

# 哈尔滨工业大学

# 实验报告

## 实 验（二）

题 目 DataLab 数据表示

专 业 计算机类

学 号 1170300821

班 级 1703008

学 生 罗瑞欣

指 导 教 师 郑贵滨

实 验 地 点 G712

实 验 日 期 2018.10.08

## 计算机科学与技术学院

# 目 录

<b>第 1 章 实验基本信息 .....</b>	<b>- 4 -</b>
1.1 实验目的.....	- 4 -
1.2 实验环境与工具.....	- 4 -
1.2.1 硬件环境.....	- 4 -
1.2.2 软件环境.....	- 4 -
1.2.3 开发工具.....	- 4 -
1.3 实验预习.....	- 4 -
<b>第 2 章 实验环境建立 .....</b>	<b>- 5 -</b>
2.1 UBUNTU 下 CODEBLOCKS 安装（5 分） .....	- 5 -
2.2 64 位 UBUNTU 下 32 位运行环境建立（5 分） .....	- 5 -
<b>第 3 章 C 语言的位操作指令 .....</b>	<b>- 6 -</b>
3.1 逻辑操作（1 分） .....	- 6 -
3.2 无符号数位操作（2 分） .....	- 6 -
3.3 有符号数位操作（2 分） .....	- 6 -
<b>第 4 章 汇编语言的位操作指令 .....</b>	<b>- 7 -</b>
4.1 逻辑运算(1 分).....	- 7 -
4.2 无符号数左右移（2 分） .....	- 7 -
4.3 有符号左右移（2 分） .....	- 7 -
4.4 循环移位（2 分） .....	- 7 -
4.5 带进位位的循环移位（2 分） .....	- 7 -
4.6 测试、位测试 BTx（2 分） .....	- 7 -
4.7 条件传送 CMOVxx（2 分） .....	- 7 -
4.8 条件设置 SETCxx（1 分） .....	- 8 -
4.9 进位位操作（1 分） .....	- 8 -
<b>第 5 章 BITS 函数实验与分析 .....</b>	<b>- 9 -</b>
5.1 函数 LSBZERO 的实现及说明 .....	- 9 -
5.2 函数 BYTE NOT 的实现及说明函数 .....	- 9 -
5.3 函数 BYTE XOR 的实现及说明函数 .....	- 10 -
5.4 函数 LOGICAL AND 的实现及说明函数 .....	- 10 -
5.5 函数 LOGICAL OR 的实现及说明函数 .....	- 11 -
5.6 函数 ROTATE LEFT 的实现及说明函数 .....	- 11 -
5.7 函数 PARITY CHECK 的实现及说明函数 .....	- 12 -
5.8 函数 MUL2OK 的实现及说明函数 .....	- 12 -
5.9 函数 MULT3DIV2 的实现及说明函数 .....	- 13 -
5.10 函数 SUBOK 的实现及说明函数 .....	- 13 -

5.11 函数 ABSVAL 的实现及说明函数 .....	- 14 -
5.12 函数 FLOAT_ABS 的实现及说明函数 .....	- 15 -
5.13 函数 FLOAT_F2I 的实现及说明函数 .....	- 15 -
5.14 函数 XXXX 的实现及说明函数（CMU 多出来的函数-不加分） .....	- 15 -
<b>第 6 章 总结 .....</b>	<b>- 16 -</b>
10.1 请总结本次实验的收获 .....	- 16 -
10.2 请给出对本次实验内容的建议 .....	- 16 -
<b>参考文献 .....</b>	<b>- 17 -</b>

## 第 1 章 实验基本信息

### 1.1 实验目的

熟练掌握计算机系统的数据表示与数据运算

通过 C 程序深入理解计算机运算器的底层实现与优化

掌握 Linux 下 makefile 与 GDB 的使用

### 1.2 实验环境与工具

#### 1.2.1 硬件环境

X64 CPU; 2GHz; 2G RAM; 256GHD Disk 以上

#### 1.2.2 软件环境

Windows7 64 位以上; VirtualBox/Vmware 11 以上; Ubuntu 16.04 LTS 64 位/  
优麒麟 64 位;

#### 1.2.3 开发工具

Visual Studio 2010 64 位以上; CodeBlocks; vi/vim/gpedit+gcc

### 1.3 实验预习

上实验课前, 必须认真预习实验指导书 (PPT 或 PDF)、了解实验的目的、实验环境与软硬件工具、实验操作步骤, 复习与实验有关的理论知识。

写出 C 语言下的位操作指令:

逻辑、无符号、有符号

写出汇编语言下的位操作指令:

逻辑运算、无符号、有符号、测试、位测试 BTx、条件传送 CMOVxx、条件设置 SETxx、进位位(CF)操作

## 第 2 章 实验环境建立

### 2.1 Ubuntu 下 CodeBlocks 安装 (5 分)

CodeBlocks 运行界面截图：编译、运行 hellolinux.c

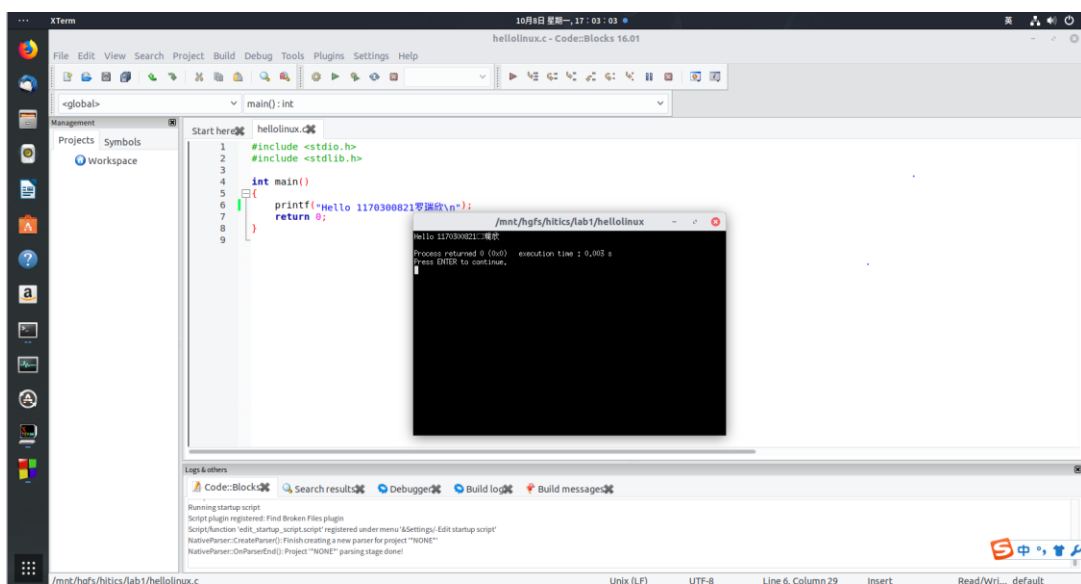


图 2-1 Ubuntu 下 CodeBlocks 截图

### 2.2 64 位 Ubuntu 下 32 位运行环境建立 (5 分)

在终端下，用 gcc 的 32 位模式编译生成 hellolinux.c。执行此文件。

Linux 及终端的截图。

```
1170300821@luoruixin:~/hitics/lab1$ gcc -m32 hellolinux.c
1170300821@luoruixin:~/hitics/lab1$ ./hellolinux
Hello 1170300821 罗瑞欣
```

图 2-2 32 位运行环境建立

## 第 3 章 C 语言的位操作指令

写出 C 语言例句

### 3.1 逻辑操作 (1 分)

```
!a;  
a&&b;  
a||b;
```

### 3.2 无符号数位操作 (2 分)

```
~a;  
a&b;  
a|b;  
a^b;  
b<<2;  
b>>2;
```

### 3.3 有符号数位操作 (2 分)

```
~a;  
a&b;  
a|b;  
a^b;  
b<<2;  
b>>2;
```

## 第 4 章 汇编语言的位操作指令

写出汇编语言例句

### 4.1 逻辑运算(1 分)

Mul %cl

And %cl %dl

Xor %cl %dl

Or %cl %dl

Not %cl

### 4.2 无符号数左右移 (2 分)

Shlq \$2 %rcx

shrq \$2 %rcx

### 4.3 有符号左右移 (2 分)

Salq \$2 %rcx

salq \$2 %rcx

### 4.4 循环移位 (2 分)

Rolq \$2 %rcx

rorq \$2 %rcx

### 4.5 带进位位的循环移位 (2 分)

Rclq \$2 %rcx

rcrq \$2 %rcx

### 4.6 测试、位测试 BTx (2 分)

Bt \$5 (%rcx)

