

BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH

LAB 7

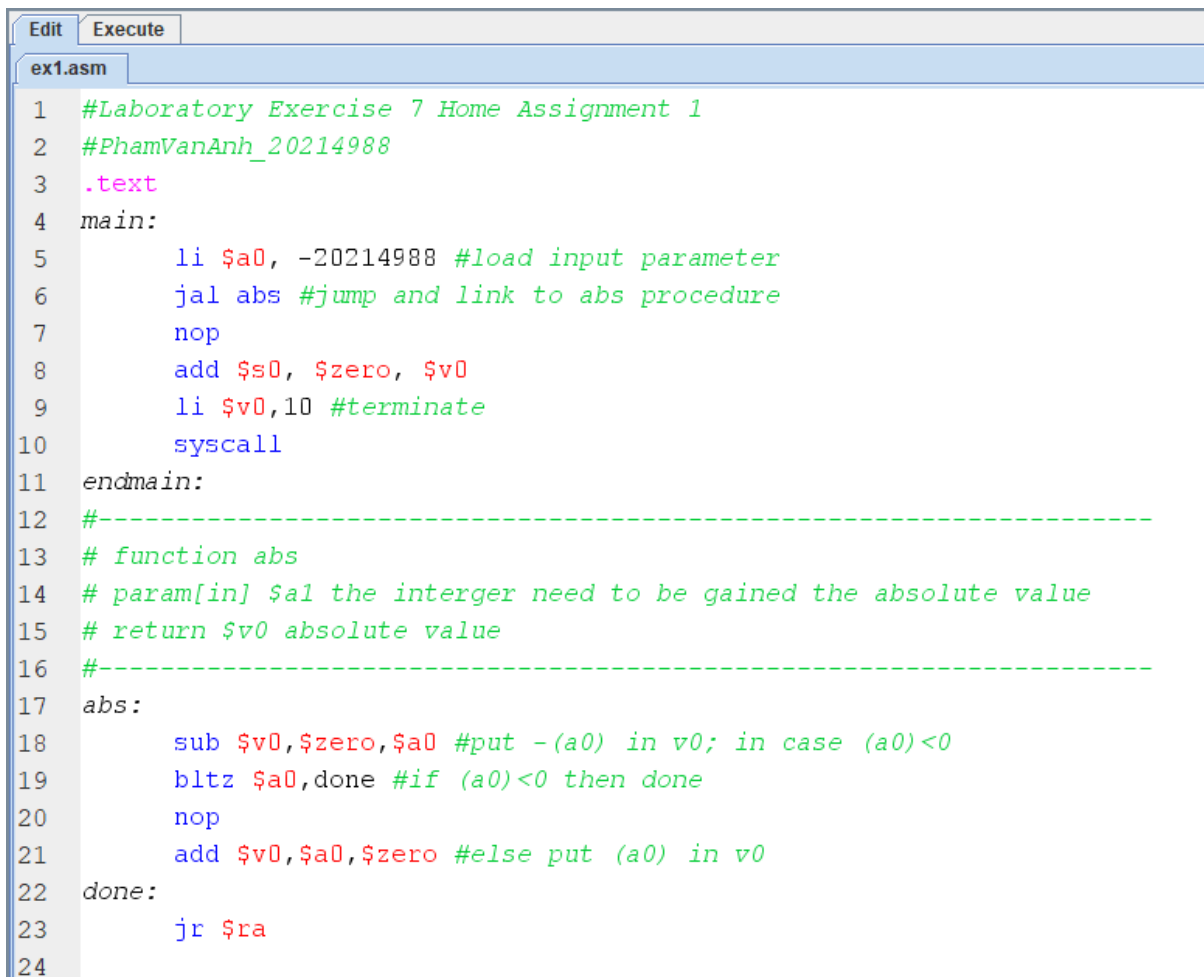
Họ và tên: Phạm Văn Anh

MSSV: 20214988

Mã lớp: 139365

ASSIGNMENT 1:

1. Code



```
1  #Laboratory Exercise 7 Home Assignment 1
2  #PhamVanAnh_20214988
3  .text
4  main:
5      li $a0, -20214988 #load input parameter
6      jal abs #jump and link to abs procedure
7      nop
8      add $s0, $zero, $v0
9      li $v0, 10 #terminate
10     syscall
11 endmain:
12 #-----
13 # function abs
14 # param[in] $a1 the interger need to be gained the absolute value
15 # return $v0 absolute value
16 #-----
17 abs:
18     sub $v0, $zero, $a0 #put -(a0) in v0; in case (a0)<0
19     bltz $a0, done #if (a0)<0 then done
20     nop
21     add $v0, $a0, $zero #else put (a0) in v0
22 done:
23     jr $ra
24
```

