

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

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 $[y]_{\text{补}}$ ，去掉符号位，操作同 ①

最后 加 $[-x]_{\text{补}}$ ，校正

③ Booth 算法（被乘数、乘数符号任意） 6.3

设 $[x]_{\text{补}} = x_0 \cdot x_1 x_2 \cdots x_n$ $[y]_{\text{补}} = y_0 \cdot y_1 y_2 \cdots y_n$

$$[x \cdot y]_{\text{补}}$$

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

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

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

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

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

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

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

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

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

附加位 y_{n+1}

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

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

④ Booth 算法递推公式

6.3

$$[z_0]_{\text{补}} = 0$$

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

⋮

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

$$[x \cdot y]_{\text{补}} = (y_1-y_0)[x]_{\text{补}} + [z_n]_{\text{补}} \qquad \text{最后一步不移位}$$

如何实现

$$+(y_{i+1}-y_i)[x]_{\text{补}} \text{ ?}$$

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6.23 已知 $x = +0.0011$ $y = -0.1011$ 求 $[x\ y]_{补}$ 6.3

解:

0 0 . 0 0 0 0	1 . 0 1 0 <u>1</u> <u>0</u>	0	$+[-x]_{补}$
+ 1 1 . 1 1 0 1			
1 1 . 1 1 0 1			
1 1 . <u>1</u> 1 1 0	1	1 0 1 0 <u>1</u>	$\rightarrow 1$
+ 0 0 . 0 0 1 1			$+ [x]_{补}$
0 0 . 0 0 0 1	1		
0 0 . <u>0</u> 0 0 0	1 1	1 0 <u>1</u> <u>0</u>	$\rightarrow 1$
+ 1 1 . 1 1 0 1			$+ [-x]_{补}$
1 1 . 1 1 0 1	1 1		
1 1 . <u>1</u> 1 1 0	1 1 1	1 <u>0</u> <u>1</u>	$\rightarrow 1$
+ 0 0 . 0 0 1 1			$+ [x]_{补}$
0 0 . 0 0 0 1	1 1 1		
0 0 . <u>0</u> 0 0 0	1 1 1 1	<u>1</u> <u>0</u>	$\rightarrow 1$
+ 1 1 . 1 1 0 1			$+ [-x]_{补}$
1 1 . 1 1 0 1	1 1 1 1		最后一步不移位

