定点数与浮点数

- 定点数的表示方法
- 浮点数的表示方法
- 定点数与浮点数的对比

定点数与浮点数

定点数的表示方法

- 小数点固定在某个位置的数称之为定点数

纯小数

纯整数

定点数与浮点数

定点数的表示方法

符号位	数值位

小数点

纯小数

符号位	数值位

小数点

纯整数

乘以比例因子以满足定点数保存格式

定点数与浮点数

定点数的表示方法

数值	符号位	数值位
0.1011	0	1011
-0.1011	1	1011

数值	符号位	数值位
1011	0	1011
-1011	1	1011

10.01　　101.1

乘以比例因子以满足定点数保存格式

定点数与浮点数

- 定点数的表示方法
- 浮点数的表示方法
- 定点数与浮点数的对比

定点数与浮点数

浮点数的表示方法

- 计算机处理的很大程度上不是纯小数或纯整数
- 数据范围很大，定点数难以表达

定点数与浮点数

浮点数的表示方法

- 浮点数的表示格式
- 浮点数的表示范围
- 浮点数的规格化

定点数与浮点数

浮点数的表示方法

- 浮点数的表示格式

定点数与浮点数

浮点数的表示格式

$$123450000000 = 1.2345 \times 10^{11}$$

1.2345：尾数

10：基数

11：阶码

科学计数法

定点数与浮点数

浮点数的表示格式

$$N = S \times r^j$$

S：尾数　　r：基数　　j：阶码

阶码符号位	阶码数值位	尾数符号位	尾数数值位

尾数规定使用纯小数

定点数与浮点数

浮点数的表示格式

$$N = S \times r^j$$

$$11.0101 = 0.110101 \times 2^{10}$$

$$11.0101 = 0.0110101 \times 2^{11}$$

阶码符号位	阶码数值位	尾数符号位	尾数数值位（8位）
0	10	0	11010100
0	11	0	01101010

定点数与浮点数

浮点数的表示方法

- 浮点数的表示格式
- 浮点数的表示范围

定点数与浮点数

浮点数的表示范围

假设阶码数值取m位，尾数数值取n位

$$N = S \times r^j$$

阶码符号位	阶码数值位	尾数符号位	尾数数值位

阶码能够表示的最大值： $\mathbf{2^m - 1}$　　$-(\mathbf{2^m - 1})$　　$\mathbf{2^m - 1}$

阶码表示范围： $[-(\mathbf{2^m - 1}), \mathbf{2^m - 1}]$

定点数与浮点数

浮点数的表示范围

假设阶码数值取m位，尾数数值取n位

$$N = S \times r^{j}$$

阶码符号位	阶码数值位	尾数符号位	尾数数值位

尾数能够表示的最大值： $1-2^{-n}$

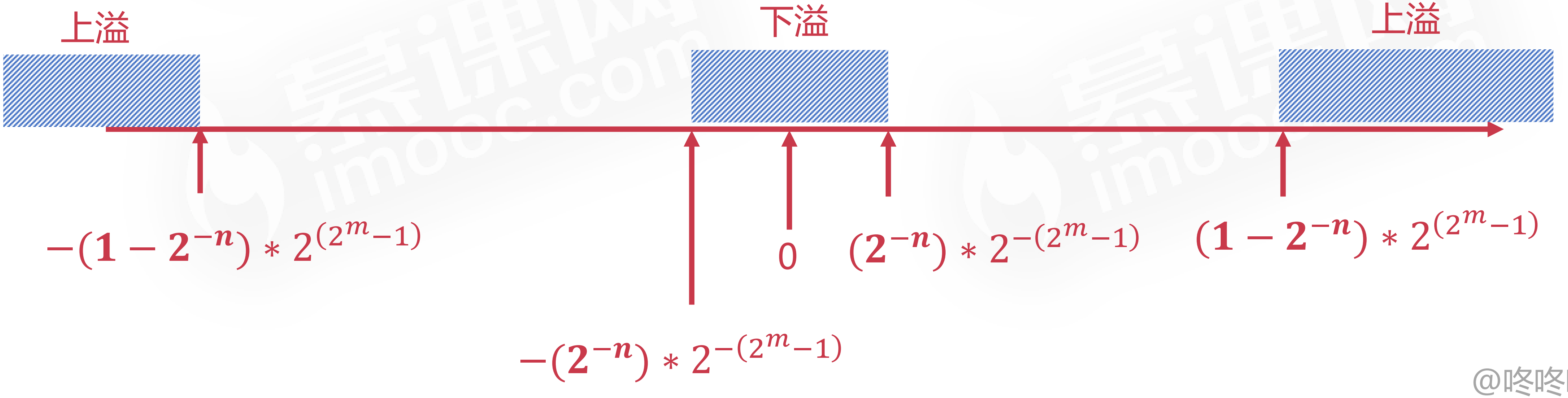
尾数能够表示的最小值： 2^{-n}

尾数表示范围： $[2^{-n}, 1-2^{-n}]$　　$[-(1-2^{-n}), -(2^{-n})]$　　$[2^{-n}, 1-2^{-n}]$

定点数与浮点数

阶码表示范围： $[-(2^m-1), 2^m-1]$

尾数表示范围： $[-(1-2^{-n}), -(2^{-n})]$　　$[2^{-n}, 1-2^{-n}]$



定点数与浮点数

浮点数的表示范围

单精度浮点数：　使用4字节、32位来表达浮点数(float)

双精度浮点数：　使用8字节、64位来表达浮点数(double)

定点数与浮点数

浮点数的表示方法

- 浮点数的表示格式
- 浮点数的表示范围
- 浮点数的规格化

定点数与浮点数

浮点数的规格化

$$123450000000 = 1.2345 \times 10^{11}$$

尾数：[1, 10)

