计算机组成原理

群名称:2022春-计组4班
群　号:139369553

文杰
计算机科学与技术学院
wenjie@hit.edu.cn
个人主页：http://faculty.hitsz.edu.cn/wenjie

计算机的运算方法

- 定点运算
  - 加减法运算
  - 一位乘法运算
  - booth算法
  - 除法运算
- 浮点运算

加减法运算

- 补码加减运算公式

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

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

(1) 加法

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

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

(2) 减法

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

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

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

连同符号位一起相加，符号位产生的进位自然丢掉

例：用补码运算求$A+B$

- 设$A = 0.1011$，$B = -0.0101$，用补码运算求$A+B$ = ？

解：

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

$\therefore A + B = 0.0110$

验证

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

- 设$A=-9$，$B=-5$（设A和B数值位数为4），求$A+B$

解：

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

$\therefore A + B = -1110$

验证

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

单选题 1分

设置

已知$[X]_{补}$，以下哪个方法可以求出$[-X]_{补}$？

A 包括符号为在内按位取反

B 包括符号位在内，每位取反，末位加 1

C 除符号位外，每位取反，末位加 1

D 符号为取反，其他保持不变

提交

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

提交

*例：已知小数$[y]_{补}$，求$[-y]_{补}$

设$[y]_{补} = y_0.y_1y_2\cdots y_n$，根据符号位$y_0$为0或1分开讨论

<I>

$[y]_{补} = 0.y_1y_2\ \ldots y_n$

$[y]_{原} = 0.y_1y_2\ \cdots y_n$

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

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

$-[y]_{补} = -0.y_1y_2\ \ldots\ y_n$

$\equiv 2-0.y_1y_2\ \ldots\ y_n \pmod 2$

$= 1.\overline{y_1}\ \overline{y_2}\ \cdots\ \overline{y_n} + 2^{-n}$

<II>

$[y]_{补} = 1.y_1y_2\ \cdots y_n$

$[y]_{原} = 1+(0.\overline{y_1}\ \overline{y_2}\ \cdots\ \overline{y_n} + 2^{-n})$

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

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

$[y]_{补} = 1.y_1y_2\ \ldots y_n$

$\equiv -(0.\overline{y_1}\ \overline{y_2}\ldots\ \overline{y_n} + 2^{-n}) \pmod 2$

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

$[y]_{补}$连同符号位在内，每位取反，末位加 1，即得$[-y]_{补}$

观察发现 $[-y]_{补} + [y]_{补} = 2 \longrightarrow 0 \pmod 2$　　$[-y]_{补} = -[y]_{补}$

*例：已知整数$[X]_{补}$，求$[-X]_{补}$

设$[X]_{补} = X_n, X_{n-1}\cdots X_1X_0$，根据符号位$X_n$为0或1分开讨论

<I> $[X]_{补} = 0, X_{n-1}X_{n-2}\cdots X_1X_0$

$X_{原} = 0, X_{n-1}X_{n-2}\cdots X_1X_0$

$[-X]_{原} = 1, X_{n-1}X_{n-2}\cdots X_1X_0$

$[-X]_{补} = 1, (\overline{X}_{n-1}\overline{X}_{n-2}\cdots\overline{X}_1\overline{X}_0 + 1)$

<II> $[X]_{补} = 1, X_{n-1}X_{n-2}\cdots X_1X_0$

$X_{原} = 1, (\overline{X}_{n-1}\overline{X}_{n-2}\cdots\overline{X}_1\overline{X}_0 + 1)$

$[-X]_{原} = 0, (\overline{X}_{n-1}\overline{X}_{n-2}\cdots\overline{X}_1\overline{X}_0 + 1)$

$[-X]_{补} = 0, (\overline{X}_{n-1}\overline{X}_{n-2}\cdots\overline{X}_1\overline{X}_0 + 1)$

