计算机组成与结构
—运算方法3

计算机科学与技术学院

温故 —关于数制表示

- **简述原码一位乘法原理?**
- **模拟手工乘法，符号位运算+数值位运算**
- **符号位：同号为正，异号为负。（相当于异或运算）**
- **数值位：将其中一个乘数固定住，另外一个乘数（1的个数较少）的每一位从低位到高位拆分出来与之相乘**
- **简述原码两位乘法原理?**
- **每次取两位进行运算**
- $+3|X|=+4|X|-|X|$，**存在进位（是否欠**$+4|X|$**）**
- **无论是否“欠”**$|X|$**，一次最多**$+2|X|$
- **如果“当轮”+“欠的”>2，继续“欠”下去**

定点数运算—乘法

原码二位乘法，

一次乘两位数字？这个在十进制下很反人类啊…，但在二进制下，似乎还好…

$$
\begin{array}{rrrrrrrrrl}
 & & & & 1 & 1 & 0 & 1 & & \\
\times & & & & 1 & 0 & 0 & 1 & & \\
\hline
 & & & & 1 & 1 & 0 & 1 & \leftarrow & +|X| \\
 & & & 1 & 1 & 0 & 1 & 0 & \leftarrow & +2|X| \\
\hline
0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & &
\end{array}
\qquad
\begin{array}{rrrrrrrrrl}
 & & & & 1 & 1 & 0 & 1 & & \\
\times & & & & 1 & 1 & 0 & 0 & & \\
\hline
 & & & & 0 & 0 & 0 & 0 & \leftarrow & +0 \\
 & & 1 & 0 & 0 & 1 & 1 & 1 & \leftarrow & +3|X| \\
\hline
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & &
\end{array}
$$

规则似乎很直观：

$$
\begin{cases}
Y_{i+1}Yi = 00, +0 \\
Y_{i+1}Yi = 01, +|X| \\
Y_{i+1}Yi = 10, +2|X| \\
Y_{i+1}Yi = 11, +3|X|
\end{cases}
$$

定点数运算—乘法

原码二位乘法，

一次乘两位数字？这个在十进制下很反人类啊…，但在二进制下，似乎还好…

$$
\begin{array}{rl}
 & 1\ 1\ 0\ 1 \\
\times & 1\ 0\ 0\ 1 \\
\hline
 & 1\ 1\ 0\ 1 \quad +|X| \\
 & 1\ 1\ 0\ 1\ 0 \quad +2|X| \\
\hline
 & 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1
\end{array}
\qquad
\begin{array}{rl}
 & 1\ 1\ 0\ 1 \\
\times & 1\ 1\ 0\ 0 \\
\hline
 & 0\ 0\ 0\ 0 \quad +0 \\
 & 1\ 0\ 0\ 1\ 1\ 1 \quad +3|X| \\
\hline
 & 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0
\end{array}
$$

规则似乎很直观：

$$
\begin{cases}
Y_{i+1}Yi = 00, +0 \\
Y_{i+1}Yi = 01, +|X| \\
Y_{i+1}Yi = 10, +2|X| \\
Y_{i+1}Yi = 11, +3|X|
\end{cases}
$$

规则还可以更进一步优化：

$$3|X| = +4|X| - |X|$$

⇒先减 $|X|$，平移2位后加 $|X|$

怎么做减法？把X和D用补码表示

+4X

$$
\begin{array}{rl}
 & 0\ 0\ 1\ 1\ 0\ 1 \\
\times & 0\ 0\ 1\ 1\ 0\ 0 \\
\hline
 & 0\ 0\ 0\ 0\ 0\ 0 \\
 & 1\ 1\ 0\ 0\ 1\ 1 \quad -|X| \\
 & 0\ 0\ \boxed{1\ 1\ 0\ 1} \quad +|X| \\
\hline
 & 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0
\end{array}
$$

定点数运算—乘法

原码二位乘法，完整规则表

标记是否进位

Y_{i+1}	Y_i	C	操作
0	0	0	+0，右移2次，C＝0
0	0	1	+|X|，右移2次，C＝0
0	1	0	+|X|，右移2次，C＝0
0	1	1	+2|X|，右移2次，C＝0
1	0	0	+2|X|，右移2次，C＝0
1	0	1	－|X|，右移2次，C＝1
1	1	0	－|X|，右移2次，C＝1
1	1	1	+0，右移2次，C＝1

例，[X]原=0.100111, [Y]原=1.100111，求积

解：[2X]补=01.001110; [-X]补=1.011001

符号位	D	A	操作
0 0 0	0 0 0 0 0 0	1 0 0 1 1 1	C=0
1 1 1	0 1 1 0 0 1		−X
1 1 1	0 1 1 0 0 1		C=1
1 1 1	1 1 0 1 1 0	0 1 1 0 0 1	→右移二次
0 0 1	0 0 1 1 1 0		C=1,+2X
0 0 1	0 0 0 1 0 0		C=0
0 0 0	0 1 0 0 0 1	0 0 0 1 1 0	→右移二次
0 0 1	0 0 1 1 1 0		C=0,+2X
0 0 1	0 1 1 1 1 1		C=0
0 0 0	0 1 0 1 1 1	1 1 0 0 0 1	→右移二次

