Introduction to IEEE Standard 754 for Binary Floating-Point Arithmetic

Computer Organization and Assembly Languages,
NTU CSIE, 2004

Speaker: Jiun-Ren Lin

Date: Oct 26, 2004

Floating point numbers

- Integers: the universe is infinite but discrete
  - No fractions
  - No number between two integers
  - A countable (finite) number of items in a finite range
- Real numbers: the universe is infinite and continuous
  - Fractions represented by decimal notation
    - Rational numbers, e.g., 5/2=2.5
    - Irrational numbers, e.g., 22/7=3.14159265…
  - Infinite numbers exist even in the smallest range

Scientific notation

- Decimal numbers
  - 0.513×105, 5.13×104 and 51.3×103 are written in scientific notation.
  - 5.13×104 is in normalized scientific notation.
- Binary numbers
  - Base 2
  - Binary point – multiplication by 2 moves the point to the left.
  - Normalized scientific notation, e.g., $1.0_{two}\times2-1$
  - Known as *floating point numbers*.

Basic floating point notation

- For example, use 8-bit word size
  - 2.5=10.1 in binary
    - One way to represent the point is to put it in the same place all the time and then not represent it explicitly at all. To do that, we must have a standard representation for a value that puts the point in the same place every time.
  - 10.1=1.01*2^1
  - Now we use 1 bit for sign, 2 bits for exponent, and rest for value (significand).
  - Sign=0(positive), exponent=01, significant=101
  - Result: 00110100

Refinement

- In order to make more reasonable range of value, we use 32 or 64 bits instead of 8 bits.
- Note that the leading 1 (as in $1.01*2^1$) is always there, so we don’t need to waste one of our precious bits on it. Just assume it is always there.
- Use excess something notation for handling negative exponents.
- These ideas came from existing schemes developed for the PDP-11 and CDC 6600 computers.

Floating point numbers

- General format

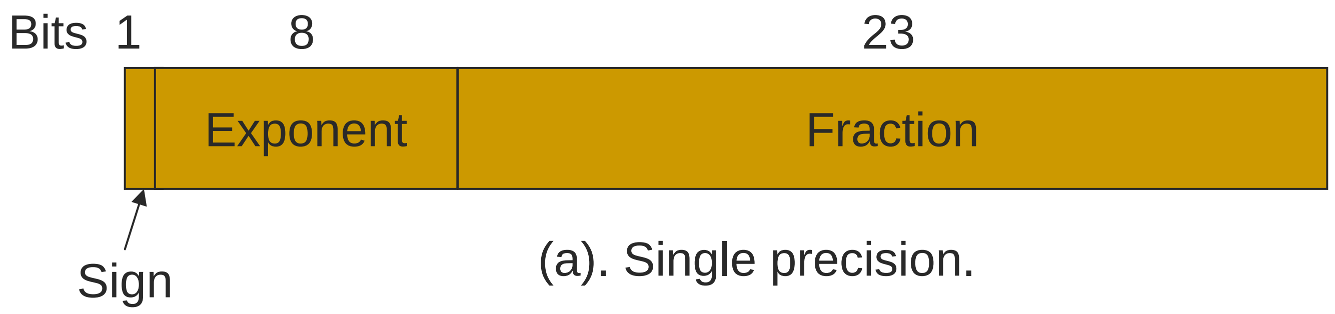
$$\pm 1.bbbbb_{two} \times 2^{eeee}$$

or $$(-1)^{S} \times (1+F) \times 2^{E}$$

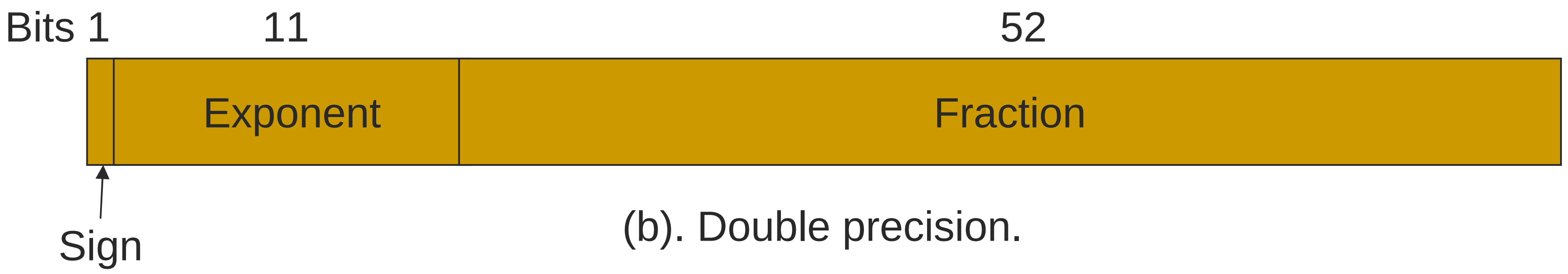
- Where
  - S = sign, 0 for positive, 1 for negative
  - F = fraction (or mantissa) as a binary integer, 1+F is called significand
  - E = exponent as a binary integer, positive or negative (two’s complement)

IEEE Standard 754

- One way to meet the need, agreed to and accepted by most computer manufacturers.



