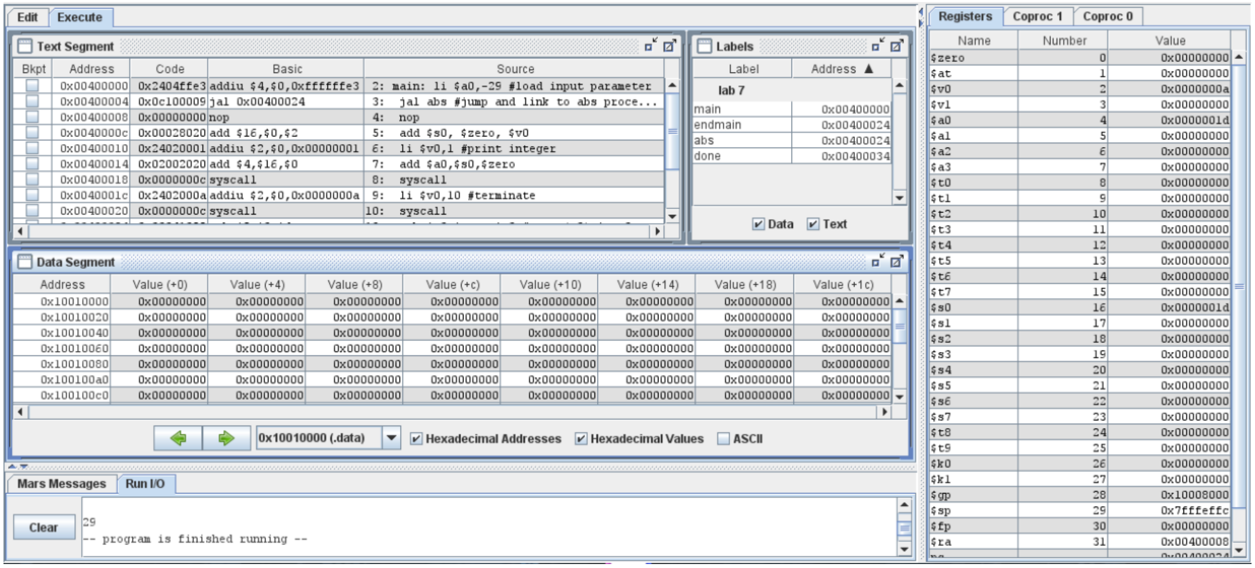
Phạm Duy Hưng – 20225850

Assignment 1:

```
.text
main: li $a0,-29 #load input parameter
 jal abs #jump and link to abs procedure
 nop
 add $s0, $zero, $v0
 li $v0,1 #print integer
 add $a0,$s0,$zero
 syscall
 li $v0,10 #terminate
 syscall
endmain:
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
```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x2404ffe3	addiu $4,$0,0xffffffe3	2: main: li $a0,-29 #load input parameter
□	0x00400004	0x0c100009	jal 0x00400024	3: jal abs #jump and link to abs proce...
□	0x00400008	0x00000000	nop	4: nop
□	0x0040000c	0x00028020	add $16,$0,$2	5: add $s0, $zero, $v0
□	0x00400010	0x24020001	addiu $2,$0,0x00000001	6: li $v0,1 #print integer
□	0x00400014	0x02002020	add $4,$16,$0	7: add $a0,$s0,$zero
□	0x00400018	0x0000000c	syscall	8: syscall
□	0x0040001c	0x2402000a	addiu $2,$0,0x0000000a	9: li $v0,10 #terminate
□	0x00400020	0x0000000c	syscall	10: syscall

Labels

Label	Address ▲
lab 7	
main	0x00400000
endmain	0x00400024
abs	0x00400024
done	0x00400034

☑ Data ☑ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Mars Messages | Run I/O

