Lecture 02 :

Number System & Bitwise Operator

User Input

cin is a predefined variable that takes user input and store it in a variable.

```cpp
#include <iostream>

using namespace std;

int main()
{
    int x;
    cout<<"Enter the value you want to store \n ";
    cin>>x;

    cout<<"you have enter the value of x is "<<x;

    return 0;

}
```

Declare and initialize variable

Declare a variable means you are specifying the type of variable like integer , character , bool etc.

```
int x;          //declare x
int x1;         //declare x1
```

Initialize a variable means you are assigning value to that variable.

```
int x;          //declare x
x=10;           //initialize x
int y;          //declare y
y=10;           //initialize y
int z=10;       //declare and initialize at same time
```

Types of Initialization

1. Static Initialization : initialize the variable in the program

```
int x;          //declare x
x=10;           //static initialization of x
int x=10;       //static initialization of x
```

2. Dynamic Initialization : initialize the variable during running time

```
int a;
cin>>a;         //dynamic initialization of a
```

```cpp
#include <iostream>

using namespace std;

int main(){
    int x;      //declare x
    int y;      //declare y

    x=100;      //initialize x or static inialization
    y=200;      //initialize y or static inialization

    int z;      //declare z

    cout<<"Enter the value of z : ";
    cin>>z;
    cout<<z;

    return 0;
}
```

Memory Table

1Bit	=Binary Digit (0 or 1)
8 Bits	=1 Byte=2 nibble
1024 Bytes	=1 KB(Kilobyte)
1024 KB	=1 MB (Megabyte)
1024 MB	=1GB (Giga Byte)
1024 GB	=1TB (Tera Byte)
1024 TB	=1PB (Peta Byte)
1024 PB	=1EB (Exa Byte)
1024 EB	=1ZB (Zetta Byte)
1024 ZB	=1YB (Yotta Byte)
1024 YB	=1 (Bronto Byte)
1024 Brontobyte	=1 (Geop Byte)

Memory Of a Program

char = 1 bytes

short int = 2 bytes

int = 4 bytes

long int = 4 bytes

long long int = 8 bytes

float = 4 bytes

double = 8 bytes

Sizeof Operator

Sizeof is used to find the size of a datatype in a program

```
#include <iostream>

using namespace std;

int main(){
    int a;
    char b;
    float c;
    long int d;
    long long int e;
    double f;

    int size=sizeof(a);
    cout<<size;

    int size1=sizeof(b);
    cout<<size1;

    int size2=sizeof(c);
    cout<<size2;

    int size3=sizeof(d);
    cout<<size3;

    int size4=sizeof(e);
    cout<<size4;

    int size5=sizeof(f);
    cout<<size5;

    return 0;
}
```

Typecasting

Converting data type from one type to another is called as type casting in c++

1. Implicit Type Conversion : automatic conversion in c++ is called as implicit type conversion , compiler will automatically convert data type from one to another.
   Small data type → large data type

```cpp
#include <iostream>
using namespace std;

int main() {
   short x=20;
   int y = x;      //implicit type casting

   cout<<x;
   cout<<y;

   return 0;
}
```

2. Explicit Type Conversion : when user convert from large data type to small data type
   Large data type → small data type

```cpp
#include <iostream>
using namespace std;

int main() {
   double x=1.2;
   int y = (int)x;       //explicit type casting
   cout<<y;
   return 0;
}
```

Scope of a variable : block of code where we can access the variable

```
#include <iostream>

using namespace std;

int main() {

   int x;
   return 0;

}
```

Local and global variable : local variable can only be access in a particular code of block where it is declare while global variable can be used in program anywhere

```
#include <iostream>
int y=10;
using namespace std;

int main() {

   int x;        //local variable
   cout<<y;      //global variable
   return 0;

}
```

