计算机组成原理

浮点运算

王浩宇,教授
haoyuwang@hust.edu.cn
https://howiepku.github.io/

本节内容

- 浮点数加减运算
- 浮点数乘除运算*

浮点数加减运算

- 设有两个浮点数$X = M_x \times 2^{E_x}$ 和 $Y = M_y \times 2^{E_y}$，且$E_x > E_y$
- 若要求X±Y的结果S，则
  - $S = X \pm Y = M_S \times 2^{E_S}$
  - 其中，$E_S = E_x$，$M_S = M_x \pm (M_y \ \mathrm{SHR}\ (E_x - E_y))$
- 浮点数加减运算的步骤
  - 零操作数检查　**一个操作数为0，则不必运算，节省运算时间**
  - 两操作数对阶　**使小数点位置对齐，为加减运算做准备**
  - 尾数相加减　**以双符号位的补码形式进行加减法操作**
  - 结果的规格化
  - 结果的舍入处理
  - 结果的溢出判断

浮点数加减运算——两操作数对阶

- **对阶的原则**
  - 以较大的阶码为标准，调整阶码较小的数据；
    - 避免阶码较大的浮点数的尾数左移，导致最高有效数位丢失；

增大阶码，
尾数右移

- **具体操作**
  - 求阶差$\triangle E = E_X - E_Y$
  - 调整阶码较小的数据
    - 若$\triangle E > 0$，则M_Y右移$\triangle E$位，结果的阶码为E_X
    - 若$\triangle E < 0$，则M_X右移$|\triangle E|$位，结果的阶码为E_Y
- **例 X——E_X=0 001，M_X=0.101；Y——E_Y=0 011，尾数M_Y=0.111**
  - 阶差$\triangle E = E_X - E_Y$ = 0 001-0 011 = - 10 < 0
  - X尾数M_X右移2位， M_X=0.001 01

浮点数加减运算——结果的规格化处理

- 两尾数加减的结果有两种情况
  - 尾数溢出 → **两符号位为01或10** → **右规**
    - 尾数右移1位，阶码加1
  - 尾数为非规格化数据 → **补码表示的符号位与最高数值位相同** → **左规**
    - 尾数左移1位，阶码减 1，直至数值位最高位与符号位相反。
- 同上例，对阶后$E=E_Y=0\ 011$，尾数$M_X=0.001\ (01)$，$M_Y=0.111$
  - 尾数求和

    $M_S = M_X + M_Y = 00.001\ 01 + 00.111 = 01.000\ 01$
  - 两符号位相反，应进行右规1位的操作
  - 则$M_S = 00.100\ (001)$， $E_S= 011+1= 0\ 100$

浮点数加减运算——结果的舍入处理

- 在对阶或右规操作时，会使加数或结果的尾数低若干位移出，影响精度，常用两种舍入处理方法
  - 方法１： ０舍１入法
    - 保留右移时的移出位，若最高位为１，则尾数加１；否则舍去
    - 特点：精度较高，但需要记录所有的移出位
  - 方法２：恒置１法
    - 若之前步骤有右移操作，则直接将结果的最低位置1
    - 特点：精度较0舍1入法较低，但应用简单
- 同上例，结果的尾数 $M_S = 00.100\ 001$
  - ０舍１入法: $M_S = 00.100$
  - 恒置１法: $M_S = 00.101$

浮点数加减运算——结果的溢出判断

- **尾数溢出**
  - 在规格化处理时，通过右规完成；
- **阶码溢出**
  - 上溢(结果绝对值太大)——置上溢标志，结束；
  - 下溢(结果绝对值太小)——置机器零；
  - 正常——运算结束；
- **同上例， 运算结果的阶码E_S= 011+1= 0 100**
  - 未溢出！

[例] 设 x = 2^{010}×0.11011011， y = 2^{100}×(-0.10101100)，求 x + y

- 设浮点数的阶码用双符号位，尾数用单符号位的补码表示
  - $[X]_{浮}$＝00 010，0.11011011
  - $[Y]_{浮}$＝00 100，1.01010100

