Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year:2023), B.Sc. in CSE (Day)

Project Report

Project Title: Design and Build a Simple Computer Architecture Program.

Course Title: Computer Architecture

Course Code: CSE-211

Section: PC-212 'DB'

Student Details:

Name	ID
Md. Anas Khan	201902037
Md. Shazzad Hossain	212902073

Submission Date : 22-06-2023
Course Teacher's Name : Jarin Tasnim Tonvi

Project report Status

Marks: **Signature:**

Comments: **Date:**

Table Of Contents

Chapter 1
Introduction

1.1 Introduction:

The purpose of this project is to design and build a simple computer architecture program that can execute basic arithmetic operations. The program is implemented using the emu8086 assembler and aims to provide insights into the execution of programs and operations within a computer architecture.

1.2 Objectives:

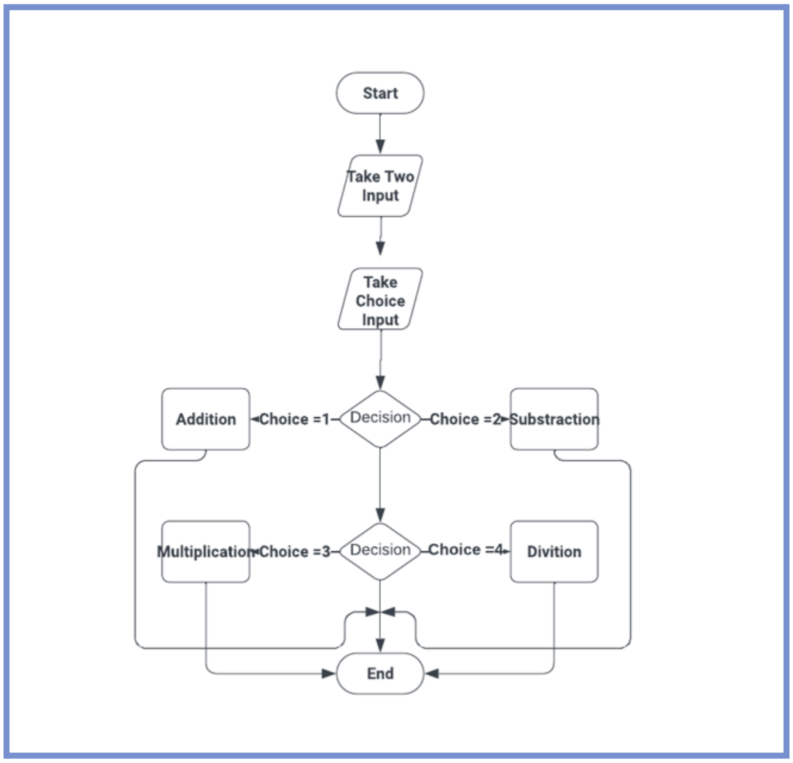
- Gain a deeper understanding of computer architecture fundamentals.
- Implement a program that can perform addition, subtraction, multiplication, and division operations.
- Learn to use the emu8086 assembler for program development.

Chapter 2
Design/Development/Implementation of the project

Design and Development:

The program is designed using the emu8086 assembler and follows a structured approach. The source code provided includes the necessary instructions and data declarations to execute the desired operations. The program prompts the user to enter two integer numbers and choose an operation from a menu. Based on the user's choice, the program performs the selected operation and displays the result.

Flowchart:



Source Code:

The provided source code consists of the necessary code segments for data initialization, input handling, operation execution, and result display. It utilizes various instructions and functions specific to the emu8086 assembler to achieve the desired functionality.

