PIC Microcontroller (Lab 3)

Basic Digital I/O with LED & Push-Button Switch

Leonardo Fusser, 1946995

Experiment Performed on **8 October 2020**
Report Submitted on **29 October 2020**

Department of Computer Engineering Technology
Microcontroller & Microprocessor systems
Day Yann Fong

Learning today Leading tomorrow

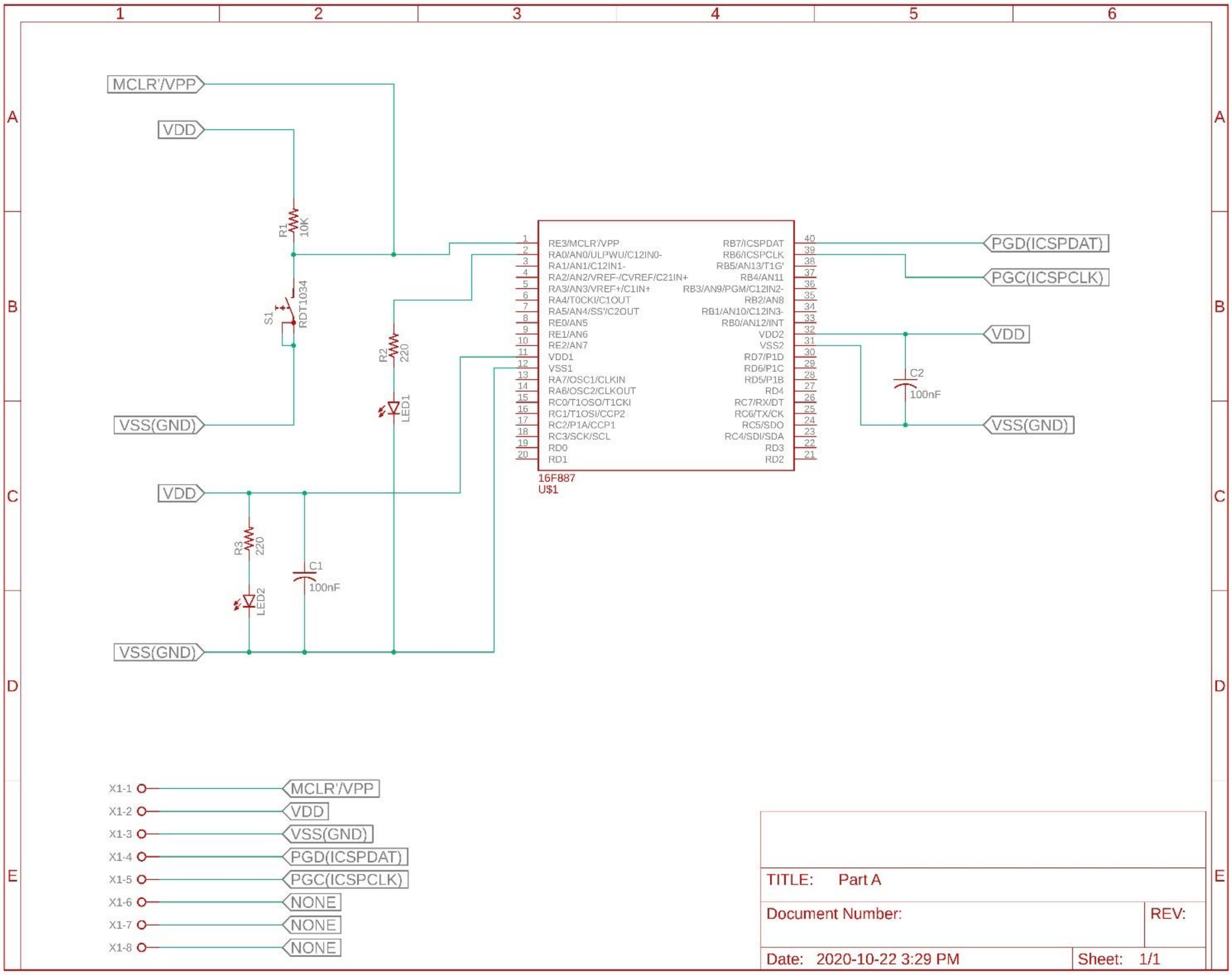
TABLE OF CONTENTS

1.0 Purpose

- Design and implement basic I/O configuration using LED and push-button switch.
- Learn to use and configure Digital I/O for PIC 16F887.
- Learn how to read a digital input from push-button switch.
- Load basic program onto PIC 16F887 using MPLAB and PICKIT 4 to turn on/off an LED.

