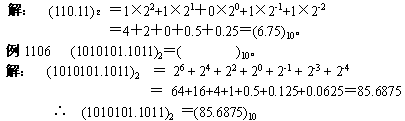


<https://www.cnblogs.com/xkfz007/articles/2590472.html>

**一、二进制数转换成十进制数**  
    由二进制数转换成十进制数的基本做法是，把二进制数首先写成加权系数展开式，然后按十进制加法规则求和。这种做法称为"按权相加"法。

**例1105** 把二进制数110.11转换成十进制数。

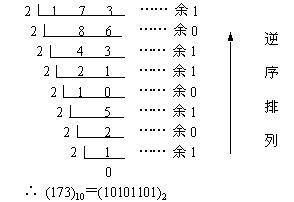


**二、十进制数转换为二进制数**

1. **十进制整数转换为二进制整数**

"除2取余，逆序排列"法。

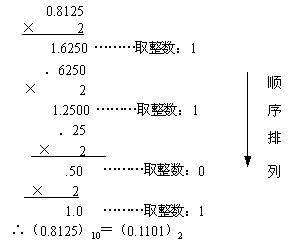
**例1107** 把 (173)10 转换为二进制数。  
    **解：**



**2．十进制小数转换为二进制小数**

"乘2取整，顺序排列"法。

**【例1108】**把（0.8125）转换为二进制小数。  
**解：**



**例1109** （173.8125）10＝（ ）2  
**解：** 由［例1107］得（173）10＝（10101101）2  
由［例1108］得（0.8125）10＝（0.1101）2  
把整数部分和小数部分合并得： （173.8125）10＝（10101101.1101）2

**证明：**

1、将二进制数.1101转换成十进制：

 （1）

2、现在要从十进制小数转成二进制小数：

对(1)式两边乘上2，

 (2)

可以看到，乘上2之后，得到1.6250的整数部分1，是最终结果的小数部分的第一位，1.6250的小数部分0.6250正好是(1)左边括号的和，即对应其他小数位；

取(2)式的小数部分，得到(3)式



 (3)

对(3)式两边再次乘上2，与上面类似，不再重复叙述。

总结：十进制小数转二进制小数， **“不断对小数部分乘2取整，顺序排列”**。

**参考：《计算机系统组成与体系结构》**

<https://blog.csdn.net/whyel/article/details/81067989>

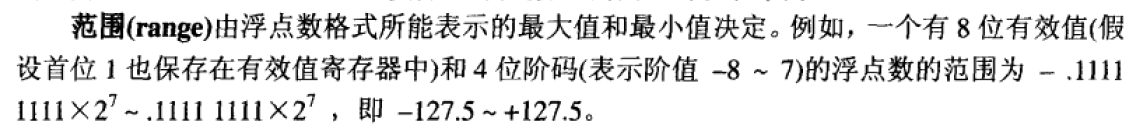
**浮点数的计算机表示法：**

**移码：**阶码用移码表示。本来阶码可以用补码表示，例如4位阶码，可以表示-8~7的补码。为了让二进制的最小值0000对应-8, 二进制最大值1111对应7，对补码加上一个偏移量8。这种0000表示-8，1111表示7的表示法就是移码。

**就近舍入法：**即“四舍五入”。例如，尾数超出规定的23位的多余位数字是10010,多余位的值超过规定的最低有效位值的一半，故最低有效位应增1。若多余的5位　是01111,则简单的截尾即可。对多余的5位10000这种特殊情况：若最低有效位现为0,则截尾；若最低有效位现为1,则向上进一位使其变为 0。

**浮点数范围：**

<https://blog.csdn.net/qq_16137569/article/details/79508091> 建议查看float.h文件.

