3.5.3 浮点数的乘除运算

刘 芳 副教授

国防科学技术大学计算机学院

浮点数乘除

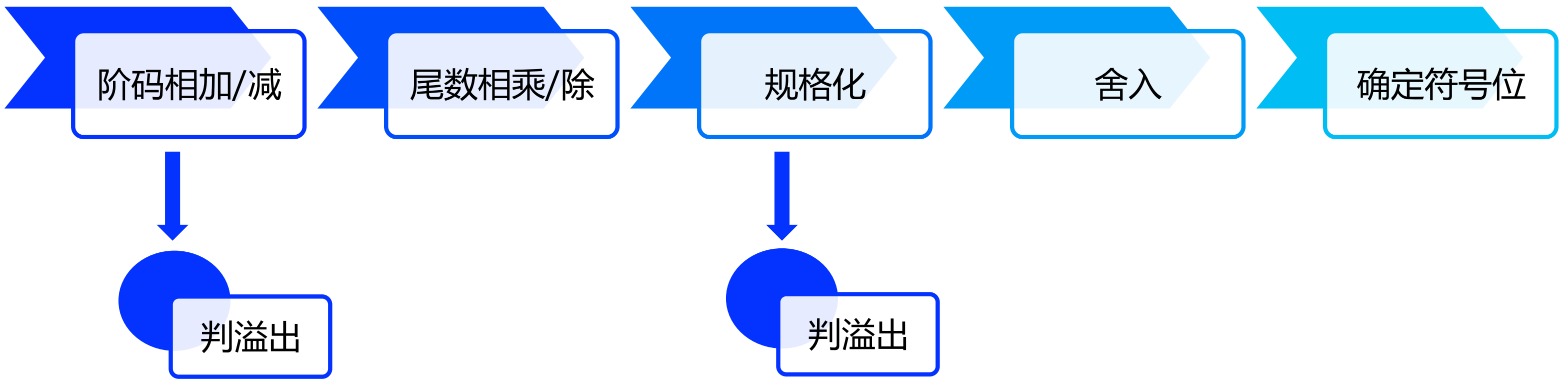
两个浮点数：

$$A=M_a \cdot 2^{Ea}，\quad B=M_b \cdot 2^{Eb}$$

$$A \times B = (M_a \times M_b) \cdot 2^{Ea+Eb}$$

$$A \div B = (M_a \div M_b) \cdot 2^{Ea-Eb}$$

计算机实现步骤：



1. 阶码运算

阶码相加（单精度浮点乘法）

$$[E_1 + E_2]_{移} = f\left([E_1]_{移}, [E_2]_{移}\right) = \boxed{([E_1]_{移} + [E_2]_{移} + 129) \bmod 2^8}$$

$$\begin{aligned}
[E_1 + E_2]_{移} &= 127 + E_1 + E_2 \\
&= 127 + E_1 + 127 + E_2 - 127 \\
&= [E_1]_{移} + [E_2]_{移} - 127 \\
&= [E_1]_{移} + [E_2]_{移} + [-127]_{补} \\
&= ([E_1]_{移} + [E_2]_{移} + 10000001B) \bmod 2^8
\end{aligned}$$

阶码相加（单精度浮点乘法）

例：若两个IEEE754操作数的阶码分别为10和－5，求10＋（－5)的移码

解：

$[E_1]_{移} = 127+10 = 137 = 1000\ 1001B$

$[E_2]_{移} = 127+(-5) = 122 = 01111010B$

$[E_1+E_2]_{移} = [E_1]_{移} + [E_2]_{移} + 129$

$= 1000\ 1001 + 0111\ 1010$

$+ 1000\ 0001 \pmod{2^8}$

$= 1000\ 0100B = 132$

其阶码的和为$132-127=5$，正好等于$10+(-5)=5$

阶码相加（单精度浮点乘法）

$$[E_1 - E_2]_{移} = f\left([E_1]_{移}, [E_2]_{移}\right) = \boxed{([E_1]_{移} + [-[E_2]_{移}]_{补} + 127) \bmod 2^8}$$

$$[E_1 - E_2]_{移} = E_1 - E_2 + 127$$
$$= 127 + E_1 - (127 + E_2) + 127$$
$$= [E_1]_{移} - [E_2]_{移} + 127$$
$$= ([E_1]_{移} + [-[E_2]_{移}]_{补} + 01111111B) \bmod 2^8$$

