第二章 计算机的运算方法

- 计算机中数的表示
- 定点运算
  - 加减法运算
  - 一位乘法运算
  - booth算法
  - 除法运算 （不考，自己看MOOC）
- 浮点运算

算术移位规则（回顾）

无论正负，算术移位，符号位不变

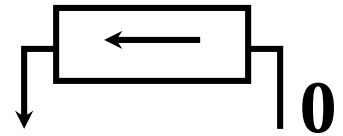
真值	码　制	添补代码
正数	原码、补码	0
负数	原　码	0
	补　码	左移 添 0
		右移 添 1

实际上，补码**算术右移**就是用符号位填充空出来的位

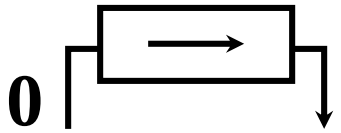
算术移位和逻辑移位的区别

算术移位是有符号数的移位，其特点是符号位不变
逻辑移位是无符号数的移位，其特点是左或右补0

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）

为了防止高位弄丢，可以设计出用硬件C_y来存储高位。

高位 1 移丢

已知$[y]_{补}$，求$[-y]_{补}$（回顾）

解：　设 $[y]_{补} = y_0.y_1y_2\cdots y_n$

<I>　$[y]_{补} = 0.y_1y_2\cdots y_n$

$$y = 0.y_1y_2\cdots y_n$$

$$-y = -0.y_1y_2\cdots y_n$$

$$[-y]_{补} = 1.\overline{y_1}\,\overline{y_2}\cdots\overline{y_n} + 2^{-n}$$

<II>　$[y]_{补} = 1.y_1y_2\cdots y_n$

$$[y]_{原} = 1.\overline{y_1}\,\overline{y_2}\cdots\overline{y_n} + 2^{-n}$$

$$y = -(0.\overline{y_1}\,\overline{y_2}\cdots\overline{y_n} + 2^{-n})$$

$$-y = 0.\overline{y_1}\,\overline{y_2}\cdots\overline{y_n} + 2^{-n}$$

$$[-y]_{补} = 0.\overline{y_1}\,\overline{y_2}\cdots\overline{y_n} + 2^{-n}$$

重要结论：

$[y]_{补}$连同符号位在内，每位取反，末位加 1，即得$[-y]_{补}$。注意：特例不适用。

第二章 计算机的运算方法

- 计算机中数的表示
- 定点运算
  - **加减法运算**
  - 一位乘法运算
  - booth算法
- 浮点运算

定点加减法运算（补码）

●加法

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

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

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

补码运算连同符号位一起相加，符号位产生的进位自然丢掉

补码加减法口诀

- **化减法为加法**

  即：$[A-B]_{补}=[A+(-B)]_{补}=A_{补}+[-B]_{补}$

- **补码符号位参与运算**

- **对于1位符号位的补码加法**

  符号位产生的进位丢掉

例：用补码运算求$A+B$

- 设 $A = 0.1011$，$B = -0.0101$，求 $A+B$

解：

$$[A]_{补} = \quad 0.1011$$
$$+[B]_{补} = \quad 1.1011$$
$$[A]_{补} + [B]_{补} = 1\,0.0110 = [A+B]_{补}$$

丢掉　$\therefore A+B = 0.0110$

验证

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

- 设$A=-9$，$B=-5$，（设A和B位数为4），求$A+B$

解：

$$[A]_{补} = \quad 1,0111$$
$$+[B]_{补} = \quad 1,1011$$
$$[A]_{补} + [B]_{补} = 1\,1,0010 = [A+B]_{补}$$

丢掉　$\therefore A+B = -1110 = -14$

验证

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

例：用补码运算求*A-B*

•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： $A = 15 = 0001111$

$B = 24 = 0011000$

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

$+[-B]_{补} = 1, 1101000$

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

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

例：用补码运算求*A-B*

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

解：　$A = -97 = -1100001$

$B = 41 = 0101001$

$[A]_{补} = 1, 0011111$

$+[-B]_{补} = 1, 1010111$

$[A]_{补} + [-B]_{补} = 10, 1110110 = [A - B]_{补}$

$\therefore A - B = +118$ （溢出）

一位符号位判溢出

- 参加**加法运算**的两个数（减法时+减数相反数的补码）
  - 如果符号不同，**不会溢出**。
  - 如果符号相同（同正或同负），**其结果的符号与原操作数的符号不同，即为溢出**
- 硬件实现（异或门）
  - 最高有效位的进位⊕符号位的进位 = 1，溢出

