

# 南京航空航天大学计算机科学与技术学院计算机组成原理

## 实验 Datalab 数据表示实验报告

### 实验信息

姓名：李元成  
学号：161630216  
班级：1616302  
指导教师：李博涵  
实验日期：2018/6/26

### 第 1 题：bitNor

#### 题目要求理解

或非门（英语：NOR gate）  
就是 x 或 y 再非一下

#### 编码思路分析

注意到要求中没有按位或, 按摩根定律把式子展开, 为非 a 与非 b

#### 解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

单击或点击此处输入文字。

### 第 2 题：bitXor

#### 题目要求理解

异或门（英语：Exclusive-OR gate，简称 XOR gate）

## 编码思路分析

$x \text{ xor } y = (\text{非 } x \text{ 与 } y) \text{ 再或下 } (x \text{ 与非 } y)$

## 解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

## 第 3 题：

### 题目要求理解

判断偶数位是否都为 1，是就返回 1，不是就是 0

## 编码思路分析

首先我想的是把所有偶数位都提取出来，但是看到 max ops 为 12，很明显这不现实，要想其他的办法。

想到可以用移位操作，每次对半切这个串，让偶数位与偶数位与，奇数位与奇数位与，最后的长串就变成的短串。看最后是否为 1 即可

## 解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

## 第 4 题：

### 题目要求理解

传进 3 个参数  $x, n, m$  把  $x$  的  $n$  和  $m$  字节位交换一下

## 编码思路分析

首先要把  $x$  的两个位提取出来，因为一个数字是 8 位，所以先把  $n$  或  $m$  乘 2 的 3 次方 ( $\ll 3$ )，再让  $x \gg$  这个数，最后和  $0xff$  与一下，就截取到了。然后把  $x$  上的  $m$ ， $n$  两个字节的值弄成 0，再或一下  $t1$  和  $t2$  就 ok 了。

## 解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

## 第 5 题：

### 题目要求理解

乘 5/8  
包括溢出

## 编码思路分析

乘 5 就是乘 4 加自己，除 8 直接右移即可，但是我发现负数是过不了的  
查阅资料后发现负数的右移和除法是不一样的，除除法运算，结果都向 0 取整；  
位运算结果向下取整，所以如果是负数，要加个 7 补偿一下

## 解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

## 第 6 题：

### 题目要求理解

C 语言中的  $x \ ? \ y \ : \ z$

## 编码思路分析

因为不能用 `if` 来做流程判断, 所以只好拿 `x` 来控制。我们想办法把 `x` 变成 `0x00000000` 和 `0xffffffff` 就行了。首先我们先把 `x` 变成 0 和 1, 取反即可。有了 1 和 0, 再把它按位取反, 加上 1, 就是我们想要的东西, 最后两个数分别与一下, 再或下就行

## 解题源代码

如题, 粘贴该函数的源代码, 此处必须为文本, 不允许截图

## 第 7 题：

### 题目要求理解

`isGreater` 比大小

## 编码思路分析

考虑到正负号, 这个问题变为 2 个问题, 符号相同和符号不同。如果符号相同, 我们就做差,  $-x = \sim x + 1$ , 即如果 `x` 比 `y` 大,  $x + \sim y$  是大于等于 0 的 (标志位为 0), 如果符号不同, 那就把差变为 0. 然后我们对符号位比较下, 特判 `x=1, y=0` 为 1, 其余都是 0. 只要这两个有一个是 1 的话, 那就是  $x \leq y$ 。

## 解题源代码

如题, 粘贴该函数的源代码, 此处必须为文本, 不允许截图

## 第 8 题：

### 题目要求理解

传进三个参数,  $x, y, n$ 。取  $x$  和  $y$  的  $n$  位上的数字与一下, 再把这个与完的数放回  $x$  的  $n$  位上

### 编码思路分析

其实就是 `byteSwap` 的变形, 中间加个与就行了

### 解题源代码

如题, 粘贴该函数的源代码, 此处必须为文本, 不允许截图

## 第 9 题：

### 题目要求理解

不使用!的情况下实现!

### 编码思路分析

取  $x$  的相反数 $\sim x+1$ , 当  $x$  为 0, 两者符号位都是 0; 当  $x$  是 `0x80000000` 的时候, 都是 1, 其他情况都是 01 或者 10. 我们构造 $\sim x+1\&-x$  这个

X 负数    1

X 正数    1

X 0        0

X `0x80`~   1

然后取反, 取标志位即可。

### 解题源代码

如题, 粘贴该函数的源代码, 此处必须为文本, 不允许截图

## 第 10 题：

### 题目要求理解

float\_abs, 给 float 取绝对值

### 编码思路分析

根据书上 float 的表格，只有当是负非规格化数、负 0、负规格化数和负无穷时，才需要使符号位取反。我们可以用 int 来进行判别。Int 小于 0xFF800000，都是需要翻转符号的。然后我们采用异或来进行翻转，和 0 异或不变，和 1 异或翻转

