

第2章 数据的表示和运算



主要内容:

(一) 数制与编码

1. 进位计数制及其相互转换
2. 真值和机器数
3. BCD 码
4. 字符与字符串
5. 校验码

(二) 定点数的表示和运算

1. 定点数的表示：无符号数的表示；有符号数的表示。
2. 定点数的运算：定点数的位移运算；原码定点数的加/减运算；补码定点数的加/减运算；定点数的乘/除运算；溢出概念和判别方法。

(三) 浮点数的表示和运算

1. 浮点数的表示：浮点数的表示范围；IEEE754 标准
2. 浮点数的加/减运算

(四) 算术逻辑单元 ALU

1. 串行加法器和并行加法器
2. 算术逻辑单元 ALU 的功能和机构

海纳百川 自强不息 厚德笃学 知行合一



2.3 浮点数的表示和运算



2.3.1 浮点数的表示

(1) 浮点数的表示范围

- 浮点数是指小数点位置可浮动的数据，通常以下式表示：

$$N=M \cdot R^E$$

- 其中，N为浮点数，M为尾数，E为阶码，R称为“阶的基数（底）”，而且R为一常数，一般为2、8或16。在一台计算机中，所有数据的R都是相同的，于是不需要在每个数据中表示出来。

海纳百川 自强不息 厚德笃学 知行合一

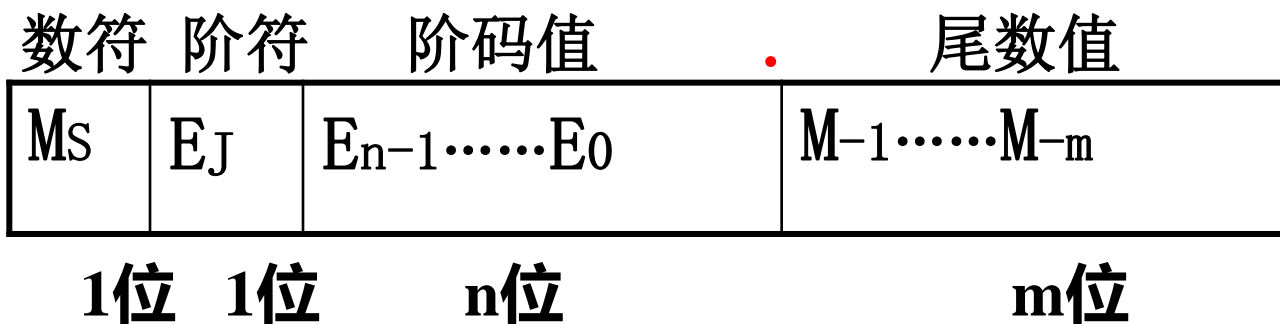


浮点数的机内表示

1949—2019
中华人民共和国成立70周年 大连理工大学 建校70周年

浮点数真值: $N = M \times 2^E$

浮点数的一般机器格式:



- Ms是尾数的符号位，设置在最高位上。
- E为阶码，有n+1位，一般为整数，其中有一位符号位E_J，设置在E的最高位上，用来表正阶或负阶。
- M为尾数，有m位，为一个定点小数。Ms=0，表示正号，Ms=1，表示负。
- 为了保证数据精度，尾数通常用规格化形式表示：当R=2，且尾数值不为0时，**其绝对值大于或等于0.5。对非规格化浮点数，通过将尾数左移或右移，并修改阶码值使之满足规格化要求。**

海纳百川 自强不息 厚德笃学 知行合一



浮点数的机内表示

➤阶码通常为定点整数，补码或移码表示。其位数决定数值范围。阶符表示数的大小。

➤尾数通常为定点小数，原码或补码表示。其位数决定数的精度。数符表示数的正负。

➤浮点数的规格化

字长固定的情况下提高表示精度的措施：

- 增加尾数位数（但数值范围减小）
- 采用浮点规格化形式

尾数规格化： $1/2 \leq |M| < 1$ 最高有效位绝对值为1

海纳百川 自强不息 厚德笃学 知行合一



浮点数规格化方法

►调整阶码使尾数满足下列关系:

