Legend

Instruction operand notation

ii	8-bit immediate data
dd	low byte of a direct address
hh ll	high and low bytes of an extended address
ff	unsigned 8-bit offset in indexed addressed instruction
jj kk	high and low bytes of 16-bit immediate data
mm	8-bit mask byte—1s in the mask select operand bits
rr	signed 8-bit relative offset in branch instruction

Cycles

3	requires three E-clock cycles for fetch and execute
?	depends on external hardware signals

Condition code bit notation

-	Bit is unaffected by this instruction.
0	Bit is always cleared to 0 by instruction.
1	Bit is always set to 1 by instruction.
↕	Bit is set or cleared depending on instruction.
↓	Bit can change from 1 to 0 but not 0 to 1, or can remain at either 1 or 0.

Motorola 68HC11 Programming Model

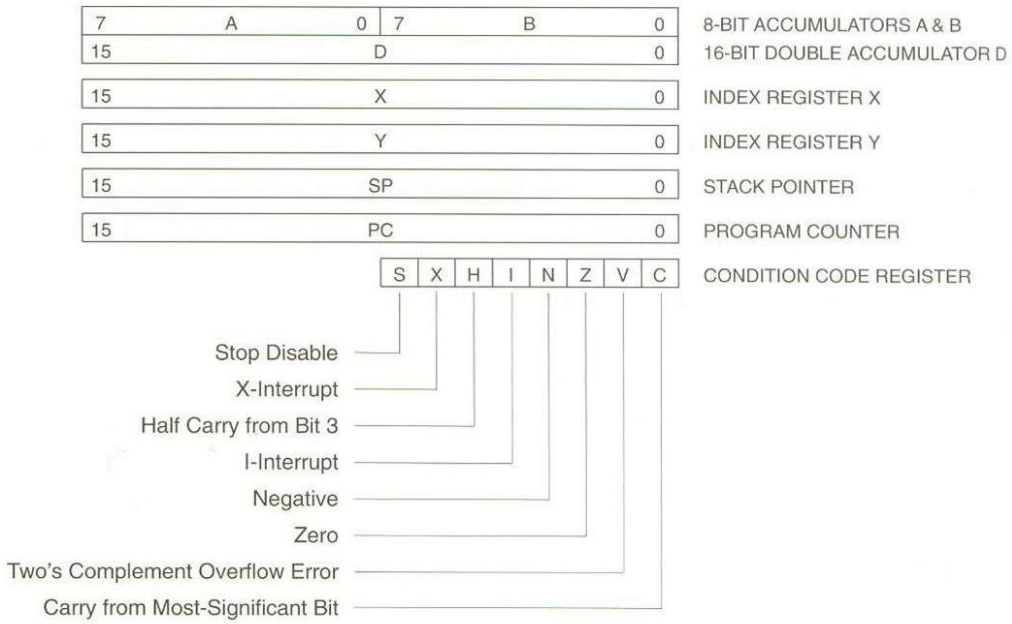
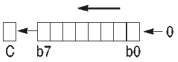
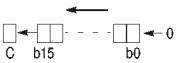
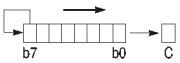


Table 10-1 MC68HC11A8 Instructions, Addressing Modes, and Execution Times (Sheet 1 of 6)

