Floating point

Zhaoguo Wang
The Real Numbers
The Real Numbers

What we have studied
Today – Represent real numbers using bits
## Decimal Representation

<table>
<thead>
<tr>
<th>Real Numbers</th>
<th>Decimal Representation (Expansion)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{11}{2}$</td>
<td>$(5.5)_{10}$</td>
</tr>
<tr>
<td>$\frac{1}{3}$</td>
<td>$(0.3333333\ldots)_{10}$</td>
</tr>
<tr>
<td>$\sqrt{2}$</td>
<td>$(1.4128\ldots)_{10}$</td>
</tr>
</tbody>
</table>
Decimal Representation

Real Numbers | Decimal Representation (Expansion)
---|---
11 / 2 | (5.5)$_{10}$
1 / 3 | (0.3333333...)$_{10}$
$\sqrt{2}$ | (1.4128...)$_{10}$

(5.5)$_{10} = 5 \times 10^0 + 5 \times 10^{-1}$

(0.3333333...)$_{10} = 3 \times 10^{-1} + 3 \times 10^{-2} + 3 \times 10^{-3} + ...$

(1.4128...)$_{10} = 1 \times 10^0 + 4 \times 10^{-1} + 1 \times 10^{-2} + 2 \times 10^{-3} + ...$
# Decimal Representation

<table>
<thead>
<tr>
<th>Real Numbers</th>
<th>Decimal Representation (Expansion)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$11 / 2$</td>
<td>$(5.5)_{10}$</td>
</tr>
<tr>
<td>$1 / 3$</td>
<td>$(0.3333333...)_ {10}$</td>
</tr>
<tr>
<td>$\sqrt{2}$</td>
<td>$(1.4128...)_ {10}$</td>
</tr>
</tbody>
</table>

$(5.5)_{10} = 5 * 10^0 + 5 * 10^{-1}$

$(0.3333333...)_{10} = 3 * 10^{-1} + 3 * 10^{-2} + 3 * 10^{-3} + ...$

$(1.4128...)_{10} = 1 * 10^0 + 4 * 10^{-1} + 1 * 10^{-2} + 2 * 10^{-3} + ...$

$r_{10} = (d_m d_{m-1}...d_1 d_0 \cdot d_{-1} d_{-2}...d_{-n})_{10}$

$$= \sum_{i=-n}^{m} 10^i \times d_i$$
Binary Representation

\[(5.5)_{10} = 4 + 1 + 1 / 2\]
\[= 1 \times 2^2 + 1 \times 2^0 + 1 \times 2^{-1}\]
Binary Representation

\[(5.5)_{10} = 4 + 1 + 1/2 \]
\[= 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1}\]
**Binary Representation**

\[(5.5)_{10} = 4 + 1 + 1 / 2 \]
\[= 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} \]
\[= (101.1)_{2} \]
Binary Representation

\[(5.5)_{10} = 4 + 1 + 1/2 \]
\[= 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} \]
\[= (101.1)_2 \]

\[(0.333333...)_{10} = 1/4 + 1/16 + 1/64 + ... \]
\[= (0.01010101...)_2 \]
Binary Representation

\[ r_{10} = (d_m d_{m-1} d_1 d_0 \cdots d_{-1} d_{-2} \cdots d_{-n})_{10} \]

\[ = (b_p b_{p-1} b_1 b_0 \cdots b_{-1} b_{-2} \cdots b_{-q})_2 \]

