

Università di Roma Tor Vergata  
Corso di Laurea triennale in Informatica  
**Sistemi operativi e reti**

A.A. 2022-2023

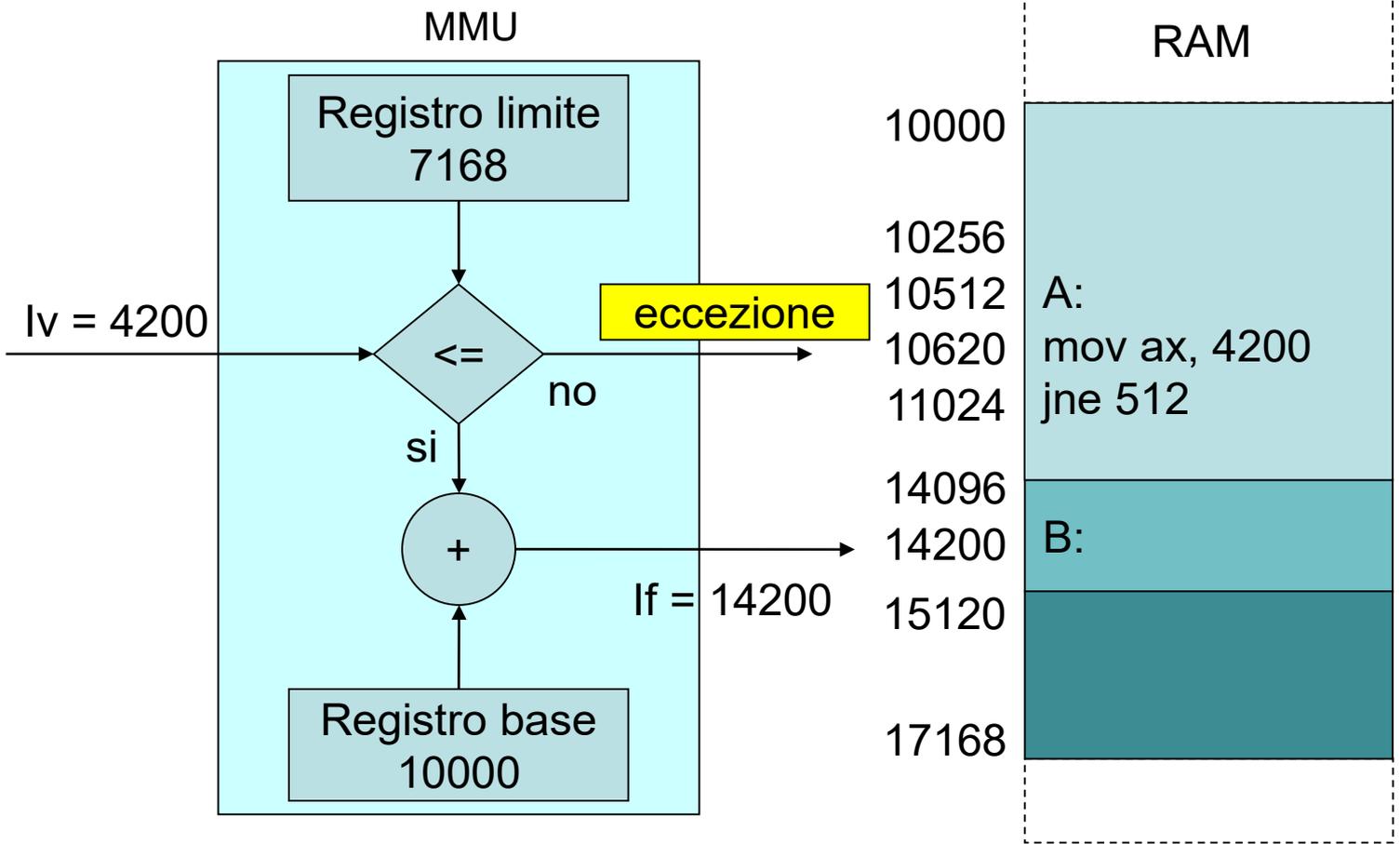
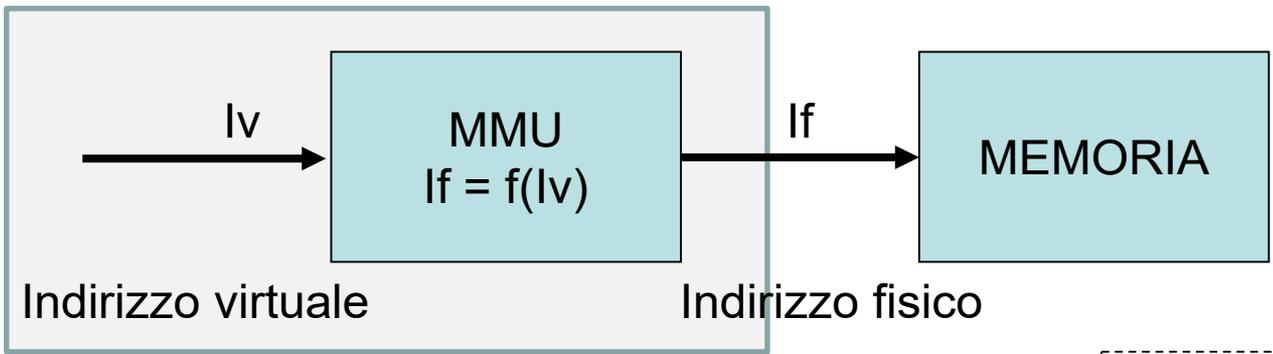
Pietro Frasca

