

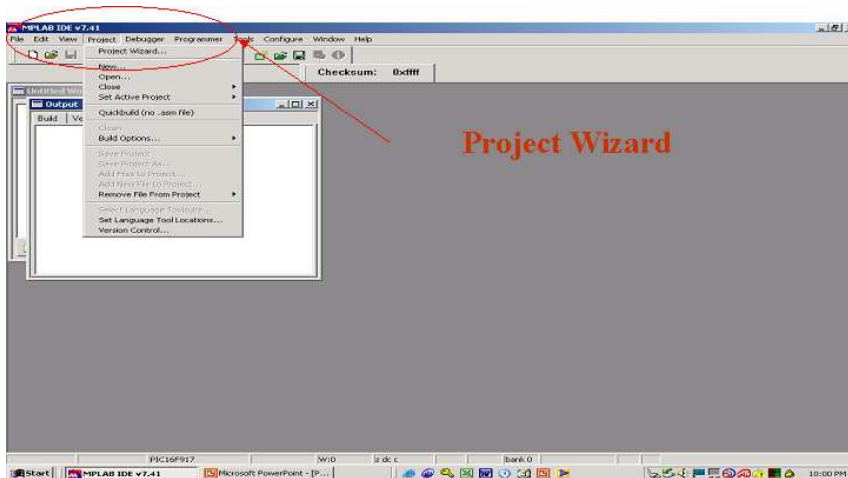
MICROCHIP Assembly Language Experiments using MPLAB Simulator

The following lab consists of several experiments involving MPLAB and MPLAB SIM. You need to install the EASYPIC folder contained on the class distributed CD onto the C: drive of your PC. The following pictorials will lead you through the lab setup. Once in setup you will select different five different assembly language routines within the fold C:\EASYPIC\ASSEMBLANG to simulate. For each you will single step through the simulator and answer the appropriate questions. The purpose is to familiarize you with MPLAB environment while conducting experiments with various Microchip assembly language constructs.

