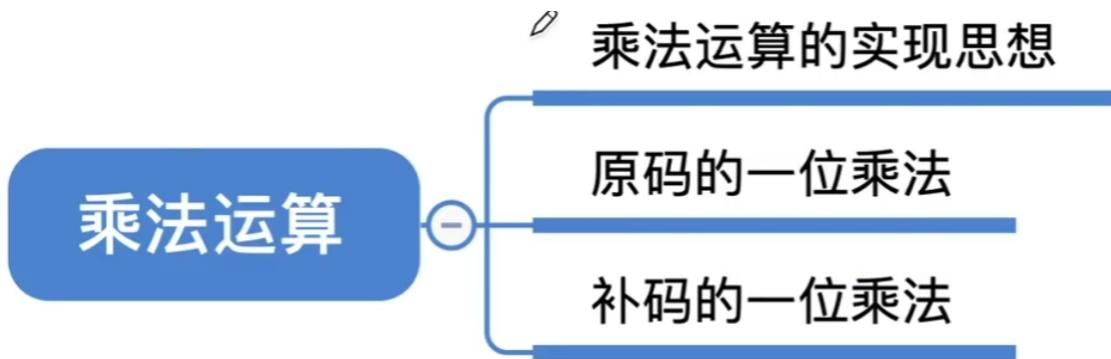


定点数原码的乘法运算



手算乘法 (十进制)

$$\begin{aligned} \text{r进制: } & K_n K_{n-1} \dots K_2 K_1 K_0 K_{-1} K_{-2} \dots K_{-m} \\ & = K_n \times r^n + K_{n-1} \times r^{n-1} + \dots + K_2 \times r^2 + K_1 \times r^1 + K_0 \times r^0 + K_{-1} \times r^{-1} + K_{-2} \times r^{-2} + \dots + K_{-m} \times r^{-m} \end{aligned}$$

$$0.211 = 2 * 10^{-1} + 1 * 10^{-2} + 1 * 10^{-3}$$

$$0.985 = 985 * 10^{-3}$$

$$0.985 * 0.211 = (985 * 1 * 10^{-6}) + (985 * 1 * 10^{-5}) + (985 * 2 * 10^{-4})$$

$$\begin{array}{r} 0.985 \\ \times 0.211 \\ \hline 985 \\ 985 \\ 1970 \\ \hline 0.207835 \end{array}$$



$$\begin{array}{r} 0.985 \\ \times 0.211 \\ \hline 0.000985 \\ 0.00985 \\ 0.1970 \\ \hline 0.207835 \end{array}$$

手算乘法 (二进制)

$$\begin{array}{r} 0.1101 \\ \times 0.1011 \\ \hline 1101 \\ 1101 \\ 0000 \\ 1101 \\ \hline 0.10001111 \end{array} \quad \rightarrow \quad \begin{array}{r} 0.1101 \\ \times 0.1011 \\ \hline 0.00001101 \\ 0.0001101 \\ 0.0000000 \\ 0.01101 \\ \hline 0.10001111 \end{array}$$

⑥

(乘数) $0.1011 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}$
(被乘数) $0.1101 = 1101 \times 2^{-4}$

用“移位”实现

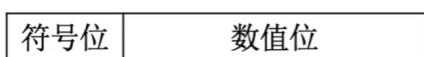
$$0.1101 \times 0.1011 = (1101 \times 1 \times 2^{-8}) + (1101 \times 1 \times 2^{-7}) + (1101 \times 0 \times 2^{-6}) + (1101 \times 1 \times 2^{-5})$$

考虑用机器实现：

- 实际数字有正负，符号位如何处理？
- 乘积的位数扩大一倍，如何处理？
- 4个位积都要保存下来最后统一相加？

原码一位乘法

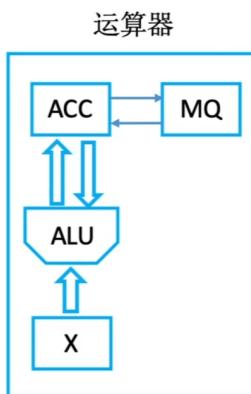
设机器字长为 $n+1=5$ 位（含1位符号位）， $[x]_{\text{原}} = 1.1101$, $[y]_{\text{原}} = 0.1011$, 采用原码一位乘法求 $x \cdot y$



符号单独处理：符号位 = $x_s \oplus y_s$

数值位取绝对值进行乘法计算

$[|x|]_{\text{原}} = 0.1101$, $[|y|]_{\text{原}} = 0.1011$



运算器：用于实现算术运算（如：加减乘除）、逻辑运算（如：与或非）

ACC：累加器，用于存放操作数，或运算结果。

MQ：乘商寄存器，在乘、除运算时，用于存放操作数或运算结果。

X：通用的操作数寄存器，用于存放操作数

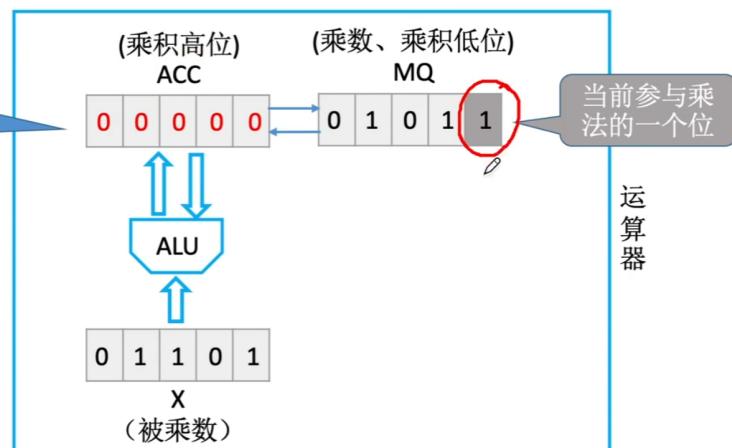
ALU：算术逻辑单元，通过内部复杂的电路实现算数运算、逻辑运算

	加	减	乘	除
Accumulator	ACC	被加数、和	被减数、差	乘积高位
Multiple-Quotient Register	MQ			被除数、余数
Arithmetic and Logic Unit	X	加数	减数	商
			乘数、乘积低位	除数
			被乘数	

