计算机组成原理

第十七讲

刘松波

哈工大计算学部

模式识别与智能系统研究中心

第 6 章 计算机的运算方法

6.1 无符号数和有符号数

6.2 数的定点表示和浮点表示

6.3 定点运算

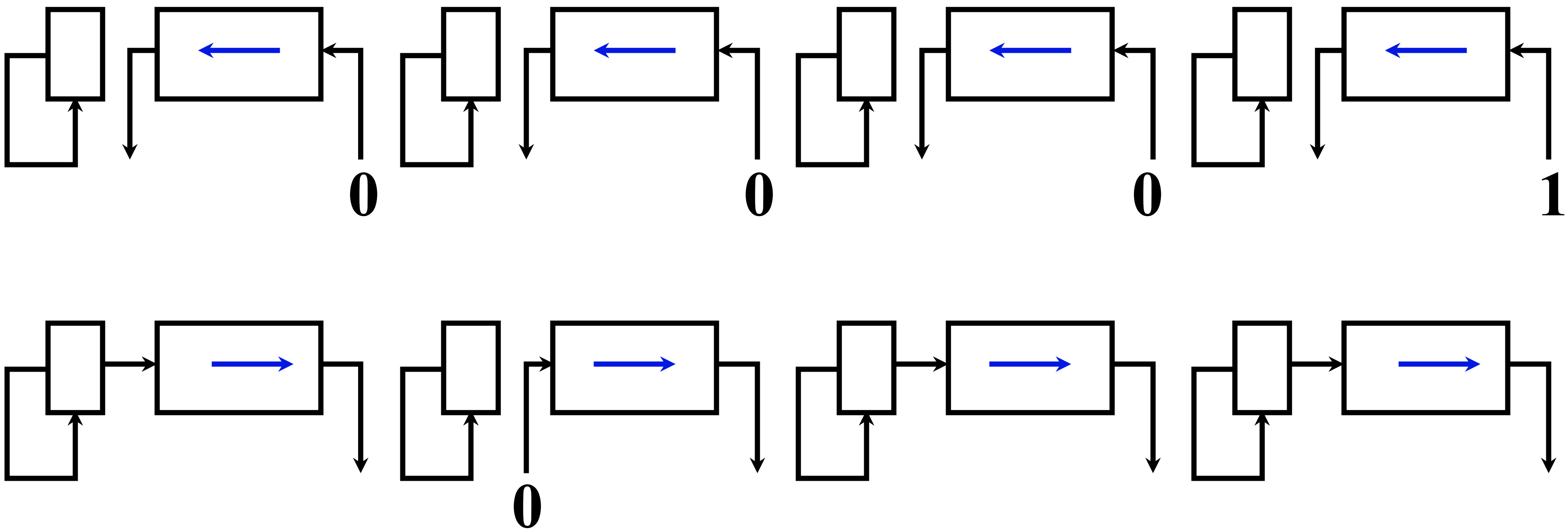
6.4 浮点四则运算

6.5 算术逻辑单元

上节课内容回顾

- 6.2 数的定点表示和浮点表示
  - 一、定点表示
  - 二、浮点表示
    - 浮点数的表示形式、范围、浮点数的规格化形式、浮点数的规格化
  - 三、IEEE 754标准
- 6.3 定点运算
  - 一、移位运算
    - 移位的意义
    - 移位的规则
    - 算术移位的硬件实现
    - 算术移位与逻辑移位的区别

3. 算术移位的硬件实现



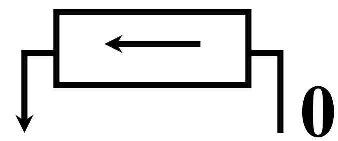
	（a）真值为正	（b）负数的原码	（c）负数的补码	（d）负数的反码
←丢 1	出错	出错	正确	正确
→丢 1	影响精度	影响精度	影响精度	正确

