

Lab 1: Instructions

Microchip MPLAB v8.92 Integrated Development Environment

Be prepared to demonstrate your Lab 1 circuit on or before:

MW Section – Mon, Feb 5

TTh Section – Tue, Feb 6

There will also be a written assignment due on the same days.

This lab familiarizes the student with Microchip's MPLAB IDE v8.92 software, and the general architecture, registers, and instruction set of the PIC16F877. We will use the Lab 1 circuit and the solderless breadboard design that is shown in the document [lab01_slides.pdf](#) available on Blackboard. Please follow the instructions below to test your circuit and familiarize yourself with the MPLAB development system.

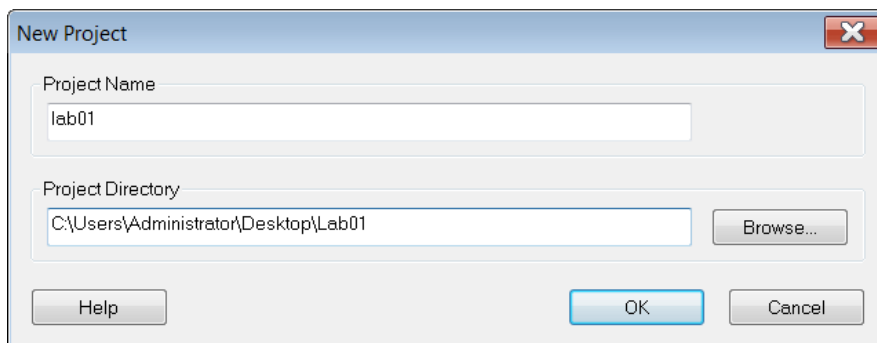
Setup

1. Carefully build the circuit as shown in the schematic and pictures in lab01_slides.pdf. Double-check your circuit before applying power. The LM7805 and the PIC chip may be slightly warm to the touch, but not hot. If either is hot, something is wrong, and the chip may already be damaged. Replacement chips cost \$8.00. Do not connect the PICKit 3 USB connector to your PC until you have installed the MPLAB software in the next step.
2. If you will be working on your own PC, download [MPLAB IDE v.8.92.zip](#) and install MPLAB. This software is already installed on the PCs in SH 309. While installing, use the **Custom** setup type, and make sure that you select the **PICKit 3** features. Also, you must be connected to the internet while installing.
3. After installing MPLAB, connect the PICKit USB connector to your PC, and connect the power supply to your solderless breadboard (also called a prototype board or protoboard).

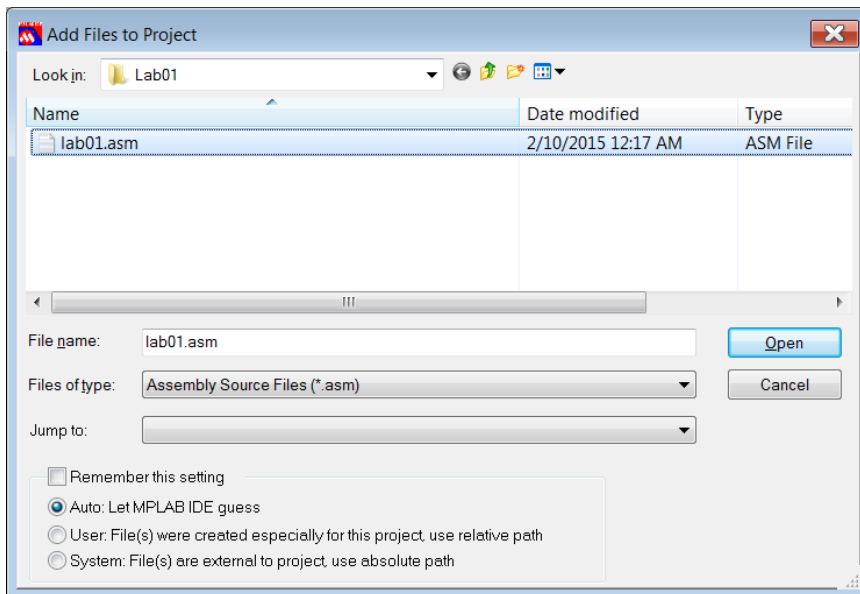
Create a Project

4. Create a directory on the PC in which to store all of your assembly code for your lab assignments. Under this directory, create a subdirectory named Lab01.
5. Copy the lab01.asm from the course web site to your Lab01 directory. The file is zipped.

