

**Architettura  
Elaboratori  
Elettronici  
ESERCITAZIONI  
STACK**

**Franco Liberati**  
**[liberati@di.uniroma1.it](mailto:liberati@di.uniroma1.it)**



# Argomenti

- ❑ STACK

- ❑ STACK NEL MARS





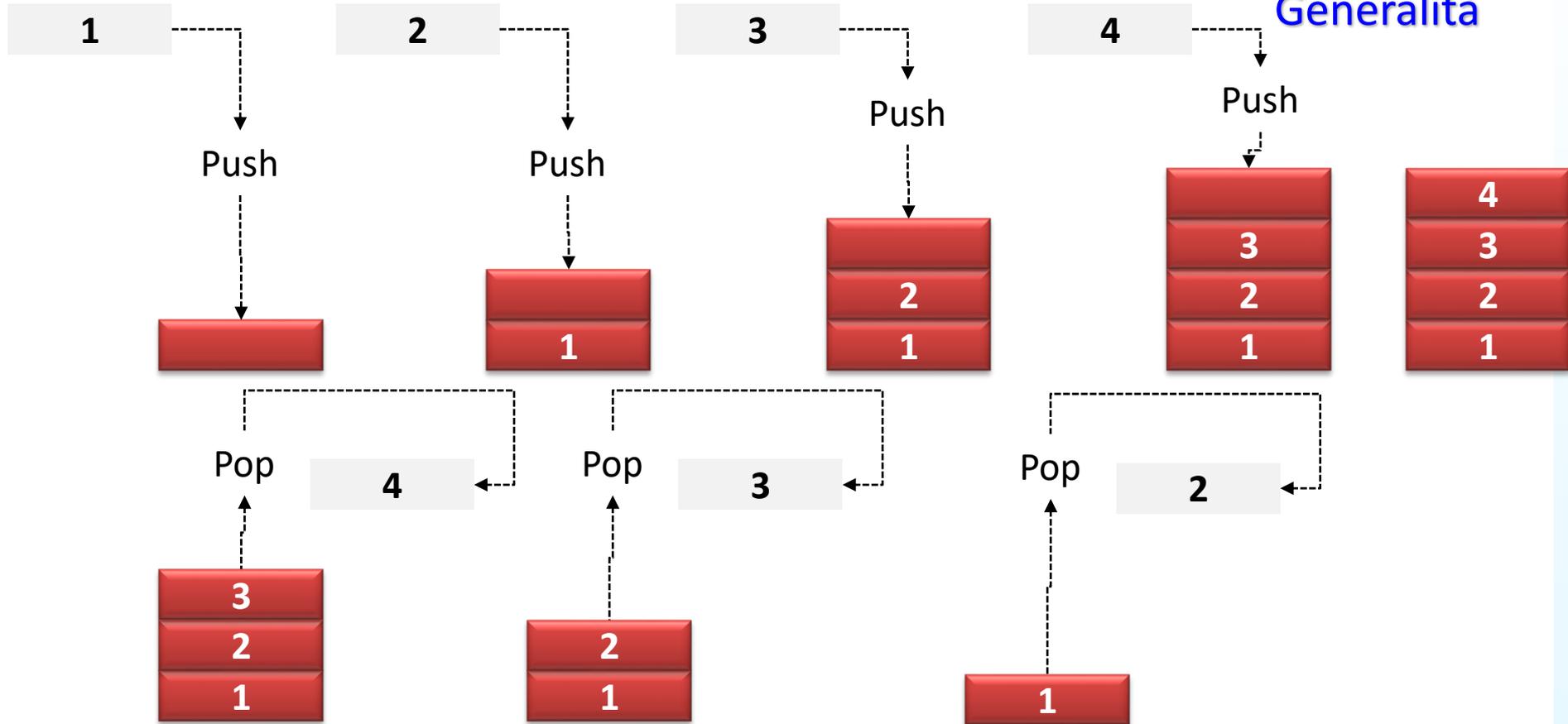
# STACK

## Generalità

- ❑ Lo **stack** (o **pila**) indica un tipo di dato astratto la cui modalità d'accesso ai dati in esso contenuti seguono la modalità LIFO (Last In First Out), ovvero tale per cui i dati sono estratti (letti) in ordine inverso rispetto a quello in cui sono stati inseriti (scritti)
  
