ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 351E

MICROCOMPUTER LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	1
EXPERIMENT DATE	:	08.11.2020
LAB SESSION	:	FRIDAY - 14.30
GROUP NO	:	G7

GROUP MEMBERS:

150210087 : HİLAL KARTAL

150210100 : SELİN YILMAZ

150210067 : ALPER DAŞGIN

FALL 2024

Contents

1 INTRODUCTION [10 points]

In this experiment, we worked directly with the MSP430 microcontroller to control the LEDs using assembly language programming. Using the MSP430G2553 as our target device using CCS, we created a project focused on tuning and managing the microcontroller's ports to produce specific LED patterns.

The experiment had three key components. Initial LED Configuration and Pattern Generation to create a repeating LED pattern. We first configured Ports 1 and 2 for the LED output, set all LEDs to the off state, and then wrote the assembly code using bitwise operations and register manipulation.

In PART 2, we modified the original code to create a new LED pattern that would cause the LEDs in Port1 and Port2 to turn on and off alternately, giving the impression of a "working light." .

Finally, we wrote the program by adding a button input from Port2. This gave us the ability to pause and resume the LED sequence with the press of a button, adding a basic interactive control feature. While working on each section, we studied and used different assembly instructions, including $mov, bis, and\, jnz$, and gained practical experience with bitwise operations, looping, and input processing at the assembly level.

1.1 LED ORGANIZATION AND P2.0, P2.1 FIX

At first we could not figure out how the LEDs were organized so we wrote the below code to figure it out.

```
1 SetupP1      mov.b #00000000b, &P2SEL
2              mov.b #11111111b,&P1DIR
3              mov.b #11111111b,&P2DIR
4 ; b -> bit ; h -> hexadecimal ; d -> decimal ; .b -> byte
5              mov.b #00000000b,&P1OUT
6              mov.b #00000000b,&P2OUT
7              mov.b #10000000b, R6
8
9 Mainloop1    bis.b R6, &P1OUT
```

Listing 1: LEDfiguration

- At 1st line, we fix the $P2.0$ and $P2.1$. So that they can be selected as output pins.
- At 2nd and 3rd line, we set each led as output.

- At 4th and 5th line, all leds are set to 0 initially
- At 6th line, $R6$'s most significant bit is set to 1 so that we can see which LED will shine as the MSB.
- At 8th line, $MainLoop1$ starts and $R6$'s is transfered to $P1$'s output by bitwise AND.

2 MATERIALS AND METHODS [40 points]

2.1 MATERIALS

- MSP430
- Selin's Laptop

2.2 METHODS

2.2.1 FIRST PART

In the first part, the aim is observing the led lights similar to figure 1.

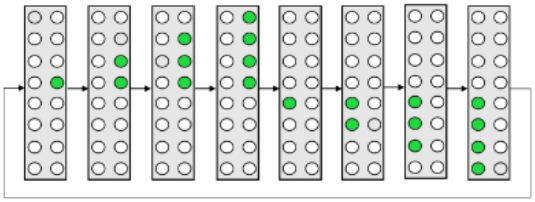

Figure 1: Part1 - expected leds

In order to do that, we first did the `SetupP1` part.

```
1 SetupP1 mov.b   #00000000b, &P2SEL
2         mov.b   #11111111b, &P1DIR
3         mov.b   #11111111b, &P2DIR
4         mov.b   #00000000b, &P1OUT
5         mov.b   #00000000b, &P2OUT
6         mov.b   #00010000b, R6
7         mov.b   #00001000b, R7
8         mov.b   #0d, R8     ;for(int R8=0; R8<4; R8++)
9         mov.b   #0d, R9     ;for(int R9=0; R9<4; R9++)
```

Listing 2: Part1 - SetupP1

Line 1: We initialized every P2 select to 0 because the laptop we used made P2's 6th and 7th bit 0.

Line 2-3: We made every P1 and P2's bits 1 to be able to use them as output later.

Line 4-5: We initialized all outputs of P1 and P2 to 0.

