**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： 王 毅**

**报告人： 钟善扬 学号： 2017303031 班级：01**

**实 验 时 间： 2020年5月27日**

**实验报告提交时间： 2020年5月27日**

**教务处制**

**一、 实验目标：**

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、实验环境：**

1. 计算机（Intel CPU）
2. Linux操作系统

**三、实验内容与步骤**

1、根据bits.c中的要求补全以下的函数：

**1、int bitXor(int x, int y)**

思路：

德摩根定律

int bitXor(int x, int y)

{

  return (~(x&y))&(~((~x)&(~y)));

}

**2、int tmin(void)**

思路：

最小值是0x80000000,用1左移31位得到。

int tmin(void)

{

  return 1<<31;

}

**3、int isTmax(int x)**

思路：

判断是否为0x7FFFFFFF，通过+1然后与自身的取反相异或得到0再取非得到1，并排除0xFFFFFFFF

int isTmax(int x)

{

  return !((x+1)^(~x))&!!(x+1);

}

**4、int allOddBits(int x)**

思路：

把x按位取反后，再和0xAAAAAAAA与，看结果是不是0。

int allOddBits(int x)

{

  return (!(((((((0xaa<<8)+0xaa)<<8)+0xaa)<<8)+0xaa)&(~x)));

}

**5、int negate(int x)**

思路：

取反加一

int negate(int x)

{

  return ~x+1;

}

**6、int isAsciiDigit(int x)**

思路：

ASCII码中的0值为48，使用x减去48所得结果符号位应为0；

ASCII码中的9值为57，使用x减去58所得结果符号位应为1。

如果同时符合这两个条件就返回1

int isAsciiDigit(int x)

{

  return (!((x+~48+1)>>31))&!!((x+~58+1)>>31);

}

**7、int conditional(int x, int y, int z)**

思路：

x?y:z，即当x为非0数时返回y，当x为0时返回z，可推出公式(a&y)|(b&z)

如果x不为0，就输出y&(0xffffffff) | z&(00000000)

如果x是0，就输出y&(0x00000000) | z&(ffffffff)

容易推出：

如果x不为0，!x-1为0x00000000,而!!x-1为0xffffffff;

如果x是0，!x-1为0xffffffff,而!!x-1为0x00000000;

int conditional(int x, int y, int z)

{

  return (((!x+(~1+1))&y)|((!(!x)+(~1+1))&z));

}

**8、int isLessOrEqual(int x, int y)**

思路：用a，b记录x，y的最高位，用c记录y减去x的值的最高位的取非值。如果a，b相同，则求亦或后取非得到1，此时c为1就返回1，c为0就返回0.而如果a，b不相同，则如果x是负号即返回1，即返回!!a。

int isLessOrEqual(int x, int y)

{

int a=x>>31;

int b=y>>31;

int c = !((y+~x+1)>>31);

return ((!(a^b))&c) | (((a^b))&(!!a));

}

**9、int logicalNeg(int x)**

思路：

如果一个数是0，取反+1后仍然是0，如果一个数非零，那么这个数和他取反+1后的数必然符号不相等。所以对每个数取反+1再取反与自己取反相与，符号位为1就是0，符号位为0就不是0。

int logicalNeg(int x)

{

  return (((~(~x+1))&(~x))>>31)&1;

}

**10、int howManyBits(int x)**

思路：

如果一个数是正数，那么只需要知道它最高位的1在哪然后再加一位符号位就是所需的位数。如果一个数是负数，那么只需知道它最高位的0在哪然后加一位符号位。所以找最高位的1所在的位置。用二分的方法，第一次需要从32位中找，所以先判断左移16位后前半段是否还有1，如果有只需查找前16位，否则只需查找后16位。依次类推，最后将所有结果加起来再加一位符号位。

然后需要特判三种情况：

如果x是负数，那么对x取反即可。

如果x是0，输出1

如果x是-1，输出1

int howManyBits(int x)

{

    int bit1, bit2, bit4, bit8, bit16,sum;

    int now = x^(x>>31);

    int isZero = !x;

    int isnone = !(~x);

    int mark1 = ((!(!x))<<31)>>31;

    int mark2 = ((!(!(~x)))<<31)>>31;

    bit16 = (!!(now>>16)) <<4;

    now = now >> bit16;

    bit8 = (!!(now>>8)) <<3;

    now = now >> bit8;

    bit4 = (!!(now>>4)) <<2;

    now = now >> bit4;

    bit2 = (!!(now>>2)) <<1;

    now = now >> bit2;

    bit1 = (!!(now>>1));

    sum = bit16+bit8+bit4+bit2+bit1+2;

    return isnone|isZero|(sum&mark1&mark2);

}

**11、unsigned float\_twice(unsigned uf)**

思路：对于0则直接返回0，对于已经溢出的数，也直接返回它自身。然后就是记录符号位flag，指数位exp和小数位fra，如果exp不为0，则只需exp+1然后fra左移1位，如果exp为0，则要判断是否有进位。最后将flag，exp，fra都相或组合起来。

unsigned float\_twice(unsigned uf)

{

if(uf==0)return 0;

unsigned flag = (uf>>31);

unsigned exp = (uf>>23)&0xFF;

unsigned fra = uf&((1<<23) - 1);

if(exp==255)//溢出

return uf;

if(exp==0)

{

if(fra>>22)//首位为1

exp++;//进位

fra<<=1;

}

else

{

exp++;

if(exp==255) fra=0;

}

flag<<=31;

exp<<=23;

fra=fra&((1<<23) - 1);

return flag|exp|fra;

}

**12、unsigned float\_i2f(int x)**

思路：

①如果该数为0，直接返回0.