- 尾数为原码表示时, 无论正负应满足 $1/2 \leq |M| < 1$

即: 小数点后的第一位数一定要为1。

正数的尾数应为 $0.1x \cdots x$

负数的尾数应为 $1.1x \cdots x$

- 尾数用补码表示时, 小数最高位应与数符符号位相反。

正数应满足 $1/2 \leq M < 1$, 即 $0.1x \cdots x$

负数应满足 $-1/2 > M \geq -1$, 即 $1.0x \cdots x$

浮点数的溢出判断——根据规格化后的阶码判断

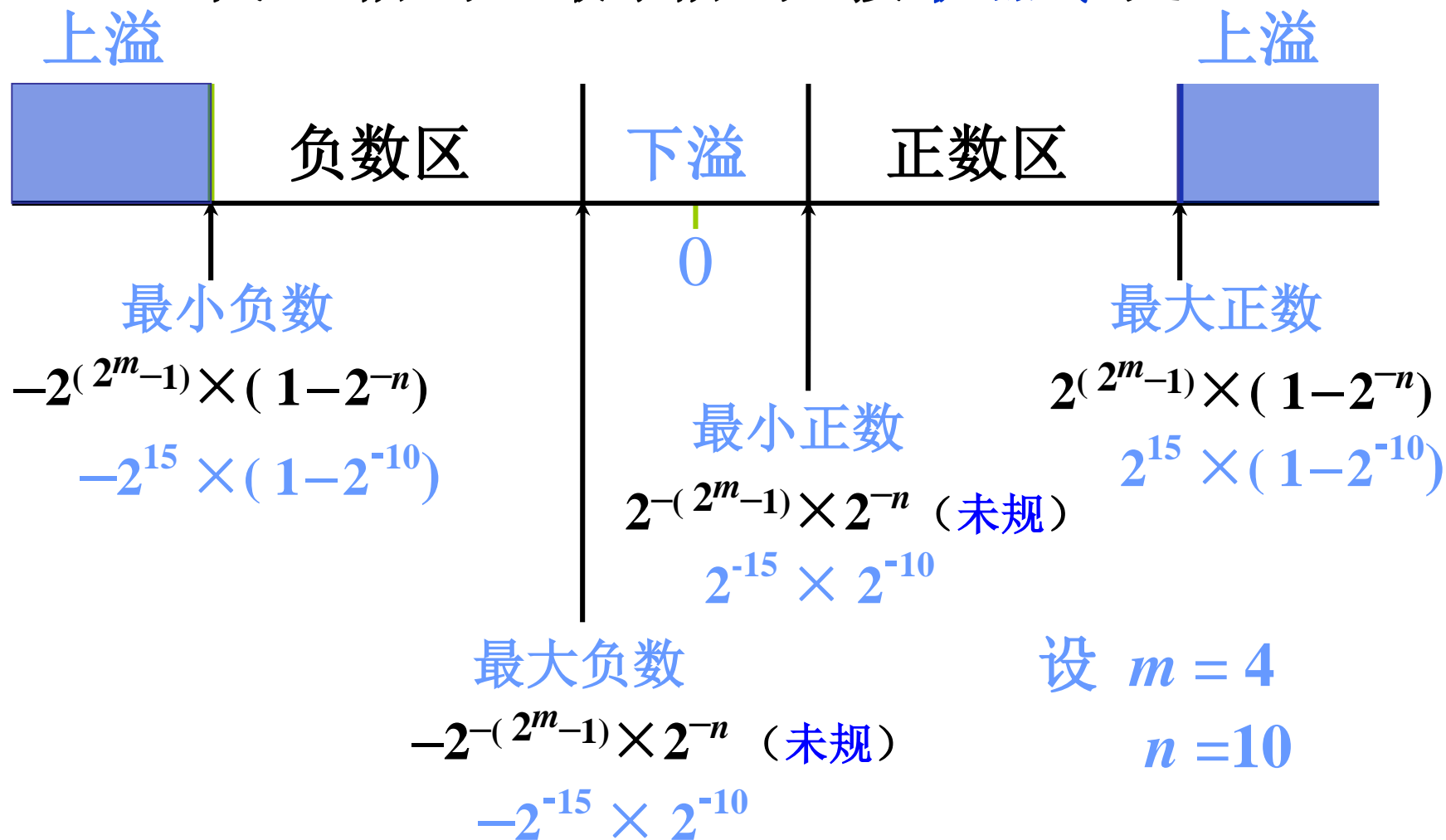
- 上溢——浮点数阶码大于机器最大阶码——中断
- 下溢——浮点数阶码小于机器最小阶码——零处理



