**THỰC HÀNH 4**

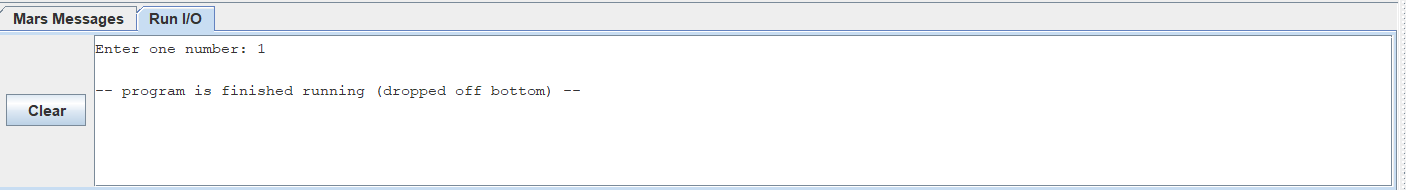
1. Sự thay đổi của các thanh ghi trong code hình 1 khi chạy từng lệnh:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Thanh ghi  Tên lệnh | PC | $ra | $sp | $fp |
| main: jal getInt | |  | | --- | | 0x0040000c | | |  | | --- | | 0x00400004 | | 0x7fffeffc | 0x00000000 |
| getInt: li $v0, 4 | 0x00400010 | |  | | --- | | 0x00400004 | | 0x7fffeffc | 0x00000000 |
| la $a0, prompt | 0x00400014 | |  | | --- | | 0x00400004 | | 0x7fffeffc | 0x00000000 |
| syscall | 0x0040001c | |  | | --- | | 0x00400004 | | 0x7fffeffc | 0x00000000 |
| li $v0, 5 | 0x00400020 | |  | | --- | | 0x00400004 | | 0x7fffeffc | 0x00000000 |
| syscall | 0x00400020 | |  | | --- | | 0x00400004 | | 0x7fffeffc | 0x00000000 |
| jr $ra | 0x00400004 | |  | | --- | | 0x00400004 | | 0x7fffeffc | 0x00000000 |
| move $s0, $v0 | 0x00400008 | |  | | --- | | 0x00400004 | | 0x7fffeffc | 0x00000000 |
| j exit | 0x00400028 | |  | | --- | | 0x00400004 | | 0x7fffeffc | 0x00000000 |

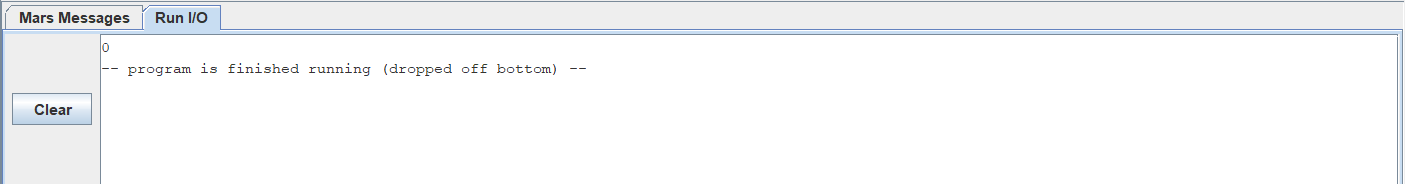
Sự thay đổi của các thanh ghi trong code hình 2 khi chạy từng lệnh:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Thanh ghi  Tên lệnh | PC | $ra | $sp | $fp |
| move $a0, $s0 | |  | | --- | | 0x00400004 | | |  | | --- | | 0x00000000 | | 0x7fffeffc | 0x00000000 |
| move $a1, $s1 | 0x00400008 | |  | | --- | | 0x00000000 | | 0x7fffeffc | 0x00000000 |
| move $a2, $s2 | 0x0040000c | |  | | --- | | 0x00000000 | | 0x7fffeffc | 0x00000000 |
| move $a3, $s3 | 0x00400010 | |  | | --- | | 0x00000000 | | 0x7fffeffc | 0x00000000 |
| jal proc\_example | 0x00400024 | |  | | --- | | 0x00400014 | | 0x7fffeffc | 0x00000000 |
| addi $sp, $sp, -4 | 0x00400028 | |  | | --- | | 0x00400014 | | 0x7fffeff8 | 0x00000000 |
| sw $s0, 0($sp) | 0x0040002c | |  | | --- | | 0x00400014 | | 0x7fffeff8 | 0x00000000 |
| add $t0, $a0, $a1 | 0x00400030 | |  | | --- | | 0x00400014 | | 0x7fffeff8 | 0x00000000 |
| add $t1, $a2, $a3 | 0x00400034 | |  | | --- | | 0x00400014 | | 0x7fffeff8 | 0x00000000 |
| sub $s0, $t0, $t1 | 0x00400038 | |  | | --- | | 0x00400014 | | 0x7fffeff8 | 0x00000000 |
| move $v0, $s0 | 0x0040003c | |  | | --- | | 0x00400014 | | 0x7fffeff8 | 0x00000000 |
| lw $s0, 0($sp) | 0x00400040 | |  | | --- | | 0x00400014 | | 0x7fffeff8 | 0x00000000 |
| addi $sp, $sp, 4 | 0x00400044 | |  | | --- | | 0x00400014 | | 0x7fffeffc | 0x00000000 |
| jr $ra | 0x00400014 | |  | | --- | | 0x00400014 | | 0x7fffeffc | 0x00000000 |
| move $a0, $v0 | 0x00400018 | |  | | --- | | 0x00400014 | | 0x7fffeffc | 0x00000000 |
| li $v0, 1 | 0x0040001c | |  | | --- | | 0x00400014 | | 0x7fffeffc | 0x00000000 |
| syscall | 0x00400020 | |  | | --- | | 0x00400014 | | 0x7fffeffc | 0x00000000 |
| j exit | 0x00400048 | |  | | --- | | 0x00400014 | | 0x7fffeffc | 0x00000000 |