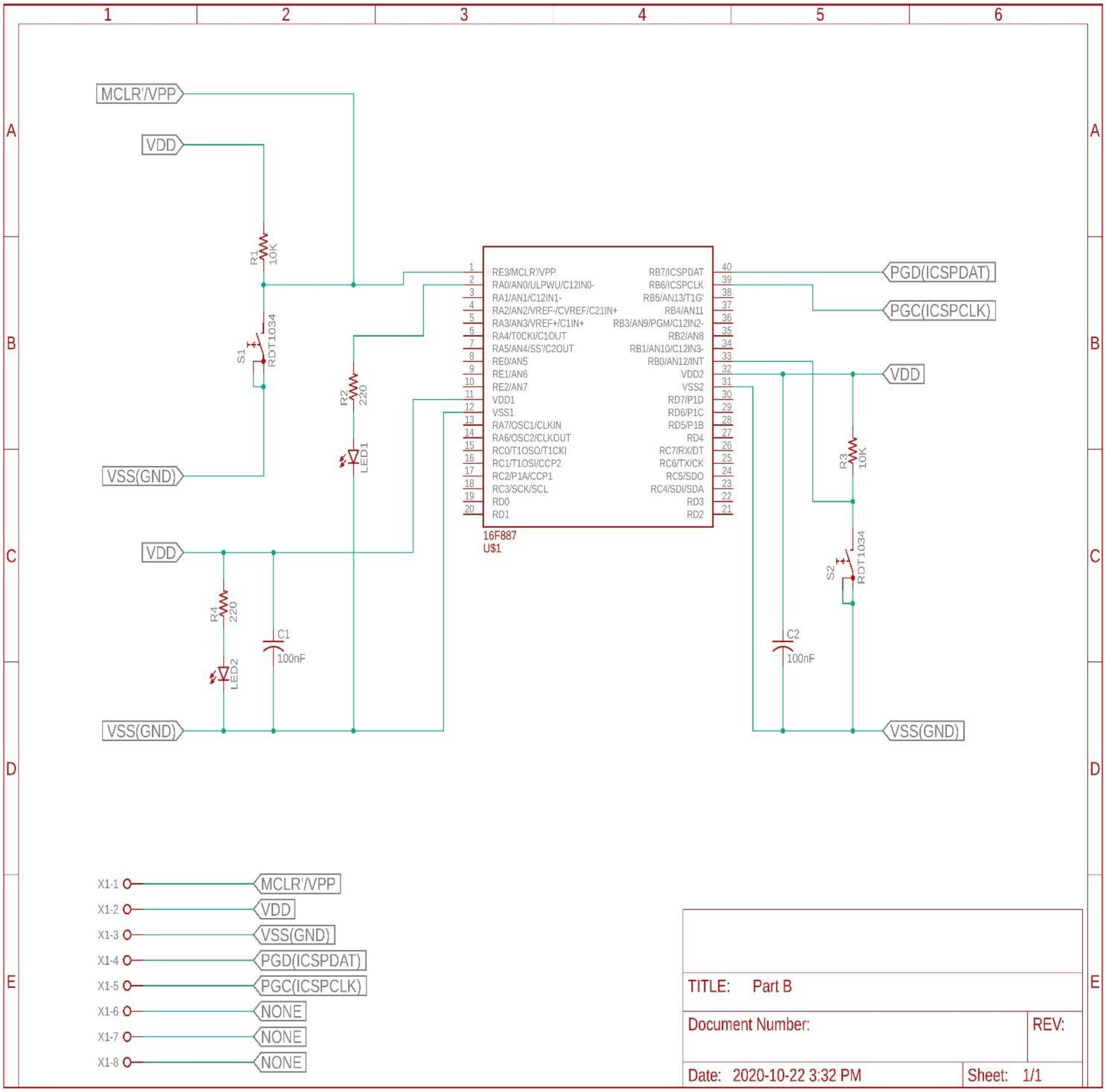
2.0 Original Design

Part A schematic



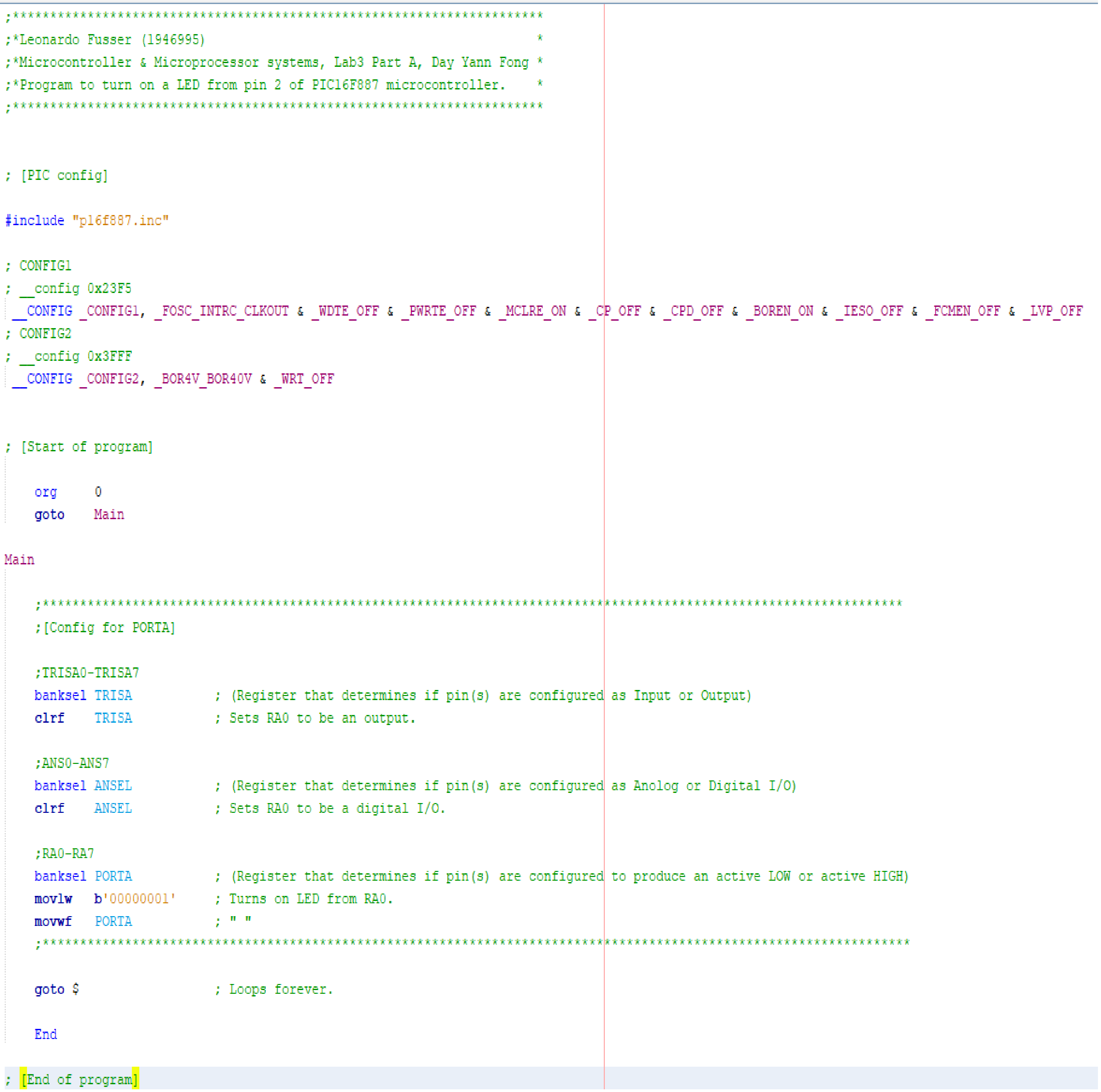
LED connected to pin 2 (RA0) of PIC 16F887 microcontroller shown above (separate schematic attached to this report submission – LeonardoFusser_Schematic-Part A.pdf)

