第 6 章　计算机的运算方法

6.3 定点运算

- 一、移位运算
- 二、加减法运算
- 三、乘法运算
- 四、除法运算

6.3 定点运算

- 一、移位运算
  - 1、移位运算的数学意义
  - 2、算术移位规则
  - 3、算术移位的硬件实现
  - 4、算术移位与逻辑移位的区别

6.3 定 点 运 算

一、移位运算

1. 移位的意义

15. m = 1500. 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

2. 算术移位规则

6.3

$x=-0.x_1x_2...x_k100...000$

$[x]_{补}=1.\overline{x}_1\overline{x}_2...\overline{x}_k100...000$

符号位不变

	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

表示真值的0

例6.16

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

移位操作	机器数 $[A]_{原}=[A]_{补}=[A]_{反}$	对应的真值
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

例6.17

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：　　$A = -26 = -11010$

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

6.3

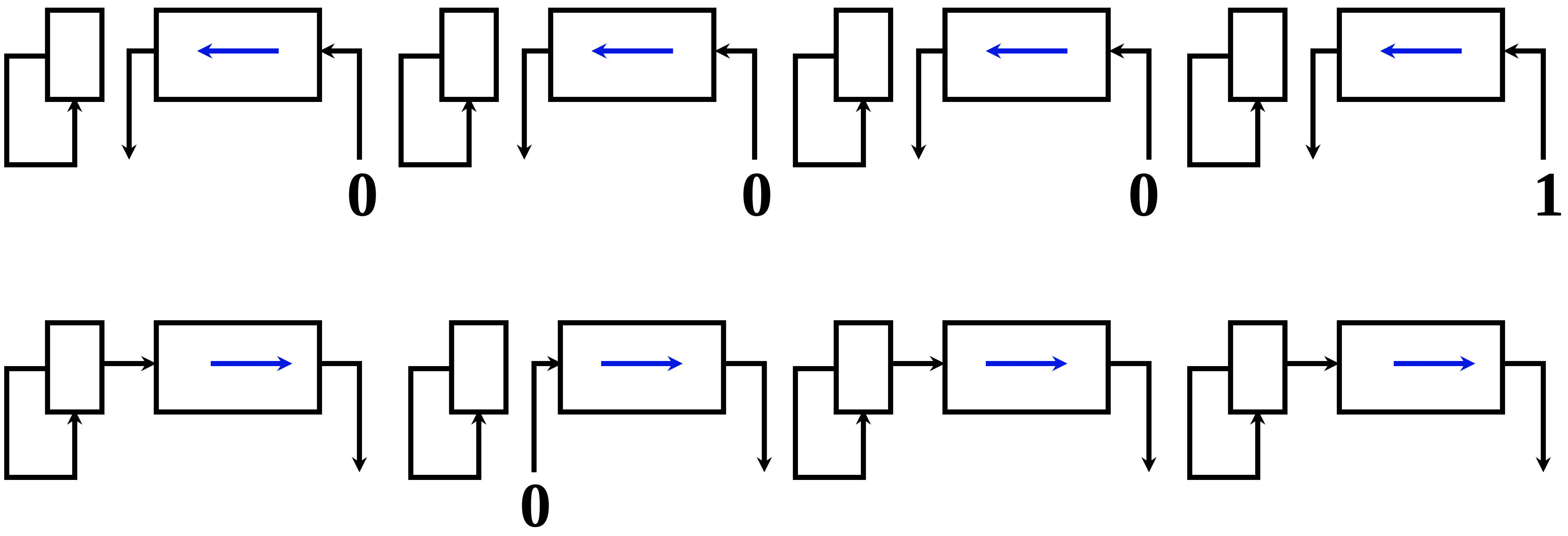
补码

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

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

3. 算术移位的硬件实现



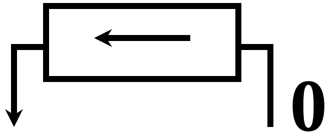
	（a）真值为正	（b）负数的原码	（c）负数的补码	（d）负数的反码
←丢 1	出错	出错	正确 丢0出错	正确
→丢 1	影响精度	影响精度	影响精度	正确

