**中山大学数据科学与计算机学院本科生实验报告**

**（2017学年春季学期）**

课程名称：**计算机组成原理实验**  任课教师：郭雪梅 助教：李声涛、王绍菊

|  |  |  |  |
| --- | --- | --- | --- |
| 年级&班级 | **1518** | 专业(方向) | **软件工程(移动信息工程)** |
| 学号 | **15352408** | 姓名 | **张镓伟** |
| 电话 | **13531810182** | Email | [**709075442@qq.com**](mailto:709075442@qq.com) |
| 开始日期 | **3.24** | 完成日期 | **4.2** |

**一、实验题目**

**1.**下载MARS汇编语言编译器，运行调试2个简单的MPIS汇编程序，分别是求和和数组求和，并按要求对两个程序作出相应修改。

三个代码的具体要求如下：

1. Sum.asm: 单步运行程序，观察每一步的结果(0…4这5个数字求和)，结果会存入寄存器$8中。修改代码，使其求5-10这六个数字的和

2. SumArray.asm: 单步运行程序，观察结果(7，8，9，10，8这五个在a数组中的数字求和)，结果会存在地址从0x0开始的sum变量中。将变量a改为half word 类型，修改程序，比较结果，观察内存。

**2.编写简单的MIPS程序**

1. 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.

2. 假定你想编写一个MIPS程序foo，该程序使用5个字的数组，数组元素初始化

为整数1, ..., 5.

.data

foo: .word 1,2,3,4,5

你用程序来把数组foo中的每个数加2再写回数组foo

3. 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)?

b. 在MARS中如何设置断点breakpoint?

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?

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.

1. **实验目的**

1. 学习使用汇编语言的编译器MARS

2. 熟悉并掌握一些简单的汇编指令

3. 提高将课堂所学知识进行实际应用的能力。

**三、实验内容**

**1.** 实验原理

1. MARS是MIPS汇编语言的模拟机，它能运行MIPS汇编程序。其设置：

Settings->Memory Configuration->Compact, Data at Address 0. 告诉了模拟机从哪个

内存地址开始存数据和代码。

2. 汇编程序用到的寄存器

寄存器号            符号名            用途  
  0                 始终为0     看起来象浪费,其实很有用  
  1                 at          保留给汇编器使用  
 2-3               v0,v1       函数返回值  
 4-7               a0-a3       前头几个函数参数  
  8-15              t0-t7       临时寄存器,子过程可以不保存就使用  
  24-25             t8,t9       同上  
  16-23             s0-s7       寄存器变量,子过程要使用它必须先保存  
                               然后在退出前恢复以保留调用者需要的值  
  26,27             k0,k1       保留给异常处理函数使用  
  28                gp          global pointer;用于方便存取全局或者静态变量  
  29                sp          stack pointer  
  30                s8/fp       第9个寄存器变量;子过程可以用它做frame pointer  
  31                 ra         返回地址

3. 用到的汇编指令：

.text address 指定了后续编译出来的内容放在代码段（可执行），地址从address

开始。address可以不写

.data address 指定读写数据段，用法同.text

.globl symbol  .globl 使得连接程序（ld）能够识别 symbl ,声明symbol是全

局可见的。标号\_start是GNU链接器用来指定第一个要执行指令

所必须的,同样的是全局可见的(并且只能出现在一个模块

中)     例如：         .global \_start    #定义 \_start 为外部程序可

以访问的标签...

