# 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：数计

指导教师：罗胜

报告人：詹耿羽 学号：2023193026 班级：数计

实验时间：2025 年 3 月 24 日

实验报告提交时间：2025 年 3 月 29 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：

```
test@szu-VirtualBox:/media/sf_计系2/datalab-handout$ sudo apt-get install gcc-multilib
```

或者：

```
test@szu-VirtualBox:/media/sf_计系2/datalab-handout$ su
Password:
root@szu-VirtualBox:/media/sf_计系2/datalab-handout# apt-get install gcc-multilib
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc-multilib is already the newest version (4:7.2.0-1ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 84 not upgraded.
```

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

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。

1. bitXor – x^y using only – and &

(1) 思路: $x \oplus y = (x\& \sim y)|(\sim x\&y)$，而 $x|y = a\&(\sim b)$.

(2) 代码:

```
145  int bitXor(int x, int y) {
146      return ~(~(x & ~y) & ~(y & ~x));
147  }
```

图 1: bitXor()函数的实现

2. tmin - return minimum two's complement integer

(1) 思路: int 的最小值为 $-2^{31}$，其补码的 MSB 为 1，其余位为 0.

(2) 代码:

```
155  int tmin(void) {
156      return 1 << 31;
157  }
```

图 2: tmin()函数的实现

3. isTmax - returns 1 if x is the maximum, two's complement number, and 0 otherwise

(1) 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.

(2) 代码:

```
167  int isTmax(int x) {
168      int y = x + 1;
169      x = x + y;
170      x = ~x;
171      y = !y;
172      x = x + y;
173      return !x;
174  }
```

图 3: isTmax()函数的实现

4. allOddBits - return 1 if all odd-numbered bits in word set to 1

(1) 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.

(2) 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。

```
183  int allOddBits(int x) {
184      int mask = (0xAA << 24) | (0xAA << 16) | (0xAA << 8) | 0xAA;
185      x = x & mask;
186      return !(x ^ mask);
187  }
```

图 4: addOddBits()函数的实现

5. negate - return -x

(1) 思路: 负数的补码是对应的正数补码取反 + 1.

(2) 代码:

```
196  int negate(int x) {
197      return ~x + 1;
198  }
```

图 5: negate()函数的实现

6. isAsciiDigit - return 1 if 0x30 <= x <= 0x39 (ASCII codes for characters '0' to '9')

(1) 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.

(2) 代码:

```
210  int isAsciiDigit(int x) {
211      int lower = ~0x30 + 1;
212      int upper = ~0x3a + 1;
213      int sgn1 = x + lower >> 31;
214      int sgn2 = x + upper >> 31;
215      return !sgn1 & sgn2;
216  }
```

图 6: isAsciiDigit()函数的实现

7. conditional - same as x ? y : z

(1) 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.

(2) 代码:

```
225  int conditional(int x, int y, int z) {
226      int neg_one = ~1 + 1;
227      int mask = !x + neg_one;
228      return (mask & y) | (~mask & z);
229  }
230
```

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。

图 7: conditional()函数的实现

8. isLessOrEqual - if x <= y then return 1, else return 0

(1) 思路: 按 x 与 y 的符号位不同分为两种情况.

① x 与 y 异号: 负数小, 返回 x 是否为负数即可.

② x 与 y 同号: 返回(y -x)的符号位取反即可.

(2) 代码:

```
238  int isLessOrEqual(int x, int y) {
239      int sgn_x = x >> 31 & 1;
240      int sgn_y = y >> 31 & 1;
241      int sgn_xor = sgn_x ^ sgn_y;
242
243      int neg_x = ~x + 1;
244      int y_minus_x = y + neg_x;
245      int sgn_y_minus_x = y_minus_x >> 31 & 1;
246      return (sgn_xor & sgn_x) | (!sgn_xor & !sgn_y_minus_x);
247  }
```

图 8: isLessOrEqual()函数的实现

9. logicalNeg - implement the !s operator, using all of the legal operators except !

(1) 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.

