De La Salle University – Manila

Gokongwei College of Engineering

Department of Electronics and Computer Engineering

Microprocessor and Microcontroller Systems and Design Laboratory

LBYEC3L

Laboratory Report #1

Introduction to Assembly Language

by

Dayrit, Bettina Gaille H.

Guevarra, Gia Kyla S.

Rodriguez, Mariah Venice A.

Sulit, Keane Dwight A.

LBYEC3L – EK1

I. Introduction

A microcontroller is a small, integrated circuit that comprises a multitude of components, such as a microprocessor, timers, counters, I/O ports, RAM, ROM, and other components [1]. These components work together to execute a pre-set series of operations. They are designed to be low-power, cost-effective, and compact, making them a popular choice for small and integrated systems. This type of integrated circuit is specifically intended for use in embedded systems, where the main objective is to control a system rather than perform computational tasks. The combination of its various components allows it to perform a range of tasks, making it a versatile solution for various applications.

A PIC16F84A is an example of a microcontroller, it is an 8-bit RISC (Reduced Instruction Set Computing) microcontroller that has a compact design, making it a common choice for small, integrated systems. It is recognized for its simplicity, energy efficiency and affordability. This microcontroller is equipped with a 1Kbyte flash program memory that can be programmed using either assembly language or high-level languages such as C and BASIC, as well as 68 bytes of RAM, 64 bytes of long-term EEPROM storage and a single peripheral [2]. One of its distinguishing features is its low power consumption, which allows it to operate on a low voltage while using minimal power, making it ideal for energy-sensitive applications.

The objectives of the experiment is to gain understanding and knowledge on simple microcontroller instructions, particularly for the PIC16F84A microcontroller. The following sub-objectives are as follows:

1. To be able to create a logical and working program using assembly language;
2. To be able to program a hardware circuit using assembly language; and
3. To be able to understand how machine language and assembly language works.

II. Methodology

A. Materials

The materials used for the conduct of the experiment are as follows:

Electronic Component	Quantity
PIC16F84A	1
LED-RED	8
Common Cathode 7-Segment Display	1
10k Resistor	1
33 uF Capacitor	2
1 uF Capacitor	2
9 VDC Supply	1
7805 IC	1
Crystal Oscillator	1

Table 1. List of Electronic Components used for Simulation

Moreover, the experimenters utilized Proteus 8 Professional for circuit simulation and MPLAB IDE for code development.

B. Procedure

Exercise 1:

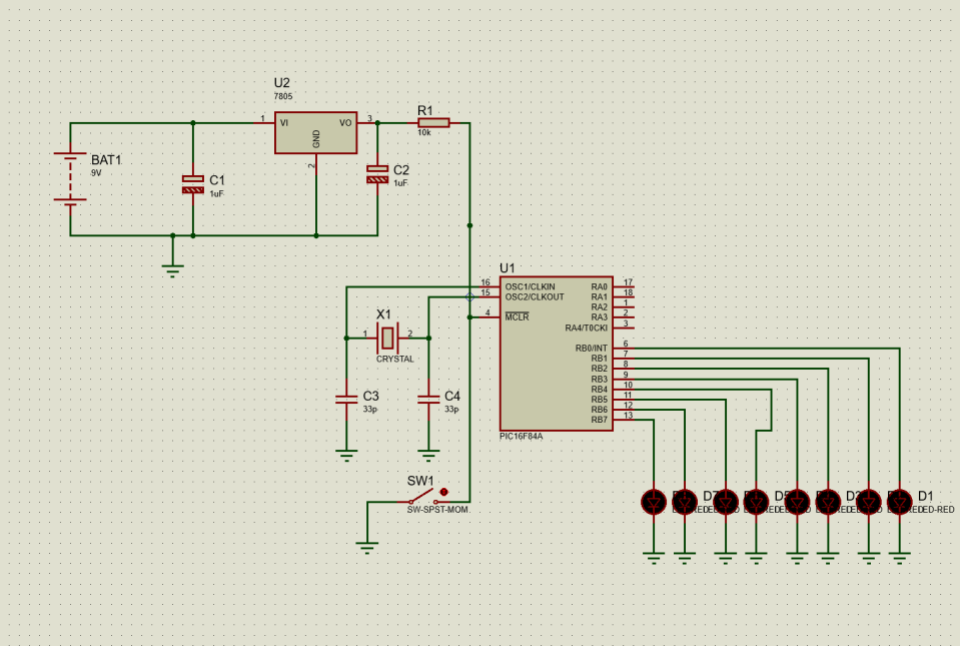


Figure 1. Circuit Configuration for Exercises 1A and 1B

For this laboratory experiment, Exercise 1 has two parts; Exercise 1A and Exercise 1B. For Exercise 1A, the main objective was to have all the even numbered LEDs turned off, while the rest of the LEDs, which are the odd numbers, turned on. Using the software Proteus, Figure 1 shows the circuit built for both Exercise 1A and 1B. This circuit contains 8 LEDs connected to their respective pins in the microcontroller, numbered 7 through 0, starting from the leftmost going to the right—these are the key indicators for this part of the exercise.

Next, using the software MPLAB, a code is then created to satisfy the requirements set for each of the exercise. For Exercise 1A, having all even numbered LEDs off basically pertains to having alternate LEDs turned on—considering that even numbered LEDs are to be turned off, the first LED to the left should be turned on. While, for Exercise 1B, the middle four of the LEDs: 5^{th}, 4^{th}, 3^{rd}, and 2^{nd}, should be turned off. For both exercises, a similar code should be produced, however, the byte configuration should differ since the pins connected to the LEDs which corresponds to what the group intends to light up, differ. The code created based on this will allow the microcontroller to execute these actions. Upon finishing the code, the students must wait for the "Build Succeeded" before finally exporting the code to Proteus.

Lastly, the code is then imported to Proteus, given that MPLAB is a supported programming language. This will then be the code that the microcontroller will process. Expected results should be the correct LEDs lighting up per exercise.

Exercise 2:

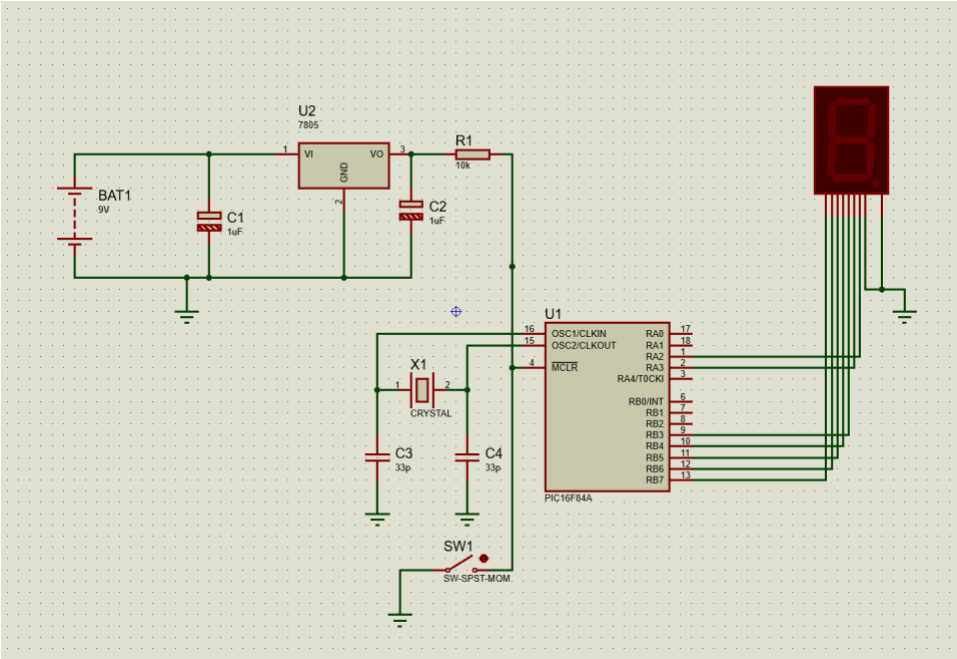


Figure 2. Circuit Construction for Exercise 2

The objective of Exercise 2 is to light up the 7-segment display to display the numbers 0-9 and the letters A-F. The following circuit design that was used is shown in Figure 2. As can be observed, the same circuit from exercise 1 was used for this exercise, but the LEDs were replaced with a 7-segment display. The circuit construction was accomplished using the Proteus software.

Afterwards, in order for the circuit in Proteus to function, the code for the microcontroller must be written in a supported programming language. MPLAB software was used to compile the code and program it into the microcontroller. The compilation process converts human-readable code into machine-readable machine code that the microcontroller can execute.