Part B schematic



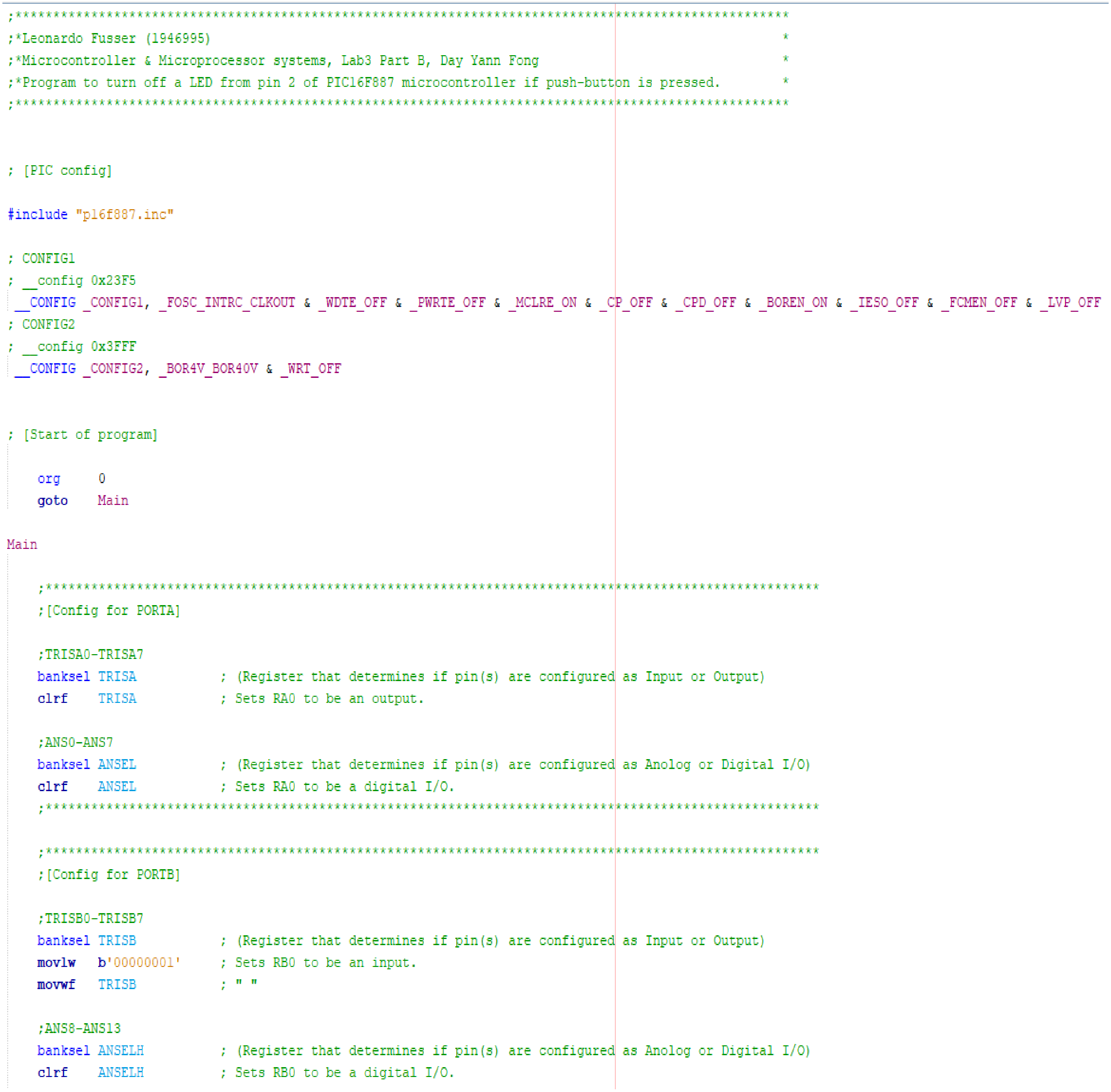
LED connected to pin 2 (RA0), and pin 33 (RB0) of PIC 16F887 microcontroller interfacing to push-button switch shown above. Push-button switch is based off of "active-low" configuration (separate schematic attached to this report submission – LeonardoFusser_Schematic-Part B.pdf)

