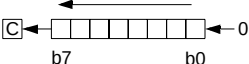


4.4 Instruction Set Summary

Table 4-10 provides a summary of the M68HC08 instruction set in all possible addressing modes. The table shows operand construction and the execution time in internal bus clock cycles of each instruction.

Table 4-10. Instruction Set Summary (Sheet 1 of 9)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ADC <i>#opr8i</i> ADC <i>opr8a</i> ADC <i>opr16a</i> ADC <i>opr16,X</i> ADC <i>opr8,X</i> ADC <i>,X</i> ADC <i>opr16,SP</i> ADC <i>opr8,SP</i>	Add with Carry	$A \leftarrow (A) + (M) + (C)$	↕	↕	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 B9 C9 D9 E9 F9 9ED9 9EE9	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 2 5 4
ADD <i>#opr8i</i> ADD <i>opr8a</i> ADD <i>opr16a</i> ADD <i>opr16,X</i> ADD <i>opr8,X</i> ADD <i>,X</i> ADD <i>opr16,SP</i> ADD <i>opr8,SP</i>	Add without Carry	$A \leftarrow (A) + (M)$	↕	↕	—	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP2 SP1	AB BB CB DB EB FB 9EDB 9EEB	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 2 5 4
AIS <i>#opr8i</i>	Add Immediate Value (Signed) to Stack Pointer	$SP \leftarrow (SP) + (M)$ M is sign extended to a 16-bit value	—	—	—	—	—	—	IMM	A7	ii	2
AIX <i>#opr8i</i>	Add Immediate Value (Signed) to Index Register (H:X)	$H:X \leftarrow (H:X) + (M)$ M is sign extended to a 16-bit value	—	—	—	—	—	—	IMM	AF	ii	2
AND <i>#opr8i</i> AND <i>opr8a</i> AND <i>opr16a</i> AND <i>opr16,X</i> AND <i>opr8,X</i> AND <i>,X</i> AND <i>opr16,SP</i> AND <i>opr8,SP</i>	Logical AND	$A \leftarrow (A) \& (M)$	0	—	—	↕	↕	—	IMM DIR EXT IX2 IX1 IX SP2 SP1	A4 B4 C4 D4 E4 F4 9ED4 9EE4	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 2 5 4
ASL <i>opr8a</i> ASLA ASLX ASL <i>opr8,X</i> ASL <i>,X</i> ASL <i>opr8,SP</i>	Arithmetic Shift Left (Same as LSL)		↕	—	—	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff ff ff	4 1 1 4 3 5

Addressing Modes

Table 4-10. Instruction Set Summary (Sheet 2 of 9)

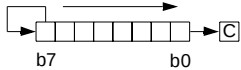
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ASR <i>opr8a</i> ASRA ASRX ASR <i>opr8,X</i> ASR <i>,X</i> ASR <i>opr8,SP</i>	Arithmetic Shift Right		↓	–	–	↓	↓	↓	DIR INH INH IX1 IX SP1	37 47 57 67 77 9E67	dd ff ff	4 1 1 4 3 5
BCC <i>rel</i>	Branch if Carry Bit Clear	Branch if (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BCLR <i>n,opr8a</i>	Clear Bit n in Memory	$M_n \leftarrow 0$	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BCS <i>rel</i>	Branch if Carry Bit Set (Same as BLO)	Branch if (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BEQ <i>rel</i>	Branch if Equal	Branch if (Z) = 1	–	–	–	–	–	–	REL	27	rr	3
BGE <i>rel</i>	Branch if Greater Than or Equal To (Signed Operands)	Branch if $(N \oplus V) = 0$	–	–	–	–	–	–	REL	90	rr	3
BGT <i>rel</i>	Branch if Greater Than (Signed Operands)	Branch if $(Z) \mid (N \oplus V) = 0$	–	–	–	–	–	–	REL	92	rr	3
BHCC <i>rel</i>	Branch if Half Carry Bit Clear	Branch if (H) = 0	–	–	–	–	–	–	REL	28	rr	3
BHCS <i>rel</i>	Branch if Half Carry Bit Set	Branch if (H) = 1	–	–	–	–	–	–	REL	29	rr	3
BHI <i>rel</i>	Branch if Higher	Branch if $(C) \mid (Z) = 0$	–	–	–	–	–	–	REL	22	rr	3
BHS <i>rel</i>	Branch if Higher or Same (Same as BCC)	Branch if (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH <i>rel</i>	Branch if IRQ Pin High	Branch if IRQ pin = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL <i>rel</i>	Branch if IRQ Pin Low	Branch if IRQ pin = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT <i>#opr8i</i> BIT <i>opr8a</i> BIT <i>opr16a</i> BIT <i>opr16,X</i> BIT <i>opr8,X</i> BIT <i>,X</i> BIT <i>opr16,SP</i> BIT <i>opr8,SP</i>	Bit Test	(A) & (M) (CCR Updated but Operands Not Changed)	0	–	–	↓	↓	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A5 B5 C5 D5 E5 F5 9ED5 9EE5	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 2 5 4
BLE <i>rel</i>	Branch if Less Than or Equal To (Signed Operands)	Branch if $(Z) \mid (N \oplus V) = 1$	–	–	–	–	–	–	REL	93	rr	3
BLO <i>rel</i>	Branch if Lower (Same as BCS)	Branch if (C) = 1	–	–	–	–	–	–	REL	25	rr	3

Table 4-10. Instruction Set Summary (Sheet 3 of 9)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BLS <i>rel</i>	Branch if Lower or Same	Branch if (C) (Z) = 1	–	–	–	–	–	–	REL	23	rr	3
BLT <i>rel</i>	Branch if Less Than (Signed Operands)	Branch if (N ⊕ V) = 1	–	–	–	–	–	–	REL	91	rr	3
BMC <i>rel</i>	Branch if Interrupt Mask Clear	Branch if (I) = 0	–	–	–	–	–	–	REL	2C	rr	3
BMI <i>rel</i>	Branch if Minus	Branch if (N) = 1	–	–	–	–	–	–	REL	2B	rr	3
BMS <i>rel</i>	Branch if Interrupt Mask Set	Branch if (I) = 1	–	–	–	–	–	–	REL	2D	rr	3
BNE <i>rel</i>	Branch if Not Equal	Branch if (Z) = 0	–	–	–	–	–	–	REL	26	rr	3
BPL <i>rel</i>	Branch if Plus	Branch if (N) = 0	–	–	–	–	–	–	REL	2A	rr	3
BRA <i>rel</i>	Branch Always	No Test	–	–	–	–	–	–	REL	20	rr	3
BRCLR <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Clear	Branch if (Mn) = 0	–	–	–	–	–	↑ ↓	DIR (b0)	01	dd rr	5
									DIR (b1)	03	dd rr	5
									DIR (b2)	05	dd rr	5
									DIR (b3)	07	dd rr	5
									DIR (b4)	09	dd rr	5
									DIR (b5)	0B	dd rr	5
									DIR (b6)	0D	dd rr	5
									DIR (b7)	0F	dd rr	5
BRN <i>rel</i>	Branch Never	Uses 3 Bus Cycles	–	–	–	–	–	–	REL	21	rr	3
BRSET <i>n,opr8a,rel</i>	Branch if Bit <i>n</i> in Memory Set	Branch if (Mn) = 1	–	–	–	–	–	↑ ↓	DIR (b0)	00	dd rr	5
									DIR (b1)	02	dd rr	5
									DIR (b2)	04	dd rr	5
									DIR (b3)	06	dd rr	5
									DIR (b4)	08	dd rr	5
									DIR (b5)	0A	dd rr	5
									DIR (b6)	0C	dd rr	5
									DIR (b7)	0E	dd rr	5
BSET <i>n,opr8a</i>	Set Bit <i>n</i> in Memory	Mn ← 1	–	–	–	–	–	–	DIR (b0)	10	dd	4
									DIR (b1)	12	dd	4
									DIR (b2)	14	dd	4
									DIR (b3)	16	dd	4
									DIR (b4)	18	dd	4
									DIR (b5)	1A	dd	4
									DIR (b6)	1C	dd	4
									DIR (b7)	1E	dd	4
BSR <i>rel</i>	Branch to Subroutine	PC ← (PC) + \$0002 push (PCL); SP ← (SP) – \$0001 push (PCH); SP ← (SP) – \$0001 PC ← (PC) + <i>rel</i>	–	–	–	–	–	–	REL	AD	rr	4
CBEQ <i>opr8a,rel</i>	Compare and Branch if Equal	Branch if (A) = (M)	–	–	–	–	–	–	DIR	31	dd rr	5
CBEQA <i>#opr8i,rel</i>		Branch if (A) = (M)							IMM	41	ii rr	4
CBEQX <i>#opr8i,rel</i>		Branch if (X) = (M)							IMM	51	ii rr	4
CBEQ <i>opr8,X+,rel</i>		Branch if (A) = (M)							IX1+	61	ff rr	5
CBEQ <i>,X+,rel</i>		Branch if (A) = (M)							IX+	71	rr	4
CBEQ <i>opr8,SP,rel</i>		Branch if (A) = (M)							SP1	9E61	ff rr	6

Addressing Modes

Table 4-10. Instruction Set Summary (Sheet 4 of 9)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
CLC	Clear Carry Bit	$C \leftarrow 0$	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask Bit	$I \leftarrow 0$	–	–	0	–	–	–	INH	9A		2
CLR <i>opr8a</i> CLRA CLR _X CLR _H CLR <i>opr8,X</i> CLR <i>,X</i> CLR <i>opr8,SP</i>	Clear	$M \leftarrow \$00$ $A \leftarrow \$00$ $X \leftarrow \$00$ $H \leftarrow \$00$ $M \leftarrow \$00$ $M \leftarrow \$00$ $M \leftarrow \$00$	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd ff ff	3 1 1 1 3 2 4
CMP <i>#opr8i</i> CMP <i>opr8a</i> CMP <i>opr16a</i> CMP <i>opr16,X</i> CMP <i>opr8,X</i> CMP <i>,X</i> CMP <i>opr16,SP</i> CMP <i>opr8,SP</i>	Compare Accumulator with Memory	$(A) - (M)$ (CCR Updated But Operands Not Changed)	↑	–	–	↑	↑	↑	IMM DIR EXT IX2 IX1 IX SP2 SP1	A1 B1 C1 D1 E1 F1 9ED1 9EE1	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 2 5 4
COM <i>opr8a</i> COMA COM _X COM <i>opr8,X</i> COM <i>,X</i> COM <i>opr8,SP</i>	Complement (One's Complement)	$M \leftarrow (\overline{M}) = \$FF - (M)$ $A \leftarrow (\overline{A}) = \$FF - (A)$ $X \leftarrow (\overline{X}) = \$FF - (X)$ $M \leftarrow (\overline{M}) = \$FF - (M)$ $M \leftarrow (\overline{M}) = \$FF - (M)$ $M \leftarrow (\overline{M}) = \$FF - (M)$	0	–	–	↑	↑	1	DIR INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd ff ff	4 1 1 4 3 5
CPHX <i>#opr</i> CPHX <i>opr</i>	Compare Index Register (H:X) with Memory	$(H:X) - (M: M + \$0001)$ (CCR Updated But Operands Not Changed)	↑	–	–	↑	↑	↑	IMM DIR	65 75	jj ii+1 dd	3 4
CPX <i>#opr8i</i> CPX <i>opr8a</i> CPX <i>opr16a</i> CPX <i>opr16,X</i> CPX <i>opr8,X</i> CPX <i>,X</i> CPX <i>opr16,SP</i> CPX <i>opr8,SP</i>	Compare X (Index Register Low) with Memory	$(X) - (M)$ (CCR Updated But Operands Not Changed)	↑	–	–	↑	↑	↑	IMM DIR EXT IX2 IX1 IX SP2 SP1	A3 B3 C3 D3 E3 F3 9ED3 9EE3	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 2 5 4
DAA	Decimal Adjust Accumulator After ADD or ADC of BCD Values	$(A)_{10}$	U	–	–	↑	↑	↑	INH	72		2
DBNZ <i>opr8a,rel</i> DBNZ _A <i>rel</i> DBNZ _X <i>rel</i> DBNZ <i>opr8,X,rel</i> DBNZ <i>,X,rel</i> DBNZ <i>opr8,SP,rel</i>	Decrement and Branch if Not Zero	Decrement A, X, or M Branch if (result) $\neq 0$ DBNZ _X Affects X Not H	–	–	–	–	–	–	DIR INH INH IX1 IX SP1	3B 4B 5B 6B 7B 9E6B	dd rr rr rr ff rr rr ff rr	5 3 3 5 4 6

Table 4-10. Instruction Set Summary (Sheet 5 of 9)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
DEC <i>opr8a</i> DECA DECX DEC <i>opr8,X</i> DEC <i>,X</i> DEC <i>opr8,SP</i>	Decrement	$M \leftarrow (M) - \$01$ $A \leftarrow (A) - \$01$ $X \leftarrow (X) - \$01$ $M \leftarrow (M) - \$01$ $M \leftarrow (M) - \$01$ $M \leftarrow (M) - \$01$	$\uparrow$	–	–	$\uparrow$	$\uparrow$	–	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd ff ff	4 1 1 4 3 5
DIV	Divide	$A \leftarrow (H:A) \div (X)$ H ← Remainder	–	–	–	–	$\uparrow$	$\uparrow$	INH	52		7
EOR <i>#opr8i</i> EOR <i>opr8a</i> EOR <i>opr16a</i> EOR <i>opr16,X</i> EOR <i>opr8,X</i> EOR <i>,X</i> EOR <i>opr16,SP</i> EOR <i>opr8,SP</i>	Exclusive OR Memory with Accumulator	$A \leftarrow (A \oplus M)$	0	–	–	$\uparrow$	$\uparrow$	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A8 B8 C8 D8 E8 F8 9ED8 9EE8	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 2 5 4
INC <i>opr8a</i> INCA INCX INC <i>opr8,X</i> INC <i>,X</i> INC <i>opr8,SP</i>	Increment	$M \leftarrow (M) + \$01$ $A \leftarrow (A) + \$01$ $X \leftarrow (X) + \$01$ $M \leftarrow (M) + \$01$ $M \leftarrow (M) + \$01$ $M \leftarrow (M) + \$01$	$\uparrow$	–	–	$\uparrow$	$\uparrow$	–	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff	4 1 1 4 3 5
JMP <i>opr8a</i> JMP <i>opr16a</i> JMP <i>opr16,X</i> JMP <i>opr8,X</i> JMP <i>,X</i>	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	2 3 4 3 3
JSR <i>opr8a</i> JSR <i>opr16a</i> JSR <i>opr16,X</i> JSR <i>opr8,X</i> JSR <i>,X</i>	Jump to Subroutine	PC ← (PC) + <i>n</i> (<i>n</i> = 1, 2, or 3) Push (PCL); SP ← (SP) – \$0001 Push (PCH); SP ← (SP) – \$0001 PC ← Unconditional Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff	4 5 6 5 4
LDA <i>#opr8i</i> LDA <i>opr8a</i> LDA <i>opr16a</i> LDA <i>opr16,X</i> LDA <i>opr8,X</i> LDA <i>,X</i> LDA <i>opr16,SP</i> LDA <i>opr8,SP</i>	Load Accumulator from Memory	$A \leftarrow (M)$	0	–	–	$\uparrow$	$\uparrow$	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A6 B6 C6 D6 E6 F6 9ED6 9EE6	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 2 5 4
LDHX <i>#opr</i> LDHX <i>opr</i>	Load Index Register (H:X) from Memory	$H:X \leftarrow (M:M + \$0001)$	0	–	–	$\uparrow$	$\uparrow$	–	IMM DIR	45 55	ii dd	3 4

Addressing Modes

Table 4-10. Instruction Set Summary (Sheet 6 of 9)

