CALIFORNIA STATE POLYTECHNIC UNIVERSITY, POMONA
COLLEGE OF ENGINEERING

ECE 3301L Spring 2023 Session 2 Microcontroller Lab **Felix Pinai**

LAB3
Introduction to Assembly language

In this lab, we are going to write in Assembly language instead of C language.

PART 1)

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 follows:

1) Go to the Projects box.
2) Select the project Lab3p1
3) Right click and scroll down to 'Copy' and click on it
4) A box will appear with the name of the original project. Change the name of the Project to be 'lab3p1' to create part 1) of lab3
5) The project location should be with the new directory lab3\Part1
6) Select the button 'Copy' to create the new project
7) Once the new project is created, go to that project in the 'Projects' area and right click on that new project and scroll down to 'Set as Main Project' and click on that. After that, the project name should be in bold
8) Go back and select that project again and right click on it
9) Scroll all the way down to 'Properties'
10) A new window will pop up. On the right side under the 'Compiler Toolchain', instead of selecting the XC8 compiler, you will need to select the 'mpasm' option. select a version of the assembler under 'mpasm' and hit 'OK'
11) We are not going to use the C source code but instead the Assembly source code. Now you are at the step to add the new file, do File>New File. A new window will appear. Select 'Assembler' 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 LEDS
; CONNECTED TO THE PINS 0 THROUGH 3 OF PORT B

#include<P18F4620.inc>

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


; Constant declarations

Delay1      equ         0xFF
Delay2      equ         0XFF

Counter_L   equ         0x20
Counter_H   equ         0x21

        ORG         0x0000

 ; CODE STARTS FROM THE NEXT LINE


START:

        MOVLW       0x0F            ;
        MOVWF       ADCON1          ;

        MOVLW       0x00            ;
        MOVWF       TRISB           ;


MAIN_LOOP:

        MOVLW       0x05            ;
        MOVWF       PORTB           ;

                                    ;
        CALL        DELAY_ONE_SEC   ;


        MOVLW       0x0A            ;
        MOVWF       PORTB           ;

        CALL        DELAY_ONE_SEC   ;


        GOTO        MAIN_LOOP       ;
```

```
DELAY_ONE_SEC:
        MOVLW       Delay1              ;
        MOVWF       Counter_H           ;

LOOP_OUTER:
        NOP                             ;
        MOVLW       Delay2              ;
        MOVWF       Counter_L           ;

LOOP_INNER:
        NOP                             ;
        DECF        Counter_L,F         ;
        BNZ         LOOP_INNER          ;

        DECF        Counter_H,F         ;
        BNZ         LOOP_OUTER          ;
        RETURN

        END
```

PART 2)

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            ;
        MOVWF       ADCON1          ;

        MOVLW       0xFF            ;
        MOVWF       TRISA           ;
        MOVLW       0x00            ;
        MOVWF       TRISB           ;

MAIN_LOOP:                          ;
        MOVF        PORTA, W        ;
        ANDLW       0x0F            ;
        MOVWF       PORTB           ;

        GOTO        MAIN_LOOP       ;
        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 3)

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

Take the provided code in the above Part 1) 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 4)

We will implement now the part 3) of Lab #2. We do need to write an infinite loop with an internal loop that count from 0 to 7 and then repeat itself while outputting that count to PORTC and then call a subroutine to delay 1 second.

The following program will implement the FOR loop by using an up counter saved at the location 0x20 and it is used as an index for the color to be outputted to the PORT. In addition, it will use another counter at location 0x21h that is initialized with the value of 08h at the start. The counter at Color_Value will be incremented by 1 each time through the loop while the counter Loop_Count will be decremented by 1. The counter Loop_Count will be initialized with the value of 8. When it reaches the value of 0, the FOR loop is completed.

The subroutine DELAY_ONE_SEC is called once a color is outputted to the port for the purpose of creating a long delay to allow the color to be displayed for a good amount of time.

```
#include <P18F4620.inc>
 config OSC = INTIO67
 config WDT = OFF
 config LVP = OFF
 config BOREN = OFF

Counter_L    equ        0x20
Counter_H    equ        0x21
Color_Value  equ        0x28
Loop_Count   equ        0x29

 ORG 0x0000


; CODE STARTS FROM THE NEXT LINE
START:
        ORG         0x0000

START:
        MOVLW       0x0F                ;
        MOVWF       ADCON1              ;

        MOVLW       0x00                ;
        MOVWF       TRISC               ;

WHILE_LOOP:                             ;
        MOVLW       0x00                ;
        MOVWF       Color_Value         ;
        MOVLW       0x08                ;
        MOVWF       Loop_Count          ;
```

```
FOR_LOOP:
        MOVF        Color_Value,W       ;
        MOVWF       PORTC               ;
        CALL        DELAY_ONE_SEC       ;

        INCF        Color_Value,F       ;
        DECF        Loop_Count,F        ;
        BNZ         FOR_LOOP            ;
                                        ;
        GOTO        WHILE_LOOP          ;

        END
```

Remember to add the code under 'DELAY_ONE_SEC' from part 1) of this lab to the above code (before 'END').

PART 5)

From the array(s) generated in lab #2 part 5), fill in two sequences of 8 values on 8 consecutive locations. For example, for the first array of data, use an arbitrary location like 0x50. Put in the first value of the array at the location. Next, take the second value and place it at the next location 0x51. Repeat the process for the remaining values of the array. The next step is to repeat the same process for the second array and pick another starting location like 0x60.

After the creation of those memory locations, now use the indirect addressing mode (with the registers FSR0 and FSR1) to fetch the color value to be outputted to the PORT(s) associated with the LEDs D2 and D3. Use the code from Part 4) above to add the changes.