
UITS

UNIVERSITY OF INFORMATION TECHNOLOGY AND SCIENCES

**Name: Nafiz Al Zawad**
**Id: 0432220005101041**
**Course Code: CSE 360**
**Submitted To: Mr. Md. Ismail**

## Lab Report: Assembly Language Simple Calculator

---

### 1. Introduction

In this lab, we implemented a simple calculator program using assembly language for the x86 architecture. The calculator performs basic arithmetic operations: addition, subtraction, multiplication, and division. It accepts user input from the keyboard, processes the operations, and displays the result on the screen.

This project aimed to demonstrate the understanding of assembly language instructions, interrupt handling (INT 21h), and data manipulation in a low-level programming environment. Additionally, it provided insight into the interaction between the processor and hardware through system calls for I/O operations.

---

### 2. Methodology

To implement the simple calculator in assembly language, we followed these key steps:

1. **Initialization**:
    - Defined necessary data segments for displaying messages and storing results.
    - Setup interrupt calls (`INT 21h`) for displaying text and accepting input from the user.
2. **User Interface**:
    - Displayed a welcome message and options for the four operations (Add, Multiply, Subtract, Divide).
    - Prompted the user to input a number between 1 and 4 to choose an operation.
    - For each operation, prompted the user to input two numbers.
3. **Input Handling**:
    - Used `INT 16h` to read a single character input from the keyboard.
    - Converted the ASCII input values into numerical digits by subtracting `30h`.
4. **Arithmetic Operations**:

- For each operation (addition, subtraction, multiplication, division), performed the required calculation using registers (`AX`, `BX`, `CX`, `DX`).

- Each operation was implemented as a separate block, with a dedicated routine for handling the logic.

5. **Result Output**:

   - Displayed the result of the operation using `INT 21h` after performing the calculation.

   - Handled formatting and output of multi-digit numbers using loops and division for conversion to ASCII.

6. **Error Handling**:

   - Checked user input for validity (1-4 for operations) and re-prompted if an invalid choice was entered.

7. **Exit**:

   - Displayed a thank-you message upon exit and waited for a final key press before terminating the program.

---

**3. Implementation**

The following assembly code implements the calculator functionality:

assembly
CopyEdit

```
org 100h
.model small
.data
; Message definitions and prompt strings for user interaction
msg0: db 0dh, 0ah, " __> Simple calculator <__", 0dh, 0ah, '$'
msg: db 0dh, 0ah, "1-Add", 0dh, 0ah, "2-Multiply", 0dh, 0ah,
"3-Subtract", 0dh, 0ah, "4-Divide", 0Dh, 0Ah, '$'
msg1: db 0dh, 0ah, "Enter a number between 1,4 if you want any
calculation ::", 0Dh, 0Ah, '$'
msg2: db 0dh, 0ah, "Enter First No : $"
```

```
msg3: db 0dh, 0ah, "Enter Second No : $"
msg4: db 0dh, 0ah, "Choice Error....please Enter any key which is in
range (1-4)", 0Dh, 0Ah, "$"
msg5: db 0dh, 0ah, "Result : $"
msg6: db 0dh, 0ah, "Thank you for using the calculator! Press any
key...", 0Dh, 0Ah, '$'

.code
start:
    mov ah, 9
    mov dx, offset msg0
    int 21h
    ; Show operation menu
    mov ah, 9
    mov dx, offset msg
    int 21h
    ; Request choice input from user
    mov ah, 9
    mov dx, offset msg1
    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
    mov ah, 09h
    mov dx, offset msg4
    int 21h
    mov ah, 0
    int 16h
    jmp start

; Code for Addition, Multiply, Subtract, Divide goes here ...
```

```
exit:
    mov dx, offset msg6
    mov ah, 09h
    int 21h
    mov ah, 0
    int 16h
    ret
```

---

**4. Input**

The calculator takes two inputs from the user for the arithmetic operations. The process is as follows:

1. **First Input**: The user is prompted to enter the first number, which is read as a character, converted to a numeric value, and stored in a register.

2. **Second Input**: Similarly, the user is prompted to enter the second number for the operation.

3. **Choice Input**: The user selects an operation by entering a number between 1 and 4, which corresponds to:

    - 1 for Addition
    - 2 for Multiplication
    - 3 for Subtraction
    - 4 for Division

---

**5. Output**

After processing the input values, the program outputs the result of the selected operation. It also handles multiple digits in the result by breaking them down and displaying them individually using ASCII codes.

For example:

For addition: If the user selects option "1", the result of adding two numbers is displayed in the following format:

rust
CopyEdit
```
Result : 15
```

- 
- For other operations like multiplication, subtraction, and division, similar output formats are used for displaying the results.

---

**6. Conclusion**

In conclusion, this lab successfully demonstrates the implementation of a simple calculator using assembly language for the x86 architecture. The key concepts explored include input handling, arithmetic operations, system interrupts, and result formatting.

The calculator efficiently handles basic arithmetic tasks and displays results. This project highlights the power of assembly language in managing hardware resources directly, making it ideal for low-level programming and systems development.

While the program works as expected, future enhancements could include adding more error handling (such as division by zero) and extending the functionality to support floating-point operations.