HCS12/9S12 Instruction Set Reference

ABA	(A)+(B) ⇒ A, Add accumulator A and B
ABX	(B)+(X) ⇒ X, Translates to LEAX B,X
ABY	(B)+(Y) ⇒ Y, Translates to LEAY B,Y
ADCA	(A)+(M) +C ⇒ A, Add with Carry to A
ADCB	(B)+(M) +C ⇒ B, Add with Carry to B
ADDA	(A)+(M) ⇒ A, Add without Carry to A
ADDB	(B)+(M) ⇒ B, Add without Carry to B
ADDD	(A:B)+(M:M+1) ⇒ A:B, add 16-bit to D
ANDA	(A)•(M) ⇒ A, Logical AND A with memory
ANDB	(B)•(M) ⇒ B, Logical AND B with memory
ANDCC	(CCR)•(M)⇒ CCR, Logical AND CCR with Memory
ASL	Arithmetic Shift Left
ASLA	Arithmetic Shift Left Accumulator A
ASLB	Arithmetic Shift Left Accumulator B
ASLD	Arithmetic Shift Left Accumulator D
ASR	Arithmetic Shift Right
ASRA	Arithmetic Shift Right Accumulator A
ASRB	Arithmetic Shift Right Accumulator B
BCC	Branch if Carry Clear (if C = 0)
BCLR	(M)•[$\overline{mm}$] ⇒ M, Clears Bit(s) in Memory
BCS	Branch if Carry Set (if C = 1)
BEQ	Branch if Equal (if Z = l)
BGE	Branch if Greater Than or Equal (if N⊕V = 0)(signed)
BGND	Place CPU in Background Mode
BGT	Branch if Greater Than (if Z+(N⊕V) = 0)(signed)
BHI	Branch if Higher
BHS	Branch if Higher or Same, (if C = 0)(unsigned)
BITA	(A)•(M) Logical AND A with memory, sets CCR
BITB	(B)•(M), Logical AND B with memory, sets CCR
BLE	Branch if Less Than or Equal, (if Z+(N⊕V) = 1)(signed)
BLO	Branch if Lower, if (C=1)(unsigned), equivalent to BCS
BLS	Branch if Lower or Same, (if C+Z = 1)(unsigned)
BLT	Branch if Less Than, (if N⊕V = 1)(signed)
BMI	Branch if Minus, (if N = l)
BNE	Branch if Not Equal, (if Z = 0)
BPL	Branch if Plus, (if N = 0)
BRA	Branch Always, A(if 1 = 1)
BRCLR	Branch if [M]•[mm] = 0, (if all selected Bit(s) Clear)
BRN	Branch Never,(if 1 = 0)
BRSET	Branch if ($\overline{M}$)•(mm) = 0, (if all selected Bit(s) Set)
BSET	(M)+(mm) ⇒ M, Set Bit(s) in memory
BSR	Branch to Subroutine
BVC	Branch if Overflow Bit Clear (if V = 0)
BVS	Branch if Overflow Bit Set (if V = 1)
CALL	Call Subroutine in Extended Memory
CBA	(A)–(B), Compare 8-bit Accumulators
CLC	0 ⇒ C, Clear Carry Bit
CLI	0 ⇒ I, Enable Interrupts
CLR	0 ⇒ M, Clear Memory Location
CLRA	0 ⇒ A, Clear Accumulator A
CLRB	0 ⇒ B, Clear Accumulator B
CLV	0 ⇒ V, Clear Overflow Bit
CMPA	(A)–(M), Compare Accumulator A with Memory
CMPB	(B)–(M), Compare Accumulator B with Memory
COM	$\left(\overline{M}\right)$ ⇒ M, One's Complement Memory Location
COMA	$\left(\overline{A}\right)$ ⇒ A, One's Complement Accumulator A
COMB	$\left(\overline{B}\right)$ ⇒ B, One's Complement Accumulator B
CPD	(A:B)–(M:M+1), Compare D to Memory (16-Bit)
CPS	(SP)–(M:M+1), Compare SP to Memory (16-Bit)
CPX	(X)–(M:M+1), Compare X to Memory (16-Bit)
CPY	(Y)–(M:M+1), Compare Y to Memory (16-Bit)
DAA	Adjust Sum to BCD
DBEQ	Decrement Counter and Branch if Equal to 0
DBNE	Decrement Counter and Branch if Not Equal to 0
DEC	Decrement Memory Location
DES	Decrement Stack Pointer
DEX	Decrement Index Register X
DEY	Decrement Index Register Y
EDIV	(Y:D)÷(X) ⇒ Y, Remainder ⇒ D, (unsigned)
