

Lab 3

Example 1

```
ORG 0x00      ; Program start address
GOTO MAIN      ; Jump to main program
MAIN:
    ; Initialization

    BANKSEL PORTC      ; Select bank containing PORTC
    CLRF PORTC          ; Clear PORTC (all outputs low initially)

    BANKSEL TRISC      ; Select bank containing TRISC
    MOVLW 0x01          ; Set RC0 as input, others as output (0000 0001)

    MOVWF TRISC         ; RC0 = input, RC1 = output
    NKSEL PORTC         ; Return to bank with PORTC for operations

    ; ===== Your logic code starts here =====

    ; Check bit 0 of PORTC

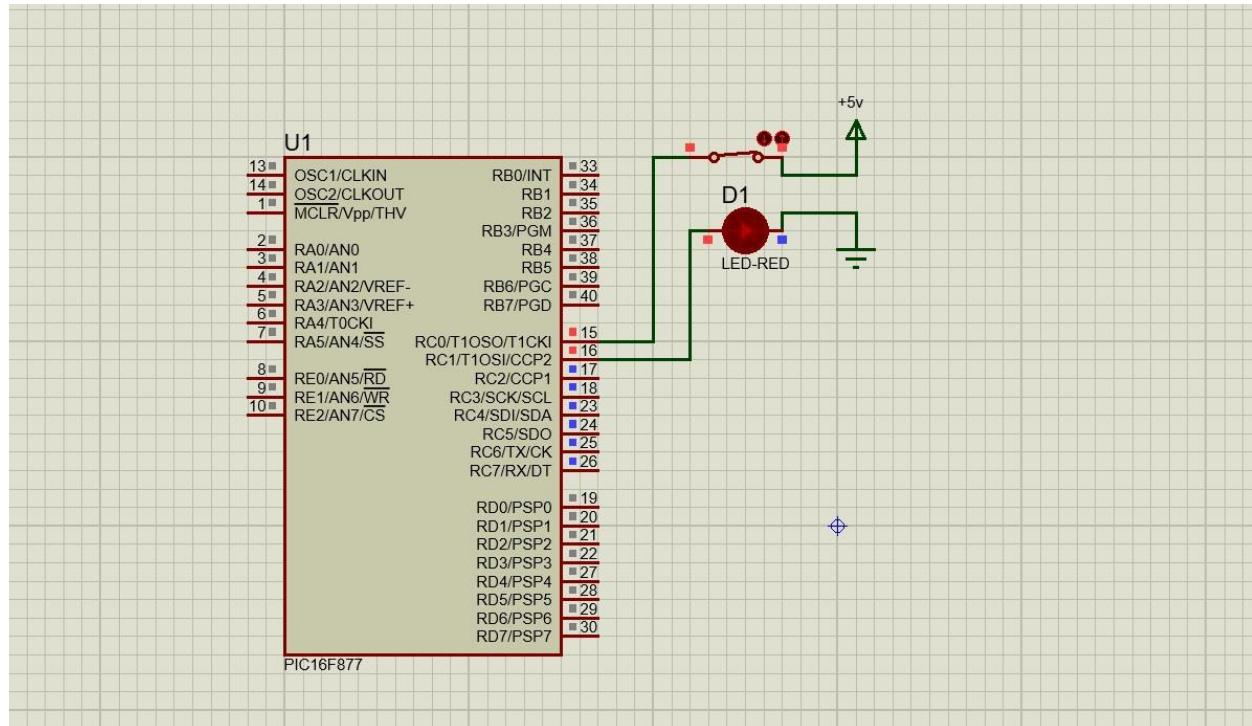
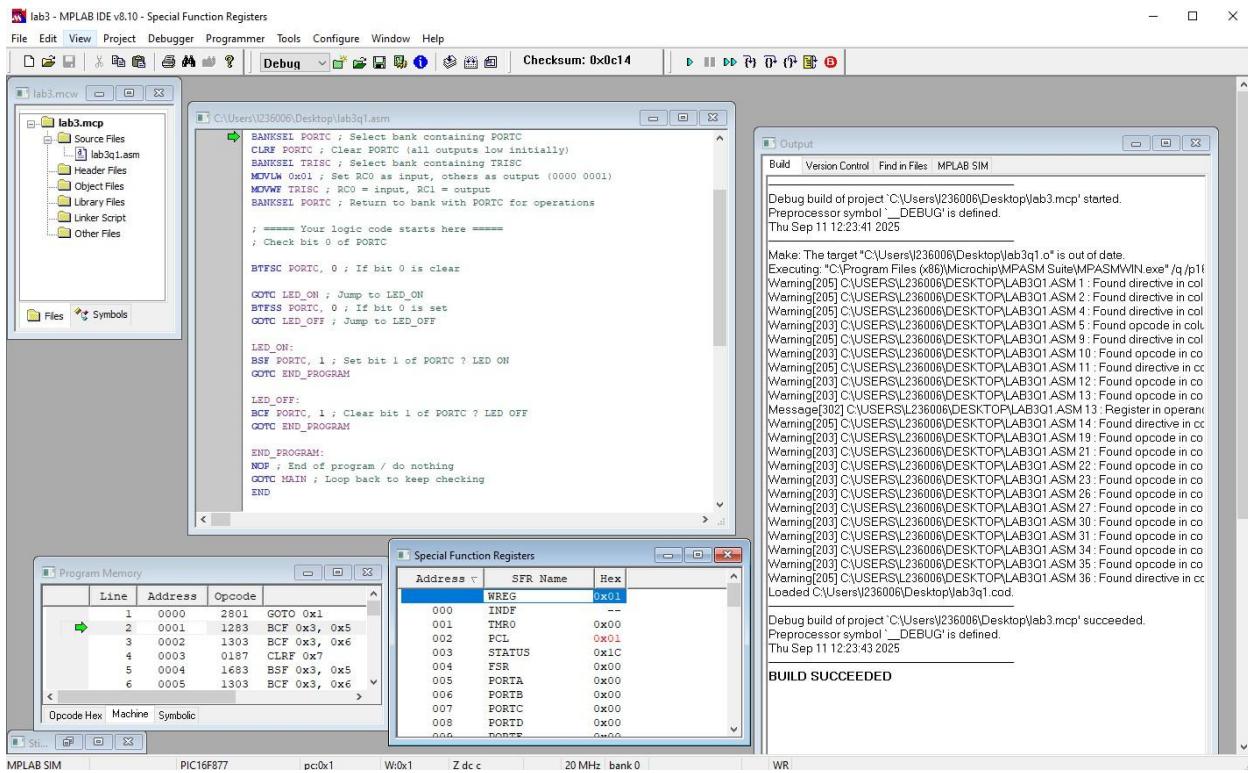
    BTFSC PORTC, 0      ; If bit 0 is clear
    GOTO LED_ON          ; Jump to LED_ON
    BTFSS PORTC, 0      ; If bit 0 is set
