University of Information Technology & Sciences

Department of
Computer Science and Engineering

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:

a. Addition
b. Subtraction
c. Multiplication
d. 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
```

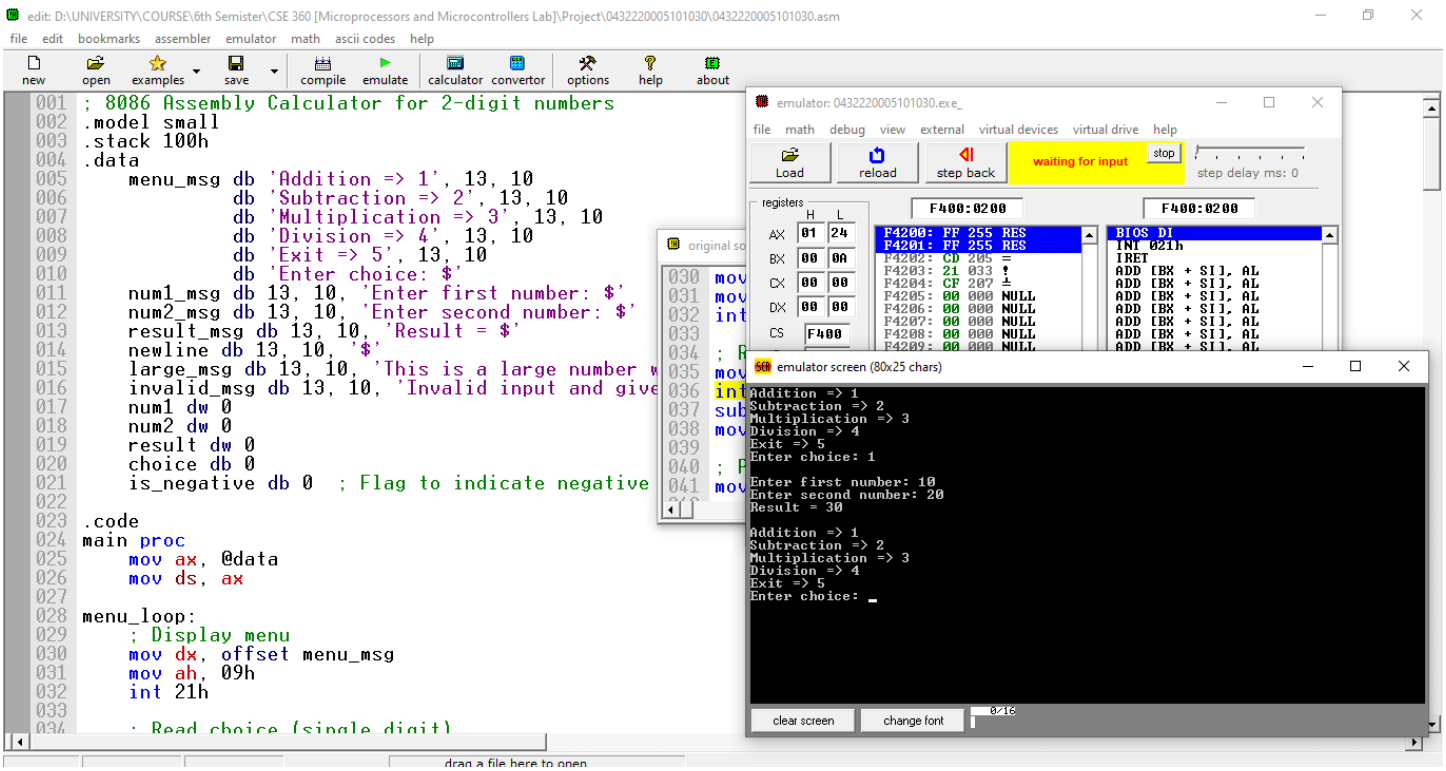
```
display_sub_result:
    call display_result
    jmp menu_loop

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.



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.