

# Thực hành Kiến trúc máy tính tuần 7

Họ tên: Đỗ Hoàng Minh Hiếu

MSSV: 20225837

## Bài 1

Code:

```
.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] $a0 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 $a0,done #if (a0)<0 then done
      nop
      add $v0,$a0,$zero #else put (a0) in v0
done:
      jr $ra
```

Trước khi thực thi lệnh jal abs

The screenshot shows the Mars MIPS simulator interface. The Text Segment window displays the assembly code, with the jal instruction highlighted. The Data Segment window shows memory addresses from 0x10010000 to 0x10010140, all containing 0x00000000. The Labels window shows the current program counter at 0x00400000. The Register window shows \$zero at 0, \$at at 1, \$v0 at 2, \$v1 at 3, \$a0 at 4 (containing 0xffffffff), \$a1 at 5, \$a2 at 6, \$t0 at 7, \$t1 at 8, \$t2 at 9, \$t3 at 10, \$t4 at 11, \$t5 at 12, \$t6 at 13, \$t7 at 14, \$a0 at 15, \$a1 at 16, \$a2 at 17, \$a3 at 18, \$a4 at 19, \$a5 at 20, \$a6 at 21, \$a7 at 22, \$a8 at 23, \$a9 at 24, \$a10 at 25, \$a11 at 26, \$a12 at 27, \$a13 at 28, \$a14 at 29, \$a15 at 30, \$ra at 31 (containing 0x00000000), \$pc at 0x00400004, \$hi at 0x00000000, and \$lo at 0x00000000.

## Sau khi thực thi lệnh jal abs

The screenshot shows a debugger interface with three main panels:

- Assembly Window:** Displays assembly code with addresses, codes, basic blocks, and source code. The instruction at address 0x00400018 is `jal abs`, which jumps to the `abs` procedure.
- Register Window:** Shows the state of registers. `$a0` is 0x00400018 and `$s0` is 0x00400008.
- Memory Dump:** Shows a hex dump of memory starting at address 0x00400000. The dump shows the state of memory after the `jal abs` instruction.

\* Ta thấy rằng, sau khi chạy lệnh, thanh ghi \$pc chuyển đến lệnh có nhãn tương ứng (0x00400018) và thanh ghi \$ra lưu giữ địa chỉ kế tiếp lệnh jal (0x00400008).

\* Với \$a0 = -45 thì kết quả là 45

\$a0	4	-45
\$s0	16	45

\* Với \$a0 = 36 thì kết quả là 36

\$a0	4	36
\$s0	16	36

=> Kết quả đúng.

## Bài 2

Code:

```

.text
main: li $a0,2 #Load test input
      li $a1,6
      li $a2,9
      jal max #call max procedure
      nop
      li $v0, 10 #terminate
      syscall
endmain:
#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

```

\*Với \$a0=2, \$a1=6, \$a2 = 9 thì kết quả là

\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	9
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$m	28	268468224

\*Với Với \$a0=-2, \$a1=4, \$a2 = 12 thì kết quả là

\$a0	4	-2
\$a1	5	4
\$a2	6	12
\$a3	7	0
\$t0	8	8
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	12

=> đúng với lý thuyết

\*Sự thay đổi thanh ghi:

Sau lệnh jal max, thanh ghi \$ra lưu giá trị là địa chỉ của lệnh ngay sau nó: (0x00400010), thanh ghi \$pc nhảy đến địa chỉ lệnh max.

Ngay sau khi chương trình con kết thúc, thanh ghi pc trở lại giá trị 0x00400010 (địa chỉ của lệnh nop).

