SFT

CRAY RESEARCH, INC.

CRAY X-MP CAL REFERENCE CARD

SQ-0067

CAL CONTROL STATEMENT

```
CAL,CPU=type,I=idn,L=ldn,B=bdn,E=edn,ABORT,DEBUG,options,
    LIST=name,S=sdn,SYM=sym,T=bst,X=xdn.
```

CPU	Omitted	Machine currently executing CAL
	CPU=*type*	Specify CRAY-1 or CRAY-XMP
I	Omitted	Source on $IN
	I=*idn*	Source on *idn*
L	Omitted	List output on $OUT
	L=0	No list output
	L=*ldn*	List output on *ldn*
B	Omitted	Binary on $BLD
	B=0	No binary
	B=*bdn*	Binary on *bdn*
E	Omitted	No error listing
	E	Error list on $OUT
	E=*edn*	Error list on *edn* unless *edn*=*ldn*, then *ldn*
ABORT	Omitted	Do not abort
	ABORT	Abort on fatal error during assembly
DEBUG	Omitted	Write binary record on fatal error and set fatal error flag
	DEBUG	Write binary record with fatal error flag clear
options:	See *options* under CAL control statement in CAL Reference Manual (*options* overrides the LIST pseudo.)	
LIST	Omitted	LIST pseudos with a null (empty) location field processed
	LIST	All LIST pseudos processed
	LIST=*name*	LIST pseudo instructions with a location field matching name processed
S	Omitted	$SYSTXT
	S=0	No system text
	S=*sdn*	System text on *sdn*
SYM	Omitted	No symbol table
	SYM	Symbol table on dataset holding binary load data
	SYM=*sym*	Symbol table on *sym*
T	Omitted	No binary system text written
	T=0	No binary system text written
	T	Binary dataset written to $BST
	T=*bst*	Binary system text written to *bst*
X	Omitted	No global cross-reference records written
	X=0	No global cross-reference records written
	X	Global cross-reference records written to $XRF
	X=*xdn*	Global cross-reference records written to *xdn*

INSTRUCTIONS

CRAY X-MP	CAL	UNIT	DESCRIPTION
000000	ERR	-	Error exit
††0010*jk*	CA,A*j* A*k*	-	Set the channel (A*j*) current address to(A*k*) and begin the I/O sequence
††0011*jk*	CL,A*j* A*k*	-	Set the channel (A*j*) limit address to (A*k*)
††0012*j*0	CI,A*j*	-	Clear channel (A*j*) interrupt flag; clear device master-clear (output channel).
††0012*j*1	MC,A*j*	-	Clear channel (A*j*) interrupt flag; set device master-clear (output channel); clear device ready-held (input channel).
††0013*j*0	XA A*j*	-	Enter XA register with (A*j*)
††0014*j*0	RT S*j*	-	Enter RTC register with (S*j*)
††001401	IP 1	-	Set interprocessor interrupt
††001402	IP 0	-	Clear interprocessor interrupt
††001403	CLN 0	-	Enter CLN register with 0
††001413	CLN 1	-	Enter CLN register with 1
††001423	CLN 2	-	Enter CLN register with 2
††001433	CLN 3	-	Enter CLN register with 3
††0014*j*4	PCI S*j*	-	Enter II register with (S*j*)
††001405	CCI	-	Clear PCI request
††001406	ECI	-	Enable PCI request
††001407	DCI	-	Disable PCI request
00200*k*	VL A*k*	-	Transmit (A*k*) to VL register
†002000	VL 1	-	Transmit 1 to VL register
002100	EFI	-	Enable interrupt on floating-point error
002200	DFI	-	Disable interrupt on floating-point error
002300	ERI	-	Enable operand range interrupts
002400	DRI	-	Disable operand range interrupts
002500	DBM	-	Disable bidirectional memory transfers
002600	EBM	-	Enable bidirectional memory transfers
002700	CMR	-	Complete memory references
0030*j*0	VM S*j*	-	Transmit (S*j*) to VM register
†003000	VM 0	-	Clear VM register
0034*jk*	SM*jk* 1,TS	-	Test & set semaphore *jk* in SM
0036*jk*	SM*jk* 0	-	Clear semaphore *jk* in SM
0037*jk*	SM*jk* 1	-	Set semaphore *jk* in SM
004000	EX	-	Normal exit
0050*jk*	J B*jk*	-	Jump to (B*jk*)
006*ijkm*	J *exp*	-	Jump to *exp*
007*ijkm*	R *exp*	-	Return jump to *exp*; set B00 to P.
010*ijkm*	JAZ *exp*	-	Branch to *exp* if (A0)=0
011*ijkm*	JAN *exp*	-	Branch to *exp* if (A0)≠0
012*ijkm*	JAP *exp*	-	Branch to *exp* if (A0) positive; 0 is positive.
013*ijkm*	JAM *exp*	-	Branch to *exp* if (A0) negative
014*ijkm*	JSZ *exp*	-	Branch to *exp* if (S0)=0
015*ijkm*	JSN *exp*	-	Branch to *exp* if (S0)≠0
016*ijkm*	JSP *exp*	-	Branch to *exp* if (S0) positive; 0 is positive.
017*ijkm*	JSM *exp*	-	Branch to *exp* if (S0) negative
†††020*ijkm*	A*i* *exp*	-	Transmit *exp*=*jkm* to A*i*
†††021*ijkm*	A*i* *exp*	-	Transmit *exp*=ones complement of *jkm* to A*i*
†††022*ijk*	A*i* *exp*	-	Transmit *exp*=*jk* to A*i*
023*ij*0	A*i* S*j*	-	Transmit (S*j*) to A*i*
023*i*01	A*i* VL	-	Transmit (VL) to A*i*
024*ijk*	A*i* B*jk*	-	Transmit (B*jk*) to A*i*
025*ijk*	B*jk* A*i*	-	Transmit (A*i*) to B*jk*
026*ij*0	A*i* PS*j*	Pop/LZ	Population count of (S*j*) to A*i*
026*ij*1	A*i* QS*j*	Pop/LZ	Population count parity of (S*j*) to A*i*
026*ij*7	A*i* SB*j*	-	Transmit (SB*j*) to A*i*