Part A assembly code

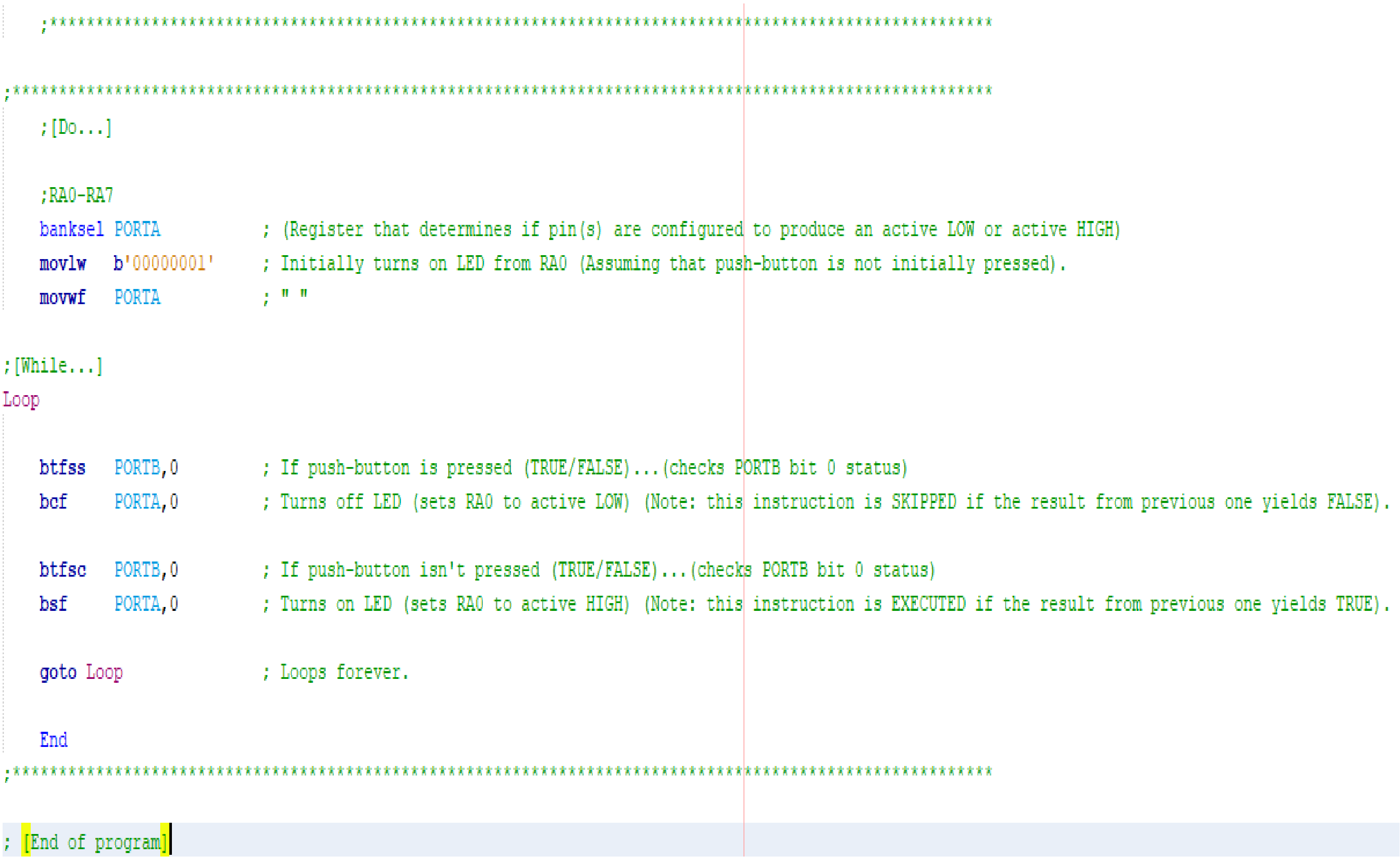


```
;***********************************************************************
;*Leonardo Fusser (1946995)                                            *
;*Microcontroller & Microprocessor systems, Lab3 Part A, Day Yann Fong *
;*Program to turn on a LED from pin 2 of PIC16F887 microcontroller.    *
;***********************************************************************



; [PIC config]

#include "p16f887.inc"

; CONFIG1
; __config 0x23F5
 __CONFIG _CONFIG1, _FOSC_INTRC_CLKOUT & _WDTE_OFF & _PWRTE_OFF & _MCLRE_ON & _CP_OFF & _CPD_OFF & _BOREN_ON & _IESO_OFF & _FCMEN_OFF & _LVP_OFF
; CONFIG2
; __config 0x3FFF
 __CONFIG _CONFIG2, _BOR4V_BOR40V & _WRT_OFF


; [Start of program]

    org     0
    goto    Main

Main

    ;*******************************************************************************************************************
    ;[Config for PORTA]

    ;TRISA0-TRISA7
    banksel TRISA          ; (Register that determines if pin(s) are configured as Input or Output)
    clrf    TRISA          ; Sets RA0 to be an output.

    ;ANS0-ANS7
    banksel ANSEL          ; (Register that determines if pin(s) are configured as Anolog or Digital I/O)
    clrf    ANSEL          ; Sets RA0 to be a digital I/O.

    ;RA0-RA7
    banksel PORTA          ; (Register that determines if pin(s) are configured to produce an active LOW or active HIGH)
    movlw   b'00000001'    ; Turns on LED from RA0.
    movwf   PORTA          ; " "
    ;********************************************************************************************************************

    goto $                 ; Loops forever.

    End

; [End of program]
```