$[X]_{补}$连同符号位在内，每位取反，末位加 1，即得$[-X]_{补}$

观察发现 $[-X]_{补} + [X]_{补} = 2^{n+1} \rightarrow 0 \pmod{2^{n+1}}$ $[-X]_{补} = -[X]_{补}$

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 $A - B$。

$$[A - B]_{补} = [A]_{补} + [-B]_{补}$$

解：$A = 15 = 0001111$

$B = 24 = 0011000$

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

$+[-B]_{补} = 1, 1101000$

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

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

投票

最多可选1项

设置

设机器数字长为 8 位（含 1 位符号位）且 $A = -97$，$B = 41$，用补码求 $A - B$（8位补码表示）。 （97= 01100001　41=00101001）

A　11110110

B　01110110

C　11001000

D　其他

提交

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 $A - B$。（97= 01100001　41=00101001）

解：　$A = -97 = -1100001$　　$[A - B]_{补} = [A]_{补} + [-B]_{补}$

$[A]_{补} = 1, 0011111$

$+[-B]_{补} = 1, 1010111$

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

$\therefore A - B = +118$ （溢出）

主观题 10分

设置

练习 1 设 $x=\frac{9}{16}$ $y=\frac{11}{16}$，用补码求 $x+y$

作答

正常使用主观题需2.0以上版本雨课堂

练习 1　设 $x = \frac{9}{16}$　$y = \frac{11}{16}$，用补码求 $x+y$

解：　$x = 9/16 = 0.1001$

$y = 11/16 = 0.1011$

$[x]_{补} = 0.1001$　　$[y]_{补} = 0.1011$

$+ \quad [y]_{补} = 0.1011$

$[x]_{补} + [y]_{补} = 1,0100 = [x + y]_{补}$

$x + y = -0.1100 = -\frac{12}{16}$　　错

一位符号位判溢出

- 一位符号位判溢出：A+B
  - **加法$[A+B]_{补}$**：若$[A]_{补}$和$[B]_{补}$两个数符号相同，而其结果的符号与$[A]_{补}$和$[B]_{补}$的**符号不同，即为溢出**
  - **减法$[A-B]_{补}=[A+(-B)]_{补}$**：若$[A]_{补}$和$[-B]_{补}$两个数符号相同，而其结果的符号与$[A]_{补}$和$[-B]_{补}$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

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

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

两位符号位判溢出

- 小数变形补码

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

- 整数变形补码

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

最高符号位 代表其 真正的符号

$$[x]_{补'}+[y]_{补'}=[\,x+y\,]_{补'} \quad (\text{mod } 4 \text{ 或 mod } 2^{n+2})$$

$$[x-y]_{补'}=[x]_{补'}+[-y]_{补'} \quad (\text{mod } 4 \text{ 或mod } 2^{n+2})$$

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. ×××××

• 例. 设 $A = +\frac{11}{16} = 0.1011$，$B = +\frac{3}{16} = 0.0011$，求 $[A+B]_{补}$

解：

$$
\begin{array}{rl}
[A]_{补} = & 00.1011 \\
+\,[B]_{补} = & 00.0011 \\
\hline
[A]_{补} + [B]_{补} = & 00.1110 = [A+B]_{补}
\end{array}
$$

结果的双符号位相同，未溢出

• 例. 设 $A = -\frac{11}{16} = -0.1011$，$B = -\frac{3}{16} = -0.0011$，求 $[A+B]_{补}$

解：

$$
\begin{array}{rl}
[A]_{补} = & 11.0101 \\
+\,[B]_{补} = & 11.1101 \\
\hline
[A]_{补} + [B]_{补} = & \cancel{1}11.0010 = [A+B]_{补}
\end{array}
$$

丢掉　　结果的双符号位相同，未溢出

- 例. 设 $A = +\frac{11}{16} = 0.1011$，$B = +\frac{7}{16} = 0.0111$，求 $[A+B]_{补'}$

解：

$$
\begin{array}{rl}
 & [A]_{补'} = \quad 00.1011 \\
+ & [B]_{补'} = \quad 00.0111 \\
\hline
 & [A]_{补'} + [B]_{补'} = \quad 01.0010 = [A+B]_{补'}
\end{array}
$$

