CRAY

RESEARCH, INC.

CRAY-1 CAL REFERENCE CARD

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-1	CAL	UNIT	DESCRIPTION
000xxx	ERR	-	Error exit
†000ijk	ERR exp	-	Error exit
††0010jk	CA,Aj Ak	-	Set the channel (Aj) current address to (Ak) and begin the I/O sequence
††0011jk	CL,Aj Ak	-	Set the channel (Aj) limit address to (Ak)
††0012jx	CI,Aj	-	Clear channel (Aj) interrupt flag
††0013jx	XA Aj	-	Enter XA register with (Aj)
††0014j0	RT Sj	-	Enter RTC register with (Sj)
††§0014j4	PCI Sj	-	Enter II register with (Sj)
††§0014x5	CCI	-	Clear PCI request
††§0014x6	ECI	-	Enable PCI request
††§0014x7	DCI	-	Disable PCI request
0020xk	VL Ak	-	Transmit (Ak) to VL register
†0020x0	VL 1	-	Transmit 1 to VL register
0021xx	EFI	-	Enable interrupt on floating-point error
0022xx	DFI	-	Disable interrupt on floating-point error
003xjx	VM Sj	-	Transmit (Sj) to VM register
†003x0x	VM 0	-	Clear VM register
004xxx	EX	-	Normal exit
†004ijk	EX exp	-	Normal exit
005xjk	J Bjk	-	Jump to (Bjk)
006ijkm	J exp	-	Jump to exp
007ijkm	R exp	-	Return jump to exp; set B00 to P.
010ijkm	JAZ exp	-	Branch to exp if (A0)=0
011ijkm	JAN exp	-	Branch to exp if (A0)≠0
012ijkm	JAP exp	-	Branch to exp if (A0)≥0
013ijkm	JAM exp	-	Branch to exp if (A0)<0
014ijkm	JSZ exp	-	Branch to exp if (S0)=0
015ijkm	JSN exp	-	Branch to exp if (S0)≠0
016ijkm	JSP exp	-	Branch to exp if (S0)≥0
017ijkm	JSM exp	-	Branch to exp if (S0)<0
†††020ijkm	Ai exp	-	Transmit exp=jkm to Ai
†††021ijkm	Ai exp	-	Transmit exp=ones complement of jkm to Ai
†††022ijk	Ai exp	-	Transmit exp=jk to Ai
023ijx	Ai Sj	-	Transmit (Sj) to Ai
024ijk	Ai Bjk	-	Transmit (Bjk) to Ai
025ijk	Bjk Ai	-	Transmit (Ai) to Bjk
026ij0	Ai PSj	Pop/LZ	Population count of (Sj) to Ai
§§026ij1	Ai QSj	Pop/LZ	Population count parity of (Sj) to Ai
027ijx	Ai ZSj	Pop/LZ	Leading zero count of (Sj) to Ai
030ijk	Ai Aj+Ak	A Int Add	Integer sum of (Aj) and (Ak) to Ai
†030i0k	Ai Ak	A Int Add	Transmit (Ak) to Ai
†030ij0	Ai Aj+1	A Int Add	Integer sum of (Aj) and 1 to Ai
031ijk	Ai Aj-Ak	A Int Add	Integer difference of (Aj) less (Ak) to Ai
† †††031i00	Ai -1	A Int Add	Transmit -1 to Ai
†031i0k	Ai -Ak	A Int Add	Transmit the negative of (Ak) to Ai
†031ij0	Ai Aj-1	A Int Add	Integer difference of (Aj) less 1 to Ai
032ijk	Ai Aj*Ak	A Int Mult	Integer product of (Aj) and (Ak) to Ai
033i0x	Ai CI	-	Channel number to Ai (j=0)
033ij0	Ai CA,Aj	-	Address of channel (Aj) to Ai (j≠0; k=0)
033ij1	Ai CE,Aj	-	Error flag of channel (Aj) to Ai (j≠0; k=1)
034ijk	Bjk,Ai ,A0	Memory	Read (Ai) words to B register jk from (A0)
†034ijk	Bjk,Ai 0,A0	Memory	Read (Ai) words to B register jk from (A0)
035ijk	,A0 Bjk,Ai	Memory	Store (Ai) words at B register jk to (A0)
†035ijk	0,A0 Bjk,Ai	Memory	Store (Ai) words at B register jk to (A0)
036ijk	Tjk,Ai ,A0	Memory	Read (Ai) words to T register jk from (A0)
†036ijk	Tjk,Ai 0,A0	Memory	Read (Ai) words to T register jk from (A0)
037ijk	,A0 Tjk,Ai	Memory	Store (Ai) words at T register jk to (A0)
†037ijk	0,A0 Tjk,Ai	Memory	Store (Ai) words at T register jk to (A0)

