Lab 2 – Part 1
Assembly Language Programming and 9S12 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:   equu    $2000       ; Starting address from program
data:   equ      $1000      ; Starting address for data

        org     prog        ; Set initial program counter value
        ldx     #1234       ; Immediate (IMM) addressing mode
        ldab     #235
        abx                 ; Inherent (INH) addressing mode
        stx     result      ; Extend (EXT) addressing mode
        swi

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

Program 2 Second demo program.

```
; MC9S12 program to copy a table of data from one location to another
; The copied data is the negative of the original data

prog:   equu    $2000       ; Starting address from program
data:   equ     $1000       ; Starting address for data
count:  equ     8           ; 8 elements in the table

        org     prog        ; Set initial program counter value
        ldab    #count      ; ACCB keeps count of number to transfer
        ldx     #table_1    ; X points at table_1
        ldy     #table_2    ; Y points at table_2
repeat: ldaa    1,X+        ; get data from table_1, X points to next element
        nega
        staa    1,Y+        ; save into table_2, Y points to next element
        decb                ; Decrement counter
        bne     repeat      ; If not done, continue with next element
        swi

        org     data        ; Put data starting at this location

; Initialize data in table

table_1: dc.b   $44,$61,$74,$61,$20,$54,$61,$62
table_2: ds.b   count       ; Reserve count bytes of memory for results
```

Program 3 Third demo program.

```
        ldy     #5000
loop1:  ldx     #5000
loop2:  dbne    x,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) How many cycles will it take to execute the program on the MC9S12?

   (b) How long in time will this take?

3. Consider Program 3.

   (a) 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) Assemble the program using CodeWarrior. 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 moves data from one table into another.

(a) Use the text editor to enter this program, and assemble it into an s19 file.
(b) Load the program into your MC9S12. Use **MD** to verify that the data is in the table at address 0x1000. Run the program, and verify that the table has been copied into table 2.

(c) Use the Block Fill option of DBug-12 to change the values in addresses 0x1000 through 0x1FFF 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 0x08 in **ACCB**, and 0x1000 in X.

(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 there should be a new entry in table 2.
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 org statements and other assembler directives to make the program work.

(b) Assemble the program and run it on the HC12. How long does it take to run? This time should match your answer which you got in the prelab.