I. Learner Objectives:

At the conclusion of this lab, participants should be able to:
- Open, compile, disassemble, and debug a project
- Identify the types of files that are created when developing a project
- Identify the general purpose and core processor registers while stepping through an assembly program
- Comment a small program

II. Prerequisites:

Before reading this lab, participants should be able to:
- Summarize the material found in the Microprocessors versus Microcontrollers unit
- Summarize the material found in the Computer Architecture Overview unit
- Run MPLAB 8.86

III. Keywords:

Integrated Development Environment, PIC32 microcontroller, Digilent Cerebot MX4cK Embedded Controller Board, MPLAB

IV. Pre-lab:

Please feel free to install Microchip's MPLAB 8.86 software on your laptop(s) and/or PC(s). The requirements to install the software are listed below in the Background section.

V. Background:

MPLAB IDE:

MPLAB is an *Integrated Development Environment* (IDE) for developing embedded applications for Microchip's PIC microcontrollers. The environment allows for the development, building, and debugging of PIC applications. The applications may be created in XP/Vista/7 environments. MPLAB provides tools for project management, source file editing, chip simulation, and in-circuit emulation for the PIC RISC family of microcontrollers. You may read more about MPLAB here.

Project management in MPLAB is achieved through the use of a file which maintains information about the project. This file is created when a project is first established in MPLAB. Other important files that are created as the project is created and the source file is compiled or assembled include: source file (C .c or assembler .s, .S), HEX File (.hex), Object File (.o), and others. These files ensure that the project supports the creation and compilation/assembly of source files.

The source file editor provided in the IDE is seen in Figure 1. The common features, such as copying, cutting, and pasting, are found within the editor. This editor also

provides syntax color coding for key instruction words found in C and the MIPS instruction set and comments. The keywords are highlighted in blue and the comments are highlighted in green. When the source code is debugged the editor is also used. Breakpoints may be set within the editor in this mode.

```
/*****************************************************************************
 * Programmer: Andrew S. O'Fallon                                            *
 * Class: EE 234                                                             *
 * Programming Assignment: Lab 1.                                            *
 * Date: August 21, 2012                                                     *
 *                                                                           *
 * Description: This program reads the states of the on-board buttons of the *
 *              Cerebot MX4cK and writes the states to the on-board LEDs     *
 *                                                                           *
 *****************************************************************************/

#include <peripheral/ports.h> // PORTSetPinsDigitalIn (), PORTSetPinsDigitalOut (), PORTRead (), PORTWrit

// Yes, don't forget your prototypes
void setup_LEDs (void);
void setup_buttons (void);
unsigned int get_button_states (void);
void output_BTNs_to_LEDs (unsigned int button_states);

int main (void)
{
    unsigned int button_states = 0;

    setup_LEDs (); // Output pins
    setup_buttons (); // Input pins

    while (1) // Embedded programs run forever
    {
        // Get the state of the buttons
        button_states = get_button_states ();
        // Write the state of the buttons to the LEDs
        output_BTNs_to_LEDs (button_states);
    }

    return 0;
}
```

Figure 1: Text Editor with Syntax Color Coding

MPLAB 8 IDE provides many different options and views that enhance a developer's ability to produce a program. The different "Views" allow the developer to see the contents of the following:

- CPU Registers
- Local variables
- Peripherals and corresponding registers
- Program and data memory
- Dissasembly listing
- Call stack

MPLAB Programmer:

The MPLAB IDE allows the developer to download the program written, to the PIC device. The program is downloaded to the program flash of the device in the .hex format. For this lab we will be working with the PIC32MX460F512L MIPS microcontroller.

VI. Lab:

PART I – Compiling/Assembling, Downloading, and Executing a Project:

Lab 1 of the Microprocessor Systems sequence of instructional units will help you become familiar with the MPLAB 8 Integrated Development Environment (IDE). This lab will step you through a simple C language project that will be downloaded to the PIC32MX460F512L microcontroller integrated on the Digilent Cerebot MX4cK board. You will need MPLAB 8, Digilent Cerebot MX4cK Embedded Controller, and USB-A to micro-B programming cable.

Before you continue on with the lab make sure you have one end of the USB cable plugged into the Cerebot MX4cK micro-B "DEBUG" port and the other end into one USB-A port on your PC. Please refer to Figure 2 to find the "DEBUG" port. Turn on power to the board by flipping the switch next to the "DEBUG" port.

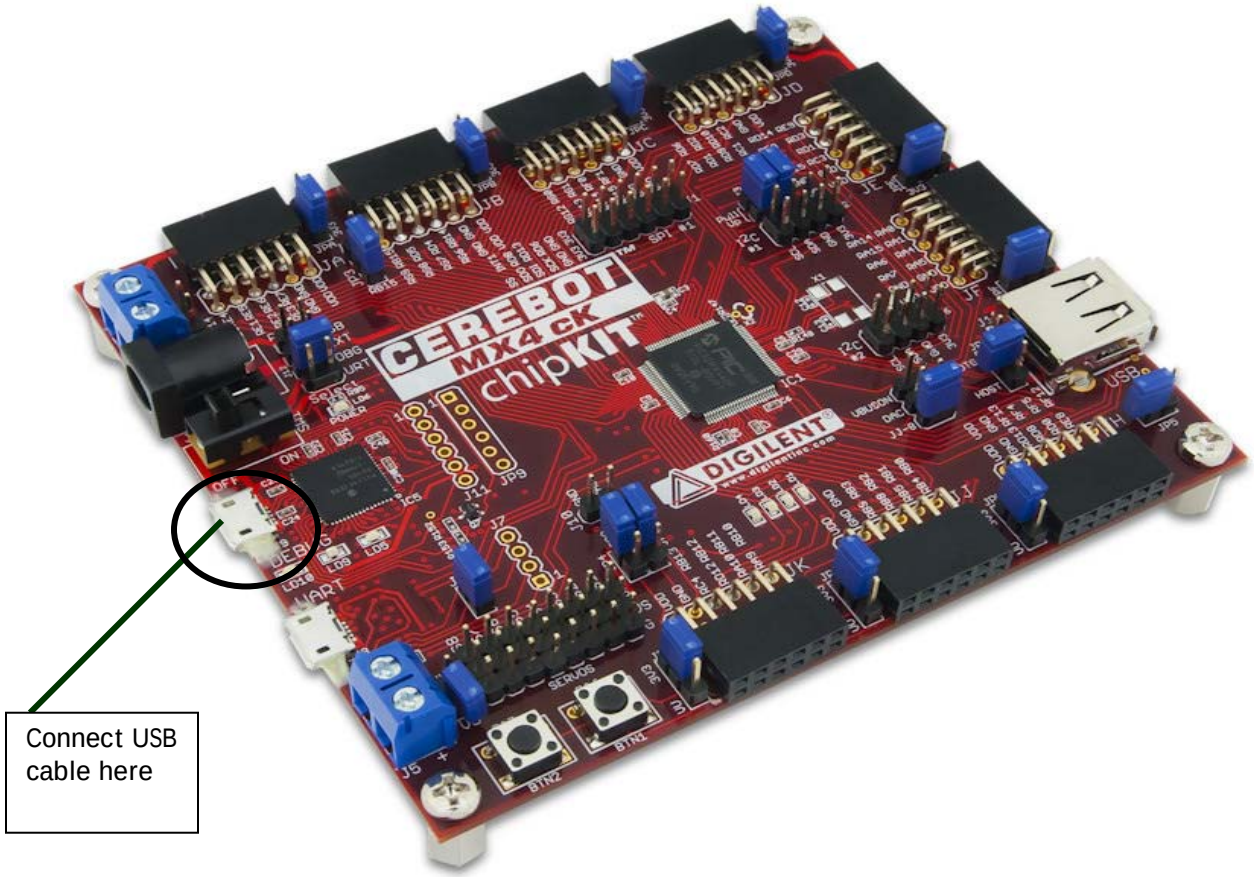

Figure 2: Digilent Cerebot MX4cK[1]

[1] Image is taken from: http://digilentinc.com/Data/Products/CEREBOT-MX4CK/Cerebot_MX4cK_rm.pdf

Setting up a Project:

Step 1: Open MPLAB IDE either by double clicking on the Desktop icon or by clicking "start" -> "All Programs" -> "Microchip" -> "MPLAB IDE v8.86" -> "MPLAB IDE". The window in Figure 3 should open. The panels within the window may not be aligned as seen in the figure, but you may arrange them according your preferences.

