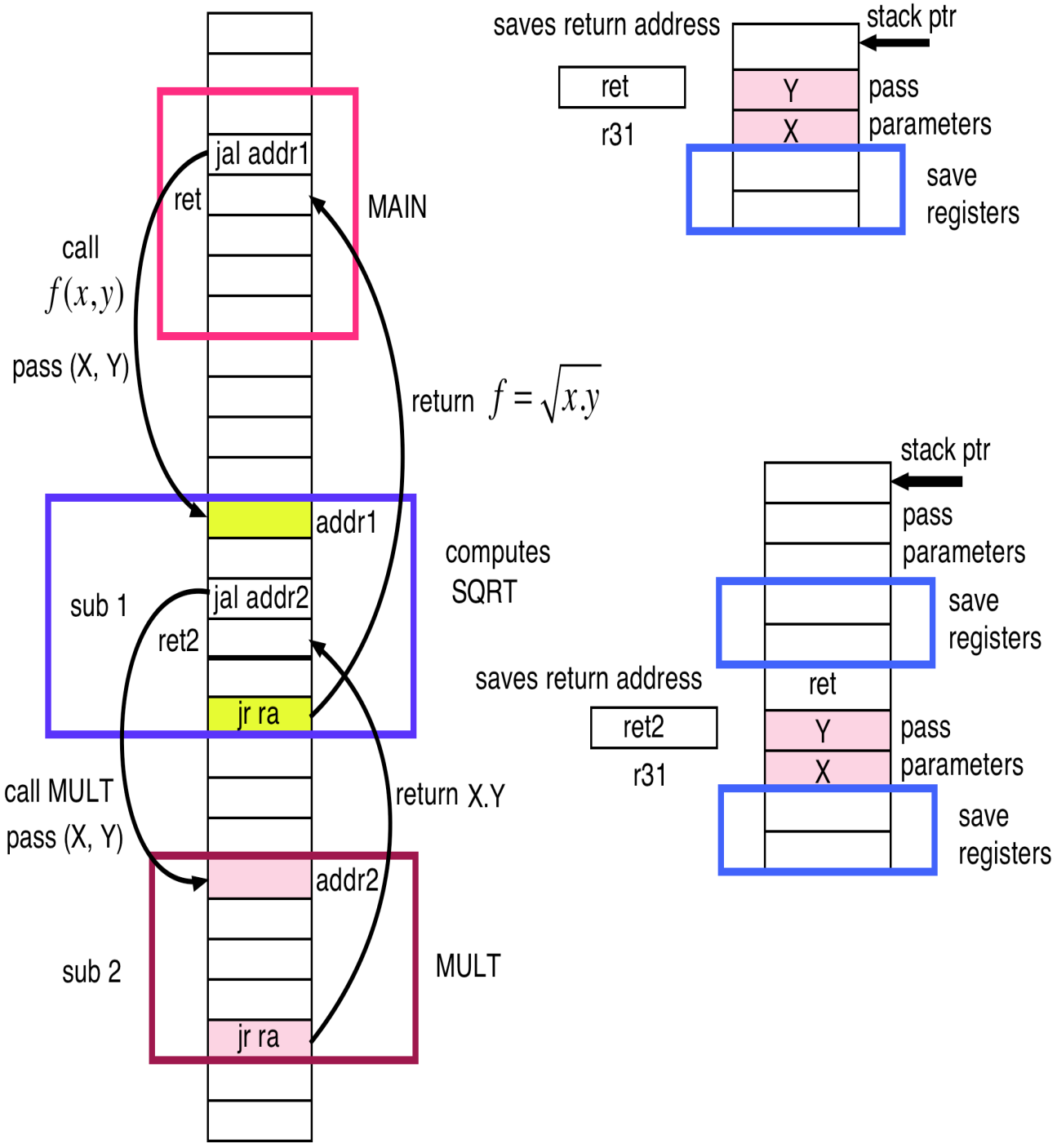
Nested subroutine call

$$f(x,y)=\sqrt{x.y}$$



Handling recursive procedure calls

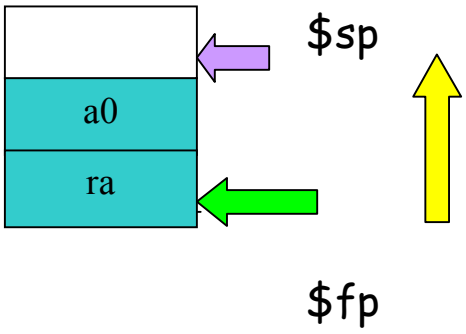
Example. Compute factorial (n)

