Cray SV1™ Series Hardware Reference Card

108-0213-001

Table of CPU Instructions:

[a] These instructions are privileged to monitor mode.

[b] Special CAL syntax.

[c] These instructions are not supported by CAL Version 2.

[d] Bit 2 of the *i* field is equal to 0.

[e] These instructions are generated depending on the value of the exponent.

Machine Code	CAL Syntax	Description
000000	ERR	Error Exit
0010*jk*[a]	CA,A*j* A*k*	Set the CA register for the channel indicated by (A*j*) to (A*k*) and activate the channel
001000	PASS	This is a no-operation instruction
0011*jk*[a]	CL,A*j* A*k*	Set the CL register for the channel indicated by (A*j*) to (A*k*) address
0012*j*0[a]	CI,A*j*	Clear the interrupt flag and error flag for the channel indicated by (A*j*); clear device master clear (output channels only)
0012*j*1[a]	MC,A*j*	Clear the interrupt flag and error flag for the channel indicated by (A*j*); set device master clear (output channels only); clear device ready-held (input channels only)
0013*j*0[a]	XA A*j*	Transfer (A*j*) to the XA register
0014*j*0[a]	RT S*j*	Load the RTC register with (S*j*)
0014*j*1[a]	SIPI A*j*	Send an interprocessor interrupt request to CPU (A*j*)
001401[a]	SIPI	Send an interprocessor interrupt request to CPU 0
001402[a]	CIPI	Clear the interprocessor interrupt
0014*j*3[a]	CLN A*j*	Load the CLN register with (A*j*)
0014*j*4[a]	PCI S*j*	Load the II register with (S*j*)
001405[a]	CCI	Clear the programmable clock interrupt request
001406[a]	ECI	Enable the programmable clock interrupt request
001407[a]	DCI	Disable the programmable clock interrupt request
0015*j*0[a,c]		Select performance monitor
001511[a,c]		No operation
001521[a,c]		Disable I/O error correction on port D

Machine Code	CAL Syntax	Description
001541[a,c]		Enable replacement of checkbyte with data for write to memory and the replacement of data with checkbytes for read from memory on ports A, B, and D
001551[a,c]		Enable replacement of checkbyte with V*k* data bits on port A or B write to memory during execution of the 177*ijk* instruction
0016*j*1[b]	IVC	Send invalidate cache request to CPU (A*j*)
002000	VL 1	Transmit 1 to VL register
00200*k*	VL A*k*	Transmit (A*k*) to VL register
002100	EFI	Enable interrupt on floating-point error
002200	DFI	Disable interrupt on floating-point error
002210	CBL	Clear the bit matrix loaded (BML) bit in the Status Register and Exchange Package. If BMM instructions are not enabled in the CPU, 002210 is a no-op.
002300	ERI	Enable interrupt on operand range error
002400	DRI	Disable interrupt on operand range error
002500	DBM	Disable bidirectional memory transfers
002600	EBM	Enable bidirectional memory transfers
002700	CMR	Complete memory references
0030*j*0	VM S*j*	Transfer (S*j*) to VM register
003000[b]	VM 0	Clear VM register
0034*jk*	SM*jk* 1,TS	Invalidate cache and test and set semaphore *jk*, $0 \le jk \le 31_{10}$
0036*jk*	SM*jk* 0	Clear semaphore *jk*, $0 \le jk \le 31_{10}$
0037*jk*	SM*jk* 1	Set semaphore *jk*, $0 \le jk \le 31_{10}$
004000	EX	Normal exit from the operating system
0050*jk*	J B*jk*	Jump to (B*jk*)
006*ijkm*	J *exp*	Jump to *exp*
007*ijkm*	R *exp*	Return jump to *exp* and set register B000 to (P) + 2
010*ijkm*[d]	JAZ *exp*	Jump to *exp* if (A0) = 0 (i_2 = 0)
011*ijkm*[d]	JAN *exp*	Jump to *exp* if (A0) ≠ 0 (i_2 = 0)
012*ijkm*[d]	JAP *exp*	Jump to *exp* if (A0) positive; (A0) ≥ 0 (i_2 = 0)
013*ijkm*[d]	JAM *exp*	Jump to *exp* if (A0) negative (i_2 = 0)
014*ijkm*[d]	JSZ *exp*	Jump to *exp* if (S0) = 0 (i_2 = 0)
015*ijkm*[d]	JSN *exp*	Jump to *exp* if (S0) ≠ 0 (i_2 = 0)
016*ijkm*[d]	JSP *exp*	Jump to *exp* if (S0) positive (i_2 = 0)
017*ijkm*[d]	JSM *exp*	Jump to *exp* if (S0) negative (i_2 = 0)
020*i*00*mn*[e] or 021*i*00*mn* or 022*ijk*[e]	A*i* *exp*	Transmit *exp* = *mn* into A*i* (020) or transmit ones complement of *exp* = *mn* into A*i* (021) or transmit *exp* = *jk* into A*i* (022)
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*

