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

Name: _Jewel Anne Reyes____ **Date Submitted: Feb. 27, 2023**

Machine Problem # 1

Input and Output Interfacing

Objective:

- To be familiar with Input/Output Interfacing technique of PIC16F84A
- To configure the PORTS of MCU for I/O Interfacing.
- To enhance skills of Assembly Programming using MPLAB and Proteus.
- To provide Multiplication and Division of Input number to 2 using RLF and RRF instruction.
- To review the important registers used in PIC16F84A.
- To understand and analyze the accepted syntax of assembly language.
- To improve skills and knowledge on memory allocations.

Background:

A Microcontroller is defined as peripherals in one electronic component. External microcontroller's connector pin can be configured as input or output. In most cases I/O pin enables a microcontroller to communicate, control or read information. Information are written in codes in which microcontroller needs in order to be able to function. Software can not have any errors if we want the program and a device to function properly. Software can be written in different languages such as: Basic, C, pascal or assembler. Physically, that is a file on computer disc.

Term "port" refers to a group of pins on a microcontroller which can be accessed simultaneously, or on which we can set the desired combination of zeros and ones, or read from them an existing status. Physically, port is a register inside a microcontroller which is connected by wires to the pins of a microcontroller. Ports represent physical connection of Central Processing Unit with an outside world. Microcontroller uses them in order to monitor or control other components or devices.

Materials:

PIC16F84A PC

PIC Programmer (MPLAB) Power Supply

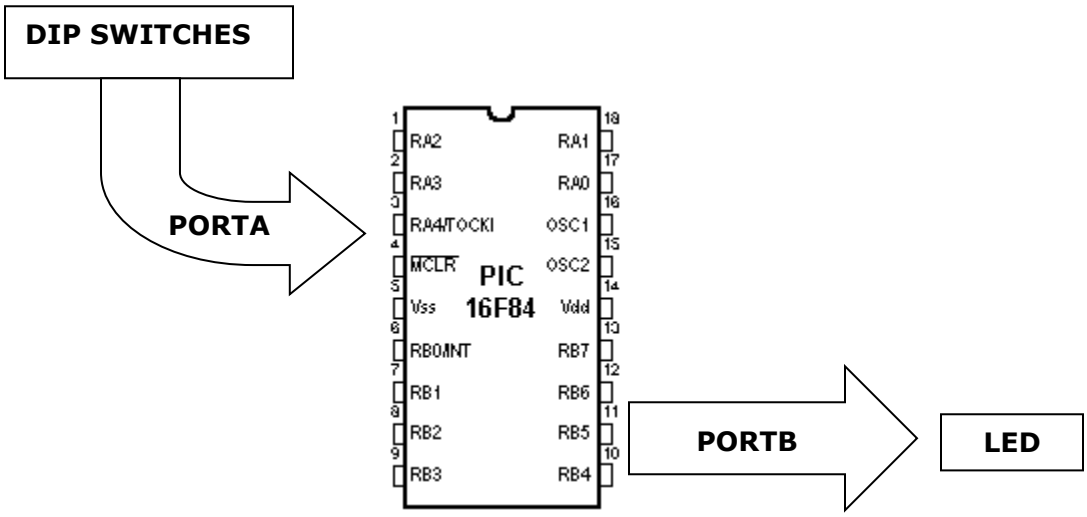
Test Jig

Procedure:

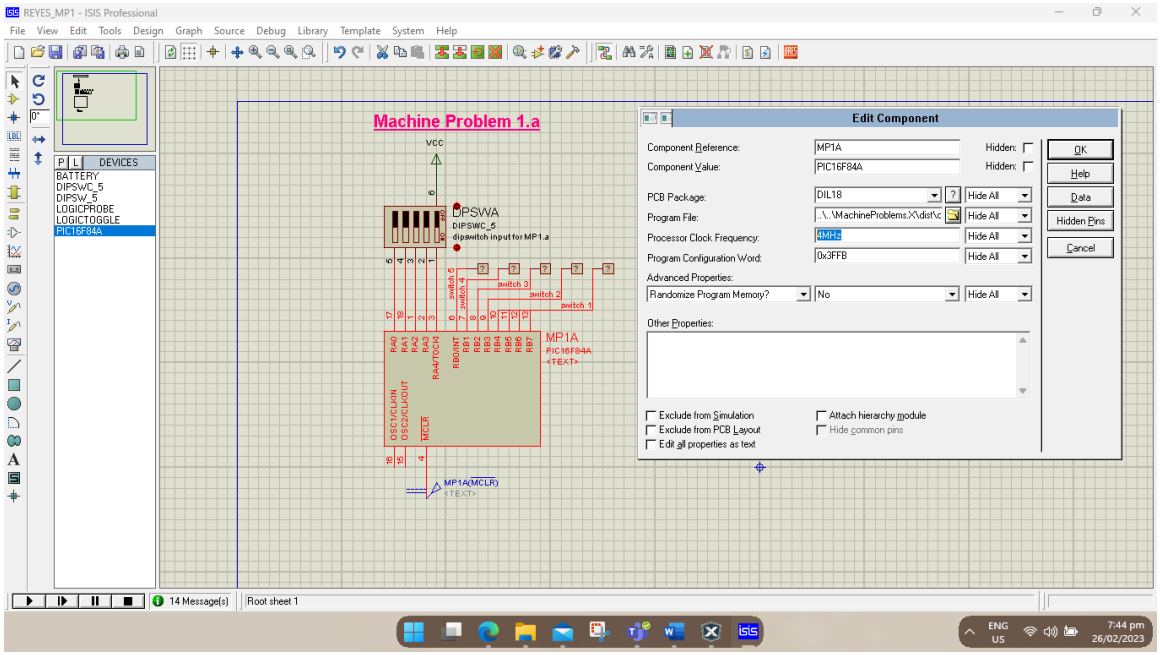
Perform the following Setup and provide a code for each setup

SETUP: Input and output Interfacing

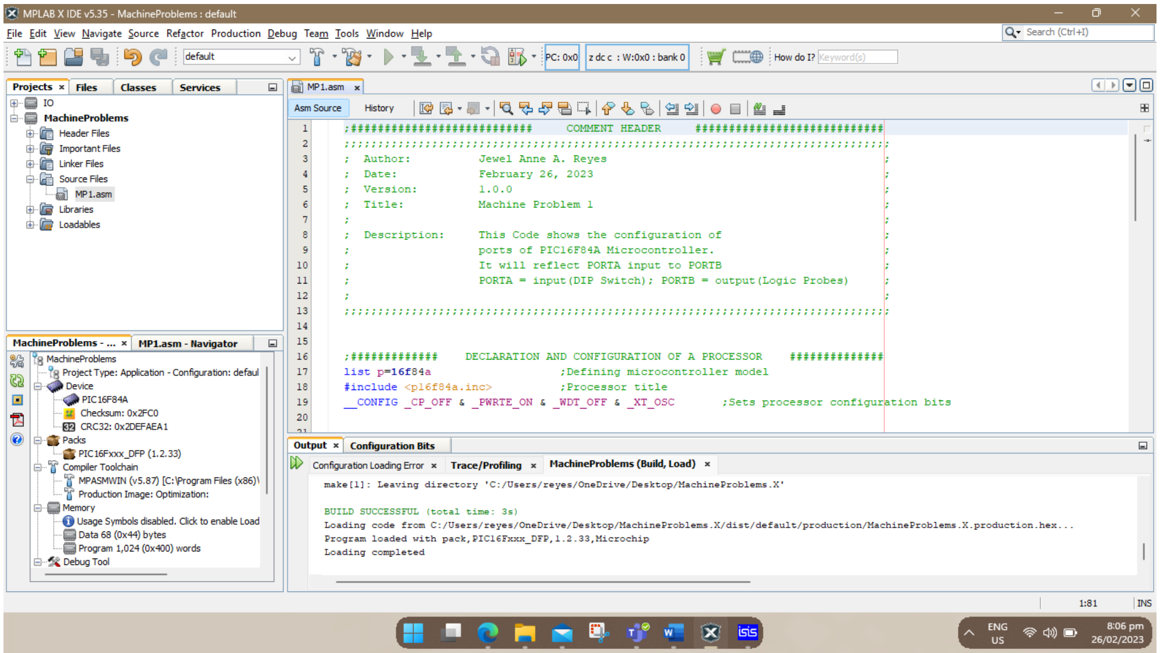
A. Configure PORTA as input and PORTB as output in the Test Jig. Connect the DIP switches on PORTA, while the Logic Probes must be connected to PORTB. Create a code in which whatever status of PORTA (DIP Switches) must reflect on PORTB (Logic Probes).