图 3-9　例 3-12 的二位乘法的运算过程

原码两位乘法运算过程

最终结果，[X]原x[Y]原 =1.010111110001

定点数运算—补码乘法

定点数运算—补码乘法

补码乘法

- ~~**校正法**~~
- **Booth法**
  - ~~**两位Booth法**~~

定点数运算—补码乘法

```
                001101
X               001110
--------------------------
                000000
               001101
              001101
             001101
            000000
           000000
--------------------------
           00010110110
```

<—无脑计算需要6步

v.s

略用心机只需3步—>

```
                001101
X               001110
--------------------------
              0001101
              001101
             001101
--------------------------
           00010110110
```

只关心Y为1的位

定点数运算—补码乘法

```
                0 0 1 1 0 1
X               0 0 1 1 1 0
---------------------------
                0 0 0 0 0 0
              0 0 1 1 0 1
            0 0 1 1 0 1
          0 0 1 1 0 1
        0 0 0 0 0 0
      0 0 0 0 0 0
---------------------------
      0 0 0 1 0 1 1 0 1 1 0
```

<—无脑计算需要6步

v.s

略用心机只需3步—>

能不能更简化！？

```
                0 0 1 1 0 1
X               0 0 1 1 1 0
---------------------------
            0 0 0 1 1 0 1
            0 0 1 1 0 1
          0 0 1 1 0 1
---------------------------
0 0 0 1 0 1 1 0 1 1 0
```

定点数运算—补码乘法

```
                    0 0 1 1 0 1
X                   0 0 1 1 1 0
-----------------------------------
                    0 0 0 0 0 0
                  0 0 1 1 0 1
                0 0 1 1 0 1
              0 0 1 1 0 1
            0 0 0 0 0 0
          0 0 0 0 0 0
-----------------------------------
          0 0 0 1 0 1 1 0 1 1 0
```

<—无脑计算需要6步

v.s

略用心机只需3步—>

```
                    0 0 1 1 0 1
X                   0 0 1 1 1 0
-----------------------------------
                0 0 0 1 1 0 1
                0 0 1 1 0 1
              0 0 1 1 0 1
-----------------------------------
          0 0 0 1 0 1 1 0 1 1 0
```

能不能更简化！？

先看一个日常使用的技巧：

$$12345 \times 1001 = 12345 \times (1000 + 1)$$

$$2342 \times 999 = 2342 \times (1000 - 1)$$

对二进制，也一样！

$$1010 \times 0111 = 1010 \times (1000 - 1)$$

$$1010 \times 1110 = 1010 \times (10000 - 10)$$

简化条件：

存在"一串1"

定点数运算—补码乘法

```
                0 0 1 1 0 1
X               0 0 1 1 1 0
---------------------------
            0 0 0 1 1 0 1
            0 0 1 1 0 1
        0 0 1 1 0 1
---------------------------
0 0 0 1 0 1 1 0 1 1 0
```

<—略用心机需要3步

v.s

运用技巧只需2步—>

在此处加$|X|$

在此处减$|X|$

```
                  0 0 1 1 0 1
X                 0 0 1 1 1 0
-----------------------------
[-X]补  1 1 1 1 1 0 0 1 1
        0 0 1 1 0 1
-----------------------------
        0 0 1 0 1 1 0 1 1 0
```

问题来了：如何高效地识别“一串1”的开始和结束？

```
  0 0 1 1 1 0 = 0 1 0 0 0 0 - 0 0 0 0 1 0
+ 0 1 0 0 0 0
— 0 0 0 0 1 0
```

结束 $+|X|$

开始 $-|X|$

定点数运算—补码乘法

```
            0 0 1 1 0 1
X           0 0 1 1 1 0
---------------------
          0 0 0 1 1 0 1
          0 0 1 1 0 1
        0 0 1 1 0 1
---------------------
0 0 0 1 0 1 1 0 1 1 0
```

<—略用心机需要3步
v.s
运用技巧只需2步—>

在此处加$|X|$
在此处减$|X|$

```
                0 0 1 1 0 1
X               0 0 1 1 1 0
-------------------------
[-X]补  1 1 1 1 1 0 0 1 1
        0 0 1 1 0 1
-------------------------
        0 0 1 0 1 1 0 1 1 0
```

问题来了：如何高效地识别“一串1”的开始和结束?

Booth算法:
- 从右向左，遇到“1 0”就是“一串1”的开始，在1的位置减$|X|$;
- 从右向左，遇到“0 1”就是“一串1”的结束，在0的位置加$|X|$;
- 中间的“00”和“11”都不管!

y_i	y_{i-1}	操　作
0	0	部分积＋0，右移1位
0	1	部分积＋[X]补，右移1位
1	0	部分积＋[-X]补，右移1位
1	1	部分积＋0，右移1位

边界?　0 1 1 1 1 ⟶ 0 1 1 1 1. 0

最低位就是1，如何统一规则

小数点后一位补0

booth乘法

Booth算法：

- 从右向左，遇到“1 0”就是“一串1”的开始，在1的位置减$|X|$；
- 从右向左，遇到“0 1”就是“一串1”的结束，在0的位置加$|X|$；
- 中间的“00”和“11”都不管！

y_i	y_{i-1}	操　作
0	0	部分积＋0，右移1位
0	1	部分积＋$[X]_{补}$，右移1位
1	0	部分积＋$[-X]_{补}$，右移1位
1	1	部分积＋0，右移1位