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Bytes	Cycle	Cycle by Cycle*	Condition Codes
				Opcode	Operand(s)				S X H I N Z V C
ABA	Add Accumulators	A + B → A	INH	1B		1	2	2-1	- - ↕ - ↕ ↕ ↕ ↕
ABX	Add B to X	IX + 00:B → IX	INH	3A		1	3	2-2	- - - - - - - -
ABY	Add B to Y	IY + 00:B → IY	INH	18 3A		2	4	2-4	- - - - - - - -
ADCA (opr)	Add with Carry to A	A + M + C → A	A IMM	89	ii	2	2	3-1	- - ↕ - ↕ ↕ ↕ ↕
			A DIR	99	dd	2	3	4-1	
			A EXT	B9	hh ll	3	4	5-2	
			A IND,X	A9	ff	2	4	6-2	
			A IND,Y	18 A9	ff	3	5	7-2	
ADCB (opr)	Add with Carry to B	B + M + C → B	B IMM	C9	ii	2	2	3-1	- - ↕ - ↕ ↕ ↕ ↕
			B DIR	D9	dd	2	3	4-1	
			B EXT	F9	hh ll	3	4	5-2	
			B IND,X	E9	ff	2	4	6-2	
			B IND,Y	18 E9	ff	3	5	7-2	
ADDA (opr)	Add Memory to A	A + M → A	A IMM	8B	ii	2	2	3-1	- - ↕ - ↕ ↕ ↕ ↕
			A DIR	9B	dd	2	3	4-1	
			A EXT	BB	hh ll	3	4	5-2	
			A IND,X	AB	ff	2	4	6-2	
			A IND,Y	18 AB	ff	3	5	7-2	
ADDB (opr)	Add Memory to B	B + M → B	B IMM	CB	ii	2	2	3-1	- - ↕ - ↕ ↕ ↕ ↕
			B DIR	DB	dd	2	3	4-1	
			B EXT	FB	hh ll	3	4	5-2	
			B IND,X	EB	ff	2	4	6-2	
			B IND,Y	18 EB	ff	3	5	7-2	
ADDD (opr)	Add 16-Bit to D	D + M:M + 1 → D	IMM	C3	jj kk	3	4	3-3	- - - - ↕ ↕ ↕ ↕
			DIR	D3	dd	2	5	4-7	
			EXT	F3	hh ll	3	6	5-10	
			IND,X	E3	ff	2	6	6-10	
			IND,Y	18 E3	ff	3	7	7-8	
ANDA (opr)	AND A with Memory	A•M → A	A IMM	84	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			A DIR	94	dd	2	3	4-1	
			A EXT	B4	hh ll	3	4	5-2	
			A IND,X	A4	ff	2	4	6-2	
			A IND,Y	18 A4	ff	3	5	7-2	
ANDB (opr)	AND B with Memory	B•M → B	B IMM	C4	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			B DIR	D4	dd	2	3	4-1	
			B EXT	F4	hh ll	3	4	5-2	
			B IND,X	E4	ff	2	4	6-2	
			B IND,Y	18 E4	ff	3	5	7-2	
ASL (opr)	Arithmetic Shift Left		EXT	78	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ ↕
			IND,X	68	ff	2	6	6-3	
ASLA			IND,Y	18 68	ff	3	7	7-3	
			A INH	48		1	2	2-1	
ASLB			B INH	58		1	2	2-1	
ASLD	Arithmetic Shift Left Double		INH	05		1	3	2-2	- - - - ↕ ↕ ↕ ↕
ASR (opr)	Arithmetic Shift Right		EXT	77	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ ↕
			IND,X	67	ff	2	6	6-3	
ASRA			IND,Y	18 67	ff	3	7	7-3	
			A INH	47		1	2	2-1	
ASRB			B INH	57		1	2	2-1	
BCC (rel)	Branch if Carry Clear	? C = 0	REL	24	rr	2	3	8-1	- - - - - - - -
BCLR (opr) (msk)	Clear Bit(s)	M•($\overline{mm}$) → M	DIR	15	dd mm	3	6	4-10	- - - - ↕ ↕ 0 -
			IND,X	1D	ff mm	3	7	6-13	
			IND,Y	18 1D	ff mm	4	8	7-10	
BCS (rel)	Branch if Carry Set	? C = 1	REL	25	rr	2	3	8-1	- - - - - - - -
BEQ (rel)	Branch if = Zero	? Z = 1	REL	27	rr	2	3	8-1	- - - - - - - -
BGE (rel)	Branch if ≥ Zero [S]	? N ⊕ V = 0	REL	2C	rr	2	3	8-1	- - - - - - - -
BGT (rel)	Branch if > Zero [S]	? Z + (N ⊕ V) = 0	REL	2E	rr	2	3	8-1	- - - - - - - -
BHI (rel)	Branch if Higher [U]	? C + Z = 0	REL	22	rr	2	3	8-1	- - - - - - - -
BHS (rel)	Branch if Higher or Same [U]	? C = 0	REL	24	rr	2	3	8-1	- - - - - - - -

*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

Table 10-1 MC68HC11A8 Instructions, Addressing Modes, and Execution Times (Sheet 2 of 6)