4. 算术移位和逻辑移位的区别

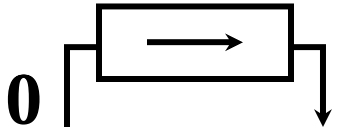
算术移位　　有符号数的移位　符号位保持不动

逻辑移位　　无符号数的移位　所有位参加移位运算

逻辑左移　　低位添 0，高位移丢



逻辑右移　　高位添 0，低位移丢

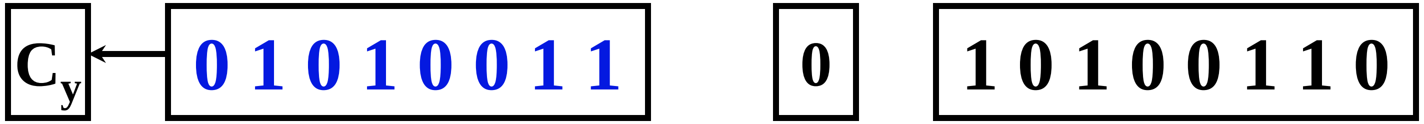


例如　　01010011　　　　10110010

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



进位位

6.3 定点运算

- 一、移位运算
  – 1、移位运算的数学意义
  – 2、算术移位规则
  – 3、算术移位的硬件实现
  – 4、算术移位与逻辑移位的区别

6.3 定点运算

- 一、移位运算
- 二、加减法运算
- 三、乘法运算
- 四、除法运算

但是用原码作加法时，会出现如下问题：

要求	数1	数2	实际操作	结果符号
加法	正	正	加	正
加法	正	负	减	可正可负
加法	负	正	减	可正可负
加法	负	负	加	负

能否 只作加法？

找到一个与负数等价的正数 来代替这个负数

就可使 减 $\longrightarrow$ 加

6.3 定点运算

- 二、加减法运算
  - 1、补码加减法运算的公式
  - 2、举例
  - 3、溢出的判断
  - 4、补码加减法的硬件配置

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

实际实现时，考虑机器字长！！！

6.3

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）6.3
且 $A = 15$，$B = 24$，用补码求 $A - B$

