《计算机系统》
浮点数

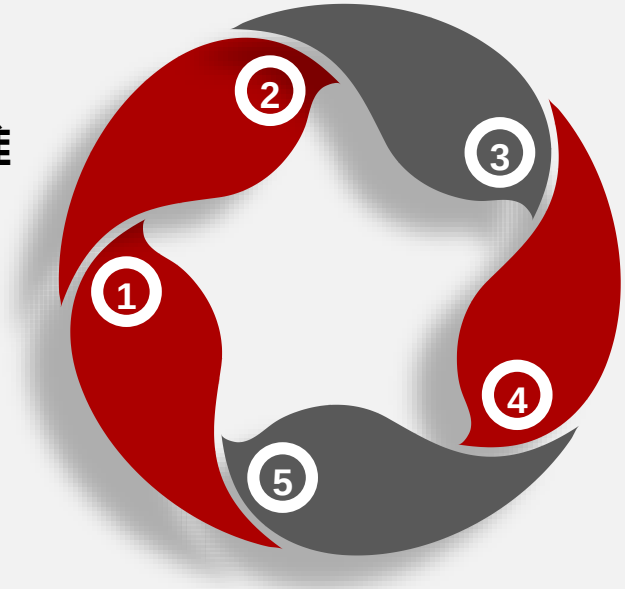
湖南大学

《计算机系统》课程教学组

内容提要

IEEE 浮点数标准

示例与性质



二进制小数

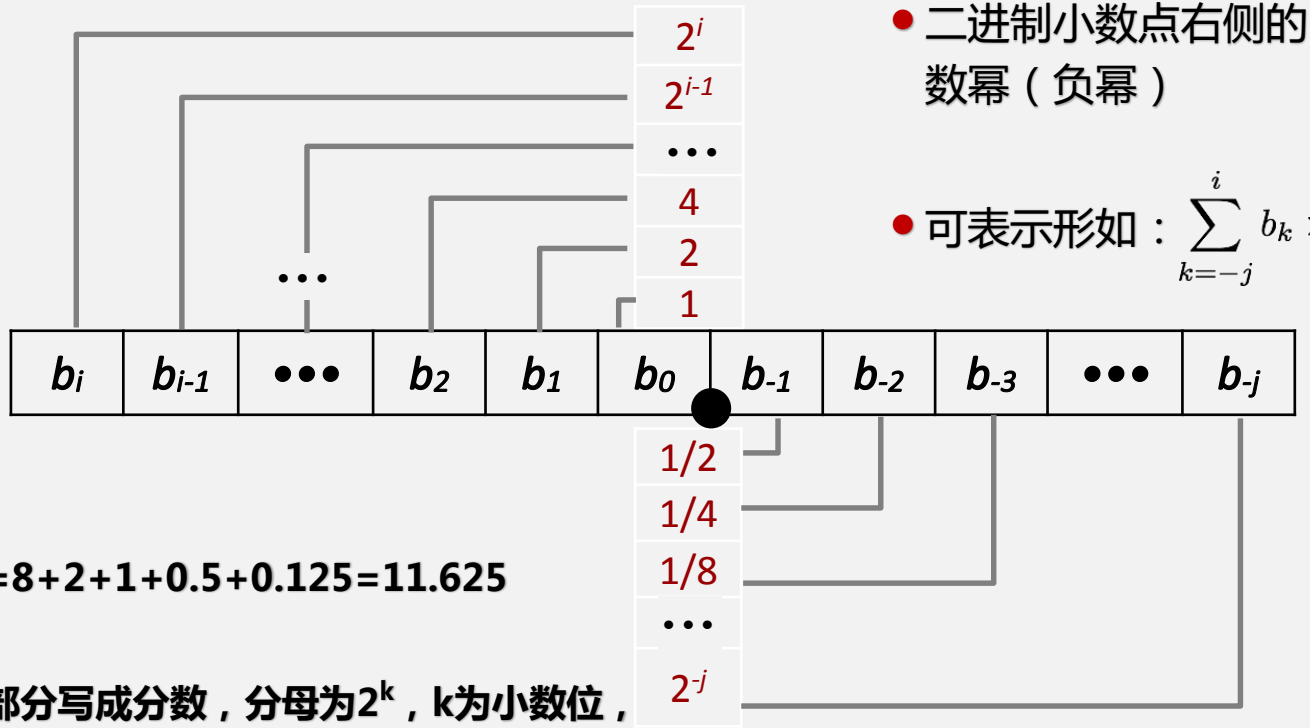
舍入与运算

C语言中的浮点数

二进制小数

- **思考：** 1011.101_2?