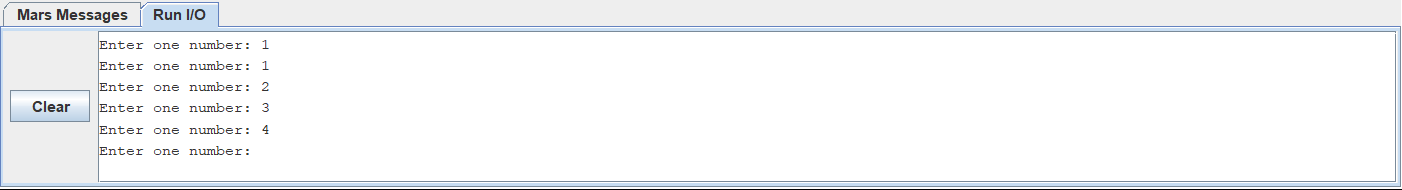
Kết quả khi chạy chương trình trong 1 lần của hình 1:



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

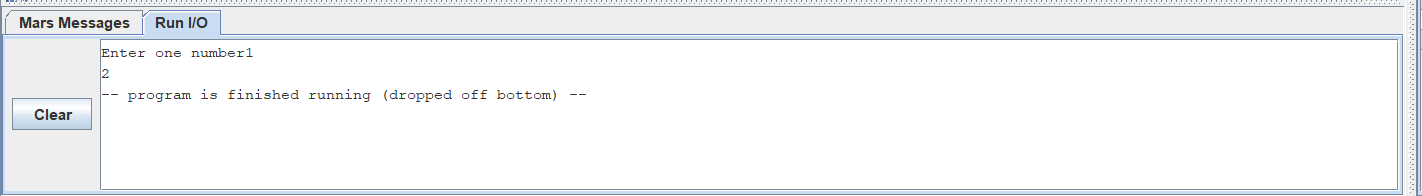


Nếu bỏ dòng j exit ở hình 1 đi chương trình sẽ nhập vô hạn lần

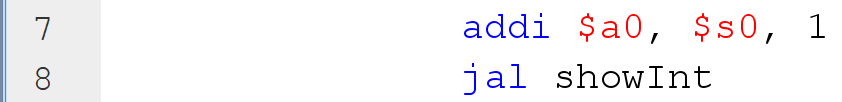


Mô phỏng thêm thủ tục ShowInt : [File code](https://github.com/kamuisi/KTMT/blob/main/lap4_1.asm)

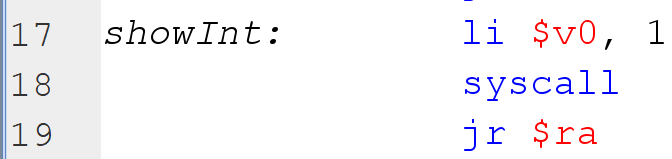
Kết quả thêm thủ tục ShowInt:



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



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

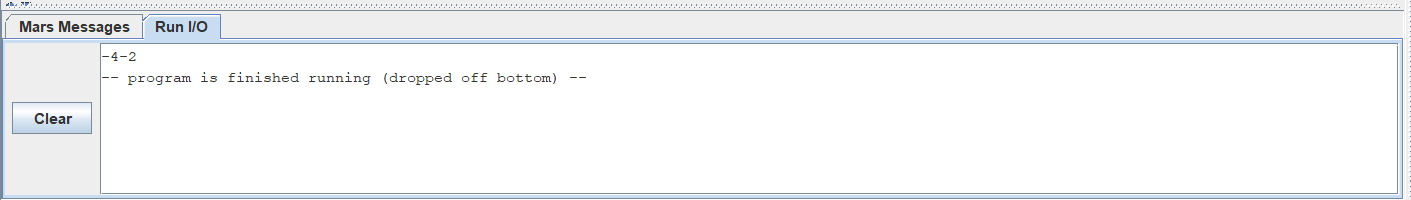


Rồi nhóm nhảy tới nhãn exit

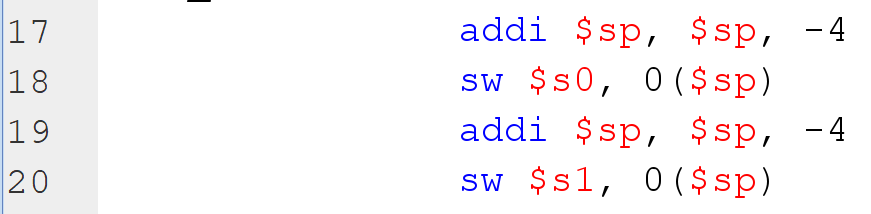


Mô phỏng code trong hình 2 sau khi chỉnh sửa: [File code](https://github.com/kamuisi/KTMT/blob/main/lap4_1_2.asm)

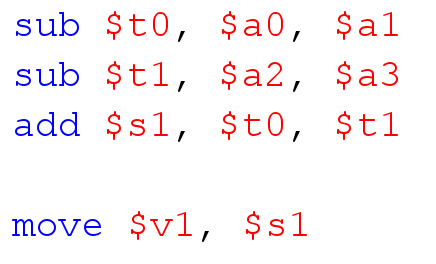
Kết quả với a = 1, b = 2, c = 3, d = 4:



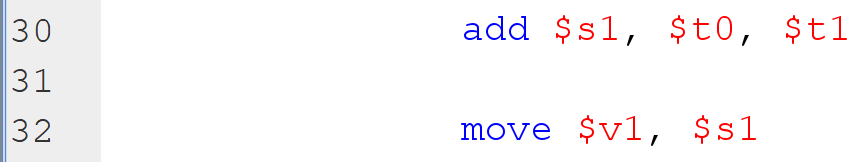
Giải thích: Đầu tiên nhóm sẽ thêm 1 dòng lưu giá trị $s1 xuống stack vào trong nhãn proc\_example



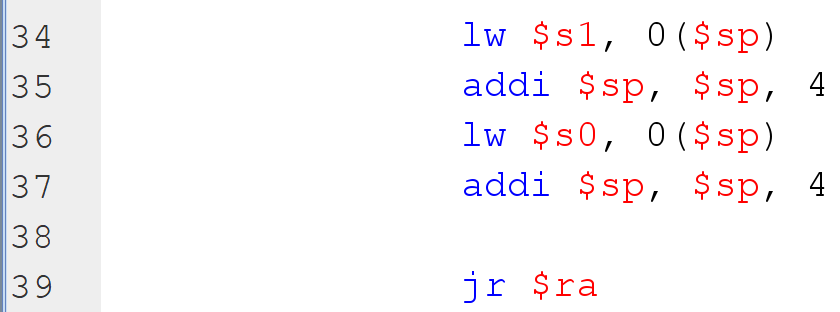
Do trong code ban đầu các biến $ax được truyền vào vẫn chưa bị thay đổi giá trị nên nhóm sẽ tái sử dụng lại bằng cách lấy $a0 trừ $a1 và $a2 trừ $a3