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Bytes	Cycle	Cycle by Cycle*	Condition Codes
				Opcode	Operand(s)				S X H I N Z V C
BITA (opr)	Bit(s) Test A with Memory	A•M	A IMM	85	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			A DIR	95	dd	2	3	4-1	
			A EXT	B5	hh ll	3	4	5-2	
			A IND,X	A5	ff	2	4	6-2	
			A IND,Y	18 A5	ff	3	5	7-2	
BITB (opr)	Bit(s) Test B with Memory	B•M	B IMM	C5	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			B DIR	D5	dd	2	3	4-1	
			B EXT	F5	hh ll	3	4	5-2	
			B IND,X	E5	ff	2	4	6-2	
			B IND,Y	18 E5	ff	3	5	7-2	
BLE (rel)	Branch if ≤ Zero ^{S}	? Z + (N ⊕ V) = 1	REL	2F	rr	2	3	8-1	- - - - - - - -
BLO (rel)	Branch if Lower ^{u}	? C = 1	REL	25	rr	2	3	8-1	- - - - - - - -
BLS (rel)	Branch if Lower or Same ^{u}	? C + Z = 1	REL	23	rr	2	3	8-1	- - - - - - - -
BLT (rel)	Branch If < Zero ^{S}	? N ⊕ V = 1	REL	2D	rr	2	3	8-1	- - - - - - - -
BMI (rel)	Branch if Minus	? N = 1	REL	2B	rr	2	3	8-1	- - - - - - - -
BNE (rel)	Branch if Not = Zero	? Z = 0	REL	26	rr	2	3	8-1	- - - - - - - -
BPL (rel)	Branch if Plus	? N = 0	REL	2A	rr	2	3	8-1	- - - - - - - -
BRA (rel)	Branch Always	? 1 = 1	REL	20	rr	2	3	8-1	- - - - - - - -
BRCLR(opr) (msk) (rel)	Branch if Bit(s) Clear	? M • mm = 0	DIR	13	dd mm rr	4	6	4-11	- - - - - - - -
			IND,X	1F	ff mm rr	4	7	6-14	
			IND,Y	18 1F	ff mm rr	5	8	7-11	
BRN (rel)	Branch Never	? 1 = 0	REL	21	rr	2	3	8-1	- - - - - - - -
BRSET(opr) (msk) (rel)	Branch if Bit(s) Set	? ($\overline{M}$) • mm = 0	DIR	12	dd mm rr	4	6	4-11	- - - - - - - -
			IND,X	1E	ff mm rr	4	7	6-14 7-	
			IND,Y	18 1E	ff mm rr	5	8	11	
BSET(opr) (msk)	Set Bit(s)	M + mm → M	DIR	14	dd mm	3	6	4-10	- - - - ↕ ↕ 0 -
			IND,X	1C	ff mm	3	7	6-13	
			IND,Y	18 1C	ff mm	4	8	7-10	
BSR (rel)	Branch to Subroutine	See Special Ops	REL	8D	rr	2	6	8-2	- - - - - - - -
BVC (rel)	Branch if Overflow Clear	? V = 0	REL	28	rr	2	3	8-1	- - - - - - - -
BVS (rel)	Branch if Overflow Set	? V = 1	REL	29	rr	2	3	8-1	- - - - - - - -
CBA	Compare A to B	A – B	INH	11		1	2	2-1	- - - - ↕ ↕ ↕ ↕
CLC	Clear Carry Bit	0 → C	INH	0C		1	2	2-1	- - - - - - - 0
CLI	Clear Interrupt Mask	0 → I	INH	0E		1	2	2-1	- - - 0 - - - -
CLR (opr)	Clear Memory Byte	0 → M	EXT	7F	hh ll	3	6	5-8	- - - - 0 1 0 0
			IND,X	6F	ff	2	6	6-3	
			IND,Y	18 6F	ff	3	7	7-3	
CLRA	Clear Accumulator A	0 → A	A INH	4F		1	2	2-1	- - - - 0 1 0 0
CLRB	Clear Accumulator B	0 → B	B INH	5F		1	2	2-1	- - - - 0 1 0 0
CLV	Clear Overflow Flag	0 → V	INH	0A		1	2	2-1	- - - - - - 0 -
CMPA (opr)	Compare A to Memory	A – M	A IMM	81	ii	2	2	3-1	- - - - ↕ ↕ ↕ ↕
			A DIR	91	dd	2	3	4-1	
			A EXT	B1	hh ll	3	4	5-2	
			A IND,X	A1	ff	2	4	6-2	
			A IND,Y	18 A1	ff	3	5	7-2	
CMPB (opr)	Compare B to Memory	B – M	B IMM	C1	ii	2	2	3-1	- - - - ↕ ↕ ↕ ↕
			B DIR	D1	dd	2	3	4-1	
			B EXT	F1	hh ll	3	4	5-2	
			B IND,X	E1	ff	2	4	6-2	
			B IND,Y	18 E1	ff	3	5	7-2	
COM (opr)	1's Complement Memory Byte	$FF – M → M	EXT	73	hh ll	3	6	5-8	- - - - ↕ ↕ 0 1
			IND,X	63	ff	2	6	6-3	
			IND,Y	18 63	ff	3	7	7-3	
COMA	1's Complement A	$FF – A → A	A INH	43		1	2	2-1	- - - - ↕ ↕ 0 1
COMB	1's Complement B	$FF – B → B	B INH	53		1	2	2-1	- - - - ↕ ↕ 0 1
CPD (opr)	Compare D to Memory 16-Bit	D – M:M + 1	IMM	1A 83	jj kk	4	5	3-5	- - - - ↕ ↕ ↕ ↕
			DIR	1A 93	dd	3	6	4-9	
			EXT	1A B3	hh ll	4	7	5-11	
			IND,X	1A A3	ff	3	7	6-11	
			IND,Y	CD A3	ff	3	7	7-8	

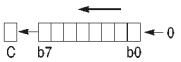
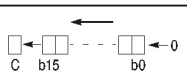
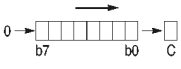
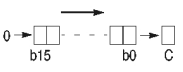
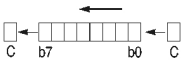
*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

