

# Sistemi Operativi

Laurea in Ingegneria Informatica

Università Roma Tre

Docente: Romolo Marotta

## Gestione della memoria

1. Requisiti
2. Allocazione di memoria contigua
3. Paginazione
4. Segmentazione
5. Memoria Virtuale

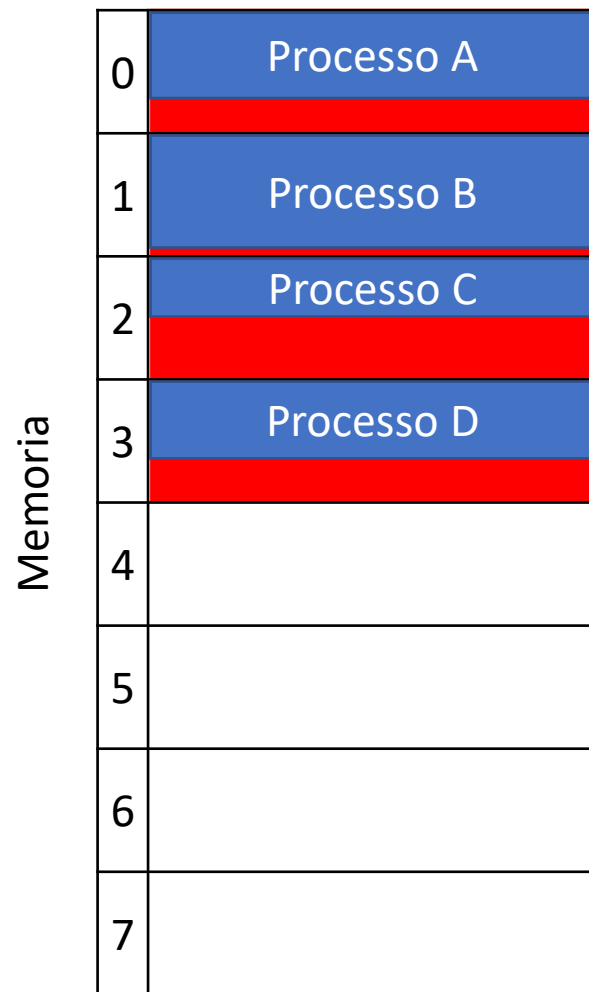
# Requisiti fondamentali

- Protezione
  - Necessaria per impedire a processi di interferire con altri processi e con il sistema operativo
- Condivisione
  - Può essere vantaggioso per ridurre la memoria richiesta e/o abilitare cooperazione/comunicazione tra processi
- Partizionamento
  - Mantenere più processi attivi in memoria al fine di massimizzare l'utilizzo delle risorse hardware

# Memory partitioning

## Partizioni fisse a taglia fissa

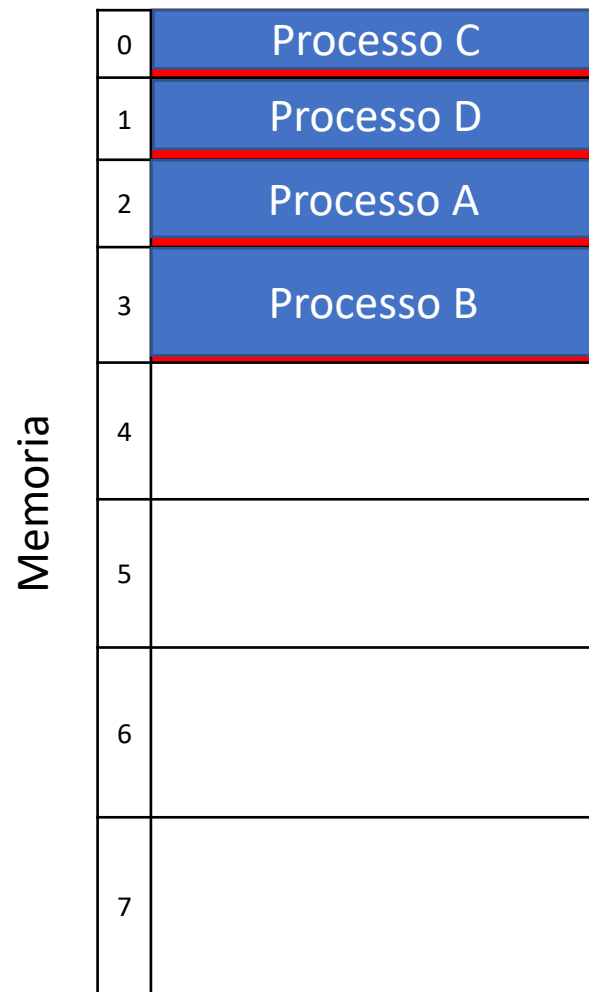
- Pros
  - Semplice da implementare e basso overhead per il SO
- Cons
  - Frammentazione interna
  - Livello di multiprogrammazione limitato dal numero di partizioni



# Memory partitioning

## Partizioni fisse a taglia variabile

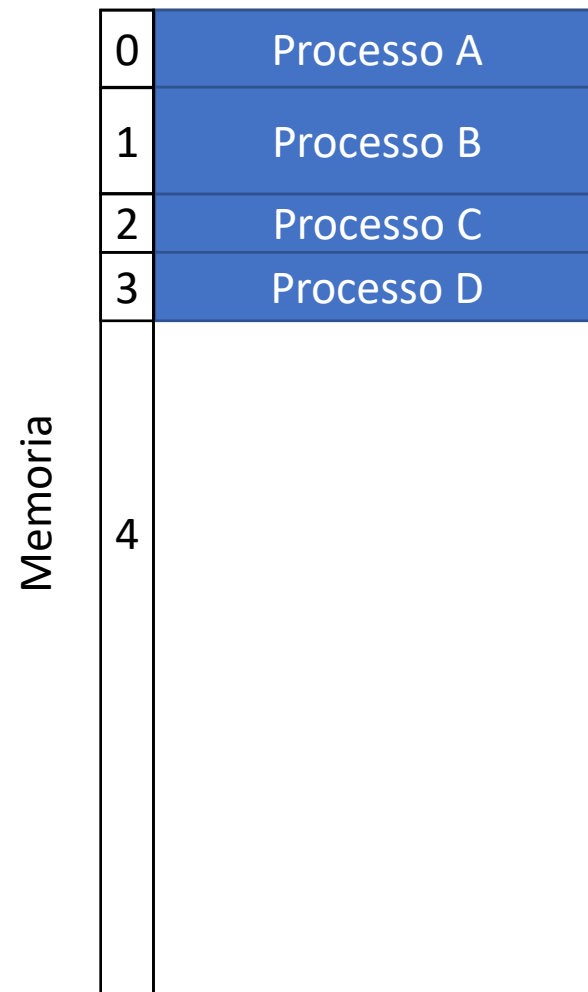
- Pros
  - Semplice da implementare e basso overhead per il SO
- Cons
  - Frammentazione interna
  - Livello di multiprogrammazione limitato dal numero di partizioni



# Memory partitioning

## Partizioni dinamiche

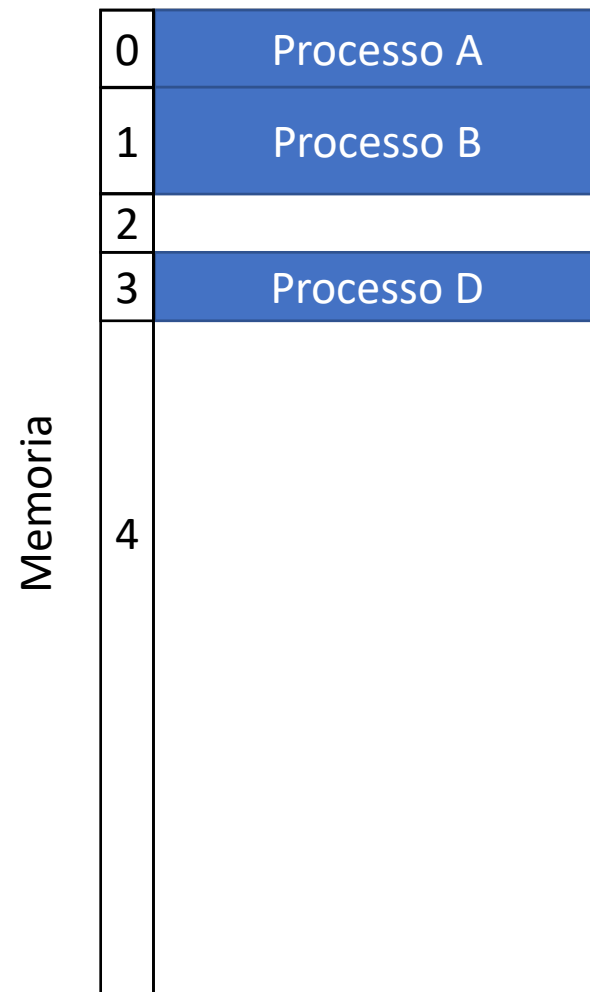
- Pros
  - Frammentazione interna ridotta o assente
- Cons
  - Frammentazione esterna
  - Schema più complesso



# Memory partitioning

## Partizioni dinamiche

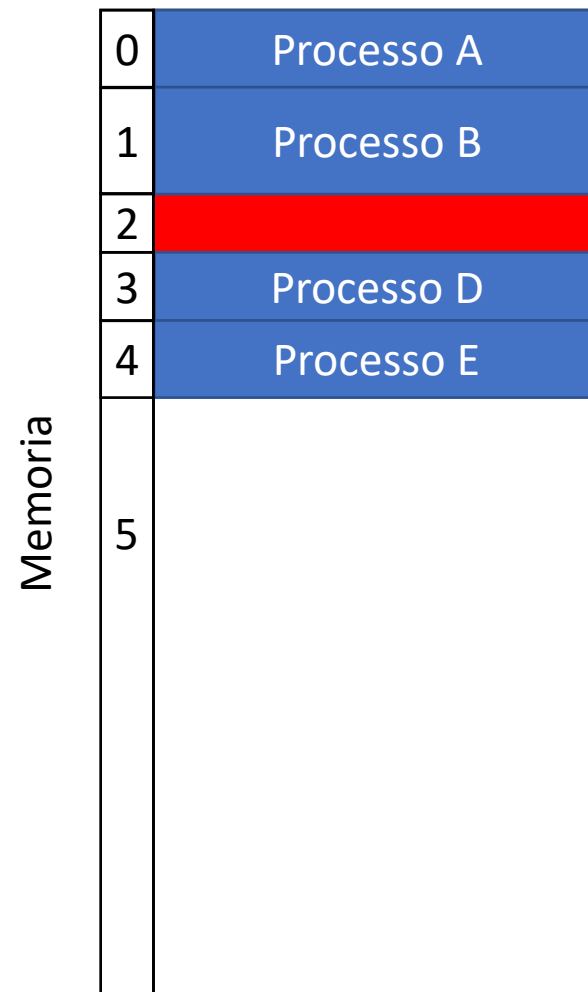
- Pros
  - Frammentazione interna ridotta o assente
- Cons
  - Frammentazione esterna
  - Schema più complesso



# Memory partitioning

## Partizioni dinamiche

- Pros
  - Frammentazione interna ridotta o assente
- Cons
  - Frammentazione esterna
  - Schema più complesso



# Memory partitioning

## Partizioni dinamiche

- Pros
  - Frammentazione interna ridotta o assente
- Cons
  - Frammentazione esterna
  - Schema più complesso
  - Algoritmi per l'allocazione:
    - Best fit

Memoria	0	Processo A
	1	
	2	Processo C
	3	Processo D
	4	Processo E
	5	Processo F
	6	Processo G
	7	Processo M
	8	Processo I
	9	Processo L
	10	



# Memory partitioning

## Partizioni dinamiche

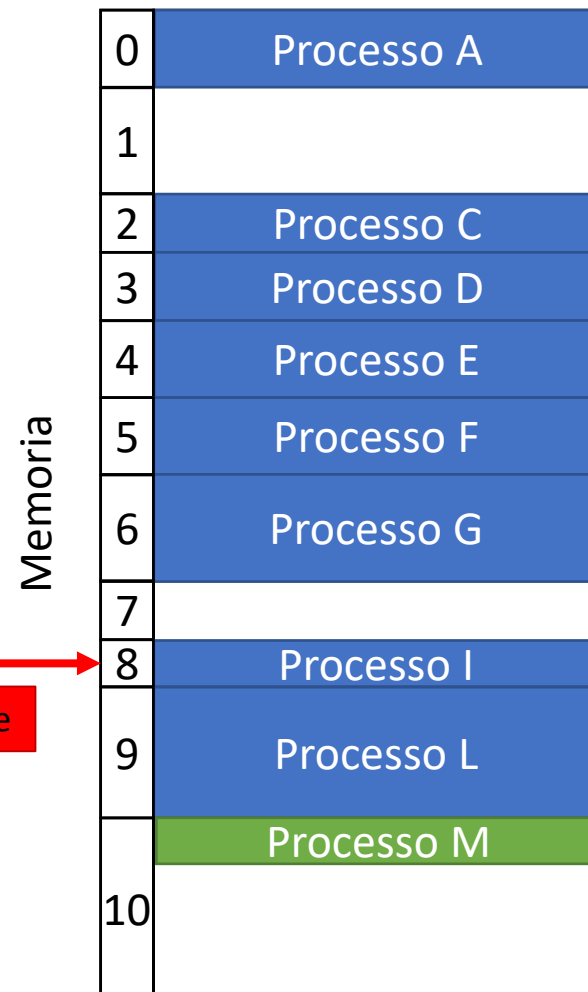
- Pros
  - Frammentazione interna ridotta o assente
- Cons
  - Frammentazione esterna
  - Schema più complesso
  - Algoritmi per l'allocazione:
    - Best fit
    - First fit

