60492800

CONTROL DATA CORPORATION

COMPASS VERSION 3 INSTANT

CDC® OPERATING SYSTEMS:
NOS 1
NOS 2
NOS/BE 1
SCOPE 2

60492800

CONTROL DATA CORPORATION

COMPASS VERSION 3 INSTANT

CDC® OPERATING SYSTEMS:
NOS 1
NOS 2
NOS/BE 1
SCOPE 2

REVISION RECORD

Revision	Description
A (04/12/76)	Original Printing.
B (05/15/78)	This revision documents COMPASS Version 3.5, PSR level 472.
C (08/06/79)	Version 3.6, PSR level 498.
D (06/25/82)	This revision documents support of the CYBER 170 model 825, 835, 855, 865, and 875 Computer Systems at PSR level 552. Extensive editorial changes were also made. This revision supersedes all previous revisions.

REVISION LETTERS I, O, Q, AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

©COPYRIGHT CONTROL DATA CORPORATION 1976, 1978, 1979, 1982
All Rights Reserved
Printed in the United States of America

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision
Front Cover	-
Title Page	-
ii	D
iii/iv	D
v/vi	D
vii	D
viii	D
1 thru 117	D
Back Cover	D

PREFACE

This instant provides a convenient summary of the COMPASS Version 3.6 language as implemented under the following operating systems:

NOS 1 for the CONTROL DATA® CYBER 170 Computer Systems; CYBER 70 models 71, 72, 73, and 74; and 6000 Computer Systems

NOS 2 for the CDC® CYBER 170 Computer Systems; CYBER 70 models 71, 72, 73, and 74; and 6000 Computer Systems

NOS/BE 1 for the CDC CYBER 170 Computer Systems; CYBER 70 models 71, 72, 73, and 74; and 6000 Computer Systems

SCOPE 2 for the CDC CYBER 170 model 176; CYBER 70 model 76; and 7600 Computer Systems

This instant is written for a programmer familiar with the COMPASS language and the operating system under which the COMPASS assembler is operating.

NOTE

Avoid continued use of COMPASS in creating application programs when possible. COMPASS and other machine-dependent languages can complicate migration to future hardware and software systems. Software mobility will be restricted by continued use of COMPASS for stand-alone programs, COMPASS subroutines embedded in programs using higher-level languages, and COMPASS owncode routines used with CDC standard products.

More detailed information can be found in the following publication:

Publication	Publication Number
COMPASS Version 3 Reference Manual	60492600

CDC manuals can be ordered from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

CONTENTS

COMPASS CONTROL STATEMENT 1

SOURCE STATEMENT FORMAT 5
- Format Conventions 5
- Format Requirements 5
  - First Character Position 6
  - Location Field 6
  - Operation Field 6
  - Variable Field 6
  - Comments Field 7

STATEMENT EDITING 8
- Concatenation 8
- Micro Substitution 8

NAMES 9

SYMBOLS 10
- Linkage Symbols 10
- Default Symbols 11
- Qualified Symbols 11

CENTRAL PROCESSOR REGISTER DESIGNATORS 12

SPECIAL ELEMENTS 13

DATA NOTATION 14
- Data Representation 14
  - Data Items 14
  - Constants 14
  - Literals 14
  - Expressions 15
- Character Data Notation 15
- Numeric Notation 15

PSEUDO INSTRUCTIONS 19

SYSTEM MICROS 58
- DATE 58
- JDATE 58
- TIME 58
- BASE 58
- CODE 58
- QUAL 58
- SEQUENCE 58
- MODLEVEL 59
- PCOMMENT 59

CENTRAL PROCESSOR INSTRUCTIONS 60

COMPARE/MOVE INSTRUCTIONS 85

PERIPHERAL PROCESSOR INSTRUCTIONS 88

COMMON COMMON DECKS 102

DIAGNOSTICS 107

CHARACTER SETS 111

FIGURES

1 Character Data Notation 16
2 Numeric Data Notation 18
3 15- and 30-Bit Central Processor Instruction Formats 60
4 CC, CU, DM, and MD Formats 87
5 Peripheral Processor Instruction Formats 88

TABLES

1 COMPASS Control Statement Parameters 1
2 Pseudo Instructions 20
3 Central Processor Instructions (Numerical Listing) 62
4 Central Processor Instructions (Alphabetical Listing) 76
5 Central Processor Instructions That Force Upper 84
6 Compare/Move Instructions 86
7 Peripheral Processor Instructions (Numerical Listing) 90
8 Peripheral Processor Instructions (Alphabetical Listing) 96
9 Common Common Decks 103
10 Fatal Errors 107
11 Informative Errors 110
12 Standard Character Sets 112

COMPASS CONTROL STATEMENT

The COMPASS control statement causes the COMPASS assembler to be loaded from the library and executed. The COMPASS control statement has the following form:

```
COMPASS(parameter-list)
```

where parameter-list is a list of optional parameters separated by commas. See table 1 for the allowable parameters.

TABLE 1. COMPASS CONTROL STATEMENT PARAMETERS

Parameter	Option	Significance
A	omitted	Do not abort
	A	Abort job to EXIT(S) (NOS/BE or SCOPE 2) or EXIT (NOS) if fatal errors occur; ignored if D is present
B	omitted or B	Binary written on LGO
	B = 0	No binary
	B = filename	Binary written on filename
BL	omitted or BL = 0	No burstable listing
	BL	Generate burstable listing
D	omitted	No debug mode
	D	Debug mode (A parameter suppressed)
E	omitted	Error list on OUTPUT
	E	Error list on ERRS
	E = filename	Error list on filename
	E = 0	No error list

TABLE 1. COMPASS CONTROL STATEMENT PARAMETERS (Contd)

Parameter	Option	Significance
F	omitted or F	Special element *F is 0
	F = n	Special element *F is n (decimal digit)
	F = name	Special element *F is a decimal digit as follows: name — *F COMPASS — 0 RUN — 1 FTN4 — 2 FTN5 — 3
G†	omitted or G = 0	No system text
	G	System text on file named SYSTEXT
	G = filename	System text is first overlay on filename
	G = filename/ovl	System text is named overlay on filename
I	omitted	Source on INPUT
	I	Source on COMPILE
	I = filename	Source on filename
	I = 0	Illegal
L	omitted or L	Full list on OUTPUT
	L = filename	List on filename
	L = 0	No full list

TABLE 1. COMPASS CONTROL STATEMENT PARAMETERS (Cont'd)

Parameter	Option	Significance
LO	omitted or LO = 0	Selects B, L, N and R
	LO	Selects C, F, G, X
	LO = string	Selects or deselects options in string. If included, B, L, N, and R are deselected; other options are selected. Nine options maximum. See LIST pseudo instruction for options.
	LO = $$$$	Select all options
ML	omitted or ML	MODLEVEL equal to JDATE
	ML = string	MODLEVEL equal to string (9 characters maximum)
N	omitted	Normal ejects
	N	No explicit ejects
O	omitted or O	Short list on OUTPUT
	O = filename	List on filename
	O = 0	No short list
P	omitted	New pagination on END
	P	Continue pagination
PC	omitted or PC	PCOMMENT is 30 blanks
	PC = string	PCOMMENT is 30-character string

TABLE 1. COMPASS CONTROL STATEMENT PARAMETERS (Contd)

Parameter	Option	Significance
PD	omitted or not 6 or 8	Defaults to IP.PD lines per inch (IP.PD is an installation parameter.)
	PD = 6	Print six lines per inch
	PD = 8	Print eight lines per inch
PS	omitted	If PD also omitted, defaults to IP.PS. If PD specified, PS=(PD*IP.PS)/IP.PD (IP.PS and IP.PD are installation parameters.)
	PS = x	Page size is x lines per page ($4 \leq x \leq 99$)
S†	omitted	Text on SYSTEXT overlay in system library
	S	Load overlay named SYSTEXT from job's global library set
	S = 0	No system text
	S = ovl	Text is named overlay on global library set
	S = lib/ovl	Text is named overlay on specified library
X	omitted	Text on OLDPL file
	X = filename	Text on filename
	X = 0	Illegal
	X	Text on OPL file

†Maximum of 7 S and G parameters allowed.

SOURCE STATEMENT FORMAT

COMPASS statements consist of the location field, operation field, variable field, and comments field. The fields are not restricted to specific character positions of the line.

FORMAT CONVENTIONS

The following format conventions are optional. The were established to standardize the appearance of COMPASS programs.

Column	Contents
1	Comma, blank, or asterisk.
2-9	Location field entry, or plus or minus (left justified)
10	Blank
11-16	Operation field entry (left justified)
17	Blank
18-29	Variable field entry (left justified)
30	Beginning of comments

FORMAT REQUIREMENTS

The COMPASS assembler imposes less rigid requirements on statement format than the optional conventions previously described. Fields are not restricted to specific character positions but must be separated by blanks. If the location, operation, and variable fields are empty, the line is treated as a comment line. The location, operation, and variable fields must not extend beyond character position 72.

FIRST CHARACTER POSITION

The first character position of a line determines the type of line:

, (comma)

Continuation line; maximum of nine per statement

* (asterisk)

Comment line

other

New statement

LOCATION FIELD

The location field begins in character position 1 or 2 and contains a symbol, name, + (force upper), or - (negate force upper). The location field is terminated by one or more blanks.

OPERATION FIELD

The operation field begins with the first nonblank character following the location field. The operation field can start in character position 3 when the location field is empty. The operation field contains a central processor or peripheral processor mnemonic operation code, a pseudo instruction, or a macro name, or the field is blank. The operation field is terminated by one or more blanks.

VARIABLE FIELD

The variable field begins with the first nonblank character following the operation field. This field must not extend beyond character position 72. The field is empty if it does not begin before the comments field character position set by the COL pseudo instruction. The variable field is terminated by one or more blanks.

The variable field contains data items, expressions, register designators, names, special elements, or entries defined for the instruction.

COMMENTS FIELD

The beginning of the comments field is determined as follows:

If the variable field is empty, the comments field begins in the comments field character position (position 30 unless changed by the COL pseudo instruction).

If the variable field is not empty, the comments field begins with the first nonblank character following the variable field.

The comments field ends with the last non-blank character on the line.

STATEMENT EDITING

COMPASS reads statements from the source file, and immediately edits and interprets each statement (unless the statement is a comment line or is part of a definition). COMPASS performs two types of editing: concatenation and micro substitution.

CONCATENATION

The concatenation character (CDC graphic character ↱; ASCII graphic character _) delimits a substitutable parameter when no other delimiter is present. Each concatenation character is removed before the statement is interpreted.

MICRO SUBSTITUTION

Pairs of micro marks (CDC graphic character ≠; ASCII graphic character ") delimit a micro reference. Editing replaces the reference, including the micro marks, with a character string. The empty string replaces ≠≠.

NAMES

Names identify programs, overlays, blocks, macros, micros, remote code, or conditional or duplicated sequences.

Names are terminated by a comma or blank and can be one through eight characters long, with the following exceptions:

External linkage names are limited to a maximum of seven characters.

Central processor subprogram and overlay names are limited to a maximum of seven characters. These names must begin with a letter (A through Z).

Peripheral processor subprogram and overlay names for CYBER 170 Computer Systems or CYBER 70 models 72, 73, or 74 are limited to a maximum of three characters.

Peripheral processing unit subprogram and overlay names for the CYBER 70 model 76 are limited to a maximum of seven characters.

SYMBOLS

A symbol is a set of one through eight characters that identifies a value and its associated attributes.

A symbol cannot include the following characters:

+	plus
-	minus
*	asterisk
Δ	blank
/	slash
,	comma
↱	concatenation character
^	caret

A symbol cannot begin with:

$	dollar sign
=	equals sign
:	colon
1-9	digit

The symbols An, Bn, Xn or A.m, B.m, X.m, are reserved as register designators in central processor programs.

LINKAGE SYMBOLS

Linkage symbols are external symbols and entry point symbols. These symbols are limited to seven characters. The first character must be a letter (A through Z); the last character must not be a colon (:).

DEFAULT SYMBOLS

Default symbols are preceded by =S, =X, or =Y and are defined as follows:

=Ssymbol

If the symbol is not defined conventionally, COMPASS assigns an address to the symbol; otherwise, the conventional definition is used.

=Xsymbol

If the symbol is not defined in the program, COMPASS assumes the symbol is a strong external; otherwise, the program definition is used.

=Ysymbol

If the symbol is not defined or referenced as =Xsymbol, COMPASS assumes the symbol is a weak external; otherwise, the program definition or the =X reference takes precedence.

QUALIFIED SYMBOLS

A symbol defined when a symbol qualifier is in effect during assembly can be referenced outside the qualifier sequence as

/qualifier/symbol

Unqualified symbols within a qualifier sequence can be referenced as //symbol. The QUAL pseudo instruction defines a qualifier sequence.

=Ssymbol is qualified by the qualifier that is in effect. =Xsymbol and =Ysymbol are unqualified.

CENTRAL PROCESSOR REGISTER DESIGNATORS

The following register designators are used in central processor programs:

Register Type	Designation
Address	An or A.m
Index	Bn or B.m
Operand	Xn or X.m

In these designations, n is a single digit in the range 0 through 7; m is a symbol or unsigned integer constant with a value in the range 0 through 7.

SPECIAL ELEMENTS

The following designators reference special elements:

* or *L

Location counter

*O

Origin counter

*F

Absolute value obtained as follows:

0 COMPASS assembler called by control statement

1 COMPASS assembler called by the FORTRAN RUN compiler (no longer supported)

2 COMPASS assembler called by the FTN4 compiler

3 COMPASS assembler called by the FTN5 compiler.

Special element *F can be defined differently by the F parameter in the COMPASS control statement.

*P

Position counter. Number of bits remaining in word: 1 through 60 for central processor programs; 1 through 12 for peripheral processor programs.

$

Position counter minus 1

DATA NOTATION

Data notation allows you to enter numeric and character string values.