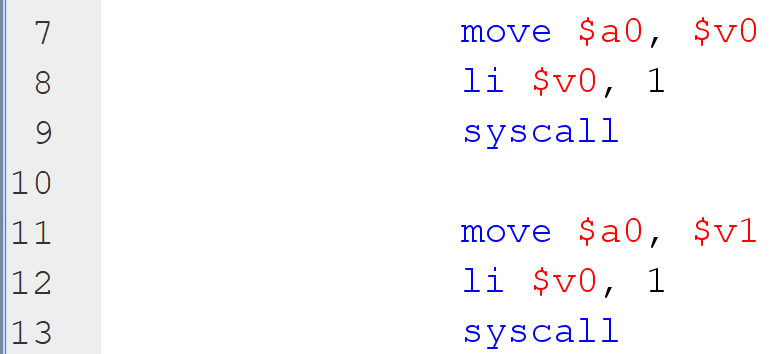
Và nhóm cộng 2 giá trị $t0 và $t1 lại, lưu vào $s1 và cho $v1 bằng $s1



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



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

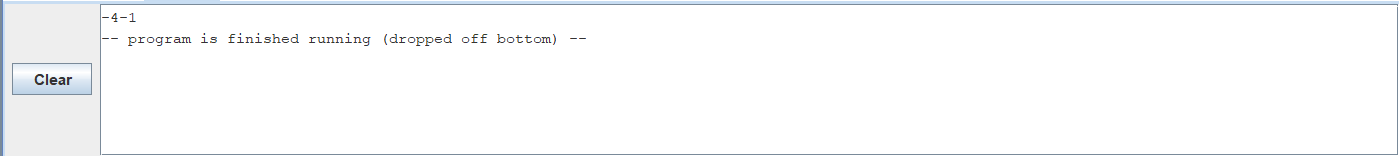


Rồi chương trình nhảy tới nhãn exit và kết thúc

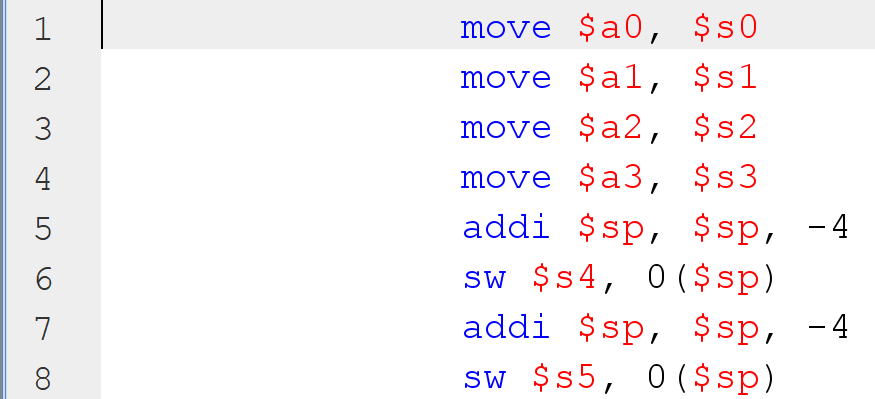


Mô phỏng code trong hình 2 sau khi chỉnh sửa lần nữa: [File code](https://github.com/kamuisi/KTMT/blob/main/lap4_1_3.asm)

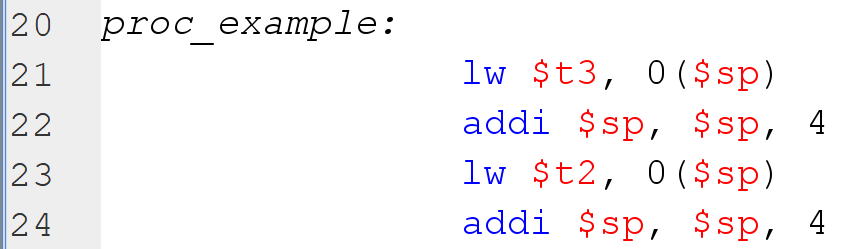
Kết quả với a = 1, b = 2, c = 3, d = 4, e = 5, f = 6:



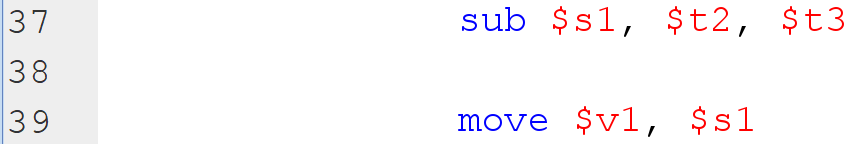
Giải thích: Do bài toán cần truyền tới 6 biến nên nhóm sẽ lưu 2 hai biến e và f xuống stack ở nhãn main



