Department of Computer Engineering

BLG 351E
Microcomputer Laboratory
Experiment Report

Experiment No : 6

Experiment Date : 28.11.2016

Group Number : Monday - 13

Group Members :

ID	Name	Surname
040130051	Halil İbrahim	Onuş
150130002	Ahmet Seha	Çelenk
150130032	Baran	Kaya

Laboratory Assistant : Gökhan Seçinti

1 INTRODUCTION

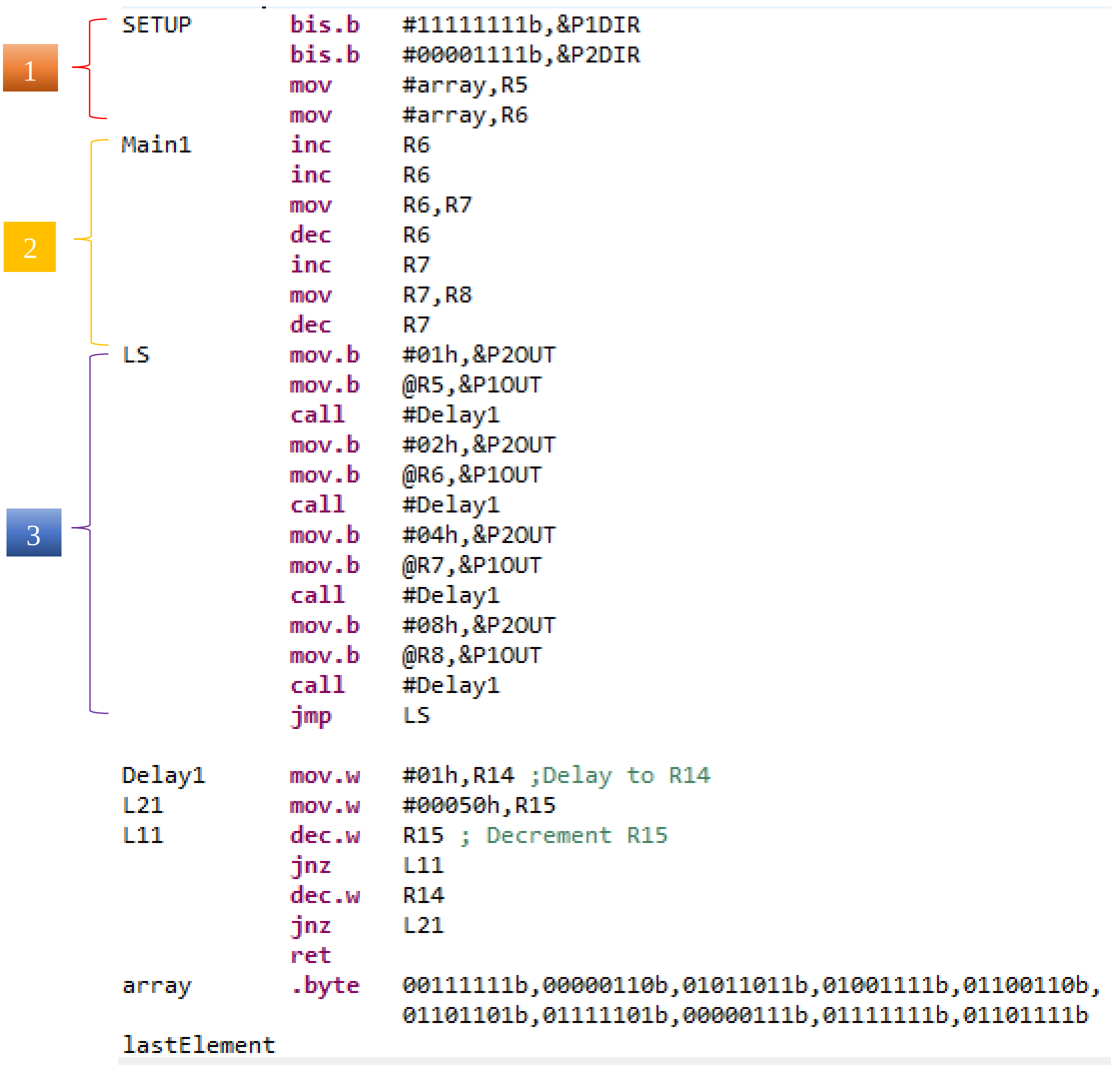
In this experiment we have learn how to define an array and usage of register as pointer, also we have been more familiar to branching. We implemented these algorithms in assembly: Bit-wise Encryption and Bubble Sort.

