Lab 2 – Part 1
Assembly Language Programming and MC9S12 Ports

In this sequence of three labs, you will learn how to write simple assembly language programs for the MC9S12 microcontroller, and how to use general purpose I/O (input/output) ports.

Introduction and Objectives

This laboratory will give you more experience with the tools we will use this semester: the Dragon12Plus evaluation board (EVB), the DBug12 monitor, and CodeWarrior. Be sure to read through the entire lab and do the prelab section before coming to lab

Program 1 First demo program.

```
prog:   equ     $2000           ; Starting address for program
data:   equ      $1000          ; Starting address for data

        org     prog            ; Set initial program counter value
        ldy     #2345           ; Immediate (IMM) addressing mode
        ldab     #123
        aby                     ; Inherent (INH) addressing mode
        sty     result          ; Extend (EXT) addressing mode
        swi

        org     data            ; Put data starting at this location
result: ds.w    1               ; Reserve one word (two bytes) for results
```

Program 2 Second demo program.

```
; MC9S12 program to add a table of 8-bit numbers, and save the 16-bit result

prog:   equ     $2000           ; Starting address for program
data:   equ     $1000           ; Starting address for data
count:  equ     10              ; 10 elements in the table

        org     prog            ; Set initial program counter value
        ldaa    #count          ; ACCA keeps count of numbers left in table
        ldx     #table          ; X points to table of data
        ldy     #0              ; Y holds sum; initialize to 0
repeat: ldab    1,X+            ; get data from table into B; X points to next element
        aby                     ; Compute 16-bit sum
        deca                    ; Decrement counter
        bne     repeat          ; If not done, continue with next element
        sty     result          ; Save sum
        swi

        org     data            ; Put data starting at this location

; Initialize data in table

table:  dc.b    $44,$AB,$74,$61,$C2,$54,$61,$62,$F2,$13
result : ds.w   1               ; Reserve one 16-bit word for result
```

Program 3 Third demo program.

```
        ldy     #50000
loop1:  ldaa    #250
loop2:  dbne    a,loop2
        dbne    y,loop1
        swi
```

1. Prelab

1.1 Questions to Answer Before Lab

1. Consider Program 1

   (a) Hand-assemble Program 1; i.e., determine the op-codes the MC9S12 will use to execute this program.

   (b) How many cycles will this take on the MC9S12? (Do not consider the swi instruction.)

   (c) How long in time will this take? (Note: the MC912 executes 24 million cycles per second.)

   (d) What will be the state of the **N**, **Z**, **V** and **C** bits after each instruction has been executed? (Ignore the swi instruction.)

   (e) What will be in address 0x1000 and 0x1001 after the program executed?

2. Consider Program 2.

   (a) Hand-assemble Program 2. Indicate the addressing mode for each of the instructions.

3. Consider Program 3.

   (a) Hand-assemble Program 3.

   (b) How many cycles will this program take on the MC9S12?

   (b) How long will it take to execute this program?

2. The Lab

2.1 Answer the Following During Lab

Be sure to answer these questions in you lab book.

1. Consider Program 1

   (a) Use a text editor to enter this program. Assemble the program using CodeWarrior. (Be sure to set CodeWarrior to generate a listing file.) Use a text editor to remove the first line (which starts with "S0") from the s19 file. Look at the lst and s19 files. You should be able to relate the opcodes from the prelab to the data in the s19 file. Verify that they agree.

   (b) Load the program onto your Dragon12 Plus board. Trace through the program. Verify that the **Z**, **N**, **V** and **C** bits are what you expect after each instruction.

   (c) Look at the contents of addresses 0x1000 and 0x1002. Do the values agree with your answers from the prelab?

2. Consider Program 2, which computes the sum of a table of 8-bit data.

   (a) Use a text editor to enter this program, and use CodeWarrior to assemble it into an s19 file. Use a text editor to remove the first line (which starts with "S0") from the s19 file.

   (b) Load the program into your MC9S12. Use **MD** to verify that the data is in the table at address 0x1000. Use **ASM** to verify that the program is loaded into memory at address $2000. Run the program, and verify that the sum is stored in the memory location for result.

   (c) Use the **BF** (Block Fill) command of DBug-12 to change the values in addresses 0x1000 through 0x2FFF to 0xFF. Reload the s19 file.

   (d) Set a breakpoint at the label repeat. (Look at the .lst file to find the address of the label.)

   (e) Execute the program again. The program should stop the first time it reaches the repeat label, with 0x0A in **ACCB**, 0x1000 in X, and 0x0000 in Y.

   (f) Continue running the program. It should stop each time it gets to the repeat label, B should be decremented by one, X should be incremented by one, and Y should contain the updated sum. Use the **RD** and **MD** commands of DBug-12 to verify this.

3. Consider the code fragment of Program 3.

(a) Use a text editor to enter the code into a program. You will have to add an **org** statement and other assembler directives to make the program so you can assemble it with CodeWarrior.

(b) Assemble the program and run it on the MC9S12. How long does it take to run? This time should match your answer which you got in the prelab.