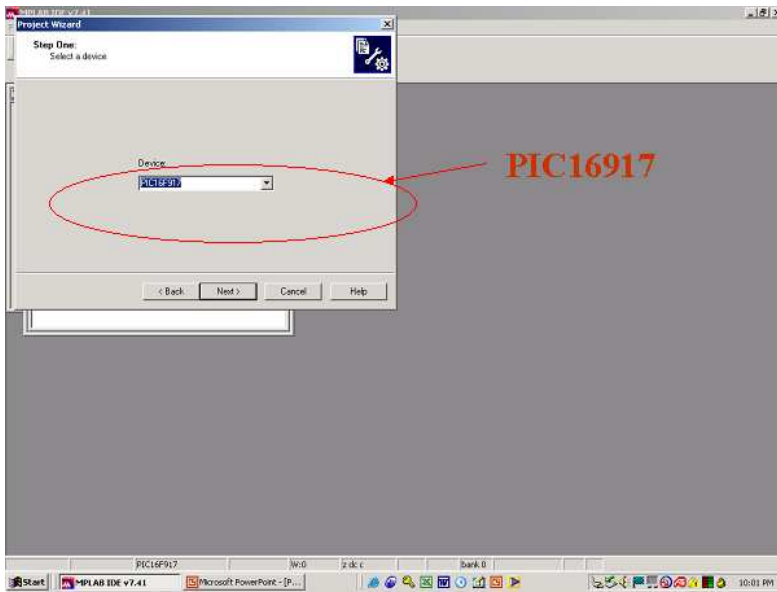
1. Step 1—Double click on MPLAB ICON



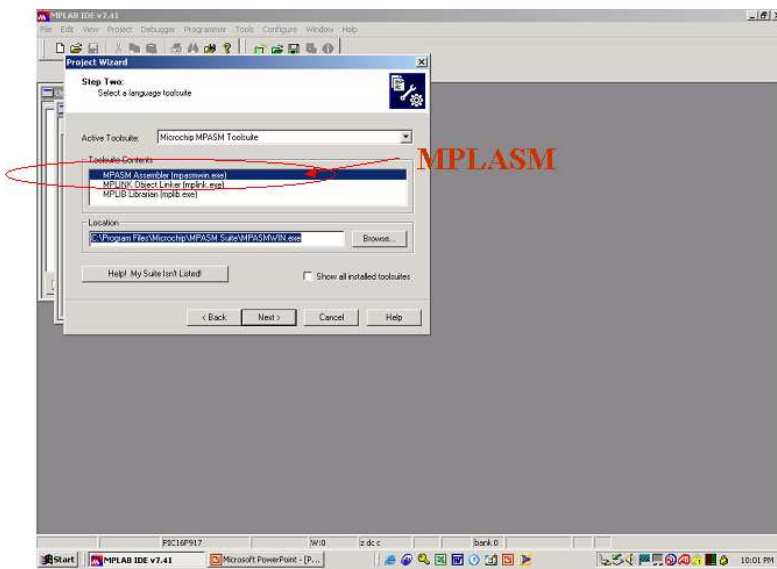
2. Step 2- Invoke Project Wizard



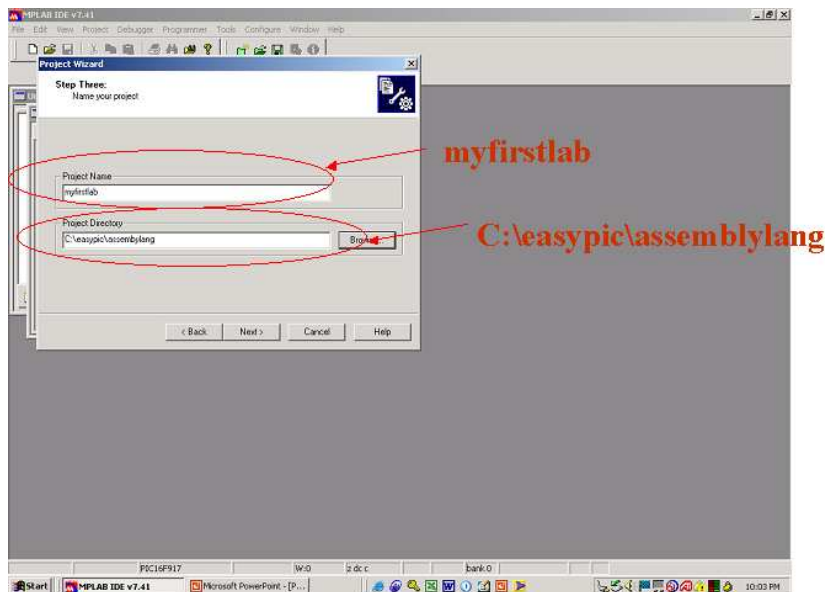
3. Step 3 Select the particular MICROCHIP Microcontroller PIC16917



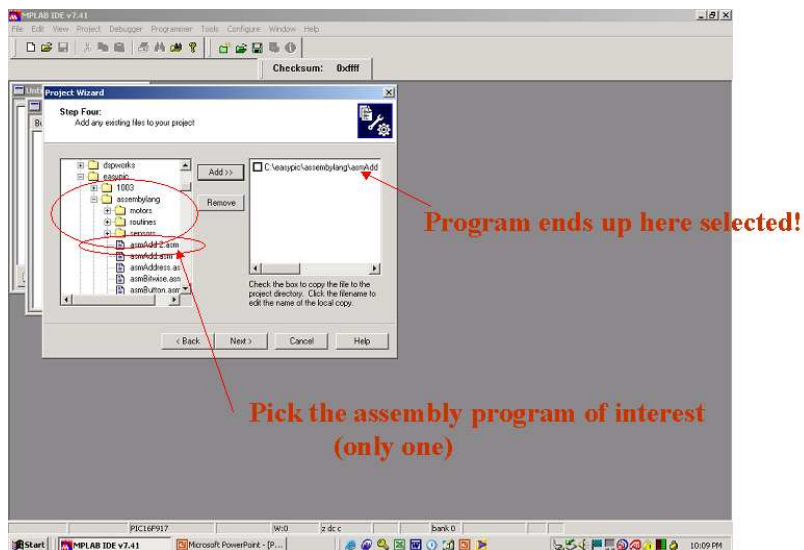
4. Step 4 – Select language suite MPLASM-Assembly Language