①：booth乘法的乘数和被乘数还有结果都应由补码表示。

②：booth乘法计算前应在乘数末尾补零。（防止末尾出现11的情况）

③：booth乘法的符号位参与计算。

④：booth乘法应以双符号位方式进行计算，防止结果溢出。

补码乘法

y_i	y_{i-1}	操　作
0	0	部分积＋0，右移1位
0	1	部分积＋$[X]_补$，右移1位
1	0	部分积＋$[-X]_补$，右移1位
1	1	部分积＋0，右移1位

【例】X＝0.1010，Y＝ - 0.1101，利用Booth法求积。

【解】：

$[X]_补$ ＝00.1010

$[-X]_补$ ＝11.0110

$[Y]_补$ ＝11.0011

$\therefore [X\cdot Y]_补$ ＝1.01111110

符号位　　在此处减|X|

符号	D	A	A_{-1}	操作说明
0 0	0 0 0 0	1 0 0 1 1	0	
1 1	0 1 1 0			＋$[-X]_补$
1 1	0 1 1 0			
1 1	1 0 1 1	0 1 0 0 1	1	右移1位
0 0	0 0 0 0			＋0
1 1	1 0 1 1			
1 1	1 1 0 1	1 0 1 0 0	1	右移1位
0 0	1 0 1 0			＋$[X]_补$
0 0	0 1 1 1			
0 0	0 0 1 1	1 1 0 1 0	0	右移1位
0 0	0 0 0 0			＋0
0 0	0 0 1 1			
0 0	0 0 0 1	1 1 1 0 1	0	右移1位
1 1	0 1 1 0			＋$[-X]_补$
1 1	0 1 1 1	1 1 1 0 1	0	不移位

补0

在此处加|X|

无需右移

$[X]_补 \cdot [Y]_补 \neq [X\cdot Y]_补$ —— 校正

定点数运算—乘法

练习，$[X]_{原}$=0.110110, $[Y]_{原}$=1.101101

采用booth算法求积

XY交换等效：

$[X]_{原}$= 1.101101, $[Y]_{原}$= 0.110110

解：[Y]补=00.110110;

[-X]补=00.101101

[X]补=11.010011

结果为： 1.0110 1000 0010

原码乘法结果： 1.1001 0111 1110

$[XY]_{补}$= 1.0110 1000 0010

$[XY]_{反}$= 1.0110 1000 0001

$[XY]_{原}$= 1.1001 0111 1110

符号	D	A	A_{-1}	操作说明
0 0	0 0 0 0 0 0	0 1 1 0 1 1 0	0	
0 0	0 0 0 0 0 0	0 0 1 1 0 1 1	0	右移1位
0 0	1 0 1 1 0 1			＋$[-X]_{补}$
0 0	1 0 1 1 0 1	0 0 1 1 0 1 1	0	
0 0	0 1 0 1 1 0	1 0 0 1 1 0 1	1	右移1位
0 0	0 0 1 0 1 1	0 1 0 0 1 1 0	1	右移1位
1 1	0 1 0 0 1 1			＋$[X]_{补}$
1 1	0 1 1 1 1 0	0 1 0 0 1 1 0	1	
1 1	1 0 1 1 1 1	0 0 1 0 0 1 1	0	右移1位
0 0	1 0 1 1 0 1			＋$[-X]_{补}$
0 0	0 1 1 1 0 0	0 0 1 0 0 1 1	0	
0 0	0 0 0 1 1 1	0 0 0 0 1 0 0	1	右移2位
1 1	0 1 0 0 1 1			＋$[X]_{补}$
1 1	0 1 1 0 1 0	0 0 0 0 1 0 0	1	
1 1	0 1 1 0 1 0	0 0 0 0 1 0 0	1	不移位

定点数运算—除法

定点数运算—除法

按理说，原码一位除法就是模拟手工除法:

- **符号**$S_m = S_A \oplus S_b$**, 数值**$|m| = |A|/|B|$
- **余数符号与被除数相同**

例，[X]原=0.1101, [Y]原=1.1011，求积

解： 符号，$S = 0 \oplus 1 = 1$**； 数值部分如下:**

模拟手工除法

```
                     0 .1  1  0  1
           ________________________
.1 1 0 1 ) .1  0  1  1
            0  0  0  0
           ------------
            1  0  1  1  0
               1  1  0  1
              ------------
               1  0  0  1  0
                  1  1  0  1
                 ------------
                  0  1  0  1  0
                     0  0  0  0
                    ------------
                     1  0  1  0  0
                        1  1  0  1
                       ------------
           .0  0  0  0  0  1  1  1
```