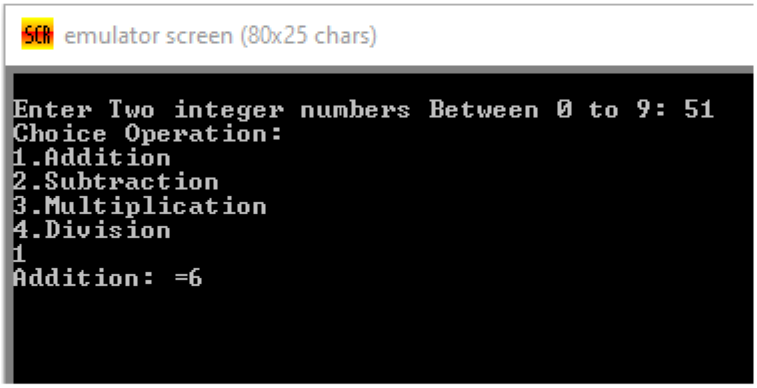
```
.model small
.stack 100h
.data
   a dw ?
   b dw ?
   result dw ?
   check dw ?
   msg1 db 0Dh, 0Ah, 'Enter Two integer numbers Between 0 to 9: $'
   msg2 db 0Dh, 0Ah, 'Choice Operation:', 0Dh, 0Ah, '1.Addition', 0Dh, 0Ah, '2.Subtraction', 0Dh, 0Ah, '3.Multiplication', 0Dh, 0Ah, '4.Division', 0Dh, 0Ah, '$'
   msg3 db 0Dh, 0Ah, 'Addition: $'
```

```
    msg4 db 0Dh, 0Ah, 'Subtraction: $'
    msg5 db 0Dh, 0Ah, 'Multiplication: $'
    msg6 db 0Dh, 0Ah, 'Division: $'
    msg7 db 0Dh, 0Ah, 'Invalid choice...!$'
.code
    mov ax, @data
    mov ds, ax

    mov ah, 9
    lea dx, msg1
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h
    mov bh, 0
    mov bl, al
    mov a, bx

    mov ah, 1
    int 21h
    sub al, 30h
    mov bh, 0
    mov bl, al
    mov b, bx

    mov ah, 9
    lea dx, msg2
    int 21h

    mov ah, 1
    int 21h
    sub al, 30h
    mov bl, al
    mov check, bx

    cmp check, 1
    je addition
    cmp check, 2
    je subtraction
    cmp check, 3
    je multiplication
    cmp check, 4
    je division
    jmp invalid

addition:
    mov ax, a
    add ax, b
    mov result, ax

    mov ah, 9
    lea dx, msg3
    int 21h
    mov ah, 2
    mov dl, '='
    int 21h
    mov ax, result
    call display_result
    jmp exit

subtraction:
    mov ax, a
    sub ax, b
    mov result, ax

    mov ah, 9
    lea dx, msg4
    int 21h
    mov ah, 2
    mov dl, '='
    int 21h
    mov ax, result
    call display_result
    jmp exit

multiplication:
    mov ax, a
    mul b
    mov result, ax

    mov ah, 9
    lea dx, msg5
    int 21h
    mov ah, 2
    mov dl, '='
    int 21h
    mov ax, result
    call display_result
    jmp exit

division:
    mov ax, a
    cwd
    idiv b
    mov result, ax

    mov ah, 9
    lea dx, msg6
    int 21h
    mov ah, 2
    mov dl, '='
    int 21h
    mov ax, result
    call display_result
    jmp exit

invalid:
    mov ah, 9
    lea dx, msg7
    int 21h
    jmp exit

display_result:
    push ax
    push bx
    push cx
    push dx

    cmp ax, 0
    jns positive
    mov dl, '-'
    int 21h
    neg ax

positive:
    mov bx, 10
    xor cx, cx

convert_loop:
    xor dx, dx
    div bx
    add dl, '0'
    push dx
    inc cx
    test ax, ax
    jnz convert_loop

display_loop:
    pop dx
    mov ah, 2
    int 21h
    loop display_loop

    pop dx
    pop cx
    pop bx
    pop ax
    ret

exit:
    mov ah, 4Ch
    int 21h
end
```

Chapter 3
Performance Evaluation

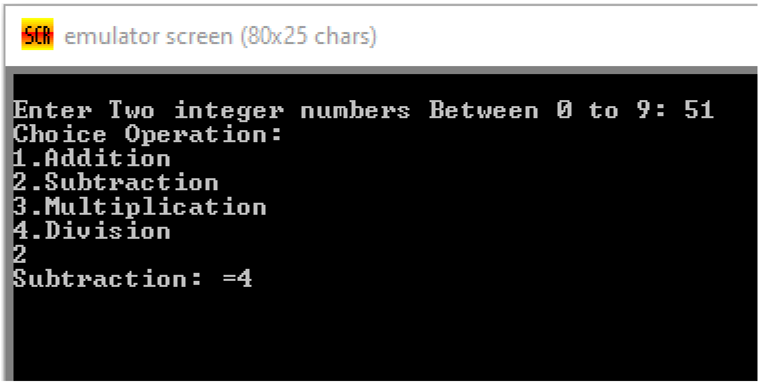
3.1 Result and Discussion:

Upon execution, the program evaluates the user's input and performs the selected arithmetic operation. The results are then displayed on the screen. The performance of the program is evaluated based on its ability to accurately calculate the desired arithmetic operation and provide the correct output. The program aims to deliver reliable and accurate results for various input scenarios.



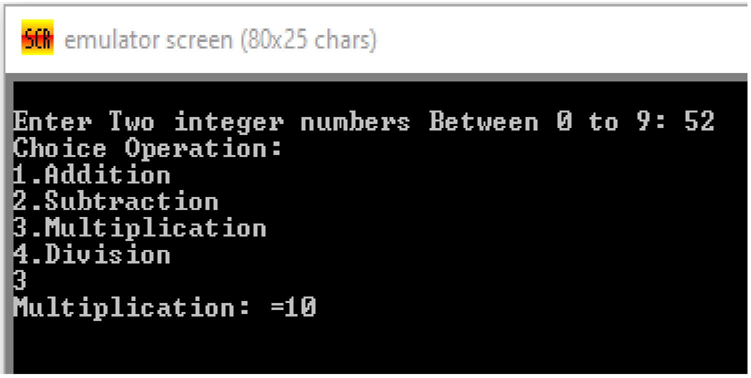
```
Enter Two integer numbers Between 0 to 9: 51
Choice Operation:
1.Addition
2.Subtraction
3.Multiplication
4.Division
1
Addition: =6
```

Fig1: Addition Operation.



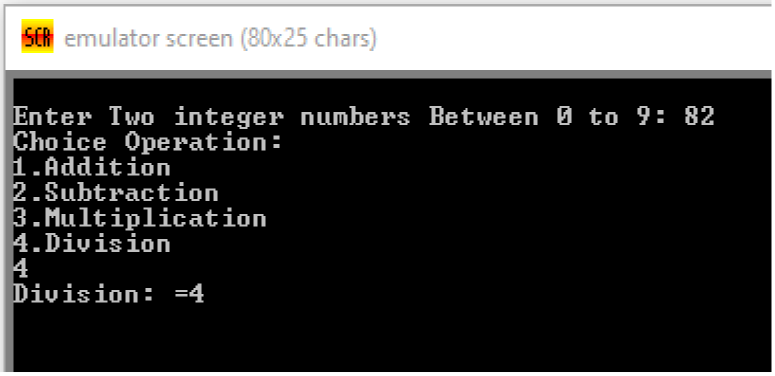
```
Enter Two integer numbers Between 0 to 9: 51
Choice Operation:
1.Addition
2.Subtraction
3.Multiplication
4.Division
2
Subtraction: =4
```

Fig2: Subtraction Operation.



```
Enter Two integer numbers Between 0 to 9: 52
Choice Operation:
1.Addition
2.Subtraction
3.Multiplication
4.Division
3
Multiplication: =10
```

Fig3: Multiplication Operation.



```
Enter Two integer numbers Between 0 to 9: 82
Choice Operation:
1.Addition
2.Subtraction
3.Multiplication
4.Division
4
Division: =4
```

Fig4: Division Operation

Chapter 4
Conclusion

4.1 Introduction:

In conclusion, the design and development of this simple computer architecture program provide valuable insights into the execution of basic arithmetic operations within a computer system. By utilizing the emu8086 assembler, we were able to create a functional program that can accurately perform addition, subtraction, multiplication, and division operations. The program demonstrates the foundational principles of computer architecture and serves as a stepping stone for further exploration and learning in this field.

4.2 Scope for future work:

Moving forward, this project can be expanded to incorporate additional features and functionalities. It can be extended to handle a wider range of input values, implement error handling and input validation, and support more complex arithmetic operations. Furthermore, the program can be enhanced to include a graphical user interface (GUI) for a more user-friendly interaction. Exploring optimization techniques and improving performance can also be considered as future enhancements to the project.