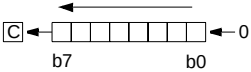
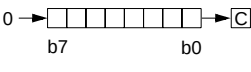
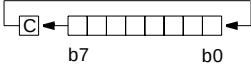
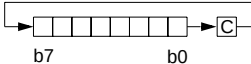
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
LDX #opr8i LDX opr8a LDX opr16a LDX oprx16,X LDX oprx8,X LDX ,X LDX oprx16,SP LDX oprx8,SP	Load X (Index Register Low) from Memory	$X \leftarrow (M)$	0	-	-	↑	↑	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	AE BE CE DE EE FE 9EDE 9EEE	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 2 5 4
LSL opr8a LSLA LSLX LSL oprx8,X LSL ,X LSL oprx8,SP	Logical Shift Left (Same as ASL)		↑	-	-	↑	↑	↑	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff ff	4 1 1 4 3 5
LSR opr8a LSRA LSRX LSR oprx8,X LSR ,X LSR oprx8,SP	Logical Shift Right		↑	-	-	0	↑	↑	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd dd ff ff	4 1 1 4 3 5
MOV opr8a,opr8a MOV opr8a,X+ MOV #opr8i,opr8a MOV ,X+,opr8a	Move	$(M)_{\text{destination}} \leftarrow (M)_{\text{source}}$ $H:X \leftarrow (H:X) + \$0001$ in IX+/DIR and DIR/IX+ Modes	0	-	-	↑	↑	-	DIR/DIR DIR/IX+ IMM/DIR IX+/DIR	4E 5E 6E 7E	dd dd dd ii dd dd	5 4 4 4
MUL	Unsigned multiply	$X:A \leftarrow (X) \times (A)$	-	0	-	-	-	0	INH	42		5
NEG opr8a NEGA NEGX NEG oprx8,X NEG ,X NEG oprx8,SP	Negate (Two's Complement)	$M \leftarrow -(M) = \$00 - (M)$ $A \leftarrow -(A) = \$00 - (A)$ $X \leftarrow -(X) = \$00 - (X)$ $M \leftarrow -(M) = \$00 - (M)$ $M \leftarrow -(M) = \$00 - (M)$ $M \leftarrow -(M) = \$00 - (M)$	↑	-	-	↑	↑	↑	DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	4 1 1 4 3 5
NOP	No Operation	Uses 1 Bus Cycle	-	-	-	-	-	-	INH	9D		1
NSA	Nibble Swap Accumulator	$A \leftarrow (A[3:0]:A[7:4])$	-	-	-	-	-	-	INH	62		3
ORA #opr8i ORA opr8a ORA opr16a ORA oprx16,X ORA oprx8,X ORA ,X ORA oprx16,SP ORA oprx8,SP	Inclusive OR Accumulator and Memory	$A \leftarrow (A) (M)$	0	-	-	↑	↑	-	IMM DIR EXT IX2 IX1 IX SP2 SP1	AA BA CA DA EA FA 9EDA 9EEA	ii dd hh ll ee ff ff ff ff	2 3 4 4 3 2 5 4
PSHA	Push Accumulator onto Stack	Push (A); $SP \leftarrow (SP) - \$0001$	-	-	-	-	-	-	INH	87		2
PSHH	Push H (Index Register High) onto Stack	Push (H); $SP \leftarrow (SP) - \$0001$	-	-	-	-	-	-	INH	8B		2

Table 4-10. Instruction Set Summary (Sheet 7 of 9)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
PSHX	Push X (Index Register Low) onto Stack	Push (X); $SP \leftarrow (SP) - \$0001$	-	-	-	-	-	-	INH	89		2
PULA	Pull Accumulator from Stack	$SP \leftarrow (SP + \$0001)$; Pull (A)	-	-	-	-	-	-	INH	86		2
PULH	Pull H (Index Register High) from Stack	$SP \leftarrow (SP + \$0001)$; Pull (H)	-	-	-	-	-	-	INH	8A		2
PULX	Pull X (Index Register Low) from Stack	$SP \leftarrow (SP + \$0001)$; Pull (X)	-	-	-	-	-	-	INH	88		2
ROL <i>opr8a</i> ROLA ROLX ROL <i>opr8,X</i> ROL <i>,X</i> ROL <i>opr8,SP</i>	Rotate Left through Carry		↑	-	-	↑	↑	↑	DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	4 1 1 4 3 5
ROR <i>opr8a</i> RORA RORX ROR <i>opr8,X</i> ROR <i>,X</i> ROR <i>opr8,SP</i>	Rotate Right through Carry		↑	-	-	↑	↑	↑	DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	4 1 1 4 3 5
RSP	Reset Stack Pointer	$SP \leftarrow \$FF$ (High Byte Not Affected)	-	-	-	-	-	-	INH	9C		1
RTI	Return from Interrupt	$SP \leftarrow (SP) + \$0001$; Pull (CCR) $SP \leftarrow (SP) + \$0001$; Pull (A) $SP \leftarrow (SP) + \$0001$; Pull (X) $SP \leftarrow (SP) + \$0001$; Pull (PCH) $SP \leftarrow (SP) + \$0001$; Pull (PCL)	↑	↑	↑	↑	↑	↑	INH	80		7
RTS	Return from Subroutine	$SP \leftarrow SP + \$0001$; Pull (PCH) $SP \leftarrow SP + \$0001$; Pull (PCL)	-	-	-	-	-	-	INH	81		4
SBC <i>#opr8i</i> SBC <i>opr8a</i> SBC <i>opr16a</i> SBC <i>opr16,X</i> SBC <i>opr8,X</i> SBC <i>,X</i> SBC <i>opr16,SP</i> SBC <i>opr8,SP</i>	Subtract with Carry	$A \leftarrow (A) - (M) - (C)$	↑	-	-	↑	↑	↑	IMM DIR EXT IX2 IX1 IX SP2 SP1	A2 B2 C2 D2 E2 F2 9ED2 9EE2	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 2 5 4
SEC	Set Carry Bit	$C \leftarrow 1$	-	-	-	-	-	1	INH	99		1
SEI	Set Interrupt Mask Bit	$I \leftarrow 1$	-	-	1	-	-	-	INH	9B		2
STA <i>opr8a</i> STA <i>opr16a</i> STA <i>opr16,X</i> STA <i>opr8,X</i> STA <i>,X</i> STA <i>opr16,SP</i> STA <i>opr8,SP</i>	Store Accumulator in Memory	$M \leftarrow (A)$	0	-	-	↑	↑	-	DIR EXT IX2 IX1 IX SP2 SP1	B7 C7 D7 E7 F7 9ED7 9EE7	dd hh ll ee ff ff ff ee ff ff	3 4 4 3 2 5 4

Addressing Modes

Table 4-10. Instruction Set Summary (Sheet 8 of 9)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
STHX <i>opr</i>	Store H:X (Index Reg.)	$(M:M + \$0001) \leftarrow (H:X)$	0	–	–	↑	↑	–	DIR	35	dd	4
STOP	Enable Interrupts: Stop Processing Refer to MCU Documentation	$I \text{ bit} \leftarrow 0$; Stop Processing	–	–	0	–	–	–	INH	8E		1
STX <i>opr8a</i> STX <i>opr16a</i> STX <i>opr16,X</i> STX <i>opr8,X</i> STX <i>,X</i> STX <i>opr16,SP</i> STX <i>opr8,SP</i>	Store X (Low 8 Bits of Index Register) in Memory	$M \leftarrow (X)$	0	–	–	↑	↑	–	DIR EXT IX2 IX1 IX SP2 SP1	BF CF DF EF FF 9EDF 9EEF	dd hh ll ee ff ff ff ee ff ff	3 4 4 3 2 5 4
SUB <i>#opr8i</i> SUB <i>opr8a</i> SUB <i>opr16a</i> SUB <i>opr16,X</i> SUB <i>opr8,X</i> SUB <i>,X</i> SUB <i>opr16,SP</i> SUB <i>opr8,SP</i>	Subtract	$A \leftarrow (A) - (M)$	↑	–	–	↑	↑	↑	IMM DIR EXT IX2 IX1 IX SP2 SP1	A0 B0 C0 D0 E0 F0 9ED0 9EE0	ii dd hh ll ee ff ff ff ee ff ff	2 3 4 4 3 2 5 4
SWI	Software Interrupt	$PC \leftarrow (PC) + \$0001$ Push (PCL); $SP \leftarrow (SP) - \$0001$ Push (PCH); $SP \leftarrow (SP) - \$0001$ Push (X); $SP \leftarrow (SP) - \$0001$ Push (A); $SP \leftarrow (SP) - \$0001$ Push (CCR); $SP \leftarrow (SP) - \$0001$ $I \leftarrow 1$; PCH $\leftarrow$ Interrupt Vector High Byte PCL $\leftarrow$ Interrupt Vector Low Byte	–	–	1	–	–	–	INH	83		9
TAP	Transfer Accumulator to CCR	$CCR \leftarrow (A)$	↑	↑	↑	↑	↑	↑	INH	84		2
TAX	Transfer Accumulator to X (Index Register Low)	$X \leftarrow (A)$	–	–	–	–	–	–	INH	97		1
TPA	Transfer CCR to Accumulator	$A \leftarrow (CCR)$	–	–	–	–	–	–	INH	85		1
TST <i>opr8a</i> TSTA TSTX TST <i>opr8,X</i> TST <i>,X</i> TST <i>opr8,SP</i>	Test for Negative or Zero	$(M) - \$00$ $(A) - \$00$ $(X) - \$00$ $(M) - \$00$ $(M) - \$00$ $(M) - \$00$	0	–	–	↑	↑	–	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff ff	3 1 1 3 2 4
TSX	Transfer SP to Index Reg.	$H:X \leftarrow (SP) + \$0001$	–	–	–	–	–	–	INH	95		2
TXA	Transfer X (Index Reg. Low) to Accumulator	$A \leftarrow (X)$	–	–	–	–	–	–	INH	9F		1

Table 4-10. Instruction Set Summary (Sheet 9 of 9)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
TXS	Transfer Index Reg. to SP	$SP \leftarrow (H:X) - \$0001$	–	–	–	–	–	–	INH	94		2
WAIT	Enable Interrupts; Wait for Interrupt	I bit $\leftarrow$ 0; Halt CPU	–	–	0	–	–	–	INH	8F		1

A	Accumulator	<i>n</i>	Any bit
C	Carry/borrow bit	<i>opr</i>	Operand (one or two bytes)
CCR	Condition code register	PC	Program counter
dd	Direct address of operand	PCH	Program counter high byte
dd rr	Direct address of operand and relative offset of branch instruction	PCL	Program counter low byte
DD	Direct to direct addressing mode	REL	Relative addressing mode
DIR	Direct addressing mode	<i>rel</i>	Relative program counter offset byte
DIX+	Direct to indexed with post increment addressing mode	rr	Relative program counter offset byte
ee ff	High and low bytes of offset in indexed, 16-bit offset addressing	SP1	Stack pointer, 8-bit offset addressing mode
EXT	Extended addressing mode	SP2	Stack pointer 16-bit offset addressing mode
ff	Offset byte in indexed, 8-bit offset addressing	SP	Stack pointer
H	Half-carry bit	U	Undefined
H	Index register high byte	V	Overflow bit
hh ll	High and low bytes of operand address in extended addressing	X	Index register low byte
I	Interrupt mask	Z	Zero bit
ii	Immediate operand byte	&	Logical AND
IMD	Immediate source to direct destination addressing mode		Logical OR
IMM	Immediate addressing mode	⊕	Logical EXCLUSIVE OR
INH	Inherent addressing mode	()	Contents of
IX	Indexed, no offset addressing mode	–()	Negation (two's complement)
IX+	Indexed, no offset, post increment addressing mode	#	Immediate value
IX+D	Indexed with post increment to direct addressing mode	«	Sign extend
IX1	Indexed, 8-bit offset addressing mode	←	Loaded with
IX1+	Indexed, 8-bit offset, post increment addressing mode	?	If
IX2	Indexed, 16-bit offset addressing mode	:	Concatenated with
M	Memory location	↑	Set or cleared
N	Negative bit	—	Not affected

4.5 Opcode Map

The opcode map is provided in [Table 4-11](#).

Addressing Modes

Table 4-11. Opcode Map

	Bit-Manipulation			Branch		Read-Modify-Write					Control			Register/Memory						
	DIR	DIR	REL	DIR	INH	INH	INH	IX1	SP1	IX	INH	INH	IMM	DIR	EXT	IX2	SP2	IX1	SP1	IX
HIGH	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	9ED	9EE	F		
LOW																				
0	BRSET0 ⁵ DIR	BSET0 ⁴ DIR	BRA ³ REL	NEG ² DIR	NEGA ¹ INH	NEG ¹ INH	NEG ¹ INH	NEG ¹ INH	NEG ¹ INH	NEG ¹ INH	RTI ⁷ INH	BGE ³ REL	SUB ² IMM	SUB ³ DIR	SUB ⁴ EXT	SUB ⁴ EXT	SUB ⁵ EXT	SUB ⁵ EXT	SUB ⁵ EXT	SUB ⁵ EXT
1	BRCLR0 ⁵ DIR	BCLR0 ⁴ DIR	BRN ³ REL	CBEQ ² DIR	CBEQA ¹ INH	CBEQ ¹ INH	CBEQ ¹ INH	CBEQ ¹ INH	CBEQ ¹ INH	CBEQ ¹ INH	RTS ⁴ INH	BLT ³ REL	CMP ² IMM	CMP ³ DIR	CMP ⁴ EXT	CMP ⁴ EXT	CMP ⁵ EXT	CMP ⁵ EXT	CMP ⁵ EXT	CMP ⁵ EXT
2	BRSET1 ⁵ DIR	BSET1 ⁴ DIR	BHI ³ REL	MUL ² DIR	MUL ¹ INH	DIV ¹ INH	NSA ¹ INH	NSA ¹ INH	NSA ¹ INH	NSA ¹ INH		BGT ³ REL	SBC ² IMM	SBC ³ DIR	SBC ⁴ EXT	SBC ⁴ EXT	SBC ⁵ EXT	SBC ⁵ EXT	SBC ⁵ EXT	SBC ⁵ EXT
3	BRCLR1 ⁵ DIR	BCLR1 ⁴ DIR	BLS ³ REL	COM ² DIR	COMA ¹ INH	COMX ¹ INH	COM ¹ INH	COM ¹ INH	COM ¹ INH	COM ¹ INH	SWI ⁹ INH	BLE ³ REL	CPX ² IMM	CPX ³ DIR	CPX ⁴ EXT	CPX ⁴ EXT	CPX ⁵ EXT	CPX ⁵ EXT	CPX ⁵ EXT	CPX ⁵ EXT
4	BRSET2 ⁵ DIR	BSET2 ⁴ DIR	BCC ³ REL	LSR ² DIR	LSRA ¹ INH	LSRX ¹ INH	LSR ¹ INH	LSR ¹ INH	LSR ¹ INH	LSR ¹ INH	TAP ² INH	TXS ² REL	AND ² IMM	AND ³ DIR	AND ⁴ EXT	AND ⁴ EXT	AND ⁵ EXT	AND ⁵ EXT	AND ⁵ EXT	AND ⁵ EXT
5	BRCLR2 ⁵ DIR	BCLR2 ⁴ DIR	BCS ³ REL	STHX ² DIR	LDHX ¹ INH	LDHX ¹ INH	CPHX ¹ IMM	CPHX ¹ IMM	CPHX ¹ IMM	CPHX ¹ IMM	TPA ¹ INH	TSX ² REL	BIT ² IMM	BIT ³ DIR	BIT ⁴ EXT	BIT ⁴ EXT	BIT ⁵ EXT	BIT ⁵ EXT	BIT ⁵ EXT	BIT ⁵ EXT
6	BRSET3 ⁵ DIR	BSET3 ⁴ DIR	BNE ³ REL	ROR ² DIR	RORA ¹ INH	RORX ¹ INH	ROR ¹ INH	ROR ¹ INH	ROR ¹ INH	ROR ¹ INH	PULA ² INH		LDA ² IMM	LDA ³ DIR	LDA ⁴ EXT	LDA ⁴ EXT	LDA ⁵ EXT	LDA ⁵ EXT	LDA ⁵ EXT	LDA ⁵ EXT
7	BRCLR3 ⁵ DIR	BCLR3 ⁴ DIR	BEQ ³ REL	ASR ² DIR	ASRA ¹ INH	ASRX ¹ INH	ASR ¹ INH	ASR ¹ INH	ASR ¹ INH	ASR ¹ INH	PSHA ² INH	TAX ¹ INH	AIS ² IMM	STA ² DIR	STA ³ EXT	STA ⁴ EXT	STA ⁵ EXT	STA ⁵ EXT	STA ⁵ EXT	STA ⁵ EXT
8	BRSET4 ⁵ DIR	BSET4 ⁴ DIR	BHCC ³ REL	LSL ² DIR	LSLA ¹ INH	LSLX ¹ INH	LSL ¹ INH	LSL ¹ INH	LSL ¹ INH	LSL ¹ INH	PULX ² INH	CLC ¹ INH	EOR ² IMM	EOR ³ DIR	EOR ⁴ EXT	EOR ⁴ EXT	EOR ⁵ EXT	EOR ⁵ EXT	EOR ⁵ EXT	EOR ⁵ EXT
9	BRCLR4 ⁵ DIR	BCLR4 ⁴ DIR	BHCS ³ REL	ROL ² DIR	ROLA ¹ INH	ROLX ¹ INH	ROL ¹ INH	ROL ¹ INH	ROL ¹ INH	ROL ¹ INH	PSHX ² INH	SEC ¹ INH	ADC ² IMM	ADC ³ DIR	ADC ⁴ EXT	ADC ⁴ EXT	ADC ⁵ EXT	ADC ⁵ EXT	ADC ⁵ EXT	ADC ⁵ EXT
A	BRSET5 ⁵ DIR	BSET5 ⁴ DIR	BPL ³ REL	DEC ² DIR	DECA ¹ INH	DECX ¹ INH	DEC ¹ INH	DEC ¹ INH	DEC ¹ INH	DEC ¹ INH	PULH ² INH	CLI ¹ INH	ORA ² IMM	ORA ³ DIR	ORA ⁴ EXT	ORA ⁴ EXT	ORA ⁵ EXT	ORA ⁵ EXT	ORA ⁵ EXT	ORA ⁵ EXT
B	BRCLR5 ⁵ DIR	BCLR5 ⁴ DIR	BMI ³ REL	DBNZ ² DIR	DBNZA ¹ INH	DBNZX ¹ INH	DBNZ ¹ INH	DBNZ ¹ INH	DBNZ ¹ INH	DBNZ ¹ INH	PSHH ² INH	SEI ¹ INH	ADD ² IMM	ADD ³ DIR	ADD ⁴ EXT	ADD ⁴ EXT	ADD ⁵ EXT	ADD ⁵ EXT	ADD ⁵ EXT	ADD ⁵ EXT
C	BRSET6 ⁵ DIR	BSET6 ⁴ DIR	BMC ³ REL	INC ² DIR	INCA ¹ INH	INCX ¹ INH	INC ¹ INH	INC ¹ INH	INC ¹ INH	INC ¹ INH	CLR ² INH	RSP ¹ INH	JMP ² IMM	JMP ³ DIR	JMP ⁴ EXT	JMP ⁴ EXT	JMP ⁵ EXT	JMP ⁵ EXT	JMP ⁵ EXT	JMP ⁵ EXT
D	BRCLR6 ⁵ DIR	BCLR6 ⁴ DIR	BMS ³ REL	TST ² DIR	TSTA ¹ INH	TSTX ¹ INH	TST ¹ INH	TST ¹ INH	TST ¹ INH	TST ¹ INH		NOP ¹ INH	BSR ² REL	JSR ³ DIR	JSR ⁴ EXT	JSR ⁴ EXT	JSR ⁵ EXT	JSR ⁵ EXT	JSR ⁵ EXT	JSR ⁵ EXT
E	BRSET7 ⁵ DIR	BSET7 ⁴ DIR	BIL ³ REL		MOV ² DIR	MOV ¹ DIR	MOV ¹ DIR	MOV ¹ DIR	MOV ¹ DIR	MOV ¹ DIR	STOP ¹ INH	*	LDX ² IMM	LDX ³ DIR	LDX ⁴ EXT	LDX ⁴ EXT	LDX ⁵ EXT	LDX ⁵ EXT	LDX ⁵ EXT	LDX ⁵ EXT
F	BRCLR7 ⁵ DIR	BCLR7 ⁴ DIR	BIH ³ REL	CLR ² DIR	CLRA ¹ INH	CLR ¹ INH	CLR ¹ INH	CLR ¹ INH	CLR ¹ INH	CLR ¹ INH	WAIT ¹ INH	TXA ¹ INH	AIX ² IMM	STX ³ DIR	STX ⁴ EXT	STX ⁴ EXT	STX ⁵ EXT	STX ⁵ EXT	STX ⁵ EXT	STX ⁵ EXT