2 EXPERIMENT

Experiment 6 consists of two parts. First part is implementing program code in order to lit different digits of 7-segment display panel simultaneously. Second part is building chronometer with seconds and centiseconds. Chronometer also has stop interrupt feature.

2.1 PART 1

Assembly code that implement part-1 is:



```
SETUP       bis.b   #11111111b,&P1DIR
            bis.b   #00001111b,&P2DIR
            mov     #array,R5
            mov     #array,R6
Main1       inc     R6
            inc     R6
            mov     R6,R7
            dec     R6
            inc     R7
            mov     R7,R8
            dec     R7
LS          mov.b   #01h,&P2OUT
            mov.b   @R5,&P1OUT
            call    #Delay1
            mov.b   #02h,&P2OUT
            mov.b   @R6,&P1OUT
            call    #Delay1
            mov.b   #04h,&P2OUT
            mov.b   @R7,&P1OUT
            call    #Delay1
            mov.b   #08h,&P2OUT
            mov.b   @R8,&P1OUT
            call    #Delay1
            jmp     LS

Delay1      mov.w   #01h,R14 ;Delay to R14
L21         mov.w   #00050h,R15
L11         dec.w   R15 ; Decrement R15
            jnz     L11
            dec.w   R14
            jnz     L21
            ret
array       .byte   00111111b,00000110b,01011011b,01001111b,01100110b,
                    01101101b,01111101b,00000111b,01111111b,01101111b
lastElement
```

1 In this part of code, we give a direction to all bits of P1 as using output that give 8-bit input to 7-segment display from array. After that we give direction to P2 for selecting 4-digit 7 segment display. Also we move R5 and R6 as pointers to array.

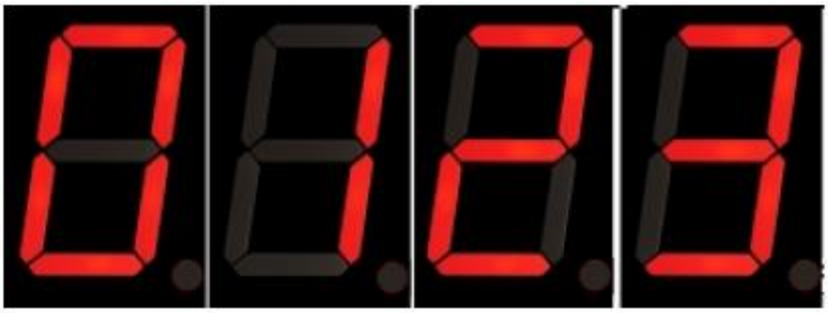
2 R5and R6 is our flag that shows array’s first element that is “0”. By incrementing and decrementing R6 and R7 values and assigning R7 and R8 registers is about to select “0123” input as an input of 4-digit 7-segment display.

inc R6: $R5 \rightarrow 0\,, R6 = R5 + 1 \Rightarrow R6 = 1$

inc R6: $R6_{new} = R6 + 1 \Rightarrow R6_{new} = 1 + 1 = 2$
mov R6,R7:$R7 = R6_{new} = 2$
dec R6:$R6 = R6_{new} - 1 = 1$ (This is for protect R6 values)

inc R7: $R7_{new} = R7 + 1 \Rightarrow R7_{new} = 2 + 1 = 3$
mov R7,R8:$R8 = R7_{new} = 3$
dec R7:$R7 = R7_{new} - 1 = 2$ (This is for protect R7 values)

3 After that we move values in R5, R6, R7, R8 registers to 7-segment display’s first, second, third and last digits. For displaying values, we implemented infinite loop and obtained this result:



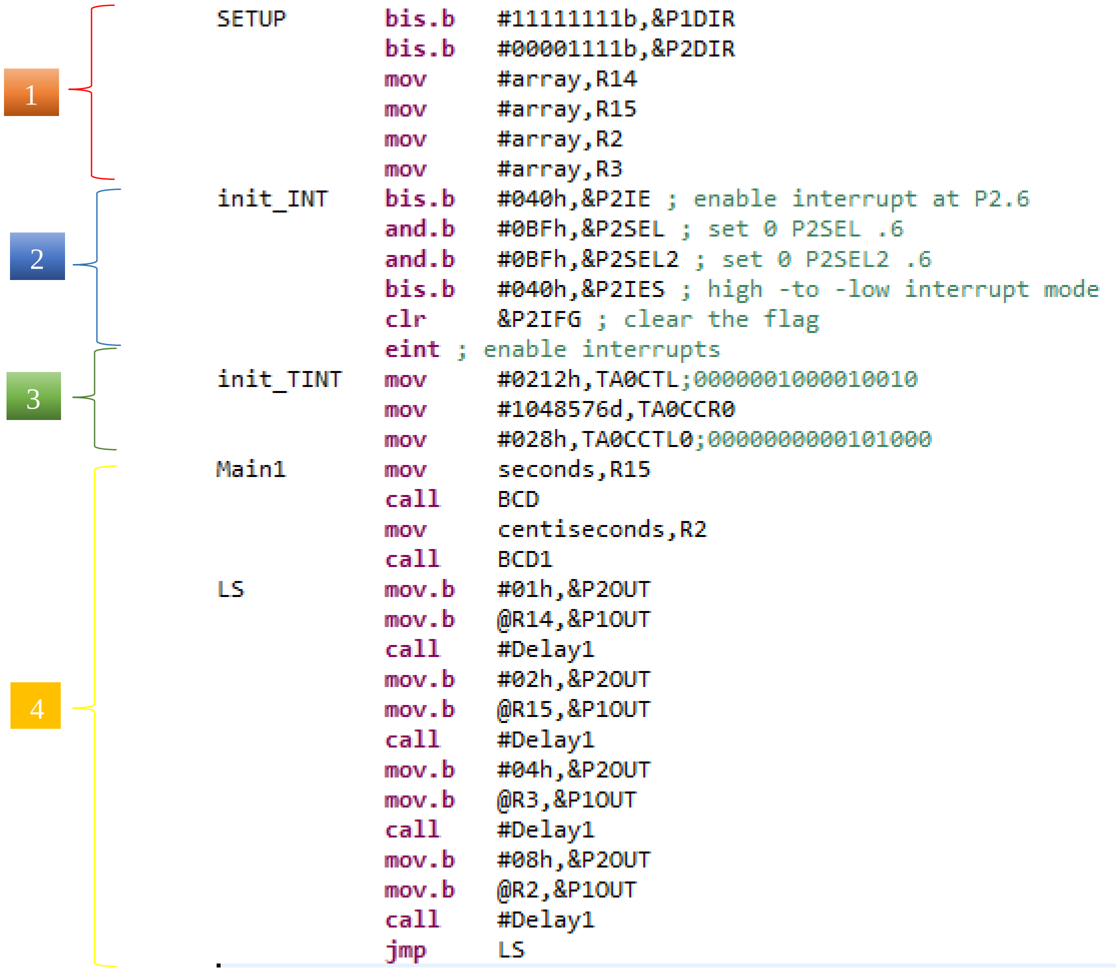
2.2 PART 2

In order to implement chronometer, we add followings:

- Timer Interrupt Subroutine: we use 16-bit Timer-A

- Interrupt Subroutine: we use button on P2.6 for start OR stop chronometer

- Convert Subroutine: conversion of seconds & centiseconds is done by BCD



```
SETUP       bis.b   #11111111b,&P1DIR
            bis.b   #00001111b,&P2DIR
            mov     #array,R14
            mov     #array,R15
            mov     #array,R2
            mov     #array,R3
init_INT    bis.b   #040h,&P2IE ; enable interrupt at P2.6
            and.b   #0BFh,&P2SEL ; set 0 P2SEL .6
            and.b   #0BFh,&P2SEL2 ; set 0 P2SEL2 .6
            bis.b   #040h,&P2IES ; high -to -low interrupt mode
            clr     &P2IFG ; clear the flag
            eint ; enable interrupts
init_TINT   mov     #0212h,TA0CTL;0000001000010010
            mov     #1048576d,TA0CCR0
            mov     #028h,TA0CCTL0;0000000000101000
Main1       mov     seconds,R15
            call    BCD
            mov     centiseconds,R2
            call    BCD1
LS          mov.b   #01h,&P2OUT
            mov.b   @R14,&P1OUT
            call    #Delay1
            mov.b   #02h,&P2OUT
            mov.b   @R15,&P1OUT
            call    #Delay1
            mov.b   #04h,&P2OUT
            mov.b   @R3,&P1OUT
            call    #Delay1
            mov.b   #08h,&P2OUT
            mov.b   @R2,&P1OUT
            call    #Delay1
            jmp     LS
```

1 In this part of code, we give direction to P1&P2 ports as defined in "PART 1.1". We move registers R2&R3(for centiseconds) and R14&R15 (for seconds) to array for conversion to BCD.

2 Interrupt subroutine initialize part. We move P2.6 button for start OR stop interruption usage.

3 Timer Interrupt subroutine initialize part. We move Timer-A Control(TA0CTL), Timer-A Compare Capture(TA0CCR0), Timer-A Comp. Cap. Control(TA0CCTL0) through followings:

TA0CTL

- SMCLK signal as counting input $bits\ 9-8 => 10$

- Up mode: the timer counts up to TACCR0 $bits\ 5-4 => 01$

- Interrupt enabled $bit\ 1 => 1$

TA0CCR0

- value to store register in order to create timer interrupts with 10 millisecond 1048576

