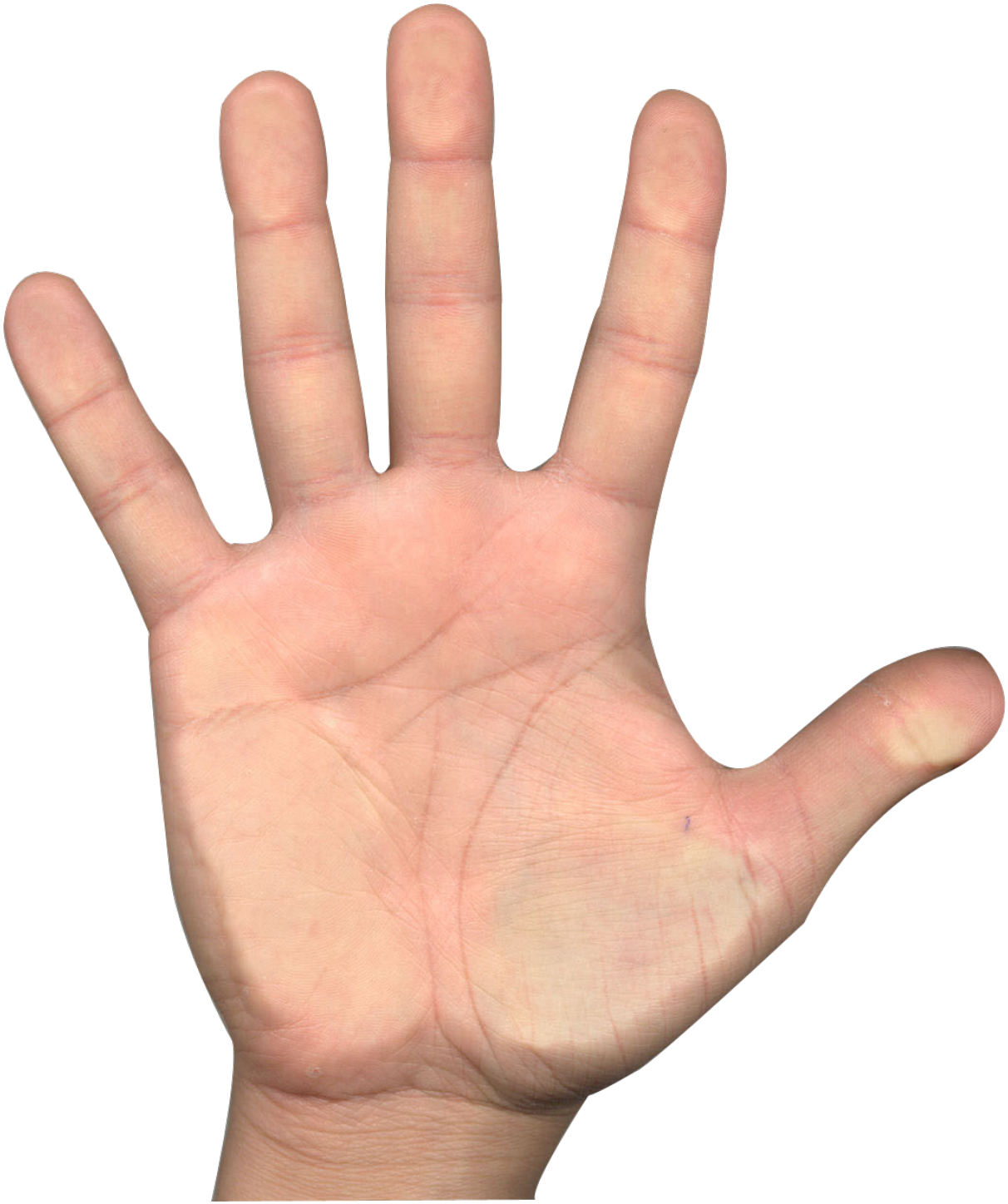
Ricorsione

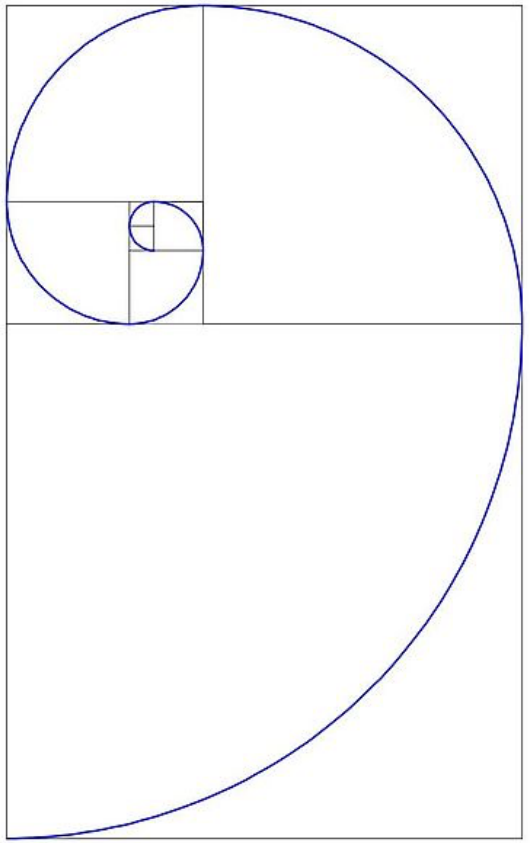


Numeri di Fibonacci

- Il rapporto tra le falangi del dito medio e anulare di un uomo adulto è pari al rapporto aureo f_n/f_{n-1} per $n \longrightarrow \infty$
- Il generico numbero di Fibonacci f_n è definito come $F = \{f_0, \ldots, f_n\}$: $f_0 = 0$ $f_1 = 1$
  - Per $n > 1$, $f_n = f_{n-1} + f_{n-2}$:
  - $f_2 = f_1 + f_0 = 1 + 0 = 1$
  - $f_3 = f_2 + f_1 = 1 + 1 = 2$
  - $f_4 = f_3 + f_2 = 2 + 1 = 3$
  - …

Altri esempi…

- Quasi tutti i fiori hanno un numero di petali che corrisponde ad un numero di Fibonacci
  - Gigli: 3 petali
  - Ranuncoli: 5 petali
  - …
  - Calendula: 13 petali
  - Margherite: 34, 55, o 89 petali
- In economia, modelli basati sulle sequenze di Fibonacci sono usati per calcolare andamenti futuri di azioni e obbligazioni (“onda di Elliot”)



Come calcolarlo?

- Assai spesso, per trovare la soluzione di un problema in un caso complicato possiamo usare la soluzione dello stesso problema in un caso più semplice

```
int fibonacci(int n) {
    int ris;
    if (n == 0) ris = 0;
    else if (n == 1) ris = 1;
    else ris = fibonacci(n–1) + fibonacci(n–2);