Table 10-1 MC68HC11A8 Instructions, Addressing Modes, and Execution Times (Sheet 3 of 6)

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Bytes	Cycle	Cycle by Cycle*	Condition Codes
				Opcode	Operand(s)				S X H I N Z V C
CPX (opr)	Compare X to Memory 16-Bit	IX – M:M + 1	IMM	8C	jj kk	3	4	3-3	- - - - ↕ ↕ ↕ ↕
			DIR	9C	dd	2	5	4-7	
			EXT	BC	hh ll	3	6	5-10	
			IND,X	AC	ff	2	6	6-10	
			IND,Y	CD AC	ff	3	7	7-8	
CPY (opr)	Compare Y to Memory 16-Bit	IY – M:M + 1	IMM	18 8C	jj kk	4	5	3-5	- - - - ↕ ↕ ↕ ↕
			DIR	18 9C	dd	3	6	4-9	
			EXT	18 BC	hh ll	4	7	5-11	
			IND,X	1A AC	ff	3	7	6-11	
			IND,Y	18 AC	ff	3	7	7-8	
DAA	Decimal Adjust A	Adjust Sum to BCD	INH	19		1	2	2-1	- - - - ↕ ↕ ↕ ↕
DEC (opr)	Decrement Memory Byte	M – 1 → M	EXT	7A	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ -
			IND,X	6A	ff	2	6	6-3	
			IND,Y	18 6A	ff	3	7	7-3	
DECA	Decrement Accumulator A	A – 1 → A	A INH	4A		1	2	2-1	- - - - ↕ ↕ ↕ -
DECB	Decrement Accumulator B	B – 1 → B	B INH	5A		1	2	2-1	- - - - ↕ ↕ ↕ -
DES	Decrement Stack Pointer	SP – 1 → SP	INH	34		1	3	2-3	- - - - - - - -
DEX	Decrement Index Register X	IX – 1 → IX	INH	09		1	3	2-2	- - - - - ↕ - -
DEY	Decrement Index Register Y	IY – 1 → IY	INH	18 09		2	4	2-4	- - - - - ↕ - -
EORA (opr)	Exclusive OR A with Memory	A ⊕ M → A	A IMM	88	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			A DIR	98	dd	2	3	4-1	
			A EXT	88	hh ll	3	4	5-2	
			A IND,X	A8	ff	2	4	6-2	
			A IND,Y	18 A8	ff	3	5	7-2	
EORB (opr)	Exclusive OR B with Memory	B ⊕ M → B	B IMM	C8	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			B DIR	D8	dd	2	3	4-1	
			B EXT	F8	hh ll	3	4	5-2	
			B IND,X	E8	ff	2	4	6-2	
			B IND,Y	18 E8	ff	3	5	7-2	
FDIV	Fractional Divide 16 by 16	D/IX → IX; r → D	INH	03		1	41	2-17	- - - - - ↕ ↕ ↕
IDIV	Integer Divide 16 by 16	D/IX → IX; r → D	INH	02		1	41	2-17	- - - - - ↕ 0 ↕
INC (opr)	Increment Memory Byte	M + 1 → M	EXT	7C	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ -
			IND,X	6C	ff	2	6	6-3	
			IND,Y	18 6C	ff	3	7	7-3	
INCA	Increment Accumulator A	A + 1 → A	A INH	4C		1	2	2-1	- - - - ↕ ↕ ↕ -
INCB	Increment Accumulator B	B + 1 → B	B INH	5C		1	2	2-1	- - - - ↕ ↕ ↕ -
INS	Increment Stack Pointer	SP + 1 → SP	INH	31		1	3	2-3	- - - - - - - -
INX	Increment Index Register X	IX + 1 → IX	INH	08		1	3	2-2	- - - - - ↕ - -
INY	Increment Index Register Y	IY + 1 → IY	INH	18 08		2	4	2-4	- - - - - ↕ - -
JMP (opr)	Jump	See Special Ops	EXT	7E	hh ll	3	3	5-1	- - - - - - - -
			IND,X	6E	ff	2	3	6-1	
			IND,Y	18 6E	ff	3	4	7-1	
JSR (opr)	Jump to Subroutine	See Special Ops	DIR	9D	dd	2	5	4-8	- - - - - - - -
			EXT	BD	hh ll	3	6	5-12	
			IND,X	AD	ff	2	6	6-12	
			IND,Y	18 AD	ff	3	7	7-9	
LDAA (opr)	Load Accumulator A	M → A	A IMM	86	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			A DIR	96	dd	2	3	4-1	
			A EXT	B6	hh ll	3	4	5-2	
			A IND,X	A6	ff	2	4	6-2	
			A IND,Y	18 A6	ff	3	5	7-2	
LDAB (opr)	Load Accumulator B	M → B	B IMM	C6	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			B DIR	D6	dd	2	3	4-1	
			B EXT	F6	hh ll	3	4	5-2	
			B IND,X	E6	ff	2	4	6-2	
			B IND,Y	18 E6	ff	3	5	7-2	
LDD (opr)	Load Double Accumulator D	M → A,M + 1 → B	IMM	CC	jj kk	3	3	3-2	- - - - ↕ ↕ 0 -
			DIR	DC	dd	2	4	4-3	
			EXT	FC	hh ll	3	5	5-4	
			IND,X	EC	ff	2	5	6-6	
			IND,Y	18 EC	ff	3	6	7-6	

