

THỰC HÀNH 4

1) Sự thay đổi của các thanh ghi trong code hình 1 khi chạy từng lệnh:

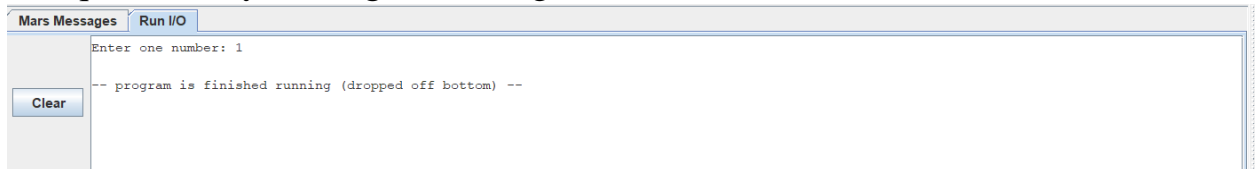
Thanh ghi Tên lệnh	PC	\$ra	\$sp	\$fp
main: jal getInt	0x0040000c	0x00400004	0x7ffefffc	0x00000000
getInt: li \$v0, 4	0x00400010	0x00400004	0x7ffefffc	0x00000000
la \$a0, prompt	0x00400014	0x00400004	0x7ffefffc	0x00000000
syscall	0x0040001c	0x00400004	0x7ffefffc	0x00000000
li \$v0, 5	0x00400020	0x00400004	0x7ffefffc	0x00000000
syscall	0x00400020	0x00400004	0x7ffefffc	0x00000000
jr \$ra	0x00400004	0x00400004	0x7ffefffc	0x00000000
move \$s0, \$v0	0x00400008	0x00400004	0x7ffefffc	0x00000000
j exit	0x00400028	0x00400004	0x7ffefffc	0x00000000

Sự thay đổi của các thanh ghi trong code hình 2 khi chạy từng lệnh:

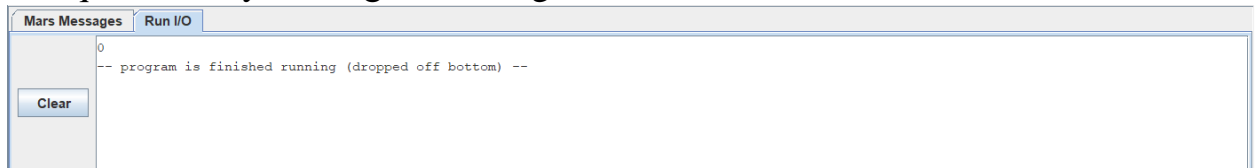
Thanh ghi Tên lệnh	PC	\$ra	\$sp	\$fp
move \$a0, \$s0	0x00400004	0x00000000	0x7ffefffc	0x00000000
move \$a1, \$s1	0x00400008	0x00000000	0x7ffefffc	0x00000000
move \$a2, \$s2	0x0040000c	0x00000000	0x7ffefffc	0x00000000
move \$a3, \$s3	0x00400010	0x00000000	0x7ffefffc	0x00000000
jal proc_example	0x00400024	0x00400014	0x7ffefffc	0x00000000
addi \$sp, \$sp, -4	0x00400028	0x00400014	0x7ffefff8	0x00000000
sw \$s0, 0(\$sp)	0x0040002c	0x00400014	0x7ffefff8	0x00000000
add \$t0, \$a0, \$a1	0x00400030	0x00400014	0x7ffefff8	0x00000000
add \$t1, \$a2, \$a3	0x00400034	0x00400014	0x7ffefff8	0x00000000
sub \$s0, \$t0, \$t1	0x00400038	0x00400014	0x7ffefff8	0x00000000
move \$v0, \$s0	0x0040003c	0x00400014	0x7ffefff8	0x00000000
lw \$s0, 0(\$sp)	0x00400040	0x00400014	0x7ffefff8	0x00000000
addi \$sp, \$sp, 4	0x00400044	0x00400014	0x7ffefffc	0x00000000
jr \$ra	0x00400014	0x00400014	0x7ffefffc	0x00000000

move \$a0, \$v0	0x00400018	0x00400014	0x7ffefffc	0x00000000
li \$v0, 1	0x0040001c	0x00400014	0x7ffefffc	0x00000000
syscall	0x00400020	0x00400014	0x7ffefffc	0x00000000
j exit	0x00400048	0x00400014	0x7ffefffc	0x00000000