Machine Code	CAL Syntax	Description
026*ij*0	A*i* PS*j*	Transmit the population count of (S*j*) to A*i*
026*ij*1	A*i* QS*j*	Transmit the population count parity of (S*j*) to A*i*
026*ij*7	A*i* SB*j*	Transmit (SB*j*) to A*i*
027*ij*0	A*i* ZS*j*	Transmit 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*	Transmit the integer sum of (A*j*) and (A*k*) to A*i*
030*i*0*k*[b]	A*i* A*k*	Transmit (A*k*) to A*i*
030*ij*0[b]	A*i* A*j*+1	Transmit the integer sum of (A*j*) and 1 to A*i*
031*ijk*	A*i* A*j*-A*k*	Transmit the integer difference of (A*j*) and (A*k*) to A*i*
031*i*00[b]	A*i* -1	Transmit -1 to A*i*
031*i*0*k*[b]	A*i* -A*k*	Transmit the negative of (A*k*) to A*i*
031*ij*0[b]	A*i* A*j*-1	Transmit the integer difference of (A*j*) and 1 to A*i*
032*ijk*	A*i* A*j**A*k*	Transmit the integer product of (A*j*) and (A*k*) to A*i*
033*i*00	A*i* CI	Transmit the channel number of the highest priority interrupt request to A*i* (*j*=0)
033*ij*0	A*i* CA,A*j*	Transmit the current address of the channel (A*j*) to A*i* (*j* ≠ 0, *k* = 0)
033*ij*1	A*i* CE,A*j*	Transmit the error flag of the channel (A*j*) to A*i* (*j* ≠ 0, *k* = 1)
034*ijk*	B*jk*,A*i* ,A0	Load (A*i*) words from memory starting at address (A0) to B registers starting at register *jk*
034*ijk*[b]	B*jk*,A*i* 0,A0	Load (A*i*) words from memory starting at address (A0) to B registers starting at register *jk*
035*ijk*	,A0 B*jk*,A*i*	Store (A*i*) words from B registers starting at register *jk* to memory starting at address (A0)
035*ijk*[b]	0,A0 B*jk*,A*i*	Store (A*i*) words from B registers starting at register *jk* to memory starting at address (A0)
036*ijk*	T*jk*,A*i* ,A0	Load (A*i*) words from memory starting at address (A0) to T registers starting at register *jk*
036*ijk*[b]	T*jk*,A*i* 0,A0	Load (A*i*) words from memory starting at address (A0) to T registers starting at register *jk*
037*ijk*	,A0 T*jk*,A*i*	Store (A*i*) words from T registers starting at register *jk* to memory starting at address (A0)
037*ijk*[b]	0,A0 T*jk*,A*i*	Store (A*i*) words from T registers starting at register *jk* to memory starting at address (A0)
040i00*mn* or 041i00*mn*	S*i* *exp*	Transmit *exp* into S*i* (040) or transmit ones complement of *exp* into S*i* (041)
042*ijk*	S*i* *<exp*	Form ones mask in S*i* *exp* bits from right; *exp* = 100_8 - *jk* bits
042*ijk*[b]	S*i* *#>exp*	Form zeroes mask in S*i* *exp* bits from left; *exp* = *jk* bits

