# The HC12/S12 Instruction Set

| | | N | Z | V | C | |
|---|---|---|---|---|---|---|
| LDAA | (M) -> A | △ | △ | 0 | - | Load A register from memory |
| LDAB | (M) -> B | △ | △ | 0 | - | Load B register from memory |
| LDD | (M:M+1) -> (A:B) | △ | △ | 0 | - | Load D register (A:B) from memory |
| LDS | (M:M+1) -> S | △ | △ | 0 | - | Load S register from memory |
| LDX | (M:M+1) -> X | △ | △ | 0 | - | Load X register from memory |
| LDY | (M:M+1) -> Y | △ | △ | 0 | - | Load Y register from memory |

| | | | | | | |
|---|---|---|---|---|---|---|
| STAA | (A) -> M | △ | △ | 0 | - | Store A register data in memory |
| STAB | (B) -> M | △ | △ | 0 | - | Store B register data in memory |
| STD | (M:M+1) -> (A:B) | △ | △ | 0 | - | (D)=(A:B) -> M:M+1 |
| STS | (SP) -> M:M+1 | △ | △ | 0 | - | Store SP register data in memory |
| STX | (X) -> M:M+1 | △ | △ | 0 | - | Store X register data in memory |
| STY | (Y) -> M:M+1 | △ | △ | 0 | - | Store Y register data in memory |

| | | N | Z | V | C | |
|---|---|---|---|---|---|---|
| EXG | (R1) <-> (R2) | - | - | - | - | Exchange data in R1 and R2 registers.<br>R1, R2 = A, B, CCR, D, X, Y or SP |
| XGDX | (D) <-> (X) | - | - | - | - | Exchange data in D and X registers |
| XGDY | (D) <-> (Y) | - | - | - | - | Exchange data in D and Y registers |

| | | | | | | |
|---|---|---|---|---|---|---|
| MOVB | (M1) -> M2 | - | - | - | - | Copy 8-bit data in memory location M1 to location M2 |
| MOVW | (M1:M1+1) -> M2:M2+1 | - | - | - | - | Copy 16-bit data in memory locations M1:M1+1 to locations M2:M2+1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| SEX | (A,B,CCR) -> X,Y, or SP | - | - | - | - | "Extends" MSB of 8-bit data to fill high byte<br>Example: *SEX A,X* |

| | | N | Z | V | C | |
|---|---|---|---|---|---|---|
| INC | (M) + $01 -> M | △ | △ | △ | - | Increment data in M by one |
| INCA | (A) + $01 -> A | △ | △ | △ | - | Increment data in Register A by one |
| INCB | (B) + $01 -> B | △ | △ | △ | - | Increment data in Register B by one |
| INS | (SP) + $01 -> SP | △ | △ | △ | - | Increment data in Register SP by one |
| INX | (X) + $01 -> X | △ | △ | △ | - | Increment data in Register X by one |
| INY | (Y) + $01 -> Y | △ | △ | △ | - | Increment data in Register Y by one |

| | | | | | | |
|---|---|---|---|---|---|---|
| DEC | (M) - $01 -> M | △ | △ | △ | - | Decrement data in M by one |
| DECA | (A) - $01 -> A | △ | △ | △ | - | Decrement data in Register A by one |
| DECB | (B) - $01 -> B | △ | △ | △ | - | Decrement data in Register B by one |
| DES | (SP) - $01 -> SP | △ | △ | △ | - | Decrement data in Register SP by one |
| DEX | (X) - $01 -> X | △ | △ | △ | - | Decrement data in Register X by one |
| DEY | (Y) - $01 -> Y | △ | △ | △ | - | Decrement data in Register Y by one |

| | | H | N | Z | V | C | |
|---|---|---|---|---|---|---|---|
| ABA | (A) + (B) -> A | △ | △ | △ | △ | △ | Add data in A and B, Store in A |
| ADDA | (A) + (M) -> A | △ | △ | △ | △ | △ | Add data in A and M without carry, Store in A |
| ADDB | (B) + (M) -> B | △ | △ | △ | △ | △ | Add data in B and M without carry, Store in B |
| ADCA | (A) + (M) + C -> A | △ | △ | △ | △ | △ | Add data in A and M with carry, Store in A |
| ADCB | (B) + (M) + C -> B | △ | △ | △ | △ | △ | Add data in B and M with carry, Store in B |
| ADDD | (D) + (M:M+!) -> D | - | △ | △ | △ | △ | Add data in D and M without carry, Store in D |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SBA | (A) - (B) -> A | - | △ | △ | △ | △ | Subtract data in B from A, Store in A |
| SUBA | (A) - (M) -> A | - | △ | △ | △ | △ | Subtract data in M from A, Store in A (no borrow) |
| SUBB | (B) - (M) -> B | - | △ | △ | △ | △ | Subtract data in M from B, Store in B (no borrow) |
| SBCA | (A) - (M) -C -> A | - | △ | △ | △ | △ | Subtract data in M from A with borrow, Store in A |
| SBCB | (B) - (M) -C -> B | - | △ | △ | △ | △ | Subtract data in M from B with borrow, Store in B |
| SUBD | (D) - (M:M+!) -> D | - | △ | △ | △ | △ | Subtract data in M:M+1 from D, Store in D |