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

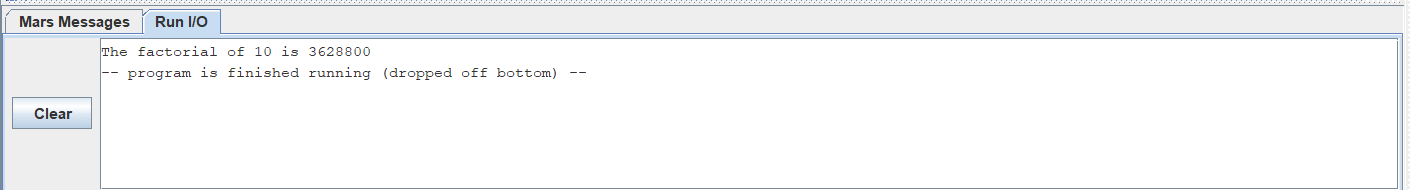


Để tính e – f thì nhóm lấy $t3 - $t4 và lưu vào $s1 rồi gán giá trị đó cho $v1



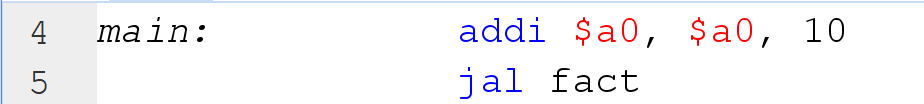
1. Mô phỏng: [File code](https://github.com/kamuisi/KTMT/blob/main/lap4_2.asm)

Kết quả:

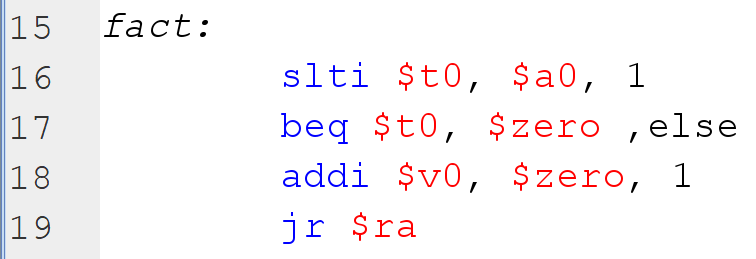


Giải thích:

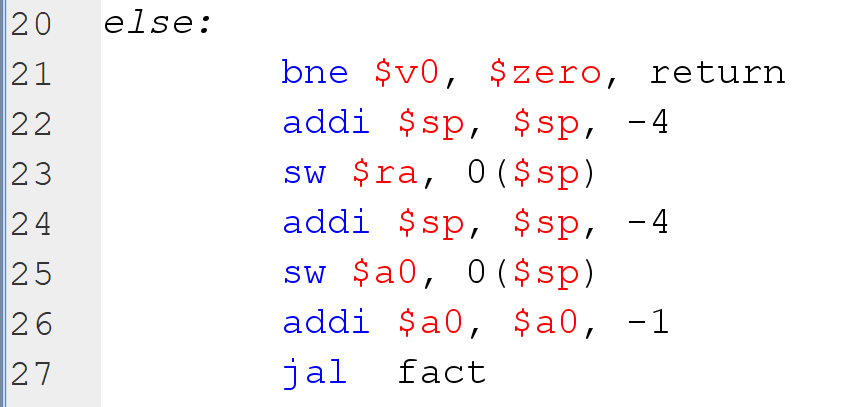
Nhóm sẽ truyền thẳng số n vào $a0. Sau đó nhảy đến label fact để tính toán n!



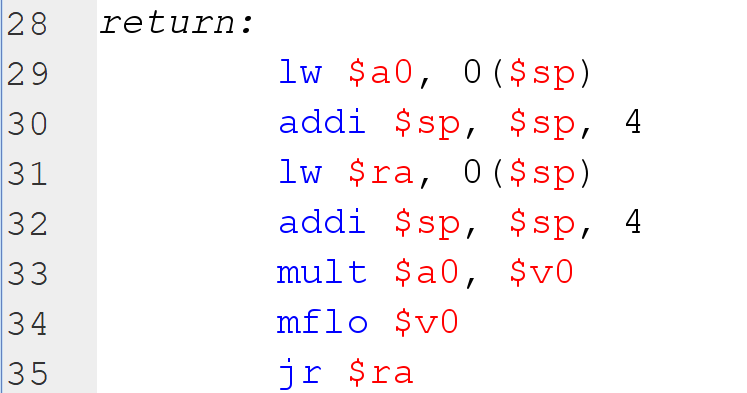
Ở label fact, nhóm sẽ so sánh $a0 với 1 xem $a0 có nhỏ hơn không. Nếu thỏa thì sẽ cho $v0 bằng 1 và quay lại địa chỉ $ra đang lưu. Nếu không thỏa thì sẽ nhảy tới label else



