ECE383: Microcomputers – Lab 5
C and PIC24 Assembly Language Programming

Goals: The goals of this lab are to introduce students to basic C language programming and equivalent PIC24 assembly language programming.

Introduction

This lab introduces basic C language programs and equivalent PIC24 assembly language programs. The tasks in this lab are:

- Implement programming tasks using the C language.
- Implement equivalent programs using the 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 any supplied files before starting work. The reporting requirements have you verify computations performed by the assembly language or C 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.

This lab will make use of several C header files discussed in the text. These files should be located in the *C:\microchip\lib\include* directory on the lab computers. These files were installed as a part of the PIC24 library available on the class website.

1. Prelab

For this lab assignment, the programs for **Task 1 and 2** should be completed as a pre-lab assignment prior to your assigned lab time.

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

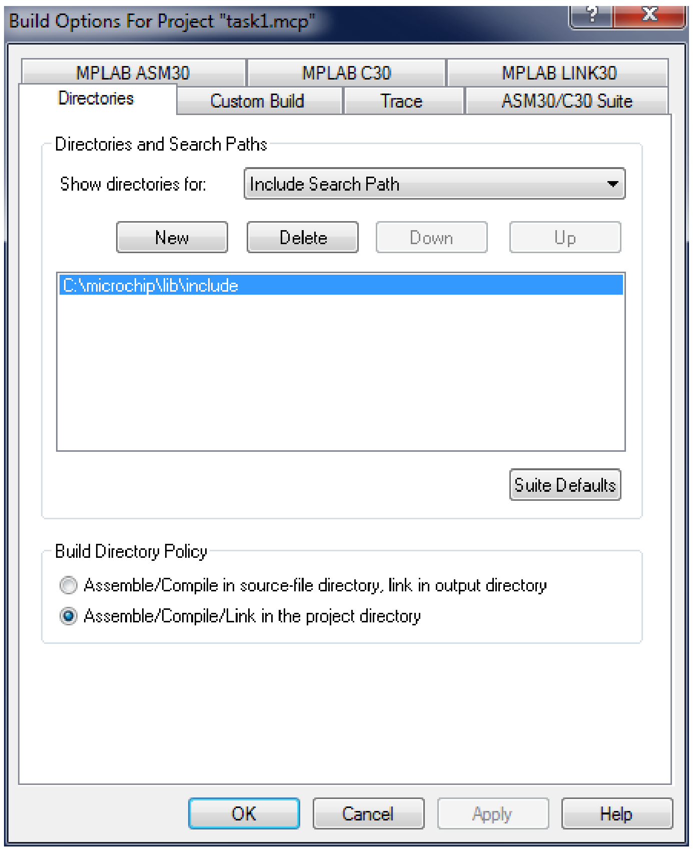
2. TASK 1: Basic C arithmetic operations

For this task you will create multiple C and assembly language projects implementing arithmetic operations. Perform the following steps:

- Start the MPLAB IDE. Use *Project->Project Wizard* for the creation of an MPLAB project.
    - Step One: Select the device *PIC24HJ128GP502*.

- o Step Two: Select the Active Toolsuite as *Microchip C30 Toolsuite*.
- o Step Three: *Create a New Project File* in a unique directory on the local hard disk (*C:\temp\task1.mcp* as an example).
- o Step Four: Skip the addition of existing files to the project.

- After the project is open, use *Project->Build Options* to add the *C:\microchip\lib\include* directory to the *Include Search Path* directory as shown in Figure 1 below.



Show directories for: Include Search Path

C:\microchip\lib\include

[ ] Assemble/Compile in source-file directory, link in output directory

[✓ Assemble/Compile/Link in the project directory]

Figure 1. Include Search Path Directory.

