

# 哈尔滨工业大学

# 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

<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)	- 6 -
<b>第 3 章 C 语言的位操作指令</b>	<b>- 7 -</b>
3.1 逻辑操作 (1 分)	- 7 -
3.2 无符号数位操作 (2 分)	- 7 -
3.3 有符号数位操作 (2 分)	- 7 -
<b>第 4 章 汇编语言的位操作指令</b>	<b>- 8 -</b>
4.1 逻辑运算(1 分)	- 8 -
4.2 无符号数左右移 (2 分)	- 8 -
4.3 有符号左右移 (2 分)	- 8 -
4.4 循环移位 (2 分)	- 8 -
4.5 带进位位的循环移位 (2 分)	- 9 -
4.6 测试、位测试 BTX (2 分)	- 9 -
4.7 条件传送 CMOVXX (2 分)	- 9 -
4.8 条件设置 SETCXX (1 分)	- 9 -
4.9 进位位操作 (1 分)	- 10 -
<b>第 5 章 BITS 函数实验与分析</b>	<b>- 10 -</b>
5.1 函数 LSBZERO 的实现及说明	- 10 -
5.2 函数 BYTENOT 的实现及说明函数	- 10 -
5.3 函数 BYTEXOR 的实现及说明函数	- 10 -
5.4 函数 LOGICALAND 的实现及说明函数	- 10 -
5.5 函数 LOGICALOR 的实现及说明函数	- 11 -
5.6 函数 ROTATELEFT 的实现及说明函数	- 11 -
5.7 函数 PARITYCHECK 的实现及说明函数	- 11 -
5.8 函数 MUL2OK 的实现及说明函数	- 11 -
5.9 函数 MULT3DIV2 的实现及说明函数	- 12 -
5.10 函数 SUBOK 的实现及说明函数	- 12 -

5.11 函数 ABSVAL 的实现及说明函数.....	- 12 -
5.12 函数 FLOAT_ABS 的实现及说明函数.....	- 13 -
5.13 函数 FLOAT_F2I 的实现及说明函数 .....	- 13 -
5.14 函数 XXXX 的实现及说明函数（CMU 多出来的函数-不加分） .....	- 14 -
<b>第 6 章 总结.....</b>	<b>- 15 -</b>
10.1 请总结本次实验的收获.....	- 15 -
10.2 请给出对本次实验内容的建议.....	- 15 -
<b>参考文献.....</b>	<b>- 16 -</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

