Laboratory Exercise 7

Procedure calls, stack and parameters

--------------------------------------------------------------------------------------------------------

Assignment 1

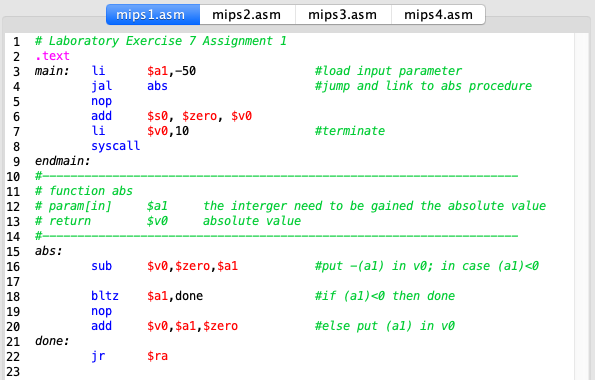
Home Assignment 1. Thay đổi các tham số đầu vào và quan sát bộ nhớ khi chạy chương trình từng bước. Hãy chú ý thanh ghi $ pc, $ ra để làm rõ quy trình gọi thủ tục (Tham khảo hình 7).

Tính giá trị tuyệt đối.

$a1 lưu giá trị cần tính, kết quả lưu $v0

So sánh tham số ở thanh ghi $a1 với 0. Nếu < 0 thì done để kết thúc.

**Trả lời:**



Đầu vào $a1 = -50



Kết quả $s0 = 50



🡪 Kết quả chính xác

(Giá trị $v0 sau lệnh gọi syscall trở thành 10)

Assignment 2

Home Assignment 2. Biên dịch và tải lên trình giả lập. Thay đổi các tham số đầu vào (thanh ghi $a0, $a1, $a2) và quan sát bộ nhớ khi chạy chương trình từng bước. Hãy chú ý thanh ghi $pc, $ra để làm rõ quy trình gọi thủ tục (Tham khảo hình 7).

Tìm giá trị lớn nhất trong 3 tham số truyền vào, lưu ở 3 thanh ghi $a0, $a1, $a2.

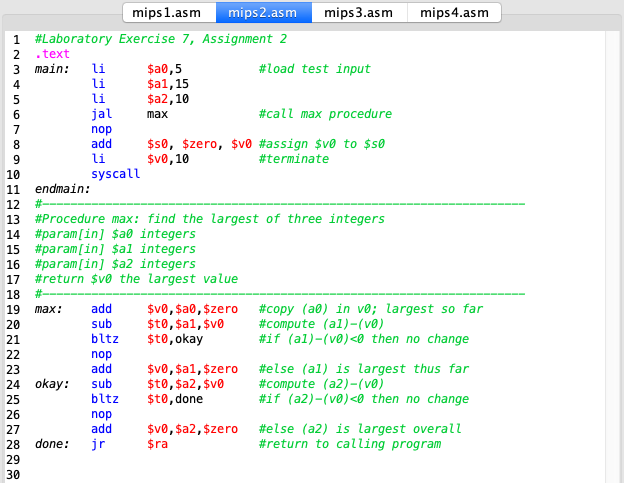
$v0 chứa kết quả.

Đầu tiên so sánh $a0 và $a1, giá trị lớn hơn lưu vào $v0.

So sánh $v0 với $a2, nếu $a2 lớn hơn thì cập nhật $v0.

Lệnh jr để kết thúc chương trình con, quay lại lệnh nop ở chương trình chính.

**Trả lời**:



Thêm lệnh gán $s0 = $v0 (vì sau đó $v0 bị ghi đè) và syscall 10 để kết thúc chương trình.