4. 算术移位和逻辑移位的区别

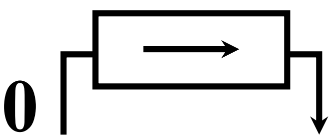
算术移位　　有符号数的移位

逻辑移位　　无符号数的移位

逻辑左移　　低位添 0，高位移丢

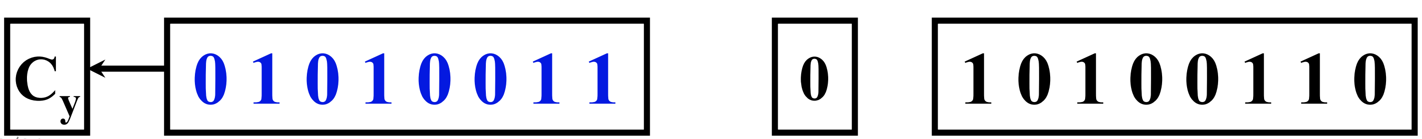


逻辑右移　　高位添 0，低位移丢



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



二、加减法运算

1. 补码加减运算公式

(1) 加法

整数 $[A]_{补} + [B]_{补} = [A+B]_{补}$ （mod 2^{n+1}）

小数 $[A]_{补} + [B]_{补} = [A+B]_{补}$ （mod 2）

(2) 减法

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

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

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

连同符号位一起相加，符号位产生的进位自然丢掉

2. 举例

例 6.18　设 $A = 0.1011$，$B = -0.0101$

求 $[A+B]_{补}$　　验证

解：

$$
\begin{array}{rcl}
[A]_{补} & = & 0.1011 \\
+[B]_{补} & = & 1.1011 \\
\hline
[A]_{补}+[B]_{补} & = & \boxed{1}\,0.0110 = [A+B]_{补}
\end{array}
$$

$\therefore A + B = 0.0110$

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

例 6.19　设 $A = -9$，$B = -5$

求 $[A+B]_{补}$　　验证

解：

$$
\begin{array}{rcl}
[A]_{补} & = & 1,0111 \\
+[B]_{补} & = & 1,1011 \\
\hline
[A]_{补}+[B]_{补} & = & \boxed{1}\,1,0010 = [A+B]_{补}
\end{array}
$$

$\therefore A + B = -1110$

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

例 6.20 设机器数字长为 8 位（含 1 位符号位）且 $A = 15$， $B = 24$，用补码求 $A - B$

解：$A = 15 = 0001111$

$B = 24 = 0011000$

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

$$+[-B]_{补} = 1,1101000$$

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

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

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

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

练习 2 设机器数字长为 8 位（含 1 位符号位）且 $A = -97$，$B = +41$，用补码求 $A - B$

$$A - B = +1110110 = +118$$ 错

3. 溢出判断

(1) 一位符号位判溢出

参加操作的 两个数（减法时即为被减数和“求补”以后的减数）符号相同，其结果的符号与原操作数的符号不同，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

$$\left.\begin{array}{l} 1 \oplus 0 = 1 \\ 0 \oplus 1 = 1 \end{array}\right\} \text{有 溢出}$$

$$\left.\begin{array}{l} 0 \oplus 0 = 0 \\ 1 \oplus 1 = 0 \end{array}\right\} \text{无 溢出}$$