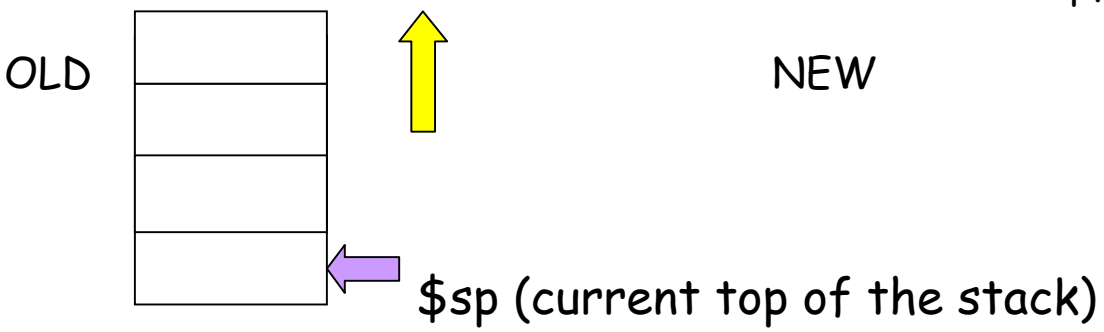
```
int fact (int n)
{
      if (n < 1) return (1);
            else return (n * fact(n-1))
}
```

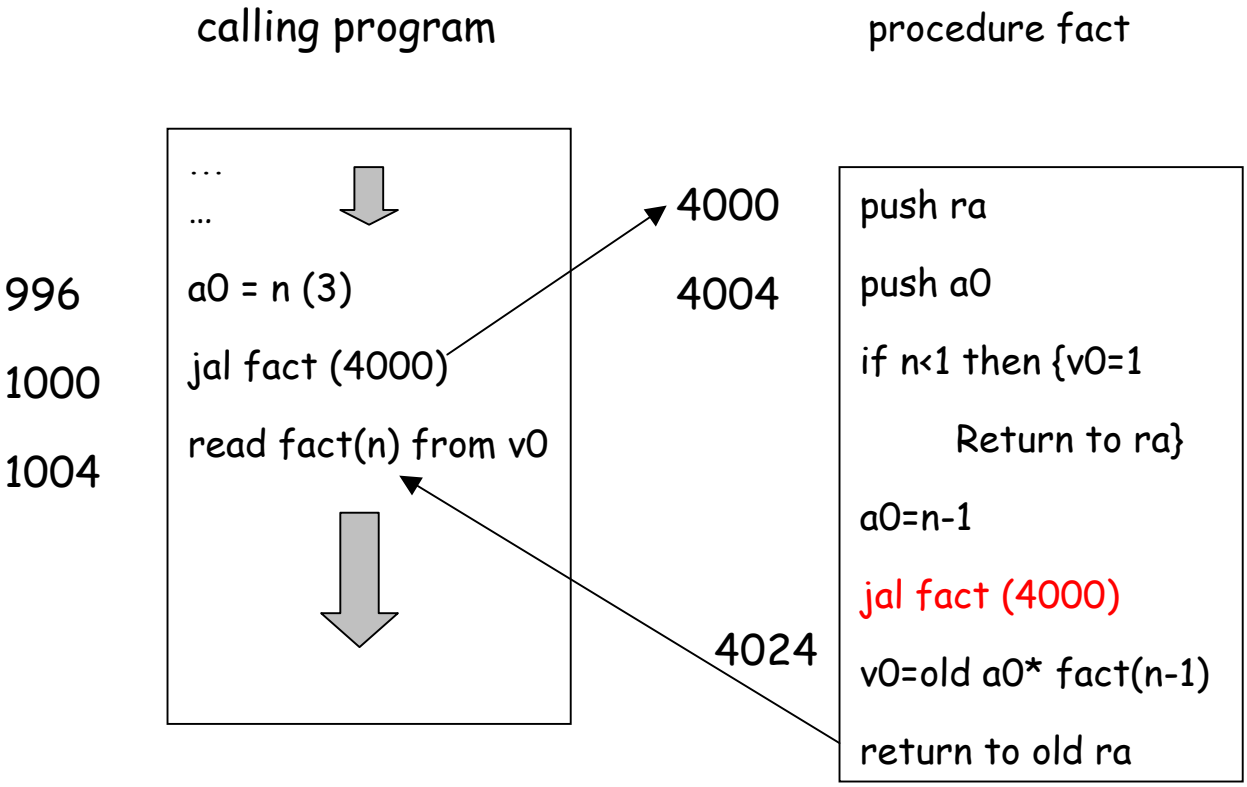
(Plan) Put n in $a0. Result should be available in $v0.

