RIPHAH
INTERNATIONAL UNIVERSITY

COMPUTER ORGANIZAION AND ASSEMBLY LANGUAGE

SIMPLE CALCULATOR

FACULTY OF COMPUTING

I-14 CAMPUS, ISLAMABAD

Names:
ABDUL RAFAY
HASSAN ZAHID

Project Report

Problem Statement

Objective:

Create an Assembly calculator performing basic operations (addition, subtraction, multiplication, division) and two advanced functions (e.g., modulus, exponentiation, square root).

Requirements:

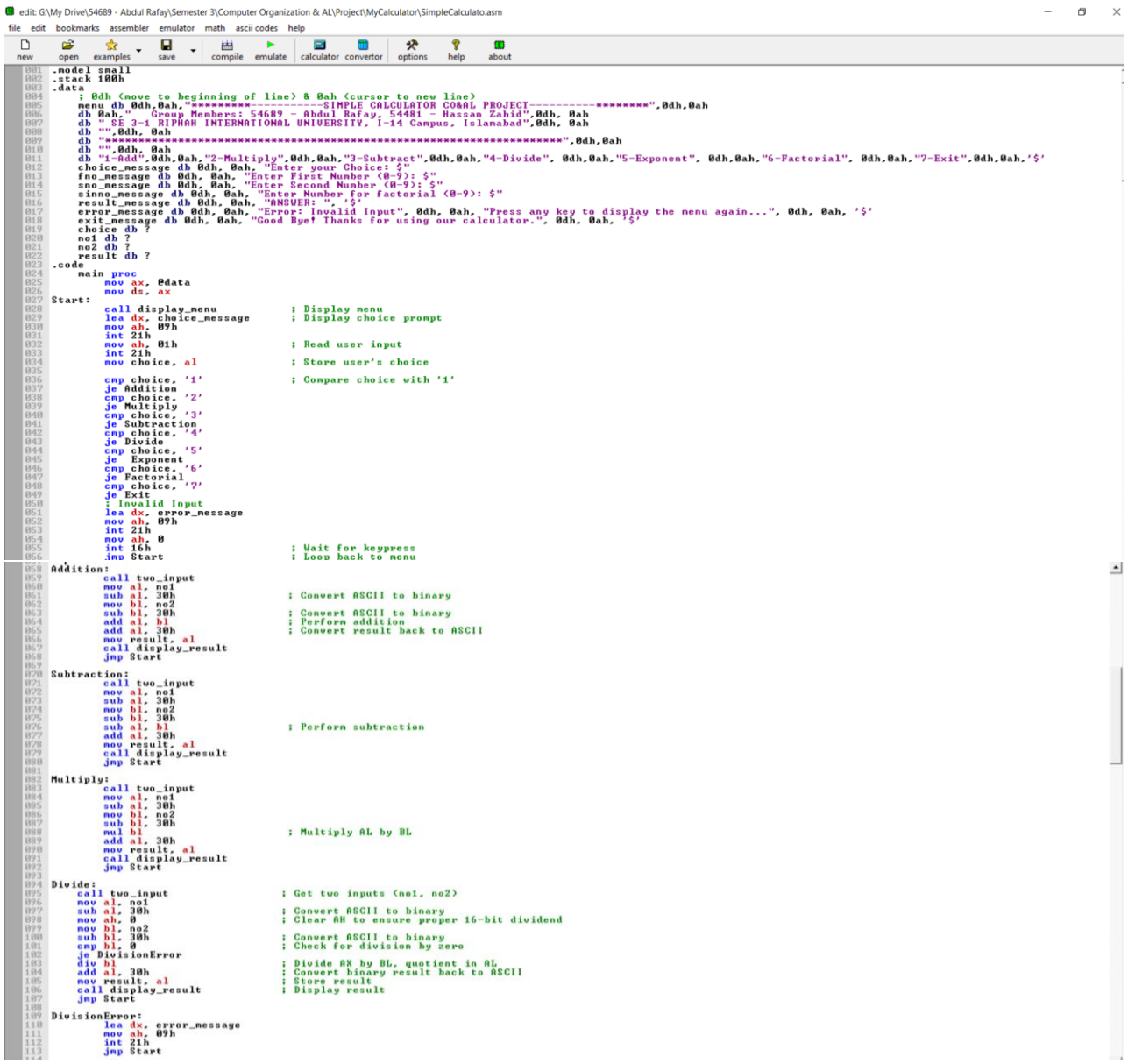
1. Basic Functions: Implement addition, subtraction, multiplication, and division
2. Advanced Functions: Include at least two advanced operations.
3. Assembly Concepts: Utilize the assembly concepts covered in the Lab.
4. User Interface: Prompt for inputs, allow operation selection, and handle errors (e.g., division by zero).

