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：计算机与软件学院所有专业

指导教师：刘　刚

报告人：林宪亮　学号：2022150130　班级：国际班

实验时间：2024 年 4 月 17 日 至 4 月 26 日

实验报告提交时间：2024 年 4 月 21 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：

```
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：

```
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

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。

三、实验思路及求解过程：

#思路、#代码及最后的成绩截图（红色部分请删除）
每张图都要在图下有图标题，截图请使用白底色。

1. 安装 gcc-multilib：

我们发现 Linux 中并没有 make 命令，所以我们需要安装 gcc-multilib，这时我们需要通过 apt-get 来安装对应的包，但是由于自带的 apt 源是在国外的，所以我们需要先把 apt 源换成国内的镜像。这里使用清华源，只需要 https://mirror.tuna.tsinghua.edu.cn/help/ubuntu/中的内容替换 root 用户下的/etc/apt/source.list 的内容即可。之后使用 apt-get update 命令更新，就可以安装 gcc-multilib 了。

```
root@lxl-virtual-machine:/etc/apt# apt-get install make gcc-multilib
正在读取软件包列表... 完成
正在分析软件包的依赖关系树... 完成
正在读取状态信息... 完成
将会同时安装下列软件：
  gcc-11-multilib lib32asan6 lib32atomic1 lib32gcc-11-dev lib32gcc-s1
  lib32gomp1 lib32itm1 lib32quadmath0 lib32stdc++6 lib32ubsan1 libc6-dev-i386
  libc6-dev-x32 libc6-i386 libc6-x32 libx32asan6 libx32atomic1
  libx32gcc-11-dev libx32gcc-s1 libx32gomp1 libx32itm1 libx32quadmath0
  libx32stdc++6 libx32ubsan1
```

图 1 安装 gcc-multilib

如图 1，gcc-multilib 安装成功。

2. 根据要求不全 bits.c 中的函数：

（1）int bitXor(int x, int y);
题目描述：
实现 x^y
示例：bitXor（4， 5） = 1
合法操作：~ &
最大运算次数：14
评分： 1

思路：此函数要求返回 x 异或 y 的结果，那么画出真值表并根据德摩根定律，很容易可以得到 X^Y=(X|Y)&(~X|~Y)=~(~X&~Y)&~(X&Y)，所以只需要把表达式的值返回即可。

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。

表一： 真值表

X	Y	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0

代码：

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

实现代码即返回 x 和 y 的表达式即可。

（2）int min(void);
题目描述：
返回最小二的补码整数
合法操作：！ ~ & ^ |+ << >>
最大运算次数：4
评分： 1

思路：这个函数是为了返回最小的补码值。而 32 位有符号整数的最小值即为首位是 1，其余位全是 0，所以直接返回 1 << 31 即可。

代码：

```
int tmin(void)
{
  // 最小补码 10000...
  return 1 << 31;
}
```

（3）int isTmax(int x);
题目描述：
如果 x 是最大值（二进制补码数），则返回 1，否则为 0
合法操作：！ ~& ^ | +
最大操作数：10
评分：1

思路：此函数是为了判断 x 是不是补码的最大值，补码的最大值为 01111..111，而补码的最大值加一为 100000...000，观察可以发现，它们两个每一位都是不同的，所以它们异或运算之后再取反结果应该是 0。但存在特殊情况，当 x=11111..111 时，要进行特判，即 x+1！=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。

代码：

```
int isTmax(int x)
{
  int y = (x + 1);
  int flag = ~(y ^ x); // 两值相异再取反
  return !flag & !!y;  // x+1!=0
}
```

变量 y 在存储 x+1 的值的同时，也可以用来特判 x 是不是为 1111....1111。

（4）int allOddBits(int x);
题目描述：
如果字中的所有奇数位均设置为 1，则返回 1
示例 allOddBits(0xFFFFFFFD) = 0, allOddBits(0xAAAAAAAA) = 1
合法操作：！ ~& ^ | + << >>
最大操作数：12
评分：2

思路：此函数是为了判断 x 的奇数位是不是全部为 1，那么我们只需要通过位移操作，构造一个奇数位全为 1，偶数位全位 0 的值 flag，用来判断 x 的奇数位是不是全为 1，如果 x 的奇数位全为 1 的话，那么相与之后的值应该和 flag 是一样的。

代码：

```
int allOddBits(int x)
{
  int y = (0xAA << 8) + 0XAA;
  int flag = (y << 16) + y;
  int vue = x & flag;
  int result = !(vue ^ flag);
  return result;
}
```

