

计算机组成原理

文杰

计算机科学与技术学院

wenjie@hit.edu.cn

个人主页: <http://faculty.hitsz.edu.cn/wenjie>

第二章 计算机的运算方法

- 计算机中数的表示
- 定点运算与浮点运算

定点运算与浮点运算

- 移位运算：算数移位与逻辑移位
- 定点运算
 - 加减法运算
 - 一位乘法运算
 - booth算法
 - 除法运算
- 浮点运算

移位运算

- 移位的意义

$$15.\text{m} = 1500.\text{cm}$$

小数点右移 2 位

机器用语 15 相对于小数点 左移 2 位
(小数点不动)

左移 绝对值扩大

右移 绝对值缩小

- 在计算机中，移位与加减配合，能够实现乘除运算

算术移位规则

- 符号位不变
- 机器数的移位 与 真值移位 一致

真值	码 制	添补代码
正数	原码、补码、反码	0
负数	原 码	0
	补 码	左移 添 0
		右移 添 1
	反 码	1

$$y = +1 y_1 y_2 \dots y_n 100 \dots 00$$

$$y = -1 y_1 y_2 \dots y_n 100 \dots 00$$

$$[y]_{\text{原}} = 1, 1 y_1 y_2 \dots y_n 100 \dots 00$$

$$[y]_{\text{反}} = 1, 0 \overline{y_1} \overline{y_2} \dots \overline{y_n} 011 \dots 11$$

$$\begin{aligned} [y]_{\text{补}} &= 1, 0 \overline{y_1} \overline{y_2} \dots \overline{y_n} 011 \dots 11 \text{ 的末位} + 1 \\ &= 1, 0 \overline{y_1} \overline{y_2} \dots \overline{y_n} 100 \dots 00 \end{aligned}$$

- 例17. 设机器数字长为 8 位（含 1 位符号位），写出 $A = +26$ 时，三种机器数左、右移一位和两位后的表示形式及对应的真值，并分析结果的正确性。

解： $A = +26 = +11010$

则 $[A]_{\text{原}} = [A]_{\text{补}} = [A]_{\text{反}} = 0,0011010$

移位操作	机 器 数	对应的真值
	$[A]_{\text{原}} = [A]_{\text{补}} = [A]_{\text{反}}$	
移位前	0,0011010	+26
左移一位	0,0110100	+52
左移两位	0,1101000	+104
右移一位	0,0001101	+13
右移两位	0,0000110	+6

→ 丢 1 影响精度

- 例18. 设机器数字长为 8 位（含 1 位符号位），写出 $A = -26$ 时，三种机器数左、右移一位和两位后的表示形式及对应的真值，并分析结果的正确性。

解： $A = -26 = -11010$

原码

移位操作	机 器 数	对应的真值
移位前	1,0011010	- 26
左移一位	1,0110100	- 52
左移两位	1,1101000	- 104
右移一位	1,0001101	- 13
右移两位	1,0000110	- 6

→ 丢 1 影响精度

补码

移位操作	机 器 数	对应的真值
移位前	1,1100110	– 26
左移一位	1,1001100	– 52
左移两位	1,0011000	– 104
右移一位	1,1110011	– 13
右移两位	1,1111001	– 7

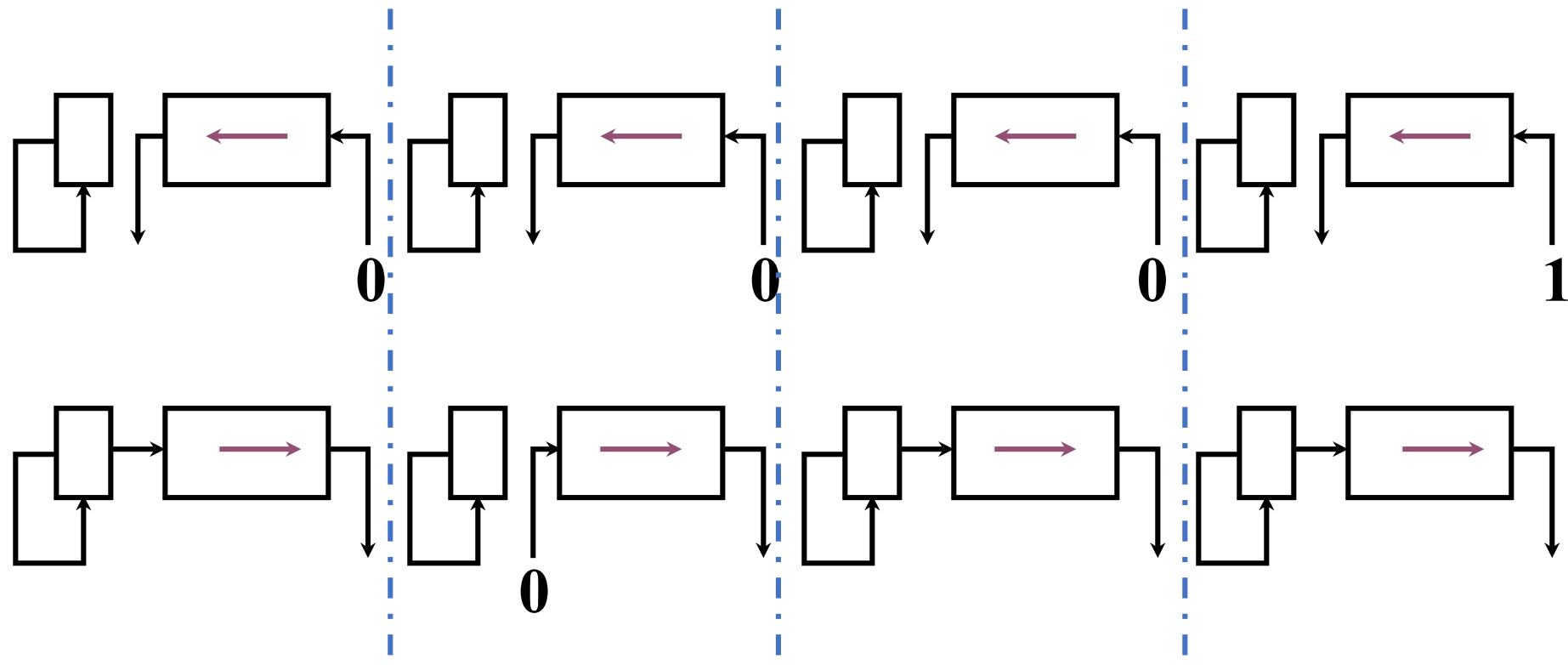
→ 丢 1 影响精度

反码

移位操作	机 器 数	对应的真值
移位前	1,1100101	– 26
左移一位	1,1001011	– 52
左移两位	1,0010111	– 104
右移一位	1,1110010	– 13
右移两位	1,1111001	– 6

→ 丢 0 影响精度

3. 算术移位的硬件实现



(a) 真值为正	(b) 负数的原码	(c) 负数的补码	(d) 负数的反码
← 丢 1 出错	出错	正确	正确
→ 丢 1 影响精度	影响精度	影响精度	正确

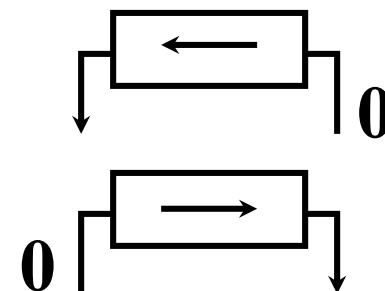
算术移位和逻辑移位的区别

算术移位 有符号数的移位

逻辑移位 无符号数的移位

逻辑左移 低位添 0，高位移丢

逻辑右移 高位添 0，低位移丢



例如 01010011

10110010

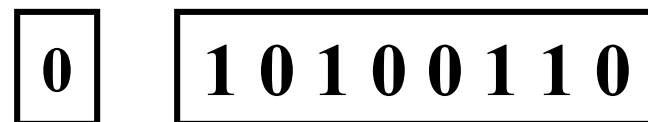
逻辑左移 10100110

逻辑右移 01011001

算术左移 00100110

算术右移 11011001 (补码)

