

**实验报告**

**实 验（二）**

题 目 DataLab 数据表示

专 业

学　　 号

班　　 级

学 生

指 导 教 师

实 验 地 点

实 验 日 期

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 4 -](#_Toc495997321)

[1.1 实验目的 - 4 -](#_Toc495997322)

[1.2 实验环境与工具 - 4 -](#_Toc495997323)

[1.2.1 硬件环境 - 4 -](#_Toc495997324)

[1.2.2 软件环境 - 4 -](#_Toc495997325)

[1.2.3 开发工具 - 4 -](#_Toc495997326)

[1.3 实验预习 - 4 -](#_Toc495997327)

[第2章 实验环境建立 - 5 -](#_Toc495997328)

[2.1 Ubuntu下CodeBlocks安装（5分） - 5 -](#_Toc495997329)

[2.2 64位Ubuntu下32位运行环境建立（5分） - 5 -](#_Toc495997330)

[第3章 C语言的位操作指令 - 6 -](#_Toc495997331)

[3.1 逻辑操作（1分） - 6 -](#_Toc495997332)

[3.2 无符号数位操作（2分） - 6 -](#_Toc495997333)

[3.3 有符号数位操作（2分） - 6 -](#_Toc495997334)

[第4章 汇编语言的位操作指令 - 7 -](#_Toc495997335)

[4.1 逻辑运算(1分) - 7 -](#_Toc495997336)

[4.2无符号数左右移（2分） - 7 -](#_Toc495997337)

[4.3有符号左右移（2分） - 7 -](#_Toc495997338)

[4.4循环移位（2分） - 7 -](#_Toc495997339)

[4.5带进位位的循环移位（2分） - 7 -](#_Toc495997340)

[4.6测试、位测试BTx（2分） - 7 -](#_Toc495997341)

[4.7条件传送CMOVxx（2分） - 7 -](#_Toc495997342)

[4.8条件设置SETCxx（1分） - 7 -](#_Toc495997343)

[4.9进位位操作（1分） - 7 -](#_Toc495997344)

[第5章 BITS函数实验与分析 - 8 -](#_Toc495997345)

[5.1 函数lsbZero的实现及说明 - 8 -](#_Toc495997346)

[5.2 函数byteNot的实现及说明函数 - 8 -](#_Toc495997347)

[5.3 函数byteXor的实现及说明函数 - 8 -](#_Toc495997348)

[5.4 函数logicalAnd的实现及说明函数 - 8 -](#_Toc495997349)

[5.5 函数logicalOr的实现及说明函数 - 8 -](#_Toc495997350)

[5.6 函数rotateLeft的实现及说明函数 - 8 -](#_Toc495997351)

[5.7 函数parityCheck的实现及说明函数 - 8 -](#_Toc495997352)

[5.8 函数mul2OK的实现及说明函数 - 9 -](#_Toc495997353)

[5.9 函数mult3div2的实现及说明函数 - 9 -](#_Toc495997354)

[5.10 函数subOK的实现及说明函数 - 9 -](#_Toc495997355)

[5.11 函数absVal的实现及说明函数 - 9 -](#_Toc495997356)

[5.12 函数float\_abs的实现及说明函数 - 9 -](#_Toc495997357)

[5.13 函数float\_f2i的实现及说明函数 - 9 -](#_Toc495997358)

[5.14函数XXXX的实现及说明函数（CMU多出来的函数-不加分） - 9 -](#_Toc495997359)

[第6章 总结 - 10 -](#_Toc495997360)

[10.1 请总结本次实验的收获 - 10 -](#_Toc495997361)

[10.2 请给出对本次实验内容的建议 - 10 -](#_Toc495997362)

[参考文献 - 11 -](#_Toc495997363)