实现方法：先加法再移位，重复n次

$$\begin{array}{r}
 & 0.1101 \\
 \times & 0.1011 \\
 \hline
 & 01101 \\
 & 01101 \\
 & 00000 \\
 & 01101 \\
 \hline
 & 0.10001111
 \end{array}$$

在正式进行乘法之前，ACC置0



当前位=1，则ACC加上被乘数

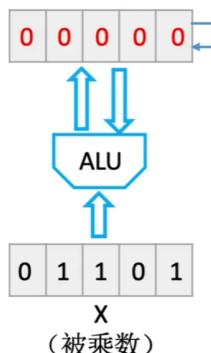
当前位=0，则ACC加上0

实现方法：先加法再移位，重复n次

$$\begin{array}{r} 0.1101 \\ \times 0.1011 \\ \hline 01101 \\ 01101 \\ 00000 \\ 01101 \\ \hline 0.10001111 \end{array}$$

在正式进行乘法之前，ACC置0

(乘积高位) ACC



当前参与乘法的一个位

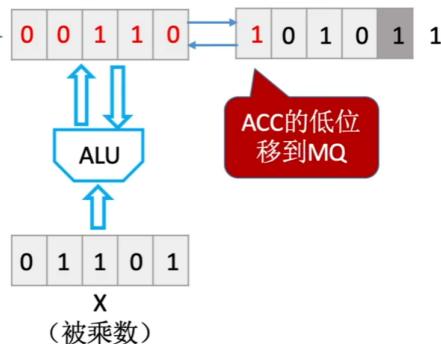
运算器

当前位=1，则ACC加上被乘数
当前位=0，则ACC加上0

$$\begin{array}{r} 0.1101 \\ \times 0.1011 \\ \hline 001101 \\ 01101 \\ 00000 \\ 01101 \\ \hline 0.10001111 \end{array}$$

逻辑右移，高位补0

(乘积高位) ACC



之后用不到了，直接丢弃

运算器

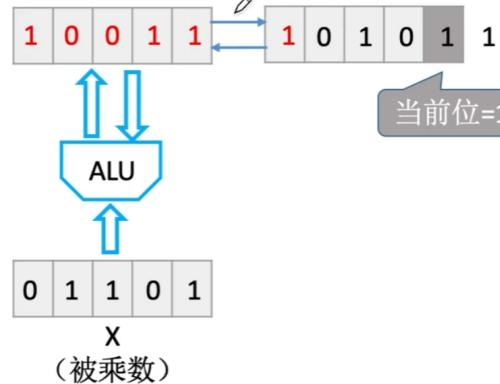
当前位=1，则ACC加上被乘数
当前位=0，则ACC加上0

$$\begin{array}{r} 0.1101 \\ \times 0.1011 \\ \hline 01101 \\ 01101 \\ 00000 \\ 01101 \\ \hline 0.10001111 \end{array}$$

(ACC)+(X)→ACC

$$00110 + 01101 = 10011$$

(乘积高位) ACC



当前位=1

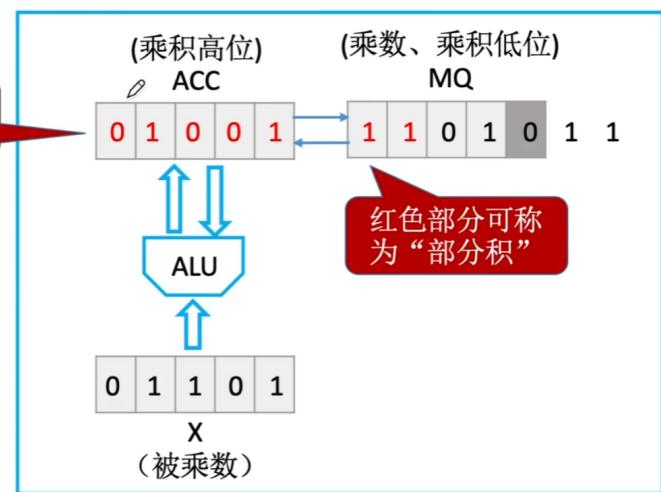
运算器

当前位=1，则ACC加上被乘数
当前位=0，则ACC加上0

实现方法：先加法再移位，重复n次

$$\begin{array}{r} 0.1101 \\ \times 0.1011 \\ \hline 01101 \\ 01101 \\ 00000 \\ 01101 \\ \hline 0.10001111 \end{array}$$