DATA REPRESENTATION

Character and numeric data can be represented as data items, constants, literals, or expressions.

DATA ITEMS

Data items are used in subfields of DATA and LIT, as specifications of length in VFD, and in register designators.

CONSTANTS

A constant is an expression element consisting of a value in either octal, decimal, hexadecimal, or character notation. Constants have these characteristics:

The first character of the constant, as expressed in the program, is numeric.

The field size is determined by the context in which the constant is used. The maximum field size is 60 bits (no double-precision floating-point numbers).

LITERALS

A literal is a read-only constant that is stored in the literals block. When a literal is used in an expression, the expression is evaluated using the address of the literal rather than the value.

Literals are specified either as data items in the LIT pseudo instruction or as expression elements preceded by =.

EXPRESSIONS

An expression is defined using elements and terms:

An element is a symbol, constant, special element, register designator, or literal.

A term is one or more elements joined by * (multiplication) or / (division).

An expression is one or more terms joined by + (addition), - (subtraction) and ∧ (logical difference).

Register operators immediately precede register designators and can be +, -, *, or /.

CHARACTER DATA NOTATION

Figure 1 shows the notations for character string data items, constants, and literals.

NUMERIC NOTATION

Figure 2 shows the notations for numeric data items, constants, and literals.

Data Item

sign	n	type	string

or

sign	type	d	string	d

Constant

n	type	string

Literal

=	sign	n	type	string

or

=	sign	type	d	string	d

sign — - + or omitted

n — Required for constant. Used for data item or literal not delimited by d

0 Data item or literal terminated by , or blank

Constant terminated by + - , * / blank or ^

n Decimal number of characters in the string

Figure 1. Character Data Notation
(Sheet 1 of 2)

type	Character string justification; for constant, C, L, and Z are the same. C Left justified, zero fill. 12 zero bits at the right of the last word guaranteed for data item or literal; no zero bits guaranteed for constant H Left justified, blank fill A Right justified, blank fill R Right justified, zero fill L Left justified, zero fill Z Left justified, zero fill. 6 zero bits at the right of the last word guaranteed for data item or literal; no zero bits guaranteed for constant
d	Characters between first and second occurrence of delimiting character comprise the string; d cannot be ↦ or =
string	n characters following type or characters delimited by d

Figure 1. Character Data Notation
(Sheet 2 of 2)

Data Item

sign	preradix	value	modifiers

Constant

value	modifiers

Literal

=	sign	preradix	value	modifiers

sign — −, +, or omitted

preradix — Optional for data items and literals; cannot be used for constants

- omitted — Radix from assembly base or postradix
- B or O — Octal notation
- D — Decimal notation

value — Octal or decimal digits optionally consisting of an integer, an octal or decimal point, and a fraction

Octal or decimal point denotes floating point value in central processor assemblies only

modifiers — Optional modifiers in any sequence

- postradix — Same as preradix
- E±n, En, or E — Single precision power of 10
- EE±n, EEn, or EE — Double precision power of 10
- S±n, Sn, or S — Power of 2 scale factor
- P±n, Pn, or P — Binary point position

Figure 2. Numeric Data Notation

PSEUDO INSTRUCTIONS

COMPASS pseudo instructions are shown in table 2. Pseudo instructions have the same format as symbolic machine instructions; however, pseudo instructions control the actions of the assembler at assembly time, rather than the actions of the machine at execution time.

In the placement column of table 2, first group refers to the first statement group. The first statement group must preceed any statements that define symbols, allocate storage, or generate object code.

TABLE 2. PSEUDO INSTRUCTIONS

Location	Operation	Variable	Description	Placement
	ABS		Declares central processor program as absolute; EXT, LCC, REP, REPC, and REPI are illegal.	First group; CP absolute
mname	BASE	mode	Sets or changes the mode of numeric data to octal (O), decimal (D), mixed (M), or the previous base (*). Default is D. A name in the location field defines a micro mname whose value is one character (O, D, or M) representing the current base.	Anywhere
symbol	BSS	aexp	Reserves the number of words of storage specified in the variable field.	Not first group
symbol	BSSZ	aexp	Reserves the number of words of storage specified in the variable field and zeros them.	Not first group; not blank common
	B1=1		Declares that B1 contains 1; changes the code generated by the R= instruction, and defines the symbol B1=1.	Anywhere; CP only

	B7=1		Declares that B7 contains 1; changes the codes generated by the R= instruction, and defines the symbol B7=1.	Anywhere; CP only
	CHAR	exp1,exp2	Redefines character codes when CODE 0 (for other) is in effect.	
mname	CODE	char	Declares that all character data constants and items are to be generated in the code specified by char: A ASCII six-bit subset D Display code E External BCD I Internal BCD 0 Other code, defined by CHAR pseudo instructions * The code that was previously in effect If mname is specified, a micro is defined with a value representing the current code (A, D, E, I, or 0).	Anywhere

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
	COL	n	Sets the number of the column in which the comments field can begin where n≥12. Default is 30.	Not first group
	COMMENT	string	Inserts string (70 characters maximum) in 8th through 14th words of PRFX table in binary output. COMMENT is meaningless if NOLABEL is used.	Anywhere
symbol	CON	expression-list	Generates 60-bit fields for a central processor assembly or 12-bit fields for a peripheral processor assembly.	Not first group; not blank common
syntax	CPOP	ctl,val,reg,type	Generates an operation code table entry that specifies the format of the central processor machine instruction. syntax Up to 3 descriptors separated by commas [r is register (A, B, or X); Q is expression]:	Anywhere; CP only

	void	r1+r2	r1+r2Q
	Q	r1-r2	r1-r2Q
	r	r1*r2	r1*r2Q
	rQ	r1/r2	r1/r2Q
	-r	-r1+r2	-r1+r2Q
	-rQ	-r1-r2	-r1-r2Q
		-r1*r2	-r1*r2Q
		-r1/r2	-r1/r2Q

ctl	0	15-bit instruction
	1	30-bit instruction
	2	15-bit instruction; force upper before
	3	30-bit instruction; force upper before
	4	15-bit instruction; force upper after

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
syntax (Contd)	CPOP (Contd)		5 30-bit instruction; force upper after 6 15-bit instruction; force upper before and after 7 30-bit instruction; force upper before and after val 9-bit operation code reg 3 octal digits specifying order of registers 0 No register 1 Operation field register 2 Second or only register 3 First of two registers	

			type Type of instruction 6 CYBER 70 Model 71, 72, 73, 74; CYBER 170 Computer Systems; or 6000 Computer Systems 7 CYBER 70 Model 76 or 7600 other No restriction	
syntax1	CPSYN	syntax2	Renders the central processor instruction described by syntax1 synonymous with the central processor or OPDEF instruction described by the syntax2.	Anywhere; CP only
name	CTEXT	string	Sets XTEXT flag. Optionally includes new subtitle in the variable field and new sub-subtitle in location field.	Not first group; CP only
symbol	DATA	item-list	Generates numeric and character data one or more words per item.	Not first group; not blank common
mname	DECMIC	aexp,n	Using a decimal conversion, converts the value of aexp into a character string. n (optional) defines length of micro.	Anywhere

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
symbol	DIS	n,string	Places string in n words.	Not first group; not blank common
symbol	DIS	,dstringd	Places string in words needed for data plus two zero characters. Character d delimits string.	Not first group; not blank common
dupname	DUP	rep,lnct	Causes sequence of lines following the statement to be assembled rep times. Range set by an ENDD or by lnct.	Not first group
dupname	ECHO	lnct,p1=(list1), p2=(list2), ...,Pn=(listn)	Causes sequence of lines following the statement to be assembled one or more times. Parameters include a set of formal substitutable parameters and list of actual parameters. Shortest list determines the number of assemblies. Range can be set by an ENDD or by a statement count. STOPDUP can end the assembly.	Not first group

name	EJECT		Advances paper, prints page heading, and continues. Location field optionally specifies new sub-subtitle.	Anywhere
ifname	ELSE	lnct	Terminates IF that is skipping (acts like ENDIF) or initiates skipping. An ifname (optional) matches a corresponding IF. Range set by ENDIF or lnct.	Anywhere
symbol	END	trasym	Terminates a subprogram deck and optionally defines the execution address (trasym) for a relocatable program; symbol assigned lwa+1. For a relocatable program, trasym must be an entry point.	Required last
dupname	ENDD		Defines range of DUP and ECHO; dupname (optional) matches a corresponding DUP or ECHO. If ENDD is unnamed, terminates all DUP and ECHO ranges.	Anywhere
ifname	ENDIF		If the location field contains a name, terminates the named IF sequence. If ENDIF is unnamed, terminates any IF sequence in effect. ENDIF is ignored in a range controlled by a line count.	Anywhere

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
mname	ENDM		Terminates MACRO, MACROE, or OPDEF definition; mname (optional) matches a corresponding MACRO or MACROE. If ENDM is unnamed, terminates all macro definitions.	Anywhere
	ENDX		Clears XTEXT flag.	Not first group
	ENTRY	symbol-list	Declares entry points to the source program.	Not first group; CP only
	ENTRYC	symbol-list	Conditionally declares entry points to the source program.	Not first group; CP only
symbol	EQU or =	exp	Assigns the value specified by the expression in the variable field to the symbol in the location field. Once defined by EQU or =, a symbol cannot be redefined.	Not first group

flag	ERR		Sets error flag indicated by a character (A, D, E, F, L, O, P, R, N, U, or V, or 1 through 9. Default is P.	Not first group
flag	ERRxx	aexp	Sets error flag if relational mnemonic xx is true for aexp. xx True Condition ZR zero NZ nonzero PL positive (including positive zero) MI negative (including negative zero) NG negative (including negative zero) The error flag is indicated by a character (see ERR above).	Not first group
	EXT	symbol-list	Declares external symbols used by the subprogram as weak externals.	Not first group; relocatable only

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
rmtname	HERE		Causes assembly of the named saved remote code.	Anywhere
	IDENT	name	Identifies CPU relocatable subprogram. name 1 through 7 character central processor name; first character alphabetic	First line relocatable only
	IDENT	name,origin, entry,p,s	Identifies central processor absolute subprogram or overlay. name 1 through 7 character central processor name; first character alphabetic origin First word address entry Subprogram entry address p,s Primary and secondary overlay numbers	First line and anywhere

	IDENT	name,origin, entry,ppu	Identifies 7600 peripheral processing unit subprogram or overlay. name 1 through 7 character 7600 peripheral processing unit name origin First word address entry Subprogram entry address ppu peripheral processing unit number	First line and anywhere
	IDENT	name,origin	Identifies 6000 Computer System peripheral processor subprogram or overlay. name 1- through 3-character 6000 Computer System peripheral processor name origin First word address	First line and anywhere
	IDENT		Generates partial binary output and starts a new set of USE blocks.	Not first group; absolute CP only

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
ifname	IF	att,exp,lnct	Tests exp according to att and assembles if att is true. However, a minus prefix to att causes assembly on a false condition. Range set by lnct or ENDIF. att True Condition ABS exp is absolute COM exp is common relocatable DEF All symbols in exp are defined EXT External symbol in exp LCM exp is LCM or ECS address LOC exp is program relocatable MAC Opcode name in second subfield MIC Micro name in field REG Register symbol in exp REL exp is relocatable SET Field contains set symbol SST Field contains system symbol	Not first group

ifname	IFC	op,dstring1 dstring2d,1nct	Compares two character strings according to op (see IFop). The single character d delimits string1 and string2. A minus prefix causes assembly on a false rather than true condition. Range set by 1nct or ENDIF.	Anywhere
ifname	IFop	exp1,exp2,1nct	Tests exp1 against exp2 according to op. Range set by 1nct or ENDIF. op — True Condition EQ — Equal NE — Not equal GT — Greater than GE — Greater than or equal LT — Less than LE — Less than or equal	Not first group
ifname	IFMI	exp,1nct	Tests for negative expression, including -0. Range set by 1nct or ENDIF.	Not first group
ifname	IFPL	exp,1nct	Tests for positive expression, including +0. Range set by 1nct or ENDIF.	Not first group

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
ifname	IFtype	lnct	Tests for processor type and assembles if right type. type — True Condition CP — Any central processor CP6 — Neither PERIPH nor PPU nor MACHINE 7 was used. Code is assembled for CYBER 170 Computer Systems; CYBER 70 model 71, 72, 73, 74; or 6000 Computer Systems central processor. CP7 — Neither PERIPH nor PPU nor MACHINE 6 was used. Code is assembled for CYBER 70 model 76 or 7600 central processor. PP — PERIPH or PPU was used.	

			PP6 One of the following: PERIPH but not MACHINE 7 PPU and MACHINE 6 Code is assembled for CYBER 170 Computer Systems; CYBER 70 model 71, 72, 73, or 74; or 6000 Computer Systems peripheral processor. PP7 One of the following: PPU but not MACHINE 6 PERIPH and MACHINE 7 Code is assembled for CYBER 70 model 76 or 7600 peripheral processing unit.	
	IRP	parameter	Indefinitely repeats inside macros. The first IRP of the pair names the repeated parameter. The reference to the macro contains a list of one or more subparameters for the repeated parameter. IRP pairs cannot be nested at the same macro level.	Inside a macro
	LCC	directive	Specifies a directive for the loader; valid in relocatable CPU assemblies only.	Not first group; CP relocatable

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
	LDSET	op-list	Generates loader LDSET directives for a relocatable program. Location field symbols are ignored. LIB — Clear local library set LIB=libname — Add specified library to local library set; separate multiples with a slash. MAP — Write load map to file OUTPUT. MAP=p — Write selected load map items to file OUTPUT. For NOS and NOS/BE, p can be the letter N or any combination of SBEX: N — No map S — Statistics B — Block list E — Entry point list X — Cross reference map	Anywhere

	LDSET (contd)	op-list (contd)		For SCOPE 2, p can be one of the following: 0 or O No map S Statistics B Statistics and block list E Statistics, block list, and entry point list X Statistics, block list, entry point list, and cross reference map.
			MAP=p/lfn	Write selections on named file.
			MAP=/lfn	Write installation default selections on named file.
			PS=p	Select page size for load map; p can be decimal 10 through 999999. Not supported on SCOPE 2.
			PD=p	Select print density of six or eight lines per inch for load map. Not supported on SCOPE 2.

