CALIFORNIA STATE POLYTECHNIC UNIVERSITY, POMONA
COLLEGE OF ENGINEERING

ECE 3301L Spring 2019 Session 2 Microcontroller Lab **Felix Pinai**

LAB3
Introduction to Assembly language

PART A)

This lab will expose 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:

Pseudo C Code:

```
Initialize PORTA as an output port
while (1)
{
        turn on the LED;
        wait();
        turn off the LED;
        wait();
}
```

Or the actual c code:

```
void main()
{
        ADCON1 = 0x0F;                  // Force Ports A & B to be all digital
        TRISA= 0x00;                    // set PORTA as outputs

        while (1)
        {
                PORTA = 0x0A;           // set bits 1 and 3 of PORTA on
                for (i=0xFF; i>0; i--)  // wait for (outer loop)
                {
                        for (j=0xFF; j>0; j--);  // inner wait loop
                }

                PORTA = 0x05;           // set bits 0 and 2 of PORTA on
                for (i=0xFF; i>0; i--)  // wait for (outer loop)
                {
                        for (j=0xFF; j>0; j--);  // inner wait loop
                }
        }
}
```

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#2. The only differences are in two areas.

First, instead of selecting the XC8 compiler, you will need to select the 'mpasm' option.

Second, the next difference is in the creation of the new file. We are not going to use the C source code but instead the Assembly source code. When you are at the step to add the new file, do File>New File. A new window will appear. Select 'Assembler' and then 'AssemblyFile.asm' and hit Next. You will need to enter the new file name. In this case, I would call it 'Lab3p1'. Hit Finish.

The next phase is to create the assembly file. Copy the following text and paste into the file.

```
; THIS FIRST ASSEMBLY LANGUAGE PROGRAM WILL FLASH AN LED CONNECTED
; TO THE PINS 0 THROUGH 3 OF PORT B


#include<P18F4321.inc>

        config      OSC = INTIO2
        config      WDT = OFF
        config      LVP = OFF
        config      BOR = OFF


; Constant declarations

Delay1 equ          0xFF
Delay2 equ          0xFF

        ORG         0x0000

 ; CODE STARTS FROM THE NEXT LINE

START:

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

        MOVLW       0x00            ; Load W with 0x00
        MOVWF       TRISA           ; Make PORT A as outputs

MAIN_LOOP:

        MOVLW       0x0A            ; Load W with value 0x0A
        MOVWF       PORTA           ; Output to PORT A
```

```
; NESTED DELAY LOOP

        MOVLW       Delay1          ; Load constant Delay1 into W
        MOVWF       0x21            ; Load W to memory 0x21

LOOP_OUTER_1:
        NOP                         ; Do nothing
        MOVLW       Delay2          ; Load constant Delay2 into W
        MOVWF       0x20            ; Load W to memory 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       0x05            ; Load W with value 0x05
        MOVWF       PORTA           ; Output to PORT A (flipping the LEDs)

; NESTED DELAY LOOP AGAIN

        MOVLW       Delay1          ; Load constant Delay1 into W
        MOVWF       0x21            ; Load W to memory 0x21

LOOP_OUTER_2:
        NOP                         ; Do nothing
        MOVLW       Delay2          ; Load constant Delay2 into W
        MOVWF       0x20            ; Load W to memory 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
```

When done, do ‘File’ and ‘Save’.

You can now build and download your program to the controller board just like you have done in the previous lab.

You will need to use the connections from the previous lab (Lab #2). The four LEDs connected to PORTA should be used. If your program was built properly, the LEDs should blink at a constant speed.

PART B)

The next example is to read the four switches connected to PORT B and display them to the LEDs connected to PORTA. This is the same exercise as PART 2) of Lab #2.

Pseudo C Code:

```
IN = PORTB & 0x0F;
PORTA = 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<P18F4321.inc>

        config      OSC = INTIO2
        config      WDT = OFF
        config      LVP = OFF
        config      BOR = 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       TRISB           ; Set PORT B as all inputs

        MOVLW       0x00            ; Load W with 0x00
        MOVWF       TRISA           ; Make PORT A as outputs

MAIN_LOOP:

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

        GOTO        MAIN_LOOP       ; Loop forever

        END
```

PART C)

The third example is to test the switch at PORT B Bit 0 and to set/clear bits 0 and 1 of PORTA according to the logic state of that bit.