$$123450000000 = 0.12345 \times 10^{12}$$

$$123450000000 = 12.345 \times 10^{10}$$

科学计数法

定点数与浮点数

浮点数的规格化

尾数规定使用纯小数

尾数最高位必须是1

定点数与浮点数

浮点数的规格化

$$11.0101 = 0.110101 \times 2^{10}$$

$$11.0101 = 0.0110101 \times 2^{11}$$

$$11.0101 = 0.00110101 \times 2^{100}$$

$$11.0101 = 1.10101 \times 2^{1}$$

定点数与浮点数

例子1：设浮点数字长为16位，阶码为5位，尾数为11位，将十进制数$\frac{13}{128}$表示为二进制浮点数。

原码=反码=补码：　$x = 0.0001101000$

浮点数规格化：　$x = \boxed{0.1101000} * \boxed{2^{-11}}$

尾数为1101000000　　尾数符为0　　阶符为1　　阶码为0011

阶码符号位	阶码数值位	尾数符号位	尾数数值位
1	0011	0	1101000000

定点数与浮点数

例子2：设浮点数字长为16位，阶码为5位，尾数为11位，将十进制数−54表示为二进制浮点数。

原码： $x = 1,110110$

浮点数规格化： $x = \boxed{-0.110110} * \boxed{2^{110}}$

尾数为1101100000　　尾数符为1　　阶符为0　　阶码为0110

尾数反码0010011111

尾数补码0010100000

阶码符号位	阶码数值位	尾数符号位	尾数数值位
0	0110	1	0010100000

定点数与浮点数

- 定点数的表示方法
- 浮点数的表示方法
- 定点数与浮点数的对比

定点数与浮点数

定点数与浮点数的对比

- 当定点数与浮点数位数相同时，浮点数表示的范围更大
- 当浮点数尾数为规格化数时，浮点数的精度更高
- 浮点数运算包含阶码和尾数，浮点数的运算更为复杂

定点数与浮点数

定点数与浮点数的对比

浮点数在数的表示范围、精度、溢出处理、编程等方面均优于定点数

浮点数在数的运算规则、运算速度、硬件成本方面不如定点数

定点数与浮点数

- 定点数的表示方法
- 浮点数的表示方法
- 定点数与浮点数的对比

定点数的加减法运算

整数加法：　$\mathrm{A}[补] + \mathrm{B}[补] = [A + B][补](mod 2^{n+1})$

小数加法：　$\mathrm{A}[补] + \mathrm{B}[补] = [A + B][补](mod 2)$

数值位与符号位一同运算，并将符号位产生的进位自然丢掉

定点数的加减法运算

例子1：A=-110010， B=001101，求A+B

$$A[补] = 1,001110$$

$$B[补] = B[原] = 0,001101$$

$$\begin{array}{r} 1,001110 \\ +\quad 0,001101 \\ \hline 1,011011 \end{array}$$

$$A[补] + B[补] = (A + B)[补] = 1,011011$$

$$A + B = -100101$$

数值位与符号位一同运算，并将符号位产生的进位自然丢掉

定点数的加减法运算

例子2：A=-0.1010010， B=0.0110100，求A+B

$A[补] = 1,1.0101110$

$B[补] = B[原] = 0,0.0110100$

$$
\begin{array}{r}
1,1.0101110 \\
+\quad 0,0.0110100 \\
\hline
1,1.1100010
\end{array}
$$

$A[补] + B[补] = (A + B)[补]$ =1,1.1100010

