Bit Manipulation - 1

Jul 17, 2023

AGENDA

- Intro to Binary no. system
- Binary to Decimal and vice-versa
- Addition of Binary nos.
- Bitwise operators & properties

CDXXVI + CXLIV = ? Roman.

Number system - way of representing a no.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

↳ 10 symbols

Decimal no. system

101

Expand the following no.

735 = 700 + 30 + 5

4379 = 4000 + 300 + 70 + 9

4374 ← place value = 4

= position of digit multiplier

face value = 4

place value = 4000 4*1000

4379 = 4*1000 + 3*100 + 7*10 + 9

= $4*10^3 + 3*10^2 + 7*10^1 + 9*10^0$

$$\overset{4\ 3\ 2\ 1\ 0}{2\ 1\ 3\ 7\ 8} = 2*10^4 + 1*10^3 + 3*10^2 + 7*10^1 + 8*10^0$$

Why 10?
Because the no. is written in decimal no. system.

Base of decimal system = 10

Binary no. system

↳ Only 2 symbols (0, 1)
Base = 2
CPUs use it.

10110, 0101, 1111

$$\overset{4\ 3\ 2\ 1\ 0}{1\ 0\ 1\ 1\ 0} = 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0$$
$$= 16 + 0 + 4 + 2 + 0$$
$$= 22$$

Binary to Decimal

Expand it.

$$\underset{3\ 2\ 1\ 0}{1101} = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0$$

$$= 8 + 4 + 1$$

$$= \underline{13}$$

$$\underset{6\ 5\ 4\ 3\ 2\ 1\ 0}{1011010}$$

$$= 2^6 + 2^4 + 2^3 + 2^1$$

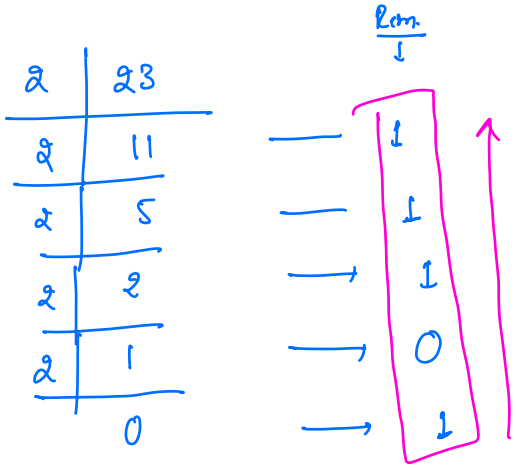
$$= 64 + 16 + 8 + 2$$

$$= (90)_{10}$$

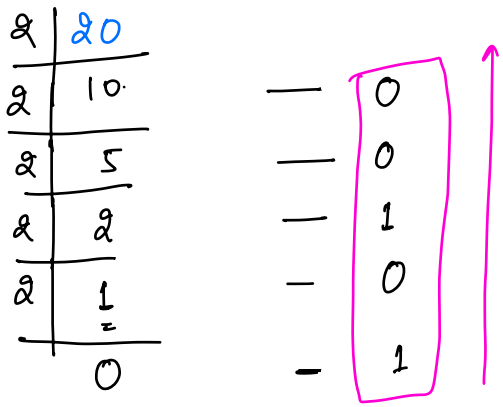
Decimal to Binary

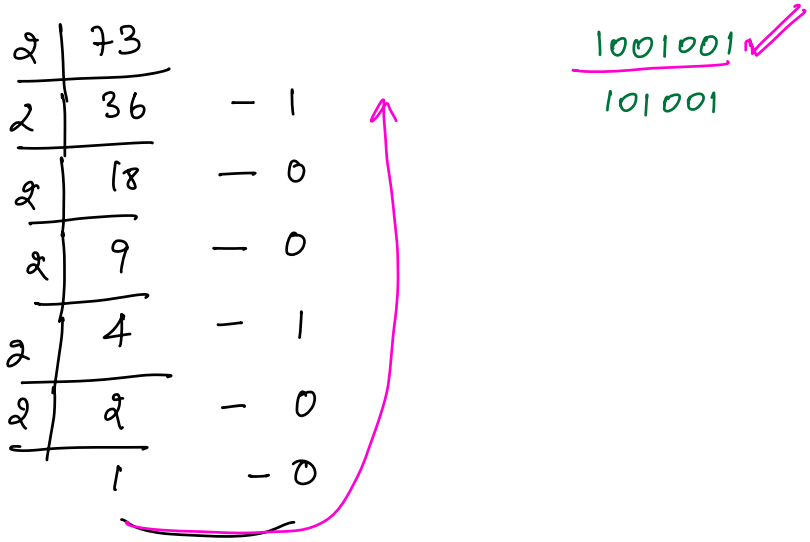
long Division method.