高位 1 移丢



定点运算与浮点运算

- 移位运算：算数移位与逻辑移位
- 定点运算
 - 加减法运算
 - 一位乘法运算
 - booth算法
 - 除法运算
- 浮点运算

加減法运算

- 补码加減运算公式

(1) 加法

整数 $[A]_{\text{补}} + [B]_{\text{补}} = [A+B]_{\text{补}} \pmod{2^{n+1}}$

小数 $[A]_{\text{补}} + [B]_{\text{补}} = [A+B]_{\text{补}} \pmod{2}$

(2) 減法

$$A-B = A+(-B)$$

整数 $[A-B]_{\text{补}} = [A+(-B)]_{\text{补}} = [A]_{\text{补}} + [-B]_{\text{补}} \pmod{2^{n+1}}$

小数 $[A-B]_{\text{补}} = [A+(-B)]_{\text{补}} = [A]_{\text{补}} + [-B]_{\text{补}} \pmod{2}$

连同符号位一起相加，符号位产生的进位自然丢掉

$$[x]_{\text{补}} = \begin{cases} 0, x & 2^n > x \geq 0 \\ 2^{n+1} + x & 0 > x \geq -2^n \pmod{2^{n+1}} \end{cases}$$

$$[x]_{\text{补}} = \begin{cases} x & 1 > x \geq 0 \\ 2 + x & 0 > x \geq -1 \pmod{2} \end{cases}$$

例：用补码运算求 $A+B$

- 设 $A = 0.1011$, $B = -0.0101$, 用补码运算求 $A+B = ?$

解： $[A]_{\text{补}} = 0.1011$

验证

$$+[B]_{\text{补}} = 1.1011$$

$$\begin{array}{r} 0.1011 \\ - 0.0101 \\ \hline 0.0110 \end{array}$$

$$[A]_{\text{补}} + [B]_{\text{补}} = \cancel{1}0.0110 = [A+B]_{\text{补}}$$

$$\therefore A+B = 0.0110$$

- 设 $A = -9$, $B = -5$ (设A和B数值位数为4), 求 $A+B$

解： $[A]_{\text{补}} = 1, 0111$

验证

$$+[B]_{\text{补}} = 1, 1011$$

$$\begin{array}{r} -1001 \\ + -0101 \\ \hline -1110 \end{array}$$

$$[A]_{\text{补}} + [B]_{\text{补}} = \cancel{1}1, 0010 = [A+B]_{\text{补}}$$

$$\therefore A+B = -1110$$

已知 $[X]_{\text{补}}$ ，以下哪个方法可以求出 $[-X]_{\text{补}}$ ？

- ☐ A 包括符号为在内按位取反
- ☒ B 包括符号位在内，每位取反，末位加 1
- ☐ C 除符号位外，每位取反，末位加 1
- ☐ D 符号为取反，其他保持不变

提交

关于 $[X]_{\text{补}}$ 和 $[-X]_{\text{补}}$ ，哪个表述正确？

- ☒ A $-[X]_{\text{补}} = +[-X]_{\text{补}}$
- ☐ B $[X]_{\text{补}} = [-X]_{\text{补}}$
- ☐ C $[X]_{\text{补}}$ 和 $[-X]_{\text{补}}$ 没有任何关系

提交

*例：已知小数 $[y]_{\text{补}}$ ，求 $[-y]_{\text{补}}$

设 $[y]_{\text{补}} = y_0.y_1y_2 \dots y_n$ ， $y \neq -1.0$ ，根据符号位 y_0 为0或1分开讨论

<I> $[y]_{\text{补}} = 0.y_1y_2 \dots y_n$

$$[y]_{\text{原}} = 0.y_1y_2 \dots y_n$$

$$-y = -0.y_1y_2 \dots y_n$$

$$[-y]_{\text{补}} = 1.\overline{y_1}\overline{y_2} \dots \overline{y_n} + 2^{-n}$$

<II> $[y]_{\text{补}} = 1.y_1y_2 \dots y_n$

$$[y]_{\text{原}} = 1 + (0.\overline{y_1}\overline{y_2} \dots \overline{y_n} + 2^{-n})$$

$$y = -(0.\overline{y_1}\overline{y_2} \dots \overline{y_n} + 2^{-n})$$

$$[-y]_{\text{补}} = 0.\overline{y_1}\overline{y_2} \dots \overline{y_n} + 2^{-n}$$

- $[y]_{\text{补}}$ 连同符号位在内，每位取反，末位加1，即得 $[-y]_{\text{补}}$
- 观察发现 $[-y]_{\text{补}} + [y]_{\text{补}} = 2 \rightarrow 0 \pmod{2}$

$$-[y]_{\text{补}} = +[-y]_{\text{补}}$$

*例：已知整数 $[X]_{\text{补}}$ ，求 $[-X]_{\text{补}}$

设 $[X]_{\text{补}} = X_n, X_{n-1} \dots X_1 X_0$ ，根据符号位 X_n 为0或1分开讨论

<I>

$$[X]_{\text{补}} = 0, X_{n-1} X_{n-2} \dots X_1 X_0$$

$$X_{\text{原}} = 0, X_{n-1} X_{n-2} \dots X_1 X_0$$

$$[-X]_{\text{原}} = 1, X_{n-1} X_{n-2} \dots X_1 X_0$$

$$[-X]_{\text{补}} = 1, (\bar{X}_{n-1} \bar{X}_{n-2} \dots \bar{X}_1 \bar{X}_0 + 1)$$

<II>

$$[X]_{\text{补}} = 1, X_{n-1} X_{n-2} \dots X_1 X_0$$

$$X_{\text{原}} = 1, (\bar{X}_{n-1} \bar{X}_{n-2} \dots \bar{X}_1 \bar{X}_0 + 1)$$

$$[-X]_{\text{原}} = 0, (\bar{X}_{n-1} \bar{X}_{n-2} \dots \bar{X}_1 \bar{X}_0 + 1)$$

$$[-X]_{\text{补}} = 0, (\bar{X}_{n-1} \bar{X}_{n-2} \dots \bar{X}_1 \bar{X}_0 + 1)$$

$[X]_{\text{补}}$ 连同符号位在内，每位取反，末位加1，即得 $[-X]_{\text{补}}$

观察发现 $[-X]_{\text{补}} + [X]_{\text{补}} = 2^{n+1} \rightarrow 0 \pmod{2^{n+1}}$ $+ [-X]_{\text{补}} = -[X]_{\text{补}}$

例：用补码运算求 $A-B$

设机器数字长为 8 位（含 1 位符号位）且 $A = 15$, $B = 24$,
用补码求 $A - B$ 。

$$[A - B]_{\text{补}} = [A]_{\text{补}} + [-B]_{\text{补}}$$

解： $A = 15 = 0001111$

$$B = 24 = 0011000$$

$$[A]_{\text{补}} = 0, 0001111 \quad [B]_{\text{补}} = 0, 0011000$$

$$+[-B]_{\text{补}} = 1, 1101000$$

$$[A]_{\text{补}} + [-B]_{\text{补}} = 1, 1110111 = [A - B]_{\text{补}}$$

$$\therefore A - B = -1001 = -9$$

设机器数字长为 8 位（含 1 位符号位）且 $A = -97$ ， $B = 41$ ，用补码求 $A - B$ （8 位补码表示）。（ $97 = 01100001$ $41 = 00101001$ ）

- ☐ A 11110110
- ☐ B 01110110
- ☐ C 11001000
- ☐ D 其他

例：用补码运算求 $A-B$

- 设机器数字长为 8 位（含 1 位符号位）且 $A = -97$, $B = 41$, 用补码求 $A-B$ 。（ $97 = 01100001$ $41 = 00101001$ ）