Thực hiện gõ chương trình vào công cụ MARS

2. Giải thích code

- Gán \$a0 = -20214988 và kết quả trị tuyệt đối của nó được lưu vào thanh ghi \$v0
- Dòng 6: jal abs (nhảy đến thủ tục có nhãn abs)
 - Lưu thanh ghi \$ra = pc = 0x004000c (địa chỉ lệnh tiếp theo là lệnh nop trong main)
 - pc thay đổi từ 0x0040008 (địa chỉ jal hiện tại) thành địa chỉ đầu tiên của abs 0x0040001c
- Sau khi thực hiện xong abs, lưu kết quả vào \$v0
- Thoát abs bằng lệnh jr \$ra → lưu thanh ghi pc = \$ra để quay lại main. pc thay đổi từ 0x0040002x (địa chỉ của jr hiện tại) thành 0x004000c (\$ra)

3. Kết quả

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0x00000000
\$at	1	0xfecb0000
\$v0	2	0x0000000a
\$v1	3	0x00000000
\$a0	4	0xfecb8b34
\$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	0x013474cc
\$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	0x0040000c
pc		0x0040001c
hi		0x00000000
lo		0x00000000

ASSIGNMENT 2:

1. Code

```

1  #Laboratory Exercise 7, Home Assignment 2
2  #PhamVanAnh_20214988
3  .text
4  main:
5      li    $a0, 5      #load test input
6      li    $a1, 4
7      li    $a2, 7
8      jal   max         #call max procedure
9      nop
10     li $v0, 10 #terminate
11     syscall
12 endmain:
13 #-----
14 #Procedure max: find the largest of three integers
15 #param[in] $a0 integers
16 #param[in] $a1 integers
17 #param[in] $a2 integers
18 #return $v0 the largest value
19 #-----
20 max:
21     add    $v1, $a0, $zero    #copy (a0) in v1; largest so far
22     sub     $t0, $a1, $v1     #compute (a1)-(v1)
23     bltz    $t0, okay        #if (a1)-(v1)<0 then no change
24     nop
25     add     $v1, $a1, $zero    #else (a1) is largest thus far
26 okay:
27     sub     $t0, $a2, $v1     #compute (a2)-(v1)
28     bltz    $t0, done        #if (a2)-(v1)<0 then no change
29     nop
30     add     $v1, $a2, $zero    #else (a2) is largest overall
31 done:
32     jr      $ra              #return to calling program
33
```

Thực hiện gõ chương trình vào công cụ MARS

2. Giải thích

- Gán giá trị $\$a0 = 5$, $\$a1 = 4$, $\$a2 = 7$. Tìm max trong 3 số và lưu vào thanh ghi $\$v1$
- Dòng 8: jal max (nhảy đến thủ tục có nhãn max)
 - Thanh ghi $\$ra = pc$ nên $\$ra = 0x0040000c$ (giá trị hiện tại của pc) và thay đổi thành $0x0040001c$ là địa chỉ của nhãn max
 - Thủ tục *max*:
 - Dòng 19: gán giá trị max tạm thời vào $\$v1 = \$a0 = 5$
 - Dòng 20: gán $\$t0 = \$a1 - \$v1 = 4 - 5 = -1$
 - Dòng 21: so sánh $\$t0$ và 0
 - + Nếu $\$t0 > 0 \rightarrow \$a1 > \$v1 = \$a0 \rightarrow$ cập nhật $\text{max} = \$v1 = \$a1$
 - + Nếu $\$t0 < 0 \rightarrow \$a1 < \$v1 = \$a0 = 5 \rightarrow$ okay
 - Dòng 24: okay
 - + Xét $\$t0 = \$a2 - \$a0$
 - + So sánh $\$t0$ và 0
 - Nếu $\$t0 > 0 \rightarrow$ cập nhật $\text{max} = \$a2$
 - Nếu $\$t0 < 0 \rightarrow \text{max} = \$a0 \rightarrow$ done
 - Dòng 29: *done*: jr $\$ra \rightarrow pc = \ra , để kết thúc trở về main nên pc đổi từ $0x00400040$ (địa chỉ jr hiện tại) thành $0x00400010$ (địa chỉ tiếp theo tức là lệnh nop)