Schematic:

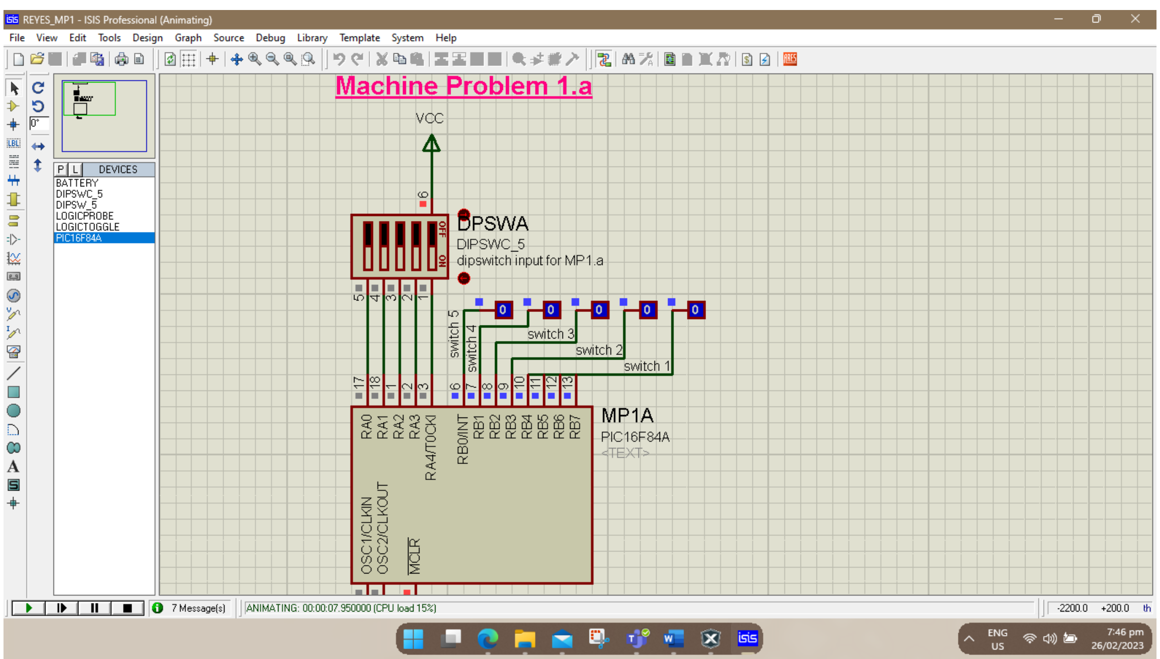


Program Build Successful:

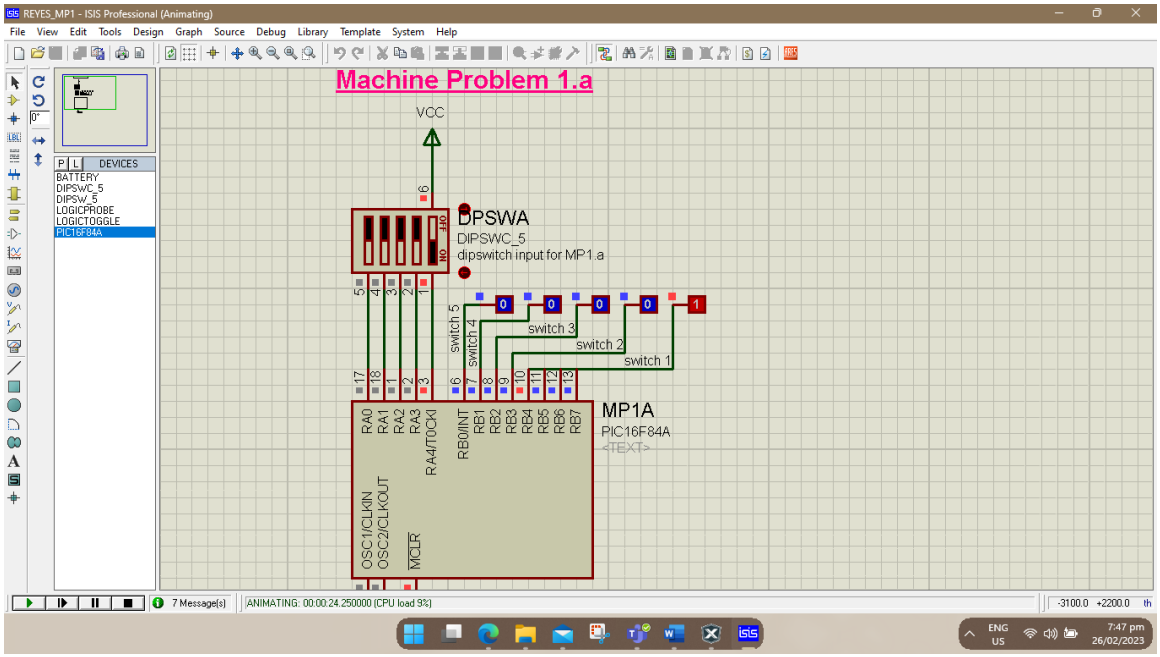


Outputs:

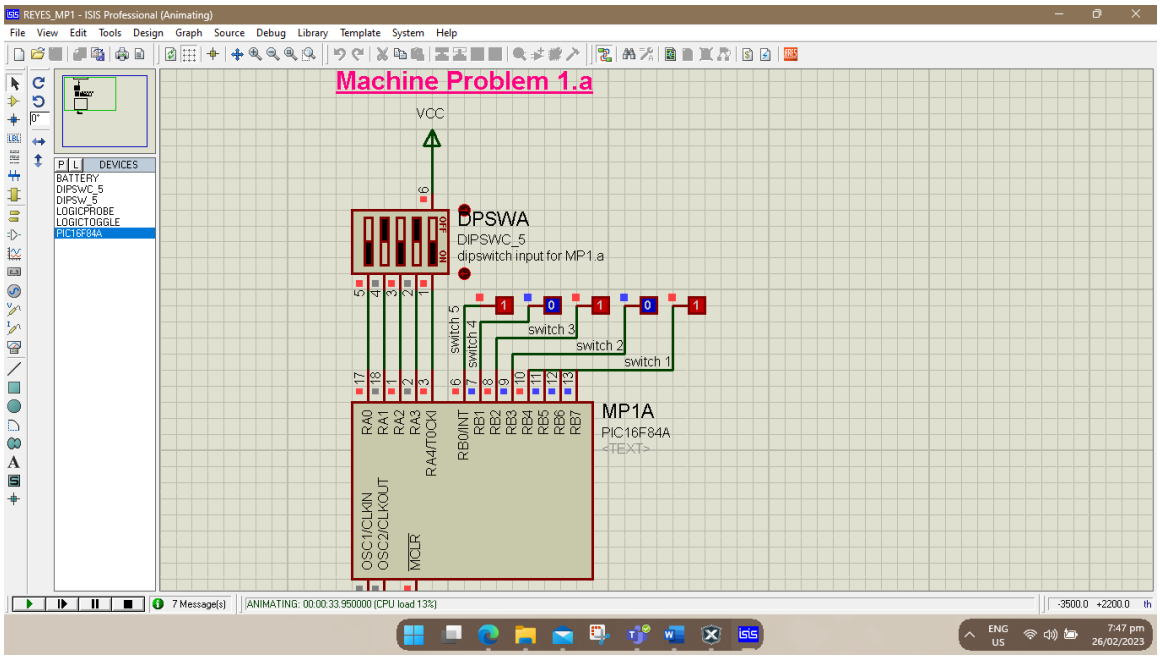
a. All logic 0:



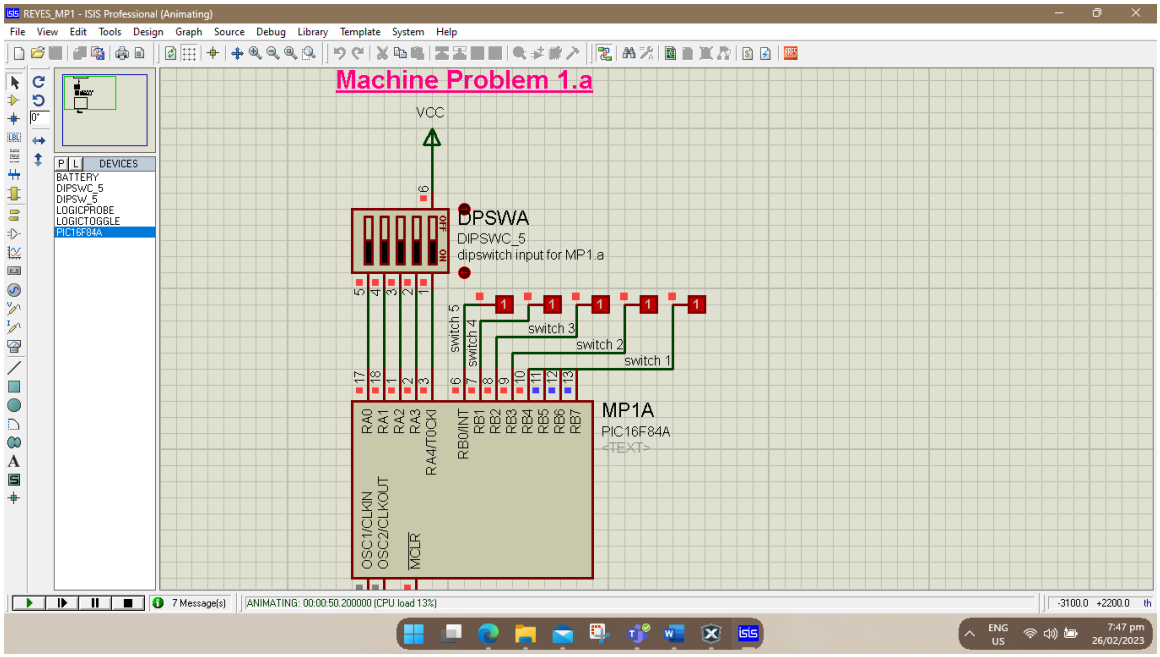
b. RA4 is Logic 1:



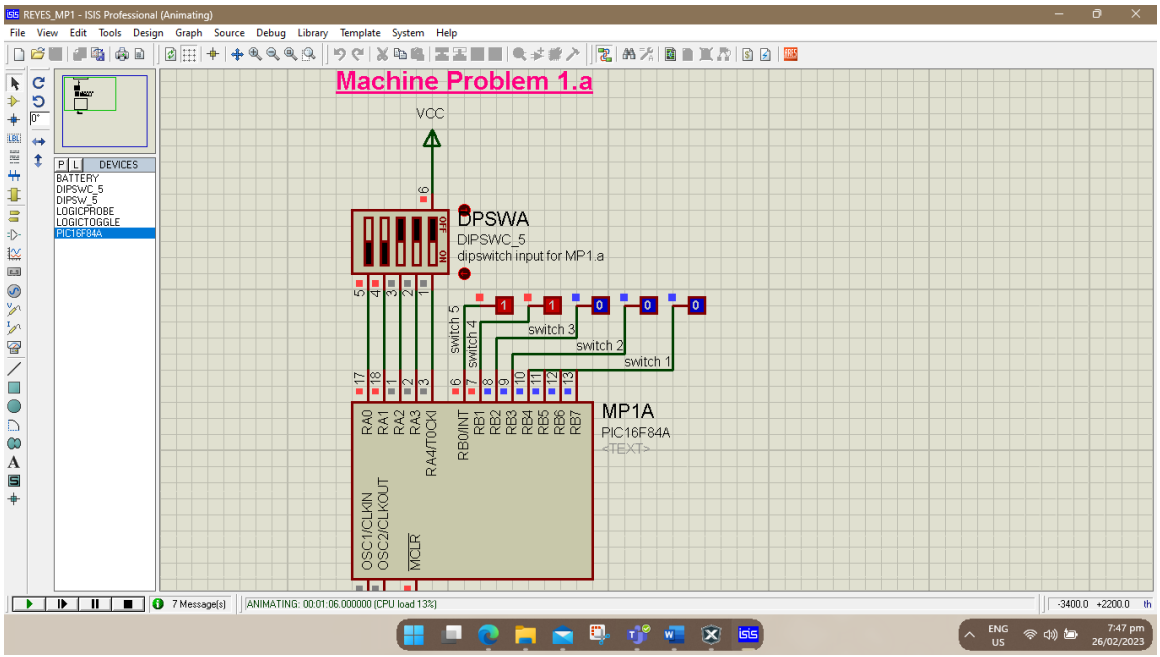
c. RA1 and RA3 are Logic 0:



d. All logic 1:

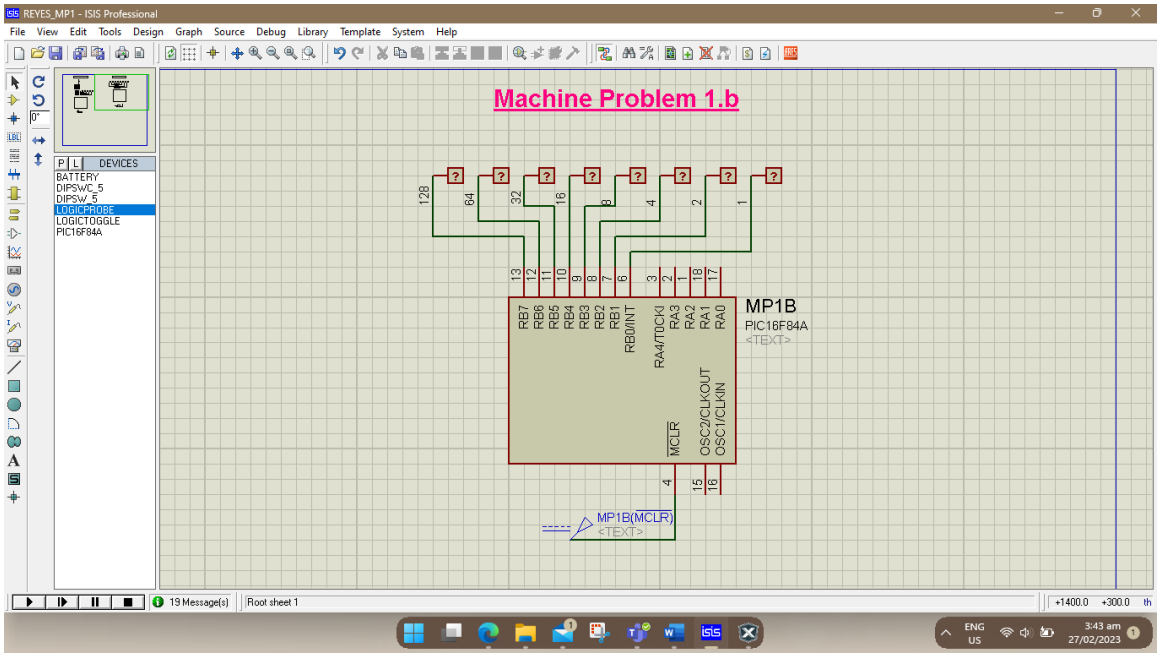


e. RA0 and RA1 are Logic 1:

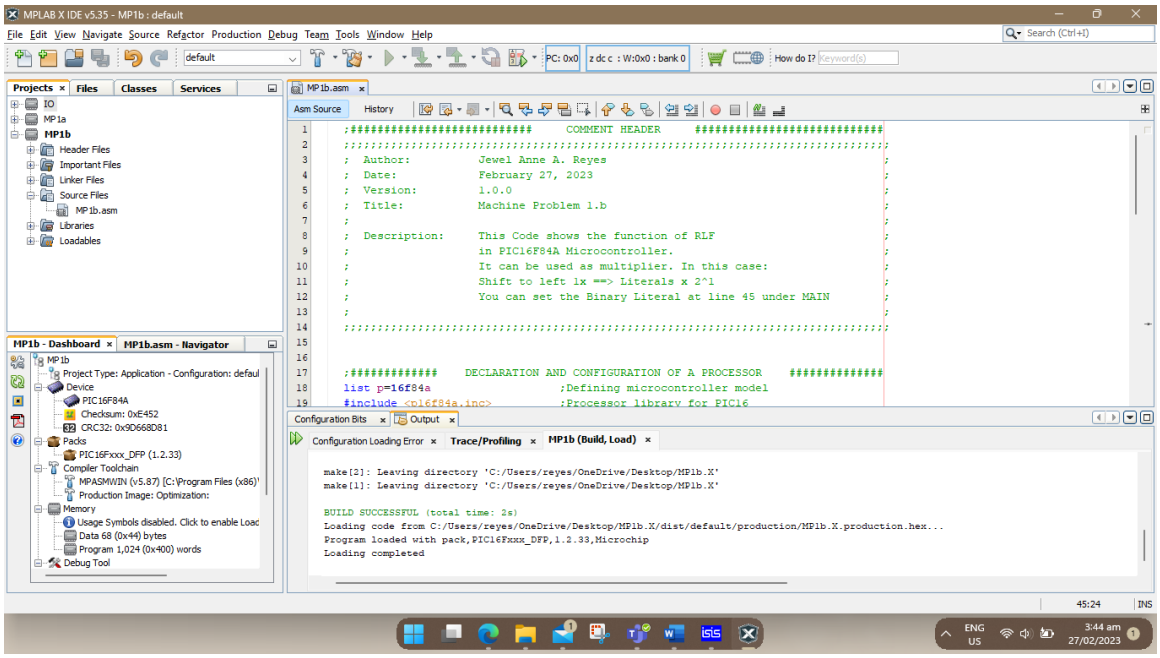


B. Using the same setup in A, create a code that will display in PORTB the product of Binary input to PORTA to decimal value 2. Use RLF instruction in performing Multiplication by 2. Give at least 5 results.

Schematic:

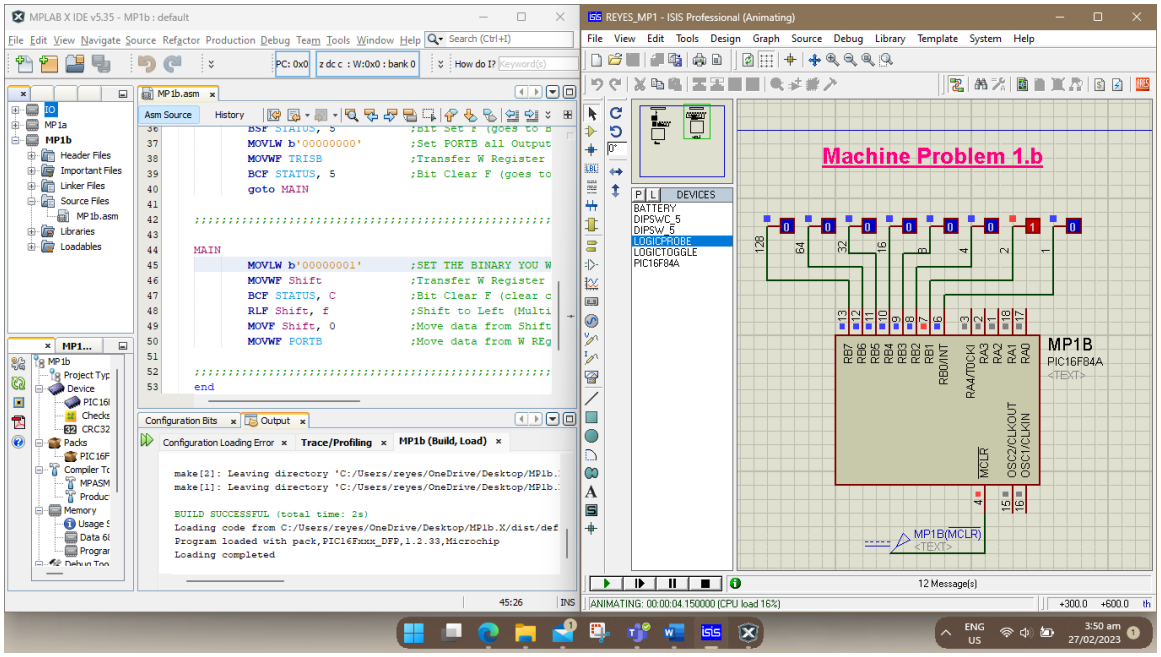


Program Build Successful:

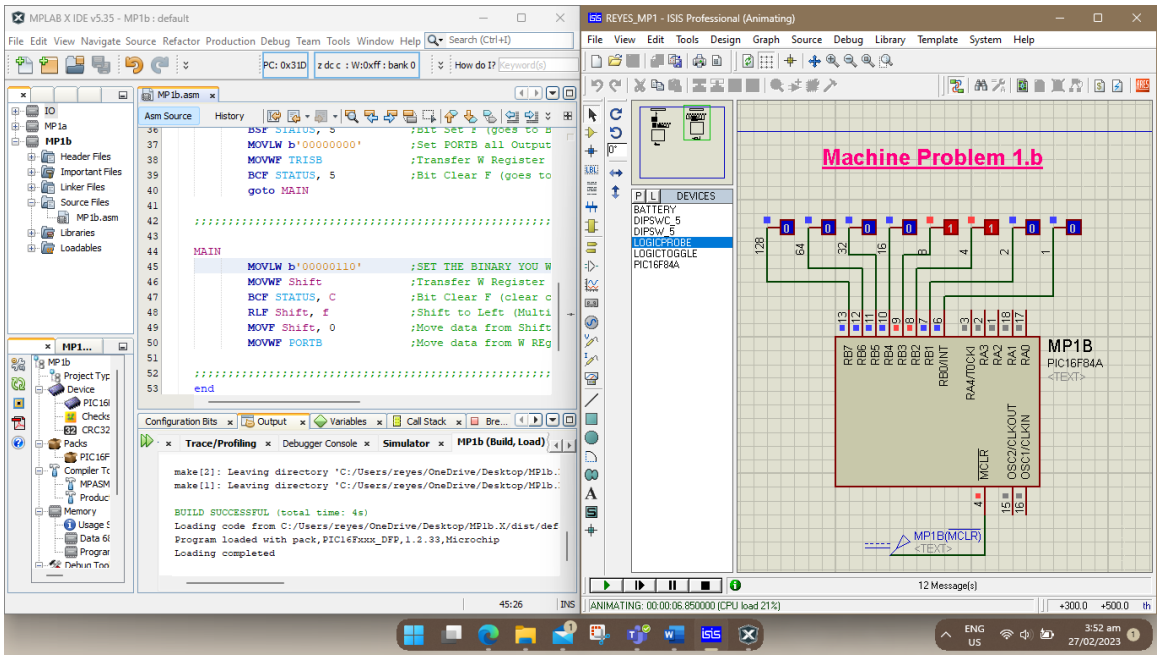


Outputs:

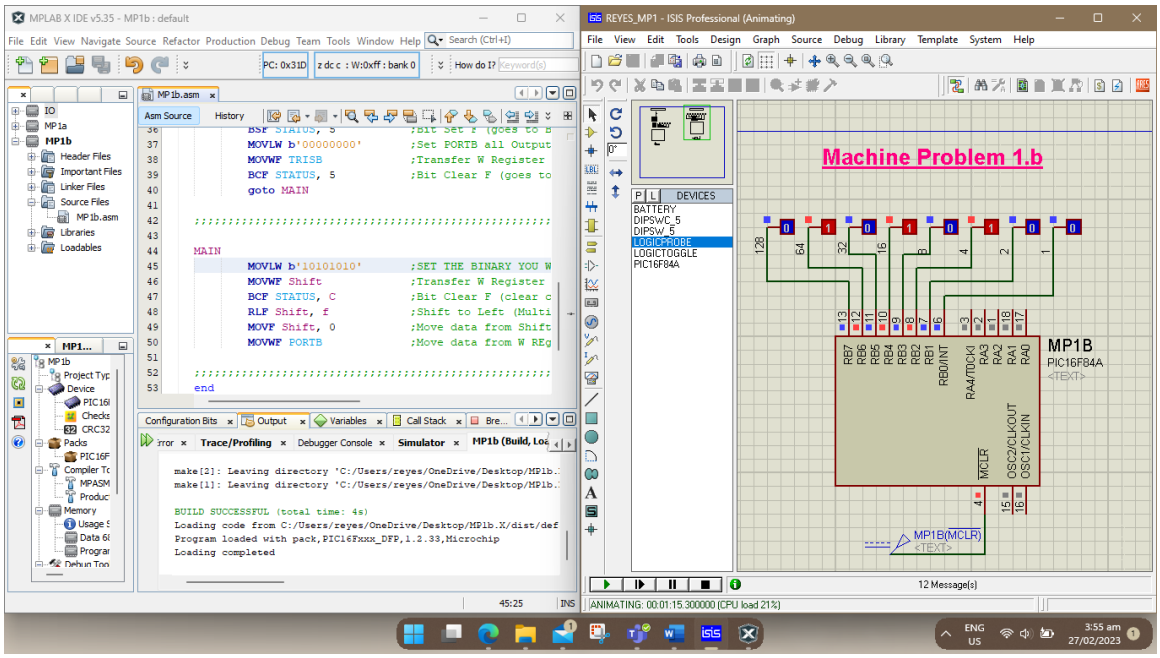
a. Binary Input = 0000 0001:



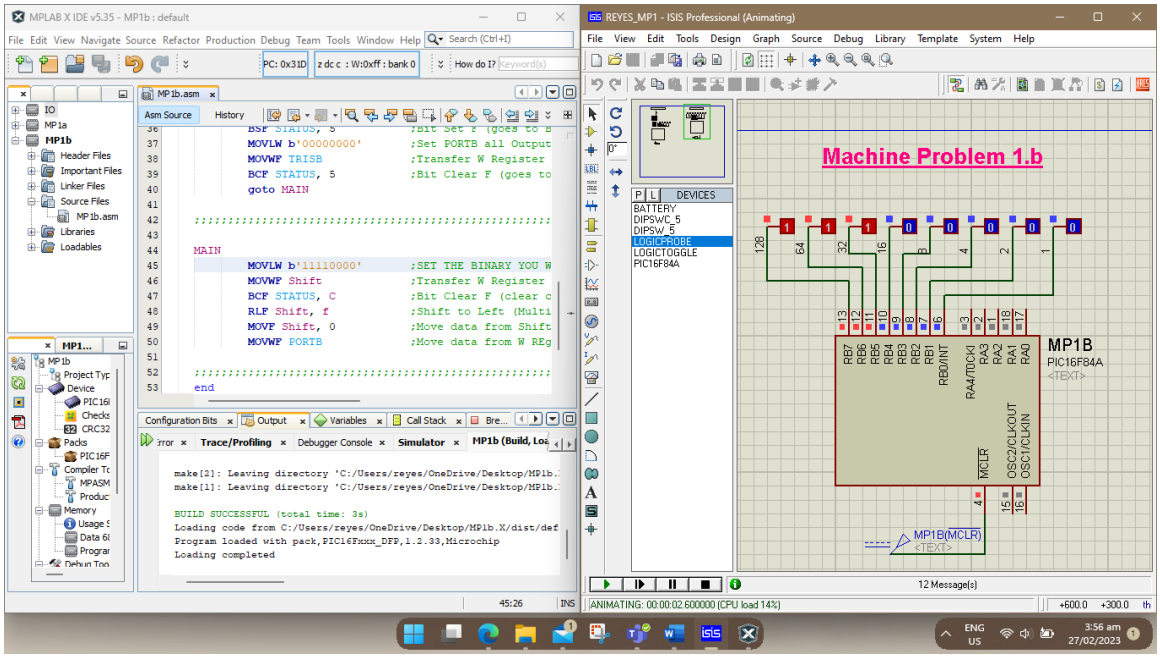
b. Binary Input = 0000 0110:



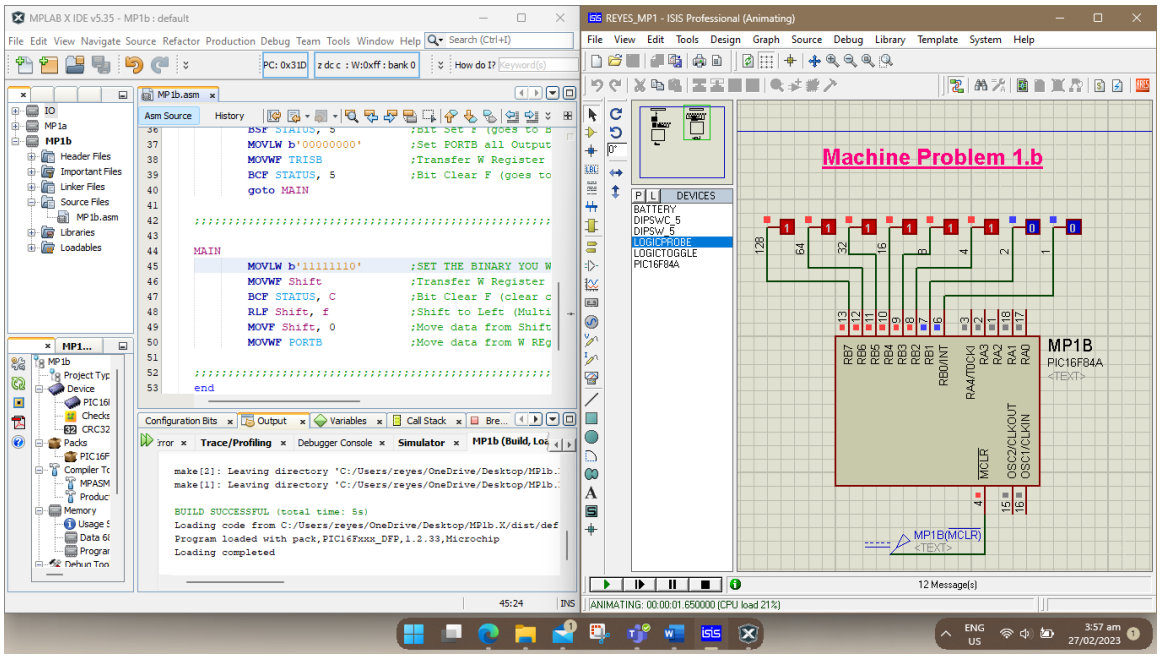
c. Binary Input = 1010 1010 (the output is D'340' but we cannot show here because it overflows):



d. Binary Input = 1111 0000 (the output is D'480' but we cannot show here because it overflows):

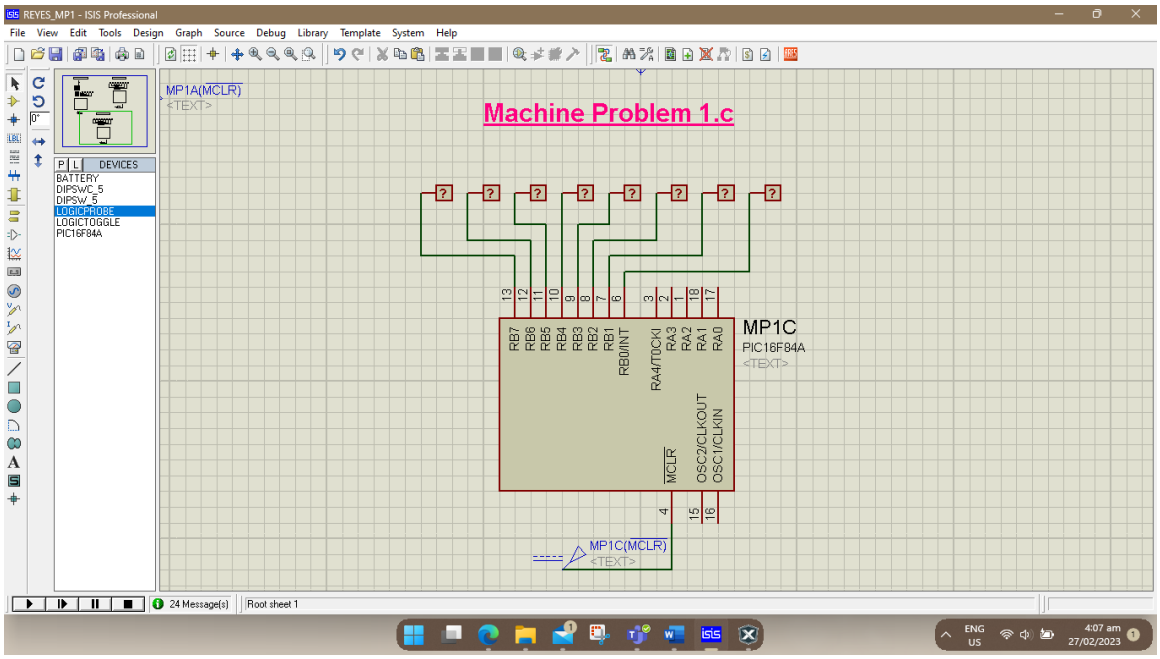


e. Binary Input = 1111 1110 (the output is D'508' but we cannot show here because it overflows):