变量声明： 例如：

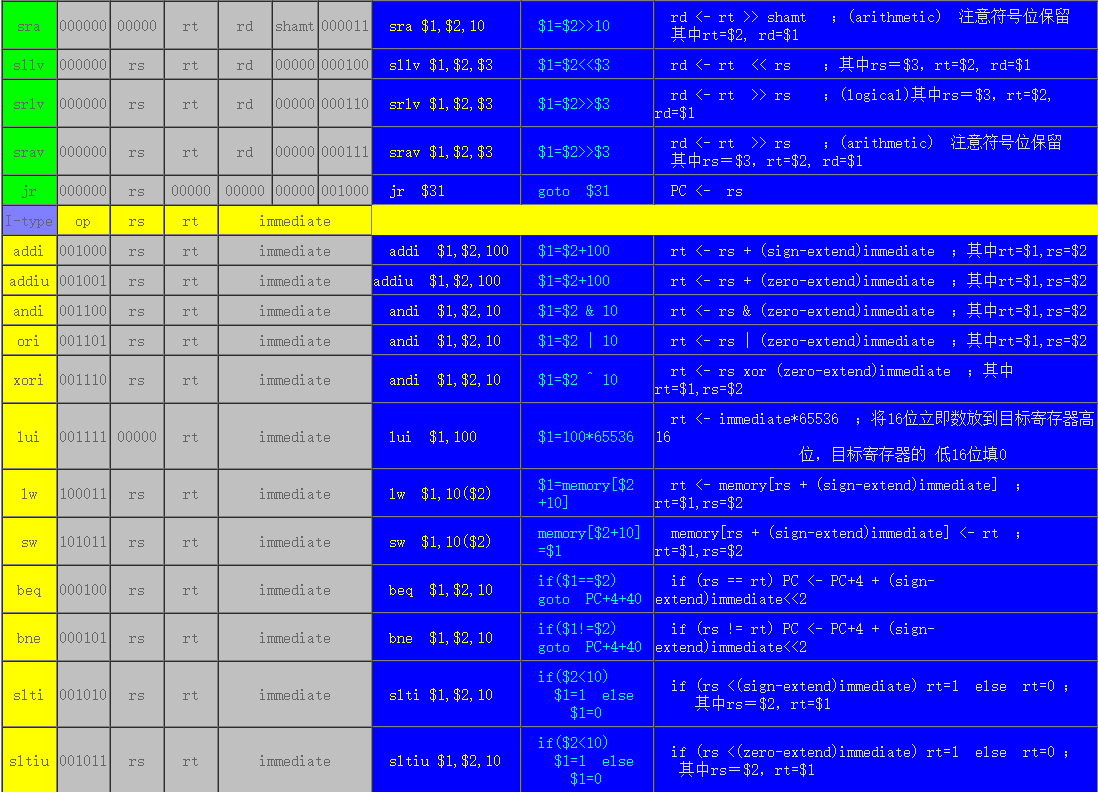
.data 0x0

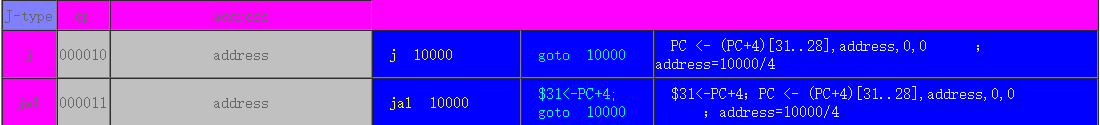
sum: .space 4

i: .space 4

a: .word 7,8,9,10,8 #（数组）







3.系统调用System calls

参数所使用的寄存器：$v0， $a0,  $a1

返回值使用： $v0

下表给出了系统调用中对应功能，代码，参数机返回值

|  |  |  |  |
| --- | --- | --- | --- |
| Service | Codein $v0  对应功能的调用码 | Arguments  所需参数 | Results  返回值 |
| print\_int  打印一个整型 | $v0 = 1 | $a0 = integer to be printed  将要打印的整型赋值给 $a0 |  |
| print\_float  打印一个浮点 | $v0 = 2 | $f12 = float to be printed  将要打印的浮点赋值给 $f12 |  |
| print\_double  打印双精度 | $v0 = 3 | $f12 = double to be printed  将要打印的双精度赋值给 $f12 |  |
| print\_string | $v0 = 4 | $a0 = address of string in memory  将要打印的字符串的地址赋值给 $a0 |  |
| read\_int | $v0 = 5 |  | integer returned in $v0  将读取的整型赋值给 $v0 |
| read\_float  读取浮点 | $v0 = 6 |  | float returned in $v0  将读取的浮点赋值给 $v0 |
| read\_double  读取双精度 | $v0 = 7 |  | double returned in $v0  将读取的双精度赋值给 $v0 |
| read\_string  读取字符串 | $v0 = 8 | $a0 = memory address of string input buffer  将读取的字符串地址赋值给 $a0  $a1 = length of string buffer (n)  将读取的字符串长度赋值给 $a1 |  |
| sbrk  应该同C中的sbrk()函数  动态分配内存 | $v0 = 9 | $a0 = amount  需要分配的空间大小（单位目测是字节 bytes） | address in $v0  将分配好的空间首地址给 $v0 |
| exit  退出 | $v0 =10 |  |  |

**2.** 实验步骤

1. 下载并打开MARS，并做好他的配置：Settings->Memory Configuration->Compact,

Data at Address 0.

