Student no: _2020-01144-MN-0 **Date Started: Feb. 22, 2023**

Name: _Jewel Anne Reyes____ **Date Submitted: Mar 1, 2023**

Machine Problem # 2

Running Light & Counters

Objective:

- To Perform Sequential Display using MCU processing.
- To utilize branching and Looping technique in Assembly Language
- To enhance skills of Assembly Programming using MPLAB and Proteus.
- To be familiar with Timing analysis in Assembly language and to use the 1-second routine.
- To learn how to use different instruction sets to a specific requirement.
- To improve on instruction analysis and tracing.
- To improve on using subroutines for more efficient programming.

Background:

One of the most exciting activities in MCU programming is performing the Sequential Logic Processing without the burden of K-Mapping. With the knowledge of timing analysis for each instruction cycle, it would be easy to calculate the overall time in which the MCU performs the set of instruction. The code below is the 1-second routine for a 4MHz clock which commonly used in most Delay function.

```
Delay:
        movlw D'6'
        movwf CounterC
        movlw D'24'
        movwf CounterB
        movlw D'167'
        movwf CounterA
loop:
        decfsz CounterA,1
        goto loop
        decfsz CounterB,1
        goto loop
        decfsz CounterC,1
        goto loop
        nop
        return
```

Materials:

PIC16F84A PC

PIC Programmer Power Supply

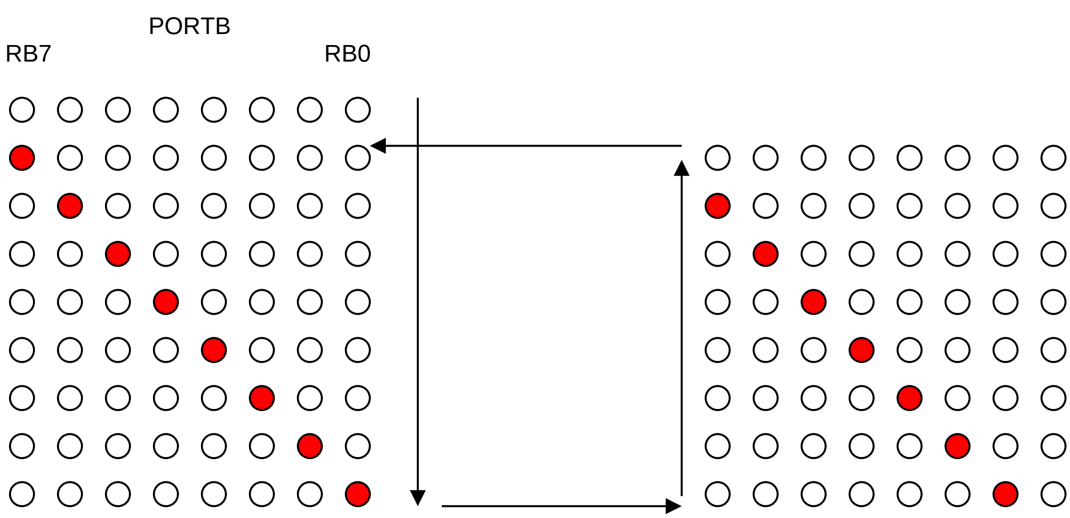
Test Jig

Procedure:

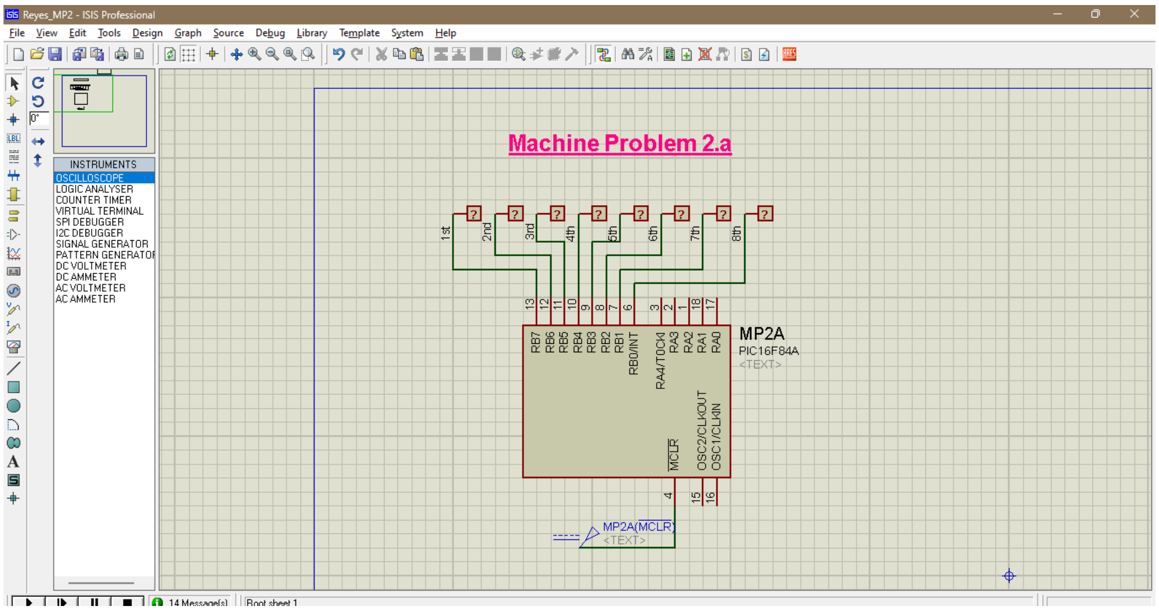
Perform the following Setup and provide a code for each setup

SETUP A: Running Light

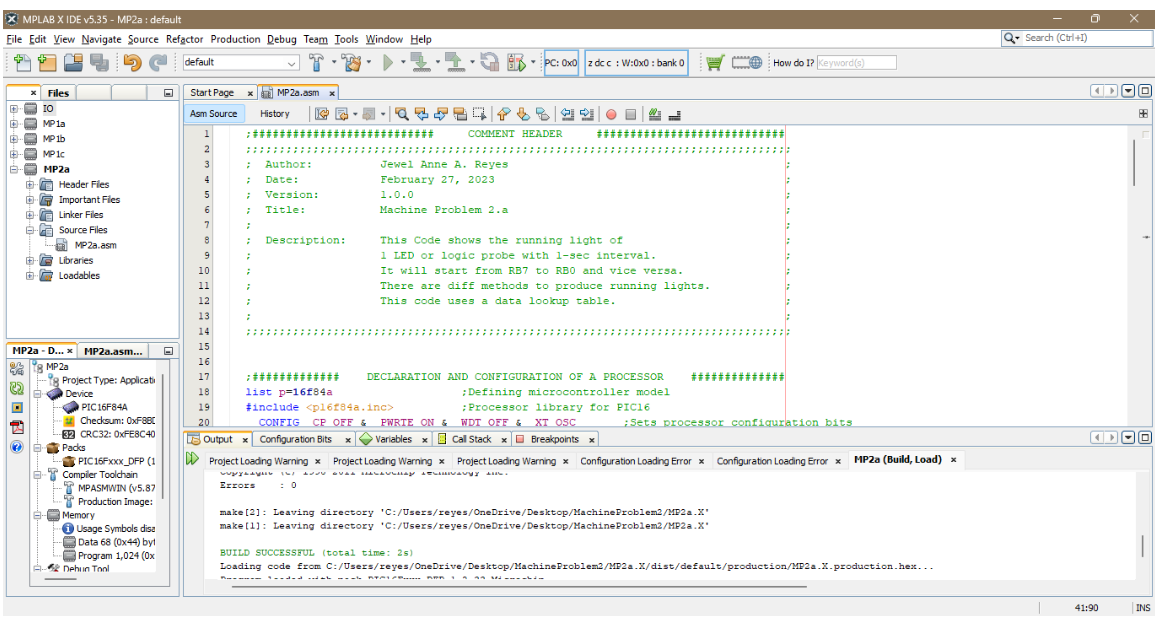
A. Configure the PORTB as output Port. Create a Code that will display a running light of One LED with 1-second interval.



Schematic:

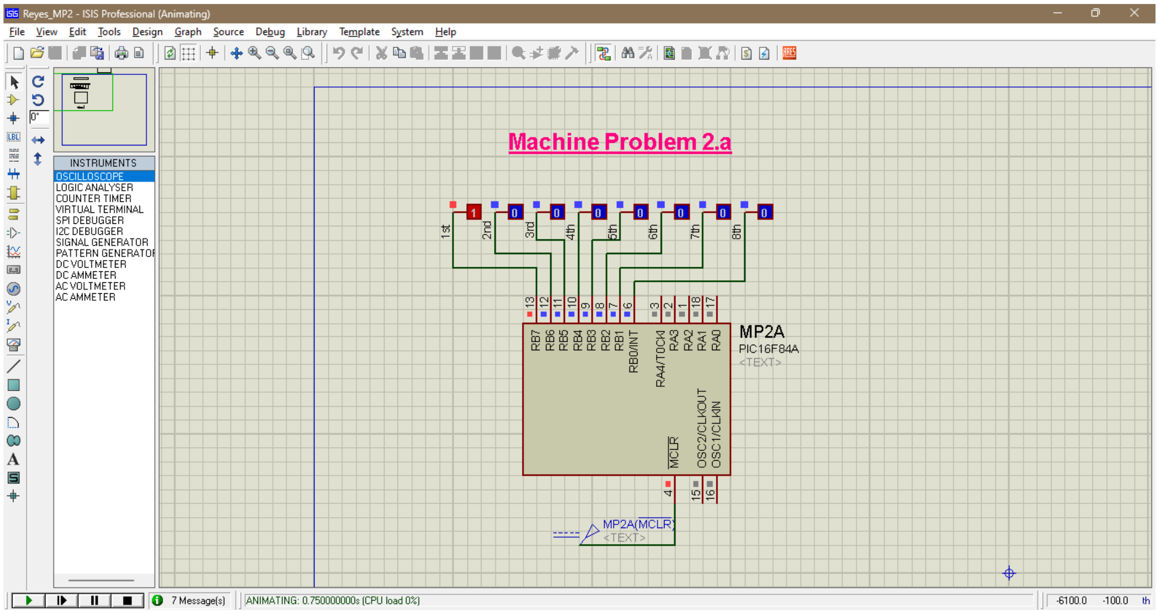


Program Build Successful:

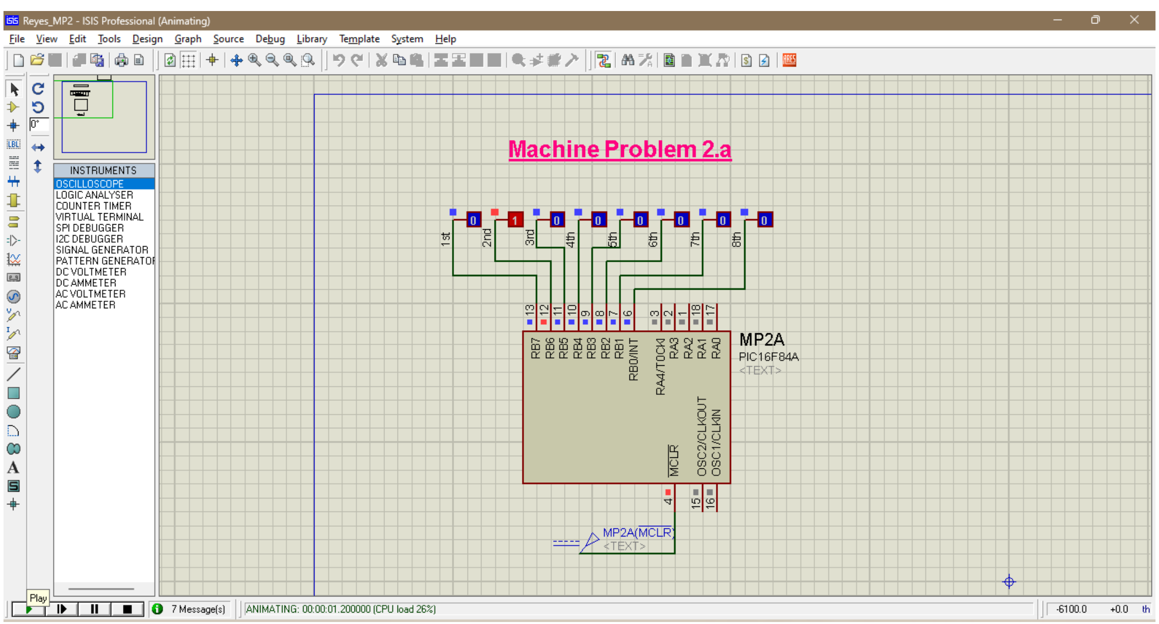


Outputs:

