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_09

Lab_No_09 Bit Manipulation And Mask Design

Professor: Dr. Rubén Velásquez, Ph.D.

Spring 2021

Student's Name: Ashahi Shafin

Student's ID: 23607352

Preparation date: *April 21, 2021*

Due date: *April 28, 2021*

1. *Table of content*

2. OBJECTIVE

Laboratory No 9. Bit Manipulation involves the usage of several different Bitwise Operations. The Bitwise Operations that Bit Manipulation requires the usage of are The Bitwise AND Operation, The Bitwise OR Operation, The Bitwise XOR Operation, The Bitwise NOT Operation, The Bit Shift Left Operation, and The Bit Shift Right Operation. Different types of Bit Manipulation Instructions will be conducted in this Lab. The Bit Manipulation Instructions that will be conducted in this Lab are Setting Bits, Clearing Bits, Inverting Bits, Extracting Bits From A Bit String, and Inserting Bits into a Bit String. Mask Design will be conducted in this Lab as well. A Mask will be Designed to Clear, Set, and Inverted Selected Bits in a Bit String. The Bitwise Operations AND, OR, XOR, SHL, SHR, ROL, and ROR will be used for Packing Bit Strings and Unpacking Bit Strings. The CPU of an 80x86 Microprocessor will Overlay 64-Bit Registers with 32-Bit Registers, Overlay 32-Bit Registers with 16-Bit Registers, and Overlay 16-Bit Registers with 8-Bit Registers. The Structure of General Purpose Registers in an 80x86 Microprocessor will be used for Bit Manipulations. In-Line Assembly Programs will be created for Packing Bit Strings and Unpacking Bit Strings for which The Outputs of them will be displayed in Decimal Format, Hexadecimal Format, and Binary Bit String Format.

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_09 Example 9.1

Source Code

```
#include <stdio.h>
#include <iostream>
#include <bitset>
```

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



int main()
{
	printf("Lab_No_09 Bit_Manipulation_and_Mask_Desigen\n");
	printf("Module: C++ progrmming \n");
	printf("Ashahi Shafin, ID#23607352\n");
	printf("CET3510-OL25\n");
	printf("Presentation date: April 21, 2021\n");
	printf("Due date: April 28, 2021\n");
	printf("Example 9.1 BitsAndMaskApp.cpp\n");
	printf("file name: 9.1 BitsAndMaskApp.cpp\n");
	printf("-----------------------------------------------------------------------------
\n");

	unsigned short packedDate;
	unsigned char m;
	unsigned char d;
	unsigned char y;
	int count = 0;

	//randomize seed
	srand(time(0));

	//generate a random number between 1 and 12
	unsigned char month = (unsigned char)rand() % 12 + 1;

	//generate a random number between 1 and 31
	unsigned char day = (unsigned char)rand() % 31 + 1;

	//generate a random number between 0 and 255
	unsigned char year = (unsigned char)rand() & 0xff;

	//generate a number between 0 and 100
	while (year > 100 || year < 0)
	{
		year = (unsigned char)rand() & 0xff;
		count++;
	}
	cout << "The value of the loop counter to generate the year between 0 and 100: " << dec
<< count << endl;
	cout << "-----------------------------------------" << endl;
	cout << "The generated month, day, and year (in decimal format) are:\n";
	printf("%u\t %u\t %u\n", month, day, year);
	cout << "The generated month, day, and year (in hexadecimal format) are:\n";
	printf("0x%x\t 0x%x\t 0x%x\n", month, day, year);

	bitset<8> monthBits(month);
	cout << "month bits:\t" << monthBits << endl;
	bitset<8> dayBits(day);
	cout << "day bits:\t" << dayBits << endl;
	bitset<8> yearBits(year);
	cout << "year bits:\t" << yearBits << endl;
	_asm
	{
```

```
              mov BL, month;
              shl BX, 5;
              or BL, day;
              shl BX, 7;
              or BL, year;
              mov packedDate, BX;
       }
       cout << "-----------------------------------------" << endl; \
              cout << "The packed date in hexadecimal is \t0x" << hex << packedDate << endl;