① 求阶差并对阶

- $\triangle E = E_X - E_Y = [E_X]_{补} + [-E_Y]_{补}$
  ＝00 010＋11 100 ＝11 110
- 则浮点数X，应使M_X右移2位，E_X加2，
  $[X]_{浮}$＝00 100，0.00110110(11)，E_S=00 100

△E= -2 <0
则E_Y为结果阶码
修改浮点数X

$$
\begin{array}{r}
0.\,0\,0\,1\,1\,0\,1\,1\,0\,(11) \\
+\quad 1.\,0\,1\,0\,1\,0\,1\,0\,0 \\
\hline
1.\,1\,0\,0\,0\,1\,0\,1\,0\,(11)
\end{array}
$$

② 尾数求和：

- M_S= 1. 1 0 0 0 1 0 1 0 (11)

[例] 设 x ＝2^{010}×0.11011011， y ＝2^{100}×(-0.10101100)，求 x + y

- **和的规格化处理** M_S= 1. 1 0 0 0 1 0 1 0 (11)
  - 结果尾数的符号位与最高数值位相同，应左规1位
  - 则 M_S=1.00010101(10)，E_S=00 100－1＝ 00 011。
- **舍入处理**
  - 若采用0舍1入法

    M_S=1.00010110

$$
\begin{array}{r}
1.00010101 \\
+\qquad\qquad 1 \\
\hline
1.00010110
\end{array}
$$

- **结果溢出判断**
  - 阶码符号位为00，和不溢出
- **最终结果S ＝ X＋Y ＝ 00 011，1. 00010110**

  ＝ 2^{011}×(－0.11101010)

课堂练习

已知： $x = -0.1000101 \times 2^{-111}$

$y = +0.0001010 \times 2^{-100}$

①用补码运算求 $x + y =$？并判断是否溢出？

②用补码运算求 $x - y =$？并判断是否溢出？

假设两数均以补码表示，且采用双符号位, 先将它们变为规格化浮点数, 则它们的浮点表示分别为

$[x]_{浮} = 11\ 001,\quad 11.0111011$

$[y]_{浮} = 11\ 100,\quad 00.0001010 = 11\ 001，\ 00.1010000$

已知$[x]_{浮}$＝11 001,11.0111011，$[y]_{浮}$＝ 11 001,00.1010000，求 x ＋ y

〈1〉 **求阶差并对阶**

E x ＝E y，即△E为0，阶码相等，无需对阶

〈2〉 **尾数求和**

Ms= 00.0 0 0 1 0 1 1

$$
\begin{array}{r}
11.0\,1\,1\,1\,0\,1\,1 \\
+\ \ 00.1\,0\,1\,0\,0\,0\,0 \\
\hline
00.0\,0\,0\,1\,0\,1\,1
\end{array}
$$

〈3〉 **规格化处理**

尾数运算结果的符号位与最高数值位同值，执行左规处理

Ms=00.1011000，Es=10 110

〈4〉 **舍入处理**

没有丢失位，不必进行舍入

〈5〉 **判溢出**

阶码符号位为10，发生下溢，故需置机器0

已知$[x]_{浮}$＝11 001,11.0111011，$[y]_{浮}$＝11 001,00.1010000，求 x － y

<1> **求阶差并对阶**

E x ＝E y，即△E为0，阶码相等，无需对阶

<2> **尾数求差**

$[-M_y]_{补}$＝11.0110000

Ms= 10.1 1 0 1 0 1 1

$$
\begin{array}{r}
11.0\,1\,1\,1\,0\,1\,1 \\
+\ \underline{11.0\,1\,1\,0\,0\,0\,0} \\
10.1\,1\,0\,1\,0\,1\,1
\end{array}
$$