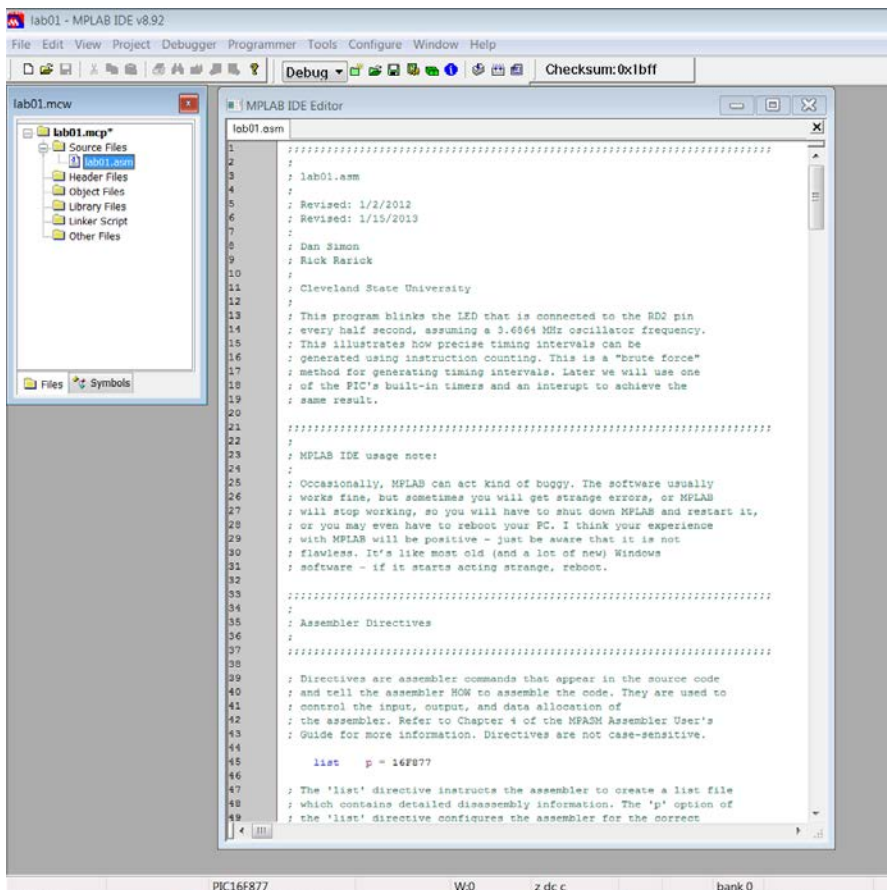
6. Start the MPLAB software. Select the **[Project → New]** menu item and create a new project called lab01 in your Lab01 directory.



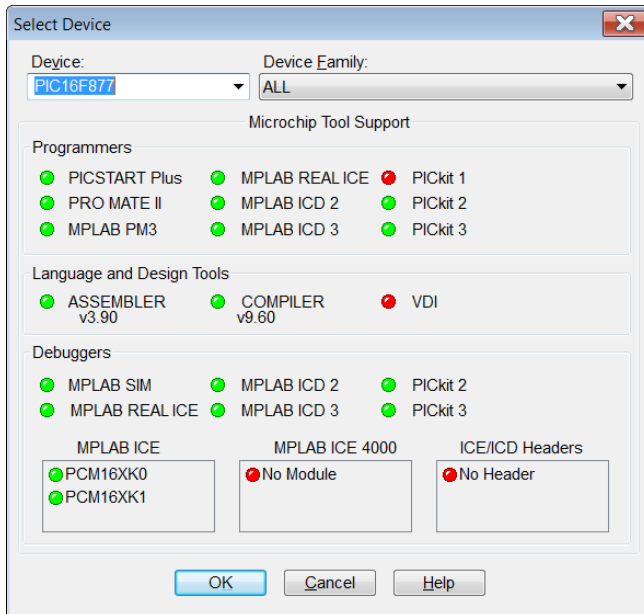
7. Select the **[Project → Add Files to Project]** menu item. Add lab01.asm to your project.



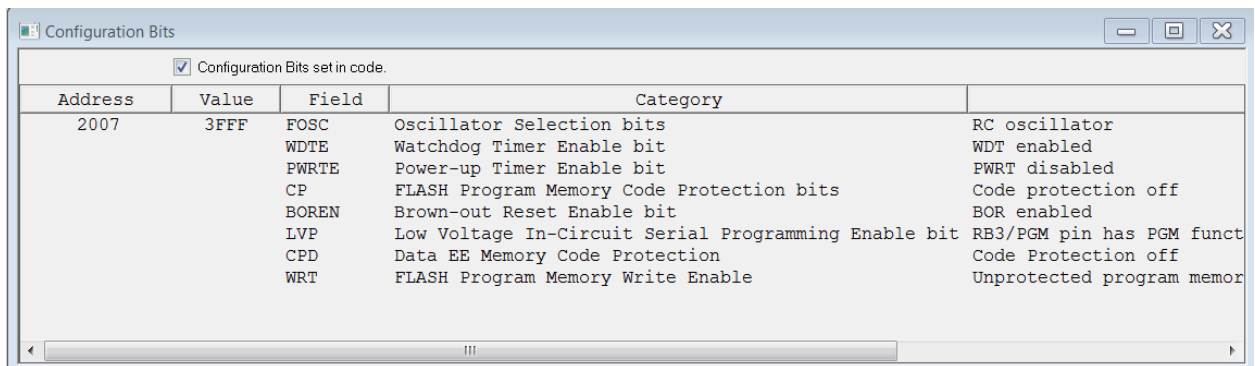
8. Select the **[View → Project]** menu item. Double click on lab01.asm, which is under the Source Files tab, to see the source code in the editor window.



