

# 计算机系统基础实验 ICS 2020 秋季学期

## Datalab: Manipulating Bits

### 1 实验环境的准备

安装 Ubuntu 虚拟机，要求虚拟机能够连接网络。

### 2 简介

这次作业你要解开 15 道编程谜题，以熟悉整型及浮点数的位级表示。

### 3 操作步骤

从 QQ 课程群共享文件中下载 datalab-handout.tar 文件后，拷贝到你的 Linux 机器（虚拟机）上，输入命令：

```
linux> tar xvf datalab-handout.tar
```

解压缩后会得到多个文件，其中只需要修改 bits.c 文件。

bits.c 文件包含 15 个编程谜题的框架。你的任务是完成每个函数的框架，只能使用 straightline 的代码（也就是，不能有循环或者条件语句），也只能使用有限数量的 C 语言算术和逻辑运算符。具体来说就是，你只能使用下述八种运算符：

! ~ & ^ | + << >>

有一些函数有更严格的要求。同时，你也不能使用超过 8 位的常数。bits.c 的注释中有详细的解释，以及所期望的代码风格。

- 计算机系统基础实验  
请见 QQ 课程群共享文件

### 4 谜题

这部分描述 bits.c 中待实现的 15 个函数功能，一共分为三种类型。

表 1 位运算

函数名称	功能描述	难度等级	最大操作数
evenBits(void)	return word with all even-numbered bits set to 1	1	8
isEqual(int x, int y)	return 1 if x == y, and 0 otherwise	2	5
byteSwap(int x, int n, int m)	swaps the nth byte and the mth byte	2	25

rotateRight(int x, int n)	Rotate x to the right by n	3	25
logicalNeg(int x)	implement the ! operator, using all of the legal operators except !	4	12

表 2 二进制补码运算

函数名称	功能描述	难度等级	最大操作数
tmax(void)	return maximum two's complement integer	1	4
sign(int x)	return 1 if positive, 0 if zero, and -1 if negative	2	10
isGreater(int x, int y)	if $x > y$ then return 1, else return 0	3	24
subOK(int x, int y)	Determine if can compute $x-y$ without overflow	3	20
satAdd(int x, int y)	adds two numbers but when positive overflow occurs, returns maximum possible value, and when negative overflow occurs, it returns minimum negative value.	4	30
howManyBits(int x)	return the minimum number of bits required to represent x in two's complement	4	90
intLog2(int x)	return floor(log base 2 of x), where $x > 0$	4	90

表 3 浮点数运算

函数名称	功能描述	难度等级	最大操作数
float_half(uf)	calculate $0.5*f$	4	30
float_f2i(x)	return (int)f	4	30
float_twice(uf)	calculate $2*f$	4	30

#### bits.c文件:

这个是源码文件，里面包含了 15 个待实现的函数，已经给出函数原型。实验内容是按照每个函数的要求编写实现其功能的代码。例如：

```

/*
 * bitXor - x^y using only ~ and &
 *   Example: bitXor(4, 5) = 1
 *   Legal ops: ~ &
 *   Max ops: 14
 *   Rating: 1
 */
int bitXor(int x, int y) {
    return 2;
}

```

函数名: bitXor 参数:int, int

功能: 实现  $x^y$

要求: 只能使用 ~ 和 & 运算符, 将结果返回。

如例子所示，你需要做的就是将 bits.c 文件中的每个函数都按照要求实现，总体来说就是使用有限的操作实现要求的功能，上面的例子就是使用两个运算符~和&来实现^运算符的功能，并且运算符的个数不能超过 Max ops:14 个，这就需要你先去推理如何用~和&实现^，然后写出表达式。

```

/*
 * bitXor - x^y using only ~ and &
 *   Example: bitXor(4, 5) = 1
 *   Legal ops: ~ &
 *   Max ops: 14
 *   Rating: 1
 */
int bitXor(int x, int y) {
    int x_and_y = x&y;
    int x_or_y = ~(~x & ~y);
    return x_or_y & ~x_and_y;
}

```

请同学们务必认真阅读 `bits.c` 文件中的说明、注意事项和示例。

## 5 实验操作

**第一步：**将 `datalab-handout.tar` 文件上传到一台 Linux 机器上，执行如下命令解压：

```
linux> tar xvf datalab-handout.tar
```

你将看到一个名为 `datalab-handout` 的目录。

若上传的文件是 `.zip` 格式，可在 Windows 下解压后拷贝到 Ubuntu 中；也可以在 Ubuntu 下用 `unzip datalab-handout.zip` 解压（先 `mkdir` 一个文件夹，将 `.zip` 文件拷贝到新建好的文件夹中，然后使用 `unzip` 命令）。