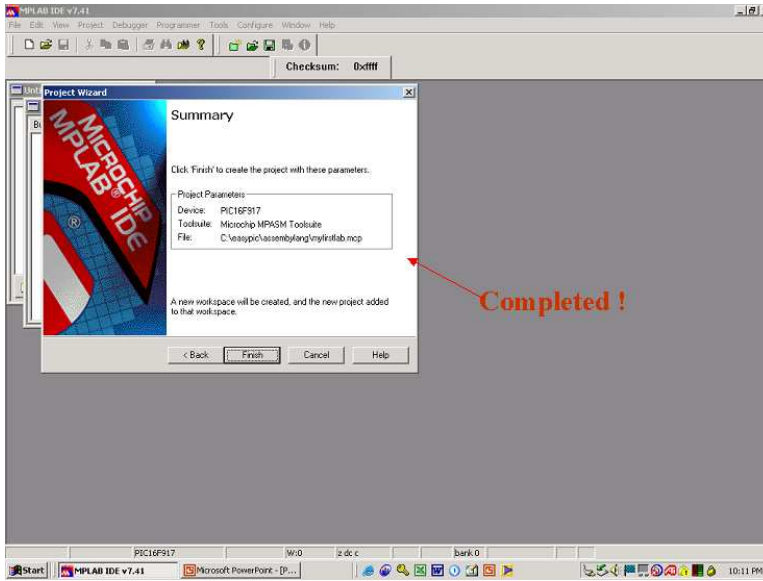
5. Step 5—Name project “myfirstlab’ and browse to C:\easypic\assemblylang



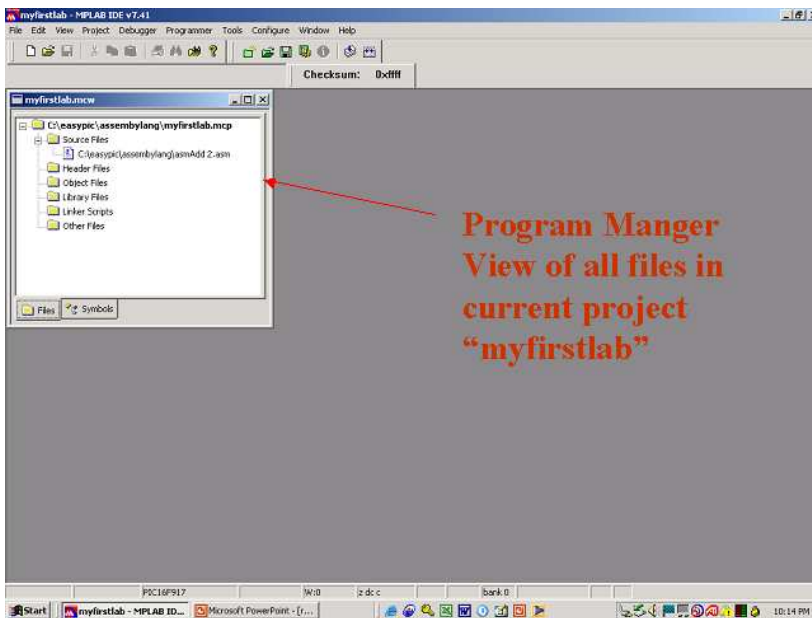
6. Step 6- Pick the .ASM File from folder ASMADD2 (adding two values)