(2) 代码:

```
258  int logicalNeg(int x) {
259      int neg_x = ~x + 1;
260      x = x | neg_x;
261      return (x >> 31) + 1;
262  }
```

图 9: logicalNeg()函数的实现

10. howManyBits - return the minimum number of bits required to represent x in two's complement

(1) 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.

(2) 代码:

```
int howManyBits(int x) {
    // if x < 0, then nothing happens, else, flip
    int sgn = x >> 31;
    x = (sgn & ~x) | (~sgn & x);

    // check if there are bits in every sections
    int b16, b8, b4, b2, b1, b0;
    // check the lowest 16 places
    b16 = !!(x >> 16) << 4;
    x = x >> b16;

    // check the lowest 8 places
    b8 = !!(x >> 8) << 3;
    x = x >> b8;

    // check the lowest 4 places
    b4 = !!(x >> 4) << 2;
    x = x >> b4;

    // check the lowest 2 places
    b2 = !!(x >> 2) << 1;
    x = x >> b2;

    // check the lowest place
    b1 = !!(x >> 1);
    x = x >> b1;

    b0 = x;
    return b16 + b8 + b4 + b2 + b1 + b0 + 1;  // + 1 for the sign bit
}
```

图 10: howManyBits()函数的实现

11. float_twice - Return bit-level equivalent of expression 2*f for floating point argument f

(1) 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:

(i) 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.

(ii) 若 exp == 255, 则返回 NaN 即可.

(2) 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。

```
319 unsigned float_twice(unsigned uf) {
320     unsigned sgn = uf & (0x80 << 24);
321     unsigned exp = uf & ((0x7f << 24) + (0x80 << 16));
322     unsigned frac = uf & ((0x7f << 16) + (0xff << 8) + 0xff);
323 
324     if (!exp) frac = frac << 1;  // Denormalized
325     else {  // uf < 0
326         int mask = (0x7f << 24) + (0x80 << 16);
327         if (exp ^ mask) {  // not all bits are 1 in exp
328             exp += 0x80 << 16;  // (exp + 1) means (x 2) for Normalized
329             if (!(exp ^ mask))  // all bits are 1 in exp, then set frac to 0 and return inf
330                 frac = 0;
331         }
332     }
333     return sgn | exp | frac;
334 }
```

图 11: float_twice()函数的实现

## 12. float_i2f - Return bit-level equivalent of expression (float) x

(1) 思路:

① 先求得 x 的符号 sgn. 求阶码 exp 和尾数 frac 前, 可特判 x 取特殊值的情况. 若 x == 0, 因 0 的非规格化表示即全 0, 故返回 x 自身即可; 若 x == -infinity, 则将 exp 置为 0x9e 后返回即可.

② 若 x 为负数, 则将其变为正数.

③ 确定小数点的位置 pos 后, 将 x 截断至剩下小数位, 并根据 e = E + bias 求得 exp.

④ 求得小数位 frac, 注意特判舍入的情况, 若舍入后发生进位, 则更新 exp, 并将 frac 截断.

(2) 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。

```c
unsigned float_i2f(int x) {
    int sgn = x >> 31 & 1;
    int exp = 0;
    int frac = 0;

    if (!x) return x;  // the float expression for 0 (Denormalized) is itself
    else if (x == (1 << 31)) exp = 0x9e;
    else {
        if (sgn) x = -x;  // transform x into a positive number

        int pos;  // binary point
        for (pos = 30; !(x >> pos); pos--);
        x = x << (31 - pos);
        exp = pos + 127;  // e = E + bias

        int mask = (0x7f << 16) | (0xff << 8) | 0xff;
        frac = (x >> 8) & mask;

        x = x & 0xff;  // lowest 8 places
        int flag = x > 128 || (x == 128 && (frac & 1));  // round to even numbers

        frac = frac + flag;
        if (frac >> 23) {  // carry
            frac = frac & mask;
            exp = exp + 1;
        }
    }
    return (sgn << 31) | (exp << 23) | frac;
}
```

图 12: float_i2f()函数的实现

13. float_f2i - Return bit-level equivalent of expression (int) f for floating point argument f

(1) 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.

