

Thực hành kiến trúc máy tính tuần 7

Hà Trung Chiến - 20225794

Assignment 1:

Edit Execute

mips1.asm

```
1  #Laboratory Exercise 7 Home Assignment 1
2  .text
3  main: li $a0,-45 #load input parameter
4  jal abs #jump and link to abs procedure
5  nop
6  add $s0,$zero,$v0
7  li $v0,10 #terminate
8  syscall
9  endmain:
10 #-----
11 # function abs
12 # param[in] $a1 the interger need to be gained the absolute value
13 # return $v0 absolute value
14 #-----
15 abs:
16 sub $v0,$zero,$a0 #put -(a0) in v0; in case (a0)<0
17 bltz $a0,done #if (a0)<0 then done
18 nop
19 add $v0,$a0,$zero #else put (a0) in v0
20 done:
21 jr $ra
22
```

- Trước khi chạy câu lệnh “jal abs”

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x2404fffd	addiu \$4,\$0,0xfffff...	3: main: li \$a0,-45 #load input parameter
<input type="checkbox"/>	0x00400004	0xc0100006	jal 0x00400018	4: jal abs #jump and link to abs procedure
<input type="checkbox"/>	0x00400008	0x00000000	nop	5: nop
<input type="checkbox"/>	0x0040000c	0x0028001d	add \$16,\$0,\$2	6: add \$a0,\$zero,\$v0
<input type="checkbox"/>	0x00400010	0x2402000a	addiu \$2,\$0,0x0000...	7: li \$v0,10 #terminate
<input type="checkbox"/>	0x00400014	0x0000000c	syscall	8: syscall
<input type="checkbox"/>	0x00400018	0x00410222	sub \$2,\$0,\$4	16: sub \$v0,\$zero,\$a0 #put -(a0) in v0; in case (a0)<0
<input type="checkbox"/>	0x0040001c	0x48000002	bltz \$4,0x00000002	17: bltz \$a0,done #if (a0)<0 then done
<input type="checkbox"/>	0x00400020	0x00000000	nop	18: nop
<input type="checkbox"/>	0x00400024	0x0081021d	add \$2,\$4,\$0	19: add \$v0,\$a0,\$zero #else put (a0) in v0
<input type="checkbox"/>	0x00400028	0x03e00008	jr \$31	21: jr \$ra

Labels

Label	Address
mips1.asm	
main	0x00400000
endmain	0x00400018
abs	0x00400018
done	0x00400028

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0xffffffff
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$a0	16	0x00000000
\$a1	17	0x00000000
\$a2	18	0x00000000
\$a3	19	0x00000000
\$a4	20	0x00000000
\$a5	21	0x00000000
\$a6	22	0x00000000
\$a7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x00008000
\$sp	29	0x7ffffefc
\$fp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x00400004
\$hi		0x00000000
\$lo		0x00000000

Mars Messages Run I/O

- Sau khi chạy câu lệnh “jal abs”

The screenshot shows the Mars MIPS simulator interface. The main window displays assembly code with columns for Address, Code, Basic, and Source. The code includes instructions like `addiu $4,$0,0xffff`, `jal abs`, `nop`, `add $a0,$zero,$v0`, `li $v0,10`, `syscall`, `sub $v0,$zero,$a0`, `bltz $a0,$done`, `add $v0,$a0,$zero`, and `jr $ra`. The right panel shows a memory dump with columns for Name, Number, and Value. The memory dump shows values for registers like `$zero`, `$at`, `$v0`, `$v1`, `$a0`, `$a1`, `$a2`, `$a3`, `$t0`, `$t1`, `$t2`, `$t3`, `$t4`, `$t5`, `$t6`, `$t7`, `$s0`, `$s1`, `$s2`, `$s3`, `$s4`, `$s5`, `$s6`, `$s7`, `$s8`, `$s9`, `$k0`, `$k1`, `$gp`, `$fp`, `$ra`, `$pc`, `$hi`, and `$lo`.

