FINAL PROJECT REPORT

COURSE TITLE : ***Microprocessors and Microcontrollers Lab***
COURSE CODE : ***CSE 360***

Submitted To :

Md. Ismail
Lecturer,
Department of Computer Science and Engineering,
University of Information Technology & Sciences
(UITS)

Submitted By :

NAME : Tonny Talukder
ID Number : 0432220005101050
Section : 6A1
Batch : 52
Department : CSE

DATE OF SUBMISSION : 26/05/2025

Disk-Based Calculator for Two-Digit Arithmetic Operations using EMU8086

1. Introduction

This project is a **disk-based calculator** developed using **EMU8086 assembly language**. It performs basic arithmetic operations (addition, subtraction, multiplication, and division) on two-digit numbers and saves the result to a disk file.

2. Tools Used : **EMU8086 Microprocessor Emulator**

3. Objectives

- **Design a simple text-based interface for user input.**
- **Implement basic arithmetic operations using assembly language.**

4. Features

- **User-friendly text interface**
- **Two-digit number operations**
- **Four operations: Addition, Subtraction, Multiplication, Division**

5. Code Explanation

The code has the following main sections:

a. Input Section:

- **First, the program displays the operation options (addition, subtraction, multiplication, division).**
- **Then, the program prompts the user to enter two two-digit numbers.**

b. Operation Section:

- **Based on the user's selection, the corresponding arithmetic operation is performed:**
  - **Addition: Sum of two numbers**
  - **Subtraction: Difference between two numbers**
  - **Multiplication: Product of two numbers**
  - **Division: Division of two numbers (with an error message for division by zero)**

CODE :

```
.model small
.stack 100h

.data
   prompt1 db 'Select operation:', 0dh, 0ah
   prompt2 db '1. Addition', 0dh, 0ah
   prompt3 db '2. Subtraction', 0dh, 0ah
   prompt4 db '3. Multiplication', 0dh, 0ah
   prompt5 db '4. Division', 0dh, 0ah
   prompt6 db 'Enter your choice: $'
   prompt7 db 'Enter first number (2 digits): $'
   prompt8 db 'Enter second number (2 digits): $'
   result_msg db 0dh, 0ah, 'Result: $'
   error_msg db 0dh, 0ah, 'Error: Division by zero$'
   choice db 0
   num1 db 0
   num2 db 0
   result dw 0
   sign db '+'
.code
main:
   mov ax, @data
   mov ds, ax

   ; Display options
   mov ah, 09h
   lea dx, prompt1
```

```
int 21h
lea dx, prompt2
int 21h
lea dx, prompt3
int 21h
lea dx, prompt4
int 21h
lea dx, prompt5
int 21h

; Ask for choice
lea dx, prompt6
int 21h
mov ah, 01h
int 21h
sub al, '0'
mov [choice], al

; First number input (2 digits)
lea dx, prompt7
mov ah, 09h
int 21h
mov ah, 01h
int 21h
sub al, '0'
mov bl, al
mov ah, 01h
```

```
int 21h
sub al, '0'
mov bh, al
mov al, bl
mov ah, 0
mov bl, 10
mul bl
add al, bh
mov [num1], al

; Second number input (2 digits)
lea dx, prompt8
mov ah, 09h
int 21h
mov ah, 01h
int 21h
sub al, '0'
mov bl, al
mov ah, 01h
int 21h
sub al, '0'
mov bh, al
mov al, bl
mov ah, 0
mov bl, 10
mul bl
add al, bh
```

```
    mov [num2], al

    ; Determine operation
    mov al, [choice]
    cmp al, 1
    je addition
    cmp al, 2
    je subtraction
    cmp al, 3
    je multiplication
    cmp al, 4
    je division

addition:
    mov al, [num1]
    cbw
    mov bl, [num2]
    add al, bl
    mov ah, 0
    mov [result], ax
    mov byte ptr [sign], '+'
    jmp display_result


subtraction:
    mov al, [num1]
    mov bl, [num2]
```