CRAY-1	CAL	UNIT	DESCRIPTION
040ijkm	Si exp	-	Transmit jkm to Si
041ijkm	Si exp	-	Transmit exp=ones complement of jkm to Si
042ijk	Si <exp	S Logical	Form ones mask exp bits in Si from the right; jk field gets 64-exp.
†042ijk	Si #>exp	S Logical	Form zeros mask exp bits in Si from the left; jk field gets exp.
†042i77	Si 1	S Logical	Enter 1 into Si
†042i00	Si -1	S Logical	Enter -1 into Si
043ijk	Si >exp	S Logical	Form ones mask exp bits in Si from the left; jk field gets exp.
†043ijk	Si #<exp	S Logical	Form zeros mask exp bits in Si from the right; jk field gets 64-exp.
†043i00	Si 0	S Logical	Clear Si
044ijk	Si Sj&Sk	S Logical	Logical product of (Sj) and (Sk) to Si
†044ij0	Si Sj&SB	S Logical	Sign bit of (Sj) to Si
†044ij0	Si SB&Sj	S Logical	Sign bit of (Sj) to Si (j≠0)
045ijk	Si #Sk&Sj	S Logical	Logical product of (Sj) and ones complement of (Sk) to Si
†045ij0	Si #SB&Sj	S Logical	(Sj) with sign bit cleared to Si
046ijk	Si Sj\Sk	S Logical	Logical difference of (Sj) and (Sk) to Si
†046ij0	Si Sj\SB	S Logical	Toggle sign bit of Sj, then enter into Si
†046ij0	Si SB\Sj	S Logical	Toggle sign bit of Sj, then enter into Si (j≠0)
047ijk	Si #Sj\Sk	S Logical	Logical equivalence of (Sk) and (Sj) to Si
†047i0k	Si #Sk	S Logical	Transmit ones complement of (Sk) to Si
†047ij0	Si #Sj\SB	S Logical	Logical equivalence of (Sj) and sign bit to Si
†047ij0	Si #SB\Sj	S Logical	Logical equivalence of (Sj) and sign bit to Si (j≠0)
†047i00	Si #SB	S Logical	Enter ones complement of sign bit into Si
050ijk	Si Sj!Si&Sk	S Logical	Logical product of (Si) and (Sk) complement ORed with logical product of (Sj) and (Sk) to Si
†050ij0	Si Sj!Si&SB	S Logical	Scalar merge of (Si) and sign bit of (Sj) to Si
051ijk	Si Sj!Sk	S Logical	Logical sum of (Sj) and (Sk) to Si
†051i0k	Si Sk	S Logical	Transmit (Sk) to Si
†051ij0	Si Sj!SB	S Logical	Logical sum of (Sj) and sign bit to Si
†051ij0	Si SB!Sj	S Logical	Logical sum of (Sj) and sign bit to Si (j≠0)
†051i00	Si SB	S Logical	Enter sign bit into Si
052ijk	S0 Si<exp	S Shift	Shift (Si) left exp=jk places to S0
053ijk	S0 Si>exp	S Shift	Shift (Si) right exp=64-jk places to S0
054ijk	Si Si<exp	S Shift	Shift (Si) left exp=jk places
055ijk	Si Si>exp	S Shift	Shift (Si) right exp=64-jk places
056ijk	Si Si,Sj<Ak	S Shift	Shift (Si and Sj) left (Ak) places to Si
†056ij0	Si Si,Sj<1	S Shift	Shift (Si and Sj) left one place to Si
†056i0k	Si Si<Ak	S Shift	Shift (Si) left (Ak) places to Si
057ijk	Si Sj,Si>Ak	S Shift	Shift (Sj and Si) right (Ak) places to Si
†057ij0	Si Sj,Si>1	S Shift	Shift (Sj and Si) right one place to Si
†057i0k	Si Si>Ak	S Shift	Shift (Si) right (Ak) places to Si
060ijk	Si Sj+Sk	S Int Add	Integer sum of (Sj) and (Sk) to Si
061ijk	Si Sj-Sk	S Int Add	Integer difference of (Sj) and (Sk) to Si
†061i0k	Si -Sk	S Int Add	Transmit negative of (Sk) to Si
062ijk	Si Sj+FSk	Fp Add	Floating-point sum of (Sj) and (Sk) to Si
†062i0k	Si +FSk	Fp Add	Normalize (Sk) to Si
063ijk	Si Sj-FSk	Fp Add	Floating-point difference of (Sj) and (Sk) to Si
†063i0k	Si -FSk	Fp Add	Transmit normalized negative of (Sk) to Si
064ijk	Si Sj*FSk	Fp Mult	Floating-point product of (Sj) and (Sk) to Si
065ijk	Si Sj*HSk	Fp Mult	Half-precision rounded floating-point product of (Sj) and (Sk) to Si
066ijk	Si Sj*RSk	Fp Mult	Full-precision rounded floating-point product of (Sj) and (Sk) to Si
067ijk	Si Sj*ISk	Fp Mult	2-floating-point product of (Sj) and (Sk) to Si
070ijx	Si /HSj	Fp Rcpl	Floating-point reciprocal approximation of (Sj) to Si
071i0k	Si Ak	-	Transmit (Ak) to Si with no sign extension