Memoria	0	Processo A
	1	Processo M
	2	Processo C
	3	Processo D
	4	Processo E
	5	Processo F
	6	Processo G
	7	
	8	Processo I
	9	Processo L
	10	

# Memory partitioning

## Partizioni dinamiche

- Pros
  - Frammentazione interna ridotta o assente
- Cons
  - Frammentazione esterna
  - Schema più complesso
  - Algoritmi per l'allocazione:
    - Best fit
    - First fit
    - Next fit



# Memory partitioning

## Partizioni dinamiche

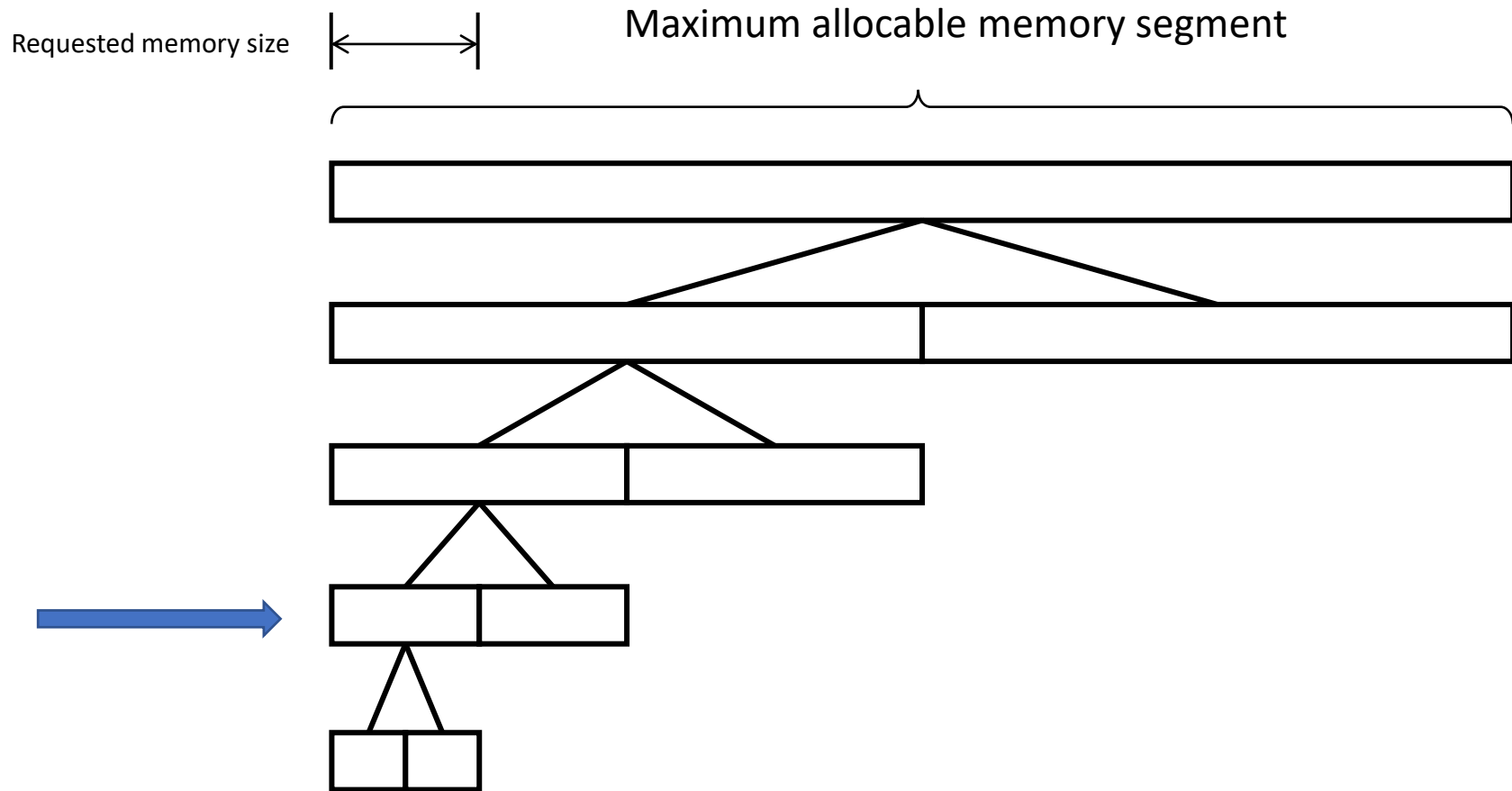
- Pros
  - Frammentazione interna ridotta o assente
- Cons
  - Frammentazione esterna
  - Schema più complesso
  - Algoritmi per l'allocazione:
    - Best fit
    - First fit
    - Next fit
  - Deframmentazione periodica

Memoria	0	Processo A
	1	Processo C
	2	Processo D
	3	Processo E
	4	Processo F
	5	Processo G
	6	Processo I
	7	Processo L
	8	

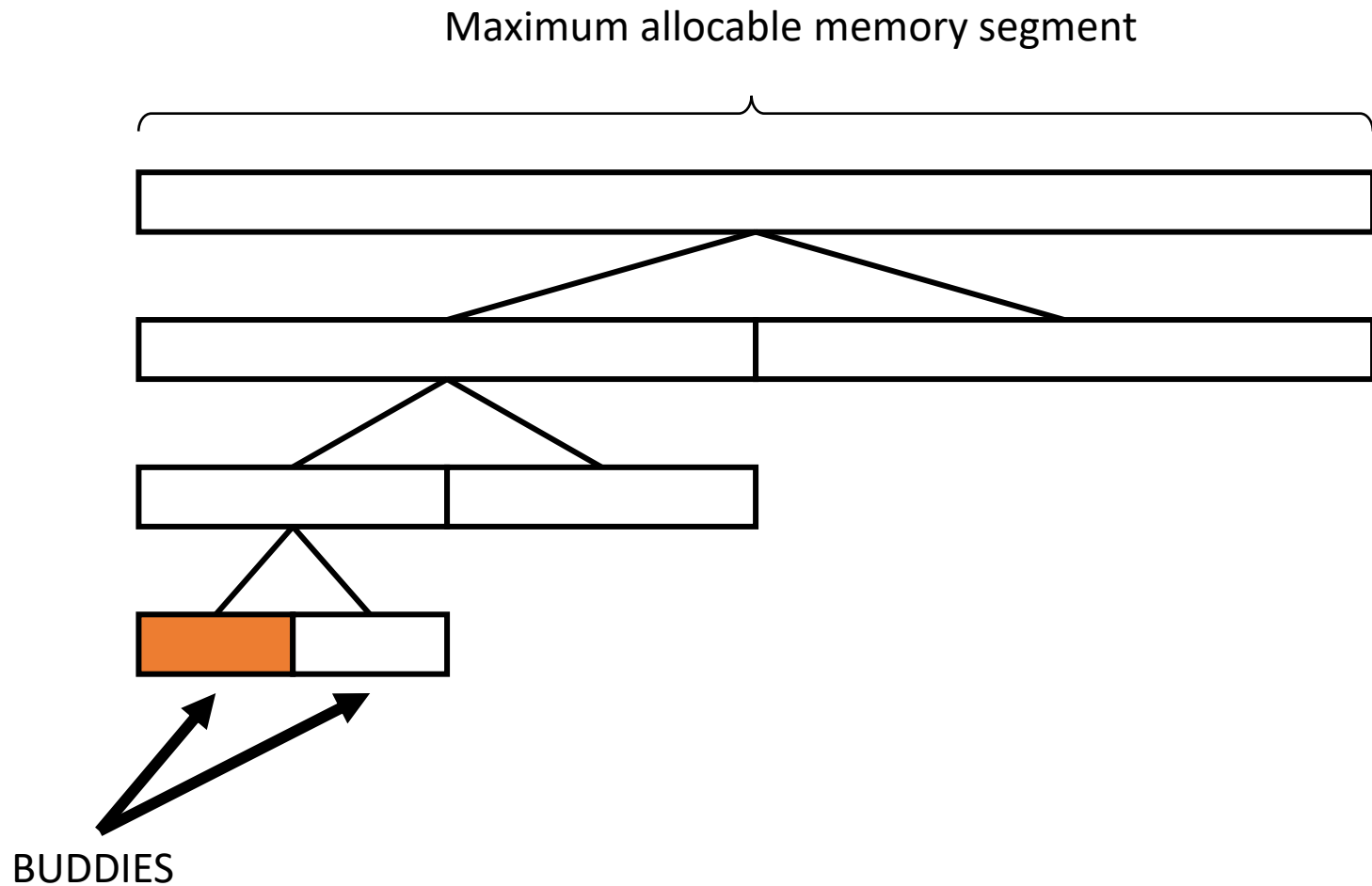
# Memory partitioning

- Partizioni fisse e dinamica hanno limitazioni comuni:
  - Frammentazione interna
  - Frammentazione esterna e gestione complessa
- Buddy system
  - Compromesso tra frammentazione interna e gestione
  - Taglia minima fissata a  $L = 2^L$
  - Taglia massima fissata a  $R = 2^U$
  - Una partizione di taglia pari a  $K$  occupa uno slot di dimensione  $L^{i+1}$  tale che  $L^i < K \leq L^{i+1}$

