Bangladesh Army University of Engineering & Technology (BAUET)
Qadirabad, Natore-6431

Department of Computer Science and Engineering (CSE)

Lab Report: 03

Course Code: CSE-3222

Course Title: Programming with Assembly language Sessional

Experiment Name: Introduction to arithmetic operations assembly language

Experiment Date: 24.08.25
Submission Date: 07.09.25

SUBMITTED BY

Name: Tanvir Akter Sonia

ID: 0812220105101068

Batch: 16th

Year: 3rd

Semester: 2nd

Section: B

SUBMITTED TO

MD. Fatin Nbbrash Nakib
Lecturer, Dept. of CSE, BAUET

Tanvir Anjom Siddique
Lecturer, Dept. of CSE, BAUET

Experiment No:03

Experiment Name : Introduction to branching structure in assembly language

Objectives:

1. To understand how conditional and unconditional branching work in assembly.
2. To use branching instructions for making program decisions.
3. To build simple problem-solving skills through branching structures.

Theory:

In assembly language, branching plays an important role because it allows a program to change its normal sequence of execution. Normally, instructions are executed line by line, but with branching, the flow can be altered depending on conditions or requirements.

There are mainly two types of branching: unconditional and conditional.

- Unconditional branching transfers control directly to another part of the program without checking any condition. For example, the JMP instruction is used to jump to a new location.
- Conditional branching happens only if a certain condition is true or false. Here, comparison instructions are combined with jumps like JE (jump if equal), JG (jump if greater), or JL (jump if less).

By using these branching techniques, we can implement decision-making in assembly programs. This makes it possible to handle different cases, repeat tasks, or skip certain instructions. In short, branching provides flexibility and control, making programs more logical and efficient.

Branching in assembly allows the program to change its normal flow of execution. It can be unconditional (direct jump) or conditional (jump based on a condition). Using branching, programmers can perform decision-making and control program logic effectively.

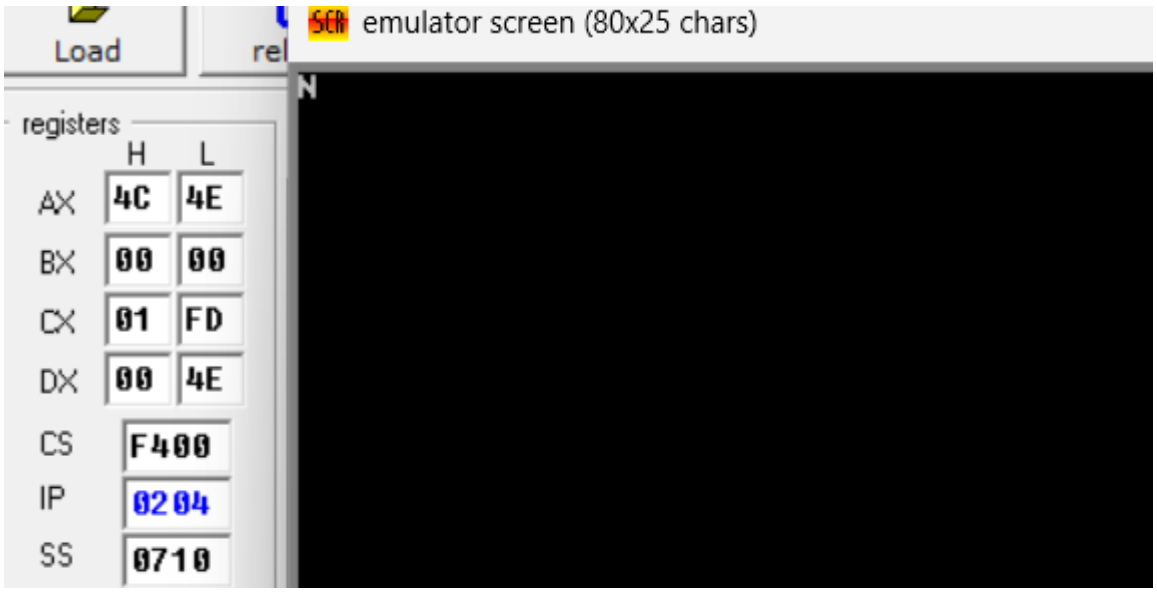
Problem No:3.1

Problem Name : write a assembly language to check Positive or Negative number (with User Input)

