CPU INSTRUCTIONS

(1) CYBER 70 Model 76, and 7600
(2) CYBER 170 Series, CYBER 70 Models 71, 72, 73, and 74, and 6000 Series
(3) Privileged to Monitor
(4) CYBER 170 Model 176, CYBER 70 Model 76, and 7600

Instruction	Operation		Variable	Description
00000	ES	(1) (3)		Error exit to EEA
0000K	PS	(2)	K	Program stop
0100K	RJ		K	Return jump to K
011jK	RL	(4)	Bj±K	Block-copy K plus (Bj) words from LCM to SCM
011jK	RE	(2)	Bj±K	Read extended core storage
012jK	WL	(4)	Bj±K	Block-copy K plus (Bj) words from SCM to LCM
012jK	WE	(2)	Bj±K	Write extended core storage
01300	MJ	(1)		Exchange-exit to NEA if exit flag clear
013jK	MJ	(1) (3)	Bj±K	Exchange-exit to K + (Bj) if exit flag set
013jK	XJ	(2) (3)	Bj±K	Central exchange jump to Bj+K
014jk	RXj	(4)	Xk	Read LCM at (Xk) to Xj
015jk	WXj	(4)	Xk	Write (Xj) into LCM at (Xk)
0160k	RI	(1) (3)	Bk	Reset channel (Bk) input buffer
016jk	IBj	(1)	Bk	Read channel (Bk) input status to Bj if j ≠ 0; otherwise, same as RI
016j0	TBj	(4)		Set Bj to current clock time
0170k	RO	(3) (4)	Bk	Reset channel (Bk) output buffer
017jk	OBj	(3) (4)	Bk	Read channel (Bk) output status to Bj if j ≠ 0; otherwise, same as RO
02iiK	JP		Bi+K	Jump to K plus (Bi)
030jK	ZR		Xj,K	Branch to K if (Xj) = 0
031jK	NZ		Xj,K	Branch to K if (Xj) ≠ 0
032jK	PL		Xj,K	Branch to K if (Xj) sign is plus
033jK	MI		Xj,K	Branch to K if (Xj) sign is minus
033jK	NG		Xj,K	Branch to K if (Xj) sign is minus
034jK	IR		Xj,K	Branch to K if (Xj) in range
035jK	OR		Xj,K	Branch to K if (Xj) not in range
036jK	DF		Xj,K	Branch to K if (Xj) definite
037jK	ID		Xj,K	Branch to K if (Xj) indefinite
0400K	EQ		K	Branch to K
0400K	ZR		K	Branch to K
04ijK	EQ		Bi, Bj, K	Branch to K if (Bi) = (Bj)
04i0K	ZR		Bi, K	Branch to K if (Bi) = 0
05ijK	NE		Bi, Bj, K	Branch to K if (Bi) ≠ (Bj)
05i0K	NZ		Bi, K	Branch to K if (Bi) ≠ 0
06ijK	GE		Bi, Bj, K	Branch to K if (Bi) ≥ (Bj)
06i0K	GE		Bi, K	Branch to K if (Bi) ≥ 0
06ijK	LE		Bj, Bi, K	Branch to K if (Bj) ≤ (Bi)
060jK	LE		Bj, K	Branch to K if (Bj) ≤ 0