Solution

Software Used:

8086 Emulator

Code:



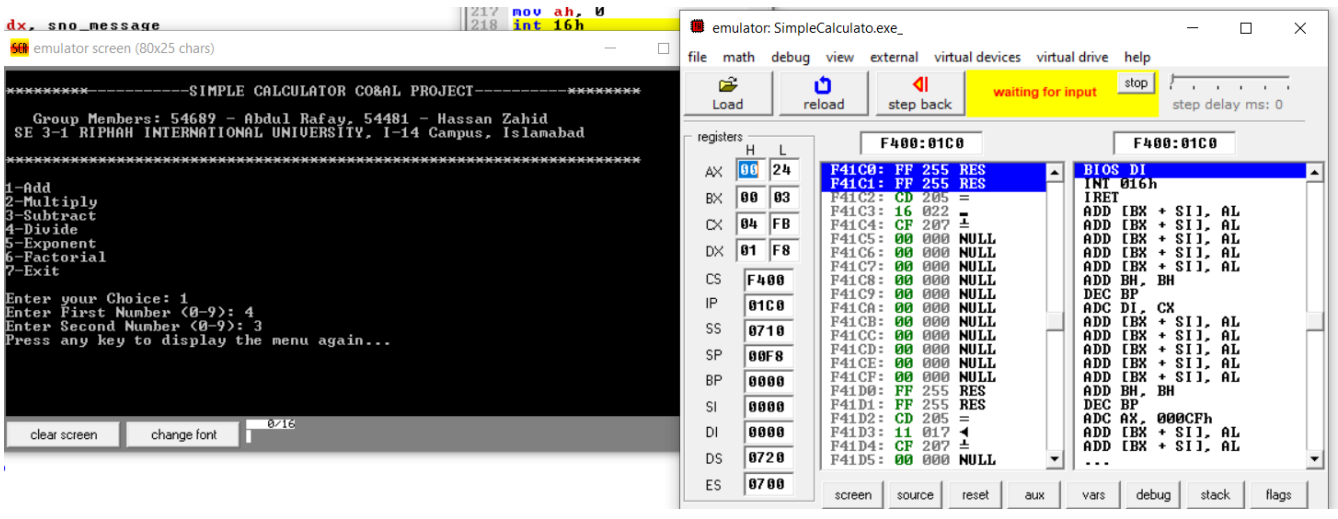

```
.model small
.stack 100h
.data
    ; 0dh (move to beginning of line) & 0ah (cursor to new line)
    menu db 0dh,0ah,"**********----------SIMPLE CALCULATOR CO&AL PROJECT----------**********",0dh,0ah
    db 0ah,"   Group Members: 54689 - Abdul Rafay, 54481 - Hassan Zahid",0dh, 0ah
    db " SE 3-1 RIPHAH INTERNATIONAL UNIVERSITY, I-14 Campus, Islamabad",0dh, 0ah
    db "",0dh, 0ah
    db "**************************************************************",0dh,0ah
    db "",0dh, 0ah
    db "1-Add",0dh,0ah,"2-Multiply",0dh,0ah,"3-Subtract",0dh,0ah,"4-Divide", 0dh,0ah,"5-Exponent", 0dh,0ah,"6-Factorial", 0dh,0ah,"7-Exit",0dh,0ah,'$'
    choice_message db 0dh, 0ah, "Enter your Choice: $"
    fno_message db 0dh, 0ah, "Enter First Number (0-9): $"
    sno_message db 0dh, 0ah, "Enter Second Number (0-9): $"
    sinno_message db 0dh, 0ah, "Enter Number for factorial (0-9): $"
    result_message db 0dh, 0ah, "ANSWER: ", '$'
    error_message db 0dh, 0ah, "Error: Invalid Input", 0dh, 0ah, "Press any key to display the menu again...", 0dh, 0ah, '$'
    exit_message db 0dh, 0ah, "Good Bye! Thanks for using our calculator.", 0dh, 0ah, '$'
    choice db ?
    no1 db ?
    no2 db ?
    result db ?
.code
    main proc
        mov ax, @data
        mov ds, ax
Start:
        call display_menu           ; Display menu
        lea dx, choice_message      ; Display choice prompt
        mov ah, 09h
        int 21h
        mov ah, 01h                 ; Read user input
        int 21h
        mov choice, al              ; Store user's choice

        cmp choice, '1'             ; Compare choice with '1'
        je Addition
        cmp choice, '2'
        je Multiply
        cmp choice, '3'
        je Subtraction
        cmp choice, '4'
        je Divide
        cmp choice, '5'
        je  Exponent
        cmp choice, '6'
        je Factorial
        cmp choice, '7'
        je Exit
        ; Invalid Input
        lea dx, error_message
        mov ah, 09h
        int 21h
        mov ah, 0
        int 16h                     ; Wait for keypress
        jmp Start                   ; Loop back to menu
Addition:
        call two_input
        mov al, no1
        sub al, 30h                 ; Convert ASCII to binary
        mov bl, no2
        sub bl, 30h                 ; Convert ASCII to binary
        add al, bl                  ; Perform addition
        add al, 30h                 ; Convert result back to ASCII
        mov result, al
        call display_result
        jmp Start

Subtraction:
        call two_input
        mov al, no1
        sub al, 30h
        mov bl, no2
        sub bl, 30h
        sub al, bl                  ; Perform subtraction
        add al, 30h
        mov result, al
        call display_result
        jmp Start

Multiply:
        call two_input
        mov al, no1
        sub al, 30h
        mov bl, no2
        sub bl, 30h
        mul bl                      ; Multiply AL by BL
        add al, 30h
        mov result, al
        call display_result
        jmp Start

Divide:
    call two_input                  ; Get two inputs (no1, no2)