补码右移

补码右移

补码右移

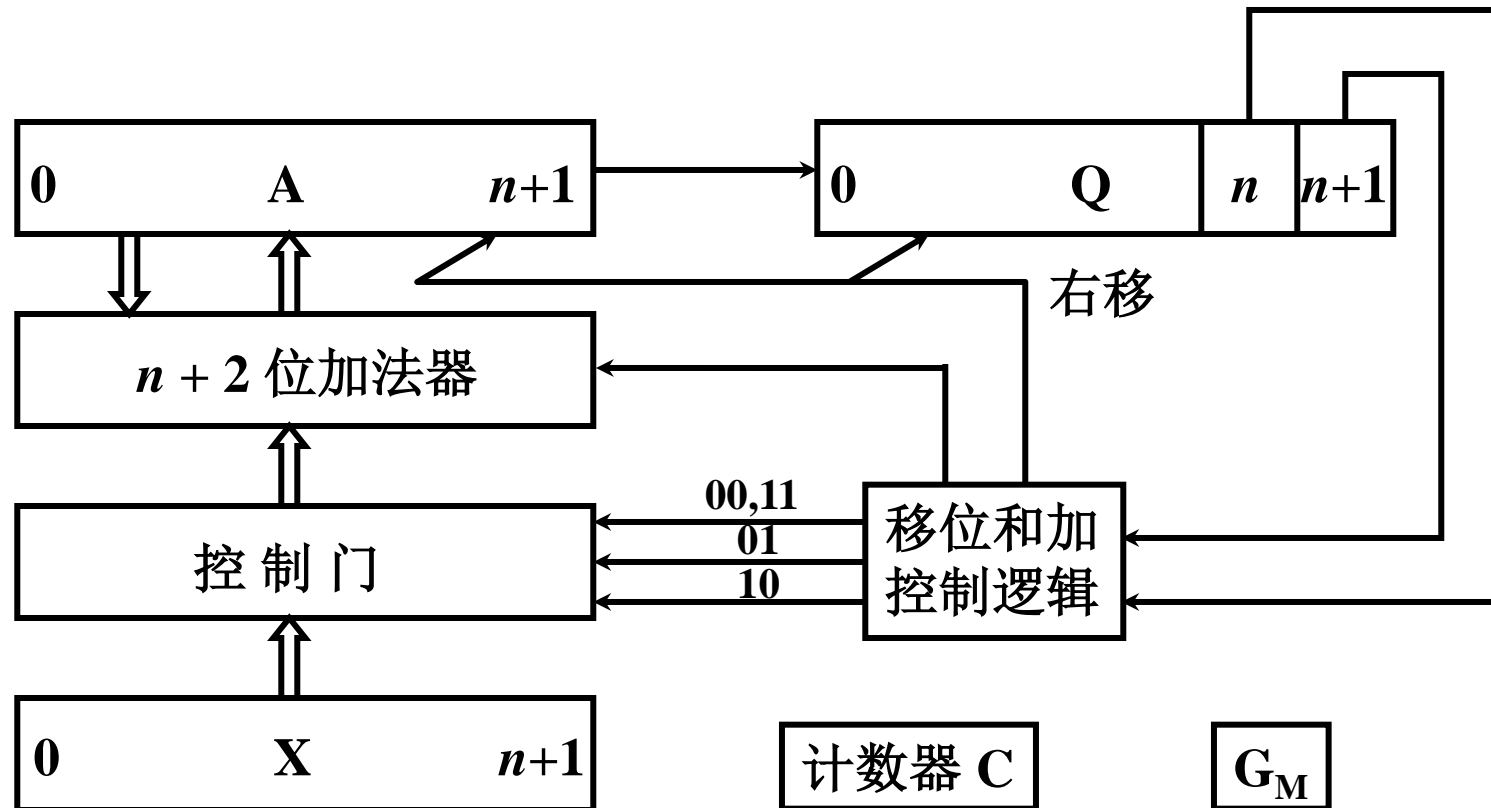
补码右移

$[x]_{补} = 0.0011$
 $[y]_{补} = 1.0101$
 $[-x]_{补} = 1.1101$

$\therefore [x\ y]_{补} = 1.11011111$

(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

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

$$\begin{array}{r} 0.1101 \overline{) 0.1101} \\ \underline{0.1101} \\ 0.0000 \\ \underline{0.0000} \\ 0.0000 \\ \underline{0.0000} \\ 0.0000 \\ \underline{0.0000} \\ 0.0000 \end{array}$$

✓ 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

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

$$\text{余数 } 0.00000111$$

2. 笔算除法和机器除法的比较

6.3

笔算除法

商符单独处理

心算上商

余数越来越长

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

$|x| - |y| > 0$ 上商 1

x余数；y右移一位后的除数

$|x| - |y| < 0$ 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

3. 原码除法

6.3

以小数为例

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

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

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

式中 $x^* = 0.x_1x_2 \dots x_n$ 为 x 的绝对值
 $y^* = 0.y_1y_2 \dots y_n$ 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

(1) 恢复余数法

6.3

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

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

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		$+ [-y^*]_{\text{补}}$
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
逻辑左移 1.0110	0	$\leftarrow 1$
+ 1.0011		$+ [-y^*]_{\text{补}}$
0.1001	01	余数为正, 上商 1
逻辑左移 1.0010	01	$\leftarrow 1$
+ 1.0011		$+ [-y^*]_{\text{补}}$

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
逻辑左移 0.1010	0110	恢复后的余数
1.0100	0110	← 1
+ 1.0011		+[-y*] _补
0.0111	01101	余数为正, 上商 1

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

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

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

(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 $[\frac{x}{y}]_{\text{原}}$