# Lezione 17

Giovedì 1-12-2022

# Protezione e condivisione di informazioni con spazio virtuale unico

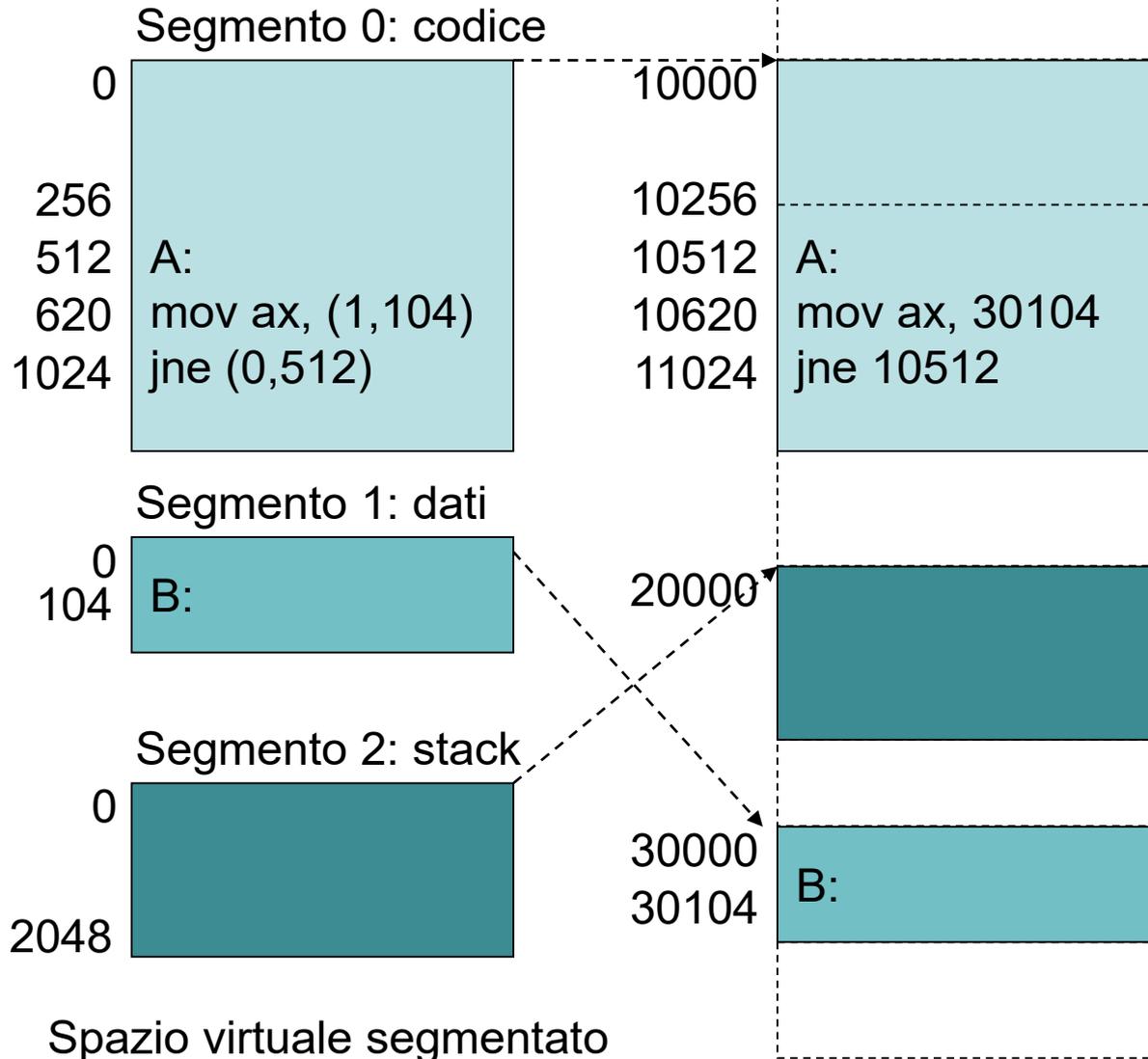
- Il problema di garantire la **protezione** ai processi allocati contemporaneamente in memoria è spesso risolto a livello di architettura del processore, ad esempio utilizzando due particolari registri, il **registro base** e il **registro limite**, contenenti rispettivamente l'indirizzo iniziale e finale della partizione assegnata al processo in esecuzione.
- Ogni indirizzo generato dal processore è confrontato con i valori presenti nei due registri e nel caso in cui l'indirizzo sia al di fuori dell'intervallo delimitato dai due registri viene generata una interruzione di errore.
- Per quanto riguarda la **condivisione** di informazioni, il fatto di allocare in locazioni contigue uno spazio virtuale unico non consente ai processi di condividere informazioni.



# Partizioni multiple

- La tecnica delle **partizioni multiple** richiede che lo spazio virtuale di un processo sia segmentato.
- Questa tecnica utilizza la **rilocazione statica** che consiste nel rilocare l'intero spazio virtuale del processo prima che questo inizi la sua esecuzione.
- Lo spazio virtuale del processo è diviso in vari segmenti, ad esempio nei tre segmenti di codice, dati e stack.
- Con questa tecnica, a ogni processo è allocato un numero di partizioni pari al numero di segmenti in cui è stato diviso lo spazio virtuale.
- I segmenti non necessariamente sono allocati in memoria in modo adiacente.
- Inoltre, essendo i segmenti separati più piccoli, rispetto al caso dello spazio virtuale unico, sarà più facile allocarli in memoria. La frammentazione esterna tende quindi a diminuire.