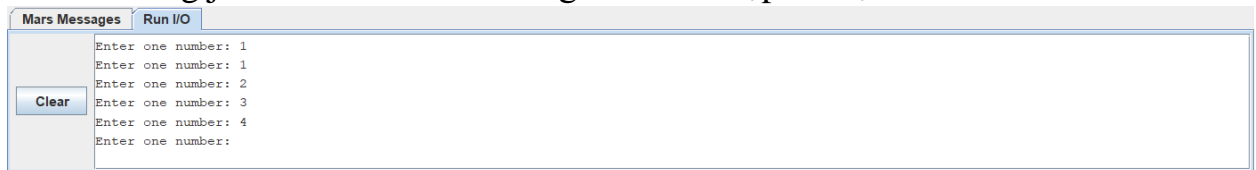
Kết quả khi chạy chương trình trong 1 lần của hình 1:



Kết quả khi chạy chương trình trong 1 lần của hình 2:

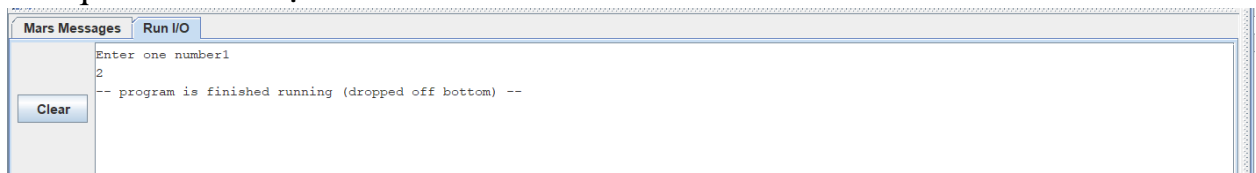


Nếu bỏ dòng j exit ở hình 1 thì chương trình sẽ nhập vô hạn lần



Mô phỏng thêm thủ tục ShowInt : [File code](#)

Kết quả thêm thủ tục ShowInt:



Giải thích: Đầu tiên nhóm sẽ lưu tổng của số input với 1 vào \$a0 để truyền cho thủ tục và gọi thủ tục ShowInt

7	addi \$a0, \$s0, 1
8	jal showInt

Trong thủ tục ShowInt nhóm sẽ cho \$v0 bằng 1 để có thể xuất số trong \$a0 ra màn hình khi gọi syscall. Và sau khi gọi syscall thì nhóm sẽ nhảy về lệnh sau lệnh gọi thủ tục ShowInt ở nhãn main

```

17  showInt:          li $v0, 1
18                      syscall
19                      jr $ra

```

Rồi nhóm nhảy tới nhãn exit

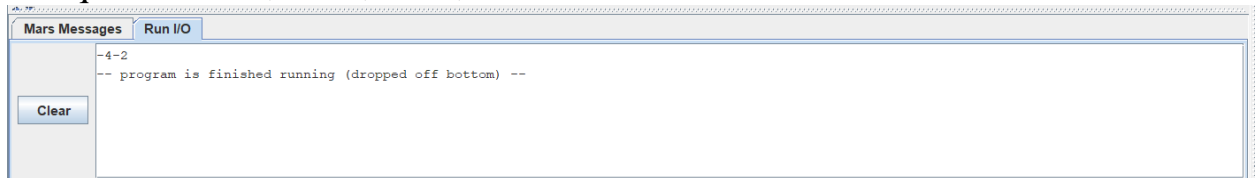
```

8      jal showInt
9      j  exit

```

Mô phỏng code trong hình 2 sau khi chỉnh sửa: [File code](#)