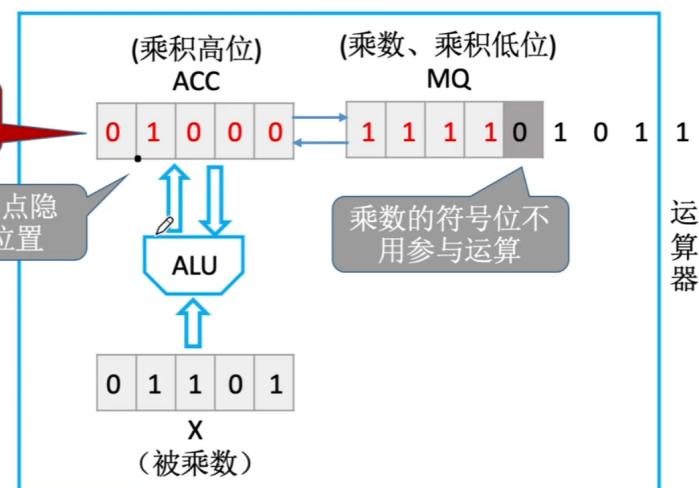
逻辑右移，高位补0



当前位=1，则ACC加上被乘数
当前位=0，则ACC加上0

$$\begin{array}{r} 0.1101 \\ \times 0.1011 \\ \hline 01101 \\ 01101 \\ 00000 \\ 01101 \\ \hline 0.10001111 \end{array}$$

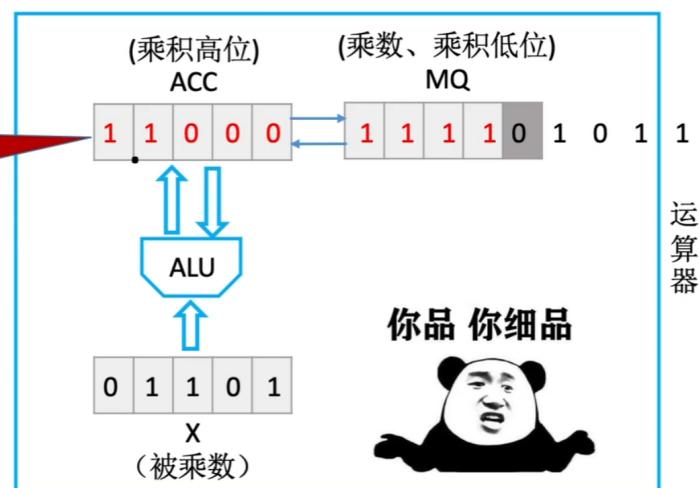
逻辑右移，高位补0



实现方法：先加法再移位，重复n次

$$\begin{array}{r} 0.1101 \\ \times 0.1011 \\ \hline 01101 \\ 01101 \\ 00000 \\ 01101 \\ \hline 0.10001111 \end{array}$$

修改符号位 $x_s \oplus y_s = 1$



原码一位乘法（手算模拟）

设机器字长为5位（含1位符号位， $n=4$ ）， $x = -0.1101$, $y = +0.1011$, 采用原码一位乘法求 $x \cdot y$



解: $|x| = 00.1101$, $|y| = 00.1011$, 原码一位乘法的求解过程如下。 MQ

通用寄存器	ACC	(高位部分积)	(低位部分积/乘数)	说明 起始情况 $C_4=1$, 则 $+ x $
	$+ x $	00.0000	1011	
		00.1101		
	右移	00.0110	1101	右移部分积和乘数 $C_4=1$, 则 $+ x $
	$+ x $	00.1101		
		01.0011		
	右移	00.1001	1110	右移部分积和乘数 $C_4=0$, 则 $+0$
	$+0$	00.0000		
		00.1001		
	右移	00.0100	1111	右移部分积和乘数 $C_4=1$, 则 $+ x $
	$+ x $	00.1101		
		01.0001		
	右移	00.1000	1111	右移部分积和乘数 乘数全部移出
				结果的绝对值部分

Tips:

- 乘数的符号位不参与运算，可以省略
- 原码一位乘可以只用单符号位
- 答题时最终结果最好写为原码机器数

原码一位乘法: 机器字长 $n+1$, 数值部分占 n 位

符号位通过异或确定; 数值部分通过被乘数和乘数绝对值的 n 轮加法、移位完成
根据当前乘数中参与运算的位确定(ACC)加什么。若当前运算位=1, 则 $(ACC)+[|x|]_{原}$;
若=0, 则 $(ACC)+0$ 。

每轮加法后ACC、MQ的内容统一逻辑右移

符号位 $P_s = x_s \oplus y_s = 1 \oplus 0 = 1$, 得 $x \cdot y = -0.10001111$ 。

定点数补码的乘法运算

设机器字长为5位（含1位符号位， $n=4$ ）， $x = -0.1101$, $y = +0.1011$, 采用Booth算法求 $x \cdot y$

$$[x]_{\text{补}} = 1.0011, [-x]_{\text{补}} = 0.1101, [y]_{\text{补}} = 0.1011$$

原码一位乘法:

进行 n 轮加法、移位

每次加法可能 $+0$ 、 $+|x|_{\text{原}}$

每次移位是“逻辑右移”



朋友, 过两招?