- ❑ Sullo **stack** si interviene con tre operazioni:
  - ❑ **PUSH**: inserisce un elemento nello stack
  - ❑ **POP**: rimuove un elemento nello stack
  - ❑ **TOP**: legge l'elemento in cima allo stack (ma non lo rimuove)

# STACK

## Generalità





# STACK

## Generalità

- ❑ Lo **stack memory** è un'area di memoria contigua - di lunghezza predefinita - usata quando, ad esempio, si ha un cambio di stato dell'elaboratore (es.: una interruzione o nella multiprogrammazione) o, in generale, quando si devono memorizzare temporaneamente dei dati da dover elaborare in un successivo momento rispettando un ordine di tipo LIFO
- ❑ Nella stack memory di solito si salvano i registri all'interno della CPU, gli indirizzi di ritorno, gli argomenti delle funzioni,...

# STACK NEL MARS

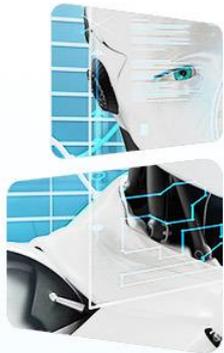




# STACK nel MARS

## Generalità

- Lo **stack memory in MARS** si usa nel caso del cambiamento di stato della macchina come anche nell'uso di funzioni ricorsive
- Di solito non si inseriscono nella pila dei singoli dati ma si agglomerano più informazioni (si creano dei frame di attivazioni dinamicamente, *stack frame*) che sono gestite in modalità LIFO



# STACK nel MARS

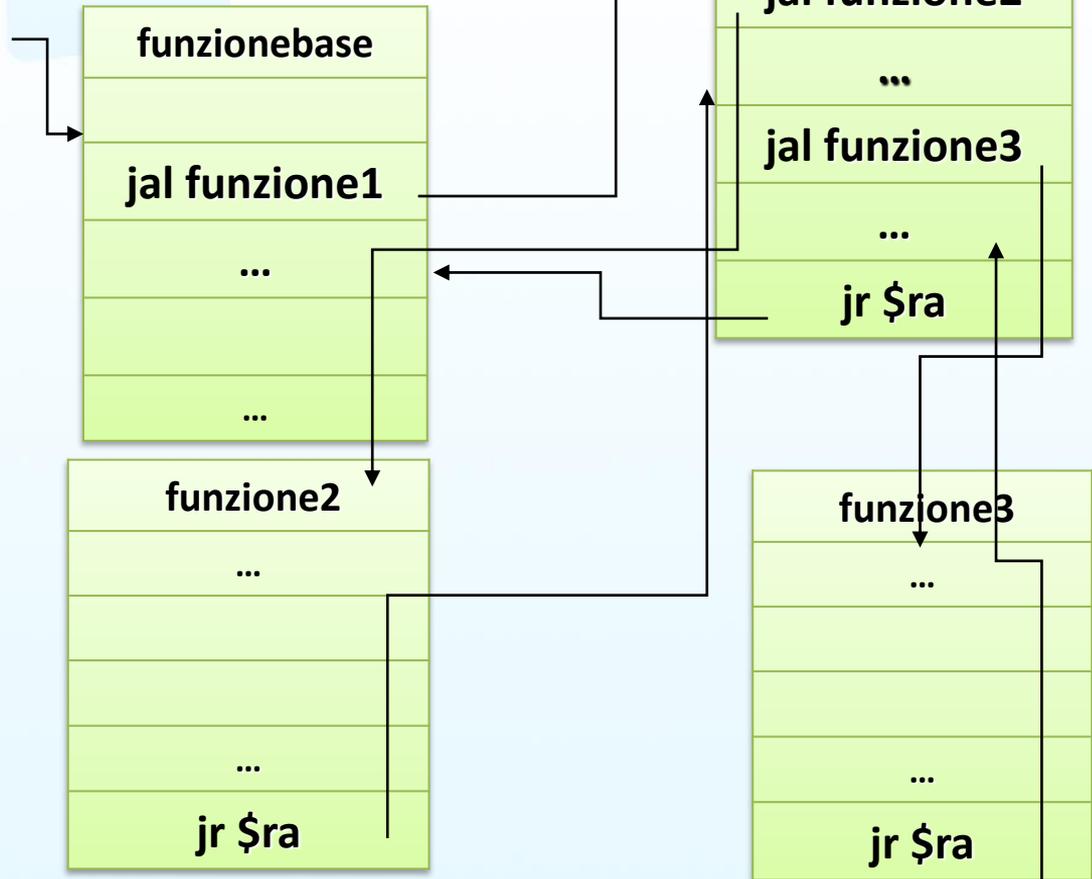
## Generalità

- ❑ Lo stack è accessibile da:
  - ❑ Il registro **stack pointer**,  $\$sp$ , che nel MIPS ha un valore iniziale prefissato a  $0x7ffeffc$
  - ❑ Lo stack cresce verso gli indirizzi più bassi della memoria (quindi per allocare spazio si deve sottrarre il valore del registro  $\$sp$ )