INH

IMM

DIR

EXT

DD

IX+D

REL

IX1

IX2

IMD

DIR

DIR

SP1

SP2

IX+

IX1+

Stack Pointer, 8-Bit Offset

Stack Pointer, 16-Bit Offset

Indexed, No Offset with Post Increment

Indexed, 1-Byte Offset with Post Increment

High Byte of Opcode in Hexadecimal

Low Byte of Opcode in Hexadecimal

F

SUB

HC08 Cycles

Opcode Mnemonic

Number of Bytes / Addressing Mode

Section 5. Instruction Set

5.1 Contents

5.2	Introduction	92
5.3	Nomenclature	92
5.4	Convention Definitions	96
5.5	Instruction Set.	96
ADC	Add with Carry	97
ADD	Add without Carry	98
AIS	Add Immediate Value (Signed) to Stack Pointer	99
AIX	Add Immediate Value (Signed) to Index Register	100
AND	Logical AND	101
ASL	Arithmetic Shift Left	102
ASR	Arithmetic Shift Right	103
BCC	Branch if Carry Bit Clear	104
BCLR <i>n</i>	Clear Bit <i>n</i> in Memory	105
BCS	Branch if Carry Bit Set	106
BEQ	Branch if Equal	107
BGE	Branch if Greater Than or Equal To	108
BGT	Branch if Greater Than	109
BHCC	Branch if Half Carry Bit Clear	110
BHCS	Branch if Half Carry Bit Set	111
BHI	Branch if Higher	112
BHS	Branch if Higher or Same	113
BIH	Branch if $\overline{\text{IRQ}}$ Pin High	114
BIL	Branch if $\overline{\text{IRQ}}$ Pin Low	115
BIT	Bit Test	116
BLE	Branch if Less Than or Equal To	117
BLO	Branch if Lower	118

BLS	Branch if Lower or Same	119
BLT	Branch if Less Than	120
BMC	Branch if Interrupt Mask Clear	121
BMI	Branch if Minus	122
BMS	Branch if Interrupt Mask Set	123
BNE	Branch if Not Equal	124
BPL	Branch if Plus	125
BRA	Branch Always	126
BRA	Branch Always	127
BRCLR <i>n</i>	Branch if Bit <i>n</i> in Memory Clear	128
BRN	Branch Never	129
BRSET <i>n</i>	Branch if Bit <i>n</i> in Memory Set	130
BSET <i>n</i>	Set Bit <i>n</i> in Memory	131
BSR	Branch to Subroutine	132
CBEQ	Compare and Branch if Equal	133
CLC	Clear Carry Bit	134
CLI	Clear Interrupt Mask Bit	135
CLR	Clear	136
CMP	Compare Accumulator with Memory	137
COM	Complement (One's Complement)	138
CPHX	Compare Index Register with Memory	139
CPX	Compare X (Index Register Low) with Memory	140
DAA	Decimal Adjust Accumulator	141
DAA	Decimal Adjust Accumulator (Continued)	142
DBNZ	Decrement and Branch if Not Zero	143
DEC	Decrement	144
DIV	Divide	145
EOR	Exclusive-OR Memory with Accumulator	146
INC	Increment	147
JMP	Jump	148
JSR	Jump to Subroutine	149
LDA	Load Accumulator from Memory	150
LDHX	Load Index Register from Memory	151
LDX	Load X (Index Register Low) from Memory	152
LSL	Logical Shift Left	153
LSR	Logical Shift Right	154
MOV	Move	155

MUL	Unsigned Multiply	156
NEG	Negate (Two's Complement).	157
NOP	No Operation	158
NSA	Nibble Swap Accumulator	159
ORA	Inclusive-OR Accumulator and Memory	160
PSHA	Push Accumulator onto Stack	161
PSHH	Push H (Index Register High) onto Stack	162
PSHX	Push X (Index Register Low) onto Stack	163
PULA	Pull Accumulator from Stack	164
PULH	Pull H (Index Register High) from Stack	165
PULX	Pull X (Index Register Low) from Stack.	166
ROL	Rotate Left through Carry	167
ROR	Rotate Right through Carry	168
RSP	Reset Stack Pointer.	169
RTI	Return from Interrupt.	170
RTS	Return from Subroutine.	171
SBC	Subtract with Carry	172
SEC	Set Carry Bit	173
SEI	Set Interrupt Mask Bit	174
STA	Store Accumulator in Memory.	175
STHX	Store Index Register	176
STOP	Enable $\overline{\text{IRQ}}$ Pin, Stop Oscillator	177
STX	Store X (Index Register Low) in Memory	178
SUB	Subtract.	179
SWI	Software Interrupt	180
TAP	Transfer Accumulator to Processor Status Byte	181
TAX	Transfer Accumulator to X (Index Register Low)	182
TPA	Transfer Processor Status Byte to Accumulator	183
TST	Test for Negative or Zero	184
TSX	Transfer Stack Pointer to Index Register	185
TXA	Transfer X (Index Register Low) to Accumulator	186
TXS	Transfer Index Register to Stack Pointer	187
WAIT	Enable Interrupts; Stop Processor	188

5.2 Introduction

This section contains detailed information for all HC08 Family instructions. The instructions are arranged in alphabetical order with the instruction mnemonic set in larger type for easy reference.

5.3 Nomenclature

This nomenclature is used in the instruction descriptions throughout this section.

Operators

()	=	Contents of register or memory location shown inside parentheses
←	=	Is loaded with (read: “gets”)
&	=	Boolean AND
	=	Boolean OR
⊕	=	Boolean exclusive-OR
×	=	Multiply
÷	=	Divide
:	=	Concatenate
+	=	Add
−	=	Negate (two's complement)
«	=	Sign extend

CPU registers

A	=	Accumulator
CCR	=	Condition code register
H	=	Index register, higher order (most significant) eight bits
X	=	Index register, lower order (least significant) eight bits
PC	=	Program counter
PCH	=	Program counter, higher order (most significant) eight bits
PCL	=	Program counter, lower order (least significant) eight bits
SP	=	Stack pointer

Memory and addressing

- M = A memory location or absolute data, depending on addressing mode
- $M:M + \$0001$ = A 16-bit value in two consecutive memory locations. The higher-order (most significant) eight bits are located at the address of M , and the lower-order (least significant) eight bits are located at the next higher sequential address.
- rel = The relative offset, which is the two's complement number stored in the last byte of machine code corresponding to a branch instruction

Condition code register (CCR) bits

- V = Two's complement overflow indicator, bit 7
- H = Half carry, bit 4
- I = Interrupt mask, bit 3
- N = Negative indicator, bit 2
- Z = Zero indicator, bit 1
- C = Carry/borrow, bit 0 (carry out of bit 7)

Bit status BEFORE execution of an instruction ($n = 7, 6, 5, \dots 0$)

For 2-byte operations such as LDHX, STHX, and CPHX, $n = 15$ refers to bit 15 of the 2-byte word or bit 7 of the most significant (first) byte.

- Mn = Bit n of memory location used in operation
- An = Bit n of accumulator
- Hn = Bit n of index register H
- Xn = Bit n of index register X
- bn = Bit n of the source operand (M , A , or X)

Bit status AFTER execution of an instruction

For 2-byte operations such as LDHX, STHX, and CPHX, $n = 15$ refers to bit 15 of the 2-byte word or bit 7 of the most significant (first) byte.

- Rn = Bit n of the result of an operation ($n = 7, 6, 5, \dots 0$)

CCR activity figure notation

–	=	Bit not affected
0	=	Bit forced to 0
1	=	Bit forced to 1
↕	=	Bit set or cleared according to results of operation
U	=	Undefined after the operation

Machine coding notation

dd	=	Low-order eight bits of a direct address \$0000–\$00FF (high byte assumed to be \$00)
ee	=	Upper eight bits of 16-bit offset
ff	=	Lower eight bits of 16-bit offset or 8-bit offset
ii	=	One byte of immediate data
jj	=	High-order byte of a 16-bit immediate data value
kk	=	Low-order byte of a 16-bit immediate data value
hh	=	High-order byte of 16-bit extended address
ll	=	Low-order byte of 16-bit extended address
rr	=	Relative offset

Source forms

The instruction detail pages provide only essential information about assembler source forms. Assemblers generally support a number of assembler directives, allow definition of program labels, and have special conventions for comments. For complete information about writing source files for a particular assembler, refer to the documentation provided by the assembler vendor.

Typically, assemblers are flexible about the use of spaces and tabs. Often, any number of spaces or tabs can be used where a single space is shown on the glossary pages. Spaces and tabs are also normally allowed before and after commas. When program labels are used, there must also be at least one tab or space before all instruction mnemonics. This required space is not apparent in the source forms.

Everything in the source forms columns, *except expressions in italic characters*, is literal information which must appear in the assembly source file exactly as shown. The initial 3- to 5-letter mnemonic is always

a literal expression. All commas, pound signs (#), parentheses, and plus signs (+) are literal characters.

The definition of a legal label or expression varies from assembler to assembler. Assemblers also vary in the way CPU registers are specified. Refer to assembler documentation for detailed information.

Recommended register designators are a, A, h, H, x, X, sp, and SP.

- n* — Any label or expression that evaluates to a single integer in the range 0–7
- opr8i* — Any label or expression that evaluates to an 8-bit immediate value
- opr16i* — Any label or expression that evaluates to a 16-bit immediate value
- opr8a* — Any label or expression that evaluates to an 8-bit value. The instruction treats this 8-bit value as the low order eight bits of an address in the direct page of the 64-Kbyte address space (\$00xx).
- opr16a* — Any label or expression that evaluates to a 16-bit value. The instruction treats this value as an address in the 64-Kbyte address space.
- opr8* — Any label or expression that evaluates to an unsigned 8-bit value; used for indexed addressing
- opr16* — Any label or expression that evaluates to a 16-bit value. Since the MC68HC08S has a 16-bit address bus, this can be either a signed or an unsigned value.
- rel* — Any label or expression that refers to an address that is within –128 to +127 locations from the next address after the last byte of object code for the current instruction. The assembler will calculate the 8-bit signed offset and include it in the object code for this instruction.

Address modes

INH	=	Inherent (no operands)
IMM	=	8-bit or 16-bit immediate
DIR	=	8-bit direct
EXT	=	16-bit extended
IX	=	16-bit indexed no offset
IX+	=	16-bit indexed no offset, post increment (CBEQ and MOV only)
IX1	=	16-bit indexed with 8-bit offset from H:X
IX1+	=	16-bit indexed with 8-bit offset, post increment (CBEQ only)
IX2	=	16-bit indexed with 16-bit offset from H:X
REL	=	8-bit relative offset
SP1	=	Stack pointer relative with 8-bit offset
SP2	=	Stack pointer relative with 16-bit offset

5.4 Convention Definitions

Set refers specifically to establishing logic level 1 on a bit or bits.

Cleared refers specifically to establishing logic level 0 on a bit or bits.

A specific bit is referred to by mnemonic and bit number. A7 is bit 7 of accumulator A. **A range of bits** is referred to by mnemonic and the bit numbers that define the range. A [7:4] are bits 7 to 4 of the accumulator.

Parentheses indicate the contents of a register or memory location, rather than the register or memory location itself. (A) is the contents of the accumulator. In Boolean expressions, parentheses have the traditional mathematical meaning.

5.5 Instruction Set

The following pages summarize each instruction, including operation and description, condition codes and Boolean formulae, and a table with source forms, addressing modes, machine code, and cycles.