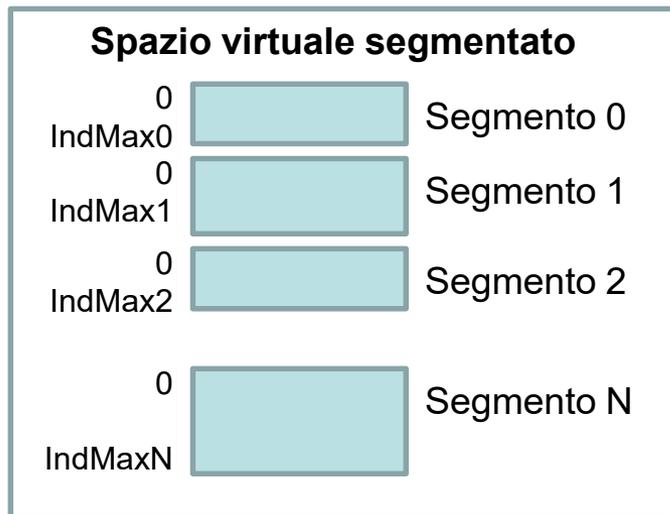
# Caricatore rilocante



- Per eliminare completamente la frammentazione della memoria bisognerebbe unire tutte le partizioni libere e spostare anche le partizioni allocate ai processi, in modo da poter ottenere un'unica partizione libera contigua. Ma per spostare in memoria un processo è necessario ricorrere a tecniche di rilocazione dinamica, ad esempio usando il supporto MMU. *Tale supporto, nel caso di processi con tre segmenti, deve possedere tre coppie di registri base/limite, una coppia per ogni segmento.*

# Memoria segmentata

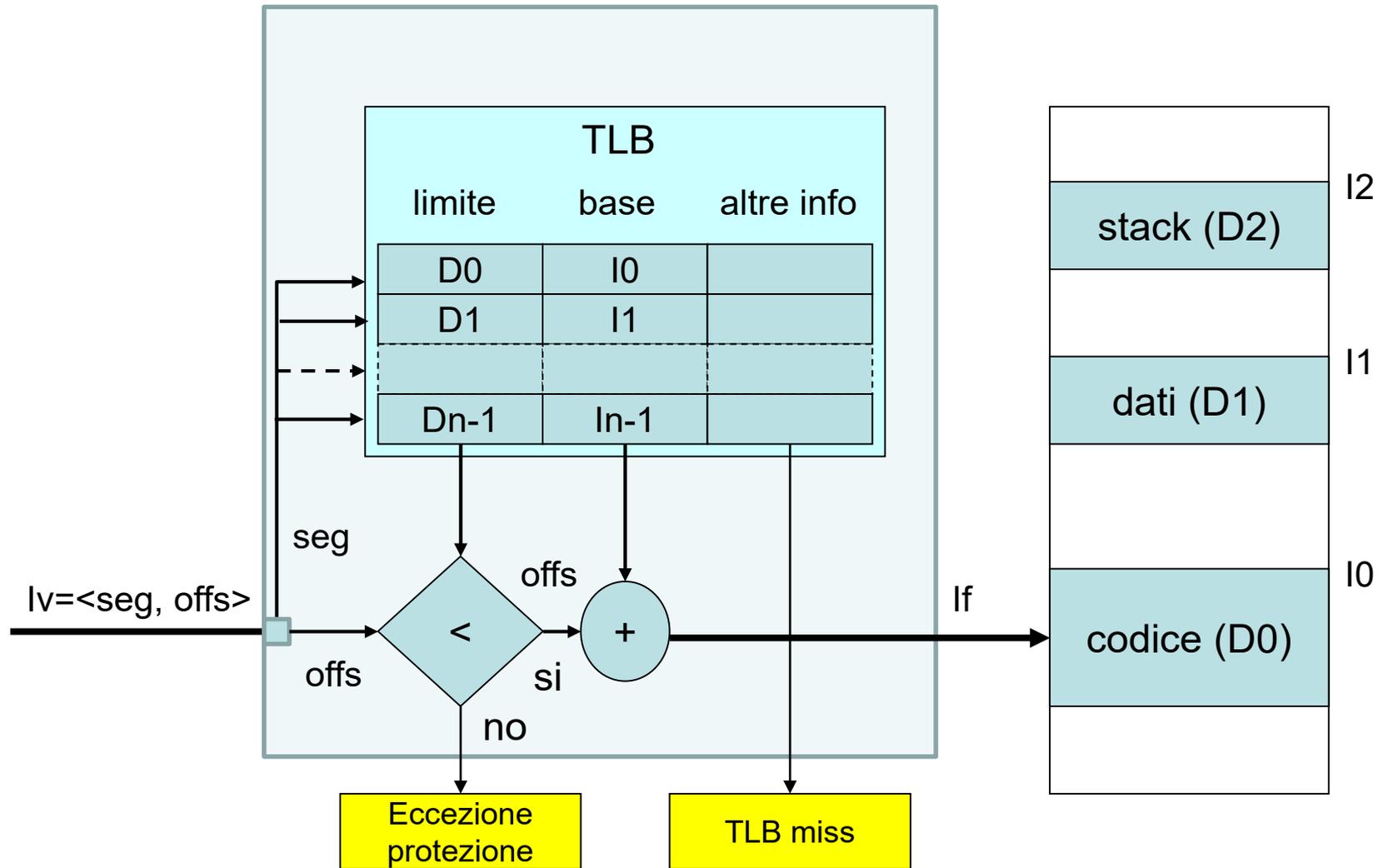
- Nei sistemi che utilizzano la segmentazione, è possibile strutturare lo spazio virtuale di un processo assegnando un segmento per ogni modulo di programma quali le funzioni, procedure, strutture dati etc. In tal modo, lo spazio virtuale del processo, è costituito da molti segmenti che rispecchiano semanticamente la struttura del programma sorgente.