定点数运算—除法

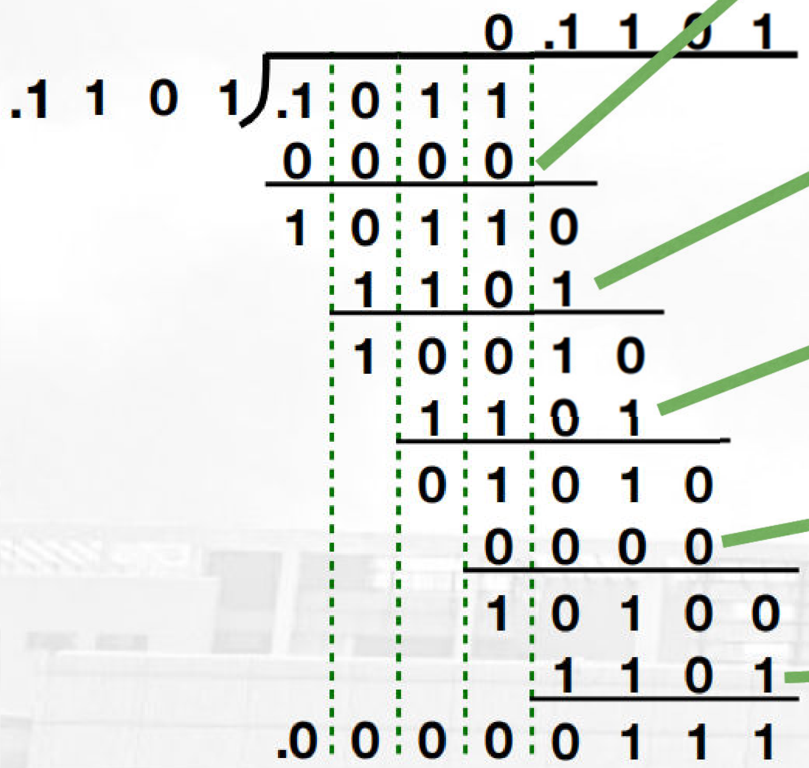
按理说，原码一位除法就是模拟手工除法:

- 符号$S_m = S_A \oplus S_b$，数值$|m| = |A|/|B|$
- 余数符号与被除数相同

例，[X]原=0.1101, [Y]原=1.1011，求积

解： 符号，$S = 0 \oplus 1 = 1$； 数值部分如下:

模拟手工除法



使用临时变量R逐层累减除数D

层次	被除数(余数) R	商↓	说明
第一层 减0	0 1 0 1 1		R
	- 0 0 0 0 0	0	不够减、商0
	0 1 0 1 1		R = R - D * 0
第二层 减1101	1 1 0 1 0		R << 1
	- 0 1 1 0 1	1	够减，商1
	0 1 0 0 1		R = R - D
第三层 减1101	1 0 0 1 0		R << 1
	- 0 1 1 0 1	1	够减，商1
	0 0 1 0 1		R = R - D
第四层 减0	0 1 0 1 0		R << 1
	- 0 0 0 0 0	0	不够减、商0
	0 1 0 1 0		R = R - D * 0
第五层 减1101	1 0 1 0 0		R << 1
	- 0 1 1 0 1	1	够减，商1
	0 0 1 1 1		R = R - D

定点数运算—除法

按理说，原码一位除法就是模拟手工除法:

- 符号$S_m = S_A \oplus S_b$，数值$|m| = |A|/|B|$
- 余数符号与被除数相同

例，[X]原=0.1101, [Y]原=1.1011，求积

解：符号，$S = 0 \oplus 1 = 1$；数值部分如下:

模拟手工除法

```
                 0 .1  1  0  1
.1  1  0  1 ) .1  0  1  1
               0  0  0  0
               1  0  1  1  0
                  1  1  0  1
                  1  0  0  1  0
                     1  1  0  1
                     0  1  0  1  0
                        0  0  0  0
                        1  0  1  0  0
                           1  1  0  1
              .0  0  0  0  0  1  1  1
```

如何高效处理商0/1问题?

