

# Assignment 1:

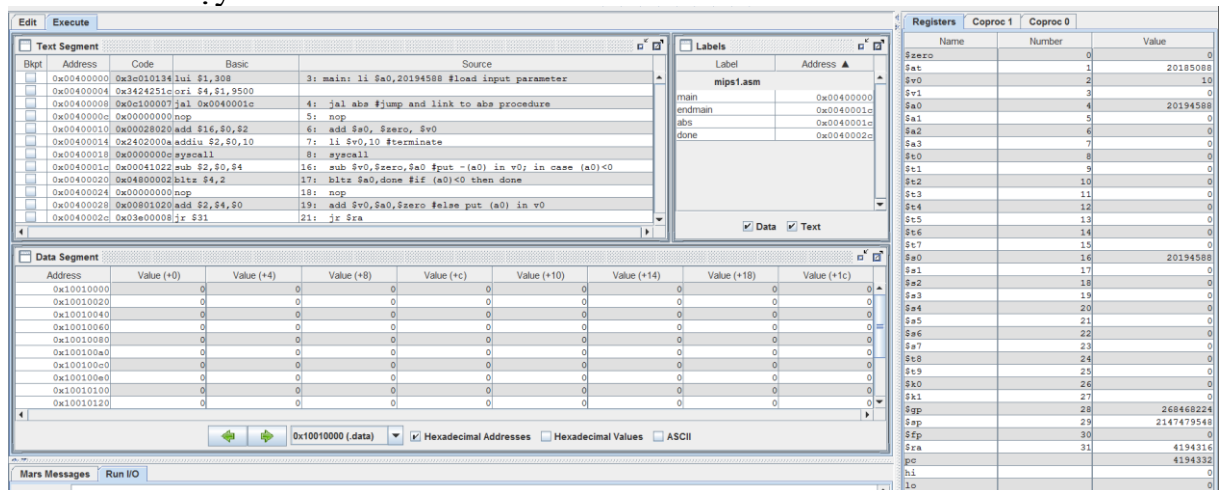
- Mã nguồn:

```

mips1.asm
3  main: li $a0,20194588 #load input parameter
4      jal abs #jump and link to abs procedure
5      nop
6      add $s0, $zero, $v0
7      li $v0,10 #terminate
8      syscall
9  endmain:
10 #-----
11 # function abs
12 # param[in] $a1 the interger need to be gained the absolute value
13 # return $v0 absolute value
14 #-----
15 abs:
16     sub $v0,$zero,$a0 #put -(a0) in v0; in case (a0)<0
17     bltz $a0,done #if (a0)<0 then done
18     nop
19     add $v0,$a0,$zero #else put (a0) in v0
20 done:
21     jr $ra
22

```

- Màn hình chạy:



\$a0	Đối số đầu vào
\$v0	Lưu giá trị tuyệt đối của đối số
\$s0	Lưu giá trị tuyệt đối để nhường thanh ghi \$v0 cho syscall
\$ra	Return, kết thúc procedure abs

- Giải thích:

- Dòng 3: Khai báo
- Dòng 4: Nhảy đến abs
- Abs (Dòng 15-19):  $v0 = -a0$  nếu  $a0 < 0$ , còn lại giữ nguyên
- Done (Dòng 20,21): nhảy đến địa chỉ lưu trong thanh ghi (\$ra)
- Dòng 6:  $s0 = v0$
- Dòng 7,8: In ra màn hình

