**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： 张欣杰 学号： 2020151091 班级： 软工二班**

**实验时间： 2022 年 04 月 07 日**

**实验报告提交时间： 2022 年 05 月 05 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. int bitXor(int x,int y); 2. Xor运算时异或预算，将X和Y的运算结果逐个列出来，即可获得仅使用~和&的运算表达式；运算表如下：  |  |  |  | | --- | --- | --- | | X | Y | X Xor Y | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 |  1. 通过上述运算表即可得出运算表达式：X Xor Y=~(~(X&~Y)&~(~X&Y)) 2. Int tmin(void) 3. 该题目的是返回最小的二进制补码，对于32位整数，其最小值为0X80000000，即将整数1左移31位即可 4. 核心代码：      1. Int isTmax(int x) 2. 函数功能是判断x是否是有符号数的最大值，也就是补码最大值，还是拿1个字节来看，最大补码数的形式为0111 1111，但为了防止错误，我们需要将-1拿出来单独判断，以排除-1的干扰。 3. 核心代码：      1. Int allOddBits(int x) 2. 本题要求是所有奇数位都为1则返回1，否则返回0，因此我们可以通过构造一个掩码来与x进行异或运算，若满足条件则返回1，不满足条件则返回0 3. 核心代码：      1. Int negate(int x) 2. 题目要求是返回x的负数，因此直接套用公式，补码按位取反然后再加1。 3. 核心代码：      1. Int isAsciiDigit(int x) 2. 题目要求是若x是ascii码中的0-9则返回1，否则返回0；对于每一个x，需要满足0<=x<=9，因此可以对x与临界值做差，通过右移31位进行判断是否在这个区间内。 3. 核心代码：      1. Int conditional(int x, int y int z) 2. 首先先判断x是否为0，若为0，则x赋值为0，若不为0，则x赋值为1，套用公式得到x的补码，用x的补码分别和y、z进行与操作。 3. 核心代码：      1. Int isLessOrEqual(int x, int y) 2. 若x<=y则返回1，否则返回0；因此我们可以根据y-x的结果的符号来判断x和y的大小关系。 3. 核心代码：      1. Int logicalNeg(int x) 2. 可以通过取相反数进行非零判断，令y=~x+1，若x为0时，两者符号位都为0，若x=0x80000000时，两个符号为都为1，若x为其他情况，则两者符号位为10或者01。 3. 核心代码：      1. Int howManyBits(int x) 2. 对于正数，从高位到低位，其位数加上符号位，最少需要n+1位；对于负数，从高位到低位，则需要n位。先定义分别表示0-15的变量，对x取符号位，如果x为正则不变，为负则取反；再从高位开始判断，从高位到低位依次赋值，返回位的值。 3. 核心代码：      1. Unsigned float\_twice(unsigned uf) 2. 将无符号浮点数乘2，对无阶码小数，对其尾部乘2即可，即直接左移一位，但要提前记录符号位。对于规格化数，直接对其阶码+1即可。 3. 核心代码：      1. Unsigned float\_i2f(int x) 2. 返回x的unsigned浮点数的二进制形式；将整形转化为无符号浮点数，即求浮点数。先取的符号位，再将剩余部分全部取为正数形式，即可以得到无符号的数值。然后将有数字的部分直接移动到最高位，记录移动的位数，再将其移动9位（因尾数只要23即可）。对于阶码部分，由于记录的是小数点从31位右数到第一个1，但实际上需要处理的是从第0位到第一位，所以E=32-left，bias为127，加上为159，if部分做舍入处理。 3. 核心代码：      1. int float\_f2i(unsigned uf) 2. 首先假设浮点数为规格化数，则exp=e-bias得到指数部分，我们知道如果exp<0 ，则计算出来的结果一定是小数（包括非规格化数），此时能直接舍入到0；如果exp>31，表示至少要将尾数部分右移31位，此时一定会超过补码的表示范围，所以直接将其溢出。可通过最低23位得到尾数部分。尾数部分需要自己在最高有效位添1，如果是负数，则补码的最高位为1，就要求其对应的无符号编码最高位不为1，否则是负溢出溢出；如果是整数，则补码的最高位为0，就要求其编码的最高位为0，否则是正溢出。 3. 核心代码： |
| **实验结论：**  使用cd datalab-handout指令进入指定文件夹，输入指令make，编译源代码文件，显示编译通过。    输入指令./btest对代码进行运行并测试    可以看到，测试样例全部通过，代码全部正确。 |
| **心得体会：**   1. 通过本次实验，加深了对位运算的理解，与高级语言不同，位运算时比较抽象的，使用位操作进行运算，需要注意更多细节、也需要对底层逻辑有更加清晰的理解。 2. 对于特地的题目，我们需要对诸如0、1、-1等特殊值进行判断，我一开始编写代码的时候没有对这些特殊值进行判断，导致出现错误，查看了错误提示之后得知这些特殊情况需要进行分开讨论。 3. 经过本次实验，我发现自己对位操作、位运算并不是很熟悉，在一些地方经常卡壳，这个实验写的时间也比较长，因此还需要对位操作进行复习，这样才能为后面的学习打下基础。 |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2018年 月 日 |
| 备注： |