浮点数表示

2023 年 3 月 1 日

目录

我们所讲的浮点数大致分为三个部分讲解, 第一个部分为小数的定点表示, 第二部分为小数的浮点表示, 第三个部分为规格化表示, 第四个部分是 IEEE 754 标准.

1 小数的定点表示

我们之前已经学习了整数的表示方法, 这里我们介绍一种并不是很实用的, 表示小数的一个方法, 称为是小数的定点表示. 这里我们可以先复习一下原码, 补码和移码.

1.1 原码, 补码, 移码

我们接下来表示的数字都是默认有符号的, 对于原码来说, 需要提供一位 bit 出来当作是符号位. 这个表示方法的特点便是, 0 有两种表示方式, 因为 $+0$ 和 -0 都是一样, 并且此等表示方式, 能够表出的正数和负数的个数是一致的. 这个是其显著特征. 那么说, 对于一个 n 位的原码表示, 其表示范围为 $2^{n-1}-1$ – $-(2^{n-1}-1)$. 那么我们接下来讲补码. 补码的定义为: 将所有的 bit 置反, 随后 $+1$. 我们可能会觉得这个是什么几把, 为什么要 $+1$? $+1$ 的作用其实仅仅是对齐而已, 比如说 `00000000` 取反之后为 `11111111`, 我总不能用这个数字表示 0 吧, 于是就设置了一个偏移量, $+1$ 使得这个数字变为了 `00000000`, 就是零了. 而后我们需要记住. -0 的原码表示被映射为了 -2^n, 这属于是规定.

随后, 我们给出补码的表示公式, 记补码长这个样子: $b_{n-1}b_{n-2}\dots b_2b_1b_0$, 记真值为 N:

$$N=\begin{cases} b_{n-2}b_{n-3}\cdots b_0 & , \text{ if } b_{n-1}=0 \\ 2^{n-1}-b_{n-2}b_{n-3}\cdots b_0 & , \text{ if } b_{n-1}=1 \end{cases} \tag{1}$$

1.2 定点表示

小数的定点表示可以表示出一个绝对值小于 1 的小数. 其中有一位符号位, 比如说我们有一个四位的定点表示, 我们可以记为: $b_0.b_1b_2b_3$ 注意这里的小数点.

对于定点表示, 也分为补码和原码两种. 对于原码, `0.111` 就是最大正数, 为 $1-2^{-3}$, 一般化表示即为 $1-2^{-(n-1)}$; 最小负数为 `1.111`, 为 $1(1-2^{-3})$. 剩下类似. 对于补码, 最大正数和原码一致, 但是最小负数为 `1.000`, 剩下类似.

2 小数的浮点表示

浮点表示类似于科学计数法, 由两个部分组成: 1. 指数部分; 2. 尾数部分. 注意到, 尾数部分是定点表示的, 指数部分则不是. 如下:

$$j_fj_1j_2\dots j_mS_fS_1S_2\dots S_n \tag{2}$$

其中 j_f, S_f 都是用来表示符号的, 可以说这个操作没什么必要. 这个时候我们就可以开始分析, 在原码表示和补码表示之下, 表示范围分别是多少. 我们只考虑补码:

1 指数部分—也称阶码—的最大值为: 2^m-1, 最小值为 -2^m

2 尾数部分, 最大正数为 $1-2^n$, 最小正值为 2^n, 对应表示分别为 `0.111`, `0.001` (以四位表示为例, $n=3$)

3 尾数部分, 最大负数为 -2^n, 最小负数为 -1, 对应表示分别为 `1.111`, `1.000`.

4 于是表示范围为 $[-2^{2^m-1}\times 1, 2^{2^m-1}\times(1-2^n)]$

2.1 上溢和下溢

准确地来说, 我们浮点数的表示区间为 [最小负数,最大负数] $\cup \{0\} \cup$ [最小正数,最大正数]. 如果说绝对值太大了, 超过了表示范围, 这样的情况称为是上溢出; 类似地, 绝对值太小, 浮点数的精度无法表示, 这样的情况称为下溢.

3 规格化表示

对于不同的底数, 规格化有些许不同, 但是总体差不多, 我们只介绍底数为 2 的规格化. 我们首先介绍原码的, 原码和补码这两种情况还有区分.