    GOTO LED_OFF         ; Jump to LED_OFF

LED_ON:
    BSF PORTC, 1          ; Set bit 1 of PORTC → LED ON
GOTO END_PROGRAM

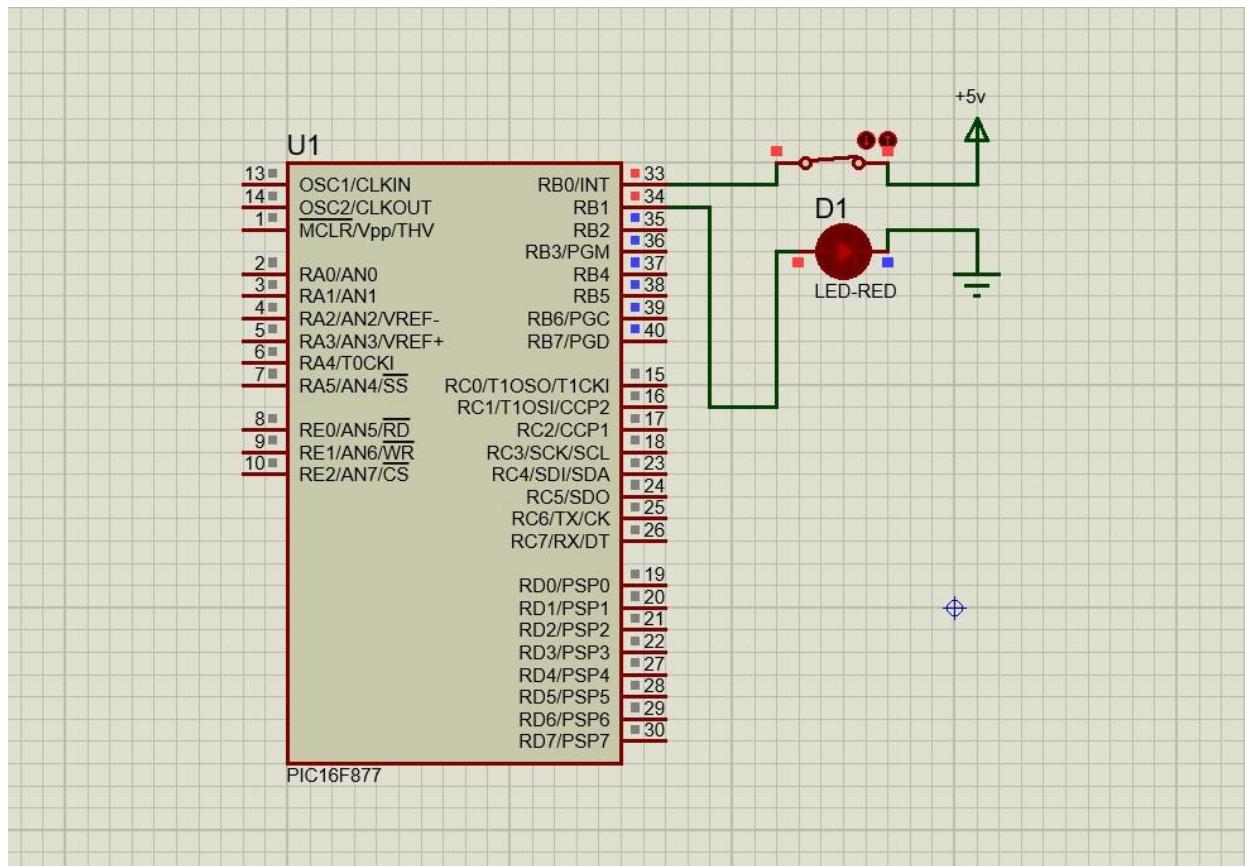
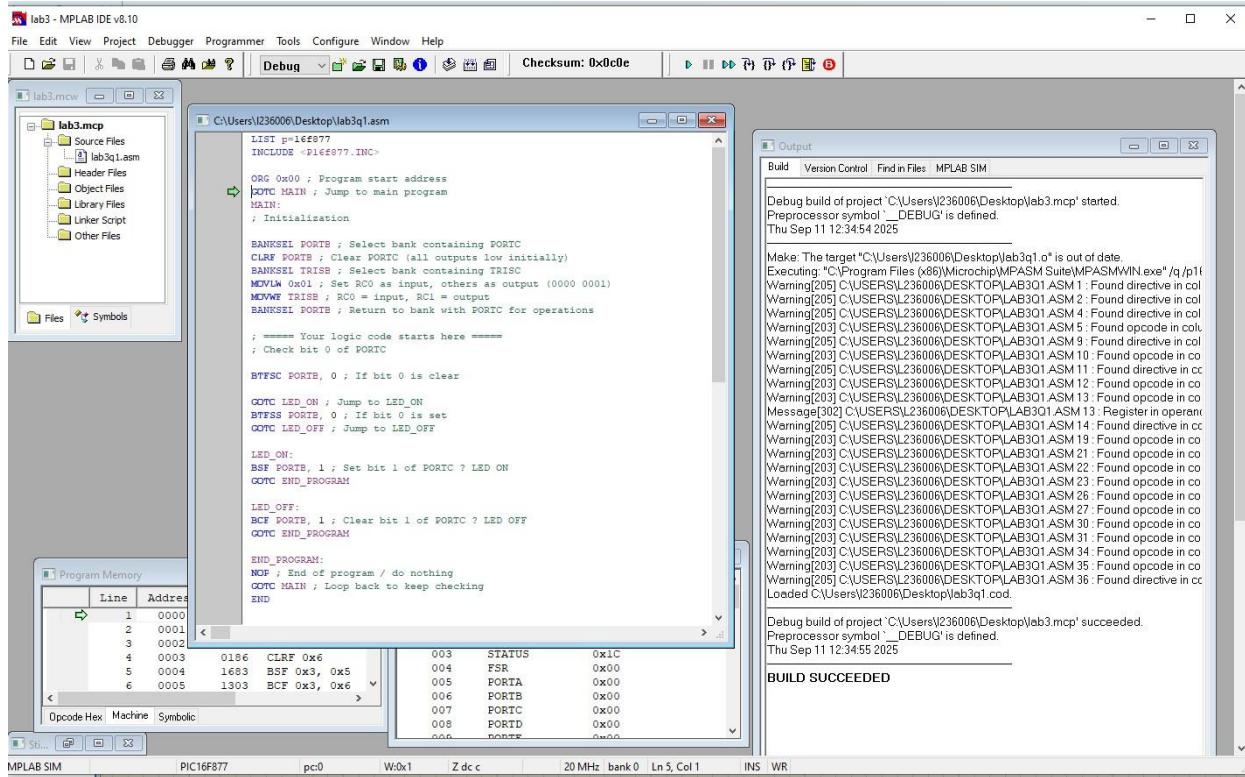
LED_OFF:
    BCF PORTC, 1          ; Clear bit 1 of PORTC → LED OFF
GOTO END_PROGRAM

END_PROGRAM:
    NOP                  ; End of program / do nothing
GOTO MAIN            ; Loop back to keep checking

END
```



