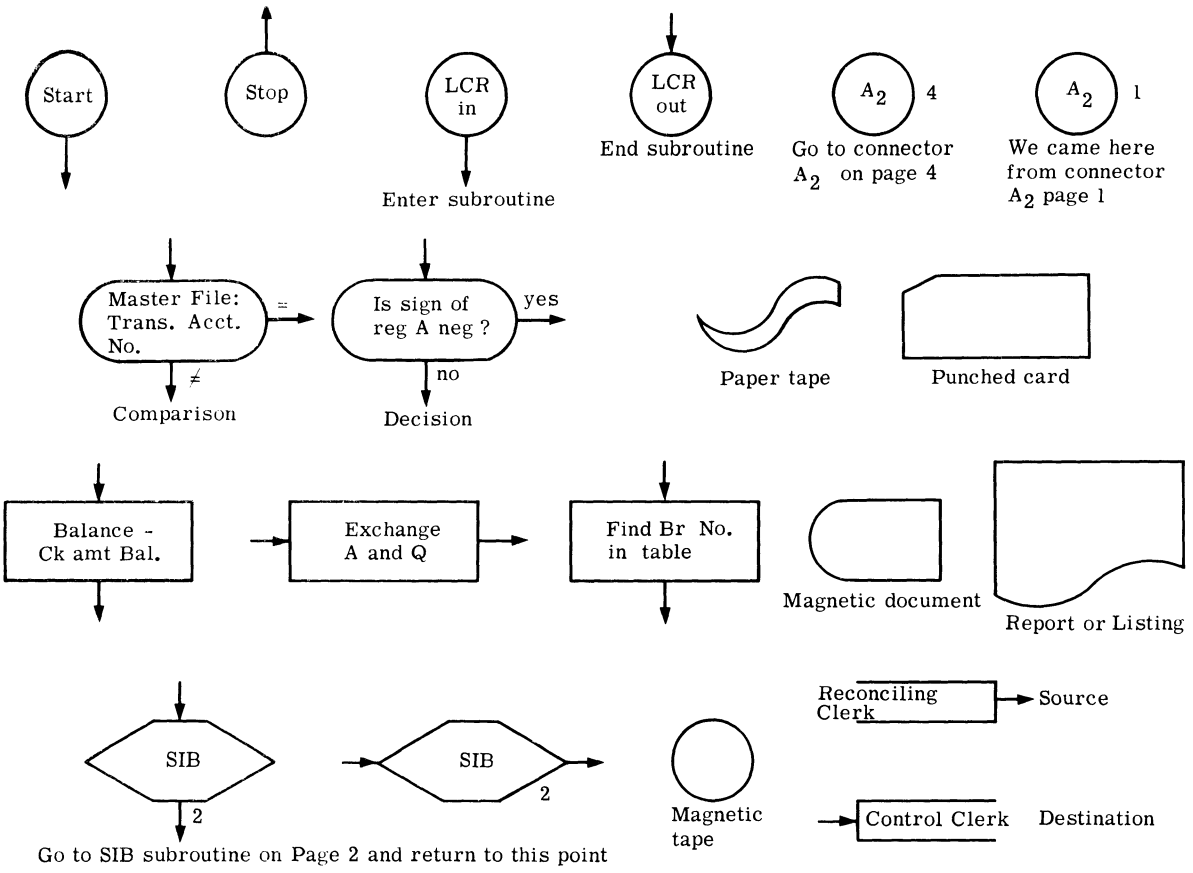
GE 225 BASIC SYSTEM

Octal Code	Description	Mnemonic
0000000	Load A	LDA
0100000	Add contents of Y to A	ADD
0200000	Subtract Y from A	SUB
0300000	Store A	STA
0400000	Branch if X is low	BXL
0500000	Branch if X is high	BXH
0700000	Store P & branch	SPB
1000000	Double Length Load	DLD
1100000	Double Length Add	DAD
1200000	Double Length Subtract	DSU
1300000	Double Length Store	DST
1400000	Increment X by K	INX
1500000	Multiply Y by Q	MPY
1600000	Divide A & Q by Y	DVD
2000000	Extract	EXT
2300000	Or A into Y	ORY
250YY00	Read Cards Decimal	RCD
250YY01	Read Cards Binary	RCB
250YY02	Write Card Decimal	WCD
250YY03	Write Card Binary	WCB
2500004	Halt Card Reader	HCR
2500005	Input-output off	OFF
2500006	Type	TYP
2500007	Typewriter on	TON
2500010	Read Paper Tape	RPT
2500011	Read control switches	RCS
2500012	Write Paper Tape	WPT
2500014	Reader on	RON

