DSA

Module 3

Bit Manipulation

Author
Srinivas Dande

3. Bitwise Operators

- These Operators will be used to perform the operation on individual bits of the value.

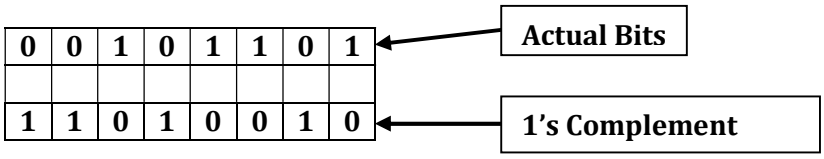
Bitwise Operators	Description
~	Bitwise NOT
&	Bitwise AND
|	Bitwise OR
^	Exclusive OR (XOR)
<<	Left Shift
>>	Right Shift
>>>	Unsigned Right Shift

- Operands for the following Bitwise Operators can be of boolean type or Integer type:
  - Bitwise AND
  - Bitwise OR
  - Exclusive OR (XOR)

- Operands for the following Bitwise Operators will be of Integer type only:
  - Bitwise NOT
  - Left Shift
  - Right Shift
  - Unsigned Right Shift

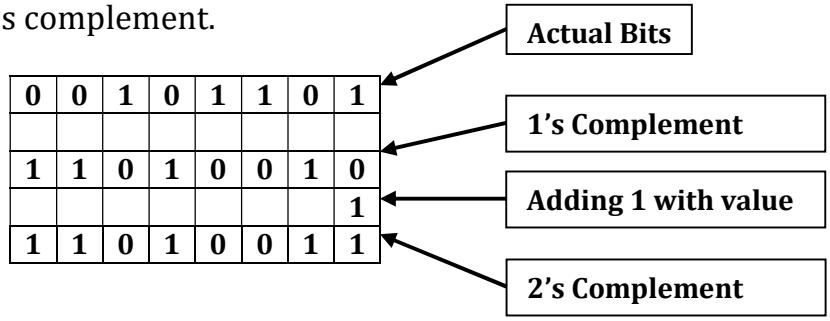
1's Complement

- 1's complement of a binary number is formed by changing 1's to 0's and 0's to 1's



2's Complement

- 2's complement of a binary number is formed by adding 1 to the least significant bit of 1's complement.



3.1. Exploring Bitwise AND

- Bitwise AND returs 1 when both the bits are 1
- Bitwise AND returs 0 when any one bit is 0.
- **Truth Table of Bitwise AND**

A	B	A & B
1	1	1
1	0	0
0	1	0
0	0	0

Lab1.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Lab1 {
	public static void main(String[] args) {
		int a= 5;
		int b= 7;
		int result= a & b;
		System.out.println(result);
	}
}
```

3.2. Exploring Bitwise OR

- Bitwise OR returs 0 when both the bits are 0
- Bitwise OR returs 1 when any one bit is 1.
- **Truth Table of Bitwise OR**

A	B	A | B
1	1	1
1	0	1
0	1	1
0	0	0

Lab2.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab2 {
	public static void main(String[] args) {
		int a= 5;
		int b= 7;

		int result= a | b;

		System.out.println(result);
	}
}
```

3.3. Exploring Bitwise XOR

- Bitwise XOR returs 0 when both the bits are same
- Bitwise XOR returs 1 when both the bits are different.
- **Truth Table of Bitwise XOR**

A	B	A ^ B
1	1	0
1	0	1
0	1	1
0	0	0

- **Imp Points to Remember with Bitwise XOR**
    1. (a^b)^c = a^(b^c)
    2. a ^ 0 = a
    3. a ^ a = 0
    4. a ^b = b ^ a

Lab3.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab3 {
	public static void main(String[] args) {
		int a= 5;
		int b= 7;

		int result= a ^ b;

		System.out.println(result);
	}
}
```

Lab4.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab4 {
	public static void main(String[] args) {

		int p= 5;

		System.out.println(p ^ p); //0
		System.out.println(p ^ 0); //p
		System.out.println(0 ^ p); //p

		int a=9;
		int b= 13;
		int c= 25;

		int x= a^b;
		int y= b^a;

		System.out.println(x);
		System.out.println(y);

		x= (a^b)^c;
		y= a^(b^c);

		System.out.println(x);
		System.out.println(y);

	}
}
```