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

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

补码加减法中双符号位判溢出

- 变形补码

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

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

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

规定
双0为正
双1为负

结果的双符号位 相同　未溢出

00, ×××××　　00. ×××××
11, ×××××　　11. ×××××

结果的双符号位 不同　溢出

10, ×××××　　10. ×××××
01, ×××××　　01. ×××××

双符号位补码，最高符号位 代表其 真正的符号

例：用补码运算求A-B

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

解：　$A = -97 = -1100001$

$B = 41 = 0101001$

$[A]_{补} = 1, 0011111$

$+[-B]_{补} = 1, 1010111$

$[A]_{补} + [-B]_{补} = 10, 1110110 = [A - B]_{补}$

丢掉

$\therefore A - B = +118$ （溢出）

双符号位判断补码运算溢出

•设机器数字长为 9 位（含 2 位符号位）且 $A = -97$，$B = 41$，用补码求 $A - B$。

解：　$A = -97 = -1100001$

$B = 41 = 0101001$

$[A]_{补} = 11, 0011111$

$+[-B]_{补} = 11, 1010111$

$[A]_{补} + [-B]_{补} = 110, 1110110 = [A - B]_{补}$

丢掉

$\therefore A - B = +118$ （溢出）

第三章 计算机的运算方法

- 计算机中数的表示
- 定点运算
  - 加减法运算
  - 一位乘法运算
  - booth算法
  - 除法运算 （不考，自己看MOOC）
- 浮点运算

乘法运算——笔算乘法

$A = -0.1101$　　$B = 0.1011$

$A \times B = -0.10001111$　　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位的积一起相加
- ✓ 乘积的位数扩大一倍

- 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

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

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 + 0倍或1倍的被乘数　　③

⋮

第八步　右移一位，得结果　　⑧

乘法运算——机器思维

$A = -\ 0.1101$

$B = \ 0.1011$

$A \times B = -\ 0.10001111$

符号位单独处理，所以先对A和B的绝对值进行运算：

手算

$$
\begin{array}{r}
0.1101 \\
\times\ 0.1011 \\
\hline
1101 \\
1101 \\
0000 \\
1101 \\
\hline
0.10001111
\end{array}
$$

机器计算

$$
\begin{array}{l}
0.0000\quad 1011 \\
+\ 0.1101 \\
\hline
0.1101 \\
0.0110\quad 1 \quad \rightarrow \\
+\ 0.1101 \\
\hline
1.0011\quad 1 \\
0.1001\quad 11 \quad \rightarrow \\
+\ 0.0000 \\
\hline
0.1001\quad 11 \\
0.0100\quad 111 \quad \rightarrow \\
+\ 0.1101 \\
\hline
1.0001\quad 111 \\
0.1000\quad 1111 \quad \rightarrow
\end{array}
$$