- **表示方法**
  - 二进制小数点右侧的“位”表示2的分数幂（负幂）
  - 可表示形如：$\sum_{k=-j}^{i} b_k \times 2^k$ 的有理数



- **答案：**

$1011.101_2=8+2+1+0.5+0.125=11.625$

窍门：小数部分写成分数，分母为2^k，k为小数位，分子为小数部分二进制的值。即：

$.101_2=5/8=0.625$

二进制小数
举例

十进制	二进制
5 3/4	101.11_2
2 7/8	10.111_2
1 13/16	1.1101_2

- 观察
  - 小数点右移一位——乘2
  - 小数点左移一位——除2
  - 形如 $0.111111\ldots_2$ 表示刚好小于1.0的数
    - $1/2 + 1/4 + 1/8 + \ldots + 1/2^i + \ldots \rightarrow 1.0$
    - 也可以简单的用 $1.0 - \varepsilon$ 来表示

二进制小数
表示范围

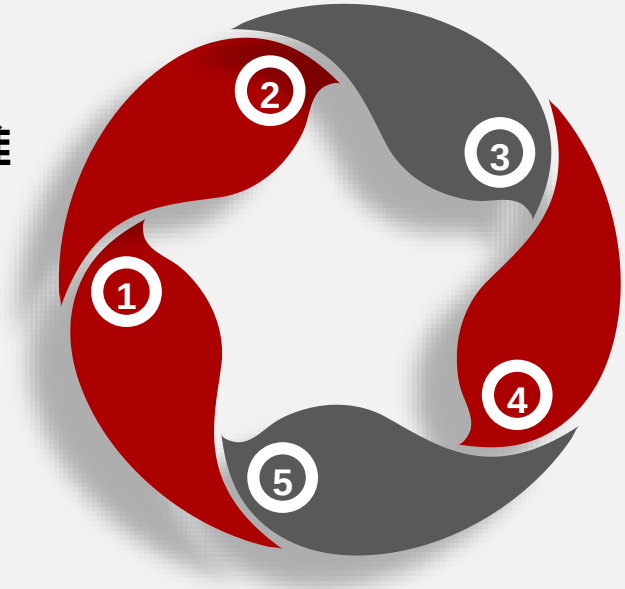
- 限制
  - 只能精确表示诸如 $x/2^k$的数
  - 其他的值只能近似表示

Value	Representation
1/3	$0.0101010101[01]\ldots_2$
1/5	$0.001100110011[0011]\ldots_2$
1/10	$0.0001100110011[0011]\ldots_2$

内容提要

IEEE 浮点数标准

示例与性质



二进制小数

舍入与运算

C语言中的浮点数

IEEE 浮点数

- **IEEE 754标准**
  - 于1985建立（之前由每个计算机制造商设计自己的规则）
  - 支持所有主流的CPU

- **面向数字运算的精确性**
  - 支持舍入，溢出等操作
  - 定义在一组小而一致的**规则**上——相当优雅，容易理解

IEEE 浮点数

- 数学形式:

$$(-1)^s\ M\ 2^E$$

  - 符号位s 确定了这个数是负数还是正数，数值0的符号位特殊处理
  - 尾数M（Significand） 是一个二进制小数，通常规定在 范围 [1.0,2.0)中.
  - 阶码 E （Exponent）表示2的幂

浮点数类型

- **单精度: 32 bits**

- **双精度: 64 bits**

- **扩展精度: 80 bits (Intel only)**

浮点数表示方法

类别1：规格化值

- 判断条件: exp ≠ 000…0 且 exp ≠ 111…1，即：阶码不为全0或全1
- 阶码字段被解释为以偏置（ biased）形式表示的有符号整数:

$$E = Exp - Bias$$

  - Exp: 无符号数 exp
  - Bias = $2^{k-1} - 1$, 其中k为阶码位数（单精度：127，双精度：1023）
- 尾数： $M = (1.xxx…x)_2$
  - xxx…x: frac的位表示
  - 最小值 000…0 (M = 1.0)
  - 最大值：111…1 (M = 2.0 – ε)