3.4. Exploring Bitwise NOT

- Bitwise NOT returs 0 when the Bit is 1
- Bitwise NOT returs 1 when the Bit is 0.
- **Truth Table of Bitwise NOT**

A	~A
1	0
0	1

Lab5.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab5 {

	public static void main(String[] args) {

		int a=8;
		int b= -7;

		int result1= ~a;
		int result2= ~b;

		System.out.println(result1);
		System.out.println(result2);

	}
}
```

3.5. Exploring Bitwise LeftShift

- Bitwise LeftShift Operator shift the Bits towards leftside for Specified number of Times
- When Bits are Shifted to Left Once then It is Equals to multiplying the number by 2

Lab6.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab6 {

	public static void main(String[] args) {

		int a=5;

		System.out.println(a<<1);
		System.out.println(a<<2);
		System.out.println(a<<3);
		System.out.println(a<<4);

	}

}
```

3.6. Exploring Bitwise Rightshift

- Bitwise RightShift Operator shift the Bits towards rightside for Specified number of Times
- When Bits are Shifted to Right Once then It is Equals to dividing the number by 2
- RightShift also called as Signed RightShift
- Signed RightShift holds the Sign i.e
    - If the Number is Positive then fills with **Zero**
    - If the Number is Negative then fills with **One**

Lab7.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab7 {

	public static void main(String[] args) {

		int a=25;

		System.out.println(a>>1);
		System.out.println(a>>2);

		int b=-25;

		System.out.println(b>>1);
		System.out.println(b>>2);
	}

}
```

3.7. Exploring Bitwise Unsigned Rightshift

- Bitwise Unsigned RightShift Operator shift the Bits towards rightside for Specified number of Times
- When Bits are Shifted to Right Once then It is Equals to dividing the number by 2
- Unsigned Signed RightShift Ignores the Sign i.e
  - If the Number is Positive then fills with **Zero**
  - If the Number is Negative then also fills with **Zero**

Lab8.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

public class Lab8 {

	public static void main(String[] args) {

		int a=5;

		System.out.println(a>>>1);
		System.out.println(a>>>2);

		int b=-5;

		System.out.println(b>>>1);
		System.out.println(b>>>2);
	}

}
```

3.8. Problems on Bitwise

Lab9.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//Check whether Last bit of given number is Set or Not
public class Lab9 {
	public static void main(String[] args) {
		int n=4;
		if((n & 1) != 0)
			System.out.println("Last Bit is Set");
		else
			System.out.println("Last Bit is Not Set");
	}
}
```

Lab10.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//Check whether given number is Odd or Even
//If Last Bit is Set then It is Odd
//If Last Bit is Un-Set then It is Even
public class Lab10 {
	public static void main(String[] args) {
		int n=6;
		if((n&1) !=0)
			System.out.println("Odd");
		else
			System.out.println("Even");
	}
}
```

Lab11.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

// Check whether Kth bit of given number is Set or Not

public class Lab11 {

	public static void main(String[] args) {

		int n=15;
		int k=5;

		for(int i=1;i<=k-1;i++)
			n= n >> 1;    //Shifting One time

		if((n & 1) != 0)
			System.out.println("Kth Bit is Set");
		else
			System.out.println("Kth Bit is Not Set");

	}

}

// Time Complexity => O(n)
```

Lab11A.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

// Check whether Kth bit of given number is Set or Not

public class Lab11A {

	public static void main(String[] args) {

		int n=15;
		int k=5;

		n = n >> k-1;   //Shifting k-1 times

		if((n & 1) != 0)
			System.out.println("Kth Bit is Set");
		else
			System.out.println("Kth Bit is Not Set");

	}

}

// Time Complexity=> O(1)
```

Lab12.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

// Count the Number of Set Bits in given number
// Visiting All the Bits and Counting only Set Bits

public class Lab12 {
	public static void main(String[] args) {

		int n = 1024;
		int count = 0;
		int loopCount = 0;

		while (n > 0) {
			loopCount++;
			if ((n & 1) != 0)
				count++;

			n = n >> 1;   //   n= n/2;

		}

		System.out.println("Loop Count : " + loopCount);
		System.out.println("Set Bit Count : " + count);
	}
}

