MPMC PROJECT: 16-bit calculator using 8086 microprocessor

By:

P VIMAL SUBBIAH (189301026)

VINAYAK SHUKLA (189302136)

SAMRIDH AGGARWAL (189301212)

MANIPAL UNIVERSITY JAIPUR

INSPIRED BY LIFE

Under the guidance of

Mr. Harsh Kishore Mishra

Asst. Professor

Department of Computer Science and Engineering

School of Computing and Information Technology

Manipal University Jaipur

Jaipur, Rajasthan

Problem Statement

To implement a 16-bit decimal calculator for performing basic arithmetic operations between two decimal numbers

Introduction

Mathematics is a building block upon which all of the sciences are dependent. Being able to perform simple arithmetic operations quickly and efficiently is a necessary tool in all scientific fields. Calculators were created in order to give people a simple, fast, and error free method of doing these calculations. I chose to prototype a calculator because they are one of the most basic and important tools for an engineer such as myself. Being able to design and understand the hardware of a calculator is a good starting point from which I can go on to design and understand more complicated devices.

The program is designed to act like a "16–bit Decimal Calculator" with the usual standard functions (addition, subtraction, multiplication, division, modulo, and power). This calculator will have the capability of performing arithmetic operations on 16-bit decimal numbers. It operates in base 10(Decimal) and can accept, displaying, and operating on any numbers in the range 0 to 65535 (16-bit). If answers occur that are not in this range an overflow will occur and the outputted number is not correct.

This calculator works by accepting three inputs from a user: a first number (from 0 to 65535), a second number (from 0 to 65535), and lastly, an operator (+, -, *, /, %, ^).

OPERATORS AVAILABLE ARE:

1. ADDITION (+): Press number 1 as an input from the keyboard

2. MULTIPLICATION (*): Press number 2 as an input from the keyboard

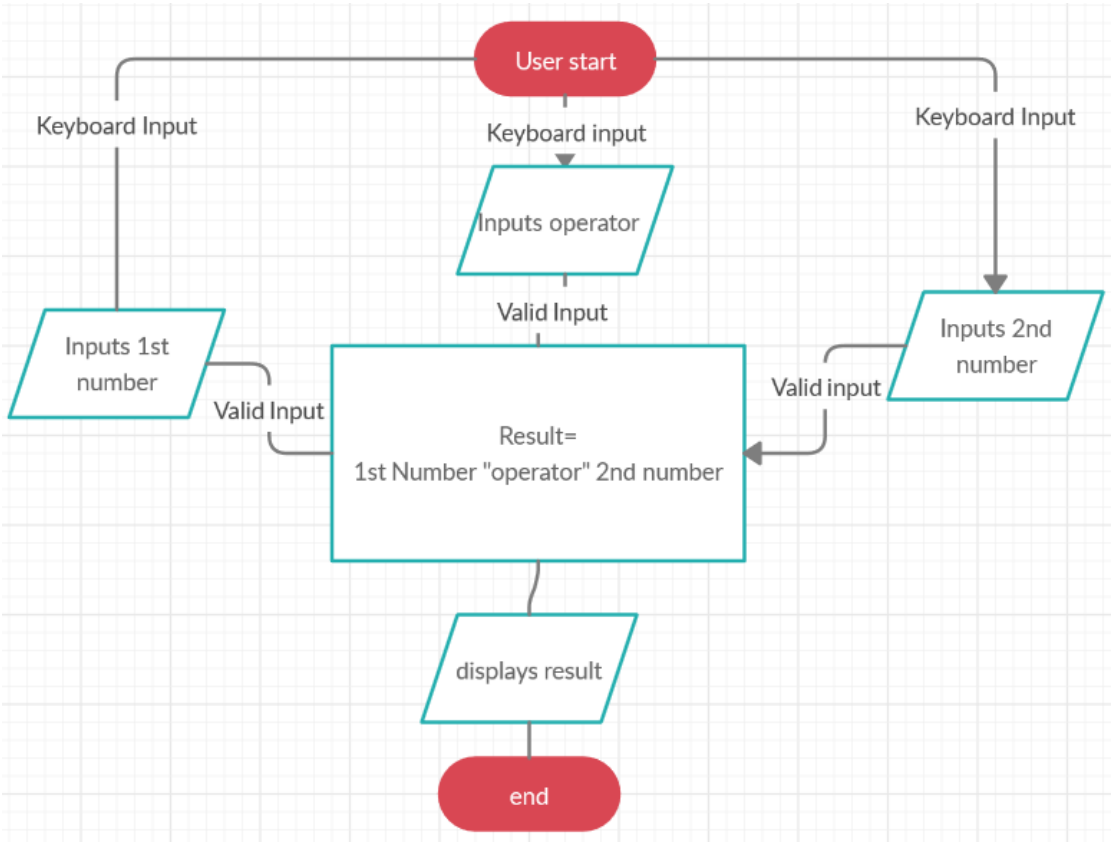
3. SUBTRACTION (-): Press 3 as an input from the keyboard