# STACK nel MARS

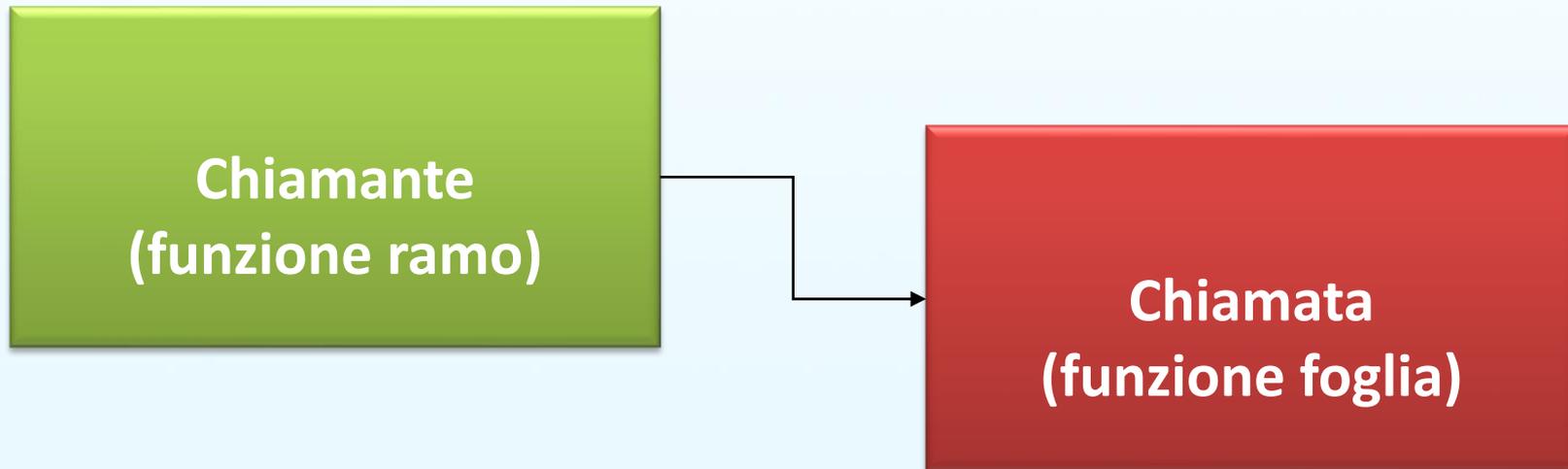




# STACK nel MARS

Norme generali per implementarlo

- ❑ Per implementare lo stack, di solito, si può procedere prendendo in considerazione la tipologia della funzione (se è foglia o ramo)