       //convert packetDate to a 16-bit number to store in a bitset
       bitset<16> packetBits(packedDate);
       cout << "packed date:\t" << packetBits << endl;
       cout << "-----------------------------------------" << endl;
       _asm
       {
              mov AX, packedDate;
              and AX, 0xf000;
              rol AX, 4;
              mov m, AL;
              mov AX, packedDate;
              and AX, 0x0f80;
              ror AX, 7;
              mov d, AL;
              mov AX, packedDate;
              and AX, 0x007f;
              mov y, AL;
       }
       cout << "The retrieved month, day, and year from bit string (in decimal format) are:\n";
       printf("%u\t %u\t %u\n", month, day, year);
       cout << "The retrieved month, day, and year from bit string (in hexadecimal format)
are:\n";
       printf("0x%x\t 0x%x\t 0x%x\n", month, day, year);
       system("pause");
       exit(0);
}
```

5. *SOURCE CODE LINE DESCRIPTION*

Laboratory No_09 Example 9.1

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

```
unsigned short packedDate;
unsigned char m;
unsigned char d;
unsigned char y;
int count = 0;
unsigned char month = (unsigned char)rand() % 12 + 1;
unsigned char day = (unsigned char)rand() % 31 + 1;
unsigned char year = (unsigned char)rand() & 0xff;
```

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. A **variable** is only a name given to a memory location, all the operations done on the **variable** effects that memory location.

06

```
while (year > 100 || year < 0)
```

A **while** loop statement repeatedly executes a target statement as long as a given condition is true.

07

```
cout << "The value of the loop counter to generate the year between 0 and 100: " << dec
<< count << endl;
cout << "-----------------------------------------" << endl;
cout << "The generated month, day, and year (in decimal format) are:\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.

08

```
printf("%u\t %u\t %u\n", month, day, year);
```

Output console display information within parentheses.

09

```
cout << "The generated month, day, and year (in hexadecimal format) are:\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.

10

```
printf("0x%x\t 0x%x\t 0x%x\n", month, day, year);
```

Output console display information within parentheses.

11

```
bitset<8> monthBits(month);
```

Bitset represents a fixed-size sequence of N bits and stores values either 0 or 1. Zero **means** value is false or bit is unset and one **means** value is true or bit is set. ... **Bitset** class provides constructors to create **bitset** from integer as well as from strings. The size of the **bitset** is fixed at compile time.

12

```
cout << "month bits:\t" << monthBits << 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.

13

```
bitset<8> dayBits(day);
```

Bitset represents a fixed-size sequence of N bits and stores values either 0 or 1. Zero **means** value is false or bit is unset and one **means** value is true or bit is set. ... **Bitset** class provides constructors to create **bitset** from integer as well as from strings. The size of the **bitset** is fixed at compile time.

14

```
cout << "day bits:\t" << dayBits << 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.

15

```
bitset<8> yearBits(year);
```

Bitset represents a fixed-size sequence of N bits and stores values either 0 or 1. Zero **means** value is false or bit is unset and one **means** value is true or bit is set. ... **Bitset** class provides constructors to create **bitset** from integer as well as from strings. The size of the **bitset** is fixed at compile time.

16

```
cout << "year bits:\t" << yearBits << 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.

17

```
_asm
       {
        For mnemonic assembly instructions declaration
        explained in line 18.
       }
```

Output console display information within parenthesis.

18

```
mov BL, month;
shl BX, 5;
or BL, day;
shl BX, 7;
or BL, year;
mov packedDate, BX;
```

Mnemonic MOV instructions load an operand source memory hexadecimal value 0 to an operand destination 8-bit BL,BX registers to be initialized to zero.

19

```
cout << "----------------------------------------" << endl; \
cout << "The packed date in hexadecimal is \t0x" << hex << packedDate << 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.

20

```
bitset<16> packetBits(packedDate);
```

Bitset represents a fixed-size sequence of N bits and stores values either 0 or 1. Zero **means** value is false or bit is unset and one **means** value is true or bit is set. ... **Bitset** class provides constructors to create **bitset** from integer as well as from strings. The size of the **bitset** is fixed at compile time.

21

```
cout << "packed date:\t" << packetBits << endl;
cout << "------------------------------------------" << 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.

