BITWISE OPERATORS

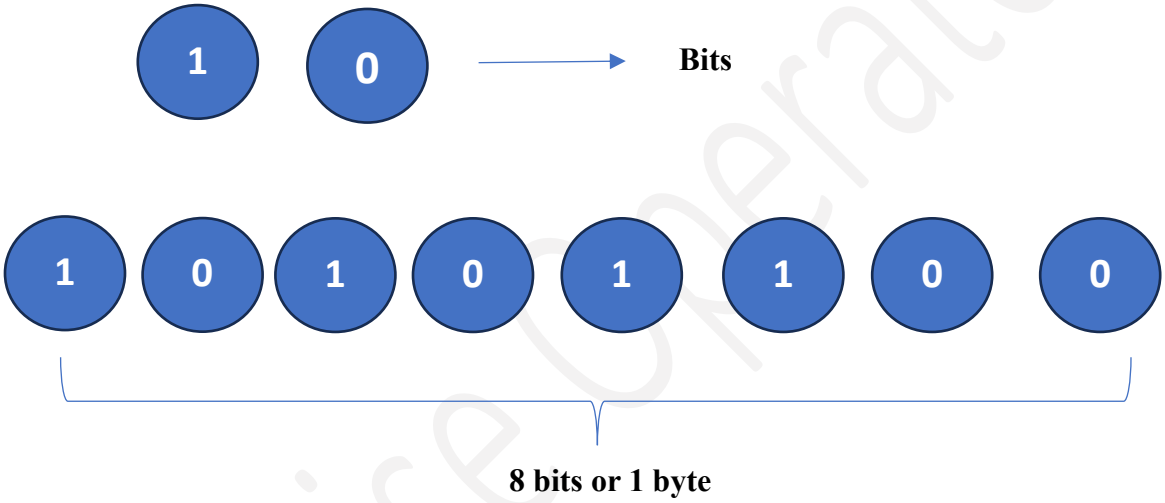
1. Bit:

A bit (binary digit) is the smallest unit of data that a computer can process and store.

- A bit is always in one of two physical states, similar to an on/off light switch.
- The state is represented by a single binary value, usually a 0 or 1.
- However, the state might also be represented by yes/no, on/off or true/false.
- Bits are stored in memory through the use of capacitors that hold electrical charges.

2. Byte:

A Byte is a unit of data measurement which mainly consists of eight bits. A byte is a series of binary digits, which contain '0' or '1'. Below shows the example of bit and byte.



3. Bitwise Operators:

The bitwise operators are the operators used to perform the operations on the data at the bit-level. When we perform the bitwise operations, then it is also known as bit-level programming. It consists of two digits, either 0 or 1. It is mainly used in numerical computations to make the calculations faster.

3.1 Types of Bitwise Operators:

There are six types of the bitwise operator in Java:

- Bitwise AND (&)
- Bitwise exclusive OR (^)
- Bitwise inclusive OR (|)
- Bitwise Compliment (~)
- Bit Shift Operators (<< and >>)

BITWISE OPERATORS

Table 3.1: Types of Bitwise Operators

Operators	Symbol	Uses
Bitwise AND	&	op1 & op2
Bitwise exclusive OR	^	op1 ^ op2
Bitwise inclusive OR	|	op1 | op2
Bitwise Compliment	~	~ op
Bitwise left shift	<<	op1 << op2
Bitwise right shift	>>	op1 >> op2

3.1.1 Bitwise AND (&):

A bitwise AND is a binary operation that takes two equal-length binary representations and performs the logical AND operation on each pair of the corresponding bits. Thus, if both bits in the compared position are 1, the bit in the resulting binary representation is 1 (1 × 1 = 1); otherwise, the result is 0 (1 × 0 = 0 and 0 × 0 = 0). It is denoted by the symbol **'&'**.

Table 3.2: Bitwise AND

x	y	x&y
0	0	0
0	1	0
1	0	0
1	1	1

For example,

0101 (decimal 5)

& 0011 (decimal 3)

= 0001 (decimal 1)

BITWISE OPERATORS

3.1.2 Bitwise Exclusive OR (^):

A bitwise Exclusive OR is a binary operation that takes two-bit patterns of equal length and performs the logical exclusive OR operation on each pair of corresponding bits. The result in each position is 1 if only one of the bits is 1, but will be 0 if both are 0 or both are 1. In this we perform the comparison of two bits, being 1 if the two bits are different, and 0 if they are the same. It is a binary operator denoted by the symbol ^ (pronounced as caret).

Table 3.3: Bitwise Ex-OR

x	y	x^y
0	0	0
0	1	1
1	0	1
1	1	0

For example,

0101 (decimal 5)

^ 0011 (decimal 3)

= 0110 (decimal 6)

3.1.4 Bitwise Inclusive OR (|):

A bitwise inclusive OR is a binary operation that takes two bit patterns of equal length and performs the logical inclusive OR operation on each pair of corresponding bits. The result in each position is 0 if both bits are 0, while otherwise the result is 1. It is a binary operator denoted by the symbol | (pronounced as a pipe).

Table 3.4: Bitwise OR

x	y	x^y
0	0	0
0	1	1
1	0	1
1	1	1

For example,

0**101** (decimal 5)

| 0**011** (decimal 3)

= 0**111** (decimal 7)

3.1.5 Bitwise Complement (~):

The bitwise NOT, or bitwise complement, is a unary operation that performs logical negation on each bit, forming the ones' complement of the given binary value. Bits that are 0 become 1, and those that are 1 become 0. It is a unary operator denoted by the symbol ~ (pronounced as the tilde).

Table 3.5: Bitwise Not

x	~ x
0	1
1	0

For example,

~ 0111 (decimal 7)

= 1000 (decimal 8)

3.1.6 Bit Shift Operators

Shift operator is used in shifting the bits either right or left. We can use shift operators if we divide or multiply any number by 2.

3.1.6.1 Bitwise Left shift (<<):

The left shift operator moves all bits by a given number of bits to the left.

For example,

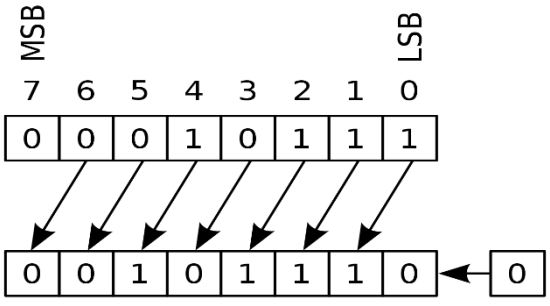


Fig 3.1: Bitwise Left Shift

3.1.6.2 Bitwise Right shift (>>):

The right shift operator moves all bits by a given number of bits to the right.

For example,

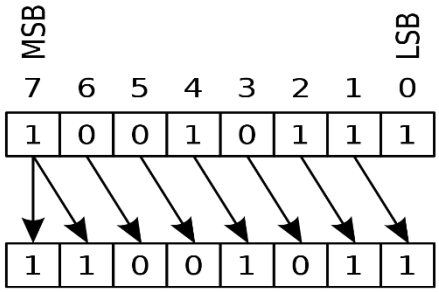


Fig 3.2: Bitwise Left Shift

Note:

- MSB is n a binary number, the bit furthest to the left is called the most significant bit (msb).
- LSB is an binary number, the bit furthest to the right is called the least significant bit (lsb)