1. Basics of Bit Manipulation:

1. How data stores i.e. Numbers
2. types of Data (Number), Range
   - → byte — nibble - 4 bits
   - short
   - int
   - long
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

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

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

$x = (111001)_2$

1. Basics of bit Mani.
2. Print values of RSB
   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 have Base - 10

Base – No. of digits in number sys.

Decimal → 10 → 0, 1, 2 . . . , 9

Binary → 2 → 0, 1

Store → Binary form

print → Decimal form

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

↳ suppose it is a data type have 4 bits

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

most significant Bit

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
1 0 0 0 →	8
1 0 0 1 →	9
1 0 1 0 →	10
1 0 1 1 →	11
1 1 0 0 →	12
1 1 0 1 →	13
1 1 1 0 →	14
1 1 1 1 →	15

x = [ ][ ]

msb — considered bit for number

NOTE:

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

Approach '0' for negative numbers: →

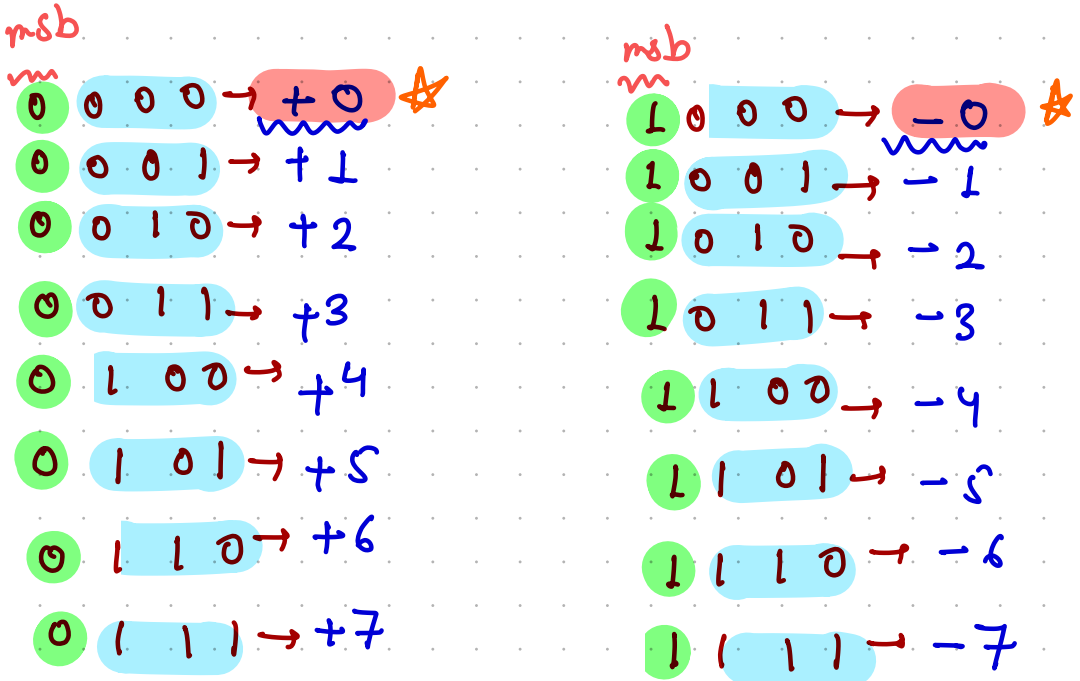
If msb for a binary represent is 0 then no. if +ve-

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.

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. (represented by stick)

2's compliment → 1's compliment + 1
- 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.

cyclic behaviour of N.S. (Number system)

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

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

to integer → +7

Binary Subtraction -

-8 1 0 0 0 $(10)_2 \rightarrow (2)_{10}$

-1 → 0 0 0 1

0 1 1 1

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

Integer → ∞ + 1 ⇒ -∞

Integer.MaxValue + 1

machine

-8 → 1 0 0 0

+ -1 → 1 1 1 1

0 1 1 1 → Addition → ALU → addition → perform

COA → computer Architecture -

(-8) + (-1) = 7

= -9

$-\infty + 1 \Rightarrow$

for any number system

(suppose here for nibble)

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

-8 and 1 → Add both of these.

$-8 \rightarrow 1000$

$1 \rightarrow 0001$

$1001 \rightarrow -7$

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

1's compliment → 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 → $(\ldots 0\,1\,1\,0\,0)_2$ → $(12)_{10}$

int → nibble → (-4)

Higher bit to lower bit → lossy conversion.

n - bit number. -2^3 — 2^3-1

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

+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 1 1 1 (1) 1 1 1 1 ... 1 1 → +ve ∞ int

31 bits | 16 bits → short

Operators in Bit Manipulation :

time complexity → O(1)

	&	^	<<	>>	>>>	~	-x
OR	AND	XOR	left shift	right shift	triple right shift	1's compliment	2's complem

| OR

a | 1 → 1

a | 0 → a

1 → Dominant

a	b	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

a	b	AND
0	0	0
0	1	0
1	0	0
1	1	1

bit off useful.

^ XOR

a ^ 1 = ~a

a ^ 0 = a

a	b	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 complem

1's + 1

x shift k shifting

↳ <<
>>
>>>

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 → 0 0 0 0 [1] 0 0 0 ← 3 shift toward left

x | 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)

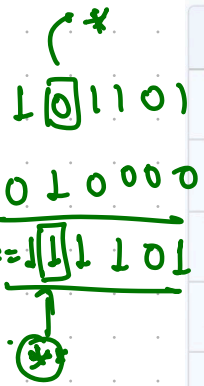
toggle a bit →

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	= += -= *= /= %= &= ^= |= <<= >>= >>>=