#### 4.7 条件传送 CMOV<sub>xx</sub> (2 分)

Cmovaq (%rdx) (%rcx)

#### 4.8 条件设置 SET<sub>xx</sub> (1 分)

Setb %rcx

#### 4.9 进位位操作 (1 分)

stc

clc

注：进位 CF



## 第 5 章 BITS 函数实验与分析

每题 8 分，总分不超过 80 分

语法检查命令 `./dlc -e bits.c` 的结果截图：

```
1170300821@luoruixin:~/hitics/lab2-handout$ ./dlc -e bits.c
Multiple input files defined, using `bits.c'
1170300821@luoruixin:~/hitics/lab2-handout$
```

**要求：每个函数不可以有非法运算符、函数调用等，否则相应函数会被扣分**

### 5.1 函数 `lsbZero` 的实现及说明

程序如下：

```
int lsbZero(int x) {
    return x>>1<<1;
}
```

`btest` (命令 `./btest -f lsbZero`) 的结果截图：

设计思想：将 `x` 右移一位后左移一位，最低位自动补 0。

### 5.2 函数 `byteNot` 的实现及说明函数

程序如下：

```
int byteNot(int x, int n) {
    return (0xff<<(n<<3))^x;
}
```

`btest` 截图：

```
1170300821@luoruixin:~/hitics/lab2-handout$ ./btest -f byteNot
Score   Rating  Errors  Function
  2       2       0      byteNot
Total points: 2/2
```

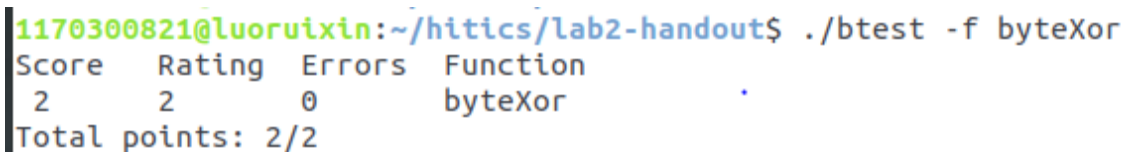
设计思想：n 左移三位，即乘八，将移动的位数变为一字节。再将 0xff 八位移动和 x 异或。x 中与 0 异或的位保持原状，与 1 异或的位取反。

### 5.3 函数 byteXor 的实现及说明函数

程序如下：

```
int byteXor(int x, int y, int n) {  
    n=n<<3;  
    x=(x>>n)&0xff;  
    y=(y>>n)&0xff;  
    return !(x^y);  
}
```

btest 截图：



```
1170300821@luoruixin:~/hitics/lab2-handout$ ./btest -f byteXor  
Score   Rating  Errors  Function  
2       2       0       byteXor  
Total points: 2/2
```

设计思想：

N 左移 3 位变为 8 倍，以对应字节。两个数右移后与 0xff 相与，取出目标字节。相异或后，若相等，则位 0，不相等就不为 0。两次取反转化为布尔代数即为所求。

### 5.4 函数 logicalAnd 的实现及说明函数

程序如下：

```
int logicalAnd(int x, int y) {  
    return !((!x)&(!y));  
}
```

btest 截图：

```
1170300821@luoruixin:~/hitics/lab2-handout$ ./btest -f logicalAnd
Score  Rating  Errors  Function
  3      3      0      logicalAnd
Total points: 3/3
```

设计思想：x 和 y 两次取非变为布尔代数，若为 0 还是 0，补位 0 变成 1。然后两个数相与，再两次取非变为布尔代数，若为 0 还是 0，补位 0 变成 1。

## 5.5 函数 logicalOr 的实现及说明函数

程序如下：

```
int logicalOr(int x, int y) {
    return !((!x)|(!y));
}
```

btest 截图：

```
1170300821@luoruixin:~/hitics/lab2-handout$ ./btest -f logicalOr
Score  Rating  Errors  Function
  3      3      0      logicalOr
Total points: 3/3
```

设计思想：x 和 y 两次取非变为布尔代数，若为 0 还是 0，补位 0 变成 1。然后两个数相或，再两次取非变为布尔代数，若为 0 还是 0，补位 0 变成 1。

