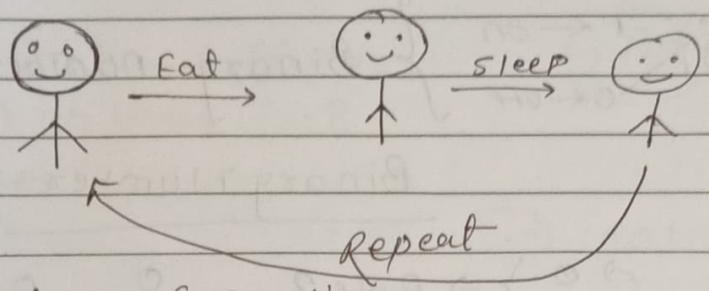
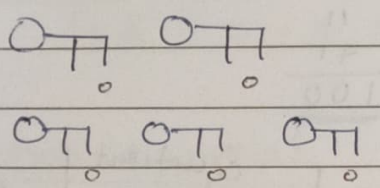
Data structure & Algorithm

1. Introduction To Programming



Evolution of counting



In Ancient time people used stone to count.

↓↓

Suppose 500 goats are there then we can't take 500 stone to our pocket. so the need of counting came into picture.

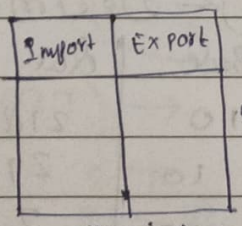
Egypt developed first counting system → base 60

India developed counting system → base 10

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

3 + 4 = 07 (round)

7 + 4 = 11 (one round)



Register

profit / loss

↓↓

for large register

↓↓

calculation very difficult

Accuracy ↑

Time ↓

→ computer

↑

To calculate

Mechanical computer
Accuracy ↑
fast ✗.

Transistors

1 ← on
0 ← off
} Binary number

Binary Number → Decimal

(2)

0
1
} ⇒ Base 2

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

(3) 10 + 1 = 11

(4) 11 + 1 = 100

27 ← Binary.

base 2 Base

Base	Quotient	Rem
~~2~~	~~27~~	
2	27	1
2	13	1
2	6	0
2	3	1
2	1	1
	0	

11011

43

Base	Q	Rem
2	43	1
2	21	1
2	10	0
2	5	1
2	2	0
2	1	1
2	0	

101011

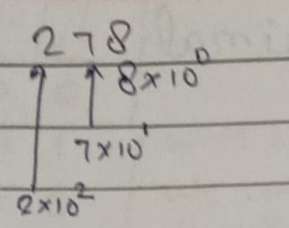
$1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

$32 + 0 + 8 + 0 + 2 + 1 = 43$.

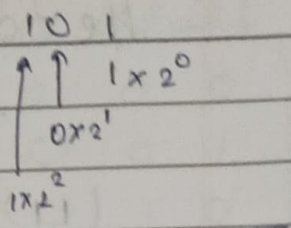
278 ← Decimal

Base	Que	Rem
10	278	8
10	27	7
10	2	2
	0	

278.



200+70+8 = 278



4+0+1 = 5

Decimal → Binary (H/W)

(i) 37 (ii) 92 (iii) 128

Binary → Decimal

(i) 1011 (ii) 111001 (iii) 10011011

Octal

↳ 8 (0, 1, 2, 3, 4, 6, 7) Base 8

Hexadecimal

↳ 16 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

Base 16

Octal → Decimal

Base	Quotient	Rem
8	23	
8	2	7 ↑
	0	2 ↑

$(23)_8 \rightarrow (27)_{10}$

$(27)_0 \rightarrow$

27

7×8^0

2×8^1

$16 + 7 = (23)_8$

Decimal to Octal.

Hexa → Decimal

base 16

$(11)_{10} \longrightarrow (B)_{16}$

B → 11

$11 \times 16^0 \rightarrow 11 \times 1 = (11)$.

$(11)_{16}$

1 1

$1 \times 16^1 + 1 \times 16^0$

$16 + 1 = (17)_{10}$

$(13)_{16}$

$1 \times 16^1 + 3 \times 16^0$

$16 + 3 = (19)_{10}$

$(A\ C\ 2)_{16}$

10×16^2 | 12×16^1 | 2×16^0

$2560 + 192 + 2 = (2754)_{10}$

0
1
2
3
4 — X
5
6
7 — ✓
8
9

Here only single Digit we can represent

0 — off
0 — off
0 — off
0 — off
0 — off
0 — off
0 — ON
0 — ON
0 — off
0 — ON

Here we can represent any digit no; so Binary is used.

