3100
3200
3300
3500

COMPUTER SYSTEMS
COMPATIBLE COMPASS

REFERENCE MANUAL

CONTROL DATA
CORPORATION

Additional copies of this manual may be obtained
from the nearest Control Data Corporation Sales office.

CONTROL DATA CORPORATION
Documentation Department
3145 PORTER DRIVE
PALO ALTO, CALIFORNIA

April, 1967
Pub. No. 60174000,A

CONTENTS

INTRODUCTION

The Compatible Compass Language Reference Manual is designed for the programmer-user of the 3100, 3200, 3300 or 3500 computer, who chooses to use COMPASS assembler language.

With COMPASS, the Comprehensive Assembly System for Control Data computers, the programmer may conveniently use mnemonic instructions and symbolic addresses to write machine language programs.

The COMPASS assembly system:

Permits the use of location symbols as addresses.

Attaches character designators to word addresses.

Causes specified initial values to be loaded into data areas in the source program.

Allows the establishment of common areas to expedite communication among subprograms.

Recognizes integer, floating point, and BCD constants in familiar notation.

Facilitates calling system library routines.

Controls the format of the assembly listing with COMPASS pseudo-instructions.

Lists diagnostics for source program errors.

Enables macro instructions to be defined and used.

This Compatible Language Reference Manual is to be used in conjunction with the appropriate COMPASS Programming Guide. The guides provide operating system information such as job deck structures and diagnostic flags for assembling under specific operating systems, such as MSOS or MASTER.

COMPASS PROGRAMS 1

In COMPASS source language, the programmer writes machine language instructions in mnemonic and symbolic form, specific constants, controls subprogram communication and directs the assembly process with a powerful set of pseudo instructions.

Subprograms in COMPASS language are assembled, linked at load time, and executed as a single unit; one program usually consists of several subprograms. The size of subprograms or the magnitude of the problems solved by a subprogram is established by the programmer.

A subprogram consists of an IDENT pseudo instruction followed by subsequent lines of coding and finally an END pseudo instruction. Storage for assembled subprograms consists of three main areas:

Data area - one area for all subprograms of a single run

Subprogram area

Common area

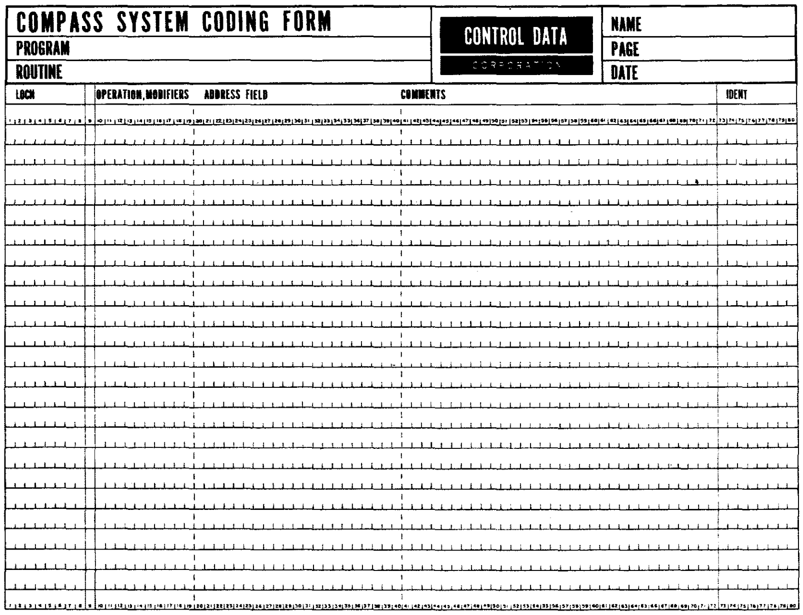
The three areas are defined at assembly time. When more than one subprogram is loaded for execution at run time, the total storage requirement must be considered. The data area must be sufficient to contain the total information assigned to it by all subprograms. The programmer must also insure that information stored in the data area by the loader will not conflict with information from other subprograms destined for the shared area.

COMPASS object code contains relocatable addresses which are modified by a relocation factor during loading to obtain the actual address in the computer memory. When assembling subprograms, COMPASS assumes that the initial location in each of the three areas - data, common and subprogram - has a relocatable address of zero. Locations are then assigned sequentially from zero unless the pseudo instruction ORGR is encountered. ORGR instructs COMPASS to assign the value in the address field of ORGR as the relocatable address of the following instruction and assign storage sequentially from that relocatable address.

Each area recognized by COMPASS has its own address counter. The address counter affected by ORGR is the counter currently in use. The address counter used by COMPASS for a given area is the same throughout the subprogram. A counter set by ORGR remains set until a subsequent ORGR. All counters are initialized before assembling a new subprogram.

1.1 INSTRUCTION FORMAT

Instructions to be assembled by COMPASS are written on coding forms and subsequently punched into cards or prepared on other media for input to COMPASS. Each line on the coding sheet is normally punched into a single card. The correspondence between columns on coding sheet and card is one-to-one.



COMPASS SYSTEM CODING FORM

PROGRAM

ROUTINE

CONTROL DATA CORPORATION

NAME

PAGE

DATE

LOCN	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS	IDENT

Each line of code has five fields; all instructions are defined in terms of the contents of these fields (Appendix A).

Field	Columns
Location	1-8 inclusive, 9 always blank.
Operation	Begins in column 10 and continues until the first blank column.
Address	It may begin after the blank terminating the operation field; it must begin before column 41; and terminates with the first blank, or column 73.
Comments	Remarks are written between the end of the address field and column 73.
Ident	Identification or sequence numbers in 73-80 are treated as a comment by COMPASS.

1.2 LOCATION FIELD

A location symbol placed anywhere in columns 1-8 specifies the address of an instruction or data item.

Location fields may be totally blank or contain symbols consisting of from 1 to 8 alphabetic or numeric characters or a period. The first character must be alphabetic. Imbedded blanks are illegal. An illegal symbol is flagged as an L error on the assembly listing.

The location field symbol may represent a 15- or 17-bit relocatable address or a 15- or 17-bit non-relocatable value. Symbols representing an address are defined under control of one of the three address counters (except in EQU); they reference the first word or character position occupied by the particular instruction.

When an asterisk appears in column 1, columns 2-72 are treated as a comment.

Examples of location field symbols:

Acceptable	Unacceptable
A123.456	12345678
H3	.2345678
ABCDEFGH	
P1234567	

1.3 OPERATION FIELD

The operation field may contain mnemonic machine instruction codes or pseudo instruction mnemonics, with specific, related modifiers, macro instruction names, the octal values 00-77, or the octal values 00-77 with the modifier C.

The field begins in column 10 and is terminated by the first blank. If column 10 is blank, an operation code of 00 is assembled. An illegal operation field is flagged as an O error on the assembly listing. Modifiers are separated from operation codes by commas; no blank columns may intervene.

Examples of acceptable operation fields:

```
BSS,C            INPC,INT,B,H
BSS              MACRO
LDA              74
                 74,C
```

1.4 ADDRESS FIELD

The address field begins anywhere before column 41 after the blank terminating the operation field and terminates with a blank or column 73. It is composed of one or more subfields, depending upon the instruction.

Machine instructions have implied subfields which may contain symbols, constants or expressions. A subfield may be assigned the value zero by giving only its trailing comma. The last address subfield may be assigned the value zero by omitting both its content and the preceding comma. But if the operation code specifies a BDP instruction, the preceding comma is required.

1.4.1 SYMBOLS

An address field symbol may occupy the entire field or it may be only one element in the field. Any symbol used in an address field must be defined by appearance in the location field of another instruction in the subprogram, or it must be declared as external. A symbol in the address field is formed and expressed exactly like a location symbol; it may be relocatable or non-relocatable.

A non-relocatable symbol is defined or equated to a value of 15 or 17 bits. The value assigned to the non-relocatable symbol will not be modified during loading.

A relocatable symbol represents either a 15- or 17-bit address. Relocatable addresses are values related to a memory area. These values will be incremented or decremented by the loader prior to storage of the instruction in which the address occurs. Relocatable symbols are local or external to a subprogram and are equated to a 15-bit word address or a 17-bit character address. Relocatable symbols may be:

subprogram relocatable	data relocatable
external symbols	common relocatable

The special character, *, may be placed in the address field and used as any symbol. The * is interpreted as the current value of the COMPASS address counter in effect when the * is encountered. The * may result in either a 15-bit or 17-bit address. If the machine instruction consumes two words, * is the address of the first word.

The special character ** may be used as the only entry in a field or subfield. The** yields a subfield containing a one in each bit position. Normally, the field represented by the ** will be modified during execution of the program and the double asterisk provides a convenient way to ascertain if the modification transpired.

1.4.2 CONSTANTS

The address field may contain signed or unsigned decimal or octal integers. If the sign is not present, the integer is assumed to be positive. Octal integers are suffixed by the character B.

1.4.3 EXPRESSIONS

In an address field or subfield, symbols, the special character *, and constants may be combined with the operators, plus or minus, to form an address expression. The value of the expression is calculated by substituting the numeric value of the symbol and performing 15- or 17-bit arithmetic with the designated operators. External symbols, the double asterisk, and literals may not appear in an address expression.

If relocatable symbols are part of an address expression, the result of the evaluated expression must be relocatable within a single area. Subprogram, data, or common relocatable symbols may be mixed:

$D_1-P_1+P_2-D_2+C_1-C_2$	non-relocatable value
$D-C_1+C_2$	positive data relocatable value
C_1-P-C_2	negative subprogram relocatable value

D_i = data relocatable addresses

P_i = subprogram relocatable addresses

C_i = common relocatable addresses

In an expression containing relocatable symbols, the algebraic sum of the relocation indicators must be either an area relocation increment or decrement, or no relocation designator and, therefore, a non-relocatable value.

The result of an address arithmetic symbol depends on the number of bits assigned to the subfield in the object code.

1.4.4 LITERALS

If the address field or subfield of an instruction refers to an operand which may be a single or double precision value, the entry may be a literal expressed as an equal sign followed by a mode designator and a value (=mv).

The equal sign denotes that the field contains a literal; m indicates the mode of the literal; v is the value of the literal. Single precision literals are expressed as above; double precision literals (48-bit) are expressed as =2mv.

The mode of a literal may be decimal, octal, Hollerith, or USACII.

Decimal literals: =Dv
The value of the decimal literal is expressed in the same manner as DEC and DECD pseudo instructions; they may be signed, cannot be more than 7 digits (14 for double precision), and may be followed by a scaling factor. A blank terminates the field. A BCD character is illegal.

Octal literals: =Ov
The value of the octal literal is written in the same manner as an OCT pseudo instruction; it may be signed, cannot be more than 8 digits (16 for double precision), and may be followed by a scaling factor. A blank terminates the field. A BCD character is illegal.

Hollerith literals: =Hv
The Hollerith literal is expressed as a string of 4 or 8 characters. The column following a Hollerith literal must contain a blank or a comma.

ASCII literals: =Iv
The ASCII literal is expressed as a string of 2 (4 for double precision) BCD characters. The characters are stored 2 per word as follows:

bits 23-20	zero
bits 19-12	first ASCII character
bits 11-08	zero
bits 07-00	second ASCII character

The column following an ASCII literal must contain a blank or a comma.

During assembly, a literal is converted to binary and assigned a relocatable address which is substituted for the literal in the object code. Literals are assigned to contiguous storage locations at the end of the subprogram. Literals of the same value and size are not duplicated in the object subprogram. Each time COMPASS encounters a literal, the value is compared against all previously assembled literals; and if an identical value exists, the address of the previously assigned literal is substituted in the object code.

1.4.5 EVALUATION OF EXPRESSIONS

Address expressions are evaluated as a word address (15 bits) or a character address (17 bits). All address expressions are converted to binary numbers of modulus $2^{15}-1$ or $2^{17}-1$, and stored in the proper subfield. No size check is made for 15- or 17-bit subfields by COMPASS.

The location terms of all instructions except BCD, C, BSS, C and EQU, C are evaluated as word addresses.

1.4.6 NON-RELOCATABLE SYMBOLS

Symbols defined as non-relocatable values are treated as integers. If the most significant bit of a non-relocatable value is one, the integer is assumed to be in complement form. A 17-bit non-relocatable value placed in an m, n or y subfield is reduced to modulo 2^{15}-1.

1.4.7 INTERCHANGE WORD/ CHARACTER ADDRESSES

A word address may be placed in a character address field or vice-versa. If a symbol defined as a word address is placed in a subfield which consists of 17 bits, the assigned binary value is shifted left two places.

If a symbol defined as a character address is placed in a subfield which has only 15 bits, the 17-bit character is shifted right two places; if a one bit is lost by the shift, a T error occurs.

1.5 COMMENTS AND IDENTIFICATION

Comments may be included with any instructions. A blank column must separate them from the last character in the address field and they may extend to column 73. Comments have no effect upon compilation, but will be included on the assembly listing.

Columns 73-80 may be used for program identification or for sequence numbers. This field has no effect upon assembly; if an asterisk is placed in column 1, the entire line will be considered a comment.

PSEUDO INSTRUCTIONS 2

COMPASS pseudo instructions control assembly process, convert constants, and reserve and assign storage.

2.1 SUBPROGRAM CONTROL

Three pseudo instructions define a subprogram and provide control information for COMPASS.

2.1.1 IDENT

LOCATION		OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8		10	20	41
		IDENT	m	

The address field contains the subprogram name; it may include as many characters as will fit into the field, but only the first 8 are used. They appear in the IDC card of the relocatable object subprogram deck and are printed as the title on the output listing unless a TITLE listing control pseudo instruction intervenes. The pseudo instruction, IDENT, will not appear on the output listing. The location field should be blank; it will be ignored by COMPASS.

The subprogram name is not an entry point name and cannot be referenced in the source subprogram. IDENT must be the first instruction of a subprogram; otherwise, the job is terminated. If it also appears elsewhere in the subprogram, an O error is indicated.

Instructions following IDENT are assembled using the subprogram address counter until the pseudo instructions DATA or COMMON intervene.

2.1.2 END

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
	END		

The final instruction in a COMPASS subprogram must be END. It terminates the subprogram and produces a TRA card in the relocatable object subprogram deck. The location field is ignored by COMPASS and should be blank.

A symbol in the address field is output to the TRA card as the symbolic transfer address. If a program is to receive control at the address indicated, the transfer address must be defined as an entry point.

2.1.3 FINIS

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
	FINIS		

FINIS signals that all subprograms have been submitted for assembly; it is the final instruction of a COMPASS input deck. Location and address fields are ignored. Normally, FINIS immediately follows an END pseudo instruction.

COMPASS will recognize FINIS at any point, however, and proceed as if END had occurred. If END is missing, the job is terminated when control is returned to the operating system.

2.2 PROGRAM STORAGE AREAS

The programmer may establish two storage areas to be shared by several subprograms. The common area may be shared for information which is processed by the running program, or accumulated during the course of execution. Information may not be assembled in the common area. At the source language level the programmer may label, reserve, or otherwise organize the common area but nothing more.

Information assembled for storage into the data area may consist of constants, message formats, masks, and other information to be used by more than one subprogram. Both the common and data areas are shared by all subprograms during execution. The significant difference is that the data area can be prestored or loaded by the loader; common cannot.

During assembly, COMPASS initially uses the subprogram address counter. When the pseudo instructions DATA or COMMON are encountered, COMPASS assembles subsequent information for the indicated area until another area assignment occurs. The pseudo instruction, PRG, returns control to the subprogram address counter.