Đầu vào: 5, 15, 10 lần lượt gán cho $a0, $a1, $a2



Kết quả:



🡪 Kết quả chính

Assignment 3

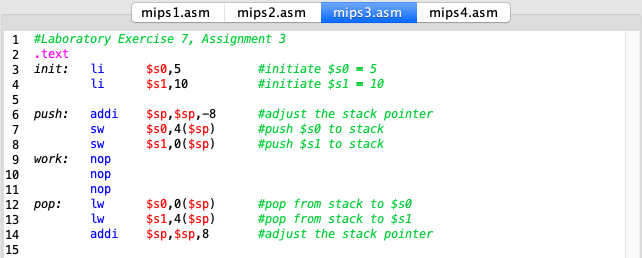
Home Assignment 3. Biên dịch và tải lên trình giả lập. Truyền giá trị kiểm tra cho các thanh ghi $s0 và $s1, quan sát quá trình chạy, chú ý đến con trỏ ngăn xếp. Đi đến k hông gian bộ nhớ được chỉ định bởi thanh ghi $sp để xem chi tiết các thao tác push và pop.

Biển diễn thao tác push và pop trên ngăn xếp, thông qua con trỏ $sp.

Push: thêm $s0 và $s1 vào ngăn xếp thì con trỏ $sp – 8 (chừa ra 2 vị trí trống trên stack)

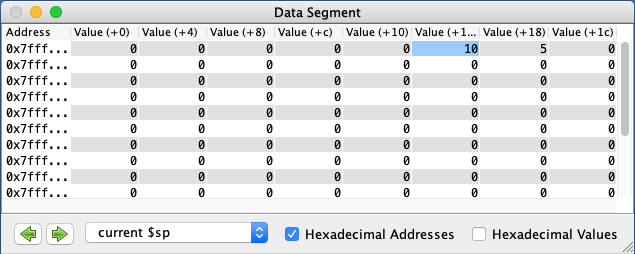
Pop: load word để lấy giá trị ở vị trí $sp lưu vào $s0, vị trí $sp + 4 vào $s1, trả 8 byte cho $sp

**Trả lời**:



Khởi tạo giá trị 5 và 10 cho 2 thanh ghi $s0, $s1



Sau khi push, 5 và 10 sẽ nằm trong vùng nhớ của stack.

Sau khi pop, giá trị đỉnh stack là 10 sẽ được đưa ra thanh ghi $s0, tiếp theo 5 lưu vào $s1



Assignment 4

Home Assignment 4. Biên dịch và tải lên trình giả lập. Truyền giá trị test đầu vào thông qua thanh ghi $a0, chạy chương trình và kiểm tra kết quả trong thanh ghi $v0. Chạy chương trình từng bước một, quan sát sự thay đổi của thanh ghi $pc, $ra, $sp và $fp. Vẽ ngăn xếp duyệt qua chương trình đệ quy này với trường hợp n = 3 (tính 3!).

Sử dụng ngăn xếp:

* Truyền nhiều tham số, lấy nhiều kết quả.
* Khi chương trình con lại gọi chương trình con khác, muốn lưu kết quả và tham số.

$fp là đáy khung thủ tục hiện tại, $sp là đỉnh ngăn xếp.

Trước khi bắt đầu gọi thủ tục, khai báo $fp = $sp để lưu vị trí gốc, sau này $sp chạy xong thì lấy ra giá trị của $fp để quay về vị trí cũ.

Main:

Jal đến WARP để tính n!

Kết quả lưu vào $v0, in ra màn hình dùng syscall 56. Sau đó syscall 10 để kết thúc.

WARP:

Lưu trữ thanh ghi $fp vào stack, khai báo $fp = $sp. Cấp phát vùng nhớ 8 byte.

Gọi đến chương trình FACT để tính giai thừa. n lưu vào $a0.

Khôi phục lại $ra, $sp. Trả lại 4 byte của $fp, quay về chương trình chính.

FACT:

Lệnh đầu tiên: lưu giá trị hiện tại $fp vào ngăn xếp, lưu $sp vào $fp. Cấp phát 12 byte

Nếu đầu vào nhỏ hơn 2, đầu ra = 1, gán $v0 = 1. Nếu ko nhảy đến recursive.

Recursive:

Giảm $a0(n) đi 1 đơn vị. Lặp lại các bước trên. $v1 chứa n, $v0 chứ (n-1)!