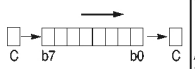
*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

Table 10-1 MC68HC11A8 Instructions, Addressing Modes, and Execution Times (Sheet 4 of 6)

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Bytes	Cycle	Cycle by Cycle*	Condition Codes
				Opcode	Operand(s)				S X H I N Z V C
LDS (opr)	Load Stack Pointer	M:M + 1 → SP	IMM	8E	jj kk	3	3	3-2	- - - - ↕ ↕ 0 -
			DIR	9E	dd	2	4	4-3	
			EXT	BE	hh ll	3	5	5-4	
			IND,X	AE	ff	2	5	6-6	
			IND,Y	18 AE	ff	3	6	7-6	
LDX (opr)	Load Index Register X	M:M + 1 → IX	IMM	CE	jj kk	3	3	3-2	- - - - ↕ ↕ 0 -
			DIR	DE	dd	2	4	4-3	
			EXT	FE	hh ll	3	5	5-4	
			IND,X	EE	ff	2	5	6-6	
			IND,Y	CD EE	ff	3	6	7-6	
LDY (opr)	Load Index Register Y	M:M + 1 → IY	IMM	18 CE	jj kk	4	4	3-4	- - - - ↕ ↕ 0 -
			DIR	18 DE	dd	3	5	4-5	
			EXT	18 FE	hh ll	4	6	5-6	
			IND,X	1A EE	ff	3	6	6-7	
			IND,Y	18 EE	ff	3	6	7-6	
LSL (opr) LSLA LSLB	Logical Shift Left		EXT	78	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ ↕
			IND,X	68	ff	2	6	6-3	
			IND,Y	18 68	ff	3	7	3-7	
			A INH	48		1	2	2-1	
			B INH	58		1	2	2-1	
LSLD	Logical Shift Left Double		INH	05		1	3	2-2	- - - - ↕ ↕ ↕ ↕
LSR (opr) LSRA LSRB	Logical Shift Right		EXT	74	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ ↕
			IND,X	64	ff	2	6	6-3	
			IND,Y	18 64	ff	3	7	7-3	
			A INH	44		1	2	2-1	
			B INH	54		1	2	2-1	
LSRD	Logical Shift Right Double		INH	04		1	3	2-2	- - - - 0 ↕ ↕ ↕
MUL	Multiply 8 by 8	AxB → D	INH	3D		1	10	2-13	- - - - - - - ↕
NEG (opr)	2's Complement Memory Byte	0 – M → M	EXT	70	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ ↕
			IND,X	60	ff	2	6	6-3	
			IND,Y	18 60	ff	3	7	7-3	
NEGA	2's Complement A	0 – A → A	A INH	40		1	2	2-1	- - - - ↕ ↕ ↕ ↕
NEGB	2's Complement B	0 – B → B	B INH	50		1	2	2-1	- - - - ↕ ↕ ↕ ↕
NOP	No Operation	No Operation	INH	01		1	2	2-1	- - - - - - - -
ORAA (opr)	OR Accumulator A (Inclusive)	A + M → A	A IMM	8A	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			A DIR	9A	dd	2	3	4-1	
			A EXT	BA	hh ll	3	4	5-2	
			A IND,X	AA	ff	2	4	6-2	
			A IND,Y	18 AA	ff	3	5	7-2	
ORAB (opr)	OR Accumulator B (Inclusive)	B + M → B	B IMM	CA	ii	2	2	3-1	- - - - ↕ ↕ 0 -
			B DIR	DA	dd	2	3	4-1	
			B EXT	FA	hh ll	3	4	5-2	
			B IND,X	EA	ff	2	4	6-2	
			B IND,Y	18 EA	ff	3	5	7-2	
PSHA	Push A onto Stack	A → Stk, SP = SP–1	A INH	36		1	3	2-6	- - - - - - - -
PSHB	Push B onto Stack	B → Stk, SP = SP–1	B INH	37		1	3	2-6	- - - - - - - -
PSHX	Push X onto Stack (Lo First)	IX → Stk, SP = SP–2	INH	3C		1	4	2-7	- - - - - - - -
PSHY	Push Y onto Stack (Lo First)	IY → Stk, SP = SP–2	INH	18 3C		2	5	2-8	- - - - - - - -
PULA	Pull A from Stack	SP = SP + 1, A←Stk	A INH	32		1	4	2-9	- - - - - - - -
PULB	Pull B from Stack	SP = SP + 1, B←Stk	B INH	33		1	4	2-9	- - - - - - - -
PULX	Pull X from Stack (Hi First)	SP = SP + 2, IX←Stk	INH	38		1	5	2-10	- - - - - - - -
PULY	Pull Y from Stack (Hi First)	SP = SP + 2, IY←Stk	INH	18 38		2	6	2-11	- - - - - - - -
ROL (opr) ROLA ROLB	Rotate Left		EXT	79	hh ll	3	6	5-8	- - - - ↕ ↕ ↕ ↕
			IND,X	69	ff	2	6	6-3	
			IND,Y	18 69	ff	3	7	7-3	
			A INH	49		1	2	2-1	
			B INH	59		1	2	2-1	