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
	LDSET (Contd)	op-list (Contd)	PRESET=a Preset memory to a 1- to 20-digit octal number plus optional + or - prefix or B suffix; or one of the following: NONE No presetting for ECS (or for LCM and SCM under SCOPE 2); same as ZERO for CM ZERO 0000 0000 0000 0000 0000 ONES 7777 7777 7777 7777 7777 INDEF 1777 0000 0000 0000 0000 INF 3777 0000 0000 0000 0000 NGINDEF 6000 0000 0000 0000 0000 NGINF 4000 0000 0000 0000 0000	

ALTZERO 2525 2525 2525 2525 2525

ALTONES 5252 5252 5252 5252 5252

DEBUG 6000 0000 0004 0040 0000

PRESETA=a Same as for PRESET; lower 17 bits (CM/SCM) or lower 24 bits (ECS/LCM) of each word contains its address.

ERR=e Select loader abort as follows:

ALL	All loader errors
FATAL	Fatal loader errors
NONE	Catastrophic loader errors

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description		Placement
EPT= eptname				Declare symbol eptname as an entry point of a CAPSULE or OVCAP. Separate names with a slash.	
			NOEPT= eptname	Do not declare eptname as an entry point of a CAPSULE or OVCAP. Separate names with a slash.	
			REWIND	Reset default for load files. NR on LOAD or SLOAD overrides for single files.	
			NOREWIN	Reset default for load files. R on LOAD or SLOAD overrides for single files.	
			USEP=pname	Load module pname unconditionally; separate names with a slash.	

			USE=eptname	Load module containing entry point eptname unconditionally; separate names with a slash.
			SUBST=pair	Treat external references to eptname1 as if they were eptname2 with pair as: eptname1-eptname2 Separate pairs with a slash.
			OMIT=eptname	Omit loading modules with eptname; separate names with a slash.
	LIST	op-list	Controls list output when COMPASS control L mode is non-zero and LO is not used or is other than LO=0. When L=0, listing contains error lines and error directory only. One or more options are indicated in the variable field. A minus prefix to an option discontinues it.	Anywhere

TABLE 2. PSEUDO INSTRUCTIONS

Location	Operation	Variable	Description	Placement
	LIST (Contd)	op-list (Contd)	op Output A Assembly; list edited and unedited lines B List binary control statements C Control statement list D Detail E Echoed lines F IF-skipped lines G Code generation list L Normally selected; when cancelled, only error-flagged lines and LIST instructions are listed. M Macro lines; nonsystem	

			N Normally selected; when cancelled, normal symbols without references are not listed in the cross-reference table. R Normally selected; when cancelled, the assembler does not accumulate or list reference table information. S Macro lines; system T Selection causes SST-defined symbols without references to be listed. X XTEXT lines and lines bracketed by CTEXT and ENDX	
	LIST	$	All options	Anywhere
	LIST	*	Selects options specified by previous LIST	Anywhere
symbol	LIT	item-list	Enters data item into the literals block. No entry is generated for string matching string already in the table.	Not first group
	LOC	exp	Causes symbolic addresses to be defined according to a location counter value; differs from origin counter value.	Not first group
	LOCAL	symbols	Lists symbols local to definition.	Macro or opdef

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
	MACHINE	type,hf-list	Specifies processor type and required hardware features. type Processor 6 CYBER 70 model 71, 72, 73, or 74; CYBER 170 Computer Systems; or 6000 Computer Systems 7 CYBER 70 model 76 or 7600 8 CYBER 170 model 825, 835, 855, 865, or 875; 8 can be specified only when S=AIDTEXT is not specified on the COMPASS control statement. hf-list List of required hardware features; maximum of 9 allowed	First group

			C Compare/Move Unit D Distributive data path I Integer multiply instruction L ECS/LCM/UEM X Central and Monitor exchange jumps	
mname	MACRO	parameters	Either form identifies the beginning of macro definition. In the first form, macro name is in location field; in the second form, the macro name is first subfield in the variable field. Additional subfields (optional) specify parameter names.	Anywhere
	MACRO	mname,parameters		
mname	MACROE	parameters	Use instead of MACRO. MACROE saves parameter names so they can be identified by name instead of sequence in the macro call. Like MACRO, it has two forms.	Anywhere
	MACROE	mname,parameters		
symbol	MAX	exp-list	Assigns largest expression value to symbol; symbol is redefinable.	Not first group
symbol	MICCNT	mname	Defines symbol as absolute redefinable value equal to number of characters in named micro.	Anywhere

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
micname	MICRO	n1,n2,dstringd	Defines a micro string and assigns a name; n1 specifies starting character, n2 specifies number of characters, d delimits string.	Anywhere
symbol	MIN	exp-list	Assigns the smallest expression value to symbol. Symbol is redefinable.	Not first group
	NIL		Does nothing; allows disabling of peripheral processor instruction, or pseudo instruction.	Anywhere
	NOLABEL	I	Suppresses prefix tables in absolute assemblies only. I suppresses 77 tables only. Otherwise, 77, 50, and 51 tables, and peripheral processor header are suppressed.	Anywhere
	NOREF	symbol-list	Suppresses symbols in the symbolic reference table. If a symbol is a qualifier enclosed by //, all symbols defined under that qualifier are suppressed.	Anywhere

micname	OCTMIC	aexp,n	Converts aexp from octal number to character string. n (optional) defines the length of the string.	Anywhere
syntax	OPDEF	parameters	Identifies OPDEF definition; syntax describes operation variable fields of OPDEF call (see CPOP for syntax).	Anywhere; CP only
name1	OPSYN	name2	Renders name1 synonymous with name2. Names are pseudo operations, macro names, or PPU instruction names.	Anywhere
	ORG	exp	Indicates block to be used and value to which origin counter is set. Absolute value indicates use of the absolute block.	Not first group
	ORGC	exp	Conditionally indicates block to be used; otherwise, same as above.	

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
namel (Contd)	PERIPH	J	Declares code is for a 6000 Computer System peripheral processor. IDENT cannot contain peripheral processor number. ENTRY, ENTRYC, SEG, EXT, REP, REPC, REPI, LCC, or CPU instructions are illegal. J alters the way COMPASS assembles r on UJN, ZJN, NJN, or PJN instructions. If J is not specified, COMPASS tests range of r against the short jump limit (±31). If r is in range, jump is assembled using the value of r. Otherwise, COMPASS tests r minus the location counter value. If value is in range, jump is assembled using r minus location counter value. The J option deletes the first test.	First group; PP only
	POS	aexp	Sets position to absolute value ≤ word length.	Not first group
name	PPOP	ctl,val,type	Generates operation code table entry that specifies the format of a PPU instruction.	Anywhere; PP only

ctl	Assembly control:
	0 Illegal
	1 24 bits; 12-bit address; no indexing
	2 12 bits; relative or absolute address (such as UJN)
	3 24 bits; 18-bit address (such as LDC)
	4 12 bits; 6-bit address (such as LDN)
	5 24 bits; 12-bit address; optional indexing (such as LDM)
	6 12 bits; signed relative address (such as SHN)
	7 24 bits; 12-bit address; required second field (such as IAM)
val	12-bit operation code

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
name (Contd)	PPOP (Contd)		type Type of instruction 6 CYBER 170 Computer Systems; CYBER 70 Model 71, 72, 73, or 74; or 6000 Computer Systems 7 CYBER 70 Model 76 other No restriction	
	PPU	J	Declares code is for 7600 peripheral processing unit. Alters interpretation of IDENT by allowing peripheral processing unit number. ENTRY, ENTRYC, EXT, REP, REPC, REPI, LCC, SEG, or CPU instructions are illegal. A J produces same effect as J for PERIPH.	First group; PP only
	PURGDEF	syntax	Disables central processor operation code entry for the syntax described.	Anywhere
	PURGMAC	name-list	Disables operation code entries for the named instructions.	Anywhere

mname	QUAL	qualifier	Declares that symbols defined after the occurrence of a QUAL block be referenced as /qualifier/ symbol. If the variable field contains an asterisk, symbols are defined using the previous qualifier. A name in the location field defines a micro mname whose value is the current qualifier (possibly null).	Anywhere
	REP, REPC and REPI	S/saddr, D/daddr, C/rep,B/bsz, I/inc	Replicates object code at load time in relocatable central processor programs only. REPI causes the loader to generate copies when it encounters the object code repeat load table. REPC is the same as REPI except the copying is conditional. REP causes replication at end-of-load. Parameters specify source code address, destination address, repetition count, code block size, and increment size.	Not first group; CP relocatable
rmtname	RMT		Delimits code (with another RMT instruction) to be saved for later assembly; rmtname (optional) is significant on first RMT of pair. RMT pairs cannot be nested at the same macro level.	Anywhere

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
symbol	R=	reg,exp	Use with B1=1 or B7=1 to generate no code, a 15-bit increment instruction, or a 30-bit instruction. S and reg are concatenated to form the instruction operation code. For example, if reg is A2, the instruction SA2 is formed. The second subfield specifies the operand register or value expression. If the value of exp is 0, the operand is B0. If B1=1 or B7=1 and exp is 1, 2, or -1, the operand field of the generated instruction is Bn, Bn+Bn, or -Bn respectively. Bn represents B1 or B7 depending on whether B1=1 or B7=1 was issued. If the second subfield is the same two characters as the first subfield, no instruction is generated. In all other cases, the operand is the register or value indicated by the expression.	Not first group; CP only

	SEG		Causes write of all binary information accumulated since the previous IDENT, SEGMENT, or SEG instruction. It is illegal in peripheral processor and relocatable central processor assembly.	Not first group; CP absolute only
name	SEGMENT	origin,entry, p,s	Produces overlays during assembly of absolute programs. Subfields specify first word address and an entry point. Location field indicates name of overlay. For a central processor assembly, p and s are the overlay level numbers; default (1,0).	Not first group; absolute only
symbol	SET	exp	Assigns exp value to symbol. The symbol can be redefined.	Not first group
ifname	SKIP	lnct	Unconditionally skips source statements. Range set by ENDIF or lnct.	Anywhere
name	SPACE	scnt,rcnt	Spaces scnt lines before next line. If less than rcnt lines remain following spacing, the page is ejected. name (optional) specifies new sub-subtitle.	Anywhere
	SST	symbol-list	Defines symbols from a system text overlay (specified by the S or G parameter on the COMPASS control statement). Listed symbols are excluded from the table.	Anywhere

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
rname	STEXT		Generates as binary output a system text overlay containing symbols, micros, and macros defined in the current program. Location field optionally contains record name.	First group
	STOPDUP		Terminates a DUP or ECHO duplication regardless of iteration count.	Not first group
name	TITLE	string	Defines the primary title, which appears on every page, or generates subtitles. If no TITLE is specified, the variable field on the IDENT instruction becomes the main title. Location field (optional) specifies new subsubtitle.	Anywhere
name	TTL	string	Defines the new primary title, clears the subtitle, and optionally specifies a new subsubtitle (location field entry).	Anywhere

	USE	block	Identifies or establishes block. 0 or blank — Nominal block // — Blank common; only BSS, ORG, ORGC, POS and LOC are allowed in relocatable central processor code /name/ — Labeled common name — Local block * — Previous block	Not first group
	USELCM	block	Initiates or continues use of ECS/LCM/UEM block. 0 or blank — Illegal // — Blank common block /name/ — Labeled common block name — Local block * — Previous block	Not first group; CP only

TABLE 2. PSEUDO INSTRUCTIONS (Contd)

Location	Operation	Variable	Description	Placement
name (Contd)	USELCM (Contd)		In an absolute central processor assembly, data cannot be preset in an ECS/LCM/UEM block.	
symbol	VFD	item1/exp1, item2/exp2,..., itemn/expn	Generates fields of binary data and packs them into consecutive words starting with the next available bit position. Item destination field length is 60 bits maximum.	Not first group; not blank common
	XREF	string	Causes symbolic cross reference table to contain addresses of references either instead of the page and number or in addition to them on the listing. If string begins with A, table lists addresses only. If string begins with B, table lists both addresses, and page and line numbers. If string begins with P, table lists page and line numbers only.	Anywhere

file	XTEXT	rname	Assembles source statements from a system-logical record on either a sequential file, an indexed file with named records, or an Update or Modify random-access program library file. If file is omitted, the file specified by the X parameter in the COMPASS control statement is used. The name of the record is given by rname and can only be specified if the file has named records; default is the first record in the file.	

SYSTEM MICROS

The following paragraphs describe micros that are predefined by the COMPASS assembler.

DATE

DATE returns the 10-character date in the form Δ yr/mo/dy. or Δ mo/dy/yr., depending on installation parameters. The date returned begins with a blank and ends with a period.

JDATE

JDATE returns the five-digit Julian date in the form yyddd.

TIME

TIME returns the time in the form Δ hr.mn.sc. The time is 10 characters long beginning with a blank and ending with a period.

BASE

BASE returns the current base (D, M, or O). The base is initially D.

CODE

CODE returns the current code (A, D, E, I, or O). The code is initially D.

QUAL

QUAL returns the current zero- to eight-character qualifier symbol. The qualifier symbol is initially null.

SEQUENCE

SEQUENCE returns 18 characters comprising the sequence field (character positions 73 through 90) of the most recent source statement.

MODLEVEL

MODLEVEL returns up to nine characters specified by the ML parameter on the COMPASS control statement. When no ML parameter is specified, MODLEVEL is identical to JDATE.

