NEW YORK CITY COLLEGE OF TECHNOLOGY

THE CITY UNIVERSITY OF NEW YORK

300 JAY STREET, BROOKLYN, NY 11201-1909

Department of Computer Engineering Technology

CET 3510 - OL25 - Microcomputer Systems Technology lab

LABORATORY REPORT No_07

Lab_No_07 Integer Arthmetic Operations

Professor: Dr. Rubén Velásquez, Ph.D.

Spring 2021

Student's Name: Ashahi Shafin

Student's ID: 23607352

Preparation date: *April 07, 2021*

Due date: *April 14, 2021*

1. Table of content

2. OBJECTIVE

Laboratory No 7. Pratic arithmetic instructions for the 80x86, such as addition, subtraction, multiplication, division, and modulo. Write an lin line assembly language module in a C/C++ program by using general purpose registers. These are eight 32 bit registers that have the following names: EAX, EBX, ECX, EDX, ESI,EDI, EBP, and ESP.

3. LABORATORY TOOLS TO PERFORM THE TASK

Computer NZXT

Microsoft visual studio 2015 Software
Microsoft Word 2019 Software
Microsoft Notepad Software

4. SOURCE CODE

Laboratory No_07 Example 7.1

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <time.h>
#include "windows.h"
using namespace std;

void addition(short int x, short int y);
void subtraction(short int x, short int y);
void un_addition(unsigned short int x, unsigned short int y);
void un_subtraction(unsigned short int x, unsigned short int y);



int main()
{
	printf(" Lab_No_07_Integer_Arithmetic_Operations\n");
	printf(" Module: C++ progrmming \n");
	printf(" Ashahi Shafin, ID#23607352\n");
	printf(" CET3510-OL25\n");
	printf(" Presentation date: April 07, 2021\n");
	printf(" Due date: April 14, 2021\n");
	printf(" Example 7.1 arithmeticOperations.cpp\n");
	printf(" file name: 7.1 arithmeticOperations.cpp\n");
	printf("--------------------------------------------------------------------------
```

```
\n");

	char ch, ch1, ch2, ch3;
	signed short r1, r2;
	unsigned short ur1, ur2;
	cout << "Start your calculator Y/N, enter (Y) Yes or (N) No" << endl;
	cin >> ch;
	ch1 = ch;
	while (ch1 == 'Y' || ch1 == 'y')
	{
	Menu:
		cout << "Menu:\n";
		cout << "1 = Signed Integer Arithmetic Operation (16-bit)\n";
		cout << "2 = Unsigned Integer Arithmetic Operation (16-bit)\n";
		cout << "3 = Exit\n";
		cout << "Menu Options:\n";
		std::cin >> ch;
		ch2 = ch;

	Submenu:
		if (ch2 == '1')
		{
			cout << "Submenu - Input your choice\n";
			cout << "a = Input two 16-bit signed number operands for addition and
display\n"
				<< " the sum in decimal and hexadecimal format, respectively.\n";

			cout << "b = Input two 16-bit signed number operands for subtraction and
display\n"
				<< " the difference in decimal and hexadecimal format,
respectively.\n";
			cin >> ch;
			ch3 = ch;
			switch (ch3)
			{
			case'a':
			{
				cout << "Input two signed numbers operands in decimal format\n";
				cin >> r1 >> r2;
				addition(r1, r2);
				cout << "=====================================================\n";
				break;
			}
			case'b':
			{
				cout << "Input two signed numbers operands in decimal format\n";
				cin >> r1 >> r2;
				subtraction(r1, r2);
				cout << "=====================================================\n";
				break;
			}
			default: goto Menu;
			}
		}
		else if (ch2 == '2')
		{
			cout << "Submenu - input your choice\n";
			cout << "a, Input two 16-bit unsigned number operands for addition, and
display\n"
```

```
                        << "the sum in decimal and hexadecimal format respectively.\n";

                    cout << "b, Input two 16-bit unsigned number operands for subtraction and
display\n"
                        << "the sum in decimal and hexadecimal format, respectively.\n";
                    cin >> ch;
                    ch3 = ch;

                    switch (ch3)
                    {
                    case'a':
                    {