# 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

Intel x86-64

### 1.2.2 软件环境

Windows 10 家庭单语言版, Virtualbox Ubuntu

### 1.2.3 开发工具

CodeBlocks for Linux

## 1.3 实验预习

**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**

# 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

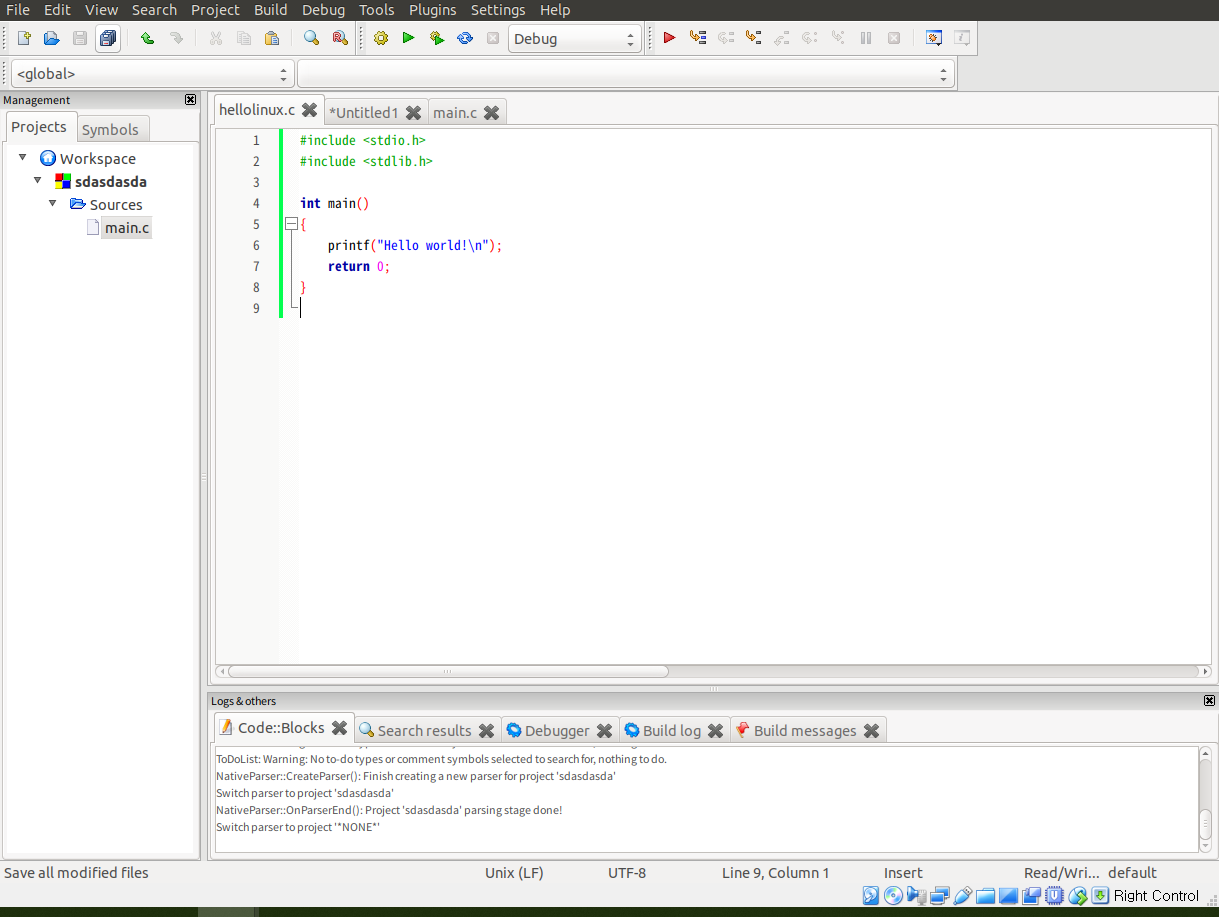


图2-1 Ubuntu下CodeBlocks截图

## 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。

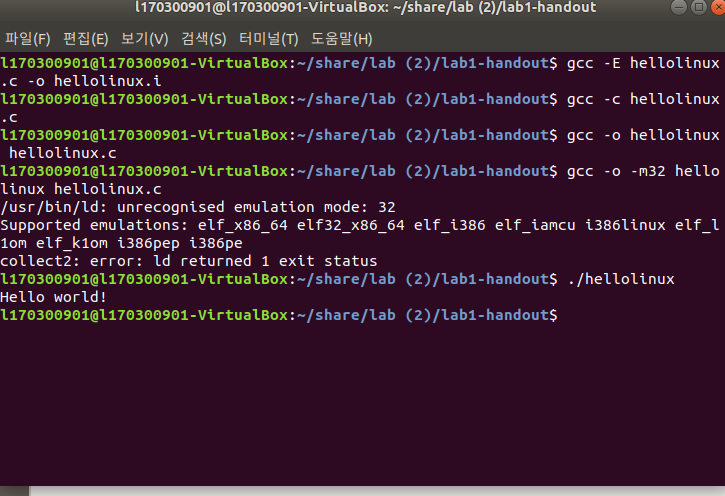


图2-2 Ubuntu与Windows共享目录截图

# 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）

& 按位与运算，如：1010 & 1100 = 1000

| 按位或运算，如：1010 & 1100 = 1110

~ 非运算，如：a = 1010; ~a = 0101

<<左移，如：1110 << 1 = 1100

>>右移，如：1000 >> 1 = 0100

## 3.2 无符号数位操作（2分）

指整个机器字长的全部二进制位均表示数值位，相当于数的绝对值。若机器字长为n+1位，则数值表示为：

X=X0X1X2...Xn 其中Xi={0,1},0<=i<=n 即X0\*2^n + X1\*2^(n-1） + X2\*2^(n-2） + ... + Xn-1\*2 + Xn

数值范围是 0≤X≤2^(n+1） - 1

例如：1111表示15。

## 3.3 有符号数位操作（2分）

所谓原码就是二进制定点表示法，即最高位为符号位，“0”表示正，“1”表示负，其余位表示数值的大小。

