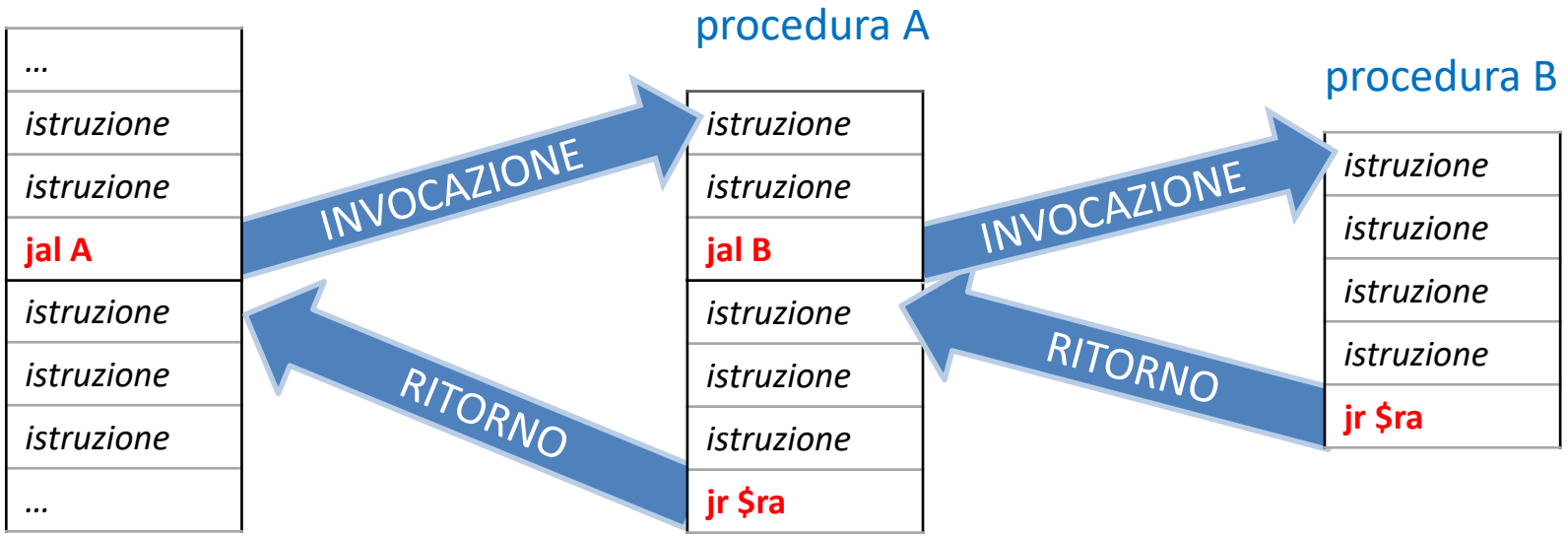
Università degli Studi di Milano
Dipartimento di Informatica “Giovanni Degli Antoni”
Corso di Laurea Triennale in Informatica

