

Computer representation of numbers (Orator, Sec 3.4)

Binary & decimal representation:

$$(71)_{10} = 7 \times 10^1 + 1 \times 1$$

$$(1000111)_2 = 1 \times 1 + 1 \times 2 + 1 \times 4 + 0 \times 8 + 0 \times 16 + 0 \times 32 + 1 \times 64$$

ith position corresponds to 2^{i-1}

$$(5.5)_{10} = 5 \times 1 + 5 \times \frac{1}{10}$$

$$(101.1)_2 = 1 \times 4 + 0 \times 2 + 1 \times 1 + 1 \times \frac{1}{2}$$

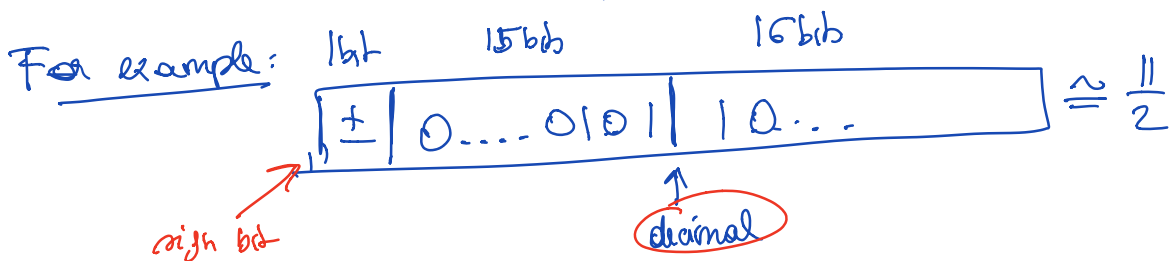
Some numbers have a finite representation with respect to one basis, but not another one:

$$\frac{1}{10} = (0.1)_{10} = (0.00011001100\dots)_2$$

finite infinite

Fixed point representation (base 2, binary)

Storage 32 bit ^{= 4 byt} single precision
 64 bit ^{= 8 byt} double precision "default"



disadvantage: some numbers cannot be represented at all if they are too small or large.

Floating point representation

base 10:

$$x = \pm S \times 10^E$$

↑
mantissa

← integer, "exponent"

$$1 \leq S < 10$$

$$2.3456 \times 10^9$$

base 2:

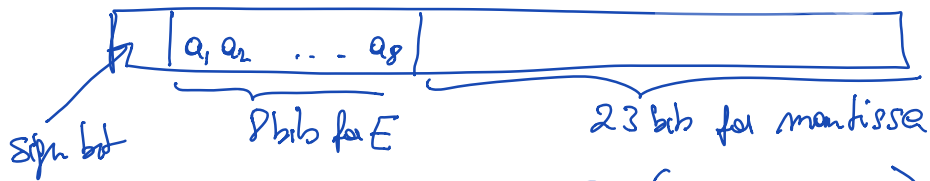
$$x = \pm S \times 2^E$$

$$1 \leq S < 2 (\rightarrow S = 1. \dots)$$

Table 4.1 IEEE Single Precision

Table 4.2 IEEE Double Precision

IEEE single precision bit pattern	IEEE single precision value represented by bit pattern
00000000 00000000 00000000 00000000	$2^{23} \times 2^{-126} = 2^{-103}$
00000001 00000000 00000000 00000000	$2^{22} \times 2^{-126} = 2^{-104}$
00000010 00000000 00000000 00000000	$2^{21} \times 2^{-126} = 2^{-105}$
00000011 00000000 00000000 00000000	$2^{20} \times 2^{-126} = 2^{-106}$
...	...
00111111 00000000 00000000 00000000	$2^1 \times 2^{-126} = 2^{-125}$
01000000 00000000 00000000 00000000	$2^0 \times 2^{-126} = 2^{-126}$
01000001 00000000 00000000 00000000	$2^{-1} \times 2^{-126} = 2^{-127}$
...	...
11111111 00000000 00000000 00000000	$2^{23} \times 2^{-126} = 2^{-103}$
11111110 00000000 00000000 00000000	$2^{22} \times 2^{-126} = 2^{-104}$
...	...
11111101 00000000 00000000 00000000	$2^{21} \times 2^{-126} = 2^{-105}$
11111100 00000000 00000000 00000000	$2^{20} \times 2^{-126} = 2^{-106}$
...	...
11111111 11111111 11111111 11111111	Not a Number (NaN)