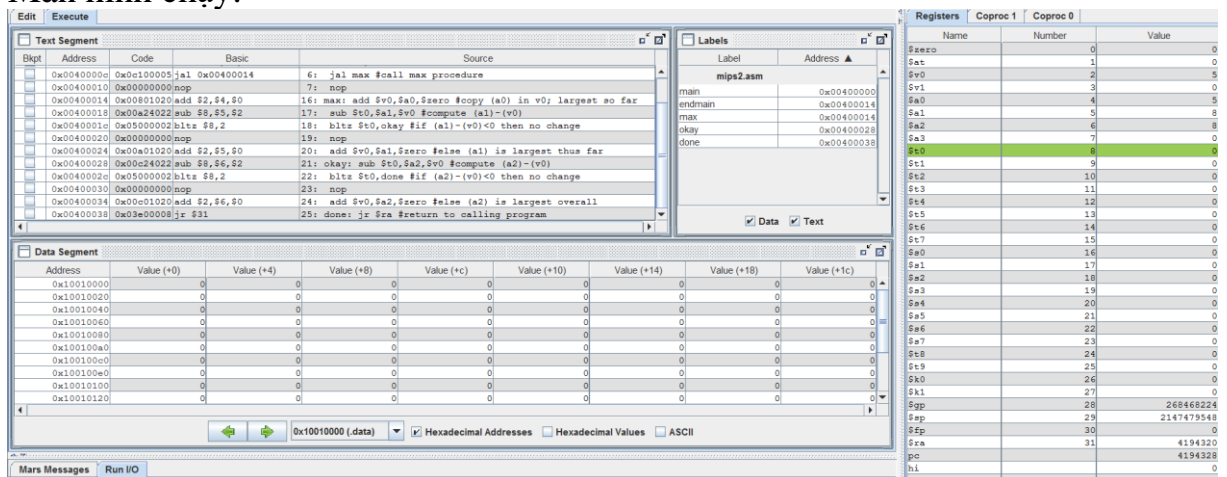
## Assignment 2:

- Mã nguồn:

```

mips1.asm  mips2.asm
1  #Laboratory Exercise 7, Home Assignment 2
2  .text
3  main: li $a0,5 #load test input
4  li $a1,8
5  li $a2,8
6  jal max #call max procedure
7  nop
8  endmain:
9  #-----
10 #Procedure max: find the largest of three integers
11 #param[in] $a0 integers
12 #param[in] $a1 integers
13 #param[in] $a2 integers
14 #return $v0 the largest value
15 #-----
16 max: add $v0,$a0,$zero #copy (a0) in v0; largest so far
17 sub $t0,$a1,$v0 #compute (a1)-(v0)
18 bltz $t0,okay #if (a1)-(v0)<0 then no change
19 nop
20 add $v0,$a1,$zero #else (a1) is largest thus far
21 okay: sub $t0,$a2,$v0 #compute (a2)-(v0)
22 bltz $t0,done #if (a2)-(v0)<0 then no change
23 nop
24 add $v0,$a2,$zero #else (a2) is largest overall
25 done: jr $ra #return to calling program
  
```

- Màn hình chạy:



\$a0	Đối số đầu vào
\$a1	Đối số đầu vào
\$a2	Đối số đầu vào
\$v0	Lưu số lớn nhất
\$t0	Biến sử dụng cho phép so sánh <0 ( hiệu của hai số)

- Giải thích:
  - Dòng 3-5: Khai báo
  - Dòng 6: Nhảy đến thủ tục max
  - Max (Dòng 16-21):
    - Dòng 17-18: Nếu  $a1 < v0$  thì đến hàm okay
    - Dòng 20:  $v0 = a1$  (Hiện tại  $a1$  đang là max)
  - Okay (Dòng 21-24): Kiểm tra tương tự Max
  - Dòng 25: Done: nhảy đến địa chỉ lưu trong thanh ghi (\$ra)

## Assignment 3:

- Mã nguồn:

```

1  #Laboratory Exercise 7, Home Assignment 3
2  .text
3  push:
4  li $s0, 2019
5  li $s1, 4588
6  addi $sp, $sp, -8 #adjust the stack pointer
7  sw $s0, 4($sp) #push $s0 to stack
8  sw $s1, 0($sp) #push $s1 to stack
9  work: nop
10 nop
11 nop
12 pop: lw $s0, 0($sp) #pop from stack to $s0
13     lw $s1, 4($sp) #pop from stack to $s1
14     addi $sp, $sp, 8 #adjust the stack pointer

```

- Màn hình chạy:

The screenshot shows the Mars MIPS simulator interface. The 'Text Segment' window displays the assembly code with addresses. The 'Data Segment' window shows memory values. The 'Registers' window shows the state of MIPS registers, with \$s0 and \$s1 containing 2019 and 4588 respectively. The 'Mars Messages' window shows the message 'program is finished running (dropped off bottom)'.