    return ris;
}
```

Ricorsione

- Un sottoprogramma P può chiamarne un altro Q, il quale a sua volta ne può chiamare un terzo R, e se R chiamasse nuovamente P (**ricorsione indiretta**)?
- Oppure P chiamasse se stesso durante la propria esecuzione (**ricorsione diretta**)?
- Ricorsione non significa chiamare un sottoprogramma più volte (scontato), ma richiamarlo -**da se stesso o altri sottoprogrammi**- prima della terminazione della sua esecuzione
- Ha un senso tutto ciò?
  - Se sottoprogramma significa “parte di programma per risolvere un sottoproblema”, richiamare P durante la sua esecuzione significa cercare di usare la soluzione del sottoproblema PR che si intende risolvere con P per risolvere PR….

Altri esempi di definizioni ricorsive

- La lista inversa L^{-1} di una lista di elementi $L = \{a_1, ..., a_n\}$ è così definita:
  - Se n = 1, $L^{-1} = L$, altrimenti $L^{-1} = \{a_n, (L_{n-1})^{-1}\}$ dove L_{n-1} indica la lista ottenuta da L cancellando l'ultimo elemento an
  - Per esempio, la lista inversa di $L = \{2,7,5,4\}$ è ottenibile come:

    $\{2,7,5,4\}^{-1} = \{4, \{2,7,5\}^{-1}\} = \{4,5, \{2,7\}^{-1}\} =$
    $\{4,5,7, \{2\}^{-1}\} = \{4,5,7,2\}$

