GENERAL ELECTRIC
COMPUTERS

GE-115
User's Guide

ADVANCE INFORMATION

GENERAL ELECTRIC

CPB-1204A

GE-115
USER'S GUIDE

REFERENCE MANUAL

November 1965

GENERAL ELECTRIC
COMPUTER DEPARTMENT

PREFACE

This manual is a user's guide and reference manual for the GE-115 Information Processing System.

An assembly language is provided which allows for program statements in simple mnemonic and symbolic phrases. All translation to the GE-115 internal system language is performed by the assembler. The GE-115 system is described in terms of the assembly language.

A knowledge of general programming principles is helpful; no knowledge of other assembler languages or systems is required.

Terminology is according to the International Federation for Information Processing and the International Computation Centre (IFIP/ICC) Vocabulary of Information Processing.

Comments on this publication may be addressed to Technical Publications, Computer Department, General Electric Company, P. O. Box 2961, Phoenix, Arizona, 85002.

(500 3-67)

CONTENTS

APPENDICES

INTRODUCTION

The GE-115 is a small scale electronic information processing system designed to serve a wide variety of user needs.

- As a card processing system, the GE-115 may be used to perform all the tasks carried out by a punched card tabulation installation.
- As a remote terminal, the GE-115 may be used in conjunction with a large scale information processing system as a data receiving and transmitting station.
- As an information processing system, the GE-115 may be programmed for applications in all fields. It is particularly suited to the processing of data for such applications as tabulations, inventories, record keeping and file updating.

This manual describes the GE-115 information processing system with card input and output only. It may serve any of the functions listed above.

Programs for the GE-115 are written in a simple symbolic language which is easy to learn and to use. No special skill, other than a knowledge of the application to be performed, is required to use the GE-115 system effectively.

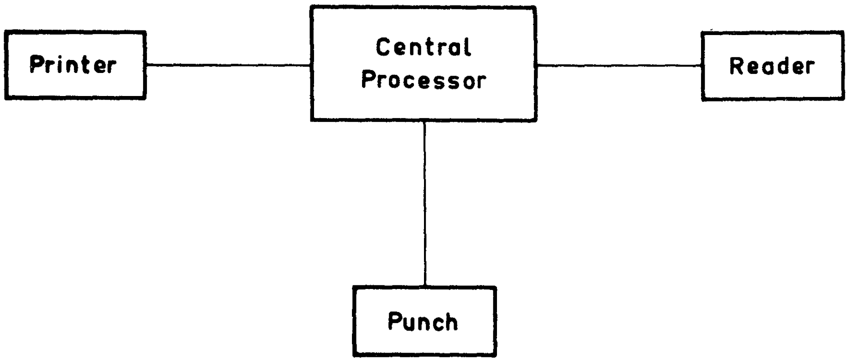
SECTION A

GENERAL INFORMATION

PART I

GE-115 SYSTEM INFORMATION

The GE-115 Information Processing System consists of a central processor and associated auxiliary store and input/output units. The minimum system configuration is shown below.



THE CENTRAL PROCESSOR

The central processor is comprised of the following units:

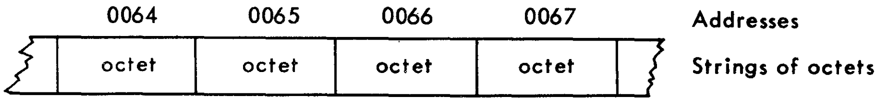
Store
Store Control Unit
Arithmetic Control Unit
Peripheral Control Unit

- Store

The store unit is an array of magnetic cores providing storage for instructions and data. A module of store contains 4096 store locations. The user may have one or two modules, that is 4096 or 8192 store locations. Information in the store is represented by the values of bits. Each bit is a binary digit and may have a value of 0 or 1.

NOTE: It is expected that the reader be familiar with the binary number system. If not, he should read Appendix C. before proceeding further in this manual.

Eight bits make up the basic unit of reference in the GE-115 system. This unit is called the OCTET. An address is used to designate each octet. The octet is the smallest addressable unit in store. Store is conventionally represented as strings of addressable octets with the addresses increasing from left to right as shown in the figure below:



Associated with each 8-bit octet is a ninth bit used by the system for parity checking. This bit does not enter into programmed operations and data values. It is used by the system to monitor its own functioning. Each time an octet is placed in store odd parity is automatically generated and stored with the octet; that is, if the number of 1-bits is even, the parity bit is set to 1. Thus all octets in store have an odd 1-bit count. This odd count is tested and if the number of 1-bits is even, a parity alert is generated and the system halts operation.

There are two special locations in the store, 0254 and 0255. These locations form a field referenced in the GE-115 Assembly Language with the name LOC. The field is used by several instructions to store an address (See "Store Control Unit" below). The significance of the address stored in 0254 and 0255 is fully explained in the descriptions of the instructions which use these locations.

These are the only special purpose locations; there are no predefined input/output areas.

- Store Control Unit

  The store control unit fetches, interprets, and controls the execution of the operations specified by the instructions. An external control panel is available to provide for manual intervention and display of internal status.

  Within the store control unit, a location counter controls the sequence of program instructions. During program execution, the location counter holds the address of the next sequential instruction in the store. When the sequence of the stored program is altered by any of the jump instructions, the new program address is entered in the location counter. The address in the location counter is displayed by lights on the control panel.

- Arithmetic Control Unit

  The arithmetic control unit controls the execution of decimal and binary arithmetic operations, logic operations, and comparisons.

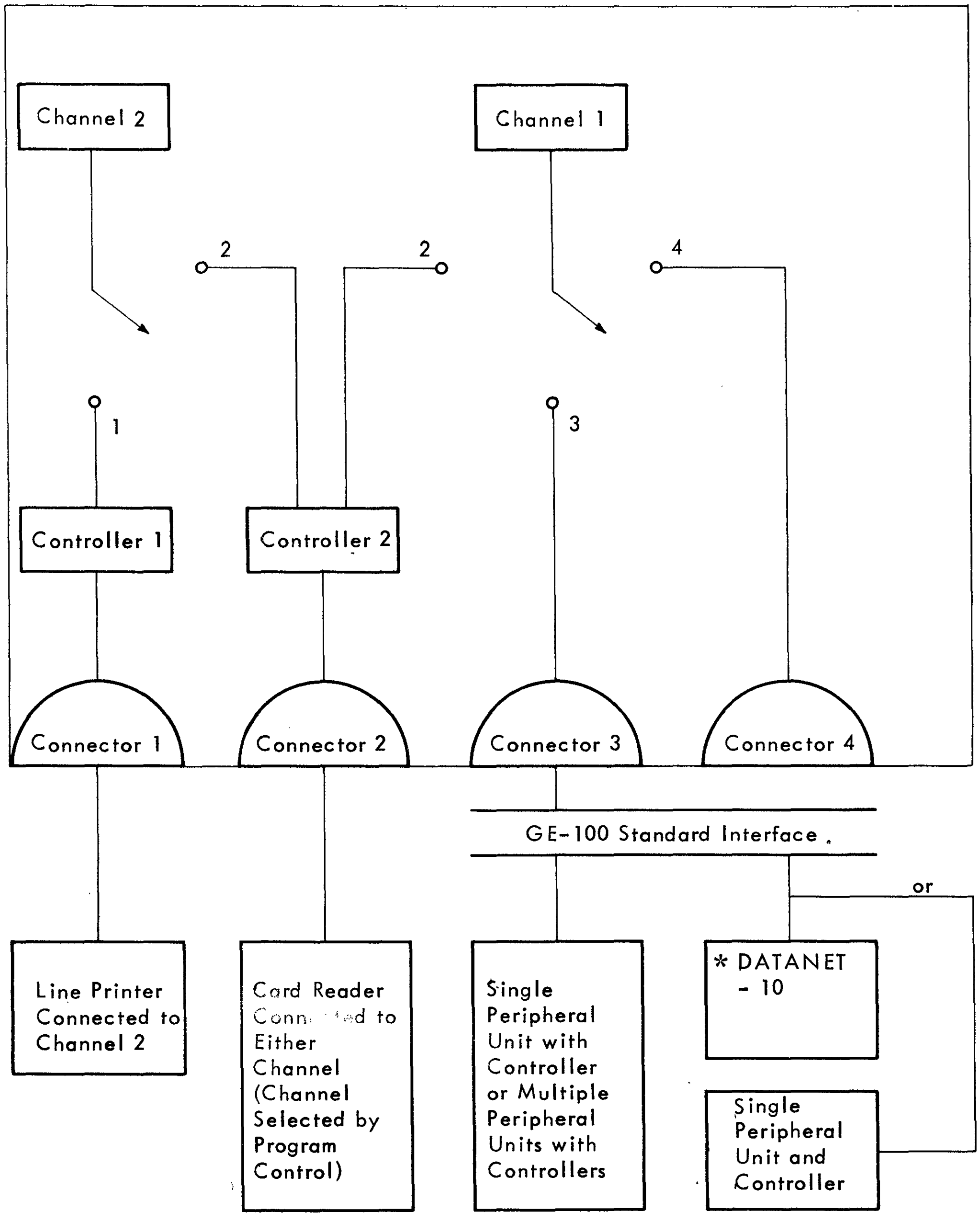
  Two 1-bit indicators reflect the results of the arithmetic, logic, and comparison operations. Jump instructions test the values in these indicators to allow alteration of program sequence. The two indicators are the Underflow/Overflow (UF/OF) Indicator, and the Zero/Non-Zero (ZE/NZ) Indicator.

  The indicators can be set to 0 or 1.

- Peripheral Control Unit

  The flow of data and instructions between the store and the input/output units is controlled by the peripheral control unit. This unit contains two channels able to operate with time-sharing of the store.

  The two channels, in turn, control four Connectors for communication with the peripheral units, as shown in Figure A-1.



* Reg. Trademark of the General Electric Company

Figure A-1

INFORMATION IN STORE

Information in store may be data or may be programmed instructions.

Information is placed in the store as binary digits, or bits.

On the following pages the formats of information in store are discussed.

Octet

Eight binary digits make up the basic unit of reference in the GE-115 Information Processing System. This unit is called an OCTET.

Each octet has its own address. An octet is the smallest addressable unit in the store.

The eight bit positions of the octet are referred to as 0-7, from right to left.

Bit position	7 6 5 4 3 2 1 0
Binary digits	0 1 0 0 1 1 1 1
	octet

The octet may be used to represent one character. Each bit of the octet may have a value of 0 or 1. There are 2^8 possible ordered combinations of binary digits in an octet, giving 256 possible internal character representations. The binary digit configuration in the example above is used internally in the GE-115 system to represent a question mark (?) character.

Quartet

The eight bits of an octet may be considered as two groups of four bits. Each group of four bits is called a QUARTET. Four bits give a set of 2^4 possible ordered combinations of 0 or 1, permitting values from 0 to 15 to be expressed.

Bit position	7 6 5 4	3 2 1 0
Binary digits	0 1 0 0	1 1 1 1
Decimal equivalent of the quartet value	4	15
	Left quartet	Right quartet
	Octet	

The decimal equivalent of the value of the left quartet of the 8-bit configuration for the question mark (?) character is 4 (four) and the decimal equivalent of the value of the right quartet is 15 (fifteen). These values are more easily represented by the HEXADECIMAL (base 16) number system.

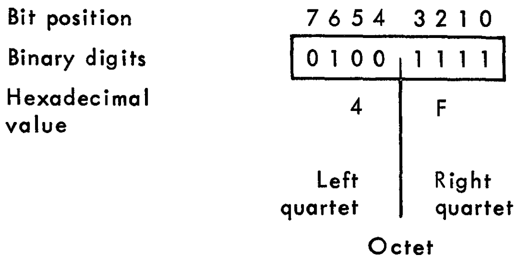
Hexadecimal Representation of the Contents of an Octet

The hexadecimal number system is to the base 16. The digits used in the hexadecimal **system** are:

0 1 2 3 4 5 6 7 8 9 A B C D E F.

The digits represent the values 0 to 15, that is, A is used to represent 10, B to represent 11, etc.

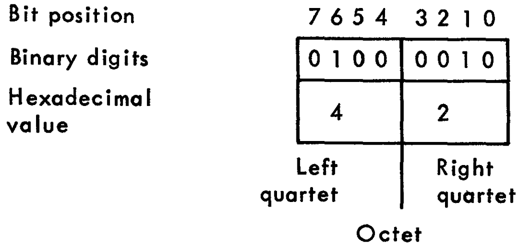
Each octet can be represented by two hexadecimal digits.



The hexadecimal representation of the 8-bit configuration for the question mark character is 4F. For convenience, the internal octet bit patterns are usually represented as two hexadecimal digits.

The pattern of the left quartet (of the left hexadecimal digit) is the zone. The pattern of the zone is identical for certain sets of characters. For example, when numeric quantities are to be represented internally using the decimal numerals 0-9, rather than their binary equivalents, all decimal digits have the value 4 (0100) in the zone position. The hexadecimal representation of the octet configurations for the decimal numerals 0 to 9 are 40 to 49. One octet contains one decimal digit. The right quartet contains the numeric value and the left the character zone.

The figure below shows the binary and hexadecimal configurations of the decimal numeral 2. Note that one octet is used to express one decimal digit or character.



PACKED DATA

Within the GE-115 system the right quartets are sometimes considered independently. To facilitate manipulation of right quartets (usually where left quartets are the same), data may be placed into octets in packed form.

To pack or condense data the like pattern in the zones of a pair of octets is omitted and the two right quartets are placed in a single octet.

Thus, if the machine recognizes

0 1 0 0	1 0 0 1

as a decimal 9, and

0 1 0 0	1 0 0 0

as a decimal 8, a packed octet containing both 8 and 9 appears as

1 0 0 0	1 0 0 1

The operations of the GE-115 system provide for condensing information in this way and expanding it again. Some operations are provided to process data in the condensed form.

It is possible to condense information with unlike zones, but the pattern must be known in order to expand the data to its original form. For example,

A		2	
0 1 0 1	0 0 0 1	0 1 0 0	0 0 1 0

in condensed form would appear

0 0 0 1	0 0 1 0

When this octet is expanded, the result would be

x x x x	0 0 0 1	x x x x	0 0 1 0

The left quartet patterns in the receiving field must be properly set by the programmer according to the intended use of the unpacked field.

THE FORMAT OF THE INTERNAL INSTRUCTION

An instruction is a statement specifying an operation to be carried out by the GE-115 system. It contains an operation code, any required constants, and references to any data fields used. The length of an instruction depends upon the operation specified, and requires two, four, or six-octets in store. Instructions can be divided into three groups according to the length of the instruction. The possible components of an instruction and resulting lengths are shown in Figure A-2.

Length in Octets	Operation	Operation Complement	Address	Address
6	▨	▨	▨	▨
4	▨	▨	▨	
2	▨	▨		
	One Octet	One Octet	Two Octets	Two Octets

Figure A-2

Figure A-3 shows the GE-115 assembly language instructions and the format of each. Note that the operation portion of an instruction requires one octet. The operation complement portion of an instruction requires one octet. Each operand address requires two octets.

Figure A-3 also shows the internal hexadecimal configuration for each operation and in some cases, the internal configuration of the operation complement. These codes are given here for the programmer's information. Although familiarity with these codes may be useful in identifying operations in an object language program listing, a knowledge of them is not required of the programmer using the GE-115 system.

It is strongly recommended that programs be written so that their logic is not dependent upon the internal instruction codes for operations.

Mnemonic	INTERNAL MACHINE INSTRUCTION CODE							
	Operation		Operation Complement		Operand Address		Operand Address	
AB	F	E			BINARY ZEROS	ALPHA Address	BINARY ZEROS	BETA Address
AD		A	L(α)	L(β)				
SB		F						
SD		B						
CMQ		9						
MVQ		8						
CMC	D	5	L(α)					
MVC		2						
NC		4						
OC		6						
XC		7						
UPK		8	L(β)					
PK		A	L(α)					
SL		B						
SR		9						
TR		C						
EDT		E						
CMI	9	5	Binary Val		BINARY ZEROS	OCTET ADDRESS		
MVI		2	Binary Val			OCTET ADDRESS		
PER		E						
JRT	4	1	F	0		GAMMA ADDRESS		
JC		3	0					
			•					
			•	all values possible				
			•					
			•					
			F					
JE			2					
JG			1					
JGE			3					
JL			C					
JLE			E					
JNE			D					
JU			F					
NOJ			0					
JS1	5		8					
JS2			4					
HLT	0	A	0	0				
NOP2		7	0					
ENS		2	1					
INS			2					
LOFF			E					
LON			8					
	1st OCTET		2nd OCTET		3rd & 4th OCTETS		5th & 6th OCTETS	

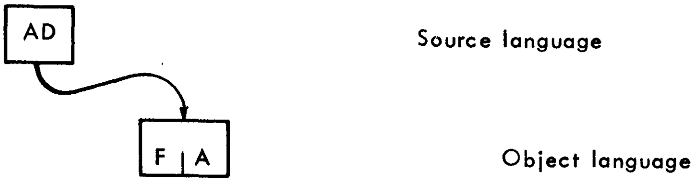
Figure A-3 : INTERNAL CONFIGURATION OF THE GE-115 INSTRUCTION SET

THE PARTS OF THE INTERNAL INSTRUCTION

The internal instruction must have an operation and an operation complement specified. It may also have one or two addresses of operands. A complete explanation of each of the internal instruction parts is given below.

The Operation

Each instruction uses one octet to define the operation to be performed. The octet contains the binary pattern translated by the assembler from the mnemonic operation specified in the Assembly Language instruction. This binary pattern is by convention represented by a pair of hexadecimal digits. For example, the mnemonic AD is translated by the assembler to a pattern which is expressed in hexadecimal as FA.



Operation Complement

The second octet is used in several different ways according to the type of operation specified in the instruction statement. In all cases the second octet complements the first. It may:

- Define the length of data fields
- Contain an immediate data item
- Define the conditions required for a jump
- Differentiate between operations having the same value in the first octet
- Contain the number of an input/output unit being used

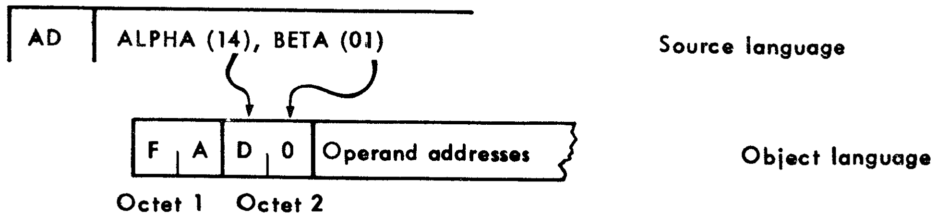
Length

Some of the GE-115 system operations may treat data fields which are more than one octet in length. There are no field defining marks in store; the length of an operation is controlled by the length (or lengths) specified in the operation complement.

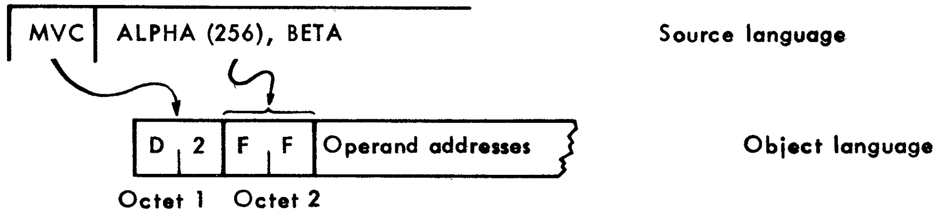
In instructions requiring two lengths for two data fields, each quartet of the operation complement specifies one length. The left quartet is the length of the first field; the right is the length of the second field. Since four bits are used for each length, the value in the quartet may range from 0000 to 1111 (0-15).

In instructions requiring that only one length be specified, the full octet of the operation complement field is used to define the length. The value for eight bits may range from 0 to 255.

In all operations which process variable length fields, the lengths really processed are one more than the lengths indicated in the operation complement. In the assembly language statement, the programmer specifies the length really processed. When two lengths must be specified these lengths may be from 01 to 16. When one length must be specified the length may be from 001 to 256. The assembler translates the lengths specified as shown in the examples below. In the first example below the Add Decimal (AD) instruction causes one decimal digit in the BETA field to be added to the decimal data 14 digits long in the ALPHA field.



The Move Complete Octets (MVC) instruction shown below causes 256 octets to be moved from BETA to ALPHA.



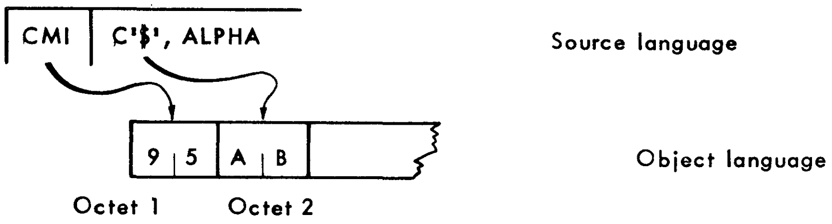
Length specification is a minimum of one octet. A zero length in the operation complement of the assembled instruction operates as 1.

Data Within the Instruction

Some operations use data contained within the instruction itself. This data is referred to as "immediate" data. In those instructions using immediate data, the operation complement field contains one data item. The assembler translates the data item from its source language representation and places it in the second octet of the object instruction. The conventions for representing the immediate data item in the assembly language instruction are treated fully in the descriptions of the particular instructions using immediate data, and in the section, "WRITING INSTRUCTION STATEMENTS IN THE GE-115 ASSEMBLY LANGUAGE".

One example of an instruction using immediate data is the Compare Immediate to Store (CMI) instruction. In the example below, the field ALPHA is compared to the internal representation of a $.

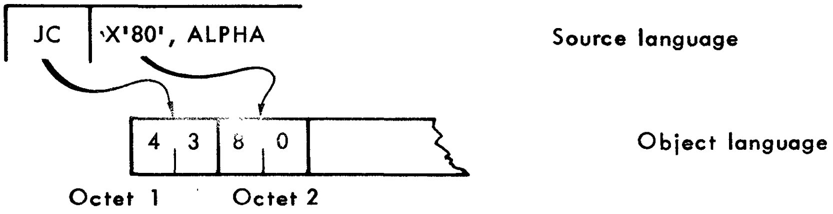
This representation is 1010 1011 (in hexadecimal, AB). The source and object language representation of the CMI instruction is:



NOTE : Instructions using immediate data also reference a data item in store (ALPHA in the example above). The assumed length of the referenced data is one octet; there is no length specified in the assembled instruction.

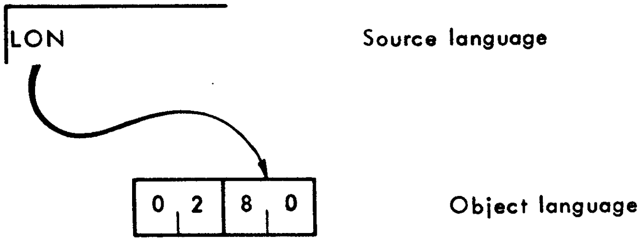
Conditions for Jumps

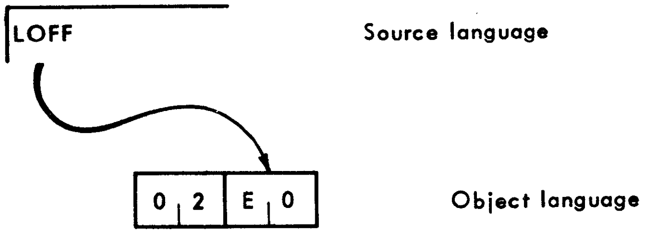
In another group of instructions, the operation complement is used to define a condition reflected by the state of the indicators. The indicators are set during execution of several of the GE-115 system operations. Instructions are provided to test the indicators and to act upon the indicated condition. The test written in the source language statement is translated by the assembler and inserted in the operation complement. In the example below, the Jump on Condition (JC) instruction causes the indicators to be tested for a condition which can be represented by the hexadecimal digits '80'. The hexadecimal digits are translated and placed in the operation complement of the assembled instruction.



Operation Differentiators

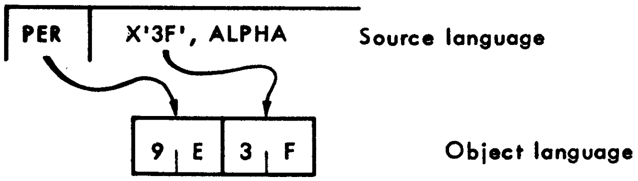
There are several special purpose instructions in which different mnemonic codes are translated into the same configuration in the first octet. The operation complement, in these cases, serves to differentiate between these operations, as shown below by the pair of instructions, Turn ALERT Light On (LON) and Turn ALERT Light Off (LOFF).





Input/Output Unit References

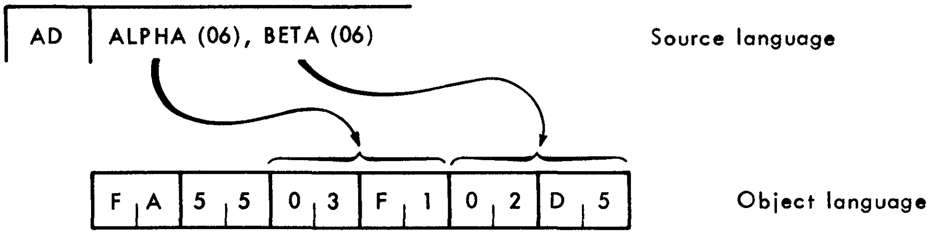
Input/output units, since they are auxiliary to the central processor, are called peripheral units. For input/output instructions, the second octet contains the number of the peripheral unit to be used by the instruction. In the example below, for the Peripheral instruction (PER), hexadecimal 3 F may be any input/output device, depending upon the peripheral configuration of the particular GE-115 system.



Addresses of Operands

An address (within the object language representation of an instruction) is contained in two octets. An address is the binary address of the octet where operation begins. When the size of the store is 8192 octets addresses go from 0000 to 8191. Therefore, within the two octets which an address occupies in the internal instruction, the three leftmost bits of the first octet are always zero.

An operation referencing only one data field requires two octets for the address of the operand and a total of four octets for the entire instruction. If there are two data field operands, the instruction requires 6 octets. In the example below, the Add Decimal (AD) operates on two operands. The operation begins in the ALPHA field at the octet with the address 00000011 11110001 and in the BETA field at the octet with the address 00000010 11010101.



PART II.

GE-115 ASSEMBLY LANGUAGE INFORMATION

An assembly language is a set of symbols and rules for writing statements to be performed by a computer. An assembly language statement is written in a format which is more convenient and easier to remember than the format for the internal instruction which the computer recognizes as an executable statement. Instead of writing the numeric values for operation codes, operand lengths, data constants, and addresses, the programmer uses assembly language instruction statements.

The GE-115 Assembly Language enables the programmer to write instructions and to specify all the required program parameters with meaningful and easily remembered codes.

The GE-115 Assembly Language provides the following features:

- Operations are specified with easy to remember alphabetic mnemonics (e.g., AD for decimal addition, PER for a peripheral unit instruction).
- Data constants may be written in various forms (hexadecimal numbers, alphanumeric characters, and special symbols as +, - $' etc.).
- Cross references between instructions and references to data fields may be accomplished with meaningful names chosen by the programmer without concern for the actual location in store of instructions and data.

Primary instructions written in the GE-115 Assembly Language are translated by the GE-115 assembler into the object language instructions acceptable for execution by the GE-115 System. Primary instructions specify the program steps, operation by operation.

Directive instructions written in the GE-115 Assembly Language are directions from the programmer to the GE-115 Assembler. Data defined in Directives will be included in the object language program. Other Directives define the assignment of store addresses and procedures for printing the assembly listing.

In this section, the symbols of the GE-115 Assembly Language are described and the rules for writing instructions in the language are given. Also, the relationship between data field references and the assembled addresses used in object language instructions is explained. This relationship is quite unusual. In most assembly languages, a symbolic name used as an operand specification references an address.
In the GE-115 Assembly Language, symbolic names used as operand specifications reference data fields.

THE SYMBOLS OF THE GE-115 ASSEMBLY LANGUAGE

The GE-115 Information Processing System recognizes the standard graphic character set defined in Figure A-4, page 19 . There are 64 graphic characters. Each number, letter or symbol in this set has a unique binary value represented internally by the eight binary digits of one octet. The internal representations of the graphic character set are only a part of the set of 256 possible binary configurations of octets.

The GE-115 System Assembler allows for the definition of data by symbolic characters, by hexadecimal digits, and by decimal digits. Binary digit patterns that are to be used for special purposes by a program must be submitted to the assembler under one of the above representations. The assembler does not recognize binary literals as data. Thus, when a programmer wishes to specify a data item with the internal configuration of 0010 0001, he may specify the hexadecimal configuration 21.

Although it is not possible to avoid specific coding of the non-graphic characters when they are required, the user is strongly advised not to construct programs in which the logic is contingent upon the internal codes of the graphic character set. Do not refer to the hexadecimal configuration of a graphic character when it is possible to use a reference to the character itself.

Binary Config.	Graphic Character	Hexadecimal	Binary Config.	Graphic Character	Hexadecimal
01000000	0	40	10100000	↑	A0
0001	1	41	0001	J	A1
0010	2	42	0010	K	A2
0011	3	43	0011	L	A3
0100	4	44	0100	M	A4
0101	5	45	0101	N	A5
0110	6	46	0110	O	A6
0111	7	47	0111	P	A7
1000	8	48	1000	Q	A8
1001	9	49	1001	R	A9
1010	[	4A	1010	-	AA
1011	#	4B	1011	$	AB
1100	@	4C	1100	*	AC
1101	:	4D	1101	)	AD
1110	>	4E	1110	;	AE
1111	?	4F	1111	' (apostrophe)	AF
01010000	(blank)	50	10110000	+	B0
0001	A	51	0001	/	B1
0010	B	52	0010	S	B2
0011	C	53	0011	T	B3
0100	D	54	0100	U	B4
0101	E	55	010	V	B5
0110	F	56	0110	W	B6
0111	G	57	0111	X	B7
1000	H	58	1000	Y	B8
1001	I	59	1001	Z	B9
1010	&	5A	1010	←	BA
1011	.	5B	1011	, (comma)	BB
1100	]	5C	1100	%	BC
1101	(	5D	1101	=	BD
1110	<	5E	1110	"	BE
1111	\	5F	1111	!	BF

Figure A-4: THE GE-115 GRAPHIC CHARACTER SET

The set of graphic characters are not only used for data definition in the GAMMA 115 **Assembly** Language.

All may be present in the coding of an assembly language program. Most of the characters in the set may be used freely. Some are reserved for special purpose. The group of symbols reserved by the assembler are

, ' ( ) + - *

These symbols must not be used by the programmer in naming data fields or instructions. (They may be used for data definition). This set of symbols is described below and in Figure A-5.

RESERVED SYMBOLS

Apostrophe (')

Two apostrophes are used as data field delimiters, one to the left and one to the right of the data being specified.

Comma (,)

A comma is used to separate operand specifications when more than one operand is specified.

Parentheses ( )

Parentheses are used to contain length definition when the specification of a length accompanies a field reference. Parentheses may also contain an address reference.

Arithmetic Signs (+ -)

Arithmetic Signs (+ or -) are used to modify a symbolic operand specification. The sign is followed by a decimal increment or decrement.

Asterisk (*)

The asterisk symbol is used in two ways.

An asterisk used for operand specifications indicates that the address of the specified operand is relative to the left octet address of the instruction in which the asterisk appears.

When an asterisk is used as an operation, it indicates that the operand specifications field contains a comment to be printed during assembly.

Figure A-5: SYMBOLS RESERVED BY THE GE-115 ASSEMBLER

SYMBOL	USE
'	Field delimiter
,	Operand specification separator
( )	Length specification or address definition
+	Increment specification
-	Decrement specification
*	As an operand specification : indicates store assignment. As an operation : indicates a comment

ALPHABETIC SYMBOLS

In addition to the above symbols, there are several alphabetic characters used by the GE-115 Assembler (A,C,X,L,R,S,D,Y). The programmer may use any of these letters in naming fields; however, it is recommended that he not use the letter Y to begin the name of subroutines, data fields, or instructions because system subroutine names begin with Y. A new system subroutine could have the same name which a programmer has defined in his program.

A

The letter A is used in a constant definition instruction to indicate an address constant.

C

The letter C in the operand specification field of an instruction indicates that the data item which follows the C, bounded by apostrophes, is a single member of the graphic set;(i.e., 0-9, A-Z, or a symbol).

C in a constant definition instruction indicates that one or more members of the graphic character set are being used to define constant data.

X

The letter X appearing in the operand specification field of an instruction indicates that the data item bounded by apostrophes is represented by a pair of hexadecimal digits ; i.e., the quartet configuration represented by each of the digits is one of those shown below :

0000 = 0
0001 = 1
0010 = 2
0011 = 3
0100 = 4
0101 = 5
0110 = 6
0111 = 7
1000 = 8
1001 = 9
1010 = A
1011 = B
1100 = C
1101 = D
1110 = E
1111 = F

X in a constant definition instruction indicates that one or more pairs of hexadecimal digits are being used to define constant data.

L

L precedes the definition of length in a store definition or constant definition instruction.

R

An R in the operand specification field of an ORG instruction sets the value in the store assignment counter to the next higher multiple of 256. The store assignment counter is a part of the assembler and is used for assigning addresses to instructions and data.

S, D

The letters S and D are used in the Line Feed (LF) as line spacers. S requests a single skip. D requests a double skip.

Y

In GE-115 system software, the letter Y has been used as the first letter of the subroutine names.

CONTROL CHARACTERS

There are three special purpose hexadecimal control characters used by the Edit (EDT) instruction to format data : 20, 21, 22. They are used in the editing mask to control data positioning. These three numbers do not correspond to any of the characters in the graphic set and must be represented in assembly language statements by pairs of hexadecimal digits.

WRITING STATEMENTS IN THE GE-115 ASSEMBLY LANGUAGE

Programs for the GE-115 Information Processing System are written in the GE-115 Assembly Language on the Programming form shown in Figure A-6.