解： $A = -97 = -1100001$ $[A-B]_{\text{补}} = [A]_{\text{补}} + [-B]_{\text{补}}$

$$[A]_{\text{补}} = 1, 0011111$$

$$+[-B]_{\text{补}} = 1, 1010111$$

$$[A]_{\text{补}} + [-B]_{\text{补}} = \mathbf{10}, 1110110 = [A-B]_{\text{补}}$$

$$\therefore A-B = +118 \quad (\text{溢出})$$

练习 1 设 $x = \frac{9}{16}$ $y = \frac{11}{16}$, 用补码求 $x+y$

作答

练习 1 设 $x = \frac{9}{16}$ $y = \frac{11}{16}$, 用补码求 $x+y$

解: $x = 9/16 = 0.1001$

$$y = 11/16 = 0.1011$$

$$[x]_{\text{补}} = 0.1001$$

$$[y]_{\text{补}} = 0.1011$$

$$+ [y]_{\text{补}} = 0.1011$$

$$[x]_{\text{补}} + [y]_{\text{补}} = 1.0100 = [x+y]_{\text{补}}$$

$$x + y = -0.1100 = -\frac{12}{16}$$

错

一位符号位判溢出

- 一位符号位判溢出： $A+B$
 - 加法 $[A+B]_{\text{补}}$ ：若 $[A]_{\text{补}}$ 和 $[B]_{\text{补}}$ 两个数符号相同，而其结果的符号与 $[A]_{\text{补}}$ 和 $[B]_{\text{补}}$ 的符号不同，即为溢出
 - 减法 $[A-B]_{\text{补}}=[A+(-B)]_{\text{补}}$ ：若 $[A]_{\text{补}}$ 和 $[-B]_{\text{补}}$ 两个数符号相同，而其结果的符号与 $[A]_{\text{补}}$ 和 $[-B]_{\text{补}}$ 的符号不同，即为溢出
- 硬件实现
 - 最高有效位的进位 \oplus 符号位的进位 = 1，溢出

如

$1 \oplus 0 = 1$	} 有 溢出
$0 \oplus 1 = 1$	
$0 \oplus 0 = 0$	} 无 溢出
$1 \oplus 1 = 0$	

$$\begin{array}{r} [A]_{\text{补}} = 1, 001111 \\ + [-B]_{\text{补}} = 1, 101011 \\ \hline [A]_{\text{补}} + [-B]_{\text{补}} = 10, 1110110 = [A-B]_{\text{补}} \\ \therefore A-B = +118 \quad (\text{溢出}) \end{array}$$

两位符号位判溢出

- 小数变形补码

$$[x]_{\text{补}} = \begin{cases} x & 1 > x \geq 0 \\ 4 + x & 0 > x \geq -1 \pmod{4} \end{cases}$$

- 整数变形补码

$$[X]_{\text{补}} = \begin{cases} X & 2^n > X \geq 0 \\ 2^{n+2} + X & 0 > x \geq -2^n \pmod{2^{n+2}} \end{cases}$$

最高符号位 代表其 真正的符号

$$[x]_{\text{补}} + [y]_{\text{补}} = [x + y]_{\text{补}} \pmod{4 \text{ 或 } 2^{n+2}}$$

$$[x - y]_{\text{补}} = [x]_{\text{补}} + [-y]_{\text{补}} \pmod{4 \text{ 或 } 2^{n+2}}$$

结果的双符号位 相同	未溢出	00, ×××××	00. ×××××
		11, ×××××	11. ×××××

结果的双符号位 不同	溢出	10, ×××××	10. ×××××
		01, ×××××	01. ×××××

• 例. 设 $A = +\frac{11}{16} = 0.1011$, $B = +\frac{3}{16} = 0.0011$, 求 $[A+B]_{\text{补}}$

$$\begin{array}{rcl}
 \text{解: } [A]_{\text{补}} & = & 00.1011 \\
 + [B]_{\text{补}} & = & 00.0011 \\
 \hline
 [A]_{\text{补}} + [B]_{\text{补}} & = & 00.1110 = [A+B]_{\text{补}}
 \end{array}$$

结果的双符号位相同，未溢出

• 例. 设 $A = -\frac{11}{16} = -0.1011$, $B = -\frac{3}{16} = -0.0011$, 求 $[A+B]_{\text{补}}$

$$\begin{array}{rcl}
 \text{解: } [A]_{\text{补}} & = & 11.0101 \\
 + [B]_{\text{补}} & = & 11.1101 \\
 \hline
 [A]_{\text{补}} + [B]_{\text{补}} & = & 111.0010 = [A+B]_{\text{补}}
 \end{array}$$

丢掉

结果的双符号位相同，未溢出

• 例. 设 $A = +\frac{11}{16} = 0.1011$, $B = +\frac{7}{16} = 0.0111$, 求 $[A+B]_{\text{补}}$

$$\text{解: } [A]_{\text{补}} = 00.1011$$

$$+ [B]_{\text{补}} = 00.0111$$

$$\hline [A]_{\text{补}} + [B]_{\text{补}} = 01.0010 = [A+B]_{\text{补}}$$

第1位符号位

结果的双符号位不同, 溢出

• 例. 设 $A = -\frac{11}{16} = -0.1011$, $B = -\frac{7}{16} = -0.0111$, 求 $[A+B]_{\text{补}}$

$$\text{解: } [A]_{\text{补}} = 11.0101$$

$$+ [B]_{\text{补}} = 11.1001$$

$$\hline [A]_{\text{补}} + [B]_{\text{补}} = 110.1110 = [A+B]_{\text{补}}$$

丢掉

结果的双符号位不同, 溢出

定点运算与浮点运算

- 移位运算：算数移位与逻辑移位
- 定点运算
 - 加减法运算
 - 一位乘法运算
 - booth算法
 - 除法运算
- 浮点运算

乘法运算

- 分析笔算乘法

$$A = -0.1101 \quad B = 0.1011$$

$$A \times B = -0.10001111 \quad \text{乘积的符号心算求得}$$

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

✓ 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

笔算乘法的改进

$$A \cdot B = A \cdot 0.1011$$

$$= 0.1A + 0.00A + \underline{0.001A} + \underline{0.0001A}$$

$$= 0.1A + 0.00A + \underline{0.001}(1 \cdot A + 0.1A)$$

$$= 0.1A + \underline{0.01}[0 \cdot A + 0.1(1 \cdot A + 0.1A)]$$

$$= \underline{0.1}\{1 \cdot A + 0.1[0 \cdot A + 0.1(1 \cdot A + \underline{0.1A})]\}$$

右移一位

$$= \underline{2^{-1}}\{1 \cdot A + \underline{2^{-1}}[0 \cdot A + \underline{2^{-1}}(1 \cdot A + \underline{2^{-1}}(1 \cdot A + 0))]\}$$

1 被乘数 $A + 0$

2 右移一位，得新的部分积

3 部分积 + 被乘数 * 系数 (0/1)

...

右移一位，得结果

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

①

②

③

⑧

• 改进后的笔算乘法过程（竖式）

$$A = -0.1101$$

$$B = 0.1011$$

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

数值部分

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

$$A \cdot 0.1011$$

$$= 2^{-1} \{ 1 \cdot A + 2^{-1} [0 \cdot A + 2^{-1} (1 \cdot A + 2^{-1} (1 \cdot A + 0))] \}$$

小结

- 乘法运算可用 **加和移位** 实现
 - $n = 4$, 加 4 次, 移 4 次

$$A \cdot 0.1011$$

$$= 2^{-1} \{ 1 \cdot A + 2^{-1} [0 \cdot A + 2^{-1} (1 \cdot A + 2^{-1} (1 \cdot A + 0))] \}$$

- 由乘数的末位决定被乘数是否与原部分积相加, 然后**->1位**形成新的部分积, 同时**乘数->1位** (末位移丢), 空出高位存放部分积的低位。

- 被乘数只与部分积的高位相加

硬件: 3个寄存器, 其中两个具有移位功能

1个全加器: 被乘数和部分积的高位相加操作 ($n+1$ 位)

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

原码乘法

- 原码一位乘运算规则

以小数为例 设 $[x]_{\text{原}} = x_0.x_1x_2 \cdots x_n$

$$[y]_{\text{原}} = y_0.y_1y_2 \cdots y_n$$

$$\begin{aligned} [x \cdot y]_{\text{原}} &= (x_0 \oplus y_0).(0.x_1x_2 \cdots x_n)(0.y_1y_2 \cdots y_n) \\ &= (x_0 \oplus y_0).x^*y^* \end{aligned}$$