2. 分别打开Sum.asm和Sum.array 进行运行调试，观察每一步各个寄存器的变化，同

时更好地理解每一条MIPS指令的意义

3. Sum.array的修改：

原代码本质是for(i=0;i<5;i++)sum+=i;

我们的目标是改成for(i=5;i<11;i++)sum+=i;

main中初始化：add $9,$0,$0 -> add addi $9,$0,5 #这一步初始化i=5

loop中：slti $10,$9,5 -> slti $10,$9,11 #这一步将i<5改为i<11

其余代码不变，完整代码见附件。

4. SumArray.asm的修改：

题目要求我们将.word 改成 .half。这里我们要明白一点，halfword所占内

存是word的一半。除了将声明变量时的word改为halfword 和将所有sw，lw

改为sh，lh外,loop中第一句sll $10, $9, 2 要改为sll $10, $9, 1。这一句的

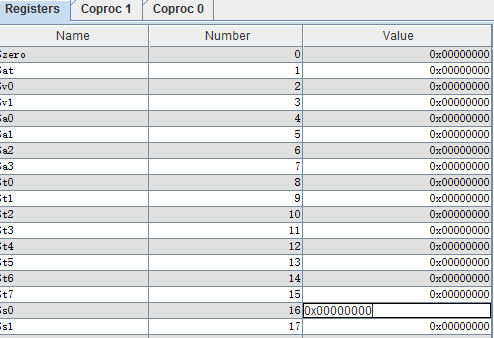
意思在于寻找得到a[i]的地址，原本word类型每个a[i]占4个字节，改为half之

后每个a[i]占2个字节。

5.编写程序的第一题：

将题目给出的代码按照顺序补充完整即可，详见附件。MARS中手动设置$s0和

$s1的值的方法是，先将代码编译和可以看到如下界面：



在运行前先双击后面的value就可以手动设置值。

6.编写程序第二题：

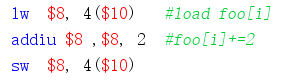
这题有我采取用循环去写。

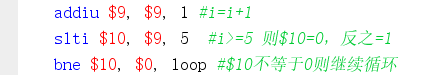
声明一个变量i变初始化为0：



运行时将i存储在$9中：

循环中i\*4得到a[i]真正的地址：

 取出foo[i]并+2再存回去：

 i+1并判断是否推出循环：

**3. 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)?

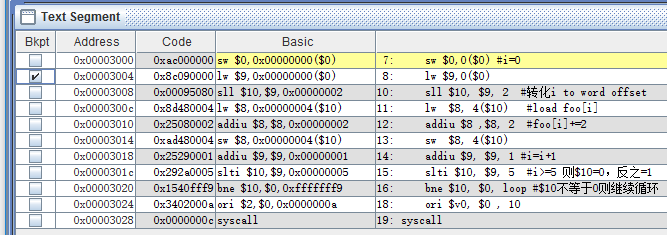
答：在汇编中，.data 是数据段的标识，即其后面可以声明变量。.word 是“双字

节”的标识，可以声明双字节变量。.text是“文本”的标识，后面跟真正的程

序代码。

b. 在MARS中如何设置断点breakpoint?

答：编译后在如下界面左边Bkpt列上选择需要加断点的行打勾即可



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?