第1位符号位　　结果的双符号位不同，溢出

- 例. 设 $A = -\frac{11}{16} = -0.1011$，$B = -\frac{7}{16} = -0.0111$，求 $[A+B]_{补'}$

解：

$$
\begin{array}{rl}
 & [A]_{补'} = \quad 11.0101 \\
+ & [B]_{补'} = \quad 11.1001 \\
\hline
 & [A]_{补'} + [B]_{补'} = \quad 110.1110 = [A+B]_{补'}
\end{array}
$$

丢掉　　结果的双符号位不同，溢出

计算机的运算方法

- 定点运算
  - 加减法运算
  - 一位乘法运算
  - Booth算法
  - 除法运算
- 浮点运算

乘法运算

- 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

```
  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

笔算乘法的改进

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

$$= 0.1A + 0.00A + \underline{0.001A + 0.0001A}$$

$$= 0.1A + 0.00A + \underline{0.001}(1 \cdot A + 0.1A)$$

$$= 0.1A + 0.01[0 \cdot A + 0.1(1 \cdot A + 0.1A)]$$

$$= 0.1\{1 \cdot A + 0.1[0 \cdot A + 0.1(1 \cdot A + \underline{0.1A})]\}$$

右移一位

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

$\underbrace{\qquad\qquad}_{③}$

1 被乘数A + 0

2 右移 一 位，得新的部分积

3 部分积 + 被乘数

…

右移 一 位，得结果

$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{⑧}$

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

• 改进后的笔算乘法过程（竖式）

$A \cdot 0.1011$

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

$A = -0.1101$

$B = 0.1011$

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

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

数值部分

小结

- 乘法运算可用 **加和移位** 实现
  - $n = 4$，加 4 次，移 4 次

$$A \cdot 0.1011$$
$$= 2^{-1}\{1 \cdot A + 2^{-1}[0 \cdot A + 2^{-1}(1 \cdot A + 2^{-1}(1 \cdot A + 0))]\}$$

- 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。

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

- **被乘数只与部分积的高位相加**

硬件：　3个寄存器，其中两个具有移位功能

1个全加器：被乘数和部分积的高位相加操作（n+1位）

计算机的运算方法

- 定点运算
  - 加减法运算
  - 一位乘法
  - Booth算法
  - 除法运算
- 浮点运算

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 $x^* \cdot y^*$

小数原码一位乘递推公式

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