PCOMMENT

PCOMMENT returns the value specified by the PC parameter on the COMPASS control statement; when no PC parameter is specified, PCOMMENT returns 30 blanks. When COMPASS is called by a language processor (such as FORTRAN), a string of characters supplied by the language processor is returned; this string usually contains information about compiler options.

CENTRAL PROCESSOR INSTRUCTIONS

The following paragraph describe the central processor instructions. Compare/move and peripheral processor instructions are described later.

Figure 3 illustrates the formats for 15- and 30-bit central processor instructions and the possible arrangements of instructions in a 60-bit central processor word.

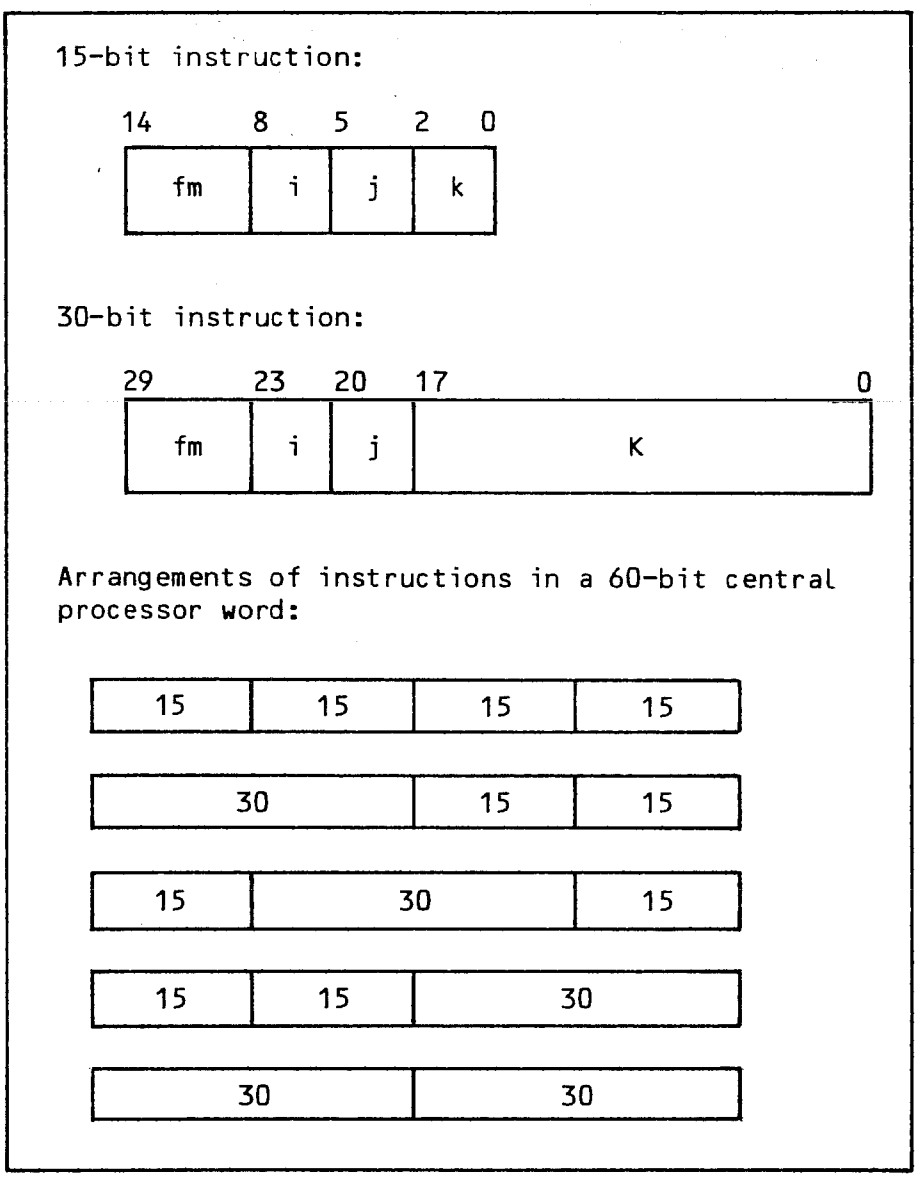


Figure 3. 15- and 30-Bit Central Processor Instruction Formats

The central processor instructions are shown in tables 3 and 4. Table 3 lists the instructions numerically by operation code; table 4 lists them alphabetically by mnemonic. The symbols used in tables 3 and 4 are:

A One of eight address registers (18 bits).

B One of eight index registers (18 bits). B0 is fixed and equal to positive zero.

fm Instruction code (6 bits).

i One of eight designated registers (3 bits).

j One of eight designated registers (3 bits).

jk A constant that indicates the number of shifts to be taken (6 bits).

k One of eight designated registers (3 bits).

K A constant that indicates the branch destination or operand (18 bits).

X One of eight operand registers (60 bits).

Table 5 lists the central processor instructions that force upper.

Some instructions in existing COMPASS programs are not valid for execution on the CYBER 170 models 825, 835, 855, 865, and 875. You can specify S=AIDTEXT in the COMPASS control statement to detect these instructions; COMPASS flags the invalid instructions with a type 0 error. S=AIDTEXT should not be specified if the 8 option is chosen for the MACHINE pseudo instruction.

An integer division macro IXi Xj/Xk is provided for central processor assemblies. This macro produces several words of code.

TABLE 3. CENTRAL PROCESSOR INSTRUCTIONS (NUMERICAL LISTING)

Operation Code	Mnemonic	Variable	Description	Model 74 and 6600/ 6700 Unit	Model 175, 176, 740, 750, 760, Model 76, and 7600 Unit	Notes
00000	ES		Error exit to EEA	---	None	2,6
00000	ES	K	Error exit to EEA	---	None	2,6
0000K	PS	K	Program Stop	Branch	---	3
0100K	RJ	K	Return Jump to K	Branch	None	
011jK	RL	Bj±K	Block-copy K plus (Bj) words from LCM to SCM	---	None	4
011jK	RE	Bj±K	Read extended core storage	Branch	---	3
012jK	WL	Bj±K	Block-copy K plus (Bj) words from SCM to LCM	---	None	4
012jK	WE	Bj±K	Write extended core storage	Branch	None	3

01300	MJ		Exchange-exit to NEA if exit flag clear	---	None	4
01300	XJ		Central exchange jump to MA	None	None	3
013jK	MJ	Bj±K	Exchange-exit to K plus (Bj) if exit flag set	---	None	2,6
013jK	XJ	Bj±K	Central exchange jump to (Bj)±K	Branch	---	3,6
014jk	RXj	Xk	Read LCM at (Xk) to Xj	---	None	4
014jk	RXj	Xk	Read UEM at (Xk)+RA_e to Xj	---	---	1
015jk	WXj	Xk	Write (Xj) into LCM at (Xk)	---	None	4
015jk	WXj	Xk	Write (Xj) to UEM at (Xk)+RA_e	---	---	1
0160k	RI	Bk	Reset channel (Bk) input buffer	---	None	4,6
016j0	TBj		Set Bj to current clock time	---	None	4
016jk	IBj	Bk	Read channel (Bk) input status to Bj if j≠0; otherwise, same as RI	---	None	4
0170k	RO	Bk	Reset channel (Bk) output buffer	---	None	4,6

TABLE 3. CENTRAL PROCESSOR INSTRUCTIONS (NUMERICAL LISTING) (Contd)

Operation Code	Mnemonic	Variable	Description	Model 74 and 6600/ 6700 Unit	Model 175, 176, 740, 750, 760, Model 76, and 7600 Unit	Notes
017jK	OBj	Bk	Read channel (Bk) output status to Bj if j≠0; otherwise, same as RO	---	None	4
02i0K	JP	Bi±K	Jump to (Bi)±K	Branch	None	5
030jK	ZR	Xj,K	Branch to K if (Xj) = 0	Branch	None	5
031jK	NZ	Xj,K	Branch to K if (Xj) ≠ 0	Branch	None	5
032jK	PL	Xj,K	Branch to K if (Xj) is plus	Branch	None	5
033jK	NG	Xj,K	Branch to K if (Xj) negative	Branch	None	5
033jK	MI	Xj,K	Branch to K if (Xj) is minus	Branch	None	5
034jK	IR	Xj,K	Branch to K if (Xj) in range	Branch	None	5
035jK	OR	Xj,K	Branch to K if (Xj) not in range	Branch	None	5

036jK	DF	Xj,K	Branch to K if (Xj) definite	Branch	None	5
037jK	ID	Xj,K	Branch to K if (Xj) indefinite	Branch	None	5
0400K	EQ	K	Branch to K	Branch	None	
0400K	ZR	K	Branch to K	Branch	None	
04i0K	ZR	Bi,K	Branch to K if (Bi) = 0	Branch	None	
04ijK	EQ	Bi,Bj,K	Branch to K if (Bi) = (Bj)	Branch	None	
05i0K	NZ	Bi,K	Branch to K if (Bi) $\neq$ 0	Branch	None	
05ijK	NE	Bi,Bj,K	Branch to K if (Bi) $\neq$ (Bj)	Branch	None	
06i0K	GE	Bi,K	Branch to K if (Bi) $\geq$ 0	Branch	None	
06ijK	GE	Bi,Bj,K	Branch to K if (Bi) $\geq$ (Bj)	Branch	None	
06i0K	PL	Bi,K	Branch to K if (Bi) $\geq$ 0	Branch	None	
06ijK	LE	Bj,Bi,K	Branch to K if (Bj) $\leq$ (Bi)	Branch	None	
060jK	LE	Bj,K	Branch to K if (Bj) $\leq$ 0	Branch	None	
070jK	GT	Bj,K	Branch to K if (Bj) > 0	Branch	None	

TABLE 3. CENTRAL PROCESSOR INSTRUCTIONS (NUMERICAL LISTING) (Contd)

Operation Code	Mnemonic	Variable	Description	Model 74 and 6600/ 6700 Unit	Model 175, 176, 740, 750, 760, Model 76, and 7600 Unit	Notes
07i0K	NG	Bi,K	Branch to K if (Bi) < 0	Branch	None	
07i0K	MI	Bi,K	Branch to K if (Bi) < 0	Branch	None	
07i0K	LT	Bi,K	Branch to K if (Bi) < 0	Branch	None	
07ijK	LT	Bi,Bj,K	Branch to K if (Bi) < (Bj)	Branch	None	
07ijK	GT	Bj,Bi,K	Branch to K if (Bj) > (Bi)	Branch	None	
10ijj	BXi	Xj	Transmit (Xj) to Xi	Boolean	Boolean	
11ijk	BXi	Xj*Xk	Logical Product of (Xj) and (Xk) to Xi	Boolean	Boolean	
12ijk	BXi	Xj+Xk	Logical sum of (Xj) and (Xk) to Xi	Boolean	Boolean	
13ijk	BXi	Xj-Xk	Logical difference of (Xj) and (Xk) to Xi	Boolean	Boolean	

14ikk	BXi	-Xk	Transmit complement of (Xk) to Xi	Boolean	Boolean	
15ijk	BXi	-Xk*Xj	Logical product of (Xj) and complement of (Xk) to Xi	Boolean	Boolean	
16ijk	BXi	-Xk+Xj	Logical sum of (Xj) and complement of (Xk) to Xi	Boolean	Boolean	
17ijk	BXi	-Xk-Xj	Logical difference of (Xj) and complement of (Xk) to Xi	Boolean	Boolean	
20ijk	LXi	±jk	Logical shift (Xi) by ±jk places	Shift	Shift	
21ijk	AXi	±jk	Arithmetic shift (Xi) by ±jk places	Shift	Shift	
22ijk	LXi	Bj,Xk	Logically shift (Xk) by (Bj) places to Xi	Shift	Shift	
22ijk	LXi	Xk,Bj	Logically shift (Xk) by (Bj) places to Xi	Shift	Shift	
22iji	LXi	Bj	Logically shift (Xi) by (Bj) places to Xi	Shift	Shift	
22i0K	LXi	Xk	Transmit (Xk) to Xi	Shift	Shift	
22ijk	AXi	-Bj,Xk	Logically shift (Xi) by (Bj) places to Xk	Shift	Shift	

TABLE 3. CENTRAL PROCESSOR INSTRUCTIONS (NUMERICAL LISTING) (Contd)

Operation Code	Mnemonic	Variable	Description	Model 74 and 6600/ 6700 Unit	Model 175, 176, 740, 750, 760, Model 76, and 7600 Unit	Notes
22ijk	AXi	Xk,-Bj	Logically shift (Xi) by (Bj) places to Xk	Shift	Shift	
22iji	AXi	-Bj	Logically shift (Xi) by (Bj) places to Xi	Shift	Shift	
23ijk	AXi	Bj,Xk	Arithmetic right shift (Xk) by (Bj) places to Xi	Shift	Shift	
23ijk	AXi	Xk,Bj	Arithmetic right shift (Xk) by (Bj) places to Xi	Shift	Shift	
23iji	AXi	Bj	Arithmetic right shift (Xk) by (Bj) places to Xi	Shift	Shift	
23i0K	AXi	Xk	Transmit (Xk) to Xi	Shift	Shift	
23ijk	LXi	-Bj,Xk	Arithmetic right shift (Xi) by (Bj) places to Xk	Shift	Shift	

23ijk	LXi	Xk,-Bj	Arithmetic right shift (Xi) by (Bj) places to Xk	Shift	Shift	
23iji	LXi	-Bj	Arithmetic right shift (Xi) by (Bj) places to Xi	Shift	Shift	
24ijk	NXi	Bj,Xk	Normalize (Xk) in Xi and Bj	Shift	Normalize	
24ijk	NXi	Xk,Bj	Normalize (Xk) in Xi and Bj	Shift	Normalize	
24i0i	NXi		Normalize (Xi) to Xi	Shift	Normalize	
24ijk	NXi,Bj	Xk	Normalize (Xk) to Xi and Bj	Shift	Normalize	
24iji	NXi,Bj		Normalize (Xi) to Xi and Bj	Shift	Normalize	
24iji	NXi	Bj	Normalize (Xi) to Xi; shift count to Bj	Shift	Normalize	
25i0k	ZXi	Xk	Round and normalize (Xk) to Xi	Shift	Normalize	
25ijk	ZXi	Bj,Xk	Round and normalize (Xk) to Xi; shift count to Bj	Shift	Normalize	
25ijk	ZXi	Xk,Bj	Round and normalize (Xk) to Xi; shift count to Bj	Shift	Normalize	

