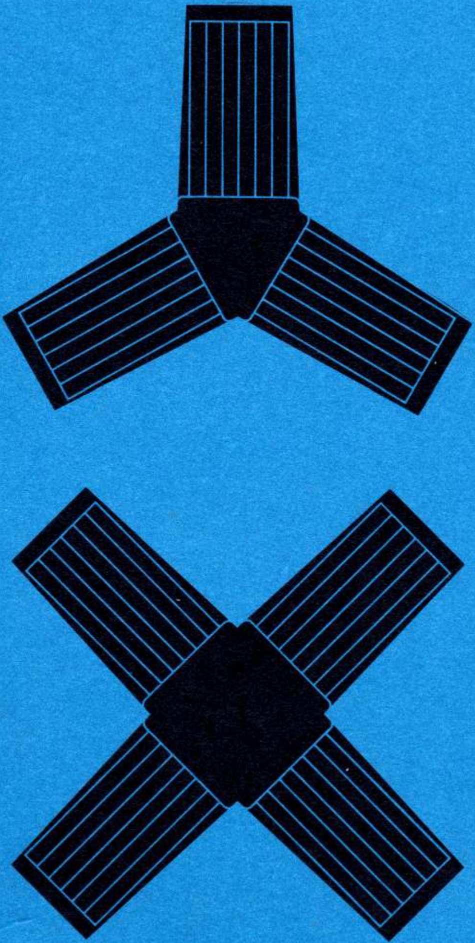
GE-425/435 Reference Manual



GENERAL ELECTRIC

GE-425/435

REFERENCE MANUAL

DECEMBER 1963

GENERAL ELECTRIC

COMPUTER DEPARTMENT

PREFACE

Although this publication is primarily a reference manual for the GE-425 and GE-435 Information Processing Systems, it is organized and illustrated to serve also as a training aid. It assumes that the reader is generally familiar with computers and computer terminology, although terms having special meanings are defined as they are presented. Broad aspects of the GE-425 and GE-435 systems are discussed with emphasis given to the functions and capabilities of the system central processor. Internal logic is described where it has a bearing on the activities of the programmer. It is recommended that this publication be used in conjunction with the reference manuals for the GE-425/435 Macro Assembly Program and individual peripheral subsystems.

CONTENTS

ILLUSTRATIONS

I. SYSTEM DESCRIPTION

INTRODUCTION

The GE-425 and GE-435 Information Processing Systems are the initial members of the powerful GE-400 family of general-purpose digital computers. Because of the high degree of compatibility and the logical similarity of the two systems, they are described together in this manual. Differences between the GE-425 and the GE-435 are specifically noted where they are significant.

This manual presents the functional details of the GE-425/435 computer systems in sufficient depth to provide the programmer with systems familiarity. A series of reference manuals for the complete line of programming systems and aids available to the user provide additional programming information.

Figure I-1 illustrates the major components of a GE-425/435 system, together with representative peripheral (input/output) subsystems. These major components are described in subsequent sections of the manual. Peripheral subsystems, which are described only briefly in this manual, are the subject of separate reference manuals on each subsystem.

CENTRAL PROCESSOR

The GE-425/435 Central Processor includes three major functional units:

1. The magnetic core memory, which provides high-speed storage for data and programs.
2. The processor section, which provides for control and execution of arithmetic, branching, and logical operations.
3. The input/output (I/O) control section, which monitors and controls all peripheral operations on as many as eight I/O channels.

In combination, these sections provide a general-purpose digital processor with a binary/decimal arithmetic unit using a relocatable, variable-length accumulator in memory.

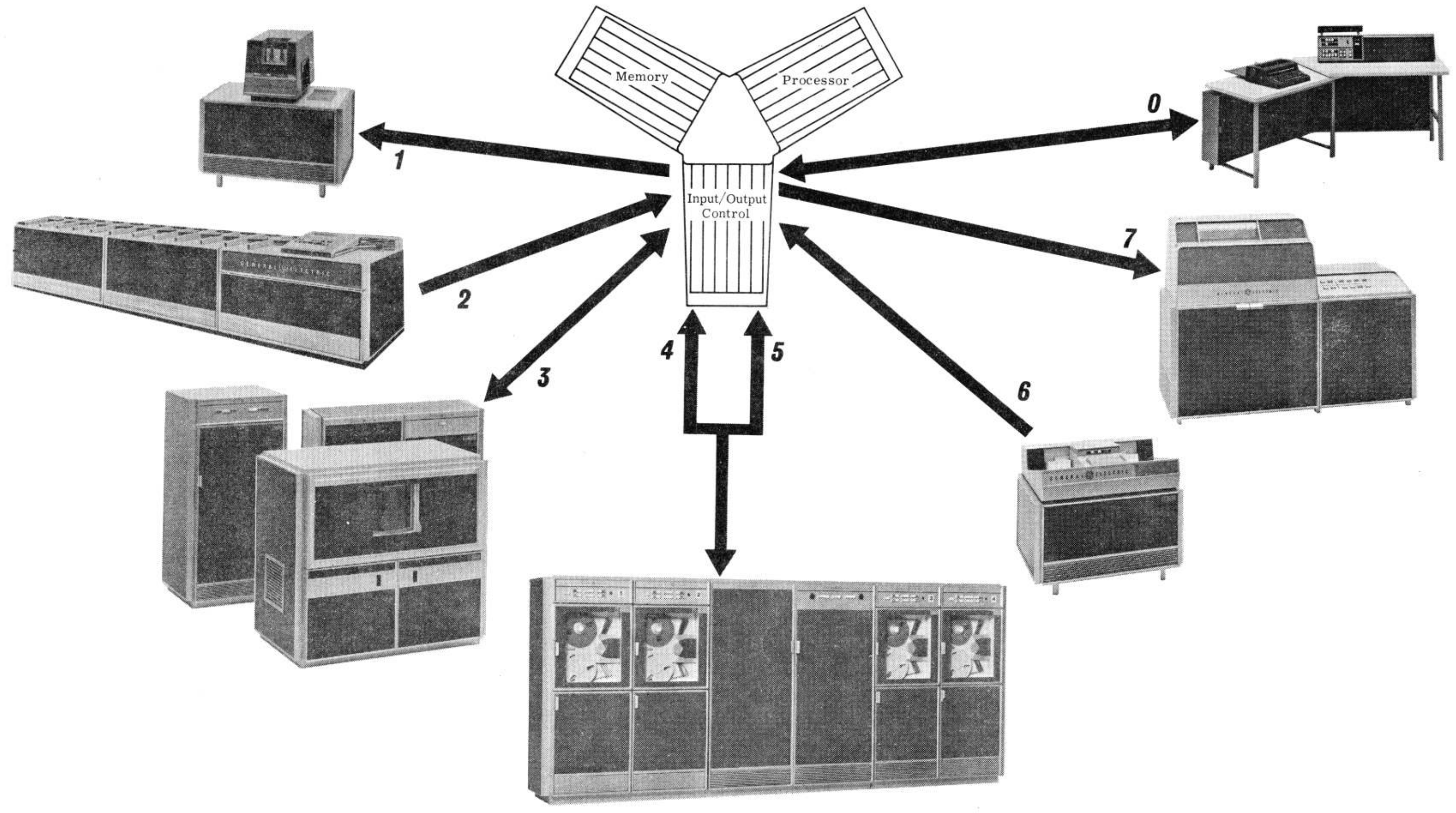


Figure I-1. GE-425 Functional Components

The basic unit of information is the word consisting of 24 bits (binary digits) plus a 25th bit used in memory for parity checking. Memory is a magnetic core memory available with 4k, 8k, 16k, or 32k words. Provision is made for as many as eight input/output channels, with each channel capable of servicing one peripheral subsystem. Data transfers between the peripheral subsystem and the I/O channels are serial by character. Both word and character buffered channels are available for data transfers between the I/O control section and memory. Memory communication with both the processor section and input/output channels is not restricted to fixed times with reference to the processor clock; memory control logic operates asynchronously with respect to the processor under control of its own clock.

Essential operator control of the central processor is provided through a minimum of switches and lights on a separate control console. Communication between the operator and central processor is provided by the input/output typewriter.

MEMORY SECTION

The primary storage device for the GE-425/435 systems is the magnetic core memory available in capacities of 4096, 8192, 16384, and 32768 words of 24 information bits (plus one even parity bit) each. The GE-425 memory has a memory access (read/write) cycle of 5.1 microseconds (usec), while the memory access cycle for the GE-435 memory is 2.7 usec. The modular design of the memory makes it possible to expand the smaller memory capacities to the larger sizes by on-site modification.

Core memory operates asynchronously with the central processor and contains its own timing and control logic and address register. Although the basic read-write memory access cycle is either 5.1 or 2.7 usec, many central processor instructions automatically take advantage of the split-cycle capability of memory (read only, write only, read-restore, clear-write) to reduce execution time. The programmer has no direct interest in this aspect of timing.

Memory Addressing

Each data word in memory can be individually accessed by specifying the location of the word in the address field of an instruction. Internal memory addressing is binary and instruction word address fields always consist of 15 bits. The Macro Assembly Program for the GE-425/435 automatically converts the programmer-generated decimal or symbolic addresses to binary. Memory addressing is binary rather than decimal to provide for efficient use of the bits in the instruction word. The instruction word address field can directly address any memory location from 0 through 32,767.

Core memory logic provides for signaling the central processor whenever an attempt is made to address a location beyond the capacity of the system memory (for example, any location higher than 8191 in an 8k memory). An attempt to access a non-existent memory location is treated as an invalid operation.

Memory Words

Each 24-bit memory word can contain any one of the following, depending upon machine interpretation:

1. Four 6-bit decimal characters
2. Four 6-bit alphanumeric characters
3. Twenty-four binary digits (equivalent to 8 octal characters).

Categories 1 and 2 are data word formats, while category 3 is normally (but not exclusively) reserved for instruction words and auxiliary words in the instruction processing sequence. Data word and instruction word formats are described in Section II, Word Formats.

Special Locations in Memory

Sixty-five of the first 72 memory locations have functions that restrict their use by the programmer. These restrictions are for either of two reasons:

1. Certain locations have special processor-oriented functions during operation

2. Other locations must not be used in GE-425/435 programs if GE-400 programming compatibility is to be maintained.

Specifically, the restricted locations and the reasons for the restrictions are (see Figure I-2):

<u>Locations 1 through 6</u>--These six locations are reserved for fixed index words which can be used in addressing and address modification, as described in Section IV. Although this does not prevent their use for other purposes, the programmer is cautioned against any use that can cause conflict in the program.

<u>Location 8</u>--This location is used by the central processor logic for working storage during execution of Shift and Move instructions. The programmer can not use location 8.

<u>Locations 10 and 11</u>--These two locations are reserved for the processor channel program interrupt control words. The programmer cannot use these locations for any <u>other</u> purpose.

<u>Locations 16 through 47</u>--These 32 locations are reserved for the input/output channel control words which are used to control memory access during data transfer operations and to service program interrupt requests by the input/output channels. The programmer cannot use these locations for other purposes.

<u>Locations 48 through 71--These locations must not be used by GE-425/435 programs if full GE-400 programming</u> compatibility is to be maintained.

Decimal Memory Locations	Reserved for		
0			
1	1	Fixed Index Words	
2	2		
3	3		
4	4		
5	5		
6	6		
7			
8	Used by Processor during Move and Shift instructions		
9			
10	Program Interrupt Word (PIW)	Processor Channel	
11	Program Interrupt Second Word (PSW)		
12			
13			
14			
15			
16	List Pointer Word (LPW)	Channel 0	Input/Output Channel Control Words
17	Data Control Word (DCW)		
18	PIW		
19	PSW		
44	LPW	Channel 7	
45	DCW		
46	PIW		
47	PSW		
48	Reserved for GE-400 Programming Compatibility		
71	Reserved for GE-400 Programming Compatibility		

Figure I-2. Special Locations in Memory

Memory Parity

Each time that a word is written into memory, parity is automatically generated. If the number of 1-bits is odd, a 1-bit is generated for the parity bit position; if the number of 1-bits is even, the parity bit position remains set to zero. Thus, all words in memory will have an even 1-bit count.

As words are read from memory, parity is checked. If the number of 1-bits in a word is odd, a parity alert is generated. Treatment of alert conditions is described in subsequent sections.

PROCESSOR SECTION

The processor section contains all logic necessary for accessing core memory and executing all operations as described in Section III, Instruction Repertoire. These include variable-length load and store, decimal and binary arithmetic, compares, shifts, branching, and logical modification instructions with both single-address and two-address capability. Variable-length operations involve single-, double-, triple-, and quadruple-length (4, 8, 12, and 16-character) fields.

Alphanumeric data is processed in binary-coded-decimal form, although certain binary arithmetic instructions are also provided. The variable-length, relocatable accumulator, in arithmetic operations, permits addition and subtraction of data fields consisting of 4, 8, 12, or 16 characters. The accumulator has sufficient capacity to perform multiplication of 8 by 8 character fields and division of 16 by 8 character fields.

Logic elements of particular significance to the programmer are:

1. Program counter (P-counter)
2. Accumulator location register
3. Accumulator length register.

The P-counter controls the sequence in which instructions are executed. It is a 15-bit register that normally holds the address of the next Program sequence memory location. The P-counter is automatically incremented by one during the time that a word in the P-sequence is being obtained from memory.

The P-counter registers sequential locations until a branch instruction is encountered in the P-sequence and the P-counter contents are replaced. Any branch instruction can change the contents of the P-counter. During a Store P and Branch instruction or a P-counter to Index and Branch instruction, the P-counter contents can be stored in a specified memory location for later recovery in order to permit a return to that point in the P-sequence.

The accumulator location register is a 13-bit register that holds the high-order 13 bits of the address of the most-significant word of the full accumulator. Under program control, the contents of this register can be changed, thereby causing a relocation of the accumulator. The Load Accumulator Location and Length (LAL) instruction establishes the location of the accumulator by replacing the contents of this register. By using a Store Accumulator Location and Length (SAL) instruction, the programmer can place the current accumulator location in memory for later recovery.

The accumulator length register is a 2-bit register that records the length of the working accumulator. The contents of this register are automatically set by any instruction that includes a set accumulator length function. The LAL instruction causes the two low-order bits of the LAL address field to be set into this register. The SAL instruction places the contents of the register in the two low-order bit positions of the memory location specified by the SAL address field.

Processor Capabilities

The processor section contains the logical elements necessary to execute decimal and binary arithmetic; bit and character shifts; data transfer, logical, and special instructions; and to perform address modification.

The basic operations that are performed by the processor section (and that in combination constitute the instruction repertoire) are:

1. Load and Store - These operations move information between memory locations. In load operations, information moves from selected memory locations into the memory locations currently being used as the accumulator. In store operations, information moves from the accumulator memory locations to other specified memory locations.

2. Addition - Instructions are provided to permit binary addition and both decimal addition and decimal subtraction.

3. Complement - Complement operations are performed automatically as required in the execution of decimal instructions. Complementing of binary values must be programmed.

4. Shift - Both bit and character shift instructions are provided. Shifting is accomplished by moving data from the accumulator, shifting, and returning the result to the accumulator. Shifting is fully described in Section III, Instruction Repertoire.

5. Branching - The P-counter contents can be changed as a result of tests of internal indicator condition. Changing the P-counter contents results in a branch in the program.

6. Logical Operations - Logical AND, OR, and exclusive OR operations are performed by loading the contents of two memory locations into internal registers, executing a combination of shifts of both registers, and placing the result in memory.

7. Edit - Edit operations are performed by simultaneously processing data to be edited and the accumulator contents (format). Data characters are gated into the accumulator under the control of the format characters. The edited result replaces the format characters in the accumulator.

In addition to the above basic operations, the processor section contributes to the execution of two special classes of instructions: the General and the Central Processor Operation instructions. These instructions are discussed in Section III, Instruction Repertoire.

Processor Control

The processor channel is associated with central processor operations and with program intervention by the operator. Its functions are:

1. To permit programmed storage of the status of certain central processor indicators which reflect various occurrences during processing.

2. To permit programmed setting of certain central processor indicators.

3. To permit program interruption and to control transfer to an interrupt subroutine upon the occurrence of certain central processor conditions.

By using a Request Status instruction, the programmer can cause the status of central processor indicators to be stored in any memory word for subsequent program testing. The programmer can set the central processor indicators on or off in a predetermined pattern by using a Set Status instruction. Together, these capabilities enable the establishment of a status word which, upon occurrence of a program interrupt, is used to retain the current status of the processor, and subsequently to restore that status. This status word can also be examined to determine the cause of the program interrupt.

Details of processor channel functions are contained in Section V, Processor Channel.

INPUT/OUTPUT CONTROL SECTION

The third major section of the central processor is the input/output control section which, under control of the processor section, performs the following:

1. Responds to input and output instructions and activates the peripheral subsystems.

2. Permits reading data into non-adjacent areas of memory (data scatter).

3. Permits writing data from non-adjacent areas of memory (data gather).

4. Controls time-sharing of memory access for data transfers in accord with established priorities in order to permit simultaneous operations.

5. Monitors data transmission between core memory and the peripheral subsystem.

6. Makes channel and peripheral status available to the central processor.

7. Initiates program interrupts upon detection of certain peripheral conditions in accord with established priorities.

The input/output control section provides for orderly data transfers between the various peripheral devices and the central processor. Each peripheral subsystem is assigned an input/output channel with a specific number, program interrupt priority, and data transfer priority. A channel receives commands from the central processor and transmits them to the peripheral device. A channel also transfers data from the peripheral device (or information relative to the device status) to memory and from memory to the peripheral device.

Data transfers are synchronized with memory and central processor operations. Also, selection and sequencing of pauses in instruction processing, such as for data transfers and program interrupts, must be controlled. The input/output control section performs these control functions in three ways:

1. Input/output command control - The input/output command control initiates peripheral operations. The General instruction specifies the operation to be performed and the channel and device to be used. The status of the channel and peripheral subsystem is stored in a location determined by the second address of the two-address General instruction.

2. Input/output control words - Four channel control words for each channel are stored in fixed memory locations. The words are used by the I/O logic during data transfers and during program interrupt operations. The list pointer word (LPW) and the data control word (DCW) determine the memory location to or from which data is to be transferred. The program interrupt word (PIW) and the program interrupt second word (PSW) are used to provide an automatic branch to an interrupt subroutine to service the condition which caused the interrupt.

3. Data transfer logic - Because several subsystems can be operating simultaneously, the information which passes between the peripheral device and the channel must wait its turn for access to memory in order to transfer data. The data transfer logic determines the order in which the channels can access memory. Selection is made on the basis of channel priority. When simultaneous data transfer requests occur, the channel with the highest priority is honored first. After a request is honored, the DCW and LPW words are used to control data flow.

Functions of the input/output control section are described in detail in Section VI, Input/Output Processing.

SYSTEM CONTROL AND COMMUNICATION

The GE-425/435 control consoles each include an operator panel, a maintenance panel, and an input/output typewriter. Figure I-3 illustrates the GE-425 Control Console. The console and typewriter provide the primary means for communication between the computer and the operator. The typewriter is connected to a standard character buffered input/output channel which is always the lowest-priority channel (zero). In addition to the channel connection to the central processor, other lines bring power to the console unit and connect control panel switches and lights to central processor logic.

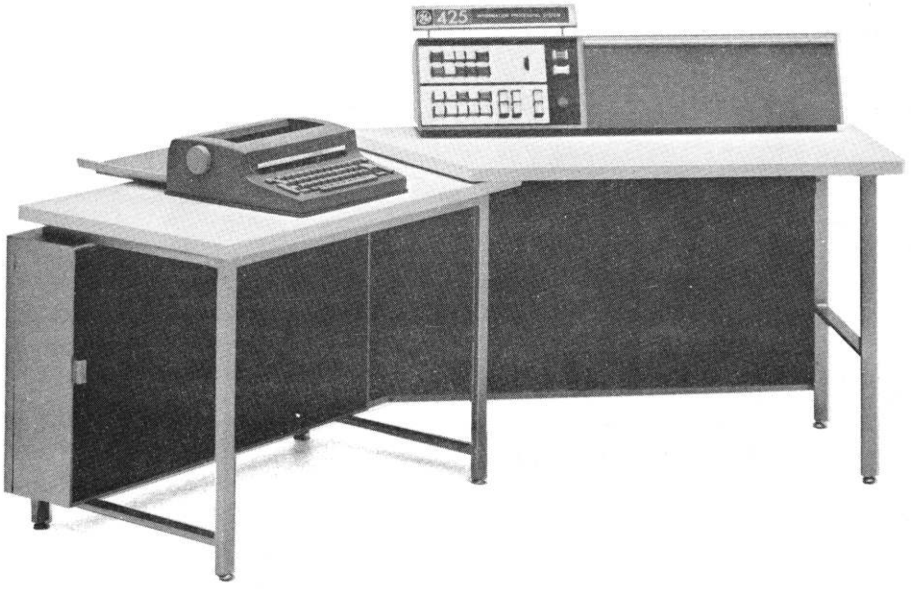


Figure I-3. GE-425 Control Console and Console Typewriter

PERIPHERAL EQUIPMENT

All input/output devices used with the GE-425/435 communicate with the central processor and core memory through input/output channels under the control of the I/O control section. The GE-425/435 can have as many as eight such channels, with one always reserved for the I/O typewriter. As many as seven additional peripheral subsystems with one or more I/O devices can be connected to the remaining channels.

Input and output media can be selected from the following:

1. Magnetic tape (input and output) - The MT-20 Magnetic Tape Unit has a forward tape speed of 75 inches per second and data transfer rates of 15,000 and 41,666 characters per second. As many as eight MT-20's can be connected to one MTC-20 Magnetic Tape Controller. As many as eight MT-20's can be connected to an MTC-21 Magnetic Tape Controller for a dual channel peripheral subsystem providing greater flexibility. Refer to the MT-20 Magnetic Tape Unit Reference Manual for operating and programming details.

2. Printer - The PR-20 Printer prints 1200 lines per minute from a selection of 46 most-used characters, or 900 lines per minute from the full GE-425/435 character set. Print line length is 136 characters, 10 characters per horizontal inch and either 6 or 8 print lines per vertical inch. One PR-20 Printer connects to an input/output channel. Refer to the PR-20 Printer Reference Manual for operating and programming details.

3. Card reader - The CR-20 Card Reader can read standard 80-column punched cards at 900 cards per minute, either in Hollerith, binary, or mixed mode. One card reader connects to one input/output channel. Refer to the CR-20 Card Reader Reference Manual for operating and programming details.

4. Card punch - The CP-10 Card Punch punches standard 80-column cards at 100 cards per minute in either Hollerith, edited Hollerith, or binary mode. One card punch connects to one input/output channel. Refer to the CP-10 Card Punch Reference Manual for operating and programming details.

5. Perforated tape (input and output) - The TR-20 Tape Reader and TP-20 Tape Punch, respectively, can read and punch 5-, 6-, 7-, and 8-level tape. Tape is read at the rate of 500 characters per second, and punched at the rate of 150 characters per second. The TR-20 and TP-20, with a tape spooler, are available singly or together in one cabinet. The TR-20 and TP-20, in combination, comprise the TS-20 Tape Reader/Punch which can be connected to a single input/output channel. Refer to the TS-20 Tape Reader/Punch Reference Manual for operating and programming details.

6. Magnetic disc storage (input and output) - The DS-20 Disc Storage Unit provides up to 23.5 million alphanumeric characters of random access storage. As many as four DS-20's can be connected to a DSC-20 Disc Storage Controller, providing 94 million characters of storage. One DSC-20 Controller connects to a single input/output channel. Refer to the DS-20 Disc Storage Unit Reference Manual for operating and programming details.

7. Magnetic ink documents (input only) - The MR-20 Magnetic Reader/Sorter reads 1200 documents per minute and sorts them into 12 pockets. One MR-20 connects to a special input/output channel, called the magnetic reader/sorter channel. Refer to the MR-20 Magnetic Reader/Sorter Reference Manual for operating and programming details.

Within certain constraints, as many as seven peripheral subsystems (in addition to the input/output typewriter) can operate simulatneously while they time-share memory access and while instruction processing continues. Channels have priorities for access to memory, as described in Section VI, Input/Output Operations. This section includes descriptions of standard and special input/output channels, as well as details of input/output operations.

MAJOR FEATURES

Several features of the GE-425/435 systems are of particular significance. These include:

1. Variable-length, relocatable accumulator
2. Edit Instruction
3. Scatter-Gather ability
4. Address Modification
5. Program Interrupt Logic
6. Checking Capability.

Variable-length, Relocatable Accumulator

The accumulator in the GE-425/435 serves several functions. It holds:

1. One of the two operands in single-address arithmetic operations.
2. The result of certain arithmetic operations.
3. Data to be shifted.
4. Data to be imploded into memory and the result after data has been exploded from memory.
5. Edit format characters prior to editing.
6. Result characters after editing.
7. One of two operands in certain compare operations.

The full accumulator is four adjacent memory locations that are assigned by the program. The programmer can relocate the accumulator in any group of four adjacent locations whose most-significant location is evenly divisible by four (0, modulo 4). Relocating the accumulator does not affect the contents of the memory locations involved.

In addition, the working (or effective) length of the accumulator can be set to single, double, triple, or quadruple word length. Regardless of the working length setting, the full accumulator always corresponds to four locations. The setting of the working length determines the length of the memory field affected by certain operations involving the accumulator. Instructions are provided for setting the working length alone or in combination with other operations.

Advantages provided by the relocatable, variable-length accumulator are significant, and include:

1. The accumulator can be moved to the data to be processed, rather than moving the data. Relocating the accumulator, which does not require memory accesses, is generally faster than data movement.

2. The accumulator working length can be adjusted to the data, thereby facilitating control and saving processing time.

3. Relocatability of the accumulator minimizes instructions required for unpacking, editing, and packing data.

For convenient referencing, the four words of the full accumulator are designated as words D, C, B, and A, ranging from the most-significant to the least-significant word. The accumulator and related terminology are illustrated graphically in Figure I-4.

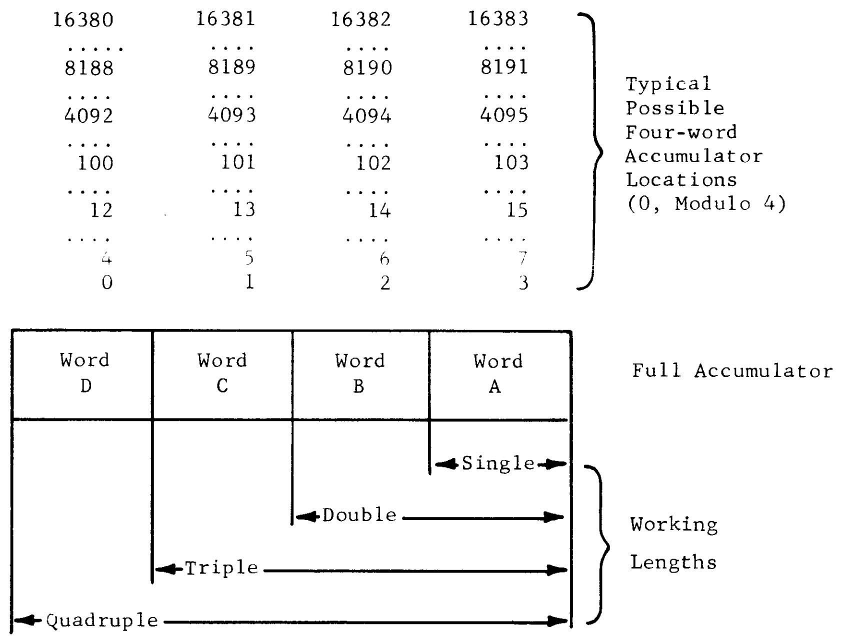


Figure I-4. Relocatable Accumulator

Note that, regardless of working length, the least-significant word of the accumulator is always word A and the actual length of the full accumulator is always four words.

In addition to instructions for setting accumulator length and location, other instructions are provided for remembering (storing) and re-establishing (loading) previous accumulator lengths and locations. These instructions are described in Section III, Instruction Repertoire.

Edit Instruction

The GE-425/435 systems have the ability to edit output data by suppressing zeros, deleting data characters, and inserting punctuation marks and special characters into the data. This ability is provided automatically through the Edit (EDT) instruction.

The Edit instruction processes individual data characters from memory under the control of format characters that are held in the relocatable accumulator. The result of editing is placed in the accumulator. The contents of the memory data field are unchanged.

During editing, three types of zero suppression are possible under the control of format characters:

1. Simple suppression
2. Suppression with Asterisk Protection
3. Suppression with Floating Dollar Sign.

Suppression causes the replacement of leading zeros, commas, and periods by spaces, asterisks, or spaces followed by a dollar sign. The Edit instruction is discussed in Section III, Instruction Repertoire.

Scatter-Gather

Because the input/output control section uses a list of words in memory (called a data control list) to determine the memory locations involved in data transfer operations, it is possible for the programmer to elect to read an input record into non-adjacent memory fields (scatter), or to write a single output record from non-adjacent memory fields (gather).

Such a technique has the advantage of reducing the number of intramemory data moves necessary and of reducing the amount of memory that need be allotted for working storage. The scatter-gather technique is illustrated in Section VI, Input/Output Processing.

Address Development and Modification

The GE-425/435 instruction repertoire includes both single-address and two-address instructions. Single-address instructions specify a single operand or memory location; two-address instructions involve two operands. Both types of instructions can require one or more words in the processing sequence (P-sequence), depending upon the operation code and address control field (ACF) of the instruction word. See Section II, Word Formats, for an illustration of the instruction word format.

During instruction processing, the central processor follows this general sequence:

1. The instruction word is obtained from memory.
2. Any modification specified by the instruction word ACF is performed, accessing words outside the P-sequence if necessary.
3. If the operation code identifies the instruction as a two-address instruction, the second address is obtained.
4. The instruction is executed.
5. The next instruction is obtained.

First-address modification capabilities, as controlled by the instruction word ACF and Indirect Address Field, include:

1. Fixed Index Word (FIW) modification - Adds the contents of one of six FIW's to the basic instruction.
2. Indexing or modification through an Address Modification Sequence (AMS) - If the instruction word ACF specifies an AMS, the next P-sequence word will be one of the following:

   a. An Index, whose address field is added to the address field of the basic instruction.

   b. An Index Pointer, whose address field specifies the location of the indexing quantity to be added to the address field of the basic instruction.

   c. An Index Link, whose address field specifies a location outside the P-sequence that contains an Index, Index Pointer, or another Index Link.
3. Indirect Addressing - Modification by replacement of the address field with the contents of any word in memory. After any Address Modification Sequence, the P-sequence AMS word that initiated the sequence is examined to determine whether indirect addressing is specified. If so, the developed address becomes the address of the location containing the replacement address field. The indirect address can also be modified.

The address field of single-address instructions can be modified by any of the above three methods. The first address of two-address instructions can be modified by an AMS and/or indirect addressing, but not by using a fixed index word. The second address cannot be modified.

The second address of a two-address instruction is generally specified by a Second Address Sequence (SAS) that is initiated with an SAS word in the P-sequence after any first-address modification is performed. The P-sequence SAS word can be an:

1. Operand, whose actual location address is the second address.

2. Operand Pointer, whose address field contains the second address.

3. Operand Link, whose address field contains the address of another SAS word which can be an Operand, Operand Pointer, or another Operand Link.

A special class of two-address instructions requires no SAS. Instead, the second address is specified in the instruction word ACF to be one of the six fixed index words. These instructions are described under the heading, Fixed Index Word Instructions, in the instruction repertoire.

Extensive address manipulation is possible and can provide a useful supplement to the basic addressing capabilities. Address modification can materially reduce total processing time and program memory requirements, and is discussed in Section IV, Addressing and Address Modification.

Program Interrupt

The program interrupt logic provides for the selection, initiation, and sequencing of automatic program interrupts on a priority basis. The processor channel has the highest priority, and input/output channel priorities are (from high to low), one through seven, and zero. Upon interrupt, the central processor executes a branch to the program interrupt word for the requesting channel. The contents of the program interrupt word determine whether the interrupt condition is to be ignored or whether a program interrupt routine is to be entered. Program interrupts for the processor channel are discussed in Section V, Processor Channel. Input/output channel program interrupts are described in Section VI, Input/Output Processing.

Checking

A high level of accuracy and reliability are provided in the GE-425/435 systems.

The GE-425/435 can be programmed so that a detected error will initiate a program interrupt that does not halt the system but causes the program to enter a recovery routine.

For example, if a parity alert is detected during a magnetic tape read operation, an alert indicator in the processor is set by the magnetic tape controller. When the status of the magnetic tape unit is examined upon conclusion of the operation, a branch occurs to a recovery routine to attempt to reread the tape under program control. The programmer can select the method for handling alerts that is best suited to his program. Other peripheral operations can be continued while the alert condition is serviced. The programmer can elect to have the operator informed of alerts through programmed messages on the input/output typewriter. Central processor alerts are discussed in Section V, Processor Channel. Input/output alerts are discussed in appropriate peripheral equipment reference manuals.

II. WORD FORMATS

Although the GE-425 and the GE-435 are both decimal computer systems, knowledge of the binary and the octal number systems, as well as the better-known decimal system, is helpful to the programmer. Acquaintance with the binary number system is important because:

1. Internal memory addressing is in binary notation and, although the Macro Assembly Program permits addresses to be represented by the programmer in decimal or symbolic notation, there are occasions when it is desirable to do binary address manipulation.

2. The GE-425 and the GE-435 have binary arithmetic and logical capabilities that require a knowledge of binary notation for their most effective use.

A knowledge of the octal number system is also useful, because it provides a more convenient equivalent of the binary number system. It also facilitates conversions between binary and decimal notation. The Appendix contains a table of powers of two and procedures for converting numbers from one system to another.

The basic unit of information within the GE-425/435 is the word, which consists of 24 information bits (plus one parity bit when the word appears in memory).

GE-425/435 words belong to three general categories as listed below:

1. Data words
2. Instruction words
3. Auxiliary words for addressing and control.

Each category is discussed in the balance of this section.

DATA WORDS

Within the GE-425/435 system, alphanumeric data is represented in BCD (binary-coded-decimal) notation. The alphanumeric word contains 24 bits numbered 0 through 23 from right to left. The 24-bit word is divided into four characters, each consisting of six bits. The four alphanumeric characters are designated as 0, 1, 2, and 3 from left to right, as illustrated in Figure II-1.

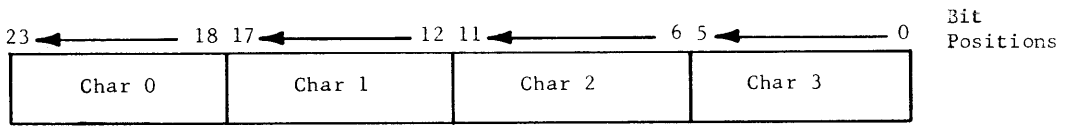


Figure II-1. GE-425/435 Alphanumeric Word

Each character of the alphanumeric word consists of two zone bits and four numeric bits in the format shown in Figure II-2.

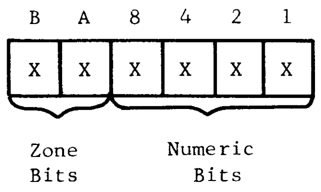


Figure II-2. BCD Character Format

Each zone and numeric bit position can contain either a 0-bit or a 1-bit. A unique bit combination is provided for each character of the GE-425/435 character set which is illustrated in Figure II-3. Note that all numeric characters contain 00 zone bit coding.

Printer Characters	BCD Code	Octal Code	Hollerith Card Code
0	00 0000	00	0
1	00 0001	01	1
2	00 0010	02	2
3	00 0011	03	3
4	00 0100	04	4
5	00 0101	05	5
6	00 0110	06	6
7	00 0111	07	7
8	00 1000	10	8
9	00 1001	11	9
‖~‖	00 1010	12	2-8
‖#‖	00 1011	13	3-8
@	00 1100	14	4-8
:	00 1101	15	5-8
‖>‖	00 1110	16	6-8
‖(Ɨ)‖	00 1111	17	7-8
‖(△)‖ Space	01 0000	20	(blank)
A	01 0001	21	12-1
B	01 0010	22	12-2
C	01 0011	23	12-3
D	01 0100	24	12-4
E	01 0101	25	12-5
F	01 0110	26	12-6
G	01 0111	27	12-7
H	01 1000	30	12-8
I	01 1001	31	12-9
&	01 1010	32	12
.	01 1011	33	12-3-8
⌑	01 1100	34	12-4-8
(	01 1101	35	12-5-8
<	01 1110	36	12-6-8
|	01 1111	37	12-7-8

Printer Characters	BCD Code	Octal Code	Hollerith Card Code
↑	10 0000	40	11-0
J	10 0001	41	11-1
K	10 0010	42	11-2
L	10 0011	43	11-3
M	10 0100	44	11-4
N	10 0101	45	11-5
O	10 0110	46	11-6
P	10 0111	47	11-7
Q	10 1000	50	11-8
R	10 1001	51	11-9
‖-‖	10 1010	52	11
$	10 1011	53	11-3-8
*	10 1100	54	11-4-8
)	10 1101	55	11-5-8
‖;‖	10 1110	56	11-6-8
‖'‖	10 1111	57	11-7-8
+	11 0000	60	12-0
/	11 0001	61	0-1
S	11 0010	62	0-2
T	11 0011	63	0-3
U	11 0100	64	0-4
V	11 0101	65	0-5
W	11 0110	66	0-6
X	11 0111	67	0-7
Y	11 1000	70	0-8
Z	11 1001	71	0-9
→	11 1010	72	0-2-8
,	11 1011	73	0-3-8
%	11 1100	74	0-4-8
=	11 1101	75	0-5-8
"	11 1110	76	0-6-8
(Λ)	11 1111	77	0-7-8

Characters inside ‖x‖ have special Edit functions
Characters inside (x) normally do not print.

Figure II-3. GE-425/435 Standard Character Set

In decimal arithmetic operations, the zone bits of all character positions of the word, except character 3, are ignored. The character 3 zone bits indicate the sign (plus or minus) of the entire word, not the sign of character 3. A minus sign is indicated by the binary configuration, 10, in the zone bits of character 3 (bits 5 and 4). If these two bit positions contain any other configuration (00, 01, or 11), the entire word is regarded as positive. The format of a word as it is treated in decimal operations is shown in Figure II-4.

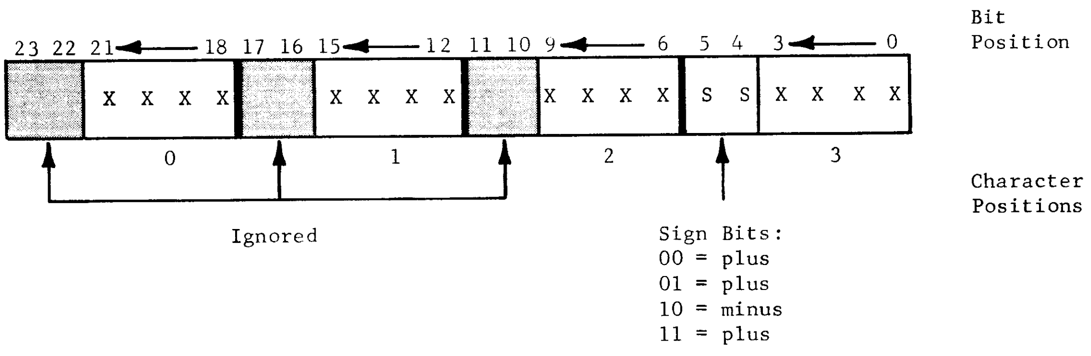


Figure II-4. GE-425/435 Decimal Arithmetic Word

If a decimal data field is longer than one word, then the sign of the entire field is contained in character 3 of the right-most (least-significant) word of the field. Zone bits of all other characters of the field are ignored automatically.

Although the GE-425 and GE-435 are essentially alphanumeric computers, both have certain binary capabilities that permit the use of binary data words. In binary data words, the entire 24 bits are treated as a single unit of information; there is no separation into characters as in alphanumeric words. The binary data word format is illustrated in Figure II-5.

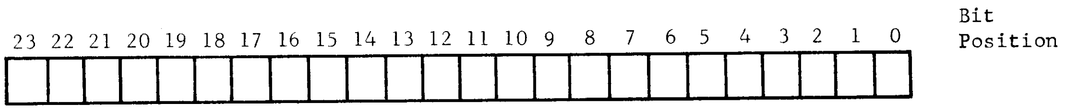


Figure II-5. Binary Data Word

Binary words are automatically treated as positive quantities, and no provision is made for indicating sign information. The 24-bit word provides for positive values ranging from zero to 16,777,215 (decimal). Note in Figure II-5 that the number designation for each bit position matches the corresponding power of two for that bit position; for example, bit position 23 corresponds to 2^{23} Binary arithmetic involving negative quantities can be performed by the programmer if appropriate programming conventions (including two's complement notation) are adopted and the overflow condition is properly interpreted.

INSTRUCTION WORDS

The GE-425 and the GE-435 are stored-program computers; that is, data processing is performed under the control of a series of instructions stored in core memory and executed (usually in sequence) one at a time. The sequence in which instructions are executed is called the program sequence, or P-sequence, and is controlled by the program counter, or P-counter

Each word taken from memory under P-counter control is normally either an instruction word or an auxiliary word. Auxiliary words are described under the next heading.

The instruction word serves these functions:

1. Defines the operation or program step to be performed.
2. Identifies the instruction as either a single-address or a two-address instruction.
3. Specifies an operand or an address.
4. Indicates the type of address control (if any) to be exercised in determining the effective address.

The instruction word, like all GE-425/435 words, consists of 24 bits of the format illustrated in Figure II-6.

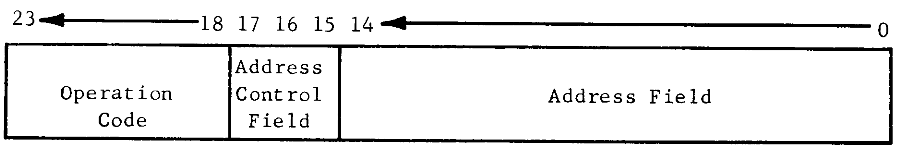


Figure II-6. GE-425/435 Instruction Word Format

The operation code field (bits 18 through 23) identifies the operation to be performed (add, subtract, shift, etc.) and determines whether the instruction is a single-address or a two-address instruction. The address control field (ACF), in conjunction with the operation code, determines the type of address modification (if any) to be performed, or for certain two-address instructions specifies the second address. The address field normally contains a memory address which is either a tentative or an effective (final) address, depending upon whether or not address modification is involved.

The operation codes for all GE-425/435 instructions are explained in Section III, Instruction Repertoire. The use of the ACF is described in Section IV, Addressing and Address Modification.

AUXILIARY WORDS

Each instruction word, under control of its ACF, may refer to a fixed index word (see Figure II-7a), or be followed directly in the P-sequence by one or more Address Modification Sequence (AMS) words, as illustrated in Figure II-7b. Depending upon the instruction, either the fixed index word contents modify the tentative address of the instruction word, or the fixed index word itself is the second address in two-address instructions. The P-sequence AMS word is invariably used to determine a modifier for the first address of the instruction.

In most two-address instructions, the instruction word (and any associated AMS words) is followed in the P-sequence by a Second Address Sequence (SAS) word that determines the second address. Figure II-7f illustrates the format of the P-sequence SAS word.

The P-sequence AMS and SAS words, depending upon their type, or class, can cause the program to refer to remote AMS and SAS words located in memory but outside the P-sequence in order to determine the address modifier or the second address for the instruction. The formats for remote AMS and SAS words are shown in Figures II-7c and 7g, respectively.

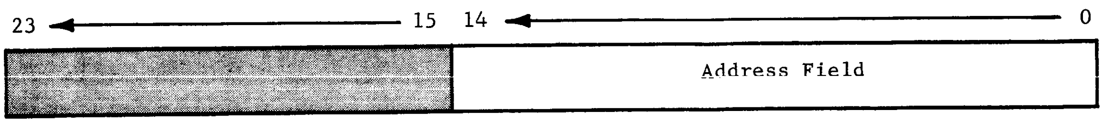
If the indirect addressing capability of the GE-425/435 is used, a P-sequence AMS word can also specify (in its indirect address field, bit 21) an Indirect Address word (IAW) to establish a replacement address field for the tentative address. The IAW can also refer to an indirect AMS word, under control of its ACF, for further address modification and/or address replacement. The formats for the IAW and the indirect AMS word are shown in Figures II-7d and 7e, respectively.

These auxiliary words are useful to the programmer for first address modification and for second address determination. Their uses are discussed in Section IV, Addressing and Address Modification.

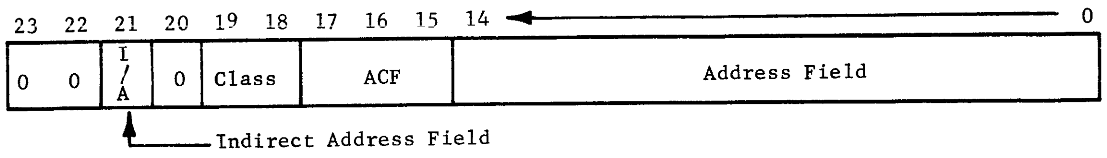
In the illustrations of the formats of the auxiliary words, Figure II-7, the bit positions shown to contain zeros must contain zeros, either because of GE-425/435 hardware requirements, or to maintain program compatibility with planned additions to the GE-400 family of computers. The shaded bit positions of the auxiliary words are ignored by the GE-400 instruction control logic.

Additional auxiliary words (such as counter control words) are required by specific instructions. Their uses and formats are described in Section III, Instruction Repertoire.

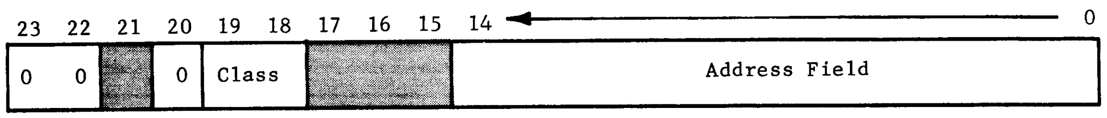
a. Fixed Index Word (when used for address modification):



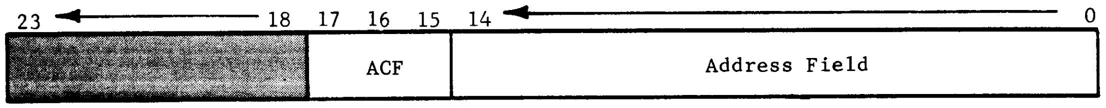
b. P-Sequence AMS Word:



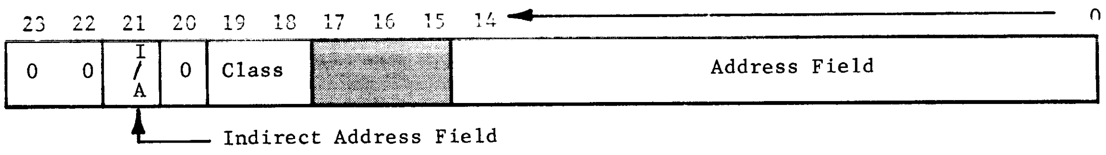
c. Remote AMS Word:



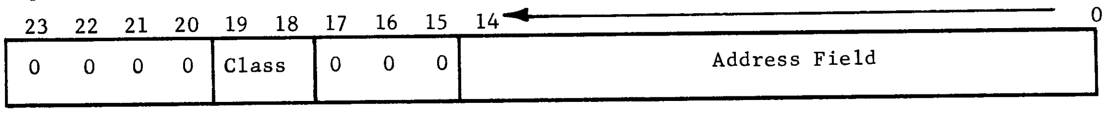
d. Indirect Address Word:



e. Indirect AMS Word:



f. P-Sequence SAS Word:



g. Remote SAS Word:

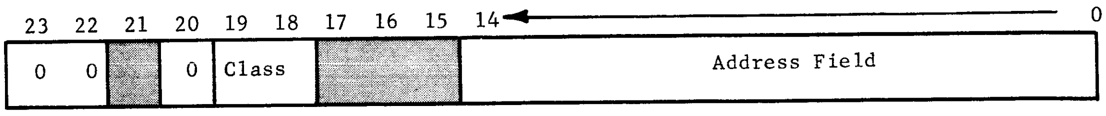


Figure II-7. Auxiliary GE-425/435 Words

III. INSTRUCTION REPERTOIRE

Tnis section presents the full instruction set for central processor operations in the GE-425/435. Input/output instructions are briefly described in order to provide overall perspective. The individual peripheral subsystem reference manuals contain more complete descriptions of peripheral operations.

The functional description of each instruction is presented, together with simple examples illustrating the effect of instruction execution. The format in which each instruction is presented is related in part to the coding sheet used with the Macro Assembly Program.

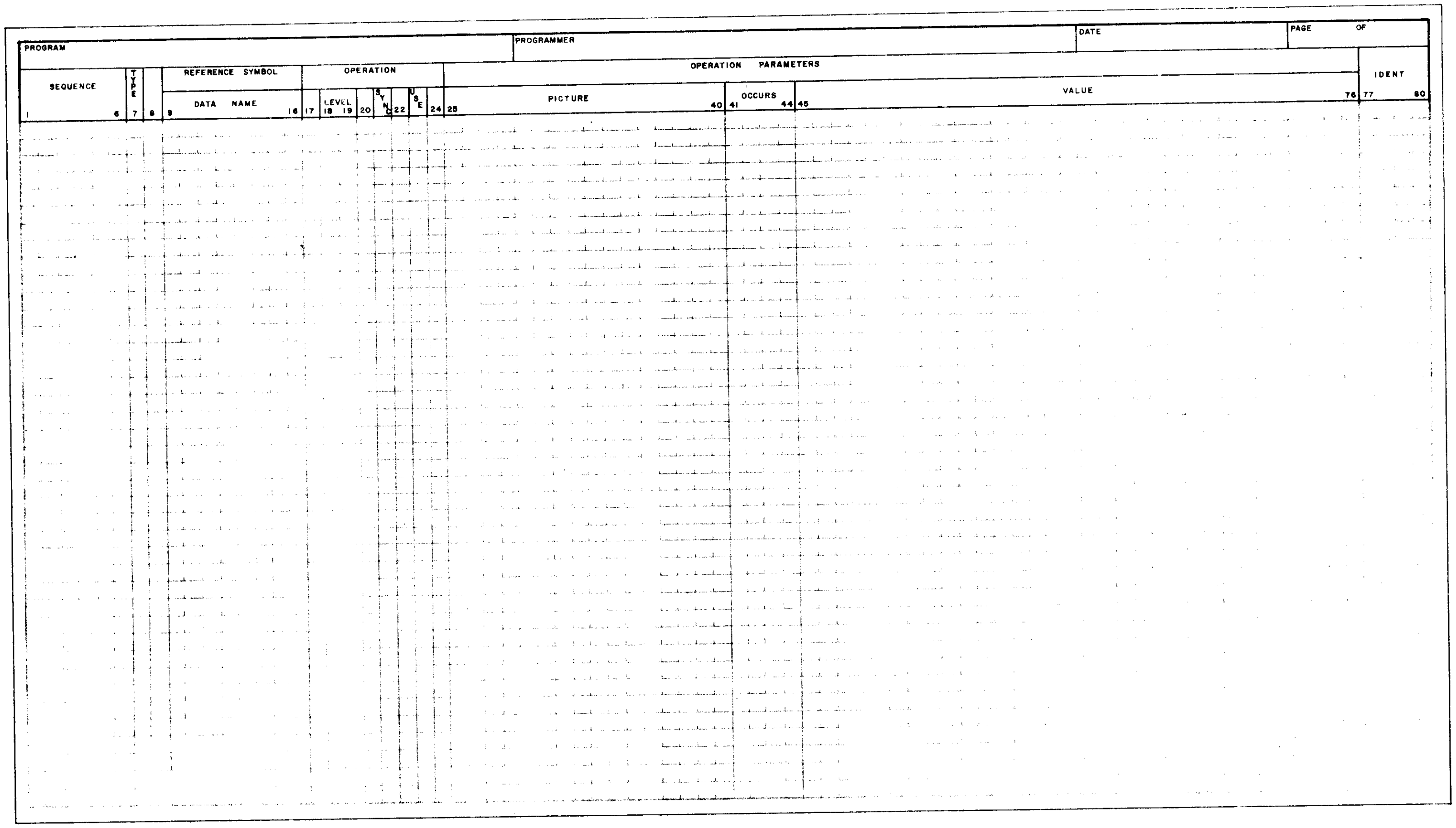
ASSEMBLY LANGUAGE PROGRAMMING

A part of the programming language made available by the Macro Assembly Program is the basic assembly language which provides the user with a symbolic language having a one-for-one correlation with the GE-425/435 instruction repertoire. The basic assembly language consists of a set of mnemonic operation codes, one code for each computer operation. In addition, mnemonic codes for pseudo-operations are provided for memory allocation, production of constants, assembly control, and miscellaneous functions. Some of these pseudo-operations are briefly mentioned in this section with the instructions to which they are directly related. For a full description of Macro Assembly Program capabilities, refer to the reference manual for the GE-425/435 Macro Assembly Program.

The coding form provided for use with the macro assembly language is illustrated in Figure III-1. Only its use with the basic assembly language portion is described here. The coding sheet is arranged in source program card image format, as follows:

Sequence (columns 1 through 6) - The programmer has the option of placing sequence numbers on his source program cards within this 6-character field.

Type (column 7) - An asterisk in this field signifies that the entire card contains only remarks.



PROGRAM						PROGRAMMER		DATE	PAGE OF	
SEQUENCE	TYPE		REFERENCE SYMBOL	OPERATION				OPERATION PARAMETERS		IDENT
			DATA NAME		LEVEL	SYNC	USE	PICTURE	OCCURS	VALUE
1 6	7	8	9 16	17	18 19	20 22	24	25 40	41 44	45 76

IDENT 77 80

Figure III-1. Macro Assembly Program Coding Sheet

Reference Symbol (columns 9 through 16) - This card field enables the programmer to assign symbols to program entries, such as instructions and constants. The Macro Assembly Program equates actual memory locations with the symbols.

Operation (columns 17 through 24) - This field is the portion of the coding sheet in which the user writes the mnemonic codes for the desired instructions and pseudo-operations.

Operation Parameters (columns 25 through 76) - This field will contain absolute or symbolic expressions of information required to complete the instruction specified in the Operation field, and must be left-justified. Comments may also be inserted following the first blank column after the Operation Parameters entry. For most instructions, this field will contain the symbolic or absolute address for the instruction, followed by identification of the type of address control (ACF), if any. The address and address control must be separated by a comma.

Identification (columns 77 through 80) - This field provides up to four characters of program identification. The field is not checked by the assembly program, but is printed on the assembly listing.

The headings in the second line (Data Name, Level, Sync, etc.) are related to Macro Assembly Program capabilities beyond the basic assembly language and are described in the reference manual for the GE-425/435 Macro Assembly Program.

FORMAT OF INSTRUCTION DESCRIPTIONS

Each instruction in the repertoire is introduced in this section by a standardized heading that illustrates both the coding sheet format and the absolute octal format of the instruction. For brevity, certain symbol conventions are used. See Figure III-2, which contains a duplication of a heading for a representative two-address instruction.

MOVE FROM FIRST MEMORY			(line 1)
MFM	Y/Y,X	30XYYYYY	(line 2)
O/OP/OL	Z	0C0ZZZZZ	(line 3)

Figure III-2. Sample Instruction Repertoire Heading

Line 1 of the heading contains the instruction name, which is a brief description of the operation performed.

Line 2 contains three items:

1. The mnemonic code (MFM, in the illustration) for the Operation field of the coding sheet.

2. An indication of the coding sheet Operation Parameters field requirement for the operation. If this space is blank, no operation parameter is required. If it contains a Y/Y,X, the field must contain a symbolic or absolute address field, and may contain, in addition, an expression for the type of address control required. The Y represents the address of single-address instructions and the first address of two-address instructions. If the heading contains Y,X only, and not Y/Y,X, the coding sheet Operation Parameters field must contain both an address field expression and an expression specifying one of the six fixed index words. Only two-address instructions referencing a fixed index word for the second address must have some address control designation.

3. A representation of the octal format of the instruction. The two most-significant digits (30, in the example) are always the octal operation code. The X (third octal digit from the left) represents the address control field and can be any digit, 0 through 7, depending upon the type of address control required. Address control is discussed in Section IV, Addressing and Address Modification. The YYYYY portion of the octal instruction word corresponds to bits 0 through 14 of the binary instruction word (the address field).

For single-address instructions, line 3 of the heading is left blank. Only two-address instructions not referencing a fixed index word must contain a second line of coding as represented in line 3 of the heading.

When used, line 3 contains:

1. The mnemonic for the P-sequence SAS word. For clarity, this entry should be indented to column 21 on the coding sheet. The heading entry in the example, O/OP/OL, indicates that either an Operand (O), an Operand Pointer (OP), or an Operand Link (OL) can be used with this instruction. For certain instructions, an Operand (O) cannot be used. In these cases, the heading only indicates OP/OL.

2. The heading, Z, indicating the Operation Parameters entry for the coding sheet. This entry is usually a symbolic or absolute address.

3. The octal representation of the P-sequence SAS word. The digits indicated as zeros must be zeros. The C represents the class field of the SAS word and is either 0, 1, or 2 for an Operand, Operand Pointer, or Operand Link, respectively. The ZZZZZ corresponds to the address field of the SAS word.

Following the heading for each instruction is the functional description that details the execution of the instruction. The functional descriptions are, for most instructions, followed by one or more illustrative examples containing a segment of the coding sheet and the before and after condition of registers and memory locations affected.

In illustrations of instructions affecting the accumulator, the full accumulator is shown as four connected blocks with a diamond marking the working length of the accumulator. To illustrate, Figure III-3 shows the accumulator with the working length set to triple.

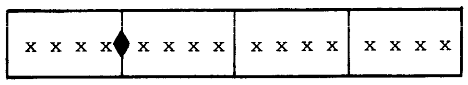


Figure III-3. Accumulator Representation

INSTRUCTION REPERTOIRE ORGANIZATION

In the remainder of this section, instructions are grouped according to function. The major groups are:

1. Data Movement
2. Shift
3. Arithmetic
4. Compare
5. Branch
6. Logic
7. Fixed Index Word
8. Accumulator
9. Special Purpose
10. Peripheral

The Special Purpose category includes the Central Processor Operations, Edit, Halt, and General instructions. The Appendix contains both alphabetic and octal indexes of the instructions, referencing the pages in this section on which the instructions are discussed.

DATA MOVEMENT INSTRUCTIONS

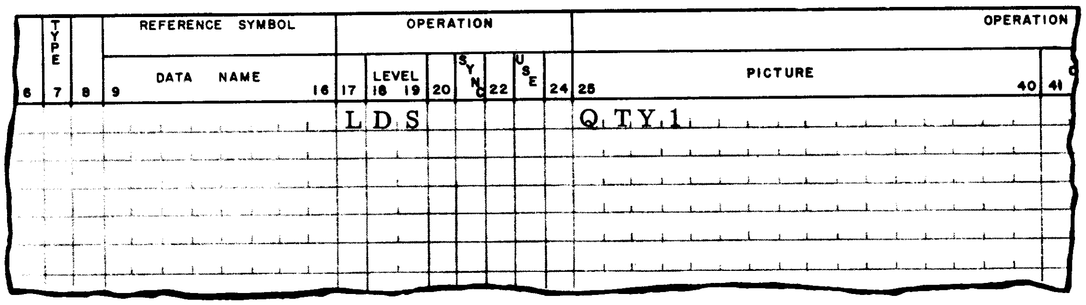
LOAD SINGLE

LDS	Y/Y, X	40XYYYYY

The accumulator length is set to single and the contents of location Y replace the contents of the single accumulator. The contents of Y are not changed.

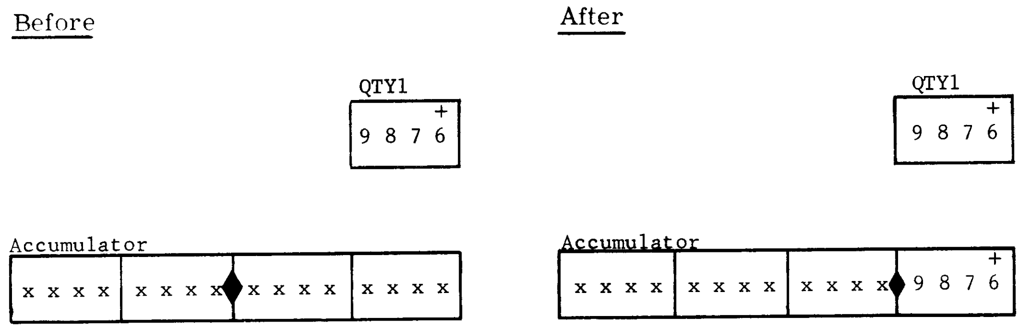
Example:

The working accumulator is double. Set the accumulator to single and load the contents of the one-word memory field in symbolic location QTY1 into the single accumulator. QTY1 contains +9876.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC	22	USE	24	PICTURE 25–40	41
				L	DS						QTY1	

Locations Affected:


Before
After
QTY1
+
9 8 7 6
QTY1
+
9 8 7 6
Accumulator
x x x x | x x x x | x x x x | x x x x
Accumulator
x x x x | x x x x | x x x x | 9 8 7 6
+

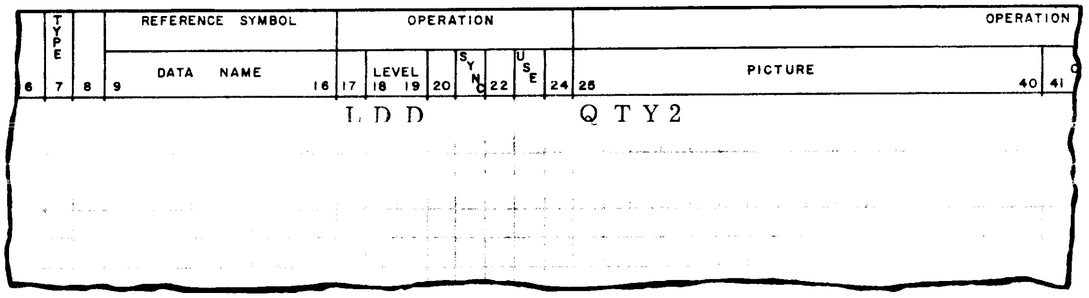
LOAD DOUBLE

LDD	Y/Y, X	41XYYYYY

The accumulator length is set to double and the contents of locations Y-1 and Y replace the contents of the double accumulator. The contents of Y-1 and Y are not changed.

Example:

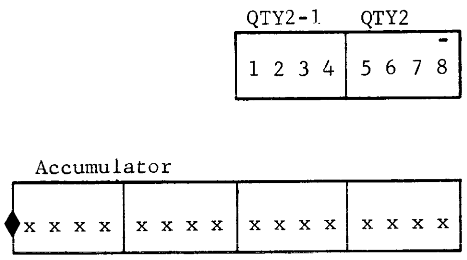
The working accumulator is quadruple. Set the accumulator to double and load the contents of the two-word memory field in symbolic locations QTY2-1 and QTY2 into the double accumulator. QTY2-1 and QTY2 contain -1234 5678.



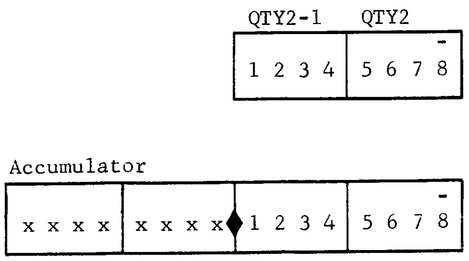
6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 22	USE 24	OPERATION / PICTURE 25–40	41
				LDD					QTY2	

Locations Affected:

Before



After



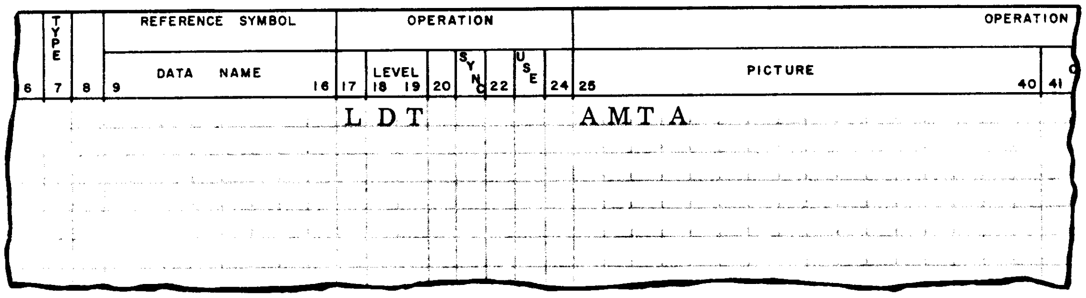
LOAD TRIPLE

LDT	Y/Y, X	42XYYYYY

The accumulator length is set to triple and the contents of locations Y-2, Y-1, and Y replace the contents of the triple accumulator. The contents of the memory field are not changed.

Example:

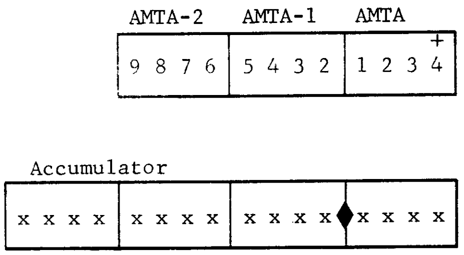
The working accumulator is single. Set the accumulator to triple and load the contents of the three-word memory field in symbolic locations AMTA-2 through AMTA into the triple accumulator. The memory field contains +9876 5432 1234.



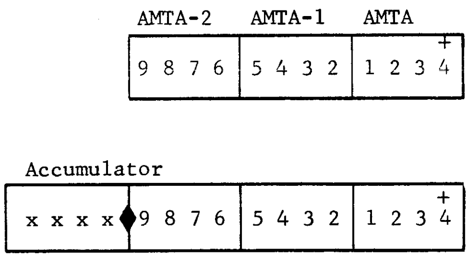
6	TYPE 7	8	REFERENCE SYMBOL: DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 22	USE 24	OPERATION: PICTURE 25–40	41
				LDT					AMTA	

Locations Affected:

Before



After



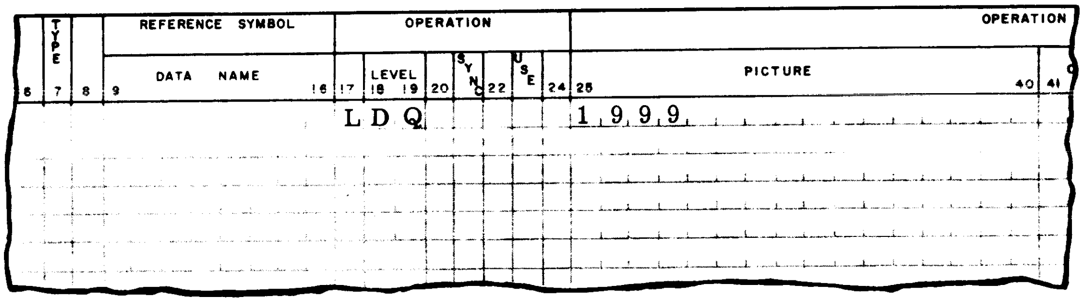
LOAD QUADRUPLE

LDQ Y/Y, X 43XYYYYY

The accumulator length is set to quadruple and the contents of locations Y-3, Y-2, Y-1 and Y replace the contents of the quadruple accumulator. The contents of the memory field are not changed.