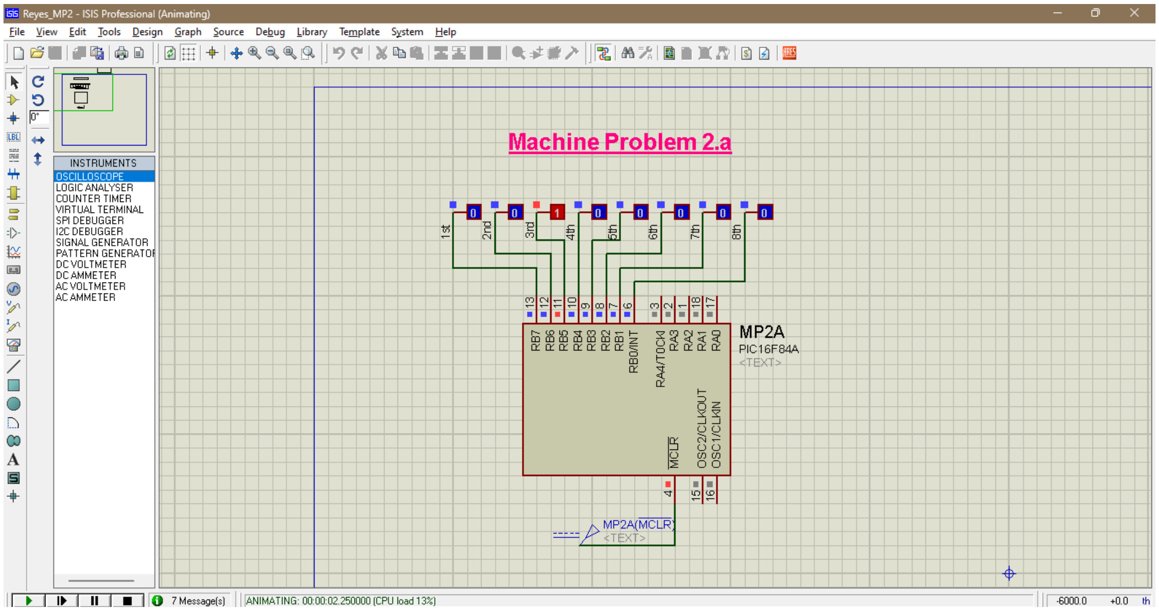
a. RB7



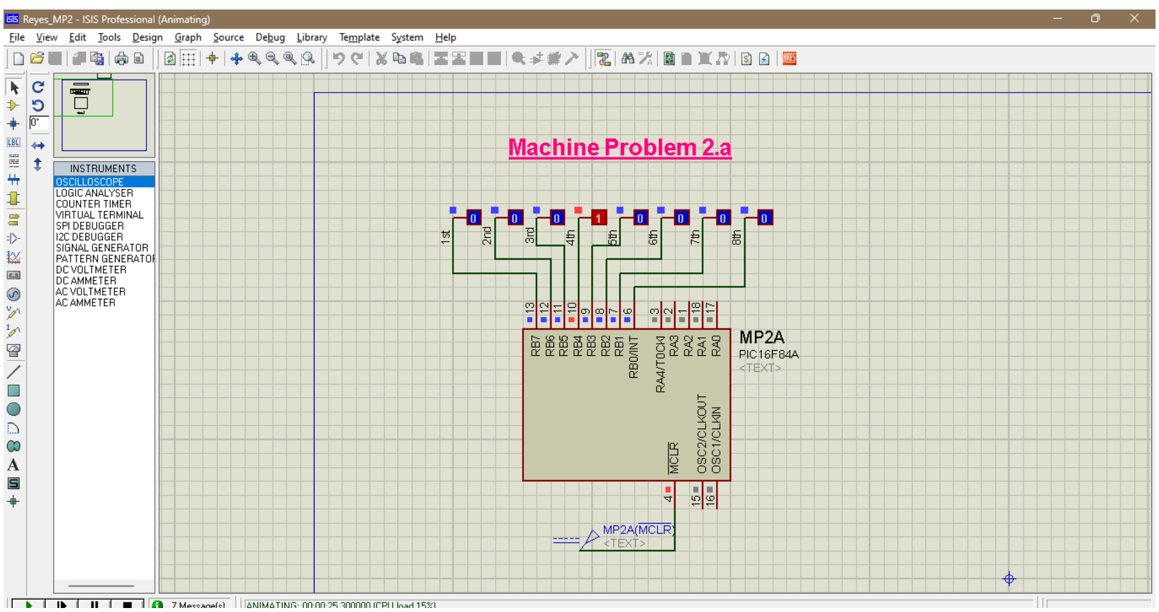
b. RB6



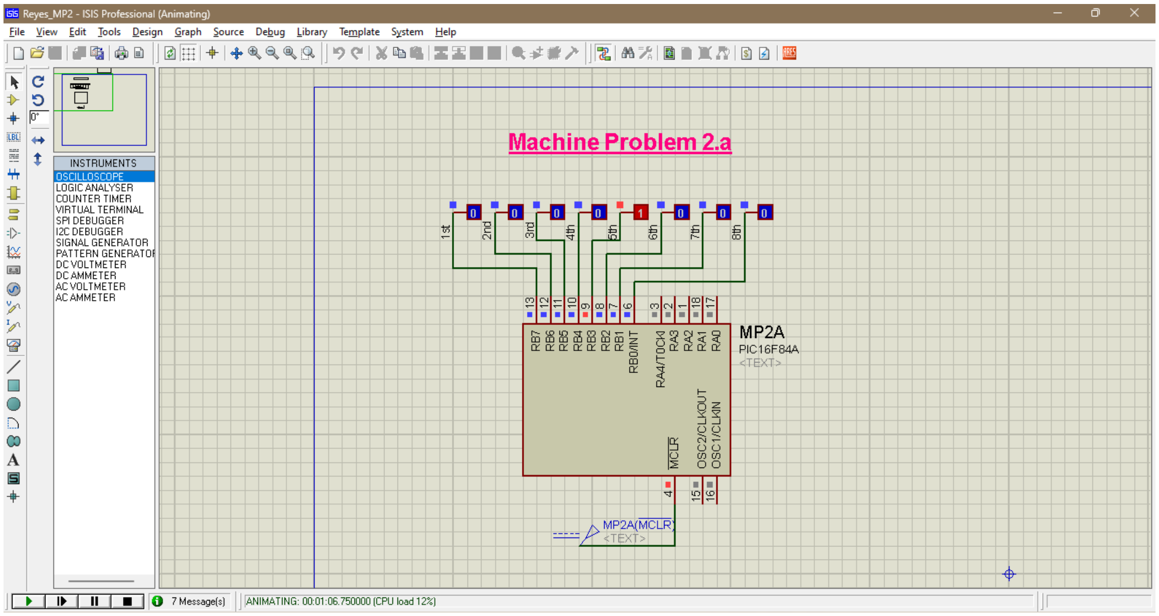
c. RB5


Reyes_MP2 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
INSTRUMENTS
OSCILLOSCOPE
LOGIC ANALYSER
COUNTER TIMER
VIRTUAL TERMINAL
SPI DEBUGGER
I2C DEBUGGER
SIGNAL GENERATOR
PATTERN GENERATOR
DC VOLTMETER
DC AMMETER
AC VOLTMETER
AC AMMETER
Machine Problem 2.a
1st
2nd
3rd
4th
5th
6th
7th
8th
RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0/INT RA4/T0CKI RA3 RA2 RA1 RA0
MP2A
PIC16F84A
<TEXT>
MCLR
OSC2/CLKOUT
OSC1/CLKIN
MP2A(MCLR)
<TEXT>
7 Message(s)
ANIMATING: 00:00:02.250000 (CPU load 13%)
-6000.0
+0.0

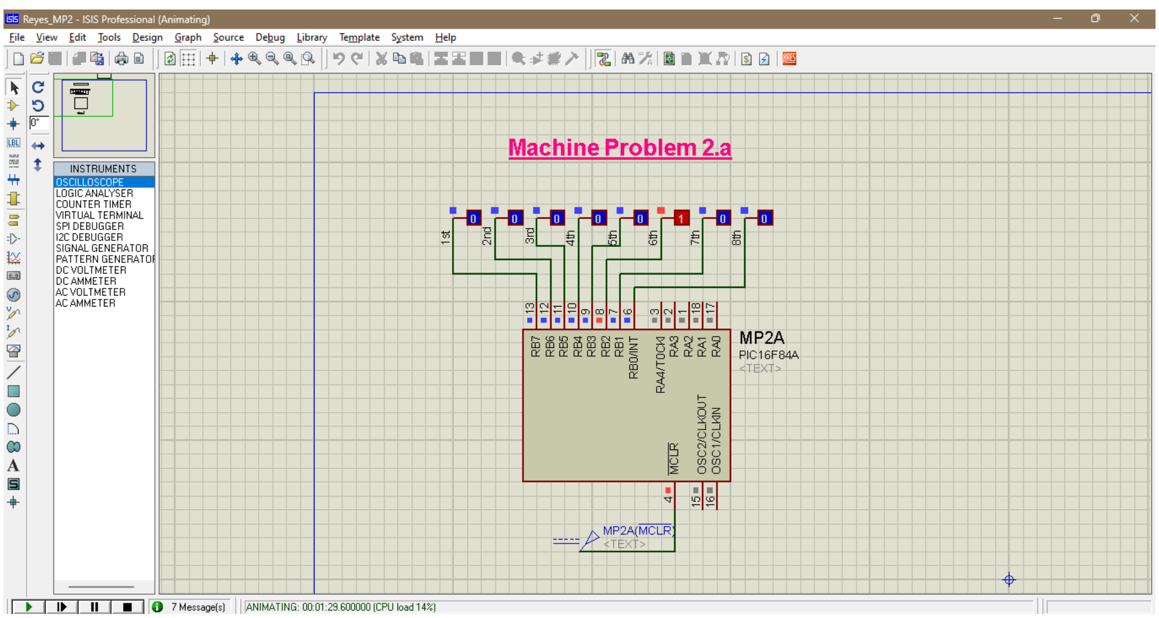
d. RB4


Reyes_MP2 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
INSTRUMENTS
OSCILLOSCOPE
LOGIC ANALYSER
COUNTER TIMER
VIRTUAL TERMINAL
SPI DEBUGGER
I2C DEBUGGER
SIGNAL GENERATOR
PATTERN GENERATOR
DC VOLTMETER
DC AMMETER
AC VOLTMETER
AC AMMETER
Machine Problem 2.a
1st
2nd
3rd
4th
5th
6th
7th
8th
RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0/INT RA4/T0CKI RA3 RA2 RA1 RA0
MP2A
PIC16F84A
<TEXT>
MCLR
OSC2/CLKOUT
OSC1/CLKIN
MP2A(MCLR)
<TEXT>