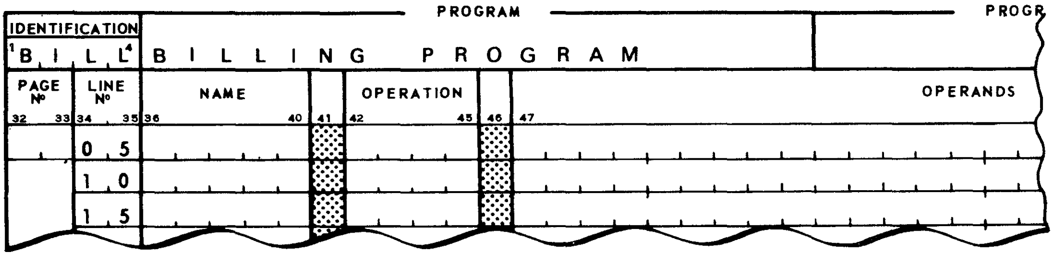
The parts of this form and the rules for their use are described below. If the programmer does not follow these rules, the GE-115 assembler will print mistake indications on the listings which are produced during assembly. A list of the possible mistake indications and their meaning is presented in Figure A-7 GE-115 ASSEMBLER MISTAKE CODES. The notations appearing in parentheses throughout the text refer to the mistakes described in this table.

THE PROGRAMMING FORM

1. IDENTIFICATION (Columns 1-4)

   Enter a 4-digit field to identify the program. Any alphanumeric combination may be used.

   This field is for program identification; therefore, it is suggested that a meaningful code be chosen. In the example below, BILL was chosen to identify a billing program.



PROGRAM — PROG

IDENTIFICATION 1 B I L L 4 — B I L L I N G P R O G R A M

PAGE № 32 33 | LINE № 34 35 | NAME 36 40 | 41 | OPERATION 42 45 | 46 | 47 OPERANDS

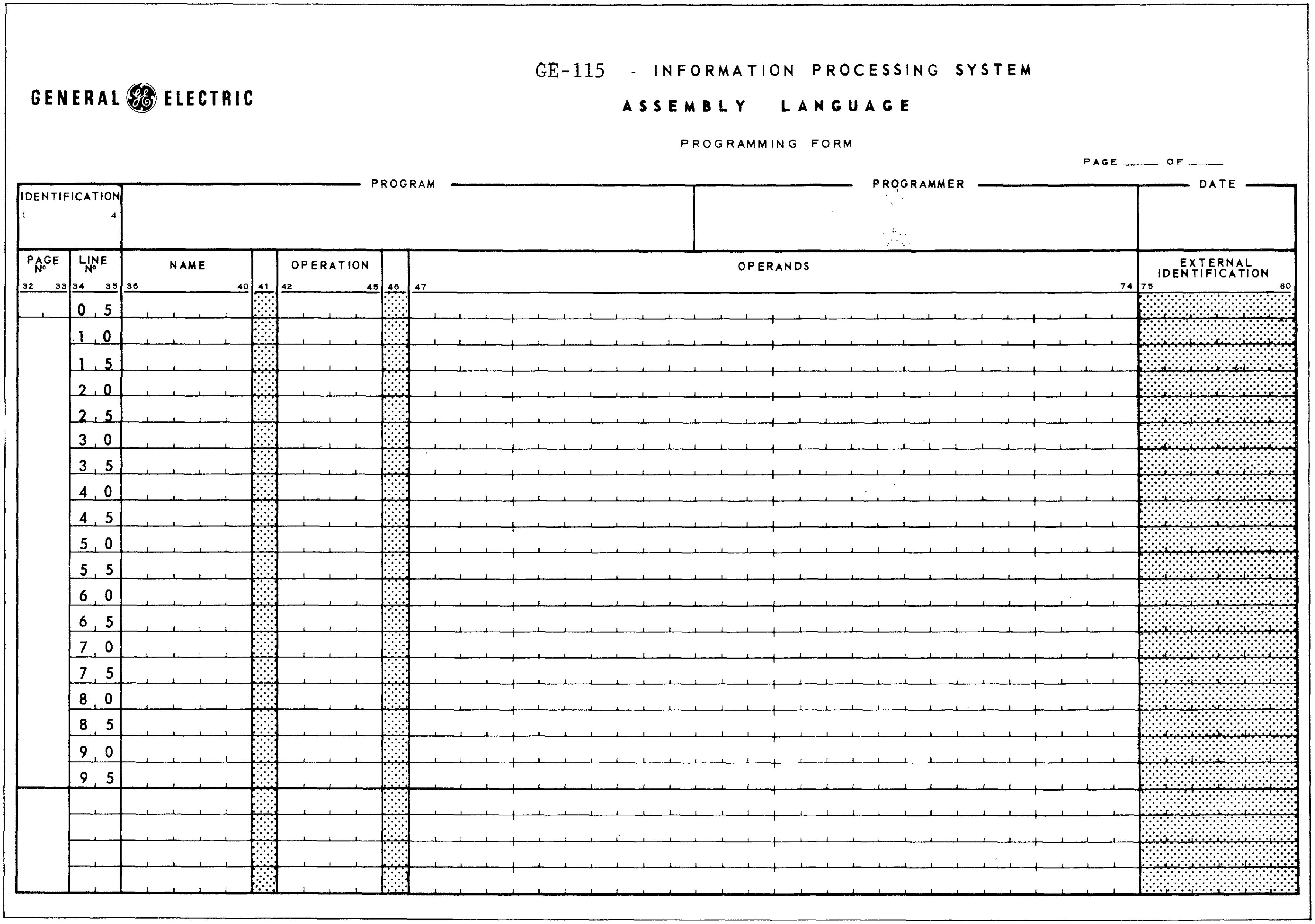
0 5

1 0

1 5

   Sequences of numbers or letters might be chosen for a set of programs which are related. A set of three programs which tabulate test scores might be identified by SCR1, SCR2, and SCR3.

   The numbers 0000 to 1000 are used for identification of system programs and should be avoided. If the identification field on any instruction is not the same as that of the first, the instruction is marked (S) by the assembler. The System Program Loader verifies (at execution time) that all cards of a program have the same identification field.



GENERAL ELECTRIC

GE-115 - INFORMATION PROCESSING SYSTEM

ASSEMBLY LANGUAGE

PROGRAMMING FORM

PAGE ____ OF ____

IDENTIFICATION 1 4	PROGRAM	PROGRAMMER	DATE

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74	EXTERNAL IDENTIFICATION 75 80
	0 5						
	1 0						
	1 5						
	2 0						
	2 5						
	3 0						
	3 5						
	4 0						
	4 5						
	5 0						
	5 5						
	6 0						
	6 5						
	7 0						
	7 5						
	8 0						
	8 5						
	9 0						
	9 5						

Figure A-6 PROGRAMMING FORM (CK-235)

2. PAGE NUMBER (Columns 32-33)

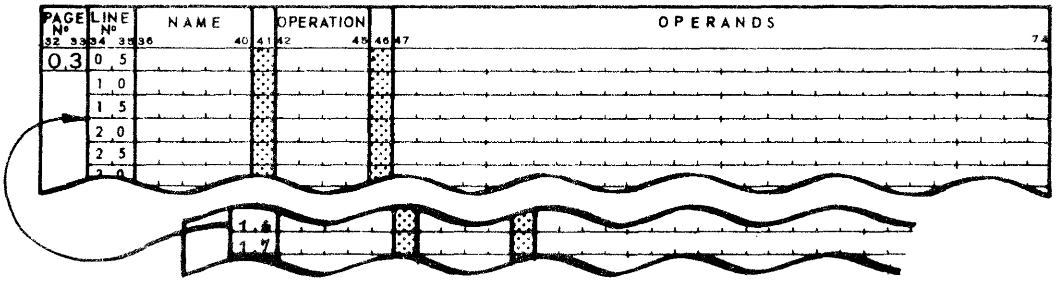
Enter a page number at the top of each page of the coding sheets. Repeat the page number on each statement when punching.

Page numbers must be in ascending order (S). The collating sequence shown in Figure 1, Appendix A, cannot be used to position non-numeric characters as sequence references.

Page and line numbers (see below for line number description) are used to order program instruction statements; they do not enter into the assembled program. It is not necessary to have a new page number for each programming sheet, nor is it necessary to use the same number for an entire sheet, as long as on each instruction the combined page and line number (taken as a 4-digit decimal number) is higher than the one before.

3. LINE NUMBER (Columns 34-35)

Enter a 2-digit decimal number for each line of coding. The numbers must be (within any page) in an ascending sequence (S). It is advisable that lines be given numbers which are multiples of 5 so that changes and corrections may be inserted without making resequencing necessary. In the following example, the lines are numbered with multiples of 5. If the programmer wishes to insert two new instructions between 15 and 20 he may number the new instructions 16 and 17.



Inserts 16 and 17 could be written on free lines at the bottom of the coding sheet and placed between 15 and 20 after they are punched.

The collating sequence shown in Figure 1, Appendix A, may not be used to position non-numeric characters. Line numbers are used to order the instructions of the program; the program sequence depends on card order in the source deck. The numbers are examined but not translated by the assembler.

4. NAME (Columns 36-40)

Names are used for cross-reference between program statements. The name of a field is defined by its appearance in the name field. The name will be equated to the actual location assigned by the assembler to the statement.

Enter a name to identify the first operation in the program.

Begin the name in column 36. Leave unused columns to the right of the name blank. Leave the name field blank if no reference to the statement is required. Column 41 must be blank.

A maximum of five characters is allowed (N). A name must begin with a letter (N). Succeeding characters may be alphabetic or numeric; no special symbols may be used (N).

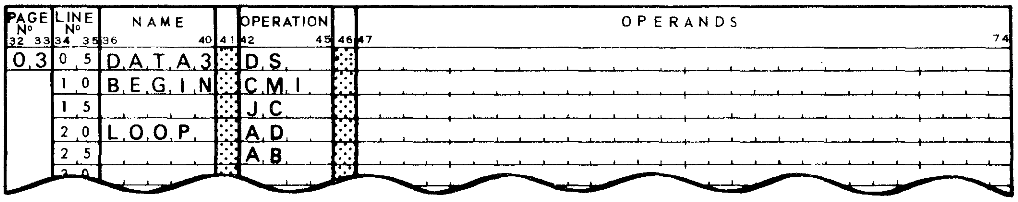
A name may appear in the name field of an APS statement only once in a program. When the same name is assigned to more than one statement, the name will appear in a multiple reference table which will be listed preceding the object program list. The location assigned to the first occurrence of the name is used for all references to the name.

System subroutine names begin with a Y. It is therefore advisable to avoid the letter Y as an initial character of a name in order to prevent duplication.

5. OPERATION CODE (Columns 42-45)

The operation code specifies the system action defined by the statement.

Enter the mnemonic for the operation, starting in column 42. Leave unused columns to the right blank. Column 46 must be blank.



PAGE N° 32 33	LINE N° 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
03	05	DATA3		DS		
	10	BEGIN		CMI		
	15			JC		
	20	LOOP		AD		
	25			AB		
	30					

The mnemonic expression must be one of those listed in Figure 3, Appendix A.

6. OPERAND SPECIFICATIONS (Columns 47-74)

An operand is the item which is operated upon by an instruction.

For example, if the number 24 is added to 92, the data items 24 and 92 are the operands. If 24 is contained in a field named (See above for naming) BETA, the instruction statement which specifies the addition operation uses the symbolic name BETA to specify that operand. Instructions in the GE-115 Assembly Language may specify one, two, or no operands depending upon the operation to be performed. The methods for specifying operands vary according to the kind of operand and the operation being specified.

When an operand is to be specified, enter the operand specification beginning in column 47. Two operand specifications must be separated by a comma (F). No blank may appear between column 47 and the end of the operand specifications (except as data definition (F,I). If an instruction does not require an operand, leave the operand specification field of the instruction blank.

Several types of operands may be specified by an instruction. The types of operand and the methods for specifying such operands are described below.

Types of Operands may be:

- Data Fields
  Data operated upon by an instruction may be elsewhere in the store, or data used by an instruction may be contained within the instruction itself. The latter type of data item is referred to in this manual as an "immediate operand" to distinguish it from data not contained within the instruction.

- Instructions
  The location of another instruction is specified as an operand when the operation may cause an interruption in the sequence of instructions executed. The location specified is the location of the instruction to which control is transferred when the sequence of instructions is interrupted. (See jump instructions, page 95)

- Hardware Items
  Operands such as input/output units, peripheral status conditions, overflow/ underflow and zero/non-zero indicator test conditions, may be specified.

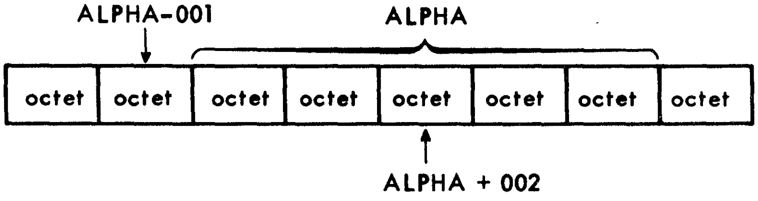
Methods of Specifying Operands are:

- Symbolic Names of Fields
  An instruction or a data field (which is not an immediate operand) may be specified

with the symbolic name of the field in which it is contained. The name must be defined (U). A symbolic name must conform to the format for names (I). In the following example the data field ALPHA is compared to a Z using the Compare Immediate to Store (CMI) instruction.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
14	05			CMI		C'Z',ALPHA
	10					
	15					
	20					
	25					
	30					

- Symbolic Names for Fields with Increments or Decrements
A symbolic name may be modified by a 3-digit decimal increment or decrement. If a programmer wishes to reference a location which has not been named, he may refer to it using the name of a location near it. For example, if ALPHA is the name of a data field of five octets and the programmer wishes to reference the third octet of ALPHA with a Compare Immediate to Store (CMI) instruction, he writes ALPHA + 002. If the programmer wishes to reference the octet to the left of ALPHA with the CMI instruction, ALPHA-001 is written.



The programmer writes an instruction using the symbolic name of the ALPHA field with an increment or decrement as follows:

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			CMI		C'$',ALPHA+002

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			CMI		C'$',ALPHA-001

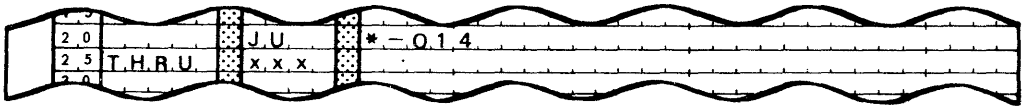
An increment or decrement must be three decimal digits (F, I). The symbols + and - are the only symbols of modification accepted by the assembler (F).

- The Asterisk (∗) Symbol

  The asterisk symbol (∗) may be used as an operand. This symbol always references the first octet of the instruction in which it appears as an operand. The ∗ symbol must be followed by an increment or decrement. When the ∗ is used without an increment or decrement, it is a mistake (I).

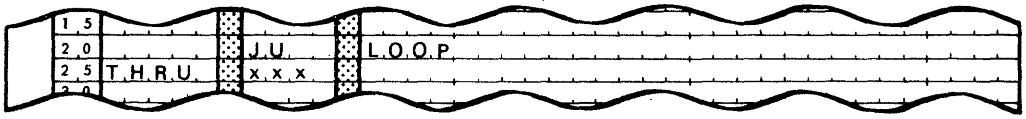
  The use of the ∗ as an operand is not recommended. Meaningful names are always easier to understand. Also mistakes can easily occur when programs are changed. For example, if another instruction were to be inserted in the following set of instructions, the programmer would have to change the ∗ -014 operand specification because the instruction named LOOP would no longer be 14 octets before the Jump Unconditional (JU) instruction.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	OPERATION 42 45	OPERANDS 47 74
04	05	LOOP	CMI	C'$',FIELD
	10		JE	THRU



	20		JU	∗-014
	25	THRU	xxx	

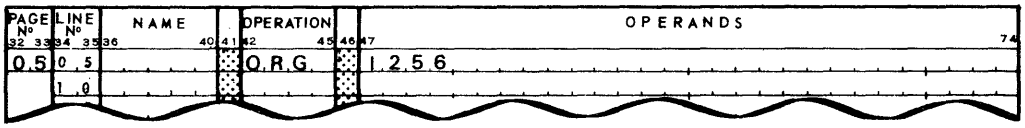
The instruction on line 20 should be written:



	15			
	20		JU	LOOP
	25	THRU	xxx	

- Absolute Addresses

  Operands may be specified with absolute location addresses. For example, if the programmer wishes to direct the assembler to set its store location assignment counter to 1256, one might code:



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	OPERATION 42 45	OPERANDS 47 74
05	05		ORG	1256
	10			

An absolute source language address used as an operand specifier must be written as a 4-digit decimal number (F, I). There cannot be an increment or decrement associated with an absolute address (F).

The programmer is advised to avoid the use of absolute address references. Names are more meaningful. The possibility of errors when a program is modified is very great when absolute address references have been used.

- Data Fields and Lengths

Operations which act upon variable length fields require a definition of the length of the field.

If the length is that of a named data field, the length need not be specified. In all other cases where a variable length operation is to be performed, length is the number of octets or quartets used in the operation.

The length is written as a decimal number enclosed in parentheses immediately to the right of the data field specification. Some operations require that the length be specified as a 2-digit decimal number. Others require that the length be specified as a 3-digit decimal number. When the number specified is not an acceptable number or it is not expressed in the correct number of digits, it is a mistake (F, I).

An example of a specified length written as three digits is:

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
04	05			MVC		ALPHA(005),BETA

An example of a specified length written as two digits is:

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
04	05			CMO		ALPHA(04),BETA(04)
	10					

Note that although only four octets enter into the last example, the number is expressed in two digits. It is written 04, not 4.

In instructions which operate on variable length fields, the length must always be specified (F) if the operand is referenced with either the * or an absolute address.

- Immediate Data Items

Immediate data items may appear in the operand specification portion of the source language instruction. The data item becomes a part of the object language instruction. The assembler, when it translates the instruction, places the data item in the operation-complement octet.

There are three ways in which immediate data may be specified. Immediate data may be coded as a character, a hexadecimal number, or a decimal number.

A single symbol preceded by C and enclosed in a pair of apostrophes signifies an immediate data item which is a character. The character must be one of those shown in Figure A-4, page 19 (F,I).

PAGE No 32 33	LINE No 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
05	05	TEST	▨	CMI	▨	C'A',ALPHA
	10		▨		▨	

A pair of hexadecimal numbers preceded by X and enclosed in a pair of apostrophes signifies an immediate data item which is any valid hexadecimal configuration up to 'FF'. The user should refer to "SYMBOLS OF THE GE-115 ASSEMBLY LANGUAGE", Figure A-4. The hexadecimal numbers and their binary, decimal, and character equivalents may be found in Figure I, Appendix A (I). It is recommended that the programmer use the C notation (see above) for any character in the graphic character set instead of the hexadecimal configuration.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
05	05	TEST		CMI		X'21',ALPHA

A 3-digit decimal number with a value from 000 to 255 may be used to specify an immediate data item. No field definers are used (F,I).

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
05	05	TEST		CMI		081,ALPHA

- Conditions of the Indicators
  Indicators are tested by the Jump on Condition instructions. The test conditions of the indicators may be specified in three ways. The condition may be written as a 2-digit hexadecimal number, a single character, or a 3-digit decimal number. The methods for coding the test conditions are the same as for immediate data items (see above). It is recommended that in Jump on Condition instructions, the X' ' form with two hexadecimal digits be used to specify conditions.

- Input/Output Units and Status
  These two operand specification types should be specified with a pair of hexadecimal digits. The rules for coding these are the same as for the X' ' form for specifying immediate data items (see above).

- Comments
  The programmer may write a comment following the last operand specification. A comment must be separated from the last operand by a blank. If the user has

placed an asterisk in the operation code, a comment may begin in column 47. Comments for one of the examples used above might be written as shown below to provide explanation and readability in a program assembly listing.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
04	05	LOOP		CMI		C'$',FIELD TEST FOR A $
	10			JE		THRU IF FOUND, THRU
	15			AB		LOOP,BIN1 IF NOT, INCRMT
	20			*		ADDRESS FOR
	25			*		SEARCH AND
	30			JU		LOOP CONTINUE
	35			*		
	40			*		** SEGMENT 2 **
	45	THRU		LON		SIGNAL OPERATOR

Figure A-7 : GE-115 . ASSEMBLER MISTAKE CODES

Code Letter	General Cause	Assembler Action
F	A mistake has been made in the format of the operand field specification. An unexpected configuration has been encountered by the assembler; e.g. a blank where a comma should appear.	The statement is marked with an F on the assembler listing. The program is not assembled.

Examples of mistakes that are marked with F:

- <u>Data Field Reference</u>

  An absolute address is written in more than 4 digits.

  An address increment or decrement is written in more than 3 digits.

  A character other than + or - appears between a data field name and increment or decrement.

  An immediate data item expressed in the C' ' notation is more than a single character in length.

  An immediate data item expressed in the X' ' notation is more than 2 digits in length.

- <u>Length</u>

  A left parenthesis is omitted.

  An absolute address is written without a length specification.

  A length is written for an operand which has an implicit length of 1.

  Three digits are used to specify a length which should be expressed in 2 digits.

Code Letter	General Cause	Assembler Action
I	A mistake has been made in the content of the operand field specification.	The statement is marked with an I on the assembler listing. The program is not assembled.

Examples of mistakes that are marked with an I:

- <u>Data Field Reference</u>

  A data field name begins with some symbol other than a letter.

  A special system symbol is used in a data name.

  A data field name is expressed in more than 5 letters.

  An address increment is written in fewer than 3 digits.

  A data item is written where a data name should appear.

  A data item expressed in the C' ' notation is written without one of the apostrophe signs.

  A data item is written without either C or X notation.

  A character other than one of the hexadecimal digits appears in the expression of a data item in the X' ' form.

- <u>Length</u>

  A two digit length is written with a value greater than 16.

  A three digit length is written with a value greater than 256.

  The right parenthesis is not written after the length specification.

Figure A-7: GE-115 ASSEMBLER MISTAKE CODES (cont'd)

Code Letter	General Cause	Assembler Action
L	An incorrect length is associated with a data field.	The statement is marked with an L on the assembler listing. Assembly continues.

Mistakes marked with an L:

The address origin defined on an ORG card is less than the upper limit of the area used by the system loader.

The program cannot be executed by a system of the size defined on the STRT card.

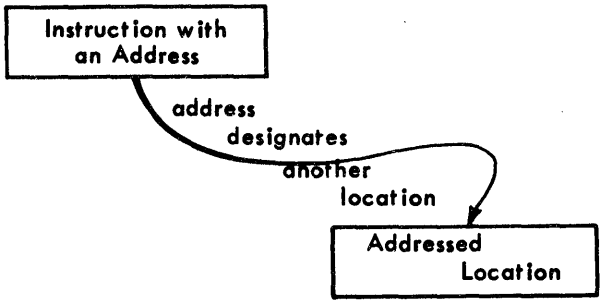
An implicit length exceeds 16. NOTE: The assembler places a value in the length field and continues. This value is generated by translation. Therefore, the value inserted differs according to the configuration of the store.

Code Letter	General Cause	Assembler Action
M	The same name is written for more than one source instruction.	The name will appear in a multiple reference table which will be listed preceding the object list. The store location assigned to the first occurence of the name is used throughout the assembly for all references.
N	The name of a field is longer than 5 characters. The name of a field contains a non-alphanumeric character. The name begins with a non-alphabetic character.	The statement is marked with an N on the assembler listing. The program is not assembled.
O	The operation code field is blank. The operation code contains some expression other than one of the assembly language mnemonic codes.	The statement is marked with an O on the assembler listing. The program is not assembled.
P	When an operation code was encountered for location assignment, the store assignment counter was set to an odd octet value.	The statement is marked with a P on the assembler listing. The location is rounded up to an even octet boundary. The assembly continues.
S	The program identification on an instruction is different from the identification on the STRT instruction. Page numbers are not in ascending sequence. Line numbers are not in ascending sequence.	The statement is marked with an S on the assembler listing. The assembly continues.
U	A name which appears in the operand field of an instruction cannot be matched with a name in the list of named fields.	The statement is marked with an U on the assembler listing. The operand is assigned a location of 0000 and an assumed length of 00. The assembly continues.

REFERENCING DATA FIELDS IN THE GE-115 ASSEMBLY LANGUAGE

The programmer who has not already programmed using the GE-115 Assembly Language should read this section very carefully. This section explains the relationship between the reference to a data field in the GE-115 Assembly Language and the address translated by the assembler.

Each octet in the store has a unique address. An address in a GE-115 system instruction can reference any octet individually, giving access for processing or control. The function of addressing is shown by the following diagram:



In an instruction in the GE-115 System Assembler Language, an operand specification may be a symbolic name or an actual reference to a particular location.

Specification of operands with symbolic names can be of three types:

1. A name assigned to a field (data or instruction) by the programmer; or,
2. A name (as in 1) modified by a 3-digit decimal increment or decrement; or,
3. An * (signifying the first octet of the instruction in which it appears), modified by a 3-digit decimal increment or decrement.

NOTE: An * must have an increment or decrement; it cannot appear alone.

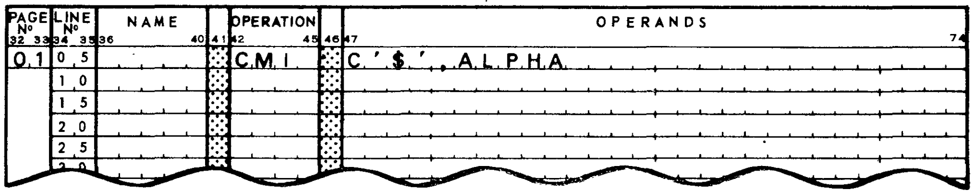
Symbolic names used as operand specifications are translated into actual addresses at assembly time. The actual address is the binary address of the particular octet referenced by the operation.

Actual addresses used for operand specification are 4-digit decimal numbers which reference explicit locations. The assembler converts the decimal address specified by the programmer to its binary equivalent and inserts this into the object language instruction.

When writing in the GE-115 Assembly Language, the programmer must be aware of the relationship between a field reference written in a source language instruction statement and the address which the assembler places in the object language instruction.

When an instruction is translated by the assembler, the operation mnemonic is translated and placed in the first octet of the object language instruction; the operation complement (immediate data, lengths, indicator test conditions, peripheral units, etc.) is placed in the second octet; the address of one operand is placed in the third and fourth octets, and, if there is a second operand, a second address is placed in the fifth and sixth octets.

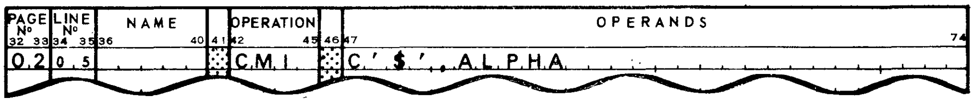
Any data field one octet in length can be referenced with the address of the octet it occupies. When an operand which is one octet in length is referenced by a symbolic name in a source language instruction statement, the address for the octet is placed in the object language instruction. In the following example, the field ALPHA (defined elsewhere in the program as a one octet field) is compared to a dollar sign. The symbolic name used as an operand specification is translated into the address of the ALPHA octet.



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			CMI		C'$',ALPHA
	10					
	15					
	20					
	25					

Data fields which occupy more than one octet may be thought of as having a left-octet address and a right-octet address. When a data field which is more than one octet in length is referenced in an instruction statement, the assembler may place either the left-octet address or the right-octet address of the data field in the object language instruction. The address which is translated for any symbolic name depends upon the operation specified.

When a symbolically named field longer than one octet is referenced as an operand and the operand can only be one octet in length, the left-octet address of the field is placed in the object language instruction (the leftmost octet is the one which is operated upon).



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			CMI		C'$',ALPHA

If the field ALPHA referenced by the Compare Immediate to Store (CMI) instruction in the example above is defined as a field of four octets, the assembler places the address of the leftmost octet of ALPHA in the third and fourth octets of the object language instruction which is produced.

When a symbolically named field is referenced as an operand and the operand may have a length greater than one octet, the address placed in the object language instruction depends upon the orientation of the operation.

Some of the GE-115 system operations which treat variable length fields process data from left to right. For example, the Move Complete Octets (MVC) begins by moving the leftmost octet. The MVC operation continues moving octets until the rightmost octet of the specified field has been moved.

When an operand is to be processed from left to right, the left-octet address of the symbolically referenced field is placed by the assembler in the object language instruction.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			MVC		ALPHA,BETA

Some of the GE-115 system operations which treat variable length fields process data from right to left. For example, the Add Binary (AB) begins addition at the rightmost octet of the two fields being summed. The operation continues to the left until the summation of the fields is completed.

When an operand is to be processed from right to left, the right-octet address of the symbolically referenced field is placed in the object language instruction.

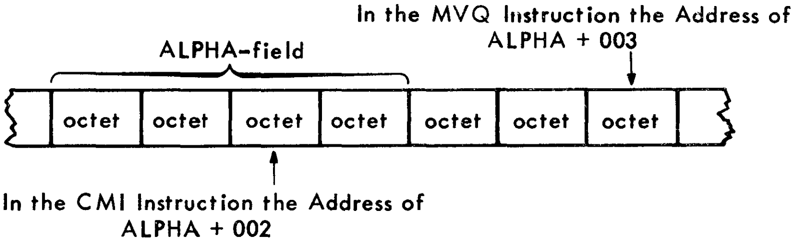
Wherever possible, the programmer should use symbolic names for operand specifications. When the programmer references fields which have been symbolically named, he does not have to be concerned with the translation of the addresses of a data field.

The assembler builds a table of the names defined in a program. The length of the data field or instruction which the name references, along with both the left-octet address and the right-octet address, are contained in this table. The assembler translates a symbolic name to the appropriate address by using this table.

When an operand is specified with a symbolic name and an increment or decrement, the increment or decrement is applied to the left-octet or right-octet address translated.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			CMI		C'$',ALPHA+002

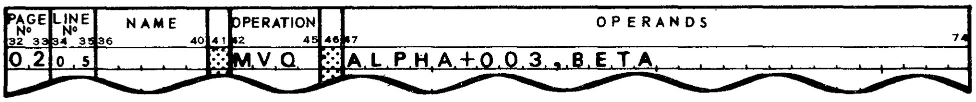
When ALPHA has been defined as a field of four octets, the symbolic reference ALPHA + 002 in the instruction above will cause the assembler to use the address of the third octet in the ALPHA field. The CMI instruction can reference a one-octet operand; ALPHA becomes the left-octet address of ALPHA and the address translated for ALPHA + 002 is two octets to the right of this address.



In the MVQ Instruction the Address of ALPHA + 003

In the CMI Instruction the Address of ALPHA + 002

When the operation proceeds from right to left, increments or decrements with a symbolic operand specification are applied to the address of the rightmost octet of the data field. Thus, the symbolic address ALPHA + 003 in the instruction:



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			MVQ		ALPHA+003,,BETA

where ALPHA has been defined as a field of four octets (see diagram above) will cause the assembler to use the address of the octet three locations to the right of the rightmost octet of ALPHA.

The programmer should note that if he uses source language actual addresses to specify operands, he must be sure that the address specified is the appropriate address according to the orientation of the operation. It is strongly recommended that the programmer use symbolic field references rather than specific addresses.

SECTION B

GE-115

ASSEMBLY LANGUAGE

INSTRUCTIONS

PART I

PRIMARY INSTRUCTIONS

Primary instructions specify machine-executable operations. Primary instructions are written as symbolic statements and translated by the GE-115 Assembler. The assembler produces one machine instruction for each Primary instruction.

Primary instructions are written according to the rules presented in SECTION A, PART II, "WRITING STATEMENTS IN THE GE-115 ASSEMBLY LANGUAGE". The programmer should review these rules before using the information presented in this section. He should also be familiar with the relationship between a Primary instruction and its machine instruction counterpart (discussed in SECTION A, PART I, "INTERNAL INSTRUCTION FORMAT").

