

BÁO CÁO THỰC HÀNH LAB 7

NGUYỄN THỊ MINH CHÂU

20214997

Bài 1:

```
1  .text
2
3  main:
4
5      li $a0, 100 #load input parameter
6      jal abs #jump and link to abs procedure
7      nop
8      add $s0, $zero, $v0
9      li $v0, 1 #terminate
10     move $a0, $s0
11     syscall
12     li $v0, 10
13     syscall
14
15 endmain:
16 |
17 abs:
18
19     sub $v0,$zero,$a0 #put -(a0) in v0; in case (a0)<0
20     bltz $a0,done #if (a0)<0 then done
21     nop
22     add $v0,$a0,$zero #else put (a0) in v0
23 done:
24
25     jr $ra
26
```

- Gán giá trị cho a0 = 100
- Sau đó nhảy tới abs
- Lưu v0 = số đối của A
- So sánh a0 có < 0 hay không, nếu đúng thì nhảy tới done
- Done nhảy tới địa chỉ của ra tức là địa chỉ của nop đầu tiên
- Nếu a0 < 0 tức là số đối của nó là v0 > 0 và nhảy tới nop đầu tiên, gán giá trị cho s0 và dùng syscall để kết thúc chương trình
- Nếu a0 > 0 tức là số đối v0 < 0 thì thực hiện lệnh v0 = a0 + 0 để v0 > 0 = a0
- Sau đó nhảy tới nop đầu tiên và in ra

100

-- program is finished running --

Clear

Bài 2:

```
1  .text
2  main:  li $a0,2 #load test input
3         li $a1,9
4         li $a2,6
5  jal max #call max procedure
6  nop
7  move $s0, $v0
8  li $v0, 1
9  move $a0, $s0
10 syscall
11 li $v0, 10
12 syscall
13 j endmain
14 max:   add $v0,$a0,$zero #copy (a0) in v0; largest so far
15       sub $t0,$a1,$v0 #compute (a1)-(v0)
16       bltz $t0,okay #if (a1)-(v0)<0 then no change
17       nop
18       add $v0,$a1,$zero #else (a1) is largest thus far
19 okay:
20       sub $t0,$a2,$v0 #compute (a2)-(v0)
21       bltz $t0,done #if (a2)-(v0)<0 then no change
22       nop
23       add $v0,$a2,$zero #else (a2) is largest overall
24 done:
25       jr $ra #return to calling program
26 endmain:
```

- Gán giá trị cho a0, a1, a2 sau đó nhảy đến max
- $v0 = a0 + 0$
- $t0 = a1 - v0$ ($a1 - a0$)
- nếu $t0 < 0$ thì nhảy tới okay ($v0$ được coi là max, và lúc này $a0$ đang lớn hơn $a1$)
- so sánh $a0$ với $a2$
- Tương tự như vậy ta nhảy tới done để nhảy tới nop kết thúc chương trình
- Còn ở bước trên nếu như $t0 > 0$ tức là $v0 = a1$ vì lúc này $a1$ đang lớn hơn $a0$
- Vậy nên ta có bước số 2 là mang $v0$ đi so sánh với $a2$ tức là mang $a1$ đi so sánh với $a2$

9

-- program is finished running --

Clear

Bài 3:

```
1  .text
2      addi $s0, $0, 4
3      addi $s1, $0, 10
4  push:
5      addi $sp, $sp, -8 #adjust the stack pointer
6      sw $s0, 4($sp) #push $s0 to stack
7      sw $s1, 0($sp) #push $s1 to stack
8  work:
9      nop
10     nop
11     nop
12  pop:
13     lw $s0, 0($sp) #pop from stack to $s0
14     lw $s1, 4($sp) #pop from stack to $s1
15
16
```

- Khai báo \$s0 = 4, \$s1 = 10
- Push: tạo ra 2 ô nhớ bằng cách $\$sp = \$sp - 8$, sau đó lưu \$s0 và \$s1 vào các địa chỉ $\$sp+4$ và $\$sp$. Giá trị ở 2 thanh ghi đã được hoán đổi.
- Pop: lưu giá trị $\$sp$ và $\$sp+4$ vào địa chỉ \$s0 và \$s1, thực hiện lấy ra 2 số của stack. Tính $sp = sp + 8$ làm cho \$sp trở lại giá trị ban đầu.

