**PROJECT REPORT**

**Project Name: “Mini-Calculator”**

**Course Code- CSE 232**

**Mini-Calculator**

****

**INDEX**

* Introduction
* Background
* Project Description
* Flowchart
* Experimental Result
* Advantage
* Disadvantage
* Conclusion

**Introduction**

An assemble (or assembler) language, often abbreviated asm, is a low-level programming language for a computer, or other programmable device, in which there is a very strong (generally one-to -one) correspondence between the language and the language and the architecture’s machine code instructions. Each assembly language is specific to particular computer architecture. In contrast, most high-level

Programming languages are generally portable across multiple architectures but require interpreting or compiling. Assembly language may also be called symbolic machine code. Assembly language is converted into executable machine code by a utility program referred to as an assembler. The conversion process is referred to as assembly, or assembling the source code. Assembly time is the computational step where an assembler is run. Assembly language uses a mnemonic to represent each low-level machine instruction or opcode, typically also each architectural register, flag, etc.

Many operations require one or more operands in order to form a complete instructions and most assembler can take expressions of numbers and named constants as well as registers and labels as operands, freeing the programmer from tedious repetitive calculations. Depending on architecture, these elements are also be combined for specific instructions or addressing mode using offsets or other data as well as fixed addresses. Many assemblers offer additional mechanisms to facilitate program development to control the assembly process, and to aid debugging.

Background

In this project we used some syntax and they are shortly described below:

INT 21h

* Here INT 21h is used for getting input.

MUL BX

* For multiplying the value of AX with BX.

CMP AL, 0DH

* For comparing the value of AL with Enter.

DIV BX

* For dividing the value of AX by BX.

ADD BX,AX

* For adding the value BX with AX

SUB BX,AX

* For subtracting the value BX with AX

CALL NL

* To call a Procedure named NL.

JMP START

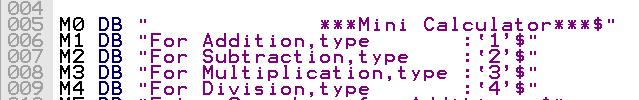
* To JUMP a Label named START.

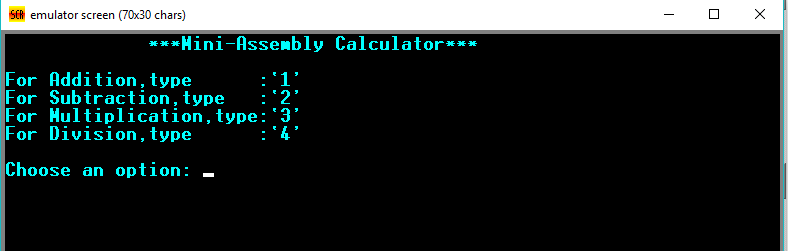
SHL BL

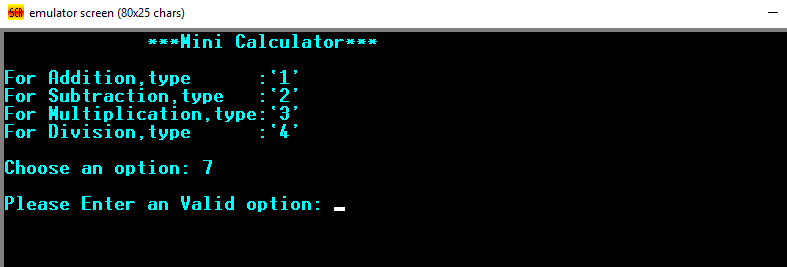
* To Shifting left the value of BL.

Project Description

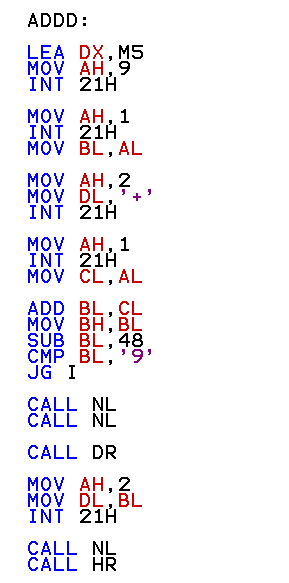
At the very first of our program it will show the options to be calculated..