(2) 两位符号位判溢出

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

$$[x]_{补'} + [y]_{补'} = [\,x + y\,]_{补'} \pmod{4}$$

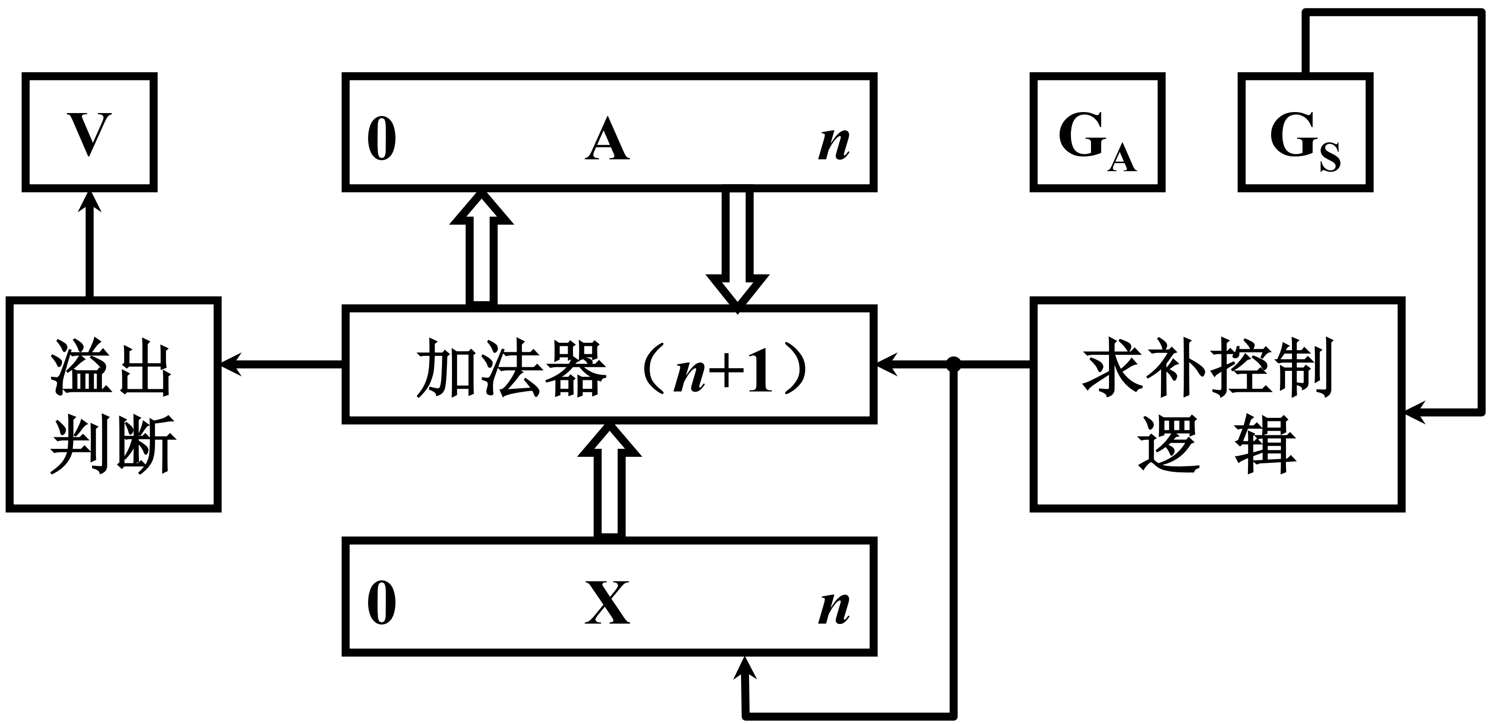
$$[x - y]_{补'} = [x]_{补'} + [-\,y]_{补'} \pmod{4}$$

