CS11001/CS11002
Programming and Data Structures (PDS)
(Theory: 3-1-0)

The IEEE Floating Point Numbers
(IEEE 754 format)

Floating Point Numbers (reals)

- To represent numbers like 0.5, 3.1415926, etc, we need to do something else. First, we need to represent them in binary, as

$$n = \cdots + a_m 2^m \cdots + a_2 2^2 + a_1 2 + a_0 + a_{-1} \times \frac{1}{2} + a_{-2} \times 2^{-2} + a_{-3} \times 2^{-3} + \cdots a_{-k} 2^{-k} + \cdots$$

  E.g. ~~11.00110 for 2+1+1/8+1/16=3.1875~~

- Next, we need to rewrite in scientific notation, as 1.100110×2^1. That is, the number will be written in the form:

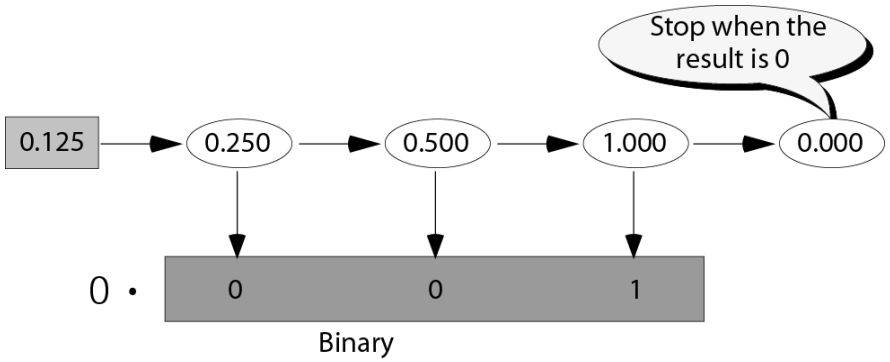
  $1.xxxxxx\ldots \times 2^e$

  x = 0 or 1

Figure 3-7

Changing fractions to binary

- Multiply the fraction by 2,...



Example 17

Transform the fraction 0.875 to binary

Solution

Write the fraction at the left corner. Multiply the number continuously by 2 and extract the integer part as the binary digit. Stop when the number is 0.0.

0.875	➔	1.750	➔	1.5	➔	1.0	➔	0.0
0	.	1		1		1		

Example 18

Transform the fraction 0.4 to a binary of 6 bits.

Solution

Write the fraction at the left cornet. Multiply the number continuously by 2 and extract the integer part as the binary digit. You can never get the exact binary representation. Stop when you have 6 bits.

0.4	➔	0.8	➔	1.6	➔	1.2	➔	0.4	➔	0.8	➔	1.6
0	.	0		1		1		0		0		1

Normalization

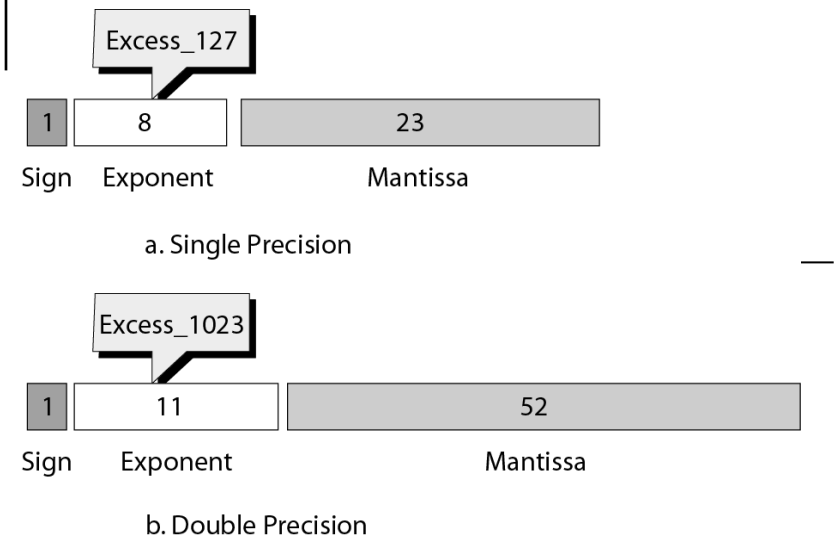
Example of normalization

Original Number	*Move*	*Normalized*
+1010001.1101	← 6	$+2^{6}$ x 1.01000111001
−111.000011	← 2	-2^{2} x 1.11000011
+0.00000111001	6 →	$+2^{-6}$ x 1.11001
−0.001110011	3 →	-2^{-3} x 1.110011