Kết quả với a = 1, b = 2, c = 3, d = 4:



Giải thích: Đầu tiên nhóm sẽ thêm 1 dòng lưu giá trị \$s1 xuống stack vào trong nhãn proc_example

```

17      addi $sp, $sp, -4
18      sw  $s0, 0($sp)
19      addi $sp, $sp, -4
20      sw  $s1, 0($sp)

```

Do trong code ban đầu các biến \$ax được truyền vào vẫn chưa bị thay đổi giá trị nên nhóm sẽ tái sử dụng lại bằng cách lấy \$a0 trừ \$a1 và \$a2 trừ \$a3

```

sub $t0, $a0, $a1
sub $t1, $a2, $a3

```

Và nhóm cộng 2 giá trị \$t0 và \$t1 lại, lưu vào \$s1 và cho \$v1 bằng \$s1

30	add \$s1, \$t0, \$t1
31	
32	move \$v1, \$s1

Rồi nhóm trả giá trị ban đầu về cho \$s0 và \$s1 rồi nhảy về lệnh sau lệnh gọi thủ tục

34	lw \$s1, 0(\$sp)
35	addi \$sp, \$sp, 4
36	lw \$s0, 0(\$sp)
37	addi \$sp, \$sp, 4
38	
39	jr \$ra

Tại đó nhóm thêm 4 lệnh để xuất ra màn hình kết quả giống như code ban đầu nhưng chỉ thay đổi số \$v0 thành \$v1

7	move \$a0, \$v0
8	li \$v0, 1
9	syscall
10	
11	move \$a0, \$v1
12	li \$v0, 1
13	syscall

Rồi chương trình nhảy tới nhãn exit và kết thúc

14	exit
----	------

Mô phỏng code trong hình 2 sau khi chỉnh sửa lần nữa: [File code](#)

Kết quả với $a = 1$, $b = 2$, $c = 3$, $d = 4$, $e = 5$, $f = 6$:

```
--4-1
-- program is finished running (dropped off bottom) --
```

Clear

Giải thích: Do bài toán cần truyền tới 6 biến nên nhóm sẽ lưu 2 hai biến e và f xuống stack ở nhãn main

1	<code>move \$a0, \$s0</code>
2	<code>move \$a1, \$s1</code>
3	<code>move \$a2, \$s2</code>
4	<code>move \$a3, \$s3</code>
5	<code>addi \$sp, \$sp, -4</code>
6	<code>sw \$s4, 0(\$sp)</code>
7	<code>addi \$sp, \$sp, -4</code>
8	<code>sw \$s5, 0(\$sp)</code>

Ở nhãn `proc_example` nhóm lấy giá trị e và f ở stack lưu vào thanh ghi \$t, do thanh ghi \$t3 và \$t4 chưa có giá trị nên nhóm sẽ không lưu giá trị cũ của nó vào trong stack

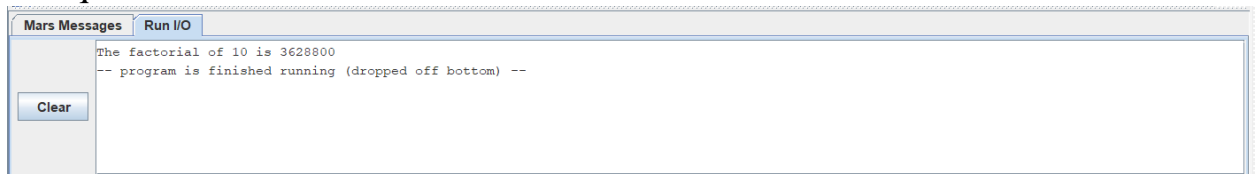
20	<code>proc_example:</code>
21	<code>lw \$t3, 0(\$sp)</code>
22	<code>addi \$sp, \$sp, 4</code>
23	<code>lw \$t2, 0(\$sp)</code>
24	<code>addi \$sp, \$sp, 4</code>

