COMPUTER ORGANIZATION & ASSEMBLY LANGUAGE

Powers Calculator

Submitted To : Dr. Muhammad Asfand-e-yar

GROUP MEMBERS
M. Hashim Nazir (01-134231-055)
Awais Ibrahim (01-134231-014)
Ammar Jamil (01-134231-010)

May 28, 2024

BAHRIA UNIVERSITY ISLAMABAD

Contents

1 **Introduction** 4

1.1 Abstract 4

1.2 Problem Statement 4

1.3 Background 4

1.4 Objectives 4

2 **Implementation** 5

2.1 Software Used 5

2.1.1 Visual Studio 5

2.2 Library Used 5

2.2.1 Irvine32 Library 5

3 **Project Code** 5

4 **Data Flow Diagram** 17

5 **Code Explanation** 18

5.1 Main Menu 18

5.1.1 Output 18

5.2 Square Root Procedure 19

5.3 Square Procedure 19

5.4 Cube Procedure 20

5.5 Cube Root Procedure 20

5.5.1 Output 20

5.6 Power Procedure 21

5.6.1 Output 21

5.7 Addition Procedure 21

5.7.1 Output 22

5.8 Subtraction Procedure 22

5.8.1 Output 22

5.9 Multiplication Procedure 23

5.9.1 Output 23

5.10 Division Procedure 23

5.10.1 Output 24

5.11 Error Handling 24

6 Conclusion **24**

1

Introduction

1.1 Abstract

Calculators are indispensable tools in numerous fields, including education, engineering, science, and everyday life. This project "Powers Calculator" focuses on developing an advanced calculator using assembly language, known for its efficiency and control over hardware resources. The goal is to create a reliable, efficient, and user-friendly tool that can handle a wide range of mathematical operations. By leveraging assembly language, we aim to achieve superior performance and precision in calculations.

1.2 Problem Statement

Mathematical computations are fundamental to many disciplines, and accurate and efficient calculation tools are essential. This project seeks to address the gap by developing a multi-functional calculator that provides a comprehensive solution for various mathematical needs. The calculator will support operations such as finding square roots, squares, cube roots, cubes, addition, subtraction, multiplication, division, and calculating powers of numbers, thus catering to a broad spectrum of users.

1.3 Background

The evolution of calculators from simple arithmetic devices to sophisticated scientific instruments reflects the growing complexity of mathematical needs. Early calculators were limited to basic operations, but advancements in technology have enabled the development of calculators that can handle complex functions. This project builds upon this legacy by creating a multi-functional calculator using assembly language, which offers high efficiency and precise control over hardware operations. Assembly language is known for its speed and low-level access to hardware, making it ideal for creating a high-performance calculator.

1.4 Objectives

- Develop a calculator that can perform a wide range of mathematical functions with high accuracy.
- Ensure the calculator has a simple and intuitive user interface, making it accessible to a broad audience.
- Achieve high performance and efficiency in all calculations by leveraging the capabilities of assembly language.
- Provide comprehensive documentation and user guides to assist users in understanding and utilizing the calculator effectively.

2

Implementation

2.1 Software Used

2.1.1 Visual Studio

Visual Studio 2022: An integrated development environment (IDE) used for writing, testing, and debugging the assembly code.

2.2 Library Used

2.2.1 Irvine32 Library

Irvine32 Library is a collection of procedures and macros designed to simplify assembly language programming on the x86 architecture.