# STACK nel MARS

## Norme generali per implementarlo (ramo)

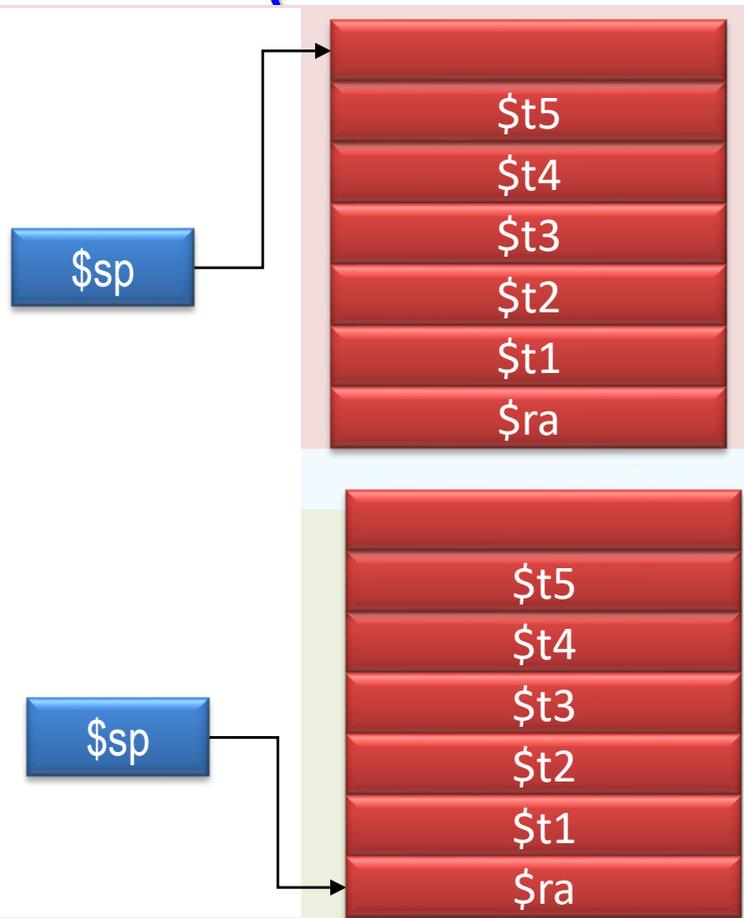
1. Si alloca nello stack uno spazio sufficiente (il *frame stack*) a contenere tutti i registri che devono essere salvati, le variabili locali e i parametri della funzione
  - ❑ Se si vogliono preservare i registri  $\$t0-\$t9$ , devono essere salvati prima della chiamata a funzione e devono essere ripristinati dopo
  - ❑ Se la funzione chiamata richiede più di 4 argomenti si può salvare 4 argomenti in  $\$a0, \$a1, \$a2, \$a3$  e i rimanenti nello stack (o salvarli tutti nello stack)
2. Si salva  $\$ra$
3. Si chiama la funzione
4. Si ripristinano i registri salvati  $\$ra, \$t0-\$t9$
5. Si libera lo spazio sullo stack allocato all'inizio



# STACK nel MARS (NON FOGLIA)

funzione\_nonfoglia:

```
subu $sp,$sp,24
sw $ra,20($sp)
sw $t1,16($sp)
sw $t2,12($sp)
sw $t3,8($sp)
sw $t4,4($sp)
sw $t5,0($sp)
jal funzione_foglia
lw $t5,0($sp)
lw $t4,4($sp)
lw $t3,8($sp)
lw $t2,12($sp)
lw $t1,16($sp)
lw $ra,20($sp)
addu $sp,$sp,24
jr $ra
```





# STACK nel MARS

Norme generali per implementarlo (foglia)

1. Si alloca nello stack uno spazio sufficiente a contenere tutti i registri che devo salvare e le eventuali variabili locali
  - Se si vogliono preservare i registri  $\$t0-\$t9$ , devono essere salvati prima della chiamata a funzione e ripristinati dopo
2. Si ripristinano i registri salvati  $\$t0-\$t9$ .
3. Si libera lo spazio sullo stack allocato all'inizio

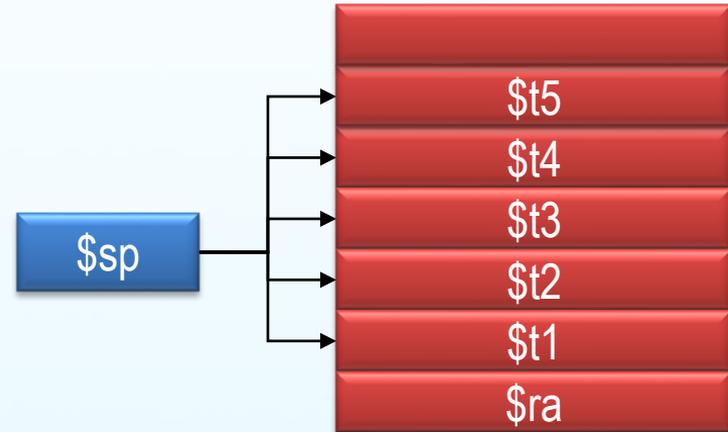
**Se non ci sono registri da memorizzare (che possono essere cambiati e rinviati alla funzione chiamante) non c'è bisogno di fare nulla e si può fare una normale chiamata a funzione**



# STACK nel MARS (FOGLIA)

funzione\_foglia:

```
lw $t5,0($sp)
lw $t4,4($sp)
lw $t3,8($sp)
lw $t2,12($sp)
lw $t1,16($sp)
...
sw $t5,0($sp)
sw $t4,4($sp)
sw $t3,8($sp)
sw $t2,12($sp)
sw $t1,16($sp)
jr $ra
```





