CONTROL DATA 3600 COMPUTER

COMPASS/REFERENCE MANUAL

PRELIMINARY

COMPASS/REFERENCE MANUAL

PRELIMINARY

CONTROL DATA CORPORATION
8100 34th Avenue South
Minneapolis 20, Minnesota

CONTROL DATA 3600 COMPUTER

PREFACE

COMPASS is the comprehensive assembly system for the *Control Data** 3600 Computing System. It provides 3600 users with a convenient means for writing machine language programs. COMPASS operates within SCOPE, the 3600 system for supervisory control of program execution. The SCOPE reference manual is *Control Data* publication number 533.

The purpose of this document is to describe the COMPASS language. Details of the assembly process are presented only when needed to clarify the functions of some of the statements in the language. Programs written in 1604 CODAP-1 language are acceptable as input to COMPASS.

It is assumed that the reader has an understanding of the 3600 instruction repertoire described in *Control Data* Publication No. 213.

*Registered trademark Control Data Corporation.

This is a major revision which obsoletes publication 525. Any comments concerning this manual should be addressed to:

CONTROL DATA CORPORATION
Documentation and Evaluation Department
3330 HILLVIEW AVENUE
PALO ALTO, CALIFORNIA

August, 1963
Pub. No. 525a

CONTENTS

LIST OF TABLES

DESCRIPTION 1

The COMPASS assembly program for the 3600 accepts as input, cards or card images containing symbolic 3600 programming instructions. COMPASS translates the symbolic instructions into 3600 machine language programs. These are prepared in a special format, called relocatable binary, for loading into any portion of memory at run time. The assembler can produce as output any combination of:

1. Output listing of the assembled program.

2. Relocatable binary card output for subsequent loading and execution of the assembled program.

3. Relocatable binary card images on a load-and-go tape for immediate loading and execution of the assembled program.

4. Compressed symbolic output deck to be used as input for subsequent modification and reassembly.

1.1 SUBPROGRAM

A program is composed of one or more independently assembled subprograms. Each subprogram must be headed by an IDENT pseudo instruction and terminated by an END pseudo instruction; it must be wholly contained in one bank of memory. Subprograms communicate with each other by use of entry points and external symbols. (See ENTRY and EXT pseudo instructions.) If subprogram 1 references the symbol ABLE which occurs in subprogram 2, ABLE must be declared as an external symbol in subprogram 1, and an entry point in subprogram 2.

Any subprogram may be the main subprogram -- the subprogram to which initial control is given. One, and only one, subprogram must close with an END pseudo instruction which contains the name of the program transfer address, an entry point in the main subprogram. When the assembled program has been loaded by the SCOPE loader, a bank return jump will be made to this transfer address. If no transfer address has been specified, the loader will terminate the job. To return control to SCOPE at the end of the program, the last executable instruction should be either a jump to this entry point or the EXIT system macro (see SCOPE Reference Manual).

1.2

DATA STORAGE

Data storage may be local (BES and BSS pseudo instructions) or non-local to a subprogram. Non-local data storage may be shared by two or more subprograms, each of which identifies that storage by BLOCK and COMMON pseudo instructions. Each block of common must fit within one memory bank. The two types of common are numbered and labeled; data may be assembled into labeled common but not into numbered common.

1.3

BANK ASSIGNMENT

Subprograms and common blocks may be assigned to any memory bank or combination of memory banks. If the programmer does not designate the bank assignments (see BANK pseudo instruction), the SCOPE loader assigns each subprogram or block in turn to the memory bank in which it most tightly fits.

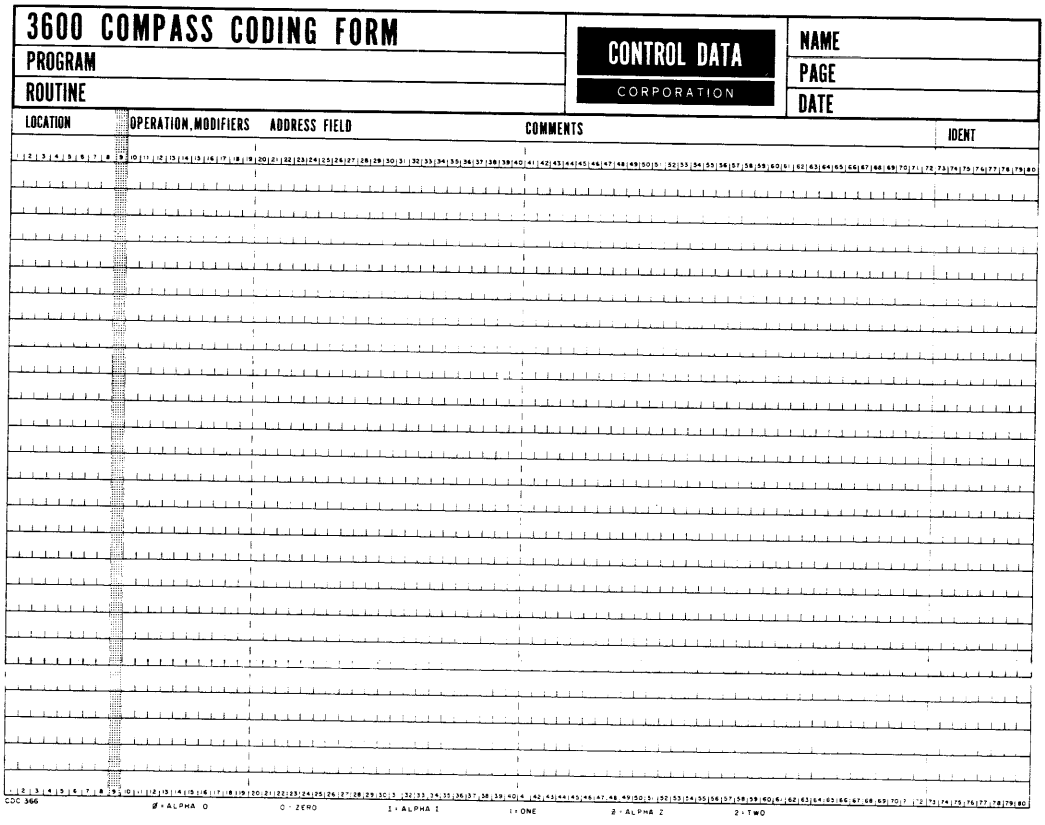
1.4

RUNNING COMPASS

Any number of subprograms may be assembled together. Ahead of the first subprogram is a control card, required by the SCOPE operating system, calling COMPASS and indicating certain assembly options. The SCOPE pseudo instruction or an end-of-file follows the last subprogram.

INSTRUCTION FORMAT 2

Input to COMPASS is in the form of Hollerith characters punched on 80-column cards or images of these cards on magnetic tape. Cards are punched from coding lines written by a programmer on the COMPASS Coding Form. Input format on the coding form is in terms of the columns of a card.



A coding line is divided into four fields: location field, L; operation field, O; address field, A; and comments field, C.

The location field covers columns 1 through 8. Column 9 must be blank in COMPASS coding lines; a non-blank character in column 9 signals a CODAP-1 line (see CODAP pseudo instruction). The operation field begins in column 10 and ends at the first blank column. The address field

follows and must start before column 41. More than one blank column may separate the address field from the operation field. Column 20 is a convenient starting point for the address field for all but a few instructions. The address field terminates at the first blank column or at column 72. The columns remaining through 72 are treated as comments. Column 41 may begin the comments field if there is no address field. Columns 73-80 may be used for identification, but are not printed on the output listing.

2.1 LOCATION FIELD

The location field may be blank, or may contain one of the following symbol types:

Type 1. A symbol containing 1 to 8 non-blank characters; the first is alphabetic, and the rest alphabetic, numeric or period.

Type 2. A plus (+) anywhere in the field and blanks in the remaining columns.

Type 3. A minus (-) anywhere in the field and blanks in the remaining columns.

Type 4. A symbol consisting of 8 characters all of which must be either numeric or blank. This type is used only for naming numbered common blocks.

2.2 OPERATION FIELD

The operation field may consist of one or more subfields separated by commas. The first subfield may contain one of the following:

One of the machine instruction mnemonics listed in Table 4.

One of the pseudo instructions listed in Table 5.

The name of a macro instruction.

One of the octal numbers from 0 to 77.

Succeeding subfields may contain the operation modifiers listed in Table 3 that are permitted to the instruction.

A blank in column 10 terminates the operation field and specifies an operation code of zero.

2.3 ADDRESS FIELD

The address field of an instruction may contain one or more of the 15 subfields listed in Table 8; however, only a few are meaningful in any one instruction. Table 4 lists those subfields permitted in the machine instructions; Chapter 4 gives the subfields of pseudo instructions and macro instructions. Subfields are separated from one another by commas or parentheses.

If the address field is blank, each of its implicit subfields assumes the value zero. An individual subfield may be skipped and assigned the value zero by giving only its trailing comma or, if it is the last subfield in the address field, by omitting both its value and the preceding comma. Skipping a bank subfield enclosed in parentheses gives it the value zero only if the instruction contains a bank usage bit; otherwise, as in XMIT and IOTR, the relocatable bank term $ is assumed (see Special Element $).

An address subfield may be defined by an arithmetic expression or by one of four special elements.

2.3.1 EXPRESSION

An expression may contain a Type 1 symbol, a constant less than 2^{15}, or the element *, or any series of these joined by the operators + - * / .

Type 1 Symbol

A Type 1 symbol is called relocatable if it is one of the following:

1) external symbol

2) location within the subprogram, called program relocatable

3) location within a common block, called common relocatable

Constant

A constant is a decimal or octal integer. It is interpreted as octal if it is suffixed by the character B; otherwise it is interpreted as decimal.

*

The element * is interpreted in one of two ways depending on the subfield in which it occurs. In the 15-bit subfields m, n, and y, its value is the current value of the location counter, that is, the relocatable address of the current instruction. In the 3-bit subfields a and i, it is a bank relocatable element and implies the bank into which the subprogram will be loaded.

The four operations permitted in an arithmetic expression are: addition (+), subtraction (-), multiplication (*), and division (/). The expression is evaluated from left to right, performing all multiplications and divisions and then all additions and subtractions. Use of parentheses for grouping is not permitted: 15 * A + 5/2 * C - 5 is evaluated: $(15 \cdot A) + ((\frac{5}{2}) \cdot C) - 5$

The following rules apply to expressions occurring in address subfields:

a. In a single term, a relocatable symbol can not be an operand in a divide operation, and no slashes (/) may occur to the right of a relocatable symbol.

b. In a divide operation, only the integer portion of the quotient is retained.