ADC

Add with Carry

ADC

Operation

$$A \leftarrow (A) + (M) + (C)$$

Description

Adds the contents of the C bit to the sum of the contents of A and M and places the result in A. This operation is useful for addition of operands that are larger than eight bits.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
↑	1	1	↑	—	↑	↑	↑

$$V: A7 \& M7 \& \overline{R7} \mid \overline{A7} \& \overline{M7} \& R7$$

Set if a two's complement overflow resulted from the operation; cleared otherwise

$$H: A3 \& M3 \mid M3 \& \overline{R3} \mid \overline{R3} \& A3$$

Set if there was a carry from bit 3; cleared otherwise

$$N: R7$$

Set if MSB of result is 1; cleared otherwise

$$Z: \overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$$

Set if result is \$00; cleared otherwise

$$C: A7 \& M7 \mid M7 \& \overline{R7} \mid \overline{R7} \& A7$$

Set if there was a carry from the most significant bit (MSB) of the result; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
ADC <i>#opr8i</i>	IMM	A9	ii	2
ADC <i>opr8a</i>	DIR	B9	dd	3
ADC <i>opr16a</i>	EXT	C9	hh ll	4
ADC <i>opr16,X</i>	IX2	D9	ee ff	4
ADC <i>opr8,X</i>	IX1	E9	ff	3
ADC <i>,X</i>	IX	F9		2
ADC <i>opr16,SP</i>	SP2	9ED9	ee ff	5
ADC <i>opr8,SP</i>	SP1	9EE9	ff	4

Instruction Set

ADD

Add without Carry

ADD

Operation

$$A \leftarrow (A) + (M)$$

Description

Adds the contents of M to the contents of A and places the result in A

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
↑	1	1	↑	—	↑	↑	↑

$$V: A7 \& M7 \& \overline{R7} \mid \overline{A7} \& \overline{M7} \& R7$$

Set if a two's complement overflow resulted from the operation; cleared otherwise

$$H: A3 \& M3 \mid M3 \& \overline{R3} \mid \overline{R3} \& A3$$

Set if there was a carry from bit 3; cleared otherwise

$$N: R7$$

Set if MSB of result is 1; cleared otherwise

$$Z: \overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$$

Set if result is \$00; cleared otherwise

$$C: A7 \& M7 \mid M7 \& \overline{R7} \mid \overline{R7} \& A7$$

Set if there was a carry from the MSB of the result; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
ADD #opr8i	IMM	AB	ii	2
ADD opr8a	DIR	BB	dd	3
ADD opr16a	EXT	CB	hh ll	4
ADD oprx16,X	IX2	DB	ee ff	4
ADD oprx8,X	IX1	EB	ff	3
ADD ,X	IX	FB		2
ADD oprx16,SP	SP2	9EDB	ee ff	5
ADD oprx8,SP	SP1	9EEB	ff	4

AIS

Add Immediate Value (Signed) to Stack Pointer

AIS

Operation

$$SP \leftarrow (SP) + (16 \ll M)$$

Description

Adds the immediate operand to the stack pointer (SP). The immediate value is an 8-bit two's complement signed operand. The 8-bit operand is sign-extended to 16 bits prior to the addition. The AIS instruction can be used to create and remove a stack frame buffer that is used to store temporary variables.

This instruction does not affect any condition code bits so status information can be passed to or from a subroutine or C function and allocation or deallocation of space for local variables will not disturb that status information.

Condition Codes
and Boolean
Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form,
Addressing Mode,
Machine Code,
Cycle, and Access
Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
AIS <i>#opr8i</i>	IMM	A7	ii	2

AIX**Add Immediate Value (Signed) to Index Register****AIX****Operation**

$$H:X \leftarrow (H:X) + (16 \ll M)$$
Description

Adds an immediate operand to the 16-bit index register, formed by the concatenation of the H and X registers. The immediate operand is an 8-bit two's complement signed offset. The 8-bit operand is sign-extended to 16 bits prior to the addition.

This instruction does not affect any condition code bits so index register pointer calculations do not disturb the surrounding code which may rely on the state of CCR status bits.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
AIX <i>#opr8i</i>	IMM	AF	ii	2

AND

Logical AND

AND

Operation

 $A \leftarrow (A) \& (M)$

Description

Performs the logical AND between the contents of A and the contents of M and places the result in A. Each bit of A after the operation will be the logical AND of the corresponding bits of M and of A before the operation.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
0	1	1	—	—	↑	↑	—

V: 0

Cleared

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

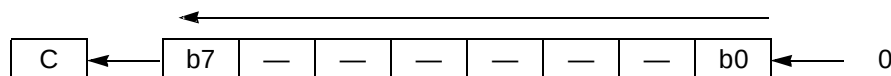
Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
AND <i>#opr8i</i>	IMM	A4	ii	2
AND <i>opr8a</i>	DIR	B4	dd	3
AND <i>opr16a</i>	EXT	C4	hh ll	4
AND <i>opr16,X</i>	IX2	D4	ee ff	4
AND <i>opr8,X</i>	IX1	E4	ff	3
AND <i>,X</i>	IX	F4		2
AND <i>opr16,SP</i>	SP2	9ED4	ee ff	5
AND <i>opr8,SP</i>	SP1	9EE4	ff	4

ASL

Arithmetic Shift Left
(Same as LSL)

ASL

Operation



Description

Shifts all bits of A, X, or M one place to the left. Bit 0 is loaded with a 0. The C bit in the CCR is loaded from the most significant bit of A, X, or M. This is mathematically equivalent to multiplication by two. The V bit indicates whether the sign of the result has changed.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
↓	1	1	—	—	↓	↓	↓

V: $R7 \oplus b7$

Set if the exclusive-OR of the resulting N and C flags is 1; cleared otherwise

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

C: b7

Set if, before the shift, the MSB of A, X, or M was set; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

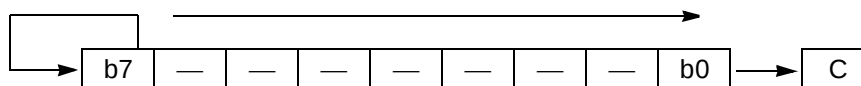
Source Form	Addr Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
ASL <i>opr8a</i>	DIR	38	dd	4
ASLA	INH (A)	48		1
ASLX	INH (X)	58		1
ASL <i>opr8,X</i>	IX1	68	ff	4
ASL <i>,X</i>	IX	78		3
ASL <i>opr8,SP</i>	SP1	9E68	ff	5

ASR

Arithmetic Shift Right

ASR

Operation



Description

Shifts all bits of A, X, or M one place to the right. Bit 7 is held constant. Bit 0 is loaded into the C bit of the CCR. This operation effectively divides a two's complement value by 2 without changing its sign. The carry bit can be used to round the result.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
↑	1	1	—	—	↑	↑	↑

V: $R7 \oplus b0$

Set if the exclusive-OR of the resulting N and C flags is 1; cleared otherwise

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

C: b0

Set if, before the shift, the LSB of A, X, or M was set; cleared otherwise

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
ASR <i>opr8a</i>	DIR	37	dd	4
ASRA	INH (A)	47		1
ASRX	INH (X)	57		1
ASR <i>opr8,X</i>	IX1	67	ff	4
ASR <i>,X</i>	IX	77		3
ASR <i>opr8,SP</i>	SP1	9E67	ff	5

BCC**Branch if Carry Bit Clear
(Same as BHS)****BCC****Operation**If (C) = 0, $PC \leftarrow (PC) + \$0002 + rel$

Simple branch

Description

Tests state of C bit in CCR and causes a branch if C is clear. BCC can be used after shift or rotate instructions or to check for overflow after operations on unsigned numbers. See the [BRA](#) instruction for further details of the execution of the branch.

**Condition Codes
and Boolean
Formulae**

None affected

V	H		I	N	Z	C
—	1	1	—	—	—	—

**Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail**

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BCC <i>rel</i>	REL	24	rr	3

See the [BRA](#) instruction for a summary of all branches and their complements.

BCLR *n***Clear Bit *n* in Memory****BCLR *n*****Operation** $Mn \leftarrow 0$ **Description**

Clear bit *n* (*n* = 7, 6, 5, ... 0) in location M. All other bits in M are unaffected. In other words, M can be any random-access memory (RAM) or input/output (I/O) register address in the \$0000 to \$00FF area of memory. (Direct addressing mode is used to specify the address of the operand.) This instruction reads the specified 8-bit location, modifies the specified bit, and then writes the modified 8-bit value back to the memory location.

**Condition Codes
and Boolean
Formulae**

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

**Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details**

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BCLR 0, <i>opr8a</i>	DIR (b0)	11	dd	4
BCLR 1, <i>opr8a</i>	DIR (b1)	13	dd	4
BCLR 2, <i>opr8a</i>	DIR (b2)	15	dd	4
BCLR 3, <i>opr8a</i>	DIR (b3)	17	dd	4
BCLR 4, <i>opr8a</i>	DIR (b4)	19	dd	4
BCLR 5, <i>opr8a</i>	DIR (b5)	1B	dd	4
BCLR 6, <i>opr8a</i>	DIR (b6)	1D	dd	4
BCLR 7, <i>opr8a</i>	DIR (b7)	1F	dd	4

Instruction Set

BCS

Branch if Carry Bit Set
(Same as BLO)

BCS

Operation

If (C) = 1, $PC \leftarrow (PC) + \$0002 + rel$

Simple branch

Description

Tests the state of the C bit in the CCR and causes a branch if C is set. BCS can be used after shift or rotate instructions or to check for overflow after operations on unsigned numbers. See the [BRA](#) instruction for further details of the execution of the branch.

Condition Codes
and Boolean
Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BCS <i>rel</i>	REL	25	rr	3

See the [BRA](#) instruction for a summary of all branches and their complements.

BEQ

Branch if Equal

BEQ

Operation

If $(Z) = 1$, $PC \leftarrow (PC) + \$0002 + rel$

Simple branch; may be used with signed or unsigned operations

Description

Tests the state of the Z bit in the CCR and causes a branch if Z is set. Compare instructions perform a subtraction with two operands and produce an internal result without changing the original operands. If the two operands were equal, the internal result of the subtraction for the compare will be zero so the Z bit will be equal to one and the BEQ will cause a branch.

This instruction can also be used after a load or store without having to do a separate test or compare on the loaded value. See the [BRA](#) instruction for further details of the execution of the branch.

Condition Codes
and Boolean
Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BEQ <i>rel</i>	REL	27	rr	3

See the [BRA](#) instruction for a summary of all branches and their complements.

Instruction Set

BGE**Branch if Greater Than or Equal To****BGE****Operation**

If $(N \oplus V) = 0$, $PC \leftarrow (PC) + \$0002 + rel$

For signed two's complement values
if (Accumulator) $\geq$ (Memory), then branch

Description

If the BGE instruction is executed immediately after execution of a CMP, CPHX, CPX, SBC, or SUB instruction, the branch occurs if and only if the two's complement number in the A, X, or H:X register was greater than or equal to the two's complement number in memory.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BGE <i>rel</i>	REL	90	rr	3

See the [BRA](#) instruction for a summary of all branches and their complements.

BGT**Branch if Greater Than****BGT****Operation**

If $(Z) \mid (N \oplus V) = 0$, $PC \leftarrow (PC) + \$0002 + rel$

For signed two's complement values
if (Accumulator) > (Memory), then branch

Description

If the BGT instruction is executed immediately after execution of a CMP, CPHX, CPX, SBC, or SUB instruction, the branch will occur if and only if the two's complement number in the A, X, or H:X register was greater than the two's complement number in memory.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BGT <i>rel</i>	REL	92	rr	3

See the **BRA** instruction for a summary of all branches and their complements.

BHCC

Branch if Half Carry Bit Clear

BHCC

Operation

If (H) = 0, $PC \leftarrow (PC) + \$0002 + rel$

Description

Tests the state of the H bit in the CCR and causes a branch if H is clear. This instruction is used in algorithms involving BCD numbers that were originally written for the M68HC05 or M68HC08 devices. The DAA instruction in the HC08 simplifies operations on BCD numbers so BHCC and BHCS should not be needed in new programs. See the [BRA](#) instruction for further details of the execution of the branch.

Condition Codes
and Boolean
Formulae

None affected

V	H		I	N	Z	C
—	1	1	—	—	—	—

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BHCC <i>rel</i>	REL	28	rr	3

See the [BRA](#) instruction for a summary of all branches and their complements.

BHCS

Branch if Half Carry Bit Set

BHCS

Operation

If (H) = 1, $PC \leftarrow (PC) + \$0002 + rel$

Description

Tests the state of the H bit in the CCR and causes a branch if H is set. This instruction is used in algorithms involving BCD numbers that were originally written for the M68HC05 or M68HC08 devices. The DAA instruction in the HC08 simplifies operations on BCD numbers so BHCC and BHCS should not be needed in new programs. See the [BRA](#) instruction for further details of the execution of the branch.

Condition Codes
and Boolean
Formulae

None affected

V	H		I	N	Z	C
—	1	1	—	—	—	—

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BHCS <i>rel</i>	REL	29	rr	3

See the [BRA](#) instruction for a summary of all branches and their complements.

BHI**Branch if Higher****BHI****Operation**

If $(C) \mid (Z) = 0$, $PC \leftarrow (PC) + \$0002 + rel$

For unsigned values, if (Accumulator) > (Memory), then branch

Description

Causes a branch if both C and Z are cleared. If the BHI instruction is executed immediately after execution of a CMP, CPHX, CPX, SBC, or SUB instruction, the branch will occur if the unsigned binary number in the A, X, or H:X register was greater than unsigned binary number in memory. Generally not useful after CLR, COM, DEC, INC, LDA, LDHX, LDX, STA, STHX, STX, or TST because these instructions do not affect the carry bit in the CCR. See the **BRA** instruction for details of the execution of the branch.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BHI <i>rel</i>	REL	22	rr	3

See the **BRA** instruction for a summary of all branches and their complements.

BHS**Branch if Higher or Same
(Same as BCC)****BHS****Operation**

If (C) = 0, $PC \leftarrow (PC) + \$0002 + rel$

For unsigned values, if (Accumulator) $\geq$ (Memory), then branch

Description

If the BHS instruction is executed immediately after execution of a CMP, CPHX, CPX, SBC, or SUB instruction, the branch will occur if the unsigned binary number in the A, X, or H:X register was greater than or equal to the unsigned binary number in memory. Generally not useful after CLR, COM, DEC, INC, LDA, LDHX, LDX, STA, STHX, STX, or TST because these instructions do not affect the carry bit in the CCR. See the [BRA](#) instruction for further details of the execution of the branch.

**Condition Codes
and Boolean
Formulae**

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

**Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail**

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BHS <i>rel</i>	REL	24	rr	3

See the [BRA](#) instruction for a summary of all branches and their complements.

Instruction Set

BIH

Branch if $\overline{\text{IRQ}}$ Pin High

BIH

Operation

If $\overline{\text{IRQ}}$ pin = 1, $\text{PC} \leftarrow (\text{PC}) + \$0002 + \text{rel}$

Description

Tests the state of the external interrupt pin and causes a branch if the pin is high. See the **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BIH <i>rel</i>	REL	2F	rr	3

See the **BRA** instruction for a summary of all branches and their complements.

BIL**Branch if $\overline{\text{IRQ}}$ Pin Low****BIL****Operation**

If $\overline{\text{IRQ}}$ pin = 0, $\text{PC} \leftarrow (\text{PC}) + \$0002 + \text{rel}$

Description

Tests the state of the external interrupt pin and causes a branch if the pin is low. See the **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae

None affected

V	H		I	N	Z	C
—	1	1	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BIL <i>rel</i>	REL	2E	rr	3

See the **BRA** instruction for a summary of all branches and their complements.

BIT

Bit Test

BIT

Operation

(A) & (M)

Description

Performs the logical AND comparison of the contents of A and the contents of M and modifies the condition codes accordingly. Neither the contents of A nor M are altered. (Each bit of the result of the AND would be the logical AND of the corresponding bits of A and M.)

