#### 原码除法

$$[x]_{\mathbb{F}} = x_f . x_{n-1} ... x_1 x_0$$
,  $[y]_{\mathbb{F}} = y_f . y_{n-1} ... y_1 y_0$ 

则商q=x/y的原码为:

$$[q]_{\mathbb{R}} = (x_f \oplus y_f) + (0.x_{n-1}...x_1x_0 / 0.y_{n-1}...y_1y_0)$$

例:被除数x=0.1001,除数y=0.1011。手工计算x/y的过程如下:

0. 1 1 0 1	
0. 1 0 1 1 0. 1 0 0 1	
<b>-0.1011</b>	x < y,商0
0. 1 0 0 1 0	得余数r <sub>0</sub> (即x)
-0.01011	除数右移一位,余数减除数,商1
0. 0 0 1 1 1 0	得余数r <sub>1</sub>
-0.001011	除数右移一位,余数减除数,商1
0. 0 0 0 0 1 1 0	得余数r <sub>2</sub>
-0.0001011	除数右移一位,余数不减除数,商0
0. 0 0 0 0 1 1 0 0	得余数r <sub>3</sub>
-0.00001011	除数右移一位,余数减除数,商1
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	得余数r <sub>4</sub>

#### 恢复余数法

被除数x=0.1001,除数y=0.1011 [-y]\*= 1.0101

为保证余数左移时符号位不 变,应采用双符号位

<del>打工并12 | 9 以外</del>1

(1)将心算比较余数和除数大小改为减法比较,并且减y改为加[-y]<sub>补</sub>。余数减除数大于等于0则商1,小于0则商0。

(2) 将除数右移改为余 数左移。

如果余数减除数的差 小于0,应将差加上除数 ,恢复原来的余数。这就 是恢复余数法。

0 0. 1 0 0 1	商 说明	
<mark>並不</mark> ├y] <sub>补</sub> 1 1. 0 1 0 1	x—y	
1 1. 1 1 1 0	0 余数r	0<0,商0
+0 0. 1 0 1 1	加y恢	复余数
0 0. 1 0 0 1		
0 1. 0 0 1 0	余数法	左移一位
$+[-y]_{*}$ 1 1. 0 1 0 1	减y比	较
0 0. 0 1 1 1	0.1 余数r	<sub>1</sub> > 0,商1
<b>←</b> 0 0. 1 1 1 0	余数法	生移一位
$+[-y]_{i}$ 1 1. 0 1 0 1	减y比	较
0 0. 0 0 1 1	0.11 余数r	<sub>2</sub> > <b>0</b> ,商 <b>1</b>
<b>←</b> 0 0. 0 1 1 0	余数	左移一位
$+[-y]_{*}$ 1 1. 0 1 0 1	减y比	较
1 1. 1 0 1 1	0.110 余数r	3 < 0,商0
+0 0. 1 0 1 1	加y恢	复余数
0 0. 0 1 1 0		
-0 0. 1 1 0 0	余数点	左移一位
$+[-y]_{i}$ 1 1. 0 1 0 1	减y比	较
0 0. 0 0 0 1	0.1101 余数r	4 > 0,商1

#### 加减交替法

恢复余数法中,设某次余数为r<sub>i</sub>,要继续进行下面的求商运算,需要将r<sub>i</sub>左移一位,然后减去除数,进行比较:

结果小于0时商上0,并加y恢复余数:

$$(2r_i-y)+y=2r_i$$

继续下面的求商,又要将它左移一位,再减去除数

$$2(2r_i)-y=4r_i-y$$

当(2r<sub>i</sub>-y)小于0时,商仍上0,但不进行加y恢复余数的操作,而是将 (2r<sub>i</sub>-y)左移一位,然后加上除数

$$2(2r_i-y)+y=4r_i-y$$

也得到同样的余数(4r<sub>i</sub>-y)。所以,当比较结果小于0时,仍将结果左移一位,然后加上除数y。这就是不恢复余数法,也称加减交替法。

加减交替法的运算规则是:

余数为正时,商上1,余数左移一位,再减去除数,得到新的余数; 余数为负时,商上0,余数左移一位,再加上除数,得到新的余数。

0 0. 1 0 0 1	商	说明
+[-y]* 1 1. 0 1 0 1		x—y
1 1. 1 1 1 0	0	余数 $r_0 < 0$ ,商 $0$
<b>←</b> 1 1. 1 1 0 0		左移一位
+y 0 0. 1 0 1 1		余数为负,加 <b>y</b>
0 0. 0 1 1 1	0.1	余数 <b>r</b> <sub>1</sub> > <b>0</b> ,商 <b>1</b>
<b>←</b> 0 0. 1 1 1 0		左移一位
+[-y] <sub>补 1 1. 0 1 0 1</sub>		余数为正,减y
0 0. 0 0 1 1	0.11	余数 $r_2 > 0$ ,商1
<b>←</b> 0 0. 0 1 1 0		左移一位
+[-y] † 1 1. 0 1 0 1		余数为正,减y
1 1. 1 0 1 1	0.110	余数 <b>r</b> <sub>3</sub> < <b>0</b> ,商 <b>0</b>
<b>←</b> 1 1. 0 1 1 0		左移一位
+y 0 0. 1 0 1 1		余数为负,加 <b>y</b>
0 0. 0 0 0 1	0.1101	余数r <sub>4</sub> > 0,商1

余数:

注意: 得到的余数是经过左移4次后的结果

故余数为 0.0001 × 2 - 4

不带符号原码除法

即:被除数、除数的绝对值(都转为正数)

## 余数符号

### 以十进制为例:

$$14 \div 3 = 4 \cdots 2$$
  
 $(-14) \div (-3) = 4 \cdots -2 \leftarrow 余数变号$   
 $14 \div (-3) = -4 \cdots 2$   
 $(-14) \div 3 = -4 \cdots -2 \leftarrow 余数变号$ 

### 被除数=除数×商+余数

• 当被除数为负时,余数要变号

· 如何变号? 从[y]\*\* 变成[-y]\*\*

• 原因: 当为负数时,变成绝对值进行处理,已经变号

- 商的符号由被除数、除数符号决定。
- 余数的符号由被除数决定

### 运算过程中,当余数为负数时

- •恢复余数法,需要对进行恢复余数
- •加减交替法的余数需要"回溯"至上一个正数

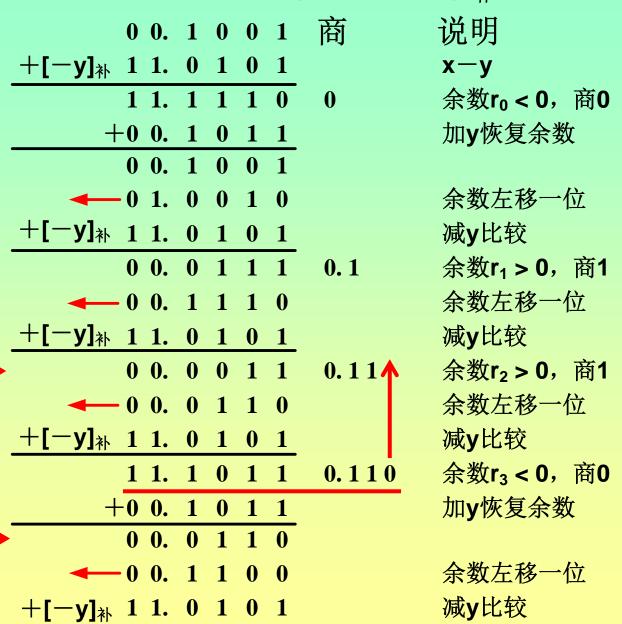
为什么?原因何在?

