University of Information Technology & Sciences

Department of

Computer Science and Engineering

A logo of a university

AI-generated content may be incorrect.

**Disk Calculator Project Report**

Course Title: Microprocessors and Microcontrollers Lab

Course Code: CSE-360

Submitted To

Md. Ismail

Lecturer

Of

CSE Department

Submitted By

Name : Sadia Akter Tuly

Id : 0432220005101030

Batch : 52

Semester: Spring 25

Section : 6A1

**1. Title**

Disk-Based Calculator for Two-Digit Arithmetic Operations using 8086 Microprocessor

**2. Objective**

To design and implement a simple calculator using assembly language for the 8086 microprocessor that performs basic arithmetic operations (addition, subtraction, multiplication, and division) on two-digit decimal numbers, with a textual menu-based interface.

**3. Apparatus Required**

* PC with EMU8086 or similar 8086 simulator
* 8086 Assembly Language
* Optional: 8086 Trainer Kit (if using hardware)

**4. Theory**

The project uses the 8086 assembly language to take user input via the keyboard and display a textual menu interface. Based on the user's choice, it performs arithmetic operations on two-digit inputs and displays the result.

Key instructions used:

1. INT 21H for I/O
2. Arithmetic instructions: ADD, SUB, MUL, DIV
3. Register usage: AX, BX, CX, DX for data manipulation

**5. Algorithm**

Display a menu:

1. Addition
2. Subtraction
3. Multiplication
4. Division

Take user choice.

* Prompt for two-digit inputs.
* Perform the selected operation.
* Display the result.
* Optionally, return to the menu.

**6. Sample Assembly Code (EMU8086)**

|  |
| --- |
| ; 8086 Assembly Calculator for 2-digit numbers  .model small  .stack 100h  .data  menu\_msg db 'Addition => 1', 13, 10  db 'Subtraction => 2', 13, 10  db 'Multiplication => 3', 13, 10  db 'Division => 4', 13, 10  db 'Exit => 5', 13, 10  db 'Enter choice: $'  num1\_msg db 13, 10, 'Enter first number: $'  num2\_msg db 13, 10, 'Enter second number: $'  result\_msg db 13, 10, 'Result = $'  newline db 13, 10, '$'  large\_msg db 13, 10, 'This is a large number which cannot be stored in 2-digit storage!', 13, 10, '$'  invalid\_msg db 13, 10, 'Invalid input and give your choice again', 13, 10, '$'  num1 dw 0  num2 dw 0  result dw 0  choice db 0  is\_negative db 0 ; Flag to indicate negative result  .code  main proc  mov ax, @data  mov ds, ax  menu\_loop:  ; Display menu  mov dx, offset menu\_msg  mov ah, 09h  int 21h  ; Read choice (single digit)  mov ah, 01h  int 21h  sub al, '0'  mov choice, al  ; Print newline after choice  mov dx, offset newline  mov ah, 09h  int 21h  ; Check for exit (5)  cmp choice, 5  je exit\_program  ; Validate choice (must be 1 to 4)  cmp choice, 1  jb invalid\_choice  cmp choice, 4  ja invalid\_choice  ; Get two numbers  call get\_two\_numbers  ; Process choice  cmp choice, 1  je addition  cmp choice, 2  je subtraction  cmp choice, 3  je multiplication  cmp choice, 4  je division  invalid\_choice:  ; Invalid choice  mov dx, offset invalid\_msg  mov ah, 09h  int 21h  jmp menu\_loop  addition:  mov ax, num1  add ax, num2  mov result, ax  call check\_result\_add  jc large\_result  call display\_result  jmp menu\_loop  subtraction:  mov ax, num1  mov bx, num2  cmp ax, bx  jb swap\_numbers  sub ax, bx  mov result, ax  mov is\_negative, 0  jmp display\_sub\_result  swap\_numbers:  mov ax, num2  mov bx, num1  sub ax, bx  mov result, ax  mov is\_negative, 1  display\_sub\_result:  call display\_result  jmp menu\_loop  multiplication:  mov ax, num1  mul num2  mov result, ax  call check\_result\_mul  jc large\_result  call display\_result  jmp menu\_loop  division:  mov ax, num1  xor dx, dx  div num2  mov result, ax  call display\_result  jmp menu\_loop  large\_result:  mov dx, offset large\_msg  mov ah, 09h  int 21h  jmp menu\_loop  exit\_program:  mov ah, 4Ch  int 21h  main endp  ; Procedure to get two 2-digit numbers on separate lines  get\_two\_numbers proc  ; Display prompt for first number  mov dx, offset num1\_msg  mov ah, 09h  int 21h  call read\_number  mov num1, ax  ; Display prompt for second number  mov dx, offset num2\_msg  mov ah, 09h  int 21h  call read\_number  mov num2, ax  ret  get\_two\_numbers endp  ; Procedure to read a 2-digit number  read\_number proc  xor bx, bx  mov cx, 2  read\_loop:  mov ah, 01h  int 21h  cmp al, 13  je end\_read  sub al, '0'  mov ah, 0  mov dx, 10  push ax  mov ax, bx  mul dx  mov bx, ax  pop ax  add bx, ax  loop read\_loop  end\_read:  mov ax, bx  ret  read\_number endp  ; Procedure to check if addition result is within limit (999)  check\_result\_add proc  cmp result, 999  ja set\_carry  clc  ret  set\_carry:  stc  ret  check\_result\_add endp  ; Procedure to check if multiplication result is within limit (9999)  check\_result\_mul proc  cmp result, 9999  ja set\_carry  clc  ret  check\_result\_mul endp  ; Procedure to display result  display\_result proc  mov dx, offset result\_msg  mov ah, 09h  int 21h  ; Check if result is negative (for subtraction)  cmp is\_negative, 1  jne display\_positive  mov dl, '-'  mov ah, 02h  int 21h  display\_positive:  mov ax, result  mov bx, 10  xor cx, cx  convert\_loop:  xor dx, dx  div bx  push dx  inc cx  cmp ax, 0  jne convert\_loop  display\_loop:  pop dx  add dl, '0'  mov ah, 02h  int 21h  loop display\_loop  mov dx, offset newline  mov ah, 09h  int 21h  mov dx, offset newline  mov ah, 09h  int 21h  ret  display\_result endp  end main |

**7. Result**

The calculator successfully performed addition, subtraction, multiplication, and division of two-digit numbers entered via the keyboard, using a simple text-based interface in EMU8086.

A screenshot of a computer

AI-generated content may be incorrect.

**8. Conclusion**

This project demonstrates the power of 8086 assembly in building interactive applications. Even with limited instructions and memory, a disk-based calculator can handle input, perform calculations, and display results using basic interrupts and logic.