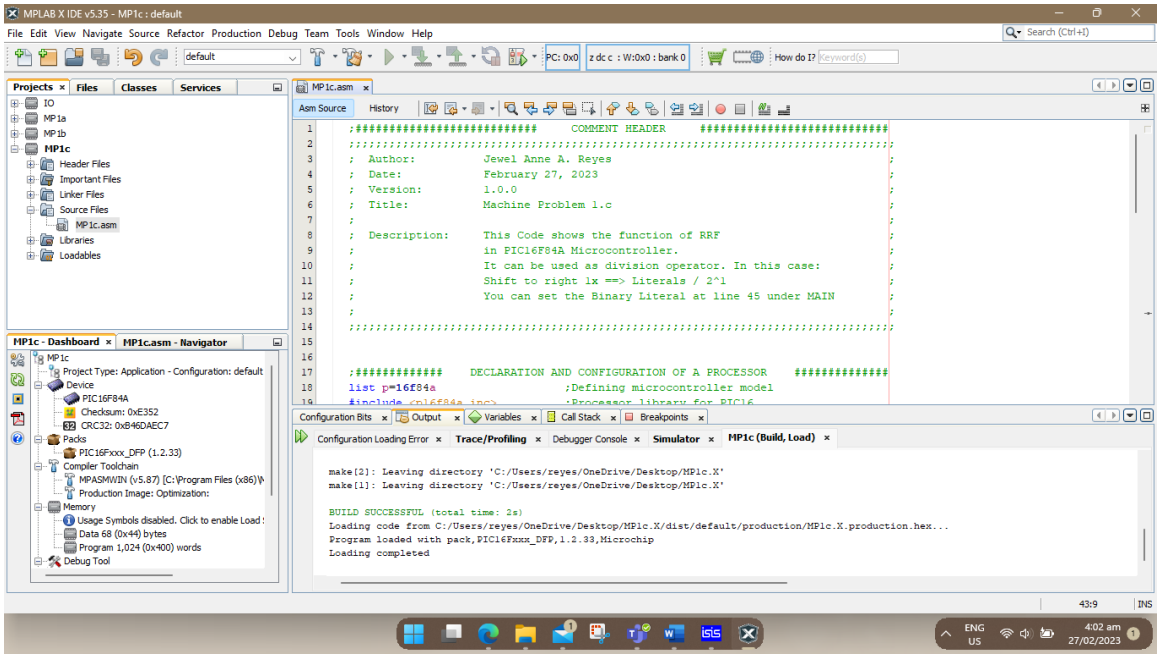


C. Using the same setup in A, create a code that will display in PORTB the quotient of Binary input to PORTA by decimal value 2. Use RRF instruction in Performing Division by 2. Give at least 5 results.

Schematic:

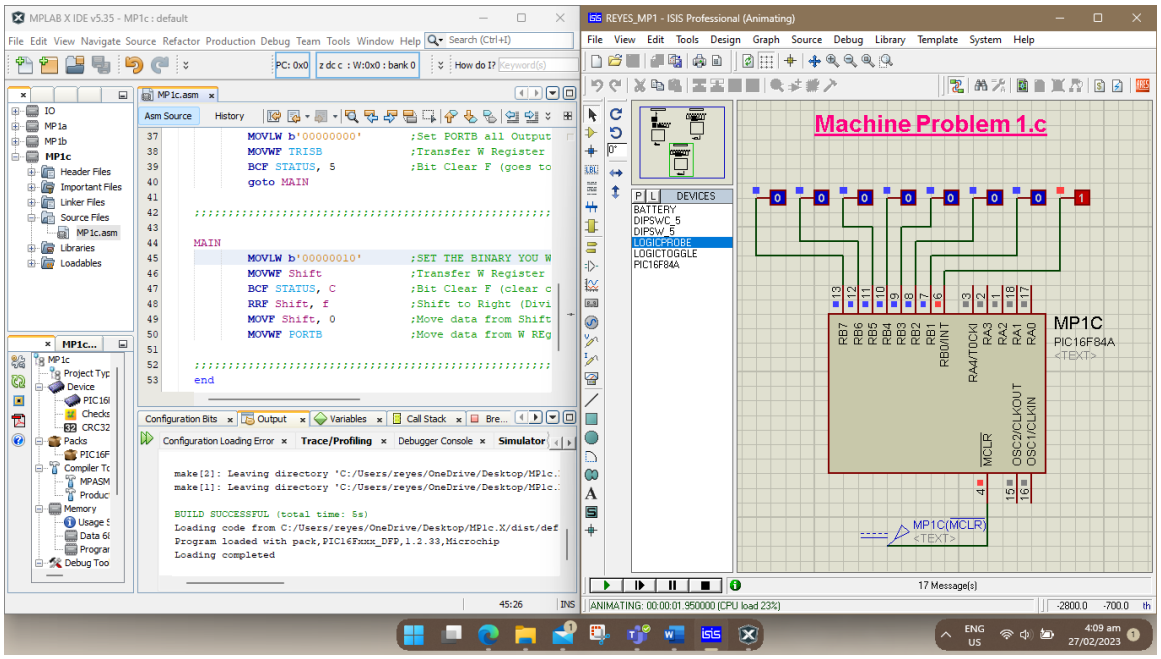


Program Build Successful:

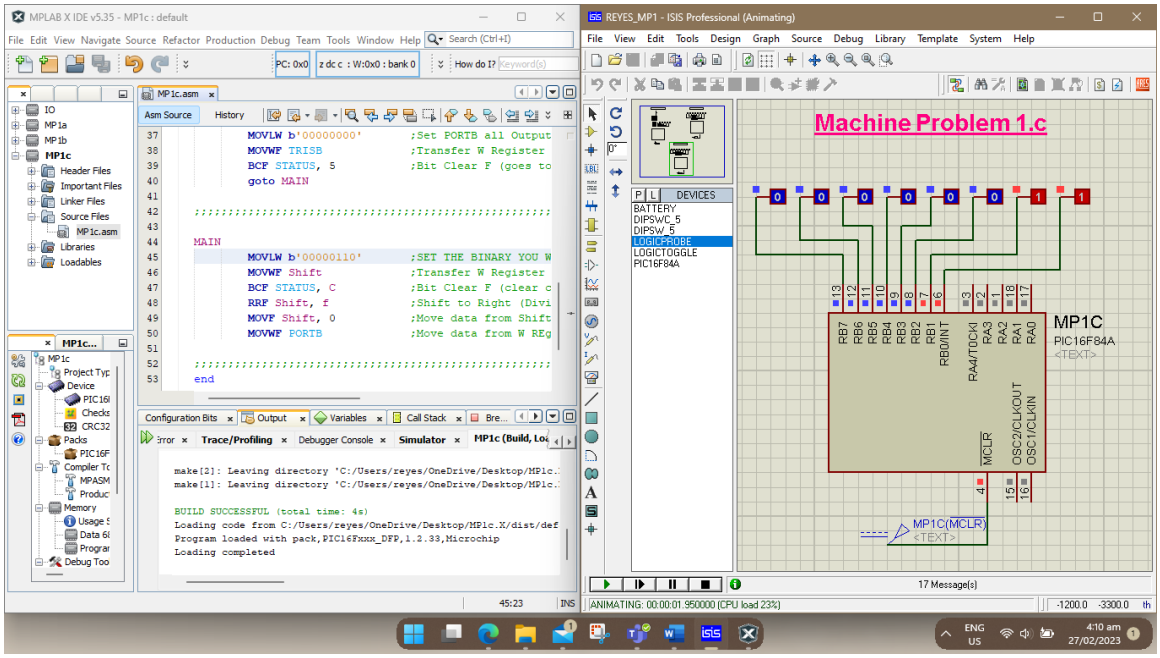


Outputs:

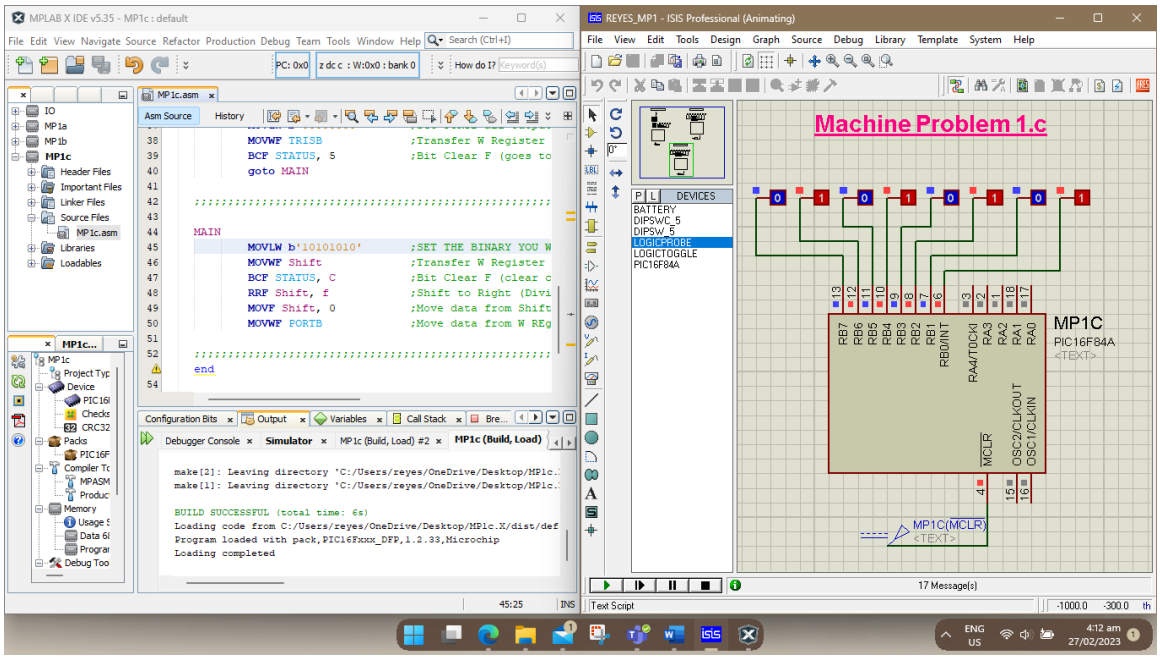
a. Binary Input = 0000 0010:



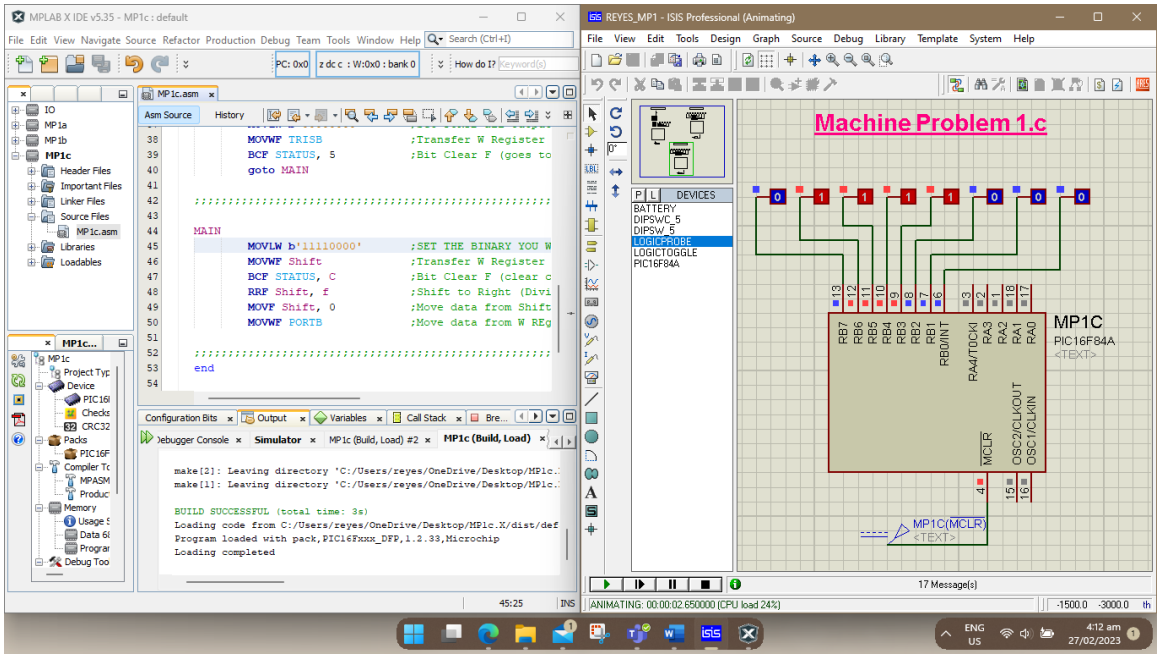
b. Binary Input = 0000 0110:



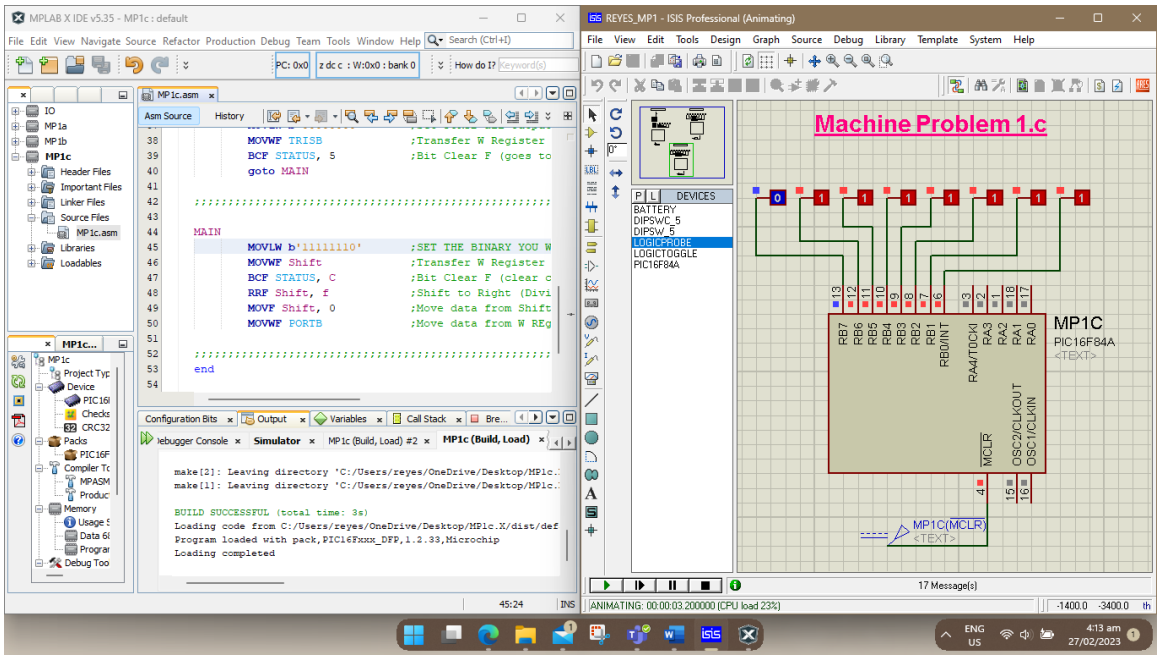
c. Binary Input = 1010 1010:



d. Binary Input = 1111 0000:

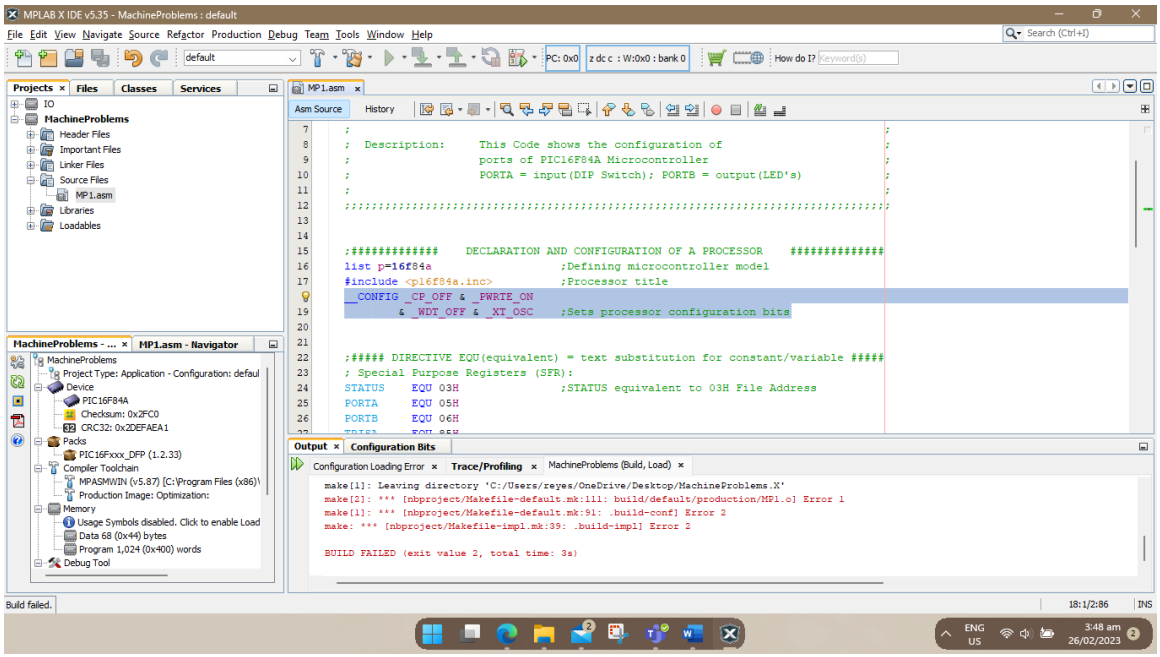


e. Binary Input = 1111 1110:



Observation:

The MPLAB program is line sensitive. I have tried to set the other commands of Assembler Directive __CONFIG into a new line just for aesthetic and organizational purposes as shown in the picture below, but unfortunately, the source code did not build properly and has an error. Turns out, it should be in the same line because the '&' is not recognized.



It is fun representing the data in different numbers (decimal, hexadecimal, binary) in assembler. Although it is personally easier for me to set it in binary as I can visualize each bit whether it is 1 (input) or 0 (output).

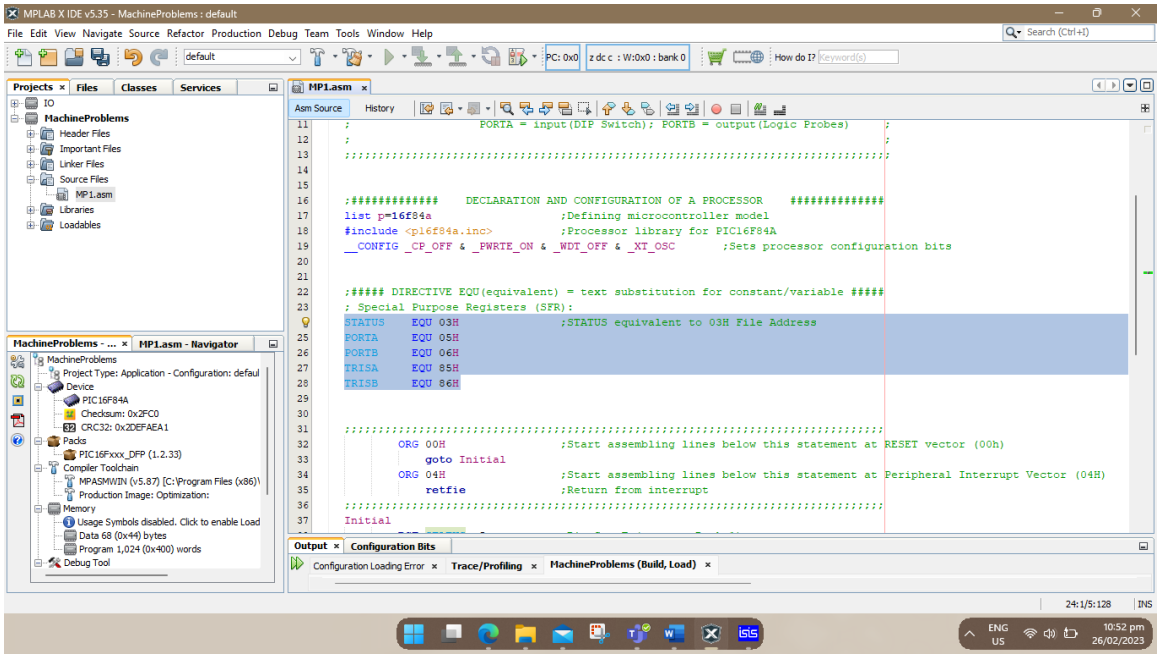
As I have checked the datasheet of the PIC16F84A, the #include can have many syntax, but the following are the preferred ones and the supported syntax are the following but does not have the # symbol:

#include include_file

#include "include_file"

#include <include_file>

And because we include the additional file for PIC16F84A, we can just delete the highlighted code in the image below. It is already set that the bits for STATUS is 03H, and the PORTA is 05H, etc. We can then use STATUS, PORTA, PORTB, etc without declaring it or using EQU. I have tried using the EQU at the first procedure (reflecting inputs of PORTA to PORTB) but I tried deleting it for the 2nd and 3rd procedure, it still turns out well.



Conclusion/ Recommendation:

Try using an LED as an output. However, we must attach a resistor to it. According to the PIC18F4550 microcontroller datasheet, the maximum current a single pin can provide is up to 25mA. Another reason is because of the peak forward current of the light-emitting device.

Also, try the procedures 2 and 3 (the one with RLF and RRF) having an input toggle in which we can manually change and see the result faster than having the binary literals in the source code. I don't know if that's possible or if there are any instruction sets that can convert inputs into specific binary. I think we need to configure the ports or something as I have observed that we can add and subtract the data from W Register from literals using ADDLW, SUBLW, etc.

I would also like to recommend using a switch in which we can choose if the instruction set is RLF or RRF since they have basically the same source code. This is to reduce the cost for the materials needed.

Source Code: (3 .asm)

(The asm files were attached with this pdf in a zip file)

```
;Machine Problem 1.a

;##############    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'11111111';Move to W Register the binary input FFH
          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)
          goto Main
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Main
          MOVF PORTA, 0                 ;Move data from PORTA to W Register
          MOVWF PORTB                   ;Move data from W REgister to PORTB
          goto Main                 ;Looping and Updating
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

end
```

```
;Machine Problem 1.b

;############# 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


; General Purpose Registers (GPR):
Shift  EQU 0CH


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
	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
   ;Setting PORTB as Output
	BSF STATUS, 5		;Bit Set F (goes to Bank 1)
	MOVLW b'00000000';Set PORTB all Outputs
	MOVWF TRISB		;Transfer W Register literals to F Register 86H
	BCF STATUS, 5		;Bit Clear F (goes to Bank 0)
	goto MAIN

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

MAIN
	MOVLW b'00000001';SET THE BINARY YOU WANT TO MULTIPLY HERE!
	MOVWF Shift		;Transfer W Register literals to F Register 0CH
	BCF STATUS, C		;Bit Clear F (clear carry bit)
	RLF Shift, f		;Shift to Left (Multiply the Literals by 2^1)
	MOVF Shift, 0		;Move data from Shift to W Register
	MOVWF PORTB		;Move data from W REgister to Output

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
end
```

```
;Machine Problem 1.c

;#############    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


; General Purpose Registers (GPR):
Shift  EQU 0CH


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        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
   ;Setting PORTB as Output
        BSF STATUS, 5                 ;Bit Set F (goes to Bank 1)
        MOVLW b'00000000';Set PORTB all Outputs
        MOVWF TRISB                   ;Transfer W Register literals to F Register 86H
        BCF STATUS, 5                 ;Bit Clear F (goes to Bank 0)
        goto MAIN

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

MAIN
        MOVLW b'00000010';SET THE BINARY YOU WANT TO DIVIDE HERE!
        MOVWF Shift                   ;Transfer W Register literals to F Register 0CH
        BCF STATUS, C                 ;Bit Clear F (clear carry bit)
        RRF Shift, f              ;Shift to Right (Divide the Literals by 2^1)
        MOVF Shift, 0                 ;Move data from Shift to W Register
        MOVWF PORTB                   ;Move data from W REgister to Output

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
end
```

Additionals:

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

The image below shows the schematics of MP1.a until MP1.c

I have changed the input of MP1.a. I used DIP switch instead of Logic Toggle.