Introduction to Computer Systems Recitation

——Floating Point

Guo Jiarui
1900012974
ntguojiarui@pku.edu.cn

Peking University

October 15, 2020

Fractional Binary Numbers

$$\begin{array}{cccccccccc} b_m & b_{m-1} & \cdots & b_1 & b_0 & . & b_{-1} & \cdots & b_{-n} \\ 2^m & 2^{m-1} & \cdots & 2^1 & 2^0 & & 2^{-1} & \cdots & 2^{-n} \end{array}$$

$$b = \sum_{i=-n}^{m} b_i \cdot 2^i$$

Note:

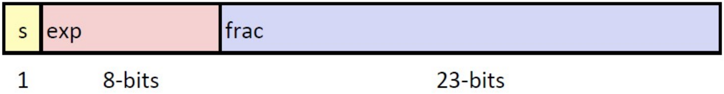
$$0.111\cdots1_2 \longleftrightarrow 1 - \varepsilon$$

IEEE Floating-Point Representation

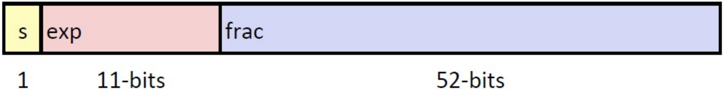
$$V = (-1)^s \times M \times 2^E$$

- s: determine whether the floating-point is positive/negative
- M: a fractional binary number ranges between 1 and $2-\varepsilon$ (or between 0 and $1-\varepsilon$)
- E: weigh the value by power of 2

Single precision: 32 bits



Double precision: 64 bits



Three different cases:

1. exp$\neq 0$ && exp $\neq 11\cdots1_2$. In this case:
   - $E = e - Bias$, here $Bias = 2^{k-1} - 1$.
   - $M = 1 + f$, here $1 \leq M < 2$.
2. exp=0. In this case: $E = 1 - Bias, M = f$.

   Reason for why $E = 1 - Bias$ rather than $E = -Bias$:
   When $exp = 0$ and $frac = 11\cdots1_2$, $V = (1-\varepsilon) \times 2^{1-Bias}$.
   When $exp = 1$ and $frac = 0, V = 1 \times 2^{1-Bias}$.
3. exp=$11\cdots1_2$.In this case:
   - $f = 0$, it represents infinity.
   - $f \neq 0$, it represents NaN.

Rounding

Four modes of rounding:

- round-to-even, or round-to-nearest (default)
- round-toward-zero
- round-down
- round-up

Rounding

$$1.\mathrm{BBGRXX}\cdots \quad (\text{Assume } k = 4, n = 3)$$

- G: LSB(Last Saved Bit) of the result.
- R: First Removed Bit.
- S: Sticky Bit, OR of remaining bits.

Round-up Conditions:

- R=0: < 0.5. Remove.
- R=1, S=0: $= 0.5$. Round to even.
  - G=0: Remove.
  - G=1: Increase.
- R=1, S=1: > 0.5. Increase.

Example: (Assume $k = 4, n = 3$)

Value	Fraction	GRS	Increase?	Rounded
128	1.000 0000	000	N	1.000
15	1.101 0000	100	N	1.101
17	1.000 1000	010	N	1.000
19	1.001 1000	110	Y	1.010
138	1.000 1010	011	Y	1.001
63	1.111 1100	111	Y	10.000

Converting an Integer into IEEE Floating Point Standard

Example: Convert 1245 into Floating Point Standard:

1. Determine the sign s:

   Here, $s = 0$ because $1245 > 0$.

2. Change the integer into binary form:

   Here, $1245 = 10011011101_2$.

3. Left shift the decimal point to get a normalize form:

   Here, $1245 = 1.0011011101_2 \times 2^{10}$.

4. Abandon the beginning 1 and add 0 at the end of the decimal point (if necessary) or round the number (if necessary) to get M.

   Here, $frac = [00110111010000000000000]$.

5. Add $Bias$ to get E.

   Here, $E = 10 + 127 = 137$ and $exp = [10001001]$.