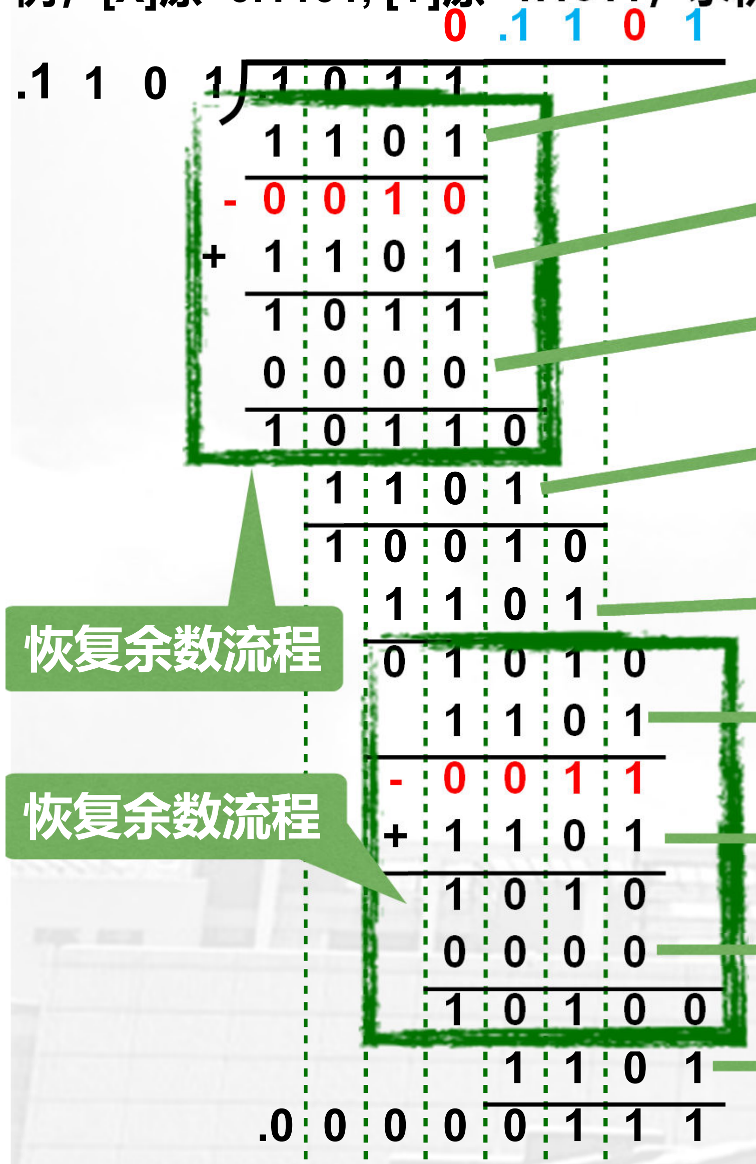
使用临时变量R逐层累减除数D

层次	被除数(余数) R	商↓	说明
第一层 减0	0 1 0 1 1		R
	- 0 0 0 0 0	0	不够减、商0
	0 1 0 1 1		R = R - D * 0
第二层 减1101	1 1 0 1 0		R << 1
	- 0 1 1 0 1	1	够减，商1
	0 1 0 0 1		R = R - D
第三层 减1101	1 0 0 1 0		R << 1
	- 0 1 1 0 1	1	够减，商1
	1 0 1		R = R - D
第四层 减0	0 1 0 1 0		R << 1
	- 0 0 0 0 0	0	不够减、商0
	0 1 0 1 0		R = R - D * 0
第五层 减1101	1 0 1 0 0		R << 1
	- 0 1 1 0 1	1	够减，商1
	0 0 1 1 1		R = R - D

定点数运算—除法

解决方案一：无脑直减，减多了原地恢复（恢复余数法）

例，[X]原=0.1101, [Y]原=1.1011，求积



使用临时变量R逐层累减除数D

层次	被除数(余数) R	商↓	说明
第一层 减1101	0 1 0 1 1 - 0 1 1 0 1 - 0 0 0 1 0		R > 0 无脑减D R = R - D
第二层 加1101	- 0 0 0 1 0 + 0 1 1 0 1 0 1 0 1 1		R<0 +D恢复R R = R + D
第三层 减0	0 1 0 1 1 - 0 0 0 0 0 0 1 0 1 1	0	0 < R < D 不够减、商0 R = R - D * 0
第四层 减1101	1 1 0 1 0 - 0 1 1 0 1 0 1 0 0 1	1	R << 1 够减，商1 R = R - D
第五层 减1101	1 0 0 1 0 - 0 1 1 0 1 0 0 1 0 1	1	R << 1 够减，商1 R = R - D
第六层 减1101	0 1 0 1 0 - 0 1 1 0 1 - 0 0 0 1 1		R > 0 无脑减D R = R - D
第七层 加1101	- 0 0 0 1 1 + 0 1 1 0 1 0 1 0 1 0		R<0 +D恢复R R = R + D
第八层 减0	0 1 0 1 0 - 0 0 0 0 0 0 1 0 1 0	0	0 < R < D 不够减、商0 R = R - D * 0
第九层 减1101	1 0 1 0 0 - 0 1 1 0 1 0 0 1 1 1	1	R << 1 够减，商1 R = R - D

定点数运算—除法

解决方案一：无脑直减，减多了原地恢复（恢复余数法）

课本P66

【例 3-15】 若被除数 $X=-0.10001011$，除数 $Y=0.1110$ 试利用原码恢复余数法求商及余数。

解：该例满足 $|X|<|Y|$，且 $|Y|\neq 0$。

写出 $[X]_{原}=1.10001011$，$[Y]_{原}=0.1110$。

商符 $=1\oplus 0=1$。绝对值除法过程如图 3-17 所示。

符号	被除数(余数)	商	操作
0 0	1 0 0 0 1 0 1 1	0	
0 1	0 0 0 1 0 1 1 0		左移一位
1 1	0 0 1 0		−|Y|
0 0	0 0 1 1 0 1 1 0	1	够减，商为1
0 0	0 1 1 0 1 1 0 1		左移一位
1 1	0 0 1 0		−|Y|
1 1	1 0 0 0 1 1 0 1	0	不够减，商为0
0 0	1 1 1 0		+|Y|
0 0	0 1 1 0 1 1 0 1	0	恢复余数
0 0	1 1 0 1 1 0 1 0		左移一位
1 1	0 0 1 0		−|Y|
1 1	1 1 1 1 1 0 1 0	0	不够减，商为0
0 0	1 1 1 0		+|Y|
0 0	1 1 0 1 1 0 1 0	0	恢复余数
0 1	1 0 1 1 0 1 0 0		左移一位
1 1	0 0 1 0		−|Y|
0 0	1 1 0 1 0 1 0 0	1	够减，商为1