（5）int negate(int x);
题目描述：
返回-x
示例：取反(1) = -1。
合法操作：！ ~& ^ | + << >>
最大操作数：5
评分：2

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。

思路：此函数的目的是返回 x 的相反数，即对 x 取反再加一即可。

代码;

```
1.  int negate(int x)
2.  {
3.    return ~x + 1;
4.  }
```

（6）int isAsciiDigit(int x);
题目描述：
如果 0x30 <= x <= 0x39，则返回 1（字符“0”到“9”的 ASCII 代码）
示例：isAsciiDigit(0x35) = 1，isAsciiDigit(0x3a) = 0，isAsciiDigit(0x05) = 0。
合法操作：！ ~& ^ | + << >>
最大操作数：15
评分：3

思路:此函数是为了判断 x 是不是整数,即判断 x 的 Ascii 值是否满足 0x30 <=x<= 0x39。那么就是判断是不是 x-0x30 >= 0，并且 x-0x3A <= 0。那么就是求出 0x30 的相反数和 0x3A 的相反数，与 x 相加，判断结果的符号位即可。
值得注意的是，我们使用的是 0x3A 而不是 0x39。

代码：

```
1.  int isAsciiDigit(int x)
2.  {
3.    int low = ~0x30 + 1;
4.    int high = ~0x3A + 1;
5.    int Flaglow = !((x + lower) >> 31);
6.    int Flaghigh = (x + upper) >> 31;
7.    return Flaglow & Flaghigh;
8.  }
```

（7）int conditional(int x, int y, int z);
题目描述：
实现 x ? y : z
示例：条件(2,4,5) = 4
合法操作：！ ~& ^ | + << >>
最大操作数：16
评分：3

思路：此函数是为了实现三元运算 x? y: z，即 x 不是 0 的时候输出 y，x 是 0 的时候输出 z，所以可以考虑使用 flag，如果 x 是 0，则设置 flag 全是 0，如果 x 不是 0，则设置 flag 全是 1。再使用 flag 跟 y 绑定，~flag 和 z 绑定，即可完

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。

成函数功能。

代码：

```
int conditional(int x, int y, int z)
{
  int flag = ~(((!x) << 31) >> 31);
  int result = (flag & y) | (~flag & z);
  return result;
}

```

使用！和位移符号可以实现 flag 跟 x 得绑定，即如果 x 是 0，则设置 flag 全是 0，如果 x 不是 0，则设置 flag 全是 1。

（8）int isLessOrEqual(int x, int y);
题目描述：
如果 x <= y 则返回 1，否则返回 0
示例：isLessOrEqual(4,5) = 1。
合法操作：！ ~& ^ | + << >>
最大操作数：24
评分：3

思路：这个函数是为了判断 x 是否<=y，那么只需要计算 y-x 的值，求得符号位是 0 还是 1 即可，即计算 y+（~x + 1），然后使用位移符号求得符号位，如果符号位是 1，则返回 0，如果符号位是 0，则返回 1。使用位移符号可以得到全 1，或者全 0 的二进制值。最后和 1 相与，即可以得到 1 或者 0 的返回值。

代码：

```
int isLessOrEqual(int x, int y)
{
  int sum = y +(~x + 1);
  return ~(sum >> 31)  & 1;
}
```

（9）int logicalNeg(int x);
题目描述：
实现！ 运算符
使用所有合法操作符除了！
示例：逻辑负数(3) = 0，逻辑负数(0) = 1
最大操作数：12
评分：4

思路：此函数是为了实现逻辑非符号，即如果输入的 x 值是 0，则返回 1，如果

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。

输入的 x 值不是 0，则返回 0。所以需要对 0 进行特判，而 0 有个特性，即 0 进行包括符号位的取反加一之后还是它自己，符号位不会改变，可以通过这一点判断输入的 x 是不是 0，当然有个特别值，即有符号整数的最小值进行包括符号位的取反加一也还是自己，所以需要进行特判然后进行排除。

代码：

```
int logicalNeg(int x)
{
  int sign = ((~x + 1) ^ x) >> 31;
  int tepan = (x >> 31);
  return (sign | tepan) + 1;
}
```

通过对 x 取反加一然后与原来的 x 异或，如果符号位没有改变则返回 0，改变了就返回-1，然后对 x 进行特判，如果 x 是 0 则会设置 tepan 的值为 0，如果 x 为 10000....000 则会设置 tepan 的值为-1。所以如果 x 为 0 就会返回 0+1=1，如果 x 不是 0 则会返回-1+1=0。