Bài 4:

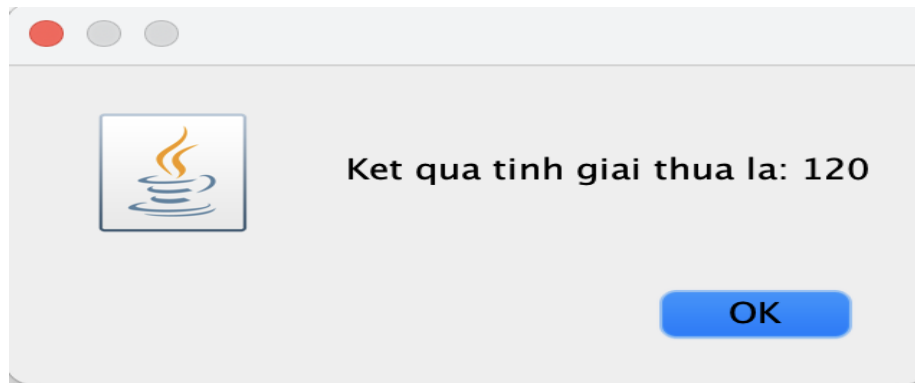
```
1  .data
2      Message: .asciiz "Ket qua tinh giai thua la: "
3  .text
4  main:
5      jal WARP
6  print:
7      add $a1, $v0, $zero    #a0 = result from N!
8      li $v0, 56
9      la $a0, Message
10     syscall
11  quit:
12     li $v0, 10             #terminate
13     syscall
14  endmain:
15 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

```

25         jal FACT                #call fact procedure
26         nop
27
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
45         slti $t0, $a0, 2         #if input argument N<2
46         beq $t0, $zero, recursive #if it is false ((a0 = N) >= 2)
47         nop
48         li $v0, 1                #return the result N! = 1
49         j done
50         nop
51 recursive:
52         addi $a0, $a0, -1        #adjust input argument
53         jal FACT                #recursive call
54         nop
55         lw $v1, 0($sp)          #load a0
56         mult $v1, $v0            #compute the result
57         mflo $v0
58 done:
59         lw $ra, 4($sp)          #restore return address
60         lw $a0, 0($sp)          #restore a0
61         addi $sp, $fp, 0         #restore stack pointer
62         lw $fp, -4($sp)         #restore frame pointer
63         jr $ra                  #jump to calling
64 fact_end:

```

- Ta đẩy \$fp và \$ra của chương trình chính vào stack
- Khởi tạo n = 5 lưu tại a0
- Sau đó nhảy đến FACT
- Tiếp tục cho đến bước n ta lại lưu \$fp, \$ra, \$a0 vào stack
- Sau đó ta kiểm tra nếu a0 > 2 thì tiếp tục đệ quy hàm Fact
- Giảm a0 đi 1 đến khi a0 = 1 thì nhảy đến done
- Quay trở lại đọc frame khi a0 = 2 trong stack thì nhân giá trị a0 trong stack vs kết quả của thủ tục trước và lưu vào v0 đến khi hết stack



						\$sp(3) ->	\$a0(2) = 1
							\$ra(2)
							\$fp(2)
						\$fp(3) ->	\$a0(1) = 2
			\$sp(2) ->	\$a0(1) = 2		\$sp(2) ->	\$a0(1) = 2
				\$ra(1)			\$ra(1)
				\$fp(1)			\$fp(1)
			\$fp(2) ->	\$a0(0) = 5		\$fp(2) ->	\$a0(0) = 5
\$sp(1) ->	\$a(0) = 5		\$sp(1) ->	\$a(0) = 5		\$sp(1) ->	\$a(0) = 5
	\$ra(0)			\$ra(0)			\$ra(0)
	\$fp(0)			\$fp(0)			\$fp(0)
\$fp(1) ->			\$fp(1) ->			\$fp(1) ->	
\$sp(0) ->			\$sp(0) ->			\$sp(0) ->	

\$fp(0) ->			\$fp(0) ->			\$fp(0) ->	
Lần gọi 1 (a0 = 3)			Lần gọi 2 (a0 = 2)			Lần gọi 3 (a0 = 3)	

Bài 5:

```

2  .data
3  message: .asciiz "Largest : "
4  message1: .asciiz "\n\nSmallest : "
5  message2: .asciiz " , "
6  .text
7      li $s0, 3
8      li $s1, 3
9      li $s2, 6
10     li $s3, 9
11     li $s4, 98
12     li $s5, 8
13     li $s6, 2
14     li $s7, 0
15     move $t0, $s0           # max = s0
16     move $t1, $s0           # min = s0
17     li $t3, 0                # i = 0
18     li $t4, 8                # n = 8
19     li $a2, 0                # index của max = 0
20     li $a3, 0                # index của min = 0
21 push:
22     addi $sp, $sp, -32
23     sw $s0, 0($sp)
24     sw $s1, 4($sp)
25     sw $s2, 8($sp)
26     sw $s3, 12($sp)
27     sw $s4, 16($sp)
28     sw $s5, 20($sp)
29     sw $s6, 24($sp)
30     sw $s7, 28($sp)
31 main:
32     jal work
33     j print
34 work:
35     slt $t5, $t3, $t4         # t3 < t4 ?
36     beq $t5, $zero, end_work  # nho hon ket thuc
37     add $t6, $t3, $t3         # t6 = t3 + t3
38     add $t6, $t6, $t6         # t6 = t6 + t6 = 4*t3
39     add $t7, $t6, $sp         # t7 = t6 + sp
40     lw $t8, 0($t7)           # lay gia tri o dia chi t7 luu vao t
41     slt $t5, $t8, $t0         #kiem tra t8 < t0 ( max > stack)
42     beq $t5, $zero, wrap
43 continuel: slt $t5, $t8, $t1    # t8 < t1 ( stack <min )
44     bne $t5, $zero, wrap1
45 continue: addi $t3, $t3, 1     # i = i + 1
46     j work
47
48 wrap:

```

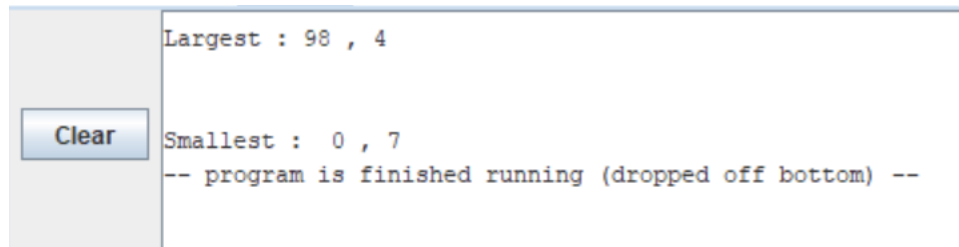
```

48 wrap:
49     add $t0, $t8, $zero      # max = t8
50     move $a2, $t3           # a2 = i
51     j continuel
52 wrap1:
53     add $t1, $t8, $zero      # min = t8
54     move $a3, $t3           # a3 = i
55     j continue
56 end_work:
57     jr $ra
58 print:
59     li $v0, 4
60     la $a0, message
61     syscall
62     li $v0, 1
63     move $a0, $t0
64     syscall
65     li $v0, 4
66     la $a0, message2
67     syscall
68     li $v0, 1
69     move $a0, $a2
70     syscall
71     li $v0, 4
72     la $a0, message1
73     syscall
74     li $v0, 1
75     move $a0, $t1
76     syscall
77     li $v0, 4
78     la $a0, message2
79     syscall
80     li $v0, 1
81     move $a0, $a3
82     syscall

```

- Khởi tạo các giá trị từ \$s0 đến \$s7 lần lượt là $i = 0$, $n = 8$, $\text{max} = \$s0$, $\text{min} = \$s0$, $\$a2 = 0$, $\$a3 = 0$
- Thực hành push các giá trị của mảng vào stack bằng \$sp.
- $\$sp = \$sp - 32$ để tạo ra 8 ô nhớ cho 8 phần tử của mảng. Lần lượt push bằng sw
- Work: so sánh $t3 < t4$ ($i < n$).
- Nếu $i > n$ thì thoát vòng lặp còn ngược lại tiếp tục thực hiện vòng lặp duyệt stack và tăng i thêm 1 đơn vị.
- $\$t6 = \$t6 + \$t6 = 4 * \$t3$, $\$t7 = \$sp + \$t6$, lw để lấy giá trị của stack. Với mỗi giá trị đó so sánh \$t8 với max và min.
- Nếu \$t8 lớn hơn max thì $\text{max} = \$t8 + 0$ và chỉ số lớn nhất bằng i .

- Nếu \$t8 nhỏ hơn min thì $\text{min} = \$t8 + 0$ và chỉ số nhỏ nhất $\$a3 = \$t3 = i$.
- Thoát chương trình con và thực hiện chương trình chính bằng lệnh jr \$ra, thực hiện in ra màn hình



The screenshot shows a window with a light gray background. On the left side, there is a vertical gray bar containing a blue button with the text "Clear". To the right of the button, the output of a program is displayed in a monospaced font. The output consists of three lines: "Largest : 98 , 4", "Smallest : 0 , 7", and "-- program is finished running (dropped off bottom) --".

```
Largest : 98 , 4  
Smallest : 0 , 7  
-- program is finished running (dropped off bottom) --
```