- 改进后的笔算乘法过程（竖式，参考唐书P244）

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

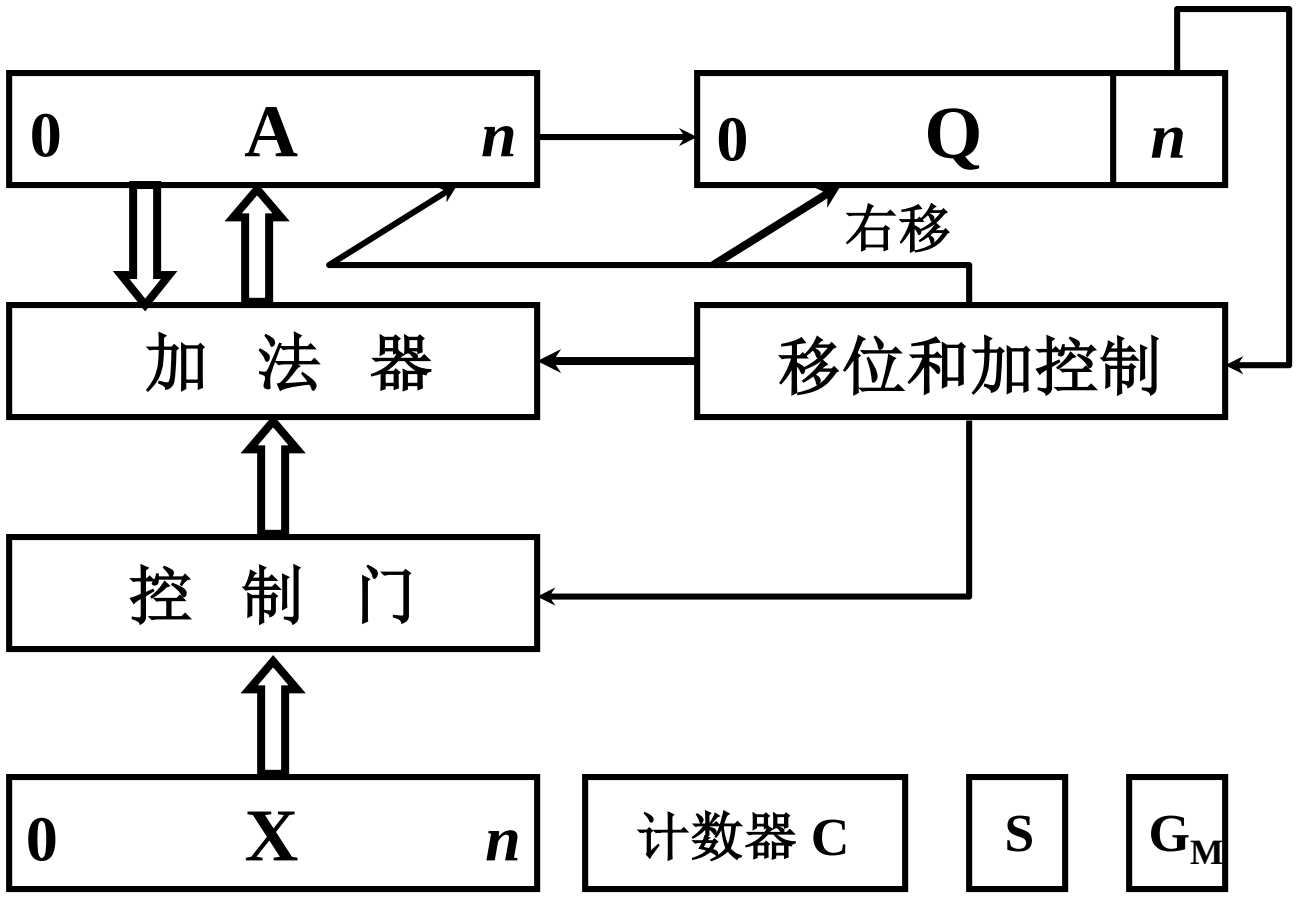
改进的笔算乘法小结

- 两个n位数乘法运算可用 **加n次和移位n次** 实现
- 由乘数的**末位决定** 被乘数**是否**与原部分积**相加**
- **乘数右移1位**（末位舍弃），空出高位存放部分积的低位，**同时**部分积 **右移 1 位** 形成新的部分积。
- 被乘数只与部分积的**高n位**相加
- 硬件需求：3个寄存器(具有移位功能), 1个全加器

- 改进后的笔算乘法过程（竖式，参考唐书P244）

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

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

原码乘法

- 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 $x^* \cdot y^*$

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. 已知 $\boldsymbol{x} = -0.1110$，$y = 0.1101$，求$[x \times y]_{原}$