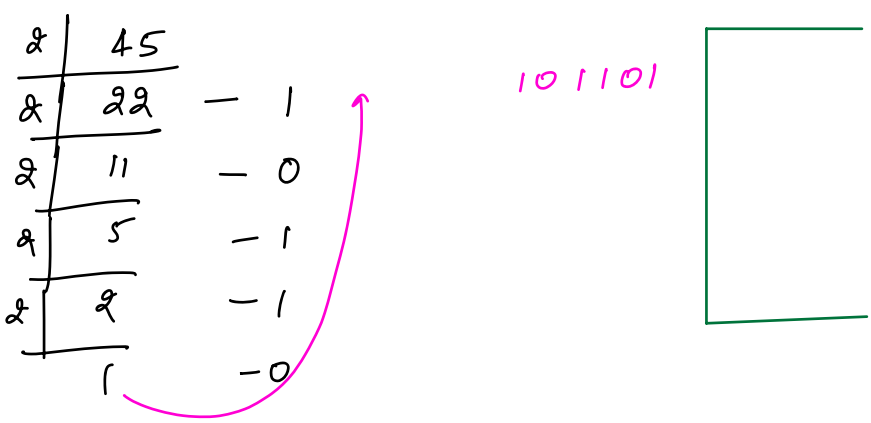
Keep dividing by 2 until you get 0.



$23 = 10111$


2 | 20
2 | 10
2 | 5
2 | 2
2 | 1
0
— 0
— 0
— 1
— 0
— 1
10100


2 | 73
2 | 36
2 | 18
2 | 9
2 | 4
2 | 2
1
— 1
— 0
— 0
— 1
— 0
— 0
1001001
101001


2 | 45
2 | 22
2 | 11
2 | 5
2 | 2
1
— 1
— 0
— 1
— 1
— 0
101101

Addition

(Decimal nos.)

```
  1 1
  3 6 8
(+) 4 5 3
-----------
  8 2 1
    11
```

8+3 = 11

11 = 10 + 1

11 % 10 = 1

11 / 10 = 1

1+6+5 = 12

12 = 10+2

12 % 10 = 2

12 / 10 = 1

```
    1 1 1
    3 7 8 9
(+) 1 4 7 8
-----------
  5 2 6 7
```

9+8 = 17

17 % 10 = 7

17 / 10 = 1

1+8+7 = 16

16 % 10 = 6

16 / 10 = 1

7+4+1 = 12

12 % 10 = 2

12 / 10 = 1

Binary Addition

```
  1 1 1 1 1 1
  1 1 0 0 1 0 1
(+) 0 0 1 1 1 1 1
-----------------
  1 0 0 0 0 1 0 0
```

1+1 = 2

2 % 2 = 0

2 / 2 = 1

1+1+1 = 3

3 % 2 = 1

3 / 2 = 1

0+0 = 0

0+1 = 1

1+0 = 1

1+1 = 0 with carry 1

```
  2 2 3 2 1
    1 0 1 1 0
    0 0 1 1 1
(+)   1 1 0 1
    0 1 1 1 1
-----------------
  1 0 0 1 0 0 1
```