Line 6: We initialized R6 for the *Mainloop2*.

Line 7: We initialized R7 for the *Mainloop1*.

Line 8: We assigned 0 to R8. We will use this R8 to keep track of the iteration in *Mainloop1*.

Line 9: We did same thing for R9 and *Mainloop2*.

Then, we wrote the first loop (`Mainloop1`).

```
1  Mainloop1    mov.b    #00000000, &P1OUT
2               bis.b    R7, &P2OUT
3               inc      R8            ;R8<-R8+1
4               rra      R7
5               mov.w    #00500000, R15
6               dec.w    R15
7               jnz      L1
8               cmp      #4d, R8       ;if R8==4d
9               jeq      Mainloop2
10              jmp      Mainloop1
```

Listing 3: Part1 - Mainloop1

Line 1: We made all the leds on left side (P1) 0 (off).

Line 2: We ORed R7 and P2OUT to decide which leds of right side (P2) were going to be on.

Line 3: We increment R8 in order to indicate when the loop ends.

Line 4: We shifted R7 to right arithmetically e.g when R7=2_00001000 if we shift it right it will become R7=2_00000100.

Line 5: We assign a big word (32-bit) to R15.

Line 6-7: We decrement R15 until it becomes 0. We did this part in order to observe led light changes. Because it creates a delay between each light switch.

Line 8-9-10: We compare R8 with 4. If they are equal we branch to *Mainloop2*. If they are not, we repeat the same loop.

Then we wrote second loop `Mainloop2`.

```
1  Mainloop2    mov.b    #00000000b, &P2OUT
2               bis.b    R6, &P1OUT
3               inc      R9               ;R9<-R9+1
4               rla      R6               ;rotate left arithmetically
```

```
5               mow.w   #00500000, R15
6               dec.w   R15
7               jnz     L2
8               cmp     #4d, R9          ;if(R9==4d)
9               jeq     SetupP1
10              jmp     Mainloop2
```

Listing 4: Part1 - Mainloop2

Line 1: We made all the leds on right side (P2) 0 (off).

Line 2: We ORen R6 and P1OUT to decide which leds of left side (P1) were going to be on.

Line 3: We increment R9 in order to indicate when the loop ends.

Line 4: We shifted R6 to left arithmetically e.g when R6=2_00010000 if we shift it left it will become R6=2_00100000.

Line 5: We assign a big word (32-bit) to R15.

Line 6-7: We decrement R15 until it becomes 0. We did this part in order to observe led light changes. Because it creates a delay between each light switch.

Line 8-9-10: We compare R9 with 4. If they are equal we branch to *SetupP1*. If they are not, we repeat the same loop.

2.2.2 SECOND PART

In the second part, the aim is observing a result similar to figure 2.

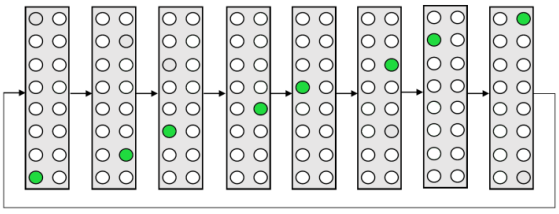

Figure 2: Part2 - expected leds

Again we started with setting up the initializations.

```
1 SetupP1   mov.b #00000000b, &P2SEL
2             mov.b #11111111b, &P1DIR
3             mov.b #11111111b, &P2DIR
4             mov.b #00000000b, &P1OUT
5             mov.b #00000000b, &P2OUT
6             mov.b #10000000b, R6
7             mov.b #0d, R8
```

Listing 5: Part2 - SetupP1

Line 1: In order to use every pin we select all of P2.

Line 2-3: We made every P1 and P2 bit 1 to be able to use them as output later.

Line 4-5: We initialized outputs to 0.

Line 6: We chose R6's MSB (chip's bottom pins are msb so in order to start from there we do the initializations according to big endian logic).

Line 7: We initialized our tracker R8 to 0.

