COMPUTER ORGANIZATION AND ARCHITECTURE LAB
(15B17CI373)

Project Title: CALCULATOR
Course Name: Computer Organization And Architecture Lab
Batch: F1

Submitted by:

1. Vaishali Ranjan (9920103013)
2. Anya Rathi (9920103001)
3. Khushi Kalra (9920103025)

Submitted to:

Dr. Rashmi Kushwah

Department of CSE
Jaypee Institute Of Information Technology, Noida

Table of Content

Problem Statement

The aim of this project is to design a program that simulates the operation of a calculator system using 8086 simulator which can perform all the basic arithmetic operations like addition, subtraction, multiplication and division.

The user will be asked to enter the numbers on which he/she wants to perform the arithmetic operations. The user will be able to decide what arithmetic operation he/she wants to perform by giving some inputs provided to them. The smart calculator which we designed will print the output on the screen.

Introduction

Assembly language, also known as assembler language, is a low-level programming language that's designed to communicate instructions with specific computer hardware and direct the flow of information. It does this using human-readable mnemonics (consisting of mnemonics like "LDA" to represent load accumulator) to form short code that makes it easier for the person trying to complete the work. These short codes are converted into machine learning language (binary, i.e., 1s and 0s) through the use of programs called assemblers.

A calculator is a device that performs arithmetic operations like addition, subtraction, multiplication, and division.

In this project we have computed the assembly language code (8086) to do the some basic arithmetic operations.

List of Technology/Data Structure Used

1. Hardware Used
   1.1. Processor: Minimum 1 GHz; Recommended 2GHz or more
   1.2. Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)
   1.3. Hard Drive: Minimum 32 GB; Recommended 64 GB or more
   1.4. Memory (RAM): Minimum 1 GB; Recommended 4 GB or above
2. Software Used
   2.1. 8086 Emulator
3. Data Structure Used
   3.1. Data transfer instructions
   3.2. Arithmetic instructions
   3.3. Control transfer instructions

Detailed Design

```
org 100h

jmp start       ; jump over data declaration

msg:    db      0dh,0ah,"CALCULATOR",3dh,0ah,"1-Add",0dh,0ah,"2-Multiply",0dh,0ah,"3-Subtract",0dh,0ah,"4-Divide", 0Dh,0Ah, '$'
msg2:    db      0dh,0ah,"Enter First No : $"
msg3:    db      0dh,0ah,"Enter Second No : $"
msg4:    db      0dh,0ah,"Choice Error $"
msg5:    db      0dh,0ah,"Result : $"
msg6:    db      0dh,0ah,"Thank You for using the calculator! Submitted by Khushi, Anya and Vaishali (F1)",0dh,0ah, '$'
CONTINUE DB ,0dh,0ah,"DO YOU WANT TO CONTINUE",0dh,0ah,'$'
CON DB ?
start:  mov ah,9
        mov dx, offset msg ;first we will display hte first message from which he can choose the operation using int 21h
        int 21h
        mov ah,0
        int 16h  ;then we will use int 16h to read a key press, to know the operation he choosed
        cmp al,31h ;the keypress will be stored in al so, we will comapre to 1 addition ..........
        je Addition
        cmp al,32h
        je Multiply
        cmp al,33h
        je Subtract
        cmp al,34h
        je Divide
        mov ah,09h
        mov dx, offset msg4
        int 21h
        mov ah,0
        int 16h
        jmp start

Addition:   mov ah,09h  ;then let us handle the case of addition operation
            mov dx, offset msg2  ;first we will display this message enter first no also using int 21h
            int 21h
            mov cx,0 ;we will call InputNo to handle our input as we will take each number seprately
            call InputNo  ;first we will move to cx 0 because we will increment on it later in 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,10000
            pop dx
            call View
            ;for continue
            mov ah,09h
            mov dx, offset CONTINUE
            int 21h
            mov ah,01h
            int 21h
            cmp al,'Y'
            je start
            cmp al,'E'
            jmp exit

InputNo:    mov ah,0
            int 16h ;then we will use int 16h to read a key press
            mov dx,0
            mov bx,1
            cmp al,0dh ;the keypress will be stored in al so, we will comapre to  0d which represent the enter key, to know wheter he
            je FormNo ;if it's the enter key then this mean we already have our number stored in the stack, so we will return it back
            sub ax,30h ;we will subtract 30 from the the value of ax to convert the value of key press from ascii to decimal
            call ViewNo ;then call ViewNo to view the key we pressed on the screen
            mov ah,0 ;we will mov 0 to ah before we push ax to the stack bec we only need the value in al
            push ax  ;push the contents of ax to the stack
            inc cx   ;we will add 1 to cx as this represent the counter for the number of digit
            jmp InputNo ;then we will jump back to input number to either take another number or press enter

;we took each number separatly so we need to form our number and store in one bit for example if our number 235
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


ViewNo:     push ax ;we will push ax and dx to the stack because we will change there values while viewing then we will pop them back
            push dx ;the stack we will do these so, we don't affect their contents
            mov dx,ax ;we will mov the value to dx as interrupt 21h expect that the output is stored in it
            add dl,30h ;add 30 to its value to convert it back to ascii
            mov ah,2
            int 21h
            pop dx
            pop ax
            ret


exit:   mov dx,offset msg6
        mov ah, 09h
        int 21h


        mov ah, 0
        int 16h

        ret


Multiply:   mov ah,09h
            mov dx, offset msg2
            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
            mov ax,dx
            mul bx
            mov dx,ax
            push dx
            mov ah,9
            mov dx, offset msg5
            int 21h
            mov cx,10000
            pop dx
            call View
            ;for continue
            mov ah,09h
            mov dx, offset CONTINUE
            int 21h
            mov ah,01h
            int 21h
            cmp al,'Y'
            je start
            cmp al,'E'
            jmp exit

Subtract:   mov ah,09h
            mov dx, offset msg2
            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
            sub bx,dx
            mov dx,bx
            push dx
```