Clear

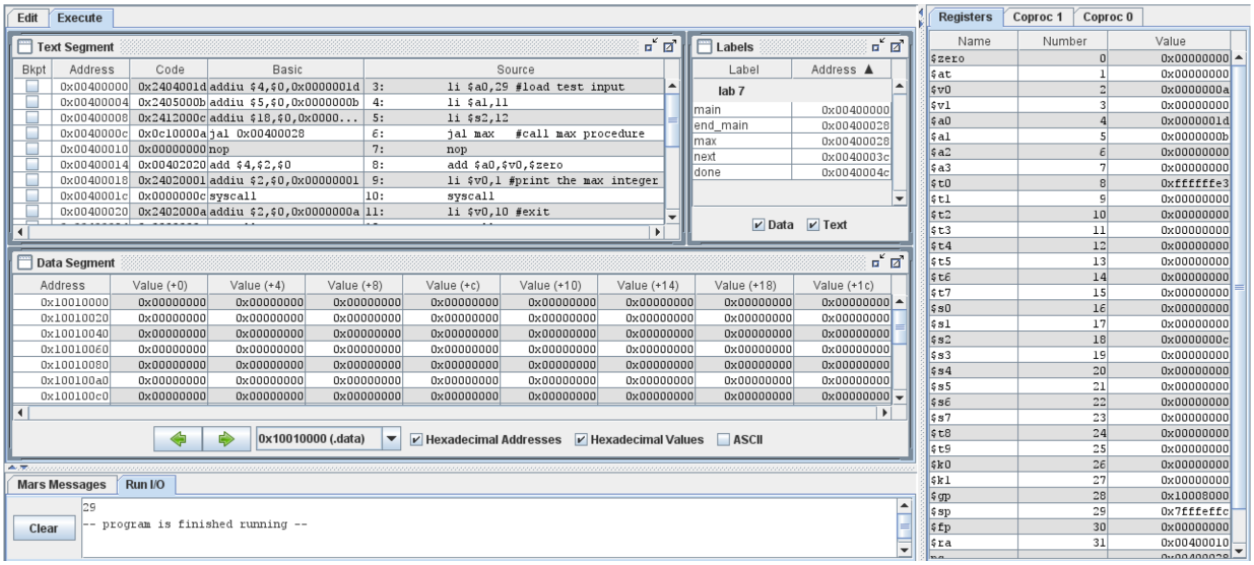
```
29
-- program is finished running --
```

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x0000001d
$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
$s0	16	0x0000001d
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400008

⇨ Kết quả chính xác

Assignment 2:

```
.text
main:
    li $a0,29 #load test input
    li $a1,11
    li $s2,12
    jal max    #call max procedure
    nop
    add $a0,$v0,$zero
    li $v0,1 #print the max integer
    syscall
    li $v0,10 #exit
    syscall
end_main:
max:
    add $v0,$a0,$zero #copy (a0) in v0; largest so far
    sub $t0,$a1,$v0   #compute (a1)-(v0)
    bltz $t0,next     #if (a1)-(v0)<0 then no change
    nop
    add $v0,$a1,$zero #else (a1) is largest thus far
 next:
    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
```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x2404001d	addiu $4,$0,0x0000001d	3: li $a0,29 #load test input
□	0x00400004	0x2405000b	addiu $5,$0,0x0000000b	4: li $a1,11
□	0x00400008	0x2412000c	addiu $18,$0,0x0000...	5: li $s2,12
□	0x0040000c	0x0c10000a	jal 0x00400028	6: jal max #call max procedure
□	0x00400010	0x00000000	nop	7: nop
□	0x00400014	0x00402020	add $4,$2,$0	8: add $a0,$v0,$zero
□	0x00400018	0x24020001	addiu $2,$0,0x00000001	9: li $v0,1 #print the max integer
□	0x0040001c	0x0000000c	syscall	10: syscall
□	0x00400020	0x2402000a	addiu $2,$0,0x0000000a	11: li $v0,10 #exit

Labels

Label	Address ▲
lab 7	
main	0x00400000
end_main	0x00400028
max	0x00400028
next	0x0040003c
done	0x0040004c

☑ Data ☑ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

```
29
-- program is finished running --
```

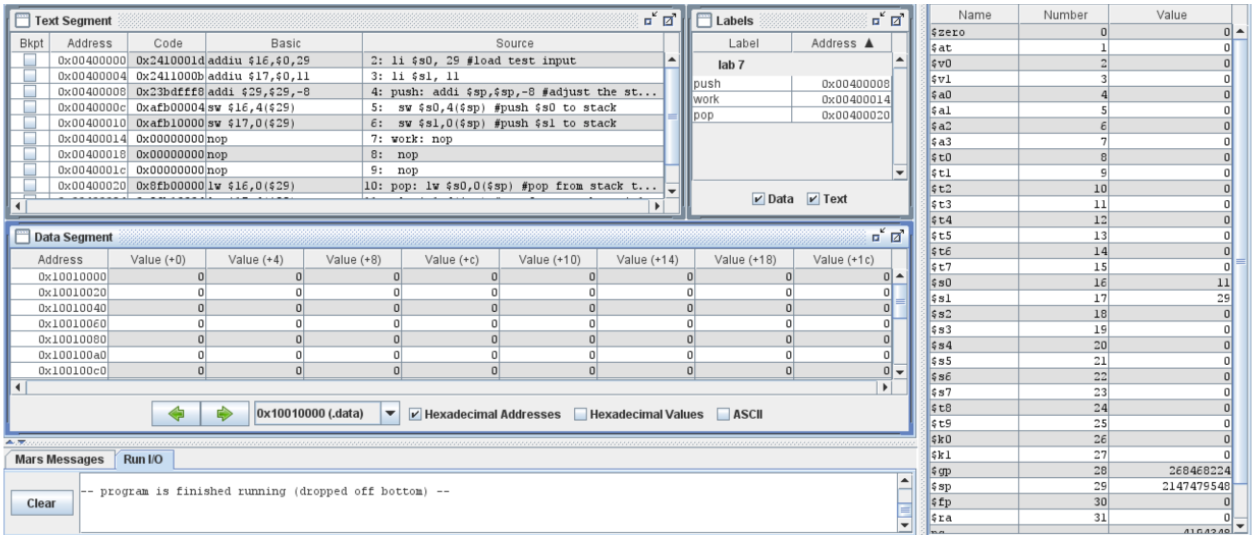
Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x0000001d
$a1	5	0x0000000b
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0xffffffe3
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x0000000c
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400010

