Floating-Point Representation

Floating-point representation is used to store real numbers, especially when dealing with very large or very small values. It approximates real numbers in a way that balances precision and range.

The IEEE 754 Standard

The IEEE 754 standard is the most common way to represent floating-point numbers. It splits a floating-point number into three components:

- **Sign (S)**: Determines if the number is positive or negative (1 bit).
- **Exponent (E)**: Represents the number range (8 bits for single-precision).
- **Mantissa (M)** (also called the significant or fraction): Represents the precision (23 bits for single-precision).

The formula:

$$ \text{value} = (-1)^S \times 1.M \times 2^{(E - \text{Bias})} $$

where **Bias** is 127 for single-precision (32-bit).

Single-Precision (32-bit) Example

Let’s break down the number **10.25** in binary to see how it’s represented:

- **Step 1**: Convert **10.25** to binary:
  - Integer part: **10** in binary is **1010**.
  - Fraction part: **0.25** is **0.01** in binary.
  - So, **10.25** in binary is **1010.01**
- **Step 2**: Normalize it into scientific notation in binary:

$$ 1010.01 = 1.01001 \times 2^3 $$

- **Step 3**: Identify the components:
  - **Sign (S)**: 0 (positive)
  - **Exponent (E)**: We add the bias (127) to the actual exponent (3), so **E = 3 + 127 = 130**. In binary: **10000010**
  - **Mantissa (M)**: We drop the leading 1 from **1.01001**, so the mantissa is 010010… (with trailing zeros to make 23 bits)

- **Step 4**: Combine them

$$ 0 | 10000010 | 01001000000000000000000 $$

Special Values in Floating-Point Representation

In the IEEE 754 floating-point standard, certain special values are reserved for *edge cases* such as

zero, **infinity**, and **undefined operations**. These values help systems represent situations that can't be expressed as regular floating-point numbers.

Special Values Overview

- **Zero**: Represents positive or negative zero
- **Infinity**: Represents positive or negative infinity, resulting from overflow or division by zero.
- **NaN (Not a Number)**: Represents undefined results, such as **0/0** or **sqrt(-1)**

Value	Sign Bit (S)	Exponent (E)	Mantissa (M)	Description
+Zero	0	00000000	00000000000000000000000	Represents positive zero
-Zero	1	00000000	00000000000000000000000	Represents negative zero
+Infinity	0	11111111	00000000000000000000000	Represents positive infinity
-Infinity	1	11111111	00000000000000000000000	Represents negative infinity
NaN	0 or 1	11111111	Non-zero value	Represents "Not a Number", used for undefined operations

Try them out here: https://evanw.github.io/float-toy/