06i0K	PL		Bi, K	Branch to K if (Bi) ≥ 0
070iK	LT		Bi, K	Branch to K if (Bi) < 0
07ijK	LT		Bi, Bj, K	Branch to K if (Bi) < (Bj)
07i0K	NG		Bi, K	Branch to K if (Bi) < 0
07ijK	GT		Bj, Bi, K	Branch to K if (Bj) > (Bi)
070jK	GT		Bj, K	Branch to K if (Bj) > 0
07i0K	MI		Bi, K	Branch to K if (Bi) < 0
10ijj	BXi		Xj	Copy (Xj) to Xi
11ijk	BXi		Xj*Xk	Logical product of (Xj) and (Xk) to Xi
12ijk	BXi		Xj+Xk	Logical sum of (Xj) plus (Xk) to Xi
13ijk	BXi		Xj−Xk	Logical difference of (Xj) minus (Xk) to Xi
14ikk	BXi		−Xk	Copy complement of (Xk) to Xi
15ijk	BXi		−Xk*Xj	Logical product of (Xj) and complement of (Xk) to Xi
16ijk	BXi		−Xk+Xj	Logical sum of (Xj) plus complement of (Xk) to Xi
17ijk	BXi		−Xk−Xj	Logical difference of (Xj) minus complement of (Xk) to Xi
20ijk	LXi		jk	Logical-shift (Xi) by ± jk
21ijk	AXi		jk	Arithmetic-shift (Xi) by ± jk
22ijk	LXi		Bj, Xk	Logical-shift (Xk) by (Bj) to Xi
22iji	LXi		Bj	Logical-shift (Xi) by (Bj) to Xi
22i0K	LXi		Xk	Transmit (Xk) to Xi
22ijk	LXi		Xk, Bj	Logical-shift (Xk) by (Bj) to Xi
23ijk	AXi		Bj, Xk	Arithmetic-shift (Xk) by (Bj) to Xi
23iji	AXi		Bj	Arithmetic-shift (Xi) by (Bj) to Xi
23i0k	AXi		Xk	Transmit (Xk) to Xi
23ijk	AXi		Xk, Bj	Arithmetic-shift (Xk) by (Bj) to Xi
24ijk	NXi, Bj		Xk	Normalize (Xk) to Xi and Bj
24i0i	NXi			Normalize (Xi) to Xi
24iji	NXi, Bj			Normalize (Xi) to Xi and Bj
24i0k	NXi		Xk	Normalize (Xk) to Xi
24ijk	NXi		Bj, Xk	Normalize (Xk) to Xi and Bj
24ijk	NXi		Xk, Bj	Normalize (Xk) to Xi and Bj
25ijk	ZXi, Bj		Xk	Round and normalize (Xk) to Xi and Bj
25i0i	ZXi			Round and normalize (Xi) to Xi
25iji	ZXi, Bj			Round and normalize (Xi) to Xi and Bj
25i0k	ZXi		Xk	Round and normalize (Xk) to Xi
25ijk	ZXi		Bj, Xk	Round and normalize (Xk) to Xi and Bj
25ijk	ZXi		Xk, Bj	Round and normalize (Xk) to Xi and Bj
26ijk	UXi, Bj		Xk	Unpack (Xk) to Xi and Bj
26i0i	UXi			Unpack (Xi) to Xi
26iji	UXi, Bj			Unpack (Xi) to Xi and Bj
26i0k	UXi		Xk	Unpack (Xk) to Xi
26ijk	UXi		Bj, Xk	Unpack (Xk) to Xi and Bj
26ijk	UXi		Xk, Bj	Unpack (Xk) to Xi and Bj