\$s0	Đối số đầu vào
\$s1	Đối số đầu vào
\$sp	Đóng vai trò như một ngăn xếp để lưu các giá trị

- Giải thích:

Thủ tục push:

- Dòng 4,5: Khai báo s0,s1
- Dòng 6: Khai báo 1 stack chứa được 2 số nguyên (giảm \$sp đi 8)
- Dòng 7,8: Lưu lần lượt \$s0, \$s1 vào \$sp[1], \$sp[0]

Thủ tục pop:

- Dòng 12,13: Lấy ra giá trị lần lượt \$sp[0], \$sp[1]
- Dòng 14: lưu vào \$s0, \$s1 (swap \$s0, \$s1)

## Assignment 4:

- Mã nguồn:

```

mips1.asm  mips2.asm  mips3.asm  mips4.asm  mips5.asm
1  #Laboratory Exercise 7, Home Assignment 4
2  .data
3  Message: .asciiz "Ket qua tinh giai thua la: "
4  .text
5  main: jal WARP
6
7  print: add $a1, $v0, $zero # $a0 = result from N!
8      li $v0, 56
9      la $a0, Message
10     syscall
11  quit: li $v0, 10 #terminate
12     syscall
13  endmain:
14  #-----
15  #Procedure WARP: assign value and call FACT
16  #-----
17  WARP: sw $fp, -4($sp) #save frame pointer (1)
18      addi $fp, $sp, 0 #new frame pointer point to the top (2)
19      addi $sp, $sp, -8 #adjust stack pointer (3)
20      sw $ra, 0($sp) #save return address (4)
21      li $a0 8 #load test input N

```

```

22  jal FACT #call fact procedure
23  nop
24
25  lw $ra,0($sp) #restore return address (5)
26  addi $sp,$fp,0 #return stack pointer (6)
27  lw $fp,-4($sp) #return frame pointer (7)
28  jr $ra
29  wrap_end:
30  #-----
31  #Procedure FACT: compute N!
32  #param[in] $a0 integer N
33  #return $v0 the largest value
34  #-----
35  FACT: sw $fp,-4($sp) #save frame pointer
36  addi $fp,$sp,0 #new frame pointer point to stack's top
37  addi $sp,$sp,-12 #allocate space for $fp,$ra,$a0 in stack
38  sw $ra,4($sp) #save return address

```

mips1.asm

mips2.asm

mips3.asm

mips4.asm

mips5.asm

```

39  sw $a0,0($sp) #save $a0 register
40
41  slti $t0,$a0,2 #if input argument N < 2
42  beq $t0,$zero,recursive #if it is false ((a0 = N) >=2)
43  nop
44  li $v0,1 #return the result N!=1
45  j done
46  nop
47  recursive:
48  addi $a0,$a0,-1 #adjust input argument
49  jal FACT #recursive call
50  nop
51  lw $v1,0($sp) #load a0
52  mult $v1,$v0 #compute the result
53  mflo $v0
54  done: lw $ra,4($sp) #restore return address
55  lw $a0,0($sp) #restore a0
56  addi $sp,$fp,0 #restore stack pointer
57  lw $fp,-4($sp) #restore frame pointer
58  jr $ra #jump to calling
59  fact_end:

```

- Edit**   **Execute**

☐ **Text Segment**

Program Arguments:

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x0c100008	jnl 0x00400020	5: main: jnl WARP
<input type="checkbox"/>	0x00400004	0x00402820	add \$5,\$2,\$0	7: print: add \$a1,\$v0,\$zero # \$a0 = result from N!
<input type="checkbox"/>	0x00400008	0x24020038	addiu \$2,\$0,\$6	8: li \$v0, \$6
<input type="checkbox"/>	0x0040000c	0xc01e0011	lui \$1,4097	9: la \$a0, Message
<input type="checkbox"/>	0x00400010	0x3a240000	ori \$4,\$1,0	
<input type="checkbox"/>	0x00400014	0x0000000c	syscall	10: syscall
<input type="checkbox"/>	0x00400018	0x2402000a	addiu \$2,\$0,\$10	11: quit: li \$v0, 10 #terminate
<input type="checkbox"/>	0x0040001c	0x0000000c	syscall	12: syscall
<input type="checkbox"/>	0x00400020	0x0000fffc	sw \$fp,-4(\$gp) #save frame pointer (1)	17: WARP: sw \$fp,-4(\$gp) #save frame pointer (1)
<input type="checkbox"/>	0x00400024	0x2bbe0000	addi \$30,\$29,0	18: addi \$fp,\$sp,0 #new frame pointer

☒ **Data Segment**

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	544499019	543257969	1752066420	1634297632
0x10010020	0	0	0	0
0x10010040	0	0	0	0
0x10010060	0	0	0	0
0x10010080	0	0	0	0
0x100100a0	0	0	0	0
0x100100c0	0	0	0	0
0x100100e0	0	0	0	0
0x10010100	0	0	0	0
0x10010120	0	0	0	0

☒ **Labels**

Label	Address
mips4.asm	0x00400000
main	0x00400000
print	0x00400004
quit	0x00400018
endmain	0x00400020
WARP	0x00400020
wrap_end	0x0040004c
FACT	0x0040004c
recursive	0x00400078
done	0x00400090
fact_end	0x00400094

☒ **Data**   ☒ **Text**

Ket qua tinh giai thua la: 40320

Mars Messages   Run IO

-- program is finished running --

\$sp	Con trỏ stack pointer
\$fp	Lưu con trỏ đến khung trang frame pointer
\$a0	Kết quả n!
\$v0	Lưu số lớn nhất, đồng thời sử dụng cho syscall
\$ra	Thanh ghi chứa địa chỉ return
\$t0	Thanh ghi chứa biến so sánh làm điều kiện kết thúc
\$v1	Thanh ghi chứa giá trị lấy ra từ stack
\$a1	Lưu kết quả cuối cùng sử dụng cho hiển thị của lệnh syscall 56