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

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

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

z_0

z_1

$\cdots$

z_n

$$z_0 = 0$$

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

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

$$\vdots$$

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

整数原码一位乘递推公式

$$X^*\times Y^*= X^*(Y_{n-1}\ \dots\ Y_1Y_0)=2^{n-1}Y_{n-1}X^*+2^{n-2}Y_{n-2}X^*+\dots+2^1Y_1X^*+2^0Y_0X^*$$

$$=2^n(2^{-1}Y_{n-1}X^*+2^{-2}Y_{n-2}X^*+\dots+2^{-n+1}Y_1X^*+2^{-n}Y_0X^*)$$

$$=2^n(2^{-1}(Y_{n-1}X^*+2^{-1}Y_{n-2}X^*+\dots+2^{-n+2}Y_1X^*+2^{-n+1}Y_0X^*))$$

$$=2^n(2^{-1}(Y_{n-1}X^*+2^{-1}(Y_{n-2}X^*+\dots+2^{-n+3}Y_1X^*+2^{-n+2}Y_0X^*)))$$

$$=2^n(2^{-1}(Y_{n-1}X^*+2^{-1}(Y_{n-2}X^*+2^{-1}(\dots+2^{-1}(Y_1X^*+2^{-1}(Y_0X^*+0))))\dots)$$

Z_0 Z_1 Z_n

$$Z_0=0$$
$$Z_1=2^{-1}(Y_0X^*+Z_0)$$
$$Z_2=2^{-1}(Y_1X^*+Z_1)$$
$$\dots$$
$$Z_n=2^{-1}(Y_{n-1}X^*+Z_{n-1})$$

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

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

补码一位乘法运算规则（小数）

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

●被乘数任意，乘数为正（类似原码乘） $y_0=0$, $[x \times y]_{补} = [x]_{补} \times y$

但加和移位按补码规则运算，积的符号自然形成

●被乘数任意，乘数为负 $[x \times y]_{补} = [x]_{补} \times 0.y_1y_2 \cdots y_n + [-x]_{补}$

乘数$[y]_{补}$，去掉符号位，操作同上，最后加$[-x]_{补}$，校正

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

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

$[x \times y]_{补} = [x \times 0.y_1y_2 \cdots y_n - x]_{补}$

$= [x]_{补} \times 0.y_1y_2 \cdots y_n + [-x]_{补}$

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

补码一位乘法运算规则（整数）

设 被乘数 $[X]_{补} = X_n \cdots X_1X_0$，乘数 $[Y]_{补} = Y_n \cdots Y_1Y_0$

- 被乘数**任意**，乘数为**正**（$[X \times Y]_{补}=[X]_{补} \times [Y]_{补}$，$Y_n=0$）
  - 类似原码乘，加和移位按**补码**规则，积的符号**自然形成**
- 被乘数**任意**，乘数为**负**