规格化值示例

- **值: Float F = 15213.0;**
  - $15213_{10} = 11101101101101_2 = 1.1101101101101_2 \times 2^{13}$

- **尾数**
  - **M = $1.\underline{1101101101101}_2$**
  - **frac = $\underline{1101101101101}0000000000_2$**

- **阶码**
  - *E* = 13
  - *Bias* = 127
  - *Exp* = 140 = **10001100_2**

- **Result:**

0	10001100	11011011011010000000000
符号	阶码	尾数

浮点数表示方法

类别2：非规格化值

- 判断条件: exp = 000…0 即：阶码为全0
- 阶码：E = 1 – Bias （为了非规格化与规格化值之间的平滑过渡）
- 尾数： M = $(0.xxx…x)_2$（xxx…x: frac的位表示）

- 例如：
  - exp = 000…0, frac = 000…0, 此时值为“0”，符号位决定“+0”或者“-0”
  - Exp = 000…0, frac ≠ 000…0，此时为非常接近0.0的数

浮点数表示方法

类别3：特殊值

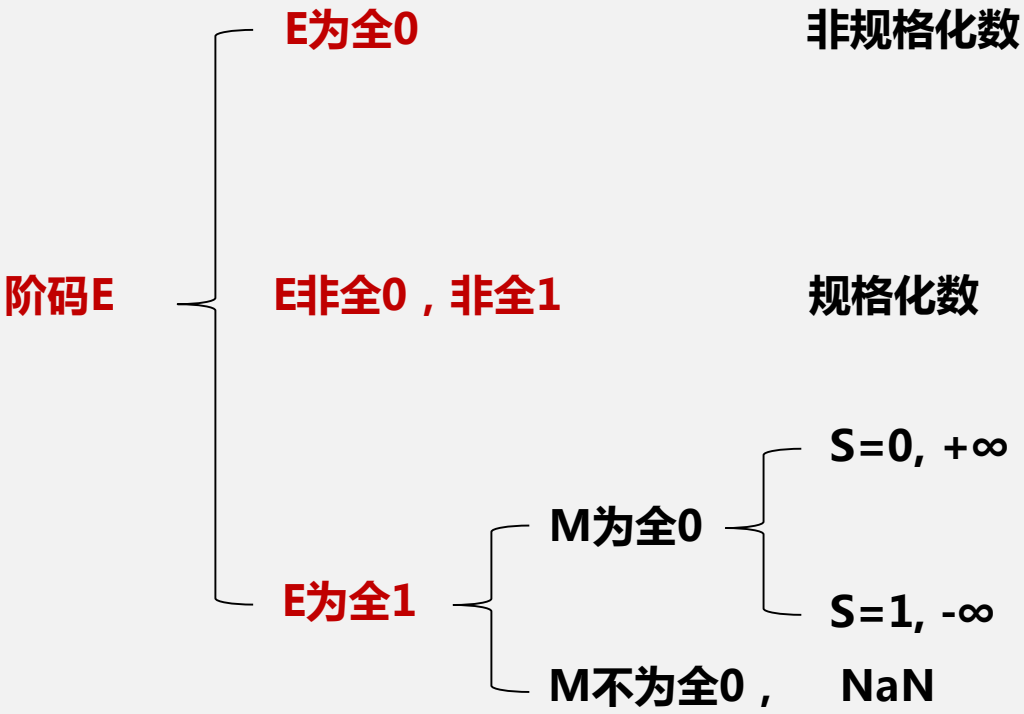
- 判断条件: exp = 111…1， 即：阶码为全1

- 情况1：exp = 111…1, frac = 000…0,
  - 表示的是无穷大，由符号位决定是“+∞”还是“-∞”
  - 可用来表示溢出结果，例如 1.0/+0.0 = +∞； 1.0/-0.0 = -∞

