- **Disassembly of MC9S12 op codes**
- **Decimal, Hexadecimal and Binary Numbers**
  - How to disassemble an MC9S12 instruction sequence
  - Binary numbers are a code and represent what the programmer intends for the code
  - Convert binary and hex numbers to unsigned decimal
  - Convert unsigned decimal to hex
  - Signed number representation – 2's complement form
  - Using the 1's complement table to find 2's complements of hex numbers
  - Overflow and Carry
  - Addition and subtraction of binary and hex numbers
  - The condition code register (CCR): N, Z, V and C bits

HC12 Instructions

1. Data Transfer and Manipulation Instructions — instructions which move and manipulate data (S12CPUV2 Reference Manual, Sections 5.3, 5.4, and 5.5).

• Load and Store — load copy of memory contents into a register; store copy of register contents into memory.

```
LDAA $2000  ; Copy contents of addr $2000 into A
STD 0,X     ; Copy contents of D to addrs X and X+1
```

• Transfer — copy contents of one register to another.

```
TBA         ; Copy B to A
TFR X,Y     ; Copy X to Y
```

• Exhange — exchange contents of two registers.

```
XGDX        ; Exchange contents of D and X
EXG A,B     ; Exchange contents of A and B
```

• Move — copy contents of one memory location to another.

```
MOVB $2000,$20A0 ; Copy byte at $2000 to $20A0
MOVW 2,X+,2,Y+   ; Copy two bytes from address held
                 ; in X to address held in Y
                 ; Add 2 to X and Y
```

2. Arithmetic Instructions — addition, subtraction, multiplication, divison (**S12CPUV2 Reference Manual**, Sections 5.6, 5.8 and 5.12).

```
ABA         ; Add B to A; results in A
```

```
SUBD $20A1  ; Subtract contents of $20A1 from D
INX         ; Increment X by 1
MUL         ; Multiply A by B; results in D
```

3. Logic and Bit Instructions — perform logical operations (**S12CPUV2 Reference Manual**, Sections 5.9, 5.10, 5.11, 5.13 and 5.14).

• Logic Instructions

```
ANDA $2000  ; Logical AND of A with contents of $2000
EORB 2,X    ; Exclusive OR B with contents of address (X+2)
```

• Clear, Complement and Negate Instructions

```
NEG -2,X    ; Negate (2's comp) contents of address (X-2)
CLRA        ; Clear Acc A
```

• Bit manipulate and test instructions — work with one bit of a register or memory.

```
BITA #$08          ; Check to see if Bit 3 of A is set
BSET $0002,#$18    ; Set bits 3 and 4 of address $002
```

• Shift and rotate instructions

```
LSLA        ; Logical shift left A
ASR $1000   ; Arithmetic shift right value at address $1000
```

4. Compare and test instructions — test contents of a register or memory (to see if zero, negative, etc.), or compare contents of a register to memory (to see if bigger than, etc.) (**S12CPUV2 Reference Manual**, Section 5.9).

```
TSTA        ; (A)-0 -- set flags accordingly
CPX #$8000  ; (X) - $8000 -- set flags accordingly
```

5. Jump and Branch Instructions — Change flow of program (e.g., goto, it-then-else, switch-case) (**S12CPUV2 Reference Manual**, Sections 5.19, 5.20 and 5.21).

```
JMP L1                  ; Start executing code at address label L1
BEQ L2                  ; If Z bit set, go to label L2
DBNE X,L3               ; Decrement X; if X not 0 then goto L3
BRCLR $1A,#$80,L4       ; If bit 7 of addr $1A clear, go to label L4
JSR sub1                ; Jump to subroutine sub1
RTS                     ; Return from subroutine
```

6. Interrupt Instructions — Initiate or terminate an interrupt call (**S12CPUV2 Reference Manual**, Section 5.22).

• Interrupt instructions

```
SWI   ; Initiate software interrupt
RTI   ; Return from interrupt
```

7. Index Manipulation Instructions — Put address into X, Y or SP, manipulate X, Y or SP (**S12CPUV2 Reference Manual**, Section 5.23).

```
ABX          ; Add (B) to (X)
LEAX 5,Y     ; Put address (Y) + 5 into X
```

8. Condition Code Instructions — change bits in Condition Code Register (**S12CPUV2 Reference Manual**, Section 5.26).

```
ANDCC #$f0  ; Clear N, Z, C and V bits of CCR
SEV         ; Set V bit of CCR
```

9. Stacking Instructions — push data onto and pull data off of stack (**S12CPUV2 Reference Manual**, Section 5.24).

```
PSHA  ; Push contents of A onto stack
PULX  ; Pull two top bytes of stack, put into X
```

10. Stop and Wait Instructions — put MC9S12 into low power mode (S12CPUV2 Reference Manual, Section 5.27).

```
STOP  ; Put into lowest power mode
WAI   ; Put into low power mode until next interrupt
```

11. Null Instructions

```
NOP   ; No operation
BRN   ; Branch never
```

12. Instructions we won't discuss or use — BCD arithmetic, fuzzy logic, minimum and maximum, multiply-accumulate, table interpolation (**S12CPUV2 Reference Manual**, Sections 5.7, 5.16, 5.17, and 5.18).

Disassembly of an HC12 Program

• It is sometimes useful to be able to convert *HC12 op codes* into *mnemonics*.

For example, consider the hex code:

ADDR DATA

1000 C6 05 CE 20 00 E6 01 18 06 04 35 EE 3F

• To determine the instructions, use Table A-2 of the HCS12 Core Users Guide.

