计算机组成原理

定点乘除法运算

王浩宇,教授
haoyuwang@hust.edu.cn
https://howiepku.github.io/

串行乘法基本原理

分析笔算乘法

$A = -0.1101$ $B = 0.1011$

$A \times B = -0.10001111$ **乘积的符号心算求得**

$$
\begin{array}{r}
0.1101 \\
\times\, 0.1011 \\
\hline
1101 \\
1101 \\
0000 \\
1101 \\
\hline
0.10001111
\end{array}
$$

符号位单独处理

乘数的某一位决定是否加被乘数

4个位积一起相加

乘积的位数扩大一倍

笔算乘法改进

$$
\begin{aligned}
A \cdot B &= A \cdot 0.1011 \\
&= 0.1A + 0.00A + 0.001A + 0.0001A \\
&= 0.1A + 0.00A + 0.001(A + 0.1A) \\
&= 0.1A + 0.01[0 \cdot A + 0.1(A + 0.1A)] \\
&= 0.1\{A + 0.1[0 \cdot A + 0.1(A + 0.1A)]\} \\
&= \boxed{2^{-1}}\{A + 2^{-1}[0 \cdot A + 2^{-1}(A + 2^{-1}(A+0))]\}
\end{aligned}
$$

右移一位

第一步 被乘数A + 0 ①

第二步 部分积右移1位，得新的部分积 ②

第三步 部分积 + 被乘数 ③

⋮

第八步 部分积右移1位，得结果 ⑧

改进后的笔算乘法过程（竖式）

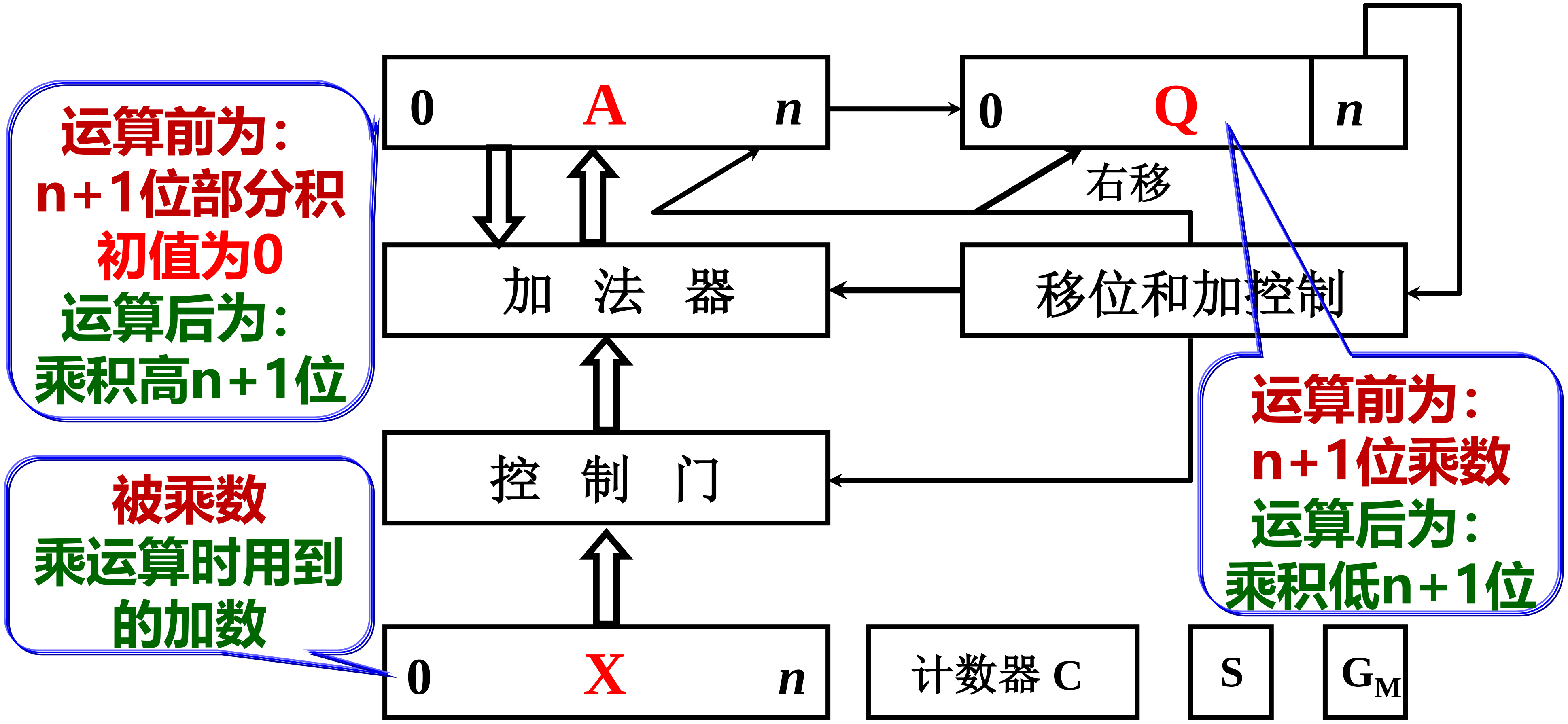
$A = -0.1101$ $B = 0.1011$

部分积	乘数	说明
0.0000 0.1101	1011	初态，部分积 = 0 乘数为 1，加被乘数
0.1101 0.0110 0.1101	1101	→1，形成新的部分积 乘数为 1，加被乘数
1.0011 0.1001 0.0000	1 1110	→1，形成新的部分积 乘数为 0，加 0
0.1001 0.0100 0.1101	11 1111	→1，形成新的部分积 乘数为 1，加 被乘数
1.0001 0.1000	111 1111	→1，得结果

串行乘法基本原理

- **乘法运算 ＝ 加法＋移位。**
  - 若乘数数值位n = 4，则累加 4 次，移位4 次；
- **乘法过程**
  - 由乘数的末位决定被乘数是否与原部分积相加；
    - 被乘数只与部分积的高位相加
