实验一 数据实验

一. 实验简介

进一步理解书中第二章《数据的机器级表示与处理》的内容，深刻理解整数、浮点数的表示和运算方法，掌握 GNU GCC 工具集的基本使用方法。

二. 实验要求

请按照要求补全实验工程目录下 bits.c 中的函数，并进行验证。包括以下 15 个函数，其中位操作函数 5 个，二进制补码运算函数 7 个，浮点运算函数 3 个。

1. 位操作函数

① int bitAnd(int x, int y)

- ✧ 功能：仅适用~和|运算符计算 x & y
- ✧ 示例：bitAnd(6, 5) = 4
- ✧ 难度：1
- ✧ 可使用的最多运算符数：8

② int getByte(int x, int n)

- ✧ 功能：从字 x 中获取第 n 个字节，其中 n=0 对应最低有效字节，n=3 对应最好有效字节
- ✧ 示例：getByte(0x12345678,1) = 0x56
- ✧ 难度：2
- ✧ 可使用的最多运算符数：6

③ int logicalShift(int x, int n)

- ✧ 功能：对有符号数 x **逻辑右移** n 位（0 <= n <= 31）
- ✧ 示例：logicalShift(0x87654321,4) = 0x08765432

- ✧ 难度：3
- ✧ 可使用的最多运算符数：20

④ int bitCount(int x)

- ✧ 功能：返回 x 中 1 的个数
- ✧ 示例：bitCount(5) = 2, bitCount(7) = 3
- ✧ 难度：4
- ✧ 可使用的最多运算符数：40

⑤ int bang(int x)

- ✧ 功能：计算!x，但不能使用运算符“!”
- ✧ 示例：bang(3) = 0, bang(0) = 1
- ✧ 难度：4
- ✧ 可使用的最多运算符数：12

2. 二进制补码运算函数

① int tmin(void)

- ✧ 功能：返回最小的 32 位补码表示的负数
- ✧ 难度：1
- ✧ 可使用的最多运算符数：4

② int fitsBits(int x, int n)

- ✧ 功能：如果 x 能够用 n 位二进制补码表示，则返回 1；否则，返回 0
- ✧ 示例：fitsBits(5,3) = 0, fitsBits(-4,3) = 1
- ✧ 难度：2
- ✧ 可使用的最多运算符数：15

③ int divpwr2(int x, int n)

- ✧ 功能：计算 $x/2^n$，0 <= n <= 30
- ✧ 示例：divpwr2(15,1) = 7, divpwr2(-33,4) = -2
- ✧ 难度：2
- ✧ 可使用的最多运算符数：15

④ int negate(int x)

- ✧ 功能：返回 x 取负的结果，即-x
- ✧ 示例：negate(1) = -1
- ✧ 难度：2
- ✧ 可使用的最多运算符数：5

⑤ int isPositive(int x)

- ✧ 功能：如果 x>0，返回 1；否则，返回 0
- ✧ 示例：isPositive(-1) = 0
- ✧ 难度：3
- ✧ 可使用的最多运算符数：8

⑥ int isLessOrEqual(int x, int y)

- ✧ 功能：如果 x<=y，返回 1；否则，返回 0
- ✧ 示例：isLessOrEqual(4,5) = 1
- ✧ 难度：3
- ✧ 可使用的最多运算符数 24

⑦ int ilog2(int x)

- ✧ 功能：返回$\lfloor \log_2(x) \rfloor$，x>0

- ✧ 示例：ilog2(16) = 4
- ✧ 难度：4
- ✧ 可使用的最多运算符数 90

2. 浮点运算函数

① unsigned float_neg(unsigned uf)

- ✧ 功能：返回单精度浮点数取负后的二进制表示，即-f。在函数中，参数和返回结果均采用 unsigned int 类型，为单精度浮点数的二进制表示。如果参数是 NaN，则直接将参数返回。
- ✧ 难度：2
- ✧ 可使用的最多运算符数：10

② unsigned float_i2f(int x)

- ✧ 功能：将整型数 x 转换为单精度浮点数，即(float) x
- ✧ 难度：4
- ✧ 可使用的最多运算符数：30

③ unsigned float_twice(unsigned uf)

- ✧ 功能：计算 2*f
- ✧ 难度：4
- ✧ 可使用的最多运算符数：30

上述三个函数中，*float_neg* 和 *float_twice* 必须能够处理所有的浮点数表示形式，即包括 NaN 和无穷。我们规定如果返回一个 NaN，则它的二进制表示为 0x7FC0000。

三. 实验规则和限制

1. 位操作函数和二进制补码运算函数

程序中允许使用：

① 运算符： ! ~ & ^ | + << >>，只能使用这 8 个运算符

② 范围在 0 - 255 之间的常数

③ 局部变量

程序内进制如下行为：

① 声明和使用全局变量

② 声明和使用定义宏

③ 声明和调用其他的函数

④ 类型的强制转换

⑤ 使用许可范围之外的运算符

⑥ 使用控制跳转语句：if else switch do while for

注意：违背以上原则均视为程序不正确!!

