

Notation Used in Instruction Set Summary

CPU Register Notation

Accumulator A — A or a
 Accumulator B — B or b
 Accumulator D — D or d
 Index Register X — X or x
 Index Register Y — Y or y
 Stack Pointer — SP, sp, or s
 Program Counter — PC, pc, or p
 Condition Code Register — CCR or c

Explanation of Italic Expressions in Source Form Column

abc — A or B or CCR
abcdxys — A or B or CCR or D or X or Y or SP
abdxys — A or B or D or X or Y or SP
msk8 — 8-bit mask (some assemblers require *#msk8*)
opr8i — 8-bit immediate value
opr16i — 16-bit immediate value
opr8a — 8-bit address used with direct address mode
opr16a — 16-bit address value
opr0_xysp — Indexed addressing postbyte code:
 opr3,-xys Predecrement X or Y or SP by 1 . . . 8
 opr3,+xys Preincrement X or Y or SP by 1 . . . 8
 opr3,xys- Postdecrement X or Y or SP by 1 . . . 8
 opr3,xys+ Postincrement X or Y or SP by 1 . . . 8
 opr5,xysp 5-bit constant offset from X or Y or SP or PC
 abd,xysp Acc. A or B or D offset from X or Y or SP or PC
opr3 — Any positive integer 1 . . . 8 for pre/post increment/decrement
opr5 — Any integer in the range -16 . . . +15
opr9 — Any integer in the range -256 . . . +255
opr16 — Any integer in the range -32,768 . . . 65,535
re9 — Label of branch destination within -512 to +511 locations
xys — X or Y or SP
xysp — X or Y or SP or PC

Address Mode Notation

INH — Inherent; no operands in object code
 IMM — Immediate; operand in object code
 DIR — Direct; operand is the lower byte of an addr. from \$0000 to \$00FF
 EXT — Operand is a 16-bit address
 REL — Two's complement relative offset; for branch instructions
 IDX — Indexed (no extension bytes); includes:
 5-bit constant offset from X, Y, SP, or PC
 Pre/post increment/decrement by 1 . . . 8
 Accumulator A, B, or D offset
 IDX1 — 9-bit signed offset from X, Y, SP, or PC; 1 extension byte
 IDX2 — 16-bit signed offset from X, Y, SP, or PC; 2 extension bytes
 [IDX2] — Indexed-indirect; 16-bit offset from X, Y, SP, or PC
 [D, IDX] — Indexed-indirect; accumulator D offset from X, Y, SP, or PC

Machine Coding


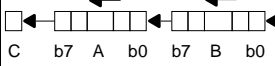

dd — 8-bit direct address \$0000 to \$00FF (high byte assumed to be \$00).
 ee — High-order byte of a 16-bit constant offset for indexed addressing.
 eb — Exchange/Transfer post-byte.
 ff — Low-order 8 bits of a 9-bit signed constant offset for indexed addr.,
 or low-order byte of a 16-bit constant offset for indexed address.
 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.
 rr — Signed relative offset \$80 (-128) to \$7F (+127). Offset relative to
 the byte following the relative offset byte.
 xb — Indexed addressing post-byte.

Operators

+ — Addition
 - — Subtraction
 & — Logical AND
 + — Logical OR (inclusive)
 ⊕ — Logical exclusive OR
 × — Multiplication
 ÷ — Division
 \bar{M} — Negation. One's complement (invert each bit of M)
 : — Concatenate
 ⇒ — Transfer
 ⇔ — Exchange

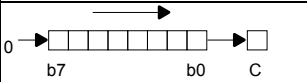
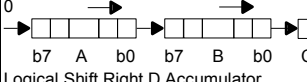
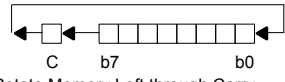
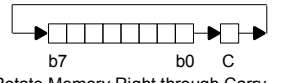
Condition Codes Columns

- — Status bit not affected by operation.
 0 — Status bit cleared by operation.
 1 — Status bit set by operation.
 Δ — Status bit affected by operation.
 ↓ — Stat. 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

