**《计算机原理与体系结构》实验报告**

**学号：16300240023 姓名：杨芗琳 学院：计算机科学技术学院 完成时间：2018.4.23**

1. 实验名称

熟悉PCSpim

1. 实验目的
2. 熟悉PCSpim模拟器
3. 熟悉汇编程序、连接器
4. 理解MIPS计算机的架构，特别是MIPS指令集、寄存器、f-register、系统调用、调用和返回、堆栈
5. 实验过程
   1. Debug p1.asm, p2.asm, p3.asm, observe and record what you have seen.

**p1.asm:**

1. 运行了第一条指令



将40+0的值赋给R10寄存器



1. 之后运行第二条指令



将17+0的值赋给R11寄存器



1. 运行第三条指令



R11寄存器存储R10+R11的内容



1. 运行第四条指令



将40+0的值赋给$10寄存器



1. 运行第五行



将寄存器$0中的值加上0赋给$t4寄存器



结果是R12寄存器的值还是为0，此时证明刚刚的赋值没有成功，$0寄存器的值始终为0

1. 运行第六行



$v0=10,表明程序准备结束



1. 运行最后一行



系统调用，程序结束；

程序目的在于计算40+17的值

**p2.asm（同1）**

程序第一行



$t2=0+40



程序第二行



$t2寄存器高位得到0x1234的值



程序第三行



$t2寄存器低位得到40



程序第四行



寄存器$t3得到0x12340028的值



程序第四行



\*Sometimes nothing is on the line after the semicolon. This means that the instruction was produced by SPIM as part of translating a pseudoinstruction into more than one actual MIPS instruction.

此式为spim在翻译一条的伪指令成真正的MIPS指令过程的产物

程序第五行



$v0=10,程序准备结束



程序第六行



系统调用，syscall，程序结束

程序目的在于测试给寄存器赋值

**p3.asm**

1. 程序第一行



寄存器$t0得到h的地址



1. 程序第二行



$t1得到数组A的地址



1. 程序第三行



$t2得到h的值



1. 程序第四行



一个整数占4个字节，$t1+上偏移量32得到A[8]的地址，$t3=A[8]



1. 程序第五行



$t3=h+A[8]



$t3=40+19=59=3\*16+11

1. 程序第六行



将$t3中的值store到A[12]中

1. 程序第七行



$v0=10,准备退出



1. 程序第八行



退出

此程序目的在于计算，存值。

* 1. Rewrite p1.asm, using MIPS & the Simulator, to accept two integers and the output the result, the following is the typical Input & output: Please enter 1st number: 20

Please enter 2nd number: 50

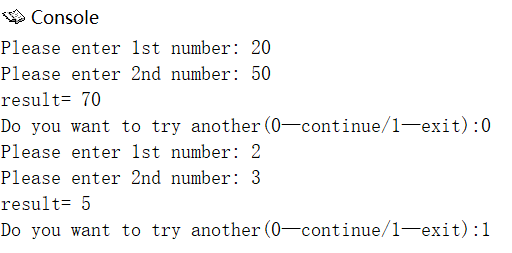
The result of 20 & 50 is: 70

Do you want to try another(0—continue/1—exit):

思路：（具体代码p1.asm）

1. 输入第一个数，存在$t0中
2. 输入第二个数，存在$t1中
3. 将$t0+$t1的值存在result中
4. 输入结果
5. 条件判断，如果输入的值等于零，跳回第一步（循环）
6. 如果输入的值不等于0，syscall，程序结束

输出截图：

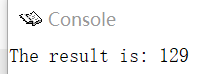


* 1. Translate the following c Code into MIPS, then debug your code and optimize your code.

思路：(具体代码3result.asm)

1. 从main函数开始，逐条翻译
2. 进入sumn函数时，采用$t0-$t2寄存器存贮中间变量
3. 循环
4. 结果返回
5. syscall，程序结束

输出截图：



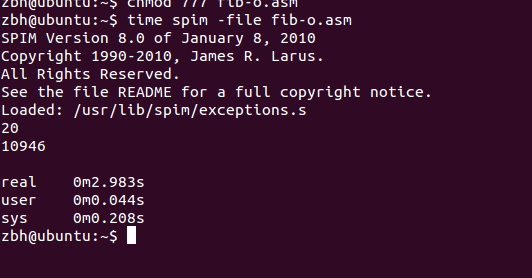
* 1. Read chapter 3 of Ref.[5], write a program called fib-o+.asm which is faster that fib-o.asm, and explain the reason.

思路：

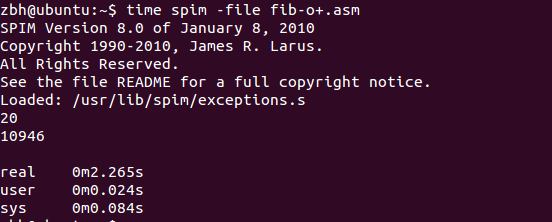
1. fib-s.asm到fib-t.asm的改进方法是改用寄存器，节省时间
2. fib-t.asm到fib-o.asm的改进方法是在n<2时不建立堆栈空间，直接返回1，从而节省了时间，并且不再保存n，因为n之后不会再调用
3. 延用这个思路，观察到当n<4时，f(n)=n，所以n<4时都不需要建立堆栈空间，可以直接返回n，节省时间

实验结果：

fib-o.asm:



fib-o+.asm:



从user time可以看到，当n=20时，时间从0.044s减少到0.024s，效率提高了近50%，并且随着n的增大，节省的时间越多。

1. 实验源代码
2. p1.asm(3.2)
3. 3result.asm(3.3)
4. fib-o.asm\fib-o+.asm(3.4)