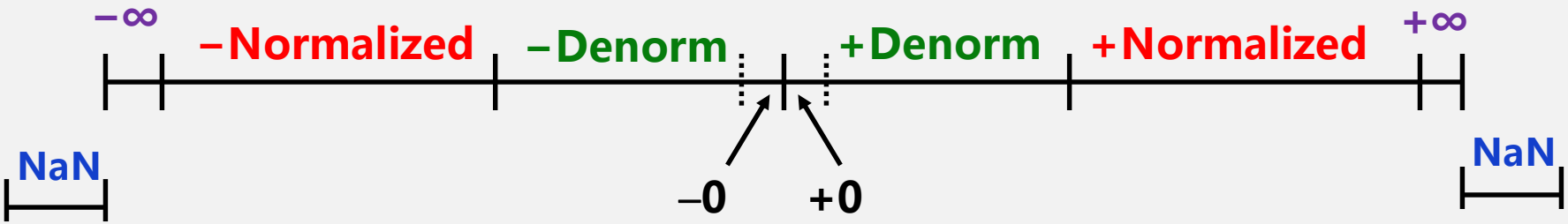
- 情况2：exp = 111…1, frac ≠ 000…0,
  - 表示的是不是一个数（NaN），用来表示一些无法表示的数，例如：

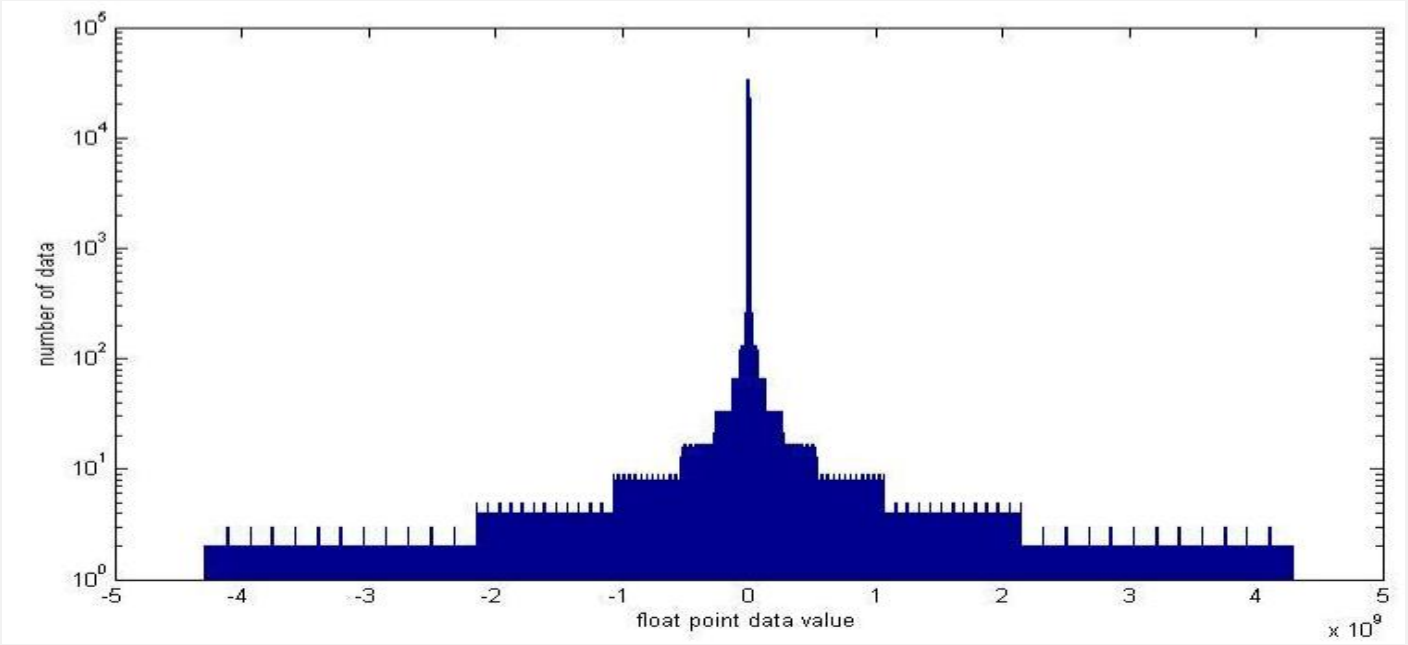
    sqrt(-1)， ∞-∞， ∞* ∞

浮点数表示方法



浮点数表示范围

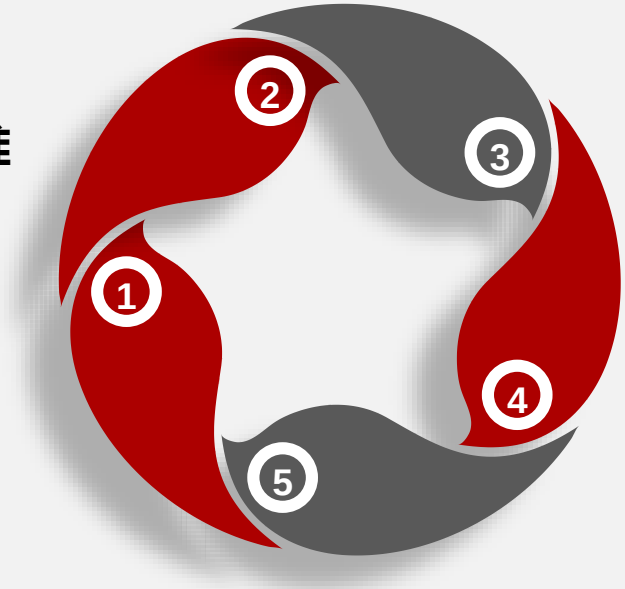

−∞
−Normalized
−Denorm
+Denorm
+Normalized
+∞
NaN