## 5.6 函数 rotateLeft 的实现及说明函数

程序如下：

```
int rotateLeft(int x, int n) {
    int t, mark;
    mark = ~((~0) >> n << n);
    t = x >> (32 + ((~n) + 1));
    x = x << n;
    t = t & mark;
    return x | t;
}
```

btest 截图:

```
1170300821@luoruixin:~/hitics/lab2-handout$ ./btest -f rotateLeft
Score   Rating  Errors  Function
   3     3      0    rotateLeft
Total points: 3/3
```

设计思想: 将 $\sim 0$ 即  $0xffffffff$  左移  $n$  位, 再右移  $n$  位, 使得右边  $n$  个 0, 左边全是 1。再取反为 `mark`, 用于处理负数逻辑右移的情况。右移  $32-n$  位, 提取出前  $n$  位为 `t`。x 左移  $n$  位, 空出后  $n$  位。`t` 与 `mark` 与, 消除负数逻辑右移的影响。x 和 `t` 相或得到结果。

## 5.7 函数 parityCheck 的实现及说明函数

程序如下:

```
int parityCheck(int x) {
    x = (x >> 16)^x;
    x = (x >> 8)^x;
    x = (x >> 4)^x;
    x = (x >> 2)^x;
    x = (x >> 1)^x;
    return x&1;
}
```

btest 截图:

```
1170300821@luoruixin:~/hitics/lab2-handout$ ./btest -f parityCheck
Score   Rating  Errors  Function
   4     4      0    parityCheck
Total points: 4/4
```

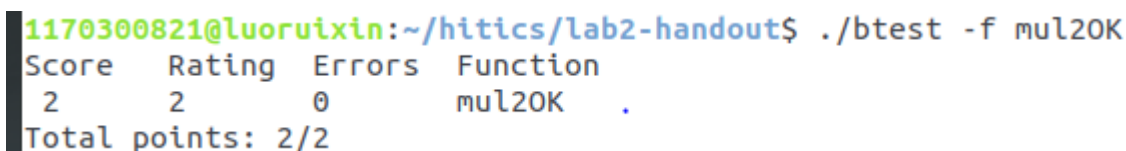
设计思想: 1 和 1 异或为 0, 一下子消去了两个 1。一个数减去 2 奇偶性不变。把 `x` 对半, 异或后得到的结果 1 的个数的奇偶性与 `x` 一致。重复此过程, 把 `x` 折叠, 最后得到只剩一位。

## 5.8 函数 mul2OK 的实现及说明函数

程序如下：

```
int mul2OK(int x) {
    x=(x>>30)^(x>>31);
    return (~x)&1;
}
```

btest 截图：



```
1170300821@luoruixin:~/hitics/lab2-handout$ ./btest -f mul2OK
Score  Rating  Errors  Function
  2      2      0      mul2OK      .
Total points: 2/2
```

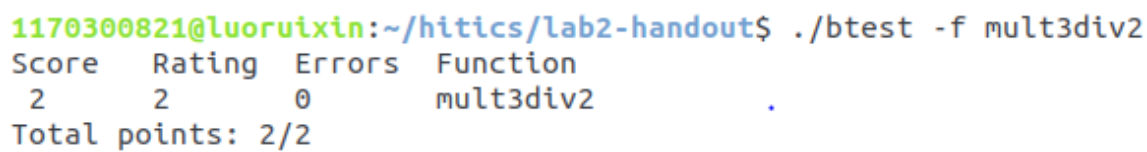
设计思想：若  $x$  有乘 2 后溢出的风险，则  $x$  的第 31 位和第 32 位必定不同，将  $x$  移位 30 和 31 位后（逻辑移位无影响），异或后取反并与 1（相当于同或），返回值即为所求。

## 5.9 函数 mult3div2 的实现及说明函数

程序如下：

```
int mult3div2(int x) {
    x=(x<<1)+x;
    return (x+((x>>31)&1))>>1;
}
```

btest 截图：



```
1170300821@luoruixin:~/hitics/lab2-handout$ ./btest -f mult3div2
Score  Rating  Errors  Function
  2      2      0      mult3div2    .
Total points: 2/2
```

设计思想：分解为先乘以三，再除以二。要点为  $x$  乘三为负数，除以二时向 0 舍入。处理方法为提取出  $x*3$  ( $x+x<<1$ ) 的符号位 1，加上  $x*3$ ，可实现向 0 舍入。