- Nelle architetture di processori con supporto di segmentazione, gli indirizzi virtuali sono generati con il seguente formato bidimensionale:

**$lv = \langle \text{segmento}, \text{offset} \rangle$**

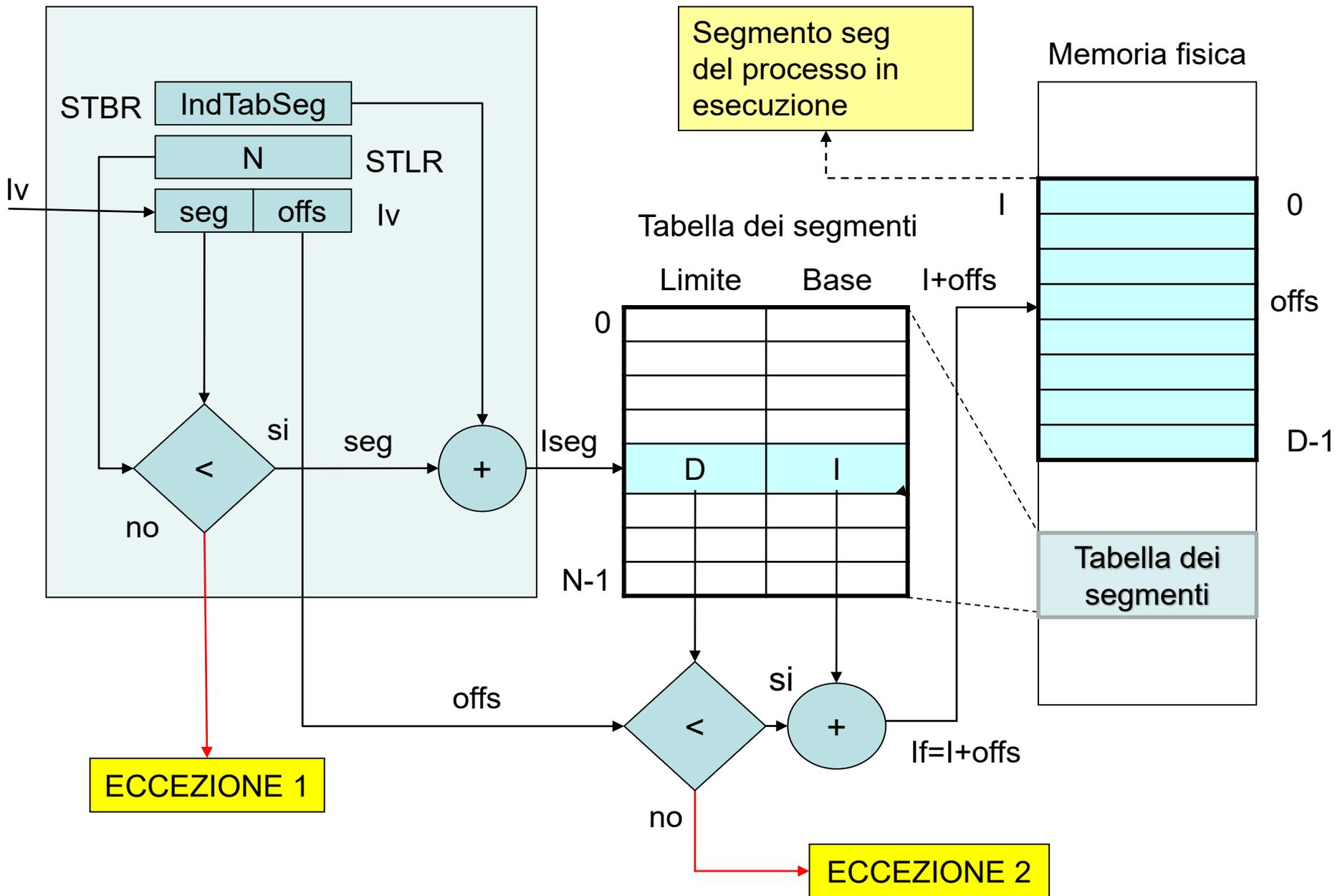
- Poiché i processori hanno un buffer (**TLB, Translation Lookaside Buffer**) composto da pochi registri associativi, tipicamente da 32 a 4096, in cui memorizzare i valori limite e base, nel caso in cui il numero di segmenti è elevato non è più possibile memorizzare nella TLB i dati relativi a tutti i segmenti.
- Ad esempio i microprocessori Intel della famiglia IA-32 hanno uno spazio virtuale che può avere fino a  $2^{14}$  (16384) segmenti.



## Traduzione degli indirizzi con TLB

- Pertanto i valori relativi all'indirizzo base e limite di ogni segmento sono memorizzati in una tabella, detta **tabella dei segmenti**, contenuta nella memoria principale e gestita dal kernel.
- Il descrittore del processo (PCB) contiene, oltre alle informazioni già descritte, due campi relativi al suo spazio virtuale. Il primo contiene il **numero di segmenti** in cui è suddiviso lo spazio virtuale e il secondo memorizza l'**indirizzo fisico della tabella dei segmenti**.
- Questi due valori sono caricati in due registri del processore, quando il processo passa in esecuzione. I due registri sono spesso indicati con i termini **STLR (Segment Table Limit Register)** e **STBR (Segment Table Base Register)**.





- L'uso della tabella dei segmenti caricata in RAM rallenta la funzione di traduzione degli indirizzi rispetto al caso in cui i valori base e limite di ogni segmento sono contenuti nella veloce TLB del processore.
- Per accedere ad una locazione di memoria sono ora **necessari due accessi**: il primo accesso alla tabella dei segmenti e il secondo alla locazione vera e propria.
- Tuttavia, anche se i processori hanno una **TLB** composta da pochi registri associativi, la traduzione degli indirizzi si ottiene per la maggior parte attraverso essa, senza accedere alla tabella in RAM.
- Infatti, quando la CPU genera un indirizzo virtuale  **$iv = \langle \text{seg}, \text{offset} \rangle$**  la sua traduzione avviene dapprima ricercando i dati riguardanti il segmento **seg** nella TLB. Se i dati sono presenti nella TLB la traduzione degli indirizzi si ottiene in base ai valori base e limite presenti nel registro associativo, altrimenti la traduzione avviene in base allo schema della figura precedente e nella TLB sono copiati i dati relativi alla traduzione.