Machine Code	CAL Syntax	Description
042*i*77[b]	S*i* 1	Enter 1 into S*i* register
042*i*00[b]	S*i* -1	Enter -1 into S*i* register
043*ijk*	S*i* *>exp*	Form ones mask in S*i* *exp* bits from left; *exp* = *jk* bits
043*ijk*[b]	S*i* *#<exp*	Form zeroes mask in S*i* *exp* bits from righ *exp* = 100_8 - *jk* bits
043*i*00[b]	S*i* 0	Clear the S*i* register
044*ijk*	S*i* S*j*&S*k*	Transmit the logical product of (S*j*) and (S*k* to S*i*
044*ij*0[b]	S*i* S*j*&SB	Transmit the sign bit of (S*j*) to S*i*
044*ij*0[b]	S*i* SB&S*j*	Transmit the sign bit of (S*j*) to S*i* (*j* ≠ 0)
045*ijk*	S*i* #S*k*&S*j*	Transmit the logical product of (S*j*) and ones complement of (S*k*) to S*i*
045*ij*0[b]	S*i* #SB&S*k*	Transmit (S*j*) with sign bit cleared to S*i*
046*ijk*	S*i* S*j*\S*k*	Transmit the logical difference of (S*j*) and (S*k*) to S*i*
046*ij*0[b]	S*i* S*j*\SB	Toggle the sign bit of (S*j*); then enter into S
046*ij*0[b]	S*i* SB\S*j*	Toggle the sign bit of (S*j*); then enter into S if *j* ≠ 0
047*ijk*	S*i* #S*j*\S*k*	Transmit the logical equivalent of (S*k*) and (S*j*) to S*i*
047*i*0*k*[b]	S*i* #S*k*	Transmit the ones complement of (S*k*) to S
047*ij*0[b]	S*i* #S*j*\SB	Transmit the logical equivalence of (S*j*) an sign bit to S*i*
047*ij*0[b]	S*i* #SB\S*j*	Transmit the logical equivalence of (S*j*) an sign bit to S*i* (*j* ≠ 0)
047*i*00[b]	S*i* #SB	Transmit the ones complement of the sign bit into S*i*
050*ijk*	S*i* S*j*!S*i*&S*k*	Transmit the logical product of (S*j*) and (S*k* complement ORed with the logical produc of (S*j*) and (S*k*) to (S*i*)
050*ij*0[b]	S*i* S*j*!S*i*&SB	Transmit the scalar merge of (S*i*) and sign bit of (S*j*) to S*i*
051*ijk*	S*i* S*j*!S*k*	Transmit the logical sum of (S*j*) and (S*k*) to S*i*
051*i*0*k*[b]	S*i* S*k*	Transmit (S*k*) to S*i*
051*ij*0[b]	S*i* S*j*!SB	Transmit the logical sum of (S*j*) and sign bi to S*i*
051*ij*0[b]	S*i* SB!S*j*	Transmit the logical sum of (S*j*) and sign bi to S*i* (*j* ≠ 0)
051*i*00[b]	S*i* SB	Transmit the sign bit into S*i*
052*ijk*	S0 S*i*<*exp*	Shift (S*i*) left *exp* places to S0; *exp* = *jk*
053*ijk*	S0 S*i*>*exp*	Shift (S*i*) right *exp* places to S0; *exp* = 100_8-*jk*
054*ijk*	S*i* S*i*<*exp*	Shift (S*i*) left *exp* places to S*i*; *exp* = *jk*
055*ijk*	S*i* S*i*>*exp*	Shift (S*i*) right *exp* places to S*i*; *exp*=100_8-*jk*
056*ijk*	S*i* S*j*,S*i*> A*k*	Shift (S*j*) and (S*i*) right (A*k*) places to S*i*
056*ij*0[b]	S*i* S*i*,S*j*< *1*	Shift (S*i*) and (S*j*) left one place to S*i*
056*i*0*k*[b]	S*i* S*i*<A*k*	Shift (S*i*) left (A*k*) places to S*i*
057*ijk*	S*i* S*i*,S*j*>A*k*	Shift (S*j*) and (S*i*) right (A*k*) places to S*i*
057*ij*0[b]	S*i* S*j*,S*i*>1	Shift (S*j*) and (S*i*) right one place to S*i*

