|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| 符号位1 | 指数部分8 | | | | | | | | 尾数部分23 | | | | | | | | | | | | | | | | | | | | | | |

最左边决定了整个浮点数的符号，0为正，1为负，我们现在不管他

指数部分一共8位，可以表达256种数字，注意了，这8位储存的并不是真正的指数数值，而是

储存的值 = 真正的指数+127

也就是我们想根据指数域保存的值去得到真实的指数的时候必须减去127，例如如果保存的数值是0100 0000，也就是64，那么真正的指数就是64（储存的值）-127 = -63，如果我们的数字的真正的指数是3，那么我们要保存的数值是3+127 = 130，也就是1000 0010，懂了不。这样子我们可以用指数域表示的范围就是-127~128了，为什么是-127？因为本来都说了八位可以表达的数字有256种，二进制形式表示的就是0~255这256个数字了，所以（0-127）~（255-127）就是-127~128了呀，所以暂且你可以理解为我们能表示的指数范围就是-127到2^128这么大，实际并不是这样，后面会解释

接下来我们讲尾数域的情况

尾数有23位，按道理来说我们可以表达的最大数字（现在你别管什么正负号）是2的23方-1，因为IEEE规定了所有浮点数小数点左侧都是1，然后他们就想到了一些办法来节省这些位了，他们干脆不储存数字的左侧那一位1了，例如1.010101，他们干脆不保留左侧那个1只保存010101这些数字，然后右边补上0填满那23位，所以可以表示的数字最大不是2的23次方-1而是2的24次方-1，也就是16,777,215，不过因为IEEE又规定了一些特别数字，例如16,777,216，这个数可以通过一些特殊办法表示，但是16,777,217就不行，你不用管有多少个数字了，现在会计算浮点数就行了

好了，我们解释下上面的指数表示范围实际IEEE规定了用-127和128这两个数作为特殊值处理，也就是说我们能使用的常规的表示最大指数范围分别为-126，127了，-127（指数域存放全为0）和128（指数域存放全为1）这个都是作为特殊的一类数处理的，例如某个数以的指数位全为0那么无论这个数字的尾数部分为什么，那么他都是特殊值了，特殊值其实就是不可以正常使用的数了，例如什么正无穷啊，NaN啊

另外，如果规定了小数点左侧必为1，那么我要表示十进制的0怎么办，IEEE没办法了，只能又规定了一个特殊值，IEEE规定0这个数字保存为尾数域为全为 0，指数域为-127，也就是说指数域也全为 0。考虑到符号域的作用，所以存在着两个零，即 +0 和 -0。并且+0和-0相等

关于特殊值，IEEE定义了一下特殊值

NaN 指数为128（指数域全为 1），且尾数域不等于零的浮点数 NaN！=NaN

±0零保存为尾数域为全为 0，指数域为-127，也就是说指数域也全为 0，符号位1时候为-0，-0==+0

±∞指数部分同样为128(全为1)，不过无穷的尾数域必须为零，符号域为1时候为负无穷

以及非规范化数

当浮点数的指数为允许的最小指数值，即 -126 时，尾数不必必须是规范化的。。注意，这里规定的是"不必"，这也就意味着"可以"。也就是说，保存时隐含着一个隐藏的尾数位。为了保存非规范浮点数，IEEE 标准采用了类似处理特殊值零时所采用的办法当指数域为-126时（但尾数域不能全0）。这样可以那个隐藏的24位上的那个1替换一个0表示很小的数