All the Primary instructions of the GE-115 system are described in this section. The descriptions of the Primary instructions are grouped according to similarities of the operation, as listed below:

ARITHMETIC - Instructions which perform addition or subtraction on binary or decimal data fields:

Add Decimal	AD
Subtract Decimal	SD
Add Binary	AB
Subtract Binary	SB

DATA MOVEMENT AND COMPARISON - Instructions which perform non-arithmetic manipulations or comparisons of data fields:

Move Immediate Octet	MVI
Move Complete Octets	MVC
Move Right Quartets	MVQ
Pack Right Quartets into Octets	PK
Unpack Octets into Right Quartets	UPK
Compare Immediate Octet to Store	CMI
Compare Complete Octets	CMC
Compare Right Quartets	CMQ
Search to the Right	SR
Search to the Left	SL

LOGIC - Instructions which perform 'and' and 'or' logical operations:

And on Complete Octets	NC
Or on Complete Octets	OC
Exclusive Or on Complete Octets	XC

JUMP - Instructions which can be used to interrupt the sequential operation of the program:

Jump on Condition	JC
Jump if Greater	JG
Jump if Equal	JE
Jump if Greater or Equal	JGE
Jump if Less	JL
Jump if Not Equal	JNE
Jump if Less or Equal	JLE
Jump Unconditional	JU
No Jump	NOJ
Jump if Switch 1 Set	JS1
Jump if Switch 2 Set	JS2
Jump and Return	JRT

EDIT - Instructions which prepare data for system use and output readability:

Edit	EDT
Translate	TR

SYSTEM ACTION - Instructions which do not treat data but allow for manual intervention:

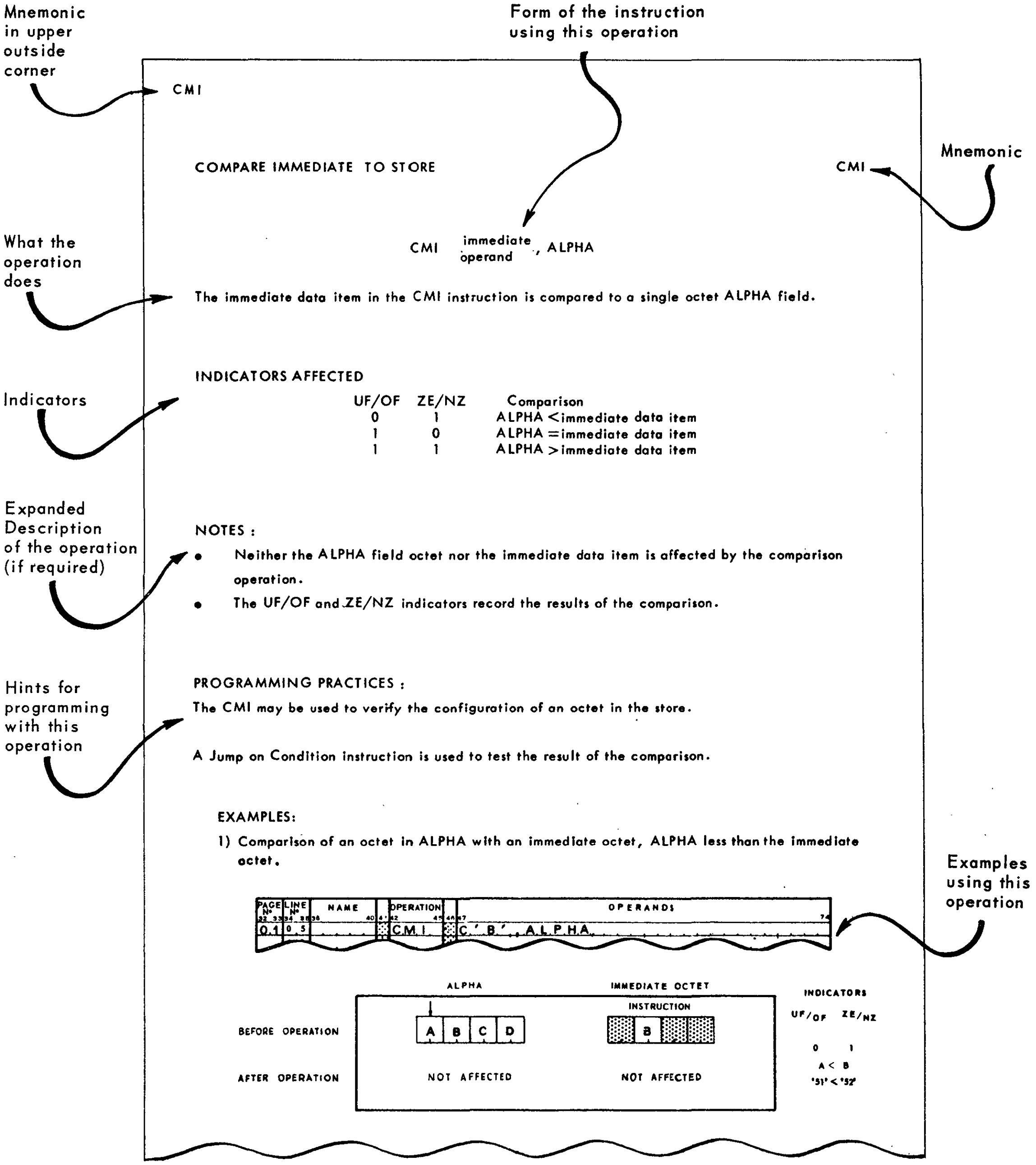
Halt System Operation	HLT
No Operation	NOP2
Turn Alert Light On	LON
Turn Alert Light Off	LOFF
Inhibit Single Stop	INS
Enable Single Stop	ENS

INPUT/OUTPUT - Instructions which execute read/write operations and test the status of the peripheral units:

Data Transfer	PER
Peripheral Status Tests	PER
Peripheral Unit Control	PER

A uniform format, as shown below, is used to explain each of the Primary instructions. For easy reference, each instruction is begun on a new page. Whenever an instruction requires more than one page for description, the mnemonic appears in the upper outside corner of each page.

Figure B-1: A SAMPLE PAGE



CMI

COMPARE IMMEDIATE TO STORE

CMI

CMI immediate operand, ALPHA

The immediate data item in the CMI instruction is compared to a single octet ALPHA field.

INDICATORS AFFECTED

UF/OF	ZE/NZ	Comparison
0	1	ALPHA <immediate data item
1	0	ALPHA =immediate data item
1	1	ALPHA >immediate data item

NOTES :

- Neither the ALPHA field octet nor the immediate data item is affected by the comparison operation.
- The UF/OF and ZE/NZ indicators record the results of the comparison.

PROGRAMMING PRACTICES :

The CMI may be used to verify the configuration of an octet in the store.

A Jump on Condition instruction is used to test the result of the comparison.

EXAMPLES:

1) Comparison of an octet in ALPHA with an immediate octet, ALPHA less than the immediate octet.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
0 1	0 5			CMI		C'B', ALPHA

In the descriptions of the Primary instructions, certain conventions have been used. These conventions are explained below.

- Conventions of notation in the general examples of instruction formats have been used. These are:

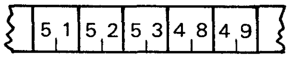
name	The use of "name" written in lower case indicates that a name is optional and where used must follow the rules for naming instructions.
op	The use of "op" written in lower case indicates that the operation is any one of a given set of operation mnemonics.
ALPHA	The symbolic name ALPHA is used to refer to the first of two operands or, where there is only one operand, the single operand. Where ALPHA is written, the programmer may write: 1) a symbolic name, with optional increment or decrement, 2) an asterisk, with a required increment or decrement, 3) an actual address.
BETA	The symbolic name BETA is used to refer to the second operand in instructions requiring two operands. BETA may be written in any of the ways listed above for ALPHA.
SIGMA	The symbolic name SIGMA is the name of the instruction to which control may be transferred by a Jump instruction. SIGMA may be written in any of the ways listed above for ALPHA.
(nn) (nnn)	The use of "n" written in lower case indicates that where the operation may process variable length fields, length is specified with two or three decimal digits. When a symbolic name is used and the length of the operation is the length associated with the definition of the name, length need not be specified in the instruction.
immediate operand	The use of the words "immediate operand" written in lower case indicates that any of the three methods for specifying immediate operands may be used. Refer to the rules in "WRITING STATEMENTS IN THE GE-115 ASSEMBLY LANGUAGE", for the three ways in which an immediate data item may be specified. One of these must be used.

condition	The use of "condition" written in lower case indicates that in the Jump on Condition (JC) instruction, a condition must be specified. It may be written in any of the ways in which an immediate data item is written.
U	**The letter U is used to refer to a peripheral unit. It may be specified in any of the ways in which an immediate data item is written. The use of the hexadecimal notation is recommended.**

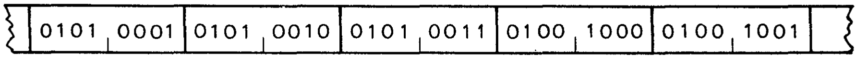
- Conventions of notation for showing data in store in the EXAMPLES portion of the descriptions vary according to the type of data being represented.

Data may be represented as characters,

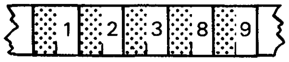
or as pairs of hexadecimal digits,



or in binary form.



Where only right quartets are involved, right hexadecimal digits (usually decimal values) are shown and left quartets are shaded.



ARITHMETIC INSTRUCTIONS

The GE-115 adds and subtracts in both the decimal mode and the binary mode. All arithmetic operations treat the data fields as unsigned quantities.

Both decimal and binary operations have the following general characteristics :

- Data fields may be from 1 to 16 octets in length. The length of each field is used in the operation.
- Operation length is governed by the length of the data field which receives the result. .
- Data fields are referenced at the right ; the data value is assumed positioned to the right in the field.
- Operation is right to left.
- The UF/OF (Underflow/Overflow) indicator is set to 1 when a carry is generated out of the result field. The result replaces the first data field.
- The ZE/NZ (Zero/Non-Zero) indicator records whether the value of the result field is zero or non-zero at the end of the operation.

Decimal and Binary operations are different in the following characteristics :

Quartets/Octets

- Decimal operations process only the right quartets of the data field.
- Binary operations process full octets.

FORM OF DATA

- Decimal operations are designed for use with decimal quantities and process data as unsigned quantities to the base 10. No check is made prior to processing to determine whether the fields to be operated upon do contain decimal configurations in the right quartets.
- Binary operations are designed for use with binary quantities and are used in the GE-115 Information Processing System primarily for address modification. Binary operations treat data as unsigned numeric quantities to the base 2.

OVERFLOW AND UNDERFLOW IN ARITHMETIC OPERATIONS

In both modes of addition, it is possible to generate an overflow. When the result field contains fewer digit places than are required to represent the sum, an overflow (or carry) occurs. The sum is not fully represented in the result field in such cases and is referred to in the discussions which follow as being in overflow form.

In both modes of subtraction, it is possible to subtract a larger quantity from a smaller. In this case the opposite of the overflow condition is present. The condition is called underflow.
The difference which occurs when a larger number is subtracted from a smaller is represented in what is defined as complement form, i.e., represented as subtracted from a power of the base of the number system being used.
For example, 6 and 4 are mutual complements in the decimal system.

$$\begin{array}{rrrr} 6 & 4 & 10 & 10 \\ +4 & +6 & -\ 6 & -\ 4 \\ \hline 10 & 10 & 4 & 6 \end{array}$$

So, too, are 60 and 40. However, 60 and 40 are expressed as multiples of 10 and require the square of 10 as a reference value for obtaining the complement.

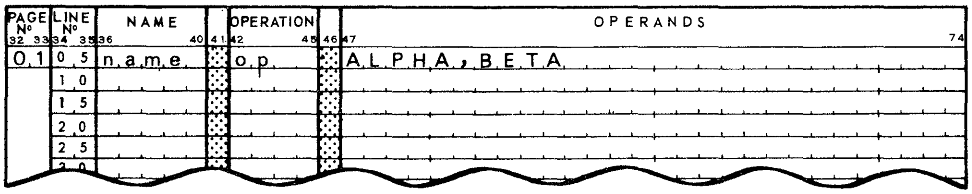
$$\begin{array}{rrrr} 60 & 40 & 100 & 100 \\ +40 & +60 & -\ 60 & -\ 40 \\ \hline 100 & 100 & 40 & 60 \end{array}$$

In the use of the arithmetic operations for the GE-115 the differences which are computed when a larger quantity is subtracted from a smaller are in complement form. They must be subtracted from the applicable power of the base used in order to obtain the true difference. The true difference also may be obtained by subtracting an underflow result from a field of zeros equal in length to the underflow result. For example, to obtain the complement of a result of 40 in a 2 digit field the subtraction appears :

$$\begin{array}{r} 00 \\ -\ 40 \\ \hline 60 \end{array}$$

Results which cause neither overflow or underflow are referred to in the descriptions of the arithmetic instructions which follow as being in true form.

The format of the arithmetic instruction is :



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05	name		op		ALPHA,BETA
	10					
	15					
	20					
	25					
	30					

ADD DECIMAL AD

AD ALPHA (nn),BETA (nn)

The unsigned sum of the right quartets of the ALPHA field and the right quartets of the BETA field replaces the right quartets of the ALPHA field. Operation is right to left, through the length of the ALPHA field (01 - 16 octets).

INDICATORS AFFECTED

UF/OF	ZE/NZ	
0	0	No overflow; result is zero.
0	1	No overflow; result is non-zero.
1	0	Overflow; result is zero.
1	1	Overflow; result is non-zero.

NOTES:

- Operation is serial, octet by octet, from right to left, through the ALPHA and BETA fields.
- Operation is terminated when the ALPHA field has been processed. If the fields are of equal length, all right quartets of BETA are added to all right quartets of ALPHA.
  If the length of the BETA field is greater than the length of the ALPHA field, the excess right quartets in the left of the BETA field do not enter into the addition.
  If the length of the BETA field is less than the length of the ALPHA field, zero right quartets are added to the excess quartets in the left of the ALPHA field.
  Whenever the generated sum of two quartets is ten or greater, it is reduced by 10 and a carry is propagated to the next quartet sum.
- A 1 in the UF/OF indicator at the end of the operation indicates that the ALPHA field is not long enough to contain the result and a carry out of the sum field has been developed.
- A 0 in the UF/OF indicator at the end of the operation indicates that the sum is contained in the ALPHA field.
- The ZE/NZ indicator is set to 0 if the result is zero, and to 1 if the result is non-zero.

- The left quartets in both fields are unaffected by the operation.
- The ALPHA field right quartets are replaced by the sum.
- The BETA field right quartets are unaffected unless some part of the ALPHA field lies in the BETA field.

  The presence of a non-decimal configuration in any right quartet of either field does not alter the above sequence of operations.

PROGRAMMING PRACTICES :

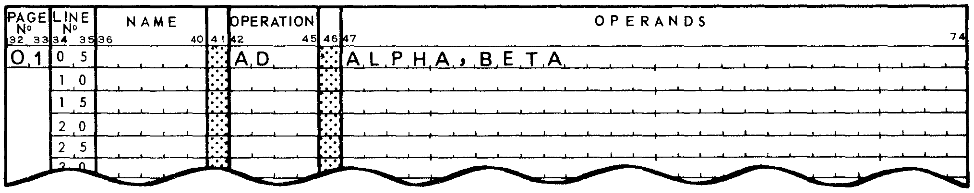
The AD is designed for use with decimal data. No check is made of the configuration of the right quartets prior to the operation. It is not recommended that the programmer use the AD operation to process data that is not decimal.

The programmer should define the ALPHA field long enough to contain the sum.

When the relative magnitudes are not known , the UF/OF indicator should be tested by a Jump on Condition (JC) instruction to determine whether overflow has occurred.

EXAMPLES :

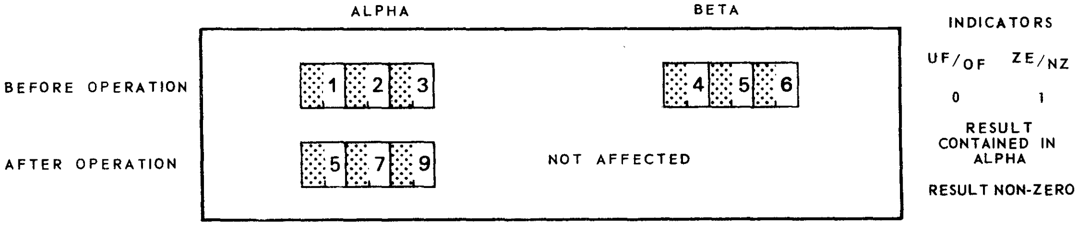
1) Addition without Overflow.



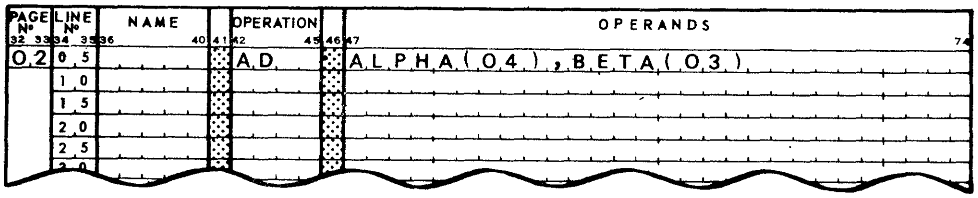
PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			AD		ALPHA, BETA
	10					
	15					
	20					
	25					
	30					

ALPHA has a defined length of 3 octets.

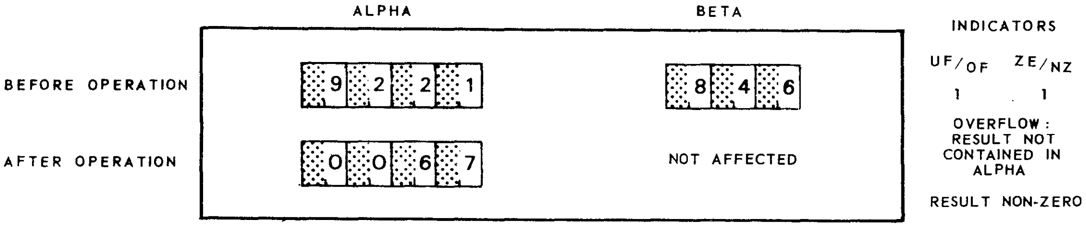
BETA has a defined length of 3 octets.



2) Addition with overflow.



PAGE N° 32 33	LINE N° 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			AD		ALPHA(04),BETA(03)
	10					
	15					
	20					
	25					
	30					



	ALPHA	BETA
BEFORE OPERATION	9 2 2 1	8 4 6
AFTER OPERATION	0 0 6 7	NOT AFFECTED

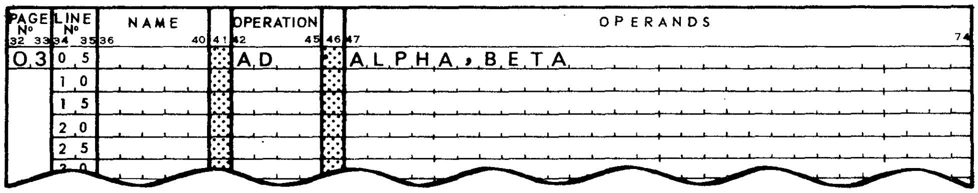
INDICATORS

UF/OF	ZE/NZ
1	1

OVERFLOW: RESULT NOT CONTAINED IN ALPHA

RESULT NON-ZERO

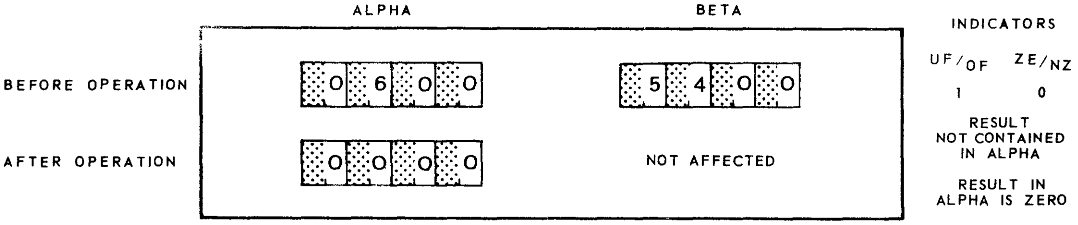
3) Addition with overflow and zero result.



PAGE N° 32 33	LINE N° 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
03	05			AD		ALPHA,BETA
	10					
	15					
	20					
	25					
	30					

ALPHA has a defined length of 3 octets

BETA has a defined length of 4 octets, but only 3 enter the operation, because the length of the operation is the length of ALPHA.



	ALPHA	BETA
BEFORE OPERATION	0 6 0 0	5 4 0 0
AFTER OPERATION	0 0 0 0	NOT AFFECTED

INDICATORS

UF/OF	ZE/NZ
1	0

RESULT NOT CONTAINED IN ALPHA

RESULT IN ALPHA IS ZERO

When the length of the BETA field is greater than the length of the ALPHA field the extra digits of the BETA field do not enter into the sum.

SUBTRACT DECIMAL SD

SD ALPHA (nn), BETA (nn)

The unsigned difference of the right quartets of the ALPHA field and the right quartets of the BETA field replaces the right quartets of the ALPHA field. Operation is right to left, through the length of the ALPHA field (01-16 octets).

INDICATORS AFFECTED

UF/OF	ZE/NZ	
0	0	Does not occur when decimal values are used.
0	1	Underflow - a larger number subtracted from a smaller ; result is non-zero.
1	0	No underflow ; result is zero.
1	1	No underflow ; result is non-zero.

NOTES :

- Operation is serial, octet by octet, through the ALPHA and BETA fields.
- Operation is terminated when the ALPHA field has been processed. If the fields are of equal length, all right quartets of the BETA field are subtracted from all right quartets of the ALPHA field

  If the length of the BETA field is greater than the length of the ALPHA field, the excess right quartets in the left of the BETA field do not enter into the subtraction. If the length of the BETA field is less than the length of the ALPHA field, zero right quartets are subtracted from the excess right quartets in the left of the ALPHA field.
- Subtraction is performed by addition. The BETA field right quartet bits are inverted and added to the bits of the ALPHA field right quartets.

  The UF/OF indicator is set to 1 prior to the operation to develop a carry into the first sum.

  Whenever a sum which is generated for a quartet exceeds 15 (a full quartet of 1's), the UF/OF indicator is set to 1 to develop a carry into the next right quartet sum. When no carry occurs, the sum is increased by 10. No carry is propagated from this second sum.
- A 1 in the UF/OF indicator at the end of the operation indicates that the difference is represented in true form in the ALPHA field.

- A 0 in the UF/OF indicator at the end of the operation indicates that the difference is represented in underflow form in the ALPHA field. (Underflow occurs when a larger number is subtracted from a smaller number.)
- The ZE/NZ indicator is set to 0 if the result is zero ; it is set to 1 if the result is non-zero.
- The left quartets in both fields are unaffected by the operation.
- The ALPHA field right quartets are replaced by the difference.
- The BETA field right quartets are unaffected unless some part of the ALPHA field lies in the BETA field.
- The presence of a non-decimal configuration in any of the right quartets of the operand fields does not alter the above sequence of operations.

PROGRAMMING PRACTICES

The SD operation is designed for use with decimal data. No check is made of the configuration of the right quartets prior to the operation. It is not recommended that the programmer use the SD instruction with data that is not decimal.

When the relative magnitudes of the quantities to be subtracted are not known, the UF/OF indicator should be interrogated to determine whether underflow has occurred.

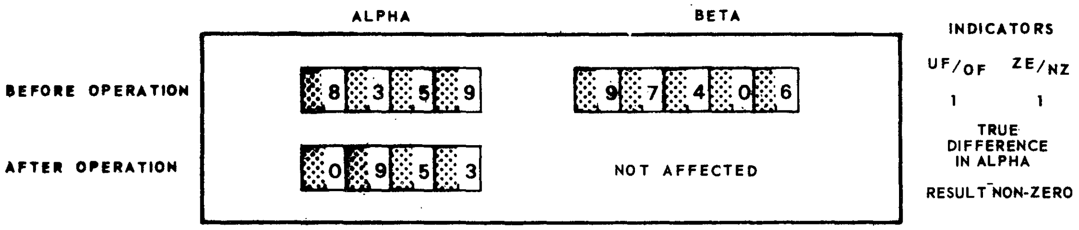
When the difference is represented in underflow form a second subtraction is required to compute the true difference. The underflow result is subtracted from a field of zero. No test is required after the second subtraction because the result is known.

EXAMPLES :

1) Subtraction of a smaller number from a larger.

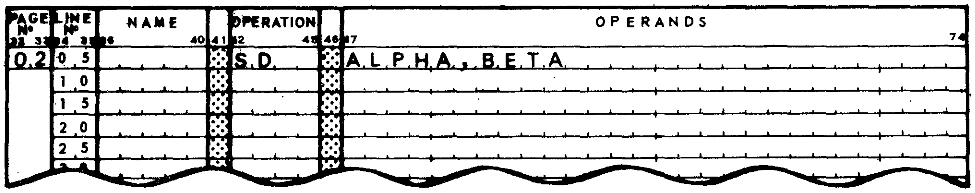
PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			SD		ALPHA, BETA
	10					
	15					
	20					
	25					

ALPHA has a defined length of 4 octets.
BETA has a defined length of 5 octets, but only 4 are used, because the length of the operation is determined by the length of the ALPHA field.



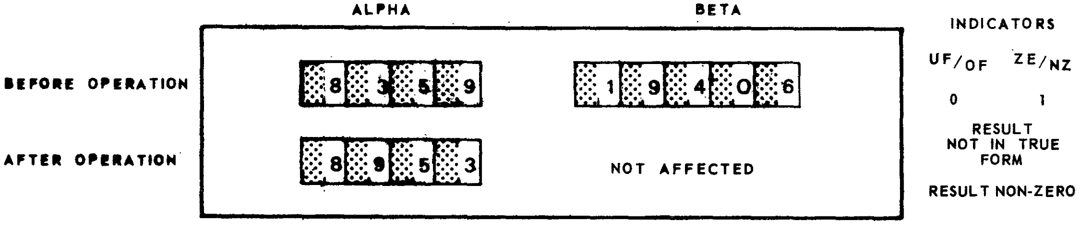
When the length of the BETA field is greater than the length of the ALPHA field, the extra digits in the left of the BETA field do not enter into the operation.

2) Subtraction of a larger number from a smaller.

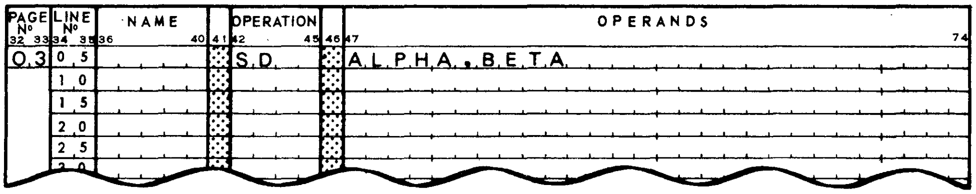


PAGE Nº	LINE Nº	NAME		OPERATION		OPERANDS
0 2	0 5			S D		ALPHA,BETA
	1 0					
	1 5					
	2 0					
	2 5					

ALPHA has a defined length of 4 octets.
BETA has a defined length of 5 octets, but only 4 enter the operation, because the length of the operation is the length of the ALPHA field.



3) Subtraction of a result not in true form from a field of zeros to obtain the true difference.

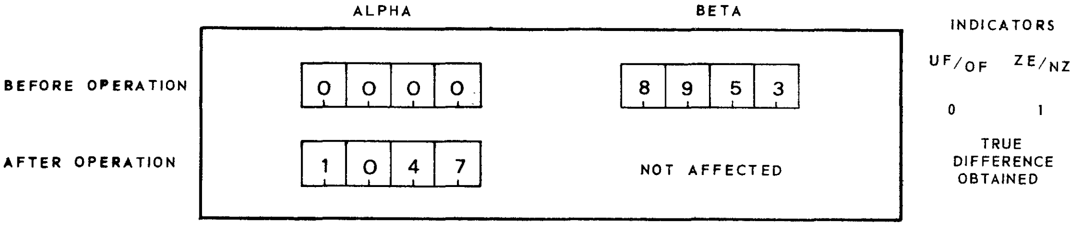


PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	OPERATION 42 45	OPERANDS 47 74
03	05		SD	ALPHA, BETA
	10			
	15			
	20			
	25			

To obtain a true result in this case, the length of the two fields should be equal. If ALPHA is longer than BETA, the excess digits in the left of the ALPHA field will contain erroneous values. The indicators need not be tested in this case, as the result is known.

ALPHA has a defined length of 4 octets.

BETA has a defined length of 4 octets.



Note that a result not in true form may be added to a positive number to produce :

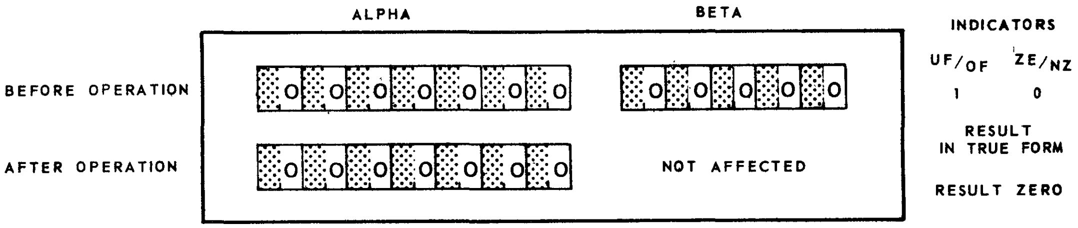
1) a true sum, in which case the UF/OF indicator is 1 after the AD operation,
2) a result not in true form in which case the UF/OF indicator is 0 after the AD operation (See Add Decimal).

4) Subtraction of zero from zero.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
04	05			SD		ALPHA,BETA
	10					
	15					
	20					
	25					
	30					

ALPHA has a defined length of 7 octets.

BETA has a defined length of 5 octets.



ADD BINARY AB

AB ALPHA(nn),BETA(nn)

The unsigned sum of the octets of the ALPHA field and octets of the BETA field replaces the octets of the ALPHA field. Operation is right to left, through the length of the ALPHA field (01 - 16 octets).

INDICATORS AFFECTED

UF/OF	ZE/NZ	
0	0	No overflow; result is zero.
0	1	No overflow; result is non-zero.
1	0	Overflow; result is zero.
1	1	Overflow; result is non-zero.

NOTES:

- Operation is serial, octet by octet, from right to left, through the ALPHA and BETA fields.
- Operation is terminated when the ALPHA field has been processed. If the fields are of equal length, all octets of the BETA field are added to all octets of the ALPHA field.
  If the length of the BETA field is greater than the length of the ALPHA field, the excess octets in the left of the BETA field do not enter into the addition.
  If the length of the BETA field is less than the length of the ALPHA field, zero octets are added to the excess octets in the left of the ALPHA field.
- The ALPHA field is replaced by the sum.
- The BETA field is unaffected unless some part of the ALPHA field lies in the BETA field.
- A 1 in the UF/OF indicator at the end of the operation indicates that the ALPHA field is not long enough to contain the result and a carry out of the sum field has been developed.
- A 0 in the UF/OF indicator at the end of the operation indicates that the sum is contained in the ALPHA field.
- The ZE/NZ indicator is set to 0 if the result is zero, and to 1 if the result is non-zero.

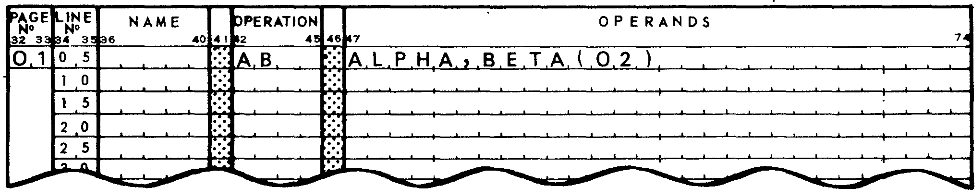
PROGRAMMING PRACTICES:

Unless the relative magnitudes of the quantities being added are known, the indicator should be tested to determine whether overflow has occurred. A Jump on Condition (JC) instruction is used to test the indicators.

The AB may be used to perform address modification. Care must be taken to avoid generating an address outside store limits. Addresses use the rightmost 13 bits of the 16 bits in the 2 octet address field of an instruction. The bits to the left can be affected by an overflow without an indication of overflow out of the octet being processed. When an address is used by the GE-115 system, the leftmost three bits of the address are not used. Thus, a value of 4096 in a store of 4096 positions references location 0. This is a valid reference and there is no immediate indication of error. Program results are unpredictable in such cases.

EXAMPLES:

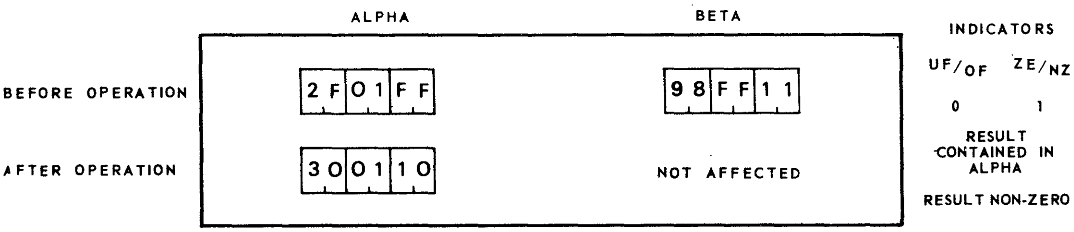
1) Binary addition without overflow.



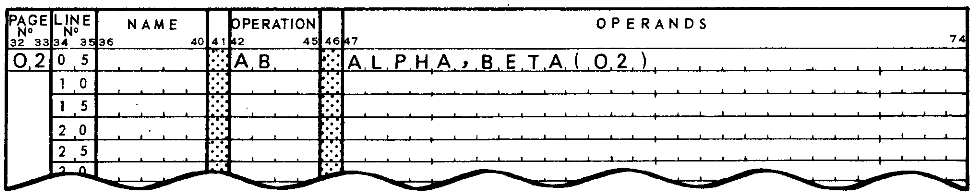
PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			AB		ALPHA,BETA(02)
	10					
	15					
	20					
	25					

ALPHA has a defined length of three octets.

BETA has a defined length of three octets, but only two are used.

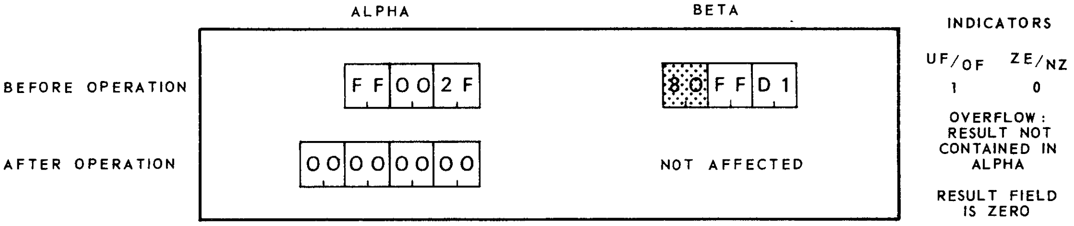


2) Binary addition with overflow and zero result.

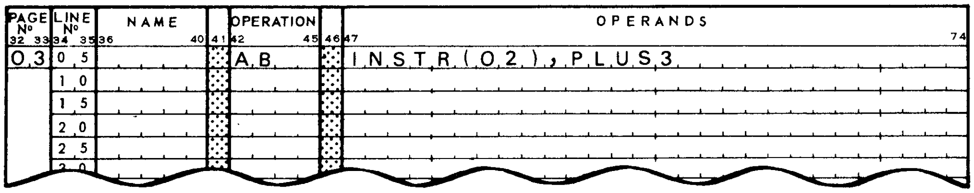


ALPHA has a defined length of 3 octets.

BETA has a defined length of three octets, but only two are used.

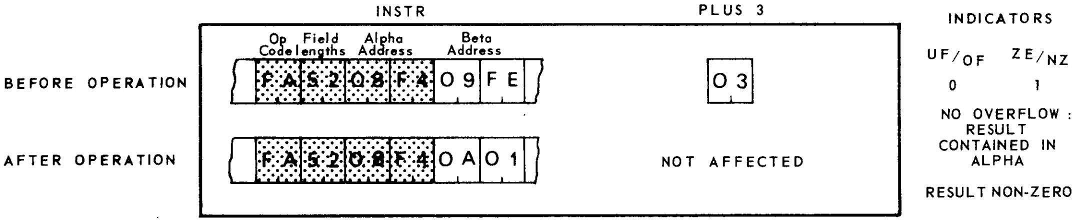


3) Address modification.