-Số lớn nhất là 29 => Kết quả đúng.

Assignment 3:

```
.text
li $s0, 29 #load test input
li $s1, 11
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
```

-Ban đầu s0 = 10, s1 = 0 sau khi chạy chương trình => s0 = 11, s1 = 10.



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x2410001d	addiu $16,$0,29	2: li $s0, 29 #load test input
□	0x00400004	0x2411000b	addiu $17,$0,11	3: li $s1, 11
□	0x00400008	0x23bdfff8	addi $29,$29,-8	4: push: addi $sp,$sp,-8 #adjust the st...
□	0x0040000c	0xafb00004	sw $16,4($29)	5: sw $s0,4($sp) #push $s0 to stack
□	0x00400010	0xafb10000	sw $17,0($29)	6: sw $s1,0($sp) #push $s1 to stack
□	0x00400014	0x00000000	nop	7: work: nop
□	0x00400018	0x00000000	nop	8: nop
□	0x0040001c	0x00000000	nop	9: nop
□	0x00400020	0x8fb00000	lw $16,0($29)	10: pop: lw $s0,0($sp) #pop from stack t...

Labels

Label	Address ▲
lab 7	
push	0x00400008
work	0x00400014
pop	0x00400020

☑ Data ☑ Text

Data Segment

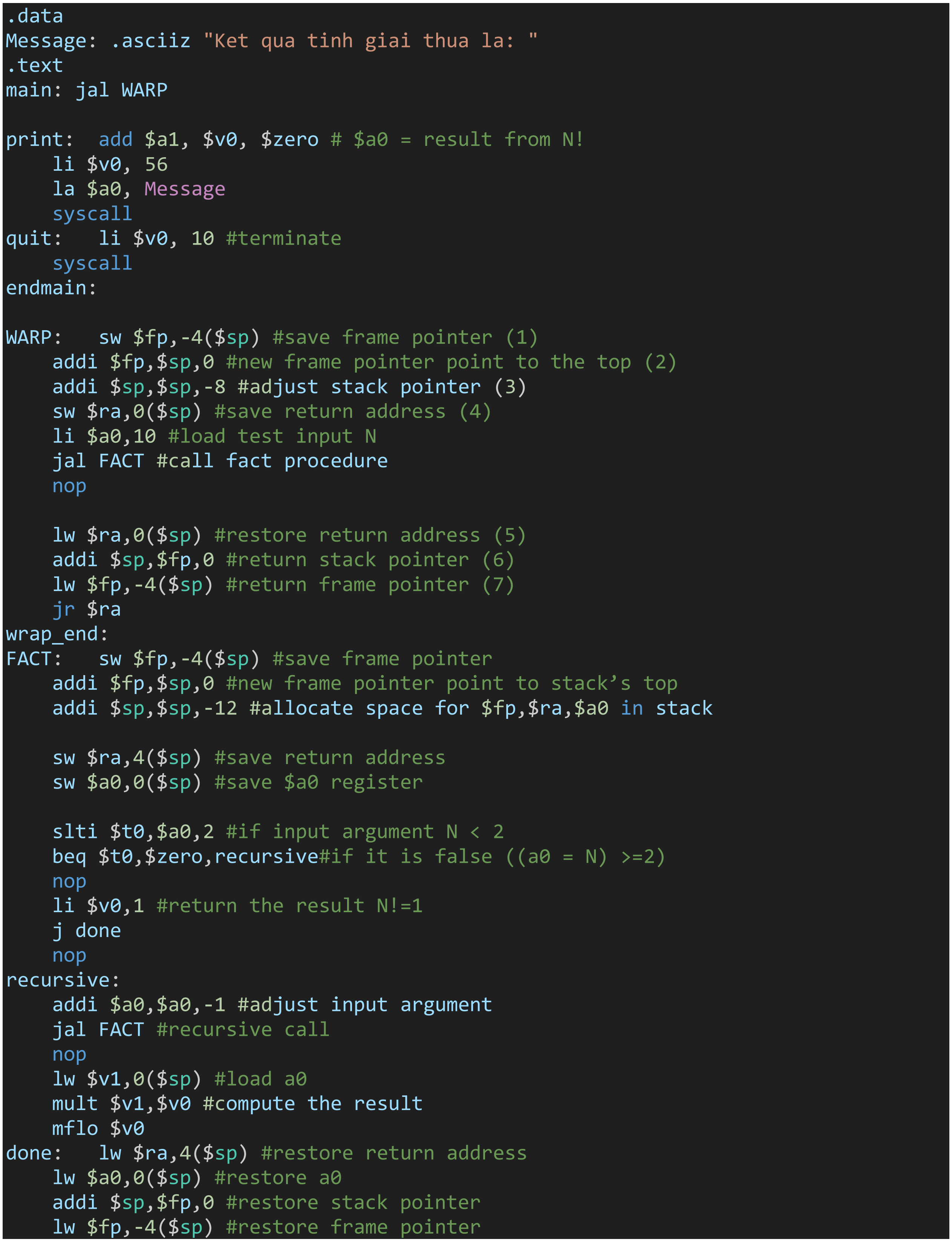
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0

