

实验报告

实 验（二）

题 目 DataLab 数据表示

专 业 计算机科学与技术

学　　 号 1171000410

班　　 级 1703005

学 生 强文杰

指 导 教 师 吴锐

实 验 地 点 G712

实 验 日 期 2018.9.30

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

X64CPU;

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 16.04LTS 64位/优麒麟64位

### 1.2.3 开发工具

Gcc , Codeblocks

## 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SETCxx

进位位操作

# 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

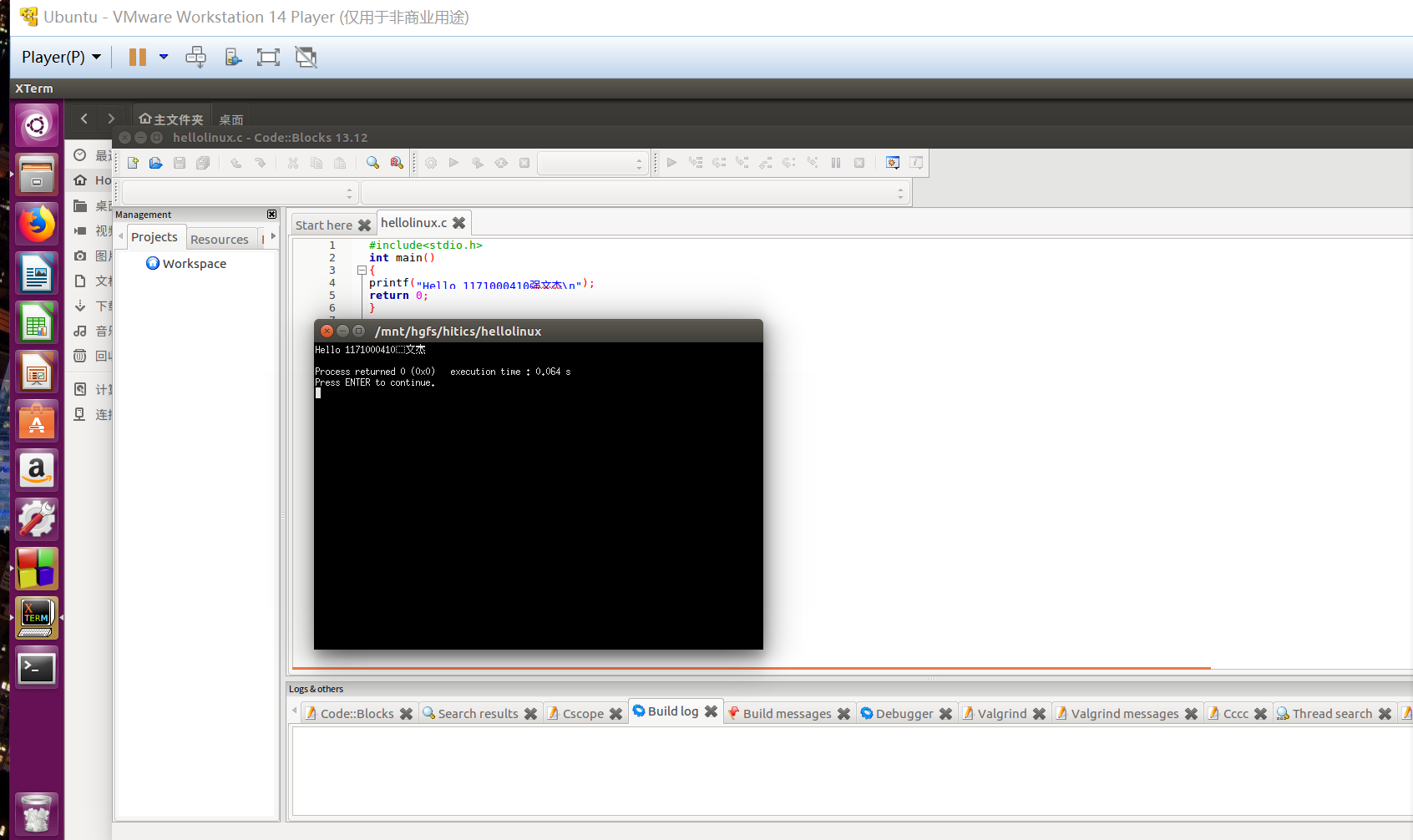