- Giacché un programma è strutturato in moduli, gli accessi in memoria sono spesso localizzati. Pertanto, si ha che con pochi registri associativi si possono tradurre l'80% degli indirizzi.
- La percentuale di volte che la traduzione dell'indirizzo avviene con la TLB è detta **hit ratio**. Un hit ratio pari a 0.8 significa che l'80% delle traduzioni è risolto con la TLB.
- I tempi di ricerca nelle TLB attuali sono praticamente trascurabili.
- Ad esempio, se sono necessari 10 ns per accedere alla memoria, allora nel caso in cui la traduzione avviene con la TLB, un accesso ad una locazione di memoria, per un'istruzione o un dato, richiede 10 ns. Se, invece, la TLB non contiene la traduzione occorrono 10 ns per accedere alla tabella dei segmenti più 10 ns per accedere al dato desiderato in memoria; in totale sono necessari  $10+10=20$  ns.
- Per calcolare **il tempo effettivo di accesso** alla memoria bisogna tener conto della probabilità dei due casi:

$$\text{Tempo effettivo d'accesso} = p \cdot (T_{\text{TLB}} + T_M) + (1-p) \cdot (T_{\text{TLB}} + 2 \cdot T_M)$$

Per l'esempio precedente si ha (trascurando  $T_{\text{TLB}}$ ):

$$\text{Tempo effettivo d'accesso} = 0.80 \cdot 10 + 0.20 \cdot 20 = 12 \text{ ns}$$

- La segmentazione dello spazio virtuale complica notevolmente la funzione di traduzione degli indirizzi ma d'altra parte presenta grandi vantaggi.
- Due importanti vantaggi prodotti dalla segmentazione sono relativi alla **protezione** e alla **condivisione** dei segmenti.
- La segmentazione consente di ottenere vari tipi di controllo quando un processo accede alla memoria.
- Oltre ai due controlli di protezione evidenziati nella figura precedente, ad ogni segmento possono essere associati **vari diritti di accesso**, come ad esempio il diritto in accesso in **sola lettura** (segmento di codice, segmento contenente solo costanti), **scrittura** (segmento dati), **lettura e scrittura** (segmento dati) etc.

- Per consentire questi controlli nella tabella dei segmenti si aggiungono nuovi campi, come mostrato nella figura seguente.

base	limite	controllo		altre info
Indirizzo del segmento	dimensione del segmento	R	W	

elemento della tabella del segmento

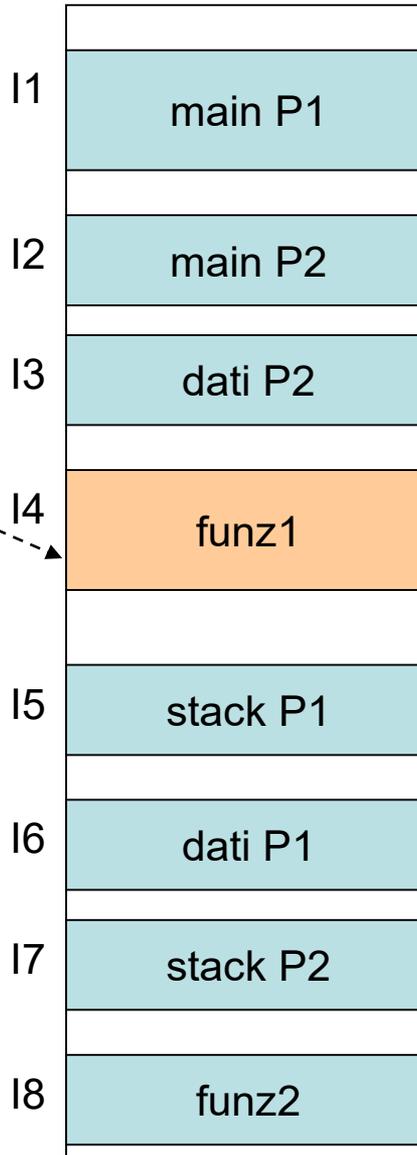
- Come già detto, la segmentazione consente una corrispondenza semantica tra moduli di un programma (codice sorgente) e segmenti. Pertanto è possibile condividere alcuni segmenti tra più processi, come mostrato nella figura seguente. Nella figura sono rappresentati gli spazi virtuali di due processi P1 e P2 mediante le relative tabelle dei segmenti. P1 ha 5 segmenti, P2 ne ha 4. Il segmento **funz1** è condiviso.

# Esempio di segmento condiviso

memoria

	base	limite	controllo	
main	I1	D1		0
funz1	I4	D4		1
funz2	I8	D4		2
dati	I6	D6		3
stack	I5	D5		4

