1. **Title:**

**A Simple Calculator Which Can Perform Eight Functions Using Assembly Language Operations and Principles**.

1. **Objective:**

The main goal of making this project is to make complete use of the concepts and to map out and outline all the learning outcomes of the related course in a single given task by the instructor. While working on emulator EMU8086 we are intended to understand, practice, implement and present all the instructed concepts related to assembly language by the instructor**.**

1. **Theory:**

The Calculator we have built or programmed can perform total eight mathematical functions that include the four fundamental functions **Addition, Subtraction**, **Multiplication**, **and Division.**

Apart from the above mentioned functions this calculator can further perform ‘Multiple Multiplied’ functions like **Cube** of a number and **Square** of a number.

It is capable of performing two more functions i.e. **Percentage** of a number from a certain total and **Comparison** between two numbers.

1. **Implementation:**

The library used in the project is **EMU8086.INC** which lets the user to perform the following built in functions:

* print ‘STRING’ – For printing String
* scan\_num – For taking Integer Input
* print\_num – For displaying value stored in register

**Interrupts Used:**

* **INT 21h/Ah=9;** To Display Output
* **INT16h/Ah=0;** To read a key press

**Functions That This Calculator Can Perform:**

* **Addition**

**Code:**

***Will be run during presentation….***

In the case of addition first we will display the message "Enter first no" using Int 21h next we will call input no as we call each number separately and push the values in dx register further we will move to cx 0 and increment in **‘input.no’** procedure later , call ‘input.no’ pops the value from the bx register and then add both the dx and bx register ,push the value in dx register and moves the ‘msg 5’ address in dx pop dx out later and call view in the end jump exit when addition performed

* **Multiplication:**

**Code:**

***Will be run during presentation….***

In the case of multiplication first moves string of characters in highest bit register moreover stored the address of msg 2 in dx register display using int21h further call input no push the value in dx register, moves the address of ‘msg 3’ in dx register as well and display again using int 21h call input no .pop the bx register moves the value of dx in ax register and perform multiplication operation similarly moves the value of ax register in dx and moves the msg 5 address in dx later pop dx out call view and jump exit when multiplication performed.

* **Subtraction:**

**Code:**

***Will be run during presentation….***

In case of subtraction just as similar to the previous arithmetic operations we moves the string of characters in the highest bit register, also move the address of msg 2 in dx register and display using int 21h, later call ‘input.num’ push the value in dx moves the address of ‘msg3’in dx register again display value using int 21h, call input no once more pop the value from bx

then perform the sub operation subtract the value of dx from bx register and later push the value in dx register also moves the address of ‘msg5’ in dx register display int 21h pop the value of dx call view and jump exit when subtraction preformed.

* **Division:**

**Code:**

***Will be run during presentation….***

For division operation mov the string of characters in the highest bit reg mov the address of msg 2 in dx reg display using int 21h call input no push the value in dx ,mov the address of msg 3 in dx reg as well display using int 21h call input no further pop the value from bx reg mov the value from bx to ax and next mov the value from dx to cx ,store 0 in to dx and 0 int to bx and perform the division operation, divide cx further mov the value from dx to bx and from ax to dx push the value in bx and push the value in dx ,next mov the address of msg 5 in dx display using int 21h pop dx out call view proc pop bx compare bx to 0 if jump is equal to 0 exit

* **Square:**

**Code:**

***Will be run during presentation….***

Performing square operation moves the string of characters in highest bit register next moves the address of msg 2 in dx display using int 21h, call input no and push the value in dx and pop bx out mov the value of dx in ax ,perform multiplication operation on bx moves the value of ax in dx ,push the value in dx further moves the address of msg 5 in dx and display using int 21h later pop dx out call view procedure and jumps exit when square operation performed.