Architettura degli Elaboratori II Laboratorio

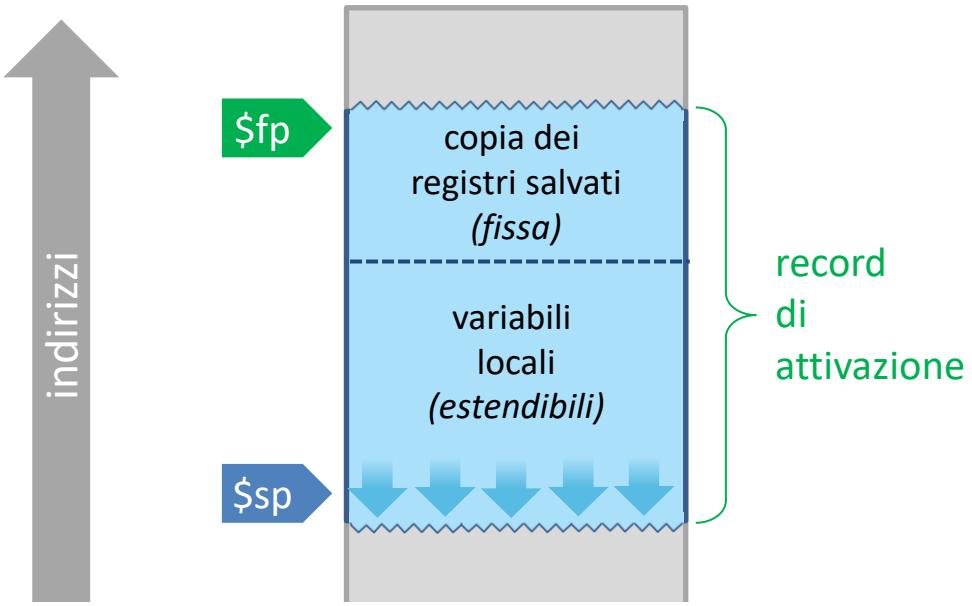
Procedure ricorsive

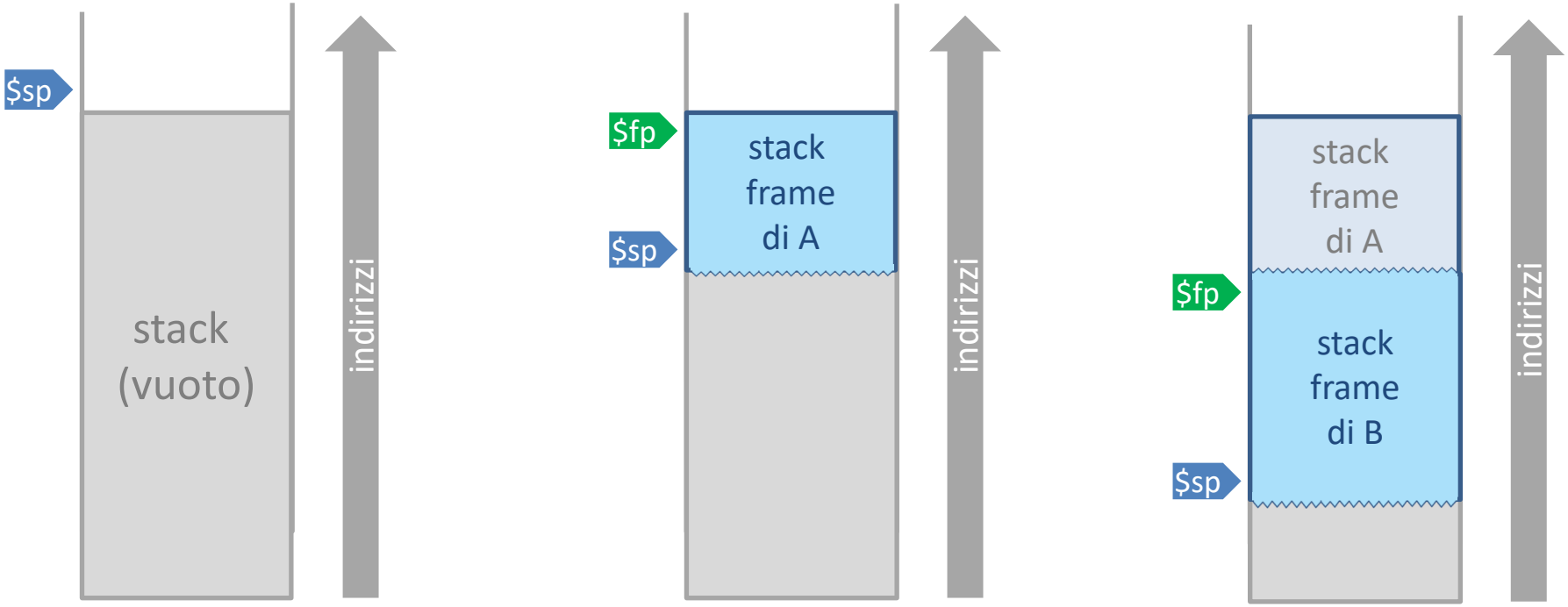
Invocazione di procedura annidate

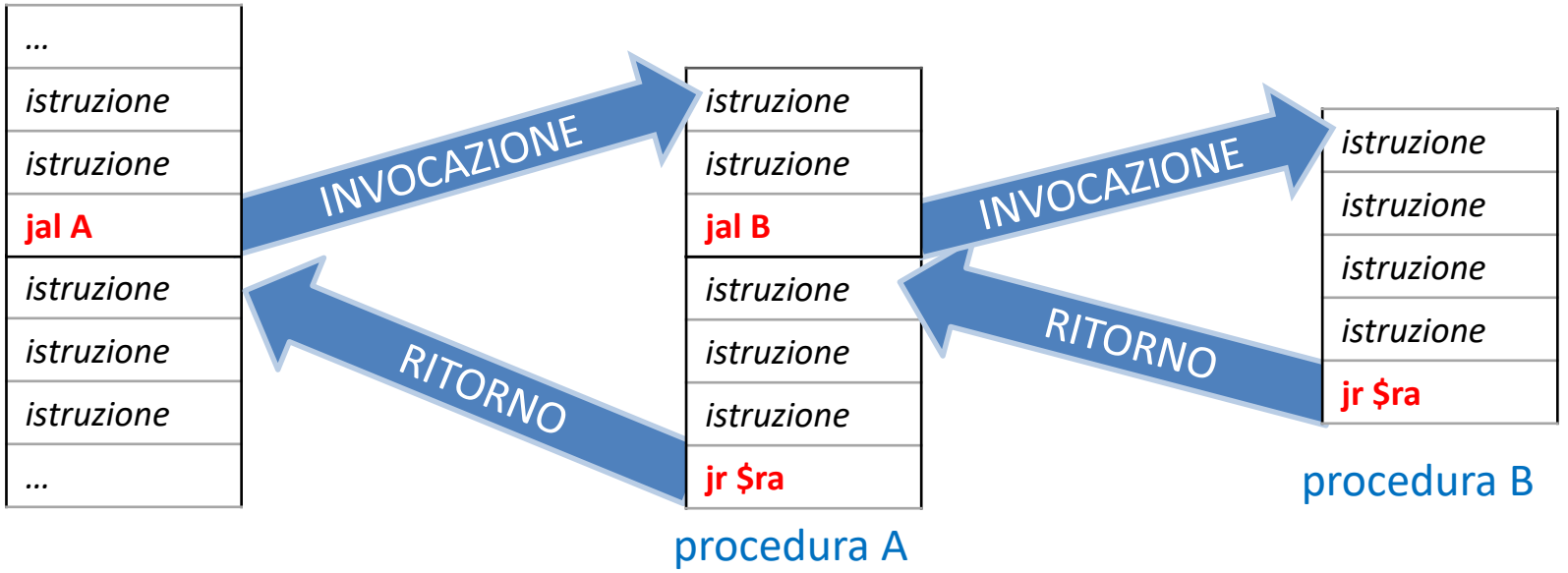


Il record di attivazione di una funzione

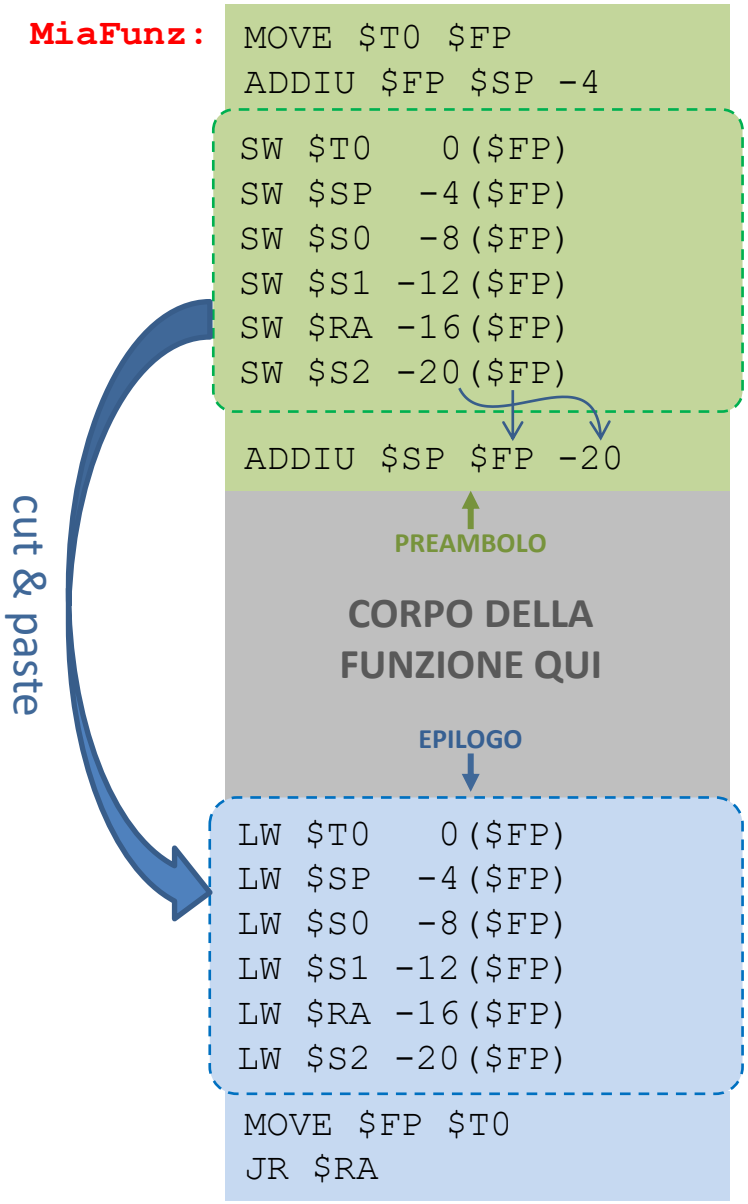
- Il record di attivazione di una procedura memorizza
  - La copia dei registri da preservare per il chiamante
  - Le variabili locali (attaverso push e pop, come normale)




$sp
stack
(vuoto)
indirizzi
$fp
stack
frame
di A
$sp
indirizzi
