Ics\_Lab1实验报告

姓名：谢志康

Userid：kurumi

学号：22307110187

完成于：2023.10.7

以下是我写每一题的主要逻辑部分——

整数部分1~15题：

1.要求输出111……00000，输入31：000……11111取反即可。

2.De Morgan’s Law ~(a & b) = (~a)|(~b)

3.判断一个数是否大于0我注意到非负数右移31位为0（最左边一个符号位为0且全部补0），负数右移31位为-1（最左边一个符号位为1，向右移位左边补1，最后11111……，因此会输出-1），判断x + y与0的大小比较即可。（总共32位）

4.C = a % b = a-[a/b]\*b。对于y=2^n有规则x % y = x & (y-1)（注意~0==-1，规定不能用负号）

5.逻辑是异或a ^ b = (a – b) | (b – a) = (a & b) | (b & a)

6.如果x>0🡪x>!x，返回x，vise versa。如果x==0，!x==1，返回x

7.分解题目成子问题：求一个二进制数字最低位的1？假设任意一个二进制数字为xxxxxx……10000……，当取反后：yyyyyy……01111……，此时将其加一，则得到yyyyyy……10000……，再与原来按位与运算，则得到最低位的1。原问题即做两次。

8.逻辑，一个byte上全给1，与x的四个byte要求交换的部分分别与操作，得到，需要交换的两个byte的内容，再按权值减去这个大小，这两个部分权值交换，在按权值加上这两个大小，最后得到答案。

9.分别取出四个byte的内容，判断非0则结果加一

10.x\*7也就是x\*8-x，x\*8也就是x<<3，x/16也就是x>>4，得到答案

11.x>=y也就是x-y>=0，接下来逻辑与第三题比较的逻辑一致。最终有两种情况返回1：符号位相同且x>=y 或 符号位不同且x>=0;y<0。其它情况，则返回0即可。

12.两种溢出情况：正+正=负,负+负=正

13.考虑到最多溢出两位，111……0101，要取到0101—将其与0xff做&操作，分别与x y z如此操作后相加，得到x y z后四位相加的结果，向右移动四位，得到进位结果，x y z分别向右移动四位，左边补0（与000011111……做交），最终相加，右移28位，刚好得到结果（反映左边四位的大小，即溢出的大小，因为最多溢出两位，所以是可行的）

14.难点在于如何把左边补0：1<<n; 000……100……（n个0），取反：111……011……（n个1），加一：11111000……（n个0），取反：00001111……（n个1）向左移动32-n位

15.逻辑简单，操作麻烦。逻辑上：将【l，h】位全部搞成1，其它全为0，再和0xAAAAAAAA，也就是偶数位为1奇数位为0的数做&操作就能得到答案。但是操作上有点麻烦：有一个验证是【0，31】，当左移32位时是未定义操作，（理论上是随机产生0和1的）但发现它实际上写x<<31;x<<1它确实只返回1，于是利用这点写对了。。。

浮点数部分16~19题：

仿照老师上课讲的计算机二进制表示原理，可以取一定长数（类似计算机32位这种），中间一个取为小数点（以随便一个constant表示），最左边一个可以设为符号数（sign，0正1负）（定点表示：一种常见的方法是使用定点表示来模拟浮点数。你可以将整数的一部分用来表示整数部分，而将另一部分用来表示小数部分。例如，你可以选择一个固定的小数点位置，然后根据这个位置来表示小数部分，后面还可以规定多少位表示指数部分。）

16.单精度浮点数标准：最左边一位是符号位0x80000000，接下来往右八位是指数位0x7f800000，接下来是尾数。关注几个特殊的浮点数：指数全1，尾数全0表示无穷大，即inf。指数全1，尾数不为全0表示NaN，即Not a Number，不是数。主要逻辑：分别取出符号位，指数位和余数位，如果指数位全为1且小数位不为0，则NaN，输出uf，否则将符号位与0111……做交，将符号位置0（取绝对值），输出即可。