\[ \left( b_p b_{p-1} \cdots b_1 b_0 \cdot b_{-1} b_{-2} \cdots b_{-q} \right)_2 = \sum_{i=-q}^{p} 2^i \times b_i \]
<table>
<thead>
<tr>
<th>Binary Expansion</th>
<th>Formula</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10.011_2$</td>
<td>$2^{-3} + 2^{-4} + 2^{-6}$</td>
<td>$2^{-1} + 2^{-2} + 2^{-3} + 2^{-4}$</td>
</tr>
<tr>
<td>Binary Expansion</td>
<td>Formula</td>
<td>Decimal</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>10.011&lt;sub&gt;2&lt;/sub&gt;</td>
<td>2&lt;sup&gt;1&lt;/sup&gt; + 2&lt;sup&gt;-2&lt;/sup&gt; + 2&lt;sup&gt;-3&lt;/sup&gt;</td>
<td>2.375&lt;sub&gt;10&lt;/sub&gt;</td>
</tr>
<tr>
<td>0.001101&lt;sub&gt;2&lt;/sub&gt;</td>
<td>2&lt;sup&gt;-3&lt;/sup&gt; + 2&lt;sup&gt;-4&lt;/sup&gt; + 2&lt;sup&gt;-6&lt;/sup&gt;</td>
<td>0.203125&lt;sub&gt;10&lt;/sub&gt;</td>
</tr>
<tr>
<td>0.1111&lt;sub&gt;2&lt;/sub&gt;</td>
<td>2&lt;sup&gt;-1&lt;/sup&gt; + 2&lt;sup&gt;-2&lt;/sup&gt; + 2&lt;sup&gt;-3&lt;/sup&gt; + 2&lt;sup&gt;-4&lt;/sup&gt;</td>
<td>0.9375&lt;sub&gt;10&lt;/sub&gt;</td>
</tr>
</tbody>
</table>
Intuitive Idea

Fixed point

**15 bits**

| s | b_{14} | b_{13} | b_{12} | ... | b_{1} | b_{0} | b_{-1} | b_{-2} | b_{-3} | ... | b_{-14} | b_{-15} | b_{-16} |

Fixed position *e.g. middle*
Intuitive Idea

Fixed point

15 bits

sign

16 bits

Fixed position e.g. middle

(10.011)₂

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
Problems of Fixed Point

Limited range and precision: e.g., 32 bits
  – Largest number: $2^{15}$

→ Rarely used for numerical computing
Floating Point

Based on exponential notation

\[ r_{10} = \pm M \times 10^E, \text{ where } 1 \leq M < 10 \]

M: significant (mantissa), E: exponent
Floating Point

Based on exponential notation

\[ r_{10} = \pm M \times 10^E, \text{ where } 1 \leq M < 10 \]

M: significant (mantissa), E: exponent

365.25 = 3.6525 \times 10^2

0.0123 = 1.23 \times 10^{-2}
Floating Point

Based on exponential notation

\[ r_{10} = \pm M \times 10^E, \text{ where } 1 \leq M < 10 \]

M: significant (mantissa), E: exponent

365.25 = 3.6525 \times 10^2

0.0123 = 1.23 \times 10^{-2}

Decimal point *floats* to the position immediately after the first nonzero digit.
Floating Point

Binary exponential representation

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]

\[ M = (1.b_1 b_2 b_3\ldots b_n)_2 \]

M: significant, E: exponent

\[ (5.5)_{10} = (101.1)_2 = (1.011)_2 \times 2^2 \]
Floating Point

Binary exponential representation

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]

\[ M = (1. b_1 b_2 b_3 \ldots b_n)_2 \]

M: significant, E: exponent

\[ (5.5)_{10} = (101.1)_2 = (1.011)_2 \times 2^2 \]

Normalization: give a number \( r \), obtain its normalized representation
Exercises

The normalized representation of $(10.25)_10$ is?
The normalized representation of $(10.25)_{10}$ is ?

$$(10.25)_{10} = (1010.01)_{2} = (1.01001)_{2} \times 2^{3}$$
Floating Point

Binary exponential representation

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]
\[ M = (1.b_1b_2b_3...b_n)_2 \]

M: significant, E: exponent

\[(5.5)_{10} = (101.1)_2 = (1.011)_2 \times 2^2\]

Store a normalized number in computer?
Normalized representation in computer

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]

\[ M = (1.b_1b_2b_3...b_n)_2 \]

\( M \): significant, \( E \): exponent
Normalized representation in computer

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]

\[ M = (1.b_1b_2b_3\ldots b_{23})_2 \]

M: significant, E: exponent

<table>
<thead>
<tr>
<th>31 30</th>
<th>23 22</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>exp (E)</td>
<td>fraction (F)</td>
</tr>
</tbody>
</table>

\[ (b_1b_2b_3\ldots b_{23})_2 \]

precision: number of bits in significant field (23 + 1).
Normalized representation in computer

