CSCI-UA.0201

Computer Systems Organization

Data Representation – Floating points

Thomas Wies
wies@cs.nyu.edu
https://cs.nyu.edu/wies

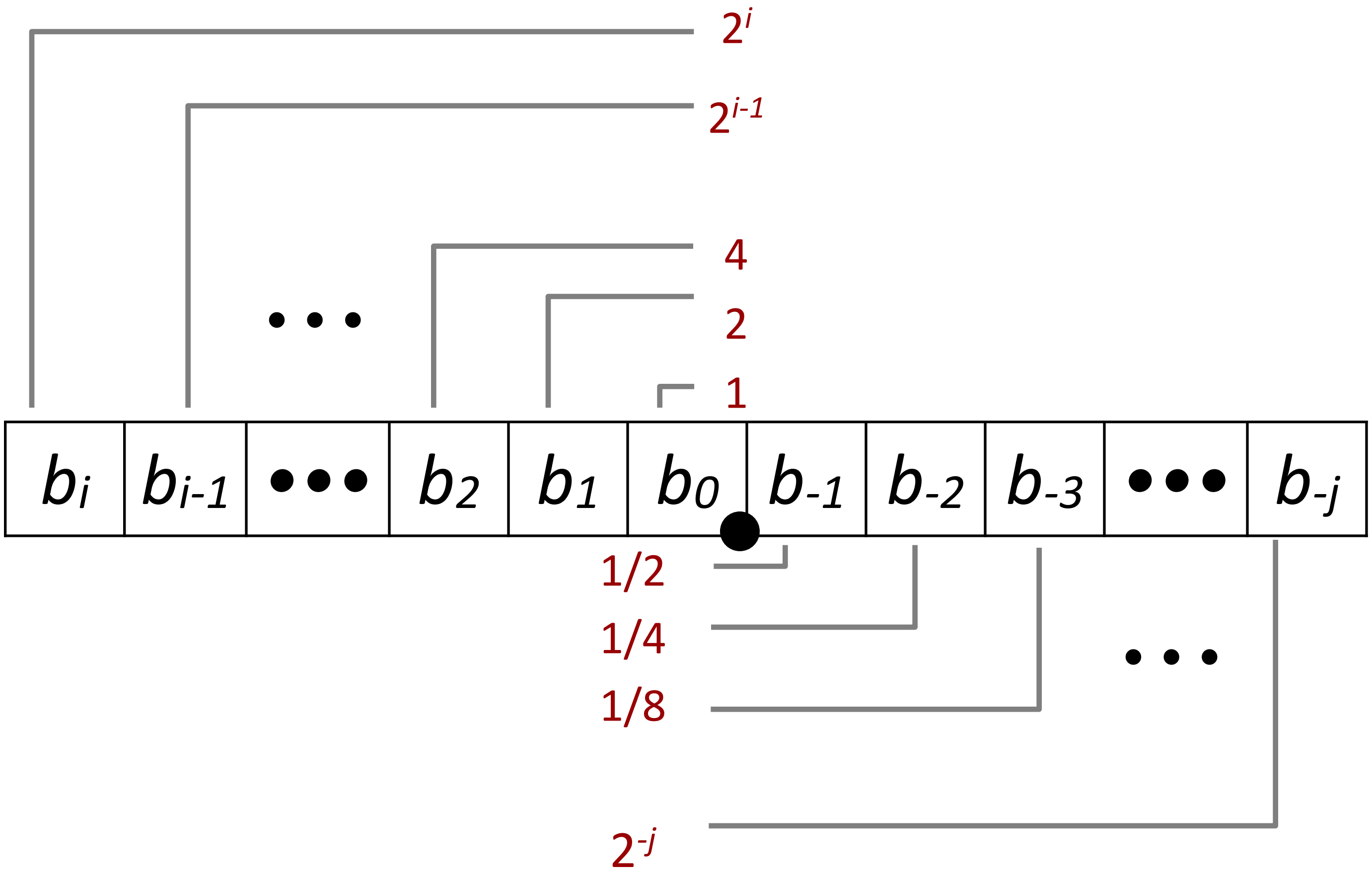
Floating Points

Some slides and information about FP are adopted from
Prof. Michael Overton book:
Numerical Computing with IEEE Floating Point Arithmetic

Background: Fractional binary numbers

- What is 1011.101_2?

