Floating-Point Representation

CS3010

Numerical Methods

Dr. Amar Raheja

Section 1.3

Lecture 3

Number System: Bases

- Most Popular Number Systems:
- Decimal: Base 10 (Digits of number system?)
  - 0-9
- Octal: Base 8 (Digits of number system?)
  - 0-7
- Binary: Base 2 (Digits of number system?)
  - 0-1
- Hexadecimal: Base 16 (Digits of number system?)
  - (0-9, A-F)

Number Representation Examples

- Other Bases to Decimal
- Base 10: $4586_{10} = 6x10^0 + 8x10^1 + 6x10^2 + 4x10^3$
- Base 8: $56327_8 = 7x8^0 + 2x8^1 + 3x8^2 + 6x8^3 + 5x8^4 = 23767$
- Base 2: $(10111)_2 = 1x2^0 + 1x2^1 + 1x2^2 + 0x2^3 + 1x2^4 = 23_{10}$
- Base 16: $(4A59F)_{16} = 15x16^0 + 9x16^1 + 5x16^2 + 10x16^3 + 4x16^4 =$ 304543
- Decimal to Bases
- Decimal to Binary: Divide by 2, put remainder in a stack (right most bit-least significant), divide quotient by 2 and keep doing this till quotient is 0
- Other conversions are similar and division is done by appropriate base

Convert Base 10 Integer to binary representation

Table 1 Converting a base-10 integer to binary representation.

	Quotient	**Remainder**
11/2	5	$1 = a_0$
5/2	2	$1 = a_1$
2/2	1	$0 = a_2$
1/2	0	$1 = a_3$

Hence $$\begin{aligned}(11)_{10} &= (a_3 a_2 a_1 a_0)_2 \\ &= (1011)_2\end{aligned}$$

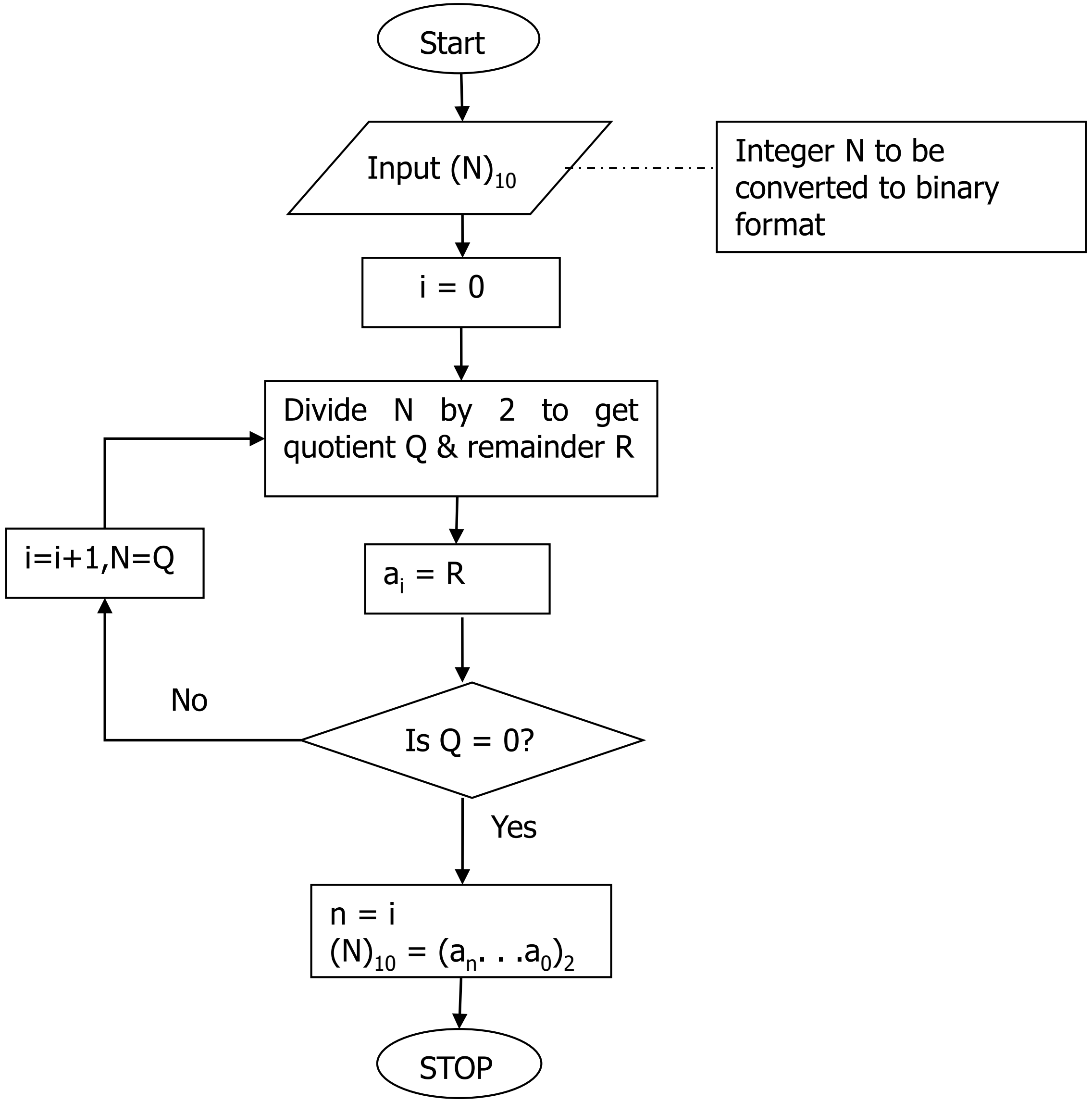
Convert Base 10 Integer to Octal representation

