APPENDIX: CRAY EL SERIES INSTRUCTION SET

	Instruction	CAL		Unit	Description
	000000	ERR			Error exit
‡	0010*jk*	CA, A*j*	A*k*		Set the channel (A*j*) CA register to (A*k*) and begin I/O sequence
	001000	PASS			Pass
‡	0011*jk*	CL, A*j*	A*k*		Set the channel (A*j*) CL register to (A*k*)
‡	0012*j*0	C*i*, A*j*			Clear channel (A*j*) interrupt and error flags; clear device Master Clear (output channel)
‡	0012*j*1	MC, A*j*			Clear channel (A*j*) interrupt and error flags; set device Master Clear (output channel); clear device ready-held (input channel)
‡	0013*j*0	XA	A*j*		Transmit (A*j*) to XA register
‡	0014*j*0	RT	S*j*		Transmit (S*j*) to RTC register
‡	0014*j*1	SIPI A*j*			Set Interprocessor Interrupt request to CPU (A*j*)
	001401	SIPI			Set Interprocessor Interrupt of CPU 0
‡	001402	CIPI			Clear Interprocessor Interrupt
‡	0014*j*3	CLN	A*j*		Transmit (A*j*) to CLN register
‡	0014*j*4	PCI	S*j*		Enter interrupt interval (II) register with (S*j*)
‡	001405	CCI			Clear programmable clock interrupt (PCI) request
‡	001406	ECI			Enable PCI request
‡	001407	DCI			Disable PCI request
	001501				Disable all error correction
	001541				Enable replacement of checkbytes with data on all ports' writes to reads
	00200k	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 error interrupts
	002400	DRI			Disable operand range error interrupts
	002500	DBM			Disable bidirectional memory transfers
E	002504	DCBW			Disable concurrent block write

	Instruction	CAL		Unit	Description
E	002506	DSBO			Disable scalar and block overlap
	002600	EBM			Enable bidirectional memory transfers
E	002604	ECBW			Enable concurrent block write
E	002606	ESBO			Enable scalar and block overlap
	002700	CMR			Complete memory reference
	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 and Set semaphore *jk*; $0 \le jk \le 37_8$
	0036*jk*	SM*jk*	0		Clear semaphore *jk*; $0 \le jk \le 37_8$
	0037*jk*	SM*jk*	1		Set semaphore *jk*; $0 \le jk \le 37_8$
	004000	EX			Normal exit
	0050*jk*	J	B*jk*		Jump to (B*jk*)
	006*ijkm*	*j*	*exp*		Jump to *exp* = *ijkm*
	007*ijkm*	R	*exp*		Return jump to *exp* = *ijkm*; set B00 to (P) + 2
	010*ijkm*	JAZ	*exp*		Jump to *exp* = *ijkm* if (A0) = 0 (2^2 of *i* = 0)
	011*ijkm*	JAN	*exp*		Jump to *exp* = *ijkm* if (A0) ≠ 0 (2^2 of *i* = 0)
	012*ijkm*	JAP	*exp*		Jump to *exp* = *ijkm* if (A0) positive (2^2 of *i* = 0)
	013*ijkm*	JAM	*exp*		Jump to *exp* = *ijkm* if (A0) negative (2^2 of *i* = 0)
	014*ijkm*	JSZ	*exp*		Jump to *exp* = *ijkm* if (S0) = 0 (2^2 of *i* = 0)
	015*ijkm*	JSN	*exp*		Jump to *exp* = *ijkm* if (S0) ≠ 0 (2^2 of *i* = 0)
	016*ijkm*	JSP	*exp*		Jump to *exp* = *ijkm* if (S0) positive (2^2 of *i* = 0)
	017*ijkm*	JSM	*exp*		Jump to *exp* = *ijkm* if (S0) negative (2^2 of *i* = 0)
	01*hijkm*	Ah	*exp*		Transmit *exp* = *ijkm* to A*h* (2^2 of *i* = 1)
‡†, X	020*iijkm*	A*i*	*exp*		Transmit *exp* = *jkm* to A*i*
‡†, Y	020*i*00*mn*	A*i*	*exp*		Transmit *exp* = *nm* to A*i*
‡†	021*ijkm*	A*i*	*exp*		Transmit one's complement of *exp* = *jkm* to A*i*
‡†, Y	021*i*00*mn*	A*i*	*exp*		Transmit one's complement of *exp* = *nm* 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*

