

ECE383: Microcomputers – Lab 4

MPLAB Introduction and PIC24 Assembly Language

Goals: The goals of this lab are to introduce students to basic PIC24 assembly language, usage of the MPLAB Integrated Development Environment (IDE) and associated tools.

1. Introduction

This lab introduces the Microchip MPLAB Integrated Design Environment. All files referenced in the lab are assumed to be in `C:\microchip\chap3`. The tasks in this lab are:

- Use MPLAB to simulate the PIC24 assembly language program in the `C:\microchip\chap3\mptst_word.mcp` project to become familiar with the MPLAB environment.
- Implement some simple programming tasks using PIC24 assembly language.

This lab requires you to capture portions of the screen. The lab computers use the Windows operating system. This includes the “Snipping Tool” that may be used to capture portions of the screen. Other third party tools are also available.

As always, read through the entire lab and scan the supplied files before starting work. The reporting requirements have you verify computations performed by the assembly language program. In all cases, make it easy for the TA to verify your computations by showing your work. **NOTE:** When writing MPLAB assembly language programs, do not use variable names **a** or **b** as these are reserved names. All hand calculations requested should be shown on a separate piece of paper in the lab report.

2. Pre-lab

For this lab assignment, Task1 and Task 2 should be completed as a pre-lab assignment prior to your assigned lab time. You may also complete all the tasks (including Task 3 and Task 4) prior to the lab time. This may reduce your time in the lab, but is not required.

TA check: As soon as you enter lab, provide the TA with a pre-lab report that includes the complete assembly language programs. Lab time will be devoted to debugging the program execution and correcting any errors within MPLAB noted by the lab instructor. Make sure your group member names and date are on the pre-lab report.

3. TASK 1: MPLAB Introduction

Perform the following steps:

- Copy the files in `C:\microchip\chap3\` to a suitable directory on your network drive or hard disk. Make sure the directory you use has no spaces in the directory path. An example of a good directory name to

use would be `C:\temp\username` where “username” is your unique login name.

- Start the MPLAB IDE. Use *Project->Open* and open the *mptst_word.mcp* project located in your directory.
- Use *Configure->Select Device* to select the PIC24HJ128GP502 device for your processor.
- Use *Project->Build All* (Ctrl+F10) to assemble the program. If the source file is not already open, double-click on the *mptst_word.s* source file to open it.
- After the project is assembled, use *View->Program Memory* and open the program memory window. Scroll the window until you find your program in memory. This should start at location 0x204.
- Use *View->File Registers* to view data memory. Scroll to location 0x800, which is where your variables will start.
- Use *View->Special Function Registers* to view the special function registers (W0-W15, etc.).
- Open a watch window (*View->Watch*) and use *Add Symbol* and *Add SFR* to watch variable values and special function register values, respectively, of the **i**, **j**, **k** variables and the **W0** (WREG0) special function register.
- Use *Debugger->Select Tool->MPLAB Sim* to select the MPLAB Simulator.
- Use *Debugger->Step Over* (F8) to single step the program. Watch both the memory locations and watch window locations, and correlate their changing values with the instructions being executed.

TA check: Show the TA the task 1 results including the final state of the program, data memory, and the watch window. Use a screen capture tool to capture these windows and include them in your lab report. Include your source assembly language program in your lab report.

Modify the `avalue` equate (statement `.equ avalue, 2047`) to be the last four digits of your student ID. Reassemble the program and re-simulate it. Take a screen shot of both the Watch window contents and the memory window contents. In your report, you must show calculations that verify the screen shot values match the expected result (note, `avalue` is in decimal, as are the **last four digits of your student ID**: the values displayed in the file registers window are in HEX).

TA check: Show the TA the modified task 1 results including the final state of the program, data memory, and the watch window. Use a screen capture tool to capture these windows and include them in your lab report. Include your source assembly language program in your lab report.

4. TASK 2: myadd.s

Use *Project->Save Project As* and save the *mptst_word* project as a new project named *myadd*. Save the *mptst_word.s* file as *myadd.s*. Right-click on the *mptst_word.s* file in the left-hand workspace window and use the *Remove* option to remove it from the *myadd* project. Right click on the *Source Files* and use *Add Files* to add the *myadd.s* file to the project. Edit the *myadd.s* file and remove all of the instructions from `mov #avalue, W0` through `mov WREG, k`. You can now use this file as a start for a new program.

Your CWID student number is an eight digit number $Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$. For this task we will consider this to be an eight digit hexadecimal number.

Write a program to add the four digit hex number formed by $0xY_3Y_2Y_1Y_0$ to the four-digit hex number formed by $0xY_7Y_6Y_5Y_4$. Reserve space for two 16-bit variables in data memory named *lsp* and *mzp* to hold the hex values

0xY₃Y₂Y₁Y₀ and 0xY₇Y₆Y₅Y₄ respectively. Reserve space for a variable named *sum* to hold the sum of *lsp* and *mzp*.

The following C code describes the program. You must translate each line of the C program to corresponding assembly instruction(s).

```
uint8 aa=100, bb=22;
uint16 lsp, mzp, sum;

lsp = 0xY3Y2Y1Y0;           // Four digits of CWID treated as hex
mzp = 0xY7Y6Y5Y4;           // Four digits of CWID treated as hex
sum = lsp + mzp;

sum = sum + aa + bb;
```

Use the watch window to watch variables *aa*, *bb*, *lsp*, *mzp*, and *sum*. Also, use the data memory window to monitor the memory locations corresponding to these variables.