  - 部分积右移一位形成新的部分积；
  - 同时乘数右移一位（末位移丢）；
  - 空出高位存放部分积的低位。
- **硬件构成**
  - 3个具有移位功能的寄存器、一个全加器

串行乘法的硬件配置



A、X、Q 均 n+1 位，移位和加受末位乘数控制

常用的串行乘法运算

- **原码乘法（符号位和数值位必须分开计算）**
  - 原码一位乘
    - 一次判断1位，需判断n次（乘数位数为n）；
  - 原码两位乘***
    - 一次判断2位，可提高乘法的运算速度;
- **补码乘法（符号位和数值位可以等同处理）**
  - 补码一位乘
    - 结果修正法——需区分乘数正负号，复杂
    - Booth算法——比较法，符号位直接参与运算

原码一位乘法

- 设$X = X_f . X_1X_2…X_n$， $Y = Y_f . Y_1Y_2…Y_n$， 乘积的符号位为P_f，则$P_f = X_f \oplus Y_f$，$|P| = |X| \cdot |Y|$
- 求|P|的运算规则如下
  - 被乘数和乘数均取绝对值参加运算，符号位单独考虑；
  - 被乘数取双符号位，部分积的长度同被乘数，初值为0；
  - 从乘数的最低位Y_n开始判断：
    - 若Y_n＝1，则部分积加上被乘数|X|，然后右移一位；
    - 若Y_n＝0，则部分积加上0，然后右移一位。
  - 重复，判断n次

例1. 若X=0. 1101，Y=−0. 1011，用原码一位乘法求$[XY]_{原}$

【解答】$|X|=00.1101$（用双符号位表示），$|Y|=0.1011$（用单符号位）

部分积	乘数 Y_n	说明	
0 0. 0 0 0 0	$0.101\underline{1}$		
+ 0 0. 1 1 0 1		Yn=1	加$\vert X\vert$
0 0. 1 1 0 1			
→ 0 0. 0 1 1 0 1	$0.10\underline{1}$	右移一位得P_1	
+ 0 0. 1 1 0 1		Yn=1	加$\vert X\vert$
0 1. 0 0 1 1 1			
→ 0 0. 1 0 0 1 1 1	$0.1\underline{0}$	右移一位得P_2	
+ 0 0. 0 0 0 0		Yn=0	加0
0 0. 1 0 0 1 1 1			
→ 0 0. 0 1 0 0 1 1 1	$0.\underline{1}$	右移一位得P_3	
+ 0 0. 1 1 0 1		Yn=1	加$\vert X\vert$
0 1. 0 0 0 1 1 1 1			
→ 0 0. 1 0 0 0 1 1 1 1	0	右移一位得P_4	

