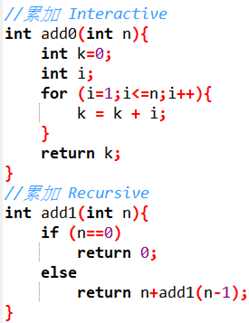
Leaf vs Non-Leaf

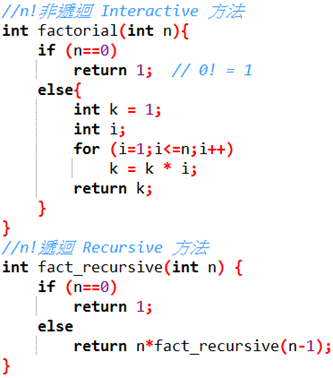
[sourece: https://sites.google.com/a/qtm.ks.edu.tw/qtm052-org/3-ban-ji-jing-ying-yu-fu-dao/dev-c/di-hui-recursive]

Leaf

Non-Leaf

[[](https://sites.google.com/a/qtm.ks.edu.tw/qtm052-org/3-ban-ji-jing-ying-yu-fu-dao/dev-c/di-hui-recursive/Add.png?attredirects=0)](https://sites.google.com/a/qtm.ks.edu.tw/qtm052-org/3-ban-ji-jing-ying-yu-fu-dao/dev-c/di-hui-recursive/Add.png?attredirects=0)

計算n!

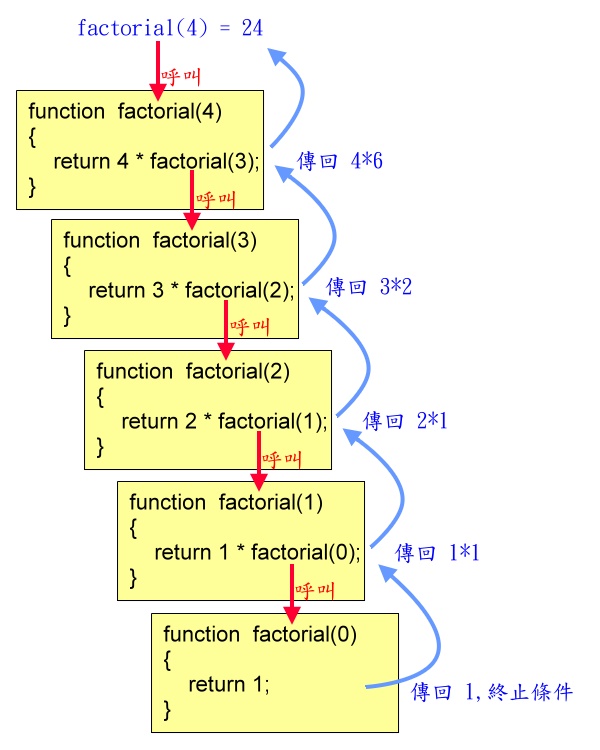
[](https://sites.google.com/a/qtm.ks.edu.tw/qtm052-org/3-ban-ji-jing-ying-yu-fu-dao/dev-c/di-hui-recursive/n%E9%9A%8E%E5%B1%A4.png?attredirects=0)

Leaf

Non-Leaf

[source: http://simonsays-tw.com/web/Recursion/Iteration&Recursion.html]

Non-Leaf 在執行Machine Code 的方式



6

24

2

1

1

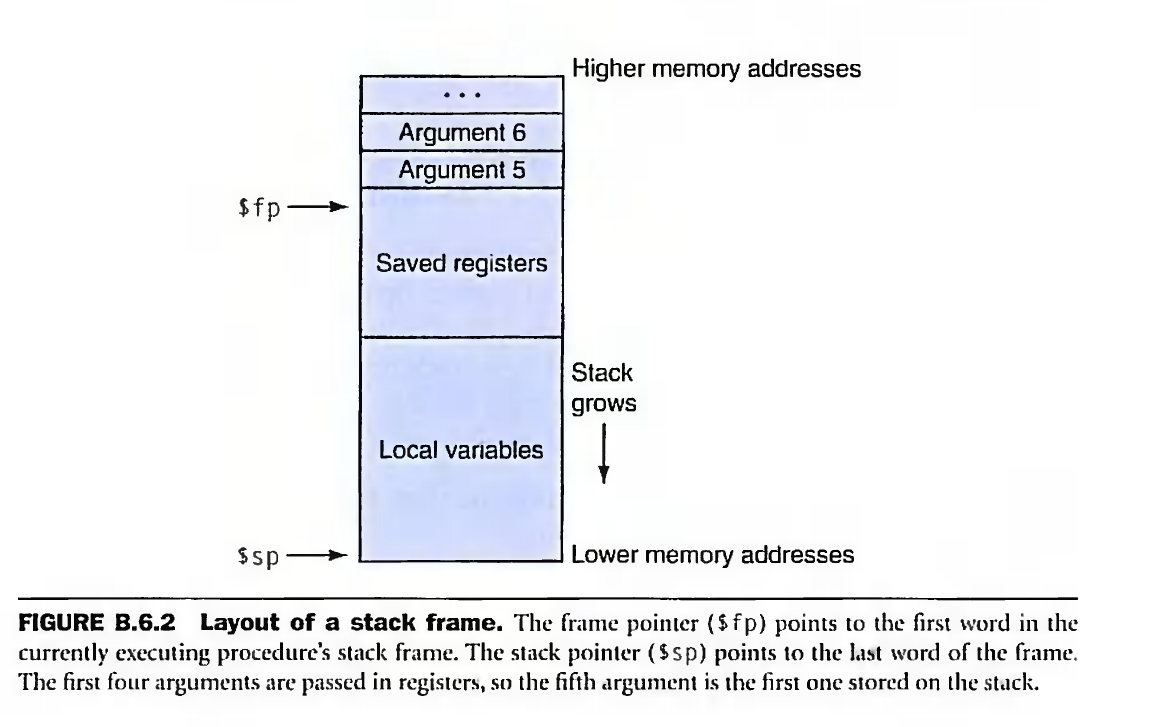
Non-Leaf Memory 內容的變化與狀態，for example，進入 fact Non-Leaf 時 $SP=100; $ra=3000，fact為記憶中的某一個位址。以 2! ($a0=2) 追蹤一下 Non-Leaf 記憶體的變化與狀態，以及如何得到2! 的結果。