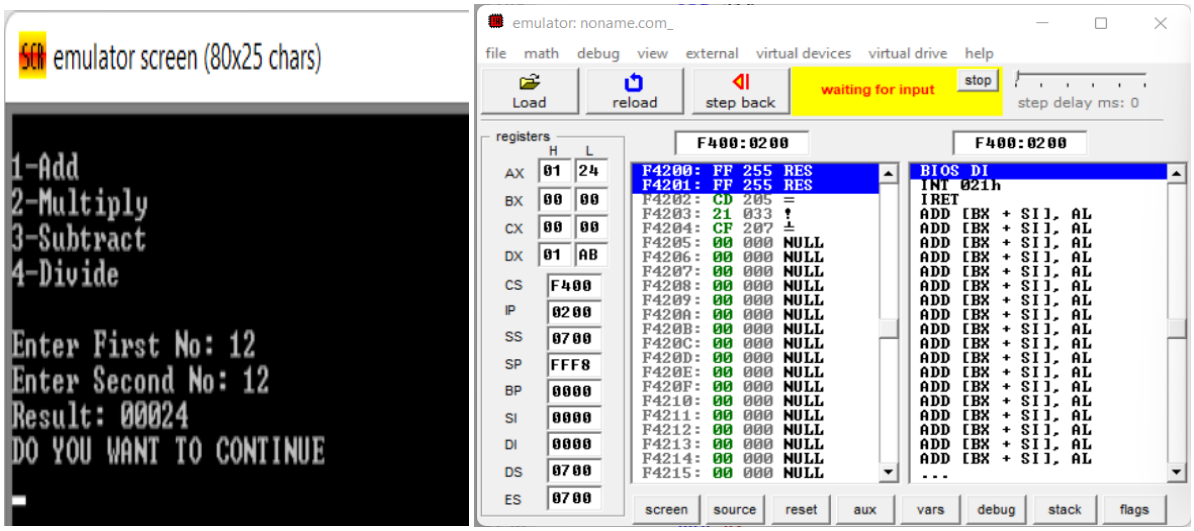
```
            mov ah,9
            mov dx, offset msg5
            int 21h
            mov cx,10000
            pop dx
            call View
            ;for continue
            mov ah,09h
            mov dx, offset CONTINUE
            int 21h
            mov ah,01h
            int 21h
            cmp al,'Y'
            je start
            cmp al,'E'
            jmp exit

Divide:     mov ah,09h
            mov dx, offset msg2
            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
            mov ax,bx
            mov cx,dx
            mov dx,0
            mov bx,0
            div cx
            mov bx,dx
            mov dx,ax
            push bx
            push dx
            mov ah,9
            mov dx, offset msg5
            int 21h
            mov cx,10000
            pop dx
            call View
            pop bx
```