INSTR is an AD instruction of 6 octets referencing 2 data fields. It is desired to modify the address specified for the BETA field of INSTR, in order to execute it again referencing the modified address, to sum the elements of field BETA. BETA is made up of 50 three-digit decimal numbers stored sequentially in the field.

PLUS 3 is a defined constant with a length of 1 octet, having the hexadecimal value 03.



SUBTRACT BINARY SB

SB ALPHA (nn), BETA (nn)

The unsigned difference of the octets of the ALPHA field and the octets of the BETA field replaces the octets of the ALPHA field. Operation is right to left, octet by octet, through the length of the ALPHA field (01 - 16 octets).

INDICATORS AFFECTED

UF/OF	ZE/NZ	
0	1	Underflow; result is non-zero.
1	0	No underflow; result is zero.
1	1	No underflow; result is non-zero.

NOTES:

- Operation is serial, octet by octet, from right to left through the ALPHA and BETA fields.
- Operation is terminated when the ALPHA field has been processed.

  If the fields are of equal length, all octets of the BETA field are subtracted from all octets of the ALPHA field.

  If the length of the BETA field is greater than the length of the ALPHA field, the excess octets in the left of the BETA field do not enter into the subtraction.

  If the length of the BETA field is less than the length of the ALPHA field, zero octets are subtracted from the excess octets in the left of the ALPHA field.
- The ALPHA field is replaced by the difference.
- The BETA field is unaffected unless some part of the ALPHA field lies in the BETA field.
- Subtraction is carried out by the addition of the complement of the BETA field octet to the ALPHA field octet.

  The UF/OF indicator is set to 1 prior to the first addition to develop a carry into the first sum.
- A 1 in the UF/OF indicator at the end of the operation indicates that the difference is represented in the ALPHA field in true form.
- A 0 in the UF/OF indicator at the end of the operation indicates that the difference is represented in the ALPHA field in underflow form.
- The ZE/NZ indicator is set to 0 if the result is zero and to 1 if the result is non-zero.

PROGRAMMING PRACTICES:

Unless the relative magnitudes of the quantities being subtracted are known, the indicator should be tested to determine whether underflow has occurred. When underflow has occurred a second subtraction is necessary to recover the true difference. The complemented difference in the ALPHA field must be subtracted from a field of equal length and zero value. No test is necessary after the second subtraction because the result is known. A Jump on Condition (JC) instruction is used to test the indicators.

The SB may be used to perform address modification. Care must be taken to avoid generating an address outside store limits. Addresses use the rightmost 13 bits of the 16 bits in the 2-octet address field of an instruction. The bits to the left of the 13 used can be affected by an underflow. When an address is used by the GE-115 system, the leftmost three bits of the address are not used. Thus, a value of 4096 in a store of 4096 positions references location 0. This is a valid reference and there is no immediate indication of error. Program results are unpredictable in such cases.

EXAMPLES:

1) Subtraction of a smaller number from a larger.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			SB		ALPHA(03),BETA(03)
	10					
	15					
	20					
	25					
	30					

	ALPHA	BETA	INDICATORS UF/OF	ZE/NZ
BEFORE OPERATION	FEDCBA	987654	1	1
AFTER OPERATION	666666	NOT AFFECTED	TRUE DIFFERENCE IN ALPHA	RESULT NON-ZERO

2) Subtraction of a larger number from a smaller.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			SB		ALPHA(03),BETA(03)
	10					
	15					
	20					
	25					
	30					

	ALPHA	BETA
BEFORE OPERATION	98 76 54	FE DC BA
AFTER OPERATION	99 99 9A	NOT AFFECTED

INDICATORS

UF/OF	ZE/NZ
0	1

RESULT IN UNDERFLOW FORM

RESULT NON-ZERO

3) Subtraction of a result in underflow form from a field of zeros to obtain a true difference.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
03	05			SB		ALPHA(03),BETA(03)
	10					
	15					
	20					
	25					
	30					

	ALPHA	BETA
BEFORE OPERATION	00 00 00	99 99 9A
AFTER OPERATION	66 66 66	NOT AFFECTED

INDICATORS

UF/OF	ZE/NZ
0	1

INDICATOR RESULT KNOWN

DATA MOVEMENT AND COMPARISON INSTRUCTIONS

The GE-115 Information Processing System performs two types of data manipulation operations. The operations performed are data movement and comparison.

All data movement operations have the following general characteristics:

- Items from one data field are moved into another.
- The configuration of the field is ignored for the purpose of movement; all moves operate on any valid store configuration.
- The first data field is replaced by the second.

All comparison operations have the following general characteristics:

- Data items from two fields are compared.
- The results of the comparison are recorded in the indicators. The Search to the Right and the Search to the Left instructions use only the ZE/NZ to record results. The remaining comparisons use both the UF/OF and the ZE/NZ indicators.
- The compared fields are not altered by the comparison.

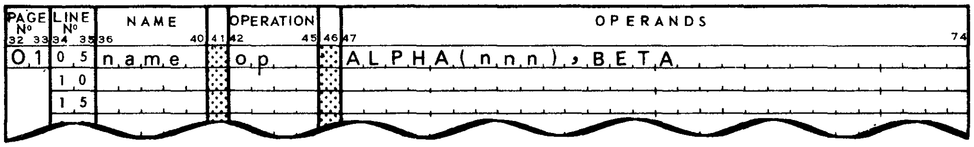
The operations described in this section treat data fields under one of the three following specifications:

Full octets in both fields.
Right quartets in both fields.
Full octets in one field, right quartets in the other.

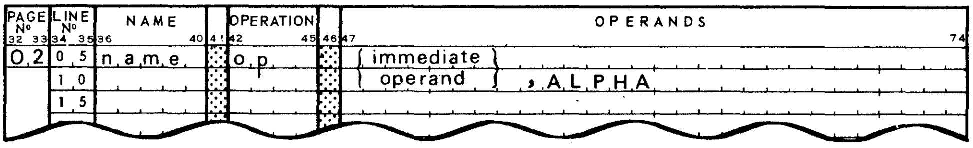
Operations which process octets have the following general characteristics:

- Data fields may be from 1 to 256 octets in length. A single length is used; this is the length of the first data field.
- Operation length is governed by the length of the first data field referenced.
- Data fields are referenced at the left; operation is left to right. There is a single exception to this rule - the SL (Search to the Left).

The **formats** of the octet data movement and comparison instructions are :



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05	name		op		ALPHA(nnn),BETA
	10					
	15					

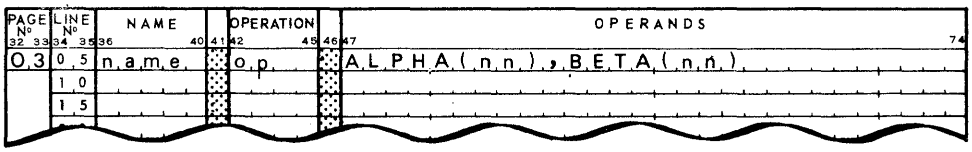


PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05	name		op		{ immediate }
	10					operand } ,ALPHA
	15					

Operations which treat only right quartets have the following general characteristics :

- Data fields may be from 1 to 16 octets in length. The length of each data field is used.
- The length of the operation is governed by the length of the first data field referenced.
- Data fields are referenced at the right ; operation is right to left.

The format of the right quartet data movement and comparison instructions is :

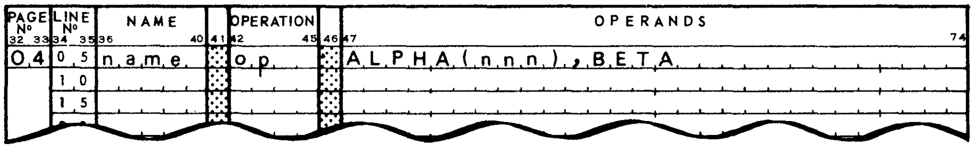


PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
03	05	name		op		ALPHA(nn),BETA(nn)
	10					
	15					

Two special purpose data movement operations treat the data as octets in one field and as right quartets in the other. These operations condense or expand data in the store.
These operations have the following general characteristics :

- A single length is used. This is the length of the field treated as full octets.
- The length of the operation is governed by the length of the data field which is treated in octet units.
- Data fields are referenced at the left.
- The first data field is replaced by the result.

The format of the octet/quartet data movement and comparison instructions is :



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
04	05	name		op		ALPHA(nnn),BETA
	10					
	15					

MOVE IMMEDIATE OCTET TO STORE MVI

MVI $\begin{matrix}\text{immediate}\\\text{operand}\end{matrix}$,ALPHA

The immediate data item in the MVI instruction is placed in the store. A single octet ALPHA field is replaced by the data item.

INDICATORS AFFECTED

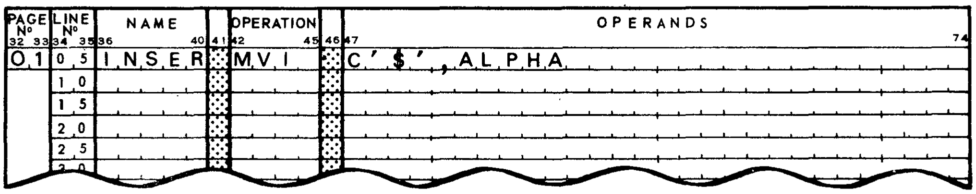
none

PROGRAMMING PRACTICE:

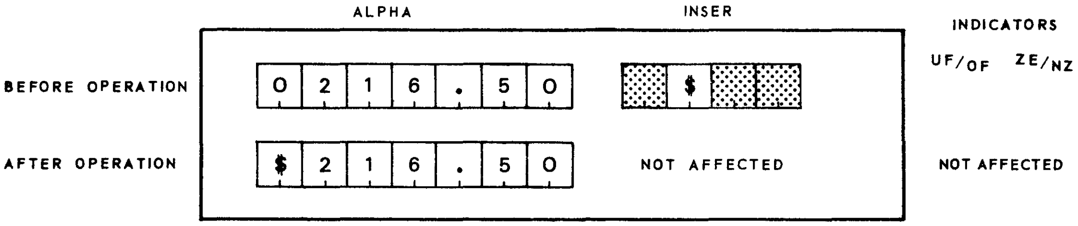
The MVI may be used in conjunction with the Move Complete Octets (MVC) instruction for a character fill. To accomplish this, the programmer writes an MVI to insert the fill character into the leftmost octet of the field he wishes to fill. He then follows the MVI with an MVC which treats the field as a pair of overlapping fields. (See the MVC,page 69).

EXAMPLES:

1) Movement of a graphic character to store.



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	47 OPERANDS 74
01	05	INSER		MVI		C'$',ALPHA
	10					
	15					
	20					
	25					



2) A hexadecimal value moved to an instruction complement.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05	MOVIM		MVI		X'00',TEST1+001
	10					
	15					
	20	TEST1		JC		X'20',SIGMA
	25					
	30					

TEST 1 is an instruction and has a length of 4 octets.

	TEST 1	MOVIM	INDICATORS UF/OF ZE/NZ
BEFORE OPERATION	43 20 23 34	▨ 00 ▨ ▨	
AFTER OPERATION	43 00 23 34	NOT AFFECTED	NOT AFFECTED

MOVE COMPLETE OCTETS MVC

MVC ALPHA (nnn),BETA

Full octets from the BETA field are placed in the ALPHA field. Movement is left to right through the common length of the fields (001-256 octets).

INDICATORS AFFECTED

none

NOTES :

- Operation is serial, octet by octet, from left to right through the ALPHA and BETA fields.
- The BETA field replaces the ALPHA field.
- BETA field octets are unaffected unless some part of the ALPHA field lies in the BETA field.

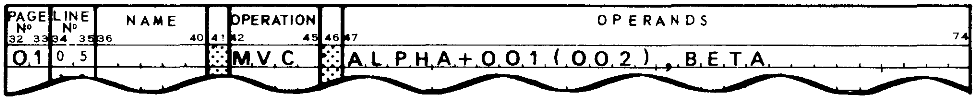
PROGRAMMING PRACTICES :

The MVC may be used to assemble a data field for output.

A field may be filled with a single character configuration by the use of the MVC. To accomplish this the programmer defines the ALPHA and BETA fields as overlapping. The ALPHA field begins in the octet to the immediate right of the first BETA octet.

EXAMPLES :

1) Data movement from one field to another.



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			MVC		ALPHA+001(002),BETA

ALPHA has a defined length of 3 octets ; 2 octets are replaced.

	ALPHA	BETA	INDICATORS UF/OF ZE/NZ
BEFORE OPERATION	$ | 5 | 0	2 | 5 | 0 | 0	
AFTER OPERATION	$ | 2 | 5	NOT AFFECTED	NOT AFFECTED

2) The use of the MVC instruction for a numeric character fill ; in this case, setting a field to decimal zeros.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			MVI		X'40',ALPHA PRESET ZERO
	10			MVC		ALPHA+001(005),ALPHA

ALPHA is a 6 octet field.

	ALPHA	IMMEDIATE OCTET	INDICATORS UF/OF ZE/NZ
BEFORE OPERATION	59 | 51 | 42 | 5A | FF | 45	40	
AFTER MVI	40 | 51 | 42 | 5A | FF | 45	NOT AFFECTED	NOT AFFECTED
AFTER MVC	40 | 40 | 40 | 40 | 40 | 40		NOT AFFECTED

MOVE RIGHT QUARTETS MVQ

MVQ ALPHA (nn), BETA

Right quartets from the BETA field are placed in right quartets of the ALPHA field. Movement is right to left through the length of the ALPHA field (01-16 octets).

INDICATORS AFFECTED

UF/OF	ZE/NZ	
0	0	The ALPHA result field contains zero in all right quartets.
0	1	At least one right quartet in the ALPHA field is non-zero.

NOTES:

- Operation is serial, quartet by quartet, from right to left through the ALPHA and BETA fields.
- Only the right quartets of the fields are processed. Left quartets in both fields are unaffected.
- The operation is terminated when the ALPHA field has been processed.

  If the fields are of equal length, all right quartets of the ALPHA field are replaced by all right quartets from the BETA field.

  If the length of the BETA field is greater than the length of the ALPHA field, the right quartets of the ALPHA field are all replaced. Excess right quartets in the left of the BETA field are not moved.

  If the length of the BETA field is less than the length of the ALPHA field, the excess right quartets in the left of the ALPHA field are replaced with zeros.

- The BETA field right quartets are unaffected unless some part of the ALPHA field lies in the BETA field.
- The UF/OF indicator is set to zero prior to the MVQ operation and is unaffected by the operation.
- The ZE/NZ indicator records the presence of an all-zero or a non-zero result in the ALPHA field.

PROGRAMMING PRACTICES:

The MVQ may be used to place zeros in the right quartets of a field used for decimal operations. To accomplish this the programmer may use a single octet BETA field with a right quartet of zero. The length of the ALPHA field governs the length of the operation. The single quartet is moved from the BETA field to the ALPHA field and the remaining ALPHA field right quartets are zero filled.

If the MVQ follows an instruction which records a result in the UF/OF indicator, the setting should be tested or saved prior to the MVQ. (See the Jump on Condition, JC, instruction on page 97 for a method of saving indicator settings for subsequent test).

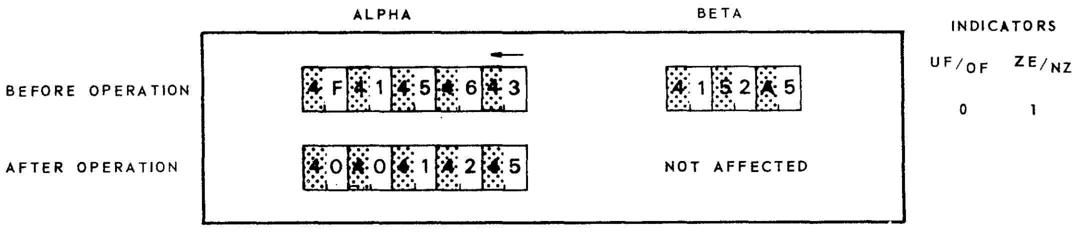
A field may be checked for zero when it is moved by means of the MVQ. To accomplish this the programmer tests the ZE/NZ indicator with a Jump on Condition instruction.

EXAMPLES:

1) Use of the MVQ to move data to a field with the desired left quartet configurations.

PAGE No 32 33	LINE No 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			MVQ		ALPHA, BETA

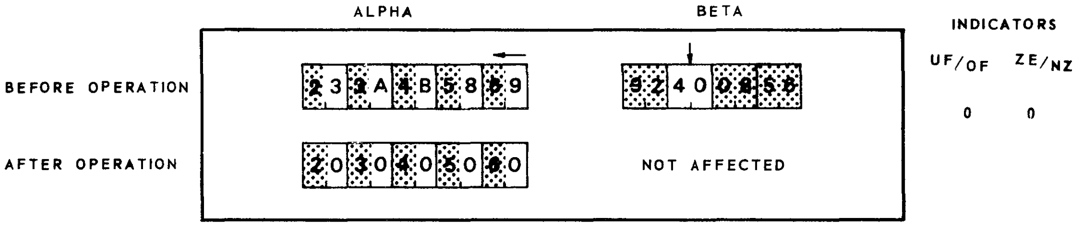
ALPHA has a defined length of 5 octets.
BETA has a defined length of 3 octets.
ALPHA field right quartets not filled from the BETA field are filled with right quartets zeros.



2) A field used for decimal operations may be reset to zero between operations by means of the MVQ to transmit right quartet zeros. To accomplish this the programmer uses a single octet BETA field. The right quartet of the BETA field octet must be zero.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			MVQ		ALPHA, BETA-002(01)

ALPHA has a defined length of 5 octets.



PACK RIGHT QUARTETS INTO OCTETS **PK**

PK ALPHA(nnn), BETA

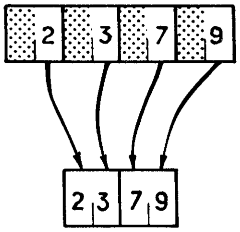
Right quartets of two BETA field octets are packed into a single octet in the ALPHA field. Packing is left to right in both fields through the length of the ALPHA field (001-256 octets).

INDICATORS AFFECTED

none

NOTES:

- Operation is serial, from left to right, through the ALPHA field.
- Two right quartets of the BETA field are packed into each octet in the ALPHA field as shown below.



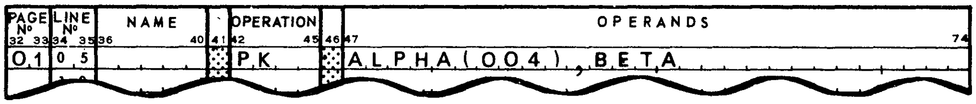
- The result replaces the octets of the ALPHA field.
- The left quartets of the BETA field are not moved.
- The BETA field is not affected unless some part of the ALPHA field lies in the BETA field.

PROGRAMMING PRACTICE:

The PK operation may be used to condense data in the store after input and prior to output or arithmetic use. This enables the programmer to economize the use of the store by halving the length of the field that is retained. The Unpack (UPK) instruction is used to recreate the field for its intended use.

EXAMPLES :

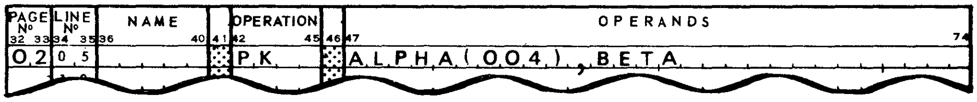
1) Decimal data packed for retention in store.



	ALPHA	BETA	INDICATORS UF/OF ZE/NZ
BEFORE OPERATION	41 44 C3 A2	41 49 41 42 43 44 48 42	
AFTER OPERATION	19 12 34 82	NOT AFFECTED	NOT AFFECTED

2) The use of the PK instruction to condense an input record.

If the left quartet values are known, it is possible to pack non decimal data and reconstruct it later. In the following example, it is assumed that the programmer knows that the first four BETA characters are letters between A and I and that the last four are decimal digits.



	ALPHA	BETA	INDICATORS UF/OF ZE/NZ
BEFORE OPERATION	50 50 50 50	A F E I 6 3 2 8 51 56 55 59 46 43 42 48 Known to be alphabetic Known to be numeric	
AFTER OPERATION	16 59 63 28	NOT AFFECTED	NOT AFFECTED

UNPACK OCTETS INTO RIGHT QUARTETS UPK

UPK ALPHA (nnn), BETA

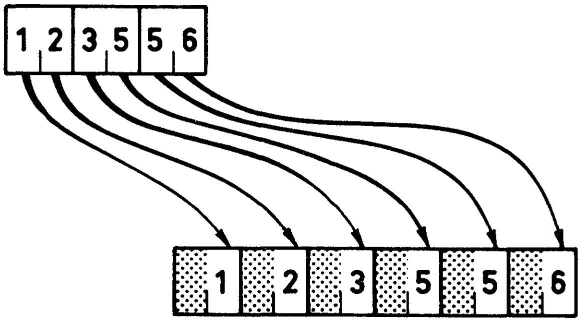
Octets from the BETA field are unpacked into the right quartets of two octets in the ALPHA field. Unpacking is left to right in both fields, through the length of the BETA field (001-256 octets).

INDICATORS AFFECTED

none

NOTES :

- Operation is serial, from left to right, through the BETA field.
- Each octet of the BETA field is unpacked into two right quartets in the ALPHA field as shown below.



- The operation is terminated when the BETA field has been unpacked.
- The right quartets of the ALPHA field are replaced by the result.
- The left quartets of the ALPHA field do not enter into the operation unless some part of the ALPHA field lies in the BETA field.
- The BETA field is not affected unless some part of the ALPHA field lies in the BETA field.

PROGRAMMING PRACTICE :

The UPK is used to recreate a field which has been condensed for retention in store. If the data is unpacked for output, the unpacking should be done into a field which has been preset with the required left quartet configuration. When data is unpacked into a work area for use with operations which do not utilize the left quartet no presetting is necessary.

Note : The operation is governed by the length of the BETA field, but the length is written with the ALPHA field operand specification.

EXAMPLE :

Data unpacked from a save area into a field preset for decimal output.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			UPK		ALPHA(003),BETA

	ALPHA	BETA	INDICATORS UF/OF ZE/NZ
BEFORE OPERATION	40 4B 4C 45 42 41	72 34 89	
AFTER OPERATION	47 42 43 44 48 49	NOT AFFECTED	NOT AFFECTED

COMPARE IMMEDIATE TO STORE CMI

$$\text{CMI} \quad \begin{matrix}\text{immediate}\\ \text{operand}\end{matrix}\text{, ALPHA}$$

The immediate data item in the CMI instruction is compared to a single octet ALPHA field.

INDICATORS AFFECTED

UF/OF	ZE/NZ	Comparison
0	1	ALPHA < immediate data item
1	0	ALPHA = immediate data item
1	1	ALPHA > immediate data item

NOTES :

- Neither the ALPHA field octet nor the immediate data item is affected by the comparison operation.
- The UF/OF and ZE/NZ indicators record the results of the comparison.

PROGRAMMING PRACTICES :

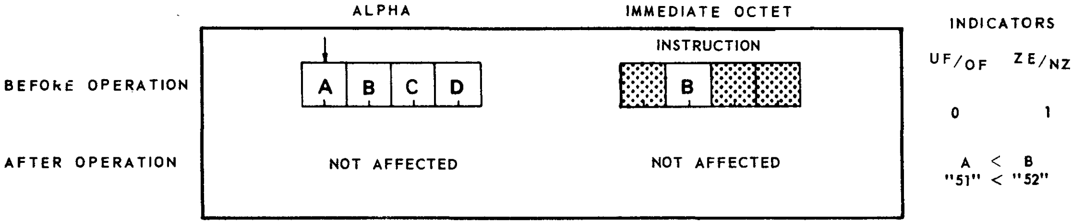
The CMI may be used to verify the configuration of an octet in the store.

A Jump on Condition instruction is used to test the result of the comparison.

EXAMPLES:

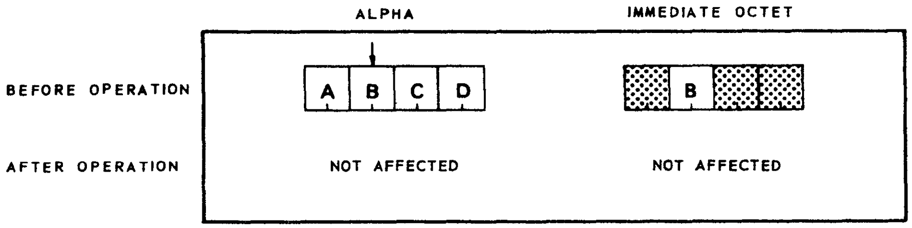
1) Comparison of an octet in ALPHA with an immediate octet, ALPHA less than the immediate octet.

PAGE N° 32 33	LINE N° 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			CMI		C'B',ALPHA



2) Comparison of an octet in ALPHA with an immediate octet, octets equal.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			CMI		C'B',ALPHA+001



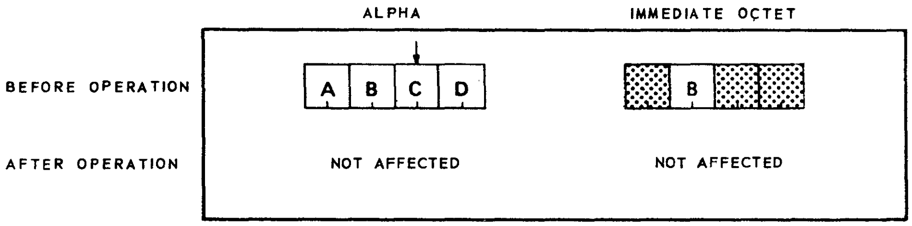
INDICATORS

UF/OF	ZE/NZ
1	0

B = B

3) Comparison of an octet in ALPHA with an immediate octet, ALPHA greater than the immediate octet.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
03	05			CMI		C'B',ALPHA+002



INDICATORS

UF/OF	ZE/NZ
1	1

C > B

COMPARE COMPLETE OCTETS CMC

CMC ALPHA (nnn), BETA

Full octets of the ALPHA field are compared to full octets of the BETA field. Comparison is from left to right, through the common length of the fields (001-256 octets).

INDICATORS AFFECTED

UF/OF	ZE/NZ	Comparison
0	1	ALPHA < BETA
1	0	ALPHA = BETA
1	1	ALPHA > BETA

NOTES

- Operation is serial, octet by octet, from left to right through the ALPHA and BETA fields.
- The operation is terminated by the recognition of the first inequality. If the contents of the fields are equal, the comparison continues through the common length of the ALPHA and BETA fields.
- The ALPHA and BETA fields are unaffected by the comparison operation.
- The UF/OF and ZE/NZ indicators record the result of the comparison.

PROGRAMMING PRATICES :

The CMC may be used as an alternate to the CMQ for comparing fields of equal length in which the left quartets are known to be equal and the right quartets determine the difference. Unless the contents of the compared fields are identical, processing is confined to fewer octets when the C MC is used rather than the CMQ thus giving a faster operation.

A Jump on Condition instruction is used to test the result of the comparison.

EXAMPLES :

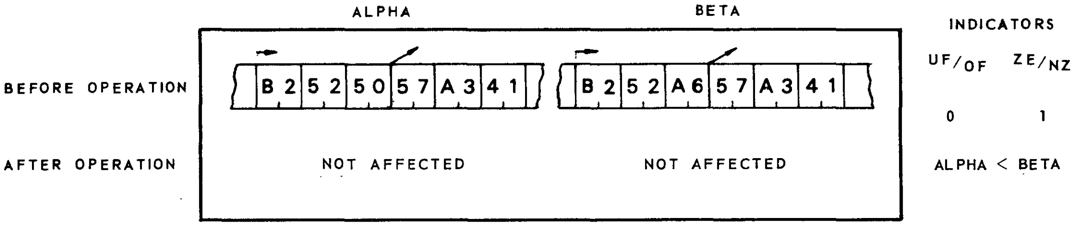
1) Comparison of two fields, the first less than the second.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			CMC		ALPHA,BETA

ALPHA has a defined length of 6 octets.

BETA has a defined length of 2 octets, but the defined length of BETA is not used in the operation.

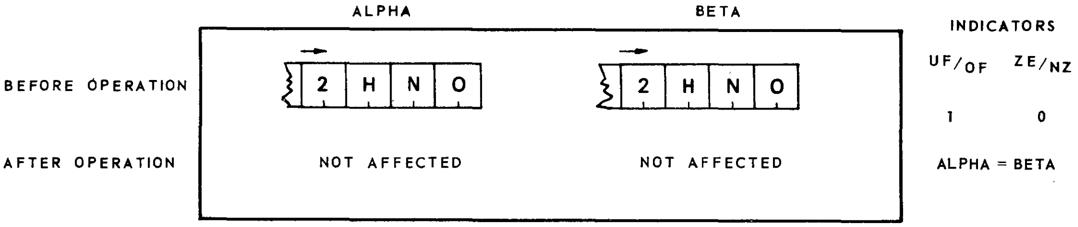
Operation terminates when the first inequality is encountered.



2) Comparison of two equal fields.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			CMC		ALPHA,BETA

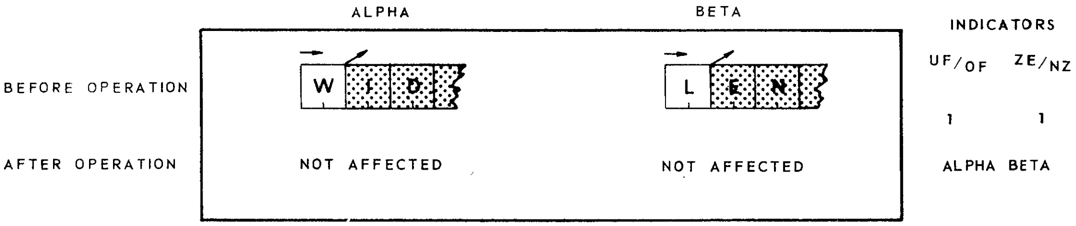
ALPHA has a defined length of 4 octets.



3) Comparison of two fields, the first greater than the second.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
03	05			CMC		ALPHA, BETA

ALPHA has a defined length of 32 octets, but operation is terminated when the first inequality is encountered.
The defined length of BETA is not used in the operation.



COMPARE RIGHT QUARTETS CMQ

CMQ ALPHA (nn), BETA (nn)

Right quartets of the ALPHA field are compared to right quartets of the BETA field. Comparison is right to left, through the length of the ALPHA field (01 - 16 octets).

INDICATORS AFFECTED

UF/OF	ZE/NZ	Compared Fields
0	1	ALPHA < BETA
1	0	ALPHA = BETA
1	1	ALPHA > BETA

NOTES:

- Operation is serial, quartet by quartet, from right to left.
- Only the right quartets of the fields are processed.
  If the fields are of equal length, all quartets in both fields are compared.
  If the length of the BETA field is greater than the length of the ALPHA field, the excess quartets in the left of the BETA field do not enter the comparison.
  If the length of the BETA field is less than the length of the ALPHA field, the excess quartets in the left of the ALPHA field are compared to zero quartets.
- The ALPHA and BETA fields are unaffected by the comparison operation.
- The UF/OF and ZE/NZ indicators record the result of the comparison.

PROGRAMMING PRACTICES:

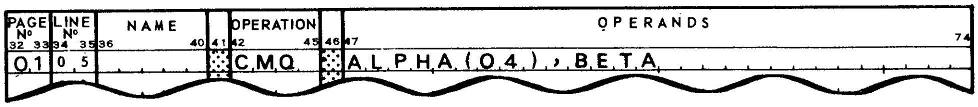
The CMQ may be used to compare fields in which only the right quartets are meaningful at the time of comparison. For example, data unpacked into a work area in which left quartet values are not the same, can be compared with the CMQ operation. The programmer should be certain that no mistakes are caused by treating the data as right quartets only.

A Jump on Condition (JC) instruction is used to test the result of the comparison.

Note : The CMC operation (see page 80) may be a more efficient operation for comparison than the CMQ if the left quartets of the fields to be compared are the same.

EXAMPLES :

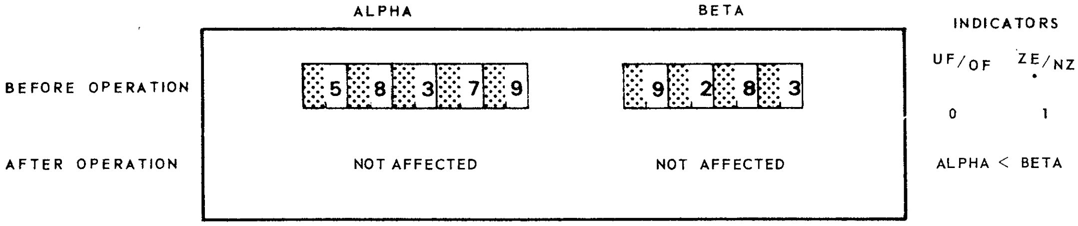
1) Comparison of right quartets in two data fields



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
0 1	0 5			CMQ		ALPHA(04),BETA

ALPHA has a defined length of 5 octets, but 4 are specified.

BETA has a defined length of 4 octets.

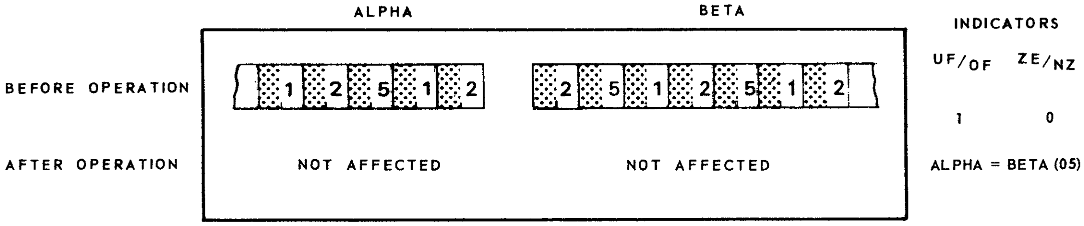


2) Comparison of right quartets in two data fields

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
0 2	0 5			CMQ		ALPHA,BETA

ALPHA has a defined length of 5 octets.