TABLE 3. CENTRAL PROCESSOR INSTRUCTIONS (NUMERICAL LISTING) (Contd)

Operation Code	Mnemonic	Variable	Description	Model 74 and 6600/ 6700 Unit	Model 175, 176, 740, 750, 760, Model 76, and 7600 Unit	Notes
25ijk	ZXi,Bj	Xk	Round and normalize (Xk) to Xi; shift count to Bj	Shift	Normalize	
25i0i	ZXi		Round and normalize (Xk) to Xi	Shift	Normalize	
25iji	ZXi	Bj	Round and normalize (Xi) to Xi; shift count to Bj	Shift	Normalize	
25iji	ZXi,Bj		Round and normalize (Xi) to Xi; shift count to Bj	Shift	Normalize	
26ijk	UXi	Bj,Xk	Unpack (Xk) to Xi and Bj	Shift	Boolean	
26ijk	UXi	Xk,Bj	Unpack (Xk) to Xi and Bj	Shift	Boolean	
26ijk	UXi,Bj	Xk	Unpack (Xk) to Xi and Bj	Shift	Boolean	
26i0i	UXi		Unpack (Xi) to Xi	Shift	Boolean	

26iji	UXi	Bj	Unpack (Xi) to Xi and Bj	Shift	Boolean	
26iji	UXi,Bj		Unpack (Xi) to Xi and Bj	Shift	Boolean	
27ijk	PXi	Bj,Xk	Pack Xi from (Xk) and (Bj)	Shift	Boolean	
27ijk	PXi	Xk,Bj	Pack Xi from (Xk) and (Bj)	Shift	Boolean	
27ijk	PXi,Bj	Xk	Pack Xi from (Xk) and (Bj)	Shift	Boolean	
27i0i	PXi		Pack (Xi) to Xi	Shift	Boolean	
27iji	PXi	Bj	Pack (Xi) and (Bj) to Xi	Shift	Boolean	
27iji	PXi,Bj		Pack (Xi) and (Bj) to Xi	Shift	Boolean	
30ijk	FXi	Xj+Xk	Floating sum of (Xj) and (Xk) to Xi	FP Add	FP Add	
31ijk	FXi	Xj-Xk	Floating difference (Xj) and Xk to Xi	FP Add	FP Add	
32ijk	DXi	Xj+Xk	Floating double precision sum of (Xj) and (Xk) to Xi	FP Add	FP Add	
33ijk	DXi	Xj-Xk	Floating double precision difference of (Xj) and (Xk) to Xi	FP Add	FP Add	
34ijk	RXi	Xj+Xk	Round floating sum of (Xj) and (Xk) to Xi	FP Add	FP Add	

TABLE 3. CENTRAL PROCESSOR INSTRUCTIONS (NUMERICAL LISTING) (Contd)

Operation Code	Mnemonic	Variable	Description	Model 74 and 6600/ 6700 Unit	Model 175, 176, 740, 750, 760, Model 76, and 7600 Unit	Notes
35ijk	RXi	Xj-Xk	Round floating difference of (Xj) and (Xk) to Xi	FP Add	FP Add	
36ijk	IXi	Xj+Xk	Integer sum of (Xj) and (Xk) to Xi	Long Add	Long Add	
37ijk	IXi	Xj-Xk	Integer difference of (Xj) and (Xk) to Xi	Long Add	Long Add	
40ijk	FXi	Xj*Xk	Floating product of (Xj) and (Xk) to Xi	Multiply	Multiply	
41ijk	RXi	Xj*Xk	Rounded floating product of (Xj) and (Xk) to Xi	Multiply	Multiply	
42ijk	IXi	Xj*Xk	Integer product of (Xj) and (Xk) to Xi	Multiply	Multiply	
42ijk	DXi	Xj*Xk	Floating double precision product of (Xj) and (Xk) to Xi	Multiply	Multiply	

43ijk	MXi	±jk	Form mask in Xi, ±jk bits	Shift	Shift	
44ijk	FXi	Xj/Xk	Floating divide (Xj) by (Xk) to Xi	Divide	Divide	
45ijk	RXi	Xj/Xk	Round floating divide (Xj) by (Xk) to Xi	Divide	Divide	
46n	NO	n	No operation	None	None	
47ikk	CXi	Xk	Count the 1's in (Xk) to Xi	Divide	Pop	
50ijk	SAi	Aj±K	Set Ai to (Aj)±K	Increment	Increment	
51ijk	SAi	Bj±K	Set Ai to (Bj)±K	Increment	Increment	
52ijk	SAi	Xj±K	Set Ai to (Xj)±K	Increment	Increment	
53ijk	SAi	Xj+Bk	Set Ai to (Xj)+(Bk)	Increment	Increment	
54ijk	SAi	Aj+Bk	Set Ai to (Aj)+(Bk)	Increment	Increment	
55ijk	SAi	Aj-Bk	Set Ai to (Aj)-(Bk)	Increment	Increment	
56ijk	SAi	Bj+Bk	Set Ai to (Bj)+(Bk)	Increment	Increment	
57ijk	SAi	Bj-Bk	Set Ai to (Bj)-(Bk)	Increment	Increment	
60ijK	SBi	Aj±K	Set Bi to (Aj)±K	Increment	Increment	

TABLE 3. CENTRAL PROCESSOR INSTRUCTIONS (NUMERICAL LISTING) (Contd)

Operation Code	Mnemonic	Variable	Description	Model 74 and 6600/ 6700 Unit	Model 175, 176, 740, 750, 760, Model 76, and 7600 Unit	Notes
61ijK	SBi	Bj±K	Set Bi to (Bj)±K	Increment	Increment	
62ijK	SBi	Xi±K	Set Bi to (Xj)±K	Increment	Increment	
63ijk	SBi	Xj+Bk	Set Bi to (Xj)+(Bk)	Increment	Increment	
64ijk	SBi	Aj+Bk	Set Bi to (Aj)+(Bk)	Increment	Increment	
65ijk	SBi	Aj-Bk	Set Bi to (Aj)-(Bk)	Increment	Increment	
660jk	CR	Xj,Xk	Read CM at (Xk) to Xj	None	None	1
66ijk	SBi	Bj+Bk	Set Bi to (Bj)+(Bk)	Increment	Increment	
670jk	CW	Xj,Xk	Write Xj to CM at (Xk)	None	None	1
67ijk	SBi	Bj-Bk	Set Bi to (Bj)-(Bk)	Increment	Increment	
70ijK	SXi	Aj±K	Set Xi to (Aj)±K	Increment	Increment	

71ijK	SXi	Bj±K	Set Xi to (Bj)±K	Increment	Increment	
72ijK	SXi	Xj±K	Set Xi to (Xj)±K	Increment	Increment	
73ijk	SXi	Xj+Bk	Set Xi to (Xj)+(Bk)	Increment	Increment	
74ijk	SXi	Aj+Bk	Set Xi to (Aj)+(Bk)	Increment	Increment	
75ijk	SXi	Aj-Bk	Set Xi to (Aj)-(Bk)	Increment	Increment	
76ijk	SXi	Bj+Bk	Set Xi to (Bj)+(Bk)	Increment	Increment	
77ijk	SXi	Bj-Bk	Set Xi to (Bj)-(Bk)	Increment	Increment	

1 CYBER 170 models 825, 835, 855, 865, and 875.

2 CYBER 70 model 76; and 7600.

3 CYBER 170 Computer Systems; CYBER 70 models 71, 72, 73, and 74; 6000 Computer Systems.

4 CYBER 170 model 176; CYBER 70 model 76; and 7600.

5 Go to K±Bi; for CYBER 70 model 74 and 6600/6700 tests made in increment unit.

Go to K if Xj; for CYBER 70 model 74 and 6600/6700 tests made in add unit.

6 Privileged to Monitor.

TABLE 4. CENTRAL PROCESSOR INSTRUCTIONS (ALPHABETICAL LISTING)

Mnemonic	Variable	Description	Operation Code	Notes
AXi	±jk	Arithmetic shift (Xi) by ±jk places	21ijk	
AXi	Bj,Xk	Arithmetic right shift (Xk) by (Bj) places to Xi	23ijk	
AXi	Bj	Arithmetic right shift (Xi) by (Bj) places to Xi	23iji	
AXi	-Bj,Xk	Logically shift (Xi) by (Bj) places to Xk	22ijk	
AXi	Xk,-Bj	Logically shift (Xi) by (Bj) places to Xk	22ijk	
AXi	-Bj	Logically shift (Xi) by (Bj) places to Xi	22iji	
BXi	Xj	Transmit (Xj) to Xi	10ijj	
BXi	Xj*Xk	Logical product of (Xj) and (Xk) to Xi	11ijk	
BXi	Xj+Xk	Logical sum of (Xj) and (Xk) to Xi	12ijk	
BXi	Xj-Xk	Logical difference of (Xj) and (Xk) to Xi	13ijk	
BXi	-Xk	Transmit the complement of (Xk) to Xi	14ikk	
BXi	-Xk*Xj	Logical product of (Xj) and complement of (Xk) to Xi	15ijk	
BXi	-Xk+Xj	Logical sum of (Xj) and complement of (Xk) to Xi	16ijk	
BXi	-Xk-Xj	Logical difference of (Xj) and complement of (Xk) to Xi	17ijk	

CR	Xj,Xk	Read CM at (Xk) to Xj	660jk	1
CW	Xj,Xk	Write Xj to CM at (Xk)	670jk	1
CXi	Xk	Count ones in (Xk) to Xi	47ikk	
DF	Xj,K	Branch to K if (Xj) definite	036jK	5
DXi	Xj+Xk	Floating double precision sum of (Xj) and (Xk) to Xi	32ijk	
DXi	Xj-Xk	Floating double precision difference of (Xj) and (Xk) to Xi	33ijk	
DXi	Xj*Xk	Floating double precision product of (Xj) and (Xk) to Xi	42ijk	
EQ	K	Branch to K	0400K	
EQ	Bi,Bj,K	Branch to K if (Bi) = (Bj)	04ijK	
ES		Error exit to EEA	00000	2,6
ES	K	Error exit to EEA	00000	2,6
FXi	Xj+Xk	Floating sum of (Xj) and (Xk) to Xi	30ijk	
FXi	Xj-Xk	Floating difference (Xj) and (Xk) to Xi	31ijk	
FXi	Xj*Xk	Floating product of (Xj) and (Xk) to Xi	40ijk	
FXi	Xj/Xk	Floating divide (Xj) by (Xk) to Xi	44ijk	
GE	Bi,Bj,K	Branch to K if (Bi) $\geq$ (Bj)	06ijK	
GE	Bi,K	Branch to K if (Bi) $\geq$ 0	06i0K	

TABLE 4. CENTRAL PROCESSOR INSTRUCTIONS (ALPHABETICAL LISTING) (Contd)

Mnemonic	Variable	Description	Operation Code	Notes
GT	Bj,Bi,K	Branch to K if (Bj) > (Bi)	07ijK	
GT	Bj,K	Branch to K if (Bj) > 0	070jK	
IBj	Bk	Input channel (Bk) status to Bi	016jk	4
ID	Xj,K	Branch to K if (Xj) indefinite	037jK	5
IR	Xj,K	Branch to K if (Xj) in range	034jK	5
IXi	Xj+Xk	Integer sum of (Xj) and (Xk) to Xi	36ijk	
IXi	Xj-Xk	Integer difference of (Xj) and (Xk) to Xi	37ijk	
IXi	Xj*Xk	Integer product of (Xj) and (Xk) to Xi	42ijk	
JP	Bi±K	Jump to (Bi)±K	02i0K	5
LE	Bj,Bi,K	Branch to K if (Bj) ≤ (Bi)	06ijK	
LE	Bj,K	Branch to K if (Bj) ≤ 0	060jK	
LT	Bi,Bj,K	Branch to K if (Bi) < (Bj)	07ijK	
LT	Bi,K	Branch to K if (Bi) < 0	07i0K	
LXi	±jk	Logically shift (Xi) by ± jk places	20ijk	

LXi	Bj	Logically shift (Xi) by (Bj) places to Xi	22iji	
LXi	Bj,Xk	Logically shift (Xk) by Bj places to Xi	22ijk	
LXi	-Bj,Xk	Arithmetic right shift (Xi) by (Bj) places to Xk	23ijk	
LXi	Xk,-Bj	Arithmetic right shift (Xi) by (Bj) places to Xk	23ijk	
LXi	-Bj	Arithmetic right shift (Xi) by (Bj) places to Xi	23iji	
MI	Xj,K	Branch to K if (Xj) negative	033jK	
MI	Bi,K	Branch to K if (Bi) < 0	07i0K	
MJ		Exchange jump to NEA (if exit flag clear)	01300	4
MJ	Bj±K	Exchange jump to Bj±K (if exit flag set)	013jK	2,6
MXi	±jk	Form mask in Xi, ± jk bits	43ijk	
NE	Bi,Bj,K	Branch to K if (Bi) ≠ (Bj)	05ijK	
NG	Bi,K	Branch to K if (Bi) < (B0)	07i0K	
NG	Xj,K	Branch to K if (Xj) negative	033jK	5
NO	n	No operation	46n	
NXi	Bj,Xk	Normalize (Xk) in Xi and Bj	24ijk	
NXi		Normalize (Xi) to Xi	24i0i	
NXi	Bj	Normalize (Xi) to Xi; shift count to Bj	24iji	
NZ	Bi,K	Branch to K if (Bi) ≠ (B0)	05i0K	
NZ	Xj,K	Branch to K if (Xj) ≠ 0	031jK	5