– If the first byte of the instruction is anything other than **$18**, use Sheet 1 of Table A.2. From this table, determine the number of bytes of the instruction and the addressing mode. For example, **$C6** is a two-byte instruction, the mnemonic is **LDAB**, and it uses the **IMM** addressing mode. Thus, the two bytes **C6 05** is the op code for the instruction **LDAB #$05**.

– If the first byte is **$18**, use Sheet 2 of Table A.2, and do the same thing. For example, **18 06** is a two byte instruction, the mnemonic is **ABA**, and it uses the **INH** addressing mode, so there is no operand. Thus, the two bytes **18 06** is the op code for the instruction **ABA**.

– Indexed addressing mode is fairly complicated to disassemble. You need to use Table A.3 to determine the operand. For example, the op code **$E6** indicates **LDAB indexed**, and may use two to four bytes (one to three bytes in addition to the op code). The postbyte **01** indicates that the operand is 0,1, which is **5-bit constant offset**, which takes only one additional byte. All 5-bit constant offset, pre and post increment and decrement, and register offset instructions use one additional byte. All **9-bit constant offset** instructions use two additional bytes, with the second byte holding 8 bits of the 9 bit offset. (**The 9th bit is a direction bit**, which is held in the first postbyte.) All 16-bit constant offset instructions use three postbytes, with the 2nd and 3rd holding the 16-bit unsigned offset.

– Transfer (**TFR**) and exchange (**EXG**) instructions all have the op code **$B7**. Use Table A.5 to determine whether it is **TFR** or an **EXG**, and to determine which registers are being used. If the most significant bit of the postbyte is **0, the instruction is a transfer instruction**.

– Loop instructions (Decrement and Branch, Increment and Branch, and Test and Branch) all have the op code **$04**. To determine which instruction the op code **$04** implies, and whether the branch is positive (forward) or negative (backward), use Table A.6. For example, in the sequence **04 35 EE**, the 04 indicates a loop

instruction. The 35 indicates it is a **DBNE X** instruction (decrement register X and branch if result is not equal to zero), and the direction is backward (negative). The **EE** indicates a branch of -18 bytes.

• Use up all the bytes for one instruction, then go on to the next instruction.

C6 05	⇒ **LDAA #$05**	two-byte LDAA, IMM addressing mode
CE 20 00	⇒ **LDX #$2000**	three-byte LDX, IMM addressing mode
E6 01	⇒ **LDAB 1,X**	two to four-byte LDAB, IDX addressing mode. Operand 01 => 1,X, a 5b constant offset which uses only one postbyte
18 06	⇒ **ABA**	two-byte ABA, INH addressing mode
04 35 EE	⇒ **DBNE X,(-18)**	three-byte loop instruction Postbyte 35 indicates DBNE X, negative
3F	⇒ **SWI**	one-byte SWI, INH addressing mode

Table A-2. CPU12 Opcode Map (Sheet 1 of 2)