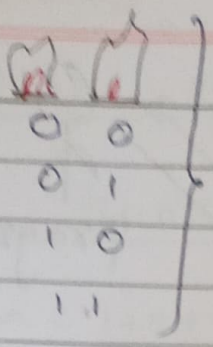
(13)

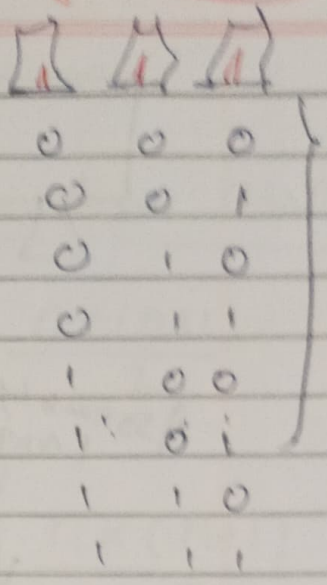
1101

00000 - 01101

11

(13),





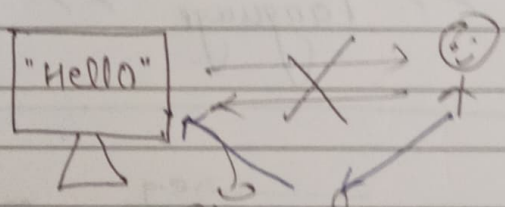
Moore's law.

Every two year, capacity of transistors will be double.

1MB → 2MB → 4MB → 8MB

2year 2year 2year.

Transistor's size is decreasing and space is increasing.

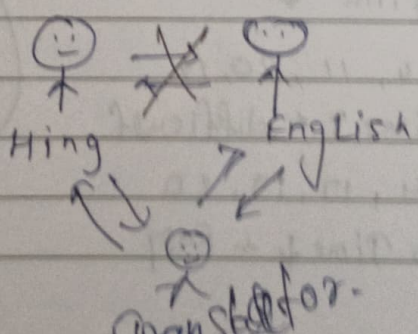


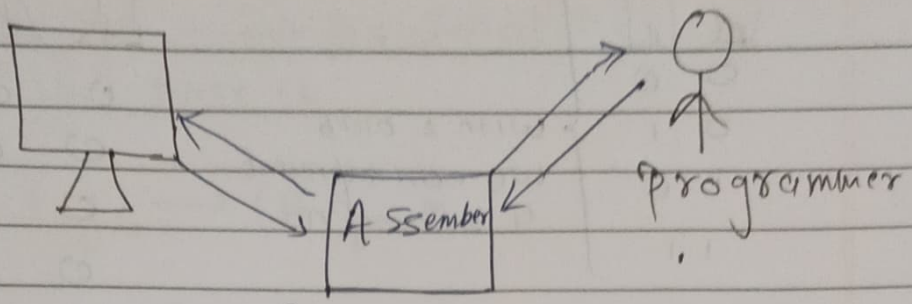
Binary form

Machine language

01011010110

Binary code: 128+ 65+97+ .... 00 no.



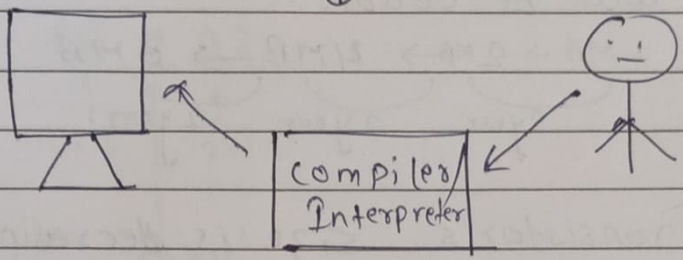


Assembly language { ADD 23 78
SUB 73 24

↓

High level language → 2+3
→ 5-4

↓



Machine language >> Assembly language >> High level language.

→ Slow

⇒ Data → e.g

↓

Store → fetch

↓

- Space ↓
- Time ↓

Unsorted → 13, 17, 14, 11, 20, 19

fetching → (6) Time ↑ ← difficult

Sort → 11, 13, 14, 17, 19, 20

fetching → (4) → Time ↓ ← Easy

Shop

Pulse
sugar
Flour
:

Randomly

↓

~~Very~~ fetching very difficult

Arranged

↓

fetching easy.

Need of DSA

Huddu Bhaiya	Bablu Bhaiy
Sum : 1+2+3+4+ . . . . + . . 100	Sum = $\frac{n \times (n+1)}{2}$
100 lac ↓ 1+2+3+4+5+6+7+ . . . 1000	$\frac{n \times (n+1)}{2}$
3 days	↓↓ 10sec.

Suppose, we have ~~two~~ two person huddu Bhaiya and babla bhaiya and given a task to calculate sum of n natural no.
There is two scenario in above example

(i) huddu bhaiya will calculate
1+2+3+ . . . .

(ii) Babla bhaiya is intelligent and he will calculate with formula
$\frac{n \times (n+1)}{2}$

if both scenario there is solution of of same problem and correct also but solution (i) is very time consuming where as solution (ii) ~~can be~~ is very easy and less time cosuming. Here Plays DSA a measure role for reducing time complexity.