Logical Operators

*	01	0	0
	11	1	1
	01	0	0
+	01	0	0
	11	1	1
	11	1	1
−	01	0	0
	11	1	1
	10	1	1

Instruction	Operation	Variable	Description
27ijk	PXi	Bj, Xk	Pack (Xk) and (Bj) to Xi
270i	PXi		Pack (Xi) to Xi
27iji	PXi	Bj	Pack (Xi) and (Bj) to Xi
270k	PXi	Xk	Pack (Xk) to Xi
27ijk	PXi	Xk, Bj	Pack (Xk) and (Bj) to Xi
30ijk	FXi	Xj+Xk	Sum of (Xj) plus (Xk) to Xi
31ijk	FXi	Xj−Xk	Difference of (Xj) minus (Xk) to Xi
32ijk	DXi	Xj+Xk	Double-precision sum of (Xj) plus (Xk) to Xi
33ijk	DXi	Xj−Xk	Double-precision difference of (Xj) minus (Xk) to Xi
34ijk	RXi	Xj+Xk	Rounded sum of (Xj) plus (Xk) to Xi
35ijk	RXi	Xj−Xk	Rounded difference of (Xj) minus (Xk) to Xi
36ijk	IXi	Xj+Xk	Integer sum of (Xj) plus (Xk) to Xi
37ijk	IXi	Xj−Xk	Integer difference of (Xj) minus (Xk) to Xi
40ijk	FXi	Xj*Xk	Product of (Xj) times (Xk) to Xi
41ijk	RXi	Xj*Xk	Rounded product of (Xj) times (Xk) to Xi
42ijk	IXi	Xj*Xk	Integer product of (Xj) times (Xk) to Xi
42ijk	DXi	Xj*Xk	Double-precision product of (Xj) times (Xk) to Xi
43ijk	MXi	±jk	Form mask of ± jk bits in Xi
44ijk	FXi	Xj/Xk	Divide (Xj) by (Xk) to Xi
45ijk	RXi	Xj/Xk	Rounded divide (Xj) by (Xk) to Xi
46000	NO		Pass (do-nothing)
47ikk	CXi	Xk	Population count of (Xk) to Xi
50ijK	SAi	Aj±K	(Aj) plus K to Ai
51ijK	SAi	Bj±K	(Bj) plus K to Ai
51i0K	SAi	K	K plus 0 to Ai
52ijK	SAi	Xj±K	(Xj) plus K to Ai
53ijk	SAi	Xj+Bk	(Xj) plus (Bk) to Ai
53ijk	SAi	Bk+Xj	(Bk) plus (Xj) to Ai
53i0k	SAi	Xj	(Xj) plus 0 to Ai
54ijk	SAi	Aj+Bk	(Aj) plus (Bk) to Ai
54ijk	SAi	Bk+Aj	(Bk) plus (Aj) to Ai
54ij0	SAi	Aj	(Aj) plus 0 to Ai
55ijk	SAi	Aj−Bk	(Aj) minus (Bk) to Ai
55ijk	SAi	−Bk+Aj	(Aj) minus (Bk) to Ai
56ijk	SAi	Bj+Bk	(Bj) plus (Bk) to Ai
56ij0	SAi	Bj	(Bj) plus 0 to Ai
57ijk	SAi	Bj−Bk	(Bj) minus (Bk) to Ai
57i0k	SAi	−Bk	0 minus (Bk) to Ai
57ijk	SAi	−Bk+Bj	(Bj) minus (Bk) to Ai
60ijK	SBi	Aj±K	(Aj) plus K to Bi
61ijK	SBi	Bj±K	(Bj) plus K to Bi
61i0K	SBi	K	K plus 0 to Bi
62ijK	SBi	Xj±K	(Xj) plus K to Bi
63ijk	SBi	Bk+Xj	(Bk) plus (Xj) to Bi
63ij0	SBi	Xj	(Xj) plus 0 to Bi
64ijk	SBi	Aj+Bk	(Aj) plus (Bk) to Bi
64ijk	SBi	Bk+Aj	(Bk) plus (Aj) to Bi
64ij0	SBi	Aj	(Aj) plus 0 to Bi
65ijk	SBi	Aj−Bk	(Aj) minus (Bk) to Bi
65ijk	SBi	−Bk+Aj	(Aj) minus (Bk) to Bi
66ijk	SBi	Bj+Bk	(Bj) plus (Bk) to Bi
66ij0	SBi	Bj	(Bj) plus 0 to Bi
67ijk	SBi	Bj−Bk	(Bj) minus (Bk) to Bi
67i0k	SBi	−Bk	0 minus (Bk) to Bi
67ijk	SBi	−Bk+Bj	(Bj) minus (Bk) to Bi
70ijK	SXi	Aj±K	(Aj) plus K to Xi
71ijK	SXi	Bj±K	(Bj) plus K to Xi
71i0K	SXi	K	K plus 0 to Xi
72ijK	SXi	Xj+K	(Xj) plus K to Xi
73ijk	SXi	Xj+Bk	(Xj) plus (Bk) to Xi
73ijk	SXi	Bk+Xj	(Bk) plus (Xj) to Xi
73ij0	SXi	Xj	(Xj) plus 0 to Xi
74ijk	SXi	Aj+Bk	(Aj) plus (Bk) to Xi
74ijk	SXi	Bk+Aj	(Bk) plus (Aj) to Xi
74ij0	SXi	Aj	(Aj) plus 0 to Xi
75ijk	SXi	Aj−Bk	(Aj) minus (Bk) to Xi
75ijk	SXi	−Bk+Aj	(Aj) minus (Bk) to Xi
76ijk	SXi	Bj+Bk	(Bj) plus (Bk) to Xi
76ij0	SXi	Bj	(Bj) plus 0 to Xi
77ijk	SXi	Bj−Bk	(Bj) minus (Bk) to Xi
77i0k	SXi	−Bk	0 minus (Bk) to Xi
77ijk	SXi	−Bk+Bj	(Bj) minus (Bk) to Xi