00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
BGND †5 IH 1	ANDCC 1 IM 2	BRA 3 RL 2	PULX 3 IH 1	NEGA 1 IH 1	NEGB 1 IH 1	NEG 3-6 ID 2-4	NEG 4 EX 3	SUBA 1 IM 2	SUBA 3 DI 2	SUBA 3-6 ID 2-4	SUBA 3 EX 3	SUBB 1 IM 2	SUBB 3 DI 2	SUBB 3-6 ID 2-4	SUBB 3 EX 3
MEM 5 IH 1	EDIV 11 IH 1	BRN 3 RL 2	PULY 3 IH 1	COMA 1 IH 1	COMB 1 IH 1	COM 3-6 ID 2-4	COM 4 EX 3	CMPA 1 IM 2	CMPA 3 DI 2	CMPA 3-6 ID 2-4	CMPA 3 EX 3	CMPB 1 IM 2	CMPB 3 DI 2	CMPB 3-6 ID 2-4	CMPB 3 EX 3
INY 1 IH 1	MUL 12 ‡1 IH 1	BHI 22 3/1 RL 2	PULA 3 IH 1	INCA 1 IH 1	INCB 1 IH 1	INC 3-6 ID 2-4	INC 4 EX 3	SBCA 1 IM 2	SBCA 3 DI 2	SBCA 3-6 ID 2-4	SBCA 3 EX 3	SBCB 1 IM 2	SBCB 3 DI 2	SBCB 3-6 ID 2-4	SBCB 3 EX 3
DEY 1 IH 1	EMUL 13 3 IH 1	BLS 23 3/1 RL 2	PULB 3 IH 1	DECA 1 IH 1	DECB 1 IH 1	DEC 3-6 ID 2-4	DEC 4 EX 3	SUBD 2 IM 3	SUBD 3 DI 2	SUBD 3-6 ID 2-4	SUBD 3 EX 3	ADDD 2 IM 3	ADDD 3 DI 2	ADDD 3-6 ID 2-4	ADDD 3 EX 3
loop * 3 RL 3	ORCC 4 IM 2	BCC 24 3/1 RL 2	PSHX 2 IH 1	LSRA 1 IH 1	LSRB 1 IH 1	LSR 3-6 ID 2-4	LSR 4 EX 3	ANDA 1 IM 2	ANDA 3 DI 2	ANDA 3-6 ID 2-4	ANDA 3 EX 3	ANDB 1 IM 2	ANDB 3 DI 2	ANDB 3-6 ID 2-4	ANDB 3 EX 3
JMP 2-4 ID	JSR 4-7 ID 2-4	BCS 25 3/1 RL 2	PSHY 2 IH 1	ROLA 1 IH 1	ROLB 1 IH 1	ROL 3-6 ID 2-4	ROL 4 EX 3	BITA 1 IM 2	BITA 3 DI 2	BITA 3-6 ID 2-4	BITA 3 EX 3	BITB 1 IM 2	BITB 3 DI 2	BITB 3-6 ID 2-4	BITB 3 EX 3
JMP 3 EX 3	JSR 4 EX 3	BNE 26 3/1 RL 2	PSHA 2 IH 1	RORA 1 IH 1	RORB 1 IH 1	ROR 3-6 ID 2-4	ROR 4 EX 3	LDAA 1 IM 2	LDAA 3 DI 2	LDAA 3-6 ID 2-4	LDAA 3 EX 3	LDAB 1 IM 2	LDAB 3 DI 2	LDAB 3-6 ID 2-4	LDAB 3 EX 3
BSR 4 RL 2	JSR 17 4 RL 2	BEQ 27 3/1 RL 2	PSHB 2 IH 1	ASRA 1 IH 1	ASRB 1 IH 1	ASR 3-6 ID 2-4	ASR 4 EX 3	CLRA 1 IH 1	TSTA 1 IH 1	NOP 1 IH 1	TFR/EXG 2 IH 1	CLRB 1 IH 1	TSTB 1 IH 1	TST 3-6 ID 2-4	TST 3 EX 3
INX 1 IH 1	Page 2 18 - -	BVC 28 3/1 RL 2	PULC 3 IH 1	ASLA 1 IH 1	ASLB 1 IH 1	ASL 3-6 ID 2-4	ASL 4 EX 3	EORA 1 IM 2	EORA 3 DI 2	EORA 3-6 ID 2-4	EORA 3 EX 3	EORB 1 IM 2	EORB 3 DI 2	EORB 3-6 ID 2-4	EORB 3 EX 3
DEX 1 IH 1	LEAY 19 2 ID 2-4	BVS 29 3/1 RL 2	PSHC 2 IH 1	LSRD 1 IH 1	ASLD 1 IH 1	CLR 3-6 ID 2-4	CLR 3 EX 3	ADCA 1 IM 2	ADCA 3 DI 2	ADCA 3-6 ID 2-4	ADCA 3 EX 3	ADCB 1 IM 2	ADCB 3 DI 2	ADCB 3-6 ID 2-4	ADCB 3 EX 3
RTC ‡7 IH 1	LEAX 1A 2 ID 2-4	BPL 2A 3/1 RL 2	PULD 3 IH 1	CALL 4A ‡7 EX 4	STAA 5A 2 DI 2	STAA 6A ‡2-4 ID 2-4	STAA 7A 3 EX 3	ORAA 8A 1 IM 2	ORAA 9A 3 DI 2	ORAA AA 3-6 ID 2-4	ORAA BA 3 EX 3	ORAB CA 1 IM 2	ORAB DA 3 DI 2	ORAB EA 3-6 ID 2-4	ORAB FA 3 EX 3
RTI †8 IH 1	LEAS 1B 2 ID 2-4	BMI 2B 3/1 RL 2	PSHD 2 IH 1	CALL 4B ‡7-10 ID 2-5	STAB 5B 2 DI 2	STAB 6B ‡2-4 ID 2-4	STAB 7B 3 EX 3	ADDA 8B 1 IM 2	ADDA 9B 3 DI 2	ADDA AB 3-6 ID 2-4	ADDA BB 3 EX 3	ADDB CB 1 IM 2	ADDB DB 3 DI 2	ADDB EB 3-6 ID 2-4	ADDB FB 3 EX 3
BSET 0C 4-6 ID 3-5	BSET 1C 4 EX 4	BGE 2C 3/1 RL 2	wavr 3C ‡+5 SP 1	BSET 4C 4 DI 3	STD 5C 2 DI 2	STD 6C ‡2-4 ID 2-4	STD 7C 3 EX 3	CPD 8C 2 IM 3	CPD 9C 3 DI 2	CPD AC 3-6 ID 2-4	CPD BC 3 EX 3	LDD CC 2 IM 3	LDD DC 3 DI 2	LDD EC 3-6 ID 2-4	LDD FC 3 EX 3
BCLR 0D 4-6 ID 3-5	BCLR 1D 4 EX 4	BLT 2D 3/1 RL 2	RTS 3D 5 IH 1	BCLR 4D 4 DI 3	STY 5D 2 DI 2	STY 6D ‡2-4 ID 2-4	STY 7D 3 EX 3	CPY 8D 2 IM 3	CPY 9D 3 DI 2	CPY AD 3-6 ID 2-4	CPY BD 3 EX 3	LDY CD 2 IM 3	LDY DD 3 DI 2	LDY ED 3-6 ID 2-4	LDY FD 3 EX 3
BRSET 0E ‡4-6 ID 4-6	BRSET 1E 5 EX 5	BGT 2E 3/1 RL 2	WAI 3E ‡†7 IH 1	BRSET 4E 4 DI 4	STX 5E 2 DI 2	STX 6E ‡2-4 ID 2-4	STX 7E 3 EX 3	CPX 8E 3 IM 3	CPX 9E 3 DI 2	CPX AE 3-6 ID 2-4	CPX BE 3 EX 3	LDX CE 2 IM 3	LDX DE 3 DI 2	LDX EE 3-6 ID 2-4	LDX FE 3 EX 3
BRCLR 0F ‡4-6 ID 4-6	BRCLR 1F 5 EX 5	BLE 2F 3/1 RL 2	SWI 3F 9 IH 1	RCLR 4 4	STS 5F 2 DI 2	STS 6F ‡2-4 ID 2-4	STS 7F 3 EX 3	CPS 8F 3 IM 3	CPS 9F 3 DI 2	CPS AF 3-6 ID 2-4	CPS BF 3 EX 3	LDS CF 3 IM 3	LDS DF 3 DI 2	LDS EF 3-6 ID 2-4	LDS FF 3 EX 3