| | | N | Z | V | C | |
|---|---|---|---|---|---|---|
| LEAS | Eff addr -> S | - | - | - | - | Load *effective address* into S register |
| LEAX | Eff addr -> X | - | - | - | - | Load *effective address* into X register |
| LEAY | Eff addr -> Y | - | - | - | - | Load *effective address* into Y register |
| TAB | (A) -> B | △ | △ | 0 | - | Copy A register data to B register |
| TBA | (B) -> A | △ | △ | 0 | - | Copy B register data to A register |
| TAP | (A) -> CCR | △ | △ | 0 | - | Copy A register data to CCR register |
| TPA | (CCR) -> A | △ | △ | 0 | - | Copy CCR register data to A register |
| TSX | (SP) -> X | - | - | - | - | Copy SP register data to X register |
| TSY | (SP) -> Y | - | - | - | - | Copy SP register data to Y register |
| TXS | (X) -> SP | - | - | - | - | Copy X register data to SP register |
| TYS | (Y) -> SP | - | - | - | - | Copy Y register data to SP register |
| TFR | (R1) -> R2 | Depends | | | | R1, R2 = A, B, CCR, D, X, Y or SP |

| | | N | Z | V | C | |
|---|---|---|---|---|---|---|
| EMUL | (D) x (Y) -> Y:D | △ | △ | - | △ | Unsigned 16-bit multiply |
| EMULS | (D) x (Y) -> Y:D | △ | △ | - | △ | Signed 16-bit multiply |
| MUL | (A) x (B) -> A:B | - | - | - | △ | Unsigned 8-bit multiply |
| EDIV | (Y:D)/(X) -> Y,D | △ | △ | △ | △ | Unsigned 32-bit by 16 divide. Quotient -> Y<br>Remainder -> D |
| EDIVS | (Y:D)/(X) -> Y,D | △ | △ | △ | △ | Signed 32-bit by 16 divide. Quotient -> Y<br>Remainder -> D |
| FDIV | (D)/(X) -> X<br>Remainder -> D | - | - | - | △ | Unsigned 16-bit fixed-point divide |
| IDIV | (D)/(X) -> X<br>Remainder -> D | - | △ | 0 | △ | Unsigned 16 by 16 integer divide |
| IDIVS | (D)/(X) -> X<br>Remainder -> D | △ | △ | △ | △ | Signed 16 by 16 integer divide |
| BCLR | (M) AND $\overline{\text{Mask}}$ -> M | △ | △ | 0 | - | Clear bits in M for "1" valued bits in 8-bit mask<br>Example: *BCLR $2000, $F0* clears bit7 to bit4 |
| BSET | (M) OR Mask -> M | △ | △ | 0 | - | Set bits in M for "1" valued bits in 8-bit mask |
| BITA | (A) AND Mask | △ | △ | 0 | - | Tests if bits in A are "1" for bits in Mask = "1" Sets CCR bits only.<br>Example: *BITA #$44* |
| BITB | (B) AND Mask | △ | △ | 0 | - | Tests if bits in B are "1" for bits in Mask = "1" Sets CCR bits only. |

| | | N | Z | V | C | |
|---|---|---|---|---|---|---|
| ANDA | (A) AND (M) -> A | △ | △ | 0 | - | AND A with M, result in A |
| ANDB | (B) AND (M) -> B | △ | △ | 0 | - | AND B with M, result in B |
| ANDCC | (CCR) AND (M) -> CCR | x | x | x | x | Bit = 0 in M forces corresponding bit in CCR to 0 |
| EORA | (A)!(M) -> A | △ | △ | 0 | - | Exclusive OR of A with M, result in A |
| EORB | (B)!(M) -> B | △ | △ | 0 | - | Exclusive OR of B with M, result in B |
| ORAA | (A) OR (M) -> A | △ | △ | 0 | - | OR of A with M, result in A |
| ORAB | (B) OR (M) -> B | △ | △ | 0 | - | OR of B with M, result in B |
| ORCC | (CCR) OR (M) -> CCR | x | x | x | x | Bit = 1 in M forces corresponding bit in CCR to 1 |
| CLC | 0 -> C in CCR | - | - | - | 0 | Clear C bit in CCR, others not affected |
| CLI | 0 -> I in CCR | - | - | - | - | Clear I bit in CCR, others not affected |
| CLV | 0 -> V in CCR | - | - | 0 | - | Clear V bit in CCR, others not affected |
| COM | $FF - (M) -> M | △ | △ | 0 | 1 | One's complement of data in M, result in M |
| COMA | $FF - (A) -> A | △ | △ | 0 | 1 | One's complement of data in A, result in A |
| COMB | $FF - (B) -> B | △ | △ | 0 | 1 | One's complement of data in B, result in B |
| NEG | $00 - (M) -> M | △ | △ | △ | △ | Two's complement of data in M, result in M |
| NEGA | $00 - (A) -> A | △ | △ | △ | △ | Two's complement of data in A, result in A |
| NEGB | $00 - (B) -> B | △ | △ | △ | △ | Two's complement of data in B, result in B |