图 3-17 例 3-15 的恢复余数法过程

定点数运算—除法

解决方案一：无脑直减，<u>减多了原地恢复</u>（恢复余数法）

课本P66

【例 3-15】 若被除数 […] 数 Y=0.1110 试利用原码恢复余数法求商及余数。

不要溢出

不能除0

解：该例满足 $|X|<|Y|$，且 $|Y|\neq 0$。

写出 $[X]_{原}=1.10001011$，$[Y]_{原}=0.1110$。

商符 $=1\oplus 0=1$。绝对值除法过程如图 3-17 所示。

符号	被除数(余数)	商	操作
0 0	1 0 0 0 1 0 1 1	0	
0 1	0 0 0 1 0 1 1 0		左移一位
1 1	0 0 1 0		−|Y|
0 0	0 0 1 1 0 1 1 0	1	够减，商为1
0 0	0 1 1 0 1 1 0 1		左移一位
1 1	0 0 1 0		−|Y|
1 1	1 0 0 0 1 1 0 1	0	不够减，商为0
0 0	1 1 1 0		+|Y|
0 0	0 1 1 0 1 1 0 1	0	恢复余数
0 0	1 1 0 1 1 0 1 0		左移一位
1 1	0 0 1 0		−|Y|
1 1	1 1 1 1 1 0 1 0	0	不够减，商为0
0 0	1 1 1 0		+|Y|
0 0	1 1 0 1 1 0 1 0 0	0	恢复余数
0 1	1 0 1 1 0 1 0 0		左移一位
1 1	0 0 1 0		−|Y|
0 0	1 1 0 1 0 1 0 0	1	够减，商为1

图 3-17 例 3-15 的恢复余数法过程

被除数符号

余数：1.1101×2^{-4}

商：1.1001

商4位小数

定点数运算—除法

解决方案一：无脑直减，<u>减多了原地恢复</u>（恢复余数法）

例，[X]=-0.10001011, [Y]=0.1110

求商及余数。

【解】

[X]原 = 1.10001011

[Y]原 = 0.1110

[- Y]补 = 1.0010

商符 = 1 $\oplus$ 0 = 1

[X÷Y]原 = 1.1001

余数 = 1.1101×2^{-4}

符号	被除数（余数）	商	操作
0 0	1 0 0 0 1 0 1 1	0	
0 1	0 0 0 1 0 1 1 0		左移1位
1 1	0 0 1 0		－|Y|
0 0	0 0 1 1 0 1 1 0	1	R≥0，商为1
0 0	0 1 1 0 1 1 0 1		左移1位
1 1	0 0 1 0		－|Y|
1 1	1 0 0 0 1 1 0 1	0	R＜0，商为0
0 0	1 1 1 0		+|Y|
0 0	0 1 1 0 1 1 0 1	0	恢复余数
0 0	1 1 0 1 1 0 1 0		左移1位
1 1	0 0 1 0		－|Y|
1 1	1 1 1 1 1 0 1 0	0	R＜0，商为0
0 0	1 1 1 0		+|Y|
0 0	1 1 0 1 1 0 1 0	0	恢复余数
0 1	1 0 1 1 0 1 0 0		左移1位
1 1	0 0 1 0		－|Y|
0 0	1 1 0 1 0 1 0 0	1	R≥0，商为1

定点数运算—除法

例，[X]原=0.1101, [Y]原=1.1011，求积

寻找解决方案二….

恢复余数法太笨了，如何优化?

当余数R<0时:

- **商0;**
- **恢复余数：R += D;**
- **余数左移：(R+D) << 1;**
- **进行第$i+1$轮，减D**
- $(R+D) \ll 1 - D = 2(R+D) - D$
- $= \boldsymbol{2R+D}$
- =$\boldsymbol{R \ll 1 + D}$