图2-1 Ubuntu下CodeBlocks截图

## 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。

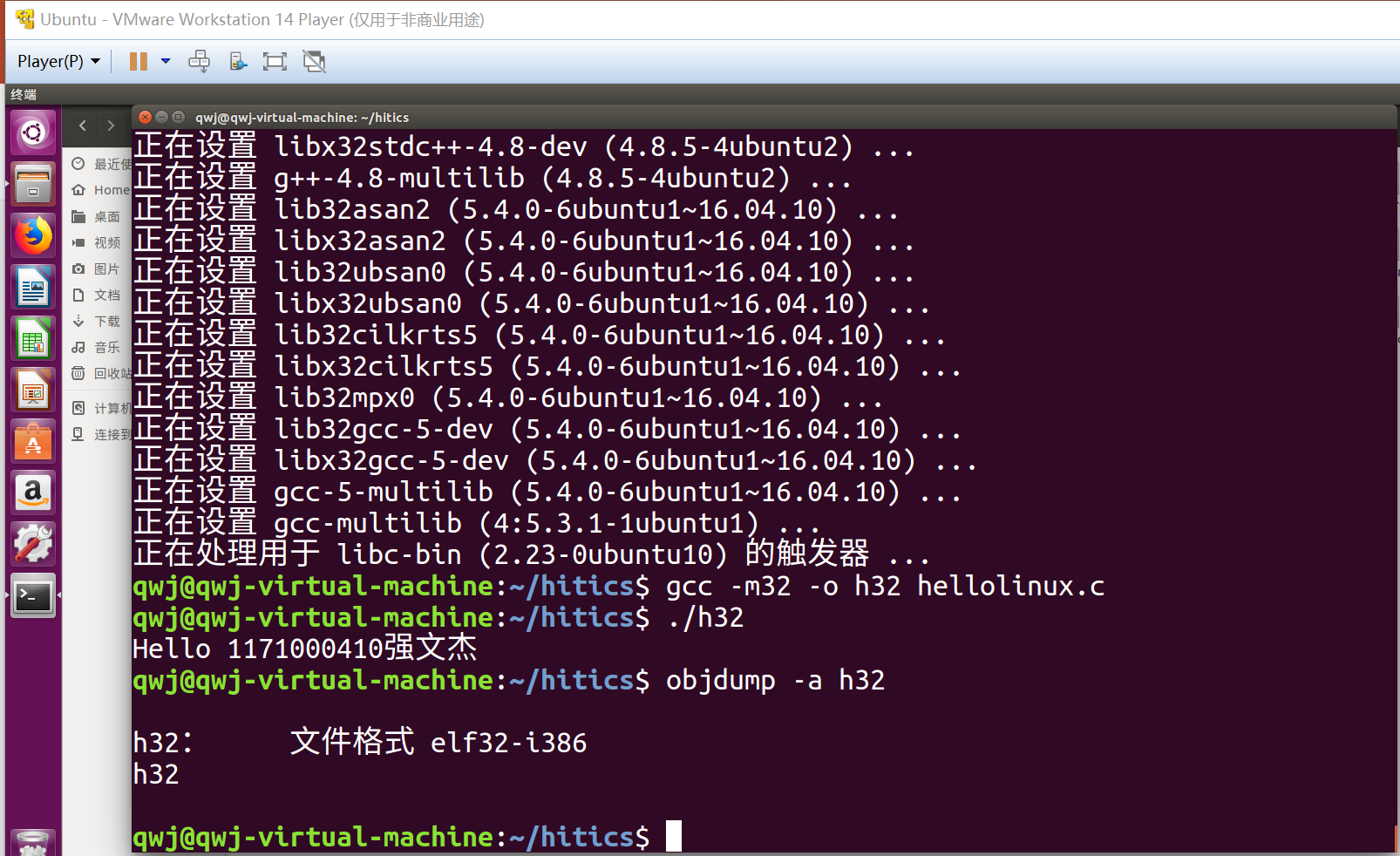


图2-2 32位运行环境建立

# 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）

||对应OR：0x69||0x55=0x01

&&对应AND：0x69&&0x55=0x01

！对应NOT：!0x41=0x00

## 3.2 无符号数位操作（2分）

取反：~(010000001)=10111110

与：(01101001)&(01010101)=01000001

或：(01101001)|(01010101)=01111101

异或：01010101^001100010=01101011

移位：对于无符号数，左移右移都是逻辑移位，即左移低位空出的补0，右移高位空出的补0。

例：无符号数x=10010101;