Task1



Task2

```
LIST P=16F877
INCLUDE <P16F877.INC>

OUTER_COUNT EQU 0x20 ; File register for outer loop counter
INNER_COUNT EQU 0x21 ; File register for inner loop counter

ORG 0x00
GOTO MAIN

MAIN:
; Initialize PORTB = 55H
MOVLW 0x55
MOVWF PORTB

; Outer loop counter = 10
MOVLW 0x0A
MOVWF OUTER_COUNT

OUTERLOOP:
; Inner loop counter = 70
MOVLW 0x46
MOVWF INNER_COUNT

INNERLOOP:
COMF PORTB, F ; Complement PORTB
DECFSZ INNER_COUNT, F ; Inner counter--
GOTO INNERLOOP

DECFSZ OUTER_COUNT, F ; Outer counter--
GOTO OUTERLOOP

END
```

The screenshot shows the assembly code for the `newpi.asm` project. The code defines constants for outer and inner loop counts, initializes PORTB to 55H, and sets up loops for both counts. The `MAIN` section includes code to initialize PORTB and set up the outer loop counter. The `OUTERLOOP` section initializes the inner loop counter to 70. The `INNERLOOP` section is currently empty.

```
1 LIST P=16F877
2 INCLUDE <P16F877.INC>
3
4 OUTER_COUNT EQU 0x20 ; File register for outer loop counter
5 INNER_COUNT EQU 0x21 ; File register for inner loop counter
6
7 ORG 0x00
8 GOTO MAIN
9
10 MAIN:
11     ; Initialize PORTB = 55H
12     MOVLW 0x55
13     MOVWF PORTB
14
15     ; Outer loop counter = 10
16     MOVLW 0xA
17     MOVWF OUTER_COUNT
18
19 OUTEROOP:
20     ; Inner loop counter = 70
21     MOVLW 0x46
22     MOVWF INNER_COUNT
23
24 INNERLOOP:
```

Task3

```

1.      LIST P=16F877
2.      #include <P16F877.inc> 3.
ORG 0x00          ; Reset vector
GOTO MAIN

MAIN:
; Disable ADC so RA0/RA1 work as digital inputs
BANKSEL ADCON1
MOVLW 0x07
MOVWF ADCON1

; Configure PORTA as input
BANKSEL TRISA
MOVLW 0xFF
MOVWF TRISA

; Configure PORTB as output
BANKSEL TRISB
CLRF TRISB

; Clear ports initially
BANKSEL PORTA
CLRF PORTA
BANKSEL PORTB
CLRF PORTB

; ----- Main Loop -----
REDO:
BANKSEL PORTA

; Check RA0 (active-low button)
BTFS PORTA,0      ; Skip if RA0=1 (not pressed)
CALL SET_HEX       ; If RA0=0 (pressed), load 0x55

; Check RA1 (active-low button)
BTFS PORTA,1      ; Skip if RA1=1 (not pressed)
CALL CLR_HEX       ; If RA1=0 (pressed), clear PORTB
GOTO REDO          ; Keep monitoring
;

; Subroutine: load 0x55 into PORTB

SET_HEX:
BANKSEL PORTB
MOVLW 0x55
MOVWF PORTB
RETURN

; Subroutine: clear PORTB

CLR_HEX:
BANKSEL PORTB

```

. BTFS PORTA,0

BTFS = Bit Test File Skip if Set

It checks bit 0 of PORTA (RA0).

If RA0 = 1 → skip next instruction.

If RA0 = 0 → execute next instruction

BTFS PORTA,1

Similar to before, checks bit 1 of PORTA (RA1).

If RA1 = 1 → skip next line.

If RA1 = 0 (button pressed) → execute next line.

```
CLRF PORTB  
RETURN
```

```
END
```