解： $A = 15 = 0001111$

$B = 24 = 0011000$

$[A]_补 = 0, 0001111$ $[B]_补 = 0, 0011000$

$+ [-B]_补 = 1, 1101000$

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

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

练习 1 设 $x = \frac{9}{16}$ $y = \frac{11}{16}$，用补码求 $x+y$ 假设机器字长5位，含1位符号位

$x + y = -0.1100 = -\frac{12}{16}$ 错 超过了小数定点机能表示的范围，溢出

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 $A - B$ 假设机器字长8位，含1位符号位

$A - B = +1110110 = +118$ 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

数值的进位　　　两个操作数都为负数时发生

最高有效位的进位 $\oplus$ 符号位的进位 = 1　　溢出

如

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

两个正数相加

两个负数相加

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

(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, ×××××

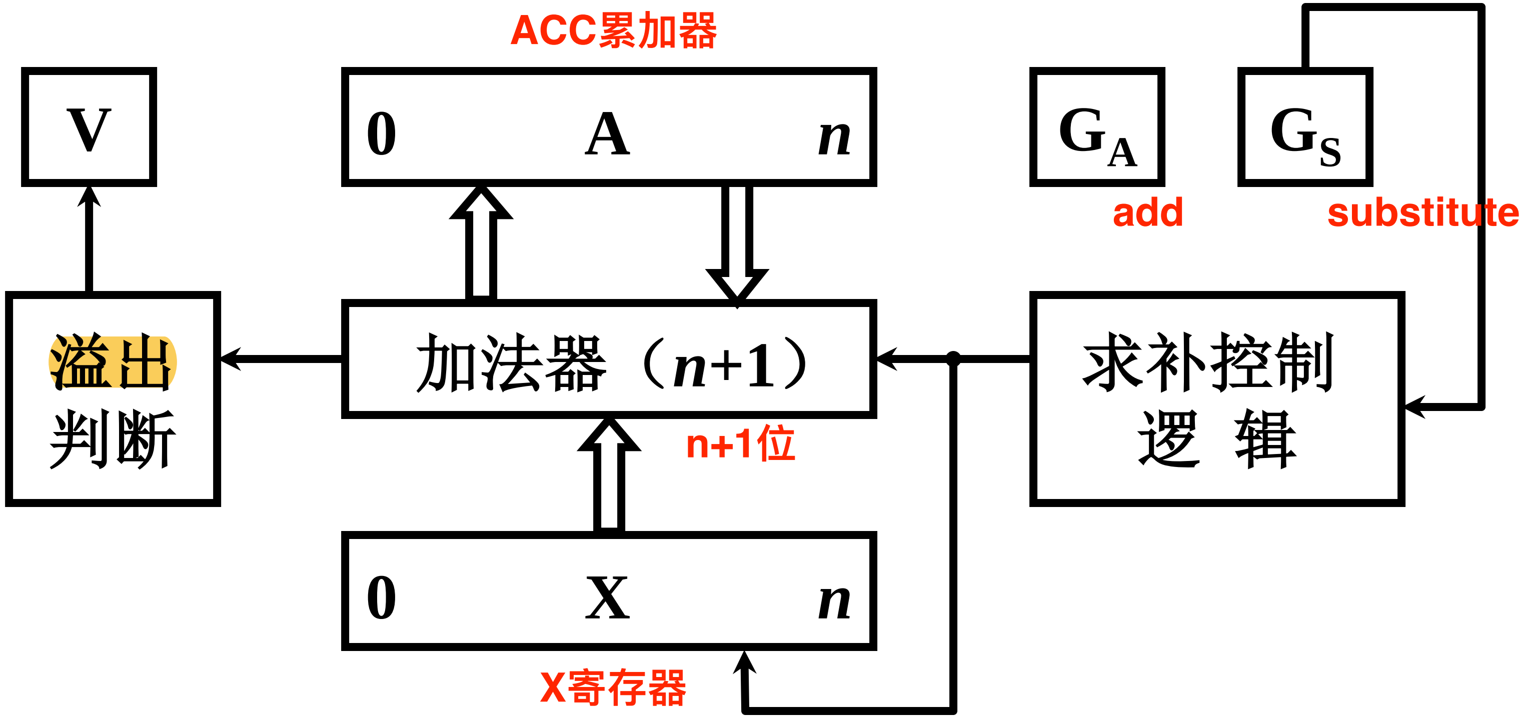
11, ×××××

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

6.3 定点运算

- 一、移位运算
- 二、加减法运算
- 三、乘法运算
- 四、除法运算

6.3 定点运算

- 三、乘法运算
  - 计算机中怎么做二进制的乘法运算呢
    - 可以分析一下笔算乘法是怎么做的
  - 笔算乘法的分析
  - 笔算乘法的改进
  - 原码的乘法运算
  - 补码的乘法运算

三、乘法运算

6.3

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 异或电路

✓ 乘数的某一位决定是否加被乘数

? 4个位积一起相加 累加器

✓ 乘积的位数扩大一倍

2. 笔算乘法改进

6.3

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

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

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

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

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

右移一位

$$= \boxed{2^{-1}}\{1 \bullet A + 2^{-1}[0 \cdot A + \underbrace{2^{-1}(1 \bullet A + \underbrace{2^{-1}(\underbrace{1 \bullet A + 0}_{①})}_{②})}_{③}]\}$$

第一步　被乘数A + 0

第二步　右移 一 位，得新的部分积

第三步　部分积 + 被乘数

⋮

第八步　右移 一 位，得结果　⑧

n次加法，n次移位

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 部分积的低位，移出来的部分 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 ⟶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
  只和ACC中的相加，不与MQ中的低位相加

硬件　3 个寄存器，其中2个具有移位功能
ACC，X，MQ
1 个全加器

6.3 定点运算

- 三、乘法运算
  - 计算机中怎么做二进制的乘法运算呢
    - 可以分析一下笔算乘法是怎么做的
  - 笔算乘法的分析
  - 笔算乘法的改进
  - 原码的乘法运算
  - 补码的乘法运算

运算规则
递推公式
举例
硬件配置

4. 原码乘法

(1) 原码一位乘运算规则

以小数为例

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

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

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

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

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

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

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

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

$$z_0 = 0$$

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

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

$$\vdots$$

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

例6.21 已知 $x = -0.1110$ $y = 0.1101$ 求$[x \cdot y]_原$ 6.3

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	→ 1，得 z_1
+ 0.0000		+ 0
0.0111	0	
逻辑右移 0.0011	1011	→ 1，得 z_2
+ 0.1110		$+ x^*$
1.0001	10	
逻辑右移 0.1000	1101	→ 1，得 z_3
+ 0.1110		$+ x^*$
1.0110	110	
逻辑右移 0.1011	0110	→ 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