EDIVS	(Y:D)÷(X) ⇒ Y, Remainder ⇒ D, (signed)
EMACS	$(M_{(x)}:M_{(x+1)})\times(M_{(Y)}:M_{(Y+1)})+(M\sim M_{+3}) \Rightarrow M\sim M_{+3}$, (signed)
EMAXD	Max{[D],[M:M+1]} ⇒ D, (unsigned)
EMAXM	Max{[D],[M:M+1]} ⇒ M:M+1, (unsigned)
EMIND	Min{[D],[M:M+1]} ⇒ D, (unsigned)
EMINM	Min{[D],[M:M+1]} ⇒ M:M+1, (unsigned)
EMUL	(Y)×(D) ⇒ Y:D, 16 by 16 Bit Multiply (unsigned)
EMULS	(Y)×(D) ⇒ Y:D, 16 by 16 Bit Multiply (signed)
EORA	(A)⊕ (M) ⇒ A, Exclusive OR A with Memory
EORB	(B)⊕ (M) ⇒ B, Exclusive OR B with Memory
ETBL	$(M:M_{+1})\times[(B)\times((M_{+2}:M_{+3})-(M:M_{+1}))] \Rightarrow$ D, Interpolate
EXG	Exchange Register to Register
FDIV	(D)÷(X) ⇒ X, Remainder ⇒ D, Fractional Divide
IBEQ	Increment Counter and Branch if = 0
IBNE	Increment Counter and Branch if ≠ 0
IDIV	(D)÷(X)⇒X, Remainder ⇒ D, Integer Divide, (unsigned)
IDIVS	(D)÷(X)⇒X, Remainder ⇒ D, Integer Divide, (signed)
INC	(M)+1 ⇒ M, Increment Memory Location
INCA	(A)+1 ⇒ A, Increment Accumulator A
INCB	(B)+1 ⇒ B, Increment Accumulator B
INS	(SP)+1 ⇒ B, equivalent to LEAS 1, SP
INX	(X)+1 ⇒ X, Increment Index Register X
INY	(Y)+1 ⇒ Y, Increment Index Register Y
JMP	Jump
JSR	Jump to Subroutine
LBCC	Long Branch if Carry Clear (if C = 0)
LBCS	Long Branch if Carry Set (if C = 1)
LBEQ	Long Branch if Equal (if Z = 1)
LBGE	Long Branch if Greater or Equal, (if N⊕V = 0), (signed)
LBGT	Long Branch if Greater Than (if Z+(N⊕V) = 0), (signed)
LBHI	Long Branch if Higher (if C+Z = 0), (unsigned)
LBHS	Long Branch if Higher or Same, (if C = 0), (unsigned)
LBLE	Long Branch if ≤, (if Z+(N⊕V)=1), (signed)
LBLO	Long Branch if Lower, if (C=1), (unsigned)
LBLS	Long Branch if Lower or Same, if (C+Z=1), (unsigned)
LBLT	Long Branch if Less Than, if (N⊕V = 1), (signed)
LBMI	Long Branch if Minus (if N = 1)
LBNE	Long Branch if Not Equal (if Z = 0)
LBPL	Long Branch if Plus (if N = 0)
LBRA	Long Branch Always (if 1 = 1)
LBRN	Long Branch Never (if 1 = 0)
LBVC	Long Branch if Overflow Bit Clear (if V = 0)
LBVS	Long Branch if Overflow Bit Set (if V = 1)
LDAA	(M) ⇒ A, Load Accumulator A
LDAB	(M) ⇒ B, Load Accumulator B
LDD	(M:M+1) ⇒ A:B, Load Double Accumulator D
LDS	(M:M+1) ⇒ SP, Load Stack Pointer
LDX	(M:M+1) ⇒ X, Load Index Register X
LDY	(M:M+1) ⇒ Y, Load Index Register Y
LEAS	Effective Address ⇒ SP, Load Effective Address into SP
LEAX	Effective Address ⇒ X, Load Effective Address into X
LEAY	Effective Address ⇒ Y, Load Effective Address into Y
LSL	Logical Shift Left (same function as ASL)
LSLA	Logical Shift Accumulator A to Left
LSLB	Logical Shift Accumulator B to Left
LSLD	Logical Shift Left D Accumulator (equiv. to ASLD)
LSR	Logical Shift Right
LSRA	Logical Shift Accumulator A to Right
LSRB	Logical Shift Accumulator B to Right
LSRD	Logical Shift Right D Accumulator
MAXA	MAX((A),(M)) ⇒ A, MAX of 2 Unsigned 8-bit Values
MAXM	MAX((A),(M)) ⇒ M, MAX of 2 Unsigned 8-bit Values
MEM	Membership function for Fuzzy Logic
MINA	MIN((A),(M)) ⇒ A, MIN of 2 Unsigned 8-bit Values
MINM	MIN((A),(M)) ⇒ M, MIN of 2 Unsigned 8-bit Values