Thực hiện phép nhân $v1, $v0, lấy kết quả trong thanh ghi low lưu vào $v0

Done:

Return…

**Trả lời**:

Code:

------------

#Laboratory Exercise 7, Assignment 4

.data

Message: .asciiz "Ket qua tinh giai thua la: "

.text

main: jal WARP

print: add $a1, $v0, $zero # $a0 = result from N!

li $v0, 56

la $a0, Message

syscall

quit: li $v0, 10 #terminate

syscall

endmain:

#---------------------------------------------------------------------

#Procedure WARP: assign value and call FACT

#---------------------------------------------------------------------

WARP: sw $fp,-4($sp) #save frame pointer (1)

addi $fp,$sp,0 #new frame pointer point to the top (2)

addi $sp,$sp,-8 #adjust stack pointer (3)

sw $ra,0($sp) #save return address (4)

li $a0,3 #load test input N

jal FACT #call fact procedure

nop

lw $ra,0($sp) #restore return address (5)

addi $sp,$fp,0 #return stack pointer (6)

lw $fp,-4($sp) #return frame pointer (7)

jr $ra

wrap\_end:

#---------------------------------------------------------------------

#Procedure FACT: compute N!

#param[in] $a0 integer N

#return $v0 the largest value

#---------------------------------------------------------------------

FACT: sw $fp,-4($sp) #save frame pointer

addi $fp,$sp,0 #new frame pointer point to stack’s

top:

addi $sp,$sp,-12 #allocate space for $fp,$ra,$a0 in

stack:

sw $ra,4($sp) #save return address

sw $a0,0($sp) #save $a0 register

slti $t0,$a0,2 #if input argument N < 2

beq $t0,$zero,recursive #if it is false ((a0 = N) >=2)

nop

li $v0,1 #return the result N!=1

j done

nop

recursive:

addi $a0,$a0,-1 #adjust input argument

jal FACT #recursive call

nop

lw $v1,0($sp) #load a0

mult $v1,$v0 #compute the result

mflo $v0

done: lw $ra,4($sp) #restore return address

lw $a0,0($sp) #restore a0

addi $sp,$fp,0 #restore stack pointer

lw $fp,-4($sp) #restore frame pointer

jr $ra #jump to calling

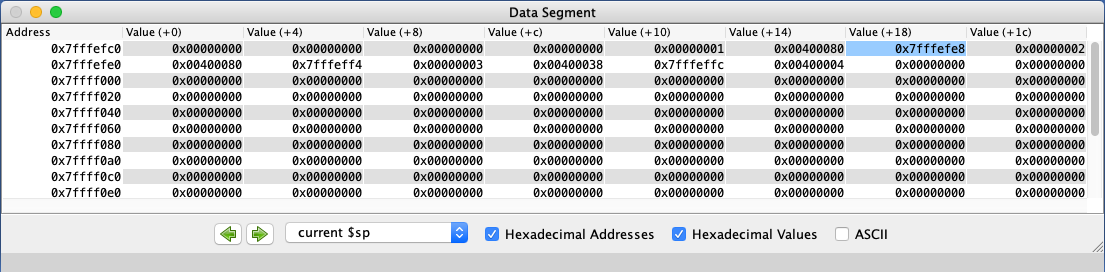
fact\_end:

--------------

Đầu vào n = 3 được lưu ở thanh ghi $a0.

Các giá trị trong vùng nhớ stack:

|  |  |
| --- | --- |
| Stack | Value |
| $a0 | 0x00000001 |
| $ra | 0x00400080 |
| $fp | 0x7fffefe8 |
| $a0 | 0x00000002 |
| $ra | 0x00400080 |
| $fp | 0x7fffeff4 |
| $a0 | 0x00000003 |
| $ra | 0x00400038 |
| $fp | 0x7fffeffc |
| $ra | 0x00400004 |
| Bottom |  |  |



Kết quả:



🡪 Kết quả chính xác

Assignment 5

Viết thủ tục để tìm các số lớn nhất, nhỏ nhất và vị trí của chúng trong danh sách 8 phần tử được lưu trữ trong thanh ghi $s0 đến $s7. Ví dụ:

Lớn nhất: 9,3 -> Phần tử lớn nhất được lưu trữ trong $s3, giá trị lớn nhất là 9

Nhỏ nhất: -3,6 -> Phần tử nhỏ nhất được lưu trữ trong $s6, giá trị nhỏ nhất là -3

**Tips**: sử dụng ngăn xếp để truyền tham số và trả về kết quả.

Đầu ra lưu trữ ở 4 thanh ghi từ $t0 đến $t3.