−0
+0
NaN


number of data
float point data value
x 10^9

内容提要

IEEE 浮点数标准

示例与性质

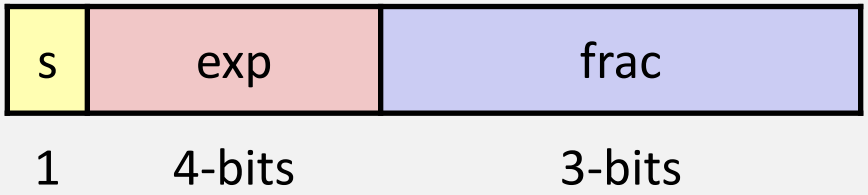


二进制小数

舍入与运算

C语言中的浮点数

浮点数示例



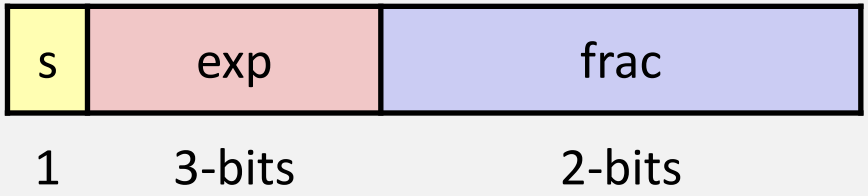
- 8-位浮点数表示
  - 一位符号位
  - 四位阶码位
  - 三位尾数位
- IEEE规范
  - 规格化数, 非规格化数
  - 能够表示 0, NaN, 无穷