由于$P_f=X_f\oplus Y_f=0\oplus 1=1$，$|P|=|X|*|Y|=0.10001111$，$[XY]_{原}=1.10001111$

原码两位乘法***

- **原码乘**
  - 符号位和数值位部分分开运算；
- **两位乘**
  - 每次用乘数的2位判断原部分积是否加和如何加被乘数；

乘数 $y_{n-1}\,y_n$	新的部分积
0 0	加 "0" $\longrightarrow$ 2
0 1	加 1 倍的被乘数 $\longrightarrow$ 2
1 0	加 2 倍的被乘数 $\longrightarrow$ 2
1 1	加 3 倍的被乘数 $\longrightarrow$ 2

右移两位
再加1倍
的被乘数

3？

$$\begin{array}{r} 4 \\ -1 \\ \hline 3 \end{array} \qquad \begin{array}{r} 100 \\ -01 \\ \hline 11 \end{array}$$

先 减 1 倍 的被乘数
再 加 4 倍 的被乘数

原码两位乘法***

- 先 减 1 倍 的被乘数
- 再 加 4 倍 的被乘数
- **WHY?**

- +(4X-X) 来代替+3X运算，在本次运算中只执行-X，而+4X 则归并到下一步执行
  - 因为下一步运算时，前一次的部分积已右移了两位，上一步欠下的+4X，在本步已变成 +X

原码两位乘法运算规则***

■运算规则如下

- 符号位单独运算，最后的符号$P_f＝X_f \oplus Y_f$。
- 部分积和被乘数均采用三位符号位；
- 乘数末位增加1位C，其初值为0，运算过程见表；

■乘数字长与运算步骤

- 若乘数字长为偶数（不含符号），最多做n/2+1次加法，需做n/2次移位，最后一步不移位；
- 若尾数字长n为奇数（不含符号），增加一位符号0，最多做n/2+1次加法，需做n/2+1次移位，最后一步右移一位

- WHY？

原码两位乘运算规则表***

Y_{n-1}	Y_n	C	操作
0	0	0	加0，右移两位，0 →C
0	1	0	加|X|，右移两位，0 →C
1	0	0	加2|X| ，右移两位，0 →C
1	1	0	减|X|，右移两位，1 →C
0	0	1	加|X|，右移两位，0 →C
0	1	1	加2|X| ，右移两位，0 →C
1	0	1	减|X|，右移两位，1 →C
1	1	1	加0，右移两位，1 →C

例.若X=-0.1101, Y=0.0110,用原码两位乘法求$[XY]_{原}$

【解答】|X|=000.1101，2|X|=001.1010（用三符号表示），

|Y|=00. 0110（用双符号位）

部分积	乘数 C	说明	
0 0 0. 0 0 0 0	$00.01\underline{100}$		
+ 0 0 1. 1 0 1 0		$Y_{n-1}Y_nC=100$	加$2\vert X\vert$
0 0 1. 1 0 1 0			
→ 0 0 0. 0 1 1 0 1 0	$00.\underline{010}$	右移两位	0 →C
+ 0 0 0. 1 1 0 1		$Y_{n-1}Y_nC=010$	加$\vert X\vert$
0 0 1. 0 0 1 1 1 0			
→ 0 0 0. 0 1 0 0 1 1 1 0	$\underline{00.0}$	右移两位	0 →C
		$Y_{n-1}Y_nC=000$	加0
		最后一步不移位	

由于$P_f=X_f\oplus Y_f=0\oplus 1=1$，|P|＝|X|*|Y|=0. 01001110，[XY]原= 1.01001110

定点乘法运算

- 原码一位乘法
- 原码两位乘法
- 补码一位乘法

补码一位乘法

以定点小数为例

- 设被乘数$[x]_补 = x_0 . x_1x_2 \dots x_n$， 乘数$[y]_补 = y_0 . y_1y_2 \dots y_n$
  - 若被乘数任意，乘数为正
    - 同原码乘法规则，但加法和移位操作按补码规则运算；
    - 乘积的符号自然形成；
    - 公式为：$[xy]_补 = [x]_补（0. y_1y_2 \dots y_n）$

算术移位

  - 若被乘数任意，乘数为负
    - 乘数$[y]_补$，去掉符号位，操作同 ①
    - 运算完成后，需对结果加$[-x]_补$校正
    - 公式为： $[xy]_补 = [x]_补（0. y_1y_2 \dots y_n）- [x]_补$