In an m, n, w, or y subfield expression the following rules apply:

a. The expression is evaluated modulo 2^{15} -1.

b. In a single expression, all relocatable symbols must be of the same kind: program relocatable, common relocatable (array names in one block), or external (one external symbol). The only exception permitted is that program or common relocatable symbols may be matched algebraically to form constants (P1 - P2 = k or C1 - C2 = k).

c. If each relocatable symbol in an expression is replaced by R, the algebraic sum of the terms in which R appears must be R, -R, or zero. If the sum is zero, the value of the expression is said to be fixed.

An a or i subfield may contain an expression, evaluated modulo 8, or a single bank relocatable term (* or $).

Expressions in all other subfields must not result in a relocatable value; expressions are evaluated modulo 2^n, where n is the number of bits in the machine word occupied by the value of the subfield.

2.3.2 SPECIAL ELEMENTS

The four special elements permitted singly in an address subfield are the following:

**

The combination ** gives the value one to each bit occupied by the subfield in the machine word, for instance, an m subfield 77777. The element normally indicates a subfield to be set during program execution.

Literal

If the operand of an instruction is a single or double precision constant, it may be written as a literal. The literal is written in the form = MV. The equal sign identifies the symbol as a literal, M specifies the mode of the value, and V is a constant which specifies the value. COMPASS assigns the literal value of the constant to one (single precision) or two (double precision) words, and assembles the location of the value into the address portion of the instruction in which the literal appears. Literals of the same value occurring more than once are not duplicated.

The mode designator, M, may be one of the following:

D Decimal constant. V is written in the format specified by the DEC or DECD pseudo instructions. The value is terminated by the first blank character, or comma if another subfield follows.

O Octal constant. V is written in the format specified by the OCT pseudo instruction. The value is terminated by the first blank character, or comma if another subfield follows.

H Hollerith codes. The first eight characters following H specify the BCD value; the ninth character must be a comma or blank.

T Typewriter codes. The first eight typewriter characters following T specify the value; the ninth character must be a comma or blank.

Mode designators D and O may be prefixed by the letter D to indicate a double precision literal.

Symbol assignment

A subfield may consist of a symbol assignment, of the form = S <symbol name > for single precision and =DS <symbol name > for double precision assignment. COMPASS reserves one (single precision) or two (double precision) machine words for each symbol, and assembles the location of the reserved words into the address portion of the instructions in which the symbol assignment appears.

$

Subfields a and i may consist of the special element $, appearing either alone or followed by an external symbol or common array name. If the element appears alone, it implies the bank associated with the name (external symbol or common array) in the operand address subfield. When such a name follows $, the element implies the bank associated with that name. (See BANK pseudo instruction for bank assignment procedures.)

2.3.3 COMMENTS FIELD

The comments field begins with the first column after the blank column terminating the address field, or with column 41 if there is no address field, and ends at column 72. The comments field is ignored by the assembler, but printed on the output listing.

INSTRUCTION PAIRING 3

Because some instructions are 24-bits (half-word) and others are 48-bits (full-word), some precautions should be kept in mind when writing a program for the 3600 system.

3.1 FORCING UPPER

An instruction is forced to begin in a new location and the preceding lower half word, if unused, is padded with a NOP instruction when one of the following conditions occurs:

a. The instruction has a Type 1 or Type 2 location symbol.

b. The instruction is a full word, 48-bits.

c. The instruction is one of the following 24-bit half-words: CPJ, DRJ, EQS, ISK, MEQ, MPJ, MTH, SSH, SSK, THS.

d. The pseudo instruction is one of the following: BCD, BES, BSS, DEC, DECD, OCT, ORGR, TYPE.

e. The instruction immediately follows a BLOCK, EJECT, or REM pseudo instruction.

3.2 FORCING LOWER

An instruction is forced into the lower half of a computer word and the upper half of that word, if unused, is padded with a NOP instruction if a Type 3 location symbol occurs with any 24-bit instruction including those in condition c. above. If a Type 3 location symbol is used with a 48-bit instruction, it will be ignored and flagged as an error on the output listing.

3.3 AUGMENTS

The single precision augment will automatically be inserted before a 24-bit instruction making it a full-word instruction if modifiers are appended to the mnemonic operation code, if a bank designator appears in parentheses in the address field, or if a second index register, v, appears in the address field.

The double precision augment instruction is automatically inserted for instructions DFAD, DFDV, DFMU, DFSB, DLDA, DSTA.

3.4 MNEMONICS

All 3-letter mnemonics are normally half-word instructions unless one of the above conditions requires the insertion of the single precision augment. All mnemonics of more than 3 letters are full-word instructions. Table 4 lists the mnemonic codes for the machine instructions with the allowable modifiers; they may appear in arbitrary order. Modifiers which are mutually exclusive are listed in a vertical column and modifiers may be omitted except as noted. In the address field, the subfields must appear in the order specified. The bank designators (a) and (i) are optional. The symbology in Table 4 is defined in Tables 1, 2, 3 and 8.

PSEUDO INSTRUCTIONS 4

Pseudo instructions are the non-machine language instructions used in preparing a subprogram for COMPASS assembly. Some of the pseudo instructions provide required information to COMPASS. Others are programmer aids for defining portions of a program. The pseudo instructions are grouped according to function.

4.1 SUBPROGRAM LINKAGE

These instructions define the name, the beginning, and end of a subprogram. They also define the subprograms which will be called during execution of the subprogram, and the banks to which subprograms and common blocks are assigned.

IDENT

	L	O	A
		IDENT	m

IDENT must be the first card of each subprogram, if it appears anywhere else it is flagged as an O error. The address field must contain a Type 1 symbol, the name of the subprogram. This name and the length of the subprogram will appear in the first card in the relocatable binary deck. A non-blank location field is meaningless.

END

	L	O	A
		END	m

END signals the end of a subprogram and causes a TRA card to be produced as the last card in the relocatable binary deck. A Type 1 symbol in the address field will appear on the TRA card as the symbolic transfer address. A non-blank location field is meaningless.

ENTRY L O A

ENTRY $m_1,m_2,m_3, \ldots ,m_n$

The address field consists of one or more subfields separated by commas, naming locations within the subprogram, entry points, which may be referenced by other subprograms. Each subfield consists of a Type 1 symbol which must be defined within the subprogram. These symbols are entered into EPT cards for the SCOPE loader. A non-blank location field is meaningless.

EXT L O A

EXT $m_1,m_2,m_3, \ldots ,m_n$

The form of the EXT address field is identical to that of ENTRY. The symbols represent entry point names of subprograms called by this subprogram. The names appearing in the address field are entered into EXT cards for the SCOPE loader. A non-blank location field is meaningless.

In the output listing of the subprogram where these symbols appear in address fields, the machine language address will contain the location of a previous reference to that external symbol. In the column preceding this octal value, an X will appear.

BANK L O A

BANK $(a_1),name_{11},name_{12}, \ldots ,(a_2),name_{21},name_{22}, \ldots$

BANK declares to which memory banks subprograms and common blocks should be assigned at load time. COMPASS produces ahead of the IDC card for the SCOPE loader a control card containing the bank declarations.

There are two kinds of address subfields. The first kind, (a), always enclosed in parentheses, designates a bank in one of three ways:

1. a digit in the range $0 \leq a \leq 7$, (n)

2. an entry point name, (entry)

3. a common block name enclosed in slashes, (/block/)

When the designator is a name, it specifies the bank to which the subprogram containing the entry point is assigned, or the bank to which the common block is assigned. These names are assigned banks either by other BANK declaration or by the SCOPE loader.

The second kind of subfield names a subprogram or common block assigned to the bank given by the preceding designator. Several names may follow a single designator. Common block names are enclosed in slashes.

The entry point and common block names must be defined when the program is loaded. However, they need not be defined or referenced in the subprogram containing the BANK pseudo instruction.

A non-blank location field is meaningless.

4.2 DATA STORAGE

These pseudo instructions allocate data storage which is local (BSS and BES) and non-local. Non-local storage is common to more than one subprogram - (BLOCK, COMMON).

BSS	L	O	A
	ℓ	BSS	m

BSS reserves a local block of consecutive addresses and assigns the location symbol to the first location of the block. The location field may be blank. The address field specifies the number of locations to be reserved. Symbols in the address field expression must be previously defined. A negative address field is considered an error and the pseudo instruction will be ignored. A zero address field leaves no space but forces the next instruction upper before assigning the location symbol.

BES

L	O	A
ℓ	BES	m

BES is identical to BSS, except that the location symbol, if present, will be assigned to the last location of the block.

BLOCK

L	O	A
ℓ	BLOCK	m

This pseudo instruction defines a block of common. The name of the block must be given in the location field by Type 1 symbol which identifies the block as labeled common or a Type 4 symbol which identifies the block as numbered common. Each block must have a unique name. If two or more blocks have the same name, a D error will be indicated on the output listing in each line where the duplicate symbol occurs. BLOCK forces the next subprogram instruction upper.

The length of the block may be specified by an expression in the address field; symbols must have been previously defined. The value of the expression must be greater than or equal to the total length expressed in the COMMON pseudo instructions which follow this BLOCK pseudo instruction. If the address field is blank or zero, the length of the block is determined by the sum of the array sizes within the block. The last numbered common block defined in a bank may vary in length from one subprogram to another.

COMMON

L	O	A
	COMMON	$A_1(i_1,j_1,k_1),A_2(i_2,j_2,k_2), \ldots ,A_n(i_n,j_n,k_n)$

COMMON defines the arrays to be included in the common block defined by the last encountered BLOCK. A non-blank location field is meaningless. The address field consists of one or more subfields, each of which defines an array to be included in the block. A subfield is terminated by a comma; the field is terminated by a blank.

The general form of the address field is:

A(i,j,k),B(l,m,n), . . .

where i, j, k, are the dimensions of the array, i varying most rapidly, j varying next most rapidly, and so on. An array is restricted to a maximum of three dimensions; i, j, k, l, m, and n must be integer constants. A 2-dimension array has two subscripts. A 1-dimension array has only one subscript. For a single element, no parenthetical term should appear.