3

Project Code

```
include irvine32.inc
.data
  prompt0 BYTE " CALCULATOR ", 0
 prompt BYTE " Enter the number :  ",0
 prompt2 byte " This number is not a perfect square ",0
 prompt3 byte " square root is ", 0
 prompt5 byte " Square of the number is : ", 0
 prompt6 byte " WRONG CHOICE !!! ", 0
 prompt7 byte " Enter the first number : ", 0
 prompt8 byte " Enter the second number : ", 0
 prompt9 byte " The Answer is ", 0
 prompt10 byte " Math error !!", 0
 prompt12 byte " cuberoot of the number is : ", 0
 prompt13 byte " cuberoot is in decimal points ", 0
 prompt14 byte " cube of the number is : ", 0
 prompt15 byte " Enter the power of the number : ", 0
 prompt16 byte " Ans = ", 0
 prompt17 byte  " 1 / ", 0
 val Dword ?
 val2 Dword ?
 prompt4 byte " 1. Find Square root of the number ",0dh,0ah
 byte " 2. Find Square of the number",0dh,0ah
 byte " 3. Find Cuberoot of the number",0dh,0ah
 byte " 4. Find Cube of the number",0dh,0ah
 byte " 5. Find Power of the number",0dh,0ah
 byte " 6. Addition ",0dh,0ah
 byte " 7. Subtraction ",0dh,0ah
 byte " 8. Multiplication ",0dh,0ah
 byte " 9. Division ",0dh,0ah
 byte " 10. Exit ",0dh,0ah
 byte " Enter your choice : "
.code
Addition PROC
   mov edx, offset prompt7
   call writestring
   call readint
   mov ebx, eax
   mov edx, offset prompt8
   call writestring
   call readint
   add eax, ebx
   mov edx, offset prompt9
   call writestring
   call writeint
   call crlf
   call crlf
   ret
Addition endp

Subtraction PROC
   mov edx, offset prompt7
   call writestring
   call readint
   mov ebx, eax
   mov edx, offset prompt8
   call writestring
   call readint
   sub ebx, eax
   mov eax, ebx
   mov edx, offset prompt9
   call writestring
   call writeint
   call crlf
   call crlf
   ret
Subtraction endp

Multiplication PROC
   mov edx, offset prompt7
   call writestring
   call readint
   mov ebx, eax
   mov edx, offset prompt8
   call writestring
   call readint
   mul ebx
   mov edx, offset prompt9
   call writestring
   call writeint
   call crlf
   call crlf
   ret
Multiplication endp
```

```
Division PROC
    mov edx, offset prompt7
    call writestring
    call readint
    mov ebx, eax
res:
    mov edx, offset prompt8
    call writestring
    call readint
cmp eax, 0
JE invalid
JNE continue
invalid:
mov edx, offset prompt10
call writestring
call crlf
call crlf
jmp res
continue:
mov edx, 0
    mov ecx, eax
    mov eax, ebx
    mov ebx, ecx
div ebx
    mov edx, offset prompt9
    call writestring
    call writeint
    call crlf
    call crlf
   ret
Division endp

square PROC
```

```
mov edx, offset prompt
call writestring
call readint
mul eax
mov edx, offset  prompt5
call writestring
call writeint
    call crlf
    call crlf
ret
square endp

findSquareRoot PROC
mov edx, offset prompt
call writestring
call readint
mov ecx, 1
cmp eax, 1
je found
mov ecx, 0
cmp eax, 0
je found
mov ebx, eax
xor edx, edx
mov ecx, 2
div ecx
mov edx, 0
loop1:
mov ecx, eax
mul eax
cmp ecx, 1
JE nosqrt
cmp eax, ebx
```

```
je found
jne notfound
notfound:
xor edx, edx
div ecx
sub eax, 1
loop loop1
found:
mov edx, offset prompt3
call writestring
mov eax, ecx
call writeint
jmp l3
nosqrt:
mov edx, offset prompt2
call writestring
jmp l3
l3:
call crlf
    call crlf
ret
findSquareRoot endp

FindCubeRoot PROC
mov edx, offset prompt
call writestring
call readint
cmp eax, 1
JE ans1
cmp eax, -1
JE ans1
cmp eax, 0
JNE elsecondition
```

```
ans1:
mov edx, offset prompt12
call writestring
call writeint
jmp quit4
elsecondition:
cmp eax, -8
JLE check
cmp eax, 8
JL notfound
check:
mov val, eax
cdq
mov ebx, 2
idiv ebx
cdq
idiv ebx
mov val2, eax
cmp eax, -8
JLE multi2
jmp check2
multi2:
sub eax, val2
sub eax, val2
check2:
mov ecx, eax
mov ebx, val
L1:
mov eax, val2
mul val2
mul val2
cmp ebx, eax
JE found
```

```
cmp ecx, 1
JE notfound
cmp val2, -1
JE notfound
cmp val2, 0
JL low1
JG high1
low1:
    add val2, 1
jmp loop1
high1:
sub val2, 1
loop1:
loop L1
found:
mov edx, offset prompt12
call writestring
mov eax, val2
call writeint
jmp quit4
notfound:
mov edx, offset prompt13
call writestring
quit4:
call crlf
    call crlf
ret
FindCubeRoot endp

cube PROC
   mov edx, offset prompt
   call writestring
   call readint
   mov ebx, eax
   mul ebx
   mul ebx
   mov edx, offset prompt14
   call writestring
   call writeint
   call crlf
   call crlf
   ret
cube endp

power PROC
   mov edx, offset prompt
   call writestring
   call readint
   mov ebx, eax
   mov edx, offset prompt15
   call writestring
   call readint
   mov ecx, eax
   mov eax, ebx
   mov val, ecx
   cmp ecx, 1
   JE out1
   cmp ecx, -1
   JE out1
   cmp ecx, 0
   JE ans1
   JL divided
   JNL continue
   divided:
sub ecx, val
sub ecx, val
```