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

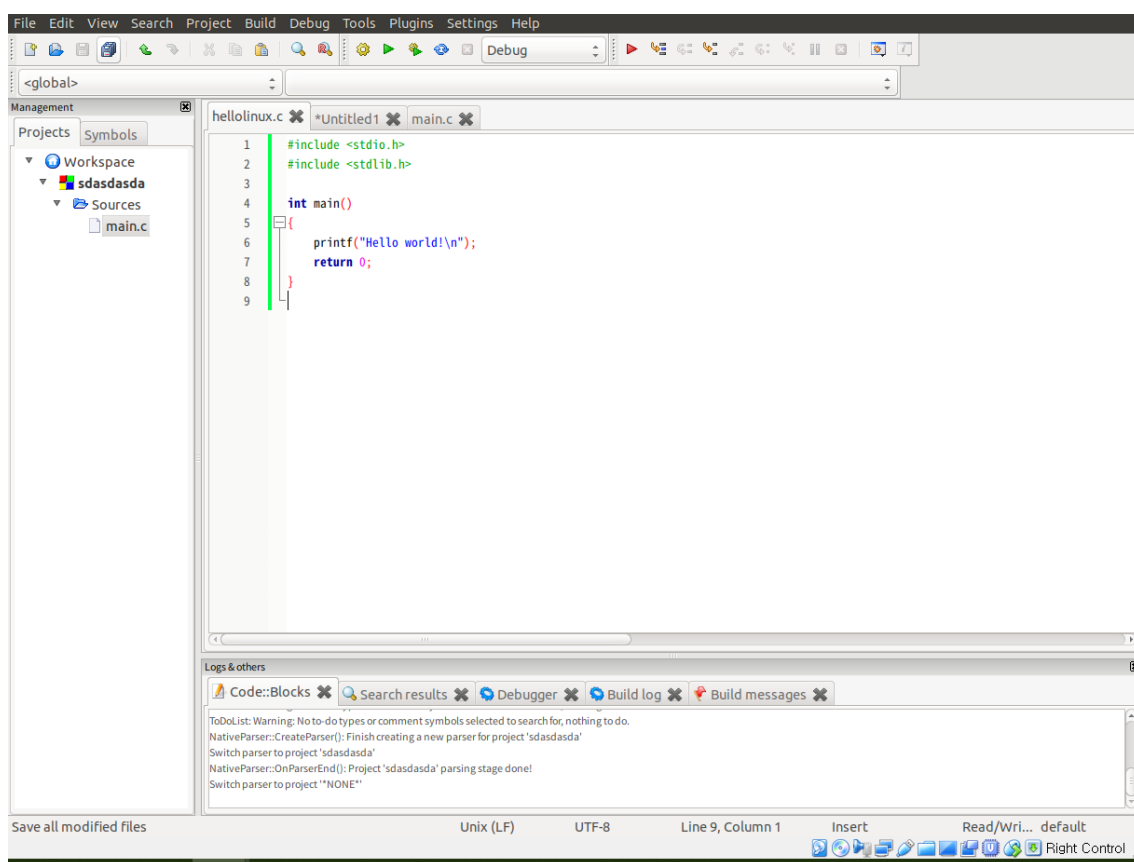
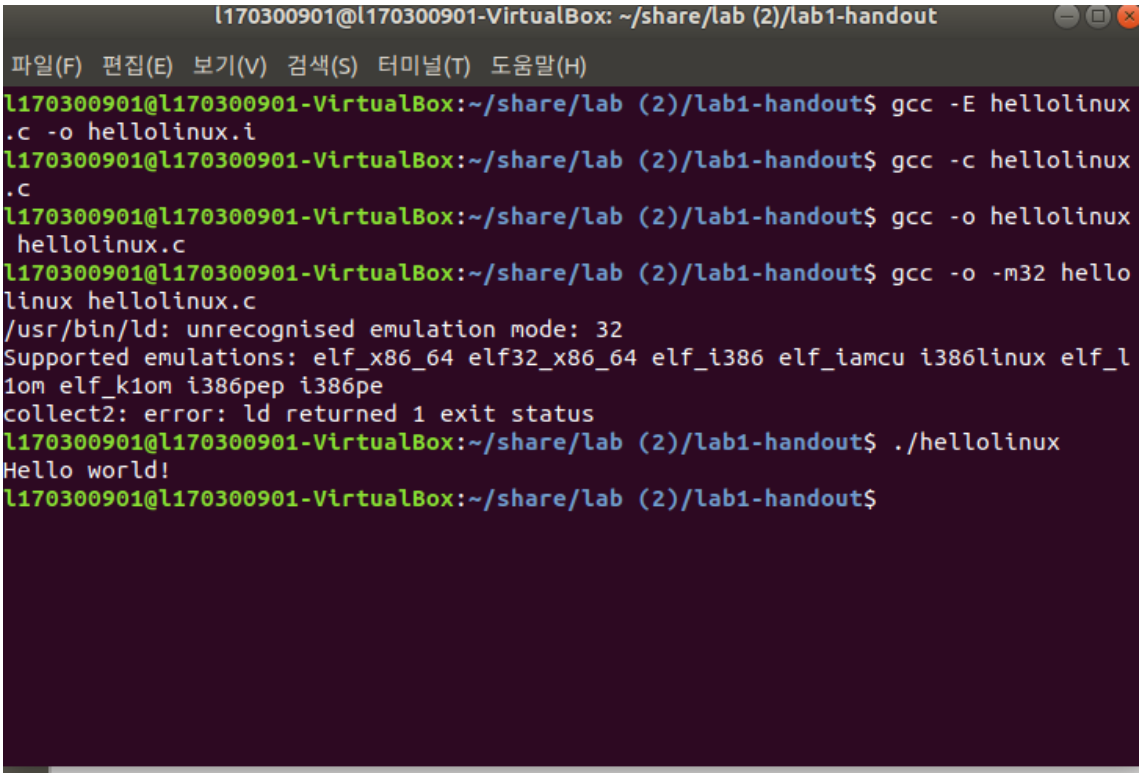