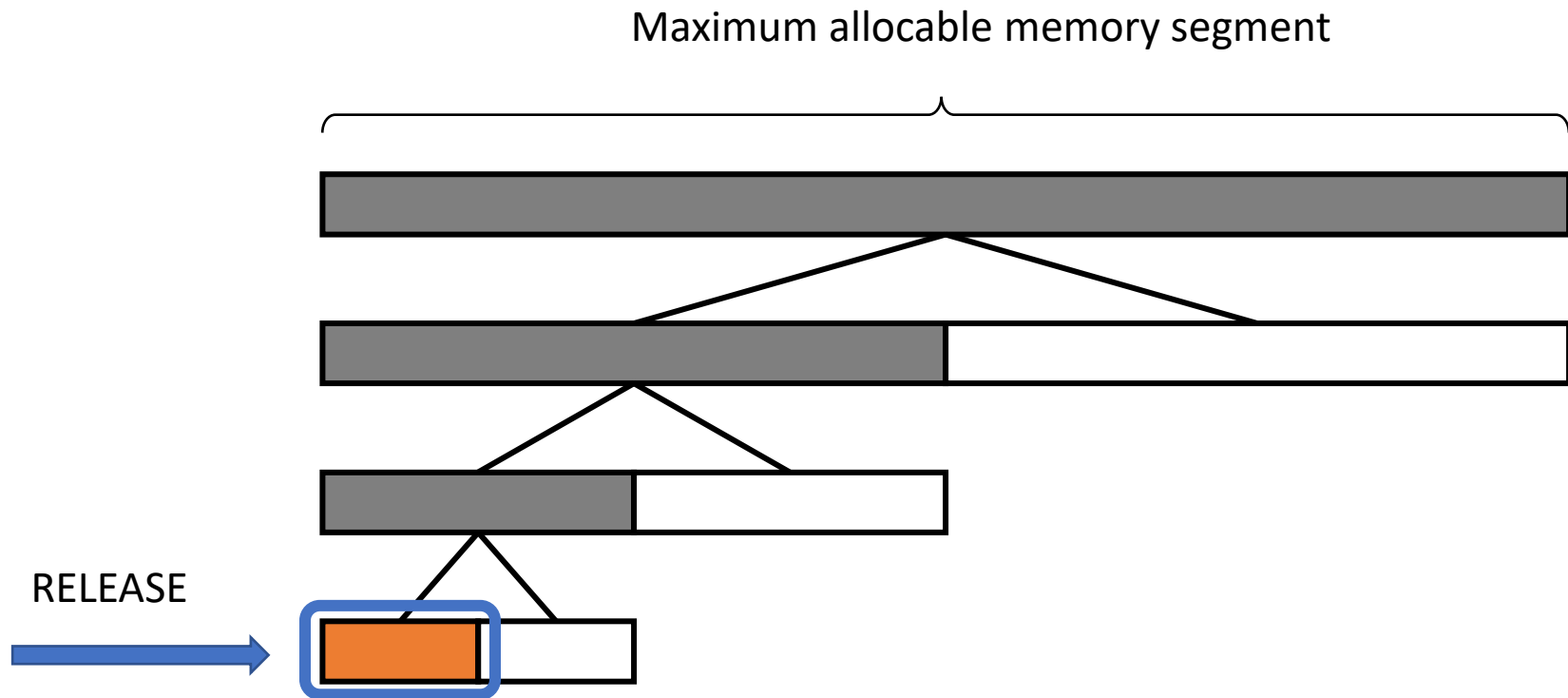
# Buddy system



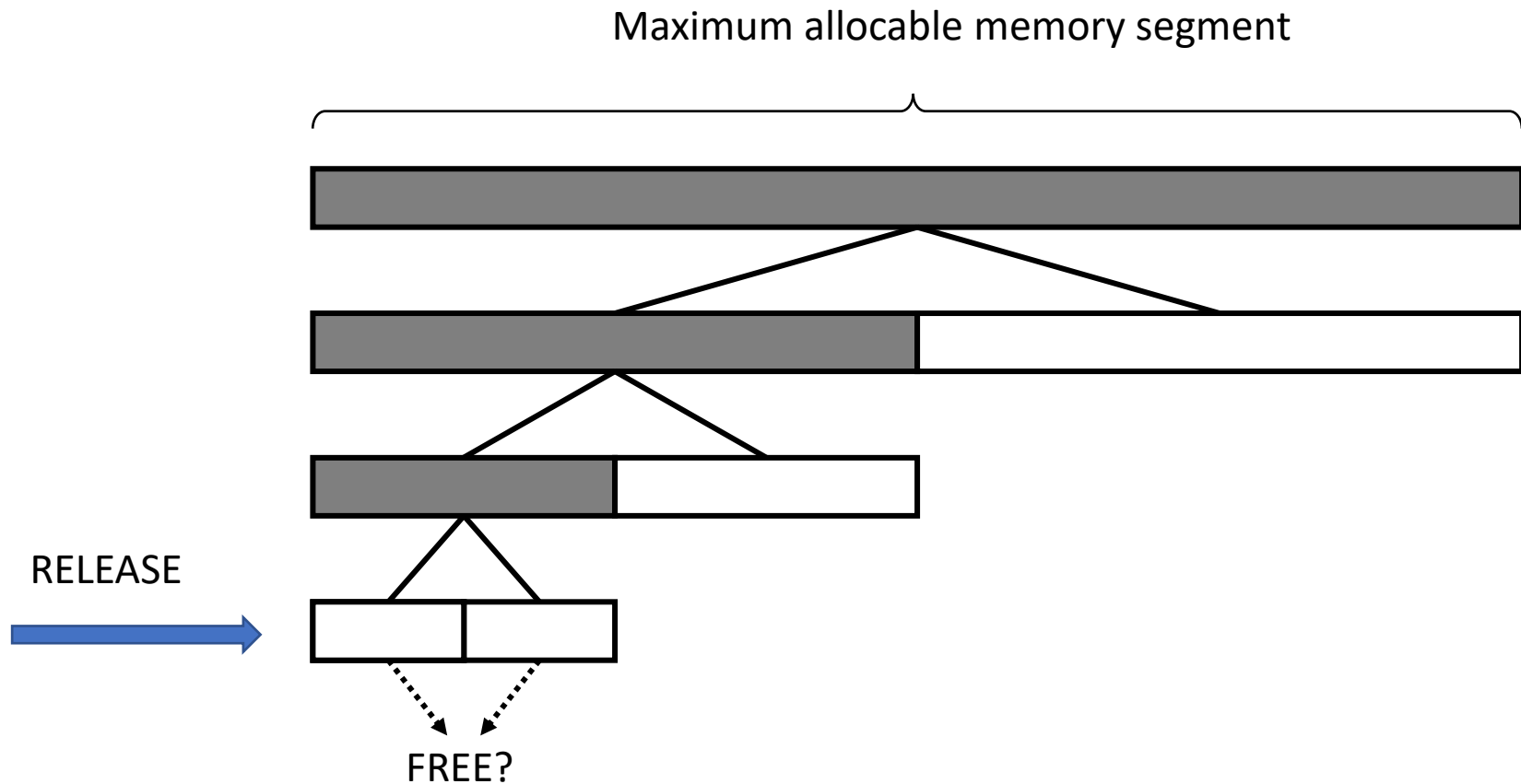
# Buddy system



# Buddy system



# Buddy system



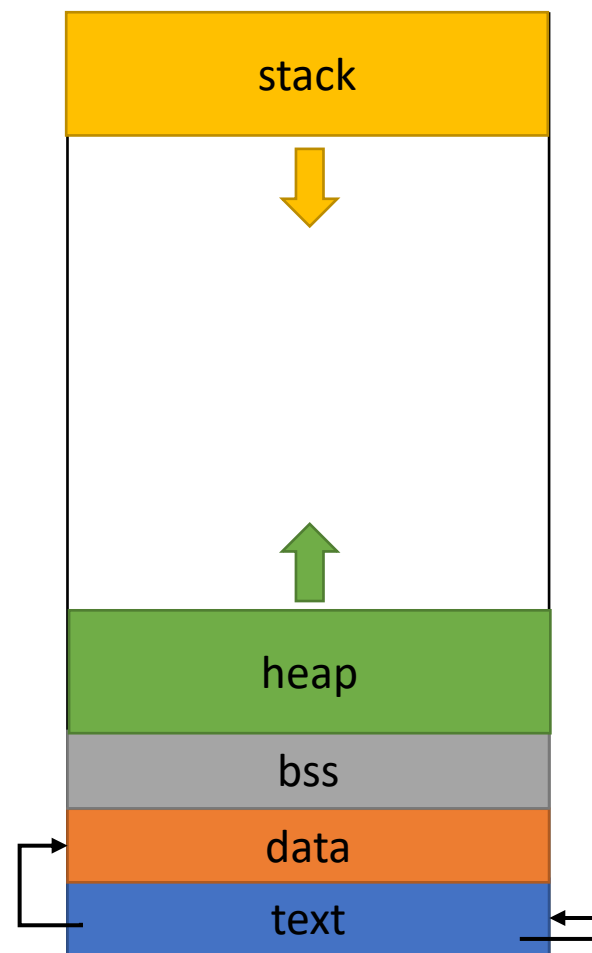


# Memory partitioning

- Partizioni fisse e dinamiche hanno limitazioni comuni:
  - Frammentazione interna
  - Frammentazione esterna e gestione complessa
- Buddy system
  - Compromesso tra frammentazione interna e gestione
- Assegnazione delle partizioni a processi
  - Statica: una volta assegnata una partizione ad un processo, l'associazione non viene riconsiderata
  - Dinamica: l'assegnazione delle partizioni ai relativi processi viene rivalutata ad ogni swap in

# Binding di indirizzi

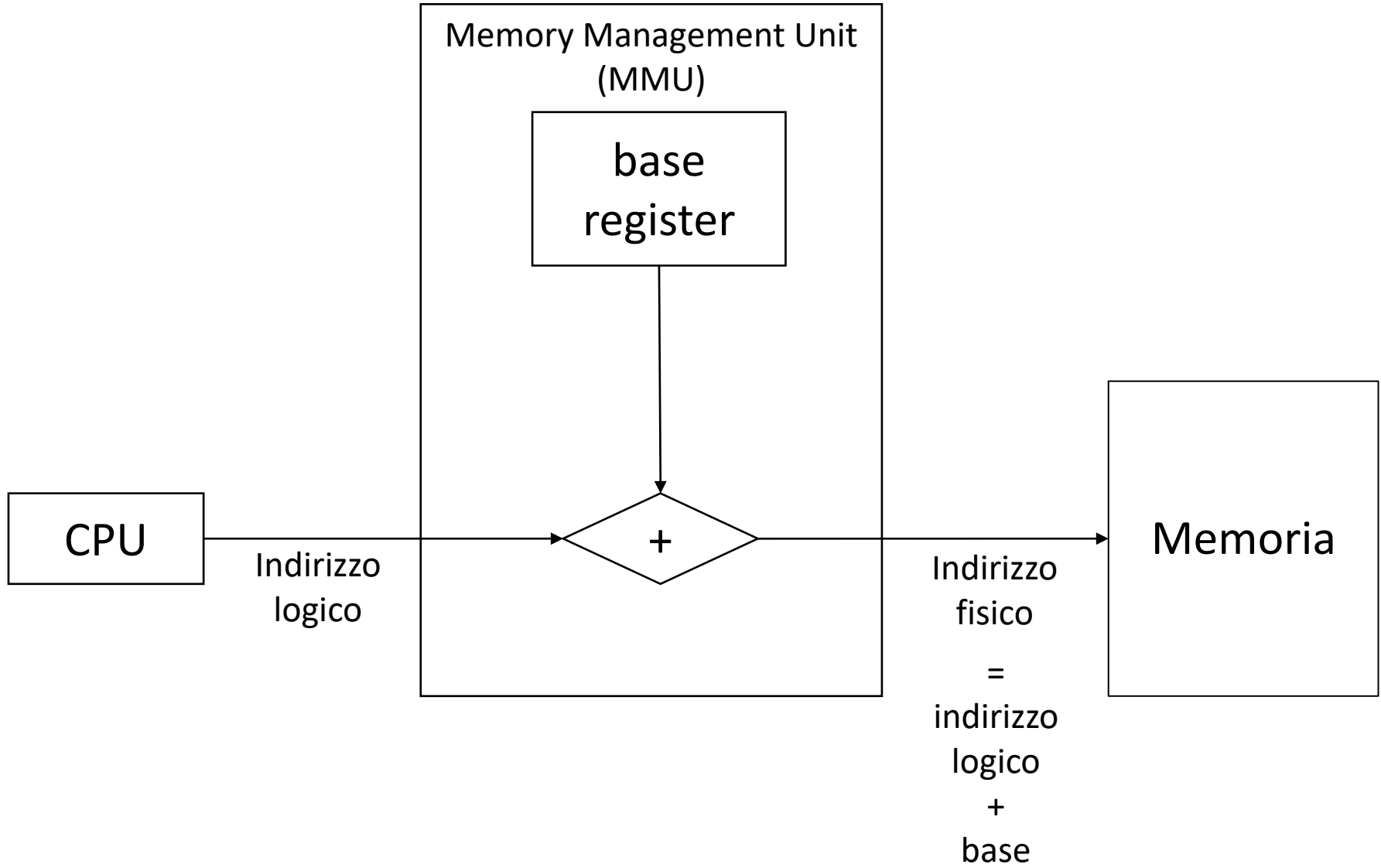
- L'operazione di mappare indirizzi da uno spazio A ad uno spazio B è denominata binding
- L'immagine di programma contiene riferimenti all'interno dell'immagine stessa (tipicamente tramite indirizzi simbolici)
- Indirizzi delle celle di memoria identificati
  - a tempo di compilazione
    - compatibile solo con approcci di (pre)assegnazione statica delle partizioni
  - a tempo di caricamento
    - generazione di codice rilocabile, ogni indirizzo è risolto tramite spiazzamento dalla base
  - a tempo di esecuzione
    - gli effettivi indirizzi vengono individuati ad ogni accesso



# Requisiti fondamentali

- Protezione
  - Necessaria per impedire a processi di interferire con altri processi e con il sistema operativo
- Condivisione
  - Può essere vantaggioso per ridurre la memoria richiesta e/o abilitare cooperazione/comunicazione tra processi
- Partizionamento
  - Mantenere più processi attivi in memoria al fine di massimizzare l'utilizzo delle risorse hardware
- Rilocazione
  - Supporto ad immagini rilocabili