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

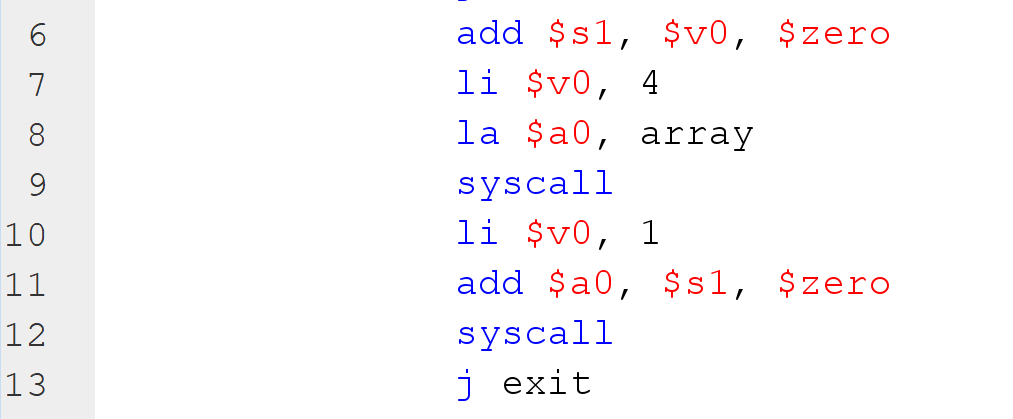


Khi fact đã được gọi bởi label else n – 1 lần thì n = 0 ($a0 = 0) dẫn đến điều kiện của if thỏa thì $v0 sẽ bằng 1 và chương trình sẽ nhảy tới địa chỉ $ra tức là địa chỉ của label return. Ở label return, nhóm sẽ lấy giá trị của $a0 và $ra (địa chỉ của return) đã lưu trong stack. Nhóm tính $v0 = $v0 x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



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



2. Stack trong trường hợp 5!

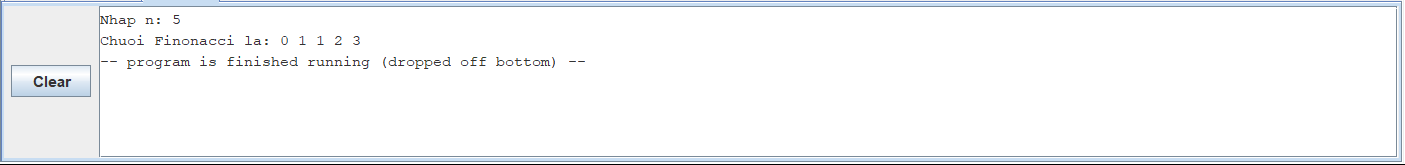
|  |  |  |
| --- | --- | --- |
| $fp | 0x00000000 | 0x00000000 |
|  | … | … |
| $sp | 0x7fffefd4 | 0x00000001 |
|  | 0x7fffefd8 | 0x00400058 |
|  | 0x7fffefdc | 0x00000002 |
|  | 0x7fffefe0 | 0x00400058 |
|  | 0x7fffefe4 | 0x00000003 |
|  | 0x7fffefe8 | 0x00400058 |
|  | 0x7fffefec | 0x00000004 |
|  | 0x7fffeff0 | 0x00400058 |
|  | 0x7fffeff4 | 0x00000005 |
|  | 0x7fffeff8 | 0x00400008 |

Stack trong trường hợp 10!

