University of Jordan
School of Engineering
Department of Mechatronics Engineering
Microprocessor and Microcontroller Laboratory
0908432
Exp. 3: MPLAB Basics

Objective

To be familiar with assembly language programming and the Microchip PIC 16 series instruction set.

Pre-lab Preparation:

Review Experiment 2 thoroughly.
Read chapter 7 of the PIC16F84 data sheet.
Review the Status Register- Section 2.2.1 in the book

Procedure:

This lab experiment is composed of three Parts. These parts are an interactive one where you will be introduced to some PIC instructions and investigates their syntax, parameters, and usage. The experiment involves using MPLAB and implementing codes to learn key issues.

1)The "**EQU**" directive

The equate directive is used to assign labels to numeric values. They are used to **DEFINE CONSTANTS** or to **ASSIGN NAMES TO MEMORY ADDRESSES OR INDIVIDUAL BITS IN A REGISTER** and then use the name instead of the numeric address.

Example1: - In this part we will learn the **equ** directive and how we can use it in our programs to make it meaningful.

A)

[ ] Open a new Text document and write on it the following:

```
Tmr0  equ 01
movlw 06
movwf Tmr0
nop
nop
end
```

[ ] Save the text file you created with Lab3_P1.1a.asm
[ ] Create a new project, Name the project Lab3Part1.1a
[ ] Compile the project and note if there are any errors, did the compiler recognize Tmr0, how?

B)

```
Num1 equ 20   ;GPR @ location 20
Num2 equ 40   ;GPR @ location 40
Movlw 5       ; move the constant 5 to the working register
Movwf  Num1   ; copy the value 5 from working register to Num1 (address 20)
Movlw 2       ; move the constant 2 to the working register
Movwf Num2    ; copy the value 2 from working register to Num2 (address 40)
Nop
End
```

2)The "**include**" directive

Suppose we are to write a huge program that uses all registers. It will be a tiresome task to define all Special Function Registers (SFR) and bit names using "equate" statements. Therefore, we use the include directive.

- The **include** directive calls a file which has all the equate statements defined for you and ready to use, its syntax is

#include "PXXXXXXX.inc" where XXXXXX is the PIC part number

Older version of include without #, still supported.

Example: #include "P16F84A.inc"

Example2: - In this part we will learn the **include** directive and how can we use it in our programs to make them meaningful.

[ ] Open a new Text document and write on it the following:

```
# include "p16f84.inc"
 movlw 0x3f
movwf STATUS
movlw 12
movwf OPTION_REG
nop
nop
 end
```

[ ] Save the text file you created in folder Lab3 with name Lab3_P1.2.asm.
[ ] Create a new project , Name the project Lab3Part1.2, and add the file Lab3_P1.2.asm
[ ] Compile the project and note if there are any errors, did the compiler recognize *OPTION_REG*, how?
[ ] View the *OPTION_REG* register from the Watch window, step through the code and notice if the value of *OPTION_REG* is being set to 12, if not why?

3- " MOVF" instruction.

The PIC-Micro instruction set has several instructions that are used to move data, literals as shown below:

MOVF	f, d	Move f
MOVWF	f	Move W to f
MOVLW	k	Move literal to W

MOVF:- This instruction moves the data stored in the register to either the working register or to the register itself. **This is the only data movement instruction that affects the STATUS register.**

MOVF	Move f
Syntax:	[*label*] MOVF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) → (destination)
Status Affected:	Z
Description:	The contents of register f are moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register, since status flag Z is affected.

[ ] Create a new project and name it Lab3Part1.3

[ ] Write the following code, and save it as Lab3_P1.3.asm

```
Location equ 0C
MOVLW 0
MOVWF Location
MOVF Location, 0
END
```

[ ] Build the project

[ ] Open the watch window, add the **WREG**, **STATUS** registers, and add the symbol Location.

[ ] Run and simulate the project. Did the value of Status Register change? If yes, why, and what is the affected bit? If not, explain.

[ ] What is the equivalent machine code of instruction MOVF TMR0, 0.

Exersice1: -

Write a code that initiates the memory location 0x0D with the value 0 and moves its contents to itself. Use the equ directive to give the GPR (General Purpose Register) a name and use it in your program. Watch the **STATUS** register, did its value change? **Explain**.

Exersice2: -

Write a code to copy the contents of location 0x0E to the location 0x1F

Exersice3: -

Write a code to exchange the contents of location 0x33 with location 0x11