Bit - Manipulation'

Binary Number system

0 & 1

Decimal Number system = 0, 1, 2, ... 8, 9

→ 10 digits

Binary (100)

D → B

$$\begin{array}{r|l} 2 & 4 \\ \hline 2 & 2 \quad 0 \\ \hline & 1 \end{array}$$

$(100)_2 = (4)_{10}$

Decimal (100)

B → D

1 0 0

$1 \times 2^2 \quad 0 \times 2^1 \quad 0 \times 2^0$

4 0 0

4+0+0

$(100)_2 = (4)_{10}$

0	→	000
1	→	001
2	→	010
3	→	011
4	→	100
5	→	101
6	→	110
7	→	111
8	→	$\boxed{1000}$

4 bits

Bitwise operators :-

Binary And &

Binary OR |

Binary XOR ^

Binary one's complement ~

Binary left shift <<

Binary Right shift >>

Binary AND &

Rules:-

0 & 0 → 0

0 & 1 → 0

1 & 0 → 0

$\boxed{1 \,\&\, 1 \rightarrow 1}$

5 & 6 A = 0101 B = 0110

$$\begin{array}{r} 101 \\ 110 \\ \hline 100 \end{array} = (4)_{10}$$

$\boxed{5+6=11}$ Arithematic operator

$\boxed{5\&6=4}$ Bitwise operator.

Code:-

```
Import java.util.*;
public class BitManipulation {

  ps vm (S A[])
    {
     System.out.print((5 & 6))
    }
  }
```

Binary XOR (^)

Rules

0 XOR 0 = 0
0 XOR 1 = 1 }
1 XOR 0 = 1 }
1 XOR 1 = 0

5 ^ 6

```
 0101
 0110
 ----
 0011
```

$=(3)_{10}$

syso(5 ^ 6)

Binary OR |

5|6   A = 0101   B = 0110

Rules

	(or)		
0	|	0	0
0	|	1	1
1	|	0	1
1	|	1	1

```
 0101
 0110
 ----
 0111.
```

= 7(10)

syso (5 | 6)

Binary one's Complement ~

Tilt (~)

~5   A = 0101

~A = 010(x)

~0 → 1
~1 – 0

5 → 0000101

0 000101
-ve
1 000101
+ve

LSB = least significant Bit
MSB = most significant Bit

2's complement → 1's complement + 1(add)

(0 0 0 0 0 1 $\overset{1}{0}$ 1) 1's complement
+1

0 0 0 0 0 1 1 0

-ve 0 ← 6
+ve 1 ↙

$\boxed{\sim 0 \rightarrow +1}$ due to MSB

~5 → 0 0 0 0 0 1 0 1
↓
1 1 1 1 1 0 1 0
↓
1's complement
↓
0 0 0 0 0 1 0 1
↓ +1

0 0 0 0 0 1 1 0 $(6)_{10}$

Binary left shift << :-

Rules:-

a << b (2)

x $\boxed{0\,0}$ 0 1 0 1

0 1 0 1 0 0

$$\boxed{a << b = a \times 2^b}$$

ex:- 5 << 2

0 0 0 1 0 1

0 1 0 1 0 0

5 << 2
$= 5 \times 2^2$
$= 5 \times 4$
$= 20$

syso (5 << 2);

op/ = 20

(5×2^p)
$= 5 \times 2^2$
$= 5 \times 4$
$= 20$

Binary Right shift :-

>> 2

ex:- 0 0 1 0 0 1 $\boxed{0\,0}$ x

0 0 0 0 1 0 0 1

$$\boxed{a >> b = a/2^b}$$

6 >> 1 = A = 0 0 0 1 1 $\boxed{0}$ x

0 0 0 0 1 1

$= (3)_{10}$

$= 6/2^1$

$= 6/2 = 3$

Check if a number is odd or even.

3 = 011   1 = 001   Bitmask = 1
= 001

3 & 1 =
```
 011
 001
-----
 001
```
$(1)_{10}$ odd

4 & 1
```
 100
 001
-----
 000
```
$(0)_{10}$ even.