TABLE 4. CENTRAL PROCESSOR INSTRUCTIONS (ALPHABETICAL LISTING) (Contd)

Mnemonic	Variable	Description	Operation Code	Notes
OBj	Bk	Channel (Bk) output status to Bj	017jK	4
OR	Xj,K	Branch to K if (Xj) out of range	035jK	5
PL	Xj,K	Branch to K if (Xj) positive	032jK	5
PL	Bi,K	Branch to K if (Bi) $\geq$ (B0)	06i0K	
PS	K	Program stop	0000K	3
PXi	Bj,Xk	Pack Xi from (Xk) and (Bj)	27ijk	
PXi		Pack (Xi) to Xi	27i0i	
PXi	Bj	Pack (Xi) and (Bj) to Xi	27iji	
RE	Bj$\pm$K	Read extended core storage	011jK	3
RI	Bk	Reset input channel (Bk) buffer	0160k	4,6
RJ	K	Return jump to K	0100K	
RL	Bj$\pm$K	Block copy K + (Bj) words from LCM to SCM	011jK	4
RO	Bk	Reset channel (Bk) output buffer	0170k	4,6
RXi	Xj+Xk	Round floating sum of (Xj) and (Xk) to Xi	34ijk	

RXi	Xj-Xk	Round floating difference of (Xj) and (Xk) to Xi	35ijk	
RXi	Xj*Xk	Round floating product of (Xj) and (Xk) to Xi	41ijk	
RXi	Xj/Xk	Round floating divide (Xj) by (Xk) to Xi	45ijk	
RXj	Xk	Read LCM at (Xk) to Xj	014jk	4
RXj	Xk	Read UEM at (Xk) + RA_e to Xj	014jk	1
SAi	Aj±K	Set Ai to (Aj) ± K	50ijK	
SAi	Bj±K	Set Ai to (Bj) ± K	51ijK	
SAi	Xj±K	Set Ai to (Xj) ± K	52ijK	
SAi	Xj+Bk	Set Ai to (Xj) + (Bk)	53ijk	
SAi	Aj+Bk	Set Ai to (Aj) + (Bk)	54ijk	
SAi	Aj-Bk	Set Ai to (Aj) - (Bk)	55ijk	
SAi	Bj+Bk	Set Ai to (Bj) + (Bk)	56ijk	
SAi	Bj-Bk	Set Ai to (Bj) - (Bk)	57ijk	
SBi	Aj±K	Set Bi to (Aj) ± K	60ijK	
SBi	Bj±K	Set Bi to (Bj) ± K	61ijK	
SBi	Xj±K	Set Bi to (Xj) ± K	62ijK	
SBi	Xj+Bk	Set Bi to (Xj) + (Bk)	63ijk	
SBi	Aj+Bk	Set Bi to (Aj) + (Bk)	64ijk	
SBi	Aj-Bk	Set Bi to (Aj) - (Bk)	65ijk	

TABLE 4. CENTRAL PROCESSOR INSTRUCTIONS (ALPHABETICAL LISTING) (Contd)

Mnemonic	Variable	Description	Operation Code	Notes
SBi	Bj+Bk	Set Bi to (Bj) + (Bk)	66ijk	
SBi	Bj-Bk	Set Bi to (Bj) - (Bk)	67ijk	
SXi	Aj±K	Set Xi to (Aj) ± K	70ijK	
SXi	Bj±K	Set Xi to (Bj) ± K	71ijK	
SXi	Xj±K	Set Xi to (Xk) ± K	72ijK	
SXi	Xj+Bk	Set Xi to (Xj) + (Bk)	73ijk	
SXi	Aj+Bk	Set Xi to (Aj) + (Bk)	74ijk	
SXi	Aj-Bk	Set Xi to (Aj) - (Bk)	75ijk	
SXi	Bj+Bk	Set Xi to (Bj) + (Bk)	76ijk	
SXi	Bj-Bk	Set Xi to (Bj) - (Bk)	77ijk	
TBj		Set Bj to current clock time	016j0	4
UXi	Bj,Xk	Unpack (Xk) to Xi and Bj	26ijk	
UXi		Unpack (Xi) to Xi	26i0i	
UXi	Bj	Unpack (Xi) to Xi and Bj	26iji	

WE	Bj±K	Write extended core storage	012jK	3
WL	Bj±K	Block copy K + (Bj) words from SCM to LCM	012jK	4
WXj	Xk	Write (Xj) into LCM at (Xk)	015jk	4
WXj	Xk	Write (Xj) to UEM at (Xk) + RA_e	015jk	1
XJ	Bj±K	Central exchange jump to Bi±K	013jK	3,6
ZR	K	Branch to K	0400K	
ZR	Xj,K	Branch to K if (Xj) = 0	030jK	5
ZR	Bi,K	Branch to K if (Bi) = 0	04i0K	
ZXi	Bj,Xk	Round and normalize (Xk) in Xi and Bj	25ijk	
ZXi		Round and normalize (Xi) to Xi	25i0i	
ZXi	Bj	Round and normalize (Xi) to Xi; shift count to Bj	25iji	

1 CYBER 170 models 825, 835, 855, 865, and 875.

2 CYBER 70 model 76; and 7600.

3 CYBER 170 Computer Systems; CYBER 70 models 71, 72, 73, and 74; 6000 Computer Systems.

4 CYBER 170 model 176; CYBER 70 model 76; and 7600.

5 Go to K±Bi; for CYBER 70 model 74 and 6600/6700, tests are made in increment unit.
Go to K if Xj; for CYBER 70 model 74 and 6600/6700, tests are made in add unit.

6 Privileged to Monitor.

TABLE 5. CENTRAL PROCESSOR INSTRUCTIONS THAT FORCE UPPER

Instruction	Force Upper Occurs Before or After Assembly of the Instruction
Unconditional EQ	After
ES	After
JP	After
MJ	After
PS	Before and After
RE	Before
RJ	After
RL †	Before
WE	Before
WL †	Before
XJ	Before and After
Unconditional ZR	After

†No force upper on SCOPE 2

COMPARE/MOVE INSTRUCTIONS

The Compare/Move Unit (CMU) is a standard central processor hardware component of the CYBER 170 models 172, 173, 174, 720 and 730, and the CYBER 70 models 72 and 73. It provides central processor hardware for moving and comparing data fields that consist of strings of 6-bit characters. The CYBER 170 models 825, 835, 855, 865, and 875 support compare/move instructions through simulation.

Data fields (strings) can:

Span word boundaries

Begin or end with any character position within a word

Data fields cannot be in either the operating registers or in ECS/LCM/UEM.

These symbols are used in the compare/move instructions:

sl String length in characters

ks First word address of the source field

kd First word address of the destination field

ka First word address of the first field

kb First word address of the second field

cs Starting character position (0 through 9) in the source data field

cd Starting character position (0 through 9) in the destination data field.

ca Starting character position (0 through 9) in the first data field.

Starting character position (0 through 9) in the second data field.

The compare/move instructions are shown in table 6. Formats for the CC, CU, DM, and MD instructions are shown in figure 4.

Each of the compare/move instructions causes a force upper before assembly of the instruction. IM also causes a force upper after assembly of the instruction.

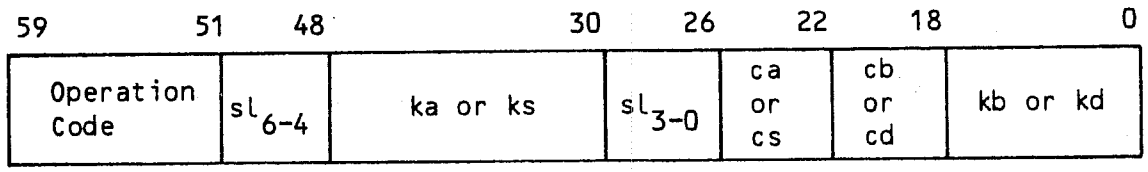
TABLE 6. COMPARE/MOVE INSTRUCTIONS

Operation	Variable	Description	Instruction Size	Operation Code
CC	sl,ka,ca,kb,cb	Compare collated	60 bits	466v†
CU	sl,ka,ca,kb,cb	Compare uncollated	60 bits	467v†
DM	sl,ks,cs,kd,cd	Direct move	60 bits	465v†
IM	K	Move data according to word at K	30 bits	4640K
IM	Bj±K	Move data according to word at Bj±K	30 bits	464jK
IM	Bj	Move data according to word at Bj	30 bits	464j000000
MD	sl,ks,cs,kd,cd	Indirect move descriptor word	60 bits	††

†v is 51 bits describing the variable field of the instruction.

††The MD instruction is a pseudo instruction that generates a descriptor word for the IM instruction. The MD instruction has no operation code; it should not be used as executable code.

CC, CU, and DM instructions:



MD pseudo instruction:

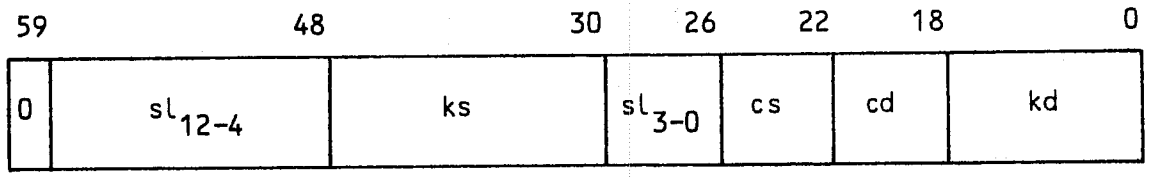


Figure 4. CC, CU, DM, and MD Formats

PERIPHERAL PROCESSOR INSTRUCTIONS

Peripheral processor instructions have two formats (see figure 5):

The 12-bit format has a 6-bit operation code f and a 6-bit operand address d.

The 24-bit format has a 6-bit operation code f (7 bits for the CCF, CFM, SCF, and SFM instructions), a 6-bit d field (5 bits for the CCF, CFM, SCF, and SFM instructions), and a 12-bit m field. In many 24-bit instructions, the d and m fields are combined to form the operand or operand address.

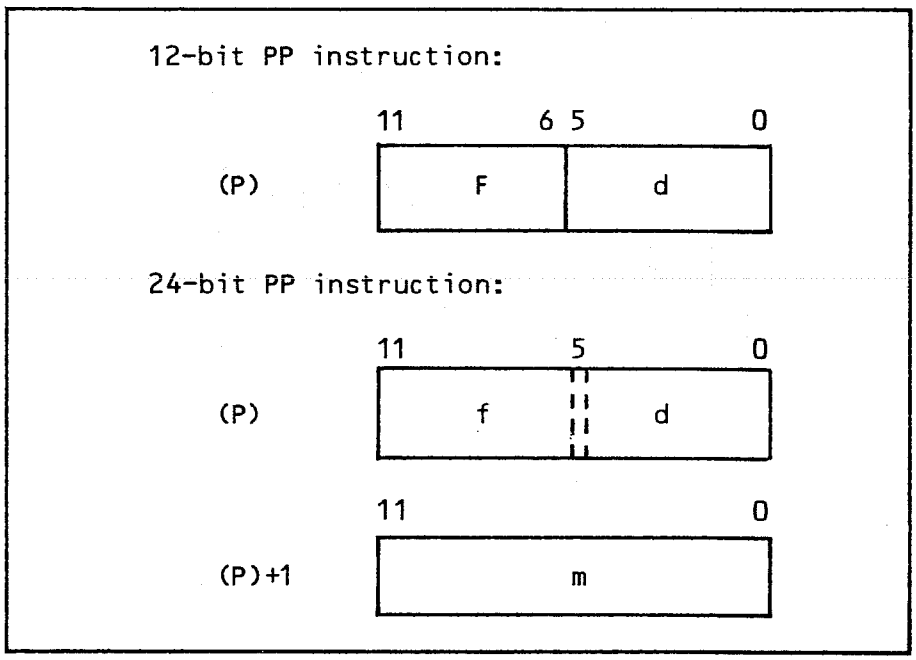


Figure 5. Peripheral Processor Instruction Formats

Peripheral processor instruction listings are given in tables 7 and 8. Table 7 lists the instructions numerically by operation code; table 8 lists them alphabetically by mnemonic. The following symbols are used in the instruction listings:

d

    d itself

(d)

    Contents of d

((d))

    Contents of the location specified by d

m

    m itself used as an address

m + (d)

    Contents of d are added to m to form an operand (jump address)

(m + (d))

    Contents of d are added to m to form the address of the operand

dm

    18-bit quantity with d as the upper 6 bits and m as the lower 12 bits

Some instructions in existing COMPASS programs are not valid for execution on the CYBER 170 models 825, 835, 855, 865, and 875. You can specify S=AIDTEXT in the COMPASS control statement to detect these instructions; COMPASS flags the invalid instructions with a type 0 error. S=AIDTEXT should not be specified if the 8 option is chosen for the MACHINE pseudo instruction.