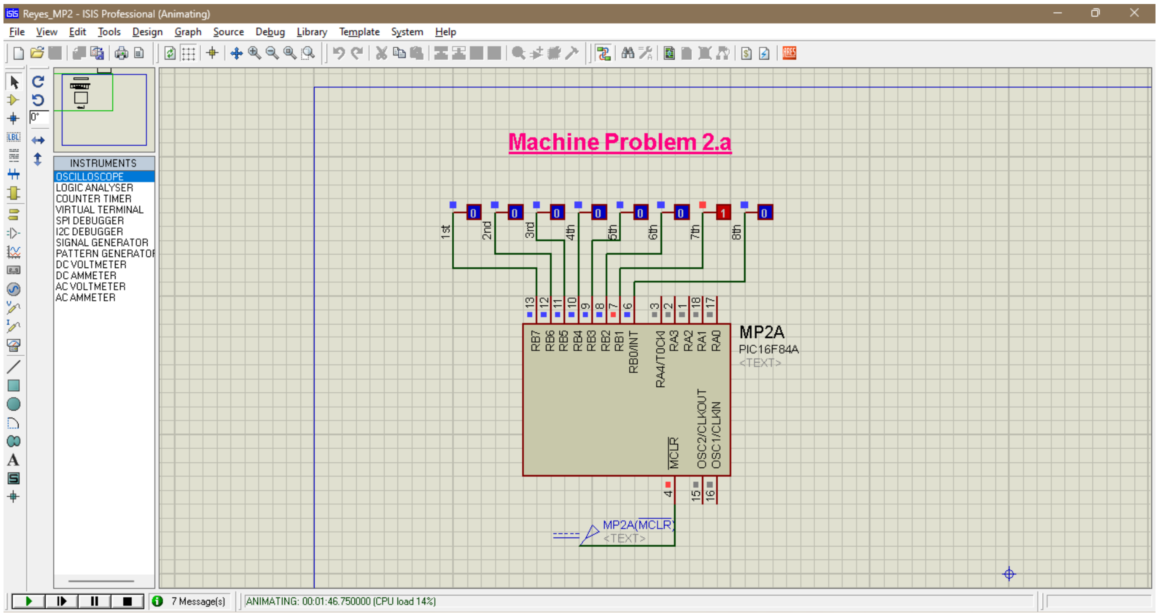
e. RB3


Reyes_MP2 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
INSTRUMENTS
OSCILLOSCOPE
LOGIC ANALYSER
COUNTER TIMER
VIRTUAL TERMINAL
SPI DEBUGGER
I2C DEBUGGER
SIGNAL GENERATOR
PATTERN GENERATOR
DC VOLTMETER
DC AMMETER
AC VOLTMETER
AC AMMETER
Machine Problem 2.a
1st 2nd 3rd 4th 5th 6th 7th 8th
0 0 0 0 1 0 0 0
RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0/INT RA4/T0CKI RA3 RA2 RA1 RA0
MCLR OSC2/CLKOUT OSC1/CLKIN
MP2A
PIC16F84A
<TEXT>
MP2A(MCLR)
7 Message(s)
ANIMATING: 00:01:06.750000 (CPU load 12%)

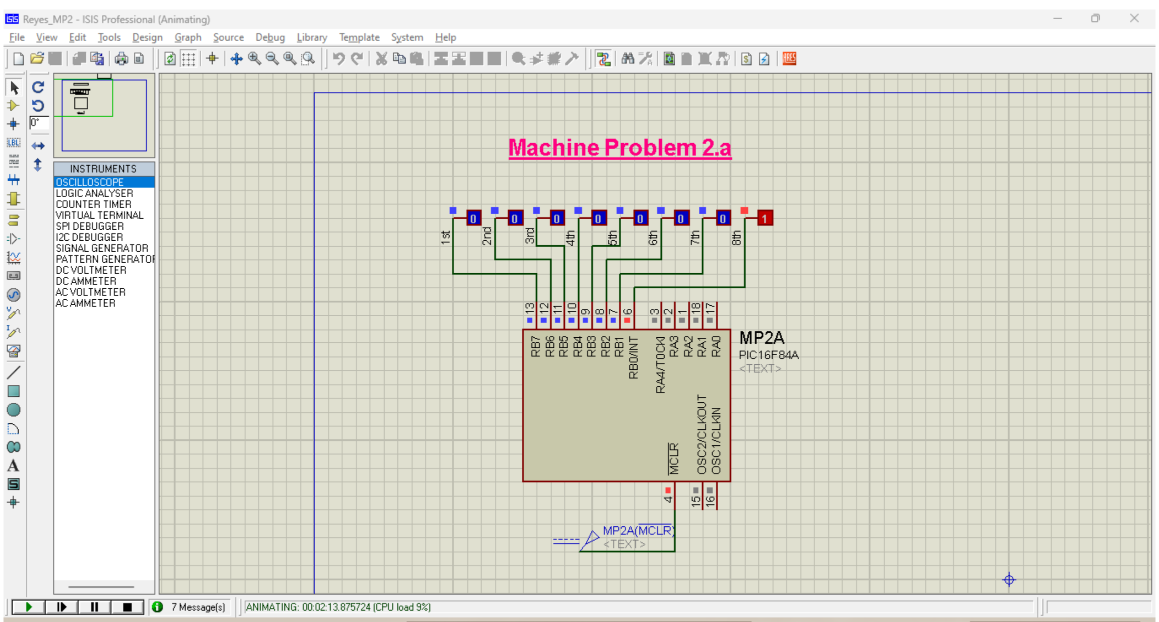
f. RB2


Reyes_MP2 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
INSTRUMENTS
OSCILLOSCOPE
LOGIC ANALYSER
COUNTER TIMER
VIRTUAL TERMINAL
SPI DEBUGGER
I2C DEBUGGER
SIGNAL GENERATOR
PATTERN GENERATOR
DC VOLTMETER
DC AMMETER
AC VOLTMETER
AC AMMETER
Machine Problem 2.a
1st 2nd 3rd 4th 5th 6th 7th 8th
0 0 0 0 0 1 0 0
RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0/INT RA4/T0CKI RA3 RA2 RA1 RA0
MCLR OSC2/CLKOUT OSC1/CLKIN
MP2A
PIC16F84A
<TEXT>
MP2A(MCLR)
7 Message(s)
ANIMATING: 00:01:29.600000 (CPU load 14%)

g. RB1

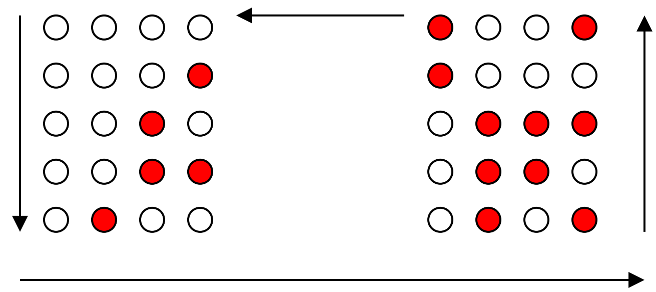

Reyes_MP2 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
INSTRUMENTS
OSCILLOSCOPE
LOGIC ANALYSER
COUNTER TIMER
VIRTUAL TERMINAL
SPI DEBUGGER
I2C DEBUGGER
SIGNAL GENERATOR
PATTERN GENERATOR
DC VOLTMETER
DC AMMETER
AC VOLTMETER
AC AMMETER
Machine Problem 2.a
1st
2nd
3rd
4th
5th
6th
7th
8th
MP2A
PIC16F84A
MP2A(MCLR)
7 Message(s)
ANIMATING: 00:01:46.750000 (CPU load 14%)

h. RB0

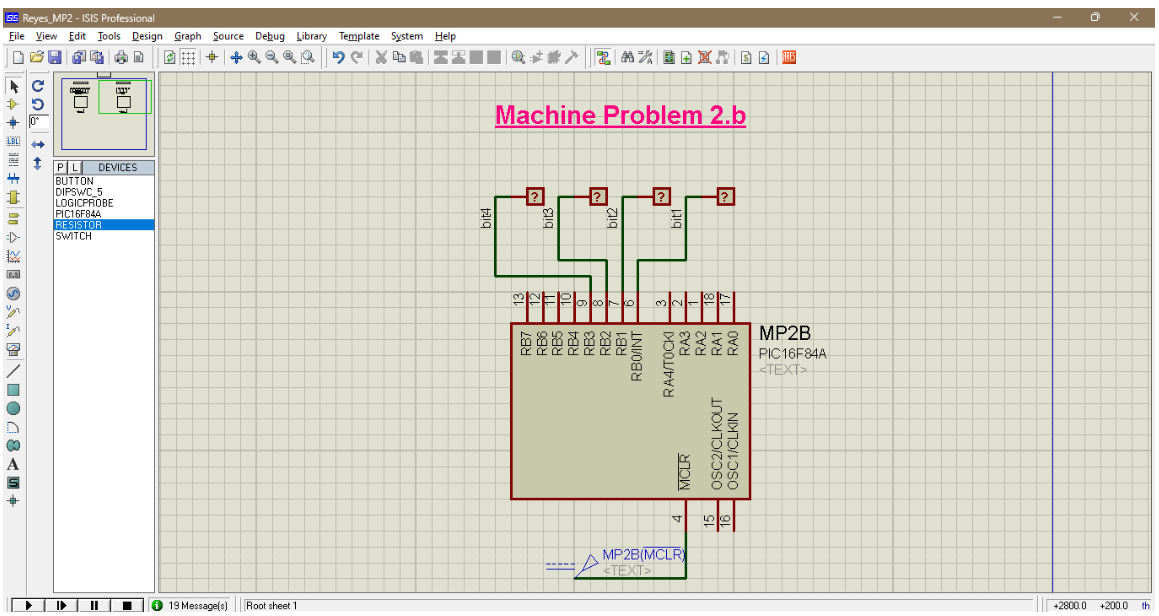

Reyes_MP2 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
INSTRUMENTS
OSCILLOSCOPE
LOGIC ANALYSER
COUNTER TIMER
VIRTUAL TERMINAL
SPI DEBUGGER
I2C DEBUGGER
SIGNAL GENERATOR
PATTERN GENERATOR
DC VOLTMETER
DC AMMETER
AC VOLTMETER
AC AMMETER
Machine Problem 2.a
1st
2nd
3rd
4th
5th
6th
7th
8th
MP2A
PIC16F84A
MP2A(MCLR)
7 Message(s)
ANIMATING: 00:02:13.875724 (CPU load 9%)

SETUP B: Counter A

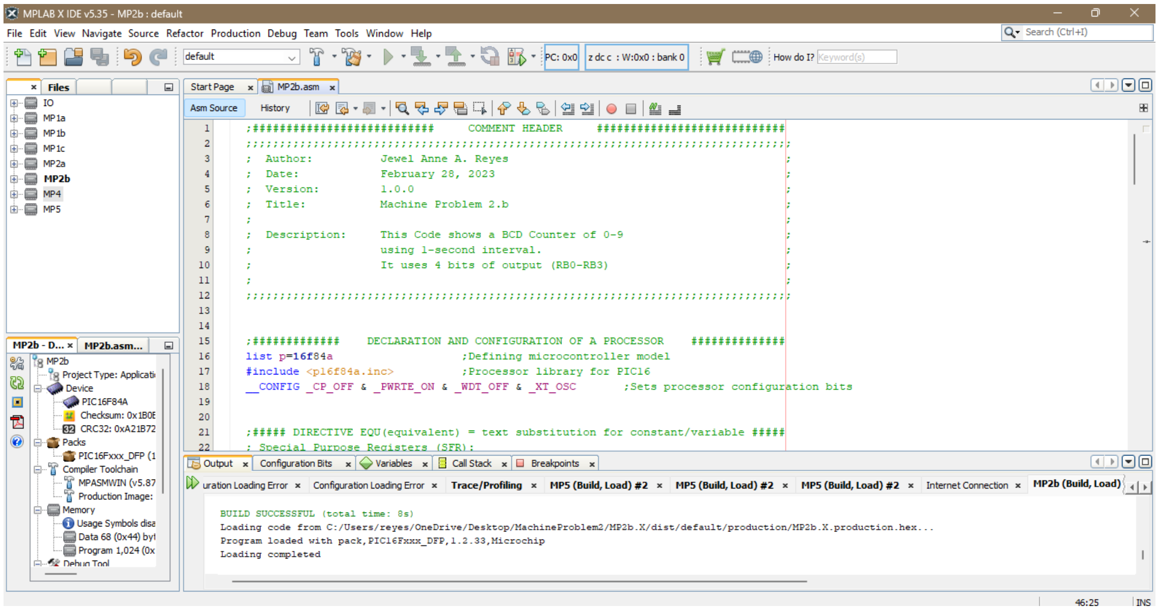
B. Using the same Setup Provide a BCD Counter 0-9 using 1-second interval. Use RB0 to RB3 for the Display.