Source Form	Operation	Addr. Mode	Machine Code (hex)	Exec. T (clk)	S	X	H	I	N	Z	V	C	Source Form	Operation	Addr. Mode	Machine Code (hex)	Exec. T (clk)	S	X	H	I	N	Z	V	C		
ABA	(A) + (B) ⇒ A Add Accumulators A and B	INH	18 06	2	--	Δ	Δ	Δ	Δ	Δ	Δ	Δ	ASL <i>opr16a</i> ASL <i>opr0_xysp</i> ASL <i>opr9_xysp</i> ASL <i>opr16_xysp</i> ASL [D, <i>xysp</i>] ASL [<i>opr16_xysp</i>]		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	Δ	Δ	Δ		
ABX	(B) + (X) ⇒ X Translates to LEAX B,X	IDX	1A E5	2	---	---	---	---	---	---	---	---	ASLA ASLB	Arithmetic Shift Left Accumulator A Arithmetic Shift Left Accumulator B	INH INH	48 58	1 1	---	---	---	---	---	---	---	---		
ABY	(B) + (Y) ⇒ Y Translates to LEAY B,Y	IDY	19 ED	2	---	---	---	---	---	---	---	---	ASLD		INH	59	1	---	---	---	---	---	---	---	---	---	---
ADCA # <i>opr8i</i> ADCA <i>opr8a</i> ADCA <i>opr16a</i> ADCA <i>opr0_xysp</i> ADCA <i>opr9_xysp</i> ADCA <i>opr16_xysp</i> ADCA [D, <i>xysp</i>] ADCA [<i>opr16_xysp</i>]	(A) + (M) + C ⇒ A Add with Carry to A	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	89 ii 99 dd B9 hh ll A9 xb A9 xb ff A9 xb ee ff A9 xb A9 xb ee ff	1 3 3 3 3 4 6 6	--	Δ	Δ	Δ	Δ	Δ	Δ	Δ	ASR <i>opr16a</i> ASR <i>opr0_xysp</i> ASR <i>opr9_xysp</i> ASR <i>opr16_xysp</i> ASR [D, <i>xysp</i>] ASR [<i>opr16_xysp</i>] ASRA ASRB		EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	77 hh ll 67 xb 67 xb ff 67 xb ee ff 67 xb 67 xb ee ff 47 57	4 3 4 5 6 6 1 1	---	Δ	Δ	Δ	Δ	Δ	Δ	Δ		
ADCB # <i>opr8i</i> ADCB <i>opr8a</i> ADCB <i>opr16a</i> ADCB <i>opr0_xysp</i> ADCB <i>opr9_xysp</i> ADCB <i>opr16_xysp</i> ADCB [D, <i>xysp</i>] ADCB [<i>opr16_xysp</i>]	(B) + (M) + C ⇒ B Add with Carry to B	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C9 ii D9 dd F9 hh ll E9 xb E9 xb ff E9 xb ee ff E9 xb E9 xb ee ff	1 3 3 3 3 4 6 6	--	Δ	Δ	Δ	Δ	Δ	Δ	BCC <i>rel8</i>	Branch if Carry Clear (if C = 0)	REL	24 rr	3/1 ¹⁾	---	---	---	---	---	---	---	---	---		
ADDA # <i>opr8i</i> ADDA <i>opr8a</i> ADDA <i>opr16a</i> ADDA <i>opr0_xysp</i> ADDA <i>opr9_xysp</i> ADDA <i>opr16_xysp</i> ADDA [D, <i>xysp</i>] ADDA [<i>opr16_xysp</i>]	(A) + (M) ⇒ A Add without Carry to A	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	8B ii 9B dd BB hh ll AB xb AB xb ff AB xb ee ff AB xb AB xb ee ff	1 3 3 3 3 4 6 6	--	Δ	Δ	Δ	Δ	Δ	Δ	Δ	BCLR <i>opr8a</i> , msk8 BCLR <i>opr16a</i> , msk8 BCLR <i>opr0_xysp</i> , msk8 BCLR <i>opr9_xysp</i> , msk8 BCLR <i>opr16_xysp</i> , msk8	(M) • (mm) ⇒ M Clear Bit(s) in Memory	DIR EXT IDX IDX1 IDX2	4D dd mm 1D hh ll mm 0D xb mm 0D xb ff mm 0D xb ee ff mm	4 4 4 4 6	---	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	