CRAY X-MP	CAL	UNIT	DESCRIPTION
027*ij*0	A*i* ZS*j*	Pop/LZ	Leading zero count of (S*j*) to A*i*
027*ij*7	SB*j* A*i*	-	Transmit (A*i*) to SB*j*
030*ijk*	A*i* A*j*+A*k*	A Int Add	Integer sum of (A*j*) and (A*k*) to A*i*
†030*i*0*k*	A*i* A*k*	A Int Add	Transmit (A*k*) to A*i*
†030*ij*0	A*i* A*j*+1	A Int Add	Integer sum of (A*j*) and 1 to A*i*
031*ijk*	A*i* A*j*-A*k*	A Int Add	Integer difference of (A*j*) less (A*k*) to A*i*
† †††031*i*00	A*i* -1	A Int Add	Transmit -1 to A*i*
†031*i*0*k*	A*i* -A*k*	A Int Add	Transmit the negative of (A*k*) to A*i*
†031*ij*0	A*i* A*j*-1	A Int Add	Integer difference of (A*j*) less 1 to A*i*
032*ijk*	A*i* A*j**A*k*	A Int Mult	Integer product of (A*j*) and (A*k*) to A*i*
033*i*00	A*i* CI	-	Channel number to A*i* (*j*=0)
033*ij*0	A*i* CA,A*j*	-	Address of channel (A*j*) to A*i* (*j*≠0; *k*=0)
033*ij*1	A*i* CE,A*j*	-	Error flag of channel (A*j*) to A*i* (*j*≠0; *k*=1)
034*ijk*	B*jk*,A*i* ,A0	Memory	Read (A*i*) words to B register *jk* from (A0)
†034*ijk*	B*jk*,A*i* 0,A0	Memory	Read (A*i*) words to B register *jk* from (A0)
035*ijk*	,A0 B*jk*,A*i*	Memory	Store (A*i*) words at B register *jk* to (A0)
†035*ijk*	0,A0 B*jk*,A*i*	Memory	Store (A*i*) words at B register *jk* to (A0)
036*ijk*	T*jk*,A*i* ,A0	Memory	Read (A*i*) words to T register *jk* from (A0)
†036*ijk*	T*jk*,A*i* 0,A0	Memory	Read (A*i*) words to T register *jk* from (A0)
037*ijk*	,A0 T*jk*,A*i*	Memory	Store (A*i*) words at T register *jk* to (A0)
†037*ijk*	0,A0 T*jk*,A*i*	Memory	Store (A*i*) words at T register *jk* to (A0)
040*ijkm*	S*i* *exp*	-	Transmit *jkm* to S*i*
041*ijkm*	S*i* *exp*	-	Transmit *exp*=ones complement of *jkm* to S*i*
042*ijk*	S*i* <*exp*	S Logical	Form ones mask *exp* bits in S*i* from the right; *jk* field gets 64-*exp*.
†042*ijk*	S*i* #>*exp*	S Logical	Form zeros mask *exp* bits in S*i* from the left; *jk* field gets *exp*.
†042*i*77	S*i* 1	S Logical	Enter 1 into S*i*
†042*i*00	S*i* -1	S Logical	Enter -1 into S*i*
043*ijk*	S*i* >*exp*	S Logical	Form ones mask *exp* bits in S*i* from the left; *jk* field gets *exp*.
†043*ijk*	S*i* #<*exp*	S Logical	Form zeros mask *exp* bits in S*i* from the right; *jk* field gets 64-*exp*.
†043*i*00	S*i* 0	S Logical	Clear S*i*
044*ijk*	S*i* S*j*&S*k*	S Logical	Logical product of (S*j*) and (S*k*) to S*i*
†044*ij*0	S*i* S*j*&SB	S Logical	Sign bit of (S*j*) to S*i*
†044*ij*0	S*i* SB&S*j*	S Logical	Sign bit of (S*j*) to S*i* (*j*≠0)
045*ijk*	S*i* #S*k*&S*j*	S Logical	Logical product of (S*j*) and ones complement of (S*k*) to S*i*
†045*ij*0	S*i* #SB&S*j*	S Logical	(S*j*) with sign bit cleared to S*i*
046*ijk*	S*i* S*j*\S*k*	S Logical	Logical difference of (S*j*) and (S*k*) to S*i*
†046*ij*0	S*i* S*j*\SB	S Logical	Toggle sign bit of S*j*, then enter into S*i*
†046*ij*0	S*i* SB\S*j*	S Logical	Toggle sign bit of S*j*, then enter into S*i* (*j*≠0)
047*ijk*	S*i* #S*j*\S*k*	S Logical	Logical equivalence of (S*k*) and (S*j*) to S*i*
†047*i*0*k*	S*i* #S*k*	S Logical	Transmit ones complement of (S*k*) to S*i*
†047*ij*0	S*i* #S*j*\SB	S Logical	Logical equivalence of (S*j*) and sign bit to S*i*
†047*ij*0	S*i* #SB\S*j*	S Logical	Logical equivalence of (S*j*) and sign bit to S*i* (*j*≠0)
†047*i*00	S*i* #SB	S Logical	Enter ones complement of sign bit into S*i*
050*ijk*	S*i* S*j*!S*i*&S*k*	S Logical	Logical product of (S*i*) and (S*k*) complement ORed with logical product of (S*j*) and (S*k*) to S*i*
†050*ij*0	S*i* S*j*!S*i*&SB	S Logical	Scalar merge of (S*i*) and sign bit of (S*j*) to S*i*
051*ijk*	S*i* S*j*!S*k*	S Logical	Logical sum of (S*j*) and (S*k*) to S*i*
†051*i*0*k*	S*i* S*k*	S Logical	Transmit (S*k*) to S*i*
†051*ij*0	S*i* S*j*!SB	S Logical	Logical sum of (S*j*) and sign bit to S*i*
†051*ij*0	S*i* SB!S*j*	S Logical	Logical sum of (S*j*) and sign bit to S*i* (*j*≠0)
†051*i*00	S*i* SB	S Logical	Enter sign bit into S*i*
052*ijk*	S0 S*i*<*exp*	S Shift	Shift (S*i*) left *exp*=*jk* places to S0

