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. 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

$x=-0.x_1x_2\ldots x_k100\ldots000$

$[x]_{补}=1.\overline{x}_1\overline{x}_2\ldots\overline{x}_k100\ldots000$

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

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

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

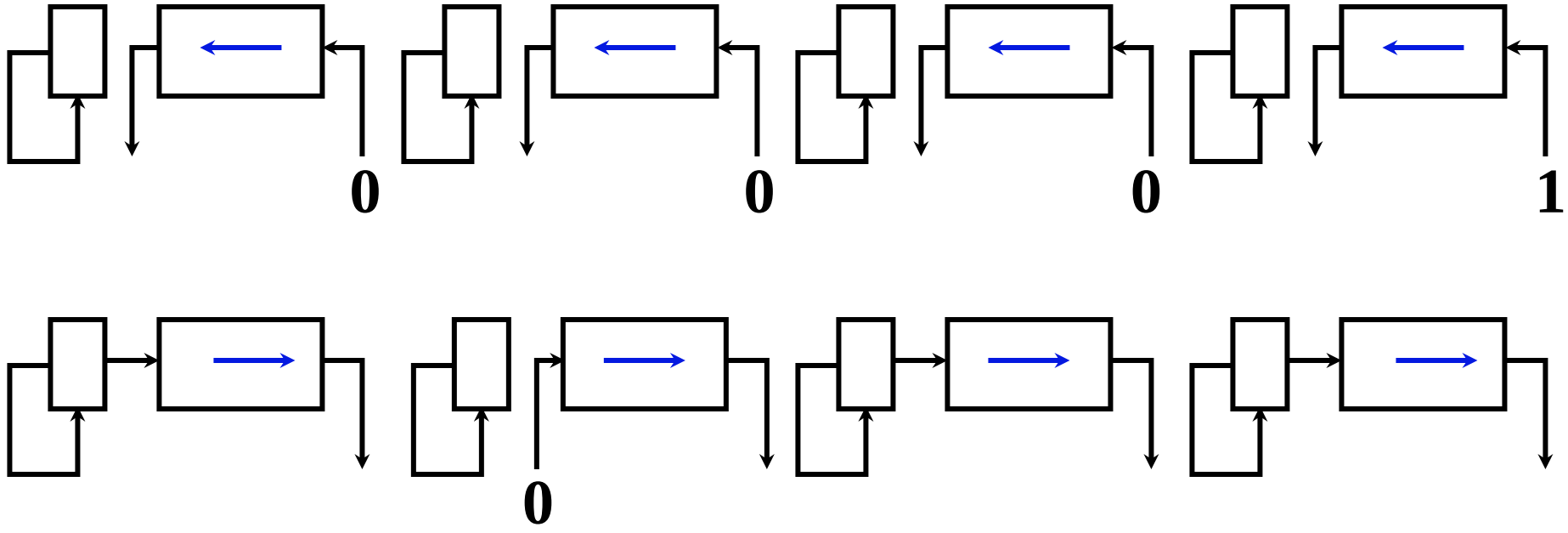
补码

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

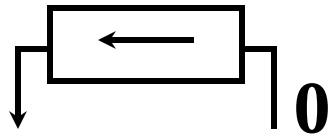
6.3

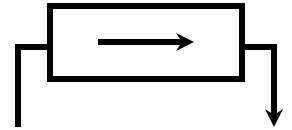
算术移位　　有符号数的移位

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　　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

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

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

(2) 减法

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

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

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

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：

$$
\begin{array}{rl}
 & [A]_{补} = 0.1011 \\
+ & [B]_{补} = 1.1011 \\
\hline
 & [A]_{补} + [B]_{补} = 10.0110 = [A + B]_{补}
\end{array}
$$

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

$\therefore A + B = 0.0110$

例 6.19　设 $A = -9$，$B = -5$

求 $[A+B]_{补}$　　验证

解：

$$
\begin{array}{rl}
 & [A]_{补} = 1,0111 \\
+ & [B]_{补} = 1,1011 \\
\hline
 & [A]_{补} + [B]_{补} = 11,0010 = [A + B]_{补}
\end{array}
$$

