

THỰC HÀNH KIẾN TRÚC MÁY TÍNH

IT-3280

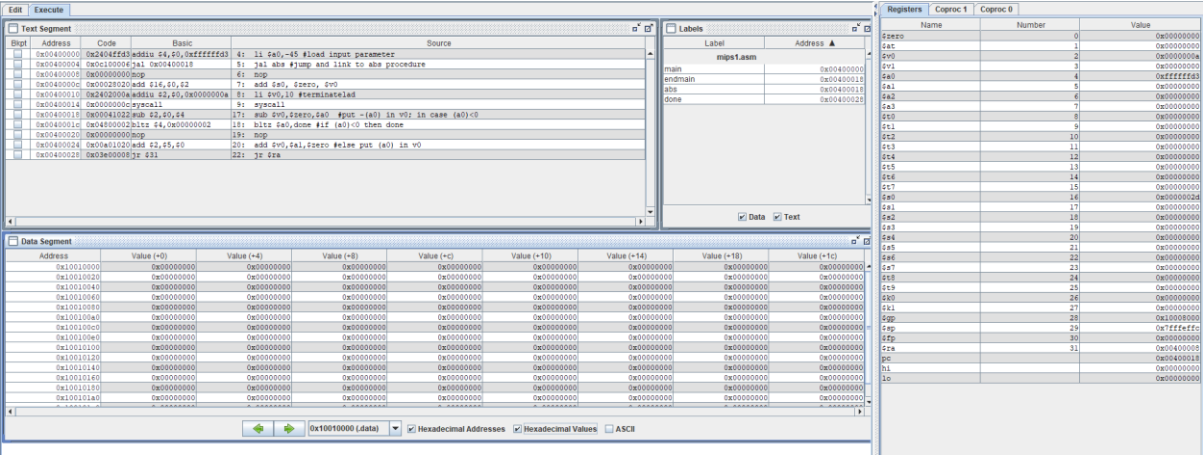
Họ tên	Thân Cát Ngọc Lan
MSSV	20225646

Assignment 1:

*Code

```
#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 #terminatelad
    syscall
endmain:
#-----
# function abs
# param[in] $a1 the interger need to be gained the absolute
# 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 $v0,$a1,$zero #else put (a0) in v0
done:
    jr $ra
```

*Kết quả



*Giải thích:

- Quan sát sự thay đổi của thanh ghi \$ra và thanh ghi \$pc:
- + Ngay trước khi thực hiện chương trình con (lệnh jal abs):

\$ra	31	0x00000000
pc		0x00400004

+ Sau lệnh jal abs (bắt đầu chương trình con), thanh ghi \$ra lưu giá trị là địa chỉ của lệnh ngay sau nó: nop (có địa chỉ 0x00400008). Thanh ghi pc hiện tại có giá trị 0x00400018 (là địa chỉ của lệnh sub \$v0,\$zero,\$a0).

\$ra	31	0x00400008
pc		0x00400018

Assignment 2:

*Code

```
#Laboratory Exercise 7, Home Assignment 2
.data
msg: .ascii " so lon nhat la:"
.text
main: li $a0,2 #load test input
      li $a1,6
      li $a2,9
      jal max #call max procedure
      nop

      add $s0, $zero, $v0
      li $v0, 4
      la $a0, msg
      syscall

      add $a0, $zero, $s0
      li $v0, 1
      syscall

      li $v0, 10
      syscall
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:

The screenshot shows a debugger interface with three main panels:

- Text Segment:** Displays assembly instructions with their addresses and sources. Key instructions include:
 - 0x00400000: `li $a1, 4`
 - 0x00400004: `jal max`
 - 0x00400008: `nop`
- Labels:** Shows a list of labels with their addresses, including `main` at 0x00400000 and `endmain` at 0x0040000c.
- Registers:** A table of registers. The `$ra` register is highlighted with a value of 31. The `$pc` register is highlighted with a value of 0x0040000c.

*Giải thích:

- Quan sát sự thay đổi của thanh ghi \$ra và thanh ghi \$pc:
- + Ngay trước khi thực hiện chương trình con (lệnh jal max):

\$ra	31	0x00000000
pc		0x0040000c

+ Sau lệnh jal max (bắt đầu chương trình con), thanh ghi \$ra lưu giá trị là địa chỉ của lệnh ngay sau nó: nop (có địa chỉ 0x00400010). Thanh ghi pc hiện tại có giá trị 0x0040003c (là địa chỉ của lệnh add \$v0,\$a0,\$zero).

\$ra	31	0x00400010
pc		0x0040003c

Assignment 3:

*Code