阶码相加（单精度浮点乘法）

例：若两个IEEE754操作数的阶码分别为10和 - 5，求10- (- 5)的移码

解：

$[E_1]_{移} = 127+10 = 137 = 1000\ 1001B$

$[E_2]_{移} = 127+(-5) = 122 = 01111010B$

$[E_1+E_2]_{移} = [E_1]_{移} + [-[E_2]_{移}]_{补} + 127$

$= 1000\ 1001 + 1000\ 0110$

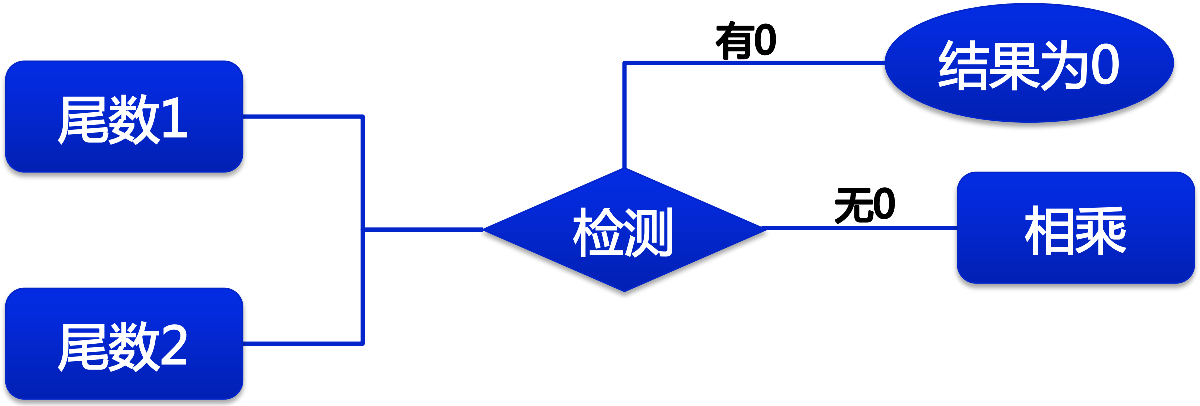
$+ 0111\ 1111 (\mathrm{mod}\ 2^8)$

$= 1000\ 1110B = 142$

其阶码的差为142–127 = 15，正好等于10 - (–5) = 15

尾数相乘

预处理

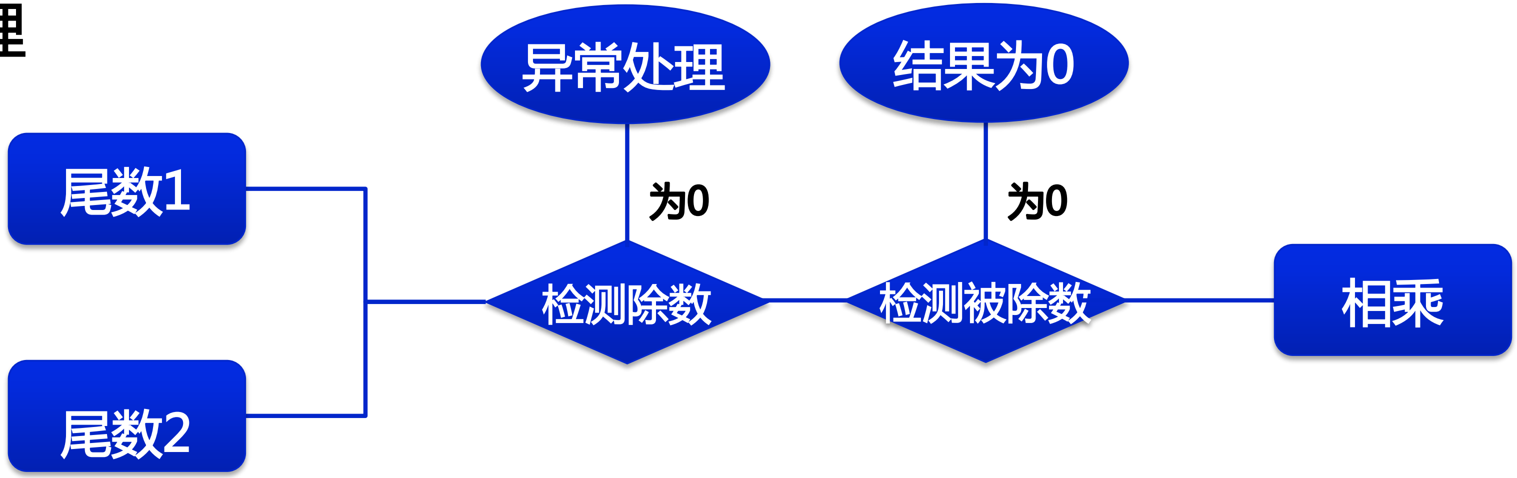


相乘

- 定点小数原码乘法运算

尾数相乘

预处理



相乘

- 定点小数原码乘法运算

规格化原则

- 当尾数高位为0，左规
- 当尾数产生进位，右规

乘法运算结果最多左规几次、右规几次？
除法呢？

规格化尾数：$1 \le$ 尾数$_1 < 2$， $1 \le$ 尾数$_2 < 2$ (尾数形为1.xxx)

