FLOATING POINT ARITHMETIC

Floating-Point Decimal Number

$$
\begin{aligned}
-123456. \times 10^{-1} &= 12345.6 \times 10^{0} \\
&= 1234.56 \times 10^{1} \\
&= 123.456 \times 10^{2} \\
&= 12.3456 \times 10^{3} \\
&= 1.23456 \times 10^{4} (normalised) \\
&\approx 0.12345 \times 10^{5} \\
&\approx 0.01234 \times 10^{6}
\end{aligned}
$$

- There are different representations for the same number and there is no fixed position for the decimal point.
- Given a fixed number of digits, there may be a loss of precession.
- Three pieces of information represents a number: sign of the number, the significant value and the signed exponent of 10.

Note

Given a fixed number of digits, the floating-point representation covers a wider range of values compared to a fixed-point representation.

Example

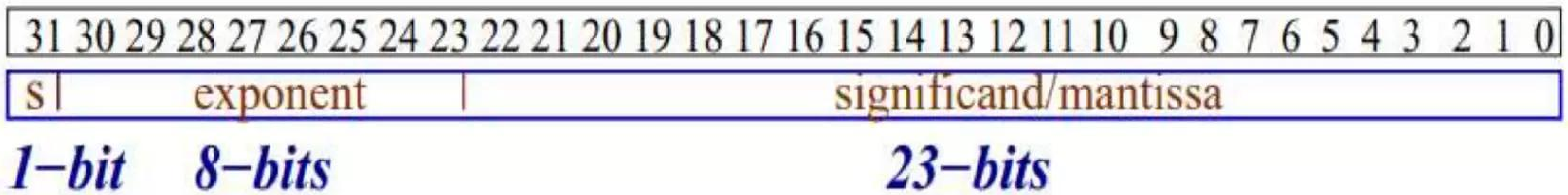
The range of a fixed-point decimal system with six digits, of which two are after the decimal point, is 0.00 to 9999.99.

The range of a floating-point representation of the form $m.mmm \times 10^{ee}$ is $0.0, 0.001 \times 10^{0}$ to 9.999×10^{99}. Note that the radix-10 is implicit.

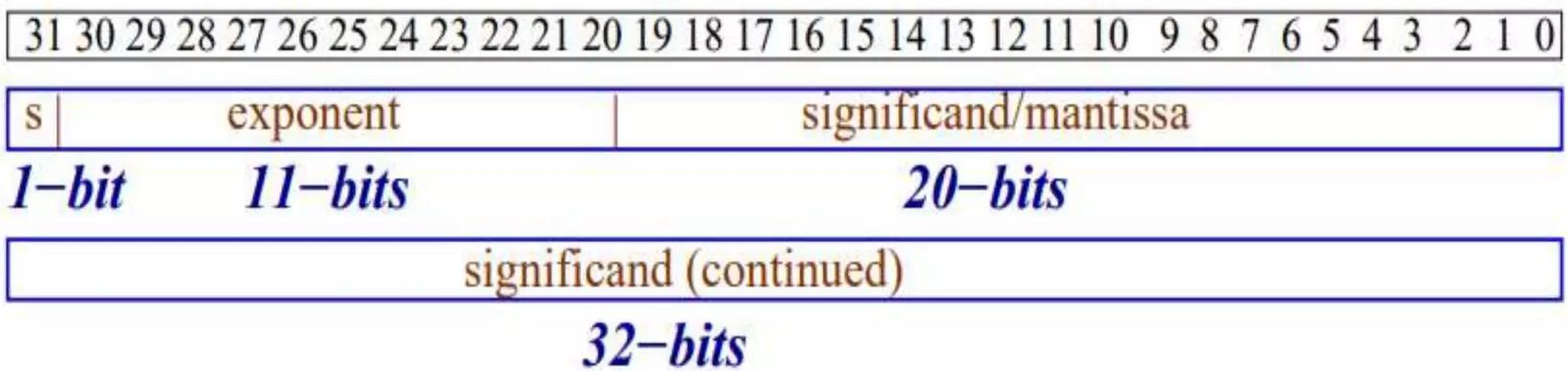
IEEE 754 Standard

Most of the binary floating-point representations follow the IEEE-754 standard. The data type `float` uses IEEE 32-bit single precision format and the data type `double` uses IEEE 64-bit double precision format.

A floating-point constant is treated as a double



Single Precession (32−bit)



Double Precession (64−bit)

Interpretation of Bits

- The most significant bit indicates the sign of the number - one is negative and zero is positive.
- The next eight bits (11 in case of double precession) store the value of the signed exponent of two ($2^{\text{biasedExp}}$).
- Remaining 23 bits (52 in case of double precession) are for the significand (mantissa).

