**浮点加减运算**

对任意一个二进制数N，总可以表示成：N=2E × M ，式中，E为数N的阶码，M称为数N的尾数，一般为绝对值小于1的规格化数（补码是允许为-1）。

两浮点数X，Y进行加减运算时，必须按以下几步执行：

①对阶，使两数的小数点位置对齐，小的阶码向大的阶码看齐。  
　　②尾数求和，将对阶后的两尾数按定点加减运算规则求和(差)。  
　　③规格化，为增加有效数字的位数，提高运算精度，必须将求和(差)后的尾数规格化。  
　　④舍入，为提高精度，要考虑尾数右移时丢失的数值位。  
　　⑤判断结果，即判断结果是否溢出。

规格化又分左规和右规两种

1. 左规。当尾数出现00.0××…×或11.1××…×时，需左规。左规时尾数左移一位，阶码减1，直到符合补码规格化表示式为止
2. 右规。当尾数出现01.××…×或10.××…×时，表示尾数溢出，这在定点加减运算中是不允许的，但在浮点运算中这不算溢出，可通过右规处理。右规时尾数右移一位，阶码加1.

例，两浮点数x＝2+010 ×0.110100，y=2+100 ×（-0.101010），求x+y。  
 解：阶码取3位，尾数取6位（均不包括符号位），机器表示的形式分别为

[x]补 = 0010 0110100

[y]补 = 0100 1010110

①对阶：先求阶差（两阶码的补码相减）

00 010

+ 11 100 （减 00 100 就是加 —00100的补码，即11 100）

11 110 其真值为-2，即x的阶码比y的阶码小2

[x]补 的阶码增大成 0100，尾数右移两位，即[x]补 = 0100 0001101

②尾数相加

00.001101

+ 11.010110

11.100011 相加结果为0100 1 100011

③规格化：

最高有效位与符号位相同，需要左规，所以结果应为：

[x+y]补 = 0011 1 000110

x+y = 2+011 ×（-0.111010）

　4．舍入

　　在对阶和右规的过程中，可能会将尾数的低位丢失，引起误差，影响了精度，为此可用舍入法来提高尾数的精度。常用的舍入方法有三种。

　　（1）截去法。将多余的位截去，剩下的位不变。其最大误差接近于数据最低位上的1。

　　特点：有舍无入，具有误差积累。

　　（2）“0舍1入”法。“0舍1入”法类似于十进制运算中的“四舍五入”法，即在尾数右移时，被移去的最高数值位为0，则舍去；被移去的最高数值位为1，则在尾数的末位加1。这样做可能使尾数又溢出，此时需再做一次右规。  
　　其最大误差是最低位上的-1/2到接近于1/2之间，正误差可以和负误差抵消。是比较理想的方法，但实现起来比较复杂。

　　（3）“恒置1”法。尾数右移时，不论丢掉的最高数值位是“1”或“0”，都使右移后的尾数末位恒置“1”。这种方法同样有使尾数变大和变小的两种可能。  
　　特点：尽管误差范围扩大了，但正负误差可以相互抵消，从统计角度，平均误差为0。因此最后运算结果的准确性提高了。  
　　综上所述，浮点加减运算经过对阶、尾数求和、规格化和舍入等步骤。与定点加减运算相比，显然要复杂得多。

例，两浮点数x＝2+10 ×0.1101，y=2+01 ×0.1011，求x+y，舍入用0舍1入法。  
 解：阶码取3位，尾数取6位（均不包括符号位），机器表示的形式分别为

[x]补 = 010 0 1101

[y]补 = 001 0 1011

①对阶：先求阶差（两阶码的补码相减）

00 10

+ 11 11 （减 00 01 就是加 —00 01的补码，即11 11）

00 01 其真值为1，即x的阶码比y的阶码大1

[y]补 的阶码增大成 10，尾数右移一位，即0 01011

由0舍1入知，此时[y]补 = 010 0 0110

②尾数相加

00.1101

+ 00.0110

01.0011

③规格化：

因尾数符号位为01，需要右规（尾数右移1位，阶码加1），所以结果应为：

[x+y]补 = 011 0 10011

由0舍1入知： x+y = 2+11 ×0.1010

{截去法

①由截去法知，此时[y]补 = 010 0 0101

②尾数相加

00.1101

+ 00.0101

01.0010

③规格化：

因尾数符号位为01，需要右规（尾数右移1位，阶码加1），所以结果应为：

[x+y]补 = 011 0 10010

由截去法知： x+y = 2+11 ×0.1001

}

由此可知，采用不同的舍入方法得到的结果可能不同，最后所导致的误差也不同。