x>>4=00001001

## 3.3 有符号数位操作（2分）

取反：~(010000001)=10111110

与：(01101001)&(01010101)=01000001

或：(01101001)|(01010101)=01111101

异或：01010101^001100010=01101011

移位：对于有符号数，左移多出的会移进符号位，右移是算术右移，即空出的高位补符号位。

例：有符号数x=10010101;

x>>4=11111001

# 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)

AND SRC,DEST ；将操作数相与，返回DEST

OR SRC,DEST ; 将操作数相或，返回DEST

NOT SRC; 将操作数SRC中每位取反

XOR SRC,DEST;将操作数相异或，并返回给DEST

TEST SRC,DEST;将操作数相与，影响状态标志，主要用于给数据转移指令传递状态标志。

例子：假设寄存器%rax的值为x，%rdx的值为y

AND %rax，%rdx指令执行后，%rdx的值为x&y

## 4.2无符号数左右移（2分）

## SHL k，DEST；将操作数DEST左移k位

SHR k，DEST；将操作数DEST右移k位

例如：假设AL= 01001101B

SHL 1，AL指令执行后AL=10011010B

## 4.3有符号左右移（2分）

SAL k，DEST；将操作数DEST左移k位

SAD k，DEST；将操作数DEST右移k位

例如 假设AL=10011001B

SAD 1，AL指令执行后AL=11001100B

## 4.4循环移位（2分）

ROL k，DEST; 把操作数的低位部分向高位方向循环移动CL/imm指定的位数,空出的低位部分由移出的高位部分来填充,同时,移出的高位部分仍然会存放在CF中;如果是循环左移N位,那么,就空出N个低位,移出N个高位,然后,把移出的这N个高位按照移出的顺序依次填入空出的N个低位中,同时,CF中只保存最后一次移出的那一位的内容

ROR k，DEST; 把操作数的高位部分向低位方向循环移动CL/imm指定的位数,空出的高位部分由移出的低位部分来填充,同时,移出的低位部分仍然会存放在CF中;如果是循环右移N位,那么,就空出N个高位,移出N个低位,然后,把移出的这N个低位按照移出的顺序依次填入空出的N个高位中,同时,CF中只保存最后一次移出

的那一位的内容

例如：假设当前，[AL](https://www.baidu.com/s?wd=AL&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)=01010011B，CF=1，则执行指令

ROL 1，AL 后，[AL](https://www.baidu.com/s?wd=AL&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)=10100110B，CF=0

## 4.5带进位位的循环移位（2分）

## RCL k，DEST；把操作数的低位部分向高位方向循环移动CL/imm指定的位数,每向左移动一位,RCL指令都会先把CF的原有值填充到空出的最低位上,再把移出的最高位存放到CF中;这样循环左移N位之后,CF中保存的仍然是最后一次移出的那一位的内容

## RCR k，DEST；把操作数的高位部分向低位方向循环移动CL/imm指定的位数,每向右移动一位,RCL指令都会先把CF的原有值填充到空出的最高位上,再把移出的最低位存放到CF中;这样循环右移N位之后,CF中保存的仍然是最后一次移出的那一位的内容

假设当前AL=01010011B，CF=1

RCL 1, AL后RCL=10100111，CF=0

RCR 1, AL后RCR=10101001，CF=1

## 4.6测试、位测试BTx（2分）

BT（位测试）

写法：BT REG16/MEM16,REG16/IMM8;或BTREG32/MEM32,REG32/IMM8;

作用：将第一个操作数的第n位拷贝到进位标志CF中

BTS（位测试并置位）

写法：BTS REG16/MEM16,REG16/IMM8;或BTSREG32/MEM32,REG32/IMM8;

作用：将第一个操作数的位n拷贝到进位标志中，同时 将位n置位

BTR(位测试并复位)

写法：BTR REG16/MEM16,REG16/IMM8;或BTRREG32/MEM32,REG32/IMM8;

作用：将第一个操作数的位n拷贝到进位标志中，同时 将位n清零

BTC(位测试并复位)

写法：BTC REG16/MEM16,REG16/IMM8;或BTCREG32/MEM32,REG32/IMM8;

