计算机组织结构

6 浮点数运算

任桐炜

2021年10月9日

南京大学
NANJING UNIVERSITY

教材对应章节

第3章 运算方法和运算部件

第9章 计算机算术

回顾：IEEE 754浮点数表示

	Single Precision (32 bits)				Double Precision (64 bits)			
	Sign	**Biased exponent**	**Fraction**	**Value**	**Sign**	**Biased exponent**	**Fraction**	**Value**
positive zero	0	0	0	0	0	0	0	0
negative zero	1	0	0	−0	1	0	0	−0
plus infinity	0	255 (all 1s)	0	∞	0	2047 (all 1s)	0	∞
minus infinity	1	255 (all 1s)	0	−∞	1	2047 (all 1s)	0	−∞
quiet NaN	0 or 1	255 (all 1s)	≠0	NaN	0 or 1	2047 (all 1s)	≠0	NaN
signaling NaN	0 or 1	255 (all 1s)	≠0	NaN	0 or 1	2047 (all 1s)	≠0	NaN
positive normalized nonzero	0	$0 < e < 255$	f	$2^{e-127}(1.f)$	0	$0 < e < 2047$	f	$2^{e-1023}(1.f)$
negative normalized nonzero	1	$0 < e < 255$	f	$-2^{e-127}(1.f)$	1	$0 < e < 2047$	f	$-2^{e-1023}(1.f)$
positive denormalized	0	0	$f \neq 0$	$2^{e-126}(0.f)$	0	0	$f \neq 0$	$2^{e-1022}(0.f)$
negative denormalized	1	0	$f \neq 0$	$-2^{e-126}(0.f)$	1	0	$f \neq 0$	$-2^{e-1022}(0.f)$

