Microcomputers 1
Lab 5
Subroutines

Concepts:

- Writing subroutines in assembly

Objectives: Get hands-on experience with stacks and subroutines

- Write a subroutine in an assembly program that implements a subroutine to conduct a given task and calls a subroutine several times from the main program with different array data sets.
- Learn how the return address is saved
- Learn how the CPU registers are stored and restored.

I. Introduction

In Lab5, you will write a subroutine that computes the average of an array with **N** **8-bit unsigned** elements. The main program will define the array and call the subroutine several times.

- **Array Data declaration:** Define three array data sets as below using assembly directives in the data section in your program.

```
Array1:  DC.B    $11, $22, $11, $22, $22, $66, $77, $88, $99, $10
N1       equ     10

Array2:   DC.B    $33, $10, $FE, $09, $FF
N2        equ   5

Array3:    DC.B    $01, $02, $03, $04, $05, $06, $07, $08, $09, $0A, $0B, $0C
N3         equ   12
```

Then the memory locations for three outcomes will be reserved as below:

```
          ORG $1020
Array1_mean:    DS.B   2
Array2_mean:    DS.B   2
Array3_mean:    DS.B   2
```

For Lab5, two parameters are passed using CPU registers to the subroutine.

- **The size of the array** is passed by value in **register Y.**
- **The beginning address of the array** is passed by reference in **register X**.
- The average values of the given array will be **returned from the subroutine** in register D (actually B register).

- The main program calls the subroutine for each array defined in your data section. Write the main program that calls the subroutine three times to calculate the average values in Array1, Array2, and Array3. Pseudocode for the main program is as below:

```
1) The X index register holds the starting address of the Array.
       ex) LDX   #Array1

 2)  Y register is   holding the size of Array.
       ex) LDY  #N1

 3)  The subroutine 'MyAvg' is called to calculate the average of the given array.
       ex) JSR  MyAvg

 4)  Store the return value in D register in the memory
       ex) STD  Array1_mean


      .... Repeat the above code for other arrays
Endmain:  BRA   Endmain
```

- Write a subroutine to calculate the given array's average. In the main program, the array's size is **passed by value in register Y**. In addition, **the beginning address of the array** is **passed by reference in register X**.

1) In the subroutine, you need to save the **size of the array passed in register Y on the top of the stack** at the beginning of the subroutine because of the reasons below:
   - register Y is used as the counter inside the loop and decremented every loop iteration.
   - the size of the array is needed when the average of the given array is calculated.

   Before register Y is modified inside the loop, register Y is pushed on the stack. When the **size of the array** is needed and the information will be pulled out.
2) The sum of the given array is accumulated in register D within the loop. Once the loop is terminated, the subroutine calculates the average. First, the array's size is pulled out from the top of the stack to register X. Then **IDIV** instruction (D register/ X register) will calculate the average value of the array.

3)

II. Flowchart to Calculate the Average of the given Array

The flowchart in Figure 1 represents an algorithm for calculating the average of an array with N 8-bit elements. According to the flowchart, you will **calculate the sum of the array first**. To calculate the sum of the array of N 8bit unsigned numbers, two HCS12 addition instructions are involved **to handle carry to the upper byte of the sum** as below inside the loop:

```
ADDB  0, X     ; add register B with the content in the memory location in index register X

ADCA  #$00     ; If there is a carry, the carry is added into the A register.   A= A + Carry + $00
               ; (Register A is the upper byte of register D).
```

At the end of the loop, the sum of the array is saved in D register.