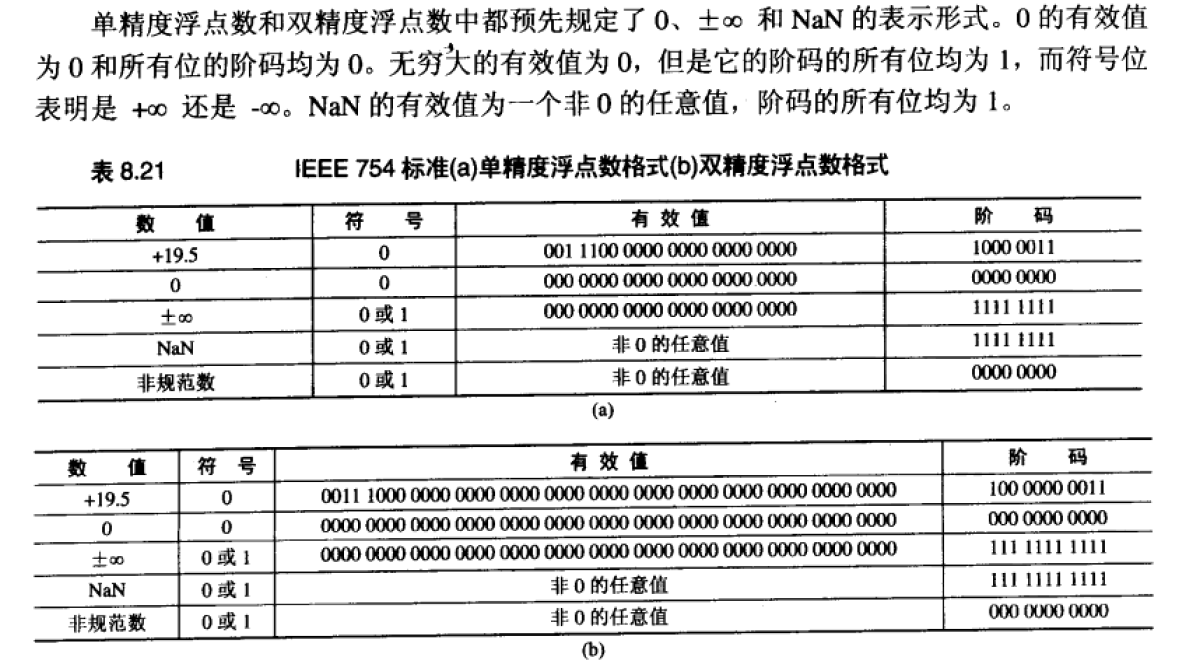


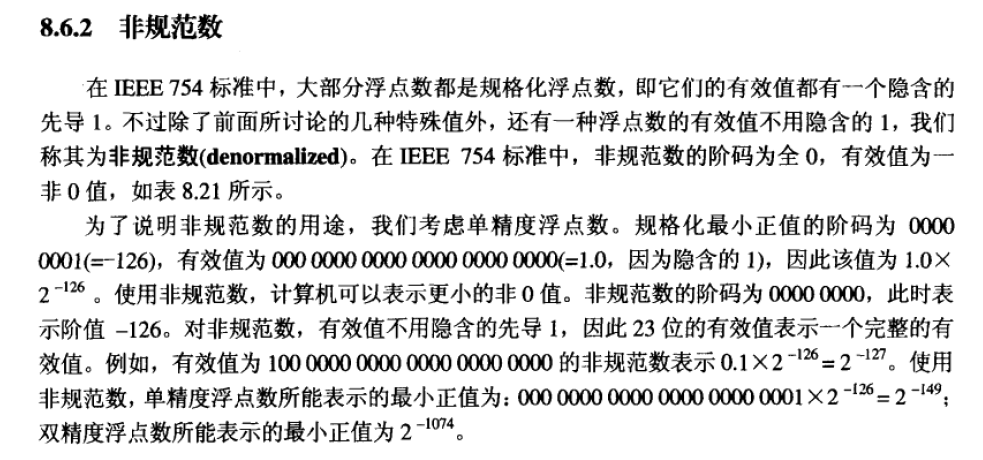
注：****

**IEEE 754标准：**

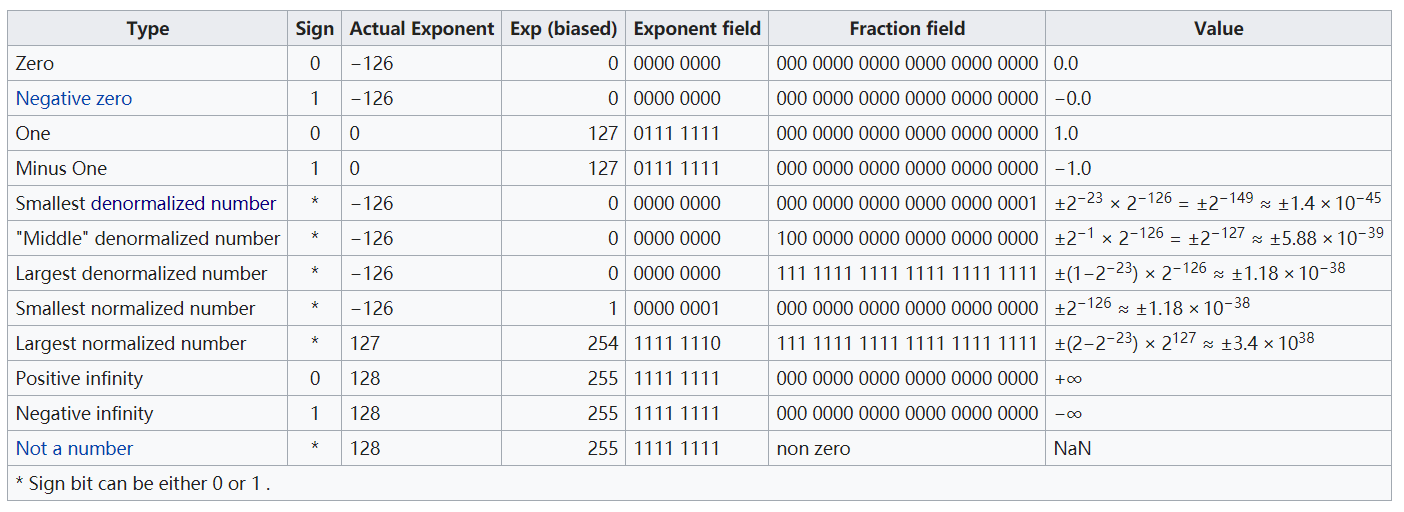
float: 32位，1位符号位，8位阶码（偏移量127），23位有效值（包含一个隐含的1在小数点的左边）。规格化有效值在1有效值<2之间。阶值在-126到+127之间，因为阶值0000 0000(=-127)和1111 1111(=+128)用来表示特殊值。

double: 64位，1位符号位，11位阶码，52位有效值（有隐含的先导1）。阶码偏移量1023，阶值在-1022到+1023之间，阶值-1023和+1024用来表示特殊值。





<https://en.wikipedia.org/wiki/IEEE_754-1985#Examples>



**总结：非规范数不含前导1，阶码全为0，有效值非0任意值。**

**例1：**19.5的float表示：



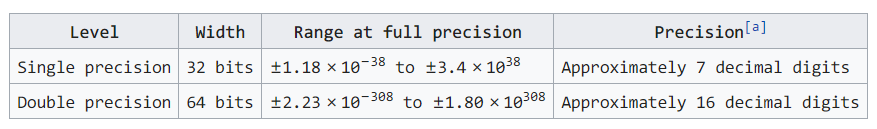
符号位：0

有效位：00111(最高位1是隐含的)

阶码：4+127=1000 0011

**例2：**float的数值范围为 （参考 <https://www.jianshu.com/p/43b1b09f27f4> ）

**Wiki:**



可见float类型, 或者说32位浮点数的取值范围是**±1.18×10−38 to ±3.4×1038**, 准确的说是**[-3.4\*10^38, -1.18\*10^-38] ∪ [1.18\*10^-38, 3.4 \* 10^38]，再加上特殊数：0，NaN，无穷，非规范数。**

**float.h:**

#define FLT\_MAX 3.402823466e+38F // max value

#define FLT\_MIN 1.175494351e-38F // min normalized positive value

**证明：**因为阶值范围是[-126,127],





这里的指的是规格化数，float可以表示特殊数0，以及更小的非规范数，用非规范数，最小的正数为2^-23\*2^-126=2^-149=1.4E-45.

**float的精度问题：**

由于float只有23bit有效值，精确表示范围为, 对应的十进制大小为 ，这个也是舍入之后的值，当然可以再乘上指数（-126~127），当超过这个范围，数值就不再精确。