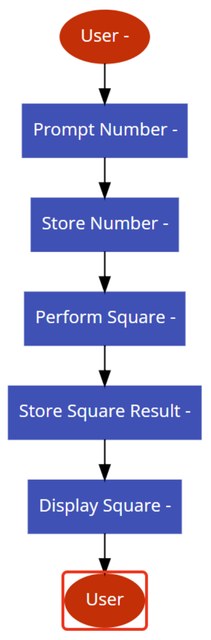
```
jmp continue
   continue:
   sub ecx , 1
   L1:
mul ebx
cmp ecx, 1
JE out1

   loop L1
   ans1:
mov eax, 1
   out1:
   mov edx, offset prompt16
       call writestring
   cmp val, 0
   JL divided2
   JNL continue2
   divided2:
mov edx,offset prompt17
call writestring
jmp continue2
   continue2:
   call writeint
call crlf
    call crlf
   ret
power endp

main PROC
mov edx, offset prompt0
call writestring
call crlf
reset:
```

```
mov edx, offset prompt4
call writestring
call readint
cmp eax, 1
JE sqrt
cmp eax, 2
JE sq
cmp eax, 3
JE CR
cmp eax, 4
JE Cub
cmp eax, 5
JE POW
cmp eax, 6
JE addi
cmp eax, 7
JE subt
cmp eax, 8
JE multi
cmp eax, 9
JE divi
cmp eax, 10
JE quit2
JNE wrong
sqrt:
call findSquareRoot
jmp reset

sq:
call square
jmp reset

CR:
```

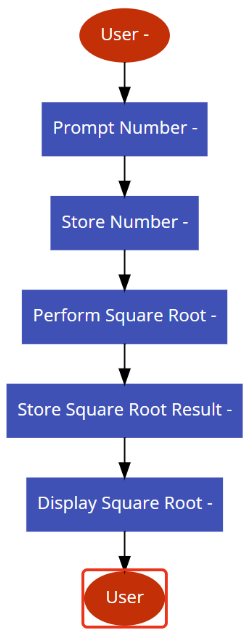
```
call FindCubeRoot
jmp reset

Cub:
call cube
jmp reset

POW:
call power
jmp reset

addi:
call Addition
jmp reset
subt:
call Subtraction
jmp reset
multi:
call Multiplication
jmp reset
divi:
call Division
jmp reset
wrong:
mov edx, offset prompt6
call writestring
call crlf
call crlf
jmp reset
quit2:
exit
main endp
end main
```

