UITS

UNIVERSITY OF INFORMATION TECHNOLOGY AND SCIENCES

Assignment on

Lab Project

Course Title

Microprocessor and MicroControllers

Course Code

CSE 360

Submitted by

KM Jakaria

Section: 6A

Batch: 53

ID: 0432310005101037

Submitted to

Md. Ismail

Lecturer

Department of CSE

UITS

Date of Submission

21 Nov, 2025

Experiment Name: Make a Simple Calculator with basic operations like Addition, Subtraction, Multiplication and Division.

Process:

Using 8086 assembly language, this project creates a basic command-line calculator that can be used with the EMU8086 emulator.

One of the four fundamental arithmetic operations is carried out by the calculator once the user enters two integers:

- Addition
- Subtraction
- Multiplication
- Division

Algorithm:

1. Start the program.
2. Initialize the Data Segment.
3. Display a message asking for the first number.
4. Call the number-input procedure to store the number.
5. Display a message asking for the second number.
6. Call the number-input procedure to store the number.
7. Show the menu of operations:
8. 1 → Addition
9. 2 → Subtraction
10. 3 → Multiplication
11. 4 → Division
12. Take the user's choice as a character input.
13. Compare the input with the valid options.
14. According to the choice:
15. Perform addition → store the result.
16. Perform subtraction → store the result.
17. Perform multiplication → store the result.
18. Perform division → store the result.

19. If the choice is invalid, display “Invalid choice!”
20. Display “Result:” followed by the computed value.
21. End the program.

edit: C:\Users\Famil\Downloads\calculator.asm

file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

.model small
.stack 100h

.data
msg1 db 10,13,'Enter first number: $'
msg2 db 10,13,'Enter second number: $'
msg3 db 10,13,'Choose operation:',10,13,'1. Addition',10,13,'2. Subtraction',10,13,'3. Multiplication',10,13,'4. Division',10,13,'Choice: $'
msg4 db 10,13,'Result: $'
msg5 db 10,13,'Invalid choice!$'
newline db 10,13,'$'

num1 dw ?
num2 dw ?
result dw ?

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

    ; First number input
    mov ah, 9
    lea dx, msg1
    int 21h

    call input_number
    mov num1, bx

    ; Second number input
    mov ah, 9
    lea dx, msg2
    int 21h

    call input_number
    mov num2, bx

    ; Operation choice
    mov ah, 9
    lea dx, msg3
    int 21h

    mov ah, 1
    int 21h

    cmp al, '1'
    je addition
    cmp al, '2'
    je subtraction
    cmp al, '3'
    je multiplication
    cmp al, '4'
    je division

    ; Invalid choice
    mov ah, 9
    lea dx, msg5
    int 21h
    jmp exit

addition:
    mov ax, num1
    add ax, num2
    mov result, ax
    jmp display_result

subtraction:
    mov ax, num1
    sub ax, num2
    mov result, ax
    jmp display_result

multiplication:
    mov ax, num1
    imul num2
    mov result, ax
    jmp display_result

division:
    mov dx, 0
    mov ax, num1
    idiv num2
    mov result, ax
    jmp display_result

display_result:
    mov ah, 9
    lea dx, msg4
    int 21h

    mov ax, result
    call print_number

exit:
    mov ah, 4ch
    int 21h
main endp

; Input number procedure
input_number proc
    mov bx, 0
input_loop:
    mov ah, 1
    int 21h
    cmp al, 13
    je input_done
    sub al, '0'
    mov cl, al
    mov ch, 0
    mov ax, bx
    mov dx, 10
    mul dx
    add ax, cx
    mov bx, ax
    jmp input_loop
input_done:
    ret
input_number endp

; Print number procedure
print_number proc
    mov cx, 0
    mov bx, 10

    cmp ax, 0
    jge positive
    neg ax
    push ax
    mov ah, 2
    mov dl, '-'
    int 21h
    pop ax

positive:
    mov dx, 0
    div bx
    push dx
    inc cx
    cmp ax, 0
    jne positive

print_loop:
    pop dx
    add dl, '0'
    mov ah, 2
    int 21h
    loop print_loop
```

```
print_loop:
    pop dx
    add dl, '0'
    mov ah, 2
    int 21h
    loop print_loop
    ret
print_number endp

end main

ret
```

Implementation:

Source Codes:

```
; You may customize this and other start-up templates;
; The location of this template is c:\emu8086\inc\0_com_template.txt

org 100h

.model small
.stack 100h

.data
msg1 db 10,13,'Enter first number: $'
msg2 db 10,13,'Enter second number: $'
msg3 db 10,13,'Choose operation:',10,13,'1. Addition',10,13,'2. Subtraction',10,13,'3. Multiplication',10,13,'4. Division',10,13,'Choice: $'
msg4 db 10,13,'Result: $'
msg5 db 10,13,'Invalid choice!$'
newline db 10,13,'$'

num1 dw ?
num2 dw ?
result dw ?