CRAY-1	CAL	UNIT	DESCRIPTION
071i1k	Si +Ak	-	Transmit (Ak) to Si with sign extension
071i2k	Si +FAk	-	Transmit (Ak) to Si as unnormalized floating-point number
071i3x	Si 0.6	-	Transmit constant 0.75*2**48 to Si
071i4x	Si 0.4	-	Transmit constant 0.5 to Si
071i5x	Si 1.	-	Transmit constant 1.0 to Si
071i6x	Si 2.	-	Transmit constant 2.0 to Si
071i7x	Si 4.	-	Transmit constant 4.0 to Si
072ixx	Si RT	-	Transmit (RTC) to Si
073ixx	Si VM	-	Transmit (VM) to Si
074ijk	Si Tjk	-	Transmit (Tjk) to Si
075ijk	Tjk Si	-	Transmit (Si) to Tjk
076ijk	Si Vj,Ak	-	Transmit (Vj, element (Ak)) to Si
077ijk	Vi,Ak Sj	-	Transmit (Sj) to Vi element (Ak)
†077i0k	Vi,Ak 0	-	Clear Vi element (Ak)
10hijkm	Ai exp,Ah	Memory	Read from ((Ah)+exp) to Ai (A0=0)
†100ijkm	Ai exp,0	Memory	Read from (exp) to Ai
†100ijkm	Ai exp,	Memory	Read from (exp) to Ai
†10hi000	Ai ,Ah	Memory	Read from (Ah) to Ai
11hijkm	exp,Ah Ai	Memory	Store (Ai) to (Ah)+exp (A0=0)
†110ijkm	exp,0 Ai	Memory	Store (Ai) to exp
†110ijkm	exp, Ai	Memory	Store (Ai) to exp
†11hi000	,Ah Ai	Memory	Store (Ai) to (Ah)
12hijkm	Si exp,Ah	Memory	Read from ((Ah)+exp) to Si (A0=0)
†120ijkm	Si exp,0	Memory	Read from (exp) to Si
†120ijkm	Si exp,	Memory	Read from (exp) to Si
†12hi000	Si ,Ah	Memory	Read from (Ah) to Si
13hijkm	exp,Ah Si	Memory	Store (Si) to (Ah)+exp (A0=0)
†130ijkm	exp,0 Si	Memory	Store (Si) to exp
†130ijkm	exp, Si	Memory	Store (Si) to exp
†13hi000	,Ah Si	Memory	Store (Si) to (Ah)
140ijk	Vi Sj&Vk	V Logical	Logical products of (Sj) and (Vk) to Vi
141ijk	Vi Vj&Vk	V Logical	Logical products of (Vj) and (Vk) to Vi
142ijk	Vi Sj!Vk	V Logical	Logical sums of (Sj) and (Vk) to Vi
†142i0k	Vi Vk	V Logical	Transmit (Vk) to Vi
143ijk	Vi Vj!Vk	V Logical	Logical sums of (Vj) and (Vk) to Vi
144ijk	Vi Sj\Vk	V Logical	Logical differences of (Sj) and (Vk) to Vi
145ijk	Vi Vj\Vk	V Logical	Logical differences of (Vj) and (Vk) to Vi
†145iii	Vi 0	V Logical	Clear Vi
146ijk	Vi Sj!Vk&VM	V Logical	Transmit (Sj) if VM bit=1; (Vk) if VM bit=0 to Vi.
†146i0k	Vi #VM&Vk	V Logical	Vector merge of (Vk) and 0 to Vi
147ijk	Vi Vj!Vk&VM	V Logical	Transmit (Vj) if VM bit=1; (Vk) if VM bit=0 to Vi.
150ijk	Vi Vj<Ak	V Shift	Shift (Vj) left (Ak) places to Vi
†150ij0	Vi Vj<1	V Shift	Shift (Vj) left one place to Vi
151ijk	Vi Vj>Ak	V Shift	Shift (Vj) right (Ak) places to Vi
†151ij0	Vi Vj>1	V Shift	Shift (Vj) right one place to Vi
152ijk	Vi Vj,Vj<Ak	V Shift	Double shift (Vj) left (Ak) places to Vi
†152ij0	Vi Vj,Vj<1	V Shift	Double shift (Vj) left one place to Vi
153ijk	Vi Vj,Vj>Ak	V Shift	Double shift (Vj) right (Ak) places to Vi
†153ij0	Vi Vj,Vj>1	V Shift	Double shift (Vj) right one place to Vi
154ijk	Vi Sj+Vk	V Int Add	Integer sums of (Sj) and (Vk) to Vi
155ijk	Vi Vj+Vk	V Int Add	Integer sums of (Vj) and (Vk) to Vi
156ijk	Vi Sj-Vk	V Int Add	Integer differences of (Sj) and (Vk) to Vi
†156i0k	Vi -Vk	V Int Add	Transmit negative of (Vk) to Vi
157ijk	Vi Vj-Vk	V Int Add	Integer differences of (Vj) and (Vk) to Vi
160ijk	Vi Sj*FVk	Fp Mult	Floating-point products of (Sj) and (Vk) to Vi