2. 浮点运算函数

程序中允许使用：

① 所有整数运算操作符，包括||，&&（以程序中的注释为准）

② 范围在 0 - 255 之间的常数

③ int 和 unsigned 类型局部变量

⑤ 有符号和无符号常量

⑥ 各种分支和循环语句，如 if else switch do while for

程序内进制如下行为：

① 声明和使用全局变量

② 声明和使用定义宏

③ 声明和调用其他的函数

④ 类型的强制转换

⑤ 使用许可范围之外的运算符

⑥ 数组、结构体 struct 和联合体 union

⑦ 任何浮点类型的变量、常量或操作

四. 评价方法

本次实验总分共 75 分，其中包括:

- **正确分：41**

每个题目都有对应的难度系数，正确完成一道题目则会获得和该题难度系数相同的分值。难度系数总和为 41。

- **性能分：30**

每到题目都可以使用布尔代数的方法进行暴力求解。但是，希望大家能够使用一些更聪明和更优雅的方式来解题。因此，我们对每个题目所使用运算符总数进行了限制，如果该题结果正确且运算符总数满足题目要求，则该题获得 2 分性能分。

- **代码风格分：4**

有意义且清晰的注释（关键在于质量而不是数量）；规范的代码书写格式。

五. 使用和相关说明

- 进入实验目录

使用账户和密码进入虚拟仿真平台，进入课程，打开实验虚拟机，可以在终端下使用以下命令进入实验目录。

```
cd Lab1/datalab-handout
```

进入目录后，可以对 bits.c 文件进行编辑（其他的文件不要进行任何修改），实现文件中定义的函数。代码编辑建议使用 vim。

在实验包中还提供了一些辅助工具，协助你进行自动评判以提高作业的质量。包括：btest、dlc 和 driver.pl。

- btest

这个程序主要用来检查你在 bits.c 中所实现的代码的正确性。主要是通过随机生成输入检查你所实现的函数在功能上的正确性。编译并使用 btest 需要在实验目录下输入以下两条命令

```
make
./best
```

每次修改 bits.c 后，都需要先重新运行 make 命令才能够使用 btest。运行 btest 可以测试所有的需要实现的函数的正确情况。如果你只需要测试其中的某一个函数，你可以使用-f 选项，并制定对应的参数名称来进行测试。例如只测试函数 bitAnd 的结果，可以输入以下命令

```
./btest -f bitAnd
```

如果需要测试指定的输入数据我们可以使用选项-1、-2 和-3 分别制定对应的第一、第二和第三个参数。下面的命令说明，我们测试函数 bitAnd，并使用指定的输入参数 0x30 和 0x20。

```
./btest -f bitAnd -1 0x30 -2 0x20
```

- dlc

这个程序主要用于检查我们所编写的代码是否符合作业中所提出的限制条

件。可以使用下面的命令来检查：

./dlc bits.c

dlc 程序只在检测出 bits.c 中存在不符合实现规则的错误时，才会打印错误信息，否则，不打印信息。

使用-e 选项可以检查每个函数中的运算符使用情况：

./dlc -e bits.c

使用-h 选项可以查看 dlc 的其他使用方法：

./dlc -h

- driver.pl

这个程序同时整合了 btest 和 dlc 的功能，将 btest 和 dlc 的结果同时输出至控制台。大家最终可以使用该程序对 bit.c 的实现进行最终的验证，并查看得分。

注意：每次修改 bits.c 文件后都需要在实验目录下首先运行 make 命令完成编译后，才可以使用上述工具检查你的程序。

bit.c 中已通过注释给出了实验中的所有注意事项，请大家实验过程中仔细阅读。

在 bit.c 中不要添加“include <stdio.h>”语句，避免与 dlc 程序冲突造成错误。但你仍然可以正常使用 printf 函数进行打印调试，gcc 会提示警告信息，你可以直接忽略这些警告。

除了上述三个程序外，为了更好的理解单精度浮点数的结构，实验还提供了 fshow 程序。编译并使用 fshow 需要在实验目录下输入以下两条命令。fshow 程序可接受任意表示形式的浮点数。

make

./fshow 2080374784

将打印如下信息：

Floating point value 2.658455992e+36

Bit Representation 0x7c000000, sign = 0, exponent = f8, fraction = 000000

Normalized. 1.0000000000 X 2^(121)

六. 作业的提交

将写好的 bits.c 文件上传至虚拟仿真实验平台。