The HC12/S12 Instruction Set

(Tewksbury)

1. Load Data into Registers

		N	Z	V	C	
LDAA	(M) -> A	Δ	Δ	0	–	Load A register from memory
LDAB	(M) -> B	Δ	Δ	0	–	Load B register from memory

LDD	(M:M+1) -> (A:B)	Δ	Δ	0	–	Load D register (A:B) from memory
LDS	(M:M+1) -> S	Δ	Δ	0	–	Load S register from memory
LDX	(M:M+1) -> X	Δ	Δ	0	–	Load X register from memory
LDY	(M:M+1) -> Y	Δ	Δ	0	–	Load Y register from memory

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

2. Store Register Data in Memory

		N	Z	V	C	
STAA	(A) -> M	Δ	Δ	0	–	Store A register data in memory
STAB	(B) -> M	Δ	Δ	0	–	Store B register data in memory

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

3. Transfer Register Data To Another Register

		N	Z	V	C	
TAB	(A) -> B	Δ	Δ	0	–	Copy A register data to B register
TBA	(B) -> A	Δ	Δ	0	–	Copy B register data to A register
TAP	(A) -> CCR	Δ	Δ	0	–	Copy A register data to CCR register
TPA	(CCR) -> A	Δ	Δ	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

4. Exchange Register Data Between Registers

		N	Z	V	C	
EXG	(R1) <-> (R2)	–	–	–	–	Exchange data in R1 and R2 registers. 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

5. Sign Extend 8-bit Signed Data into 16-bit Register

		N	Z	V	C	
SEX	(A,B,CCR) -> X,Y, or SP	–	–	–	–	"Extends" MSB of 8-bit data to fill high byte of 16-bit data. Example: *SEX A,X*

6. Move Memory Data to Another Memory Location

		N	Z	V	C	
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

7. Addition Instructions

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

8. Decrement and Increment Instructions

		N	Z	V	C	
DEC	(M) – $01 -> M	Δ	Δ	Δ	-	Decrement data in M by one
DECA	(A) – $01 -> A	Δ	Δ	Δ	-	Decrement data in Register A by one
DECB	(B) – $01 -> B	Δ	Δ	Δ	-	Decrement data in Register B by one
DES	(SP) – $01 -> SP	Δ	Δ	Δ	-	Decrement data in Register SP by one
DEX	(X) – $01 -> X	Δ	Δ	Δ	-	Decrement data in Register X by one
DECY	(Y) – $01 -> Y	Δ	Δ	Δ	-	Decrement data in Register Y by one

INC	(M) + $01 -> M	Δ	Δ	Δ	-	Increment data in M by one
INCA	(A) + $01 -> A	Δ	Δ	Δ	-	Increment data in Register A by one
INCB	(B) + $01 -> B	Δ	Δ	Δ	-	Increment data in Register B by one
INS	(SP) + $01 -> SP	Δ	Δ	Δ	-	Increment data in Register SP by one
INX	(X) + $01 -> X	Δ	Δ	Δ	-	Increment data in Register X by one
INY	(Y) + $01 -> Y	Δ	Δ	Δ	-	Increment data in Register Y by one

9. Subtraction Instructions

		H	N	Z	V	C	
SBA	(A) – (B) -> A	-	Δ	Δ	Δ	Δ	Subtract data in B from A, Store in A
SUBA	(A) – (M) -> A	-	Δ	Δ	Δ	Δ	Subtract data in M from A, Store in A (no borrow)
SUBB	(B) – (M) -> B	-	Δ	Δ	Δ	Δ	Subtract data in M from B, Store in B (no borrow)

SBCA	(A) – (M) -C -> A	-	Δ	Δ	Δ	Δ	Subtract data in M from A with borrow, Store in A
SBCB	(B) – (M) -C -> B	-	Δ	Δ	Δ	Δ	Subtract data in M from B with borrow, Store in B

SUBD	(D) – (M:M+!) -> D	–	Δ	Δ	Δ	Δ	Subtract data in M:M+1 from D, Store in D

10. Multiply Instructions

Note: Multiplication of an *M*-bit number by an *N*-bit number generates an *(M+N)*-bit number.

		N	Z	V	C	
EMUL	(D) x (Y) -> Y:D	Δ	Δ	-	Δ	Unsigned 16-bit multiply
EMULS	(D) x (Y) -> Y:D	Δ	Δ	-	Δ	Signed 16-bit multiply

MUL	(A) x (B) -> A:B	-	-	-	Δ	Unsigned 8-bit multiply

11. Divide Instructions

Note: Division of an *2N*-bit integer by an *N*-bit integer generates an *N*-bit quotient (integer) and *N*-bit remainder (after decimal point).