解：第1步：写出原码、绝对值和符号位

$[x]_{原} = 1.1110$，$x^* = 0.1110$（为绝对值），$x_0 = 1$

$[y]_{原} = 0.1101$，$y^* = 0.1101$（为绝对值），$y_0 = 0$

第2步：$x^* \cdot y^*$计算过程，见下页PPT

第2步：$x^* \cdot y^*$计算过程

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

- **第3步：计算结果**

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

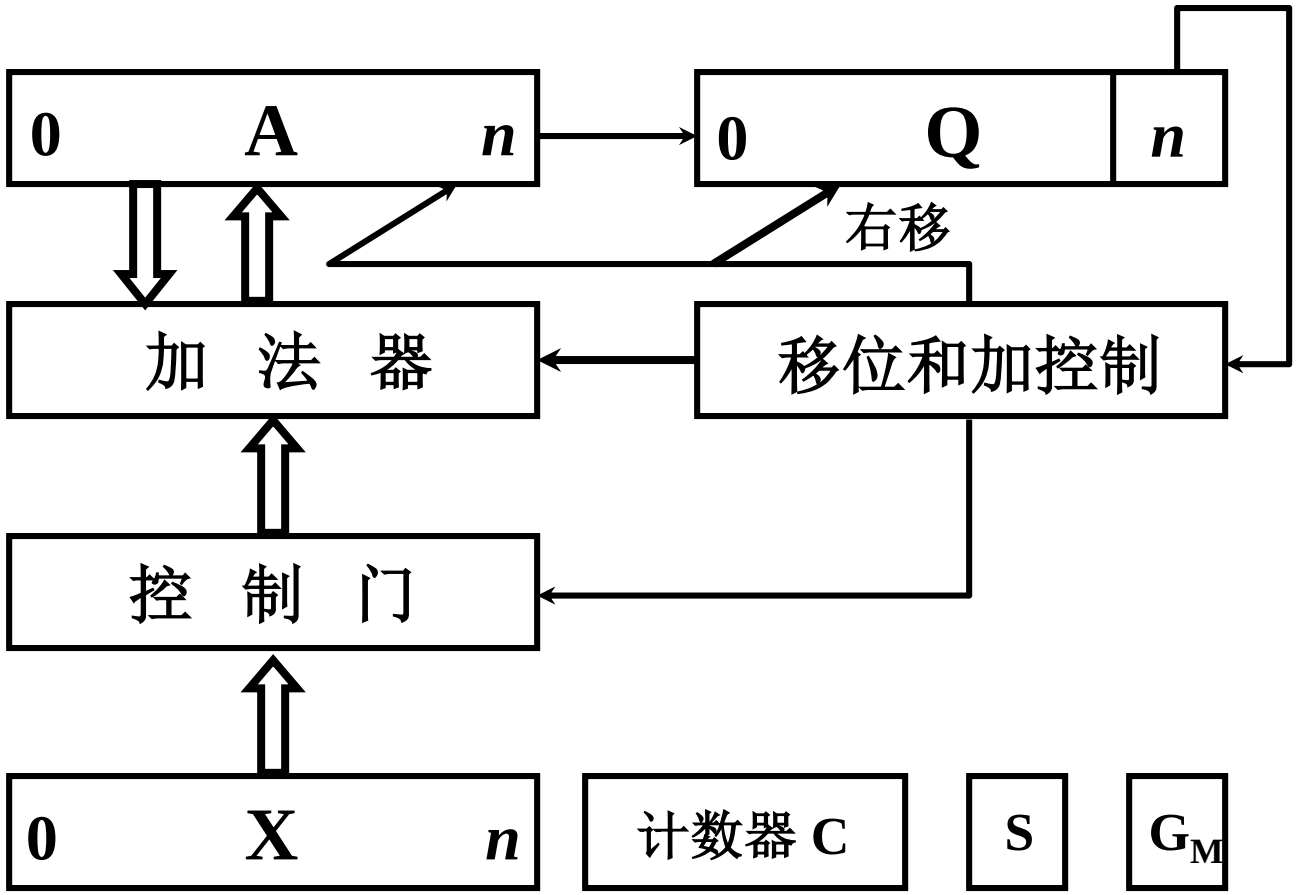
用移位的次数判断乘法是否结束

逻辑移位

算法	加法次数	移位次数	移位	符号是否参与运算
原码一位乘	n	n	部分积　逻辑右移	否

部 分 积	乘 数	说 明
0.0000 + 0.1110	1101	部分积 初态 $z_0 = 0$ $+ x^*$
逻辑右移 0.1110 0.0111 + 0.0000	0110	→1，得 z_1 + 0
逻辑右移 0.0111 0.0011 + 0.1110	0 1011	→1，得 z_2 $+ x^*$
逻辑右移 1.0001 0.1000 + 0.1110	10 1101	→1，得 z_3 $+ x^*$
逻辑右移 1.0110 0.1011	110 0110	→1，得 z_4

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

补码一位乘法

- 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

同原码乘　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

