Floating-Point Representation and Errors

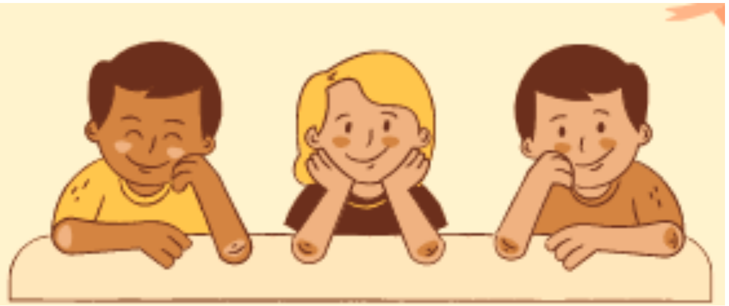
do not use base-10 arithmetic

machine numbers

finite expansion

not a continuum

only the value 0 or 1



use base-10 arithmetic

real numbers

normalized scientific notation

the leading digit in the fraction is not zero

$$37.21829 = 0.37218\,29 \times 10^2$$
$$0.00227\,1828 = 0.22718\,28 \times 10^{-2}$$
$$30\,00527.11059 = 0.30005\,27110\,59 \times 10^7$$

Normalized Floating-Point Representation

In the decimal system, any real number x can be represented in normalized floating-point form as

$$x = \pm 0.d_1 d_2 d_3 \ldots \times 10^n$$

where $\boxed{d_1 \neq 0}$ and n is an integer (positive, negative, or zero).

the real number x, if different from zero, can be represented in normalized floating-point decimal form as

$$x = \pm r \times 10^n \qquad \left(\tfrac{1}{10} \leqq r < 1\right)$$

This representation consists of three parts:

a sign that is either $+$ or $-$,

a number r in the interval $\left[\tfrac{1}{10}, 1\right)$,

and an integer power of 10.

exponent

$$x = \pm r \times 10^n \qquad \left(\tfrac{1}{10} \leqq r < 1\right)$$

normalized mantissa

$-100.253 = -0.100253 \times 10^{3}$

$20.325 = 0.20325 \times 10^{2}$ exponent

$0.000312 = 0.312 \times 10^{-3}$

$-0.1234 = -0.1234 \times 10^{0}$

mantissa

Sign: $+ \equiv$ "0", $- \equiv$ "1"

$(-1)^0 = 1$

$(-1)^1 = -1$

sign +,- 1 bit

exponent 8 bits

The floating-point representation in the binary system

If $x \neq 0$, it can be written as

$$x = \pm q \times 2^m \qquad \left(\tfrac{1}{2} \leqq q < 1\right)$$

$$q = (0.b_1 b_2 b_3 \ldots)_2, \text{ where } b_1 \neq 0$$

Hence, $b_1 = 1$ and then necessarily $q \geqq \frac{1}{2}$

A floating-point number system within a computer is similar to what we have just described, with one important difference: Every computer has only a finite word length

a finite total capacity,

so only numbers with a finite number of digits can be represented.

Numbers that have a terminating expansion in one base may have a nonterminating expansion in another.

$$
\begin{aligned}
\frac{1}{10} = (0.1)_{10} &= (0.06314\,6314\,6314\,6314\ \ldots)_8 \\
&= (0.0\,0011\,0011\,0011\,0011\,0011\,0011\,0011\,0011\ \ldots)_2
\end{aligned}
$$

The important point here is

that most real numbers cannot be represented exactly in a computer.

