How computers represent numbers

Digital computers deal with all data in the form of plenty of **bits**. Each bit is either a 0 or a 1.

> **Comment.** Quantum computers instead work with **qubits** (short for quantum bit), each of which is a linear combination $\alpha\boxed{0} + \beta\boxed{1}$ of basic bits $\boxed{0}$ and $\boxed{1}$, where α and β are complex numbers with $|\alpha|^2 + |\beta|^2 = 1$. As such a single qubit theoretically contains an infinite amount of classical information. Note that a classical bit is the special case where α and β are both 0 or 1.

For efficiency, the **CPU** (central processing unit) of a computer deals with several bits at once. Current CPUs typically work with 64 bits at a time.

> About 20 years ago, CPUs were typically working with 32 bits at a time instead.

Note that 64 bits can store $2^{64} = 18446744073709551616$ many different values. That is a large number but may be limited for certain applications.

> For instance, modern cryptography often works with integers that are 2048 bits large. Clearly, such an integer cannot be stored in a single fundamental 64 bit block.

Representations of integers in different bases

In everyday life, we typically use the **decimal system** to express numbers. For instance:
$$1234 = 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0.$$

10 is called the base, and $1, 2, 3, 4$ are the digits in base 10. To emphasize that we are using base 10, we will write $1234 = (1234)_{10}$. Likewise, we write

$$(1234)_b = 1 \cdot b^3 + 2 \cdot b^2 + 3 \cdot b^1 + 4 \cdot b^0.$$

In this example, $b > 4$, because, if b is the base, then the digits have to be in $\{0, 1, ..., b-1\}$.

> **Comment.** In the above examples, it is somewhat ambiguous to say whether 1 or 4 is the first or last digit. To avoid confusion, one refers to 4 as the **least significant digit** and 1 as the **most significant digit**.

Example 1. $25 = 16 + 8 + 1 = \boxed{1} \cdot 2^4 + \boxed{1} \cdot 2^3 + \boxed{0} \cdot 2^2 + \boxed{0} \cdot 2^1 + \boxed{1} \cdot 2^0$.
Accordingly, $25 = (11001)_2$.

While the approach of the previous example works well for small examples when working by hand (if we are comfortable with powers of 2), the next example illustrates a more algorithmic approach.

Example 2. Express 49 in base 2.

> **Solution.**
>
> - $49 = 24 \cdot 2 + \boxed{1}$. Hence, $49 = (...1)_2$ where ... are the digits for 24.
>
> - $24 = 12 \cdot 2 + \boxed{0}$. Hence, $49 = (...01)_2$ where ... are the digits for 12.
>
> - $12 = 6 \cdot 2 + \boxed{0}$. Hence, $49 = (...001)_2$ where ... are the digits for 6.
>
> - $6 = 3 \cdot 2 + \boxed{0}$. Hence, $49 = (...0001)_2$ where ... are the digits for 3.
>
> - $3 = 1 \cdot 2 + \boxed{1}$. Hence, $49 = (...10001)_2$ where ... are the digits for 1.
>
> - $1 = 0 \cdot 2 + \boxed{1}$. Hence, $49 = (110001)_2$.

Example 3. Express 49 in base 3.

Solution.

- $49 = 16 \cdot 3 + \boxed{1}$

- $16 = 5 \cdot 3 + \boxed{1}$

- $5 = 1 \cdot 3 + \boxed{2}$

- $1 = 0 \cdot 3 + \boxed{1}$

Hence, $49 = (1211)_3$.

Other bases.
What is 49 in base 5? $49 = (144)_5$.
What is 49 in base 7? $49 = (100)_7$.

Fixed-point numbers

Example 4. (warmup)

(a) Which number is represented by $(11001)_2$?

(b) Which number is represented by $(11.001)_2$?

(c) Express 5.25 in base 2.

(d) Express 2.625 in base 2. [Note that $2.625 = 5.25 / 2$.]

Solution.

(a) $(11001)_2 = 2^4 + 2^3 + 2^0 = 16 + 8 + 1 = 25$

(b) $(11.001)_2 = 2^1 + 2^0 + 2^{-3} = 3.125$
Alternatively, $(11.001)_2$ should be $(11001)_2 = 25$ divided by 2^3 (because we move the "decimal" point by three places). Indeed, $(11.001)_2 = 25/2^3 = 3.125$.
Comment. The professional term for "decimal" point would be radix point or, in base 2, binary point (but I have heard neither of these used much in my personal experience).

(c) Note that $5.25 = 2^2 + 2^0 + 2^{-2}$. Hence $5.25 = (101.01)_2$.

(d) Since multiplication (respectively, division) by 2 shifts the digits to the left (respectively, right), we deduce from $5.25 = (101.01)_2$ that $2.625 = (10.101)_2$.