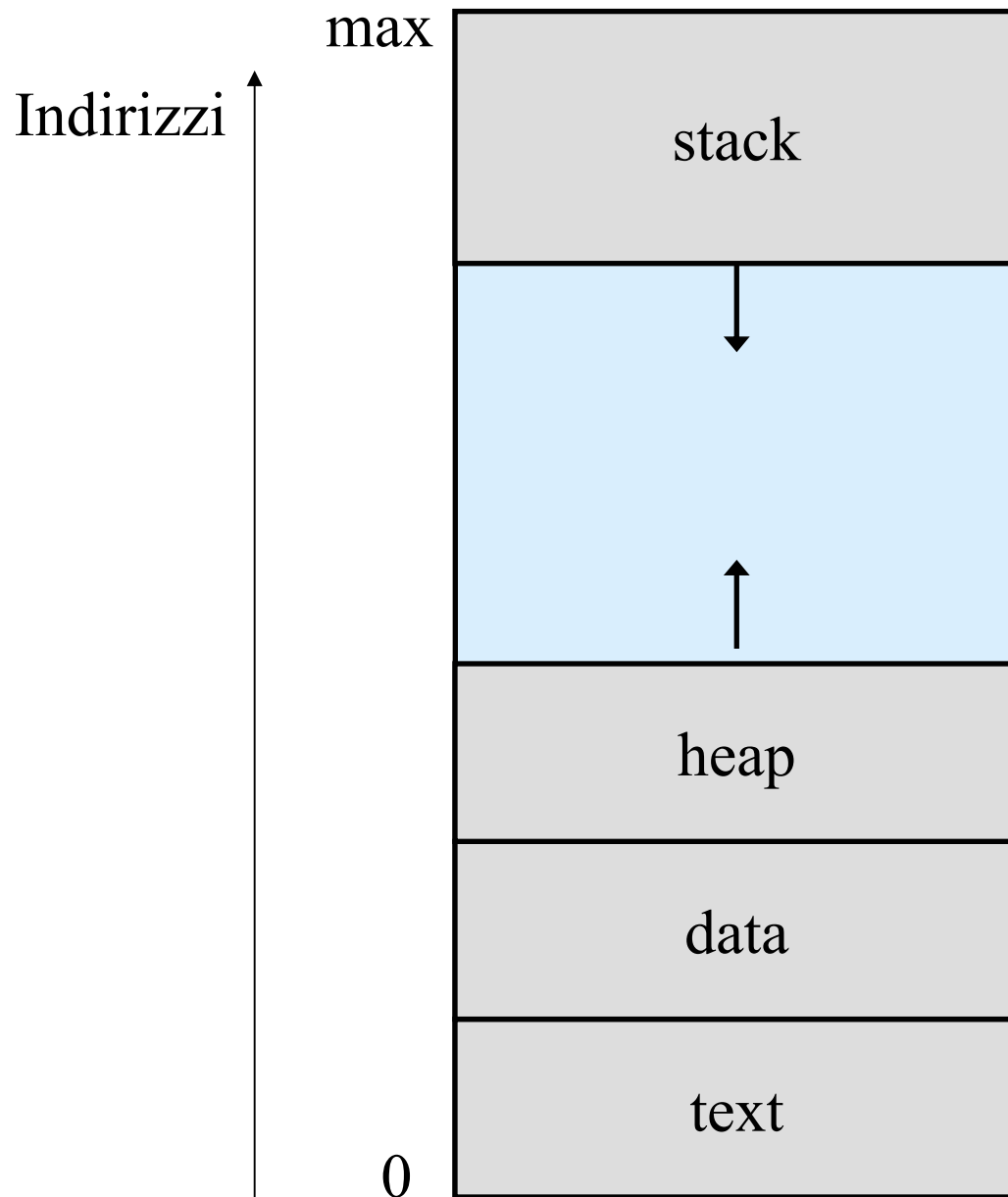


Organizzazione della Memoria usata dai Processi



La struttura dati Stack (o Pila)

- LIFO (last in - first out)

- Operazioni:

- Push (aggiunge un elemento in cima allo stack (che cresce verso gli indirizzi piu' piccoli))
 - Pop (toglie un elemento dalla cima dello stack)

- Condizioni d'Errore:

- Underflow (cercare di togliere un elemento quando lo stack è vuoto)
 - Overflow (cercare di aggiungere un elemento quando lo stack è già pieno)

- Nei processori, esiste un registro (Stack Pointer) dedicato a contenere l'indirizzo della cima dello stack.

