**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： 马晨琳**

**报告人： 学号： 班级：**

**实验时间： 2025年4月3日 至 4月16日**

**实验报告提交时间： 2024年 4月 12 日**

**教务处制**

|  |
| --- |
| **一、实验目的：**   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 |
| **二、实验内容：**   1. 安装gcc-multilib：     或者：    2、根据bits.c中的要求补全以下的函数：  int bitXor(int x, int y);  int min(void);  int isTmax(int x);  int allOddBits(int x);  int negate(int x);  int isAsciiDigit(int x);  int conditional(int x, int y, int z);  int isLessOrEqual(int x, int y);  int logicalNeg(int x);  int howManyBits(int x);  unsigned floatScale2(unsigned uf) ;  int floatFloat2Int(unsigned uf);  unsigned floatPower2(int x);  3、在Linux下测试以上函数是否正确，指令如下（详见README文件）：  \*编译：./dlc bits.c  \*测试：make btest  ./btest |
| **三、实验思路及求解过程：**  **#思路、#代码及最后的成绩截图（红色部分请删除）**  1.安装gcc-multilib：    2、根据bits.c中的要求补全以下的函数：  int bitXor(int x, int y);  int min(void);  int isTmax(int x);  int allOddBits(int x);  int negate(int x);  int isAsciiDigit(int x);  int conditional(int x, int y, int z);  int isLessOrEqual(int x, int y);  int logicalNeg(int x);  int howManyBits(int x);  unsigned floatScale2(unsigned uf) ;  int floatFloat2Int(unsigned uf);  unsigned floatPower2(int x);  （1）int bitXor(int x, int y)  思路：  a) x ^ y 等价于 (x & ~y) | (~x & y)  b) 由于只能使用 ~ 和 &，根据德摩根定律 A | B = ~(~A & ~B)，将 (x & ~y) | (~x & y) 中的或运算转换为与非运算。  代码：    （2）int tmin(void);  思路：  int tmin(void) 函数的目的是返回补码表示的最小整数。在补码表示中，对于 n 位整数，最小整数是最高位为 1，其余位为 0。在 32 位系统中，最小整数对应的二进制表示是 1000 0000 0000 0000 0000 0000 0000 0000，故直接计算即可  代码：    （3）int isTmax(int x)  思路：  将补码最大值加1后，此时新的数与补码最大值取反相同，将其与取反后的最大值进行异或，由此来判断是否是补码的最大值。但如果x也是 -1的话，那么也符合这个条件，所以还要判断这种特殊情况。  代码：    （4）int allOddBits(int x)  思路：  此函数用于判断一个整数的所有奇数位（从右往左数，第 1、3、5 等位置）是否都为 1。可以先构建一个奇数位全为 1 的掩码，再将该掩码与输入的数进行按位与操作，最后检查结果是否等于掩码。  代码：    （5）int negate(int x)  思路：  该函数用于返回一个数的相反数。在补码表示中，一个数的相反数可以通过对其取反再加 1 得到。  代码：    （6）int isAsciiDigit(int x)  思路：  此函数用于判断一个整数是否在 ASCII 码中数字 '0'（0x30）到 '9'（0x39）的范围内。可以通过计算 x - 0x30 和 0x39 - x 的结果，检查它们是否都为非负数。  代码：    （7）int conditional(int x, int y, int z)  思路：  该函数实现了三元运算符 x ? y : z 的功能。先根据 x 是否为 0 构建一个掩码，再利用该掩码选择 y 或 z。判断x是否为0，可以采用两次!运算，这里将得到的值进行取反加一的操作得到a，这样就可以从 1，0变成-1(二进制表示全为1)，0(二进制表示全为0)，由此通过运算后可以得到答案。  代码：    （8）int isLessOrEqual(int x, int y)  思路：  此函数用于判断 x 是否小于等于 y。可以通过计算 y - x 的结果，检查其符号位来判断大小关系，同时要考虑溢出的情况。一共分为两种情况：当x与y异号时，只需判断x是否为负数；当x与y同号时，返回y-x的符号位取反。  代码：    （9）int logicalNeg(int x)  思路：  该函数实现逻辑非运算符 ! 的功能。可以通过将 x 与其相反数按位或，再检查结果的符号位来判断 x 是否为 0。  代码：    （10）int howManyBits(int x)  思路：  x一共有3种情况，   1. x = 0, 需要1位 2. x > 0, 找到从最高位到最低位寻找第一个数据为1的位置，依次检查每一段中是否有1。如第一次检查x的低16位中是否有1, 若有则x至少需16 bits才可表示, 然后移除其低16位；若无，则无需移除，直接检查低8位。若有，则继续检查处理后的低8位。依次检查16、8、4、2、1位是否有1, 最终答案为各部分的1的个数加上符号位。 3. x < 0, 将x取反，转化成跟正数相同的处理方式。   代码：    （11）unsigned float\_twice(unsigned uf)  思路：  按照IEEE-754标准定义的浮点数:    图17  先截取出uf的符号s、阶码exp、尾数frac.  a) 当exp = 0xFF时，表示特殊值，  当frac不等于0表示不是一个数（NaN），返回uf  当frac等于0表示无穷，无穷 \* 2 结果还是无穷，返回uf  b) 当exp = 0时，表示非常接近0的数，此时只需要将小数字段乘以2。  c) 当exp 既不等于0，也不等于0xFF时，此时将指数字段，即exp，+1即可  代码：    （12）unsigned float\_i2f(int x)  思路：  a) 先求得x的符号位sign，阶码exp，尾数frac  b) 若x等于0，直接返回即可  c) 若x为int的最小值，将指数字段置为158  d) 若x为负，将其变为正数  e) 找到小数点的位置之后，将x截断至只剩下小数位，并计算exp，尾数部分frac计算之后，需要考虑舍入情况，若发生进位，则更新exp，并将frac截断  f) 最后将sign，exp，frac三部分或上即可。  代码：    （13）int float\_f2i(unsigned uf)  思路：  a) 从uf中截取出符号sign、阶码exp和尾数frac。  b) 然后判断是否规格化, 是否为特殊值, 是否溢出。  c) 将exp和frac转化为整型的补码, 或上符号位后即可得到答案。  代码：    3、在Linux下测试以上函数是否正确，指令如下  \*编译：make  \*测试：./btest |
| **四、实验结论及问题：**  运行测试程序，成功通过所有测试，实验完成  通过本次实验，围绕 C 语言数据类型的位级操作展开，通过完成 13 个函数的实现（涵盖整数位操作、条件判断、浮点数处理等），更加深入理解了计算机中数据的二进制表示方法，并掌握基于位运算的底层编程技巧。 |

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2023年4月 日 |
| 备注： |