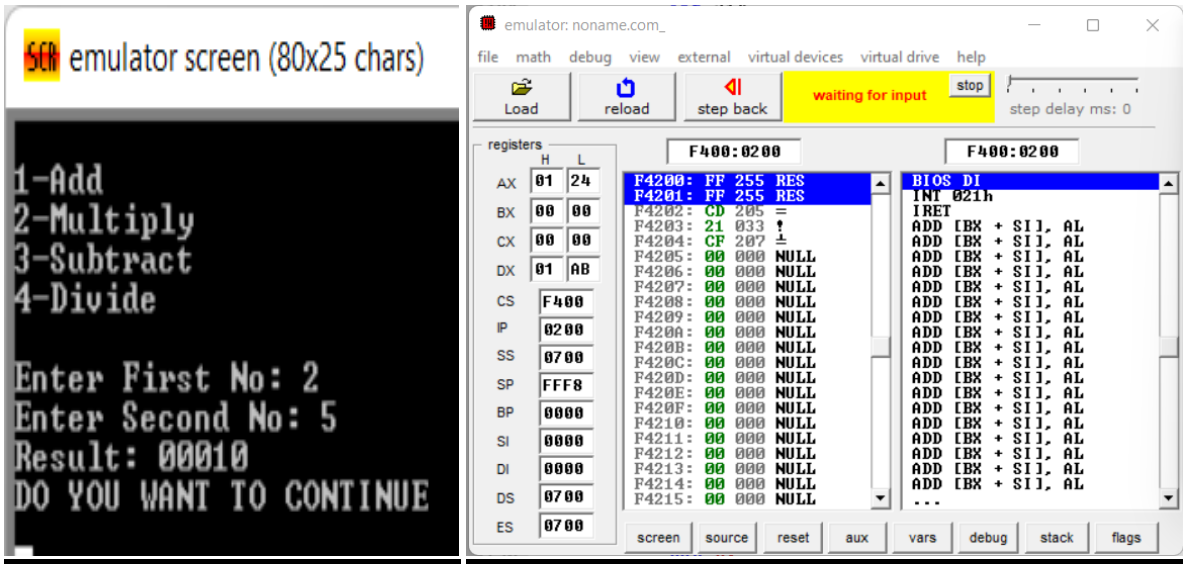
```
            cmp bx,0
            ;for continue
            mov ah,09h
            mov dx, offset CONTINUE
            int 21h
            mov ah,01h
            int 21h
            cmp al,'Y'
            je start
            cmp al,'E'
            je exit
            jmp exit
```

