**Scientific Calculator in 8086 Assembly Language**

by

**Kartik Narang**

**18BCE1199**

A project report submitted to

**Prof. Balamurugan M S**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

in partial fulfilment of the requirements for the course of

**CSE2006 – Microprocessor and Interfacing**

in

**B.Tech. COMPUTER SCIENCE AND ENGINEERING**

****

**VIT CHENNAI**

**Vandalur – Kelambakkam Road**

**Chennai – 600127**

**JUNE 2020**

**BONAFIDE CERTIFICATE**

Certified that this project report entitled “**Scientific Calculator in 8086 Assembly Language”** is a bonafide work of “**Kartik Narang”** who carried out the Project work under my supervision and guidance for **CSE2006-Microprocessor and Interfacing.**

**Prof. Balamurugan M S**

Assistant Professor

School of Electronics Engineering (SENSE),

VIT Chennai

Chennai – 600 127.

**ABSTRACT**

  In this project, I propose an eight bit scientific calculator based Intel 8086 assembly language programming. The calculator is designed over the virtual machine for Intel 8086 microprocessor using EMU8086 emulator software. Several arithmetic, logic operations as well as trigonometric functions are to be implemented in this project.

This Project is based on the idea, that every mathematical operation has basic arithmetic explanation behind it, which can be implemented in limited resources provided in X86 microprocessors.

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof. Balamurugan M S,** Assistant Professor, School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

          We are extremely grateful to “**Jagadeesh Kannan R.**” , Dean of the School of “**SCOPE**”, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

         We express our thanks to our Head of the Department “**Dr. Justus S.”** for his support throughout the course of this project.

        We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

        We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

**……………………………………………………………………..**

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SERIAL NO.** |  | **NAME** | | **PAGE NO.** |
|  |  | ABSTRACT | |  |
|  |  | ACKNOWLEDGEMENT | |  |
|  |  |  | |  |
| 1 |  | INTRODUCTION | |  |
|  |  |  | |  |
|  | 1.1 | OBJECTIVE OF THE PROJECT | |  |
|  | 1.2 | BENEFITS | |  |
|  | 1.3 | FEATURES | |  |
|  |  |  | |  |
| 2 |  | LITERATURE REVIEW  DESIGN | |  |
|  | 2.1 | Block Diagram | |  |
|  |  |  | |  |
|  |  |  | |  |
| 3 | 3.1  3.2 | Code  Code Analysis | |  |
|  |  |  | |  |
| 4 |  | RESULTS AND INFERENCES | |  |
|  | 4.1 | Screenshot | |  |
|  |  |  | |  |
| 5 |  | CONCLUSION AND FUTURE WORK | |  |
|  | |  |  |  |
|  | 5.1  5.2 | Conclusions  Future Work | |  |

6 REFERENCES

1. **INTRODUCTION**

**1.1 OBJECTIVES AND GOALS**

* Design a Scientific Calculator in 8086  ALP
* Display Various operations Possible with Calculator
* Take user input on which operation to perform
* Take desired inputs
* Perform the operation and Display Result.

**1.2 BENEFITS**

A scientific calculator is something we use in our daily life as engineers, for various purposes, academic and practical. Most of the available digital calculators are written in heavy languages as python and java, which make it a bulky app, but this calculator, being written in assembly level language is very light and needs very less memory storage with full functionality.

**1.3 FEATURES**

* Addition
* Subtraction
* Multiplication
* Division
* Modulus
* Power
* Factorial
* Division with Fraction Output upto 2 decimal place
* Permutation (nPr)
* Combination (nCr)
* Area of a Shape
  + 2D and 3D objects(Surface Areas)
* Volume of a Solid
  + 3D objects
* Logarithm