0x10010000 (.data) ☑ Hexadecimal Addresses □ Hexadecimal Values □ ASCII

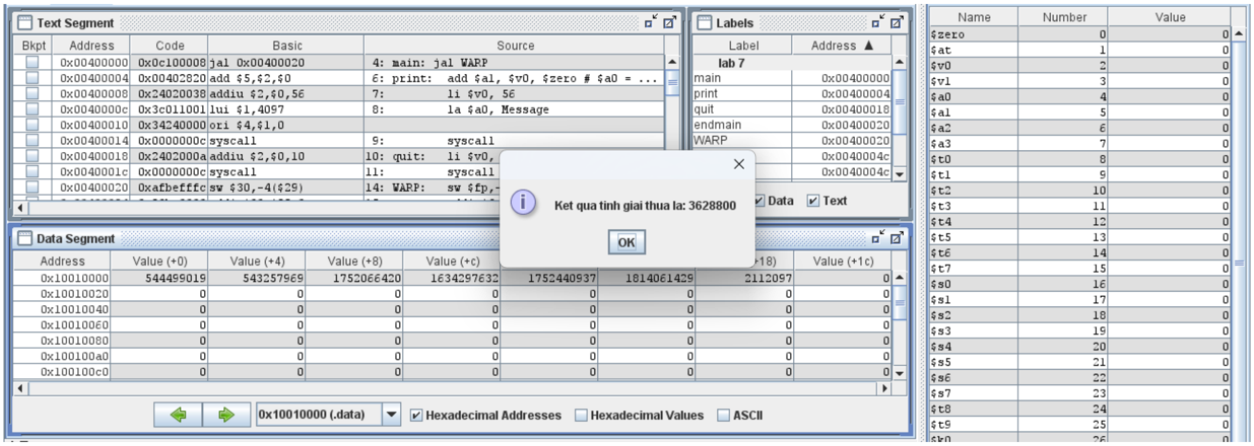
```
-- program is finished running (dropped off bottom) --
```

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$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	11
$s1	17	29
$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
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0