<3> **规格化处理**

两位符号位不同，应执行右规1位的处理，

Ms= 11.0110101（1），Es= 11 010

0舍1入法

<4> **舍入处理**

丢失位高位为1，但结果为负数，故该位真值为0，应舍去

<5> **判溢出**

阶码符号位为11，无溢出，故结果为：

$x - y = -0.1001011 \times 2^{-6}$

浮点乘法和除法运算*

- 设有两个浮点数 x 和 y：

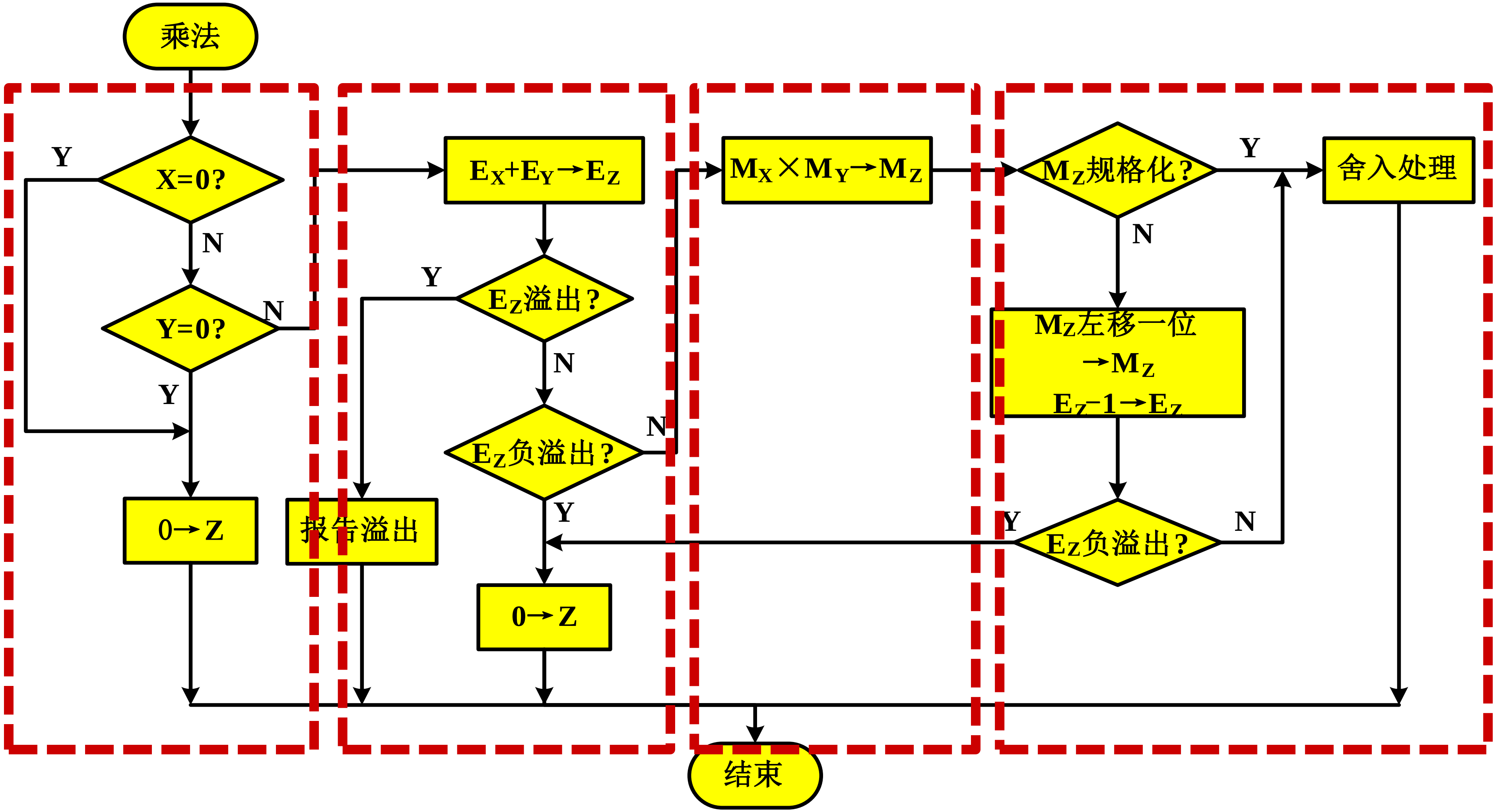
$$x = 2^{E_x} \cdot M_x, \quad y = 2^{E_y} \cdot M_y$$