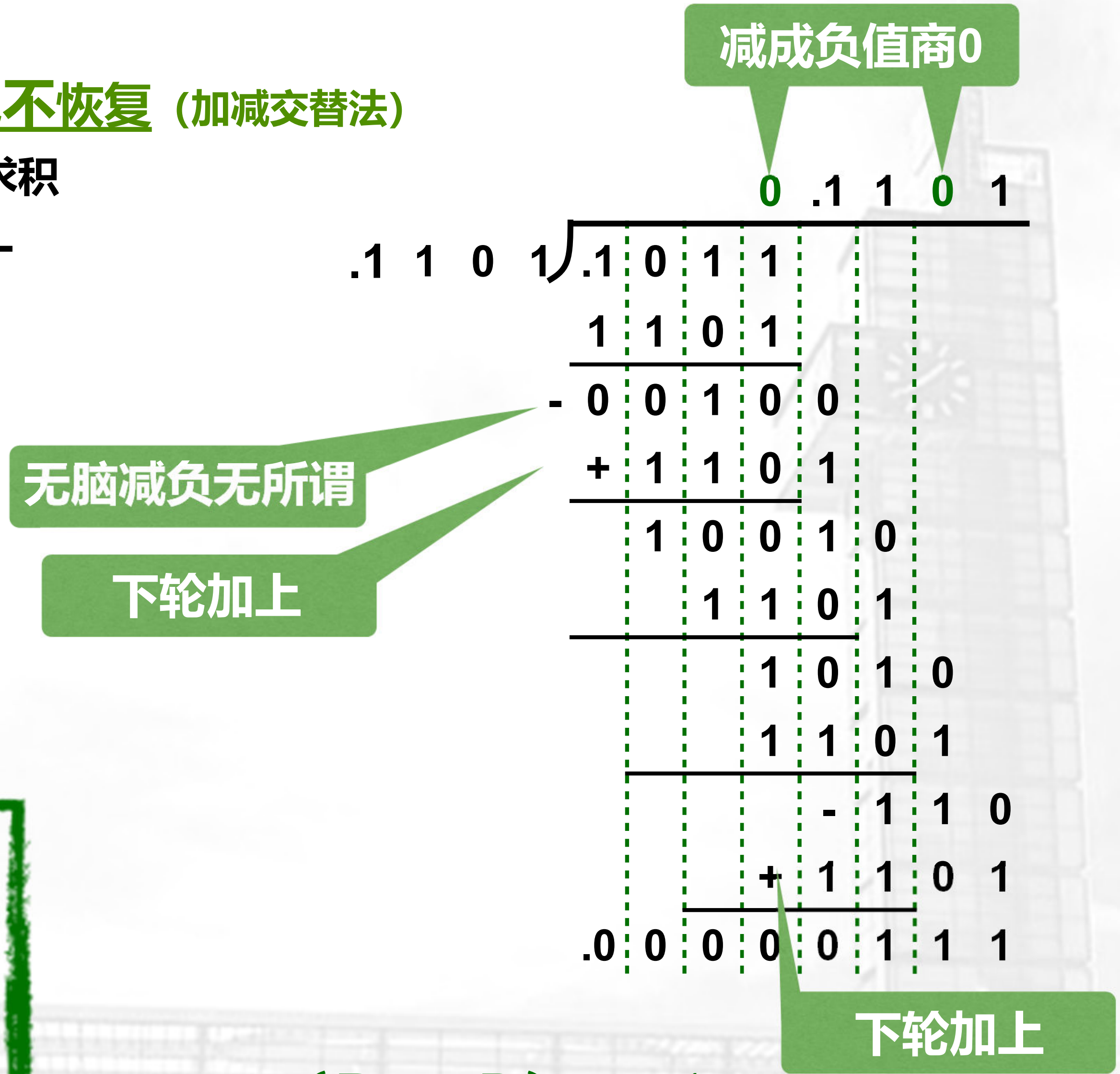
$$(R+D) \ll 1 - D = R \ll 1 + D$$

R<0的恢复流程，等效于R左移+D！！！

定点数运算—除法

解决方案二：无脑直减，减多了原地不恢复（加减交替法）

例，[X]原=0.1101, [Y]原=1.1011，求积



$$(R+D) \ll 1 - D = R \ll 1 + D$$

R<0的恢复流程，等效于R左移+D！！！

简化恢复余数流程

定点数运算—除法

解决方案二：无脑直减，<u>减多了原地不恢复</u>（加减交替法）

例，[X]原=0.1101, [Y]原=1.1011，求积

减成负值商0

```
            0 .1  1  0  1
.1 1 0 1 ) .1  0  1  1
            1  1  0  1
        - 0  0  1  0  0
        +    1  1  0  1
             1  0  0  1  0
             -  1  1  0  1
                1  0  1  0
             -  1  1  0  1
                -  1  1  0
                +  1  1  0  1
        .0  0  0  0  0  1  1  1
```

使用临时变量R逐层累减除数D

层次	被除数(余数) R	商	说明
第一层 减1101	0 0 1 0 1 1		R > 0
	- 0 0 1 1 0 1	0	无脑减D
	- 0 0 0 0 1 0		R = R - D *补
第二层 加1101	- 0 0 0 1 0 0		R<0，R << 1
	+ 0 0 1 1 0 1	1	+D
	0 0 1 0 0 1		R = R + D
第三层 减1101	0 1 0 0 1 0		R > 0
	- 0 0 1 1 0 1	1	无脑减D
	0 0 0 1 0 1		R = R - D
第四层 减1101	0 0 1 0 1 0		R > 0
	- 0 0 1 1 0 1	0	无脑减D
	- 0 0 0 0 1 1		R = R - D
第五层 加1101	- 0 0 0 1 1 0		R<0，R << 1
	+ 0 1 1 0 1	1	+D
	0 0 0 1 1 1		R = R + D

定点数运算—除法

解决方案二：无脑直减，<u>减多了原地不恢复</u>（加减交替法）

例，[X]=-0.10001011, [Y]=0.1110

求商及余数。

【解】

[X]原＝1.10001011

[|Y|]原＝0.1110

[-|Y|]补＝1.0010

商符＝1⊕0＝1

[X÷Y]原＝1.1001

余数＝1.1101×2^{-4}

符号	被除数(余数)	商	操作
0 0	1 0 0 0 1 0 1 1	0	
0 1	0 0 0 1 0 1 1		左移1位
1 1	0 0 1 0		－|Y|
0 0	0 0 1 1 0 1 1	1	R≥0，商为1
0 0	0 1 1 0 1 1		左移1位
1 1	0 0 1 0		－|Y|
1 1	1 0 0 0 1 1	0	R＜0，商为0
1 1	0 0 0 1 1 0		左移1位
0 0	1 1 1 0		＋|Y|
1 1	1 1 1 1 1	0	R＜0，商为0
1 1	1 1 1 1		左移1位
0 0	1 1 1 0		＋|Y|
0 0	1 1 0 1	1	R≥0，商为1

定点数运算—除法

原码加减交替法，其实已经是用补码在计算…

离补码除法，只一步之遥~

定点数运算—除法

回顾原码加减交替法

当余数R>0时：

- 够剪，商1；
- …

符号依赖, R > 0, |D| > 0

当余数R<0时：

- 不够剪，商0；
- 恢复余数：R += D;
- 余数左移：(R+D) << 1；
- 进行第$i+1$轮，减D
  -
- …

符号依赖, R < 0, |D| > 0

若A、B同号，则计算A-B得到余数R

若R与B同号，则说明|A|>|B|，

则够减商1，否则不够减商0；

若A、B异号，则计算A+B得到余数R

若R与B异号，则说明|A|>|B|，

则够减商0，否则不够减商1；

1、余数计算方式：
余数R=+被除数的绝对值—+除数的绝对值

2、判断够减：
被减数的绝对值大于减数的绝对值称为“够减”

