Discussion 1:

Floating point representation:

$$(a_n \cdots a_2 a_1 a_0 . b_1 b_2 \cdots)_\beta = \sum_{k=0}^{n} a_k \beta^k + \sum_{k=0}^{\infty} b_k \beta^{-k}$$

decimal : $\beta = 10$

binary : $\beta = 2$

octal : $\beta = 8$

hexadecimal : $\beta = 16$

$0 \le a_i < \beta$

In binary, define a floating point number X as

$$X = \pm q \cdot 2^m$$

q : mantissa, $q = (1.f)_2$ for normal numbers

$q = (0.f)_2$ for subnormal numbers

m : exponent

e.g. in decimal

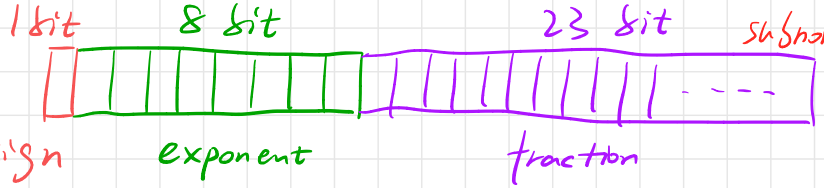
normal : 6.5×10^2

2.314×10^{-5}

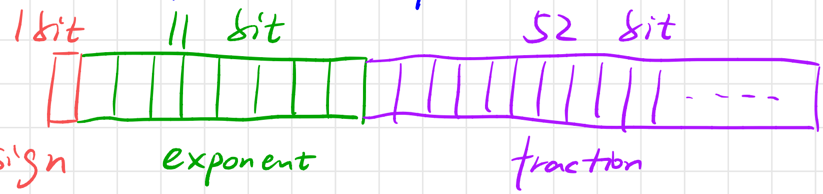
subnormal : 0.065

0.54×10^3

IEEE-754 single precision:



IEEE-754 double precision



Special cases:

Zero: when all exponent and fraction bits are 0. no restriction on the sign bit, so we have +0. -0

Infinity: all exponent bits are 1, all fraction bits are 0, $+\infty$ and $-\infty$ are distinguished from sign bit

NaN: "not a number" ($\text{Inf}-\text{Inf}$, $\frac{\text{Inf}}{\text{Inf}}$, $\frac{0}{0}$, ...). all exponent bits are 1, non-zero fraction bits

Subnormal numbers: all exponent bits are 0, non-zero fraction bits.

How do we define the exponent?

for float precision, what's the range can be represented by 8 bit?

smallest: $(00000000)_2 = 0$

largest: $(11111111)_2 = 2^0 + 2^1 + 2^2 + \cdots + 2^7 = 2^8 - 1 = 255$

the exponent is shifted by 127 to avoid storing sign for exponent, as we saw above, $(00000000)_2$ and $(11111111)_2$ are reserved for special cases. So the actual range is $(1, 254)$, after shifting by 127, the exponent range is $(-126, 127)$

the largest normal number $\sim 2^{127} \sim 10^{38}$

-- smallest normal -- $\sim 2^{-126} \sim 10^{-38}$

for double precision, similar story:

Smallest: $(\underbrace{00--0}_{11\times 0})_2 = 0$

largest: $(\underbrace{11---1}_{11\times 1})_2 = 2^{11}-1 = 2047$

the actual useful range: $(1, 2046)$, shift by 1023 gives exponent range: $(-1022, 1023)$

largest normal number $\sim 2^{1023} \sim 10^{307}$

Smallest normal number $\sim 2^{-1022} \sim 10^{-308}$

Machine epsilon: the distance between 1 and the next largest floating point number.

Single precision:

23th bit

$$1 = (+1)\times 2^0 \times (1.0000--0)_2$$

$$1+\varepsilon_m = (+1)\times 2^0 \times (1.0000---1)_2$$

$$\text{so } \varepsilon_m = 2^{-23} \sim 10^{-7}$$

double precision:

52th bit

$$1 = (+1) \times 2^0 \times (1.0000\cdots0)_2$$

$$1 + \varepsilon_m = (+1) \times 2^0 \times (1.0000\cdots1)_2$$

so $\varepsilon_m = 2^{-52} \sim 10^{-16}$