Table 1 Converting a base-10 integer to binary representation.

	Quotient	**Remainder**
145/8	18	1
18/8	2	2
2/8	0	2

Hence, $(145)_{10} = (221)_8$

Flowchart



Source: http://numericalmethods.eng.usf.edu

Fractional Decimal Number to Binary

Table 2. Converting a base-10 fraction to binary representation.

	Number	Number after decimal	Number before decimal
0.1875×2	0.375	0.375	$0 = a_{-1}$
0.375×2	0.75	0.75	$0 = a_{-2}$
0.75×2	1.5	0.5	$1 = a_{-3}$
0.5×2	1.0	0.0	$1 = a_{-4}$

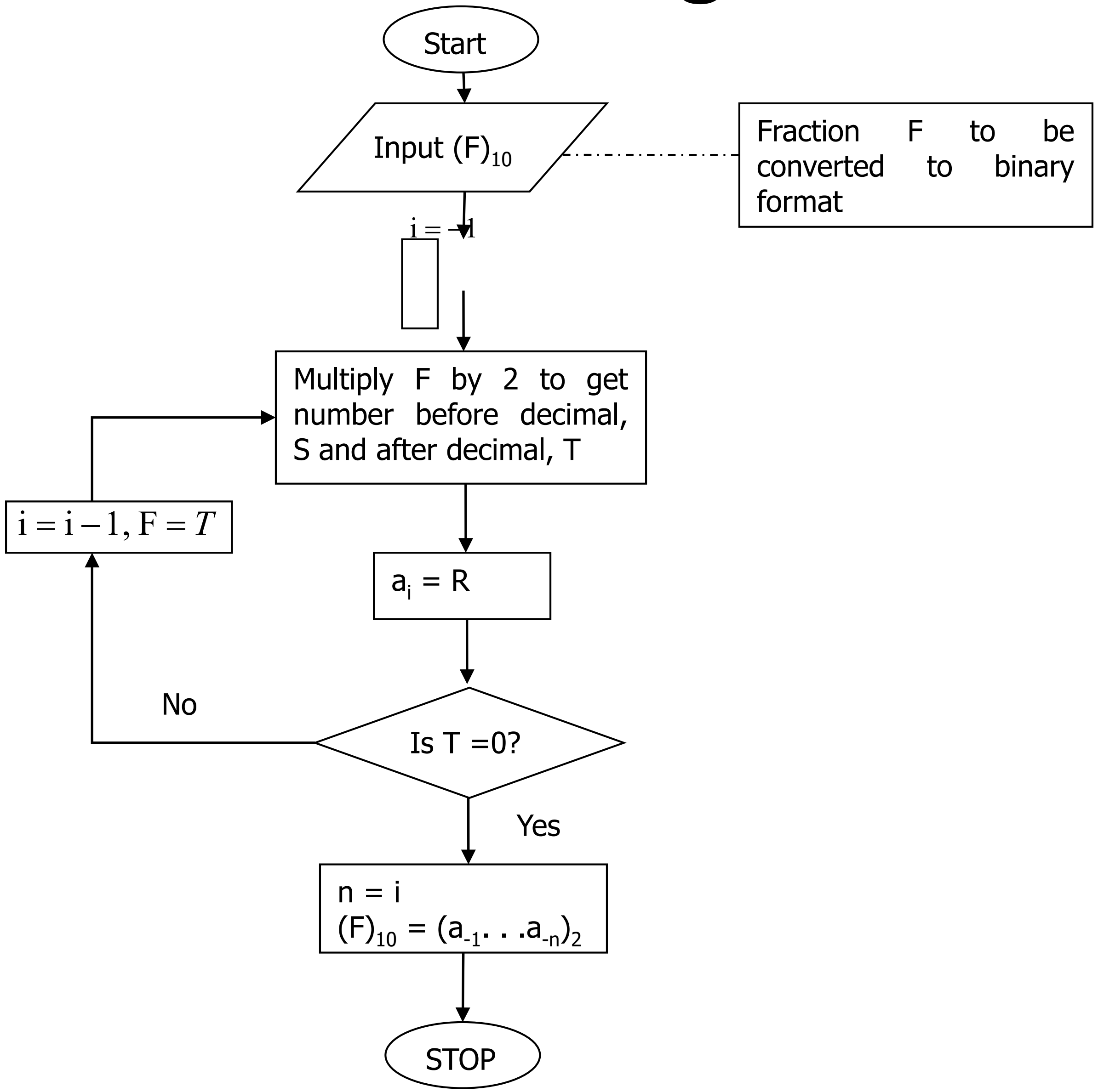
Hence

$$(0.1875)_{10} = (a_{-1}a_{-2}a_{-3}a_{-4})_2$$

$$= (0.0011)_2$$

Source: http://numericalmethods.eng.usf.edu

Flowchart for converting Fractional Part



Source: http://numericalmethods.eng.usf.edu

Decimal Number to Binary

$$(11.1875)_{10} = (?.?)_2$$

Since $(11)_{10} = (1011)_2$

and $(0.1875)_{10} = (0.0011)_2$

then, one can combine these two to get

$$(11.1875)_{10} = (1011.0011)_2$$

All Fractional Decimal Numbers Cannot be Represented Exactly

- 0.3 or 0.1 cannot be represented in a finite way in binary

Table 3. Converting a base-10 fraction to approximate binary representation.

	Number	Number after decimal	Number before Decimal
