CS1050-Computer organization and digital design

Lab 8 – Assembly Programming

Student Name: De Silva A.D.D.T.

Index Number: 220098C

Group Number: 33

1. Introduction

Main task of this lab is to learn Assembly programming and interfacing simple output devices. Assembly is the most basic programming language available for any microprocessor. Learning assembly language is well worth the time and effort of every serious programmer.

In this lab, we use Microprocessor Simulator V5.0 (smz32) developed by Neil Bauers, to develop and simulate assembly programs. The smz32 simulator emulates an 8-bit CPU.

2. Basic Mathematical Operations

- Sample codes for Addition, Subtraction, Multiplication, and Division

- Addition

```
; ===== WORK OUT 2 PLUS 2 ======================================
        CLO             ; Close unwanted windows.
        MOV AL,2        ; Copy a 2 into the AL register.
        MOV BL,2        ; Copy a 2 into the BL register.
        ADD AL,BL       ; Add AL to BL. Answer goes into AL.
        END             ; Program ends
; ===== Program Ends ===========================================
```

- Subtraction

```
        CLO             ; Close unwanted windows.
        MOV AL,4        ; Copy a 4 into the AL register.
        MOV BL,2        ; Copy a 2 into the BL register.
        SUB AL,BL       ; Sub AL by BL. Answer goes into AL.
        END             ; Program ends
; ===== Program Ends ===========================================
```

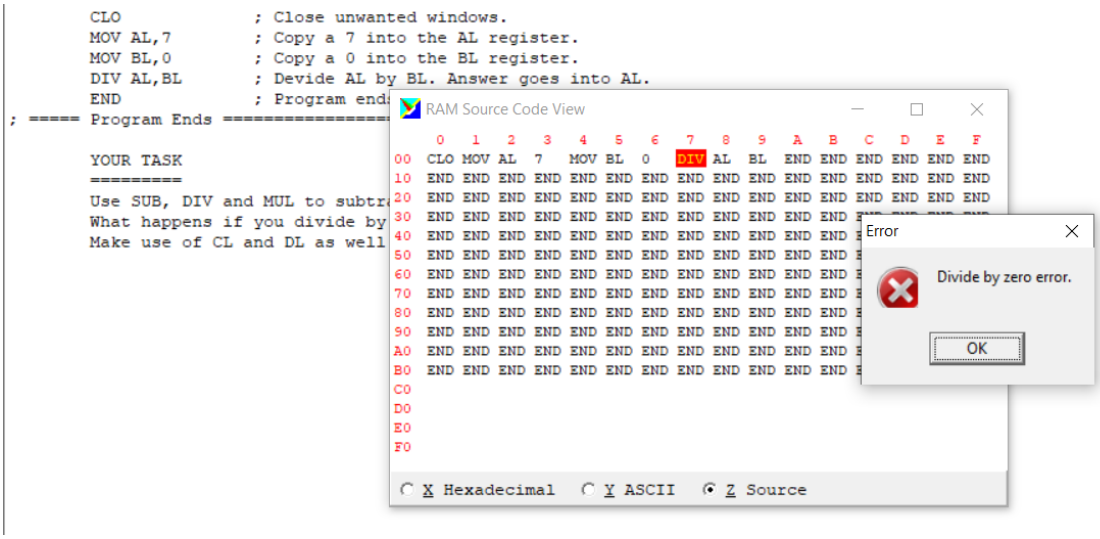
- Multiplication

```
        CLO             ; Close unwanted windows.
        MOV AL,7        ; Copy a 7 into the AL register.
        MOV BL,2        ; Copy a 2 into the BL register.
        MUL AL,BL       ; Multiply AL by BL. Answer goes into AL.
        END             ; Program ends
; ===== Program Ends ============================================
```

- Division

```
        CLO             ; Close unwanted windows.
        MOV AL,6        ; Copy a 6 into the AL register.
        MOV BL,2        ; Copy a 2 into the BL register.
        DIV AL,BL       ; Divide AL by BL. Answer goes into AL.
        END             ; Program ends
; ===== Program Ends ============================================
```

- What happen if divided by zero?



3. Traffic Lights

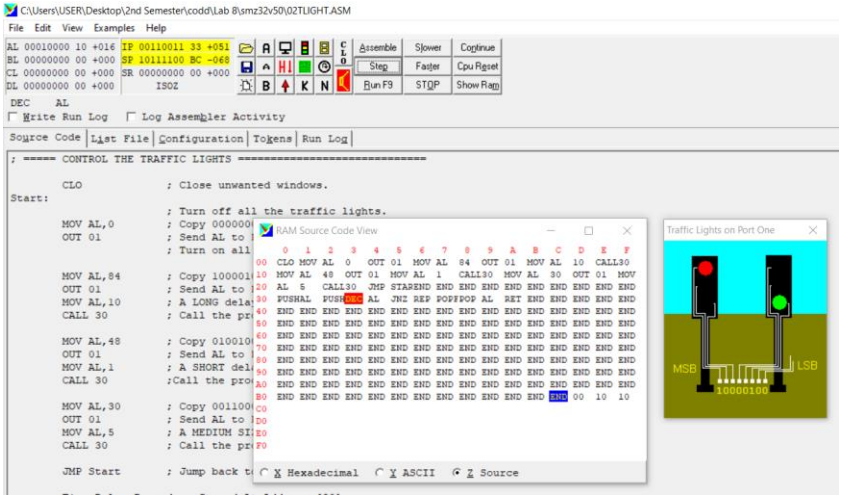
- My approach including how I implemented delaying.