// Time Complexity =>O(100) => O(1)
```

Lab12A.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

// Count the Number of Set Bits in given number
//Brain Kerningam's Algorithm
//Visit and Count only Set Bits

public class Lab12A {
	public static void main(String[] args) {

		int n = 40;
		int count = 0;

		while (n > 0) {

			n = n & (n-1);
				count++;
		}

		System.out.println("Set Bit Count : " + count);
	}
}

// Time Complexity => O(1)
```

Lab13.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

// Find whether Given Number is Power of 2 or Not
// Number of Set Bits are 1

public class Lab13 {
	public static void main(String[] args) {

		int n = 63;
		int count = 0;

		while (n > 0) {
			n = n & (n-1);
				count++;
		}

		System.out.println("Set Bit Count : " + count);

		if(count==1)
			System.out.println("2 Power ");
		else
			System.out.println("Not 2 Power ");

	}
}

// Time Complexity => O(n) /O(1)
```

Lab13A.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

// Find whether Given Number is Power of 2 or Not
// Number of Set Bits are 1

public class Lab13A {
	public static void main(String[] args) {

		int n = 64;

		if( ( n & (n-1) ) == 0)
			System.out.println("2 Power ");
		else
			System.out.println("Not 2 Power ");

	}
}

// Time Complexity => O(1)
```

Lab14.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

// Reverse the Bits of given Number

public class Lab14 {
	public static void main(String[] args) {

		int n = 15;
		int rev = 0;

		while (n > 0) {
			rev = rev << 1;
			if( (n & 1) == 1){
				rev = rev ^ 1;
			}
			n = n>>1;

		}

		System.out.println(rev);

	}
}

// Time Complexity => O(1)
```

Lab15.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

// Add two Numbers without + Operatror

public class Lab15 {
	public static void main(String[] args) {

		int a = 71;
		int b = 51;

		while (b > 0) {
			int temp = (a & b) <<1;
			a = a^b;
			b = temp;
		}

		System.out.println(a);

	}
}

// Time Complexity => O(1)
```

Lab16.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

// Find the One Odd Occurring Number in Array

public class Lab16 {
	public static void main(String[] args) {

		int arr [] = {3,5,7,3,7,5,9,9,5};
		int length = arr.length; //9

		for(int i=0; i<length; i++) {
			int count = 0;
			for(int j=0;j<length;j++) {
				if(arr[i]==arr[j]) {
					count++;
				}
			} //Inner For loop

			if(count%2!=0) {
				System.out.println(arr[i]);
				break;
			}
		} //Outer For loop

	}
}

// Time Complexity => O(n^2)
```

Lab16A.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

// Find the One Odd Occurring Number in Array

public class Lab16A {
	public static void main(String[] args) {

		int arr [] = {3,5,7,3,7,5,9,9,5};
		int length = arr.length; //9

		int result = 0;
		for(int i=0; i<length; i++) {
		  result = result ^ arr[i];
		}

		System.out.println(result);

	}
}

// Time Complexity => O(n)
```

Lab17.java

```
package com.jlcindia;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

// Find Two Odd Occurring Numbers in Array

public class Lab17 {
	public static void main(String[] args) {
		int arr[] = { 3, 5, 3, 7, 9,7 };
		int length = arr.length; // 9

		int result = 0;
		for (int i = 0; i < length; i++) {
			result = result ^ arr[i];
		}

		int k = result & (~(result - 1));

		int first = 0;
		int second = 0;

		for (int i = 0; i < length; i++) {
			if ((arr[i] & k) == 0)
				first = first ^ arr[i];
			else
				second = second ^ arr[i];
		}
		System.out.println(first);
		System.out.println(second);
	}
}
// Time Complexity => O(n)
```

Lab18.java

```
package com.jlcindia;

import java.util.*;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */

// Find the Occurence of each element in the array
public class Lab18 {
	public static void main(String[] args) {

		int arr[] = { 3, 5, 6, 3, 7, 9, 7, 7, 6, 9 };
		int length = arr.length;

		Map<Integer, Integer> mymap = new HashMap<>();

		for (int i = 0; i < length; i++) {
			int count = 0;
			for (int j = 0; j < length; j++) {
				if (arr[i] == arr[j]) {
					count++;
				}
			} // Inner For loop

			mymap.put(arr[i], count);

		} // Outer For loop

		System.out.println(mymap);
	}
}

// Time Complexity => O(n^2)
```