反码表示法规定：正数的反码与其原码相同；负数的反码是对其原码逐位取反，但符号位除外。

原码10010= 反码11101 （10010，1为符号码，故为负）

(11101) 二进制= -13 十进制

补码表示法规定：正数的补码与其原码相同；负数的补码是在其反码的末位加1。

# 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)

"∨" 表示"或"

"∧" 表示"与".

"┐"表示"非".

"=" 表示"等价".

1和0表示"真"和"假"

(还有一种表示,"+"表示"或", "·"表示"与"）

## 4.2无符号数左右移（2分）

## SAL算术移位指令在执行时，实际上把操作数看成有符号数进行移位，最高位符号位移入CF，但本身保持原值；其余位顺序左移，次高位被舍弃。 SHL逻辑移位指令在执行时，实际上把操作数看成无符号数进行移位，所有位顺序左移，最高位移入CF。

## 4.3有符号左右移（2分）

首先  
负数的二进制码=正数的二进制的补码=正数的二进制码的反码+1  
所以  
-15的二进制码 = 00001111的补码 = ~(00001111)+1 = 11110000+1 = 11110001  
>> 运算，负数右移且符号位不移的时候，低位去掉，高位(符号位不变，符号位以后的高位)补1  
>>> 运算，负数右移且符号位也移的时候，低位去掉，高位(符号位前面的高位)补0  
所以，-15的右移过程应该是  
11110001 -> 11111100(-4的补码)，怎么知道是-4的补码？把补码转成正数的二进制码就知道了，转码过程和补码的计算过程刚好互逆  
正数的二进制码=(正数的二进制补码-1)取反  
所以  
11111100的正数的二进制码 = ~(11111100-1) = ~(11111011) = 00000100(正4)

## 4.4循环移位（2分）

循环移位就像1001 1100 0000 0000左移一位变成0011 1000 000 0001,右移一位0100 1110 0000 0000不考虑移位后的数据是否溢出,向左移后移出最高位补在后面,同样右移补在前面,.