3. Kết quả

- Gán giá trị:

$\$a0$	4	0x00000005
$\$a1$	5	0x00000004
$\$a2$	6	0x00000007

- $\text{max} = \$v1$

$\$v1$	3	0x00000007
--------	---	------------

- Địa chỉ pc và $\$ra$

$\$ra$	31	0x00400010
pc		0x0040001c

ASSIGNMENT 3:

1. Code

```
ex3.asm*
1  #Laboratory Exercise 7, Home Assignment 3
2  #PhamVanAnh_20214988
3  .text
4  push:
5      li    $s0, 10
6      li    $s1, 12
7      addi  $sp, $sp, -8 #adjust the stack pointer
8      sw    $s0, 4($sp) #push $s0 to stack
9      sw    $s1, 0($sp) #push $s1 to stack
10 work:
11     nop
12     nop
13     nop
14 pop:
15     lw    $s0, 0($sp) #pop from stack to $s0
16     lw    $s1, 4($sp) #pop from stack to $s1
17     addi  $sp, $sp, 8 #adjust the stack pointer
18
```

Thực hiện gõ chương trình vào công cụ MARS

2. Giải thích

- Tạo ngăn xếp để push và pop 2 tham số 10,12 được lưu vào thanh ghi \$s0, \$s1
- Hàm push:
 - Dòng 7: thanh ghi \$sp lùi 8 byte để đủ chỗ cho 2 phần tử 10, 12 của stack vì stack có đáy ở dưới và chiều địa chỉ là từ trên xuống → trừ đi 8.
 - \$sp chứa địa chỉ của các phần tử trên cùng của stack
→ sp (ban đầu) = 0x7ffeffc thay đổi thành \$sp = 0x7ffeff4
 - Dòng 8: sw để push lần lượt tham số vào thanh ghi

ASSIGNMENT 4:

1. Code

ex1.asm	ex2.asm	ex3.asm	ex4.asm	mips2.asm
---------	---------	---------	---------	-----------

```
1  #Laboratory Exercise 7 Home Assignment 4
2  #PhamVanAnh_20214988
3  .data
4      Message: .asciiz "Ket qua tinh giai thua la: "
5  .text
6  main:
7      jal    WARP
8  print:
9      add    $a1, $v0, $zero # $a0 = result from N!
10     li     $v0, 56
11     la     $a0, Message
12     syscall
13  quit:
14     li     $v0, 10          #terminate
15     syscall
16  endmain:
17  #-----
18  #Procedure WARP: assign value and call FACT
19  #-----
20  WARP:
21     sw     $fp, -4($sp)     #save frame pointer (1)
22     addi   $fp, $sp, 0      #new frame pointer point to the top (2)
23     addi   $sp, $sp, -8     #adjust stack pointer (3)
24     sw     $ra, 0($sp)      #save return address (4)
25     li     $a0, 6           #load test input N = 8
26     jal    FACT             #call fact procedure
27     nop
28     lw     $ra, 0($sp)      #restore return address (5)
29     addi   $sp, $fp, 0      #return stack pointer (6)
30     lw     $fp, -4($sp)     #return frame pointer (7)
31     jr     $ra
32  wrap_end:
```

```

33 #-----
34 #Procedure FACT: compute N!
35 #param[in] $a0 integer N
36 #return $v0 the largest value
37 #-----
38 FACT:
39     sw    $fp, -4($sp)          #save frame pointer
40     addi  $fp, $sp, 0           #new frame pointer point to stack's top
41     addi  $sp, $sp, -12         #allocate space for $fp,$ra,$a0 in stack
42     sw    $ra, 4($sp)           #save return address
43     sw    $a0, 0($sp)           #save $a0 register
44     slti  $t0, $a0, 2           #if input argument N < 2
45     beq   $t0, $zero, recursive #if it is false ((a0 = N) >=2)
46     nop
47     li    $v0, 1                #return the result N!=1
48     j     done
49     nop
50 recursive:
51     addi  $a0, $a0, -1          #adjust input argument
52     jal   FACT                 #recursive call
53     nop
54     lw    $v1, 0($sp)          #load a0
55     mult  $v1, $v0              #compute the result
56     mflo  $v0
57 done:
58     lw    $ra, 4($sp)          #restore return address
59     lw    $a0, 0($sp)          #restore a0
60     addi  $sp, $fp, 0          #restore stack pointer
61     lw    $fp, -4($sp)         #restore frame pointer
62     jr    $ra                  #jump to calling
63 fact_end:

```

Thực hiện gõ chương trình vào công cụ MARS

2. Giải thích code

- Thanh ghi \$fp lưu con trỏ tới frame pointer, \$sp – stack pointer
- Thanh ghi \$a0 lưu kết quả tính giai thừa
- Hàm WARP:
 - Khai báo 2 vị trí trong stack cho frame pointer \$fp và địa chỉ return \$ra → Giá trị thanh ghi \$fp = 0x7ffffefc; lưu địa chỉ \$ra ra bộ nhớ ở địa chỉ 0(\$sp) bằng lệnh sw

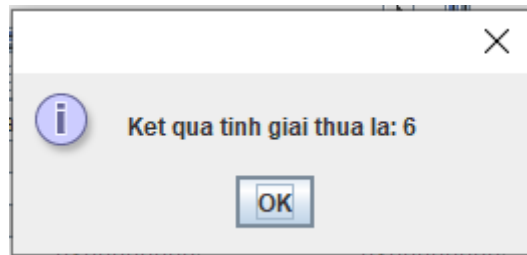
\$fp	30	0x7ffffefc
------	----	------------

- \$fp = \$sp
- Dòng 23: tạo ra 2 ngăn nhớ của stack
- Gán giá trị của \$ra vào ngăn \$sp trỏ vào
- Lưu N vào \$a0 → \$a0 = 0x00000003

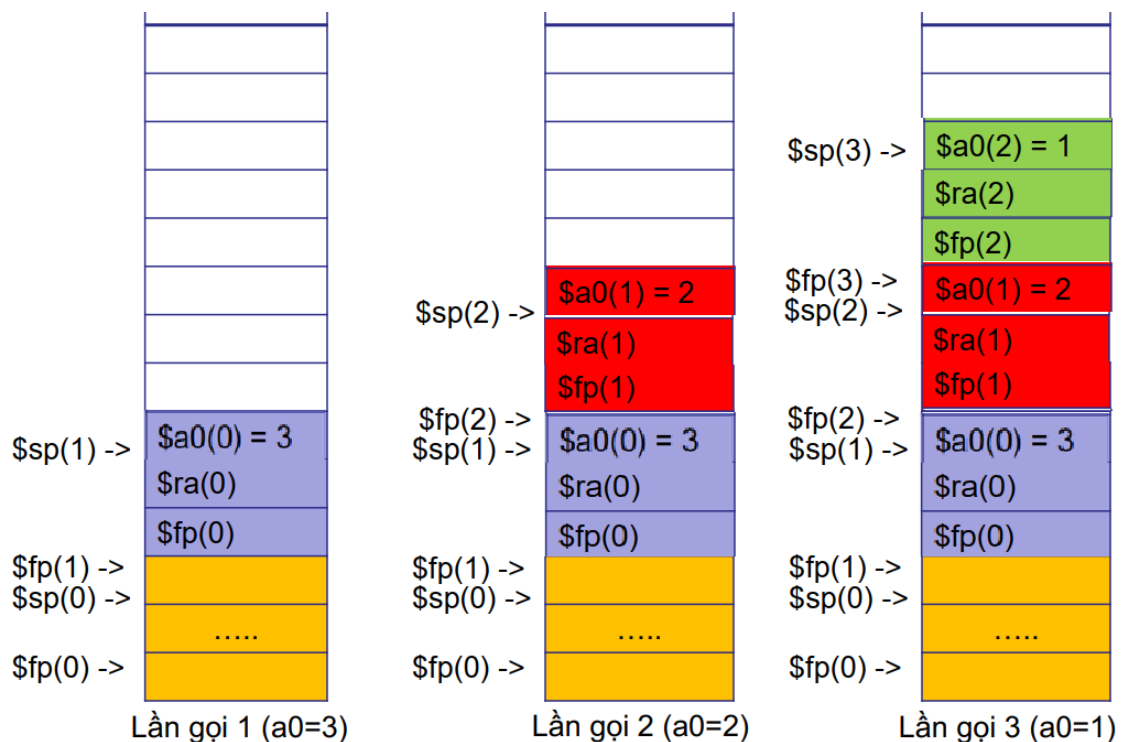
\$a0	4	0x00000003
------	---	------------

