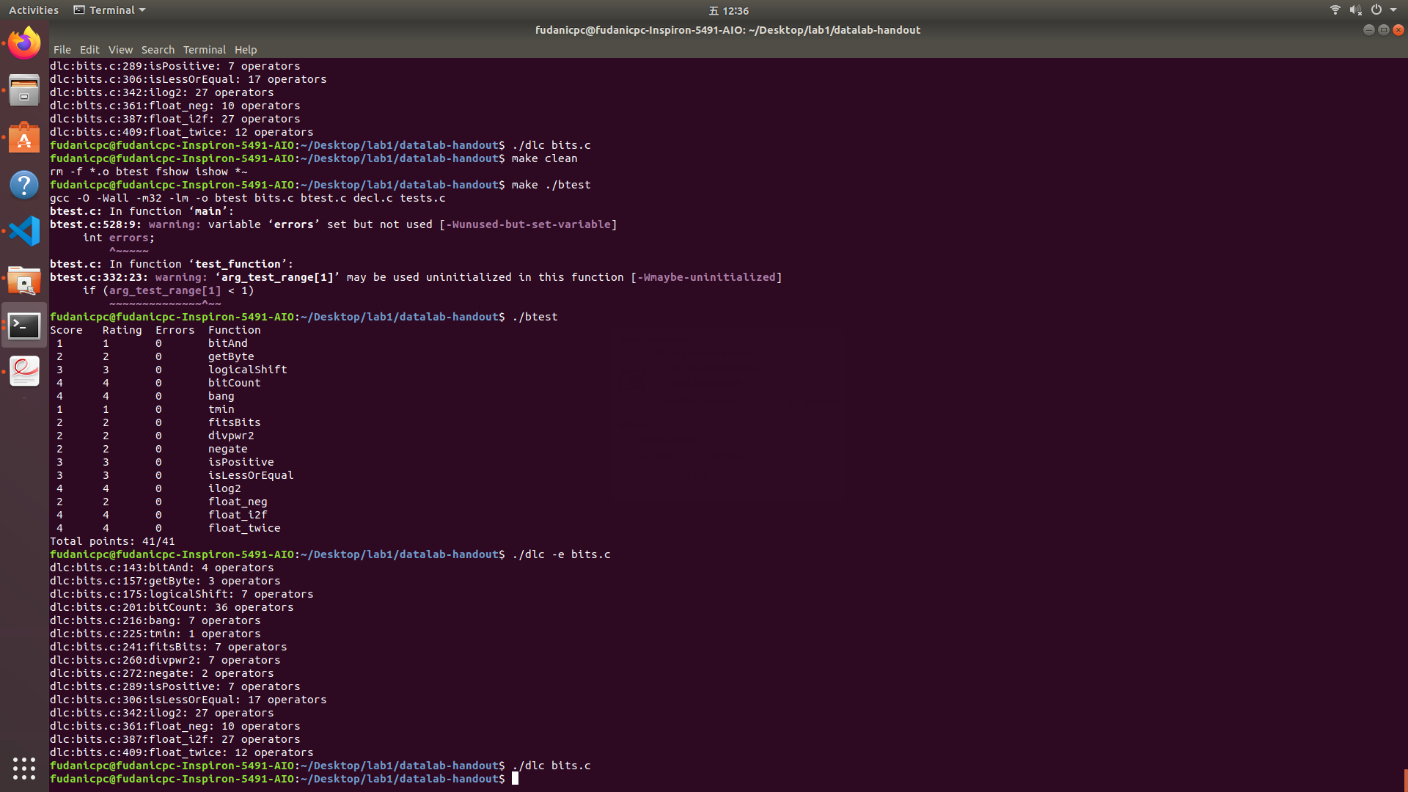
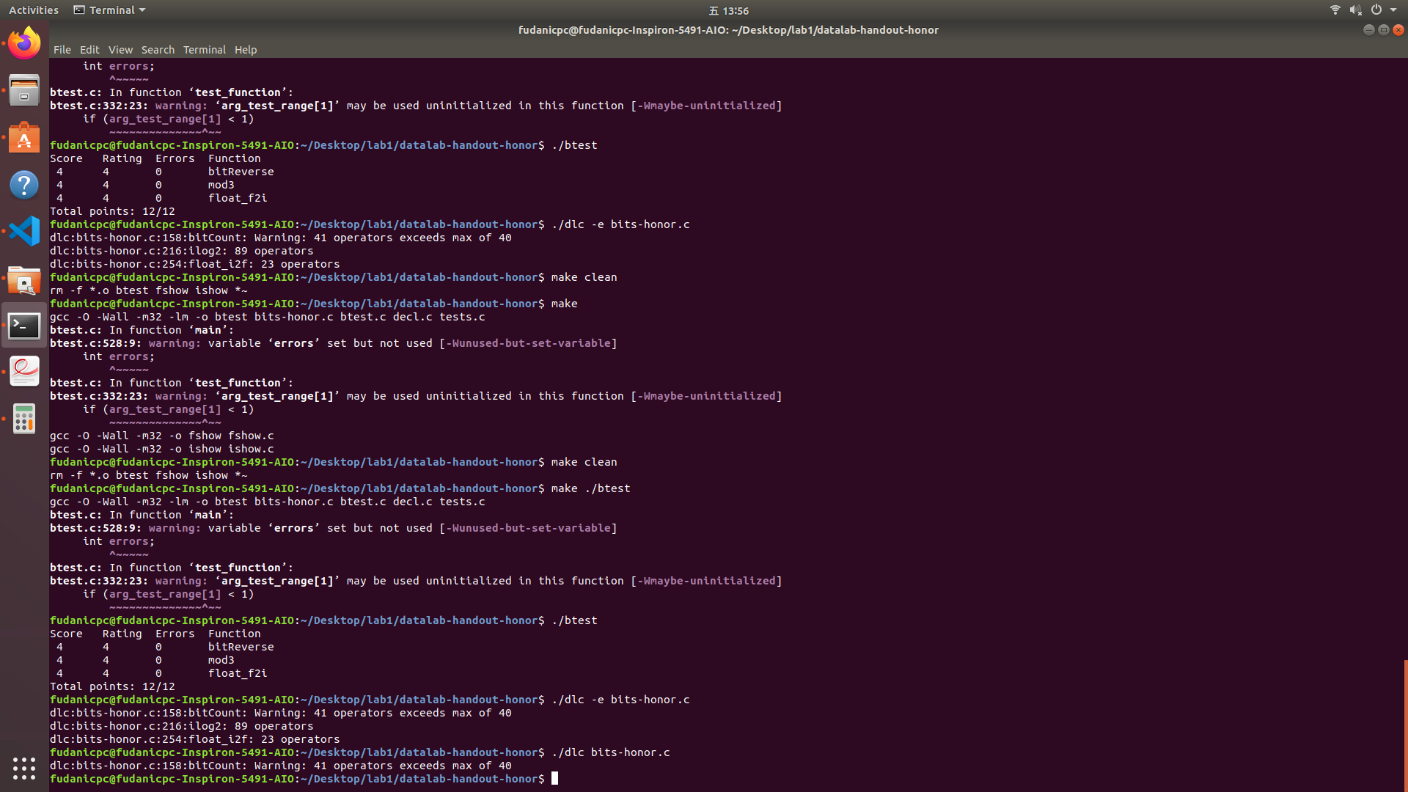
“计算机系统基础（上）Lab1”报告