Example:

COMMON A(15,15),B(3,4,5),C,D(15)

The assembler will sum the expressed sizes of the arrays for all the common in one block. This sum will be the total number of computer words reserved for the block. If the address expression of the BLOCK pseudo instruction gives a number larger than this sum, that number will be the number of words reserved for the block. The first element of the first array in the block will occupy the first word of the reserved area.

Where a reference to a common array appears in an address field, a C is printed in the output listing in the column preceding the octal equivalent of the address portion.

4.3 DATA DEFINITION

Data definition pseudo instructions cause data to be assembled into the subprogram or into a common block. (See ORGR)

OCT

	L	O	A
	ℓ	OCT	$m_1,m_2, \ldots ,m_n$

OCT inserts octal constants into consecutive machine words. A location symbol is optional; if present, it will be assigned to the first word. The address field consists of one or more consecutive subfields separated by commas. Each subfield specifies one constant as a sign (+ or - or none), followed by up to 16 octal digits. Each constant is assigned to a separate word.

DEC

L	O	A
ℓ	DEC	$d_1,d_2,d_3, \ldots ,d_n$

DEC inserts decimal constants into consecutive machine words. Each constant occupies a full computer word. A location symbol is optional; if present, it will be assigned to the first word. The address field consists of one or more consecutive subfields. Each subfield is a decimal constant consisting of a sign (no sign is assumed the same as +), a value of up to 14 decimal digits, a decimal scaling consisting of a D followed by up to three signed or unsigned decimal digits, and a binary scaling consisting of a B followed by one or two signed or unsigned decimal digits. If the value contains a decimal point in any position, the constant is packed into floating point form; otherwise a fixed point constant results. The magnitude of a fixed point number must be less than 2^{47}; the magnitude of a floating point number must fall within the range $10^{\pm 308}$. A decimal constant of the form fDdBb is equivalent to the expression $f \cdot 10^{d} \cdot 2^{b}$.

DECD

L	O	A
ℓ	DECD	$d_1,d_2,d_3, \ldots ,d_n$

DECD inserts double-precision floating point decimal constants. The format is identical to DEC except that each constant may consist of up to 28 decimal digits, and will occupy two machine words.

BCD

L	O	A
ℓ	BCD	n, <8n characters >

BCD inserts binary-coded-decimal characters into consecutive words. A location symbol is optional; if used, it will be assigned to the first word. The address field consists of single digit n, where $1 \leq n \leq 6$, or a previously defined symbol, followed by a comma and 8n succeeding characters, including blanks, ending before column 73. This results in n computer words, each containing 8 BCD characters. Anything after 8n characters is treated as remarks.

TYPE L O A

ℓ TYPE n, <8n characters>

TYPE inserts typewriter codes into consecutive words. The format and results are similar to BCD, except that resulting codes are typewriter codes. The typewriter codes are represented by the BCD (keypunch) characters which correspond to the set of lower case typewriter characters.

Each lower case typewriter character and typewriter function for which no BCD equivalent exists is represented by two BCD characters, the first of which is an asterisk. These special codes are listed in Table 6. The character count, n, pertains to the number of TYPE characters.

VFD L O A

ℓ VFD mn/v, . . .

VFD, Variable Field Definition, assigns data in continuous strings of bits rather than in word units. With this command, octal numbers, character codes, program locations, and arithmetic values may be catenated regardless of word breaks. If the last half-word is not entirely filled with data, it is padded with zeros.

The address field consists of one or more consecutive subfields separated by commas. In each subfield, m specifies the mode of the data, n the number of bits allotted, and v the value.

Five modes are allowed:

O Octal number. May be signed negative, implying the one's complement form.

H Hollerith character code; n must be a multiple of six. The number of characters designated (n/6), including imbedded spaces, follows the slash. Any printable character may appear in the v field. The last character is followed by a space or comma.

T Typewriter character code. The same rules apply as in the Hollerith mode.

B Bank term. The n field is omitted and assumed 3. The term must coincide with one of the five bank designator portions of a machine word (bits 41-39, 34-32, 26-24, 17-15, 10-8). The v field may contain an expression, evaluated modulo 8, or one of the bank relocatable terms "*" (bank of the subprogram in which VFD appears; pg. 5) or "$name" (bank associated with that external or common block name; pg. 7).

Example:

```
B/2          B/A+B          B/*          B/$ENTRY
```

A Arithmetic expression or decimal constant. The v field consists of an expression formed according to the rules for address field arithmetic, except for the following:

1. n must be less than 48; the modulus of the arithmetic is 2^n-1; constants must be less than 2^n.

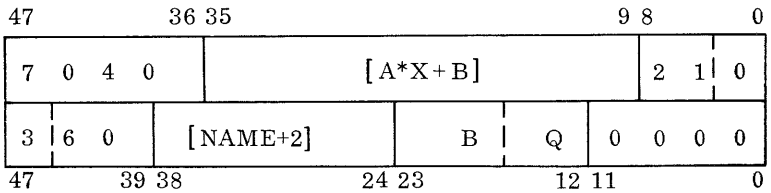
2. If an expression results in a relocatable value, it must fit into a 15-bit field right-justified at position 24 or 0. The modulus of the arithmetic is $2^{15}-1$, and constants must be less than 2^{15}.

3. If an expression results in a fixed value, n must be sufficient to contain it. If n is too small, an error is flagged and the field is set to zero. If the field length exceeds the size required for a value, the value is right-justified with the sign extended in the high order bits.

Example:

```
VFD O12/-737,A27/A*X+B,H18/A3 ,A15/NAME+2,T12/BQ
```

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



The VFD address field is terminated by the first blank column not within an H or T value.

The location field may be blank, or contain a symbol of Type 1, 2, or 3.

4.4 ASSEMBLER CONTROL

The assembly process is controlled or modified by these pseudo instructions.

CODAP

	L	O	A
		CODAP	

CODAP causes the assembler to accept cards in CODAP-1 format until a COMPASS pseudo instruction is encountered. If no CODAP pseudo instruction occurs, COMPASS format is assumed for all cards except those with a punch in column 9, which are always treated as CODAP-1 cards. A location symbol is meaningless. Everything beyond column 15 is treated as comments.

COMPASS

	L	O	A
		COMPASS	

COMPASS returns the mode of input format to COMPASS if a CODAP pseudo instruction occurred previously. If the current format mode is COMPASS this pseudo instruction will be ignored. Everything beyond column 17 is treated as comments.

EQU

	L	O	A
	ℓ	EQU	m

EQU equates a symbol to the value of the address field expression. The expression must conform to the rules for m subfields, with the exception that an external symbol may not be combined with any other element. Any symbols in the address field must be previously defined. A blank location field is meaningless.

IFZ

	L	O	A
		IFZ	m,n

IFZ causes the next n coding lines to be assembled if the value of m is zero. Only an arithmetic expression may be given in the m subfield and it must conform to the rules specified for the address subfield m. All symbols must have been previously defined. The expression is evaluated modulo $2^{15}-1$.

The n subfield must contain an expression resulting in a positive integer. A location symbol is meaningless. Other IFZ (or IFN) pseudo instructions may fall in the range of an IFZ in order to impose a series of conditions.

IFN	L	O	A
		IFN	m,n

IFN is identical to IFZ, except that the next n coding lines are assembled if the value of the m expression is non-zero.

IFT	L	O	A
		IFT,s	m,n,p

IFT may be used only within the range of an ECHO or MACRO pseudo instruction. If it occurs elsewhere, it will be flagged as an O error. An operation modifier, s, is required. The instruction reads as follows: If it is true that m s n, assemble the next p coding lines. The m and n subfields may be formal parameter names or character strings. The p subfield must be a positive integer.

Either or both of the subfields m and n may contain a formal parameter name of the general form < formal parameter > (i,j). If (i,j) is absent, the entire actual parameter will be used as the comparison quantity; otherwise, i and j define the portion of the actual parameter to be used. (i,j) may follow one of four types, in which p, q, and r are integers and k is any non-blank BCD character except slash.

Type	Interpretation
(p,q)	q consecutive characters beginning with the pth character in the actual parameter.
(r_1=k_1,q)	q consecutive characters following the r_1th occurrence of the character k_1.
(p,r_2=k_2)	consecutive characters beginning with the pth character up to, but not including the r_2th occurrence, following the pth character, of the character k_2.

$(r_1=k_1,r_2=k_2)$	consecutive characters following the r_1th occurrence of the character k_1 up to, but not including the r_2th occurrence of the character k_2.

If r, but not the equal sign, is omitted, r is assumed one.

Either, but not both, of the subfields m and n may contain a character string for literal comparison enclosed in slashes. The character string may not itself contain slashes. When the comparison quantities differ in length, they are matched character-for-character through the length of the shorter; if the match is identical, the longer is greater; if there is no match, the larger quantity, regardless of its length, is greater.

The operation modifier, s, may be any one of the following two-character mnemonics:

mnemonic	meaning
EQ	m=n
NE	$m \neq n$
LT	$m < n$
LE	$m \leq n$
GT	$m > n$
GE	$m \geq n$
IN	m included in n; the character string n contains the characters in the string m in sequence, but not necessarily consecutively.

IFF

L	O	A
	IFF,s	m,n,p

IFF is identical to IFT, except that the next p coding lines are assembled if m s n is false.

ORGR

L	O	A
	ORGR	m

ORGR causes subsequent instructions to be assembled beginning at the value of the address field. Symbols in the address field must have been

previously defined, and the value of the address field must be program or common relocatable.

If ORGR is used to originate a sequence of data in labeled common, the value of the address field is common relocatable. Address fields of subsequent instructions may not contain external symbols or relocatable bank subfields. The number of machine words generated by presetting common must be less than or equal to the length of the common block. After the presetting operation is completed, an ORGR with an asterisk (*) in the address field may be used to resume subprogram instructions. ORGR forces upper.

SCOPE L O A

SCOPE

SCOPE terminates the assembly process and causes COMPASS to return control to the operating system. The SCOPE pseudo instruction should follow the END of the last subprogram to be assembled by COMPASS. If SCOPE follows any card which is not END, control will be returned immediately to SCOPE but an O error will be flagged on the output listing. A non-blank L field is meaningless. Everything beyond column 15 is treated as comments.