Example:

The working accumulator is triple. Set the accumulator to quadruple and load the contents of the four-word memory field in decimal locations 1996 through 1999 into the quadruple accumulator. The memory field contains 2468 135A △△76 △149.



TYPE		REFERENCE SYMBOL / DATA NAME	OPERATION / LEVEL	SYNC	USE	OPERATION / PICTURE
			LDQ			1999

Locations Affected:

Before

1996	1997	1998	1999
2 4 6 8	1 3 5 A	△ △ 7 6	△ 1 4 9

Accumulator

x x x x	◆x x x x	x x x x	x x x x

After

1996	1997	1998	1999
2 4 6 8	1 3 5 A	△ △ 7 6	△ 1 4 9

Accumulator

◆2 4 6 8	1 3 5 A	△ △ 7 6	△ 1 4 9

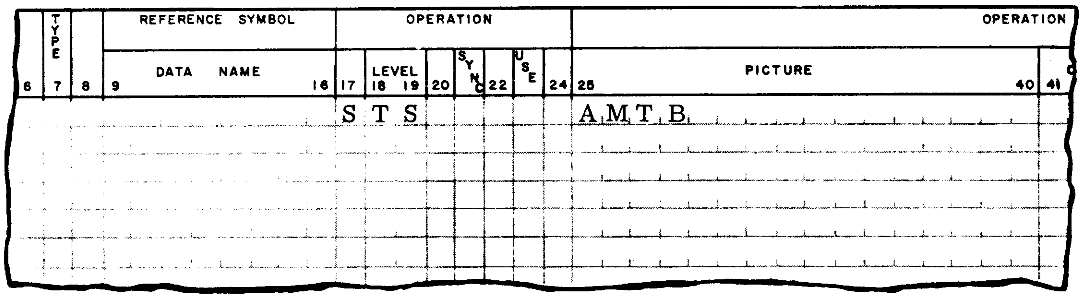
STORE SINGLE

STS Y/Y, X 44XYYYYY

The contents of location Y are replaced by the contents of the single accumulator, regardless of the accumulator length setting. The accumulator length and contents are not changed.

Example:

The working accumulator is double. Store the contents of the least-significant accumulator word in symbolic location AMTB. The working accumulator contains -1234 5678.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION / LEVEL 17–19	20	SYNC	22	USE	24	OPERATION / PICTURE 25–40	41
				STS						AMTB	

Locations Affected:

Before

Accumulator

x x x x	x x x x ◆	1 2 3 4	5 6 7 $\bar{8}$

AMTB

x x x x

After

Accumulator

x x x x	x x x x ◆	1 2 3 4	5 6 7 $\bar{8}$

AMTB

5 6 7 $\bar{8}$

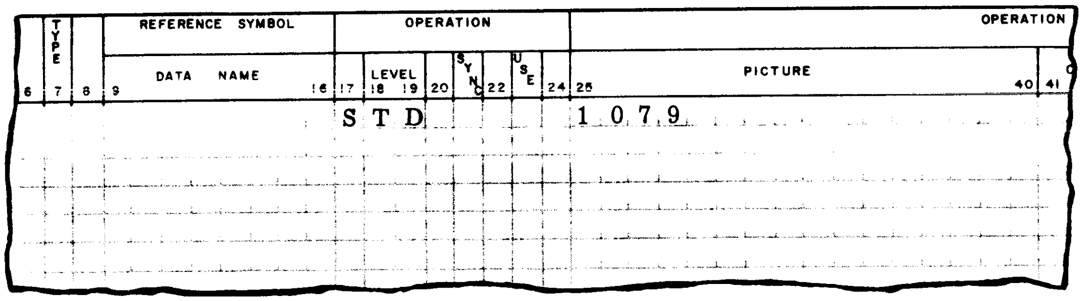
STORE DOUBLE

STD	Y/Y, X	45XYYYYY

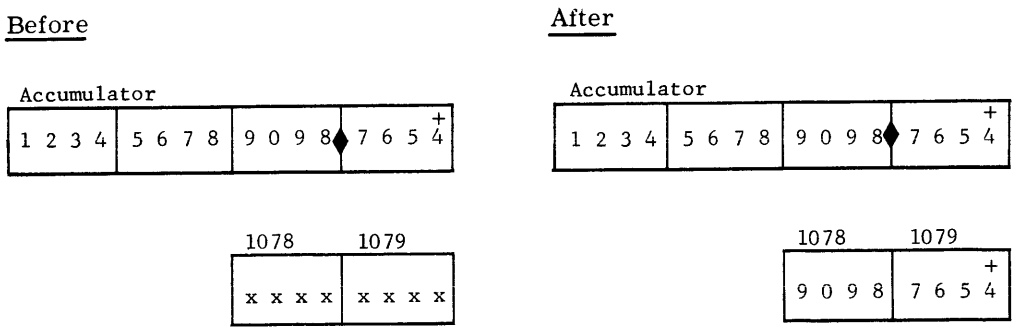
The contents of locations Y-1 and Y are replaced by the contents of the double accumulator, regardless of the accumulator length setting. The accumulator length and contents are not changed.

Example:

The working accumulator is single. Store the contents of the two least-significant accumulator words in decimal locations 1078 and 1079. The full accumulator contains +1234 5678 9098 7654.



Locations Affected:



STORE TRIPLE

STT	Y/Y, X	46XYYYYY

The contents of locations Y-2, Y-1, and Y are replaced by the contents of the triple accumulator, regardless of the accumulator length setting. The accumulator length and contents are not changed.

Example:

The working accumulator is quadruple. Store the contents of the three least-significant accumulator words in symbolic locations AMTB-2 through AMTB. The full accumulator contains NAME △RIC HARD △ROE.

	TYPE		REFERENCE SYMBOL	OPERATION					OPERATION	
			DATA NAME		LEVEL		SYNC	USE	PICTURE	
6	7	8	9 — 16	17	18 19	20	22	24	25 — 40	41
				S	T T				AMTB	

Locations Affected:

Before

Accumulator

◆N A M E	△ R I C	H A R D	△ R O E

AMTB-2	AMTB-1	AMTB
x x x x	x x x x	x x x x

After

Accumulator

◆N A M E	△ R I C	H A R D	△ R O E

AMTB-2	AMTB-1	AMTB
△ R I C	H A R D	△ R O E

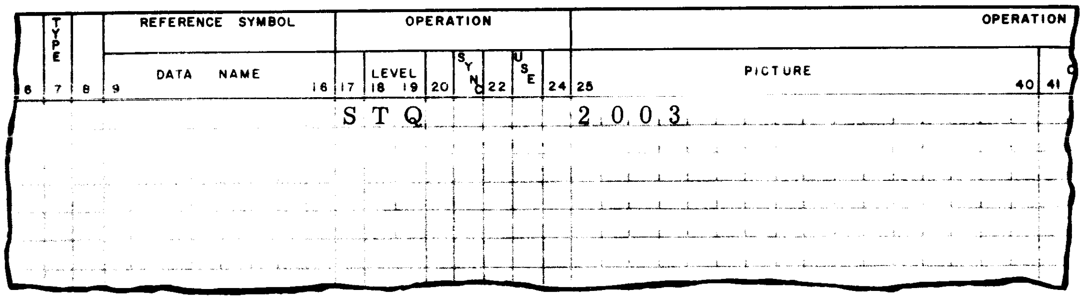
STORE QUADRUPLE

STQ　　　　Y/Y, X　　　　47XYYYYY

The contents of locations Y-3, Y-2, Y-1, and Y are replaced by the contents of the quadruple accumulator, regardless of the accumulator length setting. The accumulator length and contents are not changed.

Example:

The working accumulator is triple. Store the contents of the four accumulator words in locations 2000 through 2003. The accumulator contains ΔABR AHAM ΔLIN COLN.



TYPE	REFERENCE SYMBOL / DATA NAME	OPERATION	PICTURE
		STQ	2003

Locations Affected:

Before

Accumulator

Δ A B R	A H A M	Δ L I N	C O L N

2000	2001	2002	2003
x x x x	x x x x	x x x x	x x x x

After

Accumulator

Δ A B R	A H A M	Δ L I N	C O L N

2000	2001	2002	2003
Δ A B R	A H A M	Δ L I N	C O L N

MOVE FROM FIRST MEMORY

MFM	Y/Y, X	30XYYYYY
O/OP/OL	Z	0C0ZZZZZ

The contents of location Y are moved to a second location, leaving the contents of Y unchanged. Y can be modified under control of the instruction word ACF (X), which is either zero (no modification) or seven (modification). The second location is defined by a Second Address Sequence. The SAS word can be an Operand, Operand Pointer, or Operand Link:

OPERAND. The contents of the SAS word are replaced by the contents of Y.

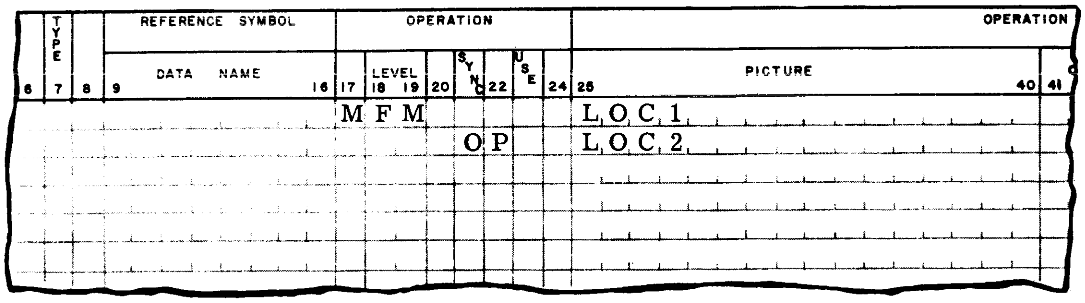
OPERAND POINTER. The contents of location Z are replaced by the contents of Y.

OPERAND LINK. Location Z contains another SAS word which is either an Operand, Operand Pointer, or another Operand Link.

With this instruction, the SAS word should not ordinarily be coded as an Operand. This instruction would cause an Operand to be changed, thus possibly changing the class of the SAS word to other than Operand.

Example:

Move the contents of memory location, LOC1, to a second memory location LOC2. The memory word LOC1 contains +1234.



6	TYPE 7	8	REFERENCE SYMBOL DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 21 22	USE 23 24	OPERATION PICTURE 25–40	41
				M	F M				LOC1	
							O P		LOC2	

Locations Affected:

Before	After
LOC1	LOC1
$1\ 2\ 3\ \overset{+}{4}$	$1\ 2\ 3\ \overset{+}{4}$
LOC2	LOC2
x x x x	$1\ 2\ 3\ \overset{+}{4}$

MOVE FROM IMMEDIATE

MFI	Y/Y, X	31XYYYYY
O/OP/OL	Z	0C0ZZZZZ

The address field Y is moved to the address field of a second location. The operation code field and ACF of the second location are not changed. Y can be modified under control of the instruction word ACF (X), which is either zero (no modification) or seven (modification). The second location is defined by a Second Address Sequence. The SAS word can be an Operand, Operand Pointer, or an Operand Link:

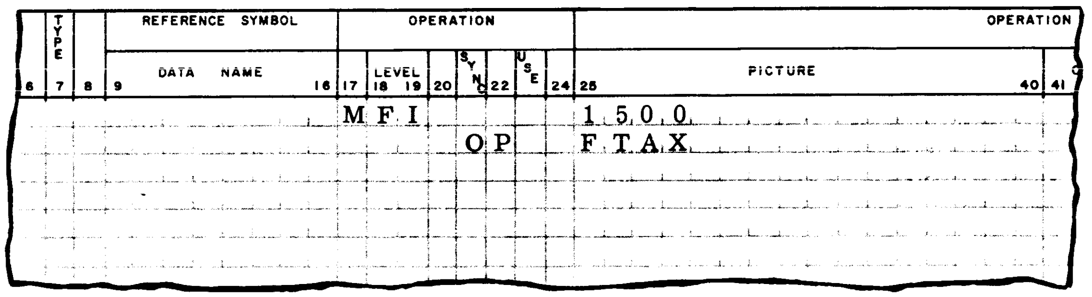
OPERAND. The address field of the SAS word is replaced by the address field Y.

OPERAND POINTER. The address field of location Z is replaced by the address field Y.

OPERAND LINK. Location Z contains another SAS word which is either an Operand, Operand Pointer, or another Operand Link.

Example:

Move the value $(1500)_{10}$ to the address field of symbolic location FTAX. The value $(1500)_{10}$ is in the address field of the MFI instruction. The MFI instruction is in location 1750.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 21–22	USE 23	24	OPERATION / PICTURE 25–40	41
				MFI						1500	
							OP			FTAX	

Locations Affected:

Before

1750

M F I	0	$(1500)_{10}$

FTAX

x x x	x x x x x

After

1750

M F I	0	$(1500)_{10}$

FTAX

x x x	$(1500)_{10}$

MOVE TO FIRST ADDRESS FIELD

MTA	Y/Y, X	32XYYYYY
O/OP/OL	Z	0C0ZZZZZ

The address field of location Y is replaced by the address field of a second location. The second location is not changed. Y can be modified under control of the instruction word ACF (X), which is either zero (no modification) or seven (modification). The second location is defined by a Second Address Sequence. The SAS word can be an Operand, Operand Pointer, or Operand Link:

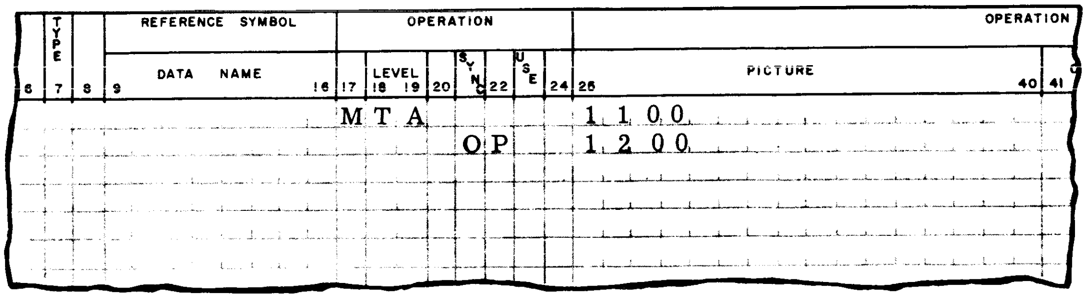
OPERAND. The address field of the SAS word replaces the address field of location Y.

OPERAND POINTER. The address field of location Z replaces the address field of location Y.

OPERAND LINK. Location Z contains another SAS word which is either an Operand, Operand Pointer, or another Operand Link.

Example:

Replace the address field of location 1100 with the contents of the address field of location 1200. The address field of location 1200 contains the address $(1300)_{10}$.



TYPE	REFERENCE SYMBOL / DATA NAME	OPERATION / LEVEL	SYNC	USE	OPERATION / PICTURE
6 7 8	9 … 16	17 18 19 20	21 22	23 24	25 … 40 41
		MTA			1100
			OP		1200

Locations Affected:

Before

1200

x x x	$(1300)_{10}$

1100

x x x	x x x x x

After

1200

x x x	$(1300)_{10}$

1100

x x x	$(1300)_{10}$

MOVE

MOV	Y/Y, X	06XYYYYY
OP/OL	Z	0C0ZZZZZ

The contents of a memory field whose most-significant word is at location Y are replaced by words moved from a second memory field. The second memory field is not changed.

The receiving field address, Y, can be modified under control of the ACF (X), which is either zero (no modification) or seven (modification).

The location of the most-significant word of the second memory field is defined in a control word which also contains a word count for N, the number of words to be moved (N can range from 1 to 512). This count is in the form 512-N, expressed in binary.

The control word format is:

23 ← 15	14 ← 0
Count, 512-N	"From Address" Field

N = Number of words to be moved.

The control word can be generated by using the MCTR pseudo-operation, which is described below.

The location of the control word is defined by a Second Address Sequence. The SAS word can be an Operand Pointer or Operand Link (because of the control word format, it should not be an SAS Operand):

OPERAND POINTER. Location Z contains the control word.

OPERAND LINK. Location Z contains another SAS word which is either an Operand Pointer or another Operand Link.

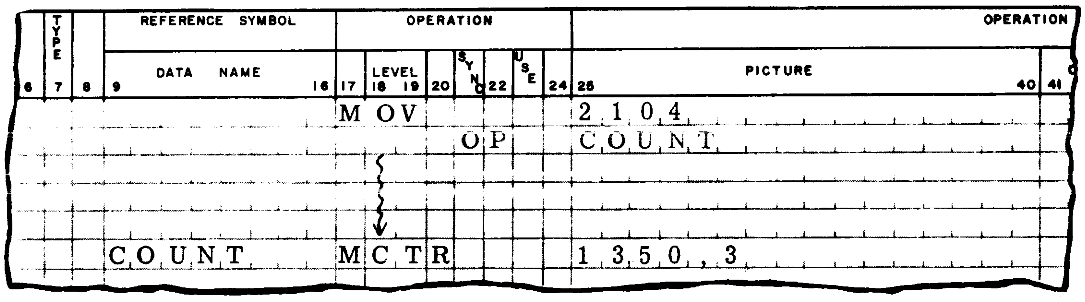
Move Counter (MCTR) Pseudo-Operation. The MCTR pseudo-operation is used to specify the control word required for the proper execution of the Move (MOV) and Move on Index Control (MXC) instructions.

To use the pseudo-operation, place MCTR in the Operation field of the coding sheet. In the Operation Parameters field, place an absolute or symbolic expression for the starting address of the memory field to be moved and an absolute or symbolic expression for the number of words to be moved. The largest count that can be specified is 512. The MCTR should also be identified by a symbol in the Reference Symbol field of the coding sheet.

The MCTR should not be placed in the P-sequence, but in an area of memory used for constants.

Example:

Move the contents of the three-word memory field that starts at location 1350 to the three-word memory field that starts at location 2104. The control word is at symbolic location COUNT.



TYPE		REFERENCE SYMBOL / DATA NAME	OPERATION / LEVEL	SYN	USE	OPERATION / PICTURE
			MOV			2104
				OP		COUNT
		COUNT	MCTR			1350,3

Locations Affected:

Before

COUNT	
(509)10	(1350)10

1350	1351	1352
JOHN	∆JOH	NSON

2104	2105	2106
xxxx	xxxx	xxxx

After

COUNT	
(509)10	(1350)10

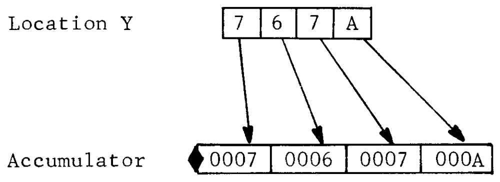
1350	1351	1352
JOHN	∆JOH	NSON

2104	2105	2106
JOHN	∆JOH	NSON

EXPLODE

EXP	Y/Y, X	20XYYYYY

The working accumulator is cleared to zero. From one to four characters from location Y are distributed to the working accumulator, one character to each accumulator word, as follows:



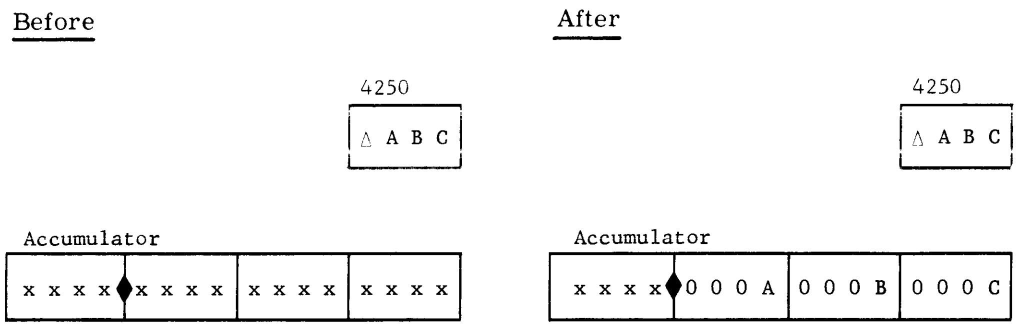
Accumulator words outside the working accumulator and the contents of location Y are not changed.

Example:

Explode the three least-significant characters of location 4250 into the triple working accumulator. (The accumulator length was previously set.) Location 4250 contains △ABC.

6	TYPE 7	8	REFERENCE SYMBOL DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 21	22	USE 23	24	OPERATION PICTURE 25–40	41
				E	X P						4 2 5 0	

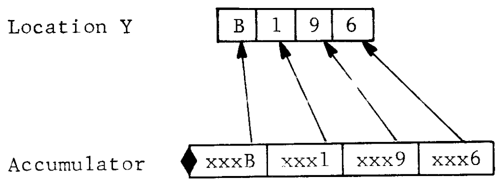
Locations Affected:


Before
After
4250
△ A B C
4250
△ A B C
Accumulator
x x x x x x x x x x x x x x x x
Accumulator
x x x x 0 0 0 A 0 0 0 B 0 0 0 C

IMPLODE

IMP	Y/Y, X	21XYYYYY

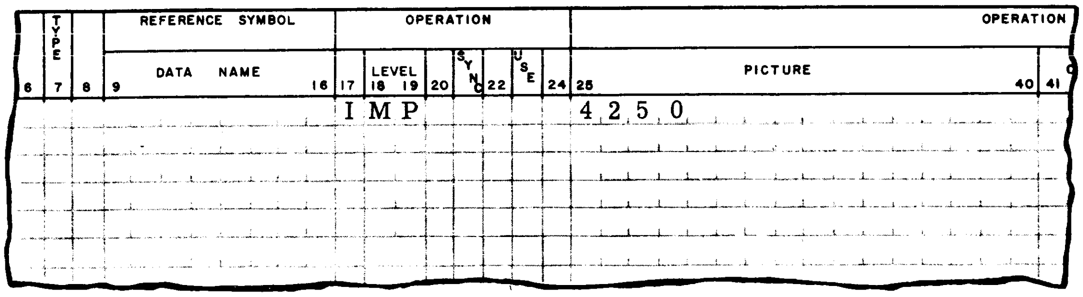
The least-significant character of each working accumulator word is gathered into location Y, as follows:



The character from the least-significant accumulator word is always placed in the least-significant character position of location Y. If any accumulator words are excluded because of a working accumulator length less than quadruple, corresponding character positions in location Y are cleared to zero. The accumulator contents and length are not changed.

Example:

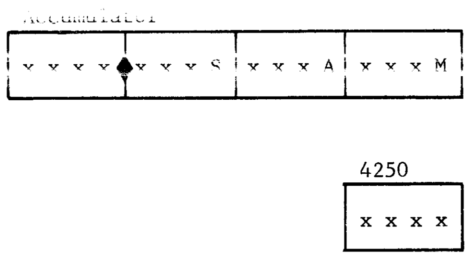
Implode the least-significant characters of each word of the triple accumulator into location 4250. (The accumulator length was previously set.) The full accumulator contains xxxx xxxS xxxA xxxM.



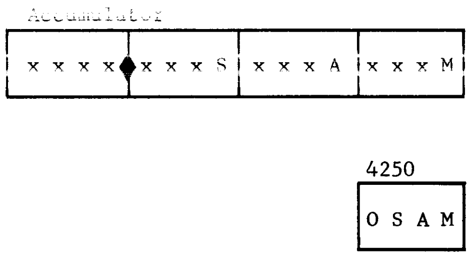
	TYPE		REFERENCE SYMBOL / DATA NAME	OPERATION	LEVEL		SYNC		USE		OPERATION / PICTURE	
6	7	8	9 — 16	17	18 19	20		22		24	25 — 40	41
				I	M P						4 2 5 0	

Locations Affected:

Before



After



SHIFT INSTRUCTIONS

The Shift instructions provide a variety of capabilities involving the accumulator. The type of shift, the length of the accumulator affected, the direction of movement, and the number of positions to be shifted are determined by the address field, which (in this case) does not refer to a memory location. Standard address modification procedures can also be used to modify the operation of the Shift instructions.

The address field functions as follows:

Bits 0 through 4 constitute the count field, expressing (in binary) the number of character or bit positions to be shifted. Any number from zero through 31 is valid.

Bits 5 and 6 have no effect, but must be zero for programming compatibility with other members of the GE-400 family.

Bits 7 through 14 control the specific type of shift and other functions to be performed.

Major characteristics of the Shift instructions are listed below:

1. Types

   a. Shift - Characters or bits shifted out of the accumulator are lost. Zeros enter the other end of the accumulator.

   b. Rotate - A circular shift in which characters or bits leaving one end of the accumulator re-enter the other end and no information is lost.

2. Direction of movement - Data movement within the accumulator can be either left (characters only) or right (characters or bits).

3. Accumulator length - The length of the accumulator affected can be single, double, triple, or quadruple and is independent of the working accumulator length setting. Binary shifts involve the single or double accumulator only. In character shifts, the working accumulator length can be set to correspond to the accumulator length to be shifted.

4. Data types - Shifting of data can be either by character (decimal or alpha) or by bit (binary).

   a. Alpha - All character positions are treated identically by the Shift instructions. The count field controls the number of character positions that the accumulator contents are shifted or rotated.

   b. Decimal - The accumulator sign is remembered outside the accumulator and the sign bits in the accumulator are set to zero before shifting occurs. After shifting, the remembered sign is restored (00 for plus and 10 for minus) to the accumulator sign bit positions. The count field controls the number of character positions that the accumulator contents are moved.

   c. Binary - All positions of the accumulator are treated identically in shifting. The count field controls the number of bit positions that the accumulator contents are moved.

5. Special functions

   a. Test (binary only) - The least-significant bit position of the accumulator can be tested after shift completion. If this bit is zero, the Compare indicators are set to the less than condition; if the bit is a one, the Compare indicators are set to the equal condition. The Compare indicators can then be tested in the program by using either the BRE or BRL instruction.

   b. Set Length (alpha or decimal only) - The working accumulator length can be set to correspond to the accumulator length to be shifted.

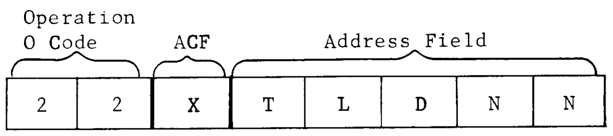
Shift Length vs. Accumulator Length

If the number of positions specified exceeds the length of the accumulator involved, the result is a function of the type of shift:

1. Rotate - The specified number of positions are rotated.
2. Shift - The portion of the accumulator involved will contain all zeros except in a decimal shift which preserves the sign when it is minus.

Instruction Word Structure

The octal format of the Shift instruction word is shown in Figure III-4. From the information given, it is possible to construct, in octal, any legal combination of shift functions. The programmer can construct his own Shift instructions in octal or use assembly language mnemonics provided for that purpose.



22 = Shift instruction operation code

X = ACF; indexing and address field modification is possible, but the nature of the shift is altered.

T = Type of shift
- 0 = Shift alpha (character)
- 1 = Rotate alpha (character)
- 2 = Shift binary (bits)
- 3 = Rotate binary (bits)
- 4 = Shift decimal (characters)
- 5 = Rotate decimal (characters)

L = Length of accumulator affected

0 = Quadruple, do not set length	Alpha or decimal only
1 = Triple, do not set length	
2 = Double, do not set length	Alpha, decimal, or binary
3 = Single, do not set length	
4 = Quadruple, set length quad	Alpha or decimal only
5 = Triple, set length triple	
6 = Double, set length double	
7 = Single, set length single	

D = Direction
- 0 = Right
- 2 = Left (alpha or decimal only)
- 4 = Right and test least-significant bit (binary only)

NN = Number of positions (bits or characters) to be shifted; 0 through 37 octal (which is the equivalent of 0 through 31 decimal).

Figure III-4. Octal Format of the Shift Instruction

Assembly Language Shift Mnemonic Codes

The basic assembly language makes available to the programmer 72 mnemonic codes for Shift instructions. From the two tables in Figure III-5 the programmer can construct a mnemonic code for each combination of shift functions that has been described. A table is provided for each of the two categories of shifts: character and bit. Mnemonic operation codes for character shifts contain either four or five letters, depending upon whether the Set Length function is included. Bit shift mnemonics contain either three or four letters, depending upon whether the Test Bit function is included.

In addition to the two tables, all 72 mnemonic operation codes and their octal equivalents are listed in both the alphabetic and the octal listing of the instruction repertoire in the Appendix.

Character Shifts (Decimal and Alpha)

S Shift	R Right	S Single	A Alpha	S Set Accum Length
		D Double		
R Rotate	L Left	T Triple	D Decimal	Δ (blank)
		Q Quadruple		

Bit Shifts (Binary)

S Shift	R Right	S Single	T Test
R Rotate		D Double	Δ (blank) No Test

Figure III-5. Tables for Constructing Mnemonic Codes for Shift Instructions

Using these tables, the programmer can construct any of the Shift mnemonic operation codes desired. For example, if a rotate right of two accumulator words in the alpha mode is required, this could be coded as follows:

Operation Code	Operation Parameters
RRDA	N

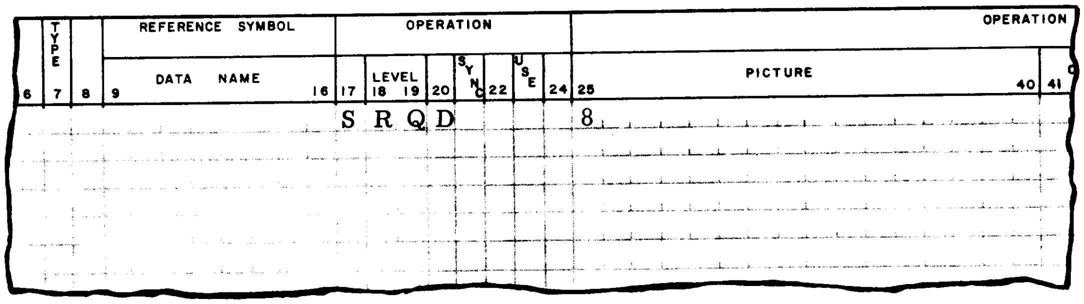
This would provide a Rotate Right Double Alpha instruction that would rotate the characters of the two least-significant words of the accumulator N places. The working accumulator length setting would not be affected.

Other typical shift instructions are listed below:

Operation Code	Operation Parameters	Instructions
SRSD	N	Shift Right Single Decimal, N characters
SLTDS	N	Shift Left Triple Decimal and Set, N characters
RLSAS	N	Rotate Left Single Alpha and Set, N characters
RRD	N	Rotate Right Double (Binary), N bits
SRST	N	Shift Right Single (Binary) Test, N bits

Example:

Shift all of the decimal information in the four-word accumulator right eight character positions. The working accumulator is presently triple, and the full accumulator contains +1692 1941 1950 1963.



	TYPE		REFERENCE SYMBOL / DATA NAME	OPERATION / LEVEL		SYN		USE		OPERATION / PICTURE	
6	7	8	9 – 16	17 – 19	20		22		24	25 – 40	41
				S R Q	D					8	

Locations Affected:

Before

Accumulator

1 6 9 2 ◆	1 9 4 1	1 9 5 0	1 9 6 $\overset{+}{3}$

After

Accumulator

0 0 0 0 ◆	0 0 0 0	1 6 9 2	1 9 4 $\overset{+}{1}$

Example:

Shift the alphanumeric information in the triple accumulator left four character positions and set the length of the working accumulator to triple. The working accumulator is presently double. The four-word accumulator contains 1234 5678 MOST △PAY.

6	TYPE 7	8	REFERENCE SYMBOL DATA NAME 9–16	OPERATION 17–24	OPERATION PICTURE 25–40	41
				SLTAS	4	

Locations Affected:

Before

Accumulator

1 2 3 4	5 6 7 8◆	M O S T	△ P A Y

After

Accumulator

1 2 3 4◆	M O S T	△ P A Y	0 0 0 0

Example:

Rotate the decimal information in the two least-significant words of the accumulator four character positions right. The working accumulator is presently triple. The 4-word accumulator contains -1234 5678 1421 5980.

6	TYPE 7	8	REFERENCE SYMBOL DATA NAME 9–16	OPERATION 17–24	OPERATION PICTURE 25–40	41
				RRDD	4	

Locations Affected:

Before

Accumulator

1 2 3 4◆	5 6 7 8	1 4 2 1	5 9 8 $\bar{0}$

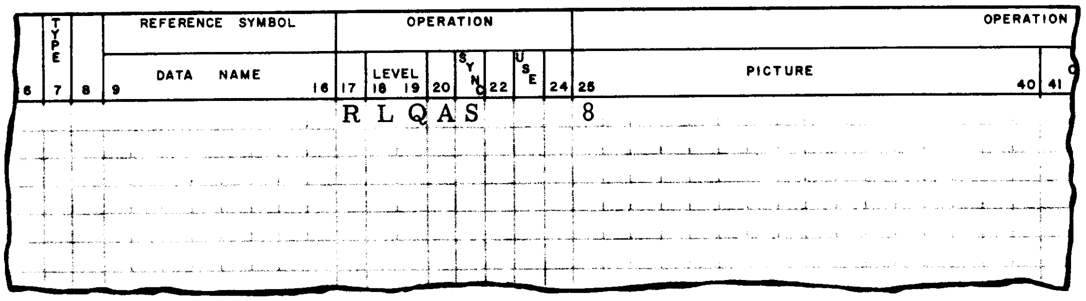
After

Accumulator

1 2 3 4◆	5 6 7 8	5 9 8 0	1 4 2 $\bar{1}$

Example:

Rotate all of the alphanumeric information in the four-word accumulator eight character positions to the left and set the accumulator to quadruple. The working accumulator is presently single. The four-word accumulator contains ACCO UNTΔ DUEΔ PAYΔ.



6	7 TYPE	8	9 REFERENCE SYMBOL / DATA NAME 16	17 OPERATION / LEVEL 18 19 20 SYN 22 USE 24	25 OPERATION / PICTURE 40	41
				RLQAS	8	

Locations Affected:

Before

Accumulator

A C C O	U N T Δ	D U E Δ	◆P A Y Δ

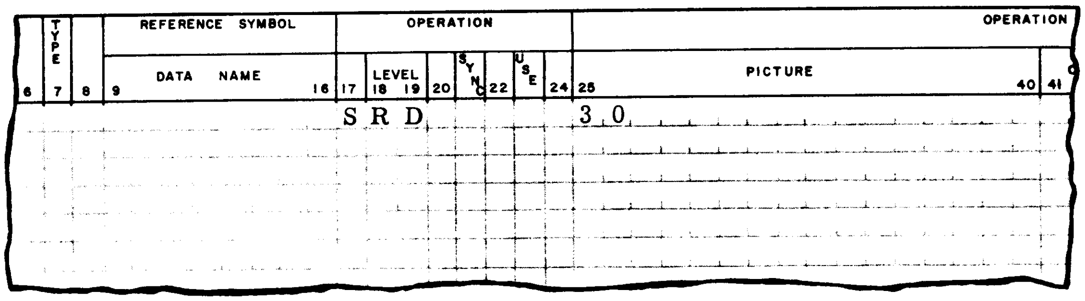
After

Accumulator

◆D U E Δ	P A Y Δ	A C C O	U N T Δ

Example:

Shift the binary information in the double accumulator 30 bit positions (10 octal digits) to the right. The working accumulator is presently quadruple. The four-word accumulator contains the octal number xxxxxxxx xxxxxxxx 32641234 01020304.



6	7 TYPE	8	9 REFERENCE SYMBOL / DATA NAME 16	17 OPERATION / LEVEL 18 19 20 SYN 22 USE 24	25 OPERATION / PICTURE 40	41
				SRD	30	

Locations Affected:

Before

Accumulator (octal)

◆xxxxxxxx	xxxxxxxx	32641234	01020304

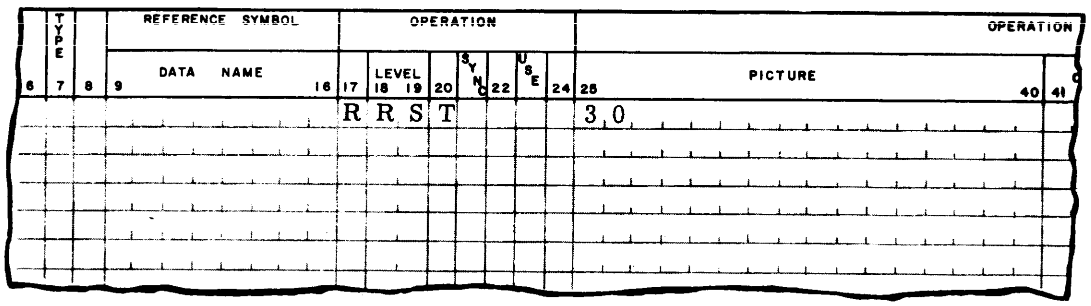
After

Accumulator (octal)

◆xxxxxxxx	xxxxxxxx	00000000	00326412

Example:

Rotate the binary information in the single accumulator 30 bit positions to the right and set the Compare indicators. The working accumulator is presently quadruple. Notice that the same results are obtained by rotating only six bit positions. The four-word accumulator contains the octal number xxxxxxxx xxxxxxxx xxxxxxxx 12345670.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 22	USE 24	OPERATION / PICTURE 25–40	41
				R	R S	T			3 0	

Locations Affected:

Before	After

Accumulator (octal)

xxxxxxxx	xxxxxxxx	xxxxxxxx	12345670

Accumulator (octal)

xxxxxxxx	xxxxxxxx	xxxxxxxx	70123456

After the shift, the least-significant bit position of the accumulator contains a zero (octal 6 is binary 110). This causes the Compare indicators to be set to "less".

LOW BIT TEST

LBT 22X23400

The least-significant bit position of the accumulator is tested. If it is zero, the Compare indicators are set to "less"; if it is a one, the Compare indicators are set to "equal".

This instruction is a special case of the Shift instruction and is the equivalent of a Shift Right Single Test with zero number of bits specified to be shifted (SRST 0).

ARITHMETIC INSTRUCTIONS

ADD DECIMAL SINGLE

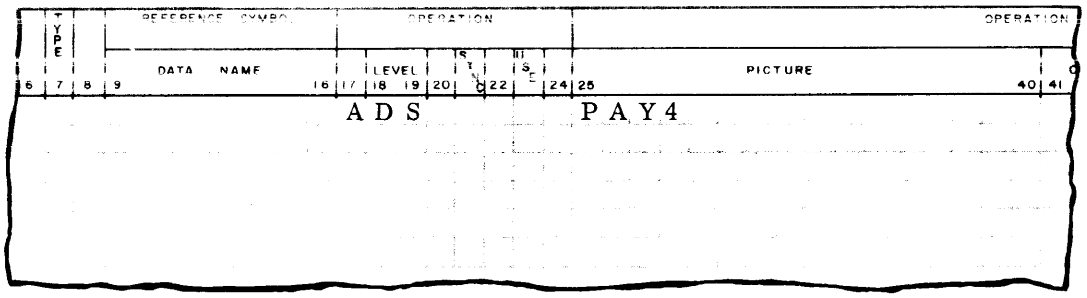
ADS	Y/Y, X	50XYYYYY

The contents of location Y are added to the contents of the working accumulator. The result is placed in the accumulator, and Y is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working accumulator are set to zero. The accumulator length is not affected by this instruction.

Overflow: A carry out of the working accumulator turns on the Overflow indicator. The carry is lost and the initial sign of the accumulator is not changed.

Example:

Add the decimal contents of location PAY4 to the accumulator. The contents of PAY4 are +1040 and the contents of the working accumulator (which is single) are -20.



TYPE
REFERENCE SYMBOL
DATA NAME
OPERATION
LEVEL
PICTURE
OPERATION

OPERATION	OPERAND
ADS	PAY4

Locations Affected:

Before

PAY4

1 0 4 $\overset{+}{0}$

Accumulator

x x x x	x x x x	x x x x ♦	0 0 2 $\overset{-}{0}$

After

PAY4

1 0 4 $\overset{+}{0}$

Accumulator

x x x x	x x x x	x x x x ♦	1 0 2 $\overset{+}{0}$

SUBTRACT DECIMAL SINGLE

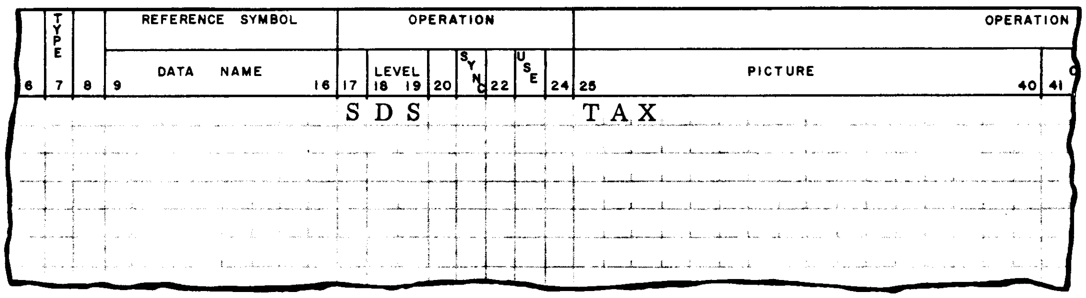
SDS Y/Y, X 60XYYYYY

The contents of location Y are subtracted from the contents of the working accumulator. The result is placed in the accumulator, and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working accumulator are set to zero. The accumulator length is not affected by this instruction.

Overflow: If the accumulator and the contents of Y have unlike signs, the result can exceed the capacity of the working accumulator, turning on the Overflow indicator. The carry is lost and the initial sign of the accumulator is not changed.

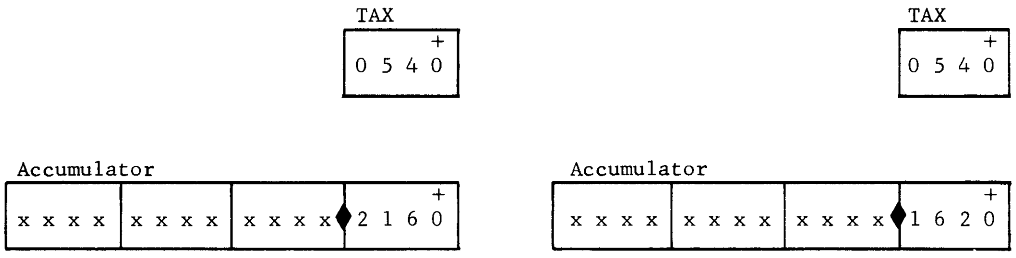
Example:

Subtract the decimal contents of location TAX from the accumulator. The contents of TAX are +540 and the contents of the working accumulator (which is single) are +2160.



Locations Affected:

Before After



ADD DECIMAL DOUBLE

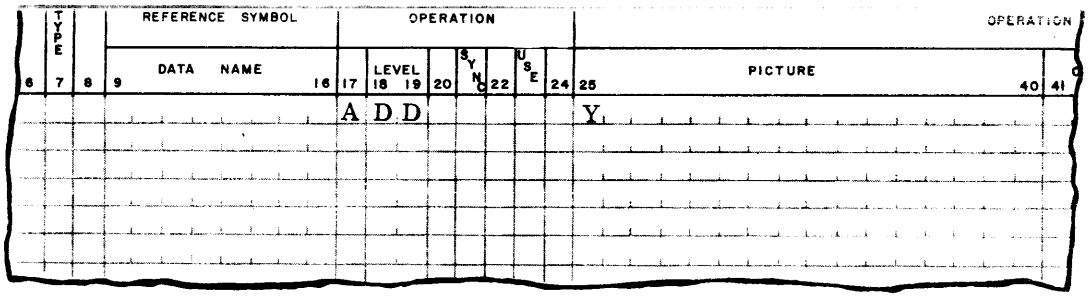
ADD Y/Y, X 51XYYYYY

The contents of locations Y-1 and Y are added to the contents of the working accumulator. The result is placed in the accumulator and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working accumulator are set to zero. If the working accumulator is double, triple, or quadruple, the length is not changed. If the working accumulator is single, the length is set to double and the added working accumulator word is cleared to zeros before addition occurs.

Overflow: A carry out of the working accumulator turns on the Overflow indicator. The carry is lost and the initial sign of the accumulator is not changed.

Example:

Add the decimal contents of Y-1 and Y to the contents of the accumulator. The contents of Y-1 and Y are +2000 3000 and the contents of the working accumulator (which is single) are +1000. (The added working accumulator word is cleared to zero before the addition takes place.)



TYPE	REFERENCE SYMBOL / DATA NAME	OPERATION	OPERATION / PICTURE
		ADD	Y

Locations Affected:

Before

Y-1	Y
2000	3000 (+)

Accumulator

xxxx	xxxx	5500♦	1000 (+)

After

Y-1	Y
2000	3000 (+)

Accumulator

xxxx	xxxx♦	2000	4000 (+)

SUBTRACT DECIMAL DOUBLE

SDD	Y/Y, X	61XYYYYY

The contents of locations Y-1 and Y are subtracted from the contents of the working accumulator. The result is placed in the accumulator, and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working accumulator are set to zero. If the working accumulator is double, triple, or quadruple, the length is not changed. If the working accumulator is single, the length is set to double and the added working accumulator word is cleared to zeros before subtraction occurs.

Overflow: If the accumulator and the contents of Y have unlike signs, the result can exceed the capacity of the working accumulator, turning on the Overflow indicator. The carry is lost and the initial sign of the accumulator is not changed.

Example:

Subtract the decimal contents of locations 2004 and 2005 from the accumulator. The contents of 2004 and 2005 are +8060 5000 and the contents of the working accumulator are +1234 1234 0061 5500. The current working accumulator is quadruple.

6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION / LEVEL 17–20	SYN 22	USE 24	OPERATION / PICTURE 25–40	41
				SDD			2005	

Locations Affected:

Before

2004	2005
8 0 6 0	5 0 0 0 +

Accumulator

♦1 2 3 4	1 2 3 4	0 0 6 1	5 5 0 0 +

After

2004	2005
8 0 6 0	5 0 0 0 +

Accumulator

♦1 2 3 4	1 2 3 3	2 0 0 1	0 5 0 0 +

ADD DECIMAL TRIPLE

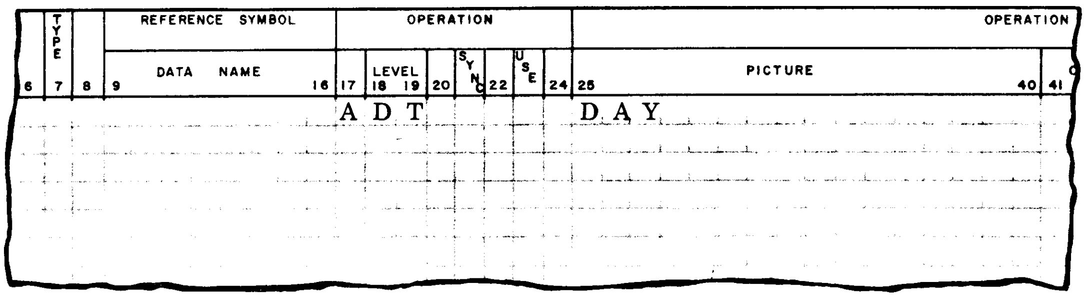
ADT	Y/Y, X	52XYYYYY

The contents of locations Y-2, Y-1, and Y are added to the contents of the working accumulator. The result is placed in the accumulator and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working accumulator are set to zero. If the working accumulator is triple or quadruple, the length is not changed. If the working accumulator is double or single, the length is set to triple and the one or two added working accumulator words are cleared to zeros before addition occurs.

Overflow: A carry out of the working accumulator turns on the Overflow indicator. The carry is lost and the initial sign of the accumulator is not changed.

Example:

Add the decimal contents of DAY-2 through DAY to the contents of the accumulator. The contents of DAY-2 through DAY are +0001 5000 4000 and the contents of the working accumulator (which is double) are +3020 3030. (The added working accumulator word is cleared to zero before the addition takes place.)



TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17 / LEVEL 18 19	20	SYN 21	22	USE 23	24	OPERAND / PICTURE 25–40	41
			ADT						DAY	

Locations Affected:

Before

DAY-2	DAY-1	DAY
0001	5000	400$\overset{+}{0}$

Accumulator

xxxx	1234◆	3020	303$\overset{+}{0}$

After

DAY-2	DAY-1	DAY
0001	5000	400$\overset{+}{0}$

Accumulator

xxxx◆	0001	8020	703$\overset{+}{0}$

SUBTRACT DECIMAL TRIPLE

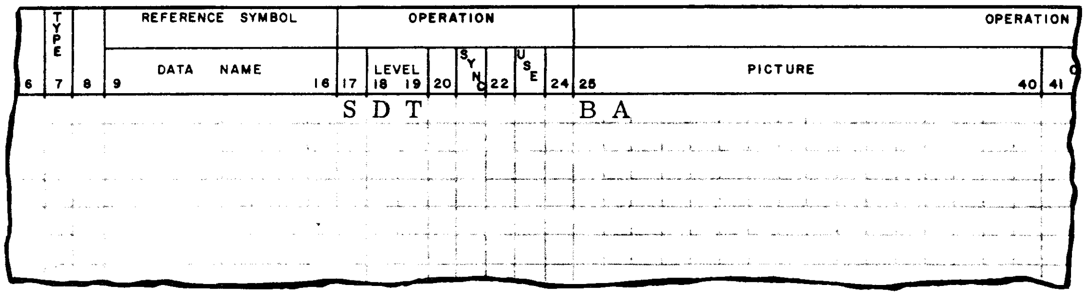
SDT	Y/Y, X	62XYYYYY

The contents of locations Y-2, Y-1, and Y are subtracted from the contents of the working accumulator. The result is placed in the accumulator, and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the working accumulator are set to zero. If the working accumulator is triple or quadruple, the length is not changed. If the working accumulator is double or single, the length is set to triple and the one or two added working accumulator words are cleared to zeros before subtraction occurs.

Overflow: If the accumulator and the contents of Y have unlike signs, the result can exceed the capacity of the working accumulator, turning on the Overflow indicator. The carry is lost and the initial sign of the accumulator is not changed.

Example:

Subtract the decimal contents of location BA-2 through BA from the accumulator. The contents of BA-2 through BA are -0000 8000 5000 and the contents of the working accumulator are +4000 1000. The current working accumulator is double.



TYPE (7)	8	REFERENCE SYMBOL / DATA NAME (9–16)	OPERATION / LEVEL (17–20)	SYN	USE	OPERATION / PICTURE (25–40)	41
			SDT			BA	

Locations Affected:

Before

BA-2	BA-1	BA
		−
0 0 0 0	8 0 0 0	5 0 0 0

Accumulator

			+
x x x x	6 5 8 5◆	4 0 0 0	1 0 0 0

After

BA-2	BA-1	BA
		−
0 0 0 0	8 0 0 0	5 0 0 0

Accumulator

			+
x x x x◆	0 0 0 1	2 0 0 0	6 0 0 0

ADD DECIMAL QUADRUPLE

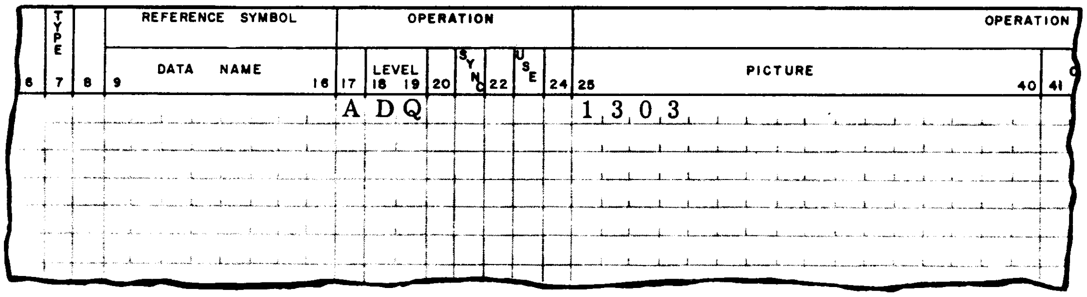
ADQ Y/Y, X 5 3 X Y Y Y Y Y

The contents of locations Y-3, Y-2, Y-1, and Y are added to the contents of the working accumulator. The result is placed in the accumulator and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the accumulator are set to zero. If the working accumulator is quadruple, the length is not changed. If the working accumulator is triple, double, or single, the length is set to quadruple and the one, two, or three added working accumulator words are cleared to zeros before addition occurs.

Overflow: A carry out of the accumulator turns on the Overflow indicator. The carry is lost and the initial sign of the accumulator is not changed.

Example:

Add the decimal contents of locations 1300 through 1303 to the contents of the accumulator. The contents of 1300 through 1303 are +0100 0200 0400 0300 and the contents of the accumulator are +1000 2000 4000 3000. The accumulator is currently set to quadruple.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 21 22	USE 23 24	OPERATION / PICTURE 25–40	41
				A	D Q				1303	

Locations Affected:

Before

1300	1301	1302	1303
0 1 0 0	0 2 0 0	0 4 0 0	0 3 0 $\overset{+}{0}$

Accumulator

◆ 1 0 0 0	2 0 0 0	4 0 0 0	3 0 0 $\overset{+}{0}$

After

1300	1301	1302	1303
0 1 0 0	0 2 0 0	0 4 0 0	0 3 0 $\overset{+}{0}$

Accumulator

◆ 1 1 0 0	2 2 0 0	4 4 0 0	3 3 0 $\overset{+}{0}$

SUBTRACT DECIMAL QUADRUPLE

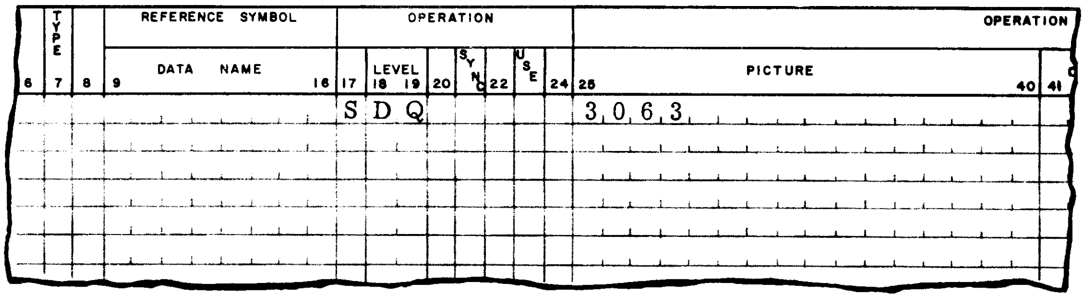
SDQ Y/Y, X 63XYYYYY

The contents of locations Y-3, Y-2, Y-1, and Y are subtracted from the contents of the working accumulator. The result is placed in the accumulator and the memory field is not changed. The sign of the result is placed in the zone bits of the least-significant result character; the remaining zone bits of the accumulator are set to zero. If the working accumulator is quadruple, the length is not changed. If the working accumulator is triple, double, or single, the length is set to quadruple and the one, two, or three added accumulator words are cleared to zeros before subtraction occurs.

Overflow: If the accumulator and the contents of Y have unlike signs, the result can exceed the capacity of the accumulator, turning on the Overflow indicator. The carry is lost and the initial sign of the accumulator is not changed.

Example:

Subtract the decimal contents of locations 3060 through 3063 from the accumulator. The contents of 3060 through 3063 are -6020 8000 5000 0010. The contents of the working accumulator are +1000 2000 0000. The current working accumulator is triple.



TYPE		REFERENCE SYMBOL / DATA NAME	OPERATION / LEVEL		SYNC	USE		PICTURE / OPERATION
			SDQ					3063

Locations Affected:

Before

3060	3061	3062	3063
6 0 2 0	8 0 0 0	5 0 0 0	- 0 0 1 0

Accumulator

0 3 2 0	◆ 1 0 0 0	2 0 0 0	+ 0 0 0 0

After

3060	3061	3062	3063
6 0 2 0	8 0 0 0	5 0 0 0	- 0 0 1 0

Accumulator

◆ 6 0 2 0	9 0 0 0	7 0 0 0	+ 0 0 1 0

ADD TO MEMORY SINGLE

AMS	Y/Y, X	54XYYYYY

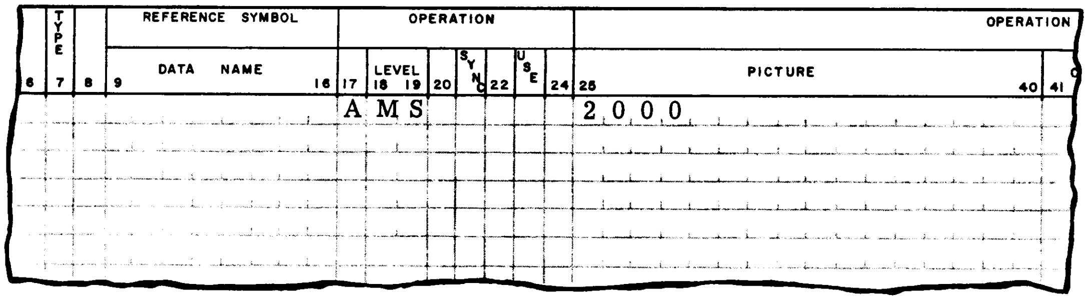
If the working accumulator is single, its contents are added to location Y and the result is stored in Y. The accumulator contents and length are not changed. The sign of the result is placed in the zone bits of the least-significant character of Y; zone bits of all other characters in Y are set to zeros.

Overflow: A carry out of the most-significant character position of Y is lost and the Overflow indicator is turned on.

If the accumulator length is double, triple, or quadruple, Y is not changed and the Overflow indicator is turned on.

Example:

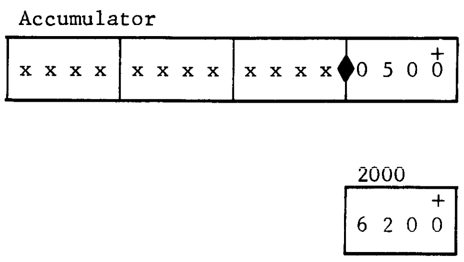
Add the contents of the working accumulator to the contents of location 2000. The working accumulator contains +0500 and location 2000 contains +6200.



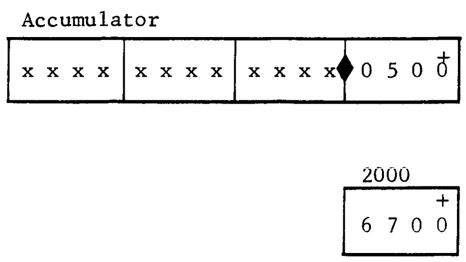
6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17–24	OPERATION / PICTURE 25–40	41
				AMS	2000	

Locations Affected:

Before



After



ADD TO MEMORY DOUBLE

AMD	Y/Y, X	55XYYYYY

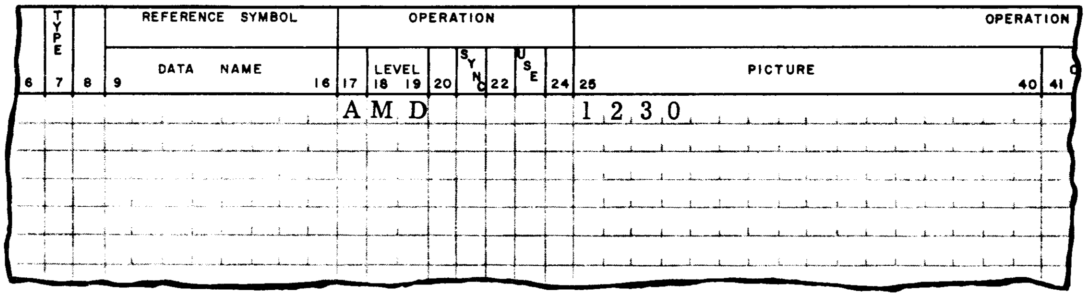
If the working accumulator is single or double, its contents are added to the contents of locations Y-1 and Y, and the result is stored in Y-1 and Y. The accumulator contents and length are not changed. The sign of the result is placed in the least-significant character position of Y; zone bits of all other characters in Y-1 and Y are set to zeros.

Overflow: A carry out of the most-significant character position of Y-1 is lost and the Overflow indicator is turned on.

If the accumulator length is triple or quadruple, Y-1 and Y are not changed and the Overflow indicator is turned on.

Example:

Add the contents of the working accumulator to the contents of location 1229 and 1230. The working accumulator contains -7050, and locations 1229 and 1230 contain -5500 5000.



TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 22	USE 24	OPERATION / PICTURE 25–40	41
			AMD					1230	

Locations Affected:

Before

Accumulator

x x x x	x x x x	x x x x ◆	7 0 5 $\overset{-}{0}$

1229	1230
5 5 0 0	5 0 0 $\overset{-}{0}$

After

Accumulator

x x x x	x x x x	x x x x ◆	7 0 5 $\overset{-}{0}$

1229	1230
5 5 0 1	2 0 5 $\overset{-}{0}$

ADD TO MEMORY TRIPLE

AMT	Y/Y, X	56XYYYYY

If the working accumulator is single, double, or triple, its contents are added to the contents of locations Y-2, Y-1, and Y, and the result is stored in those locations. The accumulator contents and length are not changed. The sign of the result is placed in the least-significant character position of Y; zone bits of all other characters of the result are set to zeros.

Overflow: A carry out of the most-significant character position of Y-2 is lost and the Overflow indicator is turned on.

If the accumulator length is quadruple, Y-2, Y-1, and Y are not changed and the Overflow indicator is turned on.

Example:

Add the contents of the working accumulator to the contents of location 1420 through 1422. The working accumulator contains +6700 3900 2100, and locations 1420 through 1422 contain +0050 0020 0090.

6	TYPE 7	8	REFERENCE SYMBOL: DATA NAME 9–16	OPERATION: 17	LEVEL 18 19	20	SYN	22	USE	24	OPERATION: PICTURE 25–40	41
				AMT							1422	

Locations Affected:

Before

Accumulator

			+
x x x x ♦	6 7 0 0	3 9 0 0	2 1 0 0

1420	1421	1422
		+
0 0 5 0	0 0 2 0	0 0 9 0

After

Accumulator

			+
x x x x ♦	6 7 0 0	3 9 0 0	2 1 0 0

1420	1421	1422
		+
6 7 5 0	3 9 2 0	2 1 9 0

ADD TO MEMORY QUADRUPLE

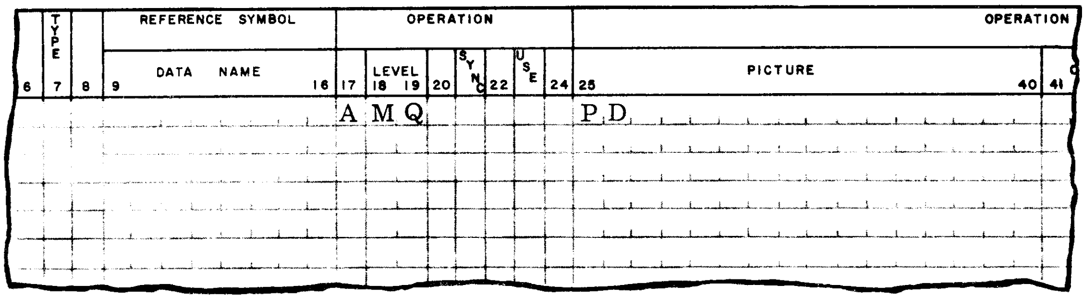
AMQ	Y/Y, X	57XYYYYY

The contents of the working accumulator are added to the four-word memory field in locations Y-3 through Y, and the result is stored in those locations. The accumulator contents and length are not changed. The sign of the result is placed in the least-significant character position of Y; zone bits of all other characters of the result are set to zeros.

Overflow: A carry out of the most-significant character position of Y-3 is lost and the Overflow indicator is turned on.

Example:

Add the contents of the working accumulator to the contents of locations PD-3 through PD. The current working accumulator is triple and contains +6200 7100 3200. Locations PD-3 through PD contain +0002 7000 2000 1000.



TYPE (7)	REFERENCE SYMBOL / DATA NAME (9–16)	OPERATION (17–24)	OPERATION / PICTURE (25–40)
		AMQ	PD

Locations Affected:

Before

Accumulator

x x x x	6 2 0 0	7 1 0 0	3 2 0 0 +

PD-3	PD-2	PD-1	PD
0 0 0 2	7 0 0 0	2 0 0 0	1 0 0 0 +

After

Accumulator

x x x x	6 2 0 0	7 1 0 0	3 2 0 0 +

PD-3	PD-2	PD-1	PD
0 0 0 3	3 2 0 0	9 1 0 0	4 2 0 0 +

ADD BINARY TO MEMORY

ABM	Y/Y, X	34XYYYYY
O/OP/OL	Z	0C0ZZZZZ

The contents of location Y are added absolutely, in binary, to the contents of a second location. The result is placed in the second location. The contents of Y are not changed. Y can be modified under control of the instruction word ACF (X), which is either zero (no modification) or seven (modification). The second location is defined by a Second Address Sequence. The SAS word can be an Operand, Operand Pointer, or Operand Link:

OPERAND. The contents of location Y are added to the contents of the SAS word and the result is stored in the SAS word.

OPERAND POINTER. The contents of location Y are added to the contents of location Z and the result is stored in location Z.

OPERAND LINK. Location Z contains another SAS word which is either an Operand, Operand Pointer, or another Operand Link.

With this instruction, the SAS word should not ordinarily be coded as an Operand. This instruction would cause an Operand to be changed, thus possibly changing the class of the SAS word to other than Operand.

Overflow: A carry out of the most-significant bit position of the second location is lost, and the Overflow indicator is turned on.

Example:

Add the contents of location 1500 absolutely in binary to the contents of location 2000. Location 1500 contains $(60566413)_8$ and location 2000 contains $(11111111)_8$.