计算机科学与技术

19307130296

孙若诗

1. 运行截图



注：中未满足操作数限制。

1. 程序解析

描述：用~和|实现&。

操作：4/8

解答：观察真值表，发现取反后或的结果刚好与&相反。

描述：取出x从低位起的第n个byte。

操作：3/6

解答：一个byte为8个bit，所以先将n左移3位，得到x需要右移的位数。x右移后与255，即8个1，即可得到第n个byte上的数。

描述：逻辑右移。

操作：7/20

解答：移出来的前n位必须为0，所以求一个只有后32-n位为1的数与上即可（似乎这种工具数被称为掩码）。可以将31个1右移n-1位，但是n为0时右移-1位会出问题，先右移n位再左移1位可以避开这个坑。

备注：之前一直以为逻辑右移需要把负数补码中前导1都消去，做这道题时才意识到只需要处理移出的空位，至于原先就存在的前导1不必在意。

描述：数int中二进制1的个数。

操作：36/40

解答：采用分治算法，先每相邻2位1的个数相加，再每相邻4位，直到前16位和后16位相加得到全部1的个数。关于取每相邻n位的操作还是用掩码，例如取相邻的1位使用010101……0101，2位使用00110011……0011，以此类推。

描述：实现!x。

操作：7/12

解答：先用~x+1求出相反数，非0数和它的相反数之间至少有一个最高位为1，取或之后把最高位移下来，然后-1变+1（也就是再取一次相反数），再异或1即可。

备注：注意到-2147483648的相反数在int范围内无法表示，~x+1仍旧是它本身，但是所幸它最高位本来就为1，并不影响结果。

描述：最小的2补码。

操作：1/4

解答：1<<31。

描述：查询x是否能用n位二补码表示。

操作：7/15

解答：注意到int本身就是补码形式，所以n位够用代表32-n位都是多余的，左移32-n位再移回来，如果数不变说明可以。如果能用n位表示，左移时失去的和右移时补回来的都是符号位上的数，所以结果不变。

备注：有了相反数操作就很容易实现减法了。但是-2147483648仍不能实现。^操作可替代==。

描述：x/(2^n)，靠近0取整。

操作：7/15

解答：直接>>n位，负数会出问题，这是因为负数的默认向下取整相当于远离0取整。我们在正数向上取整的时候经常用到一个操作，（a+b-1)/b相当于a/b向上取整，因为如果刚好整除，没有影响；只要有余数，都能向上进一位。这里我们就是相当于除数为负时改成向上取整，所以要给负数加2^n-1。