ADDB # <i>opr8i</i> ADDB <i>opr8a</i> ADDB <i>opr16a</i> ADDB <i>opr0_xysp</i> ADDB <i>opr9_xysp</i> ADDB <i>opr16_xysp</i> ADDB [D, <i>xysp</i>] ADDB [<i>opr16_xysp</i>]	(B) + (M) ⇒ B Add without Carry to B	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CB ii DB dd FB hh ll EB xb EB xb ff EB xb ee ff EB xb EB xb ee ff	1 3 3 3 3 4 6 6	--	Δ	Δ	Δ	Δ	Δ	Δ	Δ	BCS <i>rel8</i>	Branch if Carry Set (if C = 1)	REL	25 rr	3/1 ¹⁾	---	---	---	---	---	---	---	---	---	
ADDD # <i>opr16i</i> ADDD <i>opr8a</i> ADDD <i>opr16a</i> ADDD <i>opr0_xysp</i> ADDD <i>opr9_xysp</i> ADDD <i>opr16_xysp</i> ADDD [D, <i>xysp</i>] ADDD [<i>opr16_xysp</i>]	(A:B) + (M:(M+1)) ⇒ A:B Add 16-Bit to D (A:B)	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C3 jj kk D3 dd F3 hh ll E3 xb E3 xb ff E3 xb ee ff E3 xb E3 xb ee ff	2 3 3 3 3 4 6 6	---	---	---	---	---	---	---	---	BEG <i>rel8</i>	Branch if Equal (if Z = 1)	REL	27 rr	3/1 ¹⁾	---	---	---	---	---	---	---	---	---	
ANDB # <i>opr8i</i> ANDB <i>opr8a</i> ANDB <i>opr16a</i> ANDB <i>opr0_xysp</i> ANDB <i>opr9_xysp</i> ANDB <i>opr16_xysp</i> ANDB [D, <i>xysp</i>] ANDB [<i>opr16_xysp</i>]	(B) • (M) ⇒ B Logical AND B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C4 ii D4 dd F4 hh ll E4 xb E4 xb ff E4 xb ee ff E4 xb E4 xb ee ff	1 3 3 3 3 4 6 6	---	---	---	---	---	---	---	---	BGT <i>rel8</i>	Branch if Greater Than (if Z + (N ⊕ V) = 0) (signed)	REL	2E rr	3/1 ¹⁾	---	---	---	---	---	---	---	---	---	
ANDCC # <i>opr8i</i>	(CCR) • (M) ⇒ CCR Logical AND CCR with Memory	IMM	10 ii	1	↓	↓	↓	↓	↓	↓	↓	↓	BITA # <i>opr8i</i> BITA <i>opr8a</i> BITA <i>opr16a</i> BITA <i>opr0_xysp</i> BITA <i>opr9_xysp</i> BITA <i>opr16_xysp</i> BITA [D, <i>xysp</i>] BITA [<i>opr16_xysp</i>]	(A) • (M) Logical AND A with Memory Does not change Accumulator or 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	Δ	Δ	Δ	Δ	Δ		
ANDB # <i>opr8i</i> ANDB <i>opr8a</i> ANDB <i>opr16a</i> ANDB <i>opr0_xysp</i> ANDB <i>opr9_xysp</i> ANDB <i>opr16_xysp</i> ANDB [D, <i>xysp</i>] ANDB [<i>opr16_xysp</i>]	(B) • (M) ⇒ B Logical AND B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C4 ii D4 dd F4 hh ll E4 xb E4 xb ff E4 xb ee ff E4 xb E4 xb ee ff	1 3 3 3 3 4 6 6	---	---	---	---	---	---	---	---	BITB # <i>opr8i</i> BITB <i>opr8a</i> BITB <i>opr16a</i> BITB <i>opr0_xysp</i> BITB <i>opr9_xysp</i> BITB <i>opr16_xysp</i> BITB [D, <i>xysp</i>] BITB [<i>opr16_xysp</i>]	(B) • (M) Logical AND B with Memory Does not change Accumulator or 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	Δ	Δ	Δ	Δ	Δ		