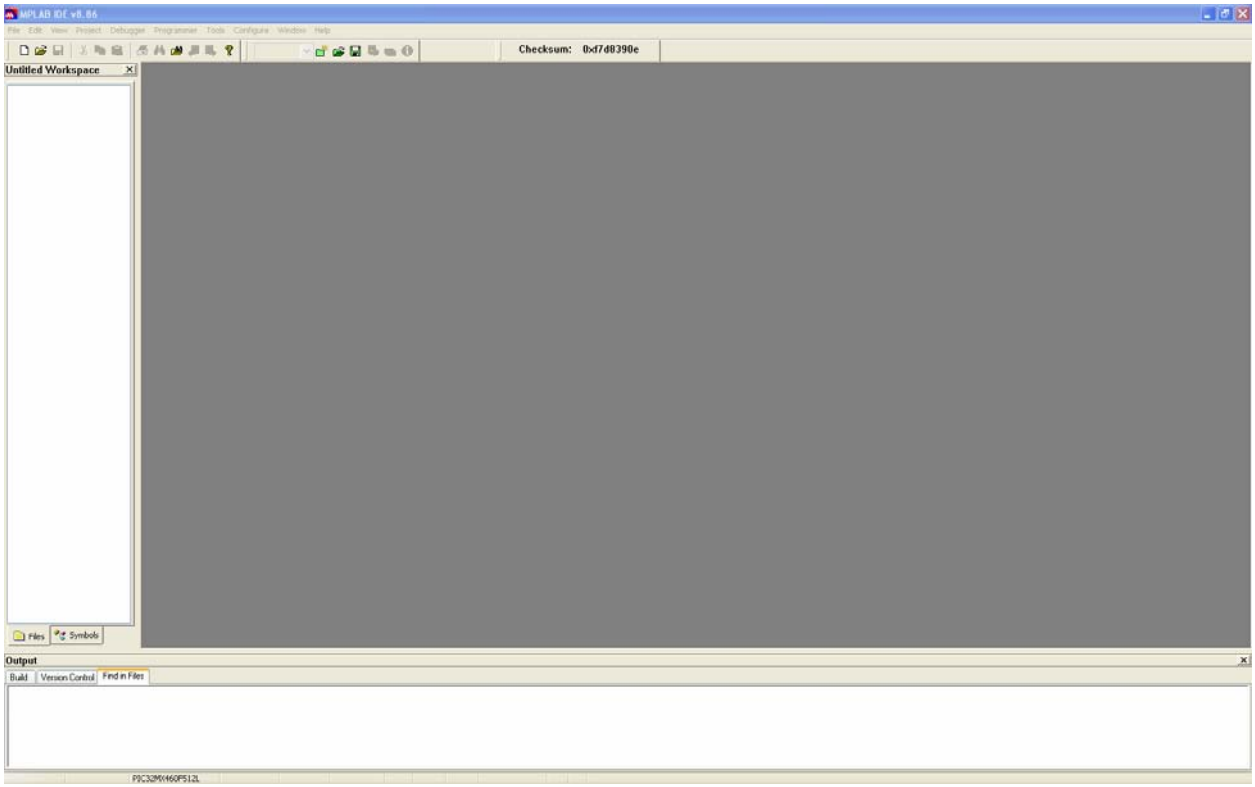

Figure 3: MPLAB Start Page

Step 2: Click on "Project" from the main toolbar and select "Project Wizard" (see Figure 4).

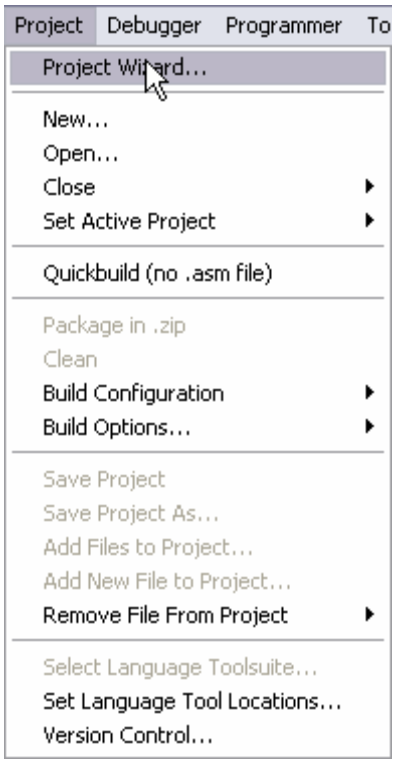

Figure 4: MPLAB New Project Window

Step 3: Click on "Next >" (Figure 5).

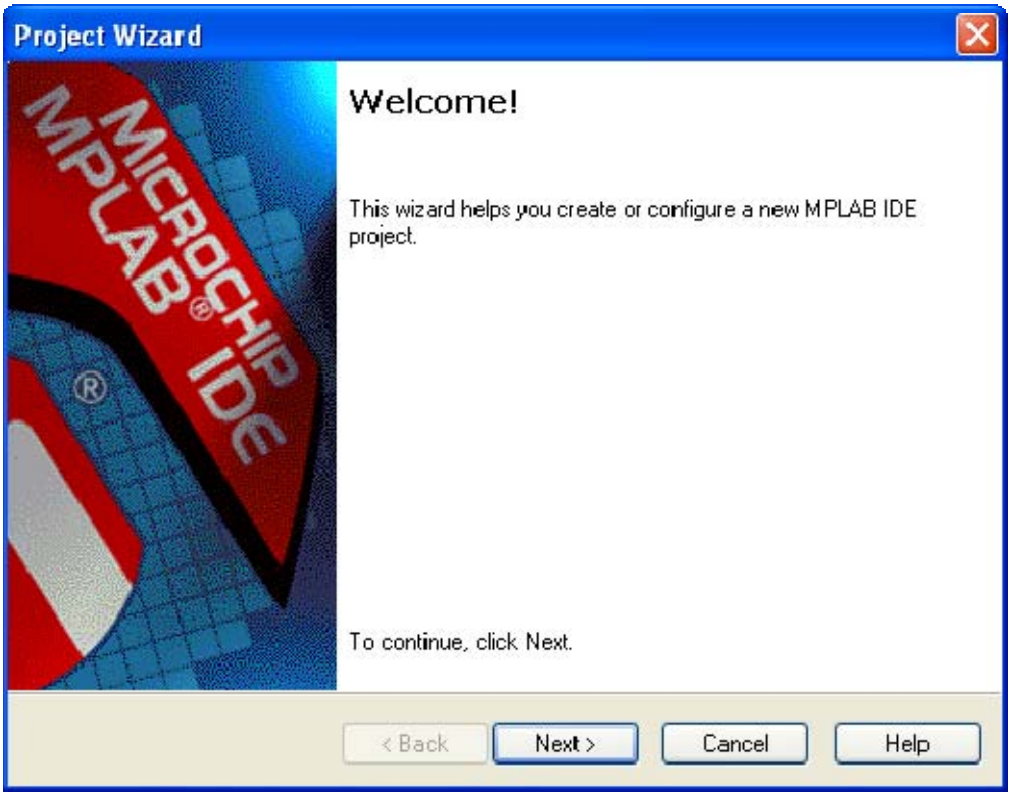

Figure 5: Project Wizard Welcome Window

Step 4: Scroll through the "Device:" list and select "PIC32MX460F512L" (see Figure 6). Recall this is the same microcontroller that you will find on the Digilent Cerebot MX4cK Embedded Controller Board. Click the "Next >" button.