1+1+1+1 = 4

4 % 2 = 0

4 / 2 = 2

6 % 2

5 % 2 =

```
    1 1
  1 0 1 1 0
(+) 0 0 1 1 1
-------------
  1 1 1 0 1
```

Break till 8:03 AM

Bitwise operators

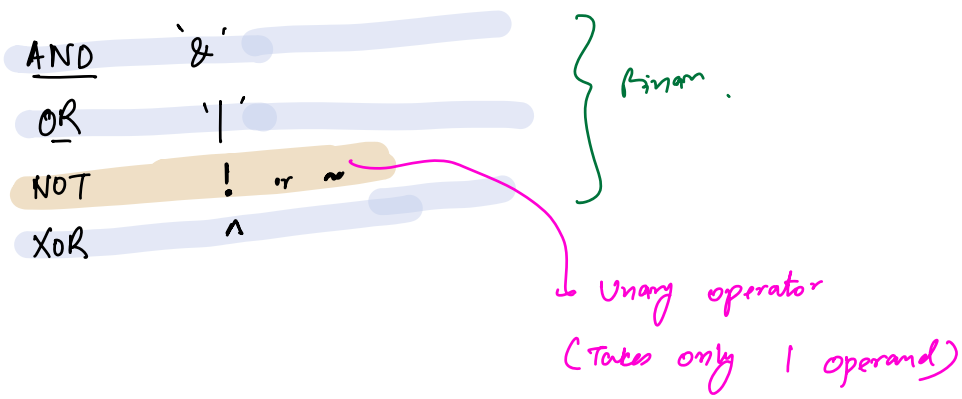
+ - * / % → operators
↳ that acted on the complete no.

734 + 628 =
729 * 73 =

Bitwise operator → operators act on each bit.

Bitwise ops.

↳



NOT

flips the bit.

NOT 0 = 1
NOT 1 = 0

~0 = 1
~1 = 0

~1001
= 0110

NOT 9 (in 4 bits) = 6

OR ⟶ Symbol |

Truth table

x	y	x | y
0	0	0
0	1	1
1	0	1
1	1	1

If any one operand, is true (1), the result is true (1).

$13 | 10 = 15$

```
  1101
(OR) 1010
-------
  1111
```

Identity of OR ?

a (op) ? = a

a OR ? = a

0

Identity of OR is 0.

+ → 0

a + ? = a

1 or 0 = 1

0 or 0 = 0

x + ? = x

x + 0 = x identity of + is 0

x | ? = x

x | 0 = x

identity of OR is 0.

x * ? = x

x * 1 = x

identity of * is 1

AND &

Truth table.

x	y	x & y
0	0	0
0	1	0
1	0	0
1	1	1

{ AND result is true only if both are true. }

13 & 10 = 8

```
  1101
& 1010
------
  1000
```

Identity of &

x & ? = x

x & 1 = x

1 is the identity of &.

0 & 1 = 0

1 & 1 = 1

XOR operator

`^`

Truth table.

x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive OR
↳ only one should be true.

Identity of XOR

$x \wedge ? = x$

0 is an identity, because.

$1 \wedge 0 = 1$
$0 \wedge 0 = 0$

```
  1101
^ 1010
------
  0111
```

$13 \wedge 10 = 7$

$20 \wedge 45 = 57$

```
  010100
^ 101101
--------
  111001
```

Truth table

x	y	~x	x & y	x | y	x ^ y
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

* Even/odd using Bitwise operator

12 & 1

= 1100

& 0001

00000

27 → 11011

(MSB) ← → Least significant Bit (LSB)

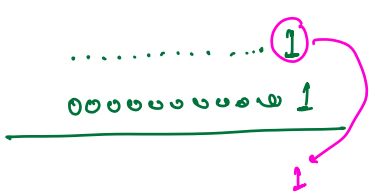
2 | 27

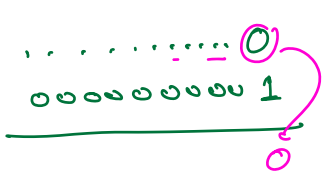
13 → 1

Even nos. → LSB = 0

Odd nos. → LSB = 1

A & 1





A & 1 = 1 if A is odd

A & 1 = 0 if A is even

$34 \& 1 = 0$

$33 \& 1 = 1$

Properties of OR, AND XOR

	OR	AND	XOR
0 (op) A	A	0	A
1 (op) A	A if A is odd A+1 if A is even	1 if A is odd (Useful) 0 if A is even	A − 1 if A is odd A + 1 if A is even.
A (op) A	A	A	0 (Useful)

A is some no.