补码的正确推导（回顾）

1）小数补码：

由定义可得：$[-y]_{补}+[y]_{补}=2$

推导可得：$[-y]_{补}=2-[y]_{补}$

补码表示法：二进制（纯）小数

$$[x]_{补}=\begin{cases} x & 1>x\geq 0 \\ 2+x & 0>x\geq -1\ (\mathrm{mod}\ 2) \end{cases}$$

其中：x 为真值

2）整数补码：

由定义可得：$[-x]_{补}+[x]_{补}=2^{n+1}$

推导可得：$[-x]_{补}=2^{n+1}-[x]_{补}$

补码表示法：二进制整数

$$[x]_{补}=\begin{cases} 0,x & 2^n>x\geq 0 \\ 2^{n+1}+x & 0>x\geq -2^n\ (\mathrm{mod}\ 2^{n+1}) \end{cases}$$

其中：x 为真值，n 为二进制整数的位数

补码一位乘法推导

① 被乘数任意，乘数y为正

$[x]_{补} = x_0.\ x_1x_2 \cdots x_n = 2 + x = 2^{n+1} + x \pmod 2$

$[y]_{补} = 0.\ y_1y_2 \cdots y_n$

$[x]_{补} = x_0.\ x_1x_2 \cdots x_n$

$[y]_{补} = y_0.\ y_1y_2 \cdots y_n$

$[x]_{补} \cdot [y]_{补} = [x]_{补} \cdot y = (2^{n+1} + x) \cdot y = 2^{n+1} \cdot y + x \cdot y$

$2^{n+1} \cdot y = 2\sum_{i=1}^{n} y_i\, 2^{n-i}$ 且 $\sum_{i=1}^{n} y_i\, 2^{n-i}$ 是一个大于等于 1的正数。

