

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

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

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

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

2. 算术移位规则

6.3

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

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

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

例6.16

6.3

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	对应的真值
	$[A]_{原}=[A]_{补}=[A]_{反}$	
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

例6.17

6.3

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

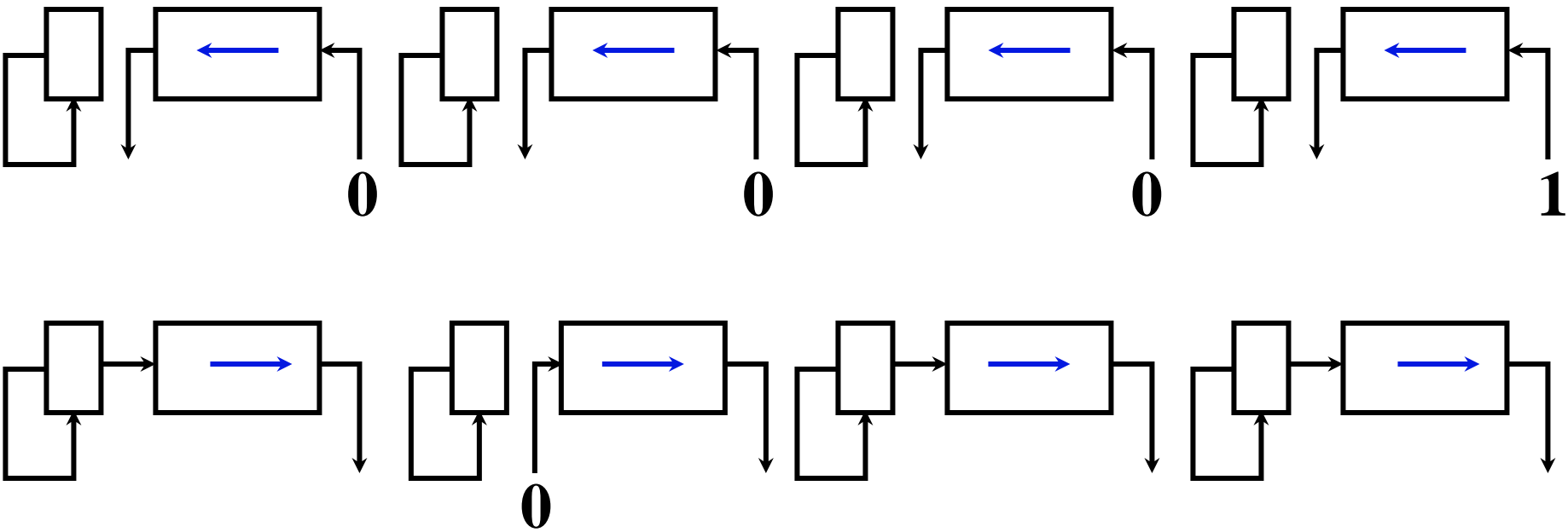
移位操作	机 器 数	对应的真值
移位前	1,1100110	– 26
左移一位	1,1001100	– 52
左移两位	1,0011000	– 104
右移一位	1,1110011	– 13
右移两位	1,1111001	– 7

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

6.3



(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

6.3

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

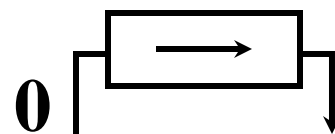
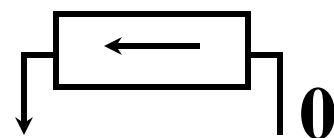
例如 **01010011**

逻辑左移 **10100110**

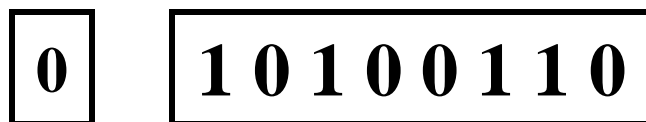
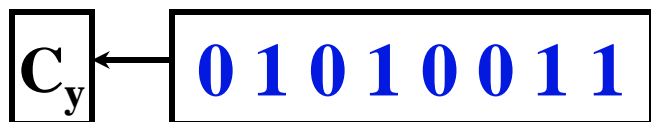
算术左移 **00100110**

逻辑右移 **01011001**

算术右移 **11011001** (补码)



高位 1 移丢



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 → 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

