Explanations for Week 4 Floating Point Problems

`24125093 - Dinh Thien Loc Nguyen`

November 17, 2025

Purpose

Short, precise explanations of the floating-point functions in `bits.c`. Each section includes goal, method, hex notes, and concrete examples.

1 floatAbsVal

Goal: Bit-level absolute value of a float.

Method:

- Detect NaN: exponent = `0xFF`, fraction $\neq 0$.
- If NaN, return original bits.
- Otherwise clear sign bit: `uf & 0x7FFFFFFF`.

Hex Example:

- `0xBF800000` $(-1.0) \rightarrow$ `0x3F800000`
- `0x7FC12345` (NaN) $\rightarrow$ unchanged

2 floatFloat2Int

Goal: Convert float bits to integer, truncating toward zero.

Method:

- Extract sign, exponent, fraction.
- Compute $E = exp - 127$.

- Normal: mantissa $M = (1 << 23)|frac$.
- Shift M left/right by E.
- If $E < 0$: result is 0.
- If $E > 30$: overflow $\rightarrow$ `0x80000000`.

Examples:

- `0x41200000` (10.0) $\rightarrow$ 10
- `0xC1200000` (-10.0) $\rightarrow -10$
- `0x7F000000` (very large) $\rightarrow$ `0x80000000`

3 floatInt2Float

Goal: Return IEEE-754 single-precision encoding of `(float)x`.

Method:

- Handle $x = 0$.
- Determine sign and absolute value.
- Find most significant 1-bit to compute exponent.
- Construct fraction by shifting and apply round-to-even.

Examples:

- 5 $\rightarrow$ `0x40A00000`
- -8 $\rightarrow$ `0xC1000000`

4 floatIsEqual

Goal: IEEE-compliant float equality.

Method:

- If either is NaN, return false.
- If bitwise equal, return true.
- If both zero (mask sign), return true.

Examples:

- `0x00000000` vs `0x80000000` → true
- Any NaN comparison → false

5 floatIsLess

Goal: Return whether $f < g$ under IEEE rules.

Method:

- NaN → false.
- If both zeros, false.
- If signs differ: negative < positive.
- If same sign:
  - Positive: unsigned comparison matches numeric order.
  - Negative: reversed ordering.

Examples:

- $-1.0 < 1.0$ → true
- $+0 < -0$ → false

6 floatNegate

Goal: Compute $-f$, except NaN stays unchanged.

Method:

- If NaN, return original.
- Otherwise flip sign bit: `uf ^ 0x80000000`.

Examples:

- $1.0 \rightarrow -1.0$
- NaN stays NaN

7 floatPower2

Goal: Compute 2^x as a float.

Method:

- $x < -149$: return 0.
- $-149 \leq x < -126$: produce denormal.
- $-126 \leq x \leq 127$: normal exponent $= x + 127$.
- $x > 127$: return $+\infty$ (`0x7F800000`).

Examples:

- $2^0 \rightarrow$ `0x3F800000`
- $2^{-149} \rightarrow$ `0x00000001`
- $2^{128} \rightarrow$ `0x7F800000`

8 floatScale1d2

Goal: Compute $0.5 \cdot f$.

Method:

- NaN/INF: unchanged.
- Denormal: right-shift fraction (round-to-even).
- Normal with exp > 1: exp–.
- Normal with exp $= 1$: convert to denormal.

Examples:

- $2.0 \rightarrow 1.0$
- Smallest normal becomes denormal

9 floatScale2

Goal: Compute $2 \cdot f$.

Method:

- Denormal: left-shift fraction.

- Normal: exponent++.
- Overflow: produce INF.

Examples:

- $1.5 \to 3.0$
- Largest finite $\to$ INF

10 floatScale4

Goal: Compute $4 \cdot f$.

Method:

- Denormal: shift left twice; may normalize.
- Normal: exponent += 2.
- Check overflow to INF.

Examples:

- $1.0 \to 4.0$

11 floatScale64

Goal: Compute $64 \cdot f$.

Method:

- Denormal: shift left six bits; normalize if hidden bit appears.
- Normal: exponent += 6.
- Overflow exponent $\to$ INF.

Examples:

- $0.125 \to 8.0$

12 floatUnsigned2Float

Goal: Convert unsigned integer u to IEEE float bits.

Method:

- If $u = 0$, return 0.
- Find MSB index to compute exponent.
- Shift appropriately to form 23-bit fraction.
- Apply round-to-even when discarding bits.

Examples:

- $1 \rightarrow$ `0x3F800000`
- $4294967295 \rightarrow$ `0x4F000000`