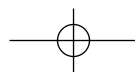


A

*68HC12 and HCS12
Instruction Set **

*Used with permission of Motorola, Inc.

A-1



Notation Used in Instruction Set Summary

Explanation of Italic Expressions in Source Form Column

<i>abc</i>	— A or B or CCR
<i>abcdxys</i>	— A or B or CCR or D or X or Y or SP. Some assemblers also allow T2 or T3.
<i>abd</i>	— A or B or D
<i>abdxys</i>	— A or B or D or X or Y or SP
<i>dxys</i>	— D or X or Y or SP
<i>msk8</i>	— 8-bit mask, some assemblers require # symbol before value
<i>opr8i</i>	— 8-bit immediate value
<i>opr16i</i>	— 16-bit immediate value
<i>opr8a</i>	— 8-bit address used with direct address mode
<i>opr16a</i>	— 16-bit address value
<i>opr0_xysp</i>	— Indexed addressing postbyte code:
<i>opr3,-xys</i>	Predecrement X or Y or SP by 1 . . . 8
<i>opr3,+xys</i>	Preincrement X or Y or SP by 1 . . . 8
<i>opr3,xys-</i>	Postdecrement X or Y or SP by 1 . . . 8
<i>opr3,xys+</i>	Postincrement X or Y or SP by 1 . . . 8
<i>opr5,xysp</i>	5-bit constant offset from X or Y or SP or PC
<i>abd,xysp</i>	Accumulator A or B or D offset from X or Y or SP or PC
<i>opr3</i>	— Any positive integer 1 . . . 8 for pre/post increment/decrement
<i>opr5</i>	— Any value in the range -16 . . . +15
<i>opr9</i>	— Any value in the range -256 . . . +255
<i>opr16</i>	— Any value in the range -32,768 . . . 65,535
<i>page</i>	— 8-bit value for PPAGE, some assemblers require # symbol before this value
<i>rel8</i>	— Label of branch destination within -256 to +255 locations
<i>rel9</i>	— Label of branch destination within -512 to +511 locations
<i>rel16</i>	— Any label within 64K memory space
<i>trapnum</i>	— Any 8-bit value in the range \$30-\$39 or \$40-\$FF
<i>xys</i>	— X or Y or SP
<i>xysp</i>	— X or Y or SP or PC

CPU12 REFERENCE GUIDE

Address Modes

IMM	— Immediate
IDX	— Indexed (no extension bytes) includes: 5-bit constant offset Pre/post increment/decrement by 1 . . . 8 Accumulator A, B, or D offset
IDX1	— 9-bit signed offset (1 extension byte)
IDX2	— 16-bit signed offset (2 extension bytes)
[D, IDX]	— Indexed indirect (accumulator D offset)
[IDX2]	— Indexed indirect (16-bit offset)
INH	— Inherent (no operands in object code)
REL	— 2's complement relative offset (branches)

Machine Coding

dd	— 8-bit direct address \$0000 to \$00FF. (High byte assumed to be \$00).
ee	— High-order byte of a 16-bit constant offset for indexed addressing.
eb	— Exchange/Transfer post-byte.
ff	— Low-order eight bits of a 9-bit signed constant offset for indexed addressing, or low-order byte of a 16-bit constant offset for indexed addressing.
hh	— High-order byte of a 16-bit extended address.
ii	— 8-bit immediate data value.
jj	— High-order byte of a 16-bit immediate data value.
kk	— Low-order byte of a 16-bit immediate data value.
lb	— Loop primitive (DBNE) post-byte.
ll	— Low-order byte of a 16-bit extended address.
mm	— 8-bit immediate mask value for bit manipulation instructions. Set bits indicate bits to be affected.
pg	— Program page (bank) number used in CALL instruction.
qq	— High-order byte of a 16-bit relative offset for long branches.
tn	— Trap number \$30–\$39 or \$40–\$FF.
rr	— Signed relative offset \$80 (–128) to \$7F (+127). Offset relative to the byte following the relative offset byte, or low-order byte of a 16-bit relative offset for long branches.
xb	— Indexed addressing post-byte.