$$x \times y = 2^{(E_x + E_y)} \cdot (M_x \times M_y)$$

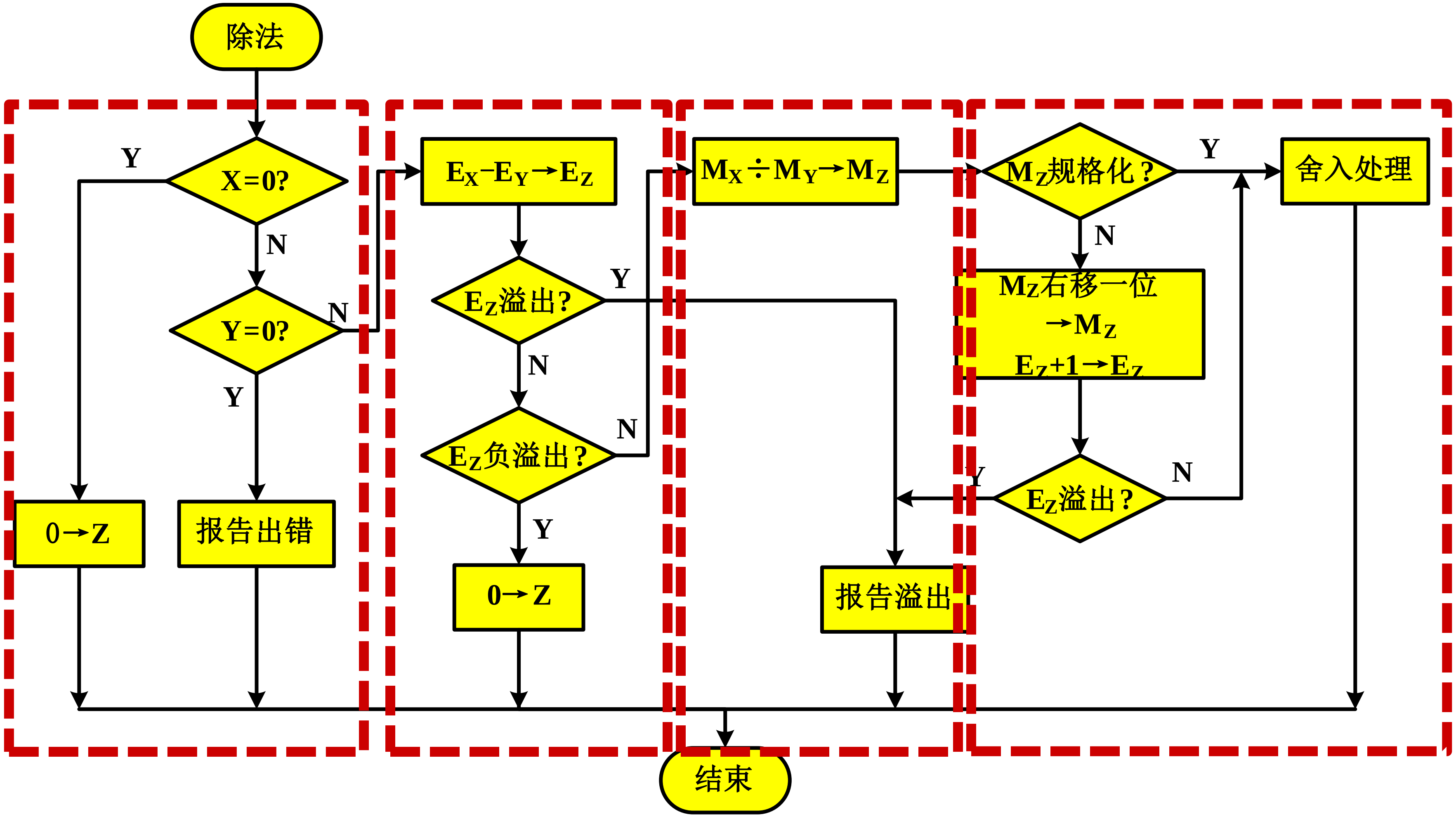
$$x \div y = 2^{(E_x - E_y)} \cdot (M_x \div M_y)$$