Machine Code	CAL Syntax	Description
057*i*0*k*[b]	*Si Si*< A*k*	Shift (S*i*) right (A*k*) places to S*i*
060*ijk*	*Si Sj*+S*k*	Transmit the integer sum of (S*j*) and (S*k*) to S*i*
061*ijk*	*Si Sj*-S*k*	Transmit the integer difference of (S*j*) and (S*k*) to S*i*
061*i*0*k*[b]	*Si* -S*k*	Transmit the negative of (S*k*) to S*i*
062*ijk*	*Si Sj*+FS*k*	Transmit the floating-point sum of (S*j*) and (S*k*) to S*i*
062*i*0*k*[b]	*Si* +FS*k*	Transmit the normalized (S*k*) to S*i*
063*ijk*	*Si Sj*-FS*k*	Transmit the floating-point difference of (S*j*) and (S*k*) to S*i*
063*i*0*k*[b]	*Si* -FS*k*	Transmit the normalized negative of (S*k*) to S*i*
064*ijk*	*Si Sj**FS*k*	Transmit the floating-point product of (S*j*) and (S*k*) to S*i*
065*ijk*	*Si Sj**HS*k*	Transmit the half-precision rounded floating-point product of (S*j*) and (S*k*) to S*i*
066*ijk*	*Si Sj**RS*k*	Transmit the rounded full-precision floating-point product of (S*j*) and (S*k*) to S*i*
067*ijk*	*Si Sj**IS*k*	Transmit the reciprocal iteration of 2 minus the floating-point product of (S*j*) and (S*k*) to S*i*
070*ij*0	*Sj* /HS*j*	Transmit the floating-point reciprocal approximation of (S*j*) to S*i*
070*ij*6	*Si Sj**BT	Transmit the bit-matrix product of (S*j*) and the matrix BT to S*i*
071*i*0*k*	*Si* A*k*	Transmit (A*k*) to S*i* with no sign extension
071*i*1*k*	*Si* +A*k*	Transmit (A*k*) to S*i* with sign extension
071*i*2*k*	*Si* +FA*k*	Transmit (A*k*) to S*i* as an unnormalized floating-point number
071*i*30	*Si* 0.6	Transmit 0.75 x 2^{48} as a normalized floating-point constant into S*i*
071*i*40	*Si* 0.4	Transmit 0.5 x 2^{48} as a normalized floating-point constant into S*i*
071*i*50	*Si* 1.0	Transmit 1.0 as a normalized floating-point constant into S*i*
071*i*60	*Si* 2.0	Transmit 2.0 as a normalized floating-point constant into S*i*
071*i*70	*Si* 4.0	Transmit 4.0 as a normalized floating-point constant into 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* ST*j*	Transmit (ST*j*) to S*i*
073*i*00	*Si* VM	Transmit (VM) to S*i*
073*i*01	*Si* SR0	Transmit (SR0) to S*i*
073*i*02	SM *Si*	Transmit (S*i*) to SM
073*ij*3	ST*j Si*	Transmit (S*i*) to ST*j*
073*i*11[a,c]		Transmit the performance counter and status bits to S*i*
073*i*21[a,c]		Increment upper performance counter and transmit status bits to S*i*