FUNCTIONAL UNITS

Model 74 Octal Codes	Model 76 Octal Codes
Branch 00-07	Boolean 10-17
Boolean 10-17	26,27
Shift 20-27	Shift 20-23
43	43
FP Add 30-35	Normalize 24, 25
Long Add 36, 37	FP Add 30-35
FP Multiply 40-42	Long Add 36, 37
FP Divide 44, 45, 47	FP Multiply 40-42
Increment 50-77	FP Divide 44, 45
	Population 47
	Increment 50-77

CMU INSTRUCTIONS

Instruction	Operation	Variable	Description
464 0 K	IM	K	Move data according to word at K
464 j K	IM	Bj+K	Move data according to word at Bj+K
464 j 000000	IM	Bj	Move data according to word at Bj
0 ℓU k_s ℓL c_s k_d c_d k_d	MD	ℓ,k_s,c_s,k_d,c_d	Indirect move descriptor word
465 ℓU k_s ℓL c_s k_d c_d k_d	DM	ℓ,k_s,d_s,k_d,c_d	Direct move
466 ℓU k_a ℓL c_a k_b c_b k_b	CC	ℓ,k_a,c_a,k_b,c_b	Compare collated
467 ℓU k_a ℓL c_a c_b k_b	CU	ℓ,k_a,c_a,k_b,c_b	Compare uncollated

PPU INSTRUCTIONS

(1) CYBER 70 Model 76, and 7600
(2) CYBER 170 Series, CYBER 70 Models 71, 72, 73, and 74, and 6000 Series
(3) d is required

Instruction	Operation	Variable	Description
01d m	LJM	m, d	Long jump to m + (d)
02d m	RJM	m, d	Return jump to m + (d)
03d	UJN	r	Unconditional jump to p + r
04d	ZJN	r	Zero jump to p + r
05d	NJN	r	Nonzero jump to p + r
06d	PJN	r	Positive jump to p + r
07d	MJN	r	Negative jump to p + r
10r	SHN	r	Shift (A) left-circular (+r) or right-end off (−r)
11d	LMN	d	Logical difference; (A)−d→A
12d	LPN	d	Logical product; (A) * d→A
13d	SCN	d	Selective clear; (A) at each d bit set
14d	LDN	d	Load d→A
15d	LCN	d	Load complement d→A
16d	ADN	d	Add d + (A)→A
17d	SBN	d	Subtract (A)−d→A
20d m	LDC	c	Load c→A
21d m	ADC	c	Add (A) + c→A
22d m	LPC	c	Logical product; (A) * c→A
23d m	LMC	c	Logical difference; (A)−c→A
2400	PSN		Pass
260d	EXN	d (2)	Exchange jump CPU d unconditionally to (A)
260d	ETN	d (2)	6416 Extended transfer
261d	MXN	d (2)	Monitor exchange jump CPU d to (A)
262d	MAN	d (2)	Monitor exchange jump CPU d to (MA)
270d	RPN	d (2)	Read program address of CPU d to A
270d	ERN	d (2)	6416 Extended read status
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
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 (A) + ((d))→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
50d m	LDM	m, d	Load (m + (d))→A
51d m	ADM	m, d	Add (m + (d)) + (A)→A
52d m	SBM	m, d	Subtract (A)−(m + (d))→A
53d m	LMM	m, d	Logical difference (A)−(m + (d))→A
54d m	STM	m, d	Store (A)→m + (d)
55d m	RAM	m, d	Replace add; (A) + (m + (d))→m + (d) and A
56d m	AOM	m, d	Replace add one; (m + (d)) + 1→m + (d) and A
57d m	SOM	m, d	Replace subtract one; (m + (d))−1→m + (d) and A
60d m	FIM	m, d (1) (3)	Jump to m on input word flag on channel d
60d	CRD	d (2)	Central read from (A) to d
61d m	EIM	m, d (1) (3)	Jump to m if no input word flag on channel d
61d m	CRM	m, d (2) (3)	Central read (d) CM words beginning from CM address (A) to beginning PPU address m
62d m	IRM	m, d (1) (3)	Jump to m on input record flag on channel d
62d	CWD	d (2)	Central write from d to (A)
63d m	NIM	m, d (1) (3)	Jump to m if no input record flag on channel d
63d m	CWM	m, d (2) (3)	Central write (d) CM words beginning from PPU address m to beginning CM address (A)
64d m	FOM	m, d (1) (3)	Jump to m on output word flag on channel d
64d m	AJM	m, d (2) (3)	Jump to m if channel d is active
65d m	EOM	m, d (1) (3)	Jump to m if no output word flag on channel d
65d m	IJM	m, d (2) (3)	Jump to m if channel d is inactive
66d m	ORM	m, d (1) (3)	Jump to m on output record flag on channel d
66d m	FJM	m, d (2) (3)	Jump to m if channel d is full
67d m	NOM	m, d (1) (3)	Jump to m if no output record flag on channel d
67d m	EJM	m, d (2) (3)	Jump to m if channel d is empty
70d	IAN	d	Input to A from channel d
71d m	IAM	m, d (3)	Input (A) words to m from channel d
72d	OAN	d (3)	Output from A on channel d
73d m	OAM	m, d	Output (A) words from m on channel d
74d	RFN	d (1)	Send record flag on channel d
74d	ACN	d (2)	Activate channel d
75d	DCN	d (2)	Disconnect channel d
76d	FAN	d (2)	Function (A) on channel d
77d m	FNC	m, d (3)	Function m on channel d
7700	ESN	d (1)	Error stop