ANDB # <i>opr8i</i> ANDB <i>opr8a</i> ANDB <i>opr16a</i> ANDB <i>opr0_xysp</i> ANDB <i>opr9_xysp</i> ANDB <i>opr16_xysp</i> ANDB [D, <i>xysp</i>] ANDB [<i>opr16_xysp</i>]	(B) • (M) ⇒ B Logical AND B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C4 ii D4 dd F4 hh ll E4 xb E4 xb ff E4 xb ee ff E4 xb E4 xb ee ff	1 3 3 3 3 4 6 6	---	---	---	---	---	---	---	---	BLE <i>rel8</i>	Branch if Less Than or Equal (if Z + (N ⊕ V) = 1) (signed)	REL	2F rr	3/1 ¹⁾	---	---	---	---	---	---	---	---	---	
ANDB # <i>opr8i</i> ANDB <i>opr8a</i> ANDB <i>opr16a</i> ANDB <i>opr0_xysp</i> ANDB <i>opr9_xysp</i> ANDB <i>opr16_xysp</i> ANDB [D, <i>xysp</i>] ANDB [<i>opr16_xysp</i>]	(B) • (M) ⇒ B Logical AND B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C4 ii D4 dd F4 hh ll E4 xb E4 xb ff E4 xb ee ff E4 xb E4 xb ee ff	1 3 3 3 3 4 6 6	---	---	---	---	---	---	---	---	BLO <i>rel8</i>	Branch if Lower (if C = 1) (unsigned) same as BCS	REL	25 rr	3/1 ¹⁾	---	---	---	---	---	---	---	---	---	
ANDB # <i>opr8i</i> ANDB <i>opr8a</i> ANDB <i>opr16a</i> ANDB <i>opr0_xysp</i> ANDB <i>opr9_xysp</i> ANDB <i>opr16_xysp</i> ANDB [D, <i>xysp</i>] ANDB [<i>opr16_xysp</i>]	(B) • (M) ⇒ B Logical AND B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C4 ii D4 dd F4 hh ll E4 xb E4 xb ff E4 xb ee ff E4 xb E4 xb ee ff	1 3 3 3 3 4 6 6	---	---	---	---	---	---	---	---	BLS <i>rel8</i>	Branch if Lower or Same (if C + Z = 1) (unsigned)	REL	25 rr	3/1 ¹⁾	---	---	---	---	---	---	---	---	---	---
ANDB # <i>opr8i</i> ANDB <i>opr8a</i> ANDB <i>opr16a</i> ANDB <i>opr0_xysp</i> ANDB <i>opr9_xysp</i> ANDB <i>opr16_xysp</i> ANDB [D, <i>xysp</i>] ANDB [<i>opr16_xysp</i>]	(B) • (M) ⇒ B Logical AND B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C4 ii D4 dd F4 hh ll E4 xb E4 xb ff E4 xb ee ff E4 xb E4 xb ee ff	1 3 3 3 3 4 6 6	---	---	---	---	---	---	---	---	BLT <i>rel8</i>	Branch if Less Than (if N ⊕ V = 1) (signed)	REL	2D rr	3/1 ¹⁾	---	---	---	---	---	---	---	---	---	---
ANDCC # <i>opr8i</i>	(CCR) • (M) ⇒ CCR Logical AND CCR with Memory	IMM	10 ii	1	↓	↓	↓	↓	↓	↓	↓	↓	BMI <i>rel8</i> BNE <i>rel8</i>	Branch if Minus (if N = 1) Branch if Not Equal (if Z = 0)	REL REL	2B rr 26 rr	3/1 ¹⁾ 3/1 ¹⁾	---	---	---	---	---	---	---	---	---	