## 5.10 函数 subOK 的实现及说明函数

程序如下：

```
int subOK(int x, int y) {
    int z=((x+((~y)+1))>>31)&1;
    x=(x>>31)&1;
    y=(y>>31)&1;
    return !(((!x)&y&z)+(x&(!y)&(!z)));
}
```

btest 截图：

```
1170300821@luoruixin:~/hitics/lab2-handout$ ./btest -f subOK
Score  Rating  Errors  Function
  3      3      0      subOK
Total points: 3/3
```

设计思想：用  $y$  取反后加一表示  $-y$ ，分别取  $x$ 、 $y$ 、 $x-y$  的符号位。当其符号位出现 100（负减正得正，小于 TMin）或 011（正减负得负，大于 Tmax）的情况，就出现溢出。相加（或）后取反，即为所求。

## 5.11 函数 absVal 的实现及说明函数

程序如下：

```
int absVal(int x) {
    return (x^(x>>31))+((x>>31)&1);
}
```

btest 截图：

```
1170300821@luoruixin:~/hitics/lab2-handout$ ./btest -f absVal
Score  Rating  Errors  Function
  4      4      0      absVal
Total points: 4/4
```

设计思想：。本题关键在于用符号位创造补码运算，来实现负数转化为正数。主要依靠逻辑右移。 $x$  若为负数，则  $x>>31$  为 0xffffffff， $x$  与其异或即为取反；同时 0xffffffff（即  $x>>31$ ）和 1 相与，即为 1；两者相加即为负数变为正数（取补码）。若  $x$  为负数或 0，则  $x>>31$  位 0， $x$  与其异或为自身； $x>>31$  与 1 相与

为 0,; 两者相加为  $x$ 。

## 5.12 函数 float\_abs 的实现及说明函数

程序如下:

```
unsigned float_abs(unsigned uf) {  
    if (!(((uf >> 23 & 0xFF) == 0xFF) && ((uf & 0x7FFFFFFF) != 0)))  
        uf = uf & (~ (1 << 31));  
    return uf;  
}
```

btest 截图:

```
1170300821@luoruixin:~/hitcs/lab2-handout$ ./btest -f float_abs  
Score   Rating  Errors  Function  
2       2       0      float_abs  
Total points: 2/2
```

设计思想: 先检验  $uf$  是否为 NAN: 将  $uf$  右移 23 位, 检验阶码是否全为 1, 再检验  $uf$  后 23 位尾码是否不为 0。若  $uf$  不是 NAN, 则  $uf$  将  $uf$  符号位置零, 其他不变; 若  $uf$  为 NAN, 则不进行操作。

## 5.13 函数 float\_f2i 的实现及说明函数

程序如下:

btest 截图:

设计思想:

## 5.14 函数 XXXX 的实现及说明函数 (CMU 多出来的函数-不加分)

## 第 6 章 总结

### 10.1 请总结本次实验的收获

- (1) 熟练掌握计算机系统的数据表示与数据运算
- (2) 通过 C 程序深入理解计算机运算器的底层实现与优化
- (3) 掌握 Linux 下 makefile 与 GDB 的使用
- (4) 掌握 Linux 下使用 vim 编辑
- (5) 熟练位运算，了解 C 语言逻辑运算和一些简单函数的基本原理
- (6) 掌握部分基本汇编指令机器应用
- (7) 进一步熟练 Linux 的指令操作
- (8) 掌握 Linux 不同语言不同位数的运行环境建立

### 10.2 请给出对本次实验内容的建议

练习题部分考察内容相互重合，例如 logicalOr 和 logicalAnd；部分习题与上次家庭作业重合，如 parityCheck。可以增加部分题目的限制，使其目标更加贴近机器的运算。

注：本章为酌情加分项。



## 参考文献

### 为完成本次实验你翻阅的书籍与网站等

- [1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.
- [2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A 集[C]. 北京：中国科学出版社，1999.
- [3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. <http://www.ie.nthu.edu.tw/info/ie.newie.htm>（Big5）.
- [4] 湛颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.
- [5] KANAMORI H. Shaking Without Quaking[J]. Science, 1998, 279 (5359): 2063-2064.
- [6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science , 1998 , 281 : 331-332[1998-09-23]. <http://www.sciencemag.org/cgi/collection/anatmorp>.