Experiment I

Sevim Eftal Akşehirli, 150190028
Hacer Yeter Akıncı, 150200006
Büşra Özdemir, 150200036
Aslı Yel, 150200054

***Abstract*—This experiment involves programming an MSP430G2553 microcontroller in assembly language to control LEDs and respond to input buttons. It consists of three parts: In Part 1, the provided code is analyzed, focusing on understanding instructions like "bis" and "mov." The goal is to comprehend the code and report the differences between these instructions. Also, The LED lights display time was adjusted using the busy waiting method. Part 2 requires modifying the code to make P1 and P2 LEDs work sequentially, changing the LED control sequence. Part 3 involves more complex modifications to achieve a specific LED pattern, and a button from PORT2 is assigned to stop the system when pressed. Overall, this experiment is a practical introduction to assembly language programming for the MSP430G2553 microcontroller. It covers LED manipulation, user interaction through buttons, and key programming concepts like bit manipulation, loops, and conditional branching. A strong understanding of MSP430G2553 assembly language programming is necessary to complete these tasks successfully.**

***Index Terms*—Education Board, Microcontroller, Assembly, LEDs, Output, Push Button, Input, Port, Busy Waiting, Looping, Conditional Branch, Label, Instruction, bis, mov etc.**

I. Introduction

This experiment highlights the importance of learning to control the LEDs of the MSP430 microcontroller board. LEDs are used as basic indicators in many electronic devices and help develop basic electronic skills. Additionally, this experiment includes not only learning LED control but also basic microcontroller programming. Understanding the MSP430 instruction set is a huge advantage in the code development process. Additionally, this experiment teaches the fundamentals of microcontroller systems by teaching assembly language programming, providing a foundational skill for more advanced projects. This experiment also highlights the importance of microcontrollers, focusing on technologies that offer low power consumption, fast processing capabilities, and a wide range of applications. These technologies are used in many areas such as embedded systems and Internet of Things (IoT).

II. Materials and Methods

In this experiment, these materials are used:

- Çizgi-Tagem MSP Education Board with a Texas Instruments produced MSP-EXP430G2 Launchpad
- Texas Instruments Code Composer Studio (CCS) IDE installed computer
- 'MSP430 Education Board Manual', 'MSP430 Architecture Chapter 4,6' and 'MSP430 Instruction Set' Documents

Education Board is connected to the computer with a data cable. A CCS Project with "MSP430G2553" as the target and Empty Assembly-only Project as the project template is created.

For different behaviors of LEDs and push buttons, different assembly codes are written, run and results are observed on the board. MSP430 Instruction Set is checked, and codes are written accordingly.

Listing 1: Assembly Code for Part1

```
SetupP1
        mov.b   #11111111b, &P1DIR
        mov.b   #11111111b, &P2DIR
        mov.b   #00000000b, &P1OUT
        mov.b   #00000000b, &P2OUT
        mov.b   #00010000b, R6
        mov.b   #00001000b, R7
        mov.b   #0d, R8

Mainloop1
        bis.b   R7, &P2OUT
        inc     R8
        rra     R7
        mov.w   #50000000, R15
        jmp     L1

Mainloop2
        mov.b   #00000000b, &P2OUT
        bis.b   R6, &P1OUT
        inc     R8
        rla     R6
        mov.w   #50000000, R15
        jmp     L2

L1
        dec.w   R15
        jnz     L1
        cmp     #4d, R8
        jeq     Mainloop2
        jmp     Mainloop1

L2
        dec.w   R15
        jnz     L2
        cmp     #8d, R8
        jeq     SetupP1
        jmp     Mainloop2
```

PART1:

SetupP1: This section sets the P1 and P2 ports as output, R6 initial value as 00010000b, R7 initial value as 00001000b, and the R8 as 0 for counting the states of led patterns.

Mainloop1: This section combines R7 with the P2OUT registers (bitwise OR operation) and writes the result to the P2OUT register. R7 arithmetically rotates left for the LEDs to light sequentially. It loads a very large number into the R15 register for the wait operation. R8 increases by one. The loop repeats until R8 reaches the decimal number 4. Jumps to the label L1.

Mainloop2: Clears the output of P2 port for obtain output only from P1 port. This section combines R6 with the P1OUT registers (bitwise OR operation) and writes the result to the P1OUT register. R6 arithmetically rotates right so that the LEDs light in sequence. It loads a very large number into the R15 register for the wait operation. R8 increases by one. The loop repeats until R8 reaches the decimal number 8. Jumps to the label L2.

L1: This label symbolizes a loop. It performs the wait operation and checks the value of R8. Mainloop1 and Mainloop2 increment R8 by one. When the R8 value is less than 4, it jumps to the Mainloop1 label, otherwise it jumps to the Mainloop2 label.

L2: This label also symbolizes a loop. Similar to L1, it controls the value of R8 by performing a waiting process. When the R8 value is less than 8 it jumps to the Mainloop2 label, otherwise it jumps to the SetupP1 label and stops the code.