0.3×2	0.6	0.6	$0=a_{-1}$
0.6×2	1.2	0.2	$1=a_{-2}$
0.2×2	0.4	0.4	$0=a_{-3}$
0.4×2	0.8	0.8	$0=a_{-4}$
0.8×2	1.6	0.6	$1=a_{-5}$

$$(0.3)_{10} \approx (a_{-1}a_{-2}a_{-3}a_{-4}a_{-5})_2 = (0.01001)_2 = 0.28125$$

Source: http://numericalmethods.eng.usf.edu

Another Way to Look at Conversion

- Convert $(11.1875)_{10}$ to base 2

$(11)_{10} = 2^3 + 3$

$= 2^3 + 2^1 + 1$

$= 2^3 + 2^1 + 2^0$

$= 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0$

$= (1011)_2$

$(0.1875)_{10} = 2^{-3} + 0.0625$

$= 2^{-3} + 2^{-4}$

$= 0x2^{-1} + 0x2^{-2} + 1x2^{-1} + 1x2^{-4}$

$= (0.0011)_2$

Scientific Notation

- Scientific Notation: Shifting the decimal point to represent decimal numbers

$$256.78 \text{ is written as } +2.5678 \times 10^{2}$$

$$0.003678 \text{ is written as } +3.678 \times 10^{-3}$$

$$-256.78 \text{ is written as } -2.5678 \times 10^{2}$$

- Normalized Scientific Notation: The number is represented by a fraction multiplied by 10^n,and the leading digit in the fraction is not zero (except when the number involved is zero).
- One can write 79325 as 0.79325 x 10^5, not 0.07932 x 10^6 or 7.9325 x10^4 or any other way.