Next, is to write a program that operates a 7-segment display to show decimal numbers from 0-9 and hexadecimal values from A-F. With a strong understanding of logic circuits, the experimenters figured out which segments need a "HIGH" input to be lit. For instance, to display the number 0, all segments of the display must be lit, and the code must reflect that. To show the number 1, only segments B and C need to be lit and coded accordingly. This approach is repeated for all remaining numbers and letters that need to be displayed. After writing the code, the next step is to compile it. The compilation process checks for any errors in the code and if there are no errors, the program will be successfully built. Finally, the compiled code is then programmed into the microcontroller and the circuit is then connected to a power source and the 7-segment display is tested to ensure that it is displaying the desired digits.

III. Results and Discussion

This section shows the results of the experiment after following the necessary procedures:

Exercise 1:

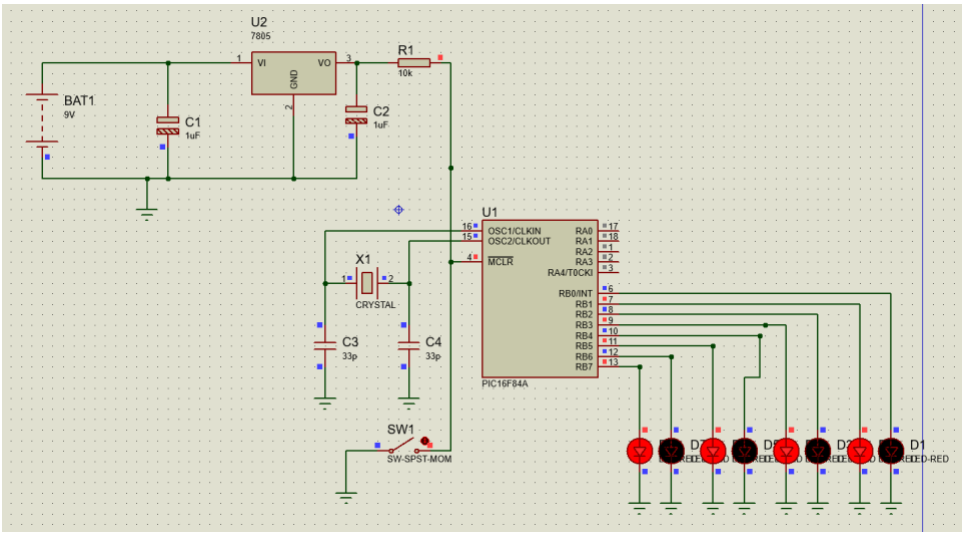


Figure 3. Exercise 1A

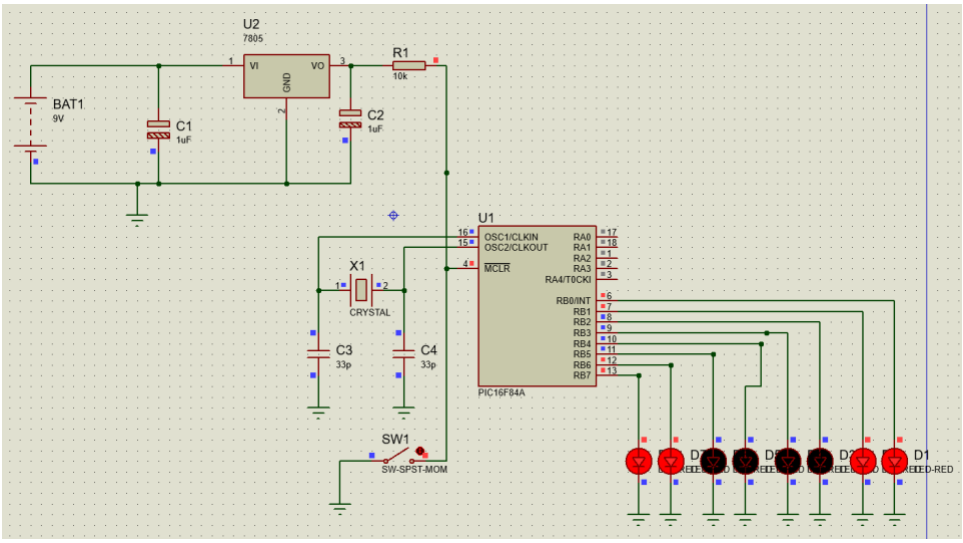


Figure 4. Exercise 1B

As stated in the previous sections, there are two types of activity to be made for this experiment. For Exercise 1, it is divided into two sub-exercises. The first one can be seen in Figure 3, wherein the assigned task for the group is to light up all the odd-numbered LEDs while the even-numbered ones are turned off. This leads to a result where the LEDs are lit up alternately. For the next part of the first exercise, the task assigned is to light up all the LEDs except the four in the middle. As seen in Figure 4, the instruction for the exercise was followed and the results were the same as expected. With the use of assembly language, the following conditions were coded and drove the PIC16F84a microcontroller to meet the necessary requirements for Exercise 1.

Exercise 2:

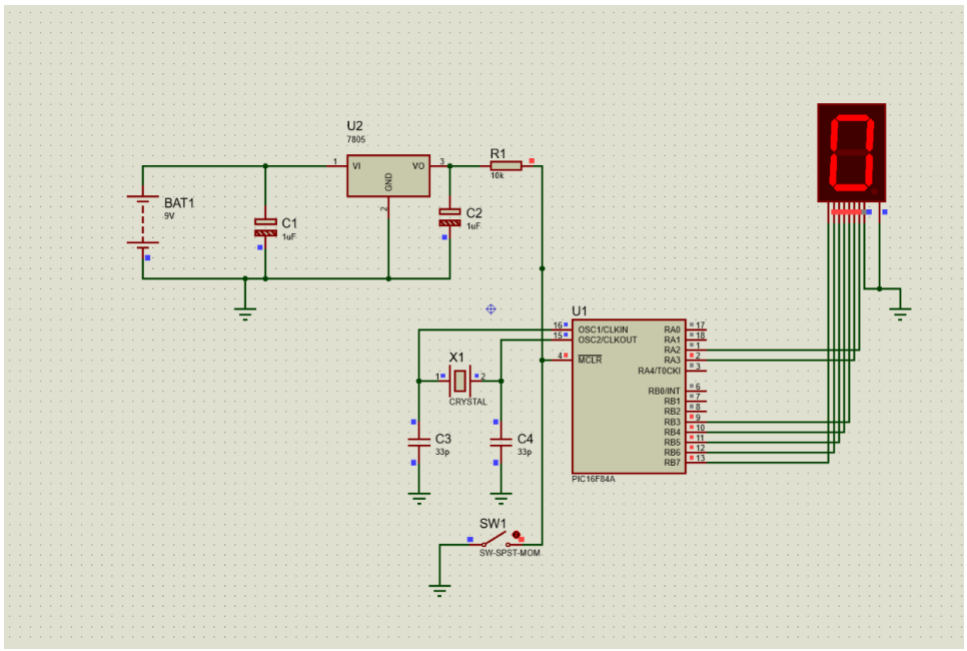


Figure 5. Output ‘0’

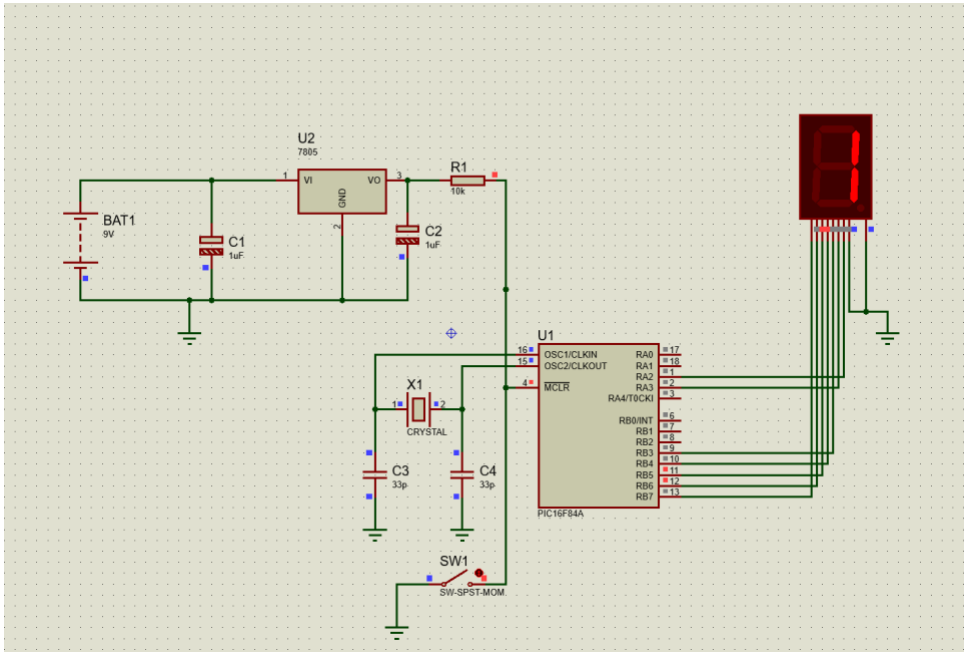


Figure 6. Output ‘1’