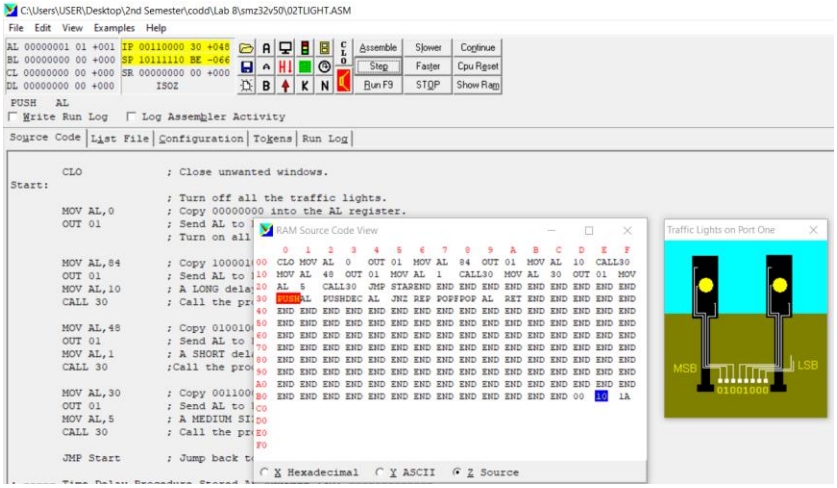
I referred “06PROC.ASM” file to learn an efficient way of introducing a delay to a code. Then I modified “02TLIGHT.ASM” program to make it real by controlling the time that each light stays on.

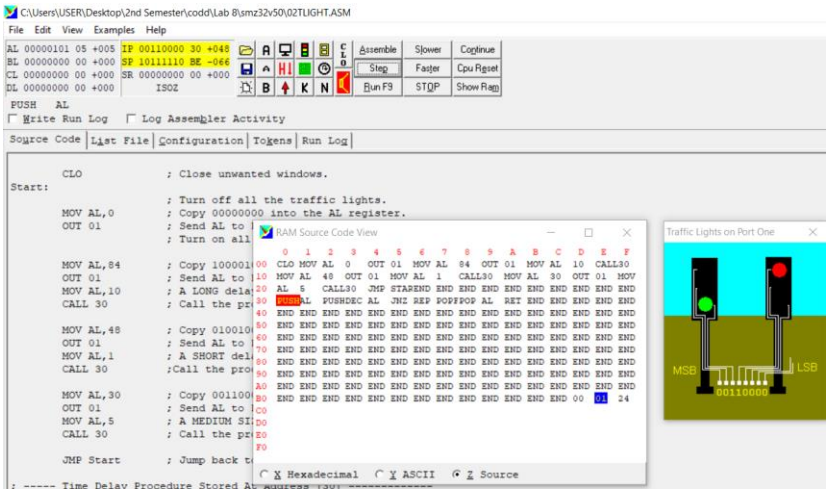
- Sample code

```
; ===== CONTROL THE TRAFFIC LIGHTS ==============================
        CLO             ; Close unwanted windows.
Start:

        MOV AL,84       ; Copy 10000100 into the AL register.
        OUT 01          ; Send AL to Port One (The traffic lights).
        MOV AL,10       ; A LONG delay.
        CALL 30         ; Call the procedure at address [30]

        MOV AL,48       ; Copy 01001000 into the AL register.
        OUT 01          ; Send AL to Port One (The traffic lights).
        MOV AL,1        ; A SHORT delay.
        CALL 30         ;Call the procedure at address [30]

        MOV AL,30       ; Copy 00110000 into the AL register.
        OUT 01          ; Send AL to Port One (The traffic lights).
        MOV AL,5        ; A MEDIUM SIZE delay.
        CALL 30         ; Call the procedure at address [30]

        JMP Start       ; Jump back to the start.

; ----- Time Delay Procedure Stored At Address [30] -------------
        ORG     30      ; Generate machine code from address [30]

        PUSH    AL      ; Save AL on the stack.
        PUSHF           ; Save the CPU flags on the stack.
Rep:
        DEC     AL      ; Subtract one from AL.
        JNZ     REP     ; Jump back to Rep if AL was not Zero.

        POPF            ; Restore the CPU flags from the stack.
        POP     AL      ; Restore AL from the stack.

        RET             ; Return from the procedure.
; ---------------------------------------------------------------
        END
; ---------------------------------------------------------------
; ===== Program Ends ============================================
```