乘法： $1 \le$ 尾数$_1$ $\times$ 尾数$_2 < 4$ 不需左规、最多右规一次

除法： $0.5 <$ 尾数$_1$ $\div$ 尾数$_2 < 2$ 不需右规、最多左规一次

4. 舍入并确定符号

舍入

- 就近舍入
- 朝+∞舍入
- 朝—∞舍入
- 朝0舍入

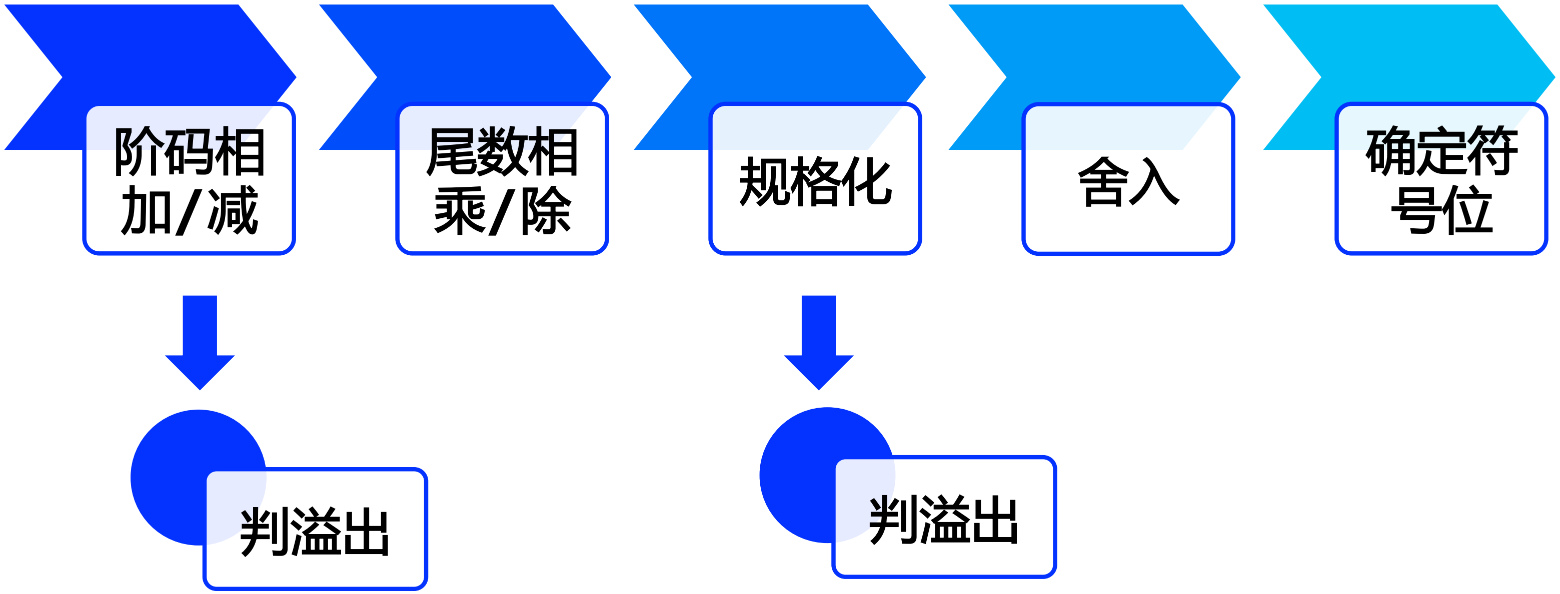
若尾数是0，则需要将阶码也置0

符号确定

- 同号：结果为正
- 异号：结果为负

5. 判溢出

阶码溢出情况



5. 判溢出

阶码溢出情况

阶码求和	E_x	+ E_y	+ 129	= E_b	
阶码求差	1xxxx	1xxxx		0xxxx	上溢