TA check: Show the TA the task 2 results including the final state of the program, data memory, and the watch window. Use a screen capture tool to capture these windows and include them in your lab report. Include your source assembly language program in your lab report. Since you will be working in teams in the lab, use only one student number for each team.

5. TASK 3: mysub.s

Create a project named *mysub* using the same procedure given in Task 2 (*Project ->Save Project As*, etc.) corresponding assembly language file named *mysub.s*. Using digits Y₅Y₄Y₃Y₂Y₁Y₀ from your student ID, write an assembly language program that implements the following C program. You must translate each line of the C program to assembly instruction(s).

```
uint16 xx=0xDEAD, yy=0xBEEF;
uint8 i, j, k, l, m;

i = Y1Y0;
j = Y3Y2;
k = Y5Y4;
l = i + k;
m = j - l;
xx=xx-yy-m;
```

Variables *i*, *j*, *k*, *l*, *m* are all 8-bit (byte) variables and *xx* and *yy* are 16-bit variables. Y₁Y₀, Y₃Y₂ and Y₅Y₄ are considered as decimal numbers for this task. Use the watch window to watch variables *i*, *j*, *k*, *l*, *m*, *xx*, and *yy*. Also, use the data memory window to monitor the memory locations corresponding to these variables. Write your program, simulate it, and verify that you calculate the correct results.

Hint: If you experience the error “Link Error: Cannot access symbol (xx) at an odd address”, initialize 16-bit variables first and then 8-bit variables. That is, all variables defined using the *.space 2* directive should appear before variables defined using the *.space 1* directive.

Give the value of the flags after the execution of each instruction. Assume that **W0 = j** and **W1 = l**.

Instruction	Value of flags			
	C	Z	OV	N
ADD.B W0,W1,W0				
SUB.B W0,W1,W0				

TA check: Show the TA the task 3 results including the final state of the program, data memory, and the watch window. Use a screen capture tool to capture these windows and include them in your lab report. Include your source assembly language program in your lab report.

6. TASK 4: mylogicops.s

Create a project named *mylogicops* using the same procedure given in Task 2 (*Project ->Save Project As*, etc.) corresponding assembly language file named *mylogicops.s*. Write an assembly language program that implements the following C program. You must translate each line of the C program to assembly instruction(s).

```
uint8 u8_a, u8_b, u8_c, u8_d, u8_e, u8_f;
uint16 u16_x=0x0001;

u8_a=0xAF;
u8_b=0x50;

u8_c= u8_a & u8_b;
u8_d= u8_a | u8_b;
u8_e= u8_a ^ u8_b;
u8_f=~u8_a;
u16_x=~u8_d | (u16_x & u8_c);
```

Use the watch window to watch variables *u16_x*, *u8_a*, *u8_b*, *u8_c*, *u8_d*, *u8_e*, and *u8_f*. Also, use the data memory window to monitor the memory locations corresponding to these variables. Write your program, simulate it, and verify that you calculate the correct results.

In addition to simulating your program within MPLAB, you must download your program to your PIC24 hardware (the Microstick II) and demonstrate the execution of your program on hardware to the TA. Make sure the Microstick II development board is attached to a USB port on your computer and make sure the slider switch on the board is set to position A. With the *mylogicops* project open, use the following steps

- If not already selected, use *Configure->Select Device* to select the PIC24HJ128GP502 device for your processor.
- Use *Debugger->Select Tool->Starter Kit on Board* to select the Microstick II as the target. You should see messages in the MPLAB Output window indicating a successful connection to the Microstick II board.
- Use *Project->Build All* (Ctrl+F10) to assemble the program. If the source file is not already open, double-click on the *mylogicops.s* source file to open it.

- Use *Debugger->Program* to download your code to the Microstick II. You should see messages in the MPLAB Output window indicating successful programming of the Microstick II board.
- After the project is downloaded, use *View->Program Memory* and open the program memory window.
- Scroll the window until you find your program in memory. This should start at location 0x204.
- Use *View->File Registers* to view data memory. Scroll to location 0x800, which is where your variables will start.
- Use *View->Special Function Registers* to view the special function registers (W0-W15, etc.).
- Open a watch window (*View->Watch*) and use *Add Symbol* and *Add SFR* to watch variable values and special function register values, respectively, of the all of the defined variables and the **W0** (WREG0) special function register.
- Set a breakpoint on the first line of your assembly language program by double clicking on the line in the assembly language program.
- Begin the execution of your program by selecting *Debugger->Run*.
- Use *Debugger->Step Over* (F8) to single step the program. Watch both the memory locations and watch window locations, and correlate their changing values with the instructions being executed.

TA check: Show the TA the task 4 results including the final state of the program, data memory, and the watch window for both the MPLAB PIC24 simulator and the PIC24 hardware. Use a screen capture tool to capture these windows and include them in your lab report. Include your source assembly language program in your lab report.

7. Report

At the next lab period, provide the TA a printed copy of the lab report following the ECE383 Lab report Template given on the class website. Each lab group will submit one joint lab report to the TA. Your report should have the reporting requirements needed for Tasks 1, 2, 3, and 4. **The TA will take off a significant number of points (up to 15% from total lab grade) if your assembly language source does not have the required comments.** The comments should indicate which assembly language source lines implement which C statements.

8. GRADING POLICY

1. Completion of the prelab assignment (20%)
2. Completion of Task 1 with results included in lab report (15%)
3. Completion of Task 2 with results included in lab report (15%)
4. Completion of Task 3 with results included in lab report (15%)
5. Completion of Task 4 with results included in lab report (15%)
6. Completeness, quality, and correctness of the lab report (20%)