CRAY-1	CAL	UNIT	DESCRIPTION
161ijk	Vi Vj*FVk	Fp Mult	Floating-point products of (Vj) and (Vk) to Vi
162ijk	Vi Sj*HVk	Fp Mult	Half-precision rounded floating-point products of (Sj) and (Vk) to Vi
163ijk	Vi Vj*HVk	Fp Mult	Half-precision rounded floating-point products of (Vj) and (Vk) to Vi
164ijk	Vi Sj*RVk	Fp Mult	Rounded floating-point products of (Sj) and (Vk) to Vi
165ijk	Vi Vj*RVk	Fp Mult	Rounded floating-point products of (Vj) and (Vk) to Vi
166ijk	Vi Sj*IVk	Fp Mult	2-floating-point products of (Sj) and (Vk) to Vi
167ijk	Vi Vj*IVk	Fp Mult	2-floating-point products of (Vj) and (Vk) to Vi
170ijk	Vi Sj+FVk	Fp Add	Floating-point sums of (Sj) and (Vk) to Vi
†170i0k	Vi +FVk	Fp Add	Normalize (Vk) to Vi
171ijk	Vi Vj+FVk	Fp Add	Floating-point sums of (Vj) and (Vk) to Vi
172ijk	Vi Sj-FVk	Fp Add	Floating-point differences of (Sj) and (Vk) to Vi
†172i0k	Vi -FVk	Fp Add	Transmit normalized negatives of (Vk) to Vi
173ijk	Vi Vj-FVk	Fp Add	Floating-point differences of (Vj) and (Vk) to Vi
174ij0	Vi /HVj	Fp Rcpl	Floating-point reciprocal approximations of (Vj) to Vi
$$174ij1	Vi PVj	V Pop	Population counts of (Vj) to Vi
$$174ij2	Vi QVj	V Pop	Population count parities of (Vj) to Vi
175xj0	VM Vj,Z	V Logical	VM=1 where (Vj)=0
175xj1	VM Vj,N	V Logical	VM=1 where (Vj)≠0
175xj2	VM Vj,P	V Logical	VM=1 where (Vj) positive
175xj3	VM Vj,M	V Logical	VM=1 where (Vj) negative
176ixk	Vi ,A0,Ak	Memory	Read (VL) words to Vi from (A0) incremented by (Ak)
†176ix0	Vi ,A0,1	Memory	Read (VL) words to Vi from (A0) incremented by 1
177xjk	,A0,Ak Vj	Memory	Store (VL) words from Vj to (A0) incremented by (Ak)
†177xj0	,A0,1 Vj	Memory	Store (VL) words from Vj to (A0) incremented by 1