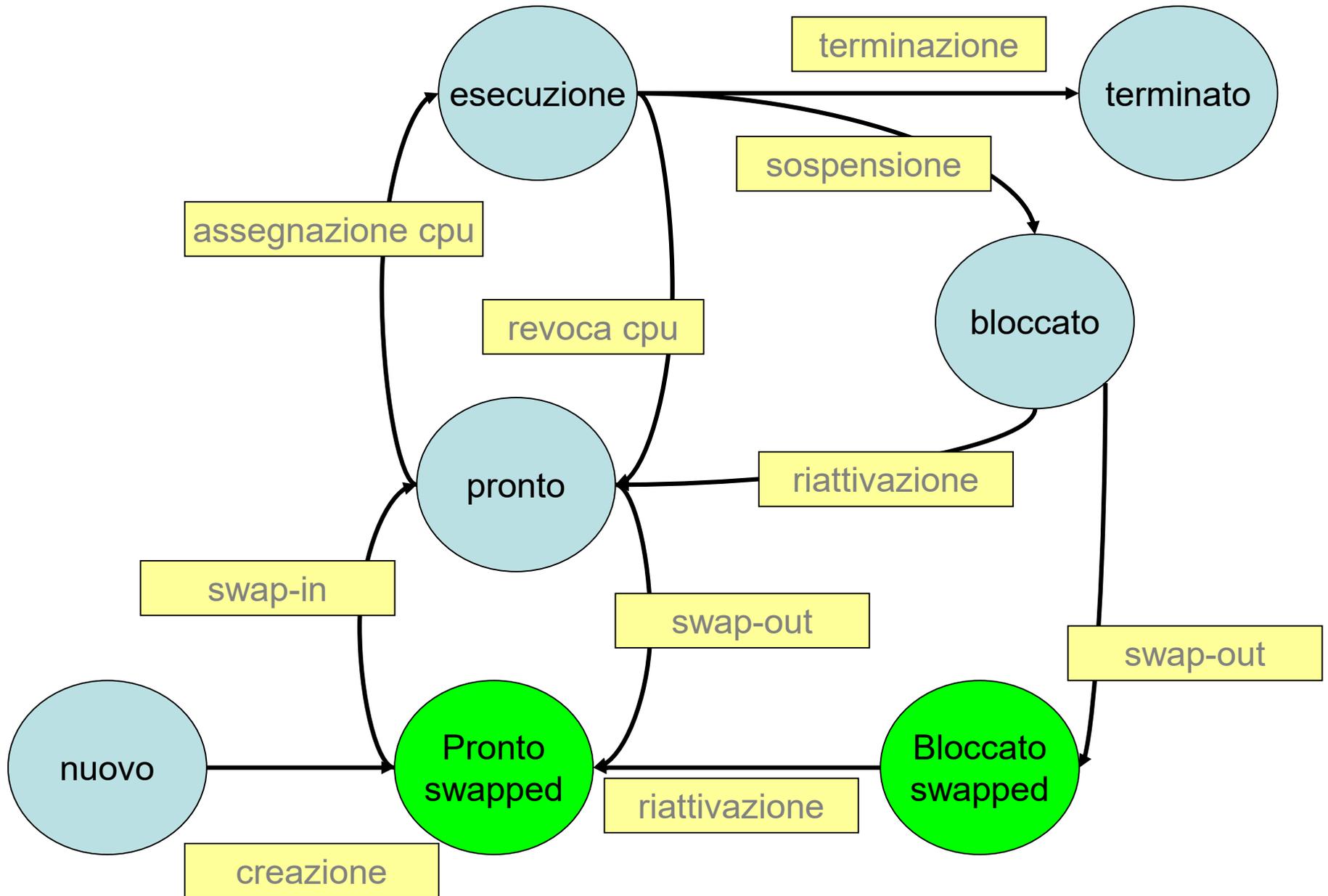
Tabella dei segmenti di P1



	base	limite	controllo	
main	I2	D2		0
funz1	I4	D4		1
dati	I3	D3		2
stack	I7	D7		3

Tabella dei segmenti di P2

- Da quanto fin ora descritto, con la tecnica della segmentazione un processo si può trovare in due condizioni:
  1. **Allocato in memoria**  
tutti i suoi segmenti sono allocati nella memoria fisica
  2. **Non allocato in memoria**  
tutti i suoi segmenti sono nella swap-area, su memoria secondaria.
- Per tener conto di questa situazione è necessario aumentare gli stati in cui si può trovare un processo.
- Si aggiungono due stati detti **pronto-swapped** e **bloccato-swapped** per rappresentare i processi quando non hanno lo spazio virtuale allocato in memoria fisica.
- Ricordiamo che le transizioni assegnazione CPU, dallo stato di pronto allo stato di esecuzione e quella dallo stato di pronto-swapped allo stato di pronto (swap-in) sono eseguite rispettivamente dallo **scheduler a breve termine** e dallo **scheduler a medio termine**.

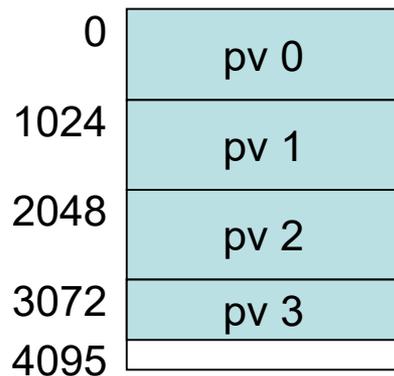


- Il caricamento in memoria dell'intero spazio virtuale, prima dell'esecuzione del processo, implica che la dimensione dello spazio virtuale debba essere inferiore alla dimensione della memoria fisica disponibile.

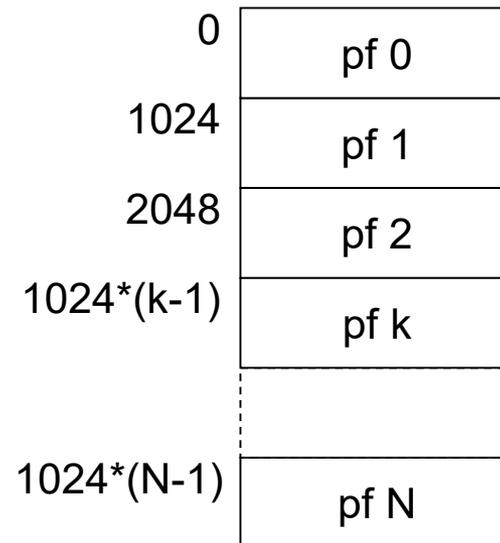
# Memoria paginata

- La gestione della memoria fisica mediante partizioni ha lo svantaggio di produrre la frammentazione della memoria.
- La tecnica della **memoria paginata** consiste nel suddividere lo spazio virtuale del processo in blocchi di dimensione fissa detti **pagine virtuali**. La dimensione delle pagine è una **potenza di 2** (esempio  $2^{10}=1024$ ) ed è stabilita dall'architettura del processore.
- Anche la memoria fisica è partizionata in **pagine fisiche** (dette anche **frame**) della **stessa dimensione** delle pagine virtuali.