Activation Records

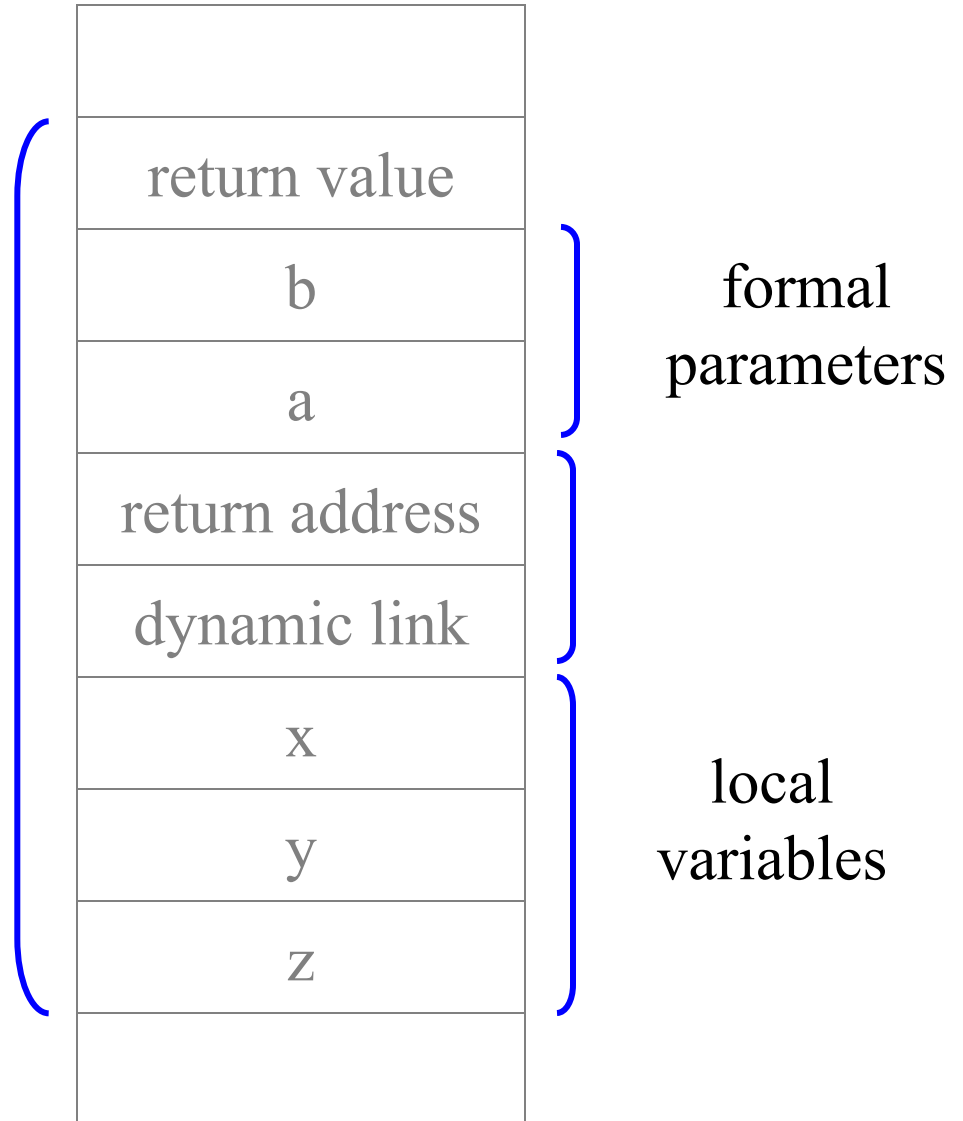
Records di Attivazione

- Activation Record

- È un insieme di bytes contigui in memoria trattati come un'unica entità
- Un diverso Activation Record viene allocato in memoria per ciascuna *invocazione di funzione (chiamata di funzione)*
- L'area di memoria in cui vengono collocati i record di attivazione è la *sezione di memoria* chiamata *stack*
- Un registro della CPU, lo *stack pointer (SP)*, punta alla cima dello stack – contiene l'indirizzo dell'ultimo byte occupato in cima allo stack, cioè l'**INDIRIZZO DI INIZIO DELLA PRIMAWORD OCCUPATA IN CIMA ALLO STACK.**
- Ogni elemento dello stack è un activation record.