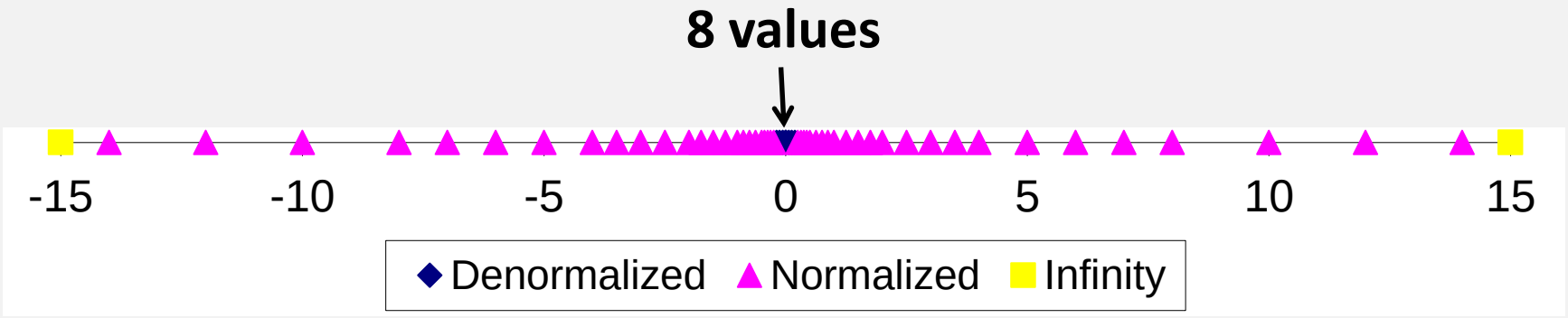
浮点数示例

	s exp frac	E	Value	
非规格化数	0 0000 000	-6	0	
	0 0000 001	-6	1/8*1/64 = 1/512	最小的非规格化数（接近0）
	0 0000 010	-6	2/8*1/64 = 2/512	
	…… ……			
	0 0000 110	-6	6/8*1/64 = 6/512	
	0 0000 111	-6	7/8*1/64 = 7/512	最大的非规格化数
规格化数	0 0001 000	-6	8/8*1/64 = 8/512	最小的规格化数
	0 0001 001	-6	9/8*1/64 = 9/512	
	…… ……			
	0 0110 110	-1	14/8*1/2 = 14/16	
	0 0110 111	-1	15/8*1/2 = 15/16	从下最靠近1
	0 0111 000	0	8/8*1 = 1	
	0 0111 001	0	9/8*1 = 9/8	从上最靠近1
	0 0111 010	0	10/8*1 = 10/8	
	…… ……			
	0 1110 110	7	14/8*128 = 224	
	0 1110 111	7	15/8*128 = 240	最大的规格化数
	0 1111 000	n/a	inf	

取值分布

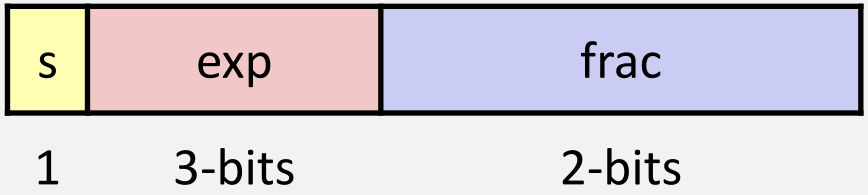
- 6 位浮点数表示
  - e = 3 3位阶码
  - f = 2 2位尾数
  - Bias 3



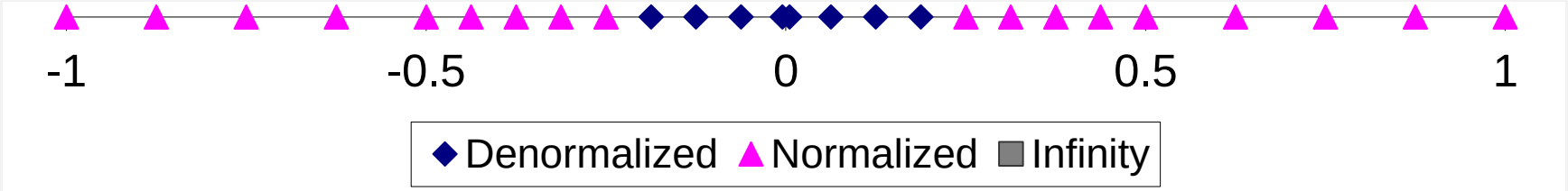


取值分布
（局部放大）

- 6 位浮点数表示
  - e = 3 3位阶码
  - f = 2 2位尾数
  - Bias 3



1 000 00	-0	+0	0 000 00
1 000 01	-1/16	1/16	0 000 01
1 000 10	-2/16	2/16	0 000 10
1 000 11	-3/16	3/16	0 000 11



