

HCS12/9S12 Instruction Set Reference

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

HCS12/9S12 Instruction Set Reference

PSHC	(SP)-1 ⇒ SP, (CCR) ⇒ M _(SP) , Push CCR onto Stack	SEX	Sign Extend 8-bit r1 to 16-bit r2
PSHD	(SP)-2 ⇒ SP, (A:B) ⇒ M _(SP) :M _(SP+1) , Push D onto Stack	STAA	(A) ⇒ M, Store A to Memory
PSHX	(SP)-2 ⇒ SP, (X _H :X _L) ⇒ M _(SP) :M _(SP+1) , Push X onto Stack	STAB	(B) ⇒ M, Store B to Memory
PSHY	(SP)-2 ⇒ SP, (Y _H :Y _L) ⇒ M _(SP) :M _(SP+1) , Push Y onto Stack	STD	(A) ⇒ M, (B) ⇒ M+1 Store D to Memory
PULA	M _(SP) ⇒ A, (SP)+1 ⇒ SP, Pull A from Stack	STOP	Stop All Clocks
PULB	M _(SP) ⇒ B, (SP)+1 ⇒ SP, Pull B from Stack	STS	(SP _H :SP _L) ⇒ M:M+1 Store Stack Pointer
PULC	M _(SP) ⇒ CCR, (SP)+1 ⇒ SP, Pull CCR from Stack	STX	(X _H :X _L) ⇒ M:M+1 Store Index Register X
PULD	M _(SP) :M _(SP+1) ⇒ A:B, (SP)+2 ⇒ SP, Pull D from Stack	STY	(Y _H :Y _L) ⇒ M:M+1 Store Index Register Y
PULX	M _(SP) :M _(SP+1) ⇒ X _H :X _L , (SP)+2 ⇒ SP, Pull X from Stack	SUBA	(A)-(M) ⇒ A, Subtract Memory from A
PULY	M _(SP) :M _(SP+1) ⇒ Y _H :Y _L , (SP)+2 ⇒ SP, Pull Y from Stack	SUBB	(B)-(M) ⇒ B, Subtract Memory from B
REV	Find smallest rule input (MIN). (8-bit offset)	SUBD	(B)-(M:M+1) ⇒ D, Subtract Memory from D
REVW	Find smallest rule input (MIN). (16-bit offset)	SWI	Software Interrupt
ROL	Rotate Memory Left through Carry	TAB	Transfer A to B
ROLA	Rotate A Left through Carry	TAP	Translates to TFR A, CCR
ROLB	Rotate B Left through Carry	TBA	Translate B to A
ROR	Rotate Memory Right through Carry	TBEQ	Test Counter and Branch if Zero
RORA	Rotate A Right through Carry	TBL	8-bit Table Lookup and Interpolate
RORB	Rotate B Right through Carry	TBNE	Test Counter and Branch if Not Zero
RTC	Return from Call	TFR	Transfer Register to Register
RTI	Return from Interrupt	TPA	Translates to TFR CCR A
RTS	Return from Subroutine	TRAP	Unimplemented opcode trap
SBA	Subtract B from A	TST	Test Memory for Zero or Minus
SBCA	Subtract with Borrow from A	TSX	Translates to TFR SP, X
SBCB	Subtract with Borrow from B	TSY	Translates to TFR SP, Y
SEC	1 ⇒ C, Translates to ORCC #S01	TXS	Translates to TFR X, SP
SEI	1 ⇒ I, Translates to ORCC #S10	TYS	Translates to TFR Y, SP
SEV	1 ⇒ V, Translates to ORCC #S02	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

