第五章　运算方法与运算器（二）

秦磊华 计算机学院

本节主要内容

基于补码数据表示研究运算方法和设计运算器（简）

5.2 浮点加/减运算

1. 浮点数加/减运算方法

$$X = 2^{E_X} M_X \qquad Y = 2^{E_Y} M_Y$$

$X \pm Y = ?$

如： $E_x = E_y$ $S = 2^{E_x}(M_x \pm M_y)$

如： $E_x \neq E_y$? 使两数的阶码变成相等 ⟹ 对阶

浮点运算步骤： **对阶、尾数运算、规格化、舍入、溢出判断**

1. 浮点数加/减运算方法

1)对阶 (使得两者的阶码相等)

大变小 or 小变大 ? ⟹ 小变大！，对阶的同时，尾数要同步右移

$2^8*(0.11000) + 2^6*(0.00111)$

♦大阶变小阶:

$2^8*(0.11000) \rightarrow 2^6*(11.000) \rightarrow 2^5*(110.000)$

♦小阶变大阶:

$2^6*(0.00111) \rightarrow 2^7*(0.000111) \rightarrow 2^8*(0.0000111)$

1. 浮点数加/减运算方法

2) 运算结果规格化

(1) 为什么需要进行规格化 -浮点数可表达的多样性

$$2^{7}*(00.11000) = 2^{8}*(00.01100)$$

$$2^{7}*(10.11000) = 2^{8}*(11.01100)$$

$$2^{7}*(01.11000) = 2^{8}*(00.11100)$$

1. 浮点数加/减运算方法

2) 运算结果规格化

(2)规格化的标准是什么?

尾数不为零时，其绝对值大于等于1/2

♦真值规格化数	0.1XXXX	-0.1XXXX
♦原码规格化数	0.1XXXX	1.1XXXX
♦补码规格化数	00.1XXXX	11.0XXXX
♦补码非规格化数	00.0XXXX	11.1XXXX
	01.XXXXX	10.XXXXX

1. 浮点数加/减运算方法

2) 运算结果规格化

(3) 如何规格化？ - 使非规格化尾数变成规格化尾数，同步变化阶码

补码规格化数　　00.1XXXX　11.0XXXX

00.0XXXX　11.1XXXX ⇨ 左移规格化 ⇨ 左移几位?

