HC12 Assembly Language Programming

Programming Model

Addressing Modes

Assembler Directives

HC12 Instructions

Flow Charts

Assembler Directives

- In order to write an assembly language program it is necessary to use *assembler directives*.
- These are not instructions which the HC12 executes but are directives to the assembler program about such things as where to put code and data into memory.
- All of the assembler directives can be found in Pages 46 through 49 of the manual for the evaluation version of the Cosmic Compiler. A PDF version of this manual can be found on the EE 308 home page.
- We will use only a few of these directives. (Note: In the following table, [] means an optional argument.) Here are the ones we will need:

Directive Name	**Description**	**Example**
equ	Give a value to a symbol	`len:   equ     100`
org	Set starting value of location counter where code or data will go	`       org     $0800`
section	Define a new program section For example, code or data	`CODE:  section .text`
dc[.size]	Allocate and initialize storage for variables. Size can be `b` (byte), `w` (two bytes) or `l` (4 bytes) If no size is specified, `b` is uses	`var:   dc.b    2,18`
ds[.size]	Allocate specified number of storage spaces. `size` is the same as for `dc` directive	`table: ds.w    10`

Using labels in assembly programs

A **label** is defined by a name followed by a colon as the first thing on a line. When the label is referred to in the program, it has a numerical value of the location counter when the label was defined.

Here is a code fragment using labels and the assembler directives `dc` and `ds`:

```
DATA:    section .data    ;The stuff which follows is data
         org     $0900
table1:  dc.b    $23,$17,$f2,$a3,$56
table2:  ds.b    5
var:     dc.w    $43af
```

Here is the listing from the assembler:

```
   9                        DATA:    section .data    ;The stuff w
  10  0900                           org     $0900
  11  0900 2317f2a356       table1:  dc.b    $23,$17,$f2,$a3,$56
  12  0905 0000000000       table2:  ds.b    5
  13  090a 43af             var:     dc.w    $43af
```

And here is the `map` file:

```
Map of demo.h12 from link file demo.lkf - Thu Jan 25 09:56:12 2

table1   00000900
table2   00000905
var      0000090a
```

Note that, `table1` is a name with the value of `$0900`, the value of the location counter defined in the `org` directive. Five bytes of data are defined by the `dc.b` directive, so the location counter is increased from `$0900` to `$0905`. `table2` is a name with the value of `$0905`. Five bytes of data are set aside for `table2` by the `ds.b 5` directive. The Cosmic assembler initialized these five bytes of data to all zeros.

HC12 Assembly Language Programming

Programming Model

Addressing Modes

Assembler Directives

HC12 Instructions

Flow Charts

1. Data Transfer and Manipulation Instructions — instructions which move and manipulate data (**CPU12 Reference Manual**, Sections 5.2, 5.3, and 5.4).

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

     ```
     LDAA  $0900  ; Copy contents of addr $0900 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 $0900 $09A0    ; Copy byte at $0900 to $09A0
     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 (**CPU12 Reference Manual**, Sections 5.5, 5.6, 5.7 5.11).

   ```
   ABA          ; Add B to A; results in A
   SUBD $09A1   ; Subtract contents of $09A1 from D
   INX          ; Increment X by 1
   MUL          ; Multiply A by B; results in D
   ```

3. Logic and Bit Instructions — perform logical operations (**CPU12 Reference Manual**, Sections 5.9, 5.10, 5.12, 5.13).

   - Logic Instructions

     ```
     ANDA $0900 ; Logical AND of A with contents of $0900
     NEG  -2,X  ; Negate (2’ comp) contents of address (X-2)
     LSLA       ; Logical shift left A by 1
     ```

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

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

4. Data 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.) (**CPU12 Reference Manaul**, Section 5.7).

   ```
   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) (**CPU12 Reference Manual**, Sections 5.18, 5.19, 5.20).

   ```
   JMP  l1    ; Start executing code at address label l1
   BEQ  l2    ; If Z bit zero, 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 set, goto l4
   ```

6. Function Call and Interrupt Instructions — initiate or terminate a subroutine; initiate or terminate and interrupt call (**CPU12 Reference Manual**, Sections 5.20, 5.21).

   - Subroutine instructions:

     ```
     JSR sub1  ; Jump to subroutine sub1
     RTS       ; Return from subroutine
     ```

   - Interrupt instructions

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

7. Stacking Instructions — push data onto and pull data off of stack (**CPU12 Reference Manual**, Section 5.23).

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

8. Stop and Wait Instructions — put HC12 into low power mode (**CPU12 Reference Manual**, Section 5.26).

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

9. Instructions we won't discuss or use — BCD arithmetic, fuzzy logic, minimum and maximum, multiply-accumulate, table interpolation (**CPU12 Reference Manual**, Sections 5.6, 5.14, 5.15, 5.16, 5.17).

Branch if A > B

Is 0xFF > 0x00?

If unsigned, 0xFF = 255 and 0x00 = 0,

so 0xFF > 0x00

If signed, 0xFF = −1 and 0x00 = 0,

so 0xFF < 0x00

```
Using unsigned numbers:  BHI   (checks C bit of CCR)

Using signed numbers:    BGT   (checks V bit of CCR)
```

For unsigned numbers, use branch instructions which check C bit

For signed numbers, use branch instructions which check V bit

Will the branch be taken?

```
LDAA  #$FF          LDAA #$FF
CMPA  #$0           CMPA #$0
BLO   label1        BLT  label2
```

```
LDX  #$C000         LDX  #$C000
CMPX #$8000         CMPS #$8000
BGT  label3         BHI  label4
```