Activation Record

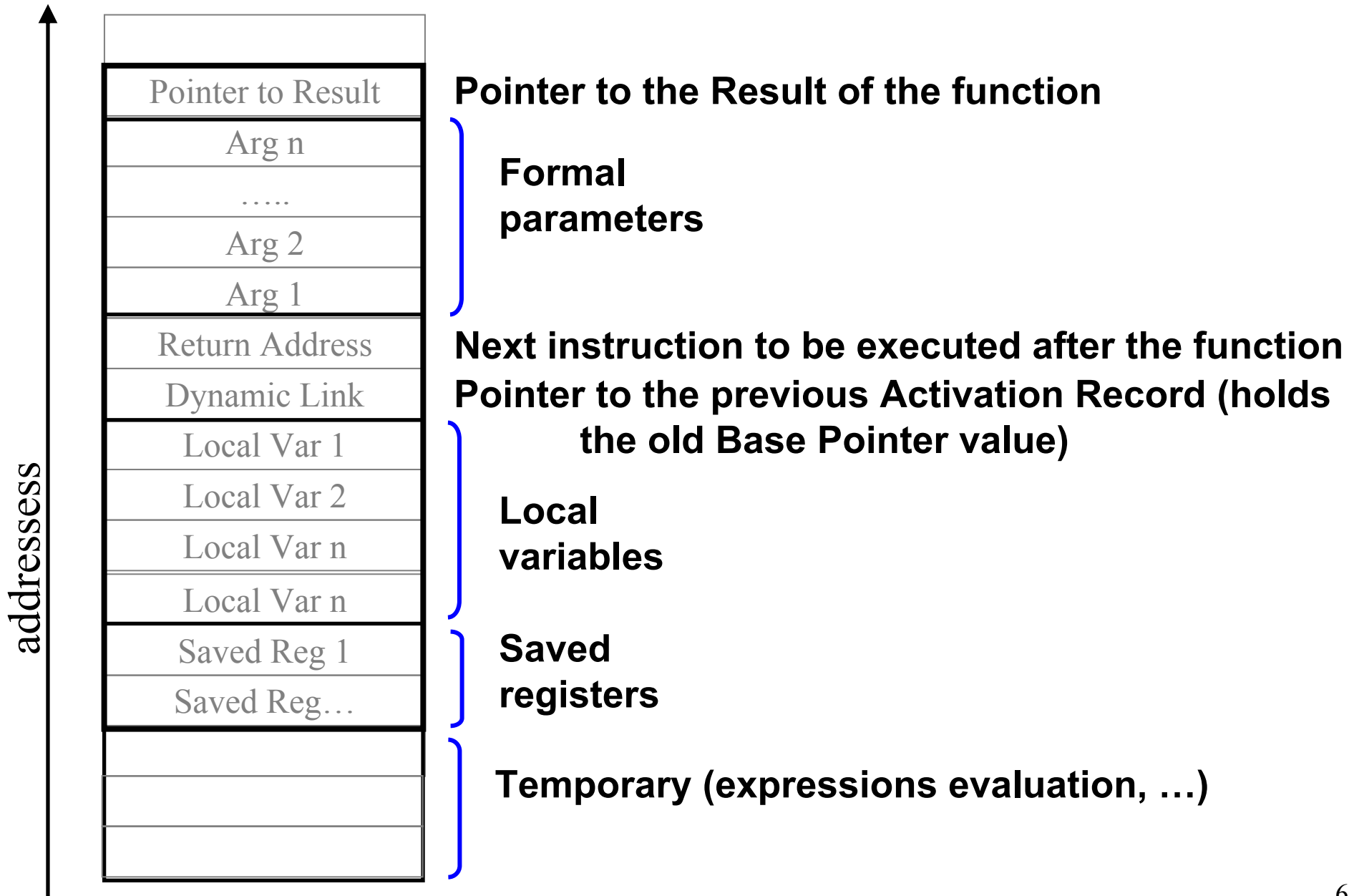
```
int funzione (int a, int b)
{
    int x, y, z;
    .....
    function body
    .....
    return z;
}
```



Activation Record – Calling Convention

- Il formato dei Record di attivazione dipende:
 - dal Processore su cui il codice esegue
 - dal compilatore che genera il codice e dai comandi impartiti al compilatore
 - dai tipi e dal numero di parametri formali della funzione
 - dal tipo di risultato che la funzione restituisce
 - Generalmente il risultato è restituito in uno o più registri
 - Se il risultato è troppo grande, viene collocato in un'area di memoria specificata dal chiamante mediante un puntatore messo sullo stack prima degli argomenti della funzione.

