|  |  |
| --- | --- |
|  | 实验六 MIPS汇编程序设计 |

**一、实验目的**

1.熟悉MIPS汇编程序开发环境，学习使用MARs工具。知道如何查看内存空间分配。

2.了解C语言语句与汇编指令之间的关系。

3. 掌握MIPS汇编程序设计，掌握MARs的调试技术。

4.了解MIPS汇编语言与机器语言之间的对应关系 。

5.熟悉常见的MIPS汇编指令

6.掌握程序的内存映像。

**熟悉MIPS汇编程序开发环境，学习使用MARs工具。知道如何查看内存空间分配**

**二、实验内容**

1. 编写 swap 函数实现两个数之间的交换

void swap (int \*px, int \*py) {

    int temp;

    temp = \*px;

    \*px = \*py;

    \*py = temp;

}

文件[swap.s](http://xgxy.cug.edu.cn/rjgcx/lzw/COD/lab2/swap.txt) 提供了一个调用你将编写的程序的代码模板。你可以在此程序上，添加你要写的swap代码，以方便测试。不能直接用寄存器完成数值交换， 而是要利用栈将$t0先储存起来，即只能使用一个$t0完成数据交换。

如果使用寄存器交换本题目很简单，本练习的目的是考查临时变量的栈存储。

提示: 一共需要使用6条lw/sw指令。完成后给老师解释你的代码.

[swap.s](http://xgxy.cug.edu.cn/rjgcx/lzw/COD/lab2/swap.txt)

.data

n1: .word 14

n2: .word 27

.text

main:

la $a0,n1

la $a1,n2

jal swap

li $v0,1 # print n1 and n2; should be 27 and 14

lw $a0,n1

syscall

li $v0,11

li $a0,' '

syscall

li $v0,1 #$a0:显示的整数值

lw $a0,n2

syscall

li $v0,11 #$a0:显示的字符

li $a0,'\n'

syscall

li $v0,10 # exit

syscall

swap: move $fp, $sp #FRAME POINTER NOW POINTS TO THE TOP OF STACK

addiu $sp,$sp,-16 # ALLOCATE 16 BYTES IN THE STACK

# your code goes here

…

addiu $sp,$sp,16

jr $31

2. 用汇编程序实现以下伪代码， 要求使用移位指令实现乘除法运算

Int main ()

{

Int K,Y;

Int Z[50];

Y=56;

For(k=0;k<50;K++) Z[k]=Y-16\*(k/4+210);

}

**三、程序设计及分析**

<练习一>  
堆栈调用框架：

复杂情况下堆栈调用框架：

sw $fp, ($sp) # push old frame pointer (dynamic link)

move $fp, $sp #frame pointer now points to the top of stack

subu $sp, $sp, 500 # allocate say 500 bytes in the stack

# (for frame size = 500)

sw $ra, -4($fp) # save return address in frame

sw $v0, -8($fp) # save static link in frame

**叶子子程序不需要保存返回地址，需要入栈保护**$t0，$t1的情况：

subu $sp, $sp, 8 # allocate say 8 bytes in the stack

sw $t0, 0($sp) # push x

sw $t1,4 ($sp) # push c

lw $t0, 0($sp) # pop x

lw $t1,4 ($sp) # push c

addu $sp, $sp, 8 # pop stack

$a0存放n1在内存中的地址，$a1存放n2在内存中的地址，也就是C 语言中的指针&n1和&n2。$a0和$a1作为参数传递给swap函数。

熟悉 MIPS 指令中关于函数调用方面的内容， 尤其是入栈和出栈。

**<练习二>**

1.C语言分析：

有两个变量是int型，一个数组型；还有一个循环执行过程。

2.汇编程序实现分析：

首先需要定义用户数据段，获得一个内存空间作为数组空间。再选定几个寄存器作为K，Y以及输出，其中输出输出和Y可以合用一个寄存器。

3.设计思路：

分配完空间地址后，最重要的是完成循环控制。循环控制有两个思路：可以是先判断后循环；或者是先循环后判断

即如图

开始

开始

赋值

int K, Y ;int Z[50] ;K=0;Y = 56;

赋值

int K, Y ;int Z[50] ;K=0;Y = 56;

计算

Z[K] = Y - 16 \* ( K / 4 + 210) ;

K<50

否 是

计算

Z[K] = Y - 16 \* ( K / 4 + 210) ;

K=**k**+1

否

K<50

结束

是

K=**k**+1

结束

slti $t2,$t0,50 #判断k是否小于50，

beq $t2,$0，#是则结束

#否,循环

结束

slti $t2,$t0,50 #判断k是否于50

beq $t2,$t3（$t2=1循环，否则结束。）

1. 程序实现及调试分析
2. 汇编程序代码实现：

方法一

方法二

1. 调试过程

编写程序：详细见代码

装载程序

如果没有错误，便运行。

运行之后点击不同的窗口便可得到我们想要的结果。具体详细结果如下图

内存占用情况映像

分析：数组地址

数据段内存映像

表格如下（数值都采用16进制）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 内存地址（16进制） | 变量名 | 值 | 内存地址（16进制） | 变量名 | 值 |
|  | **Z[0]** |  |  | **Z[25]** |  |
|  | **Z[1]** |  |  | **Z[26]** |  |
|  | **Z[2]** |  |  | **Z[27]** |  |
|  | **Z[3]** |  |  | **Z[28]** |  |
|  | **Z[4]** |  |  | **Z[29]** |  |
|  | **Z[5]** |  |  | **Z[30]** |  |
|  | **Z[6]** |  |  | **Z[31]** |  |
|  | **Z[7]** |  |  | **Z[32]** |  |
|  | **Z[8]** |  |  | **Z[33]** |  |
|  | **Z[9]** |  |  | **Z[34]** |  |
|  | **Z[10]** |  |  | **Z[35]** |  |
|  | **Z[11]** |  |  | **Z[36]** |  |
|  | **Z[12]** |  |  | **Z[37]** |  |
|  | **Z[13]** |  |  | **Z[38]** |  |
|  | **Z[14]** |  |  | **Z[39]** |  |
|  | **Z[15]** |  |  | **Z[40]** |  |
|  | **Z[16]** |  |  | **Z[41]** |  |
|  | **Z[17]** |  |  | **Z[42]** |  |
|  | **Z[18]** |  |  | **Z[43]** |  |
|  | **Z[19]** |  |  | **Z[44]** |  |
|  | **Z[20]** |  |  | **Z[45]** |  |
|  | **Z[21]** |  |  | **Z[46]** |  |
|  | **Z[22]** |  |  | **Z[47]** |  |
|  | **Z[23]** |  |  | **Z[48]** |  |
|  | **Z[24]** |  |  | **Z[49]** |  |

运行结果显示

代码段内存映像

地址 机器码 汇编指令

[00400014] 0c100009  **jal 0x00400024 [main]**

[00400018] 00000000  **nop**

[0040001c] 3402000a  **ori $2, $0, 10**            
[00400020] 0000000c  **syscall**                  
[00400024] 3c101001  **lui $16, 4097 [z]**

…