答：点 可以继续运行，同时这个按钮也是单步运行，点一次运行一次代码。

单步调试的时候，首先让程序运行到断电处，然后点上面这个按钮一步一步继

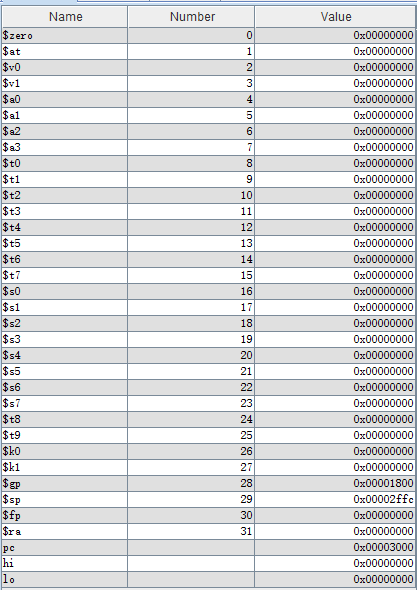
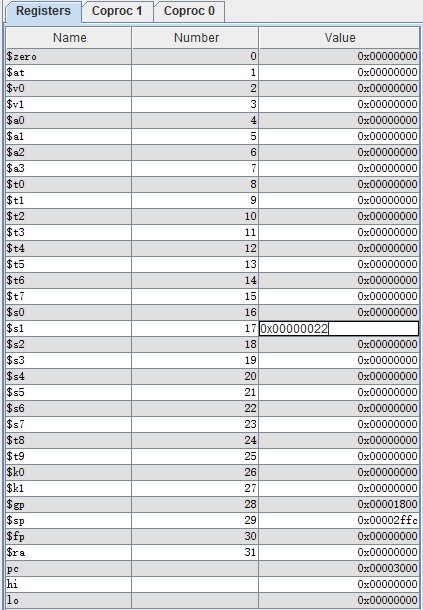
续运行。 这个按钮可以实现单步回退，返回上一句代码。

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.