**Trả lời**:

Code:

---------------

#Laboratory Exercise 7, Assignment 5

.data

comma: .asciiz ", "

newline: .asciiz "\n"

.text

main: li $s0,8

li $s1,7

li $s2,6

li $s3,5

li $s4,4

li $s5,3

li $s6,2

li $s7,1

li $t4,0 #current index in stack

li $t7,7 #last index in list

init\_max: addi $t0,$s0,0 #initiate max value = $s0

li $t1,0 #save max's position

init\_min: addi $t2,$s0,0 #initiate min value = $s0

li $t3,0 #save min's position

push: addi $sp,$sp,-32 #adjust the stack pointer

sw $s0,28($sp) #push $s0 to stack

sw $s1,24($sp) #push $s1 to stack

sw $s2,20($sp) #push $s2 to stack

sw $s3,16($sp) #push $s3 to stack

sw $s4,12($sp) #push $s4 to stack

sw $s5,8($sp) #push $s5 to stack

sw $s6,4($sp) #push $s6 to stack

sw $s7,0($sp) #push $s7 to stack

iter: lw $t5,0($sp) #pop the current top value of stack to $t5

addi $sp,$sp,4 #free the current address

beq $t4,8,print #if index = 8 -> done

beq $t5,$t0,iter #if current top value = max -> next iter

beq $t5,$t2,iter #if current top value = min -> next iter

slt $t6,$t5,$t2 #if current top value < min

bne $t6,$zero,min #then swap min

sgt $t6,$t5,$t0 #if current top value > max

bne $t6,$zero,max #then swap max

max: add $t0,$t5,$zero #set max = $t5

sub $t1,$t7,$t4 #set index of max = 8 - $t4

addi $t4,$t4,1 #increase index

j iter

min: add $t2,$t5,$zero #set Min = $t5

add $t3,$t7,$t4 #set index of Min = 7 - $t4

addi $t4,$t4,1 #increase index

j iter

print: li $v0,1

move $a0,$t0 #print max value

syscall

li $v0, 4 #print comma

la $a0, comma #load address of comma

syscall

li $v0,1

move $a0,$t1 #print max's index

syscall

li $v0, 4 #print newline

la $a0, newline #load address of newline

syscall

li $v0,1

move $a0,$t2 #print min value

syscall

li $v0, 4 #print comma

la $a0, comma #load address of comma

syscall

li $v0,1

move $a0,$t3 #print min's index

syscall

li $v0, 4 #print newline

la $a0, newline #load address of newline

syscall

li $v0,10 #terminate

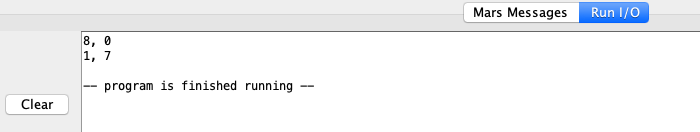
syscall

endmain:

-----------------

Dãy input: 8, 7, 6, 5, 4, 3, 2, 1

Kết quả:



🡪 Chính xác

Conclusions :

1. What registers that the Caller need to save by convention?
2. What registers that the Callee need to save by convention?
3. In push label of Home Assignment 3, could we change the order of adjust stack and store word operations? If yes, what should we have to modify?
4. What is stack pointer?
5. What is frame pointer?

**Trả lời**: Theo em hiểu

1,2,3: Em không rõ ạ

4. Stack pointer ($sp) là con trỏ trỏ đến đỉnh hiện tại của stack

5. Frame pointer ($fp) là con trỏ lưu trữ địa chỉ bắt đầu một thủ tục/1 chương trình con trong stack (tức là vị trí cũ của con trỏ $sp)

🡪 Khi thực hiện xong chương trình con đó thì $sp sẽ đc gán = $fp để quay lại vị trí ban đầu và dễ dàng giải phóng vùng bộ nhớ vừa cấp phát cho ctrinh con đó