**2 Programming Implementation**

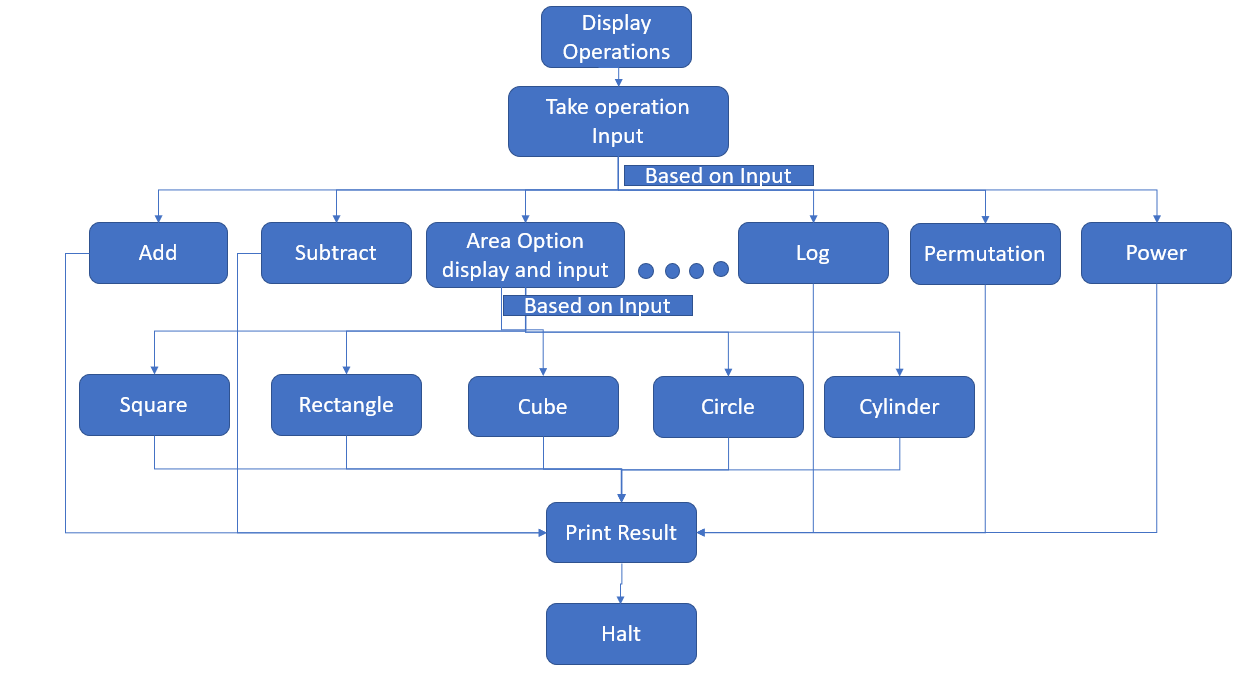
**2.1 BLOCK DIAGRAM**

The four main features of the basic block diagram (given below) are

* Printing available Operations
* Taking input of required operation
* Taking input of required arguments
* Performing Operation
* Displaying Result

**Figure 1. Working FlowChart**

Figure 1 shows the process the basic mechanism on how the program execution proceeds.



1. **SOFTWARE IMPLEMENTATION**

**3.1 Code**

data segment

    a8 db 0

    b8 db 0

    a16 dw 0

    b16 dw 0

    c16 dw 0

    d16 dw 0

    c db 0

    d db 0

    e db 0

    f db 0

    in1 db "Enter First Number: $"

    in2 db "Enter Second Number: $"

    str0 db ".$"

    strx db "All answers are floored, except for number 8$"

    str db "Enter the Serial Number Corresponding to desired Operation$"

    str1 db "1. Addition$"

    str2 db "2. Subtraction$"

    str3 db "3. Multiplication$"

    str4 db "4. Division$"

    str5 db "5. Modulus(remainder)$"

    str6 db "6. Power$"

    str7 db "7. Factorial$"

    str8 db "8. Division with Fraction Answer$"

    str9 db "9. Permutation(nPr)$"

    str10 db "10. Combination(nCr)$"

    str11 db "11. log x base y$"

    str12 db "12. Areas$"

    str13 db "13. Volumes$"

    str121 db "1. Square$"

    str122 db "2. Circle$"

    str123 db "3. Rectangle$"

    str124 db "4. Right triangle$"

    str125 db "5. Surface Area of Cube$"

    str126 db "6. Surface Area of Cuboid$"

    str127 db "7. Curvered Surface Area of Cylinder$"

    str128 db "8. Surface Area of Sphere$"

    str129 db "9. Curved Surface area of Cone$"

    str131 db "1. Cube$"