答：在运行界面右边可以查看register的值，如下左图。修改寄存器的值只需要在

下左图这个界面，双击寄存器对应的value栏，就可以手动修改寄存器的值。

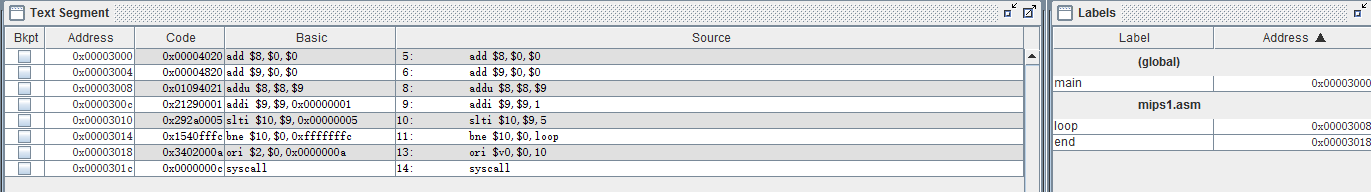
如下右图，修改$s1的值。

**四、实验结果**

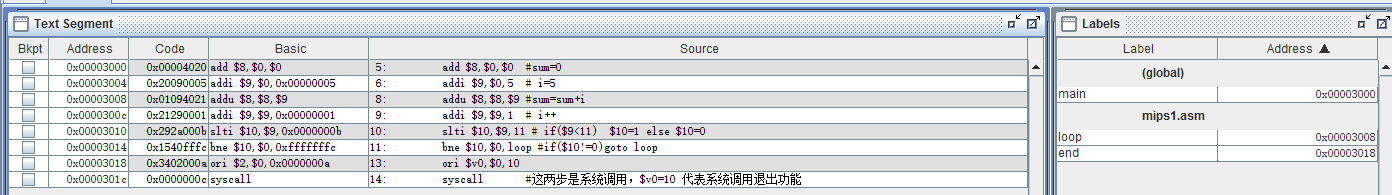
**1.Sum.asm：**

**源代码：**



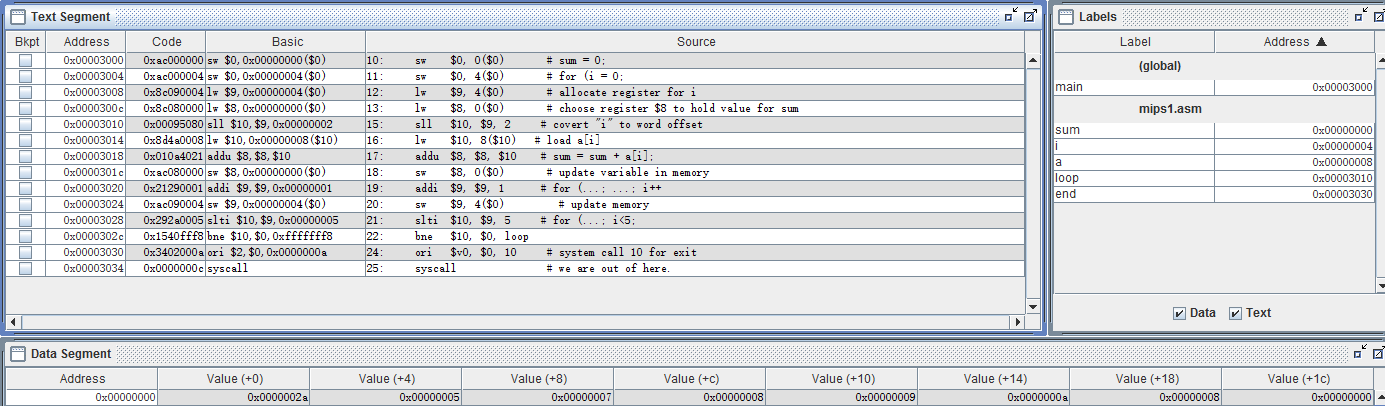


**修改后：**



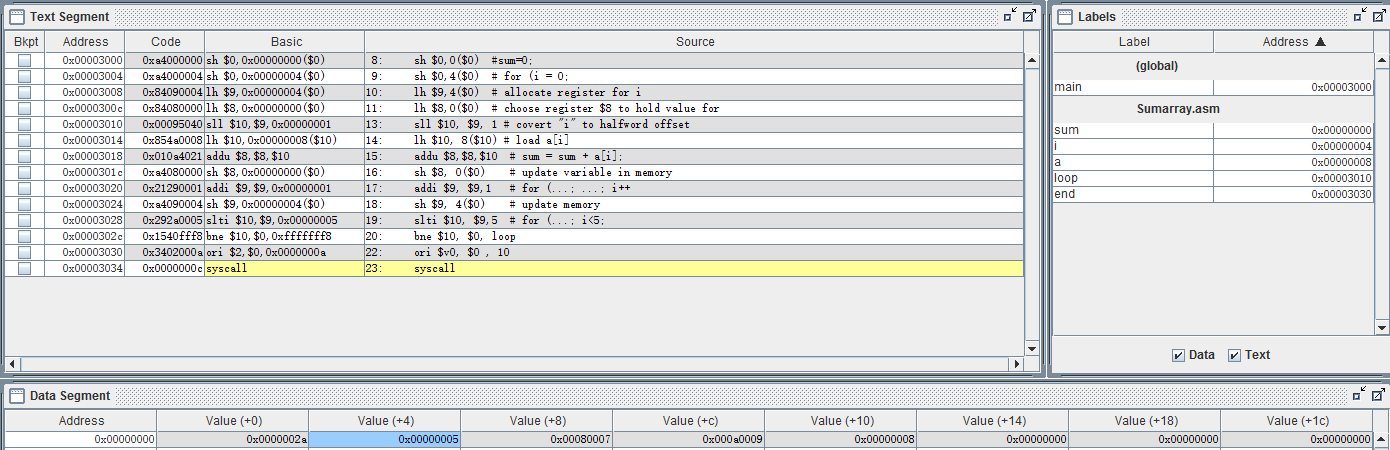


**2.Sumarray.asm:**

**源代码：**



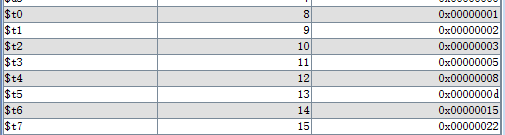
**修改后：**



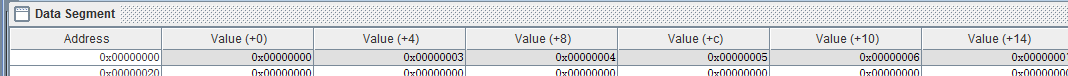