Activation Record – x86 - `__cdecl`



Example

```
int Chiamante (int32_t a, int32_t b)
{
    int32_t x, y, z;

    .....

    y = Chiamata(a);
    .....

    return y;
}
```

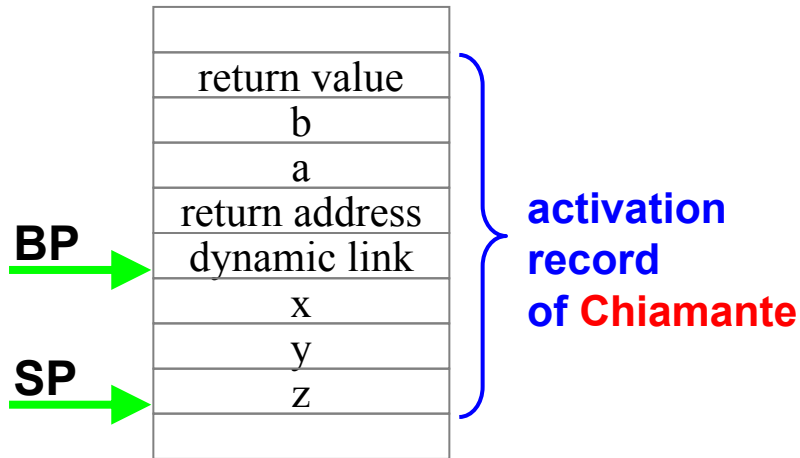
```
int Chiamata (int32_t a)
{
    int32_t x;

    .....
    body
    .....
    x = 9;

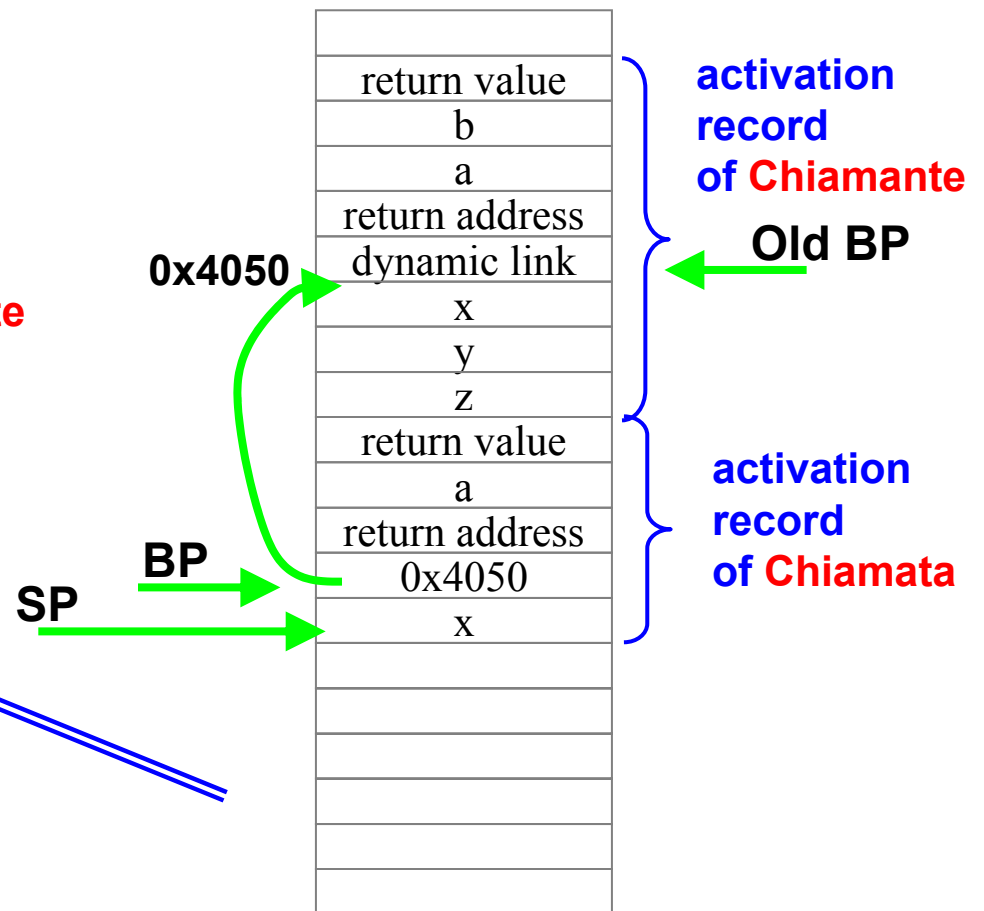
    return x;
}
```