Publications and Graphics Division
215 Moffett Park Drive
Sunnyvale, California 94086

CONTROL DATA CORPORATION

CONTROL DATA CORPORATION

COMPASS VERSION 3 SUMMARY

Pub. No. 60493000 A March 1971

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

CONTROL STATEMENT

COMPASS $(p_1,p_2,\ldots,p_n)$ **or** **COMPASS.**

A	omitted	Do not abort	ML	omitted or ML	MODLEVEL returns JDATE
	A	Abort on assembly errors		ML=string	MODLEVEL returns 9-character string
B	omitted or B	Binary on LGO	N	omitted	Normal ejects
	B=0	No binary		N	No ejects
	B=lfn	Binary on lfn	O	omitted or O	Short list on OUTPUT
D	omitted	No debug mode		O=lfn	Short list on lfn
	D	Debug mode		O=0	No short list
F	F	*F returns 0	PC	omitted or PC	PCOMMENT is 30 blanks
	F=n	*F returns n (decimal)		PC=string	PCOMMENT is 30-character string
	F=name	*F returns as follows: COMPASS 0 RUN-type 1 FTN-type 2	P	omitted	New pagination on END
				P	Continue pagination
G†	omitted or G=0	No system text	S†	omitted	SYSTEXT overlay
	G	System text on SYSTEXT		S	SYSTEXT on global library
	G=lfn	Overlay on lfn		S=0	No system text
	G=lfn/ovl	Named overlay on lfn		S=ovl	Named overlay on library
I	omitted	Source on INPUT		S=lib/ovl	Named overlay on named library
	I	Source on COMPILE			
	I=lfn	Source on lfn	X	omitted	XTEXT on OLDPL
L	omitted or L	Full list on OUTPUT		X=lfn	XTEXT on lfn
	L=lfn	List on lfn		X	XTEXT on OPL
	L=0	No full list			
LO	omitted or LO=0	Selects B,L,N, and R			
	LO	Selects C,F,G,X			
	LO=$c_1c_2\ldots c_n$	Deselects if c_i is B,L,N, or R; selects if c_i is other.			