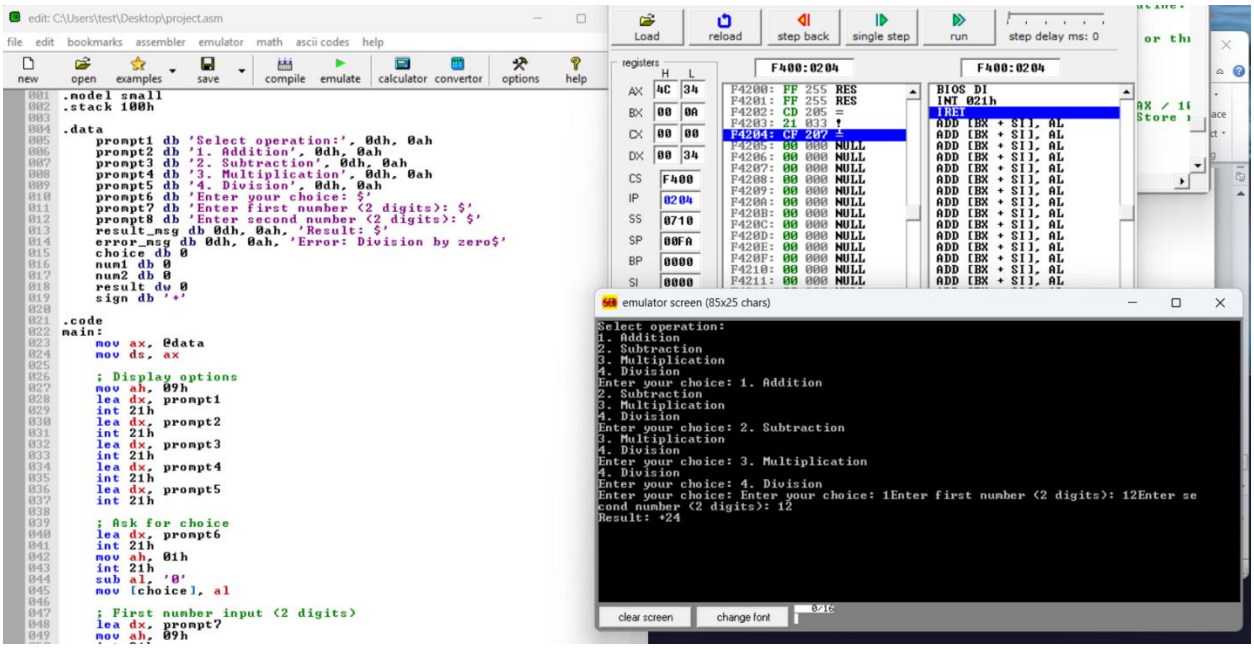
```
    sub al, bl
    cmp al, 0
    jge sub_positive
    neg al
    mov byte ptr [sign], '-'
    jmp sub_store

sub_positive:
    mov byte ptr [sign], '+'

sub_store:
    mov ah, 0
    mov [result], ax
    jmp display_result


multiplication:
    mov al, [num1]
    mov bl, [num2]
    mov ah, 0
    mul bl      ; AX = AL * BL
    mov [result], ax
    mov byte ptr [sign], '+'
    jmp display_result

division:
    mov al, [num1]
```