6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9 – 16	OPERATION 17	LEVEL 18 19	20	SYNC	22	USE	24	OPERATION / PICTURE 25 – 40	41
				A	BM						1500	
							O	P			2000	

Locations Affected:

Before		After	
1500	110 000 101 110 110 100 001 011	1500	110 000 101 110 110 100 001 011
2000	001 001 001 001 001 001 001 001	2000	111 001 110 111 111 101 010 100

SUBTRACT BINARY FROM MEMORY

SBM	Y/Y, X	35XYYYYY
O/OP/OL	Z	0C0ZZZZZ

The contents of location Y are subtracted absolutely, in binary, from the contents of a second location. The result is placed in the second location. The contents of Y are not changed. Y can be modified under control of the instruction word ACF (X), which is either zero (no modification) or seven (modification). The second location is defined by a Second Address Sequence. The SAS word can be an Operand, Operand Pointer, or Operand Link:

OPERAND. The contents of location Y are subtracted from the contents of the SAS word and the result is stored in the SAS word.

OPERAND POINTER. The contents of location Y are subtracted from the contents of location Z and the result is stored in location Z.

OPERAND LINK. Location Z contains another SAS word which is either an Operand, Operand Pointer, or another Operand Link.

With this instruction, the SAS word should not ordinarily be coded as an Operand. This instruction would cause an Operand to be changed, thus possibly changing the class of the SAS word to other than Operand.

Overflow: In the absolute subtraction, if the value to be subtracted exceeds the value contained in the second location, the result (in 2's complement form) is placed in the second location and the Overflow indicator is turned on.

Example:

Subtract the contents of location 1500 absolutely in binary from the contents of location 2000. Location 1500 contains $(11111111)_8$ and location 2000 contains $(65616627)_8$.

6	7 TYPE	8	9 REFERENCE SYMBOL / DATA NAME 16	17 OPERATION	LEVEL 18 19	20	SYNC	22	USE	24	25 OPERATION / PICTURE 40	41
				S	B M						1500	
							O	P			2000	

Locations Affected:

Before

1500

001	001	001	001	001	001	001	001

2000

110	101	110	001	110	110	010	111

After

1500

001	001	001	001	001	001	001	001

2000

101	100	101	000	101	101	001	110

ADD IMMEDIATE TO MEMORY

AIM	Y/Y,X	33XYYYYY
O/OP/OL	Z	0C0ZZZZZ

The address field Y is added in binary to the address field of a second location. The result is placed in the address field of the second location. The operation code field and ACF of the second location are not changed. Y can be modified under control of the instruction word ACF (X), which is either zero (no modification) or seven (modification). The second location is defined by a Second Address Sequence. The SAS word can be an Operand, Operand Pointer, or Operand Link:

OPERAND. Address field Y is added to the address field of the SAS word and the result is placed in the address field of the SAS word.

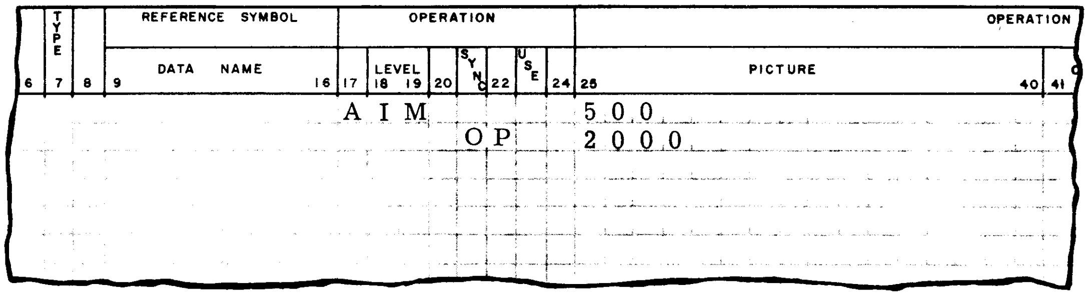
OPERAND POINTER. Address field Y is added to the address field of location Z and the result is placed in the address field of location Z.

OPERAND LINK. Location Z contains another SAS word which is either an Operand, Operand Pointer, or another Operand Link.

Overflow: A carry out of the address field is lost and the Overflow indicator remains unchanged.

Example:

Add the value $(500)_{10}$ to the contents of the address field of location 2000. The address field of location 2000 contains $(150)_{10}$. The value of $(500)_{10}$ is in the address field of the AIM instruction. The AIM instruction is in location 1700.



	TYPE		REFERENCE SYMBOL	OPERATION							OPERATION	
			DATA NAME		LEVEL		SYNC		USE		PICTURE	
6	7	8	9 16	17	18 19	20		22		24	25 40	41
				A	I M						5 0 0	
							O	P			2 0 0 0	

Locations Affected:

Before

1700

A I M	0	$(500)_{10}$

2000

x x x	$(150)_{10}$

After

1700

A I M	0	$(500)_{10}$

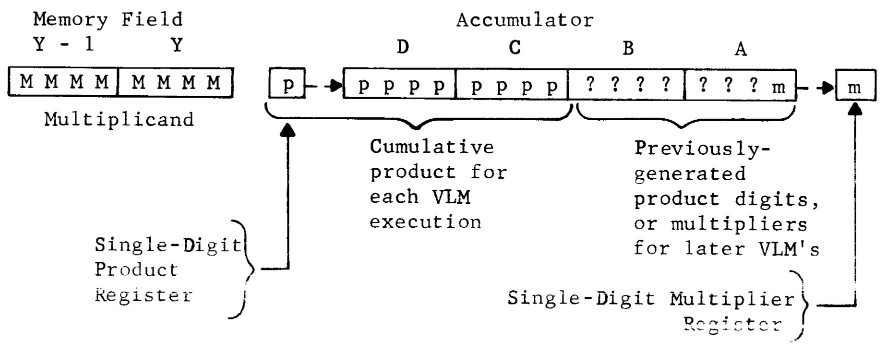
2000

x x x	$(650)_{10}$

VARIABLE LENGTH MULTIPLY

VLM	Y/Y, X	27XYYYYY

The contents of locations Y-1 and Y are multiplied by the least-significant character in the accumulator, generating a 9-digit product in the most-significant positions of the accumulator. Details of execution are dependent upon the status of the First Time indicator (FTI).


Memory Field
Y - 1
Y
M M M M
M M M M
Multiplicand
Accumulator
D
C
B
A
p
p p p p
p p p p
? ? ? ?
? ? ? m
m
Cumulative product for each VLM execution
Previously-generated product digits, or multipliers for later VLM's
Single-Digit Product Register
Single-Digit Multiplier Register

If the FTI is on:

1. The two least-significant accumulator words (B and A) are shifted right one character; the least-significant digit in the accumulator moves into the single-digit multiplier register; a zero is placed temporarily in the most-significant character position of accumulator word B.

2. Accumulator words D and C are cleared to zeros.

3. The contents of Y-1 and Y are multiplied by the contents of the single-digit multiplier register, and the nine-digit product is placed in the single-digit product register and accumulator words D and C.

4. The sign of the product, as determined by the signs of the multiplier digit and the least-significant digit of Y, is placed in the zone bits of the least-significant digit in accumulator word C.

5. The First Time indicator is turned off.

6. Accumulator words D and C are shifted right one character, and the least-significant digit of word C replaces the temporary zero previously placed in the most-significant position of word B (the remaining three digits of word B are unaffected). The single-digit product register contents enter the most-significant position of word D.

If the FTI is off:

1. The two least-significant accumulator words (B and A) are shifted right alpha one character; the least-significant digit in the accumulator moves into the single-digit multiplier register; a zero is placed temporarily in the most-significant position of accumulator word B.

2. The contents of Y-1 and Y are multiplied by the contents of the single-digit multiplier register. The sum of the nine-digit product and the contents of accumulator words D and C is placed in the single-digit product register and accumulator words D and C.

3. Accumulator words D and C are shifted right one character, and the least-significant digit of word C replaces the temporary zero previously placed in the most-significant position of word B (the remaining three digits of word B are unaffected). The single-digit product register contents enter the most-significant position of word D.

The multiplier character is lost during each VLM execution. The multiplicand field, Y-1 and Y, is not changed. The accumulator length is not changed.

The multiplicand is always considered to be an eight-digit field consisting of the memory word specified by the VLM address field and the next-most-significant word (Y-1 and Y).

To form the product of a memory field and a multi-digit multiplier, the VLM instruction must be executed once for each multiplier digit. Thus, an N-digit multiplier requires N executions of the VLM, up to a maximum of eight.

The product of a single VLM execution contains 9 digits; the product formed with an N-digit multiplier contains 8 + N digits, for a maximum of 16, left-justified in the full accumulator.

The sign of the product appears in the zone bits of the least-significant character of the product; the zone bits of all other product digits are set to zero.

A single VLM instruction can be used to form an 8 + N-digit product by controlling VLM execution with a Branch on Count (BRC) or Branch on Index Count (BXC) instruction that has been set to allow N executions of the VLM. The BRC or BXC instruction automatically turns on the FTI upon completion of the count, thereby reinitializing the FTI for subsequent instructions.

Upon completion of a multiply, if the product is less than 16 digits, it must be shifted right alpha by the program for proper positioning and sign preservation. The number of character positions to be shifted is 8 minus the number of VLM's executed.

The VLM instruction is not affected by, and does not change, the accumulator length. Execution of VLM can result in a -0 product, if one factor is zero and the other is negative. Overflow can never be caused by the VLM.

Example:

Multiply the contents of symbolic location MCND by the single-digit field in memory location MTER and right-justify the product in the accumulator. MCND-1 and MCND contain -0004 1234 and MTER contains +0002. The First Time indicator is on.

6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYN 21	22	USE 23	24	OPERATION / PICTURE 25–40	41
				L	D S						M T E R (Multiplier)	
				V	L M						M C N D (Multiplicand)	
				S	R Q	A					7 (Position the Product)	

Locations Affected:

After Execution of LDS

MCND-1	MCND
0 0 0 4	1 2 3 $\overset{-}{4}$

Accumulator			
♦x x x x	x x x x	x x x x	0 0 0 $\overset{+}{2}$

After Execution of VLM

MCND-1	MCND
0 0 0 4	1 2 3 $\overset{-}{4}$

Accumulator			
♦0 0 0 0	8 2 4 6	$\overset{-}{8}$ x x x	x 0 0 0

After Execution of SRQA

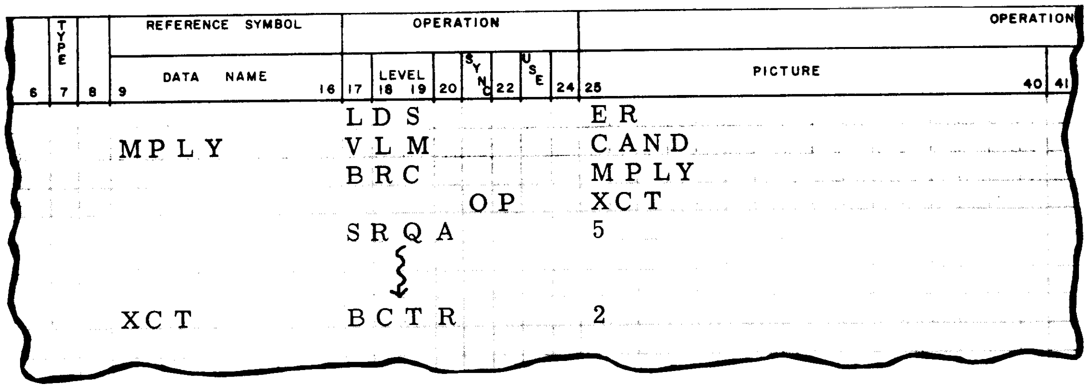
MCND-1	MCND
0 0 0 4	1 2 3 $\overset{-}{4}$

Accumulator			
♦0 0 0 0	0 0 0 0	0 0 0 8	2 4 6 $\overset{-}{8}$

The sequence above leaves FTI reset (off).

Example:

Generate a VLM-BRC loop to multiply an 8-digit memory field in symbolic locations CAND-1 and CAND by a 3-digit field in symbolic location ER, and right-justify the result in the accumulator. CAND-1 and CAND contain +6231 1231 and ER contains +0456. The First Time indicator is on.



```
         MPLY      LDS          ER
                   VLM          CAND
                   BRC          MPLY
                          OP    XCT
                   SRQA         5
         XCT       BCTR         2
```

Locations Affected:

After Execution of LDS

CAND-1	CAND		Accumulator			
6 2 3 1	1 2 3 $\overset{+}{1}$		x x x x	x x x x	x x x x ◆	0 4 5 $\overset{+}{6}$

After Execution of 1st VLM

CAND-1	CAND		Accumulator			
6 2 3 1	1 2 3 $\overset{+}{1}$		3 7 3 8	6 7 3 8	$\overset{+}{6}$ x x x ◆	x 0 4 5

After Execution of 2nd VLM

CAND-1	CAND		Accumulator			
6 2 3 1	1 2 3 $\overset{+}{1}$		3 4 8 9	4 2 8 9	3 $\overset{+}{6}$ x x ◆	x x 0 4

After Execution of 3rd VLM

CAND-1	CAND		Accumulator			
6 2 3 1	1 2 3 $\overset{+}{1}$		2 8 4 1	3 9 2 1	3 3 $\overset{+}{6}$ x ◆	x x x 0

After Execution of SRQA

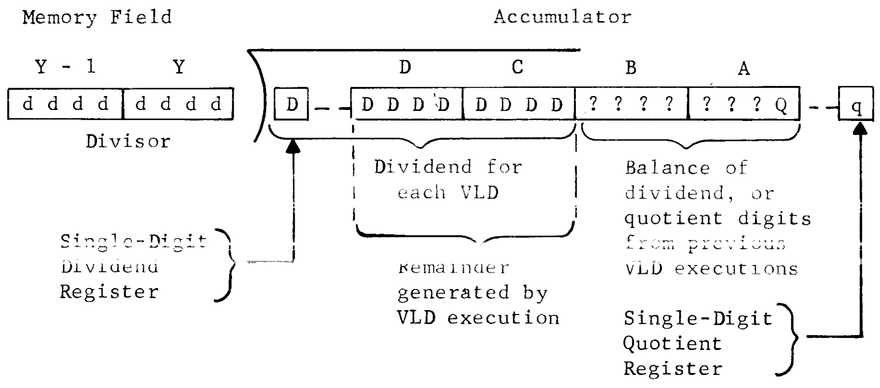
CAND-1	CAND		Accumulator			
6 2 3 1	1 2 3 $\overset{+}{1}$		0 0 0 0	0 2 8 4	1 3 9 2 ◆	1 3 3 $\overset{+}{6}$

The Branch on Count (BRC) instruction and the BCTR pseudo-operation used in the above example are both discussed in the Branch Instructions subsection of the Instruction Repertoire section.

VARIABLE LENGTH DIVIDE

VLD	Y/Y, X	26XYYYYY

The contents of Y-1 and Y are divided into the nine most-significant digits of the four-word accumulator. For each execution of a VLD, a single-digit quotient (with sign) is generated in the least-significant character position of the accumulator, with any remainder, the balance of the dividend, and any previously-generated quotient digits occupying the remaining accumulator positions. Details of VLD execution depend upon the status of the First Time indicator (FTI).


Memory Field
Accumulator
Y - 1
Y
D
C
B
A
d d d d
d d d d
Divisor
D
D D D D
D D D D
? ? ? ?
? ? ? Q
q
Dividend for each VLD
Balance of dividend, or quotient digits from previous VLD executions
Single-Digit Dividend Register
Remainder generated by VLD execution
Single-Digit Quotient Register

If the FTI is on:

1. The Overflow indicator is turned off.
2. The contents of the four-word accumulator are shifted left alpha one character. This causes the sign of the least-significant digit of the dividend to be shifted, even when the dividend consists of 16 digits. The most-significant digit shifts out of the accumulator into a special single-digit dividend register, and a zero is temporarily inserted in the least-significant character position of the accumulator.
3. The First Time indicator is turned off.
4. The 8-digit memory field (Y-1 and Y) is divided into the 9-digit dividend located in the single-digit dividend register and accumulator words D and C. The remainder, if any, is formed in the eight most-significant positions of the accumulator (words D and C). The sign of the remainder is the same as the sign of the 9-digit dividend. The remainder becomes the eight most-significant characters of the dividend for any subsequent VLD.

5. The quotient is formed in a single-digit quotient register. The sign of the quotient is determined by the sign of the least-significant characters of the divisor and the sign of the least-significant character of the 9-digit dividend.

6. The single-digit quotient, with sign, replaces the temporary zero in the least-significant accumulator position.

If the FTI is off:

1. The contents of the four-word accumulator are shifted left decimal one character. This causes the sign bits of the least-significant character of the accumulator to be reset to zero before shifting is begun. The most-significant digit shifts out of the accumulator into a special single-digit dividend register, and a zero is temporarily inserted in the least-significant character position of the accumulator.

2. The 8-digit memory field (Y-1 and Y) is divided into the 9-digit dividend located in the single-digit dividend register and accumulator words D and C. The remainder, if any, is formed in the eight most-significant positions of the accumulator (words D and C). The sign of the remainder is the same as the sign of the 9-digit dividend. The remainder becomes the eight most-significant characters of the dividend for any subsequent VLD.

3. The quotient is formed in a single-digit quotient register. The sign of the quotient is determined by the sign of the least-significant characters of the divisor and the sign of the least-significant character of the 9-digit dividend.

4. The single-digit quotient, with sign, replaces the temporary zero in the least-significant accumulator position.

The contents of the divisor memory field and the accumulator length setting are not changed.

The VLD always regards the divisor as an 8-digit, 2-word memory field, with the sign in the zone bits of the least significant digit.

Successive executions of the VLD can result in the formation of a quotient of up to eight digits, one for each VLD execution. After N executions, the N-digit quotient is found right-justified in accumulator words B and A.

The full dividend can contain 9 to 16 digits, depending upon the number of VLD executions. For N executions of the VLD, the effective dividend is 8+N digits. For each execution of a VLD, the dividend is considered to be the nine digits held in accumulator words D and C and the most-significant digit of accumulator word B. After execution, words D and C contain the portion of the remainder generated by the single VLD. This remainder and the next adjacent character in the accumulator form the effective dividend for any subsequent VLD execution. After N executions, the 8-digit remainder is in accumulator words D and C and the sign is the same as the sign of the original dividend. The least-significant N positions of the accumulator will contain the quotient.

Overflow: The Overflow indicator is automatically turned off at the beginning of the VLD, if the FTI is on.

The Overflow indicator will be turned on, and the VLD will not be completed, if either or both of the following conditions occur:

1. The absolute magnitude of the divisor is less than 100,000 (decimal).
2. The absolute magnitude of the divisor is less than or equal to the most-significant eight digits of the four-word accumulator.

To form the quotient of a divisor in memory and a dividend in the accumulator, the VLD must be repeated as many times as the number of desired quotient digits. This is most effectively done with a programmed loop containing the VLD and BRC (Branch on Count) or BXC (Branch on Index Count) instructions and providing automatic control of the FTI.

A divide subroutine for forming an 8-digit quotient and relieving the programmer of much coding detail is provided as part of the GE-425/435 programming systems. Provisions and requirements for its use are described in the subroutine write-up.

Example:

Generate a VLD-BRC loop to divide the memory field in symbolic locations DVND-3 through DVND, which contains -0000 0008 8888 8888, by the memory field in symbolic locations DVSR-1 and DVSR, which contains -0400 0000. Develop three quotient characters. The First Time indicator is on.

REFERENCE SYMBOL / DATA NAME	OPERATION / LEVEL	OPERATION / PICTURE	
	L D Q	D V N D	
	S L Q A	5	(Position Dividend)
D I V D	V L D	D V S R	
	B R C	D I V D	
	O P	D V C T	
D V C T	B C T R	2	

Locations Affected:

After Execution of SLQA

DVSR-1	DVSR
	−
0 4 0 0	0 0 0 0

Accumulator			
		−	+
0 0 8 8	8 8 8 8	8 8 8 0	0 0 0 0

After Execution of 1st VDD

DVSR-1	DVSR
	−
0 4 0 0	0 0 0 0

Accumulator			
		−	−
0 0 8 8	8 8 8 8	8 8 0 0	0 0 0 2

After Execution of 2nd VLD

DVSR-1	DVSR
	−
0 4 0 0	0 0 0 0

Accumulator			
		−	−
0 0 8 8	8 8 8 8	8 0 0 0	0 0 2 2

After Execution of 3rd VLD

DVSR-1	DVSR
	−
0 4 0 0	0 0 0 0

Accumulator			
	−		+
0 0 8 8	8 8 8 8	0 0 0 0	0 2 2 2
Remainder			Quotient

COMPARE INSTRUCTIONS

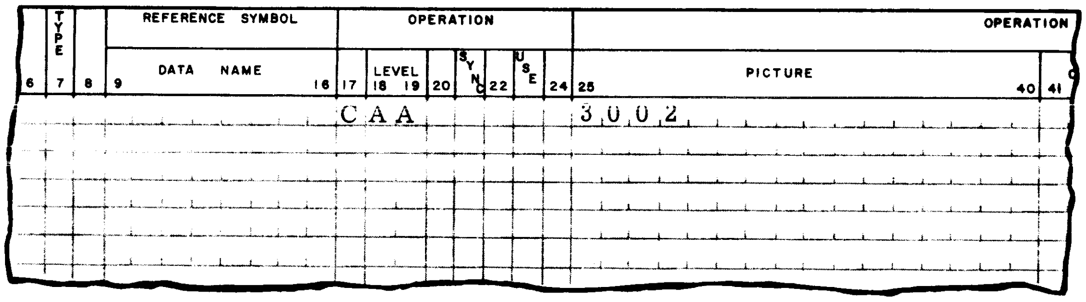
COMPARE ALPHANUMERIC ACCUMULATOR TO MEMORY

CAA	Y/Y, X	03XYYYYY

The contents of the working accumulator are compared with the contents of a memory field whose least-significant location is Y and whose length corresponds to the length of the working accumulator. Comparison is based upon the absolute binary contents of the two fields, although the fields can be either binary or alphanumeric. If the fields are considered to be alphanumeric, the result of the comparison will be in accord with the GE-400 collating sequence. The result of the comparison (accumulator less, equal, or greater) is placed in the Comparison indicators. The accumulator contents and length and the memory field contents are not changed.

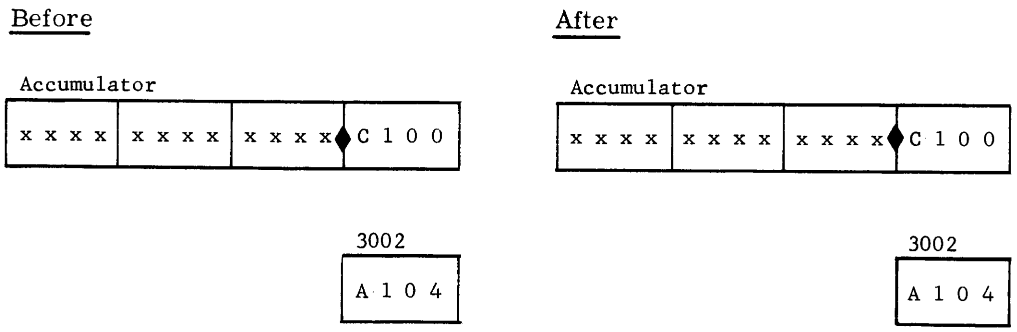
Example:

Compare the contents of the working accumulator absolutely with the contents of location 3002. The accumulator contains C100 and location 1002 contains A104.



6	7 TYPE	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 22	USE 24	OPERATION / PICTURE 25–40	41
				CAA					3002	

Locations Affected:



Before
Accumulator
x x x x x x x x x x x x C 1 0 0
3002
A 1 0 4
After
Accumulator
x x x x x x x x x x x x C 1 0 0
3002
A 1 0 4

The Comparison indicators are set to "greater".

If, in the example, the accumulator had contained 200J and location 3002 had contained 2009, the Comparison indicators would be set to "greater". It should be noted that 200J and -2001 have identical bit configurations.

COMPARE DECIMAL ACCUMULATOR TO MEMORY

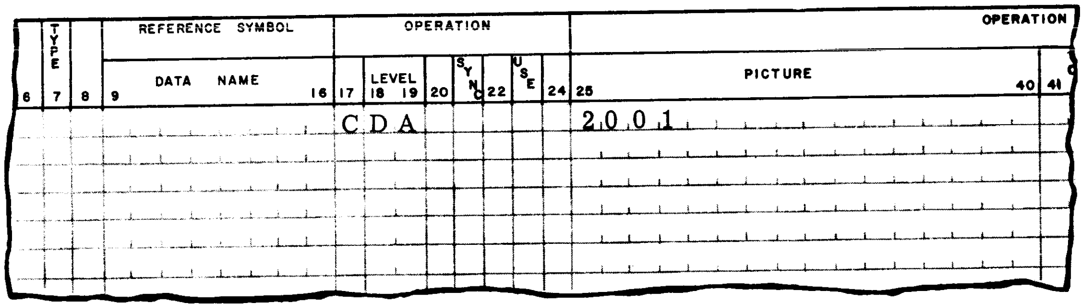
CDA	Y/Y, X	02XYYYYY

The contents of the working accumulator are compared algebraically with the contents of a memory field whose least-significant location is Y and whose length corresponds with the length of the working accumulator. The only zone bits considered in the comparison are the signs of the memory and accumulator fields. When the signs are compared, a minus sign (10 in the zone bits) is less than all three plus sign configurations (00, 01, or 11). The result of the decimal comparison (accumulator field less, equal, or greater) is placed in the Comparison indicators. The accumulator contents and length and the memory field contents are unchanged.

The 8421 numeric bits constituting the decimal value of any character can be the non-numerics, 1010, 1011, 1100, 1101, 1110 or 1111, which are greater, in the sequence shown, than the greatest numeric value 1001.

Example:

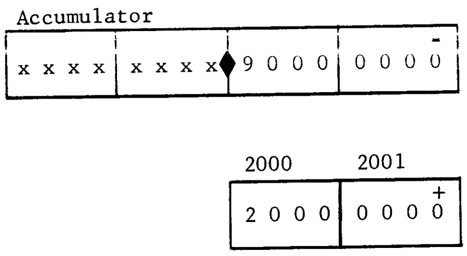
Compare the contents of the working accumulator algebraically with the contents of locations 2000 and 2001. The working accumulator contains -9000 0000 and locations 2000 and 2001 contain +2000 0000.



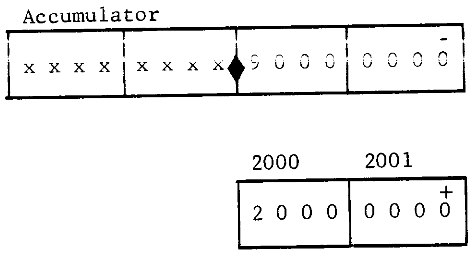
REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17–24	OPERATION / PICTURE 25–40
	CDA	2001

Locations Affected:

Before



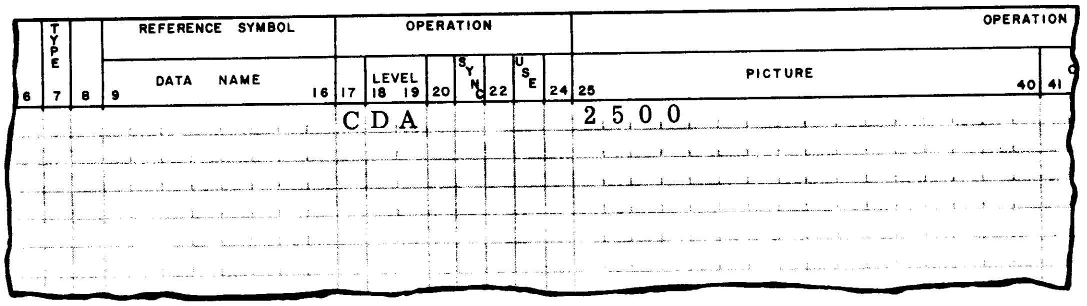
After



The Comparison indicators are set to "less".

Example:

Compare the contents of the working accumulator algebraically with the contents of location 2500. The working accumulator contains +Ɨ200 (an Ignore character is in the most-significant character position) and memory location 2500 contains +8000.



TYPE	REFERENCE SYMBOL / DATA NAME	OPERATION / LEVEL	PICTURE
		CDA	2500

Locations Affected:

Before

Accumulator

x x x x	x x x x	x x x x ◆	Ɨ 2 0 0 (+)

2500

8 0 0 0 (+)

After

Accumulator

x x x x	x x x x	x x x x ◆	Ɨ 2 0 0 (+)

2500

8 0 0 0 (+)

The Comparison indicators are set to "greater" (the Ignore character has a numeric bit configuration of 1111, which is larger than the numeric bit configuration of a decimal 8, which is 1000).

COMPARE SECOND TO FIRST MEMORY

CMM	Y/Y, X	04XYYYYY
O/OP/OL	Z	0C0ZZZZZ

The contents of location Y and the contents of a second location are compared. Comparison is based upon the absolute binary contents of the two fields, although the fields can be either binary or alphanumeric. If the fields are considered to be alphanumeric, the result of the comparison will be in accord with the GE-400 collating sequence. The result of the comparison (second location less, equal, or greater) is placed in the Comparison indicators. The contents of the compared locations are unchanged. Y can be modified under control of the instruction word ACF (X), which is either zero (no modification) or seven (modification). The second location is defined by a Second Address Sequence. The SAS word can be an Operand, Operand Pointer, or an Operand Link:

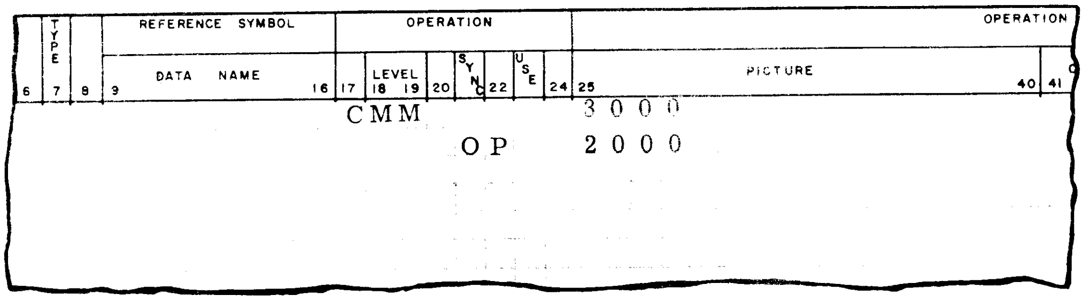
OPERAND. The contents of location Y and the SAS word are compared absolutely.

OPERAND POINTER. The contents of locations Y and Z are compared absolutely.

OPERAND LINK. Location Z contains another SAS word which is either an Operand, Operand Pointer, or another Operand Link.

Example:

Compare the contents of location 2000 absolutely with the contents of location 3000. Location 2000 contains $(70000000)_8$ and location 3000 contains $(30420000)_8$.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17 / LEVEL 18 19 / 20 / SYNC 22 / USE 24	OPERATION / PICTURE 25–40	41
				CMM	3000	
				OP	2000	

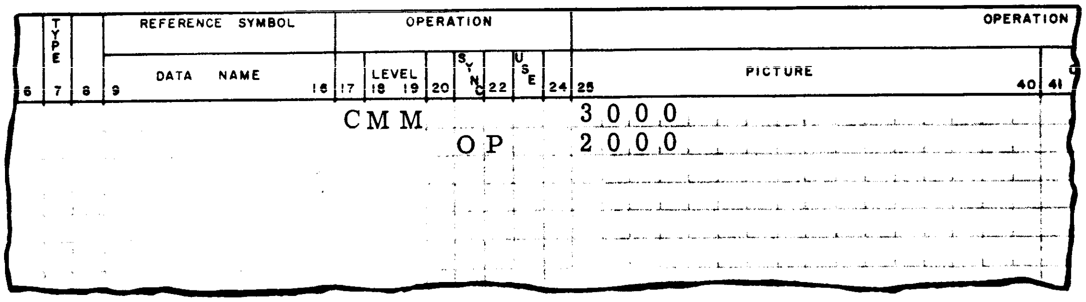
Locations Affected:

Before	After
2000	2000
$(70000000)_8$	$(70000000)_8$
3000	3000
$(30420000)_8$	$(30420000)_8$

The Comparison indicators are set to "greater".

Example:

Compare the contents of location 2000 absolutely with the contents of location 3000. Location 2000 contains C100 and location 3000 contains A104.



REFERENCE SYMBOL / DATA NAME (9–16)	OPERATION (17–24)	OPERATION / PICTURE (25–40)
	CMM	3000
	OP	2000

Locations Affected:

Before	After
2000	2000
$\boxed{C\ 1\ 0\ 0}$	$\boxed{C\ 1\ 0\ 0}$
3000	3000
$\boxed{A\ 1\ 0\ 4}$	$\boxed{A\ 1\ 0\ 4}$

The Comparison indicators are set to "greater".

COMPARE MEMORY TO IMMEDIATE

CMI	Y/Y, X	01XYYYYY
O/OP/OL	Z	0C0ZZZZZ

Address field Y and the address field of a second location are compared absolutely. The result of the comparison (second location address field less, equal, or greater) is placed into the Comparison indicators. The instruction word and the contents of the second location are unchanged. Y can be modified under control of the instruction word ACF (X), which is either zero (no modification) or seven (modification). The second location is defined by a Second Address Sequence. The SAS word can be an Operand, Operand Pointer, or Operand Link:

OPERAND. Address field Y and the address field of the SAS word are compared absolutely.

OPERAND POINTER. Address field Y and the address field of location Z are compared absolutely.

OPERAND LINK. Location Z contains another SAS word which is either an Operand, Operand Pointer, or another Operand Link.

Example:

Compare the address field of location 2000 absolutely with the address field of the CMI instruction word. The address field of the word in location 2000 contains $(3050)_{10}$ and the address field of the CMI instruction word contains $(1050)_{10}$. The CMI instruction is in location 3000.

6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17–24	OPERATION / PICTURE 25–40	41
				CMI	1050	
				OP	2000	

Locations Affected:

Before

2000

x x x	$(3050)_{10}$

3000

C M I	$(1050)_{10}$

After

2000

x x x	$(3050)_{10}$

3000

C M I	$(1050)_{10}$

The Comparison indicators are set to "greater".

BRANCH INSTRUCTIONS

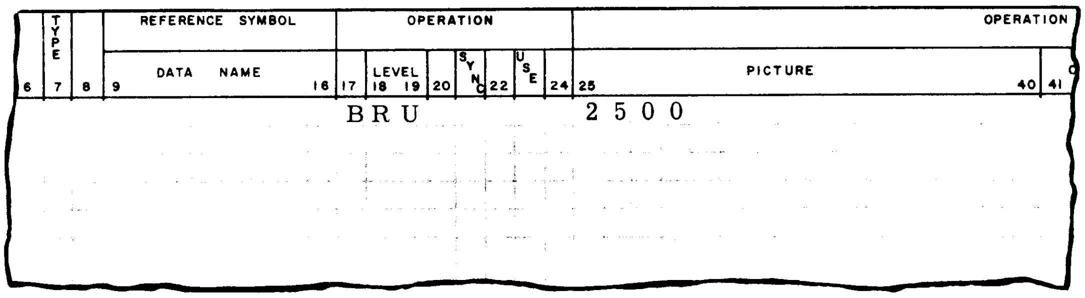
BRANCH UNCONDITIONALLY

BRU	Y/Y, X	10XYYYYYY

The contents of the P-counter are replaced by the instruction address field Y, causing a program branch to location Y.

Example:

Branch unconditionally to location 2500. The BRU instruction is in location 4000.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 22	USE 24	OPERATION / PICTURE 25–40	41
				BRU					2500	

Locations Affected:

Before	After
P-counter	P-counter
$(4001)_{10}$	$(2500)_{10}$

STORE PROGRAM COUNTER AND BRANCH

SPB	Y/Y, X	17XYYYYY
C, OP, OL	Z	0C0ZZZZZ

The contents of the P-counter (the address of the word following the SAS word in the P-sequence) are stored in memory, and the instruction address field Y is placed in the program counter, effecting a branch to location Y. Y can be modified under control of the instruction word ACF (X), which is either zero (no modification) or seven (modification). The location in which the previous contents of the program counter is stored is defined by a Second Address Sequence. The SAS word can be an Operand, Operand Pointer, or Operand Link:

OPERAND. The program counter contents are stored in the address field of the SAS word.

OPERAND POINTER. The program counter contents are stored in the address field of location Z.

OPERAND LINK. Location Z contains another SAS word which is either an Operand, Operand Pointer or another Operand Link.

In any location in which the P-counter is stored, only the address field is affected. The other bit positions remain unchanged.

Example:

Store the contents of the P-counter in the address field of location 2300 and branch to location 2500. The SPB instruction is in location 2000.

6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 21 22	USE 23 24	OPERATION / PICTURE 25–40	41
				S	P B				2 5 0 0	
							O P		2 3 0 0	

Locations Affected:

Before

P-counter

(2 0 0 2) $_{10}$

2300

x x x	x x x x x

After

P-counter

(2 5 0 0) $_{10}$

2300

x x x	(2 0 0 2) $_{10}$

BRANCH IF EQUAL

BRE	Y/Y, X	13XYYYYY

BRANCH IF LESS

BRL	Y/Y, X	14XYYYYY

BRANCH IF GREATER

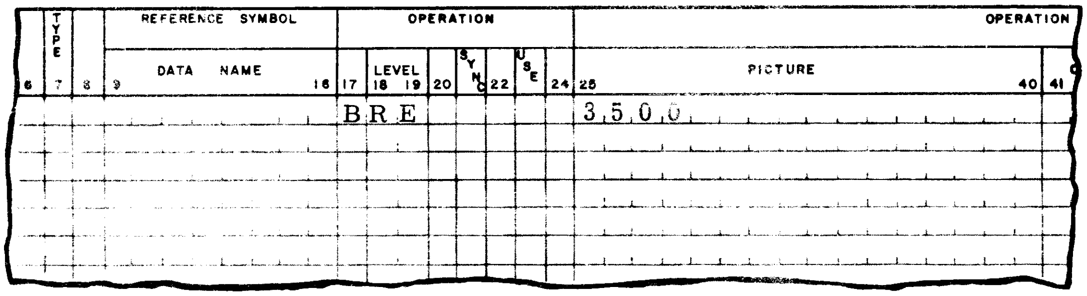
BRG	Y/Y, X	12XYYYYY

If the specified condition exists in the Comparison indicators, the contents of the program counter are replaced by the instruction address field Y, causing a program branch to the location specified. If the specified condition does not exist, the branch is not made and the next instruction is taken in normal sequence.

NOTE: Execution of the conditional branch instructions does not affect the Comparison indicators.

Example:

The Comparison indicators contain an "equal" condition. Indicate the result of executing BRE 3500. The BRE instruction is in location 2000.



	TYPE		REFERENCE SYMBOL	OPERATION								OPERATION	
			DATA NAME		LEVEL		SYN		USE			PICTURE	
6	7	8	9 16	17	18 19	20		22		24		25 40	41
				BRE								3500	

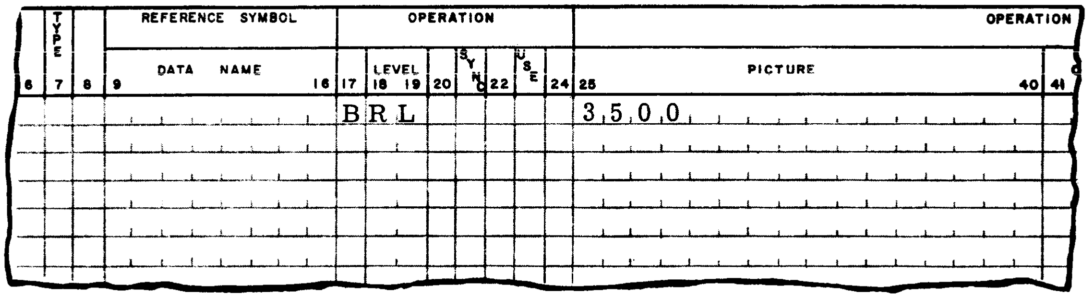
Locations Affected:

Before	After
P-counter	P-counter
$(2001)_{10}$	$(3500)_{10}$

The program branches to 3500, the new location in the P-counter, because the setting of the Comparison indicators matched the condition (equal) specified in the instruction.

Example:

The Comparison indicators contain a "greater" condition. Indicate the result of executing BRL 3500. The BRL instruction is in location 2000.



	TYPE		REFERENCE SYMBOL	OPERATION					OPERATION	
6	7	8	DATA NAME 9–16	17	LEVEL 18 19	20	SYNC 22	USE 24	PICTURE 25–40	41
				B	R L				3500	

Locations Affected:

Before	After
P-counter	P-counter
$(2001)_{10}$	$(2001)_{10}$

The program does not branch, but continues in sequence to 2001, because the setting of the Comparison indicators did not match the condition (less) specified in the instruction.

Example:

The Comparison indicators contain an "equal" condition. Indicate the result of executing BRG 3500. The BRG instruction is in location 2000.

	TYPE		REFERENCE SYMBOL	OPERATION					OPERATION	
			DATA NAME		LEVEL		SYN	USE	PICTURE	
6	7	8	9 16	17	18 19	20	22	24	25 40	41
				BRG					3500	

Locations Affected:

Before	After
P-counter	P-counter
$(2001)_{10}$	$(2001)_{10}$

The program does not branch, but continues in sequence to 2001 because the setting of the Comparison indicators did not match the condition (greater) specified in the instruction.

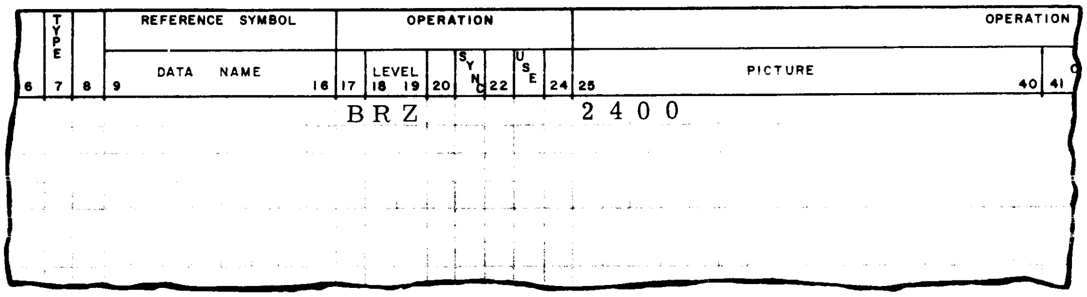
BRANCH IF ZERO

BRZ	Y/Y, X	15XYYYYY

A test is made of the working accumulator bits. If all working accumulator bits are zero, the contents of the program counter are replaced by address field Y and a branch is taken. If any working accumulator bits are not zero, the next instruction in normal sequence is executed.

Example:

The working accumulator contains all 0-bits. Indicate the result of executing BRZ 2400. The BRZ instruction is in location 2000.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION / LEVEL 17–19	20	SYNC 22	USE 24	OPERATION / PICTURE 25–40	41
				BRZ				2400	

Locations Affected:

Before

Accumulator

x x x x ♦	0 0 0 0	0 0 0 0	0 0 0 0

P-counter

$(2001)_{10}$

After

Accumulator

x x x x ♦	0 0 0 0	0 0 0 0	0 0 0 0

P-counter

$(2400)_{10}$

The program branches to location 2400, the new location in the P-counter, because the working accumulator contained all zeros, and matches the condition specified in the instruction.

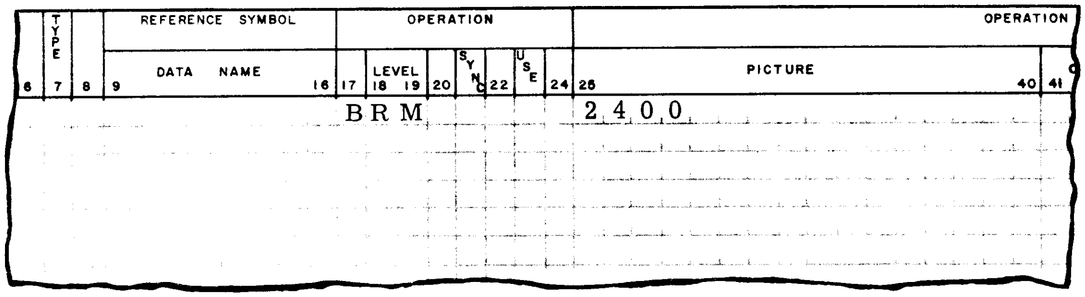
BRANCH IF MINUS

BRM	Y/Y, X	11XYYYYY

The sign of the accumulator, contained in the least-significant character of the accumulator, is tested. If the sign is minus (10 in the zone bits), the contents of the program counter are replaced by address field Y and a branch is taken. If the sign is not minus, the next instruction in normal sequence is executed.

Example:

The sign of the accumulator is plus. Indicate the result of executing BRM 2400. The BRM instruction is in location 2000.



TYPE	REFERENCE SYMBOL / DATA NAME	OPERATION / LEVEL	SYNC	USE	OPERATION / PICTURE
		BRM			2400

Locations Affected:

Before

Accumulator

x x x x	x x x x◆2 3 2 0		6 0 0 $\overset{+}{0}$

P-counter

$(2001)_{10}$

After

Accumulator

x x x x	x x x x◆2 3 2 0		6 0 0 $\overset{+}{0}$

P-counter

$(2001)_{10}$

The program does not branch, but continues in sequence to 2001, because the sign of the accumulator did not match the sign (minus) specified in the instruction.

BRANCH ON COUNT

BRC	Y/Y, X	16XYYYYY
OP/OL	Z	0C0ZZZZZ

This instruction operates as a branch for N executions. During the succeeding execution, the branch is not taken, but the instruction is automatically initialized so that, for the next N executions, it again operates as an unconditional branch.

The branch address is contained in the instruction address field Y. The count N is part of a control word held in a second location. The control word format is:

23 ◄— 15	14 ◄— 9	8 ◄— 0
Work Count 511-N	Not Used	Restore Count 511-N

The programmer must provide the control word, with the indicated binary counts in the form 511-N, where N is the number of times that a branch is taken. The control word can be generated by using the BCTR pseudo-operation. The BCTR pseudo-operation is described below.

The work count is incremented each time the instruction is executed. As long as the count remains non-zero, the instruction operates as a branch. On the execution following the Nth execution, when the count becomes zero (the count becomes zero when one is added to 511):

1. The First Time indicator is set on.
2. The restore count is set into the work count field.
3. The program counter steps to the next sequential instruction (no branch occurs).

The restore count field is not changed. A maximum of 511 branches can be specified by N.

Y can be modified under control of the instruction word ACF (X), which is either zero (no modification) or seven (modification).

The location of the control word is defined by a Second Address Sequence. The SAS word can be an Operand Pointer or Operand Link (because of the control word format, it should not be an SAS Operand).

OPERAND POINTER. Location Z contains the control word.

OPERAND LINK. Location Z contains another SAS word which is either an Operand Pointer or another Operand Link.

BTCR Pseudo-Operation - The BCTR pseudo-operation enables the programmer to specify the counter required for the proper execution of the Branch on Count (BRC) and Branch on Index Count (BXC) instructions.

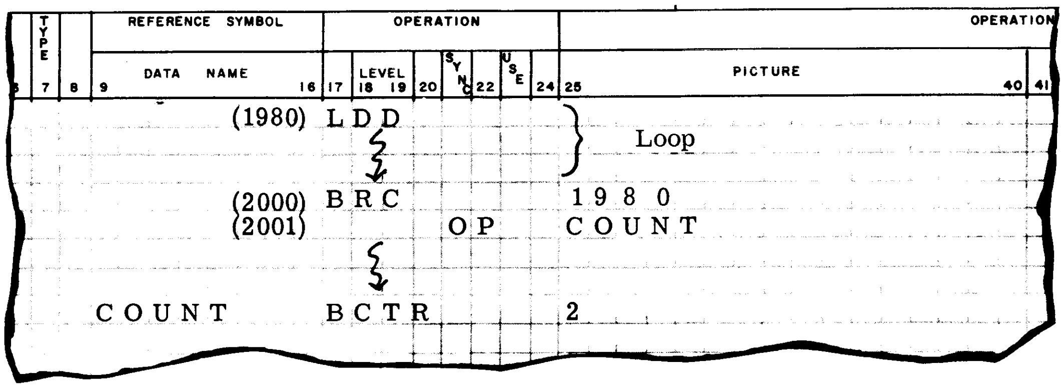
To use the pseudo-operation, the programmer places BCTR in the coding sheet Operation field and an absolute or symbolic expression in the Operation Parameters field corresponding to the number of branches to be taken. The BCTR should also be identified by a symbol in the Reference Symbol field of the coding sheet.

The largest count that can be specified by the BCTR is 511 (decimal). The count word generated controls the number of times the BRC or BXC instruction referencing the count word is executed as a branch.

The BCTR should not be placed in the P-sequence, but in an area of memory used for constants.

Example:

Execute the 20-word loop which begins at location 1980 three times before continuing to the next sequential routine. The Branch on Count instruction is in locations 2000 and 2001 and the control word is in symbolic location COUNT. Since the BRC is after the loop, the branch should be taken twice. The work count and restore count, therefore, are 511 minus 2, which is 509. (Although absolute locations in actual practice are not placed in the Reference Symbol field, they are used in the example below for added clarity.)



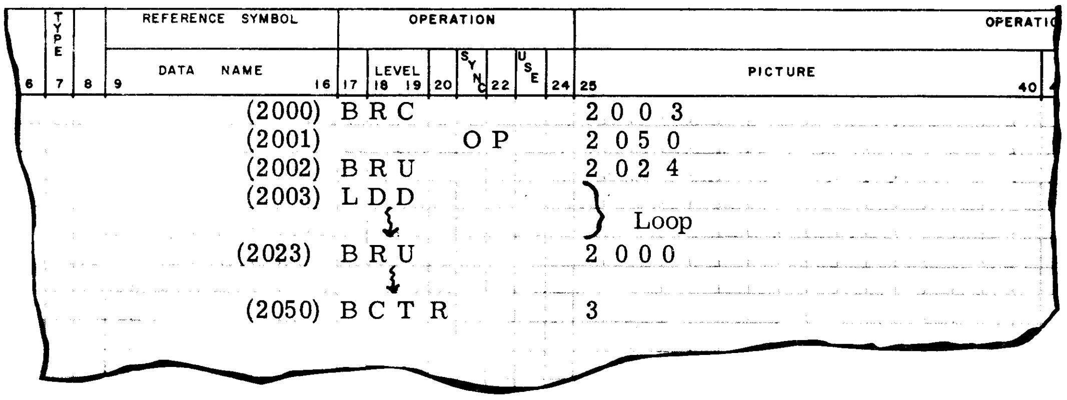
REFERENCE SYMBOL (DATA NAME 9–16)	OPERATION (17–20)	(21–24)	PICTURE (25–40)
(1980)	LDD		Loop
(2000)	BRC		1980
(2001)		OP	COUNT
COUNT	BCTR		2

Locations Affected:

P-Counter	Control Word		
Before execution of the loop:	COUNT		
$(1980)_{10}$	$(509)_{10}$		$(509)_{10}$
After first BRC:	COUNT :		
$(1980)_{10}$	$(510)_{10}$		$(509)_{10}$
After second BRC:	COUNT		
$(1980)_{10}$	$(511)_{10}$		$(509)_{10}$
After third BRC:	COUNT		
$(2002)_{10}$	$(509)_{10}$		$(509)_{10}$

Example:

Execute the 20-word loop which begins at location 2003 three times before continuing to the next sequential routine. The Branch on Count instruction is in locations 2000 and 2001 and the control word is in location 2050. Since the BRC is before the loop, the branch should be taken three times. The work count and restore count, therefore, are 511 minus 3, which is 508. (Although absolute locations in actual practice are not placed in the Reference Symbol field, they are used in the example below for added clarity.)



TYPE		REFERENCE SYMBOL / DATA NAME (9–16)	OPERATION / LEVEL (17–19)	20	SYNC (21–22)	USE (23–24)	PICTURE (25–40)
		(2000)	BRC				2003
		(2001)		OP			2050
		(2002)	BRU				2024
		(2003)	LDD				} Loop
		(2023)	BRU				2000
		(2050)	BCTR				3

Locations Affected:

P-Counter Control Word

Before execution of the loop:

P-Counter		1050		
$(2000)_{10}$		$(508)_{10}$		$(508)_{10}$

After first BRC:

P-Counter		1050		
$(2003)_{10}$		$(509)_{10}$		$(508)_{10}$

After second BRC:

P-Counter		1050		
$(2003)_{10}$		$(510)_{10}$		$(508)_{10}$

After third BRC:

P-Counter		1050		
$(2003)_{10}$		$(511)_{10}$		$(508)_{10}$

After fourth BRC:

P-Counter		1050		
$(2002)_{10}$		$(508)_{10}$		$(508)_{10}$

LOGIC INSTRUCTIONS

AND TO MEMORY

```
ANM             Y/Y, X                    24XYYYYY
 O/OP/OL        Z                         0C0ZZZZZ
```

The contents of location Y are used to modify the contents of a second location. The result is placed in the second location, leaving the contents of Y unchanged. The modification is as follows: for each bit position of location Y that contains a zero, the corresponding bit position of the second location is set to zero; for each bit position of location Y that contains a one, the corresponding bit position of the second location is left unchanged. Y can be modified under the control of the instruction word ACF (X), which is either zero (no modification) or seven (modification). The second location is defined by a Second Address Sequence. The SAS word can be an Operand, Operand Pointer, or Operand Link:

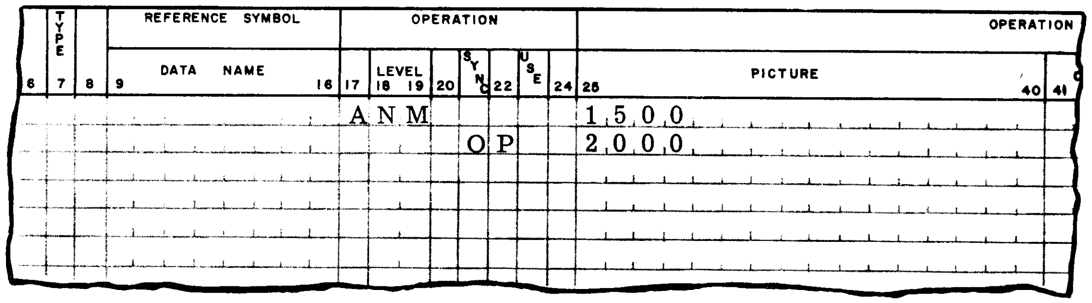
OPERAND. The contents of location Y are used to modify the contents of the SAS word.

OPERAND POINTER. The contents of location Y are used to modify the contents of location Z.

OPERAND LINK. Location Z contains another SAS word which is either an Operand, Operand Pointer, or another Operand Link.

Example:

AND the contents of location 1500 to the contents of location 2000. Location 1500 contains $(73546317)_8$ and location 2000 contains $(42473262)_8$.



			REFERENCE SYMBOL	OPERATION					OPERATION	
6	7 TYPE	8	9 DATA NAME 16	17	LEVEL 18 19	20	SYNC 22	USE 24	25 PICTURE 40	41
				A	NM				1500	
							OP		2000	

Locations Affected:

Before

1500

111	011	101	100	110	011	001	111

2000

100	010	100	111	011	010	110	010

After

1500

111	011	101	100	110	011	001	111

2000

100	010	100	100	010	010	000	010

Example:

Remove the sign bits from the word at location 2000. Location 2000 contains -1234, which is $(01020344)_8$, and location 1500 contains $(77777717)_8$.

TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17 / LEVEL 18 19 / 20 / SYNC 22 / USE 24	OPERATION / PICTURE 25–40	41
			ANM	1500	
			OP	2000	

Locations Affected:

Before

1500

111	111	111	111	111	111	001	111

2000

000	001	000	010	000	011	100	100

After

1500

111	111	111	111	111	111	001	111

2000

000	001	000	010	000	011	000	100

OR INCLUSIVE TO MEMORY

RIM	Y/Y, X	23XYYYYY
O/OP/OL	Z	0C0ZZZZZ

The contents of location Y are used to modify the contents of a second location. The result is placed in the second location, leaving the contents of Y unchanged. The modification is as follows: for each bit position of location Y that contains a one, the corresponding bit position of the second location is set to one; for each bit position of location Y that contains a zero, the corresponding bit position of the second location is left undisturbed. Y can be modified under control of the instruction word ACF (X), which is either zero (no modification) or seven (modification). The second location is defined by a Second Address Sequence. The SAS word can be an Operand, Operand Pointer, or Operand Link:

OPERAND. The contents of location Y are used to modify the contents of the SAS word.

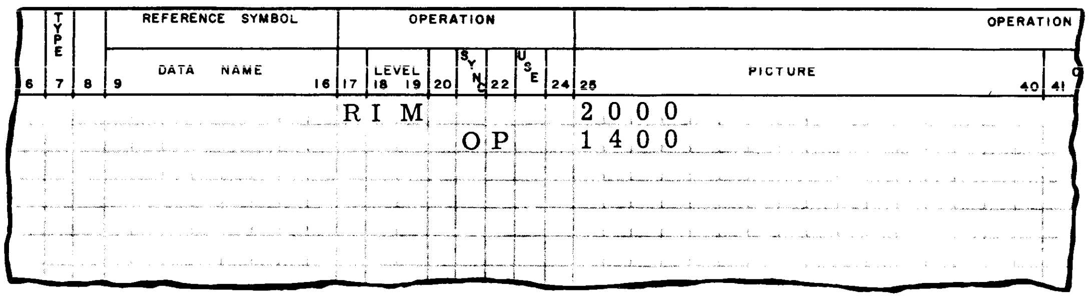
OPERAND POINTER. The contents of location Y are used to modify the contents of location Z.

OPERAND LINK. Location Z contains another SAS word which is either an Operand, Operand Pointer, or another Operand Link.

With this instruction, the SAS word should not ordinarily be coded as an Operand. This instruction would cause an Operand to be changed, thus possibly changing the class of the SAS word to other than Operand.

Example:

Modify the contents of location 1400 with the contents of location 2000 by the OR inclusive method. Location 2000 contains $(64007065)_8$ and location 1400 contains $(12660610)_8$.



TYPE	REFERENCE SYMBOL / DATA NAME	OPERATION	OPERATION / PICTURE
		RIM	2000
		OP	1400

Locations Affected:

Before

2000

110	100	000	000	111	000	110	101

1400

001	010	110	110	000	110	001	000

After

2000

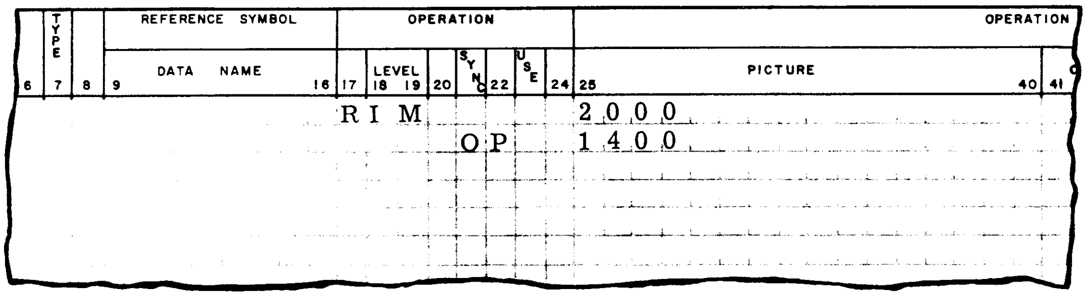
110	100	000	000	111	000	110	101

1400

111	110	110	110	111	110	111	101

Example:

Change the sign of the word at location 1400 from plus (assume 00 zone bits only) to minus (10 in the zone bits). Location 1400 contains +9876, which is $(11100706)_8$, and location 2000 contains $(00000040)_8$.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 21 22	USE 23 24	OPERATION / PICTURE 25–40	41
				RIM					2000	
							OP		1400	

Locations Affected:

<u>Before</u>

2000

000 000 000 000 000 000 100 000

1400

001 001 001 000 000 111 000 110

<u>After</u>

2000

000 000 000 000 000 000 100 000

1400

001 001 001 000 000 111 100 110

OR EXCLUSIVE TO MEMORY

RXM	Y/Y,X	25XYYYYY
O/OP/OL	Z	0C0ZZZZZ

The contents of location Y are used to modify the contents of a second location, leaving the contents of Y unchanged. The modification is as follows: for each bit position of location Y that contains a one, the corresponding bit position of the second location is inverted (zero becomes one, one becomes zero); for each bit position of location Y that contains a zero, the corresponding bit position of the second location is left undisturbed. Y can be modified under the control of the instruction word ACF (X), which is either zero (no modification) or seven (modification). The second location is defined by a Second Address Sequence. The SAS word can be an Operand, Operand Pointer, or Operand Link:

OPERAND. The contents of location Y are used to modify the contents of the SAS word.

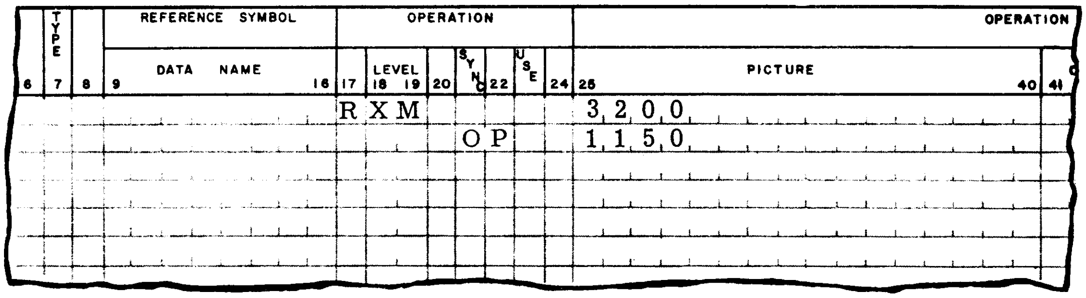
OPERAND POINTER. The contents of location Y are used to modify the contents of location Z.

OPERAND LINK. Location Z contains another SAS word which is either an Operand, Operand Pointer or another Operand Link.

With this instruction, the SAS word should not ordinarily be coded as an Operand. This instruction would cause an Operand to be changed, thus possibly changing the class of the SAS word to other than Operand.

Example:

Modify the contents of location 1150 with the contents of location 3200 by the OR exclusive method. Location 3200 contains $(61526071)_8$ and location 1150 contains $(12561140)_8$.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYN 21 22	USE 23	24	OPERATION / PICTURE 25–40	41
				R	X M					3200	
							O P			1150	

Locations Affected:

Before	After
3200	3200
110 001 101 010 110 000 111 001	110 001 101 010 110 000 111 001
1150	1150
001 010 101 110 001 001 100 000	111 011 000 100 111 001 011 001

Example:

Form the one's complement of the word at location 1150. Location 1150 contains $(00001200)_8$ and location 3200 contains $(77777777)_8$.

6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC	22	USE	24	OPERATION / PICTURE 25–40	41
				R	X M						3200	
							O	P			1150	

Locations Affected:

Before		After	
3200	$(77777777)_8$	3200	$(77777777)_8$
1150	$(00001200)_8$	1150	$(77776577)_8$

FIXED INDEX WORD INSTRUCTIONS

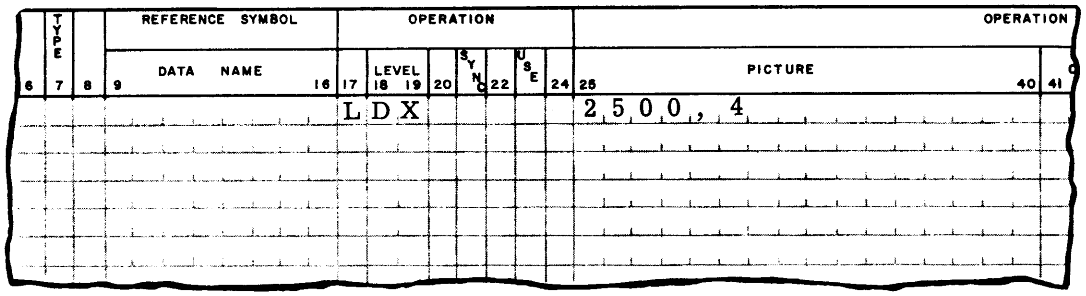
LOAD INDEX

LDX	Y, X	30XYYYYY

The entire contents of location Y are moved to fixed index word X (1-6). The contents of Y are not changed. Y cannot be modified. This instruction is a special form of the MFM instruction and has the same operation code.

Example:

Load the contents of location 2500 into fixed index word 4. Location 2500 contains +1234.



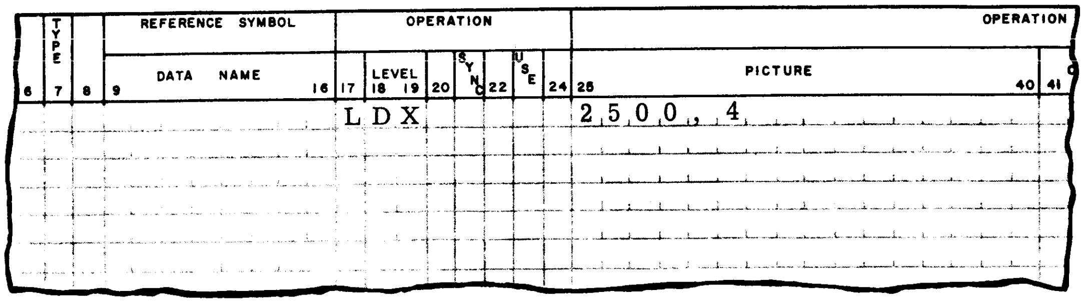
6	TYPE 7	8	REFERENCE SYMBOL DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC	22	USE	24	OPERATION PICTURE 25–40	41
				L	DX						2500, 4	

Locations Affected:

Before		After	
2500	1234+	2500	1234+
4	xxxx	4	1234+

Example:

Load the contents of location 2500 into fixed index word 4. Location 2500 contains a BRU 1000 instruction.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC	22	USE	24	OPERATION / PICTURE 25–40	41
				LDX							2500,4	

Locations Affected:

Before

2500

B R U	0	$(1000)_{10}$

0004

x x x	x x x x x

After

2500

B R U	0	$(1000)_{10}$

0004

B R U	0	$(1000)_{10}$

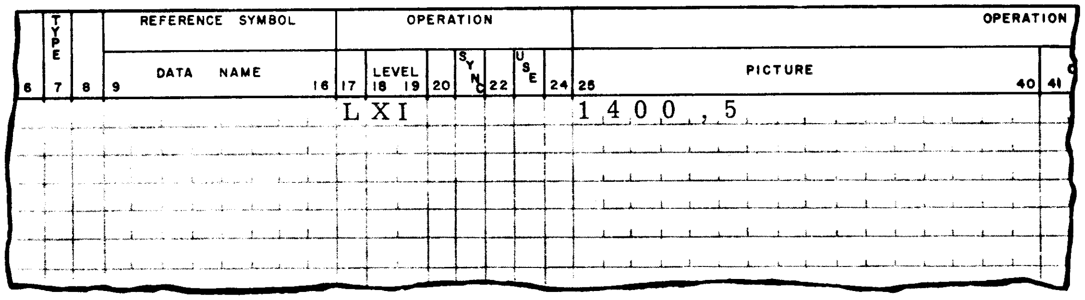
LOAD INDEX WITH IMMEDIATE

LXI	Y, X	31XYYYYY

The address field Y is moved to the address field of fixed index word X (1-6). The operation code field and ACF of index word X are unchanged. Y cannot be modified. This instruction is a special form of the MFI instruction and has the same operation code.

