Number system

→ **Decimal no. system**, ~~range~~ digits 0-9, Base/Radix = 10

(weighted no. system) eg. $15{\cdot}25 = 1\times10^1 + 5\times10^0 + 2\times10^{-1} + 5\times10^{-2}$

→ **Binary no. system** digits 0, 1

no.s are combinations of 0 & 1.

(weighted) eg. 1011, 10111·110 etc.

[→ Roman & Gray no. system are non weighted.]

In Binary, base/radix = 2

weight is a power of 2.

A Binary digit is also called a bit.

eg. $1011 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = (11)_{10}$

$10111{\cdot}110 = 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2} = (?)_{10}$

its called Binary to decimal conversion.

$\boxed{\text{try:}}$

$101{\cdot}01 = (?)_{10}$

$1011{\cdot}1 = (?)_{10}$

$0111{\cdot}0111 = (?)_{10}$

Generalising: for a radix r

digits ω can be $0, 1, 2, \cdots (r-1)$

weight will be r^K

→ $K \geq 0$, for integer part

→ $K < 0$, for fractional part

→ **Binary to decimal conversion:**

$$D = \sum_{i=-m}^{n-i} b_i 2^i \quad \text{(formula)}$$

$$B = b_{n-1} b_{n-2} \cdots b_1 b_0 . b_{-1} b_{-2} \cdots$$

$\boxed{\text{try!}}$ $0'0101 = (?)_{10}$,

$0'00001 = (?)_{10}$,

Decimal to Binary Conversion

→ Take care of integer & fractional part separately.

integer part: $(25)_{10} = (?)_2$, $(8)_{10} = (?)_2$

```
2 | 25
2 | 12 - 1
2 | 6  - 0
2 | 3  - 0
    1  - 1
```

$(11001)_2$

```
2 | 8
2 | 4 - 0
2 | 2 - 0
    1 - 0
```

$(1000)_2$

for fractional part: Repeatedly multiply the fraction 2; after every multⁿ take out the integer part,

eg. $(0.634)_{10} = (\,?\,)_2$

$0.634 \times 2 = \underline{1}.268$

$0.268 \times 2 = \underline{0}.536$

$0.536 \times 2 = \underline{1}.072$

$0.072 \times 2 = \underline{0}.144$

$0.144 \times 2 = \underline{0}.288$

⋮

follow this order to take out the integer

$(.10100\ldots)_2$

the more no. of digits you calculate, the more accurate is your answer.

→ $\boxed{\text{Try}}$ $(0.36)_{10} = (\,?\,)_2$, $(0.135)_{10} = (\,?\,)_2$

$(5.53)_{10} = (\,?\,)_2$, $(6.63)_{10} = (\,?\,)_2$

Arithmatic operation with binary no.s:

Addition:

0+0 = 0
0+1 = 1
1+1 = 10
10 + 1 = 1[illegible] & so on

eg.

```
  101101
  110010
 --------
 1011111
```

carry

try

```
  1011
+ 1111
------
    ?
```

,

```
  11111
+ 11111
-------
     ?
```

Subtraction:

```
  11101
- 10110
-------
  00111
```

try

```
  1000
- 0111
------
    ?
```

```
  1111
- 0111
------
    ?
```

Multiplication: 1101 X 1011

try 1111 X 1110 = (?)

```
      1101
   X  1011
  --------
      1101
     1101X
    0000XX
   1101XXX
  --------
  10001111
```

try → Multiply 15 X 24 using binary format.

Number system	Different Digit
Decimal (radix = 10)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Binary (radix = 2)	0, 1
Octal (radix = 8)	0, 1, 2, 3, 4, 5, 6, 7.
Hexadecimal (radix = 16)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

→ Decimal to represent a no. is less than binary
say 8 is 1000 in binary.
this is the disadvantage of binary no. system sometimes.

→ Compact form is octal & hexadecimal.

Octal

→ weighted no. system with radix = 8.

→ radix = 8 is some powers of 2. $(= 2^3)$

→ every 3-bit binary no. can be represented as a octal no.

eg.

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

What will be the next

Binary to Octal conversion

integer part.
→ Right to left. & make a group of 3 bits & replace it with octal

eg. $(101101000)_2 = (550)_8$

fractional part
left to right & make group of 3 bits & replace it with octal

$(0'100011100)_2$

$= (0'434)_2$

$\boxed{\text{try}}$ eg. $(10101'1011)_2 = (?)_8$

eg. $(1000)_2 = (?)_8$ $(1001'01)_2 = (?$

So octal no.s will be like —

0 1 2 3 4 5 6 7

10 11 12 13 14 15 16 17

20 21 27

30 31 . . . . . . & so on

Similarly Octal to Binary Conversion

$(164)_8 = (001\ 110\ 100)_2$

→ Take a octal no. & represent it with 3 binary bits

$\boxed{\text{try}}$ eg. $(1152'61)_8 = (?)_2$, $(56'45)_8 = (\ )_2$

* eg. $(78'74)_8 = (?)_2$

Decimal to Octal conversion

~~Note~~ do the same method as ~~bina~~ decimal to b

eg. $(3762)_{10} = (\ ?\ )_{2\ to\ 8}$

$$
\begin{array}{r|l}
8 & 3762 \\
8 & 470 - 2 \\
8 & 58 - 6 \\
 & 7 - 2
\end{array}
\uparrow \quad = (7262)_8
$$

