

Name: Nour Safwat Mohamed ID:22030193

Name: Moamen Wageeh Elhelbawi ID:22030140

Project Overview and Idea:

The project is a simple calculator implemented in Assembly language that allows the user to perform basic arithmetic operations. The operations include addition, subtraction, multiplication, division, negation, increment, and decrement. The calculator takes two numeric inputs from the user and performs the selected operation. It then displays the result.

The user interacts with the program via the command line, where they are prompted to select an operation and provide the necessary numbers.

Key Features:

1. **Addition:** Adds two numbers.
2. **Subtraction:** Subtracts one number from another.
3. **Multiplication:** Multiplies two numbers.
4. **Division:** Divides one number by another (with error handling for division by zero).
5. **Negation:** Negates the input number.
6. **Increment:** Increments the input number by 1.
7. **Decrement:** Decrements the input number by 1.

Data Segment:

The data segment contains the messages and variables used in the program.

.DATA

MSG1 DB 'For Add type :1\$', 0

MSG2 DB 10,13,'For Sub type :2\$', 0

...

MSG_INVALID DB 10,13, 'Invalid option. Please choose a valid option: 1, 2, 3, 4, 5, 6, or 7\$',
0

MSG DB 10,13,10,13,' **THANK YOU FOR USING MY APP\$', 0

NUM1 DB ?

NUM2 DB ?

RESULT DB ?

These messages are displayed to the user to guide them through the selection process and inform them of the result. The variables NUM1, NUM2, and RESULT are used to store the input numbers and the result of the arithmetic operations.

Code Segment:

The code segment defines the main logic of the program. It includes logic for performing arithmetic operations based on user input.

1-Display Menu: The program prompts the user with a list of operations to choose from. Each operation is associated with a number.

assembly

Copy code

```
LEA DX, MSG1
```

```
MOV AH, 9
```

```
INT 21H
```

The LEA DX, MSG1 instruction loads the address of the message for the operation into the DX register, and MOV AH, 9 instructs the program to print the string in DX

2-The LEA DX, MSG1 instruction loads the address of the message for the operation into the DX register, and MOV AH, 9 instructs the program to print the string in DX.

```
MOV AH, 1
```

```
INT 21H
```

```
MOV BH, AL
```

```
SUB BH, 30h
```

The program uses MOV AH, 1 to read a single character from the keyboard, and then it converts the character (ASCII) to a number by subtracting 30h.

3-Arithmetic Operations: Based on the user's choice, the program performs the corresponding operation. The logic for each operation is contained in separate labels such as ADD, SUB, MUL, DIV, etc.

Addition:

ADD:

```
LEA DX,MSG9
MOV AH,9
INT 21H
MOV AH,1
INT 21H
MOV BL,AL
LEA DX,MSG10
MOV AH,9
INT 21H
MOV AH,1
INT 21H
MOV CL,AL
ADD AL,BL
```

This section prompts the user for two numbers and adds them. The result is displayed after the operation is complete.

Subtraction:

SUB:

LEA DX,MSG9

MOV AH,9

INT 21H

MOV AH,1

INT 21H

MOV BL,AL

LEA DX,MSG10

MOV AH,9

INT 21H

MOV AH,1

INT 21H

MOV CL,AL

SUB BL,CL

Similarly, the subtraction operation subtracts one number from another and displays the result.

MUL:

```
LEA DX,MSG9
MOV AH,9
INT 21H
MOV AH,1
INT 21H
SUB AL,30H
MOV NUM1,AL
LEA DX,MSG10
MOV AH,9
INT 21H
MOV AH,1
INT 21H
SUB AL,30H
MOV NUM2,AL
MUL NUM1
MOV RESULT,AL, AAM
```

The multiplication operation multiplies the two numbers and then displays the result.

Error Handling: For the division operation, there is a check for division by zero:

DIV_ZERO:

```
LEA DX, MSG6
MOV AH, 9
INT 21H
LEA DX, MSG
MOV AH, 9, INT 21H
```

If the user attempts to divide by zero, an error message is displayed.