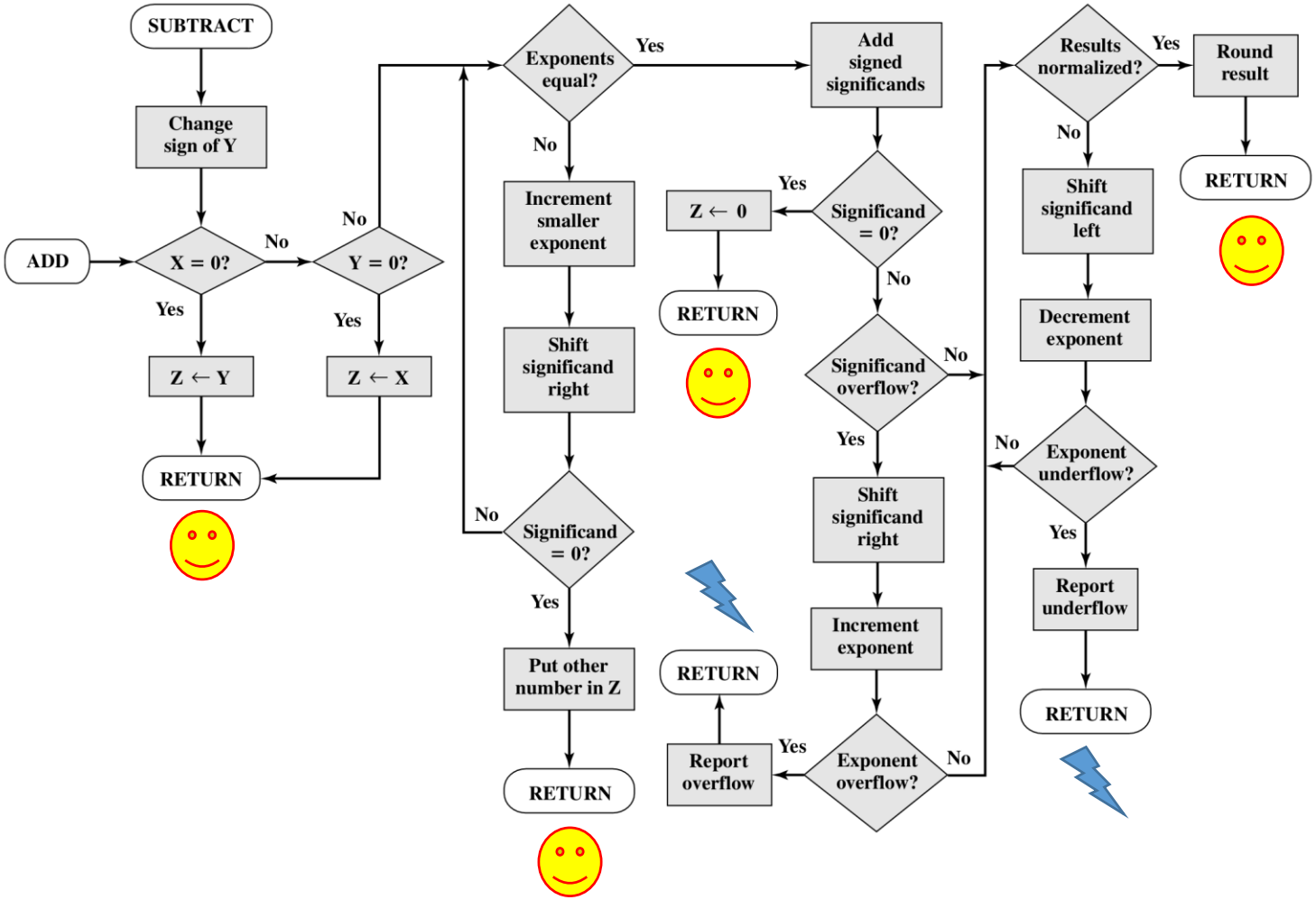
加法和减法

- 必须确保两个操作数具有相同的指数值

$$X + Y = (X_S \times B^{X_E - Y_E} + Y_S) \times B^{Y_E}$$
$$X - Y = (X_S \times B^{X_E - Y_E} - Y_S) \times B^{Y_E}$$
$$X_E \leq Y_E$$

- 步骤过程
  - 检查0
  - 对齐有效值
  - 加或减有效值
  - 规格化结果

加法和减法（续）


SUBTRACT
Change sign of Y
ADD
X = 0?
No
Yes
Y = 0?
No
Yes
Z ← Y
Z ← X
RETURN
Exponents equal?
Yes
No
Increment smaller exponent
Shift significand right
Significand = 0?
No
Yes
Put other number in Z
RETURN
Add signed significands
Significand = 0?
Yes
Z ← 0
RETURN
No
Significand overflow?
No
Yes
Shift significand right
Increment exponent
Exponent overflow?
Yes
No
Report overflow
RETURN
Results normalized?
Yes
No
Shift significand left
Decrement exponent
Exponent underflow?
No
Yes
Report underflow
RETURN
Round result
RETURN

加法和减法（续）

- 阶值上溢
  - 正阶值超过可能的最大允许阶值
  - 标记为 $+\infty$ 或者 $-\infty$
- 阶值下溢
  - 负阶值小于可能的最小允许阶值
  - 报告为 0

加法和减法（续）

- 有效值上溢
  - 同符号的两个有效值相加可能导致最高有效位的进位
  - 通过重新对齐来修补
- 有效值下溢
  - 在有效值对齐过程中，可能有数字被移出右端最低位而丢失
  - 需要某种形式的四舍五入

原码加法

- 如果两个操作数有相同的符号，做加法；否则，做减法
  - 做加法：直接相加
    - 如果最高位有进位，则溢出
    - 符号和被加数（被减数）相同
  - 做减法：加第二个操作数的补数
    - 如果最高位有进位，正确 (符号与被减数相同)
    - 否则，计算它的补码 (符号与被减数相反)

[明鑫, 171250553]

原码加法（续）

- 例子

0.8125+0.625 = 1.4375

```
   1101
 + 1010
------
  10111
```

0.4375

0.8125-0.625 = 0.1875

```
   1101
 + 0110
------
  10011
```

+ 0.1875

0.625-0.8125=-0.1875

```
   1010
 + 0011
------
   1101
```

↓

0011

- 0.1875

加法和减法（续）

- 例子 (续)

