细分浮点运算，主要有八个步骤：判断是否有零操作数-原码转补码-阶数对齐-尾数求和-规格化处理-舍入-溢出判断-补码转原码

（这里的溢出判断是对于阶数的判断，和规格化处理中的尾数溢出有所不同）

关于规格化处理，有一个关键点：是否进行尾数左移或右移，补充‘0‘还是‘1’。

解决这个问题，可以分为两个部分：什么时候需要移动？如何移动？

1. 在图中可以看到，**规格化数的补码，符号位和第一数位是不同的**。
2. 在尾数求和后，有两种情况我们需要进行规格化处理：**符号位和第一数位不同，双符号位两位不同（有正溢出或负溢出）**。这两种情况对应两个操作：左移和右移。当绝对值较小（符号位和第一数位不同）时，需要左移（等于 x2）；当绝对值较大（溢出）时，需要右移（等于/2）。一句话，**移动尾数是为了让符号位与第一数位不同**。
3. 左移时，在最低位补 0；右移时，尾数结合双符号位，共同向右移 1 位。
4. 规格化处理后，一般后续需结合舍入方法进行舍入。

Example（尾数求和后的结果）：

00 100100（符合规格化要求,符号位为 0，第一数位为 1）->0 100100

01 100101（正溢出，右移）-> 0 1100101（多一位，舍入）->0 110010（截断法）

10 000001（负溢出，右移）-> 1 0000001（多一位，舍入）->1 000001（0 舍 1 入法）

11 110110（符号位和第一数位相同，左移）-> 1 101100 -> 1 011000

课本例 10.13（大家可以结合这个方法再试一下）

实际上结合双符号位整体右移 1 位就是补码的算术右移 1 位

比如课本例 10.13 将 Y 对阶

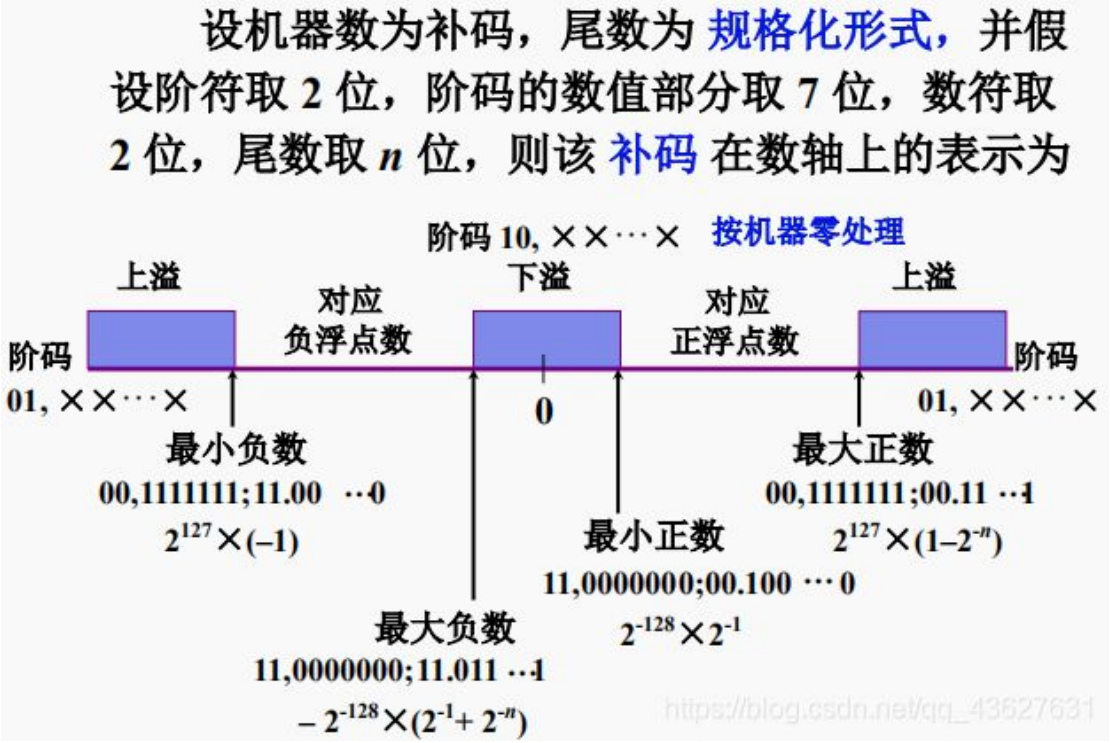
Y: 1 00010 000110（对阶，右移一位，阶码加 1）用同样的方法（结合双符号位整体右移）

11 000110 -> 1 100011（注意，此时不是规格化数）

Y(对阶后): 1 00011 100011

题外话:

溢出判断是指阶码的前两位符号位是否一致，若不一致，则有溢出



参考:

https://blog.csdn.net/qq_43627631/article/details/107109559