- Restore giá trị từ 0(\$sp) vào lạ \$ra để gọi lệnh jr \$ra trở về địa chỉ chính xác của lệnh tiếp theo trong main.
- Hàm FACT:
 - Khai báo 3 vị trí trong stack để trả về các giá trị \$fp, \$ra và \$a0
 - \$sp -12 để tạo ra 3 vị trí
 - \$sp = 0x7ffeff4 thành 0x7ffefe8, lưu lần lượt giá trị vào stack.
 - So sánh tham số đưa vào với 2
 - Nếu <2 → kết quả = 1
 - Nếu >2 → hàm RECURSIVE
 - Nhảy đến done để Restore lại các giá trị return vào thanh ghi \$ra để lệnh jr \$ra được thực thi chính xác
- Hàm RECURSIVE:
 - Giảm a0 đi 1 đơn vị và tiếp tục quay lại load đưa vào stack

3. Kết quả



- Bảng stack



ASSIGNMENT 5:

1. Code

```
ex4.asm  ex5.asm
1  #Laboratory Exercise 7 Home Assignment 5
2  .data
3      message_max:      .ascii "Largest: "
4      message_index:    .ascii ", "
5      message_min:      .ascii "Smallest: "
6      enter:            .ascii "\n"
7  .text
8  load_data:
9      li $s0, 2
10     li $s1, 0
11     li $s2, 2
12     li $s3, 1
13     li $s4, -4
14     li $s5, 9
15     li $s6, 8
16     li $s7, -8
17
18     li $t3, 8 # bien dem nguoc lai vi push vao stack thu tu bi dao nguoc
19     li $t4, 8 # bien dem nguoc lai vi push vao stack thu tu bi dao nguoc
20 main:
21     jal push
22     nop
23     jal max
24     nop
25     jal min
26     nop
27
28 print_max:
29     li $v0, 4          #code for print to console
30     la $a0, message_max
31     syscall
32     li $v0, 1
33     add $a0, $zero, $a1 # $a1 = max
34     syscall
35     li $v0, 4
36     la $a0, message_index
37     syscall
38     li $v0, 1
39     add $a0, $zero, $t1 # $t1 -> register containing the max value
40     syscall
41     li $v0, 4
42     la $a0, enter
43     syscall
```

```

43 print_min:
44     li    $v0, 4
45     la    $a0, message_min
46     syscall
47     li    $v0, 1
48     add   $a0, $zero, $a2    #$t3 = min
49     syscall
50     li    $v0, 4
51     la    $a0, message_index
52     syscall
53     li    $v0, 1
54     add   $a0, $0, $t2      #$t4 -> register containing the min value
55     syscall
56 exit:
57     li    $v0, 10
58     syscall
59 end_main:

```

```

60 #-----
61 push:
62     addi  $sp, $sp, -32    #stack chua 8 ptu
63     sw    $s0, 28($sp)
64     sw    $s1, 24($sp)
65     sw    $s2, 20($sp)
66     sw    $s3, 16($sp)
67     sw    $s4, 12($sp)
68     sw    $s5, 8($sp)
69     sw    $s6, 4($sp)
70     sw    $s7, 0($sp)
71 push_end:
72     jr    $ra
73 #-----
74 max:
75     lw    $v1, 0($sp)      #pop phan tu dau tien vao $v1
76     addi  $sp, $sp, 4
77     addi  $t3, $t3, -1     #index -1, giam sau moi lan lap
78
79     beq   $t3, 0, max_end  #kiem tra xem da den phan tu cuoi hay chua
80     slt   $t0, $v1, $a1    #$a1 luu gia tri max hien tai
81     bne   $t0, $zero, max
82     add   $t1, $zero, $t3  #luu index cua max
83     add   $a1, $zero, $v1  #luu max vao tu $v1 sang $t5
84     j     max
85 max_end:
86     add   $sp, $sp, -32    #adjust stack pointer to the top of stack
87     jr    $ra
88 #-----

```

```

88 #-----
89 min:
90     lw     $v1, 0($sp)      #pop tu stack
91     addi   $sp, $sp, 4
92     addi   $t4, $t4, -1     #$t4 -> index = 1, tang sau moi lan lap
93
94     beq    $t4, 0, min_end  # khi stack rong -> min_end
95     slt    $t0, $a2, $v1    #so sanh $1 dang chua min hien tai voi $t0 la ptu hien tai
96     bne    $t0, $zero, min
97     add    $t2, $zero, $t4  #luu index cua min
98     add    $a2, $zero, $v1  #luu gia tri min
99     j      min
100 min_end:
101     add    $sp, $sp, -32    #adjust stack pointer to the top of stack
102     jr     $ra

```

2. Giải thích

- Chương trình tìm max, min bằng cách đặt max = 1 giá trị cho trước và dịch chuyển \$sp để pop lần lượt ra các phần tử ở đầu stack và đem so sánh với max và min, nếu thay đổi thì cập nhật max & min luôn.
- Chương trình tìm max, min và vị trí thành ghi lưu trữ của nó, được lưu từ \$s0 → \$s7.

