

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

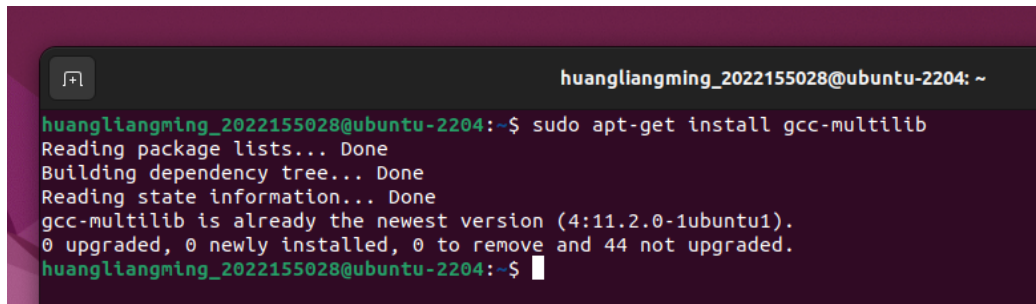
教务处制

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:



```
huangliangming_2022155028@ubuntu-2204: ~  
huangliangming_2022155028@ubuntu-2204:~$ sudo apt-get install gcc-multilib  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
gcc-multilib is already the newest version (4:11.2.0-1ubuntu1).  
0 upgraded, 0 newly installed, 0 to remove and 44 not upgraded.  
huangliangming_2022155028@ubuntu-2204:~$
```

- 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);  
inhowManyBits(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

注: 1、报告内的项目或内容设置, 可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

实验过程及内容:

1. 安装 make、gcc-multilib:

```
root@ubuntu-2204:/home/ubuntu# sudo apt-get install make
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2 liblvm13
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  make-doc
The following NEW packages will be installed:
  make
```

图 1: 安装 make

```
Processing triggers for Man-db (2.10.2-1) ...
root@ubuntu-2204:/home/ubuntu# sudo apt-get install gcc-multilib
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2 liblvm13
```

图 2: 安装 gcc-multilib

2. 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)。

```
ubuntu@ubuntu-2204:~$ su huangliangming_2022155028
Password:
huangliangming_2022155028@ubuntu-2204:/home/ubuntu$ ls
ls: cannot open directory '.': Permission denied
huangliangming_2022155028@ubuntu-2204:/home/ubuntu$ su
Password:
root@ubuntu-2204:/home/ubuntu# ls
Desktop Documents Downloads Music Pictures Public snap Templates Videos
root@ubuntu-2204:/home/ubuntu# cd Desktop
root@ubuntu-2204:/home/ubuntu/Desktop# cd datalab-handout
root@ubuntu-2204:/home/ubuntu/Desktop/datalab-handout# su huangliangming_2022155028
huangliangming_2022155028@ubuntu-2204:/home/ubuntu/Desktop/datalab-handout$ ls
bddcheck bits.c bits.h btest.c btest.h decl.c dlc Driverhdrs.pm Driverlib.pm
huangliangming_2022155028@ubuntu-2204:/home/ubuntu/Desktop/datalab-handout$
```

图 3: 打开文件目录

- 然后使用 vim 命令进入 bits.c 文件中。

```
Some of the problems restrict the set of allowed
Each "Expr" may consist of multiple operators. You
one operator per line.

You are expressly forbidden to:
1. Use any control constructs such as if, do, while, etc.
2. Define or use any macros.
3. Define any additional functions in this file.
"bits.c" [readonly] 286L, 8244B
```

图 4: bits.c 文件

注: 1、报告内的项目或内容设置, 可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

3. 根据 bits.c 中的要求补全以下的函数:

1) `int bitXor(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。

```
*/
int bitXor(int x, int y) {
    return ~((~(~x&y))&(~(x&~y)));
}
/*
```

图 5: 实现代码

2) `int tmin(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。

```
*/
int tmin(void) {
    return 1<<31;
}
//2
/*
```

图 6: 实现代码

3) `int isTmax(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。

```
*/
int isTmax(int x) {
    int tmp = x + 1;
    x = x + tmp;
    x = ~x;
    tmp = !tmp;
    x = x + tmp;
    return !x;
}
/*
```

图 7: 实现代码

4) `int allOddBits(int x);`

由于题目要求, 我们只能定义 `0xAA` (奇数位为 1, 偶数位为 0), 然后将其左移即可得到奇数位为 1, 偶数位为 0 的掩码 mask。

```
*/
int allOddBits(int x) {
    int a = 0xAA;
    int mask = a + (a << 8) + (a << 16) + (a << 24);
    return !((mask & x) ^ mask);
}
/*
```

图 8: 实现代码

注: 1、报告内的项目或内容设置, 可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

- 5) `int negate(int x);`
取反加 1 可以得到原数的负数。

```
*/
int negate(int x) {
    return ~x + 1;
}
//3
/*
```

图 9：实现代码

- 6) `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。

```
*/
int isAsciiDigit(int x) {
    int l = x - 48;
    int r = 58 - x;
    return !((l >> 31) | (r >> 31));
}
/*
```

图 10：实现代码

- 7) `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。

```
*/
int conditional(int x, int y, int z) {
    int cd1 = !x - 1; //x!=0:cd1=0xff x=0:cd1=0
    int cd2 = ~!x + 1; //x=0:cd2=0xff x!=0:cd2=0
    return (cd1 & y) | (cd2 & z);
}
/*
```