补码一位乘法:

进行 n 轮加法、移位, 最后再多来一次加法

每次加法可能 $+0$ 、 $+[x]_{\text{补}}$ 、 $+[-x]_{\text{补}}$

根据当前MQ中的最低位、辅助位 来确定加什么

辅助位 - MQ中最低位 = 1时, $(ACC)+[x]_{\text{补}}$

辅助位 - MQ中最低位 = 0时, $(ACC)+0$

辅助位 - MQ中最低位 = -1时, $(ACC)+[-x]_{\text{补}}$

补码一位乘法:

进行 n 轮加法、移位, 最后再多来一次加法

每次加法可能 $+0$ 、 $+[x]_{\text{补}}$ 、 $+[-x]_{\text{补}}$

每次移位是“补码的算数右移”

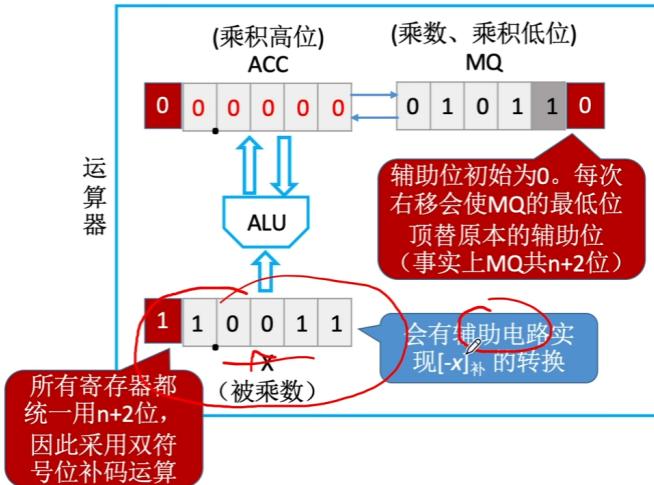
符号位参与运算

根据当前MQ中的最低位、辅助位 来确定加什么

辅助位 - MQ中最低位 = 1时, $(ACC)+[x]_{\text{补}}$

辅助位 - MQ中最低位 = 0时, $(ACC)+0$

辅助位 - MQ中最低位 = -1时, $(ACC)+[-x]_{\text{补}}$



设机器字长为5位（含1位符号位， $n=4$ ）， $x = -0.1101$, $y = +0.1011$, 采用Booth算法求 $x \cdot y$

$$[x]_{\text{补}} = 11.0011, [-x]_{\text{补}} = 00.1101, [y]_{\text{补}} = 0.1011$$

(高位部分积)	(低位部分积/乘数)	说明
00.0000	0.1011 0 丢失位	起始情况
+[-x]补	00.1101 1 2 3 0 辅助位	$Y_4 Y_5 = 10, Y_5 - Y_4 = -1$, 则 $+[x]_{\text{补}}$
00.1101		
右移	00.0110 ----- 10.101 10	右移部分积和乘数
+0	00.0000	$Y_4 Y_5 = 11, Y_5 - Y_4 = 0$, 则 $+0$
	00.0110	
右移	00.0011 ----- 010.10 110	右移部分积和乘数
+[x]补	11.0011 11.0110	$Y_4 Y_5 = 01, Y_5 - Y_4 = 1$, 则 $+[x]_{\text{补}}$
	11.0110	
右移	11.1011 ----- 0010.1 0110	右移部分积和乘数
+[-x]补	00.1101 00.1000	$Y_4 Y_5 = 10, Y_5 - Y_4 = -1$, 则 $+[x]_{\text{补}}$
	00.1000	
右移	00.0100 ----- 00010 10110	右移部分积和乘数
+[x]补	11.0011 11.0111	$Y_4 Y_5 = 01, Y_5 - Y_4 = 1$, 则 $+[x]_{\text{补}}$
	11.0111	构成 $[x \cdot y]_{\text{补}}$

n轮加法、算数右移，加法规则如下：
 辅助位 - MQ中最低位 = 1时, $(ACC)+[x]_{\text{补}}$
 辅助位 - MQ中最低位 = 0时, $(ACC)+0$
 辅助位 - MQ中最低位 = -1时, $(ACC)+[-x]_{\text{补}}$

补码的算数右移：
 符号位不动，数值位右移，正数右移补0，
 负数右移补1（符号位是啥就补啥）

$$[x \cdot y]_{\text{补}} = 11.01110001$$

即 $x \cdot y = -0.10001111$

原码一位乘法：

符号位通过异或确定，数值位由被乘数和乘数的绝对值进行 n 轮加法、移位

每次加法可能 $+0$ 、 $+[|x|]_{\text{原}}$



每次移位是“逻辑右移”

乘数的符号位不参与运算

朋友，过两招？

补码一位乘法（Booth算法）：

符号位、数值位都是由被乘数和乘数进行 n 轮加法、移位，最后再多来一次加法

每次加法可能 $+0$ 、 $+[x]_{\text{补}}$ 、 $+[x]_{\text{补}}$

每次移位是“补码的算数右移”

乘数的符号位参与运算

MQ中最低位 = 1时, $(ACC)+[|x|]_{\text{原}}$

MQ中最低位 = 0时, $(ACC)+0$

辅助位 - MQ中“最低位” = 1时, $(ACC)+[x]_{\text{补}}$

辅助位 - MQ中“最低位” = 0时, $(ACC)+0$

辅助位 - MQ中“最低位” = -1时, $(ACC)+[-x]_{\text{补}}$

定点数原码的除法运算

手算除法（十进制）

$$0.211 / 0.985 = ?$$

$$r \text{ 进制: } K_n K_{n-1} \dots K_2 K_1 K_0 K_{-1} K_{-2} \dots K_{-m}$$

$$= K_n \times r^n + K_{n-1} \times r^{n-1} + \dots + K_2 \times r^2 + K_1 \times r^1 + K_0 \times r^0 + K_{-1} \times r^{-1} + K_{-2} \times r^{-2} + \dots + K_{-m} \times r^{-m}$$

$$0.211 \div 0.985 = ?$$

你怎么这个亚子



$$\begin{array}{r} 0.214 \\ 985 \overline{)211} \\ \underline{-000} \\ 2110 \\ \underline{-1970} \\ 1400 \\ \underline{-985} \\ 4150 \\ \underline{-3940} \\ 210 \end{array}$$

$$\begin{array}{r} 0.214 \\ 985 \overline{)0.211} \\ \underline{-0.000} \\ 0.2110 \\ \underline{-0.1970} \\ 0.01400 \\ \underline{-0.00985} \\ 0.004150 \\ \underline{-0.003940} \\ 0.000210 \end{array}$$

