|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 课程编号 1502760001-07  题目类型 实验2 | |  |  |  | | --- | --- | --- | | **得分** | **教师签名** | **批改日期** | |  | 冯禹洪 |  | |

**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（腾班）**

**指导教师： 冯禹洪**

**报告人： 黄亮铭 学号： 2022155028 班级： 腾班**

**实验时间： 2024 年 4 月 18 日**

**实验报告提交时间： 2024 年 4 月 26 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：     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. 安装make、gcc-multilib：     图1：安装make    图2：安装gcc-multilib   1. 打开相应的文件  * 因为没有相关权限，因此使用root用户进入文件目录下，继而切换成新建用户名huangliangming\_2022155028（图3）。     图3：打开文件目录   * 然后使用vim命令进入bits.c文件中。     图4：bits.c文件   1. 根据bits.c中的要求补全以下的函数： 2. intbitXor(int x, int y);   使用非运算和与运算得到x某位为0，y相应位为1，或者x某位为1，y相应位为0会得到当前位为1的情况。然后使用或运算将他们合并即可得到答案。但是题目不允许，因此我们使用摩根定律将或运算转化为与运算。    图5：实现代码   1. inttmin(void);   0x80……0为最小二进制补码整数。使用移位运算，将0x1左移31位即可实现。    图6：实现代码   1. intisTmax(int x);   最大值为0x7F……F。将x左移加1取非可以得到答案。但是我们需要区分x是0x7F……F还是0xF……F。因此利用性质0xF……F+1=0，我们可以区分这一点。    图7：实现代码   1. ntallOddBits(int x);   由于题目要求，我们只能定义0xAA（奇数位位1，偶数位为0），然后将其左移即可得到奇数位位1，偶数位为0的掩码mask。    图8：实现代码   1. int negate(int x);   取反加1可以得到原数的负数。    图9：实现代码   1. intisAsciiDigit(int x);   我们需要判断x是否在[48,58]这个范围内，即需要检查x-48和58-x的符号位是否相同即可得到答案。    图10：实现代码   1. int conditional(int x, int y, int z);   思考如何能得到y和z？和0xF……F进行与运算。我们需要找到两种情况：只有x=0时才能得到0xF……F和只有x不为0时得到0xF……F。显然，图示代码中的代数可以满足要求。    图11：实现代码   1. intisLessOrEqual(int x, int y);   如果符号位异号，直接看x的符号位是否是1即可判断。如果同号，则相减（取反加1相加）后再看符号位。这里主要要排除符号位异号导致溢出的情况，使用！sig即可（！sig成立代表同号，不会溢出；！sig不成立代表异号，应由前一种情况判断）。    图12：实现代码   1. intlogicalNeg(int x);   x和-x的符号为均为0，则x=0。    图13：实现代码   1. inthowManyBits(int x);   对于负数，我们需要知道最高位0的位置；对于整数，我们需要知道最高位1的位置。然后**将该位置加上1即为所需比特数**。首先我们需要利用符号位，将正负数都统一成判断最高位1的位置，然后利用二分即可找到。以b16为例，如果x的高16位存在1，说明答案至少为16，因此实现的代码中！！（x>>16）会得到1，1<<4得到b16=16。如果x的高16位不存在1，说明答案小于16，同样的代码会得到b16=0。其他的类似，因此可以二分得到答案。    图14：实现代码   1. unsignedfloat\_twice(unsigned uf);   我们需要进行分类讨论：① 当阶码为0xFF时，直接返回本身。② 当阶码为0，且尾数最高位为1时，需要将阶码加1，并让尾数左移一位。③ 当阶码为0，且尾数最高位为0时，直接让尾数左移一位。④ 对于其他情况，阶码加1，如果此时阶码为0xFF，则将尾数全置0。    图15：实现代码   1. unsigned float\_i2f(int x);   符号位执行右移操作后简单处理即可获得。类似HowManyBits，获得x的位数，阶码为127+位数。注意尾数需要规格化。此外，我们需要注意对尾数的舍弃。    图16：实现代码   1. int float\_f2i(unsigned uf);   将float类型的数值分割为符号位s、阶码exp和尾数frac。我们需要分类讨论：① 当exp=0或exp<127时，返回0；② 当exp-127>=31时，返回0x8……0；③ 当exp-127<=23，执行移位操作，注意规格化数字要变为非规格化数字。    图17：实现代码   1. 在Linux下测试以上函数是否正确：   使用sudo make命令进行编译。    图：编译  输入命令./btest，得到结果如下图。    图：代码得分 |
| **实验结论：**  实验需要补充的函数均补充完整。经过测试，代码得分为满分37分。 |
| **心得体会：**   1. 位运算原理简单，但是可以实现非常复杂且强大的功能。 2. 虽然通过几个位运算操作即可实现很多功能，但是也意味着代码会非常复杂。这需要我们具备较强地能力去思考计算的逻辑。 3. 本次实验使我明白了int类型和float类型的存储原理，明白了如何考虑int类型等的溢出情况，也学会对float类型的阶码的一些特殊情况进行考虑。 |
| 指导教师批阅意见：  成绩评定：  指导教师签字： 冯禹洪  2024年 月 日 |
| 备注： |