$A + B$ =-0.0011110

数值位与符号位一同运算，并将符号位产生的进位自然丢掉

定点数的加减法运算

例子3：A=-10010000， B=-01010000，求A+B

$\mathrm{A}[补] = 1,01110000$

$\mathrm{B}[补] = 1,10110000$

$$
\begin{array}{rr}
 & 1{,}01110000 \\
+ & 1{,}10110000 \\
\hline
\boxed{1} & 1{,}00100000
\end{array}
$$

模2^{n+1}舍去

$\mathrm{A}[补] + \mathrm{B}[补] = (\mathrm{A} + \mathrm{B})[补]$ =1,00100000

$\mathrm{A} + \mathrm{B}$ =-11100000

$\mathrm{A} = -144 \quad \mathrm{B} = -80 \quad \mathrm{A} + \mathrm{B} = -224$

数值位与符号位一同运算，并将符号位产生的进位自然丢掉

定点数的加减法运算

例子4：A=-10010000， B=-11010000，求A+B

$$A[补] = 1,01110000$$

$$B[补] = 1,00110000$$

$$\begin{array}{r} 1,01110000 \\ +\quad 1,00110000 \\ \hline \boxed{1}\ 0,10100000 \end{array}$$

模2^{n+1}舍去

$$A[补] + B[补] = (A + B)[补] = 0,10100000$$

$$A + B = 10100000$$

数值位与符号位一同运算，并将符号位产生的进位自然丢掉

定点数的加减法运算

例子4：A=-10010000， B=-11010000，求A+B

$$A[补] = 1,01110000$$

$$B[补] = 1,00110000$$

$$\begin{array}{rr} & 1,01110000 \\ + & 1,00110000 \\ \hline & \boxed{1}\ 0,10100000 \end{array}$$

模2^{n+1}舍去

$$A[补] + B[补] = (A+B)[补] = 0,10100000$$

$$A + B = 10100000$$

$$A = -144$$

$$B = -208$$

$$A + B = 160$$

定点数的加减法运算

判断溢出

- 双符号位判断法

单符号位表示变成双符号位：0=>00,1=>11

双符号位产生的进位丢弃

结果的双符号位不同则表示溢出

定点数的加减法运算

例子4：A=-10010000， B=-11010000，求A+B

$$A[补] = 1,01110000$$

$$B[补] = 1,00110000$$

$$\begin{array}{r} 11,01110000 \\ +\quad 11,00110000 \\ \hline \boxed{1}\ 10,10100000 \end{array}$$

$\boxed{1}$ → 符号位进位舍去

$$A[补] + B[补] = (A+B)[补] = \boxed{10},10100000$$

$\boxed{10}$ → 双符号位不同，表示溢出

定点数的加减法运算

例子3：A=-10010000， B=-01010000，求A+B

$A[补] = 1,01110000$

$B[补] = 1,10110000$

$$
\begin{array}{rr}
 & 11,01110000 \\
+ & 11,10110000 \\
\hline
\boxed{1} & 11,00100000
\end{array}
$$