TA0CCTL0

- No capture $bits\ 15-14 => 00$

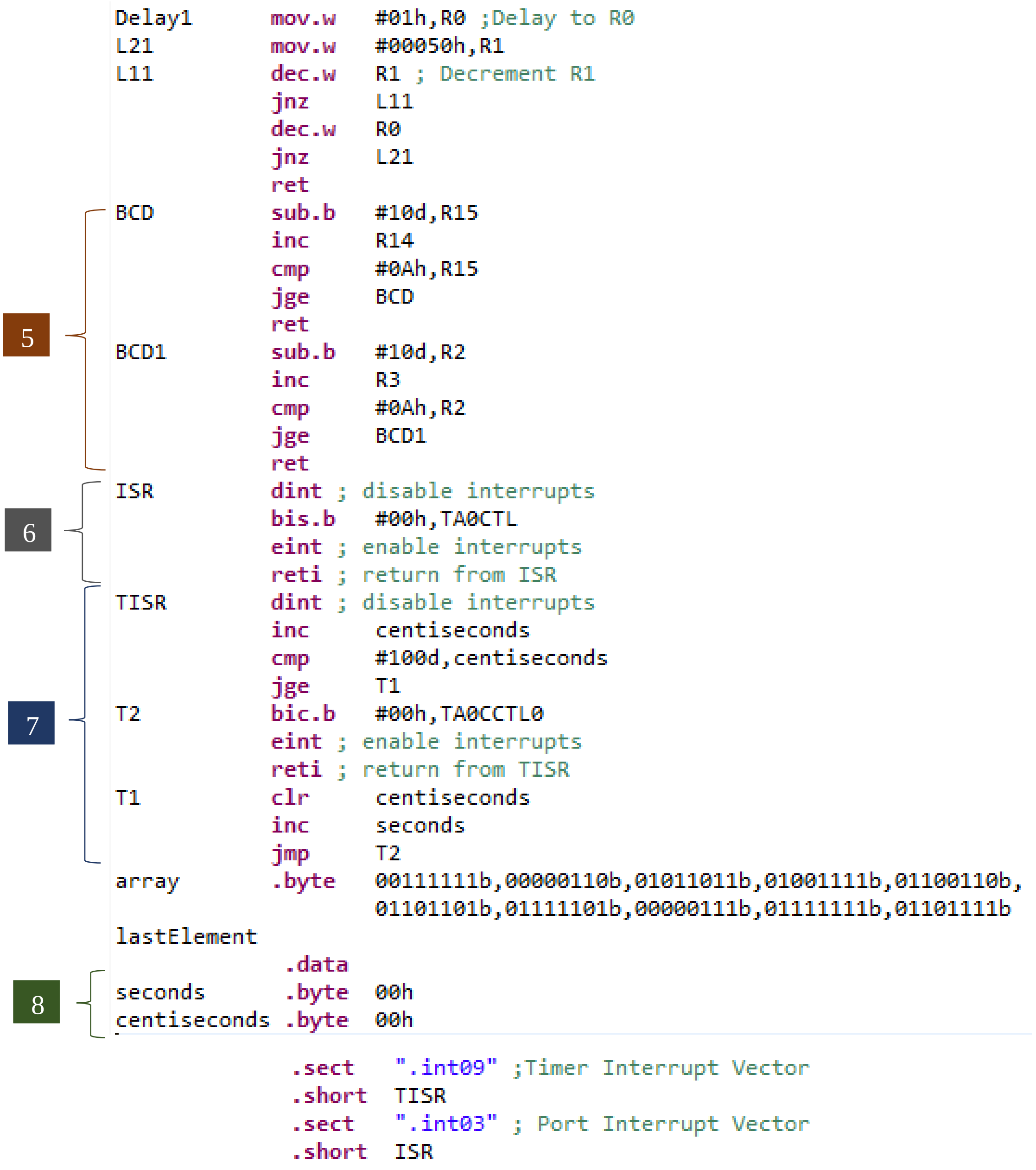
- Compare mode $bit\ 8 => 0$

- Output mode: SET $bits\ 7-5\ 00 => 001$

- Interrupt disabled *bit* 4 => 0

- Capture/compare input *bit* 3 => 1

4 Second and centisecond values inside RAM are moved to R15 and R2 registers respectively. Then call BCD conversion subroutines “BCD” for seconds, “BCD1” for centiseconds. In LS loop displaying second &centisecond on 7-segment is implemented.



```
Delay1      mov.w   #01h,R0 ;Delay to R0
L21         mov.w   #00050h,R1
L11         dec.w   R1 ; Decrement R1
            jnz     L11
            dec.w   R0
            jnz     L21
            ret
BCD         sub.b   #10d,R15
            inc     R14
            cmp     #0Ah,R15
            jge     BCD
            ret
BCD1        sub.b   #10d,R2
            inc     R3
            cmp     #0Ah,R2
            jge     BCD1
            ret
ISR         dint ; disable interrupts
            bis.b   #00h,TA0CTL
            eint ; enable interrupts
            reti ; return from ISR
TISR        dint ; disable interrupts
            inc     centiseconds
            cmp     #100d,centiseconds
            jge     T1
T2          bic.b   #00h,TA0CCTL0
            eint ; enable interrupts
            reti ; return from TISR
T1          clr     centiseconds
            inc     seconds
            jmp     T2
array       .byte   00111111b,00000110b,01011011b,01001111b,01100110b,
                    01101101b,01111101b,00000111b,01111111b,01101111b
lastElement
              .data
seconds       .byte  00h
centiseconds .byte  00h

              .sect   ".int09" ;Timer Interrupt Vector
              .short  TISR
              .sect   ".int03" ; Port Interrupt Vector
              .short  ISR
```

5 BCD conversion subroutines. Read values from seconds & centiseconds RAM addresses saved in R15 and R2 registers respectively. Subtract R15 and R2 with 10 then increase R14&R3 values as a second digit of seconds and centiseconds part until R15 and R2 values below 10.