*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

Table 10-1 MC68HC11A8 Instructions, Addressing Modes, and Execution Times (Sheet 5 of 6)

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Bytes	Cycle	Cycle by Cycle*	Condition Codes
				Opcode	Operand(s)				S X H I N Z V C
ROR (opr) RORA RORB	Rotate Right		EXT IND,X IND,Y A INH B INH	76 66 18 66 46 56	hh ll ff ff	3 2 3 1 1	6 6 7 2 2	5-8 6-3 7-3 2-1 2-1	- - - - ↕ ↕ ↕ ↕
RTI	Return from Interrupt	See Special Ops	INH	3B		1	12	2-14	↕ ↓ ↕ ↕ ↕ ↕ ↕ ↕
RTS	Return from Subroutine	See Special Ops	INH	39		1	5	2-12	- - - - - - - -
SBA	Subtract B from A	A – B → A	INH	10		1	2	2-1	- - - - ↕ ↕ ↕ ↕
SBCA (opr)	Subtract with Carry from A	A – M – C → A	A IMM A DIR A EXT A IND,X A IND,Y	82 92 B2 A2 18 A2	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	3-1 4-1 5-2 6-2 7-2	- - - - ↕ ↕ ↕ ↕
SBCB (opr)	Subtract with Carry from B	B – M – C → B	B IMM B DIR B EXT B IND,X B IND,Y	C2 D2 F2 E2 18 E2	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	3-1 4-1 5-2 6-2 7-2	- - - - ↕ ↕ ↕ ↕
SEC	Set Carry	1 → C	INH	0D		1	2	2-1	- - - - - - - 1
SEI	Set Interrupt Mask	1 → I	INH	0F		1	2	2-1	- - - 1 - - - -
SEV	Set Overflow Flag	1 → V	INH	0B		1	2	2-1	- - - - - - 1 -
STAA (opr)	Store Accumulator A	A → M	A DIR A EXT A IND,X A IND,Y	97 B7 A7 18 A7	dd hh ll ff ff	2 3 2 3	3 4 4 5	4-2 5-3 6-5 7-5	- - - - ↕ ↕ 0 -
STAB (opr)	Store Accumulator B	B → M	B DIR B EXT B IND,X B IND,Y	D7 F7 E7 18 E7	dd hh ll ff ff	2 3 2 3	3 4 4 5	4-2 5-3 6-5 7-5	- - - - ↕ ↕ 0 -
STD (opr)	Store Accumulator D	A → M, B → M + 1	DIR EXT IND,X IND,Y	DD FD ED 18 ED	dd hh ll ff ff	2 3 2 3	4 5 5 6	4-4 5-5 6-8 7-7	- - - - ↕ ↕ 0 -
STOP	Stop Internal Clocks		INH	CF		1	2	2-1	- - - - - - - -
STS (opr)	Store Stack Pointer	SP → M:M + 1	DIR EXT IND,X IND,Y	9F BF AF 18 AF	dd hh ll ff ff	2 3 2 3	4 5 5 6	4-4 5-5 6-8 7-7	- - - - ↕ ↕ 0 -
STX (opr)	Store Index Register X	IX → M:M + 1	DIR EXT IND,X IND,Y	DF FF EF CD EF	dd hh ll ff ff	2 3 2 3	4 5 5 6	4-4 5-5 6-8 7-7	- - - - ↕ ↕ 0 -
STY (opr)	Store Index Register Y	IY → M:M + 1	DIR EXT IND,X IND,Y	18 DF 18 FF 1A EF 18 EF	dd hh ll ff ff	3 4 3 3	5 6 6 6	4-6 5-7 6-9 7-7	- - - - ↕ ↕ 0 -
SUBA (opr)	Subtract Memory from A	A – M → A	A IMM A DIR A EXT A IND,X A IND,Y	80 90 B0 A0 18 A0	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	3-1 4-1 5-2 6-2 7-2	- - - - ↕ ↕ ↕ ↕
SUBB (opr)	Subtract Memory from B	B – M → B	B IMM B DIR B EXT B IND,X B IND,Y	C0 D0 F0 E0 18 E0	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	3-1 4-1 5-2 6-2 7-2	- - - - ↕ ↕ ↕ ↕
SUBD (opr)	Subtract Memory from D	D – M:M + 1→ D	IMM DIR EXT IND,X IND,Y	83 93 B3 A3 18 A3	jj kk dd hh ll ff ff	3 2 3 2 3	4 5 6 6 7	3-3 4-7 5-10 6-10 7-8	- - - - ↕ ↕ ↕ ↕
SWI	Software Interrupt	See Special Ops	INH	3F		1	14	2-15	- - - 1 - - - -
TAB	Transfer A to B	A → B	INH	16		1	2	2-1	- - - - ↕ ↕ 0 -
TAP	Transfer A to CC Register	A → CCR	INH	06		1	2	2-1	↕ ↓ ↕ ↕ ↕ ↕ ↕ ↕
TBA	Transfer B to A	B → A	INH	17		1	2	2-1	- - - - ↕ ↕ 0 -