符号位进位舍去

$A[补] + B[补] = (A + B)[补] = \boxed{11},00100000$

双符号位相同，没有溢出

$(A + B)[原] = 11,11100000 = -11100000$

定点数的加减法运算

整数减法： $A[补]-B[补]=A+(-B)[补](mod2^{n+1})$

小数减法： $A[补]-B[补]=A+(-B)[补](mod2)$

-B[补]等于B[补]连同符号位按位取反，末位加一

$B[补]=1{,}0010101$　　　　$(-B)[补]=0{,}1101011$

定点数的加减法运算

例子5：A=11001000， B=-00110100，求A-B

$$A[补] = A[原] = 0{,}11001000$$

$$B[补] = 1{,}11001100$$

$$(-B)[补] = 0{,}00110100$$

$$\begin{array}{lr} & 00{,}11001000 \\ + & 00{,}00110100 \\ \hline & 00{,}11111100 \end{array}$$

$$A[补] - B[补] = A + (-B)[补]$$

$$A + (-B)[补] = 0{,}11111100$$

$$A - B = 111111100$$

定点数的加减法运算

整数加法： $A[补] + B[补] = [A + B][补](mod2^{n+1})$

小数加法： $A[补] + B[补] = [A + B][补](mod2)$

数值位与符号位一同运算，并将符号位产生的进位自然丢掉

单符号位表示变成双符号位：0=>00,1=>11

双符号位产生的进位丢弃

结果的双符号位不同则表示溢出

溢出判断

定点数的加减法运算

整数减法：　$A[补] - B[补] = A + (-B)[补](mod2^{n+1})$

小数减法：　$A[补] - B[补] = A + (-B)[补](mod2)$

-B[补]等于B[补]连同符号位按位取反，末位加一

浮点数的加减法运算

$$x = S_x \times r^{j_x}$$

$$y = S_y \times r^{j_y}$$

对阶 → 尾数求和 → 尾数规格化 → 舍入 → 溢出判断

$$x = 0.1101 \times 2^{01}$$

$$y = (-0.1010) \times 2^{11}$$

浮点数的加减法运算

对阶

$$x = 0.1101 \times 2^{01}$$

$$y = (-0.1010) \times 2^{11}$$

- 浮点数尾数运算简单
- 浮点数位数实际小数位与阶码有关
- 阶码按小阶看齐大阶的原则

对阶的目的是使得两个浮点数阶码一致，使得尾数可以进行运算

浮点数的加减法运算

对阶

阶码符号位	阶码数值位	尾数符号位	尾数数值位
00	0001	00	1101
00	0011	11	1010

$$x = 0.1101 \times 2^{01}$$
$$y = (-0.1010) \times 2^{11}$$

$$x = 0.001101 \times 2^{11}$$
$$y = (-0.1010) \times 2^{11}$$

阶码符号位	阶码数值位	尾数符号位	尾数数值位
00	0011	00	0011(01)
00	0011	11	1010

浮点数的加减法运算

尾数求和

- 使用补码进行运算
- 减法运算转化为加法运算：A - B = A + (-B)

浮点数的加减法运算

尾数求和

阶码符号位	阶码数值位	尾数符号位	尾数数值位
00	0011	00	0011
00	0011	11	1010

$$x = 0.0011 \times 2^{11}$$

$$y = (-0.1010) \times 2^{11}$$

$$x[原] = 00.0011 \qquad x[补] = 00.0011$$

$$y[原] = 11.1010 \qquad y[补] = 11.0110$$

$$S = (x + y)[补] = 11.1001$$

阶码符号位	阶码数值位	尾数符号位	尾数数值位
00	0011	11	1001

浮点数的加减法运算

尾数规格化

◆ 对补码进行规格化需要判断两种情况：S>0和S<0

$$S[补] = 00.1xxxxxx (S > 0)$$

符号位与最高位不一致

$$S[补] = 11.0xxxxxx (S < 0)$$

如果不满足此格式，需要进行左移，同时阶码相应变化，以满足规格化

浮点数的加减法运算

尾数规格化

阶码符号位	阶码数值位	尾数符号位	尾数数值位
00	0011	11	1001

$$S = (x + y)[补] = 11.1001$$

$$S = (x + y)[补] = 11.(1)0010(左移)$$

阶码符号位	阶码数值位	尾数符号位	尾数数值位
00	0010	11	0010

$$S[补] = 00.1xxxxxx(S > 0)$$

$$S[补] = 11.0xxxxxx(S < 0)$$

$$S = (x + y)[补] = 11.0010$$

$$(x + y)[原] = -0.1110$$

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

浮点数的加减法运算

尾数规格化（右移）

$$S[补] = 00.1xxxxxx(S > 0)$$

$$S[补] = 11.0xxxxxx(S < 0)$$

- 一般情况下都是左移
- 双符号位不一致下需要右移(定点运算的溢出情况)
- 右移的话则需要进行舍入操作

浮点数的加减法运算

舍入

- “0舍1入”法（二进制的四舍五入）

记得阶码要
+1哦

S[补] = 10.10110111

S[补] = 11.01011011(1)