6 Interrupt subroutine. By $bis.b\ \#00h, TA0CTL$ we clear Timer-A interrupt flag.

7 Timer Interrupt subroutine. Increase centiseconds value whenever there is an interrupt from the timer (at TISR label). When centiseconds value reaches 100, it clears the value of centiseconds and increase

seconds value by one (at T2 label). After that Timer-A Comp. Cap. Control(TA0CCTL0) register's Capture/compare interrupt flag (bit-0) is cleared.

8 In order to store second and centisecond values inside RAM.

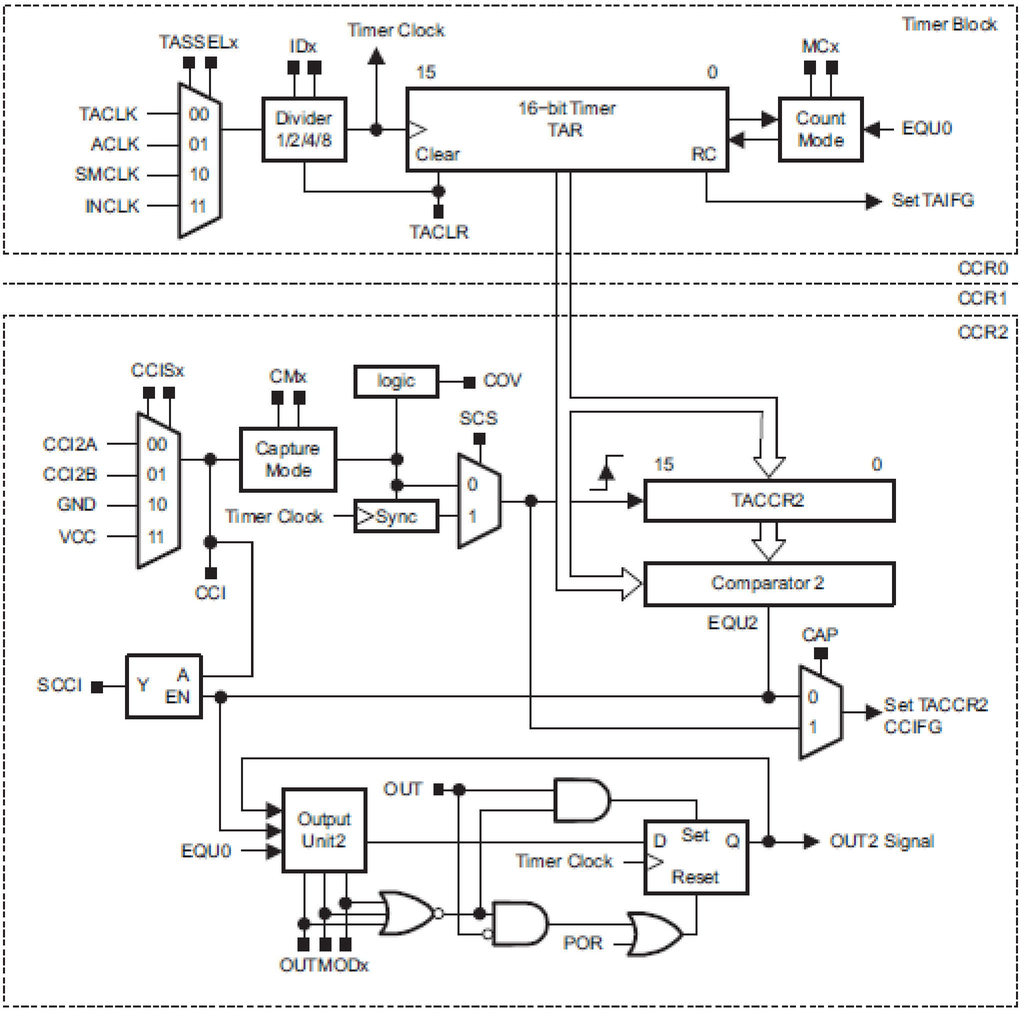


Figure 12-1. Timer_A Block Diagram

3 CONCLUSION

We have learned how to display different values at 4 digit 7-segment display. We try to implement chronometer with TISR, ISR and BCD-convert. We think our design is good but we have some mistakes lead not to work.