Exit: After displaying the result, the program exits by calling INT 21H with AH = 4Ch

LEA DX, MSG

MOV AH, 9

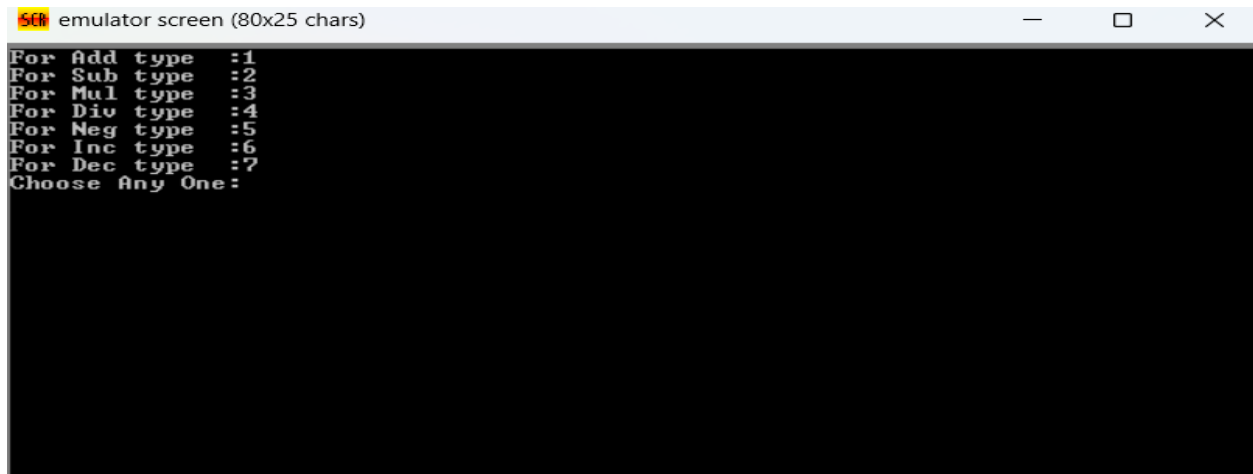
INT 21H

MOV AH, 4Ch

INT 21H

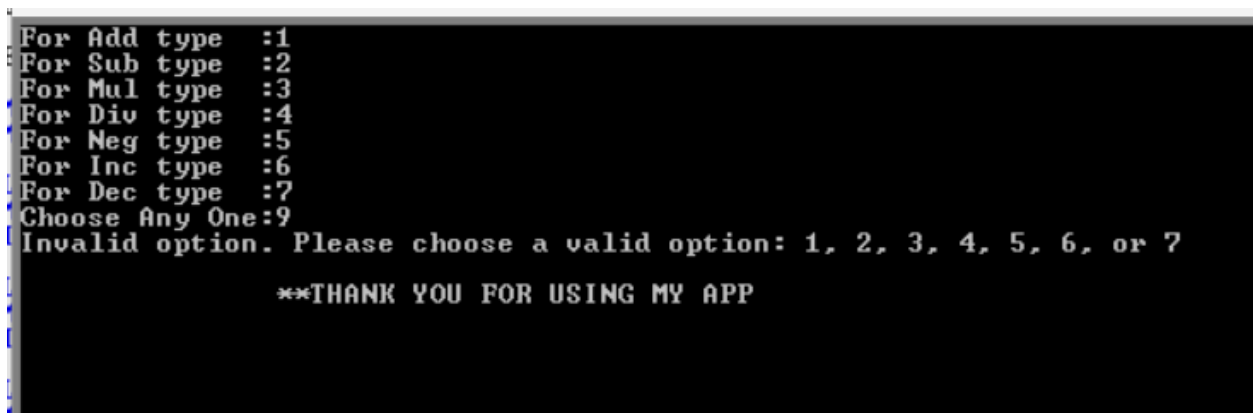
Example Output:

1-Display Menu:



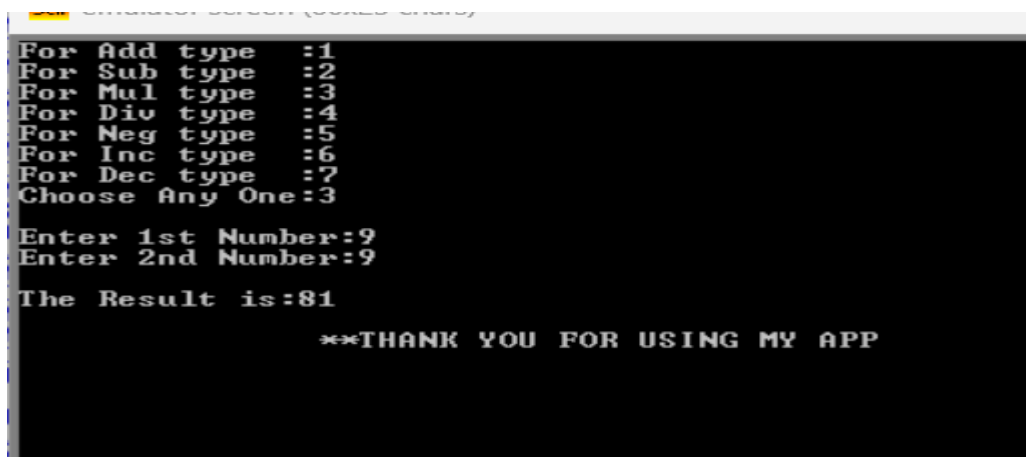
```
emulator screen (80x25 chars)
For Add type :1
For Sub type :2
For Mul type :3
For Div type :4
For Neg type :5
For Inc type :6
For Dec type :7
Choose Any One:
```

If you entered invalid number



```
For Add type :1
For Sub type :2
For Mul type :3
For Div type :4
For Neg type :5
For Inc type :6
For Dec type :7
Choose Any One:9
Invalid option. Please choose a valid option: 1, 2, 3, 4, 5, 6, or 7
**THANK YOU FOR USING MY APP
```

2- MUL:



```
emulator screen (80x25 chars)
For Add type :1
For Sub type :2
For Mul type :3
For Div type :4
For Neg type :5
For Inc type :6
For Dec type :7
Choose Any One:3
Enter 1st Number:9
Enter 2nd Number:9
The Result is:81
**THANK YOU FOR USING MY APP
```


3- SUB:

```
emulator screen (60x25 chars)
For Add type :1
For Sub type :2
For Mul type :3
For Div type :4
For Neg type :5
For Inc type :6
For Dec type :7
Choose Any One:2

Enter 1st Number:9
Enter 2nd Number:6

The Result is:3

**THANK YOU FOR USING MY APP
```

4- Div:

```
For Add type :1
For Sub type :2
For Mul type :3
For Div type :4
For Neg type :5
For Inc type :6
For Dec type :7
Choose Any One:4

Enter 1st Number:9
Enter 2nd Number:3

The Result is:03

**THANK YOU FOR USING MY APP
```

If you divided by zero

```
For Add type :1
For Sub type :2
For Mul type :3
For Div type :4
For Neg type :5
For Inc type :6
For Dec type :7
Choose Any One:4

Enter 1st Number:4
Enter 2nd Number:0
Error division

**THANK YOU FOR USING MY APP

**THANK YOU FOR USING MY APP
```

5- ADD:

```
For Add type :1
For Sub type :2
For Mul type :3
For Div type :4
For Neg type :5
For Inc type :6
For Dec type :7
Choose Any One:1

Enter 1st Number:9
Enter 2nd Number:9

The Result is:18

**THANK YOU FOR USING MY APP
```

6- INC:

```
For Add type :1
For Sub type :2
For Mul type :3
For Div type :4
For Neg type :5
For Inc type :6
For Dec type :7
Choose Any One:6

Enter 1st Number:6

The Result is:7

**THANK YOU FOR USING MY APP
```

7- DEC:

```
For Add type :1
For Sub type :2
For Mul type :3
For Div type :4
For Neg type :5
For Inc type :6
For Dec type :7
Choose Any One:7

Enter 1st Number:7

The Result is:6

**THANK YOU FOR USING MY APP
```

Conclusion:

The Assembly program demonstrates the ability to implement basic arithmetic operations. By using simple user input, conditional logic, and arithmetic instructions, the program performs calculations and handles errors, such as division by zero, effectively.

The program is modular, with each arithmetic operation encapsulated in a separate section of the code, making it easy to understand and extend if needed. The use of Assembly language also highlights the importance of low-level programming skills and memory management in computing.