1. Basics of Bit Manipulation:

1. How data stores i.e. Numbers
2. types of Data (Number), Range
   - → byte
     - short
     - int
     - long

   Nibble → 4 bits
3. Decimal to Binary | Binary to Decimal
4. Operators (Bits)
5. How to ON a bit
6. How to OFF a bit
7. toggle
8. Check if bit is ON or OFF

1. Basics of bit Mani.
2. Print values of RSB
3. Kernighan's Algorithm
4. Josephus special
5. Gray code

n = 57

↳ Data store in machines in binary form. →

Resistors: Low → 0 (False), High → 1 (True)

Base System →

Decimal base

Base - 10

No. of digits in number sys

Decimal → 10 → 0, 1, 2 ..... , 9

Binary → 2 → 0, 1

Store → Binary form

print → Decimal form

$(57)_{10} \rightarrow (x)_2$

$x = (111001)_2$

2	57	
2	28	1
2	14	0
2	7	0
2	3	1
2	1	1
	0	1

let discuss about nibble [data have only 4 bits allowed].

↳ suppose it is a data type have 4 bits

most significant Bit

0	0	0	0	→	0	1	0	0	0	→	8
0	0	0	1	→	1	1	0	0	1	→	9
0	0	1	0	→	2	1	0	1	0	→	10
0	0	1	1	→	3	1	0	1	1	→	11
0	1	0	0	→	4	1	1	0	0	→	12
0	1	0	1	→	5	1	1	0	1	→	13
0	1	1	0	→	6	1	1	1	0	→	14
0	1	1	1	→	7	1	1	1	1	→	15

x = [ ][ ]

msb — considered bit for number

from 4 bits, → we can represent number from 0 to 15.

NOTE:

* we can store 16 numbers - from 0 to 15, But -ve numbers are not there

Approach '0' for negative numbers: →

If msb for a binary represent is 0 then no. if +ve-

If msb for a binary is 1 then number is -ve-

MSB check for +ve, -ve

msb → 0 → +ve

msb → 1 → -ve

Problem arise for +ve 0 and -ve 0.

msb

0 0 0 0 → +0 ★

0 0 0 1 → +1

0 0 1 0 → +2

0 0 1 1 → +3

0 1 0 0 → +4

0 1 0 1 → +5

0 1 1 0 → +6

0 1 1 1 → +7

msb

1 0 0 0 → -0 ★

1 0 0 1 → -1

1 0 1 0 → -2

1 0 1 1 → -3

1 1 0 0 → -4

1 1 0 1 → -5

1 1 1 0 → -6

1 1 1 1 → -7

Discarded method because of +ve 0 and -ve 0

Possibility from 4 bits are following these, but we make a change to understand +ve and -ve number.

0 0 0 0 → 0	1 0 0 0 → -8
0 0 0 1 → 1	1 0 0 1 → -7
0 0 1 0 → 2	1 0 1 0 → -6
0 0 1 1 → 3	1 0 1 1 → -5
0 1 0 0 → 4	1 1 0 0 → -4
0 1 0 1 → 5	1 1 0 1 → -3
0 1 1 0 → 6	1 1 1 0 → -2
0 1 1 1 → 7	1 1 1 1 → -1

4bits → -8 to 7

double 0's problem is resolved now.

Possibility for msb
- msb → 0 → simple conversion
- msb → 1 → take 2's compliment and attach a negative sign to a number.

2's compliment → 1's compliment + 1 (represented by stick)
- Represented by double tick.
- 1's compliment: toggle of all bits is 1's compliment of a number.

Example.

$(1001)'' \rightarrow (1001)' + 1$

$\rightarrow (0110) + 1$

$\rightarrow 0111$ → Conversion

$\rightarrow -7$

-8 - 1

Range of number → -8 to 7

4 bits.

Binary	Value
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

cyclic behaviour of N.S. (Number system)

5 → 0 1 0 1

5 → 0 1 0 1

1 0 1 0 → -6

add 1 in 7 → 7+1

in Binary System ⇒ 7 → 0 1 1 1

1 → 0 0 0 1

Binary → MSB → 1 0 0 0

↳ to integer

↳ -8

if we add 1 in max. possible value of data type, then it become change in minimum possible value of data type

Binary Subtraction -

$(10)_2 \to (2)_{10}$

-8 → 1 0 0 0

1 → 0 0 0 1

0 1 1 1

to integer → +7

Integer → ∞ + 1 ⇒ -∞

Integer.MaxValue + 1

machine

-8 → 1 0 0 0

+

-1 → 1 1 1 1

0 → (+) ⇒ 0 1 1 1

Addition

COA → computer Architecture -

(-8) (+) (-1) = 7

(+) = (-9)

ALU → addition → perform

$-\infty + 1 \Rightarrow -\infty$ for any number system

(suppose here for nibble)

$-8 \rightarrow -\infty$ for nibble.

$-8 \rightarrow 1000$

$1 \rightarrow 0001$

$1001 \rightarrow -7$

MSB → 0 → Normal conversion → number is +ve

MSB → 1 → take 2's compliment and attach -ve sign to number

4 bits allowed

2's compliment → 1's compliment + 1

↳ toggle to all bits.