*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

Table 10-1 MC68HC11A8 Instructions, Addressing Modes, and Execution Times (Sheet 6 of 6)

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Bytes	Cycle	Cycle by Cycle*	Condition Codes
				Opcode	Operand(s)				S X H I N Z V C
TEST	TEST (Only in Test Modes)	Address Bus Counts	INH	00		1	**	2-20	- - - - - - - -
TPA	Transfer CC Register to A	CCR → A	INH	07		1	2	2-1	- - - - - - - -
TST (opr)	Test for Zero or Minus	M – 0	EXT IND,X IND,Y	7D 6D 18 6D	hh ll ff ff	3 2 3	6 6 7	5-9 6-4 7-4	- - - - ↕ ↕ 0 0
TSTA		A – 0	A INH	4D		1	2	2-1	- - - - ↕ ↕ 0 0
TSTB		B – 0	B INH	5D		1	2	2-1	- - - - ↕ ↕ 0 0
TSX	Transfer Stack Pointer to X	SP + 1 → IX	INH	30		1	3	2-3	- - - - - - - -
TSY	Transfer Stack Pointer to Y	SP + 1 → IY	INH	18 30		2	4	2-5	- - - - - - - -
TXS	Transfer X to Stack Pointer	IX – 1 → SP	INH	35		1	3	2-2	- - - - - - - -
TYS	Transfer Y to Stack Pointer	IY – 1 → SP	INH	18 35		2	4	2-4	- - - - - - - -
WAI	Wait for Interrupt	Stack Regs & WAIT	INH	3E		1	***	2-16	- - - - - - - -
XGDX	Exchange D with X	IX → D, D → IX	INH	8F		1	3	2-2	- - - - - - - -
XGDY	Exchange D with Y	IY → D, D → IY	INH	18 8F		2	4	2-4	- - - - - - - -

*Cycle-by-cycle number provides a reference to Tables 10-2 through 10-8 which detail cycle-by-cycle operation.
Example: Table 10-1 Cycle-by-Cycle column reference number 2-4 equals Table 10-2 line item 2-4.

**Infinity or Until Reset Occurs

***12 Cycles are used beginning with the opcode fetch. A wait state is entered which remains in effect for an integer number of MPU E-clock cycles (n) until an interrupt is recognized. Finally, two additional cycles are used to fetch the appropriate interrupt vector (14 + n total).

dd = 8-Bit Direct Address ($0000 –$00FF) (High Byte Assumed to be $00)
ff = 8-Bit Positive Offset $00 (0) to $FF (255) (Is Added to Index)
hh = High Order Byte of 16-Bit Extended Address
ii = One Byte of Immediate Data
jj = High Order Byte of 16-Bit Immediate Data
kk = Low Order Byte of 16-Bit Immediate Data
ll = Low Order Byte of 16-Bit Extended Address
mm = 8-Bit Bit Mask (Set Bits to be Affected)
rr = Signed Relative Offset $80 (– 128) to $7F (+ 127)
(Offset Relative to the Address Following the Machine Code Offset Byte)

68HC11 Instruction Set by Instruction Category

ARITHMETIC

ADDITION

ABA	A = A + B
ABX	IX = IX + B
ABY	IY = IY + B
ADCA	A = A + M + CarryFlag
ADCB	B = B + M + CarryFlag
ADDA	A = A + M
ADDB	B = B + M
ADDD	D = D + M

SUBTRACTION

SBA	A = A - B
SBCA	A = A - M - CarryFlag
SBCB	B = B - M - CarryFlag
SUBA	A = A - M
SUBB	B = B - M
SUBD	D = D - M

TWOS COMPLEMENT (NEGATE)

NEG	M = -M
NEGA	A = -A
BEGB	B = -B

DECREMENT

DEC	M = M - 1
DECA	A = A - 1
DECB	B = B - 1
DES	SP = SP - 1
DEX	IX = IX - 1

INCREMENT

INC	M = M + 1
INCA	A = A + 1
INCB	B = B + 1
INS	SP = SP + 1
INX	IX = IX + 1
INY	IY = IY + 1

MULTIPLY

MUL	D = A * B

DIVIDE

IDIV	IX = D / IX, D = D % IX
FDIV	IX = D / IX, D = D % IX (FDIV treats args as fractions)