Example:

Move the value 1400 to the address field of fixed index word 5. The LXI instruction is in location 2000.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9-16	OPERATION 17	LEVEL 18 19	20	SYNC	22	USE	24	OPERATION / PICTURE 25-40	41
				L	X I						1400, 5	

Locations Affected:

Before

2000

L X I	5	$(1400)_{10}$

5

x x x	x x x x x

After

2000

L X I	5	$(1400)_{10}$

5

x x x	$(1400)_{10}$

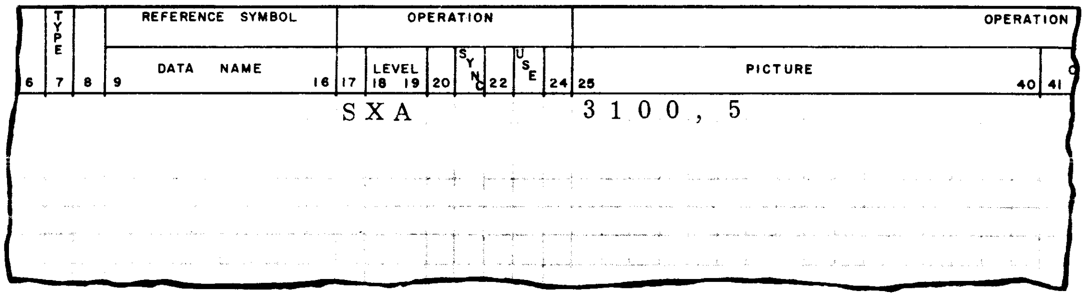
STORE INDEX IN ADDRESS FIELD

SXA	Y, X	32XYYYYY

The address field of location Y is replaced by the address field of fixed index word X (1-6). Index word X and the operation code field and ACF of Y are not changed. Y cannot be modified. This instruction is a special form of the MTA instruction and has the same operation code.

Example:

Move the contents of the address field of fixed index word 5 to the address field of location 3100. The address field of the fixed index word contains $(2500)_{10}$.



TYPE
REFERENCE SYMBOL
OPERATION
OPERATION
DATA NAME
LEVEL
SYNC
USE
PICTURE
6 7 8 9 16 17 18 19 20 22 24 25 40 41

SXA 3100, 5

Locations Affected:

Before

5

x x x	$(2500)_{10}$

3100

x x x	x x x x x

After

5

x x x	$(2500)_{10}$

3100

x x x	$(2500)_{10}$

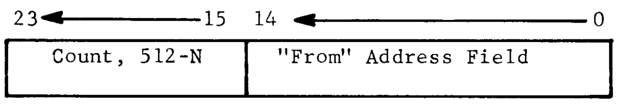
MOVE ON INDEX CONTROL

MXC Y, X 06XYYYYY

The contents of a memory field, whose most-significant word is at location Y. are replaced by words moved from a second memory field. The second memory field is not changed. Y cannot be modified.

The location of the most-significant word in the memory field to be moved is specified in a control word which also contains a control count of the number of words to be moved. The control count is expressed in binary, in the form 512-N, where N is the number of words to be moved (N can range from 1 to 512). The control word is held in fixed index word X (1-6).

The control word format is:



N = Number of words to be moved.

The control word can be generated by using the MCTR pseudo-operation as described under the MOV instruction.

This instruction is a special form of the MOV instruction and has the same operation code.

Example:

Move the contents of the four-word memory field that starts at location 3000 to the memory field that starts at location 2500. Location 3000 contains LAST △PAY △△06 1563. Fixed index word 3 contains the control word.

6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION / LEVEL 17–19	20	SYNC	22	USE	24	OPERATION / PICTURE 25–40	41
				MXC						2500 , 3	

Locations Affected:

Before

3

$(508)_{10}$	$(3000)_{10}$

3000	3001	3002	3003
L A S T	△ P A Y	△ △ 0 6	1 5 6 3

2500	2501	2502	2503
x x x x	x x x x	x x x x	x x x x

After

3

$(508)_{10}$	$(3000)_{10}$

3000	3001	3002	3003
L A S T	△ P A Y	△ △ 0 6	1 5 6 3

2500	2501	2502	2503
L A S T	△ P A Y	△ △ 0 6	1 5 6 3

ADD BINARY TO INDEX

ABX	Y, X	34XYYYYY

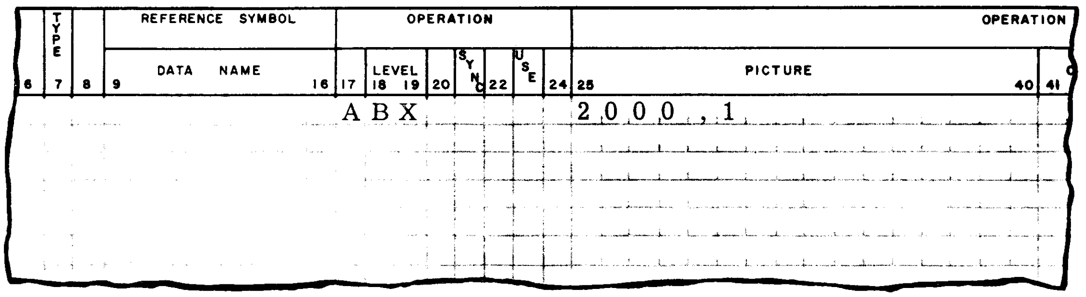
The contents of location Y are added absolutely, in binary, to the contents of fixed index word X (1-6). The result is placed in the fixed index word. The contents of Y are not changed. Y cannot be modified.

Overflow: A carry out of the most-significant bit position of the fixed index word is lost and causes the Overflow indicator to be turned on.

This instruction is a special form of the ABM instruction and has the same operation code.

Example:

Add the binary contents of location 2000 absolutely to the contents of fixed index word 1. The contents of location 2000 are $(00012000)_8$ and the contents of the fixed index word are $(00000064)_8$.



TYPE	REFERENCE SYMBOL / DATA NAME	OPERATION	OPERATION / PICTURE
		ABX	2000,1

Locations Affected:

Before		After	
2000	$(00012000)_8$	2000	$(00012000)_8$
1	$(00000064)_8$	1	$(00012064)_8$

SUBTRACT BINARY FROM INDEX

SBX	Y, X	35XYYYYY

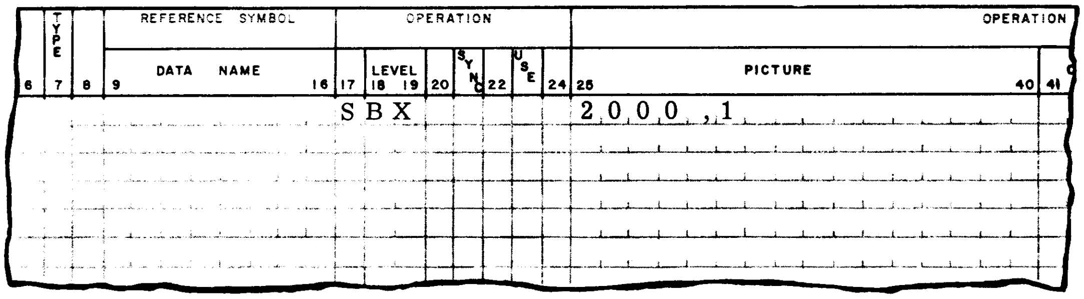
The contents of location Y are subtracted absolutely, in binary, from the contents of fixed index word X (1-6). The result is placed in the fixed index word. The contents of Y are not changed. Y cannot be modified.

Overflow: In the absolute subtraction, if the value to be subtracted exceeds the value contained in the fixed index word, the result (in 2's complement form) is placed in the fixed index word and the Overflow indicator is turned on.

This instruction is a special form of the SBM instruction and has the same operation code.

Example:

Subtract the binary contents of location 2000 absolutely from the contents of fixed index word 1. The contents of location 2000 are $(00005000)_8$ and the contents of the fixed index word are $(00025000)_8$.



TYPE	REFERENCE SYMBOL / DATA NAME	OPERATION / LEVEL	SYN	USE	OPERATION / PICTURE
		SBX			2000,,1

Locations Affected:

Before	After
2000	2000
$(00005000)_8$	$(00005000)_8$
1	1
$(00025000)_8$	$(00020000)_8$

ADD IMMEDIATE TO INDEX

AIX	Y, X	33XYYYYY

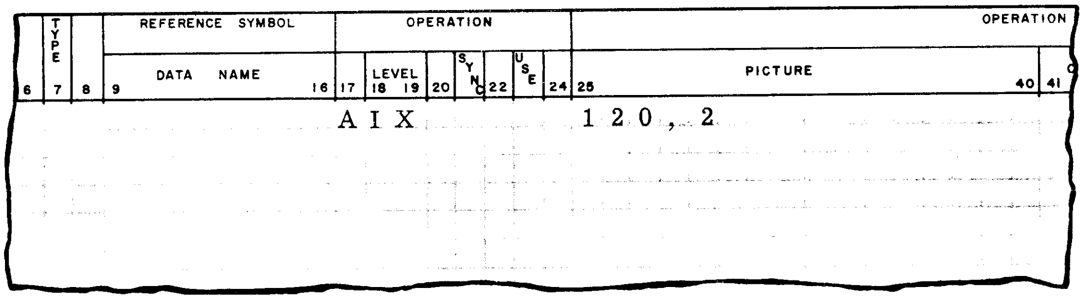
Address field Y is added in binary to the address field of fixed index word X (1-6). The operation code field and ACF of index word X are not changed. Y cannot be modified.

Overflow: A carry out of the index word address field is lost and the Overflow indicator remains unchanged.

This instruction is a special form of the AIM instruction and has the same operation code.

Example:

Add the value $(0120)_{10}$ to the value in the address field of fixed index word 2. The address field of index 2 contains $(3000)_{10}$. The AIX instruction is in location 2500.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION / 17 LEVEL 18 19 20 SYNC 22 USE 24	OPERATION / PICTURE 25–40	41
				AIX	120, 2	

Locations Affected:

Before

2500		
AIX	2	$(0120)_{10}$

2	
xxx	$(3000)_{10}$

After

2500		
AIX	2	$(0120)_{10}$

2	
xxx	$(3120)_{10}$

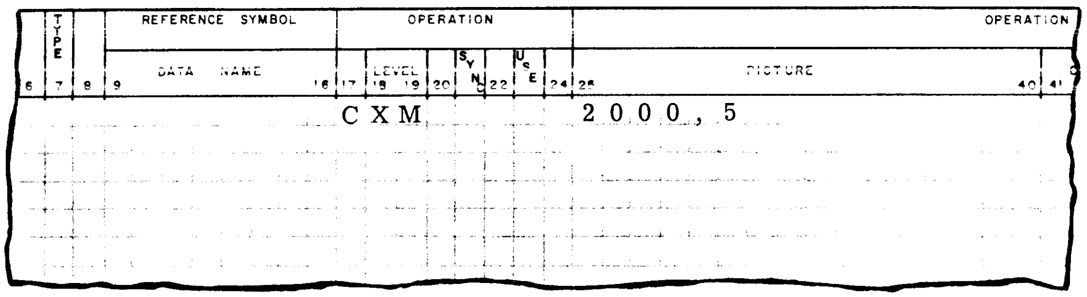
COMPARE INDEX TO MEMORY

CXM	Y, X	04XYYYYY

The contents of location Y and the contents of fixed index word X (1-6), are compared. Comparison is based upon the absolute binary contents of the two fields, although the fields can be either binary or alphanumeric. If the fields are considered to be alphanumeric, the result of the comparison will be in accord with the GE-400 collating sequence. The result of the comparison (fixed index word less, equal, or greater) is placed in the Comparison indicators. The contents of the compared locations are not changed. Y cannot be modified. This instruction is a special form of the CMM instruction and has the same operation code.

Example:

Compare the contents of fixed index word 5 absolutely with the contents of location 2000. Fixed index word 5 contains $(00155000)_8$ and location 2000 contains $(40000000)_8$.



TYPE	REFERENCE SYMBOL	OPERATION				OPERATION
	DATA NAME	LEVEL	SYN	USE		PICTURE
		CXM				2000, 5

Locations Affected:

Before	After
5	5
$(00155000)_8$	$(00155000)_8$
2000	2000
$(40000000)_8$	$(40000000)_8$

The Comparison indicators are set to "less".

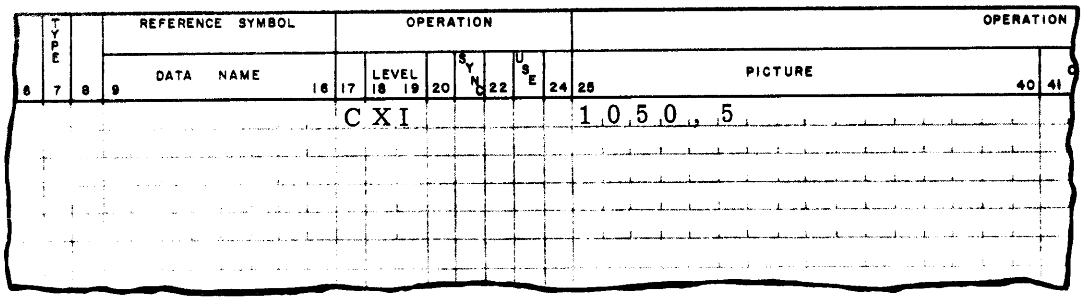
COMPARE INDEX TO IMMEDIATE

CXI	Y, X	01XYYYYY

Address field Y and the address field of fixed index word X (1-6), are compared absolutely. The result of the comparison (fixed index word address field less, equal, or greater) is placed in the Comparison indicators. The instruction word and the contents of the fixed index word are not changed. Y cannot be modified. This instruction is a special form of the CMI instruction and has the same operation code.

Example:

Compare the address field of fixed index word 5 absolutely with the address field of the CXI instruction word. The address field of the fixed index word contains $(3050)_{10}$ and the address field of the CXI instruction word contains $(1050)_{10}$. The CXI instruction is in location 2000.



TYPE	REFERENCE SYMBOL / DATA NAME (9–16)	OPERATION / LEVEL (17–19)	20	SYN (22)	USE (24)	OPERATION / PICTURE (25–40)	41
		CXI				1050, 5	

Locations Affected:

Before

5

x x x	$(3050)_{10}$

2000

CXI	5	$(1050)_{10}$

After

5

x x x	$(3050)_{10}$

2000

CXI	5	$(1050)_{10}$

The Comparison indicators are set to "greater".

PROGRAM COUNTER TO INDEX AND BRANCH

PXB	Y, X	17XYYYYY

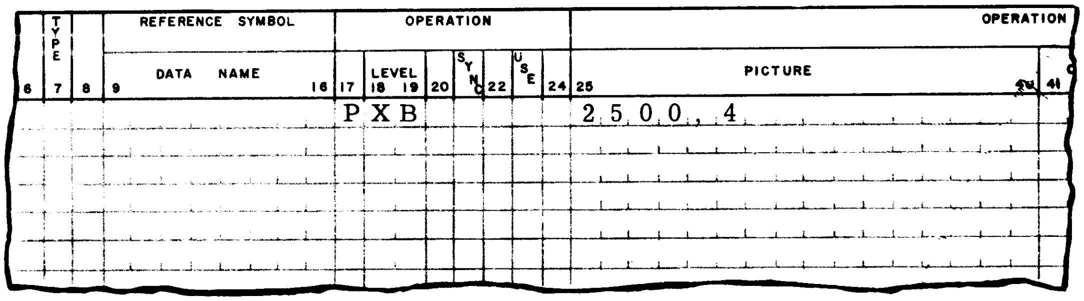
The address contained in the program counter (the address of this instruction plus one) is stored in fixed index word X (1-6), and the contents of the instruction address field, Y, are placed in the program counter, effecting a branch to the location thus specified. Y (the branch address) cannot be modified.

The instruction word and the operation code field and ACF of the fixed index word are not changed.

This instruction is a special form of the SPB instruction and has the same operation code.

Example:

Store the contents of the P-counter in the address field of fixed index word 4 and branch to location 2500. The PXB instruction is in location 2000.



6	TYPE 7	8	REFERENCE SYMBOL DATA NAME 9–16	OPERATION LEVEL 17 18 19	20	SYNC 22	USE 24	OPERATION PICTURE 25	41
				PXB				2500, 4	

Locations Affected:

Before

P-counter

(2001)$_{10}$

4

xxx	xxxxx

After

P-counter

(2500)$_{10}$

4

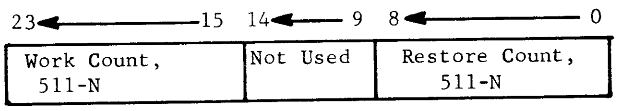
xxx	(2001)$_{10}$

BRANCH ON INDEX COUNT

BXC Y, X 16XYYYYY

This instruction operates as a branch for N executions. During the succeeding execution, the branch is not taken, but the instruction is automatically initialized so that, for the next N executions, it again operates as a branch.

The branch address is contained in the instruction address field Y and cannot be modified. The count N is part of a control word held in fixed index word X (1-6). The control word format is:



The programmer must provide the control word, with the indicated binary counts in the form 511-N, where N is the number of times that a branch is to be taken. The control word can be generated by using the BCTR pseudo-operation as described under the BRC instruction.

The work count is incremented each time the instruction is executed. As long as the count remains non-zero, the instruction operates as a branch. On the execution following the Nth execution, when the work count becomes zero:

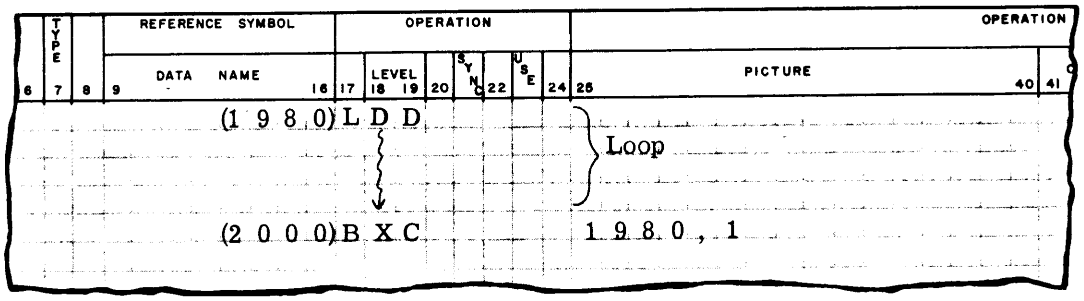
1. The First Time indicator is turned on.
2. The restore count is set into the work count field.
3. The program counter steps to the next sequential location (no branch occurs).

The restore count is not changed. A maximum of 511 branches can be specified by N.

This instruction is a special form of the BRC instruction and has the same operation code.

Example:

Execute the 20-word loop which begins at location 1980 three times before continuing to the next sequential routine. The Branch on Index Count instruction is in location 2000 and the control word is in fixed index word 1. Since the BXC is after the loop, the branch should be taken twice. The work count and restore count, therefore, are 511 minus 2, which is 509. (Although absolute locations in actual practice are not placed in the Reference Symbol field, they are used in the example below for added clarity.)



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION / LEVEL 17–20	SYNC 21–22	USE 23–24	OPERATION / PICTURE 25–40	41
			(1 9 8 0)	L D D			Loop	
			(2 0 0 0)	B X C			1 9 8 0 , 1	

Locations Affected:

P-Counter

Before execution of the loop:

$(1980)_{10}$

After first BXC:

$(1980)_{10}$

After second BXC:

$(1980)_{10}$

After third BXC:

$(2001)_{10}$

Control Word

$(509)_{10}$		$(509)_{10}$

1

$(510)_{10}$		$(509)_{10}$

1

$(511)_{10}$		$(509)_{10}$

1

$(509)_{10}$		$(509)_{10}$

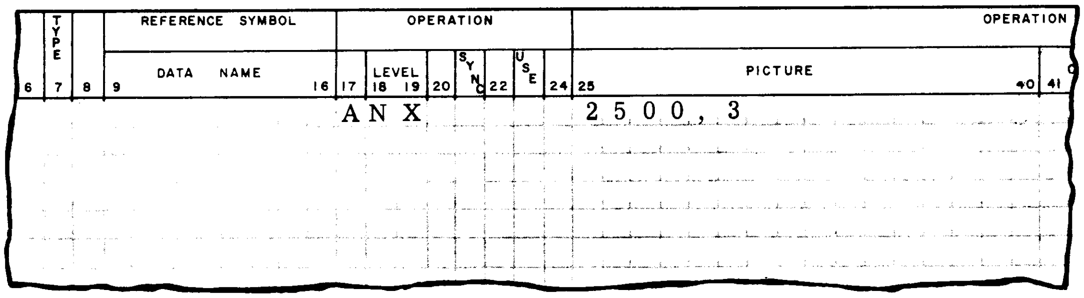
AND TO INDEX

ANX Y, X 24XYYYYY

The contents of location Y are used to modify the contents of fixed index word X (1-6). The result is placed in the fixed index word leaving the contents of location Y unchanged. The modification proceeds as follows: for each bit position of location Y that contains a zero, the corresponding bit position of the fixed index word is set to zero; for each bit position in location Y that contains a one, the corresponding bit position of the fixed index word is left unchanged. Location Y cannot be modified. This instruction is a special form of the ANM instruction and has the same operation code.

Example:

AND the contents of location 2500 to the contents of fixed index word 3. Location 2500 contains $(52702114)_8$ and fixed index word 3 contains $(34362670)_8$.



REFERENCE SYMBOL	OPERATION	OPERATION
	ANX	2500, 3

Locations Affected:

Before

2500

101 010 111 000 010 001 001 100

3

011 100 011 110 010 110 111 000

After

2500

101 010 111 000 010 001 001 100

3

001 000 011 000 010 000 001 000

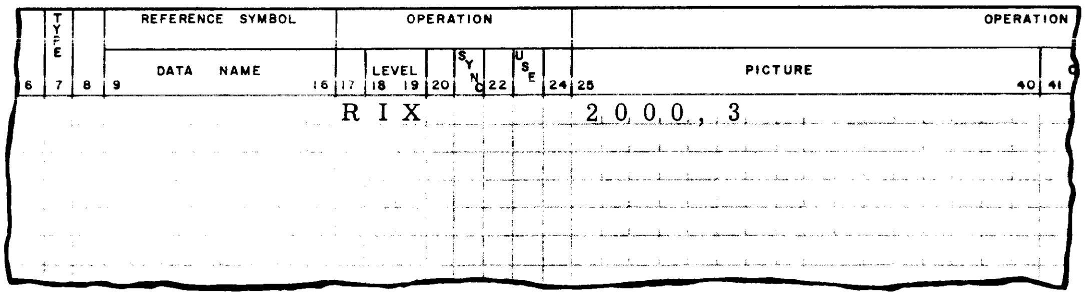
OR INCLUSIVE TO INDEX

RIX	Y, X	23XYYYYY

The contents of location Y are used to modify the contents of fixed index word X (1-6). The result is placed in the fixed index word leaving the contents of location Y undisturbed. The modification proceeds as follows: for each bit position of location Y that contains a one, the corresponding bit position of the fixed index word is set to one; for each bit position of location Y that contains a zero, the corresponding bit position of the fixed index word is left undisturbed. Location Y cannot be modified. This instruction is a special form of the RIM instruction and has the same operation code.

Example:

Modify the contents of fixed index word 3 with the contents of location 2000 by the OR inclusive method. Location 2000 contains $(61670251)_8$ and fixed index word 3 contains $(16257036)_8$.



TYPE	Reference Symbol / Data Name	Operation	Operation / Picture
		RIX	2000, 3

Locations Affected:

Before

2000

110 001 110 111 000 010 101 001

3

001 110 010 101 111 000 011 110

After

2000

110 001 110 111 000 010 101 001

3

111 111 110 111 111 010 111 111

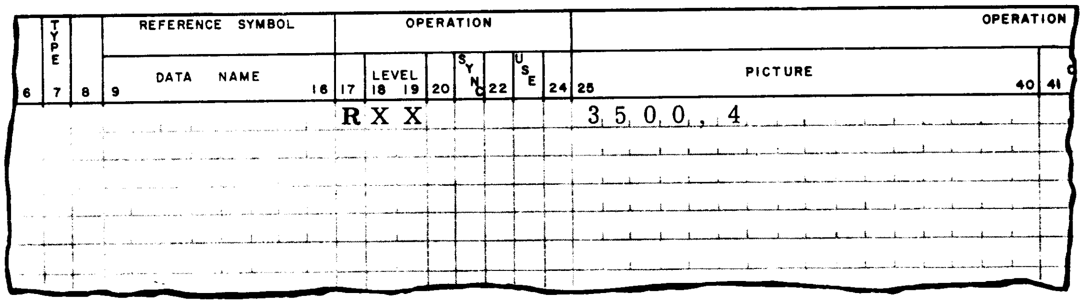
OR EXCLUSIVE TO INDEX

RXX	Y, X	25XYYYYY

The contents of location Y are used to modify the contents of fixed index word X (1-6). The result is placed in the fixed index word, leaving the contents of location Y unchanged. The modification proceeds as follows: for each bit position of location Y that contains a one, the corresponding bit position of the fixed index word is inverted (zero becomes one, one becomes zero); for each bit position of location Y that contains zero, the corresponding bit position of the fixed index word is left undisturbed. Location Y cannot be modified. This instruction is a special form of the RXM instruction and has the same operation code.

Example:

Modify the contents of fixed index word 4 with the contents of location 3500 by the OR exclusive method. Location 3500 contains $(07256152)_8$ and fixed index word 4 contains $(25416073)_8$.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION / LEVEL 17 18 19	20	SYNC 22	USE 24	OPERATION / PICTURE 25–40	41
				RXX				3500,4	

Locations Affected:

Before

3500

000 111 010 101 110 001 101 010

4

010 101 100 001 110 000 111 011

After

3500

000 111 010 101 110 001 101 010

4

010 010 110 100 000 001 010 001

ACCUMULATOR INSTRUCTIONS

LOAD ACCUMULATOR LOCATION AND LENGTH

LAL	Y/Y, X	36XYYYYY

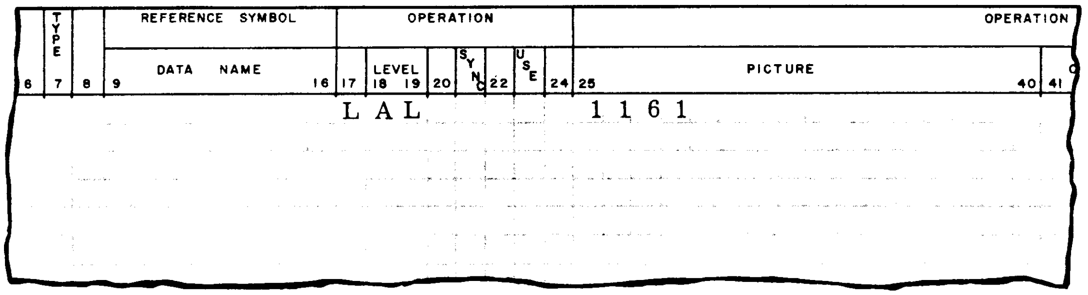
The accumulator location and length are established as specified by the instruction address field Y. The most-significant 13 bits of address field Y, together with two assumed least-significant zeros, define the location of the most-significant word of the four-word accumulator. Therefore, the most-significant word of the four-word accumulator is always at a 0, modulo 4 location. Bits one and zero of address field Y establish the length of the accumulator as follows:

00	Quadruple
01	Triple
10	Double
11	Single

Address field Y can also be interpreted as specifying the location of the most-significant word of the working accumulator. In relocating the accumulator, the contents of both the old and the new locations are undisturbed. Care should be exercised in positioning the accumulator in locations zero through 71 (excluding 12 through 15), because these locations are reserved for specific purposes and for programming compatibility with other members of the GE-400 family.

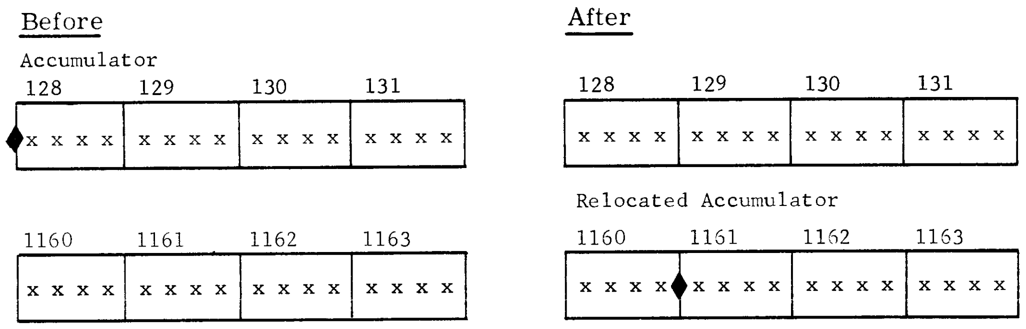
Example:

Relocate the four-word accumulator to locations 1160 through 1163 and set the working length to triple. The accumulator is presently located at 128 through 131 and the working length is quadruple.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 22	USE 24	OPERATION / PICTURE 25–40	41
				LAL					1161	

Locations Affected:


Before
Accumulator
128 129 130 131
x x x x x x x x x x x x x x x x
1160 1161 1162 1163
x x x x x x x x x x x x x x x x
After
128 129 130 131
x x x x x x x x x x x x x x x x
Relocated Accumulator
1160 1161 1162 1163
x x x x x x x x x x x x x x x x

STORE ACCUMULATOR LOCATION AND LENGTH

SAL	Y/Y, X	37XYYYYY

The current accumulator location and length are stored in the address field of location Y. The operation code field and ACF of location Y are not changed. The most-significant 13 bits stored in the address field of location Y, together with two assumed least-significant zeros, indicate the location of the most-significant word of the four-word accumulator. Bits one and zero stored in the address field of location Y indicate the working length of the accumulator as follows:

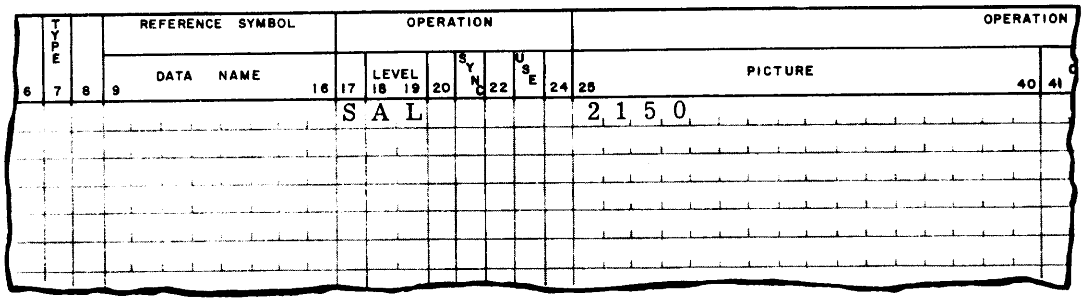
00	Quadruple
01	Triple
10	Double
11	Single

The 15-bit field stored in location Y canalso be interpreted as the location of the most-significant word of the working accumulator.

The actual accumulator location, contents, and length are not changed.

Example:

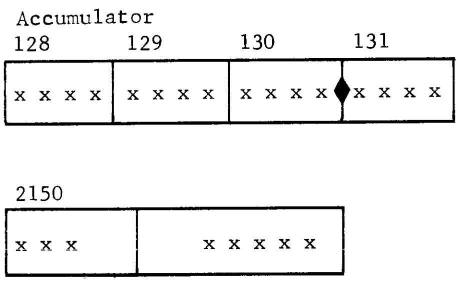
Store the accumulator location and length in the address field of location 2150. The accumulator is presently located at 128 through 131 and the working length is single.



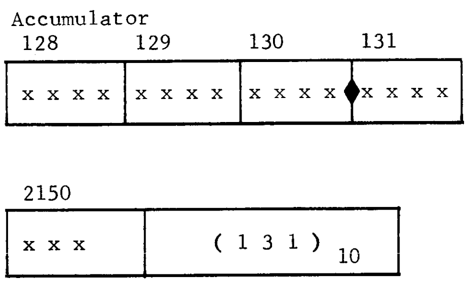
			REFERENCE SYMBOL	OPERATION							OPERATION	
	TYPE		DATA NAME		LEVEL		SYNC		USE		PICTURE	
6	7	8	9 16	17	18 19	20		22		24	25 40	41
				S	A L						2 1 5 0	

Locations Affected:

Before



After



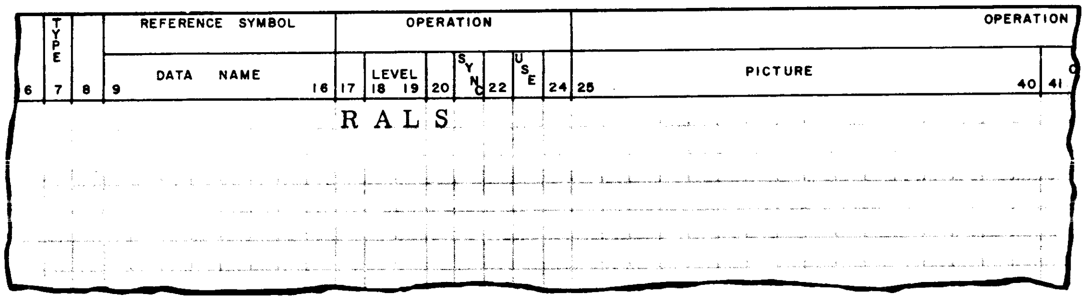
RESET ACCUMULATOR LENGTH SINGLE

RALS 22X07000

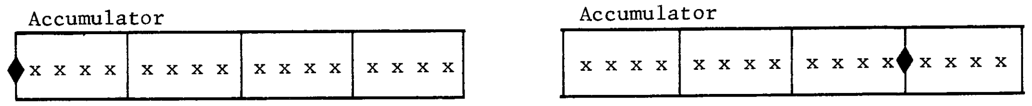
The accumulator working length is set to single. The accumulator contents and location are not affected. RALS is a special case of the Shift instruction (octal 22) and any modification of the RALS address field has the same effects as modification of a Shift instruction address field.

Example:

Change the accumulator length to single. The current length is quadruple.



Locations Affected:



Before
Accumulator
x x x x x x x x x x x x x x x x
After
Accumulator
x x x x x x x x x x x x x x x x

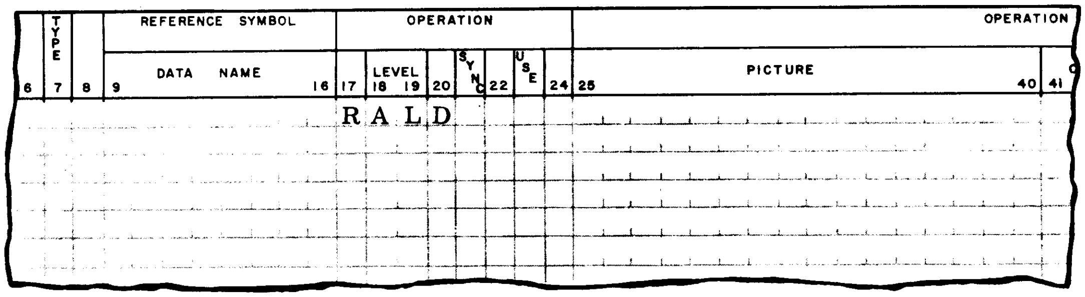
RESET ACCUMULATOR LENGTH DOUBLE

RALD 22X06000

The accumulator working length is set to double. The accumulator contents and location are not affected. RALD is a special case of the Shift instruction (octal 22) and any modification of the RALD address field has the same effects as modification of a Shift instruction address field.

Example:

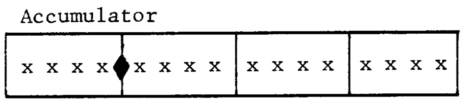
Change the accumulator length to double. The current length is triple.


TYPE
REFERENCE SYMBOL
OPERATION
OPERATION
DATA NAME
LEVEL
SYNC
USE
PICTURE
6 7 8 9 16 17 18 19 20 22 24 25 40 41

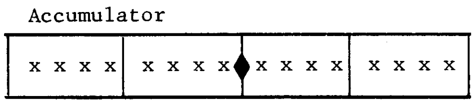
RALD

Locations Affected:

Before



After



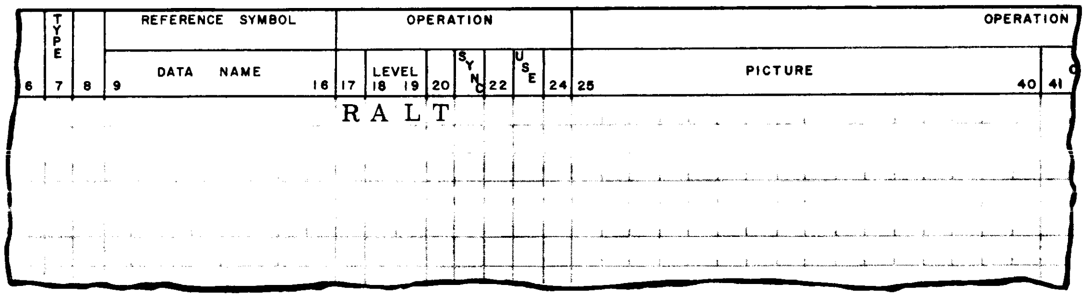
RESET ACCUMULATOR LENGTH TRIPLE

RALT 22X05000

The accumulator working length is set to triple. The accumulator contents and location are not affected. RALT is a special cast of the Shift instruction (octal 22) and any modification of the RALT address field has the same effects as modification of a Shift instruction address field.

Example:

Change the accumulator length to triple. The current length is single.



	TYPE		REFERENCE SYMBOL	OPERATION							OPERATION	
			DATA NAME		LEVEL		SYNC		USE		PICTURE	
6	7	8	9 16	17	18 19	20		22		24	25 40	41
				RALT								

Locations Affected:

Before

Accumulator

x x x x	x x x x	x x x x◆	x x x x

After

Accumulator

x x x x◆	x x x x	x x x x	x x x x

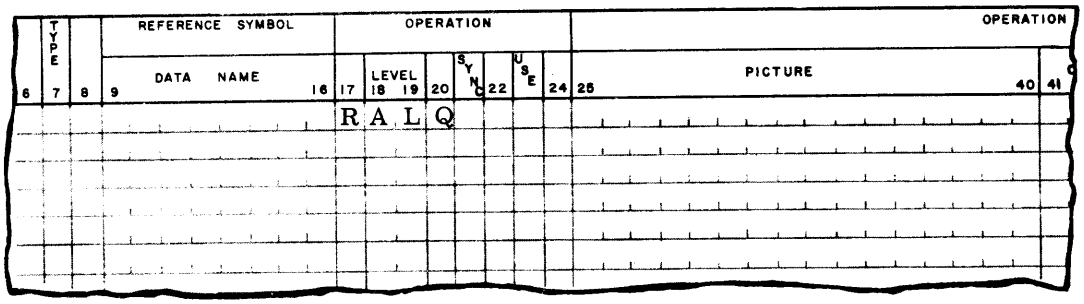
RESET ACCUMULATOR LENGTH QUADRUPLE

RALQ 22X04000

The accumulator working length is set to quadruple. The accumulator contents and location are not affected. RALQ is a special case of the Shift instruction (octal 22) and any modification of the RALQ address field has the same effects as modification of a Shift instruction address field.

Example:

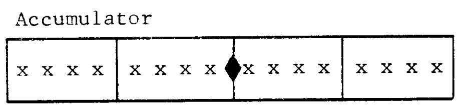
Change the accumulator length to quadruple. The current length is double.



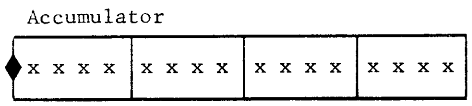
6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18	19	20	SYNC 22	USE 24	OPERATION / PICTURE 25–40	41
				R	A	L	Q				

Locations Affected:

Before



After



SPECIAL PURPOSE INSTRUCTIONS

The Edit Instruction

EDIT

EDT	Y/Y, X	05XYYYYY

The contents of a memory field whose least-significant word is at location Y are processed from right to left under control of 4, 8, 12, or 16 format characters in the working accumulator. For each format character, a single character is placed or retained in the working accumulator. The number of result characters equals the number of format characters. The memory field and the accumulator length are not changed.

Certain format characters can mark the corresponding and subsequent result characters as tentative candidates for suppression. Three types of suppression can occur under format character control:

1. Conversion of non-significant zeros, commas, and periods to blank spaces (Simple Suppress).
2. Conversion of non-significant zeros, commas, and periods to asterisks (Suppress and Asterisk Protect).
3. Same as 1, above, except that the rightmost non-significant zero, comma, or period is converted to a dollar sign (Suppress and Float $).

Each type of suppression necessitates a two-phase operation. Phase 1 produces a tentative result; Phase 2 performs the specified type of suppression. If there is no suppression character in the format, only Phase 1 is necessary, and the tentative result automatically becomes the final result.

In Phase 1, editing proceeds from right to left, with the rightmost format character and the rightmost data character being considered first. In each case, the result is determined by one of seven format codes:

1. Use - The semicolon (; octal 56) specifies that the data character is to be <u>used</u> in place of the format character in the tentative result.
2. Simple Suppress - The tilde (~ octal 12) specifies that the data character is to be used in place of the format character in the tentative result, and that <u>Simple Suppression</u> is to occur, starting with this tentative result character.
3. Suppress and Float $ - The pound sign (# octal 13) specifies that the data character is to be used in place of the format character in the tentative result, and that <u>Suppress and Float $</u> is to occur, starting with this tentative result character.

4. Suppress and Asterisk Protect - The "greater than" symbol (> octal 16) specifies that the data character is to be used in place of the format character in the tentative result, and that Suppress and Asterisk Protect is to occur, starting with this tentative result character.

5. Ignore - The Ignore symbol (1̸, octal 17) specifies that the "Ignore" format character, and not the data character, is to be retained in the tentative result.

6. Sign - The dash (- octal 52) specifies that

   a. If none of the above format characters (;,~ , #,>, or 1̸) has occurred previously in the format (that is, if the least-significant data character is still being considered), the sign information contained in the data character zone bits determines the character (space or dash) to be placed in the tentative result. After the sign information has been extracted, only the numeric bits of that data character are retained for subsequent consideration with the next format character.

   b. If one or more of the above format characters have previously occurred in the format, the dash is inserted in the tentative result. The current data character is retained for consideration with the next format character.

7. Insert - Any of the 64 characters of the character set that are not defined above but are included in the format are inserted into the tentative result. The data character is retained for consideration with the next format character.

Data characters are processed under control of format characters until all format characters have been examined. The table, Editing-Phase 1, shows how each format character affects the tentative result.

If a format suppression code (~, #, or >) is detected during the first phase, a suppression mode is established and a suppression counter counts as indicated in Figure III-6. The suppression counter is initially set to zero.

In the suppression mode, every character placed in the tentative result affects the suppression counter:

The suppression counter is reset to zero if the format character is ~, #, >, or ; and the data character is not a zero.

One is added to the suppression counter,

a. If the format character is ~, #, > , or ; and the data character is a zero.

b. If the format character is not ~, #, > , or ;.

If more than one type of suppression code occurs in the format, the suppression begins in the character position of the first (rightmost) suppression code, and the type of suppression performed is determined by the last (leftmost) suppression code.

Phase 2 is entered only if a suppression code is detected during the Phase 1. At the beginning of the Phase 2, the suppression counter indicates the number of characters in the tentative result, counting from the left, which are candidates for suppression. These characters are examined, and any zeros, commas, or periods encountered are suppressed as indicated in Figure III-7. No other characters in the tentative result are affected.

CONDITIONS								EFFECTS				
Type of Format	Format Code		Is First Data Character Still Being Considered?	Sign of Data	Suppression Established Previously ?	D = 0 ?	Tentative Result	Advance to Next Character?		Effect on Suppression Counter	Other Actions	Note
	Char.	Octal						Format	Data			
Simple Suppress	~	12				No	D	Yes	Yes	Clear to 0	Set simple suppress mode	III
		12				Yes	D	Yes	Yes	Add 1	Set simple suppress mode	III
Float $	#	13				No	D	Yes	Yes	Clear to 0	Set float dollar sign suppress mode	III
		13				Yes	D	Yes	Yes	Add 1	Set float dollar sign suppress mode	III
Asterisk Protect	>	16				No	D	Yes	Yes	Clear to 0	Set asterisk protect suppress mode	III
		16				Yes	D	Yes	Yes	Add 1	Set asterisk protect suppress mode	III
Ignore	Ɨ	17			No		F	Yes	Yes			II
		17			Yes		F	Yes	Yes	Add 1		II
Sign	—	52	No		No		F	Yes	No			I
		52	No		Yes		F	Yes	No	Add 1		I
		52	Yes	Plus			Space	Yes	No		Set zone bits of D to 00 in processor	IV
		52	Yes	Minus			F	Yes	No		Set zone bits of D to 00 in processor	I
Use	;	56			No		D	Yes	Yes			III
		56			Yes	No	D	Yes	Yes	Clear to 0		III
		56			Yes	Yes	D	Yes	Yes	Add 1		III
Insert		Other			No		F	Yes	No			I
		Other			Yes		F	Yes	No	Add 1		I

F - Format character under consideration

D - Data character under consideration

NOTE

I. The format character is retained in the tentative result and the data character is retained to be processed by the next format character.

II. The format character is retained in the tentative result and the data character is discarded, so that the next format character will process the next data character.

III. The format character is discarded and the data character is placed in the tentative result so that the next format character processes the next data character.

IV. A blank space is placed in the tentative result. The format character is discarded and the data character is retained to be processed by the next format character.

Figure III-6. Phase 1 - Editing Table

Tentative Result Character	Type of Suppression	Final Result Character	Other Actions
Zero, period, or comma	Simple	Space	
Zero, period, or comma	Float $	$	Change to simple suppression
Zero, period, or comma	Asterisk	*	
Other		No Change	

Examination of tentative result characters which are candidates for suppression.

Figure III-7. Phase 2 - Editing Table (Suppression)

Examples:

1. Use, octal 56 (;)

Memory	4 3 2 1	Data
Accumulator	; ; ; ;	Format
Accumulator	4 3 2 1	Result

2. Insert all characters, except special format codes.

Memory	0 0 1 2	5 3 6 7	Data
Accumulator	; , ; ;	; . ; ;	Format
Accumulator	1 , 2 5	3 . 6 7	Result

3. Ignore, octal 17 (1)

Memory	7 3 2 4	Data
Accumulator	; ; ; 1	Format
Accumulator	7 3 2 1	Result

4. <u>Sign</u>, octal 52 (-)

Memory	0 3 2 $\overset{+}{1}$	Data
Accumulator	; ; ; -	Format
Accumulator	3 2 1 Δ	Result
Memory	8 5 3 $\bar{1}$	Data
Accumulator	; ; ; -	Format
Accumulator	5 3 1 -	Result
Memory	5 8 3 $\bar{1}$	Data
Accumulator	; - ; -	Format
Accumulator	3 - 1 -	Result
Memory	9 3 1 $\bar{4}$	Data
Accumulator	; - - -	Format
Accumulator	4 Δ Δ -	Result

5. <u>Suppression:</u> Review of general rules:

Pass 1:

1. Any of the three suppression codes causes a counter to function.
2. Counter adds 1 for each zero data character moved to the result.
3. Counter adds 1 for each format character moved to the result.
4. Counter resets to 0 when non-zero data character moves to result. Counting continues.

Pass 2:

1. The count identifies the number of leftmost positions that are candidates for suppression.
2. Zero, comma, or period candidates are replaced by some other character.
3. The leftmost suppression code determines the type of suppression. The rightmost suppression code determines where suppression begins.

Memory	0 0 0 0	Data
Accumulator	; . ~ ;	Format
Accumulator	0 . 0 0	Result, Pass 1
Counter	3 2 1 0	
Accumulator	△△△ 0	Result, Final

Memory	0 0 1 0	0 2 0 0	Data
Accumulator	; ; , ;	. ; ~ ;	Format
Accumulator	1 0 , 0	. 2 0 0	Result, Pass 1
Counter	0 4 3 2	1 0 1 0	
Accumulator	Same as Pass 1 Result. There are no suppression candidates.		

6. Suppress and Use, octal 12 (~)
All zero, comma, and period candidates are replaced by spaces.

Memory	0 0 0 1	2 3 4 5	Data
Accumulator	; ; ; ;	; ; ; ~	Format
Accumulator	△△△ 1	2 3 4 5	Result

7. Suppress and Float $, octal 13 (#)
A dollar sign replaces the rightmost zero, comma, or period candidate. All other zero, comma, and period candidates are replaced by spaces.

Memory	0 0 0 0	0 1 9 8	Data
Accumulator	; , ; ;	# . ; ;	Format
Accumulator	△△△ $	1 . 9 8	Result

8. Suppress and * Protect, octal 16 (>)
All zero, comma, and period candidates are replaced by asterisks.

Memory	0 0 0 0	0 0 9 8	Data
Accumulator	; ; ; ;	>. ; ;	Format
Accumulator	* * * *	* . 9 8	Result

Central Processor Operations

The central processor operation instructions (SSA, SSO, SSL, and RQSP) permit the status of specific indicators to be stored in memory or permit status to be set into these indicators. The effective status word bits and the corresponding indicators are:

Bits	Indicators
1 and 0	Comparison 00 - Equal 01 - Less 10 - Greater 11 - Less and Greater
2	Program Interrupt
3	First Time
4	Overflow
5	Overflow Mode
6 - 18	(Must be zero)
19	Request Control
20	Instruction Alert
21	Control Word Alert
22, 23	(Must be zero)

For more information about these CPO indicators, see "Section V, Processor Channel."

REQUEST STATUS OF PROCESSOR

```
RQSP                                          67X00000
 OP/OL            Z                           0C0ZZZZZ
```

The status of specific central processor indicators is stored into a second location. A 0-bit is stored when the corresponding indicator is off; a 1-bit is stored when the corresponding indicator is on. However, a 0-bit is always stored in the Program Interrupt bit position (bit 2) and in bit positions 6 through 18, 22, and 23.

Execution of the instruction also turns off the Request Control, Instruction Alert, Control Word Alert, and Overflow indicators. In addition, it turns off the Overflow Mode indicator if the Program Interrupt indicator is on. It does not alter the setting of the other indicators.

Bits 2 through 14 of the address field must be zeros to preserve GE-400 programming compatibility. Modification of the instruction word address field is possible; however, any modification that results in the two least-significant address field bits being other than 00 changes the operation.

If the ACF contains 0 or 7, the location in which status is stored is defined by a Second Address Sequence. The SAS word can be an Operand Pointer or an Operand Link. Because of the status word format, it should not be an Operand:

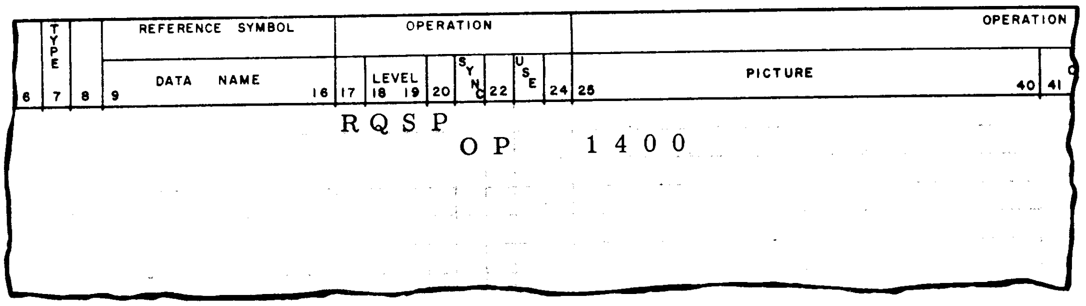
OPERAND POINTER. Location Z is the location of the status word.

OPERAND LINK. Location Z contains another SAS word which is either an Operand Pointer or another Operand Link.

If the ACF is any number from 1 through 6, the status is placed in the corresponding fixed index word, and no SAS is required.

Example:

Store the status of the central processor indicators in location 1400. In the example, the Instruction Alert, First Time, Program Interrupt and Less indicators are on.



6	7 TYPE	8	9 REFERENCE SYMBOL / DATA NAME 16	17 OPERATION / LEVEL 18 19	20	SYNC	22	USE	24	25 OPERATION / PICTURE 40	41
				RQSP							
							OP			1400	

Locations Affected:

Indicators	BEFORE	AFTER	Bit Position
Control Word Alert	0	0	21
Instruction Alert	1	0	20
Request Control	0	0	19
Overflow Mode	0	0	5
Overflow	0	0	4
First Time	1	1	3
Program Interrupt	1	1	2
GR } Compare	0	0	1
LS } Compare	1	1	0

1400 Before

23 22	21 19	18 6	5 0
x x	x x x	xxxxxxxxxxxxx	xxxxxx

1400 After

23 22	21 19	18 6	5 0
0 0	0 1 0	0000000000000	001001

Note that the Instruction Alert indicator is turned off by the RQSP, although the corresponding bit in the stored status word is on.

SET STATUS BY LOADING

SSL		67X00003
O/OP/OL	Z	0C0ZZZZZ

The contents of a stored status word are used to set the central processor indicators: for any status word bit (0 through 5) which is a 1, the corresponding indicator is turned on; for any bit (0 through 5) which is a zero, the corresponding indicator is turned off. Bits 6 through 18, 22 and 23 of the status word must be zeros to preserve GE-400 programming compatibility. Bits 19 through 21 are ignored.

When the processor is in the program interrupt mode, this instruction cannot turn the Program Interrupt indicator off. If the instruction is coded to turn the Program Interrupt indicator off, it will only condition the indicator so that the next BRU instruction executed by the processor will turn the indicator off.

When the Overflow indicator is off, this instruction coded to turn both the Overflow and the Overflow Mode indicators on results in only the Overflow Mode indicator being on.

Bits 2 through 14 of the instruction word address field must be zeros to preserve programming compatibility. Modification of the instruction word address field is possible; however, any modification that results in the two least-significant address field bits being other than 11 changes the operation.

If the ACF contains 0 or 7, the location of the status word is defined by a Second Address Sequence. The SAS word can be an Operand, an Operand Pointer, or an Operand Link:

OPERAND. The SAS word is the status word.

OPERAND POINTER. Location Z is the location of the status word.

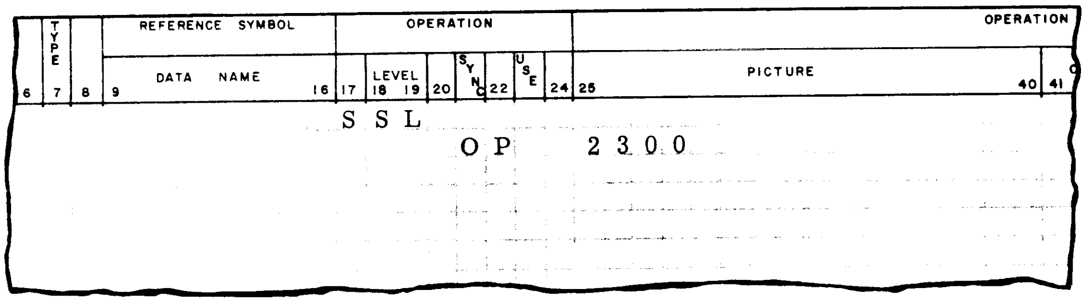
OPERAND LINK. Location Z contains another SAS word which is either an Operand Pointer or another Operand Link.

It is not advisable to code the SAS word of the SSL instruction as an Operand. If the status word was loaded into the second address as a result of an RQSP, a 1-bit could have been placed into bit position 19. This would change the class field of the SAS word (if it were an Operand) to an Operand Link.

If the ACF is any number from 1 through 6, the status word is in the corresponding fixed index word, and no SAS is required.

Example:

Load the contents of the status word stored in location 2300 into the central processor indicators in order to turn on the First Time indicator and turn off all other indicators. The status word contains $(00000010)_8$. In the example, the Overflow indicator is on.



T Y P E
REFERENCE SYMBOL
OPERATION
OPERATION
DATA NAME
LEVEL
SYNC
USE
PICTURE
6 7 8 9 16 17 18 19 20 22 24 25 40 41

```
SSL
     OP   2300
```

Locations Affected:

Before

2300

23 22	21 19	18 6	5 0
0 0	x x x	0000000000000	001000

After

2300

23 22	21 19	18 6	5 0
0 0	x x x	0000000000000	001000

Indicators	BEFORE	AFTER	Bit Position
Overflow Mode	0	0	5
Overflow	1	0	4
First Time	0	1	3
Program Interrupt	0	0	2
GR } Compare	0	0	1
LS } Compare	0	0	0

SET STATUS BY ANDING

SSA		67X00002
O/OP/OL	Z	0C0ZZZZZ

The contents of a stored status word are ANDed into the central processor indicators: for any status word bit (0 through 5) which is a one, there is no effect; for any bit (0 through 5) which is a zero, the corresponding incicator is turned off. Bits 6 through 18, 22 and 23 of the status word must be zeros to preserve GE-400 programming compatibility. Bits 19 through 21 are ignored.

When the processor is in the program interrupt mode, this instruction cannot turn the Program Interrupt indicator off. If the instruction is coded to turn the Program Interrupt indicator off, it will only condition the indicator so that the next BRU instruction executed by the processor will turn the indicator off.

Bits 2 through 14 of the instruction word address field must be zeros to preserve GE-400 programming compatibility. Modification of the instruction word address field is possible; however, any modification that results in the two least-significant address field bits being other than 10 changes the operation.

If the ACF contains 0 or 7, the location of the status word is defined by a Second Address Sequence. The SAS word can be an Operand, an Operand Pointer, or an Operand Link:

OPERAND. The SAS word is the status word.

OPERAND POINTER. Location Z is the location of the status word.

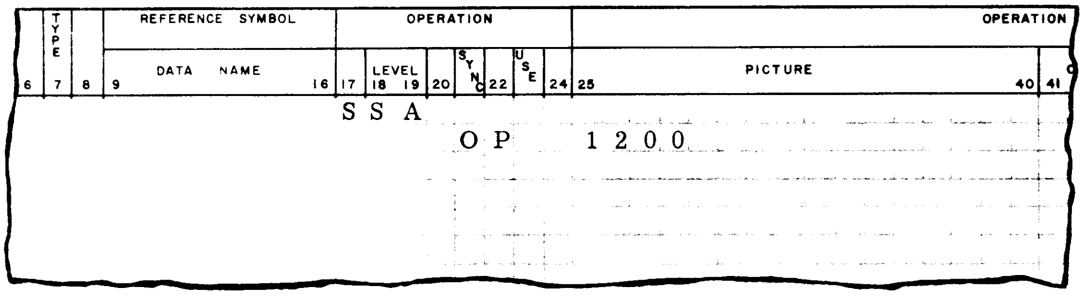
OPERAND LINK. Location Z contains another SAS word which is either an Operand Pointer or another Operand Link.

It is not advisable to code the SAS word of the SSA instruction as an Operand. If the status word was loaded into the second address as a result of an RQSP, a 1-bit could have been placed into bit position 19. This would change the class field of the SAS word (if it were an Operand) to an Operand Link.

If the ACF is any number from 1 through 6, the status word is in the corresponding fixed index word, and no SAS is required.

Example:

AND the contents of the status word stored in location 1200 into the central processor indicators in order to turn off the Overflow indicator without affecting the other indicators. The status word contains $(00000057)_8$. In the example, the Overflow, Program Interrupt, and Less indicators are on.



TYPE	REFERENCE SYMBOL / DATA NAME (9–16)	OPERATION (17–24)	OPERATION / PICTURE (25–40)
		SSA	
		OP	1200

Locations Affected:

Before

1200

23 22	21 19	18 6	5 0
0 0	x x x	0000000000000	101111

After

1200

23 22	21 19	18 6	5 0
0 0	x x x	0000000000000	101111

Indicators	BEFORE	AFTER	Bit Position
Overflow Mode	0	0	5
Overflow	1	0	4
First Time	0	0	3
Program Interrupt	1	1	2
GR } Compare	0	0	1
LS } Compare	1	1	0

SET STATUS BY ORING

```
SSO                                                  67X00001
 O/OP/OL            Z                                0C0ZZZZZ
```

The contents of a stored status word are ORed into the central processor indicators: for any status word bit (0 through 5) which is a one the corresponding status condition is turned on; for any bit (0 through 5) which is a zero, there is no effect. Bits 6 through 18, 22 and 23 of the status word must be zeros to preserve GE-400 programming compatibility. Bits 19 through 21 are ignored.

When the Overflow indicator is off, this instruction coded to:

1. Turn on both the Overflow and Overflow Mode indicators or
2. Turn on the Overflow indicator when the Overflow Mode indicator is already on

results in only the Overflow Mode indicator being on.

Bits 2 through 14 of the instruction word address field must be zeros to preserve GE-400 programming compatibility. Modification of the instruction word address field is possible; however, any modification that results in the two least-significant address field bits being other than 01 changes the operation.

If the ACF contains 0 or 7, the location of the status word is defined by a Second Address Sequence. The SAS word can be an Operand, an Operand Pointer, or an Operand Link:

OPERAND. The SAS word is the status word.

OPERAND POINTER. Location Z is the location of the status word.

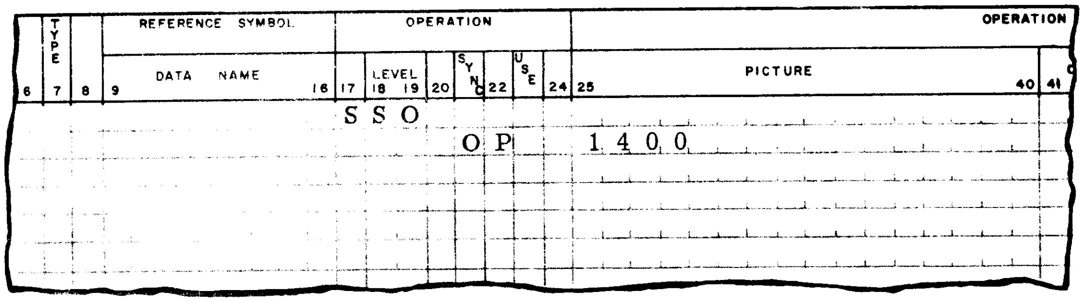
OPERAND LINK. Location Z contains another SAS word which is either an Operand Pointer or another Operand Link.

It is not advisable to code the SAS word of the SSO instruction as an Operand. If the status word was loaded into the second address as a result of an RQSP, a 1-bit could have been placed into bit position 19. This would change the class field of the SAS word (if it were an Operand) to an Operand Link.

If the ACF is any number from 1 through 6, the status word is in the corresponding fixed index word, and no SAS is required.

Example:

OR the contents of the status word stored in location 1400 into the central processor indicators in order to turn on the First Time indicator without affecting the other indicators. The status word contains $(00000010)_8$. In the example, the Overflow, Program Interrupt, and Less indicators are on.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYNC 21 22	USE 23 24	OPERATION / PICTURE 25–40	41
				S	S O					
							O P		1 4 0 0	

Locations Affected:

Before

1400

23 22	21 19	18 6	5 0
0 0	x x x	0000000000000	001000

After

1400

23 22	21 19	18 6	5 0
0 0	x x x	0000000000000	001000

Indicators	BEFORE	AFTER	Bit Position
Overflow Mode	0	0	5
Overflow	1	1	4
First Time	0	1	3
Program Interrupt	1	1	2
GR } Compare	0	0	1
LS } Compare	1	1	0

The Halt Instruction

HALT		
HLT	Y/Y, X	00XYYYYY

This instruction causes the computer to cease processing instructions. The MANUAL light on the console is turned on; and the Halt instruction word, after any address field modification, can be displayed on the console typewriter.

The Halt instruction establishes control conditions which inhibit the honoring of requests for program interrupts, but which allow data transfer sequences to occur if any peripherals have not terminated operations. A Halt should be issued only when there are no peripherals busy that require program interrupts for successful completion of their operations. The HALT routine in the Basic Input Output System brings all input/output operations to an orderly conclusion before halting the computer. Actuation of the RUN switch on the console causes continuous instruction processing to be resumed, starting with the next instruction in sequence.

The General Instructions

GENERAL

GEN	Y Y. X	07XYYYYY
OP OL	Z	0C0ZZZZZ

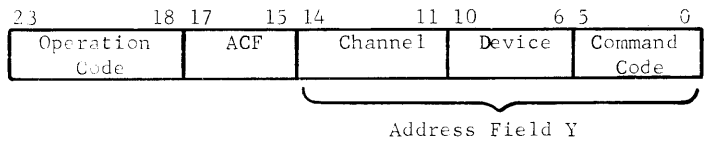
Depending upon the contents of address field Y, one of the following occurs:

1. A peripheral input output operation or a control operation is initiated (assuming the peripheral is in ready status), resettable status is reset, and the remaining peripheral status is stored in a second location.

2. Any existing resettable status in the addressed peripheral is reset and the remaining peripheral status is stored in a second location.

3. The existing peripheral status is stored in a second location.

Address field Y determines the type of operation as follows:



The operation code and ACF follow the standard instruction format and interpretation.

The peripheral channel (0 through 7) is specified in bits 14 through 11. The specific peripheral device (for multi-device subsystems) is identified in bits 10 through 6.

The command code in bits 5 through 0 identifies the operation as either a specific peripheral equipment operation, a request peripheral status operation, or a reset peripheral status operation.

Address field Y can be modified under control of the instruction word ACF (X); however, the programmer must keep in mind that such modification will affect the channel, device, and/or command code of the instruction.

Channel/peripheral status is always stored in the second location, regardless of whether the instruction is an input output operation, a request status operation, or a reset status operation.

If the ACF (X) is either zero (nohmodification) or seven (modification), the status word location is defined by a Second Address Sequence. The SAS word can be an Operand Pointer or an Operand Link. Because of the status word format, the SAS word should not be an Operand:

OPERAND POINTER. Location Z is the location of the status word.

OPERAND LINK. Location Z contains another SAS word which is either an Operand Pointer or another Operand Link.

