

Báo cáo thực hành Kiến trúc máy tính

Assignment 1:

CODE:

#Laboratory Exercise 7 Home Assignment 1

```
.text
    main: li $a0,-30          #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
```

Kết quả:

Đầu vào \$a0 = -30 đầu ra \$s0 = 30

\$t1	15	0
\$s0	16	30

Assignment 2:

CODE:

#Laboratory Exercise 7, Home Assignment 2

.text

main:

```
    li $a0, 12          # load test input
    li $a1, -3
    li $a2, 10
    jal max             # call max procedure
    nop
    add $s0, $zero, $v0  # $s0 = max   Largest number a $s0
    li $v0, 10
    syscall
```

endmain:

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 max = a1
```

okay:

```
    sub $t0, $a2, $v0    # compute (a2)-(v0)
    bltz $t0, done       # if (a2)-(v0)<0 then no change
    nop
    add $v0, $a2, $zero  # else max = a2
```

done:

```
    jr $ra              # return to calling program
```

Kết quả

Đầu vào:

\$a0	4	12
\$a1	5	-3
\$a2	6	10

Đầu ra:

\$s0	16	12
------	----	----

Detail:

Nhảy tới chương trình con max lệnh jal max

$V0 = a0$, $t0 = a1 - v0$ nếu $t0 < 0 \Rightarrow a1 < v0$ và nhảy đến okay nếu không thì $v0 = a1$ chuyển sang okay:

$T0 = a2 - v0$ nếu $t0 < 0 \Rightarrow a2 < v0$ và nhảy đến done nếu không thì $v0 = a2$ chuyển sang done

Done: nhảy về chương trình chính bằng jr \$ra

Assignment 3:

CODE:

#Laboratory Exercise 7, Home Assignment 3

.text

li \$s0, 7

li \$s1, 8

push:

addi \$sp,\$sp,-8 # add just stack pointer

sw \$s0,4(\$sp) # push \$s0 to stack

sw \$s1,0(\$sp) # push \$s1 to stack

work:

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

Kết quả:

Khởi tạo giá trị \$s1, \$s0

The screenshot displays the MIPS simulator interface. The 'Text Segment' window shows assembly code with the following instructions highlighted:

- 3: li \$s0, 7
- 4: li \$s1, 8
- 6: addi \$sp, \$sp, -8 # add just stack pointer
- 7: sw \$s0, 4(\$sp) # push \$s0 to stack
- 8: sw \$s1, 0(\$sp) # push \$s1 to stack
- 10: nop
- 11: nop
- 12: nop
- 14: lw \$s0, 0(\$sp) # pop from stack to \$s0
- 15: lw \$s1, 4(\$sp) # pop from stack to \$s1
- 16: addi \$sp, \$sp, 8 # adjust the stack pointer

The 'Data Segment' window shows a memory dump with addresses from 0x7ffffe00 to 0x7ffff100. The 'Labels' window shows the 'week7_2.asm' file with labels for 'push', 'work', and 'pop'. The 'Registers' window on the right shows the values of registers \$s0 and \$s1.

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	7
\$s1	17	8
\$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

Khởi tạo stack bằng cách đẩy stack pointer xuống 8

The screenshot displays the MIPS simulator interface. The 'Text Segment' window shows assembly code with the following instructions highlighted:

- 3: li \$s0, 7
- 4: li \$s1, 8
- 6: addi \$sp, \$sp, -8 # add just stack pointer
- 7: sw \$s0, 4(\$sp) # push \$s0 to stack
- 8: sw \$s1, 0(\$sp) # push \$s1 to stack
- 10: nop
- 11: nop
- 12: nop
- 14: lw \$s0, 0(\$sp) # pop from stack to \$s0
- 15: lw \$s1, 4(\$sp) # pop from stack to \$s1
- 16: addi \$sp, \$sp, 8 # adjust the stack pointer

The 'Data Segment' window shows a memory dump with addresses from 0x7ffffe00 to 0x7ffff100. The 'Labels' window shows the 'week7_2.asm' file with labels for 'push', 'work', and 'pop'. The 'Registers' window on the right shows the values of registers \$s0 and \$s1.

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	7
\$s1	17	8
\$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	2147479540
\$fp	30	0
\$ra	31	0

Các giá trị được load vào vùng nhớ stack

Text Segment					Labels	
Bkpt	Address	Code	Basic	Source	Label	Address
<input type="checkbox"/>	0x00400000	0x24100007	addiu \$16,\$0,7	3: li \$a0, 7	week7_2.asm	
<input type="checkbox"/>	0x00400004	0x24110008	addiu \$17,\$0,8	4: li \$a1, 8		
<input type="checkbox"/>	0x00400008	0x23bdfff8	addi \$29,\$29,-8	6: addi \$sp,\$sp,-8 # add just stack pointer		push 0x00400008
<input type="checkbox"/>	0x0040000c	0xafb00004	sw \$16,4(\$29)	7: sw \$a0,4(\$sp) # push \$a0 to stack		work 0x00400014
<input type="checkbox"/>	0x00400010	0xafb10000	sw \$17,0(\$29)	8: sw \$a1,0(\$sp) # push \$a1 to stack		pop 0x00400020
