Department of Computer Engineering

BLG 351E
Microcomputer Laboratory
Experiment Report

Experiment No	: 1
Experiment Date	: 09.10.2017
Group Number	: Monday - 8
Group Members	:

ID	Name	Surname
150150114	EGE	APAK
150140066	ECEM	GÜLDÖŞÜREN
150150701	YUNUS	GÜNGÖR
Click here to enter text.	Click here to enter text.	Click here to enter text.

Laboratory Assistant : Ahmet Arış

1 INTRODUCTION

In this experiment, we introduced with the MSP430 Education Board, MSP430G2553 microcontroller and MSP430 assembly language. We used the program that is called “Code Composer Studio” to wrote our assembly code, and we built, debugged the assembly code that we wrote by using this program and also we loaded our code to the board.

In the first part, we wrote the code that had been given for us in the experiment sheet to the place marked with “Main loop here” on our “main.asm” file. And we connected the board to our computer through USB connection. We built, run and loaded the code to the board and observed the behavior of the board.

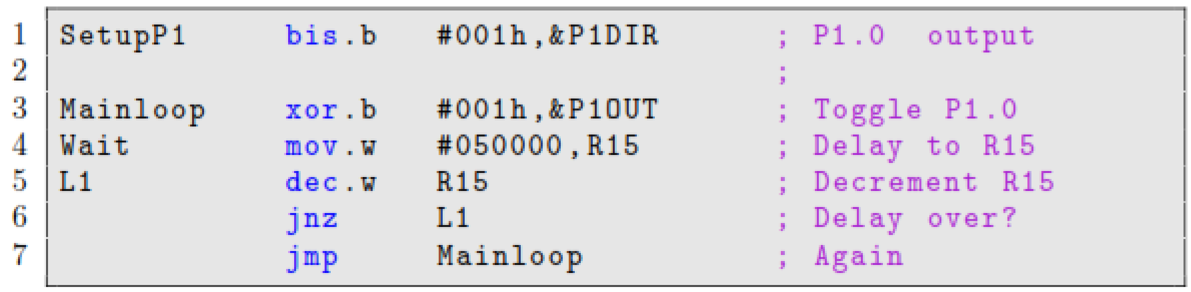
In the second part, we modificated the given code in order to change the behavior of the LEDs. After code adjustment, LEDs started to light on and of sequentially just as it’s expressed in the experiment sheet.

In the third part, we again adjusted the assembly code one more time. Thanks to these adjustments, we managed to turn on and off the LEDs sequentially as it’s expressed in the lab experiment sheet.

2 EXPERIMENT

2.1 PART 1

In this part, we placed the code that was provided us in the experiment sheet at the location that is specified. This code can been observed from the Picture 2.1.1 that is placed below.



```
1  SetupP1     bis.b   #001h,&P1DIR        ; P1.0  output
2                                          ;
3  Mainloop    xor.b   #001h,&P1OUT        ; Toggle P1.0
4  Wait        mov.w   #050000,R15         ; Delay to R15
5  L1          dec.w   R15                 ; Decrement R15
6              jnz     L1                  ; Delay over?
7              jmp     Mainloop            ; Again
```

Picture 2.1.1: the main code that is loaded to Code Composer Studio.

After we wrote this code, we loaded it to the board through CSS. We built and debugged the code in order to understand how the board works. We observed that, the first LED was turning on and off continuously.

If we investigate the code ; SetupP1, MainLoop, Wait and L1 are the labels of the lines.

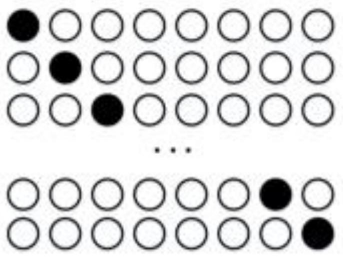
In the first line, we set the bits in destination and we specify the direction of the P1set, if P1Set is equal to 1, then P1 acts like output, otherwise P1 acts like input. In this code, P1 is set to be output.

In the third line we apply XOR operation with P1OUT and x001. So if the least significant bit is equal to 0 after XOR operation, it becomes 1 and the first LED turns on. And if the least significant bit is equal to 1, after the XOR operation it becomes 0 and LED turns off. Shortly, we specify which LED will be turned on with XOR operation.

In the forth line, with mov.w command, we assign the value of x050000 to the general purpose register R15. We decrement this value in the next line with the dec.w command. (While decrementing this value, we actually make it possible to observe the turning on and off sequence. If we assign bigger value to R15, then the LED stays on a little longer since decrementing of bigger value takes longer time.) We keep decrementing this value until it becomes zero thanks by jumping back to L1 line (where decrementing happens) thanks to jnz command. When the value in the general purpose register R15 becomes zero, then Zero flag becomes 1 and it does not jump to the line L1 and it continues from 6th line. Program continues to apply this operations because it jumps back to Mainloop line thanks to jmp operation from 7th line. The first LED of the board keeps turning on and turning off during the execution of the code.

