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	2019.10.16	完成日期	2019.10.22

一、实验题目：MIPS 实验 3

二、实验目的：学会用堆栈进行函数调用以及查看内存空间分配

三、实验内容

1. 实验步骤：

（i）编写汇编代码完成 swap 函数。将变量 temp 保存在栈中，不能使用$t0（或者其它寄存器）来对应 temp。

```
void swap (int *px, int *py) {
    int temp;
    temp = *px;
    *px = *py;
    *py = temp;
}
```

（ii）汇编一个简短的 MIPS 程序来实现以下 C 代码：

```
int main ()
{
int k,Y;
int Z[50];
Y=56;
for(k=0;k<50;k++)
Z[k]=Y-16*(k/4+210);
}
```

2. 实验原理：汇编语言的语法

四、实验结果：见附录。

五、实验感想：本次实验主要是练习使用堆栈以及分析数据段和代码段的内存映像，对于堆栈的问题还需要通过练习深刻的理解一下，而第二个实验相对来说就很简单了，只要能够把指令写出来运行成功问题就迎刃而解了。自己还要在堆栈的使用上下功夫。

附录（流程图，注释过的代码）：

(1) 练习 1 的代码如下：

```
1.  .data
2.  n1: .word    14
3.  n2: .word    27
4.
5.  .text
6.  main:
7.      la     $a0,n1
8.      la     $a1,n2
9.      jal    swap
10.     li$v0,1    # print n1 and n2; should be 27 and 14
11.     lw     $a0,n1
12.     syscall
13.     li$v0,11
14.     li$a0,' '
15.     syscall
16.     li$v0,1 #$a0:显示的整数值
17.     lw     $a0,n2
18.     syscall
19.     li$v0,11 #$a0:显示的字符
20.     li$a0,'\n'
21.     syscall
22.     li$v0,10   # exit
23.     syscall
24.
25. swap:    move    $fp, $sp #FRAME POINTER NOW POINTS TO THE TOP OF STACK
26.     addiu $sp,$sp,-16 # ALLOCATE 16 BYTES IN THE STACK
27.     sw $a0,0($sp)   #入栈 n1 地址
28.     sw $a1,4($sp)   #入栈 n2 地址
29.     lw $a0,0($a0)   #将值存入$a0
30.     lw $a1,0($a1)   #将值存入$a1
31.     sw $a0,8($sp)   #入栈 n1 的值
32.     lw $a0,0($sp)    #存 n1 的地址
33.     sw $a1,0($a0)   #改变 n1 的值
34.     lw $a1,4($sp)
35.     lw $a0,8($sp)
36.     sw $a0,0($a1)
37.     addiu $sp,$sp,16
38.     move $fp,$sp
```

```
39.     jr $ra
40. addiu $sp,$sp,16
41.             jr $31
```