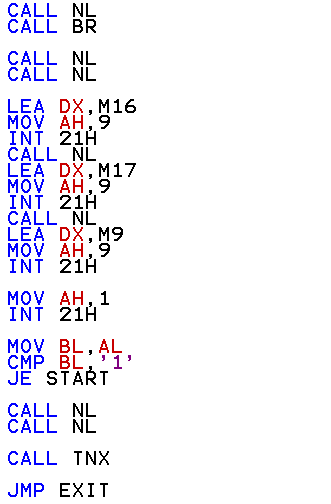




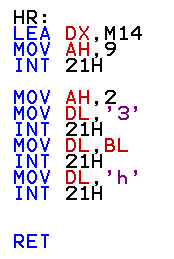
If the input is right then it will go to the next step and if it is not then it will show something like this..

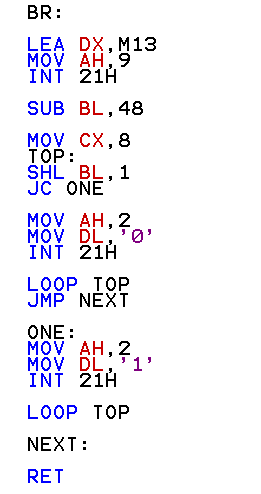
In the next step it will do the operations that user wants to operate .If user input 1 it will take input from user and will do the sum. And the result is distributed in three parts (Decimal, Binary, Hexadecimal)..





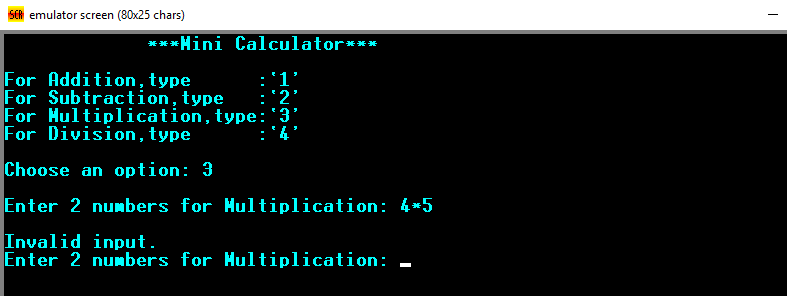
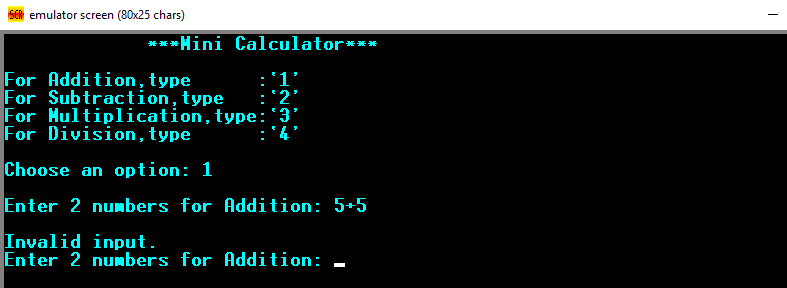
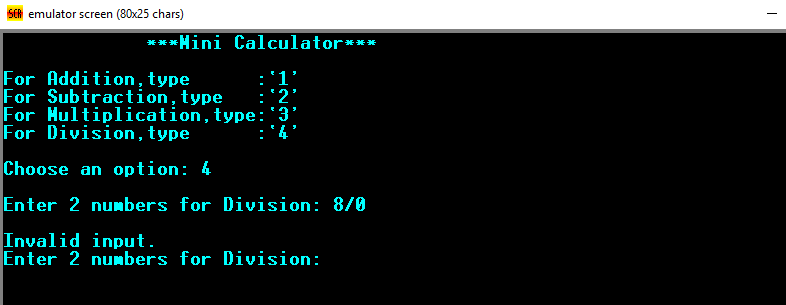
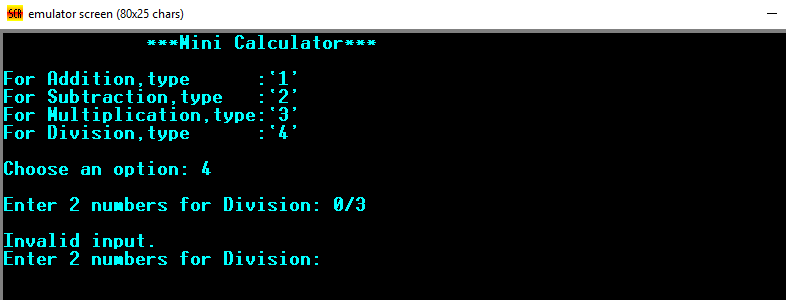
Three types of result is taken through CALL procedure.The examples are given below:

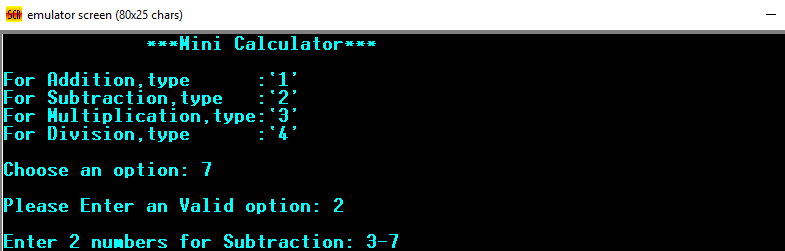


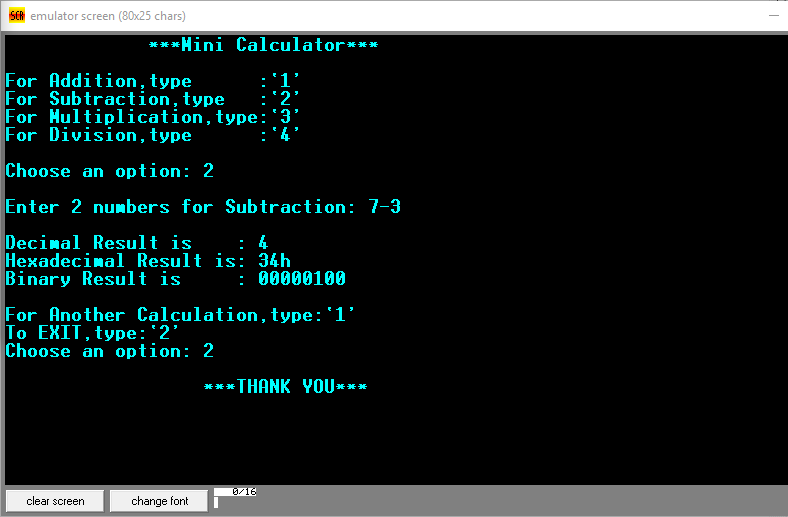


Then if the input is invalid then it will show you further input. And when will be value invalid. Our calculator can not produce result more than 9. So the input must be taken through this purpose. So we cannot take any input that will produce result more than 9. And for Subtraction and Division we cannot take input in an ascending order. Either we can divide something by 0 or 0 cannot be divide by something. Or you can say this is the disadvantage of this program. The invalid input examples are given below:

1. 0 cannot be divide by something



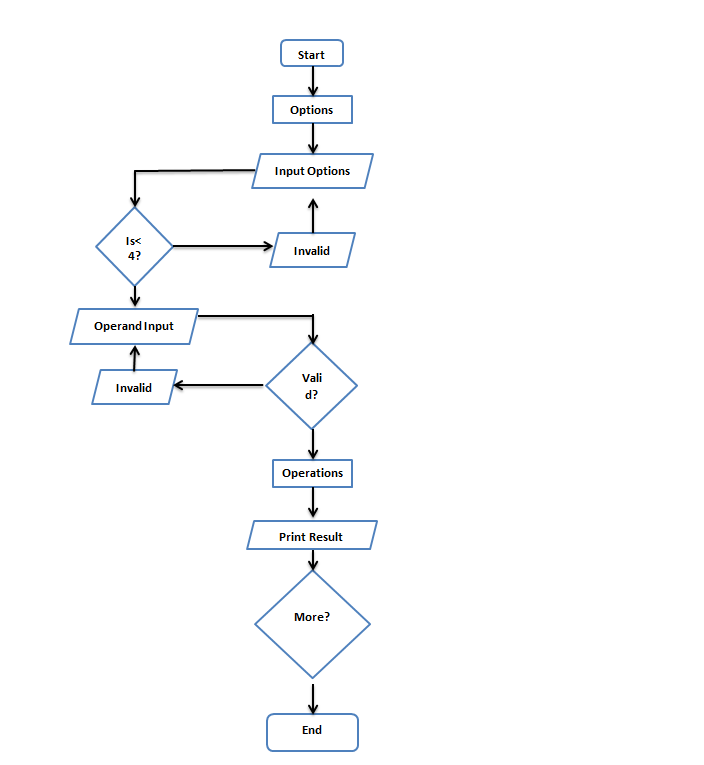


After a complete operation it will look like this..

After every operation it will take your opinion that either you want to exit or you want further calculation. If you want to exit then it will simply exit. Otherwise it will start from the beginning of the program.

That was the description of our project.