(a). Single precision.



(b). Double precision.

Steps to IEEE format

- Convert 35.75, for example
  - Convert the number to binary
    - 100011.11
  - Normalize
    - 1.0001111 x 25
  - Fit into the required format
    - 5+127=132; hide the leading 1.
    - 01000010000011110000000000000000
  - Use Hexadecimal to make it easier to read
    - 420F0000

Single and double precision

- Double precision uses more space, allows greater magnitude and greater precision.
- Other than that, it behaves just like single precision.
- We will use only single precision in examples, but any could easily be expanded to double precision.

IEEE 754 details

Item	Single precision	Double precision
Bits in sign	1	1
Bits in exponent	8	11
Bits in fraction	23	52
Bits total	32	64
Exponent system	Excess 127	Excess 1023
Exponent Range	-126 to +127	-1022 to 1023
Smallest normalized number	2^{-126}	2^{-1022}
Largest normalized number	approx. 2^{128}	approx. 2^{1024}
Decimal range	approx.10^{-38} to 10^{38}	approx.10^{-308} to 10^{308}
Smallest denormalized number	approx. 10^{-45}	approx. 10^{-324}

What's normal?

- Normalized means represented in the normal, or standard, notation. Some numbers do not fit into that scheme and have a separate definition.
- Consider the smallest normalized value:
  - 1.000—000 x 2^{-126}
  - How would we represent half of that number?
    - 1.000—000 x 2-127 *But we cannot fit 127 into the exponent field*
    - 0.100—000 x 2-126 *But we are stuck with that implied 1 before the implied point*
    - So, there are a lot of potentially useful values that don't fit into the scheme. The solution: special rules when the exponent has value 0 (which represents -126).

Denormalization

- This is denormalization: abandoning the “normal” scheme to exploit possibilities that would otherwise not be available.

Denormalized	+ -	0	Any nonzero bit pattern

No implied 1 before the implied point

Power of two multiplier is -127

Representing Zero

- How do you represent exactly 0 if there is an implied 1 in the number somewhere?
- A special case of denormalized numbers, when everything is zero, the value of the number is exactly 0.0

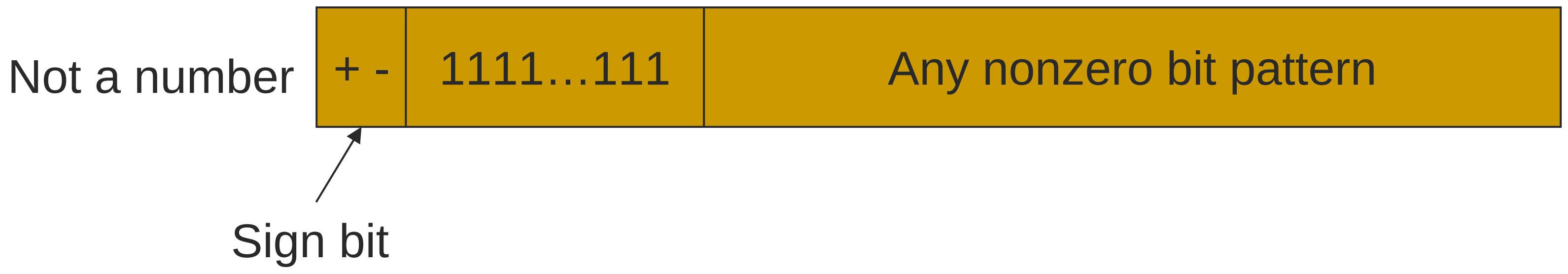
Infinity

- A special representation is reserved for infinity, because it is a useful entity to have available.

Infinity	+ -	1111...111	0

NaN (Not a Number)

- One more special case reserved for undefined results:



IEEE numerical types summary

Normalized	+ -	0<Exp<Max	Any bit pattern
Denormalized	+ -	0	Any nonzero bit pattern
Zero	+ -	0	0
Infinity	+ -	1111...111	0
Not a number	+ -	1111…111	Any nonzero bit pattern

Sign bit