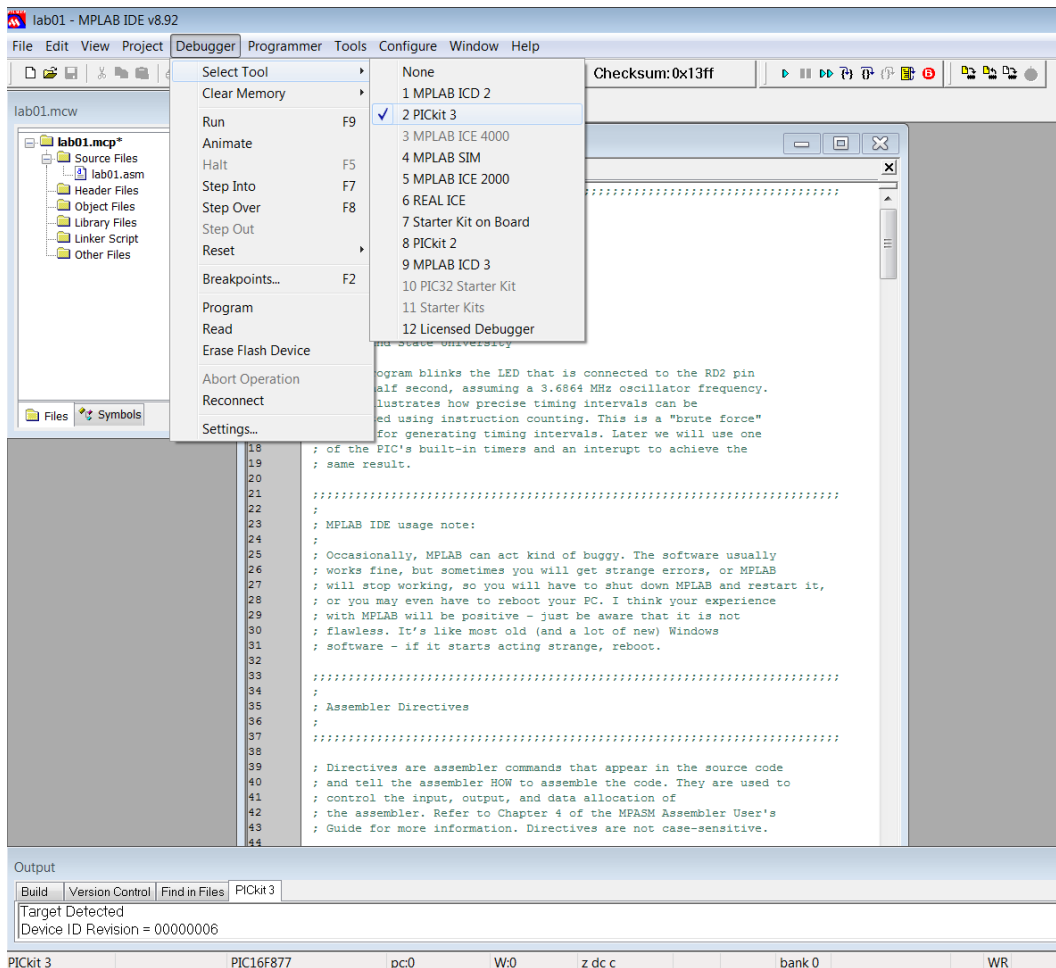
9. Select the **[Configure → Select Device]** menu item. When the **Select Device** window comes up, select the PIC16F877 as your device.



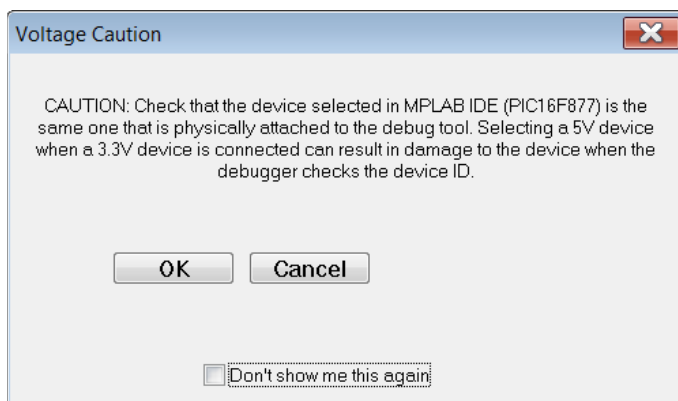
10. Select the **[Configure → Configuration Bits]** menu item. Check the “Configuration Bits set in code” check box.



11. Select the **[Debugger → Select Tool → PICKit 3]** menu item. An **Output** window should open, and you should see the “Target Detected” message in the **Output** window. If not, there is probably something wrong with your hardware or wiring.



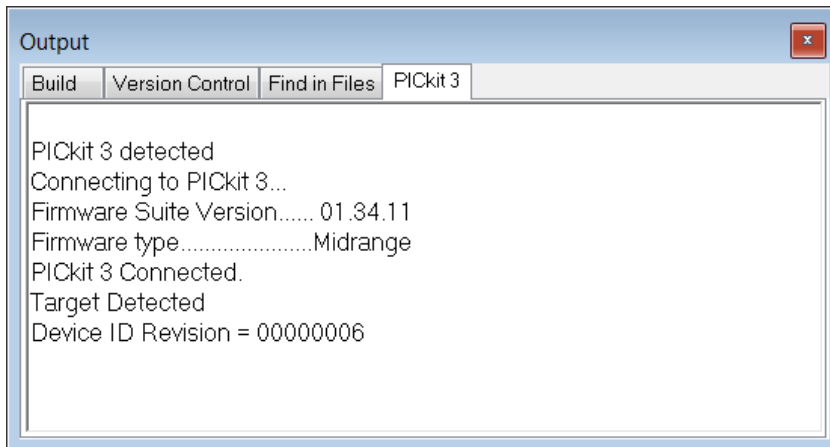
If you receive the following caution, click the checkbox then click “OK”.