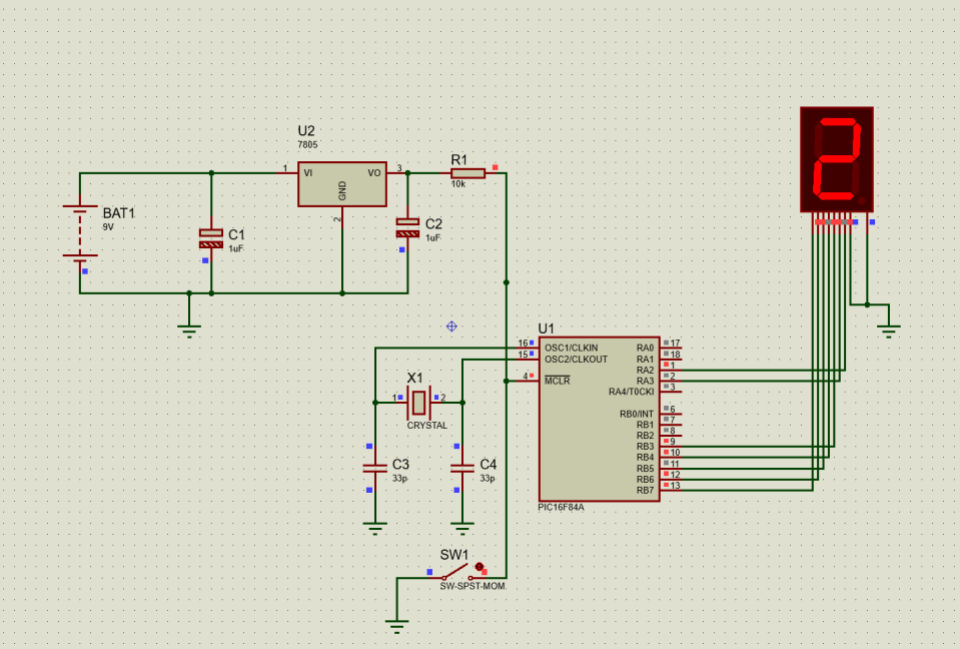


Figure 7. Output ‘2’

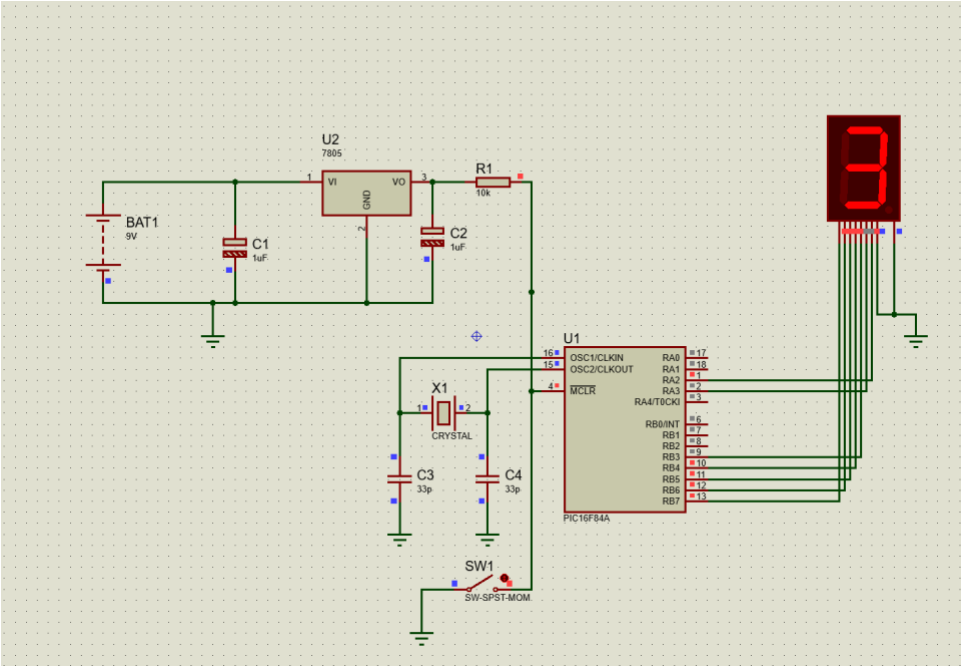


Figure 8. Output ‘3’

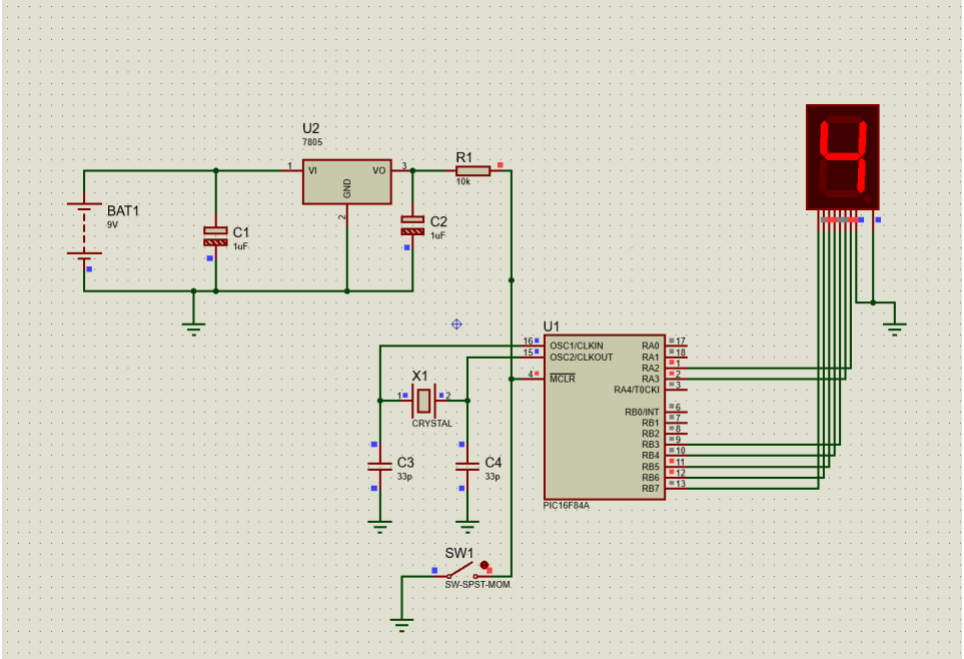


Figure 9. Output ‘4’

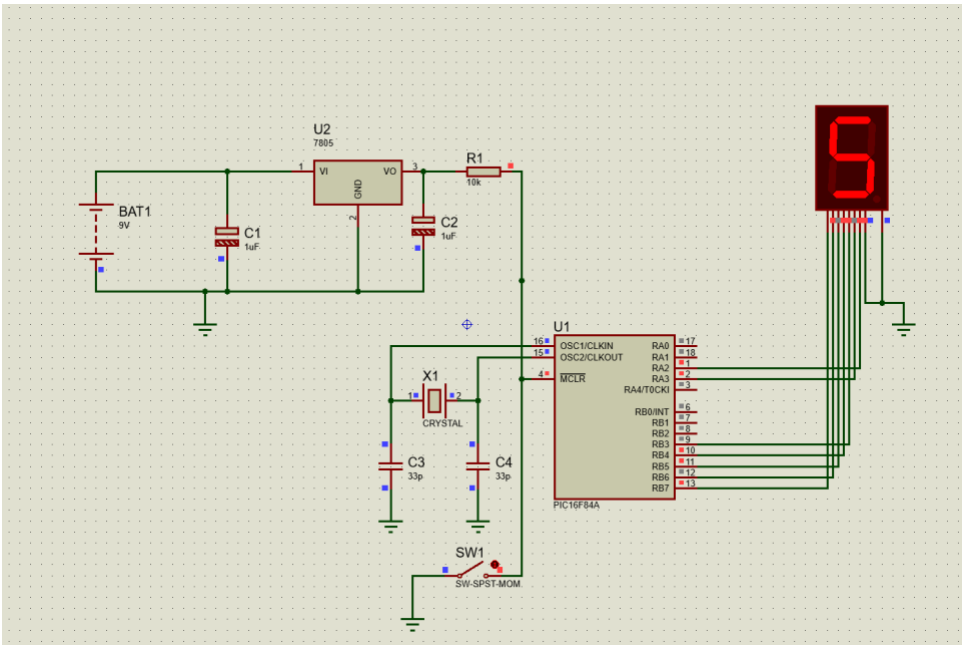


Figure 10. Output ‘5’

