

UITM

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

UITM

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\Family\Downloads\calculator.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator converter options help about

001 ; You may customize this and other start-up templates:
002 ; The location of this template is c:\emu8086\inc\0_com_template.txt
003 org 100h
004 .model small
005 .stack 100h
006 .data
010 msg1 db 10,13,'Enter first number: $',
011 msg2 db 10,13,'Enter second number: $',
012 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: $'
013 msg4 db 10,13,'Result: $'
014 msg5 db 10,13,'Invalid choice!',10,13,'$'
015 newline db 10,13,'$',10,13,'$'
016
017 num1 dw ?
018 num2 dw ?
019 result dw ?
020
021 .code
022 main proc
023     mov ax, @data
024     mov ds, ax
025
026     ; First number input
027     mov ah, 9
028     lea dx, msg1
029     int 21h
030
031     call input_number
032     mov num1, bx
033
034     ; Second number input
035     mov ah, 9
036     lea dx, msg2
037     int 21h
038
039     call input_number
040     mov num2, bx
041
042     ; Operation choice
043     mov ah, 9
044     lea dx, msg3
045     int 21h
046
047     mov ah, 1
048     int 21h
049
050     cmp al, '1'
051     je addition
052     cmp al, '2'
053     je subtraction
054     cmp al, '3'
055     je multiplication
056     cmp al, '4'
057     je division
058
059     ; Invalid choice
060     mov ah, 9
061     lea dx, msg5
062     int 21h
063     jmp exit
064
065     addition:
066     mov ax, num1
067     add ax, num2
068     mov result, ax
069     jmp display_result
070
071     subtraction:
072     mov ax, num1
073     sub ax, num2
074     mov result, ax
075     jmp display_result
076
077     multiplication:
078     mov ax, num1
079     inul num2
080     mov result, ax
081     jmp display_result
082
083     division:
084     mov ax, 0
085     mov ax, num1
086     idiv num2
087     mov result, ax
088     jmp display_result
089
090     display_results:
091     mov ah, 9
092     lea dx, msg4
093     int 21h
094
095     mov ax, result
096     call print_number
097
098     exit:
099     mov ah, 4ch
100     int 21h
101 main endp
102
103     ; Input number procedure
104     input_number proc
105     mov bx, 0
106     input_loop:
107     mov ah, 1
108     int 21h
109     cmp al, 13
110     je input_done
111     sub al, '0'
112     mov cl, al
113     mov cb, 0
114     mov cx, bx
115     mov dx, 10
116     mul dx
117     add cx, dx
118     mov bx, ax
119     jmp input_loop
120
121     input_done:
122     ret
123     input_number endp
124
125     ; Print number procedure
126     print_number proc
127     mov cx, 0
128     mov bx, 10
129
130     cmp ax, 0
131     jne positive
132     neg ax
133     push ax
134     mov dl, ' '
135     int 21h
136     pop ax
137
138     positive:
139     mov ax, 0
140     div bx
141     push dx
142     inc cx
143     cmp cx, 0
144     jne positive
145
146     print_loop:
147     pop dx
148     add dl, '0'
149     mov dl, 2
150     int 21h
151     loop print_loop

```