17.逻辑：先特判是否为NaN，即指数全为1且余数不全为0，其次，

特判正零和负零的情况。之后正常比较大小，当两个数符号不同的时候，正数肯定大；当两个数符号相同的时候，如果uf1的数字部分大于uf2的数字部分且符号为正，返回1，否则0；如果uf1的数字部分小于uf2的数字部分且符号为负，返回0，否则1。

18.在浮点数标准中：sign占一位，接下来exponent占八位，e1……e8，接下来remnant占23位，r1……r23。整个浮点数的绝对值大小就是：（exp不全为0时，全为0时是0.r1r2……）f =（1.r1r2……r23）\* 2^e1……e8-127。要求f\*（2^n），也就是（1.r1r2……r23）\* 2^e1……e8-127\*2^n = （1.r1r2……r23）\* 2^(e1……e8+n-127)。逻辑就是，将指数部分取出来，加n，判断是不是nan即可。注意的就是先特判正0和负0。其次，特判指数为0的情况，符号位不变，余数\*（2^n）再加上符号位即可，注意要处理余数溢出到指数位的情况。最后判余数会溢出的情况。

19.首先特判特殊情况。其次，提取x的符号，因为x是int类型的数据，直接将x取为绝对值x进行接下来的操作，找到x中位数为1的最高位，将其赋值给exp并加上偏移量127。（逻辑上：int类型的数可以表示为2^a1+2^a2+……+2^an，令a1是最大的那位，将2^a1提出来，则exp赋值为a1，剩余部分变为1+2^(a2-a1)+……+2^(an-a1)，也就是形如1.00a2000an这样子，也就是float里面的余数部分了。其次是最麻烦的部分——处理溢出：首先exp-23得到余数后得有多少位，1左移（exp-24）就表示余数能有的最大数，当余数大于这个最大数的时候，余数溢出没法表示，就要处理溢出。其次，进行四舍五入的操作：如果num大于等于0x3fffffff，这就表示尾数部分的小数部分非常接近或已经超过了0.5，因此需要将尾数部分向上舍入，并且指数部分加一。

荣誉题第20~22题：

20.（最大使用数字只有0xff不知道咋做到34ops，做掩码都快用完了额啊）逻辑：用五个掩码操作，统计1的个数——第一个形如0101……的掩码，如果x的一段是00🡪00（表示0个）, 11🡪10（表示2个）, 01🡪01, 10🡪01（都表示1个）。第二层掩码形如0011……，经过第一层的处理，每两位只可能是00，01，10 三种情况中的一个：若是0001或0100（一个1），则处理后变成0001（1）；若是0010或1000（两个1），处理后变成0010（2）；若是0110或1001（三个1），处理后变成0011（3）……后面三次同理，每次处理都相当于折叠一次，最后得到的数字就是1的个数。

21.和上一题计数类似：先做出五个掩码，再逐步分治操作。每四个bit一组，a1a2……a8，1🡪a5a6a7a8a1a2a3a4；2🡪a7a8a5a6a3a4a1a2；3🡪a8a7a6a5a4a3a2a1；之后将每四位翻转，example：a1是b1b2b3b4，则4🡪b3b4b1b2；5🡪b4b3b2b1。五步操作解决。但ops还是炸了（）

22.思路简单，逻辑处理麻烦（如何不用循环和if实现？）每三位都是一个周期，第一位mod7是1，第二位mod7是2，第三位mod7是3。（如0b10000是0b111左移1加上2，余数就为2）利用这点，我们做一个六位的掩码（之所以是六位是为了控制ops，以至于后面count会偏大……），后面count是每个三位除以7的余数相加，一般会比7大（但经过我的测试发现大不过500），于是可以不用循环，直接三次取余数相加操作得到小于等于7的结果，最后得特判==7的情况，==7时count赋值为0，否则还是count。最后根据符号返回即可。



