中山大学数据科学与计算机学院本科生实验报告

（2019 学年秋季学期）

课程名称：**计算机组成原理实验**　　任课教师：郭雪梅　　助教：汪庭葳、刘洋旗

年级&班级	2018 级计科一班	专业(方向)	计算机类
学号	18340013	姓名	陈琮昊
电话	15734867907	Email	1062080021@qq.com
开始日期	2019.10.9	完成日期	2019.10.15

一、实验题目：MIPS 实验 2

二、实验目的：继续熟悉 MARS 仿真器的使用，学习用它来编写、运行和调试程序

三、实验内容

1.实验步骤：

（I）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1,2,3,4,5。你用程序来把数组foo中的每个数加2再写回数组foo.
（II）从键盘输入两个数，计算并输出这两个数的和.

（III）计算 $1^2+2^2+...+100^2$ ，并输出结果.

（IV）编写两个版本的 first1pos 函数，在$a0 中给定一个数，而在$v0 中返回$a0 字中最左边的非零位的位置. 如果$a0 的值是 0，在$v0 中存-1 . 在查找此位置的过程中,允许你修改$a0 值.位置从 0 (最右位)到 31 (符号位). 其中一种解应该重复移位$a0，每次移位后,检查符号位. 另一种方法是初始时使用 0x80000000 作为掩码,并不断右移该掩码来检查$a0 的每一位.

2. 实验原理：汇编语言的语法

四、实验结果：

(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)?

.data 的含义:后续项存储在下一个可用地址的数据段中。

.word 的含义:存储列出的值为 32 位在字边界上。

.text 的含义:后续项存储在下一个可用地址的文本段中。

(2) 在 MARS 中如何设置断点 breakpoint?

如何设置断点：在 Execute 一栏中最左侧有一个“Bkpt”，在需要设置断点的那一行勾选即可。

(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?

找到该图标即可继续执行：  找到该图标即可单步调试代码： 

(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。

MARS 页面右侧一栏 Registers 中的 Value 一栏查看即可。在 Value 栏双击即可修改。

4 个练习的代码详见附录。

五、实验感想：本次实验主要是继续用 MARS 来编译 MIPS 程序，这次是要自己从头到尾编写一个完整的程序。首先要牢记指令，使用正确的指令来表达自己的意思。然而问题比较大的地方就是寄存器的使用，用着用着可能就混淆了，乱成一片，导致结果错误。因此在编写程序的时候一定要规范的使用寄存器，保持头脑清晰。

附录（流程图，注释过的代码）：

(1) 练习 1 的代码如下：

```
1.      .data
2.          foo:    .word 1,2,3,4,5
3.      .text
4.          main:
5.              la     $t0,foo              #将数组读入
6.              lw     $t1,0($t0)           #$t1 存数组元素
7.          end:
8.              addiu   $t1, $t1,2          #数组每个元素+2
9.              sw      $t1, 0($t0)
10.             addiu   $t0,$t0,4           #指向下一个
11.             lw      $t1,0($t0)
12.             bne     $t1,$zero,end #当读到后面值为 0 时意味着循环结束
```

结果如下：

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000003	0x00000004	0x00000005	0x00000006	0x00000007	0x00000000	0x00000000	0x00000000

由图可知，原数组每个数都加了 2，结果正确。

(2) 练习 2 的代码如下：

```
1.      .data
2.           Str1:   .asciiz "Enter 2 numbers:"
3.      Str2:   .asciiz "The sum is "
4.      .text
5.      main:
6.         ori $v0, $0, 4                #System call code 4 for printing a string
7.         la $a0, Str1                  #address of Str1 is in $a0
8.         syscall                       #print the string
9.         add $t0,$v0,$zero
10.        ori $v0, $0, 5
11.        syscall              #System call code 5 for read integer,$v0 contains integer read
12.        add $t0,$v0,$zero
13.        ori $v0, $0, 5
14.        syscall              #System call code 5 for read integer,$v0 contains integer read
15.        add $t1, $v0, $t0             #calculate the sum
16.        ori $v0, $0, 4                #System call code 4 for printing a string
17.        la $a0, Str2                  #address of Str2 is in $a0
18.        syscall                       #print the string
19.        ori $v0, $0, 1        #System call code 4 for print integer,$a0 = integer to print
20.       add $a0,$t1,$zero          #put sum in $a0
21.       syscall                       #print the sum
22.      exit: ori $v0, $0, 10          #System call code 10 for exit
23.       syscall                       #print the sum
```