- Kết thúc chương trình

The screenshot shows the Mars MIPS simulator interface. The main window displays assembly code with columns for Address, Code, Basic, and Source. The code includes instructions like `addiu $4,$0,0xffff`, `jal abs`, `nop`, `add $a0,$zero,$v0`, `li $v0,10`, `syscall`, `sub $v0,$zero,$a0`, `bltz $a0,$done`, `add $v0,$a0,$zero`, and `jr $ra`. The right panel shows a memory dump with columns for Name, Number, and Value. The memory dump shows values for registers like `$zero`, `$at`, `$v0`, `$v1`, `$a0`, `$a1`, `$a2`, `$a3`, `$t0`, `$t1`, `$t2`, `$t3`, `$t4`, `$t5`, `$t6`, `$t7`, `$s0`, `$s1`, `$s2`, `$s3`, `$s4`, `$s5`, `$s6`, `$s7`, `$s8`, `$s9`, `$k0`, `$k1`, `$gp`, `$fp`, `$ra`, `$pc`, `$hi`, and `$lo`.

- Nhận xét;

Khi chạy lệnh “jal abs” (địa chỉ 0x00400004) thì thanh ghi \$ra được gán bằng địa chỉ của câu lệnh tiếp theo “nop”(địa chỉ 0x00400008) và thanh ghi pc được gán địa chỉ 0x00400018 (địa chỉ tại nhãn abs)

- Giải thích:

“jal” là câu lệnh sử dụng để thực hiện một nhảy tới một địa chỉ cụ thể trong bộ nhớ, đồng thời lưu trữ địa chỉ trở về (địa chỉ của lệnh ngay sau lệnh jal) vào thanh ghi \$ra (return address). Sau khi hàm được thực thi xong, sử dụng lệnh “jr \$ra” để trở về địa chỉ mà lệnh jal đã lưu trữ.

Assignment 2:

```
mips1.asm
1  #Laboratory Exercise 7, Home Assignment 2
2  .text
3  main: li $a0,2 #load test input
4  li $a1,6
5  li $a2,9
6  jal max #call max procedure
7  nop
8  endmain:
9  #-----
10 #Procedure max: find the largest of three integers
11 #param[in] $a0 integers
12 #param[in] $a1 integers
13 #param[in] $a2 integers
14 #return $v0 the largest value
15 #-----
16 max: add $v0,$a0,$zero #copy (a0) in v0; largest so far
17 sub $t0,$a1,$v0 #compute (a1)-(v0)
18 bltz $t0,okay #if (a1)-(v0)<0 then no change
19 nop
20 add $v0,$a1,$zero #else (a1) is largest thus far
21 okay: sub $t0,$a2,$v0 #compute (a2)-(v0)
22 bltz $t0,done #if (a2)-(v0)<0 then no change
23 nop
24 add $v0,$a2,$zero #else (a2) is largest overall
25 done: jr $ra #return to calling program
```

- Trước khi chạy lệnh “jal max”

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000059
\$a1	5	0x00000074
\$a2	6	0x00000066
\$t0	7	0x00000000
\$t1	8	0x00000000
\$t2	9	0x00000000
\$t3	10	0x00000000
\$t4	11	0x00000000
\$t5	12	0x00000000
\$t6	13	0x00000000
\$t7	14	0x00000000
\$s0	15	0x00000000
\$s1	16	0x00000000
\$s2	17	0x00000000
\$s3	18	0x00000000
\$s4	19	0x00000000
\$s5	20	0x00000000
\$s6	21	0x00000000
\$s7	22	0x00000000
\$s8	23	0x00000000
\$s9	24	0x00000000
\$t8	25	0x00000000
\$t9	26	0x00000000
\$k0	27	0x00000000
\$k1	28	0x10000000
\$gp	29	0x7ffffc00
\$fp	30	0x00000000
\$ra	31	0x00400038