Normalized Scientific Notation

37.21829 = 0.37218 29 x 10^2

0.00227 1828 = 0.22718 28 x 10^{-2}

-30 00527.11059 = -0.30005 27110 59 x 10^7

- So, change number to become a fractional number which is between 0 and 1 and is called the mantissa and the exponent is the power of 10
- The form of $x = \pm 0.d_1 d_2 d_3 \ldots \times 10^n = \pm r \times 10^n$

where $d_1 \neq 0$ and n is an integer (positive, negative, or zero). The numbers $d_1, d_2, \ldots$ are the decimal digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

x = sign × mantissa × 10^n

- a sign that is either + or −, a number r in the interval $\left[\frac{1}{10}, 1\right)$ and an integer power of 10. The number r is called the normalized mantissa and n the exponent.

Normalized Floating-Point Representation for Binary

- The floating-point representation in the binary system is similar to that in the decimal system in several ways. If $x \neq 0$, it can be written as
$$x = \pm q \times 2^m \quad \left(\frac{1}{2} \leqq q < 1\right)$$
- The mantissa q would be expressed as a sequence of zeros or ones in the form $q = \pm 0.b_1 b_2 b_3 \ldots \times 2^m$ where $b_1 \neq 0$. Hence, where $b_1 = 1$
- Every computer has only a finite word length and a finite total capacity, so only numbers with a finite number of digits can be represented.
- A number is allotted only one word of storage in the single-precision mode (two or more words in double or extended precision).
- Clearly, irrational numbers cannot be represented, nor can those rational numbers that do not fit the finite format imposed by the computer.

Machine Numbers

- The real numbers that are representable in a computer are called its **machine numbers**.
- Since any number used in calculations with a computer system must conform to the format of numbers in that system, it must have a finite expansion. Numbers that have a nonterminating expansion cannot be accommodated precisely.
- Moreover, a number that has a terminating expansion in one base may have a nonterminating expansion in another.
  - Another good example of this is the following simple fraction as given in the introductory example
  - $(0.1)_{10} = (0.06314\ 6314\ 6314\ 6314\ ...)_8$
    $= (0.0\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011\ ...)_2$

5 bit Example

- A floating-point numbers must be of the form $q = \pm(0.b_1 b_2 b_3)_2 \times 2^{\pm k}$, where b_1, b_2, b_3, and m are allowed to have only the value 0 or 1.
- There are two choices for the ±, two choices for b_1, two choices for b_2, two choices for b_3, and three choices for the exponent.
- Thus, at first, one would expect 2 x 2 x 2 x 2 x 3 = 48 different numbers.

Possible positive numbers represented

- However, there is some duplication. For example, the nonnegative numbers in this system are as follows:
- Altogether there are 31 distinct numbers in the system. The positive numbers obtained are shown on a line

$$0.000 \times 2^0 = 0 \qquad 0.000 \times 2^1 = 0 \qquad 0.000 \times 2^{-1} = 0$$

$$0.001 \times 2^0 = \frac{1}{8} \qquad 0.001 \times 2^1 = \frac{1}{4} \qquad 0.001 \times 2^{-1} = \frac{1}{16}$$

$$0.010 \times 2^0 = \frac{2}{8} \qquad 0.010 \times 2^1 = \frac{2}{4} \qquad 0.010 \times 2^{-1} = \frac{2}{16}$$

$$0.011 \times 2^0 = \frac{3}{8} \qquad 0.011 \times 2^1 = \frac{3}{4} \qquad 0.011 \times 2^{-1} = \frac{3}{16}$$

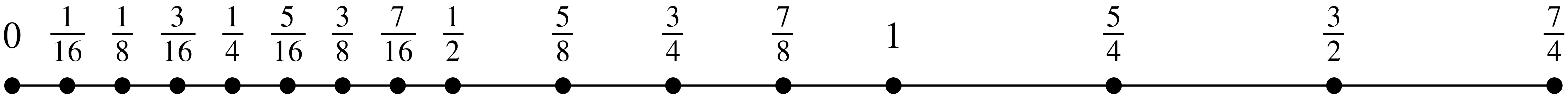
$$0.100 \times 2^0 = \frac{4}{8} \qquad 0.100 \times 2^1 = \frac{4}{4} \qquad 0.100 \times 2^{-1} = \frac{4}{16}$$

$$0.101 \times 2^0 = \frac{5}{8} \qquad 0.101 \times 2^1 = \frac{5}{4} \qquad 0.101 \times 2^{-1} = \frac{5}{16}$$

$$0.110 \times 2^0 = \frac{6}{8} \qquad 0.110 \times 2^1 = \frac{6}{4} \qquad 0.110 \times 2^{-1} = \frac{6}{16}$$

$$0.111 \times 2^0 = \frac{7}{8} \qquad 0.111 \times 2^1 = \frac{7}{4} \qquad 0.111 \times 2^{-1} = \frac{7}{16}$$