(2) 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。

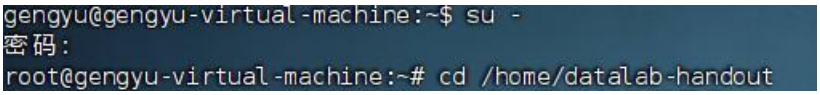
```c
int float_f2i(unsigned uf) {
    unsigned sgn = uf & (0x80 << 24);
    unsigned exp = (uf & ((0x7f << 24) + (0x80 << 16))) >> 23;
    unsigned frac = (uf & ((0x7f << 16) + (0xff << 8) + 0xff)) | (1 << 23);

    if (!exp || exp < 127) return 0;  // zero or Denormalized less than 1
    else if (exp >= 31 + 127) return 1 << 31;  // out of range

    // transform exp and frac into complement
    int E = exp - 127;
    frac = frac >> 23 - E;
    unsigned neg_frac = ~frac + 1;

    unsigned neg_one = ~1 + 1;
    unsigned mask = !sgn + neg_one;
    unsigned neg_not_s = ~(!sgn) + 1;
    frac = (mask & neg_frac) | (neg_not_s & frac);
    return sgn | frac;
}
```

图 13: float_f2i()函数的实现

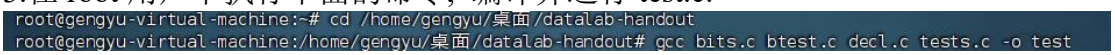
## 实验结论：

1.打开 finalshell 连接虚拟机：


2.切换到 root 用户：

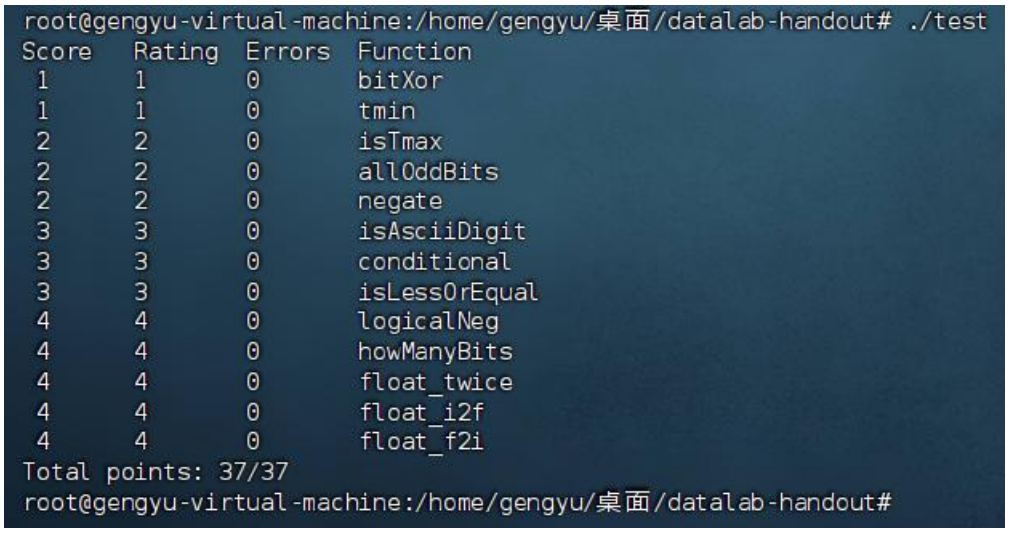



3.在 root 用户下执行下面的命令，编译并运行 test.c.




4.观察到各函数的测试都通过，实验完成！

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。




## 心得体会：

1.本次实验具有一定的挑战性，主要考察对位运算技巧的熟练掌握以及对特殊情况和边界条件的处理能力。
2.通过本次实验，我提升了位运算技能，加深了对定点数和 IEEE754 浮点数的理解，并对数值表示可能引发的程序错误有了更明确的认识。

指导教师批阅意见：

成绩评定：

指导教师签字：

2018 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。