- Il determinante D di una matrice quadrata A di ordine n è così definito:
  - Se n = 1, allora $D(A) = a_{11}$; altrimenti $D(A) = (-1)^{i+1} \cdot a_{1i} \cdot D(A_{1i})$, dove A_{1i} indica la matrice ottenuta da A eliminando la prima riga e la i-esima colonna

Ricorsione nella programmazione

- Passare da una formulazione ricorsiva del problema alla scrittura del codice semplice
- Calcolo del fattoriale (versione iterativa)

```
int  fatt(int n) {
        int  i, ris;
        ris = 1; i=1;
        while (i <=n) {
              ris = ris * i;
              i=i+1;
        }
        return ris;
}
```

Versione ricorsiva

- Se sfruttiamo la proprietà per cui n! = n (n–1)!, con 0! = 1 per convenzione

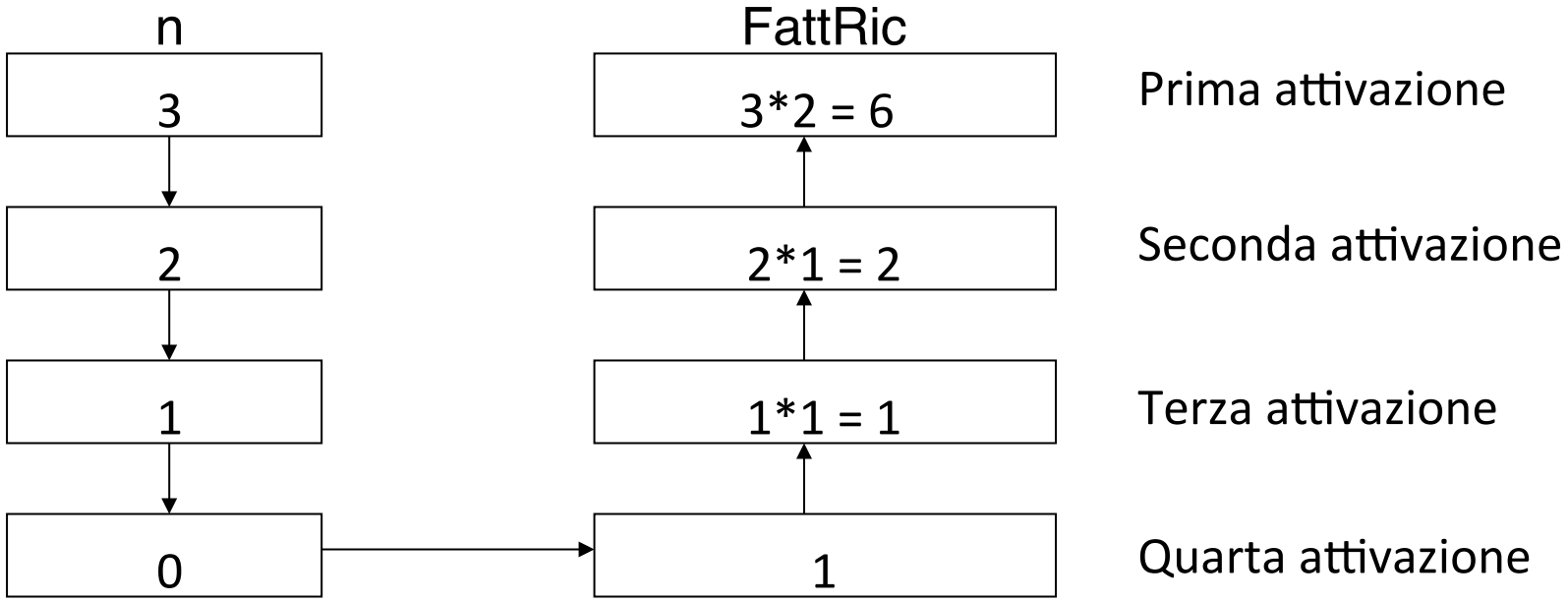
```
int FattRic(int n) {
    int ris;
    if (n == 0) ris = 1;
    else ris = n * FattRic(n–1);
    return ris;
}
```

