Department of Computer Engineering

BLG 351E
Microcomputer Laboratory
Experiment Report

Experiment No : 1

Experiment Date : 10/10/16

Group Number : M - 13

Group Members :

ID	Name	Surname
150130032	Baran	Kaya
40130051	Halil İbrahim	Onuş
150130002	Ahmet Seha	Çelenk

Laboratory Assistant : Gökhan Seçinti

1 INTRODUCTION

In this experiment, we have learned typing, building and debugging assembly code using Code Composer Studio and MSP430&kit.

2 EXPERIMENT

Experiment 1 is consisting of two parts. In the first part, the basic led blinkink code written in experiment document is typed on CCS and uploaded to MSP430. In the second part, a code is typed by us for scrolling LED lights sequentally upward and downward on board.

2.1 FIRST PART

The code given us is typed below:

```
SetupP1         bis.b   #001h,  &P1DIR                  ;P1.0 output
                                                        ;
Mainloop        xor.b   #001h,  &P1OUT                  ;Toggle P1.0

Wait            mov.w   #050000,R15                     ;Delay to R15

L1              dec.w   R15                             ;Decrement R15
                jnz     L1                              ;Delay over?
                Jmp     Mainloop                        ;Again
```

In the first line, bis.b sets and selects first bit (0000 0001) as output with setting direction to out via P1DIR. This is for using only one LED to blinking. This setup is named SetupP1, regarding to the P1 register.

After that, in the second line xor.b is used with operands #001h (0000 0001) value and P1OUT (it was #001h, as well) to toggle first led. P1OUT sets P1 register as an output device/bit. XOR operation for 2 cases we have in this program is shown below:

$$0000\ 0001 \oplus 0000\ 0001 = 0000\ 0000 \text{ (led off)}$$

$$0000\ 0000 \oplus 0000\ 0001 = 0000\ 0001 \text{ (led on)}$$

This results show that xor operation with constant value #001h toggles LSB, switching between 0 and 1. Then this values used on P1 register to turn LED on and off. This function is named as Mainloop.

In the next function Wait, a word (#050000) is copied(moved) to general purpose register R15. This value is stored for using at L1 function later on.

At the L1 stage, dec.w R15 decrements the value on GPR R15 which was 50000 by one and checks if it has decremented to 0 via jnz, if not, it jumps to L1 again and does the same decrement in the first line. It is performed till value on R15 reaches zero, which makes jnz(jump if not zero) line being passed by. This decrementing process gives us enough time to see the result with eyes. It is similar to delay functions on severel high level programming languages. After that, Jmp Mainloop is called to jump Mainoop again.

This program is repeated and a LED is being toggled until user terminates it.

2.2 SECOND PART

In the second part, a program is typed for scrolling LED lights sequentally upward and downward on board. Whole of the 8 LEDs is used this time. Related program is shown below:

```
SetupP1         bis.b   #0FFh,&P1DIR                ;P1's all pins are output
                bic.b   #0FFh,&P1OUT                ;Clearing all bits of outputs P1
                xor.b   #080h,&P1OUT                ;Toggle P1.7

L1              clrc
                rrc.b   &P1OUT
                jc      CLR1
                jmp     Wait1

Wait1           mov.w   #050000,R15                 ;Delay to R15

L3              dec.w   R15                         ;Decrement R15
                jnz     L3                          ;Delay over?
                jmp     L1                          ;

L2              clrc
                rlc.b   &P1OUT
                jc      CLR2
                jmp     Wait2

CLR1            rlc.b   &P1OUT
                clrc
                jmp     L2
CLR2            rrc.b   &P1OUT
                clrc
                jmp     L1

Wait2           mov.w   #050000,R14                 ;Delay to R15

L4              dec.w   R14                         ;Decrement R15
                jnz     L4                          ;Delay over?
                jmp     L2                          ;
```

In SetupP1 function, the 8 bits are set in outwards direction via P1DIR and they set as output device with P1OUT. The value #0FFh corresponds 0000 1111 1111, as we use only 8 leds, we can be think as the value selects these 8 bitsas output. bic clears these 8 bits for the first launch. Then xor is used to toggle first led on row. Operand are #000h (initial value of leds) and #080h (1000 0000). Corresponding operation is shown below:

$$0000\ 0000 \oplus 1000\ 0000 = 1000\ 0000$$

L1 loop is clears the carry bit first. After being sure the C is 0, it continues wtth rrc.b which rotates 8 bits right through C bit. This means LSB is assigned to C and C is assigned to MSB, as well as other bits are shifted right. Now, initial values of the output bits are:

$$0100\ 0000\ (C=0)$$

As we can see, the light is shifted by one row. Now it checks whether the carry value is set (equals 1) or not with jc. This line is passed by for now. Then by jmp, it jumps to function Wait1, assigns the word 050000 to GPR R15, which will be used on delay operations in the following lines.

As we have mentioned before (in the First Part), L3 delays for a while by decrementing 50000 to 0 and gives us the ability to see the results. Then it jumps to L1 again.

This loop is occurred until the following results are got.

0010 0000 (C=0)

0001 0000 (C=0)

0000 1000 (C=0)

0000 0100 (C=0)

0000 0010 (C=0)

0000 0001 (C=0)

0000 0000 (C=1)

The final value activates the jc(jump if C is set) line and makes us jump to CLR1 function.

CLR1 is a function that first shifts the carry value to left by rlc.b (rotate left through C), and then clears the carry bit to avoid any errors by clrc. The initial value at that time is:

0000 0001 (C=0)

After that, it jumps to L2, which is the same but left rotated version of L1. It performs the following lines:

0000 0010 (C=0)

0000 0100 (C=0)

0000 1000 (C=0)

0001 0000 (C=0)

0010 0000 (C=0)

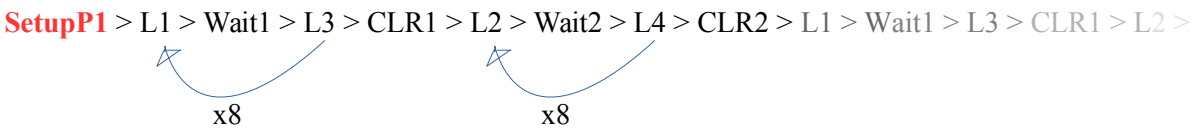
0100 0000 (C=0)

1000 0000 (C=0)

0000 0000 (C=1)

It jumps CLR2 to reverse the sequence back, as performed with CLR1.

An example order of operations related to function names is shown sequentally below:



These sequential blinkings is performed till user terminates the program.

3 CONCLUSION

This experiment is helped us to learn basic operations such as typing, building and debugging operations on CCS. Because of working on a hardware kit for the first time, we have seen that we can face with connection problems a lot. And also using assembly language for the first time made us a bit confused during the experiments. But we have observed the underlying structure of high level programming languages such as C, C++, Java etc. and we can say that this was a great experience to learn all those new things. This may help us learning the mentality of some of the high level programming features later on.