CRAY X-MP	CAL	UNIT	DESCRIPTION
053*ijk*	S\ Si>*exp*	S Shift	Shift (S*i*) right *exp*=64-*jk* places to S0
054*ijk*	Si Si<*exp*	S Shift	Shift (S*i*) left *exp*=*jk* places
055*ijk*	Si Si>*exp*	S Shift	Shift (S*i*) right *exp*=64-*jk* places
056*ijk*	Si Si,Sj<Ak	S Shift	Shift (S*i* and S*j*) left (A*k*) places to S*i*
†056*ij*0	Si Si,Sj<1	S Shift	Shift (S*i* and S*j*) left one place to S*i*
†056*i*0*k*	Si Si<Ak	S Shift	Shift (S*i*) left (A*k*) places to S*i*
057*ijk*	Si Sj,Si>Ak	S Shift	Shift (S*j* and S*i*) right (A*k*) places to S*i*
†057*ij*0	Si Sj,Si>1	S Shift	Shift (S*j* and S*i*) right one place to S*i*
†057*i*0*k*	Si Si>Ak	S Shift	Shift (S*i*) right (A*k*) places to S*i*
060*ijk*	Si Sj+Sk	S Int Add	Integer sum of (S*j*) and (S*k*) to S*i*
061*ijk*	Si Sj-Sk	S Int Add	Integer difference of (S*j*) and (S*k*) to S*i*
†061*i*0*k*	Si -Sk	S Int Add	Transmit negative of (S*k*) to S*i*
062*ijk*	Si Sj+FSk	Fp Add	Floating-point sum of (S*j*) and (S*k*) to S*i*
†062*i*0*k*	Si +FSk	Fp Add	Normalize (S*k*) to S*i*
063*ijk*	Si Sj-FSk	Fp Add	Floating-point difference of (S*j*) and (S*k*) to S*i*
†063*i*0*k*	Si -FSk	Fp Add	Transmit normalized negative of (S*k*) to S*i*
064*ijk*	Si Sj*FSk	Fp Mult	Floating-point product of (S*j*) and (S*k*) to S*i*
065*ijk*	Si Sj*HSk	Fp Mult	Half-precision rounded floating-point product of (S*j*) and (S*k*) to S*i*
066*ijk*	Si Sj*RSk	Fp Mult	Full-precision rounded floating-point product of (S*j*) and (S*k*) to S*i*
067*ijk*	Si Sj*ISk	Fp Mult	2-floating-point product of (S*j*) and (S*k*) to S*i*
070*ij*0	Si /HSj	Fp Rcpl	Floating-point reciprocal approximation of (S*j*) to S*i*
071*i*0*k*	Si Ak	-	Transmit (A*k*) to S*i* with no sign extension
071*i*1*k*	Si +Ak	-	Transmit (A*k*) to S*i* with sign extension
071*i*2*k*	Si +FAk	-	Transmit (A*k*) to S*i* as unnormalized floating-point number
071*i*30	Si 0.6	-	Transmit constant 0.75*2**48 to S*i*
071*i*40	Si 0.4	-	Transmit constant 0.5 to S*i*
071*i*50	Si 1.	-	Transmit constant 1.0 to S*i*
071*i*60	Si 2.	-	Transmit constant 2.0 to S*i*
071*i*70	Si 4.	-	Transmit constant 4.0 to S*i*
072*i*00	Si RT	-	Transmit (RTC) to S*i*
072*i*02	Si SM	-	Transmit (SM) to S*i*
072*ij*3	Si STj	-	Transmit (ST*j*) to S*i*
073*i*00	Si VM	-	Transmit (VM) to S*i*
073*ij*1	Si SRj	-	Transmit (SR*j*) to S*i* (*j*=0)
073*i*02	SM Si	-	Transmit (S*i*) to SM
073*ij*3	STj Si	-	Transmit (S*i*) to ST*j*
074*ijk*	Si Tjk	-	Transmit (T*jk*) to S*i*
075*ijk*	Tjk Si	-	Transmit (S*i*) to T*jk*
076*ijk*	Si Vj,Ak	-	Transmit (V*j*, element (A*k*)) to S*i*
077*ijk*	Vi,Ak Sj	-	Transmit (S*j*) to V*i* element (A*k*)
†077*i*0*k*	Vi,Ak 0	-	Clear V*i* element (A*k*)
10*hijkm*	Ai *exp*,Ah	Memory	Read from ((A*h*)+*exp*) to A*i* (A0=0)
†100*ijkm*	Ai *exp*,0	Memory	Read from (*exp*) to A*i*
†100*ijkm*	Ai *exp*,	Memory	Read from (*exp*) to A*i*
†10*hi*00 0	Ai ,Ah	Memory	Read from (A*h*) to A*i*
11*hijkm*	*exp*,Ah Ai	Memory	Store (A*i*) to (A*h*)+*exp* (A0=0)
†110*ijkm*	*exp*,0 Ai	Memory	Store (A*i*) to *exp*
†110*ijkm*	*exp*, Ai	Memory	Store (A*i*) to *exp*
†11*hi*00 0	,Ah Ai	Memory	Store (A*i*) to (A*h*)
12*hijkm*	Si *exp*,Ah	Memory	Read from ((A*h*)+*exp*) to S*i* (A0=0)
†120*ijkm*	Si *exp*,0	Memory	Read from *exp* to S*i*
†120*ijkm*	Si *exp*,	Memory	Read from *exp* to S*i*
†12*hi*00 0	Si ,Ah	Memory	Read from (A*h*) to S*i*
13*hijkm*	*exp*,Ah Si	Memory	Store (S*i*) to (A*h*)+*exp* (A0=0)
†130*ijkm*	*exp*,0 Si	Memory	Store (S*i*) to *exp*
†130*ijkm*	*exp*, Si	Memory	Store (S*i*) to *exp*