**3.编写程序第一题:**





**4.编写程序第二题:**



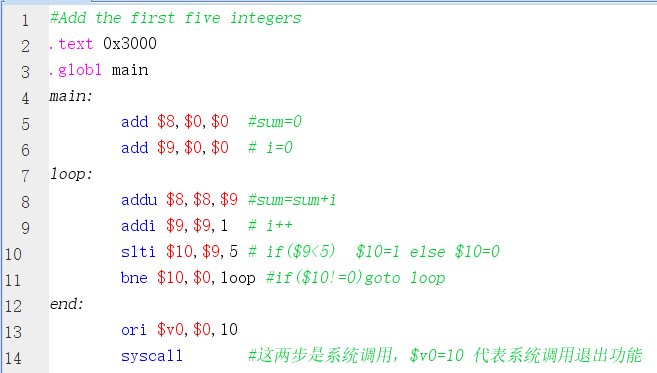
**五、实验感想**

通过这次实验，我对MIPS汇编代码的基本格式有了了解。我们在理论课上学习了一些汇编指令，然后我一直对他们一知半解，掌握地不好，做理论作业的时候很迷茫。但是通过这次实验，通过对每个代码的跟踪运行，对数据的观察，以及到后面自己动手编写代码，我觉得自己对这些指令有了一个较清晰的理解。

**附录：**

**1.Sum.asm：**

**原代码：**

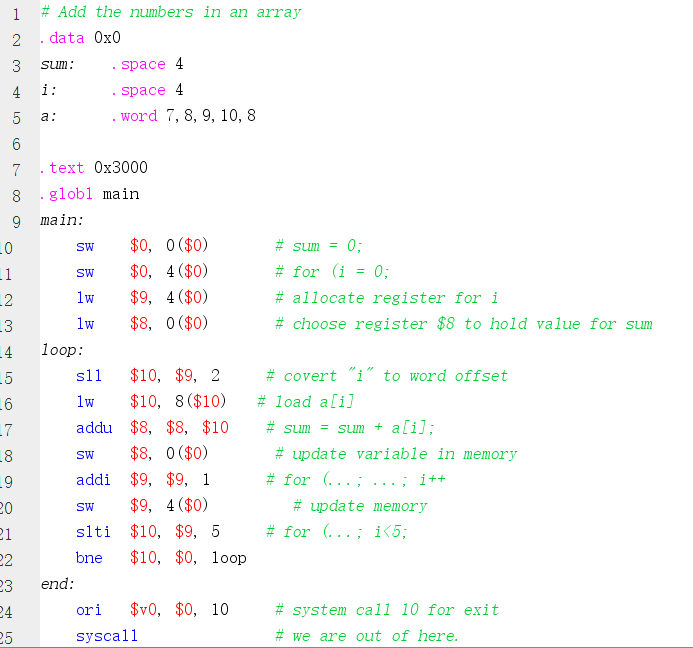


**修改后：**

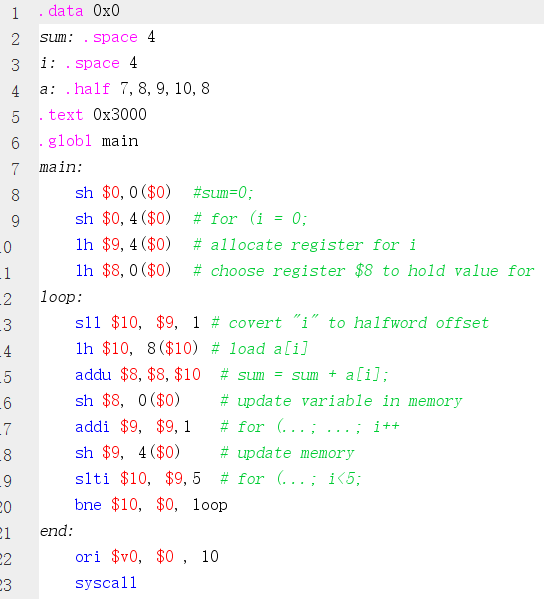


**2.SumArray.asm：**

**原代码：**



**修改后：**



**3.编写程序第一题： 4.编写程序第二题**