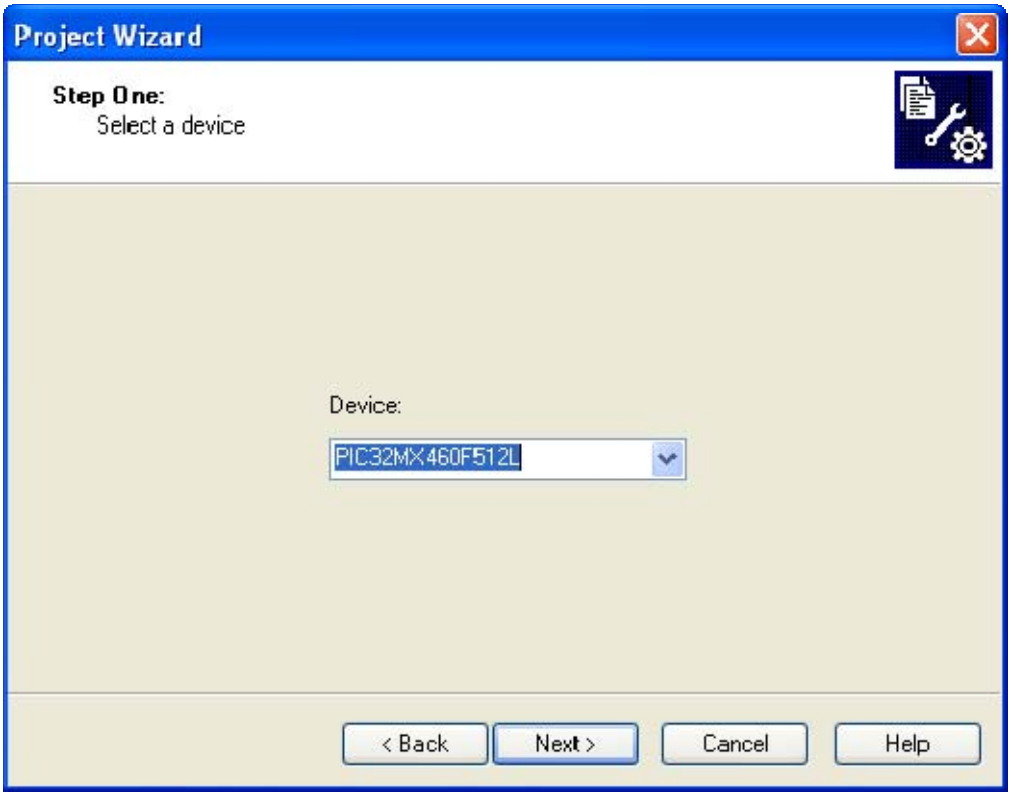

Figure 6: PIC32MX460F512L Device on Cerebot MX4cK

Step 5: First, select the "Microchip PIC32 C-Compiler Toolsuite" from the "Active Toolsuite" menu. Next, select "MPLAB C32 C Compiler" Note: you may have a different version of this compiler than the one shown in Figure 7. Before clicking "Next >" verify that the "Location" of the gcc executable is correct. In the future we will be using the MPLAB ASM32 Assembler, please observe this tool as an option.

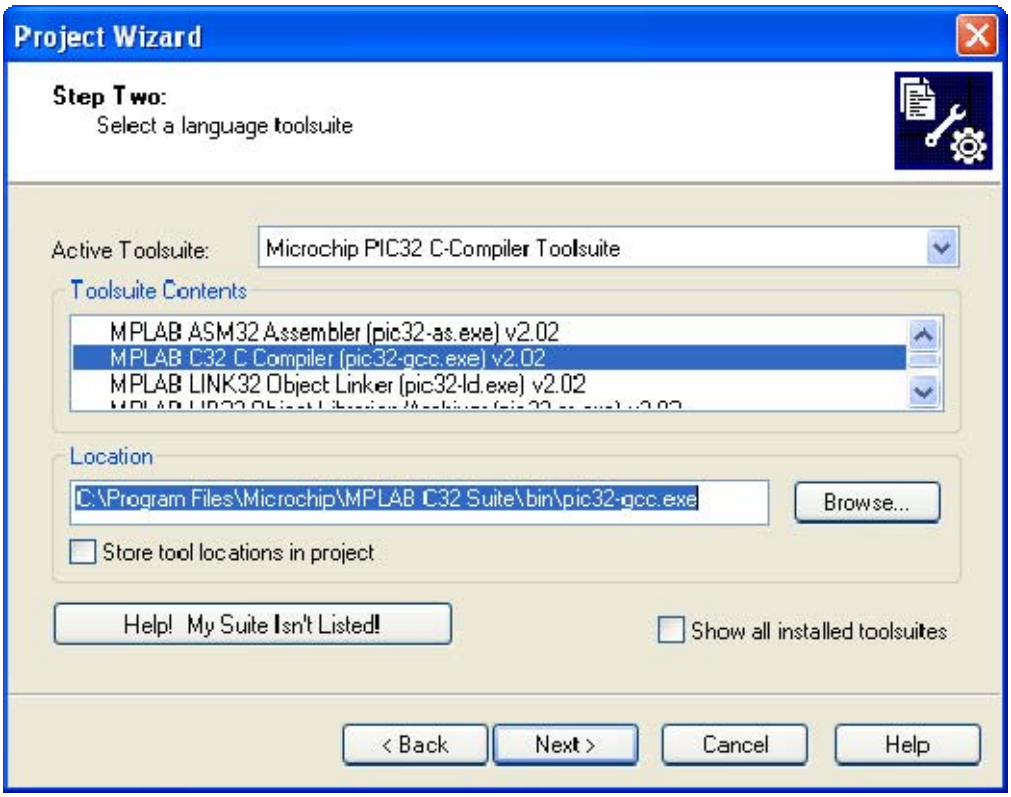

Figure 7: Compiler/Assembler Selection

Step 6: "Browse" for an appropriate path to store your project and provide an appropriate project name (Figure 8). Note: MPLAB does not, by default, create a folder for all of your files related to a project, be sure to create all appropriate folders yourself. Click "Next >".

