CSC 411

Computer Organization (Spring 2024)
Lecture 7: Floating Point

Prof. Marco Alvarez, University of Rhode Island

Fractional binary numbers

- Used to represent fractional number
- Binary point
  - separates the integer and fractional parts (similar to the decimal point)
  - bits to the right of binary point represent fractional powers of 2

2^{i}	2^{i-1}		2	1	1/2	1/4		2^{-j+1}	2^{-j}
b_{i}	b_{i-1}	…	b_{1}	b_{0} •	b_{-1}	b_{-2}	…	b_{-j+1}	b_{-j}

$$\sum_{k=-j}^{i} b_k 2^k$$

11.010 =

1101.101 =

Practice

- Convert fractional binary numbers to decimal

1.10 =

11.001 =

100.01 =

11.111 =

Observations

- Not all decimal fractions have exact binary equivalents
  - can only represent numbers of the form $\dfrac{x}{2^k}$
  - e.g., 0.3 is represented as 0.010101…
    - requires an infinite series of 0s and 1s
  - large repeating decimals can only be approximated within a certain degree of accuracy
- 0.111111… represents a number just below 1.0

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots \frac{1}{2^i} + \ldots = 1.0 - \varepsilon$$

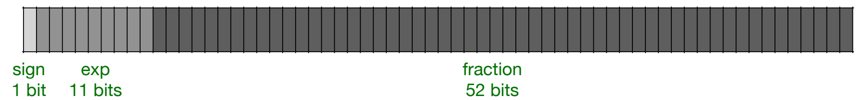
IEEE standard 754

- Defines a common format for representing real numbers in computers
  - developed in response to divergence of representations
- Supported by major CPUs (almost universally adopted)
- Provides different precision levels (single, double, extended) for various needs
- Standardizes the following layout :

Precision

- **Single-precision** (32 bits)
  - good balance of performance and range

- **Double-precision** (64 bits)
  - higher precision for demanding calculations



Precision

Format	Smallest positive value	Largest positive value	Precision
single	$\sim 1.401 \cdot 10^{-45}$	$\sim 3.403 \cdot 10^{+38}$	6-9 digits
double	$\sim 4.941 \cdot 10^{-324}$	$\sim 1.798 \cdot 10^{+308}$	15-17 digits

(*) Smallest/largest negative values are the same as their positive counterparts, but negative.

(**) Precision refers to to the number of significant digits that can be represented in a number.

Normalized and denormalized numbers

- Normalized
  - represent a wide range of values with maximum precision
  - **exp** != 000…000 and **exp** != 111…111
- Denormalized
  - used for very small numbers providing more range, they have lower precision compared to normalized number
  - **exp** == 000…000
- Special numbers
  - e.g., NaN, infinity
  - **exp** == 111…111

Floating point encoding/decoding

$$(-1)^s M 2^E$$

- **sign bit S**
  - -1^0 for positive, -1^1 for negative
- **exponent E**
  - magnitude of the number, the term 2^E scales the mantissa
- **mantissa M**
  - a.k.a. **significand**, it captures the significant digits of the number
    - scaled by the exponent, normally in range $[1.0,2.0)$

Normalized values

$$(-1)^s M 2^E$$

- **E = exp - bias**
  - exp is the unsigned integer represented by the bits in the exp field
  - bias is $2^{k-1} - 1$, where k is the length in bits of the exp field
    - single precision (bias = $2^{8-1} - 1 = 127$)
    - double precision (bias = $2^{11-1} - 1 = 1023$)
- **M = 1.bb…bb**
  - bb…bb are the bits in the fraction field
  - M is the decimal that corresponds to 1.bb…bb
    - note that M has an implied leading 1
    - what are the minimum and maximum decimal values represented by M?

sign	exp	fraction

Encoding example

$$(-1)^s M 2^E$$

- Assume a float F = 2024.0
  - fractional binary: 11111101000
  - make it form 1.xxx…: 1.1111101000 x 2^{10}
    - M = 1.1111101000
    - frac = 11111010000000000000000
  - calculate exp using E = exp - bias
    - exp = E + bias = 10 + 127 = 137 = 10001001

0 1 0 0 0 1 0 0 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0x44FD0000

Encoding example

$$(-1)^s M 2^E$$

- Assume a float F = 0.5
  - fractional binary: 0.1
  - make it form 1.xxx…: 1.0 x 2^{-1}
    - M = 1.0
    - frac = 00000000000000000000000
  - calculate exp using E = exp - bias
    - exp = E + bias = -1 + 127 = 126 = 01111110

Decoding example

$$(-1)^s M 2^E$$

- Assume a float F = 0x43CDA000
  - write the binary
    - extract s, exp, frac
    - 0 10000111 10011011010000000000000
  - calculate M and E
    - M = 1.00110110100000000000000 = 1.6064453125
    - E = exp - bias = 135 - 127 = 8
  - write final number
    - $(-1)^0$ 1.6064453125 * 2^8 = 411.25