Fixed point → decimal position of representing a number is fixed

⇒ 12345.67
↳ 5 positions ↳ only 2 positions

$-12356. \times 10^{-1} = -12345.6 \times 10^{0}$

$= -1.23456 \times 10^{4}$ (normalised)

error = $\approx -0.12345 \times 10^{5}$
$\approx -0.01234 \times 10^{6}$

Sign Exponent significand

$(-1)^{sign} \times significand \times radix^{exponent}$

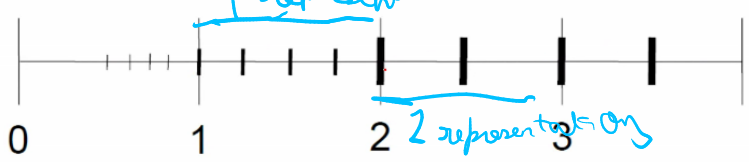
Distribution of Floating Point Numbers

- 3-bit mantissa
- Exponent: {-1,0,+1}

unequally distributed

e = -1	e = 0	e = 1
1.00 X 2^(-1) = 1/2	1.00 X 2^0 = 1	1.00 X 2^1 = 2
1.01 X 2^(-1) = 5/8	1.01 X 2^0 = 5/4	1.01 X 2^1 = 5/2
1.10 X 2^(-1) = 3/4	1.10 X 2^0 = 3/2	1.10 X 2^1= 3
1.11 X 2^(-1) = 7/8	1.11 X 2^0 = 7/4	1.11 X 2^1 = 7/2

4 representations → binary



Fixed point: Represent @ less, Floating point: Represent more but unbalanced

1.00 → 1
1.01 → 1.25
1.10 → 1.5
1.11 → 1.75

more bits for significand ⇒ more accuracy

More bits for exponent increases range

$-101.001101 * 2^{111001} = -1.01001101 * 2^{111011}$ (normalized)

if significand $\in (0,1]$ (without sign) then it's normalized

For decimal

Significand $\in (0,1]$

0.001×10^{0} to 9.999×10^{99}

radix 10, exp 2, Significand 4 places

fixed point is much less even with all 6 ~~bits~~ places for integer and 0 for decimal

max number possible = 1000 / 999999

IEEE 754 FPR

approx of Real numbers - float double

approx of Integers - int

float = 32 bit single precision

double: 64 bit double precision

GCC treats [float constant] as double precision

Sign magnitude ✗ not 2's complement

31	30 - 23	22 - 0
sign	exp	significand

31	30 - 20	19 - 0 upper
~~sign~~		31 - 0 lower

→ sign 0 - positive 1 - negative

→ Mantissa is normalized, always has a leading bit binary-point 1 bit $(1.0 \leq |significand| < 2)$

'0' is a special case

Exponent biased exponent = actual + Bias

Ensures exponent is unsigned

bias of 127 for single precision

1023 for double p

Why not sign bit?

(1) You need 2's complement for signed bit (unsigned is easier to handle)

(2) Special numbers cannot be represented eg: 0, NaN etc.

Normalized

Types of Data

- Data represented in this format are classified in five groups.
  - Normalized numbers,
  - Zeros,
  - Subnormal(denormal) numbers,
  - Infinity and not-a-number (nan).

- **Single Precision data interpretation**

Single Precision		Data Type
Exponent	*Significand*	
0	0	± 0
0	nonzero	$\pm$ subnormal number
1 - 254	anything	$\pm$ normalized number
255	0	$\pm\infty$
255	nonzero	NaN (not a number)

if we used sign:

-128 to 127

we may use -128 for '00' } or other way
127 for '0' }

our range is -127 to 126

biased: -126 to 127 } this representation is way more significant than 2^{1-127}