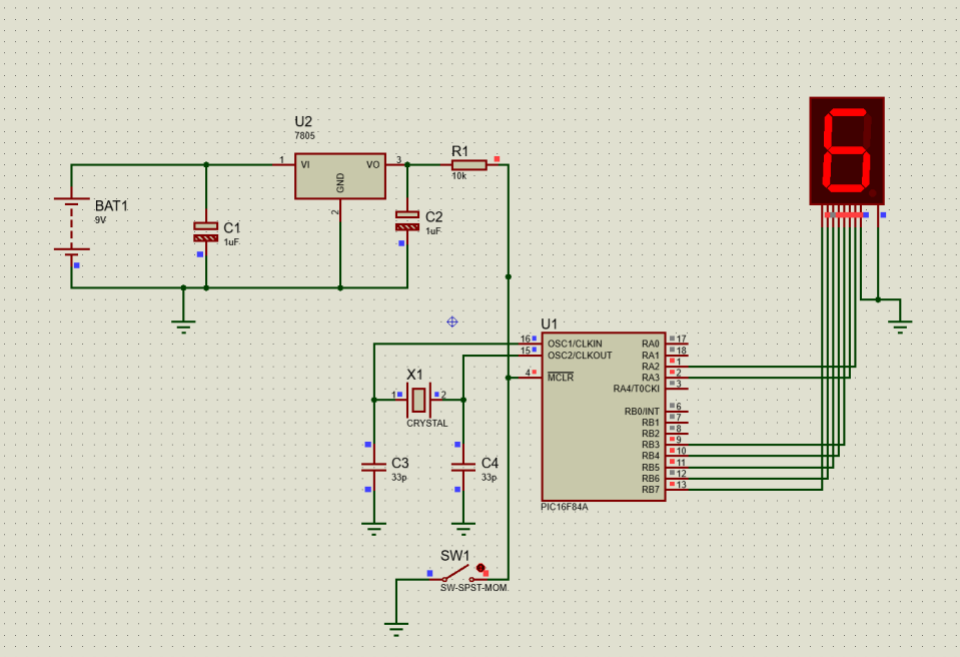


Figure 11. Output ‘6’

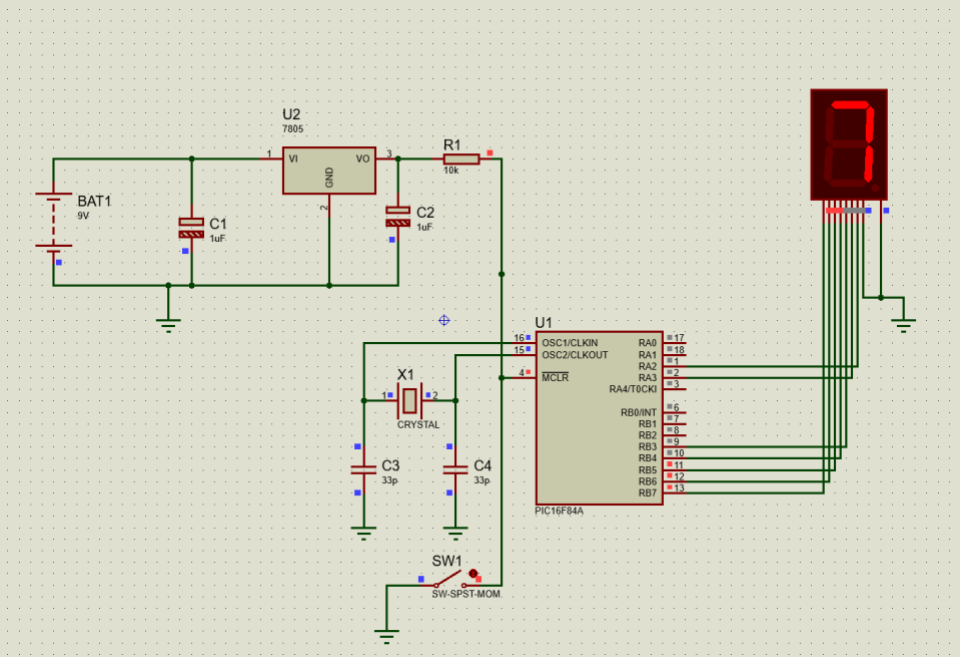


Figure 12. Output ‘7’

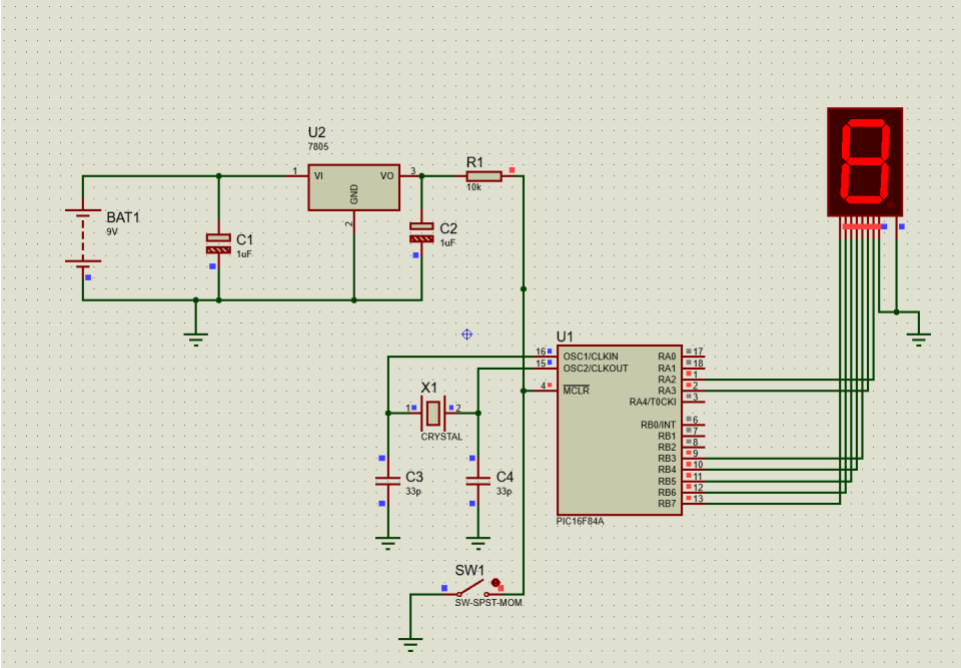


Figure 13. Output ‘8’

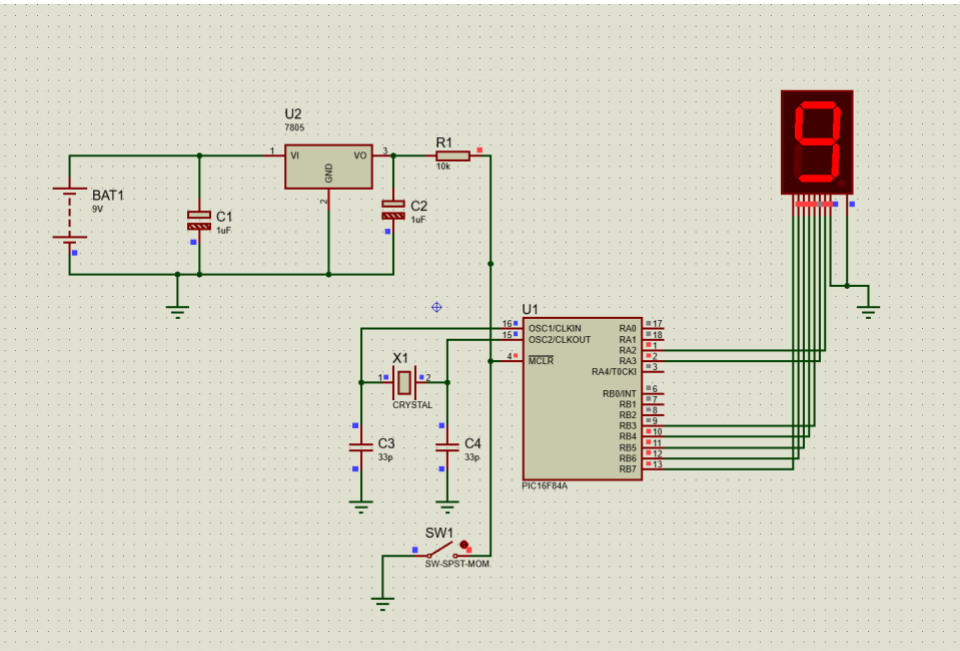


Figure 14. Output ‘9’