作用：将第一个操作数的位n拷贝到进位标志中，同时 将位n取反

例子：假设(AX)=1234H

BT 2，AX ; 指令执行后，CF=1，(AX)=1234H

## 4.7条件传送CMOVxx（2分）

## cmovcc src,dest

cc:表示条件

src: r16, r32, r64

dst: r/m16, r/m32, r/m64

**无符号数的条件传送**：

用a、b、e、n、c分别表示：大于、小于、等、否、进位

CMOVA/CMOVNBE 大于/小于或不等于 (CF或者ZF)=0

CMOVAE/CMOVNB 大于或者等于/不小于 CF = 0

CMOVNC 无进位 CF = 0

CMOVB/CMOVNAE 小于/不大于 CF = 1

CMOVC 进位 CF = 1

CMOVBE/CMOVNA 小于或者等于/不大于(CF或ZF) = 1

CMOVE/CMOVZ 等于/零 ZF = 1

CMOVNE/CMOVNZ 不等于/不为零 ZF = 0

CMOVP/CMOVPE 奇偶校验 PF = 1

**有符号数的条件传送：**

用g、l、e、n、o分别表示：大于、小于、等、否、溢出

CMOVG/CMOVNLE 大于/不小于等于 (ZF=0 and SF=OF) CMOVGE/CMOVNL 大于等于/不小于 (SF异域OF) = 0 CMOVL/CMOVNGE 小于/不大于等于 (SF民域OF) = 1 CMOVLE/CMOVNG 小于等于/不大于 ((SF异域OF)或ZF) =1

CMOVO 溢出 OF=1

CMOVNO 末溢出 OF=0

CMOVS 带符号（负） SF=1

CMOVNS 无符号（非负） SF=0

例子：cmovge %r8，%r9

cmovgel %r9，%r10

cmovgl %r8d，%r10d

cmovll %r8d，%r10d

例子：假设%ecx的值为x，%edx的值为y，%ebx的值为y-x，%eax的值为x-y

cmpl %edx，%ecx //比较x和y

cmovl %ebx，%eax //如果x小于y，eax=ebx=y-x

## 4.8条件设置SETCxx（1分）

指令 同义词 作用 设置条件

sete Setz ZF 相等 / 结果为0

setne setnz ~ZF 不相等 / 结果不为0

sets SF 结果为负数

setns ~SF 结果为非负数

setl setnge SF^OF 小于 (符号数)

setle setng (SF^OF)|ZF 小于等于 (符号数)

setg setnle ~(SF^OF)&~ZF 大于 (符号数)

setge setnl ~(SF^OF) 大于等于 (符号数)

seta setnbe ~CF&~ZF 大于 (无符号数)

setae setnb ~CF 大于等于 (无符号数)

setb setnae CF 小于 (无符号数)

setbe setna CF|ZF 小于等于 (无符号数)

具体操作如下：

SETcc DEST

Operation

IF condition THEN

DEST ← 1;

ELSE

DEST ← 0;

FI;