Access Detail

Each code letter equals one CPU cycle. Uppercase = 16-bit operation and lowercase = 8-bit operation. For complex sequences see the *CPU12 Reference Manual (CPU12RM/AD)*.

- f — Free cycle, CPU doesn't use bus
- g — Read PPAGE internally
- I — Read indirect pointer (indexed indirect)
- i — Read indirect PPAGE value (call indirect)
- n — Write PPAGE internally
- O — Optional program word fetch (P) if instruction is misaligned and has an odd number of bytes of object code — otherwise, appears as a free cycle (f)
- P — Program word fetch (always an aligned word read)
- r — 8-bit data read
- R — 16-bit data read
- s — 8-bit stack write
- S — 16-bit stack write
- w — 8-bit data write
- W — 16-bit data write
- u — 8-bit stack read
- U — 16-bit stack read
- V — 16-bit vector fetch
- t — 8-bit conditional read (or free cycle)
- T — 16-bit conditional read (or free cycle)
- x — 8-bit conditional write

Special Cases

- PPP/P — Short branch, PPP if branch taken, P if not
- OPPP/OPO — Long branch, OPPP if branch taken, OPO if not

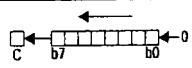
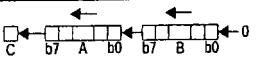
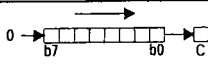
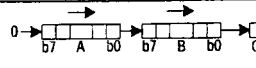
Condition Codes Columns

- — Status bit not affected by operation.
- 0 — Status bit cleared by operation.
- 1 — Status bit set by operation.
- Δ — Status bit affected by operation.
- ↓ — Status bit may be cleared or remain set, but is not set by operation.
- ↑ — Status bit may be set or remain cleared, but is not cleared by operation.
- ? — Status bit may be changed by operation but the final state is not defined.
- ! — Status bit used for a special purpose.