Key to Table A-2

Opcode → 00 5 ← Number of HCS12 cycles (‡ indicates HC12 different)
Mnemonic → BGND ←
Address Mode → IH 1 ← Number of bytes

Table A-2. CPU12 Opcode Map (Sheet 2 of 2)

00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
4	12	4	10	10	10	10	10	10	10	10	10	10	10	10	10
MOVW	IDIV	LBRA	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
IM-ID 5	IH 2	RL 4	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
01	11	21	31	41	51	61	71	81	91	A1	B1	C1	D1	E1	F1
5	12	3	10	10	10	10	10	10	10	10	10	10	10	10	10
MOVW	FDIV	LBRN	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
EX-ID 5	IH 2	RL 4	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2
5	13	4/3	10	10	10	10	10	10	10	10	10	10	10	10	10
MOVW	EMACS	LBHI	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
ID-ID 4	SP 4	RL 4	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
03	13	23	33	43	53	63	73	83	93	A3	B3	C3	D3	E3	F3
5	3	4/3	10	10	10	10	10	10	10	10	10	10	10	10	10
MOVW	EMULS	LBLS	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
IM-EX 6	IH 2	RL 4	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
04	14	24	34	44	54	64	74	84	94	A4	B4	C4	D4	E4	F4
6	12	4/3	10	10	10	10	10	10	10	10	10	10	10	10	10
MOVW	EDIVS	LBCC	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
EX-EX 6	IH 2	RL 4	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5
5	12	4/3	10	10	10	10	10	10	10	10	10	10	10	10	10
MOVW	IDIVS	LBCS	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
[illegible]	[illegible] 2	RL 4	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
06	16	26	36	46	56	66	76	86	96	A6	B6	C6	D6	E6	F6
2	2	4/3	10	10	10	10	10	10	10	10	10	10	10	10	10
ABA	SBA	LBNE	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
IH 2	[illegible] 2	RL 4	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7
2	2	4/3	10	10	10	10	10	10	10	10	10	10	10	10	10
DAA	CBA	LBEQ	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
IH 2	IH 2	RL 4	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8
4	4-7	4/3	10	10	10	10	10	10	10	10	10	10	10	10	10
MOVB	MAXA	LBVC	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
IM-ID 4	ID 3-5	RL 4	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9
5	4-7	4/3	10	10	10	10	10	10	10	10	10	10	10	10	10
MOVB	MINA	LBVS	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
EX-ID 5	ID 3-5	RL 4	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
0A	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA
5	4-7	4/3	†3n	10	10	10	10	10	10	10	10	10	10	10	10
MOVB	EMAXD	LBPL	REV	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
ID-ID 4	ID 3-5	RL 4	SP 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
0B	1B	2B	3B	4B	5B	6B	7B	8B	9B	AB	BB	CB	DB	EB	FB
4	4-7	4/3	†5n/3n	10	10	10	10	10	10	10	10	10	10	10	10
MOVB	EMIND	LBMI	REVW	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
IM-EX 5	ID 3-5	RL 4	SP 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC
6	4-7	4/3	‡†7B	10	10	10	10	10	10	10	10	10	10	10	10
MOVB	MAXM	LBGE	WAV	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
EX-EX 6	ID 3-5	RL 4	SP 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	AD	BD	CD	DD	ED	FD
5	D4-7	4/3	‡6	10	10	10	10	10	10	10	10	10	10	10	10
MOVB	MINM	LBLT	TBL	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
ID-EX 5	ID 3-5	RL 4	ID 3	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
0E	1E	2E	3E	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE
2	4-7	4/3	‡8	10	10	10	10	10	10	10	10	10	10	10	10
TAB	EMAXM	LBGT	STOP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
IH 2	ID 3-5	RL 4	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2
0F	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	FF
2	4-7	4/3	10	10	10	10	10	10	10	10	10	10	10	10	10
TBA	EMINM	LBLE	ETBL	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP	TRAP
IH 2	ID 3-5	RL 4	ID 3	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2	IH 2

* The opcode $04 (on sheet 1 of 2) corresponds to one of the loop primitive instructions DBEQ, DBNE, IBEQ, IBNE, TBEQ, or TBNE.

† Refer to instruction summary for more information.

‡ Refer to instruction summary for different HC12 cycle count.

Page 2: When the CPU encounters a page 2 opcode ($18 on page 1 of the opcode map), it treats the next byte of object code as a page 2 instruction opcode.