Then, we wrote the body part.

```
1  Mainloop1    mov.b #00000000b, &P2OUT  ;P1 yanacak
2               bis.b R6, &P1OUT
3               inc   R8
4               rra   R6
5               mov.w #00500000, R15
6  L1           dec.w R15
7               jnz   L1
8               mov.b #00000000b, &P1OUT
9               bis.b R6, &P2OUT
10              inc   R8
11              rra   R6
12              mov.b #00500000, R15     ;delay to be able to observe the
    led
13 L2           dec.w R15
14              jnz   L2
15              cmp   #8d, R8
16              jeq   SetupP1
17              jmp   Mainloop1
```

Listing 6: Part2 - Main

Line 1: We initialized P2's output pins to 0.

Line 2: R6 and P1OUT are *OR*ed and written on P1OUT.

Line 3: Tracker R8 is incremented.

Line 4: R6 is shifted to right arithmetically.

Line 5-6-7: In order to observe the led light there is a delay.

Line 8: P1's output pins are initialized to 0.

Line 9: R6 and P2OUT are *OR*ed and written on P2OUT.

Line 10: Tracker R8 is incremented.

Line 11: R6 is shifted to right arithmetically.

Line 12-13-14: This part is also a delay.

Line 15: Compare tracker R8 with 8 (decimal).

Line 16: If R8 is equal to 8 program returns to Setup1.

Line 17: If R8 is not equal to 8 program goes to Mainloop1 in order to continue.

2.2.3 THIRD PART

In the third part, the code needs to work like figure 3. Then, a button is needed to be assigned and the code stops when it is pressed.

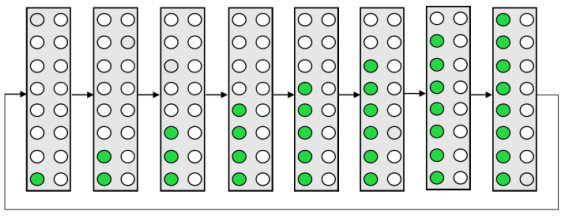

Figure 3: Part3 - expected leds

First we did the setup.

```
1  SetupP1      mov.b #00000000b, &P2SEL
2               mov.b #11111111b, &P1DIR
3               mov.b #11110111b, &P2DIR     ;input port
4               mov.b #00000000b, &P1OUT
5               mov.b #00000000b, &P2OUT     ;set pullup resistor
6               mov.b #10000000b, R6
7               mov.b #0d, R8
```

Listing 7: Part3 - SetupP1

Line 1: In order to use every pin we select all of P2.

Line 2: We made every P1 bit 1 to be able to use them as output later.

Line 3: We chose 4th pin of P2 as input pin by making it 0.

Line 4-5: We initialized outputs of P1 and P2 as 0.

Line 6: We chose R6's MSB (chip's bottom pins are msb so in order to start from there we do the initializations according to big endian logic).

Line 7: We initialized our tracker R8 to 0.

Then, we wrote the body part.

```
1  Mainloop1    bis.b R6, &P1OUT
2               inc   R8
3               rra   R6
4               mov.w #00500000, R15
5  L1           cmp   #00001000b, &P2IN
6               jeq   stop
7               dec.w   R15
8               jnz   L1
9               cmp   #8d, R8
10              jeq   SetupP1
```

```
11              jmp    Mainloop1
12 stop         jmp    stop
```

Listing 8: Part3 - Main

Line 1: R6 and P1OUT are ORed and written on P1OUT.

Line 2: Tracker R8 is incremented.

Line 3: R6 is shifted to right arithmetically.

Line 4: We assign a big word (32-bit) to R15.

Line 5-6: We check if 4th button of P2 is pressed. If yes branch to *stop* function.

Line 7-8: We decrement R15 until it becomes 0. We did this part in order to catch the P2 input.

Line 9-10-11: We compare our tracker R8 with 8. If it is equal to 8 we branch to *SetupP1*. If not we branch to *Mainloop1*.