stack
frame
di A
$fp
stack
frame
di B
$sp
indirizzi


...
istruzione
istruzione
jal A
istruzione
istruzione
istruzione
...
INVOCAZIONE
RITORNO
istruzione
istruzione
jal B
istruzione
istruzione
istruzione
jr $ra
procedura A
INVOCAZIONE
RITORNO
istruzione
istruzione
istruzione
istruzione
jr $ra
procedura B

Guida pratica per funzioni non-foglia



Copia temporanea del Frame Pointer *iniziale* (in T0).

Il *nuovo* record di attivazione comincia subito dopo il vecchio.

I valori dei registri *iniziali* sono i salvati (in qualsiasi ordine) nel (nuovo) record di attivazione.
Compreso lo stack pointer SP, il Return Address RA, e anche FP stesso (sotto forma di T0)

Aggiornamento dello SP
(punta sempre all'ultimo elemento occupato dello stack)

La funzione può

- usare i registri S solo se sono stati salvati (qui: s0, s1, s2).
- invocare altre funzioni (quindi usando RA),
- allocare variabili nello stack (quindi usano SP).
- usare i registri T, sapendo che non vengono mantenuti dopo l'invocazione di funzione

Recupero valore iniziale di tutti i registri salvati, compreso lo SP (flush dello stack)