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 $a0,10 #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:
```



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x0c100008	jal 0x00400020	4: main: jal WARP
□	0x00400004	0x00402820	add $5,$2,$0	6: print: add $a1, $v0, $zero # $a0 = ...
□	0x00400008	0x24020038	addiu $2,$0,56	7: li $v0, 56
□	0x0040000c	0x3c011001	lui $1,4097	8: la $a0, Message
□	0x00400010	0x34240000	ori $4,$1,0	
□	0x00400014	0x0000000c	syscall	9: syscall
□	0x00400018	0x2402000a	addiu $2,$0,10	10: quit: li $v0,
□	0x0040001c	0x0000000c	syscall	11: syscall
□	0x00400020	0xafbefffc	sw $30,-4($29)	14: WARP: sw $fp,-

Labels

Label	Address ▲
lab 7	
main	0x00400000
print	0x00400004
quit	0x00400018
endmain	0x00400020
WARP	0x00400020
	0x0040004c
	0x0040004c

☑ Data ☑ Text

Ket qua tinh giai thua la: 3628800

OK

Data Segment

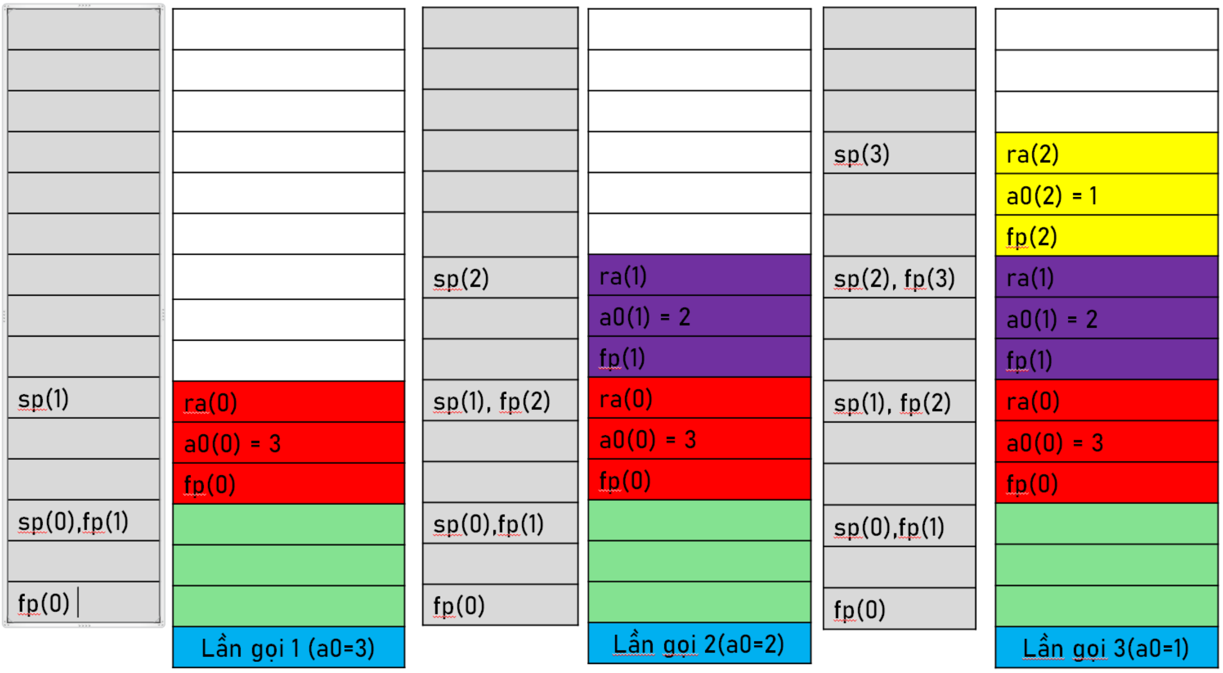
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)			+18)	Value (+1c)
0x10010000	544499019	543257969	1752066420	1634297632	1752440937	1814061429	2112097	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0

