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_06

Lab_No_06 Addressing Modes

Professor: Dr. Rubén Velásquez, Ph.D.

Spring 2021

Student's Name: Ashahi Shafin

Student's ID: 23607352

Preparation date: *March 17, 2021*

Due date: *March 24, 2021*

1. Table of content

2. OBJECTIVE

Laboratory No 6. Explain unsigned and signed numbers, zero extension, sign extension, and variable declarations. The 80x86 provides several instructions that will let you sign or zero extend a number, for instance, an 8 bit binary number extending to a 16 bit binary number ,a 16 bit binary number extending to a 32 bit binary number, or a 8 bit binary number extending to a 32 bit binary number

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_06 Example 6.1

Source Code

```
nclude <iostream>
using namespace std;


int main()
{
	printf(" Lab_No_05_Addressing_Modes\n");
	printf(" Module: C++ progrmming \n");
	printf(" Ashahi Shafin, ID#23607352\n");
	printf(" CET3510-OL25\n");
	printf(" Presentation date: March 17, 2021\n");
	printf(" Due date: March 24, 2021\n");
	printf(" Example 6.1 extendingNum.cpp\n");
	printf(" file name: 6.1 extendingNum.cpp\n");
	printf("-------------------------------------------------------------------------------
\n");

	unsigned short int us16 = 0xfffd;
	signed short int s16 = 0xffffd;
	unsigned int us32;
	signed int s32;

	//A 16-bit UNSIGNED number extending to a 32-bit UNSIGNED number
	cout << "-----------------------------------------------------------\n";
```

```
	cout << "A 16-bit unsigned number extending to a 32-bit unsigned number\n";
	//Displays the unsigned hexadecimal value of 16-bit unsigned short int, us16
	cout << "unsigned hexadecimal value is " << hex << us16 << endl;
	//Displays the unsigned decimal value of 16-bit unsigned short int, us16
	cout << "its associated unsigned decimal value is " << dec << us16 << endl;;

	_asm
	{
		MOV AX, us16;
		MOVZX EAX, AX;
		MOV us32, EAX;
	}

	//Displays the 32-bit identical hexadecimal and decimal value of us16
	cout << "After zero extension,\n the identical hex value with 32-bit unsigned number is "
<< hex << us32 << endl
		<< " and its associated decimal is " << dec << us32 << endl;


	//A 16-bit SIGNED number extending to a 32-bit SIGNED number
	cout << "-----------------------------------------------------\n";
	cout << "A 16-bit signed number extending to a 32-bit signed number\n";
	//Displays the signed hexadecimal value of 16-bit signed short int, s16
	cout << "signed hexadecimal value is " << hex << s16 << endl;
	//Displays the signed decimal value of 16-bit signed short int, s16
	cout << "its associated signed decimal value is " << dec << s16 << endl;

	_asm
	{
		MOV BX, s16;
		MOVSX EBX, BX;
		MOV s32, EBX;
	}

	//Displays the 32-bit identical hexadecimal and decimal value of s16
	cout << "After sign extension,\n" "the identical hex value with 32-bit signed number is "
<< hex << s32 << '\n'
		<< " and its associated decimal is " << dec << s32 << '\n';

	system("pause");
	exit(0);
	return 0;
}
```

5. *SOURCE CODE LINE DESCRIPTION*

Laboratory No_06 Example 6.1

Line	*Source Code Description*
01	#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.

02

```
using namespace std;
```

is an abbreviation for standard. So that means we **use** all the things with in "**std**" **namespace**.

03

```
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.

04

```
unsigned short int us16 = 0xfffd;
signed short int s16 = 0xfffffd;
unsigned int us32;
signed int s32;
```

containers for storing data values. In C++, there are different types of **variables** (defined with different keywords), for example: int - stores integers (whole numbers), without decimals, such as 123 or -123. bool - stores values with two states: true or false.

05

```
cout << "-------------------------------------------------------\n";
cout << "A 16-bit unsigned number extending to a 32-bit unsigned number\n";
cout << "unsigned hexadecimal value is " << hex << us16 << endl;
cout << "its associated unsigned decimal value is " << dec << us16 << 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.

06

```
_asm
        {
         For mnemonic assembly instructions declaration
         explained in lines.
        }
```

_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.

07

```
MOV AX, us16;
MOVZX EAX, AX;
MOV us32, EAX;
```

Mnemonic MOV and ADD instruction loads an operand source 32-bit EAX register to an operand destination 32-bit ECX register.

08

```
cout << "After zero extension,\n the identical hex value with 32-bit unsigned number is "
<< hex << us32 << endl
<< " and its associated decimal is " << dec << us32 << endl;
```

	`cout << "-----------------------------------------------------\n";`
	`cout << "A 16-bit signed number extending to a 32-bit signed number\n";`
	`cout << "signed hexadecimal value is " << hex << s16 << endl;`
	`cout << "its associated signed decimal value is " << dec << s16 << 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.

