



UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dipartimento di Ingegneria
“Enzo Ferrari”

Fondamenti di Informatica II

Record di Attivazione

LA MACCHINA VIRTUALE DEL C

MODELLO A TEMPO DI ESECUZIONE

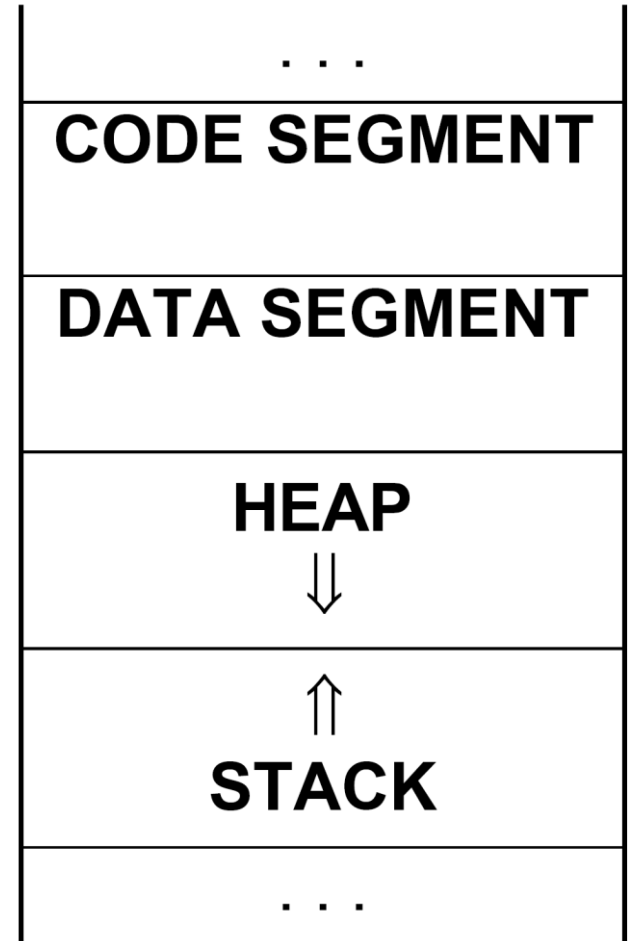
Aree di memoria:

Area del codice contiene il codice del programma

Area dati globali contiene le variabili globali e statiche

Area heap disponibile per allocazioni dinamiche

Area stack contiene i *record di attivazione* delle funzioni (variabili locali e parametri)



RECORD DI ATTIVAZIONE

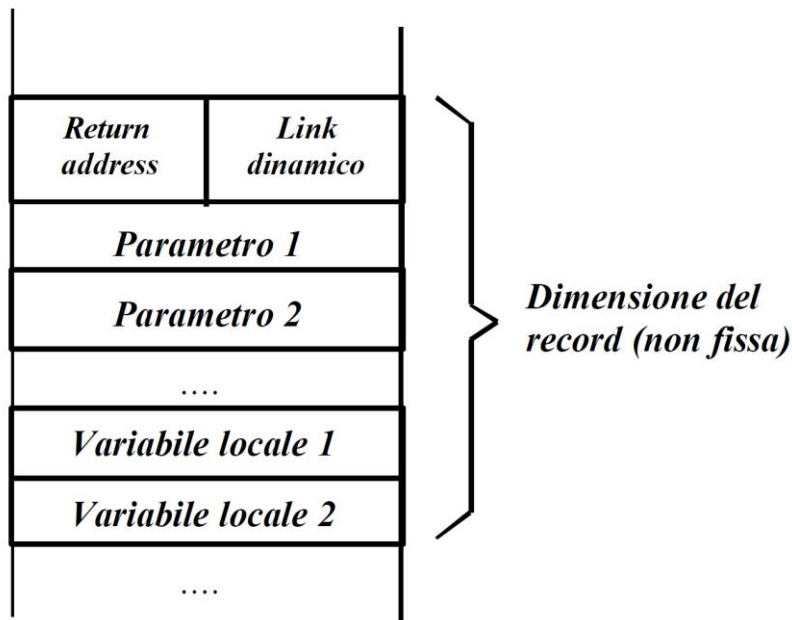
Record di Attivazione di una funzione

*Contiene tutto ciò che caratterizza l'esistenza di una funzione:
le variabili locali*

i parametri ricevuti

*l'indirizzo (del chiamante) a cui tornare al termine
dell'esecuzione*

un riferimento al record di attivazione del chiamante



Ad ogni attivazione di funzione viene creato un nuovo record di attivazione *specifico per quella chiamata*.

Per catturare la semantica delle chiamate annidate lo **stack** viene gestito con politica

LIFO (Last In First Out)

RECORD DI ATTIVAZIONE (2)

Dimensione

La dimensione del record di attivazione

- varia da una funzione all'altra
- ma, per una data funzione, è fissa e calcolabile a priori.

Creazione

Il record di attivazione:

- viene *creato dinamicamente* nel momento in cui la funzione viene chiamata
- rimane sullo stack per tutto il tempo in cui la funzione è in esecuzione
- viene *deallocato alla fine* quando la funzione termina.