结果如下，可知成功交换两数。

```
27 14

-- program is finished running --
```

(2) 练习 2 的代码如下：

第一种方法：

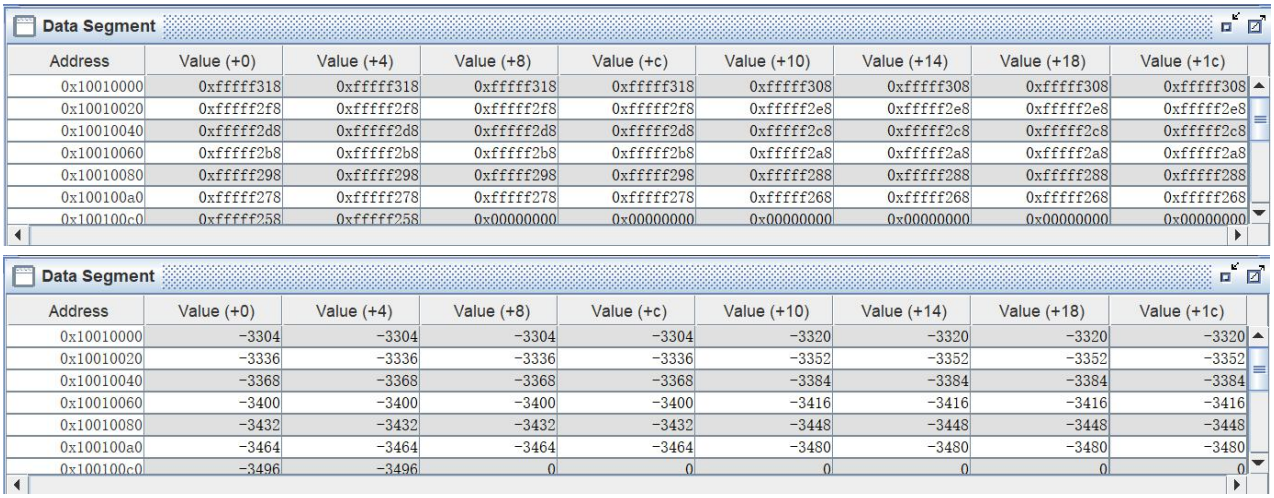
```
42. .data
43. z: .space 200
44. .text
45. main:
46. la $s0,z
47. li $t0,0
48. li $t1,56
49. loop:                    #该方法为先判断再计算
50. slti $t2,$t0,50          #判断 k 是否小于等于 50
51. beq $t2,$0,done          #如果大于 50 则跳到 done
52. srl $t3,$t0,2            #k/4
53. addi $t3,$t3,210         #k/4-210
54. sll $t3,$t3,4            #16*(k/4-210)
55. sub $t3,$t1,$t3          #56-16*(k/4-210)
56. sw $t3,0($s0)            #将结果放入数组
57. addi $s0,$s0,4           #指向下一个数组元素
58. addi $t0,$t0,1           #k++
59. j loop                   #如果小于等于 50 则继续循环
60. done:
61. li $v0,10                #大于 50 则结束
62. syscall
```

第二种方法：

```
63. .data
64. z: .space 200
65. .text
66. main:
67. la $s0,z
68. li $t0,0
69. li $t1,56
70. loop:
71. srl $t3,$t0,2
72. addi $t3,$t3,210
73. sll $t3,$t3,4
```

```
74. sub $t3,$t1,$t3
75. sw $t3,0($s0)
76. addi $s0,$s0,4
77. addi $t0,$t0,1
78. slti $t2,$t0,50
79. beq $t2,$0,done    #该方法与第一种方法几乎一样，只不过是先计算后判断
80. j loop
81. done:
82. li $v0,10
83. syscall
```

结果如图：



Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0xfffff318	0xfffff318	0xfffff318	0xfffff318	0xfffff308	0xfffff308	0xfffff308	0xfffff308
0x10010020	0xfffff2f8	0xfffff2f8	0xfffff2f8	0xfffff2f8	0xfffff2e8	0xfffff2e8	0xfffff2e8	0xfffff2e8
0x10010040	0xfffff2d8	0xfffff2d8	0xfffff2d8	0xfffff2d8	0xfffff2c8	0xfffff2c8	0xfffff2c8	0xfffff2c8
0x10010060	0xfffff2b8	0xfffff2b8	0xfffff2b8	0xfffff2b8	0xfffff2a8	0xfffff2a8	0xfffff2a8	0xfffff2a8
0x10010080	0xfffff298	0xfffff298	0xfffff298	0xfffff298	0xfffff288	0xfffff288	0xfffff288	0xfffff288
0x100100a0	0xfffff278	0xfffff278	0xfffff278	0xfffff278	0xfffff268	0xfffff268	0xfffff268	0xfffff268
0x100100c0	0xfffff258	0xfffff258	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-3304	-3304	-3304	-3304	-3320	-3320	-3320	-3320
0x10010020	-3336	-3336	-3336	-3336	-3352	-3352	-3352	-3352
0x10010040	-3368	-3368	-3368	-3368	-3384	-3384	-3384	-3384
0x10010060	-3400	-3400	-3400	-3400	-3416	-3416	-3416	-3416
0x10010080	-3432	-3432	-3432	-3432	-3448	-3448	-3448	-3448
0x100100a0	-3464	-3464	-3464	-3464	-3480	-3480	-3480	-3480
0x100100c0	-3496	-3496	0	0	0	0	0	0