$$
\begin{array}{r}
11.01011011 \\
+ \qquad\qquad\quad 1 \\
\hline
11.01011100
\end{array}
$$

可能溢出

浮点数的加减法运算

舍入

S[补] = 01.11111111

S[补] = 00.10000000(1)

$$
\begin{array}{r}
00.11111111 \\
+ \qquad\qquad 1 \\
\hline
01.00000000 \\
00.1000000(0)
\end{array}
$$

两次右规，记得阶码要+2哦

浮点数的加减法运算

溢出判断

- 定点运算双符号位不一致为溢出
- 浮点运算尾数双符号位不一致不算溢出

因为尾数双符号位可以进行右规

浮点数的加减法运算

溢出判断

- 浮点运算主要通过阶码的双符号位判断是否溢出

如果规格化后，阶码双符号位不一致，则认为是溢出

浮点数的加减法运算

例子：$x = 0.11010011 \times 2^{1101}$，$y = 0.11101110 \times 2^{1100}$，假设阶码4位，尾数8位，计算x + y

第一步：对阶

阶码符号位	阶码数值位	尾数符号位	尾数数值位
00	1101	00	11010011
00	1100	00	11101110

以大阶为准

尾数右移1位

阶码符号位	阶码数值位	尾数符号位	尾数数值位
00	1101	00	11010011
00	1101	00	01110111(0)

浮点数的加减法运算

例子：$x = 0.11010011 \times 2^{1101}$，$y = 0.11101110 \times 2^{1100}$，假设阶码4位，尾数8位，计算x + y

第二步：尾数求和

$$
\begin{array}{r}
00.11010011 \\
+ \quad 00.01110111 \\
\hline
01.01001010
\end{array}
$$

阶码符号位	阶码数值位	尾数符号位	尾数数值位
00	1101	01	01001010

浮点数的加减法运算

例子：$x = 0.11010011 \times 2^{1101}$，$y = 0.11101110 \times 2^{1100}$，假设阶码4位，尾数8位，计算x + y

第三步：规格化（右规）

$01.01001010 \Rightarrow 00.10100101(0)$

第四步：舍入

$00.10100101(0) \xRightarrow{\text{0舍1入}} 00.10100101$

阶码符号位	阶码数值位	尾数符号位	尾数数值位
00	1110	00	10100101

浮点数的加减法运算

例子：$x = 0.11010011 \times 2^{1101}$，$y = 0.11101110 \times 2^{1100}$，假设阶码4位，尾数8位，计算x + y

