|  |  |
| --- | --- |
| **计算机结构与组成** | **实验课3（7周）** |

关于STACK

Function/procedure arguments are pushed in the stack before the function call. If this is a function, then an empty placeholder (4 bytes) should be pushed in the stack before the function call; this will hold the result of the function.

Each procedure/function should begin with the following code (prologue):

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

**最简单情况下，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

**实验目的**

本次实验希望给大家更多关于函数调用练习的机会，特别是哪些东西需要入栈，另外还有逻辑操作的练习.

**阅读**

书的第2.5-2.7节.

**基本准备**

练习1

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。.

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

    int temp;

    temp = \*px;

    \*px = \*py;

    \*py = temp;

}

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

.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

.data

n1: .word 14

n2: .word 27

编写汇编代码完成上述程序。由于所有C程序的局部变量都保存在栈中，因此变量temp也应保存在栈中 (未优化时的情况)。

换言之，不能使用$t0（或者其它寄存器）来对应temp。 提示: 一共需要使用6条lw/sw指令。

如果允许使用$t0来保存temp变量，实现程序优化，本题可能会简单很多，本练习的部分目的是考查临时变量的栈存储。

完成后给老师解释你的代码.

**练习2**

编写两个版本的first1pos (starting from [first1pos.s](http://xgxy.cug.edu.cn/rjgcx/lzw/COD/lab2/first1pos.txt)) 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.

可以和你的同伴分别来做此工作, 其中一个同学做第一个版本,另一个同学做第二个版本. 工作完成后,向你的同伴解释你的代码, 然后把运行情况给老师检查.

[first1pos.s](http://xgxy.cug.edu.cn/rjgcx/lzw/COD/lab2/first1pos.txt))

main:

lui $a0,0x8000

jal first1pos

jal printv0

lui $a0,0x0001

jal first1pos

jal printv0

li $a0,1

jal first1pos

jal printv0

add $a0,$0,$0

jal first1pos

jal printv0

li $v0,10

syscall

first1pos: # your code goes here

printv0:

addi $sp,$sp,-4

sw $ra,0($sp)

add $a0,$v0,$0

li $v0,1

syscall

li $v0,11

li $a0,'\n'

syscall

lw $ra,0($sp)

addi $sp,$sp,4

jr $ra