心情复杂



$$0.214 = 2 \times 10^{-1} + 1 \times 10^{-2} + 4 \times 10^{-3}$$

$$0.985 = 985 \times 10^{-3}$$

$$0.985 \times 0.214 = (985 \times 2 \times 10^{-4}) + (985 \times 1 \times 10^{-5}) + (985 \times 4 \times 10^{-6})$$

$$= 0.1970 + 0.00985 + 0.00394$$

$$x \div y = a \text{ (余数 } b \text{)} \rightarrow x = ay + b$$

$$0.211 = 0.985 \times 0.214 + 0.000210$$

手算除法 (二进制)

符号位	绝对值
-----	-----

两个正数相除

设机器字长为5位 (含1位符号位, $n=4$) , $x=0.1011$, $y=0.1101$, 求 x/y

$$(0.1011 \times 2^4) \div (0.1101 \times 2^4)$$

$$\begin{array}{r} 0.1101 \\ 01101 \sqrt{01011} \\ 00000 \\ \hline 10110 \\ 01101 \\ \hline 10010 \\ 01101 \\ \hline 01010 \\ 00000 \\ \hline 10100 \\ 01101 \\ \hline 0111 \end{array}$$

$$\begin{array}{r} 0.1101 \\ 0.1101 \sqrt{0.1011} \\ 0.0000 \\ \hline 0.10110 \\ 0.01101 \\ \hline 0.010010 \\ 0.001101 \\ \hline 0.0001010 \\ 0.0000000 \\ \hline 0.00010100 \\ 0.00001101 \\ \hline 0.00000111 \end{array}$$

规律: 忽略小数点, 每确定一位商, 进行一次减法, 得到4位余数, 在余数末尾补0, 再确定下一位商。确定5位商即可停止 (机器字长为5位)

x/y 结果为0.1101, 余数为0.00000111

原码除法: 恢复余数法

设机器字长为5位 (含1位符号位, $n=4$) , $x=0.1011$, $y=0.1101$, 采用原码恢复余数法求 x/y

$$|x|=0.1011, |y|=0.1101, [y]_{\text{补}}=0.1101, [-y]_{\text{补}}=1.0011$$

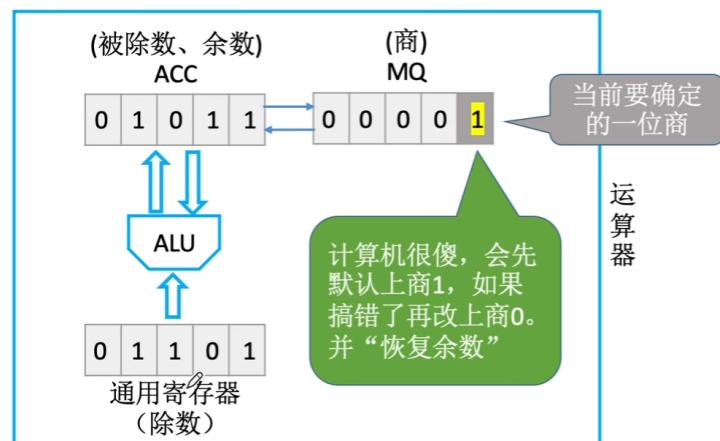
$$\text{符号单独处理: 符号位 } = x_s \oplus y_s$$

实现方法: 上商0/1, 得到余数, 余数末尾补0

数值位取绝对值进行除法计算

$$\begin{array}{r} 0.1101 \\ 01101 \sqrt{01011} \\ 00000 \\ \hline 10110 \\ 01101 \\ \hline 10010 \\ 01101 \\ \hline 01010 \\ 00000 \\ \hline 10100 \\ 01101 \\ \hline 0111 \end{array}$$

计算机很傻, 会先默认上商1, 如果搞错了再改上商0。并“恢复余数”



设机器字长为5位（含1位符号位， $n=4$ ）， $x=0.1011$, $y=0.1101$, 采用原码恢复余数法求 x/y

$$|x|=0.1011, |y|=0.1101, [y]_{\text{补}}=0.1101, [-y]_{\text{补}}=1.0011$$

符号单独处理：符号位 = $x_s \oplus y_s$

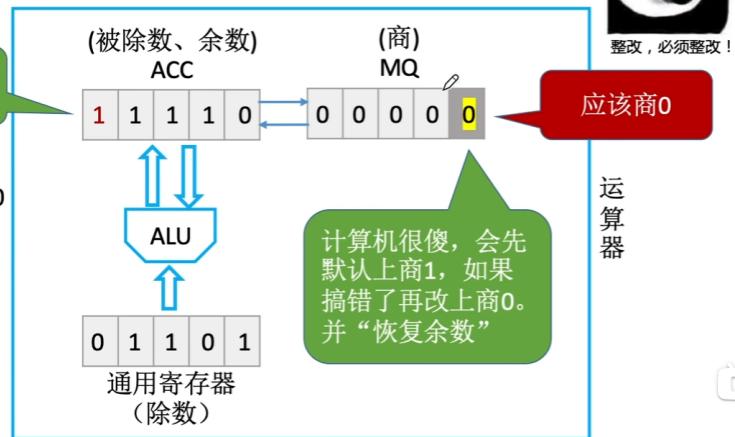
实现方法：上商0/1，得到余数，余数末尾补0

	0.1101
01101	/ 01011 00000 10110 01101
	10010 01101 01010 00000 10100 01101 0111

恢复余数：
 $(ACC) + (\text{除数}) \rightarrow ACC$

$$(ACC) + [-y]_{\text{补}} \rightarrow ACC$$

$$01011 + 10011 = 11110$$



计算机很傻，会先默认上商1，如果搞错了再改上商0。并“恢复余数”

