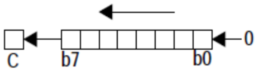
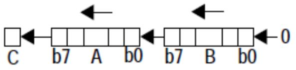
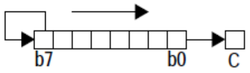
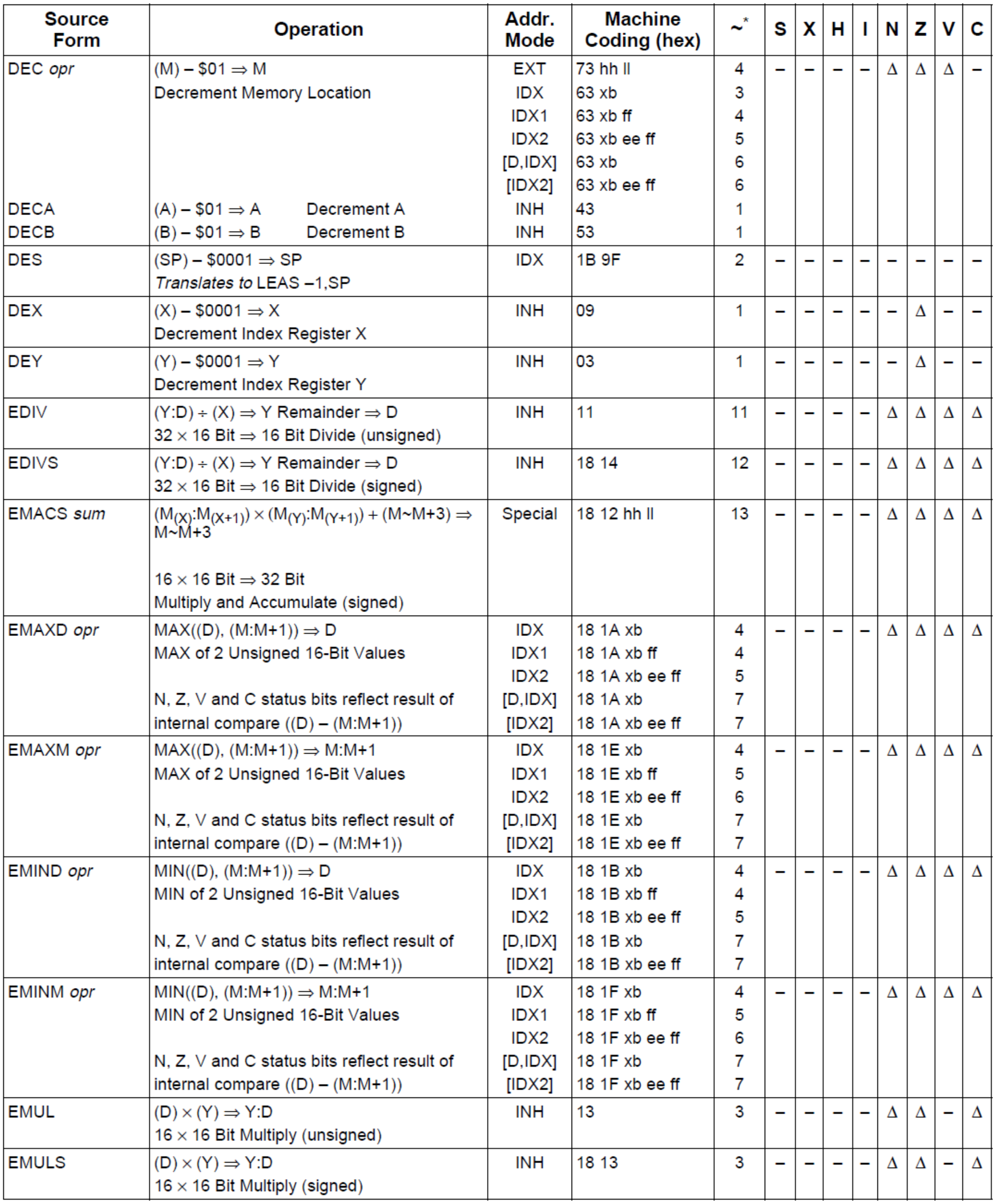
Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
ABA	(A) + (B) ⇒ A	INH	18 06	2	–	–	Δ	–	Δ	Δ	Δ	Δ
	Add Accumulators A and B											
ABX	(B) + (X) ⇒ X	IDX	1A E5	2	–	–	–	–	–	–	–	–
	Translates to LEAX B,X											
ABY	(B) + (Y) ⇒ Y	IDX	19 ED	2	–	–	–	–	–	–	–	–
	Translates to LEAY B,Y											
ADCA *opr*	(A) + (M) + C ⇒ A	IMM	89 ii	1	–	–	Δ	–	Δ	Δ	Δ	Δ
	Add with Carry to A	DIR	99 dd	3								
		EXT	B9 hh ll	3								
		IDX	A9 xb	3								
		IDX1	A9 xb ff	3								
		IDX2	A9 xb ee ff	4								
		[D,IDX]	A9 xb	6								
		[IDX2]	A9 xb ee ff	6								
ADCB *opr*	(B) + (M) + C ⇒ B	IMM	C9 ii	1	–	–	Δ	–	Δ	Δ	Δ	Δ
	Add with Carry to B	DIR	D9 dd	3								
		EXT	F9 hh ll	3								
		IDX	E9 xb	3								
		IDX1	E9 xb ff	3								
		IDX2	E9 xb ee ff	4								
		[D,IDX]	E9 xb	6								
		[IDX2]	E9 xb ee ff	6								
ADDA *opr*	(A) + (M) ⇒ A	IMM	8B ii	1	–	–	Δ	–	Δ	Δ	Δ	Δ
	Add without Carry to A	DIR	9B dd	3								
		EXT	BB hh ll	3								
		IDX	AB xb	3								
		IDX1	AB xb ff	3								
		IDX2	AB xb ee ff	4								
		[D,IDX]	AB xb	6								
		[IDX2]	AB xb ee ff	6								
ADDB *opr*	(B) + (M) ⇒ B	IMM	CB ii	1	–	–	Δ	–	Δ	Δ	Δ	Δ
	Add without Carry to B	DIR	DB dd	3								
		EXT	FB hh ll	3								
		IDX	EB xb	3								
		IDX1	EB xb ff	3								
		IDX2	EB xb ee ff	4								
		[D,IDX]	EB xb	6								
		[IDX2]	EB xb ee ff	6								
ADDD *opr*	(A:B) + (M:M+1) ⇒ A:B	IMM	C3 jj kk	2	–	–	–	–	Δ	Δ	Δ	Δ
	Add 16-Bit to D (A:B)	DIR	D3 dd	3								
		EXT	F3 hh ll	3								
		IDX	E3 xb	3								
		IDX1	E3 xb ff	3								
		IDX2	E3 xb ee ff	4								
		[D,IDX]	E3 xb	6								
		[IDX2]	E3 xb ee ff	6								

Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
ANDA *opr*	(A) • (M) ⇒ A	IMM	84 ii	1	–	–	–	–	Δ	Δ	0	–
	Logical And A with Memory	DIR	94 dd	3								
		EXT	B4 hh ll	3								
		IDX	A4 xb	3								
		IDX1	A4 xb ff	3								
		IDX2	A4 xb ee ff	4								
		[D,IDX]	A4 xb	6								
		[IDX2]	A4 xb ee ff	6								
ANDB *opr*	(B) • (M) ⇒ B	IMM	C4 ii	1	–	–	–	–	Δ	Δ	0	–
	Logical And B with Memory	DIR	D4 dd	3								
		EXT	F4 hh ll	3								
		IDX	E4 xb	3								
		IDX1	E4 xb ff	3								
		IDX2	E4 xb ee ff	4								
		[D,IDX]	E4 xb	6								
		[IDX2]	E4 xb ee ff	6								
ANDCC *opr*	(CCR) • (M) ⇒ CCR Logical And CCR with Memory	IMM	10 ii	1	⇓	⇓	⇓	⇓	⇓	⇓	⇓	⇓
ASL *opr*		EXT	78 hh ll	4	–	–	–	–	Δ	Δ	Δ	Δ
		IDX	68 xb	3								
		IDX1	68 xb ff	4								
		IDX2	68 xb ee ff	5								
	Arithmetic Shift Left	[D,IDX]	68 xb	6								
		[IDX2]	68 xb ee ff	6								
ASLA	Arithmetic Shift Left Accumulator A	INH	48	1								
ASLB	Arithmetic Shift Left Accumulator B	INH	58	1								
ASLD	 Arithmetic Shift Left Double	INH	59	1	–	–	–	–	Δ	Δ	Δ	Δ
ASR *opr*		EXT	77 hh ll	4	–	–	–	–	Δ	Δ	Δ	Δ
		IDX	67 xb	3								
		IDX1	67 xb ff	4								
	Arithmetic Shift Right	IDX2	67 xb ee ff	5								
		[D,IDX]	67 xb	6								
		[IDX2]	67 xb ee ff	6								
ASRA	Arithmetic Shift Right Accumulator A	INH	47	1								
ASRB	Arithmetic Shift Right Accumulator B	INH	57	1								
BCC *rel*	Branch if Carry Clear (if C = 0)	REL	24 rr	3/1	–	–	–	–	–	–	–	–
BCLR *opr, msk*	(M) • ($\overline{\text{mm}}$) ⇒ M	DIR	4D dd mm	4	–	–	–	–	Δ	Δ	0	–
	Clear Bit(s) in Memory	EXT	1D hh ll mm	4								
		IDX	0D xb mm	4								
		IDX1	0D xb ff mm	4								
		IDX2	0D xb ee ff mm	6								
BCS *rel*	Branch if Carry Set (if C = 1)	REL	25 rr	3/1	–	–	–	–	–	–	–	–
BEQ *rel*	Branch if Equal (if Z = 1)	REL	27 rr	3/1	–	–	–	–	–	–	–	–
BGE *rel*	Branch if Greater Than or Equal (if N ⊕ V = 0) (signed)	REL	2C rr	3/1	–	–	–	–	–	–	–	–
BGND	Place CPU in Background Mode see Background Mode section.	INH	00	5	–	–	–	–	–	–	–	–
BGT *rel*	Branch if Greater Than (if Z + (N ⊕ V) = 0) (signed)	REL	2E rr	3/1	–	–	–	–	–	–	–	–
BHI *rel*	Branch if Higher (if C + Z = 0) (unsigned)	REL	22 rr	3/1	–	–	–	–	–	–	–	–

Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
BHS *rel*	Branch if Higher or Same (if C = 0) (unsigned) same function as BCC	REL	24 rr	3/1	–	–	–	–	–	–	–	–
BITA *opr*	(A) • (M) Logical And A with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	85 ii 95 dd B5 hh ll A5 xb A5 xb ff A5 xb ee ff A5 xb A5 xb ee ff	1 3 3 3 3 4 6 6	–	–	–	–	Δ	Δ	0	–
BITB *opr*	(B) • (M) Logical And B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C5 ii D5 dd F5 hh ll E5 xb E5 xb ff E5 xb ee ff E5 xb E5 xb ee ff	1 3 3 3 3 4 6 6	–	–	–	–	Δ	Δ	0	–
BLE *rel*	Branch if Less Than or Equal (if Z + (N ⊕ V) = 1) (signed)	REL	2F rr	3/1	–	–	–	–	–	–	–	–
BLO *rel*	Branch if Lower (if C = 1) (unsigned) same function as BCS	REL	25 rr	3/1	–	–	–	–	–	–	–	–
BLS *rel*	Branch if Lower or Same (if C + Z = 1) (unsigned)	REL	23 rr	3/1	–	–	–	–	–	–	–	–
BLT *rel*	Branch if Less Than (if N ⊕ V = 1) (signed)	REL	2D rr	3/1	–	–	–	–	–	–	–	–
BMI *rel*	Branch if Minus (if N = 1)	REL	2B rr	3/1	–	–	–	–	–	–	–	–
BNE *rel*	Branch if Not Equal (if Z = 0)	REL	26 rr	3/1	–	–	–	–	–	–	–	–
BPL *rel*	Branch if Plus (if N = 0)	REL	2A rr	3/1	–	–	–	–	–	–	–	–
BRA *rel*	Branch Always (if 1 = 1)	REL	20 rr	3	–	–	–	–	–	–	–	–
BRCLR *opr, msk, rel*	Branch if (M) • (mm) = 0 (if All Selected Bit(s) Clear)	DIR EXT IDX IDX1 IDX2	4F dd mm rr 1F hh ll mm rr 0F xb mm rr 0F xb ff mm rr 0F xb ee ff mm rr	4 5 4 6 8	–	–	–	–	–	–	–	–
BRN *rel*	Branch Never (if 1 = 0)	REL	21 rr	1	–	–	–	–	–	–	–	–
BRSET *opr, msk, rel*	Branch if ($\overline{M}$) • (mm) = 0 (if All Selected Bit(s) Set)	DIR EXT IDX IDX1 IDX2	4E dd mm rr 1E hh ll mm rr 0E xb mm rr 0E xb ff mm rr 0E xb ee ff mm rr	4 5 4 6 8	–	–	–	–	–	–	–	–
BSET *opr, msk*	(M) + (mm) ⇒ M Set Bit(s) in Memory	DIR EXT IDX IDX1 IDX2	4C dd mm 1C hh ll mm 0C xb mm 0C xb ff mm 0C xb ee ff mm	4 4 4 4 6	–	–	–	–	Δ	Δ	0	–
BSR *rel*	(SP) – 2 ⇒ SP; $RTN_H:RTN_L \Rightarrow M_{(SP)}:M_{(SP+1)}$ Subroutine address ⇒ PC Branch to Subroutine	REL	07 rr	4	–	–	–	–	–	–	–	–

Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
BVC *rel*	Branch if Overflow Bit Clear (if V = 0)	REL	28 rr	3/1	–	–	–	–	–	–	–	–
BVS *rel*	Branch if Overflow Bit Set (if V = 1)	REL	29 rr	3/1	–	–	–	–	–	–	–	–
CALL *opr, page*	(SP) – 2 ⇒ SP; $RTN_H:RTN_L \Rightarrow M_{(SP)}:M_{(SP+1)}$ (SP) – 1 ⇒ SP; (PPG) ⇒ $M_{(SP)}$; pg ⇒ PPAGE register; Program address ⇒ PC Call subroutine in extended memory (Program may be located on another expansion memory page.)	EXT IDX IDX1 IDX2	4A hh ll pg 4B xb pg 4B xb ff pg 4B xb ee ff pg	8 8 8 9	–	–	–	–	–	–	–	–
CALL [D,*r*] CALL [*opr*,*r*]	Indirect modes get program address and new pg value based on pointer. *r* = X, Y, SP, or PC	[D,IDX] [IDX2]	4B xb 4B xb ee ff	10 10	–	–	–	–	–	–	–	–
CBA	(A) – (B) Compare 8-Bit Accumulators	INH	18 17	2	–	–	–	–	Δ	Δ	Δ	Δ
CLC	0 ⇒ C *Translates to* ANDCC #$FE	IMM	10 FE	1	–	–	–	–	–	–	–	0
CLI	0 ⇒ I *Translates to* ANDCC #$EF (enables I-bit interrupts)	IMM	10 EF	1	–	–	–	0	–	–	–	–
CLR *opr*	0 ⇒ M Clear Memory Location	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	79 hh ll 69 xb 69 xb ff 69 xb ee ff 69 xb 69 xb ee ff	3 2 3 3 5 5	–	–	–	–	0	1	0	0
CLRA	0 ⇒ A Clear Accumulator A	INH	87	1								
CLRB	0 ⇒ B Clear Accumulator B	INH	C7	1								
CLV	0 ⇒ V *Translates to* ANDCC #$FD	IMM	10 FD	1	–	–	–	–	–	–	0	–
CMPA *opr*	(A) – (M) Compare Accumulator A with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	81 ii 91 dd B1 hh ll A1 xb A1 xb ff A1 xb ee ff A1 xb A1 xb ee ff	1 3 3 3 3 4 6 6	–	–	–	–	Δ	Δ	Δ	Δ
CMPB *opr*	(B) – (M) Compare Accumulator B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C1 ii D1 dd F1 hh ll E1 xb E1 xb ff E1 xb ee ff E1 xb E1 xb ee ff	1 3 3 3 3 4 6 6	–	–	–	–	Δ	Δ	Δ	Δ

Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
COM *opr*	($\overline{\text{M}}$) ⇒ M *equivalent to* \$FF – (M) ⇒ M	EXT	71 hh ll	4	–	–	–	–	Δ	Δ	0	1
	1's Complement Memory Location	IDX	61 xb	3								
		IDX1	61 xb ff	4								
		IDX2	61 xb ee ff	5								
		[D,IDX]	61 xb	6								
		[IDX2]	61 xb ee ff	6								
COMA	($\overline{\text{A}}$) ⇒ A Complement Accumulator A	INH	41	1								
COMB	($\overline{\text{B}}$) ⇒ B Complement Accumulator B	INH	51	1								
CPD *opr*	(A:B) – (M:M+1)	IMM	8C jj kk	2	–	–	–	–	Δ	Δ	Δ	Δ
	Compare D to Memory (16-Bit)	DIR	9C dd	3								
		EXT	BC hh ll	3								
		IDX	AC xb	3								
		IDX1	AC xb ff	3								
		IDX2	AC xb ee ff	4								
		[D,IDX]	AC xb	6								
		[IDX2]	AC xb ee ff	6								
CPS *opr*	(SP) – (M:M+1)	IMM	8F jj kk	2	–	–	–	–	Δ	Δ	Δ	Δ
	Compare SP to Memory (16-Bit)	DIR	9F dd	3								
		EXT	BF hh ll	3								
		IDX	AF xb	3								
		IDX1	AF xb ff	3								
		IDX2	AF xb ee ff	4								
		[D,IDX]	AF xb	6								
		[IDX2]	AF xb ee ff	6								
CPX *opr*	(X) – (M:M+1)	IMM	8E jj kk	2	–	–	–	–	Δ	Δ	Δ	Δ
	Compare X to Memory (16-Bit)	DIR	9E dd	3								
		EXT	BE hh ll	3								
		IDX	AE xb	3								
		IDX1	AE xb ff	3								
		IDX2	AE xb ee ff	4								
		[D,IDX]	AE xb	6								
		[IDX2]	AE xb ee ff	6								
CPY *opr*	(Y) – (M:M+1)	IMM	8D jj kk	2	–	–	–	–	Δ	Δ	Δ	Δ
	Compare Y to Memory (16-Bit)	DIR	9D dd	3								
		EXT	BD hh ll	3								
		IDX	AD xb	3								
		IDX1	AD xb ff	3								
		IDX2	AD xb ee ff	4								
		[D,IDX]	AD xb	6								
		[IDX2]	AD xb ee ff	6								
DAA	Adjust Sum to BCD Decimal Adjust Accumulator A	INH	18 07	3	–	–	–	–	Δ	Δ	?	Δ
DBEQ *cntr, rel*	(cntr) – 1⇒ cntr if (cntr) = 0, then Branch else Continue to next instruction Decrement Counter and Branch if = 0 (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 lb rr	3	–	–	–	–	–	–	–	–
DBNE *cntr, rel*	(cntr) – 1 ⇒ cntr If (cntr) not = 0, then Branch; else Continue to next instruction Decrement Counter and Branch if ≠ 0 (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 lb rr	3	–	–	–	–	–	–	–	–

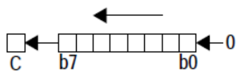
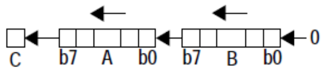
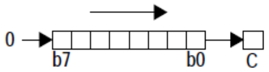
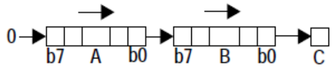


Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
DEC *opr* DECA DECB	(M) – $01 ⇒ M Decrement Memory Location (A) – $01 ⇒ A Decrement A (B) – $01 ⇒ B Decrement B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	73 hh ll 63 xb 63 xb ff 63 xb ee ff 63 xb 63 xb ee ff 43 53	4 3 4 5 6 6 1 1	–	–	–	–	Δ	Δ	Δ	–
DES	(SP) – $0001 ⇒ SP *Translates to* LEAS –1,SP	IDX	1B 9F	2	–	–	–	–	–	–	–	–
DEX	(X) – $0001 ⇒ X Decrement Index Register X	INH	09	1	–	–	–	–	–	Δ	–	–
DEY	(Y) – $0001 ⇒ Y Decrement Index Register Y	INH	03	1	–	–	–	–	–	Δ	–	–
EDIV	(Y:D) ÷ (X) ⇒ Y Remainder ⇒ D 32 × 16 Bit ⇒ 16 Bit Divide (unsigned)	INH	11	11	–	–	–	–	Δ	Δ	Δ	Δ
EDIVS	(Y:D) ÷ (X) ⇒ Y Remainder ⇒ D 32 × 16 Bit ⇒ 16 Bit Divide (signed)	INH	18 14	12	–	–	–	–	Δ	Δ	Δ	Δ
EMACS *sum*	$(M_{(X)}:M_{(X+1)}) \times (M_{(Y)}:M_{(Y+1)}) + (M\sim M+3) \Rightarrow M\sim M+3$ 16 × 16 Bit ⇒ 32 Bit Multiply and Accumulate (signed)	Special	18 12 hh ll	13	–	–	–	–	Δ	Δ	Δ	Δ
EMAXD *opr*	MAX((D), (M:M+1)) ⇒ D MAX of 2 Unsigned 16-Bit Values N, Z, V and C status bits reflect result of internal compare ((D) – (M:M+1))	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1A xb 18 1A xb ff 18 1A xb ee ff 18 1A xb 18 1A xb ee ff	4 4 5 7 7	–	–	–	–	Δ	Δ	Δ	Δ
EMAXM *opr*	MAX((D), (M:M+1)) ⇒ M:M+1 MAX of 2 Unsigned 16-Bit Values N, Z, V and C status bits reflect result of internal compare ((D) – (M:M+1))	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1E xb 18 1E xb ff 18 1E xb ee ff 18 1E xb 18 1E xb ee ff	4 5 6 7 7	–	–	–	–	Δ	Δ	Δ	Δ
EMIND *opr*	MIN((D), (M:M+1)) ⇒ D MIN of 2 Unsigned 16-Bit Values N, Z, V and C status bits reflect result of internal compare ((D) – (M:M+1))	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1B xb 18 1B xb ff 18 1B xb ee ff 18 1B xb 18 1B xb ee ff	4 4 5 7 7	–	–	–	–	Δ	Δ	Δ	Δ
EMINM *opr*	MIN((D), (M:M+1)) ⇒ M:M+1 MIN of 2 Unsigned 16-Bit Values N, Z, V and C status bits reflect result of internal compare ((D) – (M:M+1))	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1F xb 18 1F xb ff 18 1F xb ee ff 18 1F xb 18 1F xb ee ff	4 5 6 7 7	–	–	–	–	Δ	Δ	Δ	Δ
EMUL	(D) × (Y) ⇒ Y:D 16 × 16 Bit Multiply (unsigned)	INH	13	3	–	–	–	–	Δ	Δ	–	Δ
EMULS	(D) × (Y) ⇒ Y:D 16 × 16 Bit Multiply (signed)	INH	18 13	3	–	–	–	–	Δ	Δ	–	Δ

Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
EORA *opr*	(A) ⊕ (M) ⇒ A Exclusive-OR A with Memory	IMM	88 ii	1	–	–	–	–	Δ	Δ	0	–
		DIR	98 dd	3								
		EXT	B8 hh ll	3								
		IDX	A8 xb	3								
		IDX1	A8 xb ff	3								
		IDX2	A8 xb ee ff	4								
		[D,IDX]	A8 xb	6								
		[IDX2]	A8 xb ee ff	6								
EORB *opr*	(B) ⊕ (M) ⇒ B Exclusive-OR B with Memory	IMM	C8 ii	1	–	–	–	–	Δ	Δ	0	–
		DIR	D8 dd	3								
		EXT	F8 hh ll	3								
		IDX	E8 xb	3								
		IDX1	E8 xb ff	3								
		IDX2	E8 xb ee ff	4								
		[D,IDX]	E8 xb	6								
		[IDX2]	E8 xb ee ff	6								
ETBL *opr*	(M:M+1)+ [(B)×((M+2:M+3) – (M:M+1))] ⇒ D 16-Bit Table Lookup and Interpolate Initialize B, and index before ETBL. <ea> points at first table entry (M:M+1) and B is fractional part of lookup value (no indirect addr. modes allowed)	IDX	18 3F xb	10	–	–	–	–	Δ	Δ	–	?
EXG *r1, r2*	(r1) ⇔ (r2) (if r1 and r2 same size) *or* $00:(r1) ⇒ r2 (if r1=8-bit; r2=16-bit) *or* $(r1_{low}) \Leftrightarrow (r2)$ (if r1=16-bit; r2=8-bit) r1 and r2 may be A, B, CCR, D, X, Y, or SP	INH	B7 eb	1	–	–	–	–	–	–	–	–
FDIV	(D) ÷ (X) ⇒ X; r ⇒ D 16 × 16 Bit Fractional Divide	INH	18 11	12	–	–	–	–	–	Δ	Δ	Δ
IBEQ *cntr, rel*	(cntr) + 1⇒ cntr If (cntr) = 0, then Branch else Continue to next instruction Increment Counter and Branch if = 0 (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 lb rr	3	–	–	–	–	–	–	–	–
IBNE *cntr, rel*	(cntr) + 1⇒ cntr if (cntr) not = 0, then Branch; else Continue to next instruction Increment Counter and Branch if ≠ 0 (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 lb rr	3	–	–	–	–	–	–	–	–
IDIV	(D) ÷ (X) ⇒ X; r ⇒ D 16 × 16 Bit Integer Divide (unsigned)	INH	18 10	12	–	–	–	–	–	Δ	0	Δ
IDIVS	(D) ÷ (X) ⇒ X; r ⇒ D 16 × 16 Bit Integer Divide (signed)	INH	18 15	12	–	–	–	–	Δ	Δ	Δ	Δ

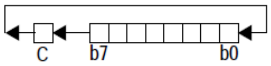
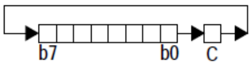
Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
INC *opr*	(M) + $01 ⇒ M	EXT	72 hh ll	4	–	–	–	–	Δ	Δ	Δ	–
	Increment Memory Byte	IDX	62 xb	3								
		IDX1	62 xb ff	4								
		IDX2	62 xb ee ff	5								
		[D,IDX]	62 xb	6								
		[IDX2]	62 xb ee ff	6								
INCA	(A) + $01 ⇒ A Increment Acc. A	INH	42	1								
INCB	(B) + $01 ⇒ B Increment Acc. B	INH	52	1								
INS	(SP) + $0001 ⇒ SP *Translates to* LEAS 1,SP	IDX	1B 81	2	–	–	–	–	–	–	–	–
INX	(X) + $0001 ⇒ X Increment Index Register X	INH	08	1	–	–	–	–	–	Δ	–	–
INY	(Y) + $0001 ⇒ Y Increment Index Register Y	INH	02	1	–	–	–	–	–	Δ	–	–
JMP *opr*	Subroutine address ⇒ PC	EXT	06 hh ll	3	–	–	–	–	–	–	–	–
		IDX	05 xb	3								
	Jump	IDX1	05 xb ff	3								
		IDX2	05 xb ee ff	4								
		[D,IDX]	05 xb	6								
		[IDX2]	05 xb ee ff	6								
JSR *opr*	(SP) – 2 ⇒ SP;	DIR	17 dd	4	–	–	–	–	–	–	–	–
	$\text{RTN}_H:\text{RTN}_L \Rightarrow M_{(SP)}:M_{(SP+1)}$;	EXT	16 hh ll	4								
	Subroutine address ⇒ PC	IDX	15 xb	4								
		IDX1	15 xb ff	4								
	Jump to Subroutine	IDX2	15 xb ee ff	5								
		[D,IDX]	15 xb	7								
		[IDX2]	15 xb ee ff	7								
LBCC *rel*	Long Branch if Carry Clear (if C = 0)	REL	18 24 qq rr	4/3	–	–	–	–	–	–	–	–
LBCS *rel*	Long Branch if Carry Set (if C = 1)	REL	18 25 qq rr	4/3	–	–	–	–	–	–	–	–
LBEQ *rel*	Long Branch if Equal (if Z = 1)	REL	18 27 qq rr	4/3	–	–	–	–	–	–	–	–
LBGE *rel*	Long Branch Greater Than or Equal (if N ⊕ V = 0) (signed)	REL	18 2C qq rr	4/3	–	–	–	–	–	–	–	–
LBGT *rel*	Long Branch if Greater Than (if Z + (N ⊕ V) = 0) (signed)	REL	18 2E qq rr	4/3	–	–	–	–	–	–	–	–
LBHI *rel*	Long Branch if Higher (if C + Z = 0) (unsigned)	REL	18 22 qq rr	4/3	–	–	–	–	–	–	–	–
LBHS *rel*	Long Branch if Higher or Same (if C = 0) (unsigned) same function as LBCC	REL	18 24 qq rr	4/3	–	–	–	–	–	–	–	–
LBLE *rel*	Long Branch if Less Than or Equal (if Z + (N ⊕ V) = 1) (signed)	REL	18 2F qq rr	4/3	–	–	–	–	–	–	–	–
LBLO *rel*	Long Branch if Lower (if C = 1) (unsigned) *same function as LBCS*	REL	18 25 qq rr	4/3	–	–	–	–	–	–	–	–
LBLS *rel*	Long Branch if Lower or Same (if C + Z = 1) (unsigned)	REL	18 23 qq rr	4/3	–	–	–	–	–	–	–	–
LBLT *rel*	Long Branch if Less Than (if N ⊕ V = 1) (signed)	REL	18 2D qq rr	4/3	–	–	–	–	–	–	–	–
LBMI *rel*	Long Branch if Minus (if N = 1)	REL	18 2B qq rr	4/3	–	–	–	–	–	–	–	–
LBNE *rel*	Long Branch if Not Equal (if Z = 0)	REL	18 26 qq rr	4/3	–	–	–	–	–	–	–	–
LBPL *rel*	Long Branch if Plus (if N = 0)	REL	18 2A qq rr	4/3	–	–	–	–	–	–	–	–
LBRA *rel*	Long Branch Always (if 1=1)	REL	18 20 qq rr	4	–	–	–	–	–	–	–	–

Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
LBRN *rel*	Long Branch Never (if 1 = 0)	REL	18 21 qq rr	3	–	–	–	–	–	–	–	–
LBVC *rel*	Long Branch if Overflow Bit Clear (if V=0)	REL	18 28 qq rr	4/3	–	–	–	–	–	–	–	–
LBVS *rel*	Long Branch if Overflow Bit Set (if V = 1)	REL	18 29 qq rr	4/3	–	–	–	–	–	–	–	–
LDAA *opr*	(M) ⇒ A	IMM	86 ii	1	–	–	–	–	Δ	Δ	0	–
	Load Accumulator A	DIR	96 dd	3								
		EXT	B6 hh ll	3								
		IDX	A6 xb	3								
		IDX1	A6 xb ff	3								
		IDX2	A6 xb ee ff	4								
		[D,IDX]	A6 xb	6								
		[IDX2]	A6 xb ee ff	6								
LDAB *opr*	(M) ⇒ B	IMM	C6 ii	1	–	–	–	–	Δ	Δ	0	–
	Load Accumulator B	DIR	D6 dd	3								
		EXT	F6 hh ll	3								
		IDX	E6 xb	3								
		IDX1	E6 xb ff	3								
		IDX2	E6 xb ee ff	4								
		[D,IDX]	E6 xb	6								
		[IDX2]	E6 xb ee ff	6								
LDD *opr*	(M:M+1) ⇒ A:B	IMM	CC jj kk	2	–	–	–	–	Δ	Δ	0	–
	Load Double Accumulator D (A:B)	DIR	DC dd	3								
		EXT	FC hh ll	3								
		IDX	EC xb	3								
		IDX1	EC xb ff	3								
		IDX2	EC xb ee ff	4								
		[D,IDX]	EC xb	6								
		[IDX2]	EC xb ee ff	6								
LDS *opr*	(M:M+1) ⇒ SP	IMM	CF jj kk	2	–	–	–	–	Δ	Δ	0	–
	Load Stack Pointer	DIR	DF dd	3								
		EXT	FF hh ll	3								
		IDX	EF xb	3								
		IDX1	EF xb ff	3								
		IDX2	EF xb ee ff	4								
		[D,IDX]	EF xb	6								
		[IDX2]	EF xb ee ff	6								
LDX *opr*	(M:M+1) ⇒ X	IMM	CE jj kk	2	–	–	–	–	Δ	Δ	0	–
	Load Index Register X	DIR	DE dd	3								
		EXT	FE hh ll	3								
		IDX	EE xb	3								
		IDX1	EE xb ff	3								
		IDX2	EE xb ee ff	4								
		[D,IDX]	EE xb	6								
		[IDX2]	EE xb ee ff	6								
LDY *opr*	(M:M+1) ⇒ Y	IMM	CD jj kk	2	–	–	–	–	Δ	Δ	0	–
	Load Index Register Y	DIR	DD dd	3								
		EXT	FD hh ll	3								
		IDX	ED xb	3								
		IDX1	ED xb ff	3								
		IDX2	ED xb ee ff	4								
		[D,IDX]	ED xb	6								
		[IDX2]	ED xb ee ff	6								
LEAS *opr*	Effective Address ⇒ SP	IDX	1B xb	2	–	–	–	–	–	–	–	–
	Load Effective Address into SP	IDX1	1B xb ff	2								
		IDX2	1B xb ee ff	2								

Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
LEAX *opr*	Effective Address ⇒ X	IDX	1A xb	2	–	–	–	–	–	–	–	–
	Load Effective Address into X	IDX1	1A xb ff	2								
		IDX2	1A xb ee ff	2								
LEAY *opr*	Effective Address ⇒ Y	IDX	19 xb	2	–	–	–	–	–	–	–	–
	Load Effective Address into Y	IDX1	19 xb ff	2								
		IDX2	19 xb ee ff	2								
LSL *opr*		EXT	78 hh ll	4	–	–	–	–	Δ	Δ	Δ	Δ
		IDX	68 xb	3								
		IDX1	68 xb ff	4								
	Logical Shift Left	IDX2	68 xb ee ff	5								
	same function as ASL	[D,IDX]	68 xb	6								
		[IDX2]	68 xb ee ff	6								
LSLA	Logical Shift Accumulator A to Left	INH	48	1								
LSLB	Logical Shift Accumulator B to Left	INH	58	1								
LSLD		INH	59	1	–	–	–	–	Δ	Δ	Δ	Δ
	Logical Shift Left D Accumulator											
	same function as ASLD											
LSR *opr*		EXT	74 hh ll	4	–	–	–	–	0	Δ	Δ	Δ
		IDX	64 xb	3								
		IDX1	64 xb ff	4								
	Logical Shift Right	IDX2	64 xb ee ff	5								
		[D,IDX]	64 xb	6								
		[IDX2]	64 xb ee ff	6								
LSRA	Logical Shift Accumulator A to Right	INH	44	1								
LSRB	Logical Shift Accumulator B to Right	INH	54	1								
LSRD		INH	49	1	–	–	–	–	0	Δ	Δ	Δ
	Logical Shift Right D Accumulator											
MAXA	MAX((A), (M)) ⇒ A	IDX	18 18 xb	4	–	–	–	–	Δ	Δ	Δ	Δ
	MAX of 2 Unsigned 8-Bit Values	IDX1	18 18 xb ff	4								
		IDX2	18 18 xb ee ff	5								
	N, Z, V and C status bits reflect result of	[D,IDX]	18 18 xb	7								
	internal compare ((A) – (M)).	[IDX2]	18 18 xb ee ff	7								
MAXM	MAX((A), (M)) ⇒ M	IDX	18 1C xb	4	–	–	–	–	Δ	Δ	Δ	Δ
	MAX of 2 Unsigned 8-Bit Values	IDX1	18 1C xb ff	5								
		IDX2	18 1C xb ee ff	6								
	N, Z, V and C status bits reflect result of	[D,IDX]	18 1C xb	7								
	internal compare ((A) – (M)).	[IDX2]	18 1C xb ee ff	7								
MEM	µ (grade) ⇒ $M_{(Y)}$; (X) + 4 ⇒ X; (Y) + 1 ⇒ Y; A unchanged if (A) < P1 or (A) > P2 then µ = 0, else µ = MIN[((A) – P1)×S1, (P2 – (A))×S2, $FF] where: A = current crisp input value; X points at 4-byte data structure that describes a trapezoidal membership function (P1, P2, S1, S2); Y points at fuzzy input (RAM location). See instruction details for special cases.	Special	01	5	–	–	?	–	?	?	?	?

Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
MINA	MIN((A), (M)) ⇒ A	IDX	18 19 xb	4	–	–	–	–	Δ	Δ	Δ	Δ
	MIN of Two Unsigned 8-Bit Values	IDX1	18 19 xb ff	4								
		IDX2	18 19 xb ee ff	5								
	N, Z, V and C status bits reflect result of	[D,IDX]	18 19 xb	7								
	internal compare ((A) – (M)).	[IDX2]	18 19 xb ee ff	7								
MINM	MIN((A), (M)) ⇒ M	IDX	18 1D xb	4	–	–	–	–	Δ	Δ	Δ	Δ
	MIN of Two Unsigned 8-Bit Values	IDX1	18 1D xb ff	5								
		IDX2	18 1D xb ee ff	6								
	N, Z, V and C status bits reflect result of	[D,IDX]	18 1D xb	7								
	internal compare ((A) – (M)).	[IDX2]	18 1D xb ee ff	7								
MOVB *opr1*, *opr2*	$(M_1) \Rightarrow M_2$	IMM-EXT	18 0B ii hh ll	4	–	–	–	–	–	–	–	–
	Memory to Memory Byte-Move (8-Bit)	IMM-IDX	18 08 xb ii	4								
		EXT-EXT	18 0C hh ll hh ll	6								
		EXT-IDX	18 09 xb hh ll	5								
		IDX-EXT	18 0D xb hh ll	5								
		IDX-IDX	18 0A xb xb	5								
MOVW *opr1*, *opr2*	$(M{:}M{+}1_1) \Rightarrow M{:}M{+}1_2$	IMM-EXT	18 03 jj kk hh ll	5	–	–	–	–	–	–	–	–
	Memory to Memory Word-Move (16-Bit)	IMM-IDX	18 00 xb jj kk	4								
		EXT-EXT	18 04 hh ll hh ll	6								
		EXT-IDX	18 01 xb hh ll	5								
		IDX-EXT	18 05 xb hh ll	5								
		IDX-IDX	18 02 xb xb	5								
MUL	(A) × (B) ⇒ A:B	INH	12	3	–	–	–	–	–	–	–	Δ
	8 × 8 Unsigned Multiply											
NEG *opr*	0 – (M) ⇒ M *or* $(\overline{M})$ + 1 ⇒ M	EXT	70 hh ll	4	–	–	–	–	Δ	Δ	Δ	Δ
	Two's Complement Negate	IDX	60 xb	3								
		IDX1	60 xb ff	4								
		IDX2	60 xb ee ff	5								
		[D,IDX]	60 xb	6								
		[IDX2]	60 xb ee ff	6								
NEGA	0 – (A) ⇒ A *equivalent to* $(\overline{A})$ + 1 ⇒ B	INH	40	1								
	Negate Accumulator A											
NEGB	0 – (B) ⇒ B *equivalent to* $(\overline{B})$ + 1 ⇒ B	INH	50	1								
	Negate Accumulator B											
NOP	No Operation	INH	A7	1	–	–	–	–	–	–	–	–
ORAA *opr*	(A) + (M) ⇒ A	IMM	8A ii	1	–	–	–	–	Δ	Δ	0	–
	Logical OR A with Memory	DIR	9A dd	3								
		EXT	BA hh ll	3								
		IDX	AA xb	3								
		IDX1	AA xb ff	3								
		IDX2	AA xb ee ff	4								
		[D,IDX]	AA xb	6								
		[IDX2]	AA xb ee ff	6								
ORAB *opr*	(B) + (M) ⇒ B	IMM	CA ii	1	–	–	–	–	Δ	Δ	0	–
	Logical OR B with Memory	DIR	DA dd	3								
		EXT	FA hh ll	3								
		IDX	EA xb	3								
		IDX1	EA xb ff	3								
		IDX2	EA xb ee ff	4								
		[D,IDX]	EA xb	6								
		[IDX2]	EA xb ee ff	6								
ORCC *opr*	(CCR) + M ⇒ CCR	IMM	14 ii	1	⇑	–	⇑	⇑	⇑	⇑	⇑	⇑
	Logical OR CCR with Memory											

Source Form	Operation	Addr. Mode	Machine Coding (hex)	$\sim^{*}$	S	X	H	I	N	Z	V	C
PSHA	$(SP) - 1 \Rightarrow SP; (A) \Rightarrow M_{(SP)}$ Push Accumulator A onto Stack	INH	36	2	–	–	–	–	–	–	–	–
PSHB	$(SP) - 1 \Rightarrow SP; (B) \Rightarrow M_{(SP)}$ Push Accumulator B onto Stack	INH	37	2	–	–	–	–	–	–	–	–
PSHC	$(SP) - 1 \Rightarrow SP; (CCR) \Rightarrow M_{(SP)}$ Push CCR onto Stack	INH	39	2	–	–	–	–	–	–	–	–
PSHD	$(SP) - 2 \Rightarrow SP; (A:B) \Rightarrow M_{(SP)}:M_{(SP+1)}$ Push D Accumulator onto Stack	INH	3B	2	–	–	–	–	–	–	–	–
PSHX	$(SP) - 2 \Rightarrow SP; (X_H:X_L) \Rightarrow M_{(SP)}:M_{(SP+1)}$ Push Index Register X onto Stack	INH	34	2	–	–	–	–	–	–	–	–
PSHY	$(SP) - 2 \Rightarrow SP; (Y_H:Y_L) \Rightarrow M_{(SP)}:M_{(SP+1)}$ Push Index Register Y onto Stack	INH	35	2	–	–	–	–	–	–	–	–
PULA	$(M_{(SP)}) \Rightarrow A; (SP) + 1 \Rightarrow SP$ Pull Accumulator A from Stack	INH	32	3	–	–	–	–	–	–	–	–
PULB	$(M_{(SP)}) \Rightarrow B; (SP) + 1 \Rightarrow SP$ Pull Accumulator B from Stack	INH	33	3	–	–	–	–	–	–	–	–
PULC	$(M_{(SP)}) \Rightarrow CCR; (SP) + 1 \Rightarrow SP$ Pull CCR from Stack	INH	38	3	Δ	⇓	Δ	Δ	Δ	Δ	Δ	Δ
PULD	$(M_{(SP)}:M_{(SP+1)}) \Rightarrow A:B; (SP) + 2 \Rightarrow SP$ Pull D from Stack	INH	3A	3	–	–	–	–	–	–	–	–
PULX	$(M_{(SP)}:M_{(SP+1)}) \Rightarrow X_H:X_L; (SP) + 2 \Rightarrow SP$ Pull Index Register X from Stack	INH	30	3	–	–	–	–	–	–	–	–
PULY	$(M_{(SP)}:M_{(SP+1)}) \Rightarrow Y_H:Y_L; (SP) + 2 \Rightarrow SP$ Pull Index Register Y from Stack	INH	31	3	–	–	–	–	–	–	–	–
REV	MIN-MAX rule evaluation Find smallest rule input (MIN). Store to rule outputs unless fuzzy output is already larger (MAX). For rule weights see REVW. Each rule input is an 8-bit offset from the base address in Y. Each rule output is an 8-bit offset from the base address in Y. $FE separates rule inputs from rule outputs. $FF terminates the rule list. REV may be interrupted.	Special	18 3A	3^{**} per rule byte	–	–	–	–	–	–	Δ	–