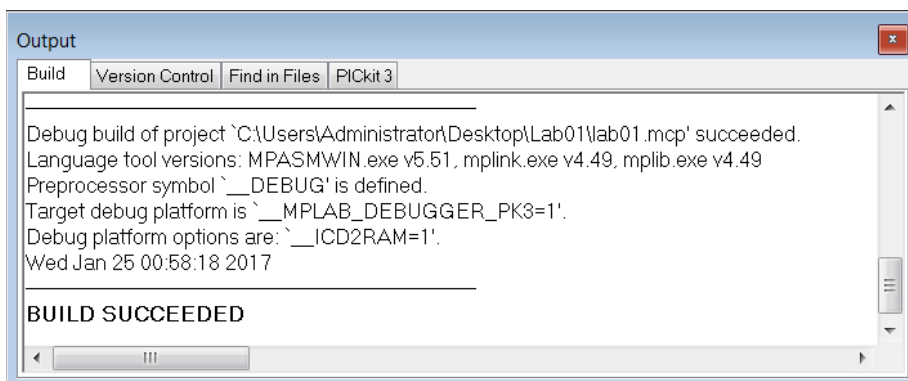
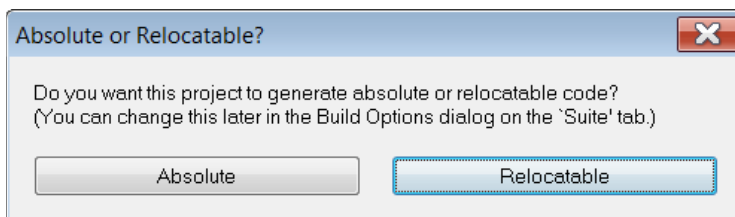
12. Select the **[Project → Save Project]** menu item to save your project.

Run the Project

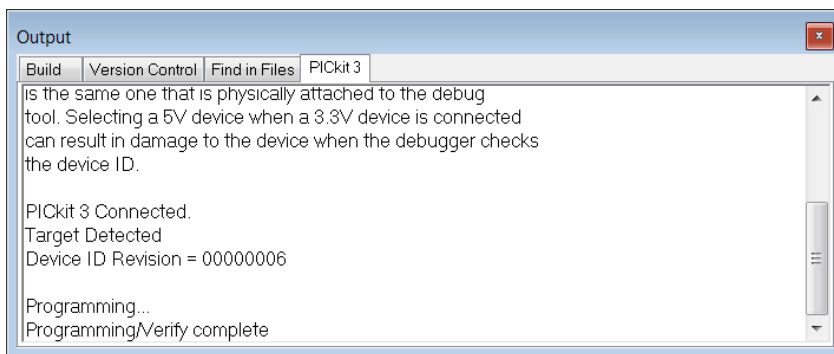
13. Select the **[Debugger → Reconnect]** menu item. Your PICKit may already be connected, but execute this command anyway to be sure. You should see a “PICKit 3 Connected” message in the PICKit tab of output window. If you do not see the **Output** window, you can open it by selecting the **[View → Output]** menu item. If you do not see the “PICKit 3 Connected” message, something is probably wrong with your circuit or connections.