# STACK nel MARS

## Esempio

Si consideri la funzione  $f$  fattoriale definita per valori interi  $n$

$$FATTORIALE(x) = x \cdot FATTORIALE(x-1)$$

$$FATTORIALE(1) = 1$$

$$FATTORIALE(0) = 1$$

Si realizzi un programma in assembly MIPS che, definito un intero positivo  $x \geq 2$ , calcola il corrispondente valore di  $FATTORIALE(x)$  in modo ricorsivo

**Esempio**

$$\begin{aligned} \text{fattoriale}(5) &= 5 \cdot \text{fattoriale}(4) = 5 \cdot 4 \cdot \text{fattoriale}(3) = 5 \cdot 4 \cdot 3 \cdot \text{fattoriale}(2) = 5 \cdot 4 \cdot 3 \cdot 2 \cdot \text{fattoriale}(1) \\ &= 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120 \end{aligned}$$



1 se  $x \leq 1$

$F(x) = x \cdot F(x-1)$  altrimenti

# STACK nel MARS

## Esempio

```
li $v0,5
syscall
move $a0,$v0
jal FATTORIALE
move $a0,$v0
li $v0,1
syscall
li $v0,10
syscall
```

### FATTORIALE:

```
li $t0,1
ble $a0,$t0, caso_base
subu $sp,$sp,8
sw $ra,0($sp)
sw $a0,4($sp)
sub $a0,$a0,1
jal FATTORIALE
lw $ra,0($sp)
lw $a0,4($sp)
addi $sp,$sp,8
mul $v0,$v0,$a0
jr $ra
```

```
caso_base:
li $v0,1
jr $ra
```



# STACK nel MARS

## Esempio

Si consideri la funzione  $f$  definita su interi

$$f(x) = f(x-1) - 1 \text{ se } x \text{ è multiplo di } 3$$

$$f(x) = f(x-1) + 1 \text{ se } x \text{ non è multiplo di } 3$$

$$f(1) = 1$$

Si realizzi un programma in assembler MIPS che, definito un intero positivo  $x \geq 2$ , calcola il corrispondente valore di  $f(x)$  in modo ricorsivo

$$f(5) = f(4) + 1 = f(3) + 1 + 1 = f(2) - 1 + 1 + 1 = f(1) + 1 - 1 + 1 + 1 = 1 + 1 - 1 + 1 + 1 = 3$$

$$f(6) = f(5) - 1 = f(4) + 1 - 1 = f(3) - 1 + 1 - 1 = f(2) + 1 - 1 + 1 - 1 = f(1) + 1 + 1 - 1 + 1 - 1 = 1 + 1 + 1 - 1 + 1 - 1 = 2$$



1 se  $x \leq 1$

$F(x) = F(x-1) - 1$  se  $x$  multiplo di 3

$F(x) = F(x-1) + 1$  altrimenti

**Funzione:**

```
li $v0,5
syscall
move $a0,$v0
jal Funzione
move $a0,$v0
li $v0,1
syscall
li $v0,10
syscall
```

```
li $t1,1
beq $a0, $t1, caso_base
subu $sp, $sp, 8
sw $a0, 0($sp)
sw $ra, 4($sp)
sub $a0, $a0, 1
jal Funzione
```

multiplo:

fine:

# STACK nel MARS

## Esempio

```
lw $a0, 0($sp)
lw $ra, 4($sp)
addi $sp, $sp, 8
li $t0, 3
rem $t1, $a0, $t0
beqz $t1, multiplo
add $v0,$v0,1
j fine
```

```
sub $v0, $v0, 1
```

```
jr $ra
```

caso\_base:

```
li $v0,1
jr $ra
```



# STACK nel MARS

## Esercizio proposto per casa

Si consideri la funzione  $f$  definita su interi

$$f(x) = f(x-2) - 2$$

$$f(1) = 14$$

$$f(0) = 10$$

Si realizzi un programma in assembler MIPS che, definito un intero positivo  $x \geq 2$ , calcola il corrispondente valore di  $f(x)$  in modo ricorsivo

**ESEMPIO:**

$$f(6) = f(4) - 2 = f(2) - 2 - 2 = f(0) - 2 - 2 - 2 = 10 - 2 - 2 - 2 = 4$$

$$f(5) = f(3) - 2 = f(1) - 2 - 2 = 14 - 2 - 2 = 10$$

FINE

