Funzioni ricorsive

Funzione C per il calcolo del fattoriale:

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

Fattoriale: 0! = 1

n! = n* (n-1)!

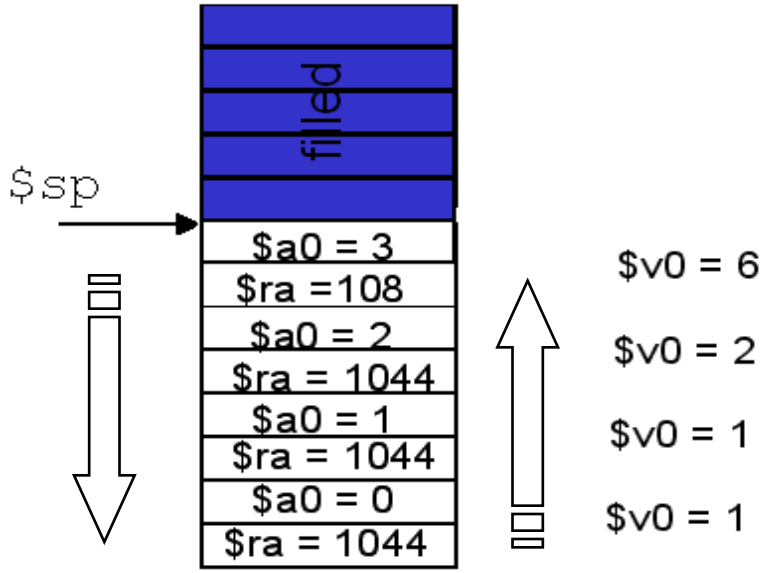
Funzione ricorsiva

Caller:

```
100 addi $a0,$zero,3
104 jal fact
108 ....
```

```
instruction
address
1004   fact: addi $sp,$sp,-8
1008         sw   $ra,0($sp)
1012         sw   $a0,4($sp)
1016         slti $t0,$a0,1      # test if n<1
1020         beq  $t0,$zero,L1   # if n>=1 goto L1
1024         addi $v0,$zero,1    # return 1
1028         addi $sp,$sp,8
1032         jr   $ra
1036   L1:   addi $a0,$a0,-1
1040         jal  fact           # call fact with (n-1)
1044         lw   $a0,4($sp)
1048         lw   $ra,0($sp)
1052         addi $sp,$sp,8
1056         mul  $v0,$a0,$v0    # return n*fact(n-1)
1060         jr   $ra
```

Funzione ricorsiva



Caller:

```
100 addi $a0,$zero,3
104 jal fact
108 ....
```

Calcolo del fattoriale
(seconda versione: esempio di uso del frame pointer)

