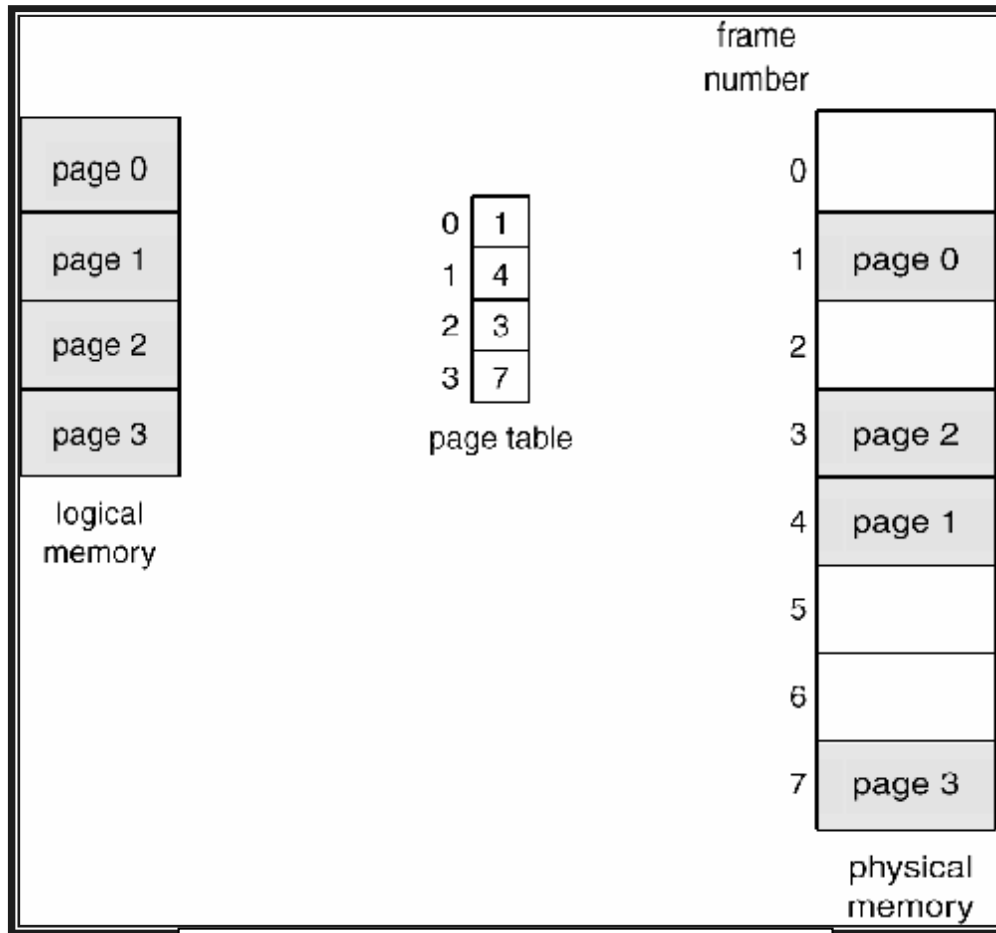


Gestione della memoria

- Paginazione
- Segmentazione
- Segmentazione con paginazione

Modello di paginazione

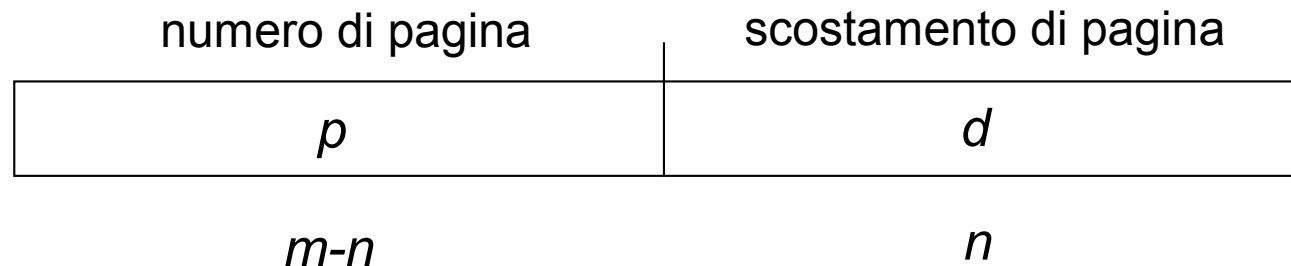


Modello di paginazione di memoria logica e memoria fisica

- Il numero di pagina serve come indice per la **tabella delle pagine**.
- Questa contiene l'indirizzo di base nella memoria fisica di ogni pagina.
- L'indirizzo di base si appaia con lo scostamento di pagina per definire l'indirizzo della memoria fisica.