Machine Code	CAL Syntax	Description
073*i*31[a,c]		Clear all maintenance modes and transmit status bits to S*i*
073*i*61[a,c]		Increment current performance counter (lower) and transmit status bits to S*i*
074*ijk*	*Si Tjk*	Transmit (T*jk*) to S*i*
075*ijk*	*Tjk Si*	Transmit (S*i*) to T*jk*
076*ijk*	*Si Vj*,A*k*	Transmit (V*j* element (A*k*)) to S*i*
077*ijk*	*Vi*,A*k Sj*	Transmit (S*j*) to V*i* element (A*k*)
077*i*0*k*[b]	*Vi*,A*k* 0	Clear element (A*k*) of register V*i*
10*hi*00*mn*	*Ai exp*,A*h*	Load from address ((A*h*) + *exp*) to A*i* (*h* ≠ 0)
100*i*00*mn*	*Ai exp*,0	Load from (*exp*) to A*i*
100*i*00*mn*	*Ai exp*,	Load from (*exp*) to A*i*
10*hi*0000	*Ai* ,A*h*	Load from address (A*h*) to A*i* (*h* ≠ 0)
11*hi*00*mn*	*exp*,A*h Ai*	Store (A*i*) to address (A*h*) + *exp* (*h* ≠ 0)
110*i*00*mn*	*exp*, 0 *Ai*	Store (A*i*) to *exp*
110*i*00*mn*	*exp*, *Ai*	Store (A*i*) to *exp*
11*hi*0000	,A*h Ai*	Store (A*i*) to address (A*h*) (*h* ≠ 0)
12*hi*00*mn*	*Si exp*,A*h*	Load from address ((A*h*) + *exp*) to S*i* (*h* ≠ 0)
120*i*00*mn*	*Si exp*,0	Load from (*exp*) to S*i*
120*i*00*mn*	*Si exp*	Load from (*exp*) to S*i*
12*hi*0000	*Si* ,A*h*	Load from address (A*h*) to S*i* (*h* ≠ 0)
13*hi*00*mn*	*exp*,A*h Si*	Store (S*i*) to address (A*h*) + *exp* (*h* ≠ 0)
130*i*00*mn*	*exp*,0 *Si*	Store (S*i*) to *exp*
130*i*00*mn*	*exp*, *Si*	Store (S*i*) to *exp*
13*hi*0000	,A*h Si*	Store (S*i*) to address (A*h*) (*h* ≠ 0)
140*ijk*	*Vi Sj*&V*k*	Transmit logical products of (S*j*) and (V*k* elements) to V*i* elements
141*ijk*	*Vi Vj*&V*k*	Transmit logical products of (V*j* elements) and (V*k* elements) to V*i* elements
142*ijk*	*Vi Sj*|V*k*	Transmit logical sums of (S*j*) and (V*k* elements) to V*i* elements
142*i*0*k*[b]	*Vi V*k	Transmit (V*k* elements) to V*i* elements
143*ijk*	*Vi Vj*|V*k*	Transmit logical sums of (V*j* elements) and (V*k* elements) to V*i* elements
144*ijk*	*Vi Sj*|V*k*	Transmit logical differences of (S*j*) and (V*k* elements) to V*i* elements
145*ijk*	*Vi Vj*|V*k*	Transmit logical differences of (V*j* elements) and (V*k* elements) to V*i* elements
145*iii*[b]	*Vi* 0	Clear V*i* elements
146*ijk*	*Vi Sj*|V*k*&VM	Transmit (S*j*) if VM bit = 1; (V*k* element) if VM bit = 0 to V*i* elements [scalar-vector merge]
146*i*0*k*[b]	*Vi* #VM&V*k*	Vector merge of (V*k* elements) and 0 to V*i* elements
147*ijk*	*Vi Vj*|V*k*&VM	Transmit (V*j* element) if VM bit = 1; (V*k* element) if VM bit = 0 to V*i* elements [vector-vector merge]
150*ijk*	*Vi Vj*<A*k*	Shift (V*j* elements) left (A*k*) places to V*i* elements

Machine Code	CAL Syntax	Description
150*ij*0[b]	*Vi Vj*<1	Shift (V*j* elements) left one place to V*i* elements
151*ijk*	*Vi Vj*>A*k*	Shift (V*j* elements) right (A*k*) places to V*i* elements
151*ij*0[b]	*Vi Vj*>1	Shift (V*j* elements) right one place to V*i* elements
152*ijk*	*Vi Vj*,V*j*<A*k*	Double shift (V*j* elements) left (A*k*) places to V*i* elements
152*ij*0[b]	*Vi Vj*,V*j*<1	Double shift (V*j* elements) left one place to V*i* elements
153*ijk*	*Vi Vj*,V*j*>A*k*	Double shift (V*j* elements) right (A*k*) places to V*i* elements
153*ij*0[b]	*Vi Vj*,V*j*>1	Double shift (V*j* elements) right one place to V*i* elements
154*ijk*	*Vi Sj*+V*k*	Transmit integer sums of (S*j*) and (V*k* elements) to V*i* elements
155*ijk*	*Vi Vj*+V*k*	Transmit integer sums of (V*j* elements) and (V*k* elements) to V*i* elements
156*ijk*	*Vi Sj*-V*k*	Transmit integer differences of (S*j*) and (V*k* elements) to V*i* elements
156*i*0*k*[b]	*Vi* -V*k*	Transmit twos complement of (V*k* elements) to V*i* elements
157*ijk*	*Vi Vj*-V*k*	Transmit integer differences of (V*j* elements) and (V*k* elements) to V*i* elements
160*ijk*	*Vi Sj**FV*k*	Transmit floating-point products of (S*j*) and (V*k* elements) to V*i* elements
161*ijk*	*Vi Vj**FV*k*	Transmit floating-point products of (V*j* elements) and (V*k* elements) to V*i* elements
162*ijk*	*Vi Sj**HV*k*	Transmit half-precision rounded floating-point products of (S*j*) and (V*k* elements) to V*i* elements
163*ijk*	*Vi Vj**HV*k*	Transmit half-precision rounded floating-point products of (V*j* elements) and (V*k* elements) to V*i* elements
164*ijk*	*Vi Sj**RV*k*	Transmit rounded floating-point products of (S*j*) and (V*k* elements) to V*i* elements
165*ijk*	*Vi Vj**RV*k*	Transmit rounded floating-point products of (V*j* elements) and (V*k* elements) to V*i* elements
166*ijk*	*Vi Sj**V*k*	Transmit 32-bit integer products of (S*j*) and (V*k* elements) to V*i* elements
167*ijk*	*Vi Vj**IV*k*	Transmit reciprocal iteration of 2 minus the floating-point product of (V*j* elements) and (V*k* elements) to V*i* elements
170*ijk*	*Vi Sj*+FV*k*	Transmit floating-point sums of (S*j*) and (V*k* elements) to V*i* elements
170*i*0*k*[b]	*Vi* +FV*k*	Transmit normalized (V*k* elements) to V*i* elements
171*ijk*	*Vi Vj*+FV*k*	Transmit floating-point sums of (V*j* elements) and (V*k* elements) to V*i* elements
172*ijk*	*Vi Sj*-FV*k*	Transmit floating-point differences of (S*j*) and (V*k* elements) to V*i* elements
172*i*0*k*[b]	*Vi* -FV*k*	Transmit normalized negative of (V*k* elements) to V*i* elements