eg. $(0.356)_{10} = (?)_8$

$0.356 \times 8 = 2.848$

$0.848 \times 8 = 6.784$ $\downarrow$

$\vdots$

$= (0.26)_8$

$\boxed{\text{try}}$ eg. $(56.25)_{10} = (?)_8$

$(79.76)_{10} = (?)_8$

Octal to Decimal

eg. $(7262)_8 = (?)_{10}$

$= 7 \times 8^3 + 2 \times 8^2 + 6 \times 8^1 + 2 \times 8^0$

$= (37\cancel{6}2)_{10}$

$\boxed{\text{try}}$

$(674.61)_8 = (?)_{10}$

$(32.14)_8 = (?)_{10}$

Scanned by CamScanner

Hexadecimal no. system

Radix = 16 {power of 2 (= 2^4)}

→ 0, 1, 2, . . . 9, A, B, C, D, E, F

→ 4 digits are sufficient to write a hexadecimal no. from 0–F.

0 0000
1 0001
2 0010
⋮
A 1010
B 1011
⋮
F 1111

write down

→ Conversion is similar to octal but take 4 bits instead of 3

Binary to hexadecimal

eg. ~~10110~~ $(\underbrace{1011}\ \underbrace{0111}\ \underbrace{1010})_2 = (B7A)_{16}$

$(0.\underbrace{1000}\underbrace{010})_2 = (0.84)_{16}$

$\boxed{\text{try}}$ $(101.0101\ 111)_2 = (\ ?\ )_{16}$

$(101.11)_2 = (\ ?\ )_{16}$

Hexadecimal to binary

$(3A5)_{16} = (0011\ 1010\ 0101)_2$

$(1{\cdot}8)_{16} = (0001 . 1000)_2$

$\boxed{\text{try}}$ $(12{\cdot}35)_{16} = (\ ?\ )_2$

Decimal to Hexadecimal

~~tool~~ $(55)_{10} = (\ ?\ )_{16}$

$16 \,\underline{|\ 55}$

$\quad ?$

$\boxed{\text{try}}$ $(238{\cdot}56)_{10} = (\ ?\ )_{16}$, $(14{\cdot}15)_{10} = (\ ?\ )_{16}$

Hexadecimal to decimal

$(AB0)_{16} = (\ ?\ )_{10}$

$= A \times 16^2 + B \times 16^1 + 0 \times 16^0$

$= 16^2 A + 16 B + 0$

$= 16^2 \times 10 + 16 \times 11 + 0$

$= ?$

$\boxed{\text{try}}$ $(FA9)_{16} = (\ ?\ )_{10}$, $(FBB.01A)_{16} = (\ ?\ )$

Octal to hexadecimal no. system

$(56'55)_8 = (?)_{16}$

→ first convert it to binary & group it into 4 bits & ~~to~~ assign a ~~b~~ hexadecimal bit to each group.

$(\underbrace{101}\,\underbrace{110}\cdot\underbrace{101}\,\underbrace{101})_2 = (\underbrace{0010}\,\underbrace{1110}\cdot\underbrace{1011}\,\underbrace{0100})_2$

$= (2E'B4)_{16}$

$\boxed{\text{try}}$ $(576'11)_8 = (?)_{16}$

Hexadecimal to Octal no. system

$(FF'34)_{16} = (?)_8$

→ convert to binary & ~~assign~~ group 3 bits, assign a octal no. to each group.

$(\underbrace{1111}\,\underbrace{1111}\cdot\underbrace{0011}\,\underbrace{0100})_2 = (\underbrace{011}\,\underbrace{111}\,\underbrace{111}\cdot\underbrace{001}\,\underbrace{101}\,\underbrace{00}$

$= (377\cdot150)_8$

→ $\boxed{\text{try}}$ $(6BB\cdot56)_{16} = (?)_8$

$(567\cdot77)_8 = (?)_{16}$

Signed and unsigned binary no. system

* Circuits are implemented using transistors.
  transistors act like switch.

  on switch → current flows → '1'
  off switch → no current → '0'

Bit → Single binary digit (0/1)
nibble → collection of 4 bits
Byte → collection of 8 bits

eg. Gigabyte, terabyte :—

* Any no. can be signed or unsigned

(magnitude as well as sign) (only magnitude, no sign)

Unsigned binary no.

→ for n bits we have 2^n distinct combinations.

→ minimum 0, maximum $2^n - 1$

(like in decimal 0–9
octal 0–7 etc.)