Esecuzione

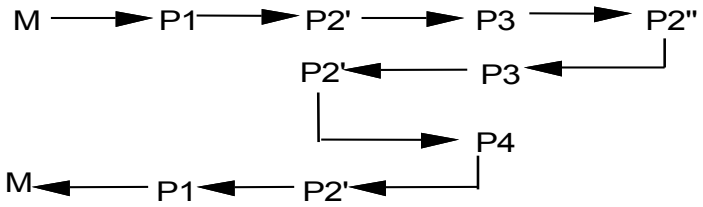
- Ad esempio, il fattoriale di 3:
  - 3 = 0? No: quindi occorre calcolare il fattoriale di 2, e, una volta ottenutolo, moltiplicarlo per 3
  - .2 = 0? No: quindi occorre calcolare il fattoriale di 1, e, una volta ottenutolo, moltiplicarlo per 2
  - .1 = 0? No: quindi occorre calcolare il fattoriale di 0, e, una volta ottenutolo, moltiplicarlo per 1
  - .0 = 0? Sì: quindi il fattoriale di 0 è 1
  - Sulla base di quanto stabilito al punto 3, il fattoriale di 1 è 1 moltiplicato per il fattoriale di 0, cioè $1 \times 1 = 1$
  - Sulla base di quanto stabilito al punto 2, il fattoriale di 2 è 2 moltiplicato per il fattoriale di 1, cioè $2 \times 1 = 2$

Memoria

- L'introduzione della ricorsione richiede una piccola rivoluzione nel meccanismo di gestione della memoria
- Invece che assegnare un area dati a ogni sottoprogramma, se ne assegna una a ogni chiamata di sottoprogramma