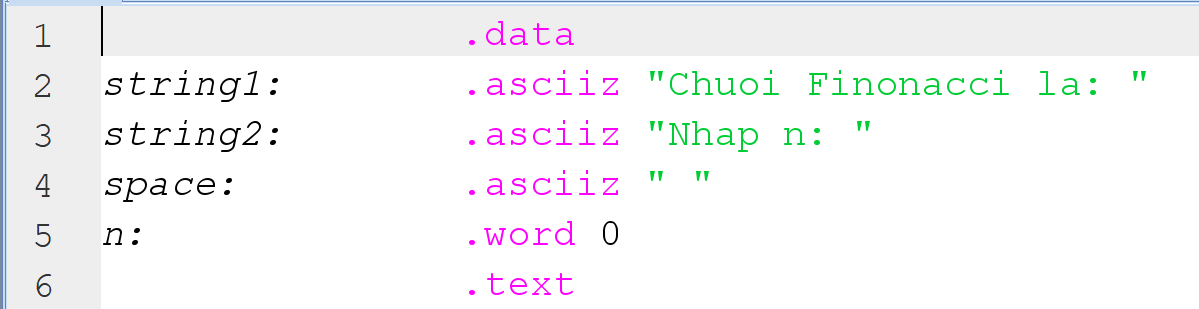
|  |  |  |
| --- | --- | --- |
| $fp | 0x00000000 | 0x00000000 |
|  | … | … |
| $sp | 0x7fffefac | 0x00000001 |
|  | 0x7fffefb0 | 0x00400058 |
|  | 0x7fffefb4 | 0x00000002 |
|  | 0x7fffefb8 | 0x00400058 |
|  | 0x7fffefbc | 0x00000003 |
|  | 0x7fffefc0 | 0x00400058 |
|  | 0x7fffefc4 | 0x00000004 |
|  | 0x7fffefc8 | 0x00400058 |
|  | 0x7fffefcc | 0x00000005 |
|  | 0x7fffefd0 | 0x00400058 |
|  | 0x7fffefd4 | 0x00000006 |
|  | 0x7fffefd8 | 0x00400058 |
|  | 0x7fffefdc | 0x00000007 |
|  | 0x7fffefe0 | 0x00400058 |
|  | 0x7fffefe4 | 0x00000008 |
|  | 0x7fffefe8 | 0x00400058 |
|  | 0x7fffefec | 0x00000009 |
|  | 0x7fffeff0 | 0x00400058 |
|  | 0x7fffeff4 | 0x0000000a |
|  | 0x7fffeff8 | 0x00400008 |

2) Mô phỏng: [File code](https://github.com/kamuisi/KTMT/blob/main/lap4_3b.asm)

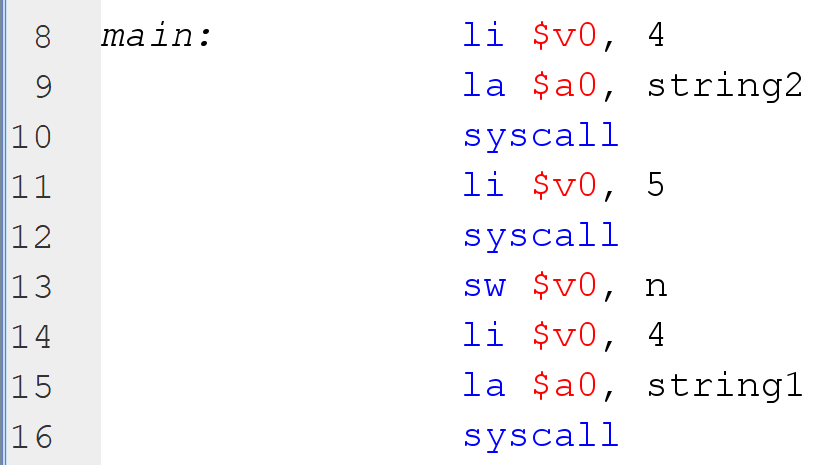
Kết quả:



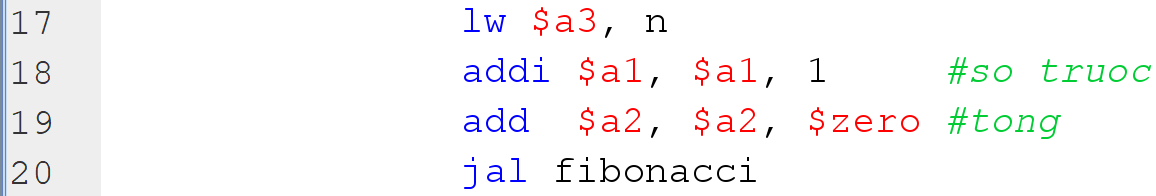
Giải thích: đầu chương trình nhóm sẽ khai báo như này



