University of Jordan
School of Engineering
Department of Mechatronics Engineering
Microprocessor and Microcontroller Laboratory
0908432
Exp. 2: Memory Decoding

Objective

To be familiar with Microchip MPLAB Integrated Development Environment (IDE) and the whole process of building a project, implementing, modifying simple codes, compiling the project, and simulating the code.

Pre-lab Preparation:

1- Read the Tutorial according using MPLAB (Start.pdf).
2- Read the PIC16F84 data sheet chapters 1, 2 especially (2.1, 2.2, 2.3).
3- Review the sections in the book regarding the Memory (Chapter 2) and MPLAB (Chapter 4).

Procedure:

This lab experiment is composed of two parts. All parts involved using MPLAB and implementing codes to learn key issues.

Part 1

In this section we will learn the steps necessary to create a project using MPLAB and then once created, we will learn how to compile it to create the necessary files that will allow us to simulate the project or alternatively, to program the microcontroller with the machine code generated.

[ ]   Create a directory on the PC in D drive under the Lab2 folder in which to store all of your work.

[ ]   Open a new text document and write the following:

```
movlw 06
movwf 01
nop
nop
end
```

[ ]   Save the text file you created with Lab2_P1.**asm (Make sure that the extension is .asm and not .txt).**

[ ]   Start the MPLAB software on your PC.

[ ] To create a project in MPLAB, follow the following simple steps: Select the Project → Project Wizard menu item. In the device selection menu, choose 16F84. Click next. In the Active Toolsuite, choose Microchip MPASM Toolsuite. Click next. Name the project Lab2Part1. For the project directory, make sure to browse to your created subdirectory. Click next. Add the file called Lab2_P1.asm. Make sure to check the box next to the name after the file has been added. Click next. Click Finish.

[ ] From the window Tab, select the Lab2Part1.mcw window. Double click on the Lab2_P1.asm file name in the project file tree. The file Lab2_P1.asm should open now in the editor window. This is where you will usually write your programs, debug them and simulate them.

[ ] To compile your project; from the Project Tab, choose Build All. Your program should compile now, and you should see a small window showing details of the compilation process and the following message "Build Succeeded" (Note the new files generated in your directory named: Lab2_P1.hex, Lab2_P1.lst, and Lab2_P1.err).

Exercise 1:

Write a code segment that initializes the INTCON register with the value 5. (You can get the address of INTCON register from the data memory map in PIC16F84 Datasheet.) Then compile it and make sure that there are no error messages.

Note: save the file as EX1.asm and the project Lab2_ex1.

Part 2

In this part we will use the same code used in Part1a.

Simulation is a very powerful tool in the hands of the embedded system developer. It allows us to run the code we have written on the computer and check whether it is working properly as expected without having to program the chip. In this part we will see an example of some of the abilities of the simulator.

[ ] From the Debugger Tab, choose Select Tool, and then enable MPLAB SIM for the simulator. Then from the same Tab, go to Settings. Select the Osc / Trace Tab and set the desired processor frequency to 4 MHz. This will tell the simulator in MPLAB to assume that Fosc is 4 MHz. Click OK to close the settings window.

[ ] From the Window tab, select the Tile Horizontally. This will show you all the current active windows in your project.

[ ] Select the Debugger → Reset, and then choose the Processor reset menu item or press F6. The software should highlight (with a green arrow). In this step, you have told the simulator to start behaving as if the microcontroller has just been given power. So, it is now ready to start execution of your program. It is now ready and waiting at the reset vector for your next command. Note that the program is not running yet.

[ ] Select the View → Watch Window. From the SFR drop down list, choose TMR0. Click the Add SFR button. Repeat the same procedure but for INTCON and OPTION _REG. You should now

note that your new Watch Window has these four register names listed, along with their addresses and contents. Select the TMR0 row. Right Click and select the Properties button. Note that you can view a register as hex, decimal, binary, or ASCII.

[ ] Hit F7 to step through the program one instruction at a time. Notice the PCL register counting in the status bar. The PCL register is the low byte of the program counter and shows what address in memory the microcontroller is going to execute next. This method of using the simulator is very useful when you want to check for errors and for debugging purposes. (Note that the value of TMR0 register after the execution of these two instructions is 06).

Exercise 2:

Write a code segment that initializes the TRISB register with the value 8. You can get the address of TRISB register from the data memory map in PIC16F84 Datasheet. To ensure that you wrote a correct code view the TRISB register from Watch window, step through the code, and notice if the value of TRISB is being initialized correctly, if not why?

Exercise 3:

Write a code segment that initializes the register with address 0X186 with the value 10, step through the code, and notice if the value of register is being initialized correctly, if not why?

……………………………………………………………………………………………………

……………………………………………………………………………………………………

Exercise 4:

Write a code segment that initializes the register with address 0X1E with the value 13, step through the code, and notice if the value of register is being initialized correctly, if not why?

……………………………………………………………………………………………………

……………………………………………………………………………………………………

Exercise 5:

Write a code segment that initializes the register with address 0X9A with the value 7, step through the code, and notice if the value of register is being initialized correctly, if not why?

……………………………………………………………………………………………………

……………………………………………………………………………………………………

Exercise 6:

Write a code segment that initializes the register with address 0X6A with the value 9, step through the code, and notice if the value of register is being initialized correctly, if not why?

……………………………………………………………………………………………………

……………………………………………………………………………………………………