09	`_asm`
	`{`
	For mnemonic assembly instructions declaration explained in lines.
	`}`

_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.

10	`MOV BX, s16;`
	`MOVSX EBX, BX;`
	`MOV s32, EBX;`

Mnemonic MOV and ADD instruction loads an operand source 32-bit EAX register to an operand destination 32-bit ECX register.

11	`cout << "After sign extension,\n" "the identical hex value with 32-bit signed number is "`
	`<< hex << s32 << '\n'`
	`<< " and its associated decimal is " << dec << s32 << '\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.

12	system("pause");

This is a Windows-specific command, which tells the OS to run the **pause** program. This program waits to be terminated, and halts the exceution of the parent **C++** program. Only after the **pause** program is terminated, will the original program continue.

13	exit(0);

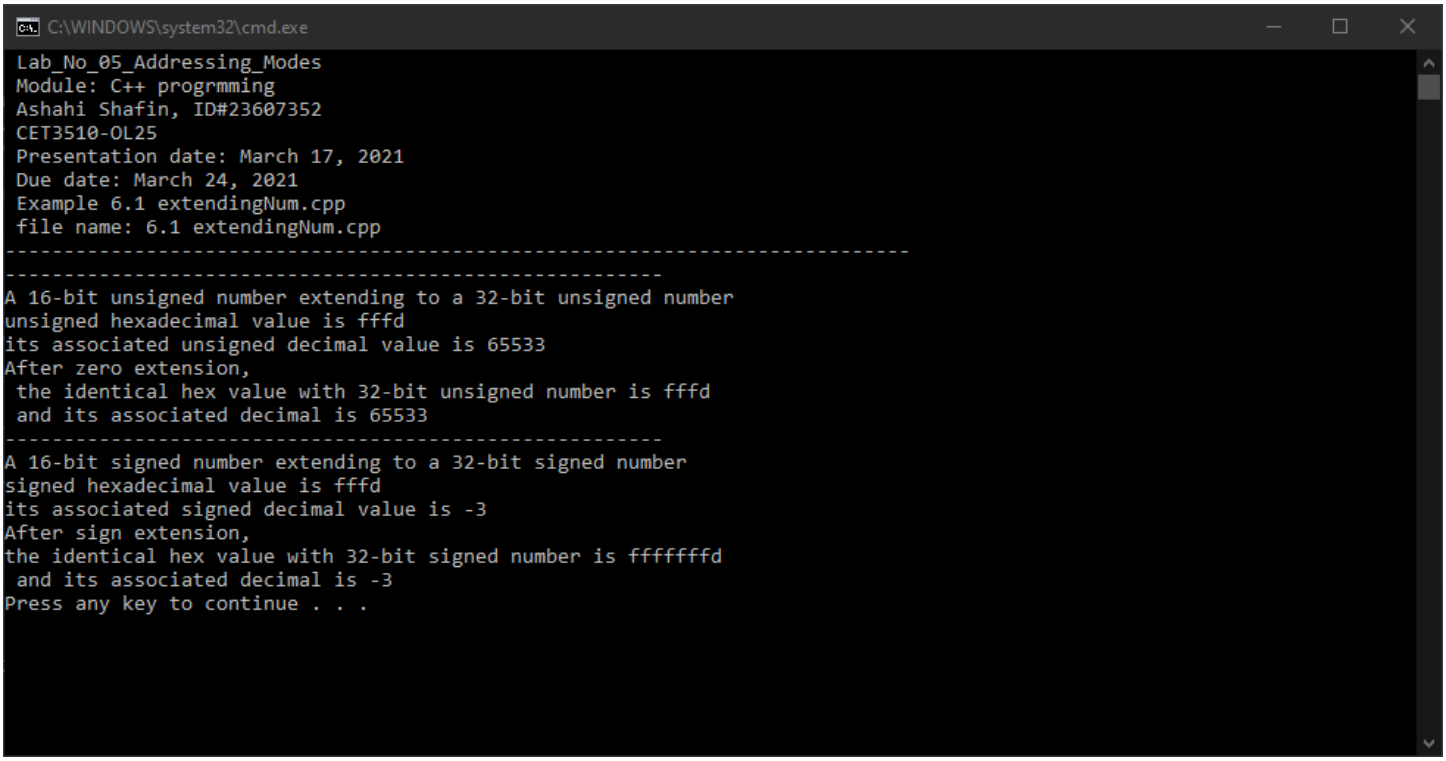
he value supplied as an argument to exit is returned to the operating system as the program's return code or exit code.