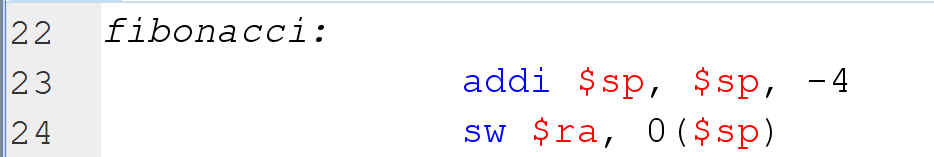
Tiếp đến nhóm sẽ in ra màn hình I/O string2 và lấy giá trị nhập từ bàn phím lưu xuống n. Sau đó nhóm in string1 ra màn hình



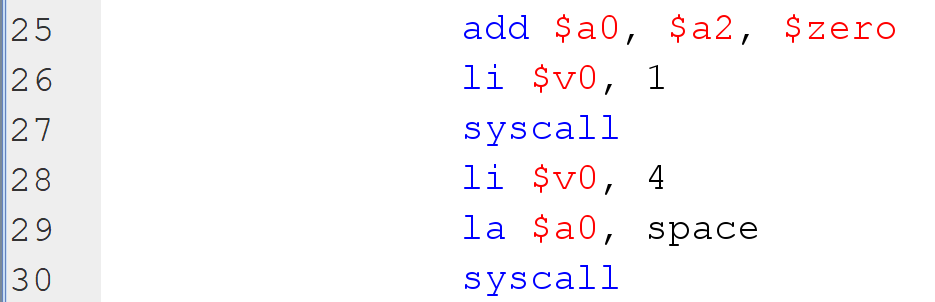
Sau đó nhóm lấy giá trị của n đã lưu trước đó gán vào $a3, khởi tạo tổng và số trước rồi nhảy tới label fibonacci để tính toán



Ở label fibonacci, nhóm sẽ lưu giá trị $ra để nhảy về



Tiếp theo nhóm xuất kết quả ra màn hình I/O kèm dấu cách



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

Số trước 1 $a1

Tổng cũ 0 $a2; $t0 = $a2

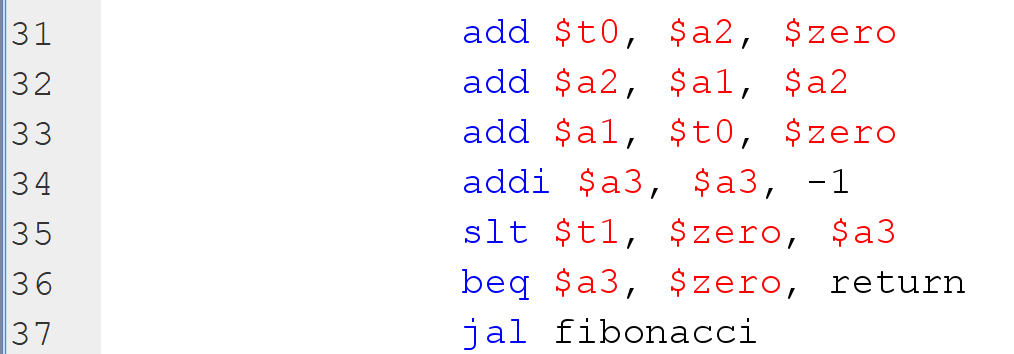
Tổng mới 1 $a2 = $a1 + $a2; $a1 = $t0

Số trước 1 $a1

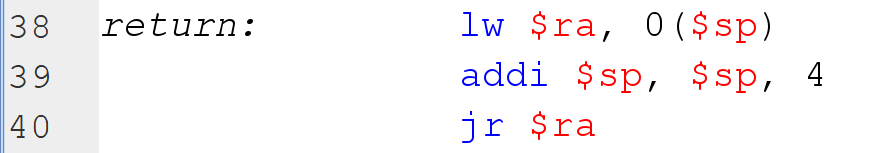
Tổng cũ 1 $a2; $t0 = $a2

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



Ở label return, nhóm sẽ lấy giá trị $ra đã lưu ở stack gán vào $ra và nhảy về địa chỉ đó



Và sau một hồi chương trình sẽ nhảy tới lệnh sau lệnh jal fibonacci đầu tiên thì sẽ nhảy tới label exit