Flowchart

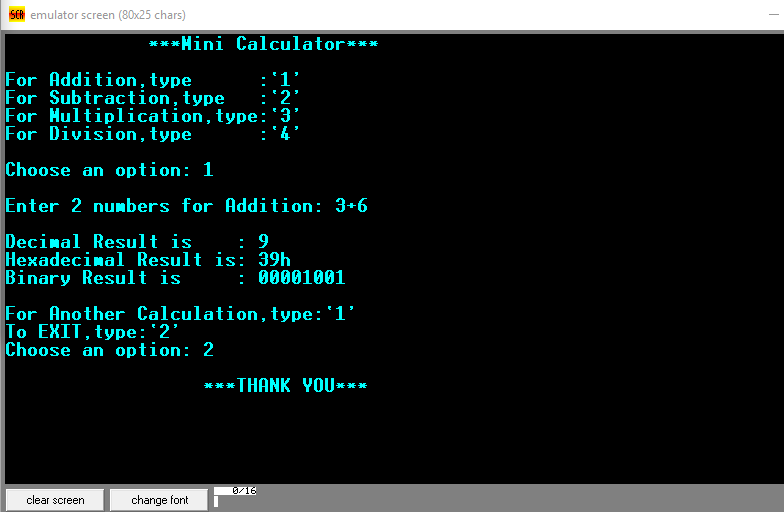
****

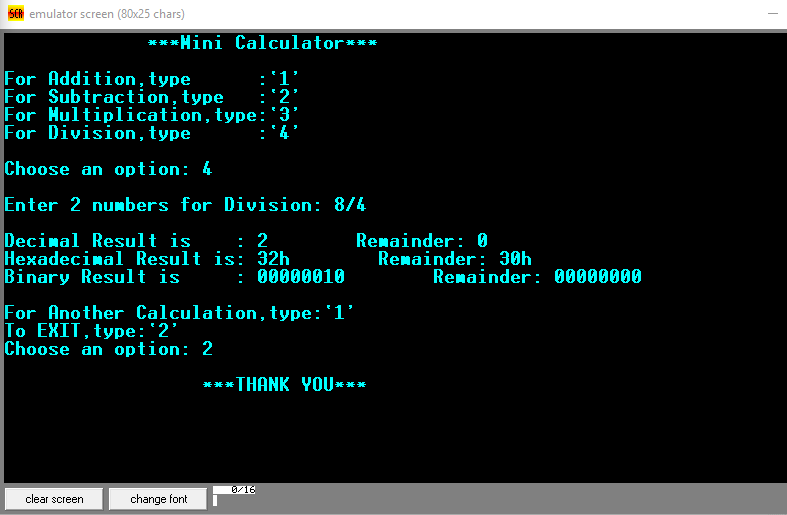
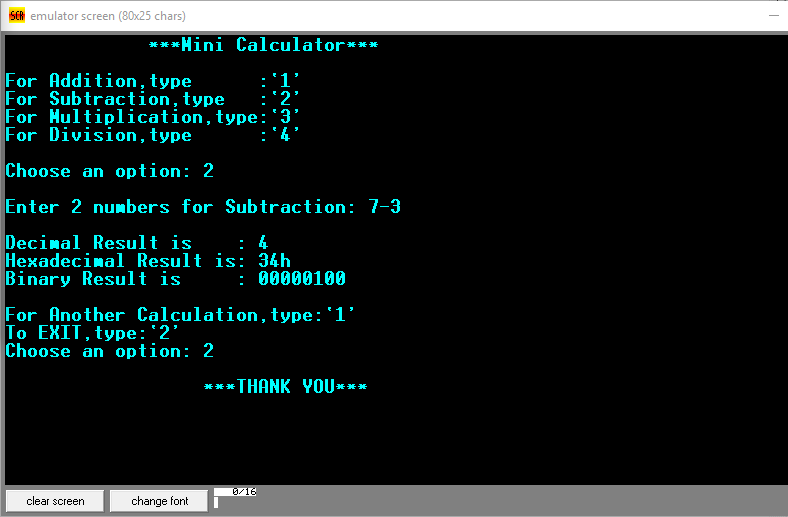
**Experimental Result**

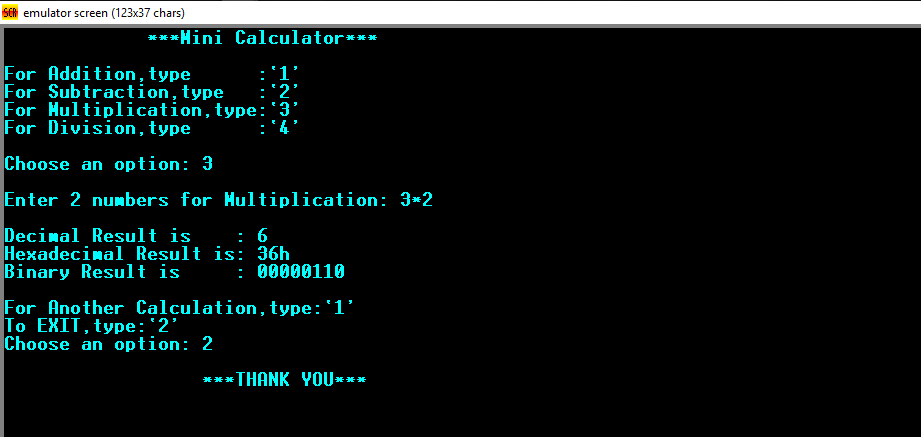
**In this project we use many things. Emu 8086 like**

* **Ascii code.**
* **Loop, jump, cmp, AND etc.**

Here is the snapshot of Addition, Subtraction, Multiplication and Dividation accordingly.







**Advantage**

* It can perform Addition.
* It can perform Subtraction.
* It can perform Dividation.
* It can perform Multiplication.
* It can distribute the result in three parts Decimal, Binary and Hexadecimal.
* It can recalculate.
* It can check inputs whether it is valid or invalid.

**Disadvantage**

* It can not produce result more than 9.
* It can not calculate signed value.
* It can not divide 0.
* It can not divide odd numbers.
* For subtraction and Dividation you have take value in

Ascending order.

**Conclusion**

Assembly language still taught in most computer science and electronic engineering programs. Although few programmers today regularly work with assembly language as a tool, the underlying concepts remain very important.

Our calculator can not calculate with big values. Despite having some limitations, we can get the concept of more perfect programs with this.

**Thank you**