Source Code:

```
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN PROC
    MOV CL,-3          ; test value (change it to 0 or negative to test)

    CMP CL,0
    JG POSITIVE       ; if CL > 0
    JL NEGATIVE       ; if CL < 0
    JE ZERO           ; if CL == 0

POSITIVE:             ; Positive case
    MOV DL,'P'
    MOV AH,2
    INT 21H
    JMP EXIT

NEGATIVE:             ; Negative case
    MOV DL,'N'
    MOV AH,2
    INT 21H
    JMP EXIT

ZERO:                 ; Zero case
    MOV DL,'Z'
    MOV AH,2
    INT 21H
    JMP EXIT

EXIT:                 ; Exit program
    MOV AH,4CH
    INT 21H

MAIN ENDP
END MAIN
```

Output:



Problem No:3.2

Problem Name: Read a number if it is 2 or 4 display it otherwise terminate the program

Source Code:

```
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN PROC
    MOV CL,2            ; test value (change it to 2, 4 or others to test)

    CMP CL,2
    JE IS_TWO           ; if CL == 2
    CMP CL,4
    JE IS_FOUR          ; if CL == 4
    JMP EXIT            ; otherwise terminate

IS_TWO:                 ; Case CL == 2
    MOV DL,'2'
    MOV AH,2
    INT 21H
    JMP EXIT

IS_FOUR:                ; Case CL == 4
    MOV DL,'4'
    MOV AH,2
    INT 21H
    JMP EXIT

EXIT:                   ; Exit program
    MOV AH,4CH
    INT 21H

MAIN ENDP
END MAIN
```

Output:

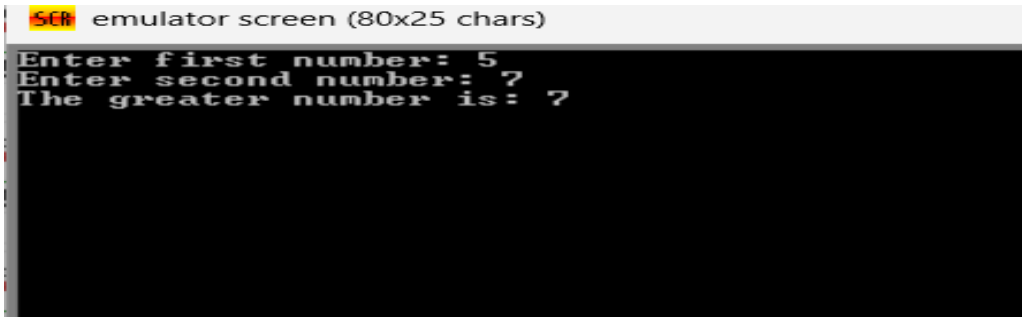
Problem No:3.3

Problem Name: Find the Larger of Two Numbers

Source Code:

```
.MODEL SMALL
.STACK 100H

.DATA
MSG1 DB 'Enter first number: $'
MSG2 DB 0DH,0AH,'Enter second number: $'
MSG3 DB 0DH,0AH,'The greater number is: $'
MSG4 DB 0DH,0AH,'Both numbers are equal.$'

.CODE
MAIN PROC
    ; Initialize DS
    MOV AX, @DATA
    MOV DS, AX

    ; -------- INPUT FIRST NUMBER --------
    LEA DX, MSG1
    MOV AH, 9
    INT 21H

    MOV AH, 1           ; Read char
    INT 21H
    SUB AL, 30H         ; Convert ASCII -> number
    MOV BL, AL          ; Store in BL

    ; -------- INPUT SECOND NUMBER --------
    LEA DX, MSG2
    MOV AH, 9
    INT 21H

    MOV AH, 1
    INT 21H
    SUB AL, 30H
    MOV BH, AL          ; Store in BH

    ; -------- COMPARE --------
    CMP BL, BH
    JG FIRST_GREATER
    JL SECOND_GREATER
    JE EQUAL

FIRST_GREATER:
    LEA DX, MSG3
    MOV AH, 9
    INT 21H
    MOV DL, BL
    ADD DL, 30H         ; Convert back to ASCII
    MOV AH, 2
    INT 21H
    JMP EXIT

SECOND_GREATER:
    LEA DX, MSG3
    MOV AH, 9
    INT 21H
    MOV DL, BH
    ADD DL, 30H
    MOV AH, 2
    INT 21H
    JMP EXIT

EQUAL:
    LEA DX, MSG4
    MOV AH, 9
    INT 21H

EXIT:
    MOV AH, 4CH
    INT 21H
MAIN ENDP
END MAIN
```

Output :



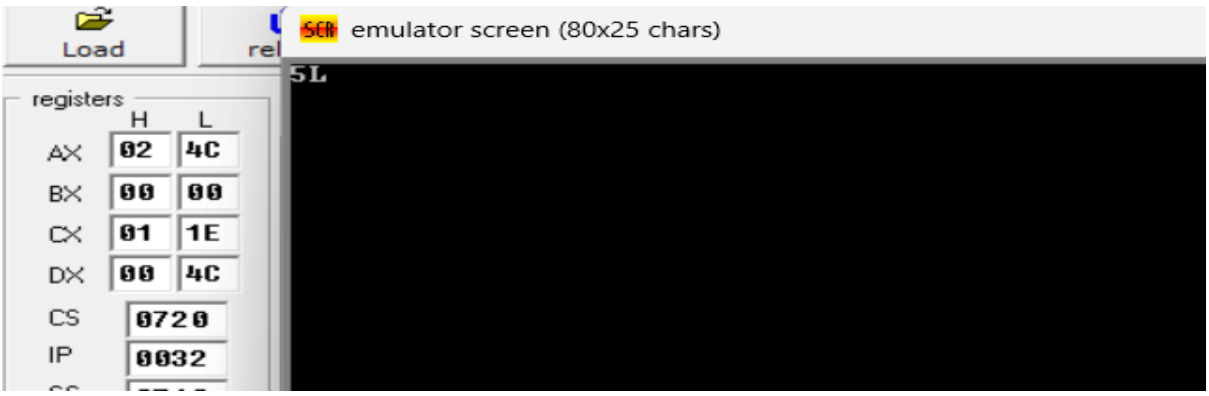
Problem No:3.4

Problem Name: write a assembly language to check upper case and lower case

Source Code:

```
;CHECK UPPER CASE AND LOWER CASE
.MODEL SMALL
.STACK 100H
.DATA
.CODE
MAIN PROC
     MOV AH,1
     INT 21H

     CMP AL,'A'
     JGE CHECK1      ;ASCII OF AL NOT>= ASCII OF A
     JMP LOWER

CHECK1:
     CMP AL,'Z'
     JLE UPPER

     JMP LOWER
LOWER:
     MOV DL,'L'
     MOV AH,2
     INT 21H
     JMP EXIT
UPPER:
     MOV DL,'U'
     MOV AH,2
     INT 21H


EXIT:
MAIN ENDP
END MAIN
```

Output :



Discussion:

In this experiment we can learn how branching structures control the program flow in assembly language. Both conditional and unconditional jumps allow the program to take decisions and change its normal sequence of execution. This technique makes programs more logical, efficient, and suitable for solving real-life problems like comparisons and decision-making.