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

$\therefore A + B = -1110$

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

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 ⊕ 符号位的进位 = 1　　溢出

如

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

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

(2) 两位符号位判溢出

$$[x]_{补'} = \begin{cases} x & 1 > x \geq 0 \\ 4 + x & 0 > x \geq -1 \end{cases} \quad (\text{mod } 4)$$

$$[x]_{补'} + [y]_{补'} = [\, x + y \,]_{补'} \quad (\text{mod } 4)$$

$$[x - y]_{补'} = [x]_{补'} + [-y]_{补'} \quad (\text{mod } 4)$$

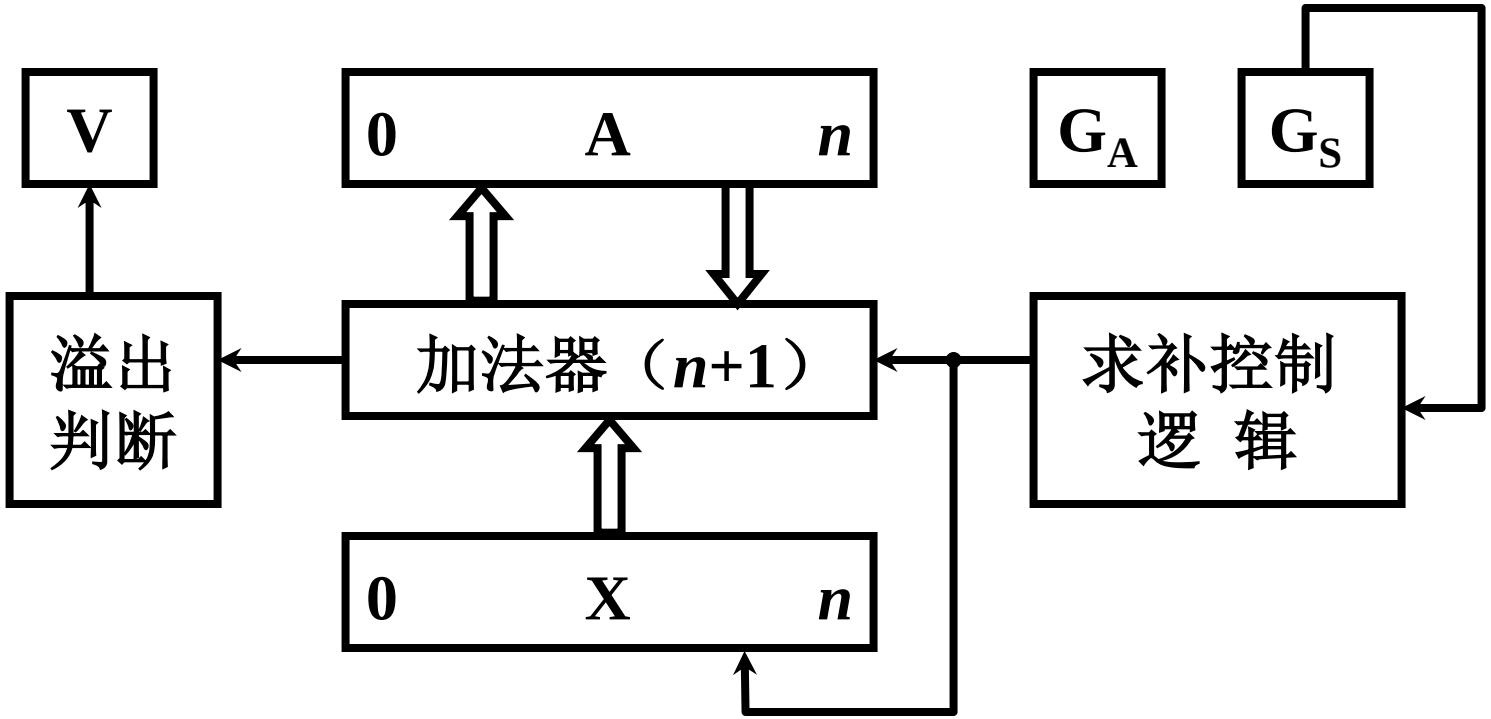
结果的双符号位 相同　　未溢出　　**00,** ×××××
11, ×××××