This instruction is typically used to see if a particular bit, or any of several bits, in a byte are 1s. A mask value is prepared with 1s in any bit positions that are to be checked. This mask may be in accumulator A or memory and the unknown value to be checked will be in memory or the accumulator A, respectively. After the BIT instruction, a BNE instruction will branch if any bits in the tested location that correspond to 1s in the mask were 1s.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
0	1	1	—	—	↑	↑	—

V: 0

Cleared

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BIT <i>#opr8i</i>	IMM	A5	ii	2
BIT <i>opr8a</i>	DIR	B5	dd	3
BIT <i>opr16a</i>	EXT	C5	hh ll	4
BIT <i>opr16,X</i>	IX2	D5	ee ff	4
BIT <i>opr8,X</i>	IX1	E5	ff	3
BIT <i>,X</i>	IX	F5		2
BIT <i>opr16,SP</i>	SP2	9ED5	ee ff	5
BIT <i>opr8,SP</i>	SP1	9EE5	ff	4

BLE

Branch if Less Than or Equal To

BLE

Operation

If $(Z) \mid (N \oplus V) = 1$, $PC \leftarrow (PC) + \$0002 + rel$

For signed two's complement numbers
if (Accumulator) $\leq$ (Memory), then branch

Description

If the BLE instruction is executed immediately after execution of a CMP, CPHX, CPX, SBC, or SUB instruction, the branch will occur if and only if the two's complement in the A, X, or H:X register was less than or equal to the two's complement number in memory.

Condition Codes and Boolean Formulae

None affected

V		H		I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BLE <i>rel</i>	REL	93	rr	3

See the [BRA](#) instruction for a summary of all branches and their complements.

Instruction Set

BLO

Branch if Lower

BLO

Operation

If (C) = 1, $PC \leftarrow (PC) + \$0002 + rel$

For unsigned values, if (Accumulator) < (Memory), then branch

Description

If the BLO instruction is executed immediately after execution of a CMP, CPHX, CPX, SBC, or SUB instruction, the branch will occur if the unsigned binary number in the A, X, or H:X register was less than the unsigned binary number in memory. Generally not useful after CLR, COM, DEC, INC, LDA, LDHX, LDX, STA, STHX, STX, or TST because these instructions do not affect the carry bit in the CCR. See the [BRA](#) instruction for further details of the execution of the branch.

Condition Codes
and Boolean
Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BLO <i>rel</i>	REL	25	rr	3

See the [BRA](#) instruction for a summary of all branches and their complements.

BLS**Branch if Lower or Same****BLS****Operation**

If $(C) \mid (Z) = 1$, $PC \leftarrow (PC) + \$0002 + rel$

For unsigned values, if $(Accumulator) \leq (Memory)$, then branch

Description

Causes a branch if (C is set) or (Z is set). If the BLS instruction is executed immediately after execution of a CMP, CPHX, CPX, SBC, or SUB instruction, the branch will occur if and only if the unsigned binary number in the A, X, or H:X register was less than or equal to the unsigned binary number in memory. Generally not useful after CLR, COM, DEC, INC, LDA, LDHX, LDX, STA, STHX, STX, or TST because these instructions do not affect the carry bit in the CCR. See the **BRA** instruction for further details of the execution of the branch.

**Condition Codes
and Boolean
Formulae**

None affected

V		H		I	N	Z	C
—	1	1	—	—	—	—	—

**Source Form,
Addressing Mode,
Machine Code,
Cycle, and Access
Detail**

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BLS <i>rel</i>	REL	23	rr	3

See the **BRA** instruction for a summary of all branches and their complements.

Instruction Set

BLT

Branch if Less Than
(Signed Operands)

BLT

Operation

If $(N \oplus V) = 1$, $PC \leftarrow (PC) + \$0002 + rel$

For signed two's complement numbers
if (Accumulator) < (Memory), then branch

Description

If the BLT instruction is executed immediately after execution of a CMP, CPHX, CPX, SBC, or SUB instruction, the branch will occur if and only if the two's complement number in the A, X, or H:X register was less than the two's complement number in memory. See the [BRA](#) instruction for further details of the execution of the branch.

Condition Codes
and Boolean
Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BLT <i>rel</i>	REL	91	rr	3

See the [BRA](#) instruction for a summary of all branches and their complements.

BMC**Branch if Interrupt Mask Clear****BMC****Operation**If (I) = 0, $PC \leftarrow (PC) + \$0002 + rel$ **Description**

Tests the state of the I bit in the CCR and causes a branch if I is clear (if interrupts are enabled). See the **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BMC <i>rel</i>	REL	2C	rr	3

See the **BRA** instruction for a summary of all branches and their complements.

BMI**Branch if Minus****BMI****Operation**

If (N) = 1, $PC \leftarrow (PC) + \$0002 + rel$

Simple branch; may be used with signed or unsigned operations

Description

Tests the state of the N bit in the CCR and causes a branch if N is set.

Simply loading or storing A, X, or H:X will cause the N condition code bit to be set or cleared to match the most significant bit of the value loaded or stored. The BMI instruction can be used after such a load or store without having to do a separate test or compare instruction before the conditional branch. See the [BRA](#) instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BMI <i>rel</i>	REL	2B	rr	3

See the [BRA](#) instruction for a summary of all branches and their complements.

BMS

Branch if Interrupt Mask Set

BMS

Operation

If (I) = 1, $PC \leftarrow (PC) + \$0002 + rel$

Description

Tests the state of the I bit in the CCR and causes a branch if I is set (if interrupts are disabled). See [BRA](#) instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae

None affected

V	H		I	N	Z	C
—	1	1	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BMS <i>rel</i>	REL	2D	rr	3

See the [BRA](#) instruction for a summary of all branches and their complements.

Instruction Set

BNE

Branch if Not Equal

BNE

Operation

If $(Z) = 0$, $PC \leftarrow (PC) + \$0002 + rel$

Simple branch, may be used with signed or unsigned operations

Description

Tests the state of the Z bit in the CCR and causes a branch if Z is clear

Following a compare or subtract instruction, the branch will occur if the arguments were not equal. This instruction can also be used after a load or store without having to do a separate test or compare on the loaded value. See the **BRA** instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae

None affected

V	H		I	N	Z	C
—	1	1	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BNE <i>rel</i>	REL	26	rr	3

See the **BRA** instruction for a summary of all branches and their complements.

BPL**Branch if Plus****BPL****Operation**If (N) = 0, $PC \leftarrow (PC) + \$0002 + rel$

Simple branch

Description

Tests the state of the N bit in the CCR and causes a branch if N is clear

Simply loading or storing A, X, or H:X will cause the N condition code bit to be set or cleared to match the most significant bit of the value loaded or stored. The BPL instruction can be used after such a load or store without having to do a separate test or compare instruction before the conditional branch. See the **BRA** instruction for further details of the execution of the branch.

**Condition Codes
and Boolean
Formulae**

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

**Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail**

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BPL <i>rel</i>	REL	2A	rr	3

See the **BRA** instruction for a summary of all branches and their complements.

BRA**Branch Always****BRA****Operation**

$$PC \leftarrow (PC) + \$0002 + rel$$
Description

Performs an unconditional branch to the address given in the foregoing formula. In this formula, *rel* is the two's-complement relative offset in the last byte of machine code for the instruction and (PC) is the address of the opcode for the branch instruction.

A source program specifies the destination of a branch instruction by its absolute address, either as a numerical value or as a symbol or expression which can be numerically evaluated by the assembler. The assembler calculates the 8-bit relative offset *rel* from this absolute address and the current value of the location counter.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BRA <i>rel</i>	REL	20	rr	3

The table on the facing page is a summary of all branch instructions.

The BRA description continues next page.

BRA

Branch Always
(Continued)

BRA

Branch Instruction
Summary

Table 5-1 is a summary of all branch instructions.

Table 5-1. Branch Instruction Summary

Branch				Complementary Branch			Type
Test	Boolean	Mnemonic	Opcode	Test	Mnemonic	Opcode	
$r > m$	$(Z) \mid (N \oplus V) = 0$	BGT	92	$r \leq m$	BLE	93	Signed
$r \geq m$	$(N \oplus V) = 0$	BGE	90	$r < m$	BLT	91	Signed
$r = m$	$(Z) = 1$	BEQ	27	$r \neq m$	BNE	26	Signed
$r \leq m$	$(Z) \mid (N \oplus V) = 1$	BLE	93	$r > m$	BGT	92	Signed
$r < m$	$(N \oplus V) = 1$	BLT	91	$r \geq m$	BGE	90	Signed
$r > m$	$(C) \mid (Z) = 0$	BHI	22	$r \leq m$	BLS	23	Unsigned
$r \geq m$	$(C) = 0$	BHS/BCC	24	$r < m$	BLO/BCS	25	Unsigned
$r = m$	$(Z) = 1$	BEQ	27	$r \neq m$	BNE	26	Unsigned
$r \leq m$	$(C) \mid (Z) = 1$	BLS	23	$r > m$	BHI	22	Unsigned
$r < m$	$(C) = 1$	BLO/BCS	25	$r \geq m$	BHS/BCC	24	Unsigned
Carry	$(C) = 1$	BCS	25	No carry	BCC	24	Simple
result=0	$(Z) = 1$	BEQ	27	result $\neq$ 0	BNE	26	Simple
Negative	$(N) = 1$	BMI	2B	Plus	BPL	2A	Simple
I mask	$(I) = 1$	BMS	2D	I mask=0	BMC	2C	Simple
H-Bit	$(H) = 1$	BHCS	29	H=0	BHCC	28	Simple
$\overline{\text{IRQ}}$ high	—	BIH	2F	—	BIL	2E	Simple
Always	—	BRA	20	Never	BRN	21	Uncond.

r = register: A, X, or H:X (for CPHX instruction) m = memory operand

During program execution, if the tested condition is true, the two's complement offset is sign-extended to a 16-bit value which is added to the current program counter. This causes program execution to continue at the address specified as the branch destination. If the tested condition is not true, the program simply continues to the next instruction after the branch.

Instruction Set

BRCLR *n*Branch if Bit *n* in Memory Clear**BRCLR *n*****Operation**If bit *n* of M = 0, PC ← (PC) + \$0003 + *rel***Description**

Tests bit *n* (*n* = 7, 6, 5, ... 0) of location M and branches if the bit is clear. M can be any RAM or I/O register address in the \$0000 to \$00FF area of memory because direct addressing mode is used to specify the address of the operand.

The C bit is set to the state of the tested bit. When used with an appropriate rotate instruction, BRCLR *n* provides an easy method for performing serial-to-parallel conversions.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
—	1	1	—	—	—	—	↑

C: Set if $M_n = 1$; cleared otherwise**Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details**

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BRCLR 0, <i>opr8a,rel</i>	DIR (b0)	01	dd rr	5
BRCLR 1, <i>opr8a,rel</i>	DIR (b1)	03	dd rr	5
BRCLR 2, <i>opr8a,rel</i>	DIR (b2)	05	dd rr	5
BRCLR 3, <i>opr8a,rel</i>	DIR (b3)	07	dd rr	5
BRCLR 4, <i>opr8a,rel</i>	DIR (b4)	09	dd rr	5
BRCLR 5, <i>opr8a,rel</i>	DIR (b5)	0B	dd rr	5
BRCLR 6, <i>opr8a,rel</i>	DIR (b6)	0D	dd rr	5
BRCLR 7, <i>opr8a,rel</i>	DIR (b7)	0F	dd rr	5

BRN**Branch Never****BRN****Operation**

$$PC \leftarrow (PC) + \$0002$$
Description

Never branches. In effect, this instruction can be considered a 2-byte no operation (NOP) requiring three cycles for execution. Its inclusion in the instruction set provides a complement for the **BRA** instruction. The BRN instruction is useful during program debugging to negate the effect of another branch instruction without disturbing the offset byte.

This instruction can be useful in instruction-based timing delays. Instruction-based timing delays are usually discouraged because such code is not portable to systems with different clock speeds.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BRN <i>rel</i>	REL	21	rr	3

See the **BRA** instruction for a summary of all branches and their complements.

Instruction Set

BRSET *n*Branch if Bit *n* in Memory Set**BRSET *n*****Operation**If bit *n* of M = 1, $PC \leftarrow (PC) + \$0003 + rel$ **Description**

Tests bit *n* (*n* = 7, 6, 5, ... 0) of location M and branches if the bit is set. M can be any RAM or I/O register address in the \$0000 to \$00FF area of memory because direct addressing mode is used to specify the address of the operand.

The C bit is set to the state of the tested bit. When used with an appropriate rotate instruction, BRSET *n* provides an easy method for performing serial-to-parallel conversions.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
—	1	1	—	—	—	—	↑

C: Set if $Mn = 1$; cleared otherwise**Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details**

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BRSET 0, <i>opr8a,rel</i>	DIR (b0)	00	dd rr	5
BRSET 1, <i>opr8a,rel</i>	DIR (b1)	02	dd rr	5
BRSET 2, <i>opr8a,rel</i>	DIR (b2)	04	dd rr	5
BRSET 3, <i>opr8a,rel</i>	DIR (b3)	06	dd rr	5
BRSET 4, <i>opr8a,rel</i>	DIR (b4)	08	dd rr	5
BRSET 5, <i>opr8a,rel</i>	DIR (b5)	0A	dd rr	5
BRSET 6, <i>opr8a,rel</i>	DIR (b6)	0C	dd rr	5
BRSET 7, <i>opr8a,rel</i>	DIR (b7)	0E	dd rr	5

BSET *n***Set Bit *n* in Memory****BSET *n*****Operation** $Mn \leftarrow 1$ **Description**

Set bit *n* (*n* = 7, 6, 5, ... 0) in location M. All other bits in M are unaffected. M can be any RAM or I/O register address in the \$0000 to \$00FF area of memory because direct addressing mode is used to specify the address of the operand. This instruction reads the specified 8-bit location, modifies the specified bit, and then writes the modified 8-bit value back to the memory location.

**Condition Codes
and Boolean
Formulae**

None affected

V	H		I	N	Z	C
—	1	1	—	—	—	—

**Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details**

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BSET 0, <i>opr8a</i>	DIR (b0)	10	dd	4
BSET 1, <i>opr8a</i>	DIR (b1)	12	dd	4
BSET 2, <i>opr8a</i>	DIR (b2)	14	dd	4
BSET 3, <i>opr8a</i>	DIR (b3)	16	dd	4
BSET 4, <i>opr8a</i>	DIR (b4)	18	dd	4
BSET 5, <i>opr8a</i>	DIR (b5)	1A	dd	4
BSET 6, <i>opr8a</i>	DIR (b6)	1C	dd	4
BSET 7, <i>opr8a</i>	DIR (b7)	1E	dd	4

BSR

Branch to Subroutine

BSR

Operation

 $PC \leftarrow (PC) + \$0002$

Advance PC to return address

 $Push(PCL); SP \leftarrow (SP) - \$0001$

Push low half of return address

 $Push(PCH); SP \leftarrow (SP) - \$0001$

Push high half of return address

 $PC \leftarrow (PC) + rel$

Load PC with start address of requested subroutine

Description

The program counter is incremented by 2 from the opcode address (so it points to the opcode of the next instruction which will be the return address). The least significant byte of the contents of the program counter (low-order return address) is pushed onto the stack. The stack pointer is then decremented by 1. The most significant byte of the contents of the program counter (high-order return address) is pushed onto the stack. The stack pointer is then decremented by 1. A branch then occurs to the location specified by the branch offset. See the [BRA](#) instruction for further details of the execution of the branch.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
BSR <i>rel</i>	REL	AD	rr	4

CBEQ

Compare and Branch if Equal

CBEQ

Operation

For DIR or IMM modes: if (A) = (M), $PC \leftarrow (PC) + \$0003 + rel$
Or for IX+ mode: if (A) = (M); $PC \leftarrow (PC) + \$0002 + rel$
Or for SP1 mode: if (A) = (M); $PC \leftarrow (PC) + \$0004 + rel$
Or for CBEQX: if (X) = (M); $PC \leftarrow (PC) + \$0003 + rel$

Description

CBEQ compares the operand with the accumulator (or index register for CBEQX instruction) against the contents of a memory location and causes a branch if the register (A or X) is equal to the memory contents. The CBEQ instruction combines CMP and BEQ for faster table lookup routines and condition codes are not changed.

The IX+ variation of the CBEQ instruction compares the operand addressed by H:X to A and causes a branch if the operands are equal. H:X is then incremented regardless of whether a branch is taken. The IX1+ variation of CBEQ operates the same way except that an 8-bit offset is added to H:X to form the effective address of the operand.