统一的补码乘法公式

$[xy]_补 = [x]_补（0. y_1y_2 \dots y_n） - [x]_补 \cdot y_0$

补码一位乘法运算的证明

- **补码与真值的转换**

设 $[X]_{补} = X_0.X_1X_2....X_n,$

（1） 当X>0时，X_0=0，

$[X]_{补} = 0.\ X_1X_2....X_n = \sum_{i=0}^{n} x^i 2^{-i} = X$

（2） 当X<0时，X_0=1，

$[X]_{补} = 1.\ X_1X_2....X_n = 2+X$

=>

$X = 1.\ X_1X_2....X_n - 2 = -1 + 0.\ X_1X_2....X_n = -1 + \sum_{i=1}^{n} x^i 2^{-i}$

因此，**$X = -X_0 + \sum_{i=1}^{n} x^i 2^{-i}$　　真值与补码之间的关系**

补码一位乘法运算的证明

- **补码的右移**
  - 正数右移一位，相当于乘1/2(即除2)
  - 负数用补码表示时，右移一位也相当于乘1/2
  - 在补码运算的机器中，一个数不论其正负，连同符号位向右移一位，若符号位保持不变，就等于乘1/2
- **证明**

$$X = -X_0 + \sum_{i=1}^{n} x^i 2^{-i}$$

$$\frac{1}{2}X = -\frac{1}{2}X_0 + \frac{1}{2}\sum_{i=1}^{n} x^i 2^{-i}$$

$$= -X_0 + \frac{1}{2}X_0 + \frac{1}{2}\sum_{i=1}^{n} x^i 2^{-i}$$

$$= -X_0 + \frac{1}{2}\sum_{i=0}^{n} x^i 2^{-(i+1)}$$

写成补码形式，即得 $[\frac{1}{2}X]_{补} = X_0 . X_0 X_1 X_2 \ldots X_n$

如果要得到$[2^{-i}X]_{补}$, 只要将$[X]_{补}$连同符号右移i位即可

补码一位乘法运算

设$[x]_{补} = x_0.x_1x_2 \cdots x_n$　　$[y]_{补} = y_0.y_1y_2 \cdots y_n$

$[x \cdot y]_{补}$　　（被乘数、乘数符号任意）

$= [x]_{补}(0.y_1 \cdots y_n) - [x]_{补} \cdot y_0$

$= [x]_{补}(y_1 2^{-1}+y_2 2^{-2}+ \cdots +y_n 2^{-n}) - [x]_{补} \cdot y_0$　　$\boxed{2^{-1} = 2^0 - 2^{-1}}$

$= [x]_{补}(-y_0+y_1 2^{-1}+y_2 2^{-2}+ \cdots +y_n 2^{-n})$　　$\boxed{2^{-2} = 2^{-1} - 2^{-2}}$

$= [x]_{补}[-y_0+(y_1-y_1 2^{-1})+(y_2 2^{-1}-y_2 2^{-2})+ \cdots +(y_n 2^{-(n-1)}-y_n 2^{-n})]$

$= [x]_{补}[(y_1-y_0)+(y_2-y_1)2^{-1}+ \cdots +(y_n-y_{n-1})2^{-(n-1)}+(0-y_n)2^{-n}]$

$= [x]_{补}[(y_1-y_0)+(y_2-y_1)2^{-1}+ \cdots +(y_{n+1}-y_n)2^{-n}]$

附加位　$y_{n+1} = 0$

补码一位乘法运算

$$[x]_补[(y_1-y_0)+(y_2-y_1)2^{-1}+\cdots+(y_{n+1}-y_n)2^{-n}]$$

$$[z_0]_补=0$$

$$[z_1]_补=2^{-1}\{[z_0]_补+(y_{n+1}-y_n)[x]_补\} \qquad (y_{n+1}=0)$$

$$\cdots$$

$$[z_i]_补=2^{-1}\{[z_{i-1}]_补+(y_{n-i+2}-y_{n-i+1})[x]_补\}$$

$$\cdots$$

$$[z_n]_补=2^{-1}\{[z_{n-1}]_补+(y_2-y_1)[x]_补\}$$

$$[z_{n+1}]_补=[z_n]_补+(y_1-y_0)[x]_补=[x\cdot y]_补$$

- 开始时，部分积为 0，即 $[z_0]_补=0$
- 每一步都是在前次部分积的基础上，由 $(y_{i+1}-y_i)$ (i = 0，1，2，…，n) 决定对$[x]_补$的操作，再右移一位，得到新的部分积
- 如此重复 $n+1$步，最后一步不移位，便得到 $[x\cdot y]_补$