算数移位左移一位1011 1000 0000 0000相当于乘2,因为是乘2所以最高位是不变的,因为最高位代表正数负数,右移一位1000 1110 0000 0000 s相当于除2

逻辑移位左移一位0011 1000 0000 0000直接向左移,最高位不要了,向右移0100 1110 0000 0000直接向右移最高位被0

## 4.5带进位位的循环移位（2分）

## 循环左移是指寄存器内的东西移动，如AH循环左移，那么移动的位数总共是8位。

## 带进位循环左移是指CY寄存器的东西也参与到移动中来。

## 举例：（为说明问题，用1-9的数字来说，其实都是0和1）

## 假定AH=12345678，cy=9

## 循环左移后AH：23456781

## 带进位循环左移AH：23456789

## 4.6测试、位测试BTx（2分）

## 你纠结这个干嘛？记住8000，还是8000 0000毫无意义。 搞明白测试最高位(或某位)的原理才是正确的作法。 ax (16位) 16进制 : 00 00 H，　2进制 : 0000 0000 0000 0000B eax (32位) 16进制: 00 00 00 00 H, 2进制 : 0000 0000 0000 0000 0000 0000 0000 0000B 判断某位，就把2进制的该位置1. 对16位寄存器的判断来说，就是1000 0000 0000 0000B，即8000H 对32位寄存器，当然就是 8 000 0000H了。