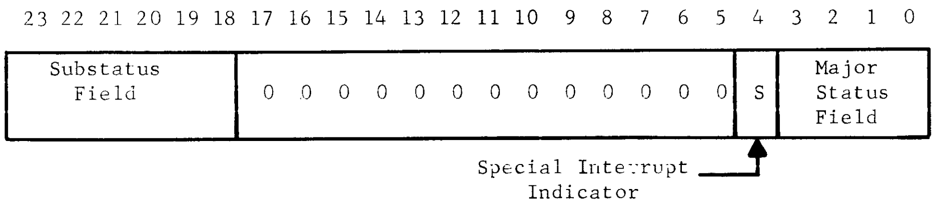
If the ACF specifies a fixed index word (1-6), no SAS word is required and the specified index word is the second location.

In addition to the GEN mnemonic code, the assembly language permits the use of unique mnemonic codes for all possible input/output operations, as well as for the request status and reset status commands. The formats for these commands follow.

REQUEST STATUS

RQS	C. D. X	07XYYY00
OP/OL	Z	0C0ZZZZZ

The existing status in the addressed peripheral subsystem (channel C, device D) is placed in a second location in the following format:



Address modification and determination of the second location for status word storage occurs as described under the General instruction heading.

RESET STATUS

RSS	C, D, X	07XYYY40
OP/OL	Z	0C0ZZZZZ

If any one of the four major status conditions (Data Alert, End of File, Command Rejected, or Load Operation Complete) exists in the addressed peripheral subsystem (channel C, device D), it is reset. The remaining status is placed in the second location.

Address modification and determination of the second location for status word storage occurs as described under the General instruction heading.

PERIPHERAL INSTRUCTIONS

All peripheral operations are identified by the command code field of the General instruction that initiates the operation. The command codes for valid peripheral operations are shown in this section, together with the mnemonic codes as provided by the basic assembly language for the GE-425. The description of each peripheral instruction is provided for general information only. For detailed information on programming peripheral operations, refer to the GE-425 Basic Input Output System and Extended Input Output System manuals and to the appropriate peripheral subsystem manuals.

Note that several command codes are used more than once, and will have different effects upon execution, depending upon the peripheral receiving the General instruction. For example, a General instruction containing the command code 01, when directed to the card reader, initiates a Read Card Binary (RCB) operation; when directed to the input/output typewriter, readies the typewriter for a type-in; and when directed to the magnetic reader/sorter, initiates the reading of a document. Command codes received by peripheral subsystems that do not recognize the code as valid will result in the command code being rejected and reflection of the Command Rejected major status.

The Macro Assembly Program automatically produces a General instruction with the appropriate command code from the mnemonic codes listed for the peripheral operations. The programmer must specify, either in actual or symbolic notation, the channel and device codes and the second location to which the status word is stored. Alternatively, he can utilize the call sequences for input/output routines, as specified in the Basic and Extended Input Output Systems, and save significant time in programming.

In addition to the instructions listed for each type of peripheral equipment, the Request Status and Set Status instructions, as already described are applicable to each peripheral subsystem.

All peripheral instructions are shown as if the second address were to be determined by an SAS. This is not essential; the second location can be specified as being a fixed index word (X = 1 through 6), in which case no SAS is required.

Card Reader Instructions

READ CARD DECIMAL

RCD	C,D,X	07XYYY02
OP/OL	Z	0C0ZZZZZ

Reads one card in the alphanumeric mode.

READ CARD BINARY

RCB	C,D,X	07XYYY01
OP/OL	Z	0C0ZZZZZ

Reads one card in the binary mode.

READ CARD MIXED

RCM	C,D,X	07XYYY03
OP/OL	Z	0C0ZZZZZ

Reads one card in either binary or decimal mode, dependent upon detection of a special character in card column one.

Card Punch Instructions

PUNCH CARD DECIMAL

PCD	C,D,X	07XYYY12
OP/OL	Z	0C0ZZZZZ

Punches one card row in the alphanumeric mode.

PUNCH CARD BINARY

PCB	C,D,X	07XYYY01
OP/OL	Z	0C0ZZZZZ

Punches one card row in the binary mode.

PUNCH CARD IN EDITED MODE

PCE	C,D,X	07XYYY13
OP/OL	Z	0C0ZZZZZ

Punches one card row in the edited alphanumeric mode. Ignore characters are not punched, nor is the corresponding column left blank; instead, it is punched with the next non-Ignore character.

PRINT EDITED MODE

PRE	C,D,X	07XYYY30
OP/OL	Z	0C0ZZZZZ

Prints one line in the edited mode. Data controls slewing.

PRINT EDITED MODE, SLEW SINGLE

PRES	C,D,X	07XYYY31
OP/OL	Z	0C0ZZZZZ

Prints one line in the edited mode and slews one line.

PRINT EDITED MODE, SLEW DOUBLE

PRED	C,D,X	07XYYY32
OP/OL	Z	0C0ZZZZZ

Prints one line in the edited mode and slews two lines.

PRINT EDITED MODE, SLEW TO TOP OF PAGE

PRET	C,D,X	07XYYY33
OP/OL	Z	0C0ZZZZZ

Prints one line in the edited mode and slews to the top of the next page.

PRINT NON-EDIT MODE

PRN	C,D,X	07XYYY10
OP/OL	Z	0C0ZZZZZ

Prints one line in the non-edit mode. Data controls slewing.

PRINT NON-EDIT MODE, SLEW SINGLE

PRNS	C,D,X	07XYYY11
OP/OL	Z	0C0ZZZZZ

Prints one line in the non-edit mode and slews one line.

PRINT NON-EDIT MODE, SLEW DOUBLE

PRND	C,D,X	07XYYY12
OP/OL	Z	0C0ZZZZZ

Prints one line in the non-edit mode and slews two lines.

PRINT NON-EDIT MODE, SLEW TO TOP OF PAGE

PRNT	C,D,X	07XYYY13
OP/OL	Z	0C0ZZZZZ

Prints one line in the non-edit mode and slews to the top of the next page.

SLEW PRINTER SINGLE LINE

SPRS	C,D,X	07XYYY61
OP/OL	Z	0C0ZZZZZ

No printing, slews one line.

SLEW PRINTER DOUBLE LINE

SPRD	C,D,X	07XYYY62
OP/OL	Z	0C0ZZZZZ

No printing, slews two lines.

SLEW PRINTER TO TOP OF PAGE

SPRT	C,D,X	07XYYY63
OP/OL	Z	0C0ZZZZZ

No printing, slews to the top of the next page.

Magnetic Tape Instructions

READ TAPE BINARY

RTB	C,D,X	07XYYY05
OP/OL	Z	0C0ZZZZZ

Reads one tape record in the binary mode.

READ TAPE DECIMAL

RTD	C,D,X	07XYYY04
OP/OL	Z	0C0ZZZZZ

Reads one tape record in the alphanumeric mode.

WRITE TAPE BINARY

WTB	C,D,X	07XYYY15
OP/OL	Z	0C0ZZZZZ

Writes one tape record in the binary mode.

WRITE TAPE DECIMAL

WTD	C,D,X	07XYYY14
OP/OL	Z	0C0ZZZZZ

Writes one tape record in the alphanumeric mode.

WRITE END OF FILE

WEF	C,D,X	07XYYY55
OP/OL	Z	0C0ZZZZZ

Write an end of file record.

BACKSPACE ONE RECORD

BSR	C,D,X	07XYYY46
OP/OL	Z	0C0ZZZZZ

Moves tape backward, without data transfer, over one record and positions tape to read or write the record passed over.

BACKSPACE ONE FILE

BSF	C,D,X	07XYYY47
OP/OL	Z	0C0ZZZZZ

Moves tape backward, without data transfer, until an end of file record is sensed.

SET DENSITY LOW

SDL	C,D,X	07XYYY61
OP/OL	Z	0C0ZZZZZ

Sets the addressed magnetic tape unit into the low density mode.

FORWARD SPACE ONE RECORD

FSR	C,D,X	07XYYY44
OP/OL	Z	0C0ZZZZZ

Moves tape forward, without data transfer, over one record.

FORWARD SPACE ONE FILE

FSF	C,D,X	07XYYY45
OP/OL	Z	0C0ZZZZZ

Moves tape forward, without data transfer, until an end of file record is detected.

REWIND

RWD	C,D,X	07XYYY70
OP/OL	Z	0C0ZZZZZ

Moves tape backward, without data transfer, until the beginning of tape marker is sensed. The tape unit enters the Device Busy major status.

REWIND AND STANDBY

RWS	C,D,X	07XYYY72
OP/OL	Z	0C0ZZZZZ

Moves tape backward, without data transfer, until the beginning of tape marker is sensed. The tape unit enters the standby Attention major status.

ERASE

ERS	C,D,X	07XYYY54
OP/OL	Z	0C0ZZZZZ

Tape is erased in the forward direction for 8½ inches.

Perforated Tape Instructions

READ PERFORATED TAPE

RPT	C,D,X	07XYYY02
OP/OL	Z	0C0ZZZZZ

Causes continuous feeding and reading of tape in the channel mode established by the plugboard.

PUNCH PERFORATED TAPE

PPT	C,D,X	07XYYY11
OP/OL	Z	0C0ZZZZZ

Causes continuous feeding and punching of tape in the 7-channel mode with odd parity (channel 5).

PUNCH EDITED TAPE

PET	C,D,X	07XYYY31
OP/OL	Z	0C0ZZZZZ

Causes continuous feeding and punching of tape in 7-channel mode with odd parity (channel 5). Ignore characters are deleted (not punched).

PUNCH SINGLE CHARACTER MODE TAPE

PST	C,D,X	07XYYY16
OP/OL	Z	0C0ZZZZZ

Causes continuous feeding and punching of tape in 5- or 6-channel mode with no unique parity punched.

PUNCH DOUBLE CHARACTER MODE TAPE

PDT	C,D,X	07XYYY13
OP/OL	Z	0C0ZZZZZ

Causes continuous feeding and punching of tape in 7- or 8-channel mode with no unique parity punched.

Disc Storage Instructions

SEEK FILE

SF	C,D,X	07XYYY34
OP/OL	Z	0C0ZZZZZ

Transfers sector address to disc storage buffer from memory and positions actuator to arm position in which sector occurs.

SEEK/READ FILE

SRF	C,D,X	07XYYY14
OP/OL	Z	0C0ZZZZZ

Transfers sector address and other control characters to buffer from memory, positions actuator, and reads 1 to 4 adjacent sectors into the buffer.

READ FILE

RF	C,D,X	07XYYY54
OP/OL	Z	0C0ZZZZZ

Verifies actuator position and reads 1 to 4 adjacent sectors into the buffer. Assumes sector address and other control characters are already in the buffer.

SEEK/READ FILE AND RELEASE SEEK

SRFR	C,D,X	07XYYY15
OP/OL	Z	0C0ZZZZZ

Transfers sector address and other control characters to buffer from memory, positions actuator, reads 1 to 4 adjacent sectors into the buffer, and releases the actuator.

READ FILE AND RELEASE SEEK

RFR	C,D,X	07XYYY55
OP/OL	Z	0C0ZZZZZ

Verifies actuator position, reads 1 to 4 adjacent sectors into the buffer, and releases the actuator. Assumes sector address and other control characters are already in the buffer.

SEEK/READ FILE AND INCREMENT ADDRESS

SRFI	C,D,X	07XYYY16
OP/OL	Z	0C0ZZZZZ

Transfers sector address and other control characters from memory to buffer, positions actuator, reads 1 to 4 sectors into the buffer, and increments by one the file address stored in the buffer.

READ FILE AND INCREMENT ADDRESS

RFI	C,D,X	07XYYY56
OP/OL	Z	0C0ZZZZZ

Assumes sector address and other control characters are already in the buffer, positions actuator, reads 1 to 4 sectors into the buffer, and increments by one the file address stored in the buffer.

SEEK/WRITE FILE

SWF	C,D,X	07XYYY10
OP/OL	Z	0C0ZZZZZ

Transfers sector address and other control characters from memory to buffer, positions actuator, and writes 1 to 4 adjacent sectors from the buffer.

WRITE FILE

WF	C,D,X	07XYYY50
OP/OL	Z	0C0ZZZZZ

Verifies actuator position and writes 1 to 4 adjacent sectors from the buffer. Assumes sector address and other control characters are already in the buffer.

SEEK/WRITE FILE AND RELEASE SEEK

SWFR	C,D,X	07XYYY11
OP/OL	Z	0C0ZZZZZ

Transfers sector address and other control characters from memory to buffer, positions actuator, writes 1 to 4 adjacent sectors from the buffer, and releases the actuator.

WRITE FILE AND RELEASE SEEK

WFR	C,D,X	07XYYY51
OP/OL	Z	0C0ZZZZZ

Verifies actuator position, writes 1 to 4 adjacent sectors from the buffer, and releases the actuator. Assumes sector address and other control characters are already in the buffer.

SEEK/WRITE FILE AND INCREMENT ADDRESS

SWFI	C,D,X	07XYYY12
OP/OL	Z	0C0ZZZZZ

Transfers sector address and other control characters from memory to buffer, positions actuator, writes 1 to 4 sectors from the buffer, and increments by one the file address stored in the buffer.

WRITE FILE AND INCREMENT ADDRESS

WFI	C,D,X	07XYYY52
OP/OL	Z	0C0ZZZZZ

Assumes sector address and other control characters are already in the buffer, positions actuator, writes 1 to 4 sectors from the buffer, and increments by one the file address stored in the buffer.

SEEK/WRITE FILE AND VERIFY

SWFV	C,D,X	07XYYY13
OP/OL	Z	0C0ZZZZZ

Transfers sector address and other control characters to the buffer, positions actuator, writes 1 to 4 adjacent sectors from the buffer, and compares data written with buffer contents.

WRITE FILE AND VERIFY

WFV	C,D,X	07XYYY53
OP/OL	Z	0C0ZZZZZ

Verifies actuator position, writes 1 to 4 adjacent sectors from the buffer, and compares data written with buffer contents. Assumes sector address and other control characters are already in the buffer.

READ FILE CONTINUOUS AND RELEASE SEEK

RFCR	C,D,X	07XYYY25
OP/OL	Z	0C0ZZZZZ

Positions actuator and reads up to 32 adjacent sectors into memory by way of the buffer. Assumes sector address and other control characters are already in the buffer.

WRITE FILE CONTINUOUS AND RELEASE SEEK

WFCR	C,D,X	07XYYY31
OP/OL	Z	0C0ZZZZZ

Verifies actuator position and writes up to 32 adjacent sectors from memory by way of the buffer. Assumes sector address and other control characters are already in the buffer.

WRITE FILE CONTINUOUS, VERIFY, AND RELEASE SEEK

WFCV	C,D,X	07XYYY33
OP/OL	Z	0C0ZZZZZ

Verifies actuator position, writes up to 32 adjacent sectors from memory by way of the buffer, and verifies each written sector check character against a new check character generated by rereading each sector. Assumes sector address and other control characters are already in the buffer.

SEEK COMPARE

SCPR	C,D,X	07XYYY17
OP/OL	Z	0C0ZZZZZ

Transfers sector address and other control characters to buffer, positions actuator, reads and compares up to 32 adjacent sectors with buffer contents, places address of matching sector in the buffer, and fills buffer with contents of up to 3 adjacent sectors.

COMPARE

CPR	C,D,X	07XYYY57
OP/OL	Z	0C0ZZZZZ

Same as SEEK/COMPARE instruction except sector address and other control characters are assumed to be in the buffer.

SEEK/LINK

SLNK	C,D,X	07XYYY35
OP/OL	Z	0C0ZZZZZ

Transfers sector address and other control characters to buffer, positions actuator, reads one sector to buffer, locates (in buffer) the address of next sector in chain, and transfers that address to the buffer address control for the next command.

LINK

LNK	C,D,X	07XYYY75
OP/OL	Z	0C0ZZZZZ

Same as SEEK LINK instruction, except sector address and other control characters are assumed to be in the buffer.

ACCEPT BUFFER ADDRESS

ABA	C,D,X	07XYYY32
OP/OL	Z	0C0ZZZZZ

Transfers a buffer address to the controller.

READ BUFFER

RB	C,D,X	07XYYY24
OP/OL	Z	0C0ZZZZZ

Transfers successive characters from buffer to memory.

WRITE BUFFER

WB	C,D,X	07XYYY30
OP/OL	Z	0C0ZZZZZ

Transfers successive characters from memory to buffer.

LOAD BUFFER FOR COMPARE

LBFC	C,D,X	07XYYY36
OP/OL	Z	0C0ZZZZZ

Transfers successive characters from memory to sequential buffer positions.

MOVE DATA

MVDT	C,D,X	07XYYY37
OP/OL	Z	0C0ZZZZZ

Transfers starting addresses for source and destination fields to buffer from memory and moves up to 240 characters from source to destination within the buffer.

Magnetic Reader/Sorter Instructions

FEED DOCUMENT

FDOC OP/OL	C,D,X Z	07XYYY41 0C0ZZZZZ

Initiates the feeding of a single document.

READ DOCUMENT

RDOC OP/OL	C,D,X Z	07XYYY01 0C0ZZZZZ

Reads one document.

POCKET DOCUMENT

PDOC OP/OL	C,D,X Z	07XYYY43 0C0ZZZZZ

Diverts the document currently in transit into the selected pocket.

Typewriter Instructions

TYPE OUTPUT OCTAL

TOO	X	07XYYY11
OP/OL	Z	0C0ZZZZZ

Initiates an octal type-out from memory

TYPE OUTPUT ALPHANUMERIC

TOA	X	07XYYY13
OP/OL	Z	0C0ZZZZZ

Initiates an alphanumeric type-out from memory.

TYPE INPUT OCTAL

TIO	X	07XYYY01
OP/OL	Z	0C0ZZZZZ

Turns on the OCT REQD light on the control console to inform the operator that the computer is ready to receive an octal type-in.

TYPE INPUT ALPHANUMERIC

TIA	X	07XYYY03
OP/OL	Z	0C0ZZZZZ

Turns on the ALP REQD light on the control console to inform the operator that the computer is ready to receive an alphanumeric type-in.

More information pertaining to programming typewriter operations is presented in the Typewriter Input/Output Programming section of this manual.

IV. ADDRESSING AND ADDRESS MODIFICATION

In the GE-425/435, instructions are executed in a sequence (P-sequence) controlled by the program counter (P-counter). The P-counter contains the memory address of one of three types of words:

1. Instruction word
2. P-sequence Address Modification Sequence (AMS) word
3. P-sequence Second Address Sequence (SAS) word

Each time one of these words is obtained from memory under P-counter control, the P-counter is incremented by one so that the next sequential word can be obtained after the current word is processed. Thus, the P-counter determines the P-sequence, which is the order in which program instructions are executed. Other than the normal incrementing by one, the contents of the P-counter can only be altered by any of ten branch instructions:

BRU	SPB	BRE	BRL	BRC
BRZ	PXB	BRM	BRG	BXC

The P-counter is not affected by any other machine operations. Each instruction word for the GE-425/435 contains an address field that, in most instructions, references a memory location. Some instructions are two-address instructions, the second address being determined by an SAS initiated in the P-sequence. The second address of two-address instructions cannot be modified.

The address field of the single-address instruction and the first address of two-address instructions can be modified.

During instruction processing, the central processor advances through the following general sequence (see the flow chart in Figure IV-1):

1. Obtains the instruction word from memory.

2. Performs any specified modification of the first address, accessing words outside the P-sequence if necessary.

3. Obtains the second address if the instruction word operation code identifies the instruction as a two-address instruction.

4. Executes the instruction.

5. Obtains the next instruction.

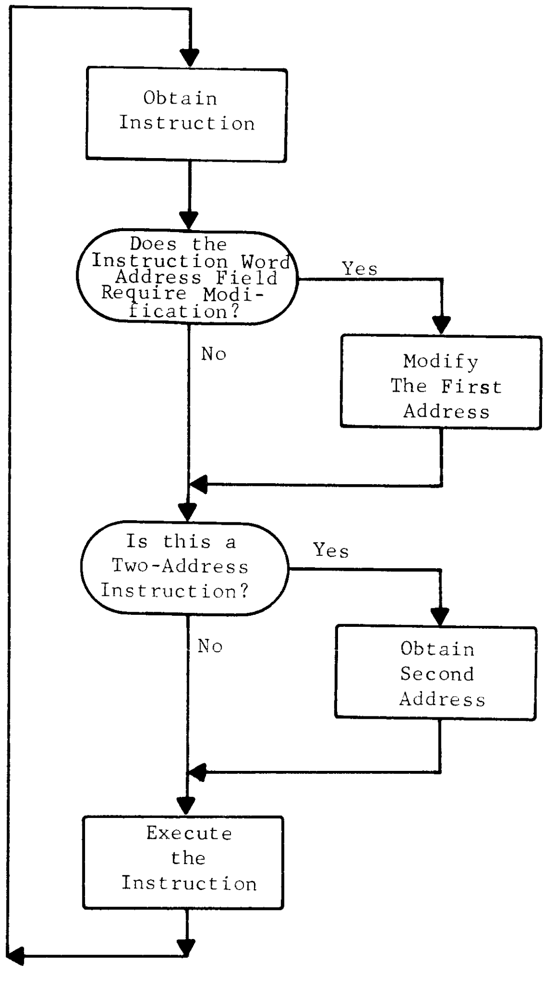


Figure IV-1. Simplified Flow Chart of Instruction Processing Sequence

The effects of individual instruction execution are described in Section III, Instruction Repertoire. The balance of this section is devoted to:

1. Modification of single-address instructions.
2. Modification of the first address of two-address instructions.
3. Determining the second address of two-address instructions.

In the discussions of address modification that follow, refer to the flow chart of GE-425/435 addressing in Figure IV-23.

MODIFICATION OF SINGLE-ADDRESS INSTRUCTIONS

Most GE-425/435 instructions are single-address instructions. The format of the single-address instruction word is shown in Figure IV-2.

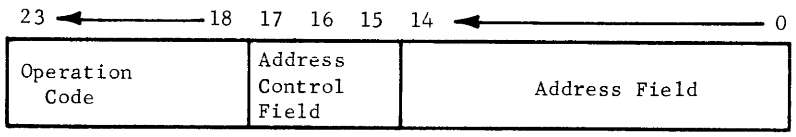


Figure IV-2. Instruction Word Format

Bits 0 through 14 of the instruction word contain the address field that, after any specified modification, becomes the effective address of the instruction.

Bits 15, 16, and 17 of the instruction word form the address control field (ACF) and determines the type of modification, if any, to be performed on the address field. The ACF can contain an octal digit (0 through 7) as follows:

ACF = 0	No address modification is specified. Execute the instruction with the address field as the effective address.
ACF = 1-6	The contents of a fixed index word (locations 1 through 6) are the modifier for the address field.
ACF = 7	The address field is to be modified through an AMS originating in the P-sequence.

Fixed Index Word Modification

If the instruction word ACF contains an octal 1 through 6, one of six fixed index words (memory locations 1 through 6) contains a modifier for the instruction word address field. The format of the fixed index word, as it is used for address modification, is shown in Figure IV-3.

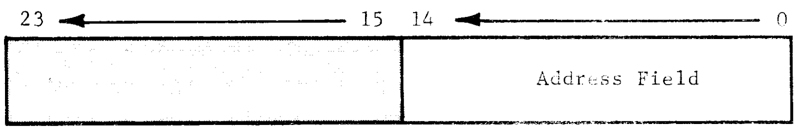
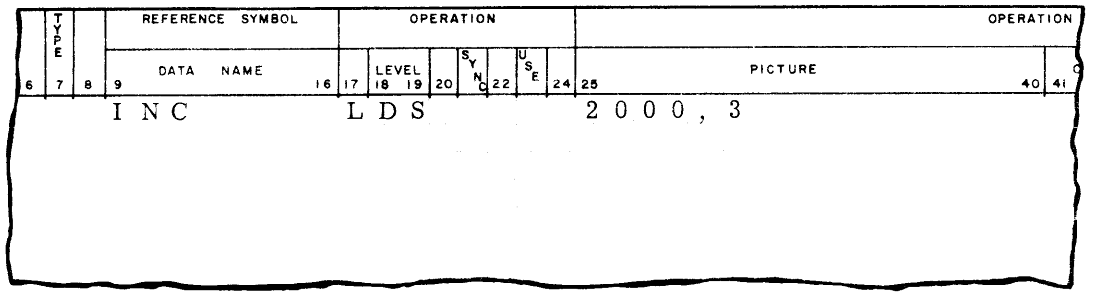


Figure IV-3. Fixed Index Word Format for Modification

When a fixed index word is referenced by a single-address instruction word, the contents of the fixed index word address field (bits 0 through 14) are added to the instruction word address field. The instruction is then executed using the modified address.

The Macro Assembly Program provides a convenient means for coding fixed index word modification. The desired fixed index word is designated in the Operation Parameters field of the coding sheet following the address expression and separated from it by a comma.

Figure IV-4 illustrates an LDS instruction that is coded for fixed index word modification.



6	TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION / LEVEL 17–24	OPERATION / PICTURE 25–40	41
			I N C	L D S	2 0 0 0 , 3	

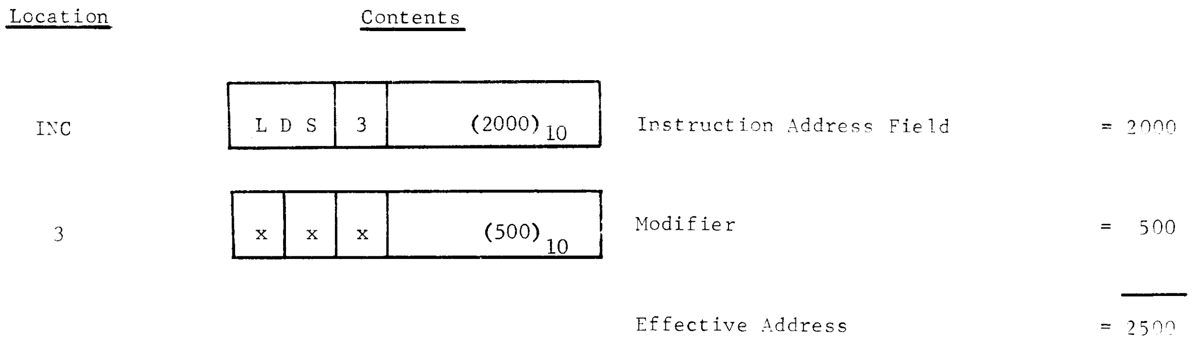


Figure IV-4. Fixed Index Word Modification

The 3 in the ACF specifies fixed index word 3. Because LDS is a single-address instruction, its address field (2000) is modified by the contents of the FIW address field (500), forming the effective address (2500). No further address modification is possible and the instruction is executed as if it were an LDS 2500.

Address Modification Sequence

If the ACF of a single-address instruction contains a 7 (111 in bit positions 15 through 17), the central processor automatically enters an Address Modification Sequence (AMS) in order to obtain the modifier or index quantity. The next word in the P-sequence is obtained under the control of the P-counter and is interpreted as a P-sequence AMS word of the format shown in Figure IV-5.

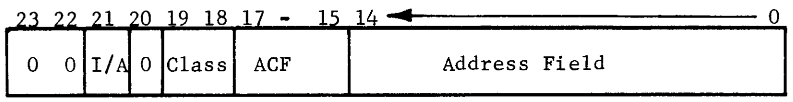


Figure IV-5. P-Sequence AMS Word Format

In the P-sequence AMS word, bit positions shown to contain zeros (positions 23, 22, and 20) must contain zeros in order to preserve GE-400 programming compatibility. Bit position 21 is the indirect address control field and is discussed under the heading, Indirect Addressing, following this subsection.

The effect of the P-sequence AMS word is determined by the Class field (bit positions 18 and 19). Three classes are possible:

	Class	Bits 19	Bits 18
1.	Index	0	0
2.	Index Pointer	0	1
3.	Index Link	1	0

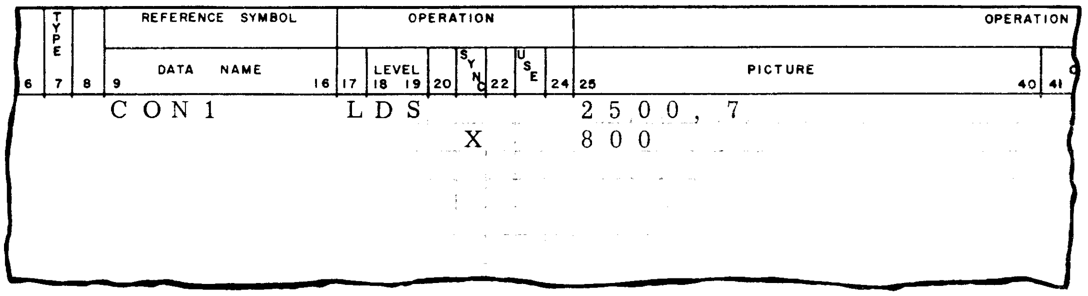
NOTE: The combination 11 in bits 19 and 18 of AMS words must not be used if GE-400 programming compatibility is to be preserved.

The Macro Assembly Program provides a convenient means for coding modifications by AMS. An AMS can be entered by coding a 7 in the Operation Parameters field of the coding sheet following the address expression and separated from it by a comma. P-sequence AMS words are entered by using mnemonics starting in column 21 of the coding sheet. These mnemonics are:

X	for Index
XP	for Index Pointer
XL	for Index Link

If the P-sequence AMS word is classified as an Index (00), the address field of the AMS word is the modifier and is added to the instruction word address field. The instruction is executed using this modified address, unless the AMS word ACF contains a 7 (indicating further modification) or unless the AMS word indicates indirect addressing.

Figure IV-6 illustrates an LDS instruction that is modified through an AMS, using a P-sequence AMS word classified as an Index.



TYPE	REFERENCE SYMBOL / DATA NAME (9–16)	OPERATION (17–24)		OPERATION / PICTURE (25–40)
	CON 1	LDS		2500, 7
			X	800

Location	Contents					
CON1	L D S	7	$(2500)_{10}$	Instruction Address Field		= 2500
CON1+1	x x	x	$(800)_{10}$	Index		= 800
				Tentative Address = Effective Address		= 3300

Figure IV-6. Modification with a P-Sequence AMS Index

The 7 in the ACF of the LDS instruction specifies an AMS and the class field of the next word in the P-sequence (location CON1+1) is examined. The P-sequence AMS word is an Index whose address field (800) is added to the instruction address field (2500), forming a tentative address (3300). The indirect address field and the ACF of the Index indicate that no further modification is to occur. The tentative address, therefore, is the effective address and the instruction is executed as if it were an LDS 3300.

If the AMS word is classified as an Index Pointer (01), then the address field of the AMS word contains the location of a word outside the P-sequence that becomes a modifier. The address field of this word is added to the instruction word address field and the instruction is executed using the modified address, unless further modification is specified by the P-sequence AMS word or unless the AMS word indicates indirect addressing. Figure IV-7 illustrates an LDS instruction that is modified through an AMS, using a P-sequence AMS word classified as an Index Pointer.

6	TYPE 7	8	REFERENCE SYMBOL DATA NAME 9–16	OPERATION 17	LEVEL 18 19	20	SYN 21	22	USE 23	24	OPERATION PICTURE 25–40	41
			CON2	LDS							4800 7	
							X	P			3000	

Location	Contents					
CON2	L D S	7		$(4800)_{10}$	Instruction Address Field	= 4800
CON2+1	0	1	0	$(3000)_{10}$	Index Pointer	
CON2+2	x	x	x	$(1350)_{10}$	Modifier	= 1350
					Tentative Address = Effective Address	= 6150

Figure IV-7. Modification with a P-Sequence AMS Index Pointer

If the AMS word is classified as an Index Link (10), then the address field of the AMS word contains the location of a word outside the P-sequence that functions as a remote AMS word with the format shown in Figure IV-8.

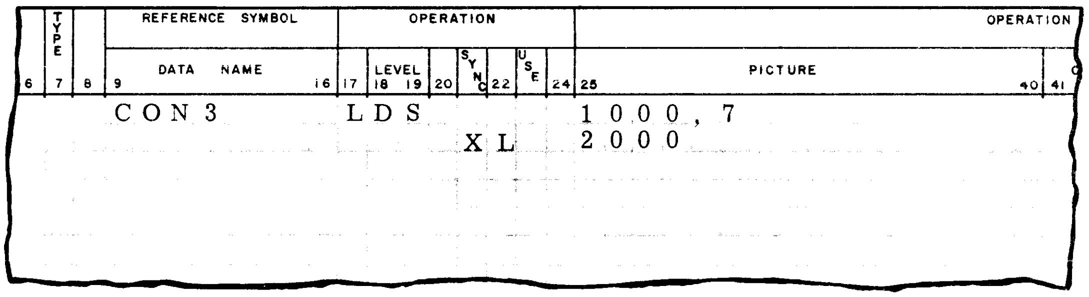
23 22	21	20	19 18	17 - 15	14 ← 0
0 0	▨	0	Class	▨	Address Field

Figure IV-8. Remote AMS Word Format

A remote AMS word functions essentially as an Index, Index Pointer, or Index Link, depending upon its class field. However, the ACF field and the indirect address field of the remote AMS word are ignored. If the remote AMS word is an Index Pointer or Index Link, it causes one or more additional words to be accessed to obtain the modifier.

The Macro Assembly Program provides the same mnemonics (X, XP, or XL) to form a remote AMS word that are used to form a P-sequence AMS word.

Figure IV-9 illustrates an LDS instruction that is modified through an AMS using a P-sequence AMS word classified as an Index Link.



REFERENCE SYMBOL / DATA NAME (9-16)	OPERATION (17-24)	OPERATION / PICTURE (25-40)
CON3	LDS	1000, 7
	XL	2000

Location	Contents					
CON3	L D S	7	$(1000)_{10}$	Instruction Address Field	= 1000
CON3+1	0	2	0	$(2000)_{10}$	Index Link	
2000	0	1	0	$(3000)_{10}$	Index Pointer	
3000	x	x	x	$(20)_{10}$	Modifier	= 20

Tentative Address = Effective Address = 1020

Figure IV-9. Modification with a P-Sequence AMS Index Link

The 7 in the ACF of the LDS instruction specifies an AMS and the class field of the next word in the P-sequence (location CON3+1) is examined. The P-sequence AMS word is an Index Link whose address field (2000) specifies the location of a remote AMS word. This remote AMS word is an Index Pointer whose address field (3000) specifies the location of another word outside the P-sequence. The address field of this word (20) is added to the instruction address field (1000), forming a tentative address (1020). The indirect address field and the ACF of the Index Link indicate that no further modification is to occur. The tentative address, therefore, is the effective address and the instruction is executed as if it were an LDS 1020.

After an effective address has been determined by virtue of processing an Index either in the P-sequence or in a remote AMS word (and assuming that the P-sequence AMS word does not specify indirect addressing), the P-sequence AMS word ACF is examined. If the ACF contains a 7, the next word in the P-sequence is accessed and is treated as another P-sequence AMS word. The effective address thus far developed is subjected to further modification.

Figure IV-10 contains a flow chart that summarizes the Address Modification Sequence from the point where the P-sequence AMS word has been accessed to the point where the modified address has been formed.

Any number of P-sequence AMS words and/or remote AMS words can be referenced in determining the effective address of an instruction. Thus, facility is provided for multiple indexing. Figure IV-11 illustrates the use of two P-sequence AMS words to establish the effective address.

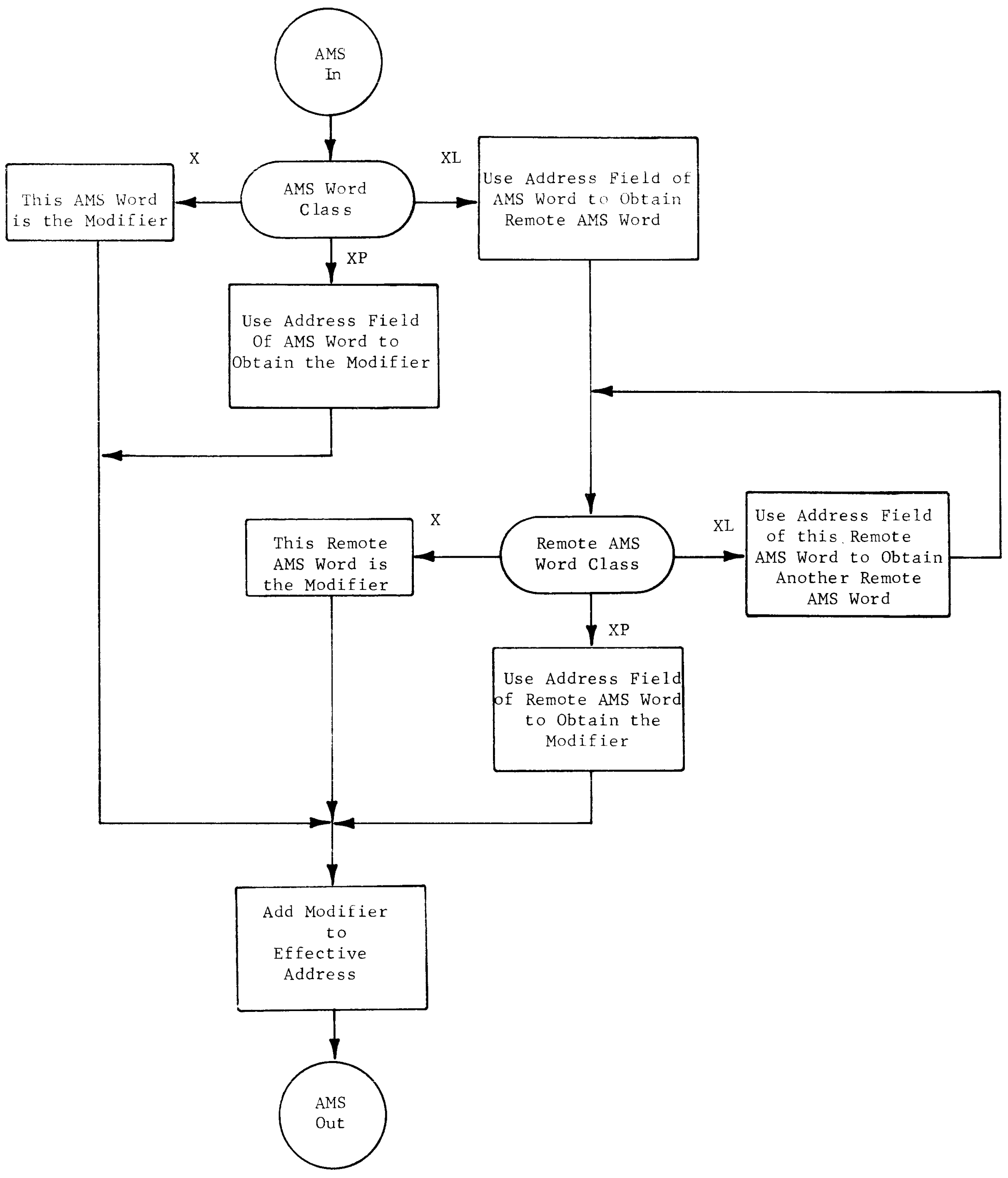
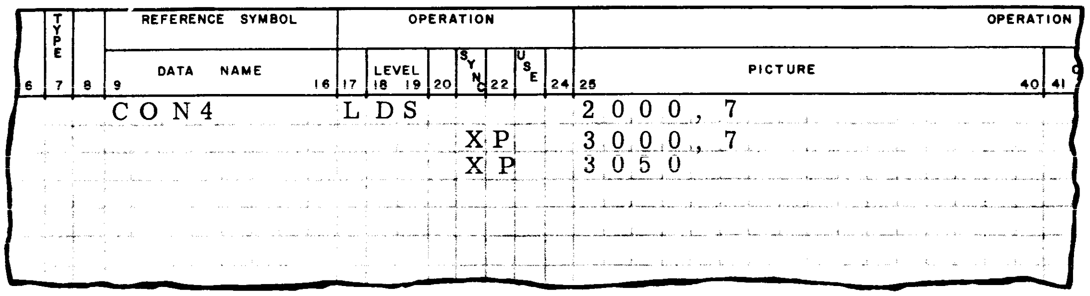

AMS In
AMS Word Class
X
XL
XP
This AMS Word is the Modifier
Use Address Field of AMS Word to Obtain Remote AMS Word
Use Address Field Of AMS Word to Obtain the Modifier
Remote AMS Word Class
X
XL
XP
This Remote AMS Word is the Modifier
Use Address Field of this Remote AMS Word to Obtain Another Remote AMS Word
Use Address Field of Remote AMS Word to Obtain the Modifier
Add Modifier to Effective Address
AMS Out

Figure IV-10. Address Modification Sequence Flow Chart



REFERENCE SYMBOL DATA NAME	OPERATION	OPERATION PICTURE
CON4	LDS	2000, 7
	XP	3000, 7
	XP	3050

Location	Contents					
CON4	L D S	7	$(2000)_{10}$	Instruction Address Field	= 2000
CON4+1	0 | 1	7	$(3000)_{10}$	Index Pointer	
3000	x | x	x	$(40)_{10}$	Modifier	= 40
				Tentative Address	= 2040
CON4+2	0 | 1	0	$(3050)_{10}$	Index Pointer	
3050	x | x	x	$(2)_{10}$	Modifier	= 2
				Tentative Address = Effective Address	= 2042

Figure IV-11. Modification with Multiple P-Sequence AMS Words

The 7 in the ACF of the LDS instruction specifies an AMS, and the class field of the next word in the P-sequence (location CON4+1) is examined. The P-sequence AMS word is an Index Pointer whose address field (3000) specifies the location of a modifier. The address field of this word (40) is added to the instruction address field (2000) forming a tentative address (2040). The indirect address field of the Index Pointer indicates no indirect addressing. However, the ACF indicates continuation of the AMS and the class field of the next word in the P-sequence (location CON4+2) is examined. This AMS word is another P-sequence Index Pointer whose address field (3050) specifies the location of another modifier.

The address field of this word (2) is added to the tentative address (2040), forming a new tentative address (2042). The indirect address field and the ACF of the second Index Pointer indicate that no further modification is to occur. The tentative address, therefore, becomes the effective address and the instruction is executed as if it were an LDS 2042.

Indirect Addressing

After the P-sequence AMS word has caused a modifier to be found and added to the instruction address field, bit position 21 of this same P-sequence AMS word is examined to determine if indirect addressing is involved. If bit position 21 contains a one, the modified address becomes the address of a location outside the P-sequence containing an indirect address word. The format of the indirect address word is shown in Figure IV-12.

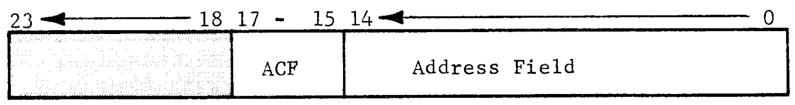


Figure IV-12. Indirect Address Word Format

When the indirect AMS word is accessed, the class field is examined to determine if the word contains a modifier (00 = Index), points to a modifier (01 = Index Pointer), or links (10 = Index Link) to another word to determine the modifier for the indirect address word. After location of the modifier and modification of the indirect address word, the I/A field (bit 21) of the indirect AMS word is examined. If bit 21 is a one, the modified address thus far developed is the location of still another indirect address word.

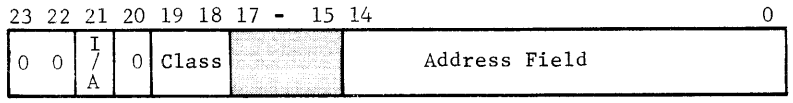


Figure IV-13. Indirect AMS Word

When the indirect AMS word is accessed, the class field is examined to determine if the word contains a modifier (00 = Index), points to a modifier (01 = Index Pointer), or links (10 = Index Link) to another word to determine the modifier for the indirect address word. After location of the modifier and modification of the indirect address word, the I/A field (bit 21) of the indirect AMS word is examined. If bit 21 is a one, the modified address thus far developed is the location of still another indirect address word.

The procedure just described continues until:

1. An indirect address word ACF does not specify continuation (contains something other than 7), <u>or</u>,

2. An indirect AMS word does not specify further indirect addressing (bit 21 contains a zero).

When an indirect address sequence is concluded, control is returned to the P-sequence AMS word that initiated the indirect address sequence. The ACF of the P-sequence word is then examined for continuation (7 in the ACF) and, if so specified, the instruction address field is subjected to further modification under control of the next P-sequence AMS word.

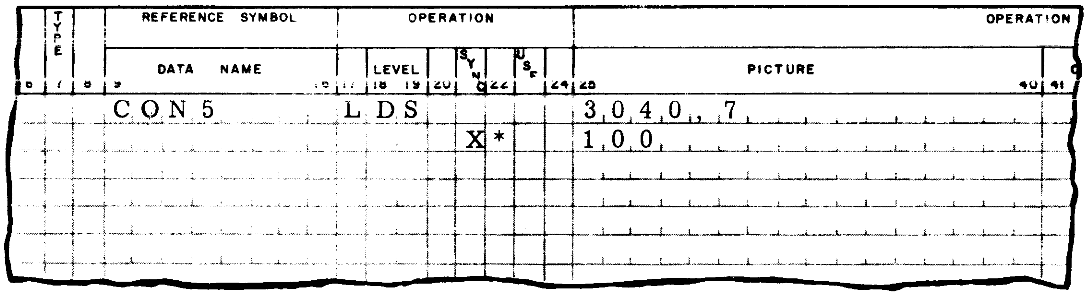
The Macro Assembly Program accepts an asterisk immediately following an AMS mnemonic to indicate indirect addressing as follows:

X*	for Index Indirect
XP*	for Index Pointer Indirect
XL*	for Index Link Indirect

The indirect address word (Figure IV-12) can be specified in an area of memory outside the P-sequence with the mnemonic code, IAW.

The IAW pseudo-instruction causes an indirect address word to be formed, with the first expression appearing in the Operation Parameters field becoming the address field of the IAW. If there is a second expression, it is used as the ACF of the IAW.

Figures IV-14 through IV-16 illustrate the use of indirect addressing capabilities.

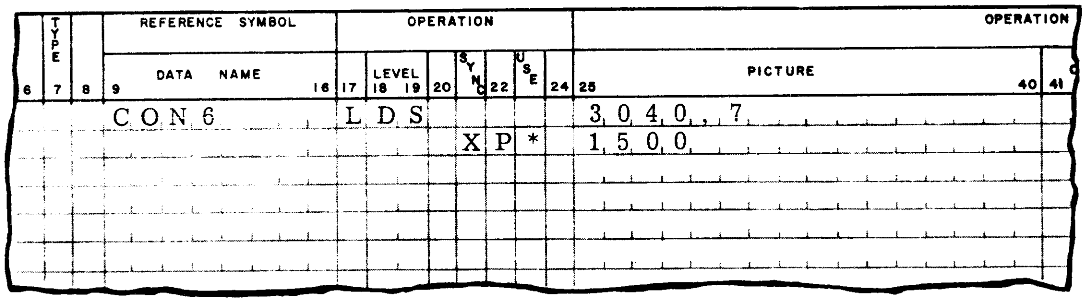


TYPE	REFERENCE SYMBOL / DATA NAME	OPERATION / LEVEL		SYNC		USF		OPERATION / PICTURE
	CON 5	LDS						3,0,4,0,,7
				X	*			1,0,0

Location	Contents					
CON5	L D S	7	$(3040)_{10}$		Instruction Address Field	= 3040
CON5+1	1	0	0	$(100)_{10}$	Index, Indirect	= 100
					Tentative Address	= 3140
3140	x	x	0	$(2100)_{10}$	IAW	
					Effective Address = Replacement Address	= 2100

Figure IV-14. Modification with a P-Sequence AMS Index, Indirect

The 7 in the ACF of the LDS instruction specifies an AMS and the class field of the next word in the P-sequence (location CON5+1) is examined. The P-sequence AMS word is an Index whose address field (100) is added to the instruction address field (3040). Because the P-sequence AMS word indicates indirect addressing, the tentative address (3140) is the address of an indirect address word (IAW). The address field of the IAW (2100) replaces the tentative address and becomes a new tentative address. The ACF of the IAW and the ACF of the Index indicate that no further modification is to occur. The tentative address, therefore, is the effective address and the instruction is executed as if it were an LDS 2100.



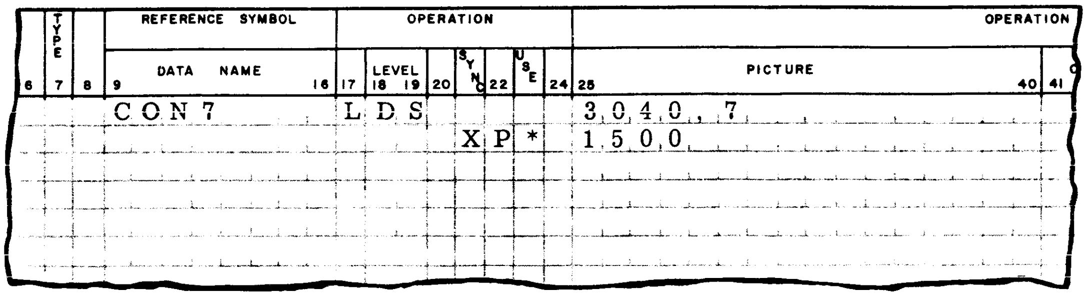
6	7 TYPE	8	9 REFERENCE SYMBOL / DATA NAME 16	17 OPERATION	18 LEVEL 19	20	21 SYNC	22	23 USE	24	25 PICTURE 40	41
			CON6	LDS							3040 7	
							X	P	*		1500	

Location	Contents					
CON6	L D S	7		$(3040)_{10}$	Instruction Address Field	= 3040
CON6+1	1	1	0	$(1500)_{10}$	Index Pointer, Indirect	
1500	x	x	x	$(100)_{10}$	Modifier	= 100
					Tentative Address	= 3140
3140	x	x	0	$(2100)_{10}$	IAW	

Replacement Address = Effective Address = 2100

Figure IV-15. Modification with a P-Sequence AMS Index Pointer, Indirect

The 7 in the ACF of the LDS instruction specifies an AMS and the class field of the next word in the P-sequence (location CON6+1) is examined. The P-sequence AMS word is an Index Pointer whose address field (1500) specifies the location of a modifier. The address field of this word (100) is added to the instruction address field (3040). Because the P-sequence AMS word indicates indirect addressing, the tentative address (3140) is the address of an indirect address word. The address field of the IAW (2100) replaces the tentative address and becomes a new tentative address. The ACF of the IAW and the ACF of the Index Pointer indicate that no further modification is to occur. The tentative address, therefore, is the effective address and the instruction is executed as if it were an LDS 2100.



TYPE 6	7	8	REFERENCE SYMBOL: DATA NAME 9–16	OPERATION: 17	LEVEL 18 19	20	SYNC 21–22	USE 23–24	OPERATION: PICTURE 25–40	41
			CON7	L	DS				3040,7	
							XP	*	1500	

Location	Contents			
CON7	L D S | 7 | $(3040)_{10}$	Instruction Address Field	= 3040
CON7+1	1 | 1 | 0 | $(1500)_{10}$	Index Pointer, Indirect	
1500	x | x | x | $(100)_{10}$	Modifier	= 100
		Tentative Address	= 3140
3140	x | x | x | $(2000)_{10}$	IAW Replacement Address	= 2000
3141	0 | 1 | x | $(1500)_{10}$	Indirect AMS (Index Pointer)	= 100
		Tentative Address = Effective Address	= 2100

Figure IV-16. Modification with a P-Sequence AMS Index Pointer, Indirect, and an Indirect AMS Word

The 7 in the ACF of the LDS instruction specifies an AMS and the class field of the next word in the P-sequence (location CON7+1) is examined. The P-sequence AMS word is an Index Pointer whose address field (1500) specifies the location of a modifier. The address field of this word (100) is added to the instruction address field (3040). Because the P-sequence AMS word indicates indirect addressing, the tentative address (3140) is the address of an indirect address word. The address field of the IAW (2000) replaces the tentative address and becomes a new tentative address. Furthermore, the ACF of the IAW specifies an indirect AMS word that causes an examination of the class field of the next sequential location (3141). The indirect AMS word is an Index Pointer whose address field (1500) specifies the location of another modifier. The address field of this word (100) is added to the tentative address (2000), forming another tentative address. The ACF of the P-sequence Index Pointer indicates that no further modification is to occur. The tentative address, therefore, is the effective address and the instruction is executed as if it were an LDS 2100. Note that, in this illustration, the modifier in location 1500 is used twice.

TWO-ADDRESS INSTRUCTIONS

Two-address instructions fall into two categories:

1. Two-address instructions specifying a fixed index word as the second address (called Fixed Index Word Instructions in Section III, Instruction Repertoire).

2. Two-address instructions using a Second Address Sequence to determine the second address.

The first address of two-address instructions cannot be modified by fixed index word modification. All other modification procedures described in the preceding pages are fully applicable to the first address of two-address instructions.

The second address of the GE-425/435 two-address instructions cannot be modified.

Fixed Index Word Instructions

The fixed index word instructions are two-address instructions that reference a fixed index word for the second address. These instructions are:

LDX	LXI	SXA	MXC	AIX
ABX	SBX	CXI	CXM	PXB
BXC	ANX	RIX	RXX	

Proper Macro Assembly Program coding for these instructions is illustrated in Section III, Instruction Repertoire.

Second Address Sequence

For two-address instructions referencing a second address other than a fixed index word, a Second Address Sequence (SAS) is required.

The SAS is initiated after any first-address modification has been completed. If the instruction word operation code identifies an instruction as a two-address instruction, the next P-sequence word that is accessed after any AMS words used to form the first address must be one of three classes of P-sequence SAS words:

	Bits	
Class	19	18
Operand	0	0
Operand Pointer	0	1
Operand Link	1	0

NOTE: The combination 11 in bits 19 and 18 of SAS words must not be used in order to preserve GE-400 programming compatibility.

The format of the P-sequence SAS word is illustrated in Figure IV-17.

23 22 21 20	19 18	17 - 15	14 ← 0
0 0 0 0	Class	0 0 0	Address Field

Figure IV-17. P-Sequence SAS Word Format

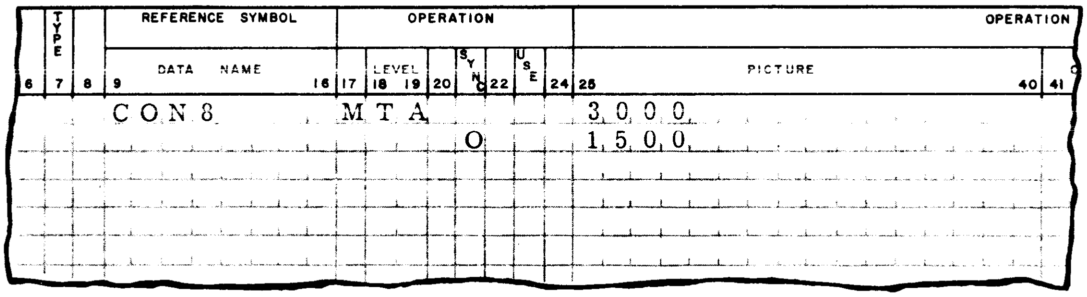
If the P-sequence SAS word is classified as an Operand (00), the location of the SAS word itself is the second address. An Operand terminates an SAS. Figure IV-18 illustrates second address determination using an Operand.

SAS words are entered in the coding sheet for the Macro Assembly Program beginning in column 21. The mnemonic codes for both P-sequence and remote SAS words are:

O for Operands

OP for Operand Pointers

OL for Operand Links



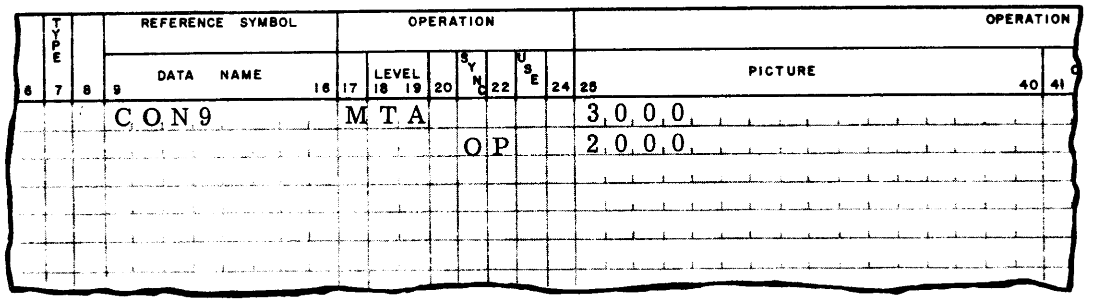
	TYPE		REFERENCE SYMBOL	OPERATION							OPERATION	
			DATA NAME		LEVEL		SYNC		USE		PICTURE	
6	7	8	9 16	17	18 19	20	21 22		23	24	25 40	41
			CON8	MTA							3000	
							O				1500	

Location	Contents				
CON8	MTA	0	$(3000)_{10}$		Instruction Word with First Address
CON8+1	0	0	0	$(1500)_{10}$	Operand (and Second Address)

Figure IV-18. Second Address Determination Using an SAS Operand

The ACF of the MTA instruction specifies that no first address modification is to occur and the class field of the next word in the P-sequence (location CON8+1) is examined. The P-sequence SAS word is an Operand. Its location (CON8+1), therefore, is the second address of the MTA instruction.

If the P-sequence SAS word is classified as an Operand Pointer (01), the address field of the P-sequence SAS word contains the second address. The word addressed by the Operand Pointer is not examined for SAS characteristics and the SAS terminates. Figure IV-19 illustrates second address determination using an Operand Pointer.



	TYPE		REFERENCE SYMBOL		OPERATION						OPERATION	
6	7	8	DATA NAME 9–16	17	LEVEL 18 19	20	SYNC 21	22	USE 23	24	PICTURE 25–40	41
			CON9	M	TA						3000	
							O	P			2000	

Location	Contents				
CON9	M T A	0	$(3000)_{10}$		Instruction Word with First Address
CON9+1	0	1	0	$(2000)_{10}$	Operand Pointer

Figure IV-19. Second Address Determination Using an SAS Operand Pointer

The ACF of the MTA instruction specifies that no first address modification is to occur and the class field of the next word in the P-sequence (location CON9+1) is examined. The P-sequence SAS word is an Operand Pointer whose address field (2000) is the second address of the MTA 3000 instruction.

If the P-sequence SAS word is classified as an Operand Link (10), the address field of the P-sequence SAS word contains the address of a remote SAS word of the format shown in Figure IV-20. The remote SAS word can be an Operand, an Operand Pointer, or another Operand Link. A remote SAS word terminates the SAS.

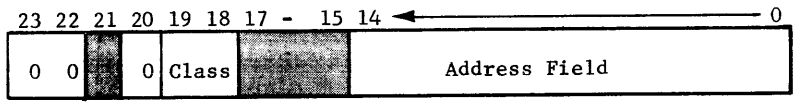
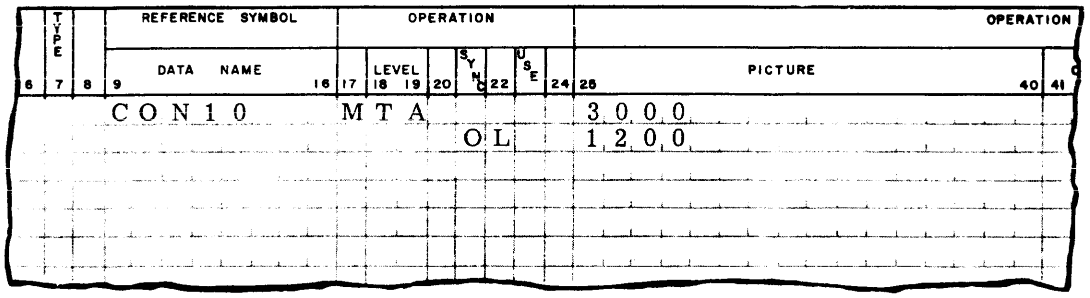


Figure IV-20. Remote SAS Word Format

The illustration in Figure IV-21 shows second address determination using an Operand Link and a remote SAS word.



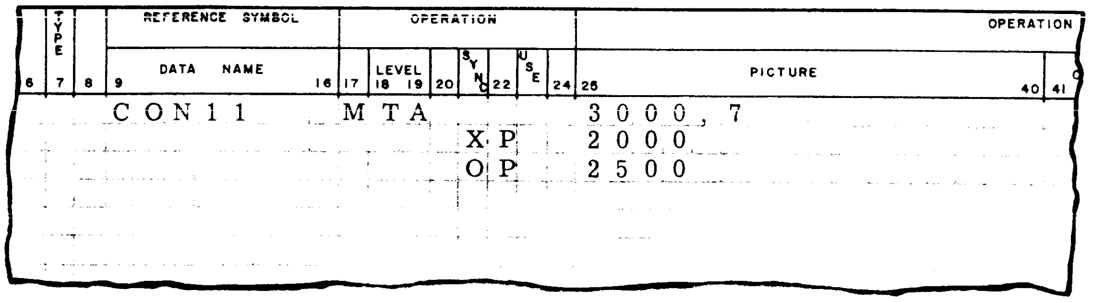
TYPE 7	8	REFERENCE SYMBOL / DATA NAME 9–16	OPERATION / LEVEL 17–20	SYNC 21–22	USE 23–24	OPERATION / PICTURE 25–40	41
		CON10	MTA			3000	
				OL		1200	

Location	Contents				
CON10	M T A	0		$(3000)_{10}$	Instruction Word with First Address
CON10+1	0	2	0	$(1200)_{10}$	Operand Link
1200	0	1	X	$(2000)_{10}$	Operand Pointer (remote SAS word)

Figure IV-21. Second Address Determination Using an SAS Operand Link and Remote SAS Word

The ACF of the MTA instruction specifies that no first address modification is to occur and the class field of the next word in the P-sequence (location CON10+1) is examined. The P-sequence SAS word is an Operand Link whose address field (1200) specifies the location of a remote SAS word. This remote SAS word is an Operand Pointer whose address field (2000) is the second address of the MTA 3000 instruction.

If a two-address instruction using an SAS for second-address determination also involves first-address modification through an AMS, the P-sequence AMS word (or words) precedes the P-sequence SAS word. Figure IV-22 shows a two-address instruction using both a P-sequence AMS and a P-sequence SAS word.

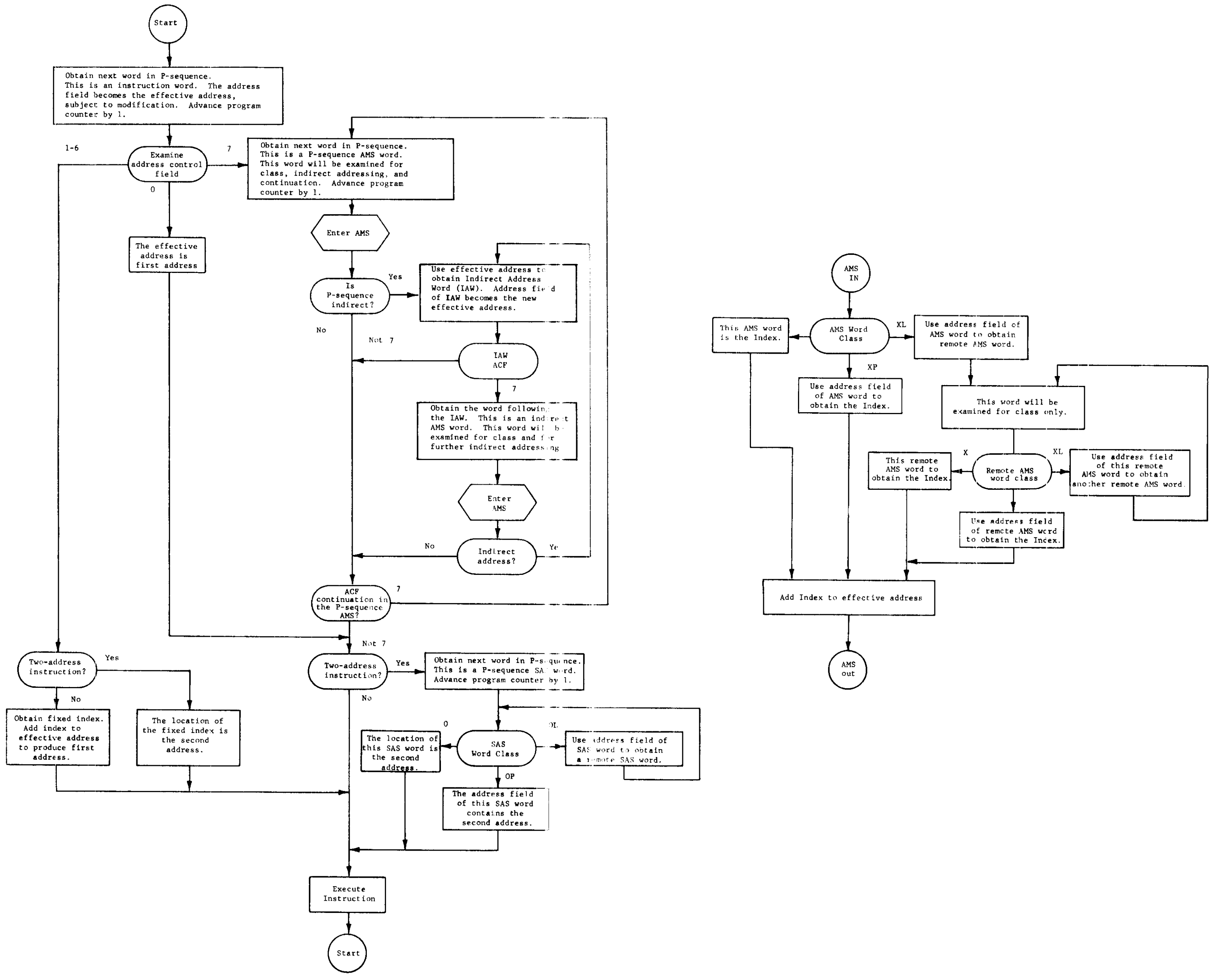


REFERENCE SYMBOL / DATA NAME (9–16)	OPERATION (17–24)	OPERATION / PICTURE (25–40)
CON11	MTA	3000, 7
	XP	2000
	OP	2500

Location	Contents				
CON11	M T A	7		$(3000)_{10}$	Instruction Word with First Address
CON11+1	0	1	0	$(2000)_{10}$	Index Pointer
2000	x	x	x	$(100)_{10}$	Modifier
CON11+2	0	1	0	$(2500)_{10}$	Operand Pointer

Figure IV-22. Use of an AMS and an SAS with the Same Instruction

The 7 in the ACF of the MTA instruction specifies an AMS and the class field of the next word in the P-sequence (location CON11+1) is examined. The P-sequence AMS word is an Index Pointer whose address field (2000) specifies the location of a modifier. The address field of this word (100) is added to the instruction address field (3000) forming a tentative first address (3100). The indirect address field and the ACF of the Index Pointer indicate that no further modification is to occur. The tentative first address, therefore, is the effective first address of the two-address instruction. The next word in the P-sequence (CON11+2) is obtained for second-address determination. This P-sequence SAS word is an Operand Pointer whose address field (2500) is the second address. The instruction is executed as if it were an MTA 3100 with 2500 as its second address.



