**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： 计算机科学与技术**

**指导教师： 罗秋明**

**报告人：刘俊楠 学号：2017303010 班级： 计科一班**

**实验时间： 2021 年 3 月 20 日**

**实验报告提交时间： 2021 年 3 月 20 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. 安装make与gcc-multilib   执行命令sudo apt-get install make gcc-multilib即可安装make与gcc-multilib。     1. 将datalab-handout文件夹及其内部文件通过pscp传输至Ubuntu的Linux系统。  1. 补全bits.c中的函数   2.1 int bitXor(int x, int y)  x^y = (~x&y) | (x&~y) = ~(~(~x&y)&~(x&~y))。    2.2 int tmin(void)  最小的二进制补码为10…00 ，因为是32位系统，所以将1左移31位。    2.3 int isTmax(int x)  若x为最大值，则其应为0111…111（31个1），所以将其左移一位再加2应该为0，在判断符号位是否为零，将x右移31位，若为零则其符号位为0。    2.4 int allOddBits(int x)  先获得1010101010…1010101，然后用x与其进行 | 操作，若x为所求，则会得到11111111…11，最后按位取反再取否。    2.5 int negate(int x)  求补码，按位取反再加一即可。    2.6 int isAsciiDigit(int x)  0x30=48,0x39=57,将x减去48再加一、x减去58，若前者为符号位0，后者符号位为1，则得到目标数。    2.7 int conditional(int x, int y, int z)  实现x?y:z，即当x为非0数时返回y，当x为0时返回z，可推出公式(a&y)|(b&z)，当a为!x+~1+1,b为(!x)+~1,此时a和b满足上述条件。    2.8 int isLessOrEqual(int x, int y)  若x与y异号且x的符号位为1，则x一定小于y，返回1；若x与y同号，则使用y减去x，判断结果的符号位，若为0说明y大于等于x，返回1。    2.9 int logicalNeg(int x)  判断~x+1与x的符号位，若都为0则x为0，（（~x&~（~x+1））>>31）&1返回为1；其他非零情况返回都为0。    2.10 int howManyBits(int x)  对于正数，位数=最高位1所在位数+1.对于负数，位数=最高位0所在位数+1.  对于0，返回1.对于-1，返回1.对于32位非零数，使用二分法，从左往右看，若前16位有1，则右移16位，移后非零数若前八位仍有1，则右移8位 位数+8，反之不动，并以此类推，最后位数等于所有移动位数和在加2。    2.11 unsignedfloat\_twice(unsigned uf)  若指数exp==0xff，则uf溢出，返回本身；  若指数exp==0，则：   1. 若表示小数部分的首位为0，则直接将小数部分左移一位，相当于1.\*\*\*； 2. 若不为0，则将exp加一，再将小数部分左移一位（类似科学计数法）； 3. 其余情况直接将exp加一，若加一之后exp==0xff，则令小数部分为0，否则不做处理。（超量程了 不输出小数位）     2.12 unsigned float\_i2f(int x)  int型（二进制表示）可假想小数点在最低有效位右边，左移shiftnum-1位，使得小数点移动到第一个有效数字（1）的右边，那么小数点右边首位到第23位就是浮点数表示的尾数部分了，因为这里只要取23位所以存在精度问题，这里采取四舍五入的方式，阶码部分看小数点移动的位数，加上偏置（float中为127）成为移码即可。即(127+32-shiftnum)<<23。    2.13 int float\_f2i(unsigned uf)  获取uf的符号位、指数和小数部分，若指数大于158，则直接返回0x80000000，若指数小于127，则直接返回0，其余情况，将小数部分右移（23-指数）（因为小数部分是从23位开始的），再根据获取的符号位判断正负然后返回结果。     1. 在Linux下测试以上函数是否正确   编译后运行(./btest)。 |
|  |
| **心得体会：**  通过本次实验，我学习到了如何在Ubuntu中进行基本的位运算操作，进一步掌握了计算机系统的位运算的要点与格式，对于计算机系统中整型数字和浮点型数字的原理与代码编写有了更加深刻的了解。 |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2018年 月 日 |
| 备注： |