* $b_{n-1}\ b_{n-2} \cdots b_2 b_1 b_0$

  b_{n-1} ↓ MSB, b_0 ↓ LSB

* Unsigned binary no. in 4 bits

+0	0000
+1	0001
⋮	⋮
+15	1111

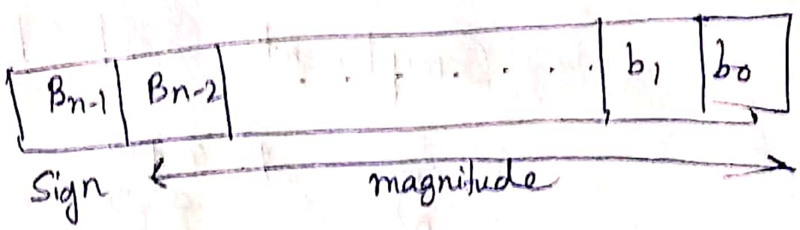
* Signed binary no. : 3 possible ways —
  1. Sign-magnitude representation
  2. 1's complement representation
  3. 2's " "

1. Sign-magnitude representation : for a n bit no.

→ MSB indicates sign (0 : +ve
1 : -ve)

→ remaining $(n-1)$ bits represent magnitude of the no.

→ Range $-(2^n-1)$ to $+(2^n-1)$



→ issue : ~~two~~ 0 is represented by two different bit pattern. one +0 , one -0 . → drawback.

Sign magnitude no. representation in 4-bits:

decimal			
+0	0000	-0	1000
+1	0001	-1	1001
+2	0010	-2	.
+3	0011	-3	.
+4	0100	.	.
+5	.	.	.
+6	.		
+7	0111	-7	1111

1's complement representation

→ +ve no.s are represented exactly as in sign-magnitude form.

→ -ve no.s are represented in 1's complement form.

Computing 1's complement of a no.

* Complement each bit of the no.

$5 \xrightarrow{\text{1's comple.}} -5$

$0101 \xrightarrow{\text{1's comp.}} 1010$

$0111 \xrightarrow{\text{1's comp.}} 1000$

Q. find ~~Ca~~ −4 representation using 1's complement

→ $+4 = 0100$

$-4 =$ 1's complement of 0100

$= 1011$

for n=4

decimal	1's complement
+0	0000
+1	0001
+2	0010
+3	.
.	.
.	.
+7	0111

decimal	1's complem
−7	1000
−6	1001
−5	1010
−4	1011
−3	1100
−2	1101
−1	1110
−0	1111

Range : $+(2^{n-1}-1)$ to $-(2^{n-1}-1)$

→ Same drawback → two different representation of 0.

→ advantage: subtraction can be done using addition (we will see later on)

2's complement representation

→ extention of 1's complement representation
→ widely used representation.
→ +ve no.s are represented exactly in sign-magnitude form.
  -ve no.s are represented in 2's complement form.
→ How to compute 2's complement of a no.
  * complement every bit of no. (or 1's comple
  * add 1 to the resulting no.

→ Represent -4 using 2's complement (n=4)

+4 = 0100

-4 = 2's complement of 0100
= 1011 + 1
= 1100

Decimal	2's com		
+0	0000	-8	1000
+1	0001	-7	1001
		-6	1010
.		-5	1011
.		-4	1100
:		-3	1101
+7	0111	-2	1110
		-1	1111

eg.. -0 = 0000 2's complement
= 1111 + 1
= 0000 in 4 bits

→ Range of no.s that can be represented in 2's complement:

max m: $+(2^{n-1}-1)$

min m: -2^{n-1}

→ All computers use 2's complement representation to store negative no.s

Some features of 2's complement

* MSB has weight of -2^{n-1}

-2^{n-1}	2^{n-2}		2^1	2^0
b_{n-1}	b_{n-2}	. . . . . .	b_1	b_0

: weights.

$D = -b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \dots + b_1 2^1 + b_0 2^0$

eg. $0101 = 5$

$1101 = -8 + 4 + 0 + 1 = -3$

* Shift left a no. by k position, multiplies the no. by 2^k.

eg. $00010011 = +19$

Shift left by 2 : $01001100 = +76$

$19 \times 2^2 = 76$

eg. $11100011 = -29$

Shift left by 2 : $10001100 = -116$

$-29 \times 2^2 = \underline{-116}$

* Shift right by k position divides the no. by 2^k.

eg. 00010110 = +22

Shift right by 2 :

00000101 = +5

eg: 11100100 = -28,

shift right by 2 :

11111001 = -7

(Since this is a -ve no., donot pad with 0, pad it with sign bit)

* Sign bit can be copied as many times as required in the begining to extend size of no. This is called sign extension.

eg:
n=8

X = 00101111 $(+47_{10})$

sign extended upto n=32

00000000 00000000 00000000 00101111

$= 47_{10}$

eg: X = 1010011 $(=-93)_{10}$, n=4

if n=32,

11111111 11111111 11111111 1010011

$= (-93_{10})$