浮点数的表示范围

上溢 阶码 > 最大阶码

下溢 阶码 < 最小阶码 按 机器零 处理



例如：设 $m = 4$, $n = 10$

尾数规格化后的浮点数表示范围



最大正数 $2^{+1111} \times \underbrace{0.1111111111}_{10 \text{ 个 } 1} = 2^{15} \times (1 - 2^{-10})$

最小正数 $2^{-1111} \times \underbrace{0.1000000000}_{9 \text{ 个 } 0} = 2^{-15} \times 2^{-1} = 2^{-16}$

最大负数 $2^{-1111} \times (-\underbrace{0.1000000000}_{9 \text{ 个 } 0}) = -2^{-15} \times 2^{-1} = -2^{-16}$

最小负数 $2^{+1111} \times (-\underbrace{0.1111111111}_{10 \text{ 个 } 1}) = -2^{15} \times (1 - 2^{-10})$

海纳百川 自强不息 厚德笃学 知行合一



【例】设某机器用32位表示一个实数，阶码部分8位（含1位阶符），用定点整数补码表示；尾数部分24位（含数符1位），用规格化定点小数补码表示，基数为2。给出 $X=256.5$ 和 $Y=-256.5$ 的浮点表示形式。

解： $X=(256.5)_{10} = +(100000000.1)_2 = +(0.1000000001 \times 2^9)_2$

8位阶码为： $(+9)_{\text{补}} = 0000\ 1001$

24位尾数为： $(+0.10\ 0000\ 0001)_{\text{补}}$

$= 0.1000\ 0000\ 0100\ 0000\ 0000\ 000$

$X=256.5$ 的浮点表示格式为：

0 0000 1001 1000 0000 0100 0000 0000 000

$Y=-(256.5)_{10} = -(100000000.1)_2 = -0.1000000001 \times 2^9$

8位阶码为： $(+9)_{\text{补}} = 0000\ 1001$

24位尾数为： $(-0.10\ 0000\ 0001)_{\text{补}}$

$= 1.0111\ 1111\ 1100\ 0000\ 0000\ 000$

$Y=-256.5$ 的浮点表示格式为：

1 0000 1001 0111 1111 1100 0000 0000 000

海纳百川 自强不息 厚德笃学 知行合一



(2) IEEE754标准

根据IEEE 754国际标准，常用的浮点数有两种格式：

- 单精度浮点数（32位），阶码8位，尾数24位（内含1位符号位）。
 - 双精度浮点数（64位），阶码11位，尾数53位（内含1位符号位）。
- 由于IEEE754标准约定尾数用原码表示，在小数点左部有一位隐含位，从而实际使得尾数的有效值变为1.M。例如，最小为x1.0...0，最大为x1.1...1。规格化表示，故小数点左边的位恒为1，可省去。
 - 阶码部分采用移码表示，移码值127，1~254 经移码为-126 ~ +127。其中00000000 11111111有特点含义，不用做阶码。
- 格式： $(-1)^S \times 2^E \times (M_0.M_{-1} \cdots M_{-(P-1)})$
 - 最高是数符S，占1位，0表示正、1表示负。
 - 指数项E，基数是2，是一个带有一定偏移量的无符号整数（移码）。
 - 尾数部分M，是一个带有一位整数位的二进制小数真值形式（原码）。其规格化形式应调整阶码使其尾数整数位M₀为1且与小数点一起隐含掉。

微机中三种不同类型浮点数的格式



参数	单精度	双精度	扩展精度
浮点数长度(位)	32	64	80
符号位数	1	1	1
尾数长度 P(位)	23+1 (隐)	52+1 (隐)	64
阶码 E 长度(位)	8	11	15
最大阶码	+127	+1023	+16383
最小阶码	-126	-1022	-16382
阶码偏移量	+127	+1023	+16383
表示数范围	$10^{-38} \sim 10^{+38}$	$10^{-308} \sim 10^{+308}$	