- Sign, exponent, and mantissa

Figure 3-8

IEEE standards for floating-point representation



Example 19

Show the representation of the normalized number $+ 2^6 \times 1.01000111001$

Solution

The sign is positive. The Excess_127 representation of the exponent is 133. You add extra 0s on the right to make it 23 bits. The number in memory is stored as:

0 10000101 01000111001000000000000

Number	*Sign*	*Exponent*	Mantissa
$-2^2 \times 1.11000011$	1	10000001	11000011000000000000000
$+2^{-6} \times 1.11001$	0	01111001	11001000000000000000000
$-2^{-3} \times 1.110011$	1	01111100	11001100000000000000000

Example of floating-point representation

Example 20

Interpret the following 32-bit floating-point number

1 01111100 11001100000000000000000

Solution

The sign is negative. The exponent is –3 (124 – 127). The number after normalization is

$$-2^{-3} \times 1.110011$$

Limitations in 32-bit Integer and Floating Point Numbers

- Limited range of values (e.g. integers only from -2^{31} to $2^{31}-1$)
- Limited resolution for real numbers. E.g., if x is a machine representable value, the next value is $x + \varepsilon$ (for some small ε). There is no value in between. This causes "floating point errors" in calculation. The accuracy of a single precision floating point number is about 6 decimal places.

Limitations of Single Precision Numbers

- Given the representation of the single precision floating point number format, what is the largest magnitude possible? What is the smallest number possible?
- With floating point number, it can happen that $1 + \varepsilon = 1$. What is that largest ε?

Normalized numbers in Single Precision Format

- The normalized numbers are:

$$(-1)^S 1.f\ 2^{E-127}$$

Here S is the sign bit, f is the Mantissa and E is the exponent.

Range of normalized numbers

- $f_{max}^{+} = (1.111\ldots1)2^{254-127}$
  - E=0 is reserved for zero (with f=0) and denormalized numbers (with f≠0).
  - E=255 is reserved for ±∞ (with f=0) and for NaN (Not a Number) (with f≠0).
- Thus, $f_{max}^{+}=(2-2^{-23})2^{127}=(1-2^{-24})2^{128}$.
- Similarly, $f_{min}^{+}=(1.0)2^{1-127}=2^{-126}$.
- The exponent bias and significand range were selected so that the reciprocal of all normalized numbers can be represented without overflow. (in particular f_{min}^{+}).

Denormalized Numbers

	f=0	f≠0
E=0	0	Denormalized
E=255	±∞	NaN

- **The denormalized numbers provide representations for values smaller than the smallest normalized number, lowering the probability of an exponent underflow.**
  - which occurs when you get numbers lesser than f_{min}^{+}.
  - Values of these numbers are $(-1)^{S}\ 0.f\ 2^{-126}$
  - Also note that there are two representations for 0 (plus and minus). You may include them as one denormalized number.

Smallest Denormalized Numbers

- Smallest Denormalized number is:

  $2^{-23}\ 2^{-126}=2^{-149}$.

  - this reduces the gap between the smallest representable number and zero.
  - note that although the true value of the exponent should have been 0-127=-127, the value of -126 was chosen as $f_{min}^{+}=2^{-126}$. This reduces the gap between the largest demormalized number and the smallest normalized number.

Limitations of Single Precision Numbers

- Given the representation of the single precision floating point number format, what is the largest magnitude possible? What is the smallest number possible?
- With floating point number, it can happen that $1 + \varepsilon = 1$. What is that largest ε?

NaN (E=255 and f≠0)

- There are two kinds of Nan
  - the signaling (trapping): sets an Invalid operation exception flag whenever any arithmetic operation with this NaN as an operand is attempted.
  - quiet (non-trapping) A signaling NaN becomes a quiet NaN, when used as an operand for an arithmetic operation with the Invalid operation exception flag disabled.

Invalid operations

1. Multiplying 0 by ∞
2. Dividing 0 by 0 or ∞ by ∞
3. Adding + ∞ and - ∞
4. Finding the square root of negative number
5. Calculating the remainder x modulo y, when y is zero or x is infinite
6. Any operation on a signaling NaN