Number System

Decimal and binary

Decimal number are numerical value like 1,2,3,4,5,6 etc

And binary are 0 and 1

2 = 10

5 = 101

10 = 1010

20 = 10100

50 = 110010

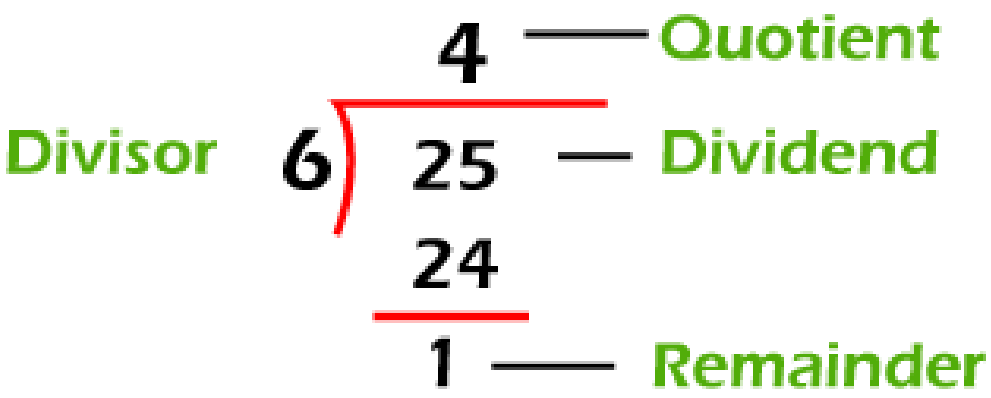
100 = 1100100

200 = 11001000

500 = 111110100

1000 = 1111101000

Dividend , Remainder , Quotient and divisor in division



Decimal to Binary Conversion

Take input as n

Divide n by 2 and write remainder and quotient as following

n / 2 = quotient (remainder)

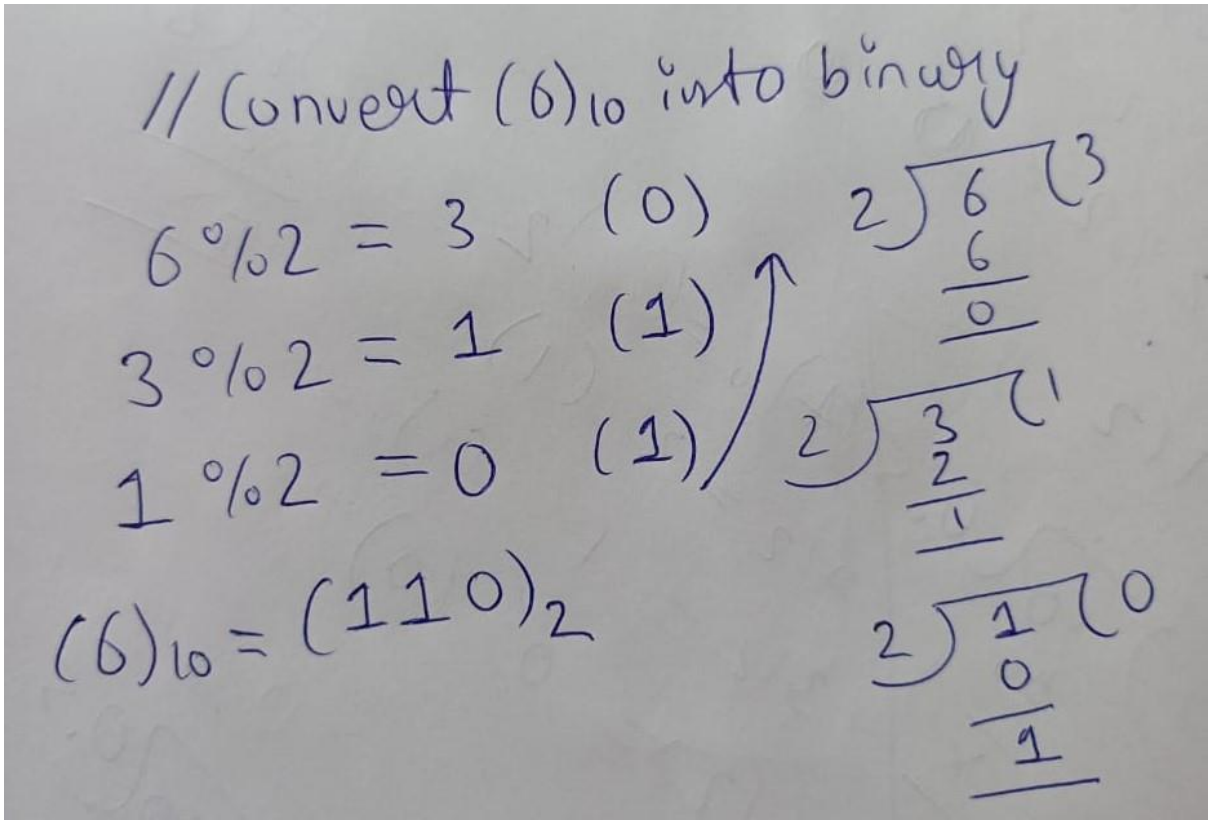
quotient / 2 = quotient1 (remainder)