结果的双符号位 相同　　未溢出　　00, ×××××
11, ×××××

结果的双符号位 不同　　溢出　　10, ×××××
01, ×××××

最高符号位 代表其 真正的符号

4. 补码加减法的硬件配置



A、X 均 n+1 位

用减法标记 G_S 控制求补逻辑

三、乘法运算

1. 分析笔算乘法

$A = -0.1101 \qquad B = 0.1011$

$A \times B = -0.10001111$ 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

2. 笔算乘法改进

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

$$= 0.1A + 0.00A + 0.001A + 0.0001A$$

$$= 0.1A + 0.00A + 0.001(A + 0.1A)$$

$$= 0.1A + 0.01[0 \cdot A + 0.1(A + 0.1A)]$$

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

右移一位

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

第一步　被乘数 $A + 0$ ①

第二步　右移 一 位，得新的部分积 ②

第三步　部分积 + 被乘数 ③

⋮

第八步　右移 一 位，得结果 ⑧

3. 改进后的笔算乘法过程（竖式） 6.3

部分积	乘数	说明
0 . 0 0 0 0 + 0 . 1 1 0 1	1 0 1 1	初态，部分积 = 0 乘数为 1，加被乘数
0 . 1 1 0 1 0 . 0 1 1 0 + 0 . 1 1 0 1	1 1 0 1	→1，形成新的部分积 乘数为 1，加被乘数
1 . 0 0 1 1 0 . 1 0 0 1 + 0 . 0 0 0 0	1 1 1 1 0	→ 1，形成新的部分积 乘数为 0，加 0
0 . 1 0 0 1 0 . 0 1 0 0 + 0 . 1 1 0 1	1 1 1 1 1 1	→ 1，形成新的部分积 乘数为 1，加 被乘数
1 . 0 0 0 1 0 . 1 0 0 0	1 1 1 1 1 1 1	→1，得结果

小结 6.3

- 乘法 运算可用 加和移位实现
  $n = 4$，加 4 次，移 4 次
- 由乘数的末位决定被乘数是否与原部分积相加，然后 $\longrightarrow$1 位形成新的部分积，同时 乘数 $\longrightarrow$ 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，具有移位功能

1 个全加器

4. 原码乘法

(1) 原码一位乘运算规则

以小数为例

设 $[x]_{原} = x_0 . x_1 x_2 \quad \cdots \quad x_n$

$[y]_{原} = y_0 . y_1 y_2 \quad \cdots \quad y_n$

$[x \cdot y]_{原} = (x_0 \oplus y_0).(0. x_1 x_2 \cdots x_n)(0. y_1 y_2 \cdots y_n)$

$= (x_0 \oplus y_0). x^* y^*$

式中 $x^* = 0. x_1 x_2 \cdots x_n$ 为 x 的绝对值

$y^* = 0. y_1 y_2 \cdots y_n$ 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 $x^* \cdot y^*$

(2) 原码一位乘递推公式

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

$$= x^*(y_1 2^{-1} + y_2 2^{-2} + \ \cdots \ + y_n 2^{-n})$$

$$= \underbrace{2^{-1}(y_1 x^* + 2^{-1}(y_2 x^* + \ \cdots \ \underbrace{2^{-1}(y_n x^* + \underbrace{0}_{z_0})}_{z_1} \cdots))}_{z_n}$$

$$z_0 = 0$$

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

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

$$\vdots$$

$$z_n = 2^{-1}(y_1 x^* + z_{n-1})$$

例6.21 已知 $x = -0.1110$ $y = 0.1101$ 求 $[x\cdot y]_{原}$

解： 数值部分的运算

	部分积	乘数	说明
	0.0000	1101	部分积 初态 $z_0 = 0$
	+ 0.1110		$+x^*$
	0.1110		
逻辑右移	0.0111	0110	$\rightarrow$1，得 z_1
	+ 0.0000		$+0$
	0.0111	0	
逻辑右移	0.0011	1011	$\rightarrow$1，得 z_2
	+ 0.1110		$+x^*$
	1.0001	10	
逻辑右移	0.1000	1101	$\rightarrow$1，得 z_3
	+ 0.1110		$+x^*$
	1.0110	110	
逻辑右移	0.1011	0110	$\rightarrow$1，得 z_4