Condition Codes
and Boolean
Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code			HC08 Cycles
		Opcode	Operand(s)		
CBEQ <i>opr8a,rel</i>	DIR	31	dd	rr	5
CBEQA <i>#opr8i,rel</i>	IMM	41	ii	rr	4
CBEQX <i>#opr8i,rel</i>	IMM	51	ii	rr	4
CBEQ <i>opr8,X+,rel</i>	IX1+	61	ff	rr	5
CBEQ <i>,X+,rel</i>	IX+	71	rr		4
CBEQ <i>opr8,SP,rel</i>	SP1	9E61	ff	rr	6

Instruction Set

CLC

Clear Carry Bit

CLC

Operation

C bit $\leftarrow$ 0

Description

Clears the C bit in the CCR. CLC may be used to set up the C bit prior to a shift or rotate instruction that involves the C bit. The C bit can also be used to pass status information between a subroutine and the calling program.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
—	1	1	—	—	—	—	0

C: 0
Cleared

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
CLC	INH	98		1

CLI**Clear Interrupt Mask Bit****CLI****Operation**I bit $\leftarrow$ 0**Description**

Clears the interrupt mask bit in the CCR. When the I bit is clear, interrupts are enabled. The I bit actually changes to zero at the end of the cycle where the CLI instruction executes. This is too late to recognize an interrupt that arrived before or during the CLI instruction so if interrupts were previously disabled, the next instruction after a CLI will always be executed even if there was an interrupt pending prior to execution of the CLI instruction.

**Condition Codes
and Boolean
Formulae**

V		H		I	N	Z	C
—	1	1	—	0	—	—	—

I: 0
Cleared

**Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail**

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
CLI	INH	9A		2

Instruction Set

CLR

Clear

CLR

Operation

$A \leftarrow \$00$
Or $M \leftarrow \$00$
Or $X \leftarrow \$00$
Or $H \leftarrow \$00$

Description

The contents of memory (M), A, X, or H are replaced with zeros.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
0	1	1	—	—	0	1	—

V: 0
Cleared

N: 0
Cleared

Z: 1
Set

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
CLR <i>opr8a</i>	DIR	3F	dd	3
CLRA	INH (A)	4F		1
CLR X	INH (X)	5F		1
CLR H	INH (H)	8C		1
CLR <i>opr8,X</i>	IX1	6F	ff	3
CLR <i>,X</i>	IX	7F		2
CLR <i>opr8,SP</i>	SP1	9E6F	ff	4

CMP

Compare Accumulator with Memory

CMP

Operation

(A) – (M)

Description

Compares the contents of A to the contents of M and sets the condition codes, which may then be used for arithmetic (signed or unsigned) and logical conditional branching. The contents of both A and M are unchanged.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
↑	1	1	—	—	↑	↑	↑

V: $\overline{A7 \& M7 \& R7} \mid \overline{A7 \& M7 \& R7}$

Set if a two's complement overflow resulted from the operation; cleared otherwise. Literally read, an overflow condition occurs if a positive number is subtracted from a negative number with a positive result, or, if a negative number is subtracted from a positive number with a negative result.

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7 \& R6 \& R5 \& R4 \& R3 \& R2 \& R1 \& R0}$

Set if result is \$00; cleared otherwise

C: $\overline{A7 \& M7} \mid M7 \& R7 \mid R7 \& \overline{A7}$

Set if the unsigned value of the contents of memory is larger than the unsigned value of the accumulator; cleared otherwise

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
CMP <i>#opr8i</i>	IMM	A1	ii	2
CMP <i>opr8a</i>	DIR	B1	dd	3
CMP <i>opr16a</i>	EXT	C1	hh ll	4
CMP <i>opr16,X</i>	IX2	D1	ee ff	4
CMP <i>opr8,X</i>	IX1	E1	ff	3
CMP <i>,X</i>	IX	F1		2
CMP <i>opr16,SP</i>	SP2	9ED1	ee ff	5
CMP <i>opr8,SP</i>	SP1	9EE1	ff	4

Instruction Set

COM

Complement (One's Complement)

COM

Operation

$A \leftarrow \bar{A} = \$FF - (A)$
Or $X \leftarrow \bar{X} = \$FF - (X)$
Or $M \leftarrow \bar{M} = \$FF - (M)$

Description

Replaces the contents of A, X, or M with the one's complement. Each bit of A, X, or M is replaced with the complement of that bit.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
0	1	1	—	—	↑	↑	1

V: 0

Cleared

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\bar{R7} \& \bar{R6} \& \bar{R5} \& \bar{R4} \& \bar{R3} \& \bar{R2} \& \bar{R1} \& \bar{R0}$

Set if result is \$00; cleared otherwise

C: 1

Set

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
COM <i>opr8a</i>	DIR	33	dd	4
COMA	INH (A)	43		1
COMX	INH (X)	53		1
COM <i>opr8,X</i>	IX1	63	ff	4
COM <i>,X</i>	IX	73		3
COM <i>opr8,SP</i>	SP1	9E63	ff	5

CPHX

Compare Index Register with Memory

CPHX

Operation

 $(H:X) - (M:M + \$0001)$

Description

CPHX compares index register (H:X) with the 16-bit value in memory and sets the condition codes, which may then be used for arithmetic (signed or unsigned) and logical conditional branching. The contents of both H:X and M:M + \$0001 are unchanged.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
↑	1	1	—	—	↑	↑	↑

V: $\overline{H7 \& M15 \& R15} \mid \overline{H7 \& M15 \& R15}$

Set if a two's complement overflow resulted from the operation; cleared otherwise

N: R15

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R15 \& R14 \& R13 \& R12 \& R11 \& R10 \& R9 \& R8}$
 $\& R7 \& R6 \& R5 \& R4 \& R3 \& R2 \& R1 \& R0$

Set if the result is \$0000; cleared otherwise

C: $\overline{H7 \& M15} \mid M15 \& R15 \mid R15 \& \overline{H7}$

Set if the absolute value of the contents of memory is larger than the absolute value of the index register; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
CPHX <i>#opr</i>	IMM	65	jj kk+1	3
CPHX <i>opr</i>	DIR	75	dd	4

Instruction Set

CPX

Compare X (Index Register Low) with Memory

CPX

Operation

(X) – (M)

Description

Compares the contents of X to the contents of M and sets the condition codes, which may then be used for arithmetic (signed or unsigned) and logical conditional branching. The contents of both X and M are unchanged.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
↑	1	1	—	—	↑	↑	↑

V: $\overline{X7 \& M7 \& R7} \mid \overline{X7 \& M7 \& R7}$

Set if a two's complement overflow resulted from the operation; cleared otherwise

N: R7

Set if MSB of result of the subtraction is 1; cleared otherwise

Z: $\overline{R7 \& R6 \& R5 \& R4 \& R3 \& R2 \& R1 \& R0}$

Set if result is \$00; cleared otherwise

C: $\overline{X7 \& M7} \mid M7 \& R7 \mid R7 \& \overline{X7}$

Set if the unsigned value of the contents of memory is larger than the unsigned value in the index register; cleared otherwise

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
CPX <i>#opr8i</i>	IMM	A3	ii	2
CPX <i>opr8a</i>	DIR	B3	dd	3
CPX <i>opr16a</i>	EXT	C3	hh ll	4
CPX <i>opr16,X</i>	IX2	D3	ee ff	4
CPX <i>opr8,X</i>	IX1	E3	ff	3
CPX <i>,X</i>	IX	F3		2
CPX <i>opr16,SP</i>	SP2	9ED3	ee ff	5
CPX <i>opr8,SP</i>	SP1	9EE3	ff	4

DAA

Decimal Adjust Accumulator

DAA

Operation $(A)_{10}$ **Description**

Adjusts the contents of the accumulator and the state of the CCR carry bit after an ADD or ADC operation involving binary-coded decimal (BCD) values, so that there is a correct BCD sum and an accurate carry indication. The state of the CCR half carry bit affects operation. Refer to [Table 5-2](#) for details of operation.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
U	1	1	—	—	↑	↑	↑

V: U

Undefined

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

C: Set if the decimal adjusted result is greater than 99 (decimal); refer to [Table 5-2](#)

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
DAA	INH	72		2

The DAA description continues next page.

DAA

Decimal Adjust Accumulator (Continued)

DAA

Table 5-2 shows DAA operation for all legal combinations of input operands. Columns 1–4 represent the results of ADC or ADD operations on BCD operands. The correction factor in column 5 is added to the accumulator to restore the result of an operation on two BCD operands to a valid BCD value and to set or clear the C bit. All values in this table are hexadecimal.

Table 5-2. DAA Function Summary

1	2	3	4	5	6
Initial C-Bit Value	Value of A[7:4]	Initial H-Bit Value	Value of A[3:0]	Correction Factor	Corrected C-Bit Value
0	0–9	0	0–9	00	0
0	0–8	0	A–F	06	0
0	0–9	1	0–3	06	0
0	A–F	0	0–9	60	1
0	9–F	0	A–F	66	1
0	A–F	1	0–3	66	1
1	0–2	0	0–9	60	1
1	0–2	0	A–F	66	1
1	0–3	1	0–3	66	1

DBNZ

Decrement and Branch if Not Zero

DBNZ

Operation

$$A \leftarrow (A) - \$01$$

Or $M \leftarrow (M) - \$01$

Or $X \leftarrow (X) - \$01$

For DIR or IX1 modes: $PC \leftarrow (PC) + \$0003 + rel$ if (result) $\neq 0$

Or for INH or IX modes: $PC \leftarrow (PC) + \$0002 + rel$ if (result) $\neq 0$

Or for SP1 mode: $PC \leftarrow (PC) + \$0004 + rel$ if (result) $\neq 0$

Description

Subtract 1 from the contents of A, M, or X; then branch using the relative offset if the result of the subtraction is not \$00. DBNZX only affects the low order eight bits of the H:X index register pair; the high-order byte (H) is not affected.

Condition Codes
and Boolean
Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
DBNZ <i>opr8a,rel</i>	DIR	3B	dd rr	5
DBNZ <i>A,rel</i>	INH	4B	rr	3
DBNZX <i>rel</i>	INH	5B	rr	3
DBNZ <i>opr8,X,rel</i>	IX1	6B	ff rr	5
DBNZ <i>,X,rel</i>	IX	7B	rr	4
DBNZ <i>opr8,SP,rel</i>	SP1	9E6B	ff rr	6

DEC

Decrement

DEC

Operation

$A \leftarrow (A) - \$01$
Or $X \leftarrow (X) - \$01$
Or $M \leftarrow (M) - \$01$

Description

Subtract 1 from the contents of A, X, or M. The V, N, and Z bits in the CCR are set or cleared according to the results of this operation. The C bit in the CCR is not affected; therefore, the BLS, BLO, BHS, and BHI branch instructions are not useful following a DEC instruction.

DECX only affects the low-order byte of index register pair (H:X). To decrement the full 16-bit index register pair (H:X), use AIX # -1.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
↑	1	1	—	—	↑	↑	—

V: $\overline{R7} \& A7$

Set if there was a two's complement overflow as a result of the operation; cleared otherwise. Two's complement overflow occurs if and only if (A), (X), or (M) was \$80 before the operation.

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
DEC <i>opr8a</i>	DIR	3A	dd	4
DECA	INH (A)	4A		1
DECX	INH (X)	5A		1
DEC <i>opr8,X</i>	IX1	6A	ff	4
DEC <i>,X</i>	IX	7A		3
DEC <i>opr8,SP</i>	SP1	9E6A	ff	5

DEX is recognized by assemblers as being equivalent to DECX.

DIV

Divide

DIV

Operation

$$A \leftarrow (H:A) \div (X); H \leftarrow \text{Remainder}$$

Description

Divides a 16-bit unsigned dividend contained in the concatenated registers H and A by an 8-bit divisor contained in X. The quotient is placed in A, and the remainder is placed in H. The divisor is left unchanged.

An overflow (quotient > \$FF) or divide-by-0 sets the C bit, and the quotient and remainder are indeterminate.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
—	1	1	—	—	—	↑	↑

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result (quotient) is \$00; cleared otherwise

C: Set if a divide-by-0 was attempted or if an overflow occurred; cleared otherwise

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
DIV	INH	52		7

Instruction Set

EOR

Exclusive-OR Memory with Accumulator

EOR

Operation

$$A \leftarrow (A \oplus M)$$

Description

Performs the logical exclusive-OR between the contents of A and the contents of M and places the result in A. Each bit of A after the operation will be the logical exclusive-OR of the corresponding bits of M and A before the operation.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
0	1	1	—	—	↑	↑	—

V: 0

Cleared

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
EOR <i>#opr8i</i>	IMM	A8	ii	2
EOR <i>opr8a</i>	DIR	B8	dd	3
EOR <i>opr16a</i>	EXT	C8	hh ll	4
EOR <i>opr16,X</i>	IX2	D8	ee ff	4
EOR <i>opr8,X</i>	IX1	E8	ff	3
EOR <i>,X</i>	IX	F8		2
EOR <i>opr16,SP</i>	SP2	9ED8	ee ff	5
EOR <i>opr8,SP</i>	SP1	9EE8	ff	4

INC

Increment

INC

Operation

 $A \leftarrow (A) + \$01$ **Or** $X \leftarrow (X) + \$01$ **Or** $M \leftarrow (M) + \$01$

Description

Add 1 to the contents of A, X, or M. The V, N, and Z bits in the CCR are set or cleared according to the results of this operation. The C bit in the CCR is not affected; therefore, the BLS, BLO, BHS, and BHI branch instructions are not useful following an INC instruction.

INCX only affects the low-order byte of index register pair (H:X). To increment the full 16-bit index register pair (H:X), use AIX #1.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
↑	1	1	—	—	↑	↑	—

V: $\overline{A7 \& R7}$

Set if there was a two's complement overflow as a result of the operation; cleared otherwise. Two's complement overflow occurs if and only if (A), (X), or (M) was \$7F before the operation.

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7 \& R6 \& R5 \& R4 \& R3 \& R2 \& R1 \& R0}$

Set if result is \$00; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
INC <i>opr8a</i>	DIR	3C	dd	4
INCA	INH (A)	4C		1
INCX	INH (X)	5C		1
INC <i>opr8,X</i>	IX1	6C	ff	4
INC <i>,X</i>	IX	7C		3
INC <i>opr8,SP</i>	SP1	9E6C	ff	5

INX is recognized by assemblers as being equivalent to INCX.

Instruction Set

JMP

Jump

JMP

Operation

PC ← effective address

Description

A jump occurs to the instruction stored at the effective address. The effective address is obtained according to the rules for extended, direct, or indexed addressing.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
JMP <i>opr8a</i>	DIR	BC	dd	2
JMP <i>opr16a</i>	EXT	CC	hh ll	3
JMP <i>opr16,X</i>	IX2	DC	ee ff	4
JMP <i>opr8,X</i>	IX1	EC	ff	3
JMP <i>,X</i>	IX	FC		3

JSR

Jump to Subroutine

JSR

Operation

$PC \leftarrow (PC) + n;$
 $n = 1, 2, \text{ or } 3$ depending on address mode
 Push (PCL); $SP \leftarrow (SP) - \$0001$ Push low half of return address
 Push (PCH); $SP \leftarrow (SP) - \$0001$ Push high half of return address
 $PC \leftarrow$ effective address Load PC with start address of requested subroutine

Description

The program counter is incremented by n so that it points to the opcode of the next instruction that follows the JSR instruction ($n = 1, 2, \text{ or } 3$ depending on the addressing mode). The PC is then pushed onto the stack, eight bits at a time, least significant byte first. The stack pointer points to the next empty location on the stack. A jump occurs to the instruction stored at the effective address. The effective address is obtained according to the rules for extended, direct, or indexed addressing.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
JSR <i>opr8a</i>	DIR	BD	dd	4
JSR <i>opr16a</i>	EXT	CD	hh ll	5
JSR <i>opr16,X</i>	IX2	DD	ee ff	6
JSR <i>opr8,X</i>	IX1	ED	ff	5
JSR <i>,X</i>	IX	FD		4

Instruction Set

LDA

Load Accumulator from Memory

LDA

Operation

 $A \leftarrow (M)$

Description

Loads the contents of the specified memory location into A. The N and Z condition codes are set or cleared according to the loaded data; V is cleared. This allows conditional branching after the load without having to perform a separate test or compare.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
0	1	1	—	—	↑	↑	—

V: 0
Cleared

N: R7
Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$
Set if result is \$00; cleared otherwise

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
LDA <i>#opr8i</i>	IMM	A6	ii	2
LDA <i>opr8a</i>	DIR	B6	dd	3
LDA <i>opr16a</i>	EXT	C6	hh ll	4
LDA <i>opr16,X</i>	IX2	D6	ee ff	4
LDA <i>opr8,X</i>	IX1	E6	ff	3
LDA <i>,X</i>	IX	F6		2
LDA <i>opr16,SP</i>	SP2	9ED6	ee ff	5
LDA <i>opr8,SP</i>	SP1	9EE6	ff	4

LDHX

Load Index Register from Memory

LDHX

Operation

$$H:X \leftarrow (M:M + \$0001)$$

Description

Loads the contents of the specified memory location into the index register (H:X). The N and Z condition codes are set according to the data; V is cleared. This allows conditional branching after the load without having to perform a separate test or compare.