式中 $x^* = 0.x_1x_2 \cdots x_n$ 为 x 的绝对值

$y^* = 0.y_1y_2 \cdots y_n$ 为 y 的绝对值

乘积的符号位单独处理 $x_0 \oplus y_0$

数值部分为绝对值相乘 $x^* \cdot y^*$

小数原码一位乘递推公式

$$x^* \cdot y^* = x^*(0.y_1y_2 \dots y_n)$$

$$= x^*(y_12^{-1} + y_22^{-2} + \dots + y_n2^{-n})$$

$$= 2^{-1}(y_1x^* + 2^{-1}y_2x^* + \dots + 2^{-n+1}y_nx^*)$$

$$= 2^{-1}(y_1x^* + 2^{-1}(y_2x^* + \dots + 2^{-1}(y_nx^* + 0) \dots))$$

The diagram illustrates the recursive structure of the formula. A large purple curly brace under the innermost part of the formula groups the terms $y_nx^* + 0$ and is labeled z_0 in blue. A smaller purple curly brace groups the terms $y_2x^* + \dots + 2^{-1}(y_nx^* + 0)$ and is labeled z_1 in blue. An ellipsis \dots is placed between the braces for z_1 and z_n . A large purple curly brace under the entire inner expression $y_1x^* + 2^{-1}(y_2x^* + \dots + 2^{-1}(y_nx^* + 0) \dots)$ is labeled z_n in blue.

$$z_0 = 0$$

$$z_1 = 2^{-1}(y_nx^* + z_0)$$

$$z_2 = 2^{-1}(y_{n-1}x^* + z_1)$$

\vdots

$$z_n = 2^{-1}(y_1x^* + z_{n-1})$$

整数原码一位乘递推公式

$$\begin{aligned}
 X^* \times Y^* &= X^*(Y_{n-1} \dots Y_1 Y_0) = 2^{n-1} Y_{n-1} X^* + 2^{n-2} Y_{n-2} X^* + \dots + 2^1 Y_1 X^* + 2^0 Y_0 X^* \\
 &= 2^n (2^{-1} Y_{n-1} X^* + 2^{-2} Y_{n-2} X^* + \dots + 2^{-n+1} Y_1 X^* + 2^{-n} Y_0 X^*) \\
 &= 2^n (2^{-1} (Y_{n-1} X^* + 2^{-1} Y_{n-2} X^* + \dots + 2^{-n+2} Y_1 X^* + 2^{-n+1} Y_0 X^*)) \\
 &= 2^n (2^{-1} (Y_{n-1} X^* + 2^{-1} (Y_{n-2} X^* + \dots + 2^{-n+3} Y_1 X^* + 2^{-n+2} Y_0 X^*))) \\
 &\quad \dots\dots\dots \dots\dots\dots \dots\dots\dots \dots\dots\dots \dots\dots\dots \dots\dots\dots \\
 &= 2^n (2^{-1} (Y_{n-1} X^* + 2^{-1} (Y_{n-2} X^* + 2^{-1} (\dots + 2^{-1} (Y_1 X^* + 2^{-1} (Y_0 X^* + \overset{Z_0}{0})))))) \dots)
 \end{aligned}$$

$$Z_0 = 0$$

$$Z_1 = 2^{-1} (Y_0 X^* + Z_0)$$

$$Z_2 = 2^{-1} (Y_1 X^* + Z_1)$$

...

$$Z_n = 2^{-1} (Y_{n-1} X^* + Z_{n-1})$$

Z_n

Z_1

• 例. 已知 $x = -0.1110$, $y = 0.1101$, 求 $[x \times y]_{\text{原}}$

解: 数值部分的运算

	部分积	乘数	说明
	$\begin{array}{r} 0.0000 \\ + 0.1110 \\ \hline 0.1110 \end{array}$	<u>1101</u>	部分积 初态 $z_0 = 0$ + x^*
逻辑右移	$\begin{array}{r} 0.1110 \\ 0.0111 \\ + 0.0000 \\ \hline 0.0111 \end{array}$	<u>0110</u>	$\rightarrow 1$, 得 z_1 + 0
逻辑右移	$\begin{array}{r} 0.0111 \\ 0.0011 \\ + 0.1110 \\ \hline 1.0001 \end{array}$	<u>1011</u>	$\rightarrow 1$, 得 z_2 + x^*
逻辑右移	$\begin{array}{r} 1.0001 \\ 0.1000 \\ + 0.1110 \\ \hline 1.0110 \end{array}$	<u>1101</u>	$\rightarrow 1$, 得 z_3 + x^*
逻辑右移	$\begin{array}{r} 1.0110 \\ 0.1011 \end{array}$	110 0110	$\rightarrow 1$, 得 z_4

$$\begin{aligned} z_0 &= 0 \\ z_1 &= 2^{-1}(y_n x^* + z_0) \\ z_2 &= 2^{-1}(y_{n-1} x^* + z_1) \\ &\vdots \\ z_n &= 2^{-1}(y_1 x^* + z_{n-1}) \end{aligned}$$

- 例结果

① 乘积的符号位 $x_0 \oplus y_0 = 1 \oplus 0 = 1$

② 数值部分按绝对值相乘

$$x^* \cdot y^* = 0.10110110$$

$$\text{则 } [x \cdot y]_{\text{原}} = 1.10110110$$

- 特点

- 绝对值运算
- 用移位的次数判断乘法是否结束
- 逻辑移位

补码一位乘法运算规则（小数）

设被乘数 $[x]_{\text{补}} = x_0 \cdot x_1 x_2 \dots x_n$ ，乘数 $[y]_{\text{补}} = y_0 \cdot y_1 y_2 \dots y_n$

●被乘数任意，乘数为正（类似原码乘） $y_0=0, [x \times y]_{\text{补}} = [x]_{\text{补}} \times y$

但加和移位按补码规则运算，积的符号自然形成

●被乘数任意，乘数为负 $[x \times y]_{\text{补}} = [x]_{\text{补}} \times 0.y_1 y_2 \dots y_n + [-x]_{\text{补}}$

乘数 $[y]_{\text{补}}$ ，去掉符号位，操作同上，最后加 $[-x]_{\text{补}}$ ，校正

$$y = [y]_{\text{补}} - 2 = 1.y_1 y_2 \dots y_n - 2 = 0.y_1 y_2 \dots y_n - 1$$

$$x \times y = x \times 0.y_1 y_2 \dots y_n - x$$

$$[x \times y]_{\text{补}} = [x \times 0.y_1 y_2 \dots y_n - x]_{\text{补}}$$

$$= [x]_{\text{补}} \times 0.y_1 y_2 \dots y_n + [-x]_{\text{补}}$$

$$[x \times y]_{\text{补}} = [x]_{\text{补}} (0.y_1 \dots y_n) + [-x]_{\text{补}} y_0$$

补码一位乘法运算规则（整数）

设 被乘数 $[X]_{\text{补}} = X_n \dots X_1 X_0$, 乘数 $[Y]_{\text{补}} = Y_n \dots Y_1 Y_0$

- 被乘数任意, 乘数为正 ($[X \times Y]_{\text{补}} = [X]_{\text{补}} \times [Y]_{\text{补}}$, $Y_n = 0$)

- 类似原码乘, 加和移位按补码规则, 积的符号自然形成

- 被乘数任意, 乘数为负

- 乘数 $[Y]_{\text{补}}$, 去掉符号位, 其他操作同上, 最后加 $2^n[-X]_{\text{补}}$ (校正)

$$[X \times Y]_{\text{补}} = [X]_{\text{补}} \times [Y_{n-1} \dots Y_1 Y_0]_{\text{补}} + 2^n [-X]_{\text{补}}$$