†Seven G and S parameters allowed

CYBER 70 MODEL 71, 72, 73, 74 AND CYBER 170 EXIT MODES

Bit 50	Bit 49	Bit 48
INDEFINITE OPERAND	OPERAND OUT OF RANGE	ADDRESS OUT OF RANGE

CYBER 70 MODEL 76 PSD REGISTER

MODE			CONDITION													
Exit	Mon	Step	Ind	Ovf	Undf	LPar	SPar	LBlk	SBlk	LDir	SDir	Prog	Bkp	Step	Ind	Ovf	Undf
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

FATAL ERRORS

A	ADDRESS FIELD BAD
B	DOUBLY DEFINED SYMBOL. THE FIRST DEFINITION HOLDS.
E	ECHO, DUP, RMT, OR MACRO ILLEGALLY NESTED.
F	NUMBER OF ENTRIES EXCEEDS PERMISSIBLE AMOUNT.
L	LOCATION FIELD BAD.
N	NEGATIVE RELOCATION ON ENTRY POINT.
O	OPERATION FIELD BAD.
P	CONSULT LISTING FOR REASON BEHIND P-ERROR.
R	DATA ORIGIN OUTSIDE BLOCK OR IN BLANK COMMON BLOCK.
U	UNDEFINED SYMBOL. VALUE ASSUMED 0.
V	BIT COUNT ERROR ON VFD (MUST BE $0 \leq \text{COUNT} \leq 60$).

INFORMATIVE ERRORS

1	LOCATION SYMBOL BAD. SYMBOL NOT DEFINED.
2	ADDRESS ERROR ON SYMBOL DEFINITION.
3	DUPLICATE MACRO DEFINITION. NEW ONE OVERRIDES.
4	BAD FORMAL PARAMETER NAME IGNORED.
5	CPU OPERATION SYNTAX INCORRECTLY SPECIFIED.
6	LOCATION FIELD MEANINGLESS.
7	ADDRESS VALUE EXCEEDS FIELD SIZE, RESULT TRUNCATED.
8	MISSING OR EXTRA ADDRESS SUBFIELD.
9	MICRO SUBSTITUTION ERROR. NO SUBSTITUTION.

CHARACTER SETS

Char.	Display	Hollerith	BCD Ext.	BCD Int.	ASCII Subset Char.	ASCII Subset Code	ASCII Subset Punch
:		8-2	00	12	:	3A	8-2
A	01	12-1	61	21	A	41	12-1
B	02	12-2	62	22	B	42	12-2
C	03	12-3	63	23	C	43	12-3
D	04	12-4	64	24	D	44	12-4
E	05	12-5	65	25	E	45	12-5
F	06	12-6	66	26	F	46	12-6
G	07	12-7	67	27	G	47	12-7
H	10	12-8	70	30	H	48	12-8
I	11	12-9	71	31	I	49	12-9
J	12	11-1	41	41	J	4A	11-1
K	13	11-2	42	42	K	4B	11-2
L	14	11-3	43	43	L	4C	11-3
M	15	11-4	44	44	M	4D	11-4
N	16	11-5	45	45	N	4E	11-5
O	17	11-6	46	46	O	4F	11-6
P	20	11-7	47	47	P	50	11-7
Q	21	11-8	50	50	Q	51	11-8
R	22	11-9	51	51	R	52	11-9
S	23	0-2	22	62	S	53	0-2
T	24	0-3	23	63	T	54	0-3
U	25	0-4	24	64	U	55	0-4
V	26	0-5	25	65	V	56	0-5
W	27	0-6	26	66	W	57	0-6
X	30	0-7	27	67	X	58	0-7
Y	31	0-8	30	70	Y	59	0-8
Z	32	0-9	31	71	Z	5A	0-9
0	33	0	12	00	0	30	0
1	34	1	01	01	1	31	1
2	35	2	02	02	2	32	2
3	36	3	03	03	3	33	3
4	37	4	04	04	4	34	4
5	40	5	05	05	5	35	5
6	41	6	06	06	6	36	6
7	42	7	07	07	7	37	7
8	43	8	10	10	8	38	8
9	44	9	11	11	9	39	9
+	45	12	60	20	+	2B	12-8-6
−	46	11	40	40	−	2D	11
*	47	11-8-4	54	54	*	2A	11-8-4
/	50	0-1	21	61	/	2F	0-1
(	51	0-8-4	34	74	(	28	12-8-5
)	52	12-8-4	74	34	)	29	11-8-5
$	53	11-8-3	53	53	$	24	11-8-3
=	54	8-3	13	13	=	3D	8-6
space	55	space	20	60	space	20	space
,	56	0-8-3	33	73	,	2C	0-8-3
.	57	12-8-3	73	33	.	2E	12-8-3
≡	60	0-8-6	36	76	#	23	8-3
[	61	8-7	17	17	[	5B	12-8-2
]	62	0-8-2	32	72	]	5D	11-8-2
%	63	8-6	16	16	%	25	0-8-4
≠	64	8-4	14	14	¬	22	8-7
→	65	0-8-5	35	75	_	5F	0-8-5
∨	66	11-0 ①	52	52	`	21	12-8-7 ③
∧	67	0-8-7	37	77	&	26	12
↑	70	11-8-5	55	55	'	27	8-5
↓	71	11-8-6	56	56	?	3F	0-8-7
<	72	12-0 ②	72	32	<	3C	12-8-4 ④
>	73	11-8-7	57	57	>	3E	0-8-6
≤	74	8-5	15	15	@	40	8-4
≥	75	12-8-5	75	35	\	5C	0-8-2
¬	76	12-8-6	76	36	^	5E	11-8-7
;	77	12-8-7	77	37	;	3B	11-8-6

