**深圳大学实验报告**

**课程名称： 计算机系统(2)**

**实验项目名称： 数据表示实验**

**学院： 计算机与软件学院**

**专业： 软件工程**

**指导教师： 罗胜**

**报告人： 谢弘烨 学号： 2020151036 班级： 软工02**

**实验时间： 2022 年 05 月 04 日**

**实验报告提交时间： 2022 年 05 月 04 日**

**教务处制**

|  |
| --- |
| **实验目的与要求：**   1. 了解各种数据类型在计算机中的表示方法 2. 掌握C语言数据类型的位级表示及操作 |
| **方法、步骤：**   1. 安装gcc-multilib：     或者：    2、根据bits.c中的要求补全以下的函数：  intbitXor(int x, int y);  inttmin(void);  intisTmax(int x);  ntallOddBits(int x);  int negate(int x);  intisAsciiDigit(int x);  int conditional(int x, int y, int z);  intisLessOrEqual(int x, int y);  intlogicalNeg(int x);  inthowManyBits(int x);  unsignedfloat\_twice(unsigned uf);  unsigned float\_i2f(int x);  int float\_f2i(unsigned uf);  3、在Linux下测试以上函数是否正确，指令如下：  \*编译：./dlcbits.c  \*测试：makebtest  ./btest |
| **实验过程及内容：**   1. bitXor：   由异或操作真值表可知，当且仅当x和y不相同时结果为1。同时由离散数学知识可知，可以利用按位非和按位与操作代替按位或。因此函数体为：     1. tMin：   易得最小的二进制补码为0x8000 0000，且将1左移31位即可得出该结果：     1. isTmax：   易得最大的二进制补码为0x7fff ffff，同时其加1后为0x8000 0000，与0x0同为加上自身结果为0的数。  若要实现要求，仅需对上述过程结果取反同时排除0xffff ffff的干扰。     1. allOddBits：   二进制数中有且只有0x5555 5555可以与奇数位上均为1的数按位或得到0xffff ffff。因此利用移位运算得到0x5555 5555后与x按位或，按位反后再逻辑反即可实现要求。     1. negate：   简单按位取反再加1即可求出x的相反数。     1. isAsciiDigit：   将x与范围两个边界相减，再右移31位。由于边界值同样可取，0x39为了统一操作需要将边界扩大至0x3a。     1. conditional：   先将x用连续两个逻辑反布尔化，x非0时为1，x为0时为0。再通过先左移31位后右移31位的操作使得每一个bit位上都有值。最后分别与y和z按位与，其中一个结果必为0，两个结果再按位或即可实现要求。     1. isLessOrEqual：   对于本题，直接想法一般是直接相减，然而容易出现溢出。  正确做法应为先判断x和y是否同符号。不是则可直接得出结果；是则两者进行相减，判断差的结果是否满足。     1. logicalNeg：   本题中只有0的逻辑反为1，其余任何数字的逻辑反都为0。容易发现有且仅有0的取反为0xffff ffff，所有位上均为1。  那么对于任何x，要想确定x为0，仅需用|运算统计其所有位置上的数字，最终结果仍为0的话即可说明x是0。然而遍历各位统计需要的运算符数量超限，可以使用二分法减少运算符用量。     1. howMangBits：   本题较为复杂。  对于给定的数字x，需要先用异或运算将其符号位处理为0以便于后续处理。  接着利用二分法。首先将x右移16位来判断前16位是否为0，是则右移重新右移8位来判断前24位是否为0，否则舍弃原x后16位，对于剩下的16位进行右移8位判断的操作。以此类推。  具体实现如下：     1. float\_twice：   本题要将浮点数进行翻倍。  首先将浮点数的各个部分分割开来，分别处理exp全为1，全为0，不全为0三种情况：全为1直接返回；全为0要将frac段左移1位，即乘以2；不全为0时直接将阶码段加1，若阶码段变成全为1则抛弃frac段，否则将三部分合并后返回。     1. float\_i2f：   本题效果为将传入的整型转成浮点数返回。  对于传入的x。  先统计符号，取其绝对值。  再通过循环找到1第一次出现的位置，统计循环的次数来计算阶码。  最后处理尾数段。经过上述处理后要得到尾数段，就需要将上述循环结果右移32-23=9位，再通过简单判断最后几位的数据以确定最后结果是否需要进位。     1. float\_f2i：   本题效果为将传入的浮点数转换成整型。  将传入的浮点数分割出符号位，阶码段，尾数段。对于超出范围以及非规格化表示进行特殊处理。对于规格化表示，由阶码段求出阶码，在尾数段前加1后右移阶码位，最后根据符号位加上正负。     1. 正确性验证 |
| **实验结论：**   1. **位操作比高级语言难得多**   相对于高级语言，通过使用移位取反等位操作对数值进行操作要比高级语言进行运算难的多。而且在运算过程中需要注意更多的细节。位运算也需要对底层逻辑更清楚更明确。在实验过程中，通过上网查阅资料了解不熟悉的细节，我最终顺利完成了实验。   1. **数学技巧可以起到另辟蹊径的作用**   在本次实验中，例如第一题等，可以利用离散数学中的真值表技巧，列出真值表对函数进行求解，这也是一种好方法。   1. **不能忽略特判**   对于后三个题中，需要特殊注意特判的情况，需要区别全0情况和全1情况进行分开讨论。对特殊值进行特判后不仅能提高代码的容错性，更能使程序逻辑更清晰。 |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2018年 月 日 |
| 备注： |