# 原码一位乘法

1. **用原码一位乘法计算X=0.1011，Y=-0.1101的乘积.**

【X】原=0.1101 【Y】原=0.1011  
部分积 乘数单元  
00.0000 1011  
+00.1101  
00.1101  
->00.0110 1101  
+00.1101  
01.0011  
->00.1001 1110  
+ 00.0000  
00.1001  
->00.0100 1111  
+00.1101  
01.0001  
->00.1000 1111  
因为X和Y异号，异号相乘为负所以  
X·Y=1.10001111

1. ****已知二进制数x= 0.1010,y= -0.0110,用原码一位乘法计算[x\*y]原,并还原成真值.****
2. **乘积的符号位：Xo♁Yo=0♁1=1**
3. **数值部分按绝对值相乘（移位相乘）：X\*·Y\*=0.01111000.部分积分别为：  
   0.0000（初态）  
   0.0000  
   0.1010  
   0.1111（最后结果,仍需逻辑右移）**
4. **[X·Y]原=1.01111000,真值为：-0.01111000B**
5. ****用原码一位乘法计算X\*Y，X=11011,Y= -11111****

**换成十进制,X=16+8+0+2+1=27  
 Y=-（16+8+4+2+1）=31  
 故X\*Y=-837  
 换成二进制**

**X\*Y=1100000101**

1. ****已知二进制数x＝－0.1011,y＝0.1101,用原码一位乘法求x\*y的值,要求写出计算机中的运算步骤.****

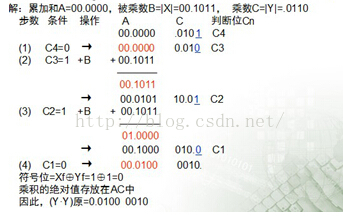
**0.1011  
 0.1101  
 ------------  
 0.00001011  
 + 0.0000000  
 + 0.001011  
 + 0.01011  
 ------------------  
 0.10001111**

**x\*y= 0.10001111**

1. ****, X =0.1101 ， Y = -0.1011,原码一位乘法计算   X \* Y****



1. ****X = -0.1011, Y = -0.0110 求（X \* Y）的原码****



**定点加减法运算**

## x = +1011, y = -0101,  求 x + y。

  解 [ x ]补  = 01011, [ y ]补 = 11011

                                    [ x ]补          01011

                           +      [ y ]补           11011

                                       ——————————

                                         [ x + y ]补        ****1****00110

                         注意[ x + y ]补  得出的结果里最左边的1，这个1在(mod 2^(n+1))的条件下是会被省略的，所以此题的结果为 ： [ x + y ]补   = 00110，然后符号位为零，故结果为正数，所以  x + y = + 0110 。