MOVB	$(M_1) \Rightarrow M_2$, Memory to Memory Byte-Move (8 Bit)
MOVW	$(M:M+1_1) \Rightarrow M:M+1_2$, Memory to Memory Word-Move
MUL	(A)×(B) ⇒ A:B, 8 by 8 Unsigned Multiply
NEG	0–(M) ⇒ M, Two's Complement Negate
NEGA	0–(A) ⇒ A, Negate Accumulator A
NEGB	0–(B) ⇒ B, Negate Accumulator B
NOP	No Operation
ORAA	(A)+(M) ⇒ A, Logical OR A with Memory
ORAB	(B)+(M) ⇒ B, Logical OR B with Memory
ORCC	(CCR)+M ⇒ CCR, Logical OR CCR with Memory
PSHA	$(SP)-1 \Rightarrow SP$, (A) ⇒ $M_{(SP)}$, Push A onto Stack
PSHB	$(SP)-1 \Rightarrow SP$, (B) ⇒ $M_{(SP)}$, Push B onto Stack

HCS12/9S12 Instruction Set Reference

PSHC	(SP)–1 ⇒ SP, (CCR) ⇒ $M_{(SP)}$, Push CCR onto Stack
PSHD	(SP)–2 ⇒ SP, (A:B) ⇒ $M_{(SP)}$:$M_{(SP+1)}$, Push D onto Stack
PSHX	(SP)–2 ⇒ SP, (X_H:X_L)⇒ $M_{(SP)}$:$M_{(SP+1)}$, Push X onto Stack
PSHY	(SP)–2 ⇒ SP, (Y_H:Y_L)⇒ $M_{(SP)}$:$M_{(SP+1)}$, Push Y onto Stack
PULA	($M_{(SP)}$) ⇒ A, (SP)+1 ⇒ SP, Pull A from Stack
PULB	($M_{(SP)}$) ⇒ B, (SP)+1 ⇒ SP, Pull B from Stack
PULC	($M_{(SP)}$) ⇒ CCR, (SP)+1 ⇒ SP, Pull CCR from Stack
PULD	($M_{(SP)}$:$M_{(SP+1)}$) ⇒ A:B, (SP)+2 ⇒ SP, Pull D from Stack
PULX	($M_{(SP)}$:$M_{(SP+1)}$)⇒ X_H:X_L, (SP)+2 ⇒ SP, Pull X from Stack
PULY	($M_{(SP)}$:$M_{(SP+1)}$)⇒ Y_H:Y_L, (SP)+2 ⇒ SP, Pull Y from Stack
REV	Find smallest rule input (MIN). (8-bit offset)
REVW	Find smallest rule input (MIN). (16-bit offset)
ROL	Rotate Memory Left through Carry
ROLA	Rotate A Left through Carry
ROLB	Rotate B Left through Carry
ROR	Rotate Memory Right through Carry
RORA	Rotate A Right through Carry
RORB	Rotate B Right through Carry
RTC	Return from Call
RTI	Return from Interrupt
RTS	Return from Subroutine
SBA	Subtract B from A
SBCA	Subtract with Borrow from A
SBCB	Subtract with Borrow from B
SEC	1 ⇒ C, Translates to ORCC #$01
SEI	1 ⇒ I, Translates to ORCC #$10
SEV	1 ⇒ V, Translates to ORCC #$02
SEX	Sign Extend 8-bit r1 to 16-bit r2
STAA	(A) ⇒ M, Store A to Memory
STAB	(B) ⇒ M, Store B to Memory
STD	(A) ⇒ M, (B) ⇒ M+1 Store D to Memory
STOP	Stop All Clocks
STS	(SP_H:SP_L) ⇒ M:M+1 Store Stack Pointer
STX	(X_H:X_L) ⇒ M:M+1 Store Index Register X
STY	(Y_H:Y_L) ⇒ M:M+1 Store Index Register Y
SUBA	(A)–(M) ⇒ A, Subtract Memory from A
SUBB	(B)–(M) ⇒ B, Subtract Memory from B
SUBD	(B)–(M:M+1) ⇒ D, Subtract Memory from D
SWI	Software Interrupt
TAB	Transfer A to B
TAP	Translates to TFR A, CCR
TBA	Translate B to A
TBEQ	Test Counter and Branch if Zero
TBL	8-bit Table Lookup and Interpolate
TBNE	Test Counter and Branch if Not Zero
TFR	Transfer Register to Register
TPA	Translates to TFR CCR A
TRAP	Unimplemented opcode trap
TST	Test Memory for Zero or Minus
TSX	Translates to TFR SP, X
TSY	Translates to TFR SP, Y
TXS	Translates to TFR X, SP
TYS	Translates to TFR Y, SP
WAI	Wait for Interrupt
WAV	Calculate Sum of Product and Sum of Weights for Weighted Average Calculation
WAVR	Resume executing an interrupted WAV instruction
XGDX	(D) ⇔ (X), Translates to EXG D, X
XGDY	(D) ⇔ (Y), Translates to EXG D, Y