4

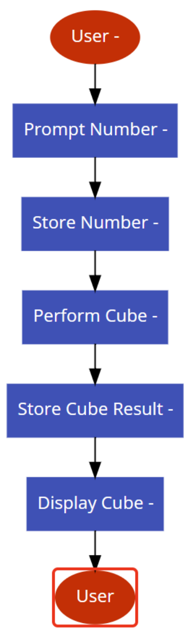
Data Flow Diagram



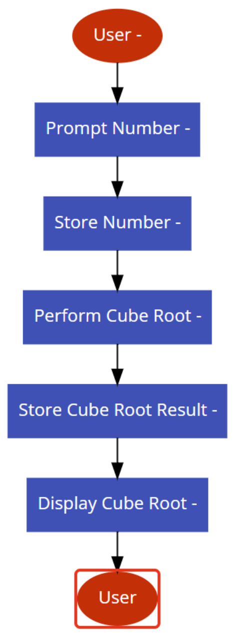
(a) Square



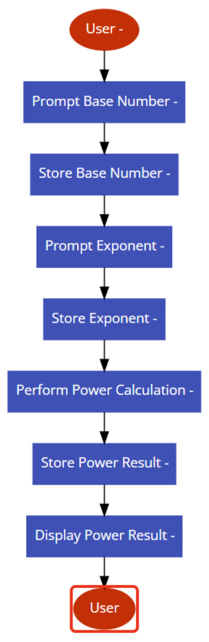
(b) Square Root



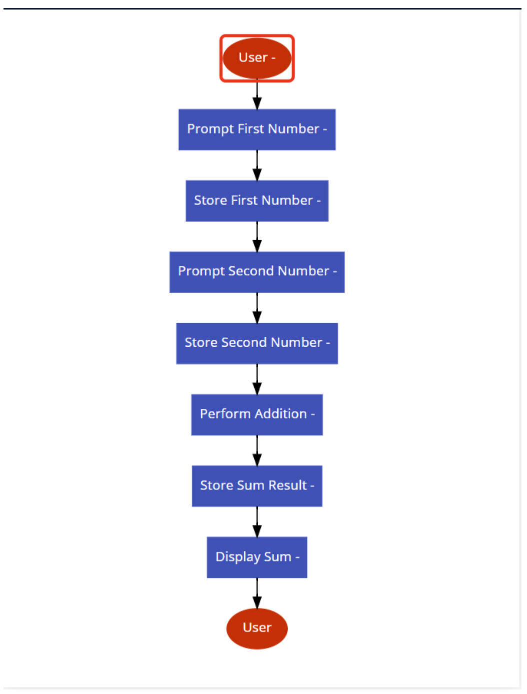
(c) Cube



(d) Cube Root

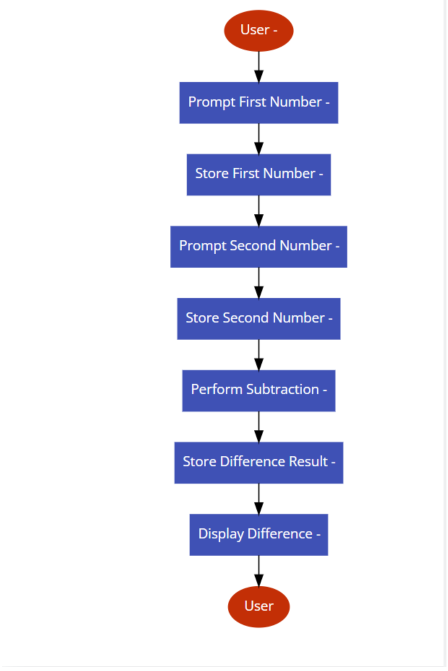


(e) Power

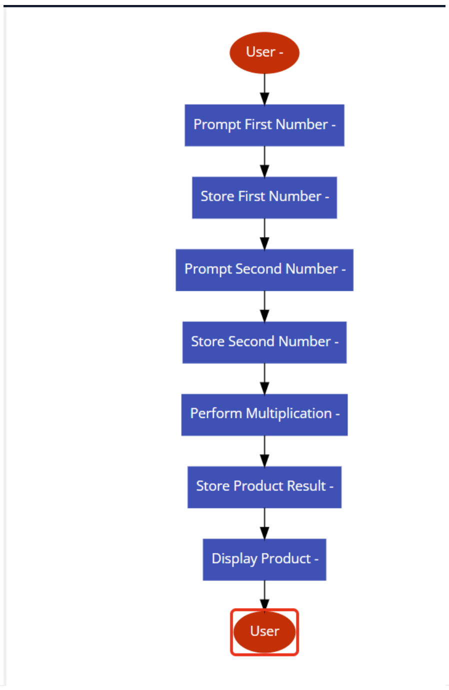


(f) Addition

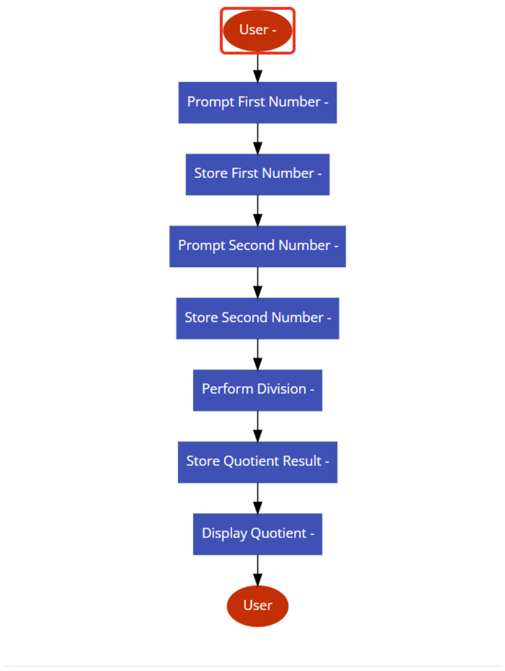
Figure 1: Data flow diagram for each procedure



(a) Subtraction



(b) Multiplication



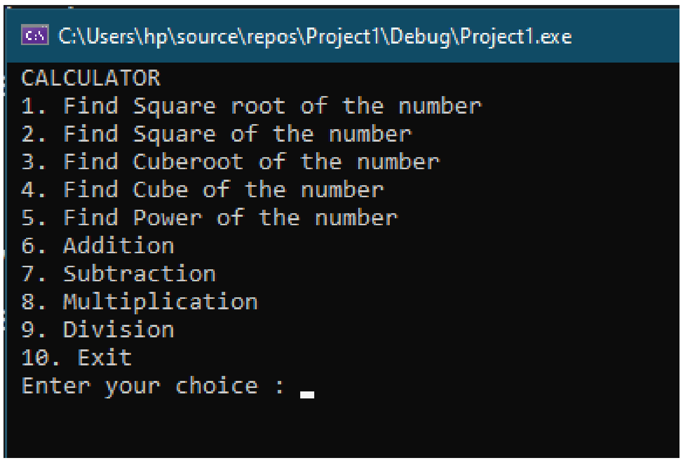
(c) Division

Figure 2: Data flow diagram for each procedure (cont.)

5 CODE EXPLANATION

5.1 MAIN MENU

5.1.1 Output



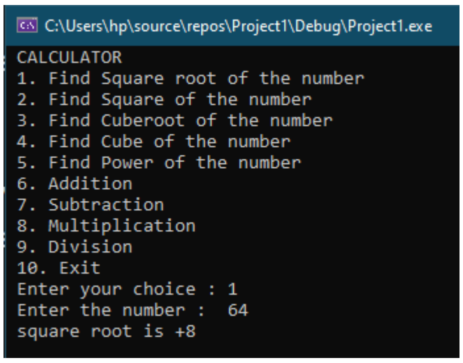
```
C:\Users\hp\source\repos\Project1\Debug\Project1.exe
CALCULATOR
1. Find Square root of the number
2. Find Square of the number
3. Find Cuberoot of the number
4. Find Cube of the number
5. Find Power of the number
6. Addition
7. Subtraction
8. Multiplication
9. Division
10. Exit
Enter your choice : _
```

Figure 3: **Main Menu**

5.2 Square Root Procedure

The Subtraction procedure allows the user to subtract one number from another. The detailed steps are:

- Display a prompt asking the user to enter the first number.
- Read the user's input and store it in a register.
- Check if the number is zero or one. If so, display the result.
- Otherwise, calculate the square root using a loop and compare method.
- If the number is not a perfect square, display an error message.
- Display the result of the calculation or an error if the number is not a perfect square.



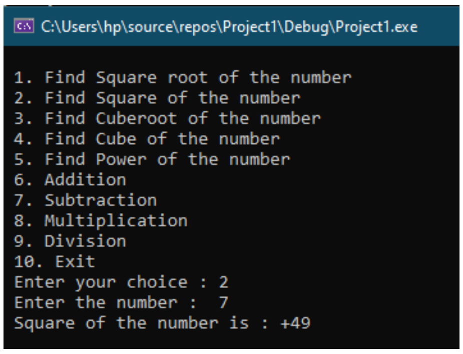
```
CALCULATOR
1. Find Square root of the number
2. Find Square of the number
3. Find Cuberoot of the number
4. Find Cube of the number
5. Find Power of the number
6. Addition
7. Subtraction
8. Multiplication
9. Division
10. Exit
Enter your choice : 1
Enter the number :  64
square root is +8
```

Figure 4: **Square Root Output**

5.3 Square Procedure

The Square procedure calculates the square of a given number entered by the user. The steps are:

- Display a prompt asking the user to enter a number.
- Read the user's input and store it in a register.
- Multiply the number by itself to calculate the square.
- Display the result of the calculation.