### Bài 3

Code:

```
.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 thực hiện lệnh addi \$sp,\$sp,-8, thanh ghi \$sp sẽ cấp phát 4 byte để lưu trữ dữ liệu nên sẽ là 0x7ffeff4. Sau đó lần lượt ghi giá trị trong \$s0 vào \$sp + 4, giá trị trong \$s1 vào \$sp+0.

\$gp	28	0x10008000
\$sp	29	0x7ffff4
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400004

Khi thực hiện lệnh `addi $sp,$sp,8`, thanh ghi \$sp sẽ trả về giá trị cũ. Thực hiện đổi chỗ hai số bằng cách load giá trị tại địa chỉ `$sp + 0` vào \$s1, load giá trị tại địa chỉ `$sp + 4` vào \$s1.

\$gp	28	0x10008000
\$sp	29	0x7ffffc
\$fp	30	0x00000000

## Bài 4

Code:

.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:

```

Với  $a0 = 6$  thì kết quả chương trình:  $6! = 720$ , kết quả thực thi đúng với lý thuyết.

Thanh ghi \$pc trở về địa chỉ tương ứng trong sau khi thực hiện lệnh jal. Sau khi thực hiện lệnh jal, thanh ghi \$ra sẽ lưu địa chỉ kế tiếp lệnh jal, nếu đã có địa chỉ lưu vào \$ra thì sẽ lưu vào thanh ghi \$sp. Thanh ghi \$fp sẽ lưu lại cấp phát của con trỏ \$sp và lưu lại.

Sự thay đổi của thanh ghi \$sp với n=3:

0x7ffff8	\$fp = 0x00000000
0x7ffff4	\$ra = 0x00400004
0x7ffff0	\$fp = 0x7ffffc
0x7fffec	\$ra = 0x00400038
0x7fffe8	\$a0 = 0x00000003
0x7fffe4	\$fp = 0x7ffff4
0x7fffe0	\$ra = 0x00400080
0x7fffdc	\$a0 = 0x00000002
0x7fffd8	\$fp = 0x7fffe4
0x7fffd4	\$ra = 0x00400080
0x7fffd0	\$a0 = 0x00000001

## Bài 5

Code:

.data

largest: .asciiz "Largest: "

smallest: .asciiz "\nSmallest: "

comma: .asciiz ", "

.text

main:

li \$s0, -11

li \$s1, -2

li \$s2, 4

li \$s3, -9

li \$s4, 12



```

li $s5, 3
li $s6, 2
li $s7, 71
add $t9,$sp,$zero # Save address $sp+0
addi $sp, $sp, -28
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)
add $t0,$s0,$zero # Max = $s0
add $t1,$s0,$zero # Min = $s0
li $t5, 0 # position of Max to 0
li $t6, 0 # position of Min to 0
li $t2, 0 # i = 0

```

find:

```

sub $t7, $t9, $sp
beq $t7, $zero, end_find
lw $t3, 0($sp)
addi $sp, $sp, 4
addi $t2, $t2, 1
slt $t4, $t3, $t0
beq $t4, $zero, swapMax
slt $t4, $t3, $t1
bne $t4, $zero, swapMin

```

continue:

j find

swapMax:

add \$t0,\$t3,\$zero

add \$t5,\$t2,\$zero

j continue

swapMin:

add \$t1,\$t3,\$zero

add \$t6,\$t2,\$zero

j continue

end\_find:

li \$v0, 4 # Print message Largest

la \$a0, largest

syscall

add \$a0, \$t0, \$zero # Print Max

li \$v0, 1

syscall

li \$v0, 4 # Print message Comma

la \$a0, comma

syscall

add \$a0, \$t5, \$zero

li \$v0, 1 # Print the register number of Max

syscall

```
li $v0, 4 # Print message Smallest  
la $a0, smallest  
syscall
```

```
add $a0, $t1, $zero # Print Min  
li $v0, 1  
syscall
```

```
li $v0, 4 # Print message Comma  
la $a0, comma  
syscall
```

```
add $a0, $t6, $zero  
li $v0, 1 # Print the register number of Max  
syscall
```

```
add $a0, $t6, $zero  
li $v0, 10  
syscall
```

\*Với  $\$s0 = -1$ ,  $\$s1 = -2$ ,  $\$s2 = 4$ ,  $s3 = -9$ ,  $\$s4 = 12$ ,  $\$s5 = 3$ ,  $\$s6 = 2$ ,  $\$s7 = 71$

Kết quả sau khi thực thi chương trình là

```
Largest: 71, 7  
Smallest: -9, 3  
-- program is finished running --
```

=> Đúng với lý thuyết.