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 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 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.

a. .data, .word, .text 指示器（directives）的含义是什么(即，在每段中放入什么内容)?

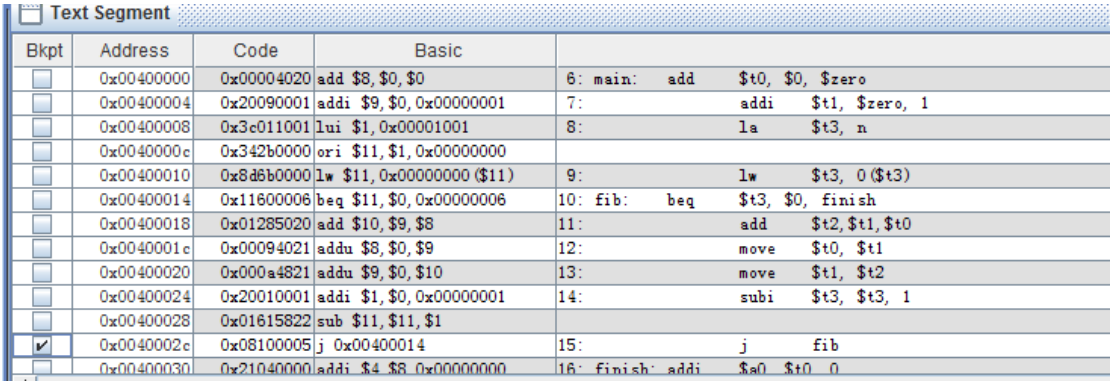
.data:将后续项存入数据段中

.word:将 32 位数据存入连续的主存字中

.text:将后续项存入用户代码段中

b. 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。

如图在第 15 行设置断点



c. 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?

继续执行  单步调试 

d. 如何知道某个寄存器 register 的值是多少? 如何修改寄存器的值.

观察 value 栏可知道 register 的值，再调试过程中双击对应 register 的 value 进行输入可修改寄存器的值

e. 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.

存储在 0x10010000，将 value 修改为 0x0000000c（第 1 个是 0x00000000，第 13 个是 0x000000c）

Address	Value (+0)
0x10010000	0x00000009

Data Segment

Address	Value (+0)
0x10010000	0x0000000c

f. 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)

第一个打印整数，第二个结束整个指令

Table of Available Services

Service	Code in $v0	Arguments	Result
print integer	1	$a0 = integer to print	
print float	2	$f12 = float to print	
print double	3	$f12 = double to print	
print string	4	$a0 = address of null-terminated string to print	
read integer	5		$v0 contains integer read
read float	6		$f0 contains float read
read double	7		$f0 contains double read
read string	8	$a0 = address of input buffer $a1 = maximum number of characters to read	*See note below table*
sbrk (allocate heap memory)	9	$a0 = number of bytes to allocate	$v0 contains address of allocated memory
exit (terminate execution)	10		
print character	11	$a0 = character to print	*See note below table*
read character	12		$v0 contains character read

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.

代码如下

```
1         .data
2
3         .text
4   main:   add $t0,$s0,$zero
5         add $t1,$s1,$zero
6         add $t2,$t0,$t1
7         add $t3,$t1,$t2
8         add $t4,$t2,$t3
9         add $t5,$t3,$t4
10        add $t6,$t4,$t5
11        add $t7,$t5,$t6
12  finish:
13        add $a0,$t7,$zero
14        li $v0,1
15        syscall
16        li $v0,10
17        syscall
18
19
```

手动设置$s0=0,$s1=1,则$t7=13

$s0	16	0x00000000
$s1	17	0x00000001

```
13
-- program is finished running --
```

练习 3：调试（Debugging）MIPS 程序

调试程序 lab4_ex3.s 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. 把程序给老师看。
Addiu 地址应该加 4，没有写 puti,puts,putc 函数,并且读到 0 没有跳出循环

```
1          .data
2  source:   .word   3, 1, 4, 1, 5, 9, 0
3  dest:     .word   0, 0, 0, 0, 0, 0, 0
4  countmsg: .asciiz " values copied. "
5          .text
6
7  main:   la      $a0,source
8          la      $a1,dest
9
10 loop:   lw      $v1, 0($a0)            # read next word from source
11         beq     $v1,$zero,loopend      # 读到0跳出循环
12         addiu   $v0, $v0, 1            # increment count words copied
13         sw      $v1, 0($a1)            # write to destination
14         addiu   $a0, $a0, 4            # advance pointer to next source
15         addiu   $a1, $a1, 4            # advance pointer to next dest
16         bne     $v1, $zero, loop       # loop if word copied not zero
17
```

```
18 loopend:
19         move    $a0,$v0                # $a0 <- count
20         jal     puti                   # print it
21
22         la      $a0,countmsg           # $a0 <- countmsg
23         jal     puts                   # print it
24
25         li      $a0,0x0A               # $a0 <- '\n'
26         jal     putc                   # print it
27
28 finish:
29         li      $v0, 10                # Exit the program
30         syscall
31 puti:   li      $v0,1
32         syscall
33         jr      $ra
34 puts:   li      $v0,4
35         syscall
36         jr      $ra
37 putc:   li      $v0,11
38         syscall
39         jr      $ra
```

```
6 values copied.

— program is finished running —
```

练习 4：编写程序

编写程序实现将一个数组 a[8]={7,8,9,10,8,1,1,1}的 8 个数平均数（只保留整数），并输出

代码如下

```
        .data
source:  .word  7, 8, 9, 10, 8, 1, 1, 1
        .text
main:   la      $a0,source              #加载数组a地址
        add     $s0,$zero,$zero         #$s0代表总和
        add     $s1,$zero,$zero         #$s1代表个数
loop:   lw      $v1, 0($a0)             #将$a0中的元素一次次加载到$v1中
        beq     $v1,$zero,finish        #如果$v1中读到0则跳出循环
        add     $s0,$s0,$v1             #总和$s0每次加上$v1
        addiu   $s1, $s1, 1             #个数$s1每次加上1
        addiu   $a0, $a0, 4             #数组$a0寻找下一个地址
        bne     $v1, $zero, loop        #如果没读到0继续循环
finish:
        div     $s0,$s1                 #用总和$s0除以个数$s1
        mflo    $a0                     #将结果平均数存入$a0中
        li      $v0, 1
        syscall
        li      $v0, 10
        syscall
```

```
5
-- program is finished running --
```

练习 5：编写程序

编写程序实现将一个数组 a[5]={7,8,9,10,8}数组中的最小值放入到 b 中

代码如下

```
          .data
source:   .word   7,8,9,10,8
maxint:   .word   999
          .text
main:   la      $a0,source              #加载数组a地址
        la      $s0,maxint              #$s0代表b
loop:   lw      $v1, 0($a0)             #将$a0中的元素一次次加载到$v1中
        beq     $v1,$zero,finish        #如果$v1中读到0则跳出循环
        slt     $t0,$v1,$s0             #如果$v1小于$s0(b)则$t0=1
        beq     $t0,$zero,else          #如果$t0=0则跳转到else
        add     $s0,$zero,$v1           #如果$t0=1则将$v1的值赋给$s0(b)
else:   addiu   $a0, $a0, 4             #数组$a0寻找下一个地址
        bne     $v1, $zero, loop        #如果没读到0继续循环
finish:
        add     $a0,$zero,$s0           #将最小数$s0(b)存入$a0中
        li      $v0, 1
        syscall
        li      $v0, 10
        syscall

```

```
7
-- program is finished running --
```