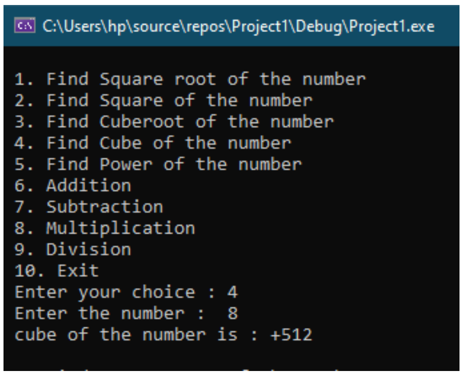
```
1. Find Square root of the number
2. Find Square of the number
3. Find Cuberoot of the number
4. Find Cube of the number
5. Find Power of the number
6. Addition
7. Subtraction
8. Multiplication
9. Division
10. Exit
Enter your choice : 2
Enter the number :  7
Square of the number is : +49
```

Figure 5: **Square Output**

5.4 CUBE PROCEDURE

The Cube procedure calculates the cube of a given number. The steps are:

- Display a prompt asking the user to enter the first number.
- Read the user's input and store it in a register.
- Multiply the number by itself twice to calculate the cube.
- Display the result of the calculation.



```
C:\Users\hp\source\repos\Project1\Debug\Project1.exe

1. Find Square root of the number
2. Find Square of the number
3. Find Cuberoot of the number
4. Find Cube of the number
5. Find Power of the number
6. Addition
7. Subtraction
8. Multiplication
9. Division
10. Exit
Enter your choice : 4
Enter the number :  8
cube of the number is : +512
```

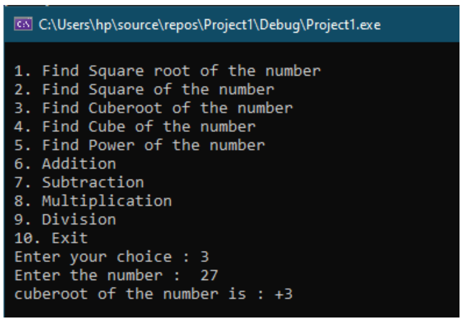
Figure 6: **Cube Output**

5.5 CUBE ROOT PROCEDURE

The Cube Root procedure calculates the cube root of a given number. The steps are:

- Display a prompt asking the user to enter the first number.
- Read the user's input and store it in a register.
- Check if the number is -1, 0, or 1. If so, display the result.
- Otherwise, calculate the cube root using iterative approximation.
- Display the result of the calculation, indicating if the cube root is a decimal.

5.5.1 Output



```
C:\Users\hp\source\repos\Project1\Debug\Project1.exe

1. Find Square root of the number
2. Find Square of the number
3. Find Cuberoot of the number
4. Find Cube of the number
5. Find Power of the number
6. Addition
7. Subtraction
8. Multiplication
9. Division
10. Exit
Enter your choice : 3
Enter the number :  27
cuberoot of the number is : +3
```

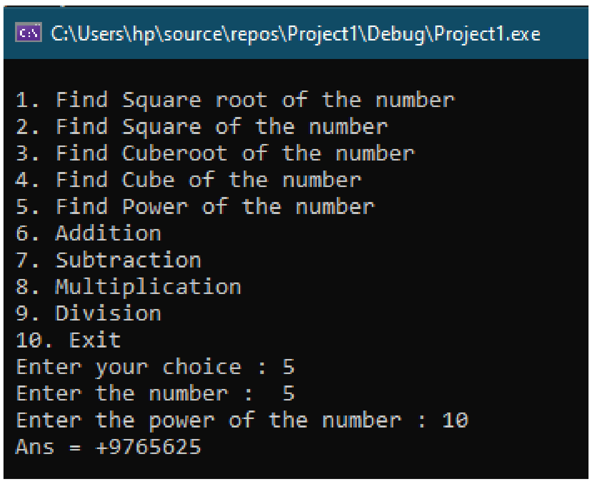
Figure 7: **Cube Root Output**

5.6 Power Procedure

The Power procedure calculates the power of a given number raised to a specified exponent. The steps are:

- Display a prompt asking the user to enter the first number.
- Read the user's input and store it in a register.
- Display a prompt asking the user to enter the exponent.
- Read the user's input and store it in another register.
- Calculate the power using a loop to multiply the base by itself the specified number of times.
- Display the result of the calculation.