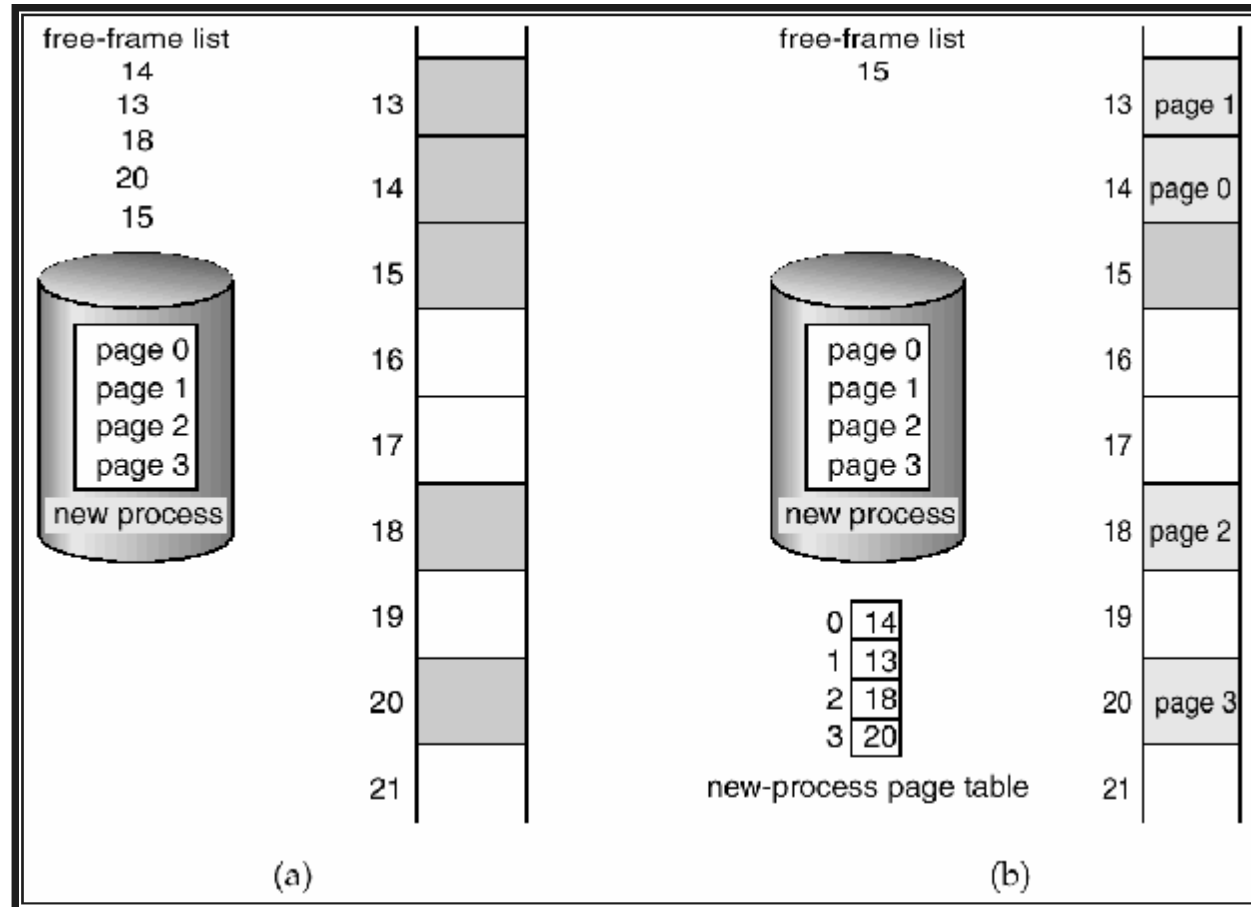
Dimensione delle pagine

- La dimensione di una pagina, così come quella di un frame, è definita dall'architettura del calcolatore ed è tipicamente una potenza di 2 compresa tra 512 byte e 16 MB.
- Le dimensioni sono potenza di 2 perché ciò facilita la traduzione degli logici in numero di pagina e scostamento.
- Se lo spazio degli indirizzi logici è 2^m e la dimensione di una pagina è 2^n , allora:
 - ◆ gli $m-n$ bit più significativi di un indirizzo logico indicano il **numero di pagina**,
 - ◆ gli n bit meno significativi indicano lo **scostamento**.



Esempi di paginazione

Frame liberi



Prima dell'allocazione

Dopo l'allocazione

Tabella dei blocchi di memoria

- Nella paginazione vi è una netta differenza tra memoria fisica e memoria logica: per il programma utente la memoria è un unico spazio contiguo.
- In realtà il programma è sparso nella memoria fisica, che contiene anche altri programmi.
- La differenza tra memoria fisica e memoria logica è contenuta nell'architettura di traduzione degli indirizzi, che fa corrispondere indirizzi fisici e logici.
- Queste operazioni sono controllate dal sistema operativo, che mantiene le informazioni sull'assegnazione in una ***tabella dei blocchi di memoria***.

Architettura di paginazione

- Ciascun sistema operativo segue strategie differenti per memorizzare le tabelle delle pagine, sebbene in genere venga memorizzata una tabella per ciascun processo.
- Il descrittore di processo contiene, insieme col valore degli altri registri, un puntatore alla tabella delle pagine.
- All'atto del cambio di contesto, il dispatcher imposta i valori corretti della tabella delle pagine fisiche, usando la tabella delle pagine relativa al processo.
- L'architettura a supporto alla tabella delle pagine può essere realizzata in modi differenti; nel caso più semplice, si può utilizzare un insieme di **registri**.
- L'efficienza dell'architettura di supporto è determinante, in quanto ogni accesso alla memoria passa attraverso di essa.