	0xxxx	0xxxx		1xxxx	下溢
规格化				11111	上溢
				00000	下溢

5. 判溢出

阶码溢出情况

- 阶码求和
- 阶码求差
- 规格化

$$E_x - E_y + 127 = E_b$$

E_x	E_y	E_b	
1xxxx	0xxxx	0xxxx	上溢
0xxxx	1xxxx	1xxxx	下溢
		11111	上溢
		00000	下溢

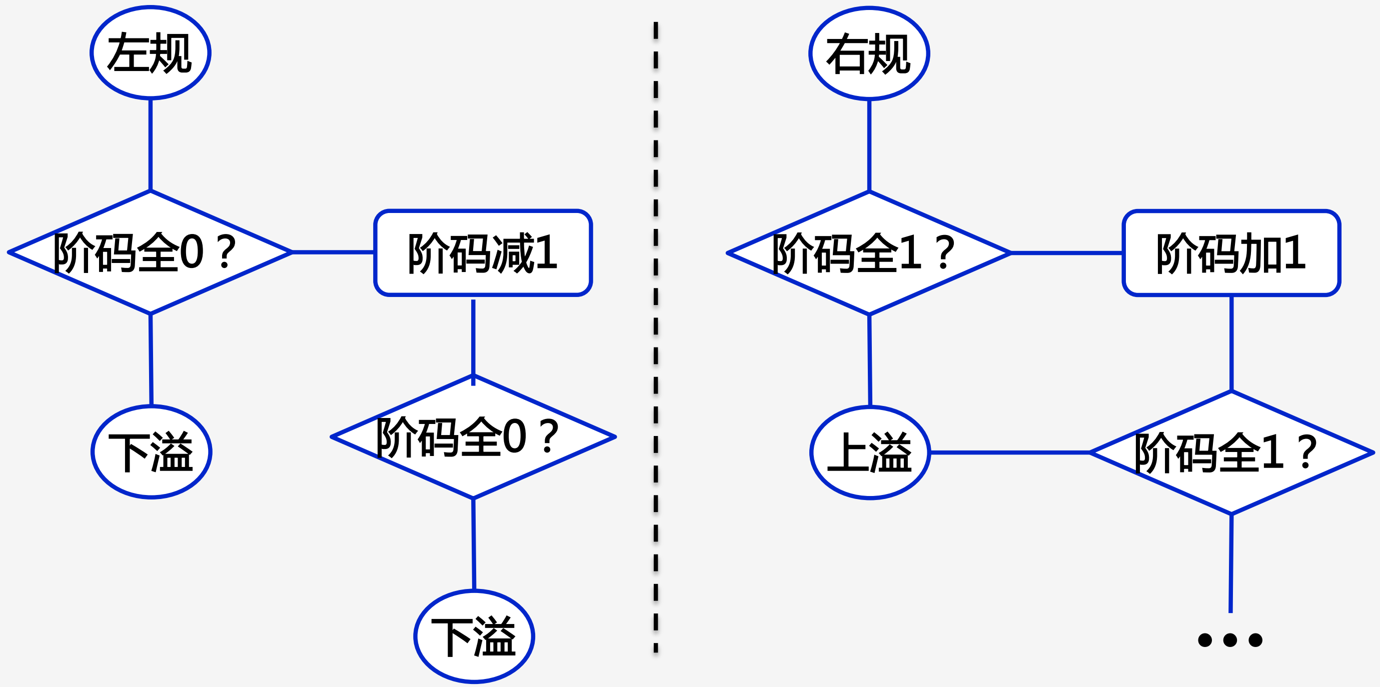
5. 判溢出

阶码溢出情况

阶码求和

阶码求差

规格化



溢出判断举例

例1：若Eb=0000 0001，则尾数左规一次后，结果的阶码Eb=?