BETA has a defined length of 7 octets, but only 5 enter the operation, because the operative length is the length of the ALPHA field.



SEARCH TO THE RIGHT SR

SR ALPHA (nnn), BETA

The ALPHA field is searched for an octet equal to the single BETA field octet. Search is from left to right through the ALPHA field (001 - 256 octets).

INDICATORS AFFECTED

UF/OF	ZE/NZ	Search Result
1	0	Search Failed
1	1	Match Found

NOTES:

- Operation is serial, octet by octet, from left to right in the ALPHA field. The BETA field is a single octet.
- The operation is terminated when the BETA field data item has been found in the ALPHA field.

  If the BETA field item is not present in the ALPHA field, the operation is terminated when all the ALPHA field octets have been examined.
- When the search is terminated, LOC (store octets 0254-0255) contains the location of the octet to the immediate right of the last ALPHA octet examined.

  If the BETA field item was found in the ALPHA field, the store location in LOC is that of the octet to the right of the matched data item.

  If the BETA field item was not found in the ALPHA field, the store location in LOC is that of the octet to the right of the last ALPHA field octet examined, i.e., the location of the octet to the immediate right of the last octet in the ALPHA field.
- The UF/OF indicator is preset to 1 by the SR and is not affected by the operation.
- The ZE/NZ indicator records the result of the search.

PROGRAMMING PRACTICES:

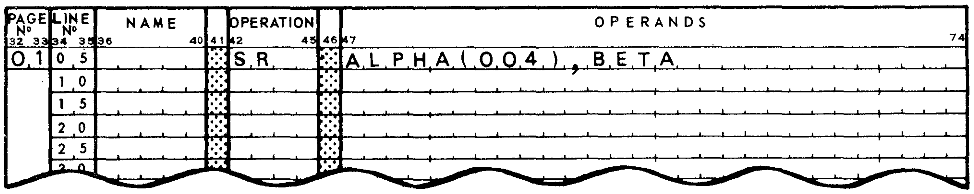
The ZE/NZ indicator must be interrogated by a Jump on Condition instruction to test the result of the search.

The address in LOC must be decremented by 1 to reference a matched item in the ALPHA field.

When the SR operation follows an operation which sets the UF/OF indicator and the setting is used by the program, the programmer should use or save the UF/OF setting prior to the SR (See the Jump on Condition on page 97 for a method of saving indicator settings for a subsequent test).

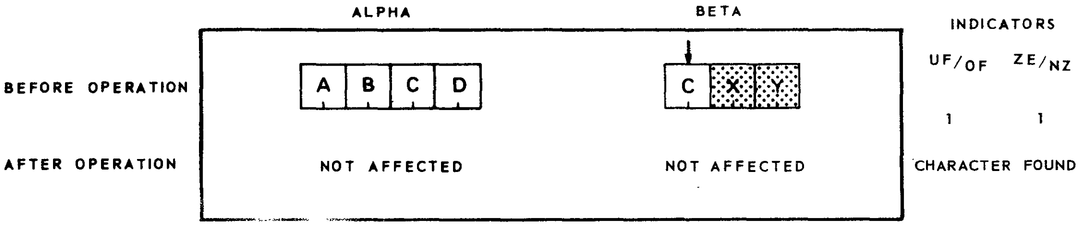
EXAMPLES:

1) Search, character found.



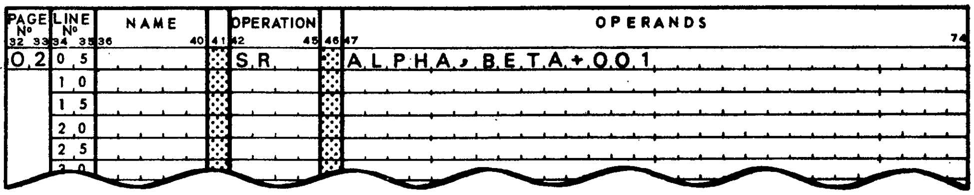
PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			SR		ALPHA(004),BETA
	10					
	15					
	20					
	25					

The character sought is the letter C.



After the operation LOC contains the address of the ALPHA octet to the immediate right of the matching character (location of D).

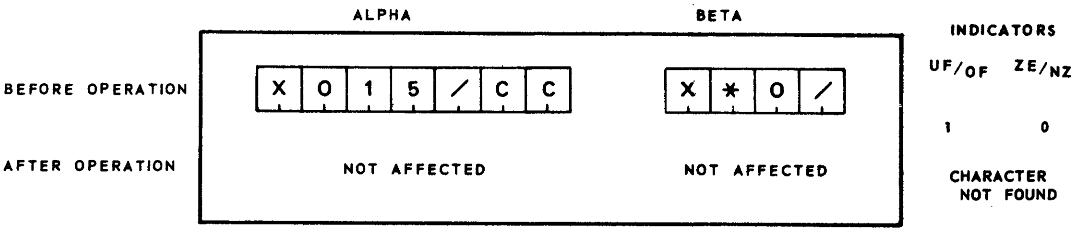
2) Search, character not found



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			SR		ALPHA,BETA+001
	10					
	15					
	20					
	25					
	30					

ALPHA has a defined length of 7 octets

The character sought is an asterisk (∗)



	ALPHA	BETA	INDICATORS UF/OF	INDICATORS ZE/NZ
BEFORE OPERATION	X 0 1 5 / C C	X ∗ 0 /		
AFTER OPERATION	NOT AFFECTED	NOT AFFECTED	1	0

After the operation LOC contains the address of the octet to the right of the rightmost octet in ALPHA

SEARCH TO THE LEFT SL

SL ALPHA (nnn), BETA

The ALPHA field is searched for an octet equal to the single BETA field octet. Search is from right to left through the ALPHA field (001 - 256 octets).

INDICATORS AFFECTED

UF/OF	ZE/NZ	Search Result
1	0	Search Failed
1	1	Match Found

NOTES:

- Operation is serial, octet by octet, from right to left in the ALPHA field. The BETA field is a single octet.
- The operation is terminated when the BETA field data item has been found in the ALPHA field.

  If the BETA field item is not present in the ALPHA field, the operation is terminated when all the ALPHA field octets have been examined.
- When the search is terminated, LOC (store octets 0254-0255) contains the location of the octet to the immediate left of the last ALPHA octet examined.

  If the BETA field item was found in the ALPHA field, the store location in LOC is that of the octet to the left of the matched data item.

  If the BETA field item was not found in the ALPHA field, the store location in LOC is that the octet to the left of the last ALPHA field data item examined.
- The UF/OF indicator is preset to 1 by the SL and is not affected by the operation.
- The ZE/NZ indicator records the result of the search.

PROGRAMMING PRACTICES :

The ZE/NZ indicator must be interrogated by a Jump on Condition instruction to test the result of the search.

The address in LOC must be incremented by 1 to reference a matched item in the ALPHA field.

When the SL operation follows an operation which sets the UF/OF indicator and the setting is used by the program, the programmer should use or save the UF/OF setting prior to the SL (See the Jump on Condition instruction on page 97 for a method of saving indicator settings for a subsequent test).

NOTE: The SL is the only one of the complete octet comparison instructions which operates from right to left.

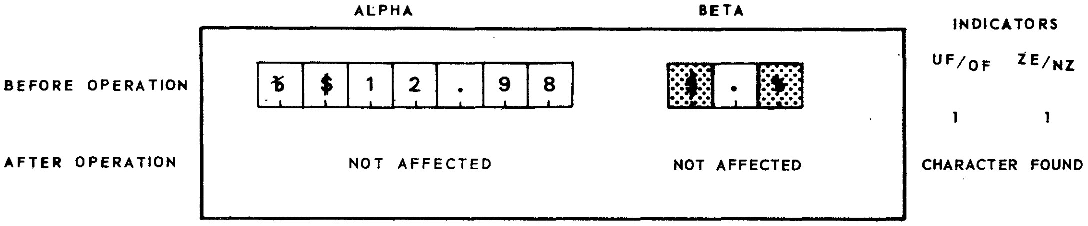
EXAMPLES:

1) Search, character found.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			SL		ALPHA(005),BETA+001

ALPHA has a defined length of 7 octets.

The character sought is a period (.).



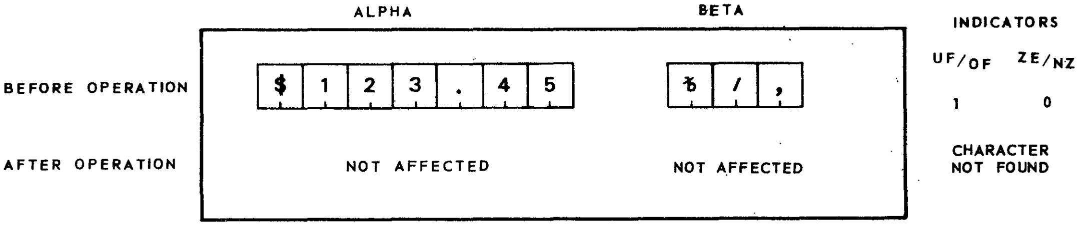
After the operation LOC contains the address of the octet to the immediate left of the matching character (location of 2).

2) Search, character not found.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			SL		ALPHA,BETA

ALPHA has a defined length of 7 octets.

The character sought is a blank.



After the operation LOC contains the address of the octet to the immediate left of the leftmost octet of the ALPHA field.

LOGIC INSTRUCTIONS

The GE-115 Information Processing System generates logical sums and products of data fields. Data treated by the logic operations is used by the system as bit patterns rather than numeric quantities or symbolic representations.

The logic operations have the following general characteristics:

- Data from two fields is matched and combined.
- Data fields may be from 1 to 256 octets in length. The length of the first data field is used.
- Operation length is governed by the length of the first data field reference.
- Data fields are referenced at the left; operation is left to right.
- The first data field is replaced by the result.
- Complete octets are processed.

The format of the logic instructions is:

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05	name		op		ALPHA(nnn),BETA
	10					
	15					
	20					
	25					
	30					

'AND' ON COMPLETE OCTETS NC

NC ALPHA (nnn), BETA

Octets in the BETA field are examined, bit by bit. Each zero bit in the BETA field is effectively transmitted to the corresponding bit position in the ALPHA field. Transmission is left to right through the common length of the fields (001 - 256 octets).

INDICATORS AFFECTED

none

NOTES :

- Operation is serial, octet by octet, from left to right, through the ALPHA and BETA fields.
- One bits in the BETA field do not affect the ALPHA field.
- The ALPHA field is replaced by the result.
- The BETA field is unaffected unless some part of the ALPHA field lies in the BETA field.

EXAMPLE :

The NC instruction used to zero the three bits in the left of an octet.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			NC		ALPHA(001), BETA

	ALPHA	BETA	INDICATORS UF/OF ZE/NZ
BEFORE OPERATION	1011 1101	0001 1111	
AFTER OPERATION	0001 1101	NOT AFFECTED	NOT AFFECTED

'OR' ON COMPLETE OCTETS **OC**

OC ALPHA(nnn),BETA

Octets in the BETA field are examined, bit by bit. Each one bit in the BETA field is effectively transmitted to the corresponding bit position in the ALPHA field. Transmission is left to right through the common length of the fields (001 - 256 octets).

INDICATORS AFFECTED

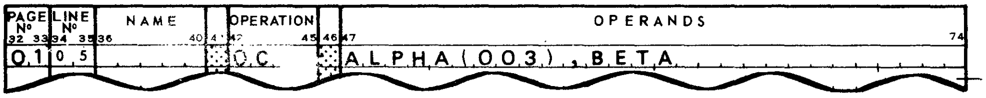
none

NOTES :

- Operation is serial, octet by octet, from left to right, through the ALPHA and BETA fields.
- Zero bits in the BETA field do not affect the ALPHA field.
- The ALPHA field is replaced by the result.
- The BETA field is unaffected unless some part of the ALPHA field lies in the BETA field.

EXAMPLE :

A logical 'or' of two three octet data field.



PAGE N° 32 33	LINE N° 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			OC		ALPHA(003),BETA

	ALPHA	BETA	INDICATORS UF/OF ZE/NZ
BEFORE OPERATION	0100 0100 0101 0110 0101 0000	0101 0011 1010 1010 0100 0001	
AFTER OPERATION	0101 0111 1111 1110 0101 0001	IS NOT AFFECTED	NOT AFFECTED

EXCLUSIVE 'OR' ON COMPLETE OCTETS XC

XC ALPHA(nnn),BETA

Octets in the BETA field are examined, bit by bit. Each one bit in the BETA field inverts the corresponding bit in the ALPHA field. Operation is left to right through the common length of the fields (001 - 256 octets).

INDICATORS AFFECTED

UF/OF	ZE/NZ	
1	0	The resultant ALPHA field is all zero.
1	1	At least one bit in the resultant ALPHA field is non-zero.

NOTES :

- Operation is serial, octet by octet, from left to right, through the ALPHA and BETA fields.
- Zero bits in the BETA field do not affect the ALPHA field.
- The ALPHA field is replaced by the result field.
- The BETA field is unaffected unless some part of the ALPHA field lies in the BETA field.
- The UF/OF indicator is set to 1 and is not affected by the operation; the ZE/NZ indicator records the value of the ALPHA result.

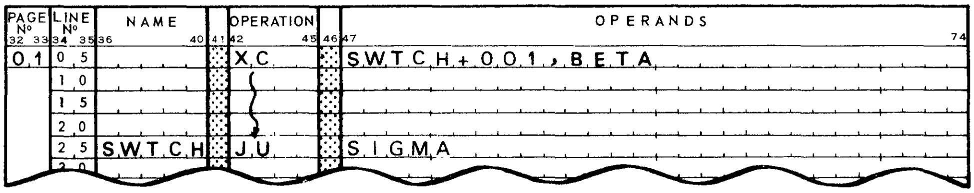
PROGRAMMING PRACTICES :

When the XC instruction follows an operation which records results in the indicators, care must be taken to preserve or use the information provided by the indicator settings, if it is required for program operation.

The XC instruction may be used to alter the mode of operation of the Jump on Condition (JC) operations. These instructions use the operation complement of the internal instruction format to differentiate conditions to be tested as directives for operation. A single BETA field octet can be set up containing a pattern which alters one of the test patterns and changes the JC action.

EXAMPLE :

An XC instruction used to alter an operation complement.



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			XC		SWTCH+001,BETA
	10					
	15					
	20					
	25	SWTCH		JU		SIGMA
	30					

	OPERATION COMPLEMENT (SWITCH + 001)	BETA
BEFORE OPERATION	1111 0000	1111 0000
AFTER FIRST XC	0000 0000	NOT AFFECTED
AFTER SECOND XC	1111 0000	NOT AFFECTED

INDICATORS

UF/OF	ZE/NZ
1	0

RESULT ZERO

UF/OF	ZE/NZ
1	1

RESULT NON ZERO

JUMP INSTRUCTIONS

The GE-115 Information Processing System acts upon instructions in the sequence of their locations in the store. The system may be directed by the jump instructions to alter that sequence.

There are two types of jump instructions. One type interrogates the condition indicators and directs the system to interrupt sequential operation when a test condition is present. The second type does not use the test indicators.

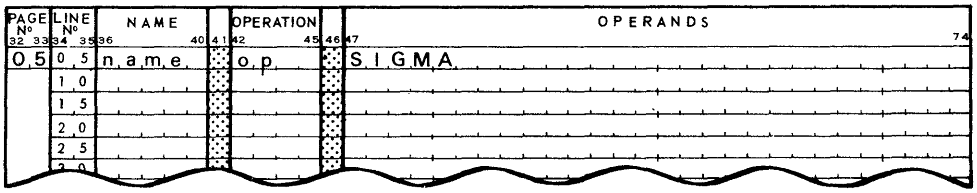
The jump instructions which test the indicators are called conditional jumps. They have the following general characteristics:

- The immediate operand in the internal instruction (second octet of the operation in the store) specifies a condition pattern for testing the indicators.
- The second operand in the internal instruction refers to an instruction in the store to which control is given if the test condition is met.

Operation continues in sequence when the condition tested is not present.

The conditional jump instructions are divided into two groups, on the basis of instruction specification. The first group has an implied first operand. The assembler translates the mnemonic for the operation into an operation code and the required operation complement to perform the test specified in that mnemonic. The second group requires an explicit first operand specification to set up the pattern which tests a condition.

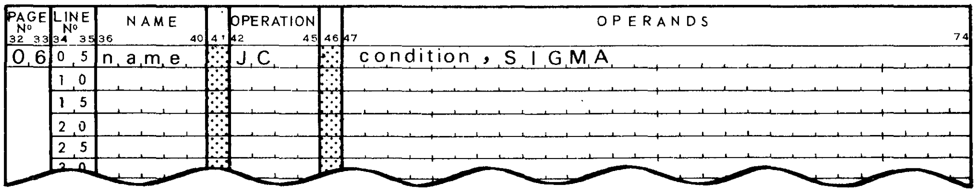
The conditional jump instructions which do not require an explicit immediate operand specification have the following format:



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
05	05	name		op		SIGMA
	10					
	15					
	20					
	25					

Figure B-3 lists the conditional jump instructions, showing conditions tested and mnemonic expressions used.

The conditional jump instructions which require an explicit immediate operand have the following format:



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
06	05	name		JC		condition , SIGMA
	10					
	15					
	20					
	25					

Figure B-2 shows the configurations of the immediate operands required to interrogate the testable conditions. It is recommended that the hexadecimal notation be used in specifying immediate operands.

There are three jump instructions which do not test the indicators. Two of these test an external condition, a switch setting. The jump is taken when the tested switch is on.

The third is a special purpose jump instruction which always alters the program sequence. In addition, this operation, the Jump and Return (JRT) places the address of the next sequential operation into LOC (store octets 0254-0255). This provides a means of returning to the operation which follows the jump instruction.

The jump instructions which do not test the indicators have the same format as the conditional jump instructions for which the test pattern is implied by the mnemonic expression.

JUMP ON CONDITION

JC

JC condition, SIGMA

The condition specified in the operation is tested. If the condition is present, the program jumps to the instruction at the SIGMA location.

INDICATORS AFFECTED

none

NOTES:

- Conditional jump instructions test the status of the UF/OF and ZE/NZ indicators which record the results of internal operations and peripheral operations.
- There are four possible patterns which may be present in the indicators:

	UF/OF	ZE/NZ
pattern 1	0	0
pattern 2	0	1
pattern 3	1	0
pattern 4	1	1

- The conditional jump operation may test for any of these patterns singly, or, it may test for combinations of these patterns.
- Each of the four bits in the left quartet of the operation complement in the internal format of the JC instruction corresponds to a pattern. If a bit is a 1, the pattern to which the bit corresponds is tested; if the bit is a 0, the pattern is not tested.
- The bits in positions 4-7 of the operation complement and the pattern to which each corresponds are shown in the figure below. Note that the right quartet (bit positions 0-3) is always zero.

Bit position	7 6 5 4 3 2 1 0
pattern 1	1 0 0 0 0 0 0 0
pattern 2	0 1 0 0 0 0 0 0
pattern 3	0 0 1 0 0 0 0 0
pattern 4	0 0 0 1 0 0 0 0

Operation Complement

In the last figure only one pattern is specified for each operation complement shown. When two bits of the left quartets are one, two condition patterns are tested. Any combination of the patterns may be tested. If one of the specified patterns is present in the indicators the condition is met and the jump is taken.

- The jump instructions do not alter the indicators they test.

PROGRAMMING PRACTICES:

Conditional jumps provide the only means of testing the indicators set during program execution. Jumps should be placed immediately after the operations that set the indicators for testing, or the condition should be saved for subsequent testing.

It is recommended that the condition patterns be specified in the hexadecimal notation. Figure B-2 lists the patterns for the tests and the hexadecimal configuration of each. Note that the hexadecimal number 00 specifies that none of the four possible patterns be tested and the hexadecimal number F0 specifies that all patterns be tested. A Jump on Condition instruction in which the hexadecimal pattern is 00 is the No Jump (See page 103) and the Jump on Condition in which the condition specified is F0 is the Jump Unconditional (See page 102).

EXAMPLES:

1) The indicators cannot be accessed by the program. The settings recorded may be saved by a sequence such as that shown below. When the jump is taken to the instruction at SAVE the pattern which specified that jump is moved for a subsequent check.

```
PAGE LINE NAME    OPERATION  OPERANDS
64   05           *          JUMP IF OVERFLOW
     10   TEST    JC         X'30',SAVE
     15           MVI        X'C0',OFTST+001 NO OVERFLOW
     20           *          COMPARE RESETS THE OVERFLOW
     25   CONT    XC         ALPHA(003),BETA
     30

65   05
     10   SAVE    MVC        OFTST+001,TEST+001
     15
     20
     25
     30

70   05
     10   OFTST   CMI        X'00',IND
     15
     20
     25
     30
```

2) When the presence of a given condition is used to direct program operation subsequent to operations which alter the indicators, it is not necessary to retain the indicator pattern. When the condition is recognized a subsequent jump can be preset to act on the results of the test.

PAGE No 32 33	LINE No 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			MVI		X'F0',SWTCH+001
	10			JC		X'30',NO
	15			MVI		X'00',SWTCH+001
	20	NO		xxx		xxx

	85					
	90	SWTCH		JU		ARND
	95			xxx		xxx

OPERATION COMPLEMENT				INDICATORS	
HEX		BINARY		UF/OF	ZE/NZ
0	0	0000	0000	╳	╳
1		0001		1	1
2		0010		1	0
3		0011		1	either 0 or 1
4		0100		0	1
5		0101		either 0 or 1	1
6		0110		0 1	1 0
7		0111		0 1 1	1 0 1
8		1000		0	0
9		1001		0 1	0 1
A		1010		either 0 or 1	0
B		1011		0 1 1	0 0 1
C		1100		0	either 0 or 1
D		1101		0 0 1	0 1 1
E		1110		0 0 1	0 1 0
F	↓	1111	↓	all possibilities	

Figure B-2 : INDICATOR SETTINGS TESTED BY CONDITIONAL JUMPS

JUMP IF GREATER	JG
JUMP IF EQUAL	JE
JUMP IF GREATER OR EQUAL	JGE
JUMP IF LESS	JL
JUMP IF NOT EQUAL	JNE
JUMP IF LESS OR EQUAL	JLE

```
op    SIGMA
```

The condition specified in the operation is tested. If the condition is met, the program jumps to the operation at the SIGMA location.

INDICATORS AFFECTED

none

NOTE:

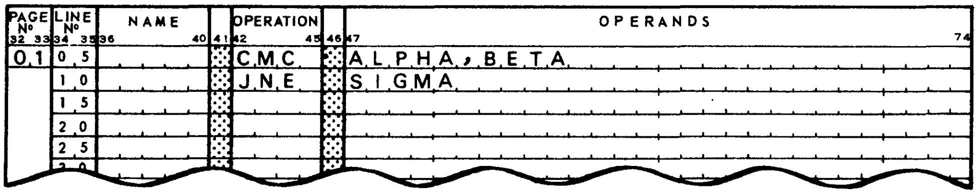
- The jump instructions do not alter the indicators they test.

PROGRAMMING PRACTICE:

The comparative conditional jump instructions are translated by the assembler into both the operation code and the operation complement which specifies the pattern for the condition or conditions to be tested. A test pattern may not be specified in the instruction statement.

EXAMPLE:

A Jump if Not Equal used to test the result of a comparison.



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			CMC		ALPHA, BETA
	10			JNE		SIGMA
	15					
	20					
	25					

If the ALPHA and BETA fields are not equal, control jumps to the operation at the SIGMA address. If the ALPHA and BETA fields are equal, the program continues in sequence.

JUMP UNCONDITIONAL JU

JU SIGMA

The program jumps to the operation located at the SIGMA field address.

INDICATORS AFFECTED

none

NOTES:

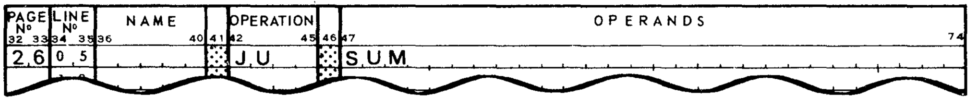
- The JU is a conditional jump which specifies all conditions.

PROGRAMMING PRACTICE:

The JU is used when a transfer of control is to be made that is independent of the status of the indicators.

EXAMPLE:

The program jumps to the sequence which begins at SUM.



PAGE No 32 33	LINE No 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
26	05			JU		SUM

NO JUMP NOJ

NOJ SIGMA

Control continues in sequence; no test pattern is specified.

INDICATORS AFFECTED

none

PROGRAMMING PRACTICE:

The NOJ may be changed to an effective jump by changing the configuration of the immediate operand. To accomplish this, the programmer may use a logical operation or an octet move. (See the XC on page 93 for a method of altering the test pattern in the jump instruction). The NOJ - JU instructions can function as alternating sequence controls.

EXAMPLE:

The program continues; the AD following the NOJ is the next instruction executed.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
21	05			NOJ		END
	10			AD		ALPHA,BETA

Figure B-3. TABLE OF CONDITIONAL JUMPS

Mnemonic	Operation Complement (hexadecimal)	Conditions indicated when a jump occurs
NOJ	00	No Jump occurs
JU	F0	Jump always occurs
		After a CMQ, CMC, or CMI, a jump occurs if:
JG	10	ALPHA > BETA; ALPHA > immediate operand
JE	20	ALPHA = BETA; ALPHA = immediate operand
JGE	30	ALPHA ⩾ BETA; ALPHA ⩾ immediate operand
JL	C0	ALPHA < BETA; ALPHA < immediate operand
JNE	D0	ALPHA ≠ BETA; ALPHA ≠ immediate operand
JLE	E0	ALPHA ⩽ BETA; ALPHA ⩽ immediate operand
JC	10	Condition present after peripheral status test
		End of operation after data transfer on channel 1
		Character found after SR or SL
JC	20	Condition not present after peripheral status test
		End of operation on length, after data transfer on channel 1
		ALPHA = 0; after SD, SB, or XC
JC	C0	Result in underflow form after SD or SB
JC	30	Overflow after AD or AB
JC	A0	ALPHA = 0; after AD or AB
JC	80	ALPHA = 0; after MVQ

JUMP IF SWITCH 1 SET JS1
JUMP IF SWITCH 2 SET JS2

```
JS1
      SIGMA
JS2
```

The status of the specified switch is interrogated. If the switch is set, program control jumps to the SIGMA field operation. If the switch is not set, the program continues in sequence.

INDICATORS AFFECTED

none

NOTE:

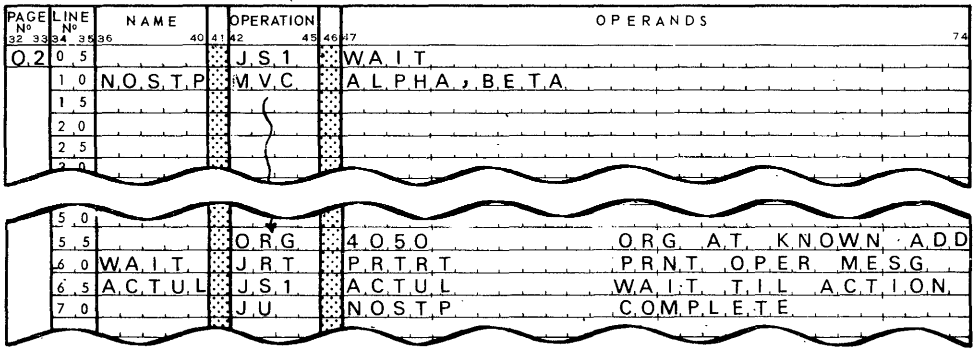
- The settings of the switches are not altered by the test. Switches are not under program control and must be set and reset externally.

PROGRAMMING PRACTICE:

Switch 1 and Switch 2 may be tested internally as a means of making certain that external operations have been carried out. An operator may be instructed to set a switch to indicate that an input file has been placed in the reader, or that cards have been set up for punching, or some other required action has been taken and the switch set to indicate the completion of the request. The program, after testing the switch by means of the jump, may reset the jump pattern to make the test ineffective. Messages should be included in the program for printing whenever operator intervention is required. Operator intervention should be restricted to the necessary minimum.

EXAMPLE:

The JS1 sets up an effective program halt. When Switch 1 is off the program goes on to execute the MVC.



PAGE No 32 33	LINE No 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			JS1		WAIT
	10	NOSTP		MVC		ALPHA,BETA
	15					
	20					
	25					
	30					
	50					
	55			ORG		4050 ORG AT KNOWN ADD
	60	WAIT		JRT		PRTRT PRNT OPER MESG
	65	ACTUL		JS1		ACTUL WAIT TIL ACTION
	70			JU		NOSTP COMPLETE

JUMP AND RETURN JRT

JRT SIGMA

The store location of the operation which follows the JRT is placed in LOC (store octets 0254-0255). Control is transferred to the operation which begins at the SIGMA field location.

INDICATORS AFFECTED

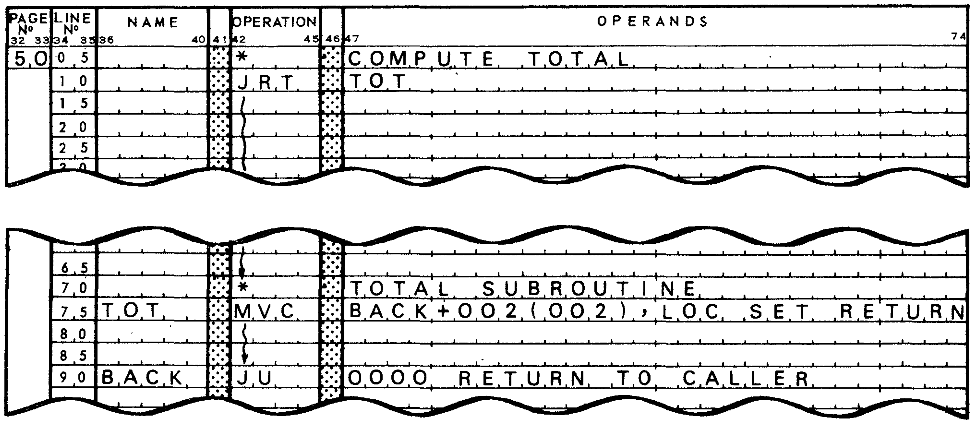
none

PROGRAMMING PRACTICE:

The JRT is used for subroutine entry. The contents of LOC must be moved to a jump instruction to effect a return to the sequence from which the subroutine was entered.

EXAMPLE:

The JRT used to jump to a subroutine named TOT. The first instruction in the TOT subroutine moves the return address from LOC to the jump instruction named BACK.



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
50	05			*		COMPUTE TOTAL
	10			JRT		TOT
	15					
	20					
	25					
	65					
	70			*		TOTAL SUBROUTINE
	75	TOT		MVC		BACK+002(002),LOC SET RETURN
	80					
	85					
	90	BACK		JU		0000 RETURN TO CALLER

EDIT INSTRUCTIONS

Data fields generated by programs in the GE-115 Information Processing System may be edited for output. Such operations as zero-suppression, character insertion, and field-spacing, may be performed. Editing simplifies the preparation of readable tabular listings, invoice sheets, and other printed reports.

Input data also may be edited. Editing of input data consists of preparing it for internal use by the GE-115 system. The varied internal codes recognized by other computers can be translated into the GE-115 internal code. Data and programs prepared by other systems can be translated for processing.

The editing instructions have the following general characteristics:

- A data field is operated upon by the use of a mask or table.
- Only the first data field length is used in the instruction.
- The length of the operation is governed by the length of the first data field.
- The operative length of the second data field is a function of the configuration of the first data field.
- Operation is from left to right in both fields.
- The first operand field is replaced by the result field.

The editing instructions have the following format:

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			op		ALPHA(nnn),BETA
	10					
	15					
	20					
	25					
	30					

EDIT EDT

EDT ALPHA(nnn),BETA

Octets from the ALPHA field are used as control characters to edit the information in the BETA field. Zero-suppression, character insertion, and spacing of fields are specified by the configuration of the ALPHA field. Editing proceeds from left to right. The length of the operation is determined by the length of the ALPHA field (001 to 256 octets). The ALPHA field contains the edited data at the end of the EDT operation.

INDICATORS AFFECTED

UF/OF	ZE/NZ	
1	0	Operation ended in the zero-suppression mode.
1	1	Operation ended in the non-zero-suppression mode.

NOTES:

- ALPHA is the control field for the EDT operation. The configuration of the ALPHA field determines the format of the edited field.
- The length of the BETA field used in the EDT operation is a function of the configuration of the ALPHA control field.
- The ALPHA field contains the edited data at the end of the EDT operation.
- Operation proceeds from left to right.
- Three types of edit control operations may be specified in the ALPHA control field. Each type of control operation is represented by a particular hexadecimal octet configuration in the ALPHA field. These are '20', '21' and '22'.
- The ALPHA field can contain, as well as the three types of hexadecimal control characters, any of the characters from the graphic set. The character in the leftmost octet of the ALPHA field serves as a "fill" character; that is, it may be used to replace any subsequent ALPHA octet that is not replaced from BETA.
- There are two modes of the editing operation: zero-suppression and non-zero-suppression.
- The action of the hexadecimal control characters in the ALPHA field is affected by the mode of operation at the time that they are encountered.
- The presence of the fill character and any other of the characters (non-control characters) in the edited field is determined by the mode of the operation when a particular octet is processed.

- Operation always begins in the zero-suppression mode. Non-zero-suppression 'begins' when a BETA octet is found to have a non-zero right quartet or when the ALPHA field octet contains the control configuration '21'.
- The first octet of the ALPHA field remains unchanged by the operation. The operation is in the zero-suppression mode and does not change. The character in the first octet of the ALPHA field will be used as the "fill" character in the remainder of the EDT operation.