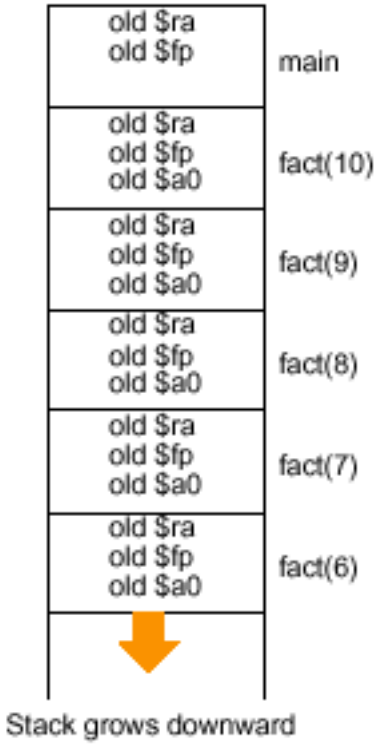
```
main()
   {
   printf("The factorial of 10 is %d \n",
   fact(10));
   }

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

Main

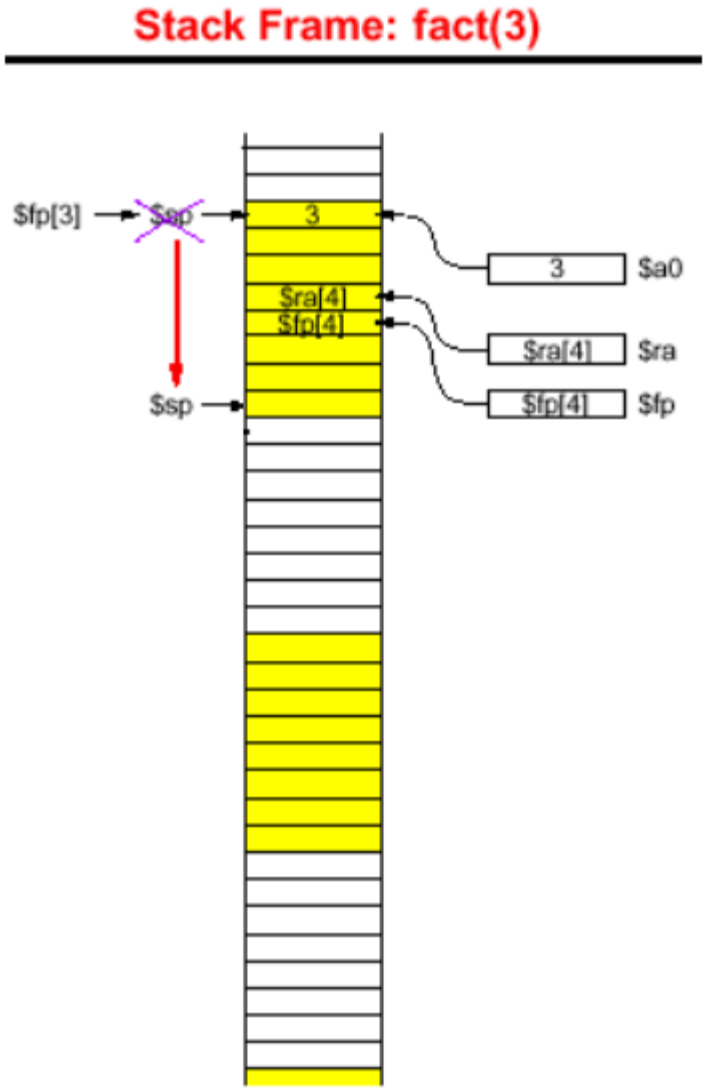
```
        .text
        .globl main
main:
        subu $sp,$sp,32          # Il frame di stack è di 32 byte
        sw $ra,16($sp)           # Salva l'indirizzo di ritorno
        sw $fp,12($sp)           # Salva il veccio fp
        addiu $fp,$sp,28         # Imposta il fp
        li $a0,10                # Mette 10 in $a0
