《计算机组成原理实验》

实验报告

（实验二）

学 院 名 称 ：	数据科学与计算机学院
专业（班级）：	18 计教学 3 班
学 生 姓 名 ：	张天祎
学　　号	18340215
时　　间 ：	2019　年　10　月　21　日

成 绩 ：

实验二 ： MIPS汇编语言程序设计实验

一. 实验目的

1. 初步认识和掌握 MIPS 汇编语言程序设计的基本方法；

2. 熟悉 QEMU 模拟器和 GNU/Linux 下 x86-MIPS 交叉编译环境的使用。

二. 实验内容

学号末两位 15。15%4+1=4。

4. 【密码】编写一个程序， 先从键盘输入一个字符串（有英文字母，可能也有数字），然后显示其中数字符的个数、英文字母的个数和字符串的长度；字符串中不能有空格，若有将其删除，并将改变后的字符串按相反的顺序显示出来；输入第二个字符串，然后将输入的字符串与前面处理后的字符串比较是否相同，若相同，输出“Password Right!”，否则输出“Password Error!”。

例如对于第一次的输入：

```
abcde fgh12 3ijkl
```

其输出为：

```
Letters: 12
Numbers: 3
String Length: 17
Reversed String: lkji32lhgfedcba
Retype:
```

接下来在“Retype:”后输入正确的密码

```
lkji32lhgfedcba
```

将得到输出：

```
Password Right!
```

否则将得到输出

```
Password Error!
```

三、 实验设备、

PC 机一台，Virtual Box 虚拟机器软件一套，已经安装好 GNU/Linux、QEMU 及x86-MIPS 交叉编译环境的虚拟计算机一套。

三. 实验过程与结果

首先无脑写了一个C++程序，用于在后续写汇编代码的过程中进行参照。（后续证明这个过程可能没多大用，需要频繁使用寄存器的汇编代码在具体的细节实现上还是和C++程序很不一样，但C++程序在整个编程的大方向上有一丢丢的指导作用）

首先看服务器上的各种文件，熟悉MIPS指令，交叉编译。当然有很多看不懂，也有部分太细不想看。本来以为可以使用SPIM仿真系统调用I/O操作，后来多次阅读实验环境指导后得知这些指令均不能使用。结合Linux内核调用和给出的io_example.S，逐渐明白了这次MIPS汇编中的I/O形式，只要配合好syscall需要的几个参数形式就好。

但话虽如此，整个程序的过程和逻辑操作还算简单，真正复杂的还是I/O操作，整个I/O全部使用字符串的形式，这给int的输出带来很大麻烦。在调用write时第三个参数a2是字符串长度，所以在写write函数前需要先写一个获取字符串长度的函数，到结尾’\0’为止，包括结尾的’\n’。写完readstr、writestr、strsize这几个函数后，开始将一些const string加入程序中，并进行一些测试。

然后开始写writeint函数。考虑到正整数每次mod10取到的是最后一位，所以为存入方便，先遍历一遍获取该正整数在十进制下的位数，然后将指针偏移到恰当的位置，先存入一个’\n’，然后从末尾往前一位一位存，存完后调用write函数输出。这个过程同样要经历一些测试。

回到main函数，有这几个函数后，整个题大大简化。输入语句和输出提示语句及int这里略过，这只是调用上述函数的过程。现在需要解决两个主要问题：一是处理原字符串，统计字母和数字个数；二是比较处理后的字符串和新输入的字符串。

解决第一个问题，因为字符串需要反向，所以将字符串从后往前处理。先获取字符串长度，将指针偏移到合适位置，指向非’\n’的最末位。若字符为’ ’，不存入，继续循环。由题有输入仅为字母、数字、空格。故以’9’为界，若字符小于等于’9’，num_count++，

否则letter_count++，再存入保存结果的字符串，如此循环直到循环变量和strsize-1相等（这里没有处理换行符）。最后在保存结果的字符串末尾加上换行符。

第二个问题其实很简单，只要先获取结果的字符串和新输入字符串长度，若不相等，直接错误，若不相等，则从头开始一位一位处理，若存在不同的字符，则输出错误，若直至结束都无不同，则输出正确。这中间当然还有很多的调试过程。

至此，程序结束。成果如下：

```
zty@ubuntu:~/Desktop$ qemu-mips -g 12324 ./test4
123 zxc ASD
Letters:6
Numbers:3
String Length: 11
Reversed String: DSAcxz321
Retype:
DSAcxz320
Password Error!
zty@ubuntu:~/Desktop$ qemu-mips -g 12324 ./test4
123 zxc ASD
Letters:6
Numbers:3
String Length: 11
Reversed String: DSAcxz321
Retype:
DSAcxz321
Password Right!
zty@ubuntu:~/Desktop$
```

四. 实验心得

存在三个从头到尾都没有解决的问题。

第一个，linux内核调用中write的第一个参数a0究竟应该是什么？在实验环境指导中，read的参数应该被设置为0，write的参数应该被设置为1；但在我测试的过程中（因为失误）发现write参数如果被设置为0，同样运行良好，我转而修改read的第一个参数为1，发现无法正常读取。于是又修改回实验环境指导中的推荐值，一切正常。

第二个，.set noat和.set noreorder存在的意义是什么？开始仿照io_example.S中两个都加了，发现总有莫名其妙的地方报错，然后把.set noat删去后一切又正常了。经查资料，.set noreorder好像是和cpu流水线操作相关的操作。一个重要的点是加上这个语句后所有跳转和调用语句都后面都要加一句nop或者干脆空一行（调试时发现空行就会默认为nop）。

第三个，函数的参数调用问题。本题中要进行很多次相似的操作，一个很自然的想法是使用函数。但其中传参的过程完全不了解，尤其在参考资料MIPS汇编简要介绍中，有提到对t寄存器和s寄存器在子程序调用中的改变情况，但说法模棱两可（可能是我太菜了看不

懂）。所以我在完成实验的时候非常小心的分配寄存器，在函数中尽量使用t寄存器，并用注释标记使用过的寄存器，以免在函数中修改了main函数中的需要的保存的重要值。另一方面，main函数中一些不重要的或者是常更新的值都用t寄存器，而重要的值在得到后都存入s寄存器，并且对它们只读不写。

虽然还有很多问题，但用最谨慎的方法完成了程序。总体上还算顺利，全程模块化函数测试，没碰到卡太久的问题，在编程的开始比较困难，后面反而觉得简单。花了不少时间完成了这个题，深刻认识到了程序设计高级语言的美好。希望随着后续学习的深入，能解决这些问题。