quotient1 / 2 = quotient3 (remainder)

quotient3 / 2 = 0 (remainder)

stop when n becomes 0

then write from bottom to top as binary number



$(13)_{10} = (1101)_2$

Decimal Binary

13 % 2 = 6 (1)

6 % 2 = 3 (0)

3 % 2 = 1 (1)

1 % 2 = 0 (1)

~~13 (Bi~~

13 (decimal) = 1101 (binary)

// Convert $(8)_{10}$ into binary

8 % 2 = 4 (0)

4 % 2 = 2 (0)

2 % 2 = 1 (0)

1 % 2 = 0 (1)

$(18)_{10} = (1000)_2$

// Convert 5 into binary

$(5)_{10}$

5 % 2 = 2 (1)

2 % 2 = 1 (0)

1 % 2 = 0 (1)

$(5)_{10} = (101)_2$

// Convert $(20)_{10}$ into binary

20 % 2 = 10 (0)

10 % 2 = 5 (0)

5 % 2 = 2 (1)

2 % 2 = 1 (0)

1 % 2 = 0 (1)

$(20)_{10} = (10100)_2$

Binary to Decimal Conversion

Write the binary code with some space

Now start writing 2 below all of them

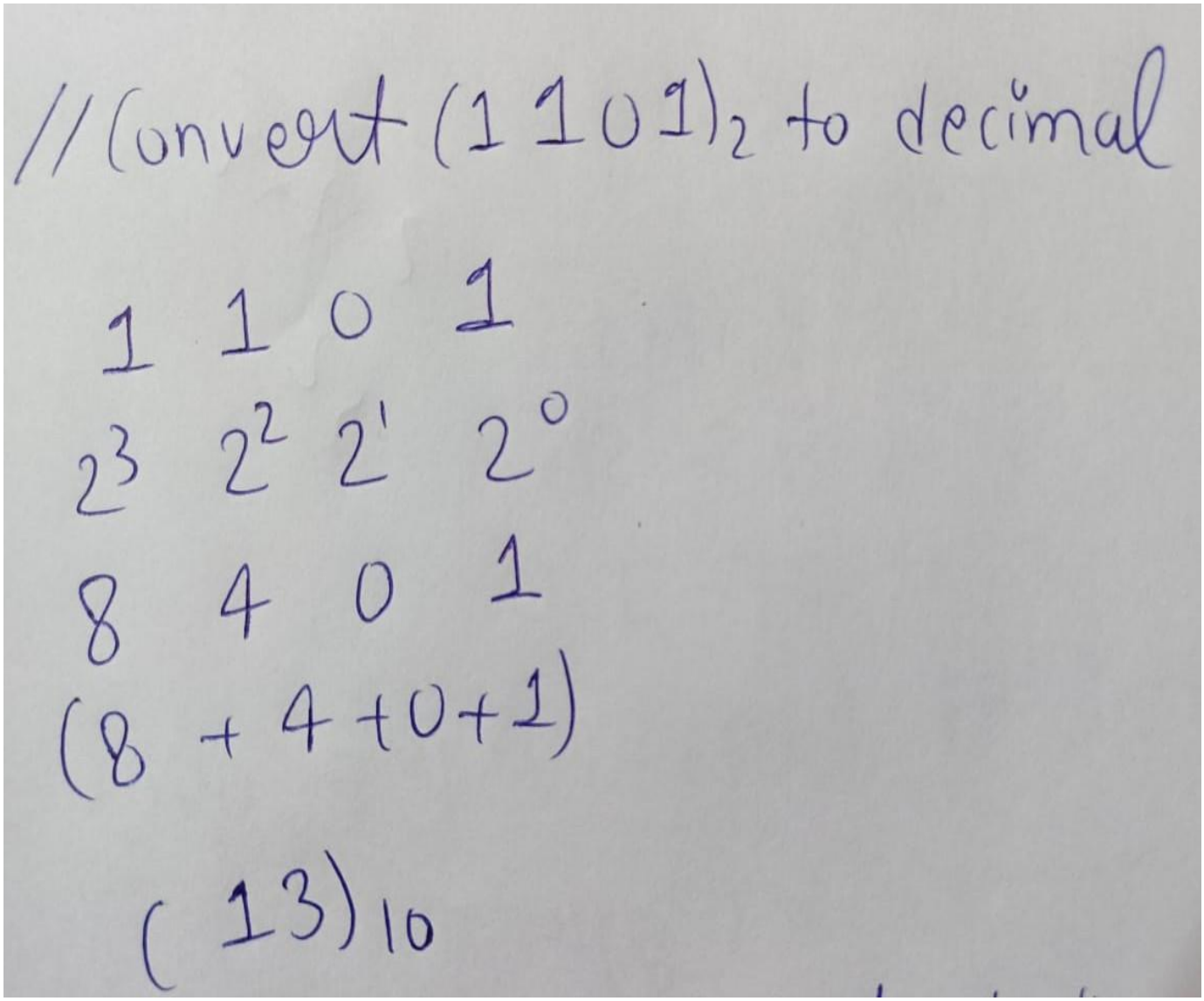