单精度浮点数最大表示范围： $\pm 3.40282 \times 10^{38}$

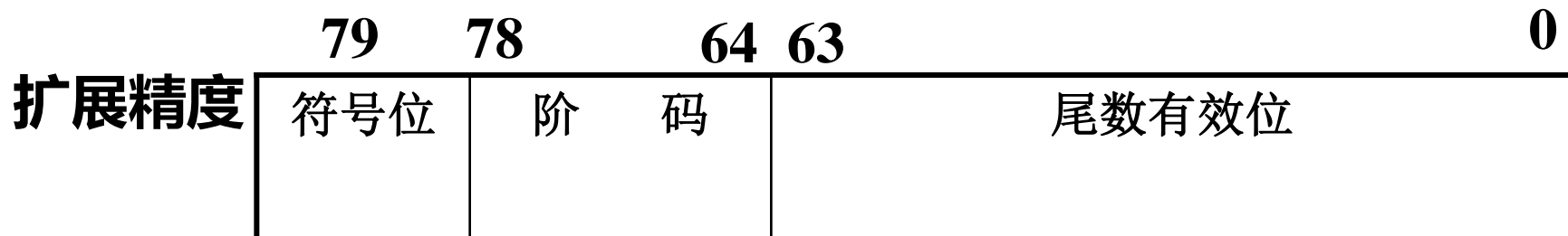
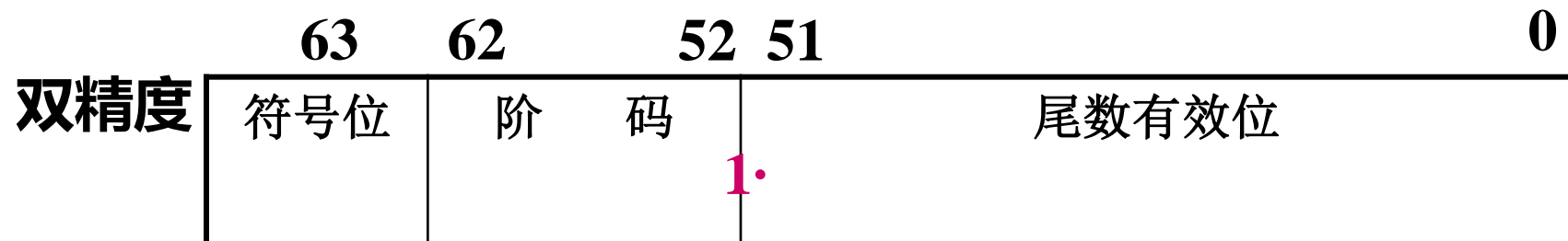
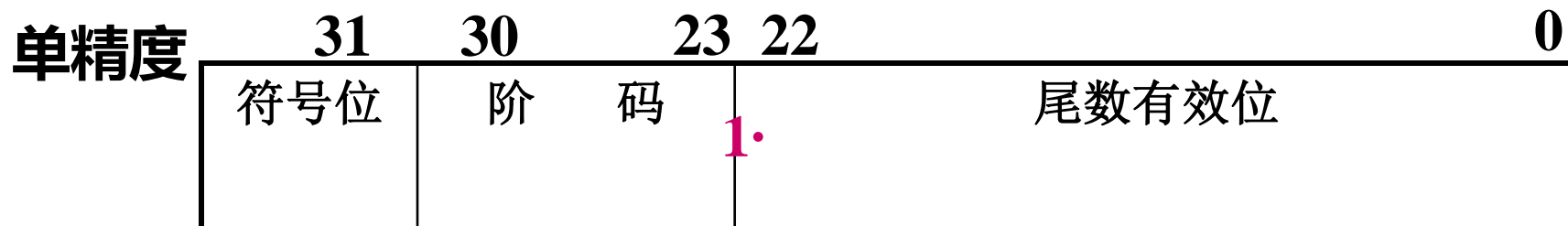
$-(1.1111...1)_2 \times 2^{+127} \sim (1.1111...1)_2 \times 2^{+127}$

接近于0的最小值：单精度浮点数可以表示 1.175×10^{-38}

$((1.00...01)_2 \times 2^{-126})$ 的数据而不损失精度。



微机中浮点数的三种表示形式



【例】将十进制数178.125表示成微机中的单精度浮点数。

解：178.125=10110010.001B

=1.0110010001B × 2⁷

指数E=7+127=0111+01111111=134=1000 0110 B

127是单精度浮点数应加的指数偏移量。

完整的浮点数形式为：

0 1000 0110 011 0010 0010 0000 0000 0000
= 43322000H

隐含

【例】将Pentium机中的单精度浮点数3F580000H表示成十进制数，其真值是多少？

解：3F580000H=

0011 , 1111, 0101, 1000, 0000, 0000, 0000, 0000 B

数符:0 (正数)

阶码: $E=(0111\ 1110)_2-127=126-127=-1$

尾数: $D=(1.1011)_2$ (此处小数点前的1不是符号位。)

