# BITWISE OPERATORS & BIT-SHIFT OPERATORS (SHIFT OPERATORS)

Bitwise operators are characters that represent actions (bitwise operations) to be performed on single bits. They operate at the binary level and perform operations on bit patterns that involve the manipulation of individual bits. Thus, unlike common logical operators like + or - which work with bytes or groups of bytes, bitwise operators can check each individual bit within a byte.

Shift operators are used to shift the bits of a number left or right, thereby multiplying or dividing the number by two, respectively. They can be used when we must multiply or divide a number by two.

## BITWISE AND &

The bitwise AND operator produces an output of 1 if the corresponding bits of both the operands are 1. If not, the output is 0.

Example 1: Bitwise AND operation of two one-bit operands.

| LEFT OPERAND | RIGHT OPERAND | RESULT |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Example 2: Bitwise AND operation of two integers: 28 and 17; the & operator compares each binary digit of these integers.

| BINARY DIGITS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 28 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 17 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| Are both digits 1? | No | No | No | Yes | No | No | No | No |
| Bitwise AND output | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Thus: 28 & 17 (bitwise AND) = 00010000 (binary) = 16 (decimal).

## BITWISE OR |

The bitwise OR operator produces an output of 1 if either one of the corresponding bits is 1. Otherwise, the output is zero.

Example 1: The bitwise OR operation of two one-bit operands.

| LEFT OPERAND | RIGHT OPERAND | RESULT |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Example 2: Let's consider the previous example of two integers: 28 and 17.

| BINARY DIGITS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 28 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 17 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| Is either digit 1? | No | No | No | Yes | Yes | Yes | No | Yes |
| Bitwise OR output | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |

Thus: 28 | 17 (bitwise OR) = 00011101 (binary) = 29 (decimal).

## Bitwise exclusive OR (XOR)

The bitwise exclusive OR (XOR) operator returns 1 if the bits of both operands are opposite. Otherwise, it returns 0.

Example 1: The bitwise XOR operation of two one-bit operands.

| LEFT OPERAND | RIGHT OPERAND | RESULT |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |

| LEFT OPERAND | RIGHT OPERAND | RESULT |
|---|---|---|
| 1 | 1 | 0 |

Example 2: Let's see how bitwise XOR works for our two integers 28 and 17.

| BINARY DIGITS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 28 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 17 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| Are the two digits opposite of each other? | No | No | No | No | Yes | Yes | No | Yes |
| Bitwise XOR output | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

Thus: 28 ^ 17 (bitwise XOR) = 00001101 (binary) = 13 (decimal).

## BITWISE LEFT SHIFT

The bitwise left shift operator shifts the bits left by the bits specified by the right operand. The positions vacated by the left shift operator are filled with 0.

Example: Let's perform the bitwise left shift operation on the integer 6. Each bit will be shifted left by 1.
6 = 0110
6<<1 = 1100 (binary) = 12 (decimal)

## BITWISE RIGHT SHIFT

Like the left shift operator, the bitwise right shift operator shifts the bits right by the bits specified by the right operand. The positions vacated by the right shift operator are filled with 0.

Example: Let's perform the right shift by two bits operations on the integer 8. Each bit will be shifted right by 2.
8 = 1000
8>>2 = 0010 (binary) = 2 (decimal)

**Here's a Java program that demonstrates the usage and applications of the bitwise operators:**

```java
class BitwiseOperators {
    public static void main(String[] args) {
        int a = 5;  // binary: 0101
        int b = 3;  // binary: 0011

        // Bitwise AND
        int resultAnd = a & b;  // binary: 0001 = 1
        System.out.println("Result of Bitwise AND: " + resultAnd);
        // Application: Setting up a mask to check the values of specific bits

        // Bitwise OR
        int resultOr = a | b;  // binary: 0111 = 7
        System.out.println("Result of Bitwise OR: " + resultOr);
        // Application: Adding two numbers if there is no carry involved

        // Left Shift
        int resultLeftShift = a << 2;  // binary: 010100 = 20
        System.out.println("Result of Left Shift: " + resultLeftShift);
        // Application: Shifting bits to align them

        // Right Shift
        int resultRightShift = a >> 1;  // binary: 0010 = 2
        System.out.println("Result of Right Shift: " + resultRightShift);
        // Application: Shifting bits to align them
    }
}
```

**Output:**

```
Result of Bitwise AND: 1
Result of Bitwise OR: 7
Result of Left Shift: 20
Result of Right Shift: 2
```

Here is the link to access the assignment / java file:
https://drive.google.com/file/d/1gXqWFFPlKrdwASHFwdzQRwqTSyrx0w4k/view?usp=drive_link

Best regards,
**SHANTHRAJ GUTHAL.B**