5.6.1 Output



```
1. Find Square root of the number
2. Find Square of the number
3. Find Cuberoot of the number
4. Find Cube of the number
5. Find Power of the number
6. Addition
7. Subtraction
8. Multiplication
9. Division
10. Exit
Enter your choice : 5
Enter the number :  5
Enter the power of the number : 10
Ans = +9765625
```

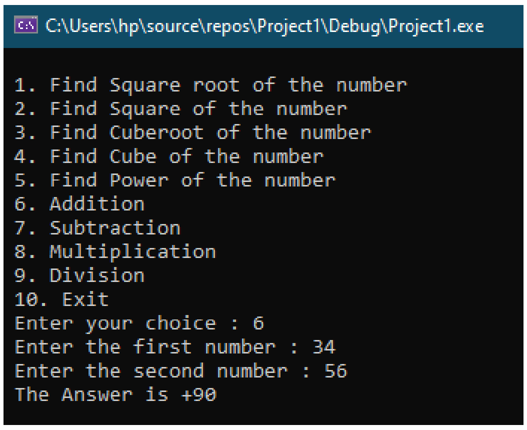
Figure 8: **Power Output**

5.7 Addition Procedure

The Addition procedure enables the user to perform the addition of two numbers. It involves the following steps:

- Display a prompt asking the user to enter the first number.
- Read the user's input and store it in a register.
- Read the user's input and store it in another register.
- Add the two numbers stored in the registers.
- Display the result of the addition.

5.7.1 Output



```
C:\Users\hp\source\repos\Project1\Debug\Project1.exe

1. Find Square root of the number
2. Find Square of the number
3. Find Cuberoot of the number
4. Find Cube of the number
5. Find Power of the number
6. Addition
7. Subtraction
8. Multiplication
9. Division
10. Exit
Enter your choice : 6
Enter the first number : 34
Enter the second number : 56
The Answer is +90
```

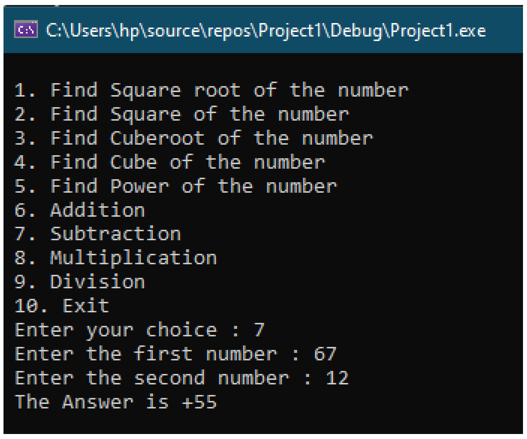
Figure 9: **Addition Output**

5.8 SUBTRACTION PROCEDURE

The Subtraction procedure allows the user to subtract one number from another. The detailed steps are:

- Display a prompt asking the user to enter the first number.
- Read the user's input and store it in a register.
- Display a prompt asking the user to enter the second number.
- Read the user's input and store it in another register.
- Subtract the second number from the first number.
- Display the result of the subtraction.

5.8.1 Output



```
C:\Users\hp\source\repos\Project1\Debug\Project1.exe

1. Find Square root of the number
2. Find Square of the number
3. Find Cuberoot of the number
4. Find Cube of the number
5. Find Power of the number
6. Addition
7. Subtraction
8. Multiplication
9. Division
10. Exit
Enter your choice : 7
Enter the first number : 67
Enter the second number : 12
The Answer is +55
```