1. x = +1101, y = +0110,  求 x - y。

解 [ x ]补  = 01101, [ y ]补 = 00110， [ -y ]补 = 11010

                                          [ x ]补          01101

                                   +    [ - y ]补           11010

                                       ——————————

                                             [ x - y ]补        ****1****00111

                           故： x - y = + 0111

1. **设x=+100010b，y=+10110b，求x+y=?**

[x+y]补=[x]补+[y]补 [x]原=00100010b [y]原=00010110b

[x]补=00100010b （+34）

+ [y]补=00010110b （+22）

[x +y]补=00111000b （+56）

∵[x +y]补=00111000b的符号位为0，结果为正

∴[x +y]原=[x+y]补=00111000b ∴x +y= + 111000b

1. **设x=-0110001b，y=+0001001b 求x+y**

解：[x+y]补=[x]补+[y]补 [x]原=10110001b [y]原=00001001b

[x]补=11001111b （-49）

+ [y]补= 00001001b （+9）

[x +y]补=11011000b （-40）

∵[x +y]补=11011000b的符号位为1，结果为负数

∴[x +y]原=[[x+y]补]补=10101000b ∴x +y=-101000b

1. **x=0.1001，y=-0.0011，用补码的减法求x-y**

　解：[x]补=0.1001，[y]补=1.1101,[-y]补=0.0011

　　[x]补-[y]补=[x]补+[-y]补=0.1001+0.0011=0.1100

x-y=0.1100

1. **设x=+1011， y=+1001，求[x+y]补。**

　解：[x]补=01011， [y]补=01001

　　   [x+y]补=01011+01001=10100

　　两个正数相加，最高两位的进位为01，表示发生了溢出，其结果为负数，显然是错误的。

1. **设机器字长为8位（含1位符号位），A=15，B=24，求[A+B]补和[A-B]补并还原成真值。**

解：A=+15=+1000111，B=+24=+0011000，则[A]补=00001111，[B]补=00011000。按变补规则求得[-B]补=11101000，所以，[A+B]补=00001111+00011000=00100111，其符号位为0，对应真值为+39。  
[A-B]补=[A]补+[-B]补=00001111+11101000=11110111，其符号位为1，对应真值为-9。

1. **已知X和Y,用变形补码计算出X-Y,并说明结果是否溢出**
2. **X=0.11100,Y=-0.11111**

. -Y=+0.11111   
. [X]补 =00.11100   
. [-Y]补 =00.11111 (+  
-----------------------------  
. [X-Y]补 =01.11011   
双符号位为01，结果为正数，正溢出(上溢)  
. X-Y =+1.11011

1. **X=-0.11101,Y=0.11010**

-Y=-0.11010  
. [X]原 =11.11101  
. [X]反 =11.00010  
. [X]补 =11.00011  
. [-Y]原 =11.11010  
. [-Y]反 =11.00101  
. [-Y]补 =11.00110  
. [X]补 =11.00011 (+  
----------------------------  
. [X-Y]补 =10.01001 最高符号位进位自然丢失  
双符号位为10，结果为负数，负溢出（下溢）  
. [X-Y]反 =10.01000  
. [X-Y]原 =10.10111  
. X-Y = -1.10111

1. **（-7）+（-6）**

1001

1,010

0,011 =溢出,1 丢掉

**浮点数的规格化表示**

1. **十进制-13/128写成二进制浮点数（规格化表示）；**

**-13/128二进制表示：-(0.0001101)**

**浮点数规格化：-(0.0001101) =-0.**1101**\*2^(-11)**

1. **十进制13/128写成二进制浮点数（规格化表示）；**

**13/128二进制表示：0.0001101000**

**浮点数规格化：0.0001101000=0.1101\*2^(-11)**

1. **十进制7.5写成二进制浮点数（规格化表示）；**

**7.5二进制表示： 111.1**

**浮点数规格化：111.1=0.1111\*2^11**

1. **十进制-27/64写成二进制浮点数（规格化表示）；**

**-27/64二进制表示： 0.011011=**

**浮点数规格化：0.011011=0.110110\*2^-1**

1. **十进制-19/128写成二进制浮点数（规格化表示）；**

**-19/128二进制表示： -0.0010011**

**浮点数规格化：-0.0010011=-0.10011\*2^(-10)**

1. **十进制-54写成二进制浮点数（规格化表示）；**

**-54二进制表示： -110110**

**浮点数规格化：-110110=-(0.11011)\*2^(110)**

1. **十进制-53/512写成二进制浮点数（规格化表示）；**

**-53/512二进制表示： -0.000110101**

**浮点数规格化：-0.000110101=-0.110101\*2^(-11)**

1. **十进制15/32写成二进制浮点数（规格化表示）；**

**15/32二进制表示： 0.01111**

**浮点数规格化：0.01111=0.1111\*2^(-1)**

1. **十进制-1.25写成二进制浮点数（规格化表示）；**

**-1.25二进制表示： -1.01**

**浮点数规格化：-1.01=-0.101\*2^1**

1. **十进制56写成二进制浮点数（规格化表示）；**

**56二进制表示： 111000**

**浮点数规格化：111000=0.111\*2^110**