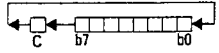
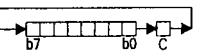
Source Form	Operation	Addr. Mode	Machine Coding (hex)	~*	S	X	H	I	N	Z	V	C
BVC <i>rel</i>	Branch if Overflow Bit Clear (if V = 0)	REL	28 rr	3/1	-	-	-	-	-	-	-	-
BVS <i>rel</i>	Branch if Overflow Bit Set (if V = 1)	REL	29 rr	3/1	-	-	-	-	-	-	-	-
CALL <i>opr, page</i>	(SP) - 2 ⇒ SP; RTN _H :RTN _L ⇒ M _(SP) :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 <i>opr,r</i>	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 <i>opr</i>	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 <i>opr</i>	(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 <i>opr</i>	(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	
DEC <i>opr</i>	(M) - \$01 ⇒ M Decrement Memory Location	EXT	73 hh ll	4	-	-	-	-	Δ	Δ	Δ	-	
		IDX	63 xb	3									
		IDX1	63 xb ff	4									
		IDX2	63 xb ee ff	5									
		[D,IDX]	63 xb	6									
		[IDX2]	63 xb ee ff	6									
DECA	(A) - \$01 ⇒ A Decrement A	INH	43	1									
DECB	(B) - \$01 ⇒ B Decrement B	INH	53	1									
DES	(SP) - \$0001 ⇒ SP <i>Translates to LEAS -1,SP</i>	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 <i>sum</i>	$(M_{(X)}:M_{(X+1)}) \times (M_{(Y)}:M_{(Y+1)}) + (M:-M+3) \Rightarrow M:-M+3$ 16 × 16 Bit ⇒ 32 Bit Multiply and Accumulate (signed)	Special	18 12 hh ll	13	-	-	-	-	Δ	Δ	Δ	Δ	
EMAXD <i>opr</i>	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	18 1A xb	4	-	-	-	-	Δ	Δ	Δ	Δ	
		IDX1	18 1A xb ff	4									
		IDX2	18 1A xb ee ff	5									
		[D,IDX]	18 1A xb	7									
[IDX2]	18 1A xb ee ff	7											
EMAXM <i>opr</i>	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	18 1E xb	4	-	-	-	-	Δ	Δ	Δ	Δ	
		IDX1	18 1E xb ff	5									
		IDX2	18 1E xb ee ff	6									
		[D,IDX]	18 1E xb	7									
[IDX2]	18 1E xb ee ff	7											
EMIND <i>opr</i>	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	18 1B xb	4	-	-	-	-	Δ	Δ	Δ	Δ	
		IDX1	18 1B xb ff	4									
		IDX2	18 1B xb ee ff	5									
		[D,IDX]	18 1B xb	7									
[IDX2]	18 1B xb ee ff	7											
EMINM <i>opr</i>	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	18 1F xb	4	-	-	-	-	Δ	Δ	Δ	Δ	
		IDX1	18 1F xb ff	5									
		IDX2	18 1F xb ee ff	6									
		[D,IDX]	18 1F xb	7									
[IDX2]	18 1F xb ee ff	7											
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 <i>opr</i>	(A) ⊕ (M) ⇒ A Exclusive-OR A with Memory	IMM	88 ii	1	-	-	-	-	Δ	Δ	0	-
		DIR	98 dd	3	-	-	-	-	Δ	Δ	0	-
		EXT	B8 hh ll	3	-	-	-	-	Δ	Δ	0	-
		IDX	A8 xb	3	-	-	-	-	Δ	Δ	0	-
		IDX1	A8 xb ff	3	-	-	-	-	Δ	Δ	0	-
		IDX2	A8 xb ee ff	4	-	-	-	-	Δ	Δ	0	-
		{D,IDX} {IDX2}	A8 xb A8 xb ee ff	6 6	-	-	-	-	Δ	Δ	0	-
EORB <i>opr</i>	(B) ⊕ (M) ⇒ B Exclusive-OR B with Memory	IMM	C8 ii	1	-	-	-	-	Δ	Δ	0	-
		DIR	D8 dd	3	-	-	-	-	Δ	Δ	0	-
		EXT	F8 hh ll	3	-	-	-	-	Δ	Δ	0	-
		IDX	E8 xb	3	-	-	-	-	Δ	Δ	0	-
		IDX1	E8 xb ff	3	-	-	-	-	Δ	Δ	0	-
		IDX2	E8 xb ee ff	4	-	-	-	-	Δ	Δ	0	-
		{D,IDX} {IDX2}	E8 xb E8 xb ee ff	6 6	-	-	-	-	Δ	Δ	0	-
ETBL <i>opr</i>	(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 <i>r1, r2</i>	(r1) ⇔ (r2) (if r1 and r2 same size) or \$00:(r1) ⇒ r2 (if r1=8-bit; r2=16-bit) or (r1 _{low}) ⇔ (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 <i>cntr, rel</i>	(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 <i>cntr, rel</i>	(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
LEAX <i>opr</i>	Effective Address \Rightarrow X Load Effective Address into X	IDX IDX1 IDX2	1A xb 1A xb ff 1A xb ee ff	2 2 2	-	-	-	-	-	-	-	-
LEAY <i>opr</i>	Effective Address \Rightarrow Y Load Effective Address into Y	IDX IDX1 IDX2	19 xb 19 xb ff 19 xb ee ff	2 2 2	-	-	-	-	-	-	-	-
LSL <i>opr</i>	 Logical Shift Left same function as ASL	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	78 hh ll 68 xb 68 xb ff 68 xb ee ff 68 xb 68 xb ee ff	4 3 4 5 6 6	-	-	-	-	Δ	Δ	Δ	Δ
LSLA LSLB	Logical Shift Accumulator A to Left Logical Shift Accumulator B to Left	INH INH	48 58	1 1	-	-	-	-	-	-	-	-
LSLD	 Logical Shift Left D Accumulator same function as ASLD	INH	59	1	-	-	-	-	Δ	Δ	Δ	Δ