Table A-3. Indexed Addressing Mode Postbyte Encoding (xb)

00 0,X 5b const	10 −16,X 5b const	20 1,+X pre-inc	30 1,X+ post-inc	40 0,Y 5b const	50 −16,Y 5b const	60 1,+Y pre-inc	70 1,Y+ post-inc	80 0,SP 5b const	90 −16,SP 5b const	A0 1,+SP pre-inc	B0 1,SP+ post-inc	C0 0,PC 5b const	D0 −16,PC 5b const	E0 n,X 9b const	F0 n,SP 9b const
01 1,X 5b const	11 −15,X 5b const	21 2,+X pre-inc	31 2,X+ post-inc	41 1,Y 5b const	51 −15,Y 5b const	61 2,+Y pre-inc	71 2,Y+ post-inc	81 1,SP 5b const	91 −15,SP 5b const	A1 2,+SP pre-inc	B1 2,SP+ post-inc	C1 1,PC 5b const	D1 −15,PC 5b const	E1 −n,X 9b const	F1 −n,SP 9b const
02 2,X 5b const	12 −14,X 5b const	22 3,+X pre-inc	32 3,X+ post-inc	42 2,Y 5b const	52 −14,Y 5b const	62 3,+Y pre-inc	72 3,Y+ post-inc	82 2,SP 5b const	92 −14,SP 5b const	A2 3,+SP pre-inc	B2 3,SP+ post-inc	C2 2,PC 5b const	D2 −14,PC 5b const	E2 n,X 16b const	F2 n,SP 16b const
03 3,X 5b const	13 −13,X 5b const	23 4,+X pre-inc	33 4,X+ post-inc	43 3,Y 5b const	53 −13,Y 5b const	63 4,+Y pre-inc	73 4,Y+ post-inc	83 3,SP 5b const	93 −13,SP 5b const	A3 4,+SP pre-inc	B3 4,SP+ post-inc	C3 3,PC 5b const	D3 −13,PC 5b const	E3 [n,X] 16b indr	F3 [n,SP] 16b indr
04 4,X 5b const	14 −12,X 5b const	24 5,+X pre-inc	34 5,X+ post-inc	44 4,Y 5b const	54 −12,Y 5b const	64 5,+Y pre-inc	74 5,Y+ post-inc	84 4,SP 5b const	94 −12,SP 5b const	A4 5,+SP pre-inc	B4 5,SP+ post-inc	C4 4,PC 5b const	D4 −12,PC 5b const	E4 A,X A offset	F4 A,SP A offset
05 5,X 5b const	15 −11,X 5b const	25 6,+X pre-inc	35 6,X+ post-inc	45 5,Y 5b const	55 −11,Y 5b const	65 6,+Y pre-inc	75 6,Y+ post-inc	85 5,SP 5b const	95 −11,SP 5b const	A5 6,+SP pre-inc	B5 6,SP+ post-inc	C5 5,PC 5b const	D5 −11,PC 5b const	E5 B,X B offset	F5 B,SP B offset
06 6,X 5b const	16 −10,X 5b const	26 7,+X pre-inc	36 7,X+ post-inc	46 6,Y 5b const	56 −10,Y 5b const	66 7,+Y pre-inc	76 7,Y+ post-inc	86 6,SP 5b const	96 −10,SP 5b const	A6 7,+SP pre-inc	B6 7,SP+ post-inc	C6 6,PC 5b const	D6 −10,PC 5b const	E6 D,X D offset	F6 D,SP D offset
07 7,X 5b const	17 −9,X 5b const	27 8,+X pre-inc	37 8,X+ post-inc	47 7,Y 5b const	57 −9,Y 5b const	67 8,+Y pre-inc	77 8,Y+ post-inc	87 7,SP 5b const	97 −9,SP 5b const	A7 8,+SP pre-inc	B7 8,SP+ post-inc	C7 7,PC 5b const	D7 −9,PC 5b const	E7 [D,X] D indirect	F7 [D,SP] D indirect
08 8,X 5b const	18 −8,X 5b const	28 8,–X pre-dec	38 8,X– post-dec	48 8,Y 5b const	58 −8,Y 5b const	68 8,–Y pre-dec	78 8,Y– post-dec	88 8,SP 5b const	98 −8,SP 5b const	A8 8,–SP pre-dec	B8 8,SP– post-dec	C8 8,PC 5b const	D8 −8,PC 5b const	E8 n,Y 9b const	F8 n,PC 9b const
09 9,X 5b const	19 −7,X 5b const	29 7,–X pre-dec	39 7,X– post-dec	49 9,Y 5b const	59 −7,Y 5b const	69 7,–Y pre-dec	79 7,Y– post-dec	89 9,SP 5b const	99 −7,SP 5b const	A9 7,–SP pre-dec	B9 7,SP– post-dec	C9 9,PC 5b const	D9 −7,PC 5b const	E9 −n,Y 9b const	F9 −n,PC 9b const
0A 10,X 5b const	1A −6,X 5b const	2A 6,–X pre-dec	3A 6,X– post-dec	4A 10,Y 5b const	5A −6,Y 5b const	6A 6,–Y pre-dec	7A 6,Y– post-dec	8A 10,SP 5b const	9A −6,SP 5b const	AA 6,–SP pre-dec	BA 6,SP– post-dec	CA 10,PC 5b const	DA −6,PC 5b const	EA n,Y 16b const	FA n,PC 16b const
0B 11,X 5b const	1B −5,X 5b const	2B 5,–X pre-dec	3B 5,X– post-dec	4B 11,Y 5b const	5B −5,Y 5b const	6B 5,–Y pre-dec	7B 5,Y– post-dec	8B 11,SP 5b const	9B −5,SP 5b const	AB 5,–SP pre-dec	BB 5,SP– post-dec	CB 11,PC 5b const	DB −5,PC 5b const	EB [n,Y] 16b indr	FB [n,PC] 16b indr
0C 12,X 5b const	1C −4,X 5b const	2C 4,–X pre-dec	3C 4,X– post-dec	4C 12,Y 5b const	5C −4,Y 5b const	6C 4,–Y pre-dec	7C 4,Y– post-dec	8C 12,SP 5b const	9C −4,SP 5b const	AC 4,–SP pre-dec	BC 4,SP– post-dec	CC 12,PC 5b const	DC −4,PC 5b const	EC A,Y A offset	FC A,PC A offset
0D 13,X 5b const	1D −3,X 5b const	2D 3,–X pre-dec	3D 3,X– post-dec	4D 13,Y 5b const	5D −3,Y 5b const	6D 3,–Y pre-dec	7D 3,Y– post-dec	8D 13,SP 5b const	9D −3,SP 5b const	AD 3,–SP pre-dec	BD 3,SP– post-dec	CD 13,PC 5b const	DD −3,PC 5b const	ED B,Y B offset	FD B,PC B offset
0E 14,X 5b const	1E −2,X 5b const	2E 2,–X pre-dec	3E 2,X– post-dec	4E 14,Y 5b const	5E −2,Y 5b const	6E 2,–Y pre-dec	7E 2,Y– post-dec	8E 14,SP 5b const	9E −2,SP 5b const	AE 2,–SP pre-dec	BE 2,SP– post-dec	CE 14,PC 5b const	DE −2,PC 5b const	EE D,Y D offset	FE D,PC D offset
0F 15,X 5b const	1F −1,X 5b const	2F 1,–X pre-dec	3F 1,X– post-dec	4F 15,Y 5b const	5F −1,Y 5b const	6F 1,–Y pre-dec	7F 1,Y– post-dec	8F 15,SP 5b const	9F −1,SP 5b const	AF 1,–SP pre-dec	BF 1,SP– post-dec	CF 15,PC 5b const	DF −1,PC 5b const	EF [D,Y] D indirect	FF [D,PC] D indirect