	Instruction	CAL		Unit	Description
	025*ijk*	B*jk*	A*i*		Transmit (A*i*) to B*jk*
	026*ij*0	A*i*	PS*j*	S Pop	Transmit population count of (S*j*) to A*i*
	026*ij*1	A*i*	QS*j*	S Pop	Transmit 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*	S/LZ	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*	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	Transmit integer sum of (A*j*) plus 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* (A*i* = 37777777777) in Y-mode
†	031*i*0k	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		Transmit lowest interrupting channel number to A*i* (*j* = 0)
	033*ij*0	A*i*	CA, A*j*		Transmit address of channel (A*j*) to A*i* (*j* ≠ 0)
	033*ij*1	A*i*	CE, A*j*		Transmit error flag of channel (A*j*) to A*i* (*j* ≠ 0)
†	034*ijk*	B*jk*, A*i*	,A0	Memory	Read (A*i*) words to B registers starting at B*jk* from memory address ((A0) + (DBA))
	034*ijk*	B*jk*, A*i*,	0, A0	Memory	Read (A*i*) words to B registers starting at B*jk* from memory address ((A0) + (DBA))
	035*ijk*	,A0	B*jk*, A*i*	Memory	Write (A*i*) words from B registers B*jk* to memory address ((A0) + (DBA))
†	035*ijk*	0, A0	B*jk*, A*i*	Memory	Write (A*i*) words from B registers B*jk* to memory address ((A0) + (DBA))
	036*ijk*	T*jk*, A*i*	,A0	Memory	Read (A*i*) words to T registers starting at T*jk* from memory address ((A0) + (DBA))
†	036*ijk*	T*jk*, A*i*	0, A0	Memory	Read (A*i*) words to T registers starting at T*jk* from memory address ((A0) + (DBA))
	037*ijk*	,A0	T*jk*, A*i*	Memory	Write (A*i*) words from T registers T*jk* to memory address ((A0) + (DBA))
†	037*ijk*	0, A0	T*jk*, A1	Memory	Write (A*i*) words from T registers T*jk* to memory address ((A0) + (DBA))
X	040*ijkm*	S*i*	*exp*		Transmit *exp* = *jkm* to S*i*
Y	040*i*00*mn*	S*i*	*exp*		Transmit *exp* = *nm* to S*i*

	Instruction	CAL		Unit	Description
X	041*ijkm*	S*i*	*exp*		Transmit one's complement of *exp* = *jkm* to S*i*
Y	041*i*00*mn*	S*i*	*exp*		Transmit one's complement of *exp* = *nm* to S*i*
	042*ijk*	S*i*	<*exp*	S Logical	Form ones mask *exp* = 100_8 – *jk* bits in S*i* from the right
†	042*ijk*	S*i*	#>*exp*	S Logical	Form zeroes mask *exp* = *jk* bits in S*i* from the left
†	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* (Si = 177777 177777 177777 177777)
	043*ijk*	S*i*	>*exp*	S Logical	Form ones mask *exp* = *jk* bits in S*i* from the left
†	043*ijk*	S*i*	#<*exp*	S Logical	Form zeroes mask *exp* = 100_8 bits in S*i* from the right
†	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 one's complement of (S*k*) to S*i*
†	045*ij*0	S*i*	#SB&S*j*	S Logical	Transmit (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*), enter into S*i*
†	046*ij*0	S*i*	SB\S*j*	S Logical	Toggle sign bit of (S*j*), 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 one's 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	Transmit one's 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* *scalar merge
†	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*

