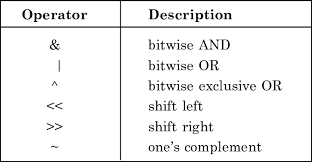
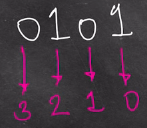
**Bit Manipulation**

****

**4 Main Operations in Bit Manipulation**

1. **Get:** Ek binary number ke ith position pe kaun sa bit h
2. **Set:** ith position pe put 1
3. **Clear:** ith position pe put 0
4. **Update:** ith position pe 0 to 1 and 1 to 0 put karna

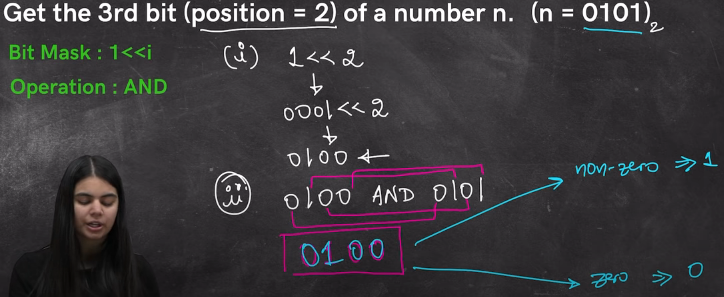
**Note:** Position right to left se count shuru hota h and left jaate rahte h. Counting is 0 based. **Ex,**

**Bit Masking:** It’s a process where we the manipulate a desired/given number (kisi position me kisi number ko laana). We basically use another number to perform Bit Masking that number is known as Bit Mask.

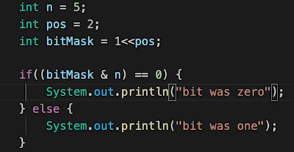
To basically,

1. Bit Mask pahle apn create krte h.
2. ii) original number and Bit Mask number me khuch operation (AND,OR,XOR,etc) krte h.

Issi process ko Bit Masking bolte h. Is process se apn apna ye upar ke 4 operation perform krte h and apna required output nikal lete h.

1. **Get Bit**
2. **Bit Mask: 1<<i** (i is the position of the bit)
3. **Operation: AND**

Yaha dekh 1 ko left shift by position kiya phir number and bit mask ka AND operation kiya and operation ke baad output non zero mtlb us position pe 1 tha and zero output aaya to vaha 0 tha

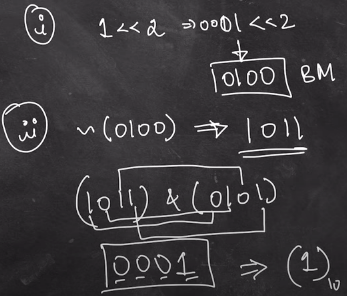
****

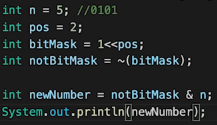
**Code:**

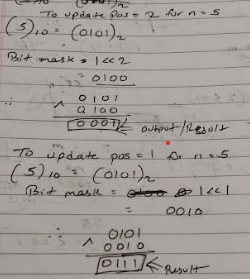
1. **Set Bit**
2. **Bit Mask: 1<<i**
3. **Operation: OR**

Same as Get bit bss AND ke badle OR operation peform kr rahe h.

1. **Clear Bit**
2. **Bit Mask: 1<<i**
3. **Operation: NOT of Bit Mask and then AND of number and Bit Mask**

****Q)

<= Bit Mask ka complement Nikala.Us complement ka given no ke saath AND nikala and done. CODE =>



1. **Update/Toggle Bit**
2. **Bit mask: 1<<i**
3. **Operation: XOR**

**LinkedList**

**LinkedList Implementation**

class Node{

    int data;

    Node next;

    Node(int data){

        this.data = data;

        next = null; // Optional hai ye line. bina assigned object null hi hota h

    }

}

class LinkedList{

    public static void main(String[] args) {

        Node head = new Node(5);

        Node temp = new Node(10);

        Node temp2 = new Node(3);

        head.next = temp;

        temp.next = temp2;

    }

}

**LinkedList Traversal**

Node cur = head;

while(cur != null){

System.out.print(cur.data + " ");

cur = cur.next;

}

**Insertion in the Beginning**

public static Node insertHead(Node head,int x){

    Node temp = new Node(x);

    temp.next = head;

    return temp;

}

public static void main(String[] args) {

    Node head = null;

    head = insertHead(head,10);

    head = insertHead(head,3);

    head = insertHead(head, 30);

    }