\$s0	16	2
\$s1	17	0
\$s2	18	2
\$s3	19	1
\$s4	20	-4
\$s5	21	9
\$s6	22	8
\$s7	23	-8

- Thanh ghi \$a1, \$a2 lưu max và min

\$a1	5	9
\$a2	6	-8

- Thanh ghi \$t1, \$t2 lưu index max, min tương ứng

\$t1	9	5
\$t2	10	7

- Thanh ghi \$t3, \$t4 lưu giá trị biến index để đếm sau các vòng lặp tìm max, min.
- Hàm main:

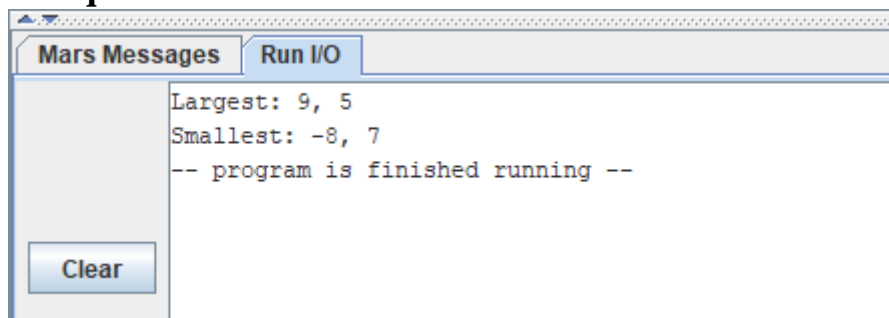
- Nhảy đến push để lưu các giá trị dãy số vào lần lượt vào stack
- Nhảy đến max để tìm max
- Nhảy đến min để tìm min

- Hàm MAX:

- Pop từ stack và lưu vào \$v1 nên giá trị sẽ thay đổi trở thành phần tử được pop ra
- \$sp tăng lên 4 để trở đến phần tử tiếp theo của stack
- Giảm biến đếm thành ghi \$t3

- Kiểm tra xem có kết thúc dãy chưa, nếu $\$t3 = 0 \rightarrow$ kết thúc
- So sánh $\$v1$ (vừa được pop ra) và $\$a0$ (max hiện tại) để xem cập nhật hay tiếp tục vòng lặp
- Sau khi hàm end thì đưa stack về dãy ban đầu
- Hàm MIN:
 - Hoạt động tương tự max
 - Giá trị min lưu vào thanh ghi $\$a2$
 - Index lưu vào thanh ghi $\$t4$
 - Vì các thủ tục không lồng nhau nên thanh ghi $\$ra$ không phải lưu lại ra bộ nhớ để restore nên nó thay đổi theo địa chỉ con trỏ pc.

3. Kết quả:



Registers	Coproc 1	Coproc 0	
Name	Number	Value	
\$zero	0	0x00000000	
\$at	1	0x10010000	
\$v0	2	0x0000000a	
\$v1	3	0x00000002	
\$a0	4	0x00000007	
\$a1	5	0x00000009	
\$a2	6	0xffffffff8	
\$a3	7	0x00000000	
\$t0	8	0x00000001	
\$t1	9	0x00000005	
\$t2	10	0x00000007	
\$t3	11	0x00000000	
\$t4	12	0x00000000	
\$t5	13	0x00000000	
\$t6	14	0x00000000	
\$t7	15	0x00000000	
\$s0	16	0x00000002	
\$s1	17	0x00000000	
\$s2	18	0x00000002	
\$s3	19	0x00000001	
\$s4	20	0xffffffffc	
\$s5	21	0x00000009	
\$s6	22	0x00000008	
\$s7	23	0xffffffff8	
\$t8	24	0x00000000	
\$t9	25	0x00000000	
\$k0	26	0x00000000	
\$k1	27	0x00000000	
\$gp	28	0x10008000	
\$sp	29	0x7ffefdc	
\$fp	30	0x00000000	
\$ra	31	0x0040003c	
pc		0x004000c8	
hi		0x00000000	
lo		0x00000000	