- In the ZERO-SUPPRESSION MODE, the ALPHA octet causes the EDT operation to proceed in the following ways:

  '20'
  When the '20' is encountered in the ALPHA field, a check is made of the current BETA field octet to be edited.
  If the BETA field has a non-zero right quartet, the BETA field octet is placed in the ALPHA field. Zero-suppression is terminated.
  If the BETA field octet has a zero right quartet, the first character of the ALPHA field (the "fill" character) is placed in the ALPHA field octet. Zero-suppression continues.

  '21'
  The operative octet in the ALPHA field is replaced with the operative octet from the BETA field. Zero-suppression is terminated.

  '22'
  The operative octet in the ALPHA field is replaced by the first octet in the ALPHA field (the "fill" character). The BETA field is not involved. Zero-suppression continues.

  Any other character
  The operative octet in the ALPHA field is replaced by the first octet in the ALPHA field (the "fill" character). The BETA field is not involved. Zero-suppression continues.

- In the NON-ZERO-SUPPRESSION MODE, the ALPHA octet causes the EDT operation to proceed as follows:

  '20' and '21'
  The operative octet in the ALPHA field is replaced by the operative BETA field octet. The non-zero-suppression mode continues.

  '22'
  The operative octet in the ALPHA field is replaced by the "fill" character. The zero-suppression mode is restored.

  Any other character
  The operative octet in the ALPHA field is unchanged. BETA is not involved. The non-zero-suppression mode continues.

- The UF/OF indicator is set to 1 at the beginning of the EDT operation and is unaffected by the operation.
- A 0 in the ZE/NZ at the end of the operation indicates that the edit ended in the zero-suppression mode.
- A 1 in the ZE/NZ at the end of the operation indicates that the edit ended in the non-zero-suppression mode.

Figure B-4 shows the possible elements of the ALPHA field before and after the EDT operation, as determined by the mode of the operation and the contents of the BETA octet.

PROGRAMMING PRACTICES:

The first octet of the ALPHA field is used in the edited field as a fill character. When no BETA octet is transferred to the ALPHA octet, the fill character maintains the spacing. In general use, this character is the blank (X'50').

The ALPHA field is destroyed in the editing process; the BETA field (with editing) replaces the ALPHA field. The edit format which is in the ALPHA field must be preserved in another area of the store if it is to be used more than once in execution of the program. It is suggested that the edit format be defined in a DC (Define Constant) and moved to a work area where editing may be performed. The program print area can be utilized in this way to receive first the ALPHA edit format and then the BETA field prepared for printing.

The length of the BETA field processed depends upon the ALPHA field configuration. Care must be taken to define the format configuration to fit the data field length as well as the data configuration.

When the last operative ALPHA mask octet is the '20' and it is encountered in the zero-suppress mode, the mode of termination of the operation is not predetermined by the ALPHA field; the BETA field octet determines the mode in which the edit ends. If the right quartet of the last BETA octet is zero, the zero-suppression mode continues. If the right quartet of the last operative BETA octet is non-zero, the zero-suppression mode is terminated.

> NOTE: The only BETA octets suppressed by the EDT instruction are those read in the zero-suppression mode in the presence of a '20'. All others enter the ALPHA field.

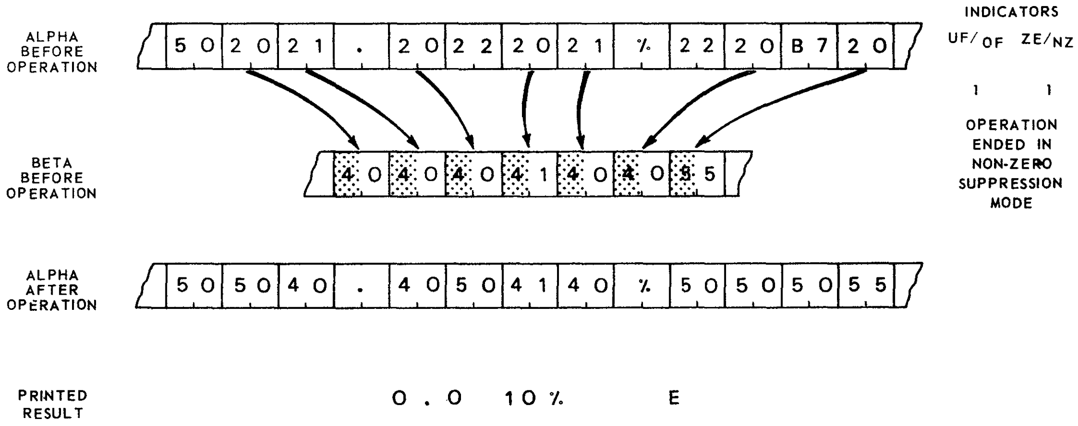
Figure B-4 ALPHA OCTETS AND THE RESULT OF THE EDT OPERATION IN EACH MODE.

	OCTET IN THE ALPHA FIELD	MODE WHEN ALPHA CHARACTER ENCOUNTERED			
		ZERO SUPPRESS		NON SUPPRESS	
		MODE BECOMES	ALPHA OCTET BECOMES	MODE BECOMES	ALPHA OCTET BECOMES
BETA OCTET ENTERS OPERATION	HEXADECIMAL '20' (BETA RIGHT QUARTET = 0)	NOT CHANGED	FILL CHARACTER	NOT CHANGED	BETA OCTET
	HEXADECIMAL '20' (BETA RIGHT QUARTET ≠ 0	NON SUPPRESS	BETA OCTET		
	HEXADECIMAL '21'	NON SUPPRESS	BETA OCTET		
BETA OCTET DOES NOT ENTER OPERATION	HEXADECIMAL '22'	NOT CHANGED	FILL CHARACTER	ZERO SUPPRESS	FILL CHARACTER
	ANY OF GRAPHIC CHARACTER SET	NOT CHANGED	FILL CHARACTER	NOT CHANGED	NOT REPLACED

EXAMPLE:

Editing of a data field for printing.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			EDT		ALPHA(013),BETA
	10					
	15					
	20					
	25					
	30					



NOTE: BETA is not affected by operation.

TRANSLATE OCTETS TR

TR ALPHA (nnn), BETA

Each octet of the ALPHA field is used to generate an effective address for locating an octet in the BETA field. The referenced BETA field octet replaces the ALPHA field octet. Operation is left to right through the length of the ALPHA field (001 - 256 octets).

INDICATORS AFFECTED

none

NOTES:

- The left octet of the address translated by the assembler for the BETA field reference is used as a "basic" address. The value present in the right octet of the address is ignored. (The BETA field is assumed to begin at an address which is a multiple of 256).
- Each octet of the ALPHA field is used serially as an increment to the "basic" BETA address. Each address formed by the "basic" BETA address and the ALPHA octet is used to reference a BETA field octet. This BETA field octet replaces the ALPHA octet used to form the referencing address.
- Any octet configuration may appear in either field.
- The length of the BETA field used in the operation is a function of the maximum range of the values any ALPHA field octet may assume.

PROGRAMMING PRACTICES:

The TR instruction is designed to facilitate translation from one character set to another.

Translation is accomplished by defining the translation table (i.e., the set of desired configurations for data) in the BETA field in terms of the data to be translated in the ALPHA field. The operation replaces each of the octets in the ALPHA field by an octet from the table in the BETA field. Each ALPHA field octet becomes the locator of a position in the table. The BETA field table must be prepared so that the translated configuration is placed in the relative location generated by using the ALPHA octet itself as an increment to the "basic" BETA address.

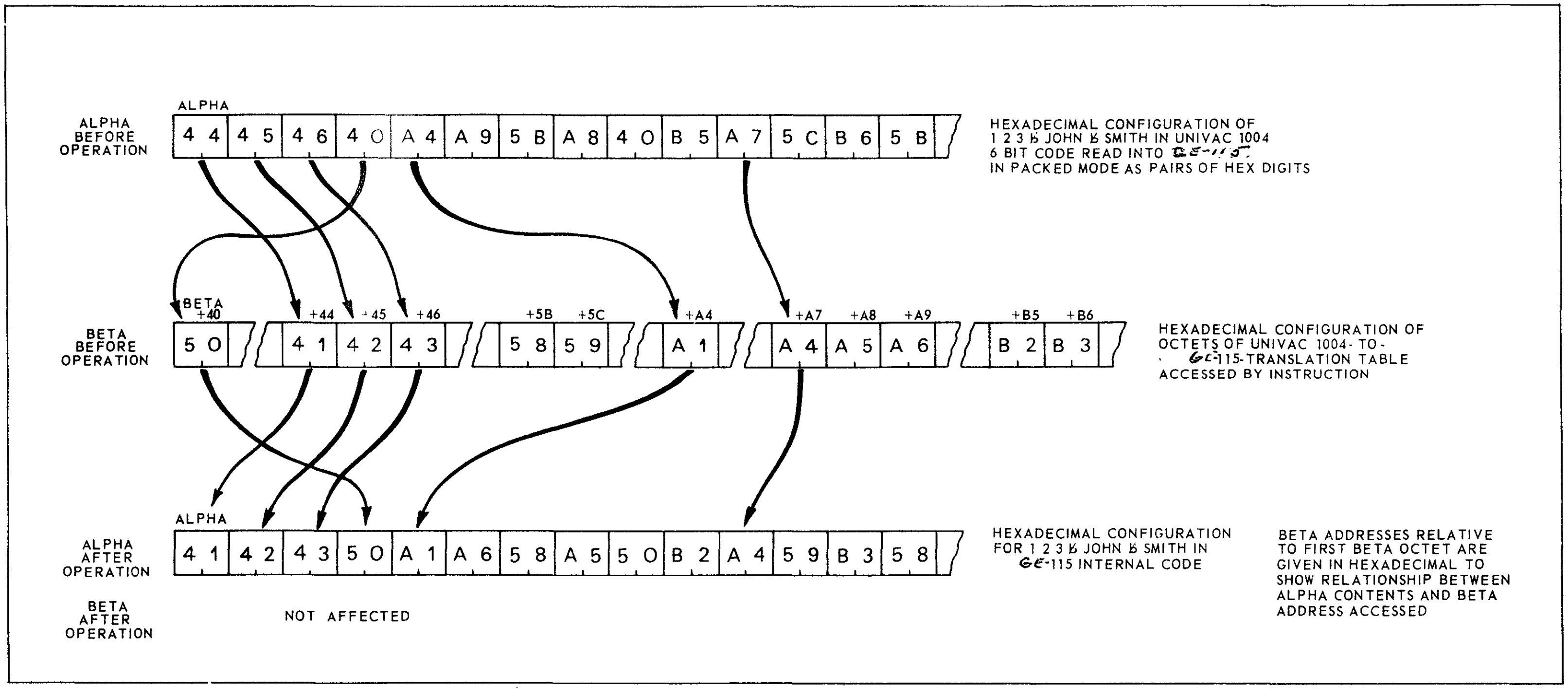
The values of the ALPHA field octet may range from 0 to 255. Therefore, the BETA field may require a maximum of 256 octets. The actual positions in the BETA field which are used by the TR operation are dependent on the possible ALPHA field values. If fewer than 256 different ALPHA field octet configurations may occur, only part of a set of 256 locations may be needed for the BETA field to translate ALPHA field values. The necessary BETA field locations may be contained within a range less than 256 octets, or the necessary locations may be scattered over the complete range of 256 octets. If it is known that the BETA field positions do not utilize parts of the full range of 256 octets, the remaining octets, outside the range required for the BETA field, may be used to contain other data.

When the BETA field begins at a multiple of 256, the Origin Assignment (ORG) instruction with an R in the operand specification field is used. The ORG instruction is followed by the necessary Define Constant (DC) instructions, or, if the BETA field is to be read into the store, a Define Store Area (DS) instruction.

EXAMPLE :

Translation of UNIVAC 1004 code image data which has been read into the GAMMA 115 from cards.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			TR		ALPHA, BETA
	10					
	15					
	20					
	25					



HEXADECIMAL CONFIGURATION OF 1 2 3 ƀ JOHN ƀ SMITH IN UNIVAC 1004 6 BIT CODE READ INTO GE-115. IN PACKED MODE AS PAIRS OF HEX DIGITS

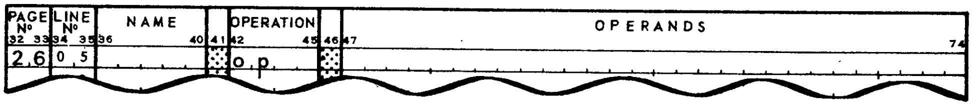
HEXADECIMAL CONFIGURATION OF OCTETS OF UNIVAC 1004-TO-GE-115-TRANSLATION TABLE ACCESSED BY INSTRUCTION

HEXADECIMAL CONFIGURATION FOR 1 2 3 ƀ JOHN ƀ SMITH IN GE-115 INTERNAL CODE

BETA ADDRESSES RELATIVE TO FIRST BETA OCTET ARE GIVEN IN HEXADECIMAL TO SHOW RELATIONSHIP BETWEEN ALPHA CONTENTS AND BETA ADDRESS ACCESSED

SYSTEM ACTION INSTRUCTIONS

The GE-115 Information Processing System operates upon data according to the instructions in the stored program. Some of the instructions can, however, direct system action which does not affect data in the store. These instructions set external indications and alter the status of system operation. The system action instructions have the following format:



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	47 OPERANDS 74
2 6	0 5			o p		

Note that the format of the System Action Instruction does not include specification of an operand.

HALT SYSTEM OPERATION HLT

HLT

System operation is terminated.

INDICATORS AFFECTED

none

NOTE:

- The system stops operation when the HLT is encountered. When the START button on the console is depressed, the program execution restarts at the next sequential operation.

PROGRAMMING PRACTICE:

The HLT may be used to separate a program into logical sections for checking. A HLT can be placed at the end of each logical section to stop program execution and allow expected results to be checked before they are used by subsequent sections. The programmer can generate programmed halts (See the Jump on Switch 1 (JS1) instruction on page 105 for a method of generating effective halts) for the same use. It is recommended that programmed halts be used rather than the HLT instruction. Messages should be included in the program and printed for the operator whenever operator action is required. Explicit instructions should be prepared for the operator describing the action to be taken. It is recommended that operator intervention be minimized.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			HLT		

NO OPERATION NOP2

NOP2

The system continues in sequence. No system operation is specified.

INDICATORS AFFECTED

none

NOTE :

- The NOP2 effects the advance of the store location counter by 2 octets.

PROGRAMMING PRACTICE :

A NOP2 may be used to overlay an instruction that is no longer needed for program operation . The NOP2 uses two octets instore. Therefore a series of NOP2 operations is needed to replace instructions of four or six octets.

EXAMPLE :

A NOP2 used to overlay an AD instruction. OVR is the name of a NOP2 instruction moved to the instruction named SUM.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
10	05	SUM		AD		ALPHA,BETA
	10			JC		X'30',OVFL
	15					
	20					
	25					
	30					
	85					
	90			*		NEED 3 NOPS FOR 6 OCTETS
	95			MVC		SUM(002),OVR
				MVC		SUM+002(004),SUM

TURN ALERT LIGHT ON LON

LON

The ALERT light on the console is turned on.

INDICATORS AFFECTED

none

NOTE:

- The LON turns on an external signal light. The system continues in sequential operation.

PROGRAMMING PRACTICE:

The LON may be used, along with the Turn ALERT Light Off (LOFF) instruction, to indicate some required operator action. The need for the operator action may be signalled by the LON. A test should be made whenever possible to determine whether the required action has been carried out. Explicit instructions should be prepared for the operator describing the action to be taken. Messages should be included in the program and printed to inform the operator of the required action. When the action has been completed the LOFF can be used to turn off the light. It is recommended that operator intervention be minimized.(See the Turn ALERT Light Off (LOFF) instruction on page 121).

EXAMPLE:

The ALERT light is turned on.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
26	05			LON		
	10					

TURN ALERT LIGHT OFF LOFF

LOFF

The ALERT light on the console is turned off.

INDICATORS AFFECTED

none

NOTE:

- The LOFF instruction turns off an external signal light. The system continues in sequential operation.

PROGRAMMING PRACTICE:

The LOFF may be used with the Turn ALERT Light On (LON) instruction to signal the need for operator intervention (See the LON instruction on page 120).

EXAMPLE:

The ALERT light is turned off.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
26	05			LOFF		
	10					
	15					
	20					
	25					

INHIBIT SINGLE STOP INS

INS

The SINGLE STOP switch on the control console is disabled.

INDICATORS AFFECTED

none

NOTE:

- The INS places the system in a continuous operation state and prevents interruption of the program by the use of the SINGLE STOP switch.

PROGRAMMING PRACTICE:

The INS may be used with the Enable Single Stop (ENS) instruction to perform a check of a program segment. (See the ENS instruction on page 123).

EXAMPLE:

The SINGLE STOP switch is disabled.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
56	05			INS		
	10					

ENABLE SINGLE STOP ENS

ENS

The SINGLE STOP switch on the control console is enabled.

INDICATORS AFFECTED

none

NOTES:

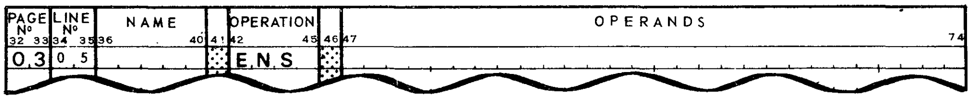
- The ENS operation allows the system to be operated step-by-step, using the SINGLE STOP switch. One instruction is executed each time the switch is set. The switch remains operative until an INS instruction is encountered.

PROGRAMMING PRACTICE:

The ENS may be used with the Inhibit Single Stop (INS) instruction to perform a check of a program segment. ENS allows the operator to stop execution of the program after each instruction has been executed. When the segment has been checked, the system is returned to normal, continuous operation by the INS. Whenever operator intervention is required, explicit instructions should be prepared describing the action to be taken. Messages should be included in the program and printed to inform the operator what action is required. It is recommended that operator intervention be minimized.

EXAMPLE:

The SINGLE STOP switch is enabled.



PAGE N° 32 33	LINE N° 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
0 3	0 5			E N S		

The Primary instructions previously discussed process data in the GE-115 Information Processing system. The instructions discussed in this section provide the means of bringing data into the system for processing and for printing or punching the results of that processing.

The system discussed is the card system. All input and output operations are described in terms of card input and card or printer output. Input/output operation charts included in this section contain references to operations and functions which apply to other forms of input and output. These other forms are not treated here. However, the chart information is included for amplification of the materials that will be released as the system is developed.

Data may be brought into the GE-115 system in a number of forms. Hollerith card codes and several special card codes (with hardware adaptation and/or software translation) may be read and punched. Card code formats can be specified under programmed control. Card formats may use one column to represent the contents of an octet or they may use two columns where internal configurations are to be expressed in the external media in two parts. Sixty-four graphic characters may be used for printing the results of program action.

The input/output operations for the GE-115 may be programmed to make optimum use of the system. The input/output operations, such as card reading and printing, require access to the central processor during their execution. Card punching, on the other hand, requires the action of the central processor during the time of preparation of the output which can be punched from an intermediate retention area without further central processor action.

For those operations which require data transfer during their execution, an optimizing use of the central processor is possible. The central processor can receive or supply data at a faster rate than the input/output (peripheral) units. This means that the central processor is free during part of the execution time required by the instruction. This free time may be utilized in the GE-115 by a method of input/output operation called time sharing.

Time sharing is accomplished by the provision of two types of input/output operation. The first type is called a presetting operation. A preset may be given for input or output operations. A presetting instruction defines an operation completely, giving the unit, channel, data area, and length of operation. The operation code requests preparation of the channel mechanism only, not initiation of the operation.

The second type of input/output operation is the execute. An execute operation may be given for input or output. An execute instruction contains the required channel, unit, data area, and length. The operation code requests that the operation be initiated when the instruction is processed. To accomplish time sharing, an input execute instruction must follow an output "preset" instruction. The instructions must be given on separate channels of communication. An execute input instruction which follows a preset output instruction causes the output instruction to be initiated as well. The input and the output unit share the time of access to the central processor.

The pròcedure for time sharing is :

1. An output preset operation, utilizing channel 2, is given.
2. An input execute operation, utilizing channel 1, is given.

The capabilities of the GE-115 allow for even further optimization of the input/output operations. The punch equipment utilizes an intermediate area for data retention. This makes it possible to have three input/output operations taking place; two on a time sharing basis and the third simultaneously with whichever of the other two is operative at any time. To use the punch in conjunction with the reader and printer, it is necessary that the punch be utilized through connector 3. When the punch operation is to take place in conjuction with a time sharing read and write operation sequence, the punch instruction must precede the output preset instruction for channel 2.

Time sharing is directed toward the optimal use of the central processor during input/output operations. There is another consideration of timing in the use of peripheral units that may be specified by program control as well. This refers to the optimum utilization of a given peripheral unit.In this purpose, input/output operations may be specified as wait or immediate instructions.

Immediate operations are requests for some peripheral unit action to take place when the instruction is given. Wait operations imply an interrogation of the status of the peripheral unit referenced. If the unit is occupied, i.e., engaged in some operation previously requested, the instruction is not carried out when it is given. When the prior operation is completed, the wait operation takes place.

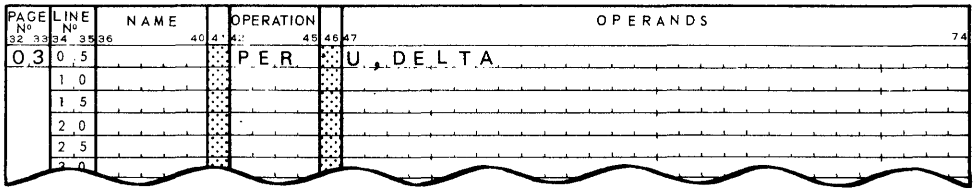
It must be noted that it is not meaningful to use immediate operations in all possible sequences of input/output operation. Instructions are executed by the GE-115 in the order in which they are placed in the store. Operation is sequential. An operation which utilizes the central processor must be completed before another can be interpreted. Therefore, an immediate input/output operation which follows a data transfer operation is not, in any case, interpreted until the data transfer is completed.

In some instances, a meaningful sequence of operations may utilize the immediate instruction for peripheral unit control. For example, an instruction to select a card stacker might be given in the immediate mode following an instruction to reset card read error. On the other hand, an instruction to select a card stacker given in the immediate mode following a card read operation would not be a meaningful sequence.

There are three types of operation which make reference to the input/output units. They are :
Data Transfer Operations
Peripheral Status Test Operations
Peripheral Unit Control Operations

All three types of operation are performed by means of a single input/output initiating instruction, the Call Peripheral (PER) instruction.

The format of the PER instruction is :



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
03	05			PER		U , DELTA
	10					
	15					
	20					
	25					

where
U specifies a peripheral unit (see figure B-5),
and
DELTA is the name of a data field which contains the operation specification. The content of the DELTA field determines the operation actually performed by the specified unit. The DELTA field may have one of three different basic configurations depending on the type of operation described. There is a special data definition instruction, the Define Peripheral (DP) instruction, for use in setting up the DELTA fields.

The format and content on the DELTA field vary with the operation being performed. There are two possible lengths. The DELTA field is :

6 octets long for the Data Transfer oparations,
2 octets long for Status Test operations, and
2 octets long for Unit Control operations.

Input/output instructions have the following general characteristics :

- The PER instruction initiate an input/output operation using a specified unit.
- A data field complete the operation definition and always contains the operation specification.

and the channel request. A define Peripheral (DP) Directive is used to set up the data field.

Data Transfer Operations

Data transfer operations have the following general characteristics :

- The data field operand which amplifies the operation is six octets in length.
- The first two octets of the data field operand contain an instruction specification.
- The second two octets of the data field operand contain the length of the data field which participates in the transfer.

  The length of a print operation is governed by the print line length of the printer model used. The length of a read or punch operation is governed by card length, i.e., 80 columns for a card.
- The fifth and sixth octets of the data field operand specify the location of the first octet in the store which participates in the data movement.
- Data fields for transfer are referenced at the left. Data is transmitted and received serialy, octet by octet, from left to right. (Reference is made in the input/output configuration to the use of descending addresses. This usage does not apply in the present context.)
- Indicators are set to record the results of conditions such as end of input and transmission error.

Peripheral Status Tests

The peripheral status test operations have the following general characteristics :

- The data field operand in a 2-octet field specifying the operation and the condition.
- A condition recorded during the use of a specified peripheral unit is tested.
- Indicators are set to record the results of the test.

Peripheral Unit Control

The peripheral unit control operations have the following general characteristics :

- The data field operand is a 2-octet field which specifies the operation and the mode of execution.
- A peripheral unit is instructed to perform some operation that does not directly involve data transmission, e.g., eject present printer page.

HEXADECIMAL CONFIGURATION	UNIT
00	CARD PUNCH
80	CARD READER
C0	PRINTER

Unit numbers given should be verified for use with the GE-115 system being programmed.

Figure B-5 UNIT NUMBERS

The GE-115 system provides prepared input/output programs which may be used with other programs. These prepared programs may be incorporated, according to conventions of use which depend on the program used, into other programs written for operation on the GE-115. The input/output programs are written for defined peripheral unit configurations. The programmer is advised to secure the input/output programs which may be used with the configuration of the GE-115 system he is using.

CALL PERIPHERAL PER

Data Transfer

PER U, DELTA

The unit specified in the PER instruction is selected to perform the data transfer operation defined in the DELTA field. Data is received or transmitted serially, octet by octet, from left to right through the specified field length.

INDICATORS AFFECTED

UF/OF	ZE/NZ	
1	0	Operation terminated under the specified count control.
1	1	An end of input signal was received.

INPUT/OUTPUT TRANSMISSION INDICATORS

Channel 1 Parity	Channel 2 Parity	
0	0	Transmission valid
1	1	Transmission parity error detected

NOTES :

- The DELTA field first octet specifies :
  Channel
  Direction of transfer - input or output
  Data format - packed or unpacked
  Time sharing status - set for time share preset, or an execute
  Operation mode - wait or immediate
  Data reference direction - ascending or descending locations
  (See Figure B-7)
- The DELTAfield second octet specifies the operation requested, as shown in Figure B-6, below :

Figure B-6 : OPERATION REQUEST CODE

Hexadecimal Configuration	Operation Requested
40	Read cards
42	Print or Punch

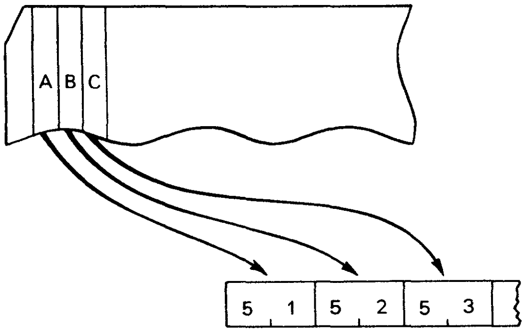
- The DELTA field third and fourth octets specify the length of the data field which participates in the data transfer operation.

- The DELTA field fifth and sixth octets specify the location of the first octet in the store which participates in the data transfer operation.

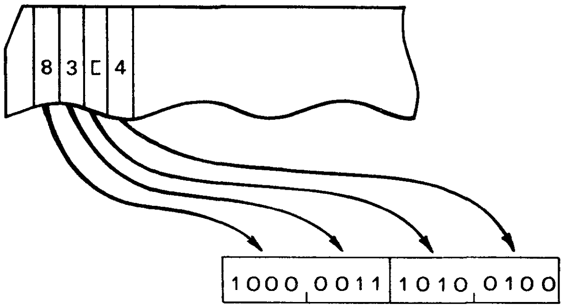
- Data transfer operations may time share the central porcessor. A pair of input/ouput operations designed to effect time sharing is given as follows :
  1. An output preset operation is specified, using channel 2.
  2. An input execute instruction is specified, using channel 1. When the channel 1 operation is initiated, the channel 2 operation is initiated as well. The input/output data transfer operation is completed when the longer of the two requests is completed.

- Data is read or written from left to right (ascending locations) by the card and printer operations.

- Cards may be read in packed or unpacked form.
  Unpacked form is standard Hollerith card code. Each column generates an octet as shown below :



Packed form is used to generate a single octet from two columns.



Only the right quartet of the standard internal configuration of the card column enters the store locations used.

- When an input/output operation, either presetting or execute, is given on channel 2, the channel 2 transfer parity error indicator is set to 0.

- When an input/output operation is given on channel 1, the channel 1 transfer parity error indicator is set to 0.

- At the end of a data transfer operation on channel 1, LOC (store octets 0254 and 0255) contains the location of the octet to the right of the last octet which participated in the data transfer.

- Data transfer operation which use channel 2 must reference a data field which has an address that is a multiple of 256 plus 2, i.e., of the form 256m+2. The contents of the two left most octets are used to control the operation, which is here assumed to be print only. (The length of the field printed is, as noted, dependent upon the physical characteristics of the printer model, and is not given here).

- Locations which participate in an output data transfer operation are unaltered by the operation. The two print control octets to the left of the field to be printed are, however, altered by the operation.

- The ZE/NZ indicator is used to record the cause of the termination of an input data transfer operation. An input data transfer operation may be terminated when a field of the specified length has been filled or when the end of the input has been detected.
  A 0 in the ZE/NZ indicator at the end of an input data transfer operation indicates that the operation terminated when a field of the requested length was transferred.

  A 1 in the ZE/NZ indicator at the end of an input data transfer operation indicates that an end of input signal was received.

- A 1 in the applicable channel parity error indicator at the end of operation on either channel indicates that a parity error was detected during the transfer of the data.

PROGRAMMING PRACTICES :

Parity error indicators are reset prior to the initiation of input/output data transfer operations and should be tested after each operation. A peripheral status test operation followed by a conditional jump tests for parity error.

A full print line is always printed. The programmer should, therefore, make certain that any unused positions are cleared before printing takes place.

The card punch buffer is cleared after punching so a partial card may be punched without the requirement that blanks be supplied for the unused columns.

The output area for channel 2 must be defined as 256m+2. An Origin Assignment (ORG) instruction with an operand specification of R can be used to define the print area. (See the ORG Directive, page 160).

EXAMPLE

Time Sharing Sequence

A output print operation is given in the preset mode, followed by an input read cards operation in the execute mode. Specification of preset or execute mode is made in the first octet of the DELTA field referenced by the instruction, as shown. After completion of both operations, tests are made for transmission errors as well as for end of cards and paper.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			*		INPUT OUTPUT TIME SHARE
	10			*		PRESET FOR CHANNEL 2
	15			*		EXECUTE FOR CHANNEL 1
	20	PREST		DP		X'59',X'42',PR1NT+002(0136)
	25	EXEC		DP		X'10',X'40',READ
	30			*		
	35	READ		DS		L080
	40			*		
	45			*		SET UP PRINT AREA NEEDED
	50			ORG		R 256+2
	55	PRINT		DS		L138
	60			*		
	65	TEST2		DP		X'CI',X'44',TEST CHANNEL 2
	70	TEST1		DP		X'C0',X'44',TEST CHANNEL 1
	75			PER		X'C0',PREST
	80			PER		X'80',EXEC
	85			PER		X'C0',TEST2 TEST PRINT ERROR
	90			JC		X'10',ERROR TO ERROR ROUTINE
	95			*		NO WRITE ERROR
	96			PER		X'80',TEST1 TEST READ ERROR
	97			JC		X'10',ERROR TO ERROR ROUTINE
	98			*		NO READ ERROR
	99					

Figure B-7: PERMISSIBLE CONFIGURATIONS OF THE FIRST OCTET IN A DATA TRANSFER INSTRUCTION

HEX VALUE	INPUT	OUTPUT	ASC. ADDR.	DESC. ADDR.	PACKED DATA	UNPACKED DATA	EXECUTE INST.	PRESET. INST.	WAIT	IMMED.	CHANNEL 1	CHANNEL 2
00	X		X		X		X		X		X	
04	X		X		X		X			X	X	
10	X		X			X	X		X		X	
11	X		X			X	X		X			X
14	X		X			X	X			X	X	
15	X		X			X	X			X		X
19	X		X			X		X	X			X
20	X			X*	X		X		X		X	
24	X			X*	X		X			X	X	
30	X			X*		X	X		X		X	
34	X			X*		X	X			X	X	
40		X	X		X		X		X		X	
44		X	X		X		X			X	X	
50		X	X			X	X		X		X	
51		X	X			X	X		X			X
54		X	X			X	X			X	X	
55		X	X			X	X			X		X
59		X	X			X		X	X			X
60		X		X	X		X		X		X	
70		X		X		X	X		X		X	
74		X		X		X	X			X	X	
5D		X	X			X		X		X		X

* for use with magnetic document readers only

Peripheral Status Test

PER U, DELTA

The status of the peripheral unit specified in the PER instruction is tested according to the specification given in the DELTA field.

INDICATORS AFFECTED

UF/OF	ZE/NZ	
1	0	Test condition not present
1	1	Test condition present

NOTES :

- The DELTA field first octet specifies the operation and the channel to be used (See Figure B-8)
- The DELTA field second octet specifies the condition which is to be tested (See Figure B-9)

PROGRAMMING PRACTICE :

A Jump on Condition (JC) instruction must be used to interrogate the indicator set in response to the status test operation.

EXAMPLE

A test is made for the end of a printer page. DELTA contains the operation specification.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05	DELTA		DP		X'C1',X'05'
	10					
	15					
	20					
	25			PER		X'C0',DELTA

Figure B-8 MODE AND CHANNEL SPECIFICATION FOR STATUS TEST INSTRUCTIONS

1st Octet	Specification
C0	Wait until the peripheral is free; use channel 1
C1	Wait until the peripheral is free; use channel 2
C4	Execute immediately on channel 1
C5	Execute immediately on channel 2

Figure B-9 STATUS TEST SPECIFICATIONS

2nd Octet	Condition tested
01	Controller ready
03	Error in transmission
05	End of cards
05	End of Page
12	Hopper Empty
12	End of Paper
14	Out-of-Service
1E	Stacker Full
42	Data Transfer Error Channel 2
44	OR, of any of the preceding tests applicable to a given peripheral unit
10	Cards ready to feed
2E	OR, of
	End-of-Service
	End-of-Medium
	End-of-File
	Error in transmission

CALL PERIPHERAL PER

Peripheral Unit Control

PER U, DELTA

The peripheral unit specified in the PER instruction is selected to perform the control operation specified by the DELTA field.