                        cout << "Input two unsigned numbers operands in decimal format\n";
                        cin >> ur1 >> ur2;
                        un_addition(ur1, ur2);
                        cout << "=====================================================\n";
                        break;
                    }
                    case'b':
                    {
                        cout << "Input two unsigned numbers operands in decimal format\n";
                        cin >> ur1 >> ur2;
                        un_subtraction(ur1, ur2);
                        cout << "=====================================================\n";
                        break;
                    }
                    default:goto Menu;
                    }
            }
            else
            {
                    goto EndLable;
            }
            cout << "Would you like to continue with the arithmetic operations (Y/N)? Enter
(Y) Yes or (N) No" << endl;
            cin >> ch;
            ch1 = ch;
    }

EndLable:
    cout << "Exit Program" << endl;
    system("pause");
    exit(0);
    return 0;
}

//addition for signed short integers
void addition(short int x, short int y)
{
    short int r;
    _asm
    {
            MOV AX, x;
            MOV BX, y;
            ADD AX, BX;
            MOV r, AX;
    }
    cout << "The decimal sum of " << dec << x << " and " << dec << y << " is " << dec << r <<
```

```
endl;
	cout << "The hexadecimal sum of " << hex << x << " and " << hex << y << " is " << hex <<
r << endl;
}

//subtraction for signed short integers
void subtraction(short int x, short int y)
{
	short int r;
	_asm
	{
		MOV AX, x;
		MOV BX, y;
		SUB AX, BX;
		MOV r, AX;
	}
	cout << "The decimal difference of " << dec << x << " minus " << dec << y << " is " <<
dec << r << endl;
	cout << "The hexadecimal difference of " << hex << x << " minus " << hex << y << " is "
<< hex << r << endl;
}

//addition for unsigned short integers
void un_addition(unsigned short int x, unsigned short int y)
{
	unsigned short int r;
	_asm
	{
		MOV AX, x;
		MOV BX, y;
		ADD AX, BX;
		MOV r, AX;
	}
	cout << "The decimal sum of " << dec << x << " and " << dec << y << " is " << dec << r <<
endl;
	cout << "The hexadecimal sum of " << hex << x << " and " << hex << y << " is " << hex <<
r << endl;
}