If any statement which results in binary output occurs while COMMON is in effect, an error indication is given and assembly continues as if PRG had occurred in the source subprogram. Any one of the three location counters may be set by the pseudo instruction ORGR. When COMPASS initiates a different area address counter, or the counter currently in effect is reset by an ORGR, or is incremented by a BSS pseudo instruction, the current RIF card is produced.

2.2.1 PRG

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
	PRG		

PRG establishes the subprogram area location counter during assembly. PRG specifies that all instructions which follow are to be assembled in a subprogram area; it restores the subprogram location counter for use by COMPASS after an area of another type has been defined. When IDENT is encountered, the subprogram counter is initialized and remains in effect until DATA or COMMON occurs. The location and address fields are ignored by COMPASS.

2.2.2 DATA

LOCATION		OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8		10	20	41
		DATA		

DATA specifies that all subsequent information is to be stored or identified as part of the data area; it indicates use of the data area location counter. The location and address fields are ignored and should be blank.

Any instruction or pseudo instruction may follow DATA, providing no reference is made to an external name and no location within the data area is declared an entry point to the subprogram. Once DATA occurs in a subprogram, the data area location counter is used for assembly to the end of the subprogram unless PRG or COMMON occur.

2.2.3 COMMON

LOCATION		OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8		10	20	41
		COMMON		

COMMON organizes, labels, and reserves space in the common area. The location and address fields are ignored and should be blank.

Information may not be assembled for storage in the common area; therefore, the only instructions which may follow COMMON are BSS, BSS,C, COMMON, EQU, EXT, ENTRY, ORGR, IFT, IFN, IFF, IFZ, listing control instructions, and PRG, DATA or END. If any other instruction is encountered, an error is flagged and assembly continues as if PRG had been encountered.

2.2.4 ORGR

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
	ORGR	m	

ORGR specifies the relocatable address for storage of instructions, constants, or reservation of space in any of the three storage areas. The location field of ORGR is ignored by COMPASS but is printed on the output listing.

The address field may contain an expression which results in a value for a relocatable address. Symbols must have been defined in the location field of a preceding instruction and, if relocatable, be assigned to the same area as the address counter currently in effect.

The incorrect sequence in the following example demonstrates how symbols must be controlled by the subprogram address counter in the area in which they were assigned.

```
         IDENT       SAM
           .
           .
           .
         DATA
         BSS         2
MAC1     OCT         63
           .
           .
           .
         PRG
         ORGR        MAC1+16
           .
           .
           .
```

Since MAC1 was assigned in the data area, MAC1+16 could not be under control of the PRG subprogram address counter.

If COMPASS is assembling into one area and an ORGR occurs with a different area relocatable symbol in the address field, an error results. All address counters remain unchanged, and COMPASS ignores the ORGR. The error flag is included on the output listing.

2.3 WORD/CHARACTER STORAGE

Full words or character positions may be reserved and labeled with the pseudo instructions BSS or BSS,C. Reservation is made in the area governed by the current address counter. The address field determines how many words or character positions are to be reserved.

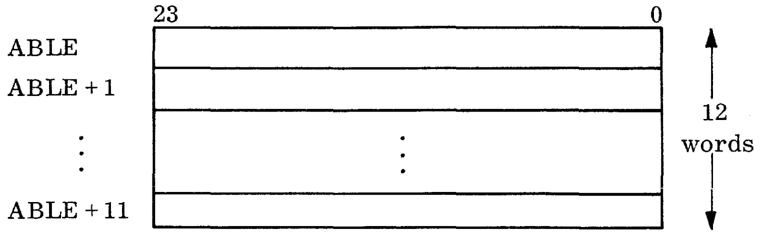
2.3.1 BSS

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
symbol or blank	BSS	m	

BSS reserves and labels a block of words in any area. The location field may be blank or contain a symbol which is defined as the 15-bit relocatable word address of the first word in the block to be reserved by BSS.

The address field, which specifies the number of words to be reserved, must contain a constant, a symbol, or an address expression which results in a non-relocatable value.

Example: ABLE BSS 12



The double asterisk in the address field is illegal; symbols in an address field must be defined in the location field of a preceding instruction.

A negative address field such as: BSS -2B
will be interpreted by COMPASS as: BSS 77775B
and 77775B words will be reserved

If the address field is in error or is zero, no storage will be reserved but a symbol in the location field will have been defined. If the address field contains zero, and the instruction is immediately preceded by BCD,C or BSS,C, the next instruction which consumes space will be forced to a new word.

Examples:

ALPHA	BCD,C	3,ABC	ALPHA	A	B	C	
	BSS	0					
	BCD,C	3,GHI	ALPHA+1	G	H	I	
ALPHA	BCD,C	3,ABC	ALPHA	A	B	C	G
	BCD,C	3,GHI	ALPHA+1	H	I		

2.3.2 BSS, C

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
symbol or blank	BSS,C	m	

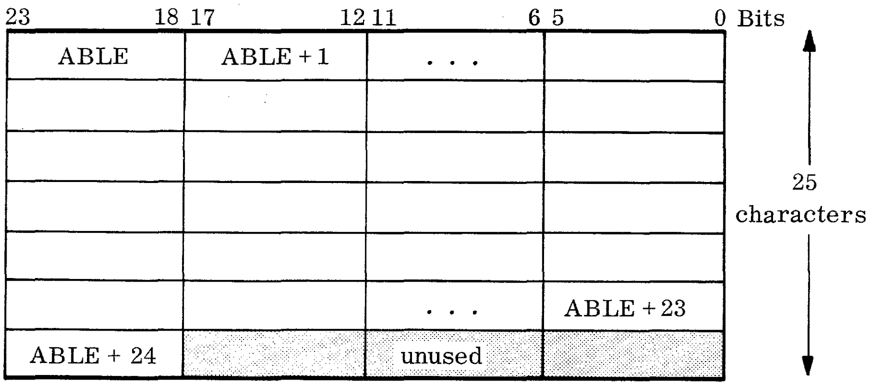
BSS,C reserves and labels a block of character positions. The location field may be blank or contain a symbol which is defined as a 17-bit relocatable address of the first character in the block to be reserved. The address field specifies the number of characters to be reserved. It must contain a constant, a symbol, or an address expression which will result in a non-relocatable value.

```
A negative address field such as:    BSS,C    -2B
will be interpreted by COMPASS as:   BSS,C     77775B
and 77775B characters will be reserved.
```

A zero address field does not reserve space, but the location symbol will be defined as above. When BSS,C is encountered, COMPASS will output the binary card it is constructing.

The following illustrates the reservation of storage for characters:

```
ABLE    BSS,C    25
```



23 18	17 12	11 6	5 0
ABLE	ABLE + 1	. . .	
		. . .	ABLE + 23
ABLE + 24	▨	unused	▨

2.4 SUBPROGRAM COMMUNICATION AND LINKAGE

The ENTRY and EXT pseudo instructions establish communication between subprograms. With ENTRY, a programmer may define locations in a subprogram and declare them to be entry points. Symbols declared as external with EXT may be referenced within a subprogram, even though they are not defined within that program. Symbols declared external in one subprogram are declared as entry points in another subprogram. Linked object subprograms are loaded at the same time, but they need not be assembled at the same time. Using ENTRY and EXT, COMPASS produces EPT and XNL loader cards.

On the COMPASS assembly listing, instructions containing references to external names have the usual format; except the address field, prefaced by X, indicates the relocatable word address of a previous instruction in the subprogram area which references the external symbol. If it is the first or only reference to the external symbol, the address field will appear as X77777.

COMPASS places the relocatable address of the last instruction referencing the external symbol into the XNL loader card to begin the threaded list. If no reference is made to the external symbol in the subprogram, the XNL loader card will contain 77777_8, indicating there is no thread.

External names can be associated with either of two threaded lists; one for 15-bit addresses and one for 17-bit addresses. All references are chained in the threaded list with only the symbol in the EXT declaration appearing in an XNL card.

2.4.1 ENTRY

LOCATION		OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS	
1 8		10	20	41	73
		ENTRY	$m_1,m_2,$...............	...............$,m_n$	

The address field contains one or more location names separated by commas; it may not contain blanks. The field terminates with the first blank or column 73. Each subfield contains a symbol defined as a relocatable word address by appearance in a location field elsewhere in the subprogram.

If an entry point symbol appears in a location field of a character definition instruction, an error will be flagged. The location field is ignored by COMPASS and should be blank.

2.4.2 EXT

LOCATION		OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS	
1 8		10	20	41	73
		EXT	$m_1,m_2,$...............	...............$,m_n$	

Symbols referenced but not defined in the subprogram must be declared as external names in EXT pseudo instructions.

The address field contains one or more subfields separated by commas; it may not contain blanks. This field terminates with column 73 or the first blank column. Each subfield contains a symbol which is output to an XNL loader card. The symbol must not be defined within the subprogram which declares it as external; it may be referenced only from an instruction assembled into the subprogram area.

The location field is ignored by COMPASS and should be blank.

2.5 DEFINITION BY EQUATING

A symbol in the location field may be defined by equating it to the value of another symbol, a constant, or an expression of the address field. It may be defined as an absolute value, a relocatable word or relocatable character address. If a symbol is declared an entry point in the subprogram, it must not be equated to a symbol declared as external. When the symbols are equated, they are identical and interchangeable.

All symbols in the address field must have been previously defined by appearance in the location field of a preceding instruction or in an EXT declaration. If an entry point is erroneously equated to an external symbol, COMPASS will not always log an error; but when the object subprogram is loaded, an error will result.

2.5.1 EQU

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
symbol	EQU	m	

The location symbol is equated to another symbol, a 15-bit word address or a 15-bit value. If the location field does not contain a symbol, an error occurs.

The address field determines the definition of the symbol in the location field. It may contain:

An integer modulo 2^{15} -1

A symbol defined by appearance in the location field of a preceding instruction. The symbol in the location field is equated to the entry in the address field. If the symbol in the address field is relocatable to a given area, the symbol in the location field is also relocatable to that area.

An address expression containing symbols defined as above, and conforming to the rules for m subfields. Expressions must not result in a complement relocatable value.

2.5.2 EQU, C

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
symbol	EQU,C	r	

The symbol is equated to a 17-bit address, 17-bit constant, or another symbol. If the location field does not contain a symbol, an error occurs.

The address field determines the definition of the symbol in the location field. It may contain:

An integer modulo 2^{17} -1.

A symbol defined by appearance in the location field of a preceding instruction. The symbol in the location field is equated to the entry in the address field. If the symbol in the address field is relocatable to a given area, the symbol in the location field is also relocatable to that area.

An address expression containing symbols defined as above and conforming to the rules for r subfields. Expressions must not result in a complement relocatable value.

2.6 ASSEMBLY OF CONSTANTS

Constants may be stated as octal, decimal, or character in the source language. They may be single, double, or variable precision of fixed or floating point format. Constants may be placed into bit positions of variable length fields. Character constants may be placed into full words or character positions.

2.6.1 OCT

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS	
1 8	10	20	41	73
symbol or blank	OCT	$m_1,m_2,$...............	...............$,m_n$	

The OCT pseudo instruction expresses constants as signed or unsigned octal integers of 8 or less digits. As many constants can be expressed in the address field as can be written from column 20 through 72; they are separated by commas. The address field is terminated by the first blank or column 73. The octal constants are assembled, right adjusted, for storage into consecutive locations.

An optional binary scale factor is specified with a B suffix and a scale factor expressed as a signed or unsigned decimal integer of not more than two digits. The magnitude of the constant after scaling must be less than 2^{24}.

The location field may be blank or contain a symbol which yields the 15-bit word address of the first constant in the address field.

Example:

```
OCT 77777777,12345670,76543210
```

	octal result
word 1	77777777
2	12345670
3	76543210

```
OCT + 1,-57,2040,-2
```

	octal result
word 1	00000001
2	77777720
3	00002040
4	77777775

```
OCT 72B2
```

octal result
00000350

2.6.2 DEC

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS	
1 8	10	20	41	73
symbol or blank	DEC	d_1,d_2,...............	...............,d_n	

The location field of the DEC instruction may be blank or contain a symbol which is the relocatable word address of the first constant in the address field. The address field may consist of as many subfields, separated by commas, as the card can contain. The first blank or column 73 terminates the address field and subsequent information is treated as remarks.

Decimal constants may be converted for storage as single precision fixed point binary constants. A decimal and/or binary scale factor may be expressed for the 24-bit constant. The decimal may consist of a sign and not more than seven digits with a magnitude of less than 2^{23}. The decimal integer may be followed by a decimal or a binary scaling factor or both; if both are stated, they may appear in either order.

Examples:

1	decimal integer
+2	decimal integer
-38	decimal integer
1D5	decimal integer, decimal scale factor
73D-2	decimal integer, decimal scale factor
-6D+1B4	decimal integer, decimal and binary scale factors
200B-7	decimal integer, binary scale factor
36B+2D1	decimal integer, binary and decimal scale factors

The magnitude of the constant after scaling must be less than 2^{23}. The conversion is performed in three steps:

1. The decimal integer is converted to binary; the binary integer must be less than or equal to 2^{23}-1 in magnitude.
2. The binary integer is multiplied or divided by 10^{d}; d is the decimal scaling factor. The magnitude of the result must be less than 2^{47}. If the decimal scaling factor is negative, a 47-bit fraction or mixed fraction is formed.
3. The result in step 2 is shifted the number of bits specified by the binary scaling factor. A negative factor produces a right shift; a positive scale factor, a left shift. If non-zero bits are lost from the high order 24 bits of the result from step 2, an error is flagged. Loss of low order bits of the intermediate result is not flagged as an error.

2.6.3 DECD

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS	
1 8	10	20	41	73
symbol or blank	DECD	$d_1,d_2,$...............	...............,d_n	

Decimal values may be stored as double precision fixed point constants or floating point constants. Either format requires 48 bits for storage. The location and address fields are treated in the same fashion for DEC and DECD. A symbol in the location field references the first of the two words assembled as the result of DECD.

Fixed point constant format differs from the DEC single precision constants in that magnitudes may be larger. Up to 14 decimal digits may be specified, expressing a value of less than 2^{47}. Decimal and binary scale factors may be used as in DEC. The signed 48-bit binary result is stored in two consecutive computer words.

Floating point constants contain a decimal point. They are stored as two 24-bit words made up of a 12-bit characteristic and a 36-bit mantissa. Negative values are held in complement form.

	23 12	11 0
word 1	characteristic	man-
word 2	tissa	

Floating point constants may contain not more than 14 decimal digits and a decimal point which may appear anywhere within the constant. Binary scaling is not permitted. Decimal scaling is specified with a D suffix followed by a signed or unsigned decimal scaling factor. In the absence of a sign, a positive value is assumed. The result after scaling must not exceed the capacity of the hardware (approximately $10^{\pm 308}$).

2.6.4 BCD

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS	
1 8	10	20	41	73
symbol or blank	BCD	n,c_1c_2..............	..............c_{4n}	

Characters are assembled for storage into consecutive computer words. They are stored as 6-bit binary coded decimal character codes (internal BCD) into addressable character positions. The location field may be blank or contain a symbol which is established as the 15-bit relocatable word address of the first word in the field.

In the address field the decimal integer n specifies the number of words to be used. Following a comma after n are the characters to be converted and stored. Four characters can be contained in one word; 4n characters may be punched in one card. If 4n is greater than the number of characters that can be contained on a card, through column 72, additional positions reserved by n will be filled with blanks. Information between 4n characters and column 73 is treated as a comment.

2.6.5 BCD, C

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS	
1 8	10	20	41	73
symbol or blank	BCD,C	n,c_1c_2..............	..............c_n	

Characters may be assembled for storage into consecutive character positions; they are converted and encoded as for BCD. The modifier, C, in the operation code indicates that character addresses and character strings rather than words are to be processed. The location field may be blank or contain a symbol established as a 17-bit character address.

