floating point

0.xxx

0.125 = 1.25 x 10^{-1}

6.000 = 6 x 10^{1}

5 000 000 = 5 x 10^{6}

General

Scientific notation

Sign bit mantissa

(+) 1.25 x 10^{1}

exponent

bits

bits

sign	exponent	mantissa/significand

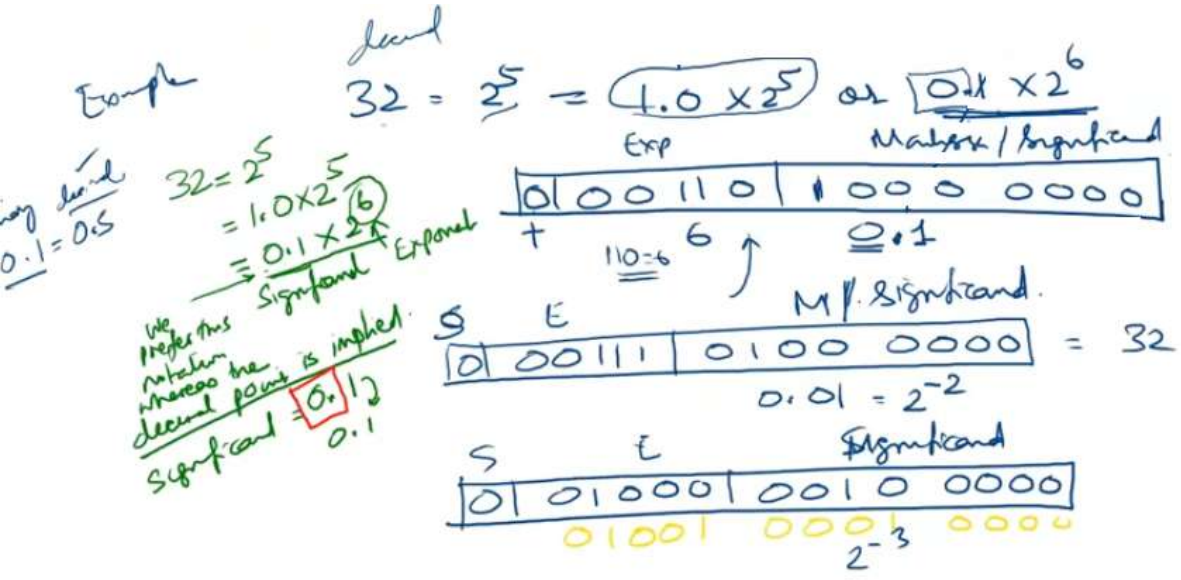
significand

no of bits

Size of mantissa determines — Precision

Size of exponent determines — Range

H/w | Computers | 10^{9} 10^{9}


Example
decimal
32 = 2^5 = 1.0 x 2^5 or 0.1 x 2^6
Exp
Mantissa / Significand
0 00110 1000 0000
+
110=6
6
0.1
32 = 2^5
= 1.0 x 2^5
= 0.1 x 2^6
Exponent
Significand
we prefer this notation where decimal point is implied
significand = 0.1
0.1
S E M/. Significand.
0 00111 0100 0000 = 32
0.01 = 2^-2
S E Significand
0 01000 0010 0000
2^-3

0.1 = 2^-1

representation).

To resolve:

- First digit of the significand must be 1.
- (normalization)

$0.1 \times \times \times \times \times \ldots$

$4.5_{10} = 100.1 \times 2^0$

$= 10.01 \times 2^1$

$= 1.001 \times 2^2$

$= 0.1001 \times 2^3$ ← normalization

- For negative exponents we use a "biased" exponent

Bias – number that is approximately in the range of values

0 ± 15

5 bit exponent = 32 bits

mid value = 16 (bias)

Excess 16 bit representation

In this model – exponent values less than 16 are negative, fractional nos.

$32 = 1.0 \times 2^5 = 0.1 \times 2^6$

we excess 16 biased exponent.

6 will be represented as $= 6 + 16 = 22$

so we have

Sign	Exponent	Significand
0	10110	1000 0000

biased on excess 16 repres.