定点运算与浮点运算

- 移位运算： 算数移位与逻辑移位
- 定点运算
 - 加减法运算
 - 一位乘法运算
 - booth算法
 - 除法运算
- 浮点运算

Booth 算法（被乘数、乘数符号任意）

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

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

$$-[x]_{\text{补}} = +[-x]_{\text{补}}$$

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

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

$$2^{-1} = 2^0 - 2^{-1}$$

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

$$2^{-2} = 2^{-1} - 2^{-2}$$

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

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

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

$$y'_1 2^{-1} + \cdots + y'_n 2^{-n}$$

附加位 y_{n+1}

Booth 算法递推公式

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

$$[z_0]_{\text{补}} = 0$$

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

$$[z_2]_{\text{补}} = 2^{-1}\{(y_n - y_{n-1})[x]_{\text{补}} + [z_1]_{\text{补}}\}$$

⋮

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

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

最后一步不移位

y_i	y_{i+1}	$y_{i+1} - y_i$	操作
0	0	0	$\rightarrow 1$
0	1	1	$+ [x]_{\text{补}} \rightarrow 1$
1	0	-1	$+ [-x]_{\text{补}} \rightarrow 1$
1	1	0	$\rightarrow 1$

• 例. 已知 $x = +0.0011$, $y = -0.1011$, 求 $[x \times y]_{\text{补}}$

解:	0 0 . 0 0 0 0	1 . 0 1 0 1	0	
	+ 1 1 . 1 1 0 1			$+[-x]_{\text{补}}$
	1 1 . 1 1 0 1			
补码右移	1 1 . 1 1 1 0	1 1 0 1 0	1	$\rightarrow 1$
	+ 0 0 . 0 0 1 1			$+ [x]_{\text{补}}$
	0 0 . 0 0 0 1	1		
补码右移	0 0 . 0 0 0 0	1 1 1 0 1	0	$\rightarrow 1$
	+ 1 1 . 1 1 0 1			$+ [-x]_{\text{补}}$
	1 1 . 1 1 0 1	1 1		
补码右移	1 1 . 1 1 1 0	1 1 1 1 0	1	$\rightarrow 1$
	+ 0 0 . 0 0 1 1			$+ [x]_{\text{补}}$
	0 0 . 0 0 0 1	1 1 1		
补码右移	0 0 . 0 0 0 0	1 1 1 1 1	0	$\rightarrow 1$
	+ 1 1 . 1 1 0 1			$+ [-x]_{\text{补}}$
	1 1 . 1 1 0 1	1 1 1 1		最后一步不移位

$$[x]_{\text{补}} = 0.0011$$

$$[y]_{\text{补}} = 1.0101$$

$$[-x]_{\text{补}} = 1.1101$$

y_i	y_{i+1}	$y_{i+1} - y_i$	操作
0	0	0	$\rightarrow 1$
0	1	1	$+ [x]_{\text{补}} \rightarrow 1$
1	0	-1	$+ [-x]_{\text{补}} \rightarrow 1$
1	1	0	$\rightarrow 1$

$$\therefore [x \times y]_{\text{补}} = 1.11011111$$

• 例. 已知 $[x]_{\text{补}} = 1.0101$, $[y]_{\text{补}} = 1.0011$, 求 $[x \times y]_{\text{补}}$

解:	0 0 . 0 0 0 0	1 . 0 0 1 1	0	
	+ 0 0 . 1 0 1 1			$+[-x]_{\text{补}}$
	0 0 . 1 0 1 1			
补码右移	0 0 . 0 1 0 1	1 1 0 0 1	1	$\rightarrow 1$
补码右移	0 0 . 0 0 1 0	1 1 1 0 0	1	$\rightarrow 1$
	+ 1 1 . 0 1 0 1			$+ [x]_{\text{补}}$
	1 1 . 0 1 1 1	1 1		
补码右移	1 1 . 1 0 1 1	1 1 1 1 0	0	$\rightarrow 1$
补码右移	1 1 . 1 1 0 1	1 1 1 1 1	0	$\rightarrow 1$
	+ 0 0 . 1 0 1 1			$+ [-x]_{\text{补}}$
	0 0 . 1 0 0 0	1 1 1 1		最后一步不移位

$$[-x]_{\text{补}} = 0.1011$$

y_i	y_{i+1}	$y_{i+1} - y_i$	操作
0	0	0	$\rightarrow 1$
0	1	1	$+ [x]_{\text{补}} \rightarrow 1$
1	0	-1	$+ [-x]_{\text{补}} \rightarrow 1$
1	1	0	$\rightarrow 1$

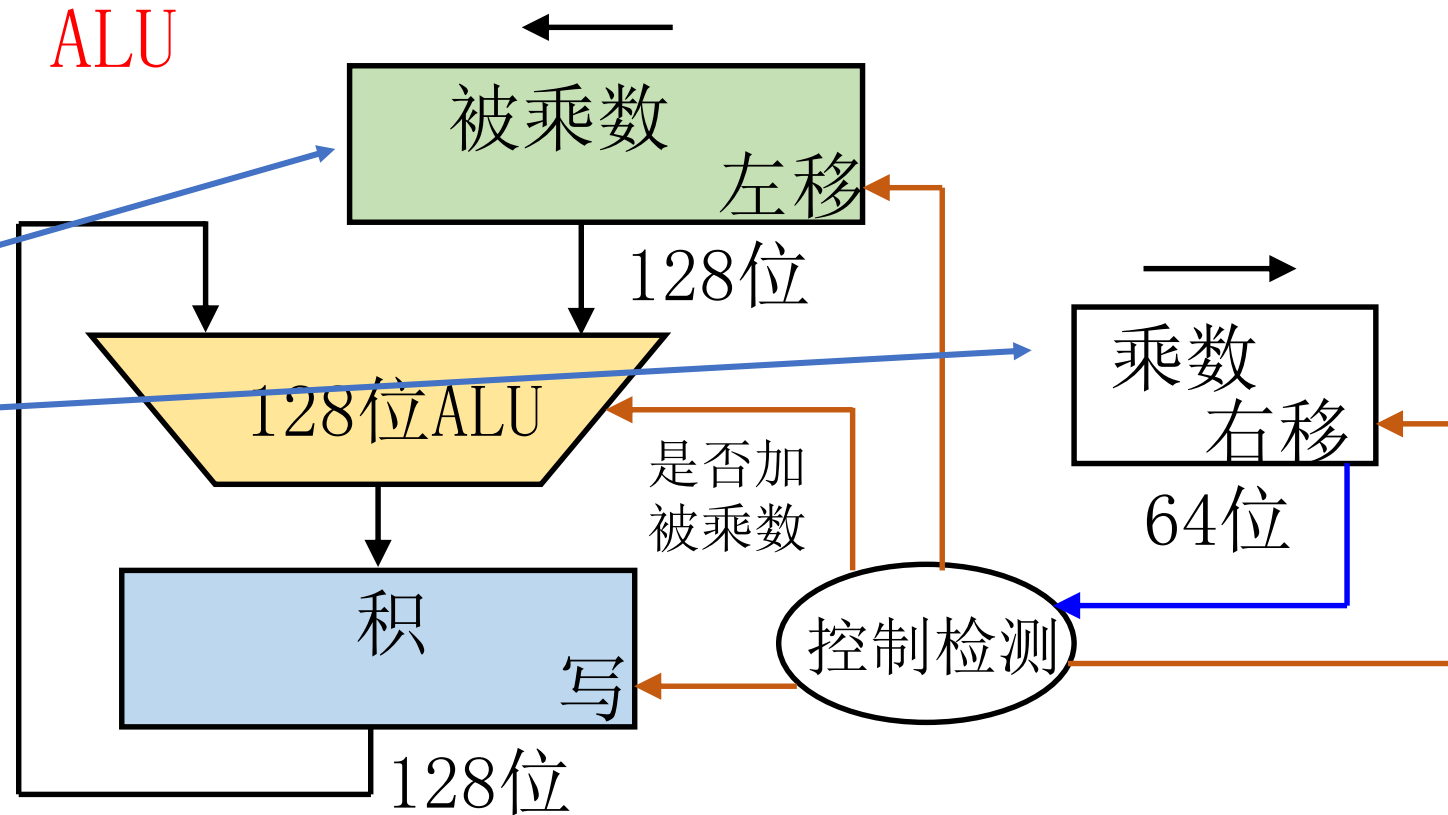
$$\therefore [x \times y]_{\text{补}} = 0.10001111$$

