BÀI BÁO CÁO TUẦN 7

Họ và tên: Nguyễn Trọng Khánh Duy

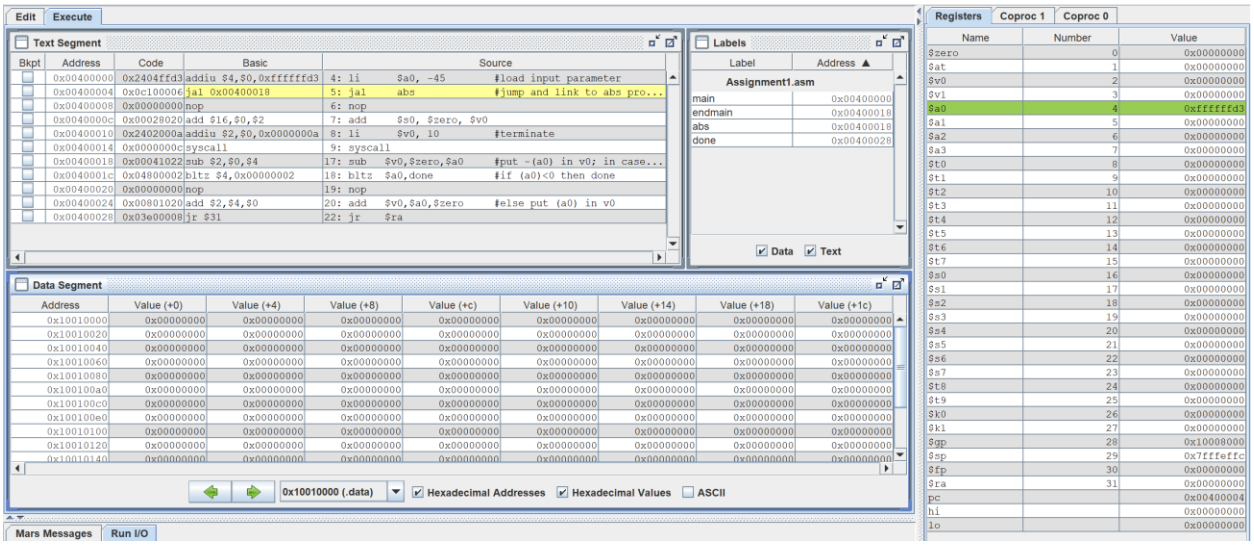
MSSV: 20210284

Assignment 1

Trước khi chạy lệnh `jal abs` :

Thanh ghi `pc` = `0x004000004` (địa chỉ của câu lệnh `jal abs`)

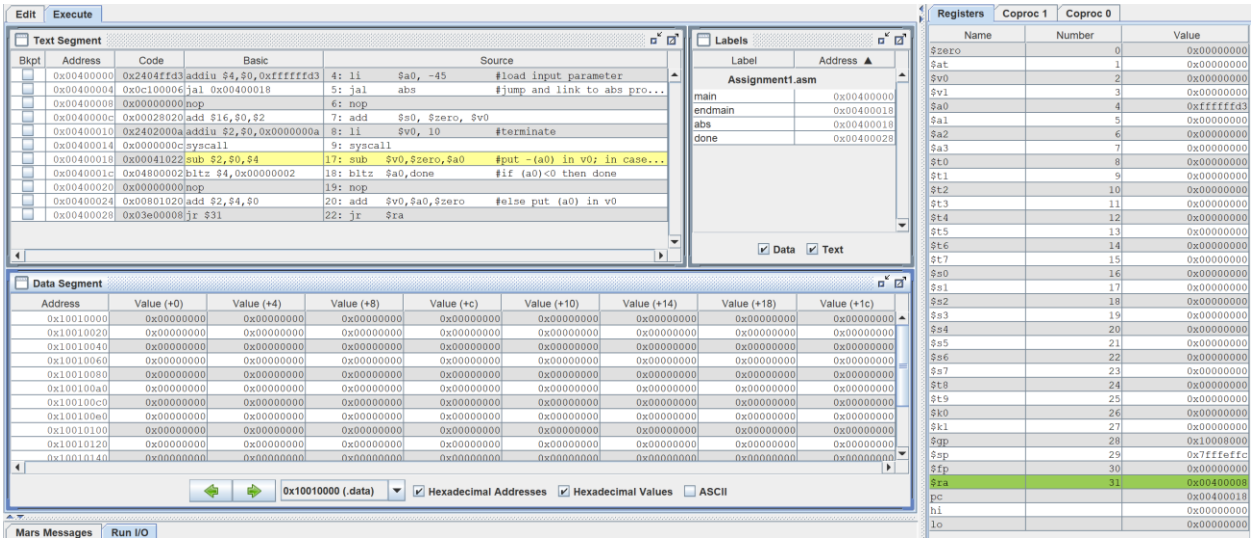
Thanh ghi `$ra` = `0x0000000` không thay đổi