Part B assembly code



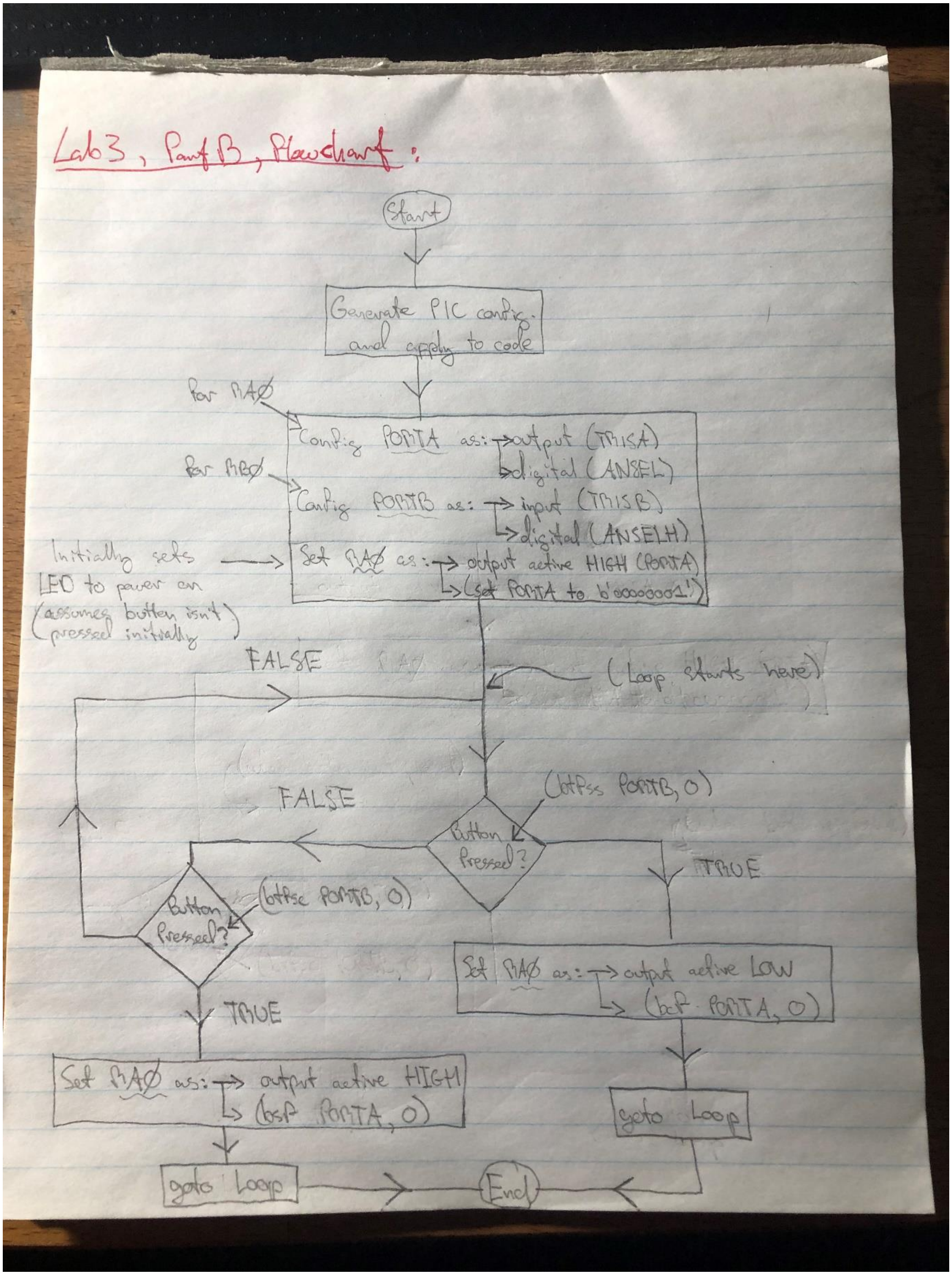
```
;*****************************************************************************************
;*Leonardo Fusser (1946995)                                                              *
;*Microcontroller & Microprocessor systems, Lab3 Part B, Day Yann Fong                   *
;*Program to turn off a LED from pin 2 of PIC16F887 microcontroller if push-button is pressed.      *
;*****************************************************************************************


; [PIC config]

#include "p16f887.inc"

; CONFIG1
; __config 0x23F5
 __CONFIG _CONFIG1, _FOSC_INTRC_CLKOUT & _WDTE_OFF & _PWRTE_OFF & _MCLRE_ON & _CP_OFF & _CPD_OFF & _BOREN_ON & _IESO_OFF & _FCMEN_OFF & _LVP_OFF
; CONFIG2
; __config 0x3FFF
 __CONFIG _CONFIG2, _BOR4V_BOR40V & _WRT_OFF


; [Start of program]

    org     0
    goto    Main

Main

    ;*****************************************************************************************
    ;[Config for PORTA]

    ;TRISA0-TRISA7
    banksel TRISA           ; (Register that determines if pin(s) are configured as Input or Output)
    clrf    TRISA           ; Sets RA0 to be an output.

    ;ANS0-ANS7
    banksel ANSEL           ; (Register that determines if pin(s) are configured as Anolog or Digital I/O)
    clrf    ANSEL           ; Sets RA0 to be a digital I/O.
    ;*****************************************************************************************

    ;*****************************************************************************************
    ;[Config for PORTB]

    ;TRISB0-TRISB7
    banksel TRISB           ; (Register that determines if pin(s) are configured as Input or Output)
    movlw   b'00000001'     ; Sets RB0 to be an input.
    movwf   TRISB           ; " "

    ;ANS8-ANS13
    banksel ANSELH          ; (Register that determines if pin(s) are configured as Anolog or Digital I/O)
    clrf    ANSELH          ; Sets RB0 to be a digital I/O.
```