The number of character positions to be reserved is specified by n (1 to 2^{15} -1), the characters to be converted are specified by $c_1 - c_n$. If n specifies more characters than can be contained on one card through column 72, excess positions to be reserved will be filled with blanks. Any information appearing between n characters and column 73 will be treated as comments.

Characters are stored in consecutive positions. If BCD,C is immediately preceded by a line of code which assigns character storage (BSS,C) rather than word storage (BSS), the character string begins in the first available character position. Should the preceding line assign word storage, the character string begins in the first character position of the first word available.

If the number of characters declared by BCD,C does not fill an entire word, the unused positions are filled with zeros. If the next instruction which consumes space in the object program is BCD,C, the positions in the partial word are assigned to the leading characters to produce a packed field.

Example:

```
MOTCC   BCD,C 35,ABCDEFGHIJKLMNOPQRSTUVWXYZ = '+ ⁺0.)-0̄$
        BCD,C 15,* /,(1234567890
```

Location	Contents			
MOTCC	A	B	C	D
	21	22	23	24
	E	F	G	H
	25	26	27	30
	I	J	K	L
	31	41	42	43
	M	N	O	P
	44	45	46	47
	Q	R	S	T
	50	51	62	63
	U	V	W	X
	64	65	66	67
	Y	Z	=	'
	70	71	13	14
	+	+0	.	)
	20	32	33	34
	-	-0	$	*
	40	52	53	54
		/	,	(
	60	61	73	74
	1	2	3	4
	01	02	03	04
	5	6	7	8
	05	06	07	10
	9	0	0	0
	11	00	00	00

2.6.6 ASCII

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
symbol or blank	ASCII,p	n,$c_1c_2c_3c_4$...........	...................c_{pn}

BCD characters are converted to ASCII characters and stored in consecutive words. The ASCII characters are stored p-characters per word; p may be 1, 2, or 3.

p = 1:	bits 23-08	zero
	bits 07-00	ASCII character
p = 2:	bits 23-20	zero
	bits 19-12	first ASCII character
	bits 11-08	zero
	bits 07-00	second ASCII character
p = 3:	bits 23-16	first ASCII character
	bits 15-08	second ASCII character
	bits 07-00	third ASCII character

In the address field, the decimal integer n specifies the number of words to be used. Following a comma after n are the BCD characters to be converted and stored. This results in n computer words, each containing p ASCII characters. Anything after pn characters is treated as remarks. If pn is greater than the number of characters that can be contained on a card, through column 72, additional positions reserved by n will be filled with blanks.

If p is omitted, p is assumed to equal 2.

The location field may be blank or contain a symbol which is established as the 15-bit relocatable word address of the first word of the field.

2.6.7 BCDN

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS	
1 8	10	20	41	73
symbol or blank	BCDN	n,sddd..............	...............ddd	

BCD numeric characters are converted to 4-bit characters and stored in n consecutive words.

In the address field, the decimal integer n specifies the number of words to be used. Following a comma after n are the BCD numeric characters and related sign to be converted and stored as 4-bit characters:

s	sign (+ or -; if omitted, + is assumed)
d	BCD numeric character

If n specifies a number of words greater than that required for the conversion, leading zeros are inserted. If the number of characters cannot be contained in n words, an A-error appears on the assembly listing. If any character d is not in the range 0-9 an A-error also appears.

The 4-bit characters are stored from right to left beginning with the least significant characters. The sign is stored in the rightmost character position (positive 1010_2, negative 1011_2).

The location field may be blank or contain a symbol which is established as the 15-bit relocatable word address of the first word of the field.

2.7 VARIABLE FIELD DEFINITION

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS	
1 8	10	20	41	73
symbol or blank	VFD	mn/v,................	..............,mn/v	

VFD enters octal numbers, character codes, relocatable addresses, or constants into variable length fields assigned as continuous strings of specified length. Information is placed regardless of word length of character position. Values are entered right adjusted and character strings left adjusted. Each VFD instruction begins filling a new computer word.

The location field may contain a legal symbol or blanks. A symbol yields a relocatable word address. As many address subfields are allowed as can be contained on a single card through column 72. The address subfield terminates with a comma; a blank terminates the VFD pseudo instruction. The mode parameter, m, may designate one of five modes; the remainder of the subfield is governed by the specified mode.

m	mode indicator
n	unsigned decimal integer specifying the number of bit positions in the variable field. The range of values for n varies with mode.
/	separates the description of the field mode and length from the statement of the field content.
v	content of the variable field; varies according to mode and is restricted by declared length.

2.7.1 VFD MODES

The statement of variable field length and content varies according to the mode. Five modes may be expressed in a VFD address subfield.

OCTAL VFD On/v

In octal, n may be 1 to 24 and v may be a maximum of 8 octal digits; the integer may be signed. If negative, the field content is stored in one's complement form. The value is entered right justified with leading bits inserted according to the sign and length. If the value exceeds the length of the field, an error is flagged and the field is set to zero. A binary scale factor may be supplied in the same manner as for the OCT pseudo instruction.

```
Example:      VFD      O5/17
```

HOLLERITH VFD Hn/v

Hollerith information is stored as 6-bit internal BCD character codes; n must be a multiple of six; v terminates with the first comma or blank. If the subfield does not terminate after the n/6 character, an error results.

```
Example:      VFD      H12/KY
```

ARITHMETIC EXPRESSION VFD An/v

The arithmetic expression consists of a constant, a symbol, or an expression formed by the rules for address field arithmetic.

If an expression yields a relocatable word address, the field length n must be at least 15, the programmer must enter the value into the computer word right justified to bit zero and the expression is evaluated as modulo 2^{15}-1. If an expression yields a fixed value, the field length n may be 1-24 and the expression is evaluated as modulo 2^{n}-1.

```
Example:      VFD      A6/63,A3/7,A15/JOE
```

CHARACTER ADDRESS VFD Cn/v

This variable field is governed by the above rules, except that a minimum of 17 bits is required for an expression which yields a relocatable character address. A relocatable expression is evaluated modulo 2^{17}-1.

Example: VFD C7/0,C17/JOE

ASCII VFD In/v

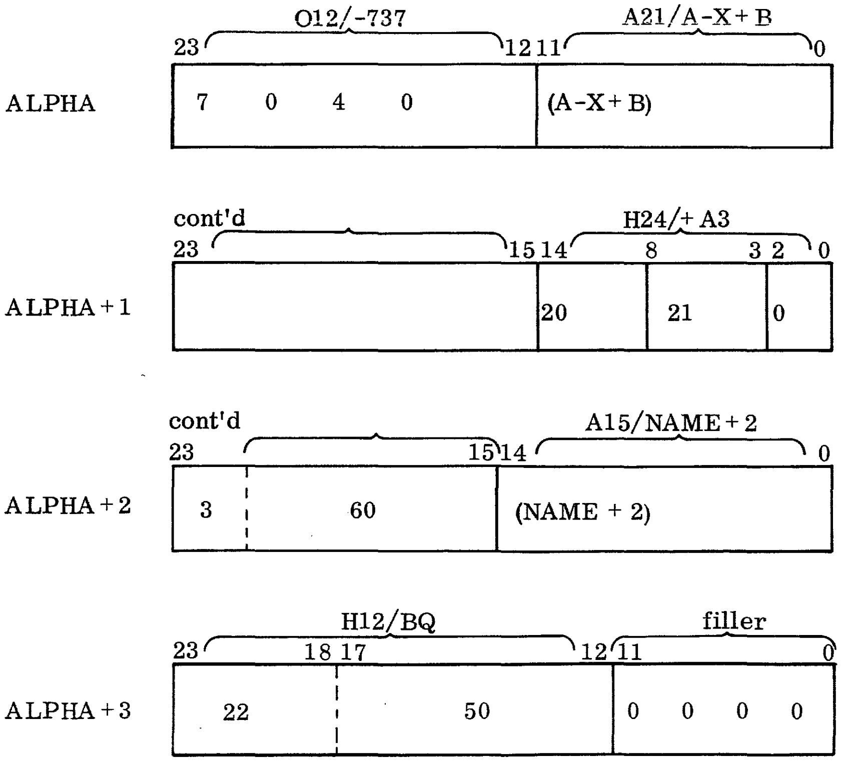
BCD characters (v) are stored as 8-bit ASCII characters. n must be a multiple of 8 and cannot exceed 96. If v does not terminate after the n/8 character, an error results. The last character is followed by a space or a comma.

Example: VFD I8/A

Example:

```
VFD012/-737,A21/A-X+B,H24/+AB,A15/NAME12,H12/BQ
```

A,X, and B are not relocatable symbols. Four words are generated, with the data placed as follows:



2.8 ASSEMBLER CONTROL

Source subprogram assembly may be conditional as stated by the pseudo instructions listed below. COMPASS tests for the condition and includes subsequent lines of code depending on the outcome of the test.

IFZ	if zero
IFN	if non-zero
IFT	if true
IFF	if false

IFZ and IFN may be used as desired in a subprogram. IFT and IFF, which compare a parameter string against stated variables, may occur only within a macro prototype; their use is discussed in the chapter on macros.

2.8.1 IFZ

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
	IFZ	m,n	

An arithmetic expression may be tested for zero to determine whether subsequent instructions should be included in a subprogram. The expression must conform to the rules for address expressions. A symbol in the location field is ignored by COMPASS but included in the output listing.

The address field consists of two subfields containing previously defined symbols.

m	is an expression, the value of which is computed as any address expression and evaluated modulo 2^{15} -1.
n	contains an integer or an expression which results in a positive non-relocatable value.

If the expression in the m subfield results in zero, the psuedo instruction IFZ is printed and the following n lines of code are assembled into the object subprogram. If the m subfield yields a non-zero value, the pseudo instruction IFZ is not printed and n lines of code are skipped. Symbols in the address field must be defined by appearance in the location field of a preceding instruction.

2.8.2 IFN

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
	IFN	m,n	

This pseudo instruction is the same as IFZ except that n lines of code are assembled if the value in the m field is non-zero.

2.9 LISTING CONTROL

The following pseudo instructions apply to COMPASS output listings:

REM	insert remarks
NOLIST	suppress output listing
LIST	resume output listing
SPACE	space lines on output listing
EJECT	eject printer paper to top of next page
TITLE	begin succeeding pages with title given
asterisk (colume one)	print card columns 2-80 as a comment

2.9.1 REM

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
any	REM	any	

Remarks may be inserted into the source program to appear on the output listing with this pseudo instruction. All fields except columns 9 to 13 of the operation code field may be used for remarks; for example:

THIS IS REM A REMARK PSEUDO-INSTRUCTION

2.9.2 NOLIST

LOCATION		OPERATION, MODIFIERS ADDRESS FIELD		COMMENTS
1 8		10	20	41
		NOLIST		

NOLIST suppresses listing of the subprogram until LIST appears in the source program. Lines in the source program containing errors will be listed regardless of NOLIST. The location and address fields are ignored by COMPASS. The pseudo instruction will not appear on the output listing. The number of instructions not listed is counted and when LIST mode resumes, the following appears:

PRINTING SUPPRESSED FOR xxx LINES
xxx = number of lines

2.9.3 LIST

LOCATION		OPERATION, MODIFIERS ADDRESS FIELD		COMMENTS
1 8		10	20	41
		LIST		

LIST resumes output listing after NOLIST has been used. If LIST occurs without a preceding NOLIST, it is ignored. The pseudo instruction will not appear on the output listing.

2.9.4 SPACE

LOCATION		OPERATION, MODIFIERS ADDRESS FIELD		COMMENTS
1 8		10	20	41
		SPACE	m	

This pseudo instruction specifies m lines are to be skipped on the printed output listing. If, as a result of SPACE, the end of the page is reached, printing resumes with the first line of the new page. A symbol in the location field is ignored. The pseudo instruction, SPACE, will not appear on the output listing.

The parameter, m, may be an unsigned decimal integer, 0 to 32767.

2.9.5 EJECT

LOCATION		OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8		10	20	41
		EJECT		

When EJECT is encountered, the printer skips to the top of a new page. A symbol in the location field is ignored. The pseudo instruction, EJECT, will not appear on the output listing.

2.9.6 TITLE

LOCATION		OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8		10	20	41
		TITLE	heading	

The TITLE pseudo instruction describes a heading to be printed at the top of each page of a listing. If the first page of the listing is to be titled, TITLE must immediately follow IDENT. A symbol in the location field is ignored. The contents of columns 20-72 of the address field contain the title. The pseudo, TITLE, will not appear on the output listing.

In the body of the subprogram, TITLE information replaces the present heading obtained from IDENT or preceding TITLE. If TITLE occurs in midpage, it is not acted on until top of the next page is encountered. However, any lines following TITLE will be printed to fill out the first page. Therefore, EJECT should follow TITLE to insure that all material succeeding TITLE is printed under the proper heading.

MACROS 3

When an operation is performed frequently in a program or in many programs, the sequence of machine or pseudo instructions which accomplish that operation may be grouped together to form a macro. This group of instructions may contain formal parameters which are given actual values when the macro is called.

Macros are defined and called by COMPASS pseudo instructions. The same code is obtained and included in the subprogram each time the macro is called.

Library macros reside on the system library. All library macros to be called in a subprogram must be declared in LIBM pseudo instructions immediately following the IDENT pseudo instruction. Since the library macros may be unique to an installation, a list of macros should be available to the programmer.

The programmer may define his own macros. They are defined only in the subprogram in which they occur and for reference within that subprogram. Programmer macro definitions may not precede the pseudo instructions LIBM or IDENT.

All macro definitions are composed of the following:

Macro heading — Names the macro and declares the formal parameters used in the prototype.

Prototype — Contains the instruction sequence with variable elements expressed as formal parameters.

Macro terminator — Defines the end of the macro definition.

The pseudo instruction which brings the prototype into the body of the program is the macro call. It consists of the macro name and a string of actual parameters to be substituted for the formal parameters in the prototype.

3.1 MACRO HEADING

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
name	MACRO	$(p_1, p_2, \ldots\ldots\ldots\ldots$	$\ldots\ldots\ldots\ldots, p_n)$

The macro heading consists of one or more lines of the pseudo instruction, MACRO. The location field contains the macro name, which may not be a hardware instruction or pseudo instruction.

The address field contains a set of formal parameters. The address subfield or a portion of it may be expressed as a single formal parameter if that portion of the subfield is set off by a plus or minus sign, a comma, a blank or, in the case of VFD, a slash.

If the formal parameter list exceeds a single code line, the list is continued in subsequent MACRO pseudo instructions with the following restrictions:

The location field is blank.

The operation field contains the mnemonic, MACRO.

A formal parameter field and its terminal comma must be on a single line prior to column 73.

In the address field, the parameter list, enclosed in parentheses, may contain alphanumeric symbols separated by commas; blanks may precede or follow the parameter but may not be embedded. The parameter symbols are local to the macro and may be used elsewhere in a program without ambiguity.

Examples:

```
DIVIDE      MACRO    (P1,P2,P3,P4)
MULTIPLY    MACRO    (P1,P2,P3,P4,
            MACRO     P5,P6)
```

The MACRO pseudo instruction must immediately follow IDENT, LIBM, ENDM, or MACRO, except that comment cards (*in column 1) and REM or TITLE may intervene. When MACRO follows IDENT, LIBM, or ENDM, it defines a macro instruction, and the location field must contain the macro name.

3.2 PROTOTYPE

A set of instructions, called the prototype, follows the heading line. It is up to the programmer to insure that when the macro is called the resulting code will not contain illegalities.