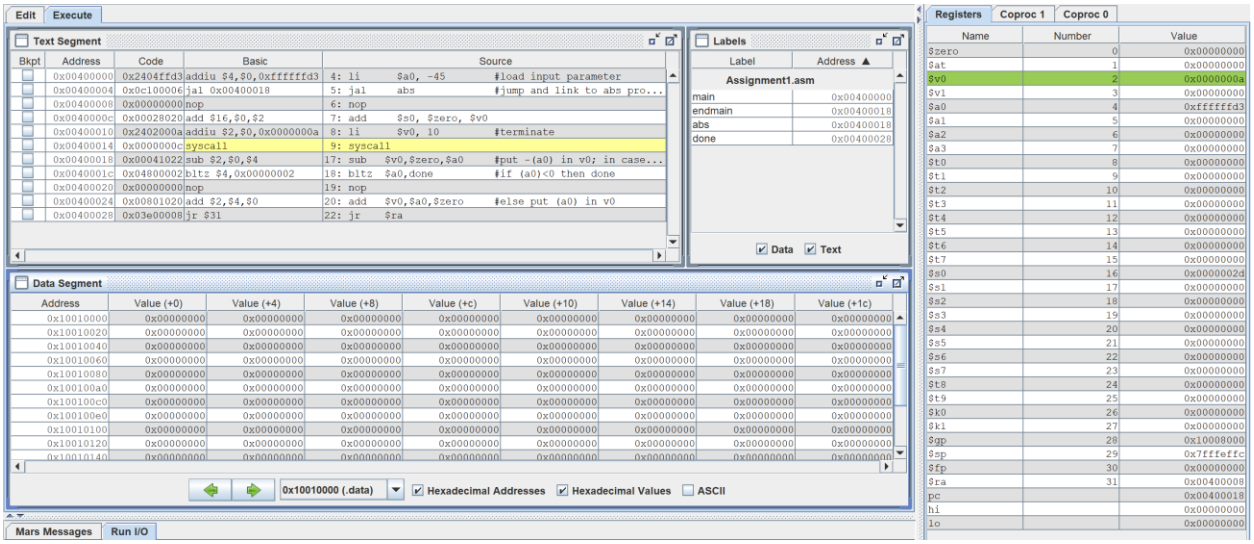
Trước khi chạy lệnh `jal abs` :

Thanh ghi `pc = 0x004000018` (địa chỉ của câu lệnh đầu của nhãn `abs`)

Thanh ghi `$ra = 0x004000008`



Sau khi chạy hết chương trình:



Nhận xét:

Sau khi chạy lệnh `jal abs` (địa chỉ câu lệnh `0x004000004`)thì thanh ghi `$ra` được gán bằng địa chỉ câu lệnh tiếp theo là `0x00400008`. Và lúc này thanh ghi `pc` sẽ nhảy đến địa chỉ `0x004000018` là địa chỉ câu lệnh đầu tiên của nhãn `abs`

Kết quả cuối cùng cho ra được giá trị tuyệt đối của số được nạp vào thanh ghi `$a0` được lưu vào thanh ghi `$s0`

$a0	4	-45

$s0	16	45

Assignment 2

Sau khi chạy lệnh:

```
li $a0, -10
li $a1, 15
li $a2, -20
```

thì giá trị của 3 thanh ghi `$a0`, `$a1`, `$a2`, được cập nhật giá trị

$a0	4	-10
$a1	5	15
$a2	6	-20

Thanh ghi `pc = 0x00400000c` (địa chỉ của câu lệnh `jal max`)

