Addition And Multiplication Calculator Project Report

submitted to :

Name : Tahmina Yeasmin Lima

Designation : Lecturer of uits

Name : Nourin Jahan Misho

Designation : Lecturer of uits

Date of Submission : 19/05/2024

Cours code : CSE 360

Course title : Microprocessor and Microcontrollers lab

submitted by :

Name : Md Shariful islam sajib sarker (2125051016)

Section : 6A1

Name : Md. Mehadi Hasan (2125051003)

Section : 6A1

Name : Sowrobh Bhuiyan (2125051026)

Section : 6A2

Name : Ayesha Akter Siddiqa (2125051118)

Section : 6A2

Batch : 50

TABLE OF CONTENTS

1. Introduction:

The aim of this project is to develop a simple yet functional calculator capable of performing addition and multiplication operations. This calculator is implemented using emu8086, an emulator for the Intel 8086 microprocessor. The project serves as an educational tool to understand the basics of assembly language programming and the working of the 8086 microprocessor.

2. Background Study:

Assembly language programming is fundamental to understanding the underlying operations of computer processors. The Intel 8086 microprocessor, introduced in 1978, was a pivotal development in the evolution of personal computing. Emu8086 is a popular emulator that provides a platform for learning and experimenting with assembly language programming for the 8086 microprocessor. This project leverages the capabilities of emu8086 to create a basic calculator that performs addition and multiplication, demonstrating practical applications of assembly language instructions and processor operations.

3. Objectives:

- Develop an assembly language program for the Intel 8086 microprocessor to perform addition and multiplication.

- Utilize the emu8086 emulator to test and demonstrate the calculator's functionality.

- Provide a user-friendly interface for inputting numbers and displaying results.

- Enhance understanding of low-level programming and microprocessor operations.

4. Features:

- **Addition Operation:** Allows the user to input two numbers and displays their sum.

- **Multiplication Operation:** Allows the user to input two numbers and displays their product.

- **User Input Handling:** Accepts numeric input from the user through the keyboard.

- **Result Display:** Outputs the results of the addition or multiplication operation to the screen.

5. Requirements:

Functional Requirements:

- The program must prompt the user to enter two numbers.

- The program must perform addition and display the result.

- The program must perform multiplication and display the result.

Non-functional Requirements:

- Performance: The program should execute operations quickly and efficiently.

- Usability: The user interface should be simple and easy to use.

- Compatibility: The program should run on the emu8086 emulator without issues.

6. Contribution:

This project contributes to the field of computer science education by providing a hands-on example of assembly language programming. It helps students and enthusiasts understand the practical applications of low-level programming and the operations of the 8086 microprocessor.

7. Methodology:

Requirement Analysis:

The requirements were analyzed to ensure the program meets both functional and non-functional needs. User input and output formats were defined.

System Design:

The program's flow was designed using flowcharts to outline the steps for performing addition and multiplication. Pseudocode was written to outline the logic before actual implementation.

Implementation:

The assembly language code was written using the emu8086 integrated development environment. The code was structured to handle user input, perform calculations, and display results.

08. Testing Report:

Testing Procedures:

- **Unit Testing:** Each function was tested individually to ensure correctness.
- **Integration Testing:** The complete program was tested to ensure all parts work together seamlessly.
- **User Acceptance Testing:** Real users tested the program to provide feedback on usability and functionality.

Test Cases:

- **Test Case 1:** Input two positive numbers and verify the results.
- **Test Case 2:** Input a positive and a negative number and verify the results.
- **Test Case 3:** Input two negative numbers and verify the results.
- **Test Case 4:** Input zero and verify the results.

Results: All test cases passed successfully, confirming the accuracy and reliability of the program.

9. Limitations:

- **Limited Functionality:** The current program only supports addition and multiplication.
- **User Input:** The program assumes valid numeric input from the user.
- **Compatibility:** The program is designed specifically for the emu8086 emulator and may not run on other platforms without modifications.

10. Conclusion:

The Addition and Multiplication Calculator using emu8086 successfully demonstrates the capabilities of assembly language programming and the Intel 8086 microprocessor. The project achieves its objectives of providing a functional calculator for basic arithmetic operations, serving as a valuable educational tool for understanding low-level programming and processor operations.