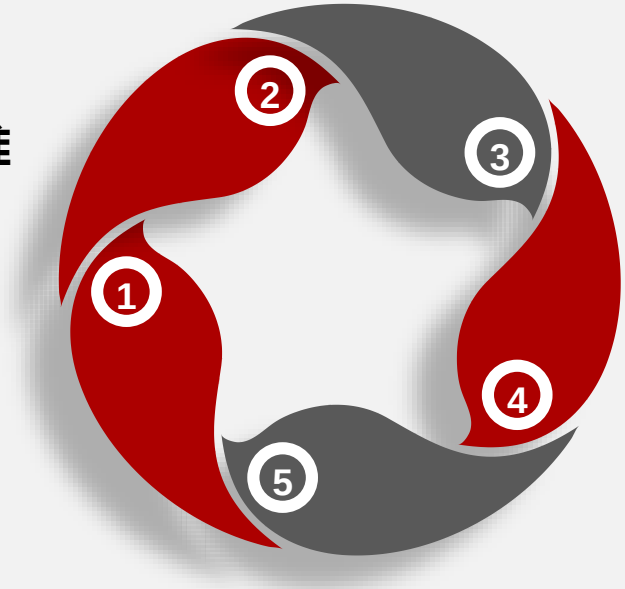
浮点数属性

描述	阶码	尾数	数值
0	00...0	00...00	0.0
最小的非规格化正数 Single ≈ 1.4×10^{-45} Double ≈ 4.9×10^{-324}	00...0	00...01	$2^{-\{23,52\}} \times 2^{-\{126,1022\}}$
最大的非规格化正数 Single ≈ 1.18×10^{-38} Double ≈ 2.2×10^{-308}	00...00	11...11	$(1.0 - \varepsilon) \times 2^{-\{126,1022\}}$
最小的规格化正数 正好比最大的非规格化正数大1	00...01	00...00	$1.0 \times 2^{-\{126,1022\}}$
1	01...11	00...00	1.0
最大的规格化正数 Single ≈ 3.4×10^{38} Double ≈ 1.8×10^{308}	11...10	11...11	$(2.0 - \varepsilon) \times 2^{\{127,1023\}}$

内容提要

IEEE 浮点数标准

示例与性质



二进制小数

舍入与运算

C语言中的浮点数

浮点运算思路

- $x +_f y = Round(x + y)$
- $x \times_f y = Round(x \times y)$
- 基本思路
  - 首先计算精确结果
  - 然后通过“舍入”来得到近似结果

浮点数的舍入

- 舍入方式( )

	$1.40	$1.60	$1.50	$2.50	–$1.50
Towards zero	$1	$1	$1	$2	–$1
Round down (−∞)	$1	$1	$1	$2	–$2
Round up (+∞)	$2	$2	$2	$3	–$1
Nearest Even (默认)	$1	$2	$2	$2	–$2

- 各种模式的优点是什么?

  向偶数舍入能找到最接近的匹配值；其它三种用于计算上界和下界。

向偶数舍入

- 缺省的舍入方案
  - 其他的方案都会产生统计偏差
- 也可以舍入到其他数位
  - 中间值舍入到偶数
  - 例如：舍入到百分位
    - 1.2349999 1.23 (Less than half way)
    - 1.2350001 1.24 (Greater than half way)
    - 1.2350000 1.24 (Half way—round up)
    - 1.2450000 1.24 (Half way—round down)

二进制舍入

- 二进制数舍入
  - “偶数” 是指 0
  - “中间值” 是指舍入位的右边正好是 $100\ldots_2$的形式
- 例如：

  舍入到 1/4 (小数点右边两位)

Value	Binary Rounded	Rounded Value	Action	
2 3/32	10.00011_2	10.00_2	(<1/2—down)	2
2 3/16	10.00110_2	10.01_2	(>1/2—up)	2 1/4
2 7/8	10.11100_2	11.00_2	(1/2—up)	3
2 5/8	10.10100_2	10.10_2	(1/2—down)	2 1/2

浮点数乘法

$(-1)^{s1}\ M1\ \ 2^{E1}\ \ x\ \ (-1)^{s2}\ M2\ \ 2^{E2}$

- 精确结果: $(-1)^{s}\ M\ \ 2^{E}$
  - 符号位 s: s1 ^ s2
  - 尾数M: M1 x M2
  - 阶码 E: E1 + E2
- 调整
  - 如果M ≥ 2, M 右移一位, E = E+1
  - 如果 E超出表示范围，溢出
  - 将M 舍入到 frac 的位数范围