		N	Z	V	C	
EDIV	(Y:D)/(X) -> Y,D	Δ	Δ	Δ	Δ	Unsigned 32-bit by 16 divide. Quotient -> Y Remainder -> D
EDIVS	(Y:D)/(X) -> Y,D	Δ	Δ	Δ	Δ	Signed 32-bit by 16 divide. Quotient -> Y Remainder -> D

FDIV	(D)/(X) -> X Remainder -> D	-	-	-	Δ	Unsigned 8-bit multiply

IDIV	(D)/(X) -> X Remainder -> D	-	Δ	0	Δ	Unsigned 16 by 16 integer divide
IDIVS	(D)/(X) -> X Remainder -> D	Δ	Δ	Δ	Δ	Signed 16 by 16 integer divide

12. Bit Test And Manipulate

		N	Z	V	C	
BCLR	(M) AND Mask -> M	Δ	Δ	0	-	Clear bits in M for "0" valued bits in 8-bit mask Example: *BCLR $2000, $F0*
BSET	(M) OR Mask -> M	Δ	Δ	0	-	Set bits in M for "1" valued bits in 8-bit mask
BITA	(A) AND Mask	Δ	Δ	0	-	Tests if bits in A are "1" for bits in Mask = "1" Sets CCR bits only. Example: *BITA #$44*
BITB	(B) AND Mask	Δ	Δ	0	-	Tests if bits in B are "1" for bits in Mask = "1" Sets CCR bits only.

13. Logical Shift, Arithmetic Shift, and Rotate Instructions

These instructions shift the bit positions of a binary number to the right or left by one position. There are three different types of this operation, summarized below. For each type and direction, the N, Z, V bits of the CCR are set according to the data after the shift/rotate and the C bit of the CCR holds the data bit shifted/rotated out of the binary word. See pages 29-31 of your textbook for figures showing what's happening.

- Logic shift instructions: These are used essentially to move data bits into the "C" (carry) bit of the CCR register.
    - When shifting to the left, 0 is entered into the least significant bit of the number and the most significant bit is shifted into the "C" carry bit of the CCR.
    - When shifting to the right, 0 is entered into the most significant bit position and the least significant bit is shifted into the "C" carry bit of the CCR.
- Arithmetic shift instructions:
    - Arithmetic shift to the left is the same as the logic shift operation above. 0 is entered into the least significant bit of the number and the most significant bit is shifted into the "C" carry bit of the CCR.
    - When shifting to the right, the number is assumed to be a two's complement signed number (with the most significant bit extended indefinitely to the right. Therefore, the most significant bit is re-entered into the most significant bit position. The least significant bit is shifted into the "C" carry bit of the CCR.

- Rotate Instructions: These instructions rotate the binary number "through the "C" carry bit of the CCR
    - When shifting to the left, the "C" carry bit of the CCR is entered into the least significant bit position and the most significant is entered into the "C" carry bit of the CCR.
    - When shifting to the right, the "C" carry bit of the CCR is entered into the most significant bit position and the least signficant bit is entered into 0 is entered into the "C" carry bit of the CCR.

12. Boolean Logic Instructions

Note: Division of an *M*-bit integer by an *N*-bit integer generates a quotient (integer) and remainder (after decimal point).

		N	Z	V	C	
ANDA	(A) AND (M) -> A	Δ	Δ	0	-	AND A with M, result in A
ANDB	(B) AND (M) -> B	Δ	Δ	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) $\oplus$ (M) -> A	Δ	Δ	0	-	Exclusive OR of A with M, result in A
EORB	(B) $\oplus$ (M) -> B	Δ	Δ	0	-	Exclusive OR of B with M, result in B
ORAA	(A) OR (M) -> A	Δ	Δ	0	-	OR of A with M, result in A
ORAB	(B) OR (M) -> B	Δ	Δ	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	Δ	Δ	0	1	One's complement of data in M, result in M
COMA	$FF - (A) -> A	Δ	Δ	0	1	One's complement of data in A, result in A
COMB	$FF - (B) -> B	Δ	Δ	0	1	One's complement of data in B, result in B

NEG	$00 - (M) -> M	Δ	Δ	Δ	Δ	Two's complement of data in M, result in M
NEGA	$00 - (A) -> A	Δ	Δ	Δ	Δ	Two's complement of data in A, result in A
NEGB	$00 - (B) -> B	Δ	Δ	Δ	Δ	Two's complement of data in B, result in B