14	return 0;

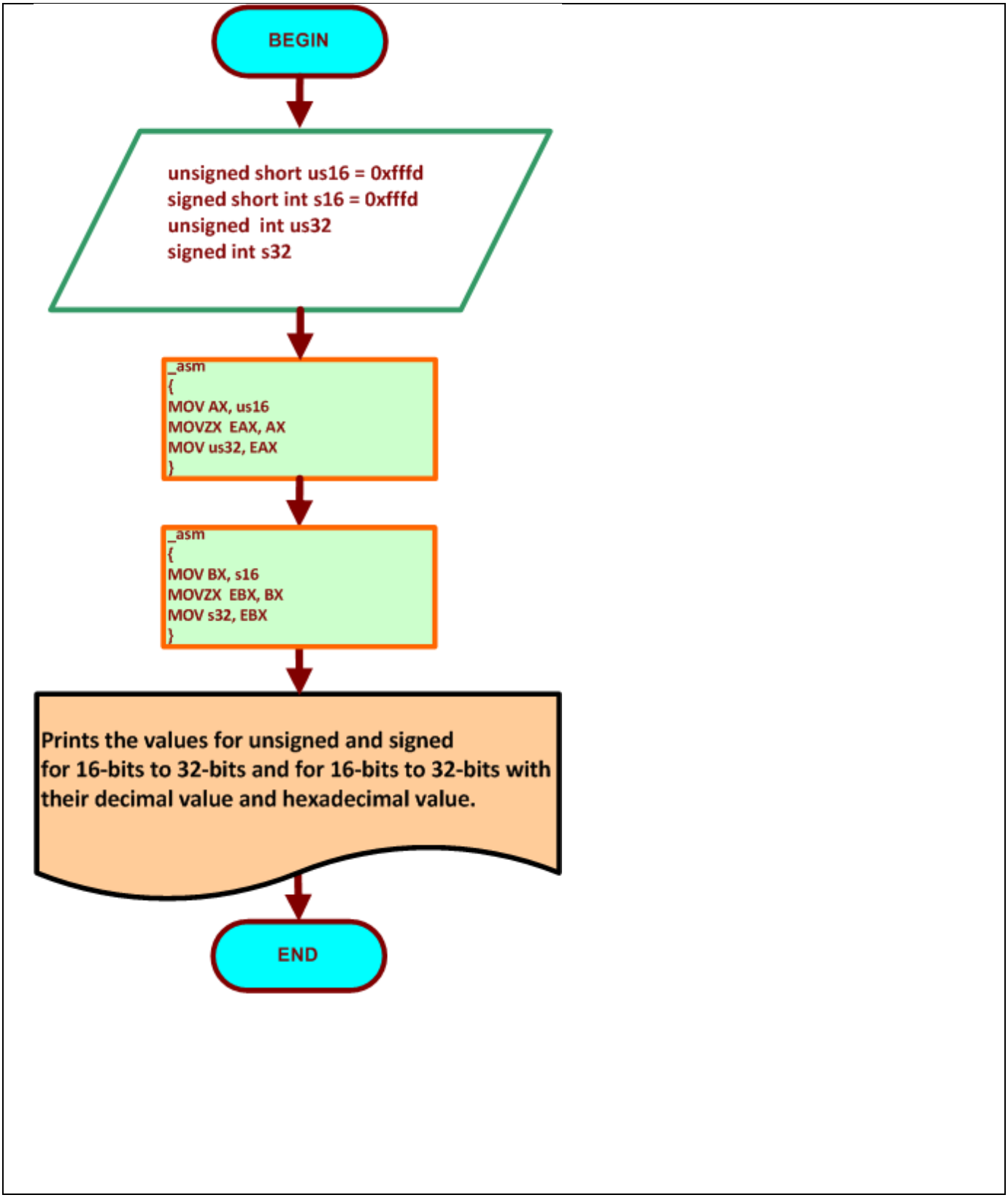
6. EXPLANING OUTPUT RESULTS

Laboratory No_06 Example 6.1

i. This is the output console of example 6.1 for laboratory 6. It shows the 32 bit in signed and unsigned number hex and decimal values after moving in the values to the us32 and s32.



7. PROGRAM FLOW CHART:


BEGIN
unsigned short us16 = 0xfffd
signed short int s16 = 0xfffd
unsigned int us32
signed int s32
_asm
{
MOV AX, us16
MOVZX EAX, AX
MOV us32, EAX
}
_asm
{
MOV BX, s16
MOVZX EBX, BX
MOV s32, EBX
}
Prints the values for unsigned and signed
for 16-bits to 32-bits and for 16-bits to 32-bits with
their decimal value and hexadecimal value.
END

8. COMMENTS

Comments on Outputs and Tables show is that it shows how 16-bit number become a 32 bit number and how 32 bit number become a 16 bit number.

9. CONCLUSION

In this experiment, the module shows that when you use 32 bits umber it can convert into 16 bit both in signed and unsigned. The same for 16 to 32 bits. They also show the hexadecimal value and decimal value too. What was hard for me in this lab was understanding the conversation to 16 to 32 and 32 to 16.

A real-life application for laboratory No 6 would be in a computer because computer they are always moving files and sharing them and when this is happen they have transfer 16 and 32 at the same time and this very useful with servers because they need to transfer files and connect with each others.