- 乘除运算分为四步
  - 0操作数检查
  - 阶码加减操作
  - 尾数乘除操作
  - 结果规格化和舍入处理

浮点乘法和除法运算


乘法
X=0?
Y
N
Y=0?
N
Y
0→Z
$E_X+E_Y \to E_Z$
E_Z溢出?
Y
N
E_Z负溢出?
N
Y
报告溢出
0→Z
$M_X \times M_Y \to M_Z$
M_Z规格化?
Y
N
舍入处理
M_Z左移一位
$\to M_Z$
$E_Z-1 \to E_Z$
E_Z负溢出?
Y
N
结束

浮点乘法和除法运算


除法
X=0?
Y
N
Y=0?
N
Y
0→Z
报告出错
$E_X-E_Y→E_Z$
E_Z溢出?
Y
N
E_Z负溢出?
N
Y
0→Z
$M_X÷M_Y→M_Z$
报告溢出
M_Z规格化?
Y
N
舍入处理
M_Z右移一位
$→M_Z$
$E_Z+1→E_Z$
E_Z溢出?
Y
N
结束

浮点乘法和除法运算

- **浮点数的阶码运算（移码的运算规则）**
  - 将阶码运算变为移码运算

$[X]_{移}=2^n+x \qquad 2^n> x \geqslant -2^n$

$\Rightarrow [X]_{移}+[Y]_{移}= 2^n+X+2^n+Y= 2^n+(2^n+X+Y) =2^n+[X+Y]_{移}$,

可以证明：

$$[X+Y]_{移} = [X]_{移}+[Y]_{移}+ 2^n = [X]_{移}+[Y]_{补}\ (\text{mod } 2^{n+1}) = [Y]_{移}+[X]_{补}\ (\text{mod } 2^{n+1})$$

$$[X-Y]_{移} = [X]_{移}+[-Y]_{补} = [-Y]_{移}+[X]_{补}\ (\text{mod } 2^{n+1})$$

浮点乘法和除法运算

- **溢出问题**
  - 乘法下溢的情况
    - 求阶码和时已经下溢（负阶码）
    - 乘积左规时阶码减1而造成下溢
  - 乘法上溢的情况
    - 求阶码和时已经上溢（正阶码）
    - 乘积右规时阶码加1而造成上溢

浮点乘法和除法运算

- **移码采用双符号位，为了对溢出进行判断**
  - 规定移码的第二个符号位，即最高符号位恒用 0 参加运算，则溢出条件是结果的最高符号位为1
  - 01 为正 00 为负
  - 10 上溢 11 下溢

浮点乘法和除法运算

例： $x=+011, y=+110$,求$[x+y]_{移}$ 和 $[x-y]_{移}$,并判断是否溢出。

$[x]_{移}=01\ 011$, $[y]_{补}=00\ 110$, $[-y]_{补}=11\ 010$

$[x+y]_{移}=[x]_{移}+[y]_{补}=10\ 001$, 结果上溢

$[x-y]_{移}=[x]_{移}+[-y]_{补}=00\ 101$, 结果正确,为$-3$

浮点乘法和除法运算

- **浮点数的溢出**
  - 阶码上溢——超过阶码可能表示的最大值的正指数值，一般将其认为是＋∞和－∞
  - 阶码下溢——超过阶码可能表示的最小值的负指数值，一般将其认为是0
  - 尾数上溢——两个同符号尾数相加产生最高位向上的进位，将尾数右移，阶码增1来重新对齐
  - 尾数下溢——在将尾数右移时，尾数的最低有效位从尾数域右端流出，要进行舍入处理

浮点乘法和除法运算

例： 设有浮点数 $x = 2^{-5} \times 0.0110011$, $y = 2^{3} \times (-0.1110010)$, 阶码用4位移码表示，尾数（含符号位）用8位补码表示。求$[x \times y]_{浮}$。要求用补码完成尾数乘法运算，运算结果尾数保留高8位（含符号位），并用尾数低位字长值处理舍入操作。