Start
Obtain next word in P-sequence. This is an instruction word. The address field becomes the effective address, subject to modification. Advance program counter by 1.
1-6
Examine address control field
7
0
Obtain next word in P-sequence. This is a P-sequence AMS word. This word will be examined for class, indirect addressing, and continuation. Advance program counter by 1.
Enter AMS
The effective address is first address
Is P-sequence indirect?
Yes
No
Use effective address to obtain Indirect Address Word (IAW). Address field of IAW becomes the new effective address.
Not 7
IAW ACF
7
Obtain the word following the IAW. This is an indirect AMS word. This word will be examined for class and for further indirect addressing
Enter AMS
No
Indirect address?
Yes
ACF continuation in the P-sequence AMS?
7
Not 7
Two-address instruction?
Yes
No
Obtain fixed index. Add index to effective address to produce first address.
The location of the fixed index is the second address.
Two-address instruction?
Yes
No
Obtain next word in P-sequence. This is a P-sequence SAS word. Advance program counter by 1.
0
OL
The location of this SAS word is the second address.
SAS Word Class
Use address field of SAS word to obtain a remote SAS word.
OP
The address field of this SAS word contains the second address.
Execute Instruction
Start
AMS IN
This AMS word is the Index.
AMS Word Class
XL
Use address field of AMS word to obtain remote AMS word.
XP
Use address field of AMS word to obtain the Index.
This word will be examined for class only.
X
XL
This remote AMS word to obtain the Index.
Remote AMS word class
Use address field of this remote AMS word to obtain another remote AMS word.
Use address field of remote AMS word to obtain the Index.
Add Index to effective address
AMS out

Figure IV-23. GE-425/435 Addressing Flow Chart

V. THE PROCESSOR CHANNEL

The processor channel is associated with central processor operations. It permits the program to examine and use indicators which reflect specific processor conditions, such as instruction alerts, arithmetic overflows, and results of comparisons. The channel is also used during program operations - either operator-initiated or equipment-initiated interrupts that can result from various occurrences during operation. The channel operates with the Basic Input/Output System to provide for error detection, error correction, and error recovery procedures. Unlike input/output channels, the processor channel has no data control word or list pointer word. It does have a program interrupt word (PIW) and program interrupt second word (PSW) in memory locations 10 and 11.

The central processor operation instructions (octal code 67) provide several methods for affecting the central processor indicators. One instruction (Request Status of Processor, RQSP) causes the indicators to be examined and their on or off condition to be reflected in a memory word called the processor channel status word. Three Set Status instructions (SSL, SSO, and SSA) cause the processor indicators to be set or changed according to the bit pattern of a memory word. The processor channel status word can be tested and altered by the program.

The remainder of this section describes processor channel operations under the following headings:

1. Processor channel status word
2. Processor channel program interrupts
3. Processor channel indicators

PROCESSOR CHANNEL STATUS WORD

Status Word Format

The programmer can store the status of the central processor indicators in any word in memory. The word is called the central processor status word and reflects the status of the following central processor indicators (these indicators are described in detail later in this section).

Bit Position	Indicator
0	Comparison (Less <)
1	Comparison (Greater >)
2	Program Interrupt (PI)
3	First Time Indicator (FTI)
4	Overflow
5	Overflow Mode
19	Request Control
20	Instruction Alert
21	Control Word Alert

Request Status Instruction

The Request Status of Processor (RQSP) instruction stores indicator status in memory. It is used when indicator status must be preserved upon entry into a program interrupt subroutine which could alter the indicators. It is also used when programmed analysis is required to determine the cause of an interrupt to the processor channel.

Indicators which must be tested to determine the cause of a program interrupt to the processor channel are:

Overflow (when overflow mode is on)
Request Control
Instruction Alert
Control Word Alert

Setting Status

When the status word is used to restore or alter the central processor indicators, only the indicators whose status information is stored in bit positions 0 through 5 can be affected by any of the three Set Status instructions.

The three Set Status instructions restore or alter indicator status in different ways. In every case, however, the second address specified in the instruction contains the status word. The second address can be a fixed index word or a location specified by a Second Address Sequence.

The Set Status by Loading (SSL) instruction is normally used to restore the central processor indicators to the same status that they held when the last RQSP instruction placed the status word in memory. The following indicators must have their status preserved upon entry into a program interrupt subroutine and restored upon exit from the program interrupt subroutine:

Comparison
First Time
Overflow
Overflow Mode

PROCESSOR CHANNEL PROGRAM INTERRUPTS

A program interrupt is an unprogrammed transfer of control from the program being executed without alteration of the P-counter. When the processor channel requests a program interrupt, the interrupt can be caused:

1. Manually by the operator when he depresses the REQUEST CONTROL switch on the operator panel in order to intervene in a program.

2. Automatically by the hardware when error conditions such as invalid operation codes and invalid addresses are encountered.

3. Automatically when an overflow occurs and the Overflow Mode indicator is on.

Program Interrupt Words

The processor channel has two memory locations (10 and 11) assigned for channel control words which are used in accessing a processor channel interrupt subroutine. When a program interrupt occurs for any of the above-listed reasons, the processor channel control words in locations 10 and 11 of memory are automatically accessed to obtain the next instruction. Figure V-1 illustrates the format of the two processor channel control words.

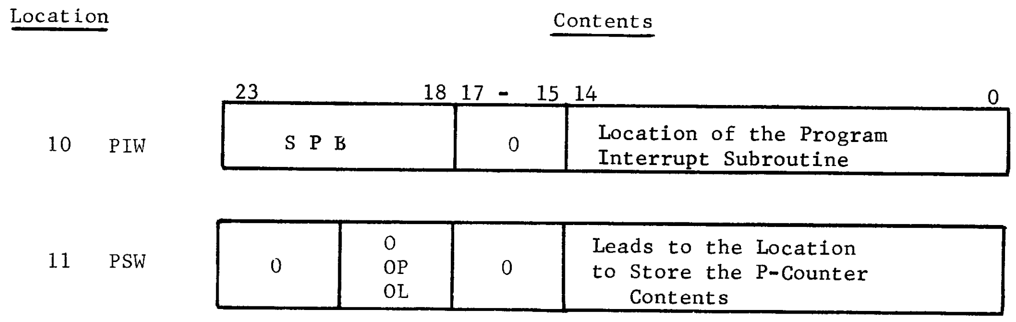

Location
Contents
23
18 17 - 15 14
0
10 PIW
S P B
0
Location of the Program Interrupt Subroutine
11 PSW
0
0 OP OL
0
Leads to the Location to Store the P-Counter Contents

Figure V-1. Format of Processor Channel Control Words

These two words function as follows:

1. Location 10 contains the processor channel program interrupt word (PIW). This word must be an SPB instruction that stores the P-counter and branches to a program interrupt subroutine. Execution of the SPB instruction in the PIW turns on the Program Interrupt indicator. The PIW cannot be indexed or address modified. The ACF must be zero. If a General instruction is in location 10, the instruction will be executed repeatedly in a perpetual loop, stalling the program. If an instruction other than an SPB or General instruction is in location 10, a control word alert will result and the computer will halt.

2. Location 11 contains the program interrupt second word (PSW). This word must be the SAS word for the SPB instruction in the PIW. It defines the location into which the P-counter contents are stored. The stored P-counter location then serves as a point of return from the subroutine to the program which was interrupted.

Program Interrupt Indicator and Interrupt Priority

The Program Interrupt indicator locks out further program interrupts, thereby permitting only one interrupt at a time. It is possible that the operator may depress the REQUEST CONTROL switch (which turns on the Request Control indicator) at the same time that another processor channel program interrupt request occurs. If this happens, only one program interrupt results.

The Program Interrupt indicator is automatically turned on when a processor channel program interrupt request is honored by execution of an SPB instruction from the channel PIW location. The indicator remains on until it is turned off by the program. While the Program Interrupt indicator is on, no other program interrupt request can be honored. Attempted program interrupts by input/output channels or by the central processor channel during the lockout will be remembered, except for an instruction alert and a control word alert which cause an error halt. Remembered interrupt requests will be serviced when the Program Interrupt indicator is turned off and when control has been returned to the interrupted program by means of a BRU instruction. Because the processor channel has the highest program interrupt priority of all channels, its requests for interrupt are honored first from those remembered. Remembered requests for interrupts from input/output channels are then taken in the order of regular channel program interrupt priority which is, in high-to-low order: 1, 2, 3, 4, 5, 6, 7, and 0. When two or more input/output channels request program interrupt simultaneously, the same channel priority is followed. The request of the highest priority channel is honored and the other requests are remembered.

The Program Interrupt indicator must be turned off before returning to the program which was interrupted. In addition, it should be turned off as soon as possible to allow recognition of other PI requests.

The method of turning off the Program Interrupt indicator varies somewhat, depending upon the method by which the indicator was turned on. If the indicator was turned on by the SPB instruction in the PIW (as is most often the case), the Set Status SSL or SSA instruction (with status word coded to turn off the Program Interrupt indicator) only <u>conditions</u> the indicator to be turned off. The actual turn-off is made by the first BRU instruction after the SSL or SSA instruction. This prevents recognition of the next program interrupt request until the BRU address field (usually the return address link to the program which was interrupted) is in the P-counter. A BRU instruction must also follow the SSL or SSA instruction when the Program Interrupt indicator is turned off during the subroutine rather than at the end of it. In this case, the BRU can simply be to the next instruction in sequence in the interrupt subroutine.

If the Program Interrupt indicator was turned on by an SSL or SSO instruction rather than by an SPB in the PIW, an SSL or SSA instruction with appropriate coding in the status word turns off the indicator without a subsequent BRU.

The Status Word and Program Interrupts

The RQSP instruction always stores a 0 in bit position 2 of the status word. When central processor indicator status is restored after a program interrupt, the Program Interrupt indicator must be turned off to permit further interrupts. The 0 in bit position 2 conditions the turning off of the indicator when either an SSL or an SSA instruction is used.

PROCESSOR CHANNEL INDICATORS

Indicators are described and various applications of indicator status are discussed in this section. Figure V-2 at the end of the section summarizes information about the indicators and their status.

Comparison Indicators (Bit Positions 0 and 1)

The Comparison indicators are set to "greater," "less," or "equal" when the various compare instructions (CAA, CDA, CMI, CXI, CMM and CXM) are executed. These indicators can also be set by the Set Status instructions. The condition of "less-greater" (not "equal") can only be set by a Set Status instruction. Four of the binary shift instructions (SRDT, SRST, RRDT, and RRST) also set the indicators. These binary shift and test instructions set the indicators to "less" if the least-significant bit position of the accumulator contains a zero after the shift is completed. These instructions set the indicators to "equal" if the least-significant bit position contains a 1. The processor uses the indicators to control conditional branching by the BRL, BRE, and BRG instructions. The "less-greater" setting (both indicators on) causes a branch to occur with either a BRL or BRG instruction. Indicators remain in a specific status until changed by:

1. A Compare instruction
2. A Shift (and test) instruction
3. A Set Status instruction.

The Comparison indicators have the following four configurations in bit positions 0 and 1 of the status word:

Bit Position 1	0	Status
0	0	Equal
0	1	Less
1	0	Greater
1	1	Less-greater

Figure V-2. Summary of Processor Channel Status Indicators

Program Interrupt Indicator (Bit Position 2)

Refer to "Processor Channel Program Interrupts" earlier in this section.

First Time Indicator (Bit Position 3)

The First Time indicator (FTI) is used during multiply and divide operations.

The FTI is turned on by:

1. Depression of the RESET COMPUTER switch.
2. The work count of a BRC instruction rolling over to zero.
3. A properly-coded SSL or SSO instruction.

The FTI is turned off by:

1. The execution of a VLM instruction.
2. The execution of a VLD instruction.
3. A properly-coded SSL or SSA instruction.

Overflow Indicator (Bit Position 4)

An overflow and the turning on of the Overflow indicator can arise from the execution or attempted execution of the following arithmetic instructions:

1. Add Decimal Instructions (ADS, ADD, ADT, or ADQ)
2. Subtract Decimal Instructions (SDS, SDD, SDT, or SDQ)
3. Add to Memory Instructions (AMS, AMD, AMT, or AMQ)
4. Add Binary to Memory Instructions (ABM or ABX)
5. Subtract Binary from Memory Instructions (SBM or SBX)
6. Variable Length Divide Instructions (VLD).

A properly-coded SSL or SSO instruction can also turn on the Overflow indicator.

The Overflow indicator is turned off:

1. Automatically when the processor channel status is stored by an RQSP instruction.
2. By a properly-coded SSA or SSL instruction.
3. By the actuation of the RESET COMPUTER switch.
4. By the execution of the first VLD instruction (FTI is on) in a division operation.

The effect of overflow on the program is determined by whether the Overflow Mode indicator is on or off. If it is <u>off</u> at the time of overflow, the overflow condition turns on the Overflow indicator and the next instruction is executed in normal sequence. If the Overflow Mode indicator is <u>on</u> at the time of an overflow condition, the overflow turns on the Overflow indicator and a program interrupt to the processor channel occurs.

Overflow Mode Indicator (Bit Position 5)

The Overflow Mode indicator is used to control the effects of overflow, as described under "Overflow Indicator" above. When the Overflow Mode indicator is on, any arithmetic overflow that occurs will cause a program interrupt. The program interrupt causes a branch to a program interrupt subroutine to correct the overflow condition.

The Overflow Mode indicator is turned on only by a properly-coded SSO or SSL instruction.

When the Overflow indicator is off, an SSL or SSO instruction coded to turn on both the Overflow indicator and the Overflow Mode indicator results in only the Overflow Mode indicator being turned on. When the Overflow indicator is off, an SSO instruction coded to turn on the Overflow indicator when the Overflow Mode indicator is already on, results in only the Overflow Mode indicator being on.

The Overflow Mode indicator is turned off:

1. Automatically when the processor status is stored by a RQSP instruction while the Program Interrupt indicator is on.

2. By a properly-coded SSA or SSL instruction.

3. By actuation of the RESET COMPUTER switch.

Request Control Indicator (Bit Position 19)

Depression of the REQUEST CONTROL switch on the operator panel turns on the Request Control indicator. This causes a request for program interrupt to the processor channel.

The Request Control indicator is turned off:

1. Automatically when the processor status is stored by an RQSP instruction.
2. By actuation of the RESET COMPUTER switch.

Unlike other indicators described thus far in this section, the Request Control indicator cannot be changed by any of the three forms of the Set Status instructions.

Instruction Alert Indicator (Bit Position 20)

The Instruction Alert indicator is turned on automatically when any of the following three conditions exist:

1. Invalid operation code. This alert occurs when any bit configuration is detected in bit positions 18 through 23 of the instruction word other than those specified in Section III, Instruction Repertoire.

2. Invalid address. This alert occurs when a memory address (bit positions 0 through 14) larger than the memory capacity of the computer is detected. The address is tested for validity as it is used to access memory. The Instruction Alert indicator is not affected by the detection of an invalid address during a Data Transfer Sequence.

3. Invalid accumulator location. This alert occurs when an accumulator location (set by an LAL instruction) larger than the memory capacity of the computer is detected. The invalidity is not detected until an attempt is made to access the accumulator.

Whenever the Instruction Alert indicator is turned on, the INSTR ALERT light on the operator panel is lit.

The effects of an instruction alert indication for the above reasons vary somewhat, depending upon the following three conditions.

1. If the PI indicator is off (no program interrupts being serviced), the current instruction is aborted and there is a program interrupt to the processor channel.

2. If the PI indicator is on (a program interrupt being serviced), the current instruction is aborted and the computer halts with the Instruction Alert indicator and the INSTR ALERT light on the console turned on.

3. If the invalid operation code was in a PIW (any channel), the instruction is aborted and the computer halts. The Instruction Alert and Control Word Alert indicators and lights are turned on. The P-counter contains the address of the next location in the program which would have been accessed if the program interrupt had not occurred.

The Instruction Alert indicator and light are turned off:

1. Automatically when the processor status is stored by an RQSP instruction.
2. By actuation of the INSTR ALERT switch on the operator panel.
3. By actuation of the RESET COMPUTER switch.

The Instruction Alert indicator cannot be changed by any of the three Set Status instructions.

Control Word Alert Indicator (Bit Position 21)

Whenever the Control Word Alert indicator is turned on, the CONTROL WORD ALERT light on the operator panel is also turned on. The Control Word Alert indicator is turned on automatically when either of the following two conditions exist:

1. When an invalid address is used in an attempt to access memory during a Data Transfer Sequence. This means that an address taken from either an LPW or DCW location for an I/O channel was larger than the memory capacity. The effects are:

   a. The Data Transfer Sequence is aborted.

   b. An "End of Data Transfer" signal is transmitted to the selected channel.

   c. If the PI indicator is off (no program interrupt being serviced), a program interrupt to the processor channel occurs.

   d. If the program interrupt indicator is on (a program interrupt is being serviced), the computer halts with the Control Word Alert indicator and the CONTROL WORD ALERT light on the operator panel turned on.

2. When an operation code other than SPB or GEN is found in the PIW location at time of program interrupt, the effects are:

   a. The instruction in the PIW is aborted.

   b. The processor halts.

   c. If the instruction has an invalid operation code, the Instruction Alert indicator and light are turned on in addition to the Control Word Alert indicator and light. The P-counter contains the address of the next location in the program which would have been executed if the program interrupt had not occurred.

The Control Word Alert indicator and lights are turned off:

1. Automatically when the processor status is stored by an RQSP instruction.
2. By actuation of the CONTROL WORD ALERT switch on the operator panel.
3. By actuation of the RESET COMPUTER switch.

The Control Word Alert indicator cannot be changed by any of the three Set Status instructions.

Processor Channel Indicators	Corresponding Status Word Bit Position	Conditions That Turn Indicator on (Excluding Set Status Instruction)	Indicator On Causes PI	Bit Stored By RQSP If Indicator On	Condition Of Indicator After RQSP	Indicator Affected By Set Status
Less	0	Compare or Shift And Test Instruction	No	1	Unchanged	Yes
Greater	1	Compare or Shift And Test Instruction	No	1	Unchanged	Yes
Program Interrupt	2	SPB in PIW for any channel	No (1)	0	Unchanged	Yes (2)
First Time	3	BRC count rolls over to zero or RESET COMPUTER switch	No	1	Unchanged	Yes
Overflow	4	Occurrence of overflow	Yes, if OFLO MODE is on (3)	1	Off	Yes (4)
Overflow Mode	5	None	No	1	Unchanged or off (5)	Yes (4)
Request Control	19	REQUEST CONTROL Switch	Yes (3)	1	Off	No
Instruction Alert	20	Invalid operation code or instruction address	Yes (3)	1	Off	No
Control Word Alert	21	Invalid Address in LPW or DCW, or no SPB or GEN in PIW	Yes (3) Only for Invalid Ad-dress in LPW or DCW	1	Off	No

1) The PI indicator is on as a <u>result</u> of a PI. While on, no further interrupts will occur but requests will be remembered.
2) PI indicator can be turned on or off by the Set Status instruction. When the PI indicator has been turned on by a PI, the Set Status to turn the PI indicator off will only condition a change to the PI indicator. The next BRU instruction will actually turn the PI indicator off.
3) PI will occur only when PI indicator is off.
4) If the OFLO indicator is off, a Set Status instruction cannot cause both the OFLO and the OFLO MODE indicators to be on.
5) Will be turned off when RQSP is executed while the PI indicator is on.

Figure V-2. Summary of Processor Channel Status Indicators

VI. INPUT/OUTPUT PROCESSING

INTRODUCTION

All peripheral operations occur through input/output channels which are part of the central processor input/output control section. As shown in Figure VI-1, as many as eight I/O channels can be provided in the I/O control section; each channel can service one peripheral subsystem.

An I/O channel provides two-way communication between a peripheral subsystem and the central processor. Each channel provides the means for:

1. Transmitting instructions from the central processor to the peripheral subsystem.

2. Conducting buffered data transfers between core memory and the peripheral subsystem.

3. Transmitting information about the peripheral subsystem operating condition and the channel operating condition to the central processor.

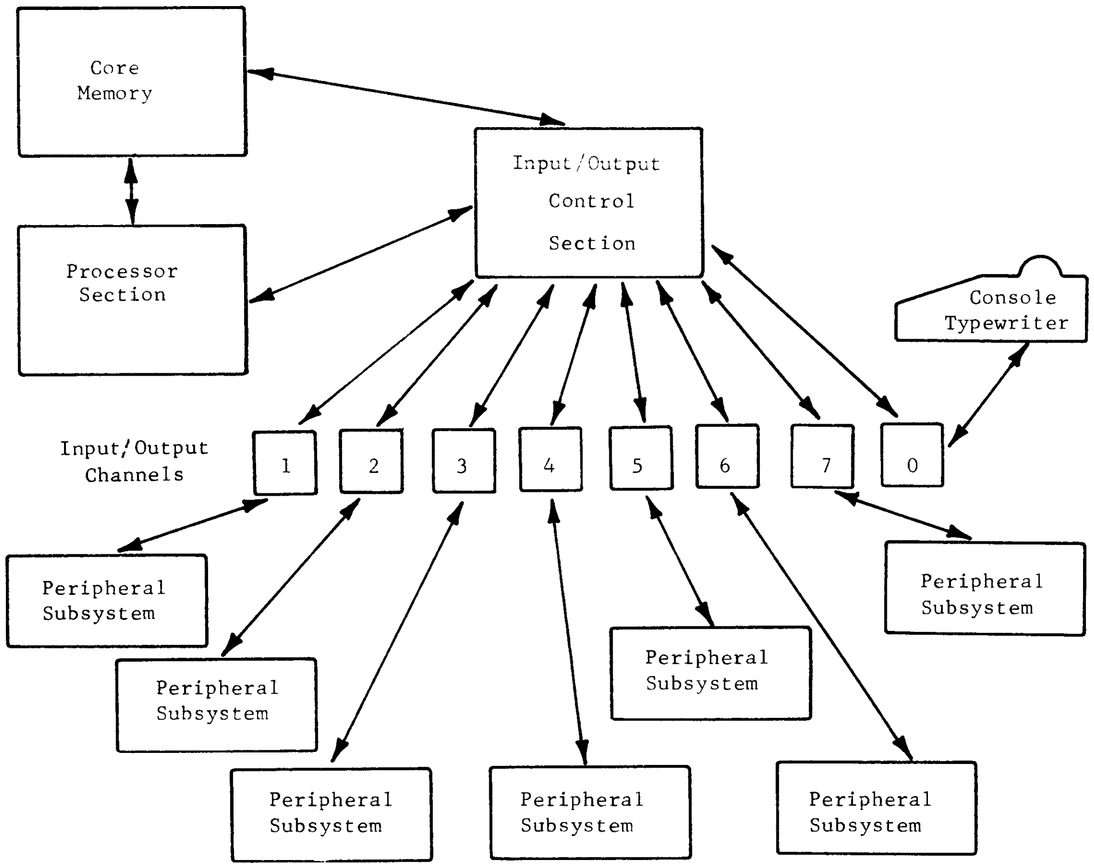


Figure VI-1. GE-425/435 Input/Output Control

In peripheral operations involving data transfers, the I/O channel provides either one character, one word, or two words of buffering between core memory and the subsystem. The amount of buffering provided depends upon the type of channel:

Character buffered channel
Word buffered channel
Double buffered word channel.

Data is moved one character at a time between memory and channel in character buffered channels, while the word channels permit data transfers one word at a time.

All data movement between a peripheral subsystem and its associated I/O channel is always one character at a time.

Each peripheral operation is initiated by the execution of a General instruction which designates:

1. The channel (and thus the peripheral subsystem)
2. The device (for multi-device subsystems)
3. The operation to be performed (read document, punch card, rewind tape, etc).

Once an operation is initiated, it proceeds under the control of the I/O channel and peripheral controller, and the central processor is free to continue processing instructions.

When a General instruction involving data transfer has been executed and the peripheral subsystem is ready for data movement to begin, the I/O channel issues a request for a Data Transfer Sequence (DTS). When a DTS is granted to the requesting channel, sequential processing of instructions stops momentarily and the processor enters a Data Transfer Sequence.

The primary purpose of a DTS is to effect a data transfer between the requesting I/O channel and core memory with a minimum of delay in program execution. The amount of data transferred during a single DTS is a function of the channel type. For character channels, one character is transferred; for single or double word channels, a full word is normally transferred. As soon as the data transfer is completed, program execution is resumed. When the channel is ready for the next data transfer, another request for a DTS is initiated. A DTS is required for each memory access involved in a given peripheral operation. Between Data Transfer Sequences, and while data is being processed within the channel and peripheral subsystem, program execution continues. Thus, internal processing occurs simultaneously with peripheral read/write operations.

During Data Transfer Sequences, memory access for data is controlled automatically by two of the channel control words for the selected channel--the list pointer word (LPW) and the data control word (DCW). These channel control words, located in memory at fixed locations for each channel, are effective <u>only</u> during peripheral operations involving data transfers. They control memory addressing and character counting during each memory access.

Data transfers between memory and a peripheral subsystem are field-oriented rather than record-oriented. Successive fields, although part of a single physical record, need not be transferred from or to consecutive memory locations.

The address and size of each field to be transferred in a given data transfer operation are specified in individual words of a field description table in memory called a data control list (DCL). Each word in a DCL specifies the number of characters in the field and the starting address in memory to or from which the field is to be transferred.

The LPW consists of a count field and an address field that are used automatically to reference the DCL and access a word from the list in order to control the transfer of a field.

During a data transfer operation, the LPW for the selected channel causes access to successive words in the DCL. When the last word of the DCL has been processed (as determined by the count field of the LPW), the data transfer operation is complete.

Upon completion of a data transfer operation by a peripheral subsystem, the related I/O channel normally initiates a program interrupt request. When the program interrupt is granted, two other channel control words, the program interrupt word (PIW) and the program interrupt second word (PSW), are accessed. The PIW and PSW can contain either an SPB instruction to effect a jump to an interrupt subroutine or a General instruction that causes the operating condition (or status) of the peripheral subsystem and channel to be stored in a specified location.

To provide for monitoring peripheral operating conditions or status, each time that a General instruction is executed a status word for the selected channel and subsystem is stored in a location specified by the second address of the instruction. By programmed interrogation of the status word, the programmer can determine whether the channel or peripheral subsystem is ready, busy, or off-line, whether an error has occurred during an operation, or whether a number of other possible conditions exist in a given subsystem. The General instruction can also be used to reset certain of these conditions as described in a later section, Operating Status.

The programming systems available with the GE-425/435 include the Basic and the Extended Input Output Systems, described in separate reference manuals. Routines made available in these two programming systems provide:

1. I/O control, including error analysis and recovery
2. Program interrupt processing
3. Record advancement and buffering.

The Basic and Extended Input Output Systems reduce the need for detailed knowledge of GE-425/435 internal I/O logic. Background information is provided in the balance of this section.

INPUT/OUTPUT CHANNELS

Peripheral input/output channels are modular units contained in the central processor, and can be installed, removed, or exchanged at the installation site. Channels can operate with either single or multiple device subsystems. Only one peripheral subsystem can be connected to a given channel. As many as eight channels of any type and configuration can be connected to the GE-425/435. Channels are designated as channel 0, channel 1, channel 2, etc., with channel 0 always reserved for the input/output typewriter.

The basic functions performed by a channel are:

1. Transfer of input/output command codes.
2. Transfer of peripheral subsystem and I/O channel status information.
3. Transfer of data.

All data transfers between a peripheral device and a channel are serial by character, while data transfers between a channel and memory can be by word or character. To perform its major functions, the channel must communicate with both the peripheral subsystem and the central processor. To satisfy the widest possible range of peripheral operational requirements, three standard types of channels are available:

Character buffered channel
Word buffered channel
Double buffered word channel

The primary difference between channels is in the amount of buffering provided between the peripheral subsystem and the central processor. A character buffered channel provides one character of buffering for data transfers in either direction, while the word buffered channels provide either one or two words of buffering for data transfers.

Data is transferred between the channel and the peripheral subsystem by character, whereas data is transferred between the channel and memory by character or by word. All data being transferred between memory and the peripheral subsystem is checked for parity.

In addition to the three standard channels, special channels (such as the magnetic reader/sorter channel) are available.

Character Buffered Channel

The character buffered channel is available for peripheral subsystems having low data transfer rates or large internal buffering capacity, or for minimal systems having no requirement for a high degree of peripheral simultaneity. This type of channel requires one memory access for each character transferred between memory and the channel.

The character buffered channel permits one character of temporary storage during reading and writing. In read operations, the channel obtains a character from the peripheral subsystem and the character must be transmitted from the channel to memory before the next character can be accepted by the channel. In write operations, the peripheral subsystem requests a character from the character channel. Then the channel must obtain the character from memory and send it to the peripheral subsystem before the subsystem can request another character.

After a data transfer operation has been initiated, the peripheral subsystem provides a busy signal to prevent the initiation of further input/output operations affecting the channel. For a peripheral read operation, events occur in this sequence:

1. The peripheral subsystem transmits one data character (and parity) to the channel. The channel stores the character, acting as a buffer at this time.

2. The channel requests a Data Transfer Sequence.

3. When the DTS request for the character channel is recognized, the character is transferred to memory under the control of the DCW for that channel.

4. When the character held in the channel is transferred to memory, the channel notifies the peripheral subsystem that the channel is ready to receive another character.

For write operations, when the peripheral subsystem is ready, it requests a data character from the channel. The channel obtains the character from memory in a sequence similar to that described for the read operation, and then signals the peripheral subsystem that the character is available for writing.

The character buffered channel should be used only with:

Perforated Tape Subsystems
Card Punch Subsystems
Printer Subsystems
Card Reader Subsystems
Input/Output Typewriter

Word Buffered Channel

One word of temporary storage is provided by the word buffered channel. In read operations, the channel receives characters from the peripheral subsystem, one at a time. The characters are placed in the single-word buffer from left to right. When the single-word buffer is filled, the channel issues a request for a DTS to the input/output control section so that the entire word can be transferred to core memory in a single memory access.

During write operations, the channel initially obtains a four-character word from memory and places it in the single-word buffer. The most-significant character in the word buffer is transferred to the peripheral subsystem when the subsystem requests a character. The remaining characters are transmitted, one at a time upon subsequent peripheral subsystem requests. When the last character is sent to the subsystem, the channel issues another request for a DTS in order to obtain the next word from memory.

For a given number of characters to be transferred, the word buffered channel requires only 25% as many accesses to memory as the character buffered channel. The word buffered channel can be used with any peripheral subsystem having a data rate of less than 10,000 characters per second, except the input/output typewriter.

Double Buffered Word Channel

The double buffered word channel provides two words (or eight characters) of temporary storage for data transfers.

In double buffered word channel read operations, as the channel receives characters from the peripheral subsystem, one at a time, the characters are placed in the first word of the two-word buffer. When the first word of the two-word buffer is filled, the channel issues a data transfer request to the input/output control section so that the data word can be transferred to memory. At this time, another word of temporary storage is available in the channel. Thus, additional characters can be received from the peripheral subsystem before a Data Transfer Sequence must occur.

During write operations, the channel initially obtains two words from core memory, a word at a time, and places them in the two-word buffer. The most-significant character of the first word is transferred to the peripheral subsystem when the peripheral subsystem requests a character. The remaining characters in the buffer are transmitted sequentially as the subsystem requests characters. When one word of the two-word buffer is empty, the channel issues a request for a DTS to obtain another word from memory. Thus, additional characters are available for transmission to the peripheral subsystem before a Data Transfer Sequence must occur.

Like the word buffered channel, the double buffered word channel requires only 25% as many accesses to memory as the character buffered channel, for a given number of characters to be transferred.

The double buffered word channel is used with any peripheral subsystem that does not contain complete internal buffering and does have a data rate faster than 10,000 characters per second, such as the magnetic tape subsystem. The channel can be used with peripheral subsystems having lower data rates, but cannot be used with the input/output typewriter.

CHANNEL CONTROL WORDS

Four sequential memory words are associated with each channel. Each set of four words consists of:

List Pointer Word (LPW) Data Control Word (DCW)	Data Transfer Control Words
Program Interrupt Word (PIW) Program Interrupt Second Word (PSW)	Program Interrupt Control Words

Memory locations 16 through 47 are assigned to these channel control words as shown in Figure VI-2.

Channel Number	Decimal Location of:			
	LPW	DCW	PIW	PSW
0	16	17	18	19
1	20	21	22	23
2	24	25	26	27
3	28	29	30	31
4	32	33	34	35
5	36	37	38	39
6	40	41	42	43
7	44	45	46	47

Figure VI-2. Channel Control Word Table

Data Control Lists (DCL)

All data transfers are performed using a data control list (DCL) which is a field description table stored in memory.

Each word except the last word in a DCL specifis the number of characters in a field and the first memory location to be used in transferring that field. This first memory location is the most-significant word of the field when the field is larger than one word.

The DCL contains one word more than the number of fields that it defines. The list can contain 2 to 513 words. The last word of the DCL is the LPW Restore word which specifies the location of the DCL (the same DCL or an alternate one) to be used in the next data transfer operation for the same channel. The LPW Restore word contains the count of the words in the next DCL and the location of its first word. When the last word of the DCL is accessed, it is placed in the channel LPW, hence the name LPW Restore.

List Pointer Word (LPW)

The list pointer word (LPW) for each channel is used automatically to access a word from the DCL, preparatory to transferring the field defined by that DCL word. As each DCL word is accessed (except the last word), it is placed in the channel DCW location.

Before I/O initiation, the LPW (see Figure VI-3) must contain the address of the first word of a DCL and the number of words in the DCL. For lists of 2 to 512 words, the count field of the LPW must contain the value 512-M, in binary, where M equals the number of words in the list. For example, if the DCL contains 3 words, the LPW count field would contain 509, in binary.

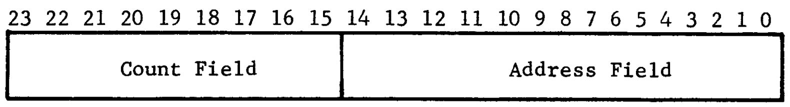


Figure VI-3. List Pointer Word Format

Upon I/O initiation, the LPW count field is incremented by one as the first word in the DCL is placed in the channel DCW location. The LPW address field is not incremented at this time. Subsequently, whenever the LPW is used to access a word from the DCL, the LPW count and address fields are both incremented by one. This permits automatic accessing of successive words in the DCL. When the LPW count field has reached 511 (in binary, 111111111) and is incremented by one, the count field "rolls over" to zero, or overflows. This is referred to as "LPW roll-over" or "LPW overflow," and signifies that the last word in the DCL has been accessed. This word is the LPW Restore word.

If an overflow occurs on the first DCL access due to an <u>initial</u> count of 511, the overflow is ignored. Thus, a DCL of 513 words will be accessed when the <u>initial</u> count field is 511.

Data Control Words (DCW)

During data transfer operations, the contents of the DCW for the selected channel determine the number of characters in the field to be transferred and the location in memory to or from which data is to be moved.

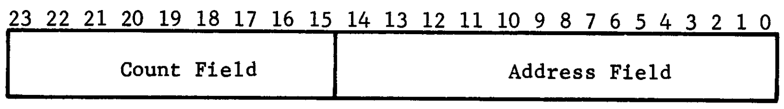


Figure VI-4. Data Control Word Format

As illustrated in Figure VI-4, the address field of the DCW initially contains the address of the memory field to be used for data transfer between the peripheral subsystem and core memory. The count field of the DCW contains the number of characters to be transferred (N), and is expressed as 512 - N, in binary.

As data characters are transferred between the channel and memory, the count field of the channel DCW is incremented by the number of data characters transferred. Normally, as every fourth character is about to be transferred, the address field is incremented by one, forming the address of the word to which these characters are to be transferred. Address field incrementation does not take place on the first Data Transfer Sequence for each DCW because the initial address in the DCW already specifies the receiving location.

Like the LPW count field, the maximum count of the DCW count field is 511 (in binary) and roll-over or overflow can occur. When the DCW count field overflows, all characters of the current field have been transferred and the next DCLword is accessed. DCW overflow causes the channel LPW to be accessed for transfer of the next DCL word to the DCW, or to the LPW if the LPW overflows. When the LPW overflows, the DCW location is cleared to zero.

At the beginning of each peripheral operation involving data transfer, the DCW is automatically loaded with the first word from a data control list (DCL). Thus, the DCW is ready to control the transfer of the first field when the first Data Transfer Sequence is initiated.

During read operations, if the number of characters specified initially by a DCW is not a multiple of four, the number of characters necessary to reduce the count to a multiple of four are transferred to the low-order character positions of the first word along with the number of leading zeros needed to fill the word. For each data character being transferred, the count of the number of characters still to be transferred is reduced by one.

To illustrate, assume that the count field of a data control word specifies that seven characters, 1234567 (count field = 505) are to be read into memory, starting at location 800. The first three characters will be placed, right-justified, in location 800 because the number of characters specified (7) is three more than a multiple of four. One leading zero is automatically inserted. The final four characters are placed in location 801. The two words containing the data field would appear as:

800	801	Memory Locations
0 1 2 3	4 5 6 7	Contents

During write operations, if the number of characters specified initially by a DCW is not a multiple of four, the number of characters necessary to reduce the count to a multiple of four are transferred from the low-order character positions of the first word. The high-order character positions of the word are ignored. As each data character is transferred, the count of the number of characters still to be transferred is reduced by one.

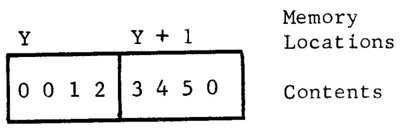
If the count field of a DCW specifies that seven-characters, 1234567 (count field = 505), are to be written from memory locations Y and Y+1, the data must have been previously placed in memory as follows:

Y	Y + 1	Memory Locations
x 1 2 3	4 5 6 7	Contents

The first three characters are obtained from the three low-order character positions of the first location (Y) because the number of characters specified (7) is three more than a multiple of four. One high-order character is ignored. The final four characters are obtained from location Y+1.

During read operations, the number of characters in the input record can be less than that specified by the data control word. In such cases, zeros are automatically placed in any remaining low-order character positions of the last word receiving input data. Any remaining words that were to have received data (according to the count and address field of the data control word) are not disturbed.

To illustrate, assume that the data control word specifies that six characters are to be read, but only five characters are received before the physical end of record is sensed. The low-order character position of the final memory location receiving data will contain a zero when data transfer is terminated:



During write operations, if the number of characters specified by the data control word exceeds the capacity of the physical device, it represents a program error. This situation is further discussed under the heading, Data Transfer Terminations.

Control Word Pseudo-Operations

The Macro Assembly Program makes available to the programmer several pseudo-operations that facilitate the generation of the list pointer word and the data control list. The mnemonic codes for the most-used two of these pseudo-operations are : LPW and DCWC.

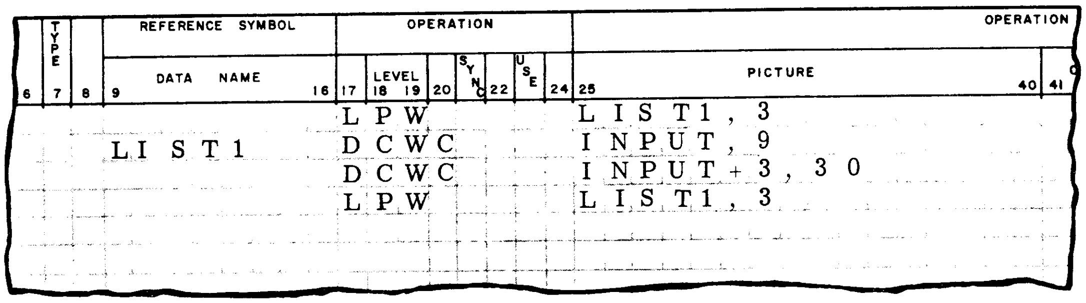
LPW, LIST POINTER WORD. The mnemonic code, LPW, enables the programmer to specify list pointer words to indicate the location and size of associated data control lists.

To generate a list pointer word, the programmer specifies the mnemonic code, LPW, in the Operation field of the coding sheet and an expression (absolute or symbolic) in the Operation Parameters field for the address, followed by another expression (absolute or symbolic) in the Operation Parameters field for the number of words in the list.

DCWC, DATA CONTROL WORD--CHARACTER COUNT. The mnemonic code, DCWC, enables the programmer to construct input and output data control lists easily by indicating in a simple form the individual data control words that comprise the lists.

To generate a word for a data control list, the programmer specifies the mnemonic code, DCWC, in the Operation field of the coding sheet. He must also include an expression (absolute or symbolic) in the Operation Parameters field for the address field of the data control word, followed by an expression (absolute or symbolic) in the Operation Parameters field for the number of characters to be specified by the data control word.

Figure VI-5 illustrates the use of LPW and DCWC pseudo-operatons.



REFERENCE SYMBOL (DATA NAME)	OPERATION	OPERATION
	LPW	LIST1, 3
LIST1	DCWC	INPUT, 9
	DCWC	INPUT+3, 30
	LPW	LIST1, 3

Figure VI-5. Use of LPW and DCWC Pseudo-Operations

The LPW on line 1 establishes the beginning location of the data control list as LIST1 and the length of the list as three words. In LIST1, a control word is generated with a count field specifying nine characters and an address field corresponding to symbolic location INPUT. In LIST1+1, another control word is generated with a count field specifying 30 characters and an address field corresponding to INPUT+3. LIST1+2 contains an LPW to generate the LPW Restore word for re-initializing the channel list pointer word.

SCATTER/GATHER OPERATIONS

The manner in which input/output control is implemented in the GE-425 automatically provides scatter-read and gather-write capabilities at the programmer's option. Whether scatter or gather capability is used is determined by the LPW and the DCL that is accessed.

A DCL consisting of only one DCW and an LPW Restore word would not allow scatter or gather during data transfer. All data transferred would be between a single field in memory and the input/output channel.

However, if the DCL consists of three or more words, then two or more of the DCW's will be used to control transfer of data, and the record can be scattered into, or gathered from, two or more separate areas in memory.

An example of scatter-read from a punched card is illustrated in Figure VI-6. The card reader is assumed to be on input/output channel 6. Memory locations 40 through 43 contain the control words for channel 6 (LPW, DCW, PIW, and PSW). The data control list, as indicated by the address field of the channel LPW, starts in memory location 2100 and consists of five words, as specified by the LPW count field. In the illustration, the count field is shown as a decimal 5; in actuality, the count field would be the binary equivalent of 507 (512-5).

Channel 6 Control Words

Location	Contents		
40	5	2100	LPW
41	xx	xxxx	DCW
42	SPB	PIRT	PIW
43	OP	EXIT	PSW

Data Control List (DCL) in Memory

Location	Contents		Card Field
2100	4	3200	A
2101	7	3300	B
2102	9	3302	C
2103	1	3317	D
2104	5	2100	LPW Restore

Channel 6 Program Interrupt Subroutine

Location	Contents
PIRT	
EXIT	

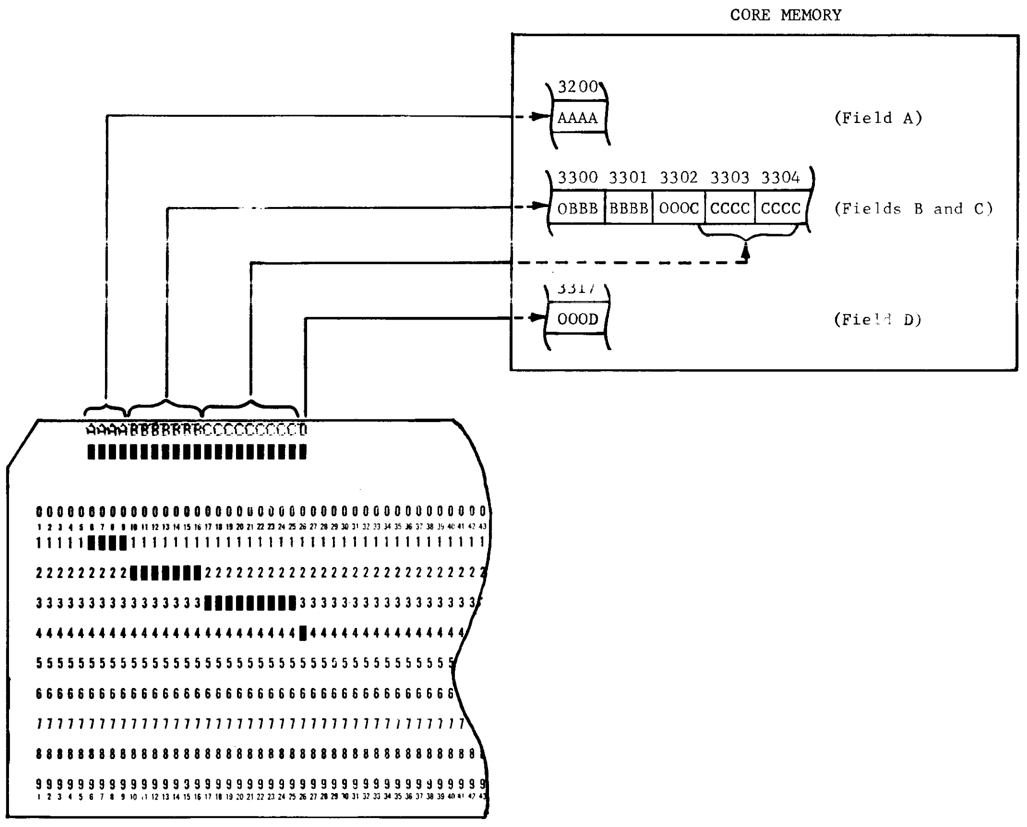


Figure VI-6. Card Scatter-Read Under Control of a Data Control List

The initial contents of the DCW in location 41 are ignored. The DCW is automatically loaded with successive words from the DCL to control the memory locations into which successive card fields are read.

The channel control words, PIW and PSW, in location 42 and 43 provide the branch to the program interrupt subroutine for channel 6. Program interrupts are discussed in the Program Interrupt section.

Each data control word of the DCL defines the length, in characters, of its related card field, and the starting location in memory into which the card field is to be read. Note that the count fields of the DCL control words are illustrated in decimal; actual counts would be the binary equivalent of 512-N, where N is the character count.

The fifth and last word of the DCL (location 2104) contains the LPW Restore word which, when the LPW rolls over to zero, is used to restore the contents of the channel LPW, leaving the LPW ready for the next input/output operation on this channel.

DATA TRANSFER TERMINATIONS

Terminations of input/output operations can be grouped into two categories:

1. Those occurring when the channel LPW count field rolls over to zero during a Data Transfer Sequence, and
2. Those occurring because the physical end of record is sensed before the LPW count field rolls over to zero, or because of detection of errors or mechanical malfunction in the peripheral device.

LPW-Controlled Terminations

A data transfer termination caused by the overflow of the count field of the LPW indicates that the desired number of characters has been transferred. When the count field overflows during a Data Transfer Sequence, the following events occur:

1. The LPW Restore word is moved into the channel LPW location, initializing the LPW for subsequent operations.
2. The channel DCW is cleared to zeros; this provides the means for program-determination that LPW overflow has occurred.
3. An end-data-transfer signal is sent to the peripheral subsystem, so that no further data transfers can occur.
4. When the physical end of record is sensed in the peripheral subsystem, the terminate signal is sent from the subsystem to the channel.

5. When the channel receives the terminate signal, a program interrupt request is initiated (see the subsection on program interrupts).

Other Data Transfer Terminations

When the physical end of record is sensed before the LPW rolls over to zero, or when a premature termination occurs because of error or mechanical malfunction, the sequence of events depends upon the type of channel and type of data transfer operation (read or write):

1. Read Operations, Word Channels.

   a. If there are data characters in the channel buffer upon termination, a special termination Data Transfer Sequence is required. If there are no data characters in the buffer, this special DTS is not required and a program interrupt request is initiated immediately.

   b. The special termination Data Transfer Sequence causes the DCW address field to be incremented by one. Characters accumulated in the channel buffer since the last Data Transfer Sequence are placed in the high-order character positions of the memory location now addressed by the DCW. Any unfilled low-order positions are cleared to zeros.

   c. The DCW count field is adjusted automatically to reflect the number of characters that are not transmitted under control of this DCW because the physical end of record was sensed. The DCW address field, at this point, specifies the memory location to which input data was stored. The channel LPW count field reflects the number of words unused in the DCL. The LPW address field specifies the location of the control word most recently moved to the DCW from the DCL.

   d. A program interrupt request is initiated.

2. Write Operations, Word Channels.

   a. The count and address fields of the LPW and DCW cannot be interpreted conclusively. However, the contents of both the LPW and the DCW will be other than zero.

   b. A program interrupt request is initiated.

3. Read and Write Operations, Character Channels.

   a. The DCW address field specifies the most-recent memory location involved in a data transfer and the DCW count field reflects the number of characters that were not transmitted under control of this DCW because the physical end of record was sensed. The channel LPW count field reflects the number of words remaining to be used in the DCL. The LPW address field specifies the location of the control word most recently moved from the DCL to the DCW.

   b. A program interrupt request is initiated.

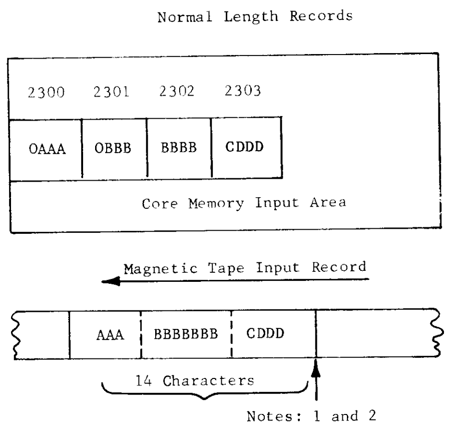
Terminations with Various Physical Record Sizes

A normal length record is a physical record containing the same number of characters as that specified in the data control list.

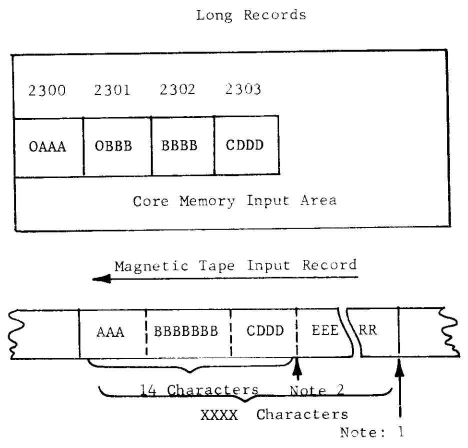
A short record is a physical record containing fewer characters than the character count specified in the DCL, or one in which termination occurs early because of error malfunction.

A long record is a physical record containing more characters than the character count specified in the DCL.

Normal Length Records



Long Records



Channel Control Words

	As Read Tape Instruction Begins			When LPW Overflow Occurs		
LPW	508		2100	508		2100
DCW	509		2300	000		2000
PIW	SPB	MBZ	ISR	SPB	MBZ	ISR
PSW	OP	MBZ	BACK	OP	MBZ	BACK

Notes: 1 = Terminate Signal by Peripheral, Causing Program Interrupt
2 = LPW Overflow Occurs

Data Control List

2100	509	2300	A
2101	505	2301	B
2102	508	2303	C D
2103	508	2100	LPW Restore

A, B, C, D: Scatter Fields, Character Count = 14

I/O Operation Does Not Change DCL

Figure VI-7. Normal Length and Long Record Terminations

Both normal length and long records result in an LPW-controlled termination. Within the program, the programmer cannot readily differentiate between normal length and long records. Figure VI-7 illustrates normal length and long record terminations.

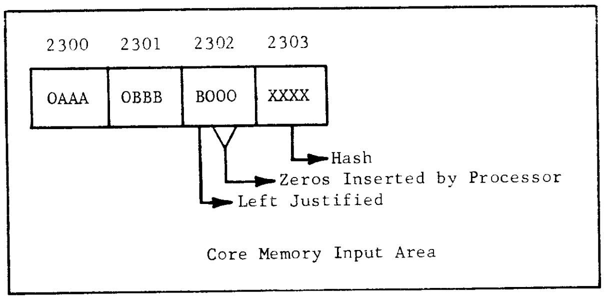
The programmer may choose to treat records as long records by cutting off data transfer after the pertinent portion of the record has been transferred, thereby eliminating meaningless Data Transfer Sequences and increasing available processing time. Such treatment also reduces memory space allocated to input/output, particularly for input operations.

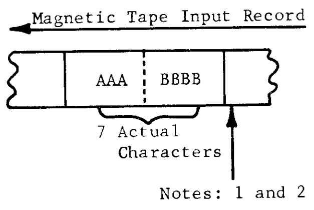
During data transfers, short records do not result in LPW overflow because the peripheral subsystem sends a terminate signal first, causing a program interrupt. The terminate signal occurs first either because the physical record is shorter than the record specified in the DCL or because an error or malfunction occurred.

During output, short record conditions can occur in the card punch, printer, and disc storage subsystems because these systems involve fixed-dimension physical records. In such cases, a short record termination represents a programmer error. Figures VI-8 and VI-9 illustrate short record terminations during input and output.

The programmer can readily determine if a termination is a short record termination rather than a long or normal length record termination by checking the DCW for zero content after termination. If the DCW does not contain zeros, a short record condition caused the termination. If errors or malfunctions caused the short record condition, these are reflected in the status word and can be determined by the program interrupt routine.

The LPW is not automatically initialized with an LPW Restore word when a short record termination occurs.





Data Control List

			Fields
2100	509 (3)	2300	A
2101	505 (7)	2301	B
2102	508 (4)	2303	C D
2103	508 (4)	2100	LPW Restore

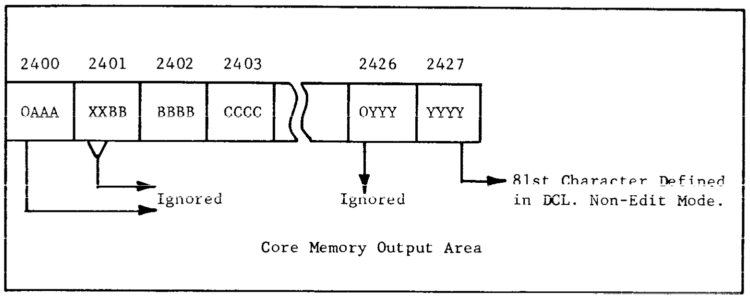
DCL Defines 14 Characters

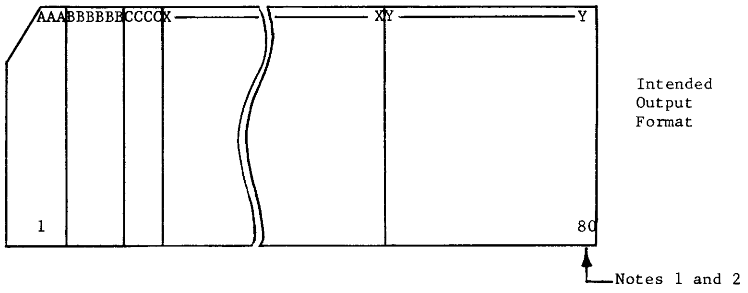
Channel Control Words

	As Read Tape Instruction Begins			When Terminate Occurs		
LPW	508 (4)		2100	(Note 3) 510(2)		(Note 4) 2101
DCW	509 (3)		2300	(Note 5) 509(3)		(Note 6) 2302
PIW	SPB	0	ISR	SPB	0	ISR
PSW	OP	0	BACK	OP	0	BACK

Notes:
1. Terminate signal by peripheral, causing Program Interrupt
2. No LPW Overflow has occurred, since DCL Defines 14 characters
3. 512-510= 2 more DCL Words not referenced
4. 2101= Address of DCL Word for field which ended short
5. 512-509= 3 characters of originally-defined field length (7) not transferred
6. 2302= Address involved in last Data Transfer for the record

Figure VI-8. Short Record Termination - - Input





Data Control List (11 words)

2100	509	2400	A
2101	506	2401	B
2102	508	2403	C
2109	505	2426	Y
2110	501	2100	LPW Restore

81 Characters

Channel Control Words

LPW	501		2100
DCW	509		2400
PIW	SPB	MBZ	ISR
PSW	OP	MBZ	BACK

As Punch Instruction Begins

Character Channel

LPW	(Note 3) 511	(Note 4) 2109
DCW	(Note 5) 511	(Note 6) 2427
PIW	SPB | 0	ISR
PSW	OP | 0	BACK

When Terminate Occurs

Word Channels

LPW	XXX		X ——— X
DCW	XXX		X ——— X
PIW	SPB	0	ISR
PSW	OP	0	BACK

Not useful, but not Zero

When Terminate Occurs

1. Terminate signal by peripheral, causing Program Interrupt
2. No LPW overflow has occurred, since DCL defines 81 characters
3. 512-511= 1 more DCL word (LPW Restore word) not referenced
4. 2109= Address of DCL word for field which ended short
5. 512-511= 1 character of originally-defined field length (7) not transferred
6. 2427= Address involved in last data transfer for the record (card)

Figure VI-9. Short Record Termination - - Punched Card Output

DATA TRANSFER PRIORITY

Because the processor and the input/output control sections of the central processor time-share the use of certain internal registers and access to core memory, the processor section can execute instructions only when there are no Data Transfer Sequences to be processed. When one or more channels issue requests for Data Transfer Sequences while the processor section is processing instructions, instruction execution continues until a point is reached when a Data Transfer Sequence can occur. At this time, control is transferred to the input/output control section, normally within one memory cycle of the data transfer request.

Channels are assigned priority for data transfer requests by means of wiring within the central processor, and can be changed by Product Service personnel when required.

When more than one channel requests a Data Transfer Sequence, the channel having the higher priority is serviced. Before any Data Transfer Sequence is completed, all channels are tested for unserviced data transfer requests. If any such requests exist, they are serviced in accordance with the established priority system before instruction processing is resumed.

Double buffered word channels have an additional high-level priority over the normal data transfer request priority. The high-level priority becomes effective in only two situations:

1. When eight characters accumulate in the channel buffer during read operation, or
2. When no character remains in the channel during write operations.

High-level requests always have priority over normal data transfer requests. Two or more simultaneous high-level requests (on two or more double buffered word channels) are serviced according to the same priority assignment applicable to normal requests.

The table in Figure VI-10 illustrates a typical assignment of priorities to various channel positions and the relationship of the normal and high-level priority assignments to the sequence in which Data Transfer requests are serviced. The table assumes that the indicated channels have requested Data Transfer Sequences before the beginning of the first Data Transfer Sequence, and that no further data transfer requests are initiated until the ones listed in the table have been serviced.

Channel Number	Normal Priority Level	Channel Type	Type of Request	Order of Service
1	7	Character	Normal	3
2	5	Word Buffered	Normal	4
3	3	Character	None	
4	1	Double Buffered Word	Normal	5
5	6	Double Buffered Word	None	
6	4	Double Buffered Word	High	1
7	2	Double Buffered Word	High	2
0	0	Character	Normal	6

The larger the number in the normal priority column, the higher the normal priority.

Figure VI-10. Table of Normal and High-Level Priority Relationships

PROGRAM INTERRUPT CONTROL WORDS

Two locations for each I/O channel are reserved in memory for the program interrupt control words. These control words are the program interrupt word (PIW) and the program interrupt second word (PSW). Together, the words contain a two-address instruction that, depending upon the instruction, determines whether program interrupt requests for the associated channel are to be honored or not.

Program Interrupt Word (PIW)

Upon completion of an input or output operation, the input/output channel requests that the central processor interrupt the instruction processing sequence. When the central processor acknowledges the request, the PIW for the requesting channel is accessed to obtain the next instruction.

Either a Request Status (RQS) or a Store P and Branch (SPB) instruction must be in the PIW, to control how the program interrupt is to be serviced. If the PIW does not contain an SPB or a General instruction, the computer halts with a control word alert. A General instruction other than an RQS in the PIW will be executed but, because it resets status, information about the last peripheral operation can be lost. Execution of a PIW instruction differs from the execution of a normal P-sequence instruction in that its ACF is ignored; there can be no address modification of the instruction in the PIW.

Program Interrupt Second Word (PSW)

The PSW contains the Second Address Sequence word for the PIW, above, and will be one of the following:

1. An Operand, causing the P-counter contents to be stored into the PSW, if the PIW contains an SPB instruction. If the PIW contains an RQS instruction, the PSW should not contain an operand.
2. An Operand Pointer that indicates the location into which:
   a. The P-counter contents are to be stored by the SPB instruction, or
   b. The peripheral subsystem status is to be stored by the RQS instruction.
3. An Operand Link leading to the second address.

PROGRAM INTERRUPT

A program interrupt was defined in the Processor Channel section as being an unprogrammed transfer of control from the program being executed, without altering the program counter. This same definition is also valid for program interrupts originating in the I/O channels. Program interrupts are requested by input/output channels for either of two reasons:

1. An operation that previously placed a channel and its associated peripheral subsystem in the busy state is terminated.
2. A special event has occurred in the peripheral subsystem.

A program interrupt request is granted (for either reason) <u>only</u> if all three of the following conditions are satisfied:

1. The Program Interrupt indicator is off (no other program interrupt request is being serviced).
2. No channel with a higher priority for program interrupt is requesting a program interrupt.
3. Execution of the current instruction is completed.

Figure VI-11 contains a flow chart illustrating program interrupt.

The effect of granting a program interrupt depends upon the instruction in the PIW.

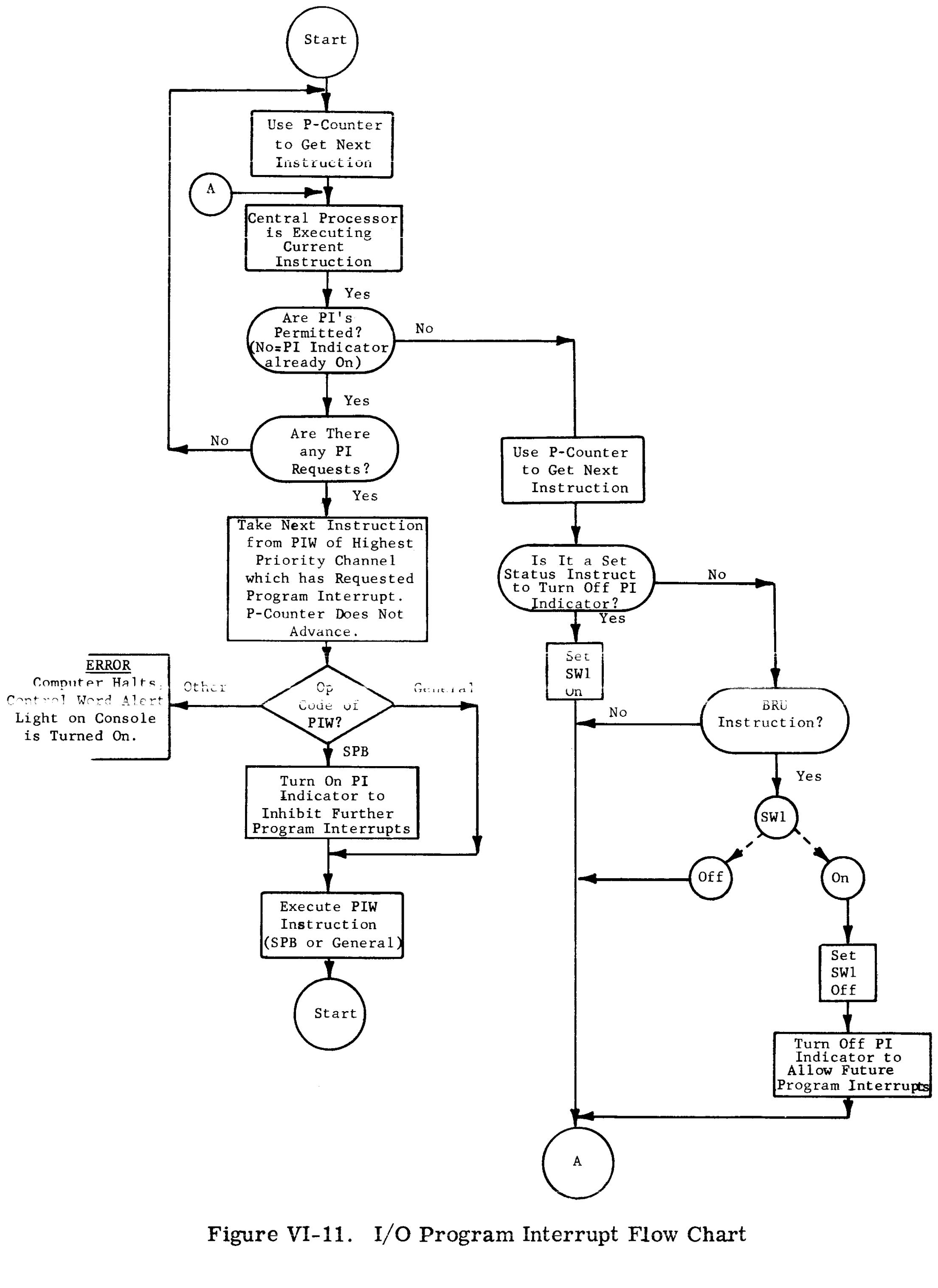

Start
Use P-Counter to Get Next Instruction
A
Central Processor is Executing Current Instruction
Yes
Are PI's Permitted? (No=PI Indicator already On)
No
Yes
No
Are There any PI Requests?
Yes
Take Next Instruction from PIW of Highest Priority Channel which has Requested Program Interrupt. P-Counter Does Not Advance.
ERROR
Computer Halts; Control Word Alert Light on Console is Turned On.
Other
Op Code of PIW?
General
SPB
Turn On PI Indicator to Inhibit Further Program Interrupts
Execute PIW Instruction (SPB or General)
Start
Use P-Counter to Get Next Instruction
Is It a Set Status Instruct to Turn Off PI Indicator?
No
Yes
Set SW1 On
No
BRU Instruction?
Yes
SW1
Off
On
Set SW1 Off
Turn Off PI Indicator to Allow Future Program Interrupts
A

Figure VI-11. I/O Program Interrupt Flow Chart

Program Interrupt Request with an SPB in the PIW

When a program interrupt request is granted, the SPB instruction stores the contents of the program counter in the location specified by the second address and replaces the P-counter contents with the first address of the SPB, effectively interrupting the program and transferring control to a routine designed to service the cause of program interrupt. Execution of the SPB instruction also turns on the Program Interrupt indicator, preventing further program interrupts from being granted until the program interrupt indicator is turned off.