乘法小结

- 整数乘法与小数乘法基本相同
 - 可用 逗号 代替小数点
- 原码乘：符号位 单独处理
补码乘：符号位 自然形成
- 原码乘去掉符号位运算, 即为无符号数乘法
- 不同的乘法运算需有不同的硬件支持

乘法器硬件示意图

- 被乘数寄存器128位
 - 被乘数64位，要左移一位64次。
- 浪费：被乘数寄存器、ALU
 - 多数时间只用64位

$$\begin{array}{r} 1101 \\ \times 1011 \\ \hline 1101 \\ 1101 \\ 0000 \\ 1101 \\ \hline 10001111 \end{array}$$


黑书图3-3

改良版乘法器硬件

A. 积寄存器129位(初值为: 65个0 和 64位的乘数)

- 最高位用于保存加法器的进位

B. 若乘数最右端为1

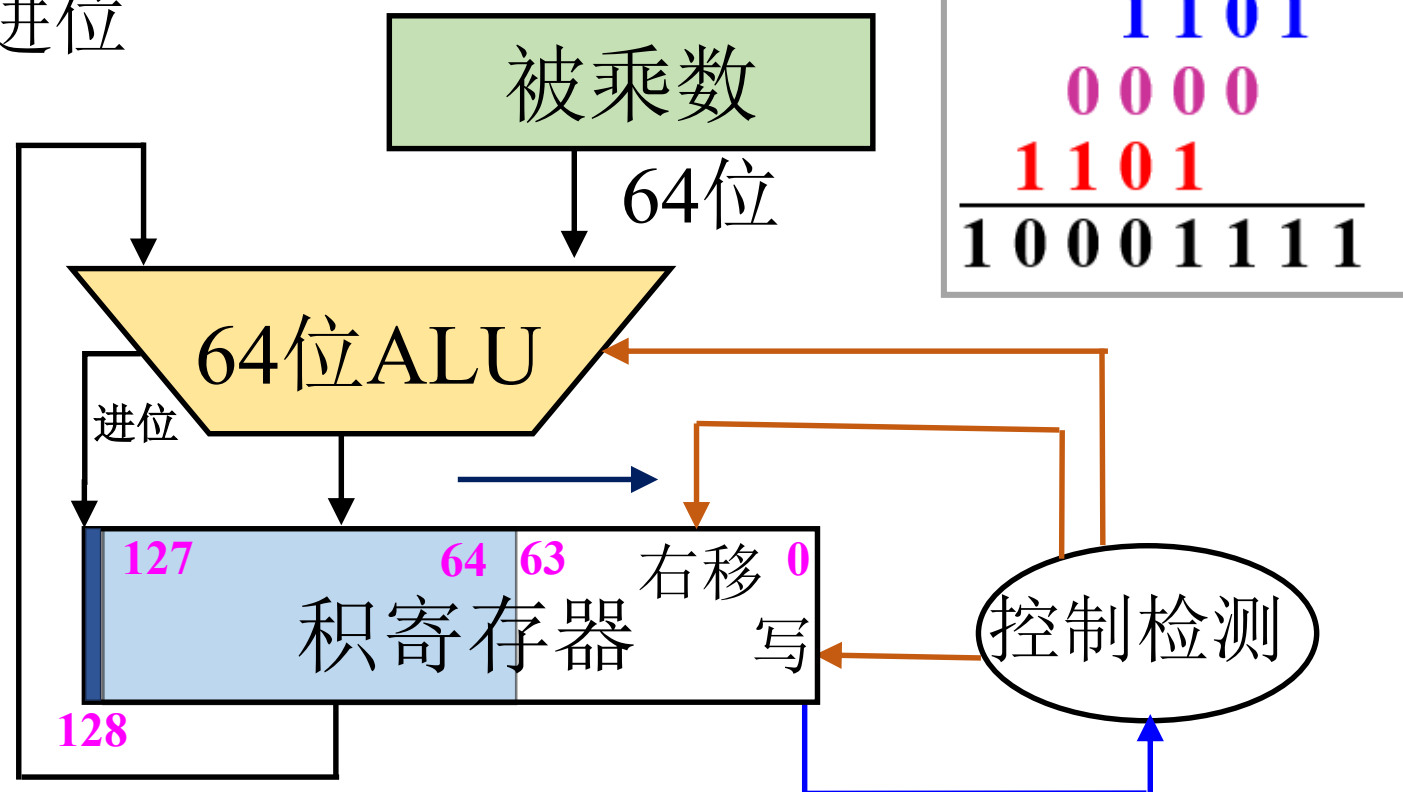
- 取积寄存器[127:64]
- 取出的值加上被乘数;
- **和**写入积寄存器[**128:64**]

C. 积寄存器整体右移一位

*B和C循环64次

*结果为积寄存器[127:0]

*快速乘法: 黑书3.3.3



黑书图3-5

定点运算与浮点运算

- 移位运算：算数移位与逻辑移位
- 定点运算
 - 加减法运算
 - 一位乘法运算
 - booth算法
 - 除法运算
- 浮点运算

除法运算

- 分析笔算除法

$$x = -0.1011 \quad y = 0.1101 \quad \text{求 } x \div y$$

$$\begin{array}{r} 0.1101 \overline{) 0.10110} \\ \underline{0.01101} \\ 0.010010 \\ \underline{0.001101} \\ 0.00010100 \\ \underline{0.00001101} \\ 0.000000111 \end{array}$$

✓ 商符单独处理

? 心算上商

? 余数不动低位补“0”
减右移一位的除数

? 上商位置不固定

$$x \div y = -0.1101 \quad \text{商符心算求得}$$

$$\text{余数 } 0.000000111$$

笔算除法和机器除法的比较

笔算除法

商符单独处理

心算上商

余数 **不动** 低位补“0”
减右移一位 的除数

2 倍字长加法器

上商位置 **不固定**

机器除法

符号位异或形成

$|x| - |y| \geq 0$ 上商 1

$|x| - |y| < 0$ 上商 0

余数 **左移一位** 低位补“0”
减 除数

1 倍字长加法器

在寄存器 **最末位**上商

$$\begin{array}{r} x = -0.1011 \quad y = 0.1101 \\ 0.1101 \overline{) 0.10110} \\ \underline{0.01101} \\ 0.010010 \\ \underline{0.001101} \\ 0.00010100 \\ \underline{0.00001101} \\ 0.00000111 \end{array}$$

原码除法

- 以小数为例

$$[x]_{\text{原}} = x_0 \cdot x_1 x_2 \dots x_n$$

$$[y]_{\text{原}} = y_0 \cdot y_1 y_2 \dots y_n$$

$$\left[\frac{x}{y}\right]_{\text{原}} = (x_0 \oplus y_0) \cdot \frac{x^*}{y^*}$$

式中 $x^* = 0.x_1 x_2 \dots x_n$ 为 x 的绝对值
 $y^* = 0.y_1 y_2 \dots y_n$ 为 y 的绝对值

商的符号位单独处理 $x_0 \oplus y_0$

数值部分为绝对值相除 $\frac{x^*}{y^*}$

约定 小数定点除法 $x^* < y^*$ 整数定点除法 $x^* > y^*$

被除数不等于 0

除数不能为 0

恢复余数法

• 例. $x = -0.1011$, $y = -0.1101$, 求 $\left[\frac{x}{y}\right]_{\text{原}}$

假设机器数是5位，符号位1位

$$[x]_{\text{原}} = 1.1011 \quad [y]_{\text{原}} = 1.1101 \quad [y^*]_{\text{补}} = 0.1101 \quad [-y^*]_{\text{补}} = 1.0011$$

① $x_0 \oplus y_0 = 1 \oplus 1 = 0$

② 被除数 (余数)	商	说 明
<u>0.1011</u>	0.0000	
+ 1.0011		$+[-y^*]_{\text{补}}$
1.1110	0	余数为负，上商 0
+ 0.1101		恢复余数 $+ [y^*]_{\text{补}}$
<u>0.1011</u>	0	恢复后的余数
1.0110	0	$\leftarrow 1$
+ 1.0011		$+ [-y^*]_{\text{补}}$
0.1001	01	余数为正，上商 1
1.0010	01	$\leftarrow 1$
+ 1.0011		$+ [-y^*]_{\text{补}}$