Formal parameters may represent any portion of an instruction or an entire instruction except for the location field. This flexibility is attained through the use of parentheses as delimiters.

In the prototype, the location field of an instruction may contain a symbol of four characters or less. Any location defined in the subprogram may be referenced within the prototype; however, a location within a macro prototype is local to the macro and may not be referenced from outside the macro. COMPASS will substitute an internally generated symbol for the local location symbol and for all references to it within the macro.

Reference may be made within the prototype to symbols external to the subprogram if they are declared by EXT pseudo instructions within either the macro or the subprogram. An EXT declaration within the macro remains in force for the entire subprogram.

If the EQU pseudo instruction appears within the macro instruction prototype, the symbol in the location field is considered local to the macro and treated as any location symbol in the macro.

3.2.1 IFT

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
	IFT	m,p,n	

Within a macro prototype, lines of code may be excluded or included in an object subprogram with the IFT pseudo instruction which compares the first two subfields in its address field for literal equality. If the two character strings are equal, subsequent lines of code are assembled; otherwise they are excluded from the object program.

A symbol in the location field is ignored by COMPASS. If the IFT condition is met, the IFT instruction appears in the output listing; otherwise it is not printed.

The three address subfields are:

m first comparand

p second comparand

n must result in a positive non-relocatable value denoting the number of lines of code to be assembled or excluded

The m and p terms may be character strings or formal parameters; the character string may not include slashes. If a character string is identical to a formal parameter, the string must be enclosed in slashes.

The actual values compared are obtained by COMPASS as follows:

If the subfield is enclosed in slashes, the content is used in the comparison.

If the subfield contains a formal parameter, COMPASS substitutes the actual parameter before the test is made.

If the subfield is not a formal parameter and is not enclosed in slashes, the character string is used as though slashes had appeared.

The n term must be a symbol, constant, or expression which results in a non-relocatable value. Symbols in the address field must be previously defined.

If the m and p terms compare bit for bit, n lines of code immediately following the IFT pseudo instruction are assembled into the subprogram. If the m and p terms are unlike, n lines are skipped and not assembled by COMPASS.

Examples:

Macro definitions:

```
COMPUTE     MACRO      (P1,P2,P3,P4,P5,P6)
            LDA        P1
            DVA        P2
            STQ        P3
            IFT        /P6/,P5,2
            ENA        P4
            ENI        P6
            ENDM
```

The following sequence of instructions occurs within a subprogram and the call refers to the previously defined macro set.

Macro call 1:

```
CAKE      STA         TABLE
          COMPUTE     (B,C,A,LOC1,P6,56)
          LDAQ        QUANTITY
            .
            .
            .
```

The assembler would generate:

```
CAKE      STA         TABLE
          LDA         B
          DVA         C
          STQ         A
          IFT         P6,P6,2
          ENA         LOC1
          ENI         56
          LDAQ        QUANTITY
```

Since the actual parameter substituted for P5 is identical to the character string "P6", the assembler includes the two instructions, ENA and ENI. The IFT instruction does not appear in the object subprogram.

Macro call 2:

```
          STA         TABLE
          COMPUTE     (B,C,A,LOC2,54,56)
          LDAQ        QUANTITY
```

The assembler would generate:

```
          STA         TABLE
          LDA         B
          DVA         C
          STQ         A
          LDAQ        QUANTITY
            .
            .
            .
```

Since 54 is not equal to the characters enclosed in slashes in the IFT pseudo instruction, the assembler does not assemble the two instructions, ENA and ENI. Assembly continues with the next instruction from the input deck.

3.2.2 IFF

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
	IFF	m,p,n	

The conditional pseudo instruction IFF functions the same as IFT, except if the comparands are unlike, the next n lines of code are assembled. If the m and p terms are identical, the n lines of code are excluded.

3.3 MACRO TERMINATOR

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
	ENDM		

ENDM terminates a macro definition. A symbol in the location field will be ignored by COMPASS but included on the output listing.

Example:

```
AOK    MACRO    (P3,P2,P1,P4)     Macro Heading
       ENI      P1,1
A      LDA      P2,1
       P3       B                 Prototype
       STA      P2
       IJD      A,1
       UJP      SCRAM
B      DEC      P4
       ENDM                       Macro Terminator
```

Formal parameter P1 represents an operand, P2 an address, P3 an operation code, and P4 a decimal constant. Locations A and B are local to the macro and may be used elsewhere without ambiguity.

3.4 MACRO CALLS

LOCATION	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8	10	20	41
symbol or blank	macro name	$(p_1,p_2,$...............	..............$,p_n)$

The macro call names the macro to be inserted at this point in the program and assigns a set of actual parameters to be substituted for the formal parameters in the prototype. The actual parameters, p1,...,pn, must appear in the same order as the formal parameter list in the macro heading.

The location field may be blank or contain a symbol which is the relocatable address of the first instruction that consumes space in the assembled macro.

The operation field may contain any macro instruction name defined for the subprogram by LIBM and MACRO pseudo instructions and prototypes. If the macro is defined for the subprogram, COMPASS will assemble and insert the macro code at the point at which the macro name appears in the operation field.

The address field of the macro name instruction contains the list (enclosed by parentheses) of actual parameters, separated by commas. Single actual parameters may also be enclosed by parentheses within the list. This allows an entire instruction or several subfields of an instruction in the macro prototype to be expressed as a single actual parameter.

Single actual parameters may not include blanks or commas unless the entire actual parameter is enclosed in parentheses. If a single actual parameter is enclosed by parentheses, it may contain any character legal for the portion of the instruction it represents, except a right parenthesis (see example below). An actual parameter may be omitted but a trailing comma must appear. Actual parameters not expressed are assembled as zeros.

The address field of the macro name instruction may contain constants, symbols, expressions, or Hollerith literals. Actual parameters retain the sequence of the formal parameter list in the macro definition. When COMPASS assembles the macro, the actual parameters are transferred to the position at which the formal parameters are referenced in the prototype. The address field of a single line of code terminates at column 72 or with a right parenthesis.

If the list of actual parameters is too long for a single line of code, it may be continued on subsequent lines with blank location fields and the macro name operation code repeated. An actual parameter must be wholly contained on a single line. If the list is not closed by a right parenthesis, an error results.

Actual parameters may not contain entries for location fields in the prototype. These fields will not be modified by COMPASS in assembling a macro instruction.

Examples of Programmer Macro Definition:

Example A:

```
DIVIDE      MACRO       (P1,P2,P3)
            LDAQ        P1
            DVA         P2
            STQ         P3
            ENDM
```

Macro call:

```
            DIVIDE      (DICK,DAVE,DAN)
```

Assembled:

```
            LDAQ        DICK
            DVA         DAVE
            STQ         DAN
```

Example B:

Macro Definition:

```
COMPUTE     MACRO       (P1,P2,P3,P4,P5,P6,P7,P8,P9)
            LDA         P1
            LDQ         P1
            ADA         P2
            SBAQ        P3
            VFD         P4/P5
            P4          **
            DVA         P6
            STQ         P7
            LDA         P7
            P8          *-P9
            ENDM
```

The above example shows how parameters may be specified in the operation or address field, or both, within the macro set of instructions. It also shows that a parameter may appear more than once in a set of instructions.

Example C:

Macro Definition:

```
TOSS      MACRO       (P1,P2,P3,P4)
          LDA         TOM,3
          ADA         DICK
          STA         MARV
          UJP,P2
TOM       P1
DICK      DEC         P4
MARV      P3
          ENDM
```

Macro call:

```
BETTY     TOSS        ((BCD 6,A), (I JOE,2))
```

Assembled:

```
BETTY     LDA         TOM,3
          ADA         DICK
          STA         MARV
          UJP,I JOE,2
TOM       BCD 6,A
DICK      DEC         0
MARV      00
```

This example demonstrates how multiple parentheses are used and how actual parameters are assembled as zeros when they are not expressed before the formal parameter list terminates.

3.5 NESTING OF MACROS

A macro definition may, itself, contain an unlimited number of macro calls to library or programmer macros defined for the subprogram. These inner macro calls become effective at the time a call is made to the outer macro.

A macro definition may be passed as an actual parameter at call time or a macro definition may contain calls to itself; it is the programmer's responsibility to prevent infinite recursion through the use of conditionals.

The use of local symbols is the same as that explained in 3.2. Local location symbols are unique to each macro call.

The parameter list in the address field of an inner macro instruction must be enclosed in parentheses. The list may not contain imbedded blanks; the occurrence of a blank will terminate parameter substitution.

The parameters of the inner macro may consist of parameters of the outer macro. At macro call time, inner macro parameters will be substituted with the actual corresponding parameters of the outer macro.

An inner macro may use the continuation feature for lengthy parameter lists. When coding a macro prototype which contains a macro instruction, consideration should be given to the length of the actual parameters. The continuation feature enables the programmer to divide the list to avoid overflow of a card image. When overflow occurs, information is lost and the macro is improperly generated.

Example:

```
A      MACRO          (P1,P2,P3,P4,P5,P6,P7,P8)
       B              (P1,P2,P3,P4,P5,P6,P7,P8)
       ENDM
```

where B is a macro instruction.

The following modification with the continuation feature will ensure that an overflow will not occur when the A macro is called:

```
A      MACRO          (P1,P2,P3,P4,P5,P6,P7,P8)
       B              (P1,P2,P3,P4,
       B              P5,P6,P7,P8)
       ENDM
```

Example:

Macro Definition:

```
COMPUTE      MACRO      (P1,P2,P3)
             ENA        0
             STA        P1
             ENI        47,P3
HELP         AZJ,P2     WOOF
JOHN         SHQ        1
             IJD        HELP,P3
             UJP        *+2
WOOF         RAD        P1
             UJP        JOHN
             ENDM
```

Macro Definition:

```
JUMP       MACRO      (P1,P2,P3)
           LDQ        P1
           LDL        P2
           SHAQ       24
           COMPUTE    (D,LT,P3)
           ADA        TOT
           STA        TOT
           ENDM
```

A macro call of the form:

```
           JUMP       (ONE,M2,3)
```

will generate the following sequence of instructions:

```
           LDQ        ONE
           LDL        M2
           SHAQ       24
           ENA        0
           STA        D
           ENI        47,3
HELP       AZJ,LT     WOOF
JOHN       SHQ        1
           IJD        HELP,3
           UJP        *+2
WOOF       RAD        D
           UJP        JOHN
           ADA        TOT
           STA        TOT
```

3.6 LIBRARY MACROS

LOCATION		OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS
1 8		10	20	41
		LIBM	$name_1$,$name_2$,.........	..........., $name_n$

LIBM instructs COMPASS to call the named library macros from the system library. A location symbol will be ignored by COMPASS but included on the output listing. The address subfields contain the names of library macros separated by commas; the address field is terminated by the first blank or column 73.

All library macros to be called in a subprogram must be declared in LIBM pseudo instructions immediately following the IDENT pseudo instruction; otherwise an error will result. Comment cards with an asterisk in column one or the pseudo instructions REM and TITLE may intervene, however. LIBM does not consume space in the object program.

The programmer may use as many LIBM pseudo instructions as required. However, a macro name must be wholly contained within a single subfield on a single line of code.

Available library macros are defined in the operating system reference manual and the Data Processing Package reference manual.

APPENDIX SECTION

MACHINE LANGUAGE INSTRUCTIONS A

The Control Data instruction repertoire for data processing, scientific, and logical programming contains optional sets of BCD, floating point, and double precision instructions for the hardware. All of these, including the optional commands, may be coded in the COMPASS language using mnemonic codes and comprehensive symbolic programming techniques. This appendix describes how machine language instructions are expressed in COMPASS, how COMPASS assembles them, and how they appear in the object program. †

Control Data provides a set of simulation routines for the optional instructions. For optional sets not included at an installation, simulator routines may be placed on the library tape and called as subroutines. Therefore, a programmer may use the mnemonics in the source subprogram as if the hardware were present.

INSTRUCTION SUBFIELDS

Instruction fields may be optional or mandatory: an optional field may be expressed or not, as the programmer requires; a mandatory field must be present and must contain only specific parameters. The indirect addressing field and the b field are examples of optional fields. The conditional modifiers for the AZJ instruction are an example of a mandatory field.

ADDRESS SUBFIELDS

m, n and y — The m, n and y subfields for machine instructions may be represented by a symbol, the special symbols * and **, a constant, an expression, or a literal. The m and n subfields represent operand addresses; the y subfield represents an operand.

r and s — Machine language instructions using a 17-bit character address contain r or s subfields which may be represented as a symbol, literal, constant, external symbol, expression, or the special characters, * and **.

b — The b subfield may be represented by a digit 1, 2, 3, a symbol equated to 1, 2, or 3, an expression with a non-relocatable value of 1, 2, 3, or **. The b subfield designates an index register.

i — The i subfield occurs in the MEQ and MTH instructions; it may be a symbol, constant, or expression which results in a non-relocatable value from 0 to 7, or **.

† For more detailed explanations of the machine language instructions, see 3200 Computer System Reference Manual, Pub. No. 60043800 or 3300 Computer System Reference Manual, Pub. No. 60157000.

In the following example ABLE = 100_8 INTERVAL = 1.

Coding:	Results (in octal)		
MEQ ABLE, INTERVAL	06	1	00100
MEQ ABLE, INTERVAL+1	06	2	00100
MEQ ABLE, 2	06	2	00100
MEQ ABLE, 8	06	0	00100
MEQ ABLE, **	06	7	00100

v The v subfield machine language instruction denotes a location in the register file. It may be any symbol, constant, or expression which results in a non-relocatable value 0 to 63_{10} or **.

In the following examples, ABLE is equated to the value 0011_8 elsewhere in the program.

Coding:	Results (in octal)				
TMA ABLE	53	0	2	. . .	11
TMA 77B	53	0	2	. . .	77
TMA **	53	0	2	. . .	77
TMA ABLE+22B	53	0	2	. . .	33

x The connect code for input/output units or the comparison mask for interrupt instructions is represented by x. This subfield may contain a symbol, constant, or expression which results in a non-relocatable value $0 \leq x \leq 2^{12}-1$, or **.

ch This subfield contains the channel designator for input/output instructions. It may contain a symbol, constant, or expression which results in a non-relocatable value $0 \leq ch \leq 7$, or **.

ℓ The ℓ subfield specifies the length of a character field.

MOVE: The ℓ subfield may be a symbol or an expression which results in a non-relocatable value from 1 to 177_8, or **.

cm The 8-bit channel mask for CILO and CLCA instructions is represented by cm. This subfield may contain a symbol, constant, or expression which results in a non-relocatable value $0 \leq cm \leq 2^8-1$ or **.

In the following examples, ABLE is equated to 100_8 elsewhere in the program, BAKER to 00200_8. Both are equated as 17-bit character addresses.

Coding:

Fields: ℓ, r, s		Results (in octal)
MOVE ABLE,BAKER,BAKER+100B	word 1	72000300
	2	40000200
	3	
MOVE 128,BAKER,BAKER+128	word 1	72000400
	2	00000200
	3	
MOVE 27B,BAKER,BAKER+27B	word 1	72000227
	2	13400200
	3	
MOVE **,BAKER,BAKER+100B	word 1	72000300
	2	77400200
	3	

c The c subfield specifies a search character.

SRCE or SRCN: The c subfield may be any symbol, constant, or ** which represents the 6-bit character code of the character for which the search is made, $00 \leq c \leq 77_8$

A is defined elsewhere in the program as 21_8; ABLE and BAKER are defined as 00200_8 and 00100_8.