例6.21 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

$$x^* \cdot y^* = 0.\,1\,0\,1\,1\,0\,1\,1\,0$$

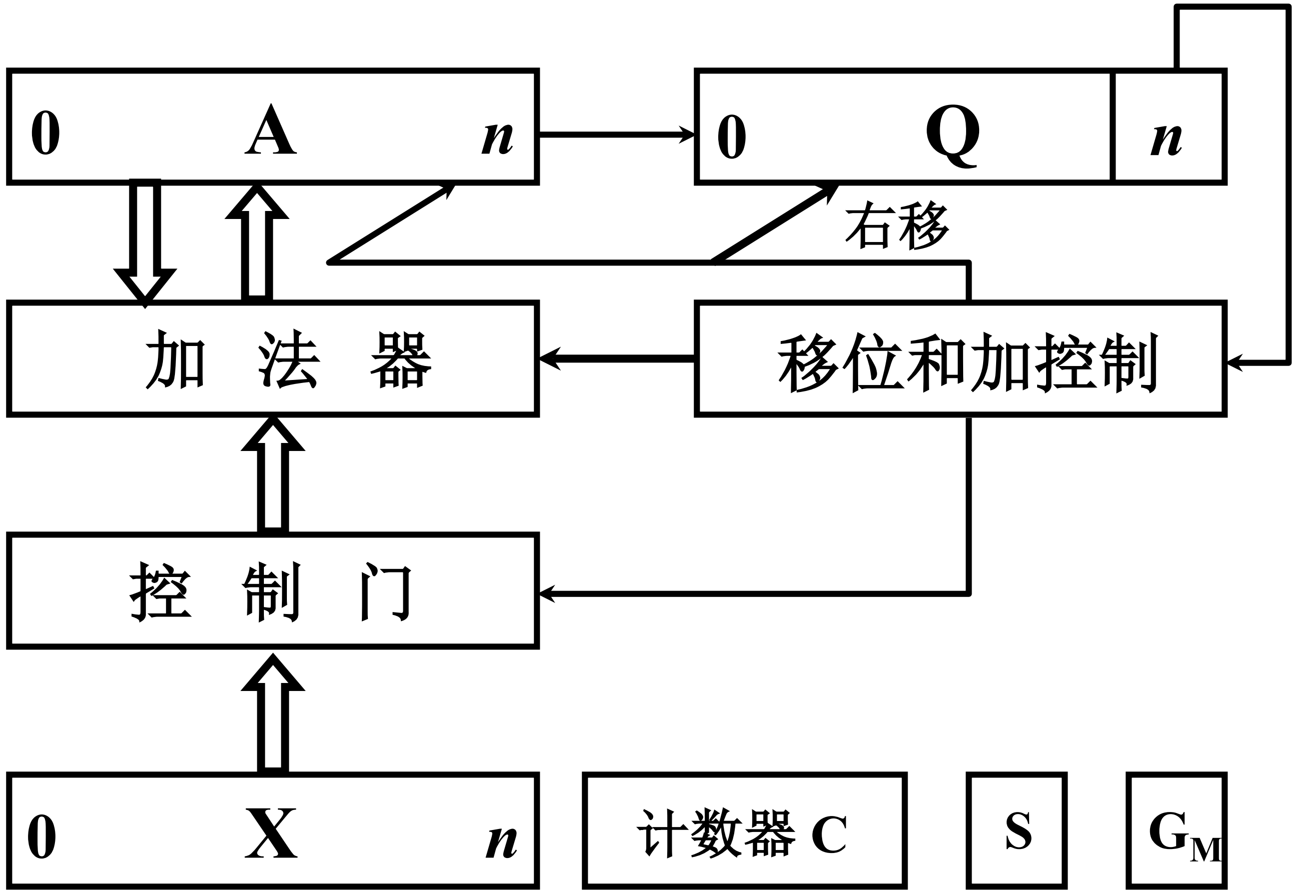
则 $[x \cdot y]_{原} = 1.\,1\,0\,1\,1\,0\,1\,1\,0$

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

(3) 原码一位乘的硬件配置



A、X、Q 均 n+1 位

移位和加受末位乘数控制