ECHO L O A

ECHO m,n,(p_1=a_1,a_2, . . . ,a_n,p_2=b_1,b_2,

ECHO . . . ,b_n, . . . ,p_k=k_1,k_2, . . . ,k_n)

ECHO causes replication of the following m coding lines n times. In the first replication the actual parameters a_1,b_1, . . . ,k_1 are substituted for the formal parameters p_1,p_2, . . . ,p_k. In the second replication a_2,b_2, . . . ,k_2 are substituted for p_1,p_2, . . . ,p_k, and so on. The number of actual parameters given for each formal parameter should be equal to or greater than the number of replications.

```
ECHO    3,3,(T1=A,B,C,T2=D,E,F,T3=G,H,I)
LDA     T1
FAD     T2
```

```
        STA     T3
```

expands to nine symbolic instructions:

```
        LDA     A
        FAD     D
        STA     G
        LDA     B
        FAD     E
        STA     H
        LDA     C
        FAD     F
        STA     I
```

The IDENT, ORGR, END, MACRO, ENDM, BLOCK, and ECHO pseudo instructions are excluded from the range of an ECHO; all other machine instructions and pseudo instructions are permitted. The address field of ECHO may be terminated by a blank column following n if no parameters are required.

Example:

```
        ECHO    2,3
        OCT     1
        OCT     0
```

expands to six assembled computer words, in the order:

```
        OCT     1
        OCT     0
        OCT     1
        OCT     0
        OCT     1
        OCT     0
```

This may be condensed to:

```
ECHO    1,3
OCT     1,0
```

which is an equivalent form producing the same six computer words.

Blank columns are permitted within the formal parameter list and will be ignored by COMPASS. The formal parameter names are local to the range of the ECHO, and must be Type 1 symbols. The actual parameters may be any expressions which legally may appear where the formal parameters occur.

In any actual parameter, except a Hollerith or typewriter literal, blanks are ignored, and parentheses must be matched; if the parameter contains subfields separated by commas, the entire parameter must be enclosed in parentheses.

A location symbol within an ECHO range is assigned only in the first replication and ignored in successive replications. The ECHO pseudo instruction may not be labeled. If it is, the location symbol will be ignored. Comments within the range of ECHO will appear on the output listing only in the first replication.

The parameter list in ECHO may be continued on subsequent cards by repeating the ECHO pseudo instruction on each card. Up to ten cards may be used for one ECHO. A parameter name must be contained entirely on one card.

Example:

```
ECHO    3,2,(P1=A,B,P2=
ECHO    C,D,P3=E,F)
LDA     P1
  .
  .
  .
```

The instructions within the range of an ECHO need not generate an integral number of machine words. Consider the two examples:

```
ECHO    1,3        and    ECHO    2,3
LDA     0                 OCT     1
                          LDA     0
```

The first produces three 24-bit sequential machine instructions; the second produces five and a half computer words.

4.5

OUTPUT LISTING

These pseudo instructions control the printing of the output listing; they are not printed on the listing.

EJECT

	L	O	A
		EJECT	

EJECT will produce a character in column one of the next record of the output listing unit which causes the line printer to eject paper to the top of the next page. EJECT forces the next instruction upper.

SPACE

	L	O	A
		SPACE	m

SPACE will cause the listing to be spaced the number of lines specified by the address field. If this spacing would cause an overflow at the bottom of the page, the page is ejected to the top of the next page only.

LIST

	L	O	A
		LIST	

COMPASS will resume the output listing when LIST is encountered if it was previously suppressed by NOLIST. If NOLIST has not been encountered previously, LIST will be ignored.

TITLE

	L	O	A
		TITLE	

The legend in columns 16-72 is printed at the top of each page of the program listing. Any printable character may appear in the legend.

The TITLE pseudo instruction may be given at any point in the program; when it is received, a page eject character is written on the output listing unit and the title is written on that page and every page thereafter until a new title is given.

NOLIST

L	O	A
	NOLIST	

NOLIST will suppress the output listing until a LIST pseudo instruction is encountered. However, lines with error flags will still be listed.

REM

L	O	A
	REM	

REM produces an output record which contains remarks only. All columns except 9-13 inclusive are available for remarks. REM forces the next instruction upper.

4.6 MACRO DEFINITION

L	O	A
ℓ	MACRO	$(p_1,p_2,\ldots,p_n)$

MACRO delineates an often used set of instructions, a macro, which may be called by a macro instruction. The pseudo instruction signals a macro definition. It is not a macro instruction and it does not generate any machine words.

The location field of MACRO must contain a Type 1 location symbol which names the macro instruction. This name may not be any of the machine language mnemonics or pseudo instructions. The address field, which may not extend beyond column 72, contains the formal parameter list; the parameters are separated by commas, and the entire list is enclosed in parentheses. Blanks are permitted within the parentheses, but will be ignored by the assembler. The parameter list may be continued on subsequent cards by repeating the MACRO pseudo instruction; the location symbol must not be repeated. A parameter name must be contained entirely on one card. The address field on the last card terminates with a right parenthesis. The formal parameter names are local to the macro definition and may be used elsewhere in the subprogram without ambiguity.

Following MACRO are the input records containing a prototype of the set of instructions. These instructions may be any of the machine instructions or pseudo instructions acceptable to COMPASS except MACRO, ORGR, IDENT, and END. The prototype may also contain macro instructions. Fields and subfields of instructions within a prototype may be text or formal parameters. If a location symbol appears within the prototype but is not a formal parameter, it is local to the macro and may be used elsewhere in the subprogram without ambiguity.

The three type of macros in COMPASS are programmer-defined macros, library macros, and system macros. All programmer-defined macros must be defined by macro pseudo instructions immediately following any LIBM pseudo instructions. Library macros are contained in an expandable macro library on the library tape. Any library macro called by a subprogram must be declared in LIBM pseudo instructions. System macros are also contained in the macro library but need not be declared by the programmer. The system macros are calling sequences to routines in the SCOPE system, described in the SCOPE Reference Manual.

ENDM

	L	O	A
		ENDM	

ENDM signals the end of a macro definition. A location symbol will be ignored

LIBM

	L	O	A
		LIBM	$m_1,m_2,m_3, \ldots$

LIBM provides COMPASS with the names of the library macros to be called by the subprogram. These names appear in the address field in the form NAME1, NAME2, NAME3, . . . , with the field terminated by the first blank column.

All LIBM pseudo instructions must occur together immediately following IDENT. An improperly placed LIBM is ignored and an O error is flagged. A location symbol is ignored.

4.7 MACRO INSTRUCTION

A macro instruction causes the macro named in the operation field to be inserted at that point in the program.

L	O	A
ℓ	<macro name>	$(p_1,p_2, \ldots ,p_n)$

If a location symbol appears in the macro instruction, it will be assigned to the first word. The address field contains the actual parameter list in the same order as that of the formal parameter list in the macro definition. The parameter list may be continued on subsequent cards

by repeating the macro instruction name; the location term should not be repeated. A parameter name must be contained entirely on one card. The address field on the last card terminates with a right parenthesis.

In any actual parameter, except a Hollerith or typewriter literal, blanks are ignored, and parentheses must be matched; if the parameter contains subfields separated by commas, the entire parameter must be enclosed in parentheses. A macro instruction may also be used as a parameter. All other symbolic names in the parameter list are assumed to be defined somewhere within the subprogram. The number of machine words generated by the macro instruction will depend upon the length of the prototype and the occurrence of IFZ, IFN, IFT, and IFF pseudo instructions within the prototype.

Examples:

The following would define a macro called XYZ.

```
XYZ     MACRO   (P1,P2,P3,P4,
        MACRO   P5,P6)
        ENI     P5,P6
P4      DLDA    P3
        P1      P2
        DFAD    SYMBOL1
        IFN     P5,3
        IFT,EQ  P1,/DFAD/,2
        DSTA    SYMBOL3
        DLDA    SYMBOL2
        ENDM
```

The macro instruction, XYZ, might be used as follows:

```
        XYZ     (DFAD,(SYMBOL4,1),((1)SYMBOL5),HERE,
        XYZ     3,$B4)
```

The macro instruction above would generate the following set of instructions:

```
          ENI     3,$B4
HERE      DLDA    (1)SYMBOL5
          DFAD    SYMBOL4,1
          DFAD    SYMBOL1
          DSTA    SYMBOL3
          DLDA    SYMBOL2
            .
            .
            .
```

Another call of the macro, XYZ, might be:

```
Q3        XYZ     (DFSB,SYMBOL7,SYMBOL8,THERE,
          XYZ     2,$B2)
```

This reference to the macro, XYZ, would cause the following instructions to be inserted:

```
Q3        ENI     2,$B2
THERE     DLDA    SYMBOL8
          DFSB    SYMBOL7
          DFAD    SYMBOL1
            .
            .
            .
```

The following would define a macro called GOUT:

```
GOUT      MACRO   (M,P)
          RTJ     GOUT
          ECHO    1,M,(Q=P)
+                 Q
          ENDM
```

The macro instruction

```
          GOUT    (5,(A,B,C,D,E))
```

would cause the following instructions to be assembled:

```
        RTJ     GOUT
+               A
                B
                C
                D
                E
```

4.8 PROGRAM MODIFICATION

4.8.1 COSY

In addition to listable and relocatable binary output, the COMPASS user may elect to receive a COSY deck as output from COMPASS. The COSY deck contains a compressed symbolic form, in binary, of the program which may be used as input for subsequent assemblies. The COSY option has four advantages:

- The size of the input deck may be reduced by a maximum ratio of 19:1.
- The time required for subsequent assemblies is decreased.
- The COSY deck may be modified using the COMPASS symbolic language.
- An up-to-date COSY deck may be punched with each subsequent assembly.

The deck consists of COSY text cards and a COSY end card. Starting with 00001 for the IDENT card, a sequence number assigned to each input card is printed on the right side of the output listing opposite the input line. This sequence number is used as the reference point when modifying a COSY deck.

4.8.2 MODIFICATION INSTRUCTIONS

The pseudo instructions which may be used to modify a COSY deck are punched in the COMPASS symbolic format. All modifications must precede the COSY identification card, but need not appear in the order of occurrence on the associated output listing.

DELETE	L	O	A
		DELETE	m,n

This pseudo instruction deletes symbolic input lines m through n where m and n are sequence numbers.