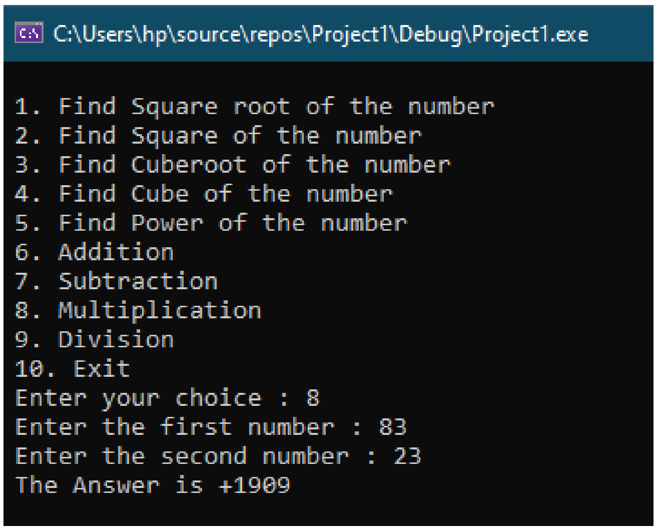
Figure 10: **Subtraction Output**

5.9 Multiplication Procedure

The Multiplication procedure performs the multiplication of two numbers entered by the user. The process is as follows:

- Display a prompt asking the user to enter the first number.
- Read the user's input and store it in a register.
- Display a prompt asking the user to enter the second number.
- Read the user's input and store it in another register.
- Multiply the two numbers stored in the registers.
- Display the result of the multiplication.

5.9.1 Output



```
C:\Users\hp\source\repos\Project1\Debug\Project1.exe

1. Find Square root of the number
2. Find Square of the number
3. Find Cuberoot of the number
4. Find Cube of the number
5. Find Power of the number
6. Addition
7. Subtraction
8. Multiplication
9. Division
10. Exit
Enter your choice : 8
Enter the first number : 83
Enter the second number : 23
The Answer is +1909
```

Figure 11: **Multiplication Output**

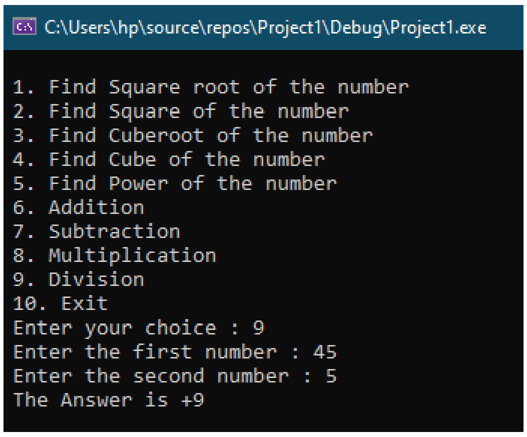
5.10 Division Procedure

The Division procedure handles the division of one number by another and includes error handling for division by zero. The steps are:

- Display a prompt asking the user to enter the first number.
- Read the user's input and store it in a register.
- Display a prompt asking the user to enter the second number.
- Read the user's input and store it in another register.
- Check if the second number is zero. If it is, display an error message and prompt for a new number.
- If the second number is not zero, perform the division.

- Display the result of the division.

5.10.1 Output



```
C:\Users\hp\source\repos\Project1\Debug\Project1.exe

1. Find Square root of the number
2. Find Square of the number
3. Find Cuberoot of the number
4. Find Cube of the number
5. Find Power of the number
6. Addition
7. Subtraction
8. Multiplication
9. Division
10. Exit
Enter your choice : 9
Enter the first number : 45
Enter the second number : 5
The Answer is +9
```

Figure 12: **Division Output**

5.11 Error Handling

Effective error handling is crucial for ensuring the reliability and usability of the calculator. The project will implement the following error handling mechanisms:

- Input Validation: Ensure that the user inputs valid numerical values. Display appropriate error messages for invalid inputs.
- Division by Zero: Handle division by zero errors gracefully by prompting the user to enter a valid divisor.
- Overflow and Underflow: Implement checks to detect and handle overflow and underflow conditions during calculations..
- Square Root of Negative Numbers: Provide meaningful error messages for invalid operations such as calculating the square root of negative numbers.
- General Errors: Catch and handle any unexpected errors to prevent the calculator from crashing.

6 Conclusion

This project report outlines the detailed plan for developing a multi-functional calculator using assembly language. The calculator will offer a wide range of mathematical functions with high accuracy and efficiency. The project schedule and team tasks are clearly defined to ensure successful completion of the calculator.