Floating Point Representation

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.

https://cs.wellesley.edu/~cs240/s20/

Fractional Binary Numbers

Value | 5 and 3/4

2 and 7/8

47/64

Observations

Shift left =
Shift right =
Numbers of the form 0.111111...2 are...?

Limitations:

Exact representation possible when?

1/3 = 0.333333...10 = 0.

Fractional Binary Numbers

\[ \sum_{k=-j}^{i} b_k \cdot 2^k \]

Fixed-Point Representation

Implied binary point.

\[ b_7 b_6 b_5 b_4 b_3 \ldots b_2 b_1 b_0 \]

\[ b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 \ldots \]

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

IEEE = Institute of Electrical and Electronics Engineers

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 \( s \)
- \( \text{exp} \) field encodes \( E \) (but is \textit{not equal} to \( E \))
- \( \text{frac} \) field encodes \( M \) (but is \textit{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:
   - \( 0.0: \quad s = 0 \quad \text{exp} = 00...0 \quad \text{frac} = 00...0 \)
   - \( +\text{inf}, -\text{inf}: \quad \text{exp} = 11...1 \quad \text{frac} = 00...0 \)
   - Division by 0.0
   - NaN (“Not a Number”): \( \text{exp} = 11...1 \quad \text{frac} \neq 00...0 \)
   - \( \sqrt{-1}, -\infty - \infty, \infty \times 0, \text{etc.} \)

Value distribution

Precisions

Single precision (\texttt{float}): 32 bits

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

Double precision (\texttt{double}): 64 bits

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

Finite representation of infinite range...
Normalized values, with float example

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

**Value:** float \( f = 12345.0; \)
\[ 12345_{10} = 11000000111001_2 = 1.1000000111001 \times 2^{11} \text{ (normalized form)} \]

**Significand:**
\[ M = 1.1000000111001, \]
\[ \text{frac} = 1000000111001000000000000 \]

**Exponent:**
\[ E = \text{exp} - \text{Bias} \rightarrow \text{exp} = E + \text{Bias} \]
\[ E = 13 \]
\[ \text{Bias} = 127 \]
\[ \text{exp} = 140 = 10001100 \]

**Result:**
\[ 0 \ 10001100 \ 1000000111001000000000000 \]

---

Denormalized Values: near zero

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

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

**Significand:** leading zero
\[ M = 0. \text{xxx}...x_2 \]
\[ \text{frac} = \text{xxx}...x \]

**Cases:**
\[ \text{exp} = 000...0, \ \text{frac} = 000...0 \quad 0.0, \ -0.0 \]
\[ \text{exp} = 000...0, \ \text{frac} \neq 000...0 \]

---

Value distribution example

6-bit IEEE-like format

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

**Full Range**
\[ s=1, \ \text{exp}=101 \]
\[ E = 5-3 = 2 \]
\[ \text{frac} = 00, \ 01, \ 10, \ 11 \]
\[ M = 1.00, 1.01, 1.10, 1.11 \]

**Zoom in to 0**
\[ s=1, \ \text{exp}=010 \]
\[ E = 2-3 = -1 \]
\[ \ \text{frac} = \text{evenly spaced} \]
\[ \text{Denormalized} \]
\[ \text{same spacing} \]

**Try to represent 3.14, 6-bit example**

6-bit IEEE-like format

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

**Value:** 3.14;
\[ 3.14 = 11.0010 0011 1110 0000 1010 000... \]
\[ = 1.1001 0001 1110 1011 1000 0010 1000..._2 \times 2^{11} \text{ (normalized form)} \]

**Significand:**
\[ M = 1.100100011110101110101000000101000 ... \]
\[ \text{frac} = \text{100}_2 \]

**Exponent:**
\[ E = 1 + \text{exp} - \text{Bias} = 1 - \text{Bias} \]
\[ E = 1 \quad \text{Bias} = 3 \quad \text{exp} = 4 = 	ext{100}_2 \]

**Result:**
\[ 0 \ \text{100} \ 10 = \text{1.10}_2 \times 2^{1} = 3 \quad \text{next highest?} \]
Floating Point Arithmetic*

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

1. Compute exact result.
2. Fix/Round, roughly:
   - Adjust \( M \) to fit in \([1.0, 2.0)\).
   - If \( M = 2.0 \): 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 \( \text{exp} \).
   - Round* \( M \) if too wide for \( \text{frac} \).
   - Underflow if nearest representable value is 0.

Lessons for programmers

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

float ≠ real number ≠ double

Rounding breaks associativity and other properties.

double \( a = \ldots \), \( b = \ldots \);

...\[ \times \]

\( \text{if} \ (a == b) \) ...

...\[ \times \]

if (abs(a - b) < epsilon) ...

*complicated...