LSR <i>opr</i>	 Logical Shift Right	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	74 hh ll 64 xb 64 xb ff 64 xb ee ff 64 xb 64 xb ee ff	4 3 4 5 6 6	-	-	-	-	0	Δ	Δ	Δ
LSRA LSRB	Logical Shift Accumulator A to Right Logical Shift Accumulator B to Right	INH INH	44 54	1 1	-	-	-	-	-	-	-	-
LSRD	 Logical Shift Right D Accumulator	INH	49	1	-	-	-	-	0	Δ	Δ	Δ
MAXA	MAX((A), (M)) \Rightarrow A MAX of 2 Unsigned 8-Bit Values N, Z, V and C status bits reflect result of internal compare ((A) - (M)).	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 18 xb 18 18 xb ff 18 18 xb ee ff 18 18 xb 18 18 xb ee ff	4 4 5 7 7	-	-	-	-	Δ	Δ	Δ	Δ
MAXM	MAX((A), (M)) \Rightarrow M MAX of 2 Unsigned 8-Bit Values N, Z, V and C status bits reflect result of internal compare ((A) - (M)).	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1C xb 18 1C xb ff 18 1C xb ee ff 18 1C xb 18 1C xb ee ff	4 5 6 7 7	-	-	-	-	Δ	Δ	Δ	Δ
MEM	μ (grade) \Rightarrow M _(Y) ; (X) + 4 \Rightarrow X; (Y) + 1 \Rightarrow Y; A unchanged if (A) < P1 or (A) > P2 then $\mu = 0$, else $\mu = \text{MIN}(((A) - P1) \times S1, (P2 - (A)) \times S2, \$FF)$ 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
PSHA	(SP) - 1 ⇒ SP; (A) ⇒ M _(SP) Push Accumulator A onto Stack	INH	36	2	-	-	-	-	-	-	-	-
PSHB	(SP) - 1 ⇒ SP; (B) ⇒ M _(SP) Push Accumulator B onto Stack	INH	37	2	-	-	-	-	-	-	-	-
PSHC	(SP) - 1 ⇒ SP; (CCR) ⇒ M _(SP) Push CCR onto Stack	INH	39	2	-	-	-	-	-	-	-	-
PSHD	(SP) - 2 ⇒ SP; (A:B) ⇒ M _(SP) :M _(SP+1) Push D Accumulator onto Stack	INH	3B	2	-	-	-	-	-	-	-	-
PSHX	(SP) - 2 ⇒ SP; (X _H :X _L) ⇒ M _(SP) :M _(SP+1) Push Index Register X onto Stack	INH	34	2	-	-	-	-	-	-	-	-
PSHY	(SP) - 2 ⇒ SP; (Y _H :Y _L) ⇒ M _(SP) :M _(SP+1) Push Index Register Y onto Stack	INH	35	2	-	-	-	-	-	-	-	-
PULA	M _(SP) ⇒ A; (SP) + 1 ⇒ SP Pull Accumulator A from Stack	INH	32	3	-	-	-	-	-	-	-	-
PULB	M _(SP) ⇒ B; (SP) + 1 ⇒ SP Pull Accumulator B from Stack	INH	33	3	-	-	-	-	-	-	-	-
PULC	M _(SP) ⇒ CCR; (SP) + 1 ⇒ SP Pull CCR from Stack	INH	38	3	Δ	↓	Δ	Δ	Δ	Δ	Δ	Δ
PULD	M _(SP) :M _(SP+1) ⇒ A:B; (SP) + 2 ⇒ SP Pull D from Stack	INH	3A	3	-	-	-	-	-	-	-	-
PULX	M _(SP) :M _(SP+1) ⇒ X _H :X _L ; (SP) + 2 ⇒ SP Pull Index Register X from Stack	INH	30	3	-	-	-	-	-	-	-	-
PULY	M _(SP) :M _(SP+1) ⇒ Y _H :Y _L ; (SP) + 2 ⇒ 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
REWV	<p>MIN-MAX rule evaluation Find smallest rule input (MIN), Store to rule outputs unless fuzzy output is already larger (MAX).</p> <p>Rule weights supported, optional.</p> <p>Each rule input is the 16-bit address of a fuzzy input. Each rule output is the 16-bit ad- dress of a fuzzy output. The value \$FFFE separates rule inputs from rule outputs. \$FFFF terminates the rule list.</p> <p>REWV may be interrupted.</p>	Special	18 3B	3** per rule byte; 5 per wt.	-	-	?	-	?	?	Δ	!
ROL <i>opr</i>	 <p>Rotate Memory Left through Carry</p>	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 ROLB	<p>Rotate A Left through Carry Rotate B Left through Carry</p>	INH INH	45 55	1 1	-	-	-	-	-	-	-	-
ROR <i>opr</i>	 <p>Rotate Memory Right through Carry</p>	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 RORB	<p>Rotate A Right through Carry Rotate B Right through Carry</p>	INH INH	46 56	1 1	-	-	-	-	-	-	-	-
RTC	<p>$(M_{(SP)}) \Rightarrow PPAGE; (SP) + 1 \Rightarrow SP;$ $(M_{(SP)}; M_{(SP+1)}) \Rightarrow PC_H; PC_L;$ $(SP) + 2 \Rightarrow SP$</p> <p>Return from Call</p>	INH	0A	6	-	-	-	-	-	-	-	-
RTI	<p>$(M_{(SP)}) \Rightarrow CCR; (SP) + 1 \Rightarrow SP$ $(M_{(SP)}; M_{(SP+1)}) \Rightarrow B; A; (SP) + 2 \Rightarrow SP$ $(M_{(SP)}; M_{(SP+1)}) \Rightarrow X_H; X_L; (SP) + 4 \Rightarrow SP$ $(M_{(SP)}; M_{(SP+1)}) \Rightarrow PC_H; PC_L; (SP) - 2 \Rightarrow SP$ $(M_{(SP)}; M_{(SP+1)}) \Rightarrow Y_H; Y_L;$ $(SP) + 4 \Rightarrow SP$</p> <p>Return from Interrupt</p>	INH	0B	8	Δ	↓	Δ	Δ	Δ	Δ	Δ	Δ
RTS	<p>$(M_{(SP)}; M_{(SP+1)}) \Rightarrow PC_H; PC_L;$ $(SP) + 2 \Rightarrow SP$</p> <p>Return from Subroutine</p>	INH	3D	5	-	-	-	-	-	-	-	-
SBA	<p>$(A) - (B) \Rightarrow A$ Subtract B from A</p>	INH	18 16	2	-	-	-	-	Δ	Δ	Δ	Δ