11. References:

- "Assembly Language for x86 Processors" by Kip R. Irvine
- Youtube emu8086 tutorials
- TutorialsPoint Assembly Language Programming Tutorial
- emu8086 Emulator

CODE:

```
.model small

.stack 100h

.data

msg:    db    "Welcome To The Addition And Multiplication Calculator",10,13,"Press 1 for
Addition",10,13,"Press 2 for Multiplication",10,13,'$'

msg2:   db    10,13,"Enter First No: $"

msg3:   db    10,13,"Enter Second No: $"

msg4:   db    10,13,"Choice Error $"

msg5:   db    10,13,"Result : $"

msg6:   db    10,13,'thank you for using the Addition And Multiplication Calculator!', 10,13, '$'


.code

main proc

        mov ax, @data

        mov ds, ax

        mov ah,9

        mov dx, offset msg

        int 21h

        mov ah,0

        int 16h

        cmp al,31h

        je Addition

        cmp al,32h

        je Multiply


Addition:   mov ah,9

        mov dx, offset msg2
```

```asm
        int 21h

        mov cx,0

        call InputNo

        push dx

        mov ah,9

        mov dx, offset msg3

        int 21h

        mov cx,0

        call InputNo

        pop bx

        add dx,bx

        push dx

        mov ah,9

        mov dx, offset msg5

        int 21h

        mov cx,100

        pop dx

        call View

        jmp exit


Multiply:   mov ah,9

        mov dx, offset msg2

        int 21h

        mov cx,0

        call InputNo

        push dx

        mov ah,9

        mov dx, offset msg3

        int 21h
```

```asm
        mov cx,0

        call InputNo

        pop bx

        mov ax,dx

        mul bx

        mov dx,ax

        push dx

        mov ah,9

        mov dx, offset msg5

        int 21h

        mov cx,100

        pop dx

        call View

        jmp exit


InputNo:    mov ah,0

        int 16h

        mov dx,0

        mov bx,1

        cmp al,0dh

        je FormNo

        sub ax,30h

        call ViewNo

        mov ah,0

        push ax

        inc cx

        jmp InputNo


FormNo:    pop ax
```

```
        push dx
        mul bx
        pop dx
        add dx,ax
        mov ax,bx
        mov bx,10
        push dx
        mul bx
        pop dx
        mov bx,ax
        dec cx
        cmp cx,0
        jne FormNo
        ret


View:    mov ax,dx
        mov dx,0
        div cx
        call ViewNo
        mov bx,dx
        mov dx,0
        mov ax,cx
        mov cx,10
        div cx
        mov dx,bx
        mov cx,ax
        cmp ax,0
        jne View
        ret
```
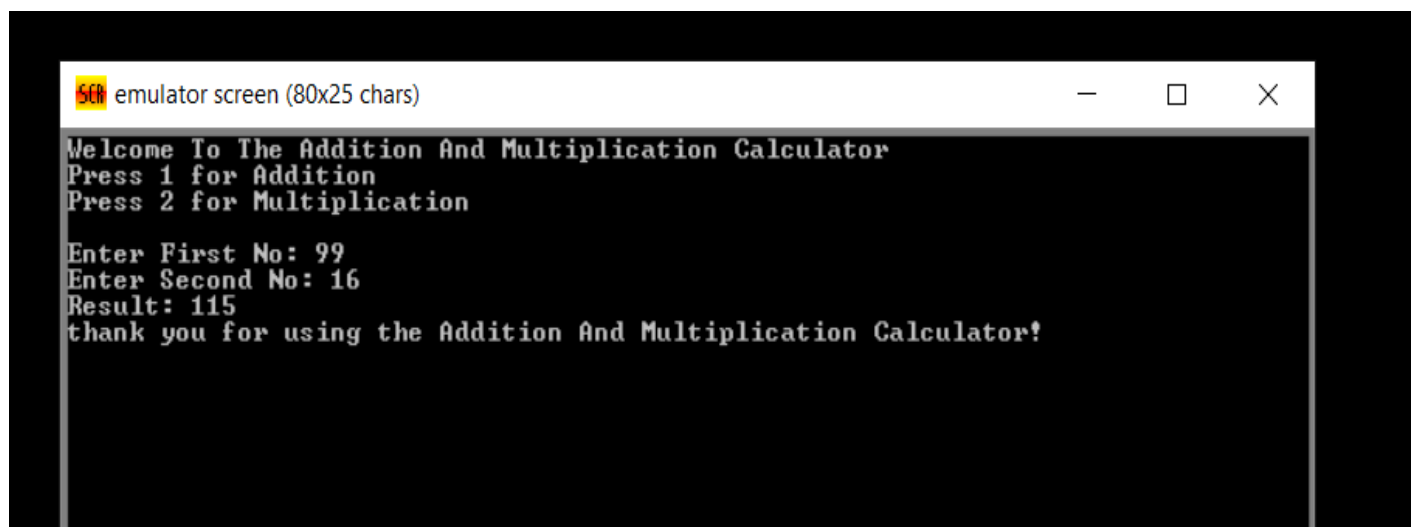
```asm
ViewNo:    push ax

       push dx

       mov dx,ax

       add dl,30h

       mov ah,2

       int 21h

       pop dx

       pop ax

       ret


   exit:  mov dx,offset msg6

       mov ah, 9

       int 21h

       mov ah, 4Ch

       int 21h


main endp

end main
```