逻辑左移

逻辑左移

	0.1101
0.1101	0.10110
	0.01101
	0.010010
	0.001101
	0.00010100
	0.00001101
	0.00000111

上商前，比较被除数和除数：

$$[x^* - y^*]_{\text{补}} = [x^*]_{\text{补}} + [-y^*]_{\text{补}}$$

0.1101	0.1101
	0.10110
	0.01101
	0.010010
	0.001101
	0.00010100
	0.00001101
	0.00000111

上商前，比较被除数和除数：
 $[x^* - y^*]_{\text{补}} = [x^*]_{\text{补}} + [-y^*]_{\text{补}}$

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

$$\frac{x^*}{y^*} = 0.1101$$

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

余数为正 上商 1

余数为负 上商 0，恢复余数

上商 5 次

第一次上商判溢出

移 4 次

不恢复余数法（加减交替法）

• 恢复余数法运算规则

① 余数 $R_i > 0$ 上商 “1”， $2R_i - y^*$

② 余数 $R_i < 0$ 上商 “0”， $R_i + y^*$ 恢复余数

$$2(R_i + y^*) - y^* = 2R_i + y^*$$

被除数（余数）	商	说 明
0.1011	0.0000	
+ 1.0011		$+[-y^*]_{\text{补}}$
1.1110	0	余数为负，上商 0
+ 0.1101		恢复余数 $+ [y^*]_{\text{补}}$
0.1011	0	恢复后的余数
1.0110	0	$\leftarrow 1$
+ 1.0011		$+ [-y^*]_{\text{补}}$
0.1001	01	余数为正，上商 1
1.0010	01	$\leftarrow 1$
+ 1.0011		$+ [-y^*]_{\text{补}}$

• 不恢复余数法运算规则

① 余数 $R_i > 0$ ： 上商 “1”， $2R_i - y^*$

② 余数 $R_i < 0$ ： 上商 “0”， $2R_i + y^*$

加减交替

• 例. $x = -0.1011$, $y = -0.1101$, 求 $\left[\frac{x}{y}\right]_{\text{原}}$

解:

	0.1011	0.0000	
	+1.0011		$+[-y^*]_{\text{补}}$
逻辑左移	1.1110	0	余数为负, 上商 0
	1.1100	0	$\leftarrow 1$
	+0.1101		$+ [y^*]_{\text{补}}$
逻辑左移	0.1001	01	余数为正, 上商 1
	1.0010	01	$\leftarrow 1$
	+1.0011		$+ [-y^*]_{\text{补}}$
逻辑左移	0.0101	011	余数为正, 上商 1
	0.1010	011	$\leftarrow 1$
	+1.0011		$+ [-y^*]_{\text{补}}$
逻辑左移	1.1101	0110	余数为负, 上商 0
	1.1010	0110	$\leftarrow 1$
	+0.1101		$+ [y^*]_{\text{补}}$
	0.0111	01101	余数为正, 上商 1

$$[x]_{\text{原}} = 1.1011$$

$$[y]_{\text{原}} = 1.1101$$

$$[x^*]_{\text{补}} = 0.1011$$

$$[y^*]_{\text{补}} = 0.1101$$

$$[-y^*]_{\text{补}} = 1.0011$$

• 不恢复余数法运算规则

① 余数 $R_i > 0$ 上商 “1” $2R_i - y^*$

② 余数 $R_i < 0$ 上商 “0” $2R_i + y^*$

• 例. 结果

$$\textcircled{1} x_0 \oplus y_0 = 1 \oplus 1 = 0$$

$$\textcircled{2} \frac{x^*}{y^*} = 0.1101$$

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

特点 上商 $n+1$ 次

第一次上商判溢出

移 n 次，加 $n+1$ 次

用移位的次数判断除法是否结束

• 例. $x = -0.1011$, $y = -0.1101$, 求 $\left[\frac{x}{y}\right]_{\text{原}}$

解:

	0.1011	0.0000		
	+1.0011			+ $[-y^*]_{\text{补}}$
逻辑左移	1.1110	0		余数为负, 上商 0
	1.1100	0		← 1
	+0.1101			+ $[y^*]_{\text{补}}$
逻辑左移	0.1001	0 1		余数为正, 上商 1
	1.0010	0 1		← 1
	+1.0011			+ $[-y^*]_{\text{补}}$
逻辑左移	0.0101	0 1 1		余数为正, 上商 1
	0.1010	0 1 1		← 1
	+1.0011			+ $[-y^*]_{\text{补}}$
逻辑左移	1.1101	0 1 1 0		余数为负, 上商 0
	1.1010	0 1 1 0		← 1
	+0.1101			+ $[y^*]_{\text{补}}$
	0.0111	0 1 1 0 1		余数为正, 上商 1

$[x]_{\text{原}} = 1.1011$

$[y]_{\text{原}} = 1.1101$

$[x^*]_{\text{补}} = 0.1011$

$[y^*]_{\text{补}} = 0.1101$

$[-y^*]_{\text{补}} = 1.0011$

• 不恢复余数法运算规则

① 余数 $R_i > 0$ 上商 “1” $2R_i - y^*$

② 余数 $R_i < 0$ 上商 “0” $2R_i + y^*$

47

定点运算与浮点运算

- 移位运算： 算数移位与逻辑移位
- 定点运算
 - 加减法运算
 - 一位乘法运算
 - booth算法
 - 除法运算
- 浮点运算

浮点四则运算

- 一、浮点加减运算

$$x = S_x \cdot 2^{j_x} \quad y = S_y \cdot 2^{j_y}$$

1. 对阶

(1) 求阶差

$$\Delta j = j_x - j_y = \begin{cases} = 0 & j_x = j_y & \text{已对齐} \\ > 0 & j_x > j_y & \begin{cases} x \text{ 向 } y \text{ 看齐} & S_x \leftarrow 1, j_x - 1 \\ y \text{ 向 } x \text{ 看齐} & \checkmark S_y \rightarrow 1, j_y + 1 \end{cases} \\ < 0 & j_x < j_y & \begin{cases} x \text{ 向 } y \text{ 看齐} & \checkmark S_x \rightarrow 1, j_x + 1 \\ y \text{ 向 } x \text{ 看齐} & S_y \leftarrow 1, j_y - 1 \end{cases} \end{cases}$$

(2) 对阶原则

小阶向大阶看齐

3. 规格化

- (1) 规格化数的定义

$$r = 2 \quad \frac{1}{2} \leq |S| < 1$$

- (2) 规格化数的判断

$S > 0$	规格化形式	$S < 0$	规格化形式
真值	$0.1 \times \times \dots \times$	真值	$-0.1 \times \times \dots \times$
原码	$0.\boxed{1} \times \times \dots \times$	原码	$1.\boxed{1} \times \times \dots \times$
补码	$\boxed{0.1} \times \times \dots \times$	补码	$\boxed{1.0} \times \times \dots \times$
反码	$0.1 \times \times \dots \times$	反码	$1.0 \times \times \dots \times$

原码 不论正数、负数，第一数位为1

补码 符号位和第一数位不同

特例

$$S = -\frac{1}{2} = -0.100 \dots 0$$

$$[S]_{\text{原}} = 1.100 \dots 0$$

$$[S]_{\text{补}} = \boxed{1.1}00 \dots 0$$

$\therefore [-\frac{1}{2}]_{\text{补}}$ 不是规格化的数

$$S = -1$$

$$[S]_{\text{补}} = \boxed{1.0}00 \dots 0$$

$\therefore [-1]_{\text{补}}$ 是规格化的数

- (3)左规

尾数左移一位，阶码减 1，直到数符和第一数位不同为止

$$\text{上例 } [x+y]_{\text{补}} = 00, 11; 11. 1001$$

$$\text{左规后 } [x+y]_{\text{补}} = 00, 10; 11. 0010$$

$$\therefore x + y = (-0.1110) \times 2^{10}$$

- (4)右规

当 尾数溢出（>1）时，需 右规

即尾数出现 01. $\times \times \dots \times$ 或 10. $\times \times \dots \times$ 时

尾数右移一位，阶码加 1

- 例. $x = 0.1101 \times 2^{10}$, $y = 0.1011 \times 2^{01}$, 求 $x + y$
(除阶符、数符外, 阶码取 3 位, 尾数取 6 位)