- Screenshots of the output





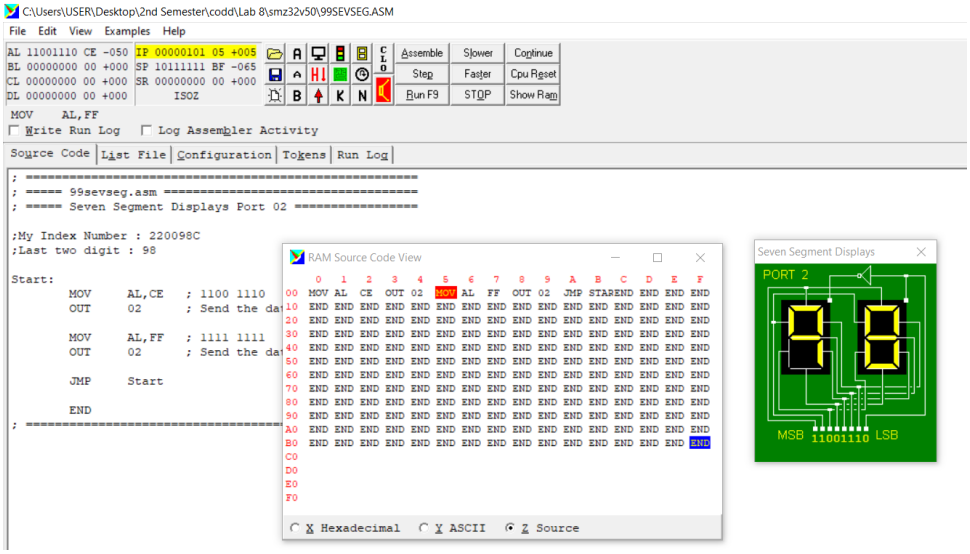


4. Seven-Segment Display

- Sample code

```
; ========================================================
; ===== 99sevseg.asm ====================================
; ===== Seven Segment Displays Port 02 ==================

;My Index Number : 220098C
;Last two digit : 98

Start:
        MOV     AL,CE    ; 1100 1110
        OUT     02       ; Send the data in AL to Port 02

        MOV     AL,FF    ; 1111 1111
        OUT     02       ; Send the data in AL to Port 02

        JMP     Start

        END
; ========================================================
```

- Screenshot of the output

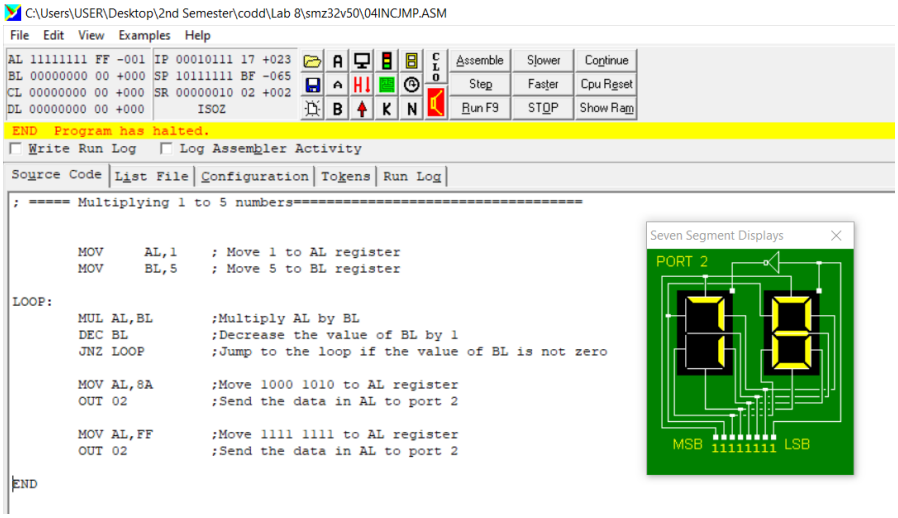


5. Factorial

- Sample code

```
; ===== Multiplying 1 to 5 numbers==================================


        MOV     AL,1    ; Move 1 to AL register
        MOV     BL,5    ; Move 5 to BL register

LOOP:
        MUL AL,BL       ;Multiply AL by BL
        DEC BL          ;Decrease the value of BL by 1
        JNZ LOOP        ;Jump to the loop if the value of BL is not zero

        MOV AL,8A       ;Move 1000 1010 to AL register
        OUT 02          ;Send the data in AL to port 2

        MOV AL,FF       ;Move 1111 1111 to AL register
        OUT 02          ;Send the data in AL to port 2

END
```

- Screenshot of the output



6. Conclusions

In this lab, first we tried a couple of examples given with smz32. Then we modified some of those examples to implement detailed behavior.

ex: - using signal lights & seven segment display

Finally, we developed a new Assembly program to calculate the product of integers from 1 to 5 and showed the hexadecimal result on 7-segment display.