Thanh ghi `$ra = 0x0000000` không thay đổi

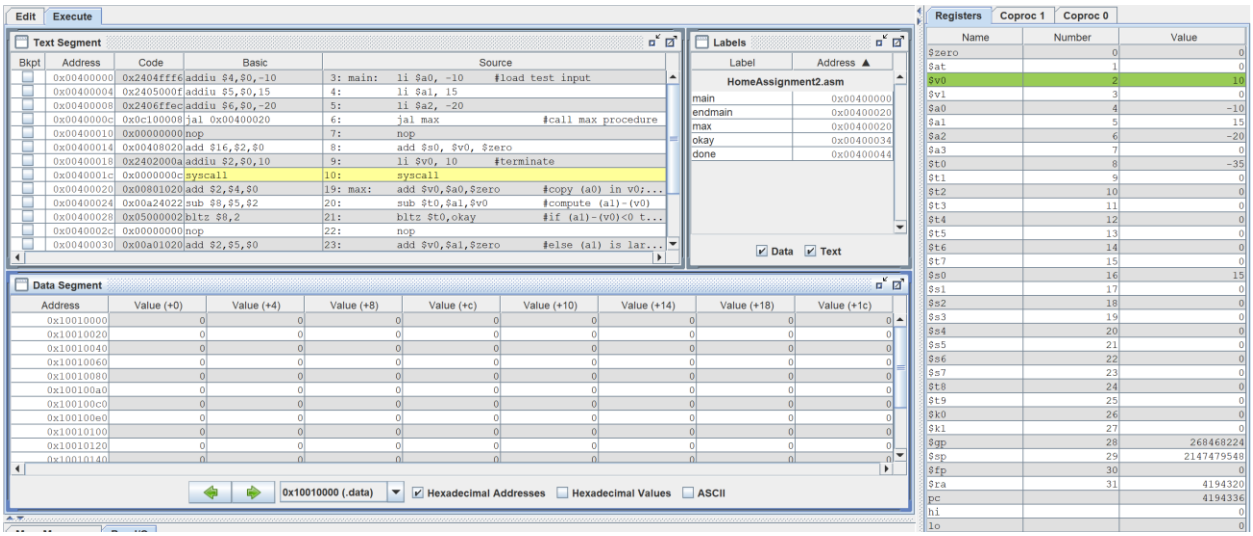
$ra	31	0x00000000
pc		0x0040000c

Sau khi chạy lệnh: `jal max`

Thanh ghi `pc = 0x004000020` (địa chỉ của câu lệnh đầu của nhãn `max`)

Thanh ghi `$ra = 0x004000010`

Sau khi chạy hết chương trình:



Kết quả trả về: `$s0 = 15`

<u>Nhận xét:</u>

Sau khi chạy lệnh `jal max` (địa chỉ câu lệnh `0x00400000c`)thì thanh ghi `$ra` được gán bằng địa chỉ câu lệnh tiếp theo là `0x00400010`. Và lúc này thanh ghi `pc` sẽ nhảy đến địa chỉ `0x004000020` là địa chỉ câu lệnh đầu tiên của nhãn `max`

Assignment 3

CODE:

```
.text
        li $s0, 6
        li $s1, -9
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
```

Nhận xét:

Khi chạy lệnh addi $sp,$sp,-8 thanh ghi sp trừ đi 8 để chuẩ bị lưu giá trị cho $s0 và $s1

$sp	29	0x7fffeffc
$sp	29	0x7fffeff4

Sau hai lệnh sw, giá trị $s0 và $s1 được lưu vào stack:

0x7fffefe0	..	..	..	..	0xfffffff7	0x00000006
0x7ffff000	..	..	..	..	0x00000000	0x00000000