描述：求相反数。

操作：2/5

解答：~x+1。

描述：判断是否为正数。

操作：7/8

解答：只有0是符号位为0的非正数，特判一下即可，其他数只需要看符号位是否为0。

描述：x是否小于等于y。

操作：17/24

解答：直观的想法是判x-y是否为小于等于0，但是x为-inf，y为inf时可能会溢出，因此先判一下它们是否异号，异号则可以直接判断大小。

描述：log以2为底x，向下取整，x>0。

操作：27/90

解答：显然答案在0~31之间，显然任何0~31的整数都可以由16、8、4、2、1从大到小拼凑得到，并且如果当前数为x，可加数为y，目标数为z，x+y<=z时给x加上y一定不会出错。因此按从高到低顺序检查它是否大于等于2的这些幂次，是则累加这个指数，并减去这个幂次检查下一个即可。

描述：给一个浮点数取相反数。

操作：10/10

解答：nan的定义为指数位全为1，分数值非0。除去nan，余下的数符号位取反即可。

备注：可以发现inf也是分正负的。

描述：将int转换为IEEE标准下的float。

操作：27/30

解答：模拟题，首先把符号位取出来。然后看一看绝对值一共有多少位。左移到最高位，右移8位作为分数值，后面的8位要被舍去。如果最后一位为1或者后8位不为空，需要进1。指数需要加127，最后结果把符号位、指数位和分数值依次拼起来即可。

备注：-2147483648和0是特殊情况，-2147482648在int内无法取绝对值，0没有1，因此特判掉这两个数。

描述：给一个浮点数乘2。

操作：12/30

解答：分别取出指数位和分数值，若指数位为0，给分数值左移1，加上符号位。若指数位不为0也未满，直接给指数值+1。

备注：验证了课上所讲的原理，若分数值超出23位，多出的一位恰好加给指数位，无需特别处理。

描述：左右翻转二进制位。

操作：41/40

解答：类似bitcount，依次交换相邻2位、4位……32位。

备注：以上思路需要41次操作，并未达到题目要求。

描述：模3。

操作：89/90

解答：观察易知2的偶数幂模3均为1，2的奇数幂模3均为2，因此类似bitcount地分治统计有多少个偶数幂和奇数幂，可以把被模数降到16\*1+15\*2=46及以下规模，这时只剩下6位，可以每两位&3相加，得到的结果仍然模3同余。如此迭代下去，3次后只剩下2位，只要判断一下剩下的是否恰好为3，是则减去3即可。

备注：原本对后期一位位右移统计，只能卡到103次操作，每两位一处理的思路得到了高庆麾同学的指导。关于符号的讨论详见总结。

描述：将float转化为int。

操作：23/30

解答：取出符号位、指数位和分数值。分数值需要左移指数位、右移23位，因此先用指数值减去127再减23，和分数值位数比较，若超过int范围就直接返回结果（-inf或者0），确定未超过限制再移动分数值。

备注：int右移超过位数会错误，并不是因为前面补0就没有影响。

1. 总结
2. 质疑mod3的check程序是否有负数情况，程序完全未处理符号位时也通过了btest测试。最后手动调整了负数的偏差，负数时结果-1，因为-2147483648%3=-2=1，但当作第32位处理，偶数位的贡献为2，多了1。但-2时输出为1，在模运算下等价。

题目描述中的mod3(-100)=-1本身也不太妥当，一般数学定义的整数模运算结果应为[0,n-1]区间内整数，若出现负数被模也应当加到正数域内，因此mod3(-100)=2似乎更规范。但是使用C++测试，（-100）%3确实为-1，可能是绝对值取模后再加符号。

1. 第一次写C语言程序，了解到函数要先定义所有变量，不能中途定义的规则。
2. Linux系统的programming mode calculator，Windows系统的程序员计算器都有较为直观的数制转换、同步显示功能，对于这次lab有很大帮助。
3. 熟悉了一些常用的替代关系：(~0)=1，^可替代==，|对bool型可替代+，~x+1可替代-，|可替代||，&可替代&&，x为0或1时&xx……x可部分替代if(x)，！！可以将一个int转为bool。
4. 认识到了自己对于计算机系统的无知。完成lab过程中虚拟机无法开机，尝试了网络上搜索到的netsh winsock reset、关闭3D优化等各种方法，均未奏效。解决方法似乎只剩下重装，深感困扰，又暂时不愿意陷入安装双系统的麻烦。最终还是在一台主系统为Ubuntu的机器上完成了这次lab。
5. 参考文献
6. <https://zh.wikipedia.org/wiki/NaN>
7. <https://zh.wikipedia.org/wiki/IEEE_754>
8. https://zh.wikipedia.org/wiki/模除
9. https://zh.wikipedia.org/wiki/C语言