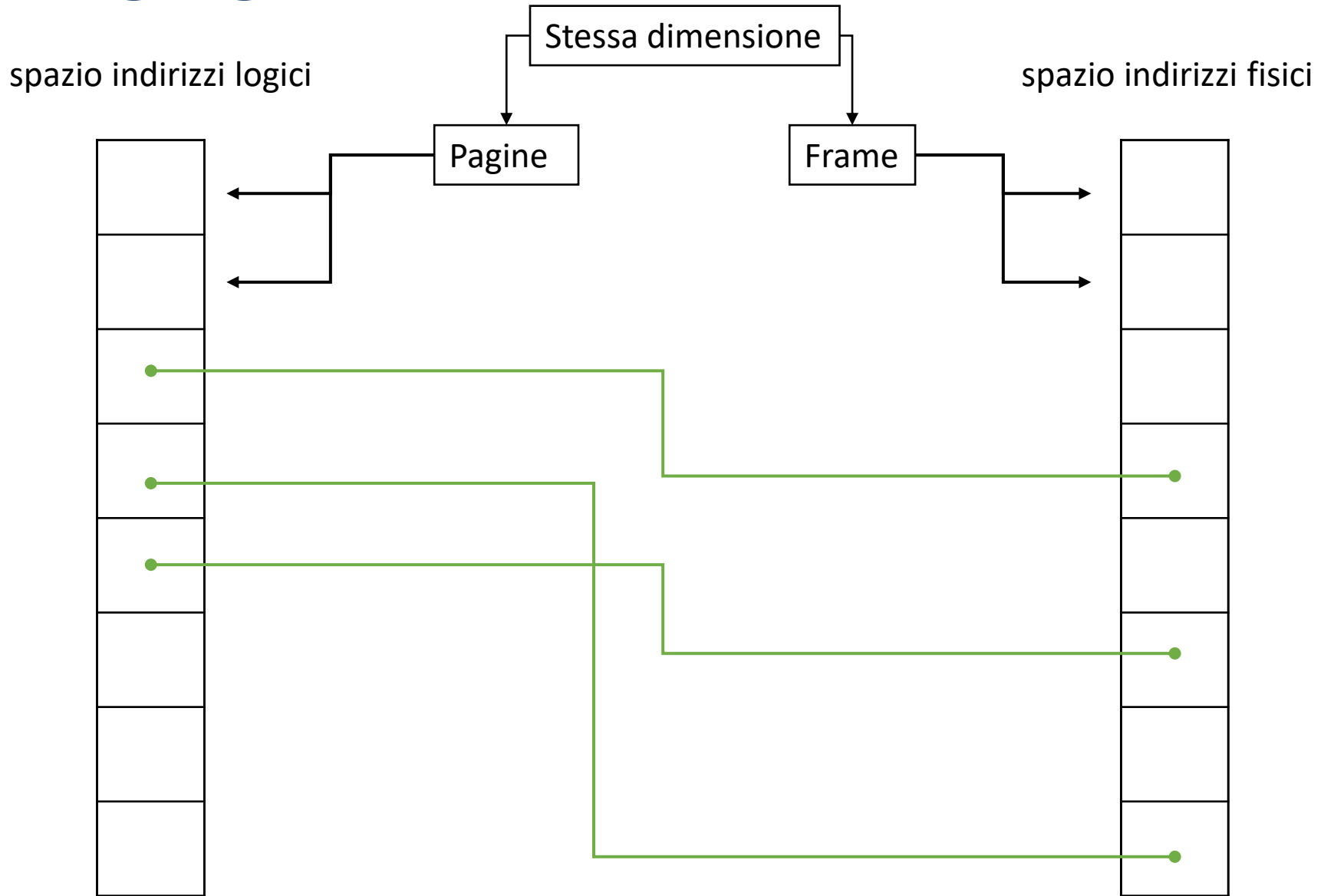
# Supporti alla rilocazione



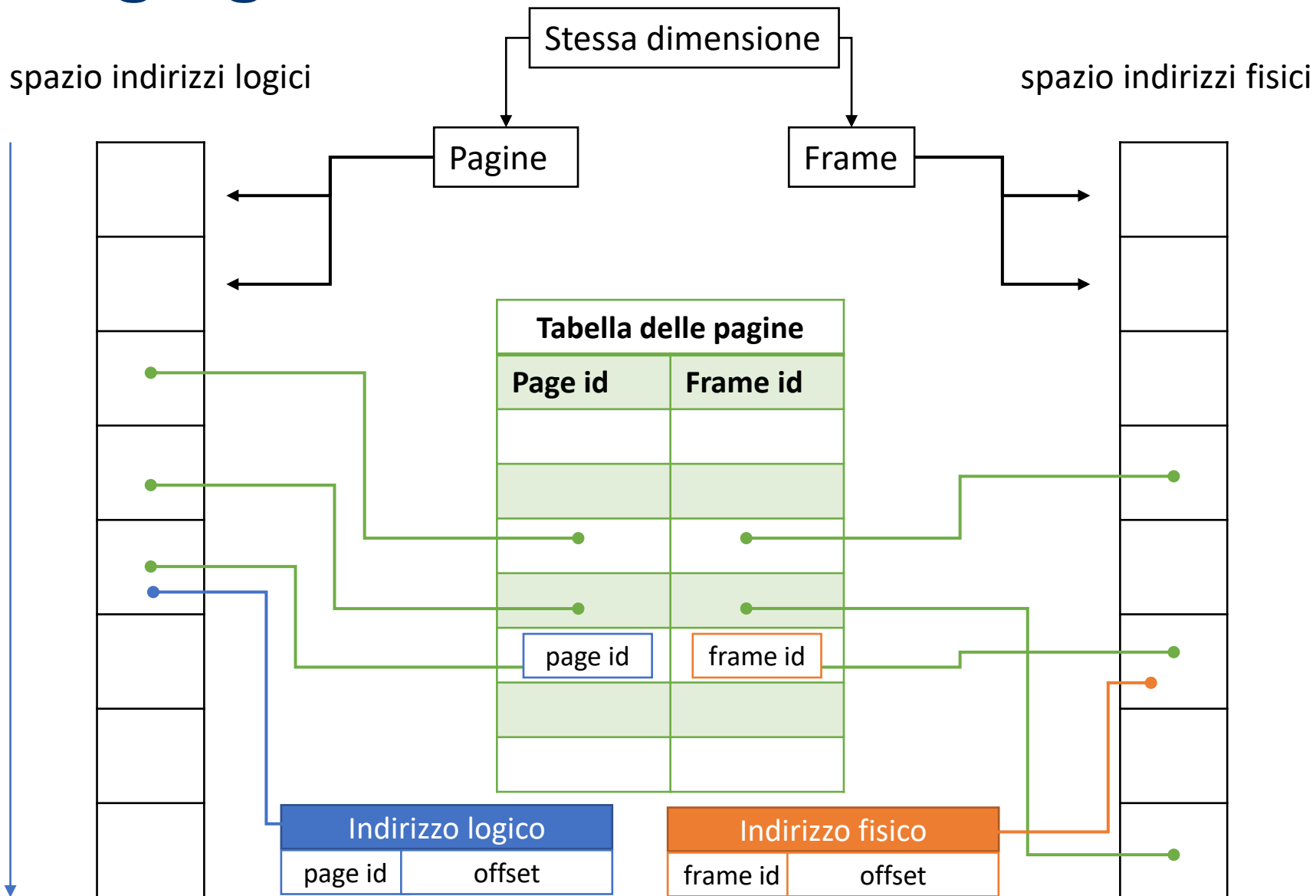
# Ancora sul memory partitioning

- Partizioni fisse e dinamiche hanno limitazioni comuni:
  - Frammentazione interna (partizioni fisse)
  - Frammentazione esterna (partizioni dinamiche)
- La criticità è strettamente legata alla necessità di mantenere lo spazio degli indirizzi fisici contiguo in memoria
- Ammettendo un spazio di indirizzi fisici non contigui è possibile:
  - Eliminare frammentazione esterna
  - Ridurre frammentazione interna