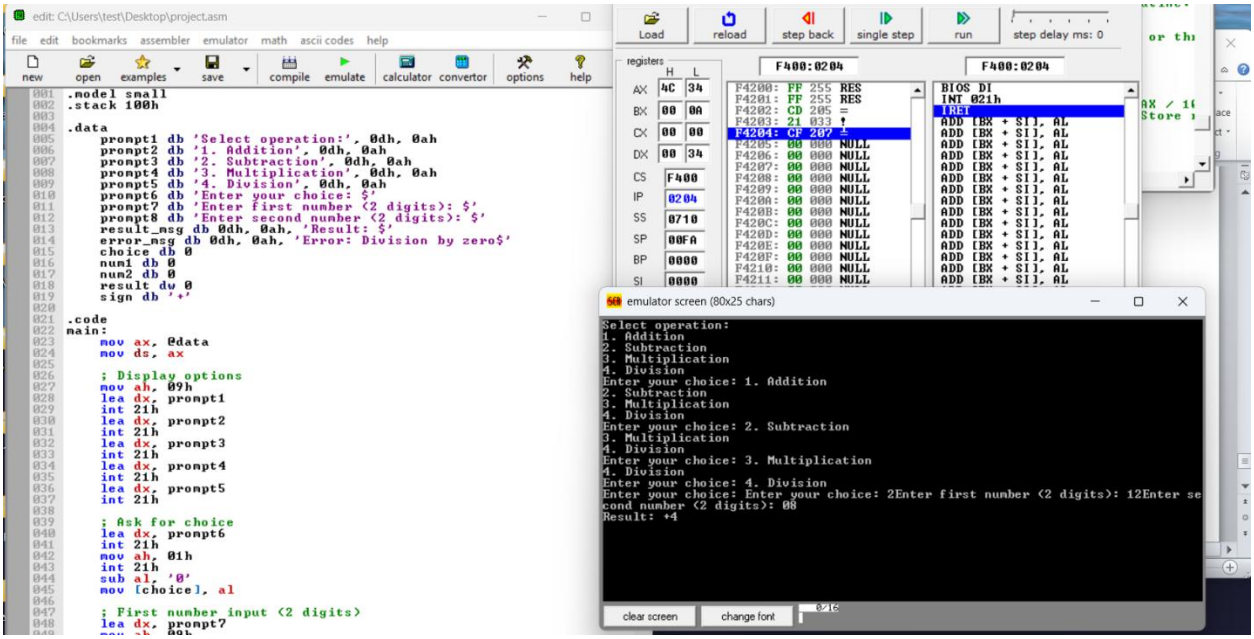
```
    mov ah, 0
    mov bl, [num2]
    cmp bl, 0
    je division_error
    div bl
    mov ah, 0
    mov [result], ax
    mov byte ptr [sign], '+'
    jmp display_result

division_error:
    mov ah, 09h
    lea dx, error_msg
    int 21h
    jmp exit_program

display_result:
    mov ah, 09h
    lea dx, result_msg
    int 21h

    ; Show sign
    mov dl, [sign]
    mov ah, 02h
    int 21h

    ; Show result
```

```
    mov ax, [result]
    call print_number

exit_program:
    mov ah, 4Ch
    int 21h

;---------- Subroutine: print_number ----------
print_number proc
    ; AX = number
    ; print two-digit or three-digit number
    mov cx, 0
    mov bx, 10

next_digit:
    xor dx, dx
    div bx        ; AX / 10 ? quotient in AX, remainder in DX
    push dx       ; Store remainder (digit)
    inc cx
    cmp ax, 0
    jne next_digit

print_loop:
    pop dx
    add dl, '0'
    mov ah, 02h
    int 21h
```

```
    loop print_loop
    ret
print_number endp

end main
```