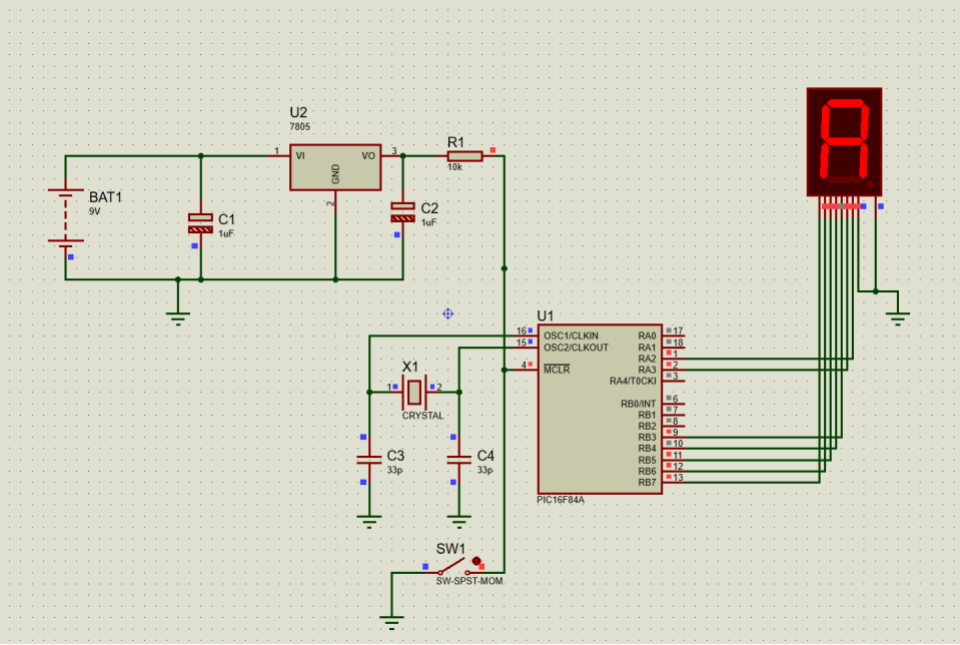


Figure 15. Output ‘A’

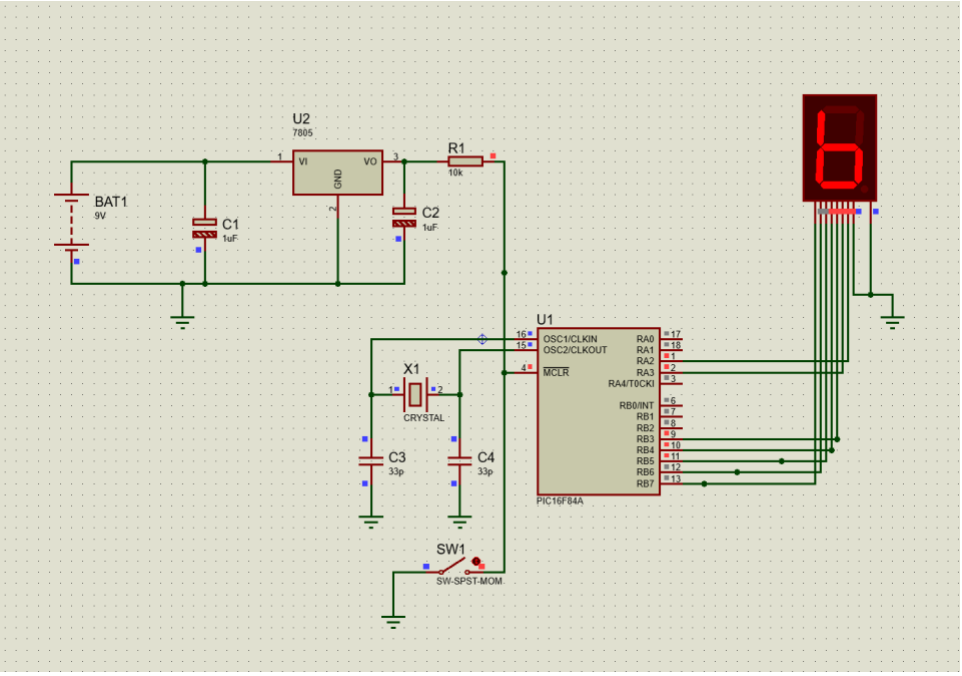


Figure 16. Output ‘B’

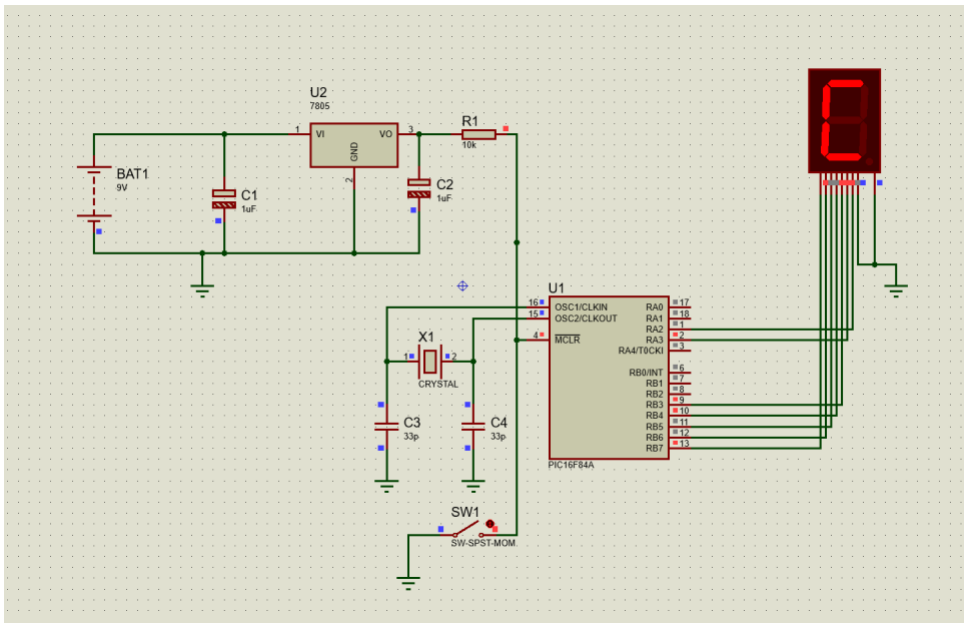


Figure 17. Output ‘C’

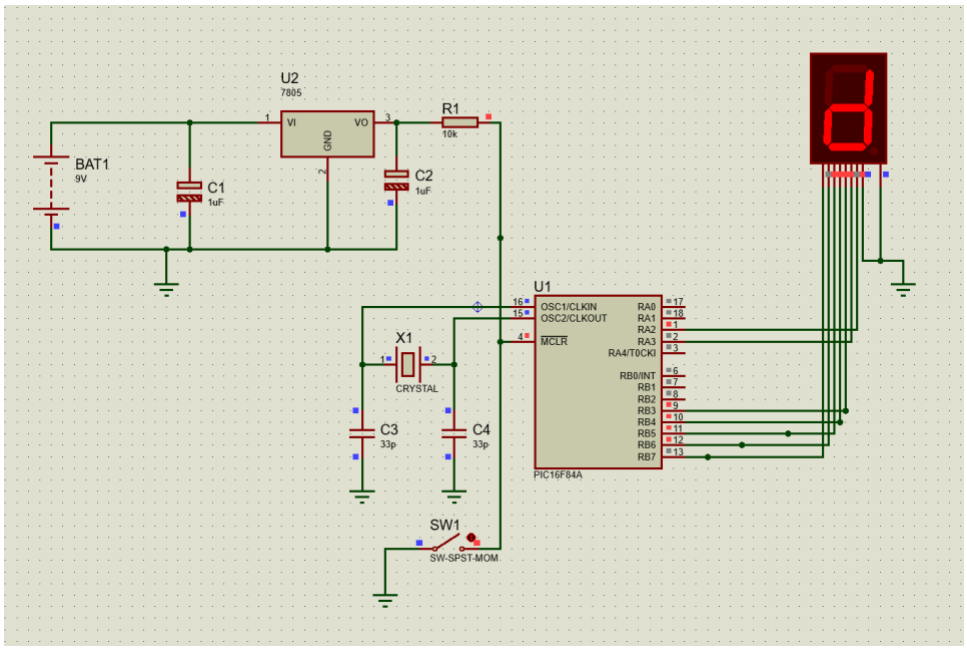


Figure 18. Output ‘D’

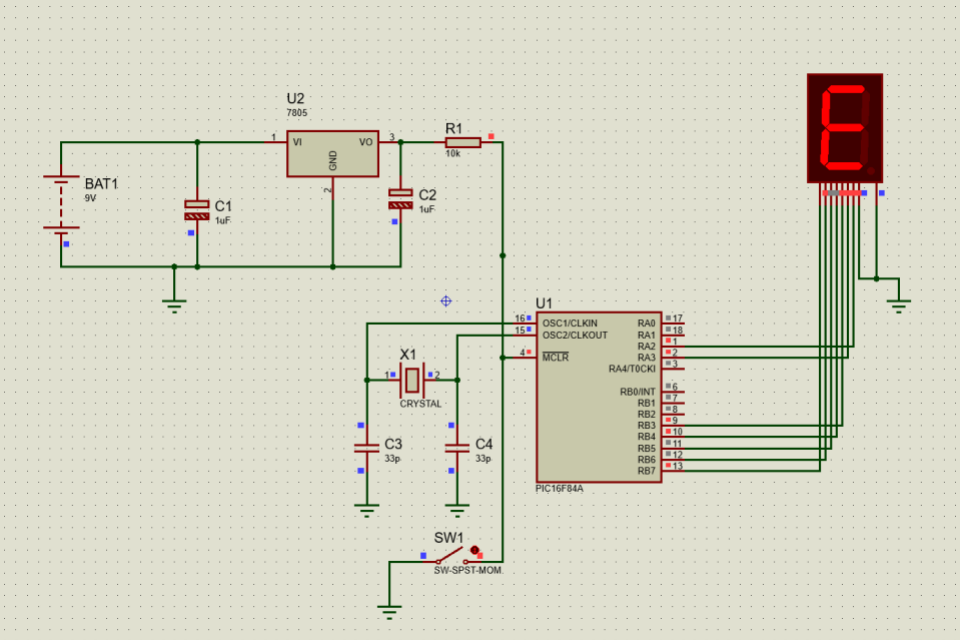


Figure 19. Output ‘E’