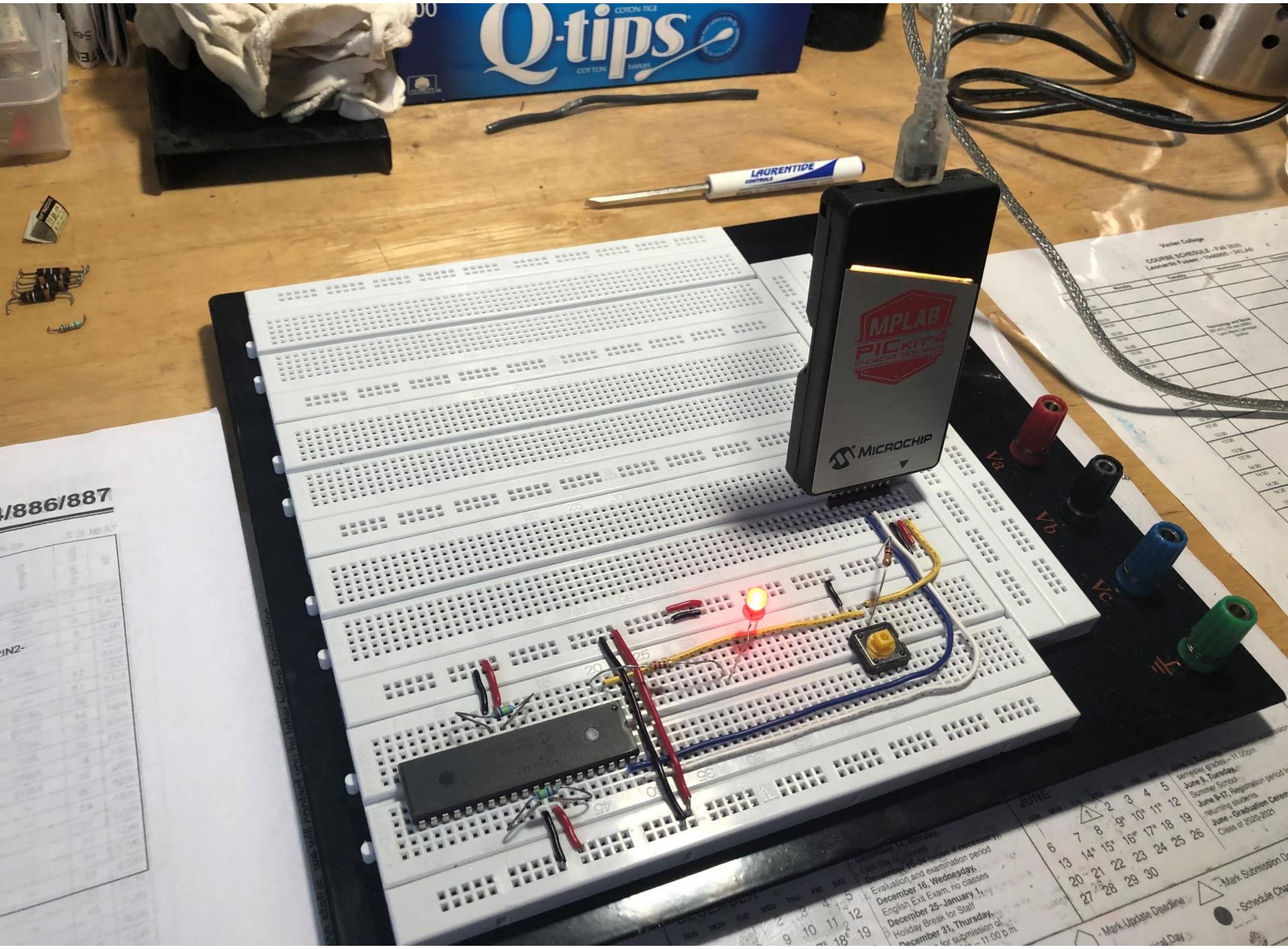


```
    ;*****************************************************************************************************

;*********************************************************************************************************
    ;[Do...]

    ;RA0-RA7
    banksel PORTA           ; (Register that determines if pin(s) are configured to produce an active LOW or active HIGH)
    movlw   b'00000001'     ; Initially turns on LED from RA0 (Assuming that push-button is not initially pressed).
    movwf   PORTA           ; " "

;[While...]
Loop

    btfss   PORTB,0         ; If push-button is pressed (TRUE/FALSE)...(checks PORTB bit 0 status)
    bcf     PORTA,0         ; Turns off LED (sets RA0 to active LOW) (Note: this instruction is SKIPPED if the result from previous one yields FALSE).

    btfsc   PORTB,0         ; If push-button isn't pressed (TRUE/FALSE)...(checks PORTB bit 0 status)
    bsf     PORTA,0         ; Turns on LED (sets RA0 to active HIGH) (Note: this instruction is EXECUTED if the result from previous one yields TRUE).

    goto Loop               ; Loops forever.

    End
;*********************************************************************************************************

; [End of program]
```

<u>Note</u>: ***.asm* source files for both Part A and Part B are attached to this submission for your convenience.**