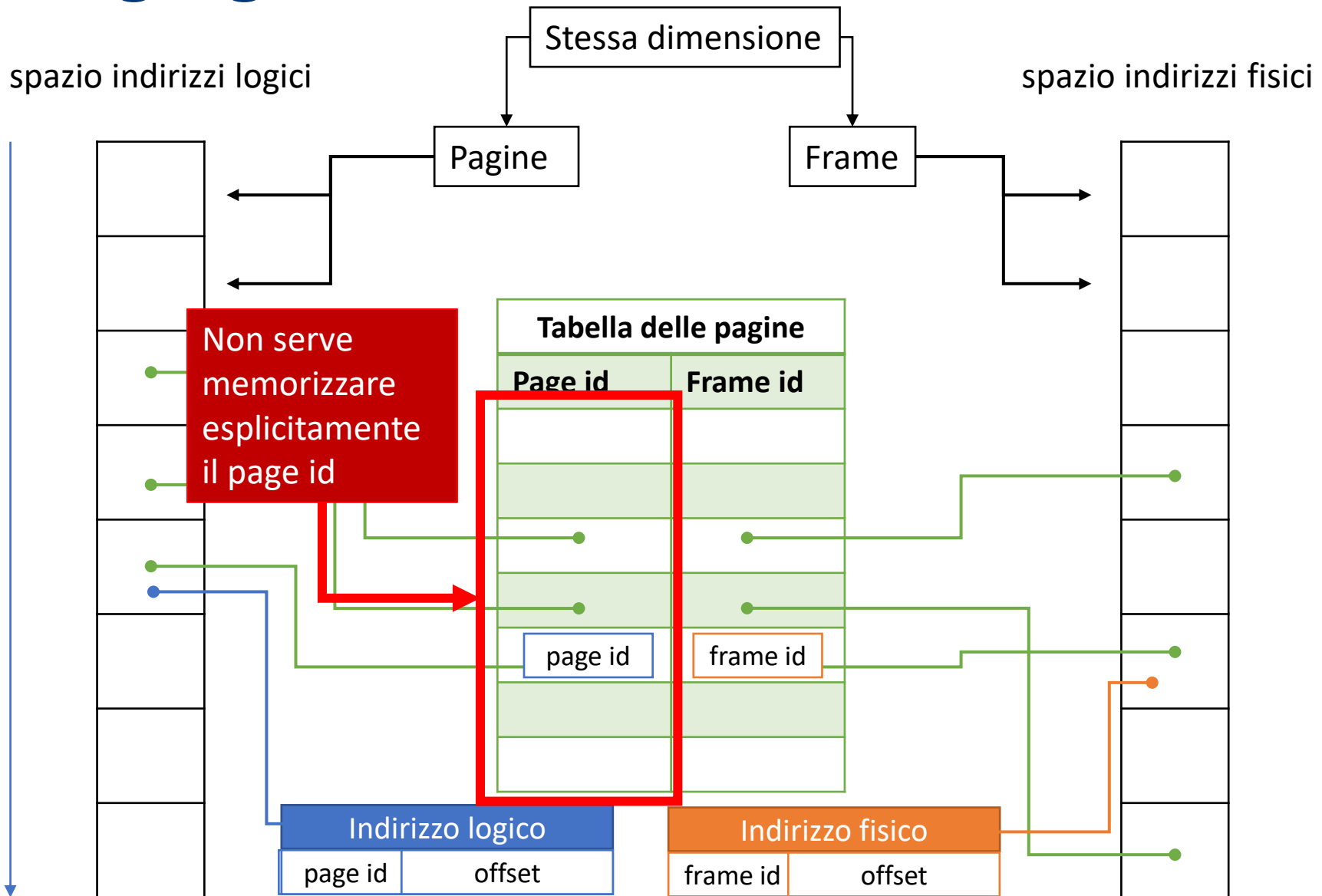
# Paging



# Paging

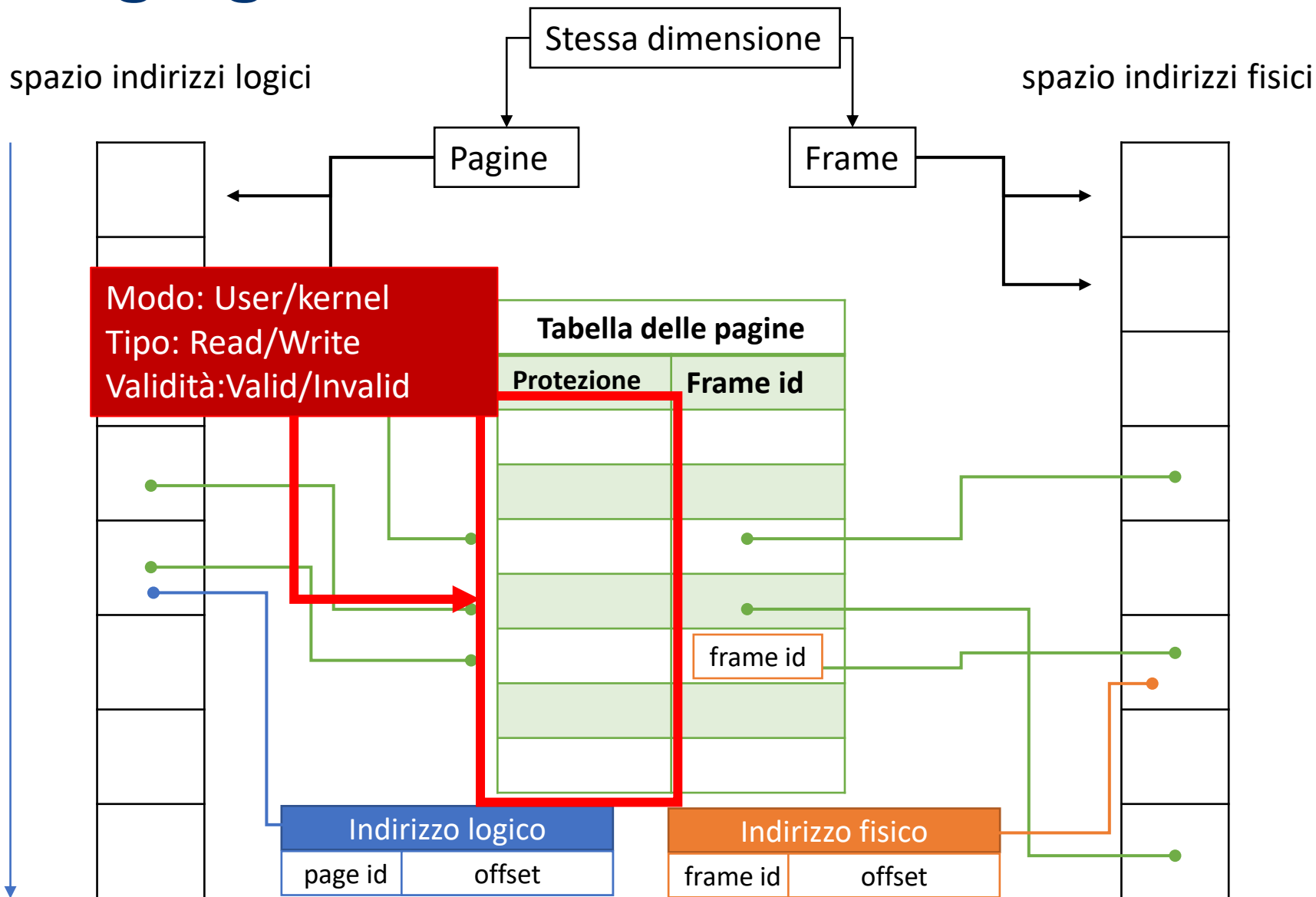


# Paging

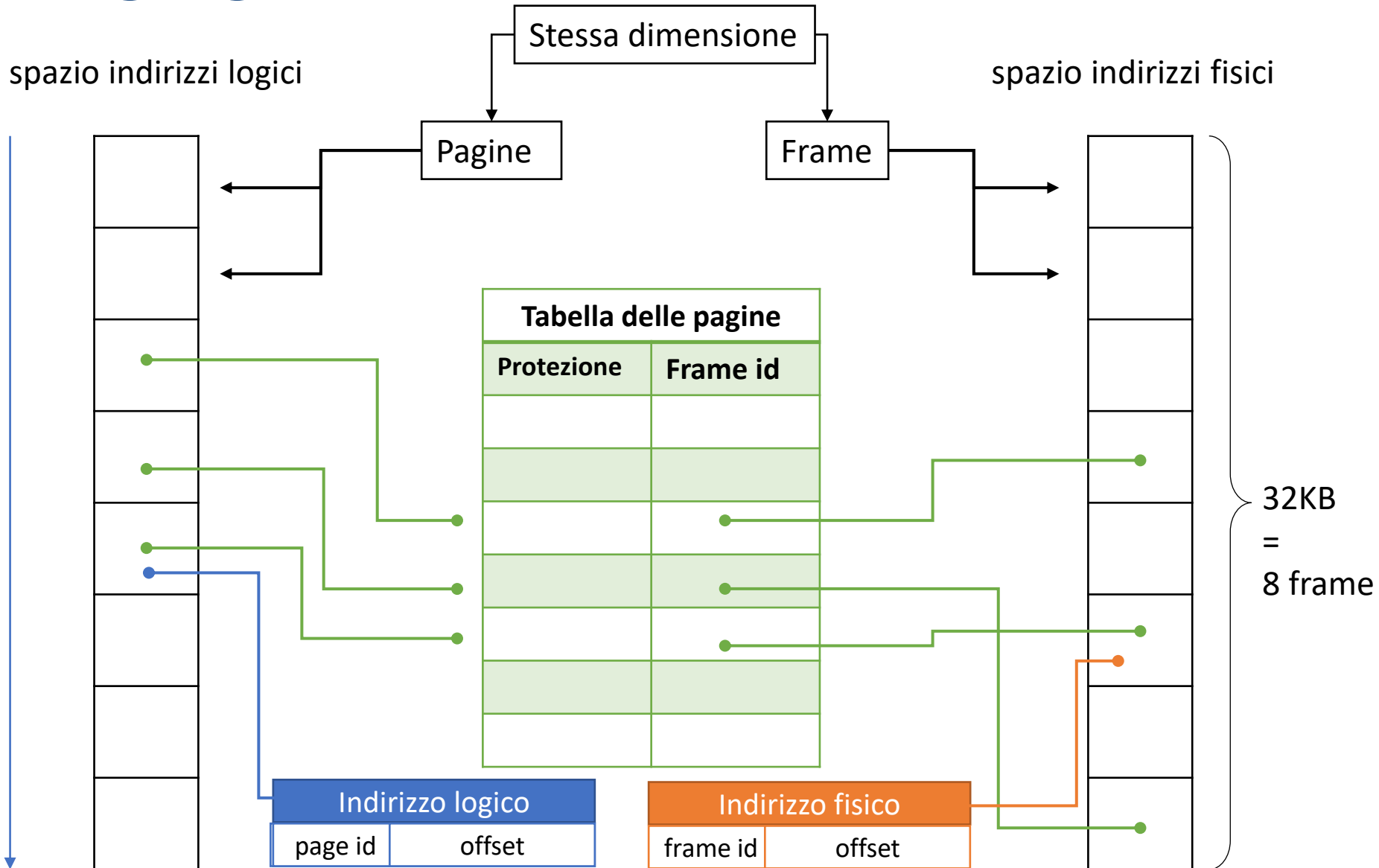





# Paging



# Paging - esempio <sup>4KB</sup>

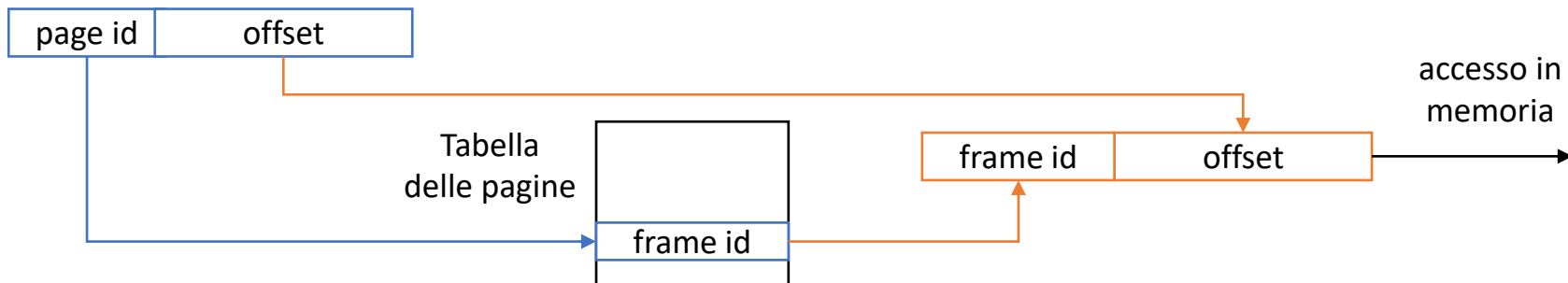


# Dimensioni della tabella delle pagine

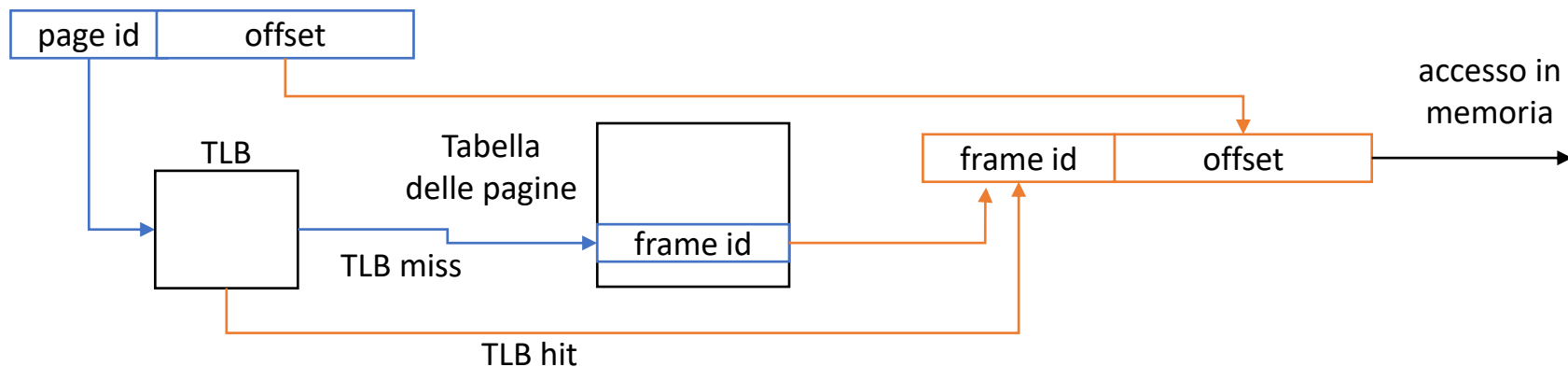
- Per ciascun processo è necessaria una tabella
- La tabella deve essere memorizzata in una porzione contigua di memoria
- Esempio 1:
  - Indirizzi logici a 32 bit
  - Pagine da 4KB
  - $\#Entry = \frac{2^{32}}{2^{12}} = 2^{20}$
  - Dimensione di una entry = 20 bit + #flags = 32bit = 4B
  - Dimensione della tabella =  $2^{20} \cdot 4B = 4MB$
- Registri non sono sufficienti  Memoria principale

# Paginazione e performance

- Performance ridotte
  - Ad ogni accesso in memoria è richiesto un ulteriore accesso



- Introduzione di una cache per la tabella delle pagine
  - Translation Lookaside Buffer (TLB)



# Paginazione e hardware

La struttura della tabella dipende dalla specifica architettura hardware

- Se la tabella delle pagine è memorizzata in RAM, esiste un registro per indicare dove è posizionata (e.g. CR3 in x86)
- Alcuni controlli di protezione possono dipendere dallo stato di processore (e.g. current privilege level e pagina user/supervisor)

La protezione della memoria è garantita

- L'offset permette di accedere solo all'interno di una pagina
- L'accesso ad una singola pagina è protetto durante la fase di risoluzione degli indirizzi logici interpretando alcuni bit contenuti nella rispettiva entry

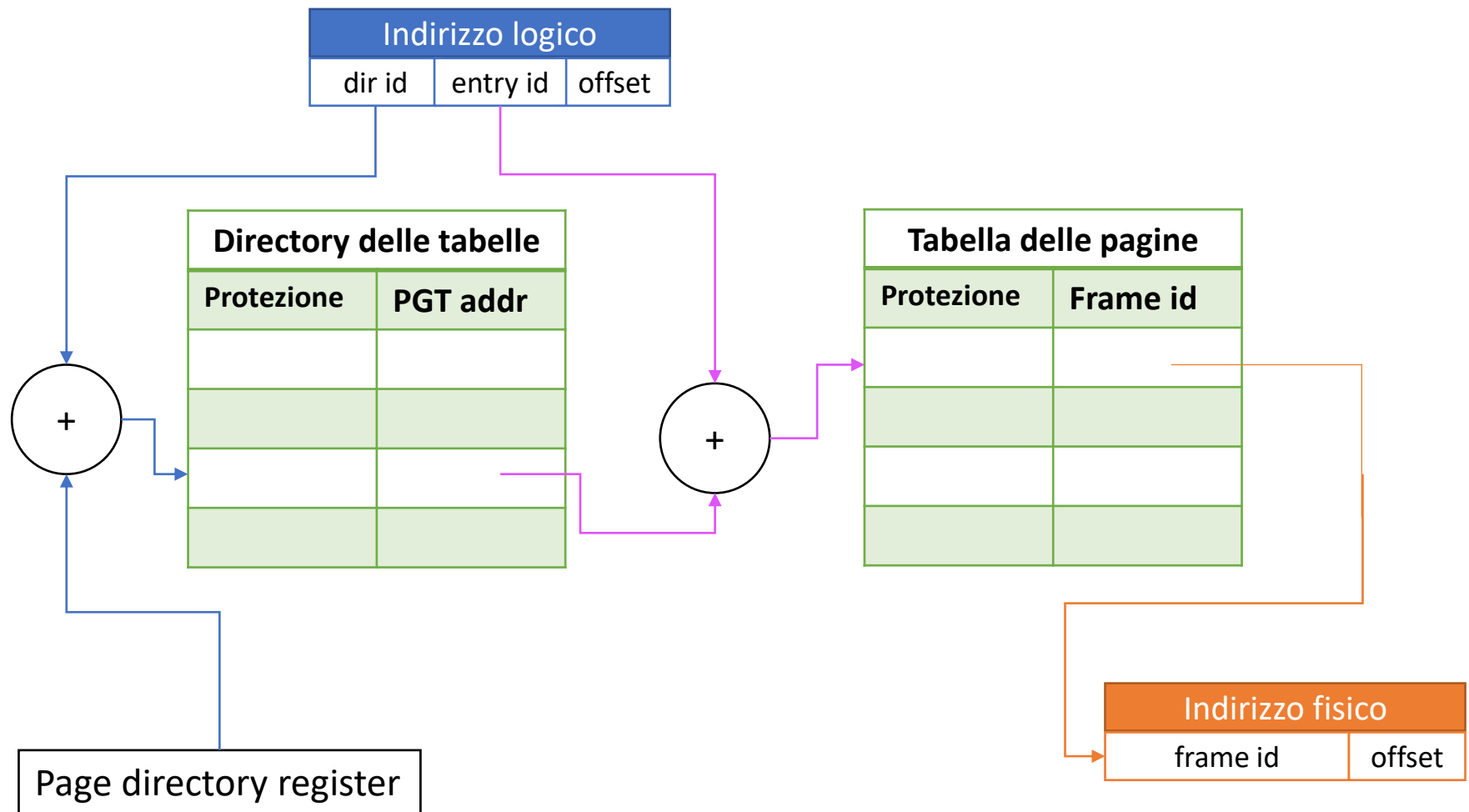
# Ancora sulla paginazione

- Permette di frazionare l'immagine di un processo (in pagine) e mantenerla in frammenti (frame) di memoria non contigui
- La tabella delle pagine mantiene la corrispondenza tra pagine e frame
- Il sistema operativo tiene traccia dei frame liberi
- Come condividere la memoria tra processi?
- Ad una pagina è necessariamente associato un frame?
  - Esempio 2
    - Indirizzi logici a 46 bit
    - Dimensione della tabella =  $2^{46-12+2}B = 2^{36}B = 64GB$