CRAY X-MP	CAL		UNIT	DESCRIPTION
†13*hi*00 0	,A*h*	S*i*	Memory	Store (S*i*) to (A*h*)
140*ijk*	V*i*	S*j*&V*k*	V Logical	Logical products of (S*j*) and (V*k*) to V*i*
141*ijk*	V*i*	V*j*&V*k*	V Logical	Logical products of (V*j*) and (V*k*) to V*i*
142*ijk*	V*i*	S*j*!V*k*	V Logical	Logical sums of (S*j*) and (V*k*) to V*i*
†142*i*0*k*	V*i*	V*k*	V Logical	Transmit (V*k*) to V*i*
143*ijk*	V*i*	V*j*!V*k*	V Logical	Logical sums of (V*j*) and (V*k*) to V*i*
144*ijk*	V*i*	S*j*\V*k*	V Logical	Logical differences of (S*j*) and (V*k*) to V*i*
145*ijk*	V*i*	V*j*\V*k*	V Logical	Logical differences of (V*j*) and (V*k*) to V*i*
†145*iii*	V*i*	0	V Logical	Clear V*i*
146*ijk*	V*i*	S*j*!V*k*&VM	V Logical	Transmit (S*j*) if VM bit=1; (V*k*) if VM bit=0 to V*i*.
†146*i*0*k*	V*i*	#VM&V*k*	V Logical	Vector merge of (V*k*) and 0 to V*i*
147*ijk*	V*i*	V*j*!V*k*&VM	V Logical	Transmit (V*j*) if VM bit=1; (V*k*) if VM bit=0 to V*i*.
150*ijk*	V*i*	V*j*<A*k*	V Shift	Shift (V*j*) left (A*k*) places to V*i*
†150*ij*0	V*i*	V*j*<1	V Shift	Shift (V*j*) left one place to V*i*
151*ijk*	V*i*	V*j*>A*k*	V Shift	Shift (V*j*) right (A*k*) places to V*i*
†151*ij*0	V*i*	V*j*>1	V Shift	Shift (V*j*) right one place to V*i*
152*ijk*	V*i*	V*j*,V*j*<A*k*	V Shift	Double shift (V*j*) left (A*k*) places to V*i*
†152*ij*0	V*i*	V*j*,V*j*<1	V Shift	Double shift (V*j*) left one place to V*i*
153*ijk*	V*i*	V*j*,V*j*>A*k*	V Shift	Double shift (V*j*) right (A*k*) places to V*i*
†153*ij*0	V*i*	V*j*,V*j*>1	V Shift	Double shift (V*j*) right one place to V*i*
154*ijk*	V*i*	S*j*+V*k*	V Int Add	Integer sums of (S*j*) and (V*k*) to V*i*
155*ijk*	V*i*	V*j*+V*k*	V Int Add	Integer sums of (V*j*) and (V*k*) to V*i*
156*ijk*	V*i*	S*j*-V*k*	V Int Add	Integer differences of (S*j*) and (V*k*) to V*i*
†156*i*0*k*	V*i*	-V*k*	V Int Add	Transmit negative of (V*k*) to V*i*
157*ijk*	V*i*	V*j*-V*k*	V Int Add	Integer differences of (V*j*) and (V*k*) to V*i*
160*ijk*	V*i*	S*j**FV*k*	Fp Mult	Floating-point products of (S*j*) and (V*k*) to V*i*
161*ijk*	V*i*	V*j**FV*k*	Fp Mult	Floating-point products of (V*j*) and (V*k*) to V*i*
162*ijk*	V*i*	S*j**HV*k*	Fp Mult	Half-precision rounded floating-point products of (S*j*) and (V*k*) to V*i*
163*ijk*	V*i*	V*j**HV*k*	Fp Mult	Half-precision rounded floating-point products of (V*j*) and (V*k*) to V*i*
164*ijk*	V*i*	S*j**RV*k*	Fp Mult	Rounded floating-point products of (S*j*) and (V*k*) to V*i*
165*ijk*	V*i*	V*j**RV*k*	Fp Mult	Rounded floating-point products of (V*j*) and (V*k*) to V*i*
166*ijk*	V*i*	S*j**IV*k*	Fp Mult	2-floating-point products of (S*j*) and (V*k*) to V*i*
167*ijk*	V*i*	V*j**IV*k*	Fp Mult	2-floating-point products of (V*j*) and (V*k*) to V*i*
170*ijk*	V*i*	S*j*+FV*k*	Fp Add	Floating-point sums of (S*j*) and (V*k*) to V*i*
†170*i*0*k*	V*i*	+FV*k*	Fp Add	Normalize (V*k*) to V*i*
171*ijk*	V*i*	V*j*+FV*k*	Fp Add	Floating-point sums of (V*j*) and (V*k*) to V*i*
172*ijk*	V*i*	S*j*-FV*k*	Fp Add	Floating-point differences of (S*j*) and (V*k*) to V*i*
†172*i*0*k*	V*i*	-FV*k*	Fp Add	Transmit normalized negatives of (V*k*) to V*i*
173*ijk*	V*i*	V*j*-FV*k*	Fp Add	Floating-point differences of (V*j*) and (V*k*) to V*i*
174*ij*0	V*i*	/HV*j*	Fp Rcpl	Floating-point reciprocal approximations of (V*j*) to V*i*
174*ij*1	V*i*	PV*j*	V Pop	Population counts of (V*j*) to V*i*
174*ij*2	V*i*	QV*j*	V Pop	Population count parities of (V*j*) to V*i*
1750*j*0	VM	V*j*,Z	V Logical	VM=1 where (V*j*)=0