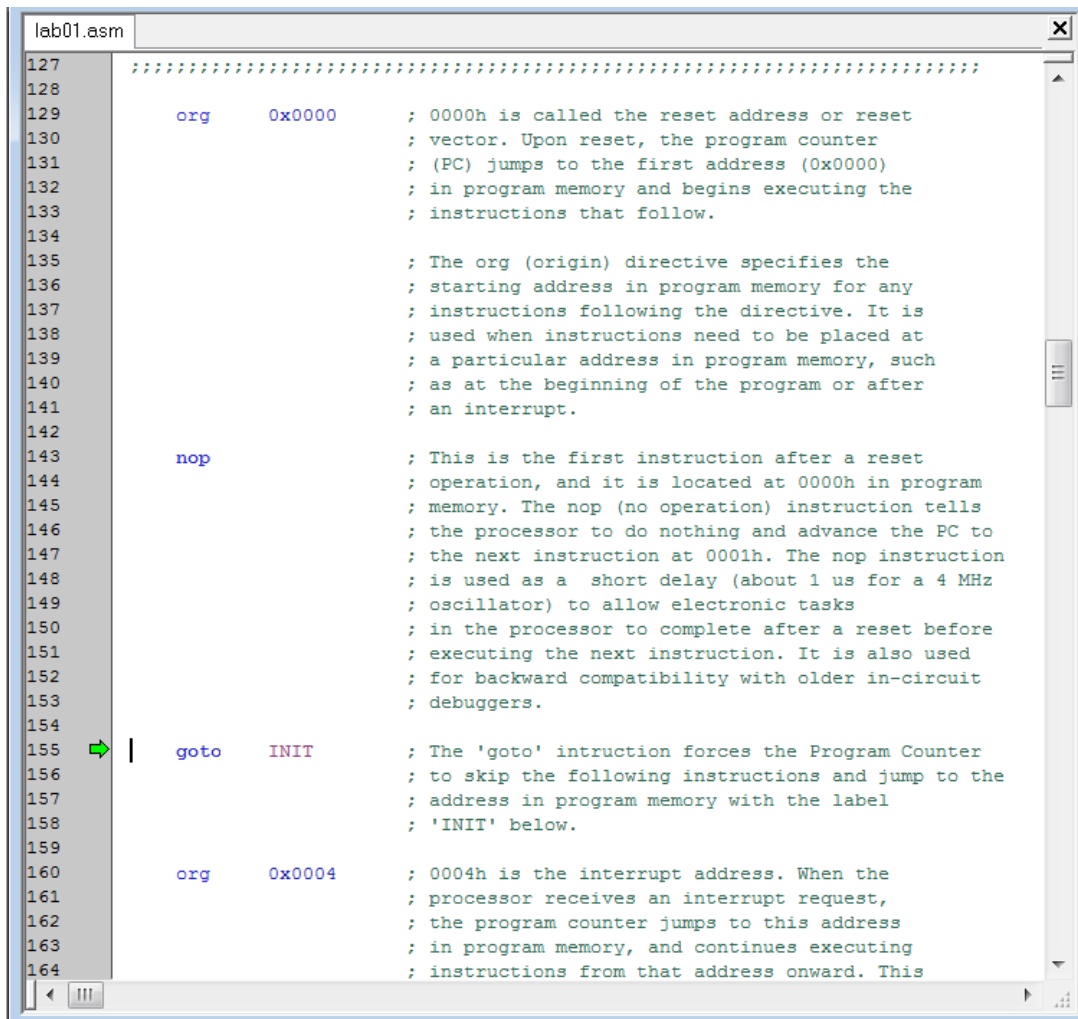
14. Select the **[Project → Build All]** menu item. Specify “Absolute” when you are prompted for your link option. You should see several messages followed by the “Build Succeeded” message in the Output window. Note that there are icons on the MPLAB toolbar for common operations such as **[Build All]**. If you do not get the “Build Succeeded” message, something probably was accidentally changed in the code, since the lab01.asm code has been run hundreds of times.



15. After you see the “Build Succeeded” message, scroll through the Build tab on the Output window. You will see a line that begins with “Message[302]”. Double-click the “Message[302]” text to see the line in the assembly code that caused this message. (You can suppress this assembler message, but it’s better not to.)
16. Read the information in the MPASM User’s guide on “Assembler Messages” for a brief explanation of the “Message[302]” message. To open the MPASM User’s Guide, select the [\[Help → Topics\]](#) menu item; then double-click the “MPASM Assembler” text; then click the “MPASM Assembler” book in the left pane; then click the “Errors, Warnings, Messages, and Limitations” link in the right pane; then click the “Assembler Messages” link in the right pane. There is also a copy of the MPASM User’s Guide in PDF format on the course website, and it may be easier to read and search the PDF.
17. You will also see several warning messages in the Output window that say, “Found label after column 1.” The message is generated whenever a label does not begin in column 1 (See the “label” directive in the MPASM User’s Guide). If you double-click on the warning message, MPLAB will take you directly to the line in your ASM file that caused the warning. Correct the problems that caused these warning messages, rebuild the project, and verify that these warnings are gone.
18. Select the [\[Debugger → Program\]](#) menu item or select the “Program” icon. You should get a “Programming/Verify complete” message in the PICKit3 tab of the Output window.



19. Select the **[Debugger → Reset → Processor Reset]** menu item. The green program counter arrow should appear next to the “nop” instruction in the ASM file (or maybe one instruction later). The arrow indicates which instruction will be executed next.