Coding		Result (in octal)
SRCE A,ABLE,BAKER	word 1	71000200
	2	21000100
	3	
SRCE 21B,ABLE,BAKER	word 1	71000200
	2	21000100
	3	
SRCE A+21B,ABLE,BAKER	word 1	71000200
	2	42000100
	3	

Term	Meaning
A	24-bit A register
b	index register designator 1 to 3
B	index register defined by B^b
B_m	index register flag, M = m+(B_m) for these instructions only
B_r	index register flag. If B_r = 1 or 3, R = r+(B^1). If B_r = 2, R = r+(B^2). If B_r = 0, R = r.
B_s	index register flag. If B_s = 1 or 3, S = s+(B^1). If B_s = 2, S = s+(B^2). If B_s = 0, S = s.
c	$00\text{-}77_8$ BCD code of search character
cm	8-bit channel mask
D	D register
E	48 (52)-bit E register
E_ℓ	lower half of 48-bit E register (bits 23-00)
E_u	upper half of 48-bit E register (bits 47-24)
i	increment or decrement, 0 to 7
k	shift count
ℓ	field length of block, $0\text{-}177_8$
ℓ_r	number of characters in field R
ℓ_s	number of characters in field S
m	15-bit word address, first operand or jump address

Term	Meaning
M	actual operand or jump address as modified; $M = m+(B^b)$
n	same as m, second operand address
p	15 (or 17)-bit P register
Q	24-bit Q register
r	17-bit character address
R	actual character address as modified; $R = r+(B^b)$
s	same as r, second operand address
S	same as R, second operand address; $S = s+(B^b)$
v	6-bit address in register file
sc	scan character
w	page index file address
x	connect code or interrupt mask
y	15-bit operand

Instruction Modifiers	
A	conversion
B	backward read or write
C	evaluate address expression modulo $2^{17}-1$
dc	delimiting character
EQ	equal
GE	greater than or equal
H	half assembly or disassembly
I	indirect addressing
INT	interrupt on completion
N	no assembly or disassembly
NC	no conversion
NE	not equal
S	instruction modifier denoting sign extension S present, sign extended S omitted, no sign extension

In the following instructions, ( ) → ( ) indicates the contents of one register, operand, or address field is replaced by the contents of another register, operand field, or address field. For example: (M) → (A) means "replace contents of A register with contents of M operand field"

3100/3200 INSTRUCTION LIST
(also 3300/3500 non-executive mode)

Mnemonic Code	Octal Code	Address Field	Operation Performed
ADA,I	30	m,b	$(M)+(A) \rightarrow (A)$
ADAQ,I	32	m,b	$(M,M+1)+(A,Q) \rightarrow (A,Q)$
ADE	66	r,3	Up to twelve 4-bit characters added to (E) (most significant character at address R). Sum appears in E. (D) specifies field length
AEU	55.6		$(A) \rightarrow (E_U)$
AIA	53.(0+b)4	b	$(B^b)+(A) \rightarrow (A)$
ANA	17.6	y	$y \wedge (A) \rightarrow (A)$, no sign extension
ANA,S	17.4	y	$y \wedge (A) \rightarrow (A)$, sign of y extended
ANI	17.0	y	No operation
ANI	17.1-3	y,b	$y \wedge (B^b) \rightarrow (B^b)$
ANQ	17.7	y	$y \wedge (Q) \rightarrow (Q)$, no sign extension
ANQ,S	17.5	y	$y \wedge (Q) \rightarrow (Q)$, sign of y extended
AQA	53.04		$(A)+(Q) \rightarrow (A)$
AQE	55.7		$(A,Q) \rightarrow (E_U,E_L)$
AQJ,EQ	03.4	m	If $(A) = (Q)$, RNI m, otherwise RNI P+1
AQJ,GE	03.6	m	If $(A) \geq (Q)$, RNI m, otherwise RNI P+1
AQJ,LT	03.7	m	If $(A) < (Q)$, RNI m, otherwise RNI P+1
AQJ,NE	03.5	m	If $(A) \neq (Q)$, RNI m, otherwise RNI P+1
ASE	04.6	y	If $y = (A_{00-14})$, RNI P+2, otherwise RNI P+1
ASE,S	04.4	y	If $y = (A)$, RNI P+2, otherwise RNI P+1. Sign of y is extended
ASG	05.6	y	If $(A_{00-14}) \geq y$, RNI P+2, otherwise RNI P+1
ASG,S	05.4	y	If $(A) \geq y$, RNI P+2, otherwise RNI P+1. Sign of y is extended

Mnemonic Code	Octal Code	Address Field	Operation Performed
AZJ,EQ	03.0	m	If $(A) = 0$, RNI m, otherwise RNI P+1
AZJ,GE	03.2	m	If $(A) \geq 0$, RNI m, otherwise RNI P+1
AZJ,LT	03.3	m	If $(A) < 0$, RNI m, otherwise RNI P+1
AZJ,NE	03.1	m	If $(A) \neq 0$, RNI m, otherwise RNI P+1
CINS	77.3	ch	Internal status code → (A_{11-0}); (Interrupt Mask Register) → (A_{23-10}); RNI P+1
CON	77.0	x,ch	If channel ch is busy, reject instruction, RNI P+1. If channel ch is not busy, 12-bit connect code sent on channel ch with connect enable, RNI P+2.
COPY	77.2	ch	External status code from I/O channel ch → (A_{11-0}); (interrupt mask register) → (A_{23-12}) RNI P+1.
CPR,I	52	m,b	$(M) > (A)$, RNI P+1 $(Q) > (M)$, RNI P+2 $(A) \geq (M) \geq (Q)$, RNI P+3 (A) and (Q) are unchanged
CTI	77.75		Set Type In Beginning character address must be preset in location 23 of register file and last character address + 1 must be preset in location 33 of the file.
CTO	77.76		Set Type Out
DINT	77.73		Interrupt control is disabled
DVA,I	51	m,b	(A,Q) / (M) → (A), Remainder → (Q)
DVAQ,I	57	m,b	(A,Q,E) / (M,M+1) → (A,Q) and remainder with sign extended → (E). Divide fault, halts operation and program advances to next instruction.
EAQ	55.3		(E_U, E_L) → (A,Q)
ECHA	11	r	0 → (A), then r → (A_{00-16})
ECHA,S	11	r	0 → (A), then r → (A_{00-16}), sign extended

Mnemonic Code	Octal Code	Address Field	Operation Performed
EINT	77.74		Enables interrupt control; allows one more instruction to be executed before interrupt.
ELQ	55.1		$(E_L) \rightarrow (Q)$
ENA	14.6	y	$0 \rightarrow (A)$, then $y \rightarrow (A_{00-14})$
ENA,S	14.4	y	$0 \rightarrow (A)$, then $y \rightarrow (A)$, sign extended
ENI	14.0	y	No operation
ENI	14.1-3	y,b	$0 \rightarrow (B^b)$, then $y \rightarrow (B^b)$
ENQ	14.7	y	$0 \rightarrow (Q)$, then $y \rightarrow (Q_{00-14})$
ENQ,S	14.5	y	$0 \rightarrow (Q)$, then $y \rightarrow (Q_{00-14})$, sign extended
EOJ	70.6	m	Jump to m if E overflows, otherwise RNI P+1.
EUA	55.2		$(E_U) \rightarrow (A)$
EXS	77.2	x,ch	Sense external status. If 1 bits occur on status lines in any of the same positions as 1 bits in the mask, RNI P+1. If no comparison, RNI P+2.
EZJ,EQ	70.4	m	$(E) = 0$, jump to m; $(E) \neq 0$, RNI P+1
EZJ,LT	70.5	m	$(E) < 0$, jump to m; $(E) \geq 0$, RNI P+1
FAD,I	60	m,b	Floating point addition of (M,M+1) to (A,Q) $\rightarrow$ (A,Q)
FDV,I	63	m,b	Floating point division of (A,Q) by (M,M+1) $\rightarrow$ (A,Q). Remainder with sign extended $\rightarrow$ (E)
FMU,I	62	m,b	Floating point multiplication of (A,Q) and (M,M+1) $\rightarrow$ (A,Q)
FSB,I	61	m,b	Floating point subtraction of (M,M+1) from (A,Q) $\rightarrow$ (A,Q)
HLT	00.0	m	Unconditional stop, RNI m upon restarting
IAI	53.(4+b)4	b	$(A)+(B^b) \rightarrow (B^b)$, sign of B^b extended prior to addition

Mnemonic Code	Octal Code	Address Field	Operation Performed
IAPR	77.57		Interrupt associated processor
IJD	02.4	m	No operation
IJD	02.5-7	m,b	If $(B^b) = 0$,RNI P+1: If $(B^b) \neq 0$, $(B^b) - 1 \rightarrow (B^b)$, RNI m
IJI	02.0	m	No operation
IJI	02.1-3	m,b	If $(B^b) = 0$, RNI P+1: If $(B^b) \neq 0$, $(B^b)+1 \rightarrow (B^b)$, RNI m
INA	15.6	y	Increase (A) by y
INA,S	15.4	y	Increase (A) by y, sign of y is extended
INAC,INT	73.1	ch	(A) is cleared and a 6-bit character is transferred from a peripheral device to the lower 6 bits of A.
INAW,INT	74.1	ch	(A) is cleared and a 12- or 24-bit word is read from a peripheral device into the lower 12 bits or all of A (word size depends on I/O channel).
INCL	77.50	x	Interrupt faults defined by x are cleared.
INI	15.0	y	No operation
INI	15.1-3	y,b	Increase (B^b) by y, signs of y and B^b are extended.
INPC,INT,B,H	73.0	ch,r,s	A 6- or 12-bit character is read from a peripheral device and stored in memory at a given location.
INPW,INT,B,N	74.0	ch,m,n	Word Address is placed in bits 00-14, 12- or 24-bit words are read from a peripheral device and stored in memory.
INQ	15.7	y	Increase (Q) by y.
INQ,S	15.5	y	Increase (Q) by y, sign of y is extended.
INS	77.3	x,ch	Sense internal status. If 1 bits occur on status lines in any of the same positions as 1 bits in the mask, RNI P+1. If no comparison, RNI P+2.

Mnemonic Code	Octal Code	Address Field	Operation Performed
INTS	77.4	x,ch	Sense for interrupt condition; if 1 bits occur simultaneously in interrupt lines and in the interrupt mask, RNI P+1; if not, RNI P+2.
IOCL	77.51	x	Clears I/O channel or search/move control as defined by bits 00-07, 08, and 11 of x.
ISD	10.4	y	If $y = 0$, RNI P+2. If $y \neq 0$, RNI P+1.
ISD	10.5-7	y,b	If $(B^b) = y$, clear B^b and RNI P+2. If $(B^b) \neq y$, $(B^b) - 1 \rightarrow (B^b)$, RNI P+1.
ISE	04.0	y	If $y = 0$, RNI P+2, otherwise RNI P+1.
ISE	04.1-3	y,b	If $y = (B^b)$, RNI P+2, otherwise RNI P+1.
ISG	05.0	y	If $y \geq 0$, RNI P+2, otherwise RNI P+1.
ISG	05.1-3	y,b	If $(B^b) \geq y$, RNI P+2, otherwise RNI P+1.
ISI	10.1-3	y,b	If $(B^b) = y$, clear B^b and RNI P+2. If $(B^b) \neq y$, $(B^b) + 1 \rightarrow (B^b)$, RNI P+1.
LACH	22	r,1	$0 \rightarrow (A), (R) \rightarrow (A_{00-05})$
LCA,I	24	m,b	$(\overline{M}) \rightarrow (A)$
LCAQ,I	26	m,b	$(\overline{M}) \rightarrow A$, $(\overline{M+1}) \rightarrow (Q)$
LDA,I	20	m,b	$(M) \rightarrow A$
LDAQ,I	25	m,b	$(M) \rightarrow A$, $(M+1) \rightarrow (Q)$
LDE	64	r,l	Load E with up to 12 numeric BCD characters from storage. BCD field length specified by contents of D register. (SET, instruction 70.7). Characters are read consecutively from least significant character (at address (R + (D) - 1) until the most significant character (at address R) is in E. E is shifted right as loading progresses. The sign is acquired along with the least significant character.
LDI,I	54	m,b	$(M_{00-14}) \rightarrow (B^b)$

Mnemonic Code	Octal Code	Address Field	Operation Performed
LDL,I	27	m,b	$(M)_\wedge (Q) \rightarrow (A)$
LDQ,I	21	m,b	$(M) \rightarrow (Q)$
LPA,I	37	m,b	$(M)_\wedge (A) \rightarrow (A)$
LQCH	23	r,2	$0 \rightarrow (Q)$, $(R) \rightarrow (Q_{00-05})$
MEQ	06.0-7	m,i	$(B^1) - i \rightarrow (B^1)$; if (B^1) negative, RNI P+1. If (B^1) positive, test $(A) = (Q)_\wedge (M)$; if true, RNI P+2, if false, repeat sequence.
MOVE,INT	72	ℓ,r,s	Move ℓ characters from r to s; $1 \leq \ell \leq 128_{10}$
MTH	07.0-7	m,i	$(B^2) - i \rightarrow (B^2)$; if (B^2) negative, RNI P+1. If (B^2) positive, test $(A) \geq (Q)_\wedge (M)$, if true, RNI P+2; if false, repeat sequence.
MUA,I	50	m,b	$(A)*(M) \rightarrow (Q,A)$
MUAQ,I	56	m,b	$(A,Q)*(M,M+1) \rightarrow (A,Q,E)$
NOP	14.0		No operation (COMPASS assembled NOP)
OTAC,INT	75.1	ch	Character from (A_{00-05}) is sent to peripheral device, (A) retained.
OTAW,INT	76.1	ch	Transfer (A_{00-11}) or (A_{00-23}), depending on type of I/O channel, to peripheral device.
OUTC,INT, B,H	75.0	ch,r,s	Storage words assembled into 6 or 12-bit characters and sent to a peripheral device.
OUTW,INT, B,H	76.0	ch,m,n	12 or 24-bit words transferred from storage to a peripheral device.
PAUS	77.6	x	Sense busy lines. If 1 appears on a line corresponding to 1 bits in x, do not advance p. If p inhibited for longer than 40 msec., read reject from p + 1. If no comparison, RNI p + 2.
QEL	55.5	y	$(Q) \rightarrow (E_L)$
QSE	04.7	y	If $y = (Q_{00-14})$, RNI P+2, otherwise RNI P+1.