<input type="checkbox"/>	0x00400014	0x00000000	nop	10: nop		
<input type="checkbox"/>	0x00400018	0x00000000	nop	11: nop		
<input type="checkbox"/>	0x0040001c	0x00000000	nop	12: nop		
<input type="checkbox"/>	0x00400020	0x8fb00000	lw \$16,0(\$29)	14: lw \$a0,0(\$sp) # pop from stack to \$a0		
<input type="checkbox"/>	0x00400024	0x8fb10004	lw \$17,4(\$29)	15: lw \$a1,4(\$sp) # pop from stack to \$a1		
<input type="checkbox"/>	0x00400028	0x23bd0008	addi \$29,\$29,8	16: addi \$sp,\$sp,8 # adjust the stack pointer		

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7ffffe0	0	0	0	0	0	8	7	0

Giá trị 2 thanh ghi ban đầu

\$s0	16	7
\$s1	17	8

Giá trị 2 thanh ghi sau khi pop ra từ stack

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x24100007	addiu \$16,\$0,7	3: li \$a0, 7
<input type="checkbox"/>	0x00400004	0x24110008	addiu \$17,\$0,8	4: li \$a1, 8
<input type="checkbox"/>	0x00400008	0x23bdfff8	addi \$29,\$29,-8	6: addi \$sp,\$sp,-8 # add just stack pointer
<input type="checkbox"/>	0x0040000c	0xafb00004	sw \$16,4(\$29)	7: sw \$a0,4(\$sp) # push \$a0 to stack
<input type="checkbox"/>	0x00400010	0xafb10000	sw \$17,0(\$29)	8: sw \$a1,0(\$sp) # push \$a1 to stack
<input type="checkbox"/>	0x00400014	0x00000000	nop	10: nop
<input type="checkbox"/>	0x00400018	0x00000000	nop	11: nop
<input type="checkbox"/>	0x0040001c	0x00000000	nop	12: nop
<input type="checkbox"/>	0x00400020	0x8fb00000	lw \$16,0(\$29)	14: lw \$a0,0(\$sp) # pop from stack to \$a0
<input type="checkbox"/>	0x00400024	0x8fb10004	lw \$17,4(\$29)	15: lw \$a1,4(\$sp) # pop from stack to \$a1
<input type="checkbox"/>	0x00400028	0x23bd0008	addi \$29,\$29,8	16: addi \$sp,\$sp,8 # adjust the stack pointer

Labels

Label	Address
week7_2.asm	
push	0x00400008
work	0x00400014
pop	0x00400020

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7ffffe0	0	0	0	0	0	8	7	0
0x7ffff00	0	0	0	0	0	0	0	0
0x7ffff04	0	0	0	0	0	0	0	0
0x7ffff08	0	0	0	0	0	0	0	0
0x7ffff0c	0	0	0	0	0	0	0	0
0x7ffff10	0	0	0	0	0	0	0	0
0x7ffff14	0	0	0	0	0	0	0	0
0x7ffff18	0	0	0	0	0	0	0	0
0x7ffff1c	0	0	0	0	0	0	0	0
0x7ffff20	0	0	0	0	0	0	0	0
0x7ffff24	0	0	0	0	0	0	0	0
0x7ffff28	0	0	0	0	0	0	0	0
0x7ffff2c	0	0	0	0	0	0	0	0
0x7ffff30	0	0	0	0	0	0	0	0
0x7ffff34	0	0	0	0	0	0	0	0
0x7ffff38	0	0	0	0	0	0	0	0
0x7ffff3c	0	0	0	0	0	0	0	0
0x7ffff40	0	0	0	0	0	0	0	0
0x7ffff44	0	0	0	0	0	0	0	0
0x7ffff48	0	0	0	0	0	0	0	0
0x7ffff4c	0	0	0	0	0	0	0	0
0x7ffff50	0	0	0	0	0	0	0	0
0x7ffff54	0	0	0	0	0	0	0	0
0x7ffff58	0	0	0	0	0	0	0	0
0x7ffff5c	0	0	0	0	0	0	0	0
0x7ffff60	0	0	0	0	0	0	0	0
0x7ffff64	0	0	0	0	0	0	0	0
0x7ffff68	0	0	0	0	0	0	0	0
0x7ffff6c	0	0	0	0	0	0	0	0
0x7ffff70	0	0	0	0	0	0	0	0
0x7ffff74	0	0	0	0	0	0	0	0
0x7ffff78	0	0	0	0	0	0	0	0
0x7ffff7c	0	0	0	0	0	0	0	0
0x7ffff80	0	0	0	0	0	0	0	0
0x7ffff84	0	0	0	0	0	0	0	0
0x7ffff88	0	0	0	0	0	0	0	0
0x7ffff8c	0	0	0	0	0	0	0	0
0x7ffff90	0	0	0	0	0	0	0	0
0x7ffff94	0	0	0	0	0	0	0	0
0x7ffff98	0	0	0	0	0	0	0	0
0x7ffff9c	0	0	0	0	0	0	0	0
0x7ffffa0	0	0	0	0	0	0	0	0
0x7ffffa4	0	0	0	0	0	0	0	0
0x7ffffa8	0	0	0	0	0	0	0	0
0x7ffffac	0	0	0	0	0	0	0	0
0x7ffffb0	0	0	0	0	0	0	0	0
0x7ffffb4	0	0	0	0	0	0	0	0
0x7ffffb8	0	0	0	0	0	0	0	0
0x7ffffbc	0	0	0	0	0	0	0	0
0x7ffffc0	0	0	0	0	0	0	0	0
0x7ffffc4	0	0	0	0	0	0	0	0
0x7ffffc8	0	0	0	0	0	0	0	0
0x7ffffcc	0	0	0	0	0	0	0	0
0x7ffffd0	0	0	0	0	0	0	0	0
0x7ffffd4	0	0	0	0	0	0	0	0
0x7ffffd8	0	0	0	0	0	0	0	0
0x7ffffdc	0	0	0	0	0	0	0	0
0x7ffffe0	0	0	0	0	0	0	0	0
0x7ffffe4	0	0	0	0	0	0	0	0
0x7ffffe8	0	0	0	0	0	0	0	0
0x7ffffec	0	0	0	0	0	0	0	0
0x7fffff0	0	0	0	0	0	0	0	0
0x7fffff4	0	0	0	0	0	0	0	0
0x7fffff8	0	0	0	0	0	0	0	0
0x7fffffc	0	0	0	0	0	0	0	0

current \$sp

☒ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

Name

Number

Value

\$zero	0	0
\$a0	1	0
\$a1	2	0
\$a2	3	0
\$a3	4	0
\$a4	5	0
\$a5	6	0
\$a6	7	0
\$a7	8	0
\$a8	9	0
\$a9	10	0
\$a10	11	0
\$a11	12	0
\$a12	13	0
\$a13	14	0
\$a14	15	0
\$a15	16	8
\$a16	17	7
\$a17	18	0
\$a18	19	0
\$a19	20	0
\$a20	21	0
\$a21	22	0
\$a22	23	0
\$a23	24	0
\$a24	25	0
\$a25	26	0
\$a26	27	0
\$a27	28	268468224
\$a28	29	2147479540
\$a29	30	0
\$a30	31	0

a) Sử dụng \$fp

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 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:
#-----
#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