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]

\[ M = (1.b_1 b_2 b_3 \ldots b_{23})_2 \]

M: significant, E: exponent

(5.5)_{10} = (101.1)_2 = (1.011)_2 \times 2^2
Exercise

Given the normalized representation (by computer) of \((71)_{10}\) and \((10.25)_{10}\) (hint: \(0.25 = 2^{-2}\))
Exercise

Given the normalized representation (by computer) of \((71)_{10}\) and \((10.25)_{10}\) (hint: \(0.25 = 2^{-2}\))

\[ (10.25)_{10} = (1010.01)_2 = (1.01001)_2 \times 2^3 \]

\[ (71)_{10} = (1000111)_2 = (1.000111)_2 \times 2^6 \]
Toy Number System

6-bit floating point representation
- exponent: 3 bits
- fraction: 2 bits

Largest positive number?
Toy Number System

6-bit floating point representation
- exponent: 3 bits
- fraction: 2 bits

Largest positive number?

\[(1.11)_2 \times 2^7 = 224\]
Toy Number System

6-bit floating point representation
- exponent: 3 bits
- fraction: 2 bits

Largest positive number: 224

Smallest positive number?
Toy Number System

6-bit floating point representation
- exponent: 3 bits
- fraction: 2 bits

Largest positive number: 224
Smallest positive number: 1

\[ (1.00)_2 \times 2^0 = 1 \]
Toy Number System

6-bit floating point representation
- exponent: 3 bits
- fraction: 2 bits

Positive number: 1 to 224
Negative number: -224 to -1
Normalized representation in computer

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]

\[ M = (1.b_1b_2b_3...b_n)_2 \]

M: significant, E: exponent

How to represent numbers between -1 and 1?
Questions

How to represent
1. number close or equal to 0?
2. larger numbers, even $\infty$ ?
3. the result of dividing by 0 ?
Questions

How to represent
1. number close or equal to 0?
2. larger numbers, even \( \infty \) ?
3. the result of dividing by 0 ?

Lots of different implementations around 1950s!
IEEE Floating Point Standard

IEEE p754
A standard for binary floating point representation

Prof. William Kahan
University of California at Berkeley
Turing Award (1989)
The Only Book Focuses On IEEE Floating Point Standard

Numerical Computing with IEEE Floating Point Arithmetic

Including One Theorem, One Rule of Thumb, and One Hundred and One Exercises

Michael L. Overton
Courant Institute of Mathematical Sciences
New York University
New York, New York

Lots of interesting gossip!

With you nyu netid/password. You can also search the pdf with google.
Goals of IEEE Standard

Consistent representation of floating point numbers by all machines adopting the standard

Correctly rounded floating point operations, using several rounding modes, since exact answers often cannot be represented exactly on the computer

Consistent treatment of exceptional situations such as division by zero
Restrictions for Normalized Representation

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]

\[ M = (1.b_0b_1b_2b_3...b_n)_2 \]

M: significant, E: exponent

31 30 23 22

\[ s \quad \exp (E) \quad \text{fraction (F)} \]

\[ (b_0b_1b_2b_3...b_n)_2 \]

E can not be \((1111 1111)_2\) or \((0000 0000)_0\)
Restrictions for Normalized Representation

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]

\[ M = (1.b_0b_1b_2b_3...b_n)_2 \]

M: significant, E: exponent

\[ E_{\text{max}} = ? \]
\[ E_{\text{min}} = ? \]
Restrictions for Normalized Representation

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]

\[ M = (1.b_0b_1b_2b_3...b_n)_2 \]

M: significant, E: exponent

\[ s \]

exp (E)  fraction (F)

\[ (b_0b_1b_2b_3...b_n)_2 \]