Mnemonic Code	Octal Code	Address Field	Operation Performed
QSE,S	04.5	y	If y = (Q), RNI P+2, otherwise RNI P+1. Sign of y is extended.
QSG	05.7	y	If (Q)≥y, RNI P+2, otherwise RNI P+1.
QSG,S	05.5	y	If (Q)≥y, RNI P+2, otherwise RNI P+1. Sign of y is extended.
RAD,I	34	m,b	$(A)+(M)\rightarrow(M)$
RTJ	00.7	m	$(P)+1\rightarrow(m_{00-14})$, RNI m+1
SACH	42	r,2	$(A_{00-05})\rightarrow(R)$
SBA,I	31	m,b	$(A)-(M)\rightarrow(A)$
SBAQ,I	33	m,b	$(A,Q)-(M,M+1)\rightarrow(A,Q)$
SBCD	77.72		BCD fault set to 1
SBE	67	r,3	Up to twleve 4-bit characters (most significant character at address r) is subtracted from E. Difference appears in E. (D) register specifies field length.
SCA,I	36	m,b	Where (M) contains a 1 bit, complement the corresponding bit in (A).
SCAQ	13.4-7	y,b	Shift (A,Q) left end around until upper 2 bits of A are unequal. Residue K = k-shift count. If b = 1, 2, or 3, $K\rightarrow(B^b)$; if b = 0, K is discarded.
SCHA,I	46	m,b	$(A_{00-16})\rightarrow(M_{00-16})$
SCIM	77.53	x	Selectively clear interrupt mask register for each 1 bit in x; corresponding bit in the mask register is set to 0.
SEL	77.1	x,ch	If channel ch is busy, read reject instruction from P+1. If channel ch is not busy, a 12-bit function code is sent on channel ch with a function enable, RNI P+2.
SET	70.7	y	$Y_{00-3}\rightarrow(D)$

Mnemonic Code	Octal Code	Address Field	Operation Performed
SFE	70.0-3	k,b	Shift (E) in one character (4-bit) steps. Left shift: bit 23 = 0, magnitude of shift = lower 4 bits of K = k+(B^b). Right shift: bit 23 = 1, magnitude of shift lower 4 bits of complement of K = k+(B^b).
SFPF	77.71		Set floating point fault.
SHA	12.0-3	k,b	Shift (A). Shift count K = k+(B^b) (signs of k and B^b extended). If bit 23 of K = 1, shift right; complement of lower 6 bits equal shift magnitude. If bit 23 of K = 0, shift left; lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off.
SHAQ	13.0-3	k,b	Shift (A,Q) as one register. Shift count K = k+(B^b) (signs of k and B^b extended). If bit 23 of K = 1, shift right and complement of lower 6 bits equal shift magnitude. If bit 23 of K = 0, shift left; lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off.
SHQ	12.4-7	k,b	Shift (Q), Shift count K = k+(B^b) (signs of k and B^b extended). If bit 23 of K = 1, shift right; complement of lower 6 bits equal shift magnitude. If bit 23 of K = 0, shift left; lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off.
SJ1	00.1	m	If jump key 1 is set, jump to m
SJ2	00.2	m	If jump key 2 is set, jump to m
SJ3	00.3	m	If jump key 3 is set, jump to m
SJ4	00.4	m	If jump key 4 is set, jump to m
SJ5	00.5	m	If jump key 5 is set, jump to m
SJ6	00.6	m	If jump key 6 is set, jump to m
SLS	77.70		Program stops if selective stop switch is on; upon restarting RNI P+1.
SQCH	43	r,l	(Q_{00-05}) → (R)

Mnemonic Code	Octal Code	Address Field	Operation Performed
SRCE,INT	71.0	c,r,s	Search for equality of character c in a list beginning at location r until an equal character is found, or until character location s is reached; $0 \le c \le 63_{10}$.
SRCN,INT	71.1	c,r,s	Inequality search; same as SRCE.
SSA,I	35	m,b	Where (M) contains a 1 bit, set the corresponding bit in A to 1.
SSH	10.0	m	Test sign of (m), shift (m) left one place, end around and replace in storage. If negative sign, RNI P+2; otherwise RNI P+1.
SSIM	77.52	x	Selectively set interrupt mask register for each 1 bit in x. The corresponding bit in the mask register is set to 1.
STA,I	40	m,b	$(A) \rightarrow (M)$
STAQ,I	45	m,b	$(A, Q) \rightarrow (M, M+1)$
STE	65	r,2	Store up to 13 numeric BCD characters from E. Least significant character stored at R+(D) - 1 continuing back to most significant character stored at R.
STI,I	47	m,b	$(B^b) \rightarrow (M_{00-14})$
STQ,I	41	m,b	$(Q) \rightarrow (M)$
SWA,I	44	m,b	$(A_{00-14}) \rightarrow (M_{00-14})$
TAI	53.40-70	b	$(A_{00-14}) \rightarrow (B^b)$; if b = 0 becomes a no operation instruction.
TAM	53.42	v	$(A) \rightarrow (v)$
TIA	53.0-3	b	$0 \rightarrow (A)$, $(B^b) \rightarrow (A_{00-14})$; if b = 0, $0 \rightarrow (A)$.
TIM	53.(4+b)3	v,b	$(B^b) \rightarrow (v_{00-14})$
TMA	53.02	v	$(v) \rightarrow (A)$
TMI	53.(0+b)3	v,b	$(v_{00-14}) \rightarrow (B^b)$
TMQ	53.01	v	$(v) \rightarrow (Q)$

Mnemonic Code	Octal Code	Address Field	Operation Performed
TQM	53.41	v	(Q) → (v)
UCS	77.77		Unconditional stop. Upon restarting RNI P+1
UJP,I	01	m,b,b	Unconditional jump to M
XOA	16.6	y	y V (A) → (A), no sign extension
XOA,S	16.4	y	y V (A) → (A), sign of y is extended
XOI	16.0	y	No operation
XOI	16.1-3	y,b	y V $(B^b) \rightarrow (B^b)$
XOQ	16.7	y	y V (Q) → (Q) no sign extension
XOQ,S	16.5	y	y V (Q) → (Q) sign of y is extended

3300/3500 INSTRUCTION LIST
(executive mode)

Mnemonic Code	Octal Code	Address Field	Operation Performed
ACI†	77.54		$(A_{00-02}) \rightarrow$ channel index register
ADA,I	30	m,b	$(A)+(M) \rightarrow (A)$
ADAQ,I	32	m,b	$(A,Q)+(M,M+1) \rightarrow (A,Q)$
ADM	67	$r,B_r,\ell_r,$ s,B_s,ℓ_s	Add field R to field S → field S
AEU	55.6		$(A) \rightarrow (E_U)$
AIA	53.1-3	b	$(A)+(B^b) \rightarrow (A)$, sign of (B^b) is extended prior to addition.
AIS†	77.664		$(A_{00-02}) \rightarrow$ instruction state register
ANA	17.6	y	$y \wedge (A) \rightarrow (A)$
ANA,S	17.4	y	$y \wedge (A) \rightarrow (A)$, sign of y extended
ANI	17.0	y	No operation
ANI	17.1-3	y,b	$y \wedge (B^b) \rightarrow (B^b)$
ANQ	17.7	y	$y \wedge (Q) \rightarrow (Q)$
ANQ,S	17.5	y	$y \wedge (Q) \rightarrow (Q)$, sign of y extended
AOS†	77.66		$(A_{00-02}) \rightarrow$ operand state register
APF†	77.64	w,2	$(A_{00-11}) \rightarrow$ page file
AQA	53.04		$(A)+(Q) \rightarrow (A)$
AQE	55.7		$(A,Q) \rightarrow (E_U,E_L)$
AQJ,EQ	03.4	m	If $(A) = (Q)$, RNI m, otherwise RNI P+1
AQJ,GE	03.6	m	If $(A) \geq (Q)$ RNI m, otherwise RNI P+1
AQJ,LT	03.7	m	If $(A) < (Q)$, RNI m, otherwise RNI P+1

† In the program state, an attempt to execute instructions indicated by † on the following pages will generate an executive interrupt and the processor will revert to the monitor state.

Mnemonic Code	Octal Code	Address Field	Operation Performed
AQJ,NE	03.5	m	If (A) ≠ (Q), RNI m, otherwise RNI P+1
ASE	04.6	y	If y = (A_{00-14}), RNI P+2, otherwise RNI P+1
ASE,S	04.4	y	If y = (A_{00-14}), RNI P+2, otherwise RNI P+1, sign of y is extended
ASG	05.6	y	If (A)≥y, RNI P+2, otherwise RNI P+1
ASG,S	05.4	y	If (A)≥y, RNI P+2, otherwise RNI P+1, sign of y is extended.
ATD	66	m,B_m,ℓ_m, s,B_s	Translate American Standard Code field M ⟶ BCD character field S
ATD,dc	66	m,B_m,ℓ_m, s,B_s	Translate American Standard Code field M ⟶ BCD character field S with delimiting character possibility.
AZJ,EQ	03.0	m	If (A) = 0, RNI m, otherwise RNI P+1
AZJ,GE	03.2	m	If (A)≥0, RNI m, otherwise RNI P+1
AZJ,LT	03.3	m	If (A) < 0, RNI m, otherwise RNI P+1
AZJ,NE	03.1	m	If (A) ≠ 0, RNI m, otherwise RNI P+1
CIA†	77.55		0 ⟶ (A), then channel index register ⟶ (A_{00-02})
CILO†	77.51	cm	Lockout external interrupt on masked channels, cm, until channel(s) is not busy
CINS†	77.3	ch	Interrupt mask and internal status (A)
CLCA†	77.512	cm	Clear the specified channel(s), but not external equipment
CMP	67	r,B_r,ℓ_r, s,B_s,ℓ_s	Compare field R to field S, exit upon encountering ≠ characters
CMP,dc	67	r,B_r, s,B_s,ℓ_s	Compare field R to field C, exit upon encountering ≠ characters; delimiting character possibility

Mnemonic Code	Octal Code	Address Field	Operation Performed
CON†	77.0	x,ch	If channel ch is busy, reject instruction, RNI P+1. If channel ch is not busy, send 12-bit connect code (x) on channel ch with connect enable, RNI P+2
COPY†	77.2	ch	External status code from I/O channel ch $\rightarrow (A_{00-11})$, (interrupt mask register) $\rightarrow (A_{12-23})$, RNI P+1
CPR,I	52	m,b	(M) > (A), RNI P+1 (Q) > (M), RNI P+2 (A)≥(M)≥(Q), RNI P+3 (A) and (Q) are unchanged
CTI†	77.75		Set console typewriter input Beginning character address must be present in location 23 of register file and last character +1 must be present in location 33 of the file.
CTO†	77.76		Set console typewriter output
CVBD	66	m,B_n,n,B_n	Convert binary field M to BCD → field N
CVDB	66	$r,B_r,\ell_r,$ m,B_m	Convert BCD field R to binary → field M
DINT†	77.73		Disable interrupt control
DTA	66	$r,B_r,\ell_r,$ m,B_m	Translate BCD field R to American Standard Code → field M
DTA,dc	66	$r,B_r,\ell_r,$ m,B_m	Translate BCD field R to American Standard Code → field M; delimiting character possibility
DVA,I	51	m,b	(A,Q)/(M) → (A), remainder → (Q)
DVAQ,I	57	m,b	(A,Q,E)/(M,M+1) → (A,Q), remainder with sign extended → (E)
EAQ	55.3		$(E_U,E_L) \rightarrow (A,Q)$
ECHA	11	r	$0 \rightarrow (A)$, then $r \rightarrow (A_{00-16})$
ECHA,S	11	r	$0 \rightarrow (A)$, then $r \rightarrow (A_{00-16})$, sign extended

Mnemonic Code	Octal Code	Address Field	Operation Performed
EDIT	64	$r, B_r, \ell_r,$ s, B_s, ℓ_s	Field R → field S with COBOL type of editing specified by picture previously stored in field S
EINT†	77.74		Interrupt control enabled; allows one more instruction to be executed before interrupt.
ELQ	55.1		$(E_L) \rightarrow (Q)$
ENA	14.6	y	$0 \rightarrow (A)$, then $y \rightarrow (A_{00-14})$
ENA,S	14.4	y	$0 \rightarrow (A)$, then $y \rightarrow (A_{00-14})$, sign extended
ENI	14.0	y	No operation
ENI	14.1-3	y,b	$0 \rightarrow (B^b)$, then $y \rightarrow (B^b)$
ENQ	14.7	y	$0 \rightarrow (Q)$, then $y \rightarrow (Q_{00-14})$
ENQ,S	14.5	y	$0 \rightarrow (Q)$, then $y \rightarrow (Q_{00-14})$, sign extended
EUA	55.2		$(E_U) \rightarrow (A)$
EXS†	77.2	x,ch	Sense external status. If 1 bits occur on status lines in any of the same positions as 1 bits in the mask, RNI P+1. If no comparison, RNI P+2.
FAD,I	60	m,b	Floating point addition of (M,M+1) to (A,Q) → (A,Q)
FDV,I	63	m,b	Floating point division of (A,Q) by (M,M+1) → (A,Q). Remainder with sign extended → (E).
FMU,I	62	m,b	Floating point multiplication of (A,Q) and (M,M+1) → (A,Q)
FRMT	64	$r, B_r, \ell_r,$ s, B_s, ℓ_s	Move field R → field S: replace leading zeros with blanks; insert a comma after every three characters moved; insert a decimal point in third lowest order position in S field.
FSB,I	61	m,b	Floating point subtraction of (M,M+1) from (A,Q) → (A,Q)
HLT†	00	m	Unconditional stop, RNI m upon restarting

Mnemonic Code	Octal Code	Address Field	Operation Performed
IAI	53.(5-7)4	b	$(A)+(B^b) \rightarrow (B^b)$, sign of B^b is extended prior to addition
IAPR†	77.57		Interrupt associated processor
IJD	02.4	m	No operation
IJD	02.5-7	m,b	If $(B^b) = 0$, RNI P+1: If $(B^b) \neq 0$, $(B^b) - 1 \rightarrow (B^b)$, RNI m
IJI	02.0	m	No operation
IJI	02.1-3	m,b	If $(B^b) = 0$, RNI P+1: If $(B^b) \neq 0$, $(B^b) + 1 \rightarrow (B^b)$, RNI m
INA	15.6	y	Increase (A) by y
INA,S	15.4	y	Increase (A) by y, sign of y is extended
INAC,INT†	73	ch	(A) is cleared and a 6-bit character is transferred from a peripheral device to the lower 6 bits of A.
INAW,INT†	74	ch	(A) is cleared and a 12- or 24-bit word is read from a peripheral device into the lower 12 bits or all of A (word size depends on I/O channel).
INCL†	77.50	x	Interrupt faults defined by x are cleared
INI	15.0	y	No operation
INI	15.1-3	y,b	Increase (B^b) by y, signs of y and B^b extended
INPC,INT,B,H†	73	ch,r,s	A 6- or 12-bit character is read from a peripheral device and stored in memory at a given location.
INPW,INT,B,N†	74	ch,m,n	Word address is placed in bits 00-14, 12- or 24-bit words are read from a peripheral device and stored in memory.
INQ	15.7	y	Increase (Q) by y
INQ,S	15.5	y	Increase (Q) by y, sign of y extended

Mnemonic Code	Octal Code	Address Field	Operation Performed
INS†	77.3	x, ch	Sense internal status. If 1 bits occur on status lines in any of the same positions as 1 bits in the mask, RNI P+1. If no comparison, RNI P+2.
INTS†	77.4	x, ch	Sense for interrupt condition; if 1 bits occur simultaneously in interrupt lines and in the interrupt mask, RNI P+1. If not RNI P+2.
IOCL†	77.51	x	Clears I/O channel or search/move control as defined by bits 00-07, 08, and 11 of x.
ISA†	77.674		$0 \rightarrow (A)$, instruction state register $\rightarrow (A_{00-02})$
ISD	10.4	y	If $y = 0$, RNI P+2. If $y \neq 0$, RNI P+1
ISD	10.5-7	y, b	If $(B^b) = y$, clear B^b and RNI P+2; if $(B^b) \neq y$, $(B^b) - 1 \rightarrow (B^b)$, RNI P+1
ISE	04.0	y	If $y = 0$, RNI P+2, otherwise RNI P+1
ISE	04.1-3	y, b	If $y = (B^b)$, RNI P+2, otherwise RNI P+1
ISG	05.0	y	If $y \geq 0$, RNI P+2, otherwise RNI P+1
ISG	05.1-3	y, b	If $(B^b) \geq y$, RNI P+2, otherwise RNI P+1
ISI	10.1-3	y, b	If $(B^b) = y$, clear B^b and RNI P+2; if $(B^b) \neq y$, $(B^b) + 1 \rightarrow (B^b)$, RNI P+1
JAA†	77.56		Last executed jump address $\rightarrow (A_{00-14})$
JMP, HI	70.0	m	Jump if BDP condition register > 0 or +
JMP, LOW	70.2	m	Jump if BDP condition register < 0 or -
JMP, ZRO	70.1	m	Jump if BDP condition register = 0
LACH	22	r, l	$0 \rightarrow (A), (R) \rightarrow (A_{00-05})$
LBR	70.	m	Load BDP conditions with the contents of m.
LCA, I	24	m, b	(M) (A)
LCAQ, I	26	m, b	(M) (A), (M + 1) (Q)
LDA, I	20	m, b	(M) (A)