0x10010000 (.data) ☑ Hexadecimal Addresses □ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$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	0
$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

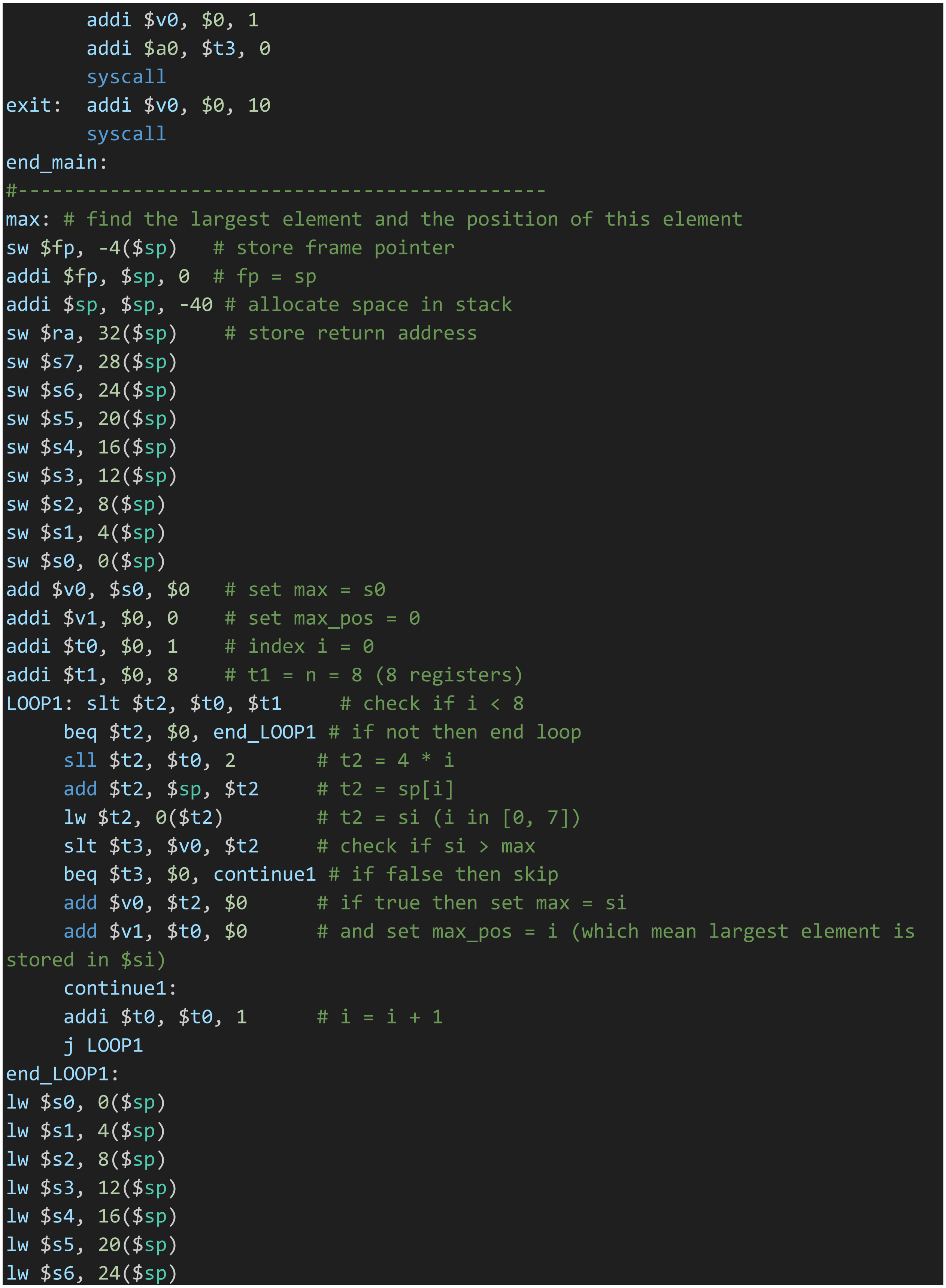
10! = 3628800 => Kết quả đúng.

- Draw the stack through this recursive program in case of n=3 (compute 3!).

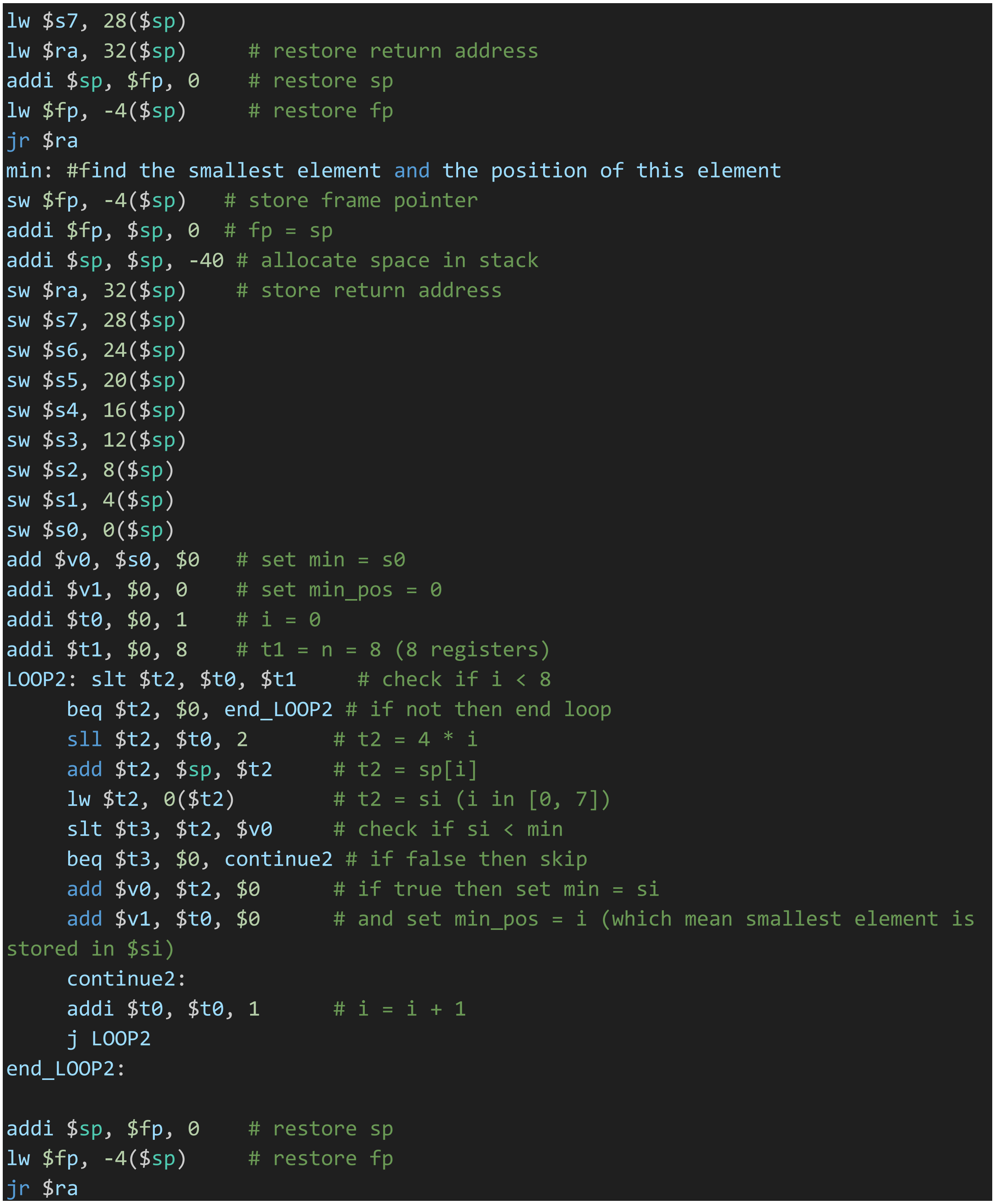


Assignment 5:

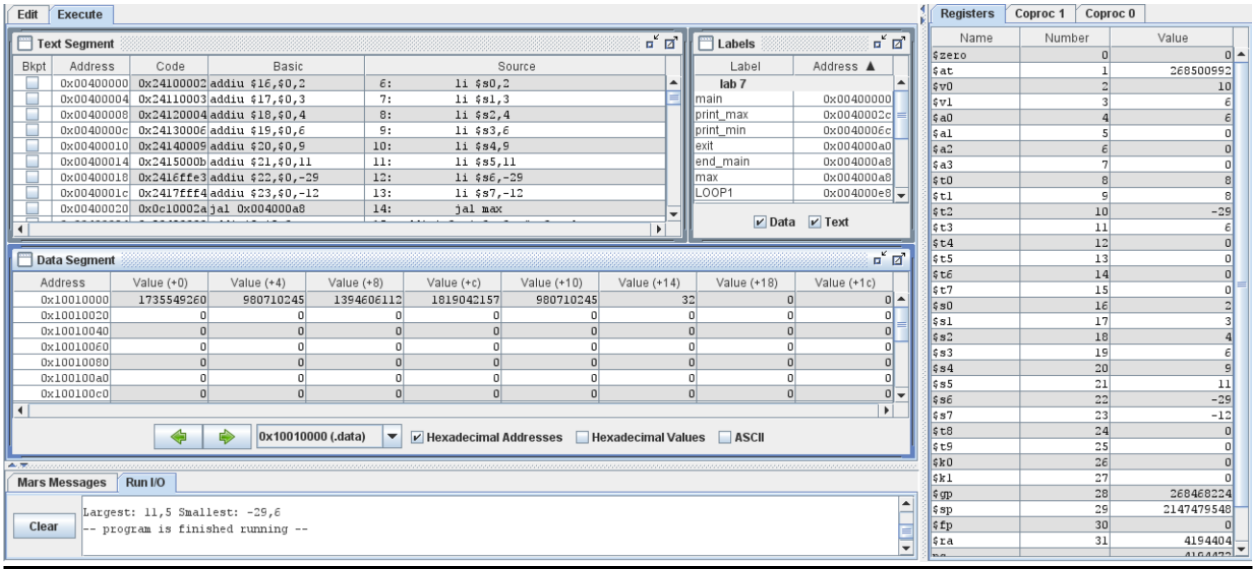
```
.data
mes1: .asciiz "Largest: "
mes2: .asciiz " Smallest: "
.text
main:
    li $s0,2
    li $s1,3
    li $s2,4
    li $s3,6
    li $s4,9
    li $s5,11
    li $s6,-29
    li $s7,-12
    jal max
addi $t0, $v0, 0  # t0 = largest
addi $t1, $v1, 0  # t1 = largest position
print_max:  # Print the largest element and the positon of this element
       addi $v0, $0, 4
       la $a0, mes1
       syscall
       addi $v0, $0, 1
       addi $a0, $t0, 0
       syscall
       addi $v0, $0, 11
       addi $a0, $0, 44
       syscall
       addi $v0, $0, 1
       addi $a0, $t1, 0
       syscall
jal min # Print the smallest element and the positon of this element
addi $t2, $v0, 0  # t2 = smallest
addi $t3, $v1, 0  # t3 = smallest position