//subtraction for unsigned short integers
void un_subtraction(unsigned short int x, unsigned short int y)
{
	unsigned short int r;
	_asm
	{
		MOV AX, x;
		MOV BX, y;
		SUB AX, BX;
		MOV r, AX;
	}
	cout << "The decimal difference of " << dec << x << " minus " << dec << y << " is " <<
dec << r << endl;
	cout << "The hexadecimal difference of " << hex << x << " minus " << hex << y << " is "
<< hex << r << endl;
}
```

5. SOURCE CODE LINE DESCRIPTION

Laboratory No_07 Example 7.1

Line	Source Code Description
01	*#include "stdafx.h"*
Precompiled Header **stdafx**. **h** is basically used in Microsoft Visual Studio to let the compiler know the files that are once compiled and no need to compile it from scratch. ... **h**" before this **includes** then the compiler will find the compiled header files from **stdafx**. **h** and does not compiled it from scratch.	
02	*#include "stdio.h"*
The stdio.h (standard library header) is a file with ".h" extension that contains the prototypes of standard input-output functions used in c.	
03	#include<iostream>
h, **iostream** provides basic input and output services for C++ programs. **iostream uses** the objects cin , cout , cerr , and clog for sending data to and from the standard streams input, output, error (unbuffered), and log (buffered) respectively.	

04	int main(void)
	{
	MAIN PROGRAM
	}

In C and C++ **int main(void) means** that the function takes NO arguments. ... **Int main(void)** is used in C to restrict the function to take any arguments, if you do not put **void** in those brackets, the function will take ANY number of arguments you supply at call.	

05	`char ch, ch1, ch2, ch3;`
	`signed short r1, r2;`
	`unsigned short ur1, ur2;`

A **variable** is a name given to a memory location. It is the basic unit of storage in a program. The value stored in a **variable** can be changed during program execution.	
06	`cout << "Start your calculator Y/N, enter (Y) Yes or (N) No" << endl;`
The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout.	
07	`while (ch1 == 'Y' || ch1 == 'y')`
A **while** loop statement repeatedly executes a target statement as long as a given condition is true.	

08	`cout << "Menu:\n";`
	`cout << "1 = Signed Integer Arithmetic Operation (16-bit)\n";`
	`cout << "2 = Unsigned Integer Arithmetic Operation (16-bit)\n";`
	`cout << "3 = Exit\n";`

	`cout << "Menu Options:\n";`

The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout.

09	`if (ch2 == '1')`

The **if** statement allows you to control **if** a program enters a section of code or not based on whether a given condition is true or false. One of the important functions of the **if** statement is that it allows the program to select an action based upon the user's input.

10	`cout << "Submenu - Input your choice\n";`
	`cout << "a = Input two 16-bit signed number operands for addition and display\n"`
	`<< " the sum in decimal and hexadecimal format, respectively.\n";`
	`cout << "b = Input two 16-bit signed number operands for subtraction and display\n"`
	`<< " the difference in decimal and hexadecimal format, respectively.\n";`

The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout.

11	`switch (ch3)` `case'a':`

A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

12	`cout << "Input two signed numbers operands in decimal format\n";`
	`cin >> r1 >> r2;`
	`addition(r1, r2);`
	`cout << "=======================================================\n";`

The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout.

13	`break;`

The **break** in C or **C++** is a loop control statement which is used to terminate the loop. As soon as the **break** statement is encountered from within a loop, the loop iterations stops there and control returns from the loop immediately to the first statement after the loop.

14	`cout << "Input two signed numbers operands in decimal format\n";`
	`cin >> r1 >> r2;`
	`subtraction(r1, r2);`
	`cout << "=======================================================\n";`

The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout.

15	`break;`

The **break** in C or **C++** is a loop control statement which is used to terminate the loop. As soon as the **break** statement is encountered from within a loop, the loop iterations stops there and control returns from the loop immediately to the first statement after the loop.

16	`default: goto Menu;`

A **default** argument is a value provided in a function declaration that is automatically assigned by the compiler if the caller of the function doesn't provide a value for the argument with a **default** value.

17	`else if (ch2 == '2')`