	Instruction	CAL		Unit	Description
†	051*ij*0	S*i*	SBIS*j*	S Logical	Logical sum of (S*j*) and sign bit to S*i* (*j* ≠ 0)
†	051*i*00	S*i*	SB	S Logical	Transmit sign bit into S*i*
	052*ijk*	S0	S*i* < *exp*	S Shift	Shift (S*i*) left *exp* = *jk* places to S0
	053*ijk*	S0	S*i* > *exp*	S Shift	Shift (S*i*) right *exp* = $100_8 - jk$ places to S0
	054*ijk*	S*i*	S*i* < *exp*	S Shift	Shift (S*i*) left *exp* = *jk* places to S*i*
	055*ijk*	S*i*	S*i* > *exp*	S Shift	Shift (S*i*) right *exp* = $100_8 - jk$ places to S*i*
	056*ijk*	S*i*	S*i*, S*j* < A*k*	S Shift	Shift (S*i* and S*j*) left (A*k*) places to S*i*
†	056*ij*0	S*i*	S*i*, S*j* < 1	S Shift	Shift (S*i* and S*j*) left one place to S*i*
†	056*i*0*k*	S*i*	S*i* < A*k*	S Shift	Shift (S*i*) left (A*k*) places to S*i*
	057*ijk*	S*i*	S*j*, S*i* > A*k*	S Shift	Shift (S*j* and S*i*) right (A*k*) places to S*i*
†	057*ij*0	S*i*	S*j*, S*i* > 1	S Shift	Shift (S*j* and S*i*) right one place to S*i*
†	057*i*0*k*	S*i*	S*i* > A*k*	S Shift	Shift (S*i*) right (A*k*) places to S*i*
	060*ijk*	S*i*	S*j* + S*k*	S Int Add	Integer sum of (S*j*) and (S*k*) to S*i*
†	060*i*0*k*	S*i*	S*k*	S Int Add	Transmit (S*k*) to S*i*
†	060*ij*0	S*i*	S*j* + S0	S Int Add	Integer sum 2^{63} and (S*j*) to S*i*
	061*ijk*	S*i*	S*j* – S*k*	S Int Add	Integer difference of (S*j*) less (S*k*) to S*i*
†	061*i*0*k*	S*i*	–S*k*	S Int Add	Transmit negative of (S*k*) to S*i*
†	061*ij*0	S*i*	S*j* – S0	S Int Add	Integer difference of (S*j*) less 2^{63} to S*i*
	062*ijk*	S*i*	S*j* – FS*k*	Fp Add	Floating-point sum of (S*j*) and (S*k*) to S*i*
†	062*i*0*k*	S*i*	+FS*k*	Fp Add	Normalize (S*k*) to S*i*
	063*ijk*	S*i*	S*j* – FS*k*	Fp Add	Floating-point difference of (S*j*) and (S*k*) to S*i*
†	063*i*0*k*	S*i*	–FS*k*	Fp Add	Transmit normalized negative of (S*k*) to S*i*
	064*ijk*	S*i*	S*j* * FS*k*	Fp Mult	Floating-point product of (S*j*) and (S*k*) to S*i*
	065*ijk*	S*i*	S*j* * HS*k*	Fp Mult	Half-precision rounded floating-point product of (S*j*) and (S*k*) to S*i*
	066*ijk*	S*i*	S*j* * RS*k*	Fp Mult	Full-precision rounded floating-point product of (S*j*) and (S*k*) to S*i*
	067*ijk*	S*i*	S*j* * IS*k*	Fp Mult	Two minus the floating-point product of (S*j*) and (S*k*) to S*i*
	070*ij*0	S*i*	/HS*j*	Fp Recp	Floating-point reciprocal approximation of (S*j*) to S*i*
	071*i*0*k*	S*i*	A*k*		Transmit (Ak) to S*i* with no sign extension
	071*i*1*k*	S*i*	+A*k*		Transmit (Ak) to S*i* with sign extension
	071*i*2*k*	S*i*	+FA*k*		Transmit (Ak) to S*i* as unnormalized floating-point number (exponent = 40060)

