Instruction Set of 68HC11

- Transfer
  - Register to register
- Load
  - Memory to register
- Store
  - Register to memory
- Math
  - Addition
  - Increment
  - Decrement
  - Subtract
  - Negate
    - 2's complement
  - Complement
    - 1's complement

1

Update: Direct Mode

- Use * to ensure direct addressing mode…
  - Explicit addresses
    - staa *0xB0
    - stab *0x00B2 ; still direct (fits in 8 bits)
    - std *0x0110 ; assembler error
  - Or, use labels
    - memory0 = 0xB0
    - stx *(memory0+4)
- Some linkers automatically “condense” EXT to DIR when possible.
  - The ld we are using does **not** do this.

2

Transfer

- TAB – transfer A to B
- TBA – transfer B to A
- TAP – transfer A to CCR
- TPA – transfer CCR to A
- XGDX – exchange D with X
- XGDY – exchange D with Y
  - D is A:B

3

Transfer – stack

- PSHA – push A
- PSHB – push B
- PSHX – push X
- PSHY – push Y
- PULA – pull A
- PULB – pull B
- PULX – pull X
- PULY – pull Y

4

Transfer – clearing

- CLR – clear a memory byte (to zero)
- CLRA – clear A
- CLRB – clear B

5

Loading

- LDAA
  - LDAA #xx – loads a constant
  - LDAA xx – load memory location 00xx
  - LDAA xxxx – load memory location xxxx
  - LDAA xx,X – load memory location X + xx
  - LDAA xx,Y – load memory location Y + xx

6

Loading

- LDAB – load accumulator B
- LDD – load D (“big endian”)
- LDS – load stack pointer
- LDX – load X
- LDY – load Y

- All support the same addressing modes as LDAA

7

Storing

- STAA – store A
- STAB – store B
- STD – store D
- STS – store SP (stack pointer)
- STX – store X
- STY – store Y

- Same modes as LD... (except for immediate, which wouldn’t make sense)

8

Math – addition

- ADDA – add memory to A
  - ADDA #xx – Add a constant
    - “Memory” is a constant
  - ADDA xx – Add contents of 00xx
  - ADDA xxxx – Add contents of xxxx
  - ADDA xx,X – Add contents of X + xx
  - ADDA xx,Y – Add contents of Y + xx

9

Math – addition

- ADDB – add to B (same modes)
- ADDD – add to D (same modes)
- ADCA – add to A with carry
- ADCB – add to B with carry
- no “ADCD”!
- ABA – add B to A
- ABX – add B to X
- ABY – add B to Y (no A version)

10

Math – increment

- “Counter”-style: Z updated, C not
  - NV also updated
    - INC – increment memory (ext, IX, IY)
    - INCA – increment A
    - INCB – increment B
  - INX – increment X
  - INY – increment Y
- INS – increment stack pointer
  - No CCR update

11

Math – subtract

- SUBA – subtract memory from A
- SUBB – subtract memory from B
- SUBD – subtract memory from D
- SBCA – subtract with carry from A (A – memory – C)
- SBCB – subtract with carry from B
- SBA – subtract B from A

12

Math – decrement

- DEC – decrement memory
- DECA – decrement A
- DECB – decrement B
- DEX – decrement X
- DEY – decrement Y
- DES – decrement stack pointer

13

Math – negate

- NEG – negate (2's complement) of memory
- NEGA – negate A
- NEGB – negate B
- Remember a 2's complement flips the bits and adds one

14

Math – (1's) complement

- COM – complement a memory byte (flip bits – ext, IX, IY)
- COMA – complement A
- COMB – complement B
- Remember – a complement only flips the bits

15

Sample Problem

```
.section .rodata
str:    .asciz  “Ruby3”
; 0x52 0x75 0x62 0x79
; 0x33 0x00
```

```
.section .text
.global _start
_start:
        ldaa    #0xA5
        ldab    #0x05
        ldx     #str+2
        adda    #0x03
        dex
        addb    3,x
        ldx     #str+4
        comb
        tba
        clrb
        xgdx
```

What do accumulators A and B, index register IX, and CCR bits N, Z, V, and C contain after each instruction?

16