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

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

$A = -0.1101$　　$B = 0.1011$

$A \times B = -0.10001111$　乘积的符号心算求得

```
  0.1101
 ×0.1011
 ------
    1101
   1101
  0000
 1101
 ------------
0.10001111
```

✓ 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

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　　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
0.0000 +0.1101	1011	初态，部分积 = 0 乘数为 1，加被乘数
0.1101 0.0110 +0.1101	1101	→1，形成新的部分积 乘数为 1，加被乘数
1.0011 0.1001 +0.0000	1 1110	→ 1，形成新的部分积 乘数为 0，加 0
0.1001 0.0100 +0.1101	11 1111	→ 1，形成新的部分积 乘数为 1，加 被乘数
1.0001 0.1000	111 1111	→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

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

(1) 原码一位乘运算规则

以小数为例

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

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

$[x \cdot y]_{原} = (x_0 \oplus y_0).(0.\ x_1x_2 \cdots\ x_n)(0.y_1y_2\ \cdots y_n)$

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

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

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

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

$$= 2^{-1}(y_1 x^* + 2^{-1}(y_2 x^* + \cdots 2^{-1}(y_n x^* + 0) \cdots))$$

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		
0.0111	0110	→ 1，得 z_1
+ 0.0000		+ 0
0.0111	0	
0.0011	1011	→ 1，得 z_2
+ 0.1110		+ x^*
1.0001	10	
0.1000	1101	→ 1，得 z_3
+ 0.1110		+ x^*
1.0110	110	
0.1011	0110	→ 1，得 z_4

逻辑右移

逻辑右移

逻辑右移

逻辑右移

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 $[x \cdot y]_{原} = 1.1\,0\,1\,1\,0\,1\,1\,0$