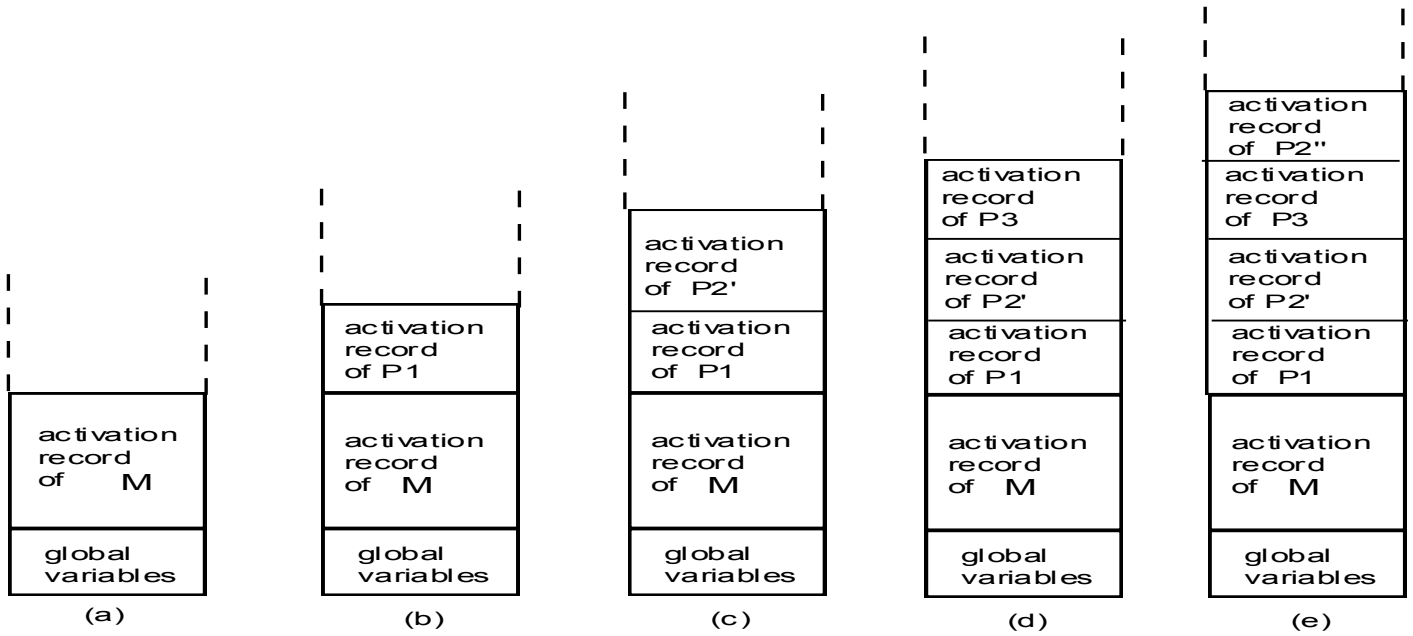
La gestione a pila o stack



Legend

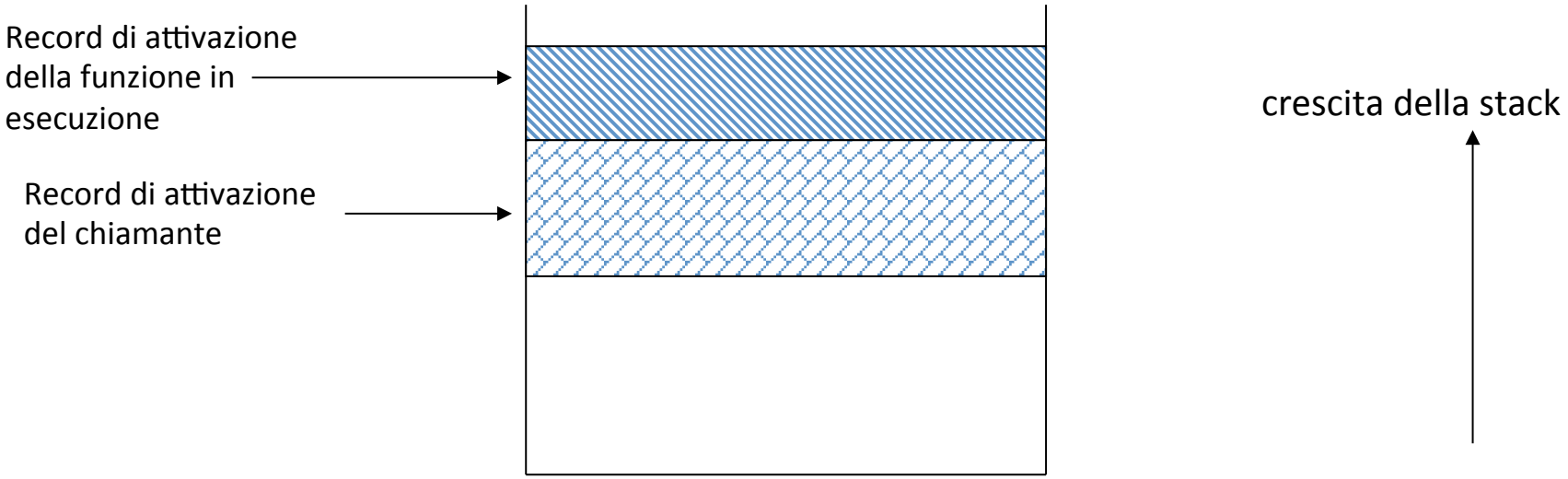
A rightwards arrow denotes a call
A leftwards arrow denotes the return to the calling subprogram