	Instruction	CAL		Unit	Description
	071*i*30	S*i*	0.6		Transmit constant 0.75 x 2^{48} to S*i* (Si = 040060 140000 000000 000000)
	071*i*40	S*i*	0.4		Transmit constant 0.5 to S*i* (Si = 040000 100000 000000 000000)
	071*i*50	S*i*	1.0		Transmit constant 1.0 to S*i* (Si = 040001 100000 000000 000000)
	071*i*60	S*i*	2.0		Transmit constant 2.0 to S*i* (Si = 040002 100000 000000 000000)
	071*i*70	S*i*	4.0		Transmit constant 4.0 to S*i* (Si = 040003 100000 000000 000000)
	072*i*00	S*i*	RT		Transmit (RTC) to S*i*
	072*i*02	S*i*	SM		Transmit (SM) to S*i*
	072*ij*3	S*i*	ST*j*		Transmit (ST*j*) to S*i*
	073*i*00	S*i*	VM		Transmit (VM) to S*i*
	073*i*01	S*i*	SR*j*		Transmit status register (SR*j*) bits to S*i*
	073*i*11	¶¶			Read performance counter to S*i*
E	073*i*31	¶¶			Clear memory maintenance modes and read status register into S*i*
	073*i*02	SM	S*i*		Transmit (S*i*) to SM
	0731*j*3	ST*j*	S*i*		Transmit (S*i*) to ST*j*
	074*ijk*	S*i*	T*jk*		Transmit (T*jk*) to S*i*
	075*ijk*	T*jk*	S*i*		Transmit (S*i*) to T*jk*
	076*ijk*	S*i*			Transmit (V*j*, element (A*k*)) to S*i*
	077*ijk*	V*i*, A*k*	S*j*		Transmit (S*j*) to V*i* element (A*k*)
†	077*i*0*k*	V*i*, A*k*	0		Clear V*i* element (A*k*)
X	10*hijkm*	A*i*	*exp*, A*h*	Memory	Read from memory address ((A*h*) + *jkm* + (DBA)) to A*i* (*h* ≠ 0)
Y	10*hi*00*mn*	A*i*	*exp*, A*h*	Memory	Read from memory address ((A*h*) + *nm* + (DBA)) to A*i* (*h* ≠ 0)
†, X	100*ijkm*	A*i*	*exp*, 0	Memory	Read from memory address (*jkm* + (DBA)) to A*i*
Y	100*i*00*mn*	A*i*	*exp*, 0	Memory	Read from memory address (*nm* + (DBA)) to A*i*
†, X	100*ijkm*	A*i*	*exp*,	Memory	Read from memory address (*jkm* + (DBA)) to A*i*
Y	100*i*00*mn*	A*i*	*exp*,	Memory	Read from memory address (*nm* + (DBA)) to A*i*
X	10*hi*000	A*i*	,A*h*	Memory	Read from memory address ((A*h*) + (DBA)) to A*i* (*h* ≠ 0)
Y	10*hi*0000	A*i*	,A*h*	Memory	Read from memory address ((A*h* + (DBA)) to A*i* (*h* ≠ 0)
X	11*hijkm*	*exp*,A*h*	A*i*	Memory	Write (A*i*) to memory address ((A*h*) + *jkm* + (DBA)) (*h* ≠ 0)

	Instruction	CAL		Unit	Description
Y	11*hi*00*mn*	*exp*,A*h*	A*i*	Memory	Write (A*i*) to memory address ((A*h*) + *nm* + (DBA)) (A*h* ≠ 0)
†, X	110*ijkm*	*exp*,0	A*i*	Memory	Write (A*i*) to memory address (*jkm* + (DBA))
Y	110*i*00*mn*	*exp*,0	A*i*	Memory	Write (A*i*) to memory address (*nm* + (DBA))
†, X	110*ijkm*	*exp*,	A*i*	Memory	Write (A*i*) to memory address (*jkm* + (DBA))
Y	110*i*00*mn*	*exp*,	A*i*	Memory	Write (A*i*) to memory address (*nm* + (DBA))
†, X	11*hi*000	,A*h*	A*i*	Memory	Write (A*i*) to memory address ((A*h*) + (DBA)) (*h* ≠ 0)
Y	11*hi*0000	,A*h*	A*i*	Memory	Write (A*i*) to memory address ((A*h* + (DBA)) (*h* ≠ 0)
X	12*hijkm*	S*i*	*exp*,A*h*	Memory	Read from memory address ((A*h*) + *jkm* + (DBA)) to S*i* (*h* ≠ 0)
Y	12*hi*00*mn*	S*i*	*exp*,A*h*	Memory	Read from memory address ((A*h*) + *nm* + (DBA)) to S*i* (*h* ≠ 0)
X	120*ijkm*	S*i*	*exp*,0	Memory	Read from memory address (*jkm* + (DBA)) to S*i*
Y	120*i*00*mn*	S*i*	*exp*,0	Memory	Read from memory address (*nm* + (DBA)) to S*i*
X	120*ijkm*	S*i*	*exp*,	Memory	Read from memory address (*jkm* + (DBA)) to S*i*
Y	120*i*00*mn*	S*i*	*exp*,	Memory	Read from memory address (*nm* + (DBA)) to S*i*
, X	12*hi*000	S*i*	,A*h*	Memory	Read from memory address ((A*h*) + (DBA)) to S*i* (*h* ≠ 0)
Y	12*hi*0000	S*i*	,A*h*	Memory	Read from memory address ((A*h*) + (DBA)) to S*i* (*h* ≠ 0)
X	13*hijkm*	*exp*,A*h*	S*i*	Memory	Write (S*i*) to memory address ((A*h*) + *jkm* + (DBA)) (*h* ≠ 0)
Y	13*hi*00*mn*	*exp*,A*h*	S*i*	Memory	Write (S*i*) to memory address ((A*h*) + *nm* + (DBA)) (*h* ≠ 0)
†, X	130*ijkm*	*exp*,0	S*i*	Memory	Write (S*i*) to memory address (*jkm* + (DBA))
Y	130*i*00*mn*	*exp*,0	S*i*	Memory	Write (S*i*) to memory address (*nm* + (DBA))
X	130*ijkm*	*exp*,	S*i*	Memory	Write (S*i*) to memory address (*jkm* + (DBA))
Y	130*i*00*mn*	*exp*,	S*i*	Memory	Write (S*i*) to memory address (*nm* + (DBA))
X	13*hi*000	,A*h*	S*i*	Memory	Write (S*i*) to memory address ((A*h*) + (DBA)) (*h* ≠ 0)
Y	13*hi*0000	,A*h*	S*i*	Memory	Write (S*i*) to memory address ((A*h*) + (DBA)) (*h* ≠ 0)