②如果该数为最小的int，那么这个数就是-2的31次方，把指数位置位158就可以了。

③分三步得到这个数的符号位，指数位，小数位

首先取出这个数的符号位。

然后用一个while循环得出这个数最高位的1在哪一位，这个数加上127就是这个数的指数位的值了

最后来求小数位，先把这个数左移到第一个1与第31位齐平，用0x7fffff和这个数右移八位相与取出了前23位。

还要判断后面的数是否会导致进位以及进位之后是否会导致指数增加：

剩下的数取出来，如果大于128或者等于128同时小数位最后一位为1，那么就产生进位，于是让这个数+1；只要检查+1后的数的第24位是否为1，如果为1，指数加一就可以了。

unsigned float\_i2f(int x)

{

    int ch = x&0x80000000;

    int exp,num = 0,i = 30, is\_plu;

    if(x==0)

        return x;

    else if(x==0x80000000)

            exp = 158;

else

{

        if(x<0)

            x = -x;

        while(!(x>>i))

            i--;

        exp = i+127;

        x = x<<(31-i);

        num = 0x7fffff&(x>>8);//取出前23位

        x = x&0xff;

        is\_plu = ((x>128)||((x==128)&&(num&1)));

        num+=is\_plu;

        if(num>>23)

{

            num&=0x7fffff;

            exp++;

        }

    }

    return ch|(exp<<23)|num;

}

**13、int float\_f2i(unsigned uf)**

思路：

对于绝对值大于2^31或者小于1的直接返回负无穷或者0。分别求出符号位flag，指数位exp，小数位frac，对于exp在[0,23]之间的，frac向左移动（23-exp）位，对于exp在(23,31]之间的，frac向右移动（exp-23）位，最后返回frac或者-frac。

int float\_f2i(unsigned uf)

{

int exp = (0x7F800000&uf)>>23;

unsigned frac = (0x007FFFFF&uf)+0x00800000;

unsigned flag= 0x80000000&uf;

exp-=127;

if(exp<0)return 0;

else if(exp>=31)return 0x80000000;

else if(exp<=23)frac>>=(23-exp);

else frac<<=(exp-23);

if(flag)return -frac;

else return frac;

}

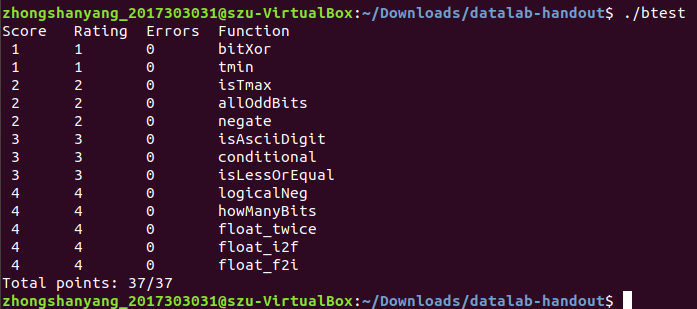
2、在Linux下测试以上函数是否正确，指令如下：

\*编译：./dlc bits.c

\*测试：make btest

./btest

**四、实验结果**

****

1. **实验总结与体会**

默认的Apt源来自Ubuntu官网，国内已经无法访问，因此无法下载gcc-multilib，需要更改Apt源。清华大学的镜像可能出现包的依赖问题，尝试使用aptitude-get，aptitude给出的解决方案需要卸载网络配置相关包导致无法上网。最后将sources.list更改为以下源地址后依赖问题得到解决：

deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse

deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe multiverse

deb http://mirrors.163.com/ubuntu/ xenial main restricted universe multiverse

deb http://mirrors.163.com/ubuntu/ xenial-security main restricted universe multiverse

deb http://mirrors.163.com/ubuntu/ xenial-updates main restricted universe multiverse

deb http://mirrors.163.com/ubuntu/ xenial-backports main restricted universe multiverse

##测试版源

deb http://mirrors.163.com/ubuntu/ xenial-proposed main restricted universe multiverse

## 源码

deb-src http://mirrors.163.com/ubuntu/ xenial main restricted universe multiverse

deb-src http://mirrors.163.com/ubuntu/ xenial-security main restricted universe multiverse

deb-src http://mirrors.163.com/ubuntu/ xenial-updates main restricted universe multiverse

deb-src http://mirrors.163.com/ubuntu/ xenial-backports main restricted universe multiverse

##测试版源

deb-src http://mirrors.163.com/ubuntu/ xenial-proposed main restricted universe multiverse

本次实验难点在于，思路很容易得到，但是用位运算操作写给计算机执行比较麻烦，而且容易出现很多bug，特别是最后三个难题。但是，如果使用高级语言，这些问题就是很简单的。所以最后发展出了宏。

最后三个难题让我加深了对浮点数的IEEE标准的认识和对数值机器码转化的理解。

|  |
| --- |
| **指导教师批阅意见：**  **成绩评定：**  指导教师签字： 王毅    2020年5月15日 |
| 备注： |