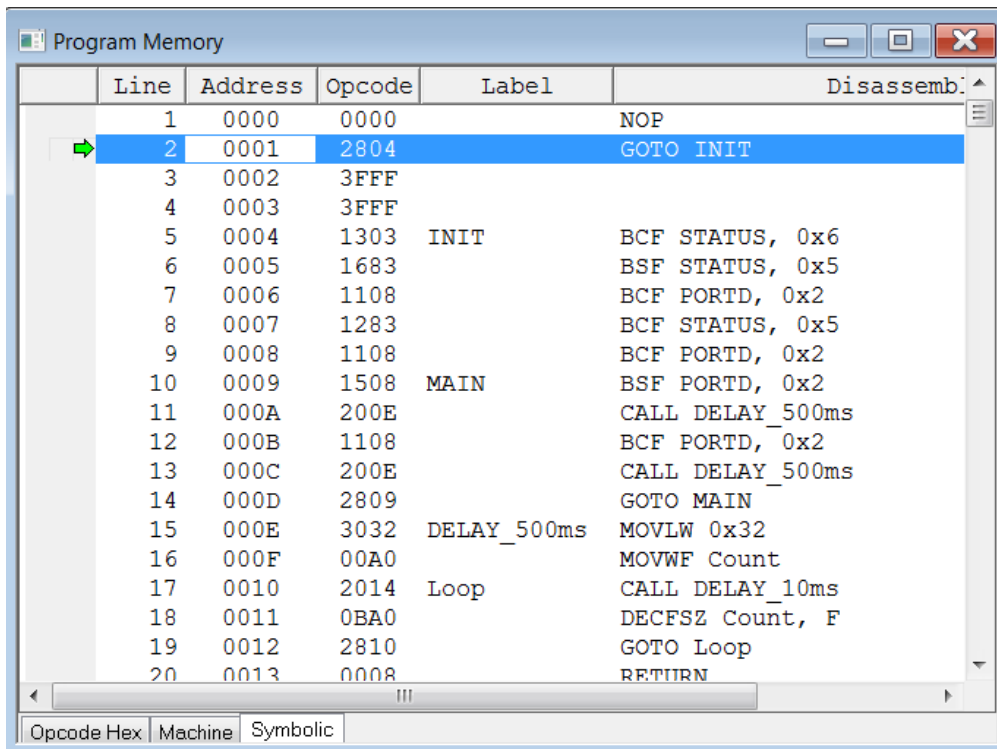


```

127  //////////////////////////////////////
128
129      org    0x0000    ; 0000h is called the reset address or reset
130                      ; vector. Upon reset, the program counter
131                      ; (PC) jumps to the first address (0x0000)
132                      ; in program memory and begins executing the
133                      ; instructions that follow.
134
135                      ; The org (origin) directive specifies the
136                      ; starting address in program memory for any
137                      ; instructions following the directive. It is
138                      ; used when instructions need to be placed at
139                      ; a particular address in program memory, such
140                      ; as at the beginning of the program or after
141                      ; an interrupt.
142
143      nop              ; This is the first instruction after a reset
144                      ; operation, and it is located at 0000h in program
145                      ; memory. The nop (no operation) instruction tells
146                      ; the processor to do nothing and advance the PC to
147                      ; the next instruction at 0001h. The nop instruction
148                      ; is used as a short delay (about 1 us for a 4 MHz
149                      ; oscillator) to allow electronic tasks
150                      ; in the processor to complete after a reset before
151                      ; executing the next instruction. It is also used
152                      ; for backward compatibility with older in-circuit
153                      ; debuggers.
154
155      goto    INIT     ; The 'goto' instruction forces the Program Counter
156                      ; to skip the following instructions and jump to the
157                      ; address in program memory with the label
158                      ; 'INIT' below.
159
160      org    0x0004    ; 0004h is the interrupt address. When the
161                      ; processor receives an interrupt request,
162                      ; the program counter jumps to this address
163                      ; in program memory, and continues executing
164                      ; instructions from that address onward. This

```

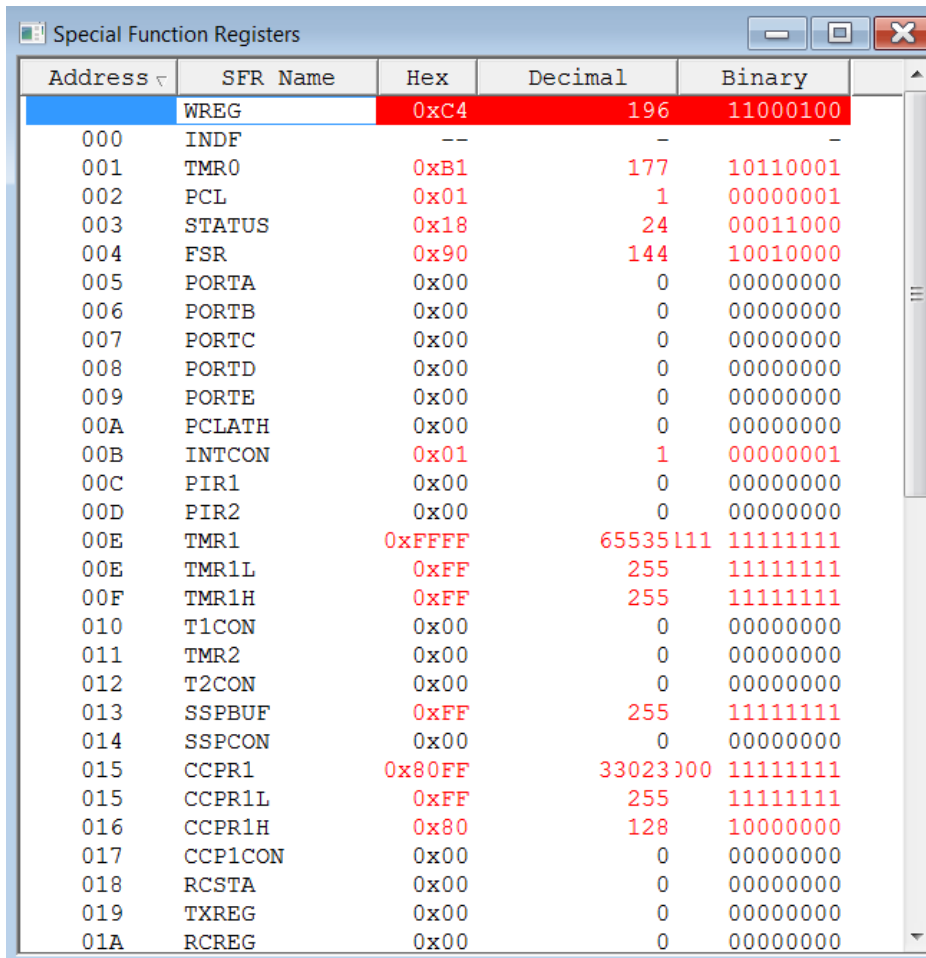

20. Select the **[View → Program Memory]** menu item. This allows you to view the PIC's program memory, and the green arrow will be pointing to the "nop" instruction at address 0x0000 (or maybe one instruction later).