设机器字长为5位（含1位符号位， $n=4$ ）， $x=0.1011$, $y=0.1101$, 采用原码恢复余数法求 x/y

$$|x|=0.1011, |y|=0.1101, [y]_{\text{补}}=0.1101, [-y]_{\text{补}}=1.0011$$

符号单独处理：符号位 = $x_s \oplus y_s$

实现方法：上商0/1，得到余数，余数末尾补0

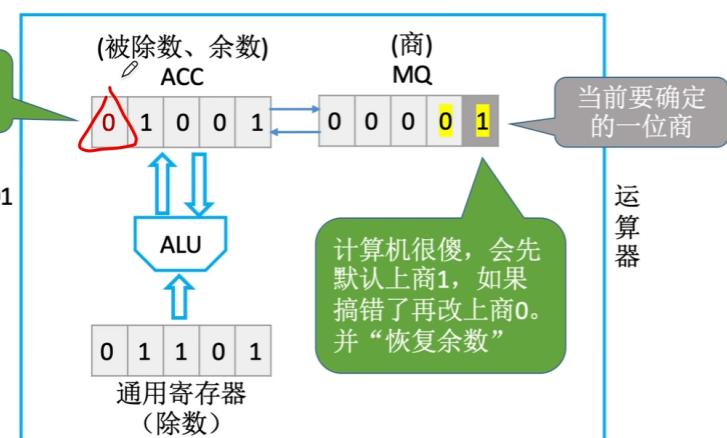
	0.1101
01101	/ 01011 00000 10110 01101
	10010 01101 01010 00000 10100 01101 0111

求余数：
 $(ACC) - (\text{除数}) \rightarrow ACC$

$$(ACC) + [-y]_{\text{补}} \rightarrow ACC$$

$$10110 + 10011 = 01001$$

相减结果
是个正数，
上商1是
没错滴~



计算机很傻，会先默认上商1，如果搞错了再改上商0。并“恢复余数”

设机器字长为5位（含1位符号位， $n=4$ ）， $x=0.1011$, $y=0.1101$, 采用原码恢复余数法求 x/y

$$|x|=0.1011, |y|=0.1101, [y]_{\text{补}}=0.1101, [-y]_{\text{补}}=1.0011$$

符号单独处理：符号位 = $x_s \oplus y_s$

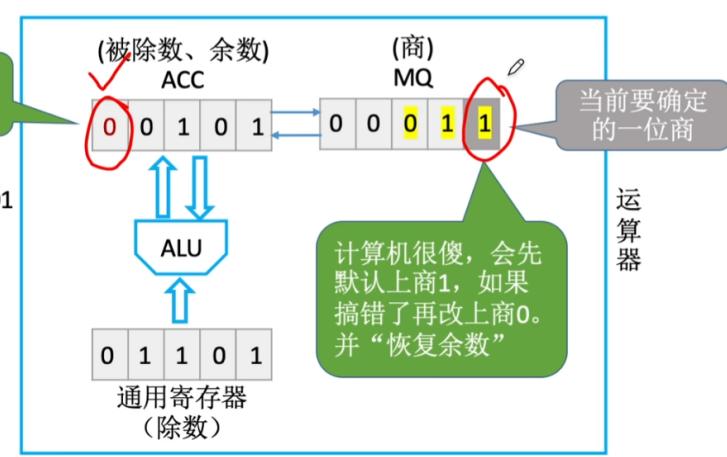
实现方法：上商0/1，得到余数，余数末尾补0

	0.1101
01101	/ 01011 00000 10110 01101
	10010 01101 01010 00000 10100 01101 0111

求余数：
 $(ACC) - (\text{除数}) \rightarrow ACC$

$$(ACC) + [-y]_{\text{补}} \rightarrow ACC$$

$$10010 + 10011 = 00101$$



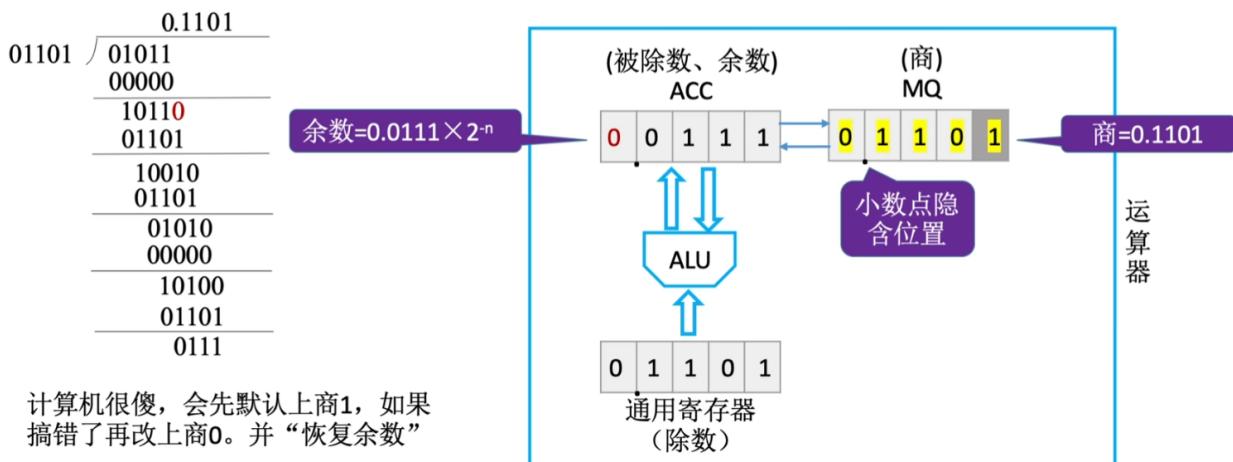
计算机很傻，会先默认上商1，如果搞错了再改上商0。并“恢复余数”