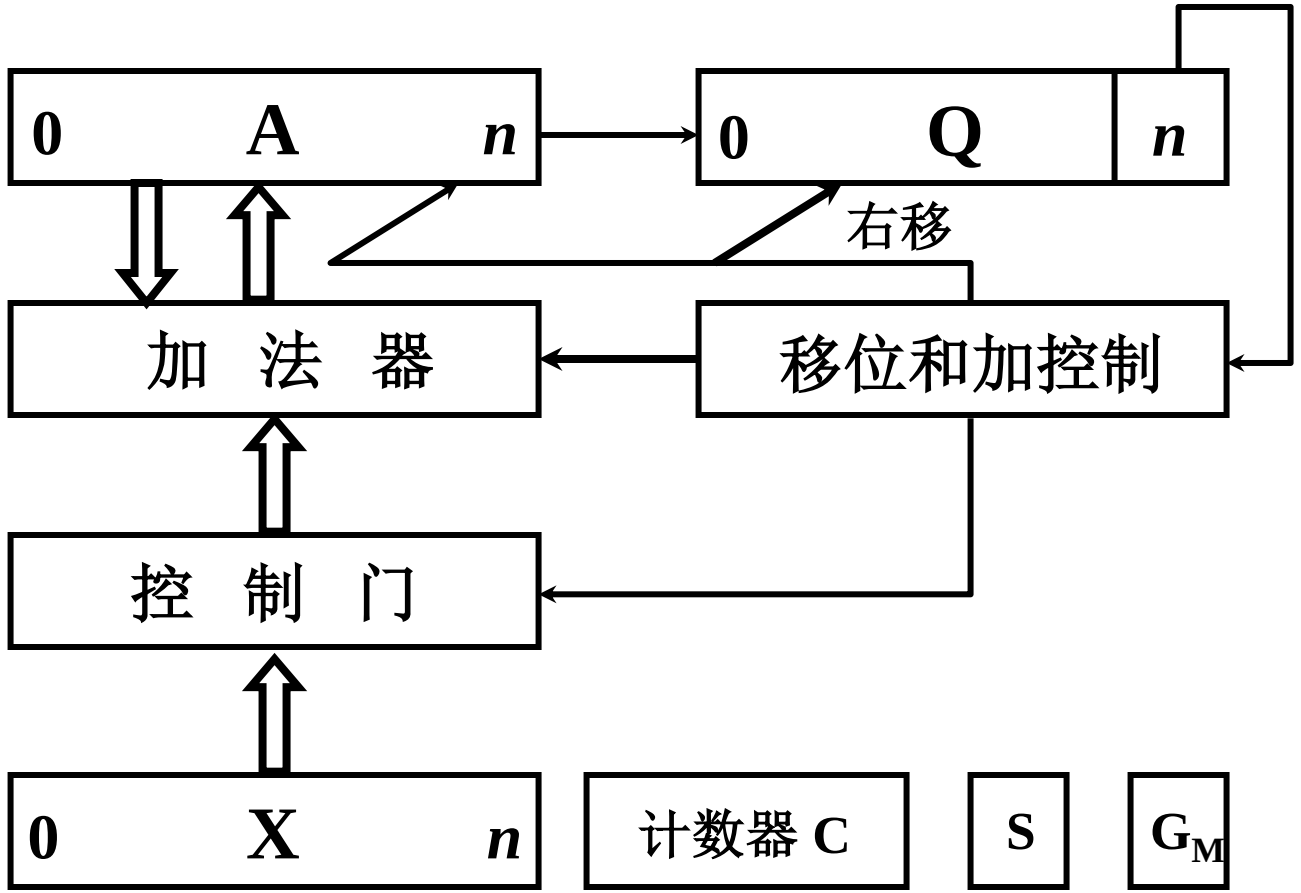
特点　绝对值运算

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

6.3



A、X、Q 均 *n*+1 位

移位和加受末位乘数控制

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
  - 1. 分析笔算乘法
  - 2. 笔算乘法的改进
  - 3. 改进后的乘法笔算过程
  - 4. 原码一位乘
  - 5. 补码一位乘

5. 补码乘法

6.3

(1) 补码一位乘运算规则

以小数为例　设 被乘数　$[x]_{补} = x_0.\ x_1x_2 \cdots\ \ x_n$

乘数　　$[y]_{补} = y_0.\ y_1y_2 \cdots\ \ y_n$

① 被乘数任意，乘数为正

与原码乘相似 但 加 和 移位 按 补码规则 运算

乘积的符号自然形成

② 被乘数任意，乘数为负

乘数$[y]_{补}$，去掉符号位，操作同 ①

最后 加$[-x]_{补}$，校正

③ Booth 算法（被乘数、乘数符号任意） 6.3

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

$[x \cdot y]_{补}$

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

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

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

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

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

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

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

$= y'_0[x]_{补}+2^{-1}(y'_1[x]_{补}+2^{-1}(y'_2[x]_{补}+ \cdots 2^{-1}(y'_n[x]_{补} + 0) \cdots))$

④ Booth 算法递推公式

$$[z_0]_{补} = 0$$

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

$$\vdots$$

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

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

最后一步不移位

如何实现 $+(y_{i+1}-y_i)[x]_{补}$?