INDICATORS AFFECTED

UF/OF	ZE/NZ	
1	0	set prior to operation

NOTES:

- The DELTA field first octet specifies the mode of operation and the channel to be used. (See Figure B-10)
- The DELTA field second octet specifies the control operation to be performed. (See Figure B-11)
- The UF/OF indicator is made 1 and the ZE/NZ indicator is made 0 prior to operation. Neither is affected by the operation.

PROGRAMMING PRACTICES:

If the status of the UF/OF and/or ZE/NZ indicator is meaningful, it should be saved or utilized prior to the peripheral unit control operation.

The peripheral unit control operation may be used to reset some error conditions detected by the peripheral status operation, namely, a read or punch error indication.

Spacing of the printer pages may be performed using Peripheral Unit Control PER instructions. The operations affect the spacing of the printer page according to a format controlled by the position of punches in a paper tape loop inserted in the printer.

The spacing operation performed by the selection of a given carriage control tape channel should be checked against the information provided with the printer which is used.

The Bypass operation, referred to in Figure B-11, is not discussed. Information about its use will be given in future documents.

The Feed cards operation is utilized with some card reader models. Information provided with the equipment should be checked.

Figure B-10 :CHANNEL SELECTION FOR PERIPHERAL CONTROL

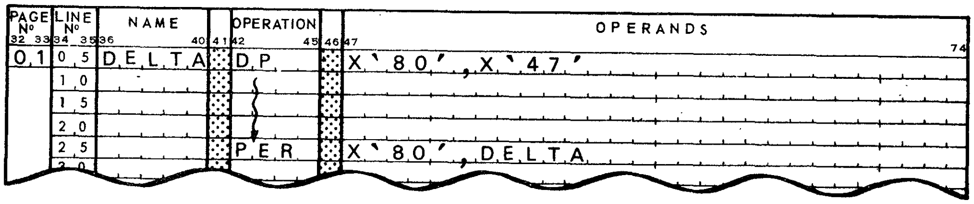
1st octet	Specification
80	Use channel 1; wait until the peripheral unit is free
81	Use channel 2; wait until the peripheral unit is free
84	Use channel 1; execute immediately
85	Use channel 2; execute immediately

Figure B-11 : PERIPHERAL OPERATIONS FOR UNIT CONTROL

2nd octet	Action Requested	
0A	Single Space	
0C	Feed Card	
47	Reset Error	
48	Select Stacker	
51	Vertical Paper Throw, channel 1	printer carriage control paper tape loop
52	Vertical Paper Throw, channel 2	
57	Vertical Throw, channel 7	
59	Double Space	
A0	Switch on Bypass	
A1	Switch off Bypass	

EXAMPLE :

A read error is reset on the card. DELTA contains the operation specification.



PAGE No 32 33	LINE No 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05	DELTA		DP		X'80',X'47'
	10					
	15					
	20					
	25			PER		X'80',DELTA
	30					

PART II

DIRECTIVE INSTRUCTIONS

Directive instructions specify action to be taken by the assembler rather than by the system. Directive instructions are not translated into executable machine language instructions; they provide parameters for use by the assembler in setting up data fields and give direction for assembler action and program loading.

Directive instructions are written in the same format as the Primary instructions, according to the rules presented in SECTION A, PART II, "WRITING STATEMENTS IN THE GE-115 ASSEMBLY LANGUAGE". There are additional conventions used in specifying the operand fields of the definition Directive instructions. These are explained in the description of the Directives, below.

All the Directives of the GE-115 Assembly Language are described in this section. The Directive instructions are grouped according to similarities of assembler action as shown below:

DEFINITION - Instructions which direct the assembler to allocate store areas and define data:

Define Store Area	DS
Define Constant	DC
Define Peripheral Field	DP

PROGRAM CONTROL - Instructions which direct operations of the assembler that affect the assembled program:

Start Program	STRT
End Program	END
Origin Assignment	ORG

ASSEMBLY LISTING FORMAT - Instruction which direct operations of the assembler that affect the format of the listing produced by the assembler during assembly of the program :

Comment	*
Eject Present	EJEC
Line Feed	LF

The format shown in Figure B-1 in the introduction to the Primary Instructions is used also to explain each of the Directive instructions. The conventions of notation described in the discussion of the Primary instructions apply to the Directives as well. There is an additional notation used in the descriptions of the definition instructions, as shown below :

ddd	The use of "d", written in lower case indicates that a three digit duplication factor is written with a field definition.
(nnnn)	The use of "n", written in lower case is used for field lenght. A four digit lenght may be specified in the DP Directive.
constant	The use of the word "constant" written in lower case indicates that character, hexadecimal, or address constants may be defined. The ways of writing the constants are explained in the description for each of the instructions for defining constants.
descriptor	The use of the word "descriptor" written in lower case indicates that an octet is defined which specifies the characteristics of a peripheral operation. The descriptor may be written in any of the ways an immediate data item is written. (See WRITING STATEMENTS IN THE GE-115 ASSEMBLY LANGUAGE). It is recommended that the hexadecimal representation be used.
operation	The use of the word "operation" written in lower case indicates that an octet is defined specifying a type of peripheral operation. The operation may be written in any of the ways an immediate data item is written. It is recommended that the hexadecimal representation be used.

Definition statements direct the assembler to allocate store area to data fields and to generate constant values to be incorporated into the assembled program. Names may be associated with data fields to permit field references in the source program. Every named data field must be defined by a definition statement. The assembler uses the information contained in the definition statements (name, length, area reservation) to translate field references and assign locations to data and constants. Defined constants are included as data in the assembled program.

Definition statements are operative only at assembly time. At execution time they are present only in the form of defined constants and data fields. If they are placed between executable instructions, the system will encounter them in the course of sequential instruction execution and will attempt to interpret data as instructions. Program results are unpredictable in such cases. If definition statements are included between executable instructions they must be preceded by an unconditional jump to the next instruction to be executed. It is strongly recommended that the programmer avoid this waste of store area and operating time by placing all data and constant fields outside the sequence of (i.e., before or after) executable instructions.

The Define Store Area (DS) instruction has the following characteristics:

- A field length is specified in the instruction. Duplication factors may specify that store area is to be reserved for 1 to 256 fields. Each field has the length specified.
- The name of the field is associated with a length. The field name and length are saved for use in translating primary instructions.

The Define Constant (DC) and the Define Peripheral Field (DP) instructions have the following characteristics:

- A single field may be specified; no duplication factor is used.
- The operand specification field of the source language instruction contains constant data which is translated by the assembler into the internal configuration of the data and which is included in the assembled program.
- The name of the field is associated with a length. The field name and length are saved for use in translating the primary instructions.

The definition instructions differ in the way length is specified in each:

- The Define Store Area instruction requires length specification; the length defined may be from 1 to 256 octets. A duplication factor may be used.
- The Define Constant instruction requires a length specification; the length of a data constant may be from 1 to 10 octets. No duplication factor may be specified.
- The Define Peripheral Field has an implied length which is either 2 or 6 octets. Neither length nor duplication factor may be specified.

Length is specified by an L and three decimal digits. If a duplication factor is specified it is written before the L and is a three digit decimal number.

The format of the store definition instruction (DS) is:

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
05	05	name		DS		dddLnnn
	10					
	15					
	20					
	25					

The formats of the data definition instructions (DC and DP) are:

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
05	05	name		DC		ALnnn(nnnn) binary equivalent
	10	name		DC		CLnnn 'constant' alphanumeric
	15	name		DC		XLnnn 'constant' hexadecimal
	20					
	25					

and:

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
05	05	name		DP		descriptor, operation, ALPHA(nnnn)
	10					
	15	name		DP		descriptor, operation
	20					
	25					

DEFINE STORE AREA DS

DS ddd L nnn

The assembler is directed to reserve from 0 to 256 fields in the store. The length of each field may be from 1 to 256 octets.

NOTES:

- The assembler reserves the requested number of contiguous fields in the store. Each field is given the length specified.
- The assembler advances the store location counter by (ddd) x (nnn) octets.
- When no explicit number of fields is requested, a single field is reserved.
- When the number of fields requested is explicitly 000, the assembler does not alter the store location assignment counter. No area is reserved. However, the name and length of the field are saved for translation of field references.
- When a name is written with a DS instruction which requests store area for more than one field, the name is associated with the first field reserved. The length associated with the name is nnn.

PROGRAMMING PRACTICE:

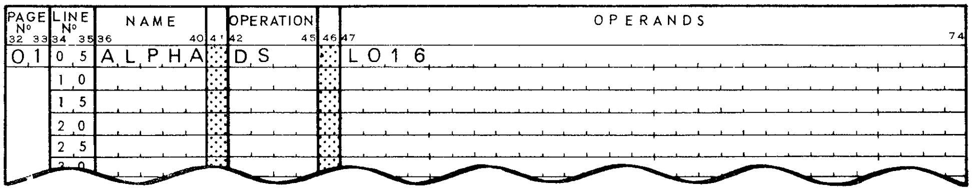
The DS is used to name and reserve data areas in the store. Data may be generated by the program and placed in the reserved areas, or may be read into the areas allocated.

The DS with a duplication factor of 000 may be used to name a major field which contains named subfields. To accomplish this, the programmer gives a name and a duplication factor of 000 to the major data field. The named subfields are assigned duplication factors of at least 001. The major field name is associated with a length but does not cause store to be reserved. The subfield definitions each cause a name and length to be associated with a reserved store area. The total store area reserved by all subfields should be equal in length to the length specified for the major field they constitute.

The DS with a duplication factor of 000 may be used to assign several data areas at the same point because no store area reservation takes place.

EXAMPLES:

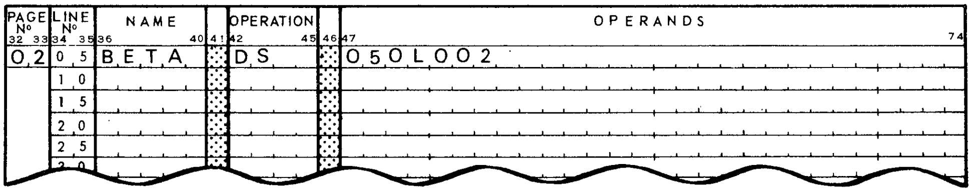
1) Assigning a left octet address, a right octet address and a length to a named field.



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05	ALPHA		DS		L016
	10					
	15					
	20					
	25					
	30					

The assembler assigns a left octet address, a length and a right octet address to the name, ALPHA. The store location assignment counter is increased by 16 octets.

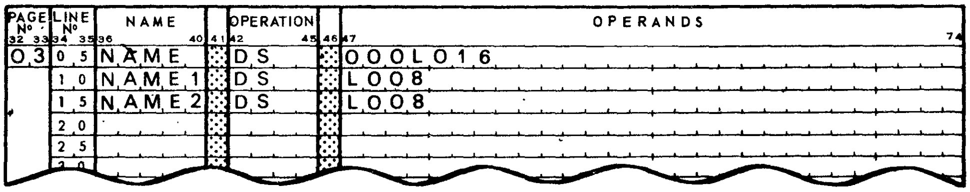
2) Use of the duplication factor to reserve store area.



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05	BETA		DS		050L002
	10					
	15					
	20					
	25					
	30					

The assembler assigns a left octet address, a right octet address and a length of two octets to BETA. The duplication factor causes the store assignment counter to be increased by 050 x 002 = 100 octets.

3) Use of a duplication factor of 000 to define subfields within a field.



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
03	05	NAME		DS		000L016
	10	NAME1		DS		L008
	15	NAME2		DS		L008
	20					
	25					
	30					

NAME 1 and NAME 2 are subfields of the field NAME.

The assembler assigns two addresses and a length of 16 octets to NAME but reserves no store area.

The assembler assigns two addresses and a length of 8 octets to NAME 1 and reserves 8 octets of store for the field.

The left octet address of NAME and NAME 1 are the same.

The assembler also assigns two addresses and a length of 8 octets to NAME 2, and reserves store area.

The right octet address of NAME 2 is the same as the right octet address of NAME.

The three definition statements cause the store assignment counter to be increased by 0 + 8 + 8 = 16 octets.

DEFINE CONSTANT DC

Character Constant

DC CL0nn'.......'

The assembler is instructed to translate the specified character constant contained within the pair of apostrophes. If the DC instruction is named, the length (001 to 010 characters become 1 to 10 octets) and store addresses are saved for translating symbolic references to the constant field.

NOTES:

- The assembler translates the specified constant into the internal format used by the system.
- Each graphic character is translated into a full octet.
- When the length specification exceeds the number of characters written in the operand specification, the assembler fills the field with blanks to produce a field of the specified length. Blanks are placed at the right of the explicit constant.
- When the length is less than the number of characters written it is a mistake (See Figure A-7, GE-115 ASSEMBLER MISTAKE CODES).

PROGRAMMING PRACTICES:

The character constant definition may be used to prepare both numeric values which serve in arithmetic (decimal) operations and alphanumeric fields for printing.

Note: The first digit of the length must be zero; a maximum of 10 octets may be specified.

EXAMPLES :

1) Using the DC to define a character constant.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05	ITEM		DC		CL009'@$1.25'
	10					
	15					
	20					
	25					

The assembler assigns a left octet address, a right octet address and a length of 9 octets to ITEM and generates for placement in the field the internal representation of the defined constant. As the specified length (9 octets) is greater than the explicit length (6 characters), 3 blanks will be inserted in the field to the right of the explicit constant.

2) Assigning a name to a defined constant longer than 10 octets.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05	DEPT		DS		000L020
	10			DC		CL010'TECHNICAL'
	15			DC		CL010'ASSISTANCE'
	20					
	25					

The assembler assigns a left octet address, a right octet address and a length of 20 octets to DEPT and generates for placement in the field the internal representation of the two defined constants, reserving 20 octets of store area for the defined constants.

3) Special case DC statement for use of the apostrophe.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
03	05	APOST		DC		CL002'''' Use of "C" type DC stmt
	10	APOST		DC		XL002'AF' hexadecimal configuration
	15					
	20					
	25					

DEFINE CONSTANT DC

Hexadecimal Constant

DC XL0 nn' ......'

The assembler is instructed to translate the specified hexadecimal characters contained within the pair of apostrophes. If the DC instruction is named, the length (001 to 010 digit pairs become 1 to 10 octets) and store addresses are saved for translating symbolic references to the constant field.

NOTES:

- The assembler translates the specified constant into the internal format used by the system.
- When the length specified exceeds the number of digit pairs, the assembler creates full octet zeros in the left of the defined field for each pair omitted.
- When the length specified is less than the number of digit pairs written, it is a mistake. (See Figure A-7, GE-115 ASSEMBLER MISTAKE CODES).

PROGRAMMING PRACTICE

The hexadecimal constant definition may be used to prepare translating tables or editing masks. (See TR, page 114 , and EDT, page 109). Codes which cannot be read as graphic characters may be placed in the store as hexadecimal digit pairs.

Note : The first digit of the length specification must be zero. A maximum of 10 octets may be specified.

EXAMPLE:

Use of the DC Hexadecimal Constant to define on editing mask.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05	MASK		DC		XL009'5020BB2020215B2020'
	10					
	15					
	20					
	25					

The assembler is directed to assign addresses and a length of 9 octets to MASK, and to generate and store in the 9 reserved octets the internal representation of the defined hexadecimal constant. The special characters of the editing mask do not have graphic representations : therefore they must be defined as hexadecimal constants.

DEFINE CONSTANT DC

Address Constant

DC AL00n (ALPHA)

The assembler is instructed to translate the specified address reference contained within the pair of parentheses. If the DC instruction is named, the length (001 or 002 octets) and location of the constant in the store are saved for translating references to the constant field.

NOTES:

- Translation of an address constant depends upon the way the address is specified. The address specified may be written in any of the formats used for operand addresses. A symbolic name of a field or instruction is translated as the left octet address of the field. Any increment or decrement is computed from that address.

  An absolute address (never given an increment or decrement), written as four decimal digits, is translated as the internal equivalent of the number.
- When the program is loaded, the translated address is placed in the specified number of octets. Addresses require two octets. When only one octet is specified, the rightmost of the pair generated is used in the DC field.

PROGRAMMING PRACTICES:

The address constant may be used to set up a value to reset an address modified during program execution.

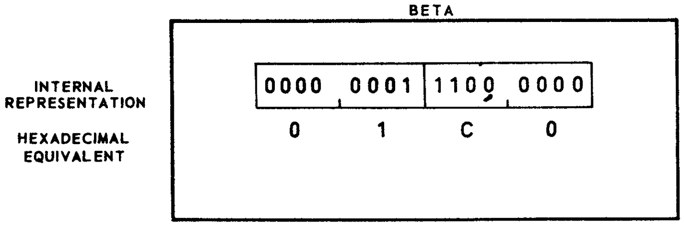
Note: The first two digits of the length specification must be zero ; a maximum of 2 octets may be specified.

EXAMPLES:

1) BETA is defined as an address constant of two octets:

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
03	05	BETA		DC		AL002(0448)
	10					
	15					
	20					
	25					

The assembler assigns a field of two octets to the name BETA and stores in the field the internal representation of the defined address constant.

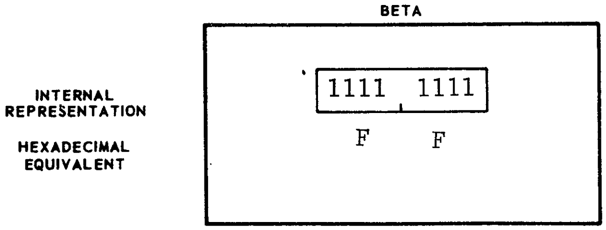


2) BETA is defined as an address constant of one octet:

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05	BETA		DC		AL001(ALPHA+001)
	10					
	15					
	20					
	25					

The assembler assigns a field of one octet to the name BETA and stores in the field the internal representation of the low-order octet of the generated address.

ALPHA has been defined as a 4 octet field, stored in locations 0510-0513.



DEFINE PERIPHERAL INSTRUCTION DP

Data Transfer

DELTA DP descriptor, operation, ALPHA (nnnn)

The assembler is directed to set up a 6-octet field for reference by a data transfer Call Peripheral (PER) instruction.

NOTES:

- The first octet specifies :

  Channel
  Direction of transfer-input or output
  Data Format - packed or unpacked
  Time sharing status - preset or execute
  Operation mode - immediate or wait
  Data Reference direction - right to left or left to right

- The second octet specifies the operation
- The third and fourth octets specify the length of the data field which participates in the transfer operation.
- The fifth and sixth octets specify the location of the first octet in the store which participates in the data transfer operation.

PROGRAMMING PRACTICES :

The hexadecimal configuration of the first octet is given in Figure B-7 of the second octet in Figure B-6 .Any of the forms for specifying an immediate operand may be used, but it is recommended that the hexadecimal configurations shown in the table be given in the standard hexadecimal specification format.

In ALPHA field reference indicates the location of the first octet that is to receive data or of the location of the first octet to be transmitted. ALPHA may be written in any of the standard address reference forms. If a length is specified, four digits are used to write field length.

EXAMPLE :

The assembler is instructed to set up a data field for a card read operation. A card is to be read into the field ALPHA. Assume ALPHA to be located at store location 0830.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05	DELTA		DP		X'10',X'40',ALPHA(0064)
	10					
	15					
	20					
	25					

The assembler assigns a field of six octets to the name DELTA and stores in the field the internal representation of the defined peripheral constant.

INTERNAL REPRESENTATION	00010000	01000000	00000000	01000000	00000011	00111110

HEXADECIMAL EQUIVALENT

$$1\ 0\ 4\ 0\ \underbrace{0\ 0\ 4\ 0}_{\text{Length = 64}}\ \underbrace{0\ 3\ 3\ E}_{\text{Location Alpha}}$$

DEFINE PERIPHERAL INSTRUCTION DP

Peripheral Status Test

DELTA DP descriptor , operation

The assembler is directed to set up a two octet field for reference by a peripheral unit control operation. (See PER, page 135)

NOTES :

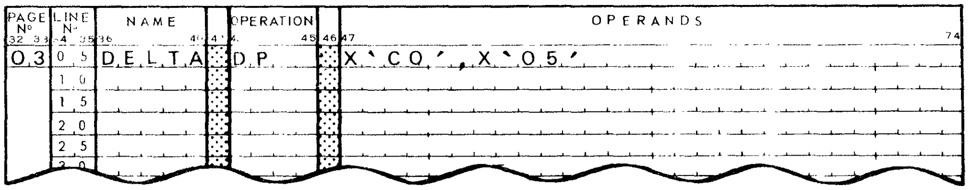
- The first octet specifies the mode operation and the channel to be used.
- The second octet specifies the condition to be tested.

PROGRAMMING PRACTICES :

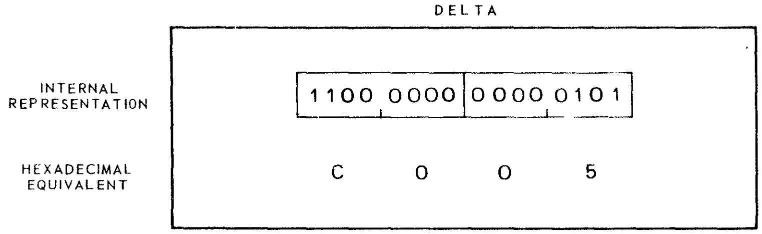
The hexadecimal configuration for the first octet is given in Figure B-8 , and the hexadecimal configuration for the second octet is given in Figure B-9 . Any of the forms for specifying an immediate operand may be used, but it is recommended that the hexadecimal configurations shown in the table be given in the standard hexadecimal specification format.

EXAMPLE

The assembler is instructed to set up for an instruction to test end of a printer page.



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
03	05	DELTA		DP		X'C0',X'05'
	10					
	15					
	20					
	25					
	30					



The assembler assigns a field of two octets to the name DELTA and stores in the field the internal representation of the defined peripheral constant.

DEFINE PERIPHERAL INSTRUCTION DP

Peripheral Unit Control

DELTA DP descriptor, operation

The assembler is directed to set up a two octet field for reference by a peripheral unit control operation. (See PER, page 135)

NOTES :

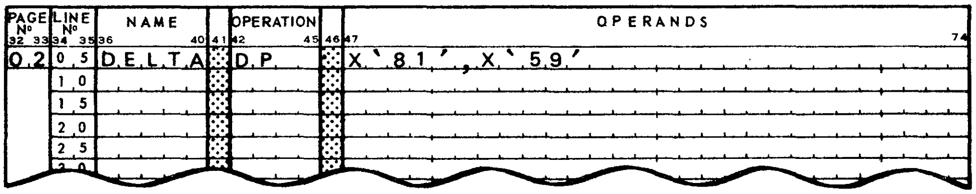
- The first octet specifies the mode of operation and the channel to be used.
- The second octet specifies the action to be taken.

PROGRAMMING PRACTICES :

The hexadecimal configuration for the first octet is given in Figure B-10, and the hexadecimal configuration for the second octet is given in Figure B-11. Any of the forms for specifying an immediate operand may be used, but it is recommended that the hexadecimal configurations shown in the table be given in the standard hexadecimal specification format.

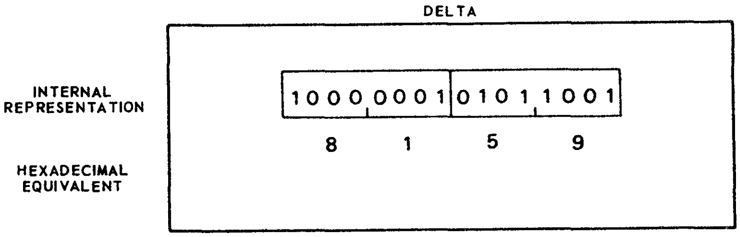
EXAMPLE :

The assembler is instructed to set up a field for an instruction to double space on a print page.



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
0 2	0 5	DELTA		DP		X'81',X'59'
	1 0					
	1 5					
	2 0					
	2 5					

The assembler assigns a field of two octets to the name DELTA and stores in the field the internal representation of the defined peripheral constant.



The assembler assigns a field of two octets to the name DELTA and stores in the field the internal representation of the defined peripheral constant.

THE ASSEMBLER PROGRAM CONTROL INSTRUCTIONS

The GE-115 Assembler accepts directives for the placement of the assembled program and for the specification of the first instruction to be executed. The assembler program control statements make it possible to specify some of the location assignments for the assembled programs as well.

Assembler program control instructions are processed as the source program is assembled and affect the assembled program. One of the control Directive operations, the Start Program Assembly (STRT) instruction, includes the specification of the peripheral controller to be used by the assembler in preparing output cards.

The assembler program control instructions have the same general format as the other Directive instructions and the Primary instructions. The name field is not used because the Directive instructions are operative only at assembly time and may not be cross-referenced by assembled program instructions at execution time.

One of the formats of the control instructions, shown below, introduces a new notation in the operand field as follows:

s	used to indicate that a one-digit numeric code specifying the size of the store must be written,
cc	used to indicate that a numeric two-digit specification for an input/output controller must be written.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
07	05			op		ALPHA
	10					
	15					
	20					
	25					

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
03	05			STRT		dddd,s,cc
	10					
	15					
	20					
	25					

START PROGRAM ASSEMBLY STRT

STRT dddd,s,cc

The assembler is directed to start assembling the program at the store location specified by dddd. The program is assembled for a store of the size indicated by s. The cc value specifies the peripheral controller used by the assembler to punch cards produced during assembly.

NOTES:

- The first octet in the store used by the assembler for assigning locations to the program is the octet specified by the value of dddd.
- The store size available for the assembled program is assumed to be that indicated by the code value s.
- The punch instruction used by the assembler is executed through controller cc.

PROGRAMMING PRACTICES:

The STRT card must be the first card in the program being assembled. An error halt occurs when any other card is read as the first card of a source program.

The address for program assignment must be expressed in four decimal digits and may not specify a value below the limit of the store area required by the system loader and subroutines, as shown below:

SYSTEM	LOADER LIMIT
Punched Cards Only	0448
Paper Tape	0512

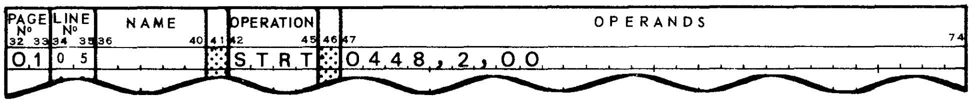
The value of s indicates a store size by a code. The values for the store sizes used are shown below:

STORE SIZE	s
4096	1
8192	2

The connectors that may be used for punch attachment and the code values are shown below:

CONNECTOR	cc
3	00
4	64

EXAMPLE:



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			STRT		0448,2,00

The assembler is directed to assemble a program for a store size of 8192 octets. Assembly begins at location 0448. The punch unit is specified as attached to connector 3.

END OF PROGRAM END

END SIGMA

The assembler is given the indication that the last card of the source program has been read. When the assembled program is loaded at execution time, the END card causes the instruction specified by SIGMA to be the first instruction executed.

NOTES:

- The END card must be present. When no END card is present, the assembler attempts to read cards seeking the END card. An end-of-file condition occurs on the card reader.
- SIGMA may be a symbolic or an actual address.

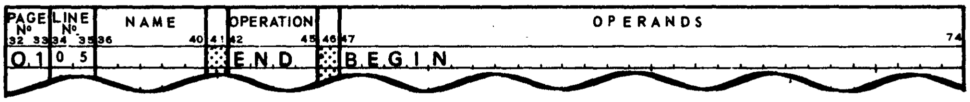
PROGRAMMING PRACTICES:

The END card terminates source program reading by the assembler, wherever it appears.

The SIGMA field operand specification must refer to the first instruction to be executed in the assembled program.
The instruction referenced need not be physically the first executable instruction in the program. It is logically the first instruction, i.e., the first operation to be performed.

EXAMPLE:

The instruction named BEGIN is the first instruction to be executed in the assembled program.



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			END		BEGIN

ORIGIN ASSIGNMENT ORG

```
ORG  dddd
     * +nnn
     * -nnn
     R
```

The assembler is directed to use the value specified in the operand field of the ORG as a store assignment value.

NOTES:

- The assembler program maintains a location counter for store assignment. Store addresses are assigned sequentially. When the ORG is encountered, the assembler resets the store assignment counter to the value specified by the ORG instruction.
- R as an operand causes the assembler to reset the store assignment counter to the next higher octet location which is a multiple of 256 octets. When the R operand is encountered by the assembler at a point at which the store assignment counter contains a value that is a multiple of 256 octets, no resetting takes place.
- The portion of the program following the ORG is assigned store locations sequentially from the specified octet unless another ORG is encountered.

PROGRAMMING PRACTICES:

An ORG with an absolute address operand may be used to define data fields at desired points in the store.

The ORG with the operand R is used to set up fields for use with the Translate (TR) instruction and for input/output on channel 2.

The ORG with the asterisk and an increment or decrement may be used to modify the store assignment counter with respect to its current value.

The absolute address assignment allows for defining different data areas at the same fixed store address. Different names can be used with Define Store Area instructions at the same actual address. However, the use of absolute values is not recommended. The DS instruction with a zero duplication factor can be used without the ORG instruction to effect the assignment of several fields to the same area (See DS, page 143).

EXAMPLES:

1) The assembler is directed to reset the value in the store assignment counter to 0512.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			ORG		0512
	10					
	15					
	20					
	25					

2) The assembler is directed to reset the value in the store assignment counter to the next higher multiple of 256.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
02	05			ORG		R
	10					
	15					
	20					
	25					

3) The assembler is directed to advance the value in the store assignment counter by 126.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
03	05			ORG		*+126
	10					
	15					
	20					
	25					

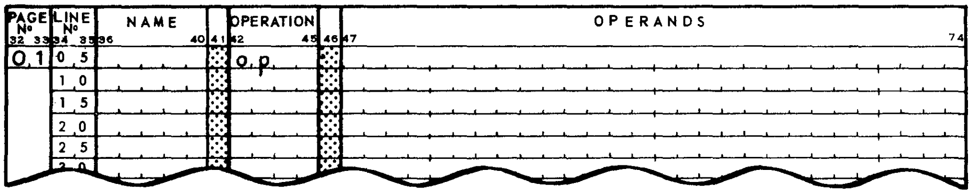
ASSEMBLY LISTING FORMAT INSTRUCTIONS

The GE-115 Assembler allows for control of the format of the program listing. The listing format statements provide a means for the programmer to specify both spacing on a page and the points at which a new page should begin. Text commentary is accepted for insertion into the listing of the program.

The assembly listing format instructions are operative only at the time the listing is printed. They allow for improved readability through formatting. Comments should be freely used as aids to documentation.

Assembly listing format instructions are written in the same general format as the other directive instructions and the primary instructions. The use of a name field in an assembly listing format instruction is meaningless because the instructions are not present in the translated program at execution time.

The format of the assembly listing format instructions is shown below :



COMMENT

*

* Text

The asterisk (*) directs the assembler to print the text in the operand specification field.

NOTE :

- The assembler inserts the text into the program listing. The card sequence determines the position of the comment. The assembled program is not affected by the comment instruction.

PROGRAMMING PRACTICES :

Column 46 must be blank. Any comment used must not begin to the left of column 47.

Comments should be used to head program sections and to describe the process performed by each.

The text field of the Comment instruction may contain any of the print characters. Blanks may be used to improve readability.

The comment card may be used to continue a comment which begins in the operand field of an instruction. This should be done to avoid the use of cryptic comments on instructions. Comments are an important form of documentation.

EXAMPLE :

The comments will be printed within the assembly listing.

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			*		
	10			*		COMMENTS MAY BE INSERTED AT
	15			*		
	20			*		ANY POINT IN A PROGRAM
	25			*		

EJECT PRESENT PAGE EJEC

EJEC

The assembler is directed to advance the paper on which the listing is being printed. An advance to the top of the next page is requested.

NOTE:

- The present print page is advanced. Printing continues at the top of the next page.

PROGRAMMING PRACTICE:

The EJEC is used to improve the readability and format of the assembly listing. Logically separate routines should begin on new page.

EXAMPLE:

PAGE No 32 33	LINE No 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			EJEC		
	10			*		TOP OF PAGE 2
	15					
	20					
	25					

The assembler listing continues at the top of the following page.

LINE FEED LF

```
LF    S
      D
      1
      2
      7
```

The assembler is directed to advance the paper on which the listing is being printed. The operand symbol specifies the spacing requested.

NOTES :

- The next line printed is spaced according to the LF request.
- Standard spacing continues after that line is printed.

PROGRAMMING PRACTICES :

The LF is used to improve listing readability.

Spacing that may be requested is shown below :

S	Skip one line
D	Skip two lines
1	Spacing as indicated by channel 1 control tape punch
2	Spacing as indicated by channel 2 control tape punch
7	Spacing as indicated by channel 7 control tape punch

EXAMPLE :

The assembler listing page is advanced two lines before the comment is printed.

PAGE N° 32 33	LINE N° 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			LF		D
	10			*		COMMENT

SECTION C

GE-115

ARITHMETIC SUBROUTINES

GE-115 SUBROUTINES

A subroutine is an independent sequence of programmed instructions which performs a standard information processing task for a main program. A subroutine program is designed to function in a manner that is independent of the program which utilizes the process. Data is supplied according to the subroutine requirements. The results of the subroutine operation are placed in a pre-defined area for use by the main program. A subroutine may be utilized repeatedly. All necessary resetting of store areas used is done within the subroutine itself.

A subroutine must provide an entry point for use by the main program, referred to as the calling program. This allows the calling program to jump to the subroutine. A subroutine which performs a number of related functions, such as addition and subtraction, may have more than one entry point to permit direct reference to each of the functions.

There must be, as well, a mechanism by which control can be returned to the calling program. In the GE-115 , a single instruction is used to effect entry and to prepare for return. The Jump and Return (JRT) instruction makes the entry to the subroutine and places the location of the next sequential operation after the JRT instruction in LOC (store octets 0254 - 0255).

The flexibility of the GE-115 assembler allows the programmer to use a meaningful mnemonic in the place of the JRT when writing subroutine calls. The mnemonic SUB is used.