浮点乘法和除法运算

解：移码采用双符号位，尾数补码采用单符号位，则有

$[M_x]_补=0.0110011$，$[M_y]_补=1.0001110$，

$[E_y]_移=01\ 011$，

$[E_y]_补=00\ 011$

$[E_x]_移=00\ 011$，

$[x]_浮=00\ 011, 0.0110011$；

$[y]_浮=01\ 011, 1.0001110$

(1)判断操作是否为"0"，求阶码和

$[E_x+E_y]_移=[E_x]_移+[E_y]_补$

$=00\ \ 011+00\ \ 011=00\ \ 110$，值为移码形式－2。

(2) 尾数乘法运算可采用补码阵列乘法器实现，即有

$[M_x]_补\times[M_y]_补$

$=[0.0110011]_补\times[1.0001110]_补$

$=[1.1010010,1001010]_补$

浮点乘法和除法运算

（3） 规格化处理

乘积的尾数符号位与最高数值位符号相同，不是规格化的数，需要左规，阶码变为00 101（-3），

尾数变为 1.0100101，0010100。

（4） 舍入处理

尾数为负数，取尾数高位字长，按舍入规则，舍去低位字长，故尾数为 1.0100101

最终相乘结果为 $[x \times y]_{浮}=00\ 101,1.0100101$

其真值为 $x \times y = 2^{-3} \times (-0.1011011)$

浮点乘法和除法运算

- **尾数处理：有两种方法（截尾法、舍入法）**
  - 截断
  - 舍入(尾数用原码表示时)
    - 只要尾数最低为1或者移出位中有1数值位，使最低位置1
    - 0舍1入
  - 舍入(尾数用补码表示时)
    - 丢失的位全为0，不必舍入
    - 丢失的最高位为0，以后各位不全为0时；或者最高为1，以后各位全为0时，**则舍去丢失位上的值**
    - 丢失的最高位为1，以后各位不全为0时，则在尾数的最低位入1的修正操作

舍入处理的补充与解释

- **舍入处理**
  - 对于原码，采用0舍1入法时，不论其值是正数或负数，“舍”使得数的绝对值变小，“入”使得数的绝对值变大
  - 对于补码，采用0舍1入法时，若丢失的位不全是0，对正数来说，“舍”、“入”的结果与原码正好相同；对负数来说，“舍”、“入”的结果与原码分析正好相反，即“舍”使绝对值变大，“入”使绝对值变小。

$X_{[补]}$ = 1.01111000 真值：-0.10001000

补码直接0舍1入：

$X_{[补]}$ = 1.10000000 真值：-0.10000000 绝对值变小

先转化为原码再0舍1入：

1.10001000-> 1.10010000 真值：-0.10010000 绝对值增大

舍入处理：解释和补充

- **为了使原码、补码舍入处理后的结果相同，对负数的补码可采用如下规则进行舍入处理：**
  - 当丢失的各位均为0时，不必舍入
  - 当丢失的各位数中的最高位为0时，且以下各位不全为0；或者丢失的各位数中的最高位为1，且以下各位均为0时，则舍去被丢失的各位
  - 当丢失的各位数中的最高位为1，且以下各位又不全为0时，则在保留尾数的最末位加1修正

- 如何理解？

舍入处理：解释和补充

原码舍入后?

$X_{[补]}$ 舍入前	原码	$X_{[补]}$舍入后	对应的真值
1.01110000	-0.10010000	1.0111（不舍不入）	-0.1001
1.01111000	-0.10001000	1.0111（舍）	-0.1001
1.01110101	-0.10001011	1.0111（舍）	-0.1001
1.01111100	-0.10000100	1.1000 （入）	-0.1000

$X_{[补]}$ 舍入后的值 与其转换为原码之后舍入后的值 所对应的真值应完全相同！

浮点乘法和除法运算

- 设$[x_1]_补=11.01100000$,$[x_2]_补=11.01100001$,$[x_3]_补=11.01101000$, $[x_4]_补=11.01111001$,求执行只保留小数点后4位有效数字的舍入操作值