Then write power of 2 from 0

Move from left to right when writing power

Count 0 binary as 0

And add remaining

You will get the decimal number



// Convert $(1000)_2$ into decimal

1 0 0 0

2^3 2^2 2^1 2^0

8 + 0 + 0 + 0

$(8)_{10}$

$(1000)_2 = (8)_{10}$

// Convert $(1010\ 10)_2$ into decimal

1 0 1 0 1 0

2^5 2^4 2^3 2^2 2^1 2^0

32 + 0 + 8 0 + 2 0

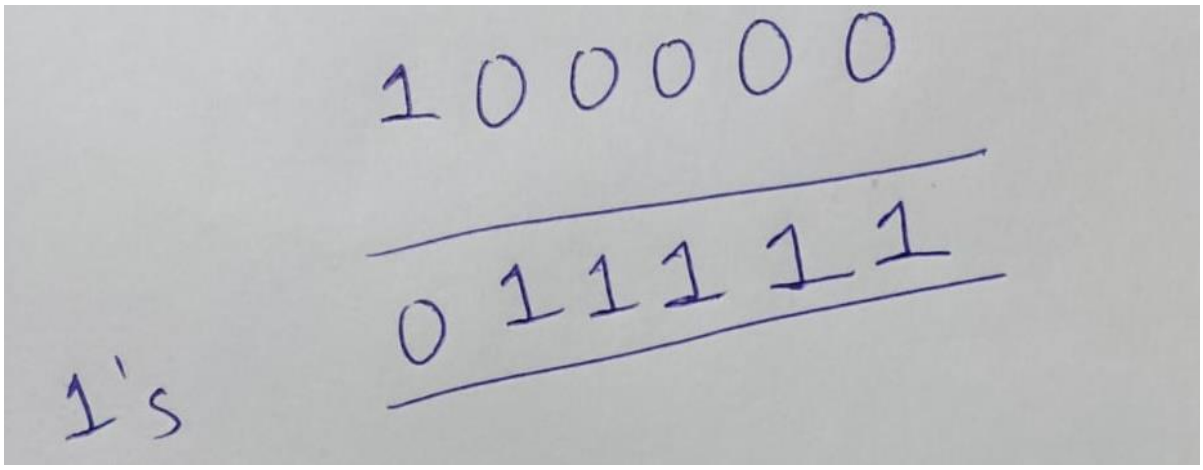
$(42)_{10}$

$(101010)_2 = (42)_{10}$

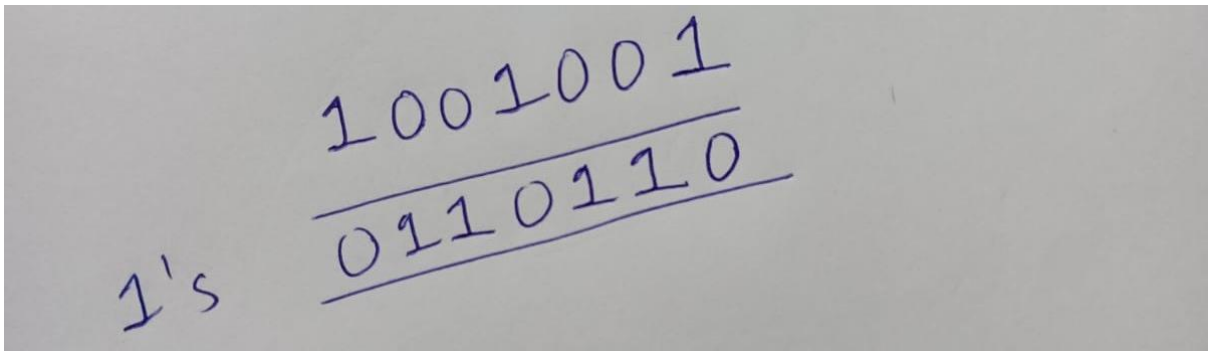
One's & Two's Complement

For 1's complement : reverse the digits

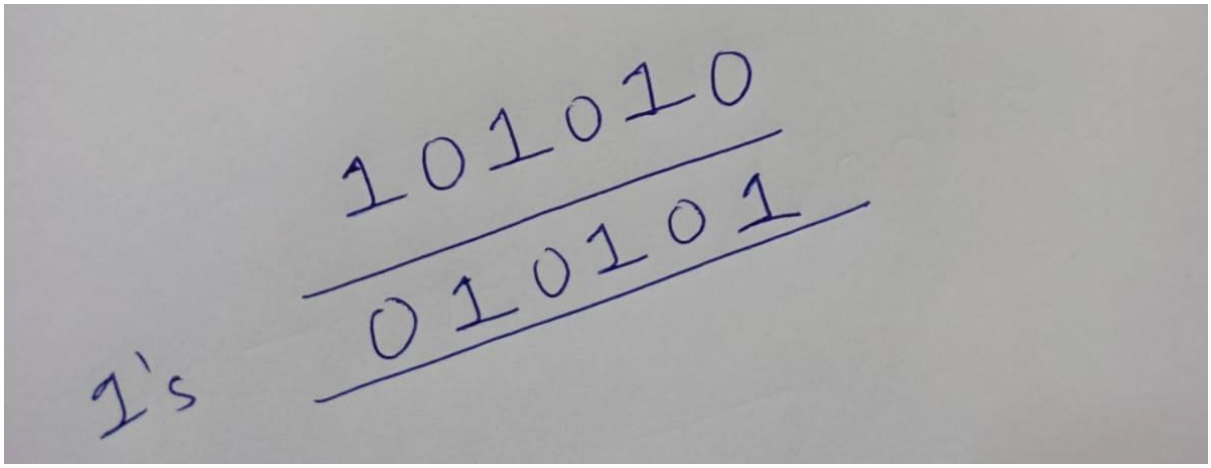
//find 1's complement of 100000



//find 1's complement of 1001001



//find 1's complement of 101010

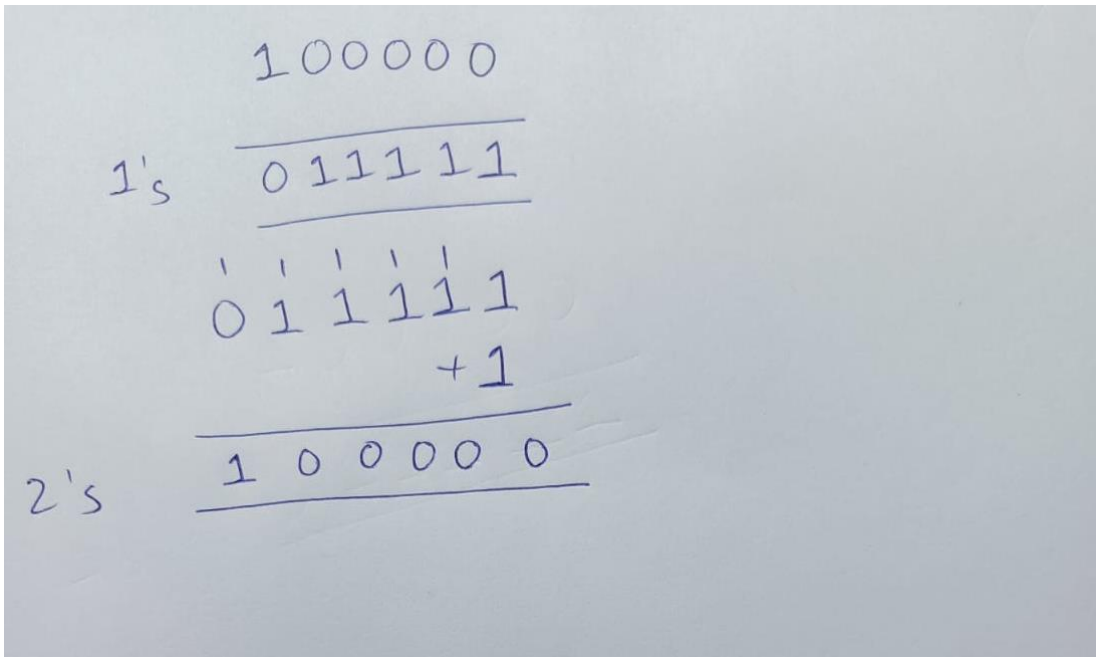


For 2's complement : add 1 to 1's complement