  - 乘数$[Y]_{补}$，**去掉符号位**，其他操作同上，最后**加$2^n[-X]_{补}$**(**校正**)

$$[X \times Y]_{补}=[X]_{补} \times [Y_{n-1} \cdots Y_1Y_0]_{补} + 2^n[-X]_{补}$$

计算机的运算方法

- 定点运算
  - 加减法运算
  - 一位乘法
  - Booth算法
  - 除法运算
- 浮点运算

Booth 算法（被乘数、乘数符号任意）

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

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

$-[x]_{补} = +[-x]_{补}$

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

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

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

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

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

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

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

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

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

附加位 y_{n+1}

Booth 算法递推公式

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

$$[z_0]_{补}=0$$

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

$$\vdots$$

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

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

最后一步不移位

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

解：

	$[y]_{补}$		
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

补码右移

补码右移

补码右移

$[x]_{补} = 0.0011$

$[y]_{补} = 1.0101$

$[-x]_{补} = 1.1101$

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

∴ $[x \times y]_{补}$

$=1.11011111$

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

$[-x]_{补} = 0.1011$

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

$\therefore [x \times y]_{补}$

$= 0.10001111$

乘法小结

- **整数乘法**与**小数乘法**基本相同
  - 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**

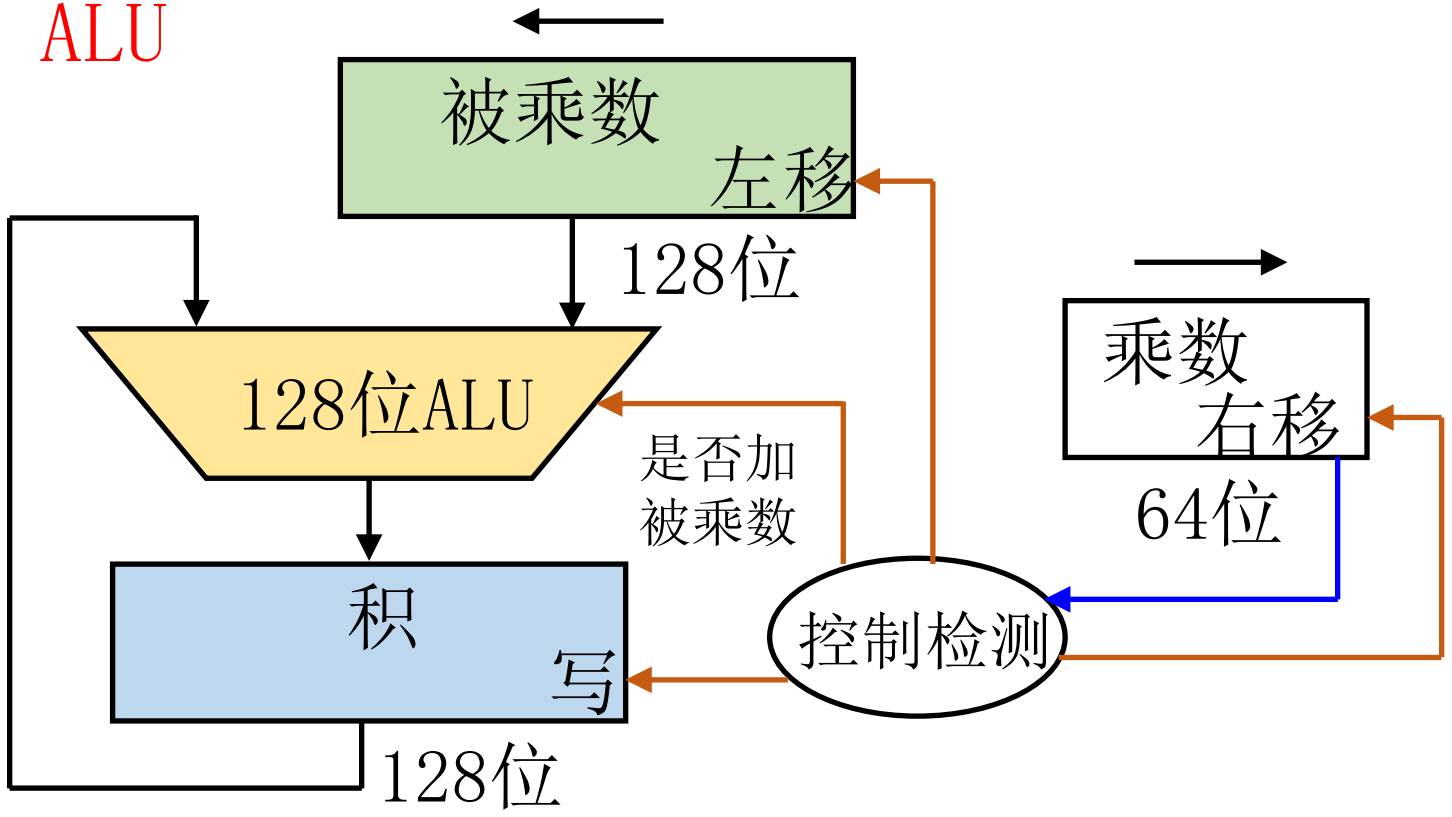
- 原码乘去掉符号位运算, 即为无符号数乘法

- 不同的乘法运算需有不同的硬件支持

乘法器硬件示意图

- 被乘数寄存器128位
  - 被乘数64位，要左移一位64次。
- 浪费：被乘数寄存器、ALU
  - 多数时间只用64位