Each subroutine uses the contents of LOC to return to the operation following the entry.

Data to be acted upon by the subroutines is placed in standard pre-defined areas in the store. The locations used are described in the individual subroutine descriptions.

The general format of the entry to a subroutine is:

PAGE No 32 33	LINE No 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			SUB		SIGMA
	10					
	15					
	20					
	25					

SIGMA is the name of the subroutine being used.

Each of the myriad applications that may make use of information processing systems has its own set of standard procedures. Any standard procedure may be programmed as a subroutine. One area of general application, which may be included in more specific tasks, is that of arithmetic calculation.

This section discusses four arithmetic subroutines prepared for use with the GE-115 system.

The arithmetic subroutines described in this section process decimal data. Data is treated in one of two ways : as unsigned values or as signed numeric quantities.

Data which is treated as signed data must be placed in the input area with an associated sign. It should be noted that signs used by the subroutines do not correspond to the internal configurations for the graphic characters + and -. Signs are recognized in the subroutines according to the configuration of the left quartet of the applicable sign octet. The negative sign is an A (1010); the positive sign is a 4 (0100). The manner in which the signs are treated is described in the discussions of the individual subroutines.

Subroutines make reference to pre-defined areas in which data is expected to be placed. It is recommended that the areas used by a subroutine be defined and given meaningful names by the calling program. This can be accomplished by using the DS instruction to associate names with the locations used by the subroutines. Any fields used, of course, must be defined in the octets used by the subroutines. The ORG Directive instruction with an absolute address can be used to place the data areas in the store octets referenced by the subroutines.

This might be done for the addition subroutine , YADS, for example by means of the sequence of instructions shown below:

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
06	05			ORG		0204
	10	ADND		DS		001L016 ADDEND
	15			ORG		0233
	20	ADSIN		DS		001L001 SIGN OF ADDEND
	25	AUGSN		DS		001L001 SIGN OF AUGEND
	30			ORG		0236
	35	AUGND		DS		001L016 AUGEND
	40			*		THE SUM REPLACES THE ADDEND
	45			ORG		0204
	50	SUM		DS		001L016
	55			ORG		0233
	60	SUMSN		DS		001L001

A similar sequence could be used to position data for use with any of the other subroutines. Several subroutines use the same areas for input and output. Therefore, it might be convenient to use common names such as TERM 1, TERM 2, and RESLT, to avoid individual definitions like SUM, DIFF, PROD, etc., for the same area.

It should be noted that the use of an absolute origin within the system software area causes the assembler to print an L on the assembler listing. The mistake indication does not prevent assembly.

SUB YADS

YADS forms the signed sum of two signed quantities. Prior to the subroutine call, the addend must be placed in store octets 0204 through 0219 (16 octets) and the augend must be placed in store octets 0236 through 0251 (16 octets). The sign of the addend must be in octet 0233; the sign of the augend must be in octet 0234. The sum replaces the addend.

INDICATORS AFFECTED

UF/OF	ZE/NZ	
-	0	The sum is zero.
-	1	The sum is non-zero.

NOTES:

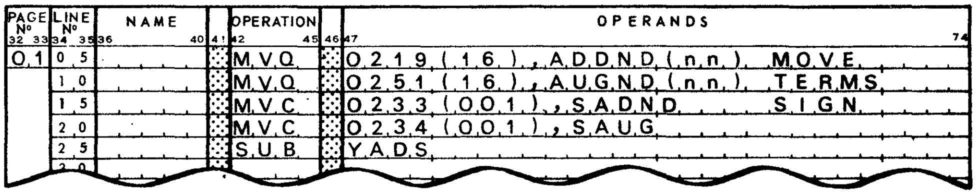
- The addend, augend and sum are each assumed to be 16 octets in length.
- YADS may use either the AD or the SD operation to generate the sum. YADS and YSDS are entries to a single subroutine.
- Signs are examined prior to the operation. The right quartets of the sign octets are made zero before the signs are checked. Each sign octet is then checked against a value of A0 (the negative sign configuration). If a sign is not A0, it is assumed positive. The subroutine sets any sign octet that does not contain A0 to a value of 40 (the positive sign configuration).
- The status of the UF/OF indicator depends on the signs of the terms and their relative magnitudes and does not necessarily reflect the result of the addition.

  If the terms have the same sign, UF/OF = 0 indicates no overflow, and UF/0F = 1 indicates overflow.

  If the terms have different signs, UF/OF = 0 indicates that the augend is greater in absolute value than the addend, and UF/OF = 1 indicates that the augend is smaller than or equal to the addend in absolute value.
- The addend is always replaced by the sum. If the signs of the terms are different and the augend is greater in absolute value than the addend, the sum is generated in the augend field and moved to the addend field prior to return.
- Quantities are assumed to be decimal. No check is made of the right quartets of the terms to be added.

PROGRAMMING PRACTICE:

It is recommended that standard, named fields be defined in the manner described in the introduction to this section. An alternate method for setting up the quantities for processing by the subroutine is shown below:



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			MVQ		0219(16),ADDND(nn) MOVE
	10			MVQ		0251(16),AUGND(nn) TERMS
	15			MVC		0233(001),SADND SIGN
	20			MVC		0234(001),SAUG
	25			SUB		YADS

The order of the move operations shown above is immaterial. The length of the moves is shown as the maximum field length to ensure that right quartet zeros are inserted in the left of the addend and augend fields whenever the terms used do not occupy the full field allowed. The subroutine always treats a pair of 16 octet fields.

PROGRAMMING PRACTICE:

It is recommended that standard, named fields be defined in the manner described in the introduction to this section. An alternate method for setting up the quantities for processing by the subroutine is shown below:

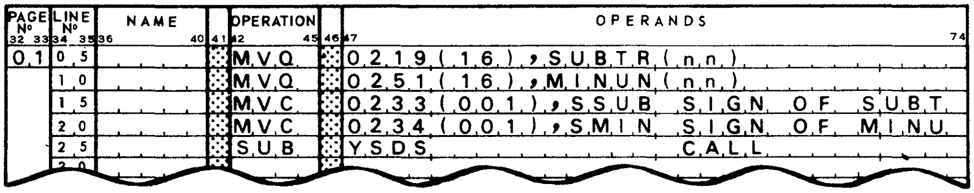
```
PAGE  LINE                                          
 No    No    NAME       OPERATION       OPERANDS
32 33 34 35 36     40 41 42     45 46 47                                74
 01    05                  MVO          0219(16),ADDND(nn)    MOVE
       10                  MVO          0251(16),AUGND(nn)    TERMS
       15                  MVC          0233(001),SADND       SIGN
       20                  MVC          0234(001),SAUG
       25                  SUB          YADS
```

The order of the move operations shown above is immaterial. The length of the moves is shown as the maximum field length to ensure that right quartet zeros are inserted in the left of the addend and augend fields whenever the terms used do not occupy the full field allowed. The subroutine always treats a pair of 16 octet fields.

- The subtrahend is always replaced by the difference. If the signs are the same and the minuend is greater than the subtrahend in absolute value, the difference is generated in the minuend field and moved to the subtrahend field.
- Quantities are assumed to be decimal in the right quartets. No check is made of the right quartets of the fields which are processed.

PROGRAMMING PRACTICE:

It is recommended that standard, named fields be defined in the manner described in the introduction to this section. An alternate method for setting up the quantities for processing by the subroutine is shown below:



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			MVQ		0219(16),SUBTR(nn)
	10			MVQ		0251(16),MINUN(nn)
	15			MVC		0233(001),SSUB SIGN OF SUBT
	20			MVC		0234(001),SMIN SIGN OF MINU
	25			SUB		YSDS CALL

The order of the move operations shown above is immaterial. The length of the moves is shown as the maximum field length to ensure that right quartet zeros are inserted in the left of the subtrahend and minuend fields whenever the terms used do not occupy the full field allowed. The subroutine always treats a pair of 16 octet fields.

SUB YMULF

YMULF forms the unsigned product of two unsigned quantities. Prior to the subroutine call, the multiplier must be placed in store octets 0204 through 0219 (16 octets), and the multiplicand in store octets 0238 through 0251 (14 octets). A field of fewer than 16 octets may be specified for the multiplier. An asterisk inserted to the left of the most significant digit in the multiplier field acts as a field delimiter. The product is formed in store octets 0205 through 0234 (30 octets).

INDICATORS AFFECTED

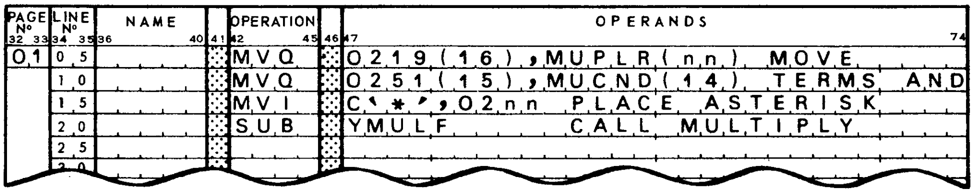
UF/OF	ZE/NZ	
1	0	Always set

NOTES :

- The multiplier is assumed to be 16 octets in length , unless an asterisk is present in the multiplier field. All octets to the right of the asterisk are treated as part of the multiplier field.
- The multiplicand is assumed to be 14 octets in length. A zero right quartet must be placed in the octet to the left, store location 0237.
- The product is placed partially in the octets which contained the multiplier ; the multiplicand is unaffected by the subroutine operation.
- Multiplication is performed by use of the AD operation.
- The UF/OF and ZE/NZ indicators do not reflect the result of the operation.
- The terms which enter into the multiplication are assumed to be decimal in the right quartet. No check is made.

PROGRAMMING PRACTICES :

It is recommended that standard, named fields be defined in the manner described in the introduction to this section. An alternate method for setting up the quantities for processing by the subroutine is shown below :



PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			MVQ		0219(16),MUPLR(nn) MOVE
	10			MVQ		0251(15),MUCND(14) TERMS AND
	15			MVI		C'*',02nn PLACE ASTERISK
	20			SUB		YMULF CALL MULTIPLY
	25					

The length of the fields should be given as 16 for the multiplier, when no asterisk is used, as well, and 15 for the multiplicand to ensure that high order left quartet zeros are inserted. The subroutine treats a field of 16 octets for the multiplier, unless the asterisk is present, and a 14 octet multiplicand field. The high order zero quartet in the 15 octet multiplicand field is assumed zero.

SUB YDIVF

YDIVF forms the unsigned quotient of two unsigned quantities. Prior to the subroutine call, the dividend must be placed in store octets 0204 through 0219 (16 octets) and the divisor must be placed in store octets 0238 through 0251 (14 octets). A field of fewer than 16 octets may be specified for the dividend. An asterisk inserted to the left of the most significant digit in the dividend field acts as a field delimiter. The quotient is formed in store octets 0219 through 0234 (16 octets). The remainder is left in store octets 0238 through 0251 (14 octets).

INDICATORS AFFECTED

UF/OF	ZE/NZ	
0	0	The quotient is zero.
0	1	The quotient is non-zero.

NOTES:

- The dividend is assumed to be 16 octets in length, unless an asterisk is present in the dividend field. All octets to the right of the asterisk are treated as part of the dividend field.
- The divisor is assumed to be 14 octets in length. A zero right quartet must be placed in the octet to the left of the most significant digit in the divisor, store location 0237.
- The quotient is placed partially in the octets which contained the dividend; the divisor is replaced by the remainder.
- Division is performed by means of the SD operation.
- The UF/OF indicator always contains a zero at the end of the subroutine operation.
- A 0 in the ZE/NZ indicator at the exit from the subroutine indicates that the quotient is zero; a 1 in the ZE/NZ indicator at the exit from the subroutine indicates that the quotient is non-zero.
- The terms which enter into the division are assumed to be decimal in the right quartet. No check is made.
- Division by zero causes an endless series of subtractions. No check is made of the divisor before the subtraction is attempted.

PROGRAMMING PRACTICES:

It is recommended that standard, named fields be defined in the manner described in the introduction to this section. An alternate method for setting up the quantities for processing by the subroutine is shown below:

PAGE Nº 32 33	LINE Nº 34 35	NAME 36 40	41	OPERATION 42 45	46	OPERANDS 47 74
01	05			MVQ		0219(16),DIVD(nn)
	10			MVQ		0251(15),DIVR(14)
	15			MVI		C'*',Q2nn
	20			SUB		YDIVF
	25					

The dividend field is assumed to be fewer than 16 octets in length. The length of the move is written as 16 octets to ensure that a high order right quartet zero is inserted. The divisor, 14 digits in length, is moved into a 15-octet field to place a high order right quartet zero. No asterisk is used when the dividend occupies the full field.

A check for a zero divisor may be performed by means of a conditional jump following the MVQ instruction used to position the divisor in the subroutine area.

SECTION D

APPENDICES

APPENDIX A

Figure 1 : TABLE OF CARD AND PRINTER CHARACTER REPRESENTATIONS IN THE GE-115 INFORMATION PROCESSING SYSTEM

CARD CODE	BINARY CODE	PRINTER CHARACTER	HEXADECIMAL	BINARY ORDER
0	01000000	0	40	1
1	01000001	1	41	2
2	01000010	2	42	3
3	01000011	3	43	4
4	01000100	4	44	5
5	01000101	5	45	6
6	01000110	6	46	7
7	01000111	7	47	8
8	01001000	8	48	9
9	01001001	9	49	10
2-8	01001010	[	4A	11
3-8	01001011	#	4B	12
4-8	01001100	@	4C	13
5-8	01001101	: (colon)	4D	14
6-8	01001110	>	4E	15
7-8	01001111	?	4F	16
	01010000	b̸ or _	50	17
12-1	01010001	A	51	18
12-2	01010010	B	52	19
12-3	01010011	C	53	20
12-4	01010100	D	54	21
12-5	01010101	E	55	22
12-6	01010110	F	56	23
12-7	01010111	G	57	24
12-8	01011000	H	58	25
12-9	01011001	I	59	26
12	01011010	&	5A	27
12-3-8	01011011	. (period)	5B	28
12-4-8	01011100	]	5C	29
12-5-8	01011101	(	5D	30
12-6-8	01011110	<	5E	31
12-7-8	01011111	\	5F	32

Figure 1 : TABLE OF CARD AND PRINTER CHARACTER REPRESENTATIONS IN THE GE-115 INFORMATION PROCESSING SYSTEM

CARD CODE	BINARY CODE	PRINTER CHARACTER	HEXADECIMAL	BINARY ORDER
11-0	1 0 1 0 0 0 0 0	↑	A 0	33
11-1	1 0 1 0 0 0 0 1	J	A 1	34
11-2	1 0 1 0 0 0 1 0	K	A 2	35
11-3	1 0 1 0 0 0 1 1	L	A 3	36
11-4	1 0 1 0 0 1 0 0	M	A 4	37
11-5	1 0 1 0 0 1 0 1	N	A 5	38
11-6	1 0 1 0 0 1 1 0	O	A 6	39
11-7	1 0 1 0 0 1 1 1	P	A 7	40
11-8	1 0 1 0 1 0 0 0	Q	A 8	41
11-9	1 0 1 0 1 0 0 1	R	A 9	42
11	1 0 1 0 1 0 1 0	-(minus or hyphen)	A A	43
11-3-8	1 0 1 0 1 0 1 1	$	A B	44
11-4-8	1 0 1 0 1 1 0 0	*	A C	45
11-5-8	1 0 1 0 1 1 0 1	)	A D	46
11-6-8	1 0 1 0 1 1 1 0	;	A E	47
11-7-8	1 0 1 0 1 1 1 1	'(apostrophe)	A F	48
12-0	1 0 1 1 0 0 0 0	+	B 0	49
0-1	1 0 1 1 0 0 0 1	/	B 1	50
0-2	1 0 1 1 0 0 1 0	S	B 2	51
0-3	1 0 1 1 0 0 1 1	T	B 3	52
0-4	1 0 1 1 0 1 0 0	U	B 4	53
0-5	1 0 1 1 0 1 0 1	V	B 5	54
0-6	1 0 1 1 0 1 1 0	W	B 6	55
0-7	1 0 1 1 0 1 1 1	X	B 7	56
0-8	1 0 1 1 1 0 0 0	Y	B 8	57
0-9	1 0 1 1 1 0 0 1	Z	B 9	58
0-2-8	1 0 1 1 1 0 1 0	←	B A	59
0-3-8	1 0 1 1 1 0 1 1	(comma)	B B	60
0-4-8	1 0 1 1 1 1 0 0	%	B C	61
0-5-8	1 0 1 1 1 1 0 1	=	B D	62
0-6-8	1 0 1 1 1 1 1 0	"	B E	63
0-7-8	1 0 1 1 1 1 1 1	!	B F	64

Figure 2: TABLE OF GE-115 OPERATIONS BY HEXADECIMAL REPRESENTATION

HEXADECIMAL REPRESENTATION		MNEMONIC EXPRESSION	SYSTEM ACTION
Operation Code	Operation Complement		
02	10	*ENS	Enable Single Stop
	20	*INS	Inhibit Single Stop
	80	*LON	Turn Alert Light On
↓	E0	*LOFF	Turn Alert Light Off
07	00	*NOP2	No Operation
0A	00	*HLT	Halt System Operation
41	F0	*JRT	Jump and Return
↓	↓	*SUB	Subroutine Call
43	00	*NOJ	No Jump
	10	*JG	Jump if Greater
	20	*JE	Jump if Equal
	30	*JGE	Jump if Greater or Equal
	40		
	50		
	60		
	70		Jump on
	80	JC	Condition
	90		
	A0		
	B0		
	C0	*JL	Jump if Less
	D0	*JNE	Jump if Not Equal
	E0	*JLE	Jump if Less or Equal
↓	F0	*JU	Jump Unconditional
53	40	JS2	Jump on Switch 2
	80	JS1	Jump on Switch 1
92	octet	MVI	Move Immediate to Store
95	↓	CMI	Compare Immediate to Store
9E	unit	PER	Call Peripheral
D2	one length	MVC	Move Complete Octets
D4		NC	And on Complete Octets
D5		CMC	Compare Complete Octets
D6		OC	Or on Complete Octets
D7		XC	Exclusive Or on Complete Octets
D8		UPK	Unpack Octets into Right Quartets
D9		SR	Search to the Right
DA		PK	Pack Right Quartets into Octets
DB		SL	Search to the Left
DC		TR	Translate
DE	↓	EDT	Edit
F8	two lengths	MVQ	Move Right Quartets
F9		CMQ	Compare Right Quartets
FA		AD	Add Decimal
FB		SD	Subtract Decimal
FE		AB	Add Binary
FF	↓	SB	Subtract Binary

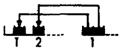
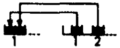
* Indicates a mnemonic expression which is translated into the operation code and the operation complement.

Figure 3 : TABLE OF GE-115 OPERATIONS BY MNEMONIC EXPRESSION

MNEMONIC EXPRESSION	ACTION	HEXADECIMAL REPRESENTATION*		Page
AB	Add Binary	FE		59
AD	Add Decimal	FA		51
CMC	Compare Complete Octets	D5		80
CMI	Compare Immediate to Store	95		78
CMQ	Compare Right Quartets	F9		83
ENS	Enable Single Stop	02	10	123
EDT	Edit	DE		109
HLT	Halt System Operation	0A	00	118
INS	Inhibit Single Stop	02	20	122
JC	Jump on Condition	43		97
JE	Jump if Equal		20	101
JG	Jump if Greater		10	101
JGE	Jump if Greater or Equal		30	101
JL	Jump if Less		C0	101
JLE	Jump if Less or Equal		E0	101
JNE	Jump if Not Equal		D0	101
JRT	Jump and Return	41	F0	107
JS1	Jump on Switch 1	53	80	105
JS2	Jump on Switch 2		40	105
JU	Jump Unconditional	43	F0	102
LOFF	Turn Alert Light Off	02	E0	121
LON	Turn Alert Light On	02	80	120
MVI	Move Immediate Octet	92		67
MVC	Move Complete Octets	D2		69
MVQ	Move Right Quartets	F8		71
NC	And on Complete Octets	D4		91
NOP2	No Operation	07	00	119
NOJ	No Jump	43	00	103
OC	Or on Complete Octets	D6		92
PER	Call Peripheral	9E		129
PK	Pack Right Quartets into Octets	DA		74
SB	Subtract Binary	FF		62
SD	Subtract Decimal	FB		54
SL	Search to the Left	DB		88
SR	Search to the Right	D9		85
SUB	Subroutine Call	41	F0	169
TR	Translate Octets	DC		114
UPK	Unpack Octets into Right Quartets	D8		76
XC	Exclusive Or on Complete Octets	D7		93

* The operation complement is given where it is translated from the mnemonic expression.

Figure 4 : GE-115 INSTRUCTION REFERENCE CHART

Instruct. Length	OPERATION CODES		OP COMP		FIELDS		INDICATORS		NOTES
	Symbol.	hexa.	Left	Right	Alpha	Beta	UF/OF	ZE/NZ	
6 OCTETS	AD	FA	Length - 1 of ALPHA (00 to 16)	Length - 1 of BETA (00 to 16)	A + B = ↩		0 no overflow 1 overflow	0 result = 0 1 result ≠ 0	AD, AB, SD, SB, MVQ, CMQ and SL are the only GE- 115 operations that process data from right to left. All the others process data from left to right. The operation whose mnemonics end by "D" and "Q" treat the right quartets only.
	AB	FE							
	SD	FB			A − B = ↩		0 underflow 1 true form	"	
	SB	FF							
	MVQ	F8			A ← B		0	"	
	CMQ	F9			A : B		0 $A < B$ 1 $A \geqslant B$	0 $A = B$ 1 $A \neq B$	
	XC	D7	Common Length - 1 of ALPHA and BETA (000 to 255)		A [+] B		1	0 result = 0 1 result ≠ 0	exclusive "or"
	OC	D6			A (+) B		Unchanged		logical "or"
	NC	D4			A [×] B		"		logical "and"
	MVC	D2			A ← B		"		The common length is the one defined in symbolic language for ALPHA.
	CMC	D5			A : B		0 $A < B$ 1 $A > B$	0 $A = B$ 1 $A \neq B$	
	UPK	D8	Length - 1 of BETA (000 to 255)		normal form	condensed	Unchanged		 2×(length of BETA) = number of unpacked quartets. Left quartets remain unchanged.
	PK	DA	Length - 1 of ALPHA (000 to 255)		condensed	normal form	"		 2×(length of ALPHA) = number of BETA octets.
	TR	DC			area to be translated	table origin	"		Table origin + ALPHA value = address of translated octet.
	SR	D9			area searched	character sought	1	0 not found 1 found	If match found: octet address + 1 − 1 If search failed: last searched address + 1 − 1 is stored in LOC.
	SL	DB							
	EDT	DE			mask + result	area edited	1	0 Z sup. 1 non Z sup.	See special chart in E D T.
4 OCTETS	MVI	92	Immediate octet		single octet		Unchanged		ALPHA octet replaced by immediate octet.
	CMI	95			"		0< A imm. 1> A imm.	0 A = imm. 1 A ≠ imm.	ALPHA octet compared to immediate octet.
	JC	43	Condition for Jump		Address jumped to		Unchanged		See special chart.
	JRT	41	F0				"		Stores return address in L O C and jumps to S I G M A.
	JS1 JS2	53	80 for sw. 1 40 for sw. 2				"		The operation complement defines the switch.
	PER	9E	peripheral unit number		Delta		See special chart		For data transfer the two leftmost octets must contain the data length - 1
2 OCTETS	HLT	0A	00				Unchanged		Brings the program to a halt. To continue press "START".
	NOP2	07	"				"		No operation.
	ENS	02	10				"		Allows a program halt by means of the "SINGLE-STOP" switch.
	INS	"	20				"		Disables the "SINGLE-STOP" switch.
	LON	"	80				"		Lights the "A L E R T" light on the console panel.
	LOFF	"	E0				"		Shuts the "A L E R T" light off on the console panel.

LOC 0254 0255

OPERATION		UF/OF 0 ZE/NZ 0	UF/OF 0 ZE/NZ 1	UF/OF 1 ZE/NZ 0	UF/OF 1 ZE/NZ 1
AD, SB *		ALPHA = 0 No Overflow	ALPHA ≠ 0 No Overflow	ALPHA = 0 With Overflow	ALPHA ≠ 0 With Overflow
SD, SB		▨	ALPHA < 0 In Underflow Form	ALPHA = 0	ALPHA > 0
CMI, CMC, CMQ *		▨	ALPHA < BETA or Char.	ALPHA = BETA or Char.	ALPHA > BETA or Char.
MVQ		ALPHA = 0	ALPHA ≠ 0	▨	▨
XC		▨	▨	ALPHA = 0	ALPHA ≠ 0
SL, SR		▨	▨	Character Not Found	Character Found
EDT		▨	▨	End in Zero-Suppression Mode	End in Non-Zero Suppression Mode
PER	Status Test	▨	▨	Condition Present	Condition Not Present
	Data Transfer	▨	▨	End on Length	End of Input File
	Control	▨	▨	Always	▨

* The operands are Treated As Unsigned

Figure 5 : INDICATOR SETTINGS

APPENDIX B

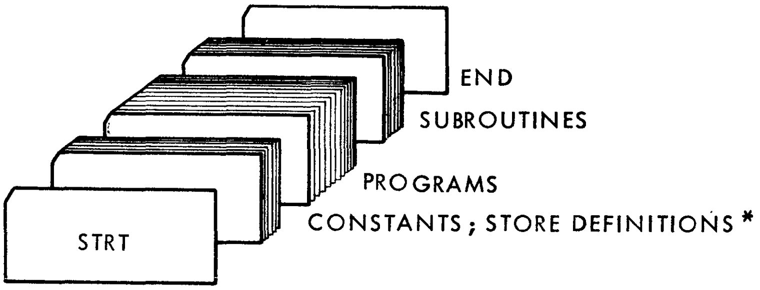
APPENDIX B

ASSEMBLING A PROGRAM

PROGRAMMER PREPARATION

Prior to submitting a program for assembly, the programmer should obtain a listing of the source cards. This listing should be checked against the "PROGRAMMER'S CHECK LIST" (See Figure 1). Corrections that are required should be noted on the listing and a corrected deck prepared.

When the source program cards are correct, the deck for assembly should be prepared as shown below:



* Shown as a block in the beginning of the deck to indicate that constant definitions should not appear within the program instruction sequence.

ASSEMBLER ACTION

The action of the assembler in translating the source program is divided into three parts:

PART 1 - Source card format scan and content verification, source program listing, control and allocation of addresses,

PART 2 - Source program translation,

PART 3 - Listing of the source program and translated formats, production of object program cards.

PART 1 carries out the following operations:

- Reads and verifies the format and contents of the source program cards.
- Prints the source program statements, followed by error indications, if required.
- Builds an address table for names occurring in the source program, and punches a table of names and locations.

PART 2 carries out program translation in one or more stages, depending on the size of the name table. The following two cases are differentiated:

Case 1

100 or fewer names (4096 octets of store)
600 or fewer names (8192 octets of store)

Case 2

More than 100 names (4096 octets of store)
More than 600 names (8192 octets of store)

In case 1, PART 2:

- Reads the name table and stores all of it. Repunches the name table.
- Reads the source program cards and translates them completely.
- Punches out cards containing the source program and the assembled program.

In case 2, PART 2:

- Reads the name table and stores 100 (or 600) elements of the name table.
- Repunches the name table, with an identifying flag on the cards for which information is placed in store.
- Reads the source program cards and translates all references to names for which information is retained in store.
- Punches out cards containing the source program and the partially translated program.
- Repeats the above operations until the source program is completely translated.

PART 3 carries out the following operations:

- Prints the listings of the source program and the translated format of the program.
- Punches out the object program.

AFTER ASSEMBLY

When an error-free assembly is obtained, the assembler listing becomes the primary documentation for the program. Coding sheets, source card lists and any assembler lists with mistake indications are no longer valuable. All notes and corrections should be made on the most recent assembler listing to keep program documentation current.

The source program deck should be kept current as well. Whenever a change or correction is noted, the source card should be prepared and inserted in the program deck. The same procedures, of listing and checking the source cards, should be followed for re-assembly as for a first assembly.

Figure 1 : PROGRAMMER'S CHECK LIST

1. Are the cards in the correct sequence?
2. Is the format of the STRT card correct?
3. Are any names repeated?
4. Does the first instruction to be executed have a name?
5. Does each name used as an operand field specification match a name used in a name field?
6. Are any names in the name field unused? Why?
7. Are operand specifications separated by commas?
8. Are lengths enclosed in parentheses?
9. Are lengths correctly specified according to the data fields which enter the operations?
10. Are increments and decrements to data field references correctly computed?
11. Does data format agree with the expected format for the instructions which process it?
12. Are definition statements entirely separate from the sequence of executable instructions? Are there jumps around any included data or store definitions?
13. Are there any internal code or system dependencies? Why?
14. Are logical sections of the program separated for checking?
15. Are there test output operations included?
16. Are required operator messages included? Are they clear? Is operator intervention really required?
17. Are all indicator tests properly placed?
18. Are input/output operations tested for error? end of file?
19. Are there sufficient comments?
20. Are all subroutines present?
21. Is the END card correct?

APPENDIX C

APPENDIX C

BINARY NOTATION

Digital computers store information in the form of on-or-off conditions of electronic devices such as vacuum tubes, transistors, or magnetic cores. The fact that each of these devices can record only two states or conditions naturally gives rise to binary notation for expressing the values. In binary notation two values (0 or 1) may be expressed by each digit, just as in decimal notation ten digit values (0 to 9) are possible.

Binary notation uses the base 2 just as standard decimal notation uses the base ten. That is, if 115 in decimal notation means

$$1 \times 10^2 + 1 \times 10^1 + 5 \times 10^0$$

the same value expressed in binary as 1110011 means

$$1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 .$$

Thus,

$$1110011_2 = 115_{10} .$$

The binary equivalents of the digits 0 to 9 using 4 binary digits are:

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Binary addition and subtraction are performed as shown below :

$$\begin{array}{r} 0 \\ +0 \\ \hline 0 \end{array} \quad \begin{array}{r} 0 \\ +1 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ +0 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ +1 \\ \hline 10 \end{array} \quad \begin{array}{r} 10 \\ +1 \\ \hline 11 \end{array} \quad \begin{array}{r} 11 \\ +1 \\ \hline 100 \end{array}$$

$$\begin{array}{r} 0 \\ -0 \\ \hline 0 \end{array} \quad \begin{array}{r} 0 \\ -1 \\ \hline -1 \end{array} \quad \begin{array}{r} 1 \\ -0 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ -1 \\ \hline 0 \end{array} \quad \begin{array}{r} 10 \\ -1 \\ \hline 1 \end{array} \quad \begin{array}{r} 11 \\ -1 \\ \hline 10 \end{array}$$

```
            0011  (=3)        101110  (=46)
          + 0011  (=3)      + 10100  (=20)
          ------------      ---------------
            0110  (=6)      1000010  (=66)
carries                                     carries

Borrows                     Borrows
            11010  (=26)              10110101  (=181)
          - 01100  (=12)            - 01101100  (=108)
          --------------            -----------------
            01110  (=14)              01001001  (= 73)
```

Decimal values are not always translated into pure binary when stored in or operated upon by a computer. Frequently, a fixed number of bits is used to express each decimal digit. If four bits are used for each digit, the value 115 can be represented as

	DIGIT 1	DIGIT 2	DIGIT 3
bit pattern	0001	0001	0101
decimal value	1	1	5

Four bits permit values from 0 to 15 to be expressed. While these 16 distinct values are more than enough to express the 10 decimal digits, they are not sufficient to give distinct representation to each alphabetic character. The GE-115 Information Processing System uses eight bits, which have 2^8 or 256 possible configurations, to represent the 10 digits, 26 letters, and other characters, as well as pure binary values from 0 to 255.

APPENDIX D

APPENDIX D

HEXADECIMAL-TO-DECIMAL CONVERSION CHART

The table in this appendix may be used for conversion of hexadecimal to decimal numbers, and vice versa, in the following ranges :

Hexadecimal	Decimal
000....FFF	0000....4095

For numbers outside these ranges, add hexadecimal 1000 or decimal 4096 to the table figures.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
01	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
02	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
03	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
04	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
05	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
06	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
07	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
08	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
09	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255
10	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
11	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
12	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
13	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
14	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
15	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
16	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
17	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
18	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
19	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
21	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
22	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
23	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
24	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
25	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
26	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
27	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
28	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
29	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
30	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
31	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
32	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
33	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
34	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
35	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
36	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
37	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
38	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
39	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023
40	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
41	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
42	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
43	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
44	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
45	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
46	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
47	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
48	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
49	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
50	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
51	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
52	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
53	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
54	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
55	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
56	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
57	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
58	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
59	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535
60	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
61	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
62	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
63	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
64	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
65	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
66	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
67	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
68	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
69	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791
70	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
71	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
72	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
73	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
74	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
75	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
76	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
77	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
78	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
79	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
80	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
81	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
82	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
83	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
84	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
85	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
86	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
87	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
88	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
89	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
90	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
91	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
92	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
93	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
94	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
95	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
96	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
97	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
98	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
99	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559
A0	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A1	2576	2577	2678	2679	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A2	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A3	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A4	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A5	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A6	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A7	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A8	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A9	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
B0	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B1	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B2	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B3	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B4	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B5	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B6	2912	2913	2914	2915	2916	2917	2818	2919	2920	2921	2922	2923	2924	2925	2926	2927
B7	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B8	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B9	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071
C0	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C1	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C2	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C3	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C4	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C5	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C6	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C7	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C8	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C9	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
D0	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D1	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D2	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D3	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D4	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D5	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D6	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D7	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D8	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D9	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E0	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E1	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E2	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E3	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E4	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E5	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E6	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E7	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E8	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E9	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F0	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F1	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F2	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F3	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F4	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F5	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F6	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F7	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F8	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F9	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

INDEX

INDEX

INDEX (contd.)

INDEX (contd.)

INDEX (contd.)

INDEX (contd.)

INDEX (contd.)

Progress Is Our Most Important Product

GENERAL ELECTRIC

INFORMATION SYSTEMS DIVISION

LITHO U.S.A.

841