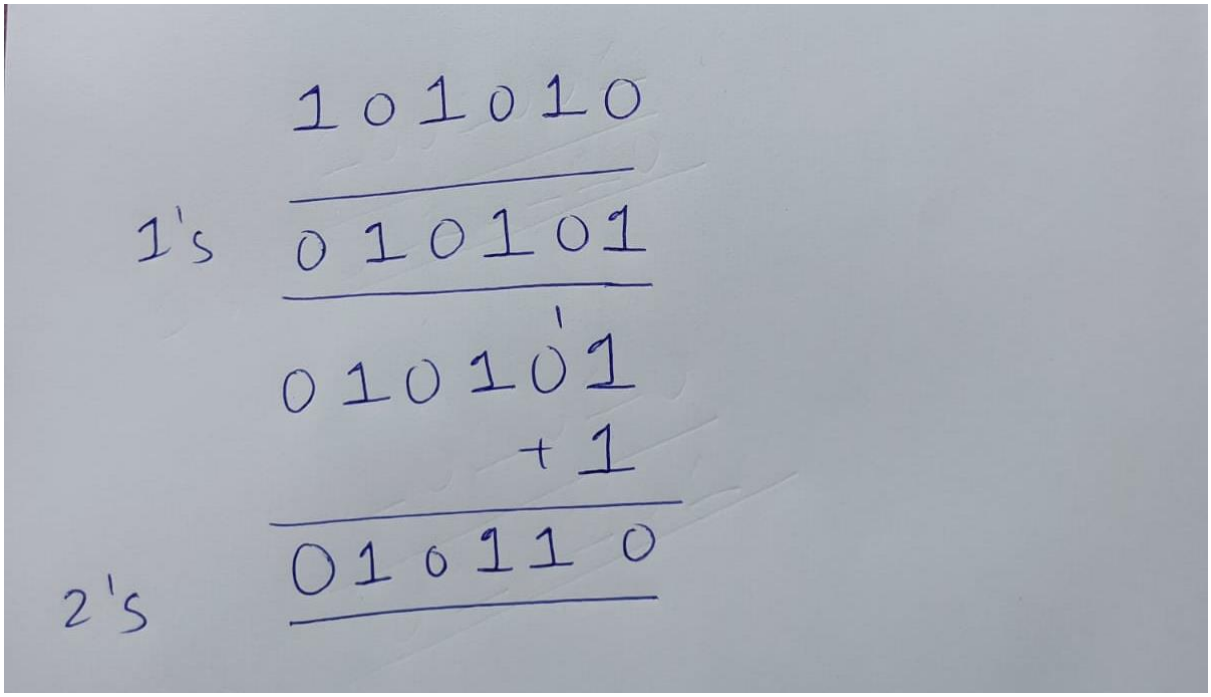
In binary

0 + 0 = 0 , 1 + 0 = 0 , 0 + 1 = 0 , 1 + 1 = 10

//find 1's and 2's complement of 100000



//find 1's and 2's complement of 101010



Decimal to binary or binary to decimal conversion website :

https://www.rapidtables.com/convert/number/decimal-to-binary.html

Complement calculator

https://ncalculators.com/digital-computation/1s-2s-complement-calculator.htm

Bitwise Operator

bitwise operators perform operations on integer data at the individual bit-level

1.bitwise AND	**&**
2.bitwise OR	**|**
3.bitwise XOR	**^**
4.bitwise complement	**~**
5.bitwise left shift	**<<**
6.bitwise right shift	**>>**

1.Bitwise AND operator (&)

Return 1 if and only if both operator are 1

Otherwise it will return 0

For example : a&b

a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

