《计算机系统设计》平时作业 2

计算机科学与技术 2110939 李颖

1. 浮点数有没有移位操作和扩展操作？为什么？

因为浮点数的存储和整型数据不一样。

整型只有符号位和数值位，移位一般等价于乘以 2^n 或除以 2^n；浮点数由数符、阶码、尾数三部分组成，移位操作会使阶码和尾数的部分发生移动，导致结果无意义。

扩展操作同理。当整型数据扩展为宽数据类型时，需要进行符号扩展或零扩展；而 IEEE754 浮点数分为单精度和双精度，阶码和尾数所占的位数不同，因此，从单精度扩展到双精度时，若直接进行符号扩展，则导致阶码与尾数表示错乱，导致结果无意义。

2. 为什么整数除以 0 会发生异常，而浮点数不会？

整数类型的除法是以截断方式进行的，即结果只保留整数部分。当整数除以 0 时，由于除数为 0，计算机无法进行有效的除法运算，导致无法得到一个确定的结果，因此会引发异常，通常是被称为“除零异常”或“被零除错误”。

浮点数除以 0 时，计算机会根据 IEEE 浮点数标准规定的特殊规则，返回相应的特殊值。例如 0.0/0 返回 NaN(不是数字)；1.0/0 返回 Infinity(无穷大)；-1.0/0 返回-Infinity(负无穷大)。因此，当除数为 0 时，浮点数类型的除法规则能够处理这种情况，并返回特殊值。

对于 IEEE 754 格式的浮点数，对于正无穷大和负无穷大有特别的解释：正/负无穷大为全 1 阶码全 0 尾数，表示在数值上大于(或小于)所有有限数。引入无穷大数的目的是在计算过程中出现异常的情况下使得程序能继续进行下去。

3. 为何 IEEE 754 加减运算右规时最多只需一次？

浮点数在进行加减法运算之前，首先要进行对阶，使两个数的阶码相等。因此，进行尾数求和时，可能会出现 1.x…x + 1.x…x=±1x.xxxx 的结果。只有该情况需要右规，而即使是两个最大的尾数相加，小数点前也只有两位，为 11.1…1，因此，在加减运算右规时最多只需要一次。

出现 1.x…x - 1.x…x=±0.0…01xxx 时需要左规，由于可能出现前面有多个 0 的情况，因此左规时可能需要多次。

4. 为什么同一个实数赋值给 float 型变量和 double 型变量，输出的结果会有所不同呢？

float 为单精度浮点数，格式为：符号(1 位)、阶码(8 位)、尾数(23 位)

double 为双精度浮点数，格式为：符号(1 位)、阶码(11 位)、尾数(52 位)

因此，float 型变量和 double 型变量能够表示的数据精度不同。例子中，a=123456.789e4，即 1234567890，转换为二进制，可得 1001001100101100000001011010010，用浮点数表示，为 1.00100110010110000000101101001x2^30。尾数为 30 位，超过 float 所能表示的尾数范围，数据截断，发生舍入，尾数部分变为 00100110010110000000101，舍入时在尾数末位+1，变为 00100110010110000000110。最终浮点数的表示变为 1.00100110010110000000110x2^30，重新转换为导致结果不准确。将丢失尾数的二进制数重新转为十进制，得到 1234567936，打印结果反而比原来的大。

对于 double 型变量，由于 30 位的尾数小于双精度浮点型所能表示的尾数范围，因此不会发生截断，数据精度得到保留，最终打印的数据和原来一致。