(memory address)

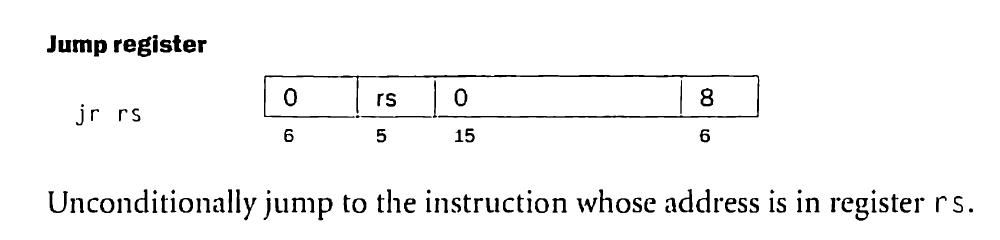
fact:  
 addi $sp, $sp, -8 # adjust stack for 2 items  
fact+4 sw $ra, 4($sp) # save return address  
fact+8 sw $a0, 0($sp) # save argument  
fact+12 slti $t0, $a0, 1 # test for n < 1  
fact+16 beq $t0, $zero, L1  
fact+20 addi $v0, $zero, 1 # if so, result is 1  
fact+24 addi $sp, $sp, 8 # pop 2 items from stack  
fact+28 jr $ra # and return  
L1: addi $a0, $a0, -1 # else decrement n #L1==fact+32  
L1+4 jal fact # recursive call  
L1+8 lw $a0, 0($sp) # restore original n  
L1+12 lw $ra, 4($sp) # and return address  
L1+16 addi $sp, $sp, 8 # pop 2 items from stack  
L1+20 mul $v0, $a0, $v0 # multiply to get result  
L1+24 jr $ra # and return

**Review:**

* **$sp & $fp Register**

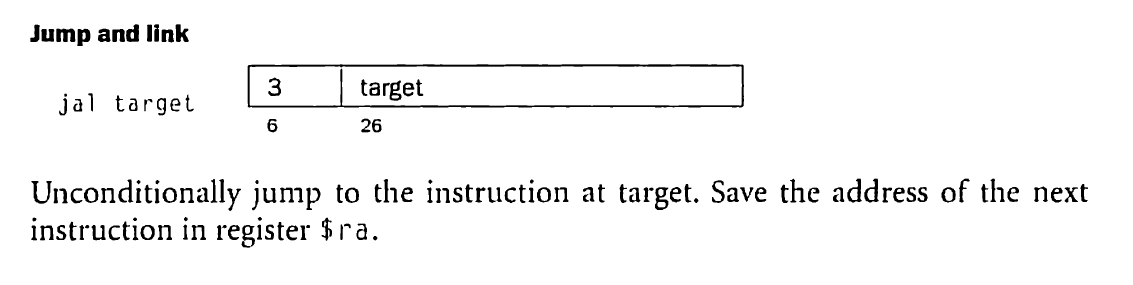
****

* **jr $ra**



Here, rs == $ra

* **jal fact**



$ra=3000, $a0=2(i.e. n=2, 2!) when fact is called.

fact:  
 addi $sp, $sp, -8 # adjust stack for 2 items  
 sw $ra, 4($sp) # save return address  
 sw $a0, 0($sp) # save argument

slti $t0, $a0, 1 # test for $a0 < 1🡪yes, $t0=1

****beq $t0, $zero, L1 # go to L1: if $t0=0

L1: addi $a0, $a0, -1 # else decrement n   
 jal fact # recursive call

**……….. until $a0=0**

fact:  
 addi $sp, $sp, -8 # adjust stack for 2 items  
 sw $ra, 4($sp) # save return address  
 sw $a0, 0($sp) # save argument

# Here $a0=0

# back to fact

fact:  
 addi $sp, $sp, -8 # adjust stack for 2 items  
 sw $ra, 4($sp) # save return address  
 sw $a0, 0($sp) # save argument  
 slti $t0, $a0, 1 # test for n < 1  
 beq $t0, $zero, L1  
 addi $v0, $zero, 1 # if so, result is 1  
 addi $sp, $sp, 8 # pop 2 items from stack  
 jr $ra # and return

here:

#v0 = 1

#ra=L1+8

#$sp=84

**L1+8:**  
 lw $a0, 0($sp) # restore original n  
 lw $ra, 4($sp) # and return address  
 addi $sp, $sp, 8 # pop 2 items from stack

here:

#a0 = 1

#ra=L1+8

#$sp = 92

mul $v0, $a0, $v0 # multiply to get result  
 jr $ra # and return

here:

#v0 = 1\*1

#ra=L1+8

**L1+8:**  
 lw $a0, 0($sp) # restore original n  
 lw $ra, 4($sp) # and return address  
 addi $sp, $sp, 8 # pop 2 items from stack

here:

#a0 = 2

#ra=3000

#$sp = 100

mul $v0, $a0, $v0 # multiply to get result  
 jr $ra # and return

here:

#v0 = 2\*1

# return to caller