Line 12: This is a endless loop. This state happens when the P2's 4th button is pressed.

3 RESULTS [15 points]

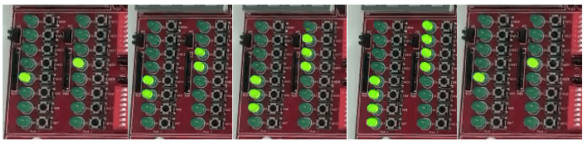

Figure 4: Initial code

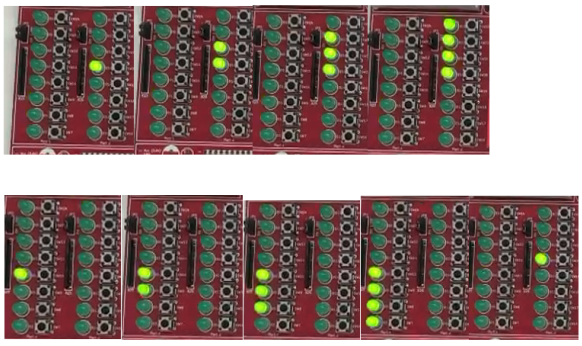

Figure 5: Part 1 results

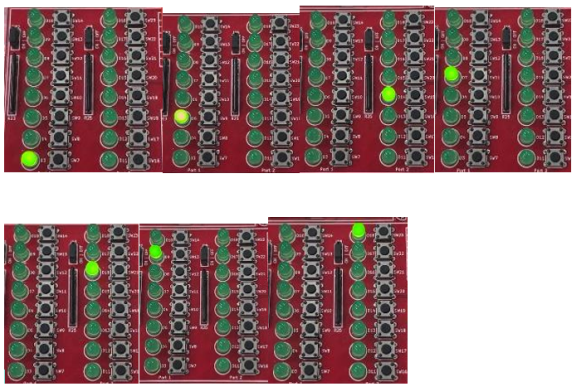

Figure 6: Part 2 results

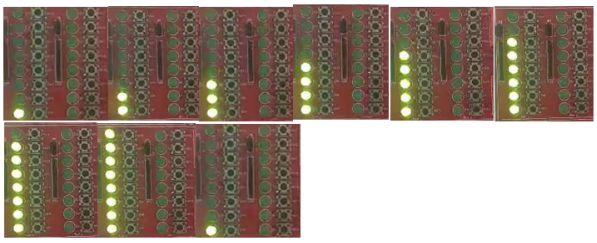

Figure 7: Part 3 results

4 DISCUSSION [25 points]

1. In the first part we were asked to explain the difference between *bis* and *mov*.

   - *BIS*: Sets the bits in destination. It ORs *src* and *dest* and loads it to destination register.
   - *MOV*: Moves the source to destination register.

2. In the first part, a four-state pattern was generated by the LEDs in Ports 1 and 2, rotating through a repeating sequence as expected. Bit shifts and register values controlled which lights would turn on and off in which grouping.

3. In the second part, the LEDs of both Port 1 and Port 2 were turned on sequentially in a "running light" pattern, resulting in seamless switching from one LED to the next. Infinite loops on the pattern have been completed successfully.

4. In the final section, the LED layout looked as it should by adding the ability to stop the sequence by pressing a button on Port 2. Pressing the button instantly stopped the LED array. Implementing *rra* (rotate right arithmetic) and *rla* (rotate left arithmetic) allowed us to dynamically change LED states;

5 CONCLUSION [10 points]

- With this experiment, we got familiarized with the MSP430 chip and wrote some code to see the LEDs light up in different cycles.
- We learned how $Input - Output\,Ports$ work on the chip and how to use them in our code.
- We have struggled the understand the LEDs organization at first but we wrote a test code to figure it out.
- We also struggled to see some of our results due to using a laptop and it's default configurations but by adding a fixer code line we were able to see the desired results.

REFERENCES

[1] Microcomputer Lab. Micro_experiment_1. *Lab Booklet*, 2024.