01.XXXXX　10.XXXXX ⇨ 右移规格化 ⇨ 右移1位

1. 浮点数加/减运算方法

3) 舍入处理

0	1	1	0	0	0	1	1

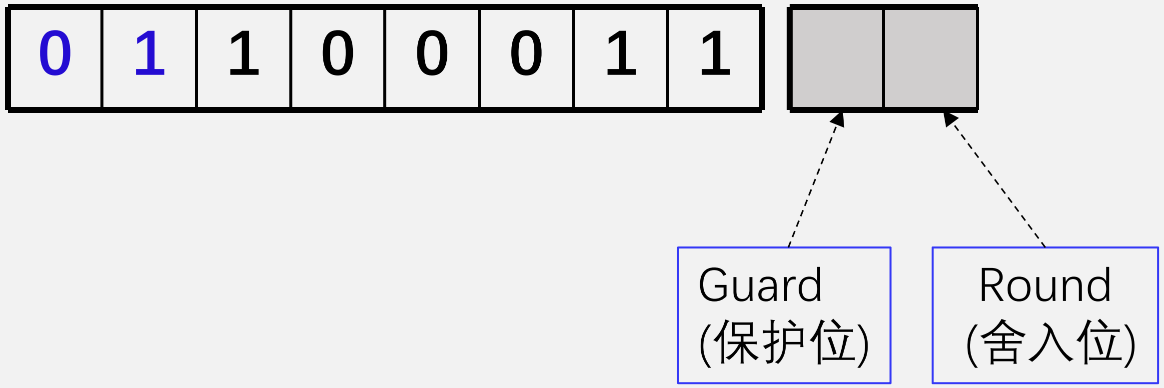
0	0	1	1	0	0	0	1	1

- ♦右移动规格化后低位部分丢失了数据位，从而产生误差
- ♦有多种处理方法：0舍1入，截去法
- ♦不同舍入方式对精度产生不同的影响！如果减少影响？

1. 浮点数加/减运算方法

3) 舍入处理

增加浮点运算附加位



IEEE 754

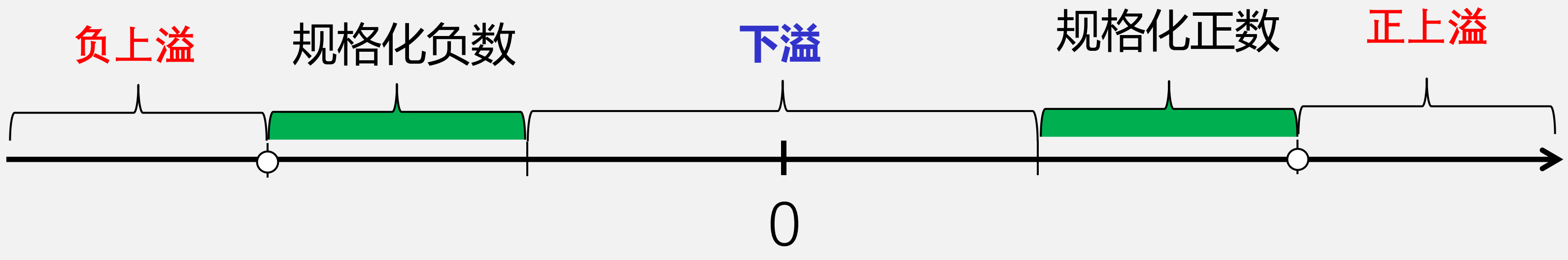
增加浮点运算附加位 意义何在?

5.2 浮点运算(加/减)

1. 浮点数加/减运算方法

4) 溢出的标准及处理

- 通过阶码是否溢出判断浮点数的溢出情况(双符号位检测)
- 阶码符号位为 01 → 浮点数上溢 |X|→∞
- 阶码符号位为 10 → 浮点数下溢 |X|→0



2. 浮点数加/减运算举例

例1 两浮点数 $x = 2^{101} \times 0.11011011$，$y = 2^{111} \times (-0.10101100)$。假设尾数在计算机中以补码表示，尾数位共12位，采用双符号位，阶码以补码表示，共5位，也采用双符号位, 求 x + y。

解：将x , y转换成浮点数据格式

$[x]_{浮}$ = 00 101, 00.11011011

$[Y]_{浮}$ = 00 111, 11.01010100

1) 对阶

$[E_x - E_y]_{补} = [E_x]_{补} + [-E_y]_{补} = 00101 + 11001 = 11110 = -2 < 0$

小阶对大阶， X阶码加2， X尾数右移2位

2. 浮点数加/减运算举例

$[x]_{浮}$ = **00 111, 00.0011011011** 保留位

$[Y]_{浮}$ = 00 111, 11.01010100

2)尾数求和

$[X+Y]_{浮}$ = **00** 111, **11**.10001010 11 保留位参与运算

3)结果规格化

$[X+Y]_{浮}$ = **00** 110, **11**.000101011 非规数，左归一位, 阶码减一

4)舍入处理

$[X+Y]_{浮}$ = **00** 110, **11**.00010110 (0舍1入法)

$[X+Y]_{浮}$ = **00** 110, **11**.00010101 (截去法)

5)溢出判断

$[X+Y]_{浮} = 2^{110} \times (-0.11101011)$ **无溢出**

2. 浮点数加/减运算举例

例2， 已知 **$X=2^{111}\times 0.11111111$**， **$Y=2^{111}\times 0.10000001$**， 求 **X+Y**

解： $[X]_{浮}$=00111, 00.1111 1111 ， $[Y]_{浮}$=00111, 00.1000 0001

1)尾数求和

$[X+Y]_{浮}$ = 00111, 01.1000 0000

2)结果规格化

$[X+Y]_{浮}$ = **01** 000, **00**.1100 0000　右移一位规格化, 阶码 +1

3)舍入处理

$[X+Y]_{浮}$ = **01** 000, **00**.11000000

4)溢出判断

阶码双符号位 01， 上溢

3. 计算机中关于运算的案例分析

```c
union { char c[4]; float f;  int i;} t1,t2,t3;
main(){
   t1.i = 0X7F000000; t2.i = 0X7F7FFFFF;  // Max float
   t3.f = t1.f + t1.f;
   printf("%08X   %f\n",t1.i,t1.f);
   printf("%08X   %f\n",t2.i,t2.f);
   printf("%08X   %f\n",t3.i,t3.f);
}
```

7F000000	170141183460469230000000000000000000000.000000
7F7FFFFF	340282346638528860000000000000000000000.000000
7F800000	1.#INF00

华中科技大学 计算机科学与技术学院 School of Computer Science & Technology, HUST

3. 计算机中关于运算的案例分析

```
union { char c[4]; float f;  int i;}  t1,t2,t3,t4;
main()  {
     t1.i = 0X00000001;  t2.i = 0X00C00000; t3.i = 0X00800000; //Minus float
     t4.f = t2.f - t3.f;
     printf("%08X   %.61f\n",t1.i,t1.f);  printf("%08X   %.61f\n",t2.i,t2.f);
     printf("%08X   %.61f\n",t3.i,t3.f);  printf("%08X   %.61f\n",t4.i,t4.f);}
```

t2.f	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	⇨ 1.1(尾数)
t3.f	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	⇨ 1.0(尾数)

00000001   0.0000000000000000000000000000000000000000000014012984643248171

00C00000   0.0000000000000000000000000000000000000176324152623343130000000

00800000   0.0000000000000000000000000000000000000117549435082228750000000

00400000   0.0000000000000000000000000000000000000058774717541114375000000

第二部分完