**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： 计算机与软件学院所有专业**

**指导教师： 刘 刚**

**报告人： 郭天朗 学号：2023150243 班级： 计科02**

**实验时间： 2025年4月 日 至 4月 日**

**实验报告提交时间： 2025年4月 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：**   1. bitXor – x^y using only – and &   (1) 思路: , 而.  (2) 代码：     1. tmin - return minimum two's complement integer 2. 思路: int的最小值为, 其补码的MSB为1, 其余位为0. 3. 代码:      1. isTmax - returns 1 if x is the maximum, two's complement number, and 0 otherwise 2. 思路: 若x是int的最大值, 则其补码的MSB为0, 其余位为1. 考虑将其转化为0来判断.    1. y = x + 1为int的最小值, 其补码的MSB为1, 其余位为0.    2. x = x + y的补码全为1, 此时对x按位取反即为0.    3. 注意x = -1, 即x的补码全为1时, 会被①②误判为int的最大值. 此时y = x + 1的补码全为0, 则可通过检查!y是否为0判断. 3. 代码:      1. allOddBits - return 1 if all odd-numbered bits in word set to 1 2. 思路: 取掩码mask = 0xAAAAAAAA, 则它的二进制表示的奇数位为1, 偶数位为0, 检查x和x & mask是否相等即可. 因不允许使用大常数, 可将AA平移到对应数位后相加. 3. 代码:      1. negate - return -x 2. 思路: 负数的补码是对应的正数补码取反 + 1. 3. 代码:      1. isAsciiDigit - return 1 if 0x30 <= x <= 0x39 (ASCII codes for characters '0' to '9') 2. 思路:    1. 判断x >= 0x30, 可判断(x – 0x30)的符号位是否为0 .    2. 判断x <= 0x39, 可判断(x – 0x3A)的符号位是否为1, 此处不选减0x39是因为减完后符号位仍为0. 3. 代码:      1. conditional - same as x ? y : z 2. 思路:    1. 判断x是否为0, 可用!x将x先变为0或1. 考虑将x变为掩码: x = 0时, mask = !x – 1 = 0xffffffff; x != 0时, mask = !x – 1 = 0x00000000.    2. 返回y时, 需将z置0; 返回z时, 需将y置0. 上述操作可用mask、~mask分别与y和z相与实现. 3. 代码:      1. isLessOrEqual - if x <= y then return 1, else return 0 2. 思路: 对x，y作差，看符号位 3. 代码      1. logicalNeg - implement the !s operator, using all of the legal operators except ! 2. 思路:    1. 若x != 0且x != 0x8000, 则!x和x的符号位相反.    2. 若x == 0, 则!x和x的符号位都为0.    3. 若x == 0x8000, 则!x和x的符号位都为1.   综上, 只需!x和x的符号位至少有一个为1即可保证x非零.   1. 代码:      1. howManyBits - return the minimum number of bits required to represent x in two's complement 2. 思路:    1. 若x == 0, 则只需要1 bit.    2. 若x > 0, 设其MSB为第n位, 则只需再加上符号位, 即用(n + 1)可表示.    3. 若x < 0, 需找到其最高的位0的位置. 为简化, 将x取反. 将x的二进制表示按2的幂次的长度分段, 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, 移除其低16位, 检查接下来的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. 代码:      1. floatScale2 - Return bit-level equivalent of expression 2\*f for floating point argument f 2. 思路: 先按IEEE-754标准定义的浮点数, 分别截取出uf的符号sgn、阶码exp、尾数frac. 按uf是规格化或非规格化分类:    1. uf是非规格化浮点数, 即exp = 0时, frac乘2即可.    2. exp != 0时:       1. 若exp != 255, 则uf是规格化浮点数, exp加1即可. 注意若exp加1后exp变为全1, 则应返回Infinity, 即将frac置为0.       2. 若exp == 255, 则返回NaN即可. 3. 代码:      1. floatFloat2Int - Return bit-level equivalent of expression (int) f for floating point argument f 2. 思路:    1. 从uf中截取出符号sgn、阶码exp和尾数frac.    2. 判断是否规格化, 是否为特殊值, 是否溢出.    3. 将exp和frac转化为整型的补码, 或上符号位后返回即可.      1. floatPower2 - Return bit-level equivalent of the expression 2.0^x (2.0 raised to the power x) for any 32-bit integer x. 2. 思路：先特判阶码过大与过小的情况，然后再分别处理规格数与非规格数 3. 代码： |
| **四、实验结论及问题：**  **编译：**    **测试结果：**    **各个测试均通过，实验完成**  **体会：**  **本次实验进一步熟悉了位运算，对于数据的底层表示有了更深的理解** |

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2025年4月 日 |
| 备注： |