解：Eb = Eb+$[-1]_{补}$ = 0000 0001 + 1111 1111

= 0000 0000　阶码下溢！

例2：若Ex=1111 1110， Ey=1000 0000，则乘法运算时，结果的阶码Eb=?

解：Eb = Ex+Ey+129 = 1111 1110 + 1000 0000 + 1000 0001

= 1111 1111　阶码上溢！

浮点数乘法实例

例：用二进制的形式求出浮点数0.5_{10}与-0.4375_{10}之积(假设保留4位有效数位的精度)。

解： $0.5_{10} = 1/2_{10} = 0.1_2 = 1.000_2 \times 2^{-1}$

$-0.4375_{10} = -7/16_{10} = -0.0111_2 = -1.110_2 \times 2^{-2}$

- ■阶码相加： $-1 + (-2) = -3$
- ■尾数相乘：
  $1.000_2 \times 1.110_2 = 1.110000_2$
  乘积为：$1.110000_2 \times 2^{-3}$

$$
\begin{array}{r}
1.000_{\text{two}} \\
\times \quad 1.110_{\text{two}} \\
\hline
0000 \\
1000 \\
1000 \\
1000 \\
\hline
1110000_{\text{two}}
\end{array}
$$

浮点数乘法实例

例：用二进制的形式求出浮点数0.5_{10}与-0.4375_{10}之积(假设保留4位有效数位的精度)。

解： $0.5_{10} = 1/2_{10}=0.1_2=1.000_2\times2^{-1}$

$-0.4375_{10} = -7/16_{10} = -0.0111_2= -1.110_2\times2^{-2}$

■阶码相加： $-1+(-2)=-3$

■尾数相乘：

$1.000_2\times1.110_2=1.110000_2$

乘积为：$1.110000_2\times2^{-3}$

■规格化并判溢出：

尾数1.110000已是规格化数,

且127≥-3≥-126，没有溢出！

■舍入：$1.110_2\times2^{-3}$

■确定符号位：

乘积为负数，$-1.110_2\times2^{-3}$

所以浮点乘法结果：

$$-1.110_2\times2^{-3} = -0.00111_2 = -7/32_{10} = -0.21875_{10}$$

浮点数除法实例

例：用二进制的形式求出浮点数0.5_{10}与-0.4375_{10}的商(假设保留4位有效数位的精度)。

解： $0.5_{10}=1/2_{10}=0.1_2=1.000_2\times2^{-1}$

$-0.4375_{10}=-7/16_{10}=-0.0111_2=-1.110_2\times2^{-2}$

■阶码相减： $-1-(-2)=1$

■尾数相除

：$1.000_2\div1.110_2=0.1001001_2$

商为：$0.1001001_2\times2^1$

■规格化并判溢出：

规格化：$1.0010010_2\times2^0$

且$127\geq0\geq-126$，没有溢出

■舍入：$1.001_2\times2^0$

■确定符号位：

商为负数，$-1.001_2\times2^0$

所以浮点除法结果：

$$-1.001_2\times2^0=-1.001_2$$
$$=-9/8_{10}$$
$$=-1.125_{10}$$

浮点数运算小结

浮点数运算：由多个ALU + 移位器实现

加/减运算

- 对阶 、尾数相加减、规格化处理、舍入、判断溢出

乘/除运算

- 尾数用定点原码乘/除运算实现，阶码用定点数加/减运算实现

溢出判断

- 当结果发生阶码上溢时，结果发生溢出；发生阶码下溢时，结果为0

精确表示运算结果

- 中间结果增设保护位、舍入位，等
- 最终结果四种舍入方式：就近舍入 / 正向舍入 / 负向舍入 / 截去