

Excercise 7

Võ Anh Khôi – 20225870

Bài 1

#Laboratory Exercise 7 Home Assignment 1

.text

main: li \$a0,-45 #load input parameter

jal abs #jump and link to abs procedure

nop

add \$s0, \$zero, \$v0

li \$v0,10 #terminate

syscall

endmain:

#-----

function abs

param[in] \$a1 the interger need to be gained the absolute

value

return \$v0 absolute value

#-----

abs:

sub \$v0,\$zero,\$a0 #put -(a0) in v0; in case (a0)<0

bltz \$a1,done #if (a0)<0 then done

nop

add \$v0,\$a0,\$zero #else put (a0) in v0

done:

jr \$ra

\$v1	3	0
\$a0	4	-45
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	45

Ban đầu \$a0 được lưu là -45 và kết quả cuối cùng sau khi đã chạy chương trình con abs được lưu vào \$s0 và có giá trị bằng 45

Bài 2

#Laboratory Exercise 7, Home Assignment 2

.text

main: li \$a0,2 #load test input

li \$a1,6

li \$a2,9

jal max #call max procedure

nop

endmain:

#-----

#Procedure max: find the largest of three integers

#param[in] \$a0 integers

#param[in] \$a1 integers

#param[in] \$a2 integers

#return \$v0 the largest value

```

#-----
max: add $v0,$a0,$zero #copy (a0) in v0; largest so far

sub $t0,$a1,$v0 #compute (a1)-(v0)

bltz $t0,okay #if (a1)-(v0)<0 then no change

nop

add $v0,$a1,$zero #else (a1) is largest thus far

okay: sub $t0,$a2,$v0 #compute (a2)-(v0)

bltz $t0,done #if (a2)-(v0)<0 then no change

nop

add $v0,$a2,$zero #else (a2) is largest overall

done: jr $ra #return to calling program

```

kết quả sau khi chạy thử chương trình mẫu

Registers	Coproc 1	Coproc 0	
Name	Number	Value	
\$zero	0	0	
\$at	1	0	
\$v0	2	9	
\$v1	3	0	
\$a0	4	2	
\$a1	5	6	
\$a2	6	9	

Khi chạy lệnh jal thì thanh ghi \$ra được gán bằng giá trị địa chỉ của câu lệnh tiếp theo trong main. Còn thanh ghi \$pc được gán bằng địa chỉ của nhãn Max để thực hiện chương trình con Max. sau khi chạy hết chương trình con đến lệnh jr thì \$pc được gán bằng giá trị trong \$ra để tiếp tục chạy tiếp các câu lệnh trong main

Bài 3

#Laboratory Exercise 7, Home Assignment 3

.text

push: addi \$sp,\$sp,-8 #adjust the stack pointer

```

sw $s0,4($sp) #push $s0 to stack
sw $s1,0($sp) #push $s1 to stack
work: nop
nop
nop
pop: lw $s0,0($sp) #pop from stack to $s0
lw $s1,4($sp) #pop from stack to $s1
addi $sp,$sp,8 #adjust the stack pointer