Key to Table A-3

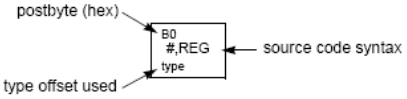


Table A-5. Transfer and Exchange Postbyte Encoding

TRANSFERS									
⇩LS	MS⇨	0	1	2	3	4	5	6	7
0		A ⇒ A	B ⇒ A	CCR ⇒ A	$TMP3_L$ ⇒ A	B ⇒ A	X_L ⇒ A	Y_L ⇒ A	SP_L ⇒ A
1		A ⇒ B	B ⇒ B	CCR ⇒ B	$TMP3_L$ ⇒ B	B ⇒ B	X_L ⇒ B	Y_L ⇒ B	SP_L ⇒ B
2		A ⇒ CCR	B ⇒ CCR	CCR ⇒ CCR	$TMP3_L$ ⇒ CCR	B ⇒ CCR	X_L ⇒ CCR	Y_L ⇒ CCR	SP_L ⇒ CCR
3		sex:A ⇒ TMP2	sex:B ⇒ TMP2	sex:CCR ⇒ TMP2	TMP3 ⇒ TMP2	D ⇒ TMP2	X ⇒ TMP2	Y ⇒ TMP2	SP ⇒ TMP2
4		sex:A ⇒ D SEX A,D	sex:B ⇒ D SEX B,D	sex:CCR ⇒ D SEX CCR,D	TMP3 ⇒ D	D ⇒ D	X ⇒ D	Y ⇒ D	SP ⇒ D
5		sex:A ⇒ X SEX A,X	sex:B ⇒ X SEX B,X	sex:CCR ⇒ X SEX CCR,X	TMP3 ⇒ X	D ⇒ X	X ⇒ X	Y ⇒ X	SP ⇒ X
6		sex:A ⇒ Y SEX A,Y	sex:B ⇒ Y SEX B,Y	sex:CCR ⇒ Y SEX CCR,Y	TMP3 ⇒ Y	D ⇒ Y	X ⇒ Y	Y ⇒ Y	SP ⇒ Y
7		sex:A ⇒ SP SEX A,SP	sex:B ⇒ SP SEX B,SP	sex:CCR ⇒ SP SEX CCR,SP	TMP3 ⇒ SP	D ⇒ SP	X ⇒ SP	Y ⇒ SP	SP ⇒ SP
EXCHANGES									
⇩LS	MS⇨	8	9	A	B	C	D	E	F
0		A ⇔ A	B ⇔ A	CCR ⇔ A	$TMP3_L$ ⇒ A $00:A ⇒ TMP3	B ⇒ A A ⇒ B	X_L ⇒ A $00:A ⇒ X	Y_L ⇒ A $00:A ⇒ Y	SP_L ⇒ A $00:A ⇒ SP
1		A ⇔ B	B ⇔ B	CCR ⇔ B	$TMP3_L$ ⇒ B $FF:B ⇒ TMP3	B ⇒ B $FF ⇒ A	X_L ⇒ B $FF:B ⇒ X	Y_L ⇒ B $FF:B ⇒ Y	SP_L ⇒ B $FF:B ⇒ SP
2		A ⇔ CCR	B ⇔ CCR	CCR ⇔ CCR	$TMP3_L$ ⇒ CCR $FF:CCR ⇒ TMP3	B ⇒ CCR $FF:CCR ⇒ D	X_L ⇒ CCR $FF:CCR ⇒ X	Y_L ⇒ CCR $FF:CCR ⇒ Y	SP_L ⇒ CCR $FF:CCR ⇒ SP
3		$00:A ⇒ TMP2 $TMP2_L$ ⇒ A	$00:B ⇒ TMP2 $TMP2_L$ ⇒ B	$00:CCR ⇒ TMP2 $TMP2_L$ ⇒ CCR	TMP3 ⇔ TMP2	D ⇔ TMP2	X ⇔ TMP2	Y ⇔ TMP2	SP ⇔ TMP2
4		$00:A ⇒ D	$00:B ⇒ D	$00:CCR ⇒ D B ⇒ CCR	TMP3 ⇔ D	D ⇔ D	X ⇔ D	Y ⇔ D	SP ⇔ D
5		$00:A ⇒ X X_L ⇒ A	$00:B ⇒ X X_L ⇒ B	$00:CCR ⇒ X X_L ⇒ CCR	TMP3 ⇔ X	D ⇔ X	X ⇔ X	Y ⇔ X	SP ⇔ X
6		$00:A ⇒ Y Y_L ⇒ A	$00:B ⇒ Y Y_L ⇒ B	$00:CCR ⇒ Y Y_L ⇒ CCR	TMP3 ⇔ Y	D ⇔ Y	X ⇔ Y	Y ⇔ Y	SP ⇔ Y
7		$00:A ⇒ SP SP_L ⇒ A	$00:B ⇒ SP SP_L ⇒ B	$00:CCR ⇒ SP SP_L ⇒ CCR	TMP3 ⇔ SP	D ⇔ SP	X ⇔ SP	Y ⇔ SP	SP ⇔ SP