$[x_1]_补=11.0110$ (不舍不入)

$[x_2]_补=11.0110$ (舍)

$[x_3]_补=11.0110$ (舍)

$[x_4]_补=11.1000$ (入)

浮点乘法和除法运算

关于乘法溢出判定问题：

- 求阶码和后判下溢可能出现的问题

例：已知：$A=-1\times2^{-128}$，$B=-1\times2^{-1}$，求$A\times B=?$

$[A]_{补}$=11 0000000，11.00…0

$[B]_{补}$=11 1111111，11.00…0

- 两数阶码之和为10 1111111(-129)，尾数之积为01.00…0(+1)
- 结果需右规，即阶码加1，最后阶码为

  11 0000000 (-128)，尾数为00.10…0。无溢出
- 若在求阶码之和后判溢出，就会误认为下溢(阶码两符号位为10表示下溢)，错误地扩大溢出范围。因此，应该在规格化时判下溢

浮点乘法和除法运算

- 在规格化后判下溢可能出现的问题

例：已知：$A=0.5\times2^{-128}$，$B=0.5\times2^{-128}$，求A×B=？

$[A]_{补}$=11 0000000，00.10···0

$[B]_{补}$=11 0000000，00.10···0

- 两数阶码之和为10 0000000 （-256)，尾数之积为00.01...0(注意此时阶码下溢)
- 左规后，积的尾数变成00.1...0，而修改后的阶码变成：10 0000000+11 1111111(-1)=01 1111111 （上溢）
- 若在规格化过程中，即在求积的阶码时判溢出，就可能把本来的阶下溢错判为阶上溢

浮点乘法和除法运算

- **解决办法：**
  - 在规格化后判下溢可能出现的问题

    解决办法：在规格化时判下溢，加入一些条件，即阶码寄存器的最高位（R_{ES} ）参与控制
    - 设求阶码和后，阶码寄存器的内容为：

      $R_{ES}\ R_{E0}\ .\ \ R_{E1}R_{E2}R_{E3}R_{E4}R_{E5}R_{E6}R_{E7}$
    - 左规时，阶码加法器的输出为

      $\Sigma_{ES}\Sigma_{E0}\ .\ \ \Sigma_{E1}\Sigma_{E2}\Sigma_{E3}\Sigma_{E4}\Sigma_{E5}\Sigma_{E6}\Sigma_{E7}$
    - 当R_{ES}=1且$\Sigma_{ES}\neq\Sigma_{E0}$时下溢，即下溢条件为：　　　$R_{ES}\cdot(\Sigma_{ES}\oplus\Sigma_{E0})$
    - 这样在规格化时判下溢，不会扩大溢出范围，也不会错判成上溢
    - 阶码下溢处理　　乘积为0

浮点乘法和除法运算

- 规格化后判上溢

例：已知：$A=0.5\times2^{+127}$，$B=0.5\times2^{+1}$，求$A\times B=?$

$[A]_{补}$=00 1111111，00.10…0

$[B]_{补}$=00 0000001，00.10…0

- 两数阶码之和为01 0000000（+128），尾数之积为00.01...0。
- 求阶码之和后，积的阶码为01 0000000，此时判断为上溢
- 结果左规后，乘积的阶码减1后又被修正为01 0000000＋11 1111111＝00 1111111，刚好不上溢

浮点乘法和除法运算

- 规格化后判上溢

例：已知：$A=-1\times 2^{+127}$，$B=-1\times\ 2^{+127}$，求$A\times B=?$

$[A]_{补}$=00 1111111，11.00…0

$[B]_{补}$=00 1111111，11.00…0

  - 两数阶码之和为01 1111110(+254)，尾数之积为01.00…0(+1)
  - 结果需右规，即阶码加1，最后阶码为

    01 1111111 (+255)，尾数为00.10…0。
  - 若在求阶码之和后判溢出，就会判为上溢。在规格化时判上溢，仍保持上溢，不会错判成下溢

- 在规格化之后判上溢，并不会出现误判，将上溢判成下溢