        jal fact                 # Chiama fact
        la $a0,LC                # Mette la stringa di formato in $a0
        move $a1,$v0             # Copia il risultato di fatt in $v0
        jal printf               # Stampa
        lw $ra,16($sp)           # Ripristina $ra
        lw $fp,12($sp)           # Ripristina $fp
        addiu $sp,$sp,32         # Rimuove il frame di stack
        jr $ra                   # Ritorna al chiamante
.data
        LC:.asciiz "The factorial of 10 is %d \n"
```

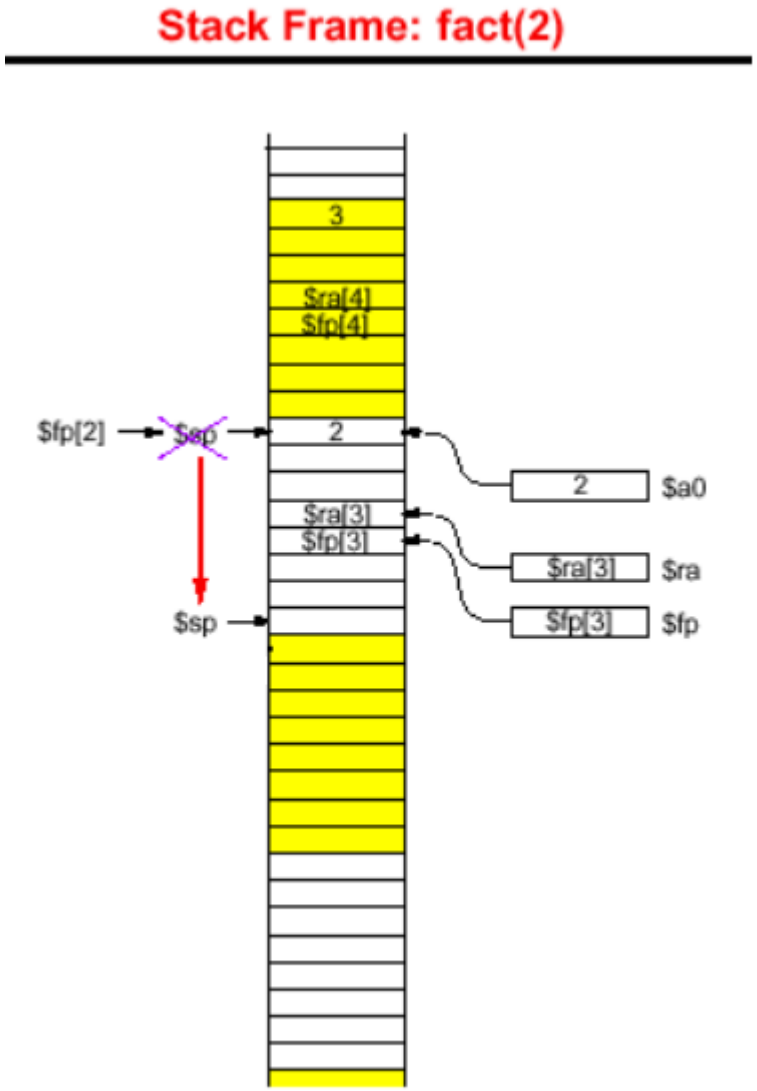
Fact

