2017011620 计 73 李家昊

1.假设存在一种 7 位浮点数（符合 IEEE 浮点数标准），1 个符号位，3 个阶码位，3 位尾数。其数值被表示为 $V = (-1)^S \times M \times 2^E$。请在下表中填空（Binary: 这一列请填入 7 位二进制表示；M: 这一列为十进制尾数；E: 用整数表示；Value: 被表示的具体数值，十进制表示。“—”表示无需填入）。

描述	Binary	M	E	Value
负 0	1000000	0	-2	-0.0
正无穷	0111000	—	—	+∞
—	0110110	7/4	3	14.0
最小的大于零的数	0000001	1/8	-2	1/32
1	0011000	1	0	1.0

2、已知某 32 位整数 X，其值为-102（十进制），则其 16 进制补码为 0xFFFFFF9A ，另一 32 位整数 Y 的补码为 0xFFFFFF6A，则 X+Y 的 16 进制补码(32 位)为 0xFFFFFF04 ，X-Y 的 16 进制补码为 0x00000030 。

3、计算机中表示带符号整数的编码方式是补码，补码的一个性质是：将某个数的补码表示按位取反再加 1，就可以得到该数的相反数的补码表示。试简单证明之。

假设存在另一种带符号整数的编码方式：最高位只用于表示该数值

的符号，后续数位只表示数值本身（如同无符号数的表示），请比较一下这一种编码方式与补码编码方式 （从相同位宽下能够表示的值的范围、以及完成加减法运算的方式等方面入手）。

证明：记 $\overline{x}$ 为 x 按位取反得到的数，则有 $x+\overline{x}=111\cdots111_2$，即 $x+\overline{x}+1=0$，移项得 $-x=\overline{x}+1$。证明完毕。

这种编码方式即原码表示。下面比较原码和补码。

设位宽相同为 n，则原码表示范围：$[-2^{n-1}+1,2^{n-1}-1]$，补码表示范围：$[-2^{n-1},2^{n-1}-1]$，原因是原码中的0有000…000和111…111两种表示方法。对于加法，原码的操作十分复杂，需要根据符号位判断数值部分是加法还是减法，同号求和，异号求差，而补码只需要直接相加即可。减法同理。

4、判断是否成立，如不成立请给出反例或说明：

已知int x = …; float f = …; double d = …; f与d都不是NaN

- x == (int)(float) x　　不成立，反例：x=0x0FFFFFFF时，等式左边=268435455，等式右边=268435456。
- x == (int)(double) x　　成立。
- f == (float)(double) f　成立。
- d == (float) d　　不成立，反例：d=0.3时，等式左边=0.300000011920928955078125，等式右边=0.299999999999999988897769753748434595763683319091796875

- f == -(-f);　成立。
- 2/3 == 2/3.0　不成立，左边为0，右边为0.6666…
- d < 0.0 -》((d*2) < 0.0)　成立
- d > f　-》 -f > -d　成立
- d * d >= 0.0　成立
- (d+f)-d == f　不成立，反例：d = 1e100, f = 1e-8时，等式左边=0，右边=1e-8

P.S.以上所有反例均在 Ubuntu 16.04 g++ 5.4.0 环境下测试得到