人工:心算,不够减就不减。

计算机: 先减,发现结果为负,说明不够减,商0余数是已经减完之后的结果,需要恢复。

#### 恢复余数法

被除数x=0.1001,除数y=0.1011 [-y]<sub>补</sub>= 1.0101

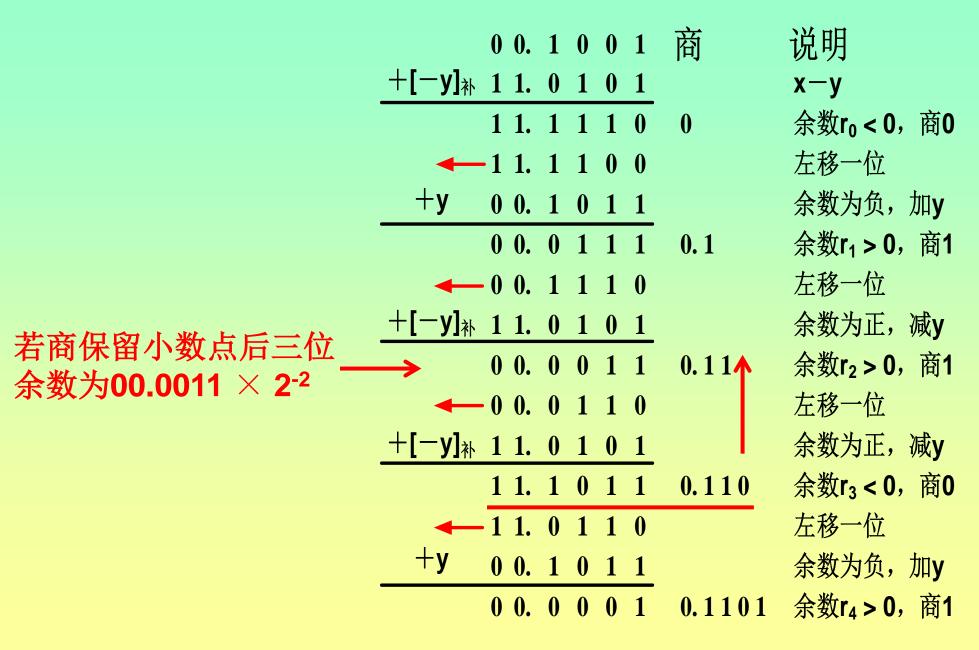


0.1101

若商保留小数点后三位 余数为00.0011 × 2-2

若商保留小数点后三位 余数为00.0110 × 2<sup>-3</sup>

余数r<sub>4</sub> > 0,商1



## 总结

- 若余数为负数,需回溯至上一个正数
- 余数是经过左移后的结果,需进行还原,即×2-n
- 当被除数为负时,余数要变号

## x= - 01011, y=11001, 计算x÷y

- 取|x|来参加运算
- 乘一个比例因子(2-5)变成小数

 $|x| \rightarrow 00.01011, y \rightarrow 00.11001$ 

 $-y \rightarrow 11.00111$ 

00.01011

-y <u>11.00111</u>

11.10010  $\rightarrow$  0.

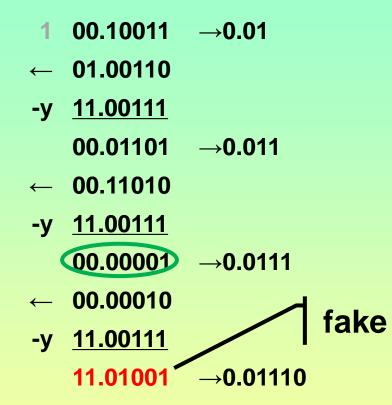
**← 11.00100** 

+y <u>00.11001</u>

**11.11101** →**0.0** 

**← 11.11010** 

+y <u>00.11001</u>



- ①:最后一位商为0,所以为负的余数是"假"的。真实的余数需要回溯到上一个商1处:
- ②: **0.00001**是左移了四次的结果,所以真正的余数还需要右移四次,为**0.00000001** ( 2<sup>-9</sup> )
- ③:此时的余数是绝对值相除的余数,余数还需要根据被除数的符号变号,为-0.000000001
- ④: 因为被除数和除数异号,所以商为-0.01110

### 验算

- 被除数=除数×商+余数(被除数和除数都变回整数,乘25)
- 被除数: -01011(乘2<sup>5</sup>),除数11001(乘2<sup>5</sup>), 商-0.01110(不变),余数-0.0001(乘2<sup>5</sup>),

- 除数×商 = 11001 × (-0.0111) = -1010.1111
- 加上余数 0.0001 得 1011

# 总结

• 若余数为负数,需回溯至上一个正数

• 余数是经过左移后的结果,需还原,即× 2-n

• 当被除数为负时,余数要变号

• 若被除数、除数乘比例因子,余数需还原。