Line	Address	Opcode	Label	Disassembly
1	0000	0000		NOP
2	0001	2804		GOTO INIT
3	0002	3FFF		
4	0003	3FFF		
5	0004	1303	INIT	BCF STATUS, 0x6
6	0005	1683		BSF STATUS, 0x5
7	0006	1108		BCF PORTD, 0x2
8	0007	1283		BCF STATUS, 0x5
9	0008	1108		BCF PORTD, 0x2
10	0009	1508	MAIN	BSF PORTD, 0x2
11	000A	200E		CALL DELAY_500ms
12	000B	1108		BCF PORTD, 0x2
13	000C	200E		CALL DELAY_500ms
14	000D	2809		GOTO MAIN
15	000E	3032	DELAY_500ms	MOVLW 0x32
16	000F	00A0		MOVWF Count
17	0010	2014	Loop	CALL DELAY_10ms
18	0011	0BA0		DECFSZ Count, F
19	0012	2810		GOTO Loop
20	0013	0008		RETURN

21. Select the **[Debugger → Run]** menu item. Your LED should be blinking: on for a half-second, and off for a half-second. You should notice a green bar scrolling at the bottom of the MPLAB window, which indicates that the PIC is running. You should notice the message "Running Target" in the Output window.
22. Select the **[Debugger → Halt]** menu item (F5). After you do this, the green program counter arrow will be pointing to whatever instruction the PIC was at when it stopped.
23. Select **[Debugger → Reset → Processor Reset]**.

24. Select **[View → Special Function Registers]**. This new window shows the value of each SFR. Note that you can sort them in the window by clicking the Address or SFR Name column heading. Scroll until the STATUS register is in view. Note that you can right click on the column headings and choose to view the registers as hex, decimal, binary, or character.



Address	SFR Name	Hex	Decimal	Binary
000	WREG	0xC4	196	11000100
001	INDF	--	--	--
002	TMR0	0xB1	177	10110001
003	PCL	0x01	1	00000001
004	STATUS	0x18	24	00011000
005	FSR	0x90	144	10010000
006	PORTA	0x00	0	00000000
007	PORTB	0x00	0	00000000
008	PORTC	0x00	0	00000000
009	PORTD	0x00	0	00000000
00A	PORTE	0x00	0	00000000
00B	PCLATH	0x00	0	00000000
00C	INTCON	0x01	1	00000001
00D	PIR1	0x00	0	00000000
00E	PIR2	0x00	0	00000000
00F	TMR1	0xFFFF	65535	11111111
010	TMR1L	0xFF	255	11111111
011	TMR1H	0xFF	255	11111111
012	T1CON	0x00	0	00000000
013	TMR2	0x00	0	00000000
014	T2CON	0x00	0	00000000
015	SSPBUF	0xFF	255	11111111
016	SSPCON	0x00	0	00000000
017	CCPR1	0x80FF	33023	10000000
018	CCPR1L	0xFF	255	11111111
019	CCPR1H	0x80	128	10000000
01A	CCP1CON	0x00	0	00000000
01B	RCSTA	0x00	0	00000000
01C	TXREG	0x00	0	00000000
01D	RCREG	0x00	0	00000000

Run the Project in Animation Mode

25. Select the **[Debugger → Animate]** menu item. Observe the code running one instruction at a time, and the value of the STATUS register changing. Note that after a short time, the program counter begins cycling in the InnerLoop loop.
26. Select **[Debugger → Halt]**.

Run the Project in Single-Step Mode

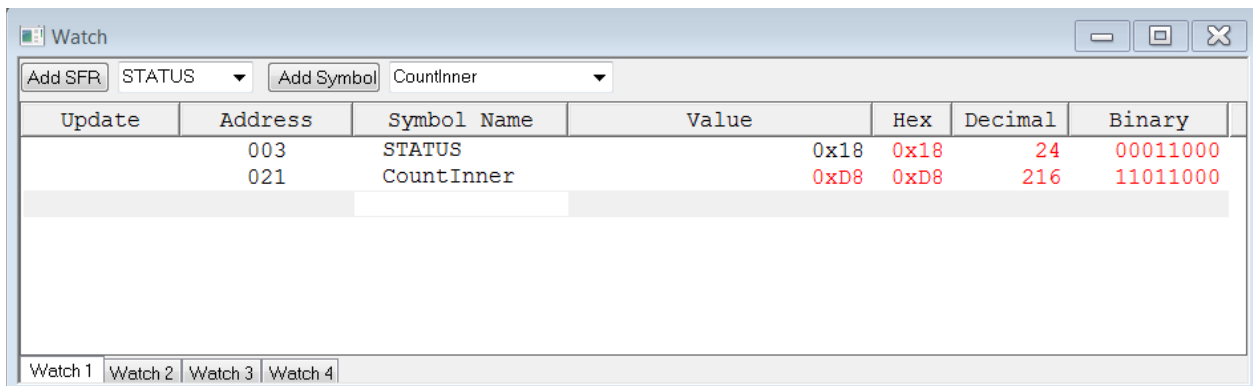
27. Press the F7 key to step through the program one instruction at a time. Notice the PCL register counting in the SFR Window and in the MPLAB status bar at the bottom of the MPLAB window. The PCL register is the low byte of the program counter and shows what address in memory the microcontroller is going to execute next. The status bar at the bottom