22

```
_asm
       {
        For mnemonic assembly instructions declaration
        explained in line 23.
       }
```

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

23

```
mov AX, packedDate;
and AX, 0xf000;
rol AX, 4;
mov m, AL;
mov AX, packedDate;
and AX, 0x0f80;
ror AX, 7;
mov d, AL;
mov AX, packedDate;
and AX, 0x007f;
mov y, AL;
```

Mnemonic MOV instructions load an operand source memory hexadecimal value 0 to an operand destination 16-bit AX, AL registers to be initialized to zero.

24

```
cout << "The retrieved month, day, and year from bit string (in decimal format) are:\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.

25

```
printf("%u\t %u\t %u\n", month, day, year);
```

Output console display information within parentheses.

27

```
cout << "The retrieved month, day, and year from bit string (in hexadecimal format)
are:\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.

28

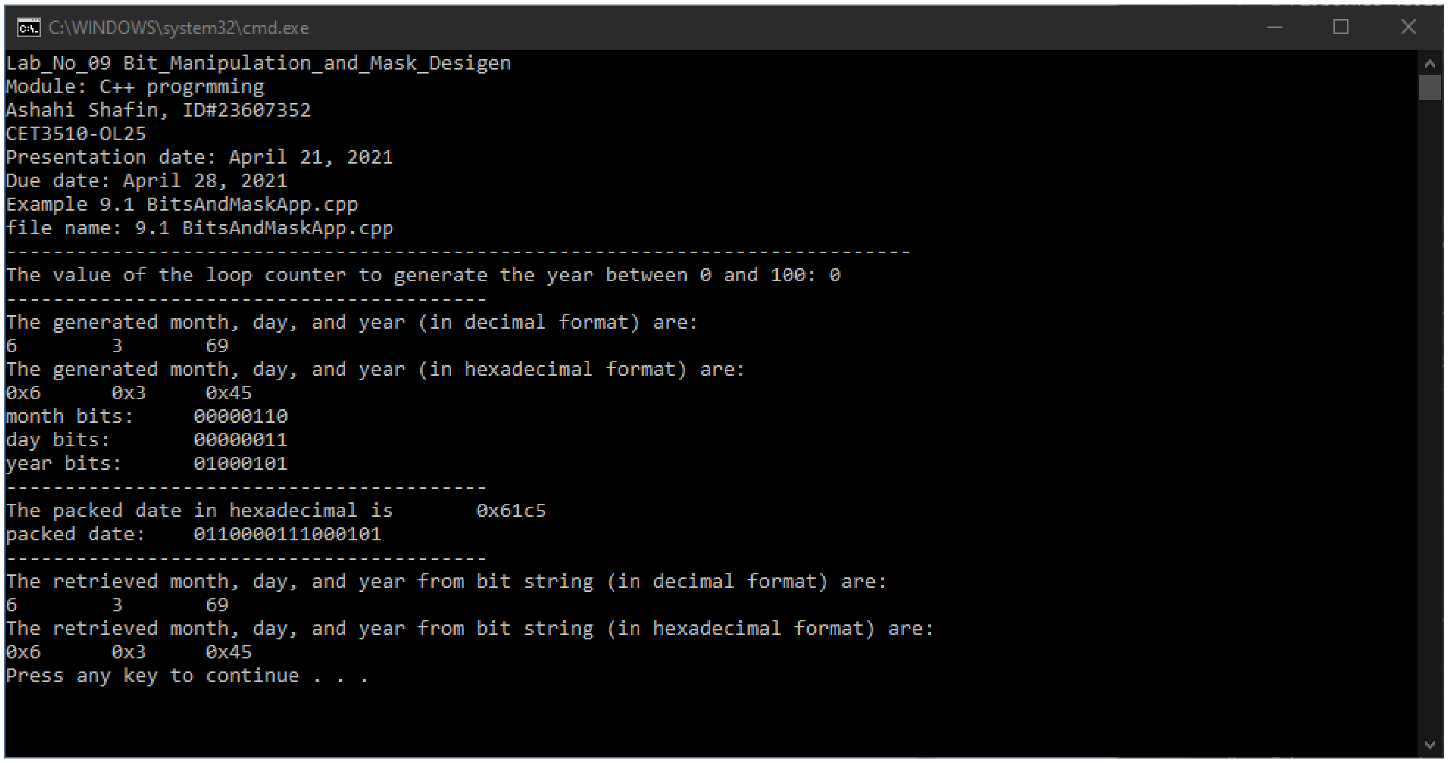
```
printf("0x%x\t 0x%x\t 0x%x\n", month, day, year);
```