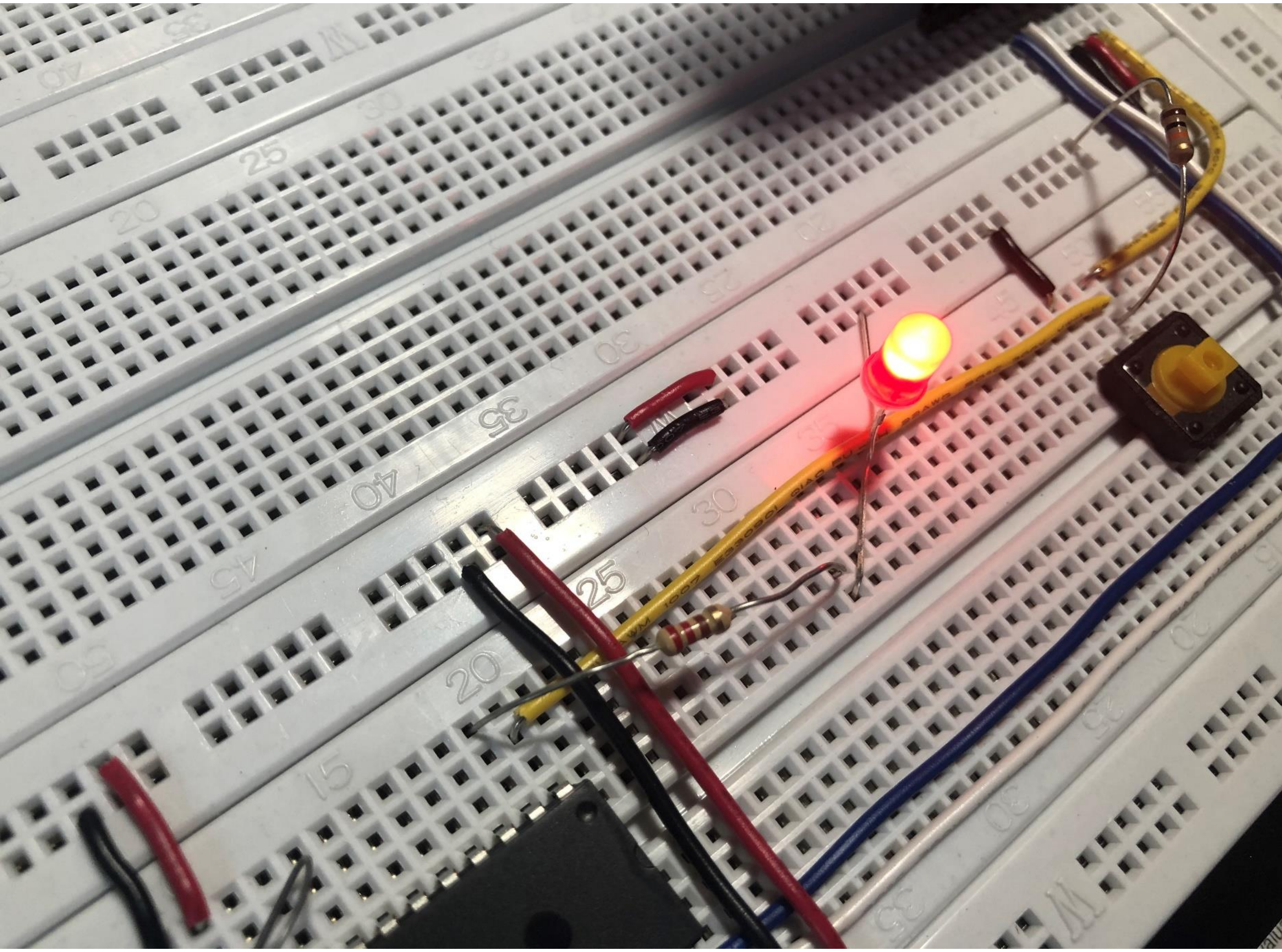
Part B flowchart


Lab3, Part B, Flowchart:
Start
Generate PIC config. and apply to code
For RAØ
Config PORTA as: → output (TRISA)
→ digital (ANSEL)
For RBØ
Config PORTB as: → input (TRISB)
→ digital (ANSELH)
Initially sets LED to power on (assumes button isn't pressed initially)
Set RAØ as: → output active HIGH (PORTA)
→ (set PORTA to b'00000001')
FALSE
(Loop starts here)
(btfss PORTB, 0)
FALSE
Button Pressed?
TRUE
Button Pressed?
(btfsc PORTB, 0)
Set RAØ as: → output active LOW
→ (bcf PORTA, 0)
TRUE
Set RAØ as: → output active HIGH
→ (bsf PORTA, 0)
goto Loop
goto Loop
End



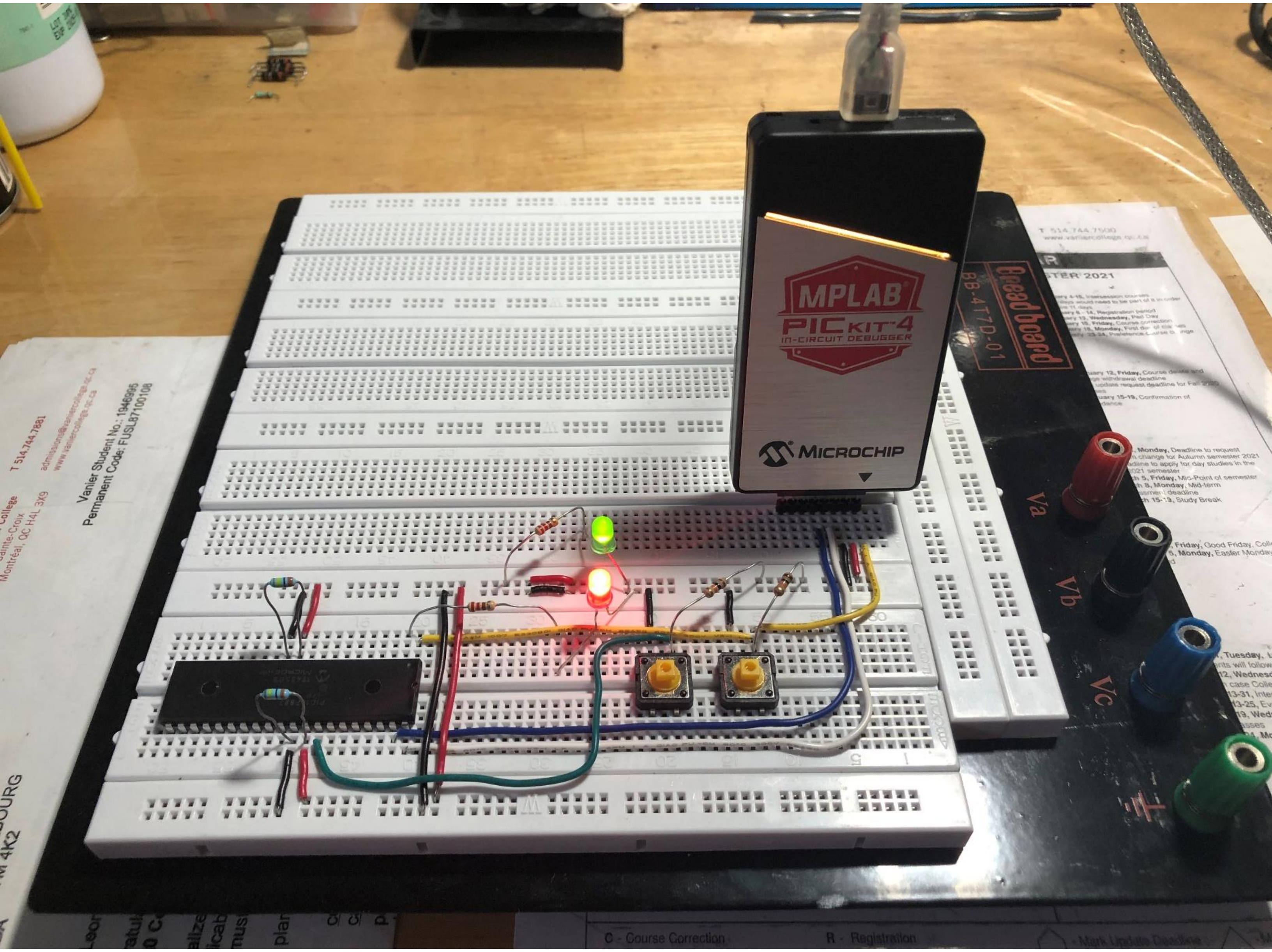
Circuit prototype for Part A (I)

<u>Green light</u>: power-up LED (not shown here, shown in Part B circuit prototype), <u>Red light</u>: LED for RA0.
<u>Button closest to PICKIT</u>: push-button for reset circuit.