  0.5 – (-0.4375) = 0.9375

  0.5　　0 01111110 000…00 (23)

  -0.4375　1 01111101 110…00 (21)

  01111110 – 01111101 = 01111110 + 10000011 = 00000001

  ```
      1 0000…00
  +   0 1110…00
  ---------------
      1 1110…00
  ```

  0 01111110 1110…00 (20)

[沈佳楠, 121250118]

加法和减法（续）

- 例子 (续)

  0.5 + (-0.4375) = 0.0625

  0.5　　0 01111110 000…00 (23)

  -0.4375　1 01111101 110…00 (21)

  01111110 – 01111101 = 01111110 + 10000011 = 00000001

  1 0000…00

  \+ 1 0010…00

  ---

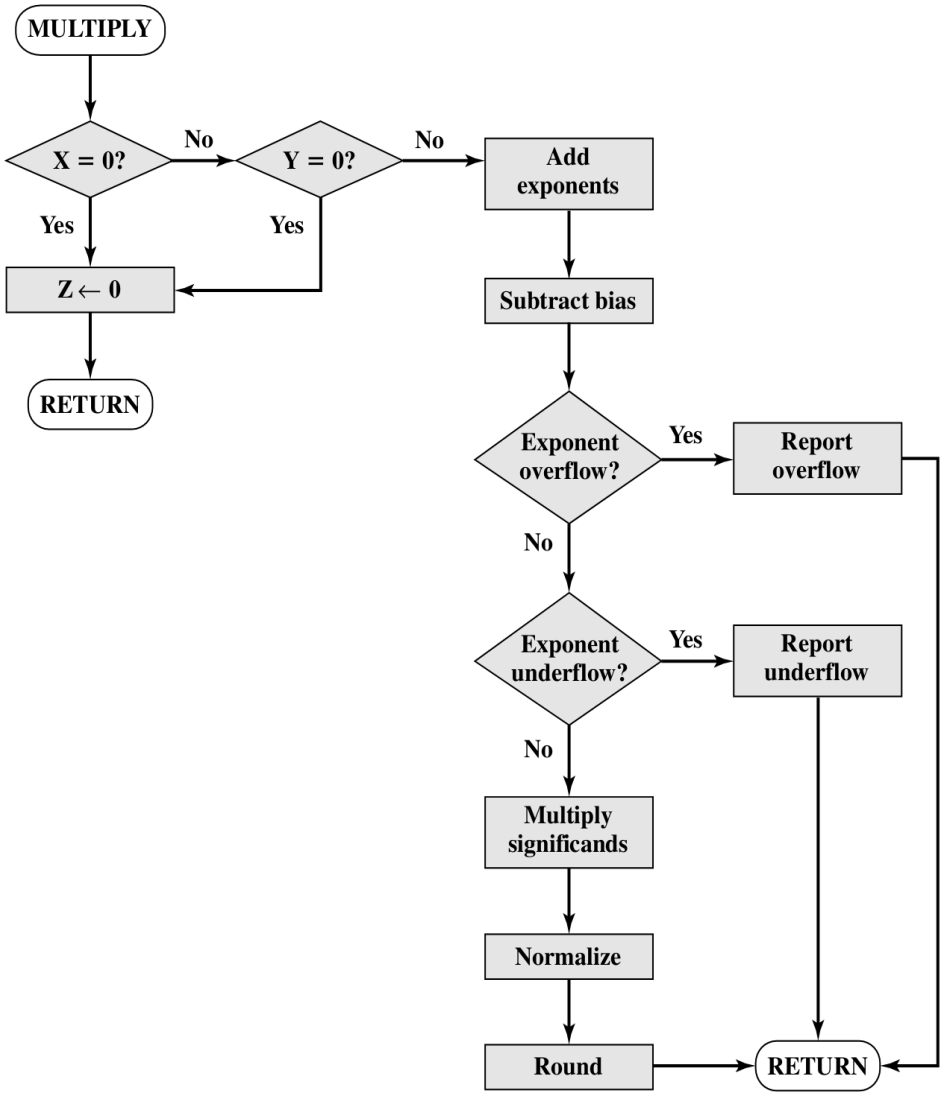
  1 0 0010…00

  0 01111011 000…00 (23)