REPLACE	L	O	A
		REPLACE	m,n

REPLACE causes COMPASS to replace lines m through n with the lines which follow, up to the next modification pseudo instruction or up to the COSY identification card. The replacement need not be one to one.

INSERT	L	O	A
		INSERT	m

This pseudo instruction inserts the lines which follow up to the next modification pseudo instruction or up to the COSY identification card. The lines are inserted after input record m in the COSY deck.

The address subfields, m and n, of the modification pseudo instructions must comply with the following rules:

1. m and n must be less than or equal to the sequence number of the END record.

2. The modified deck must contain an END card.

3. In the address field of DELETE or REPLACE, m must be less than n.

4. If only one input card is to be affected, n should not appear; only line m will be deleted or replaced.

An address field error will void the modification.

The formats of the COSY text and end cards are given below:

COSY Text Cards

Columns	1	12,7,9 punch rows 4-6 contain the high order digit of a 5-digit card sequence number
	2	4 lower order digits of 5-digit card sequence number
	3-4	24-bit checksum
	5-80	19 words of compressed symbolic

COSY End Card

Columns	1	12,3,7,9 punch rows 4-6 contain the high order digit of a 5-digit card sequence number
	2	4 lower order digits of 5-digit card sequence number
	3-4	24-bit checksum
	5-80	zeros

COSY	L	O	A
		COSY	

The COSY identification card signals the end of the modification deck and the beginning of the COSY deck. The location and address fields are ignored.

OUTPUT LISTING 5

The output listing produced by COMPASS consists of the input card images preceded, on the left, by a column of error codes, a column of machine locations, and the assembled contents of the machine locations. If a COSY deck is requested, a sequence number will appear to the right of each input line. The error code column may contain up to eight of the error codes described in Table 7.

The machine location column appears in octal, opposite the upper half of each assembled machine word.

The machine contents column contains three terms, a 2-digit operation, a 1-digit index designator, and a 5-digit address term. The address term will be preceded by P if the address is a program location. If the address is relocatable with respect to common, it will be preceded by C. If the address term of the input line references an external symbol, the address term of the machine contents column will contain the assembled machine location of a previous reference to the same external symbol; this address will be preceded by an X.

COMPASS will always produce a list of entry points, block common names, external symbols, undefined symbols, doubly defined symbols, nulls (location symbols which have not been used), the length of the subprogram, any flagged coding lines (regardless of the list option), and number of errors in octal.

Example of a COMPASS output listing

```
                ORIGINAL SOURCE DECK LISTING
                    IDENT       SENSLESS
A                   BLOCK       0
                    COMMON      ARRAY1(100B), ARRAY1A(20B)
PROGRAMA            SLJ         **
                    STA         TEMP,1
                    STQ         TEMP,2
                    LDA,MG      TEMP,1
                    RTJ         PROGRAMB
                    EXT         PROGRAMB
+                   LDQ         =07700000
                    STL         TEMP
                    ENTRY       PROGRAMA
                     .
                     .
                     .
                    LIU         PROGRAMB,3
                    SIL         ARRAY1+20B,3
                     .
                     .
                     .
                    SLJ         PROGRAMA
B                   BLOCK       0
                    COMMON      ARRAY2(100)
                    ORGR        ARRAY2
                    OCT         0,1,2,3
                    ORGR        *
EXCH                SLJ         **
                    LBYT,A0,E6,R1,CL    0,3,4
                    INI         1,3
                    ENI         42,4
                    SLJ         EXCH
                    END
```

OUTPUT LISTING FROM ABOVE SOURCE DECK

```
                                        IDENT    SENSLESS                        00001

PROGRAM LENGTH              00307

BLOCK NAMES
                    A       00120
                    B       00144
ENTRY POINTS
                    PROGRAMA 00000

EXTERNAL SYMBOLS
                    PROGRAMB
                            A           BLOCK    0                               00002
00000                                   COMMON   ARRAY1(100B),ARRAY1A(20B)       00003
00100                                            ARRAY1A(20B)
00000    75  0   77777   PROGRAMA SLJ    **                              00004
         20  1   P00221           STA    TEMP,1                          00005
00001    21  2   P00221           STQ    TEMP,2                          00006
         50  0    00000
00002    77  1    00004           LDA,MG TEMP,1                          00007
         12  1   P00221
00003    75  4   X77777           RTJ    PROGRAMB                        00008
                                    EXT    PROGRAMB                        00009
         50  0    00000
00004    16  0   P00306   +       LDQ    =O7700000                       00010
         47  0   P00221           STL    TEMP                            00011
                                    ENTRY  PROGRAMA                        00012
  .                                  .                                       .
  .                                  .                                       .
  .                                  .                                       .
  .                                  .                                       .
00207    52  3   X00003           LIU    PROGRAMB,3                      00131
         57  3   C00020           SIL    ARRAY1+20B,3                    00132
                                    .
                                    .
00301    75  0   P00000           SLJ    PROGRAMA                        00178
         50  0    00000
                            B       BLOCK    0                               00179
00000                               COMMON   ARRAY2(100)                     00180
                    C00000          ORGR     ARRAY2                          00181
00000    00  0    00000           OCT    0,1,2,3                         00182
         00  0    00000
00001    00  0    00000                  1
         00  0    00001
00002    00  0    00000                  2
         00  0    00002
00003    00  0    00000                  3
         00  0    00003
                    P00302          ORGR     *                               00183
```

```
00302    75  0   77777  EXCH     SLJ      **                          00184
         50  0   00000
00303    63  3   40006           LBYT,A0,E6,R1,CL   0,3,4             00185
         52  0   00000
00304    51  3   00001           INI      1,3                         00186
         50  4   00052           ENI      42,4                        00187
00305    75  0  P00302           SLJ      EXCH                        00188
         50  0   00000
00306    00  0   00000
         07  7   00000
                                 END                                  00189
```

CORRECTION DECK TO BE ASSEMBLED WITH COSY DECK ASSOCIATED WITH ABOVE LISTING

```
                                 REPLACE  3
                                 COMMON   ARRAY1(120B)
                                 COMMON   ARRAY1A
                                 DELETE   5,6
                                 DSTA     TEMP+2
                                 INSERT   10
                                 ADD      =0400000
                                 DELETE   131
```

OUTPUT LISTING FROM ASSEMBLY OF ABOVE CORRECTION DECK AND COSY DECK WITH COSY OUTPUT OPTION SELECTED

```
                                 COMMON   ARRAY1(120B)          ***INSERTED
                                 COMMON   ARRAY1A               ***INSERTED
                                 STA      TEMP,1                    DELETED
                                 STQ      TEMP,2                    DELETED
                                 DSTA     TEMP+2                ***INSERTED
                                 ADD      =O400000              ***INSERTED
                                 LIU      PROGRAMB,3                DELETED

                                 IDENT    SENSLESS                    00001

PROGRAM LENGTH                00310

BLOCK NAMES
                    A         00121
                    B         00143
ENTRY POINTS
                    PROGRAMA  00000

EXTERNAL SYMBOLS
                    PROGRAMB
                            A        BLOCK    0                        00002
00000                                COMMON   ARRAY1(120B)          ***00003
00120                                COMMON   ARRAY1A               ***00004
00000    75  0   77777  PROGRAMA SLJ      **                          00005
         50  0   00000
00001    77  2   00000           DSTA     TEMP+2                   ***00006
         20  0  P00220
```

```
00002    77  1   00004          LDA,MG   TEMP,1                    00007
         12  1   P00216
00003    75  4   X77777          RTJ      PROGRAMB                 00008
                                EXT      PROGRAMB                 00009
         50  0   00000
00004    16  0   P00306          LDQ      =O7700000                00010
         14  0   P00307          ADD      =O400000                 ***00011
00005    47  0   P00216          STL      TEMP                     00012
                                ENTRY    PROGRAMA                 00013
  .                               .                                  .
  .                               .                                  .
00207    57  3   C00020          SIL      ARRAY1+20B,3             00131
  .                               .                                  .
  .                               .                                  .
  .                               .                                  .
00301    75  0   P00000          SLJ      PROGRAMA                 00178
         50  0   00000
                         B      BLOCK    0                        00179
00000                           COMMON   ARRAY2(100)              00180
                 C00000         ORGR     ARRAY2                   00181
00000    00  0   00000          OCT      0,1,2,3                  00182
         00  0   00000
00001    00  0   00000                   1
         00  0   00001
00002    00  0   00000                   2
         00  0   00002
00003    00  0   00000                   3
         00  0   00003
                 P00302         ORGR     *                        00183
00302    75  0   77777   EXCH   SLJ      **                       00184
         50  0   00000
00303    63  3   40006          LBYT,A0,E6,R1,CL    0,3,4         00185
         52  0   00000
00304    51  3   00001          INI      1,3                      00186
         50  4   00052          ENI      42,4                     00187
00305    75  0   P00302          SLJ      EXCH                     00188
         50  0   00000
00306    00  0   00000
         07  7   00000
00307    00  0   00000
         00  4   00000          END                               00189
```

CONTROL CARDS AND DECK STRUCTURES

6

6.1 CONTROL CARD

The control card appears immediately before the first card of the first subprogram to be assembled. Column 1 of the control card contains punches in rows 7 and 9. Starting in column 2, the word COMPASS appears, followed by a comma and up to 6 free field parameters separated by commas. The control card is terminated with a period or end of card.

6.2 ASSEMBLY OPTIONS

There are six options which may be specified on the control card; each option has the general form O=o, where O is the parameter and o is the value. The parameter must begin with one of the following characters, I, Y, P, C, X, L; any characters following, up to an equals sign or comma, are ignored. Thus, LIST is equivalent to L. The value of the parameter may be specified by "=o" where o is a decimal integer; if "=o" is absent, COMPASS will assume a value as described below.

I = i Hollerith input medium where the logical unit i may assume values 1-49, 60; if the parameter is absent, input from unit 60 (standard input unit-INP) is assumed.

Y = y COSY input medium where the logical unit y may assume values 1-49, 60; if the parameter is absent, input from unit 60 (standard input unit-INP) is assumed.

P = p Punch option where the logical unit p may assume values 0-49, 62; if the parameter is absent or equal to zero, no binary output will be produced. If only P appears, binary output will be produced on unit 62 (standard punch unit-PUN).