7. Step 7- Finish project Wizard

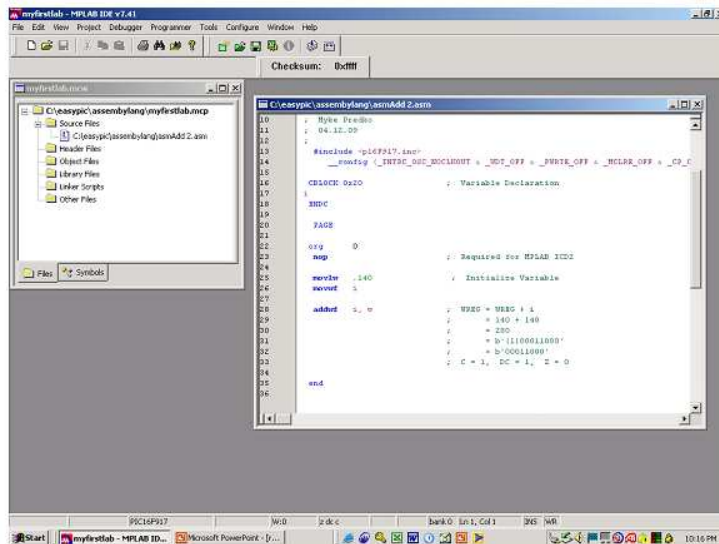


8. Step 8— examine all files contained in project

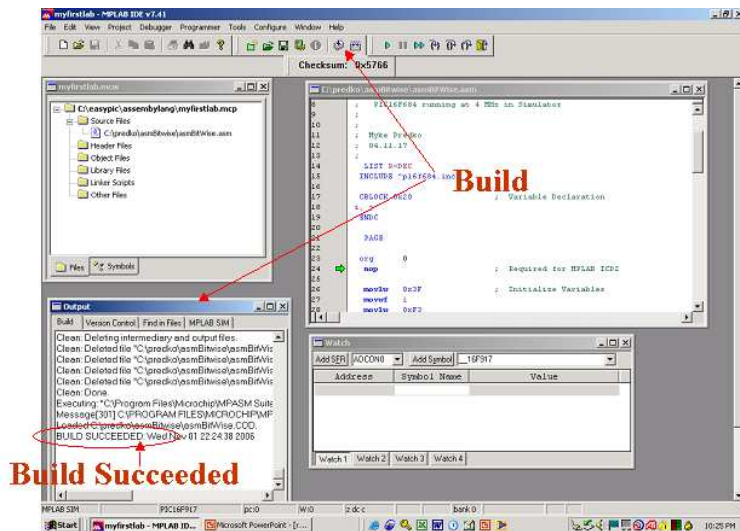


9. Step 9— View Assembly source by double clicking on .asm file name in program Manager View

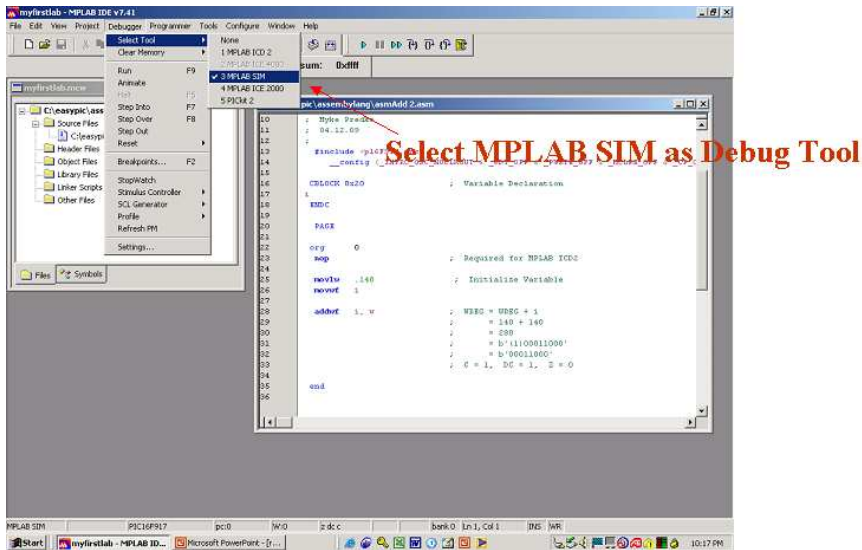
Double Click on Source code to view



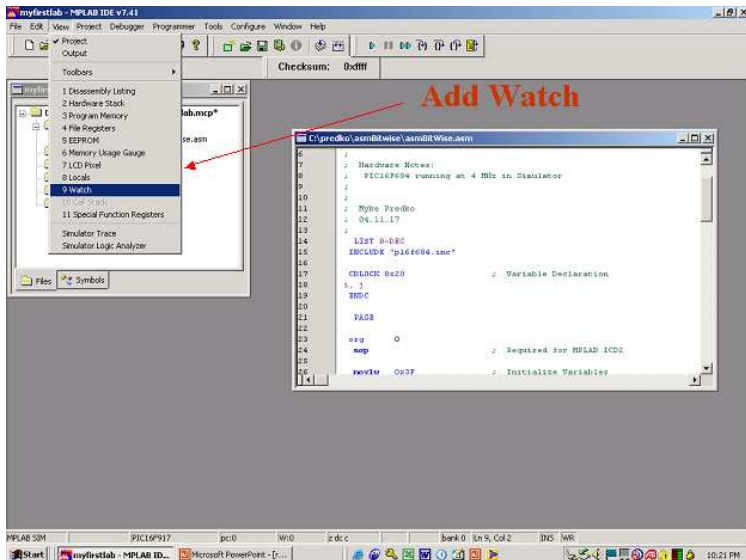
10. Step 10- invoke Build ICON and check that build succeeded