6. Write the floating point:

   Here, we get [0100 0100 1001 1011 1010 0000 0000 0000].
   Converting into hexadecimal form, we get 0x449BA000.

Reference:`http://www.styb.cn/cms/ieee_754.php`

Floating-Point Operations — Multiplication

$$(-1)^{s_1} \cdot M_1 \cdot 2^{E_1} \times (-1)^{s_2} \cdot M_2 \cdot 2^{E_2} = (-1)^{s} \cdot M \cdot 2^{E}$$

- Exact Result:
  - $s = s_1 \text{ \^{} } s_2$.
  - $M = M_1 \times M_2$.
  - $E = E_1 + E_2$.
- Fixing:
  - if $M \geq 2$:
    $M >>= 1$; $E++$;
  - if E out of range: return infinity;
  - round M to fit the precision.

Floating-Point Operations — Multiplication

Properties:

1. closed under multiplication:
   may generate infinity or NaN.
2. commutative:
   $a * b = b * a$ (even if we can get infinity).
3. not associative:
   $(1e20 * 1e20) * 1e\text{-}20 = \infty$; $1e20 * (1e20 * 1e\text{-}20) = 1e20$.
4. not distribute over addition:
   $1e20 * (1e20 - 1e\text{-}20) = 0$; $1e20 * 1e20 - 1e20 * 1e20 = NaN$
5. multiplicative identity: 1
6. monotonicity:
   $a \geq b, c \geq 0 \Rightarrow a * c \geq b * c$ (except infinity and NaN).
7. positive definite:
   for $a \neq$NaN, $a * a \geq 0$.

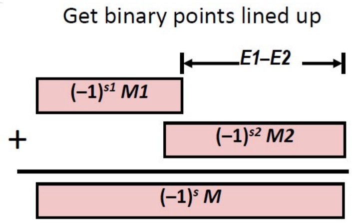
   Question: The maximum a for $a * a = 0$?

Floating-Point Operations — Addition

$$(-1)^{s_1} \cdot M_1 \cdot 2^{E_1} + (-1)^{s_2} \cdot M_2 \cdot 2^{E_2} = (-1)^s \cdot M \cdot 2^E$$

(We assume that $E_1 > E_2$)

- Exact Result: $s = s_1, E = E_1$.



- Fixing:
  - if $M \geq 2$:
    $M >>= 1;\ E++;$
  - if $M < 1$:
    $M <<= k;\ E -= k;$
  - if E out of range: return infinity;
  - round M to fit the precision.

Floating-Point Operations — Addition

Properties:

1. closed under addition:
   may generate infinity or NaN.
2. commutative:
   $a + b = b + a$ (even if we can get infinity).
3. not associative:
   $(3.14 + 1\text{e}20) - 1\text{e}20 = 0; 3.14 + (1\text{e}20 - 1\text{e}20) = 3.14.$
4. additive identity: 0
5. monotonicity:
   $a \geq b \Rightarrow a + c \geq b + c$ (except infinity and NaN).

   Special: About NaN and inf:
   (-inf) + inf = NaN;
   inf == inf;
   NaN != NaN;
   Expression if(NaN) will return 1.
6. additive inverse: except infinity and NaN.

Floating Point in C

About casting values between int, float and double:

- int -> float: not overflow, possibly rounded

  Question: The smallest positive integer that cannot be represented exactly for single-precision format?

- int, float -> double: precise (because double has greater range and greater precision)
- double -> float: possibly overflow & rounded
- float, double -> int: round-to-zero, possibly overflow

Additional Slides

Homework 2.84
Fill in the return value for the following procedure which tests whether its its first argument is less than or equal to its second.

One possible answer:

```
(sx && !sy) || (!sx && !sy && ux <= uy) || (sx && sy && ux >= uy)
```

Additional Slides

Comparisons of binary representations between int and float: Homework 2.6, 2.48

decimal	hexadecimal	binary
3510593	0x00359141	[00000000001101011001000101000001]
3510593.0	0x4A564504	[01001010010101100100010100000100]

Homework 2.89

A	True
B	Possibly overflow
C	True
D	Precision can be different
E	dx (or dz) can be 0