¹⁾ 3/1 indicates this instruction takes 3 cycles to refill the instruct. queue if the branch is taken and 1 cycle if the branch is not taken.

Source Form	Operation	Addr. Mode	Machine Code (hex)	Exec. T (clk)	S	X	H	I	N	Z	V	C	Source Form	Operation	Addr. Mode	Machine Code (hex)	Exec. T (clk)	S	X	H	I	N	Z	V	C
EORB #opr8i EORB opr8a EORB opr16a EORB oprx0_xysp EORB oprx9_xysp EORB oprx16_xysp EORB [D,xysp] EORB [oprx16_xysp]	(B) ⊕ (M) ⇒ B Exclusive-OR B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C8 ii D8 dd F8 hh ll E8 xb E8 xb ff E8 xb ee ff E8 xb	1 3 3 3 3 4 6 6	-	-	-	-	Δ	Δ	0	-	LDAB #opr8i LDAB opr8a LDAB opr16a LDAB oprx0_xysp LDAB oprx9_xysp LDAB oprx16_xysp LDAB [D,xysp] LDAB [oprx16_xysp]	(M) ⇒ B Load Accumulator B	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C6 ii D6 dd F6 hh ll E6 xb E6 xb ff E6 xb ee ff E6 xb ee ff	1 3 3 3 3 4 6 6	-	-	-	-	Δ	Δ	0	-
EXG abcdxys,abcdxys	(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	-	-	-	-	-	-	-	-	LDD #opr16i LDD opr8a LDD opr16a LDD oprx0_xysp LDD oprx9_xysp LDD oprx16_xysp LDD [D,xysp] LDD [oprx16_xysp]	(M:M+1) ⇒ A:B Load Double Accumulator D (A:B)	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CC jj kk DC dd FC hh ll EC xb EC xb ff EC xb ee ff EC xb ee ff	2 3 3 3 3 4 6 6	-	-	-	-	Δ	Δ	0	-
FDIV	(D) ÷ (X) ⇒ X; Remainder ⇒ D 16 by 16 Bit Fractional Divide	INH	18 11	12	-	-	-	-	Δ	Δ	Δ	Δ	LDD #opr16i LDD opr8a LDD opr16a LDD oprx0_xysp LDD oprx9_xysp LDD oprx16_xysp LDD [D,xysp] LDD [oprx16_xysp]	(M:M+1) ⇒ SP Load Stack Pointer	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CF jj kk DF dd FF hh ll EF xb EF xb ff EF xb ee ff EF xb ee ff	2 3 3 3 3 4 6 6	-	-	-	-	Δ	Δ	0	-
IBEQ abdxys,rel9	(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	-	-	-	-	-	-	-	-	LDS #opr16i LDS opr8a LDS opr16a LDS oprx0_xysp LDS oprx9_xysp LDS oprx16_xysp LDS [D,xysp] LDS [oprx16_xysp]	(M:M+1) ⇒ SP Load Stack Pointer	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CF jj kk DF dd FF hh ll EF xb EF xb ff EF xb ee ff EF xb ee ff	2 3 3 3 3 4 6 6	-	-	-	-	Δ	Δ	0	-
IBNE abdxys,rel9	(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	-	-	-	-	-	-	-	-	LDS #opr16i LDS opr8a LDS opr16a LDS oprx0_xysp LDS oprx9_xysp LDS oprx16_xysp LDS [D,xysp] LDS [oprx16_xysp]	(M:M+1) ⇒ X Load Index Register X	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CE jj kk DE dd FE hh ll EE xb EE xb ff EE xb ee ff EE xb ee ff	2 3 3 3 3 4 6 6	-	-	-	-	Δ	Δ	0	-
IDIV	(D) ÷ (X) ⇒ X; Remainder ⇒ D 16 by 16 Bit Integer Divide (unsigned)	INH	18 10	12	-	-	-	-	Δ	0	Δ	Δ	LDX #opr16i LDX opr8a LDX opr16a LDX oprx0_xysp LDX oprx9_xysp LDX oprx16_xysp LDX [D,xysp] LDX [oprx16_xysp]	(M:M+1) ⇒ X Load Index Register X	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CE jj kk DE dd FE hh ll EE xb EE xb ff EE xb ee ff EE xb ee ff	2 3 3 3 3 4 6 6	-	-	-	-	Δ	Δ	0	-