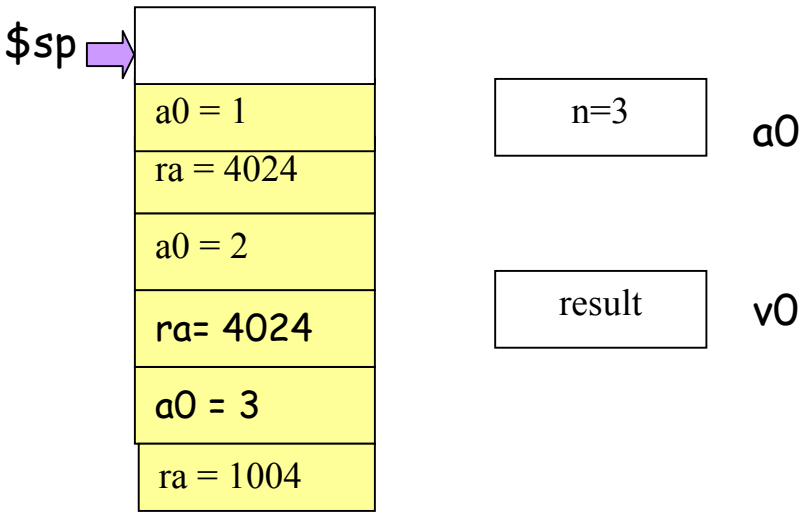
{Structure of the fact procedure}

```
fact:    subi $sp, $sp, 8
         sw   $ra, 4($sp) {why?}
         sw   $a0, 0($sp)
```









The growth of the stack as the recursion unfolds

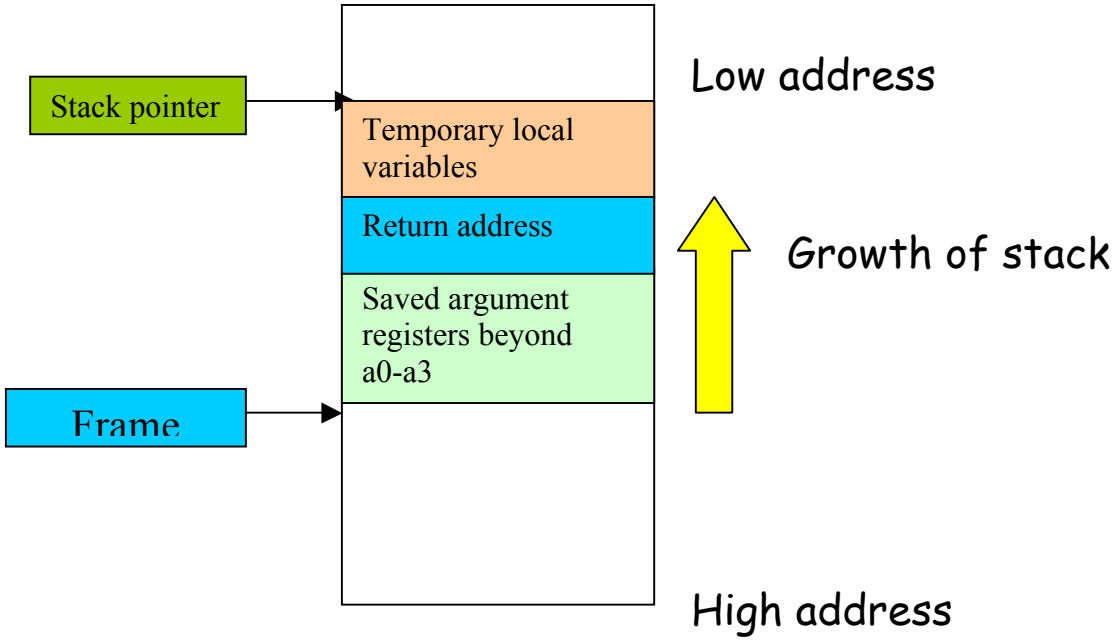
Now test if n < 1 (i.e. n = 0). In that case return 0 to $v0.

```
      slti  $t0, $a0, 1        # if n ≥ 1 then goto L1
      beq   $t0, $zero, L1
      addi  $v0, $zero, 1      # return 1 to $v0
      addi  $sp, $sp, 8        # pop 2 items from stack
      jr    $ra                # return
L1:   addi  $a0, $a0, -1       # decrement n
      jal   fact               # call fact with (n – 1)
```