## 4.9进位位操作（1分）

adc src,dest 将src与dest相加，并且加上CF，结果返回dest

例子：

mov 2，ax

mov 1，bx

sub ax，bx

adc 1，ax

执行后，(ax)=4.adc执行时，相当于计算：(ax)+1+CF=2+1+1=4

# 

# 第5章 BITS函数实验与分析

每题8分，总分不超过80分

截图： $ ./btest –f 函数名

## 5.1 函数lsbZero的实现及说明

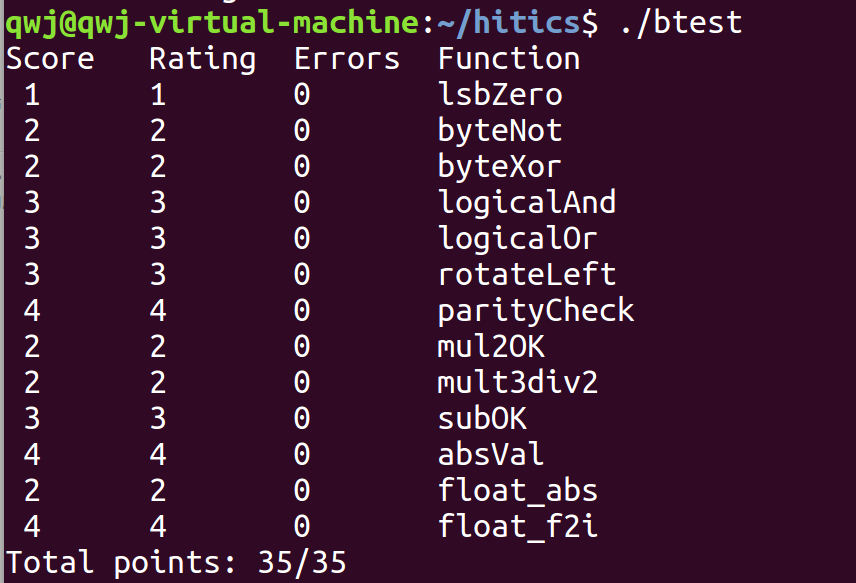
程序如下：int lsbZero(int x) {

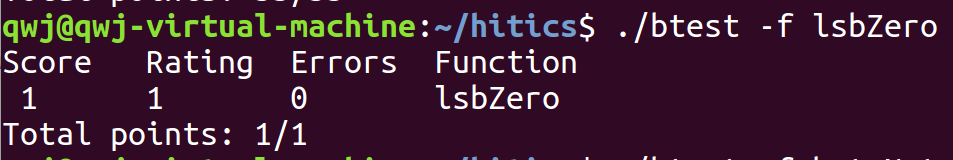
x=x>>1;

x=x<<1;

return x;

}

btest截图：



设计思想：先将x右移一位，再将x左移一位，实现将最后一位变为0。

## 5.2 函数byteNot的实现及说明函数

程序如下：int byteNot(int x, int n) {

int y = 0xff;

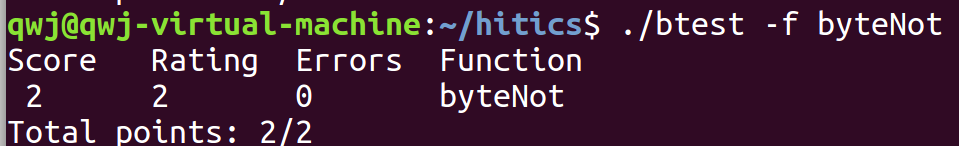
n=n<<3;

y=y<<n;

x=x^y;

return x;

}

btest截图：

设计思想：A^0=A , A^1 = 非A。将x的第n个字节的每位都与1异或，实现取反。

## 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) {

x=x>>n;

y=y>>n;

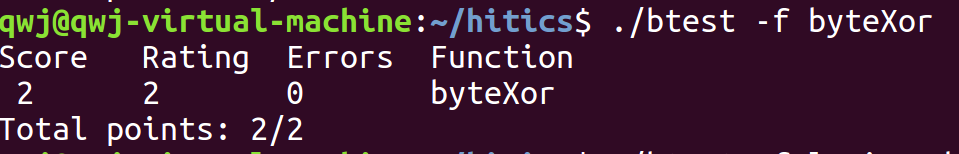
n=n<<3;

x=x&0xff;

y=y&0xff;

return !!(x^y);

}

btest截图：

设计思想：0^0=0 , 1^1=0 , 0^1=1 . 取出x和y的第n个字节进行异或，最后转化成逻辑上的0和1并返回。

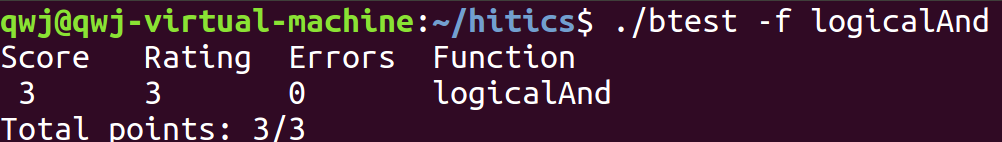
## 5.4 函数logicalAnd的实现及说明函数

程序如下：int logicalAnd(int x, int y) {

x=!((!x)|(!y));

return x;

}

btest截图：

设计思想：把x和y分别取NOT，二者相或后再取NOT，即可得到逻辑与。

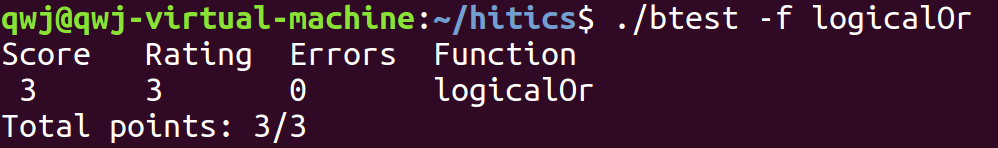
## 5.5 函数logicalOr的实现及说明函数

程序如下：int logicalOr(int x, int y) {

x=(!(!x))|(!(!y));

return x;