Annidamento

Funzioni che chiamano altre funzioni danno luogo a una *sequenza* di record di attivazione gestiti secondo la politica LIFO:

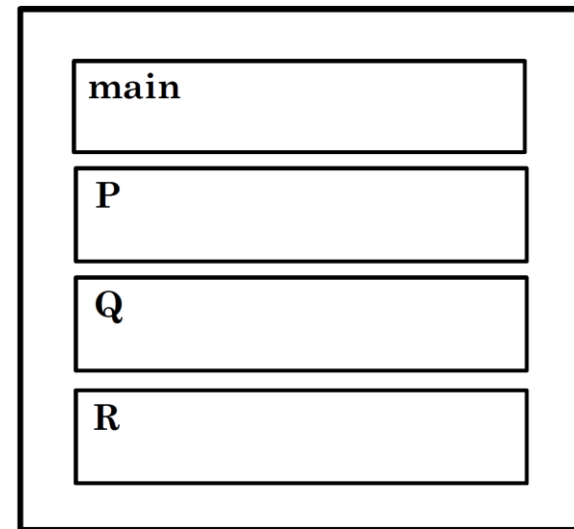
- allocati secondo l'ordine delle chiamate
- deallocati in ordine inverso

ESEMPIO 1

Dato il seguente programma:

```
void R(int a) {  
    int x = a; }  
void Q(int x) {  
    R(x); }  
void P() {  
    int a=10;  
    Q(a); }  
  
int main(void) {  
    P();  
    return 0;  
}
```

Le attivazioni sono le seguenti
(S.O.) → main → P → Q → R



Return Address e Link Dinamico

Quando la funzione termina, il controllo torna al chiamante, che deve riprendere la sua esecuzione dall'istruzione successiva alla chiamata della funzione.

A questo scopo, quando il chiamante chiama una funzione, nel suo record di attivazione si inseriscono:

l'indirizzo di ritorno (RTA) ossia l'indirizzo della prossima istruzione del chiamante che andrà eseguita quando la funzione terminerà

il link dinamico (DL), ossia un collegamento al RdA del chiamante, in modo da poter ripristinare l'ambiente del chiamante quando la funzione terminerà. Si costituisce così la **catena dinamica** che rappresenta *la storia* delle attivazioni.

PARAMETRI e RISULTATO

Passaggio dei parametri

Per i parametri *passati per valore*, nella cella del record di attivazione viene copiato *il valore* assunto dal parametro attuale all'atto della chiamata.

Per i parametri *passati per riferimento*, nella cella del record di attivazione viene copiato *l'indirizzo* della variabile che costituisce il parametro attuale all'atto della chiamata.

Restituzione del risultato

Per le *funzioni*, spesso il record di attivazione prevede anche una ulteriore cella, destinata a *contenere il risultato* della funzione.

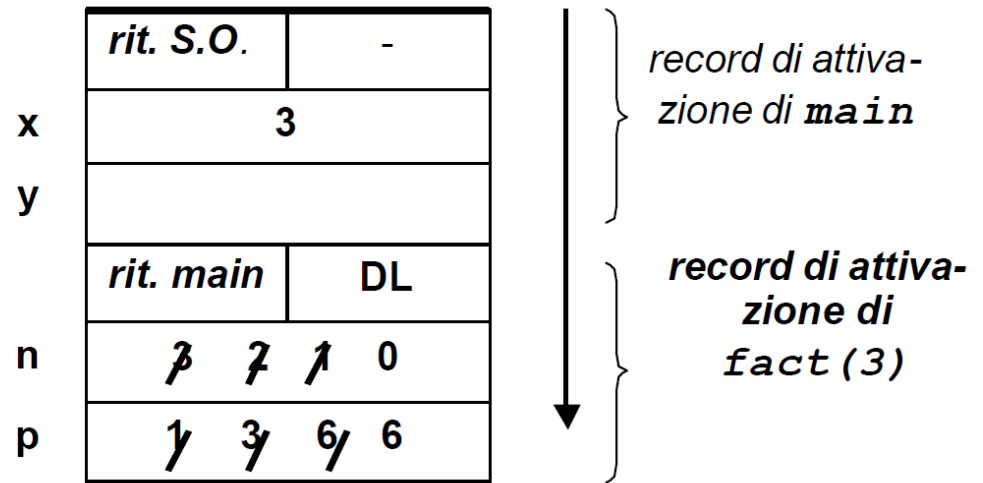
Il contenuto viene copiato dal chiamante *prima* di distruggere il record della funzione appena terminata.

In alternativa, il risultato viene restituito dalla funzione al chiamante semplicemente *lasciandolo in un registro* della CPU.

ESEMPIO 2

```
int fact(int n) {
    int p = 1;
    while (n-- > 0) p *= n;
    return p;
}

main(){
    int x = 3, y;
    y = fact(x);
}
```



La figura illustra la situazione nello stack nel momento di massima espansione (al momento del return).

Successivamente la funzione *fact* termina, il suo RdA viene deallocato e il risultato viene trasferito (tramite un registro macchina) nella cella di nome *y*.

ESEMPIO 3

Programma per l'ordinamento di variabili (SBAGLIATO):

```
main() {  
    int X = 33, Y = 5;  
    ordina(X,Y);  
}  
void ordina(int A, int B) {  
    int T;  
    if (A>B) { T=A; A=B; B=T; }  
}
```

Poiché le variabili sono passate per valore, le variazioni sono locali alla procedura e non sono visibili dal main.

	DL	RA
X	33	
Y	5	
	DL	RA
A	33	5
B	5	33
T	33	
	↓	

ESEMPIO 3

Programma per l'ordinamento di variabili (CORRETTO):

```
main() {  
    int X = 33, Y = 5;  
    ordina(&X,&Y);  
}  
void ordina(int *A, int *B) {  
    int T;  
    if (*A>*B)  
        { T=*A; *A=*B; *B=T; }  
}
```

Le variabili A e B sono passate per riferimento e contengono un link alla corrispondente variabile X o Y del chiamante cioè l'indirizzo (qui indicato con α , β) di tali variabili.