The routine to which control is transferred will normally:

1. Turn off the request for program interrupt by this channel only. The first General instruction in the interrupt routine must specify the channel causing the interrupt and turns off the request by that channel. In addition, the first General instruction in the routine causes the status of the peripheral subsystem to be stored. Any subsequent General instruction in the routine will not turn off requests for program interrupt for the channel they specify.

2. Preserve the status of the central processor by executing a Request Status of Processor (RQSP) instruction, so that the processor can be returned to that status upon conclusion of the interrupt routine.

3. Service the cause of the program interrupt.

4. At the conclusion of the interrupt routine, restore the processor to the status existing immediately preceding the program interrupt by executing a Set Status by Loading (SSL) instruction using the status obtained by the RQSP described in step 2 above. Bit position 2 of the word used to set status must be zero in order to permit the BRU instruction (see step 5 below) to turn off the Program Interrupt indicator.

5. Turn off the Program Interrupt indicator and return control to the interrupted program, by executing a BRU instruction subsequent to the SSL instruction that restored central processor status.

Program Interrupt Request with an RQS in the PIW

If the program interrupt word for the channel requesting a program interrupt contains a Request Status instruction for that channel:

1. The program interrupt indicator is not turned on.

2. The status of the peripheral subsystem is stored in a location determined by the RQS instruction.

3. The next instruction is obtained under the control of the P-counter, which still refers to the program that was momentarily interrupted.

4. The request for program interrupt is turned off.

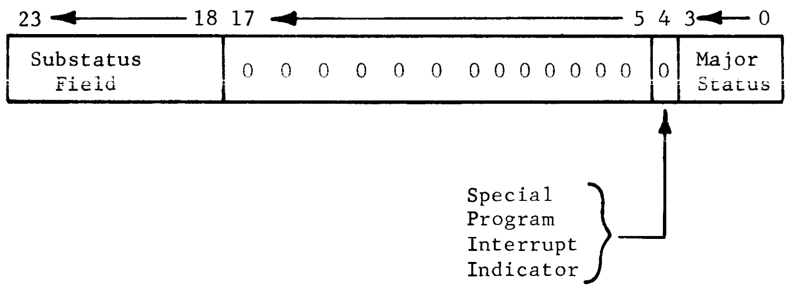
Thus, the Request Status instruction in the PIW provides a means for cancelling a request for program interrupt and continuing the processing of instructions in the main program.

However, if the RQS instruction does not specify the same channel as the one requesting the program interrupt, the request is not turned off and the RQS instruction is repetitively executed when the PIW is accessed.

Program Interrupt Caused by Termination

When a data transfer is terminated because of LPW rollover, end of physical record, error, or peripheral malfunction, the peripheral subsystem sends a terminate signal to the channel. At this time, the major status of the subsystem changes from Channel/Peripheral Subsystem Busy to Channel/Peripheral Subsystem Ready or some other major status reflecting the reason for termination. When the channel receives the terminate signal, a program interrupt request is initiated.

A Request Status instruction following a program interrupt caused by the terminate signal results in the storage of a status word as follows:

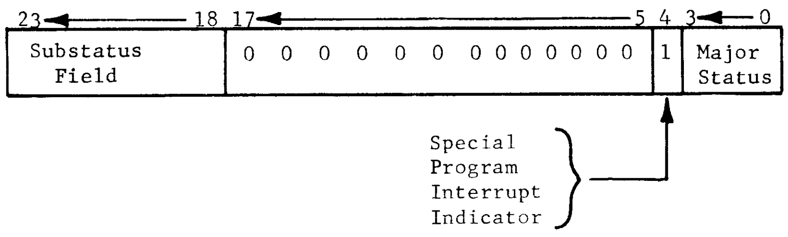


A zero in bit position 4 of the status word indicates that program interrupt was requested because of a data transfer termination. Status words are fully described under the heading, Operating Status.

Special Program Interrupt

Special events occur in some peripheral subsystems that are of interest to the program. For example, when an off-line operation such as magnetic tape rewind is complete, it may be desirable to so notify the program. Another example might be the manual correction of an Attention major status and return to Channel/Peripheral Subsystem Ready status. Such status changes cannot cause a request for program interrupt through a data transfer termination. Instead, a special interrupt signal can be sent from the peripheral subsystem to the channel under conditions dependent upon the subsystem type and the nature of the event causing the special interrupt. Conditions causing special interrupts are described in the individual peripheral subsystem reference manuals.

When a channel receives a special interrupt signal, a request for program interrupt is initiated. A Request Status instruction following the interrupt caused by the special interrupt signal resets the request for program interrupt and the special interrupt condition, and stores a status word with this format:



A one in bit position 4 of the status word indicates that program interrupt was requested because of a special event in the requesting peripheral subsystem.

It is possible that some peripheral subsystems will issue a request for special interrupt and a terminate signal at the same time, in which case two requests for interrupt will be initiated. Both requests will be honored, the terminate interrupt being serviced first.

In certain subsystems, two or more special interrupt signals can occur in rapid succession. Thus, a special program interrupt indicates that one or more special events have occurred in one peripheral subsystem.

OPERATING STATUS

When a General instruction is executed by the program, the desired input/output operation is not initiated unless both the channel and the peripheral subsystem are in a condition that permits the operation to be initiated. The operating condition of the peripheral subsystem or the channel at any given time is called the status. Each time any General instruction is executed, a word of status information is stored into core memory at a location determined by the second address of the General instruction. Subsequent examination of the status word by the program can determine if the desired operation was initiated, and, if it was not, the reason why.

The format of the status word stored into memory is shown in Figure VI-12.

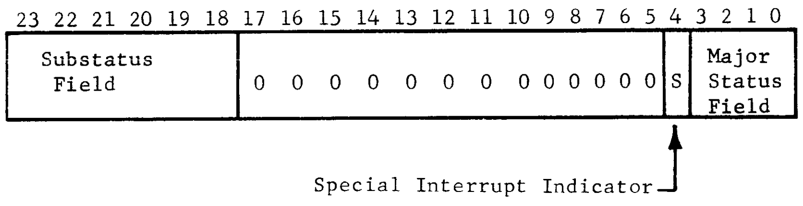


Figure VI-12. Status Word Format

One of eleven general operating conditions is indicated in the major status field of the status word. However, a given subsystem is not necessarily capable of reflecting each of the eleven major states. For most peripheral subsystems, additional information about the operating condition of the subsystem is provided in the substatus field. The special interrupt indicator in bit 4 of the status word is used to distinguish between special program interrupt and program interrupts caused by termination.

Major Status

Both the major status and the substatus information reflected upon execution of a General instruction depend upon the type of peripheral subsystem and the type of General instruction. General instructions are used for any of three different purposes:

1. To attempt the initiation of a peripheral operation and to obtain the peripheral subsystem status indicating whether the operation was initiated.
2. To request peripheral subsystem status, usually after a program interrupt request for that subsystem is granted.
3. To reset any existing resettable status in the peripheral subsystem and obtain the remaining status.

The eleven major status returns are described below. The corresponding bit configurations for the major status field of the status word are also shown. The permissible status returns in response to the three types of General instructions listed above are shown in Figure VI-13.

When status changes from Channel/Peripheral Subsystem Busy to any other status, a terminate signal occurs. The terminate signal results in a request for program interrupt in all cases, except Load Operation Complete.

Major Status	Bit Code	Description
Channel/Peripheral Subsystem Ready	0000	1. If in response to an RQS, the last operation performed was terminated successfully with no error. The channel and subsystem are ready to accept a new instruction.
		2. If in response to an RSS, the channel and subsystem are ready to accept a new instruction.
		3. If in response to any other General instruction, that instruction was accepted and the operation has been initiated or committed.

Major Status	Bit Code	Description
Device Busy	0001	1. This major status can occur only in certain multi-device subsystems, such as magnetic tape and disc storage subsystems. 2. The particular device specified in the General instruction device code is busy and cannot accept a new instruction. 3. The device is not affected by the new instruction. 4. This major status can be reflected to <u>any</u> General instruction and cannot be reset by the program.
Attention	0010	1. The peripheral subsystem cannot accept a new operation because a previous operation created an inoperable condition (card jam, feed hopper empty, stacker full, etc.) 2. This major status cannot be reset by the program; it will be reset automatically when the operator corrects the inoperable condition. 3. Attention status also occurs when the subsystem is taken from operation by depressing its manual halt or standby switch.
Data Alert*	0011	1. A data alert (such as a parity error, transmission timing error, validity error) occurred during the last data transfer operation with this subsystem. The data transfer operation continued to completion. 2. Data Alert status can be indicated only in response to an RQS. All other valid General instructions automatically reset this status.

*Reset (cleared) by initial portion of any form of General instruction except Request Status. In multi-device type subsystems, this status reflects the last device active, but is remembered by the controller for the subsystem as a whole, <u>not by each individual device.</u> Programmer must not issue an initiate or Reset status type of General instruction, or he will lose this type of status pertaining to the last active device.

Major Status	Bit Code	Description
End of File*	0100	1. An end-of-file record was detected by the device during its last operation. 2. This status is reset by _any_ valid General instruction _except_ RQS. End of File status is detected only by RQS.
Command Rejected	0101	1. The General instruction was rejected because of invalid command code, invalid device code, parity error on command code or device code, or command code inconsistent with device status. 2. Subsystem status is not changed or reset by the rejected command. 3. This status does not actually represent peripheral status and therefore would not be reflected to any subsequent General instruction accepted by the subsystem.
Intermediate	0110	1. The subsystem is not presently busy, but is waiting for another instruction to continue a sequence that was started by a prior instruction. Examples: card punch ready to receive next Punch instruction in a sequence of instructions, or magnetic reader/sorter feeding a document and awaiting a pocket decision. 2. This status can occur in card punch, magnetic reader/sorter, and disc storage subsystems. 3. This status can be detected _only_ by RQS or RSS. Any other General instruction resets this status.

*Reset (cleared) by initial portion of any form of General instruction except Request Status. In multi-device type subsystems, this status reflects the last device active, but is remembered by the controller for the subsystem as a whole, _not by each individual device._ Programmer must not issue an initiate or Reset status type of General instruction, or he will lose this type of status pertaining to the last active device.

Major Status	Bit Code	Description
Load Operation Complete*	0111	1. A device on the subsystem has successfully read one physical record as a result of the operator depressing the LOAD switch. 2. The program cannot initiate an operation resulting in this status. 3. This status is reflected <u>only</u> to an RQS. Any other General instruction resets this status. 4. Successful completion of the load operation caused a terminate signal after which this status is reflected. No request for program interrupt is made.
Channel/Peripheral Subsystem Busy	1000	1. A data transfer operation is in progress between memory and the subsystem and no new data transfer operation can begin. 2. Channel or subsystem status is reflected, but <u>device</u> status cannot be obtained. 3. If a General instruction with correct <u>channel</u> coding, but invalid device coding, is attempted and channel or subsystem is busy, then this is the status reflected, rather than Command Rejected. Busy status has a higher priority than Command Rejected.
Peripheral Subsystem Absent or Off-Line	1001	The General instruction was coded to address a channel in which: the subsystem is disconnected or without power, or the selected channel is not installed.

*Reset (cleared) by initial portion of any form of General instruction except Request Status. In multi-device type subsystems, this status reflects the last device active, but is remembered by the controller for the subsystem as a whole, <u>not by each individual device</u>. Programmer must not issue an initiate or Reset status type of General instruction, or he will lose this type of status pertaining to the last active device.

Major Status	Bit Code	Description
Channel Error	1010	1. During the last data transfer operation (input only) a character with incorrect parity was received in the channel from the peripheral subsystem. 2. The incorrect character was stored in memory. 3. This status is reset after it has been indicated in response to any General instruction. 4. The data transfer operation continued to completion.

Major Status	Response to Request Status?	Response to Reset Status?	Response to any other General Instruction ?	Any General Instruction Resets Except Request Status ?	Does Operator Reset ?	Common to all Subsystems ?	May have Substatus? (1)	Response to Request Status in Program Interrupt Routine after Program Interrupt	
								If no Error Occurred	If Error Occurred
Channel/Sub-system Ready 0000	Yes	Yes	Yes	No	No	Yes (6)	Yes	Yes	No
Device Busy 0001	Yes	Yes	Yes	No	No	No (4)	Yes		
Attention 0010	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes
Data Alert 0011	Yes	No	No	Yes	No	Yes	Yes	No	Yes
End of File 0100	Yes	No	No	Yes	No	No	Yes	Yes	No
Command Rejected 0101	Yes (2)	Yes (2)	Yes	No (2)	No	Yes	Yes	Yes (The RQS)	Yes (The RQS)
Intermediate 0110	Yes	Yes	No	Yes (3)	No	No	No	Yes	No
Load Operation Complete 0111	Yes	No	No	Yes	No	No	No (5)	No (No PI)	No (No PI)
Channel/Sub-system Busy 1000	Yes	Yes	Yes	No	No	Yes	No		
Subsystem Absent/Off-line 1001	Yes	Yes	Yes	No	Possibly	Yes	No		
Channel Error 1010	Yes	Yes	Yes	Yes (7)	No	Input Only	No	No	Yes

Notes:
1. Depends on type of subsystem.
2. A General instruction causing this status has not changed existing status of the subsystem or channel.
3. Reset by a General instruction other than Request or Reset Status.
4. Only on magnetic tape and disc storage (multi-device) subsystems.
5. Data Alert substatus will be present instead, if Load not completed successfully, Attention is also possible.
6. This is only status indicating successful initiation of operation code in General instruction.
7. Reset after indication in response to any General instruction.

Figure VI-13.

Summary of Basic Relation Between General Instruction and Major Status Conditions

Substatus

The type of substatus information reflected in the status word is contingent upon the peripheral subsystem and the major status. For example, during an attempt to initiate a card read operation, if the major status returned is Attention (requires operator attention), some additional information is needed to pinpoint the reason that operator intervention is required: Is the output hopper full; did cards fail to feed; did a card jam occur; etc? The substatus identifies the specific condition, or conditions if more than one. Substatus is reflected in bits 18 through 23 of the status word.

Some major status returns require no elaboration. If the channel and card reader subsystem are ready to read cards, for example, then no other status information is required, and the desired operation is initiated upon execution of the General instruction.

Detailed descriptions of substatus information for each peripheral subsystem are contained in the individual peripheral subsystem reference manuals.

Status Response to I/O Initiation

When a General instruction is executed, one of three situations always exists:

1. The channel and peripheral subsystem can already be busy. If so, then a new operation cannot usually be initiated; the exceptions are certain magnetic reader/sorter operations, and are explained in the magnetic reader/sorter reference manual. The status word stored in memory will indicate Channel/Peripheral Subsystem Busy major status with no substatus information.

2. The channel and peripheral subsystem are not busy, but the peripheral subsystem for some reason cannot initiate the desired operation. The status word stored in memory will indicate the major status reason for inability to initiate and, when applicable, the appropriate substatus.

3. The channel and peripheral subsystem are not busy, and the peripheral subsystem initiates the operation. The status word stored in memory will indicate Channel/ Peripheral Subsystem Ready major status with no substatus information, except for certain magnetic tape and perforated tape conditions.

If the third situation is the case and if the operation is one involving data transfer between the peripheral subsystem and memory, the General instruction causes the first word of the data control list to be moved into the DCW for the involved channel by using the address field of the channel LPW.

Major status returns that are reset during an attempt to initiate a peripheral operation are:

End of File

Data Alert

Load Operation Complete

Intermediate

Channel Error - Reset after it has been indicated in response to the initiate instruction.

Status Resettable by RSS Instruction

Execution of a Reset Status instruction causes any of the major status returns listed below to be reset. The status reflected after resetting is stored in the RSS second location.

End of File

Data Alert

Load Operation Complete

Channel Error - Reset after it has been indicated.

Status Resettable by RQS Instruction

The Request Status instruction cannot reset any major status return from either the channel or the peripheral subsystem. Channel Error major status, however, is reset after it has been indicated in response to an RQS instruction.

VII. CONTROL CONSOLE

GENERAL DESCRIPTION

The control console for the GE-425 is illustrated in Figure VII-1; the control console for the GE-435 is similar. Information in this section is applicable to both the GE-425 and GE-435 consoles.

Major features of the control console are the operator panel, the maintenance panel, and the input/output typewriter. The typewriter is connected to a standard character buffered channel which transmits either one alphanumeric or two octal characters at a time to the central processor.

This section includes functional descriptions of the operator panel and the typewriter, as well as console operating procedures. Normal operator maintenance procedures, such as changing the typewriter ribbon and paper, are covered in operator training. Programming procedures for typewriter input and output are in Section VIII, Typewriter Input/Output Programming.

OPERATOR PANEL

The operator panel (see Figure VII-2) provides the means to control and observe computer operation. Switches control computer power, load programs, reset alert conditions, and halt the computer. Many of the pushbutton switches are back-lighted to glow under certain conditions and thereby display the status of the computer. The lights and switches are color coded:

1. Red indicates that there is an alert or malfunction.
2. Amber indicates caution, power is on, etc.
3. Blue indicates that operator intervention is necessary because of a condition other than an error.
4. Green indicates a ready condition.
5. White indicates general information.

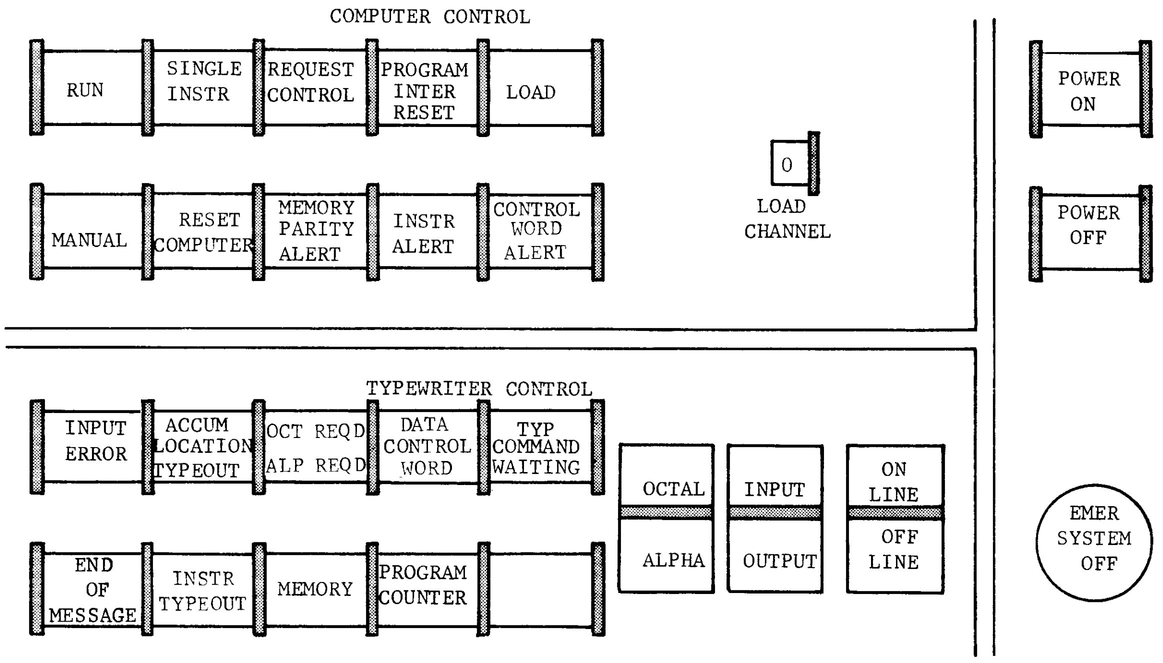


Figure VII-2. Control Console Operator Panel

At the upper left of the panel are the switches and lights that pertain to computer control. At the lower left are the switches and lights for typewriter control. At the extreme right are switches and lights controlling computer power. Most of the switches are pushbuttons; other types are indicated in the switch descriptions.

COMPUTER CONTROL SWITCHES

Computer control switches affect the operation of the central processor by establishing operational modes, controlling program loading, or initiating certain computer actions. The back-lighted switches display the status of computer operation.

RUN (Green)

Light off indicates that instruction processing has ceased. Light on indicates that:

The central processor is continuously executing instructions
The central processor is in the Run mode.

Actuation of the switch in the Manual mode:

Turns off the LOAD light, if it is on
Turns off the MANUAL light
Turns on the RUN light
Causes continuous sequential processing of instructions
Sets the central processor to the Run mode.

Actuation of the switch in the Run mode has no effect.

Actuation of the switch when a manual I/O operation is in process has no effect.

MANUAL (Blue)

Light off indicates that the central processor is <u>not</u> in the Manual mode.

Light on indicates that:

Instruction processing has ceased

Programmed data transfers have been completed

Requests for program interrupt will be delayed until actuation of the SINGLE INSTR (Single instruction) switch or the RUN switch

Central processor is in the Manual mode.

Actuation of the switch in Manual mode has no effect.

Actuation of the switch in the Run mode:

Causes cessation of instruction processing

Inhibits servicing of requests for program interrupts

Allows continued servicing of data transfer requests (when all data transfer requests have been serviced, the MANUAL light is turned on)

Conditions the computer for console I/O procedures

Turns off the RUN light

Sets the central processor to the Manual mode.

SINGLE INSTR (Single Instruction, White)

No light is associated with this switch.

Actuation of the switch in the Manual mode:

Turns off the LOAD light, if it is on

Causes only the next instruction in the program sequence to be executed (If a program interrupt request is present when the switch is actuated, the next instruction to be executed is taken from the PIW for the channel requesting the program interrupt)

Causes the central processor to enter the Run mode to execute one instruction and wait until all data transfers are completed, then return to the Manual mode.

Actuation of the switch in the Run mode has the same effect as actuation of the MANUAL switch in the Run mode, above.

Actuation of the switch when a manual I/O operation is in process has no effect.

RESET COMPUTER (White)

No light is associated with this switch.

Actuation of this switch in Manual mode turns off the LOAD light, if it is on.

Certain computer indicators are conditioned:

First Time indicator is turned on
Overflow indicator is turned off

Overflow Mode indicator is turned off
Instruction Alert indicator is turned off
Control Word Alert indicator is turned off
Program Interrupt indicator is turned off
Request Control indicator is turned off.

All requests for program interrupts are turned off.

MEMORY PARITY ALERT light is turned off.

Turns off any typewriter control conditions and lights that may have been set up by actuation of the following console switches:

MEMORY
DATA CONTROL WORD
PROGRAM COUNTER
INSTR (Instruction) TYPEOUT
ACCUM (Accumulator) LOCATION TYPEOUT

Clears memory location 8 to zeros (this location is used by the central processor for temporary storage during program execution).

Actuation of this switch in the Run mode has no effect.

REQUEST CONTROL (White)

No light is associated with this switch.

Actuation of the switch in the Manual mode:

Turns on the Request Control indicator
Causes a request for program interrupt by the processor channel

NOTE: If the processor is servicing another program interrupt when the switch is actuated, the Request Control interrupt is automatically delayed until completion of the interrupt routine in progress.

Actuation of the switch in the Run mode has the same effect as in the Manual mode.

MEMORY PARITY ALERT (Red)

Light off indicates that no memory parity alert exists.

Light on indicates that a memory parity error was detected during a data transfer from memory.

Actuation of the switch in the Manual mode causes the MEMORY PARITY ALERT light to be turned off.

Actuation of the switch in the Run mode causes unpredictable results.

> NOTE: Detection of a memory parity error causes the processor to halt.

PROGRAM INTER (Interrupt) RESET (White)

Light off indicates that:

The Program Interrupt indicator is off
The central processor is not in the Program Interrupt mode.

Light on indicates that:

The Program Interrupt indicator is on
The central processor is in the Program Interrupt mode.

> NOTE: When the processor is in the Program Interrupt mode, other requests for program interrupt are automatically delayed until completion of the interrupt routine in progress.

Actuation of the switch in the Manual mode:

Causes the computer to leave the Program Interrupt mode
Causes the Program Interrupt indicator to be turned off
Turns off the PROGRAM INTER RESET light.

Actuation of the switch in the Run mode has no effect.

INSTR (Instruction) ALERT (Red)

Light off indicates that:

No instruction alert exists
The Instruction Alert indicator is off.

Light on indicates that:

An invalid operation code was detected in the last instruction or an invalid address was detected (except during a Data Transfer Sequence)

The Instruction Alert indicator is on.

Actuation of the switch in the Manual mode causes the INSTR ALERT light and indicator to be turned off.

Actuation of the switch in the Run mode causes unpredictable results.

LOAD (Green)

Light off indicates that:

If the previous operation was a Load operation, it was not successful; <u>or</u>

The RUN, SINGLE INSTR, or RESET COMPUTER switch has been actuated since the last Load operation.

Light on indicates that:

The last operation was a Load operation which terminated with no errors being detected

The Program Load Termination major status (0111) is available to the I/O channel specified by the LOAD CHANNEL switch.

Actuation of the switch in the Manual mode:

Establishes the Load mode

Begins data transfer from the channel specified by the LOAD CHANNEL switch

NOTE: The first four characters received by the channel are used as a control word and are placed in the channel DCW location. At the same time that the control word is being placed into the DCW of the selected channel, the address field of this control word is also placed into the P-counter.

The count field of the control word should be able to control the entire record without rolling over to zero. To accomplish this, the programmer should place zeros in the count field of the control word. Termination of the loading operation is caused by the detection of the physical end of record by the peripheral subsystem, and no program interrupt is requested.

Establishes the Program Load Termination major status and turns on the LOAD light, if no errors were detected.

Actuation of the switch in the Run mode has no effect.

CONTROL WORD ALERT (Red)

Light off indicates that:

No control word alert exists
The Control Word Alert indicator is off.

Light on indicates that:

An invalid address during a Data Transfer Sequence was detected or an operation code other than SPB or a General instruction was detected in a PIW location

The Control Word Alert indicator is on.

Actuation of the switch in the Manual mode causes the CONTROL WORD ALERT light and indicator to be turned off.

Actuation of the switch in the Run mode causes unpredictable results.

LOAD CHANNEL (Black with white numbers)

No light is associated with this switch.

Selects any one of eight I/O channels, to which the signal from the LOAD switch is sent.

POWER SWITCHES

Three switches at the right of the operator panel control power for the GE-425/435 system. Two switches are for routine use, while the third is for emergency use only.

POWER ON (Amber)

Light off indicates that the central processor power is not on.

Light on indicates that the central processor power is on.

Actuation of the switch when central processor power is not on causes the power supply to be sequenced on. When power supply sequencing is complete, the receivers at all peripherals are left in a reset state and the POWER ON light is illuminated.

Actuation of the switch when central processor power is on has no effect.

POWER OFF (White)

No light is associated with this switch.

Actuation of the switch when central processor power is on:

Causes central processor power to be turned off
Causes the POWER ON light to be turned off.

EMER (Emergency) SYSTEM OFF (Red)

No light is associated with this switch.

Actuation of the switch disconnects all power to the computer system, peripherals, and any air conditioning provided for the system.

TYPEWRITER CONTROL SWITCHES AND LIGHTS

The typewriter control switches are used to control operation of the input/output typewriter. The lights are used to signal the operator that typewriter operation is to follow or that typewriter operation is in progress.

INPUT ERROR (White)

No light is associated with this switch.

When a programmed type-in is being performed, actuation of this switch causes:

The cessation of data transfer requests

Typewriter I/O logic to be reset

Data Alert major status to be indicated in response to the next General instruction to the typewriter

A program interrupt to be requested by the typewriter channel

The OCT REQD/ALP REQD light to be turned off

NOTE: The operator may actuate the INPUT ERROR switch when he recognizes that he has made a typing error. The program interrupt subroutine is assumed to request status. Upon detection of the input error condition (Data Alert status), the typewriter channel LPW can be re-initialized by the program. The subroutine can then reissue the General instruction to read, permitting the operator to accomplish the correct type-in.

Actuation of the switch during a programmed typeout has no effect.

When the computer is halted and a manual typewriter input operation is being performed, actuation of the INPUT ERROR switch causes:

The cessation of data transfer requests
Typewriter input/output logic to be reset (no program interrupt is requested)
Any of the following lights that are on to be turned off:

OCT REQD/ALD REQD
MEMORY
DATA CONTROL WORD
PROGRAM COUNTER

NOTE: The operator can actuate the INPUT ERROR switch to terminate a manual typewriter input operation.

Actuation of the switch during a manual typeout has no effect.

END OF MESSAGE (White)

No light is associated with this switch.

When the typewriter is engaged in a programmed input or output operation, actuation of the END OF MESSAGE switch causes:

The cessation of data transfer requests
Typewriter I/O logic to be reset
A program interrupt to be requested by the typewriter channel
Ready major status to be indicated, if the data transfer was successful
The OCT REQD/ALP REQD light to be turned off, if it is on

NOTE: The operator can actuate the END OF MESSAGE switch when he wishes to terminate a programmed typewriter I/O operation.

When the computer is halted and a manual typewriter input or output operation is being performed, actuation of the END OF MESSAGE switch causes:

The cessation of data transfer requests
Typewriter I/O logic to be reset (no program interrupt is requested)
Any of the following lights that may be on to be turned off:

OCT REQD/ALP REQD
MEMORY
DATA CONTROL WORD
PROGRAM COUNTER

NOTE: The operator can actuate the END OF MESSAGE switch to terminate a manual typewriter I/O operation.

ACCUM (Accumulator) LOCATION TYPEOUT (White)

No light is associated with this switch.

Actuation of the switch in the Manual mode causes the current location of the most-significant word of the working accumulator to be typed (the typed format depends upon the setting of the OCTAL/ALPHA switch).

> NOTE: Octal typeout is recommended. The three most-significant characters of the octal word are not significant and are always zeros. The five least-significant octal characters specify the accumulator location.

Actuation of the switch in the Run mode has no effect.

INSTR (Instruction) TYPEOUT (White)

No light is associated with this switch.

Actuation of the switch in the Manual mode causes a typeout as follows:

> If the computer halts due to the execution of a Halt instruction, the Halt instruction with its modified address is typed.
>
> If a computer error halt occurs (see note below), or the computer is halted by actuation of the MANUAL or SINGLE INSTR switch, the next instruction to be executed (with its unmodified address) is typed.
>
> > NOTE: When an error halt occurs, the instruction under execution may not have been completely executed; therefore, the P-counter does not necessarily contain the address of the next instruction, but may contain the address of an AMS or SAS word. Thus, the word typed by the depression of the INSTR TYPEOUT switch will be the next word in the P-sequence.
> >
> > If a request for program interrupt occurred during the execution of the last instruction, the next instruction to be executed will be taken from the PIW for the channel requesting the interrupt.
>
> The typeout format depends upon the OCTAL/ALPHA switch setting. The INPUT/OUTPUT switch has no effect when the INSTR TYPEOUT switch is actuated. The instruction typeout logic is reset automatically when eight octal or four alpha characters have been typed.

Actuation of the switch in the Run mode has no effect.

OCT REQD/ALP REQD (Octal Required/Alpha Required, Blue)

This light has two fields labeled OCT REQD and ALP REQD. There is no switch associated with these lights.

The OCT REQD field is illuminated when the typewriter has been conditioned for an octal type-in

The ALP REQD field is illuminated when the typewriter has been conditioned for an alphanumeric type-in.

For a programmed type-in, the command code of the General instruction specifies whether octal or alpha format is to be used. When the instruction is executed, the appropriate field of the light is illuminated.

The light turns off and the programmed type-in ceases when:

The typewriter channel LPW overflows
The END OF MESSAGE switch is actuated
The INPUT ERROR switch is actuated.

The operator can manually condition the typewriter for either octal or alpha type-in as follows:

Halt the computer, if it is not already halted

Set the INPUT/OUTPUT switch to INPUT

Set the OCTAL/ALPHA switch to the desired position

Actuate either the MEMORY, DATA CONTROL WORD, or PROGRAM COUNTER switch. When one of the three switches is actuated, the field of the OCT REQD/ALP REQD light that corresponds to the OCTAL/ALPHA switch setting is illuminated.

During a manual type-in, the light turns off for the following reasons:

The typewriter channel DCW count overflows when typing into memory under control of the MEMORY switch

During an octal type-in to the P-counter or typewriter channel DCW under control of the PROGRAM COUNTER or DATA CONTROL WORD switch, the light turns off after the ninth character is typed. The ninth character is the formatting character which is not transmitted to the central processor

During an alpha type-in to the P-counter or typewriter channel DCW under control of the PROGRAM COUNTER or DATA CONTROL WORD switch, the light turns off after the fourth alpha character is typed.

Actuation of the END OF MESSAGE switch

Actuation of the INPUT ERROR switch

Actuation of the RESET COMPUTER switch.

MEMORY (White)

Light off indicates that information is not being, or is not permitted to be, transferred between the typewriter and the computer memory under control of the MEMORY switch.

Light on indicates that:

The switch has been actuated
The transfer of data between the typewriter and the computer memory has not been completed.

Actauation of the switch in the Manual mode:

Causes or permits data transfer between the typewriter and the computer memory under control of the typewriter channel DCW

NOTE: The count field of the DCW determines the number of characters to be transferred. A maximum of 512 alphanumeric characters can be transferred if the DCW count is zero at the start

Causes the direction of data transfer to be controlled by the INPUT/OUTPUT switch setting

Causes the format to be controlled by the OCTAL/ALPHA switch

NOTE: The data transfer will cease, I/O logic will be reset, and the MEMORY light will turn off when:

The DCW count overflows, or

Any of the following switches are actuated:

END OF MESSAGE
INPUT ERROR (input operations only)
RESET COMPUTER.

When any of the above conditions occur, there is no program interrupt and no automatic manipulation of the typewriter channel LPW. The capability is thus provided for communication with memory without jeopardizing channel controls established by and for the operating program.

Actuation of the switch in the Run mode has no effect.

DATA CONTROL WORD (White)

Light off indicates that information is not being transferred between the typewriter and the typewriter channel DCW under control of the DCW switch.

Light on indicates that:

The switch has been actuated
The transfer of data between the typewriter and the typewriter DCW has not been completed.

Actuation of the switch in the Manual mode:

Causes or permits the transfer of eight octal or four alpha characters between the typewriter and the typewriter channel DCW

Causes the direction of data transfer to be controlled by the INPUT/OUTPUT switch setting

Causes the data format to be controlled by the OCTAL/ALPHA switch setting

NOTE: If the INPUT/OUTPUT switch is set to INPUT, both the PROGRAM COUNTER and the DATA CONTROL WORD switches can be actuated. Subsequent type-in causes data transfer to both the P-counter and the typewriter DCW.

The data transfer sequences will cease, the I/O logic will be reset, and the DATA CONTROL WORD light will be turned off when:

Eight octal characters have been transferred, <u>or</u>
Four alpha characters have been transferred, <u>or</u>
Any of the following switches are actuated:

END OF MESSAGE
INPUT ERROR (input operations only)
RESET COMPUTER.

When typing octal input, the fourth Data Transfer Sequence (for the seventh and eighth octal characters) occurs when the ninth formatting character is typed. Two octal characters are transferred during each Data Transfer Sequence.

Actuation of the switch in the Run mode has no effect.

PROGRAM COUNTER (White)

Light off indicates that information is not being transferred between the typewriter and the P-counter under control of the PROGRAM COUNTER switch.

Light on indicates that:

The switch has been actuated
The transfer of data between the typewriter and the P-counter has not been completed.

Actuation of the switch in the manual mode:

Causes or permits the transfer of data between the typewriter and the P-counter

Turns on the PROGRAM COUNTER light. The light remains on as long as information is being transferred between the typewriter and the P-counter

Causes the direction of data transfer to be controlled by the INPUT/OUTPUT switch setting

NOTE: When the INPUT/OUTPUT switch is in the INPUT position and the PROGRAM COUNTER switch is actuated while the computer is halted, information can be typed to the P-counter. Eight octal or four alpha characters must be typed. The type-in should be octal because only 15 bits (5 octal characters) are transferred to the P-counter. Three octal zeros should be typed, followed by the meaningful 5 octal characters constituting the address to be placed in the P-counter, followed by the ninth formatting character.

Causes the data format to be controlled by the OCTAL/ALPHA switch

NOTE: The Data Transfer Sequences will cease, the I/O logic will be reset, and the PROGRAM COUNTER light will be turned off when:

Eight octal characters have been transferred, <u>or</u>
Four alpha characters have been transferred, <u>or</u>
Any of the following switches have been actuated:

END OF MESSAGE
INPUT ERROR (Input only)
RESET COMPUTER

When typing octal input, the fourth Data Transfer Sequence (for the seventh and eighth octal characters) occurs when the ninth formatting character is typed. Two octal characters are transferred during each Data Transfer Sequence.

Actuation of the switch in the Run mode has no effect.

TYP (Typewriter) COMMAND WAITING (Blue)

The TYP COMMAND WAITING light is illuminated when a read or write instruction is issued to the console typewriter <u>and</u> the typewriter is off-line. When this situation occurs, the typewriter control logic stores the instruction, but does not issue any data transfer requests until the typewriter is switched on-line. After the typewriter is switched on-line, the TYP COMMAND WAITING light is automatically turned off and Data Transfer Sequences commence.

There is no switch associated with this light.

OCTAL/ALPHA (White)

No light is associated with this switch.

The OCTAL/ALPHA switch is a two-position switch and is effective only when the computer is halted. The switch controls the form in which information is displayed on the typewriter.

INPUT/OUTPUT (White)

No light is associated with this switch.

The INPUT/OUTPUT switch is a two-position switch and is effective only when the computer is halted. The switch controls the direction of manual data transfers when any of the following switches are actuated:

PROGRAM COUNTER
DATA CONTROL WORD
MEMORY.

ON LINE/OFF LINE (White)

No light is associated with this switch.

The ON LINE/OFF LINE switch is a two-position switch. With the switch in the ON LINE position, the console typewriter can be used as an input/output device. With the switch in the OFF LINE position, the typewriter can be used by the operator without causing any input to the central processor.

The ON LINE/OFF LINE switch is effective only when there is no input or output operation in process.

INPUT/OUTPUT TYPEWRITER

General Description

Communication between the operator and the central processor is almost exclusively via the input/output typewriter. The two-way communication is through a standard character buffered channel. The typewriter can also be used off-line for recording purposes, such as keeping a daily log of computer use.

Typewriter Keyboard

All 64 characters of the GE-425/435 character set can be typed into the central processor from the typewriter keyboard. All input and output typing produces printed copy at the typewriter carriage. The space code (010000) is the only code generated by more than one typewriter key. It can be entered by a delta (Δ) key or by the space bar. As a safety feature, the typewriter keyboard is physically locked during on-line operation under program control, except when an input is initiated by the program.

Typewriter Carriage and Carriage Return

The standard typewriter carriage accepts paper 8-1/2 inches wide. Typewriters with special carriages can accept wider paper. The typewriter, during output, has an automatic carriage return which occurs when a space character (010000) is transmitted after the adjustable right margin contact has been closed by the movement of the carriage. The space character is automatically generated at the end of each word when output is in the octal mode. When in the alphanumeric mode, the space character must appear in the transferred data. During typewriter input, the carriage is returned by the operator when he depresses the "carriage return" key on the keyboard.

Typewriter Input/Output Channel

The typewriter is connected to a standard character channel. Because the typewriter normally needs no priority for program interrupts, it must be assigned to channel zero, which is the lowest priority channel.

Typewriter Controller

The typewriter control logic is housed beneath the table of the control console. It includes an I/O buffer, and logic to generate parity, to control sequencing, and to designate status. The following descriptions of data transfer between the typewriter and the central processor illustrate the part played by the typewriter controller.

<u>INFORMATION TRANSFER-INPUT.</u> Characters that are typed by the operator are encoded and odd parity is generated by the controller logic. Information is then stored temporarily in the controller buffer. When the typewriter I/O channel acknowledges readiness to receive information, data is transmitted to the typewriter I/O channel and subsequently to the central processor. Alphanumeric transfers occur one character at a time; octal transfers occur two characters at a time.

<u>INFORMATION TRANSFER-OUTPUT.</u> For on-line typewriter output, 6-bit characters (or two octal digits) and an odd parity bit are received serially from the central processor via the typewriter I/O channel. The information is stored temporarily in the character buffer of the controller until typed. After typing, the controller clears the buffer and signals readiness to receive another character from the typewriter I/O channel. A character counter provides a space or carriage return automatically after each word of octal output.

TYPEWRITER OPERATING PROCEDURES

On-Line Operation under Manual Control

In the Manual mode of operation, the typewriter functions under the control of console switches on the operator panel. The typewriter provides input or output in octal or alphanumeric characters.

Typing in Octal vs. Typing in Alphanumeric

The operator should know the procedures for typing in octal and for typing in alphanumeric.

OCTAL TYPING. When set for octal operation, the OCTAL/ALPHA switch is set to OCTAL, and the OCT REQD portion of the OCT REQD/ALP REQD switch is conditioned to glow when typing is initiated. Octal characters are placed in memory eight characters to a word. Transfer of data from the typewriter controller buffer to memory is in steps of two octal characters at a time, meaning that there is a transfer of information to memory with the depression of every other typewriter key. After typing each word of information (8 octal characters), a ninth formatting character must be typed. Because this ninth character is not transmitted, it can be any of the 64 characters or a carriage return. The transfer to memory of the seventh and eighth characters in a word does not occur until the ninth character is typed.

ALPHANUMERIC TYPING. When set for alphanumeric operation, the OCTAL/ALPHA switch is set to ALPHA, and the ALP REQD portion of the OCT REQD/ALP REQD switch is conditioned to glow when typing is initiated. Alphanumeric characters are placed in memory four characters to a word. Transfer from the typewriter controller buffer is one 6-bit character at a time, meaning that there is a transfer of information to memory each time a typewriter key is depressed. No extra formatting characters should be typed.

Manual Typing Procedures

Input and output operations can be performed as follows when the typewriter is under manual control:

1. Type into or from the P-counter.
2. Type into or from the typewriter channel data control word.
3. Type into or from the address specified by the data control word.
4. Type information from the accumulator location register.
5. Type out the current instruction.

The following paragraphs describe the steps necessary to perform each of the above operations as well as correction procedures for errors during type-ins.

Typed Input

To type into the P-counter:

1. Make sure that the computer is halted (MANUAL light is on).
2. Set the INPUT/OUTPUT switch to INPUT.
3. Set the OCTAL/ALPHA switch to OCTAL.
4. Depress the PROGRAM COUNTER switch.
5. After the OCT REQD light comes on, type three octal zeros, then the five address characters and a carriage return (or any standard character).

To type into the typewriter DCW location:

1. Make sure that the computer is halted (MANUAL light is on).
2. Set the INPUT/OUTPUT switch to INPUT.
3. Set the OCTAL/ALPHA switch to OCTAL.
4. Depress the DATA CONTROL WORD switch.
5. After the OCT REQD light comes on, type eight octal characters and a carriage return (or any standard character).

To type into the address specified by the typewriter DCW:

1. Make sure that the computer is halted (MANUAL light is on).
2. Set the INPUT/OUTPUT switch to INPUT.
3. Set the OCTAL/ALPHA switch to the desired setting.
4. Depress the MEMORY switch.
5. Wait for the OCT REQD or ALP REQD light to come on.
6. Type eight octal or four alpha characters (to correspond with the switch setting of step 3). If in octal, type a carriage return (or any standard character).
7. Repeat step 6 for each additional word to be entered.
8. After the last word is entered, depress the END OF MESSAGE switch if the OCT REQD or ALP REQD light is still on.

To correct manual control typing errors:

When the operator makes a typing error while typing under manual control, it is usually easier to retype the message completely, rather than to retype the message from the point where the error occurred. The correction procedure is:

1. Depress the INPUT ERROR switch.

2. Set the OCTAL/ALPHA switch to OCTAL.

3. Type into the typewriter channel DCW location the DCW with the count field and address field that was initially used for the message.

4. Set the OCTAL/ALPHA switch to correspond to the format of the input message.

5. Depress the MEMORY switch.

6. Retype the input message, starting from the beginning.

Typed Output

To type out the contents of the P-counter:

1. Make sure that the computer is halted (MANUAL light is on).
2. Set the INPUT/OUTPUT switch to OUTPUT.
3. Set the OCTAL/ALPHA switch to OCTAL.
4. Depress the PROGRAM COUNTER switch.

Three zeros will be typed, followed by the five-character contents of the P-counter. These will be followed by either a space or a carriage return, depending upon the position of the carriage.

To type out the contents of the typewriter DCW location:

1. Make sure that the computer is halted (MANUAL light is on).
2. Set the INPUT/OUTPUT switch to OUTPUT.
3. Set the OCTAL/ALPHA switch to OCTAL.
4. Depress the DATA CONTROL WORD switch.

The contents of the DCW will then be typed as eight octal characters, followed by either a space or a carriage return, depending on the position of the carriage.

To type out the contents of the address specified by the typewriter DCW:

1. Make sure that the computer is halted (MANUAL light is on).
2. Set the INPUT/OUTPUT switch to OUTPUT.

3. Set the OCTAL/ALPHA switch to the desired setting.
4. Depress the MEMORY switch.

If the OCTAL/ALPHA switch is set to OCTAL, this procedure will type eight characters followed by a space or carriage return, depending on the position of the carriage. If set for ALPHA, this procedure will type four characters. This four- or eight-character typeout will be repeated for each word of information to be typed.

5. Depress the END OF MESSAGE switch. Typing will halt either when the DCW overflows or the END OF MESSAGE switch is depressed.

<u>To type out the contents of the working accumulator:</u>

This procedure first requires learning the location and size of the working accumulator. The steps are as follows:

1. Make sure the computer is halted (MANUAL light is on).

2. Set the OCTAL/ALPHA switch to OCTAL.

3. Depress the ACCUM LOCATION TYPEOUT switch. This causes an octal typeout of three zeros followed by a five-character address of the most-significant word of the working accumulator. In this address, the least-significant octal character is always an indication of the length of the working accumulator, according to the following code:

0 or 4 = Quadruple	2 or 6 = Double
1 or 5 = Triple	3 or 7 = Single

4. Set the INPUT/OUTPUT switch to INPUT.

5. Depress the DATA CONTROL WORD switch (OCTAL REQD light will be on).

6. Convert the working length of the accumulator (obtained in step 3) to the following code (Note that $(774)_8 = (512-4)_{10}$):

   Quadruple = 760
   Triple = 764
   Double = 770
   Single = 774

Example:

If, in step 3, the following is typed,

00001646

the accumulator address is $(01646)_8$ and the working accumulator length is 2 words (see step 3)

To obtain the contents of location $(01646)_8$ follow steps 4, 5, 6, then type the numbers 770 (for double-length accumulator).

The typeout from the double accumulator will be either 16 octal characters or 8 alphanumeric characters.

7. Type the three octal code numbers for the proper accumulator length. These three characters enter the count field of the typewriter channel DCW.

8. Type the address of the accumulator. The address is designated by the five least-significant octal characters typed in step 3. When these five characters are typed as input, they enter the address field of the typewriter channel DCW word.

9. Depress the carriage return (or any standard character). This turns off the OCTAL REQD light.

10. Set the INPUT/OUTPUT switch to OUTPUT.

11. If the accumulator contents are to be typed in alphanumeric rather than octal, set the OCTAL/ALPHA switch to ALPHA.

12. Depress the MEMORY switch. The contents of the working accumulator will be typed in either octal or alphanumeric.

To type out an instruction:

1. Make sure that the computer is halted (MANUAL light is on).
2. Set the INPUT/OUTPUT switch to OUTPUT.
3. Set the OCTAL/ALPHA switch to OCTAL.
4. Depress the INSTR TYPEOUT switch.

The current instruction will then be typed as eight octal characters, followed by either a space or a carriage return, depending upon the position of the carriage when the last character is typed.

Programmed Input

Type-ins can be made by the operator when they are requested by the program. The need for a typed input is indicated by either an OCT REQD or an ALP REQD light. When a programmed typewriter input is demanded the operator takes the following steps:

1. Determine the input required. There will be either a typeout on the typewriter specifying the needed input or directions in the run book or other operating instructions.

2. Type the required information.

3. If directed to do so by operating instructions, depress the END OF MESSAGE switch. This switch must be depressed if the LPW count has not rolled over to zero after the message was typed. The OCT REQD or ALP REQD light is on until the LPW count rolls over to zero or the END OF MESSAGE switch is depressed.

Correction of Typing Errors in Programmed Input

When the operator makes a typing error while he is typing as a result of a requested input by the program, the correction procedure is as follows (this applies to either octal or alphanumeric input):

1. Depress the INPUT ERROR switch. This causes a program interrupt to the typewriter channel. An interrupt subroutine can detect that an error exists by examining the typewriter channel status word and should re-initiate the instruction which will cause the OCT REQD/ALP REQD light to come on again.

2. When the OCT REQD/ALP REQD light comes on, type the input message in which the error was made, starting from the beginning of the message.

On-Line Operation under Program Control

When the computer is operating under program control, it is said to be in the Run mode, and the green RUN light is illuminated. In this mode, the typewriter can function as both an input and an output device under control of the stored program. See Section VIII, Typewriter Input/Output Programming.

TYPICAL CONSOLE OPERATIONS

To illustrate the use of console switches and lights, a few examples are given of typical operating procedures.

Load Operation

It is assumed for this Load operation example that the peripheral device has been readied for input. The following steps cause one record (generally called a "bootstrap" program) to be read into memory and program operation to start:

1. Make sure the computer is halted (MANUAL light is on).

2. Set the LOAD CHANNEL switch to the channel number that selects the peripheral subsystem from which a record is to be loaded.

3. If loading is to be from a multi-device peripheral subsystem, select the appropriate device at the peripheral controller.

4. Depress the LOAD switch. This causes the device to read one record into memory. The first word of that record goes into the DCW for the selected channel, and the address portion of the first word also goes into the P-counter. This is the address of the first instruction in the program. The remainder of the record is transferred into memory under the control of the word placed in the DCW. The character count of the DCW must be sufficient to permit the full record to be read.

5. Check to see if the LOAD light is on, indicating that the reading was successful. If an error occurred during loading (LOAD light not on), the error condition must be corrected, the RESET COMPUTER switch must be depressed, and the operator must return to step 4 of this procedure.

6. Depress the RUN switch to start instruction processing.

Start Program at Specified Address

In the following example, it is assumed that the program is already in memory but the desired starting location is not in the P-counter. To start the program at location 250_8, take the following steps:

1. Make sure the computer is halted (MANUAL light is on).

2. Set the INPUT/OUTPUT switch to INPUT.

3. Set the OCTAL/ALPHA switch to OCTAL.

4. Depress the PROGRAM COUNTER switch (OCTAL REQD will be illuminated).

5. Type the eight characters 00000250 followed by a carriage return, or any standard character.

6. Depress the RUN switch.

Recognize a Programmed Halt

The operator can recognize that the computer has halted as a result of a programmed Halt instruction if the following occurs:

1. The MANUAL light is on.
2. No error lights are on.
3. The MANUAL switch was not depressed.

The Halt instruction can be typed on the typewriter by setting the OCTAL/ALPHA switch to OCTAL and depressing the INSTR TYPEOUT switch.

Reset a Memory Parity Alert

When a memory parity error is detected, the computer halts with MEMORY PARITY ALERT and MANUAL lights on. Depressing either of the following switches resets the alert condition:

1. MEMORY PARITY ALERT
2. RESET COMPUTER.

Reset an Instruction Alert

An instruction alert means that there was an invalid operation code or an invalid memory address. Normally, this causes a program interrupt to the processor channel. However, if the computer is in the Program Interrupt mode, the alert condition causes the computer to halt with the MANUAL, PROGRAM INTER RESET, and INSTR ALERT lights on. Depressing either of the following switches resets the alert condition:

1. INSTR ALERT
2. RESET COMPUTER.

Reset a Control Word Alert

When there is a control word alert, the computer halts with the MANUAL and CONTROL WORD ALERT lights on. This means either of the following two things, depending upon whether PROGRAM INTER RESET is illuminated or not:

1. If it is illuminated, an invalid address was detected in an LPW or DCW while the computer was in the Program Interrupt mode.

2. If it is not illuminated, an operation code other than SPB or a General instruction was detected in a PIW location.

In either case, depressing either of the following switches resets the alert condition:

1. CONTROL WORD ALERT
2. RESET COMPUTER.

Status and the ON LINE/OFF LINE Switch

The typewriter major status is not changed when the ON LINE/OFF LINE switch is changed to OFF LINE. If a read or write instruction is issued while the typewriter is off-line:

1. The TYP COMMAND WAITING light is turned on.

2. The typewriter major status, indicating that the instruction has been accepted, is stored in the location specified by the instruction.

3. The typewriter becomes busy and remains busy until the ON LINE/OFF LINE switch is set to ON LINE and the required typing operation is completed. Any other typewriter instruction issued during this period will reflect the busy status to the control processor.

The normal operating position for the ON LINE/OFF LINE switch is ON LINE. While the switch is in the OFF LINE position, the operator can use the typewriter without effecting any input to the processor and no program interrupts can be requested by the typewriter channel.

Effect of Typing in the Wrong Mode

The setting of the OCTAL/ALPHA switch is significant only during manual type-ins and typeouts. The switch has no effect on programmed type-ins or programmed typeouts.

OCTAL/ALPHA SWITCH SET TO OCTAL

Input. When typing in the octal mode, only the numerals 0 through 7 (plus a formatting character) should be typed. If characters other than 0 through 7 are typed, input information will be unintelligible in memory. If the operator thinks he is typing in the alphanumeric mode, there is no feedback to tell him of his mistake (at least not until the program identifies the error).

Output. For manual typeouts, information which is known to be octal should be typed in the octal mode (only characters 0 through 7 from memory). If the memory location contains alphanumeric characters (including 0 through 7 in alphanumeric form), these will be interpreted as two octal characters and typed as such. For example, the octal representation of the character A is 21, and in the octal mode it would be typed as 21. For programmed typeouts, the OCTAL/ALPHA switch has no effect.

OCTAL/ALPHA SWITCH SET TO ALPHA.

Input. All 64 characters can be typed in the alphanumeric mode. If however, the operator thinks that the system is in the octal mode and types a numerical message, these numbers enter memory in their 6-bit alphanumeric form. Because the word in memory contains only 4 characters (instead of 8 as in octal), only half of the numerals in the message are entered in the space allotted. The operator may notice his error if the ALP REQD light goes off before his message is completed.

Output. If, during manual typeouts, the information in memory is in octal but the operator has the switch set to ALPHA, each two octal characters in memory are interpreted as one alphanumeric character and typed as such. The appearance of non-octal characters in the information should indicate to the operator the mistake he has made.

MAINTENANCE PANEL

The maintenance panel (see Figure VII-3) is at the right of the operator panel and is covered by a hinged door which must be opened for access to panel switches and lights. The panel has controls and display lights for monitoring and manipulating central processor internal conditions. The panel is intended to be used only by maintenance personnel, but can be observed by programmers during program debugging.

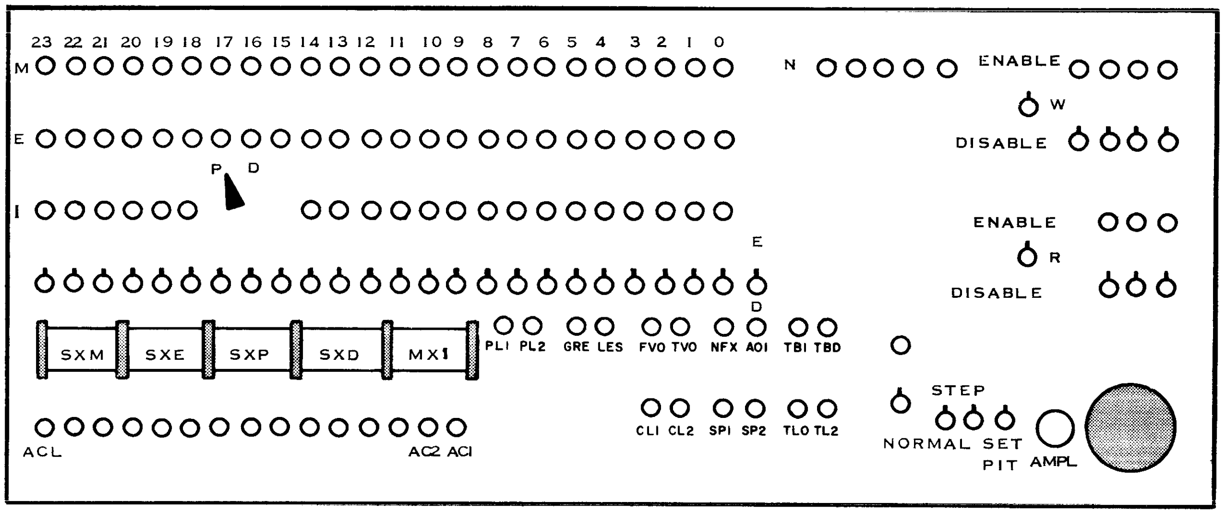


Figure VII-3. Maintenance Panel (Cover Removed)

VIII. TYPEWRITER INPUT/OUTPUT PROGRAMMING

USE OF THE GENERAL INSTRUCTION

Programmed input or output typing operations are initiated by a General instruction. In addition to initiating typewriter output or requesting typewriter input, the General instruction is also used to store typewriter status in the memory location indicated by the second address of the instruction. The General instruction is described in detail in Section III, Instruction Repertoire. The format of the instruction word for typewriter instructions is:

23 ← 18	17 ← 15	14 ← 11	10 ← 6	5 ← 0
- - -	X	0	0	Command
Op Code	ACF	Channel	Device	Code

The format of the status word specified by the SAS is:

23 ← 4	3 ← 0
Zeros	Major Status

The major status indicates one of seven conditions for the typewriter. There is no substatus information for typewriter operations.

MAJOR STATUS

The following is a list of major status returns for the typewriter:

0000	Typewriter Ready - The command has been accepted by the typewriter and the operation has been initiated. If the command was Request Status of Typewriter or Reset Status of Typewriter, the typewriter is ready to receive a subsequent command. No abnormal conditions are present in the channel or the typewriter.
0010	Operator Attention - The typewriter is unable to accept an instruction because of an inoperable condition requiring operator intervention (typewriter motor AC power is off).
0011	Data Alert - Indicates that the operator has signaled a typing error by depressing the INPUT ERROR switch.
0101	Command Rejected - An illegitimate command has been given to the typewriter.
1000	Typewriter Busy - The channel/peripheral subsystem is busy executing a previous command, which prevents the channel from transmitting another command at this time.
1001	Typewriter Absent - The channel is unable to communicate with the typewriter because: a. The typewriter is disconnected. b. The typewriter controller is without DC power. c. The channel is absent
1010	Channel Error - During the last input operation using the typewriter, an error occurred in transmitting data from the typewriter to the channel.

Mnemonic codes are provided in the basic assembly language to specify typewriter input and output. Codes are also available to request and reset typewriter status. These codes use the same format as the General instruction and can be used with either a fixed index, an Operand Pointer, or an Operand Link. The six mnemonic codes which apply to typewriter input and output are described in the following paragraphs.

TYPEWRITER INSTRUCTIONS

TYPE OUTPUT OCTAL

TOO	X	0 7 X 0 0 0 1 1
OP/OL	Z	0 C 0 Z Z Z Z Z

Any existing resettable status (Data Alert or Command Reject) of the typewriter or typewriter channel is reset and the remaining status is placed in the location specified by the second address. If the typewriter is ready, octal characters are typed out, starting at the memory location specified by the first word of the data control list. Typing continues until all characters specified by the list are typed. If the typewriter is not ready, the status word stored in the location specified by the second address indicates the major status that prevented the typeout from occurring.

Note that the data control list specifies alphanumeric character counts; thus, two octal characters are typed for each alphanumeric character counted.

TYPE OUTPUT ALPHANUMERIC

TOA	X	0 7 X 0 0 0 1 3
OP/OL	Z	0 C 0 Z Z Z Z Z

Any existing resettable status (Data Alert or Command Reject) of the typewriter or typewriter channel is reset and the remaining status is placed in the location specified by the second address. If the typewriter is ready, alphanumeric characters are typed out, starting at the memory location specified by the first word of the data control list. Typing continues until all characters specified by the list are typed. If the typewriter is not ready, the status word stored in the location specified by the second address indicates the major status that prevented the typeout from occurring.

TYPE INPUT OCTAL

TIO	X	0 7 X 0 0 0 0 1
OP/OL	Z	0 C 0 Z Z Z Z Z

Any existing resettable status (Data Alert or Command Reject) of the typewriter or typewriter channel is reset and the remaining status is placed in the location specified by the second address. The OCT REQD light on the operator panel is lit to inform the operator that the computer is ready to receive the requested input. It remains on until the number of octal characters specified by the data control list have been typed or until either the END OF MESSAGE or INPUT ERROR switch is depressed.

Note that the data control list specifies <u>alphanumeric</u> count fields; thus, two <u>octal</u> characters must be typed for each alphanumeric character counted.

TYPE INPUT ALPHANUMERIC

TIA	X	0 7 X 0 0 0 0 3
OP/OL	Z	0 C 0 Z Z Z Z Z

Any existing resettable status (Data Alert or Command Reject) of the typewriter or typewriter channel is reset and the remaining status is placed in the location specified by the second address. The ALP REQD light on the operator panel is lit to inform the operator that the computer is ready to receive the requested input, and remains on until the number of alphanumeric characters specified by the data control list have been typed or until either the END OF MESSAGE or INPUT ERROR switch is depressed.

REQUEST STATUS OF TYPEWRITER

RQST	X	0 7 X 0 0 0 0 0
OP/OL	Z	0 C 0 Z Z Z Z Z

The existing status of the typewriter is placed in the location specified by the second address.

RESET STATUS OF TYPEWRITER

RSST	X	0 7 X 0 0 0 4 0
OP/OL	Z	0 C 0 Z Z Z Z Z

Any existing resettable status (Data Alert or Command Reject) of the typewriter is reset and the remaining status is placed in the location specified by the second address.

POSITIONING OF TYPED INFORMATION

Memory words and typed words will have corresponding character counts if the count field of the typewriter DCW is zero or an even multiple of four (0, modulo 4) at the time that input or output is initiated. The DCW count field for alphanumeric operations can specify any number of characters from 1 to 512 and need not be a 0, modulo 4, number. All octal typing operations, however, should use a DCW count field that is 0, modulo 4.

If the DCW count field is not an even multiple of four when an alphanumeric input is initiated, a number of leading zeros (1, 2, or 3) will be inserted in the memory word receiving the first input character.

If either an alphanumeric or octal type-in is terminated in the middle of a memory word (because either the END OF MESSAGE or the INPUT ERROR switch was depressed), unused least-significant positions of that word are automatically filled with zeros.

During alphanumeric output operations, the first character typed is determined by the remainder when the DCW count field (512 - N) is divided by 4. The remainder also determines the number of leading characters that will be ignored in the first memory word to be typed.

During both alphanumeric and octal output operations, the last character to be typed is always the least-significant character of a memory word.

Examples of octal input and alphanumeric input and output with different DCW count field characteristics follow.

Octal Input

Only the numerals 0 through 7 are used in octal input and output. Eight octal characters are placed in a word of memory. When the DCW count is zero or an even multiple of 4, input characters are left-justified in a word. If the END OF MESSAGE or INPUT ERROR switch terminates input before a memory word is filled, the least-significant positions of that word are automatically filled with zeros. In the following examples, the space bar (sp) is used as the ninth formatting character (required after each eight characters of octal input). If the DCW count were not an even multiple of 4, octal input characters would automatically be preceded by a number of leading zeros corresponding to the remainder after the DCW count is divided by 4.

Example 1: The DCW count is 504 (770_8), which is an even multiple of 4. Characters typed in groups of 8 octal numbers enter memory in the same configuration, and each group of 8 characters occupies one memory location. The DCW count cuts off the input after 16 characters have been typed.

Characters typed:	0	1	2	3	4	5	6	7	sp	7	6	5	4	3	2	1	0
Memory (octal):	0	1	2	3	4	5	6	7		7	6	5	4	3	2	1	0

Example 2: The DCW count is again 504 (770_8), but the input is terminated by the END OF MESSAGE switch. Zeros fill the remaining positions of the last memory word.

Characters typed:	0	1	2	3	4	5	6	7	sp	7	6	5	End of Message				
Memory (octal):	0	1	2	3	4	5	6	7		7	6	5	0	0	0	0	0

Octal Output

During octal output, a space character is automatically actuated after each group of eight characters. If the typewriter carriage is sufficiently far to the right at the time the space character is generated, the carriage return is activated and typing continues from the left margin. If the count field of the typewriter DCW is zero or an even multiple of 4 at the time typed output commences, typed words are reflections of memory words.

Example: The DCW count is 504 (770_8), which is an even multiple of 4. The typed words are identical to memory words.

Memory (octal):	0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7
Characters typed:	0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7

Alphanumeric Input

All 64 characters of the GE-425/435 character set (including the space character) can be entered by typed alphanumeric input. Four alphanumeric characters are placed in a word in memory.

No extra character should be typed between words. When the DCW count is zero or an even multiple of 4, input characters are left-justified in a word (most-significant positions are filled first). When type-in is terminated by the END OF MESSAGE switch before a memory location is filled, the least-significant positions are automatically filled with zeros. When the DCW count is not an even multiple of 4, alphanumeric input characters are moved right and are preceded by leading zeros. The number of positions moved and the number of leading zeros entered varies according to the remainder after the DCW count (512 - N) is divided by 4, as indicated in the following formula:

$$\frac{(\text{DCW Count})_{10}}{(4)_{10}} = \left\{ \begin{array}{ccc} \underline{\text{Quotient}} & \underline{\text{Remainder}} & \underline{\text{Number of Leading Zeros}} \\ X & 1 & 1 \\ X & 2 & 2 \\ X & 3 & 3 \end{array} \right.$$

Example 1: The DCW count is 504 (770_8), which is an even multiple of 4. Each group of 4 characters typed fills a memory word. The DCW count terminates the input after 8 characters have been typed.

Characters typed:	A	B	C	D	E	F	G	H
Memory (alphanumeric):	A	B	C	D	E	F	G	H

Example 2: The DCW count is again 504 (770_8), an even multiple of 4, but the input is terminated by the END OF MESSAGE switch. Zeros fill the last memory word.