Schematic:

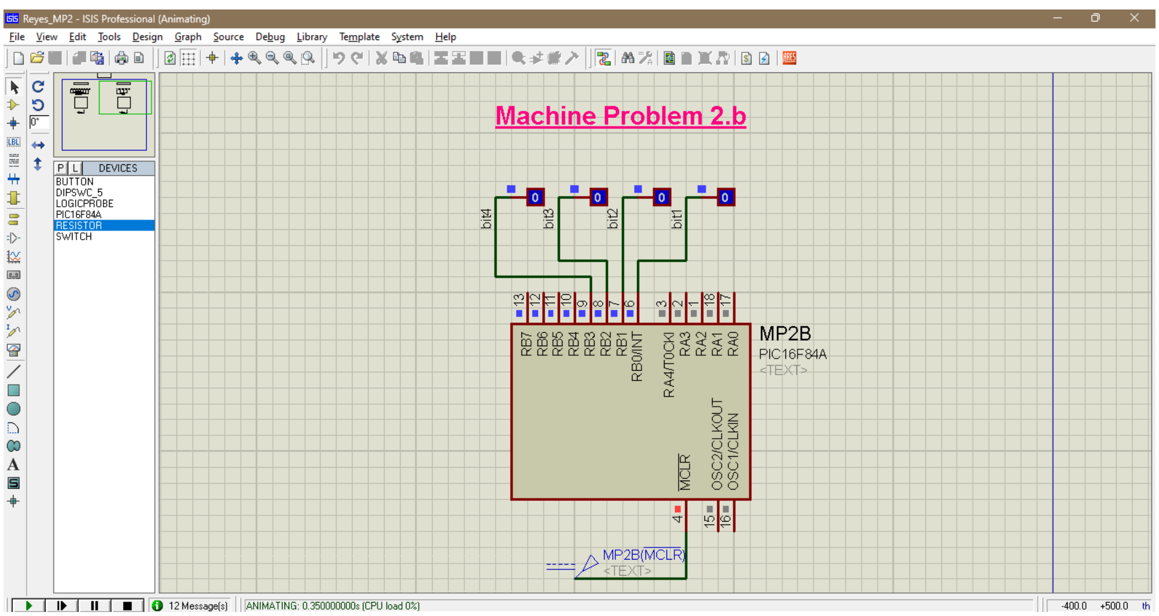


Program Build Successful:

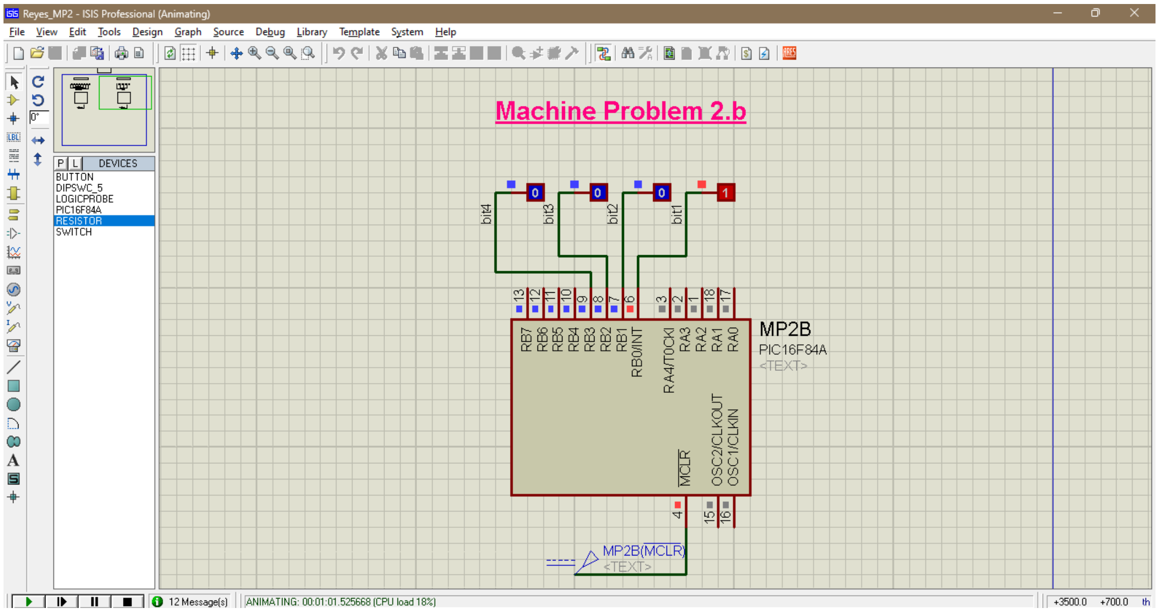


Outputs:

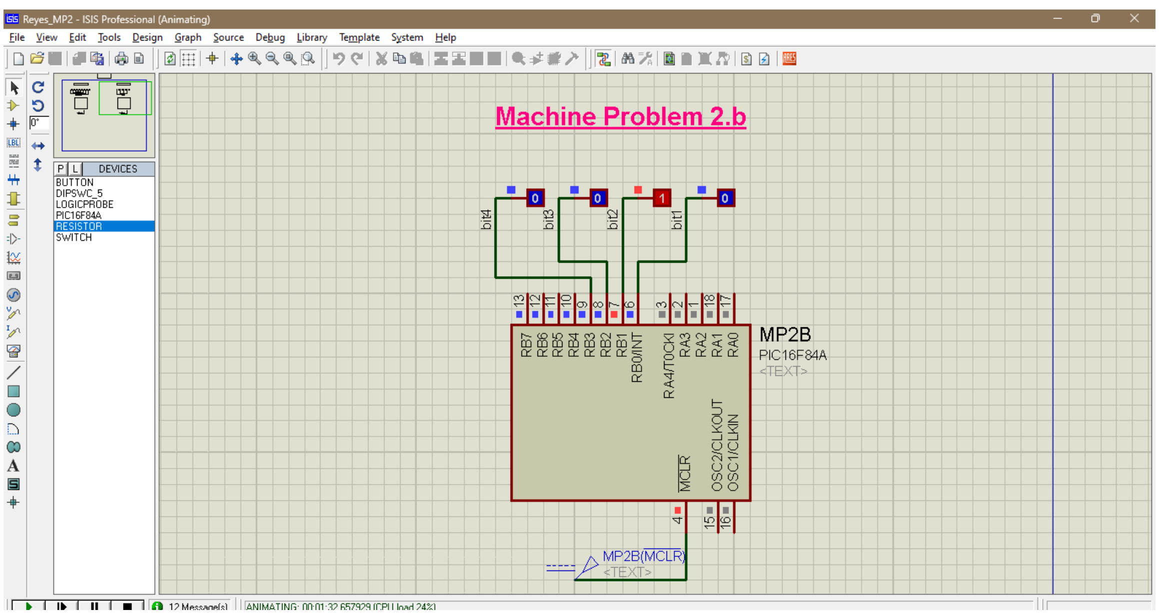
a. Decimal == 0:



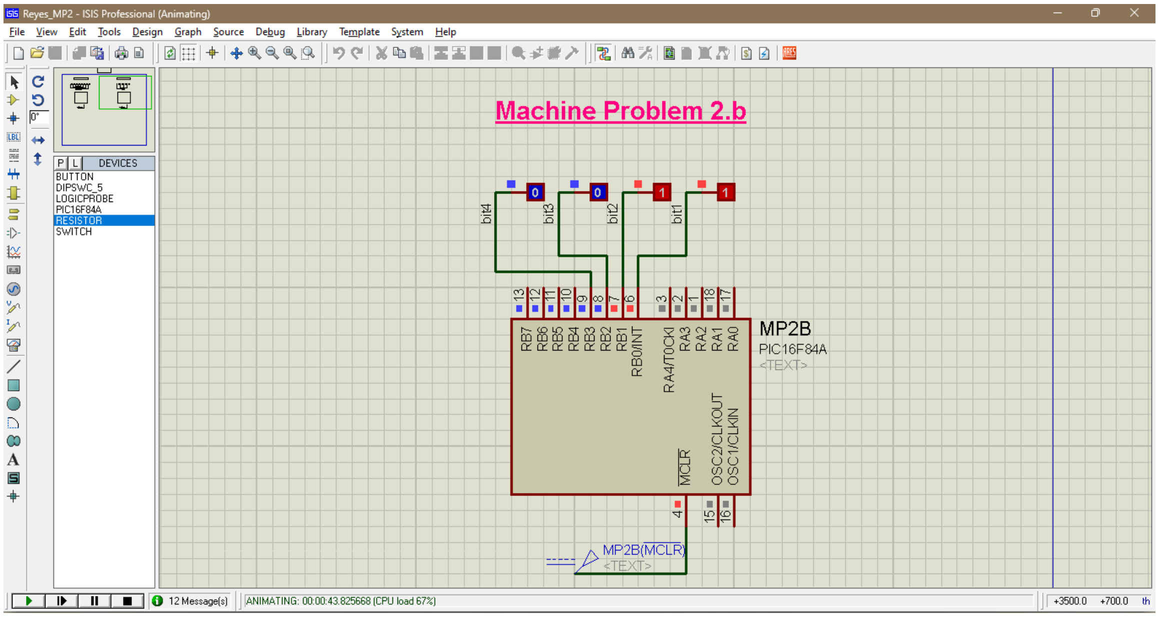
b. Decimal == 1:



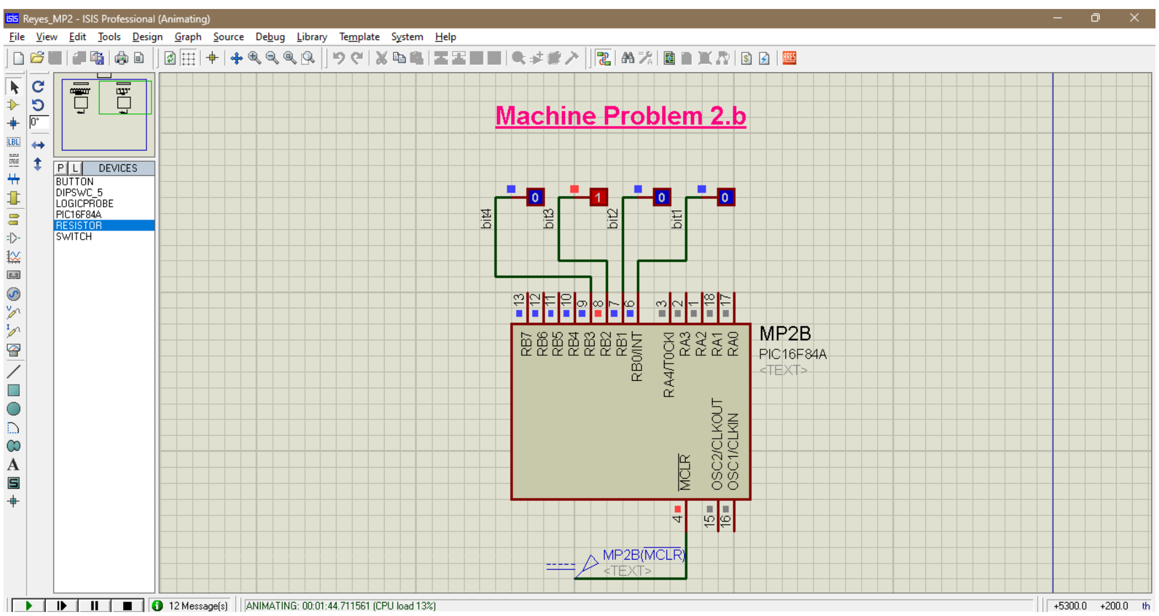
c. Decimal == 2:



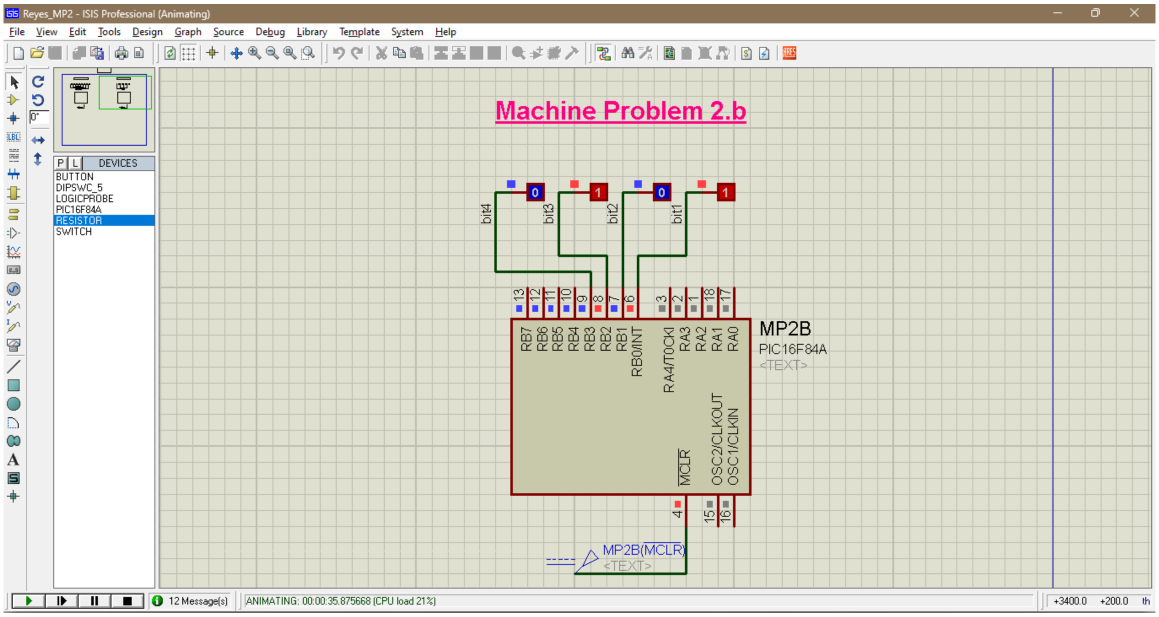
d. Decimal == 3:



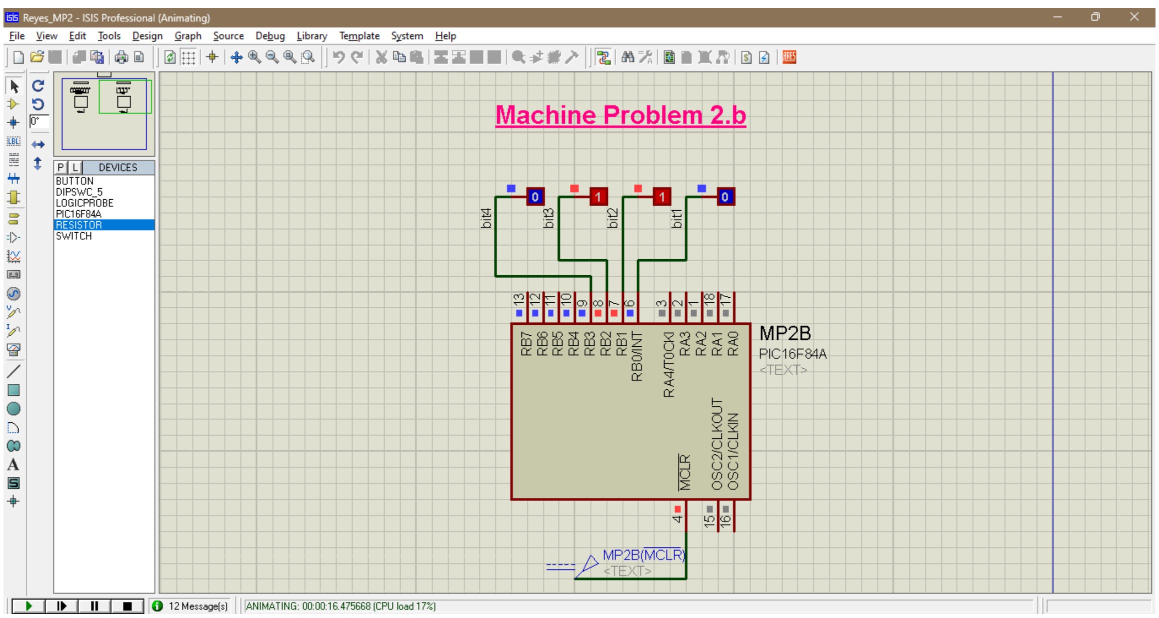
e. Decimal == 4:



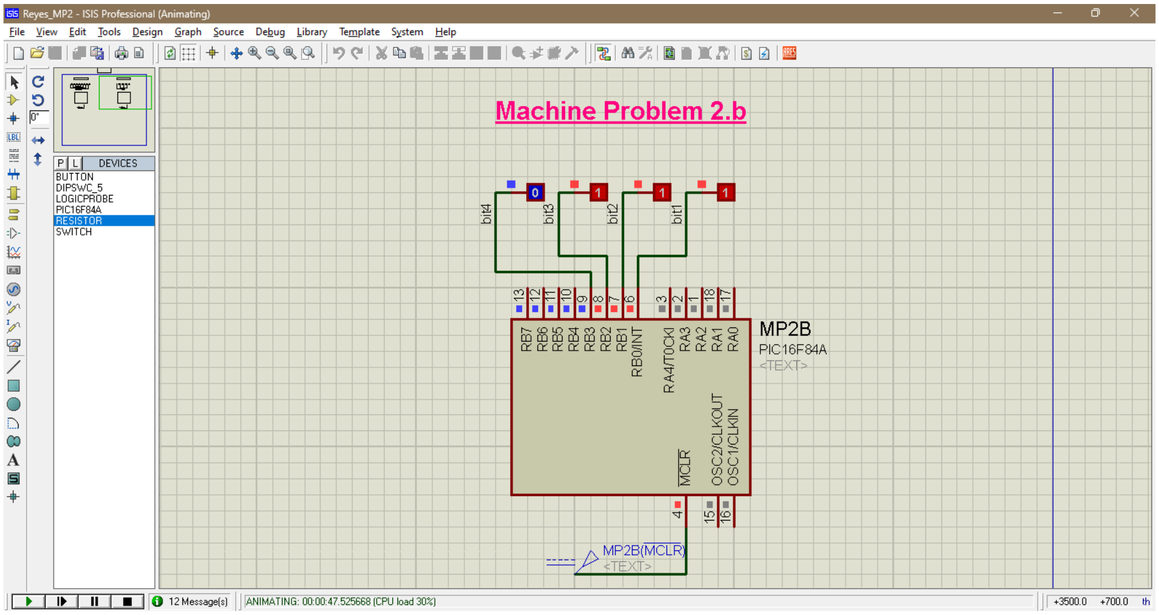
f. Decimal == 5:



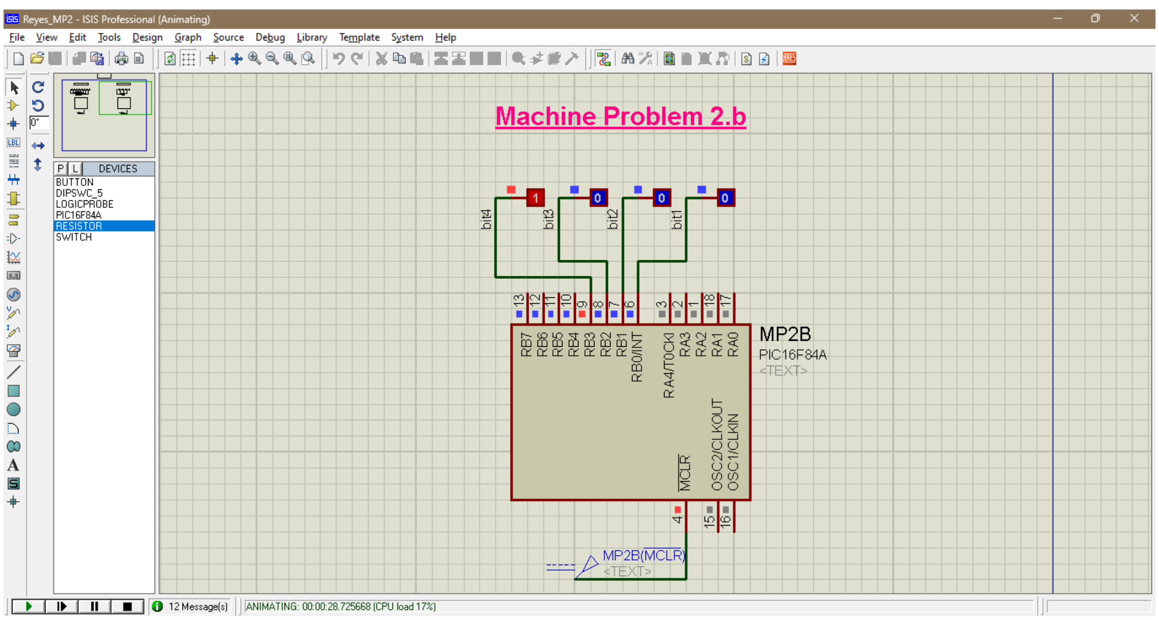
g. Decimal == 6:



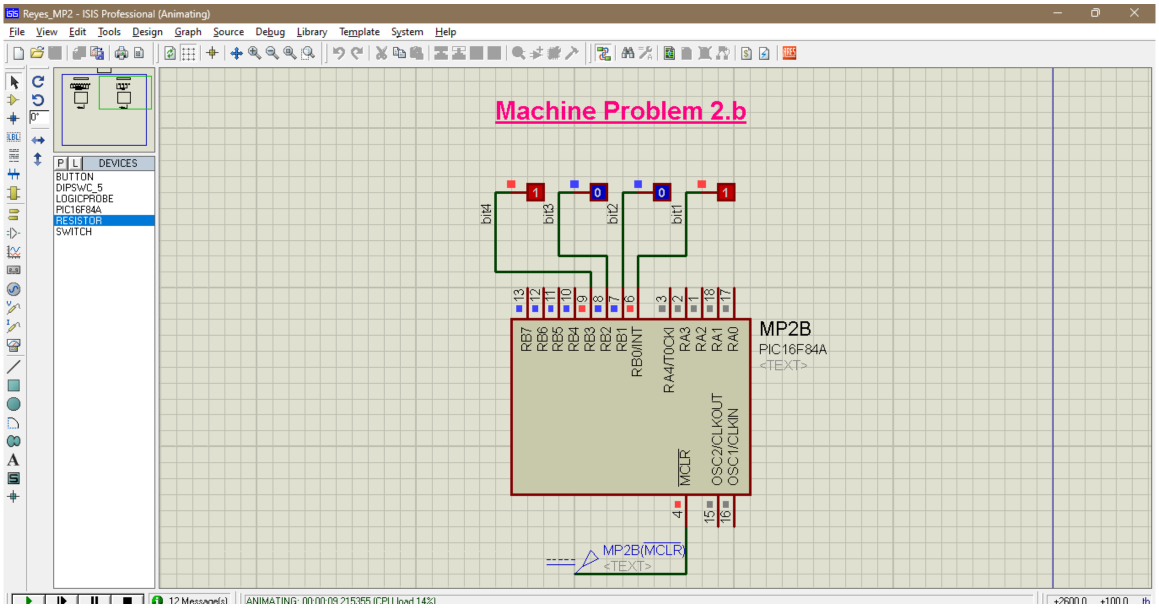
h. Decimal == 7:



i. Decimal == 8:

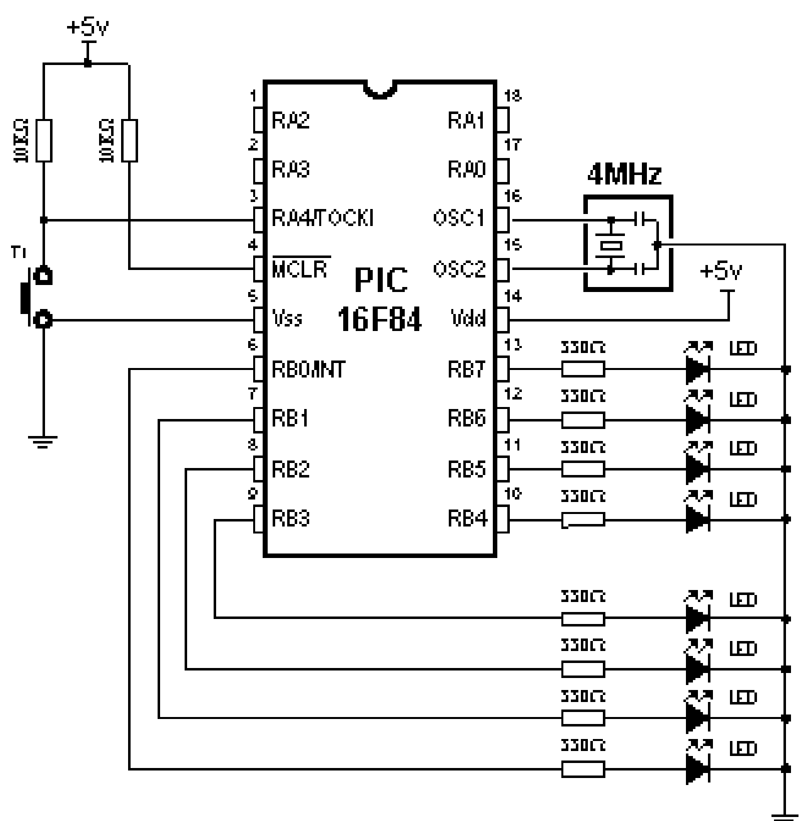


j. Decimal == 9:



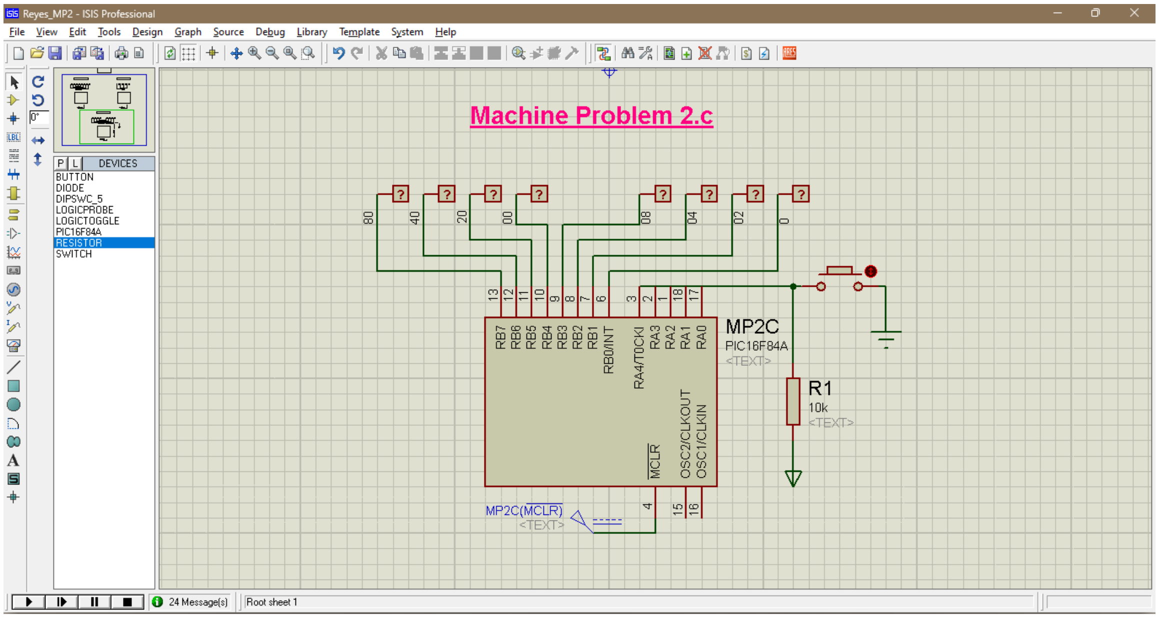
SETUP C: Counter C

C. Configure RA4 as input pin, and PORTB as Output PORT. Create a code that will Count 00H to FFH in every negative edge input in RA4. Refer to setup below:

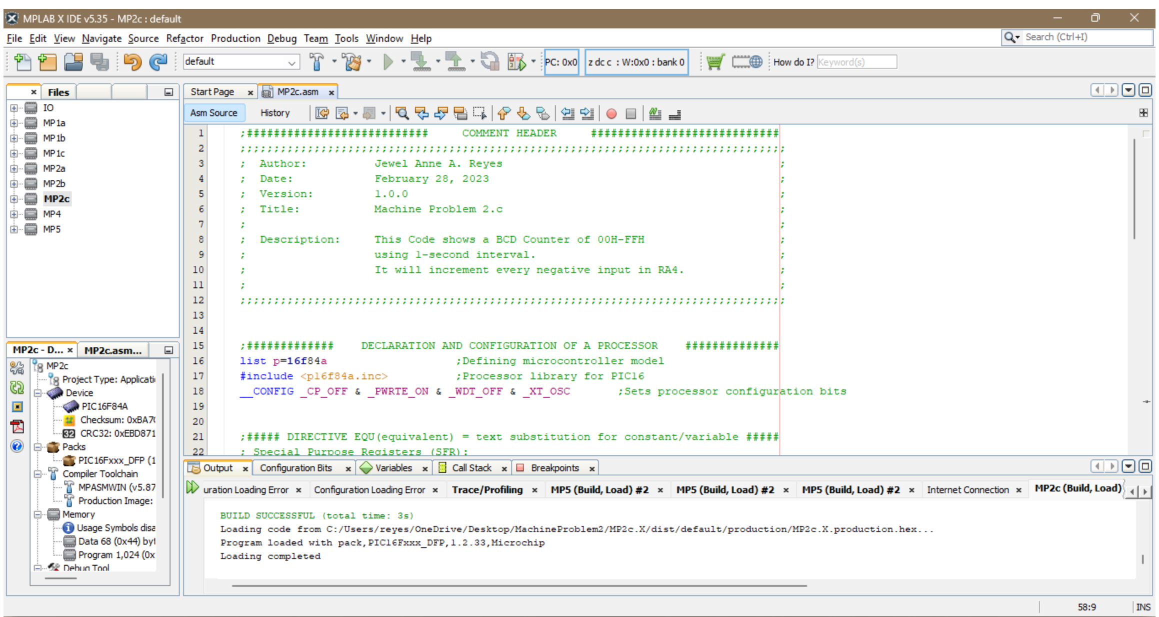


PIC16F84A

Schematic:

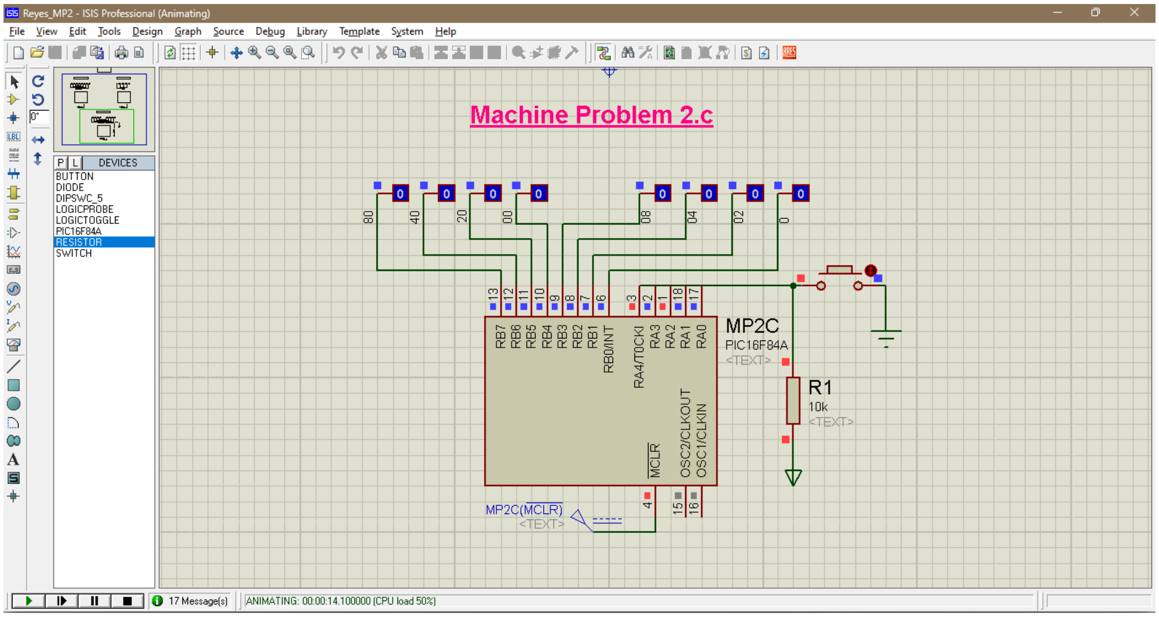


Program Build Successful:

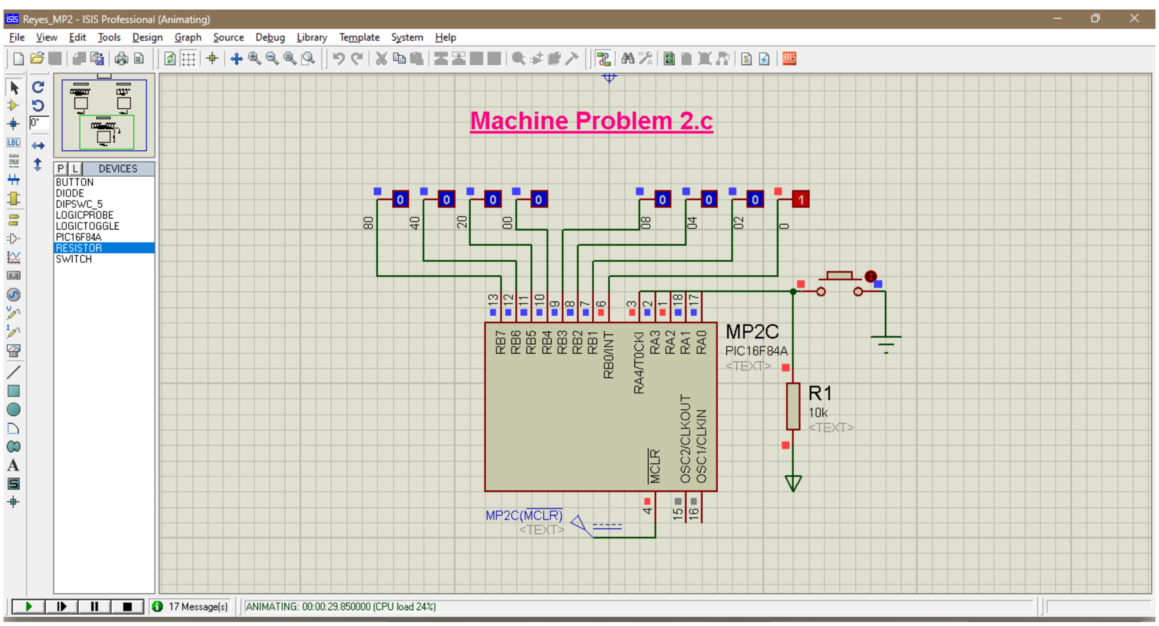


Outputs:

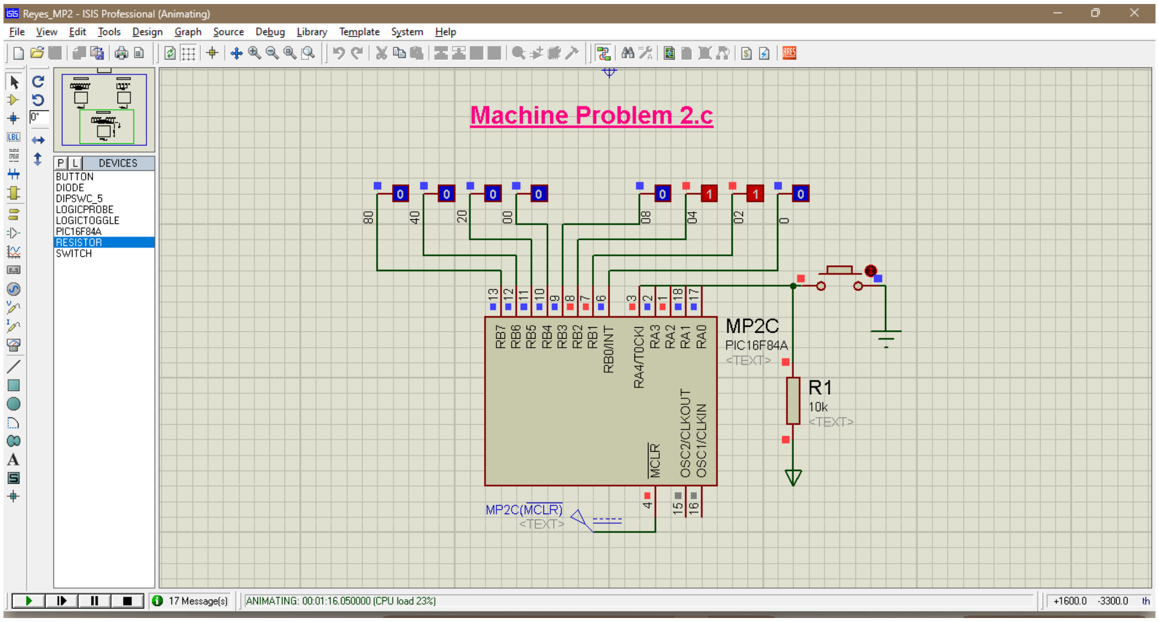
a. 00H:



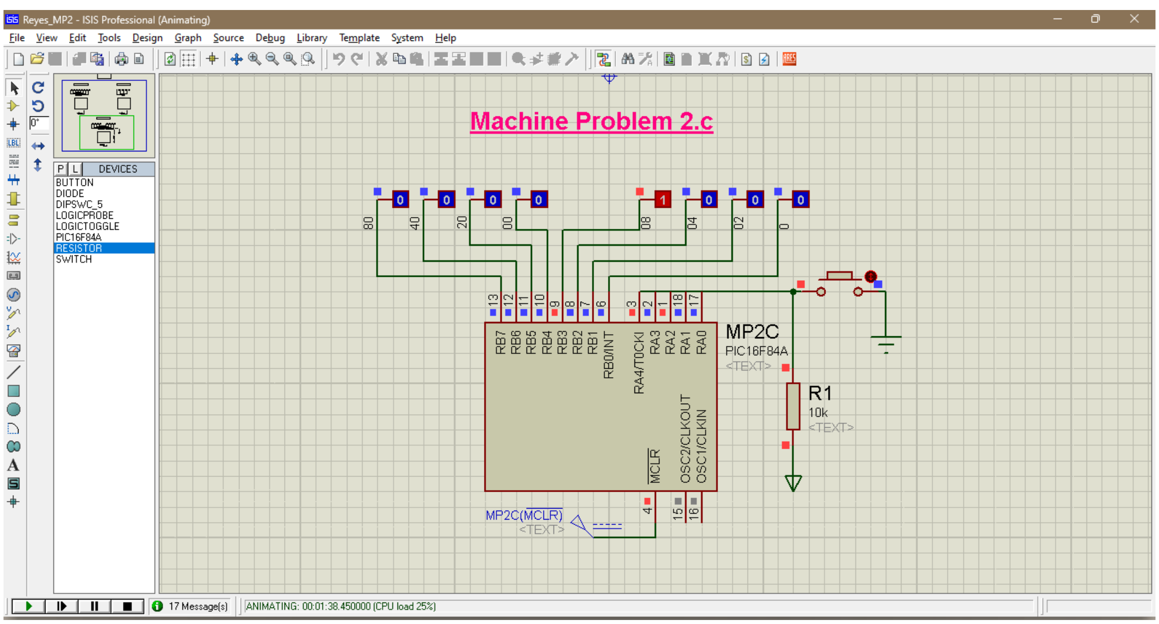
b. 01H:



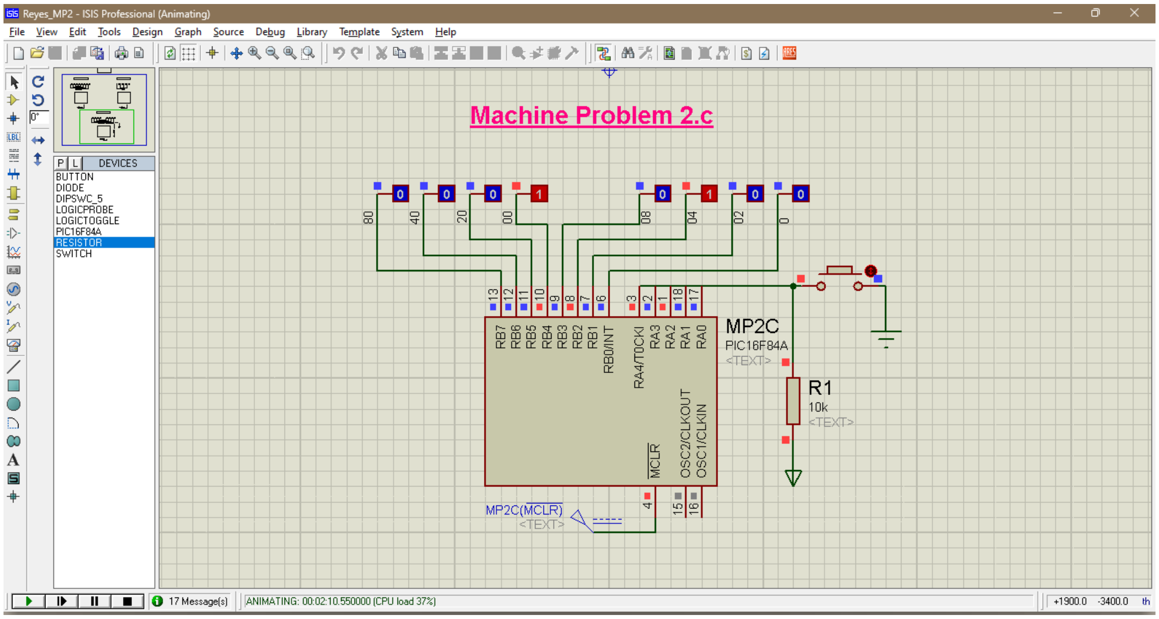
c. 06H:


Reyes_MP2 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
DEVICES
BUTTON
DIODE
DIPSWC_5
LOGICPROBE
LOGICTOGGLE
PIC16F84A
RESISTOR
SWITCH
Machine Problem 2.c
MP2C
PIC16F84A
R1
10k
MP2C(MCLR)
17 Message(s)
ANIMATING: 00:01:16.050000 (CPU load 23%)

d. 08H:


Reyes_MP2 - ISIS Professional (Animating)
File View Edit Tools Design Graph Source Debug Library Template System Help
DEVICES
BUTTON
DIODE
DIPSWC_5
LOGICPROBE
LOGICTOGGLE
PIC16F84A
RESISTOR
SWITCH
Machine Problem 2.c
MP2C
PIC16F84A
R1
10k
MP2C(MCLR)
17 Message(s)
ANIMATING: 00:01:38.450000 (CPU load 25%)

e. 14H:



Observation:

I have noticed a different way of storing data or setting the direction or destination of byte-oriented instructions other than 0 (destination is W register) and 1 (destination is itself). We can set or store the data in the W register by setting the d to 'W' and we can set it to itself by setting the d to 'F'.

I have decided to use the instruction RETLW for the Setup A: Running Lights and discover different ways of having one LED to shift from RB7 to RB0 and vice versa. This instruction return literal 'k' in W register and the program counter was loaded from the top pf the stack. Although using RLF and RRF requires lesser lines than the RETLW, I chose to use RETLW to try different commands and find out what would be the result. On the Setup B: Counter, I used the MOVLW instruction set and directly show the output from W register to the PORTB.

I also have computed the delay when using 4 MHz to have an output of 1-second interval between output lights. The detailed computation is in the source code. I found out that the number of cycles the routine I used is 1,003,834 us which is equivalent to 1.003834 seconds. The formula is deduced as follows:

Let's first ask ourselves how many cycles a program like the following takes:

DECFSZ x, f == This line runs x times, x-1 times does not jump and 1 time jumps, so the number of cycles is (x-1) +2 goto loop.

This runs x-1 times so cycles are 2 (x-1) in total, cycles are (x-1) + 2 + 2 (x-1) = 3x-1 cycles.

Let's consider that x is between 1 and 256 (includes 1 and 256, if x is 0, it equals 256). So the number of cycles of the complete program is:

Counter cycles A The first time when B has its initial value 3A-1 The following times of counter B (B-1) * (3 * 256-1) The next times of counter C (C-1) * 256 * 767 Counter B cycles The first time when C is 6 3B-1 The following times of C (C-1) * 767 Counter C cycles The first and only time 3C-1 The call, the Return and 6 initialization lines 10.

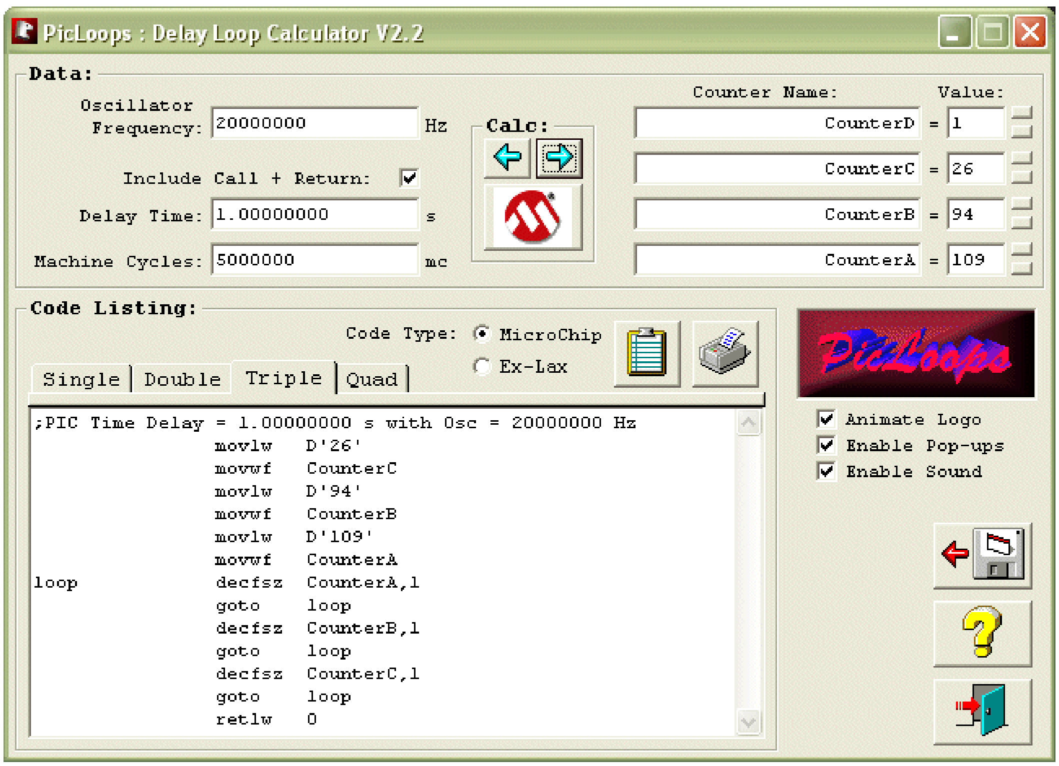
Total: 3A + 770B + 197 122C-197879

Upon doing the Setup C, I have noticed that there is an instruction INCFSZ which means increment F skip if zero. I wonder why there is no INCFSS or skip is set like the one with BTFSS. If only there is such an instruction, I think the Setup C would be easier as it will only require lesser commands and instruction cycle.