Output console display information within parentheses.	
29	`system("pause");`
System("**pause**") runs the Windows command-line "**pause**" program and waits for that to terminate before it continues execution of the program , the console window stays open so you can read the output.	
30	`exit(0);`
The **exit** function, declared in <stdlib. h>, terminates a **C++** program.	

6. EXPLANING OUTPUT RESULTS

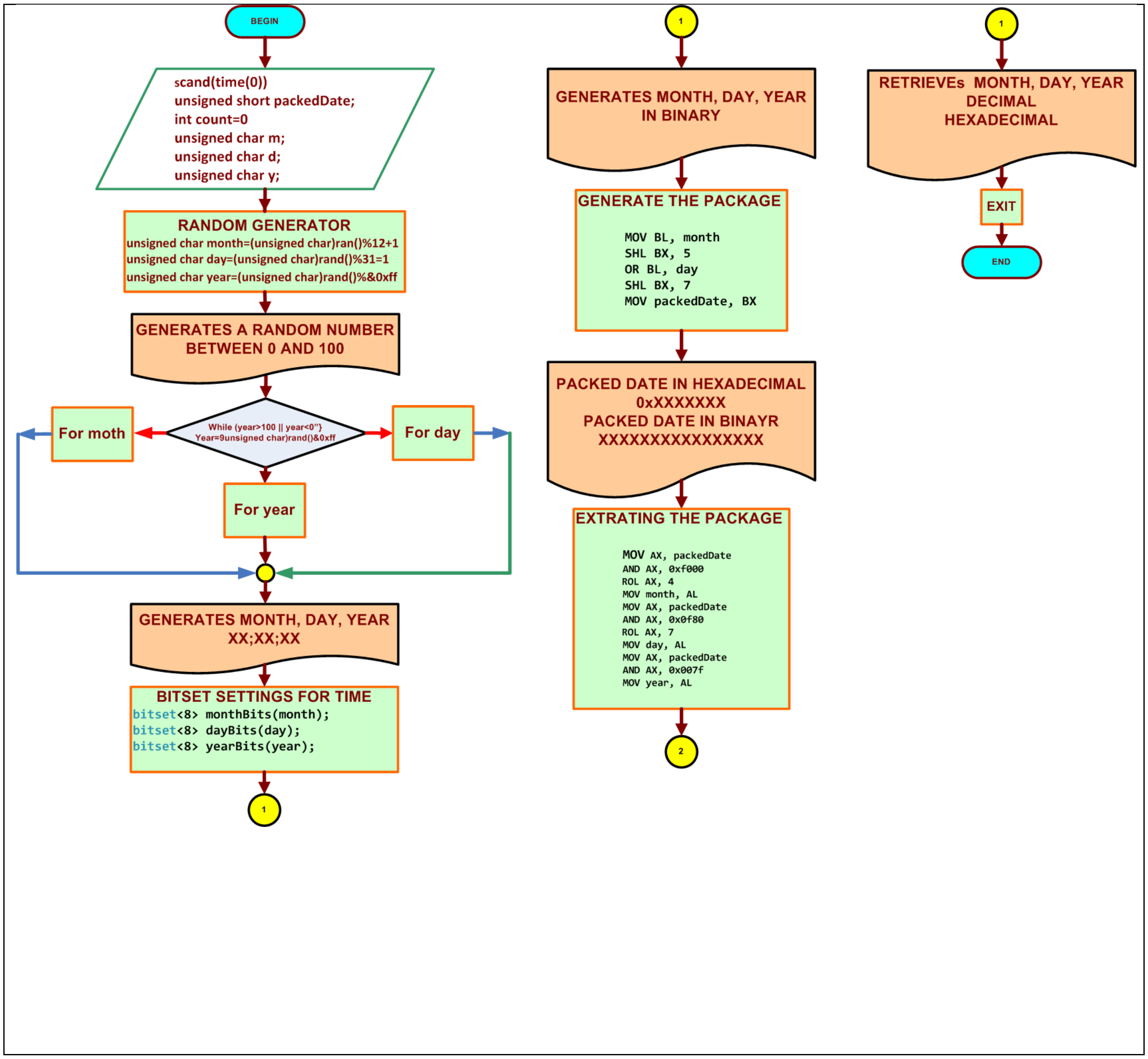
Laboratory No_09 Example 9.1

i. This is the output console of example 9.1 for laboratory 9. It shows the how the 8 bit and 16 bit loop worked with when it generate between 0 to 100 for the day month and year and then it show the hex value and decimal value. Then it does the same in 16 bit.



```
Lab_No_09 Bit_Manipulation_and_Mask_Desigen
Module: C++ progrmming
Ashahi Shafin, ID#23607352
CET3510-OL25
Presentation date: April 21, 2021
Due date: April 28, 2021
Example 9.1 BitsAndMaskApp.cpp
file name: 9.1 BitsAndMaskApp.cpp
-----------------------------------------------------------------------------
The value of the loop counter to generate the year between 0 and 100: 0
-----------------------------------------
The generated month, day, and year (in decimal format) are:
6        3       69
The generated month, day, and year (in hexadecimal format) are:
0x6      0x3     0x45
month bits:     00000110
day bits:       00000011
year bits:      01000101
-----------------------------------------
The packed date in hexadecimal is       0x61c5
packed date:    0110000111000101
-----------------------------------------
The retrieved month, day, and year from bit string (in decimal format) are:
6        3       69
The retrieved month, day, and year from bit string (in hexadecimal format) are:
0x6      0x3     0x45