Code :-

```
public static evenOrodd(int n) {
    int bitmask = 1;
    if ((n & bitmask) == 0) {
        syso ("even");
    }
    else {
        syso ("odd");

psvm SA() {
    int n = arr 5;
    even or odd(n);
```

Operations :-

Get ith bit

Set ith bit

clear ith bit :

(1)
```
public static int getibit (int n, int i)
    int Bitmask = (n<<i);
    ~~fo~~ if ( ~~xxx~~(n & bimask) == 0)
    {
        return 0;
    }
    else
    {
        return 1 }
psvmSA() {
    syso ( get i bit (10, 3));
```

2) set ith bit

```
    0 0 0 0 1 0 1 0   i=2
10= 0 0 0 0 1 1 1 0
            2^3 2^2 2^1 2^0
            8+4+2+0 =14
```

```
public static int getith(int n, int i)
  int bit mask = n<<i;

  if (n & ~~bitmask ==0)~~
    {
    return n | bitmask;

  }

  psvm SA() {
    syso(geti(int n, int i);
      }
```

3) clear ith bit

```
ps i f( int n , int i ){
int bitmask = ~(1<<i)

  return n & bit mask;
}
  psvmSA()
~~syso~~   {
    ~~8~~ fun
  sysopln(10, 2);

  }
```

```
1 0 1 0
1 0 0 1
  1 1 0
2^4
8-1 2^4 2^2 2^0
4+2+1 =7
```

4) update ith bit :-

```
public static update ith bit ( int n , int i , int newBit ) {

    if (newbit == 0) {
      return clearithBit (n, i);

    }
    else {
      return setInBit (n, i)
      }

      (or)

    n = clearBit (n, i);
    int nBitmask =    n<<i;
    return n | Bitmask;
```

Clear last i bits :-

n= 1111 , i= 2 BM= ~0<<i

↓

1100

```
Public static int clearBits (int n, int i ){
        int BitMask = (~0) << i ;
        return n & bitmask
        }

    psvmsA {
        clearBits ( 15, 2 ); = 12
```

Clear Range of Bits :-

n = 10011101001l , i=2 , j = 7

```
public static int clearBits (int n , int i, int j ) {
    int  a = ((~0) << (j+1)) ;
    int  b = (1<<i) -1 ;
    int bmask = a|b ;
    return  n & bitmask ;

      psvmsA {
        clearBits (10, 2, 4) ; = 2.
      }
```

Q-2

check if a number is a power of 2 or not :-

$4 = 2^2$
$8 = 2^3$
$7 = 2^7$ ✗

4 → 100 & → 0
3 = 011

8 → 1000 & → 0
7 → 0111

16 → 10000 & → 0.
15 → 01111

$\boxed{n \,\&\, n-1} = 0$

```
public static Booleen ispowof2 (int n){
        return (n & (n-1))==0;
    }
    psumsA ()
    {
    is pow (2 (15)); false.
```

Q-3

count setBits in a Number :-

10 → 1010

no of set Bits = 2.

```
1)  n = 101|0 → x
        n>>1           count = 0
2)  0101
        n>>1           count = 1
3)  0010
4)  0001               cont = 1
(v) 0000               cot = 2
```

```
public static in countBits (int n)
{
    int count = 0;
    while (n >0) {
        if (n & 1 ) != 0) {
            count ++;
        }
        n = n>>1;
    }
    ps v msA ( )
    {
        set bin in ((setB (16)) ?
    }
```

Fast Exponentiation :-

a^n = a x a x a x a → n times.

3^5

5^3

code :- a^n.

```
Public static int fast exp (int a , int n) {
    int ans = 1;
    while (n>0) {
        if (n & 1 != 0) { // check LSB
            {
            ans = ans * a;
            }
            a = a * a;
            n = n>>1;
        }
        return ans;
    }
    psvmSA ( ) {
        syso( fast expo (3,5));
```

Modular Exponentiation => $\{a^n \% x\}$ (google)

Assignment Questions:-

1) $x \wedge x = 0$

2) swap 2 Numbers without using 3rd variable.

```
psvmSA () {
    int x = 3, int y = 4
    syso (x , y ) // before swapping
    x = x ^ y;
    y = x ^ y;
    x = x ^ y;
    syso (x, y) // after swapping
    }
```