An **if** statement can be followed by an optional **else** statement, which executes when the boolean expression is false.

18	`cout << "Submenu - input your choice\n";`
	`cout << "a, Input two 16-bit unsigned number operands for addition, and display\n"`
	`<< "the sum in decimal and hexadecimal format respectively.\n";`
	`cout << "b, Input two 16-bit unsigned number operands for subtraction and display\n"`
	`<< "the sum in decimal and hexadecimal format, respectively.\n";`

The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout.

19	`switch (ch3)`
	`{`
	`case'a':`

A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

20	`cout << "Input two unsigned numbers operands in decimal format\n";`
	`cin >> ur1 >> ur2;`
	`un_addition(ur1, ur2);`
	`cout << "=====================================================\n";`

The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout.

21	`break;`

The **break** in C or **C++** is a loop control statement which is used to terminate the loop. As soon as the **break** statement is encountered from within a loop, the loop iterations stops there and control returns from the loop immediately to the first statement after the loop.

22	`cout << "Input two unsigned numbers operands in decimal format\n";`
	`cin >> ur1 >> ur2;`
	`un_subtraction(ur1, ur2);`
	`cout << "=====================================================\n";`

The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout.

23	`break;`

The **break** in C or **C++** is a loop control statement which is used to terminate the loop. As soon as the **break** statement is encountered from within a loop, the loop iterations stops there and control returns from the loop immediately to the first statement after the loop.

24	`default:goto Menu;`

A **default** argument is a value provided in a function declaration that is automatically assigned by the compiler if the caller of the function doesn't provide a value for the argument with a **default** value.

25	`else`

Use if to specify a block of code to be executed, if a specified condition is true. Use **else** to specify a block of code to be executed, if the same condition is false. Use **else** if to specify a new condition to

test, if the first condition is false.

27	`cout << "Would you like to continue with the arithmetic operations (Y/N)? Enter (Y) Yes or (N) No" << endl;`

The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout.

28	`cout << "Exit Program" << endl;`

The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout.

29	`system("pause");`

The **system**() function is a part of the C/**C++** standard library. It is used to pass the commands that can be executed in the command processor or the terminal of the operating **system**, and finally returns the command after it has been completed.

30	`exit(0);`

The **exit** function, declared in <stdlib. h>, terminates a **C++** program.

31	`return 0;`

Terminates the execution of a function and **returns** control to the calling function (or to the operating system if you transfer control from the main function). Execution resumes in the calling function at the point immediately following the call.

32	`_asm { Defined in lins 33 to 35 }`

asm-declaration gives the ability to embed assembly language source code within a **C++** program. This declaration is conditionally-supported and implementation **defined**, **meaning** that it may not be present and, even when provided by the implementation, it does not have a fixed **meaning**.

33	`MOV AX, x;` `MOV BX, y;`

The mov instruction copies the data item referred to by its second operand (i.e. register contents, **memory** contents, or a constant **value**) into the location referred to by its first operand (i.e. a register or **memory**).

34	`ADD AX, BX;`

The add instruction add the data item referred to by its second operand (i.e. register contents, **memory** contents, or a constant **value**) into the location referred to by its first operand (i.e. a register or **memory**).

35	`MOV r, AX;`

The mov instruction copies the data item referred to by its second operand (i.e. register contents, **memory** contents, or a constant **value**) into the location referred to by its first operand (i.e. a register or **memory**).

36	`cout << "The decimal sum of " << dec << x << " and " << dec << y << " is " << dec << r << endl;` `cout << "The hexadecimal sum of " << hex << x << " and " << hex << y << " is " << hex << r << endl;`

The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout.

37	`_asm`
	`{`
	`Defined in lines 38 to 40 }`