## Indirizzi lineari



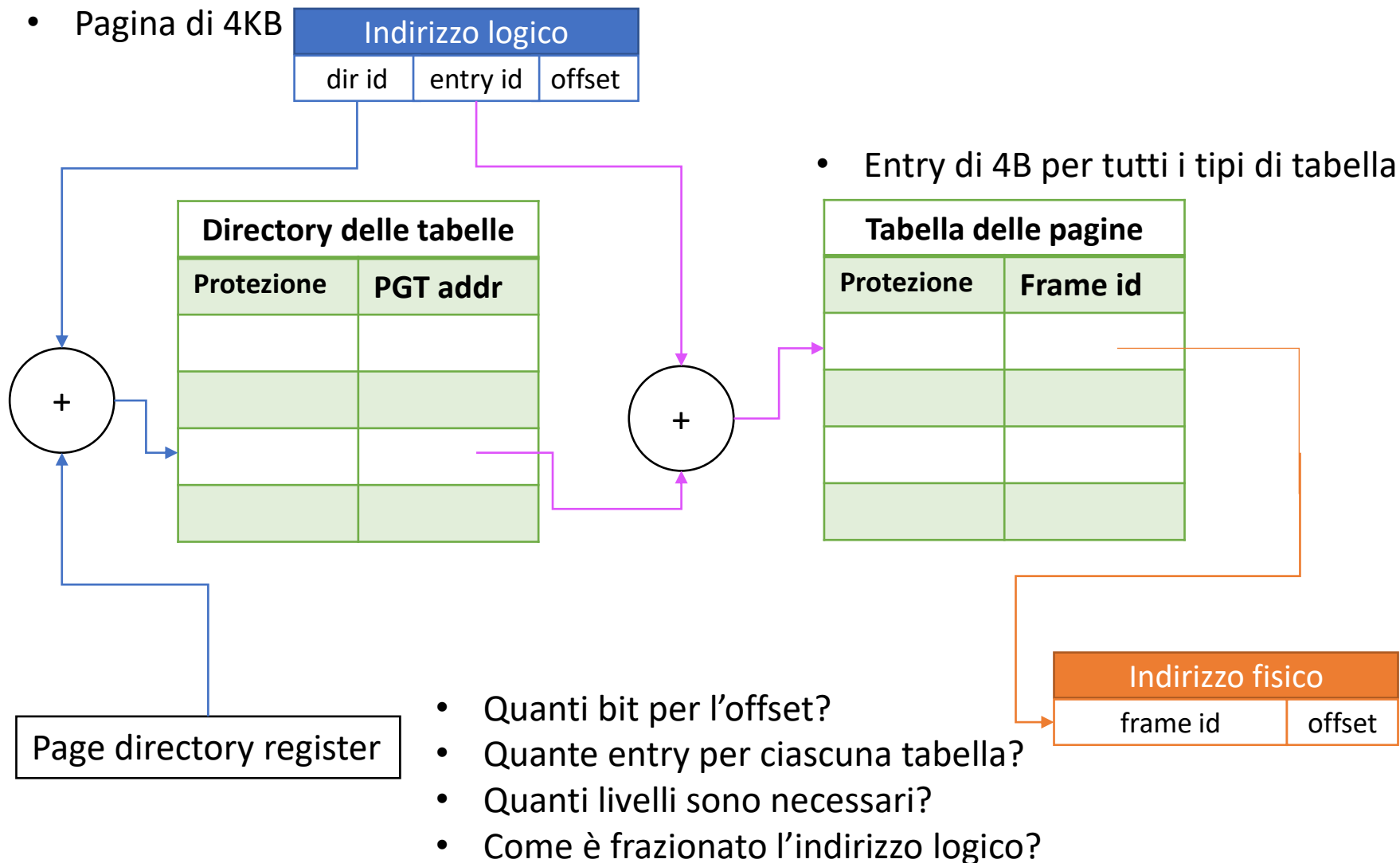
# Paginazione gerarchica





# Paginazione gerarchica - esempio

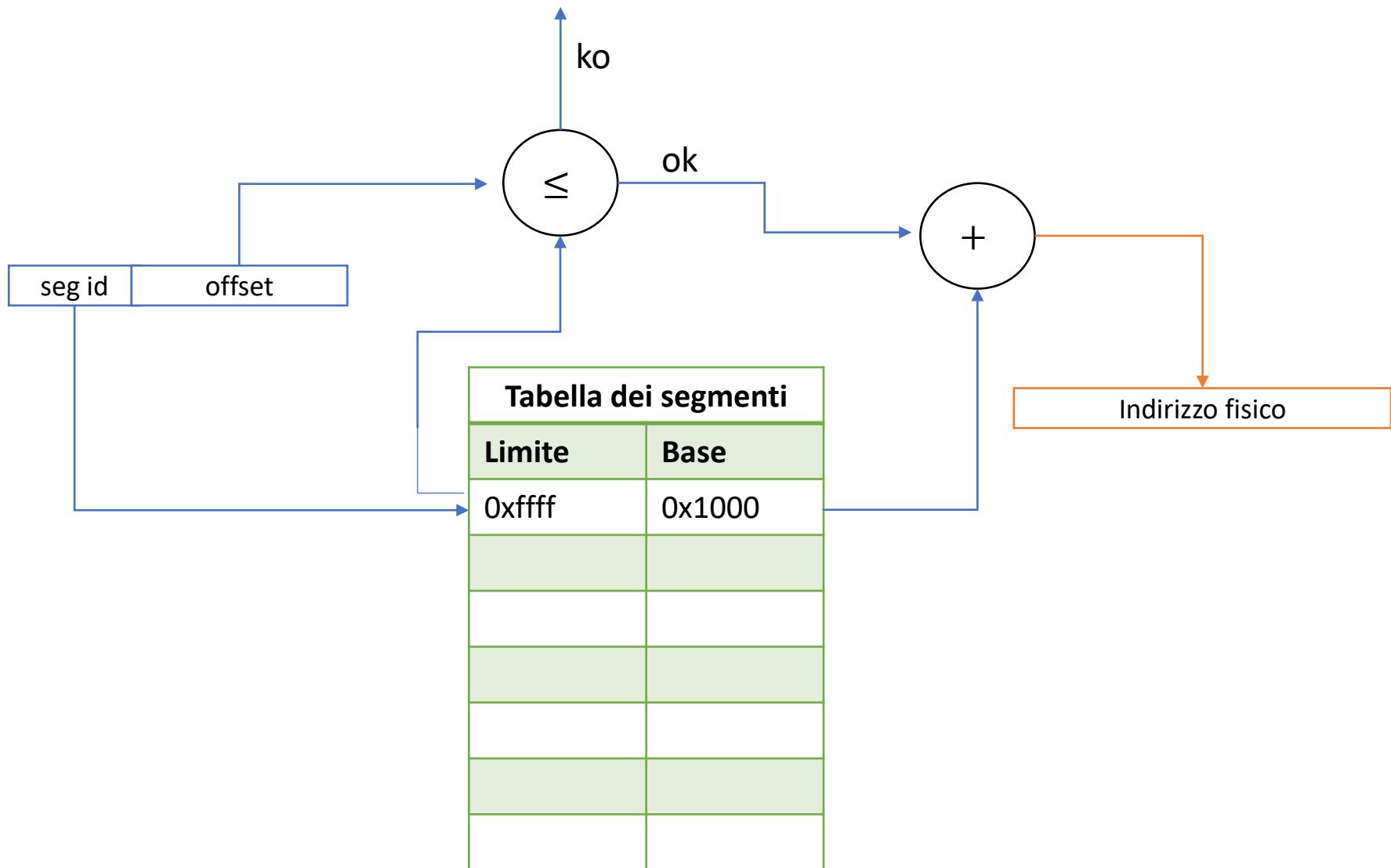
- Indirizzi lineari da 32 bit
- Pagina di 4KB



# Segmentazione

- Spazio di indirizzamento viene visto come un insieme di segmenti distinti
- La segmentazione è visibile al programmatore
- Diversi segmenti possono essere caricati in partizioni di memoria fisica non contigue
- Ogni segmento può avere una taglia differente
- Indirizzi logici sono formati da:
  - Numero di segmento
  - Spiazzamento all'interno del segmento (offset)
- Esiste una tabella che mantiene la corrispondenza tra:
  - Id del segmento
  - Dimensione del segmento
  - Posizione del segmento nello spazio di indirizzamento

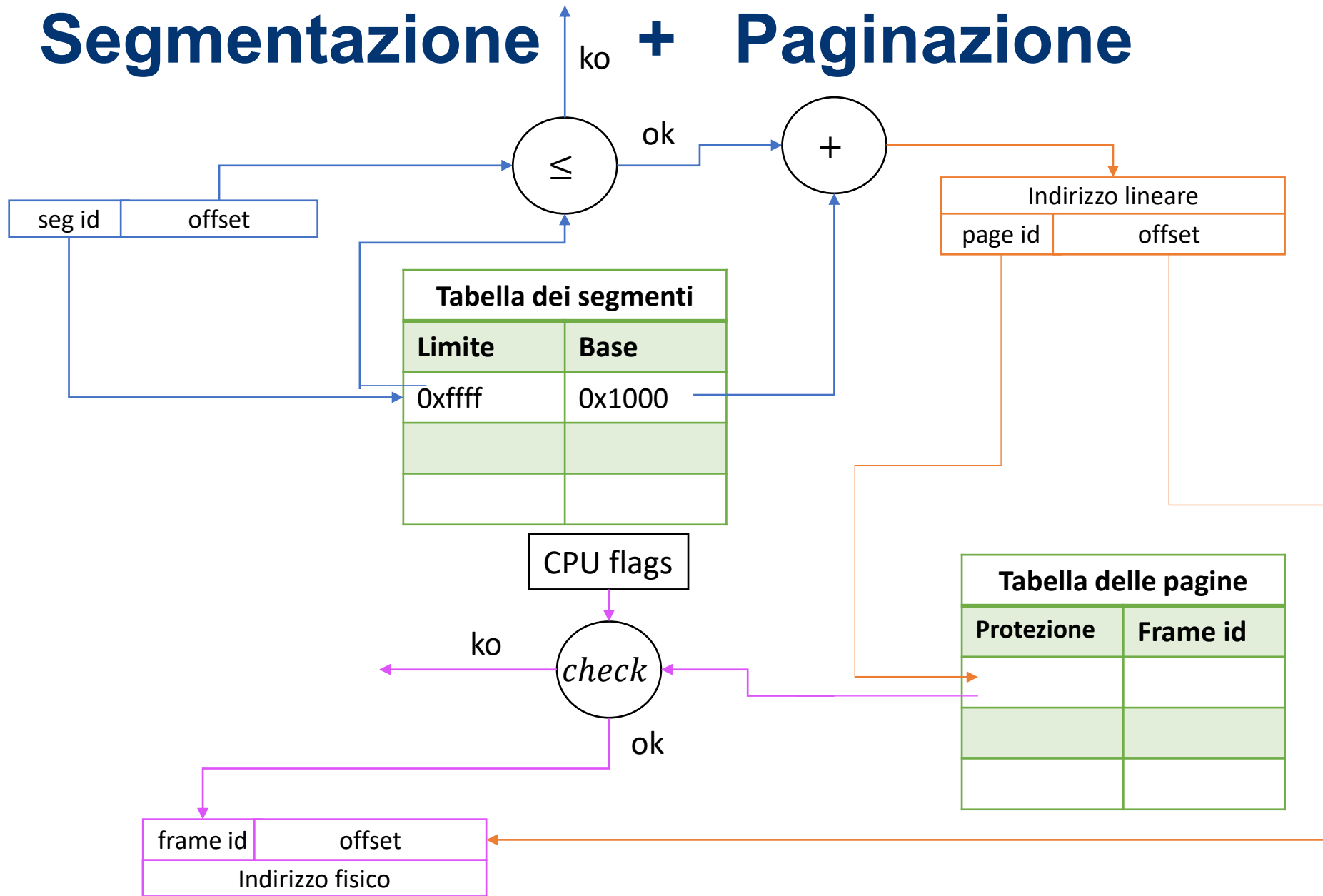
# Segmentazione



# Segmentazione

- Induce a frammentazione esterna
- Segmentazione paginata
  - L'indirizzo risultante dalla risoluzione della segmentazione è un **indirizzo lineare**
  - L'indirizzo lineare è utilizzato per accedere alla tabella delle pagine ed ottenere l'indirizzo fisico

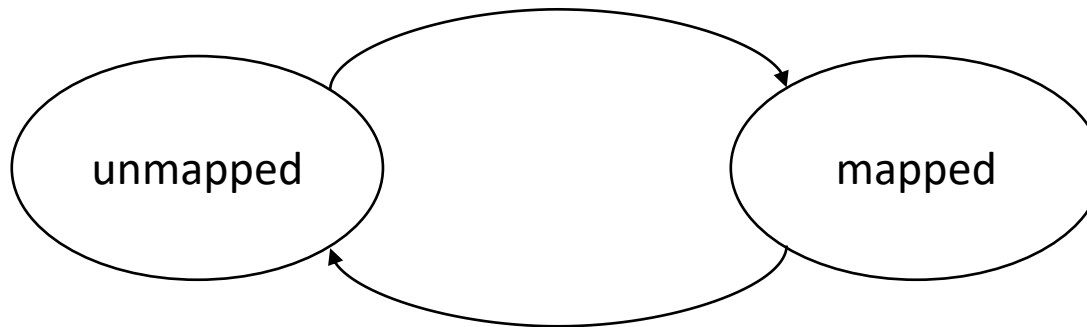
# Segmentazione + Paginazione



# Stato di una porzione di address space

Una porzione di address space logico (pagina/segmento) è:

- Mappata
  - i relativi indirizzi lineari hanno una traduzione nello spazio di indirizzi fisici
- Non mappata
  - i relativi indirizzi lineari non hanno una traduzione nello spazio di indirizzi fisici



# Servizi di sistema per la mappatura

Nei sistemi operativi moderni il programma può richiedere la mappatura a tempo di esecuzione

- Esistono servizi di sistemi per la mappatura/demappatura di porzioni dello spazio di indirizzamento logico

Sistemi POSIX:

```
#include <sys/mman.h>
```

```
void *mmap(void *addr, size_t len,  
           int prot, int flags,  
           int fildes, off_t off);
```

- prot: PROT\_EXEC, PROT\_READ, PROT\_WRITE, PROT\_NONE
- flags: MAP\_PRIVATE, MAP\_SHARED, MAP\_FIXED
- fildes: descrittore ad un oggetto di memoria
- off: spiazzamento all'interno dell'oggetto

Alcuni sistemi operativi (e.g., Linux) supportano mapping anonimi (senza oggetto di memoria)