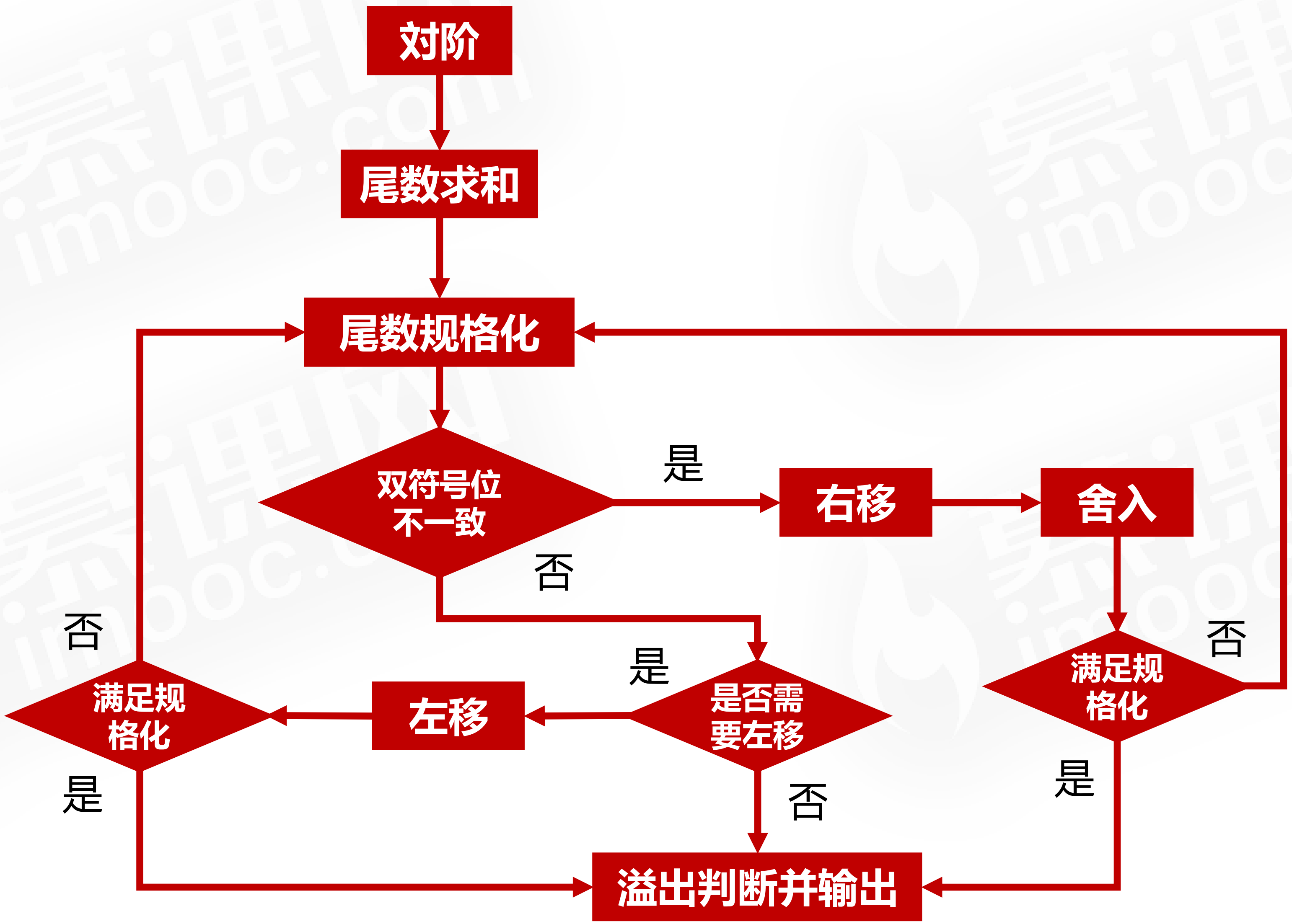
第五步：溢出判断

阶码符号位	阶码数值位	尾数符号位	尾数数值位
00	1110	00	10100101

阶码符号位一致，没有溢出

$$x + y[原] = x + y[补] = 0.10100101 \times 2^{1110}$$

浮点数的加减法运算



对阶
尾数求和
尾数规格化
双符号位不一致
是
右移
舍入
否
是否需要左移
是
否
左移
满足规格化
否
是
满足规格化
否
是
溢出判断并输出

浮点数的乘除法运算

$$x = S_x \times r^{j_x}$$

$$y = S_y \times r^{j_y}$$

乘法：阶码相加，尾数求积

$$x \times y = (S_x \times S_y) \times r^{(j_x + j_y)}$$

浮点数的乘除法运算

$$x = S_x \times r^{j_x}$$

$$y = S_y \times r^{j_y}$$

除法：阶码相减，尾数求商

$$x/y = (S_x/S_y) \times r^{(j_x - j_y)}$$

浮点数的乘除法运算

阶码运算 → 尾数运算 → 尾数规格化 → 舍入 → 溢出判断

浮点数的乘除法运算

例子：$x = 0.11010011 \times 2^{1101}$，$y = 0.11101110 \times 2^{0001}$，假设阶码4位，尾数8位，计算x * y

$$x \times y = (S_x \times S_y) \times r^{(j_x + j_y)}$$

$$= (0.11010011 \times 0.11101110) \times r^{1101+0001}$$

$$= 0.11000100(\text{保留八位}) \times r^{1110}$$