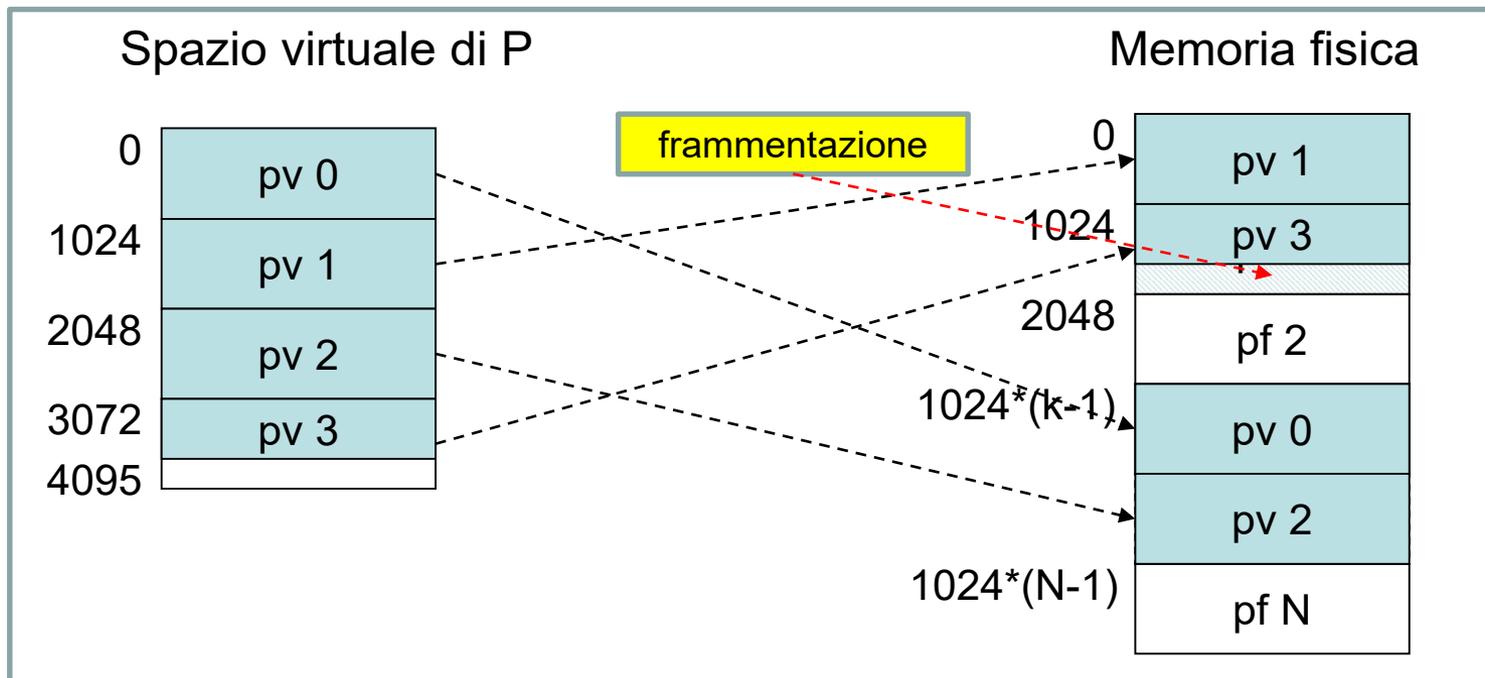
Spazio virtuale di P



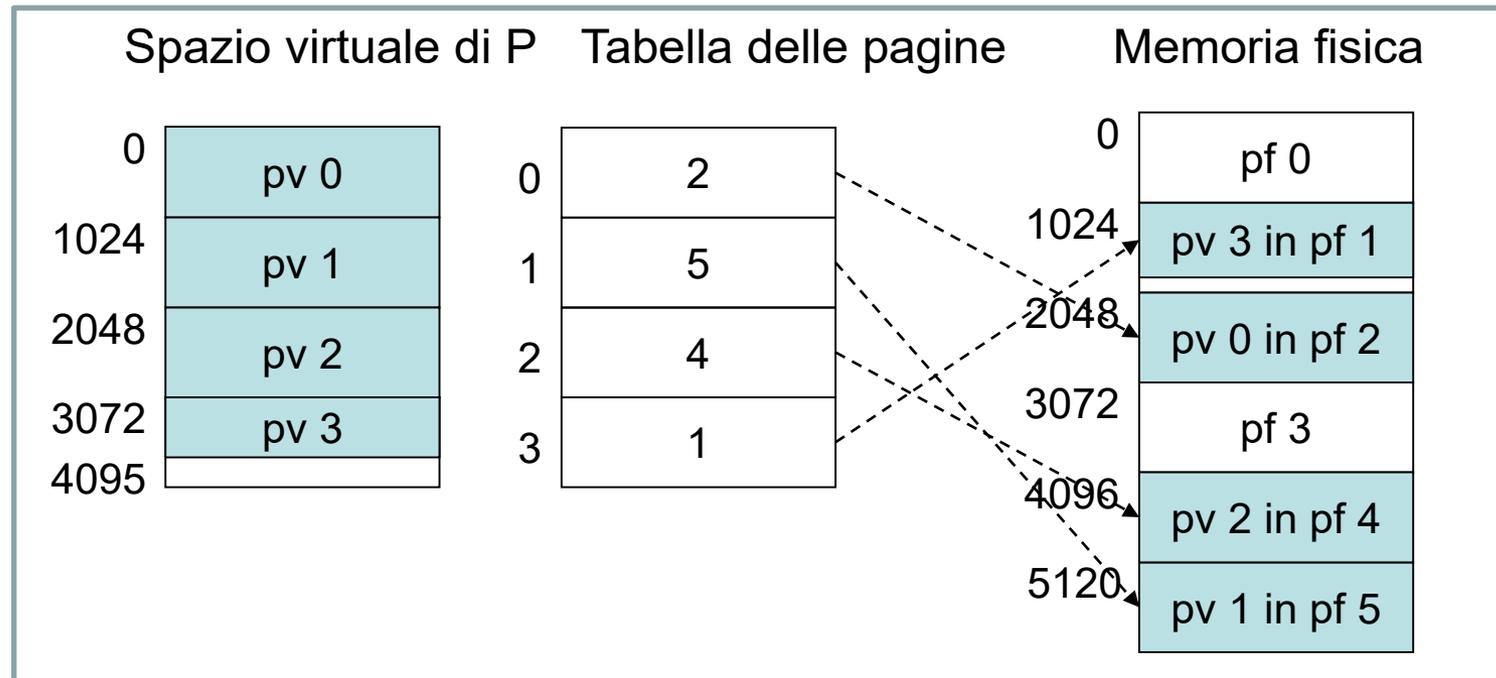
Memoria fisica



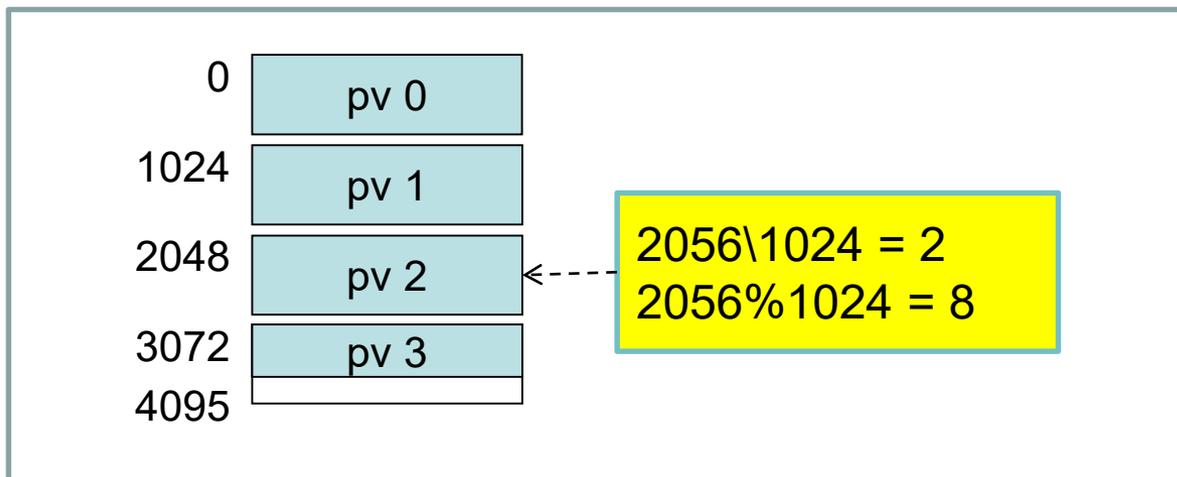
- In tal modo ciascuna pagina virtuale di un processo può essere allocata in memoria in qualsiasi pagina fisica e in modo indipendente.
- La pagina fisica, allocata per l'ultima pagina virtuale del processo potrebbe non essere completamente utilizzata. Infatti è molto difficile che la dimensione dello spazio virtuale di un processo sia un multiplo intero della dimensione della pagina. In tal caso si produce frammentazione interna.



- Per **ogni processo**, il kernel, gestisce una **tabella delle pagine** che realizza la funzione di traduzione degli indirizzi tra le pagine virtuali e le pagine fisiche nelle quali le pagine virtuali sono caricate.
- L'indirizzo della tabella delle pagine e il numero di pagine di cui è costituito lo spazio virtuale di un processo sono memorizzate in campi del descrittore del processo (PCB).



- La dimensione della tabella delle pagine dipende dalla dimensione dello spazio virtuale del processo e dalla dimensione delle pagine.
- In pratica, per tradurre un indirizzo virtuale **lv** è necessario sapere a quale pagina virtuale esso appartiene. Dividendo l'indirizzo virtuale per la dimensione **D** della pagina si ha che:
  - **Il quoziente** della divisione è **l'indice** della pagina virtuale;
  - **Il resto** della divisione è **l'offset** dell'indirizzo all'interno della pagina.
- Ad esempio se la pagina è di dimensioni di 1024 byte, **l'indirizzo 2056** appartiene alla **pagina 2** ed ha **offset pari a 8** (la numerazione delle pagine inizia da 0).



- Per gestire lo stato di allocazione della memoria fisica, il gestore della memoria mantiene aggiornata una tabella, detta **tabella delle pagine fisiche**, che ha un numero di righe pari al numero delle pagine fisiche in cui è suddivisa la memoria.
- L'elemento della tabella indica se la pagina fisica è libera o occupata. Se la pagina è occupata l'elemento contiene due campi: **il pid del processo** a cui appartiene la pagina e **l'indice pv** della pagina virtuale corrispondente.

	pagina pid virtuale	
0	P1	1
1	P2	6
2	free	
3	P3	4
k	free	
N-1	P2	10

**Tabella delle pagine fisiche**