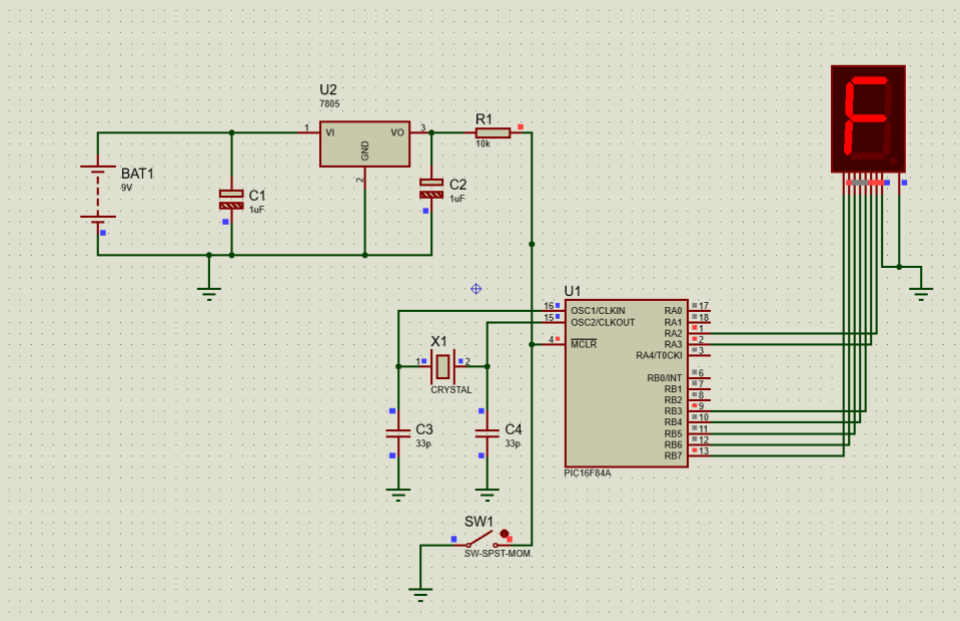


Figure 20. Output ‘F’

For Exercise 2, it deals with configuring 8 of the 13 input/output pins, from PORTA and PORTB, of the PIC16F84A to drive a 7-segment display that shows the decimal numbers 0-9 and the hexadecimal values A-F. It is important to note in this part that in order to display ‘B’ and ‘D’, lowercase letters must be used in order to display the letter itself. This is due to capital letter ‘B’ being synonymous to number 8 and ‘D’ looks like ‘0’ as well. Hence, displaying a small letter ‘b’ and ‘d’. A good background on logic circuits allowed the experimenters to know what segments needed to be provided with a HIGH logic input for it to light up.

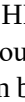
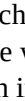
Digit	7-Segments							abcdefg	gfedcba	Display
	a	b	c	d	e	f	g			
A	1	1	1	0	1	1	1	0x77	0x77	
a	1	1	1	1	1	0	1	0x7D	0x5F	
b	0	0	1	1	1	1	1	0x1F	0x7C	
C	1	0	0	1	1	1	0	0x4E	0x39	
c	0	0	0	1	1	0	1	0x0D	0x58	
d	0	1	1	1	1	0	1	0x3D	0x5E	
E	1	0	0	1	1	1	1	0x4F	0x79	
F	1	0	0	0	1	1	1	0x47	0x71	

Figure 21. HEX Table for 7-Segment Display of Letters.

Figure 21 above shows the HEX Table displaying letters to a 7-segment display. It can be observed that ‘b’ and ‘d’ are in lowercase because its capital form can be confused with number ‘8’ and ‘0’.

IV. Analysis and Conclusion

Before doing the experiment proper, the group first understood and learned how the software, Proteus, and programming software, MPLAB, works. Since it is the first time that the experimenters will work with these applications, it is essential that the understanding or the basic knowledge of it is covered, hence achieving the 3rd objective, being able to understand how machine language and assembly language works. With this, the group was able to perform the set exercises for the laboratory experiment, which includes Exercise 1A, 1B, and 2.

As for Exercise 1, the group was able to create both the working circuit in Proteus, as well as the working code created from MPLAB, effectively achieving both objectives 1 and 2. A working circuit from Proteus, as seen in the section Procedures and the code written in MPLAB that was imported to the microcontroller led to the success of Exercises 1A and 1B. Similar in code, the findings of the group are as follows; when working with multiple pins, one should consider the byte, instead of the individual registers, therefore, there is a change in the code from simply *BSF*, *BCF*, to the addition of *MOVLW* and *MOVWF*. The difference between the two instructions is simple; *MOVLW* basically loads the binary value into the W register, while *MOVWF* basically copies the content which indicates which bits in the byte has the logic HIGH and logic LOW—proper addressing are as follows. A visualization of the pins helps— in the group’s case, the initialization stage follows that the complement of the byte is the content, which can be easier to input following that a visualization of the registers is present. With this, the group has established the habit of being careful with the binary input, especially since 1A and 1B differed in this area.

Although the students needed time to process the concept addressing and the ports, patterns are still recognized which helped the students with the creation of the code. As discussed earlier, this is the first experience with the software used, therefore mistakes and confusion are bound to happen. The process of writing the code in this part of the experiment went smoothly, simply because the group took their time with understanding the basic concept behind it. However, confusion in this part can come from

the difference between *MOVLW*, *MOVWF*, *BSF*, and *BCF*. Admittedly, there is an apparent learning curve to the process of this experimental procedure. In this part of the exercise, the same code was used, only differing in the byte configuration.

For Exercise 2, similar to the first exercise, the experimenters were able to program and configure the 2 ports (PORTA and PORTB) of the PIC16F84A microcontroller. From the assembly codes (refer to Appendices), a pattern can be recognized in the process of lighting up specific segments. The most significant skill that must be learned for this exercise is **how to instruct the PIC microntroller to do the desired output based on the available instruction set**.

```
; Segment A
1    BSF 03h, 5
2    BCF 86h, 7
3    BCF 03h, 5
4    BSF 06h, 7
```

Figure 22. General Assembly Code for Lighting up a Segment in PORTB

```
; Segment F
1    BSF 03h, 5
2    BCF 85h, 3
3    BCF 03h, 5
4    BSF 05h, 3
```

Figure 23. General Assembly Code for Lighting up a Segment in PORTA

As seen in Figure 22., the first instruction is to first operate on Memory Bank 1 to access the TRISB register. As Jasio, et al. [3] stated, PIC microcontroller ports are bidirectional. As a result, each port pin may be utilized as either an input or an output pin. This explains the second line of the general code. The register is known as the TRIS register, and each port has a TRIS register that is named after the port name (85h for PORTA, and 86h for PORTB). TRISB, in the case above, is the PORTB direction control register. Similarly, TRISA is the PORTA direction control register, which is required for the lighting of Segment F and G (assigned by the instructor) as seen in Figure 23. The port direction control register specifies whether the port pins are inputs or outputs (BSF or "*bit set f*" is **1** which indicates an **input**, and BCF or "*bit clear f*" is **0** which indicates an **output**). Afterwhich, the third line means that the microcontroller should now operate back on Memory Bank 0 to access the PORTB register. Finally, the last line of code sets the output value, whether logic HIGH (BSF) or LOW (BCF), of the pin that corresponds to the D_n register of the PORT. Summing it all up, the following logic flow can be followed:

STEP 1. *Operate on Memory Bank 1 to access the TRIS register.*
STEP 2. *Make the desired pin as output.*
STEP 3. *Operate on Memory Bank 0 to access the PORT register.*
STEP 4. *Set the output of the desired pin to HIGH for it to light*

Figure 24. Logical Steps of Programming a Segment to Turn On

The process of omitting and adding back the specific 4 lines of instruction for the lighting of a segment was done and aligned based on the function table (as seen in Figure 21) of a 7-segment display.

A notable hurdle experienced by the researchers was the confusion on what the display should be for the hexadecimal value, '**b**' and '**d**'. At first, the build was similar to that of the decimal number '**8**' and '**0**', respectively since these visually corresponds to the capitalized shapes of the said output (**B** and **D**) as seen in Figure 24. below.