- flags: MAP\_x | MAP\_ANONYMOUS
- fildes= -1, offset = 0

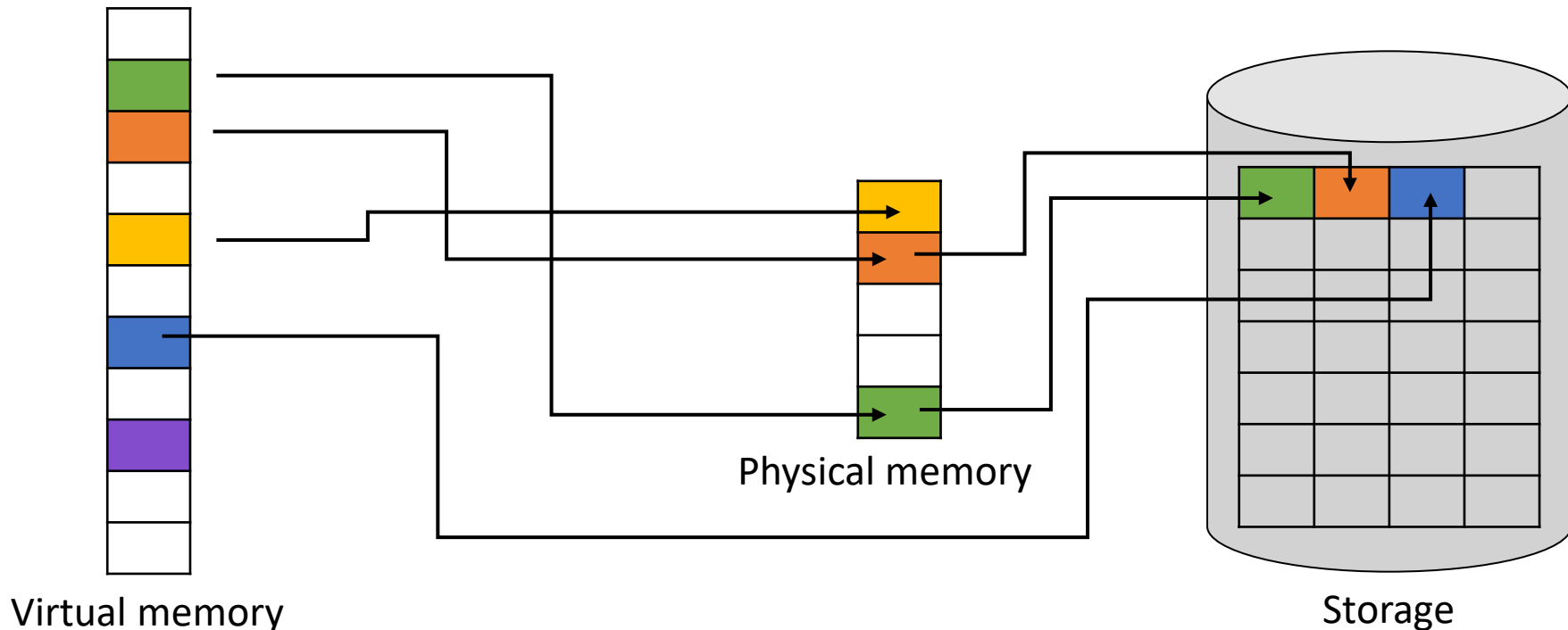
# Memoria virtuale

- Il livello di multiprogrammazione impatta l'utilizzazione delle risorse
  - Massimizzare il numero di processi ready-to-run
- Segmentazione e paginazione facilitano tale compito
  1. traduzione da indirizzo logico a fisico a tempo di esecuzione
  2. processi occupano frammenti in memoria non contigui
- Swap out/in di frammenti di processo
  - Maggior numero di processi in memoria principale
  - Un processo è ready-to-run anche se un qualche frame non è presente in memoria principale
  - Un processo può richiedere più memoria di quanta disponibile in memoria principale

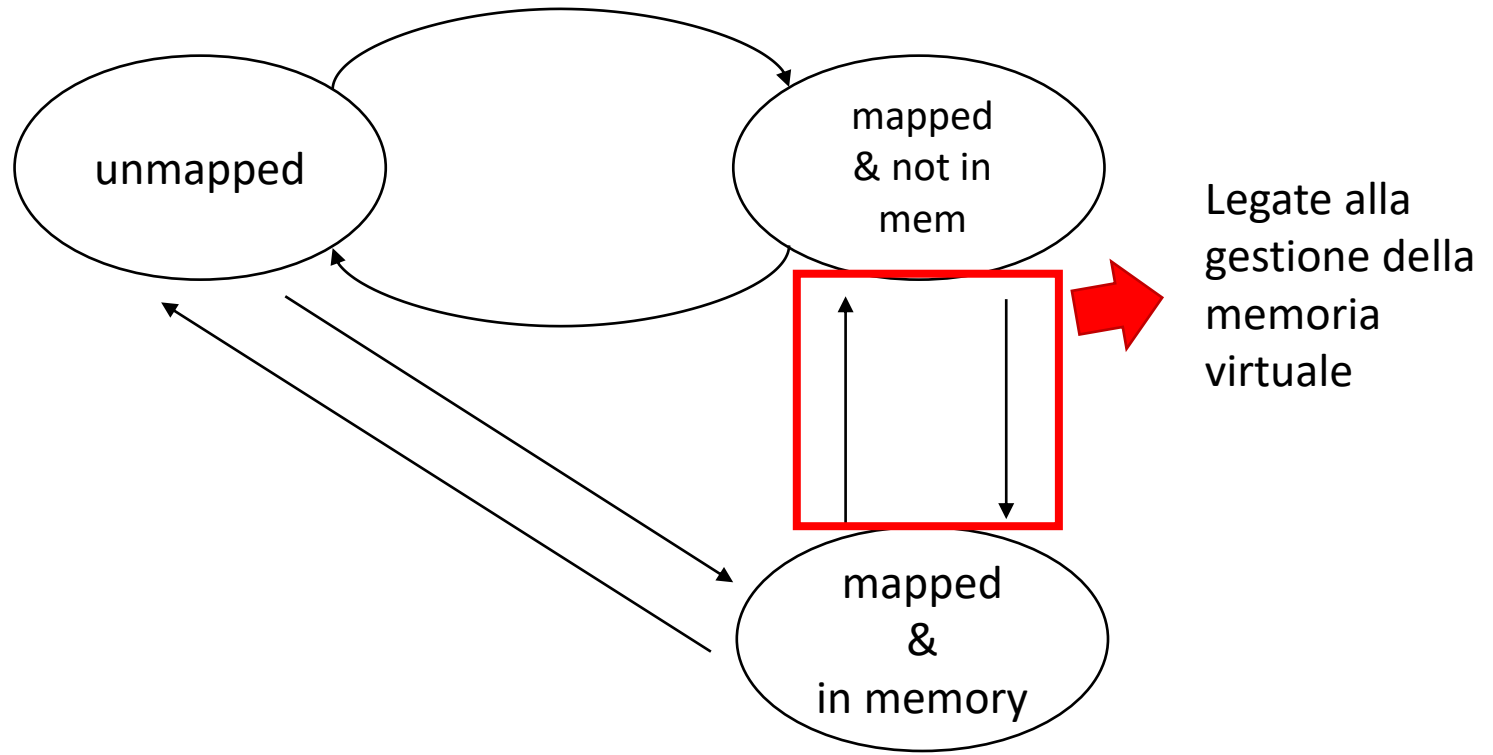


# Memoria virtuale

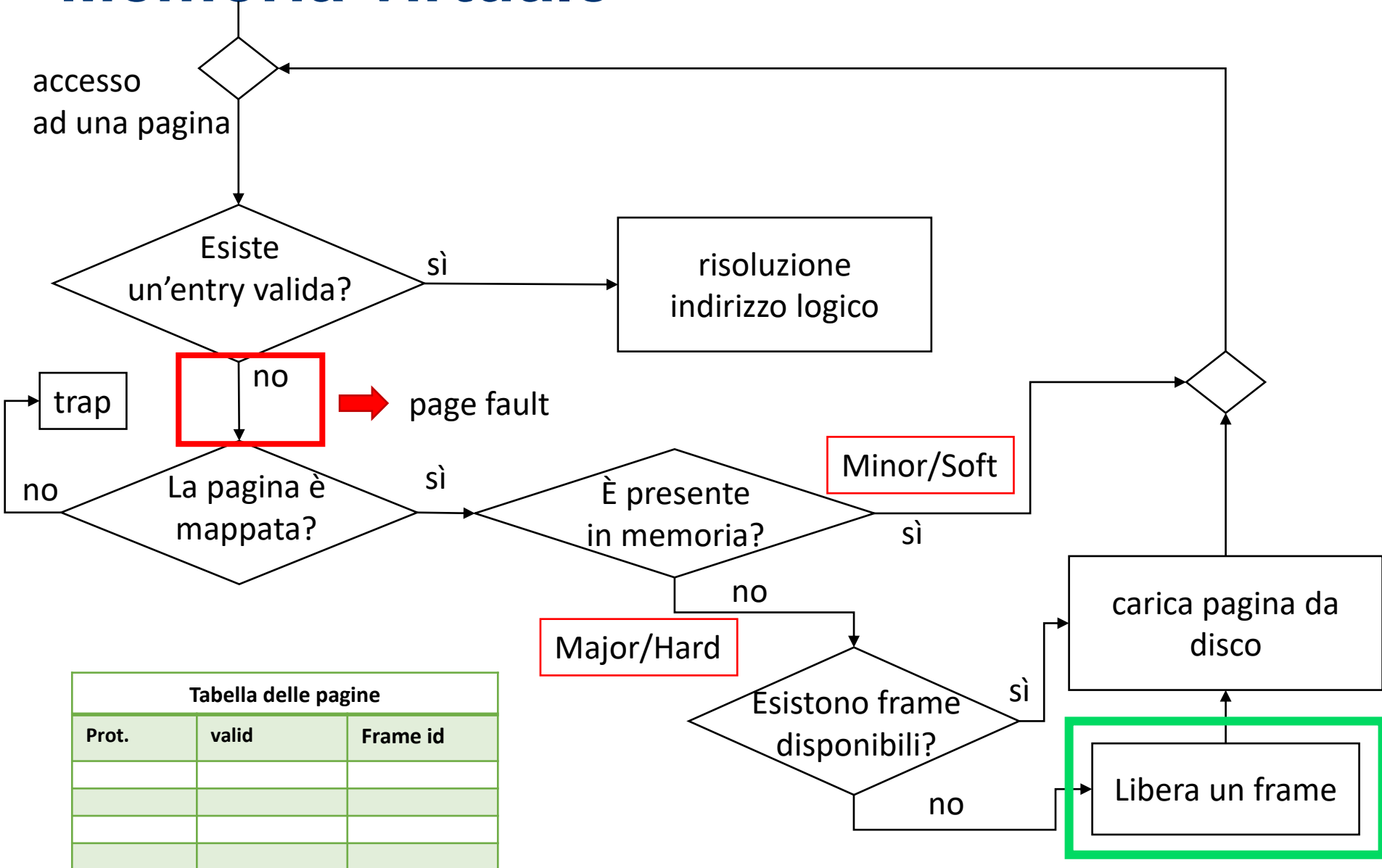
- Memoria Reale
  - Memoria principale effettivamente accessibile dal processore
- Memoria Virtuale
  - Memoria che un processo può richiedere al sistema operativo
  - Allocata su dispositivi di storage secondario (e.g., disco) **E** su memoria principale



# Stato di una pagina



# Memoria virtuale



# Liberazione di un frame

- Necessario individuare una pagina da rimuovere da memoria
  - Algoritmo di replacement per la selezione della pagina vittima
- La pagina rimossa da memoria potrebbe esser richiesta in seguito
  - Necessità di salvare il contenuto del frame su dispositivo di storage
- **Ottimizzazione:**
  - la pagina è già presente su disco
  - il contenuto del frame è invariato
  - non è necessario trasferire la pagina su disco e il frame può essere riutilizzato
- Tenere traccia di accessi in scrittura alla pagina
  - Dirty bit nella tabella delle pagine

# Performance della memoria virtuale

- Nonostante l'utilizzo di I/O la memoria virtuale è un meccanismo efficace
  - Tempo medio di accesso in memoria:  $m_a$  (centinaia di nanosecondi)
  - Probabilità di page fault:  $p$
  - Tempo medio di gestione page fault:  $m_f$  (decine di millisecondi)
  - Tempo effettivo di accesso a memoria:  $m_a + p \cdot m_f$
  - Esempio:
    - $m_a=100\text{ns}$ ,  $m_f=10\text{ms}$
    - Overhead < 10% implica  $p < 10^{-6}$
- Sfruttare proprietà di località
  - Un processo tende a concentrare accessi a dati e/o codice

# Gestione della memoria virtuale

- Mantenere in memoria principale un numero ridotto di pagine (**Resident Set**) per processo
  - Massimizzare il livello di multiprogrammazione
- Minimizzare la frequenza di page fault
- Alcuni aspetti di gestione:
  - Politica di caricamento delle pagine
  - Politica di posizionamento delle pagine
  - Politica di sostituzione delle pagine
  - Gestione del resident set

# Caricamento e posizionamento delle pagine

## Caricamento

- Demand paging
  - Una pagina viene caricata se e solo se è richiesta
- Prepaging
  - Caricare da disco più pagine consecutive in blocco è più efficiente rispetto a caricare una pagina alla volta
  - È possibile che pagine caricate non vengano referenziate

## Posizionamento

- Grazie alla paginazione è irrilevante rispetto a problematiche di frammentazione
- Rilevante in termini di affinity (e.g., NUMA)

# Sostituzione delle pagine

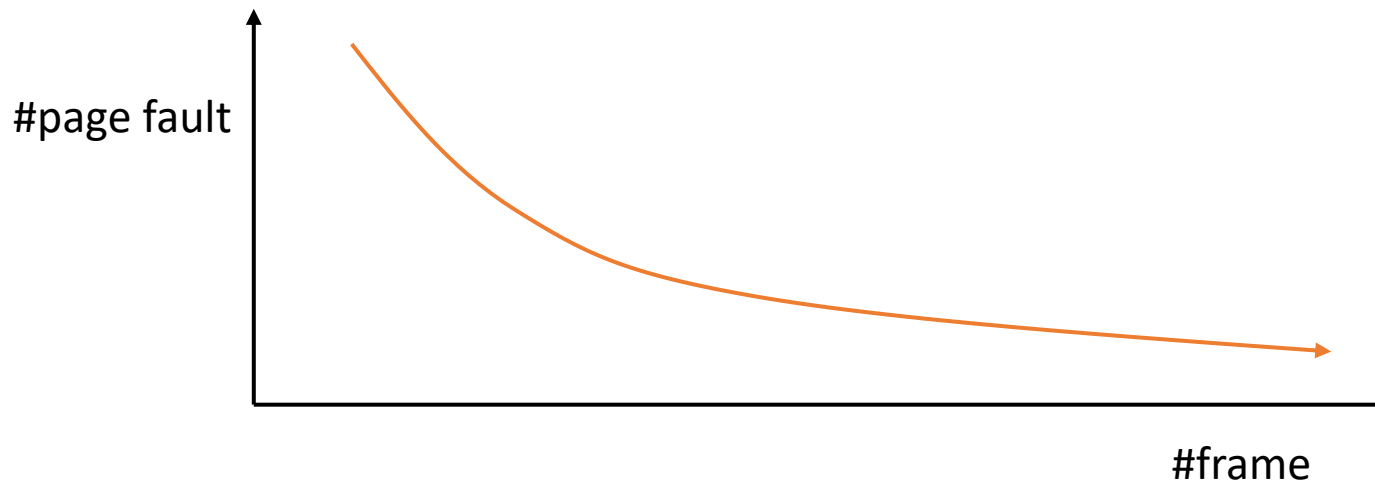
Alcuni aspetti coinvolti nella sostituzione