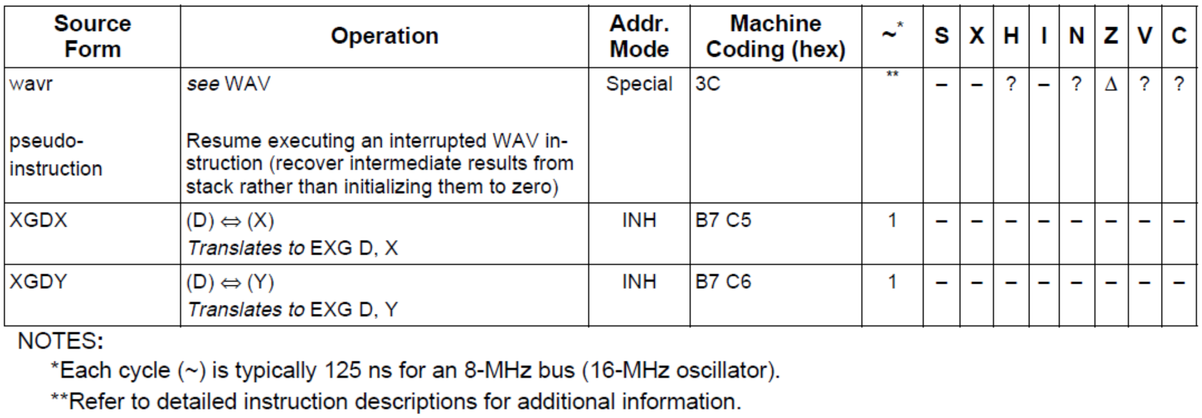
Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
REVW	MIN-MAX rule evaluation Find smallest rule input (MIN), Store to rule outputs unless fuzzy output is already larger (MAX). Rule weights supported, optional. Each rule input is the 16-bit address of a fuzzy input. Each rule output is the 16-bit address of a fuzzy output. The value $FFFE separates rule inputs from rule outputs. $FFFF terminates the rule list. REVW may be interrupted.	Special	18 3B	3^{**} per rule byte; 5 per wt.	–	–	?	–	?	?	Δ	!
ROL *opr*	 Rotate Memory Left through Carry	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	75 hh ll 65 xb 65 xb ff 65 xb ee ff 65 xb 65 xb ee ff	4 3 4 5 6 6	–	–	–	–	Δ	Δ	Δ	Δ
ROLA	Rotate A Left through Carry	INH	45	1								
ROLB	Rotate B Left through Carry	INH	55	1								
ROR *opr*	 Rotate Memory Right through Carry	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	76 hh ll 66 xb 66 xb ff 66 xb ee ff 66 xb 66 xb ee ff	4 3 4 5 6 6	–	–	–	–	Δ	Δ	Δ	Δ
RORA	Rotate A Right through Carry	INH	46	1								
RORB	Rotate B Right through Carry	INH	56	1								
RTC	$(M_{(SP)}) \Rightarrow PPAGE; (SP) + 1 \Rightarrow SP;$ $(M_{(SP)}:M_{(SP+1)}) \Rightarrow PC_H:PC_L;$ $(SP) + 2 \Rightarrow SP$ Return from Call	INH	0A	6	–	–	–	–	–	–	–	–
RTI	$(M_{(SP)}) \Rightarrow CCR; (SP) + 1 \Rightarrow SP$ $(M_{(SP)}:M_{(SP+1)}) \Rightarrow B:A; (SP) + 2 \Rightarrow SP$ $(M_{(SP)}:M_{(SP+1)}) \Rightarrow X_H:X_L; (SP) + 4 \Rightarrow SP$ $(M_{(SP)}:M_{(SP+1)}) \Rightarrow PC_H:PC_L; (SP) - 2 \Rightarrow SP$ $(M_{(SP)}:M_{(SP+1)}) \Rightarrow Y_H:Y_L;$ $(SP) + 4 \Rightarrow SP$ Return from Interrupt	INH	0B	8	Δ	⇓	Δ	Δ	Δ	Δ	Δ	Δ
RTS	$(M_{(SP)}:M_{(SP+1)}) \Rightarrow PC_H:PC_L;$ $(SP) + 2 \Rightarrow SP$ Return from Subroutine	INH	3D	5	–	–	–	–	–	–	–	–
SBA	$(A) - (B) \Rightarrow A$ Subtract B from A	INH	18 16	2	–	–	–	–	Δ	Δ	Δ	Δ

Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
SBCA *opr*	(A) – (M) – C ⇒ A Subtract with Borrow from A	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	82 ii 92 dd B2 hh ll A2 xb A2 xb ff A2 xb ee ff A2 xb A2 xb ee ff	1 3 3 3 3 4 6 6	–	–	–	–	Δ	Δ	Δ	Δ
SBCB *opr*	(B) – (M) – C ⇒ B Subtract with Borrow from B	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C2 ii D2 dd F2 hh ll E2 xb E2 xb ff E2 xb ee ff E2 xb E2 xb ee ff	1 3 3 3 3 4 6 6	–	–	–	–	Δ	Δ	Δ	Δ
SEC	1 ⇒ C *Translates to* ORCC #$01	IMM	14 01	1	–	–	–	–	–	–	–	1
SEI	1 ⇒ I; (inhibit I interrupts) *Translates to* ORCC #$10	IMM	14 10	1	–	–	–	1	–	–	–	–
SEV	1 ⇒ V *Translates to* ORCC #$02	IMM	14 02	1	–	–	–	–	–	–	1	–
SEX *r1, r2*	$00:(r1) ⇒ r2 if r1, bit 7 is 0 *or* $FF:(r1) ⇒ r2 if r1, bit 7 is 1 Sign Extend 8-bit r1 to 16-bit r2 r1 may be A, B, or CCR r2 may be D, X, Y, or SP *Alternate mnemonic for* TFR r1, r2	INH	B7 eb	1	–	–	–	–	–	–	–	–
STAA *opr*	(A) ⇒ M Store Accumulator A to Memory	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5A dd 7A hh ll 6A xb 6A xb ff 6A xb ee ff 6A xb 6A xb ee ff	2 3 2 3 3 5 5	–	–	–	–	Δ	Δ	0	–
STAB *opr*	(B) ⇒ M Store Accumulator B to Memory	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5B dd 7B hh ll 6B xb 6B xb ff 6B xb ee ff 6B xb 6B xb ee ff	2 3 2 3 3 5 5	–	–	–	–	Δ	Δ	0	–
STD *opr*	(A) ⇒ M, (B) ⇒ M+1 Store Double Accumulator	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5C dd 7C hh ll 6C xb 6C xb ff 6C xb ee ff 6C xb 6C xb ee ff	2 3 2 3 3 5 5	–	–	–	–	Δ	Δ	0	–

Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
STOP	$(SP) - 2 \Rightarrow SP$; $RTN_H{:}RTN_L \Rightarrow M_{(SP)}{:}M_{(SP+1)}$; $(SP) - 2 \Rightarrow SP$; $(Y_H{:}Y_L) \Rightarrow M_{(SP)}{:}M_{(SP+1)}$; $(SP) - 2 \Rightarrow SP$; $(X_H{:}X_L) \Rightarrow M_{(SP)}{:}M_{(SP+1)}$; $(SP) - 2 \Rightarrow SP$; $(B{:}A) \Rightarrow M_{(SP)}{:}M_{(SP+1)}$; $(SP) - 1 \Rightarrow SP$; $(CCR) \Rightarrow M_{(SP)}$; STOP All Clocks If S control bit = 1, the STOP instruction is disabled and acts like a two-cycle NOP. Registers stacked to allow quicker recovery by interrupt.	INH	18 3E	9^{**} +5 or $+2^{**}$	–	–	–	–	–	–	–	–
STS *opr*	$(SP_H{:}SP_L) \Rightarrow M{:}M+1$ Store Stack Pointer	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5F dd 7F hh ll 6F xb 6F xb ff 6F xb ee ff 6F xb 6F xb ee ff	2 3 2 3 3 5 5	–	–	–	–	Δ	Δ	0	–
STX *opr*	$(X_H{:}X_L) \Rightarrow M{:}M+1$ Store Index Register X	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5E dd 7E hh ll 6E xb 6E xb ff 6E xb ee ff 6E xb 6E xb ee ff	2 3 2 3 3 5 5	–	–	–	–	Δ	Δ	0	–
STY *opr*	$(Y_H{:}Y_L) \Rightarrow M{:}M+1$ Store Index Register Y	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5D dd 7D hh ll 6D xb 6D xb ff 6D xb ee ff 6D xb 6D xb ee ff	2 3 2 3 3 5 5	–	–	–	–	Δ	Δ	0	–
SUBA *opr*	$(A) - (M) \Rightarrow A$ Subtract Memory from Accumulator A	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	80 ii 90 dd B0 hh ll A0 xb A0 xb ff A0 xb ee ff A0 xb A0 xb ee ff	1 3 3 3 3 4 6 6	–	–	–	–	Δ	Δ	Δ	Δ
SUBB *opr*	$(B) - (M) \Rightarrow B$ Subtract Memory from Accumulator B	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C0 ii D0 dd F0 hh ll E0 xb E0 xb ff E0 xb ee ff E0 xb E0 xb ee ff	1 3 3 3 3 4 6 6	–	–	–	–	Δ	Δ	Δ	Δ

Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
SUBD *opr*	(D) – (M:M+1) ⇒ D	IMM	83 jj kk	2	–	–	–	–	Δ	Δ	Δ	Δ
	Subtract Memory from D (A:B)	DIR	93 dd	3								
		EXT	B3 hh ll	3								
		IDX	A3 xb	3								
		IDX1	A3 xb ff	3								
		IDX2	A3 xb ee ff	4								
		[D,IDX]	A3 xb	6								
		[IDX2]	A3 xb ee ff	6								
SWI	(SP) – 2 ⇒ SP; RTN$_H$:RTN$_L$ ⇒ M$_{(SP)}$:M$_{(SP+1)}$; (SP) – 2 ⇒ SP; (Y$_H$:Y$_L$) ⇒ M$_{(SP)}$:M$_{(SP+1)}$; (SP) – 2 ⇒ SP; (X$_H$:X$_L$) ⇒ M$_{(SP)}$:M$_{(SP+1)}$; (SP) – 2 ⇒ SP; (B:A) ⇒ M$_{(SP)}$:M$_{(SP+1)}$; (SP) – 1 ⇒ SP; (CCR) ⇒ M$_{(SP)}$ 1 ⇒ I; (SWI Vector) ⇒ PC Software Interrupt	INH	3F	9	–	–	–	1	–	–	–	–
TAB	(A) ⇒ B Transfer A to B	INH	18 0E	2	–	–	–	–	Δ	Δ	0	–
TAP	(A) ⇒ CCR *Translates to* TFR A , CCR	INH	B7 02	1	Δ	⇓	Δ	Δ	Δ	Δ	Δ	Δ
TBA	(B) ⇒ A Transfer B to A	INH	18 0F	2	–	–	–	–	Δ	Δ	0	–
TBEQ *cntr, rel*	If (cntr) = 0, then Branch; else Continue to next instruction Test Counter and Branch if Zero (cntr = A, B, D, X,Y, or SP)	REL (9-bit)	04 lb rr	3	–	–	–	–	–	–	–	–
TBL *opr*	(M) + [(B) × ((M+1) – (M))] ⇒ A 8-Bit Table Lookup and Interpolate Initialize B, and index before TBL. <ea> points at first 8-bit table entry (M) and B is fractional part of lookup value. (no indirect addressing modes allowed.)	IDX	18 3D xb	8	–	–	–	–	Δ	Δ	–	?
TBNE *cntr, rel*	If (cntr) not = 0, then Branch; else Continue to next instruction Test Counter and Branch if Not Zero (cntr = A, B, D, X,Y, or SP)	REL (9-bit)	04 lb rr	3	–	–	–	–	–	–	–	–
TFR *r1, r2*	(r1) ⇒ r2 *or* $00:(r1) ⇒ r2 *or* (r1[7:0]) ⇒ r2 Transfer Register to Register r1 and r2 may be A, B, CCR, D, X, Y, or SP	INH	B7 eb	1	– or Δ	– ⇓	– Δ	– Δ	– Δ	– Δ	– Δ	– Δ
TPA	(CCR) ⇒ A *Translates to* TFR CCR , A	INH	B7 20	1	–	–	–	–	–	–	–	–

Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
TRAP	$(SP) - 2 \Rightarrow SP$; $RTN_H{:}RTN_L \Rightarrow M_{(SP)}{:}M_{(SP+1)}$; $(SP) - 2 \Rightarrow SP$; $(Y_H{:}Y_L) \Rightarrow M_{(SP)}{:}M_{(SP+1)}$; $(SP) - 2 \Rightarrow SP$; $(X_H{:}X_L) \Rightarrow M_{(SP)}{:}M_{(SP+1)}$; $(SP) - 2 \Rightarrow SP$; $(B{:}A) \Rightarrow M_{(SP)}{:}M_{(SP+1)}$; $(SP) - 1 \Rightarrow SP$; $(CCR) \Rightarrow M_{(SP)}$ $1 \Rightarrow I$; (TRAP Vector) $\Rightarrow PC$ Unimplemented opcode trap	INH	18 tn tn = \$30–\$39 or \$40–\$FF	10	–	–	–	1	–	–	–	–
TST *opr*	(M) – 0 Test Memory for Zero or Minus	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	F7 hh ll E7 xb E7 xb ff E7 xb ee ff E7 xb E7 xb ee ff	3 3 3 4 6 6	–	–	–	–	Δ	Δ	0	0
TSTA	(A) – 0 Test A for Zero or Minus	INH	97	1								
TSTB	(B) – 0 Test B for Zero or Minus	INH	D7	1								
TSX	(SP) ⇒ X *Translates to* TFR SP,X	INH	B7 75	1	–	–	–	–	–	–	–	–
TSY	(SP) ⇒ Y *Translates to* TFR SP,Y	INH	B7 76	1	–	–	–	–	–	–	–	–
TXS	(X) ⇒ SP *Translates to* TFR X,SP	INH	B7 57	1	–	–	–	–	–	–	–	–
TYS	(Y) ⇒ SP *Translates to* TFR Y,SP	INH	B7 67	1	–	–	–	–	–	–	–	–
WAI	$(SP) - 2 \Rightarrow SP$; $RTN_H{:}RTN_L \Rightarrow M_{(SP)}{:}M_{(SP+1)}$; $(SP) - 2 \Rightarrow SP$; $(Y_H{:}Y_L) \Rightarrow M_{(SP)}{:}M_{(SP+1)}$; $(SP) - 2 \Rightarrow SP$; $(X_H{:}X_L) \Rightarrow M_{(SP)}{:}M_{(SP+1)}$; $(SP) - 2 \Rightarrow SP$; $(B{:}A) \Rightarrow M_{(SP)}{:}M_{(SP+1)}$; $(SP) - 1 \Rightarrow SP$; $(CCR) \Rightarrow M_{(SP)}$; WAIT for interrupt	INH	3E	8** (in) + 5 (int)	– or – or –	– – 1	– – –	– 1 1	– – –	– – –	– – –	– – –
WAV	$\sum_{i=1}^{B} S_i F_i \Rightarrow Y{:}D$ $\sum_{i=1}^{B} F_i \Rightarrow X$ Calculate Sum of Products and Sum of Weights for Weighted Average Calculation Initialize B, X, and Y before WAV. B specifies number of elements. X points at first element in S_i list. Y points at first element in F_i list. All S_i and F_i elements are 8-bits. If interrupted, six extra bytes of stack used for intermediate values	Special	18 3C	8** per lable	–	–	?	–	?	Δ	?	?



Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
wavr pseudo-instruction	*see* WAV Resume executing an interrupted WAV instruction (recover intermediate results from stack rather than initializing them to zero)	Special	3C	**	–	–	?	–	?	Δ	?	?
XGDX	(D) ⇔ (X) *Translates to* EXG D, X	INH	B7 C5	1	–	–	–	–	–	–	–	–
XGDY	(D) ⇔ (Y) *Translates to* EXG D, Y	INH	B7 C6	1	–	–	–	–	–	–	–	–

NOTES:

*Each cycle (~) is typically 125 ns for an 8-MHz bus (16-MHz oscillator).

**Refer to detailed instruction descriptions for additional information.