Background: Fractional Binary Numbers



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

Fractional Binary Numbers: Examples

Value	Representation
5 3/4	101.11_2
2 7/8	10.111_2

Why not fractional binary numbers?

- Not efficient
  - $3 * 2^{100} \rightarrow 1010\underbrace{000000\ \ldots\ldots}_{\text{100 zeros}}\ \ 0$
  - Given a finite length (e.g. 32-bits), cannot represent very large numbers nor numbers very close to 0

IEEE Floating Point

- IEEE Standard 754
  - Supported by all major CPUs
  - The IEEE standards committee consisted mostly of hardware people, plus a few academics led by W. Kahan at Berkeley.

- Main goals:
  - Consistent representation of floating point numbers by all machines .
  - Correctly rounded floating point operations.
  - Consistent treatment of exceptional situations such as division by zero.

Floating Point Representation

- Numerical Form:

$$(-1)^s \boldsymbol{M} \; 2^{\boldsymbol{E}}$$

  - **Sign bit *s*** determines whether number is negative or positive
  - **Significand *M*** a fractional value
  - **Exponent *E*** weights value by power of two

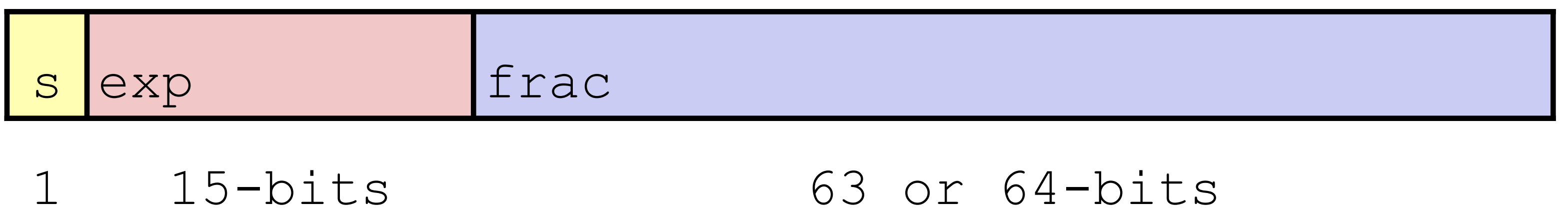
- Encoding
  - MSB `s` is sign bit ***s***
  - `exp` field encodes ***E***
  - `frac` field encodes ***M***

Precisions

- Single precision: 32 bits

- Double precision: 64 bits

- Extended precision: 80 bits (Intel only)



Based on exp we have 3 encoding schemes

- exp ≠ 0..0 or 11…1 → normalized encoding
- exp = 0… 000 → denormalized encoding
- exp = 1111…1 → special value encoding
  - frac = 000…0
  - frac = something else

1. Normalized Encoding

- Condition: exp ≠ 000…0 and exp ≠ 111…1

- Exponent is: $E = Exp - \underbrace{(2^{k-1} - 1)}^{\text{referred to as Bias}}$, k is the # of exponent bits
  - Single precision: E = exp – 127
  - Double precision: E = exp – 1023

Range(E)=[-126,127]
Range(E)=[-1022,1023]

- Significand is: $M = 1.\overbrace{\mathtt{xxx...x}}^{\text{frac}}{}_2$
  - Range(M) = [1.0, 2.0-ε)
  - Get extra leading bit for free

Normalized Encoding Example

- Value: `Float F = 15213.0;`

  $15213_{10} = 11101101101101_2$

  $= 1.1101101101101_2 \times 2^{13}$

- Significand

  $M = 1.\underline{1101101101101}_2$

  frac= $\underline{1101101101101}0000000000_2$

- Exponent

  E = exp – Bias = exp - 127 = 13

  ➔ exp = 140 = $\mathbf{10001100_2}$

- Result:

0	10001100	11011011011010000000000
s	**exp**	**frac**

2. Denormalized Encoding
(called subnormal in revised standard 854)

- Condition: `exp = 000…0`

- Exponent value: ***E*** = 1 – ***Bias*** (instead of ***E*** = 0 – ***Bias***)
- Significand is: ***M*** = $0.\underbrace{\text{xxx…x}}_{\text{frac}}{}_2$ (instead of $\boldsymbol{M=1.xxx_2}$)

- Cases
  - **`exp`** = `000…0`, **`frac`** = `000…0`
    - Represents zero
    - Note distinct values: +0 and –0
  - **`exp`** = `000…0`, **`frac`** ≠ `000…0`
    - Numbers very close to 0.0

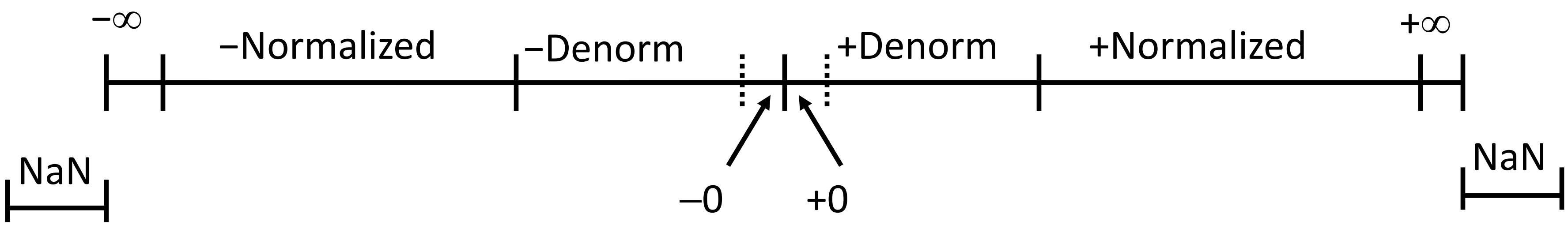
3. Special Values Encoding

- Condition: **`exp`** = `111…1`

- Case: **`exp`** = `111…1`, **`frac`** = `000…0`
  - Represents value ∞ (infinity)
  - Used for operations that overflow
  - E.g., 1.0/0.0 = −1.0/−0.0 = +∞, 1.0/−0.0 = −∞

- Case: **`exp`** = `111…1`, **`frac`** $\neq$ `000…0`
  - Not-a-Number (NaN)
  - Represents case when no numeric value can be determined
  - E.g., sqrt(–1), $\infty - \infty$, $\infty \times 0$

Visualization: Floating Point Encodings



Rounding modes

IEEE 754 supports five rounding modes:

- Round to nearest even (default)
  - if fractional part < .5, round to 0
  - if fractional part > .5, round away from 0
  - if fractional part = .5, round to nearest even digit
- Round to nearest (tie: round away from 0)
- Round to 0
- Round down (to $-\infty$)
- Round up (to $+\infty$)

Floating Point Operations

Example: Compute $z = x + y$ where
$x = 123456.7 = 1.234567 \times 10^5$
$y = 101.7654 = 1.017654 \times 10^2$

x: exp = 5 frac = 1.234567
y: exp = 2 frac = 1.017654

Adjust exp of y by shifting frac:
y: exp = 5 frac = 0.001017654

Add frac of x and y:
z: exp = 5 frac = 1.235584654

Round frac
z: exp = 5 frac = 1.235585

Floating Point in C

- C:
  - `float` single precision
  - `double` double precision
- Conversions/Casting
  - Casting between **`int`**, **`float`**, and **`double`** changes bit representation, examples:
  - **`double`**/**`float`** → **`int`**
    - Truncates fractional part
    - Not defined when out of range or NaN
  - **`int`** → **`double`**
    - Exact conversion

Conclusions

- Everything is stored in memory as 1s and 0s
- The binary presentation by itself does not carry a meaning, it depends on the interpretation.
- IEEE Floating Point has clear mathematical properties
  – Represents numbers as: $(-1)^S \times M \times 2^E$