Listing 2: Assembly Code for Part2

```
SetupP1
            mov.b   #11111111b, &P1DIR
            mov.b   #11111111b, &P2DIR
            mov.b   #00000000b, &P1OUT
            mov.b   #00000000b, &P2OUT
            mov.b   #10000000b, R6
            mov.b   #0d, R8

Mainloop1
            mov.b   R6, &P1OUT
            inc     R8
            rra     R6
            mov.w   #50000000, R15
            jmp     L1

L1
            dec.w   R15
            jnz     L1
            jmp     Mainloop2

Mainloop2
            mov.b   R6, &P2OUT
            inc     R8
            rra     R6
            mov.w   #50000000, R15
            jmp     L2

L2
            dec.w   R15
            jnz     L2
            cmp     #8d, R8
            jeq     SetupP1
            jmp     Mainloop1
```

PART2:

SetupP1: This section sets the P1 and P2 ports as output, initializes R6 with the value 10000000b, and initializes R8 as a loop variable for counting up to 8.

Mainloop1: It updates the output pins of P1 based on R6, increments the loop counter (R8), performs a right rotation on R6 using rra. It sets R15 register to a huge value for delay.

L1: It is a delay loop.

Mainloop2: It works as Mainloop1, but it updates the output port as P2OUT instead of P1OUT.

L2: It is a delay loop for Mainloop2. It checks if R8 equal is 8. If R8 is not 8, the program continues to MainLoop1. If R8 is 8, it jumps back to SetupP1, resets and starts the sequence again.

Listing 3: Assembly Code for Part3

```
SetupP1
            mov.b   #11111111b, &P1DIR
            mov.b   #00000000b, &P2DIR
            mov.b   #00000000b, &P1OUT
            mov.b   #10000000b, R6
            mov.b   #0d, R8

Mainloop1
            bis.b   R6, &P1OUT
            inc     R8
            rra     R6
            mov.w   #50000000, R15
            jmp     L1

L1
            dec.w   R15
            jnz     L1
            bit.b   #00000001b, &P2IN
            jne     End
            cmp     #8d, R8
            jeq     SetupP1
            jmp     Mainloop1

End
            jmp     End
```

PART3:

SetupP1: This section sets the P1 as output and P2 as input, initializes R6 with the value 10000000b, and initializes R8 as a loop variable for counting up to 8.

Mainloop1: In this phase, we use bis.b instruction to set R6 to P1OUT. The reason of using bis is preserve the previous state of P1. It increments R8, shifts right R6 using rra. It loads a large value into R15 register for delay.

L1: It is a time delay for Mainloop1. It checks if first bit of P2IN is high. If it is high, it enters an infinite loop. If it is not, the program continues to sequence. Additionally, it checks if R8 equal is 8. If R8 is not 8, the program continues to MainLoop1. If R8 is 8, it jumps back to SetupP1, resets and starts the sequence again.

End: It is an infinite loop which stops the process.

III. Results

For the first part of the experiment the assembly code was run and the result was observed from the MSP430 Education Board. It was obtained as expected which is seen in the picture.

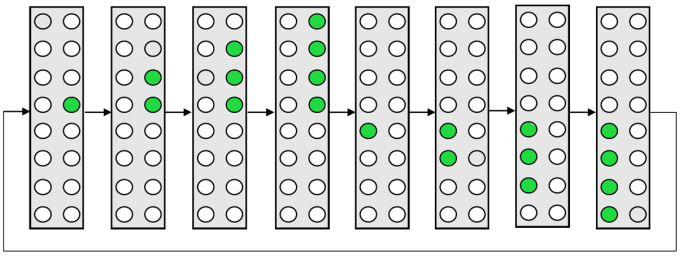

Fig. 1: Result of Part-1

For the second part of the experiment the assembly code was run and the result was observed from the MSP430 Education Board. It was obtained as expected which is seen in the picture.

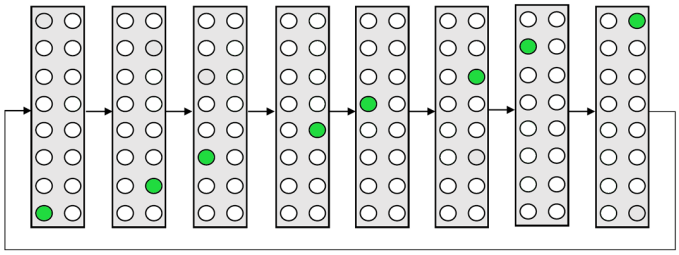

Fig. 2: Result of Part-2

For the third part of the experiment the assembly code was run and the result was observed from the MSP430 Education Board. It was obtained as expected which is seen in the picture.

It also preserves the current state of the led pattern whenever the specified button from P2 is pushed.

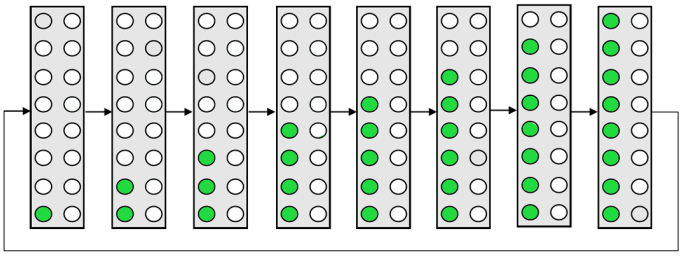

Fig. 3: Result of Part-3

IV. Discussion and Summary

Our laboratory experience improved our understanding of assembly language. We gained an understanding of fundamental assembly instructions, such as ”bit”, ”cmp” and grasped the difference of ”mov” and ”bis” operations. CCS was a new environment for us, but we got used to work with it in a short time. For the first time we worked on an equipment like MSP430. We learned how to configure directions of ports as inputs or outputs, and control LEDs of MSP430 education board.

In conclusion, this laboratory experience appears to have provided us with valuable insights into both technical knowledge and teamwork and learning skills. This experience will prove beneficial for our future projects and learning process.