（10）int howManyBits(int x);
题目描述：
返回表示 x 所需的最小位数
示例：howManyBits(12) = 5
howManyBits(298) = 10
howManyBits(-5) = 4
howManyBits(0) = 1
howManyBits(-1) = 1
howManyBits(0x80000000) = 32
合法操作：！ ~& ^ | + << >>
最大操作数：90
评分：4

思路：此函数是为了计算表示输入值 x 所需要的最小比特数。因为是带符号的整数，所以最少需要 1 个比特表示符号位。对于正数，需要的最小比特从第一个出现的 1 开始统计，最后加上符号位即可，对于负数，则从第一个出现的 0 开始统计，然后加上符号位所需的一个比特即可。但是这样太麻烦了，所以我讲负数直接取反，这样就统一成了从第一个出现的 1 开始统计。

代码：

```
int howManyBits(int x)
{
```

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 sign = x >> 31;
    int result = 0;
    int temp = (sign & (~x)) | (~sign & x);
    x = temp;
    result = !!(temp >> 15) << 4; // x 右移 15 位，若非 0，说明 x 至少要 16 比特
    temp = x >> result;
    result |= !!(temp >> 7) << 3; // x 再右移 7 位，若非 0，说明 x 至少还要 8 比特
    temp = x >> result;
    result |= !!(temp >> 3) << 2; // x 再右移 3 位，若非 0，说明 x 至少还要 4 比特
    temp = x >> result;
    result |= !!(temp >> 1) << 1; // x 再右移 1 位，若非 0，说明 x 至少还要 2 比特
    temp = x >> result;
    result |= !!(temp); // 前面最多可移 30 比特。若 x 仍非 0，则说明 x 还要 1 比特，共 31 比特
    return result + 1; // 加上符号位所需的 1 比特
}
```

上面的代码先检查 x 是否需要超过 15 位表示，如果需要超过 15 位表示则会先将 result 设置位 10000，然后将 x 右移 16 位，如果不需要超过 15 位表示，则不移动 x，然后检查需不需要 8 位表示，之后就是同样的流程，对于例如 10 位表示的数，则会通过需要八位表示，然后右移八位，然后在检查是否还需要 2 位表示的时候加上剩下的 2，组成需要 10 位表示，最后返回 10 加上符号位的 1，即 11 位表示。这也证明了代码的正确性。

（11）unsigned floatScale2(unsigned uf);
题目描述：
返回表达式 2*f
合法操作：任何整数。
最大操作数：30
评分：4

思路：此函数的作用是将输入的浮点数×2，对于规格化数，让指数部分加一即可，但是需要判断是否溢出。对于非规格化数，指数全为 0，让小数部分左移一位即可。

代码：

```
unsigned float_twice(unsigned uf)
{
  unsigned sign = uf & 0x80000000;
  unsigned exp = uf & 0x7F800000;
  unsigned xiaoshu = uf & 0x007FFFFF;
  if (exp == 0x7F800000)
  { return uf;}
  if (exp == 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。

```
    { xiaoshu <<= 1;
      return sign | exp | xiaoshu; }// 将符号位、指数位、小数位组合
    exp += 0x00800000;
    if (exp == 0x7F800000)
      return sign | (0xFF << 23);
    return sign | exp | xiaoshu;
}
```

Sign，exp，xiaoshu 分别代表符号位，指数部分以及小数部分，如果指数部分全是 1 则直接返回，如果指数为 0，但是小数部分不是 0，则这不是个规格化的数，即使用位移符号使小数部分左移一位然后返回。如果是规格化的数，则指数部分加一，再判断是否溢出。

（12）int floatFloat2Int(unsigned uf);
题目描述：
返回表达式 (int) f 对于浮点参数 f。参数作为 unsigned int 传递，但是它被解释为一个的位级表示单精度浮点值。任何超出范围的内容（包括 NaN 和无穷大）都应该返回 0x80000000u。
合法操作：任何整数。
最大操作数：30
评分：4

思路：这个函数的目的是把有符号整数转化为浮点数表示。先分别求出浮点数的符号位，指数位，小数位。然后进行判断，如果实际的指数部分小于 0，则直接返回 0。如果实际的指数为 31 且小数位为 0，符号位是 1，则直接返回 0x80000000。如果指数部分超过 30，则表示超出范围，直接返回 0x80000000u。然后根据指数大小进行小数部分的左移或者右移。最后根据符号位返回补码形式。