$2^{22-16} = 2^6$

$0.1 = \pm 0.1 \times 2^6$

Ex. **0.0625 = 1.0 x 2^-4 = 0.1 x 2^-3**

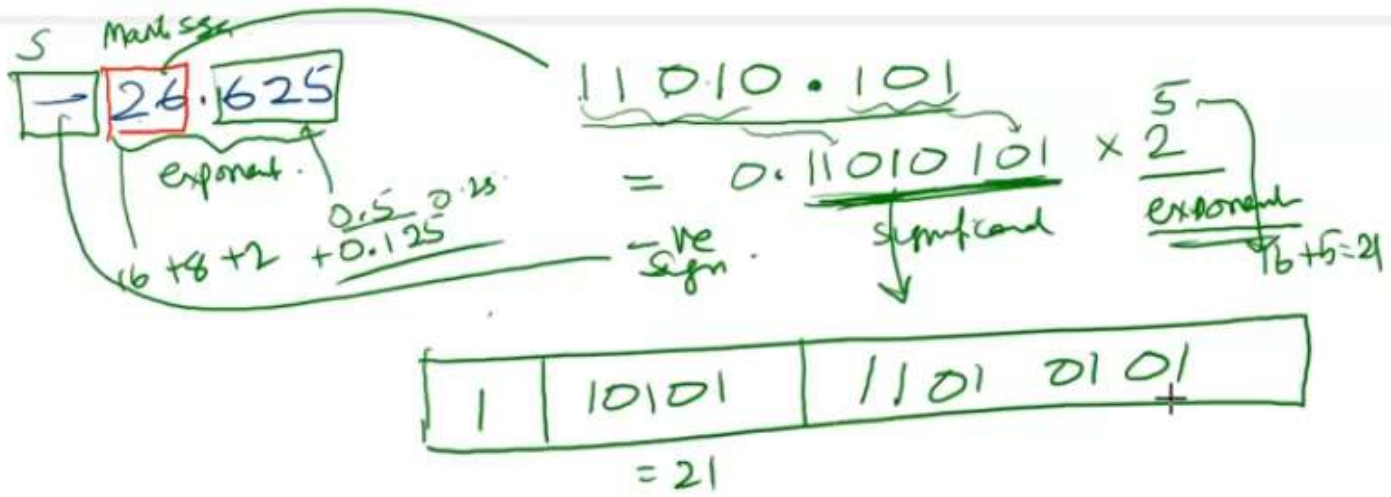
now in terms of excess 16 baised exponent = 16-3 = 13

sign bit	Exponent	Significand
0	01101	1000 0000

- The first digit of the significand should be / must be 1 with no ones to the left of the radix point.
  - this process is called normalization

0.1xxxxx. all significands or mantissa

$4.5 = 100.1 \times 2^0 = 1.001 \times 2^2$ or $= 0.1001 \times 2^3$


S
Mantissa
-
26.625
exponent.
0.5 0.25
16 +8 +2 +0.125
-ve sign.
11010.101
= 0.11010101 × 2^5
significand
exponent
16+5=21
1
10101
1101 0101
= 21

Standard

Single precision double precision

Standard is called IEEE 754.

IEEE 754 Single precision floating point Standard uses.

(32 bit) 8 bit exponent 23 bit for significand

Bias = 127

The largest no represented by 7 bit is 1111111 = 127

IEEE 754 double precision standard uses.

(64 bits) 11 bit exponent, 52 bit significand.

Bias = 1023.

In both — the significand has an implied 1 to the left of the radix point.

format is $1.XXXX$ instead of $0.1XXX$

$4.5 = 0.1001 \times 2^3$

IEEE Std format = 1.001×2^2

mantissa

0	2+bias	001
+ve +	exp.	Significand

S = 011

After the 0.5 +0.25

−3.75 IEEE single precision.