Booth算法运算规则

- 符号位参加运算，运算的数均以补码表示
- 被乘数一般取双符号位参加运算
- 乘数可取单符号位
- 乘数末位增设附加位Y_{n+1}，且初值为0
- 按右表操作
- 最后一步不移位，仅根据Y_0与Y_1的比较结果作相应的运算

Y_n（高位）	Y_{n+1}（低位）	操作
0	0	部分积右移一位
0	1	部分积加$[X]_{补}$，右移一位
1	0	部分积加$[-X]_{补}$，右移一位
1	1	部分积右移一位

例. 若X=-0.1101，Y=0.1011，用补码一位乘法求$[XY]_{补}$

[解答] $[X]_{补}$=11.0011，$[-X]_{补}$=00.1101（用双符号表示），

$[Y]_{补}$=0. 1011（用单符号位）

部分积	乘数 Y_n Y_{n+1}	说 明
0 0. 0 0 0 0	0.1 0 1 1 0	
+ 0 0. 1 1 0 1		Y_nY_{n+1}=10 加$[-X]_{补}$
0 0. 1 1 0 1		
→ 0 0. 0 1 1 0 1	0. 1 0 1 1	右移一位
→ 0 0. 0 0 1 1 0 1	0. 1 0 1	Y_nY_{n+1}=11 右移一位
+ 1 1. 0 0 1 1		Y_nY_{n+1}=01 加$[X]_{补}$
1 1. 0 1 1 0 0 1		
→ 1 1. 1 0 1 1 0 0 1	0. 1 0	右移一位
+ 0 0. 1 1 0 1		Y_nY_{n+1}=10 加$[-X]_{补}$
0 0. 1 0 0 0 0 0 1		
→ 0 0. 0 1 0 0 0 0 0 1	0. 1	右移一位
+ 1 1. 0 0 1 1		Y_nY_{n+1}=01 加$[X]_{补}$
1 1. 0 1 1 1 0 0 0 1		最后一步不移位

因此，$[XY]_{补}$= 1. 01110001

例.若X=-0.10010, Y=0.10101，用补码一位乘法求$[XY]_{补}$

【解答】$[X]_{补}$=11.01110，$[-X]_{补}$=00.10010（用双符号表示）

$[Y]_{补}$=0. 10101（用单符号位）

部分积	乘数 $Y_n\ Y_{n+1}$	说　明
0　0 0. 0 0 0 0 0	0.1 0 1 0 1　0	
+ 0 0. 1 0 0 1 0		Y_nY_{n+1}=10　加$[-X]_{补}$
0 0. 1 0 0 1 0		
→1 0 0. 0 1 0 0 1 0	0.1 0 1 0　1	右移一位
+ 1 1. 0 1 1 1 0		Y_nY_{n+1}=01　加$[X]_{补}$
1 1. 1 0 1 1 1 0		
→2 1 1. 1 1 0 1 1 1 0	0. 1 0 1　0	右移一位
+ 0 0. 1 0 0 1 0		Y_nY_{n+1}=10　加$[-X]_{补}$
0 0. 0 1 1 0 1 1 0		
→3 0 0. 0 0 1 1 0 1 1 0	0. 1 0　1	右移一位
+ 1 1. 0 1 1 1 0		Y_nY_{n+1}=01　加$[X]_{补}$
1 1. 1 0 1 0 0 1 1 0		
→ 1 1. 1 1 0 1 0 0 1 1 0	0. 1　0	右移一位

部分积	乘数 $Y_n\ Y_{n+1}$	说　明
1 1. 1 1 0 1 0 0 1 1 0		
+ 0 0. 1 0 0 1 0		Y_nY_{n+1}=10　加$[-X]_{补}$
0 0. 0 1 1 0 0 0 1 1 0		
→ 0 0. 0 0 1 1 0 0 0 1 1 0	0 . 1	右移一位
+ 1 1. 0 1 1 1 0		Y_nY_{n+1}=01　加$[X]_{补}$
1 1. 1 0 1 0 0 0 0 1 1 0		