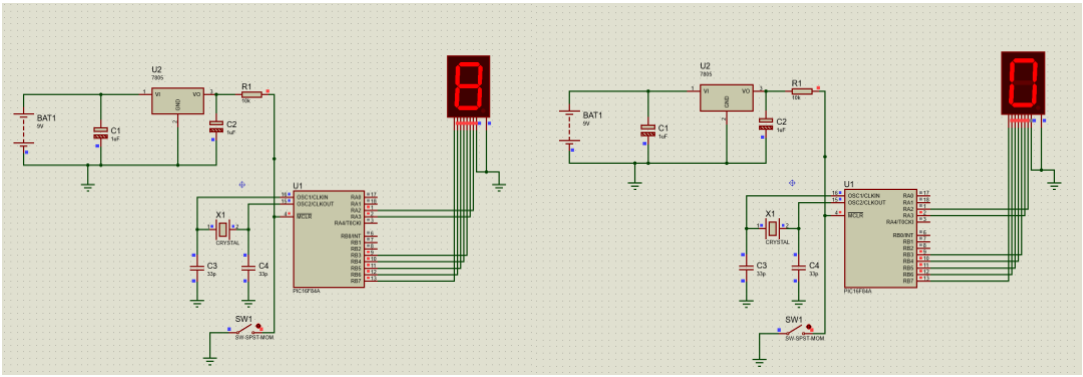


Figure 25. Initial Output for the Letters 'B' and 'D'

Overall, from the working embedded programs, it can be deduced that programming a microcontroller is similar to that of higher-level languages such as C/C++ or Python. However, for a PIC16F84A microcontroller, assembly language (.asm) is used. These readable instructions are then comprehended by the electronic component into machine language (.hex). All throughout the exercises, the experimenters assembled instructions and built firmwares that would essentially give commands to the microcontrollers. Machine language are not intelligible to humankind considering that the information in these are purely based on number systems. A comparison between the two languages can be seen in Figure 26. below (the first few lines of the .hex file built in Exercise 1A are shown due to its length). For students trying these exercises for the first time, it is recommendable that the study of how the code works, as well as the familiarization of the datasheet containing everything an individual needs to know upon coding is necessary. This allows the person to easily manipulate the code into what the exercise calls for.

Figure 26. Assembly Language vs. Machine Language

Computers, from the word itself, are only able to compute. Hence, programmers are needed to give directives to such devices, and succeedingly, these are then translated to executable sequences of instructions.

For assembly language coding, the process is much more challenging considering that the PIC series are not the only microcontrollers that exist to date. The utilization of assembly language to program functional digital systems entail a huge dedication in reading its corresponding datasheets. These materials provided by various manufacturers include information that can be convoluting to someone at first, but understanding the workings of the whole chip by analyzing block diagrams, instruction sets, and operands can hone one's skill not only in programming but also in digital electronics, as a whole.

All in all, the objectives of the experiment were met as the experimenters were able to explore the workings of a microcontroller. Moreover, the experimenters were able to create logical and fully functional programs using assembly language, program a hardware circuit implemented via simulation, and understand how machine and assembly language drive digital circuits to work.

V. References

[1] B. Lutkevich, “What is a Microcontroller and How Does it Work?,” *IoT Agenda*, Nov. 2019. https://www.techtarget.com/iotagenda/definition/microcontroller

[2] TronicsBench, “A PIC16F84 Introduction.,” *Best Microcontroller Projects*. https://www.best-microcontroller-projects.com/pic16f84.html (accessed Feb. 01, 2023).

[3] Lucio Di Jasio *et al.*, *PIC Microcontrollers: Know It All*. Newnes, 2007.

VI. Appendices

1. Exercise 1a. 8 LED Configuration A Assembly Code

```
;=====================================;
; Author: Charmbix                    ;
; Version: 2.0                        ;
; Course: LBYEC3F - EK1               ;
; Title: Experiment 1: Exercise 1     ;
;=====================================;

MOVLW B'01010101'
BSF 03h, 5
MOVWF 86h

MOVLW B'10101010'
BCF 03h, 5
MOVWF 06h

GOTO $
END
```

2. Exercise 1b. 8 LED Configuration B Assembly Code

```
;=====================================;
; Author: Charmbix                    ;
; Version: 2.0                        ;
; Course: LBYEC3F - EK1               ;
; Title: Experiment 1: Exercise 1     ;
;=====================================;

MOVLW B'00111100'
BSF 03h, 5
MOVWF 86h

MOVLW B'11000011'
BCF 03h, 5
```

```
MOVWF 06h

GOTO $
END
```

3. Exercise 2. 7-Segment Display Assembly Codes

3.1. 0_{10} in 7-Segment Display

```
;====================================;
; Author: Charmbix                   ;
; Version: 1.0                       ;
; Course: LBYEC3F – EK1              ;
; Title: Experiment 1: Exercise 2    ;
;====================================;

; Start
; Segment A
BSF 03h, 5
BCF 86h, 7
BCF 03h, 5
BSF 06h, 7

; Segment B
BSF 03h, 5
BCF 86h, 6
BCF 03h, 5
BSF 06h, 6

; Segment C
BSF 03h, 5
BCF 86h, 5
BCF 03h, 5
BSF 06h, 5

; Segment D
BSF 03h, 5
BCF 86h, 4
BCF 03h, 5
BSF 06h, 4

; Segment E
BSF 03h, 5
```

```
BCF 86h, 3
BCF 03h, 5
BSF 06h, 3

; Segment F
BSF 03h, 5
BCF 85h, 3
BCF 03h, 5
BSF 05h, 3

GOTO $
END
```

3.2. 1_{10} in 7-Segment Display

```
;===================================;
; Author: Charmbix                  ;
; Version: 1.0                      ;
; Course: LBYEC3F – EK1             ;
; Title: Experiment 1: Exercise 2   ;
;===================================;

; Start

; Segment B
BSF 03h, 5
BCF 86h, 6
BCF 03h, 5
BSF 06h, 6

; Segment C
BSF 03h, 5
BCF 86h, 5
BCF 03h, 5
BSF 06h, 5
GOTO $
END
```

3.3. 2_{10} in 7-Segment Display

```
;===================================;
; Author: Charmbix                  ;
; Version: 1.0                      ;
; Course: LBYEC3F - EK1             ;
```

```
; Title: Experiment 1: Exercise 2     ;
;=====================================;

; Start
; Segment A
BSF 03h, 5
BCF 86h, 7
BCF 03h, 5
BSF 06h, 7

; Segment B
BSF 03h, 5
BCF 86h, 6
BCF 03h, 5
BSF 06h, 6

; Segment D
BSF 03h, 5
BCF 86h, 4
BCF 03h, 5
BSF 06h, 4

; Segment E
BSF 03h, 5
BCF 86h, 3
BCF 03h, 5
BSF 06h, 3

;Segment G
BSF 03h, 5
BCF 85h, 2
BCF 03h, 5
BSF 05h, 2

GOTO $
END
```

3.4. 3_{10} in 7-Segment Display

```
;=====================================;
; Author: Charmbix                    ;
; Version: 1.0                        ;
; Course: LBYEC3F - EK1               ;
```

```
; Title: Experiment 1: Exercise 2     ;
;=====================================;

; Start
; Segment A
BSF 03h, 5
BCF 86h, 7
BCF 03h, 5
BSF 06h, 7

; Segment B
BSF 03h, 5
BCF 86h, 6
BCF 03h, 5
BSF 06h, 6

; Segment C
BSF 03h, 5
BCF 86h, 5
BCF 03h, 5
BSF 06h, 5

; Segment D
BSF 03h, 5
BCF 86h, 4
BCF 03h, 5
BSF 06h, 4

;Segment G
BSF 03h, 5
BCF 85h, 2
BCF 03h, 5
BSF 05h, 2

GOTO $
END
```

3.5. 4_{10} in 7-Segment Display

```
;=====================================;
; Author: Charmbix                    ;
; Version: 1.0                        ;
; Course: LBYEC3F - EK1               ;
```

```
; Title: Experiment 1: Exercise 2     ;
;=====================================;

; Start

; Segment B
BSF 03h, 5
BCF 86h, 6
BCF 03h, 5
BSF 06h, 6

; Segment C
BSF 03h, 5
BCF 86h, 5
BCF 03h, 5
BSF 06h, 5

; Segment F
BSF 03h, 5
BCF 85h, 3
BCF 03h, 5
BSF 05h, 3

;Segment G
BSF 03h, 5
BCF 85h, 2
BCF 03h, 5
BSF 05h, 2

GOTO $
END
```

3.6. 5_{10} in 7-Segment Display

```
;=====================================;
; Author: Charmbix                    ;
; Version: 1.0                        ;
; Course: LBYEC3F - EK1               ;
; Title: Experiment 1: Exercise 2     ;
;=====================================;

; Start
```

```
; Segment A
BSF 03h, 5
BCF 86h, 7
BCF 03h, 5
BSF 06h, 7

; Segment C
BSF 03h, 5
BCF 86h, 5
BCF 03h, 5
BSF 06h, 5

; Segment D
BSF 03h, 5
BCF 86h, 4
BCF 03h, 5
BSF 06h, 4

; Segment F
BSF 03h, 5
BCF 85h, 3
BCF 03h, 5
BSF 05h, 3

;Segment G
BSF 03h, 5
BCF 85h, 2
BCF 03h, 5
BSF 05h, 2

GOTO $
END
```