TMP2 and TMP3 registers are for factory use only.

Table A-6. Loop Primitive Postbyte Encoding (lb)

00 A DBEQ (+)	10 A DBEQ (–)	20 A DBNE (+)	30 A DBNE (–)	40 A TBEQ (+)	50 A TBEQ (–)	60 A TBNE (+)	70 A TBNE (–)	80 A IBEQ (+)	90 A IBEQ (–)	A0 A IBNE (+)	B0 A IBNE (–)
01 B DBEQ (+)	11 B DBEQ (–)	21 B DBNE (+)	31 B DBNE (–)	41 B TBEQ (+)	51 B TBEQ (–)	61 B TBNE (+)	71 B TBNE (–)	81 B IBEQ (+)	91 B IBEQ (–)	A1 B IBNE (+)	B1 B IBNE (–)
02 —	12 —	22 —	32 —	42 —	52 —	62 —	72 —	82 —	92 —	A2 —	B2 —
03 —	13 —	23 —	33 —	43 —	53 —	63 —	73 —	83 —	93 —	A3 —	B3 —
04 D DBEQ (+)	14 D DBEQ (–)	24 D DBNE (+)	34 D DBNE (–)	44 D TBEQ (+)	54 D TBEQ (–)	64 D TBNE (+)	74 D TBNE (–)	84 D IBEQ (+)	94 D IBEQ (–)	A4 D IBNE (+)	B4 D IBNE (–)
05 X DBEQ (+)	15 X DBEQ (–)	25 X DBNE (+)	35 X DBNE (–)	45 X TBEQ (+)	55 X TBEQ (–)	65 X TBNE (+)	75 X TBNE (–)	85 X IBEQ (+)	95 X IBEQ (–)	A5 X IBNE (+)	B5 X IBNE (–)
06 Y DBEQ (+)	16 Y DBEQ (–)	26 Y DBNE (+)	36 Y DBNE (–)	46 Y TBEQ (+)	56 Y TBEQ (–)	66 Y TBNE (+)	76 Y TBNE (–)	86 Y IBEQ (+)	96 Y IBEQ (–)	A6 Y IBNE (+)	B6 Y IBNE (–)
07 SP DBEQ (+)	17 SP DBEQ (–)	27 SP DBNE (+)	37 SP DBNE (–)	47 SP TBEQ (+)	57 SP TBEQ (–)	67 SP TBNE (+)	77 SP TBNE (–)	87 SP IBEQ (+)	97 SP IBEQ (–)	A7 SP IBNE (+)	B7 SP IBNE (–)

Key to Table A-6

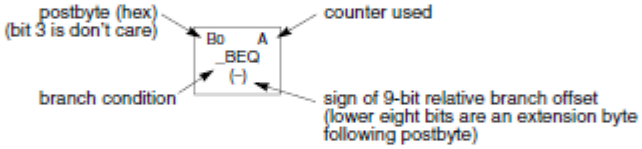


Table A-7. Branch/Complementary Branch

Branch				Complementary Branch			
Test	**Mnemonic**	**Opcode**	**Boolean**	**Test**	**Mnemonic**	**Opcode**	**Comment**
r>m	BGT	2E	Z + (N ⊕ V) = 0	r≤m	BLE	2F	Signed
r≥m	BGE	2C	N ⊕ V = 0	r<m	BLT	2D	Signed
r=m	BEQ	27	Z = 1	r≠m	BNE	26	Signed
r≤m	BLE	2F	Z + (N ⊕ V) = 1	r>m	BGT	2E	Signed
r<m	BLT	2D	N ⊕ V = 1	r≥m	BGE	2C	Signed
r>m	BHI	22	C + Z = 0	r≤m	BLS	23	Unsigned
r≥m	BHS/BCC	24	C = 0	r<m	BLO/BCS	25	Unsigned
r=m	BEQ	27	Z = 1	r≠m	BNE	26	Unsigned
r≤m	BLS	23	C + Z = 1	r>m	BHI	22	Unsigned
r<m	BLO/BCS	25	C = 1	r≥m	BHS/BCC	24	Unsigned
Carry	BCS	25	C = 1	No Carry	BCC	24	Simple
Negative	BMI	2B	N = 1	Plus	BPL	2A	Simple
Overflow	BVS	29	V = 1	No Overflow	BVC	28	Simple
r=0	BEQ	27	Z = 1	r≠0	BNE	26	Simple
Always	BRA	20	—	Never	BRN	21	Unconditional

For 16-bit offset long branches precede opcode with a $18 page prebyte.