Condition Codes
and Boolean
Formulae

V		H		I	N	Z	C
0	1	1	—	—	↑	↑	—

V: 0

Cleared

N: R15

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R15} \& \overline{R14} \& \overline{R13} \& \overline{R12} \& \overline{R11} \& \overline{R10} \& \overline{R9} \& \overline{R8}$
 $\& \overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if the result is \$0000; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
LDHX <i>#opr</i>	IMM	45	jj kk	3
LDHX <i>opr</i>	DIR	55	dd	4

Instruction Set

LDX

Load X (Index Register Low) from Memory

LDX

Operation

$$X \leftarrow (M)$$

Description

Loads the contents of the specified memory location into X. The N and Z condition codes are set or cleared according to the loaded data; V is cleared. This allows conditional branching after the load without having to perform a separate test or compare.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
0	1	1	—	—	↑	↑	—

V: 0

Cleared

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

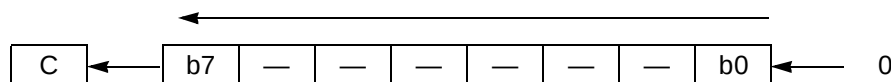
Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
LDX <i>#opr8i</i>	IMM	AE	ii	2
LDX <i>opr8a</i>	DIR	BE	dd	3
LDX <i>opr16a</i>	EXT	CE	hh ll	4
LDX <i>opr16,X</i>	IX2	DE	ee ff	4
LDX <i>opr8,X</i>	IX1	EE	ff	3
LDX <i>,X</i>	IX	FE		2
LDX <i>opr16,SP</i>	SP2	9EDE	ee ff	5
LDX <i>opr8,SP</i>	SP1	9EEE	ff	4

LSL

Logical Shift Left
(Same as ASL)

LSL

Operation



Description

Shifts all bits of the A, X, or M one place to the left. Bit 0 is loaded with a 0. The C bit in the CCR is loaded from the most significant bit of A, X, or M.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
↑	1	1	—	—	↑	↑	↑

V: $R7 \oplus b7$

Set if the exclusive-OR of the resulting N and C flags is 1; cleared otherwise

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

C: b7

Set if, before the shift, the MSB of A, X, or M was set; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
LSL <i>opr8a</i>	DIR	38	dd	4
LSLA	INH (A)	48		1
LSLX	INH (X)	58		1
LSL <i>opr8,X</i>	IX1	68	ff	4
LSL <i>,X</i>	IX	78		3
LSL <i>opr8,SP</i>	SP1	9E68	ff	5

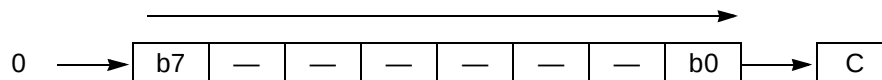
Instruction Set

LSR

Logical Shift Right

LSR

Operation



Description

Shifts all bits of A, X, or M one place to the right. Bit 7 is loaded with a 0. Bit 0 is shifted into the C bit.

Condition Codes
and Boolean
Formulae

V	H		I	N	Z	C
↑	1	1	—	—	0	↑

V: $0 \oplus b0 = b0$

Set if the exclusive-OR of the resulting N and C flags is 1; cleared otherwise. Since N = 0, this simplifies to the value of bit 0 before the shift.

N: 0

Cleared

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

C: b0

Set if, before the shift, the LSB of A, X, or M, was set; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
LSR <i>opr8a</i>	DIR	34	dd	4
LSRA	INH (A)	44		1
LSRX	INH (X)	54		1
LSR <i>opr8,X</i>	IX1	64	ff	4
LSR <i>,X</i>	IX	74		3
LSR <i>opr8,SP</i>	SP1	9E64	ff	5

MOV

Move

MOV

Operation

 $(M)_{\text{Destination}} \leftarrow (M)_{\text{Source}}$

Description

Moves a byte of data from a source address to a destination address. Data is examined as it is moved, and condition codes are set. Source data is not changed. The accumulator is not affected.

The four addressing modes for the MOV instruction are:

1. IMM/DIR moves an immediate byte to a direct memory location.
2. DIR/DIR moves a direct location byte to another direct location.
3. IX+/DIR moves a byte from a location addressed by H:X to a direct location. H:X is incremented after the move.
4. DIR/IX+ moves a byte from a direct location to one addressed by H:X. H:X is incremented after the move.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
0	1	1	—	—	↑	↓	—

V: 0

Cleared

N: R7

Set if MSB of result is set; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
MOV <i>opr8a,opr8a</i>	DIR/DIR	4E	dd dd	5
MOV <i>opr8a,X+</i>	DIR/IX+	5E	dd	4
MOV <i>#opr8i,opr8a</i>	IMM/DIR	6E	ii dd	4
MOV <i>,X+,opr8a</i>	IX+/DIR	7E	dd	4

Instruction Set

MUL

Unsigned Multiply

MUL

Operation

$$X:A \leftarrow (X) \times (A)$$

Description

Multiplies the 8-bit value in X (index register low) by the 8-bit value in the accumulator to obtain a 16-bit unsigned result in the concatenated index register and accumulator. After the operation, X contains the upper eight bits of the 16-bit result and A contains the lower eight bits of the result.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
—	1	1	0	—	—	—	0

H: 0
Cleared

C: 0
Cleared

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
MUL	INH	42		5

NEG

Negate (Two's Complement)

NEG

Operation

 $A \leftarrow - (A)$ **Or** $X \leftarrow - (X)$ **Or** $M \leftarrow - (M);$

this is equivalent to subtracting A, X, or M from \$00

Description

Replaces the contents of A, X, or M with its two's complement. Note that the value \$80 is left unchanged.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
↑	1	1	—	—	↑	↑	↑

V: M7&R7

Set if a two's complement overflow resulted from the operation; cleared otherwise. Overflow will occur only if the operand is \$80 before the operation.

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

C: $R7|R6|R5|R4|R3|R2|R1|R0$

Set if there is a borrow in the implied subtraction from 0; cleared otherwise. The C bit will be set in all cases except when the contents of A, X, or M was \$00 prior to the NEG operation.

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
NEG <i>opr8a</i>	DIR	30	dd	4
NEGA	INH (A)	40		1
NEGX	INH (X)	50		1
NEG <i>opr8,X</i>	IX1	60	ff	4
NEG <i>,X</i>	IX	70		3
NEG <i>opr8,SP</i>	SP1	9E60	ff	5

Instruction Set

NOP

No Operation

NOP

Operation

Uses one bus cycle

Description

This is a single-byte instruction that does nothing except to consume one CPU clock cycle while the program counter is advanced to the next instruction. No register or memory contents are affected by this instruction.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
NOP	INH	9D		1

NSA

Nibble Swap Accumulator

NSA

Operation

$$A \leftarrow (A[3:0]:A[7:4])$$

Description

Swaps upper and lower nibbles (4 bits) of the accumulator. The NSA instruction is used for more efficient storage and use of binary-coded decimal operands.

Condition Codes
and Boolean
Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
NSA	INH	62		3

Instruction Set

ORA

Inclusive-OR Accumulator and Memory

ORA

Operation

$$A \leftarrow (A) \mid (M)$$

Description

Performs the logical inclusive-OR between the contents of A and the contents of M and places the result in A. Each bit of A after the operation will be the logical inclusive-OR of the corresponding bits of M and A before the operation.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
0	1	1	—	—	↑	↑	—

V: 0

Cleared

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
ORA <i>#opr8i</i>	IMM	AA	ii	2
ORA <i>opr8a</i>	DIR	BA	dd	3
ORA <i>opr16a</i>	EXT	CA	hh ll	4
ORA <i>opr16,X</i>	IX2	DA	ee ff	4
ORA <i>opr8,X</i>	IX1	EA	ff	3
ORA <i>,X</i>	IX	FA		2
ORA <i>opr16,SP</i>	SP2	9EDA	ee ff	5
ORA <i>opr8,SP</i>	SP1	9EEA	ff	4

PSHA

Push Accumulator onto Stack

PSHA

OperationPush (A); $SP \leftarrow (SP) - \$0001$ **Description**

The contents of A are pushed onto the stack at the address contained in the stack pointer. The stack pointer is then decremented to point to the next available location in the stack. The contents of A remain unchanged.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
PSHA	INH	87		2

Instruction Set

PSHH

Push H (Index Register High) onto Stack

PSHH

Operation

Push (H); $SP \leftarrow (SP) - \$0001$

Description

The contents of H are pushed onto the stack at the address contained in the stack pointer. The stack pointer is then decremented to point to the next available location in the stack. The contents of H remain unchanged.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
PSHH	INH	8B		2

PSHX

Push X (Index Register Low) onto Stack

PSHX

OperationPush (X); $SP \leftarrow (SP) - \$0001$ **Description**

The contents of X are pushed onto the stack at the address contained in the stack pointer (SP). SP is then decremented to point to the next available location in the stack. The contents of X remain unchanged.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
PSHX	INH	89		2

PULA

Pull Accumulator from Stack

PULA

Operation

$$SP \leftarrow (SP + \$0001); \text{pull (A)}$$
Description

The stack pointer (SP) is incremented to address the last operand on the stack. The accumulator is then loaded with the contents of the address pointed to by SP.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
PULA	INH	86		2

PULH

Pull H (Index Register High) from Stack

PULH

Operation

$$SP \leftarrow (SP + \$0001); \text{pull (H)}$$
Description

The stack pointer (SP) is incremented to address the last operand on the stack. H is then loaded with the contents of the address pointed to by SP.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
PULH	INH	8A		2

PULX

Pull X (Index Register Low) from Stack

PULX

Operation $SP \leftarrow (SP + \$0001); \text{pull } (X)$ **Description**

The stack pointer (SP) is incremented to address the last operand on the stack. X is then loaded with the contents of the address pointed to by SP.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

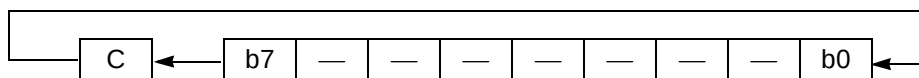
Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
PULX	INH	88		2

ROL

Rotate Left through Carry

ROL

Operation



Description

Shifts all bits of A, X, or M one place to the left. Bit 0 is loaded from the C bit. The C bit is loaded from the most significant bit of A, X, or M. The rotate instructions include the carry bit to allow extension of the shift and rotate instructions to multiple bytes. For example, to shift a 24-bit value left one bit, the sequence (ASL LOW, ROL MID, ROL HIGH) could be used, where LOW, MID, and HIGH refer to the low-order, middle, and high-order bytes of the 24-bit value, respectively.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
↑	1	1	—	—	↑	↑	↑

V: $R7 \oplus b7$

Set if the exclusive-OR of the resulting N and C flags is 1; cleared otherwise

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if result is \$00; cleared otherwise

C: b7

Set if, before the rotate, the MSB of A, X, or M was set; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
ROL <i>opr8a</i>	DIR	39	dd	4
ROLA	INH (A)	49		1
ROLX	INH (X)	59		1
ROL <i>opr8,X</i>	IX1	69	ff	4
ROL <i>,X</i>	IX	79		3
ROL <i>opr8,SP</i>	SP1	9E69	ff	5

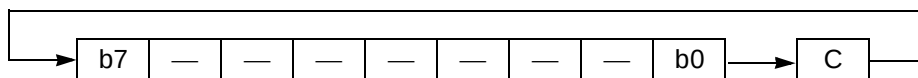
Instruction Set

ROR

Rotate Right through Carry

ROR

Operation



Description

Shifts all bits of A, X, or M one place to the right. Bit 7 is loaded from the C bit. Bit 0 is shifted into the C bit. The rotate instructions include the carry bit to allow extension of the shift and rotate instructions to multiple bytes. For example, to shift a 24-bit value right one bit, the sequence (LSR HIGH, ROR MID, ROR LOW) could be used, where LOW, MID, and HIGH refer to the low-order, middle, and high-order bytes of the 24-bit value, respectively.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
↑	1	1	—	—	↑	↑	↑

V: $R7 \oplus b0$

Set if the exclusive-OR of the resulting N and C flags is 1; cleared otherwise

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7 \& R6 \& R5 \& R4 \& R3 \& R2 \& R1 \& R0}$

Set if result is \$00; cleared otherwise

C: b0

Set if, before the shift, the LSB of A, X, or M was set; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
ROR <i>opr8a</i>	DIR	36	dd	4
RORA	INH (A)	46		1
RORX	INH (X)	56		1
ROR <i>opr8,X</i>	IX1	66	ff	4
ROR <i>,X</i>	IX	76		3
ROR <i>opr8,SP</i>	SP1	9E66	ff	5

RSP

Reset Stack Pointer

RSP

Operation

 $SP \leftarrow \$FF$

Description

In most M68HC05 MCUs, RAM only goes to \$00FF. In most HC08s, however, RAM extends beyond \$00FF. Therefore, do not locate the stack in direct address space which is more valuable for commonly accessed variables. In new HC08 programs, it is more appropriate to initialize the stack pointer to the address of the last location (highest address) in the on-chip RAM, shortly after reset. This code segment demonstrates a typical method for initializing SP.

```
LDHX  #ram_end+1    ; Point at next addr past RAM
TXS                      ; SP <- (H:X)-1
```

Condition Codes
and Boolean
Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
RSP	INH	9C		1

Instruction Set

RTI

Return from Interrupt

RTI

Operation

$SP \leftarrow SP + \$0001$; pull (CCR) Restore CCR from stack
 $SP \leftarrow SP + \$0001$; pull (A) Restore A from stack
 $SP \leftarrow SP + \$0001$; pull (X) Restore X from stack
 $SP \leftarrow SP + \$0001$; pull (PCH) Restore PCH from stack
 $SP \leftarrow SP + \$0001$; pull (PCL) Restore PCL from stack

Description

The condition codes, the accumulator, X (index register low), and the program counter are restored to the state previously saved on the stack. The I bit will be cleared if the corresponding bit stored on the stack is 0, the normal case.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
↓	1	1	↓	↓	↓	↓	↓

Set or cleared according to the byte pulled from the stack into CCR.

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
RTI	INH	80		7

RTS

Return from Subroutine

RTS

Operation

$$SP \leftarrow SP + \$0001; \text{pull (PCH) Restore PCH from stack}$$

$$SP \leftarrow SP + \$0001; \text{pull (PCL) Restore PCL from stack}$$

Description

The stack pointer is incremented by 1. The contents of the byte of memory that is pointed to by the stack pointer are loaded into the high-order byte of the program counter. The stack pointer is again incremented by 1. The contents of the byte of memory that are pointed to by the stack pointer are loaded into the low-order eight bits of the program counter. Program execution resumes at the address that was just restored from the stack.