则 $2^{n+1} \cdot y = 2 \pmod 2$

$[x]_{补} \cdot [y]_{补} = 2^{n+1} \cdot y + x \cdot y = 2 + xy = [x \cdot y]_{补}$

即 $[x \cdot y]_{补} = [x]_{补} \cdot y$

同原码乘　但 加 和 移位 按 补码规则 运算

乘积的符号自然形成

补码一位乘法推导

② 被乘数任意，乘数y为负

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

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

$[y]_{补} = 1.\ y_1y_2 \ldots y_n = 2 + y \pmod 2$

$y = [y]_{补} - 2 = 1.\ y_1y_2 \ldots y_n - 2 = 0.\ y_1y_2 \ldots y_n - 1$

$x \cdot y = x(0.\ y_1y_2 \ldots y_n - 1) = x(0.\ y_1y_2 \ldots y_n) - x$

$[x \cdot y\ ]_{补} = [x(0.\ y_1y_2 \ldots y_n)\ ]_{补} + [-x]_{补}$

$[x(0.\ y_1y_2 \ldots y_n)\ ]_{补} = [x]_{补}(0.\ y_1y_2 \ldots y_n)$ （与① 相同）

$[x \cdot y\ ]_{补} = [x]_{补}(0.\ y_1y_2 \ldots y_n) + [-x]_{补}$

乘数$[y]_{补}$，去掉符号位，操作同 ①

最后 加$[-x]_{补}$，校正

补码一位乘法

- 补码一位乘运算规则

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

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

① 被乘数任意，乘数为正

$[x \cdot y\ ]_{补} = [x\ ]_{补} \cdot y$

② 被乘数任意，乘数为负

$[x \cdot y\ ]_{补} = [x]_{补}(0.\ y_1y_2 \ \ldots\ y_n) + [-x]_{补}$

统一算法

$[x \cdot y\ ]_{补} = [x]_{补}(0.\ y_1y_2 \ \ldots\ y_n) + [-x]_{补} \cdot y_0$

$= [x]_{补}(0.\ y_1y_2 \ \ldots\ y_n) + (2-[x]_{补})\ y_0$

$[-x]_{补}+[x]_{补}=2$

补码的正确推导

1）小数补码：

由定义可得：$[-y]_{补}+[y]_{补}=2$

推导可得：$[-y]_{补}=2-[y]_{补}$

补码表示法：二进制（纯）小数

$$[x]_{补}=\begin{cases} x & 1>x\geq 0 \\ 2+x & 0>x\geq -1\ (\mathrm{mod}\ 2) \end{cases}$$

其中：x 为真值

2）整数补码：

由定义可得：$[-x]_{补}+[x]_{补}=2^{n+1}$

推导可得：$[-x]_{补}=2^{n+1}-[x]_{补}$

补码表示法：二进制整数

$$[x]_{补}=\begin{cases} 0,x & 2^n>x\geq 0 \\ 2^{n+1}+x & 0>x\geq -2^n\ (\mathrm{mod}\ 2^{n+1}) \end{cases}$$

其中：x 为真值，n 为二进制整数的位数

单选题 1分

设置

关于$[X]_{补}$和$[-X]_{补}$，哪个表述正确？

A $-[X]_{补}=[-X]_{补}$

B $[X]_{补}=[-X]_{补}$

C $[X]_{补}$和$[-X]_{补}$没有任何关系

A选项（唐书P253）这个结论不严谨

正确使用是在运算过程中化减为加替换：算式$-[X]_{补}$=算式$+[-X]_{补}$

而且只能单向替换（即$+[-X]_{补}$能替换$-[X]_{补}$，反之不行）

提交

第二章 计算机的运算方法

- 计算机中数的表示
- 定点运算
  - 加减法运算
  - 一位乘法运算
  - booth算法
  - 除法运算 （不考，自己看MOOC）
- 浮点运算

Booth 算法（被乘数、乘数符号任意）

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

$[x \cdot y]_{补}$

$2-[x]_{补} = +[-x]_{补}$

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

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

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

拆解

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

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

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

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

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

附加位 $y_{n+1}=0$

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

Booth 算法递推公式

$$[z_0]_{补} = 0$$

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

$$\vdots$$

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

$$[x \cdot y]_{补} = [z_n]_{补} + (y_1 - y_0)[x]_{补} + 2y_0$$ 最后一步不移位

如何实现 $y_{i+1} - y_i$?

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

Booth算法的三个计算要诀（重要）

1）双符号位运算

2）乘数前有符号位，后有附加0，
两两比较决定怎么加

3）最后一步不移位

注意：Booth算法是补码运算，
所以移位是算术右移

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

解：

00.0000	1.0101 0	
+ 11.1101		$+[-x]_{补}$
11.1101		
11.1110	1 1010 1	→1
+ 00.0011		$+[x]_{补}$
00.0001	1	
00.0000	11 101 0	→1
+ 11.1101		$+[-x]_{补}$
11.1101	11	
11.1110	111 10 1	→1
+ 00.0011		$+[x]_{补}$
00.0001	111	
00.0000	1111 1 0	→1
+ 11.1101		$+[-x]_{补}$
11.1101	1111	最后一步不移位