```

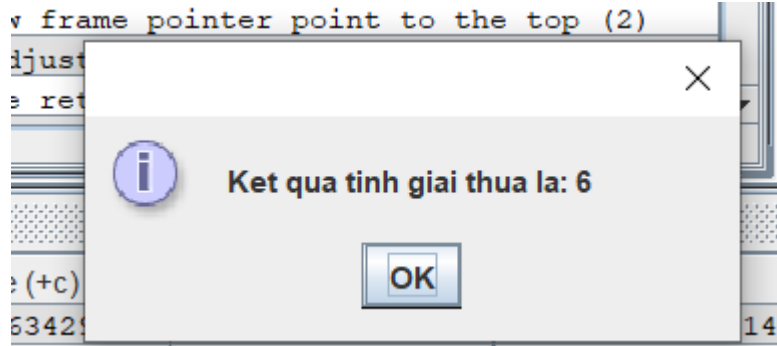
`addi $sp,$fp,0 #restore stack pointer`

`lw $fp,-4($sp) #restore frame pointer`

`jr $ra #jump to calling`

`fact_end:`

Kết quả: Đầu vào a0 =3; đầu ra a1 =6



Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
2147479488	0	0	0	0	1	4194432	2147479528	2
2147479520	4194432	2147479540	3	4194360	2147479548	4194308	0	0