[沈佳楠, 121250118]

乘法

- 无论哪个操作数是0，乘积即为0
- 从阶值的和中减去一个偏移量
- 有效值相乘
- 结果的规格化和舍入处理
  - 规格化可能导致阶值下溢



乘法（续）

- 例子

0.5×0.4375 = 0.21875

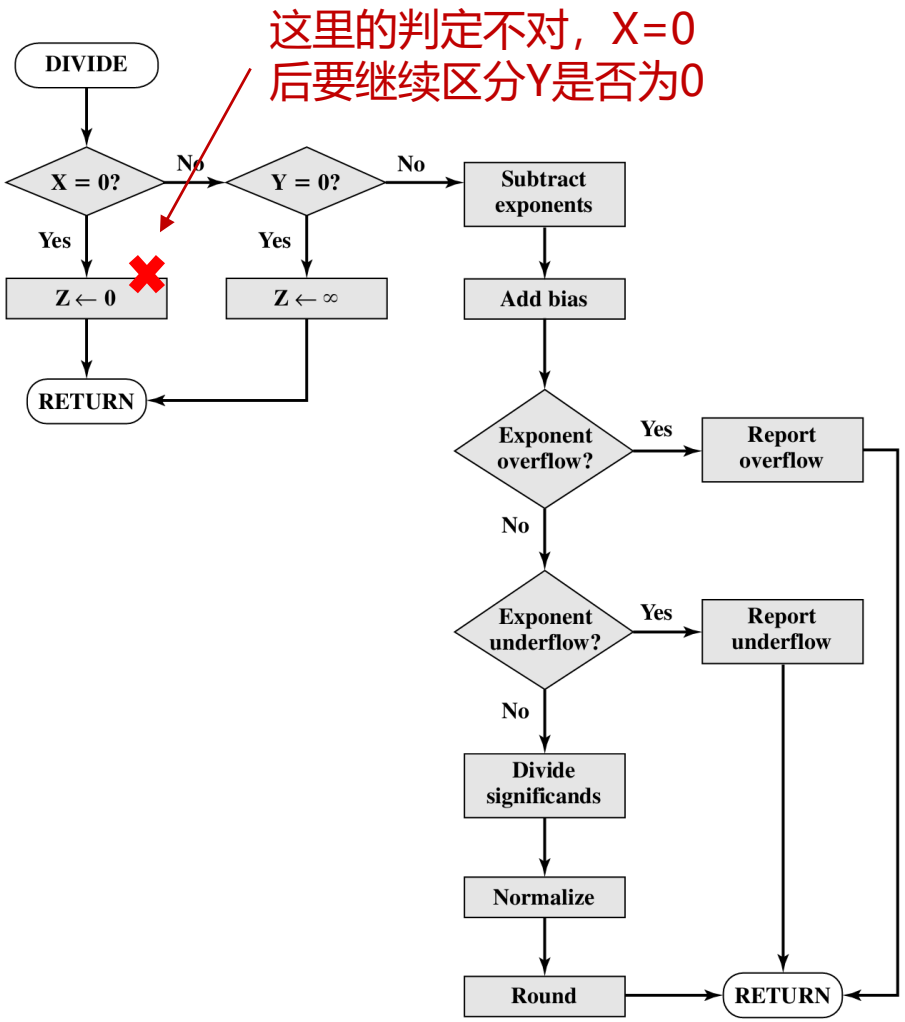
```
   01111110
 + 01111101
------------
   11111011
 - 01111111
------------
   01111100
```