4. DIVISION (/): Press 4 as an input from the keyboard

5. MODULUS (%): Press 5 as an input from the keyboard

6. POWER (^): Press 6 as an input from the keyboard

Flowchart to describe workflow



- ➔ **Input procedure**
  - Read one operator (user input)
  - Read two numbers (User input)
  - Converts character to digit
  - Stores digits in memory
- ➔ **Fetch and conversion procedure:**
  - Fetch digits from memory
  - Converts digits into number
- ➔ **Operation procedure:**
  - Perform arithmetic operation
  - Stores the result
- ➔ **Conversion and storage procedure:**
  - Converts the resultant number to digits
  - Store the digits for printing result.
- ➔ **Print procedure:**
  - Prints the result on screen.

SOURCE CODE

START:

```
org 100h

jmp start

msg:    db      "1-Add",0dh,0ah,"2-Multiply",0dh,0ah,"3-Subtract",0dh,0ah,"4-Divide", 0Dh,0Ah,"5-Modulus",0dh,0ah,"6-Power",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 : $"
msg6:    db      0dh,0ah ,'Operation Complete, press any key to continue  ', 0Dh,0Ah, '$'

start:  mov ah,9
        mov dx, offset msg
        int 21h
        mov ah,0
        int 16h
        cmp al,31h
        je Addition
        cmp al,32h
        je Multiply
        cmp al,33h
        je Subtract
        cmp al,34h
        je Divide
        cmp al,35h
        je Modulus
        cmp al,36h
        je Power
        mov ah,09h
        mov dx, offset msg4
        int 21h
        mov ah,0
        int 16h
        jmp start
```

INPUT:

```
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:

```
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
            push dx
            mov dx,ax
            add dl,30h
            mov ah,2
            int 21h
            pop dx
            pop ax
            ret
```

ADDITON:

```
Addition:   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
            add dx,bx
            push dx
            mov ah,9
            mov dx, offset msg5
            int 21h
            mov cx,10000
            pop dx
            call View
            jmp exit
```

MULTIPLICATION:

```
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
            jmp exit
```

SUBTRACTION:

```
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
            jmp exit
```

DIVISION

```
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
            je exit
            jmp exit
```

MODULO:

```
Modulus:    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,CX
            MOV DX,00H
            ADD BX,DX
            JZ DbZ
            DIV BX
            MOV AX,DX
           DbZ: RET
            mov ah,9
            mov dx, offset msg5
            int 21h
            mov cx,10000
            pop dx
            call View
            jmp exit
```

POWER

```
Power:      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,CX
            MOV CX,BX
            ADD CX,00h
            JZ Lc
            SUB CX,01h
            JZ La
            JNZ Lb
      La:   ret
      Lb:   MOV BX,AX
            MOV DX,00h
      L1:   MUL BX
            LOOP L1
            ret
      Lc:   MOV AX,01h
            ret
            mov ah,9
            mov dx, offset msg5
            int 21h
            mov cx,10000
            pop dx
            call View
            jmp exit
```

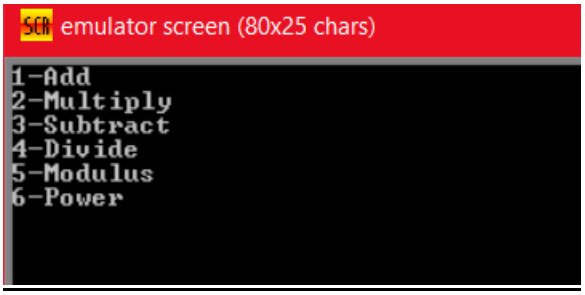
EXIT:

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


        mov ah, 0
        int 16h

        ret
```