```

Khi vừa bắt đầu chương trình

\$sp	29	0x7ffffeffc
------	----	-------------

Sau khi chạy lệnh addi \$sp, \$sp, -8

\$sp	29	0x7ffffeff4
------	----	-------------

Thanh ghi \$sp được giảm đi 8 byte để có thể có bộ nhớ để lưu

Lần lượt ghi giá trị của thanh ghi \$s0, \$s1 vào stack

Sau khi chạy chương trình con

Tải lại các giá trị được lưu ở stack lần lượt vào các thanh ghi \$s0, \$s1

Sau khi chạy lệnh addi \$sp, \$sp, 8

\$sp	29	0x7ffffeffc
------	----	-------------

Bài 4

#Laboratory Exercise 7, Home 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 $a0,6 #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:
```

Sau khi thực hiện chương trình thu được kết quả tính giá trị giai thừa là 6 với input N = 3

Khi chạy đệ quy lần đầu tiên

\$sp	29	0x7ffffeff4
\$fp	30	0x7ffffeffc

Khi chạy đệ quy lần thứ 2

\$sp	29	0x7ffffefe8
\$fp	30	0x7ffffeff4

Khi chạy đệ quy lần thứ 3

\$sp	29	0x7ffffefdc
\$fp	30	0x7ffffefe8

Khi chạy để lưu lần cuối cùng trước khi bắt đầu trả lại

\$sp	29	0x7fffffd0
\$fp	30	0x7fffffd0

Về sự thay đổi của thanh ghi \$pc và thanh ghi \$ra thì nó sẽ thay đổi tùy vào việc câu lệnh tiếp theo nó cần phải thực hiện là gì và nó cần được trả lại đâu để có thể tiếp tục chạy chương trình chính:

Ví dụ lần chạy đầu tiên

\$ra	31	0x00400038
pc		0x0040004c

Lần chạy thứ 2,3

\$ra	31	0x00400080
pc		0x0040004c

Sau khi đã chạy xong và N hiện tại =1

\$ra	31	0x00400004
pc		0x00400040

Cuối cùng trả về đầu chương trình để có thể in ra màn hình đáp án

\$ra	31	0x00400004
pc		0x00400004

0x7ffeff8	\$fp 0x00000000
0x7ffeff4	\$ra 0x00400004
0x7ffeff0	\$fp 0x7ffeffc
0x7ffefec	\$ra 0x00400038
0x7ffefe8	\$a0 0x00000003
0x7ffefe4	\$fp 0x7ffeff4
0x7ffefe0	\$ra 0x00400080
0x7ffefdc	\$a0 0x00000002
0x7ffefd8	\$fp 0x7ffefe4
0x7ffefd4	\$ra 0x00400080
0x7ffefd0	\$a0 0x00000001

Bài 5

.data

```
largest: .asciiz "largest: "
smallest: .asciiz "smallest: "
phay: .asciiz ", "
break1: .asciiz "\n"
```

.text

main:

```
li $s0, 23
li $s1, 2
li $s2, 35
li $s3, 34
li $s4, 412
li $s5, 1
li $s6, 14
li $s7, -0
```

```
jal save
# t1 = max
# t2 = indexmax
# t3 = min
```

```
# t4 = index min
```

```
li $v0, 4
```

```
la $a0, largest
```

```
syscall
```

```
li $v0, 1
```

```
add $a0, $0, $t1
```

```
syscall
```

```
li $v0, 4
```

```
la $a0, phay
```

```
syscall
```

```
li $v0, 1
```

```
add $a0, $0, $t2
```

```
syscall
```

```
li $v0, 4
```

```
la $a0, break1
```

```
syscall
```

```
li $v0, 4
```

```
la $a0, smallest
```

```
syscall
```

```
li $v0, 1
```

```
add $a0, $0, $t3
```

```
syscall
```

```
li $v0, 4
```

```
la $a0, phay
```

```
syscall
```

```
li $v0, 1
```

```
add $a0, $0, $t4
```

```
syscall
```

```
li $v0, 10
```

```
syscall
```

```
end_main:
```

```
swap_max:
```

```
addi $t1, $t5, 0
```

```
addi $t2, $t0, 0
```

```
j set_max
```

```
swap_min:
```

```
addi $t3, $t5, 0
```

```
addi $t4, $t0, 0
```

```
j set_min
```

```
save: addi $fp, $sp, 0
```

```
addi $sp, $sp, -32
```

```
sw $s1, 0($sp)
```

```
sw $s2, 4($sp)
```

```
sw $s3, 8($sp)
```

```
sw $s4, 12($sp)
```

```
sw $s5, 16($sp)
```

```
sw $s6, 20($sp)
```

```
sw $s7, 24($sp)
```

```
sw $ra, 28($sp)
```

```
addi $t1, $s0, 0
```

```
li $t2, 0
```

```
addi $t3, $s0, 0
```

```
li $t4, 0
```

find:

```
addi $sp, $sp, 4
```

```
lw $t5, -4($sp)
```

```
sub $a0, $fp, $sp
```

```
beq $a0, $0, done
```

```
addi $t0, $t0, 1
```

```
sub $t8, $t1, $t5
```

```
slt $t8, $0, $t8
```

```
beq $t8, $0, swap_max
```

set_max:

```
sub $t8, $t5, $t3
```

```
slt $t8, $0, $t8
```

```
beq $t8, $0, swap_min
```

set_min:

```
j find
```

done:

```
lw $ra, -4($sp)
```

```
jr $ra
```