E cannot be \((1111\ 1111)_2\) or \((0000\ 0000)_0\)

\[ E_{\text{max}} = 254, \ (1111\ 1110)_2 \]

\[ E_{\text{min}} = 1, \ (0000\ 0001)_2 \]
represent values between -1 and 1 in normalized representation
Exponential Bias

\[ r_{10} = \pm M \times 2^E, \quad \text{where } 1 \leq M < 2 \]

\[ M = (1.b_0b_1b_2b_3\ldots b_n)_2 \]

M: significant, E: exponent

Bias: 127
Exponential Bias

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]

\[ M = (1.b_0b_1b_2b_3...b_n)_2 \]

M: significant, E: exponent

Bias: 127

\[ E_{\text{max}} = ? \]

\[ E_{\text{min}} = ? \]
Exponential Bias

\[ r_{10} = \pm M \times 2^E, \quad \text{where} \quad 1 \leq M < 2 \]

\[ M = (1.b_0b_1b_2b_3...b_n) \_2 \]

M: significant, E: exponent

Bias: 127

\[ E_{\text{max}} = 254 - 127 = 127 \]

\[ E_{\text{min}} = 1 - 127 = -126 \]
Exponential Bias

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]

\[ M = (1.b_0 b_1 b_2 b_3 \ldots b_n)_2 \]

M: significant, E: exponent

Bias: 127

\[ E_{\text{max}} = 254 - 127 = 127 \]

\[ E_{\text{min}} = 1 - 127 = -126 \]

Smallest positive number: ?

Negative number with smallest absolute value: ?
Exponential Bias

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]

\[ M = (1.b_0 b_1 b_2 b_3 \ldots b_n)_2 \]

M: significant, E: exponent

Bias: 127

\[ E_{\text{max}} = 254 - 127 = 127 \]

Smallest positive number: \(2^{-126}\)

\[ E_{\text{min}} = 1 - 127 = -126 \]

Negative number with smallest absolute value: \(-2^{-126}\)
Questions from Munachiso

Q1. Why does it need bias?

Q2. Why is the bias 127?
Questions from Munachiso

Q1. Why does it need bias?

A1. Use the unsigned number to represent negative numbers (-1 ~ -126)
Q2. Why is 127?

A2. Balance positive numbers and negative numbers
Represent negative exponent

\[ r_{10} = \pm M \times 2^E, \text{ where } 1 \leq M < 2 \]

\[ M = (1.b_0b_1b_2b_3...b_n)_2 \]

M: significant, E: exponent

Intuitive solution: two’s complement
Represent negative exponent

\[ a = 1.0 \times 2^3 \]
\[ b = 1.0 \times 2^{-3} \]

Intuitive solution: two’s complement
Represent negative exponent

\[ a = 1.0 \times 2^3 \]
\[ b = 1.0 \times 2^{-3} \]

To compare \( a \) and \( b \), CPU needs to have hardware logic for two’s complement.
Represent negative exponent

\[ a = 1.0 \times 2^3 \]
\[ b = 1.0 \times 2^{-3} \]

With bias 127
Represent negative exponent

\[ a = 1.0 \times 2^3 \]
\[ b = 1.0 \times 2^{-3} \]

With bias, a comparison can be done easily: compare their representations bitwise from left to right, stopping as soon as the first differing bit is encountered.

*(Refer to Exercise 4.3 in Michael’s book)*
Toy Number System

6-bit floating point representation
- exponent: 3 bits
- fraction: 2 bits
- bias: 3

Smallest positive number ?
Toy Number System

6-bit floating point representation
- exponent: 3 bits
- fraction: 2 bits
- bias: 3

Smallest positive number: 0.25

\[ (1.00)_2 \times 2^{-2} = 0.25 \]
Toy Number System

6-bit floating point representation
- exponent: 3 bits
- fraction: 2 bits
- bias: 3
Toy Number System

6-bit floating point representation
- exponent: 3 bits
- fraction: 2 bits
- bias: 3
Toy Number System

6-bit floating point representation
- exponent: 3 bits
- fraction: 2 bits
- bias: 3
represent values which are close and equal to 0
Denormalization

\[ r_{10} = \pm M \times 2^E, \text{ M: significant, E: exponent} \]