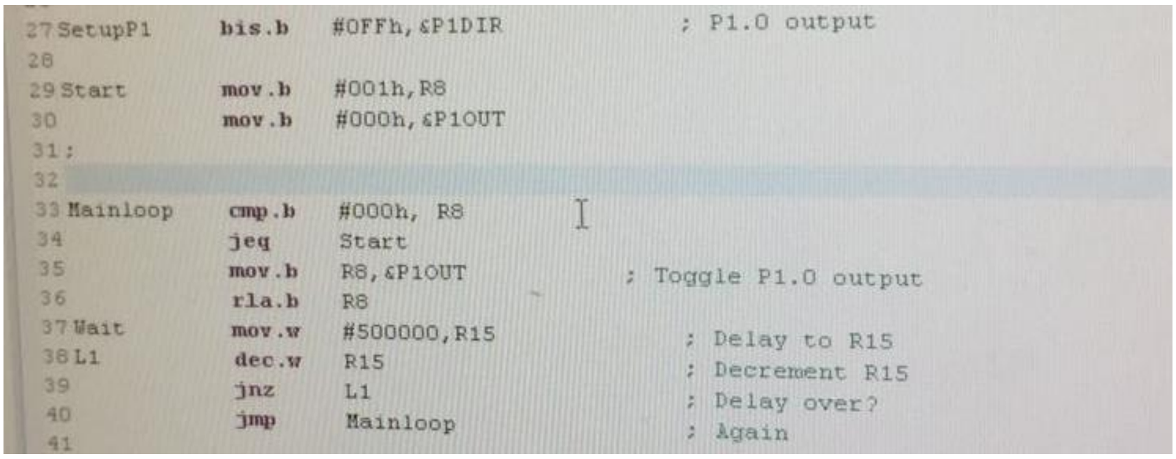
2.2 PART 2

In this experiment, we modified our code in the first part in order to change the behavior of LEDs so that we can observe the sequential operation of the LEDs as shown below.



Picture 2.2.1: sequential operation of LEDs

The assembly code of ours can be seen as shown below.



```
27 SetupP1      bis.b   #0FFh,&P1DIR          ; P1.0 output
28
29 Start        mov.b   #001h,R8
30              mov.b   #000h,&P1OUT
31 ;
32
33 Mainloop     cmp.b   #000h, R8
34              jeq     Start
35              mov.b   R8,&P1OUT           ; Toggle P1.0 output
36              rla.b   R8
37 Wait         mov.w   #500000,R15           ; Delay to R15
38 L1           dec.w   R15                   ; Decrement R15
39              jnz     L1                    ; Delay over?
40              jmp     Mainloop              ; Again
41
```

Picture 2.2.2: the assembly code for part2 of the experiment

Firsly, in the line 27, we set the bits of P1DIR with #0FFh, using “bis.b” command so that all lambs(bitts) can be used as output.

In the line, we use “mov.b” command to set the bits of general purpose register R8. I’ll use this register as counter. (There is no specific reason for selecting R8 register, any of the general purpose register could have been selected.)

In the line 30, using “mov.b” command; we moved #000h to P1OUT just to make sure that, the value of all the LED bits (lambs) are 0 and none of them turns on.

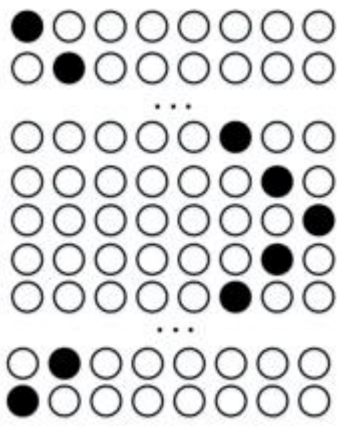
In the main loop, firsly we check if the value in R8 register is #000h or not with “cmp.b” command and if the result of the comparison operation is true, we go back to the line which is labeled as START and start all over again. The reason for checking if the value of R8 is #000h or not is actually shifting operation. Because when we shift the value (1000000) to the left, 1 goes to the carry and the value becomes (0000 0000). In order to keep the sequence, we need to transform this value to (0000 0001).

When the program starts, the least significant 1 by making the value of LSB 1 ant the other bits 0 in the R8 register; and then we move the value to P1OUT. So the first LED lights up.

In the 36th line, we use "rla.b" command to rotate left the value of R8. (This means the next LED is ready to turn on.) In the lines 37-40 we perform same operations that are described in the first part of the experiment. We assign #500000 to general purpose register R15, we start to decrement this value until it becomes 0, when Zero flag becomes 1, program jumps back to the beginning of the mainloop and obtains the same operations in order to turn up the next light.