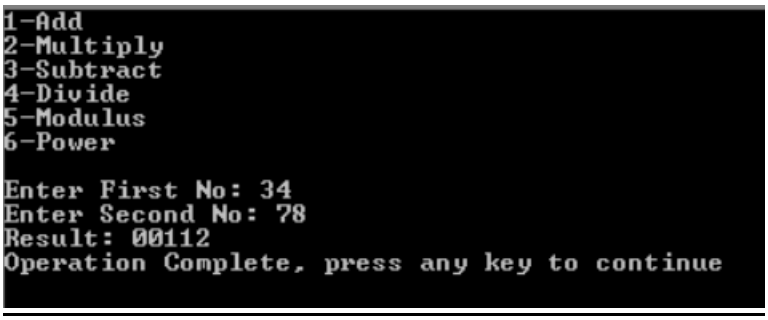
***ALL CODE POSTED IS IN ORDER**

OUTPUT

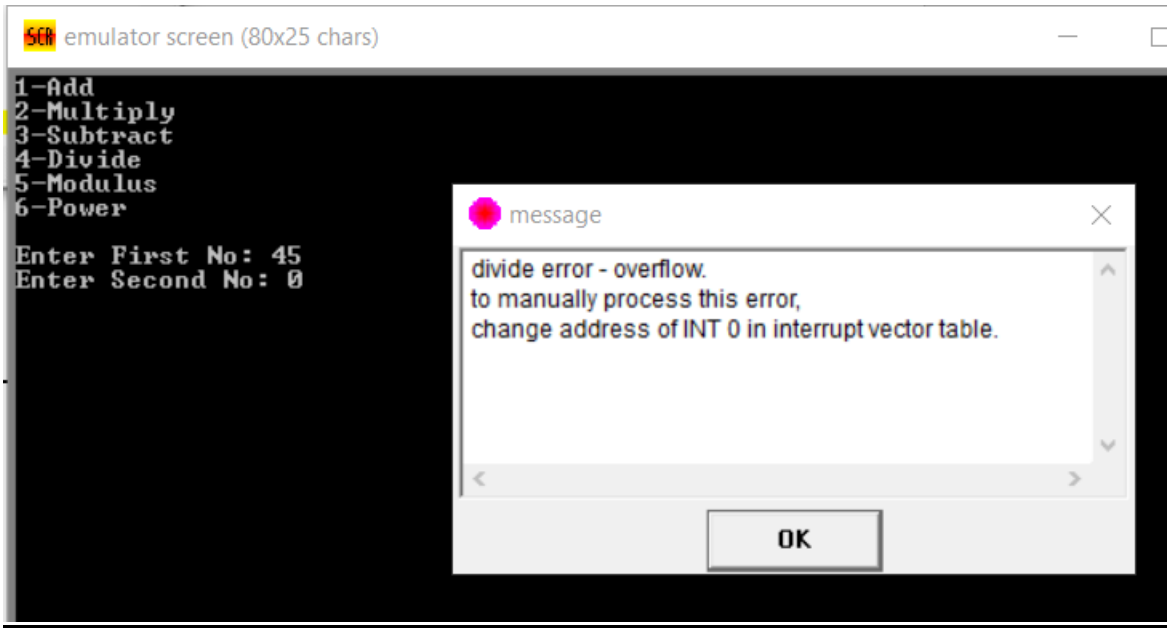
FIRST RUN SCREEN:



SAMPLE OPERATION RESULTS:



SAMPLE ERROR :



Division by zero

If the user requests division by zero, the application will cause the divide error-overflow in the display area and reset number entry.

SOFTWARE

The program is written in assembly language for 8086 microprocessors using EMU8086, EMU8086 is a Microprocessor emulator with integrated 8086 Assembler

CONCLUSION:

The project has been successfully completed. It allows user to perform basic arithmetic operations on 16-bit decimal number (range 0-65535), Moreover we can work on better user interface for the future but that requires the use of a different software though. You can moreover the slow the speed of the process run by adjusting the step delay in emu8086.

MOREOVER YOU CAN GET THE STEP BY STEP COMMAND EXPLANATION AND THE CODE ON MY GITHUB REPOSITORY :

GitHub Repository

Thank you