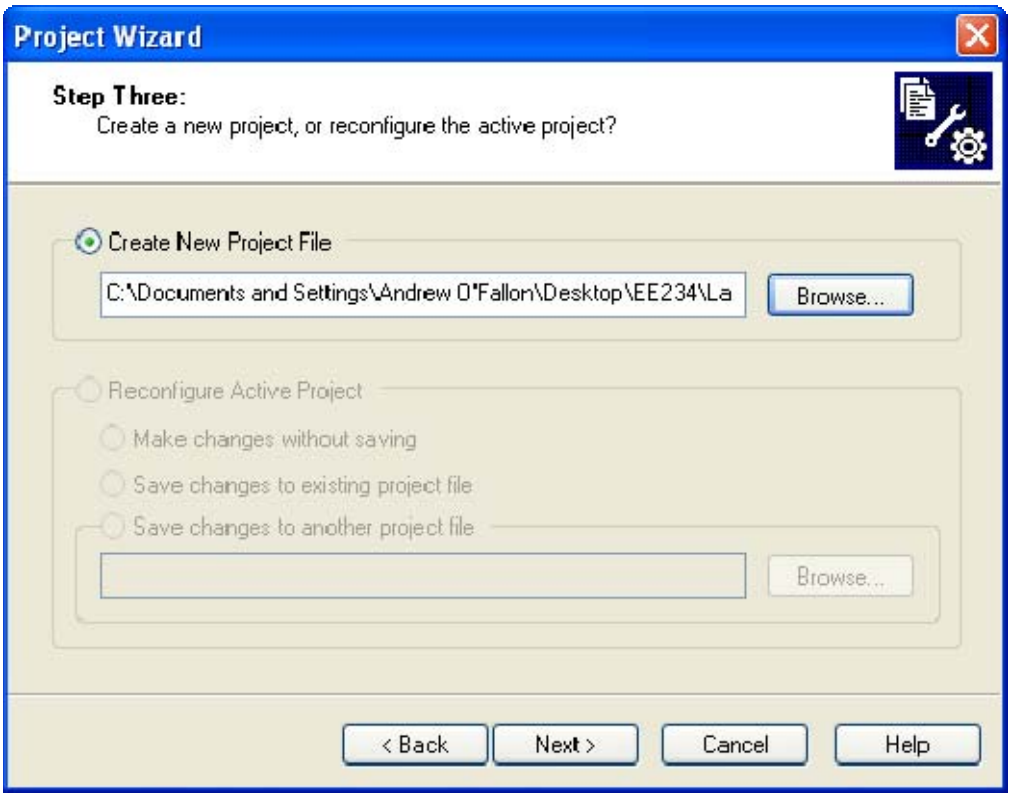

Figure 8: New Project Location

Step 7: At this point in our class you most likely do not have any pre-existing files that you want to "Add >>" to the project, so click the "Next >" button (see Figure 9).

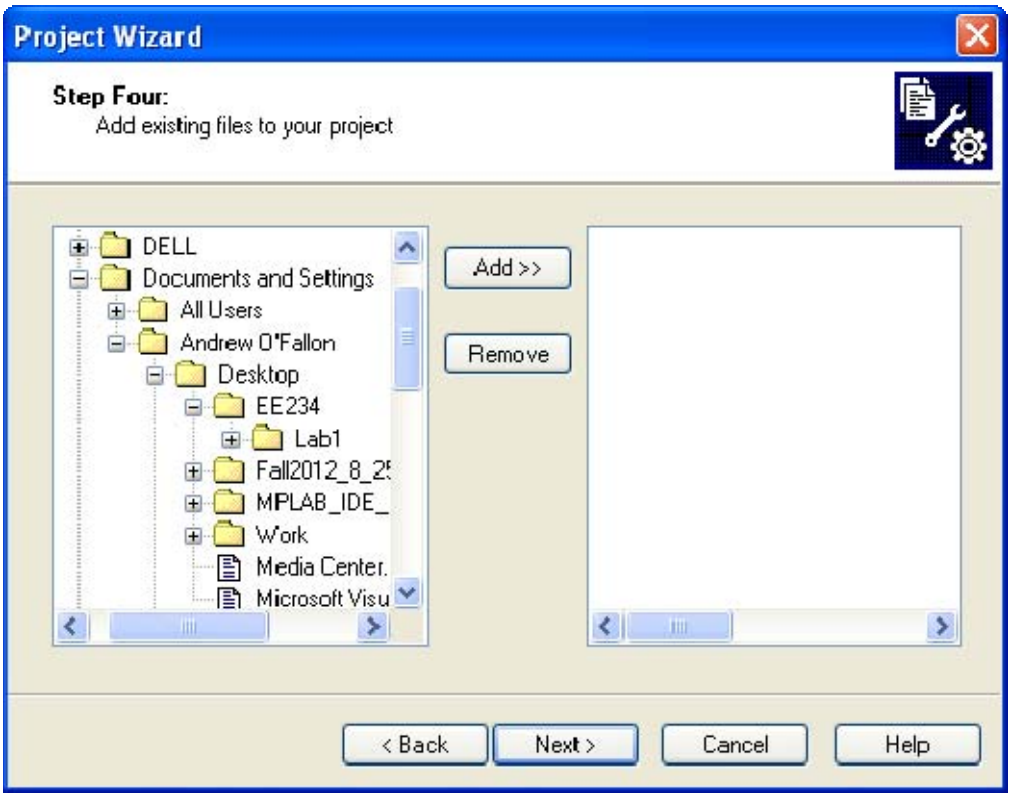

Figure 9: Adding Existing Files to Project

Step 8: Click "Finish" (Figure 10) to complete the project setup phase.

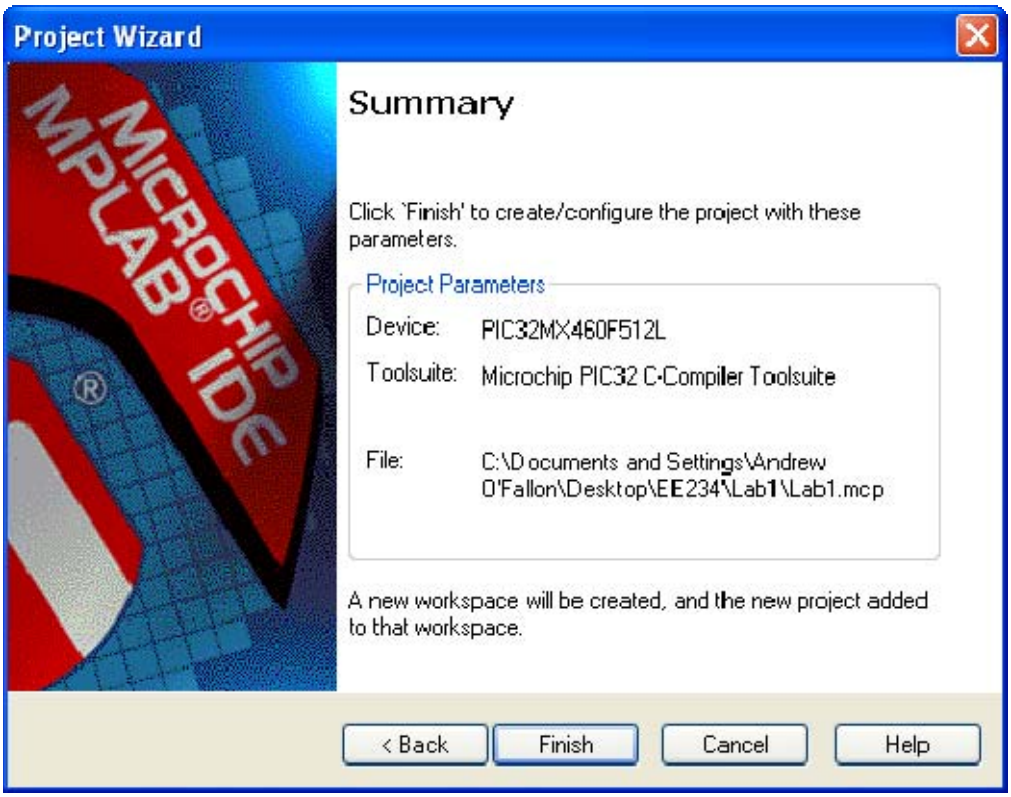

Figure 10: Summary Window

Step 9: Verify that the .mcw window listed in Figure 11 exists on your project workspace. If not you will want to display it.

