|  |  |
| --- | --- |
| **计算机结构与组成** | **实验1** |

**实验目的**

本实验的目的是让你熟悉MARS仿真器的使用，学习用它来运行和调试程序。如果你需要在其他地方使用MARS，可以直接下载mars.jar。MARS程序由[密苏里州立大学](http://courses.missouristate.edu/KenVollmar/MARS/)开发。UCB的教师消除了一些BUG，但还没被主要开发者接受. MARS程序是用JAVA写的，因此使用它之前，需要在你的机器上安装Java J2SE 1.5.0 SDK或者更高版本,该程序可以从[Sun](http://java.sun.com/)公司下载。

**运行MIPS汇编程序**

汇编程序一般放在.s为扩展名的文本文件中. 程序必须包含标识 "main:" (和C程序的main函数相似)，而以语句指令"addi $v0,$0,10"，后跟一个"syscall"（系统调用）结束. 和普通的使用"jr $ra"返回不同，主函数的特别之处在于在其结束后，必须将控制权交给操作系统，而不是简单的返回。

在本次实验中，我们将在MARS中来运行我们的代码, MARS是一个MIPS仿真器，它提供了丰富的图形化接口的调试，而不仅仅是一个运行的裸处理器. 一般而言，汇编程序员更喜欢这种开发模式，因为这样调试更方便。

你可以通过直接点击MARS来运行你的程序。这是一个可执行的JAR文件。不过如果你的电脑上没有JRE，可能打开后是解压的文件包。

打开mars, 可以使用File->Open装入.s文件。可以点击"Edit"页来编辑代码、点击"Execute"页来运行或者调试程序.为了能运行程序，需要首先汇编你的代码，方法是使用Run->Assemble (F3).

为了在MARS中调试汇编代码，可以设置断点，一步步运行（单步运行）,同时注意观察寄存器或内存的变化,同时也请留意在程序开始前各寄存器的初始值。如果可能，实验前应花些时间来熟悉MARS.

**练习**

**Setup**

首先，你可以把我们提供的程序放到一个合适的目录下。

**练习1: 熟悉MARS**

将[lab4\_ex1.s](file:///D:\\快盘\\我的资料\\计算机体系结构\\2017\\实验\\实验\\lab4_ex1.txt)载入MARS，并汇编代码.假定fib[0] = 0; fib[1] = 1; fib[n] = fib[n-1] + fib[n-2]  
使用Help(问号图标)回答下列关于MARS的问题.

1. .data, .word, .text 指示器（directives）的含义是什么(即, 在每段中放入什么内容)?
2. 在MARS中如何设置断点breakpoint? 请在第15行设置断点，并在所有问题解答完后，将此结果给老师检查。
3. 在程序运行到断点处停止时，如何继续执行? 如何单步调试代码?
4. 如何知道某个寄存器register的值是多少? 如何修改寄存器的值.
5. n 存储在内存中的哪个地址? 通过修改此内存处的值来计算第13个fib数.
6. 16 和 18 行使用了syscall指令. 其功能是什么，如何使用它? (提示: syscall 在Help中有说明!如何英文不是太好，可以一边运行，一边看效果，来体会其用途)

把答案给老师看.

**练习 2: 一个简短的MIPS程序**

编写MIPS代码完成：在给定$s0 和 $s1的值的前提下，将下列值放到 $t? 寄存器中（其中？表示任意0-7之间的数）:

$t0 = $s0

$t1 = $s1

$t2 = $t0 + $t1

$t3 = $t1 + $t2

...

$t7 = $t5 + $t6

换言之, 对$t2 到 $t7的每个寄存器，都存储其前两个$t? 寄存器的值. 寄存器$s0 和 $s1 中包含初始值.

不要在代码中设置$s0 和 $s1 的值. 取而代之, 学会如何在MARS中手动设置它们的值.

将你的代码存储到文件lab4\_ex2.s 中，然后给老师检查.

**练习 3: 调试（Debugging）MIPS程序**

调试程序[lab4\_ex3.s](file:///D:\快盘\我的资料\计算机体系结构\2017\实验\实验\lab4_ex3.txt)中的循环. 该程序将从$a0所指示的内存地址中复制一个整数到$a1所指示的内存地址, 起到读入一个zero值时结束. 复制的整数的个数(**不**含zero值)应存储在中$v0.

请在文件lab4\_ex3.txt 中描述代码的错误bug(s). 新建一个文件lab4\_ex3\_ok.s ，其中放的是没有bug的代码[lab4\_ex3.s](file:///D:\\快盘\\我的资料\\计算机体系结构\\2017\\实验\\实验\\lab4_ex3.s).把程序给老师看。

**Exercise 4: Compiling from C to MIPS**

The file [lab4\_ex4.c](file:///D:\快盘\我的资料\计算机体系结构\2017\实验\实验\lab4_ex4.c) is a C version of the program in the previous part of the lab. Compile this program into MIPS code using the command:

step1: "开始"->"运行"->"cmd"

step2: telnet 192.168.1.86

step3: linux-mr22 login:s11

step4: password:11

step5: ./mips-gcc -S -O2 -fno-delayed-branch lab4\_ex4.c -o lab4\_ex4.s (可自己命名)

step6: more lab4\_ex4.s (可自己命名)

The -O2 option (letter "O" and 2) turns on a level of optimization. The -S option generates assembly code. Don't worry about the delayed branch option for now; we will revisit this topic again when we talk about pipelining. The above command should generate assembly language output for the C code. Please note that you will not be able to run this code through MARS.

Find the assembly code for the loop that copies sources values to destination values. Then, find where the source and dest pointers you see in [lab4\_ex4.c](file:///D:\快盘\我的资料\计算机体系结构\2017\实验\实验\lab4_ex4.c) are originally stored in the assembly file. Finally, explain how these pointers are manipulated through the loop.

Find the section of code in lab4\_ex4.s that corresponds to the copying loop and explain how each line is used in manipulating the pointer.