CRAY X-MP	CAL	UNIT	DESCRIPTION
1750*j*1	VM Vj,N	V Logical	VM=1 where (V*j*)≠0
1750*j*2	VM Vj,P	V Logical	VM=1 if (V*j*) positive; 0 is positive.
1750*j*3	VM Vj,M	V Logical	VM=1 if (V*j*) negative
176*i*0*k*	V*i* ,A0,A*k*	Memory	Read (VL) words to V*i* from (A0) incremented by (A*k*)
†176*i*00	V*i* ,A0,1	Memory	Read (VL) words to V*i* from (A0) incremented by 1
1770*jk*	,A0,A*k* V*j*	Memory	Store (VL) words from V*j* to (A0) incremented by (A*k*)
†1770*j*0	,A0,1 V*j*	Memory	Store (VL) words from V*j* to (A0) incremented by 1

† Special syntax form
†† Privileged to monitor mode
††† Generated depending on the value of *exp*

REGISTER	VALUE
A*h*, *h*=0	0
A*i*, *i*=0	(A0)
A*j*, *j*=0	0
A*k*, *k*=0	1
S*i*, *i*=0	(S0)
S*j*, *j*=0	0
S*k*, *k*=0	2^{63}

LOGICAL OPERATORS	
&	0101
AND	1100
	0100
!	0101
OR	1100
	1101
\	0101
XOR	1100
	1001

CRAY RESEARCH, INC.
Software Development Division
1440 Northland Drive, Mendota Heights, Minnesota 55120