† Special syntax form

†† Privileged to monitor mode

††† Generated depending on value of *exp*

$ Programmable clock (optional on CRAY-1 Models A and B)

$$ Vector Population Count (optional on CRAY-1 Models A and B)

x Field not used by hardware; assembler generates zero in this position.

REGISTER	VALUE
Ah, h=0	0
Ai, i=0	(A0)
Aj, j=0	0
Ak, k=0	1
Si, i=0	(S0)
Sj, j=0	0
Sk, 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	CHAR	ASCII	ASCII CARD CODE
NUL	000	12-0-9-8-1	@	100	8-4
SOH	001	12-9-1	A	101	12-1
STX	002	12-9-2	B	102	12-2
ETX	003	12-9-3	C	103	12-3
EOT	004	9-7	D	104	12-4
ENQ	005	0-9-8-5	E	105	12-5
ACK	006	0-9-8-6	F	106	12-6
BEL	007	0-9-8-7	G	107	12-7
BS	010	11-9-6	H	110	12-8
HT	011	12-9-5	I	111	12-9
LF	012	0-9-5	J	112	11-1
VT	013	12-9-8-3	K	113	11-2
FF	014	12-9-8-4	L	114	11-3
CR	015	12-9-8-5	M	115	11-4
SO	016	12-9-8-6	N	116	11-5
SI	017	12-9-8-7	O	117	11-6
DLE	020	12-11-9-8-1	P	120	11-7
DC1	021	11-9-1	Q	121	11-8
DC2	022	11-9-2	R	122	11-9
DC3	023	11-9-3	S	123	0-2
DC4	024	9-8-4	T	124	0-3
NAK	025	9-8-5	U	125	0-4
SYN	026	9-2	V	126	0-5
ETB	027	0-9-6	W	127	0-6
CAN	030	11-9-8	X	130	0-7
EM	031	11-9-8-1	Y	131	0-8
SUB	032	9-8-7	Z	132	0-9
ESC	033	0-9-7	[	133	12-8-2
FS	034	11-9-8-4	\	134	0-8-2
GS	035	11-9-8-5	]	135	11-8-2
RS	036	11-9-8-6	^	136	11-8-7
US	037	11-9-8-7	_	137	0-8-5
Space	040	None	`	140	8-1
!	041	12-8-7	a	141	12-0-1
"	042	8-7	b	142	12-0-2
#	043	8-3	c	143	12-0-3
$	044	11-8-3	d	144	12-0-4
%	045	0-8-4	e	145	12-0-5
&	046	12	f	146	12-0-6
'	047	8-5	g	147	12-0-7
(	050	12-8-5	h	150	12-0-8
)	051	11-8-5	i	151	12-0-9
*	052	11-8-4	j	152	12-11-1
+	053	12-8-6	k	153	12-11-2
,	054	0-8-3	l	154	12-11-3
-	055	11	m	155	12-11-4
.	056	12-8-3	n	156	12-11-5
/	057	0-1	o	157	12-11-6
0	060	0	p	160	12-11-7
1	061	1	q	161	12-11-8
2	062	2	r	162	12-11-9
3	063	3	s	163	11-0-2
4	064	4	t	164	11-0-3
5	065	5	u	165	11-0-4
6	066	6	v	166	11-0-5
7	067	7	w	167	11-0-6
8	070	8	x	170	11-0-7
9	071	9	y	171	11-0-8
:	072	8-2	z	172	11-0-9
;	073	11-8-6	{	173	12-0
<	074	12-8-4	¦	174	12-11
=	075	8-6	}	175	11-0
>	076	0-8-6	~	176	11-0-1
?	077	0-8-7	DEL	177	12-9-7