① 11-0 and 11-8-2 are equivalent
② 12-0 and 12-8-2 are equivalent
③ 11-0 and 12-8-7 are equivalent
④ 12-0 and 12-8-4 are equivalent

LANGUAGE ELEMENTS

SPECIAL CHARACTERS

Any Field

≠	Micro substitution
↦	Concatenation

First Column

*	Comments line
'	Continuation

Location Field

+	Force Upper
−	Negate Force Upper

Variable Field

=item	Literal
=$sym	Deferred Symbol
=Xsym	Strong External Symbol
=Ysym	Weak External Symbol
*O	Origin Counter
* or *L	Location Counter
*P	Position Counter
$	Position Counter − 1
*F	Caller: COMPASS 0 RUN-type 1 FTN-type 2
,	Subfield Delimiter
/name/	Symbol Qualifier

CHARACTER NOTATION

Data Item: sign | n | t | string
or
sign | t | d | string | d

Constant: n | t | string

Literal: = | sign | n | t | string
or
= | sign | t | d | string | d

t = C Left, 12 zero bits
H Left, blank fill
A Right, blank fill
R Right, zero fill
L Left, zero fill
Z Left, 6 zero bits
d = delimiter character
n = no. of characters
string = Code characters

NUMERIC NOTATION

Data Item: sign | prerx | value | mods

Constant: value | mods

Literal: = | sign | prerx | value | mods

Mods	Spec	Default
Sign	+ or −	+
Radix (pre/post)	O or B / D	BASE / Pseudo
Integer	n	0
Fraction	.n or .	none
Pwr 10 sgl	E or En or E±n	none
Pwr 10 dbl	EE or EEn or EE±n	none
Pwr 2	S or Sn or S±n	none
Binary pt	P or Pn or P±n	none

SYSTEM MICROS

DATE, JDATE, TIME, BASE, CODE, QUAL, SEQUENCE, MODLEVEL, PCOMMENT

PSEUDO INSTRUCTIONS

PROGRAM DEFINITION

	IDENT	*name,fwa,eptsym*
sym	END	*trasym*

BINARY CONTROL

	ABS	
	MACHINE	type,hf_1,hf_2,...,hf_n type = 6 or 7 hf_i = C,D,I,L,X
	PPU	J
	PERIPH	J
	IDENT	*name, org, entry, ℓ₁,ℓ₂*
	IDENT	*name, org, entry, ppu*
name	SEGMENT	*fwa,eptsym*
	SET	
record	STEXT	
	LDSET	$p_1,p_2,\ldots,p_n$
	LCC	*directive*
	COMMENT	*string*
	NOLABEL	I

MODE CONTROL

mname	BASE	O *or* D *or* M *or* *
	CODE	A *or* D *or* E *or* I *or* *
	QUAL	*name or**
	B1=1	
	B7=1	
	COL	*n*

COUNTER CONTROL

	USE	**or name or // or /name/*
	USELCM	**or name or // or /name/*
	ORG	*exp*
	ORGC	*exp*
sym	BSS	*exp*
	LOC	*exp*
	POS	*aexp*