代码：

```
int float_f2i(unsigned uf)
{
  unsigned sign = uf >> 31;
  int exp = ((uf & 0x7f800000) >> 23) - 127;
  unsigned xiaoshu = (uf & 0x007fffff) | 0x00800000;
  if (exp == 31 && xiaoshu == 0x00800000 && sign == -1)
    return 0x80000000;
  if (exp > 30)   return 0x80000000u;
  if (exp < 0)    return 0;
  if (exp > 23)  xiaoshu <<= (exp - 23);
  else  xiaoshu >>= (23 - exp);
  if (sign) xiashu = ~xiaoshu + 1;
  return xiaoshu;
}
```

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。

（13）unsigned floatPower2(int x);
题目描述：
返回表达式 2.0ˆx（2.0 的 x 次方）对于任何 32 位整数 x。
返回的无符号值应该具有相同的位
表示为单精度浮点数 2.0ˆx。
如果结果太小而无法表示为分母，则返回 0，如果太大，返回+INF。
合法操作：任何整数。
最大操作数：30
评分：4

思路：
如果指数 x 大于 127，这意味着结果将太大而无法表示为单精度浮点数（因为单精度浮点数的指数范围是 -126 到 127）。因此，代码返回一个表示正无穷大（+INF）的特殊浮点数值。在 IEEE 754 标准中，表示正无穷大的方法是将指数部分全部置为 1（0xFF），而尾数部分为 0。
如果指数 x 小于 -148，这意味着结果将太小而无法表示为非规格化数（因为非规格化数的指数范围是 -149 到 -126）。因此，代码直接返回 0，表示这个数太小以至于无法用单精度浮点数表示。
在 x >= -126 这个范围内，指数 x 能够表示规格化的单精度浮点数。首先，计算出对应的指数部分 exp，规格化数的指数部分是 x + 127。然后将指数部分 exp 左移 23 位（因为单精度浮点数的指数部分占据 8 位，尾数部分占据 23 位），最终得到表示 2 的 x 次幂的单精度浮点数。
最后一种情况，指数 x 超过了非规格化数的范围，因此需要将这个数转换为非规格化数。计算出对应的指数部分 t，非规格化数的指数部分是 148 + x。然后将尾数部分的某一位设置为 1（因为非规格化数的尾数部分的最高位必须为 0，这样才能区分规格化数和非规格化数）。这里将 1 左移 t 位来设置尾数部分的某一位为 1。

代码：

```
unsigned floatPower2(int x) {
  if (x > 127)
  {return (0xFF << 23);}
  else if (x < -148)
  {return 0;}
  else if (x >= -126)
  {int exp = x + 127;
   return (exp << 23);}
  else
  {int t = 148 + x;
   return (1 << t);}
}
```

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. 测试正确性：

通过 make 指令进行编译再使用 ./btest 进行测试。

```
lxl@lxl-virtual-machine:~/下载/datalab-handout$ make
gcc -O -Wall -m32 -lm -o btest bits.c btest.c decl.c tests.c
btest.c: In function 'test_function':
btest.c:334:23: warning: 'arg_test_range' may be used uninitialized [-Wm
nitialized]
  334 |     if (arg_test_range[2] < 1)
      |         ~~~~~~~~~~~~~~^~~
btest.c:299:9: note: 'arg_test_range' declared here
  299 |     int arg_test_range[3]; /* test range for each argument */
      |         ^~~~~~~~~~~~~~
lxl@lxl-virtual-machine:~/下载/datalab-handout$ ./btest
```

图 2 编译测试

得到测试结果如下图：

```
lxl@lxl-virtual-machine:~/下载/datalab-handout$ ./btest
Score   Rating  Errors  Function
 1      1       0       bitXor
 1      1       0       tmin
 1      1       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       floatScale2
 4      4       0       floatFloat2Int
 4      4       0       floatPower2
Total points: 36/36
```

图 3 测试结果

由图可以看出，编写的函数通过了所有测试并取得了满分。实验成功。

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。

四、实验结论及问题：

1. 浮点数的处理是比较麻烦的，不仅需要分成符号位，指数，小数三部分，更需要进行许多特别的判定，考虑它的特殊情况。
2. 在计算表示带符号整数所需要的最小比特数的时候，依次考虑是否需要 16，8，4，2，1 位，这样的设计十分精妙，可以覆盖所有的可能性。
3. “！！”两个非的使用，可以很好的把零值设置为 0，把非零值设置为 1，方便后续的返回操作。
4. 位移操作的使用可以把正数设置为全 0，负数设置为全 1，方便后续的操作。
5. “！”符号和“~”符号有区别，一个是把 0 值设置为 1，把非 0 值设置为 0，另一个是按位取反。

指导教师批阅意见：

成绩评定：

指导教师签字：

2024 年 4 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。