- Sau khi chạy câu lệnh “jal max”

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000059
\$a1	5	0x00000074
\$a2	6	0x00000066
\$t0	7	0x00000000
\$t1	8	0x00000000
\$t2	9	0x00000000
\$t3	10	0x00000000
\$t4	11	0x00000000
\$t5	12	0x00000000
\$t6	13	0x00000000
\$t7	14	0x00000000
\$s0	15	0x00000000
\$s1	16	0x00000000
\$s2	17	0x00000000
\$s3	18	0x00000000
\$s4	19	0x00000000
\$s5	20	0x00000000
\$s6	21	0x00000000
\$s7	22	0x00000000
\$s8	23	0x00000000
\$s9	24	0x00000000
\$t8	25	0x00000000
\$t9	26	0x00000000
\$k0	27	0x00000000
\$k1	28	0x10000000
\$gp	29	0x7ffffc00
\$fp	30	0x00000000
\$ra	31	0x00400014

- Kết thúc chương trình:

- Sau lệnh "lw \$s0, 4(\$sp)"

\$s0	16	0x00000009
\$s1	17	0x00000005

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7fffffe0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000009	0x00000005	0x00000000
0x7fffff00	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Nhận xét:

- Ban đầu thanh ghi \$sp lưu địa chỉ là 0x7fffffc, \$s0 = 5, \$s1 = 9.
- Sau lệnh addi \$sp,\$sp,-8 # cấp phát bộ nhớ . thì giá trị của \$sp là 0x7ffffc.
- Lệnh push tiếp theo lần lượt lưu giá trị của \$s0 và \$s1 vào địa chỉ 0x7ffff4 và 0x7ffff8.
- Ở lệnh pop lần lượt lấy ra 2 giá trị trên lưu vào \$s1 và \$s0.
- Kết quả thu được là \$s0 = 9, \$s1 = 5.

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:

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:

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

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

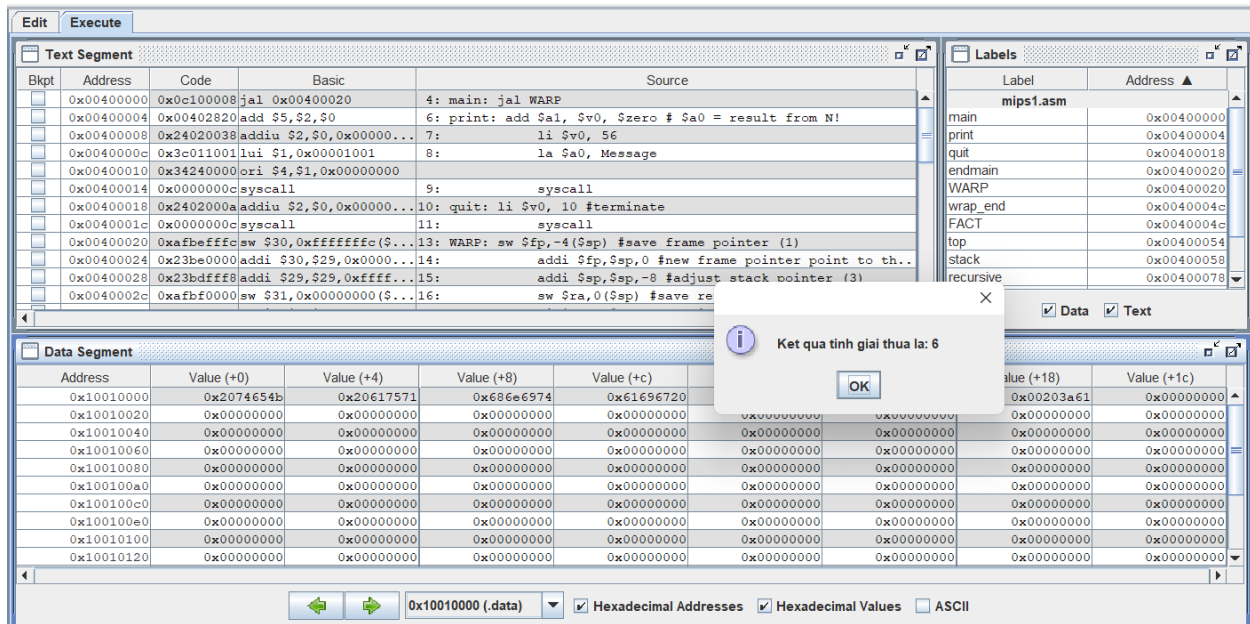
```

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:

```

Kết quả: $3! = 6$

→ Kết quả thu được đúng với lý thuyết

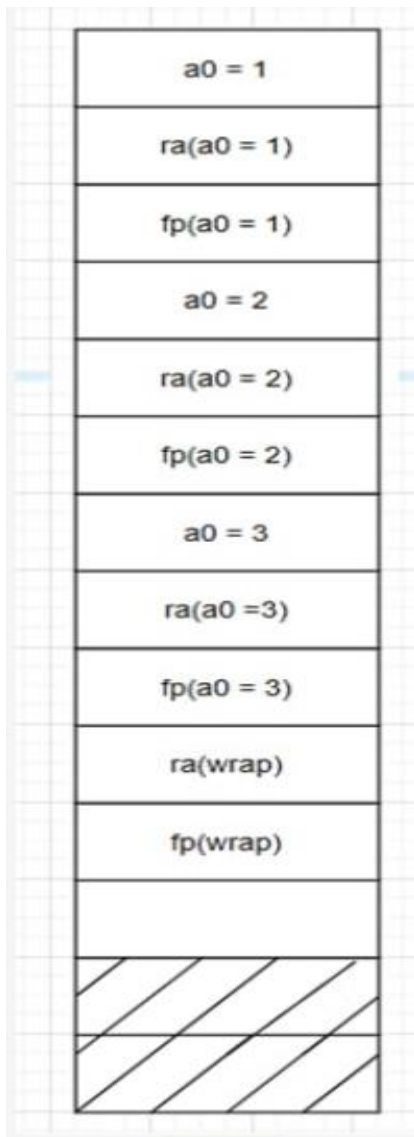


Bảng đệ quy:

\$a0	0x00000001	
\$ra	0x00400080	
\$fp	0x7ffefec	New \$fp
\$a0	0x00000002	
\$ra	0x00400080	
\$fp	0x7ffefe8	New \$fp
\$a0	0x00000003	
\$ra	0x00400038	
\$fp	0x7ffeff4	New \$fp
\$ra	0x00400004	
\$fp	0x7ffeffc	\$sp = \$sp

• Giải thích:

Khi nhảy đến nhãn done ta lần lượt pop ra các giá trị đã lưu ra từ trước để tính toán. Các giá trị thanh ghi \$ra trước đó đã lưu trong stack giúp ta return về vị trí lệnh jal qua đó kết thúc từng thủ tục một và quay về với chương trình chính (main). Các giá trị của thanh ghi \$fp đưa ta đến từng khung trang, nơi mà mỗi vị trí tương ứng sẽ có 3 tham số n, \$ra, \$fp



Assignment 5:

#code:

.data

Mess1 : .asciiz "Largest : "

Mess2 : .asciiz "Smallest : "

Mess3: .asciiz ","

newline: .asciiz "\n"

.text

Main:

li \$s0, 2

li \$s1, 8

li \$s2, 0

li \$s3, 4

li \$s4, -10


```

li $s5, 7
li $s6, -3
li $s7, 6
add $t8, $s0, $0 # Max value
add $t9, $s0, $0 # Min value
addi $t0,$0,1 # i = 1
addi $t1,$0,1 # j = 1
add $t6,$0,$0 # addr Max
add $t7,$0,$0 # addr Min
la $fp,0($sp) # addr $sp