补码右移

$[x]_{补} = 0.0011$

$[y]_{补} = 1.0101$

$[-x]_{补} = 1.1101$

$\therefore [x \cdot y]_{补}$

$= 1.11011111$

• 例. 已知 $[x]_{补} = 1.0101$， $[y]_{补} = 1.0011$， 求$[x \times y]_{补}$

解：

00.0000	1.0011	0	
+ 00.1011			$+[-x]_{补}$
00.1011			
00.0101	1 1001	1	→1
00.0010	11 100	1	→1
+ 11.0101			$+[x]_{补}$
11.0111	11		
11.1011	111 10	0	→1
11.1101	1111 1	0	→1
+ 00.1011			$+[-x]_{补}$
00.1000	1111		最后一步不移位

补码右移

补码右移

补码右移

补码右移

$[-x]_{补} = 0.1011$

$\therefore [x \cdot y]_{补}$

$=0.10001111$

算法	加法次数	移位次数	移位	符号是否参与运算
Booth算法	n+1	n	部分积　**算数右移**	是

- 例. 已知 $[x]_{补} = 1.0101$，$[y]_{补} = 1.0011$，求$[x \times y]_{补}$

解：

00.0000	1.0011 0	
+ 00.1011		$+[-x]_{补}$
00.1011		
00.0101	1 1001 1	→1
00.0010	11 100 1	→1
+ 11.0101		$+[x]_{补}$
11.0111	11	
11.1011	111 10 0	→1
11.1101	1111 1 0	→1
+ 00.1011		$+[-x]_{补}$
00.1000	1111	最后一步不移位

补码右移

补码右移

补码右移

补码右移

$[-x]_{补} = 0.1011$

$\therefore [x \cdot y]_{补}$

$=0.10001111$

乘法小结

- **整数乘法**与**小数乘法**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

第二章 计算机的运算方法

- 计算机中数的表示
- 定点运算
  - 加减法运算
  - 一位乘法运算
  - booth算法
  - 除法运算 （不考，自己看MOOC）
- 浮点运算

浮点四则运算

- 一、浮点加减运算

$$x = S_x \cdot 2^{j_x} \qquad y = S_y \cdot 2^{j_y}$$

1. 对阶

(1) 求阶差

$$\Delta j = j_x - j_y = \begin{cases} = 0 & j_x = j_y \quad \text{已对齐} \\ > 0 & j_x > j_y \begin{cases} x \text{ 向 } y \text{ 看齐} & S_x \leftarrow 1,\ j_x - 1 \\ y \text{ 向 } x \text{ 看齐} & \checkmark\ S_y \rightarrow 1,\ j_y + 1 \end{cases} \\ < 0 & j_x < j_y \begin{cases} x \text{ 向 } y \text{ 看齐} & \checkmark\ S_x \rightarrow 1,\ j_x + 1 \\ y \text{ 向 } x \text{ 看齐} & S_y \leftarrow 1,\ j_y - 1 \end{cases} \end{cases}$$

(2) 对阶原则

小阶向大阶看齐（有可能带来精度损失）

• 例：$x = 0.1101\times2^{01}, y = (-0.1010)\times2^{11}$，求$x+y$

解：$[x]_{补} = 00, 01; 00.1101$　　$[y]_{补} = 00, 11; 11.0110$

1. 对阶

① 求阶差　$[\Delta j]_{补} = [j_x]_{补} - [j_y]_{补}$

$$\begin{array}{rr} & 00, 01 \\ + & 11, 01 \\ \hline & 11, 10 \end{array}$$

阶差为负（-2）　　$\therefore S_x \longrightarrow 2$　$j_x + 2$

② 对阶　　$[x]_{补}' = 00, 11; 00.0011$

2. 尾数求和

$$\begin{array}{rll} & [S_x]_{补}' & = 00.0011 \\ + & [S_y]_{补} & = 11.0110 \\ \hline & & 11.1001 \end{array}$$

对阶后的$[S_x]_{补}$

$\therefore [x+y]_{补} = 00, 11; 11.1001$

3. 规格化

- (1) 规格化数的定义

$$r = 2 \qquad \frac{1}{2} \leq |S| < 1$$

- (2) 规格化数的判断

$S>0$	规格化形式	$S<0$	规格化形式
真值	$0.1\times\times\cdots\times$	真值	$-0.1\times\times\cdots\times$
原码	$0.1\times\times\cdots\times$	原码	$1.1\times\times\cdots\times$
补码	$0.1\times\times\cdots\times$	补码	$1.0\times\times\cdots\times$
反码	$0.1\times\times\cdots\times$	反码	$1.0\times\times\cdots\times$