数据段内存映像：

内存地址（16 进制）	变量名	值	内存地址（16 进制）	变量名	值
0x10010000	Z[0]	0xfffff318	0x10010064	Z[25]	0xfffff2b8
0x10010004	Z[1]	0xfffff318	0x10010068	Z[26]	0xfffff2b8
0x10010008	Z[2]	0xfffff318	0x1001006c	Z[27]	0xfffff2b8
0x1001000c	Z[3]	0xfffff318	0x10010070	Z[28]	0xfffff2a8
0x10010010	Z[4]	0xfffff308	0x10010074	Z[29]	0xfffff2a8
0x10010014	Z[5]	0xfffff308	0x10010078	Z[30]	0xfffff2a8
0x10010018	Z[6]	0xfffff308	0x1001007c	Z[31]	0xfffff2a8
0x1001001c	Z[7]	0xfffff308	0x10010080	Z[32]	0xfffff298
0x10010020	Z[8]	0xfffff2f8	0x10010084	Z[33]	0xfffff298
0x10010024	Z[9]	0xfffff2f8	0x10010088	Z[34]	0xfffff298
0x10010028	Z[10]	0xfffff2f8	0x1001008c	Z[35]	0xfffff298
0x1001002c	Z[11]	0xfffff2f8	0x10010090	Z[36]	0xfffff288
0x10010030	Z[12]	0xfffff2e8	0x10010094	Z[37]	0xfffff288
0x10010034	Z[13]	0xfffff2e8	0x10010098	Z[38]	0xfffff288

0x10010038	Z[14]	0xfffff2e8	0x1001009c	Z[39]	0xfffff288
0x1001003c	Z[15]	0xfffff2e8	0x100100a0	Z[40]	0xfffff278
0x10010040	Z[16]	0xfffff2d8	0x100100a4	Z[41]	0xfffff278
0x10010044	Z[17]	0xfffff2d8	0x100100a8	Z[42]	0xfffff278
0x10010048	Z[18]	0xfffff2d8	0x100100ac	Z[43]	0xfffff278
0x1001004c	Z[19]	0xfffff2d8	0x100100b0	Z[44]	0xfffff268
0x10010050	Z[20]	0xfffff2c8	0x100100b4	Z[45]	0xfffff268
0x10010054	Z[21]	0xfffff2c8	0x100100b8	Z[46]	0xfffff268
0x10010058	Z[22]	0xfffff2c8	0x100100bc	Z[47]	0xfffff268
0x1001005c	Z[23]	0xfffff2c8	0x100100c0	Z[48]	0xfffff258
0x10010060	Z[24]	0xfffff2b8	0x100100c4	Z[49]	0xfffff258

代码段内存映像：

Address	Code	Basic
0x00400000	0x3c011001	lui $1, 4097
0x00400004	0x34300000	ori $16, $1, 0
0x00400008	0x24080000	addiu $8, $0, 0
0x0040000c	0x24090038	addiu $9, $0, 56
0x00400010	0x290a0032	slti $10, $8, 50
0x00400014	0x11400008	beq $10, $0, 8
0x00400018	0x00085882	srl $11, $8, 2
0x0040001c	0x216b00d2	addi $11, $11, 210
0x00400020	0x000b5900	sll $11, $11, 4
0x00400024	0x012b5822	sub $11, $9, $11
0x00400028	0xae0b0000	sw $11, 0($16)
0x0040002c	0x22100004	addi $16, $16, 4
0x00400030	0x21080001	addi $8, $8, 1
0x00400034	0x08100004	j 0x00400010
0x00400038	0x2402000a	addiu $2, $0, 10
0x0040003c	0x0000000c	syscall