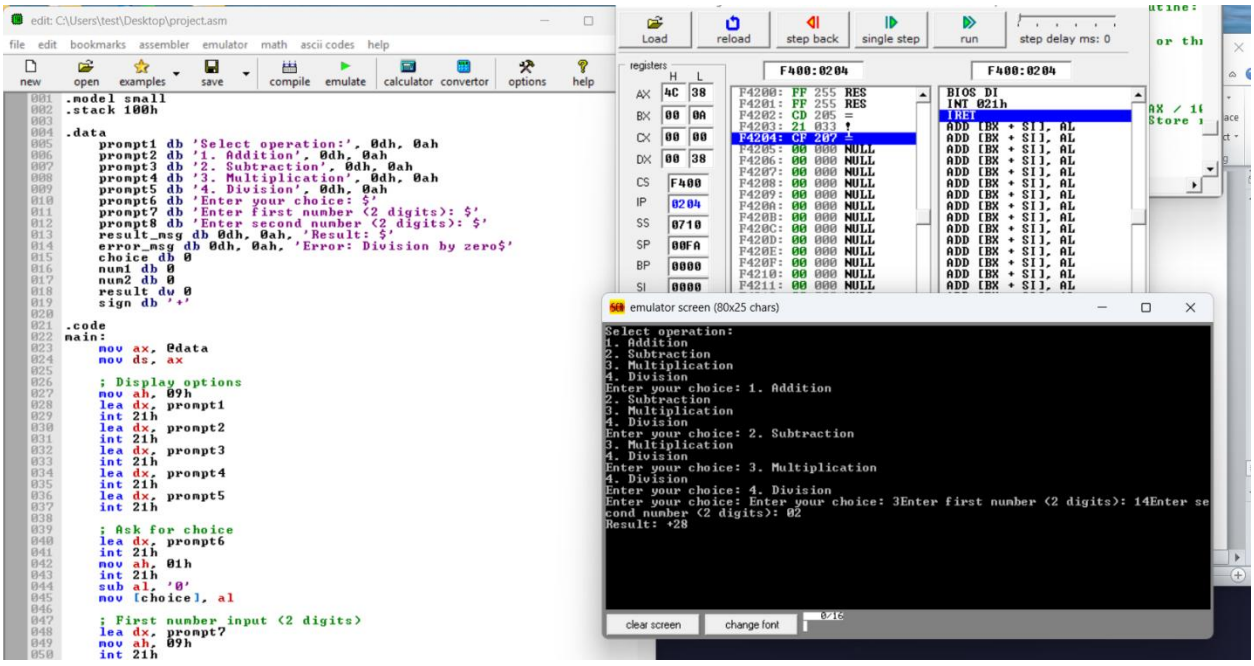
OUTPUT :