print_min:
       addi $v0, $0, 4
       la $a0, mes2
       syscall
       addi $v0, $0, 1
       addi $a0, $t2, 0
       syscall
       addi $v0, $0, 11
       addi $a0, $0, 44
       syscall
```



```
       addi $v0, $0, 1
       addi $a0, $t3, 0
       syscall
exit:  addi $v0, $0, 10
       syscall
end_main:
#-----------------------------------------------
max: # find the largest element and the position of this element
sw $fp, -4($sp)   # store frame pointer
addi $fp, $sp, 0  # fp = sp
addi $sp, $sp, -40 # allocate space in stack
sw $ra, 32($sp)    # store return address
sw $s7, 28($sp)
sw $s6, 24($sp)
sw $s5, 20($sp)
sw $s4, 16($sp)
sw $s3, 12($sp)
sw $s2, 8($sp)
sw $s1, 4($sp)
sw $s0, 0($sp)
add $v0, $s0, $0   # set max = s0
addi $v1, $0, 0    # set max_pos = 0
addi $t0, $0, 1    # index i = 0
addi $t1, $0, 8    # t1 = n = 8 (8 registers)
LOOP1: slt $t2, $t0, $t1     # check if i < 8
     beq $t2, $0, end_LOOP1 # if not then end loop
     sll $t2, $t0, 2       # t2 = 4 * i
     add $t2, $sp, $t2     # t2 = sp[i]
     lw $t2, 0($t2)        # t2 = si (i in [0, 7])
     slt $t3, $v0, $t2     # check if si > max
     beq $t3, $0, continue1 # if false then skip
     add $v0, $t2, $0      # if true then set max = si
     add $v1, $t0, $0      # and set max_pos = i (which mean largest element is
stored in $si)
     continue1:
     addi $t0, $t0, 1      # i = i + 1
     j LOOP1
end_LOOP1:
lw $s0, 0($sp)
lw $s1, 4($sp)
lw $s2, 8($sp)
lw $s3, 12($sp)
lw $s4, 16($sp)
lw $s5, 20($sp)
lw $s6, 24($sp)
```