INC opr16a INC oprx0_xysp INC oprx9_xysp INC oprx16_xysp INC [D,xysp] INC [oprx16_xysp] INCA INCB	(M) + \$01 ⇒ M Increment Memory Byte	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	72 hh ll 62 xb 62 xb ff 62 xb ee ff 62 xb 62 xb ee ff	4 3 4 5 6 6	-	-	-	-	Δ	Δ	Δ	Δ	LDY #opr16i LDY opr8a LDY opr16a LDY oprx0_xysp LDY oprx9_xysp LDY oprx16_xysp LDY [D,xysp] LDY [oprx16_xysp]	(M:M+1) ⇒ Y Load Index Register Y	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CD jj kk DD dd FD hh ll ED xb ED xb ff ED xb ee ff ED xb ee ff	2 3 3 3 3 4 6 6	-	-	-	-	Δ	Δ	0	-
INS	(SP) + \$0001 ⇒ SP Translates to LEAS 1,SP	INH	1B 81	2	-	-	-	-	-	-	-	-	LEAS oprx0_xysp LEAS oprx9_xysp LEAS oprx16_xysp	Effective Address ⇒ SP Load Effective Address into SP	IDX IDX1 IDX2	1B xb 1B xb ff 1B xb ee ff	2 2 2	-	-	-	-	-	-	-	-
INX	(X) + \$0001 ⇒ X Increment Index Register X	INH	08	1	-	-	-	-	Δ	-	-	-	LEAX oprx0_xysp LEAX oprx9_xysp LEAX oprx16_xysp	Effective Address ⇒ X Load Effective Address into X	IDX IDX1 IDX2	1A xb 1A xb ff 1A xb ee ff	2 2 2	-	-	-	-	-	-	-	-
INY	(Y) + \$0001 ⇒ Y Increment Index Register Y	INH	02	1	-	-	-	-	Δ	-	-	-	LEAY oprx0_xysp LEAY oprx9_xysp LEAY oprx16_xysp	Effective Address ⇒ Y Load Effective Address into Y	IDX IDX1 IDX2	19 xb 19 xb ff 19 xb ee ff	2 2 2	-	-	-	-	-	-	-	-
JMP opr16a JMP oprx0_xysp JMP oprx9_xysp JMP oprx16_xysp JMP [D,xysp] JMP [oprx16_xysp]	Routine address ⇒ PC Jump	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	06 hh ll 05 xb 05 xb ff 05 xb ee ff 05 xb 05 xb ee ff	3 3 3 4 6 6	-	-	-	-	-	-	-	-	LSL opr16a LSL oprx0_xysp LSL oprx9_xysp LSL oprx16_xysp LSL [D,xysp] LSL [oprx16_xysp]	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
JSR opr8a JSR opr16a JSR oprx0_xysp JSR oprx9_xysp JSR oprx16_xysp JSR [D,xysp] JSR [oprx16_xysp]	(SP) - 2 ⇒ SP; RTNH:RTNL ⇒ M(SP):M(SP+1); Subroutine address ⇒ PC Jump to Subroutine	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	17 dd 16 hh ll 15 xb 15 xb ff 15 xb ee ff 15 xb 15 xb ee ff	4 4 4 4 5 6 6	-	-	-	-	-	-	-	-	LSLA LSLB LSLD	Logical Shift Accumulator A to Left Logical Shift Accumulator B to Left	INH INH INH	48 58 59	1 1 1	-	-	-	-	Δ	Δ	Δ	Δ
LDAA #opr8i LDAA opr8a LDAA opr16a LDAA oprx0_xysp LDAA oprx9_xysp LDAA oprx16_xysp LDAA [D,xysp] LDAA [oprx16_xysp]	(M) ⇒ A Load Accumulator A	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	86 ii 96 dd B6 hh ll A6 xb A6 xb ff A6 xb ee ff A6 xb A6 xb ee ff	1 3 3 3 3 4 6 6	-	-	-	-	Δ	Δ	0	-	LSLA LSLB LSLD	Logical Shift Accumulator A to Left Logical Shift Accumulator B to Left	INH INH INH	48 58 59	1 1 1	-	-	-	-	Δ	Δ	Δ	Δ