3.1 原码的规格化

对于原码, 尾数必须为 `0.1XX…XX` 的这种形式. 当尾数并不满足这种形式的时候, 需要进行移位, 移位分为左规和右规:

1 左规: 尾数算数左移 (后面补上 0), 阶码减一

2 右规: 当浮点数加法之中出现了溢出, 将尾数右移一位, 阶码加一

Example 3.1. *左规:*

$$b = 2^1 \times (+0.01001) \implies b = 2^0 \times (+0.10010)$$

右规: $a = 2^2 \times (00.1100), b = 2^2 \times 00.1000$

$$\begin{aligned} a + b &= 2^2 \times (00.1100 + 00.1000) \\ &= 2^2 \times 01.0100 \\ &= 2^3 \times 00.1010 = 2^3 \times 0.1010 \end{aligned}$$

注意到, 使用两个 bit 表示符号是为了方便.

3.2 补码的规格化

补码的规定如下:

1 对于负数, 其补码形式应为 `1.0XXX…`

2 对于正数, 其补码形式应为 `0.1XXX…`

额, 等一会, 这里还有左规和右规吗? 什么够吧.

随后, 我们不管那么多, 我们想要知道这种情况之下, 规格化数的表示范围. 我们分为正数和负数两个部分讨论. 只考察尾数部分. 最大正数为 $1-2^{-n}$, 对应编码为 `0.1111`; 最小正数为 $1/2$, 对应编码为 `0.100`. 至于负数, 我们先考虑其原码, 再看转换的补码是否符合要求. 最小负数是比较好找的, -1 , 对应编码的补码编码为 `1.000`; 最大负数则为 $-(1/2+2^{-n})$, 我们来看看为什么.

考虑原码为 `1.100` 的负数, 其值为 $-1/2$, 绝对值刚好等于最小正数. 可是其补码却不是规格化数: `1.100`—补码是其本身. `1.100` 的负数, 小数点后取反的话, 便是 `1.011` 如果将其看作是补码, 那么这确实是最大负数, 可是反码到补码还需要加一, 这就使得, 我们原码也要加一. 所以说, 最大负数为 1.011 对应的值, 也就是 $-(1/2+2^{n})$.

上面的解释是说为什么最大负数不是 $-1/2$, 我们是容易知道 $-(1/2+2^{-n})$ 为最大负数的.

3.3 规格化的目的

3.4 浮点数溢出和机器 0

4 IEEE 754 标准

IEEE 754 浮点数表示标准是由 IEEE 电气电子工程师学会规定出的, 双精度或者单精度浮点数表示的规范. 其实我们可以只学习这个东西, 而不用管前面的那些够吧.

4.1 构成

对于单精度浮点, 32-bit, 构成如下

$$s\ \overbrace{e_7\ldots e_0}^{\text{8-bit}}\ \underbrace{w_{22}\ldots w_0}_{\text{23-bit}} \tag{3}$$

s 为符号位, 而 e_i 为无符号指数, w_i 为无符号尾数. 说实话和我们上面学的几把东西没什么关联. 随后我们需要注意到, 虽然说 e_i 是无符号数, 但是, 我们之后会让其减去一个 bias 让其能够表示负数. bias 在不同情况下是不同的, 所以我们现在不能武断地说 e_i 是移码表示的, 我们后面将会看到其和移码表示差不多, 仅仅在一小点地方有差别.

尾数的表示也分情况. IEEE 为了让其最大化表示精度, 设定了一些规定. 我们需要知道: 分为规格化数 (浮点数) 和非规格化数进行讨论.

4.2 尾数

IEEE 规定, 在规格化数表示的时候, 尾数默认省略了一个前置的 1, 就是说真实的尾数是 $1 + W$[1].

在非规格化数表示的时候, 就默认没有前置的 1, 真实的尾数就是 W. 原因也很简单, 当我们表示的数字特别小的时候, 小于了 $1 \times 2^{\text{最小指数}}$ 的时候, 我们别无选择, 只能够消去前面这个默认的 1.

4.3 指数

指数是有 bias 的, 这使得, E 为 0 的时候, 其表示的数是最小的, 是最小负数. 在规格化数那里, 这个 bias 是 -127[2], 考虑规格化数, 真实指数的最小值—$1 - 127 = -126$. 若是再继续往下, 要让表示的数字更小, 就会走到非规格化数, 可是由于非规格化数的没有前置 1, 所以说, 真实指数的值不

[1] 设大写字母代表其真值

[2] 但同时, E 的取值范围为 $[1, 254]$. $0, 255$ 都用来表述特殊值, 其中 $E = 0$ 的时候, 说明表示的是非规格化数.

应该发生变化, 否则表述的区间就不再连续, 故真实指数为 -126, 所以这个 bias 变为了 -126, 因为 $0 - 126 = -126$.

4.4 特殊值

$E = 255$ 的时候, 浮点数表示特殊值.

当 $E = 255, W = 0$ 时候 浮点数表示 ∞.

当 $E = 255, W$ 为非零值的时候 浮点数表示 NaN (Not a Number).