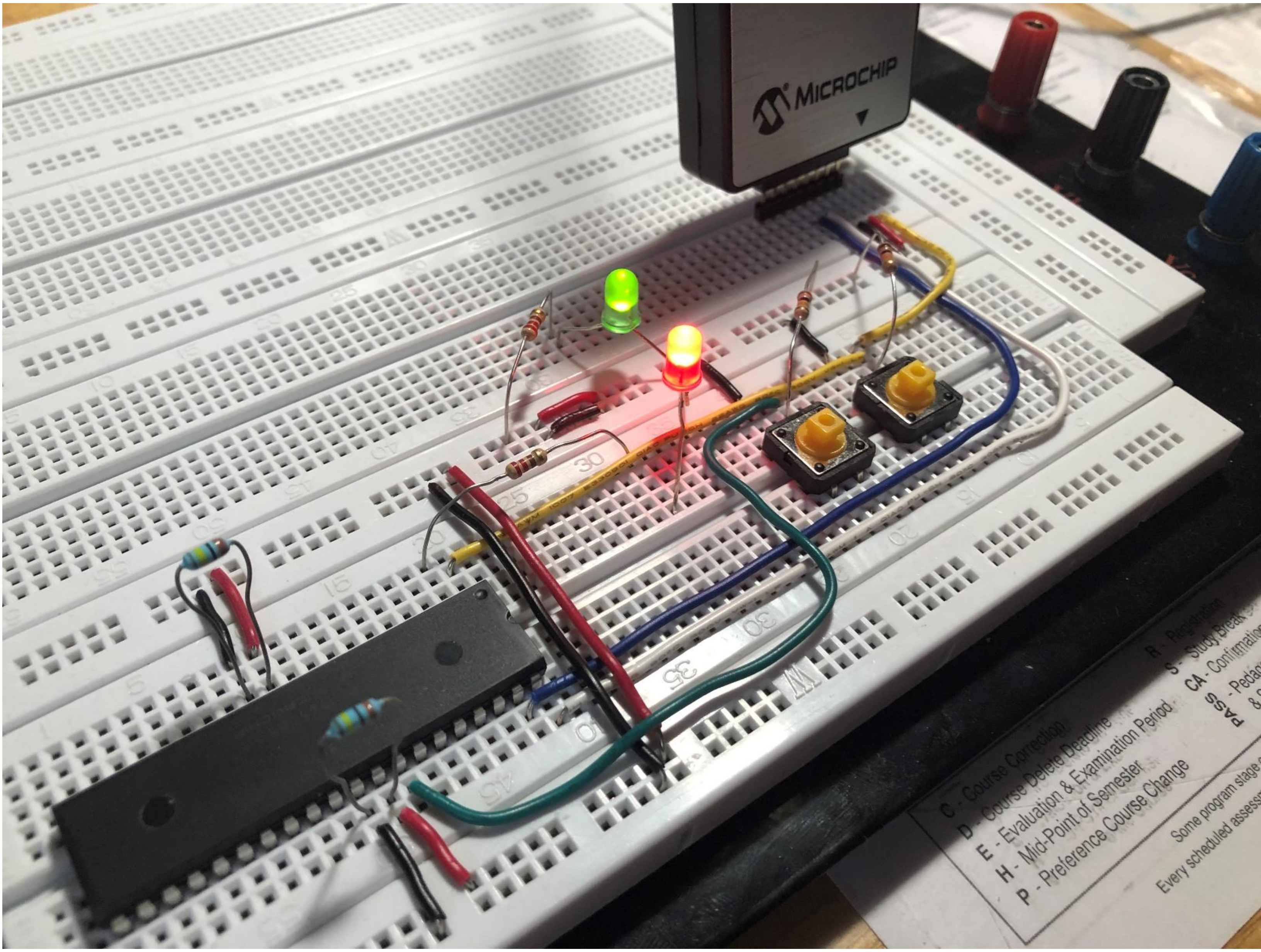
Circuit prototype for Part A (II)

***Green light**: power-up LED (not shown here, shown in Part B circuit prototype), **Red light**: LED for RA0.*
***Button closest to PICKIT**: push-button for reset circuit.*



Circuit prototype for Part B (I)

<u>Green light</u>: power-up LED, <u>Red light</u>: LED for RA0. <u>Button next to red LED</u>: push-button for RB0, <u>Button closest to PICKIT</u>: push-button for reset circuit.



Circuit prototype for Part B (II)

<u>Green light</u>: power-up LED, <u>Red light</u>: LED for RA0. <u>Button next to red LED</u>: push-button for RB0, <u>Button closest to PICKIT</u>: push-button for reset circuit.

3.0 Observations

Observations from the lab

- Circuit prototype (for Part A) was missing two jumper connections for VDD and VSS columns on breadboard. This needed to be corrected for Part B circuit to work.
- Breadboard is designed to have split power and ground columns, so without jumpers VDD (pin 11 & 32) and VSS (pin 12 & 31) were not getting power from PICKIT4, causing programming errors in MPLAB (error: Target device invalid 0x0).
- Assembly code wasn't done properly in beginning.
- Assembly isn't the same as C/C++ (used references in course notes to review assembly programming basics).
- Encountered compiling errors when doing Part A (mostly syntax).
- Formatting is very important in assembly (ex: spaces, syntax).
- Corrected issues encountered above (Part A).
- Minor compiling errors encountered in Part B (mostly syntax).
- Loop structure took a while to get working in Part B.
- Logic for Part B took a while to understand (mostly figuring out how different switch configurations affected circuit behavior – each with different logic).
- Corrected issues with understanding of logic using flowchart.
- Corrected issues encountered above (Part B).
- Learned how to implement logic of Part B through bit-oriented instructions on PIC.

4.0 Conclusion

- Purpose of this lab has been achieved.
- Understood how to design basic I/O configuration using LED and switch.
- Understood how to implement LED and switch with PIC 16F887.
- Understood how to perform basic digital I/O configuration for PIC 16F887.
- Understood how to read status of pin from switch using PIC 16F887.
- Understood how to load a basic program to turn LED on/off onto the PIC16F887.
- <u>Problem</u>: Flowchart incorrect for Part B, caused problems after writing program (circuit behavior was not what was intended). Logic was incorrect.
- <u>Solution</u>: Corrected flowchart after some further understanding of the situation, afterwards program was updated to match flowchart and circuit worked as it should.