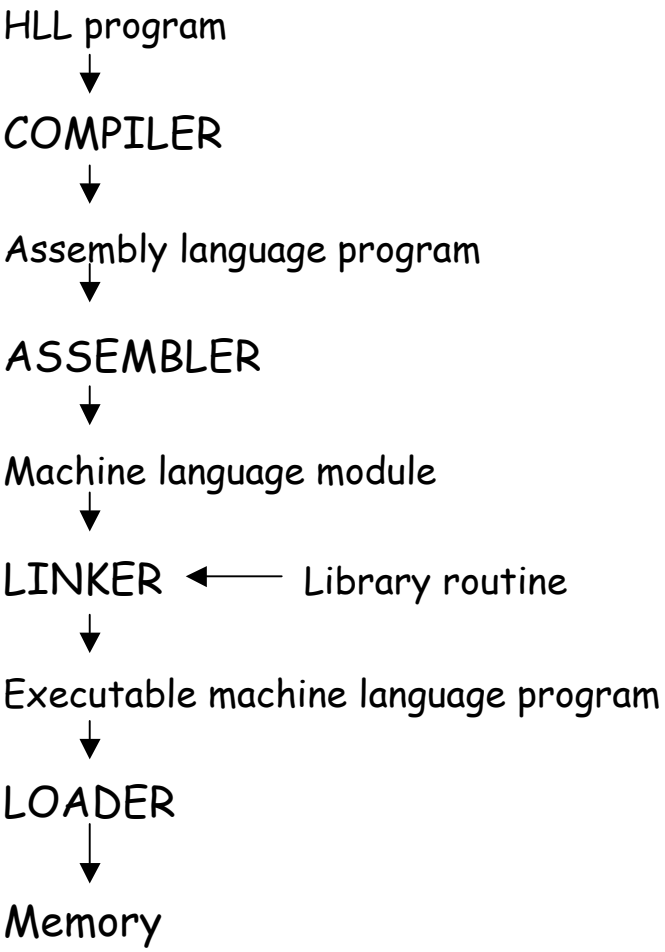
Now, we need to compute n * fact (n-1)

```
      lw    $a0, 0($sp)        # restore argument n
      lw    $ra, 4($sp)        # restore return address
      addi  $sp, $sp, 8        # pop 2 items
      mult  $v0, $a0, $v0      # return n * fact(n-1)
      jr    $ra                # return to caller
```

Run time environment of a MIPS program



A translation hierarchy



What are Assembler directives?

Instructions that are not executed, but they tell the assembler about how to interpret something. Here are some examples:

```
. text
{Program instructions here}

. data
{Data begins here}
. byte 84, 104, 101
. asciiz "The quick brown fox"
. float f1,. . . , fn
. word w1, . . . . wn
. space n  {reserve n bytes of space}
```

How does an assembler work?

In a two-pass assembler

PASS 1: Symbol table generation

PASS 2: Code generation

Follow the example in the class.