Normalized Encoding:

\[ \begin{array}{cccccc}
31 & 30 & 23 & 22 & 21 & 0 \\
\text{s} & \text{exp} (E) + 127 & \text{fraction (F)} & \\
\end{array} \]

\[ 1 \leq M < 2, \ M = (1.F)_2 \]
Denormalization

\[ r_{10} = \pm M \times 2^E, \ M: \text{significant}, \ E: \text{exponent} \]

**Normalized Encoding:**

\[
\begin{array}{cccccc}
31 & 30 & 29 & 28 & 23 & 22 & 21 & \cdots & 0 \\
\text{s} & \text{exp (E) + 127} & \text{fraction (F)} & \\
\end{array}
\]

\[ 1 \leq M < 2, \ M = (1.F)_2 \]

**Denormalized Encoding:**

\[
\begin{array}{cccccc}
31 & 30 & 29 & 28 & 23 & 22 & 21 & \cdots & 0 \\
\text{s} & 0000 \ 0000 & \text{fraction (F)} & \\
\end{array}
\]

\[ E = 1 - \text{Bias} = -126 \]

\[ 0 \leq M < 1, \ M = (0.F)_2 \]
Zeros

+0.0

0 | 0000 0000 | 0000 0000 0000 0000 0000 0000 000

-0.0

1 | 0000 0000 | 0000 0000 0000 0000 0000 0000 000
Examples

(0.1)_2 \times 2^{-126}

\begin{array}{c|c}
0 & 0000 0000 \\
\hline
1 & 0000 0000 \\
\end{array}
\begin{array}{c|c}
\vdots & 1000 0000 0000 0000 0000 0000 000 \\
\hline
1 & 0101 0100 0000 0000 0000 0000 000 \\
\end{array}

-(0.010101)_2 \times 2^{-126}

\begin{array}{c|c}
0 & 0000 0000 \\
\hline
1 & 0000 0000 \\
\end{array}
\begin{array}{c|c}
\vdots & 1000 0000 0000 0000 0000 0000 000 \\
\hline
1 & 0101 0100 0000 0000 0000 0000 000 \\
\end{array}
Toy Number System

6-bit floating point representation
- exponent: 3 bits
- fraction: 2 bits
- bias: 3
- **Denormalized encoding**
## Special Values

### Special Value's Encoding:

<table>
<thead>
<tr>
<th>values</th>
<th>sign</th>
<th>frac</th>
</tr>
</thead>
<tbody>
<tr>
<td>+∞</td>
<td>0</td>
<td>all zeros</td>
</tr>
<tr>
<td>- ∞</td>
<td>1</td>
<td>all zeros</td>
</tr>
<tr>
<td>Nan</td>
<td>any</td>
<td>non-zero</td>
</tr>
</tbody>
</table>
## Exercises

<table>
<thead>
<tr>
<th>representation</th>
<th>E</th>
<th>M</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100 1001 0101 0000 0000 0000 0000 0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111 1111 1111 1111 0000 0000 0000 0000</td>
<td></td>
<td></td>
<td>2.5 * 2^{-127}</td>
</tr>
<tr>
<td>1111 1111 1000 0000 0000 0000 0000 0000</td>
<td></td>
<td></td>
<td>-1.25 * 2^{-111}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.5 * 2^{-127}</td>
</tr>
</tbody>
</table>
### Exercises

<table>
<thead>
<tr>
<th>representation</th>
<th>E</th>
<th>M</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100 1001 0101 0000000000</td>
<td>146 – 127 = 19</td>
<td>(1.101)₂ = 1.625</td>
<td>1.625 * 2^19</td>
</tr>
<tr>
<td>0000 0000 1010 00000000000000</td>
<td>1 – 127 = -126</td>
<td>(1.01)₂ = 1.25</td>
<td>2.5 * 2^-127 = (1.01)₂ * 2^-126</td>
</tr>
<tr>
<td>1000 1000 0010 00000000000000</td>
<td>16 – 127 = -111</td>
<td>(1.01)₂ = 1.125</td>
<td>-1.25 * 2^-111</td>
</tr>
<tr>
<td>1111 1111 1111 1111 0000 00000000000000</td>
<td>-</td>
<td>-</td>
<td>Nan</td>
</tr>
<tr>
<td>1111 1111 1000 00000000000000</td>
<td>-</td>
<td>-</td>
<td>- ∞</td>
</tr>
<tr>
<td>0000 0000 0110 000000000000000000</td>
<td>-126</td>
<td>(0.11)₂</td>
<td>(0.11)₂ * 2^-126 = 1.5 * 2^-127</td>
</tr>
</tbody>
</table>
Distribution of Representable Values