因此，$[XY]_{补}$= 1. 1010000110

定点除法运算

- 原码除法（恢复余数法）
- 原码除法（加减交替法）
- 补码除法***

原码除法算法原理

分析笔算除法

$x = -0.1011 \quad y = 0.1101$ 求 $x \div y$

$$
\begin{array}{r l}
 & 0.\,1101 \\
0.1101 \,\big)\!\!\! & \overline{0.10110} \\
 & \underline{0.01101} \\
 & 0.010010 \\
 & \underline{0.001101} \\
 & 0.00010100 \\
 & \underline{0.00001101} \\
 & 0.00000111
\end{array}
$$

✓商符单独处理

? 心算上商（比较大小）

? 余数不动低位补“0”
减右移一位的除数

$x \div y = -0.1101$ **商符 心算求得**

余数 $= -0.00000111$

笔算除法和机器除法的比较

笔算除法	机器除法
商符单独处理	符号位异或形成
心算上商	$\lvert x \rvert - \lvert y \rvert > 0$ 上商 1
	$\lvert x \rvert - \lvert y \rvert < 0$ 上商 0
余数 不动 低位补 “0” 减右移一位 的除数	余数 左移一位 低位补 “0” 减 除数

除法的数据约定

- **约定**
  - 小数定点除法 |x| < |y|，整数定点除法 |x| > |y|
    - 避免商溢出
  - 被除数不等于 0，除数不能为 0

定点除法运算

- 机器除法运算的特点：先减，后判
- 方法一：不够减，恢复原来的余数。
  然后再进行减运算，叫恢复余数法
  特点：运算次数不固定，控制复杂，早期使用
- 方法二：不够减时，不必恢复余数
  本次余数为正，商1，下次右移一位做减法运算
  本次余数为负，商0，下次右移一位做加法运算
  这种方法称为加减交替法，也称不恢复余数法
  运算次数固定，控制简单，目前广泛使用

原码恢复余数法

- **运算规则：**
  - 被除数（或余数）的绝对值减去除数的绝对值；
    - 机器内部用补码的加法运算实现 $+[-|Y|]_{补}$
  - 判别余数正负：
    - 若为正数，上商1；
    - 若为负数，上商0，并恢复余数； $+[|Y|]_{补}$
  - 余数和商共同左移一位；
    - 下次减除数，按低位对齐；
  - 重复上述过程(左移n次，上商n + 1次)。

例. $x = -0.1011$，$y = -0.1101$，用原码恢复余数法求X/Y

解： $[x]_{原} = 1.1011$ $[y]_{原} = 1.1101$ $[|y|]_{补} = 0.1101$ $[-|y|]_{补} = 1.0011$

① $x_0 \oplus y_0 = 1 \oplus 1 = 0$

②

被除数（余数）	商	说　明
0 . 1 0 1 1	0 . 0 0 0 0	
1 . 0 0 1 1		$+[-\|y\|]_{补}$
1 . 1 1 1 0	0	余数为负，上商 0
0 . 1 1 0 1		恢复余数 $+[\|y\|]_{补}$
0 . 1 0 1 1	0	恢复后的余数
逻辑左移 1 . 0 1 1 0	0	← 1
1 . 0 0 1 1		$+[-\|y\|]_{补}$
0 . 1 0 0 1	0 1	余数为正，上商 1
1 . 0 0 1 0	0 1	← 1
1 . 0 0 1 1		$+[-\|y\|]_{补}$

例. $x = -0.1011$，$y = -0.1101$，用原码恢复余数法求X/Y

被除数（余数）	商	说　明
0.0101 0.1010 1.0011	011 011	余数为正，上商 1 ← 1 $+[-\lvert y\rvert]_{补}$
1.1101 0.1101	0110	余数为负，上商 0 恢复余数 $+[\lvert y\rvert]_{补}$
0.1010 1.0100 1.0011	0110 0110	恢复后的余数 ← 1 $+[-\lvert y\rvert]_{补}$
0.0111	01101	余数为正，上商 1

$$\frac{|x|}{|y|} = 0.1101 \qquad \therefore \left[\frac{x}{y}\right]_{原} = 0.1101$$