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

例6.23 已知 $x = +0.0011$ $y = -0.1011$ 求$[x\,y]_{补}$

6.3

解：

	00.0000	1.0101	0
	+ 11.1101		$+[-x]_{补}$
补码右移	11.1101		
	11.1110	1 1010	1 $\rightarrow$1
	+ 00.0011		$+[x]_{补}$
补码右移	00.0001	1	
	00.0000	11 101	0 $\rightarrow$1
	+ 11.1101		$+[-x]_{补}$
补码右移	11.1101	11	
	11.1110	111 1	0 1 $\rightarrow$1
	+ 00.0011		$+[x]_{补}$
补码右移	00.0001	111	
	00.0000	1111 1	0 $\rightarrow$1
	+ 11.1101		$+[-x]_{补}$
	11.1101	1111	最后一步不移位

$[x]_{补} = 0.0011$

$[y]_{补} = 1.0101$

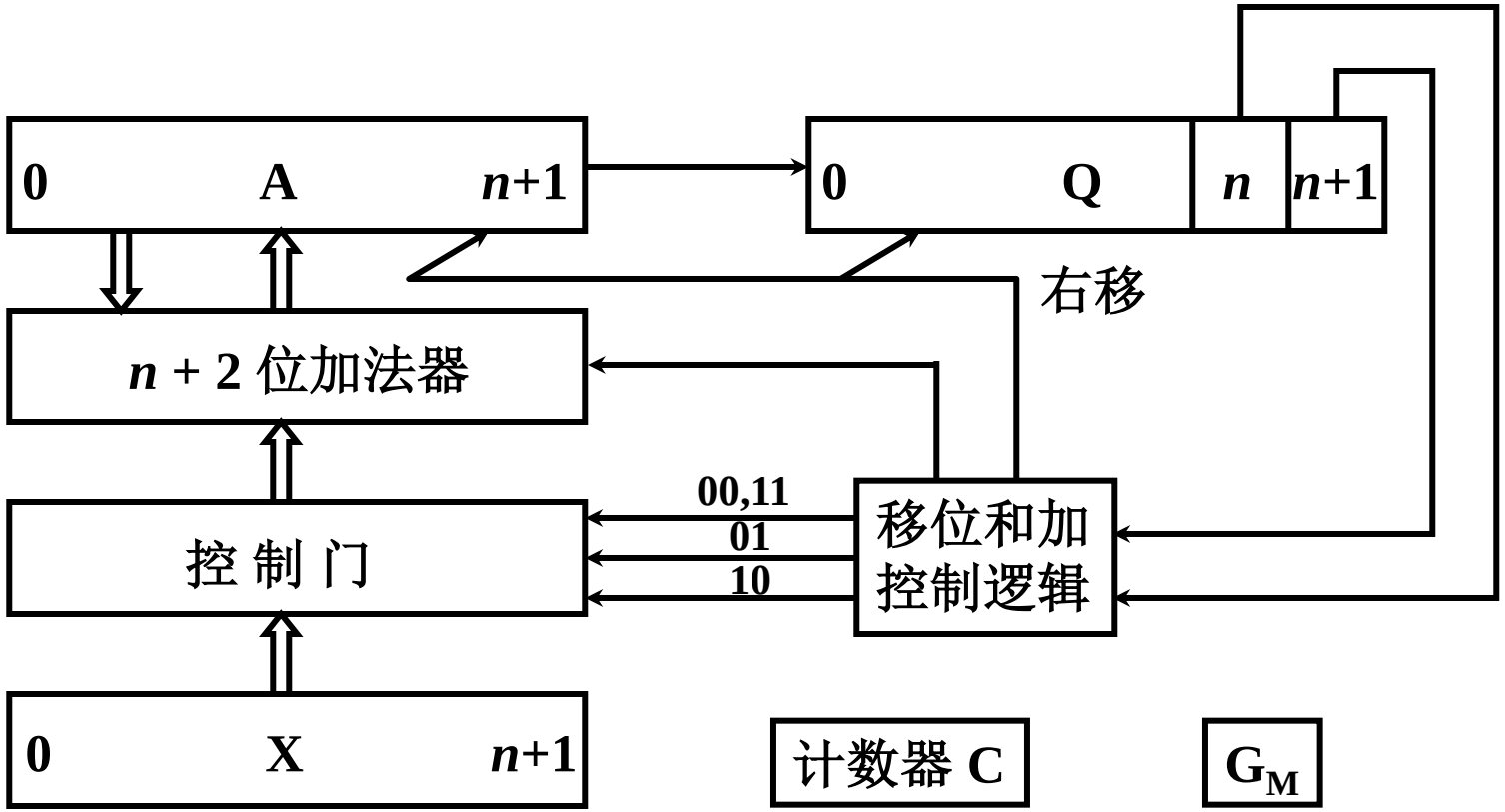
$[-x]_{补} = 1.1101$

$\therefore [x\,y]_{补}$

$= 1.11011111$

(2) Booth 算法的硬件配置

6.3



A、X、Q　均 $n+2$ 位

移位和加法操作受乘数末两位控制

乘法小结

6.3

- 整数乘法与小数乘法过程完全相同
  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

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

- 四、除法运算
  - 1. 笔算除法是怎么做的
  - 2. 如何用计算机硬件来模拟笔算除法的过程
    - 恢复余数法
    - 加减交替法

四、除法运算

6.3

1. 分析笔算除法

$x = -0.1011 \quad y = 0.1101 \quad$ 求 $x \div y$

```
                  0 . 1 1 0 1
              ------------------
0 . 1 1 0 1 ) 0 . 1 0 1 1 0
                0 . 0 1 1 0 1
              ------------------
                0 . 0 1 0 0 1 0
                0 . 0 0 1 1 0 1