```
        .text
fact:subu $sp,$sp,32
     sw $ra,16($sp)
     sw $fp,12($sp)
     addiu $fp,$sp,28
     sw $a0,0($fp)
     bgtz $a0,THEN
     li $v0,1
     j EXIT
THEN:subu $a0,$a0,1
     jal fact
     lw $a0,0($fp)
     mul $v0,$v0,$a0
EXIT:lw $ra,16($sp)
     lw $fp,12($sp)
     addiu $sp,$sp,32
     jr $ra
```

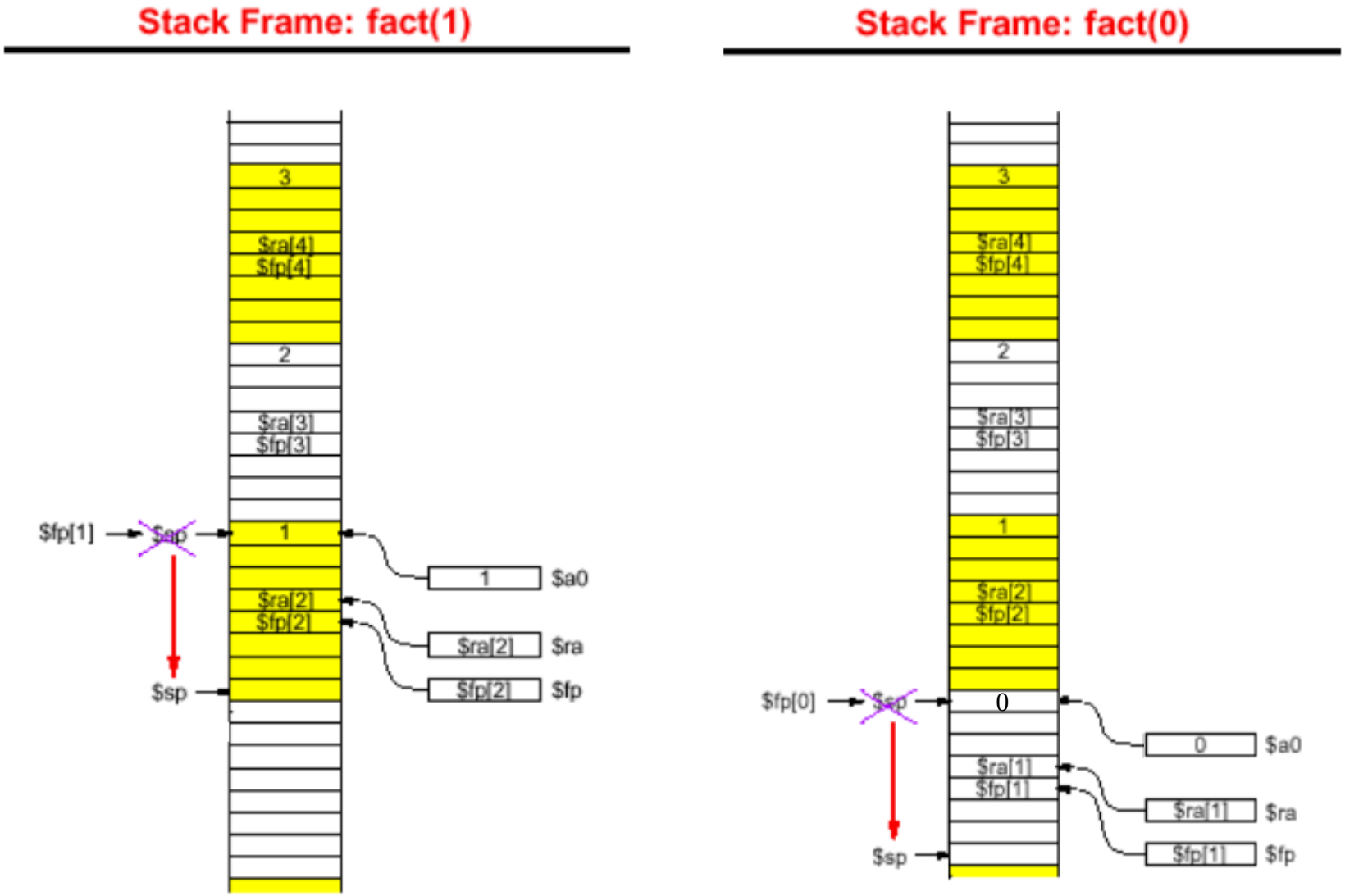


Evoluzione dello stack

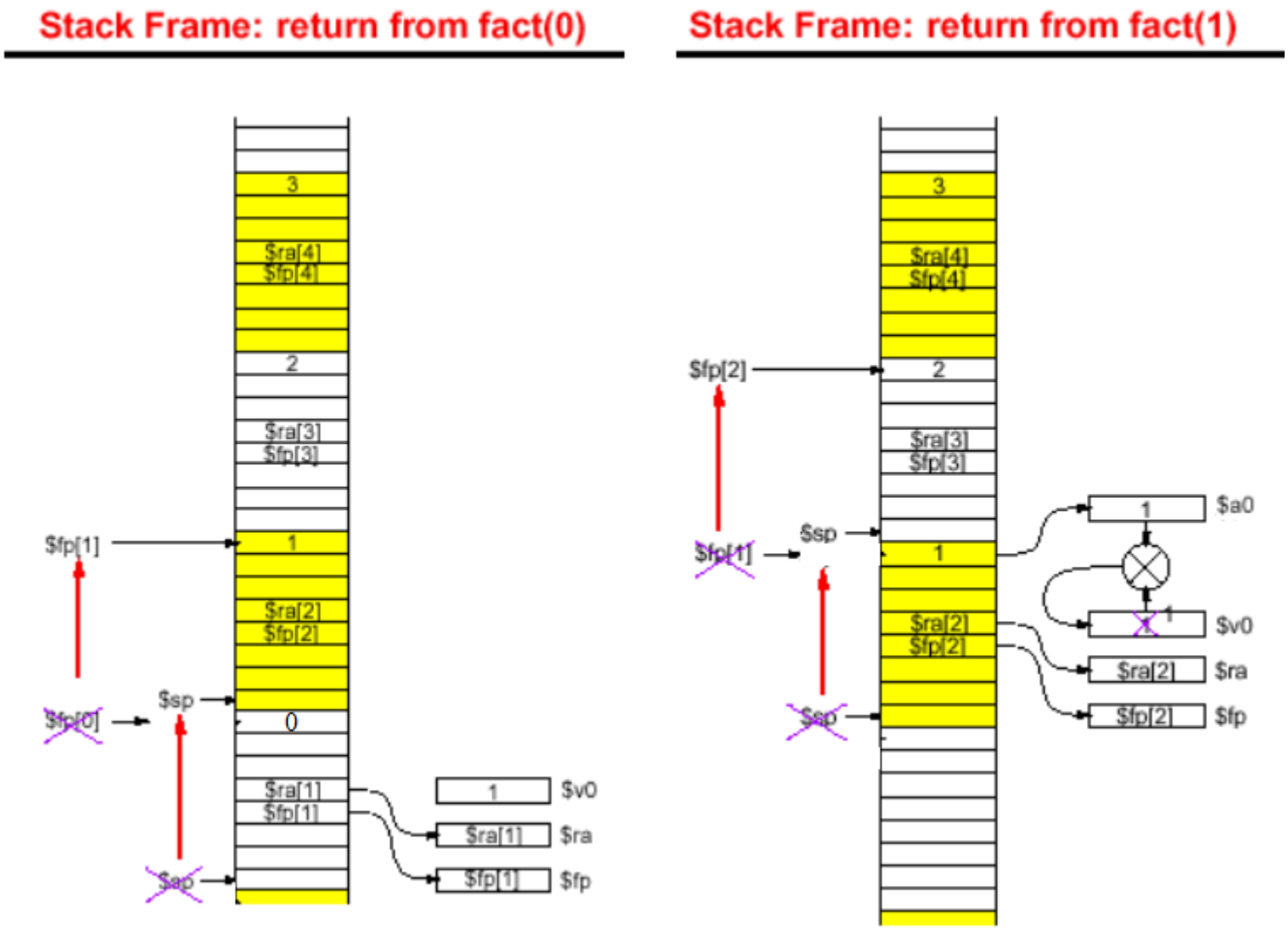




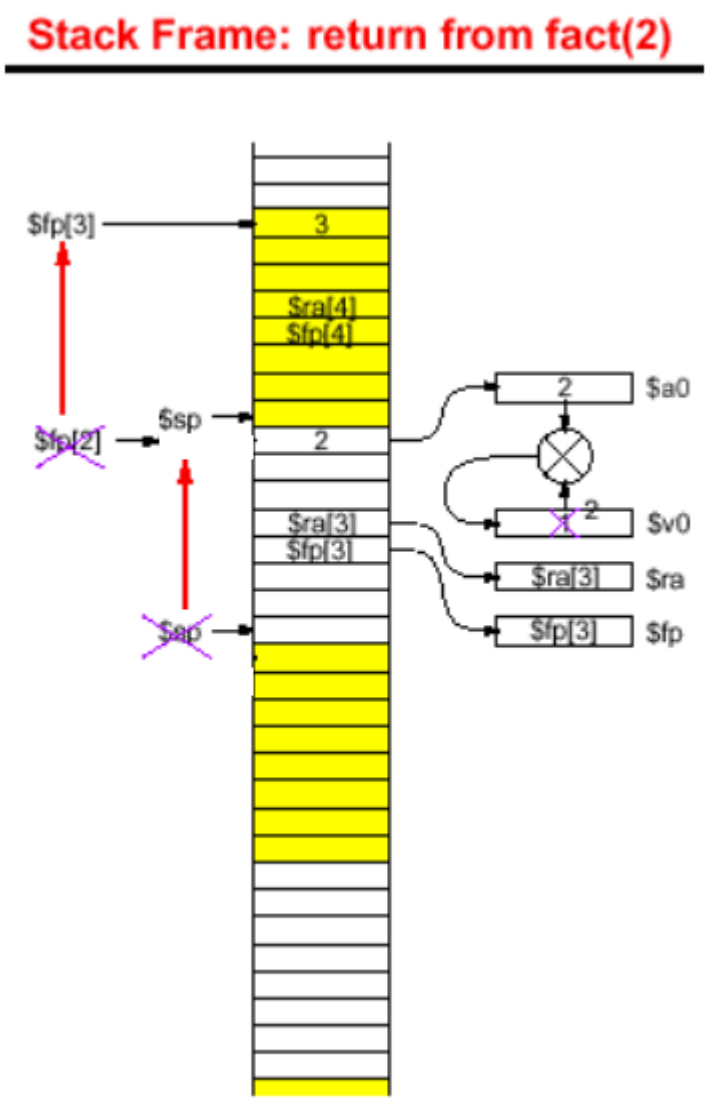
Evoluzione dello stack

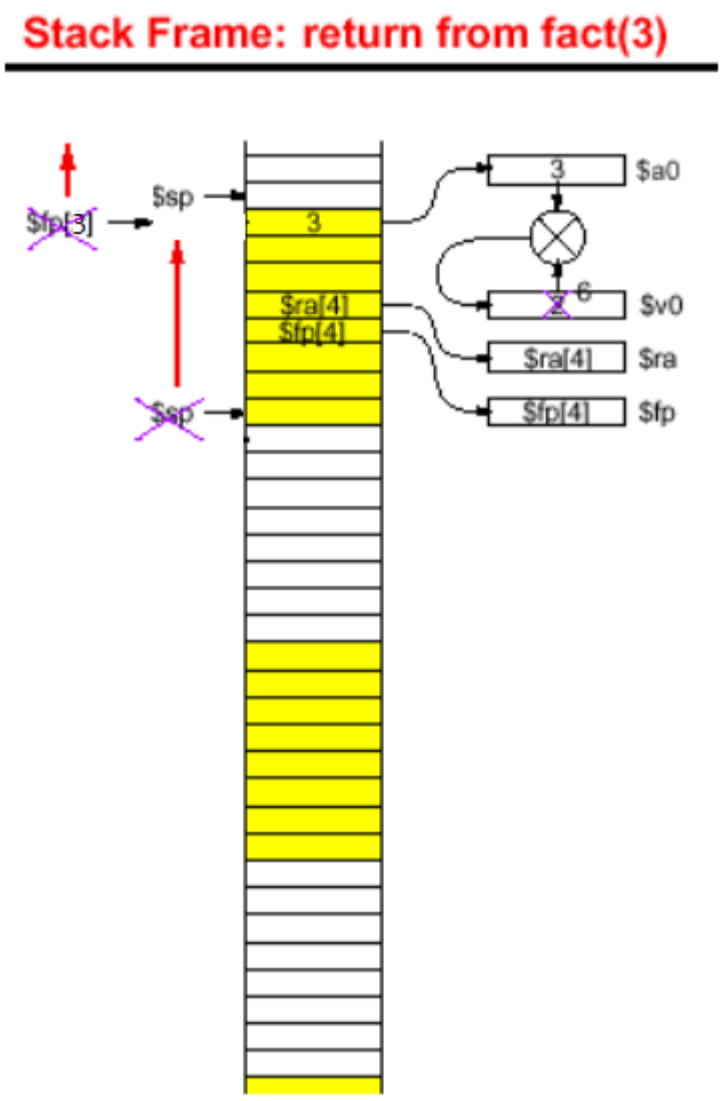


Evoluzione dello stack



Evoluzione dello stack





Suggerimenti per evitare errori comuni nelle chiamate a procedura

- **Il vostro main è anch'esso una procedura chiamata (c'e' un "jal main" nell'exception handler precaricato nel simulatore), quindi anche per il main devono essere rispettate le convenzioni fra chiamante e chiamato.**
- **Il chiamante ed il chiamato "comunicano" fra di loro attraverso i registri $ai e $vi. Questo significa che:**
  - se il chiamante vuole rendere disponibile al chiamato il contenuto di un certo registro (es. $s0) allora deve passarlo come argomento (es. move $a0, $s0).
  - se il chiamato vuole rendere disponibile al chiamante il contenuto di un registro (es. $s0) allora deve passarlo come risultato (es. move $v0, $s0).
- **Se il chiamante intende riutilizzare il contenuto di un registro non preservato (es. $t0) dopo una chiamata a procedura, prima della jal deve salvare sullo stack il contenuto di quel registro, indipendentemente dal fatto che il chiamato ne modifichi effettivamente il contenuto (non può averne garanzia).**