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

<table>
<thead>
<tr>
<th>Value</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 3/4</td>
<td>101.11_2</td>
</tr>
<tr>
<td>2 7/8</td>
<td>10.111_2</td>
</tr>
</tbody>
</table>
Why not fractional binary numbers?

• Not efficient
  – $3 \times 2^{100} \rightarrow 10100000000 \ldots 0$

  – 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:
  \((-1)^s \, M \, 2^E\)
  – 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

- s
- exp
- frac

  1  8-bits  23-bits

• Double precision: 64 bits

- s
- exp
- frac

  1  11-bits  52-bits

• Extended precision: 80 bits (Intel only)

- s
- exp
- frac

  1  15-bits  63 or 64-bits
Based on $\exp$

we have 3 encoding schemes

- $\exp \neq 0..0$ or $11...1 \rightarrow$ normalized encoding
- $\exp = 0...000 \rightarrow$ denormalized encoding
- $\exp = 1111...1 \rightarrow$ special value encoding
  - $\text{frac} = 000...0$
  - $\text{frac} = $ something else
1. Normalized Encoding

- Condition: \( \text{exp} \neq 000...0 \) and \( \text{exp} \neq 111...1 \)

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

Significand is: \( M = \text{1.} \times \times \times \times _{2} \)
  - Range(\( M \)) = [1.0, 2.0-\( \varepsilon \)]
  - Get extra leading bit for free
Normalized Encoding Example

- **Value:** `Float F = 15213.0;`
  
  \[
  15213_{10} = 111011011011012
  = 1.11011011011012 \times 2^{13}
  \]

- **Significand**
  
  \[
  M = 1.11011011011012
  \frac{\text{frac}}{\text{frac}} = 11011011011010000000000002
  \]

- **Exponent**
  
  \[
  E = \exp - \text{Bias} = \exp - 127 = 13
  \Rightarrow \exp = 140 = 100011002
  \]

- **Result:**
  
  \[
  0 \begin{array}{c}
  10001100 \\
  1101101101101000000000000
  \end{array}
  \]

  \[s \quad \text{exp} \quad \text{frac}\]
2. Denormalized Encoding
(called subnormal in revised standard 854)

• Condition: $\exp = 000\ldots0$

• Exponent value: $E = 1 - \text{Bias}$ (instead of $E = 0 - \text{Bias}$)

• Significand is: $M = 0.xxx\ldots x_2$ (instead of $M=1.xxx_2$)

• Cases
  
  $\exp = 000\ldots0$, $\frac{\text{frac}}{\text{frac}} = 000\ldots0$
  
  • Represents zero
  
  • Note distinct values: +0 and −0

  $\exp = 000\ldots0$, $\frac{\text{frac}}{\text{frac}} \neq 000\ldots0$
  
  • Numbers very close to 0.0
3. Special Values Encoding

• Condition: \( \text{exp} = 111...1 \)

• Case: \( \text{exp} = 111...1, \frac{\text{frac}}{} = 000...0 \)
  – Represents value \( \infty \) (infinity)
  – Used for operations that overflow
  – E.g., \( 1.0/0.0 = -1.0/-0.0 = +\infty, 1.0/-0.0 = -\infty \)

• Case: \( \text{exp} = 111...1, \frac{\text{frac}}{} \neq 000...0 \)
  – 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

-∞  -Normalized  -Denorm  +Denorm  +Normalized  +∞

-0  +0

NaN  NaN
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
\[
\begin{align*}
x &= 123456.7 = 1.234567 \times 10^5 \\
y &= 101.7654 = 1.017654 \times 10^2
\end{align*}
\]

\[
\begin{align*}
x: \ exp &= 5 \quad \text{frac} = 1.234567 \\
y: \ exp &= 2 \quad \text{frac} = 1.017654
\end{align*}
\]

Adjust exp of \( y \) by shifting frac:
\[
\begin{align*}
y: \ exp &= 5 \quad \text{frac} = 0.001017654
\end{align*}
\]

Add frac of \( x \) and \( y \):
\[
\begin{align*}
z: \ exp &= 5 \quad \text{frac} = 1.235584654
\end{align*}
\]

Round frac
\[
\begin{align*}
z: \ exp &= 5 \quad \text{frac} = 1.235585
\end{align*}
\]
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: \((-1)^S \times M \times 2^E\)