```

Push_Stack:

```

sw $s0,0($sp)
sw $s1,4($sp)
sw $s2,8($sp)
sw $s3,12($sp)
sw $s4,16($sp)
sw $s5,20($sp)
sw $s6,24($sp)
sw $s7,28($sp)
jal Check_Max
nop
move $sp,$fp
jal Check_Min
nop
j Print
nop

```

Check_Max:

```

slti $t2, $t0,8 # $t2 = 1 if i < 8
beq $t2,$0,Done
nop
addi $sp,$sp,4
lw $t3,0($sp)
slt $t4,$t8,$t3 # $t4 = 1 if $t8 < $t3
bne $t4,$0,Update_Max
nop
addi $t0,$t0,1
j Check_Max

```

Update_Max:

```

move $t8,$t3
add $t6,$t0,$0
addi $t0,$t0,1 # i += 1
j Check_Max

```

Check_Min:

```

    slti $t2, $t1,8 # $t2 = 1 if j < 8
    beq $t2,$0,Done
    nop
    addi $sp,$sp,4
    lw $t3,0($sp)
    slt $t4,$t3,$t9 # $t4 = 1 if min > $t3
    bne $t4,$0,Update_Min
    nop
    addi $t1,$t1,1
    j Check_Min

```

Update_Min:

```

    move $t9,$t3
    add $t7,$t1,$0
    addi $t1,$t1,1 # j += 1
    j Check_Min

```

Done:

```

    jr $ra

```

Print:

```

    li $v0,4
    la $a0,Mess1
    syscall
    li $v0,1
    move $a0,$t8
    syscall
    li $v0,4
    la $a0,Mess3
    syscall
    li $v0,1
    move $a0,$t6
    syscall
    li $v0,4
    la $a0,newline
    syscall
    li $v0,4
    la $a0,Mess2
    syscall
    li $v0,1
    move $a0,$t9
    syscall
    li $v0,4
    la $a0,Mess3
    syscall
    li $v0,1

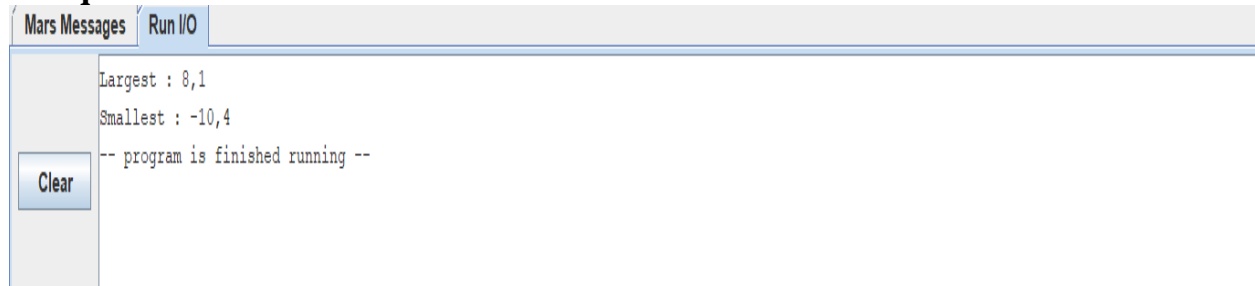
```

```
move $a0,$t7  
syscall
```

Exit:

```
li $v0,10  
syscall
```

Kết quả:



The screenshot shows the Mars MIPS simulator's output window. It has two tabs: 'Mars Messages' and 'Run I/O'. The 'Run I/O' tab is active, displaying the following text: 'Largest : 8,1', 'Smallest : -10,4', and '-- program is finished running --'. On the left side of the window, there is a 'Clear' button.

→ Nhận xét: Kết quả thu được đúng với lí thuyết