```
    1 1 0 1
  × 1 0 1 1
  ---------
    1 1 0 1
  1 1 0 1
0 0 0 0
1 1 0 1
---------------
1 0 0 0 1 1 1 1
```



黑书图3-3

改良版乘法器硬件

A. 积寄存器129位(初值为：65个0 和 64位的乘数）

- 最高位用于保存加法器的进位

B. 若乘数最右端为1

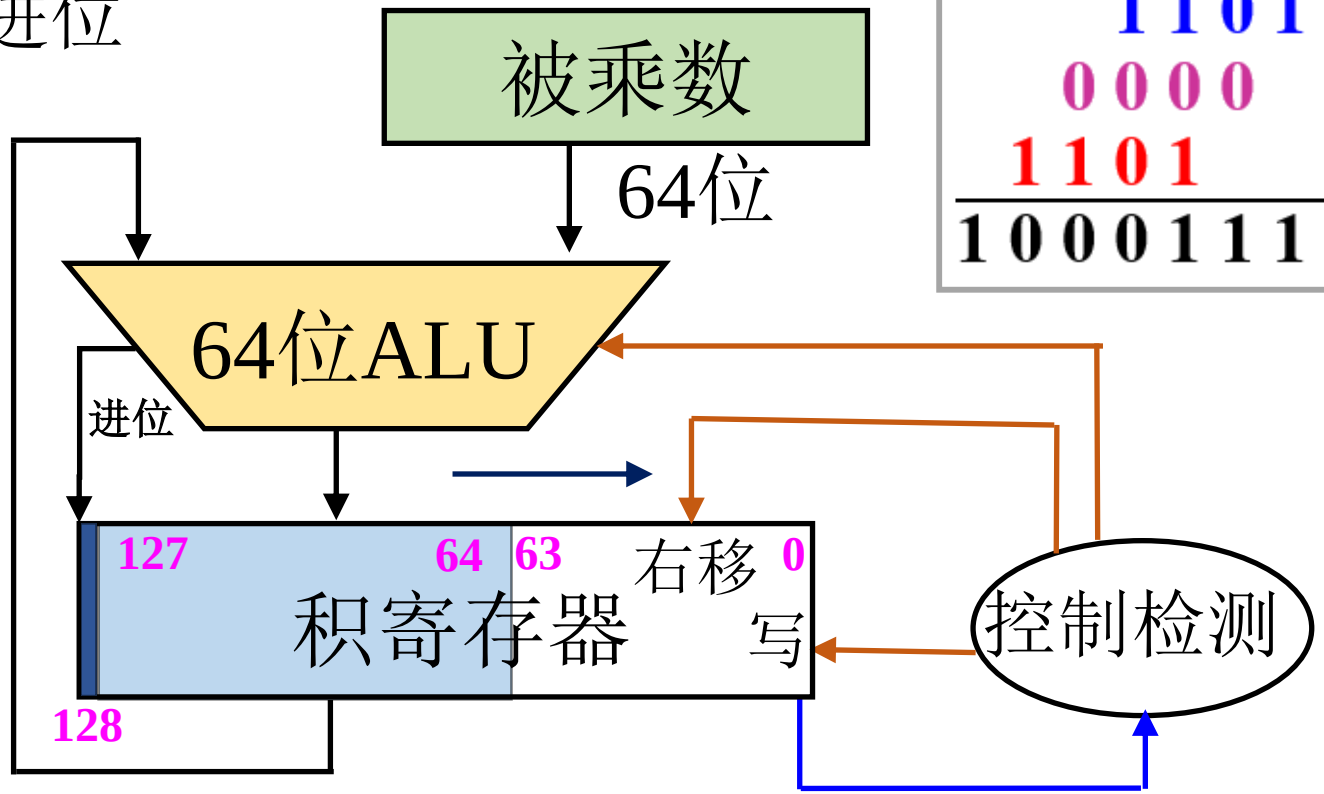
- 取积寄存器[127:64]
- 取出的值加上被乘数；
- 和写入积寄存器[128:64]

C. 积寄存器整体右移一位

*B和C循环64次

*结果为积寄存器[127:0]

*快速乘法： 黑书3.3.3



黑书图3-5