An Example

Consider the decimal number: $+105.625$. The equivalent binary representation is

$$
\begin{aligned}
& +1101001.101 \\
= \; & +1.101001101 \times 2^{6} \\
= \; & +1.101001101 \times 2^{133-127} \\
= \; & +1.101001101 \times 2^{10000101-01111111}
\end{aligned}
$$

In IEEE 754 format:

$\boxed{0\ 1000\ 0101\ 101\ 0011\ 0100\ 0000\ 0000\ 0000}$

An Example

Consider the decimal number: $+2.7$. The equivalent binary representation is

$$\begin{aligned} & +10.10\ 1100\ 1100\ 1100\cdots \\ = \ & +1.010\ 1100\ 1100\cdots \times 2^{1} \\ = \ & +1.010\ 1100\ 1100\cdots \times 2^{128-127} \\ = \ & +1.010\ 1100\cdots \times 2^{10000000-01111111} \end{aligned}$$

In IEEE 754 format (approximate):

$\boxed{0\ 1000\ 0000\ 010\ 1100\ 1100\ 1100\ 1100\ 1101}$

BINARY REPRESENTATION OF FLOATING POINT NUMBERS

Converting decimal fractions into binary representation.

Consider a decimal fraction of the form: 0.d1d2...dn

We want to convert this to a binary fraction of the form:

0.b1b2...bn (using binary digits instead of decimal digits)

Algorithm for conversion

Let X be a decimal fraction: $\mathbf{0.d_1d_2..d_n}$

```
i = 1
Repeat until X = 0 or i = required no.
of binary fractional digits {
    Y = X * 2
    X = fractional part of Y
    B_i = integer part of Y
    i = i + 1
}
```

EXAMPLE 1

Convert 0.75 to binary

X = 0.75   (initial value)

X* 2   = 1.50. Set   b1 = 1, X = 0.5

X* 2   = 1.0.   Set   b2 = 1, X = 0.0

The binary representation for 0.75 is thus

0.b1b2 = 0.11b

Let's consider what that means...

In the binary representation

$$0.b_1b_2...b_m$$

b_1 represents 2^{-1} (i.e., 1/2)

b_2 represents 2^{-2} (i.e., 1/4)

...

b_m represents 2^{-m} ($1/(2^m)$)

So, 0.11 binary represents

$2^{-1} + 2^{-2} = 1/2 + 1/4 = 3/4 = 0.75$

EXAMPLE 2

Convert the decimal value 4.9 into binary

Part 1: convert the integer part into binary: 4 = 100b

Part 2.

Convert the fractional part into binary using multiplication by 2:

$X = .9*2 = 1.8.$ Set $b_1 = 1$, $X = 0.8$

$X*2 = 1.6.$ Set $b_2 = 1$, $X = 0.6$

$X*2 = 1.2.$ Set $b_3 = 1$, $X = 0.2$

$X*2 = 0.4.$ Set $b_4 = 0$, $X = 0.4$

$X*2 = 0.8.$ Set $b_5 = 0$, $X = 0.8$,

which repeats from the second line above.

Since X is now repeating the value 0.8, we know the representation will repeat.

The binary representation of 4.9 is thus:

100.1110011001100...

COMPUTER REPRESENTATION OF FLOATING POINT NUMBERS

In the CPU, a 32-bit floating point number is represented using IEEE standard format as follows:

S | EXPONENT | MANTISSA

where S is one bit, the EXPONENT is 8 bits, and the MANTISSA is 23 bits.

- The ***mantissa*** represents the leading significant bits in the number.

- The ***exponent*** is used to adjust the position of the binary point (as opposed to a "decimal" point)

The mantissa is said to be **normalized** when it is expressed as a value between 1 and 2. I.e., the mantissa would be in the form 1.xxxx.

The leading integer of the binary representation is not stored. Since it is always a 1, it can be easily restored.

The "S" bit is used as a sign bit and indicates whether the value represented is positive or negative

(0 for positive, 1 for negative).

If a number is smaller than 1, normalizing the mantissa will produce a negative exponent.

But 127 is added to all exponents in the floating point representation, allowing all exponents to be represented by a positive number.

Example 1. Represent the decimal value 2.5 in 32-bit floating point format.

$$2.5 = 10.1b$$

In normalized form, this is: $1.01 * 2^1$

The mantissa: M = 01000000000000000000000
(23 bits without the leading 1)

The exponent: E = 1 + 127 = 128 = 10000000b
The sign: S = 0 (the value stored is positive)

So, 2.5 = 01000000001000000000000000000000

Example 2: Represent the number -0.00010011b in floating point form.

0.00010011b = 1.0011 * 2^{-4}

Mantissa: M = 00110000000000000000000 (23 bits with the integral 1 not represented)

Exponent: E = -4 + 127 = 01111011b

S = 1 (as the number is negative)

Result: 1 01111011 00110000000000000000000

Exercise 1: represent -0.75 in floating point format.

Exercise 2: represent 4.9 in floating point format.