共上商 5 次，移位4次　　　　第一次上商判溢出

原码加减交替法

恢复余数法运算规则

若第i次求的余数为R_i,下一次求的余数为R_{i+1},则

余数R_i>0 商1, R_i左移一位，然后减Y，得到R_{i+1}

$R_{i+1} = 2R_i - |y|$

余数R_i<0 商0, 恢复余数, 左移一位, 然后减Y,

$R_{i+1} = 2(R_i+|y|) - |y| = 2R_i + |y|$

不恢复余数法运算规则

上商“1” $2R_i - |y|$

上商“0” $2R_i + |y|$

加减交替

把恢复余数，左移，减Y三个步骤简化成左移，加Y两个步骤

原码加减交替法的运算规则

- **运算规则：**
  - 符号位不参加运算，取双符号位；
  - 用被除数减去除数：
    - 当余数为正时，商上1，余数左移一位，再减去除数；
    - 当余数为负时，商上0，余数左移一位，再加上除数。
  - 根据余数的正负，再做如上处理（上商、加减除数）
  - 最后一次余数若为负值，还应加上除数以得到正确的余数

（最后一次上商为0而又需得到正确余数，则在这最后一次仍需恢复余数）

  - 运算中，共左移n次，相当于乘2^n，最后的余数应为R_n*2^{-n}

    余数的符号与被除数一致

例. $x = -0.1011$，$y = -0.1101$，用原码加减交替法求X/Y

被除数（余数）	商	说　明
0.1011 1.0011	0.0000	 $+[-\lvert y\rvert]_{补}$
1.1110 1.1100 0.1101	0 0	余数为负，上商0 $\leftarrow 1$ $+[\lvert y\rvert]_{补}$
0.1001 1.0010 1.0011	01 01	余数为正，上商1 $\leftarrow 1$ $+[-\lvert y\rvert]_{补}$
0.0101 0.1010 1.0011	011 011	余数为正，上商1 $\leftarrow 1$ $+[-\lvert y\rvert]_{补}$
1.1101 1.1010 0.1101	0110 0110	余数为负，上商0 $\leftarrow 1$ $+[\lvert y\rvert]_{补}$
0.0111	01101	余数为正，上商1

$[x]_{原} = 1.1011$

$[y]_{原} = 1.1101$

$[\lvert y\rvert]_{补} = 0.1101$

$[-\lvert y\rvert]_{补} = 1.0011$

例. $x = -0.1011$，$y = -0.1101$，用原码加减交替法求X/Y

① $x_0 \oplus y_0 = 1 \oplus 1 = 0$

② $\frac{|x|}{|y|} = 0.1101$

$\therefore \quad [\frac{x}{y}]_{原} = 0.1101$

特点　上商 $n+1$ 次（n为商的精度）

第一次上商判溢出（1—溢出）

移 n 次，加 $n+1$ 次

用移位的次数判断除法是否结束（是否达到要求的精度）

补码加减交替法***（选学）

- 当除数和被除数用补码表示时，判别是否够减，要比较它们绝对值的大小，因此，若两数同符号，要用减，若异号，则要用加。
- 对于判断是否够减，及确定本次上商1还是0的规则，还与结果的符号有关。
- 当商为正时，商的每一位上值与原码表示一致；当商为负时，商的各位应是补码形式的值，一般先按反码值上商，除法完成后，再用最低位加1，形成正确的补码值。

如何判断是否够减？如何上商？如何确定商符？

补码加减交替法——商值的确定

① 比较被除数和除数绝对值的大小

x 与 *y* 同号

$x = 0.1011$　　$[x]_{补} = 0.1011$

$y = 0.0011$　　$[y]_{补} = \boxed{0.}0011$

$$
\begin{array}{rl}
 & [x]_{补} = 0.1011 \\
+ & [-y]_{补} = 1.1101 \\
\hline
 & [R_i]_{补} = \boxed{0.}1000
\end{array}
$$

$|x| > |y|$

$[R_i]_{补}$与$[y]_{补}$同号

“够减”

$x = -0.0011$　　$[x]_{补} = 1.1101$

$y = -0.1011$　　$[y]_{补} = \boxed{1.}0101$

$$
\begin{array}{rl}
 & [x]_{补} = 1.1101 \\
+ & [-y]_{补} = 0.1011 \\
\hline
 & [R_i]_{补} = \boxed{0.}1000
\end{array}
$$

$|x| < |y|$

$[R_i]_{补}$与$[y]_{补}$异号

“不够减”

补码加减交替法——商值的确定

x 与 y 异号

$x = 0.1011$　$[x]_{补} = 0.1011$

$y = -0.0011$　$[y]_{补} = \boxed{1.}1101$

$$\begin{array}{rl} & [x]_{补} = 0.1011 \\ + & [y]_{补} = 1.1101 \\ \hline & [R_i]_{补} = \boxed{0.}1000 \end{array}$$