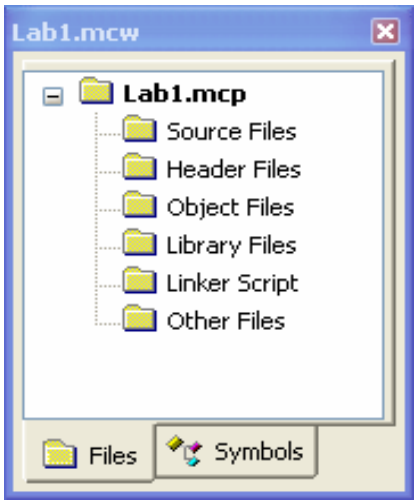

Figure 11: .mcw Window

Step 10: Create a new file by selecting "File" -> "New" (Figure 12).

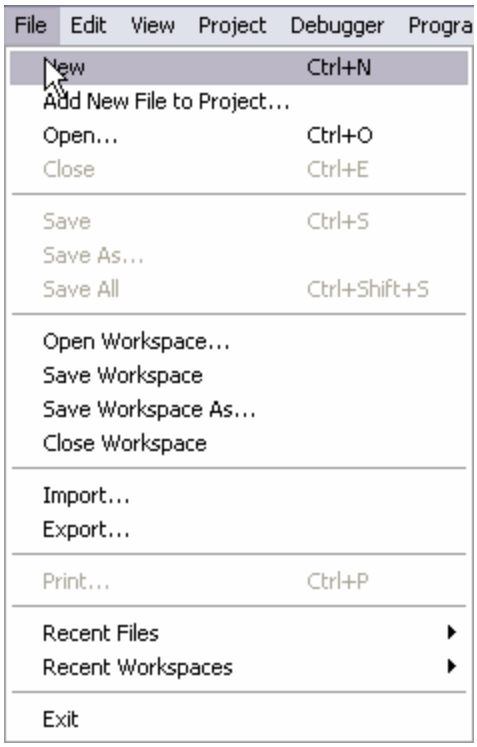

Figure 12: New File

The window in Figure 13 should display. You will need to name save it and add it to the project workspace.

Figure 13: Blank Source File

Step 11: To save the file select "File" -> "Save As…" (Figure 14).

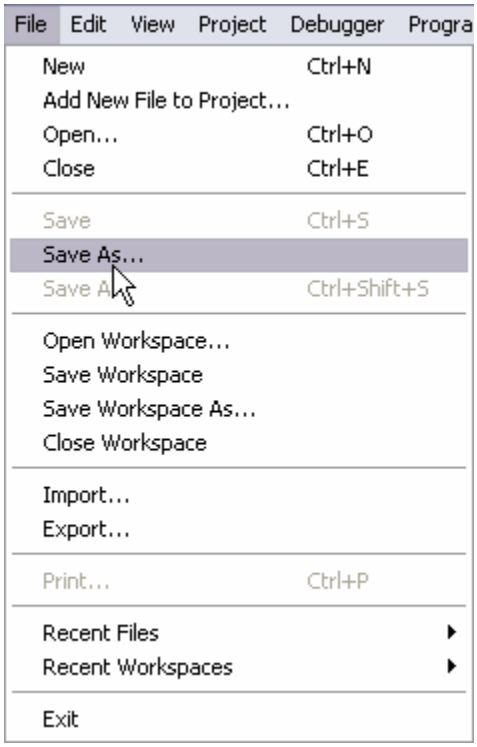

Figure 14: Saving a Source File

Step 12: Provide an appropriate filename. Be sure to add the .c extension since we are starting with the C compiler. Click "Save" (Figure 15).

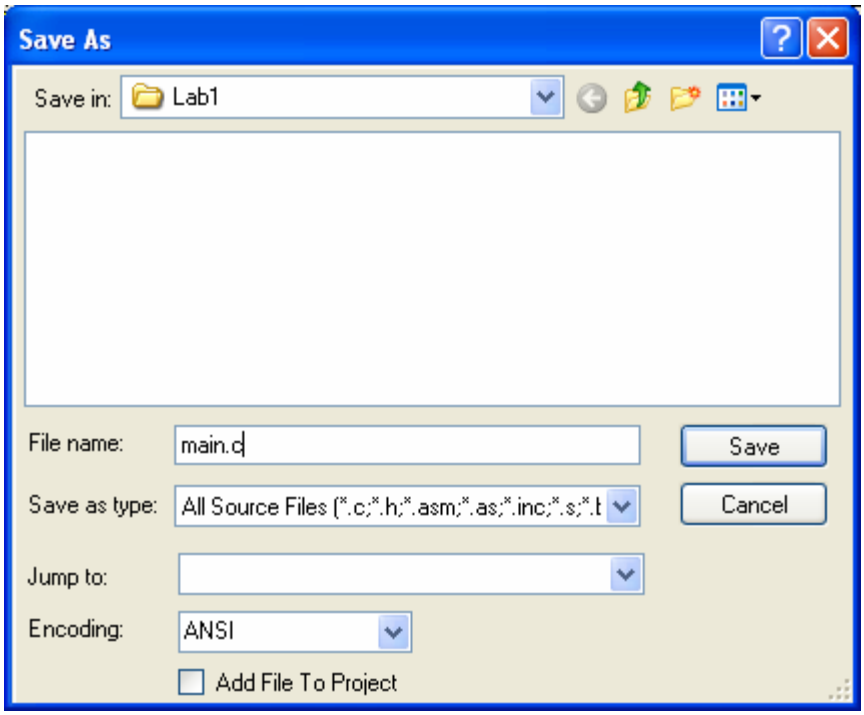

Figure 15: "Save As" Window

Your text editor, shown in Figure 16, should now be labeled with the name provided in the window of Figure 15.