−3.75 = −11.11 = −1.111 × 2 (one shift)

= (1).111 × 2^1

Bias is 127 ∴ no is = 127 + 1 = 128

−(1).111 × 2^(128−127) = −1.111₂ × 2^1 = −11.11 = −3.75

value Bias

Range of fl number
for a 32 bit - 8bit exponent
$\pm 2^{256} \approx 1.5 \times 10^{77}$ } largest no

Accuracy → 23 bit significand → $2^{-23} \approx 1.2 \times 10^{-7}$
Abont 6 decimal places.

$2^{10} = 1024 \sim 1K$.

64 bit nos Range $\pm 2^{1024}$

Accuracy $\frac{1}{2^{-52}}$

bits

Decimal value of an IEEE no

$$(1-2s) * (1+f) * 2^{e-bias}$$

s, f, e = decimal.

(1-2s) = 1 or −1 (whether s = 0 or 1

fractional field; we add the implicit 1.

bias = 127 or 1023
SBP. DBP

1 | 01111100 | 11000.....0

s

e

0.5+0.25

0.75

124

s = 1, (−)

$(1-2^{1}) * (1+0.75) * 2^{124-127}$

$= -1 * 1.75 * 2^{-3} = -\frac{1.75}{8} = -0.21875$

$\frac{43}{8} = -0.21875$

Ex:

347.625

Binary .101

0.5 x 1
+ 0.25 x 0
+ 0.125 x 1

Binary to binary
Decimal to FP-

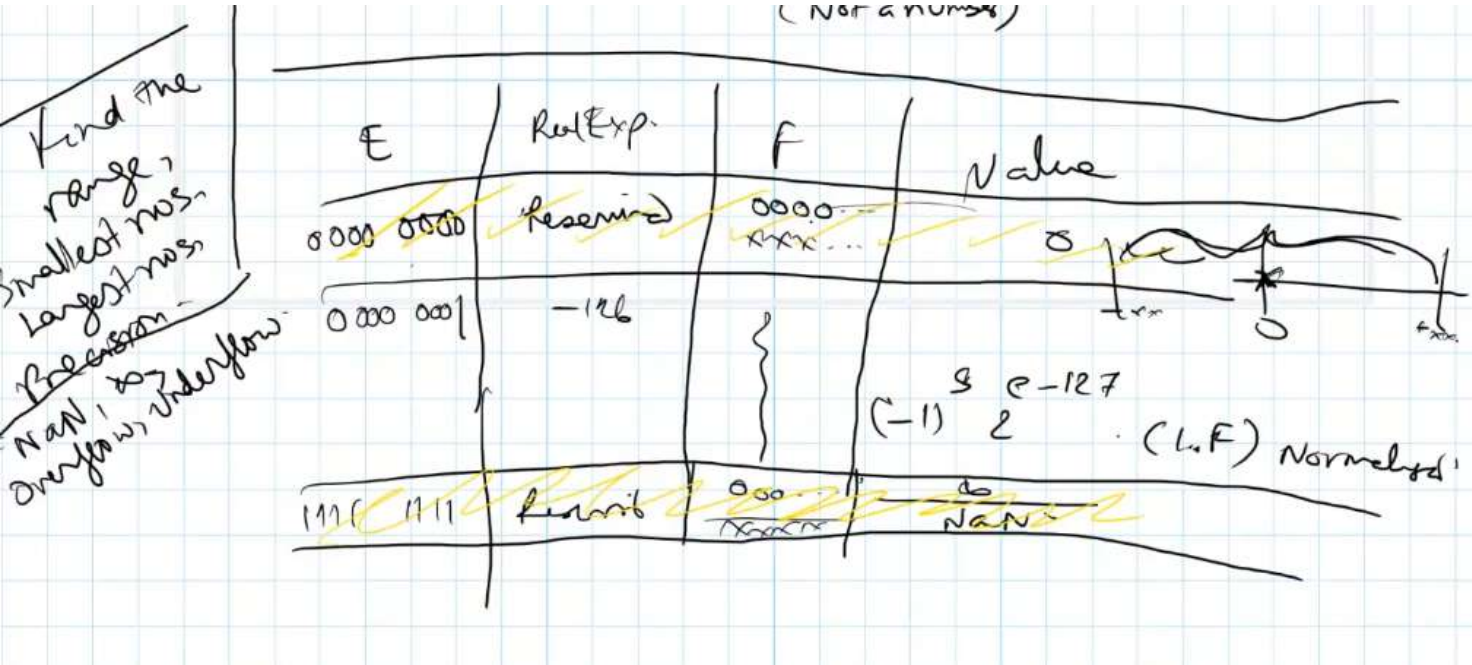
101011011.101×2^0

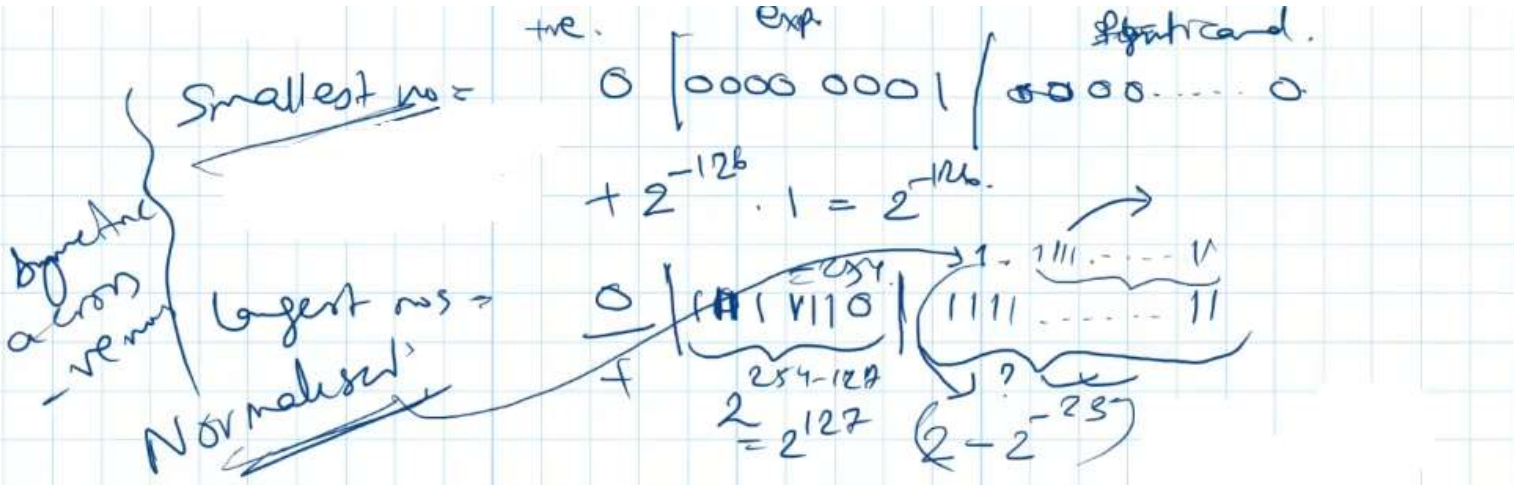
1.01011011101×2^8

$S = 0$, $exp = 8 + 127 = 135$, $01011011101\ldots$

Special values:

E	F	Meaning	Explanation
0000 0000	000...0	0	+0.0 or -0.0.
0000 0000	xxxx	Valid no.	$(-1)^S \times 2^{-126} (0.f)$ (unnormalized)
1111 1111	000000...	∞	
1111 1111	xxxxx	NaN. (Not a number)	


Find the range, Smallest nos, Largest nos, Precision - NaN, Overflow, Underflow
E
Real Exp.
F
Value
0000 0000
Reserved
0000...
xxx...
0
0 000 000
-126
-127
(-1)^S 2
(1.F) Normalized
111 1111
Reserved
000...
xxxxx
to
NaN



Smallest and highest number in 32 bit floating point

int x = 33554431 ← In float it can't be represented, (causes issue)

float y = ·-

0.10 = 0.0001 0011 0011 ...

This 0.10 is not represented in floating point representation

H/W

Floating point * multiplication & Addition:-