* **Cube:**

**Code:**

***Will be run during presentation….***

In the case of cube operation first we move the string of characters in the highest 8 bit register ah also move the address of msg 2 in the dx register ,which moves the first message input 1 to dx register int 21h helps in displaying the message "enter first no" next call input no ,calling input no as our single input will be counted twice then push the values in dx pop the value of bx out moves the value of dx in ax which moves the first input in the ax register now apply multiplication operation multiplication dx to ax now moves the ax to dx ,mul bx implement the multiplication part again mov ax to dx now finally mov the result to dx reg push the value in bx and dx reg and then push the final result in to stack also mov the address of msg 5 in dx display using int 21h . Print the result pop the dx out and call view proc to show output and jump exit when the cube operation is done.

* **Percentage:**

**Code:**

***Will be run during presentation….***

For percentage operation first we move to the new line and print obtain Mark's next take number of subject as input and mov 1st input from cx to ax move to the next new line using printn enter total number moves the value 100 in to bx reg mul ax with bx and div cx, div ax with cx and finally print the percentage and jump exit when the percentage operation performed.

* **Comparison :**

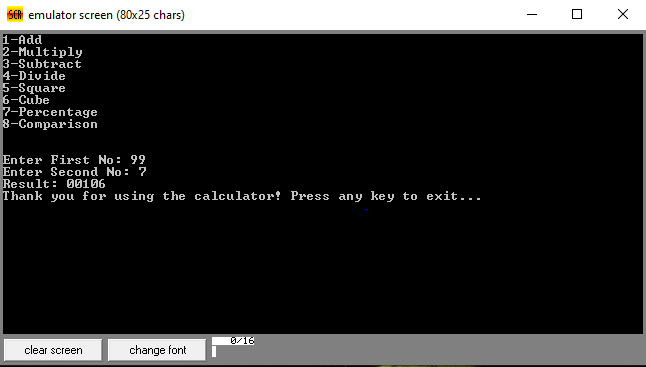
**Code**

***Will be run during presentation….***

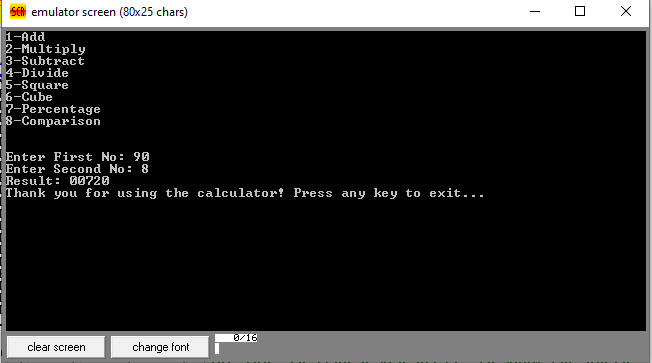
In comparison operation we take input of the first number and mov the value from cx to ax reg next we take input for the second number moving value from cx to ax reg now we compare if they are equal or not we apply je (jump if equal) function ,we again compare to check the first is greater or lesser we apply the jl( jump if less function ) if greater then jump to greater to print a greater msg we once again compare to check the first is greater or lesser we use jg func Jump if greater to point a greater msg print first num is greater than second num then exit for smaller proc print first num is smaller than sec num and jmp exit for equal proc print both numbers are equal and finally jmp exit and ret.

