Binary representation

- Basic element : a bit 1/0 on/off , true/false , ...
- This gives us two digits (0,1) we can represent physically, so better use a number sys. that only uses two different digits
  $\Longrightarrow$ binary (base 2) system (fewer different digits, need more places)

- Integers

Example: 137 in base 10 and base 2

$(137)_{10} = (10001001)_2$

- $(137)_{10} = \dfrac{10^2 \quad 10^1 \quad 10^0}{1 \quad\; 3 \quad\; 7} = (1\times 10^2) + (3\times 10^1) + (7\times 10^0)$
  $= 100 + 30 + 7$

- $(10001001)_2 = \dfrac{2^7\,2^6\,2^5\,2^4\,2^3\,2^2\,2^1\,2^0}{1\;0\;0\;0\;1\;0\;0\;1} = (1\times 2^7) + (0\times 2^6) + \ldots + (1\times 2^3) + \ldots + (1\times 2^0)$
  $= 128 + 0 + 0 + 0 + 8 + 0 + 0 + 1$

Easy way to find rep:

Integer division and record remainders

	Remainder	Position	
$137 \backslash 2 = 68$	1	2^0	(1×2^0)
$68 \backslash 2 = 34$	0	2^1	
$34 \backslash 2 = 17$	0	2^2	
$17 \backslash 2 = 8$	1	2^3	(1×2^3)
$8 \backslash 2 = 4$	0	2^4	
$4 \backslash 2 = 2$	0	2^5	
$2 \backslash 2 = 1$	0	2^6	
$1 \backslash 2 = 0$	1	2^7	(1×2^7)

The more bits (0's and 1's) we use, the larger the integer we can store!

+ One bit for the sign! $(-1)^0$ or $(-1)^1$

Floating point numbers

- How to represent real numbers ($\mathbb{R}$) in binary?
- Effectively need three pieces of info
  - The sign
  - The digits appearing in the number
  - The location of the point

  Ex: $\ominus \boxed{235.713664}$

- Strategy: use scientific notation in base 2

- In decimal (base 10):

$$-9.90625 \times 10^{0} \qquad -0.990625 \times 10^{1}$$

[integer exp.]

General: $\pm \left[\text{number in } \left(\tfrac{1}{10}, 1\right)\right] \times 10$

- In binary (base 2):

[integer exp]

$$\boxed{\pm \left[\text{number in } \left(\tfrac{1}{2}, 1\right)\right] \times 2}$$

[exponent]

- Names: $[\text{sign}] \times [\text{mantissa}] \times 2$

- Binary repr. of mantissa

$2^0 \; 2^{-1} \; 2^{-2} \; 2^{-3} \; 2^{-4}$

$$(0.1001)_2 = (0 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (0 \times 2^{-3}) + (1 \times 2^{-4})$$

$$= (0 + 0.5 + 0 + 0 + 0.0625)_{10}$$

$$= (0.5625)_{10}$$

- "Single precision" : 32 bits in total (4 bytes)
  - Sign : 1 bit
  - Exponent : 8 bits
  - Mantissa : 23 bits

Ex : $-3.25 = (-1)^{①} \times (0.8125) \times 2^{②}$

Different standards exist!

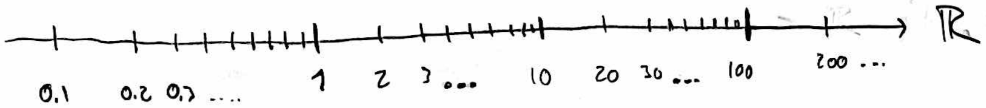
In memory, something like this :

Sign bit	8-bit exponent (2)	23-bit mantissa (0.8125)
1	0 0 0 0 0 0 1 0	0 1 1 0 1 0 0 0 .... 0 0 0 0 0 0

- "Double precision" : 64 bits (8 bytes)
  - Sign : 1 bit
  - Exp : 11 bits
  - Mant. : 52 bits

- Source of unavoidable problems
  - Limited number of bits for exp. $\Rightarrow$ limited range of $\mathbb{R}$ can be repr.
  - Limited number of bits for mantissa $\Rightarrow$ limited "resolution" in representation of the cont. $\mathbb{R}$ (only discrete set can be written exact)

- Intuitive example :
  - Base 10
  - Assume we only had memory for 1 digit in exp. and one digit in mantissa
  - Could repr. numbers $\ldots, 1\times10^{-1}, 2\times10^{-1}, \ldots, 9\times10^{-1}, 1\times10^{0}, 2\times10^{0}, 3\times10^{0}, \ldots,$
  - Range : 10^{-9} to 10^{9}



Only numbers we could use!