图 11：实现代码

- 8) `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）。

```
*/
int isLessOrEqual(int x, int y) {
    int sig1 = x >> 31;
    int sig2 = y >> 31;
    int sig = sig1 ^ sig2;
    int tmp = y + ~x + 1;
    tmp = tmp >> 31;
    return (sig & sig1 & 1) | ((!sig & (!tmp)));
}
//4
```

图 12：实现代码

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

- 9) `int logicalNeg(int x);`
x 和 -x 的符号均为 0，则 `x=0`。

```
*/  
int logicalNeg(int x) {  
    return ((~(~x + 1) & ~x) >> 31)  
}  
/* howManyBits - return the minimum
```

图 13: 实现代码

- 10) `int howManyBits(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。

```
*/  
int howManyBits(int x) {  
    int sig = x >> 31;  
    x = (sig & ~x) | (~sig & x);  
    int b16, b8, b4, b2, b1, b0;  
    b16 = !!((x >> 16) << 4);  
    x = x >> b16;  
    b8 = !!((x >> 8) << 3);  
    x = x >> b8;  
    b4 = !!((x >> 4) << 2);  
    x = x >> b4;  
    b2 = !!((x >> 2) << 1);  
    x = x >> b2;  
    b1 = !!((x >> 1));  
    x = x >> b1;  
    b0 = !!(x);  
    return b16 + b8 + b4 + b2 + b1 + b0 + 1;  
}  
//float
```

图 14: 实现代码

- 11) `unsigned float_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。

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

```

unsigned float_twice(unsigned uf) {
    unsigned int sig = 0x80000000 & uf;
    unsigned int exp = 0x7f800000 & uf;
    unsigned int frac = 0x007fffff & uf;
    if (!((~exp) & 0x7f800000))
        return uf;
    if (!exp) {
        frac = frac << 1;
        if (uf & 0x00400000)
            exp = 0x00800000;
    }
    else {
        exp = exp + 0x00800000;
        if (!((~exp) & 0x7f800000))
            frac = 0;
    }
    return (sig | exp | frac);
}

```

图 15: 实现代码

12) 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。

```

unsigned float_i2f(int x) {
    unsigned absx;
    if (x < 0) absx = -x;
    else absx = x;
    unsigned sig = 0x80000000 & x;
    unsigned t = 0x40000000, t2 = 0;
    unsigned exp = 0, exp_sig = 0;
    unsigned frac = 0, r_off = 0;
    if (!x)
        return x;
    else if (x == 0x80000000)
        return sig + (158 << 23);
    else {
        while (!(t & absx)) {
            t = t >> 1;
            exp_sig = exp_sig + 1;
        }
        exp = 30 - exp_sig;
        frac = (t - 1) & absx;
        if (exp > 23) {
            t2 = exp - 23;
            if (((frac << (31 - (t2 - 1))) == 0x80000000)) {
                if (((frac & (1 << t2)) != 0)) r_off = 1;
            }
            else if ((frac & (1 << (t2 - 1))) != 0) {
                r_off = 1;
            }
            frac = frac >> t2;
        }
        else {
            frac = frac << (23 - exp);
        }
        exp = 157 - exp_sig;
        return (sig + (exp << 23) + frac + r_off);
    }
}

```

图 16: 实现代码

13) 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。

注: 1、报告内的项目或内容设置, 可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

```

int float_f2i(unsigned uf) {
    int exp = 0x7f800000 & uf;
    int exp_sig = ((exp >> 23) - 127);
    unsigned sig = 0x80000000 & uf;
    int frac = 0x007fffff & uf;
    int frac2 = frac + 0x008fffff;
    if (exp == 0x7f800000)
        return 0x80000000;
    if (!exp)
        return 0;
    if (exp_sig >= -126 && exp_sig <= -1)
        return 0;
    if (exp == 31 && !frac && sig)
        return 0x80000000;
    if (exp_sig >= 31)
        return 0x80000000;
    if (exp_sig <= 23)
        frac2 = frac2 >> (23 - exp);
    else
        frac2 = frac2 << (exp - 23);
    if (!sig)
        return frac2;
    else
        return -frac2;
}

```

图 17: 实现代码

4. 在 Linux 下测试以上函数是否正确：
使用 `sudo make` 命令进行编译。

```

huangliangming_2022155028@ubuntu-2204:/home/ubuntu/Desktop/datalab-handout$ sudo
make
gcc -O1 -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c

```

图: 编译

输入命令 `./btest`, 得到结果如下图。

```

huangliangming_2022155028@ubuntu-2204:/home/ubuntu/Desktop/datalab-handout$ sudo
./btest
Score Rating Errors Function
1      1      0    bitXor
1      1      0    tmin
2      2      0    isTmax
2      2      0    alloddBits
2      2      0    negate
3      3      0    isAsciiDigit
3      3      0    conditional
3      3      0    isLessOrEqual
4      4      0    logicalNeg
4      4      0    howManyBits
4      4      0    float_twice
4      4      0    float_i2f
4      4      0    float_f2i
Total points: 37/37

```

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。

注: 1、报告内的项目或内容设置, 可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

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:

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。