CHARACTER SET

CHAR	ASCII	ASCII CARD CODE
NUL	000	12-0-9-8-1
SOH	001	12-9-1
STX	002	12-9-2
ETX	003	12-9-3
EOT	004	9-7
ENQ	005	0-9-8-5
ACK	006	0-9-8-6
BEL	007	0-9-8-7
BS	010	11-9-6
HT	011	12-9-5
LF	012	0-9-5
VT	013	12-9-8-3
FF	014	12-9-8-4
CR	015	12-9-8-5
SO	016	12-9-8-6
SI	017	12-9-8-7
DLE	020	12-11-9-8-1
DC1	021	11-9-1
DC2	022	11-9-2
DC3	023	11-9-3
DC4	024	9-8-4
NAK	025	9-8-5
SYN	026	9-2
ETB	027	0-9-6
CAN	030	11-9-8
EM	031	11-9-8-1
SUB	032	9-8-7
ESC	033	0-9-7
FS	034	11-9-8-4
GS	035	11-9-8-5
RS	036	11-9-8-6
US	037	11-9-8-7
Space	040	None
!	041	12-8-7
"	042	8-7
#	043	8-3
$	044	11-8-3
%	045	0-8-4
&	046	12
'	047	8-5
(	050	12-8-5
)	051	11-8-5
*	052	11-8-4
+	053	12-8-6
,	054	0-8-3
-	055	11
.	056	12-8-3
/	057	0-1
0	060	0
1	061	1
2	062	2
3	063	3
4	064	4
5	065	5
6	066	6
7	067	7
8	070	8
9	071	9
:	072	8-2
;	073	11-8-6
<	074	12-8-4
=	075	8-6
>	076	0-8-6
?	077	0-8-7

CHAR	ASCII	ASCII CARD CODE
@	100	8-4
A	101	12-1
B	102	12-2
C	103	12-3
D	104	12-4
E	105	12-5
F	106	12-6
G	107	12-7
H	110	12-8
I	111	12-9
J	112	11-1
K	113	11-2
L	114	11-3
M	115	11-4
N	116	11-5
O	117	11-6
P	120	11-7
Q	121	11-8
R	122	11-9
S	123	0-2
T	124	0-3
U	125	0-4
V	126	0-5
W	127	0-6
X	130	0-7
Y	131	0-8
Z	132	0-9
[	133	12-8-2
\	134	0-8-2
]	135	11-8-2
^	136	11-8-7
_	137	0-8-5
`	140	8-1
a	141	12-0-1
b	142	12-0-2
c	143	12-0-3
d	144	12-0-4
e	145	12-0-5
f	146	12-0-6
g	147	12-0-7
h	150	12-0-8
i	151	12-0-9
j	152	12-11-1
k	153	12-11-2
l	154	12-11-3
m	155	12-11-4
n	156	12-11-5
o	157	12-11-6
p	160	12-11-7
q	161	12-11-8
r	162	12-11-9
s	163	11-0-2
t	164	11-0-3
u	165	11-0-4
v	166	11-0-5
w	167	11-0-6
x	170	11-0-7
y	171	11-0-8
z	172	11-0-9
{	173	12-0
|	174	12-11
}	175	11-0
~	176	11-0-1
DEL	177	12-9-7

PSEUDO INSTRUCTIONS

PROGRAM CONTROL

```
IDENT     name
END
ABS
COMMENT   'string'
```

MICROS

```
name   MICRO   'string',exp1,exp2
name   MICRO   'string',exp1
name   MICRO   'string'
name   OCTMIC  exp,count
name   DECMIC  exp,count
```

DATA DEFINITION

```
symbol   CON    exp1,exp2,...,expn
symbol   BSSZ   exp
symbol   DATA   data1,data2,...,datan
symbol   VWD    n1/exp1,n2/exp2,...,nm/expm
         REP    ct,swa,inc,bsz
```

LISTING CONTROL

```
name   LIST      op1,op2,...,opn
       LIST      *
       SPACE     count
       EJECT
       TITLE     'string'
       SUBTITLE  'string'
name   TEXT      'string'
       ENDTEXT
```

options

ON	OFF
XRF	NXRF
XNS	NXNS
DUP	NDUP
MAC	NMAC
MIF	NMIF
MIC	NMIC
LIS	NLIS
WEM	NWEM
TXT	NTXT
WRP	NWRP
WMR	NWMR

ERROR CONTROL

```
code   ERROR
code   ERRIF   exp1,op,exp2

op:    LT, LE, GT, GE, EQ, or NE
code:  See Fatal or Warning Errors
```

LOADER LINKAGE

```
ENTRY    symbol1,symbol2,...,symboln
EXT      sym1,sym2,...,symn
MODULE   modtype
START    symbol
```

CODE DUPLICATION

```
dupname   DUP      times
          DUP      times,count
dupname   ECHO     e1=(list1),e2=(list2),...,en=(listn)
dupname   ENDDUP
dupname   STOPDUP
```

SYMBOL DEFINITION

```
symbol   =        exp,attribute
symbol   SET      exp,attribute
symbol   MICSIZE  name

attribute:  P, W, V, or blank
```

CONDITIONAL ASSEMBLY

```
ifname   IFA    attribute,exp
         IFA    attribute,exp,count
ifname   IFE    exp1,op,exp2
         IFE    exp1,op,exp2,count
ifname   IFC    'string1',op,'string2'
         IFC    'string1',op,'string2',count
ifname   SKIP   count
ifname   ENDIF
ifname   ELSE
```

attribute

PA	parcel
WA	word
VAL	value
EXT	external
REL	relocatable
ABS	absolute
COM	common
DEF	defined
SET	SET-defined
REG	register
MIC	micro