浮点数加法

- ① 对阶，小阶向大阶对齐
- ② 尾数进行加法运算
- ③ 结果规格化并进行舍入处理
- ④ 判断溢出

浮点数加法

- ① 对阶，小阶向大阶对齐

- 两个浮点数进行加减运算时，首先要使两个数的阶码相同，即小数点的位置对齐。若两个数的阶码相同，表示小数点的位置是对齐的，就可以对尾数进行加减运算。反之，若两个数的阶码不相同，表示小数点的位置没有对齐，此时必须使两个数的阶码相同，这个过程称为对阶。
- 将原来阶码小的数的尾数右移|△E|位,其阶码值加上|△E|,即每右移一次尾数要使阶码加1,则该浮点数的值不变(但精度变差了)

浮点数加法

- ② 尾数进行加法运算

- 实现尾数的加运算,对两个完成对阶后的浮点数执行求和操作。

浮点数加法

- ③ 结果规格化并进行舍入处理
  - 如果尾数不是规格化数，则需要进行规格化处理，并进行舍入。
- ④ 判断溢出
  - 根据阶码来判断是否溢出

浮点数加法的数学特性

- 与阿贝尔群比较
  - 有交换性
  - 没有结合性（由于舍入）
  - 3.14+1e10 - 1e10 = 0 vs. 3.14+(1e10 - 1e10) = 3.14
- 单调性
  - a ≥ b ⇒ a+c ≥ b+c?
    - Except for infinities & NaNs

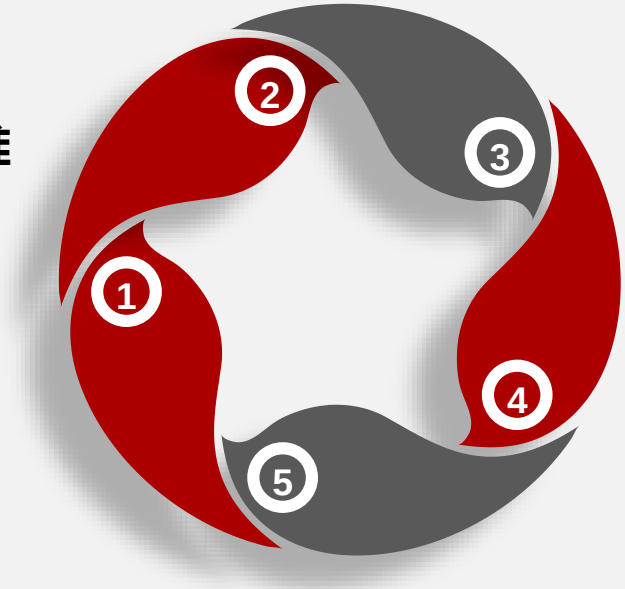
浮点数乘法的数学特性

- 可交换性
  - a*b = b*a
- 不可结合性
  - a*b*c ≠ a*(b*c)
- 不具备分配性
  - a*(b+c) ≠ a*b + a*c
- 单调性
  - a ≥ b & c ≥ 0 ⇒ a * c ≥ b *c （Except for infinities & NaNs）

内容提要

IEEE 浮点数标准

示例与性质



二进制小数

舍入与运算

C语言中的浮点数

C语言中的浮点数

- C 提供了两种浮点数表示方式
  - float　　single precision
  - double　　double precision

- 转换
  - 在 int, float, and double 的转换过程中位级表示会改变
  - double/float → int
    - 值向零舍入，对于无法表示或超出范围的值没有进行定义
  - int → double
    - 能够保留精确值
  - int → float
    - 数字不会溢出，但可能被舍入

浮点数谜题

- For each of the following C expressions, either:
  - Argue that it is true for all argument values
  - Explain why not true

```
int x = ...;

float f = ...;

double d = ...;
```

Assume neither `d` nor `f` is NaN

- x == (int)(float) x
- x == (int)(double) x
- f == (float)(double) f
- d == (float) d
- f == -(-f);
- 2/3 == 2/3.0
- d < 0.0 ⇒ ((d*2) < 0.0)
- d > f ⇒ -f > -d
- d * d >= 0.0
- (d+f)-d == f

浮点数总结

- IEEE浮点数采用 $M \times 2^E$的形式表示（近似表示）
- 提供了表示一些特殊值（正负无穷，NaN）的方法
- 只有有限的范围和精度
- 不遵守普遍的算术属性（例如结合性）

下一节：
程序的机器级表示：基础

湖南大学

《计算机系统》课程教学组