}

btest截图：

设计思想：把x和y分别转化成逻辑的0和1，在相或，实现逻辑或。

## 5.6 函数rotateLeft的实现及说明函数

程序如下：int rotateLeft(int x, int n) {

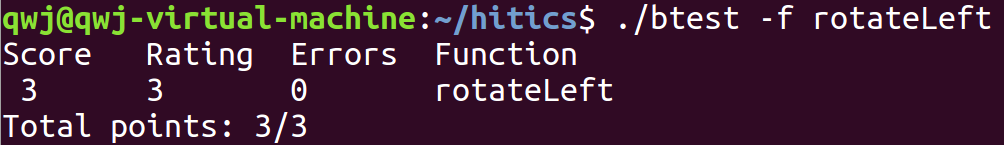
int y;

y=~((~0)<<n);

x=(x<<n)+((x>>(32+(~n+1)))&y);

return x;

}

btest截图：

设计思想：先构造y为高（32-n）位为0的y，再与x右移（32-n）的x相与，相当于储存了x的高n位数，最后再与x左移n位相加即可。

## 5.7 函数parityCheck的实现及说明函数

程序如下：int parityCheck(int x) {

x=x^(x<<16);

x=x^(x<<8);

x=x^(x<<4);

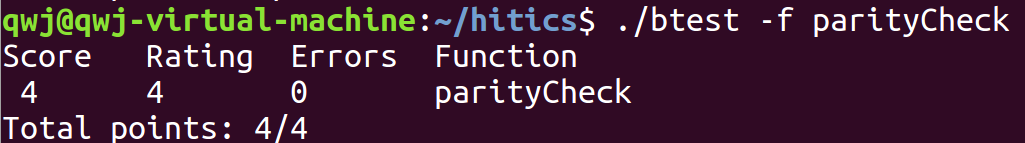
x=x^(x<<2);

x=x^(x<<1);

x=x>>31;

return !(!x);

}

btest截图：

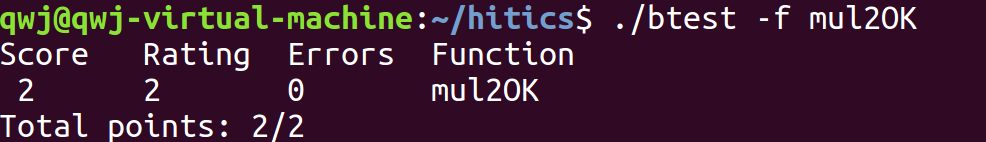
设计思想：每次移位将x的低半位数与高半位数进行异或，实现得到的x位表示总是减去偶数个1，并不影响x位表示中1个数的奇偶性。最后把得到的x右移31位，并变成逻辑1和0.

## 5.8 函数mul2OK的实现及说明函数

程序如下：int mul2OK(int x) {

return (((x>>31)&0x1)^((x>>30)&0x1))^0x1;

}

btest截图：

设计思想：若x的符号位为1，则第30位为0时会溢出；若x符号位为0，则第30位为1时会溢出；于是将x的31位和30位分别与1相与，再异或，再与1相与即可。

## 5.9 函数mult3div2的实现及说明函数

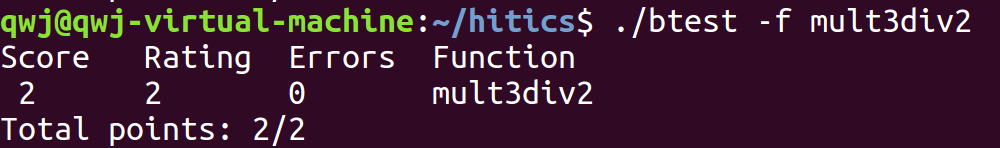
程序如下：int mult3div2(int x) {

x=x+(x<<1);

x=(x>>1)+(((x>>31)&0x1)&(((x<<31)>>31)&0x1));

return x;

}

btest截图：

设计思想：先实现x\*3/2，要注意的是x为负数且最低有效位为0时，相乘的结果要向0舍入，故此时要加上1.