asm-declaration gives the ability to embed assembly language source code within a **C++** program. This declaration is conditionally-supported and implementation **defined**, **meaning** that it may not be present and, even when provided by the implementation, it does not have a fixed **meaning**.

38	`MOV AX, x;` `MOV BX, y;`

The mov instruction copies the data item referred to by its second operand (i.e. register contents, **memory** contents, or a constant **value**) into the location referred to by its first operand (i.e. a register or **memory**).

39	`SUB AX, BX;`

The sub instruction subtract the data item referred to by its second operand (i.e. register contents, **memory** contents, or a constant **value**) into the location referred to by its first operand (i.e. a register or **memory**).

40	`MOV r, AX;`

The mov instruction copies the data item referred to by its second operand (i.e. register contents, **memory** contents, or a constant **value**) into the location referred to by its first operand (i.e. a register or **memory**).

41	`cout << "The decimal difference of " << dec << x << " minus " << dec << y << " is " << dec << r << endl;` `cout << "The hexadecimal difference of " << hex << x << " minus " << hex << y << " is " << hex << r << endl;`

The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout.

42	`_asm`
	`{`
	`Defined in lines 43 to 45 }`

asm-declaration gives the ability to embed assembly language source code within a **C++** program. This declaration is conditionally-supported and implementation **defined**, **meaning** that it may not be present and, even when provided by the implementation, it does not have a fixed **meaning**.

43	`MOV AX, x;` `MOV BX, y;`

The mov instruction copies the data item referred to by its second operand (i.e. register contents, **memory** contents, or a constant **value**) into the location referred to by its first operand (i.e. a

register or **memory**).

44	`ADD AX, BX;`

The add instruction add the data item referred to by its second operand (i.e. register contents, **memory** contents, or a constant **value**) into the location referred to by its first operand (i.e. a register or **memory**).

45	`MOV r, AX;`

The mov instruction copies the data item referred to by its second operand (i.e. register contents, **memory** contents, or a constant **value**) into the location referred to by its first operand (i.e. a register or **memory**).

46	`cout << "The decimal sum of " << dec << x << " and " << dec << y << " is " << dec << r << endl;` `cout << "The hexadecimal sum of " << hex << x << " and " << hex << y << " is " << hex << r << endl;`

The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard output device i.e. monitor. It is associated with the standard C output stream stdout.

47	`_asm`
	`{`
	`Defined in lines 48 to 50 }`

asm-declaration gives the ability to embed assembly language source code within a **C++** program. This declaration is conditionally-supported and implementation **defined**, **meaning** that it may not be present and, even when provided by the implementation, it does not have a fixed **meaning**.

48	`MOV AX, x;` `MOV BX, y;`

The mov instruction copies the data item referred to by its second operand (i.e. register contents, **memory** contents, or a constant **value**) into the location referred to by its first operand (i.e. a register or **memory**).

49	`SUB AX, BX;`

The sub instruction subtract the data item referred to by its second operand (i.e. register contents, **memory** contents, or a constant **value**) into the location referred to by its first operand (i.e. a register or **memory**).

50	`MOV r, AX;`

The mov instruction copies the data item referred to by its second operand (i.e. register contents, **memory** contents, or a constant **value**) into the location referred to by its first operand (i.e. a register or **memory**).

51	`cout << "The decimal difference of " << dec << x << " minus " << dec << y << " is " << dec << r << endl;` `cout << "The hexadecimal difference of " << hex << x << " minus " << hex << y << " is " << hex << r << endl;`

The **cout** object in **C++** is an object of class ostream. It is used to display the output to the standard

output device i.e. monitor. It is associated with the standard C output stream stdout.

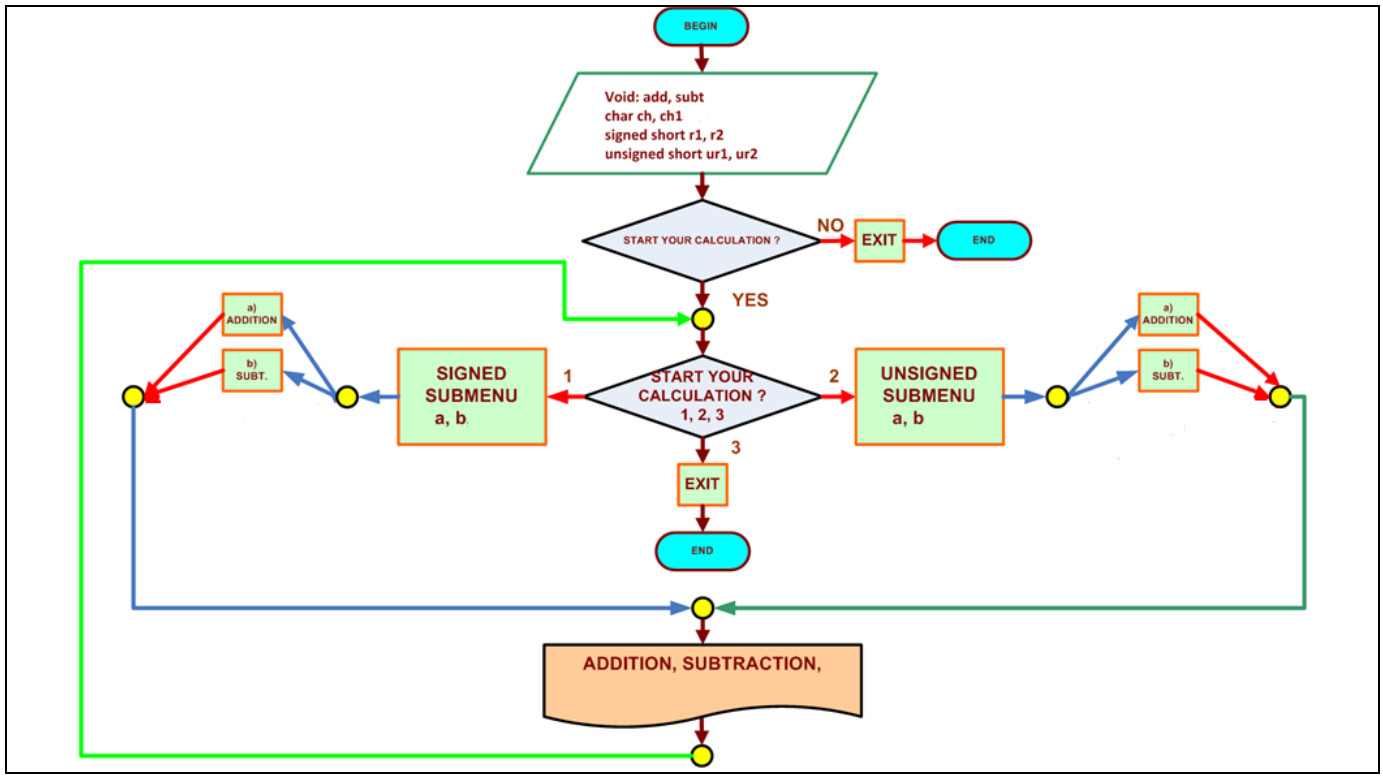
6. EXPLANING OUTPUT RESULTS

Laboratory No_07 Example 7.1

i. This is the output console of example 7.1 for laboratory 7. It shows the addition and subtraction in unsigned and signed for -36 and -5, 36 and 5. This is done with the register mov add and sub, then it displays it in the cout.