SYMBOL DEFINITION

sym	=	*exp*
sym	EQU	*exp*
sym	SET	*exp*
sym	MAX	$exp_1,exp_2,\ldots,exp_n$
sym	MIN	$exp_1,exp_2,\ldots,exp_n$
sym	MICCNT	*mname*
	SST	*sym ...*

LINKAGE CONTROL

	ENTRY	$sym_1,sym_2,\ldots,sym_n$
	ENTRYC	$sym_1,sym_2,\ldots,sym_n$
	EXT	$sym_1,sym_2,\ldots,sym_n$

DATA GENERATION

sym	BSSZ	*exp*
sym	DATA	$item_1,\ldots,item_n$
sym	DIS	*n,string*
sym	DIS	*,dstringd*
sym	LIT	$item_1,\ldots,item_n$
sym	VFD	$item_1/exp_1,\ldots,exp_n$
sym	CON	$item_1/exp_1,\ldots,exp_n$
sym	R=	*reg,exp*
	REP	S/addr,D/addr,C/rep,B/bsz,I/inc
	REPC	S/addr,D/addr,C/rep,B/bsz,I/inc
	REPI	S/addr,D/addr,C/rep,B/bsz,I/inc

ERROR FLAG

t	ERR	
t	ERR*op*	*aexp*

op = ZR, NZ, PL, NG, MI

CONDITIONAL ASSEMBLY

name	IF*op*	$exp_1,exp_2,\ell nct$	*op*	*att*
name	IFMI	*exp,ℓnct*	EQ	ABS
name	IFPL	*exp,ℓnct*	GE	COM
name	IFCP	*ℓnct*	GT	DEF
name	IFCP6	*ℓnct*	LE	EXT
name	IFCP7	*ℓnct*	LT	LOM
name	IFPP	*ℓnct*	NE	LOC
name	IFPP6	*ℓnct*		MIC
name	IFPP7	*ℓnct*		REG
name	IF	*att,exp,ℓnct*		REL
name	IF	*−att,exp,ℓnct*		SET
name	IFC	*op,dstring₁ dstring₂* d,ℓnct		SST
name	IFC	*−op,dstring₁ dstring₂* d,ℓnct		
name	ENDIF			
name	ELSE	*ℓnct*		
name	SKIP	*ℓnct*		

LIST CONTROL

	LIST	$p_1,p_2,\ldots,p_n$ *or* *
	p = A	Assembly
	B	Binary control
	C	Control statements
	D	Detail
	E	Echoed lines
	F	IF - Skipped lines
	G	Code generation
	L	Reference table only
	M	User macros
	N	Referenced symbols only
	R	No references
	S	System macros
	T	SST symbols
	X	XTEXT lines
name	EJECT	
name	SPACE	$aexp_1,aexp_2$
name	TITLE	*string*
	NOREF	$sym_1,\ldots,sym_n$
name	TTL	*string*
name	CTEXT	*string*
	ENDX	
	XREF	A *or* B

DEFINITION OPERATIONS

file	XTEXT	*record*
name	DUP	*rep,ℓnct*
name	ECHO	*ℓnct,*p_1*=(list*$_1$*),*p_2*=(list*$_2$*),...*
name	ENDD	
	STOPDUP	
name	RMT	
name	HERE	
name	MACRO	$p_1,p_2,\ldots,p_n$
	MACRO	$name,p_1,p_2,\ldots,p_n$
name	MACROE	$p_1,p_2,\ldots,p_n$
	MACROE	$name,p_1,p_2,\ldots,p_n$
sytx	OPDEF	$p_1,p_2,\ldots,p_n$
name	ENDM	
	LOCAL	$sym_1,\ldots,sym_n$
	IRP	*p*

OP CODE MANAGEMENT

name	PPOP	*ctl,val,type*
$name_1$	OPSYN	$name_2$
	NIL	
	PURGMAC	$name_1,name_2,\ldots,name_n$
	PURGDEF	*sytx*
sytx	CPOP	*ctl,val,reg,type*
$sytx_1$	CPSYN	$sytx_2$

MICRO

mname	MICRO	$n_1,n_2,dstringd$
mname	MICRO	$n_1,dstringd$
mname	DECMIC	*aexp,n*
mname	OCTMIC	*aexp,n*