Press any key to continue . . .
```

7. PROGRAM FLOW CHART:


BEGIN
RANDOM GENERATOR
GENERATES A RANDOM NUMBER BETWEEN 0 AND 100
For moth
For day
For year
GENERATES MONTH, DAY, YEAR XX;XX;XX
BITSET SETTINGS FOR TIME
1
1
GENERATES MONTH, DAY, YEAR IN BINARY
GENERATE THE PACKAGE
PACKED DATE IN HEXADECIMAL 0xXXXXXXX PACKED DATE IN BINAYR XXXXXXXXXXXXXXXX
EXTRATING THE PACKAGE
2
1
RETRIEVEs MONTH, DAY, YEAR DECIMAL HEXADECIMAL
EXIT
END

```
scand(time(0))
unsigned short packedDate;
int count=0
unsigned char m;
unsigned char d;
unsigned char y;
```

```
unsigned char month=(unsigned char)ran()%12+1
unsigned char day=(unsigned char)rand()%31=1
unsigned char year=(unsigned char)rand()%&0xff
```

```
While (year>100 || year<0"}
Year=9unsigned char)rand()&0xff
```

```
bitset<8> monthBits(month);
bitset<8> dayBits(day);
bitset<8> yearBits(year);
```

```
MOV BL, month
SHL BX, 5
OR BL, day
SHL BX, 7
MOV packedDate, BX
```

```
MOV AX, packedDate
AND AX, 0xf000
ROL AX, 4
MOV month, AL
MOV AX, packedDate
AND AX, 0x0f80
ROL AX, 7
MOV day, AL
MOV AX, packedDate
AND AX, 0x007f
MOV year, AL
```

8. COMMENTS

Comments on Outputs and Tables is that this can be used to find the month day and year and it hex and decimal value while it being generated and then it will do it in 16 bits instead of 8 bits.

9. CONCLUSION

In this experiment, the module shows that you can find the month day and year in hex and decimal value in 8 bits and 16 bits while it being generated. One thing that was hard for me is making the code because it took time to make and I had a lot of errors and took a lot of time to fix it but I was able to fix it and then the output work and got no errors.

A real-life application for laboratory No 9 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.