Source Form	Operation	Addr. Mode	Machine Code (hex)	Exec. T (clk)	S	X	H	I	N	Z	V	C	Source Form	Operation	Addr. Mode	Machine Code (hex)	Exec. T (clk)	S	X	H	I	N	Z	V	C														
LSR <i>opr16a</i> LSR <i>opr0_xysp</i> LSR <i>opr9_xysp</i> LSR <i>opr16_xysp</i> LSR [D, <i>xysp</i>] LSR [<i>opr16_xysp</i>] LSRA LSRB	 Logical Shift Right	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	74 hh ll 64 xb 64 xb ff 64 xb ee ff 64 xb 64 xb ee ff 44 54	4 3 4 5 6 6 1 1	-	-	-	-	0	Δ	Δ	Δ	PSHX PSHY PULA PULB PULC PULD	(SP) - 2 ⇒ SP; (XH:XL) ⇒ M _(SP) ; M _(SP+1) Push Index Register X onto Stack (SP) - 2 ⇒ SP; (YH:YL) ⇒ M _(SP) ; M _(SP+1) Push Index Register Y onto Stack (M _(SP)) ⇒ A; (SP) + 1 ⇒ SP Pull Accumulator A from Stack (M _(SP)) ⇒ B; (SP) + 1 ⇒ SP Pull Accumulator B from Stack (M _(SP)) ⇒ CCR; (SP) + 1 ⇒ SP Pull CCR from Stack (M _(SP) ; M _(SP+1)) ⇒ A:B; (SP) + 2 ⇒ SP Pull D from Stack	INH INH INH INH INH INH INH	34 35 32 33 38 3A	2 2 3 3 3 3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-						
LSRD	 Logical Shift Right D Accumulator	INH	49	1	-	-	-	-	0	Δ	Δ	Δ	PULX PULY ROL <i>opr16a</i> ROL <i>opr0_xysp</i> ROL <i>opr9_xysp</i> ROL <i>opr16_xysp</i> ROL [D, <i>xysp</i>] ROL [<i>opr16_xysp</i>] ROLA ROLB	(M _(SP) ; M _(SP+1)) ⇒ X _H :X _L ; (SP) + 2 ⇒ SP Pull Index Register X from Stack (M _(SP) ; M _(SP+1)) ⇒ Y _H :Y _L ; (SP) + 2 ⇒ SP Pull Index Register Y from Stack  Rotate Memory Left through Carry Rotate A Left through Carry Rotate B Left through Carry	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	75 hh ll 65 xb 65 xb ff 65 xb ee ff 65 xb 65 xb ee ff 45 55	4 3 4 5 6 6 1 1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
MOVB # <i>opr8</i> , <i>opr16a</i> ²⁾ MOVB # <i>opr8i</i> , <i>opr0_xysp</i> ²⁾ MOVB <i>opr16a</i> , <i>opr16a</i> ²⁾ MOVB <i>opr16a</i> , <i>opr0_xysp</i> ²⁾ MOVB <i>opr0_xysp</i> , <i>opr16a</i> ²⁾ MOVB <i>opr0_xysp</i> , <i>opr0_xysp</i> ²⁾	(M ₁) ⇒ M ₂ Memory to Memory Byte-Move (8-Bit)	IMM-EXT IMM-IDX EXT-EXT EXT-IDX IDX-EXT IDX-IDX	18 0B ii hh ll 18 08 xb ii 18 0C hh ll hh ll 18 09 xb hh ll 18 0D xb hh ll 18 0A xb xb	4 4 6 5 5 5	-	-	-	-	-	-	-	-	ROR <i>opr16a</i> ROR <i>opr0_xysp</i> ROR <i>opr9_xysp</i> ROR <i>opr16_xysp</i> ROR [D, <i>xysp</i>] ROR [<i>opr16_xysp</i>] RORA RORB	 Rotate Memory Right through Carry Rotate A Right through Carry Rotate B Right through Carry	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	76 hh ll 66 xb 66 xb ff 66 xb ee ff 66 xb 66 xb ee ff 46 56	4 3 4 5 6 6 1 1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-					
MOVW # <i>opr16</i> , <i>opr16a</i> ²⁾ MOVW # <i>opr16i</i> , <i>opr0_xysp</i> ²⁾ MOVW <i>opr16a</i> , <i>opr16a</i> ²⁾ MOVW <i>opr16a</i> , <i>opr0_xysp</i> ²⁾ MOVW <i>opr0_xysp</i> , <i>opr16a</i> ²⁾ MOVW <i>opr0_xysp</i> , <i>opr0_xysp</i> ²⁾	(M:M+1) ⇒ M:M+1 ₂ Memory to Memory Word-Move (16-Bit)	IMM-EXT IMM-IDX EXT-EXT EXT-IDX IDX-EXT IDX-IDX	18 03 jj kk hh ll 18 00 xb jj kk 18 04 hh ll hh ll 18 01 xb hh ll 18 05 xb hh ll 18 02 xb xb	5 4 6 5 5 5	-	-	-	-	-	-	-	-	MUL	(A) × (B) ⇒ A:B - 8 × 8 Unsigned Multiply	INH	12	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-						