**第二步：**实现 `bits.c` 中的函数，使用 `dlc` 编译器（`datalab checker`）检查代码是否满足编码要求，命令如下：

```
linux> ./dlc bits.c
```

如果没有问题，则不返回任何提示。

**第三步：**使用 `btest` 程序测试函数功能正确性。编译 `btest` 程序并进行测试，命令如下：

```
linux> make btest
```

可能会有一些 `warning`，请忽略。

```
linux> ./btest
```

注意，只要修改 `bits.c` 文件，就需要重新编译 `btest` 程序，命令如下：

```
linux> make clean
```

```
linux> make btest
```

`btest` 程序将自动运行很多组测试用例来检查你实现的函数，下面是一些 `btest` 的使用技巧：

```
linux> ./btest -h #输出 btest 命令的帮助信息
```

```
linux> ./btest -f foo #测试指定函数 foo 的正确性
```

```
linux> ./btest -f foo -1 27 -2 0xf #指定输入参数，测试函数 foo 的正确性
```

注意：如果对实验操作有不懂的地方，可自行阅读 `datalab-handout` 目录中的 `README` 文件。

## 6 提交要求

在目录下输入

```
linux> sudo chmod a+x ./driver.pl
```

```
linux> ./driver.pl -u “你的学号”
```

例如：

```
linux> ./driver.pl -u 2017202210003
```

```
4      4      0      ilog2
4      4      0      float_half
4      4      0      float_f2i
4      4      0      float_twice
Total points: 43/43
qinliu@ubuntu:~/ics/datalab-handout$ ./driver.pl -u 2017202210003
1. Running './dlc -z' to identify coding rules violations.
/usr/include/stdc-predef.h:1: Warning: Non-includable file <command-line> included from includable file /usr/include/stdc-predef.h.

Compilation Successful (1 warning)

2. Compiling and running './btest -g' to determine correctness score.
gcc -O -Wall -m64 -std=gnu89 -lm -o btest bits.c btest.c decl.c tests.c

3. Running './dlc -Z' to identify operator count violations.
/usr/include/stdc-predef.h:1: Warning: Non-includable file <command-line> included from includable file /usr/include/stdc-predef.h.

Compilation Successful (1 warning)

4. Compiling and running './btest -g -r 2' to determine performance score.
gcc -O -Wall -m64 -std=gnu89 -lm -o btest bits.c btest.c decl.c tests.c

5. Running './dlc -e' to get operator count of each function.

Correctness Results      Perf Results
Points  Rating  Errors  Points  Ops    Puzzle
1       1       0       2       4     bitAnd
2       2       0       2       3     getByte
3       3       0       2      14    logicalShift
4       4       0       2      25    bitCount
3       3       0       2       7    conditional
1       1       0       2       1     tmin
2       2       0       2       7     fitsBits
2       2       0       2       7     divpwr2
2       2       0       2       2     negate
4       4       0       2      37    howManyBits
3       3       0       2      13    isLessOrEqual
4       4       0       2      78    ilog2
4       4       0       2      19    float_half
4       4       0       2      13    float_f2i
4       4       0       2      15    float_twice

Score = 73/73 [43/43 Corr + 30/30 Perf] (245 total operators)
Success: Sent autoresult string for qinliu:2017202210003 to the result server.
qinliu@ubuntu:~/ics/datalab-handout$
```

注意：需要给 perl 脚本可执行权限（使用命令 `chmod a+x *`）

可以在 <http://47.92.205.8:8900/> 下看到提交的结果，网页每 30 秒更新一次，提交结果后最多需要等待 30 秒能在页面上看到自己的提交记录。（注意：服务器端并发处理比较慢，请不要一直刷屏）

Beat the Prof. 网页上 Score 这一列显示的分数是指 Instructor 的总操作数减去学生实验的总操作数，差值越大说明学生实验的操作数越少。注意，Instructor 的操作数比题目中允许的最大操作数要小，所以如果仅仅只是通过了 dlc 和 btest 的检查提交的结果在此得分可能是负数。

Instructor 的总操作数仍有较大的优化空间。为了鼓励同学们做出更优化的结果，本实验的成绩转换为百分制为  $55 + \text{Score} * 0.7$  （may be subject to change）

提交的实验操作数与 Instructor 的操作数相同，Score 显示为 0，如下图所示。

← → ↻ ⓘ 不安全 47.92.205.8:8900

应用 百度 必应 Google Scholar 研究生新系统 研究生老系统 本科系统 Java SE 8 Doc JCDK 301

### Scoreboard for the Data Lab "Beat the Prof" Contest

This page shows the operator counts for the students who have submitted entries to the Data Lab "Beat the Prof" contest.