结果如下：

```
Enter 2 numbers:3
5
The sum is 8
```

```
Enter 2 numbers:10
11
The sum is 21
```

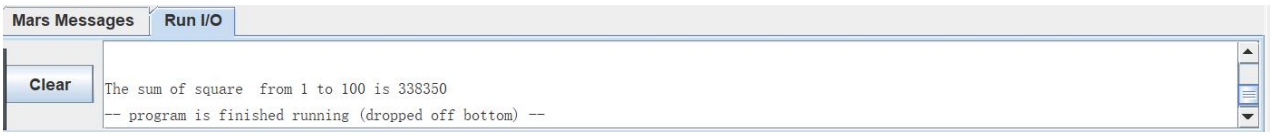
可以看出输出为输入两个数的和，实验成功。

(3) 练习 3 的代码如下：

```
1.      .data
2.      .align 2
```

```
3.    Str: .asciiz "The sum of square  from 1 to 100 is "
4.    .text
5.    main:
6.    addi $t1,$zero,100
7.    ori $v0, $0, 4          #System call code 4 for printing a string
8.    la $a0, Str             #address of Str1 is in $a0
9.    syscall                 #print the string
10.   li $t0,0
11.   li $a0,0
12.   loop:
13.   addi $t0,$t0,1          #n=n+1
14.   mult $t0,$t0            #calculate n²
15.   mflo $a2                #the result must be in lower bits
16.   add $a0,$a0,$a2         #add n²
17.   bne  $t0,$t1,loop       #loop until n=100
18.   li $v0,1
19.   syscall
```

结果如下：



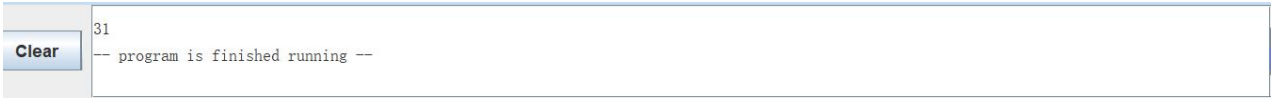
(4) 练习 4 的代码如下：

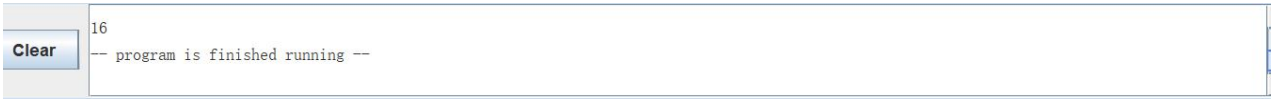
第一种方法：当输入为 0 时由题意输出-1；当输入不为 0 时，由于最高位是符号位，所以如果输入小于 0，则最高位必为 1，故此时直接输出 31；如果输入大于 0，则需要对输入的数进行左移，直到那个最先出现的 1 左移到符号位，即小于 0 时，可以进行判断；每左移 1 次，输出的结果要从 31 开始递减，则最终结果=31-左移次数。

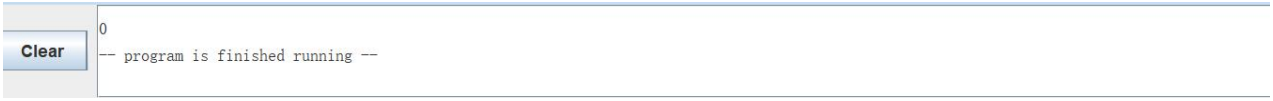
```
1.    .data
2.    .text
3.    main:
4.    lui   $a0,0x8000
5.    #lui   $a0,0x0001
6.    #li    $a0,1            #测例
7.    #add   $a0,$0,$0
8.    beq $a0,$0,end    #若输入为 0 跳转至 end
9.    li $t2,1
10.   li $t3,31              #最高位为 31
11.   loop:
12.   slt $a2,$a0,$0     #若输入不为 0
13.   beq $a2,$t2,en1  #当输入小于 0 时跳转至 en1
```