2.3 Part 3

In this part of the experiment, we modified our assembly code in Part2 in order to change the behavior of LEDs so that they are turned on and turned off sequentially as shown below.



Picture 2.3.1: Sequential Operation of the LEDs

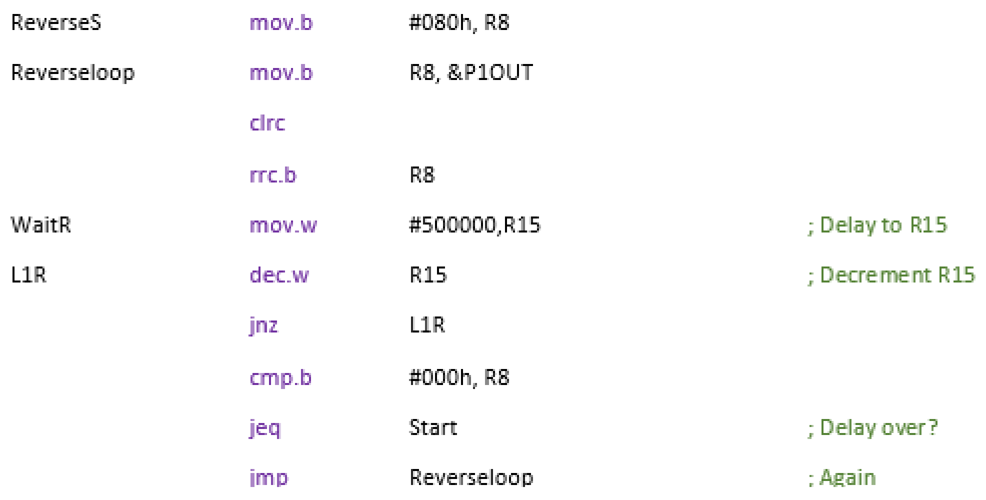
Our code implementation is shown below too.

```
SetupP1     bis.b     #0FFh, &P1DIR        ; P1.0 output
            mov.b     #000h, &P1OUT
Start       mov.b     #001h, R8


Mainloop    cmp.b     #000h, R8
            jeq       ReverseS
            mov.b     R8, &P1OUT           ; Toggle P1.0 output
            rla.b     R8
Wait        mov.w     #500000, R15         ; Delay to R15
L1        . dec.w     R15                  ; Decrement R15
            jnz       L1                   ; Delay over?
            jmp       Mainloop             ; Again
```

Picture 2.3.1: first part of the assembly code

In this part of the experiment, we firstly



```
ReverseS        mov.b   #080h, R8
Reverseloop     mov.b   R8, &P1OUT
                clrc
                rrc.b   R8
WaitR           mov.w   #500000,R15             ; Delay to R15
L1R             dec.w   R15                     ; Decrement R15
                jnz     L1R
                cmp.b   #000h, R8
                jeq     Start                   ; Delay over?
                jmp     Reverseloop             ; Again
```

Picture 2.3.3: assembly code (continue)

In the first line; we firstly used “bis.b” command and set the bits of P1DIR with (#0FFh) in order to let all of the 8 bits (LEDs) can be used as output.

Secondly, we used “mov.b” command and moved #000h to P1OUT just to make sure that, the value of all of the LED bits (lambs) are 0 and none of them turns on.

In the line that we labeled as Start, we also used “mov.b” command to move (#001h) to the general purpose register R8. We used the value of the R8 as counter.

We used 2 loops in this code in order to keep the sequence the same as what we’re asked to do.

Mainloop compares the value in the R8 register with (#000h) and check if they’re the same or not. If there not the same this means that we don’t need to shift right yet, we do the same operations with the code of 2nd part of the experiment. The only difference in this Mainloop from the 2nd part’s Mainloop is that here, we jump to ReverseS in the case of the value of R8 is (#000h). This meas we need to start right shifting.

In the line that we labeled as ReverseS, we use “mov.b” command in order to load (#080h) to the our counter R8.

In the Reverseloop, we again use “mov.b” command to move the content of R8 to P1OUT and the necessary LED turns on.

Then we clear our carry bit with “clrc” command.

We rotate right through carry the content of the R8 with “rrc.b” command.

Then we used “mov.w” command to move (#500000) to general purpose register R15 and we kept decrementing this value until it becomes 0.

We check if this value became (#000h) or not, if it is then this means that we need to stop shifting right and start shifting left. So if it is equal to (#000h) then we jump back to the Start label again. If it is not (#000h) then this means that we need to keep shifting right so we jump back to Reverseloop.

3 CONCLUSION

In this experiment, we learnt how to use CCS, how to built, run, debug an assembly code on CCS. We also learnt how to load the assembly code to board. We explored the structure of the board.

We had some trouble with computers. The number of computers were not enough and this situation lead some disarrangements. But laboratory assistant gave us extra laptops and allowed us to use our own computers so this situation solved immediately.