Floating-point numbers

Fractional binary numbers
IEEE floating-point standard
Floating-point operations and rounding

Lessons for programmers

Many more details we will skip (it’s a 58-page standard...)
See CSAPP 2.4 for more detail.

Fractional Binary Numbers

Value  Representation
5 and 3/4
2 and 7/8
47/64

Observations
Shift left =
Shift right =
Numbers of the form $0.11111\ldots_2$ are...?

Limitations:
Exact representation possible when?

$$1/3 = 0.333333\ldots_{10}$$

Fixed-Point Representation

Implied binary point.
$$b_7 b_6 b_5 b_4 b_3 \ldots b_2 b_1 b_0$$
$$b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$$

range: difference between largest and smallest representable numbers
precision: smallest difference between any two representable numbers

fixed point = fixed range, fixed precision
IEEE Floating Point Standard 754

Numerical form:

\[ V_{10} = (-1)^s \times M \times 2^E \]

Sign bit \( s \) determines whether number is negative or positive
Significand (mantissa) \( M \) usually a fractional value in range \([1.0,2.0)\)
Exponent \( E \) weights value by a \((-+/+)\) power of two
Analogous to scientific notation

Representation:

MSB \( s = \) sign bit \( s \)
exp field encodes \( E \) (but is not equal to \( E \))
frac field encodes \( M \) (but is not equal to \( M \))

Numerically well-behaved, but hard to make fast in hardware

Three kinds of values

\[ V = (-1)^s \times M \times 2^E \]

1. Normalized: \( M = 1.xxxxx\)
As in scientific notation: \(0.011 \times 2^5 = 1.1 \times 2^3\)
Representation advantage?

2. Denormalized, near zero: \( M = 0.xxxxx..., \) smallest \( E \)
Evenly space near zero.

3. Special values:
   0.0: \( s = 0 \) \( \exp = 00...0 \) \( \frac{\text{frac}}{\text{frac}} = 00...0 \)
   +inf, -inf: \( \exp = 11...1 \) \( \frac{\text{frac}}{\text{frac}} = 00...0 \)
   division by 0.0
   NaN ("Not a Number"): \( \exp = 11...1 \) \( \frac{\text{frac}}{\text{frac}} \neq 00...0 \)
sqrt(-1), \( -\infty, \infty \times 0 \), etc.

Value distribution

Infinitely many values:

Asymptotes:

- \( -\infty \) -Normalized
- \( +\infty \) +Normalized
- \( -0.0 \) -Denormalized
- \( +0.0 \) +Denormalized
- NaN

Precisions

Single precision (float): 32 bits

<table>
<thead>
<tr>
<th>1 bit</th>
<th>8 bits</th>
<th>23 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s )</td>
<td>exp</td>
<td>frac</td>
</tr>
</tbody>
</table>

Double precision (double): 64 bits

<table>
<thead>
<tr>
<th>1 bit</th>
<th>11 bits</th>
<th>52 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s )</td>
<td>exp</td>
<td>frac</td>
</tr>
</tbody>
</table>

Finite representation of infinite range...
2. Denormalized Values: near zero

"Near zero": \(\exp = 000...0\)

Exponent:

\[ E = 1 + \exp - \text{Bias} = 1 - \text{Bias} \quad \text{not: } \exp - \text{Bias} \]

Significand: leading zero

\[ M = 0.\text{xxx}...\text{x}_2 \]
\[ \frac{\text{frac}}{\text{xxx}...\text{x}} \]

Cases:

\[ \exp = 000...0, \frac{\text{frac}}{\text{xxx}...\text{x}_2} = 0.0, -0.0 \]
\[ \exp = 000...0, \frac{\text{frac}}{\text{xxx}...\text{x}_2} \neq 000...0 \]

Value distribution example (zoom in on 0)

6-bit IEEE-like format

Bias = \(2^{3-1} - 1 = 3\)

\[ s \ exp \ frac \]

\[ 1 \ 3 \ 2 \]

\(s=1, \exp=101\)

\(E = 5-3 = 2\)

\(s=0, \exp=010\)

\(E = 2-3 = 1\)

\(s=0, \exp=000\)

\(E = 1-3 = -2\)

\(s=0, \exp=001\)

\(E = 1-3 = -2\)

Denormalized = evenly spaced

same spacing
Try to represent 3.14, 6-bit example

6-bit IEEE-like format

<table>
<thead>
<tr>
<th>s</th>
<th>exp</th>
<th>frac</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Value: 3.14;
3.14 = 11.0010 0011 1101 0111 0000 1010 000...
= 1.1001 0001 1110 1011 1000 0101 000... \(2 \times 2^1\) (normalized form)

Significand:

\[ \frac{M}{10} = 1.100100011110101110000101000 \ldots \]

Exponent:

\[ E = 1 \quad \text{Bias} = 3 \quad \exp = 4 = 100_2 \]

Result:

\[ 010010 = 1.10_2 \times 2^1 = 3 \quad \text{next highest?} \]

Floating Point Arithmetic*

\[ V = (-1)^s \times M \times 2^E \]

\[
\begin{align*}
\text{double } x &= \ldots, \quad y = \ldots; \\
\text{double } z &= x + y;
\end{align*}
\]

1. Compute exact result.
2. Fix/Round, roughly:
   - Adjust \(M\) to fit in \([1.0, 2.0)\).
     - If \(M \geq 2.0\): shift \(M\) right, increment \(E\).
     - If \(M < 1.0\): shift \(M\) left by \(k\), decrement \(E\) by \(k\).
   - Overflow to infinity if \(E\) is too wide for \(\exp\).
   - Round \(M\) if too wide for \(\frac{M}{10}\).
   - Underflow if nearest representable value is 0.

Lessons for programmers

\[ V = (-1)^s \times M \times 2^E \]

\[
\begin{align*}
\text{float } \neq \text{real number } \neq \text{double} \\
\text{Rounding breaks associativity} \quad \text{and other properties.}
\end{align*}
\]

\[
\begin{align*}
\text{double } a &= \ldots, \quad b = \ldots; \\
\ldots \\
\times \text{if } (a == b) \ldots \\
\text{if } (\text{abs}(a - b) < \text{epsilon}) \ldots
\end{align*}
\]