Machine Code	CAL Syntax	Description
173*ijk*	*Vi Vj*-FV*k*	Transmit floating-point differences of (V*j* elements) and (V*k* elements) to V*i* elements
174*ij*0	*Vi* /HV*j*	Transmit floating-point reciprocal approximation of (V*j* elements) to V*i* elements
174*ij*1	*Vi* PV*j*	Transmit population count of (V*j* elements) to V*i* elements
174*ij*2	*Vi* QV*j*	Transmit population count parity of (V*j* elements) to V*i* elements
174*ij*3	*Vi* ZV*j*	Transmit the leading zero count of (V*j* elements) to V*i* elements
1740*j*4	BMM LV*j*	Transmit V*j* elements to BT matrix. This load of the 64-register BT matrix clears out allocations of the matrix that are not loaded.
174*ij*6	*Vi Vj**BT	Transmit bit-matrix product of (V*j* elements) and (BT) to V*i* elements
1750*j*0	VM V*j*,Z	Set VM bit if (V*j* element) = 0
1750*j*1	VM V*j*,N	Set VM bit if (V*j* element) ≠ 0
1750*j*2	VM V*j*,P	Set VM bit if (V*j* element) ≥ 0
1750*j*3	VM V*j*,M	Set VM bit if (V*j* element) < 0 (V*j* is negative)
175*ij*4	V*i*,VM V*j*,Z	Set VM bit if (V*j* element) = 0; also, the compressed indices of the V*j* element = 0 are stored in V*i*
175*ij*5	V*i*,VM V*j*,N	Set VM bit if (V*j* element) ≠ 0; also, the compressed indices of the V*j* element ≠ 0 are stored in V*i*
175*ij*6	V*i*,VM V*j*,P	Set VM bit if (V*j* element) ≥ 0; also, the compressed indices of the V*j* element ≥ 0 are stored in V*i*
175*ij*7	V*i*,VM V*j*,M	Set VM bit if (V*j* element) < 0; also, the compressed indices of the V*j* element < 0 are stored in V*i* elements
176*i*0*k*	*Vi* ,A0,A*k*	Load (VL) words from address (A0) incremented by (A*k*) to V*i* elements
176*i*00	*Vi* ,A0,1	Load (VL) words from address (A0) incremented by 1 to V*i* elements
176*i*1*k*	*Vi* ,A0,V*k*	Load (VL) words from address ((A0) + (V*k* elements)) to V*i* elements
1770*jk*	,A0,A*k V*j	Store (VL) words from V*j* elements to address (A0) incremented by (A*k*)
1770*j*0	,A0,1 V*j*	Store (VL) words from V*j* elements to address (A0) incremented by 1
1771*jk*	,A0,V*k* V*j*	Store (VL) words from V*j* elements to address ((A0) + (V*k* elements))