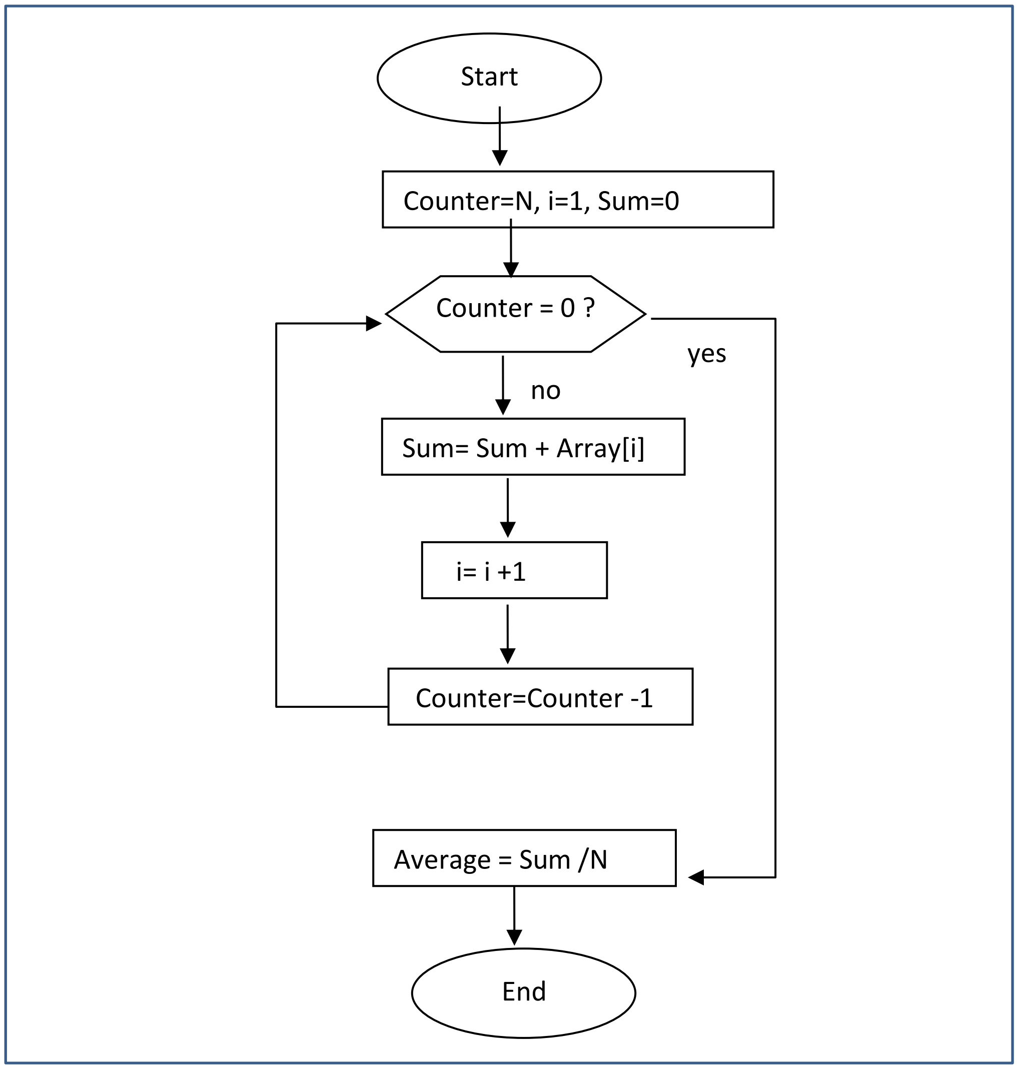
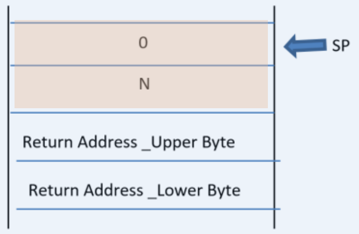


Figure 1 – Flowchart for calculating the average of an array with N 8-bit elements.

The pseudo code for the subroutine is below:

```
MyAvg:      ; The label  is served as the subroutine' name. The subroutine must start with the label.

    Step 1.   Clear register A
    Step 2.   Clear register B
    Step 3.   Register Y holds the size of the array. Since Y is going to be decremented every loop
              iteration inside the loop. The size of the array in register Y is saved on the top of the
              stack.

              PSHY            ; save the array length N onto the stack. Y is a 2-byte register.
```



```
    Step 4.   Loop:   CPY    #00                    ; Y register is a counter.
    Step 5.       Branch to Exit the loop if  Y is zero
    Step 6.       Add the array element at the memory location in X to register B.
    Step 7.       Add the carry to register A.
    Step 8.       Increment register X to hold the address of the next element.
    Step 9.       Decrement the loop counter register Y.
    Step 10.      BRA Loop

   Step 11: ExitLoop:
              Retrieve the size of the array from the top of the stack into register X.
              Use the PULX operation.  Register X is a divider in the IDIV instruction.
   Step 12:   Calculate the average of the given array using the instruction IDIV.

   Step 13:   After execution of the instruction IDIV, register X holds the quotient and register D
              holds the remainder.
              Save the Quotient in register D by exchanging register D with register X.
   Step 14:   RTS          ; end of the subroutine.
```