(# can precede *attribute*)

BLOCK CONTROL

```
         BLOCK    name
         COMMON   name
         ORG      exp
symbol   BSS      exp
         LOC      exp
         BITW     exp
         BITP     exp
symbol   ALIGN
```

MODE CONTROL

```
BASE   O, D, M, or *
QUAL   qualification
QUAL   *
QUAL
```

MACRO DEFINITION

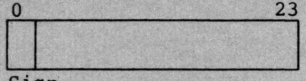
```
       MACRO
lfp    name    p1,p2,...,pn,e1=d1,e2=d2,...,em=dm
       LOCAL   sym1,...,symn
         .
         .          (body of definition)
         .
name   ENDM

name1  OPSYN   name2
```

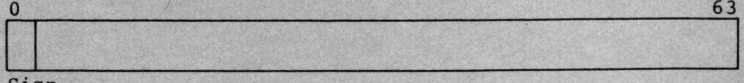
OPDEF DEFINITION

```
name   OPDEF
lfp    synres  synop
       LOCAL   sym1,...,symn
         .
         .          (body of definition)
         .
name   ENDM
```

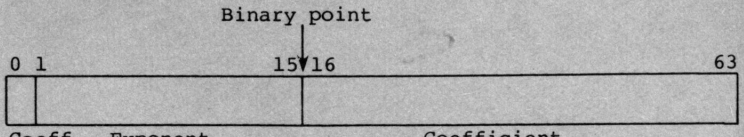
DATA FORMATS



Twos Complement Integer (24 bits)



Twos Complement Integer (64 bits)



Signed Magnitude Floating-point (64 bits)

FATAL ERRORS

C	Name, symbol, constant or data item error
D	Double defined symbol or duplicate parameter name
E	Definition or conditional sequence illegally nested
F	Too many entries
I	Instruction placement error
L	Location field error
N	Relocatable field error
O	Operand field error
P	Programmer error
R	Result field error
S	Syntax error
T	Type error
U	Undefined symbol or operation
V	Register expression or field width error
X	Expression error

WARNING ERRORS

W	Programmer warning error
W1	Location field symbol ignored
W2	Bad location symbol
W3	Expression element type error
W4	Possible symbolic machine instruction error
W5	Truncation error
W6	Location field symbol not defined
W7	Micro substitution error
W8	Address counter boundary error
Y1	External declaration error
Y2	Macro or opdef redefined

CONSTANT AND DATA NOTATION

Integer constant

$$\begin{Bmatrix}O'\\D'\\X'\end{Bmatrix}\ [integer]\ \begin{Bmatrix}S+n\\S-n\end{Bmatrix}$$

Floating-point constant

$$\begin{Bmatrix}O'\\D'\\X'\end{Bmatrix}\ \begin{bmatrix}integer.\\integer.fraction\\.fraction\end{bmatrix}\ \begin{Bmatrix}E+n\\En\\D+n\\Dn\end{Bmatrix}\ \begin{Bmatrix}S+n\\Sn\end{Bmatrix}$$

or

$$\begin{Bmatrix}O'\\D'\\X'\end{Bmatrix}\ [integer]\ \begin{Bmatrix}E+n\\En\\D+n\\Dn\end{Bmatrix}\ \begin{Bmatrix}S+n\\Sn\end{Bmatrix}$$

Character constant

$$\begin{Bmatrix}A\\C\\E\end{Bmatrix}\ ['character\ string']\ \begin{Bmatrix}H\\L\\R\\Z\end{Bmatrix}$$

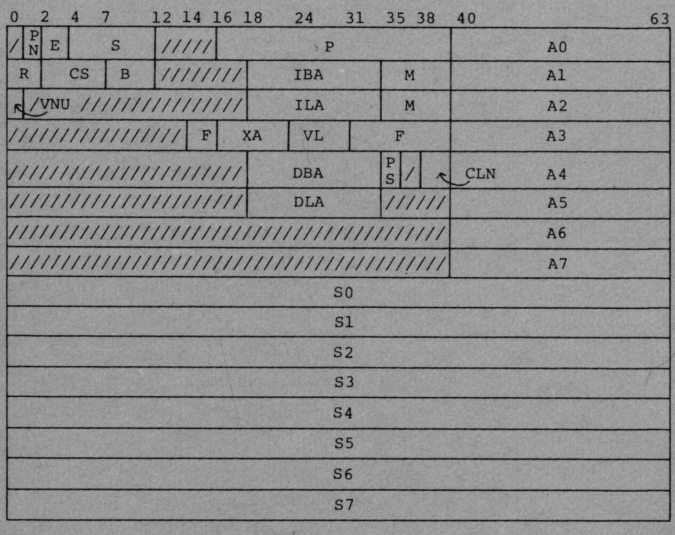
Character data

$$\begin{Bmatrix}A\\C\\E\end{Bmatrix}\ ['character\ string']\{count\}\begin{Bmatrix}H\\L\\R\\Z\end{Bmatrix}$$