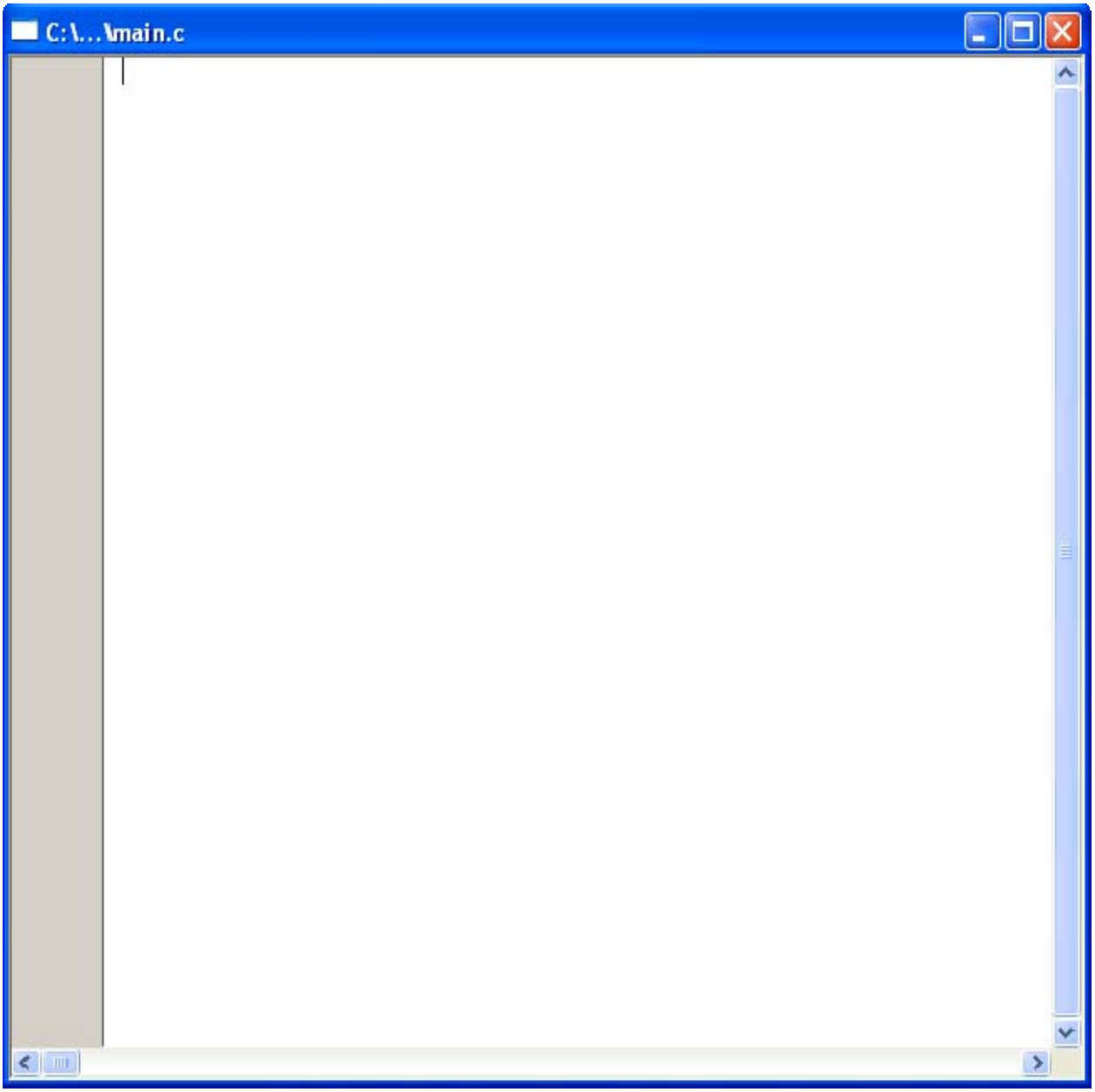

Figure 16: Renamed Source File

Step 13: Add the .c file to your project by right clicking on the "Source Files" folder in the .mcw window and selecting "Add Files…" (Figure 17). If you do not add the file to your project, you will not be able to build it. If this error occurs initially, you did not add the file correctly.

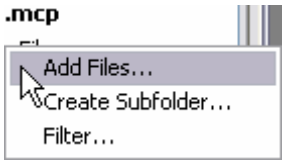

Figure 17: Adding the Source File to the Project

Step 14: Select your .c file from the possible files listed (Figure 18). In this case "main.c" and click "Open".

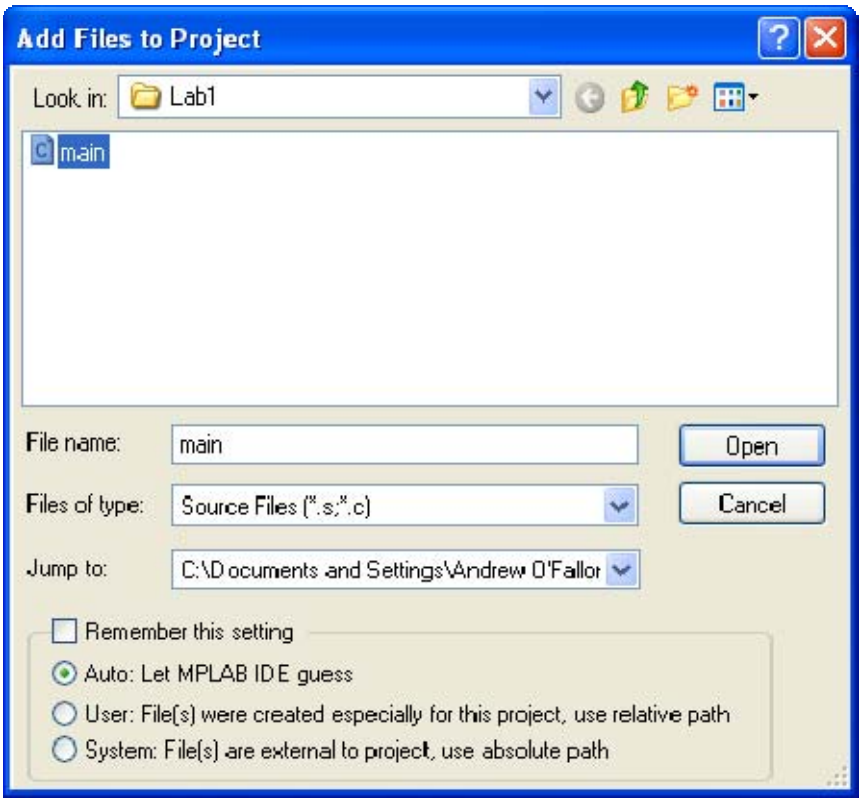

Figure 18: Selecting a File

You will now notice the file has been added to the project workspace (Figure 19). Once again, if the file is not correctly added, then you will not be able to compile it.

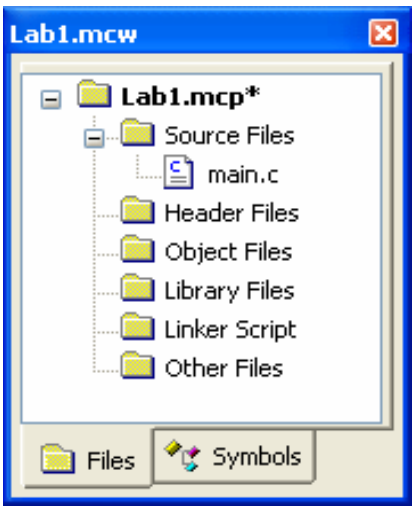

Figure 19: .mcw Window with Added Source File

Step 15: Develop your PIC32MX C program in the provided blank file.

```
/*****************************************************************************
 * Programmer: Andrew S. O'Fallon                                            *
 * Class: EE 234                                                             *
 * Programming Assignment: Lab 1.                                            *
 * Date: August 21, 2012                                                     *
 *                                                                           *
 * Description: This program reads the states of the on-board buttons of the *
 *              Cerebot MX4cK and writes the states to the on-board LEDs     *
 *                                                                           *
 *****************************************************************************/

#include <peripheral/ports.h> // PORTSetPinsDigitalIn (), PORTSetPinsDigitalOut (), PORTRead (), PORTWrit

// Yes, don't forget your prototypes
void setup_LEDs (void);
void setup_buttons (void);
unsigned int get_button_states (void);
void output_BTNs_to_LEDs (unsigned int button_states);

int main (void)
{
    unsigned int button_states = 0;

    setup_LEDs (); // Output pins
    setup_buttons (); // Input pins

    while (1) // Embedded programs run forever
    {
        // Get the state of the buttons
        button_states = get_button_states ();
        // Write the state of the buttons to the LEDs
        output_BTNs_to_LEDs (button_states);
    }

    return 0;
}
```

Figure 20: PIC32MX C Program

Lab 1 C Project Code

Copy or type the code from the link above into the provided editor shown in Figure 20.