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
0x7ffffefc0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000001	0x00400080	0x7ffffef8	0x00000002
0x7ffffefc4	0x00400080	0x7ffffef4	0x00000003	0x00400038	0x7ffffefc	0x00400004	0x00000000	0x00000000

=>Stack

\$a0	0x00000001
\$ra	0x00400080
\$fp	0x7ffffef8
\$a0	0x00000002
\$ra	0x00400080
\$fp	0x7ffffef4
\$a0	0x00000003
\$ra	0x00400038
\$fp	0x7ffffefc
\$ra	0x00400004

\$a0, \$fp, \$ra lần gọi vòng lặp thứ ba

0x00000001	0x00400080	0x7ffffef8
------------	------------	------------

\$a0, \$fp, \$ra lần gọi gọi vòng lặp thứ hai

0x00000002	0x7ffffeff4	0x00000003
------------	-------------	------------

\$a0, \$fp, \$ra lần gọi gọi vòng lặp thứ nhất

0x00000003	0x00400038	0x7ffffeffc
------------	------------	-------------

b) Không sử dụng \$fp

#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

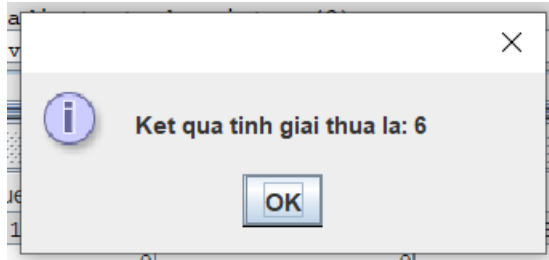
```

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:



Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
2147479488	0	0	0	0	0	0	0	1
2147479520	4194416	2	4194416	3	4194352	4194308	0	0

Điểm khác không sử dụng \$fp thì ta trừ lại kích cỡ của stack ở mỗi vòng lặp để lấy giá trị và quay lại vòng lặp ban đầu

Assignment 5:

#Laboratory Exercise 7, Assignment 5

.data

Mess1: .asciiz "Largest: "

Mess2: .asciiz "\nSmallest:"

Comma: .asciiz ", "

.text

main: li \$s0, 2 # Load input

li \$s1, 3

li \$s2, -1

li \$s3, 4

li \$s4, 9

li \$s5, -2

li \$s6, 8

```

li $s7, 5
jal init
nop
li $v0, 4
la $a0, Mess1      #print mess1
syscall
li $v0 , 1
add $a0,$t0,$zero
syscall            #print max value
li $v0, 4
la $a0, Comma      #print ","
syscall
li $v0 , 1
add $a0,$t5,$zero  # print max value's position
syscall
li $v0, 4
la $a0, Mess2      #print mess2
syscall
li $v0 ,1
add $a0,$t1,$zero
syscall            #print min value
li $v0 ,4
la $a0, Comma      #print ","
syscall
li $v0 , 1
add $a0,$t6,$zero
syscall            # print min value's position
li $v0, 10
syscall            # exit

```

endmain:

swapMax:add \$t0,\$t3,\$zero # set Max = \$t3

 add \$t5,\$t2,\$zero # set i of max = \$t2

 jr \$ra

swapMin:add \$t1,\$t3,\$zero # set Min = \$t3

 add \$t6,\$t2,\$zero # set i of min = \$t2

 jr \$ra

init: add \$fp,\$sp,\$zero # save address of origin sp

 addi \$sp,\$sp, -32 # create space for stack

 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) # save \$ra for main

 add \$t0,\$s0,\$zero # set Max = \$s0

 add \$t1,\$s0,\$zero # set Min = \$s0

 li \$t5, 0 # set \$t5 to 0

 li \$t6, 0 # set \$t6 to 0

 li \$t2, 0 # set \$t2 to 0 , i = 0

max_min:addi \$sp,\$sp,4

 lw \$t3,-4(\$sp)

 sub \$t4, \$sp, \$fp # check if meet \$ra

 beq \$t4,\$zero, done # if true then done

 addi \$t2,\$t2,1 # i++

 sub \$t4,\$t0,\$t3 # Max - \$t3

 bltzal \$t4, swapMax # if Max - \$t3 < 0, swap Max

```

        sub $t4,$t3,$t1      # $t3 - Min
        bltzal $t4, swapMin   # if $t3 – Min < 0 , swap Min
        j max_min             # repeat
done:   lw $ra, -4($sp)        # load #$ra
        jr $ra                # return

```

Kết quả:

Init : Khởi tạo giá trị cho mảng và lưu vào stack

In ra giá trị lớn nhất và vị trí : 9 tại \$s4

In ra giá trị bé nhất và vị trí : -2 tại \$s5