```cpp
#include <iostream>
using namespace std;

int main() {
    // declare variables
    int a = 12, b = 25;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "a & b = " << (a & b) << endl;

    return 0;
}
```

2.bitwise OR operator (|)

Return 1 if any one of two operator are 1

Otherwise it will return 0

For example : a|b

```cpp
#include <iostream>

int main() {
    int a = 12, b = 25;
```

```
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "a | b = " << (a | b) << endl;

    return 0;
}
```

3.bitwise XOR operator (^)

returns 1 if and only if one of the operands is 1. However, if both the operands are 0, or if both are 1, then the result is 0.

For example : a^b

```
#include <iostream>

int main() {
    int a = 12, b = 25;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "a ^ b = " << (a ^ b) << endl;

    return 0;
}
```

4.bitwise complement operator (~)

The bitwise complement operator is a unary operator (works on only one operand). It is denoted by ~ that changes binary digits 1 to 0 and 0 to 1.

For example : ~a

```
#include <iostream>

int main() {
    int num1 = 35;
    int num2 = -150;
    cout << "~(" << num1 << ") = " << (~num1) <<
endl;
    cout << "~(" << num2 << ") = " << (~num2) <<
endl;

    return 0;
}
```

5.bitwise left shift operator (<<)

The right shift operator shifts all bits towards the right by a certain number of specified bits.

For example : a<<1

6.bitwise right sift operator (>>)

The left shift operator shifts all bits towards the left by a certain number of specified bits.

For example : a>>1