**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（创新班）**

**指导教师： 刘 刚**

**报告人：何泽锋 学号：2022150221 班级： 高性能特色班**

**实验时间： 2024年4月 12 日 至 4月17 日**

**实验报告提交时间： 2024年4月 17 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：     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：**  1) 根据bits.c中的要求补全以下的函数：  1.int bitXor(int x, int y);  题目要求：使用~和&两种位运算实现异或操作  思路：使用德摩根律实现    图1 int bitXor(int x, int y)代码  2.int min(void);  题目要求：获取最小的int值  思路：二进制下int为32个比特位，根据补码知识可知其最小值是符号位为1其余全为0，由此可知，通过左移的方式可以获得答案    图2 int min(void)代码  3.int isTmax(int x);  题目要求：判断x是否为补码最大值  思路：首先需要获得int的最大值，即0x7fffffff，按位异或即可得到答案    图3 int isTmax(int x)代码  4.int allOddBits(int x);  题目要求：判断所有奇数位是否都为1  思路：首先将x与上0xAAAAAAAA,将奇数位取出，再将取出的结果异或a，得到的结果取否，即可    图4 int allOddBits(int x)代码  5.int negate(int x);  题目要求：不使用 - 操作符，求 -x 值  思路：取反加一    图5 int negate(int x)代码  6.int isAsciiDigit(int x);  题目要求：计算输入值是否是数字 0-9 的 ASCII 值（0x30-0x39）  思路：分别判断两边界，将0和9的ascll码取反加一，再与x进行与，分别判断当前符号位，0对于符号位需要为0，9对应符号位为1，此处将与0的结果取非得到1，表示满足条件，将与9的结果取两次非是将其转为0或1，最后二者相与得到结果    图6 int isAsciiDigit(int x)代码  7.int conditional(int x, int y, int z);  题目要求：使用位级运算实现C语言中的 x?y:z三目运算符  思路：将 x 的布尔值转换为全0或全1，即 x==0 时位表示是全0的， x!=0 时位表示是全1的    图7 int conditional(int x, int y, int z)代码  8.int isLessOrEqual(int x, int y);  题目要求：使用位运算符实现<=  思路：两种情况：一是符号不同正数大，二是符号相同看差值符号。    图8 int isLessOrEqual(int x, int y)代码  9.int logicalNeg(int x);  题目要求: 使用位运算求逻辑非 !  思路：若为0返回1，其余数返回0。只需要判断每个数与自身相反数或非结果的符号位即可，此处结果与1是为了将算数右移变为逻辑右移    图9 int logicalNeg(int x)代码  10.int howManyBits(int x);  题目要求：一个数用补码表示最少需要几位  思路：计算x的符号位mask。如果x为正数或0，则mask值等于0；如果x为负数，则mask值等于-1。对于负数x，为了能正确地计算它所需的位数，需要将x取反后再进行计算。因此，需要把x取反。接下来，采用分治的方式，每次去掉一半不需要的位，从而减少所需比特位数的计算。具体来说，首先判断x的高16位是否有值。如果有值，则至少需要16位才能表示x；否则，需要用低位存储x。3.接下来，使用x = x >> bit\_num将剩余需要表示的比特位数保存在x中。接着，在x中使用类似的方法计算剩余比特位数，直到判断完成8位、4位、2位和1位。在计算完所有的位数后，将所有的位数相加，得到需要表示    图10 int howManyBits(int x)代码  11.unsigned floatScale2(unsigned uf) ;  题目要求：求2乘一个浮点数  思路: 首先排除无穷小、0、无穷大和非数值NaN，此时浮点数指数部分（真正指数+bias）分别存储的为0，255。这些情况，无穷大和NaN都只需要返回参数，无穷小和0只需要将原数乘二再加上符号位就行了（并不会越界）。剩下的情况，如果指数+1之后为指数为255则返回原符号无穷大，否则返回指数+1之后的原符号数。    图11 unsigned floatScale2(unsigned uf)代码  12.int floatFloat2Int(unsigned uf);  题目要求：将浮点数转换为整数  思路：首先提取浮点数的符号位、指数部分和尾数部分，考虑特殊情况：如果原浮点值为0则返回0；如果真实指数大于31（frac部分是大于等于1的，1<<31位会覆盖符号位），返回规定的溢出值0x80000000u；如果 exp<0（1右移x位,x>0，结果为0）则返回0。剩下的情况：首先把小数部分（23位）转化为整数（和23比较），然后判断是否溢出：如果和原符号相同则直接返回，否则如果结果为负（原来为正）则溢出返回越界指定值0x80000000u，否则原来为负，结果为正，则需要返回其补码（相反数）。    图12 int floatFloat2Int(unsigned uf)代码  13.unsigned floatPower2(int x);  题目要求：计算  思路：首先得到偏移之后的指数值e，如果e小于等于0（为0时，结果为0，因为2.0的浮点表示frac部分为0），对应的如果e大于等于255则为无穷大或越界了。否则返回正常浮点值，frac为0，直接返回对应指数。    图13 unsigned floatPower2(int x)代码  **2)在Linux下测试以上函数是否正确**  编译运行展示:此处运行采用了./btest -T 20，这是因为直接运行最后一个案例会超时，但并没有出现代码上的错误，原因是数据量较大导致运行超时，因此增加判断时间即可避免超时    图14 编译运行结果  **3)了解btest的运行原理**  打开btest.c的文件可以看到，其本质就是一个用c语言写的运行程序，并且其数据也是随机生成的，但生成的数据是围绕在边界范围附近，比较能够测试代码的正确。同时，代码运用了较多的switch-case结构，减少了if-else的使用，一方面代码可读性较高，另一方面，转为汇编时也可以减少jmp的使用，代码效率较高。 |
| **四、实验结论及问题：**  本次实验考查了许多关于位运算的知识，对于数字逻辑的基本知识有了更深入的理解，同时也更加清晰的了解刀算数右移和逻辑右移的区别，需要严格控制好类别，不然容易导致错误。本次实验也对浮点数进行了了解，浮点数是由符号位、指数、和尾数组成，通过对三个部分的分别处理，可以实现浮点数与整形数据的相互转换。实验过程遇到了TLE的问题，排查后发现，是因为数据量较大导致超时，打开btest.c文件查看可知，测试程序将TLE的时间设置为10，对于不同的电脑运算速度会有一定的差异，因此导致超时发生，处理方法有两种，一种是直接修改测试程序，但这样修改有失严谨。因此本次实验通过在运行是设置超时的范围来处理的，解决后，可以发现所有案例都已正确，总分达到36分 |

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2024年4月 日 |
| 备注： |