```

146 print_loop:
147     pop dx
148     add dl, '0'
149     mov ah, 2
150     int 21h
151     loop print_loop
152     ret
153     print_number endp
154
155     end main
156
157     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
    ; Invalid choice
    call input_number
    mov num1, bx
    int 21h
    ; Second number input
    mov ah, 9
    lea dx, msg2
    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           cmp al, 13          mov dx, 0
mov result, ax     je input_done      div bx
jmp display_result sub al, '0'        push dx
                           mov cl, al      inc cx
                           mov ch, 0       cmp ax, 0
division:          mov ax, bx         jne positive
                   mov dx, 10
                   idiv num2      mul dx       print_loop:
                   mov result, ax   add ax, cx      pop dx
                   jmp display_result  mov bx, ax      add dl, '0'
                                         jmp input_loop    mov ah, 2
display_result:    input_done:        int 21h
                   mov ah, 9        ret
                   lea dx, msg4    input_number endp  loop print_loop
                   int 21h          ; Print number procedure
                           mov ax, result  print_number proc  end main
                           call print_number  mov cx, 0
                           mov bx, 10      ret
exit:              mov ah, 4ch      cmp ax, 0
                   int 21h        jge positive
main endp:         neg ax
                   push ax
; Input number procedure      mov ah, 2
input_number proc:  mov dl, '-'      int 21h
                   mov bx, 0
                   input_loop:      pop ax
                   mov ah, 1
                   int 21h        positive:

```

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

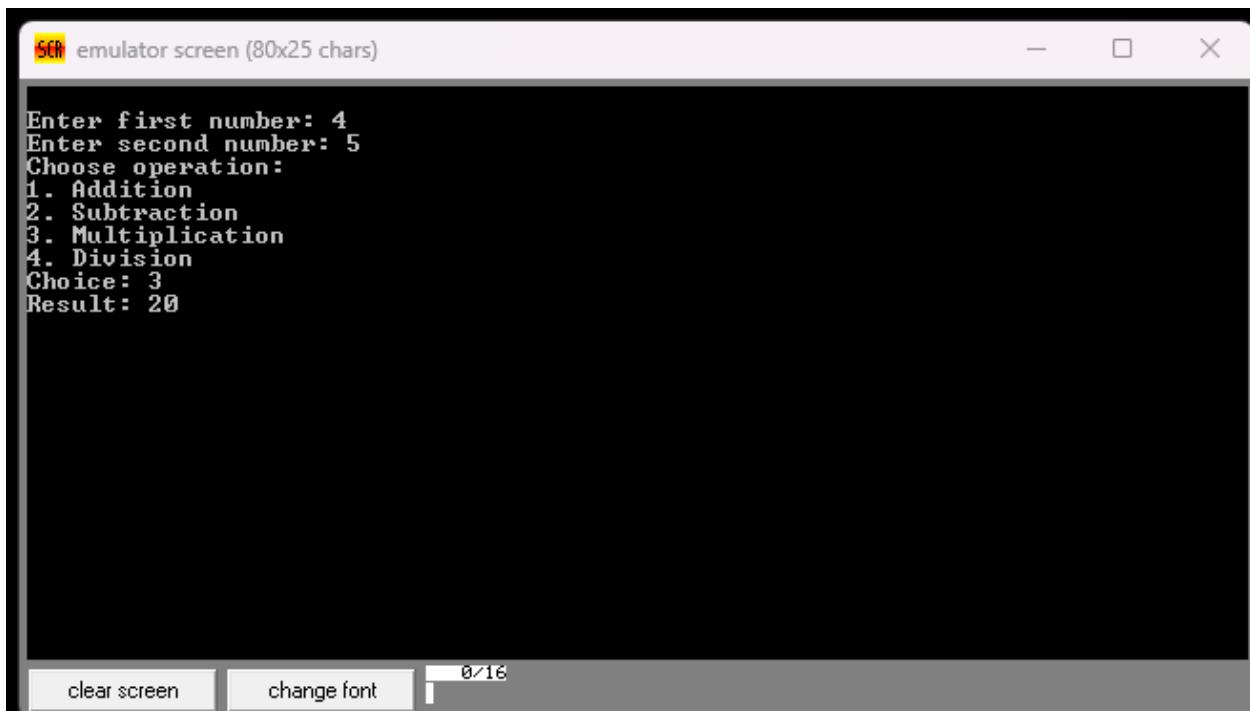
Depending on the choice:

- **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



```
SCR emulator screen (80x25 chars)
Enter first number: 4
Enter second number: 5
Choose operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
Choice: 3
Result: 20
```

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.