of MPLAB shows the program counter (PC), which is the same as PCL. The status bar also shows the processor (PIC16F877), the value of the W register, the active data memory bank, and couple of other important items.

28. Close the Program Memory and SFR windows.

29. Select **[View → Watch]**.

30. Use the drop down menu and select the STATUS register. Click “Add SFR” to add the STATUS register to the watch window. Similarly, use the “Add Symbol” button to add the CountInner register to the window. Note that you can right click on the column headings in the watch window and choose to view the registers as hex, decimal, binary, or character.



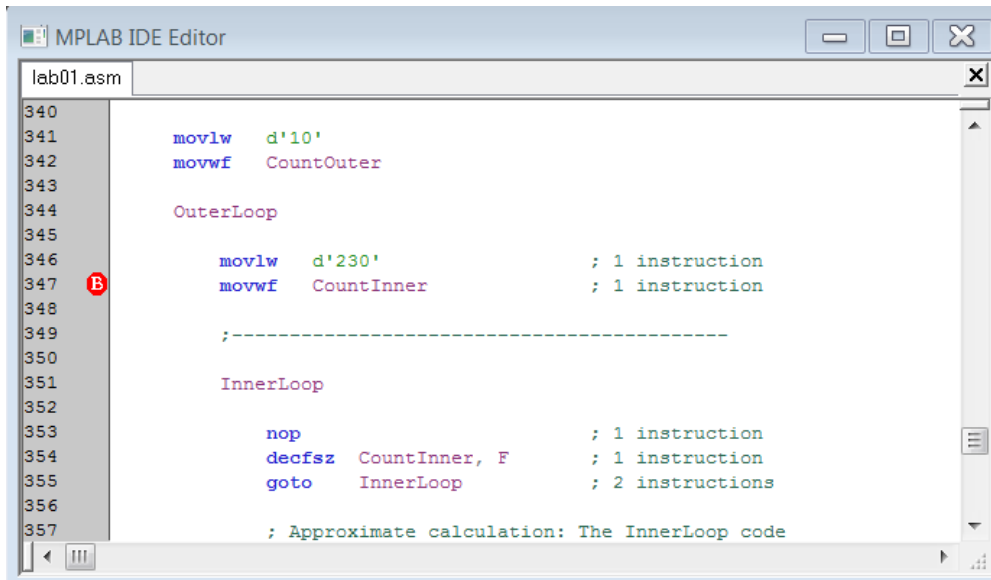
The screenshot shows the MPLAB Watch window. At the top, there are buttons for 'Add SFR' and 'Add Symbol', and dropdown menus for 'STATUS' and 'CountInner'. Below these is a table with columns: Update, Address, Symbol Name, Value, Hex, Decimal, and Binary. The table contains two rows: one for STATUS at address 003 with value 0x18 (decimal 24, binary 00011000) and one for CountInner at address 021 with value 0xD8 (decimal 216, binary 11011000). At the bottom, there are tabs for Watch 1, Watch 2, Watch 3, and Watch 4.

Update	Address	Symbol Name	Value	Hex	Decimal	Binary
	003	STATUS	0x18	0x18	24	00011000
	021	CountInner	0xD8	0xD8	216	11011000

31. Continue stepping through the program with the F7 key, and watch the CountInner variable change its value in the Watch window.

Breakpoints

32. Find the “movwf CountInner” instruction in the ASM code window. Double click on the left margin adjacent to the instruction. Note that a breakpoint symbol appears in the margin. You can add and remove breakpoints by double clicking.



```
lab01.asm
340
341     movlw    d'10'
342     movwf    CountOuter
343
344     OuterLoop
345
346     movlw    d'230'           ; 1 instruction
347     movwf    CountInner      ; 1 instruction
348
349     ;-----
350
351     InnerLoop
352
353     nop                ; 1 instruction
354     decfsz    CountInner, F    ; 1 instruction
355     goto      InnerLoop      ; 2 instructions
356
357     ; Approximate calculation: The InnerLoop code
```

33. Select **[Debugger → Run]**. Note that program execution halts at the breakpoint (or one instruction later). At this point you can inspect variables in the watch window, continue execution, step through the program, or do any other debugging that you need to do.
34. Select **[Debugger → Breakpoints]**. This opens the **Breakpoints** window, which allows you to remove, enable, or disable breakpoints.
35. Close the **Breakpoints** window.

Close the Project and Exit

36. Select **[Project → Close]** to close your project.
37. Close MPLAB.
38. Save the files from your Lab01 directory to a secure storage medium so that you don't lose your work.