图 2-1 Ubuntu 下 CodeBlocks 截图

## 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。



```
l170300901@l170300901-VirtualBox: ~/share/lab (2)/lab1-handout
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ gcc -E hellolinux.c -o hellolinux.i
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ gcc -c hellolinux.c
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ gcc -o hellolinux hellolinux.c
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ gcc -o -m32 hello
linux hellolinux.c
/usr/bin/ld: unrecognized emulation mode: 32
Supported emulations: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu i386linux elf_l
1om elf_k1om i386pep i386pe
collect2: error: ld returned 1 exit status
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ ./hellolinux
Hello world!
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$
```

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:  $1010 \mid 1100 = 1110$

~ 非运算, 如:  $a = 1010; \sim a = 0101$

<<左移, 如:  $1110 \ll 1 = 1100$

>>右移, 如:  $1000 \gg 1 = 0100$

### 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:

$X = X_0X_1X_2 \dots X_n$  其中  $X_i = \{0, 1\}, 0 \leq i \leq n$  即  $X_0 \cdot 2^n + X_1 \cdot 2^{(n-1)} + X_2 \cdot 2^{(n-2)} + \dots + X_{n-1} \cdot 2 + X_n$

数值范围是  $0 \leq X \leq 2^{(n+1)} - 1$

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)

" $\vee$ " 表示"或"

" $\wedge$ " 表示"与".

" $\neg$ " 表示"非".

"=" 表示"等价".

1 和 0 表示"真"和"假"

(还有一种表示,"+"表示"或", " $\cdot$ "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 =  $\sim(00001111)+1 = 11110000+1 = 11110001$

>> 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 =  $\sim(11111100-1) = \sim(11111011) = 00000100$ (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 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：

```
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ ./btest -f lsbZero
Score  Rating  Errors  Function
1      1      0      lsbZero
Total points: 1/1
```

设计思想：

## 5.2 函数 byteNot 的实现及说明函数

程序如下：

```
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ ./btest -f byteNot
Score  Rating  Errors  Function
2      2      0      byteNot
Total points: 2/2
```

设计思想：

## 5.3 函数 byteXor 的实现及说明函数

程序如下：

```
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ ./btest -f byteXor
Score  Rating  Errors  Function
2      2      0      byteXor
Total points: 2/2
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$
```

设计思想：

## 5.4 函数 logicalAnd 的实现及说明函数

程序如下：

```
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ ./btest -f logicalAnd
Score   Rating  Errors  Function
3       3       0       logicalAnd
Total points: 3/3
```

设计思想：

## 5.5 函数 logicalOr 的实现及说明函数

程序如下：

```
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ ./btest -f logicalOr
Score   Rating  Errors  Function
3       3       0       logicalOr
Total points: 3/3
```

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：

```
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ ./btest -f parityCheck
Score   Rating  Errors  Function
4       4       0       parityCheck
Total points: 4/4
```

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 mul20K 的实现及说明函数

程序如下：

```
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ ./btest -f mul20K
Score  Rating  Errors  Function
  2      2      0      mul20K
Total points: 2/2
```

设计思想:

X 和 2X 的符号位异或的方式将得到结果，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:

```
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ ./btest -f mult3div2
Score  Rating  Errors  Function
  2      2      0      mult3div2
Total points: 2/2
```

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:

```
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ ./btest -f subOK
Score  Rating  Errors  Function
  3      3      0      subOK
Total points: 3/3
```

设计思想:

先提出两个数的符号位。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:

```
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ ./btest -f absVal
Score  Rating  Errors  Function
4      4      0      absVal
Total points: 4/4
```

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  $\sim x+1$ 。

## 5.12 函数 float\_abs 的实现及说明函数

程序如下:

```
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ ./btest -f float_abs
Score  Rating  Errors  Function
ERROR: Test float_abs(-4194304[0xffc00000]) failed...
...Gives 2143289344[0x7fc00000]. Should be -4194304[0xffc00000]
Total points: 0/2
```

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:

```
l170300901@l170300901-VirtualBox:~/share/lab (2)/lab1-handout$ ./btest -f float_f2i
Score  Rating  Errors  Function
ERROR: Test float_f2i(8388608[0x800000]) failed...
...Gives 8388608[0x800000]. Should be 0[0x0]
Total points: 0/4
```

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>.