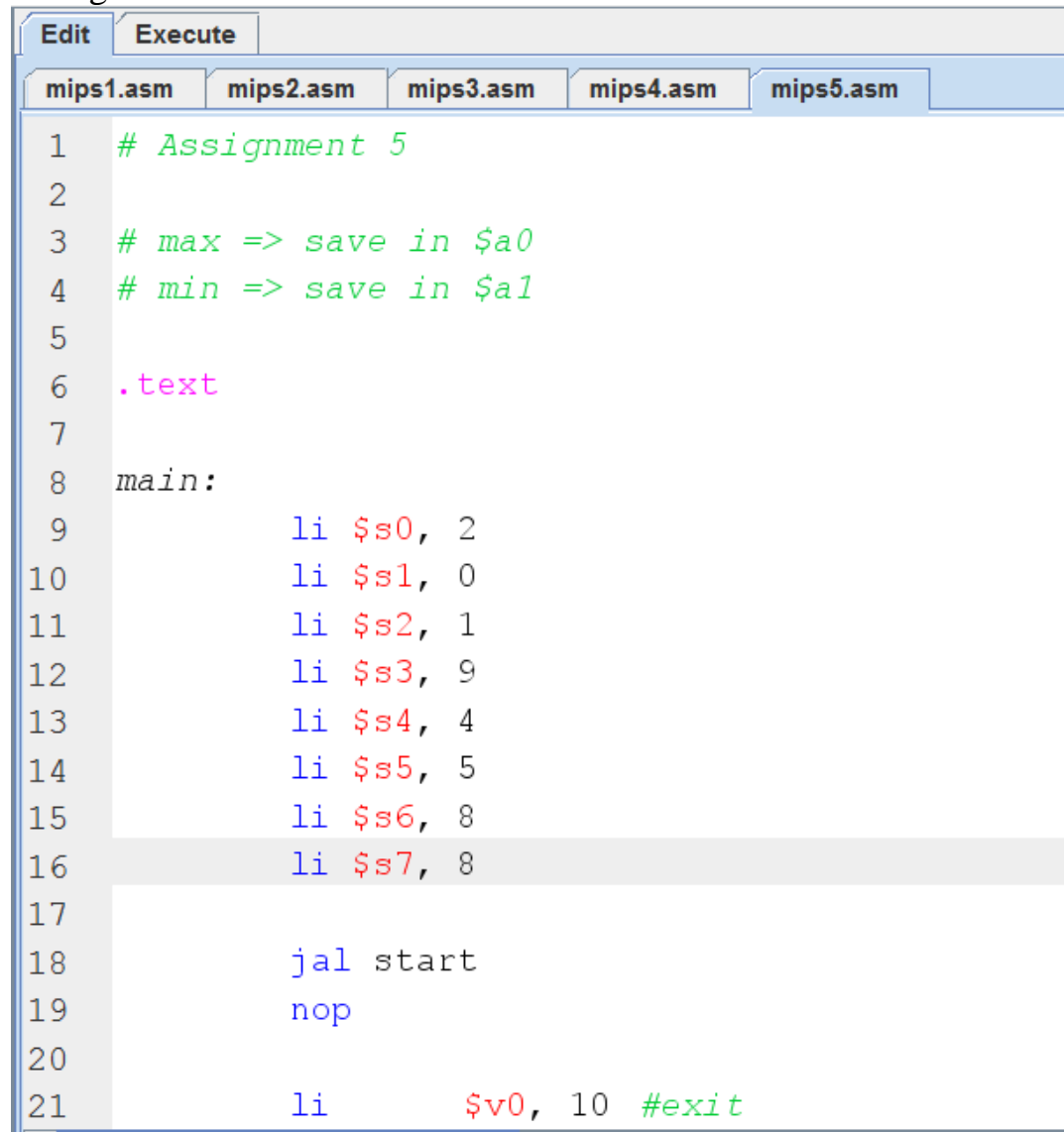
- Giải thích:
  - Dòng 3: Khai báo massage
  - Dòng 5: Nhảy đến hàm thủ tục WARP
  - WARP (Dòng 17-28):
    - Lưu con trỏ khung cũ
    - Tạo con trỏ khung mới \$fp=\$sp
    - Khai báo con trỏ stack mới
    - Lưu địa chỉ trả về \$ra vào \$sp[0]
    - Lưu giá trị input \$a0=8
    - Gọi đến thủ tục FACT
  - FACT (Dòng 35-46):
 

Lặp lại các thao tác:

    - Lưu con trỏ khung cũ
    - Tạo con trỏ khung mới \$fp=\$sp
    - Khai báo stack mới chứa \$a0, \$ra, \$fp
    - Khi \$a0>=2, thực hiện recursive : \$a0=\$a0-1
    - Khi \$a0=1, \$v0=1 và nhảy đến nhãn done
  - Done (Dòng 54-58): lặp lại việc lấy các giá trị \$ra, \$a0, \$sp, \$fp đã lưu, thực hiện tính  $n!=1.2....n$

## Assignment 5:

- Mã nguồn:



```
1  # Assignment 5
2
3  # max => save in $a0
4  # min => save in $a1
5
6  .text
7
8  main:
9      li $s0, 2
10     li $s1, 0
11     li $s2, 1
12     li $s3, 9
13     li $s4, 4
14     li $s5, 5
15     li $s6, 8
16     li $s7, 8
17
18     jal start
19     nop
20
21     li $v0, 10 #exit
```