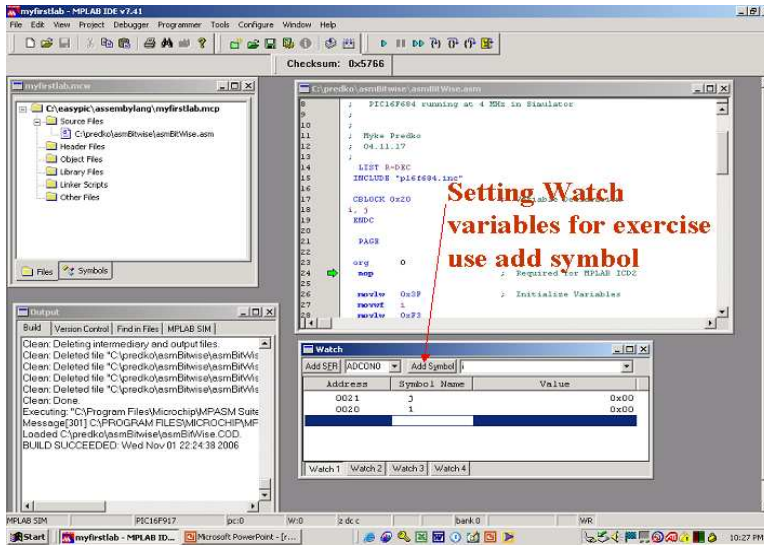
11. Step 11—Select MPLAB SIM as Debugger Tool



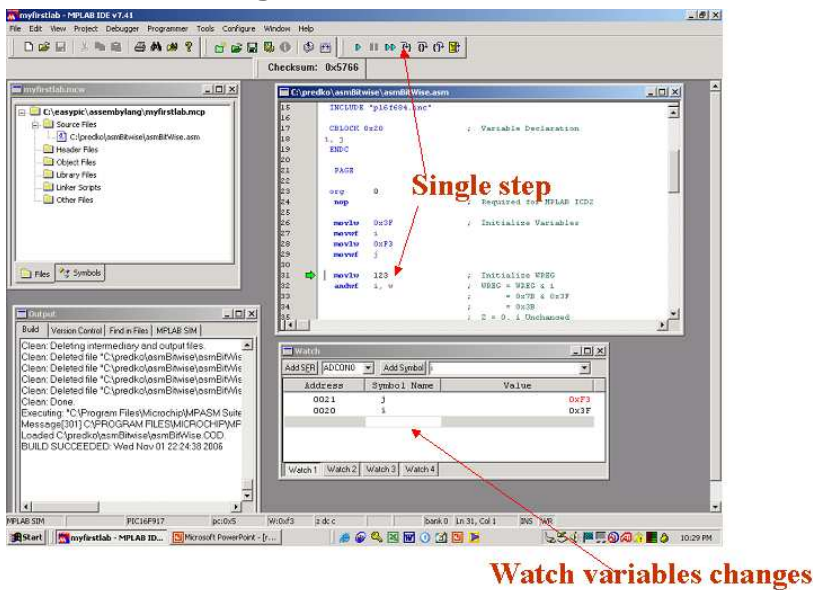
12. Step 12 – Add watch variables to Simulator (watch variables are the particular variables listed in the assembly file between Assembler CBLOCK and CEND directives).