6.3

1. 补码加减运算公式

(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) 减法

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

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

2. 举例

6.3

例 6.18 设 $A = 0.1011$, $B = -0.0101$

求 $[A + B]_{\text{补}}$

验证

解: $[A]_{\text{补}} = 0.1011$

$+ [B]_{\text{补}} = 1.1011$

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

$\therefore A + B = 0.0110$

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

例 6.19 设 $A = -9$, $B = -5$

求 $[A + B]_{\text{补}}$

验证

解: $[A]_{\text{补}} = 1, 0111$

$+ [B]_{\text{补}} = 1, 1011$

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

$\therefore A + B = -1110$

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

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: $A = 15 = 0001111$

$B = 24 = 0011000$

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

$+ [-B]_{\text{补}} = 1, 1101000$

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

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

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

6.3

(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

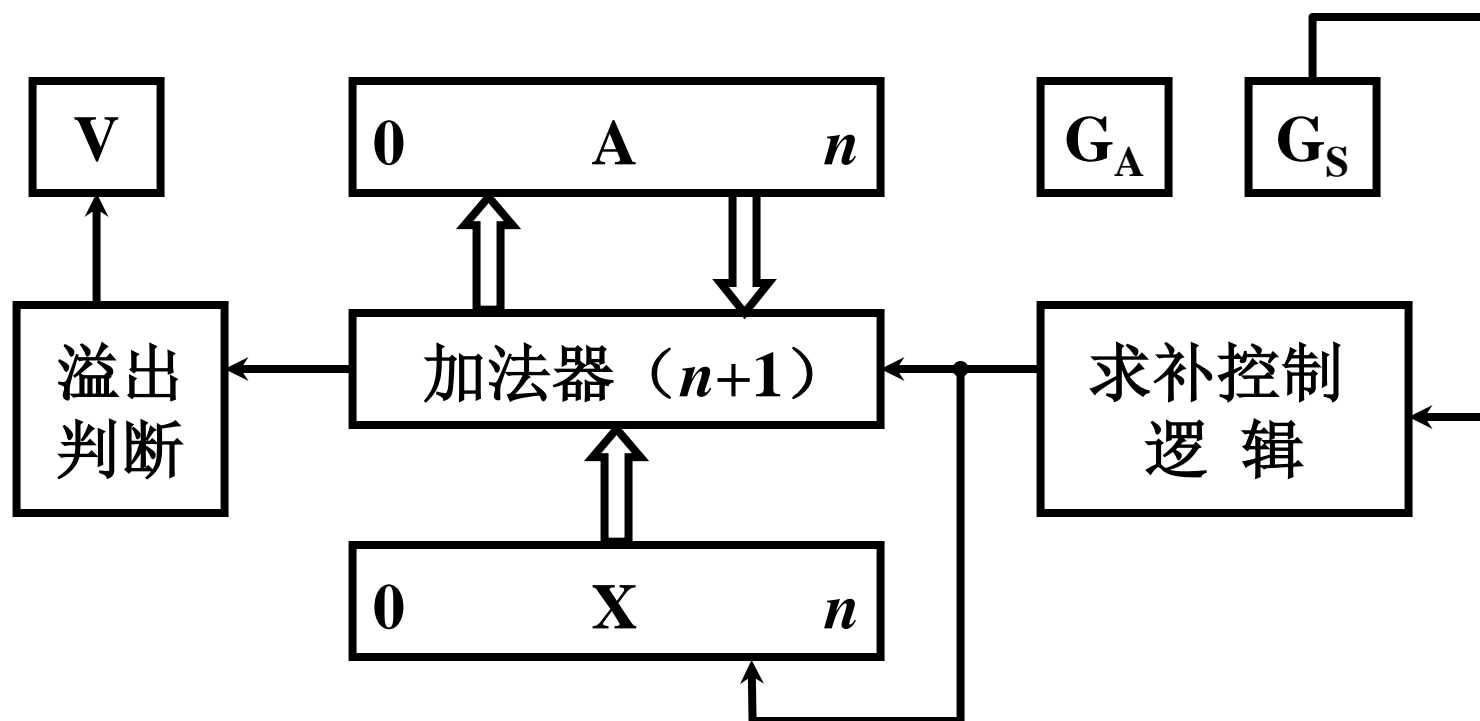
如

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

6.3

4. 补码加减法的硬件配置

6.3



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

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

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

0.1101	
× 0.1011	✓ 符号位单独处理
<hr/>	
1101	✓ 乘数的某一位决定是否加被乘数
1101	
0000	? 4个位积一起相加
1101	
<hr/>	
0.10001111	✓ 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

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

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

①

②

③

3. 改进后的笔算乘法过程（竖式） 6.3

部 分 积	乘 数	说 明
<div>0.0000</div> <div>+ 0.1101</div>	<div>1011</div> <div>=</div>	初态，部分积 = 0 乘数为 1，加被乘数
<div>0.1101</div> <div>0.0110</div> <div>+ 0.1101</div>	<div>1101</div> <div>=</div>	→1，形成新的部分积 乘数为 1，加被乘数
<div>1.0011</div> <div>0.1001</div> <div>+ 0.0000</div>	<div>1</div> <div>1110</div> <div>=</div>	→1，形成新的部分积 乘数为 0，加 0
<div>0.1001</div> <div>0.0100</div> <div>+ 0.1101</div>	<div>11</div> <div>1111</div> <div>=</div>	→1，形成新的部分积 乘数为 1，加 被乘数
<div>1.0001</div> <div>0.1000</div>	<div>111</div> <div>1111</div>	→1，得结果

- 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 \rightarrow 1 位形成新的部分积, 同时 乘数 \rightarrow 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2个具有移位功能