ARITHMETIC SHIFT

LEFT: (Multiply by 2)

ASL	Arithmetic Shift Left (M)
ASLA	Arithmetic Shift Left (A)
ASLB	Arithmetic Shift Left (B)
ASLD	Arithmetic Shift Left (D)

RIGHT: (Divide By 2)

ASR	Arithmetic Shift Right (M)
ASRA	Arithmetic Shift Right (A)
ASRB	Arithmetic Shift Right (B)

LOGICAL SHIFT

LEFT:

LSL	Logical Shift Left (M)
LSLA	Logical Shift Left (A)
LSLB	Logical Shift Left (B)
LSLD	Logical Shift Left (D)

SHIFT RIGHT:

LSR	Logical Shift Right (M)
LSRA	Logical Shift Right (A)
LSRB	Logical Shift Right (B)
LSRD	Logical Shift Right (D)

ROTATE

LEFT: (used to extend multiply)

ROL	ROtate Left (M)
ROLA	ROtate Left (A)
ROLB	ROtate Left (B)

RIGHT: (used to extend divide)

ROR	ROtate Right (M)
RORA	ROtate Right (A)
RORB	ROtate Right (B)

BINARY CODED DECIMAL (BCD)

DAA	Decimal Adjust after Addition

Branch & Jump

BRA	Branch Always
BRN	Branch Never
JMP	Jump to Address
JSR	Jump to Subroutine
NOP	No OPeration ; i.e do nothing but fetch next instruction

CLEAR (bit(s) = 0) & SET (bit(s) = 1)

CLR	M = 0
CLRA	A = 0
CLRB	B = 0
BCLR	Clear Bits (M)
BSET	Set Bits (M)

COMPARE & TEST

CONDITION CODE MANIPULATION

CLC	CarryFlag = 0	Clear Carry Flag
CLV	OVerflowFlag = 0	Clear Overflow Flag
SEC	CarryFlag = 1	Set Carry Flag
SEV	OVerflowFlag = 1	Set Overflow Flag
TAP	CCR = A	Transfer A to Condition Codes Register (CCR)
TPA	A = CCR	Transfer CCR to A

CONDITIONAL Branches

BEQ	Branch if EQual
BNE	Branch if Not Equal
BCC	Branch if CarryFlag is Clear

BCS Branch if CarryFlag is Set
BRCLR Branch if bits clear
BRSET Branch if bits set

Conditional Branches using SIGNED NUMERIC INTERPRETATION

BMI Branch if MInus
BPL Branch if PLus
BVS Branch if oVerflow Set
BVC Branch if oVerflow Clear
BLT Branch if Less Than
BGE Branch if Greater-Than or Equal-to
BLE Branch if Less-Than or Equal-to

Branches for UN-SIGNED NUMERIC INTERPRETATION

BHI Branch if HIgher than
BHS Branch if Higher or Same
BLS Branch if Lower or Same
BLO Branch if Lower

DATA MOVEMENT

Push - Push register value onto stack

PSHA M[SP--] = A
PSHB M[SP--] = B
PSHX M[SP--] = IX.LOW ;
M[SP--] = IX.HIGH
PSHY M[SP--] = IY.LOW ;
M[SP--] = IY.HIGH

Pull - Pull (POP) value from stack to Register

PULA A = M[++Sp]
PULB B = M[++SP]
PULX X.HIGH = M[++SP] ; X.LOW = M[++SP]
PULY Y.HIGH = M[++SP] ; Y.LOW = M[++SP]

Load Register

LDAA A = M
LDAB B = M
LDD D = M
LDS SP = M
LDX X = M
LDY Y = M

Store Register

STAA A -> M
STAB B -> M
STD D -> M:M+1
([M] = A, [M+1] = B)
STX IX -> M:M+1
STY IY -> M:M+1

Transfer Registers

TAB A = B
TBA B = A
TSX IX = SP + 1
TSY IY = SP + 1
TXS SP = IX - 1
TXY SP = IY - 1

Exchange Registers

XGDX D <=> IX
XGDY D <=> IY

INTERRUPT HANDLING:

CLI ; Clear interrupt Mask
SEI ; Set interrupt Mask
SWI ; Software Interrupt
RTI ; Return from Interrupt
WAI ; Wait for interrupt

LOGICAL

LOGICAL AND

ANDA A = A & M
ANDB B = B & M

LOGICAL EXCLUSIVE OR

EORA A = A ^ M
EORB B = B ^ M

LOGICAL OR

ORAA A = A | M
ORAB B = B | M

ONES COMPLEMENT (NOT)

COM M = M#
COMA A = A#
COMB B = B#

MISCELLANEOUS

STOP Stop clocks
TEST Special test mode