NEG <i>opr16a</i> NEG <i>opr0_xysp</i> NEG <i>opr9_xysp</i> NEG <i>opr16_xysp</i> NEG [D, <i>xysp</i>] NEG [<i>opr16_xysp</i>] NEGA NEGB	0 - (M) ⇒ M equivalent to $\overline{M} + 1$ ⇒ M Two's Complement Negate 0 - (A) ⇒ A equivalent to $\overline{A} + 1$ ⇒ A Negate Accumulator A 0 - (B) ⇒ B equivalent to $\overline{B} + 1$ ⇒ B Negate Accumulator B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	70 hh ll 60 xb 60 xb ff 60 xb ee ff 60 xb 60 xb ee ff 40 50	4 3 4 5 6 6 1 1	-	-	-	-	Δ	Δ	Δ	Δ	RTI	(M _(SP)) ⇒ CCR; (SP) + 1 ⇒ SP (M _(SP) ; M _(SP+1)) ⇒ B:A; (SP) + 2 ⇒ SP (M _(SP) ; M _(SP+1)) ⇒ X _H :X _L ; (SP) + 4 ⇒ SP (M _(SP) ; M _(SP+1)) ⇒ PC _H :PC _L ; (SP) - 2 ⇒ SP (M _(SP) ; M _(SP+1)) ⇒ Y _H :Y _L ; (SP) + 4 ⇒ SP Return from Interrupt	INH	0B	8/ 10 with interrupt pending	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ		
NOP	No Operation	INH	A7	1	-	-	-	-	-	-	-	-	RTS	(M _(SP) ; M _(SP+1)) ⇒ PC _H :PC _L ; (SP) + 2 ⇒ SP Return from Subroutine	INH	3D	5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-						
ORAA # <i>opr8i</i> ORAA <i>opr8a</i> ORAA <i>opr16a</i> ORAA <i>opr0_xysp</i> ORAA <i>opr9_xysp</i> ORAA <i>opr16_xysp</i> ORAA [D, <i>xysp</i>] ORAA [<i>opr16_xysp</i>]	(A) + (M) ⇒ A Logical OR A with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	8A ii 9A dd BA hh ll AA xb AA xb ff AA xb ee ff AA xb AA xb ee ff	1 3 3 3 3 4 6 6	-	-	-	-	Δ	Δ	0	-	SBA	(A) - (B) ⇒ A Subtract B from A	INH	18 16	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-							
ORAB # <i>opr8i</i> ORAB <i>opr8a</i> ORAB <i>opr16a</i> ORAB <i>opr0_xysp</i> ORAB <i>opr9_xysp</i> ORAB <i>opr16_xysp</i> ORAB [D, <i>xysp</i>] ORAB [<i>opr16_xysp</i>]	(B) + (M) ⇒ B Logical OR B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CA ii DA dd FA hh ll EA xb EA xb ff EA xb ee ff EA xb EA xb ee ff	1 3 3 3 3 4 6 6	-	-	-	-	Δ	Δ	0	-	SBCA # <i>opr8i</i> SBCA <i>opr8a</i> SBCA <i>opr16a</i> SBCA <i>opr0_xysp</i> SBCA <i>opr9_xysp</i> SBCA <i>opr16_xysp</i> SBCA [D, <i>xysp</i>] SBCA [<i>opr16_xysp</i>]	(A) - (M) - C ⇒ A Subtract with Borrow from A CA ii A2 xb A2 xb ff A2 xb ee ff A2 xb A2 xb ee ff	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	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-			
ORCC # <i>opr8i</i>	(CCR) + M ⇒ CCR Logical OR CCR with Memory	IMM	14 ii	1	↑	-	↑	↑	↑	↑	↑	↑	SBCB # <i>opr8i</i> SBCB <i>opr8a</i> SBCB <i>opr16a</i> SBCB <i>opr0_xysp</i> SBCB <i>opr9_xysp</i> SBCB <i>opr16_xysp</i> SBCB [D, <i>xysp</i>] SBCB [<i>opr16_xysp</i>]	(B) - (M) - C ⇒ B Subtract with Borrow from B C2 ii D2 dd F2 hh ll E2 xb E2 xb ff E2 xb ee ff E2 xb E2 xb ee ff	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	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PSHA	(SP) - 1 ⇒ SP; (A) ⇒ M _(SP) Push Accumulator A onto Stack	INH	36	2	-	-	-	-	-	-	-	-	SEC	1 ⇒ C Translates to ORCC #01	IMM	14 01	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-							
PSHB	(SP) - 1 ⇒ SP; (B) ⇒ M _(SP) Push Accumulator B onto Stack	INH	37	2	-	-	-	-	-	-	-	-	SEI	1 ⇒ I; (inhibit I interrupts) Translates to ORCC #10	IMM	14 10	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-						
PSHC	(SP) - 1 ⇒ SP; (CCR) ⇒ M _(SP) Push CCR onto Stack	INH	39	2	-	-	-	-	-	-	-	-	SEV	1 ⇒ V Translates to ORCC #02	IMM	14 02	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-						
PSHD	(SP) - 2 ⇒ SP; (A:B) ⇒ M _(SP) ; M _(SP+1) Push D Accumulator onto Stack	INH	3B	2	-	-	-	-	-	-	-	-																											

²⁾The first operand in the source code statement specifies the source for the move.