Mnemonic Code	Octal Code	Address Field	Operation Performed
LDAQ,I	25	m,b	$(M) \longrightarrow (A)$, $(M + 1) \longrightarrow (Q)$
LDI,I	54	m,b	$(M_{00-14}) \longrightarrow (B^b)$
LDL,I	27	m,b	$(M) \wedge (Q) \longrightarrow (A)$
LDQ,I	21	m,b	$(M) \longrightarrow (Q)$
LPA	37	m,b	$(M) \wedge (A) \longrightarrow (A)$
LQCH	23	r,2	$0 \longrightarrow (Q)$, $(R) \longrightarrow (Q_{00-05})$
MEQ	06	m,i	$(B^1) - i \longrightarrow (B^1)$; if (B^1) negative, RNI P+1. If (B^1) positive, test $(A) = (Q) \wedge (M)$; if true, RNI P+2, if false, repeat sequence.
MOVE,INT[†]	72	ℓ,r,s	Move ℓ characters from r to s; $0 \leq \ell \leq 127_{10}$
MTH	07.0-7	m,i	$(B^2) - i \longrightarrow (B^2)$, if (B^2) negative, RNI P+1. If (B^2) positive, test $(A) \geq (Q) \wedge (M)$, if true, RNI P+2; if false, repeat sequence.
MUA,I	50	m,b	$(A)*(M) \longrightarrow (Q,A)$
MUAQ,I	56	m,b	$(A,Q)*(M,M+1) \longrightarrow (A,Q,E)$
MVBF	64	$r,B_r,\ell_r,$ s,B_s,ℓ_s	Move characters from field R → field S; if field S > Field R, blank fill.
MVE	64	$r,B_r,\ell_r,$ s,B_s,ℓ_s	Move characters from field R → field S according to parameters.
MVE,dc	64	$r,B_r,$ s,B_s,ℓ_s	Move characters from field R → field S. Delimiting character possibility
MVZF	64	$r,B_r,\ell_r,$ s,B_s,ℓ_s	Move characters from field R → field S; if field S > field R, zero fill
MVZS	64	$r,B_r,\ell_r,$ s,B_s,ℓ_s	Move characters from field R → field S; suppress leading zeros
MVZS,dc	64	$r,B_r,$ s,B_s,ℓ_s	Move characters from field R → field S; suppress leading zeros. Delimiting character possibility.

Mnemonic Code	Octal Code	Address Field	Operation Performed
NOP	14.0		No operation (COMPASS assembled NOP)
OSA[†]	77.67		0 ⟶ (A); operand state register ⟶ (A_{00-02})
OTAC,INT[†]	75	ch	Character from (A_{00-05}) is sent to peripheral device, (A) retained.
OTAW,INT[†]	76	ch	Transfers (A_{00-11}) or (A_{00-23}), depending on type of I/O channel, to a peripheral device.
OUTC,INT,B,H[†]	75	ch,r,s	Storage words assembled into 6- or 12-bit characters and sent to a peripheral device
OUTW,INT,B,N[†]	76	ch,m,n	Transfer 12- or 24-bit words from storage to a peripheral device
PAK	66	r,B_r,ℓ_r, m,B_m	Convert and pack a 6-bit numeric BCD field R to a 4-bit numeric BCD field and store the result in field M
PAUS[†]	77.60	x	Sense busy lines. If 1 appears on a line corresponding to 1 bits in x, do not advance P. If P is inhibited for longer than 40 ms, read reject instruction from P+1. If no comparison, RNI P+2.
PFA[†]	77.65	w,2	0 ⟶ (A), then page index file ⟶ (A_{00-11})
PRP[†]	77.61	x	Same as PAUS, except real-time clock cannot increment during the pause
QEL	55.5		$(Q) \longrightarrow (E_L)$
QSE	04.7	y	If $y = (Q_{00-14})$, RNI P+2, otherwise RNI P+1
QSE,S	04.5	y	If y = (Q), RNI P+2, otherwise RNI P+1, sign of y is extended
QSG	05.7	y	If $(Q_{00-14}) \geq y$, RNI P+2, otherwise RNI P+1
QSG,S	05.5	y	If $(Q) \geq y$, RNI P+2, otherwise RNI P+1, sign of y is extended
RAD,I	34	m,b	(M)+(A) ⟶ (M)

Mnemonic Code	Octal Code	Address Field	Operation Performed
RCR[†]	77.634		Subcondition register → condition register
RIS	55.0		Relocate to instruction state
ROS	55.4		Relocate to operand state
RTJ	00.7	m	$(P)+1 \longrightarrow (m_{00-14})$, RNI m+1
SACH	42	r,2	$(A_{00-05}) \longrightarrow (R)$
SBA,I	31	m,b	$(A) - (M) \longrightarrow (A)$
SBAQ,I	33	m,b	$(A,Q) - (M,M+1) \longrightarrow (A,Q)$
SBCD	77.72		Set BCD fault logic
SBJP[†]	77.62		Transfer system from monitor state to program state when next jump occurs
SBM	67	$r,B_r,\ell_r,$ s,B_s,ℓ_s	Subtract field R from field S → field S
SBR	70.	m	Store BDP conditions in m.
SCA,I	36	m,b	Where (M) contains a 1 bit, complement the corresponding bit in (A)
SCAN,LR,EQ,dc	65	r,B_r,ℓ_r,sc	Scan field R from left to right, stop on = condition; delimiting character possibility
SCAN,LR,NE,dc	65	r,B_r,ℓ_r,sc	Scan field R from left to right, stop on ≠ condition; delimiting character possibility
SCAN,RL,EQ,dc	65	r,B_r,ℓ_r,sc	Scan field R from right to left, stop on = condition; delimiting character possibility
SCAN,RL,NE,dc	65	r,B_r,ℓ_r,sc	Scan field R from right to left, stop on ≠ condition; delimiting character possibility
SCAN,LR,EQ	65	r,B_r,ℓ_r,sc	Scan field R from left to right, stop on = condition
SCAN,LR,NE	65	r,B_r,ℓ_r,sc	Scan field R from left to right, stop on ≠ condition

Mnemonic Code	Octal Code	Address Field	Operation Performed
SCAN,RL,EQ	65	r,B_r,ℓ_r,sc	Scan field R from right to left, stop on = condition
SCAN,RL,NE	65	r,B_r,ℓ_r,sc	Scan field R from right to left, stop on ≠ condition
SCAQ	13.4-7	k,b	Shift (A,Q) left end around until upper 2 bits of A are unequal. Residue K = k-shift count. If b = 1,2, or 3, K ⟶ (B^b); if b = 0, K is discarded.
SCHA,I	46	m,b	$(A_{00-16}) \longrightarrow (M_{00-16})$
SCIM,I[†]	77.53	x	Selectively clear interrupt mask register for each 1 bit in x; corresponding bit in the mask register is set to 0.
SDL[†]	77.624		Upon next LDA instruction: 1. (M) ⟶ (A) 2. 77777777 ⟶ (M)
SEL[†]	77.1	x,ch	If channel ch is busy, read reject instruction from P+1. If not busy, send a 12-bit function code on channel ch with a function enable, RNI P+2.
SFPF	77.71		Set floating point fault logic
SHA	12.(0-3)	k,b	Shift (A). Shift count K=k + (B^b) (signs of k and B^b extended). If bit 23 of K=1, shift right; complement of lower 6 bits equals shift magnitude. If bit 23 of K = 0, shift left; lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off.
SHAQ	13.(0-3)	k,b	Shift (A,Q) as one register. Shift count K = k + (B^b) (signs of k and B^b extended). If bit 23 of K = 1, shift right and complement of lower 6 bits equals shift magnitude. If bit 23 of K = 0, shift left and lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off.

Mnemonic Code	Octal Code	Address Field	Operation Performed
SHQ	12.(4-7)	k,b	Shift (Q). Shift count K=k + (B^b) (signs of k and B^b extended). If bit 23 of K = 1, shift right, complement of lower 6 bits equals shift magnitude. If bit 23 of K = 0, shift left, lower 6 bits equal shift magnitude. Left shifts end around; right shifts end off.
SJ1	00.1	m	If jump key 1 is set, jump to m
SJ2	00.3	m	If jump key 2 is set, jump to m
SJ3	00.3	m	If jump key 3 is set, jump to m
SJ4	00.4	m	If jump key 4 is set, jump to m
SJ5	00.5	m	If jump key 5 is set, jump to m
SJ6	00.6	m	If jump key 6 is set, jump to m
SLS†	77.70		Program stops if selective stop switch is on; upon restarting RNI P+1.
SQCH	43	r,l	$(Q_{00-05}) \longrightarrow (R)$
SRA†	77.63		$0 \longrightarrow (A)$; subcondition register $\longrightarrow (A_{00-02})$
SRCE,INT†	71	c,r,s	Search for equality of character c in list beginning at r until an equal character is found, or until character at s is reached; $0 \leq c \leq 63_{10}$
SRCN,INT†	71	c,r,s	Inequality search; same as SRCE
SSA,I	35	m,b	Where (M) contains a 1 bit, set the corresponding bit in A to 1.
SSH	10.0	m	Test sign of (m), shift (m) left one place, end around and replace in storage. Negative sign, RNI P + 2; otherwise RNI P+1.
SSIM†	77.52	x	Selectively set interrupt mask register for each 1 bit in x. Corresponding bit in the mask register is set to 1.
STA,I	40	m,b	$(A) \longrightarrow (M)$
STAQ,I	45	m,b	$(A,Q) \longrightarrow (M, M+1)$
STI,I	47	m,b	$(B^b) \longrightarrow (M_{00-14})$

Mnemonic Code	Octal Code	Address Field	Operation Performed
STQ,I	41	m,b	$(Q) \rightarrow (M)$
SWA,I	44	m,b	$(A_{00-14}) \rightarrow (M_{00-14})$
TAI	53.4-7	b	$(A_{00-14}) \rightarrow (B^b)$; becomes a no-operation instruction if b = 0.
TAM†	53.42	v	$(A) \rightarrow (v)$
TIA	53.0-3	b	$0 \rightarrow (A), (B^b) \rightarrow (A_{00-14})$; if b = 0, $0 \rightarrow (A)$.
TIM†	53.(4-7)3	v,b	$(B^b) \rightarrow (v_{00-14})$
TMA	53.02	v	$(v) \rightarrow (A)$
TMAV†	77.61		Initiate memory request. If reply occurs within 5 usec., RNI P+2; if not RNI P+1. Storage address is (B^b) with (operand state register) or zero appended.
TMI	53.(0-4)3	v,b	$(v_{00-14}) \rightarrow B^b$
TMQ	53.0	v	$(v) \rightarrow (Q)$
TQM†	53.41	v	$(Q) \rightarrow (v)$
TST	67	r, B_r, ℓ_r	Test field R; -, 0, or +
UCS†	77.77		Unconditional stop.
UJP,I	01	m	Unconditional jump to M.
UPAK	66	m, B_m, s B_s, ℓ_s	Unpack 4-bit BCD field M into 6-bit BCD field S
XOA	16.6	y	$y \vee (A) \rightarrow (A)$
XOA,S	16.4	y	$y \vee (A) \rightarrow (A)$, sign of y is extended
XOI	16.0	y	No operation
XOI	16.1-3	y,b	$y \vee (B^b) \rightarrow (B^b)$
XOQ	16.7	y	$y \vee (Q) \rightarrow (Q)$
XOQ,S	16.5	y	$y \vee (Q) \rightarrow (Q)$, sign of y extended
ZADM	67	r, B_r, ℓ_r, s, B_s, ℓ_s	Clear field S; field R — field S, right justify

3100/3200/3300/3500
COMPATIBLE INSTRUCTION LIST††

Mnemonic Code	Address Field	Mnemonic Code	Address Field	Mnemonic Code	Address Field
ADA,I	m,b	CPR,I	m,b	IJD	m,b
ADAQ,I	m,b	CTI†		IJI	m
AEU		CTO†		IJI	m,b
AIA		DINT†		INA	y
ANA	y	DVA,I	m,b	INA,S	y
ANA,S	y	DVAQ,I	m,b	INAC,INT†	ch
ANI	y	EAQ		INAW,INT†	ch
ANI	y,b	ECHA	r	INCL†	x
ANQ	y	ECHA,S	r	INI	y
ANQ,S	y	EINT†		INI	y,b
AQA		ELQ		INPC,INT,B,H†	ch,r,s
AQE		ENA	y	INPW,INT,B,N†	ch,m,n
AQJ,EQ	m	ENA,S	y	INQ	y
AQJ,GE	m	ENI	y	INQ,S	y
AQJ,LT	m	ENI	y,b	INS†	x,ch
AQJ,NE	m	ENQ	y	INTS †	c,ch
ASE	y	ENQ,S	y	IOCL †	x
ASE,S	y	EUA		ISD	y
ASG	y	EXS†	x,ch	ISD	y,b
ASG,S	y	FAD,I	m,b	ISE	y
AZJ,EQ	m	FDV,I	m,b	ISE	y,b
AZJ,GE	m	FMU,I	m,b	ISG	y
AZJ,LT	m	FSB,I	m,b	ISG	y,b
AZJ,NE	m	HLT †	m	ISI	y,b
CINS†	ch	IAI	b	LACH	r,1
CON†	x,ch	IAPR†		LCA,I	m,b
COPY†	ch	IJD	m	LCAQ,I	m,b

† When the 3300/3500 is operating in the program state of executive mode, an attempt to execute instructions indicated by † will generate an executive interrupt and the processor will revert to the monitor state.

†† These instructions may be used on 3100/3200 or 3300/3500 in either the non-executive or executive mode.