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

6.3

(1) 原码一位乘运算规则

以小数为例

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

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

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

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 $x^* \cdot y^*$

(2) 原码一位乘递推公式

6.3

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

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

$$= 2^{-1}(y_1x^* + \underbrace{2^{-1}(y_2x^* + \dots 2^{-1}(y_nx^* + \underbrace{0}_{z_0}) \dots)}_{z_1} \dots)_{z_n}$$

$$z_0 = 0$$

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

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

$$\vdots$$

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

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

解：数值部分的运算

	部分积	乘数	说明
	0.0000	110 <u>1</u>	部分积 初态 $z_0 = 0$
	+ 0.1110		+ x^*
逻辑右移	0.1110 0.0111	011 <u>0</u>	$\rightarrow 1$, 得 z_1
	+ 0.0000		+ 0
逻辑右移	0.0111 0.0011	0 101 <u>1</u>	$\rightarrow 1$, 得 z_2
	+ 0.1110		+ x^*
逻辑右移	1.0001 0.1000	10 110 <u>1</u>	$\rightarrow 1$, 得 z_3
	+ 0.1110		+ x^*
逻辑右移	1.0110 0.1011	110 0110	$\rightarrow 1$, 得 z_4

例6.21 结果

6.3

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

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

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

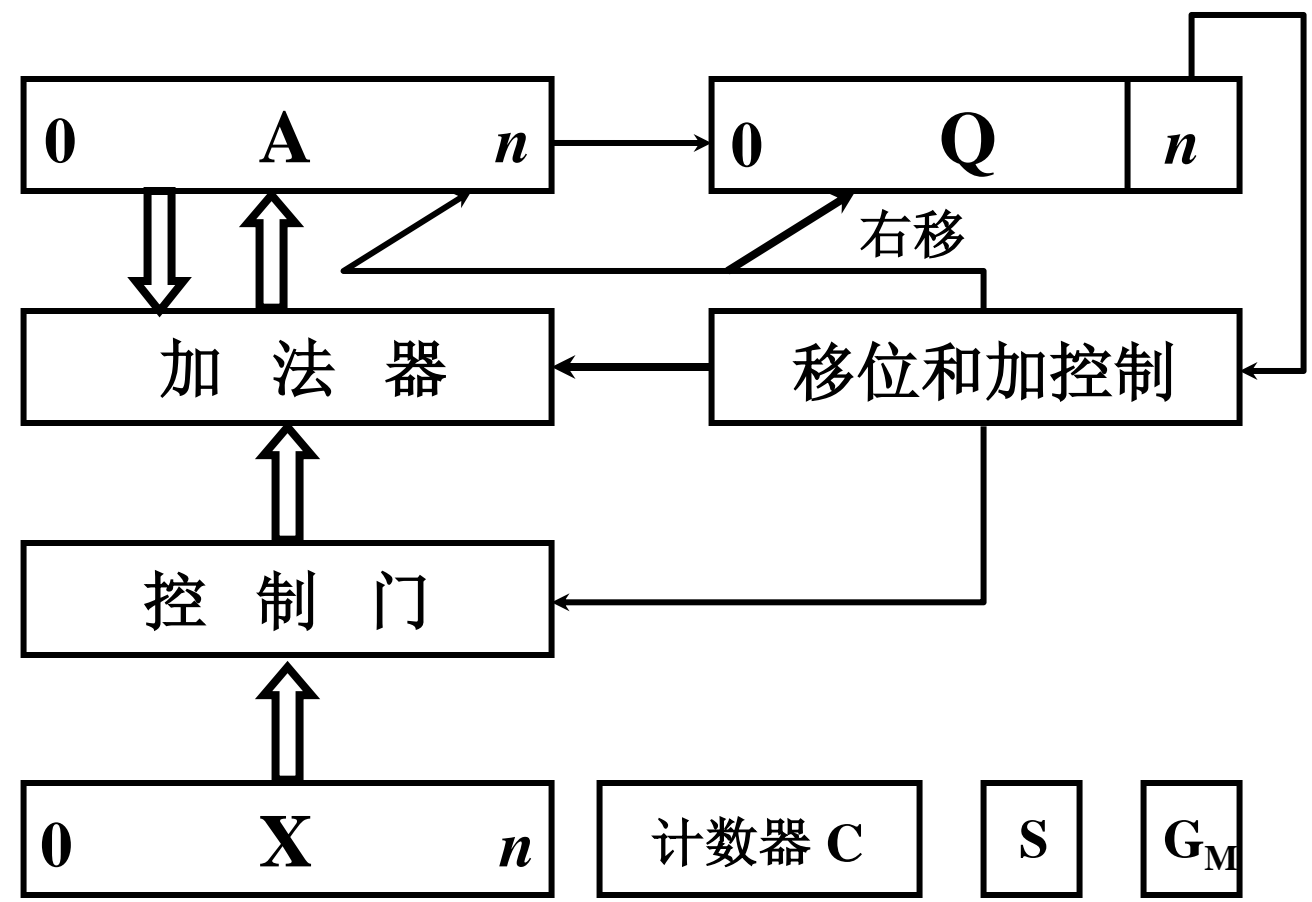
特点 绝对值运算

用移位的次数判断乘法是否结束

逻辑移位

(3) 原码一位乘的硬件配置

6.3



A、X、Q 均 $n+1$ 位

移位和加受末位乘数控制