**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： 郑雨婷 学号：2021150122 班级： 高性能**

**实验时间： 2023年3月30日 至 4月13日**

**实验报告提交时间： 2023年4月 9 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：   阅读注释后输入Y同意，等待安装即可（安装时间较长，耐心等待）。    安装make。等待安装即可。    查看是否安装成功。下图表示安装成功。    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中的要求补全函数。  首先使用vi文本编辑器打开bits.c文件。    （1）int bitXor(int x, int y);  ①要求：    ②思路：  我们知道亦或是不进位的加法，所以共有四种可能：0+0=0；0+1=1；1+0=1；1+1=0；依次画出真值表：   |  |  |  | | --- | --- | --- | | x | y | Output | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 |   根据真值表可以求出Output=(~x&y)|(x&~y),但题目要求只能使用&和~,再使用摩根定律得：Output=~(~(~x&y)&~(x&~y))  ③代码实现：    （2）int tmin(void);  ①要求：    ②思路：32位整数中，最小值为0x 8000 0000，转化为二进制为1000 0000 0000 0000 0000 0000 0000 0000；即将1左移31位；  ③代码实现：    （3）int isTmax(int x);  ①要求：    ②思路：最大的二进制补码是0x7fff ffff; 0x7fff ffff+1=0x8000 0000；0x8000 0000加本身为0。0加本身也为0。只有0和0x8000 0000加本身为0。所以若x+1+x+1=0，x有两种可能的取值，0x7fff ffff或0x1111 1111。因此要排除掉0x1111 1111.  !(x+1+x+1)&!!(x+1)  ③代码实现：    （4）int allOddBits(int x);  ①要求：    ②思路：  令y=0xAAAAAAAA是奇数位为1，偶数位为0。将x和y进行按位与运算，所有偶数位都变为0，只剩下奇数位。这时若与y相等，说明原本的x所有奇数位为1，返回1；否则返回0。判断是否与0xAAAAAAAAA相等，只需与0xAAAAAAAAA异或，若为0则相等，否则不相等。  其中0xAAAAAAAA不能直接得到，因为题目要求整型常量仅可以用0~255；  ③代码实现：    （5）int negate(int x);  ①要求：    ②思路：按位取反再加一。  ③代码实现：    （6）int isAsciiDigit(int x);  ①要求：    ②思路：0~9在Ascii码表中对应着48~57。所以x在48~57即返回1。x-48的符号位为0则说明x大于等于48，57-x的符号位为0就说明x小于等于57。获取符号位的方法为右移31位两次逻辑非！！。注意题目中不能用-运算符，要按位取反再加一取得相反数后相加。  ③代码实现：    （7）int conditional(int x, int y, int z);  ①要求：    ②思路：构造出一个数t，在x=0时为1111111，~t=000000。这时候返回(t&z)|(~t&y)；  ③代码实现：    （8）int isLessOrEqual(int x, int y);  ①要求：    ②思路：判断x是否小于等于y，直接用y-x可能会超出int的表示范围，故而： 考虑按符号来分情况讨论：   * 1. x为负，y为正；   2. x与y同号，这样做减法不会溢出。y-x>=0符号位位0。   ③代码实现：    （9）int logicalNeg(int x);  ①要求：    ②思路：  思路：  可以通过取相反数进行非零判断。令y=~x+1（y=-x）并讨论x与y的符号位，有如下几种情况：  A.当x为0时，两者符号位都为0；  B.当x=0x8000 0000时，两者符号位都为1；  C.当x既不为0也不为0x8000 0000时，两者符号位为01或10。   |  |  |  | | --- | --- | --- | | x的符号位 | Y的符号位 | Output | | 0 | 0 | 1 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 0 |   因此，可依真值表得 a n s = ( ∼ x ) & ( ∼ y ) ，则当且仅当x=0x0时，其符号位为1。  ③代码实现：    （10）int howManyBits(int x);  ①要求：    ②思路：先考虑正数，每一次将x右移几位，如果得到的x非零，说明他至少有几位了，继续向右移，直到x变为0。至于每次位移几步，我们采用二分查找的思想。  如果是负数，就按位取反，要求的是位数，不需要加一了。只需把符号位变为0，即可用正数的方法。  ③代码实现：    （11）unsigned floatScale2(unsigned uf) ;  ①要求：    ②思路：  s表示符号位、exp 表示阶码位、frac 表示尾数位。 共存在三种可能的情况：  a.当 exp=0xff 时，是无穷大，无穷大的2倍还是无穷大,返回它本身。  b.当 exp=0 时，是非规格化的。分两种情况考虑： 当 uf[22]=0 时，然后将 frac 左移一位即可 当 uf[22]=1 时，将 exp 自增 1，然后再将 frac 左移一位即可  c.对于其他情况，是规格化的情况。将 exp 自增 1，然后再分两种情况： 当 exp==0xff，令 frac=0 即可。 其余情况正常左移并返回即可  ③代码实现：    （12）int floatFloat2Int(unsigned uf);  ①要求：    ②思路:  s表示符号位、exp 表示阶码位、frac 表示尾数位。int占32位，取值范围为-2147483648～2147483647（-2^{31} \sim 2^{31}-1），要考虑溢出的情况。分情况讨论：  a.当 exp=0 或 exp-127<0 时，返回 0；非规格化的舍入后得到0；  b.当 exp-127>=31 时候，超出表示范围，于是返回 0x80000000u；  c.当 exp-127<=23，根据符号位返回值 num>>(23-(exp-127))  ③代码实现：    （13）unsigned floatPower2(int x);  ①要求：    ②思路：  s表示符号位、exp 表示阶码位、frac 表示尾数位。分4类情况分类讨论  a. 溢出  b.规格化的[-126,127]  c.非规格化的[-149,-127]  d.x<-149,很小，返回0.  ③代码实现：    3.在 Linux 下测试以上函数是否正确，指令如下：  \*编译：./dlc bits.c  \*测试：make btest  ./btest  完成代码编写后进行编译测试：    运行程序并进行测试：    由上图看到，测试结果全部正确。 |
| **四、实验结论及问题：**  本次实验是通过位运算来进行数据表示。前十个函数是整型的题目，通过实验，现能够更熟练地运用~、!、^、&、|、<<、>>等运算符，对位运算的底层逻辑更加清楚了。后三道是浮点数的题目，难度非常大！要求我们对IEEE754规则很了解，并且要理解符号位、解码、尾数的含义。在进行整型与浮点数的转换，对非规格化和特殊情况全0全1等都需要进行特殊判断。通过实验，我牢牢记住了IEEE浮点标准包括其中的三种情况（规格化的、非规格化的、特殊的）。 |

|  |
| --- |
| 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： |