设机器字长为5位（含1位符号位， $n=4$ ）， $x=0.1011$, $y=0.1101$, 采用原码恢复余数法求 x/y

$|x|=0.1011$, $|y|=0.1101$, $[|y|]_{\text{补}}=0.1101$, $[-|y|]_{\text{补}}=1.0011$ 符号单独处理：符号位 = $x_s \oplus y_s$

实现方法：上商0/1，得到余数，余数末尾补0

数值位取绝对值进行除法计算



恢复余数法（手算）

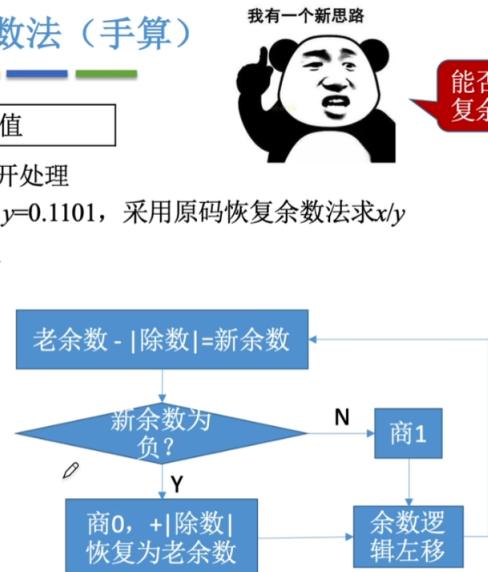
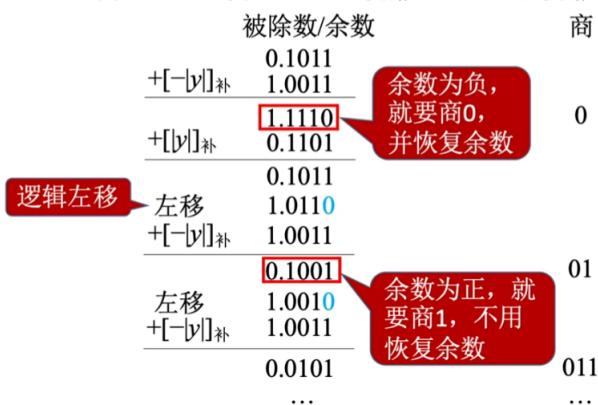
原码除法：恢复余数法（手算）

我有一个新思路

能否不恢复余数？

设机器字长为5位（含1位符号位， $n=4$ ）， $x=0.1011$, $y=0.1101$, 采用原码恢复余数法求 x/y

$|x|=0.1011$, $|y|=0.1101$, $[|y|]_{\text{补}}=0.1101$, $[-|y|]_{\text{补}}=1.0011$



原码除法：恢复余数法（手算）

我有一个新思路



能否不恢
复余数？

符号位	绝对值
-----	-----

符号位与数值位分开处理

设机器字长为5位（含1位符号位， $n=4$ ）， $x=0.1011$, $y=0.1101$, 采用原码恢复余数法求 x/y

$$|x|=0.1011, |y|=0.1101, [y]_{\text{补}}=0.1101, [-y]_{\text{补}}=1.0011$$

被除数/余数	商
0.1011	
$+[-y]_{\text{补}}$	
1.0011	
$\underline{+ [y]_{\text{补}}}$	
1.1110	a
$\underline{+ [y]_{\text{补}}}$	b
0.1101	$a+b$
$\underline{\text{左移}}$	$(a+b) \times 2 = 2a + 2b$
1.0110	
$+[-y]_{\text{补}}$	$(a+b) \times 2 - b = 2a + 2b - b = 2a + b$
1.0011	
0.1001	0
$\underline{\text{左移}}$	
1.0010	
$+[-y]_{\text{补}}$	
1.0011	
0.0101	01
\dots	\dots