3.7. 6_{10} in 7-Segment Display

```
;===================================;
; Author: Charmbix                  ;
; Version: 1.0                      ;
; Course: LBYEC3F - EK1             ;
; Title: Experiment 1: Exercise 2   ;
;===================================;
```

```
; Start

; Segment A
BSF 03h, 5
BCF 86h, 7
BCF 03h, 5
BSF 06h, 7

; Segment C
BSF 03h, 5
BCF 86h, 5
BCF 03h, 5
BSF 06h, 5

; Segment D
BSF 03h, 5
BCF 86h, 4
BCF 03h, 5
BSF 06h, 4

; Segment D
BSF 03h, 5
BCF 86h, 3
BCF 03h, 5
BSF 06h, 3

; Segment F
BSF 03h, 5
BCF 85h, 3
BCF 03h, 5
BSF 05h, 3

;Segment G
BSF 03h, 5
BCF 85h, 2
BCF 03h, 5
BSF 05h, 2

GOTO $
END
```

3.8. 7_{10} in 7-Segment Display

```
;===================================;
; Author: Charmbix                  ;
; Version: 1.0                      ;
; Course: LBYEC3F - EK1             ;
; Title: Experiment 1: Exercise 2   ;
;===================================;

; Start

; Segment A
BSF 03h, 5
BCF 86h, 7
BCF 03h, 5
BSF 06h, 7

; Segment B
BSF 03h, 5
BCF 86h, 6
BCF 03h, 5
BSF 06h, 6

; Segment C
BSF 03h, 5
BCF 86h, 5
BCF 03h, 5
BSF 06h, 5

GOTO $
END
```

3.9. 8_{10} in 7-Segment Display

```
;===================================;
; Author: Charmbix                  ;
; Version: 1.0                      ;
; Course: LBYEC3F - EK1             ;
; Title: Experiment 1: Exercise 2   ;
;===================================;

; Start
```

```
; Segment A
BSF 03h, 5
BCF 86h, 7
BCF 03h, 5
BSF 06h, 7

; Segment B
BSF 03h, 5
BCF 86h, 6
BCF 03h, 5
BSF 06h, 6

; Segment C
BSF 03h, 5
BCF 86h, 5
BCF 03h, 5
BSF 06h, 5

; Segment D
BSF 03h, 5
BCF 86h, 4
BCF 03h, 5
BSF 06h, 4

; Segment E
BSF 03h, 5
BCF 86h, 3
BCF 03h, 5
BSF 06h, 3

; Segment F
BSF 03h, 5
BCF 85h, 3
BCF 03h, 5
BSF 05h, 3

;Segment G
BSF 03h, 5
BCF 85h, 2
BCF 03h, 5
BSF 05h, 2
```

```
GOTO $
END
```

3.10. 9_{10} in 7-Segment Display

```
;====================================;
; Author: Charmbix                   ;
; Version: 1.0                       ;
; Course: LBYEC3F - EK1              ;
; Title: Experiment 1: Exercise 2    ;
;====================================;

; Start

; Segment A
BSF 03h, 5
BCF 86h, 7
BCF 03h, 5
BSF 06h, 7

; Segment B
BSF 03h, 5
BCF 86h, 6
BCF 03h, 5
BSF 06h, 6

; Segment C
BSF 03h, 5
BCF 86h, 5
BCF 03h, 5
BSF 06h, 5

; Segment D
BSF 03h, 5
BCF 86h, 4
BCF 03h, 5
BSF 06h, 4

; Segment F
BSF 03h, 5
BCF 85h, 3
BCF 03h, 5
```

```
BSF 05h, 3

; Segment G
BSF 03h, 5
BCF 85h, 2
BCF 03h, 5
BSF 05h, 2

GOTO $
END
```

3.11. A_{16} in 7-Segment Display

```
;====================================;
; Author: Charmbix                   ;
; Version: 1.0                       ;
; Course: LBYEC3F - EK1              ;
; Title: Experiment 1: Exercise 2    ;
;====================================;

; Start

; Segment A
BSF 03h, 5
BCF 86h, 7
BCF 03h, 5
BSF 06h, 7

; Segment B
BSF 03h, 5
BCF 86h, 6
BCF 03h, 5
BSF 06h, 6

; Segment C
BSF 03h, 5
BCF 86h, 5
BCF 03h, 5
BSF 06h, 5

; Segment E
BSF 03h, 5
```

```
BCF 86h, 3
BCF 03h, 5
BSF 06h, 3

; Segment F
BSF 03h, 5
BCF 85h, 3
BCF 03h, 5
BSF 05h, 3

;Segment G
BSF 03h, 5
BCF 85h, 2
BCF 03h, 5
BSF 05h, 2

GOTO $
END
```

3.12. b_{16} in 7-Segment Display

```
;====================================;
; Author: Charmbix                   ;
; Version: 1.0                       ;
; Course: LBYEC3F - EK1              ;
; Title: Experiment 1: Exercise 2    ;
;====================================;

; Start

; Segment C
BSF 03h, 5
BCF 86h, 5
BCF 03h, 5
BSF 06h, 5

; Segment D
BSF 03h, 5
BCF 86h, 4
BCF 03h, 5
BSF 06h, 4
```

```
; Segment E
BSF 03h, 5
BCF 86h, 3
BCF 03h, 5
BSF 06h, 3

; Segment F
BSF 03h, 5
BCF 85h, 3
BCF 03h, 5
BSF 05h, 3

; Segment G
BSF 03h, 5
BCF 85h, 2
BCF 03h, 5
BSF 05h, 2

GOTO $
END
```

3.13. C_{16} in 7-Segment Display

```
;====================================;
; Author: Charmbix                   ;
; Version: 1.0                       ;
; Course: LBYEC3F - EK1              ;
; Title: Experiment 1: Exercise 2    ;
;====================================;

; Start

; Segment A
BSF 03h, 5
BCF 86h, 7
BCF 03h, 5
BSF 06h, 7

; Segment D
BSF 03h, 5
BCF 86h, 4
```

```
BCF 03h, 5
BSF 06h, 4

; Segment E
BSF 03h, 5
BCF 86h, 3
BCF 03h, 5
BSF 06h, 3

; Segment F
BSF 03h, 5
BCF 85h, 3
BCF 03h, 5
BSF 05h, 3

GOTO $
END
```

3.14. d_{16} in 7-Segment Display

```
;====================================;
; Author: Charmbix                   ;
; Version: 1.0                       ;
; Course: LBYEC3F - EK1              ;
; Title: Experiment 1: Exercise 2    ;
;====================================;

; Start

; Segment B
BSF 03h, 5
BCF 86h, 6
BCF 03h, 5
BSF 06h, 6

; Segment C
BSF 03h, 5
BCF 86h, 5
BCF 03h, 5
BSF 06h, 5

; Segment D
BSF 03h, 5
```

```
BCF 86h, 4
BCF 03h, 5
BSF 06h, 4

; Segment D
BSF 03h, 5
BCF 86h, 3
BCF 03h, 5
BSF 06h, 3

; Segment G
BSF 03h, 5
BCF 85h, 2
BCF 03h, 5
BSF 05h, 2

GOTO $
END
```

3.15. E_{16} in 7-Segment Display

```
;=====================================;
; Author: Charmbix                    ;
; Version: 1.0                        ;
; Course: LBYEC3F - EK1               ;
; Title: Experiment 1: Exercise 2     ;
;=====================================;

; Start

; Segment A
BSF 03h, 5
BCF 86h, 7
BCF 03h, 5
BSF 06h, 7

; Segment D
BSF 03h, 5
BCF 86h, 4
BCF 03h, 5
BSF 06h, 4
```

```
; Segment E
BSF 03h, 5
BCF 86h, 3
BCF 03h, 5
BSF 06h, 3

; Segment F
BSF 03h, 5
BCF 85h, 3
BCF 03h, 5
BSF 05h, 3

; Segment G
BSF 03h, 5
BCF 85h, 2
BCF 03h, 5
BSF 05h, 2

GOTO $
END
```

3.16. F_{16} in 7-Segment Display

```
;====================================;
; Author: Charmbix                   ;
; Version: 1.0                       ;
; Course: LBYEC3F - EK1              ;
; Title: Experiment 1: Exercise 2    ;
;====================================;

; Start

; Segment A
BSF 03h, 5
BCF 86h, 7
BCF 03h, 5
BSF 06h, 7

; Segment E
BSF 03h, 5
BCF 86h, 3
BCF 03h, 5
BSF 06h, 3
```

```
; Segment F
BSF 03h, 5
BCF 85h, 3
BCF 03h, 5
BSF 05h, 3

; Segment G
BSF 03h, 5
BCF 85h, 2
BCF 03h, 5
BSF 05h, 2

GOTO $
END
```