              ------------------
                0 . 0 0 0 1 0 1 0 0
                0 . 0 0 0 0 1 1 0 1
              ------------------
                0 . 0 0 0 0 0 1 1 1
```

✓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

$x \div y = -0.1101$　　商符心算求得

余数 0.0000111

2. 笔算除法和机器除法的比较 6.3

笔算除法	机器除法
商符单独处理	符号位异或形成
心算上商	$\|x\|-\|y\|>0$ 上商 1
	$\|x\|-\|y\|<0$ 上商 0
余数 不动 低位补“0” 减右移一位 的除数	余数 左移一位 低位补“0” 减 除数
2 倍字长加法器	1 倍字长加法器
上商位置 不固定	在寄存器 最末位上商

3. 原码除法

以小数为例

$$[x]_{原} = x_0.x_1x_2 \cdots x_n$$

$$[y]_{原} = y_0.y_1y_2 \cdots y_n$$

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

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

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

(1) 恢复余数法

例6.24　$x = -0.1011$　$y = -0.1101$　　求　$[\frac{x}{y}]_{原}$

解：$[x]_{原} = 1.1011$　$[y]_{原} = 1.1101$　$[y^*]_{补} = 0.1101$　$[-y^*]_{补} = 1.0011$

① $x_0 \oplus y_0 = 1 \oplus 1 = 0$

②

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

逻辑左移

逻辑左移

6.3

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		$+[-y^*]_{补}$
1.1101	0110	余数为负，上商 0
+ 0.1101		恢复余数 $+[y^*]_{补}$
0.1010	0110	恢复后的余数
逻辑左移 1.0100	0110	← 1
+ 1.0011		$+[-y^*]_{补}$
0.0111	01101	余数为正，上商 1