```
lw $s7, 28($sp)
lw $ra, 32($sp)     # restore return address
addi $sp, $fp, 0    # restore sp
lw $fp, -4($sp)     # restore fp
jr $ra
min: #find the smallest element and the position of this element
sw $fp, -4($sp)   # store frame pointer
addi $fp, $sp, 0  # fp = sp
addi $sp, $sp, -40 # allocate space in stack
sw $ra, 32($sp)    # store return address
sw $s7, 28($sp)
sw $s6, 24($sp)
sw $s5, 20($sp)
sw $s4, 16($sp)
sw $s3, 12($sp)
sw $s2, 8($sp)
sw $s1, 4($sp)
sw $s0, 0($sp)
add $v0, $s0, $0   # set min = s0
addi $v1, $0, 0    # set min_pos = 0
addi $t0, $0, 1    # i = 0
addi $t1, $0, 8    # t1 = n = 8 (8 registers)
LOOP2: slt $t2, $t0, $t1     # check if i < 8
     beq $t2, $0, end_LOOP2 # if not then end loop
     sll $t2, $t0, 2       # t2 = 4 * i
     add $t2, $sp, $t2     # t2 = sp[i]
     lw $t2, 0($t2)        # t2 = si (i in [0, 7])
     slt $t3, $t2, $v0     # check if si < min
     beq $t3, $0, continue2 # if false then skip
     add $v0, $t2, $0      # if true then set min = si
     add $v1, $t0, $0      # and set min_pos = i (which mean smallest element is
stored in $si)
     continue2:
     addi $t0, $t0, 1      # i = i + 1
     j LOOP2
end_LOOP2:

addi $sp, $fp, 0    # restore sp
lw $fp, -4($sp)     # restore fp
jr $ra
```



Text Segment

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x24100002	addiu $16,$0,2	6: li $s0,2
☐	0x00400004	0x24110003	addiu $17,$0,3	7: li $s1,3
☐	0x00400008	0x24120004	addiu $18,$0,4	8: li $s2,4
☐	0x0040000c	0x24130006	addiu $19,$0,6	9: li $s3,6
☐	0x00400010	0x24140009	addiu $20,$0,9	10: li $s4,9
☐	0x00400014	0x2415000b	addiu $21,$0,11	11: li $s5,11
☐	0x00400018	0x2416ffe3	addiu $22,$0,-29	12: li $s6,-29
☐	0x0040001c	0x2417fff4	addiu $23,$0,-12	13: li $s7,-12
☐	0x00400020	0x0c10002a	jal 0x004000a8	14: jal max

Labels

Label	Address ▲
lab 7	
main	0x00400000
print_max	0x0040002c
print_min	0x0040006c
exit	0x004000a0
end_main	0x004000a8
max	0x004000a8
LOOP1	0x004000e8

☑ Data ☑ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1735549260	980710245	1394606112	1819042157	980710245	32	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0

0x10010000 (.data) ☑ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

```
Largest: 11,5 Smallest: -29,6
-- program is finished running --
```

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	6
$a0	4	6
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	8
$t1	9	8
$t2	10	-29
$t3	11	6
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	2
$s1	17	3
$s2	18	4
$s3	19	6
$s4	20	9
$s5	21	11
$s6	22	-29
$s7	23	-12
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	4194404

-Giá trị lớn nhất là 11 được lưu ở thanh ghi $s5, giá trị bé nhất là -29 được lưu ở thah ghi $s6 => Kết quả đúng.
]