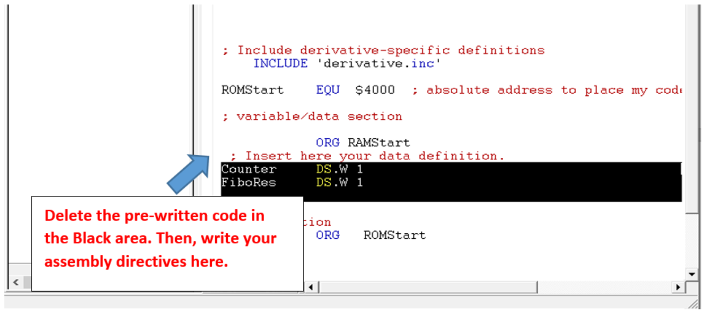
Tips:

1. Use register Y as a loop counter and register X as an index register to access the array element.
2. Calculate the sum of the array using two addition instructions inside the loop. The sum is saved in register D.
   i. **ADDB** 0, X ; add the array element to B register
   ii. **ADCA** **#0** ; add carry to the upper byte of the sum

3. Calculate the average using the IDIV instruction. To execute this instruction, the array length is pulled out from the stack to register X. After the division operation, register X holds the quotient and register D holds the remainder.
4. Exchange register X with register D to save the quotient in register D.

III. Lab5 Project creation

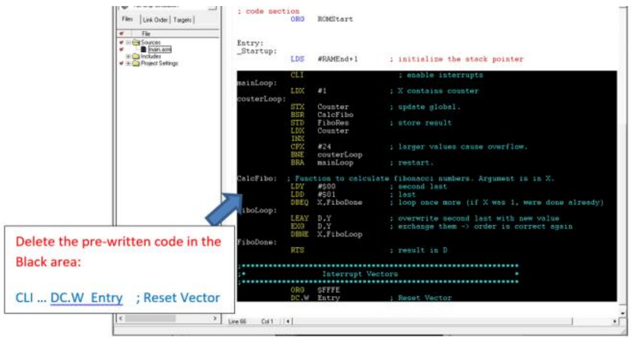
1. Create a new project with the name '**Lab5.mcp**' as you did in the previous labs. Open the main.asm file in the project to write the code.From the dropdown list on the left, select "**Full Chip Simulation**" if it is not so yet.
2. Delete the prewritten data allocation marked with a black background. Then, write your data allocations on page 1 here.



```
; Include derivative-specific definitions
    INCLUDE 'derivative.inc'

ROMStart    EQU  $4000  ; absolute address to place my code

; variable/data section

            ORG RAMStart
 ; Insert here your data definition.
Counter     DS.W 1
FiboRes     DS.W 1

; code section
            ORG    ROMStart
```

Delete the pre-written code in the Black area. Then, write your assembly directives here.

3. Delete the prewritten sample code marked with a black background in main.asm and write your code here.



```
; code section
            ORG   ROMStart


Entry:
_Startup:
            LDS   #RAMEnd+1       ; initialize the stack pointer

            CLI                     ; enable interrupts
mainLoop:
            LDX   #1              ; X contains counter
couterLoop:
            STX   Counter         ; update global.
            BSR   CalcFibo
            STD   FiboRes         ; store result
            LDX   Counter
            INX
            CPX   #24             ; larger values cause overflow.
            BNE   couterLoop
            BRA   mainLoop        ; restart.

CalcFibo:  ; Function to calculate fibonacci numbers. Argument is in X.
            LDY   #$00            ; second last
            LDD   #$01            ; last
            DBEQ  X,FiboDone      ; loop once more (if X was 1, were done already)
FiboLoop:
            LEAY  D,Y             ; overwrite second last with new value
            EXG   D,Y             ; exchange them -> order is correct again
            DBNE  X,FiboLoop
FiboDone:
            RTS                   ; result in D

;**************************************************************
;*                 Interrupt Vectors                          *
;**************************************************************
            ORG   $FFFE
            DC.W  Entry           ; Reset Vector
```

Delete the pre-written code in the Black area:

CLI ... DC.W Entry ; Reset Vector

5. After you complete the code, then, click on the green debug arrow on the menu bar. The simulator/Debugger window opens up.
6. In Lab5, **the program counter (PC) in the register window starts at $4000.** The instruction starting at the memory location $4000 is to initialize the stack pointer as below:

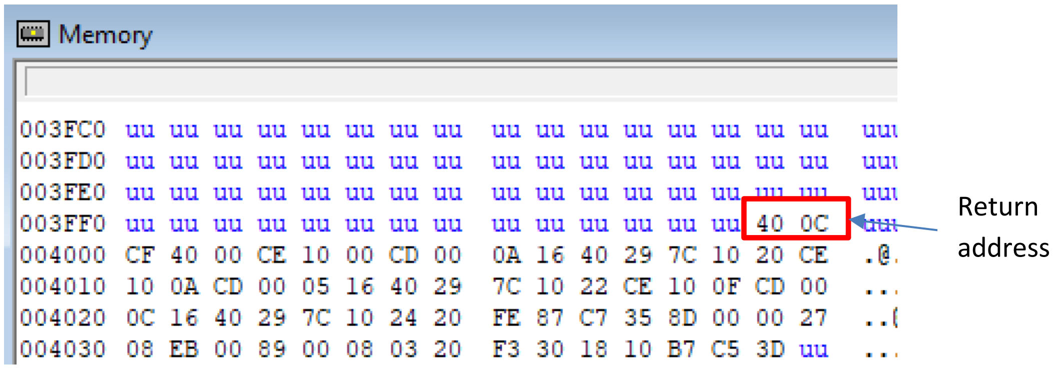
    LDS #RAMEnd+1

7. For the stack trace, run the program with single step (or F11). The Stack Pointer (SP) holds an address of the top of the stack. It will be decreased when the data is pushed and increased when the data is pulled. In Lab5, the stack changes when the following instructions are executed: **JSR, RTS, PSHY, PULX**

Complete the stack diagrams in the provided excel sheet ('Stack Diagram.xlsx'). You need to mark the stack whenever **the stack has** changed.

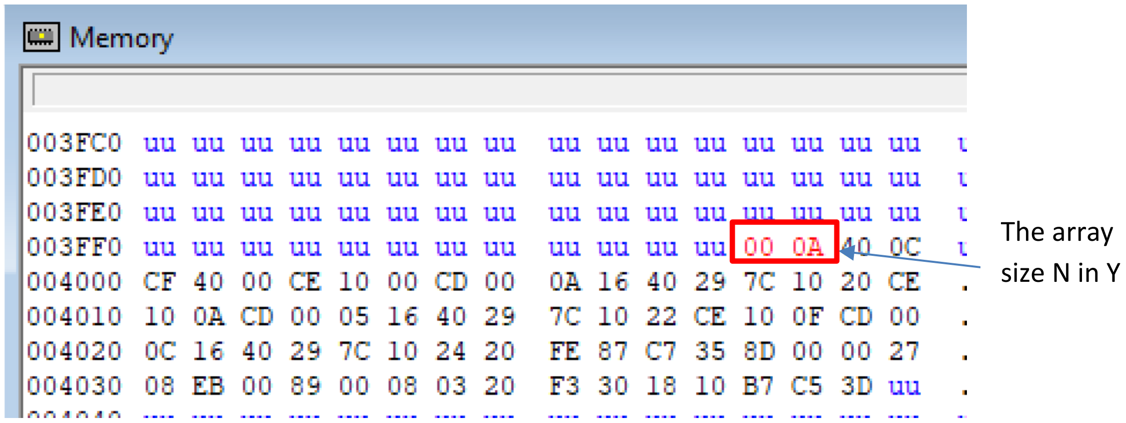
Please see the Stack Diagram when the first subroutine is called

-**JSR** MyAvg instruction is executed



```
003FC0 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuı
003FD0 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuı
003FE0 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  uuı
003FF0 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu 40 0C  uuı
004000 CF 40 00 CE 10 00 CD 00  0A 16 40 29 7C 10 20 CE  .@.
004010 10 0A CD 00 05 16 40 29  7C 10 22 CE 10 0F CD 00  ...
004020 0C 16 40 29 7C 10 24 20  FE 87 C7 35 8D 00 00 27  ..(
004030 08 EB 00 89 00 08 03 20  F3 30 18 10 B7 C5 3D uu  ...
```

-**PSHY** in the subroutine is executed



```
003FC0 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  ı
003FD0 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  ı
003FE0 uu uu uu uu uu uu uu uu  uu uu uu uu uu uu uu uu  ı
003FF0 uu uu uu uu uu uu uu uu  uu uu uu uu 00 0A 40 0C  ı
004000 CF 40 00 CE 10 00 CD 00  0A 16 40 29 7C 10 20 CE  .
004010 10 0A CD 00 05 16 40 29  7C 10 22 CE 10 0F CD 00  .
004020 0C 16 40 29 7C 10 24 20  FE 87 C7 35 8D 00 00 27  .
004030 08 EB 00 89 00 08 03 20  F3 30 18 10 B7 C5 3D uu  .
```

What to Submit in Blackboard: Group submission

1) **Per each member: 50pts**
   Completed the stack diagram in the provided excel sheet ('Stack Diagram.xlsx').

2) **One copy per group: 50 pts**

   - The assembly source code file under the "Sources" folder (main.asm files only)