$X=(1.1011)_2 \times 2^{-1} = (0.11011)_2 = 0.84375$

解题技巧：首先将十六进制数转换为二进制数，并分离出符号位、阶码和尾数。然后按照IEEE 754 短浮点数的格式可以得出结果。


海纳百川 自强不息 厚德笃学 知行合一



解：表示成规格化二进制真值格式为

$$+76.75 = +1001100.11\text{B} = +1.0011\ 0011\text{B} \times 2^{+110\text{B}}$$

表示成规格化尾数格式为：（将整数位的1隐含）

+1.0011 0011——— 0 0011 0011
 计算出阶码为： **尾数符号**

110+0111 1111=1000 0101 (6+127=133)

于是该数的短浮点数格式为:

0100 0010 1001 1001 1000 0000 0000 0000
= 42998000H



2.3.2 浮点数的加/减运算



➤ 两数首先**均为规格化数**，进行规格化浮点数的加减运算需经过5步完成：

(1) **对阶**
操作：低阶向高阶补齐，使阶码相等。

(2) **尾数运算**：阶码对齐后直接对尾数运算。

(3) **结果规格化**：对运算结果进行规格化处理（使补码尾数的最高位和尾数符号相反）。如溢出则需右规；如不是规格化时应左规。

(5) **判断溢出**：判断阶码是否溢出，下溢则将运算结果置0（机器0），上溢则中断。

(4) **舍入操作**：丢失位进行0舍1入或恒置1处理。

海纳百川 自强不息 厚德笃学 知行合一



具体说明



对阶运算(小阶向大阶对齐)

尾数为原码时，尾数右移，符号位不动，最高位补0

尾数为补码时，尾数右移，符号也移位，最高位补符号位。

(1) 求阶差

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

(2) 对阶原则

小阶向大阶看齐

海纳百川 自强不息 厚德笃学 知行合一





例： 求 $0.1101 \times 2^3 + 0.1001 \times 2^2 = ?$

小阶对大阶 $0.1101 \times 2^3 + 0.0010 \times 2^3 = 0.1111 \times 2^3$

舍掉的是 0.0001×2^3

如大阶对小阶 $0.0100 \times 2^4 + 0.1001 \times 2^3 = 0.1101 \times 2^4$

则舍掉的是 0.1100×2^3

海纳百川 自强不息 厚德笃学 知行合一



【例】 $x = (0.1101)_2 \times 2^{(01)_2}$ $y = (-0.1010)_2 \times 2^{(11)_2}$

求 $x+y$

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

(1) 对阶 (设阶码的符号也为双符号位)

$$\textcircled{1} \text{ 求阶差 } [\Delta j]_{\text{补}} = [j_x]_{\text{补}} - [j_y]_{\text{补}} = 00, 01$$

$$\begin{array}{r} + \quad 11, 01 \\ \hline 11, 10 \end{array}$$

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

② 对阶 $[x]_{\text{补}}' = 00, 11; 00.0011$

(2) 尾数求和

$$\begin{array}{r} [S_x]_{\text{补}}' = 00.0011 \quad \text{对阶后的}[S_x]_{\text{补}}' \\ + \quad [S_y]_{\text{补}} = 11.0110 \\ \hline 11.1001 \\ \therefore [x+y]_{\text{补}} = 00, 11; 11. 1001 \end{array}$$

■规格化：原码尾数值高位为1，补码尾数值高位与符号相反

(1) 规格化数的定义

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

(2) 规格化数的判断

$S > 0$ 正数	规格化形式	$S < 0$ 负数	规格化形式
真值	$0.1 \times \times \cdots \times$	真值	$-0.1 \times \times \cdots \times$
原码	$0.\boxed{1} \times \times \cdots \times$	原码	$1.\boxed{1} \times \times \cdots \times$
补码	$\boxed{0.1} \times \times \cdots \times$	补码	$\boxed{1.0} \times \times \cdots \times$
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 符号位和最高数值位不同