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

    ; First number input
    mov ah, 9
    lea dx, msg1
    int 21h

    call input_number
    mov num1, bx

    ; Second number input
    mov ah, 9
    lea dx, msg2
    int 21h

    call input_number
    mov num2, bx

    ; Operation choice
    mov ah, 9
    lea dx, msg3
    int 21h

    mov ah, 1
    int 21h

    cmp al, '1'
    je addition
    cmp al, '2'
    je subtraction
    cmp al, '3'
    je multiplication
    cmp al, '4'
    je division

    ; Invalid choice
    mov ah, 9
    lea dx, msg5
    int 21h
    jmp exit

addition:
    mov ax, num1
    add ax, num2
    mov result, ax
    jmp display_result

subtraction:
    mov ax, num1
    sub ax, num2
    mov result, ax
    jmp display_result

multiplication:
    mov ax, num1
```

```
    imul num2
    mov result, ax
    jmp display_result

division:
    mov dx, 0
    mov ax, num1
    idiv num2
    mov result, ax
    jmp display_result

display_result:
    mov ah, 9
    lea dx, msg4
    int 21h

    mov ax, result
    call print_number

exit:
    mov ah, 4ch
    int 21h
main endp

; Input number procedure
input_number proc
    mov bx, 0
input_loop:
    mov ah, 1
    int 21h
    cmp al, 13
    je input_done
    sub al, '0'
    mov cl, al
    mov ch, 0
    mov ax, bx
    mov dx, 10
    mul dx
    add ax, cx
    mov bx, ax
    jmp input_loop
input_done:
    ret
input_number endp

; Print number procedure
print_number proc
    mov cx, 0
    mov bx, 10

    cmp ax, 0
    jge positive
    neg ax
    push ax
    mov ah, 2
    mov dl, '-'
    int 21h
    pop ax

positive:
    mov dx, 0
    div bx
    push dx
    inc cx
    cmp ax, 0
    jne positive

print_loop:
    pop dx
    add dl, '0'
    mov ah, 2
    int 21h
    loop print_loop
    ret
print_number endp

end main

ret
```

Method:

1. Taking Input

The program asks the user to enter two numbers.
A custom procedure `input_number` reads multi-digit numbers by:

- Taking input one character at time
- Converting from ASCII to integer
- Building the final number using multiplication by 10

The result is stored in BX and then saved in `num1`and `num2`.

2. Operation Selection

The user selects one of the operations by pressing:

1 Addition

2 Subtraction

3 Multiplication

4 Division

The program reads a single character using `int 21h`(AH = 1) and matches it with the options using `cmp` and `je`.

3. Performing the Operation

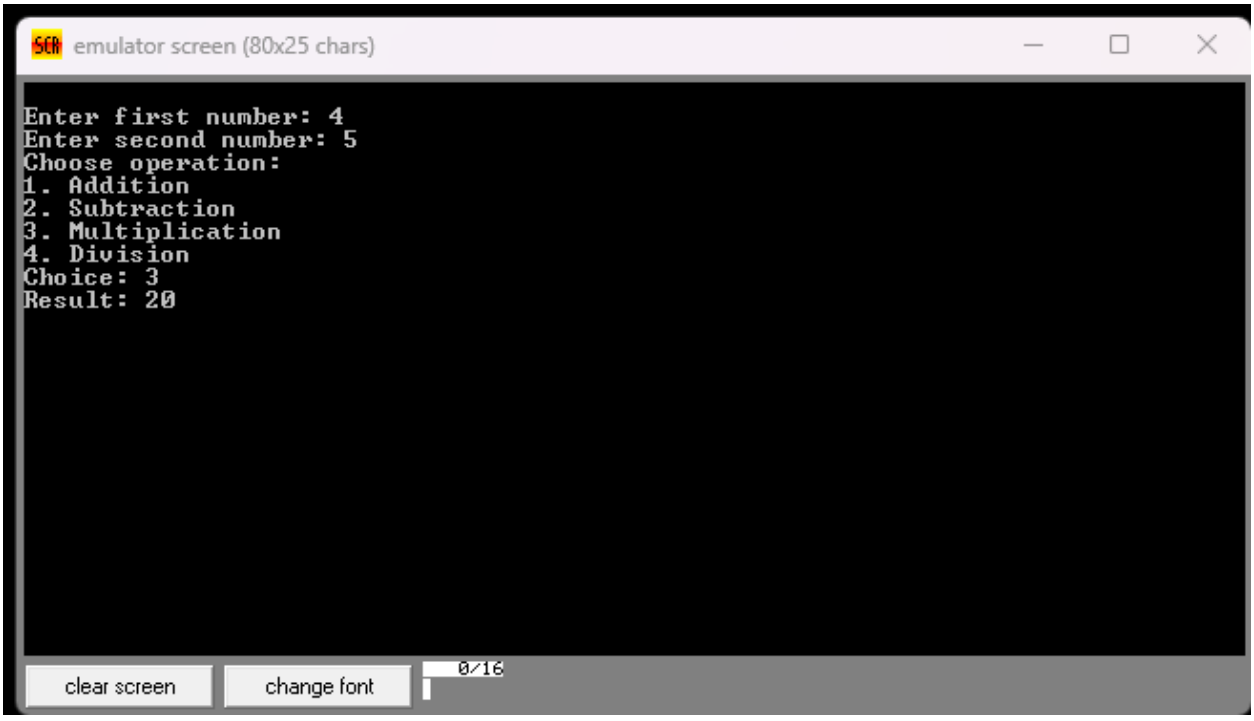
Depending on the choice:

- **Addition:** `ADD AX, num2`
- **Subtraction:** `SUB AX, num2`
- **Multiplication:** `IMUL num2`
- **Division:**
    - Clear DX
    - `IDIV num2`

The result is stored in the variable `result`.

Result:

The application displays as like user friendly calculator



Conclusion:

This project effectively illustrates how simple arithmetic operations can be carried out using assembly language.

This calculator application teaches us:

- Using interruptions to handle user input
- Putting multi-digit numeric input into practice
- Using CPU registers to do arithmetic
- Using bespoke processes to display numbers
- Conditional branching in menu-based systems

The project offers a solid basis for comprehending low-level programming ideas and the direct hardware execution of tasks.