```
  1 1 0 1
×1 0 1 1
-------
  1 1 0 1
1 1 0 1
0 0 0 0
1 1 0 1
---------
1 0 0 0 1 1 1 1
```

计算机的运算方法

- 定点运算
  - 加减法运算
  - 一位乘法
  - Booth算法
  - **除法运算**
- 浮点运算

除法运算

- 分析笔算除法

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

```
                0.1101
          ┌──────────
0.1101 ) 0.10110
         0.01101
         ─────────
         0.010010
         0.001101
         ─────────
         0.0001010 0
         0.0000110 1
         ─────────
         0.0000011 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

笔算除法和机器除法的比较

$x = -0.1011 \quad y = 0.1101$

$$
\begin{array}{r}
0.1101 \\
\hline
0.1101\overline{)\,0.101100} \\
\underline{0.01101} \\
0.010010 \\
\underline{0.001101} \\
0.000101\,00 \\
\underline{0.000011\,01} \\
0.000001\,11
\end{array}
$$

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	$\lvert x\rvert - \lvert y\rvert \geq 0$ 上商 1
	$\lvert x\rvert - \lvert y\rvert < 0$ 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

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

恢复余数法

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

假设机器数是5位，符号位1位

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

① $x_0 \oplus y_0 = 1 \oplus 1 = 0$

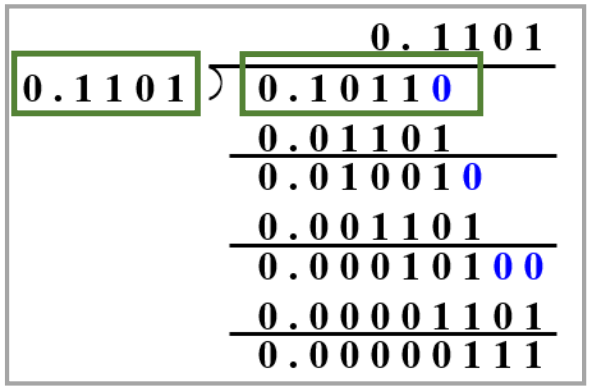
②

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

```
                0.1101
0.1101 ) 0.101110
         0.01101
         0.010010
         0.001101
         0.0001010 0
         0.0000110 1
         0.0000011 1
```

上商前，比较被除数和除数：

$[x^*-y^*]_{补}=[x^*]_{补}+[-y^*]_{补}$



上商前，比较被除数和除数：

$[x^*-y^*]_{补}=[x^*]_{补}+[-y^*]_{补}$

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

不恢复余数法（加减交替法）

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		$+[-y^*]_{补}$

- 恢复余数法运算规则

① 余数　$R_i>0$　上商 “1”，$2R_i-y^*$

② 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

① 余数　$R_i>0$：　上商 “1”，　$2R_i-y^*$

② 余数　$R_i<0$：　上商 “0”，　$2R_i+y^*$

加减交替

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

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

$[x]_{原} = 1.1011$

$[y]_{原} = 1.1101$

$[x^*]_{补} = 0.1011$

$[y^*]_{补} = 0.1101$

$[-y^*]_{补} = 1.0011$

• 不恢复余数法运算规则

① 余数 $R_i>0$ 上商“1” $2R_i-y^*$

② 余数 $R_i<0$ 上商“0” $2R_i+y^*$

- 例. 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

第三章 计算机的运算方法

- 定点运算
- 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

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

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

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

- (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××…×	真值	−0.1××…×
原码	0.1××…×	原码	1.1××…×
补码	0.1××…×	补码	1.0××…×
反码	0.1××…×	反码	1.0××…×

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

特例

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

$$[S]_{原} = 1.100\cdots0$$

$$[S]_{补} = \boxed{1.1}00\cdots0$$

∴ $[-\frac{1}{2}]_{补}$ 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 尾数溢出（ >1）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　　∴ $S_y \longrightarrow 1$，j_y+1

∴ $[y]_{补'} = 00,010;\ 00.010110$

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

$[x+y]_{补} = 00,011;\ 00.100101$

$\therefore\ x+y = 0.100101 \times 2^{11}$

- 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

$$[S_x]_{补'} = 11.\ 101100$$
$$+\ [-S_y]_{补} = 11.\ 001000$$
$$\overline{\qquad\qquad\qquad\qquad}$$
$$10.\ 110100$$