Compiling/Assembling a Project:

Step 1: First, select "Debugger" -> "Select Tool" -> "Licensed Debugger" (Figure 21). If the power to your board is not on, you will not be able to select the "Licensed Debugger".

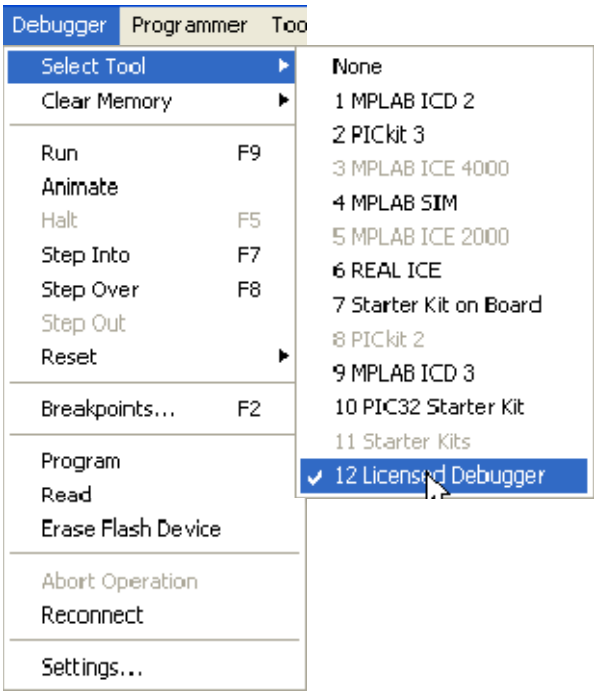

Figure 21: Selecting a Debugger Tool

Step 2: To compile your project, select "Project" -> "Build All" (Figure 22). The compiler will report any syntax errors at this point.

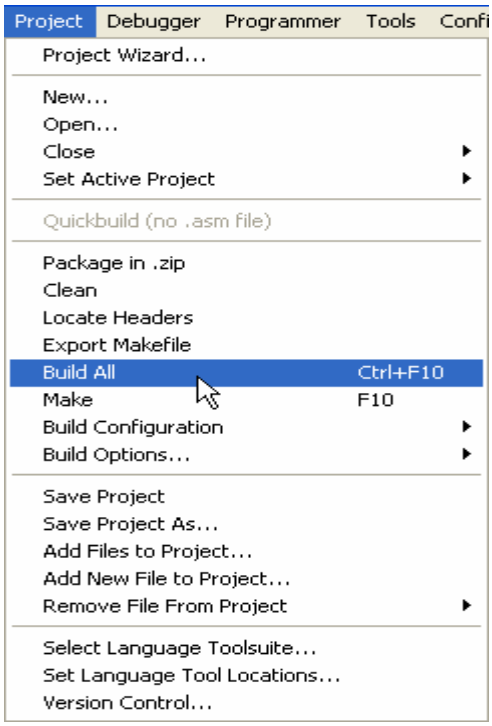

Figure 22: Building your Project

Step 3: Select "Debugger" -> "Program" (Figure 23) to program the microcontroller. Analyze your results with awe!

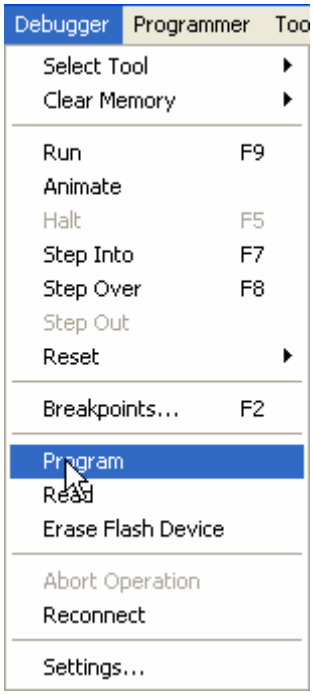

Figure 23: Programming your Microcontroller

Step 4: Select "Debugger" -> "Run" (Figure 24) to execute the program. Analyze your results with awe!

Figure 24: Running your Program

To understand the function of the program, push the two on-board buttons (BTN1 and BTN2) and see what happens. The onboard LEDS (LD1 and LD2) should turn on when the buttons are pushed and turn off when they are released.

Step 5: If you would like to stop your project from executing, select "Debugger" -> "Halt" (Figure 25).

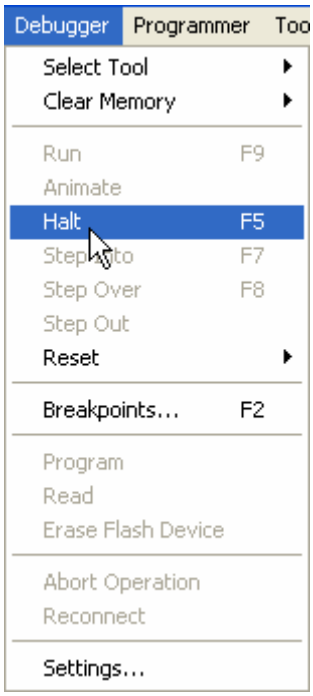

Figure 25: Halting the Execution of your Program

In the future consider setting up breakpoints and taking advantage of the in-circuit debugger.

After you have become somewhat familiar with the C code provided, move onto analyzing the disassembly listing for the code. To view the "Disassembly Listing" select "View" -> "Disassembly Listing" (Figure 26).

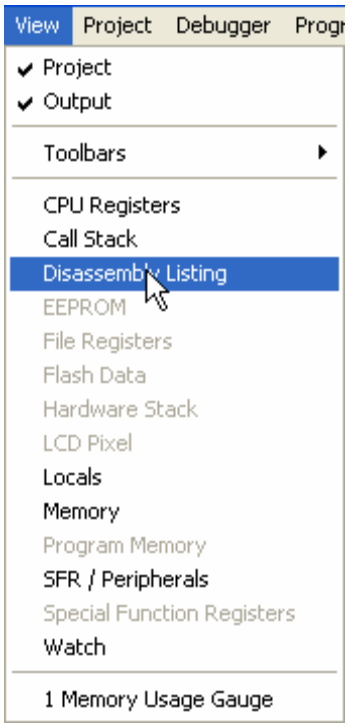

Figure 26: Selecting the Disassembly Listing

Try to familiarize yourself with some of the MIPS assembly instructions (represented in blue) shown in the listing (Figure 27). Also, familiarize yourself with some of the core/general purpose registers available. You may identify the registers because they are the operands to many of the MIPS instructions. If you would like to try to understand more about the MIPS instruction set and corresponding registers, please refer to the Quick Reference Guide. You may also find more in depth information about these instructions and registers by downloading and viewing the MIPS32 Volume II document. This may be downloaded at http://www.mips.com/products/architectures/mips32/. You will need to register to obtain the three volumes associated with the MIPS32 architecture.