Để tính $e - f$ thì nhóm lấy \$t3 - \$t4 và lưu vào \$s1 rồi gán giá trị đó cho \$v1

37	<code>sub \$s1, \$t2, \$t3</code>
38	
39	<code>move \$v1, \$s1</code>

2) Mô phỏng: [File code](#)

Kết quả:



Giải thích:

Nhóm sẽ truyền thẳng số n vào \$a0. Sau đó nhảy đến label fact để tính toán n!

```
4  main:                addi $a0, $a0, 10
5                          jal fact
```

Ở label fact, nhóm sẽ so sánh \$a0 với 1 xem \$a0 có nhỏ hơn không. Nếu thỏa thì sẽ cho \$v0 bằng 1 và quay lại địa chỉ \$ra đang lưu. Nếu không thỏa thì sẽ nhảy tới label else

```
15  fact:
16      slti $t0, $a0, 1
17      beq $t0, $zero, else
18      addi $v0, $zero, 1
19      jr $ra
```

Ở label else (lệnh bne \$v0, \$zero, return bỏ nha thầy), nhóm sẽ lưu lần lượt giá trị \$ra, \$a0 xuống stack. Sau đó trừ \$a0 cho 1 rồi nhảy tới label fact để đệ quy

```

20  else:
21      bne $v0, $zero, return
22      addi $sp, $sp, -4
23      sw $ra, 0($sp)
24      addi $sp, $sp, -4
25      sw $a0, 0($sp)
26      addi $a0, $a0, -1
27      jal fact

```

Khi fact đã được gọi bởi label else $n - 1$ lần thì $n = 0$ ($\$a0 = 0$) dẫn đến điều kiện của if thỏa thì $\$v0$ sẽ bằng 1 và chương trình sẽ nhảy tới địa chỉ $\$ra$ tức là địa chỉ của label return. Ở label return, nhóm sẽ lấy giá trị của $\$a0$ và $\$ra$ (địa chỉ của return) đã lưu trong stack. Nhóm tính $\$v0 = \$v0 \times \$a0$ rồi nhảy tới label return lần nữa

```

28  return:
29      lw $a0, 0($sp)
30      addi $sp, $sp, 4
31      lw $ra, 0($sp)
32      addi $sp, $sp, 4
33      mult $a0, $v0
34      mflo $v0
35      jr $ra

```

Khi return đã được lặp lại $n - 1$ lần thì $\$ra$ được lấy ra từ trong stack sẽ là địa chỉ sau lệnh jal fact của label main, nên jr $\$ra$ sẽ đưa chương trình thoát khỏi đệ quy và lấy được kết quả 10!

Tại đây nhóm tiến hành in array và kết quả 10! ra màn hình I/O. Sau đó nhảy tới label exit

6	-	add \$s1, \$v0, \$zero
7		li \$v0, 4
8		la \$a0, array
9		syscall
10		li \$v0, 1
11		add \$a0, \$s1, \$zero
12		syscall
13		j exit

3)

1) Stack trong trường hợp 5!

\$fp	0x00000000	0x00000000

\$sp	0x7ffefd4	0x00000001
	0x7ffefd8	0x00400058
	0x7ffefdc	0x00000002
	0x7ffefe0	0x00400058
	0x7ffefe4	0x00000003
	0x7ffefe8	0x00400058
	0x7ffefec	0x00000004
	0x7ffeff0	0x00400058
	0x7ffeff4	0x00000005
	0x7ffeff8	0x00400008

Stack trong trường hợp 10!