C = c COSY output option where the logical unit c may assume values 0-49, 62, if the parameter is absent or equal to zero, no COSY output will be produced. If only C appears, COSY output will be produced on unit 62 (standard punch unit -- PUN).

X = x Binary output for load-and-go option where x may assume values 0-49, 69; if the parameter is absent or equal to zero, no load-and-go tape will be written. If only X appears, binary output for load and go will be produced on unit 69 (standard load-and-go unit -- LGO).

L = ℓ List option where ℓ may assume any value; if the parameter is absent or equal to zero, no listing will be produced; otherwise the listing will be produced on unit 61 (standard listable output-OUT).

I and Y need be explicitly declared only if input from a unit other than the standard input is desired.

An unrecognizable option is ignored.

Examples:

$\begin{matrix}7\\9\end{matrix}$ COMPASS, L,P,C. equivalent to $\begin{matrix}7\\9\end{matrix}$ COMPASS,LIST,PUNCH,COSY.
The above will produce a listing on unit 61 with COSY output and binary output on unit 62.

$\begin{matrix}7\\9\end{matrix}$ COMPASS, Y=1,L,C=2.

The above will read COSY input from unit 1 and produce a listing on unit 61 with COSY output on unit 2.

6.3 SUBPROGRAM DECKS

One or more subprograms follow the control card. Subprograms may be input in Hollerith or Hollerith/COSY; the two forms may be mixed within one batch of assemblies.

A Hollerith subprogram consists of Hollerith punched cards, or magnetic tape images of Hollerith cards; the parameter Y on the control card is meaningless in an assembly of a Hollerith subprogram.

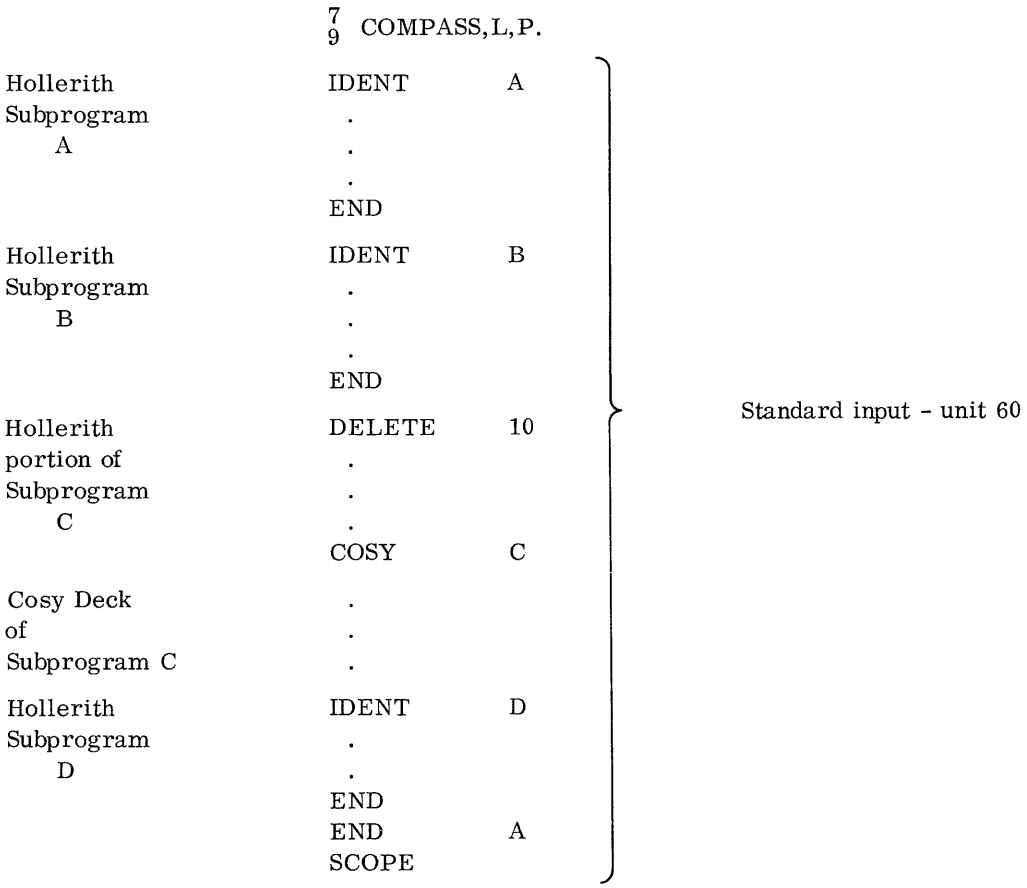
A Hollerith/COSY subprogram consists of Hollerith corrections followed by a COSY deck obtained from a previous assembly. The Hollerith portion of the subprogram consists of modifications to be made to the COSY deck and is terminated by a COSY card (see page 31). If there are no modifications, the COSY card comprises the entire Hollerith portion of the subprogram.

The Hollerith portion of the subprogram is read from the unit defined by I on the control card, or from unit 60 if I is not declared.

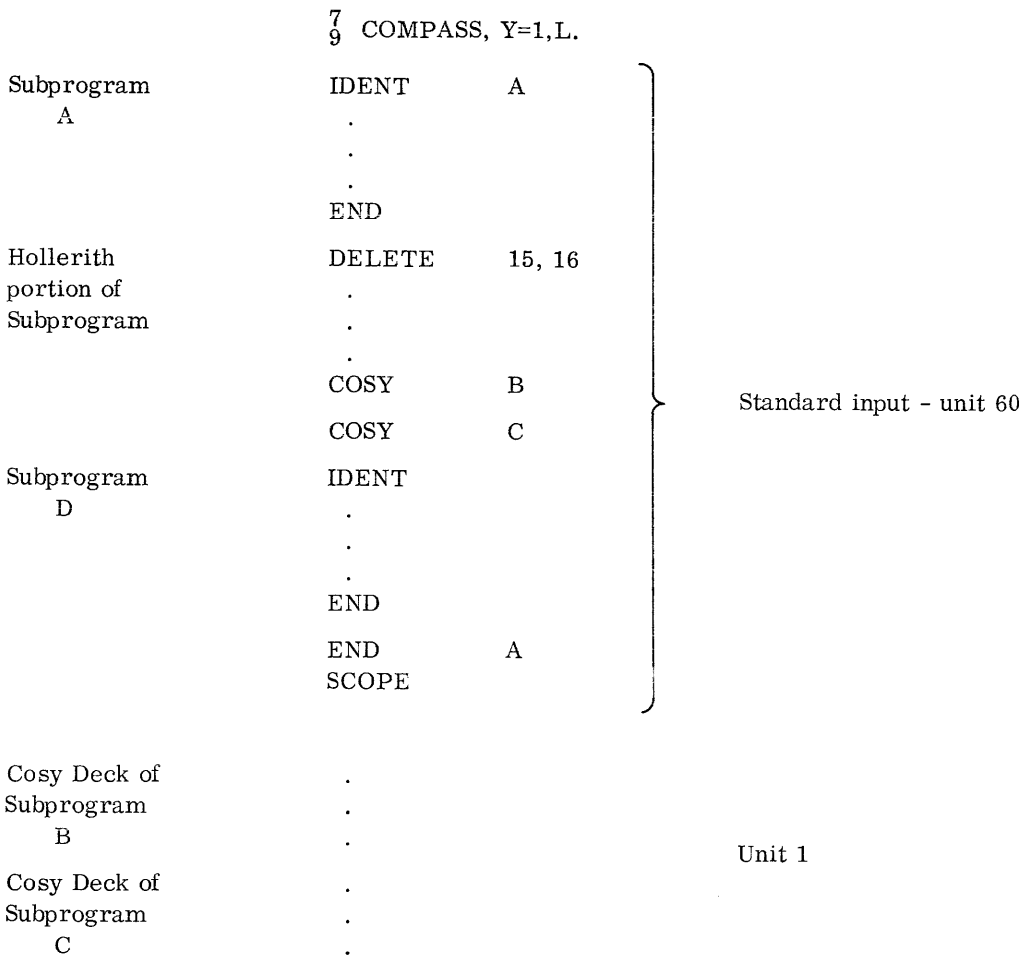
The COSY portion of the subprogram follows the COSY card which terminates the Hollerith portion. The COSY portion is read from the unit defined by Y on the control card, or from unit 60 if Y is not declared.

I may differ from Y.

Examples:



In the above example, subprograms A, B, C, and D would be assembled from unit 60 with listings produced on unit 61 and binary output on unit 62.



In this example, subprograms A and D would be assembled from Hollerith on unit 60, and subprograms B and C from Hollerith/COSY on units 60 and 1.

TABLE 1

MNEMONIC CODES FOR 3600 OPERATION REGISTERS

(Source and Destination)

Mnemonic Code	Register Name
LM	Limit Register
B1	B^1 (Index Register 1)
B2	B^2 (Index Register 2)
B3	B^3 (Index Register 3)
B4	B^4 (Index Register 4)
B5	B^5 (Index Register 5)
B6	B^6 (Index Register 6)
AL	A - Lower Address
AU	A - Upper Address
QL	Q - Lower Address
QU	Q - Upper Address
A	A - Full 48 bits
Q	Q - Full 48 bits
D	D Register
BR	Bounds Register
IM	Interrupt Mask Register
OB	Operand Bank Register

TABLE 2

MNEMONIC CODES FOR 3600 OPERATIONAL REGISTERS

(Source Only)

Mnemonic Code	Register Name
IR	Interrupt Register
PZ	All Zeros
P1	Plus One
MZ	Minus Zero (all ones)
IB	Instruction Bank Register
NC	Normalization Count Register
MS	Mode Selection Register
P	P Register
CK	Clock