Example →

$1101 \rightarrow$ 2's complement

MSB = 1

↳ -ve Number

$(1101)'' \rightarrow (1101)' + 1$

$= 0010$

1

$0011 \rightarrow -3$

msB = 1 equal to 3

because MSB is 1

Range of Numbers: →

Data type	no. of bits.	Range.
nibble →	4	-2^3 to 2^3-1
byte →	8	-2^7 to 2^7-1
short →	16	-2^{15} to $2^{15}-1$
int →	32	-2^{31} to $2^{31}-1$
long →	64	-2^{63} to $2^{63}-1$

12 → $(1100)_2$ → $(12)_{10}$

int → nibble → (-4)

Higher bit to lower bit → lossy conversion.

n - bit number. -2^3 — 2^3-1

0 000 →	0
0 001 →	1
0 010 →	2
0 011 →	3
0 100 →	4
0 101 →	5
0 110 →	6
0 111 →	7

1 000 →	-8
1 001 →	-7
1 010 →	-6
1 011 →	-5
1 100 →	-4
1 101 →	-3
1 110 →	-2
1 111 →	-1

+ve Range → $2^{n-1}-1$

-ve Range → -2^{n-1}

Total Range → -2^{n-1} to $2^{n-1}-1$

msb for short → 1 → -ve

0 111 1111 ... 11 → +ve ∞ integers

31 bits | 16 bits | short

operators in Bit Manipulation :

time complexity → O(1)

|

OR

↓

a | 1 → 1

a | 0 → a

1 → Dominant OR

0 0 → 0

0 1 → 1

1 0 → 1

1 1 → 1

bits ON useful

&

AND

↓

a & 1 → a

a & 0 → 0

0 → dominant AND

0 0 → 0

0 1 → 0

1 0 → 0

1 1 → 1

bit off useful.

^

XOR

↓

a ^ 1 = ~a

a ^ 0 = a

XOR

0 0 → 0

0 1 → 1

1 0 → 1

1 1 → 0

same 0

otherwise 1

toggle.

ON → OFF

OFF → ON

toggling

<<

left shift

↓

shift bits toward left side.

x 0 0 1 1

↳ after left shift

0 1 1 0

>>

right shift

↓

shift bits toward right

↳ new incoming bit same as MSB

4 bit number.

msB → 1 1 0 0

↳ right shift

1 1 1 0

msB → 0 1 1 0

↳ 0 0 1 1

>>>

triple right shift

↓

shift bits toward right

↳ new incoming bit always '0'

1 1 0 0

↳ 0 1 1 0

0 1 1 0

↳ 0 0 1 1

~

1's compliment

↓

1's complement

toggle all bits.

-x

2's complement

↓

1's + 1

x shift k shifting

↳ TC >> >>>

Bit ON

$x \rightarrow$ 1 0 1 1 [0]* 1 0 1

↳ on this bit

$x \rightarrow$ 1 0 1 1 0 1 0 1

mask $\rightarrow$ 0 0 0 0 [1] 0 0 0

↳ 3 shift toward left

$x \mid mask$ (or) 1 0 1 1 1 1 0 1

$7 << 3 \rightarrow (1\,1\,1\,0\,0\,0)_2$

↳ $(1\,1\,1)_2$

on third bit on number 'x'

How to prepare mask

can we prepare → 0 0 0 0 0 0 0 ... 1 (Binary)

mask = 1 (in decimal)

Shift toward left in mask

mask = (1) << (3) ← no. of shift

↳ left shift

↑ number

no. of which we are performing shifting.

Bit OFF

$x =$ 1 0 1 [1]* 0 1 0 1

$x =$ 1 0 1 [1] 0 1 0 1

mask = 1 1 1 0 1 1 1 1

$x \,\&\, mask =$ 1 0 1 [0] 0 1 0 1

mask = (~(1 << 4))

1's compli ↑

0 0 0 0 ... 1

0 0 ... 1 0 0 0 0

is → 1 1 1 .. 0 1 1 1

Complexity : O(1)

O(1)

toggle a bit →

x = 1 0 1 1 0 1 0 1

mask → 0 0 0 1 0 0 0 0

x ∧ mask 1 0 1 0 0 1 0 1

↳ toggle.

Check if bit is ON or OFF :→

x = 1 1 0 1 0 0 1 1 → check if it is ON or OFF

mask = 0 0 0 1 0 0 0 0

x & mask = 0 0 0 1 0 0 0 0

(x & mask) == mask → bit is ON

(x & mask) == 0 → bit is OFF

Java Operator Precedence

Operators	Precedence
postfix increment and decrement	`++` `--`
prefix increment and decrement, and unary	`++` `--` `+` `-` `~` `!`
multiplicative	`*` `/` `%`
additive	`+` `-`
shift	`<<` `>>` `>>>`
relational	`<` `>` `<=` `>=` `instanceof`
equality	`==` `!=`
bitwise AND	`&`
bitwise exclusive OR	`^`
bitwise inclusive OR	`|`
logical AND	`&&`
logical OR	`||`
ternary	`? :`
assignment	`=` `+=` `-=` `*=` `/=` `%=` `&=` `^=` `|=` `<<=` `>>=` `>>>=`