13. Step 13 – Set Watch Variables in Watch window by clicking on Add Symbol and scrolling to desired variable



14. Step 14— Single Stepping execution and watching variable values change in WATCH Window



15. Exercise for ASMADD2.ASM

- Add watch i
- Add Watch w
- Add SFR Status
- Reset processor Simulator (press ICON)



- Single step --note for each of the following three assembly steps the values for w, status and i. Is Carry bit set after add?

```
org    0

        movlw 140          ; Initialize Variable
        movwf i

        addwf i, w         ; WREG = WREG + i
        ;   = 140 + 140
        ;   = 280
        ;   = b'(1)00011000'
        ;   = b'00011000'
        ; C = 1, DC = 1, Z = 0

end
```

- ## 16. Move to program manager window and right click on .asm program and remove. Click on source folder and then add asmDECfZ.asm , build, and remove all watch variable except STATUS and I for this file. How many times does Loop occur? Check the zero flag in STATUS at Loop end. Is it set? What does goto \$ do?

```
Variables
CBLOCK 0x20
i
ENDC
PAGE
org    0
        movlw 7           ; i = 7
        movwf i

Loop:   ; Loop here 7x
        nop              ; Do nothing in Loop
        decfsz i, f      ; i = i - 1, if result == 0, skip
        goto Loop
        goto $          ; Finished, just loop around
end
```

- ## 17. Move to program manager window and right click on .asm program and remove. Click on source folder and then add

asmWREG.asm file. Build , and remove all watch variables and then add STATUS , FSR and WREG. Single step. Why is the value of WREG not changing during each MOVLW step. Check STATUS zero flag after movf FSR,f. Is it set? Why?

```

org 0
nop ; Required for MPLAB ICD2
movlw 123 ; Load WREG with Literal
clrw ; Clear/Load WREG with 0
movlw 55 ; Load WREG with Decimal Literal
movlw 0x55 ; Load WREG with Hex Literal
movlw b'00110111' ; Load WREG with Binary Literal
movlw 'U' ; Load WREG with ASCII Character
movwf FSR ; Store WREG in a Register
clrw ; Clear WREG
movf FSR, f ; Set Zero Flag According to the
; Contents of Register
movf FSR, w ; Load WREG with Register
goto $ ; Finished, Everything Okay
end

```

18. Move to program manager window and right click on .asm program and remove. Click on source folder and then add asmCondition.asm, build. Remove all watch variables and then add STATUS, I and J. Verify the contents of I and J afeter all math operations and verifiy the status of flags that cause jump.

```

CBLOCK 0x20 ; Variable Declaration
i,j
ENDC
org 0
nop ; Required for MPLAB ICD2
movlw 12 ; Initialize Test Variables
movwf i
movlw 34
movwf j
movf i, w ; if i > j then ErrorLoop
subwf j, w
btfss STATUS, C
goto ErrorLoop
movf j, w ; if i >= j then ErrorLoop
subwf i, w
btfsc STATUS, C
goto ErrorLoop
movf i, w ; if i == j then ErrorLoop
subwf j, w
btfsc STATUS, Z
goto ErrorLoop
goto $ ; Finished, Everything Okay
ErrorLoop: ; Compare Operation didn't work correctly
goto ErrorLoop

```

19. Move to program manager window and right click on .asm program and remove. Click on source folder and then add asmBITWISE.asm . Build it. Put I , J WREG, STATUS in watch. Single step program and verify comment operation with math and falg settings.

```

CBLOCK 0x20          ; Variable Declaration
i,j
ENDC
org 0
nop                ; Required for MPLAB ICD2
movlw 0x3F         ; Initialize Variables
movwf i
movlw 0xF3
movwf j
movlw 123          ; Initialize WREG
andwf i, w         ; WREG = WREG & i
                    ; = 0x7B & 0x3F
                    ; = 0x3B
                    ; Z = 0, i Unchanged
movlw 123          ; Initialize WREG
andwf i, f         ; i = WREG & i
                    ; = 0x7B & 0x0F
                    ; = 0x3B
                    ; Z = 0, WREG Unchanged
iorwf j, w         ; WREG = WREG | j
                    ; = 0x7B + 0xF3
                    ; = 0xFB
                    ; Z = 0, j Unchanged
comf i, f          ; i = i ^ 0xFF
                    ; = 0x3B ^ 0xFF
                    ; = 0xC4
                    ; WREG Unchanged
xorwf i, f         ; i = WREG ^ i
                    ; = 0xFB ^ 0xC4
                    ; = 0x3F
xorwf i, w         ; WREG = WREG ^ i
                    ; = 0xFB ^ 0x3F
                    ; = 0xC4
xorwf i, f         ; i = WREG ^ i
                    ; = 0x3B ^ 0xC4
                    ; = 0xFB
goto $            ; Finished, Everything Okay

```