		乘积	Y
initial		0000…00	1110…00
0	->	0000…00	01110…0
		……	
0	->	0000…00	00…0111
1	+	1000…00	00…0111
	->	0100…00	00…0011
1	+	1100…00	00…0011
	->	0110…00	00…0001
1	+	1110…00	00…0001
	->	0111…00	00…0000

0 01111100 110…00 (21)

除法

- 如果除数为0，则报告出错，或将结果设置为无穷大
- 如果被除数是0，则结果是0
- 被除数的阶值减除数的阶值，加上偏移量
- 有效值相除
- 结果规格化和舍入处理



[张许妍，181250187]

除法（续）

除数 1000…00

- 例子

0.4375/0.5=0.875

```
   01111101
 - 01111110
------------
   11111111
 + 01111111
------------
   01111110
```

		余数	商
initial		1110…00	00…0000
enough	-	0110…00	00…0000 1
	<-	1100…00	00…0001
enough	-	0100…00	00…0001 1
	<-	1000…00	00…0011
enough	-	0000…00	00…0011 1
	<-	0000…00	00…0111
not		0000…00	00…0111 0
	<-	0000…00	0…01110
		.....	
not		0000…00	01110…0 0
	<-	0000…00	11100…0

0 01111110 110…00 (21)

精度考虑

- 保护位
  - 寄存器的长度几乎总是大于有效值位长与一个隐含位之和
  - 寄存器包含的这些附加位，称为**保护位**
  - 保护位用0填充，用于扩充有效值的右端

$$x = 1.00\ldots00 \times 2^1, y = 1.11\ldots11 \times 2^0$$

```
 x = 1.000.....00 × 2^1
-y = 0.111.....11 × 2^1
 z = 0.000.....01 × 2^1
   = 1.000.....00 × 2^-22
```

不使用保护位

```
 x = 1.000.....00 0000 × 2^1
-y = 0.111.....11 1000 × 2^1
 z = 0.000.....00 1000 × 2^1
   = 1.000.....00 0000 × 2^-23
```

使用保护位

精度考虑（续）

- 舍入
  - 对有效值操作的结果通常保存在更长的寄存器中
  - 当结果转换回浮点格式时，必须要去掉多余的位
    - 就近舍入：结果被舍入成最近的可表示的数
    - 朝+∞舍入：结果朝正无穷大方向向上舍入
    - 朝−∞舍入：结果朝负无穷大方向向下舍入
    - 朝 0 舍入：结果朝 0 舍入

精度考虑（续）

- 例子
  - 假设用16位表示一个浮点数，其中1位为符号，9位为有效值，6位为阶值（偏移量为31）
  - 代表 652.13 和 -7.48

+ 652.13	0 101000 010001100	+ 652.0
-7.48	1 100001 110111101	- 7.4765625

精度考虑（续）

- 例子 (续)
  - 假设 ALU 有 16 位，分别用使用保护位和不使用保护位的方法计算下面式子
    - 652.13 + (-7.48)
    - 652.13 -(-7.48)

精度考虑（续）

- 例子 (续)：652.13 + (-7.48) = 644.65

101000 – 100001 = 000111

```
   1 010001100
 - 0 000000111
-------------
   1 010000101
```

0 101000 010000101

645.0

```
   1 010001100 000000
 - 0 000000111 011110
-------------------
   1 010000100 100010
```

0 101000 010000100

644.0

精度考虑（续）

- 例子 (续)：652.13 - (-7.48) = 659.61

101000 – 100001 = 000111

```
   1 010001100
 + 0 000000111
-------------
   1 010010011
```

0 101000 010010011

659.0

```
   1 010001100 000000
 + 0 000000111 011110
-------------------
   1 010010011 011110
```

0 101000 010010011

659.0

[薛家豪，121250185]

总结

- 浮点算数运算
  - 加法
  - 减法
  - 乘法
  - 除法
- 精度考虑

谢谢

rentw@nju.edu.cn

南京大學
NANJING UNIVERSITY