    mov al, no1
    sub al, 30h                     ; Convert ASCII to binary
    mov ah, 0                       ; Clear AH to ensure proper 16-bit dividend
    mov bl, no2
    sub bl, 30h                     ; Convert ASCII to binary
    cmp bl, 0                       ; Check for division by zero
    je DivisionError
    div bl                          ; Divide AX by BL, quotient in AL
    add al, 30h                     ; Convert binary result back to ASCII
    mov result, al                  ; Store result
    call display_result             ; Display result
    jmp Start

DivisionError:
        lea dx, error_message
        mov ah, 09h
        int 21h
        jmp Start
```

```
Exponent:
    call two_input                  ; Get base and exponent (no1 and no2)
    mov al, no1
    sub al, 30h                     ; Convert ASCII base to binary
    mov bl, no2
    sub bl, 30h                     ; Convert ASCII exponent to binary
    mov cl, bl                      ; Store exponent in CL for loop counter
    mov bl, al                      ; Store base in BL
    dec cl                          ; Decrement exponent by 1 (base^1 = base)

ExponentLoop:
    cmp cl, 0                       ; Check if exponent loop is complete
    je ExponentDone                 ; If exponent = 0, exit loop
    mul bl                          ; Multiply AL by BL (base)
    dec cl                          ; Decrement loop counter
    jmp ExponentLoop                ; Repeat loop

ExponentDone:
    add al, 30h                     ; Convert result back to ASCII
    mov result, al                  ; Store result
    call display_result             ; Display result
    jmp Start                       ; Return to menu

Factorial:
    call one_input                  ; Get input (ASCII in no1)
    mov al, no1                     ; Move input to AL
    sub al, 30h                     ; Convert ASCII to binary (decimal)
    cmp al, 0                       ; Check if input is 0
    je FactorialIsOne               ; Factorial of 0 is 1
    mov bl, al                      ; Copy input to BL for multiplication
    dec bl                          ; Decrement BL (n-1)
    mov cl, bl                      ; Store BL in CL for loop counter
    mov bl, al                      ; Restore input in BL

FactorialLoop:
    cmp cl, 1                       ; Check if CL <= 1
    jl FactorialDone                ; Exit loop if counter < 1
    mul cl                          ; Multiply AL by CL (AL *= CL)
    dec cl                          ; Decrement counter
    jmp FactorialLoop               ; Repeat loop

FactorialDone:
    add al, 30h                     ; Convert result to ASCII
    mov result, al                  ; Store result in `result`
    call display_result             ; Display result
    jmp Start                       ; Return to menu

FactorialIsOne:
    mov result, '1'                 ; Factorial of 0 is 1
    call display_result             ; Display result
    jmp Start                       ; Return to menu

Exit:
        call goodbye                ; Display goodbye message
        mov ah, 4ch                 ; Terminate program
        int 21h
main endp
```