- To enter the contest, run the driver with the -u option: `./driver.pl -u "nickname"`.
- Enter as often as you like. The page will show only your most recent submission.
- Blue entries match the instructor. Red entries beat the instructor. Incorrect entries are denoted by "`_`".
- Entries are sorted by score, defined as  $(total\ instructor\ operations - total\ student\ operations)$ . Higher scores are better.
- If all of your puzzle entries are correct and they each match or beat the instructor, then you're a **winner!**

Puzzle key: 1=bitAnd, 2=getByte, 3=logicalShift, 4=bitCount, 5=conditional, 6=tmin, 7=fitsBits, 8=divpwr2, 9=negate, 10=howManyBits, 11=isLessOrEqual, 12=ilog2, 13=float\_half, 14=float\_f2i, 15=float\_twice

Last updated: Thu Oct 10 01:43:59 2019 (updated every 30 secs)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Winner?	Score	Nickname
4	3	14	25	7	1	7	7	2	37	13	78	19	13	15	Winner!	0	2017202210003
4	3	14	25	7	1	7	7	2	37	13	78	19	13	15		0	TheProf

提交的实验操作数比 Instructor 的操作数大，Score 显示为负数，如下图所示。

← → ↻ ⓘ 不安全 47.92.205.8:8900

应用 百度 必应 Google Scholar 研究生新系统 研究生老系统 本科系统 Java SE 8 Doc JCDK 301

### Scoreboard for the Data Lab "Beat the Prof" Contest

This page shows the operator counts for the students who have submitted entries to the Data Lab "Beat the Prof" contest.

- To enter the contest, run the driver with the -u option: `./driver.pl -u "nickname"`.
- Enter as often as you like. The page will show only your most recent submission.
- Blue entries match the instructor. Red entries beat the instructor. Incorrect entries are denoted by "`_`".
- Entries are sorted by score, defined as  $(total\ instructor\ operations - total\ student\ operations)$ . Higher scores are better.
- If all of your puzzle entries are correct and they each match or beat the instructor, then you're a **winner!**

Puzzle key: 1=bitAnd, 2=getByte, 3=logicalShift, 4=bitCount, 5=conditional, 6=tmin, 7=fitsBits, 8=divpwr2, 9=negate, 10=howManyBits, 11=isLessOrEqual, 12=ilog2, 13=float\_half, 14=float\_f2i, 15=float\_twice

Last updated: Thu Oct 10 01:58:30 2019 (updated every 30 secs)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Winner?	Score	Nickname
4	3	14	25	7	1	7	7	2	37	13	78	19	13	15		0	TheProf
4	3	14	25	7	4	7	7	2	37	13	78	19	13	15		-3	2017202210003

- 结果提交截止时间：**2020 年 10 月 25 日 22 时**（截止时间前可提交任意次，每次提交覆盖前次提交；截止时间后服务器关闭，无法继续提交，以最后一次提交为该实验评分依据）
- 若实验提交出现异常，请联系教辅；每位同学成功提交了最终版的结果后，**请对网页相关记录截图，并记录下时间**，以备出现问题方便核对。
- 最终版的 bits.c 文件，请命名为 **学号\_姓名\_bits.c** 在 **2020 年 10 月 26 日 24 时之前**提交课代表，邮件主题请写为 **学号\_姓名\_bits.c**
- **每道题的注释部分都应包括清晰必要的解题思路说明，否则视为抄袭。**

## 7 注意事项

- 如果编译时发现很多头文件找不到，尝试执行如下命令安装必要的库：

```
sudo apt-get install build-essential libc6-dev libc6-dev-i386
```

```
sudo apt-get install gcc-4.7-multilib g++-4.7-multilib
```

遇到缺库的报错请先百度自己解决

- 如果你的机器是 32 位的，那么将 Makefile 中

```
CFLAGS = -O -Wall -m64 -std=gnu89
```

改为 CFLAGS = -O -Wall -m32 -std=gnu89

本实验对 32 位或 64 位系统均适用。