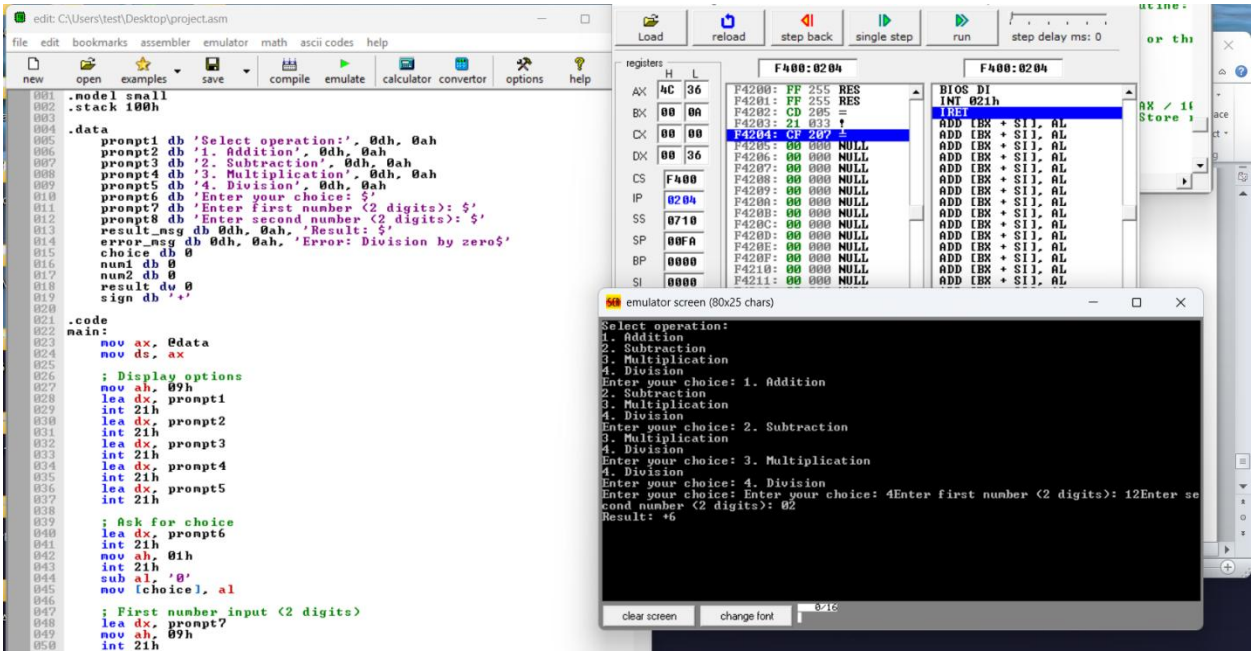
```
.model small
.stack 100h

.data
    prompt1 db 'Select operation:', 0dh, 0ah
    prompt2 db '1. Addition', 0dh, 0ah
    prompt3 db '2. Subtraction', 0dh, 0ah
    prompt4 db '3. Multiplication', 0dh, 0ah
    prompt5 db '4. Division', 0dh, 0ah
    prompt6 db 'Enter your choice: $'
    prompt7 db 'Enter first number (2 digits): $'
    prompt8 db 'Enter second number (2 digits): $'
    result_msg db 0dh, 0ah, 'Result: $'
    error_msg db 0dh, 0ah, 'Error: Division by zero$'
    choice db 0
    num1 db 0
    num2 db 0
    result dw 0
    sign db '+'

.code
main:
    mov ax, @data
    mov ds, ax

    ; Display options
    mov ah, 09h
    lea dx, prompt1
    int 21h
    lea dx, prompt2
    int 21h
    lea dx, prompt3
    int 21h
    lea dx, prompt4
    int 21h
    lea dx, prompt5
    int 21h

    ; Ask for choice
    lea dx, prompt6
    int 21h
    mov ah, 01h
    int 21h
    sub al, '0'
    mov [choice], al

    ; First number input (2 digits)
    lea dx, prompt7
```

```
Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice: 1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice: 2. Subtraction
3. Multiplication
4. Division
Enter your choice: 3. Multiplication
4. Division
Enter your choice: 4. Division
Enter your choice: Enter your choice: 2Enter first number (2 digits): 12Enter se
cond number (2 digits): 08
Result: +4
```



```
.model small
.stack 100h

.data
    prompt1 db 'Select operation:', 0dh, 0ah
    prompt2 db '1. Addition', 0dh, 0ah
    prompt3 db '2. Subtraction', 0dh, 0ah
    prompt4 db '3. Multiplication', 0dh, 0ah
    prompt5 db '4. Division', 0dh, 0ah
    prompt6 db 'Enter your choice: $'
    prompt7 db 'Enter first number (2 digits): $'
    prompt8 db 'Enter second number (2 digits): $'
    result_msg db 0dh, 0ah, 'Result: $'
    error_msg db 0dh, 0ah, 'Error: Division by zero$'
    choice db 0
    num1 db 0
    num2 db 0
    result dw 0
    sign db '+'

.code
main:
    mov ax, @data
    mov ds, ax

    ; Display options
    mov ah, 09h
    lea dx, prompt1
    int 21h
    lea dx, prompt2
    int 21h
    lea dx, prompt3
    int 21h
    lea dx, prompt4
    int 21h
    lea dx, prompt5
    int 21h

    ; Ask for choice
    lea dx, prompt6
    int 21h
    mov ah, 01h
    int 21h
    sub al, '0'
    mov [choice], al

    ; First number input (2 digits)
    lea dx, prompt7
    mov ah, 09h
    int 21h
```

```
Select operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice: 1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice: 2. Subtraction
3. Multiplication
4. Division
Enter your choice: 3. Multiplication
4. Division
Enter your choice: 4. Division
Enter your choice: Enter your choice: 3Enter first number (2 digits): 14Enter se
cond number (2 digits): 02
Result: +28
```



6. Challenges and Solutions

a. Challenges:

- **Handling input and output in assembly language was challenging.**
- **Converting ASCII values to numbers and vice versa had to be done manually.**

b. Solutions:

- **A custom subroutine was written for ASCII-to-number conversion.**
- **Interrupts were used for input/output handling.**

7. Conclusion

This project successfully demonstrates the creation of a disk-based calculator that allows users to perform basic arithmetic operations on two-digit numbers. It serves as a practical example of assembly language programming and disk file handling.