As part of this lab experience you may also want to look at CPU registers, the call stack, and memory by selecting the appropriate window under the "View" menu.

```
---  C:\Documents and Settings\Andrew O'Fallon\Desktop\EE234\Lab1\main.c  ----------------------
1:                    /*****************************************************************************
2:                     * Programmer: Andrew S. O'Fallon                                            *
3:                     * Class: EE 234                                                             *
4:                     * Programming Assignment: Lab 1.                                            *
5:                     * Date: August 21, 2012                                                     *
6:                     *                                                                          *
7:                     * Description: This program reads the states of the on-board buttons of the  *
8:                     *              Cerebot 32MX4 and writes the states to the on-board LEDs      *
9:                     *                                                                          *
10:                    *****************************************************************************/
11:
12:                   #include <peripheral/ports.h> // PORTSetPinsDigitalIn (), PORTSetPinsDigitalOut (), PORTRead ()
13:
14:                   // Yes, don't forget your prototypes
15:                   void setup_LEDs (void);
16:                   void setup_buttons (void);
17:                   unsigned int get_button_states (void);
18:                   void output_BTNs_to_LEDs (unsigned int button_states);
19:
20:                   int main (void)
21:                   {
9D000008  27BDFFE0    addiu       sp,sp,-32
9D00000C  AFBF001C    sw          ra,28(sp)
9D000010  AFBE0018    sw          s8,24(sp)
9D000014  03A0F021    addu        s8,sp,zero
22:                      unsigned int button_states = 0;
9D000018  AFC00010    sw          zero,16(s8)
23:
24:                      setup_LEDs (); // Output pins
9D00001C  0F400013    jal         0x9d00004c
9D000020  00000000    nop
```

Figure 27: Disassembly Listing

VII. Questions:

Please include the answers to the following questions in your formal lab write-up!

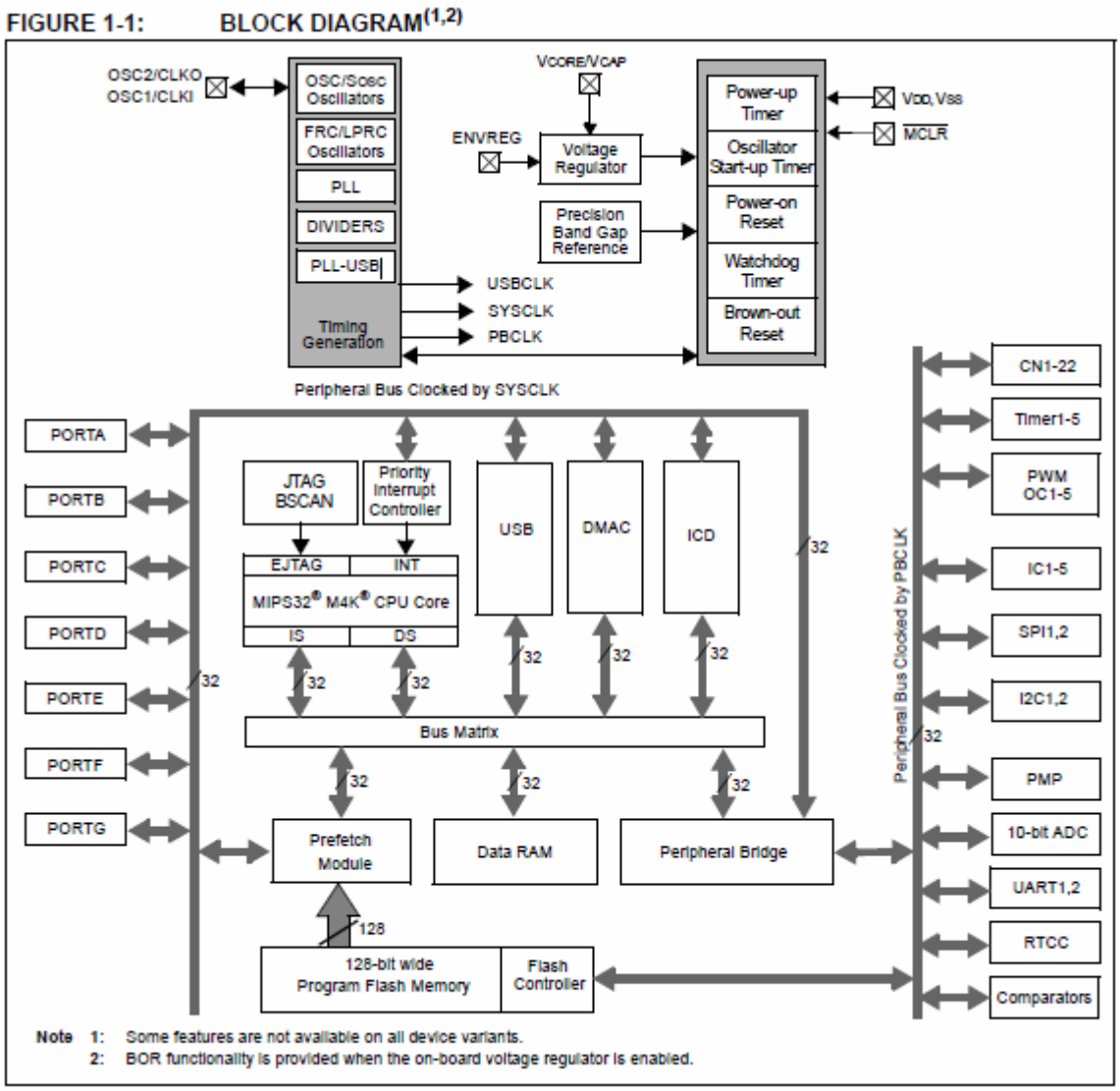

Figure 28: Block Diagram of PIC32 MIPS32 MCU Architecture, p. 21 of PIC32MX4 Data Sheet[2]

[2] PIC32MX4 Data Sheet

1. Pick out one C function from the example program provided and explain its operation. You can find out more about some of the included library functions by opening the header available with the PIC32. Follow: C:\Program Files\Microchip\MPLAB C32 Suite\pic32-libs\include\peripheral or a similar path on your machine.

2. Pick out one MIPS assembly instruction from the disassembly listing corresponding to the provided C program and explain its operation. If you would like to try to understand more about the MIPS instruction set and corresponding registers, please refer to the Quick Reference Guide. You may also find more in depth information about these instructions and registers by downloading and viewing the MIPS32 Volume II document. This may be downloaded at http://www.mips.com/products/architectures/mips32/.

3. In the program you notice that we refer to IOPORT_A and IOPORT_B. Which reference document(s) indicate(s) that PORTB is connected to the LEDs? For example is this explained in the PIC32MX4 Data Sheet and/or the Cerebot MX4cK reference manual? Think logically about what would be contained in the manuals!

4. Explain the general process that is followed when an instruction is executed. Please refer to Figure 28 and specify the hardware components, registers, and busses that are used in the process. In which of the memory blocks are instructions and data stored? For this question you will have some gaps in your understanding as of now, that is fine!

5. How many bytes of Flash memory are required for the provided program? How many bytes of data memory (SRAM) are required for the program? Recall that you may find these numbers when you build the program. See Figure 29.

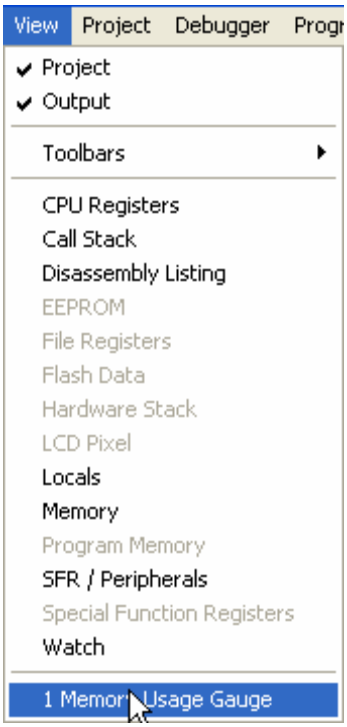

Figure 29: Memory Usage

6. Describe the purpose of the ALU. Be as specific as possible. Try to rely on some of the knowledge you gained related to ALU's from EE 214 (Design of Logic Circuits) for this question. I would also like for you to try to explore what some of the blocks in the diagram represent by searching the web!

7. Any comments you would like to make to spice-up or improve this lab for the future!