TABLE 3

MNEMONIC CODES FOR INSTRUCTION MODIFIERS

AND	Register and -- ROP instruction
Ao	Use A register in the LBYT or SBYT instruction; o is a one or two digit decimal integer which specifies the right most bit of the byte in A.
AUG	Augment -- XMIT instruction
C	Chain to next control word -- I/O control words
CL	a) Clear source -- augmented instructions b) Clear unused portion of destination -- LBYT, SBYT instructions c) Clear bit g in register p after testing -- NBJP, ZBJP instructions
CM	a) Complement operand -- augmented instructions b) Complement bit g in register p after testing -- NBJP, ZBJP instructions c) Transmit complement -- XMIT instruction
CQ	Clear unused portion of q in RSW and RXT instructions
CR	Clear unused portion of r in RSW and RXT instructions
CW	Control Word to A -- COPY instruction
CWA	Control Word Address to Q -- COPY instruction
D	Conditional decrementing -- RGJP instruction
Ee	In the LBYT, SBYT, and SCAN instructions, e is a one or two digit decimal integer which specifies the byte size in bits.
EO	End Off; shift is end off and no sign extension -- augmented instructions
EQ	Equal test -- RGJP, IFF, IFT instructions, register equivalence -- ROP instruction
GE	Greater or equal test -- RGJP, IFF, IFT instructions
GT	Greater test -- RGJP, IFF, IFT instructions
I	Indirect addressing -- SEQU, SMEQ, SEWL, SMWL instructions
IMP	Register implication -- ROP instruction

IN	Inclusion test -- IFF, IFT instructions
LE	Less or equal test -- RGJP, IFF, IFT instructions
LI	Left indexing -- LBYT, SBYT instructions
LT	Less test -- RGJP, IFF, IFT instructions
MG	Magnitude of operand -- augmented instructions
MI	Minus -- AJP, QJP, ARJ, QRJ instructions
MK	Transmit masked -- XMIT instructions
NE	Not equal test -- RGJP, IFF, IFT instructions
NZ	Non-zero -- AJP, QJP, ARJ, QRJ instructions
OR	Register or -- ROP instruction
PC	Transmit plus constant (in A) -- XMIT instruction
PL	Plus -- AJP, QJP, ARJ, QRJ instructions
Qo	Use Q register in the LBYT, SBYT, or SCAN instruction; o is a one or two digit decimal integer which specifies right-most bit of the byte in Q.
RI	Right indexing -- LBYT, SBYT instructions
RP	Replace operation -- augmented instructions
SS	Signed shift -- (Direction of shift determined by sign of shift count) -- augmented instructions
ST	Set to one -- NBJP, ZBJP instructions
TR	Truncated -- DVF instruction
UN	Un-normalize arithmetic -- augmented instructions
UR	Unrounded arithmetic -- augmented instructions
XOR	Register exclusive or -- ROP instruction
ZR	Zero -- AJP, QJP, ARJ, QRJ instructions
+	Register sum -- ROP instruction
-	Register difference -- ROP instruction

TABLE 4

MNEMONIC MACHINE INSTRUCTIONS

Operation Field	Address Field	Instruction
Inter-Register		
ROP, OR	p, q, r	Register operation
XOR		r = p op q
AND		
IMP		
EQ		
+		
-		
RSW, CQ, CR	q, r	Register swap
RXT, CQ, CR	q, r	Register transmit
Full Word Transmission		
LDA, CM, MG	(a) m, b, v	Load A
LAC, CM, MG	(a) m, b, v	Load A complement
LDQ, CM, MG	(a) m, b, v	Load Q
LQC, CM, MG	(a) m, b, v	Load Q complement
STA, CM, CL, MG	(a) m, b, v	Store A
STQ, CM, CL, MG	(a) m, b, v	Store Q
XMIT, CM, AUG	(a) m, (i) n	Transmit*
MK		
PC		

*If either bank term is missing, it is assumed ($).

Operation Field	Address Field	Instruction
Address Transmission		
LIU, CM, MG	(a) m, b, v	Load index upper
LIL, CM, MG	(a) m, b, v	Load index lower
SIU	(a) m, b, v	Store index upper
SIL	(a) m, b, v	Store index lower
SAU, CM, MG	(a) m, b, v	Substitute address upper
SAL, CM, MG	(a) m, b, v	Substitute address lower
ENI	(a) y, b, v	Enter index
ENA, CM	(a) y, b, v	Enter A
ENQ, CM	(a) y, b, v	Enter Q
Fixed Point Arithmetic		
ADD, CM, MG	(a) m, b, v	Add
SUB, CM, MG	(a) m, b, v	Subtract
MUI, CM, MG	(a) m, b, v	Multiply integer
DVI, CM, MG	(a) m, b, v	Divide integer
MUF, CM, MG	(a) m, b, v	Multiply fractional
DVF, CM, MG, TR	(a) m, b, v	Divide fractional
Address Arithmetic		
INA, CM	(a) y, b, v	Increase A
INI	(a) y, b, v	Increase index
ISK	(a) y, b, v	Index skip
Single Precision Floating Point Arithmetic		
FAD, RP, CM, MG, UN, UR	(a) m, b, v	Floating add
FSB, RP, CM, MG, UN, UR	(a) m, b, v	Floating subtract
FMU, CM. MG, UN, UR	(a) m, b, v	Floating multiply
FDV, CM, MG, UR	(a) m, b, v	Floating divide
ADX	w	Add to exponent

Operation Field	Address Field	Instruction
Double Precision Floating Point Arithmetic		
DLDA, CM, MG	(a) m, b, v	Double precision load A
DSTA, CM, CL, MG	(a) m, b, v	Double precision store A
DFAD, RP, CM, MG, UN, UR	(a) m, b, v	Double precision floating add
DFSB, RP, CM, MG, UN, UR	(a) m, b, v	Double precision floating subtract
DFMU, CM, MG, UN, UR	(a) m, b, v	Double precision floating multiply
DFDV, CM, MG, UR	(a) m, b, v	Double precision floating divide
Logical Operations		
SST, CM, MG	(a) m, b, v	Selective set
SCM, CM, MG	(a) m, b, v	Selective complement
SCL, CM, MG	(a) m, b, v	Selective clear
SSU, CM, MG	(a) m, b, v	Selective substitute
LDL	(a) m, b, v	Load logical
ADL, RP, CM, MG	(a) m, b, v	Add logical
SBL, RP, CM, MG	(a) m, b, v	Subtract logical
STL, CM, MG	(a) m, b, v	Store logical
Shifting Operations		
ARS, EO, SS	(a) y, b, v	A right shift
ALS, EO, SS	(a) y, b, v	A left shift
QRS, EO, SS	(a) y, b, v	Q right shift
QLS, EO, SS	(a) y, b, v	Q left shift
LRS, EO, SS	(a) y, b, v	Long right shift
LLS, EO, SS	(a) y, b, v	Long left shift
SCA	(a) y, b, v	Scale A
SCQ	(a) y, b, v	Scale AQ

Operation Field	Address Field	Instruction
Replace Operations		
RAD, CM, MG	(a) m, b, v	Replace add
RSB, CM, MG	(a) m, b, v	Replace subtract
RAO, CM, MG	(a) m, b, v	Replace add one
RSO, CM, MG	(a) m, b, v	Replace subtract one
Storage Test		
SSK	(a) m, b, v	Storage skip
SSH	(a) m, b, v	Storage shift
Search		
EQS	(a) m, b, v	Equality search
THS	(a) m, b, v	Threshold search
MEQ	(a) m, b, v	Masked equality search
MTH	(a) m, b, v	Masked threshold search
SEQU, I	(a) m, n	Search for equality
SMEQ, I	(a) m, n	Search for masked equality
SEWL, I	(a) m, n	Search within limits
SMWL, I	(a) m, n	Search magnitude within limits
LSTU	b, v	Locate list element upper
LSTL	b, v	Locate list element lower
Jumps and Stops		
AJP, ZR NZ PL MI	(a) m, v	A jump*
QJP, ZR NZ PL MI	(a) m, v	Q jump*

*In AJP, QJP, ARJ and QRJ a modifier is required and does not cause insertion of the single precision augment instruction.

Operation Field	Address Field	Instruction
ARJ, ZR NZ PL MI	(a) m, v	A return jump*
QRJ, ZR NZ PL MI	(a) m, v	Q return jump*

*In AJP, QJP, ARJ and QRJ a modifier is required and does not cause insertion of the single precision augment instruction.

IJP	(a) m, b, v	Index jump
SLJ	(a) m, b, v	Selective jump
SJ1	(a) m, v	Selective jump key 1
SJ2	(a) m, v	Selective jump key 2
SJ3	(a) m, v	Selective jump key 3
RTJ	(a) m, v	Return jump
RJ1	(a) m, v	Selective return jump key 1
RJ2	(a) m, v	Selective return jump key 2
RJ3	(a) m, v	Selective return jump key 3
SLS	(a) m, b, v	Selective stop
SS1	(a) m, v	Selective stop jump key 1
SS2	(a) m, v	Selective stop jump key 2
SS3	(a) m, v	Selective stop jump key 3
SRJ	(a) m, v	Stop return jump
SR1	(a) m, v	Selective stop return jump key 1
SR2	(a) m, v	Selective stop return jump key 2
SR3	(a) m, v	Selective stop return jump key 3
EXEC	(a) m, b, v	Execute
RGJP, EQ GT LT NE LE GE LT, D GE, D	p, y, m, b	Register jump*

*In RGJP modifier is required.

Operation Field	Address Field	Instruction
UBJP	(a) m, b, i	Unconditional bank jump
BJPL	(a) m, b, i	Unconditional bank jump lower
BRTJ	(a) m, b, i	Unconditional bank return jump
BJSX	(a) m, b	Bank jump and set index
NBJP, ST CL CM	p, g, m, b	Non zero bit jump
ZBJP, ST CL CM	p, g, m, b	Zero bit jump
MPJ		Main product register jump
CPJ	x	Channel product register jump
DRJ		D Register Jump

Variable Data Field

LBYT, Ao, Ee, LI, CL Qo RI	m, b, v	Load byte*
SBYT, Ao, Ee, LI, CL Qo RI	m, b, v	Store byte*
SCAN, Qo, Ee, EQ GT LT NE LE GE	m, b, v	Scan byte**

*In LBYT and SBYT, the modifiers Ao or Qo and Ee are required; if neither LI or RI appears, no indexing will be assumed.

**In SCAN, the modifiers Qo, Ee, and one of the comparison modifiers are required.

Input/Output

CONN	x, e, u, n	Connect
EXTF	x, w, n	External function
BEGR	x, (a) m, n	Begin read*
BEGW	x, (a) m, n	Begin write*

*If the bank term is missing, it is assumed ($).