## 4.7条件传送CMOVxx（2分）

## 4.8条件设置SETCxx（1分）

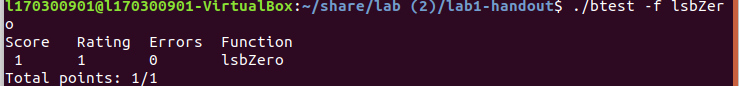
## 4.9进位位操作（1分）

# 第5章 BITS函数实验与分析

每题8分，总分不超过80分

## 5.1 函数lsbZero的实现及说明

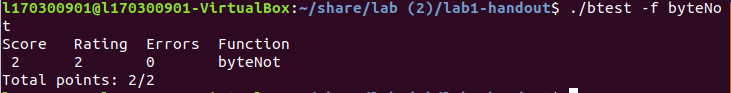
程序如下：



设计思想：

## 5.2 函数byteNot的实现及说明函数

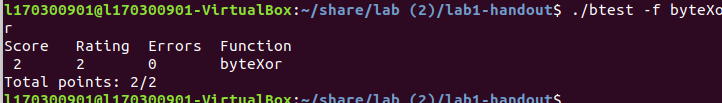
程序如下：



设计思想：

## 5.3 函数byteXor的实现及说明函数

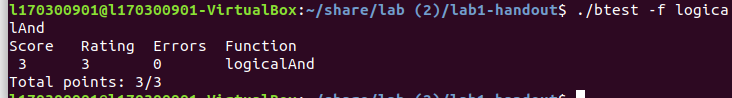
程序如下：



设计思想：

## 5.4 函数logicalAnd的实现及说明函数

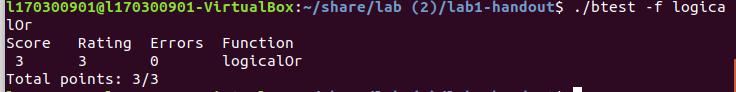
程序如下：



设计思想：

## 5.5 函数logicalOr的实现及说明函数

程序如下：



设计思想：

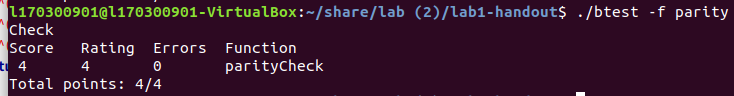
## 5.6 函数rotateLeft的实现及说明函数

程序如下：

设计思想：

## 5.7 函数parityCheck的实现及说明函数

程序如下：

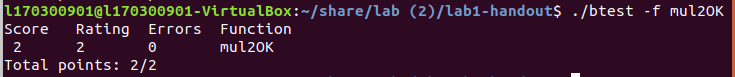


设计思想：

有点类似对折的相法。 右移一次是第0位和第一位异或。 第二句右移两位是第0位到第4位的异或

## 5.8 函数mul2OK的实现及说明函数

程序如下：

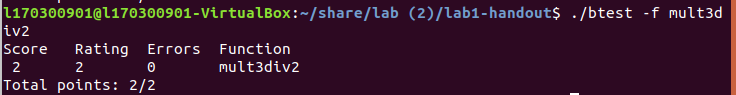


设计思想：

X和2X 的符号位异或的方式将德到结束，因为不能使用！故采用&1的方式得到符号位，然后将两个符号位异或再与1异或得到结果。

## 5.9 函数mult3div2的实现及说明函数

程序如下：

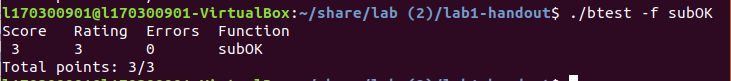


设计思想：

先乘3再除以2--按位计算，先向左移动一位即乘2再加上本身即乘三得到数m，将m右移一位即除以2。

## 5.10 函数subOK的实现及说明函数

程序如下：

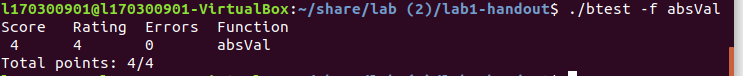


设计思想：

先提出来两个数的符号位。然后把两个数加起来。如果加起来的数的符号位和那两个数的符号位相同，才不溢出。

## 5.11 函数absVal的实现及说明函数

程序如下：

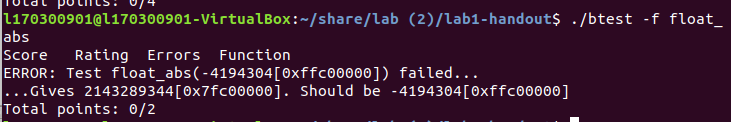


设计思想：

先通过把这个数右移动提取最高位，如果是0数位不变，如果是1数位变成～x+1。

## 5.12 函数float\_abs的实现及说明函数

程序如下：

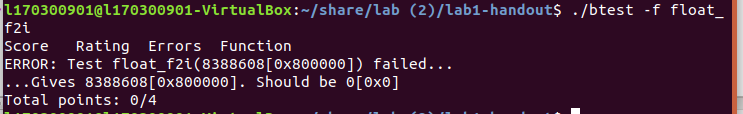


设计思想：

首先这个函数中输入的参数是无符号型整数，不是浮点数，所以所有的操作都是整数，只是假设这个无符号数表示一个浮点数。浮点数最高位是符号位，代表正负。求绝对值，就是将负数变成正数，正数不变，所以只需要将最高位置为0就可以了，因此左移一位再右移一位就可以。但是还有一种情况，就是这32个0、1能否表示一个浮点数，如果不是浮点数就直接返回原值，就需要进行判断，判断条件就应该是最低23位（小数位）都是0，且指数位都为1（第23~30位），所以先判断一下是否为浮点数，如果不是直接返回，如果是返回左移一位再右移一位的数字。

## 5.13 函数float\_f2i的实现及说明函数

程序如下：



设计思想：

x为uf首位置0的结果，也就是绝对值，y为阶码部分，对阶码进行判断，如果不是全1，或者uf是正负无穷，就返回绝对值x，否则直接返回uf。

## 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

## 我是韩国留学生. 因此用中文学习这个课程是非常困难的. 但是很多中国朋友告诉我和帮助我了解了课堂内容 所以我提高了很多汉语水平,也了解了很多linux. 对我来说,这似乎是一次很好的经验,真的很幸福

## 10.2 请给出对本次实验内容的建议

我是留学生 。请多多关照。 谢谢老师。

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] 谌颖. 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.