PSEUDO INSTRUCTIONS

PROGRAM CONTROL

IDENT	*name*
END	
ABS	
COMMENT	*'string'*

MICROS

name	MICRO	$'string', exp_1, exp_2$
name	MICRO	$'string', exp_1$
name	MICRO	*'string'*
name	OCTMIC	*exp, count*
name	DECMIC	*exp, count*

DATA DEFINITION

symbol	CON	$exp_1, exp_2, \ldots, exp_n$
symbol	BSSZ	*exp*
symbol	DATA	$data_1, data_2, \ldots, data_n$
symbol	VWD	$n_1/exp_1, n_2/exp_2, \ldots, n_m/exp_m$
	REP	*ct, swa, inc, bsz*

LISTING CONTROL

name	LIST	$op_1, op_2, \ldots, op_n$
	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	exp_1, op, exp_2

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

LOADER LINKAGE

ENTRY	$symbol_1, symbol_2, \ldots, symbol_n$
EXT	$sym_1, sym_2, \ldots, sym_n$
MODULE	*modtype*
START	*symbol*

CODE DUPLICATION

dupname	DUP	*times*
	DUP	*times, count*
dupname	ECHO	$e_1=(list_1), e_2=(list_2), \ldots, e_n=(list_n)$
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	exp_1, op, exp_2
	IFE	$exp_1, op, exp_2, count$
ifname	IFC	$'string_1', op, 'string_2'$
	IFC	$'string_1', op, 'string_2', count$
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*	$p_1, p_2, \ldots, p_n, e_1=d_1, e_2=d_2, \ldots, e_m=d_m$
	LOCAL	$sym_1, \ldots, sym_r$
	.	
	.	(body of definition)
	.	
name	ENDM	
$name_1$	OPSYN	$name_2$

OPDEF DEFINITION

name	OPDEF	
lfp	*synres*	*synop*
	LOCAL	$sym_1, \ldots, sym_r$
	.	
	.	(body of definition)
	.	
name	ENDM	

FUNCTIONAL UNITS

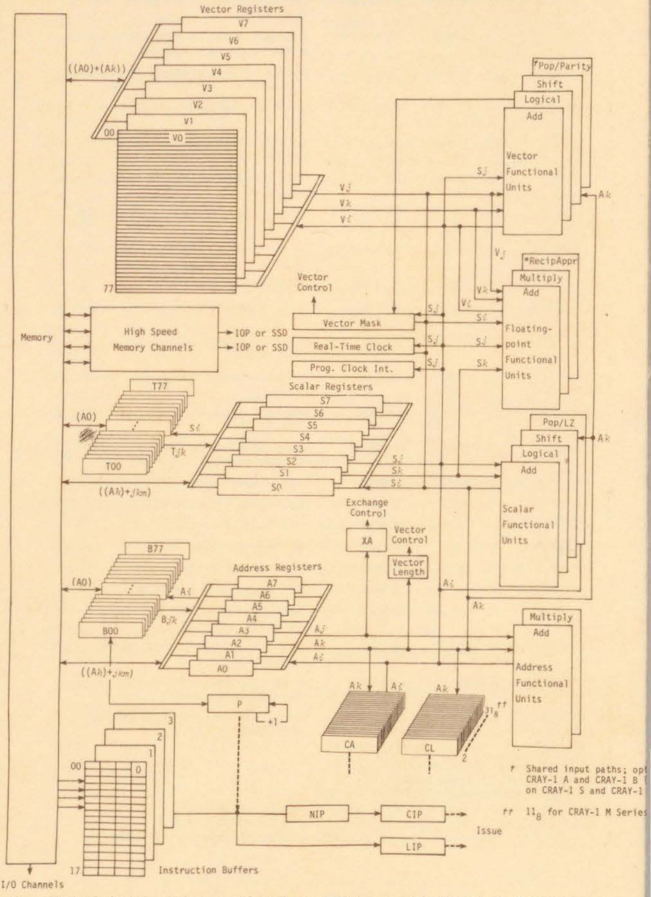
Functional Unit	Unit Time (Clock Periods)	Instructions
Address integer add	2	030, 031
Address integer multiply	6	032
Scalar integer add	3	060, 061
Scalar logical	1	042-051
Scalar shift	2	052-055
	3	056, 057
Scalar pop/parity[†]/	4	026
leading zero	3	027
Vector integer add	3	154-157
Vector logical	2	140-147, 175
Vector shift	4	150-153
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)	11[††]	100-130
Memory (vector)	7[††,†††]	176, 177