Possible positive numbers



- If, in the course of a computation, a number x is produced of the form $= \pm q \times 2^m$, where m is outside the computer's permissible range, then we say that an overflow or an underflow has occurred or that x is outside the range of the computer.
- Generally, an overflow results in a fatal error (or exception), and the normal execution of the program stops.
- An underflow, however, is usually treated automatically by setting x to zero without any interruption of the program but with a warning message in most computers.
- In this specific example, any number closer to zero than 1/16 would underflow to zero, and any number outside the range −1.75 to +1.75 would overflow to machine infinity.

Hole at Zero

- If, in this Example, we allow only normalized floating-point numbers, then all our numbers (with the exception of zero) have the form

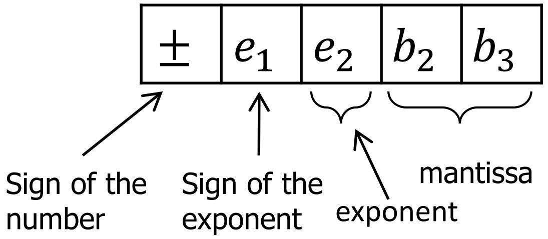
$$x = \pm(0.1b_2b_3)_2 \times\ 2^{\pm k}$$

- This creates a phenomenon known as the **hole at zero**. The nonnegative machine numbers are now distributed as in Figure below.

- There is a relatively wide gap between zero and the smallest positive machine number, which is $(0.100)_2\ \times\ 2^1\ =\ \frac{1}{4}$

5-bits to represent this F-P example

- These normalized floating-point numbers can be stored in a 5-bit computer with 1 bit for sign of the number, two bits for exponent (in which 1 bit is for exponent sign) and two bits for mantissa



- All possible combinations of positive normalized F-P numbers are

$$(0.1b_2b_3)_2 \times 2^m = \left\{ \begin{array}{l} (0.100)_2 = \frac{1}{2} \\ (0.101)_2 = \frac{5}{8} \\ (0.110)_2 = \frac{3}{4} \\ (0.111)_2 = \frac{7}{8} \end{array} \right\} \times 2^{-1,0,1} = \left\{ \begin{array}{l} \frac{1}{4}, \frac{1}{2}, 1 \\ \frac{5}{16}, \frac{5}{8}, \frac{5}{4} \\ \frac{3}{8}, \frac{3}{4}, \frac{3}{2} \\ \frac{7}{16}, \frac{7}{8}, \frac{7}{4} \end{array} \right.$$

- So, a machine number in floating-point single-precision is of the form

$$(-1)^s q \times 2^m = (-1)^s \times 2^{c-1} \times (1.b_2b_3)_2$$

IEEE-754 Floating Point Standard

- Standardizes representation of floating point numbers on different computers in single and double precision.
- Standardizes representation of floating point operations on different computers.

Precision	Bits	Sign	Exponent	Mantissa
Single	32	1	8	23
Double	64	1	11	52
Long Double	80	1	15	64