③ 右规

$$[x-y]_{补} = 11,\ 100;\ 10.\ 110100$$

右规后

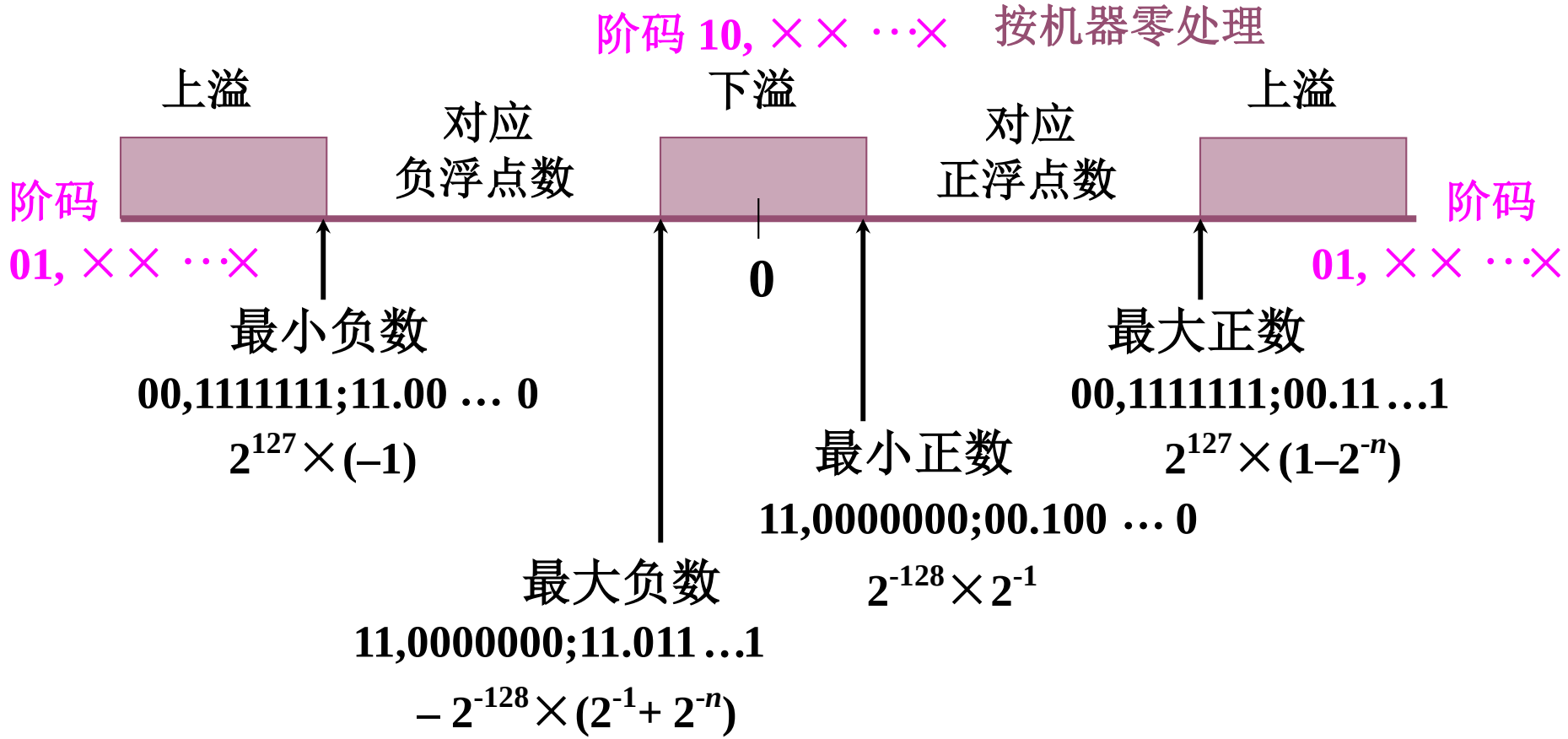
$$[x-y]_{补} = 11,\ 101;\ 11.\ 011010$$

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

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

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



投票 最多可选6项

设置

关于计算机的运算方法，哪部分有疑问?

A 定点加减运算

B 原码一位乘法

C Booth乘法

D 定点除法

E 浮点运算

F 其他可发弹幕

提交