TABLE 7. PERIPHERAL PROCESSOR INSTRUCTIONS (NUMERICAL LISTING)

Operation Code	Mnemonic	Variable	Description	Notes
01dm	LJM	m,d	Long jump to m + (d)	
02dm	RJM	m,d	Return jump to m + (d)	
03d	UJN	r	Unconditional jump p $\pm$ r	
04d	ZJN	r	Zero jump p $\pm$ r	
05d	NJN	r	Nonzero jump p $\pm$ r	
06d	PJN	r	Plus jump p $\pm$ r	
07d	MJN	r	Minus jump p $\pm$ r	
10d	SHN	r	Shift (A) + (left) or - (right) r bits	
11d	LMN	d	Logical difference (A) - d $\longrightarrow$ A	
12d	LPN	d	Logical product (A) * d $\longrightarrow$ A	
13d	SCN	d	Selective clear (A)	
14d	LDN	d	Load d $\longrightarrow$ A	
15d	LCN	d	Load complement d $\longrightarrow$ A	

16d	ADN	d	Add d + (A) → A	
17d	SBN	d	Subtract (A) - d → A	
20dm	LDC	c	Load dm → A	
21dm	ADC	c	Add (A) + dm → A	
22dm	LPC	c	Logical product (A) * dm → A	
23dm	LMC	c	Logical difference (A) - dm → A	
2400	PSN		Pass	
24d	LRD	d	Load (R) from d and d + 1	1
25d	SRD	d	Store (R) into d and d + 1	1
260d	EXN	d	Exchange jump to central processor d to (A)	3
261d	MXN	d	Monitor exchange jump central processor d to (A)	3
262d	MAN	d	Monitor exchange jump central processor d to (MA)	5
270d	RPN	d	Read program address of central processor d → A	4
30d	LDD	d	Load (d) → A	
31d	ADD	d	Add (A) + (d) → A	
32d	SBD	d	Subtract (A) - (d) → A	
33d	LMD	d	Logical difference (A) and (d) → A	
34d	STD	d	Store (A) → d	

TABLE 7. PERIPHERAL PROCESSOR INSTRUCTIONS (NUMERICAL LISTING) (Contd)

Operation Code	Mnemonic	Variable	Description	Notes
35d	RAD	d	Replace add (d) + (A) → d and A	
36d	AOD	d	Replace add one (d) + 1 → d and A	
37d	SOD	d	Replace subtract one (d) - 1 → d and A	
40d	LDI	d	Load ((d)) → A	
41d	ADI	d	Add ((d)) + (A) → A	
42d	SBI	d	Subtract (A) - ((d)) → A	
43d	LMI	d	Logical difference (A) - ((d)) → A	
44d	STI	d	Store (A) → (d)	
45d	RAI	d	Replace add (A) + ((d)) → (d) and A	
46d	AOI	d	Replace add one ((d)) + 1 → (d) and A	
47d	SOI	d	Replace subtract one ((d)) - 1 → (d) and A	
50dm	LDM	m,d	Load (M + (d)) → A	
51dm	ADM	m,d	Add (m + (d)) + (A) → A	

52dm	SBM	m,d	Subtract (A) - (m + (d)) ⟶ A	
53dm	LMM	m,d	Logical difference (A) - (m + (d)) ⟶ A	
54dm	STM	m,d	Store (A) ⟶ m + (d)	
55dm	RAM	m,d	Replace add (m + (d)) + (A) ⟶ m + (d) and A	
56dm	AOM	m,d	Replace add one (M + (d)) + 1 ⟶ m + (d) and A	
57dm	SOM	m,d	Replace subtract one (m + (d)) - 1 ⟶ m + (d) and A	
60d	CRD	d	Central read from (A) to d	3
60dm	FIM	m,d	Jump to m; input word flag on channel d	2, 6
61d	CRM	m,d	Central read (d) words from (A) to m	3, 6
61dm	EIM	m,d	Jump to m; no input word flag on channel d	2, 6
62d	CWD	d	Central write to (A) from d	3
62dm	IRM	m,d	Jump to m; input record flag on channel d	2, 6
63dm	CWM	m,d	Central write (d) words to (A) from m	3, 6
63dm	NIM	m,d	Jump to m; no input record flag on channel d	2, 6
64dm	AJM	m,d	Jump to m if channel d active	3, 6
64dm	FOM	m,d	Jump to m; output word flag on channel d	2, 6
644dm	SCF	m,d	Jump to m if channel d flag set	1, 7
65dm	IJM	m,d	Jump to m if channel d inactive	3, 6

TABLE 7. PERIPHERAL PROCESSOR INSTRUCTIONS (NUMERICAL LISTING) (Contd)

Operation Code	Mnemonic	Variable	Description	Notes
65dm	EOM	m,d	Jump to m; no output word flag on channel d	2, 6
654dm	CCF	m,d	Clear channel d flag	1, 7
66dm	FJM	m,d	Jump to m if channel d full	3, 6
66dm	ORM	m,d	Jump to m; output record flag on channel d	2, 6
664dm	SFM	m,d	Jump to m if channel d error flag set	1, 7
67dm	EJM	m,d	Jump to m if channel d empty	3, 6
67dm	NOM	m,d	Jump to m; no output record on channel d	2, 6
674dm	CFM	m,d	Jump to m if channel d error flag clear	1, 7
70d	IAN	d	Input to A from channel d	
71dm	IAM	m,d	Input (A) words to m from channel d	
72d	OAN	d	Output from A on channel d	
73dm	OAM	m,d	Output (A) words to channel d from m	
74d	ACN	d	Activate channel d	3

74d	RFN	d	Send record flag on channel d	2
75d	DCN	d	Disconnect channel d	3
76d	FAN	d	Function (A) on channel d	3
77dm	FNC	m,d	Function m on channel d	3, 6
7700	ESN	d	Error stop	2

1 CYBER 170 models 825, 835, and 855 only

2 CYBER 70 model 76 and 7600 only

3 CYBER 170 Computer Systems; CYBER 70 models 71, 72, 73, 74; and 6000 Computer Systems only

4 CYBER 170 Computer Systems except models 825, 835, 855, 865, and 875; CYBER 70 models 71, 72, 73, 74; and 6000 Computer Systems only

5 CYBER 170 Computer Systems and CYBER 70 models 72, 73, and 74 only

6 d is required

7 7-bit operation code, 5-bit d field

TABLE 8. PERIPHERAL PROCESSOR INSTRUCTIONS (ALPHABETICAL LISTING)

Mnemonic	Variable	Description	Operation Code	Notes
ACN	d	Activate channel d	74d	3
ADC	c	Add (A) + dm → A	21dm	
ADD	d	Add (A) + (d) → A	31d	
ADI	d	Add ((d)) + (A) → A	41d	
ADM	m,d	Add (m + (d)) + (A) → A	51dm	
ADN	d	Add d + (A) → A	16d	
AJM	m,d	Jump to m if channel d active	64dm	3, 6
AOD	d	Replace add one (d) + 1 → d and A	36d	
AOI	d	Replace add one ((d)) + 1 → (d) and A	46d	
AOM	m,d	Replace add one (m + (d)) + 1 → m + (d) and A	56dm	
CCF	m,d	Clear channel d flag	654dm	1, 7
CFM	m,d	Jump to m if channel d flag clear	674dm	1, 7
CRD	d	Central read from (A) to d	60d	3, 6

CRM	m,d	Central read (d) words from (A) to m	61d	3, 6
CWD	d	Central write to (A) from d	62d	3
CWM	m,d	Central write (d) words to A from m	63dm	3, 6
DCN	d	Disconnect channel d	75d	3
EIM	m,d	Jump to m; no input word flag on channel d	61dm	2, 6
EJM	m,d	Jump to m if channel d empty	67dm	
EOM	m,d	Jump to m; no output word flag on channel d	65dm	2, 6
ESN	d	Error stop	7700	2
EXN	d	Exchange jump central processor d to (A)	260d	3
FAN	d	Function (A) on channel d	76d	3
FIM	m,d	Jump to m; input word flag on channel d	60dm	2, 6
FJM	m,d	Jump to m if channel d full	66dm	3, 6
FNC	m,d	Function m on channel d	77dm	3
FOM	m,d	Jump to m; output word flag on channel d	64dm	2, 6
IAM	m,d	Input (A) words to m from channel d	71dm	
IAN	d	Input to A from channel d	70d	
IJM	m,d	Jump to m if channel d inactive	65dm	3, 6
IRM	m,d	Jump to m; input record flag on channel d	62dm	2, 6

TABLE 8. PERIPHERAL PROCESSOR INSTRUCTIONS (ALPHABETICAL LISTING) (Contd)

Mnemonic	Variable	Description	Operation Code	Notes
LCN	d	Load complement d → A	15d	
LDC	c	Load dm → A	20dm	
LDD	d	Load (d) → A	30d	
LDI	d	Load ((d)) → A	40d	
LDM	m,d	Load (m + (d)) → A	50dm	
LDN	d	Load d → A	14d	
LJM	m,d	Long jump to m + (d)	01dm	
LMC	c	Logical difference (A) - dm → A	23dm	
LMD	d	Logical difference (A) and (d) → A	33d	
LMI	d	Logical difference (a) - ((d)) → A	43d	
LMM	m,d	Logical difference (A) - (m + (d)) → A	53dm	
LMN	d	Logical difference (A) - d → A	11d	
LPC	c	Logical product (A) * dm → A	22dm	

LPN	d	Logical product (A) * d → A	12d	
LRD	d	Load (R) from d and d + 1	24d	1
MAN	d	Monitor exchange jump central processor d to (MA)	262d	5
MJN	r	Minus jump p ± r	07d	
MXN	d	Monitor exchange jump central processor d to (A)	261d	2
NIM	m,d	Jump to m; no input record flag on channel d	63dm	2, 6
NJN	r	Nonzero jump p ± r	05d	
NOM	m,d	Jump to m; no output record flag on channel d	67dm	2, 6
OAM	m,d	Output (A) words to channel d from m	73dm	
OAN	d	Output from A on channel d	72d	
ORM	m,d	Jump to m; output record flag on channel d	66dm	2, 6
PJN	r	Plus jump to P ± r	06d	
PSN		Pass	2400	
RAD	d	Replace add (d) + (A) → d and A	35d	
RAI	d	Replace add (A) + ((d)) → (d) and A	45d	
RAM	m,d	Replace add (m + (d)) + (A) → m + (d) and A	55dm	
RFN	d	Send record flag on channel d	74d	2

TABLE 8. PERIPHERAL PROCESSOR INSTRUCTIONS (ALPHABETICAL LISTING) (Contd)

Mnemonic	Variable	Description	Operation Code	Notes
RJM	m,d	Return jump to m + (d)	02dm	
RPN	d	Read program address of central processor d to A	270d	4
SBD	d	Subtract (A) - (d) → A	32d	
SBI	d	Subtract (A) - ((d)) → A	42d	
SBM	m,d	Subtract A - (m + (d)) → A	52dm	
SBN	d	Subtract (A) - d → A	17d	
SCF	m,d	Jump to m if channel d flag set	644dm	1, 7
SCN	d	Selective clear (A)	13d	
SFM	m,d	Jump to m if channel d error flag set	664dm	1, 7
SHN	r	Shift (A) + (left) or - (right) r bits	10d	
SOD	d	Replace subtract one (d) - 1 → d and A	37d	
SOI	d	Replace subtract one ((d)) - 1 → (d) and A	47d	
SOM	m,d	Replace subtract one (m + (d)) - 1 → m + (d) and A	57dm	

SRD	d	Store (R) into d and d + 1	25d	1
STD	d	Store (A) ⟶ d	34d	
STI	d	Store (A) ⟶ (d)	44d	
STM	m,d	Store (A) ⟶ m + (d)	54dm	
UJN	r	Unconditional jump p ± r	03d	
ZJN	r	Zero jump p ± r	04d	

1 CYBER 170 models 825, 835, 855, 865, and 875 only.

2 CYBER 70 models 76 and 7600 only

3 CYBER 170 Computer Systems; CYBER 70 models 71, 72, 73, 74; and 6000 Computer Systems only

4 CYBER 170 Computer Systems except models 825, 835, 855, 865, and 875; CYBER 70 models 71, 72, 73, 74; and 6000 Computer Systems only

5 CYBER 170 Computer Systems, CYBER 70 models 72, 73, and 74 only

6 d is required

7 7-bit operation code; 5-bit d field

COMMON COMMON DECKS

The common common decks are a set of COMPASS subroutines that:

Convert data formats

Manage tables dynamically

Save and restore registers

Provide an input/output interface at the Combined Input/Output (CIO) and File Environment Table (FET) level

All of the common common decks run on NOS and NOS/BE; some run on SCOPE 2.

Most of the common common decks are available as relocatable subroutines that reside on the system library SYSLIB. To use these decks, relocatable central processor programs need only include external references (=Xsymbol) to the entry points in the common common decks. The external references are satisfied from SYSLIB at load time. The CYBER Loader searches SYSLIB by default when satisfying external references; the SCOPE 2 Loader does not. Under SCOPE 2, you must specify SYSLIB as part of the library set (for example, through the LDSET pseudo instruction).

The source code for all of the common common decks also resides on COMCPL; this source code references the system texts IPTEXT and CPUTEXT. COMCPL is an Update old program library.

Table 9 shows, for each common common deck, the deck name, relocatable program name, entry point names, and availability under SCOPE 2.