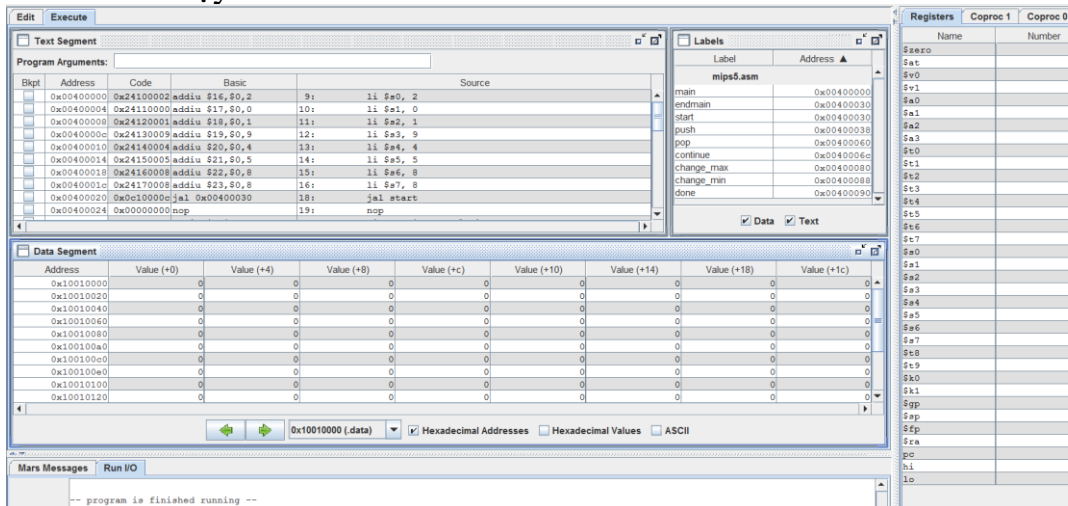
```

22         syscall
23
24     endmain:
25
26     start:
27         add $a0, $zero, $s0        # max
28         add $a1, $zero, $s0        # min
29
30     push:
31         add $t7, $sp, $zero        # luu gia tri sp de ke
32         addi $sp, $sp, -32         # push cac gia tri s0
33         sw $s0, 28($sp)
34         sw $s1, 24($sp)
35         sw $s2, 20($sp)
36         sw $s3, 16($sp)
37         sw $s4, 12($sp)
38         sw $s5, 8($sp)
39         sw $s6, 4($sp)
40         sw $s7, 0($sp)
41
42     pop:
43         lw $t0, 0($sp)
44         sub $t1, $t0, $a0          # if $t0 - $a0 > 0 => $t0 > $a0 => Max moi
45         bgez $t1, change_max
46     continue:
47         sub $t2, $t0, $a1          # if $t0 - $a1 < 0 => $t0 < $a1 => Min moi la
48         bltz $t2, change_min
49         beq $t7, $sp, done         # Neu $sp = gia tri $sp ban dau thi ket thu
50         addi $sp, $sp, 4
51         j     pop
52
53     change_max:
54         add $a0, $zero, $t0
55         j     continue
56
57     change_min:
58         add $a1, $zero, $t0
59         j     pop
60
61     done:
62         jr $ra

```



- Màn hình chạy:



- Giải thích:

- Dòng 8-16: Khai báo đầu vào
- Dòng 18: Đến thủ tục Start
- Thủ tục Start (Dòng 26-29): Khai báo max, min vào \$a0, \$a1
- Thủ tục push: Dòng 30-40:
  - Dòng 31-32: Lưu giá trị sp để kết thúc
  - Dòng 33-40: Lăn lượt push từng phần tử bằng lệnh sw
- Thủ tục pop: Dòng 42-45: Nếu  $t0 > a0$  thì đến thủ tục change\_max
- Thủ tục change\_max:  $a0 \text{ (max)} = t0$
- Thủ tục continue (Dòng 46-51):
  - Dòng 47,48: Nếu  $t0 < a1$  thì đến thủ tục change\_min
  - Dòng 49: Kiểm tra xem  $sp =$  giá trị \$sp ban đầu  $\Rightarrow$  done
  - Dòng 50: Tăng sp lên 4
  - Dòng 51: Nhảy đến pop
- Thủ tục chane\_min (Dòng 57-58):  $a1 = t0$
- Thủ tục done: nhảy đến địa chỉ lưu trong thanh ghi (\$ra)