《计算机系统》理论课—浮点数

课堂要点

本次课堂教学主要内容为：**二进制小数、IEEE754 浮点数（包括其格式，三种分类及特点等）、浮点数简单示例及其相关性质、浮点数的运算及舍入原则、C 语言中的浮点数**。通过介绍计算机中的浮点数据类型，让学生了解和掌握浮点数的特性及运算规则。

1. 二进制小数与十进制小数一样，都是利用小数点后的的负数幂来表示分数或者小数部分。从小数点往右，依次是 1/2 位，1/4 位，1/8 位……（请参照十进制的 1/10 位、1/100 位、1/1000 位来理解）。对于一个二进制小数，其小数点后有 k 位，则表示的分数的分母为 2^k，分子为小数部分的十进制值。

例如：1010.1011，整数部分为 10，小数部分为 11/16，因此该数的是十进制为：10.6875

2. **IEEE754 浮点数**：数学形式是 $(-1)^S M\ 2^E$。

二进制格式为：

符号　　阶码　　尾数

计算机中的浮点数据类型有：

单精度：　32 位，符号位 1 位；阶码 8 位，尾数 23 位

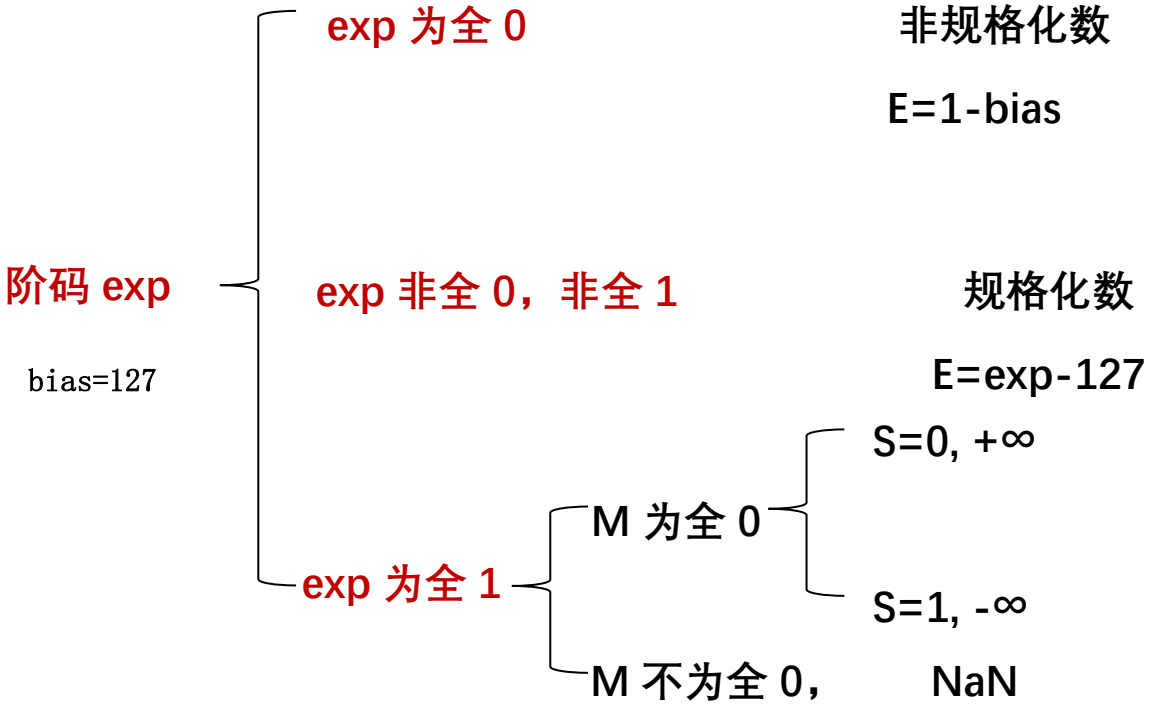
双精度：　64 位，符号位 1 位，阶码 11 位，尾数 52 位

扩展精度：80 位，符号位 1 位，阶码 15 位，尾数 64 位

虽然扩展精度浮点数是 80 位，但是在 32 位机器中占据 12 个字节，在 64 位机器中占据 16 个字节。

3. **浮点数表示方法**：

依据阶码的不同类型，浮点数有三种类别：



将非规格化数（靠近 0 的数）的指数规定为 1-bias，是为了实现非规格化数与规格化数之间的平滑过渡，注意该平滑过渡的间隔就是浮点数所能表示的最小分度，这个与浮点数的位数有关（双精度比单精度毫无疑问具有更小的最小分度）。对于一个固定的阶码，不同尾数表示的各个数之间跨度相同；而阶码变大或者变小，各个数之间的跨度也就相应地变大或者变小，因此越靠近正负无穷，浮点数表示的跨度越大，越靠近 0，跨度越小。

4.**浮点数的运算不同于整数，要将阶码和尾数分别进行运算。**

浮点数乘法：尾数相乘，阶码相加；

浮点加法：1)对阶，2)尾数相加，3)规格化并舍入，4)判断溢出

以课堂上的 8 位浮点数为例，计算 0 0110 110 + 0 0111 010 = ？

要将 0 0110 110 增大实现对阶，注意是 1.110 一起右移！变成 111，阶码加 1，所以对阶后变成：0 0111 111

```
          0 0111     0.111
+         0 0111     1.010
-----------------------------
          0 0111     10.001  （阶码不变，尾数相加）
```

因为此时尾数大于等于 2，所以要：

1. 尾数右移一位变成 1.000，注意因为尾数只有 3 位，所以末尾的 1 被舍掉了（向偶数舍入，1.0001 向 1/8 位舍入，末尾的 1 刚好是个中间值 1/16，因

此向 1/8 位为 0 的偶数 1.000 舍入)。

2. 阶码因为尾数的右移要加 1，变成 1000

最后结果的浮点表示是：0 1000 000

阶码为 1000，bias = 7， 因此 E = 1

结果为： $2^1 * 1.000 = 2.0 = 32/16$

注意丢弃的 0.0001 在这里乘以 2 等于 2/16

而我们在表格中实际查询 0 0110 110 和 0 0111 010 的值分别为 14/16 和 16/20，相加等于 34/16，刚好相差 2/16，就是因为舍弃最后一位 1 造成的。

5.浮点数运算具备可交换性和单调性，不具备结合性和分配性

其本质原因是浮点数每运算一次都会进行规格化和舍入，加上本身就是舍弃有效数字（精度）换取更大的数值表示范围。

6.C 语言中整数与浮点可以进行强制转换，但是一定要注意因为格式的转换导致的精度差异!。