6.3

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

$[x]_{\text{原}} = 1.1011$

$[y]_{\text{原}} = 1.1101$

$[x^*]_{\text{补}} = 0.1011$

$[y^*]_{\text{补}} = 0.1101$

$[-y^*]_{\text{补}} = 1.0011$

例6.25 结果

6.3

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

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

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

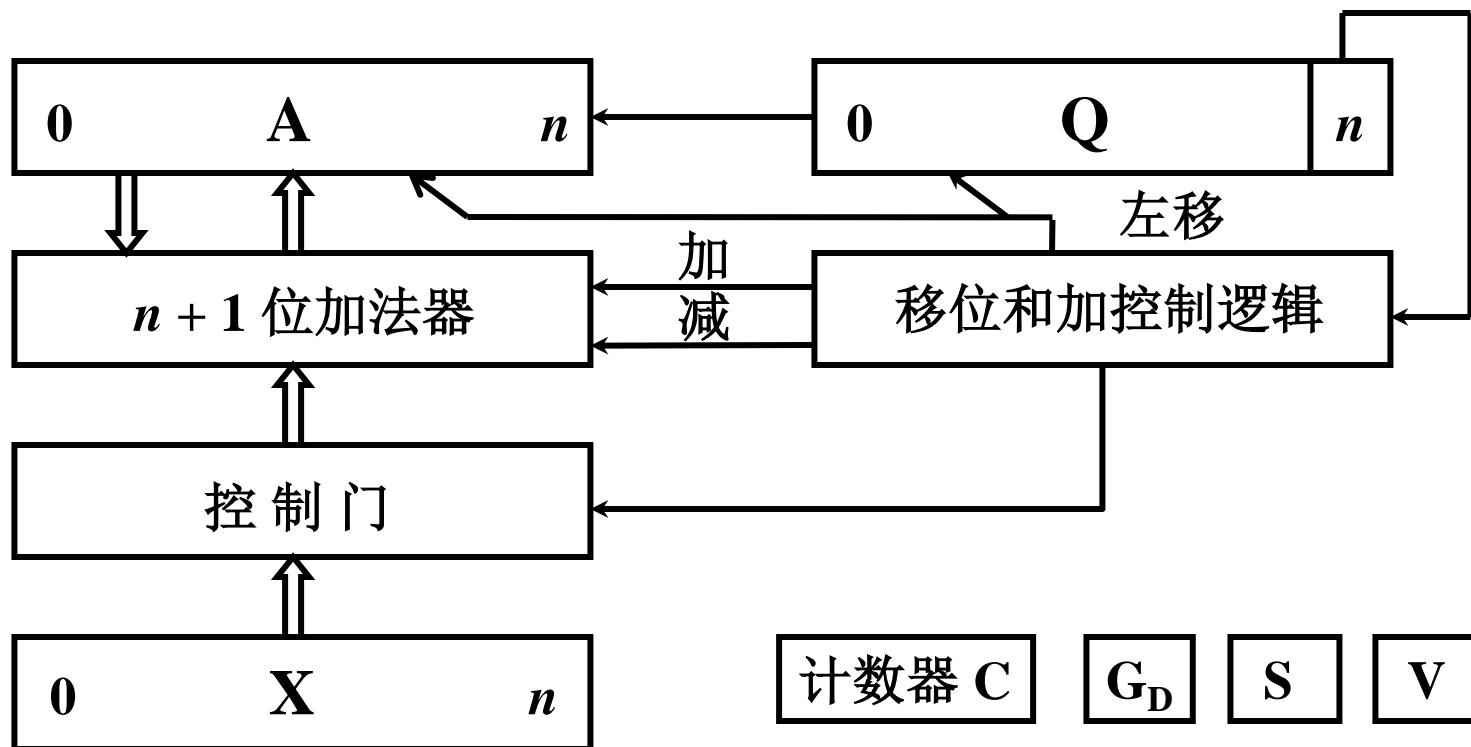
特点 上商 $n+1$ 次

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

(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