TABLE 9. COMMON COMMON DECKS

Common Common Deck Name	Description	Relocatable Program Name	Entry Points
COMCARG	Process control statement arguments	CPU.ARG	ARG=
COMCCDD	Convert constant to decimal display code	CPU.CDD	CDD=
COMCCFD	Convert constant to F10.3 format	CPU.CFD	CFD=
COMCCIO†	Process I/O operation	CPU.CIO	CIO=
COMCCOD	Convert constant to octal display code	CPU.COD	COD=
COMCCPT	Extract comments field from prefix table	CPU.CPT	CPT=
COMCDXB	Convert display code to binary	CPU.DXB	DXB=
COMCMNS	Move non-overlapping bit string	CPU.MNS	MNS=
COMCMOS	Move overlapping bit string	CPU.MOS	MOS=
COMCMTM	Managed table macros	††	

TABLE 9. COMMON COMMON DECKS (Contd)

Common Common Deck Name	Description	Relocatable Program Name	Entry Points
COMCMTP	Managed table processors	††	
COMCMVE	Move block of data	CPU.MVE	MVE=
COMCRDC†	Read coded line, C format	CPU.RDC	RDC=
COMCRDH†	Read coded line, H format	CPU.RDH	RDH=
COMCRDO†	Read one word	CPU.RDO	RDO=
COMCRDS†	Read coded line to string buffer	CPU.RDS	RDS=
COMCRDW†	Read words to working buffer	CPU.RDW	RDW= RDX= LCB=
COMCRSR	Restore all registers	CPU.RSR	RSR=
COMCSFN	Space fill name on right	CPU.SFN	SFN=

COMCSRT	Set record type	CPU.SRT	SRT=
COMCSST	Shell sort table	CPU.SST	SST=
COMCSTF†	Set terminal file	CPU.STF	STF=
COMCSVR	Save all registers	CPU.SVR	SVR=
COMCSYS†	Process system request	CPU.SYS	SYS= RCL= WNB= MSG=
COMCUPC	Unpack control statement arguments	CPU.UPC	UPC=
COMCWOD	Convert word to octal display code	CPU.WOD	WOD=
COMCWTC†	Write coded line, C format	CPU.WTC	WTC=
COMCWTH†	Write coded line, H format	CPU.WTH	WTH=
COMCWTO†	Write one word	CPU.WTO	WTO=
COMCWTS†	Write coded line from string buffer	CPU.WTS	WTS=

TABLE 9. COMMON COMMON DECKS (Contd)

Common Common Deck Name	Description	Relocatable Program Name	Entry Points
COMCWTW†	Write words from working buffer	CPU.WTW	WTW= WTX= DCB=
COMCXJR	Restore all registers with a system XJR call	CPU.XJR	XJR=
COMCZTB	Convert all 00 characters to blanks	CPU.ZTB	ZTB=

†Not available on SCOPE 2

††Not a relocatable subroutine on SYSLIB

DIAGNOSTICS

The COMPASS assembler produces two types of diagnostic messages:

Fatal error messages (table 10) are flagged with an alphabetic character.

Informative error messages (table 11) are flagged with a digit.

TABLE 10. FATAL ERRORS

Error Type	Description
A	ADDRESS FIELD BAD. An error exists in a variable subfield entry. For example: CODE character not A, D, E, I, O, or * Symbol or name greater than 8 characters Expression does not reduce to one external term; relocatable terms do not cancel properly; instruction disallows register designators; instruction requires absolute expression; and so forth. Data error: 8 or 9 encountered in octal data; modifier not S, P, O, E, D, or B No data in variable field of LIT instruction Relative jump out of range (-31>r>31) on peripheral processor instruction BASE character not O, M, D, or * Register illegal in CON instruction Unable to locate synonymous instruction for OPSYN or CPSYN Micro count less than zero or greater than 10 NOLABEL character not I

TABLE 10. FATAL ERRORS (Contd)

Error Type	Description
	Negative relocation on ORG or ORGC POS value less than 0 or greater than word size Erroneous OPDEF reference No comma following DIS word count No symbol after =S, =X, or =Y Illegal entry in the variable field of IDENT
D	DOUBLY DEFINED SYMBOL. THE FIRST DEFINITION HOLDS. Symbol previously defined or declared external.
E	ECHO, DUP, RMT, OR MACRO ILLEGALLY NESTED. Definition not wholly within next outer definition.
F	NUMBER OF ENTRIES EXCEEDS PERMISSIBLE AMOUNT. LIT generates more than 100 words Data missing or erroneous on XTEXT file More than 63 formal parameters and local names in macro definition More than 255 blocks More than 511 external symbols
L	LOCATION FIELD BAD. Required location field entry is erroneous. Format two macro definition has no substitutable parameters.

TABLE 10. FATAL ERRORS (Contd)

Error Type	Description
N	NEGATIVE RELOCATION ON ENTRY POINT.
O	OPERATION FIELD BAD. Instructions unrecognizable, out of sequence (for example, ABS or PPU pseudo instruction not in first statement group or instruction is illegal for binary mode), or relational mnemonic on IF statement is erroneous. Location symbol begins beyond column 2. As a result of the S=AIDTEXT compiler option or the MACHINE 8 pseudo instruction, COMPASS has determined that the instruction is not valid for the CYBER 170 models 825, 835, 855, 865, and 875.
P	CONSULT LISTING FOR REASON BEHIND P-ERROR. Error flag generated by ERR or ERRxx pseudo instruction.
R	DATA ORIGIN OUTSIDE BLOCK OR IN BLANK COMMON. Attempt made to set data into blank common or beyond block limits.
U	UNDEFINED SYMBOL. VALUE ASSUMED 0. Reference to a symbol that is not defined; for example, IF statement line count, DIS word count, or unrecognizable attribute on IF statement.
V	BIT COUNT ERROR ON VFD (MUST BE 0 $\leq$ COUNT $\leq$ 60). VFD field size erroneous.

TABLE 11. INFORMATIVE ERRORS

Error Type	Description
1	LOCATION SYMBOL BAD, SYMBOL NOT DEFINED. Location field entry erroneous. The instruction does not require an entry.
2	ADDRESS ERROR ON SYMBOL DEFINITION. Erroneous variable field entry. The location field symbol is not defined.
3	DUPLICATE MACRO DEFINITION. NEW ONE OVERRIDES. Macro, OPDEF, or synonymous operation redefines operation code.
4	BAD FORMAL PARAMETER NAME IGNORED. Macro or ECHO formal parameter name repeated or illegal.
5	CPU OPERATION SYNTAX INCORRECTLY SPECIFIED. OPDEF, CPOP, or CPSYN specifies illegal syntax.
6	LOCATION FIELD MEANINGLESS. Entry in location field is ignored.
7	ADDRESS VALUE EXCEEDS FIELD SIZE, RESULT TRUNCATED. Value of expression exceeds size of destination field. BSS address expression value is negative. MICRO starting character position or character count negative.
8	MISSING OR EXTRA ADDRESS SUBFIELD. Variable subfield entry missing or superfluous.
9	MICRO SUBSTITUTION ERROR. NO SUBSTITUTION. Micro reference unrecognized.

CHARACTER SETS

Table 12 shows the standard character sets used in CDC operating systems.

TABLE 12. STANDARD CHARACTER SETS

CDC Graphic	ASCII Graphic Subset	Display Code	Hollerith Punch (026)	External BCD Code	ASCII Punch (029)	ASCII Code
:†	:	00†	8-2	00	8-2	3A
A	A	01	12-1	61	12-1	41
B	B	02	12-2	62	12-2	42
C	C	03	12-3	63	12-3	43
D	D	04	12-4	64	12-4	44
E	E	05	12-5	65	12-5	45
F	F	06	12-6	66	12-6	46
G	G	07	12-7	67	12-7	47
H	H	10	12-8	70	12-8	48
I	I	11	12-9	71	12-9	49

J	J	12	11-1	41	11-1	4A
K	K	13	11-2	42	11-2	4B
L	L	14	11-3	43	11-3	4C
M	M	15	11-4	44	11-4	4D
N	N	16	11-5	45	11-5	4E
O	O	17	11-6	46	11-6	4F
P	P	20	11-7	47	11-7	50
Q	Q	21	11-8	50	11-8	51
R	R	22	11-9	51	11-9	52
S	S	23	0-2	22	0-2	53
T	T	24	0-3	23	0-3	54
U	U	25	0-4	24	0-4	55
V	V	26	0-5	25	0-5	56
W	W	27	0-6	26	0-6	57

TABLE 12. STANDARD CHARACTER SETS (Contd)

CDC Graphic	ASCII Graphic Subset	Display Code	Hollerith Punch (026)	External BCD Code	ASCII Punch (029)	ASCII Code
X	X	30	0-7	27	0-7	58
Y	Y	31	0-8	30	0-8	59
Z	Z	32	0-9	31	0-9	5A
0	0	33	0	12	0	30
1	1	34	1	01	1	31
2	2	35	2	02	2	32
3	3	36	3	03	3	33
4	4	37	4	04	4	34
5	5	40	5	05	5	35
6	6	41	6	06	6	36

7	7	42	7	07	7	37	
8	8	43	8	10	8	38	
9	9	44	9	11	9	39	
+	+	45	12	60	12-8-6	2B	
-	-	46	11	40	11	2D	
*	*	47	11-8-4	54	11-8-4	2A	
/	/	50	0-1	21	0-1	2F	
(	(	51	0-8-4	34	12-8-5	28	
)	)	52	12-8-4	74	11-8-5	29	
$	$	53	11-8-3	53	11-8-3	24	
=	=	54	8-3	13	8-6	3D	
blank	blank	55	no punch	20	no punch	20	
(comma)	(comma)	56	0-8-3	33	0-8-3	2C	
(period)	(period)	57	12-8-3	73	12-8-3	2E	

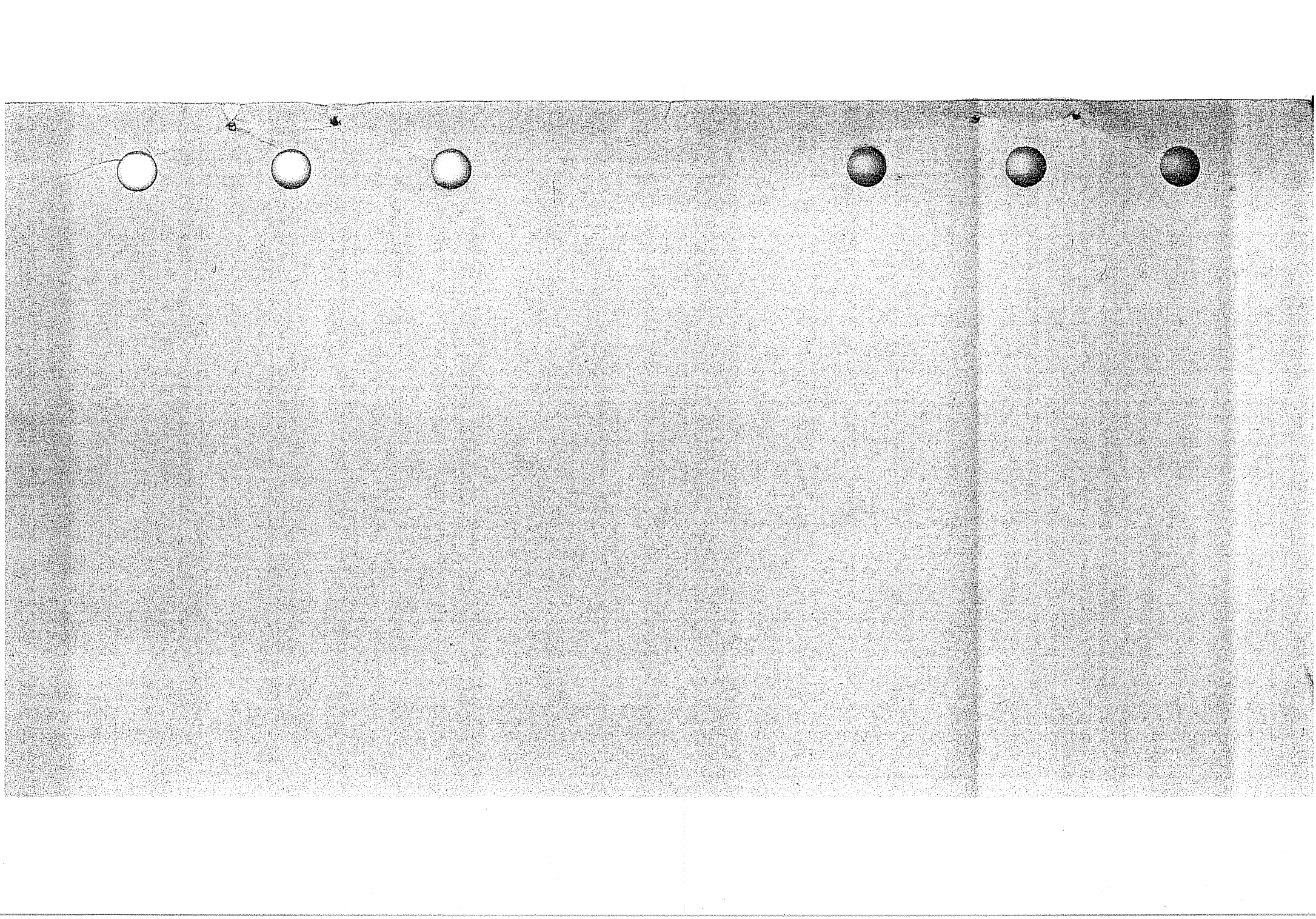
TABLE 12. STANDARD CHARACTER SETS (Contd)

CDC Graphic	ASCII Graphic Subset	Display Code	Hollerith Punch (026)	External BCD Code	ASCII Punch (029)	ASCII Code
≡	#	60	0-8-6	36	8-3	23
[	[	61	8-7	17	12-8-2	5B
]	]	62	0-8-2	32	11-8-2	5D
%††	%	63††	8-6	16	0-8-4	25
≠	"(quote)	64	8-4	14	8-7	22
→	_(underline)	65	0-8-5	35	0-8-5	5F
∨	!	66	11-0	52	12-8-7	21
∧	&	67	0-8-7	37	12	26
↑	′(apostrophe)	70	11-8-5	55	8-5	27
↓	?	71	11-8-6	56	0-8-7	3F

<	<	72	12-0	72	12-8-4	3C
>	>	73	11-8-7	57	0-8-6	3E
≤	@	74	8-5	15	8-4	40
≥	\	75	12-8-5	75	0-8-2	5C
¬	^(circumflex)	76	12-8-6	76	11-8-7	5E
;(semicolon)	;(semicolon)	77	12-8-7	77	11-8-6	3B

†Twelve or more zero bits at the end of a 60-bit word are interpreted as end-of-line mark rather than two colons. End-of-line mark is converted to external BCD 1632.

††In installations using the CDC 63-graphic set, display code 00 has no associated graphic or Hollerith code; display code 63 is the colon (8-2 punch).



CONTROL DATA CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO.
MINNEAPOLIS, MINN. 55440

SALES OFFICES AND SERVICE CENTERS
IN MAJOR CITIES THROUGHOUT THE WORLD