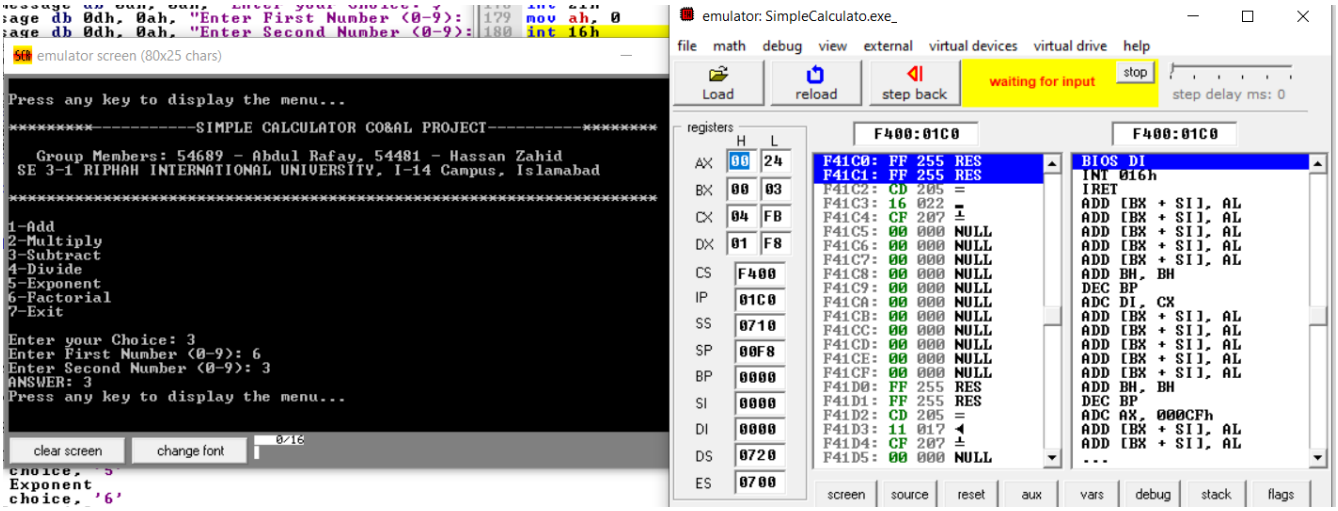
```
; Procedure to get one input
one_input proc
        lea dx, sinno_message
        mov ah, 09h
        int 21h
        mov ah, 01h
        int 21h
        mov no1, al                 ; Saving the number

ret
one_input endp

; Procedure to get two inputs
two_input proc
        lea dx, fno_message
        mov ah, 09h
        int 21h
        mov ah, 01h
        int 21h
        mov no1, al                 ; Save first number

        lea dx, sno_message
        mov ah, 09h
        int 21h
        mov ah, 01h
        int 21h
        mov no2, al                 ; Save second number
        ret
two_input endp

; Procedure to display result
display_result proc
        lea dx, result_message
        mov ah, 09h
        int 21h

        mov dl, result
        mov ah, 02h
        int 21h                     ; Display result
        ret
display_result endp

; Procedure to display goodbye message
goodbye proc
        lea dx, exit_message
        mov ah, 09h
        int 21h
        ret
goodbye endp

end main
```