$|x| > |y|$

$[R_i]_{补}$与$[y]_{补}$异号

"够减"

$x = -0.0011$　$[x]_{补} = 1.1101$

$y = 0.1011$　$[y]_{补} = \boxed{0.}1011$

$$\begin{array}{rl} & [x]_{补} = 1.1101 \\ + & [y]_{补} = 0.1011 \\ \hline & [R_i]_{补} = \boxed{0.}1000 \end{array}$$

$|x| < |y|$

$[R_i]_{补}$与$[y]_{补}$同号

"不够减"

小结

$[x]_{补}$和$[y]_{补}$	求 $[R_i]_{补}$	$[R_i]_{补}$与 $[y]_{补}$
同号	$[x]_{补} - [y]_{补}$	同号，"够减"
异号	$[x]_{补} + [y]_{补}$	异号，"够减"

补码加减交替法——商值的确定

末位恒置“1”法

$[x]_{补}$与$[y]_{补}$同号　$\times.\underbrace{\times\times\times\times}$1　　　　“够减”上“1”

正商　0.　原码　1　　按原码上商　“不够减”上“0”

$[x]_{补}$与$[y]_{补}$异号　$\times.\underbrace{\times\times\times\times}$1　　　　“够减”上“0”

负商　1.　反码　1　　按反码上商　“不够减”上“1”

小 结

$[x]_{补}$与$[y]_{补}$	商	$[R_i]_{补}$与$[y]_{补}$	商 值	
同 号	正	够减　（同号）	1	原码上商
		不够减（异号）	0	
异 号	负	够减　（异号）	0	反码上商
		不够减（同号）	1	

简 化 为

$[R_i]_{补}$与$[y]_{补}$	商值
同 号	1
异 号	0

补码加减交替法——商符的形成

除法过程中自然形成

$[x]_{补}$和$[y]_{补}$同号 ⟶ $[x]_{补}-[y]_{补}$ ⟶ 比较$[R_i]_{补}$和$[y]_{补}$

同号(够) “1”

异号(不够) “0”

原码上商

小数除法 第一次不够上 “0”

正商

$[x]_{补}$和$[y]_{补}$异号 ⟶ $[x]_{补}+[y]_{补}$ ⟶ 比较$[R_i]_{补}$和$[y]_{补}$

异号(够) “0”

同号(不够) “1”

反码上商

小数除法 第一次不够上 “1”

负商

补码加减交替法——新余数的形成

加减交替

$[R_i]_补$和 $[y]_补$	商	新余数
同 号	1	$2[R_i]_补 + [-y]_补$
异 号	0	$2[R_i]_补 + [\ y\ ]_补$

补码加减交替法的运算规则

- **运算规则**
  - 符号位参加运算，除数和被除数均用双符号位补码表示；
  - 第一步的运算
    - 被除数与除数同号，被除数减去除数；
    - 被除数与除数异号，被除数加上除数；
  - 后续步骤的运算
    - 余数与除数同号，商上1，余数左移一位减去除数；
    - 余数与除数异号，商上0，余数左移一位加上除数。
  - 重复步骤 ，包括符号位在内，共做n＋1步。
- **如果对商的精度没有特殊要求，一般可采用“末位恒置1”法，此法操作简单，易于实现，且最大误差仅为2^{-n} 。**

若要求商的精度较高，可按②再进行一次操作，以求得商的第n位。
当除不尽时，若商为负，要在商的最低一位加1，使商从反码值转变成补码值；若商为正，最低位不需加1。

例.设 $x = -0.1011$，$y = 0.1101$，求 $[\frac{x}{y}]_{补}$ 并还原真值

被除数(余数)	商	说　明
1.0101	0.0000	
0.1101		异号做加法
0.0010	1	同号上“1”
0.0100	1	←1
1.0011		$+[-y]_{补}$
1.0111	10	异号上“0”
0.1110	10	←1
0.1101		$+[y]_{补}$
1.1011	100	异号上“0”
1.0110	100	←1
0.1101		$+[y]_{补}$
0.0011	1001	同号上“1”
0.0110	10011	←1 末位恒置“1”

$[x]_{补} = 1.0101$

$[y]_{补} = 0.1101$

$[-y]_{补} = 1.0011$

$\therefore [\frac{x}{y}]_{补} = 1.0011$

则 $\frac{x}{y} = -0.1101$