Example - 2

1) **Chiamante** is executing



2) After **Chiamante** calls **Chiamata**



3) **Chiamata** terminates

address

Terminazione della funzione Chiamata

Chiamata:

```
..... body .....
```

```
; da qui il codice che termina la funzione
```

```
mov  sp, bp  
pop  bp
```

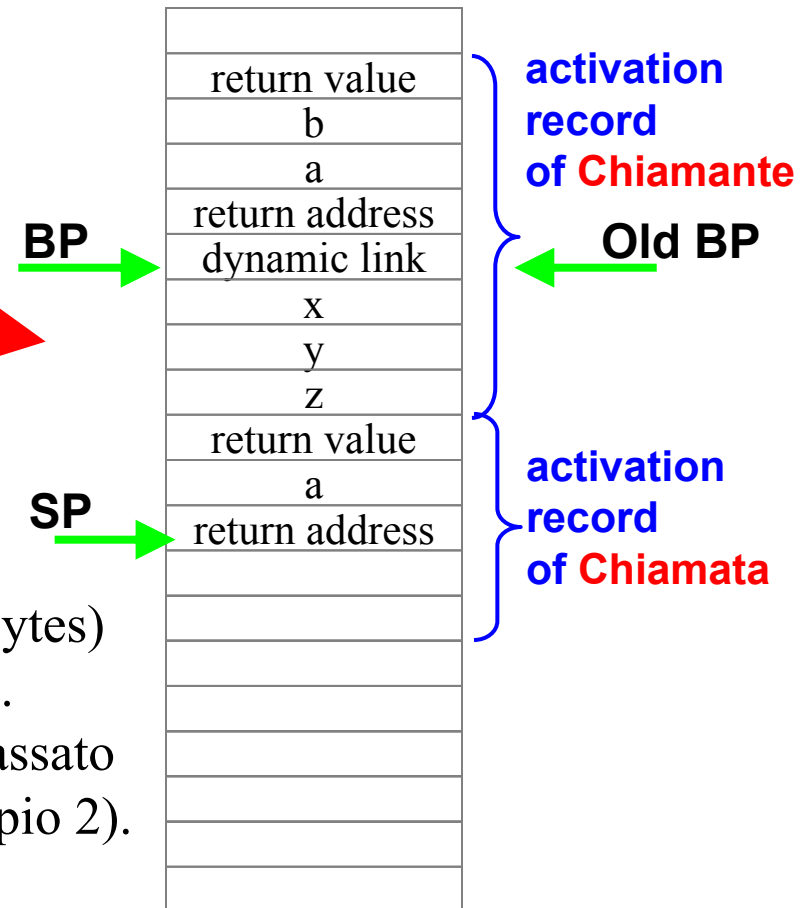
```
; A questo punto lo stack si presenta così
```

```
ret 2
```

```
; Carica dallo stack (pop) l'indirizzo della  
; prossima istruzione da eseguire  
; incrementando SP di 4 (dimensione dello  
; indirizzo "return address"),  
; poi incrementa SP dello spazio (espresso in bytes)  
; occupato dal parametro formale "a" (2 bytes).  
; La dimensione dei parametri formali viene passato  
; come argomento all'istruzione ret (nell'esempio 2).
```

IPOTESI:

- l'argomento a è un int16_t
- la word occupa 2 bytes
- gli indirizzi occupano 4 bytes



Chi carica sullo stack il record di attivazione della funzione Chiamata