特例



$$S = -\frac{1}{2} = -0.100 \dots 0$$

$$[S]_{\text{原}} = 1.100 \dots 0$$

$$[S]_{\text{补}} = \boxed{1.1}00 \dots 0$$

$\therefore [-\frac{1}{2}]_{\text{补}}$ 不是规格化的数，须用原码表示

$$S = -1$$

$$[S]_{\text{补}} = \boxed{1.0}00 \dots 0$$

$\therefore [-1]_{\text{补}}$ 是规格化的数

海纳百川 自强不息 厚德笃学 知行合一



(3) 左规

尾数左移1位，阶码减1，直到数符和第一数位不同为止

上例 $[x+y]_{\text{补}} = 00, 11; 11. 1001$

左规后 $[x+y]_{\text{补}} = 00, 10; 11. 0010$

$$\therefore x + y = (-0.1110)_2 \times 2^{(10)2}$$

(4) 右规

当尾数溢出（>1）时，需右规

即尾数出现 **01.** $\times \times \cdots \times$ 或 **10.** $\times \times \cdots \times$ 时

尾数右移1位，阶码加1

海纳百川 自强不息 厚德笃学 知行合一



【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: $[x]_{\text{补}} = 00, 010; 00. 110100$

$[y]_{\text{补}} = 00, 001; 00. 101100$

① 对阶 (设阶码的符号也为双符号位)

$$[\Delta j]_{\text{补}} = [j_x]_{\text{补}} - [j_y]_{\text{补}} = \begin{array}{r} 00, 010 \\ + 11, 111 \\ \hline \end{array}$$

丢失  $100, 001$

阶差为 +1

$\therefore S_y \rightarrow 1, j_y + 1$

$\therefore [y]_{\text{补}'} = 00, 010; 00. 010110$

② 尾数求和

$[S_x]_{\text{补}} = 00. 110100$

$+ [S_y]_{\text{补}'} = 00. 010110$

$\hline 01. 001010$

对阶后的 $[S_y]_{\text{补}'}$

尾数溢出需右规

③ 右规

$$[x+y]_{\text{补}} = 00, 010; 01. 001010$$

右规后

$$[x+y]_{\text{补}} = 00, 011; 00. 100101$$

$$\therefore x+y = 0. 100101 \times 2^{11}$$

■ 舍入操作：0舍1入 或 恒置1

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



中华人民共和国成立70周年



1949—2019
大连理工大学 建校70周年

海纳百川 自强不息 厚德笃学 知行合一



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

$[x]_{\text{补}} = 11, 011; 11. 011000$ $[y]_{\text{补}} = 11, 100; 00. 111000$

① 对阶

$$\begin{array}{r} [\Delta j]_{\text{补}} = [j_x]_{\text{补}} - [j_y]_{\text{补}} = 11, 011 \\ + 00, 100 \\ \hline 11, 111 \end{array}$$

阶差为 -1 $\therefore S_x \longrightarrow 1, j_x + 1$

$\therefore [x]_{\text{补}}' = 11, 100; 11. 101100$

(尾数右移: 第一符号位不变。)



② 尾数求和

$$\begin{array}{r}
 [S_x]_{\text{补}'} = 11.101100 \\
 + [-S_y]_{\text{补}} = 11.001000 \\
 \hline
 110.110100
 \end{array}$$

③ 右规

$$[x-y]_{\text{补}} = 11, 100; 10.110100$$

右规后

$[x-y]_{\text{补}} = 11, 101; 11.011010$ (采用0舍1入方式, 无进位。) 将其转换二进制真值后为:

$$x - y = (-0.100110) \times 2^{-11} = \left(-\frac{19}{32}\right) \times 2^{-3}$$

海纳百川 自强不息 厚德笃学 知行合一

浮点数加/减运算小结



加减法执行下述五步完成运算：

① “对阶”操作

比较两浮点数阶码的大小，求出其差 ΔE ，保留其大值 E ， $E = \max(E_x, E_y)$ 。当 $\Delta E \neq 0$ 时，将阶码小的尾数右移 ΔE 位，并将其阶码加上 ΔE ，使两数的阶码值相等。**对阶原则：小阶向大阶看齐。**

② 尾数加减运算

执行对阶之后，两尾数进行加减操作。

③ 规格化操作

规格化的目的是使得尾数部分的绝对值尽可能以最大值的形式出现。

⑤ 检查阶码是否溢出

阶码溢出表示浮点数溢出。在规格化和舍入时都可能发生溢出，若阶码正常，加/减运算正常结束。若阶码下溢，则设置机器运算结果为机器零，若上溢，则设置溢出标志。

④ 舍入

在执行右规或者对阶时，尾数的低位会被移掉，使数值的精度受到影响，常用“0”舍“1”入法。当移掉的部分最高位为1时，在尾数的末尾加1，如果加1后又使得尾数溢出，则要再进行一次右规。