… e compreso il FP

Ritorno al chiamante

Ricorsione

- La risoluzione di un problema P è costruita sulla base della risoluzione di un sottoproblema di P

- Esempio classico: il fattoriale di n

$$n! = \prod_{k=1}^{n} k = n \prod_{k=1}^{n-1} k = n \times (n-1)!$$

- il fattoriale di n è uguale a n moltiplicato per il fattoriale di n-1, non vero se n=0! Serve una regola aggiuntiva

$$n! = \begin{cases} n \times (n-1)! & \text{if } n > 0 \\ 1 & \text{if } n = 0. \end{cases}$$

Ricorsione

- Applico la regola in cascata

$$n! = \begin{cases} n \times (n-1)! & \text{if } n > 0 \\ 1 & \text{if } n = 0. \end{cases}$$

$$4! = 4 \times (3)!$$

Ricorsione

- Applico la regola in cascata

$$n! = \begin{cases} n \times (n-1)! & \text{if } n > 0 \\ 1 & \text{if } n = 0. \end{cases}$$

$$4! = 4 \times (3)!$$

$$3! = 3 \times (2)!$$

Ricorsione

- Applico la regola in cascata

$$n! = \begin{cases} n \times (n-1)! & \text{if } n > 0 \\ 1 & \text{if } n = 0. \end{cases}$$

$$4! = 4 \times \underbrace{(3)!}$$

$$3! = 3 \times \underbrace{(2)!}$$

$$2! = 2 \times (1)!$$

Ricorsione

- Applico la regola in cascata

$$n! = \begin{cases} n \times (n-1)! & \text{if } n > 0 \\ 1 & \text{if } n = 0. \end{cases}$$

$$4! = 4 \times \underbrace{(3)!}$$
$$3! = 3 \times \underbrace{(2)!}$$
$$2! = 2 \times \underbrace{(1)!}$$
$$1! = 1 \times (0)!$$

Ricorsione

- Applico la regola in cascata

$$n! = \begin{cases} n \times (n-1)! & \text{if } n > 0 \\ 1 & \text{if } n = 0. \end{cases}$$

$$4! = 4 \times \underbrace{(3)!}$$
$$3! = 3 \times \underbrace{(2)!}$$
$$2! = 2 \times \underbrace{(1)!}$$
$$1! = 1 \times \underbrace{(0)!}$$
$$0! = 1$$

Ricorsione

- Applico la regola in cascata

$$n! = \begin{cases} n \times (n-1)! & \text{if } n > 0 \\ 1 & \text{if } n = 0. \end{cases}$$

$$4! = 4 \times \underbrace{(3)!}$$
$$3! = 3 \times \underbrace{(2)!}$$
$$2! = 2 \times \underbrace{(1)!}$$
$$1! = 1 \times \underbrace{(0)!}$$
$$0! = 1$$
$$= 4 \times 3 \times 2 \times 1 \times 1$$

Procedure ricorsive

- Procedura ricorsiva: è una procedura che per risolvere il problema P invoca se stessa per risolvere un sotto-problema di P

- In generale una procedura ricorsiva non è una procedura foglia: invoca se stessa per sua definizione

- Una procedura ricorsiva è:
  - Un callee: deve salvare i registri callee-saved ($s0, …, $ra, $fp)
  - Un caller: deve salvare i registri caller-saved ($t0, …, $a0, …, $v0, $v1)

- Al momento del ritorno dalla chiamata ricorsiva è necessario ripristinare il valore di `$ra`

Procedure ricorsive

Possono essere strutturate in diversi blocchi funzionali:

- Punto di ingresso
- Push sullo stack dei registri usati
- Check caso base / step ricorsivo
  - Caso base
  - Step ricorsivo
- Ripristino dei registri usati
- `jr $ra`

Università degli Studi di Milano
Dipartimento di Informatica “Giovanni Degli Antoni”
Corso di Laurea Triennale in Informatica

Architettura degli Elaboratori II Laboratorio