解: $[x]_{\text{补}} = 00, 010; 00.110100$
 $[y]_{\text{补}} = 00, 001; 00.101100$

① 对阶

$$[\Delta j]_{\text{补}} = [j_x]_{\text{补}} - [j_y]_{\text{补}} = \begin{array}{r} 00, 010 \\ + 11, 111 \\ \hline 100, 001 \end{array}$$

阶差为 +1 $\therefore S_y \rightarrow 1, j_y + 1$

$\therefore [y]_{\text{补}}' = 00, 010; 00.010110$

② 尾数求和

$$\begin{array}{r} [S_x]_{\text{补}} = 00.110100 \\ + [S_y]_{\text{补}}' = 00.010110 \\ \hline 01.001010 \end{array} \quad \begin{array}{l} \text{对阶后的 } [S_y]_{\text{补}}' \\ \text{尾数溢出需右规} \end{array}$$

③ 右规

$$[x+y]_{\text{补}} = 00, 010; 01. 001010$$

右规后

$$[x+y]_{\text{补}} = 00, 011; 00. 100101$$

$$\therefore x+y = 0. 100101 \times 2^{11}$$

$$[x-y]_{\text{补}} = 11, 100; 10. 110001$$

$$\text{右规后得 } [x-y]_{\text{补}} = 11, 101; 11. 011000 \quad \boxed{1}$$

采用“0 舍 1 入”法,其尾数右规时末位丢 1,则有

$$\begin{array}{r} 11.011000 \\ + \quad \quad \quad 1 \\ \hline 11.011001 \end{array}$$

• 4. 舍入

- 在 对阶 和 右规 过程中,可能出现尾数末位丢失引起误差,需考虑舍入

(1) 0 舍 1 入法: 类似于四舍五入,尾数右移时,被移去的最高数值位为0,则舍去;被移去的最高数值位为1,则在尾数的末位加1。

(2) 恒置 “1” 法: 尾数右移时,不论丢掉的最高数值位是1或0,都使右移后的尾数末位横置1

- 例. $x = (-\frac{5}{8}) \times 2^{-5}$, $y = (\frac{7}{8}) \times 2^{-4}$, 求 $x-y$ (除阶符、数符外, 阶码取 3 位, 尾数取 6 位)

解: $x = (-0.101000) \times 2^{-101}$ $y = (0.111000) \times 2^{-100}$

$$[x]_{\text{补}} = 11, 011; 11. 011000 \quad [y]_{\text{补}} = 11, 100; 00. 111000$$

① 对阶

$$\begin{array}{r} [\Delta j]_{\text{补}} = [j_x]_{\text{补}} - [j_y]_{\text{补}} = 11, 011 \\ \quad \quad \quad + 00, 100 \\ \hline \quad \quad \quad 11, 111 \end{array}$$

$$\text{阶差为 } -1 \quad \therefore S_x \longrightarrow 1, j_x + 1$$

$$\therefore [x]_{\text{补}}' = 11, 100; 11. 101100$$

② 尾数求和

$$\begin{array}{r} [S_x]_{\text{补}'} = 11.101100 \\ + [-S_y]_{\text{补}} = 11.001000 \\ \hline 110.110100 \end{array}$$

③ 右规

$$[x-y]_{\text{补}} = 11, 100; 10.110100$$

右规后

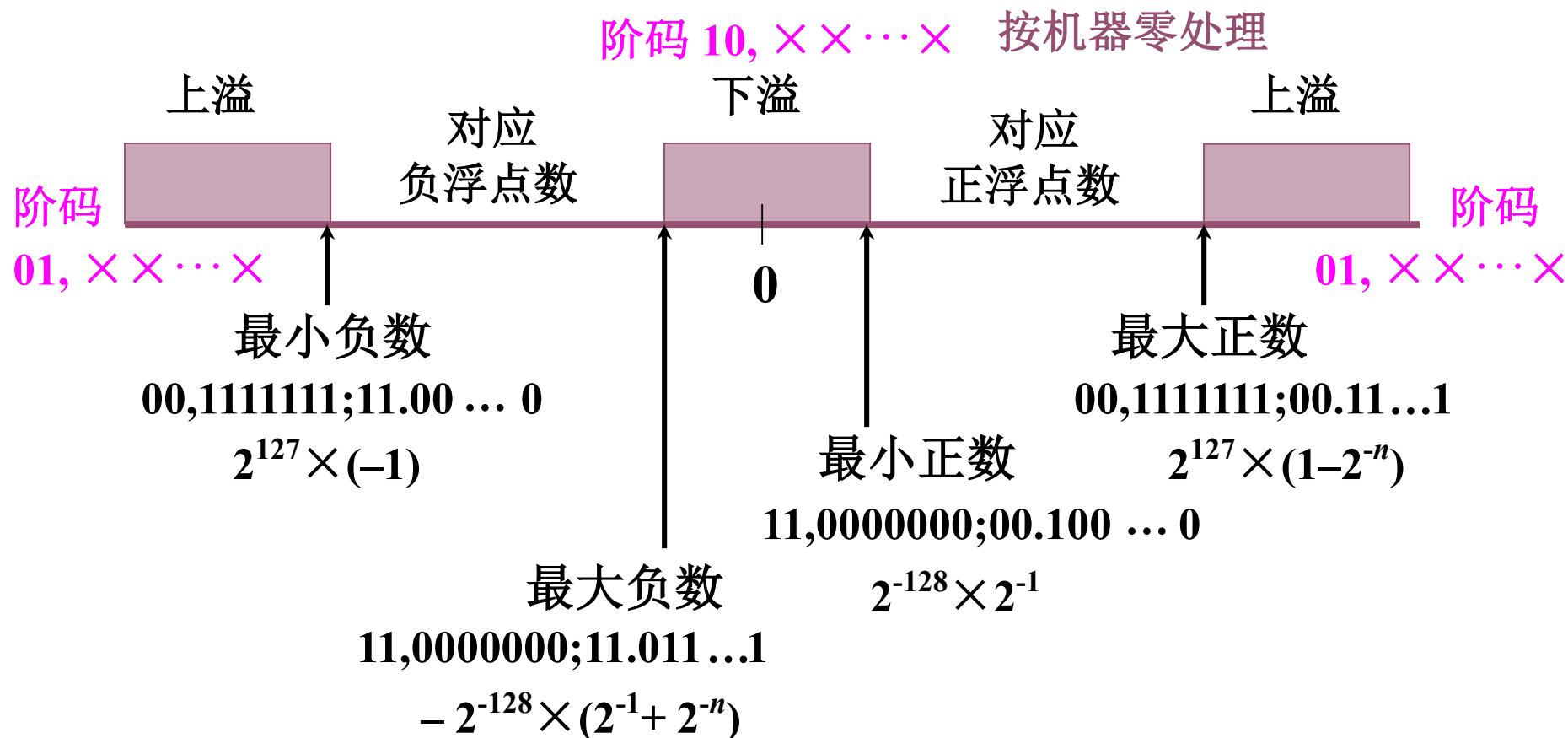
$$[x-y]_{\text{补}} = 11, 101; 11.011010$$

$$\begin{aligned} \therefore x-y &= (-0.100110) \times 2^{-11} \\ &= \left(-\frac{19}{32}\right) \times 2^{-3} \end{aligned}$$

$$x = S \cdot 2^j$$

溢出判断

- 设机器数为补码，尾数为规格化形式，并假设阶符取 2 位，阶码的数值部分取 7 位，数符取 2 位，尾数取 n 位，则该补码在数轴上的表示为



关于计算机的运算方法，哪部分有疑问？

- ☐ A 定点加减运算
- ☐ B 原码一位乘法
- ☐ C Booth乘法
- ☐ D 定点除法
- ☐ E 浮点运算
- ☐ F 其他可发弹幕

提交