### 解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

## 第 11 题：

### 题目要求理解

float\_i2f, 把 int 转为 float

### 编码思路分析

首先分离符号位、阶码分离，分别放到 sign 和 exp 中。通过表可以知道，非规格化数整数是 0，除了 0、非规格化数和规格化数外。其他的数都输出 0x80000000，对于 0 特判输出 0。当阶码  $\text{exp} < 127$ ，输出 0，阶码大于 31，是溢出。输出 0x80000000。其余情况直接左移，符号位为 1 取反。

### 解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

## 第 12 题：

### 题目要求理解

Float\*0.5

### 编码思路分析

通过阶码来特判 NaN 或者 infinite，阶码 $\geq 126$ 的时候，直接减 1，其他情况下阶码和尾码统一右移一位。

### 解题源代码

如题，粘贴该函数的源代码，此处必须为文本，不允许截图

### 本地检查情况

### 语法检查情况

使用`./dlc bits.c`命令对实现代码进行检查，并将检查情况截图粘贴在下方。

```
root@root:~/lab1-handout# ./dlc bits.c
/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)
```

### 结果检查情况

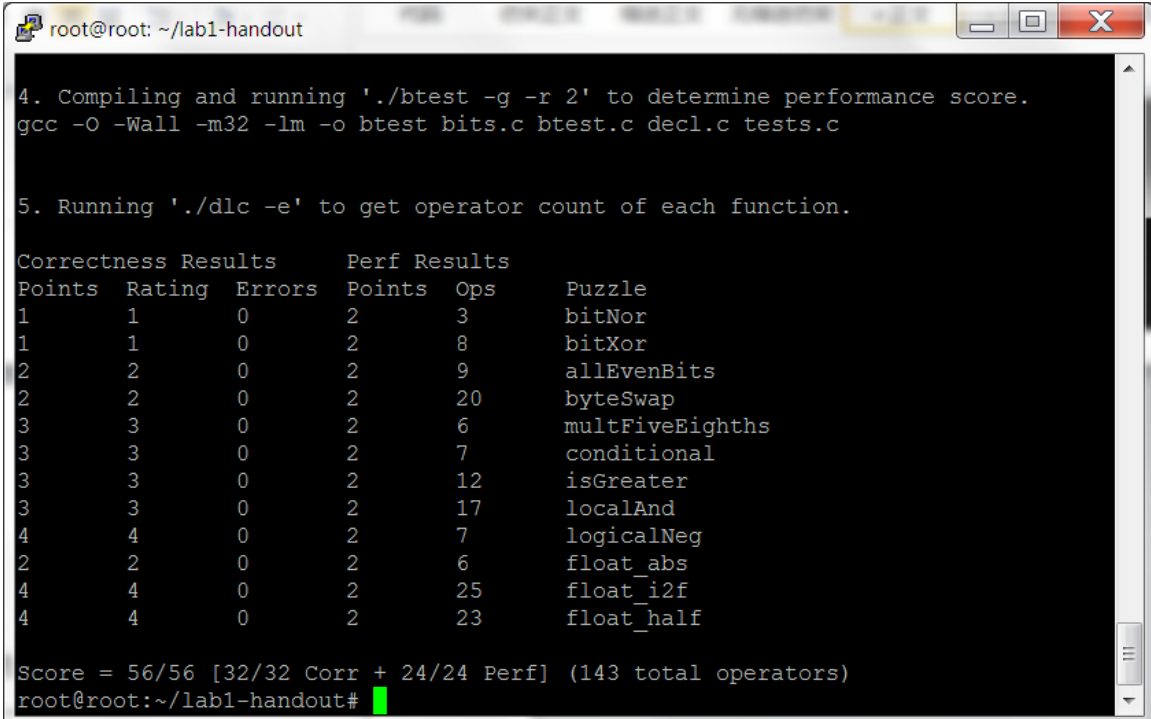
使用`./btest bits.c`命令对实现代码的运行情况进行检查，并将检查情况截图

粘贴在下方。

```
root@root:~/lab1-handout# ./btest bits.c
Score  Rating  Errors  Function
1      1       0     bitNor
1      1       0     bitXor
2      2       0    allEvenBits
2      2       0    byteSwap
3      3       0  multFiveEighths
3      3       0  conditional
3      3       0  isGreater
3      3       0  localAnd
4      4       0  logicalNeg
2      2       0  float_abs
4      4       0  float_i2f
4      4       0  float_half
Total points: 32/32
```

## 完整检查情况

使用`./driver.pl`命令对实现代码进行完全检查和打分，并将检查情况截图粘贴在下方。



```
root@root: ~/lab1-handout

4. Compiling and running './btest -g -r 2' to determine performance score.
gcc -O -Wall -m32 -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      3     bitNor
1      1       0     2      8     bitXor
2      2       0     2      9     allEvenBits
2      2       0     2     20     byteSwap
3      3       0     2      6     multFiveEighths
3      3       0     2      7     conditional
3      3       0     2     12     isGreater
3      3       0     2     17     localAnd
4      4       0     2      7     logicalNeg
2      2       0     2      6     float_abs
4      4       0     2     25     float_i2f
4      4       0     2     23     float_half

Score = 56/56 [32/32 Corr + 24/24 Perf] (143 total operators)
root@root:~/lab1-handout#
```



## 思考与体会

对于一些逻辑运算的技巧要熟练掌握，不会就多翻书，多看 int 和 float 的内存分布。