Condition Codes
and Boolean
Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
RTS	INH	81		4

SBC

Subtract with Carry

SBC

Operation

$$A \leftarrow (A) - (M) - (C)$$

Description

Subtracts the contents of M and the contents of the C bit of the CCR from the contents of A and places the result in A. This is useful for multi-precision subtract algorithms involving operands with more than eight bits.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
↑	1	1	—	—	↑	↑	↑

$$V: A7\&\overline{M7}\&\overline{R7} \mid \overline{A7}\&M7\&R7$$

Set if a two's complement overflow resulted from the operation; cleared otherwise. Literally read, an overflow condition occurs if a positive number is subtracted from a negative number with a positive result, or, if a negative number is subtracted from a positive number with a negative result.

$$N: R7$$

Set if MSB of result is 1; cleared otherwise

$$Z: \overline{R7}\&\overline{R6}\&\overline{R5}\&\overline{R4}\&\overline{R3}\&\overline{R2}\&\overline{R1}\&\overline{R0}$$

Set if result is \$00; cleared otherwise

$$C: \overline{A7}\&M7 \mid M7\&R7 \mid R7\&\overline{A7}$$

Set if the unsigned value of the contents of memory plus the previous carry are larger than the unsigned value of the accumulator; cleared otherwise

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
SBC <i>#opr8i</i>	IMM	A2	ii	2
SBC <i>opr8a</i>	DIR	B2	dd	3
SBC <i>opr16a</i>	EXT	C2	hh ll	4
SBC <i>opr16,X</i>	IX2	D2	ee ff	4
SBC <i>opr8,X</i>	IX1	E2	ff	3
SBC <i>,X</i>	IX	F2		2
SBC <i>opr16,SP</i>	SP2	9ED2	ee ff	5
SBC <i>opr8,SP</i>	SP1	9EE2	ff	4

SEC

Set Carry Bit

SEC

Operation

C bit $\leftarrow$ 1

Description

Sets the C bit in the condition code register (CCR). SEC may be used to set up the C bit prior to a shift or rotate instruction that involves the C bit.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
—	1	1	—	—	—	—	1

C: 1
Set

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
SEC	INH	99		1

Instruction Set

SEI

Set Interrupt Mask Bit

SEI

Operation

I bit $\leftarrow$ 1

Description

Sets the interrupt mask bit in the condition code register (CCR). The microprocessor is inhibited from responding to interrupts while the I bit is set. The I bit actually changes at the end of the cycle where SEI executed. This is too late to stop an interrupt that arrived during execution of the SEI instruction so it is possible that an interrupt request could be serviced after the SEI instruction before the next instruction after SEI is executed. The global I-bit interrupt mask takes effect before the next instruction can be completed.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
—	1	1	—	1	—	—	—

I: 1
Set

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
SEI	INH	9B		2

STA

Store Accumulator in Memory

STA

Operation

 $M \leftarrow (A)$

Description

Stores the contents of A in memory. The contents of A remain unchanged. The N condition code is set if the most significant bit of A is set, the Z bit is set if A was \$00, and V is cleared. This allows conditional branching after the store without having to do a separate test or compare.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
0	1	1	—	—	↑	↑	—

V: 0

Cleared

N: A7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{A7 \& A6 \& A5 \& A4 \& A3 \& A2 \& A1 \& A0}$

Set if result is \$00; cleared otherwise

Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
STA <i>opr8a</i>	DIR	B7	dd	3
STA <i>opr16a</i>	EXT	C7	hh ll	4
STA <i>opr16,X</i>	IX2	D7	ee ff	4
STA <i>opr8,X</i>	IX1	E7	ff	3
STA <i>,X</i>	IX	F7		2
STA <i>opr16,SP</i>	SP2	9ED7	ee ff	5
STA <i>opr8,SP</i>	SP1	9EE7	ff	4

Instruction Set

STHX

Store Index Register

STHX

Operation

$$(M:M + \$0001) \leftarrow (H:X)$$

Description

Stores the contents of H in memory location M and then the contents of X into the next memory location (M + \$0001). The N condition code bit is set if the most significant bit of H was set, the Z bit is set if the value of H:X was \$0000, and V is cleared. This allows conditional branching after the store without having to do a separate test or compare.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
0	1	1	—	—	↑	↑	—

V: 0

Cleared

N: R15

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R15} \& \overline{R14} \& \overline{R13} \& \overline{R12} \& \overline{R11} \& \overline{R10} \& \overline{R9} \& \overline{R8}$
 $\& \overline{R7} \& \overline{R6} \& \overline{R5} \& \overline{R4} \& \overline{R3} \& \overline{R2} \& \overline{R1} \& \overline{R0}$

Set if the result is \$0000; cleared otherwise

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
STHX <i>opr</i>	DIR	35	dd	4

STOP

Enable $\overline{\text{IRQ}}$ Pin, Stop Oscillator

STOP

Operation

I bit $\leftarrow$ 0; stop oscillator

Description

Reduces power consumption by eliminating all dynamic power dissipation. (See module documentation for module reactions to STOP instruction.) The external interrupt pin is enabled and the I bit in the condition code register (CCR) is cleared to enable the external interrupt. Finally, the oscillator is inhibited to put the MCU into the STOP condition.

When either the $\overline{\text{RESET}}$ pin or $\overline{\text{IRQ}}$ pin goes low, the oscillator is enabled. A delay of 4095 processor clock cycles is imposed allowing the oscillator to stabilize. The reset vector or interrupt request vector is fetched and the associated service routine is executed.

External interrupts are enabled after a STOP command.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
—	1	1	—	0	—	—	—

I: 0
Cleared

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
STOP	INH	8E		1

Instruction Set

STX

Store X (Index Register Low) in Memory

STX

Operation

$$M \leftarrow (X)$$

Description

Stores the contents of X in memory. The contents of X remain unchanged. The N condition code is set if the most significant bit of X was set, the Z bit is set if X was \$00, and V is cleared. This allows conditional branching after the store without having to do a separate test or compare.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
0	1	1	—	—	↑	↑	—

V: 0

Cleared

N: X7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{X7 \& X6 \& X5 \& X4 \& X3 \& X2 \& X1 \& X0}$

Set if X is \$00; cleared otherwise

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
STX <i>opr8a</i>	DIR	BF	dd	3
STX <i>opr16a</i>	EXT	CF	hh ll	4
STX <i>opr16,X</i>	IX2	DF	ee ff	4
STX <i>opr8,X</i>	IX1	EF	ff	3
STX <i>,X</i>	IX	FF		2
STX <i>opr16,SP</i>	SP2	9EDF	ee ff	5
STX <i>opr8,SP</i>	SP1	9EEF	ff	4

SUB

Subtract

SUB

Operation $A \leftarrow (A) - (M)$ **Description**

Subtracts the contents of M from A and places the result in A

**Condition Codes
and Boolean
Formulae**

V			H	I	N	Z	C
↑	1	1	—	—	↑	↑	↑

V: $\overline{A7 \& M7 \& R7} \mid \overline{A7 \& M7 \& R7}$

Set if a two's complement overflow resulted from the operation; cleared otherwise. Literally read, an overflow condition occurs if a positive number is subtracted from a negative number with a positive result, or, if a negative number is subtracted from a positive number with a negative result.

N: R7

Set if MSB of result is 1; cleared otherwise

Z: $\overline{R7 \& R6 \& R5 \& R4 \& R3 \& R2 \& R1 \& R0}$

Set if result is \$00; cleared otherwise

C: $\overline{A7 \& M7} \mid \overline{M7 \& R7} \mid \overline{R7 \& A7}$

Set if the unsigned value of the contents of memory is larger than the unsigned value of the accumulator; cleared otherwise

**Source Forms,
Addressing
Modes, Machine
Code, Cycles, and
Access Details**

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
SUB <i>#opr8i</i>	IMM	A0	ii	2
SUB <i>opr8a</i>	DIR	B0	dd	3
SUB <i>opr16a</i>	EXT	C0	hh ll	4
SUB <i>opr16,X</i>	IX2	D0	ee ff	4
SUB <i>opr8,X</i>	IX1	E0	ff	3
SUB <i>X</i>	IX	F0		2
SUB <i>opr16,SP</i>	SP2	9ED0	ee ff	5
SUB <i>opr8,SP</i>	SP1	9EE0	ff	4

SWI

Software Interrupt

SWI

Operation

$PC \leftarrow (PC) + \$0001$ Increment PC to return address
 $Push(PCL); SP \leftarrow (SP) - \$0001$ Push low half of return address
 $Push(PCH); SP \leftarrow (SP) - \$0001$ Push high half of return address
 $Push(X); SP \leftarrow (SP) - \$0001$ Push index register on stack
 $Push(A); SP \leftarrow (SP) - \$0001$ Push A on stack
 $Push(CCR); SP \leftarrow (SP) - \$0001$ Push CCR on stack
 $I \text{ bit} \leftarrow 1$ Mask further interrupts
 $PCH \leftarrow (\$FFFC)$ Vector fetch (high byte)
 $PCL \leftarrow (\$FFFD)$ Vector fetch (low byte)

Description

The program counter (PC) is incremented by 1 to point at the instruction after the SWI. The PC, index register, and accumulator are pushed onto the stack. The condition code register (CCR) bits are then pushed onto the stack, with bits V, H, I, N, Z, and C going into bit positions 7 and 4–0. Bit positions 6 and 5 contain 1s. The stack pointer is decremented by 1 after each byte of data is stored on the stack. The interrupt mask bit is then set. The program counter is then loaded with the address stored in the SWI vector located at memory locations \$FFFC and \$FFFD. This instruction is not maskable by the I bit.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
—	1	1	—	1	—	—	—

I: 1
Set

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
SWI	INH	83		9

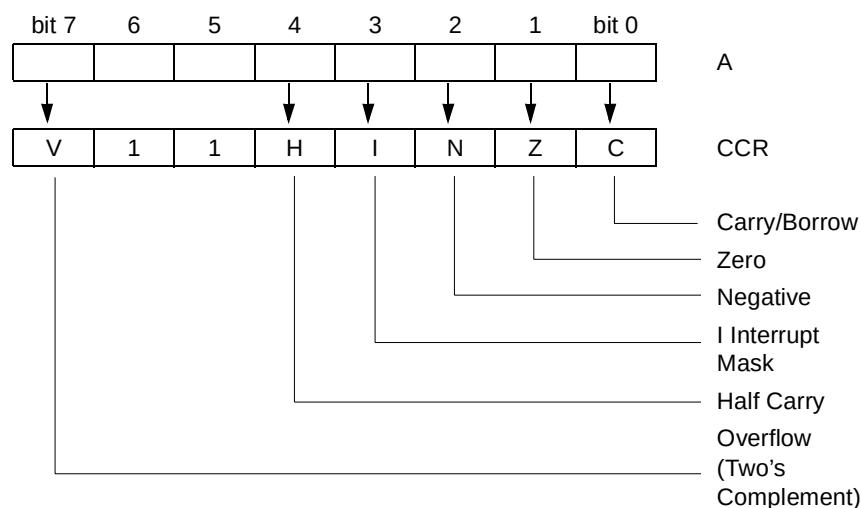
TAP

Transfer Accumulator to Processor Status Byte

TAP

Operation

$CCR \leftarrow (A)$



Description

Transfers the contents of A to the condition code register (CCR). The contents of A are unchanged. If this instruction causes the I bit to change from 0 to 1, a one bus cycle delay is imposed before interrupts become masked. This assures that the next instruction after a TAP instruction will always be executed even if an interrupt became pending during the TAP instruction.

Condition Codes and Boolean Formulae

V			H	I	N	Z	C
↓	1	1	↓	↓	↓	↓	↓

Set or cleared according to the value that was in the accumulator.

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
TAP	INH	84		2

TAX**Transfer Accumulator to X (Index Register Low)****TAX****Operation** $X \leftarrow (A)$ **Description**

Loads X with the contents of the accumulator (A). The contents of A are unchanged.

Condition Codes and Boolean Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
TAX	INH	97		1

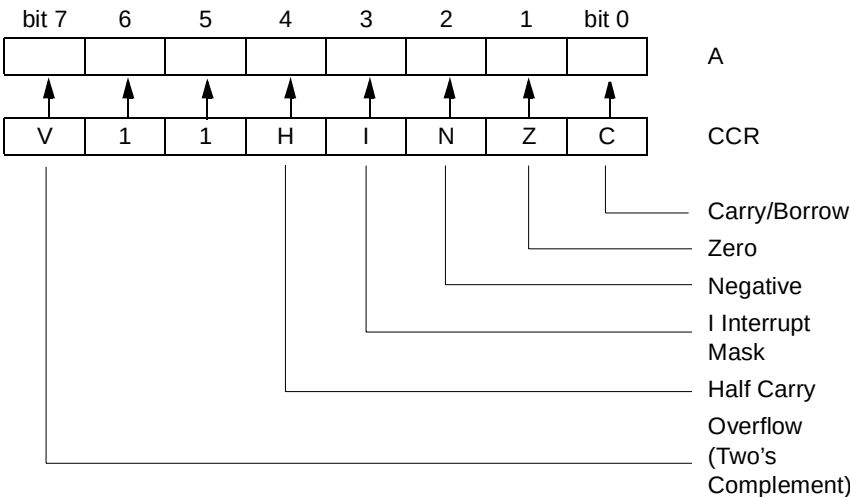
TPA

Transfer Processor Status Byte to Accumulator

TPA

Operation

$$A \leftarrow (CCR)$$



Description

Transfers the contents of the condition code register (CCR) into the accumulator (A)

Condition Codes
and Boolean
Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
TPA	INH	85		1

Instruction Set

TST

Test for Negative or Zero

TST

Operation

(A) – \$00
 Or (X) – \$00
 Or (M) – \$00

Description

Sets the N and Z condition codes according to the contents of A, X, or M. The contents of A, X, and M are not altered.

Condition Codes and Boolean Formulae

V		H		I	N	Z	C
0	1	1	—	—	↑	↑	—

V: 0
 Cleared

N: M7
 Set if MSB of the tested value is 1; cleared otherwise

Z: $\overline{M7} \& \overline{M6} \& \overline{M5} \& \overline{M4} \& \overline{M3} \& \overline{M2} \& \overline{M1} \& \overline{M0}$
 Set if A, X, or M contains \$00; cleared otherwise

Source Forms, Addressing Modes, Machine Code, Cycles, and Access Details

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
TST <i>opr8a</i>	DIR	3D	dd	3
TSTA	INH (A)	4D		1
TSTX	INH (X)	5D		1
TST <i>opr8,X</i>	IX1	6D	ff	3
TST <i>,X</i>	IX	7D		2
TST <i>opr8,SP</i>	SP1	9E6D	ff	4

TSX**Transfer Stack Pointer to Index Register****TSX****Operation** $H:X \leftarrow (SP) + \$0001$ **Description**

Loads index register (H:X) with 1 plus the contents of the stack pointer (SP). The contents of SP remain unchanged. After a TSX instruction, H:X points to the last value that was stored on the stack.

Condition Codes and Boolean Formulae

None affected

V	H		I	N	Z	C
—	1	1	—	—	—	—

Source Form, Addressing Mode, Machine Code, Cycles, and Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
TSX	INH	95		2

Instruction Set

TXA

Transfer X (Index Register Low) to Accumulator

TXA

Operation

 $A \leftarrow (X)$

Description

Loads the accumulator (A) with the contents of X. The contents of X are not altered.

Condition Codes
and Boolean
Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
TXA	INH	9F		1

TXS

Transfer Index Register to Stack Pointer

TXS

Operation

$$SP \leftarrow (H:X) - \$0001$$

Description

Loads the stack pointer (SP) with the contents of the index register (H:X) minus 1. The contents of H:X are not altered.

Condition Codes
and Boolean
Formulae

None affected

V			H	I	N	Z	C
—	1	1	—	—	—	—	—

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
TXS	INH	94		2

Instruction Set

WAIT

Enable Interrupts; Stop Processor

WAIT

Operation

I bit $\leftarrow$ 0; inhibit CPU clocking until interrupted

Description

Reduces power consumption by eliminating dynamic power dissipation in some portions of the MCU. The timer, the timer prescaler, and the on-chip peripherals continue to operate (if enabled) because they are potential sources of an interrupt. Wait causes enabling of interrupts by clearing the I bit in the CCR and stops clocking of processor circuits.

Interrupts from on-chip peripherals may be enabled or disabled by local control bits prior to execution of the WAIT instruction.

When either the $\overline{\text{RESET}}$ or $\overline{\text{IRQ}}$ pin goes low or when any on-chip system requests interrupt service, the processor clocks are enabled, and the reset, $\overline{\text{IRQ}}$, or other interrupt service request is processed.

Condition Codes
and Boolean
Formulae

V			H	I	N	Z	C
—	1	1	—	0	—	—	—

I: 0
Cleared

Source Form,
Addressing Mode,
Machine Code,
Cycles, and
Access Detail

Source Form	Address Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
WAIT	INH	8F		1