Conclusion/ Recommendation:

Substituting A for 168, B = 24 and C = 6 gives the value of 1,003,837 better values would have been A = 172 B = 19 C = 6 which produces 999,999 adding a nop in variable initialization would give 1,000,000

I have not tried the PicLoops application but apparently, it can help creating code of delay specifically with its delay time, oscillator frequency, machine cycles, and more. I think it would be great to use this especially when applying in real-life PIC 16F84A to have a more accurate computation. The image below shows the UI of the application. Here is the link: http://www.biltronix.com/picloops.html



Source Code: (4 asm)

(Please Attach the asm file)

```
;###########################     COMMENT HEADER            ##############################
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Author:          Jewel Anne A. Reyes                                              ;
;  Date:    February 27, 2023                                  ;
;  Version:         1.0.0                                                ;
;  Title:    Machine Problem 2.a                                         ;
;                                                                        ;
;  Description:     This Code shows the running light of                 ;
;                   1 LED or logic probe with 1-sec interval.            ;
;                   It will start from RB7 to RB0 and vice versa.        ;
;                   There are diff methods to produce running lights.    ;
;                   This code uses a data lookup table.                  ;
;                                                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;#############    DECLARATION AND CONFIGURATION OF A PROCESSOR    ##############
list p=16f84a                    ;Defining microcontroller model
#include <p16f84a.inc>                  ;Processor library for PIC16
__CONFIG _CP_OFF & _PWRTE_ON & _WDT_OFF & _XT_OSC        ;Sets processor configuration bits


;##### DIRECTIVE EQU(equivalent) = text substitution for constant/variable #####
; Special Purpose Registers (SFR):
STATUS          EQU 03H                 ;STATUS equivalent to 03H File Address
PORTA           EQU 05H
PORTB           EQU 06H
TRISA     EQU 85H
TRISB     EQU 86H
; General Purpose Registers (GPR):
    cblock          0x20             ;start of general purpose registers
            count                    ;used in table read routine
            CounterA                 ;used in delay routine
            CounterB                 ;used in delay routine
            CounterC                 ;used in delay routine
    endc

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
            ORG 00H                         ;Start assembling lines below this statement at RESET vector (00h)
                GOTO Initial
            ORG 04H                         ;Start assembling lines below this statement at Peripheral Interrupt Vector (04H)
                RETFIE                  ;Return from interrupt
```

```
            RETFIE              ;Return from interrupt
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Initial
        BSF STATUS, RP0             ;Bit Set F (goes to Bank 1)

   ;Setting PORTB as Output
        MOVLW b'00000000';Move to W Register the binary input 00H
        MOVWF TRISB                 ;Transfer W Register literals to F Register 86H

        BCF STATUS, RP0             ;Bit Clear F (goes to Bank 0)
        CLRF PORTB          ;set all outputs low
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Start
        CLRF  count         ;set counter register to zero
Read
        MOVF count, w       ;put counter value in W
        CALL  Table
        MOVWF       PORTB
        CALL  Delay         ;1 second delay
        INCF  count,  w     ;increment F
        XORLW       d'14'           ;check for last (14th) entry
        BTFSC       STATUS,         Z
        GOTO Start          ;if start from beginning
        INCF  count,  f     ;else do next
        GOTO Read
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Table
   ADDWF  PCL, f            ;data table for bit pattern
   RETLW  b'10000000'               ;RETLW Return with Literal in W
   RETLW  b'01000000'
   RETLW  b'00100000'
   RETLW  b'00010000'
   RETLW  b'00001000'
   RETLW  b'00000100'
   RETLW  b'00000010'
   RETLW  b'00000001'
   RETLW  b'00000010'
   RETLW  b'00000100'
   RETLW  b'00001000'
   RETLW  b'00010000'
   RETLW  b'00100000'
   RETLW  b'01000000'
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Delay
   MOVLW D'6'                       ;1 us
```

```
    MOVWF CounterC          ;1 us
    MOVLW D'24'                     ;1 us
    MOVWF CounterB          ;1 us
    MOVLW D'167'            ;1 us
    MOVWF CounterA                  ;1 us
loop
    DECFSZ CounterA,1               ;167 x 1 cycle + 1 cycle = 168 us
    GOTO loop               ;166 * 2 us = 332 us
    DECFSZ CounterB,1               ;24 x 1 cycle + 1 cycle = 25 us
    GOTO loop               ;24 * 2 us = 48 us
    DECFSZ CounterC,1               ;6 x 1 cycle + 1 cycle = 7 us
    GOTO loop               ;5 * 2 us = 10 us
    NOP                             ;1 us
                            ;TOTAL = 597 us
                            ;3A + 770B + 197,122C - 197,879
                            ;3(167) + 770(24) + 197,122(6) - 197,879
                            ;501 + 18,480 + 1,182,732 - 197,879
                            ;1,201,713 - 197,879
                            ;DELAY = 1,003,834 us == 1.003 seconds
    RETURN                  ;2 us
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
END
```

```
;###########################     COMMENT HEADER            ############################
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Author:          Jewel Anne A. Reyes                                      ;
;  Date:    February 28, 2023                                   ;
;  Version:         1.0.0                                                ;
;  Title:   Machine Problem 2.b                                          ;
;                                                                        ;
;  Description:     This Code shows a BCD Counter of 0-9                 ;
;                   using 1-second interval.                             ;
;                   It uses 4 bits of output (RB0-RB3)                   ;
;                                                                        ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;#############    DECLARATION AND CONFIGURATION OF A PROCESSOR    ##############
list p=16f84a                   ;Defining microcontroller model
#include <p16f84a.inc>                 ;Processor library for PIC16
__CONFIG _CP_OFF & _PWRTE_ON & _WDT_OFF & _XT_OSC       ;Sets processor configuration bits


;##### DIRECTIVE EQU(equivalent) = text substitution for constant/variable #####
; Special Purpose Registers (SFR):
STATUS          EQU 03H                 ;STATUS equivalent to 03H File Address
PORTA           EQU 05H
PORTB           EQU 06H
TRISA    EQU 85H
TRISB    EQU 86H
; General Purpose Registers (GPR):
cblock 0x20                     ;start of general purpose registers
        CounterA                ;used in delay subroutine
        CounterB                ;used in delay subroutine
        CounterC                ;used in delay subroutine
endc

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ORG 00H                         ;Start assembling lines below this statement at RESET vector (00h)
           goto Initial
        ORG 04H                         ;Start assembling lines below this statement at Peripheral Interrupt Vector (04H)
           retfie               ;Return from interrupt
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Initial
        BSF STATUS, 5                   ;Bit Set F (goes to Bank 1)

   ;Setting PORTB as Output
        MOVLW b'00000000';Move to W Register the binary input 00H
```

```
        MOVWF TRISB             ;Transfer W Register literals to F Register 86H

        BCF STATUS, 5           ;Bit Clear F (goes to Bank 0)
        BCF STATUS, 0           ;Bit Clear F (Move the Carry in the register)

        goto Main
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Main
        MOVLW b'00000000';Set 1st bit into decimal 0
        MOVWF PORTB             ;Transfer W Register literals to F Register 06H
        Call Delay          ;Subroutine of 1 second delay

        MOVLW b'00000001';Decimal == 1
        MOVWF PORTB
        Call Delay

        MOVLW b'00000010';Decimal == 2
        MOVWF PORTB
        Call Delay

        MOVLW b'00000011';Decimal == 3
        MOVWF PORTB
        Call Delay

        MOVLW b'00000100';Decimal == 4
        MOVWF PORTB
        Call Delay

        MOVLW b'00000101';Decimal == 5
        MOVWF PORTB
        Call Delay

        MOVLW b'00000110';Decimal == 6
        MOVWF PORTB
        Call Delay

        MOVLW b'00000111';Decimal == 7
        MOVWF PORTB
        Call Delay

        MOVLW b'00001000';Decimal == 8
        MOVWF PORTB
        Call Delay

        MOVLW b'00001001';Decimal == 9
```

```
        MOVWF PORTB
        Call Delay

        goto Main

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Delay
   MOVLW D'6'                 ;1 us
   MOVWF CounterC        ;1 us
   MOVLW D'24'                ;1 us
   MOVWF CounterB        ;1 us
   MOVLW D'167'          ;1 us
   MOVWF CounterA             ;1 us
loop
   DECFSZ CounterA,1          ;167 x 1 cycle + 1 cycle = 168 us
   GOTO loop             ;166 * 2 us = 332 us
   DECFSZ CounterB,1          ;24 x 1 cycle + 1 cycle = 25 us
   GOTO loop             ;24 * 2 us = 48 us
   DECFSZ CounterC,1          ;6 x 1 cycle + 1 cycle = 7 us
   GOTO loop             ;5 * 2 us = 10 us
   NOP                        ;1 us
                         ;TOTAL = 597 us
                         ;3A + 770B + 197,122C - 197,879
                         ;3(167) + 770(24) + 197,122(6) - 197,879
                         ;501 + 18,480 + 1,182,732 - 197,879
                         ;1,201,713 - 197,879
                         ;DELAY = 1,003,834 us == 1.003 seconds
   RETURN                ;2 us
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

end
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Author:          Jewel Anne A. Reyes                                   ;
;  Date:    February 28, 2023                                    ;
;  Version:         1.0.0                                              ;
;  Title:    Machine Problem 2.c                                       ;
;  Description:     This Code shows a BCD Counter of 00H-FFH              ;
;                   using 1-second interval.                           ;
;                   It will increment every negative input in RA4.     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;#############    DECLARATION AND CONFIGURATION OF A PROCESSOR    ##############
list p=16f84a                ;Defining microcontroller model
#include <p16f84a.inc>                ;Processor library for PIC16
__CONFIG _CP_OFF & _PWRTE_ON & _WDT_OFF & _XT_OSC        ;Sets processor configuration bits

;##### DIRECTIVE EQU(equivalent) = text substitution for constant/variable #####
; Special Purpose Registers (SFR):
STATUS          EQU 03H               ;STATUS equivalent to 03H File Address
PORTA           EQU 05H
PORTB           EQU 06H
TRISA     EQU 85H
TRISB     EQU 86H
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
          ORG 00H                     ;Start assembling lines below this statement at RESET vector (00h)
             goto Initial
          ORG 04H                     ;Start assembling lines below this statement at Peripheral Interrupt Vector (04H)
             retfie          ;Return from interrupt
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Initial
          BSF STATUS, 5               ;Bit Set F (goes to Bank 1)
   ;Setting PORTA as Input and PORTB as Output
          MOVLW b'00010000';Move to W Register the binary input 10H
          MOVWF TRISA                 ;Transfer W Register literals to F Register 85H
          MOVLW b'00000000';Move to W Register the binary input 00H
          MOVWF TRISB                 ;Transfer W Register literals to F Register 86H

          BCF STATUS, 5               ;Bit Clear F (goes to Bank 0)
          CLRF PORTB          ;Initialize PORTB to 0
          goto Main
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Main
   BTFSS PORTA, 4            ;Check if RA4 is high
   goto Main                 ;If it's high, keep waiting for falling edge
   BTFSC PORTA, 4            ;Check if RA4 is now low
   goto Main                 ;If it's still high, keep waiting for falling edge
   INCF PORTB, 1             ;If it's low, increment PORTB
   goto Main                 ;Wait for next falling edge
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
end
```

Additionals:

(Please attach any changes in the setup including images and schematics)