1. **Debugging Test Run:**

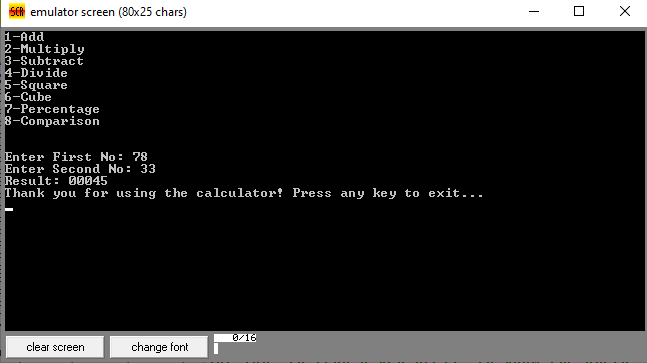
* Addition



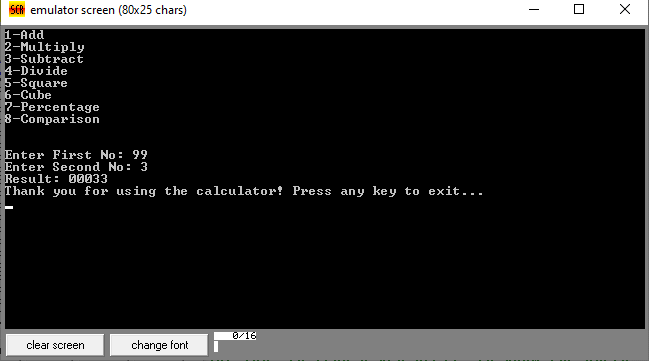
* Multiplication



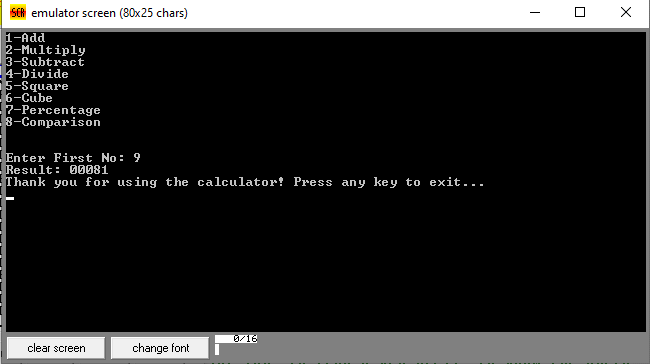
* Subtraction:



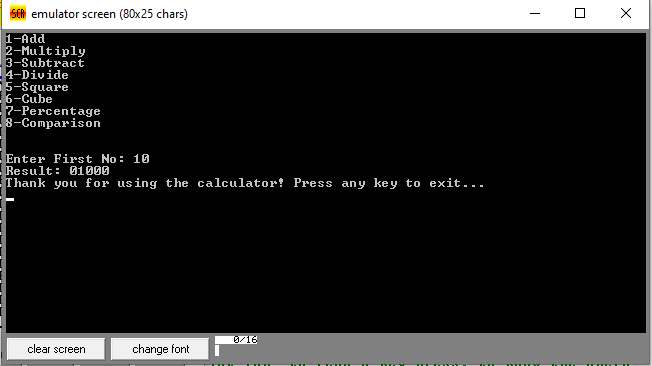
* Division



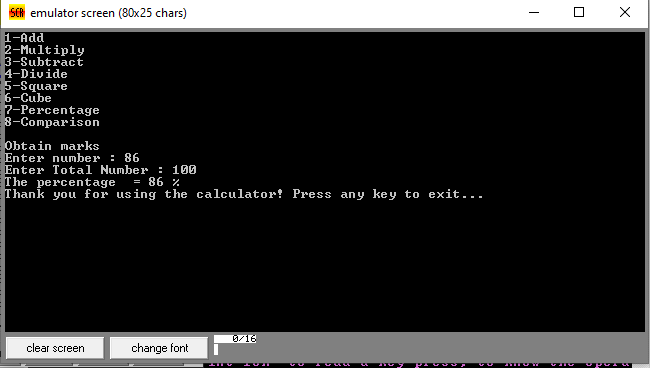
* Square



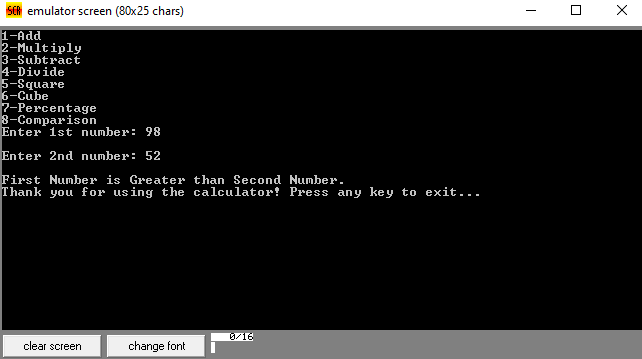
* Cube



* Percentage



* Comparison



* Invalid Input