[†] Only with vector population

[††] For Serial 1: scalar 10, vector 6

[†††] For CRAY-1 M Series: 8, 9, or 10

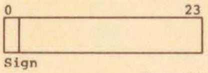
BLOCK DIAGRAM OF REGISTERS



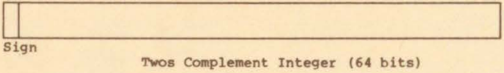
† Shared input paths; optional on CRAY-1 A and CRAY-1 B (standard on CRAY-1 S and CRAY-1 M Series)

†† 11_8 for CRAY-1 M Series

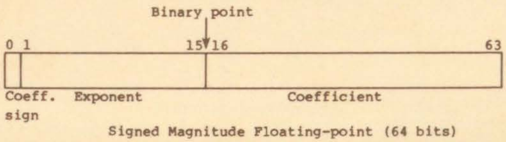
DATA FORMATS



Twos Complement Integer (24 bits)

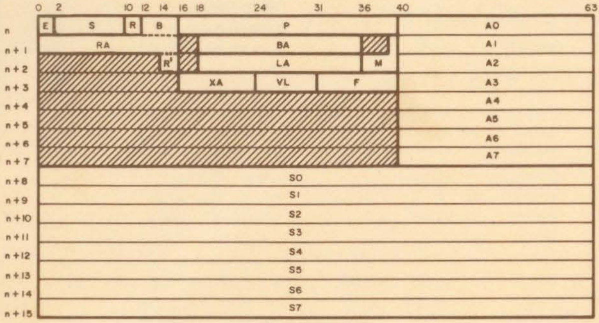


Twos Complement Integer (64 bits)



Signed Magnitude Floating-point (64 bits)

EXCHANGE PACKAGE



<u>Registers</u>

S	Syndrome bits
R'RAB	Read address for error (where B is bank)
P	Program Address, 24 bits
BA	Base Address, 18 bits
LA	Limit Address, 18 bits
XA	Exchange Address, 8 bits
VL	Vector Length, 7 bits

<u>E - Error type (bits 0,1 of n)</u>

10	Uncorrectable memory
01	Correctable memory

<u>R - Read mode (bits 10,11 of n)</u>

onal on
tandard
Series)

00	Scalar
01	I/O
10	Vector
11	Fetch

Word Offset	Bit	M - Modes
n+1	39	Interrupt monitor mode[†]
n+2	36	Interrupt on correctable memory error
n+2	37	Interrupt on floating-point error
n+2	38	Interrupt on uncorrectable memory error
n+2	39	Monitor mode

Word Offset	Bit	F - Flags
n+3	31	Programmable Clock Interrupt (PCI)[††]
n+3	32	MCU interrupt
n+3	33	Floating-point error
n+3	34	Operand range error
n+3	35	Program range error
n+3	36	Memory error
n+3	37	I/O interrupt
n+3	38	Error exit
n+3	39	Normal exit

[†] Supports Monitor Mode Interrupt option

[††] Supports Programmable Clock option (optional on CRAY-1 Models A and B; standard on CRAY-1 S Series and CRAY-1 M Series computers)

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

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

Floating-point constant

$$\begin{Bmatrix} \texttt{O'} \\ \texttt{D'} \\ \texttt{X'} \end{Bmatrix} \begin{bmatrix} \textit{integer.} \\ \textit{integer.fraction} \\ \textit{.fraction} \end{bmatrix} \begin{Bmatrix} \texttt{E}\pm n \\ \texttt{E}n \\ \texttt{D}\pm n \\ \texttt{D}n \end{Bmatrix} \begin{Bmatrix} \texttt{S}\pm n \\ \texttt{S}n \end{Bmatrix}$$

or

$$\begin{Bmatrix} \texttt{O'} \\ \texttt{D'} \\ \texttt{X'} \end{Bmatrix} [\textit{integer}] \begin{Bmatrix} \texttt{E}\pm n \\ \texttt{E}n \\ \texttt{D}\pm n \\ \texttt{D}n \end{Bmatrix} \begin{Bmatrix} \texttt{S}\pm n \\ \texttt{S}n \end{Bmatrix}$$

Character constant

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

Character data

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

Numeric data

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