```
C:\WINDOWS\system32\cmd.exe
 Lab_No_07_Integer_Arithmetic_Operations
 Module: C++ progrmming
 Ashahi Shafin, ID#23607352
 CET3510-OL25
 Presentation date: April 07, 2021
 Due date: April 14, 2021
 Example 7.1 arithmeticOperations.cpp
 file name: 7.1 arithmeticOperations.cpp
--------------------------------------------------------------------------------
Start your calculator Y/N, enter (Y) Yes or (N) No
y
Menu:
1 = Signed Integer Arithmetic Operation (16-bit)
2 = Unsigned Integer Arithmetic Operation (16-bit)
3 = Exit
Menu Options:
1
Submenu - Input your choice
a = Input two 16-bit signed number operands for addition and display
 the sum in decimal and hexadecimal format, respectively.
b = Input two 16-bit signed number operands for subtraction and display
 the difference in decimal and hexadecimal format, respectively.
a
Input two signed numbers operands in decimal format
-36
-5
The decimal sum of -36 and -5 is -41
The hexadecimal sum of ffdc and fffb is ffd7
==================================================
Would you like to continue with the arithmetic operations (Y/N)? Enter (Y) Yes or (N) No
y
Menu:
1 = Signed Integer Arithmetic Operation (16-bit)
2 = Unsigned Integer Arithmetic Operation (16-bit)
3 = Exit
Menu Options:
1
Submenu - Input your choice
a = Input two 16-bit signed number operands for addition and display
 the sum in decimal and hexadecimal format, respectively.
b = Input two 16-bit signed number operands for subtraction and display
 the difference in decimal and hexadecimal format, respectively.
b
Input two signed numbers operands in decimal format
-36
-5
The decimal difference of -36 minus -5 is -31
The hexadecimal difference of ffdc minus fffb is ffe1
==================================================
Would you like to continue with the arithmetic operations (Y/N)? Enter (Y) Yes or (N) No
y
Menu:
1 = Signed Integer Arithmetic Operation (16-bit)
2 = Unsigned Integer Arithmetic Operation (16-bit)
3 = Exit
Menu Options:
2
Submenu - input your choice
a, Input two 16-bit unsigned number operands for addition, and display
the sum in decimal and hexadecimal format respectively.
b, Input two 16-bit unsigned number operands for subtraction and display
the sum in decimal and hexadecimal format, respectively.
a
Input two unsigned numbers operands in decimal format
36
5
The decimal sum of 36 and 5 is 41
The hexadecimal sum of 24 and 5 is 29
==================================================
Would you like to continue with the arithmetic operations (Y/N)? Enter (Y) Yes or (N) No
y
Menu:
1 = Signed Integer Arithmetic Operation (16-bit)
2 = Unsigned Integer Arithmetic Operation (16-bit)
3 = Exit
Menu Options:
2
Submenu - input your choice
a, Input two 16-bit unsigned number operands for addition, and display
the sum in decimal and hexadecimal format respectively.
b, Input two 16-bit unsigned number operands for subtraction and display
the sum in decimal and hexadecimal format, respectively.
b
Input two unsigned numbers operands in decimal format
36
5
The decimal difference of 36 minus 5 is 31
The hexadecimal difference of 24 minus 5 is 1f
==================================================
Would you like to continue with the arithmetic operations (Y/N)? Enter (Y) Yes or (N) No
y
Menu:
1 = Signed Integer Arithmetic Operation (16-bit)
2 = Unsigned Integer Arithmetic Operation (16-bit)
3 = Exit
Menu Options:
```

7. PROGRAM FLOW CHART:



8. COMMENTS

Comments on Outputs and Tables show is that when you pick the number it lets you add and sub the number depending on your choice which is useful the user if they want to pick a number.

9. CONCLUSION

In this experiment, the module shows that when you use 16-bit number and signed and unsigned numbers to add them and subtract them and make it so we can pick the numbers. The hardest part of the lab was when I

was making the code because I had a lot of error with cout which was a easy fix since my problem was I type count which isn't a c++ command.

A real-life application for laboratory No 7 would be in a computer because computer they are always moving files and sharing them and when this is happen they have transfer 16 at the same time and this very useful with servers because they need to transfer files and connect with each other's.