Binary, Hex and Decimal Numbers (4-bit representation)

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

What does a number represent?

Binary numbers are a code, and represent what the programmer intends for the code.

0x72 Some possible meanings:

'r' (ASCII)
INC MEM (hh ll) (HC12 instruction)
2.26V (Input from A/D converter)
114_{10} (Unsigned number)
$+114_{10}$ (Signed number)
Set temperature in room to 69 °F
Set cruise control speed to 120 mph

Binary to Unsigned Decimal:

Convert Binary to Unsigned Decimal

1111011_{2}

$1 \times 2^{6} + 1 \times 2^{5} + 1 \times 2^{4} + 1 \times 2^{3} + 0 \times 2^{2} + 1 \times 2^{1} + 1 \times 2^{0}$

$1 \times 64 + 1 \times 32 + 1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$

123_{10}

Hex to Unsigned Decimal

Convert Hex to Unsigned Decimal

$82D6_{16}$

$8 \times 16^3 + 2 \times 16^2 + 13 \times 16^1 + 6 \times 16^0$

$8 \times 4096 + 2 \times 256 + 13 \times 16 + 6 \times 1$

33494_{10}

Unsigned Decimal to Hex

Convert Unsigned Decimal to Hex

Division	Q	R	
		Decimal	Hex
721/16	45	1	1
45/16	2	13	D
2/16	0	2	2

$721_{10} = 2D1_{16}$

Signed Number Representation in 2's Complement Form:

If the most significant bit (MSB) is 0 (most significant hex digit 0–7), then the number is positive.
Get decimal equivalent by converting number to decimal, and use the + sign.

Example for 8–bit number:

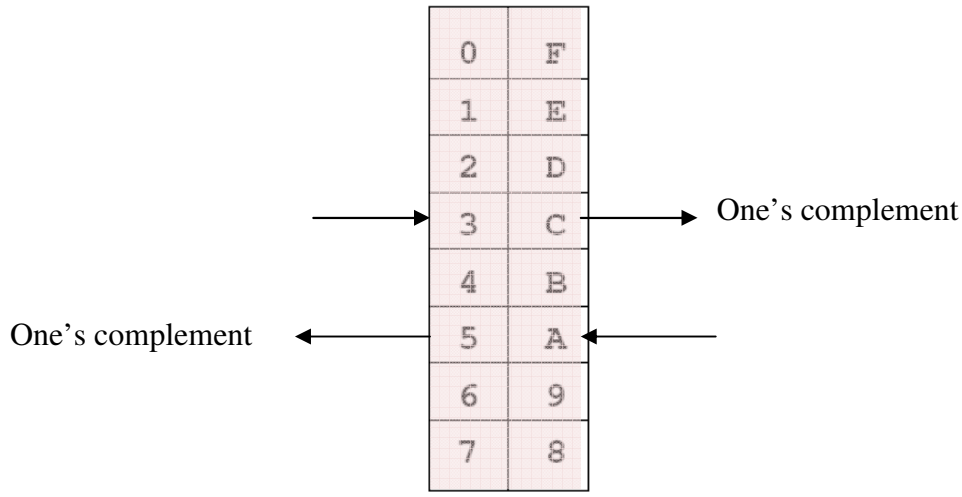
3A $_{16}$ –> + (3 x 16^1 + 10 x 16^0) $_{10}$
+ (3 x 16 + 10 x 1) $_{10}$
+ 58 $_{10}$

If the most significant bit is 1 (most significant hex digit 8–F), then the number is negative.
Get decimal equivalent by taking 2's complement of number, converting to decimal, and using – sign.

Example for 8–bit number:

A3$_{16}$ –> - (5D) $_{16}$
- (5 x 16^1 + 13 x 16^0) $_{10}$
- (5 x 16 + 13 x 1) $_{10}$
- 93 $_{10}$

One's complement table makes it simple to finding 2's complements



To take two's complement, add one to one's complement.

Take two's complement of **D0C3**:

2F3C + 1 = **2F3D**

Addition and Subtraction of Binary and Hexadecimal Numbers

Setting the C (Carry), V (Overflow), N (Negative) and Z (Zero) bits

How the C, V, N and Z bits of the CCR are changed

N bit is set if result of operation is negative (MSB = 1)

Z bit is set if result of operation is zero (All bits = 0)

V bit is set if operation produced an overflow

C bit is set if operation produced a carry (borrow on subtraction)

Note: Not all instructions change these bits of the CCR

Addition of Hexadecimal Numbers

ADDITION:

C bit set when result does not fit in word

V bit set when P + P = N or N + N = P

N bit set when MSB of result is 1

Z bit set when result is 0

7A	2A	AC	AC
+52	+52	+8A	+72
---	---	---	---
CC	7C	36	1E
C: 0	C: 0	C: 1	C: 1
V: 1	V: 0	V: 1	V: 0
N: 1	N: 0	N: 0	N: 1
Z: 0	Z: 0	Z: 0	Z: 0

Subtraction of Hexadecimal Numbers

SUBTRACTION:

C bit set on borrow (when the magnitude of the subtrahend is greater than the minuend

V bit set when N - P = P or P - N = N

N bit set when MSB is 1

Z bit set when result is 0

7A	8A	5C	2C
-5C	-5C	-8A	-72
---	---	---	---
1E	2E	D2	BA
C: 0	C: 0	C: 1	C: 1
V: 0	V: 1	V: 1	V: 0
N: 0	N: 0	N: 1	N: 1
Z: 0	Z: 0	Z: 0	Z: 0