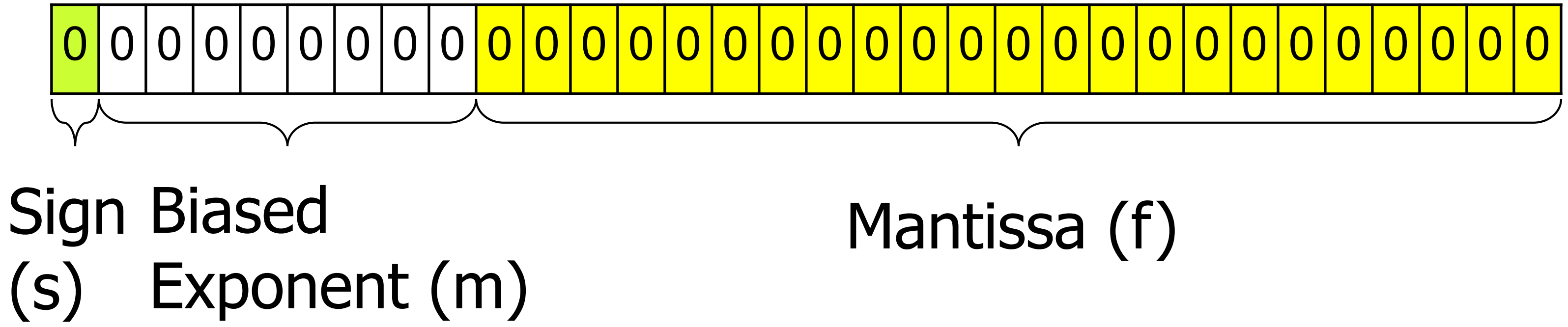
IEEE-754 Format Single Precision

$$x = \pm q \times 2^{m}$$

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

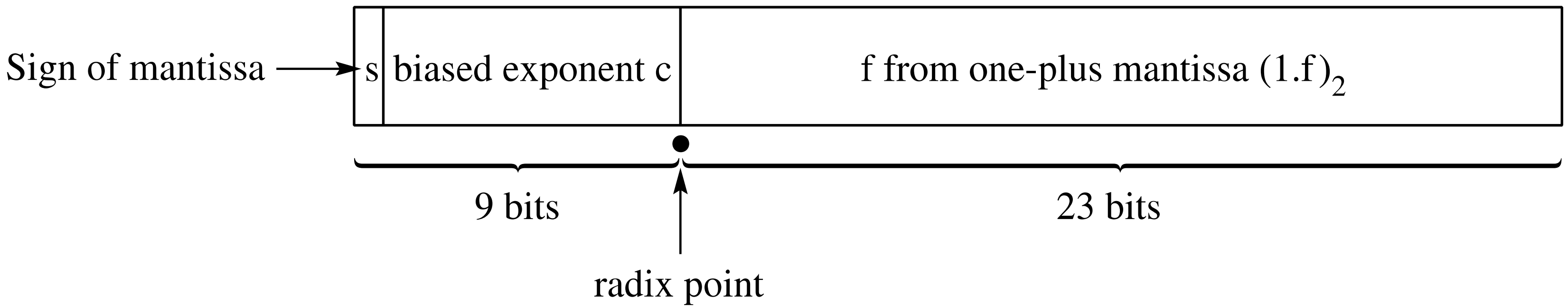
$$(-1)^{s} \times 2^{c-127} \times (1.f\,)_2$$

32 bits for single precision



Single-Precision Floating-Point Form

- In the normalized representation of a nonzero floating-point number, the first bit in the mantissa is always 1 so that this bit does not have to be stored.
- This can be accomplished by shifting the binary point to a "1-plus" form $(1.f)_2$.
- The mantissa is the rightmost 23 bits and contains f with an understood binary point as in this Figure .
- So the mantissa (significand) actually corresponds to 24 binary digits since there is a hidden bit. (An important exception is the number ±0.)



Single-Precision Floating-Point Form

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

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

- The values 0 and 255 are reserved for special cases, including ± 0 and $\pm\infty$, respectively.
- Hence, the actual exponent of the number is restricted by the inequality

$$-126 \leq c - 127 \leq 127$$

- Likewise, we find that the mantissa of each nonzero number is restricted by the inequality
- $1 \leq (1.f\,)_2 \leq (1.111\ 111\ 111\ 111\ 111\ 111\ 111\ 11)_2 = 2 - 2^{-23}$

Single-Precision Floating-Point Form

- 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 $2^{-126} \approx 1.2 \times 10^{-38}$
- The binary machine floating-point number $\varepsilon = 2^{-23}$ is called the machine epsilon when using single precision. 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 may be obtained in single precision. Recall that 23 bits are allocated for the mantissa.

