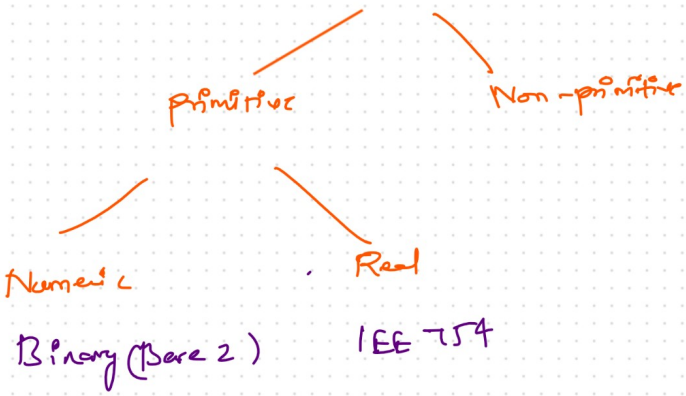
Data Types



Bits : used by computers to represent integers & real numbers

i.e 2 digits → 0 & 1 [powers of 2]

Positive integers : represented directly by converting decimal value into powers of 2

8 4 2 1

Ex: decimal number 5 ——— 0 1 0 1

Revised

$0101 \longrightarrow 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

$\Rightarrow 4 + 1$

$= 5$

Negative Integers : typically represented using a technique called 2's complement

1 Byte = 8 bits −128 0 +127

Ex: −5 in 8-bit system

1	1	1	1	1	0	1	1

2's compliment

Step1 5 in 8-bit system = 00000101

128	64	32	16	8	4	2	1
0	0	0	0	0	1	0	1

Step2 1's complement [flip all the bits i.e 1 → 0, 0 → 1]

MSB
1 — −ve
0 — +ve

0	0	0	0	0	1	0	1

left-most bit = MSB

1	1	1	1	1	0	1	0

Step3 Add 1 bit to the 1's complement

```
  1 1 1 1 1 0 1 0
              1
-------------------
  1 1 1 1 1 0 1 1
```

Verification (Reversed)

(1) 1111 1011 ⟶ −5

(2) Substract 1 bit :

```
  1111 1011
       1
-----------
  1111 1010
```

left most bit (MSB) → 1111 1010

(3) flip bits :

```
  1111 1010
-------------
  0000 0101
```

8 4 2 1
0 1 0 1
= 5

Real Numbers 5.625 ⟶ IEEE754 format (binary representation)

used for floating point arithmetic representation

for both single precision (32 bit) & double precision (64 bits)

Step1 : Convert integral part to binary

8 4 2 1
1 0 1

$5_{10} = 101_2$

Step2 : Convert fractional part to binary

5.625 ⟶ 0.625

0.625 to binary, repeatedly multiply by 2 & note the integer part

$0.625 \times 2 = 1.25$ — (1)

$0.25 \times 2 = 0.5$ — 0

$0.5 \times 2 = 1.0$ — (1)

Stop here

$0 \times 2 = 0$

$\therefore \quad 0.625 \longrightarrow 0.101$

Step 3 : Combine both integer & fractional part

$5.625_{10} = 101.101_2$

Step 4 : Normalize binary number in scientific notation → $1.xxxxx \times 2^n$ (n → exponent)

2

$101.101 \longrightarrow 1.01101 \times 2^2$

(1) move toward fraction & keep only 1 digit before point

(2) The exponent will be 2 cause we moved 2 bit

Step 5 : Now represent in IEEE 754 format (Single precision)

float f = (float) 5.625 32 bit

double d = 5.625 64 bit

Sign . exponent . mantissa

→ Sign bit : 0

→ exponent = $2 + 127 = 129_{10} \longleftrightarrow (10000001)_2$

→ Mantissa = The fractional part from step 4 then padded with 23 0's

i.e 01101000000....0

23

5.625

32 bit

Sign	exponent	fraction
0	10000001	01101000000....0
		23

bias of IEEE 754

Single precision : 127

double precision : 1023

8 bit

128	64	32	16	8	4	2	1
1	0	0	0	0	0	0	1

Mantissa

32 (single prec) : 23

64 (double prec) : 52

32 bit 5.125 — Binary format

0 10000001 01101 0000 - - - - 0

4 + 19 = 23

Reverse

0 10000001 01101 0000 - - - - 0

1 bit 8 bit 23

5 + 18 = 23

Convert back to real number

Exponent = 129 − 127 = (2) 1.01101×2^{n}

Normalized mantissa: 1.01101×2^{n}

$101.101_2 = (\quad)_{10}$

Binary fraction 0.101_2

$\Rightarrow 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$

$= 1 \times \frac{1}{2} + 0 \times \frac{1}{4} + 1 \times \frac{1}{8}$

$= 0.5 + 0 + 0.125$

$= 0.625$

$101.101_2 = 5.625$

8 4 2 1

5

Bit wise Not (!)

1) flip the bit
2) change the sign
3) Interpret 2's complement
   1) flip the bit (1's complement)
   2) add extra bit

$(-12)_{10}$ → 8 bit system

128	64	32	16	8	4	2	1
0	0	0	0	1	1	0	0

Step 1:

Step 2:

1	1	1	1	0	0	1	1

MSB → -ve

Step 3: 2's complement

1111 0011

(1) 1's complement [flip bits]

1111 0011

0000 1100

(2) Add 1 bit

0000 1100

+ 1

0000 1101

-8 4 2 1

(-13)