Introduction to Assembly language

This lab will introduce you to the concept of writing PIC18 assembly language. For this purpose, we are going to implement a simple code. Let us take the following exercise:

In this lab, we are going to write in Assembly language instead of C language. As you are familiar by now with the use of MPLAB X, you will need to do the same to compile an assembly program as you have done in lab#1 part 3.

PART A)

The first project is to implement the assembly code that is equivalent to part 1 of lab#2. In short, we are to read the four switches connected to PORT A and display them to the LEDs connected to PORTB.

C Code:

```
ADCON1 = 0x0f;
TRISA = 0xff;
TRISB = 0x00;

while (1)
{
        IN = PORTA & 0x0F;
        PORTB = IN;
}
```

Compile and run the following program (make sure that this is in a new folder called lab3p2):

```
; THIS SECOND ASSEMBLY LANGUAGE PROGRAM WILL READ THE VALUES OF
; ALL THE BITS 0-3 OF PORT A AND OUTPUT THEM
; TO THE PINS 0 THROUGH 3 OF PORT B

#include <P18F4620.inc>

        config      OSC = INTIO67
        config      WDT = OFF
        config      LVP = OFF
        config      BOREN = OFF

        ORG         0x0000

START:

        MOVLW       0x0F            ; Load W with 0x0F0
        MOVWF       ADCON1          ; Make ADCON1 to be all digital

        MOVLW       0xFF            ; Load W with 0xFF
        MOVWF       TRISA           ; Set PORT A as all inputs

        MOVLW       0x00            ; Load W with 0x00
        MOVWF       TRISB           ; Make PORT B as outputs
```

```
MAIN_LOOP:

        MOVF    PORTA, W    ; Read from PORT A and move it into W
        ANDLW   0x0F        ; Mask with 0x0F
        MOVWF   PORTB       ; Move from W to PORT B

        GOTO    MAIN_LOOP   ; Loop forever
        END
```

After you have compiled and downloaded the program into the board, change one switch at a time and check that the corresponding LED does change according to the logic state of the switch.

PART B)

Next, your team will implement part 2) of lab #2 in assembly.

Take the provided code in Part A) and modify it to control the RGB LED D1 connected to PORTC. Just use the c code done in lab 2) part 2) as reference and change it into assembly based on the example code provided above.

PART C)

The third example is to test the switch at PORT A it 0 and to set/clear the bit 0 of PORTB according to the logic state of PORT A bit 0.

If PORTA Bit 0 == 0, then clear bit 0 of PORTB
If PORTA Bit 0 == 1, then set bit 1 of PORTB.

Pseudo C Code:

```
If (PORTA & 0x01 == 0) PORTB = 0;
else PORTB = 0x01;
```

Compile and run the following program:

```
#include <P18F4620.inc>

        config      OSC = INTIO67
        config      WDT = OFF
        config      LVP = OFF
        config      BOREN = OFF


        ORG         0x0000

; CODE STARTS FROM THE NEXT LINE

START:
        MOVLW       0xFF        ; Load W with 0xFF
        MOVWF       TRISA       ; Set PORT A as all inputs

        MOVLW       0x00        ; Load W with 0x00
        MOVWF       TRISB       ; Make PORTB bits 0-7 as outputs

        MOVLW       0x0F        ; Load W with 0x0F
        MOVWF       ADCON1      ; Set ADCON1

MAIN_LOOP:
        BTFSC       PORTA, 0    ; If Bit 0 of PORTA = 0 skip the next instruction
        GOTO        CASE_A0EQ1  ; else go to CASEA0EQ1 (PORTA Bit 0 = 1)

CASE_A0EQ0:
        BCF         PORTB, 0    ; case PORTB bit 0 = 0, clear bit 0 of PORTB
        GOTO        MAIN_LOOP   ; go back to Main Loop

CASE_A0EQ1:
        BSF         PORTB, 0    ; case PORTB bit 0 = 1, set bit 0 of PORTB
        GOTO        MAIN_LOOP   ; go back to Loop
```

PART D)

Take the Assembly program below and modify it to meet the following conditions:

1) The RGB LED at D1 (connected to PORTC bits 0 to 2) should show the color WHITE.
2) The RGB LED at D2 (connected to PORTD bits 0 to 2) should show the color RED.
3) When the program is properly executed, both the RGB LED D1 and D2 will be blinking.
4) Play around with the value of 'OUTER_VALUE' by modify its value and reprogram the code. Check if the blinking rate does vary with the variation of that variable.

```
#include <P18F4620.inc>

    config          OSC = INTIO67
    config          WDT = OFF
    config          LVP = OFF
    config          BOREN = OFF


Color_PORTC     equ     0x??    ;<- replace ?? with proper value
Color_PORTD     equ     0x??    ;<- replace ?? with proper value
Color_Off       equ     0x??    ;<- replace ?? with proper value

OUTER_VALUE     equ     0x10    ;<- value of outer loop

Saved_D1_loc    equ     0x22    ; memory address for saved Color for D1
Saved_D2_loc    equ     0x23    ; memory address for saved Color for D2
Saved_OV_loc    equ     0x24    ; memory address for saved Outer Value

    ORG         0x0000

; CODE STARTS FROM THE NEXT LINE

START:

    MOVLW   0x00            ; Load W with 0x00
    MOVWF   TRISC           ; Make PORT C bits 0-7 as outputs
    MOVWF   TRISD           ; Make PORT D bits 0-7 as outputs

MAIN_LOOP:

    MOVLW   Color_PORTC     ; Load W with the WHITE color for D1 at PORTC
    MOVWF   Saved_D1_loc    ; save desired color into memory location Saved_D1_loc

START_TEST:

    MOVLW   Color_PORTD     ; Load W with the desired RED color for D2 at PORTD
    MOVWF   Saved_D2_loc    ; save desired color into memory location Saved_D2_loc
    MOVLW   OUTER_VALUE     ; Load OUTER_VALUE into W
    MOVWF   Saved_OV_loc    ; save it to the memory location Saved_OV_loc

COLOR_LOOP:

    MOVFF   Saved_D1_loc,PORTC  ; Get saved color of PORTC and output to that Port
    MOVFF   Saved_D2_loc ,PORTD ; Get saved color of PORTD and output to that Port
    MOVFF   Saved_OV_loc,0x21   ; Copy saved outer loop value  to 0x21

; NESTED DELAY LOOP TO HAVE THE FIRST HALF OF WAVEFORM

LOOP_OUTER_1:
    NOP                     ; Do nothing
```

```
   MOVLW     0x80
   MOVWF     0x20                ; Load saved inner loop value to 0x20


LOOP_INNER_1:
   NOP                           ; Do nothing
   DECF      0x20,F              ; Decrement memory location 0x20
   BNZ LOOP_INNER_1              ; If value not zero, go back to LOOP_INNER_1

   DECF      0x21,F              ; Decrement memory location 0x21
   BNZ LOOP_OUTER_1              ; If value not zero, go back to LOOP_OUTER_1

   MOVLW     Color_Off           ; Load W with the second desired color
   MOVWF     PORTC               ; Output to PORT C to turn off the RGB LED D1
   MOVWF     PORTD               ; Output to PORT D to turn off the RGB LED D2
   MOVFF     Saved_OV_loc,0x21   ; Copy saved outer loop value  to 0x21

; NESTED DELAY LOOP TO HAVE THE FIRST HALF OF WAVEFORM BEING LOW

LOOP_OUTER_2:
   NOP                           ; Do nothing
   MOVLW     0x80
   MOVWF     0x20                ; Load saved inner loop value to 0x20

LOOP_INNER_2:
   NOP                           ; Do nothing
   DECF      0x20,F              ; Decrement memory location 0x20
   BNZ LOOP_INNER_2              ; If value not zero, go back to LOOP_INNER_2

   DECF      0x21,F              ; Decrement memory location 0x21
   BNZ LOOP_OUTER_2              ; If value not zero, go back to LOOP_OUTER_2


; START ALL OVER AGAIN

  GOTO       MAIN_LOOP           ; Go back to main loop
 END
```

PART E)

Take the Assembly program in part D) and modify it to meet the following conditions:

1) Use the two switches connected to PORT A bits 1 and 0 as inputs.
2) The LED D1 will always blink with the WHITE color
3) Based on those two inputs, make the LED D2 blink ON and OFF with the color and blinking rate as indicated on the table below.

PORT A		RGB LED D2 at PORT D bits 0-2
Bit_1	Bit_0	**Action**
0	0	RED color blinking at original rate of part D)
0	1	GREEN color blinking at twice the rate of part D)
1	0	BLUE color blinking at three times the rate of part D)
1	1	WHITE color blinking at four times the rate of part D)

Hint:

1) Make sure to start the program by setting the TRISB, TRISC and TRISD registers for proper direction of the input and output pins. Also, don't forget to program the ADCON1 register.

2) Use the 'BTFSC' instruction to test the logic state of the input bit you want to check. You will need to one BTFSC to test the upper bit (bit 1) to determine what group (RED/GREEN) versus (BLUE/WHITE). Once the group is isolated, then use another BTFSC to test the lower bit to determine the individual color to display for each group.

3) Place the checking of the switches after the label: START_TEST. You will end up having four sets of the following lines:

```
MOVLW   Color_xx          ; Load W with the desired RED color for D2 at PORTD
MOVWF   Saved_D2_loc      ; save desired color into memory location Saved_D2_loc
MOVLW   OUTER_VALUE_xx    ; Load OUTER_VALUE into W
MOVWF   Saved_OV_loc      ; save it to the memory location Saved_OV_loc
```

depending on the switch settings. Once the values have been determined, make a jump to the common point at the label 'COLOR_LOOP'.