Characters typed:	A	B	C	D	E	F	End of Message	
Memory (alphanumeric):	A	B	C	D	E	F	0	0

Example 3: The DCW count is 505 (771_8). After division by 4, the remainder is 1. This calls for one leading zero in the first word and positioning of all characters one place to the right. Input is terminated by the END OF MESSAGE switch. A zero is automatically placed in the last word.

```
Characters typed:          A  B  C  D  E  F  End of Message
Memory (alphanumeric):   | 0  A  B  C | D  E  F  0 |
```

Example 4: The DCW count is 506 (772_8). After division by 4, the remainder is 2. This calls for 2 leading zeros in the first word and positioning of all characters 2 places to the right.

```
Characters typed:          A  B  C  D  E  F
Memory (alphanumeric):   | 0  0  A  B | C  D  E  F |
```

Example 5: The DCW count is 507 (773_8). After division by 4, the remainder is 3. This calls for 3 leading zeros in the first word and positioning of all characters 3 places to the right.

```
Characters typed:          A  B  C  D  E
Memory (alphanumeric):   | 0  0  0  A | B  C  D  E |
```

Alphanumeric Output

In alphanumeric output, all formatting characters must be included in the coding. For example, a space occurs after a word only if it is included in the coding. When the DCW count is zero or an even multiple of 4, typed information is a reflection of memory words. There are no spaces between words unless they have been programmed. If the count field of the DCW is not zero or an even multiple of 4, typed format depends upon the remainder after the DCW count is divided by 4. Characters are ignored at the most-significant end of the first word. The following diagram and examples illustrate typed alphanumeric output from memory. In each case, the DCW count terminates the output operation.

$$\frac{(\text{DCW Count})_{10}}{(4)_{10}} =$$

Quotient	Remainder	Number of Characters Ignored
X	1	1
X	2	2
X	3	3

Example 1: The DCW count is 504 (770_8), which is an even multiple of 4. Letters typed are identical to those in two memory words. There is no space between the words because none was programmed.

Memory (alphanumeric):	A	B	C	D	E	F	G	H
Characters typed:	A	B	C	D	E	F	G	H

Example 2: The DCW count is 505 (771_8). After division by 4, the remainder is 1. The most-significant character of the first word is ignored.

Memory (alphanumeric):	A	B	C	D	E	F	G	H
Characters typed:	B	C	D	E	F	G	H	

Example 3: The DCW count is 506 (772_8). After division by 4, the remainder is 2. The two most-significant characters of the first word are ignored.

Memory (alphanumeric):	A	B	C	D	E	F	G	H
Characters typed:	C	D	E	F	G	H		

Example 4: The DCW count is 507 (773_8). After division by 4, the remainder is 3. The three most-significant characters of the first word are ignored.

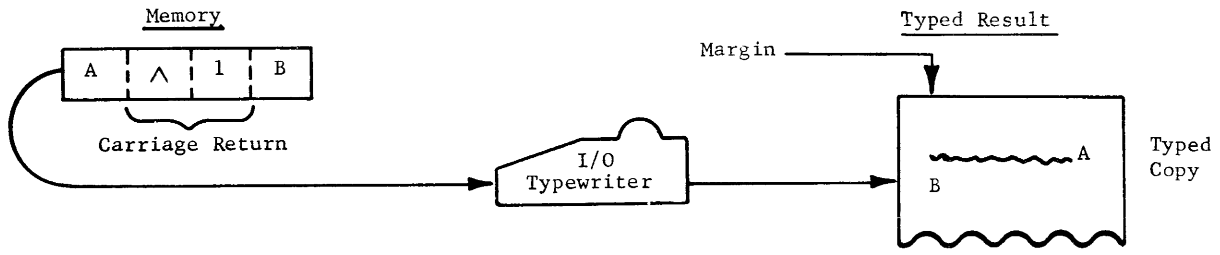
```
Memory (alphanumeric):  | A  B  C  D | E  F  G  H |
Characters typed:          D  E  F  G  H
```

ALPHANUMERIC OUTPUT FORMATTING

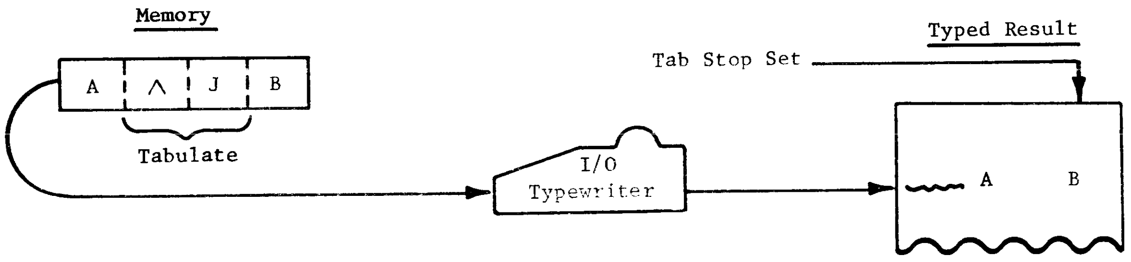
Special Functions of the Escape Character

In alphanumeric output, the Escape character (∧) causes the character following to assume a quality different from that which it normally has. Characters are combined with the Escape character to perform special functions as follows:

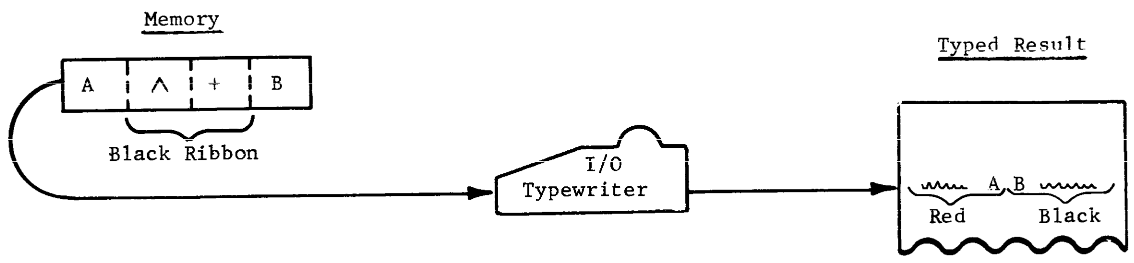
1. The Escape (∧) followed by a 1 causes a carriage return. (No print action occurs.) The carriage return can be caused at any place on the line.



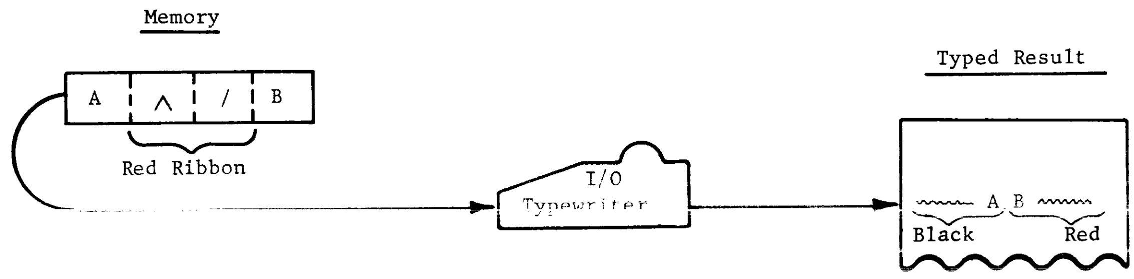
2. The Escape (∧) followed by a J causes a tabulate. (No print action occurs.)



3. The Escape (∧) followed by a plus sign (+) causes a black-ribbon shift. Subsequent typewriter printing is performed with the black (lower) portion of the typewriter ribbon. (No print action occurs.)



4. The Escape (∧) followed by a slash (/) causes a red-ribbon shift. Subsequent typewriter printing is performed with the red (upper) portion of the typewriter ribbon. (No print action occurs.)

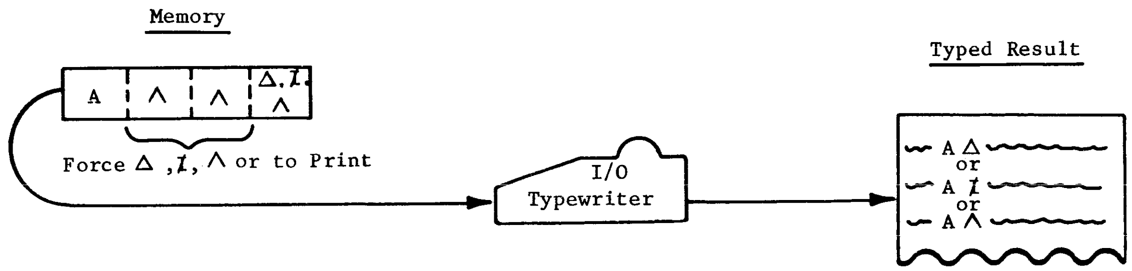


5. The Escape (∧) followed by another Escape character causes the character following the two Escape characters to be printed, no matter what the character is. This provides a means for printing a symbol for each of three characters that do not ordinarily print symbols. Characters that are printed only as the result of following two Escape characters are:

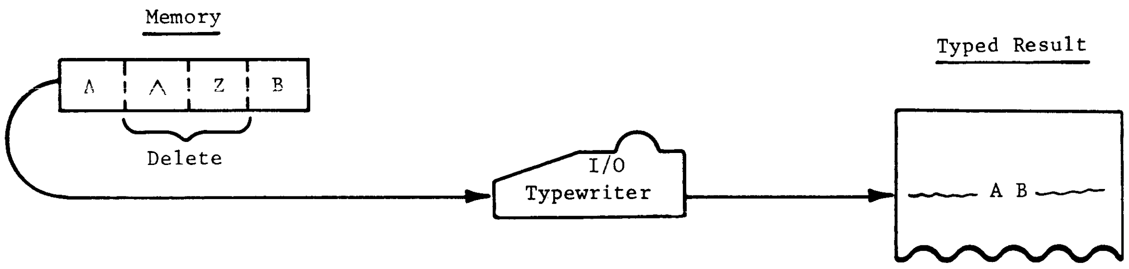
The space (010000), which prints as a Δ (delta).

The Ignore (001111), which prints as a ⫽ (slashed vertical).

The Escape (111111), which prints as a ∧ (caret).



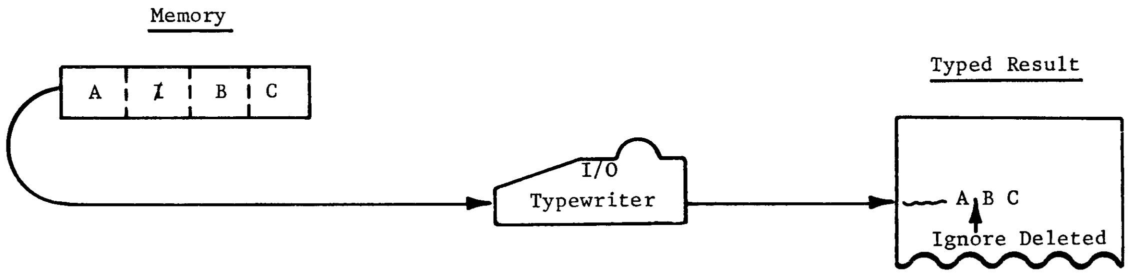
6. The Escape (∧) followed by any character other than 1, J, +, /, or ∧ results in the deletion of both the Escape character and the character which follows it.



Special Functions of the Ignore Character

The Ignore character (Ɨ) can be used two ways in typewriter output:

1. When the Ignore character does not follow an Escape character, the Ignore character is deleted from the data received from the central processor. No print action occurs (not even a space).



2. When the Ignore character follows two Escape characters, the Ignore symbol is printed as Ɨ. See the illustration in item five of the list under the heading Special Functions of the Escape Character.

APPENDIX 1.

CONVERTING BETWEEN NUMBER SYSTEMS

Because the programmer frequently must convert between number systems, a discussion of the methods for doing conversions, and examples, is outlined in this appendix. Conversions may be done with arithmetic, or by using charts showing the conversion relationships between the systems. A useful table showing the powers of 8 and 2 appears in Figure A-1. In practice, most conversions use the octal number system as an intermediate step between the decimal and binary number systems.

The conversion examples and tables which follow use octal numbers below 77,777,777. This number represents the full 24-bit word length of the GE-425 system with all bits "on", (binary 111 111 111 111 111 111 111 111), and thus is usually sufficient.

DECIMAL-TO-OCTAL CONVERSION

The usual procedure for converting a quantity expressed in one number system to another number system is to divide by the base number of the latter system. Thus, to convert a decimal number to the octal system, the decimal number is divided repeatedly by 8. The remainders from each division are combined in sequence from right to left to form an octal number which is equivalent to the decimal number.

Example: Convert $349,798_{10}$ to octal notation.

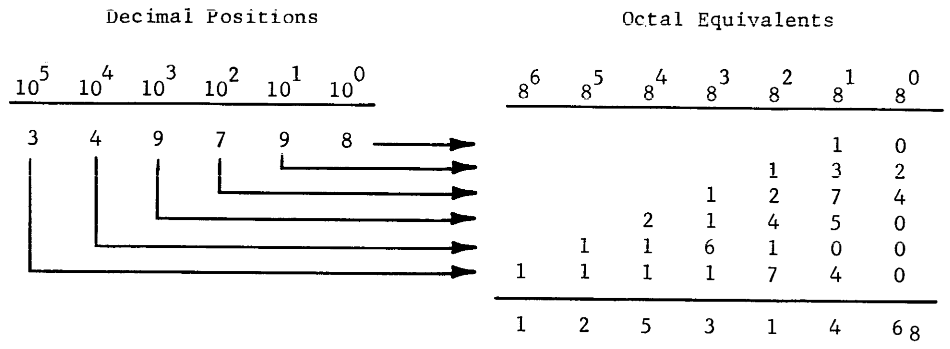
8) 349798	Remainder = 6 $\longrightarrow$ 6	1st remainder
8) 43624	R = 4 $\longrightarrow$ 46	1st two remainders
8) 5465	R = 1 $\longrightarrow$ 146	1st three remainders
8) 683	R = 3 $\longrightarrow$ 3146	1st four remainders
8) 85	R = 5 $\longrightarrow$ 53146	1st five remainders
8) 10	R = 2 $\longrightarrow$ 253146	1st six remainders
8) 1	R = 1 $\longrightarrow$ 1253146_8	all remainders
0		

Powers of 8	Decimal Number	Powers of 2
8^0=	1	2^0
	2	2^1
	4	2^2
8^1=	8	2^3
	16	2^4
	32	2^5
8^2=	64	2^6
	128	2^7
	256	2^8
8^3=	512	2^9
	1 024	2^{10}
	2 048	2^{11}
8^4=	4 096	2^{12}
	8 192	2^{13}
	16 384	2^{14}
8^5=	32 768	2^{15}
	65 536	2^{16}
	131 072	2^{17}
8^6=	262 144	2^{18}
	524 288	2^{19}
	1 048 576	2^{20}
8^7=	2 097 152	2^{21}
	4 194 304	2^{22}
	8 388 608	2^{23}
8^8=	16 777 216	2^{24}

Figure A-1. Table of Powers of 2 and 8

Another method of conversion from decimal to octal notations involves the use of the two conversion charts in Figure A-2. This method has the advantage of requiring no arithmetic more complex than decimal addition. The octal equivalent of each decimal digit of the number to be converted is located in the upper table. These equivalents are then added one column at a time in decimal. After the sum for a column is found, the octal equivalent of the sum is located in the lower table and written below the column with any carry being carried over to the next column as in ordinary addition.

Example: Convert $349{,}798_{10}$ to octal notation using the charts in Figure A-2.



Decimal Positions		Octal Equivalents						
		8^6	8^5	8^4	8^3	8^2	8^1	8^0
10^0	8						1	0
10^1	9					1	3	2
10^2	7				1	2	7	4
10^3	9			2	1	4	5	0
10^4	4		1	1	6	1	0	0
10^5	3	1	1	1	1	7	4	0
		1	2	5	3	1	4	6_8

In this example, each digit of the decimal number was converted into its octal equivalent. For example, the digit 7 in the 10^2 position was located in the Decimal Digit column of the upper table in Figure A-2. Reading across to the 10^2 Decimal Position column, the octal equivalent is found to be 1274.

After the octal equivalents of all digits are located, they are added decimally one column at a time.

Adding the 8^0 column in the example gives 6, which is the same in octal and decimal.

Adding the 8^1 column gives 20_{10}, which is 24_8. The 4 is written as the sum of the 8^1 column and the 2 is carried into the 8^2 column.

The decimal sum of the digits in the 8^2 column (including the 2 carry) is 17_{10} which is 21_8. The 1 is written as the sum of the 8^2 column and the 2 is carried.

The decimal sum of the 8^3 column (including the 2 carry) is 11_{10} which is 13_8. The 3 is written as the sum of the 8^3 column and the 1 is carried.

The decimal sum of the 8^4 column (including the 1 carry) is 5 in both decimal and octal. The 5 is written as the sum of the 8^4 column.

Similarly, the sums of columns 8^5 and 8^6 are determined.

The resultant sum is 1253146_8, the same answer obtained when the decimal number was converted by repeated division by 8.

DECIMAL DIGIT	DECIMAL POSITION							
	10^7	10^6	10^5	10^4	10^3	10^2	10^1	10^0
1	46,113,200	3,641,100	303,240	23,420	1,750	144	12	1
2	114,226,400	7,502,200	606,500	47,040	3,720	310	24	2
3	162,341,600	13,343,300	1,111,740	72,460	5,670	454	36	3
4	230,455,000	17,204,400	1,415,200	116,100	7,640	620	50	4
5	276,570,200	23,045,500	1,720,440	141,520	11,610	764	62	5
6	344,703,400	26,706,600	2,223,700	165,140	13,560	1,130	74	6
7	413,016,600	32,547,700	2,527,140	210,560	15,530	1,274	106	7
8	461,132,000	36,411,000	3,032,400	234,200	17,500	1,440	120	10
9	527,245,200	42,252,100	3,335,640	257,620	21,450	1,604	132	11

DECIMAL	OCTAL	DECIMAL	OCTAL	DECIMAL	OCTAL	DECIMAL	OCTAL
1	1	15	17	29	35	43	53
2	2	16	20	30	36	44	54
3	3	17	21	31	37	45	55
4	4	18	22	32	40	46	56
5	5	19	23	33	41	47	57
6	6	20	24	34	42	48	60
7	7	21	25	35	43	49	61
8	10	22	26	36	44	50	62
9	11	23	27	37	45	51	63
10	12	24	30	38	46	52	64
11	13	25	31	39	47	53	65
12	14	26	32	40	50	54	66
13	15	27	33	41	51	55	67
14	16	28	34	42	52	56	70

Figure A-2. Decimal-to-Octal Conversion Charts

OCTAL-TO-DECIMAL CONVERSION

Numbers expressed in octal notation can be converted to decimal notation by a mathematical process called expansion. This is performed by multiplying each octal digit by powers of 8 according to its position factor. The results of each such multiplication are then added together, the sum being the decimal equivalent of the octal number.

To illustrate this process, a sample conversion of an octal number into its equivalent decimal number follows.

Example: Convert $1,253,146_8$ to decimal notation.

1 2 5 3 1 4 6

6	$6(8)^0$	=	6 x 1	=	6
4	$4(8)^1$	=	4 x 8	=	32
1	$1(8)^2$	=	1 x 64	=	64
3	$3(8)^3$	=	3 x 512	=	1,536
5	$5(8)^4$	=	5 x 4096	=	20,480
2	$2(8)^5$	=	2 x 32768	=	65,536
1	$1(8)^6$	=	1 x 262144	=	262,144
					$349,798_{10}$

Another method of octal-to-decimal conversion is to use a conversion table and merely look up the equivalent decimal number. The conversion table in Figure A-3 has been compiled for converting octal numbers up to 77,777,777 directly to decimal notation. The table shows the decimal equivalents of all octal digits as a function of their position in the octal number.

To illustrate the use of the table, consider the octal number 1761354. To convert this number to its decimal equivalent, read the equivalent decimal value of each octal digit from the table and add them to find the total decimal equivalent, as shown below.

Example: Convert $1,761,354_8$ to decimal notation.

Octal Positions: 8^6 8^5 8^4 8^3 8^2 8^1 8^0

1 7 6 1 3 5 4

Octal digit		10^5	10^4	10^3	10^2	10^1	10^0
4	=						4
5	=					4	0
3	=				1	9	2
1	=				5	1	2
6	=		2	4,	5	7	6
7	=	2	2	9,	3	7	6
1	=	2	6	2,	1	4	4
thus, 1761354_8	=	5	1	6,	8	4	4_{10}

OCTAL DIGIT VALUE	OCTAL DIGIT POSITION							
	8^7	8^6	8^5	8^4	8^3	8^2	8^1	8^0
1	2,097,152	262,144	32,768	4,096	512	64	8	1
2	4,194,304	524,288	65,536	8,192	1,024	128	16	2
3	6,291,456	786,432	98,304	12,288	1,536	192	24	3
4	8,388,608	1,048,576	131,072	16,384	2,048	256	32	4
5	10,485,760	1,310,720	163,840	20,480	2,560	320	40	5
6	12,882,912	1,572,864	196,608	24,576	3,072	384	48	6
7	14,680,064	1,835,008	229,376	28,672	3,584	448	56	7

Figure A-3. Octal-to-Decimal Conversion Chart

OCTAL-TO-BINARY CONVERSION

The converting of numbers from octal notation to their equivalent in binary notation is simply a process of substituting the binary equivalent of each octal number. Since octal digits range from 0 to 7, each binary equivalent will have three digits, or bits. Beginning with the right digit of the octal number, each digit is converted to its binary equivalent. It is a simple matter to memorize the binary equivalent of each octal number, as shown below.

Example: Convert 1234567_8 into binary notation.

Octal Number

$$\begin{tikzcd}
1 \arrow[d] & 2 \arrow[d] & 3 \arrow[d] & 4 \arrow[d] & 5 \arrow[d] & 6 \arrow[d] & 7 \arrow[d] \\
001 & 010 & 011 & 100 & 101 & 110 & 111
\end{tikzcd}$$

binary equivalent

BINARY-TO-OCTAL CONVERSION

By reversing the above process, conversion from binary notation back to octal is simplified. Each group of three binary bits becomes one octal digit in the range of 0 to 7 in octal notation.

APPENDIX 2.
ALPHABETIC INDEX OF GE-425/435 INSTRUCTIONS

MNEMONIC CODE	DESCRIPTION	OCTAL OP CODE OR INSTRUCTION WORD	PAGE
ABM (2a)	Add Binary to Memory	34	III-47
ABX (2a)	Add Binary to Index	34	III-98
ADD	Add Decimal Double	51	III-37
ADQ	Add Decimal Quadruple	53	III-41
ADS	Add Decimal Single	50	III-35
ADT	Add Decimal Triple	52	III-39
AIM (2a)	Add Immediate to Memory	33	III-51
AIX (2a)	Add Immediate to Index	33	III-100
AMD	Add to Memory Double	55	III-44
AMQ	Add to Memory Quadruple	57	III-46
AMS	Add to Memory Single	54	III-43
AMT	Add to Memory Triple	56	III-45
ANM (2a)	AND to Memory	24	III-83
ANX (2a)	AND to Index	24	III-106
BRC (2a)	Branch on Count	16	III-79
BRE	Branch if Equal	13	III-73
BRG	Branch if Greater	12	III-73
BRL	Branch if Less	14	III-73
BRM	Branch if Minus	11	III-78
BRU	Branch Unconditionally	10	III-70
BRZ	Branch if Zero	15	III-77
BXC (2a)	Branch on Index Count	16	III-104
CAA	Compare Alphanumeric Accumulator to Memory	03	III-61
CDA	Compare Decimal Accumulator to Memory	02	III-62
CMI (2a)	Compare Memory to Immediate	01	III-68
CMM (2a)	Compare Second to First Memory	04	III-65
CXI (2a)	Compare Index to Immediate	01	III-102
CXM (2a)	Compare Index to Memory	04	III-101
EDT	Edit	05	III-117
EXP	Explode	20	III-22
GEN (2a)	General	07	III-134
HLT	Halt	00	III-133
IMP	Implode	21	III-24
LAL	Load Accumulator Location	36	III-109
LBT	Low Bit Test	22	III-34
LDD	Load Double	41	III-7
LDQ	Load Quadruple	43	III-9
LDS	Load Single	40	III-6
LDT	Load Triple	42	III-8
LDX (2a)	Load Index	30	III-92
LXI (2a)	Load Index with Immediate	31	III-94
MFI (2a)	Move from Immediate	31	III-16
MFM (2a)	Move from First Memory	30	III-14
MOV (2a)	Move	06	III-20
MTA (2a)	Move to First Address Field	32	III-18
MXC (2a)	Move on Index Control	06	III-96

MNEMONIC CODE	DESCRIPTION	OCTAL OP CODE OR INSTRUCTION WORD	PAGE
PXB (2a)	Program Counter to Index and Branch	17	III-103
RALD	Reset Accumulator Length Double	22X06000	III-114
RALQ	Reset Accumulator Length Quadruple	22X04000	III-116
RALS	Reset Accumulator Length Single	22X02000	III-113
RALT	Reset Accumulator Length Triple	22X07000	III-115 III-86
RIM (2a)	OR Inclusive to Memory	23	III-107
RIX (2a)	OR Inclusive to Index	23	III-26
RLDA	Rotate Left Double Alpha	22X122NN	III-26
RLDAS	Rotate Left Double Alpha, Set	22X162NN	III-26
RLDD	Rotate Left Double Decimal	22X522NN	III-26
RLDDS	Rotate Left Double Decimal, Set	22X562NN	III-26
RLQA	Rotate Left Quadruple Alpha	22X102NN	III-26
RLQAS	Rotate Left Quadruple Alpha, Set	22X142NN	III-26
RLQD	Rotate Left Quadruple Decimal	22X502NN	III-26
RLQDS	Rotate Left Quadruple Decimal, Set	22X542NN	III-26
RLSA	Rotate Left Single Alpha	22X132NN	III-26
RLSAS	Rotate Left Single Alpha, Set	22X172NN	III-26
RLSD	Rotate Left Single Decimal	22X532NN	III-26
RLSDS	Rotate Left Single Decimal, Set	22X572NN	III-26
RLTA	Rotate Left Triple Alpha	22X112NN	III-26
RLTAS	Rotate Left Triple Alpha, Set	22X152NN	III-26
RLTD	Rotate Left Triple Decimal	22X512NN	III-26
RLTDS	Rotate Left Triple Decimal, Set	22X552NN	III-26
RQS (2a)	Request Status	07X00000	III-136
RQSP (2a)	Request Status of Processor	67X00000	III-125
RQST	Request Status of Typewriter	07X00000	VIII-4
RRD	Rotate Right Double	22X320NN	III-26
RRDA	Rotate Right Double Alpha	22X120NN	III-26
RRDAS	Rotate Right Double Alpha, Set	22X160NN	III-26
RRDD	Rotate Right Double Decimal	22X520NN	III-26
RRDDS	Rotate Right Double Decimal, Set	22X560NN	III-26
RRDT	Rotate Right Double, Test	22X324NN	III-26
RRQA	Rotate Right Quadruple Alpha	22X100NN	III-26
RRQAS	Rotate Right Quadruple Alpha, Set	22X140NN	III-26
RRQD	Rotate Right Quadruple Decimal	22X500NN	III-26
RRQDS	Rotate Right Quadruple Decimal, Set	22X540NN	III-26
RRS	Rotate Right Single	22X330NN	III-26
RRSA	Rotate Right Single Alpha	22X130NN	III-26
RRSAS	Rotate Right Single Alpha, Set	22X170NN	III-26
RRSD	Rotate Right Single Decimal	22X530NN	III-26
RRSDS	Rotate Right Single Decimal, Set	22X570NN	III-26
RRST	Rotate Right Single, Test	22X334NN	III-26

MNEMONIC CODE	DESCRIPTION	OCTAL OP CODE OR INSTRUCTION WORD	PAGE
RRTA	Rotate Right Triple Alpha	22X110N	III-26
RRTAS	Rotate Right Triple Alpha, Set	22X150NN	III-26
RRTD	Rotate Right Triple Decimal	22X510NN	III-26
RRTDS	Rotate Right Triple Decimal, Set	22X550NN	III-26
RSS (2a)	Reset Status	07X00040	III-137
RXM (2a)	OR Exclusive to Memory	25	III-89
RXX (2a)	OR Exclusive to Index	25	III-108
SAL	Store Accumulator Location	37	
	and Length		III-111
SBM (2a)	Subtract Binary from Memory	35	III-49
SBX (2a)	Subtract Binary from Index	35	III-99
SDD	Subtract Decimal Double	61	III-38
SDQ	Subtract Decimal Quadruple	63	III-42
SDS	Subtract Decimal Single	60	III-36
SDT	Subtract Decimal Triple	62	III-40
SLDA	Shift Left Double Alpha	22X022NN	III-26
SLDAS	Shift Left Double Alpha, Set	22X062NN	III-26
SLDD	Shift Left Double Decimal	22X422NN	III-26
SLDDS	Shift Left Double Decimal, Set	22X462NN	III-26
SLQA	Shift Left Quadruple Alpha	22X002NN	III-26
SLQAS	Shift Left Quadruple Alpha, Set	22X042NN	III-26
SLQD	Shift Left Quadruple Decimal	22X402NN	III-26
SLQDS	Shift Left Quadruple Decimal, Set	22X442NN	III-26
SLSA	Shift Left Single Alpha	22X032NN	III-26
SLSAS	Shift Left Single Alpha, Set	22X072NN	III-26
SLSD	Shift Left Single Decimal	22X432NN	III-26
SLSDS	Shift Left Single Decimal, Set	22X472NN	III-26
SLTA	Shift Left Triple Alpha	22X012NN	III-26
SLTAS	Shift Left Triple Alpha, Set	22X052NN	III-26
SLTD	Shift Left Triple Decimal	22X412NN	III-26
SLTDS	Shift Left Triple Decimal, Set	22X452NN	III-26
SPB (2a)	Store Program Counter and	17	III-71
	Branch		III-26
SRD	Shift Right Double	22X220NN	III-26
SRDA	Shift Right Double Alpha	22X020NN	III-26
SRDAS	Shift Right Double Alpha, Set	22X060NN	III-26
SRDD	Shift Right Double Decimal	22X420NN	III-26
SRDDS	Shift Right Double Decimal, Set	22X460NN	III-26
SRDT	Shift Right Double, Test	22X224NN	III-26
SRQA	Shift Right Quadruple Alpha	22X000NN	III-26
SRQAS	Shift Right Quadruple Alpha, Set	22X040NN	III-26
SRQD	Shift Right Quadruple Decimal	22X400NN	III-26
SRQDS	Shift Right Quadruple Decimal, Set	22X440NN	III-26
SRS	Shift Right Single	22Z230NN	III-26
SRSA	Shift Right Single Alpha	22X030NN	III-26
SRSAS	Shift Right Single Alpha, Set	22X070NN	III-26
SRSD	Shift Right Single Decimal	22X430NN	III-26
SRSDS	Shift Right Single Decimal, Set	22X470NN	III-26

MNEMONIC CODE	DESCRIPTION	OCTAL OP CODE OR INSTRUCTION WORD	PAGE
SRST	Shift Right Single, Test	22X234NN	III-26
SRTA	Shift Right Triple Alpha	22X010NN	III-26
SRTAS	Shift Right Triple Alpha, Set	22X050NN	III-26
SRTD	Shift Right Triple Decimal	22X410NN	III-26
SRTDS	Shift Right Triple Decimal, Set	22X450NN	III-26
SSA (2a)	Set Status by ANDing	67X00002	III-129
SSL (2a)	Set Status by Loading	67X00003	III-127
SSO (2a)	Set Status by ORing	67X00001	III-131
STD	Store Double	45	III-11
STQ	Store Quadruple	47	III-13
STS	Store Single	44	III-10
STT	Store Triple	46	III-12
SXA (2a)	Store Index in Address Field	32	III-95
VLD	Variable Length Divide	26	III-57
VLM	Variable Length Multiply	27	III-53

APPENDIX 3.
OCTAL INDEX OF GE-425/435 INSTRUCTIONS

OCTAL OP CODE	MNEMONIC CODE	DESCRIPTION	PAGE
00	HLT	Halt	III-133
01	CXI (2a)	Compare Index to Immediate	III-102
01	CMI (2a)	Compare Memory to Immediate	III-68
02	CDA	Compare Decimal Accumulator to Memory	III-62
03	CAA	Compare Alphanumeric Accumulator to Memory	III-61
04	CMM (2a)	Compare Second to First Memory	III-65
04	CXM (2a)	Compare Index to Memory	III-101
05	EDT	Edit	III-117
06	MOV (2a)	Move	III-20
06	MXC (2a)	Move on Index Control	III-96
07	---	See separate list of General Instructions	---
10	BRU	Branch Unconditionally	III-70
11	BRM	Branch if Minus	III-78
12	BRG	Branch if Greater	III-73
13	BRE	Branch if Equal	III-73
14	BRL	Branch if Less	III-73
15	BRZ	Branch if Zero	III-77
16	BXC (2a)	Branch on Index Count	III-104
16	BRC (2a)	Branch on Count	III-79
17	PXB (2a)	Program Counter to Index and Branch	III-103
17	SPB (2a)	Store Program Counter and Branch	III-71
20	EXP	Explode	III-22
21	IMP	Implode	III-24
22	---	See Separate list of Shift Instructions	---
23	RIM (2a)	OR Inclusive to Memorv	III-86
23	RIX (2a)	OR Inclusive to Index	III-107
24	ANM (2a)	AND to Memory	III-83
24	ANX (2a)	AND to Index	III-106
25	RXM (2a)	OR Exclusive to Memory	III-89
25	RXX (2a)	OR Exclusive to Index	III-108
26	VLD	Variable Length Divide	III-57
27	VLM	Variable Length Multiply	III-53
30	LDX (2a)	Load Index	III-92
30	MFM (2a)	Move from First Memory	III-14
31	LXI (2a)	Load Index with Immediate	III-94
31	MFI (2a)	Move from Immediate	III-16
32	MTA (2a)	Move to First Address Field	III-18
32	SXA (2a)	Store Index in Address Field	III-95
33	AIM (2a)	Add Immediate to Memory	III-51
33	AIX (2a)	Add Immediate to Index	III-100
34	ABM (2a)	Add Binary to Memory	III-47
34	ABX (2a)	Add Binary to Index	III-98
35	SBM (2a)	Subtract Binary from Memory	III-49
35	SBX (2a)	Subtract Binary from Index	III-99
36	LAL	Load Accumulation Location and Length	III-109
37	SAL	Store Accumulator Location and Length	III-111

OCTAL OP CODE	MNEMONIC CODE	DESCRIPTION	PAGE
40	LDS	Load Single	III-6
41	LDD	Load Double	III-7
42	LDT	Load Triple	III-8
43	LDQ	Load Quadruple	III-9
44	STS	Store Single	III-10
45	STD	Store Double	III-11
46	STT	Store Triple	III-12
47	STQ	Store Quadruple	III-13
50	ADS	Add Decimal Single	III-35
51	ADD	Add Decimal Double	III-37
52	ADT	Add Decimal Triple	III-39
53	ADQ	Add Decimal Quadruple	III-41
54	AMS	Add to Memory Single	III-43
55	AMD	Add to Memory Double	III-44
56	AMT	Add to Memory Triple	III-45
57	AMQ	Add to Memory Quadruple	III-46
60	SDS	Subtract Decimal Single	III-36
61	SDD	Subtract Decimal Double	III-38
62	SDT	Subtract Decimal Triple	III-40
63	SDQ	Subtract Decimal Quadruple	III-42
67	---	See separate Central Processor Operation Instruction listing	---

SHIFT INSTRUCTIONS

OCTAL FORMAT	MNEMONIC CODE	DESCRIPTION	PAGE
22X000NN	SRQA	Shift Right Quadruple Alpha	III-26
22X002NN	SLQA	Shift Left Quadruple Alpha	III-26
22X010NN	SRTA	Shift Right Triple Alpha	III-26
22X012NN	SLTA	Shift Left Triple Alpha	III-26
22X020NN	SRDA	Shift Right Double Alpha	III-26
22X022NN	SLDA	Shift Left Double Alpha	III-26
22X030NN	SRSA	Shift Right Single Alpha	III-26
22X032NN	SLSA	Shift Left Single Alpha	III-26
22X04000	RALQ	Reset Accumulator Length Quadruple	III-116
22X040NN	SRQAS	Shift Right Quadruple Alpha, Set	III-26
22X042NN	SLQAS	Shift Left Quadruple Alpha, Set	III-26
22X05000	RALT	Reset Accumulator Length Triple	III-115
22X050NN	SRTAS	Shift Right Triple Alpha, Set	III-26
22X052NN	SLTAS	Shift Left Trible Alpha, Set	III-26
22X06000	RALD	Reset Accumulator Length Double	III-114
22X060NN	SRDAS	Shift Right Double Alpha, Set	III-26
22X062NN	SLDAS	Shift Left Double Alpha, Set	III-26
22X07000	RALS	Reset Accumulator Length Single	III-113
22X070NN	SRSAS	Shift Right Single Alpha, Set	III-26
22X072NN	SLSAS	Shift Left Single Alpha, Set	III-26
22X100NN	RRQA	Rotate Right Quadruple Alpha	III-26
22X102NN	RLQA	Rotate Left Quadruple Alpha	III-26
22X110NN	RRTA	Rotate Right Triple Alpha	III-26
22X112NN	RLTA	Rotate Left Triple Alpha	III-26
22X120NN	RRDA	Rotate Right Double Alpha	III-26
22X122NN	RLDA	Rotate Left Double Alpha	III-26
22X130NN	RRSA	Rotate Right Single Alpha	III-26
22X132NN	RLSA	Rotate Left Single Alpha	III-26
22X140NN	RRQAS	Rotate Right Quadruple Alpha, Set	III-26
22X142NN	RLQAS	Rotate Left Quadruple Alpha, Set	III-26
22X150NN	RRTAS	Rotate Right Triple Alpha, Set	III-26
22X152NN	RLTAS	Rotate Left Triple Alpha, Set	III-26
22X160NN	RRDAS	Rotate Right Double Alpha, Set	III-26
22X162NN	RLDAS	Rotate Left Double Alpha, Set	III-26
22X170NN	RRSAS	Rotate Right Single Alpha, Set	III-26
22X172NN	RLSAS	Rotate Left Single Alpha, Set	III-26
22X220NN	SRD	Shift Right Double	III-26
22X224NN	SRDT	Shift Right Double, Test	III-26
22X230NN	SRS	Shift Right Single	III-26
22X234NN	SRST	Shift Right Single, Test	III-26
22X320NN	RRD	Rotate Right Double	III-26
22X324NN	RRDT	Rotate Right Double, Test	III-26
22X330NN	RRS	Rotate Right Single	III-26
22X334NN	RRST	Rotate Right Single, Test	III-26
22X400NN	SRQD	Shift Right Quadruple Decimal	III-26
22X402NN	SLQD	Shift Left Quadruple Decimal	III-26

OCTAL FORMAT	MNEMONIC CODE	DESCRIPTION	PAGE
22X410NN	SRTD	Shift Right Triple Decimal	III-26
22X412NN	SLTD	Shift Left Triple Decimal	III-26
22X420NN	SRDD	Shift Right Double Decimal	III-26
22X422NN	SLDD	Shift Left Double Decimal	III-26
22X430NN	SRSD	Shift Right Single Decimal	III-26
22X432NN	SLSD	Shift Left Single Decimal	III-26
22X440NN	SRQDS	Shift Right Quadruple Decimal, Set	III-26
22X442NN	SLQDS	Shift Left Quadruple Decimal, Set	III-26
22X450NN	SRTDS	Shift Right Triple Decimal, Set	III-26
22X452NN	SLTDS	Shift Left Triple Decimal, Set	III-26
22X460NN	SRDDS	Shift Right Double Decimal, Set	III-26
22X462NN	SLDDS	Shift Left Double Decimal, Set	III-26
22X470NN	SRSDS	Shift Right Single Decimal, Set	III-26
22X472NN	SLSDS	Shift Left Single Decimal, Set	III-26
22X500NN	RRQD	Rotate Right Quadruple Decimal	III-26
22X502NN	RLQD	Rotate Left Quadruple Decimal	III-26
22X510NN	RRTD	Rotate Right Triple Decimal	III-26
22X512NN	RLTD	Rotate Left Triple Decimal	III-26
22X520NN	RRDD	Rotate Right Double Decimal	III-26
22X522NN	RLDD	Rotate Left Double Decimal	III-26
22X530NN	RRSD	Rotate Right Single Decimal	III-26
22X532NN	RLSD	Rotate Left Single Decimal	III-26
22X540NN	RRQDS	Rotate Right Quadruple Decimal, Set	III-26
22X542NN	RLQDS	Rotate Left Quadruple Decimal, Set	III-26
22X550NN	RRTDS	Rotate Right Triple Decimal, Set	III-26
22X552NN	RLTDS	Rotate Left Triple Decimal, Set	III-26
22X560NN	RRDDS	Rotate Right Double Decimal, Set	III-26
22X562NN	RLDDS	Rotate Left Double Decimal, Set	III-26
22X570NN	RRSDS	Rotate Right Single Decimal, Set	III-26
22X572NN	RLSDS	Rotate Left Single Decimal, Set	III-26

CENTRAL PROCESSOR INSTRUCTIONS

OCTAL FORMAT	MNEMONIC CODE	DESCRIPTION	PAGE
67X00000	RQSP (2a)	Request Status of Processor	III-125
67X00001	SSO (2a)	Set Status by ORing	III-131
67X00002	SSA (2a)	Set Status by ANDing	III-129
67X00003	SSL (2a)	Set Status by Loading	III-127

GENERAL INSTRUCTIONS

OCTAL FORMAT	MNEMONIC CODE	DESCRIPTION	
07XYYYYY	GEN (2a)	General	III-134
07XYYY00	RQS (2a)	Request Status	III-136
07XYYY40	RSS (2a)	Reset Status	III-137

TYPEWRITER

OCTAL FORMAT	MNEMONIC CODE	DESCRIPTION	
07XYYY00	RQST (2a)	Request Status of Typewriter	VIII-4
07XYYY01	TIO (2a)	Type Input Octal	III-157
07XYYY03	TIA (2a)	Type Input Alphanumeric	III-157
07XYYY11	TOO (2a)	Type Output Octal	III-157
07XYYY13	TOA (2a)	Type Output Alphanumeric	III-157
07XYYY40	RSST (2a)	Reset Status of Typewriter	VIII-4

CARD READER

OCTAL FORMAT	MNEMONIC CODE	DESCRIPTION	
07XYYY01	RCB (2a)	Read Card Binary	III-139
07XYYY02	RCD (2a)	Read Card Decimal	III-139
07XYYY03	RCM (2a)	Read Card Mixed	III-139

CARD PUNCH

OCTAL FORMAT	MNEMONIC CODE	DESCRIPTION	PAGE
07XYYY11	PCB (2a)	Punch Card Binary	III-140
07XYYY12	PCD (2a)	Punch Card Decimal	III-140
07XYYY13	PCE (2a)	Punch Card Edited Mode	III-140

HIGH SPEED PRINTER

OCTAL FORMAT	MNEMONIC CODE	DESCRIPTION	
07XYYY10	PRN (2a)	Print Non-Edited Mode	III-141
07XYYY11	PRNS (2a)	Print Non-Edited Mode, Slew Single	III-142
07XYYY12	PRND (2a)	Print Non-Edited Mode, Slew Double	III-142
07XYYY13	PRNT (2a)	Print Non-Edited Mode, Slew To Top of Page	III-142
07XYYY30	PRE (2a)	Print Edited Mode	III-141
07XYYY31	PRES (2a)	Print Edited Mode, Slew Single	III-141
07XYYY32	PRED (2a)	Print Edited Mode, Slew Double	III-141
07XYYY33	PRET (2a)	Print Edited Mode, Slew To Top of Page	III-141
07XYYY61	SPRS (2a)	Slew Printer Single Line	III-142
07XYYY62	SPRD (2a)	Slew Printer Double Line	III-142
07XYYY63	SPRT (2a)	Slew Printer To Top of Page	III-143

MAGNETIC TAPE

OCTAL FORMAT	MNEMONIC CODE	DESCRIPTION	
07XYYY04	RTD (2a)	Read Tape Decimal	III-144
07XYYY05	RTB (2a)	Read Tape Binary	III-144
07XYYY14	WTD (2a)	Write Tape Decimal	III-144
07XYYY15	WTB (2a)	Write Tape Binary	III-144
07XYYY44	FSR (2a)	Forward Space One Record	III-146
07XYYY45	FSF (2a)	Forward Space One File	III-146
07XYYY46	BSR (2a)	Backspace One Record	III-145
07XYYY47	BSF (2a)	Backspace One File	III-145
07XYYY54	ERS (2a)	Erase	III-146
07XYYY55	WEF (2a)	Write End of File	III-145
07XYYY61	SDL (2a)	Set Density Low	III-145
07XYYY70	RWD (2a)	Rewind	III-146
07XYYY72	RWS (2a)	Rewind and Standby	III-146

PERFORATED TAPE

OCTAL FORMAT	MNEMONIC CODE	DESCRIPTION	PAGE
07XYYY02	RPT (2a)	Read Perforated Tape	III-147
07XYYY11	PPT (2a)	Punch Perforated Tape	III-147
07XYYY13	PDT (2a)	Punch Double Character Mode Tape	III-148
07XYYY16	PST (2a)	Punch Single Character Mode Tape	III-147
07XYYY31	PET (2a)	Punch Edited Tape	III-147

MAGNETIC READER/SORTER

OCTAL FORMAT	MNEMONIC CODE	DESCRIPTION	
07XYYY01	RDOC (2a)	Read Document	III-156
07XYYY41	FDOC (2a)	Feed Document	III-156
07XYYY43	PDOC (2a)	Pocket Select	III-156

DISC STORAGE

OCTAL FORMAT	MNEMONIC CODE	DESCRIPTION	
07XYYY10	SWF (2a)	Seek/Write File	III-150
07XYYY11	SWFR (2a)	Seek/Write File and Release Seek	III-151
07XYYY12	SWFI (2a)	Seek/Write File and Increment Address	III-151
07XYYY13	SWFV (2a)	Seek/Write File and Verify	III-152
07XYYY14	SRF (2a)	Seek/Read File	III-149
07XYYY15	SRFR (2a)	Seek/Read File and Release Seek	III-149
07XYYY16	SRFI (2a)	Seek/Read File and Increment Address	III-150
07XYYY17	SCPR (2a)	Seek/Compare	III-153
07XYYY24	RB (2a)	Read Buffer	III-154
07XYYY25	RFCR (2a)	Read File Continuous and Release Seek	III-152
07XYYY30	WB (2a)	Write Buffer	III-155
07XYYY31	WFCR (2a)	Write File Continuous and Release	III-153
07XYYY32	ABA (2a)	Accept Buffer Address	III-154
07XYYY33	WFCV (2a)	Write File Continuous, Verify and Release Seek	III-153
07XYYY34	SF (2a)	Seek/File	III-149
07XYYY35	SLNK (2a)	Seek/Link	III-154
07XYYY36	LBFC (2a)	Load Buffer for Compare	III-155
07XYYY37	MVDT (2a)	Move Date	III-155
07XYYY50	WF (2a)	Write File	III-151
07XYYY51	WFR (2a)	Write File and Release Seek	III-151
07XYYY52	WFT (2a)	Write File and Increment Address	III-152
07XYYY53	WFV (2a)	Write File and Verify	III-152
07XYYY54	RF (2a)	Read File	III-149
07XYYY55	RFR (2a)	Read File and Release Seek	III-150
07XYYY56	RFI (2a)	Read File and Increment Address	III-150
07XYYY57	CPR (2a)	Compare	III-153
07XYYY75	LNK (2a)	Link	III-154

APPENDIX 4.
GE-425 INSTRUCTION TIMING

This appendix contains timing information for GE-425 instructions. Most of the basic timings are shown in alphabetic (by mnemonic code) and tabular form. Others are shown following the tables in formula form. All times shown include instruction access time as well as execution time.

The timings shown in the tables and formulas assume no address modification and, for two-address instructions, assume that the P-sequence SAS word is an Operand Pointer. If address modification is used, or if two-address instructions use a P-sequence SAS word other than an Operand Pointer, instruction timing is changed according to the factors described below.

Address Modification Timing Factors

To establish execution times for instructions involving address modification, these timing factors must be added to the basic instruction times:

Fixed Index Word:	Add 5.1 usec
Index:	Add 5.1 usec
Index Pointer:	Add 10.2 usec
Index Link:	Add 15.3 usec, if linking to an Index Pointer
	Add 10.2 usec, if linking to an Index
	Add 5.1 usec for each additional Index Link.

Second Address Sequence Timing Factors

To establish execution times for instructions requiring a Second Address Sequence, these timing factors must be used with the basic instruction times:

Operand:	Subtract 5.1 usec from time shown in tables
Operand Pointer:	No timing factor required. Listed times are for two-address instructions with Operand Pointer
Operand Link:	No timing factor required if OL leads directly to an Operand
	Add 5.1 usec, if linking to an Operand Pointer
	Add 5.1 usec for each additional Operand Link.

TIMING TABLES

INSTRUCTION		REMARKS	TIME (in μ sec)			
			FIELD LENGTH =			
			Single	Double	Triple	Quad
ABM	Add Binary to Memory	w/OP	22.0			
ABX	Add Binary to Index		16.9			
ADD	Add Decimal Double			26.7	37.5	48.3
		*		40.2	57.7	75.2
ADQ	Add Decimal Quadruple					48.3
		*				75.2
ADS	Add Decimal Single		15.9	26.7	37.5	48.3
		*	17.9	40.2	57.7	75.2
ADT	Add Decimal Triple				37.5	48.3
		*			57.7	75.2
AIM	Add Immediate to Memory	w/OP	16.9			
AIX	Add Immediate to Index		11.8			
AMD	Add to Memory Double		26.7	26.7	5.1	5.1
		*	40.2	40.2		
AMQ	Add to Memory Quadruple		48.3	48.3	48.3	48.3
		*	75.2	75.2	75.2	75.2
AMS	Add to Memory Single		15.9	5.1	5.1	5.1
		*	17.9			
AMT	Add to Memory Triple		37.5	37.5	37.5	5.1
		*	57.7	57.7	57.7	
ANM	AND to Memory	w/OP	29.0			
ANX	AND to Index		23.9			
BRC	Branch on Count	w/OP	16.9			
BRE	Branch on Equal		5.1			
BRG	Branch on Greater		5.1			
BRL	Branch on Less		5.1			
BRM	Branch if Minus		10.2			
BRU	Branch Unconditionally		5.1			
BRZ	Branch if Zero		10.2	15.3	20.4	25.5
BXC	Branch on Index Count		11.8			
CAA	Compare Alphanumeric Accumulator to Memory		15.9	26.7	37.5	48.3
CDA	Compare Decimal Accumulator to Memory	if same signs	15.9	26.7	37.5	48.3
		different signs	15.9	15.9	15.9	15.9
CMI	Compare Memory to Immediate	w/OP	18.3			
CMM	Compare Second to First Memory	w/OP	23.4			
CPO	Central Processor Operation	w/OP	16.9			
CXI	Compare Index to Immediate		13.2			
CXM	Compare Index to Memory		18.3			
EDT	Edit (See Formulas)					
EXP			15.3	23.8	32.2	40.7
GEN	General	Channel busy	16.9			
		not busy, no DT	26.4			
		not busy, DT	42.8			
HLT	Halt		5.1			

*Applicable if recomplementing is necessary

TIMING TABLES (CONTINUED)

INSTRUCTION		TIME (in μ sec)				
		REMARKS	FIELD LENGTH =			
			Single	Double	Triple	Quad
IMP	Implode		17.8	29.0	40.3	51.6
LAL	Load Accumulator Location and Length		5.1			
LDD	Load Double			25.5		
LDQ	Load Quadruple					45.9
LDS	Load Single		15.3			
LDT	Load Triple				35.7	
LDX	Load Index		16.9			
LXI	Load Index with Immediate		11.8			
MFI	Move from Immediate	w/OP	16.9			
MFM	Move from First Memory	w/OP	22.0			
MTA	Move First Address	w/OP	21.0			
MOV	Move (See Formulas)					
MXC	Move on Index Control (See Formulas)					
PXB	Program Counter to Index and Branch		11.8			
RIM	OR Inclusive to Memory	w/OP	29.0			
RIX	OR Inclusive to Index		23.9			
RXM	OR Exclusive to Memory	w/OP	29.0			
RXX	OR Exclusive to Index		23.9			
SAL	Store Accumulator Location and Length		10.2			
SBM	Subtract Binary from Memory	w/OP	22.0			
SBX	Subtract Binary from Index		16.9			
SDD	Subtract Decimal Double			26.7	37.5	48.3
		*		40.2	57.7	75.2
SDQ	Subtract Decimal Quadruple					48.3
		*				75.2
SDS	Subtract Decimal Single		15.9	26.7	37.5	48.3
		*	17.9	40.2	57.7	75.2
SDT	Subtract Decimal Triple				37.5	48.3
		*			57.7	75.2
Shift	(See Shift Table)					
SPB	Store P-Counter and Branch	w/OP	16.9			
STD	Store Double			25.5		
STQ	Store Quadruple					45.9
STS	Store Single		15.3			
STT	Store Triple				35.7	
SXA	Store Index in Address Field		15.9			
VLD	Variable Length Divide (See Formulas)					
VLD	Variable Length Multiply (See Formulas)					

*Applicable if recomplementing is necessary

TIMING FORMULAS

Edit Instruction

The two Edit formulas are expressed in terms of the following variables:

- L = length of working accumulator
- D = number of data words to be edited
- B = number of times that a new data word must be obtained when a new format word is not obtained.
- N = number of candidates for suppression

Time to Edit, no suppression (T_N):

$$T_N = 5.1 + 20.7L + 5.1D + 4.86B.$$

Time to Edit, suppression (T_S):

$$T_S = T_N + 14.31L + .8975N.$$

Normally it is not necessary to calculate precisely the time necessary to execute an Edit instruction. In such cases, the average or maximum times are sufficient:

Type of Editing	Working Accumulator			
	S	D	T	Q
No suppression, average	30.9	56.6	82.3	108.0
No suppression, maximum	30.9	61.6	92.3	123.0
Suppression, average	47.0	88.8	130.6	172.4
Suppression, maximum	48.8	97.4	146.0	194.6

Move Instruction

The formulas for calculating the execution times (in usec) for the MOV and MXC instructions are:

MOV	$T = 27.6 + 10.2N$
MXC	$T = 27.6 + 5.1N$

where N = the number of words to be moved.

VLD Instruction

Formulas are provided for calculating the average and maximum times for a VLD execution.

Average Time	$= 109.2 + 2.1Q$
Maximum Time	$= 132.9 + 2.1Q$

where Q = the value of the single quotient digit (0 through 9).

Error 1 time (divisor $<$ 100,000) = 85.0 usec

Error 2 time (divisor $\leq$ high-order 8 characters of the dividend) = 97.0 usec.

VLM Instruction

Execution time for the VLM instruction is determined by the formula:

$$T = 56.5 + 2.1M$$

where M = the value of the single digit multiplier (0 through 9).

SHIFT TIMING TABLES

Character Shift or Rotate Right

Shift Count	Working Accumulator S	D	T	Q
0	15.7	15.7	15.7	15.7
1	23.0	34.2	45.3	56.5
2	25.1	38.4	51.6	64.9
3	27.2	42.6	57.9	73.3
4	17.5	23.1	28.7	34.3
5		51.0	67.7	84.5
6		55.2	74.0	92.9
7		59.4	80.3	101.3
8		39.9	51.1	62.3
9			90.1	112.5
10			96.4	120.9
11			102.7	129.3
12			73.5	90.3
13				140.5
14				148.9
15				157.3
16				118.3

Character Shift or Rotate Left

Shift Count	Working Accumulator S	D	T	Q
0	8.1	8.1	8.1	8.1
1	27.2	42.6	57.9	73.3
2	25.1	38.4	51.7	64.9
3	23.0	34.2	45.3	56.5
4	17.5	23.1	28.7	34.3
5		59.4	80.3	101.3
6		55.2	74.0	92.9
7		51.0	67.7	94.5
8		39.9	51.1	62.3
9			102.7	129.3
10			96.4	120.9
11			90.1	112.5
12			73.5	90.5
13				137.3
14				148.9
15				140.5
16				118.3

Bit Shift Right

Shift	Working Accumulator	
Count	S	D
0	15.7	15.7
1	21.3	30.7
2	21.6	31.4
3	22.0	32.1
4	22.3	32.8
5	22.7	33.5
6	23.0	34.2
7	23.4	34.9
8	23.7	35.6
9	24.1	36.3
10	24.4	37.0
11	24.8	37.7
12	25.1	38.4
13	25.5	39.1
14	25.8	39.8
15	26.2	40.5
16	26.5	41.2
17	26.9	41.9
18	27.2	42.6
19	27.6	43.3
20	27.9	44.0
21	28.3	44.7
22	28.6	45.4
23	29.0	46.1
24	17.5	23.1
25	32.5	47.5
26	32.8	48.2
27	33.2	48.9
28	33.5	49.6
29	33.9	50.3
30	34.2	51.0
31	34.6	51.7

APPENDIX 5.
GE-435 INSTRUCTION TIMING

This appendix contains timing information for GE-435 instructions. Most of the basic timings are shown in alphabetic (by mnemonic code) and tabular form. Others are shown following the tables in formula form. All times shown include instruction access time as well as execution time.

The timings shown in the tables and formulas assume no address modification and, for two-address instructions, assume that the P-sequence SAS word is an Operand Pointer. If address modification is used, or if two-address instructions use a P-sequence SAS word other than an Operand Pointer, instruction timing is changed according to the factors described below.

Address Modification Timing Factors

To establish execution times for instructions involving address modification, these timing factors must be added to the basic instruction times:

Fixed Index Word:	Add 2.7 usec
Index:	Add 2.7 usec
Index Pointer:	Add 5.4 usec
Index Link:	Add 8.1 usec, if linking to an Index Pointer Add 5.4 usec, if linking to an Index Add 2.7 usec for each additional Index Link.

Second Address Sequence Timing Factors

To establish execution times for instructions requiring a Second Address Sequence, these timing factors must be used with the basic instruction times:

Operand:	Subtract 3.05 usec from time shown in tables
Operand Pointer:	No timing factor required. Listed times are for two-address instructions with Operand Pointer
Operand Link:	No timing factor required if OL leads directly to an Operand Add 3.05 usec, if linking to an Operand Pointer Add 3.05 usec for each additional Operand Link.

TIMING TABLES

INSTRUCTION	REMARKS	TIME (in usec)			
		FIELD LENGTH=			
		Single	Double	Triple	Quad
ABM Add Binary to Memory	w/OP	12.9			
ABX Add Binary to Index		9.9			
ADD Add Decimal Double			14.2	20.0	25.7
	*		23.1	33.3	43.5
ADQ Add Decimal Quadruple					25.7
	*				43.5
ADS Add Decimal Single		8.8	14.2	20.0	25.7
	*	10.2	23.1	33.3	43.5
ADT Add Decimal Triple				20.0	25.7
	*			33.3	43.5
AIM Add Immediate to Memory	w/OP	10.2			
AIX Add Immediate to Index		7.2			
AMD Add to Memory Double		14.2	14.2	2.7	2.7
	*	23.2	23.2		
AMQ Add to Memory Quadruple		25.7	25.7	25.7	25.7
	*	43.6	43.6	43.6	43.6
AMS Add to Memory Single		8.8	2.7	2.7	2.7
	*	10.2			
AMT Add to Memory Triple		20.0	20.0	20.0	2.7
	*	33.4	33.4	33.4	
ANM AND to Memory	w/OP	22.4			
ANX AND to Index		19.3			
BRC Branch on Count	w/OP	10.6			
BRE Branch on Equal		2.7			
BRG Branch on Greater		2.7			
BRL Branch on Less		2.7			
BRM Branch if Minus		5.4			
BRU Branch Unconditionally		2.7			
BRZ Branch if Zero		5.4	8.1	10.8	13.5
BXC Branch on Index Count		7.5			
CAA Compare Alphanumeric Accumulator to Memory		8.5	14.2	20.0	25.7
CDA Compare Decimal Accumulator	if same signs	8.5	14.2	20.0	25.7
to Memory	different signs	8.5	8.5	8.5	8.5
CMI Compare Memory to Immediate	w/OP	11.3			
CMM Compare Second to First Memory	w/OP	14.0			
CPO Central Processor Operation	w/OP	10.2			
CXI Compare Index to Immediate		8.2			
CXM Compare Index to Memory		10.9			

*Applicable if recomplementing is necessary.

TIMING TABLES (CONTINUED)

INSTRUCTION	REMARKS	TIME (in usec)			
		FIELD LENGTH=			
		Single	Double	Triple	Quad
EDT Edit (See Formulas)					
EXP		8.5	14.3	20.2	26.0
GEN General	Channel busy	10.2			
	Not busy,no DT	13.7-22.1			
	Not busy, DT	23.8-32.2			
HLT Halt		2.7			
IMP Implode		9.5	20.6	29.6	36.5
LAL Load Accumulator Location and Length		2.7			
LDD Load Double			13.0		
LDQ Load Quadruple					23.3
LDS Load Single		7.9			
LDT Load Triple				18.2	
LDX Load Index		9.9			
LXI Load Index with Immediate		7.2			
MFI Move from Immediate	w/OP	10.2			
MFM Move from First Memory	w/OP	12.9			
MTA Move First Address	w/OP	11.9			
MOV Move (See Formulas)					
MXC Move on Index Control (See Formulas)					
PXB Program Counter to Index and Branch		7.2			
RIM OR Inclusive to Memory	w/OP	22.4			
RIX OR Inclusive to Index		19.3			
RXM OR Exclusive to Memory	w/OP	22.4			
RXX OR Exclusive to Index		19.3			
SAL Store Accumulator Location and Length		5.8			
SBM Subtract Binary from Memory	w/OP	12.9			
SBX Subtract Binary from Index		9.9			
SDD Subtract Decimal Double			14.2	20.0	25.7
	*		23.1	33.3	43.5

*Applicable if recomplementing is necessary.

TIMING TABLES (CONTINUED)

INSTRUCTION		REMARKS	TIME (in usec) FIELD LENGTH= Single	Double	Triple	Quad
SDQ	Subtract Decimal Quadruple					25.7
		*				43.5
SDS	Subtract Decimal Single		8.8	14.2	20.0	25.7
		*	10.2	23.1	33.3	43.5
SDT	Subtract Decimal Triple				20.0	25.7
		*			33.3	43.5
Shift	(See Shift Table)					
SPB	Store P-Counter and Branch	w/OP	10.2			
STD	Store Double			13.0		
STQ	Store Quadruple					23.3
STS	Store Single		7.9			
STT	Store Triple				18.2	
SXA	Store Index in Address Field		8.8			
VLD	Variable Length Divide (See Formulas)					
VLM	Variable Length Multiply (See Formulas)					

*Applicable if recomplementing is necessary.

TIMING FORMULAS

Edit Instruction

The two Edit formulas are expressed in terms of the following variables:

L	=	length of working accumulator
D	=	number of data words to be edited
B	=	number of times that a new data word must be obtained when a new format word is not obtained.
N	=	number of candidates for suppression
W	=	full words not candidates for suppression
R	=	characters in partial word not candidates for suppression

Time to Edit, no suppression (T_N):

$$T_N = 2.7 + 17.05L + 2.7D + 3.05B$$

Time to Edit, suppression (T_S):

$$T_S = 2.7D + 20.1L + 3.05B + 3.15N + 9.1W + 2.1R + 3.4 \quad \text{(Subtract 0.7, if } R = 0\text{)}$$

Normally it is not necessary to calculate precisely the time necessary to execute an Edit instruction. In such cases, the average or maximum times are sufficient:

Type of Editing	Working Accumulator			
	S	D	T	Q
No suppression, minimum	22.5	39.5	56.6	73.6
No suppression, maximum	22.5	45.3	68.1	90.9
Suppression, minimum	35.7	64.9	94.1	123.3
Suppression, maximum	38.8	77.3	115.7	154.2

Move Instructions

The formulas for calculating the execution times (in usec) for the MOV and MXC instructions are:

MOV $T = 16.8 + 5.2N$
MXC $T = 13.7 + 5.2N$

where N = the number of words to be moved.

VLD Instruction

Formulas are provided for calculating the average and maximum times for a VLD execution.

Average Time $= 79.3 + 2.1Q$
Maximum Time $= 90.8 + 2.1Q$

where Q = the value of the single quotient digit (0 through 9).

Error 1 time (divisor $<$ 100,000) = 62.6 usec

Error 2 time (divisor $\leq$ high-order 8 characters of the dividend) = 74.1 usec.

VLM Instruction

Execution time for the VLM instruction is determined by the formula:

$T = 38.9 +$ the greater of 1.4 or 2.1M

where M = the value of the single digit multiplier (0 through 9).

SHIFT TIMING TABLES

Character Shift or Rotate Right

Shift Count	Working Accumulator			
	S	D	T	Q
0	10.0	10.0	10.0	10.0
1	14.8	22.7	30.7	38.6
2	16.9	26.9	37.0	47.0
3	19.0	31.1	43.3	55.4
4	9.9	12.9	16.0	19.0
5		31.9	42.9	53.9
6		36.1	49.2	62.3
7		40.3	55.5	70.7
8		22.1	28.2	34.3
9			55.1	69.1
10			61.4	77.5
11			67.7	85.9
12			40.4	49.5
13				84.4
14				92.8
15				101.2
16				64.8

Character Shift or Rotate Left

Shift Count	Working Accumulator			
	S	D	T	Q
0	5.2	5.2	5.2	5.2
1	19.0	31.1	43.3	55.4
2	16.9	26.9	37.0	47.0
3	14.8	22.7	30.7	38.6
4	9.9	12.9	16.0	19.0
5		40.3	55.5	70.7
6		36.1	49.2	62.3
7		31.9	42.9	53.9
8		22.1	28.2	34.3
9			67.7	85.9
10			61.4	77.5
11			55.1	69.1
12			40.4	49.5
13				101.2
14				92.8
15				84.4
16				64.8

Bit Shift Right

Shift	Working Accumulator	
Count	S	D
0	10.0	10.0
1	13.0	19.2
2	13.4	19.9
3	13.7	20.6
4	14.1	21.3
5	14.4	22.0
6	14.8	22.7
7	15.1	23.4
8	15.5	24.1
9	15.8	24.8
10	16.2	25.5
11	16.5	26.2
12	16.9	26.9
13	17.2	27.6
14	17.6	28.3
15	17.9	29.0
16	18.3	29.7
17	18.6	30.4
18	19.0	31.1
19	19.3	31.8
20	19.7	32.5
21	20.0	33.2
22	20.4	33.9
23	20.7	34.6
24	9.9	12.9
25	19.1	28.4
26	19.5	29.1
27	19.8	29.8
28	20.2	30.5
29	20.5	31.2
30	20.9	31.9
31	21.2	32.6

Progress Is Our Most Important Product

GENERAL ELECTRIC

COMPUTER DEPARTMENT • PHOENIX, ARIZONA

CPB-320 LITHO IN U.S.A.