余数a为负

若余数为负，则可直
接商0，并让余数左
移1位再加上|除数|

原码除法：加减交替法

又名：不恢复余数法

符号位与数值位分开处理

设机器字长为5位（含1位符号位， $n=4$ ）， $x=0.1011$, $y=0.1101$, 采用原码加减交替除法求 x/y

$$|x|=0.1011, |y|=0.1101, [y]_{\text{补}}=0.1101, [-y]_{\text{补}}=1.0011$$

若余数为负，
则可直接商
0，让余数
左移1位再
加上|除数|，
得到下一个
新余数

若余数为正，
则商1，让
余数左移1
位再减去
|除数|，得
到下一个新
余数

被除数/余数	商	ACC	MQ
0.1011		01011	00000
$+[-y]_{\text{补}}$			
1.0011			
$\underline{+ [y]_{\text{补}}}$			
1.1110	0	11110	00000
$\underline{\text{左移}}$		11100	00000
1.1100			
$+[-y]_{\text{补}}$			
0.1101			
$\underline{+ [y]_{\text{补}}}$			
0.1001	01	01001	00001
$\underline{\text{左移}}$		10010	00010
1.0010			
$+[-y]_{\text{补}}$			
1.0011			
0.0101	011	00101	00011
$\underline{\text{左移}}$		01010	00110
0.1010			
$+[-y]_{\text{补}}$			
1.0011			
0.0101	0110	11101	00110
$\underline{\text{左移}}$		11010	01100
1.1010			
$+[-y]_{\text{补}}$			
0.1101	01101	00111	01101
$\underline{+ [y]_{\text{补}}}$			
0.0111			

恢复余数法：当余数为负时商0，
并 $+|$ 除数 $|$ ，再左移，再 $-|$ 除数 $|$

加减交替法：当余数为负时商0，
并左移，再 $+|$ 除数 $|$

原码除法：加减交替法

又名：不恢复余数法

符号位与数值位分开处理

设机器字长为5位（含1位符号位， $n=4$ ）， $x=0.1011$, $y=0.1101$, 采用原码加减交替法求 x/y

$$|x|=0.1011, |y|=0.1101, [y]_{\text{补}}=0.1101, [-y]_{\text{补}}=1.0011$$

若余数为负，则可直接商0，让余数左移1位再加上|除数|，得到下一个新余数

若余数为正，则商1，让余数左移1位再减去|除数|，得到下一个新余数

被除数/余数	商	ACC	MQ
0.1011	0	01011	00000
+[-y]_{\text{补}}			
1.0011			
1.1110	0	11110	00000
左移			
1.1100			
+[y]_{\text{补}}			
0.1101			
0.1001	01	01001	00001
左移			
1.0010			
+[y]_{\text{补}}			
1.0011			
0.0101			
0.1010	011	00101	00011
左移			
1.0100			
+[y]_{\text{补}}			
1.0011			
1.1101	0110	01010	00110
左移			
1.1010			
+[y]_{\text{补}}			
0.1101			
0.0111	01101	11101	00110
0.0111			

$$Q_s = x_s \oplus y_s = 0 \oplus 0 = 0$$

得 $x/y=+0.1101$

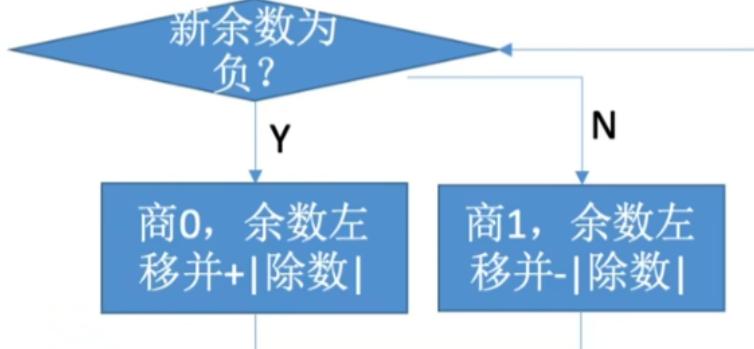
余 0.0111×2^{-4}

注：余数的正负性与商相同

恢复余数法：当余数为负时商0，并 $+|$ 除数|，再左移，再 $-|$ 除数|

加减交替法：当余数为负时商0，并左移，再 $+|$ 除数|

被除数-|除数|=新余数



加/减n+1次，每次加减确定一位商；
左移n次（最后一次加减完不移位）
最终可能还要再多一次加

定点数补码的除法运算

补码除法：加减交替法

设机器字长为5位（含1位符号位， $n=4$ ）， $x=+0.1000$, $y=-0.1011$, 采用补码加减交替除法求 x/y

$[x]_{\text{补}}=00.1000$, $[y]_{\text{补}}=11.0101$, $[-y]_{\text{补}}=00.1011$ $\frac{[x/y]_{\text{补}}}{[y]_{\text{补}}} = 1.0101$, 余数 0.0111×2^{-4}

被除数/余数		ACC	MQ
00.1000		001000	00000
+ $[y]_{\text{补}}$	11.0101	111101	00001
11.1101		111010	00010
左移	11.1010	000101	00010
+ $[-y]_{\text{补}}$	00.1011	001010	00100
00.0101		111111	00101
左移	00.1010	111110	01010
+ $[y]_{\text{补}}$	11.0101	001001	01010
11.1111		010010	10100
左移	11.1110	000111	10101
+ $[-y]_{\text{补}}$	00.1011		
00.1001			
左移	01.00010		
+ $[y]_{\text{补}}$	11.0101		
00.0111			

补码除法：

- 符号位参与运算
- 被除数/余数、除数采用双符号位

被除数和除数同号，则被除数减去除数；
异号则被除数加上除数。

余数和除数同号，商1，余数左移一位减去除数；
余数和除数异号，商0，余数左移一位加上除数。
重复n次

精度误差
不超过 2^{-n}

末位商恒置1

除法类型	符号位参与运算	加减次数	移位		上商、加减原则	说明
			方 向	次 数		
原码加减交替法	否	N+1或N+2	左	N	余数的正负	若最终余数为负，需恢复余数
补码加减交替法	是	N+1	左	N	余数和除数是否同号	商末位恒置1