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

Pseudo C Code:

```
if ((PORTB & 0x01) == 0) PORTA = 0x01;
else PORTA = 0x02;
```

Compile and run the following program:

```
#include<P18F4321.inc>

        config      OSC = INTIO2
        config      WDT = OFF
        config      LVP = OFF
        config      BOR = OFF


        ORG         0x0000

; CODE STARTS FROM THE NEXT LINE

START:
        MOVLW       0xFF        ; Load W with 0xFF
        MOVWF       TRISB       ; Set PORT B as all inputs

        MOVLW       0x00        ; Load W with 0x00
        MOVWF       TRISA       ; Make PORTA bits 0-7 as outputs

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

MAIN_LOOP:
        BTFSC       PORTB, 0    ; If Bit 0 of PORTB = 0 skip the next instruction
        GOTO        CASEB1      ; else go to CASEB1 (PORTB Bit 0 = 1)

CASEB0:
        BSF         PORTA, 0    ; case PORTB bit 0 = 0, set bit 0 of PORTA
        BCF         PORTA, 1    ; and clear bit 1 of  PORTA
        GOTO        LOOP        ; go back to Main Loop

CASEB1:
        BCF         PORTA, 0    ; case PORTB bit 0 = 1, clear bit 0 of PORT A
        BSF         PORTA, 1    ;set bit 1 of  PORTA
        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 should show the color RED.
2) The RGB LED at D2 should show the color WHITE.
3) Both LEDs D1 and D2 should blink ON and OFF with a period of 20 msec (10 msec being ON and 10 msec being OFF)
4) Place a scope probe at the pin of either D1 or D2 with the active color to measure the period of that signal. The precision should be at +/- 0.2 msec.

```
#include<P18F4321.inc>

   config              OSC = INTIO2
   config              WDT = OFF
   config              LVP = OFF
   config              BOR = 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     0x??    ;<- replace ?? with proper value
INNER_VALUE         equ     0x??    ;<- replace ?? with proper 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 desired color for PORTC
   MOVWF    0x22                ; save desired color into register 0x22
   MOVLW    Color_PORTD         ; Load W with the desired color for PORTD
   MOVWF    0x23                ; save desired color into register 0x23


   MOVLW    OUTER_VALUE         ; Load OUTER_VALUE into W
   MOVWF    0x24                ; save it o register 0x24

   MOVLW    INNER_VALUE         ; Load INNER_VALUE into W
   MOVWF    0x25                ; save it to register 0x25
```

```
COLOR_LOOP:

   MOVFF      0x22,PORTC          ; Get saved color of PORTC and output to that Port
   MOVFF      0x23,PORTD          ;Get saved color of PORTD and output to that Port
   MOVFF      0x24,0x21           ; Copy saved outer loop cnt from 0x24 to 0x21

; NESTED DELAY LOOP TO HAVE THE FIRST HALF OF WAVEFORM

LOOP_OUTER_1:
   NOP                            ; Do nothing
   MOVFF      0x25,0x20           ; Load saved inner loop cnt from 0x25 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      0x24,0x21           ; Copy saved outer loop cnt from 0x24 to 0x21

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

LOOP_OUTER_2:
   NOP                            ; Do nothing
   MOVFF      0x25,0x20           ; Load saved inner loop cnt from 0x25 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 B bits 1 and 0 as inputs.
2) Based on those two inputs, make the LED D1 blink ON and OFF with the color and frequency as indicated on the table below. The LED D2 will always blink with the WHITE color
3) Place a scope probe at any pin of D2 to measure the period of that signal. The precision should be at +/-0.2 msec. Since D2 is always WHITE, any pin of D2 should reflect the blinking period of the LED D1

PORT B		**RGB LED D1 at PORT D bits 0-2**
Bit_1	Bit_0	Action
0	0	RED color blinking every 20 msec
0	1	GREEN color blinking every 40 msec
1	0	BLUE color blinking every 80 msec
1	1	WHITE color blinking every 160 msec

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.
3) Add codes between the labels 'MAIN_LOOP' and 'COLOR_LOOP' to determine what color to output and how long to generate the timing for that color. Use the instruction 'BTFSC' used in PART C) to test the bits of PORTB to determine the color to display