Octal Code	Description	Mnemonic
2500015	Punch on	PON
2504	Load Zero into A	LDZ
2504001	Load A from Q	LAQ
2504004	Load Q from A	LQA
2504005	Exchange A & Q	XAQ
2504006	Move A to Q	MAQ
2504012	No operation	NOP
2504022	Load one into A	LDO
2504032	Add one	ADO
2504040	Change sign of A	CHS
2504102	Load Minus one into A	LMO
2504112	Subtract one	SBO
2504502	Complement A	CPL
2504522	Negate A	NEG
25100	Shift Right A	SRA
251004	Shift Circular A	SCA
25101	Shift N & A Right	SNA
25104	Shift A & N Right	SAN
25110	Shift Right Double	SRD
25111	Shift N, A & Q Right	NAQ
25112	Shift Circular Double, Right	SCD
25114	Shift A into N & Q	ANQ
25120	Shift Left A	SLA
25122	Shift Left Double	SLD
25130	Normalize A Register	NOR
25132	Double Length Normalize	DNO
2514000	Branch on odd	BOD
2514001	Branch on Minus	BMI
2514002	Branch on Zero	BZE
2514003	Branch on overflow	BOV

Octal Code	Description	Mnemonic
2514004	Branch on Parity Error	BPE
2514005	Branch on N Register Ready	BNR
2514006	Branch on Card Reader Ready	BCR
2514007	Branch on Card Punch Ready	BPR
2516000	Branch on even	BEV
2516001	Branch on Plus	BPL
2516002	Branch on no zero	BNZ
2516003	Branch on no Overflow	BNO
2516004	Branch on Parity Correct	BPC
2516005	Branch on N register not ready	BNN
2516006	Branch on Card Reader Not Ready	BCN
2516007	Branch on Card Punch Not Ready	BPN
2600000	Branch unconditionally	BRU
2700000	Store Operand Address	STO

STANDARD FLOW CHART SYMBOLS

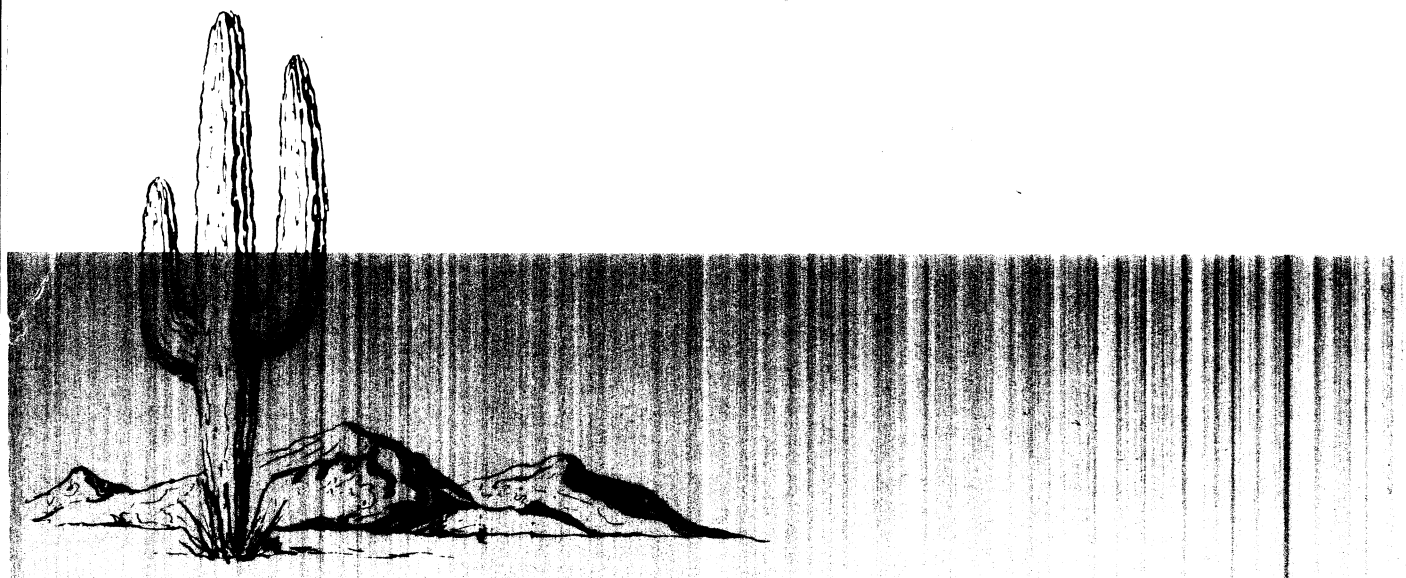


:	Comparison (the nature of the comparison is indicated separately)
=	Equal to
≠	Not equal to
>	Greater than
<	Less than
≥	Greater than or equal to
≤	Less than or equal to

AUTOMATED BY GENERAL ELECTRIC

Progress Is Our Most Important Product

GENERAL ELECTRIC



CPB 126A (3M 5-61)