3、商0还是商1：
同号：“够减” 商1，“不够减”商0；
异号：“够减” 商0，“不够减”商1；

定点数运算—除法

补码加减交替法

若A、B同号，则计算A-B得到余数R

若R与B同号，则说明|A|>|B|，

则够减商1，否则不够减商0；

若A、B异号，则计算A+B得到余数R

若R与B异号，则说明|A|>|B|，

则够减商0，否则不够减商1；

同号相除时， R与B同号够减商1，R与B异号不够减商0；

异号相除时， R与B异号够减商0， R与B同号不够减商1。

R与B，同号商1，异号商0

定点数运算—除法

补码交替除法完整规则

1. R与D同号，-D
2. R与D异号，+D
3. 当新余数R与D相同符号时，商1
4. 当新余数R与D不同符号时，商0；
5. R左移1位，下一轮回到1
6. 除不尽时，商恒置1

此规则与原码交替除法完全兼容

关于除法的“表格式计算”

例，例，[X]原=-0.1011, [Y]原=0.1101，求商/余数，用补码除法

【解】

[X]补 = 1.0101

[Y]原 = 0.1101

[- Y]补 = 1.0011

1. R与D同号，-D
2. R与D异号，+D
3. 当新余数R与D相同时，商1
4. 当新余数R与D不同时，商0；
5. R左移1位，下一轮回到1
6. 除不尽时，商恒置1

符号	余数（R）	商	操作说明
1 1	0 1 0 1		R初始化

关于除法的“表格式计算”

例，例，[X]原=-0.1011, [Y]原=0.1101，求商/余数，用补码除法

【解】

[X]补 = 1.0101

[Y]原 = 0.1101

[- Y]补 = 1.0011

1. R与D同号，-D
2. R与D异号，+D
3. 当新余数R与D相同时，商1
4. 当新余数R与D不同时，商0；
5. R左移1位，下一轮回到1
6. 除不尽时，商恒置1

符号		余数（R）				商	操作说明
1	1	0	1	0	1		R初始化
0	0	1	1	0	1		R与D异号，+D
0	0	0	0	1	0	1	R与D同号，商1
0	0	0	1	0	0		R左移1位
1	1	0	0	1	1		R与D同号，-D
1	1	0	1	1	1	0	R与D异号，商0
1	0	1	1	1	0		R左移1位
0	0	1	1	0	1		R与D异号，+D
1	1	1	0	1	1	0	R与D异号，商0
1	1	0	1	1	0		R左移1位
0	0	1	1	0	1		R与D异号，+D
0	0	0	0	1	1	1	R与D同号，商1
0	0	0	1	1	0		R左移1位
1	1	0	0	1	1		R与D同号，-D
1	1	1	0	0	1	1	末尾恒1

[X÷Y]补 = 1.0011

余数 = 1.1001×2^{-4}

定点数运算—除法

解决方案三：补码交替加减法

例，[X]=-0.10001011, [Y]=0.1110，求商及余数

【解】

$[X]_{补} = 1.01110101$

$[Y]_{补} = 0.1110$

$[-Y]_{补} = 1.0010$

1. R与D同号，-D
2. R与D异号，+D
3. 当新余数R与D相同时，商1
4. 当新余数R与D不同时，商0；
5. R左移1位，下一轮回到1
6. 除不尽时，商恒置1

符号	余数（R）	商	操作说明
1 1	0 1 1 1 0 1 0 1		R初始化

定点数运算—除法

解决方案三：补码交替加减法

例，[X]=-0.10001011, [Y]=0.1110，求商及余数

【解】

$[X]_补 = 1.01110101$

$[Y]_补 = 0.1110$

$[-Y]_补 = 1.0010$

1. R与D同号，-D
2. R与D异号，+D
3. 当新余数R与D相同时，商1
4. 当新余数R与D不同时，商0；
5. R左移1位，下一轮回到1
6. 除不尽时，商恒置1

符号	余数（R）	商	操作说明
1 1	0 1 1 1 0 1 0 1		R初始化
0 0	1 1 1 0 0 0 0 0		R与D异号，+D
0 0	0 1 0 1 0 1 0 1	1	R与D同号，商1
0 0	1 0 1 0 1 0 1 0		R左移1位
1 1	0 0 1 0 0 0 0 0		R与D同号，-D
1 1	1 1 0 0 1 0 1 0	0	R与D异号，商0
1 1	1 0 0 1 0 1 0 0		R左移1位
0 0	1 1 1 0 0 0 0 0		R与D异号，+D
0 0	0 1 1 1 0 1 0 0	1	R与D同号，商1
0 0	1 1 1 0 1 0 0 0		R左移1位
1 1	0 0 1 0 0 0 0 0		R与D同号，-D
0 0	0 0 0 0 1 0 0 0	1	R与D同号，商1
0 0	0 0 0 1 0 0 0 0		R左移1位
1 1	0 0 1 0 0 0 0 0		R与D同号，-D
1 1	0 0 1 1 0 0 0 0	1	末尾恒1

$[X \div Y]_补 = 1.0111$

$余数 = 1.0011 \times 2^{-4}$