\$fp	0x00000000	0x00000000

\$sp	0x7ffefac	0x00000001
	0x7ffefb0	0x00400058
	0x7ffefb4	0x00000002
	0x7ffefb8	0x00400058

0x7ffefbc	0x00000003
0x7ffefc0	0x00400058
0x7ffefc4	0x00000004
0x7ffefc8	0x00400058
0x7ffefcc	0x00000005
0x7ffefd0	0x00400058
0x7ffefd4	0x00000006
0x7ffefd8	0x00400058
0x7ffefdc	0x00000007
0x7ffefe0	0x00400058
0x7ffefe4	0x00000008
0x7ffefe8	0x00400058
0x7ffefec	0x00000009
0x7ffeff0	0x00400058
0x7ffeff4	0x0000000a
0x7ffeff8	0x00400008

2) Mô phỏng: [File code](#)

Kết quả:

```

Nhập n: 5
Chuoi Finonacci la: 0 1 1 2 3
-- program is finished running (dropped off bottom) --

```

Giải thích: đầu chương trình nhóm sẽ khai báo như này

```

1 | .data
2 | string1: .ascii "Chuoi Finonacci la: "
3 | string2: .ascii "Nhập n: "
4 | space: .ascii " "
5 | n: .word 0
6 | .text

```

Tiếp đến nhóm sẽ in ra màn hình I/O string2 và lấy giá trị nhập từ bàn phím lưu xuống n. Sau đó nhóm in string1 ra màn hình

```

8  main:                li $v0, 4
9                          la $a0, string2
10                         syscall
11                         li $v0, 5
12                         syscall
13                         sw $v0, n
14                         li $v0, 4
15                         la $a0, string1
16                         syscall

```

Sau đó nhóm lấy giá trị của n đã lưu trước đó gán vào \$a3, khởi tạo tổng và số trước rồi nhảy tới label fibonacci để tính toán

```

17                         lw $a3, n
18                         addi $a1, $a1, 1      #so truoc
19                         add  $a2, $a2, $zero #tong
20                         jal  fibonacci

```

Ở label fibonacci, nhóm sẽ so sánh \$a2 có bằng 0 hay không. Nếu khác thì nhảy sang label else. Nếu bằng thì sẽ lưu giá trị \$ra để nhảy về

```

22  fibonacci:          bne $a2, $zero, else
23                          addi $sp, $sp, -4
24                          sw  $ra, 0($sp)
25

```

Ở label else, nhóm xuất kết quả ra màn hình I/O kèm dấu cách

```

26  else:                                add $a0, $a2, $zero
27                                     li $v0, 1
28                                     syscall
29                                     li $v0, 4
30                                     la $a0, space
31                                     syscall

```

Sau đó nhóm gán $\$t0 = \$a2$, $\$a2 = \$a1 + \$a2$ và $\$a1 = \$t0$ vd:

Số trước 1 $\$a1$

Tổng cũ 0 $\$a2$; $\$t0 = \$a2$

Tổng mới 1 $\$a2 = \$a1 + \$a2$; $\$a1 = \$t0$

Số trước 1 $\$a1$

Tổng cũ 1 $\$a2$; $\$t0 = \$a2$

Tổng mới 2 $\$a2 = \$a1 + \$a2$; $\$a1 = \$t0$

Rồi nhóm trừ $\$a3$ đi cho 1 (bỏ dòng `slt $t1, $zero, $a3` nha thầy). Nếu $\$a3$ mà bằng 0 thì sẽ nhảy tới label `return`. Ngược lại thì lặp lại label `fibonacci`

```

32  add $t0, $a2, $zero
33  add $a2, $a1, $a2
34  add $a1, $t0, $zero
35  addi $a3, $a3, -1
36  slt $t1, $zero, $a3
37  beq $a3, $zero, return
38  jal fibonacci

```

Ở label `return`, nhóm sẽ lấy giá trị $\$ra$ (địa chỉ lệnh sau lệnh `jal fibonacci` ở label `main`) đã lưu ở stack gán vào $\$ra$ và nhảy về địa chỉ đó

```
39  return:                lw $ra, 0($sp)
40                                addi $sp, $sp, 4
41                                jr $ra
```

Và nhảy tới label exit

```
20                                jal fibonacci
21                                j exit
```