	Instruction	CAL		Unit	Description
	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* *scalar *vector merge
†	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* *vector *vector merge
	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*
X	166*ijk*	V*i*	S*j* * IV*k*	Fp Mult	Two minus the floating-point products of (S*j*) and (V*k*) to V*i*
Y	166*ijk*	V*i*	S*j* * V*k*	Fp Mult	32-bit integer products of (S*j*) and (V*k*) to V*i*

	Instruction	CAL		Unit	Description
	167*ijk*	V*i*	V*j* * IV*k*	Fp Mult	Two minus the 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 Recp	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 if (V*j*) = 0
	1750*j*1	VM	V*j*,N	V Logical	VM = 1 if (V*j*) ≠ 0
	1750*j*2	VM	V*j*,P	V Logical	VM = 1 if (V*j*) positive; 0 is positive
	1750*j*3	VM	V*j*,M	V Logical	VM = 1 if (V*j*) negative; 1 is negative
	175*ij*4	V*i*,VM	V*j*,Z	V Logical	VM bit = 1 if (V*j* element) = 0 and element index is loaded into (compressed V*i*)
	175*ij*5	V*i*,VM	V*j*,N	V Logical	VM bit = 1 if (V*j* element) ≠ 0 and element index is loaded into (compressed V*i*)
	175*ij*6	V*i*,VM	V*j*,P	V Logical	VM bit = 1 if (V*j* element) ≥ 0 and element index is loaded into (compressed V*i*)
	175*ij*7	V*i*,VM	V*j*,M	V Logical	VM bit = 1 if (V*j* element) < 0 and element index is loaded into (compressed V*i*)
	176*i*0*k*	V*i*	A0,A*k*	Memory	Read (VL) words to V*i* from memory address ((A0) + (DBA)) incremented by (A*k*)
	176*i*00	V*i*	A0,1	Memory	Read (VL) words to V*i* from memory address ((A0 + (DBA)) incremented by 1
	176*i*1*k*	V*i*	A0,V*k*	Memory	Read (VL) words to V*i* from memory address ((A0 + (V*k*) + (DBA)) *gather
	1770*jk*	,A0,A*k*	V*j*	Memory	Write (VL) words from V*j* to memory address ((A0) + (DBA)) incremented by (A*k*)

Instruction	CAL		Unit	Description
1770*j*0	,A0, 1	V*j*	Memory	Write (VL) words from V*j* to memory address ((A0) + (DBA)) incremented by 1
1771*jk*	,A0,V*k*	V*j*	Memory	Write (VL) words from V*j* to memory address ((A0) + (V*k*) + (DBA)) *scatter

† – Special syntax mode
‡ – Privileged to monitor mode
‡† – Generated depending on *exp.*
¶¶ – Not supported by CAL version 2
X – X-mode instruction
Y – Y-mode instruction
E – E-mode instruction
( ) – Read as *the contents of ...*

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}