EXAMPLE

List all the floating-point numbers that can be expressed in the form

$$x = \pm(0.b_1 b_2 b_3)_2 \times 2^{\pm k} \qquad (k, b_i \in \{0, 1\})$$

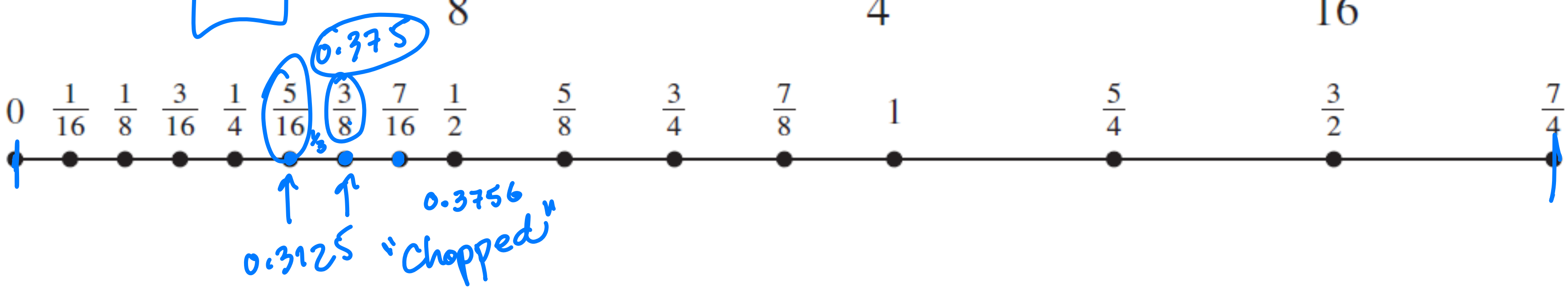
Solution There are two choices for the $\pm$,

two choices for b_1, two choices for b_2, two choices for b_3,

and three choices for the exponent.

For example, the nonnegative numbers in this system are as follows:

$$
\begin{array}{lll}
0.\boxed{000} \times 2^{0} = 0 & 0.000 \times 2^{1} = 0 & 0.000 \times 2^{-1} = 0 \\[6pt]
0.\boxed{001} \times 2^{0} = \dfrac{1}{8} & 0.001 \times 2^{1} = \dfrac{1}{4} & 0.001 \times 2^{-1} = \dfrac{1}{16} \\[6pt]
0.\boxed{010} \times 2^{0} = \dfrac{2}{8} & 0.010 \times 2^{1} = \dfrac{2}{4} & 0.010 \times 2^{-1} = \dfrac{2}{16} \\[6pt]
0.\boxed{011} \times 2^{0} = \dfrac{3}{8} & 0.011 \times 2^{1} = \dfrac{3}{4} & 0.011 \times 2^{-1} = \dfrac{3}{16} \\[6pt]
0.\boxed{100} \times 2^{0} = \dfrac{4}{8} & 0.100 \times 2^{1} = \dfrac{4}{4} & 0.100 \times 2^{-1} = \dfrac{4}{16} \\[6pt]
0.\boxed{101} \times 2^{0} = \dfrac{5}{8} & 0.101 \times 2^{1} = \dfrac{5}{4} & 0.101 \times 2^{-1} = \dfrac{5}{16} \\[6pt]
0.\boxed{110} \times 2^{0} = \dfrac{6}{8} & 0.110 \times 2^{1} = \dfrac{6}{4} & 0.110 \times 2^{-1} = \dfrac{6}{16} \\[6pt]
0.\boxed{111} \times 2^{0} = \dfrac{7}{8} & 0.111 \times 2^{1} = \dfrac{7}{4} & 0.111 \times 2^{-1} = \dfrac{7}{16}
\end{array}
$$



Floating-Point Representation

Many binary computers have a word length of 32 bits

(binary digits)

The internal representation of numbers and their storage is

standard floating-point form,

By **single-precision floating-point numbers,**

Recall that most real numbers

cannot be represented exactly as floating-point numbers,

since they have infinite decimal or binary expansions the 32-bit word-length, the normalized floating-point number

exponent

$\pm q \times 2^m$ must be contained in those 32 bits.

mantissa

One way of allocating the 32 bits is as follows:

sign of q	1 bit
integer $|m|$	8 bits
number q	23 bits

Information on the sign of m is contained in the eight bits

we can represent real numbers with $|m|$ as large as $2^7 - 1 = 127$.

exponent represents numbers from -127 through 128.