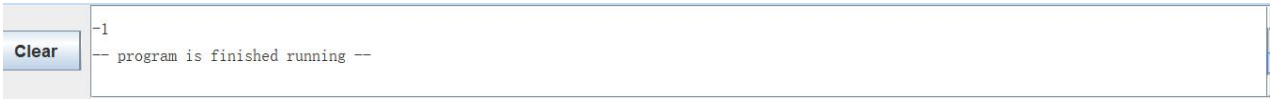
```
14.  beq $a2,$0,en2   #当输入大于 0 时跳转至 en2
15.  end:
16.  li $a0,-1          #由题意输入为 0 直接输出-1
17.  li $v0,1
18.  syscall
19.  li $v0,10
20.  syscall
21.  en1:
22.  add $a0,$0,$t3    #输入小于 0 时最高位必定为 1，故直接输出 31
23.  li $v0,1
24.  syscall
25.  li $v0,10
26.  syscall
27.  en2:
28.  sll $a0,$a0,1      #输入大于 0 时则通过左移 1 位判断，直到最高位出现 1 为止
29.  addi $t3,$t3,-1    #每左移 1 位意味着寻找的位数的序号要减 1
30.  j loop
```

结果如下：









四个测例的结果正确。

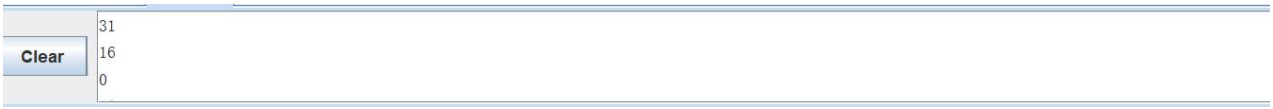
第二种方法：用 0x80000000 作为掩码进行判断，该数第一位是 1 后面都是 0。跟输入的数进行按位与，只要没有到最先出现的 1 时，相与的结果都是 0；所以一直将掩码进行右移，直到移动到最先出现 1 的那个位置时相与的结果才不会为 0。因此 0 可以作为一个判断条件。（最初的想法是两个数进行异或，但是后来发现如果输入的数中有多个 1 时，异或并不能找到判断条件，最终是选择将两个数相与）代码如下：

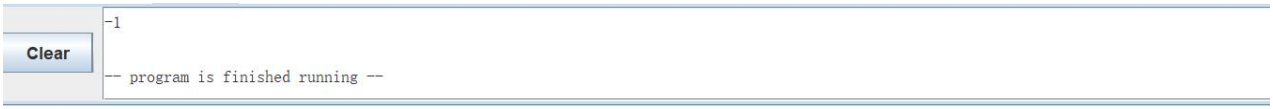
```
1.  .data
2.  .text
3.  main:
```

```
4.      li    $t0,31              #位数
5.      li    $t1,0x80000000 #掩码
6.      lui   $a0,0x8000
7.      jal   first1pos
8.      jal   printv0
9.      lui   $a0,0x0001
10.     jal   first1pos
11.     jal   printv0
12.     li    $a0,1
13.     jal   first1pos
14.     jal   printv0
15.     add   $a0,$0,$0
16.     jal   first1pos
17.     jal   printv0
18.     li    $v0,10
19.     syscall
20.
21.   first1pos:
22.       beq   $a0,0,end   #如果$a0 为 0，则跳转到 end
23.       and   $t2,$a0,$t1 #掩码和输入的$a0 按位与
24.       beq   $t2,0,en1   #如果等于 0，则跳转到 en1
25.       bne   $t2,0,en2   #如果不等于 0，则跳转到 en2
26.
27.   end:
28.   la   $v0,-1             #根据题意，输入为 0，输出为-1
29.   jr   $ra
30.   en1:
31.   srl   $t1,$t1,1         #如果为 0，说明没检测到 1，需要将掩码右移
32.   addi   $t0,$t0,-1       #没检测到 1 则需要将结果对应的$t0 递减
33.   j   first1pos           #继续循环
34.   en2:
35.   la   $v0,($t0)          #如果不为 0，则说明已经检测到了 1，输出此时 1 对应的所在位
36.   jr   $ra
37.
38.
39.
40.   printv0:
41.     addi    $sp,$sp,-4
42.     sw    $ra,0($sp)
43.     add   $a0,$v0,$0
44.     li    $v0,1
45.     syscall
46.     li    $v0,11
47.     li    $a0,'\n'
48.     syscall
49.     lw    $ra,0($sp)
```

```
50.     addi   $sp,$sp,4
51.     jr   $ra
```

输出结果如图：





四个测例的结果正确。