Operation Field	Address Field	Instruction
COPY, CW, CWA	x, b	Copy status
CLCH	x	Clear channel
IPA		Input to A
ALG	w	Perform algorithm

Input/Output Control Words

IOSW, C	(a) m, w	Skip words (write zeros under word count control)*
IOTW, C	(a) m, w	Transmit data under word count control*
IOSR, C	(a) m, w	Skip words (write zeros) under word count or to end of record (and write end of record)*
IOTR, C	(a) m, w	Transmit data under word count or to end of record (and write end of record)*
IOJP	(a) m	Jump to (a) m for next control word*

*If the bank term is missing, it is assumed ($).

Others

INF	w	Internal function
NOP		No operation
ENO	a	Enter operand bank register (single precision augment instruction in upper or lower half word)
00	m, b	Octal instruction from 00-77
.	or	
.		
77	(a) m	

TABLE 5

PSEUDO INSTRUCTIONS

Mnemonic	Use	Page
BANK	Declare subprogram and common block banks	12
BCD	Insert BCD characters	16
BES	Reserve block of storage	14
BLOCK	Specify block of common	14
BSS	Reserve block of storage	13
CODAP	Change input to CODAP-1 format	19
COMMON	Declare array in common	14
COMPASS	Change input to COMPASS format	19
COSY	COSY identification	32
DEC	Insert single precision decimal constants	16
DECD	Insert double precision decimal constants	16
DELETE	Delete portions of program	31
ECHO	Replicate a sequence	22
EJECT	Eject a page on the output listing	25
END	Specify the end of a subprogram	11
ENDM	Terminate a macro-definition	27
ENTRY	Define entry points in a subprogram	12
EQU	Equate an undefined symbol to a defined symbol	19
EXT	Define external symbols	12
IDENT	Identify the subprogram by name	11
IFF	Control pseudo instruction	21
IFN	Control pseudo instruction	20
IFT	Control pseudo instruction	20
IFZ	Control pseudo instruction	19
INSERT	Insert changes in a program	31
LIBM	Declare library macros	27
LIST	Resume output listing	25
MACRO	Define a macro	26

TABLE 5 (cont'd)

Mnemonic	Use	Page
MACRO Instruction	Call a macro	27
NOLIST	Suppress output listing	26
OCT	Insert octal constants	15
ORGR	Set location counter	21
REM	Insert remarks on the output listing	26
REPLACE	Replace portions of a program	31
SCOPE	Terminates assembly process	22
SPACE	Insert spaces in the output listing	25
TITLE	Title pages with program name	25
TYPE	Insert typewriter codes	16
VFD	Assign data in variable byte sizes	17

TABLE 6

SPECIAL CODES FOR TYPE ENTRIES

BCD Characters	Type Equivalent
*R	Carriage Return
*U	Shift to Upper Case
*L	Shift to Lower Case
*B	Backspace
*T	Tab
*X	!
*A	'
*S	;

TABLE 7

ERROR CODES

A	Address Field Error. An A error will occur if there is a format error in the address field.
C	Presetting Common Error. A C error will occur if the subprogram attempts to preset numbered common or if data overflows a labeled common block. Processing and listing continues, but binary output is suppressed until an ORGR record occurs.
D	Duplicate Symbol. A D error results if a symbol occurs more than once in a location field of a subprogram. The symbol will be ignored on the second and subsequent occurrences.
F	Full Symbol Table. No assignment is made if a table entry would cause overflow of the symbol table, the block common name table, or linkage address table.
L	Location Field Error. An L error occurs if a symbol appears where none is allowed, if a symbol is absent where one is required, or if an illegal type symbol appears.
M	Modifier Error. An M error will result if an illegal modifier appears, if modifiers appear where none are allowed, or if a required modifier is absent.
O	Operation Code Error. An O error occurs if an illegal operation code is used. Zeros are substituted.
R	Range Error. The range of lines within IFF, IFN, IFT, or IFZ tests exceeds range of macro, echo or subprogram.
U	Undefined Symbol. A symbol referenced in the address field has not been defined; zeros are substituted.

TABLE 8

ADDRESS SUBFIELDS

Relocatable or fixed		Number of Bits
a	first bank designator	3
i	second bank designator	3
m	first operand address	15
n	second operand address	15
y	operand	15
Fixed only		
b	first index register	3
e	equipment designator	3
g	bit designator	6
p	first source register	5
q	second source register	5
r	destination register	5
u	unit designator	9
v	second index register	3
w	operand	15
x	channel number	16

PROGRAMMING TRAINING CENTERS

3330 Hillview Ave.
Palo Alto, California

•

8100 - 34th Ave. South
Minneapolis, Minnesota

•

11428 Rockville Pike
Rockville, Maryland

•

Room 223, Terminal Building
Newark Airport
Newark, New Jersey

•

5630 Arbor Vitae
Los Angeles 45, California

COMPASS Reference Manual October 21, 1963

Publication #525 a

Page

Literal

7 A double precision decimal constant must be in floating point format.

9 FORCING UPPER Should Read: "The instruction has a TYPE 1 or TYPE 2 symbol in the location field."

12 Replace the discussions of ENTRY and EXT with the following:

ENTRY

Symbols to be referenced by other subprograms must be declared as entry points within the subprogram that defines these symbols. (A symbol is defined when it appears in the location field of a machine or pseudo instruction or as array name in a COMMON instruction.) Symbols are declared as entry points by placing them in the address field of one or more ENTRY pseudo instructions. Two or more symbols in the address field are separated by commas. No spaces (blanks) can appear within a string of symbols, as a space indicates the end of the string. The address field of the ENTRY pseudo instruction may be extended out to column 72. The location field must be blank.

Example:

```
ENTRY    SYM1, SYM2, SYM3
```

SYM1, SYM2, and SYM3 can now be referenced by other subprograms.

EXT

Symbols used by a subprogram which are defined in another subprogram are declared external symbols by placing them in the address field of one or more EXT pseudo instructions contained in the user subprogram. For example, to use the symbols SYM1, SYM2 and SYM3 declared as entry symbols in another subprogram, the pseudo instruction would be written as:

```
EXT    SYM1, SYM2, SYM3
```

The address field may be extended to column 72; symbols are separated from each other by commas. No space (blanks) can appear in a string of symbols, as a space indicates the end of the string. The location field of an EXT must be blank.

Page

14 Include the discussion below as an introduction to BLOCK and COMMON:
BLOCK and COMMON pseudo instructions reserve storage areas that can be referenced by more than one subprogram. They are placed at the beginning of a subprogram.

If the location symbol in the BLOCK pseudo instruction is alphanumeric, the storage area is reserved at the beginning of the subprogram and is termed labeled common. If the location symbol is numeric or blank, the storage area is assigned a separate portion of core storage termed numbered common. Constants can be assembled into a labeled common area, but not into a numbered common area.

COMMON pseudo instructions describe data arrays within an area assigned by BLOCK. To reference a block of common storage reserved in another subprogram, the location field symbol of the BLOCK pseudo instruction must be identical in each subprogram, and the length of the two blocks must be the same.

A location symbol of BLOCK is never referenced by any other instruction.

15 Add, after the last sentence in the OCT description:
"If fewer than 16 digits are written, they will be right-justified in the word with leading zeros inserted."

DEC

In the sentence beginning "Each subfield is a decimal constant...", delete "a value of up to 14 decimal digits" and replace it with: "a string of decimal digits".

DECD

Second sentence, delete "may consist of up to 28 decimal digits, and will".

BCD and TYPE

If a BCD or TYPE statement appears in the range of a MACRO or ECHO instruction its address field may consist of a single formal parameter name. The octal parameter which corresponds to this parameter must have the form n, 8 n characters.

In the range of a MACRO or ECHO instruction, the following statements are illegal; P is a formal parameter: BCD P, characters; TYPE P, characters.

Page

16, 17 BCD and TYPE

Delete "where $\leq$ n $\leq$ 6" under BCD. The value of n is limited only by the number of characters which can be punched into a single card. The characters may begin immediately after the comma but must end before column 73.

17 VFD

Add after the 0 mode description: "n may not exceed 48."
In the next to the last sentence, the H mode description should read: "Any printable character may appear in the v field unless the VFD instruction appears in the range of a MACRO or ECHO instruction; in this case, blanks are not permitted in the v field."

23 The first complete sentence on the page should begin:
"The IDENT, EQU, END, MACRO, ENDM, BLOCK and ECHO pseudo instruction are excluded from the range of ECHO;....."

25 OUTPUT LISTING

Title: Replace the TITLE discussion with the following:
A TITLE pseudo instruction may appear anywhere in a program between the IDENT and END instructions. Columns 16-72 of the first TITLE instruction, no matter where it appears, will be printed on top of the first page of the program listing and on the top of all subsequent pages until another TITLE instruction is encountered.

27 MACRO INSTRUCTION

The following SCOPE mnemonics used in system macros to indicate standard units, interrupts, and equipment designation may not be used in system macros for any other purpose.

ACC	EXUN	LO	PUN
ADDR	HY	M1604	RO
BCD	ICM	MANUAL	RW
BIN	INP	OCM	SCR
BY	INST	OPER	SHIFT
DIVIDE	LGO	OUT	SV
EXON	LIB	OVER	TRACE

32 COSY Text Cards

Change all occurrences of the phrase: "high order digit" to "high order octal digit" and change "lower order digit" to "lower order octal digits". In the COSY text cards format, columns 5-80 should be defined: "19 words of compressed symbolic program instructions."

CONTROL DATA SALES OFFICES

ALBUQUERQUE • BEVERLY HILLS • BIRMINGHAM • BOSTON
CHICAGO • CLEVELAND • DALLAS • DAYTON
DENVER • DETROIT • HONOLULU • HOUSTON
HUNTSVILLE • ITHACA • KANSAS CITY • LOS ALTOS • MINNEAPOLIS • NEWARK
NEW YORK CITY • NORFOLK • ORLANDO • PALO ALTO • PHILADELPHIA • SAN DIEGO
SAN FRANCISCO • SEATTLE • WASHINGTON, D.C.

INTERNATIONAL OFFICES

BAD HOMBURG, GERMANY • MELBOURNE, AUSTRALIA • LUCERNE, SWITZERLAND
STOCKHOLM, SWEDEN • ZURICH, SWITZERLAND • PARIS, FRANCE
MEXICO CITY, MEXICO • OTTAWA, CANADA

CONTROL DATA
CORPORATION

8100 34th AVENUE SOUTH, MINNEAPOLIS 20, MINNESOTA

Pub. No. 525a