- Enter the C program below and save it with the name *task1.c* as a part of the project.

```
#include "pic24_all.h"

uint16 u16_a, u16_b, u16_c, u16_d;
uint8  u8_x,  u8_y,  u8_z;

void main(void) {
  u8_x=0xFF;
  u8_y=0x01;
  u16_a = 0xFFFF;
  u16_b = 0x0001;

  u8_z=u8_x+u8_y;
  u16_d=(uint16) u8_x + (uint16) u8_y;
  u16_c=u16_a+u16_b;
}
```

- Use *Project->Build All* to compile the program. If the source file is not already open, double-click on the *task1.c* source file to open it.
- After the project is compiled, use *View->Program Memory* and open the program memory window. Scroll the window until you find your program in memory. Your program should begin at location 0x200.
- 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 (these will appear as WREG0-WREG15, etc).
- Open a watch window *(View->Watch)* and use *Add Symbol* to watch variable values of the all the program variables. Use *Add SFR* to watch the **SR** (status register) special function register value.
- Use *Debugger->Select Tool->MPLAB Sim* to select the MPLAB Simulator.
- Use *Debugger->Step Into (F7)* to single step through the program. Watch both the memory locations and watch window locations, and correlate their changing values with the instructions being executed. For all three of the arithmetic operations, record the value of the result and the value of the sign/negative (N), carry (C), zero (Z), and overflow (V) flags. The value of the flags can be determined from the SR register and are also indicated at the bottom of the MPLAB IDE.

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 language program in your lab report.

3. TASK 2: C Program check_val

Create an MPLAB project (*task2.mcp*) containing a C program (*task2.c*) that counts the number of one bits in a 16-bit unsigned integer named **`check_val`**. The count value should be stored in an 8-bit unsigned variable named **`ones_count`**. The program should also determine which is the first bit set in the **`check_val`** variable. The location of the first bit set should be stored in an 8-bit unsigned variable named **`first_one`**. For example, if **`check_val`**=0xF508 then the computed values should be **`ones_count`**=7 and **`first_one`**=3.

Hint: Use a for loop and shift right every iteration of the loop to simplify testing of a bit value. You must download your program to your PIC24 hardware (i.e. the Microstick II) and demonstrate the execution of your program on hardware to the TA.

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 language program in your lab report.

4. TASK 3: Assembly Language Program check_val

Create an MPLAB project (*task3.mcp*) containing an assembly language program (*task3.s*) that implements the equivalent of task 2.

You must download your program to your PIC24 hardware and demonstrate the execution of your program on hardware to the TA.

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 language program in your lab report.

5. TASK 4: Assembly to C Example

Create a new assembly project and input the assembly code as shown below. See the program results after executing the program. Create an MPLAB project (*task4.mcp*) containing a C program (*task4.c*) that implements the equivalent of the assembly program.

```
      .include "p24Hxxxx.inc"
      .global __reset

      .bss              ; Uninitialized data section
                        ; Variables start at location 0x0800
x:       .space 2       ; Allocating space (two bytes) to variable.
y:       .space 2       ; Allocating space (two bytes) to variable.
count:   .space 1       ; Allocating space (one byte) to variable.

;.........................................................................
; Code Section in Program Memory
;.........................................................................

      .text                        ; Start of Code section
__reset:                           ; first instruction located at __reset label
        mov #__SP_init, w15        ; Initialize the Stack Pointer
        mov #__SPLIM_init,W0
        mov W0, SPLIM              ; Initialize the stack limit register
                                   ; __SP_init set to be after allocated data

; User Code starts here.
           mov #0x3, w0
           mov.b wreg, count
           mov #0x1, w1
           mov w1, x
           mov #0x3, w2
           mov w2, y
top:
           cp0.b count
           bra z, done

           cp w1, w2
           bra nz, next
           inc w2, w2
           mov w2, y
next:
           cp w1, w2
           bra GEU, next2
           add #0x2, w1
           mov w1, x
next2:
           dec.b count
           bra top
done:      goto done  ; Place holder for last line of executed code
.end                  ; End of program code in this file
```

You must download your program to your PIC24 hardware and demonstrate the execution of your program on hardware to the TA.

TA check: Show the TA the task 4 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 language program in your lab report.

6. Laboratory Report

No later than a week from the day the lab is performed, 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 (15 points from total lab grade) if your C and assembly language source does not have the required comments.** The C programs should be appropriately commented. For the assembly programs, the comments should indicate which assembly language source lines implement which C statements.

7. 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%)