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

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

上商 5 次

第一次上商判溢出

移 4 次

余数为正　上商 1

余数为负　上商 0，恢复余数

(2) 不恢复余数法（加减交替法）

6.3

- 恢复余数法运算规则

余数 $R_i>0$ 上商 “1”， $2R_i-y^*$

余数 $R_i<0$ 上商 “0”， R_i+y^* 恢复余数

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

- 不恢复余数法运算规则

上商 “1” $2R_i-y^*$

上商 “0” $2R_i+y^*$

加减交替

例6.25 $x = -0.1011$ $y = -0.1101$ 求 $[\frac{x}{y}]_{原}$ 6.3

解：

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

逻辑左移

逻辑左移

逻辑左移

逻辑左移

$[x]_{原} = 1.1011$

$[y]_{原} = 1.1101$

$[x^*]_{补} = 0.1011$

$[y^*]_{补} = 0.1101$

$[-y^*]_{补} = 1.0011$

例6.25 结果

① $x_0 \oplus y_0 = 1 \oplus 1 = 0$

② $\frac{x^*}{y^*} = 0.1101$

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

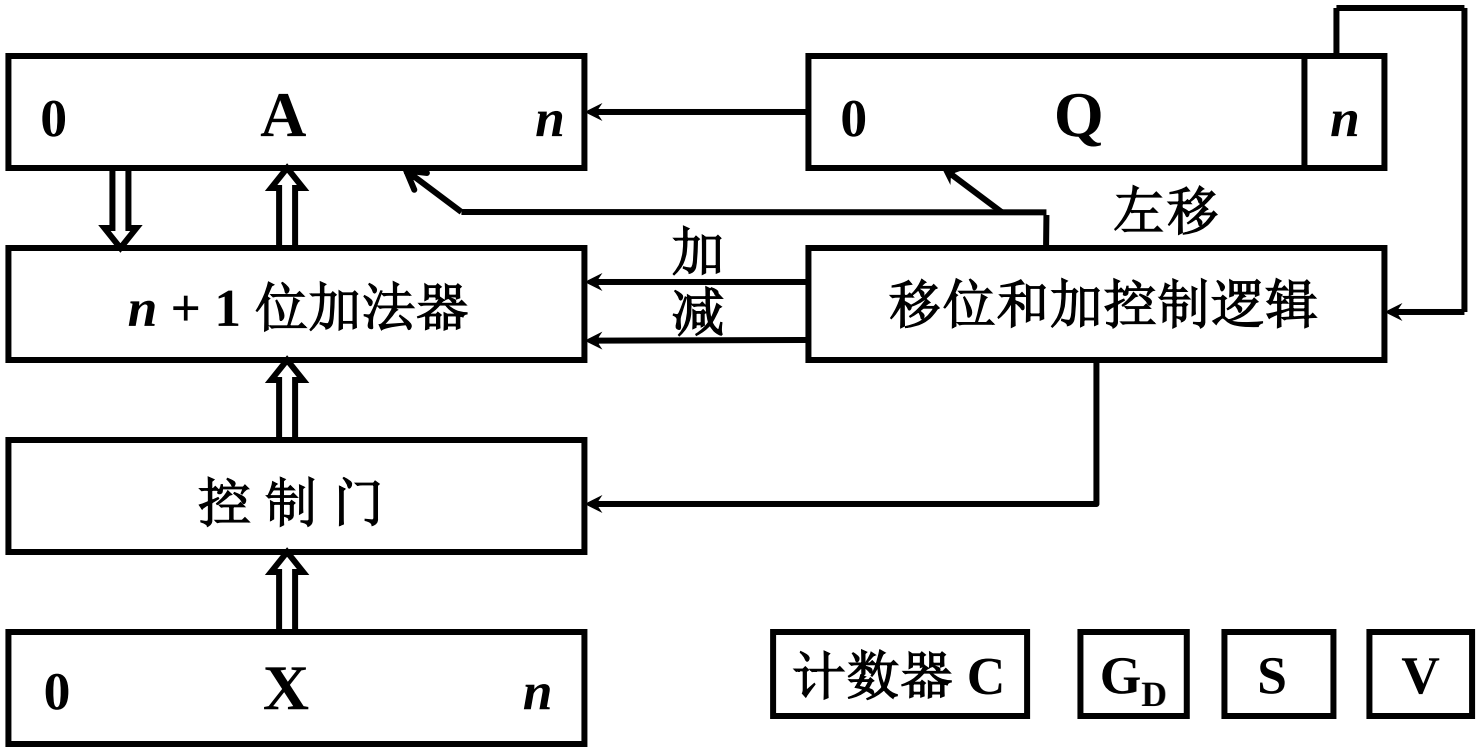
特点 上商 $n+1$ 次

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

(3) 原码加减交替除法硬件配置



A、X、Q 均 $n+1$ 位

用 Q_n 控制加减交替