Kết quả cuối cùng hai giá trị $s0 và $s1 được đổi chỗ cho nhau

$s0	16	-9
$s1	17	6

Thanh ghi sp được trả về giá trị cũ bằng lệnh addi $sp,$sp,8

$sp	29	0x7fffeff4

$sp	29	0x7fffeffc

Assignment 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 $a0,3          #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
```

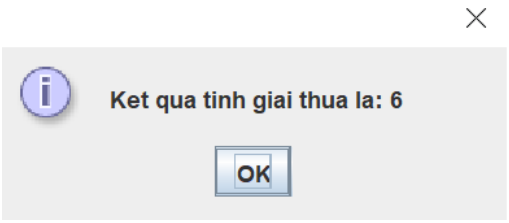
```
        mflo $v0
done:
        lw $ra,4($sp)     #restore return address
        lw $a0,0($sp)     #restore a0
        addi $sp,$fp,0    #restore stack pointer
        lw $fp,-4($sp)    #restore frame pointer
        jr $ra            #jump to calling
fact_end:
```

Kết quả chạy:

Các giá trị được lưu trong stack:

0x7fffefc0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000001	0x00400080	0x7fffefe8	0x00000002
0x7fffefe0	0x00400080	0x7fffeff4	0x00000003	0x00400038	0x7fffeffc	0x00400004	0x00000000	0x00000000

Kết quả chạy xong chương trình tính 3!:



Bảng thể hiện giá trị ngăn xếp:

0x7fffefd0	$a0 = 0x00000001
0x7fffefd4	$ra = 0x00400080
0x7fffefd8	$fp = 0x7fffefe8
0x7fffefdc	$a0 = 0x00000002
0x7fffefe0	$ra = 0x00400080
0x7fffefe4	$fp = 0x7fffeff4
0x7fffefe8	$a0 = 0x00000003
0x7fffefec	$ra = 0x00400038
0x7fffeff0	$fp = 0x7fffeffc
0x7fffeff4	$ra = 0x00400004
0x7fffeff8	$fp = 0x00000000

Assignment 5

CODE:

```
.data
    largest: .asciiz "Largest: "
    smallest: .asciiz "\nSmallest: "
    comma: .asciiz ", "
.text
main:
    li $s0, 3
    li $s1, 5
    li $s2, -45
    li $s3, 6
    li $s4, 27
    li $s5, -1
    li $s6, 666
    li $s7, -9

    jal saveNumbers
    nop
    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 Min
    syscall
endmain:
    li $v0, 10          # Exit
    syscall
#------------------------------------------------------------------
# Return $t0 = Max
# Return $t1 = Min
# Index of Max = $t5
# Index of Min = $t6
#return $v0 the largest value
#------------------------------------------------------------------
swapMax:
    add $t0,$t3,$zero
    add $t5,$t2,$zero
    jr $ra
swapMin:
    add $t1,$t3,$zero
    add $t6,$t2,$zero
    jr $ra
saveNumbers:
    add $t9,$sp,$zero       # Save address of origin $sp
    addi $sp,$sp, -32
    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         # Max = $s0
    add $t1,$s0,$zero         # Min = $s0
    li $t5, 0            # Index of Max to 0
    li $t6, 0            # Index of Min to 0
    li $t2, 0            # i = 0
findMaxMin:
    addi $sp,$sp,4
    lw $t3,-4($sp)
    sub $t4, $sp, $t9
    beq $t4,$zero, done # If $sp = $fp branch to the 'done'
    nop
    addi $t2,$t2,1       # i++
    sub $t4,$t0,$t3
    bltzal $t4, swapMax # If $t3 > Max branch to the
swapMax:
    nop
    sub $t4,$t3,$t1
    bltzal $t4, swapMin # If $t3 < Min branch to the
swapMin:
    nop
    j findMaxMin              # Repeat
done:
    lw $ra, -4($sp)
    jr $ra               # Return to calling program
```

Kết quả chương trình:

Các giá trị trong ngắn xếp:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x7fffefc0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000005
0x7fffefe0	0xfffffffd3	0x00000006	0x0000001b	0xffffffff	0x0000029a	0xfffffff7	0x00400024	0x00000000
0x7ffff000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffff020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffff040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffff060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x7ffff080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Kết quả cuối:

```
Largest: 666, 6
Smallest: -45, 2
-- program is finished running --
```