    str132 db "2. Cuboid$"

    str133 db "3. Sphere$"

    str134 db "4. Cone$"

    str135 db "5. Cylinder$"

data ends

code segment

assume cd:code, ds:data

start:

fractional\_div macro a,b

    pusha

    mov al, a

    div b

    mov bh,al

    mov al,ah

    mov dl,0ah

    mul dl

    div b

    mov c,ah

    mul dl

    mov cl,al

    mov al,c

    mul dl

    xor dx,dx

    div b

    add cl,al

    mov c,bh

    mov d,cl

    popa

fractional\_div endm

    mov ax,data

    mov ds,ax

    lea dx,str

    call print\_str

    lea dx,strx

    call print\_str

    lea dx,str1

    call print\_str

    lea dx,str2

    call print\_str

    lea dx,str3

    call print\_str

    lea dx,str4

    call print\_str

    lea dx,str5

    call print\_str

    lea dx,str6

    call print\_str

    lea dx,str7

    call print\_str

    lea dx,str8

    call print\_str

    lea dx,str9

    call print\_str

    lea dx,str10

    call print\_str

    lea dx,str11

    call print\_str

    lea dx,str12

    call print\_str

    lea dx,str13

    call print\_str

    mov al,0

    mov c,0

    next4:

        mov ah,01h

        int 21h

        cmp al,0dh

        jz next5

        mov d,al

        sub d,30h

        mov al,c

        mov dh,0ah

        mul dh

        add d,al

        mov cl,d

        mov c,cl

        jmp next4

    next5:

    mov ah,0

    mov al,c

    cmp al,1

    jz add\_num

    cmp al,2

    jz sub\_num

    cmp al,3

    jz mul\_num

    cmp al,4

    jz div\_num

    cmp al,5

    jz modulo

    cmp al,6

    jz power

    cmp al,7

    jz fact

    cmp al,8

    jz frac\_div

    cmp al,9

    jz npr

    cmp al,10

    jz ncr

    cmp al,11

    jz log

    cmp al,12

    jz area

    cmp al,13

    jz volume

    cmp al,0ah

    jz area\_master

volume:

    lea dx,str131

    call print\_str

    lea dx,str132

    call print\_str

    lea dx,str133

    call print\_str

    lea dx,str134

    call print\_str

    lea dx,str135

    call print\_str

    mov al,0

    mov c,0

    next8:

        mov ah,01h

        int 21h

        cmp al,0dh

        jz next9

        mov d,al

        sub d,30h

        mov al,c

        mov dh,0ah

        mul dh

        add d,al

        mov cl,d

        mov c,cl

        jmp next8

    next9:

    mov ah,0

    mov al,c

    cmp al,1

    jz cube

    cmp al,2

    jz cuboid

    cmp al,3

    jz sphere

    cmp al,4

    jz cone

    cmp al,5

    jz cylinder

cube:

    call one\_input

    mul al

    mul al

    call print\_al

    jmp done

cuboid:

    call one\_input

    mov a16,ax

    call two\_input

    mul bx

    mov cx,a16

    mul cx

    call print\_ax

    jmp done

sphere:

    call one\_input

    mov cx,ax

    mul cx

    mul cx

    mov cx,58h

    mul cx

    xor dx,dx

    mov cx,15h

    div cx

    call print\_ax

    jmp done

cone:

    call two\_input

    mul ax

    mul bx

    mov cx,16h

    mul cx

    xor dx,dx

    mov cx,15h

    div cx

    call print\_ax

    jmp done

cylinder:

    call two\_input

    mul ax

    mul bx

    mov cx,16h

    mul cx

    xor dx,dx

    mov cx,07h

    div cx

    call print\_ax

    jmp done

area:

    lea dx,str121

    call print\_str

    lea dx,str122

    call print\_str

    lea dx,str123

    call print\_str

    lea dx,str124

    call print\_str

    lea dx,str125

    call print\_str

    lea dx,str126

    call print\_str

    lea dx,str127

    call print\_str

    lea dx,str128

    call print\_str

    lea dx,str129

    call print\_str

    mov al,0

    mov c,0

    next6:

        mov ah,01h

        int 21h

        cmp al,0dh

        jz next7

        mov d,al

        sub d,30h

        mov al,c

        mov dh,0ah

        mul dh

        add d,al

        mov cl,d

        mov c,cl

        jmp next6

    next7:

    mov ah,0

    mov al,c

    cmp al,1

    jz square

    cmp al,2

    jz circle

    cmp al,3

    jz rect

    cmp al,4

    jz triangle

    cmp al,5

    jz cube\_sur

    cmp al,6

    jz cuboid\_sur

    cmp al,7

    jz cylinder\_sur

    cmp al,8

    jz sphere\_sur

    cmp al,9

    jz cone\_sur

square:

    call one\_input

    mul ax

    call print\_ax

    jmp done

rect:

    call two\_input

    mul bx

    call print\_ax

    jmp done

circle:

    call one\_input

    mul ax

    mov cx,16h

    mul cx

    xor dx,dx

    mov cx,07h

    div cx

    call print\_ax

    jmp done

triangle:

    call two\_input

    mul bx

    call print\_ax

    jmp done

cube\_sur:

    call one\_input

    mul ax

    mov cx,06h

    mul cx

    call print\_ax

    jmp done

cuboid\_sur:

    call one\_input

    mov a16,ax

    call two\_input

    mov b16,ax

    mov c16,bx

    mul bx

    mov d16,ax

    mov ax,b16

    mov bx,a16

    mul bx

    add d16,ax

    mov ax,a16

    mov bx,c16

    mul bx

    add ax,d16

    mov cx,02h

    mul cx

    call print\_ax

    jmp done

cylinder\_sur:

    call two\_input

    mul bx

    mov cx,2

    mul cx

    mov cx,16h

    mul cx

    xor dx,dx

    mov cx,07h

    div cx

    call print\_ax

    jmp done

sphere\_sur:

    call one\_input

    mul ax

    mov cx,04h

    mul cx

    mov cx,16h

    mul cx

    xor dx,dx

    mov cx,07h

    div cx

    call print\_ax

    jmp done

cone\_sur:

    call two\_input

    mul bx

    mov cx,16h

    mul cx

    xor dx,dx

    mov cx,07h

    div cx

    call print\_ax

    jmp done

add\_num:

    call two\_input

    add al,bl

    jmp outp

sub\_num:

    call two\_input

    sub al,bl

    jmp outp

mul\_num:

    call two\_input

    mul bl

    jmp outp

div\_num:

    call two\_input

    div bl

    jmp outp

modulo:

    call two\_input

    div bl

    mov al,ah

    jmp outp

frac\_div:

    call two\_input

    fractional\_div al,bl

    call print\_frac

    mov al,c

    jmp done

power:

    call two\_input

    mov a8,al

    mov b8,bl

    call power\_macro

    mov al,c

    jmp outp

fact:

    call one\_input

    mov a16,ax

    call factorial

    mov ax,c16

    call print\_ax

    jmp done

log:

    call two\_input

    mov cx,0

    loop1:

        cmp ax,0

        jz ok

        xor dx,dx

        div bx

        inc cx

        jmp loop1

    ok:

        mov ax,cx

        sub ax,1

        call print\_ax

        jmp done

npr:

    call two\_input

    mov a16,ax

    call factorial

    sub ax,bx

    mov bx,c16

    mov a16,ax

    call factorial

    mov ax,bx

    mov cx,c16

    mov dx,0

    div cx

    call print\_ax

    jmp done

ncr:

    call two\_input

    mov a16,ax

    call factorial

    sub ax,bx

    mov dx,c16

    mov a16,ax

    call factorial

    mov cx,c16

    mov a16,bx

    call factorial

    mov ax,cx

    mov cx,dx

    mul c16

    mov bx,ax

    mov ax,cx

    xor dx,dx

    div bx

    call print\_ax

    jmp done

area\_master:

    call new\_line

outp:

    call print\_al

done:

    hlt

print\_str proc

    mov ah,09

    int 21h

    call new\_line

    ret

print\_str endp

power\_macro proc

    pusha

    mov cl,a8

    rpt1:

        cmp b8,1

        jz power\_macro\_store

        mul cl

        sub b8,1

        jmp rpt1

    power\_macro\_store:

        mov c,al

    popa

    ret

power\_macro endp

new\_line proc

    MOV AH, 2                 ; set output function

    MOV DL, 0DH               ; carriage return

    INT 21H

    MOV DL, 0AH               ; line feed

    INT 21H

    ret

new\_line endp

print\_al proc

    mov ah,0

    call print\_ax

    ret

print\_al endp

one\_input proc

    call new\_line

    lea dx,in1

    call print\_str

    mov al,0

    mov c,0

    next01:

        mov ah,01h

        int 21h

        cmp al,0dh

        jz next02

        mov d,al

        sub d,30h

        mov al,c

        mov dh,0ah

        mul dh

        add d,al

        mov cl,d

        mov c,cl

        jmp next01

    next02:

    mov ah,0

    mov al,c

    push ax

    call new\_line

    pop ax

    ret

one\_input endp

two\_input proc

    call new\_line

    lea dx,in1

    call print\_str

    mov al,0

    mov c,0

    next0:

        mov ah,01h

        int 21h

        cmp al,0dh

        jz next1

        mov d,al

        sub d,30h

        mov al,c

        mov dh,0ah

        mul dh

        add d,al

        mov cl,d

        mov c,cl

        jmp next0

    next1:

    mov ah,0

    mov al,c

    push ax

    call new\_line

    lea dx,in2

    call print\_str

    mov al,0

    mov c,0

    next2:

        mov ah,01h

        int 21h

        cmp al,0dh

        jz next3

        mov d,al

        sub d,30h

        mov al,c

        mov dh,0ah

        mul dh

        add d,al

        mov cl,d

        mov c,cl

        jmp next2

    next3:

    mov ah,0

    mov al,c

    push ax

    call new\_line

    pop bx

    pop ax

    ret

two\_input endp

print\_frac proc

    mov al,c

    call print\_al

    lea dx,str0

    mov ah,09

    int 21h

    mov al,d

    call print\_al

    call new\_line

    ret

print\_frac endp

print\_ax proc

    mov cx,0

    mov dx,0

    label1:

        cmp ax,0

        je print1

        mov bx,10

        div bx

        push dx

        inc cx

        mov dx,0

        jmp label1

    print1:

        cmp cx,0

        je exit

        pop dx

        add dx,48

        mov ah,02h

        int 21h

        dec cx

        jmp print1

    exit:

        ret

print\_ax endp

factorial proc

    pusha

    mov cx,a16

    mov ax,1

    loop4:

        cmp cx,0

        jz fact\_end

        mul cx

        dec cx

        jmp loop4

    fact\_end:

        mov c16,ax

        popa

    ret

factorial endp

code ends

end start

**3.2  Code Analysis**

**Number of Lines: 728**

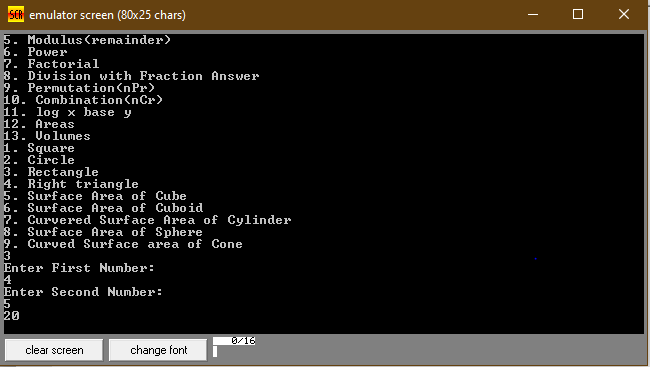
**Number of Functions:**

**Procedure: 9**

**Macro: 1**

1. **Result and Conclusion**

**4.1   Screenshot**

****

**5. CONCLUSION AND FUTURE WORK**

**5.1 CONCLUSION**

* All the proposed functionalities were achieved.
* EMU8086

**5.2 FUTURE WORK**

* Make the program recursive so that it asks for more calculations rather than halting automatically.
* Make the calculator work for bigger inputs.

**6. REFERENCES**

1. www.google.com
2. [www.stackoverflow.com](http://www.ti.com)
3. [www.sciencedirect.com](http://www.electrosome.com)