## 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;

z=x+(~y+1);

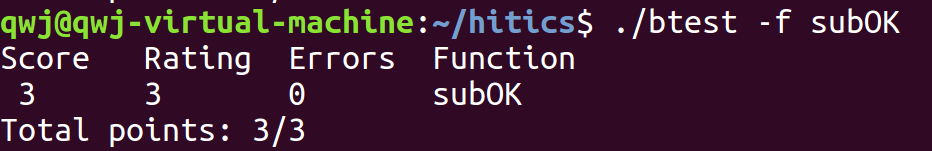
z=(z>>31)&0x1;

x=(x>>31)&0x1;

y=(y>>31)&0x1;

return ((x^y)&(x^z))^0x1;

}

btest截图：

设计思想：x符号位与y符号位不同，且x符号位与（x-y）符号位也不同时，即溢出。

## 5.11 函数absVal的实现及说明函数

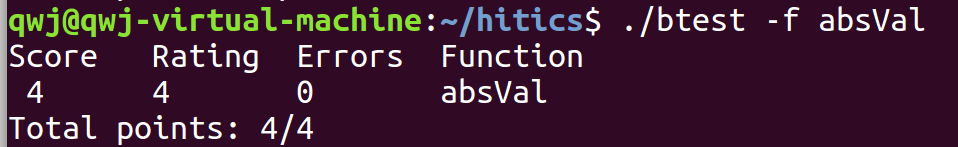
程序如下：int absVal(int x) {

int y=x>>31;

x=(y&(~x+1))+((~y)&x);

return x;

}

btest截图：

设计思想：当x符号位为0时，x绝对值就是x；当x符号位为1时，x绝对值是（~x+1）。

## 5.12 函数float\_abs的实现及说明函数

程序如下：unsigned float\_abs(unsigned uf) {

int x=uf&0x7fffffff;

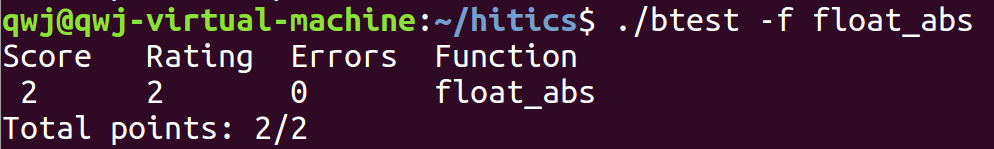
if(x>0x7f800000)

return uf;

else

return x;

}

btest截图：

设计思想：NaN表示的数阶码全为1，且小数域为非0。将uf的符号位变为0，得到的数若比0x7f80000大，即为NaN。

## 5.13 函数float\_f2i的实现及说明函数

程序如下：int float\_f2i(unsigned uf) {

int x,y;

unsigned mini=0x80000000;

x=(uf>>23)&0xff;

y=(uf&0x007fffff)^0x00800000;

if(x>158)

{

return mini;

}

if(x<127)

{

return 0;

}

else if(((uf>>31)&0x1)==1)

{

if(x>150)

return ((~(y<<(x-150)))+1);

else

return ((~(y>>(150-x)))+1);

}

else

{

if(x>150)

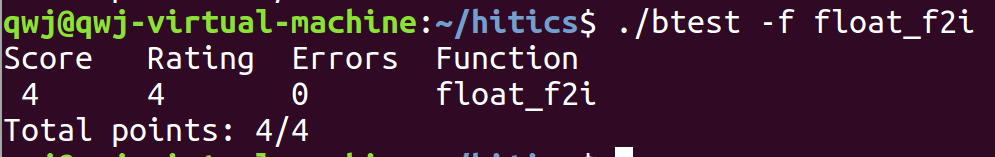
return (y<<(x-150));

else

return (y>>(150-x));

}

}

btest截图：

设计思想：将uf左移23位并与0xff相与来取出阶码域；设置y来取出小数域，并令其第23位为1，为了规格化的值隐含的以1开头的表示。

当阶码的值E大于31时，表示float转化为int会溢出，或者出现阶码域全为1的情况；当阶码的值小于0，返回0；当uf最高位为1时，判断x与150的大小，决定移位的方向，并以补码形式将数值存储；当uf最高位为0时，判断x与150大小，决定移位的方向。

## 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

系统学习了汇编语言指令；

熟练掌握位运算和逻辑运算来实现函数；

通过C程序深入理解计算机底层实现与优化。

## 10.2 请给出对本次实验内容的建议

关于第三章写出C语言例句和第四章写出汇编语言例句，题目中未给出范例，因此关于例句的形式易产生困惑。

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.