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 <i>opr</i>	$(SP_H:SP_L) \Rightarrow M:M+1$ 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 <i>opr</i>	$(X_H:X_L) \Rightarrow M:M+1$ 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 <i>opr</i>	$(Y_H:Y_L) \Rightarrow M:M+1$ 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 <i>opr</i>	$(A) - (M) \Rightarrow A$ 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 <i>opr</i>	$(B) - (M) \Rightarrow B$ 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
TRAP	(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; (TRAP Vector) ⇒ PC Unimplemented opcode trap	INH	18 tn tn = \$30-\$39 or \$40-\$FF	10	-	-	-	1	-	-	-	-
TST <i>opr</i>	(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 <i>Translates to TFR SP,X</i>	INH	B7 75	1	-	-	-	-	-	-	-	-
TSY	(SP) ⇒ Y <i>Translates to TFR SP,Y</i>	INH	B7 76	1	-	-	-	-	-	-	-	-
TXS	(X) ⇒ SP <i>Translates to TFR X,SP</i>	INH	B7 57	1	-	-	-	-	-	-	-	-
TYS	(Y) ⇒ SP <i>Translates to TFR Y,SP</i>	INH	B7 67	1	-	-	-	-	-	-	-	-
WAI	(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) ; WAIT for interrupt	INH	3E	8* (in) + 5 (int)	-	-	-	-	-	-	-	-
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	<i>see WAV</i>	Special	3C	**	-	-	?	-	?	Δ	?	?
pseudo-instruction	Resume executing an interrupted WAV instruction (recover intermediate results from stack rather than initializing them to zero)											
XGDX	(D) ⇔ (X) <i>Translates to EXG D, X</i>	INH	B7 C5	1	-	-	-	-	-	-	-	-
XGDY	(D) ⇔ (Y) <i>Translates to EXG D, Y</i>	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.