1. Quanti frame mantenere al più per processo
  2. Le pagine candidate per la sostituzione sono limitate al processo che ha generato il fault oppure pagine in qualsiasi frame
  3. Dato un insieme di pagine che possono essere sostituite
- I punti 1 e 2 rientrano nella gestione del resident set
  - Il punto 3 è la policy per la sostituzione delle pagine



# Sostituzione delle pagine


- Metrica di performance
  - frequenza di hard/major page fault
- Modalità di valutazione
  - Si prende una traccia di riferimenti ad indirizzi logici
    - Generata casualmente o basata su esecuzione reale
- Aspettativa



# Algoritmo FIFO

- Seleziona la pagina presente da più tempo in memoria
- Semplice da implementare
- Non favorisce la località

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0
7	0	1	2	2	3	0	4	2	3	0	0	0	1	2	0	0	7	0
	7	0	1	1	2	3	0	4	2	3	3	3	0	1	2	2	2	7
		7	0	0	1	2	3	0	4	2	2	2	3	0	1	1	1	2



# Anomalia di Belady

- Il numero di page fault può incrementare all'aumentare del numero di frame

0	1	2	3	0	1	4	0	1	2	3	4
0	1	2	3	0	1	4	4	4	2	3	3
	0	1	2	3	0	1	1	1	4	2	2
		0	1	2	3	0	0	0	1	4	4



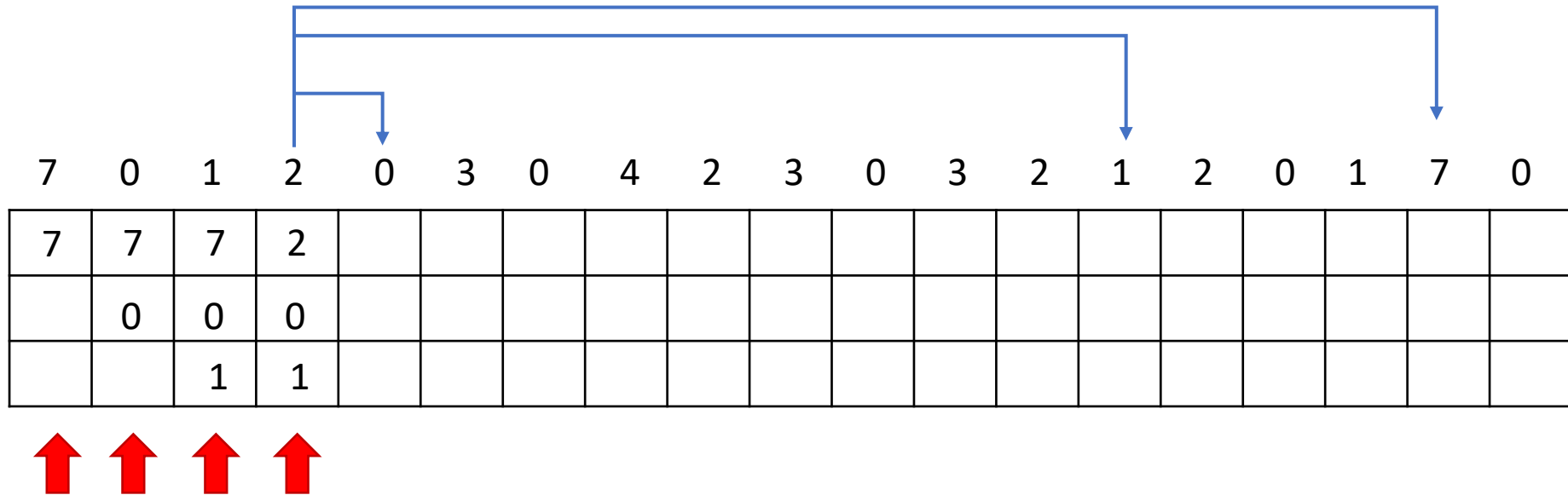
9 page fault

0	1	2	3	0	1	4	0	1	2	3	4
0	1	2	3	3	3	4	0	1	2	3	4
	0	1	2	2	2	3	4	0	1	2	3
		0	1	1	1	2	3	4	0	1	2
			0	0	0	1	2	3	4	0	1

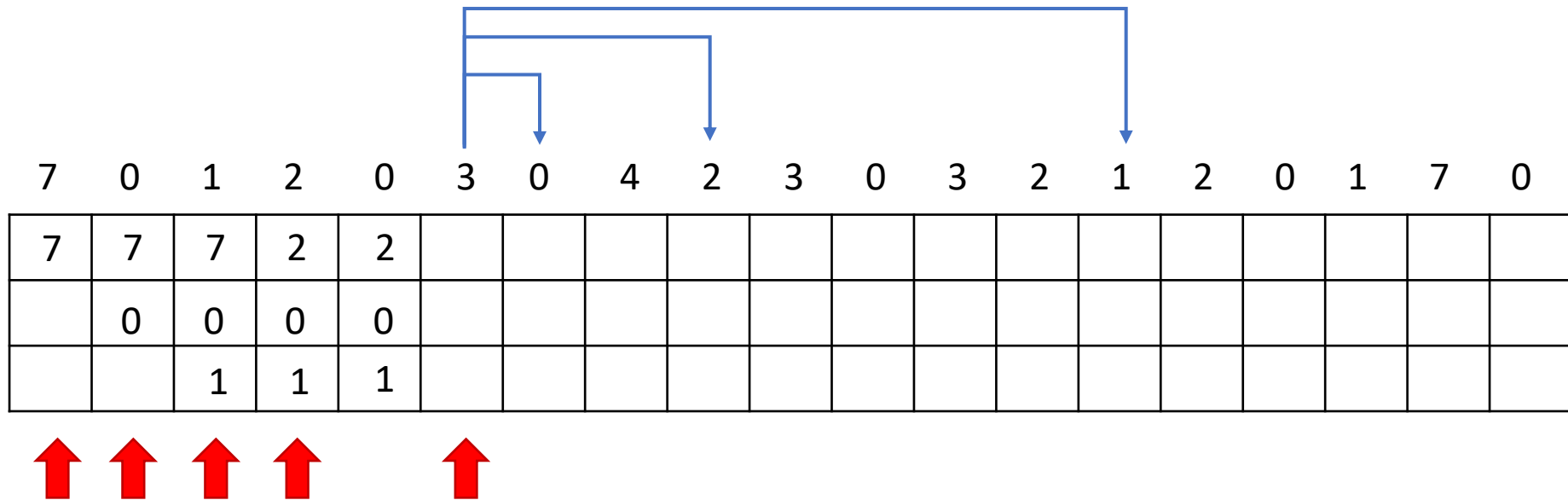


10 page fault

# Algoritmo ottimo



# Algoritmo ottimo



# Algoritmo ottimo

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	4	4	0	0	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	3	3	3	3	1	1	1	1	1	1	1

↑ ↑ ↑ ↑            ↑            ↑            ↑            ↑            ↑

# Algoritmo ottimo

- Seleziona la pagina che non **sarà** riferita per più tempo
- Impossibile da implementare
- Algoritmo di riferimento

**Principio di località:**  
Pagine accedute di recente  
hanno una probabilità  
maggiore di essere accedute  
nel prossimo futuro


7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	4	4	0	0	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	3	3	3	3	1	1	1	1	1	1	1

↑ ↑ ↑ ↑            ↑            ↑            ↑            ↑

# Algoritmo Least-Recently Used

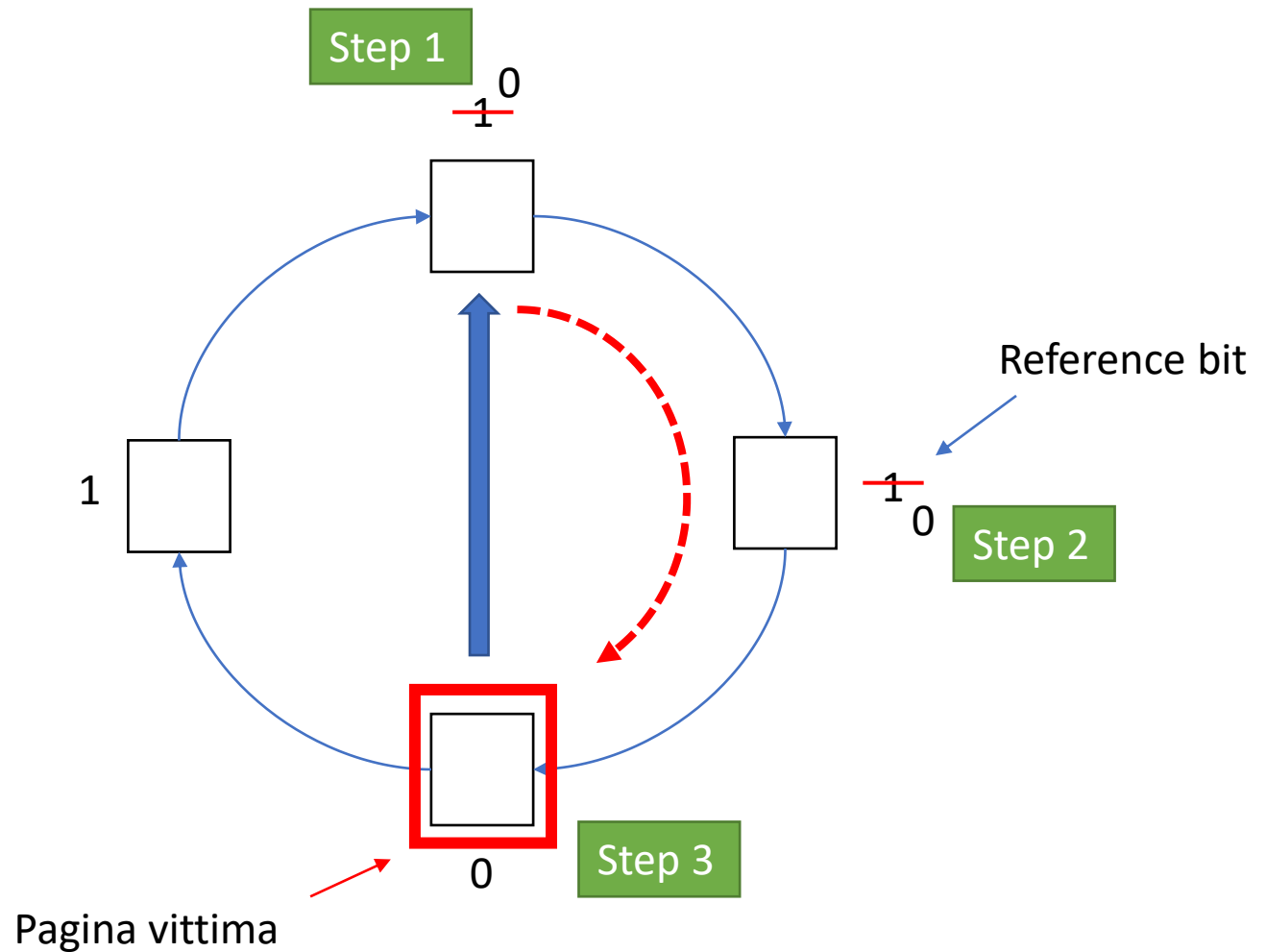
- Seleziona la pagina che non è riferita per più tempo
- Costoso da implementare
- È possibile approssimarne il comportamento

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0
7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0
	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7
		7	0	1	2	2	3	0	4	2	2	0	3	3	1	2	0	1





# Algoritmo dell'orologio



# Algoritmo dell'orologio

- Reference bit (RB): indica che la pagina è stata referenziata dopo l'ultimo reset
- Dirty bit (DB): la pagina è stata acceduta in scrittura dopo il caricamento
- (RB, DB):
  - (0,0)
  - (1,0)
  - (0,1)
  - (1,1)

# Gestione del resident set

- Taglia del resident set
  - Minore la taglia maggiore è il livello di multiprogrammazione
  - Minor taglia maggiore è la frequenza di page fault
  - Due approcci:
    - Allocazione fissa
    - Allocazione variabile
- Definizione dell'insieme delle pagine candidate per il rimpiazzamento
  - Locale
  - Globale

# Working set

- L'obiettivo è quello di caratterizzare le pagine rilevanti per l'attività di un processo
- È necessario indentificare un periodo di osservazione
  - Le pagine referenziate nel periodo di osservazione fanno parte del working set
  - Il numero di pagine referenziate è la taglia del working set
- Obiettivo:
  - Stimare la taglia del working set ed usarla per dimensionare il resident set
- Criticità:
  - Il passato non è sempre un esempio per predire il futuro
  - Una misura precisa del working set richiede di tracciare ogni singolo accesso a memoria
  - Difficile impostare il periodo di osservazione in modo ottimo

# Page fault frequency

