

**EE 371 LAB 1 - F05**  
**MC68HCS12 Programming - Introduction**  
*Why is it your feet smell and your nose runs?*

References: EE371 texts, *Hitchhiker's Guide to CodeWarrior*  
Schedule: 9/7, 8 Do the lab  
9/14, 15 Programs demonstrated

1. Pre-Lab: Read
2. Follow steps 1-18 in the *Hitchhiker's Guide to CodeWarrior* to enter and make the following program.
  - a. Type the following code into main.asm (step 10) after the comment ; Your program code goes here.

```
;*****  
; Your program code goes here  
; Initialize I/O  
main_loop:  
; DO  
    nop            ; No operation  
    ldx    #main_loop ; Initialize pointer  
    ldab   #$02      ; Load Reg B  
  
loop:  
    ldaa  0,x       ; Load Reg A  
    staa  VarData   ; Save it  
    ldaa  2,x       ; Load Reg A again  
    staa  VarData+1 ; Save it again  
    inx           ; Increment pointer  
    decb           ; Decrement counter  
    bne   loop      ; If counter not zero loop  
    nop           ; Do nothing again  
; FOREVER  
    bra   main_loop  
  
MyConst: SECTION  
; Place constant data here  
;*****  
MyData: SECTION  
; Place variable data here  
VarData: DS.B    2 ; Two bytes of storage
```

After you have successfully compiled (assembled!) and made the program, go to step 16 and 17 to use the True-Time Simulator and P&E ICD Target.

After the simulator has been launched, have a look at the display.

The **Source** window shows your source program.

The **Assembly** window shows the program as it looks in memory. **Right Click** in the **Assembly** window and check **Display Code** to make your display show the code in the memory locations \$C000 - \$C025. **Right Click** again and check **Display Symbolic** to show symbolic instead of decimal addresses.

The **Registers** window shows the current status/contents of all registers.

The **Data** window will show the current value of variable data in your program. Click on the +VarData to show the two variable bytes. **Right Click** on VarData and select **Format... Hex**.

The **Procedure** window shows what procedure you are currently in and the **Command** window allows you to enter simulator/debugger commands. We will see more of these windows in future labs.

The **Memory** window allows the display of memory contents. **Right Click** in the **Memory** window and select **Address...** and in the **Display Address** window enter C000.

In the Source window position the cursor pointer over the **nop** instruction and **Right Click** and select **Set Breakpoint**.

Click the **Green Run Arrow** or click **Run > Start/Continue**. The program should run to your breakpoint and stop. (Note: A breakpoint stops the program before it executes that statement.)

Name \_\_\_\_\_

Partner \_\_\_\_\_

Meeting Day \_\_\_\_\_

HR \_\_\_\_\_

Demo:

(3) \_\_\_\_\_

(8) \_\_\_\_\_

Extra credit (9) \_\_\_\_\_

Now, step through your program one assembly language statement at a time by clicking on the **single-step** button or pressing **F11** until you reach the nop instruction at the beginning of the program you entered. (We will see what the LDS instruction is all about later.)



At each step inspect the Register window to see what is happening to the registers. Try to predict what will happen at each step BEFORE you click on the step button.

3. Explain to your lab instructor what you see on the screen. In particular, explain what is happening to each register and to the condition code register.
4. Close the simulator to return to the CodeWarrior project manager.
5. Create a new project by entering the following program. To save some typing, you can get the source file from the “N” drive. (If you haven’t done so all ready, you can map drive N to ohm\ecourses\$.) The source file is N:\EE371\lab01\_f05\lab\_01\_f05.asm. In the project manager, **File > Open**, browse to the file, and then after opening the file do **File > Save As ...** and be sure to position to your project folder in the **Sources** folder. You can also rename the file if you wish. Then **Project > Add lab\_01\_f05.asm to Project...**

```
; EE371, Lab 1, Fall 2005
; Blink LED1 and LED2 on the CSM-12C32 board
;*****
; Define the entry point for the main program
        XDEF    Entry, main
        XREF    __SEG_END_SSTACK ; Note double underbar
;*****
; Include files
;*****
; Register definitions
BASE:   EQU    0           ; Base address for registers
PORTA:  EQU    BASE+0     ; Port A
DDRA:   EQU    BASE+2     ; Data Dir Reg A
PORTB:  EQU    BASE+1     ; Port B
DDRB:   EQU    BASE+3     ; Data Dir Reg B
LED1:   EQU    %00000001 ; LED1 on Port A-0
LED2:   EQU    %00010000 ; LED2 on Port B-4
;*****
; Constants definitions
DELAY1: EQU    $ffff     ; Inner loop counter
DELAY2: EQU    $04       ; Outer loop counter
;*****
; Constants definitions
;*****
; Code Section
MyCode: SECTION
Entry:
main:
;*****
; Initialize stack pointer register
        lds    #__SEG_END_SSTACK
;*****
; Your program code goes here
; Initialize I/O
        bset  DDRA,LED1 ; Make Port A-0 output
        bset  DDRB,LED2 ; Make Port B-4 output
        ldaa #LED1     ; Initial LED display
        ldab #~LED2    ; ~ = 1's complement
main_loop:
; DO
;   Display the current pattern on the LEDs
```

```

        staa  PORTA
        stab  PORTB
;   Delay some time
        jsr   delay_sub  ; Use a subroutine
;   Complement the display values
        coma   ; Complement A register
        comb   ; Complement B register
; FOREVER
        bra   main_loop
;*****
; Simple Delay Subroutine
; Registers modified: CCR
;*****
delay_sub:
        pshx           ; Save the registers
        pshy
        ldy   #DELAY2   ; Initialize outer loop counter
outer:
;   WHILE the Y register is not zero
;   Decrement the outer loop counter
        dey
        beq   all_done  ; Branch when Y equal to zero
;   DO the inner loop
        ldx   #DELAY1   ; Initialize inner loop counter
inner:
;   WHILE the X register is not zero
;   DO Decrement the inner loop counter
        dex
        bne   inner     ; Branch when X NOT equal to zero
; ENDWHILE X not zero
        bra   outer
;   ENDWHILE Y not zero
all_done:
        puly           ; Restore the registers
        pulx
        rts
;*****
MyConst:SECTION
; Place constant data here
;*****
MyData: SECTION
; Place variable data here

```

6. Run the program. You should see the two LEDs on the CSM-12C32 microcontroller board flashing.
7. Step through the program to see how it works.
8. Modify the program so that it flashes at about 1/2 the rate.
9. Extra Credit: Modify the program so that it flashes LED1 twice while LED2 is off and then LED2 twice while LED1 is off and repeats.
10. Grading: Program demo s 5 points each, extra credit 5 points