The different activations of P2 are marked by apexes

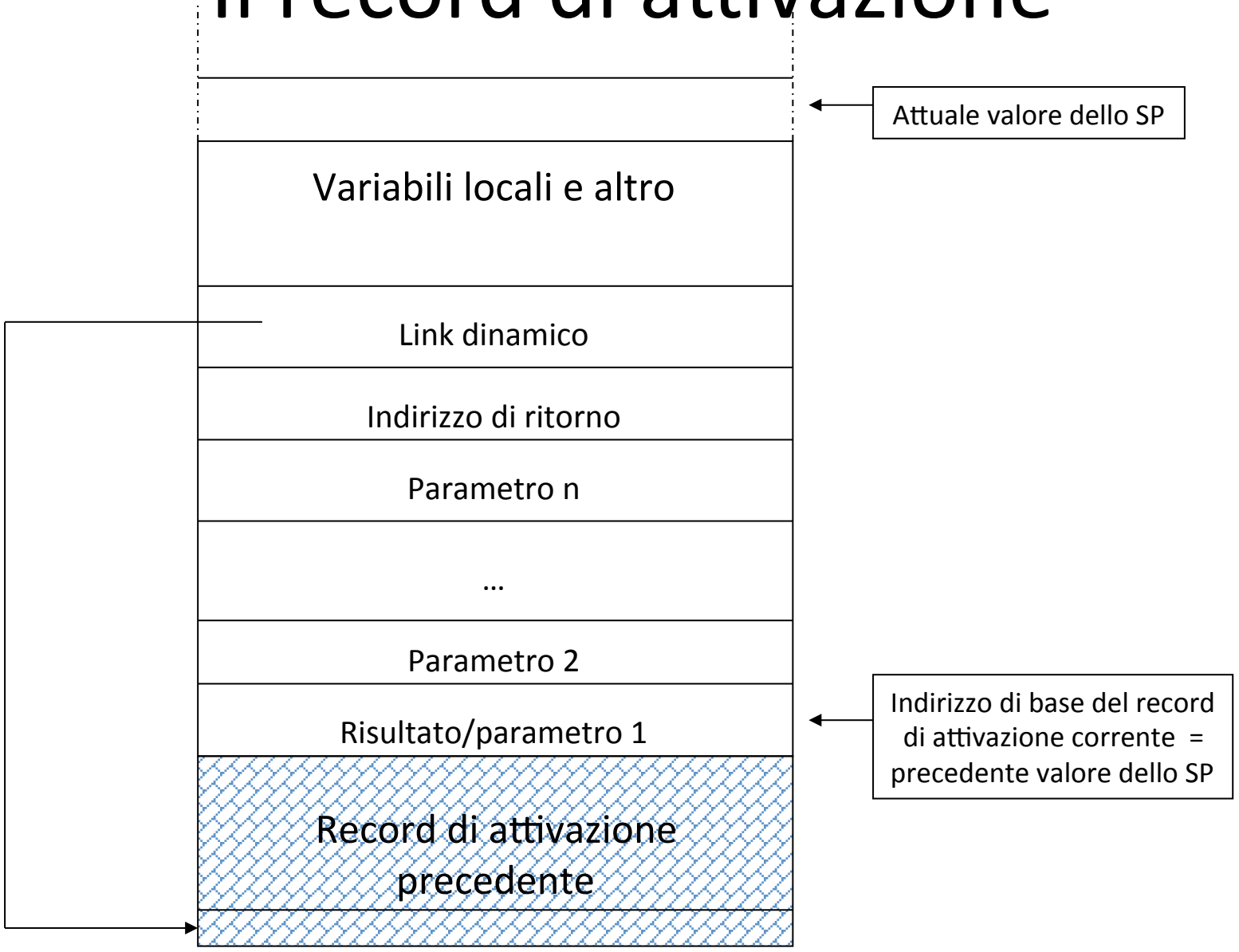


Dettagli su gestione stack

- Il record di attivazione:
  - **Parametri attuali**
  - **Variabili locali**
  - **Indirizzo di ritorno** (RetAdd)
  - (Valore precedente dello) **stack pointer** (SP)



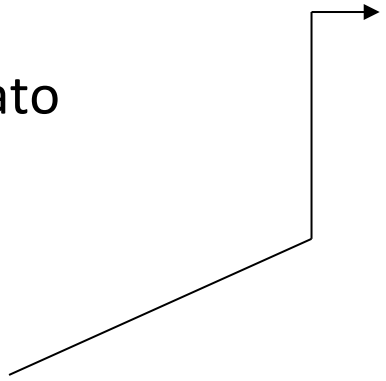
Il record di attivazione



La chiamata

Codice del chiamante

- Riserva memoria per il risultato (se previsto)
- Assegna valore ai parametri attuali
- Assegna l'indirizzo di ritorno
- Assegna il link dinamico
- Aggiorna l'indirizzo di base del nuovo record di attivazione
- Assegna il nuovo valore allo SP
- Salta alla prima istruzione del chiamato



Codice del chiamato

Il ritorno

Codice del chiamato

- Riporta il valore di SP al valore precedente
- Riporta il valore dell'indirizzo di base al valore precedente
- Salta all'indirizzo di ritorno

Codice del chiamante

- Codice della chiamata
- [Indirizzo di ritorno]
- Recupera eventuale risultato