原码　不论正数、负数，第一数位为1

补码　符号位和第 一数位不同

补码规格化要求：符号位和数值位不同

$$S = -\frac{1}{2} = -0.100\cdots0$$

$$[S]_{原} = 1.100\cdots0$$

$$[S]_{补} = \boxed{1.1}00\cdots0$$

∴ $[-\frac{1}{2}]_{补}$ 不是规格化的数

$$S = -1$$

$$[S]_{补} = \boxed{1.0}00\cdots0$$

∴ $[-1]_{补}$ 是规格化的数

- (3)左规

尾数左移一位，阶码减 1，直到数符和第一数位不同为止

上例　$[x+y]_{补} = 00,\ 11;\ 11.\ 1001$

左规后　$[x+y]_{补} = 00,\ 10;\ 11.\ 0010$

$$\therefore\ x + y = (-0.1110) \times 2^{10}$$

- (4)右规

当 尾数双符号位不同时，需 右规

即尾数出现 01.××…×或 10.××…×时

尾数右移一位，阶码加 1

- 例. $x = 0.1101 \times 2^{10}$，$y = 0.1011 \times 2^{01}$，求 $x+y$
  （除阶符、数符外，阶码取 3 位，尾数取 6 位）

解：$[x]_{补} = 00,010;\ 00.110100$

$[y]_{补} = 00,001;\ 00.101100$

① 对阶

$[\Delta j]_{补} = [j_x]_{补} - [j_y]_{补}$ $= 00,010$

$+\ 11,111$

$100,001$

阶差为 +1　　$\therefore S_y \longrightarrow 1,\ j_y+1$

$\therefore [y]_{补}' = 00,010;\ 00.010110$

② 尾数求和

$[S_x]_{补} = 00.110100$

$+\ [S_y]_{补}' = 00.010110$　　对阶后的$[S_y]_{补'}$

01.001010　　尾数溢出需右规

③ 右规

$[x+y]_{补}$ = 00, 010; 01. 001010

右规后

$[x+y]_{补}$ = 00, 011; 00. 100101

∴ $x+y = 0.100101 \times 2^{11}$

- 4. 舍入
  - 在 **对阶** 和 **右规** 过程中，可能出现尾数末位丢失引起误差，需考虑舍入
    (1)0 舍 1 入法
    (2)恒置 “1” 法

- 例. $x = (-\frac{5}{8}) \times 2^{-5}$，$y = (\frac{7}{8}) \times 2^{-4}$，求 $x-y$（除阶符、数符外，阶码取 3 位，尾数取 6 位）

解：$x = (-0.101000) \times 2^{-101}$　　$y = (0.111000) \times 2^{-100}$

$[x]_{补} = 11,011;\ 11.011000$　　$[y]_{补} = 11,100;\ 00.111000$

① 对阶

$$[\Delta j]_{补} = [j_x]_{补} - [j_y]_{补} = 11,011 + 00,100 = 11,111$$

阶差为 –1　　∴ $S_x \longrightarrow 1$，$j_x + 1$

∴ $[x]_{补}' = 11,100;\ 11.101100$

② 尾数求和

$$[S_x]_{补}' = 11.101100$$
$$+\ [-S_y]_{补} = 11.001000$$
$$\overline{\qquad\qquad\qquad\qquad}$$
$$110.110100$$

③ 右规

$$[x-y]_{补} = 11,100;\ 10.110100$$

右规后

$$[x-y]_{补} = 11,101;\ 11.011010$$

$$\therefore\ x-y = (-0.100110)\times 2^{-11}$$

$$= \left(-\frac{19}{32}\right)\times 2^{-3}$$

溢出判断

- 设机器数为补码，尾数为 **规格化形式**，并假设阶符取 2 位，阶码的数值部分取 7 位，数符取 2 位，尾数取 n 位，则该 **补码** 在数轴上的表示为