Numeric data

Same as constant but may be preceded by $\begin{Bmatrix}+\\\#\end{Bmatrix}$

EXCHANGE PACKAGE



Registers

S Syndrome bits, 8 bits
P Program Address, 24 bits
CSB Read address for error, 10 bits
IBA Instruction Base Address, 17 bits
ILA Instruction Limit Address, 17 bits
XA Exchange Adress, 8 bits
VL Vector Length, 7 bits
DBA Data Base Address, 17 bits
PS Program State, 1 bit
CLN Cluster Number, 2 bits
DLA Data Limit Address, 17 bits

PN - Processor Number (bit 1 of n)

0 Executed in CPU 0
1 Executed in CPU 1

E - Error type (bits 2,3 of n)

10 Uncorrectable error
01 Correctable error

R - Read mode (bits 0,1 of n+1)

00 Scalar (includes memory references with A or S)
01 I/O
10 Vector, B, or T
11 Instruction fetch or exchange

VNU - Vector not used (bit 0 of n+2)

1 For instructions 076, 077, or 140-177

M - Modes

Word	Bit	Description
n+1	35	Waiting on semaphore
n+1	36	Floating-point error status
n+1	37	Bidirectional memory mode flag
n+1	38	External interrupts
n+1	39	Interrupt monitor mode
n+2	35	Operand error mode flag
n+2	36	Correctable memory error mode flag
n+2	37	Floating-point error mode flag
n+2	38	Uncorrectable memory error mode flag
n+2	39	Monitor mode

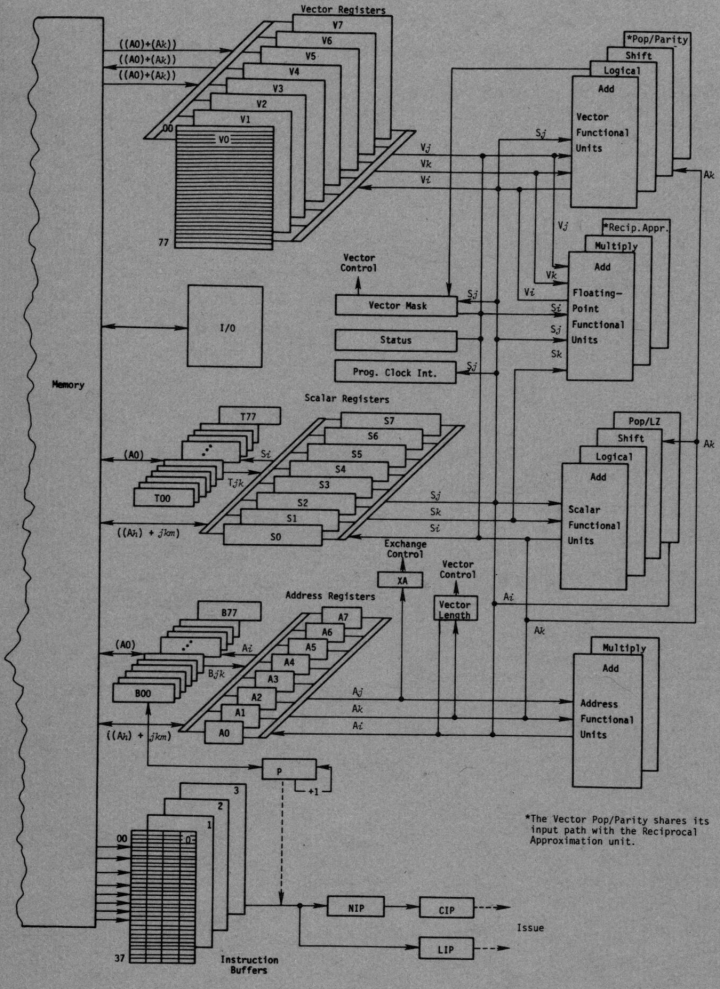
F - Flags (word n+3)

Bit	Description
14	Interrupt from Internal CPU
15	Deadlock
31	Programmable Clock Interrupt
32	MCU Interrupt
33	Floating-point Error
34	Operand Range
35	Program Range
36	Memory Error
37	I/O Interrupt
38	Error Exit
39	Normal Exit

FUNCTIONAL UNITS

Functional Unit	Unit Time (Clock Periods)	Instructions
Address integer add	2	030, 031
Address integer multiply	4	032
Scalar integer add	3	060, 061
Scalar logical	1	042-051
Scalar shift	2	052-055
	3	056, 057
Scalar pop/parity/ leading zero	4	026
	3	027
Vector integer add	3	154-157
Vector logical	2	140-147, 175
Vector shift	3	150, 151, 153
	4	152
Vector pop/parity	6	174*ij*1, 174*ij*2
Floating-point add	6	062, 063, 170-173
Floating-point multiply	7	064-067, 160-167
Floating-point reciprocal	14	070, 174*ij*0
Memory (scalar)	14	100-130

BLOCK DIAGRAM OF REGISTERS FOR A CPU



*The Vector Pop/Parity shares its input path with the Reciprocal Approximation unit.