Single-Precision Floating-Point Form

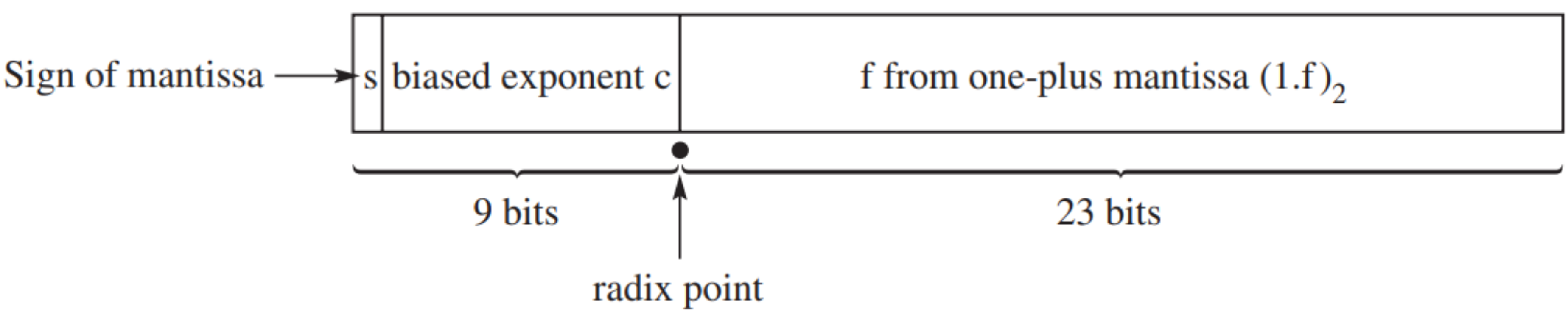
We now describe a machine number of the following form

in **standard single-precision floating-point** representation:

$$(-1)^s \times 2^{c-127} \times (1.f)_2$$

The leftmost bit is used for the sign of the mantissa,

where $s = 0$ corresponds to $+$ and $s = 1$ corresponds to $-$.



$0.1110110 \quad > \quad 0.1011011001$

The value of c in the representation of a floating-point number in single precision is restricted by the inequality

$c-127$

7 6 5 4 3 2 1 0

$$0 < c < (11\,111\,111)_2 = 255$$

$2^7+2^6+2^5+2^4+2^3+2^2+2^1+2^0$

The values 0 and 255 are reserved for special cases, including ± 0 and $\pm\infty$, respectively.

$\boxed{11111110}$

254

Hence, the actual exponent of the number is restricted by

$$-126 \leqq c - 127 \leqq 127$$

254

− 127 +

Likewise, we find that the mantissa of each nonzero number is

The largest number representable is therefore

$$(2 - 2^{-23})2^{127} \approx 2^{128} \approx 3.4 \times 10^{38}.$$

The smallest positive number is $2^{-126} \approx 1.2 \times 10^{-38}$.

0.25675×10^{2} , 0.25675×10^{18}

The binary machine floating-point number $\varepsilon = 2^{-23}$ is called the

the **machine epsilon**

It is the smallest positive machine number ε such that $1 + \varepsilon \neq 1$.

Because $2^{-23} \approx 1.2 \times 10^{-7}$, we infer that in a simple computation,

approximately six significant decimal digits of accuracy

Double-Precision Floating-Point Form

1023

11111111110

$$(-1)^s \times 2^{c-1023} \times (1.f)_2$$

leftmost bit is used for the sign of the mantissa

2047 −
1
2046

11 bits allowed for the exponent

52 bits represent f from the fractional part of the mantissa

EXAMPLE

Determine the single-precision machine representation of

-52.234375 in both single precision and double precision.

Solution

integer part $(52.)_{10} = (110\,100.)_2$.

fractional part, we have $(.234375)_{10} = (.001\,111)_2$.

$$(52.234375)_{10} = (110\,100.001\,111)_2 = (1.\boxed{101\,000\,011\,110)_2} \times 2^5$$

Next the exponent is $(5)_{10}$, and since $c - 127 = 5$

$$c = (132)_{10} = \boxed{(10\,000\,100)_2}$$

Thus, the single-precision machine

$$[\boxed{1}\,\boxed{10\,000\,100}\,\boxed{101\,000\,011\,110\,000\,000\,000\,00}]_2$$

whole number

$$
\begin{array}{r|l l}
2 & 52 & \\
2 & 26 & 0 \\
2 & 13 & 0 \\
2 & 6 & 1 \\
2 & 3 & 0 \\
 & 1 & 1
\end{array}
$$

$$(52)_{10} = (110100)_2$$

$$
\begin{aligned}
(.234375) \times 2 &= 0.46875 \\
0.46875 \times 2 &= 0.9375 \\
0.9375 \times 2 &= 1.875 \\
1.875 \times 2 &= 1.75 \\
1.75 \times 2 &= 1.5 \\
1.5 \times 2 &= 1
\end{aligned}
$$

$$\boxed{(0.001111)_2}$$

In double precision,

$\boxed{\text{64 bits}}$

we let $c - 1023 = 5$

$$(1028)_{10} = (10\,000\,000\,100)_2$$

$$[1\ 10\,000\,000\,100\ 101\,000\,011\,110\,000 \cdots 00]_2$$