Implementation Details And Results

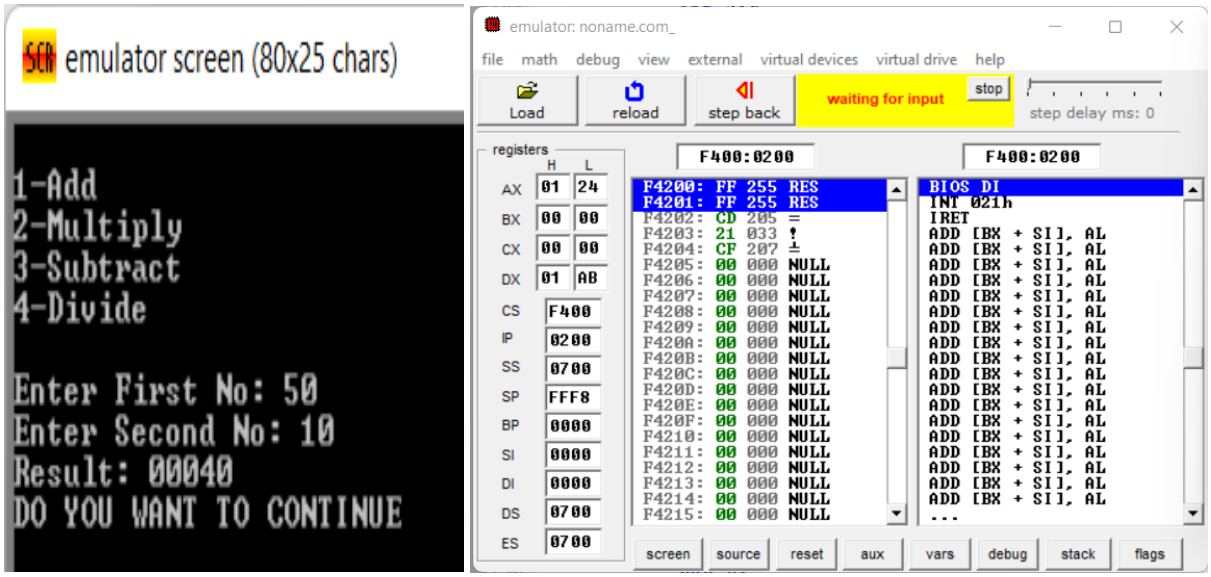
ADDITION:



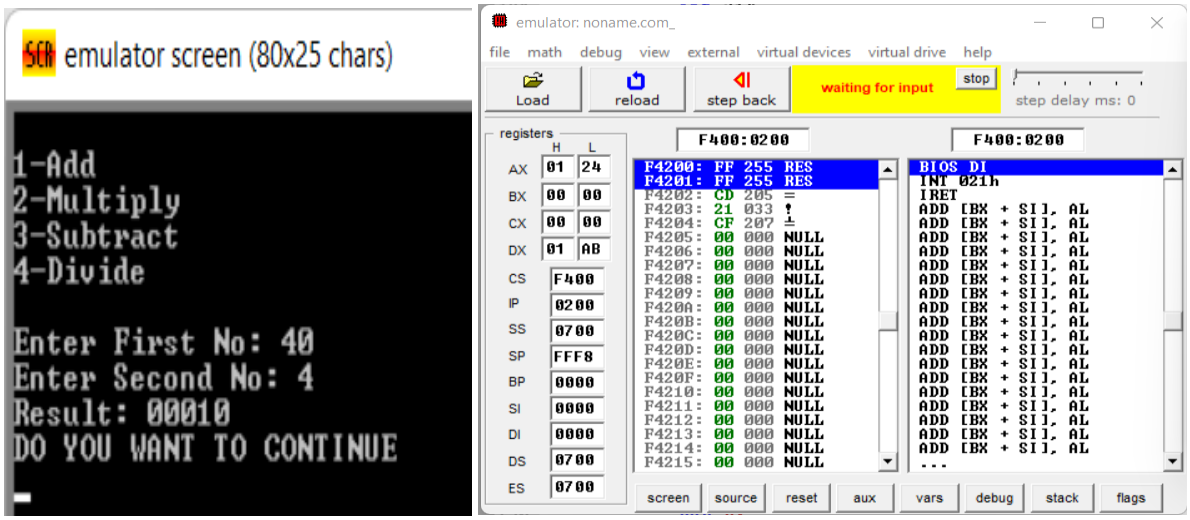
MULTIPLICATION:



SUBTRACTION:



DIVISION:



Conclusion

After complete execution of code we were able to successfully compute the basic instruction of calculator like addition, subtraction, multiplication, division in 8086 emulator.

Assembly language is not a day-to-day language but a computer engineering developer should know the assembly language so that by the piece of code one can understand what is going in the central processing unit and most important thing about assembly language is where a person wants to work at byte-bit level.