$37 \text{ OR } 0 = 37$

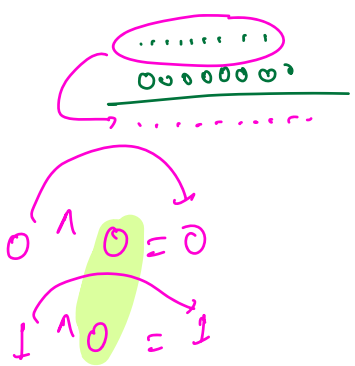
$37 | 1 = 37$

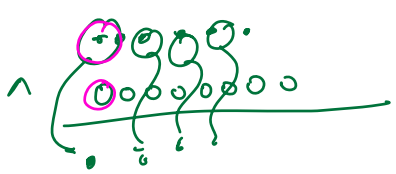
$38 | 1 = 39$

$39 | 1 = 39$

.....

0000000 1





$0 \wedge 0 = 0$

$1 \wedge 0 = 1$

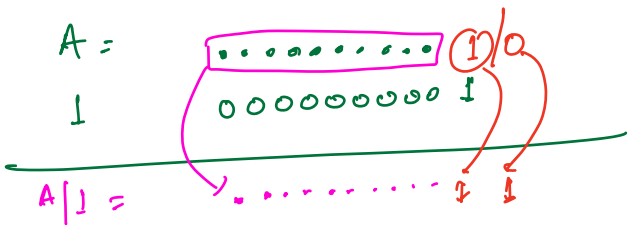
$1 \wedge 0 = 1$

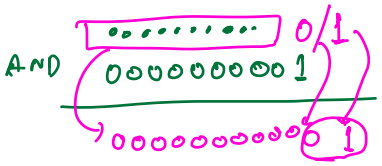
$0 \wedge 0 = 0$

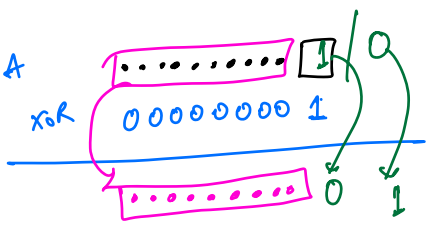
$37 = 100101$

$\wedge \quad 000000$

100101 = 37







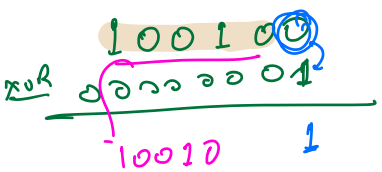
37 ^ 1 = 36

36 ^ 1 = 37

(37) → 100101 = 37
xor 000001
100100 = 36

A-1 if A is odd
A+1 if A is even.

11p01101
^ 11001101
00000000



Associative & Commutative property.

$a \wedge b = b \wedge a$ $\qquad$ $(a \wedge b) \wedge c = a \wedge (b \wedge c)$

$a | b = b | a$ $\qquad$ $(a | b) | c = a | (b | c)$

$a \& b = b \& a$ $\qquad$ $(a \& b) \& c = a \& (b \& c)$

Q. Given a N length array where every no. appears twice, except 1 number which appears only once. Find that no.

{ 1, 1, 3, 4, 5, 5, 4, 6, 3, 7, 6 }

Ans = 7

Approaches

1. Brute force :

   Nested loop to count = $O(N^2)$

2. In one iteration, store the count of each no. (hashmap).

   T.C. → $O(N)$

   S.C. → $O(N)$

3. $A \wedge A = 0$

$(5 \wedge 5) \wedge 6 = 0 \wedge 6 = 6$

$5 \wedge 5 \wedge 4 \wedge 4 \wedge 6 = 6$

$5 \wedge 4 \wedge 5 \wedge 4 \wedge 6 = 6$

{ 1, 1, 3, 4, 5, 5, 4, 6, 3, 7, 6 }

```
ans = 0
for(int i=0; i<n; i++)
{
        ans = ans ^ arr[i]
}
return ans
```

ans = 1

ans = 0

ans = 3

ans = 3 ^ 4

Pls revise before next class.

"Bit masking"

↓

filters