Implementation details and Results:

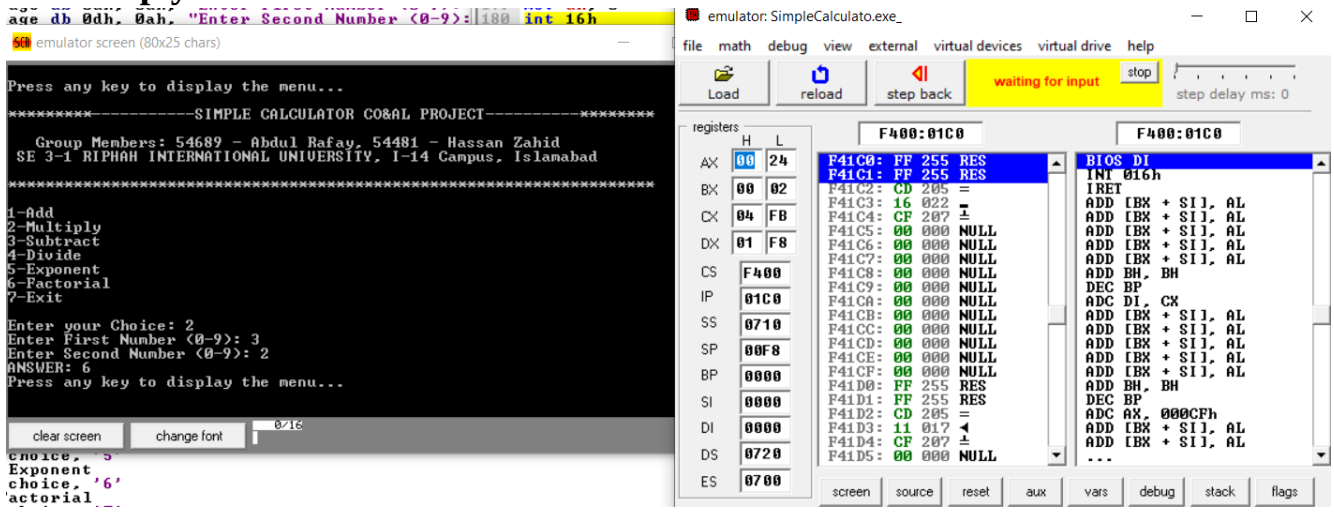
1. Addition:



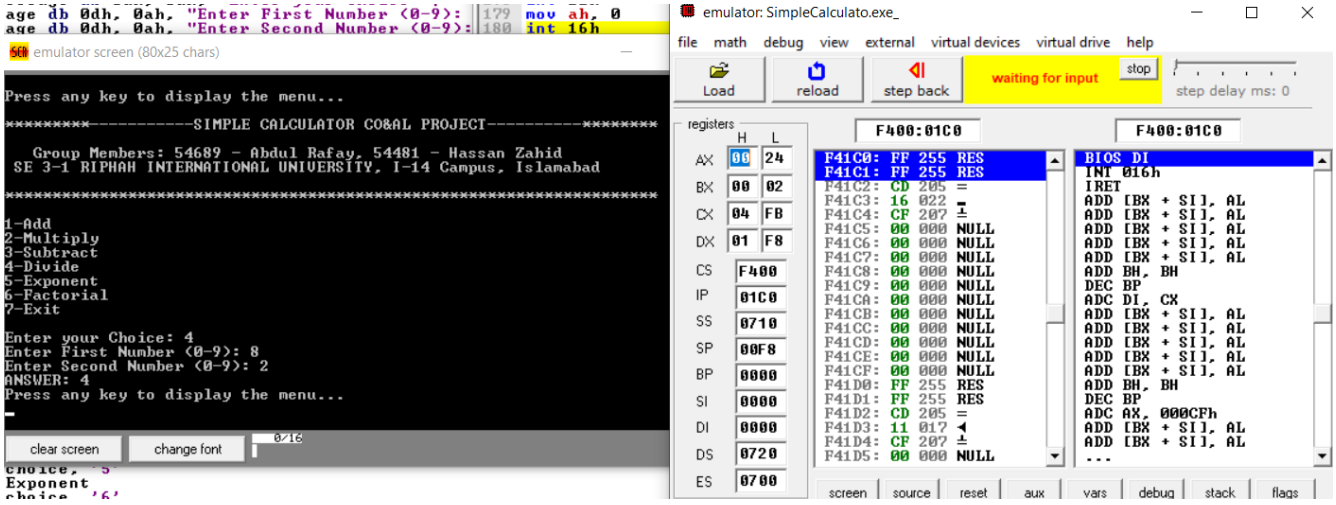
2. Subtraction:



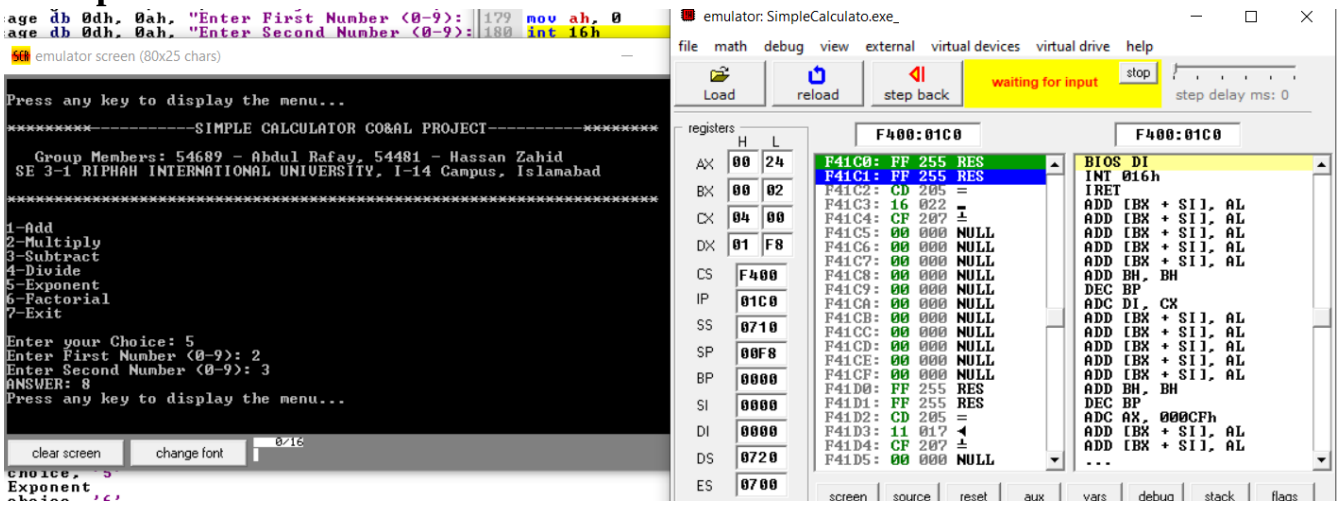
3. Multiply:



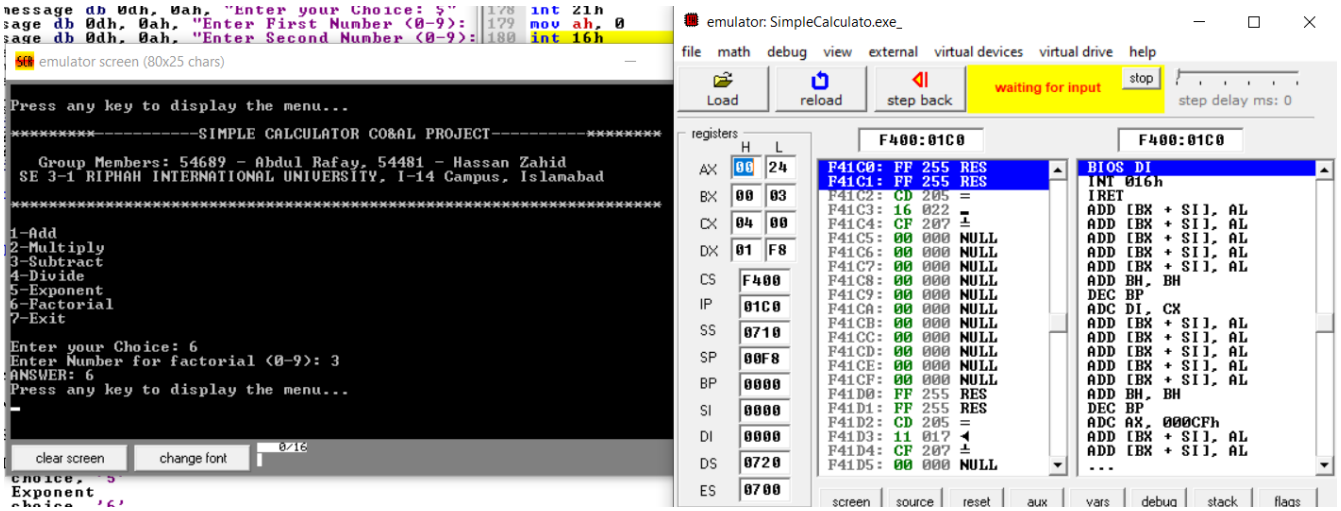
4. Divide:



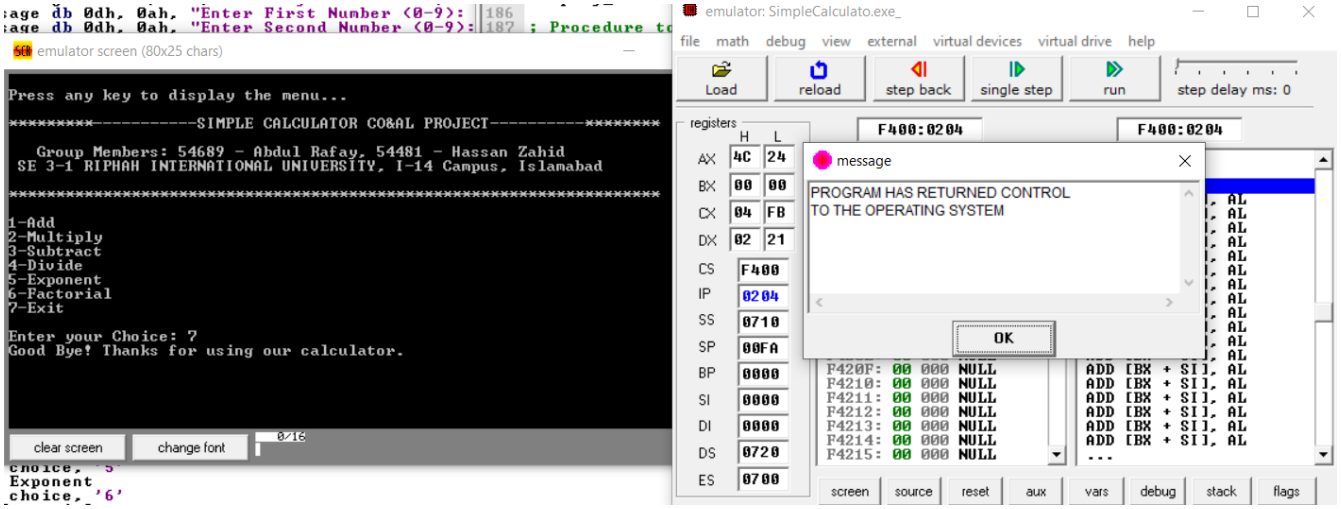
5. Exponent:



6. Factorial:



7. Exit:



Conclusion

After complete execution of code, we were able to successfully compute the basic instruction of calculator like addition, subtraction, multiplication, division in 8086 emulators.

--- ☺The End ☺ ---