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

NaN −0 +0 NaN
Distribution of Representable Values

How to represent the point \( \bullet \) in the format?
Rounding

Goal

– Given a value $x$, finding the “closest” matching representable value $x'$.

Round modes

– Round-down
– Round-up
– Round-toward-zero
– Round-to-nearest (Round to even in text book)
Round down

Round(x) = x

Close, but no larger than the true result (toward -∞)
Round down

Round(x) = x_
Close, but no larger than the true result (toward -∞)

Round(-0.86) = ?
Round(0.55) = ?
Round down

Round(x) = x_
Close, but no larger than the true result (toward - \infty )

Round(-0.86) = -0.875
Round(0.55) = 0.5
Round up

\[ \text{Round}(x) = x_+ \]
Close, but no less than the true result (toward \( +\infty \))
Round up

Round(x) = x_+

Close, but no less than the true result (toward +∞)

Round(-0.86) = ?

Round(0.55) = ?
Round up

Round(x) = x_+
Close, but no less than the true result (toward + ∞)

Round(-0.86) = -0.75
Round(0.55) = 0.625
Round towards zero

\[ \text{Round}(x) = x_+ \text{ if } x < 0 \]
\[ \text{Round}(x) = x_- \text{ if } x > 0 \]
Round towards zero

Round\( (x) = x_+ \) if \( x < 0 \)

Round\( (x) = x_- \) if \( x > 0 \)

Round\(-0.86\) = ?

Round\(0.55\) = ?
Round towards zero

Round\(x\) = \(x_+\) if \(x < 0\)

Round\(x\) = \(x_-\) if \(x > 0\)

Round\((-0.86) = -0.75\)

Round\((0.55) = 0.5\)
Round to nearest

Round(x) either $x_+$ or $x_-$, whichever is nearer to $x$. 
Round to nearest

Round(x) either $x_+$ or $x_-$, whichever is nearer to $x$.

Round(-0.86) = ?
Round(0.55) = ?
Round to nearest

Round(x) either \(x_+\) or \(x_-\), whichever is nearer to \(x\).

Round(-0.86) = -0.875
Round(0.55) = 0.5
Round to nearest

Round(x) either $x_+$ or $x_-$, whichever is nearer to $x$.

\[
\begin{align*}
\text{Round}(-0.86) &= -0.875 \\
\text{Round}(0.55) &= 0.5
\end{align*}
\]

In case of a tie, the one with its least significant bit equal to zero is chosen.
single/ double precision

**single precision** (32 bits)

float f = 0.1
double d = 0.1

double precision (64 bits)
## single/ double precision

<table>
<thead>
<tr>
<th></th>
<th>$E_{\text{min}}$</th>
<th>$E_{\text{max}}$</th>
<th>$N_{\text{min}}$</th>
<th>$N_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Float</td>
<td>-126</td>
<td>127</td>
<td>$\approx 2^{-126}$</td>
<td>$\approx 2^{128}$</td>
</tr>
<tr>
<td>Double</td>
<td>-1022</td>
<td>1023</td>
<td>$\approx 2^{-1022}$</td>
<td>$\approx 2^{1024}$</td>
</tr>
</tbody>
</table>
How does CPU know if it is floating point or integers?

By having specific instruction for floating point operation.
int d = 1 + 2

CPU

decoder

FPU

IU

add $1, $2

Memory

0x00...00b0
0x00...00a9
0x00...00a8
0x00...00a7
0x00...00a6
0x00...00a5
0x00...00a4
0x00...00a3
0x00...00a2
0x00...00a1

...
addss $1, $2

float f = 0.1 + 0.2
Our first lab