$$S = (1. b_1 b_2 \dots)$$

IEEE Standardized representation: don't store 1, "hidden bit"
in 70-80s

round-off, dealing with Inf, -Inf, NaN

Single format



biased exponent stored is $E+127$

$$\left(\sum_{i=1}^{23} b_i 2^{-i} + 1 \right) \times 2^{\left[\sum_{i=1}^8 a_i 2^{8-i} - 127 \right]}$$

smallest number:



$$\approx 1 \times 2^{-126} \approx 1.2 \times 10^{-38} \quad \text{real min (single)}$$

large number:



$$\approx 10^{38} \quad \text{real max (single)}$$

special:

$$a_1 \dots a_8 = 1$$



→ ±∞



→ NaN



→ 0

Table 4.1: IEEE Single Format

±	$a_1 a_2 a_3 \dots a_8$	$b_1 b_2 b_3 \dots b_{23}$
If exponent bitstring $a_1 \dots a_8$ is		
Then numerical value represented is		
(0000000) ₂ = (0) ₁₀	$\pm(0.b_1 b_2 b_3 \dots b_{23})_2 \times 2^{-126}$	
(0000001) ₂ = (1) ₁₀	$\pm(1.b_1 b_2 b_3 \dots b_{23})_2 \times 2^{-126}$	
(0000010) ₂ = (2) ₁₀	$\pm(1.b_1 b_2 b_3 \dots b_{23})_2 \times 2^{-125}$	
(0000011) ₂ = (3) ₁₀	$\pm(1.b_1 b_2 b_3 \dots b_{23})_2 \times 2^{-124}$	
↓	↓	
(0111111) ₂ = (127) ₁₀	$\pm(1.b_1 b_2 b_3 \dots b_{23})_2 \times 2^0$	
(1000000) ₂ = (128) ₁₀	$\pm(1.b_1 b_2 b_3 \dots b_{23})_2 \times 2^1$	
↓	↓	
(1111110) ₂ = (252) ₁₀	$\pm(1.b_1 b_2 b_3 \dots b_{23})_2 \times 2^{125}$	
(1111110) ₂ = (253) ₁₀	$\pm(1.b_1 b_2 b_3 \dots b_{23})_2 \times 2^{126}$	
(1111111) ₂ = (254) ₁₀	$\pm(1.b_1 b_2 b_3 \dots b_{23})_2 \times 2^{127}$	
(1111111) ₂ = (255) ₁₀	±∞ if $b_1 = \dots = b_{23} = 0$, NaN otherwise	

Subnormals:

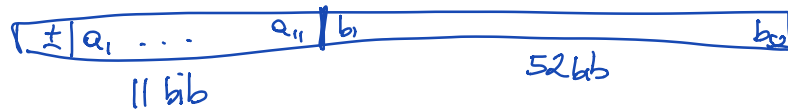


$$\approx (0.1)_2 \times 2^{-126} \approx 2^{-127}$$



$$= 2^{-149}$$

Double precision: 64 bits



Machine epsilon ($\text{eps}(1)$): gap between 1 and next largest number



$$\begin{aligned} \rightarrow \text{eps} &= \frac{1.0 \dots 0 \times 2^0}{2^{-52}} \approx 10^{-16} \quad (\text{double}) \\ &\approx 10^{-7} \quad (\text{single}) \end{aligned}$$