```

1  #Laboratory Exercise 7, Home Assignment 3
2  li $s1, 2023
3  li $s0, 2024
4  .text
5  push: addi $sp,$sp,-8 #Tạo 2 vị trí cho stack
6  sw $s0,4($sp) # push $s0 vào stack
7  sw $s1,0($sp) # push $s1 vào stack
8  work: nop
9  nop
10 nop
11 pop: lw $s0,0($sp) # pop from stack to $s0 -> Cho thanh ghi $s0 lấy giá trị tại địa chỉ thanh ghi $s1
12 lw $s1,4($sp) # pop from stack to $s1 -> Cho thanh ghi $s1 lấy giá trị tại địa chỉ thanh ghi $s0
13 addi $sp,$sp,8 #Trả lại vị trí cho stack
14

```

*Kết quả

Khởi tạo \$s0 = 2024, \$s1 = 2023

Dispt	Address	Code	Basic	Source
00000000	00000000	addi \$t0,\$t0,2023	21: li \$t0, 2023	
00000004	00000004	addi \$t1,\$t1,2024	31: li \$t1, 2024	
00000008	00000008	addi \$sp,\$sp,-8	51: pushi \$sp,\$sp,-8 #Tho 2 v7 tr1 cho stack	
0000000C	0000000C	sw \$t0,4(\$sp)	61: sw \$t0,4(\$sp) # push \$t0 vào stack	
00000010	00000010	sw \$t1,8(\$sp)	71: sw \$t1,8(\$sp) # push \$t1 vào stack	
00000014	00000014	nop	81: worki nop	
00000018	00000018	sw \$t0,4(\$sp)	91: sw \$t0,4(\$sp)	
0000001C	0000001C	nop	101: nop	
00000020	00000020	sw \$t1,8(\$sp)	111: sw \$t1,8(\$sp) # push \$t1 vào stack	
00000024	00000024	sw \$t0,4(\$sp)	121: sw \$t0,4(\$sp) # push \$t0 vào stack	
00000028	00000028	addi \$sp,\$sp,8	131: addi \$sp,\$sp,8 #Tho 2 v7 tr1 cho stack	

Label	Address
mpst1.asm	00000000
push	0000000C
work	00000014
pop	00000028

Name	Number	Value
\$sp	0	0
\$t0	1	0
\$t1	2	0
\$t2	3	0
\$t3	4	0
\$t4	5	0
\$t5	6	0
\$t6	7	0
\$t7	8	0
\$t8	9	0
\$t9	10	0
\$t10	11	0
\$t11	12	0
\$t12	13	0
\$t13	14	0
\$t14	15	0
\$t15	16	2023
\$t16	17	2024
\$t17	18	0
\$t18	19	0
\$t19	20	0
\$t20	21	0
\$t21	22	0
\$t22	23	0
\$t23	24	0
\$t24	25	0
\$t25	26	0
\$t26	27	0
\$t27	28	2023
\$t28	29	2024
\$t29	30	0
\$t30	31	0
\$t31	32	4194348
\$t32	33	0

⇒ Sau khi chạy chương trình: \$s0 = 2023, \$s1 = 2024. Kết quả chạy đúng.

*Giải thích:

- Sau khi gặp lệnh addi \$sp,\$sp,-8. Địa chỉ thanh ghi \$sp giảm 8. Có thể hiểu stack dành ra 2 vị trí để lưu các giá trị. Khi nạp giá trị vào lúc này địa chỉ thanh ghi \$sp không thay đổi.
- Giải thích chi tiết trong code.

Assignment 4:

*Code:

```
#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 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:

```

*Kết quả:

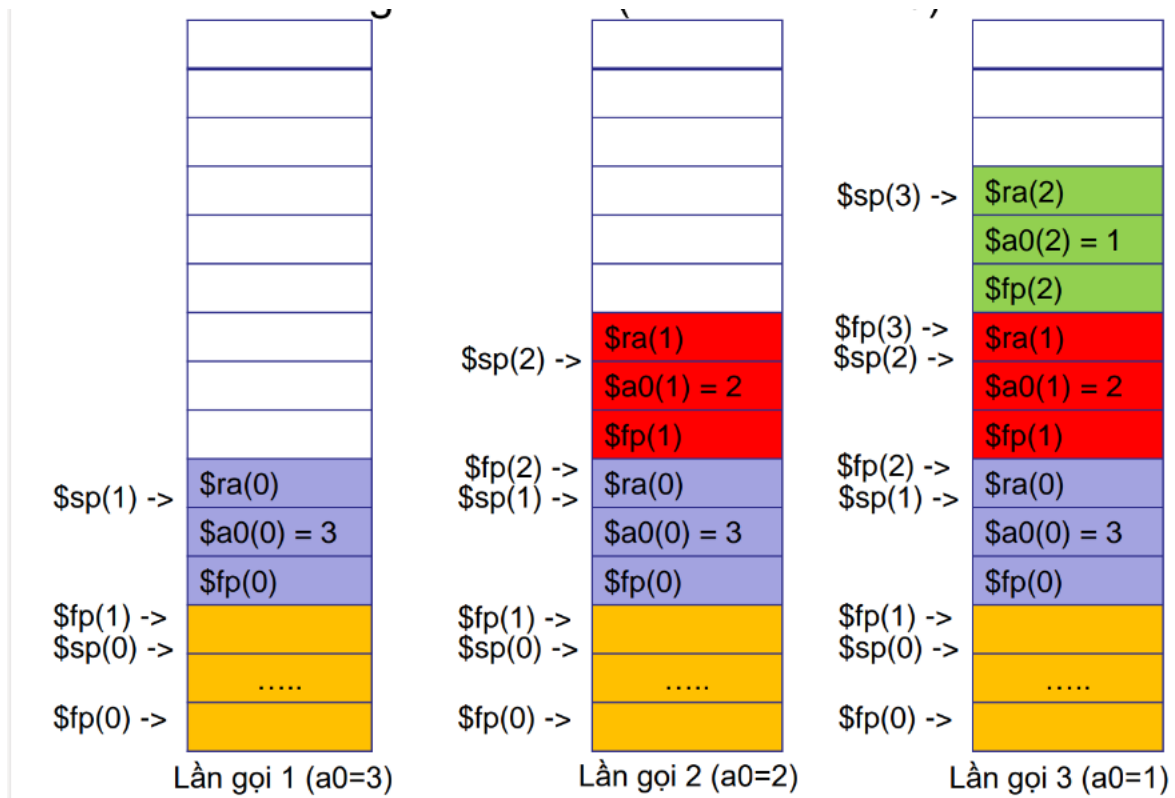
Khởi tạo n=3;

The screenshot shows a debugger window with the following components:

- Text Segment:** Displays assembly code with addresses, codes, and comments. The code implements a recursive function 'FACT' to calculate the factorial of 3. Key instructions include saving the frame pointer, adjusting the stack pointer, recursive calls, and restoring registers.
- Labels:** A list of labels with their corresponding addresses, including 'main', 'fact', 'recursive', and 'done'.
- Data Segment:** A table showing memory addresses and their values in hexadecimal. The values are mostly zeros, indicating uninitialized memory.
- Dialog Box:** A small window titled 'Kết quả tính giai thừa là: 6' (The result of the factorial calculation is: 6) with an 'OK' button.

*Giải thích:

Vẽ ngăn xếp đệ quy:



Assignment 5:

*Code:

```
.data
msg1: .ascii " largest:"
msg2: .ascii "\n smallest:"
.text
main:
# Khởi tạo giá trị từ thanh ghi $a0 đến thanh ghi $a7
li $a0, 2
li $a1, 3
li $a2, 5
li $a3, -3
li $a4, 26
li $a5, 8
li $a6, 1
li $a7, -8

jal produce
nop

li $v0, 4
la $a0, msg1
syscall
add $a0, $t0, $zero
li $v0, 1
syscall
li $v0, 11
li $a0, ','
syscall
add $a0, $a1, $zero
li $v0, 1
syscall
li $v0, 4
la $a0, msg2
syscall
add $a0, $t1, $zero
li $v0, 1
syscall
li $v0, 11
li $a0, ','
syscall
add $a0, $a2, $zero
li $v0, 1
syscall
li $v0, 10
syscall
endmain:
```

```

45 endmain:
46
47 swapMax: add $t0,$t3,$zero
48 add $a1,$t2,$zero
49 jr $ra
50
51 swapMin: add $t1,$t3,$zero
52 add $a2,$t2,$zero
53 jr $ra
54
55 produce: #Tìm giá trị nhỏ nhất và lớn nhất
56 add $a3,$sp,$zero # Lưu địa chỉ bắt đầu của thanh ghi $sp
57 addi $sp, $sp, -32 # integer5 in stack
58 sw $s1, 0($sp)
59 sw $s2, 4($sp)
60 sw $s3, 8($sp)
61 sw $s4, 12($sp)
62 sw $s5, 16($sp)
63 sw $s6, 20($sp)
64 sw $s7, 24($sp)
65 sw $ra, 28($sp)
66 add $t0,$s0,$zero # Max = $s0
67 add $t1,$s0,$zero # Min = $s0
68 li $a1, 0 # Index of Max
69 li $a2, 0 # Index of Min
70 li $t2, 0 # i = 0
71 loop:
72 addi $sp, $sp, 4
73 lw $t3, -4($sp)
74 sub $t6, $sp, $a3
75 beq $t6,$zero, done # Nếu $sp = $fp nhảy tới 'done'
76 nop
77 addi $t2,$t2,1 # i++
78 sub $t6,$t0,$t3
79 bltzal $t6, swapMax # If $t3 > Max branch to the swapMax
80 nop
81 sub $t6,$t3,$t1
82 bltzal $t6, swapMin # If $t3 < Min branch to the swapMin
83 nop
84 j loop
85 done:
86 lw $ra, -4($sp)
87 jr $ra # Trở lại

```

***Kết quả:**

The screenshot displays a MIPS simulator interface with several panels:

- Text Segment:** Shows assembly code with addresses, instructions, and comments. Key instructions include `add $a3,$sp,$zero`, `addi $sp, $sp, -32`, and various `sw` (store word) and `lw` (load word) operations.
- Registers:** A table showing the state of MIPS registers. Notable values include `$s0` at 0x00000000, `$s1` at 0x00000000, and `$t0` at 0x00000000.
- Memory:** A table showing memory addresses and their corresponding values. The memory is mostly zero, with some non-zero values at higher addresses.
- Execution Results:** A section at the bottom showing the output of the program. It indicates that the program is finished running and displays the largest and smallest values found: `largest:26,4` and `smallest:-5,7`.

⇒ **Kết quả chạy đúng.**