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

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

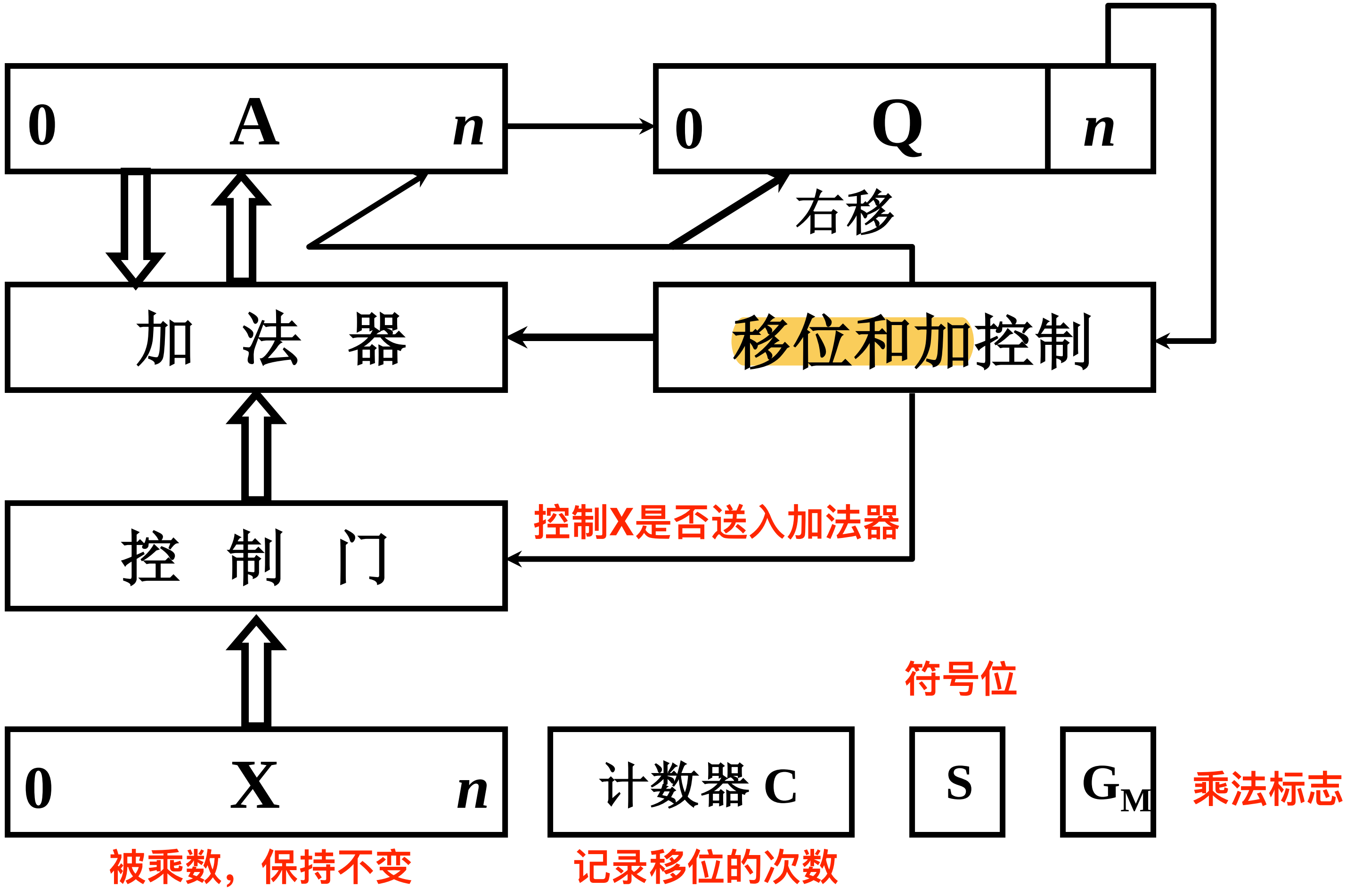
特点　　绝对值运算

用移位的次数判断乘法是否结束

而不是加法的次数，因为+0可以不做加法

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