Special Representations

- Special cases:
- 0 represented as sign bit s = 0 or 1, c = 0 and f = 0 (all zeros)
- ±∞ represented as sign bit s = 0 or 1, c = 255 and f = 0 (all zeros)
- NaN (division by 0) represented as c = 255 and f $\neq$ 0

s	c	f	Represents
0	all zeros	all zeros	0
1	all zeros	all zeros	-0
0	all ones	all zeros	+∞
1	all ones	all zeros	-∞
0 or 1	all ones	non-zero	NaN

Example 1

- Determine the single-precision machine representation of the decimal number−52.23437 5
- The whole part is $(52.)_{10}$ =$(64.)_8$ =$(110\ 100.)_2$
- The fractional part, we have $(.23437\ 5)_{10} = (.17)_8 = (.001\ 111)_2$
- $(52.23437\ 5)_{10} = (110\ 100.001\ 111)_2 = (1.101\ 000\ 011\ 110)_2 \times 2^5$
- $(.101\ 000\ 011\ 110)_2$ is the stored mantissa.
- The exponent is $(5)_{10}$, and since $c−127 = 5$
- Hence $c = (132)_{10} = (204)_8 = (10\ 000\ 100)_2$ is the stored exponent.
- So, the single-precision machine representation

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

$$[1100\ 0010\ 0101\ 0000\ 1111\ 0000\ 0000\ 0000]_2 = [C250F000]_{16}$$

Example 2

- What is the single-precision representation of $(24.625)_{10}$

$(24.)_{10} = (11000.)_2$ and $(0.625)_{10} = (0.101)_2$

Hence, $(24.625)_{10} = (11000.101)_2 = (1.1000101)^2 = (1.1000101) \times 2^4$

$c - 127 = 4$, so $c = 131 = (10000011)_2$

So, the single-precision representation is

$[0\ 10\ 000\ 011\ 100\ 010\ 100\ 000\ 000\ 000\ 000\ 00]_2 =$

$[0100\ 0001\ 1100\ 1010\ 0000\ 0000\ 0000\ 0000]_2 = [41CA0000]_{16}$

Single Precision to Decimal

- What decimal number is represented by

01000001011111000000000000000000

Sign bit is 0, so positive number

c = $(10000010)_2$ = 130 , so m = c-127 = 3

mantissa f = $(1.\ 11111000000000000000000)_2$

So, number is

$(1.\ 11111000000000000000000)_2 \times 2^3$

= $(1111.11000000000000000000)_2$

= 15.0 + 0.5 + 0.25 = $(15.75)_{10}$

Double Precision Floating Point Representation

- In double precision, there are 52 bits allocated for the mantissa. The double precision machine epsilon is $2^{-52} \approx 2.2 \times 10^{-16}$, so approximately 15 significant decimal digits of precision are available.
- There are 11 bits allowed for the exponent, which is biased by 1023.
- Finally, 64 bits represent the double precision number.
- The exponent represents numbers from −1022 through 1023.
- A machine number in standard double precision floating-point form corresponds to

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

Double Precision Floating Point Representation

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

$$0 < c < (1\ 111\ 111\ 111)_2 = 2047$$

- As in single precision, the values at the ends of this interval are reserved for special cases.
- Hence, the actual exponent of the number is restricted by the inequality

$$-1022 \leq c-1023 \leq 1023$$

- We find that the mantissa of each nonzero number is restricted by the inequality
- $1 \leq (1.f\,)_2 \leq (1.111\ 111\ 111 \cdots 111\ 111\ 111\ 1)_2 = 2-2^{-52}$

Double Precision Floating Point Representation

- Recall that 52 bits are allocated for the mantissa.
- The largest double-precision machine number is $(2 - 2^{-52})2^{1023} \approx 2^{1024} \approx 1.8 \times 10^{308}$.
- The smallest double-precision positive machine number is $2^{-1022} \approx 2.2 \times 10^{-308}$.
- Consequently, the range for integers is from $-(2^{31} - 1)$ to $(2^{31} - 1) =$ 21474 83647.
- In double precision, 63 bits are used for integers giving integers in the range $-(2^{63} - 1)$ to $(2^{63} - 1)$.

Double-Precision Representation

- −52.234375 in double precision
- for the exponent $(5)_{10}$,we let c−1023=5,and we have $(1028)_{10}$ = $(2004)_8$ = $(10\ 000\ 000\ 100)_2$, which is the stored exponent.
- Thus, the double-precision machine representation of −52.234375 is
- $[11000000010010100001111 0000 \cdots 00]_2$ = $[1100000001001010000111100000\cdots0000]_2$ =$[C04A1E0000000000]_{16}$

- Class Exercise: Find Double-Precision representation of

-285.75

$[C071DC0000000000]_{16}$