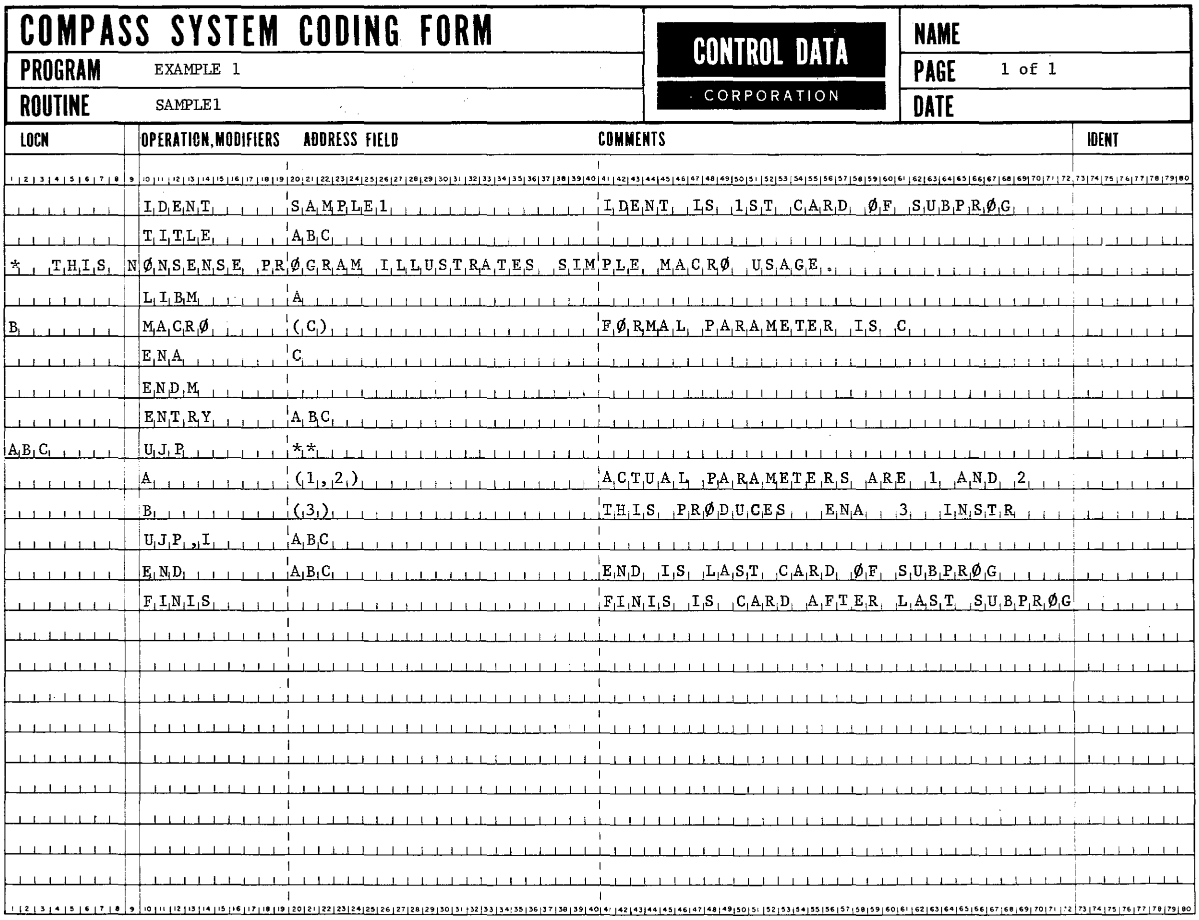
3100/3200/3300/3500
COMPATIBLE INSTRUCTION LIST

Mnemonic Code	Address Field	Mnemonic Code	Address Field	Mnemonic Code	Address Field
LDA,I	m,b	RTJ	m	SSA,I	m,b
LDAQ,I	m,b	SACH	r,2	SSH	m
LDI,I	m,b	SBA,I	m,b	SSIM†	x
LDL,I	m,b	SBAQ,I	m,b	STA,I	m,b
LDQ,I	m,b	SBCD		STAQ,I	m,b
LPA,I	m,b	SCA,I	m,b	STI,I	m,b
LQCH	r,2	SCAQ	y,b	STQ,I	m,b
MEQ	m,i	SCHA,I	m,b	SWA,I	m,b
MOVE,INT†	ℓ,r,s	SCIM,I †	x	TAI	b
MTH	m,i	SEL†	x,ch	TAM†	v
MUA,I	m,b	SFPF		TIA	b
MUAQ,I	m,b	SHA	k,b	TIM†	v,b
NOP		SHAQ	k,b	TMA	v
OTAC,INT †	ch	SHQ	k,b	TMI	v,b
OTAW,INT †	ch	SJ1	m	TMQ	v
OUTC,INT,B,H†	ch,r,s	SJ2	m	TQM†	v
OUTW,INT,B,N †	ch,m,n	SJ3	m	UCS †	
PAUS†	x	SJ4	m	UJP,I	m,b
QEL		SJ5	m	XOA	y
QSE	y	SJ6	m	XOA,S	y
QSE,S	y	SLS †		XOI	y
QSG	y	SQCH	r,1	XOI	y,b
QSG,S	y	SRCE,INT	c,r,s	XOQ	y
RAD,I	m,b	SRCN,INT	c,r,s	XOQ,S	y

PSEUDO INSTRUCTIONS INDEX B

Pseudo Instruction	Meaning or Use	Section
IFN	Assemble following lines of code if the first entry in the address field is non-zero	2.8.2
IFT	Assemble following lines of macro prototype code if the first two entries in the address field are alike	3.2.1
IFZ	Assemble following lines of code if the first entry in the address field is zero	2.8.1
LIBM	Instructs COMPASS to call a library macro from the system library	3.6
LIST	Resume output listing	2.9.3
MACRO	Names a macro and declares the formal parameters used in the prototype	3.1
NOLIST	Suppress output listing	2.9.2
OCT	Expresses constants as signed or unsigned octal integers	2.6.1
ORGR	Controls the relocatable address for storage of instructions, constants, or the reservation of space in PRG, DATA, or COMMON	2.2.4
PRG	Establishes the subprogram location counter during assembly	2.2.1
REM	Print the following remarks on the output listing	2.9.1
SPACE	Indicates line spacing for output listing	2.9.4
TITLE	Print title at top of each page of output listing	2.9.6
VFD	Enter octal numbers, character codes, relocatable addresses, or constants into variable length fields	2.7

COMPASS PROGRAM EXAMPLES C



COMPASS SYSTEM CODING FORM

CONTROL DATA CORPORATION

PROGRAM EXAMPLE 1

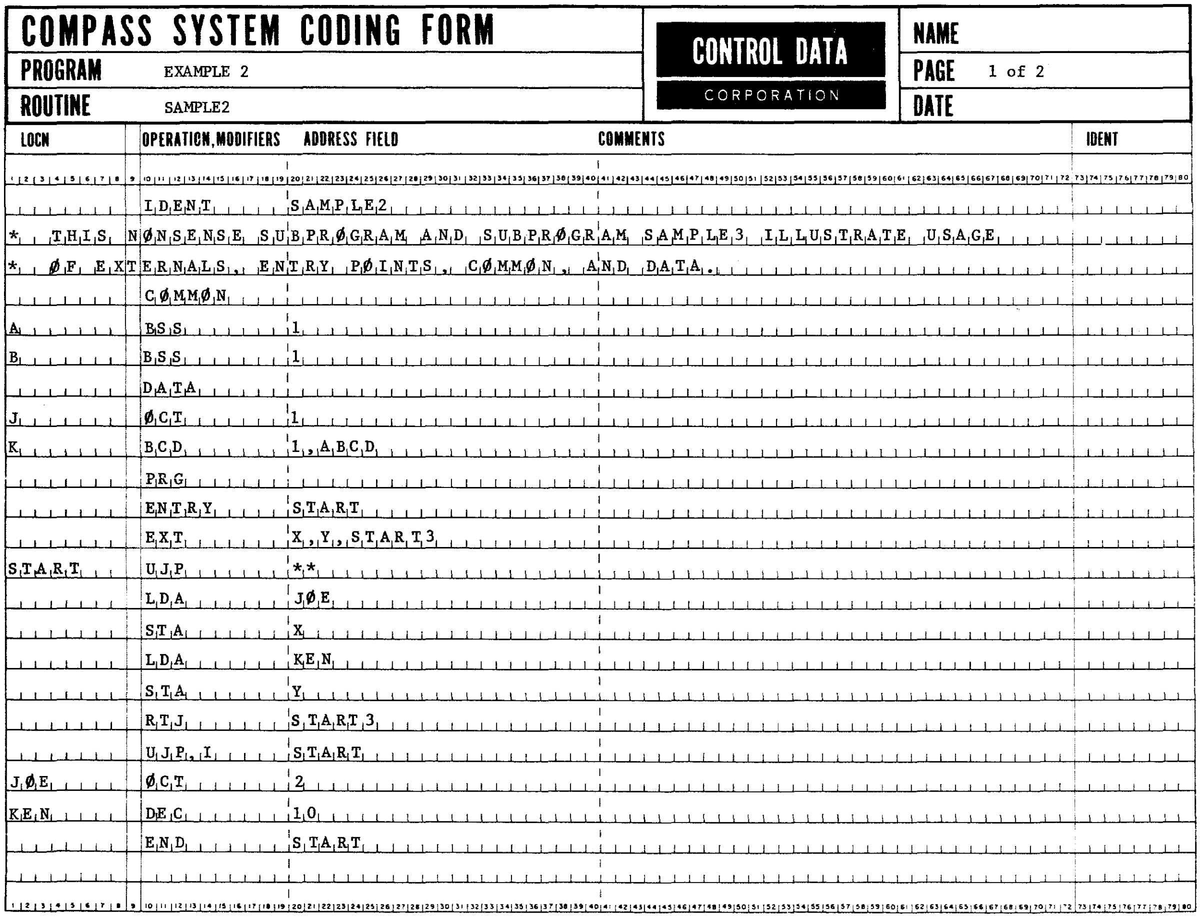
ROUTINE SAMPLE1

NAME

PAGE 1 of 1

DATE

```
LOCN     OPERATION,MODIFIERS  ADDRESS FIELD        COMMENTS                                IDENT
         IDENT     SAMPLE1                         IDENT IS 1ST CARD ØF SUBPRØG
         TITLE     ABC
*  THIS NØNSENSE PRØGRAM ILLUSTRATES SIMPLE MACRØ USAGE.
         LIBM      A
B        MACRØ     (C)                             FØRMAL PARAMETER IS C
         ENA       C
         ENDM
         ENTRY     ABC
ABC      UJP       **
         A         (1,2)                           ACTUAL PARAMETERS ARE 1 AND 2
         B         (3)                             THIS PRØDUCES  ENA  3  INSTR
         UJP,I     ABC
         END       ABC                             END IS LAST CARD ØF SUBPRØG
         FINIS                                     FINIS IS CARD AFTER LAST SUBPRØG
```

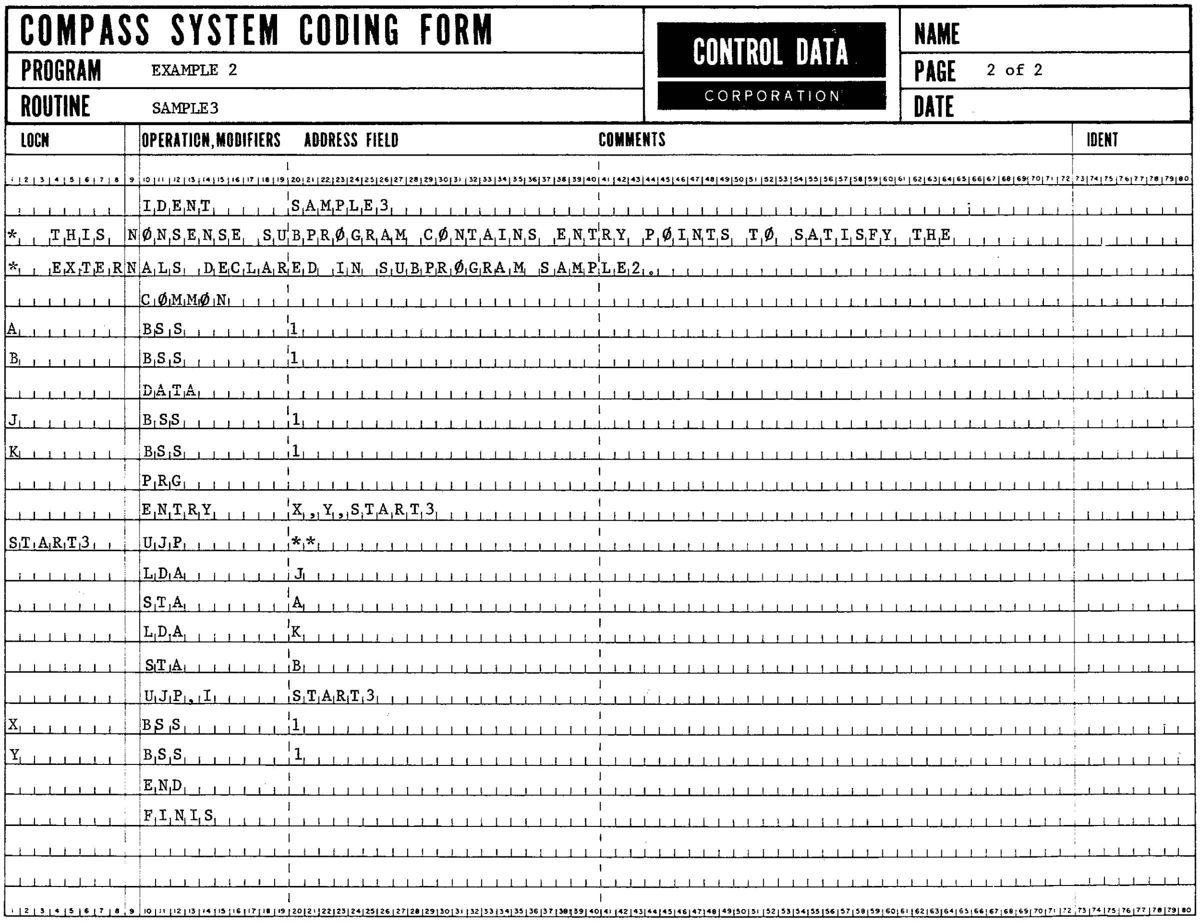


COMPASS SYSTEM CODING FORM

CONTROL DATA CORPORATION

NAME	
PROGRAM	EXAMPLE 2
PAGE	1 of 2
ROUTINE	SAMPLE2
DATE	

```
LOCN     OPERATION,MODIFIERS                  COMMENTS                        IDENT
         ADDRESS FIELD
         IDENT     SAMPLE2
*  THIS NØNSENSE SUBPRØGRAM AND SUBPRØGRAM SAMPLE3 ILLUSTRATE USAGE
*  ØF EXTERNALS, ENTRY PØINTS, CØMMØN, AND DATA.
         CØMMØN
A        BSS       1
B        BSS       1
         DATA
J        ØCT       1
K        BCD       1,ABCD
         PRG
         ENTRY     START
         EXT       X,Y,START3
START    UJP       **
         LDA       JØE
         STA       X
         LDA       KEN
         STA       Y
         RTJ       START3
         UJP,I     START
JØE      ØCT       2
KEN      DEC       10
         END       START
```



COMPASS SYSTEM CODING FORM

CONTROL DATA CORPORATION

PROGRAM EXAMPLE 2

ROUTINE SAMPLE3

NAME

PAGE 2 of 2

DATE

LOCN | OPERATION,MODIFIERS | ADDRESS FIELD | COMMENTS | IDENT

```
         IDENT     SAMPLE3
*  THIS NØNSENSE SUBPRØGRAM CØNTAINS ENTRY PØINTS TØ SATISFY THE
*  EXTERNALS DECLARED IN SUBPRØGRAM SAMPLE2.
         CØMMØN
A        BSS       1
B        BSS       1
         DATA
J        BSS       1
K        BSS       1
         PRG
         ENTRY     X,Y,START3
START3   UJP       **
         LDA       J
         STA       A
         LDA       K
         STA       B
         UJP,I     START3
X        BSS       1
Y        BSS       1
         END
         FINIS
```

INDEX

CONTROL DATA

CORPORATION

COMMENT AND EVALUATION SHEET

3100/3200/3300/3500 Computer Systems
Compatible COMPASS Language Reference Manual

Pub. No. 60174000,A April, 1967

THIS FORM IS NOT INTENDED TO BE USED AS AN ORDER BLANK. YOUR EVALUATION OF THIS MANUAL WILL BE WELCOMED BY CONTROL DATA CORPORATION. ANY ERRORS, SUGGESTED ADDITIONS OR DELETIONS, OR GENERAL COMMENTS MAY BE MADE BELOW. PLEASE INCLUDE PAGE NUMBER REFERENCE.

FROM NAME : ______________________________

BUSINESS
ADDRESS : ______________________________

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Documentation Department
3145 PORTER DRIVE
PALO ALTO, CALIFORNIA

FOLD FOLD

STAPLE STAPLE

CONTROL DATA
CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO., MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

Pub. No. 60174000 A

Litho in U.S.A.