Supporto alla tabella delle pagine

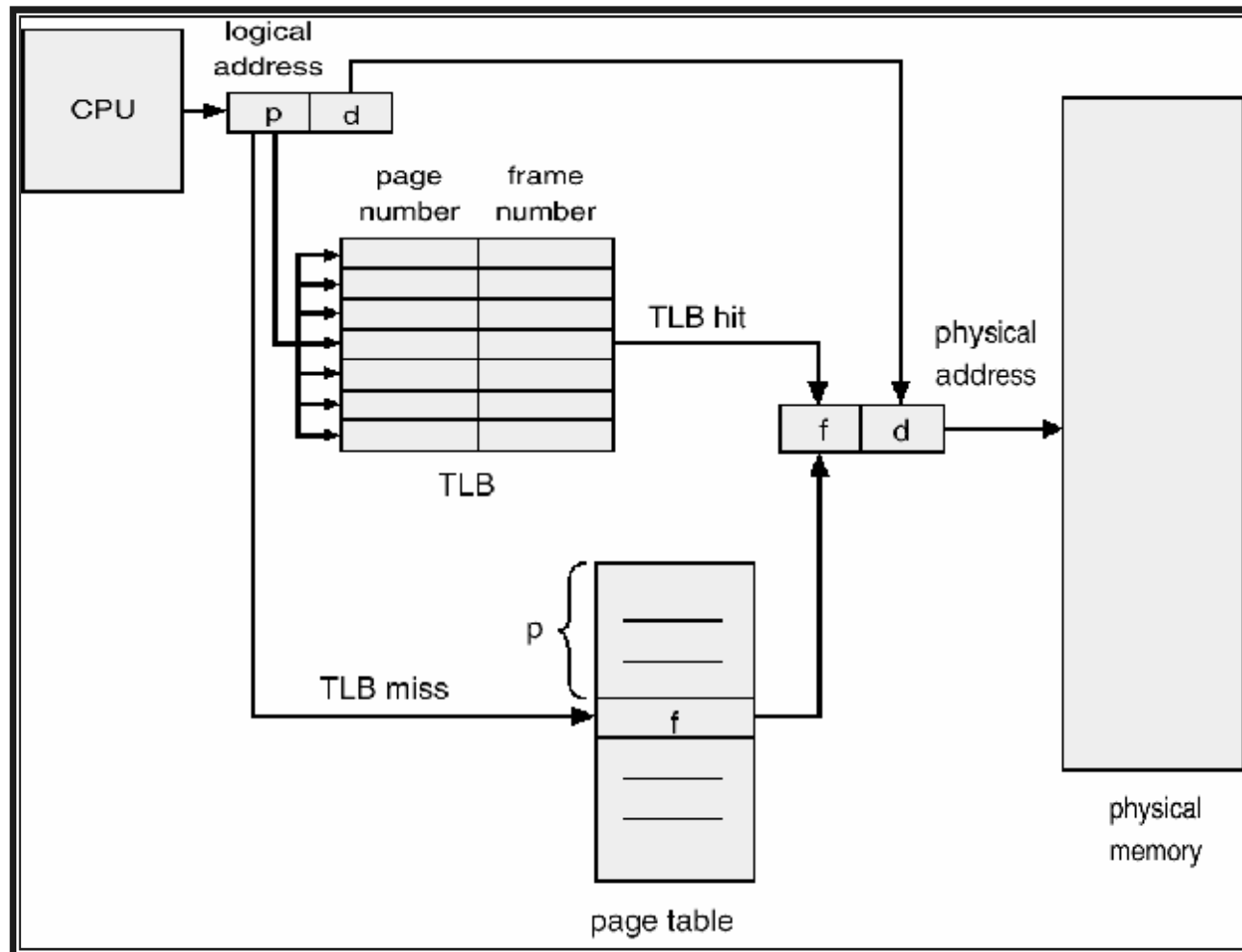
- La tabella delle pagine risiede in memoria centrale, con una tabella per ogni processo.
 - ◆ Il registro *Page–Table Base Register (PTBR)* punta all'inizio della tabella.
 - ◆ Il registro *Page–Table Length Register (PRLR)* indica la dimensione della tabella.
- Con questo schema, ogni accesso a dati o istruzioni richiede di fatto due accessi alla memoria: uno per la tabella e uno per le istruzioni/dati.

Supporto alla tabella delle pagine

- Il problema dei due accessi alla memoria può essere risolto per mezzo dei **registri associativi** (altrimenti detti *translation look-aside buffer*, **TLB**) attraverso i quali si effettua una ricerca parallela veloce su una piccola tabella.
- Traduzione dell'indirizzo (A' , A'')
 - ◆ Se A' si trova nel registro associativo, si estrae il corrispondente # di frame dal registro A'' ;
 - ◆ Altrimenti, occorre un riferimento di memoria alla tabella delle pagine.
- In caso di **insuccesso**, si possono inserire i numeri di pagina e di blocco prelevati dalla tabella nel TLB; se il TLB è già pieno, è possibile sostituire uno degli elementi presenti.

# Pagina	# Frame

Hardware di paginazione con TLB



- Alcuni TLB memorizzano gli **identificatori dello spazio di indirizzi (ASID)**, per identificare in maniera univoca ciascun processo e proteggerne lo spazio di indirizzi.

Tempo di accesso effettivo

- La percentuale di volte che un numero di pagina si trova nel TLB si chiama **tasso di successi (hit ratio)**.

- Detti:

- ◆ α il tasso di successi,
- ◆ ε unità di tempo per la ricerca nel TLB e
- ◆ γ il tempo di accesso alla memoria,

il **tempo di accesso effettivo (EAT, Effective Access Time)** è:

$$EAT = (\gamma + \varepsilon) \alpha + (2\gamma + \varepsilon)(1 - \alpha)$$

- **Esempio:** hit ratio dell'80%, ricerca nel TLB 20 μs , tempo di accesso alla memoria 100 μs , allora:

$$EAT = (100 + 20) \times 0,80 + (200 + 20) \times 0,20 = 140 \mu\text{s}$$

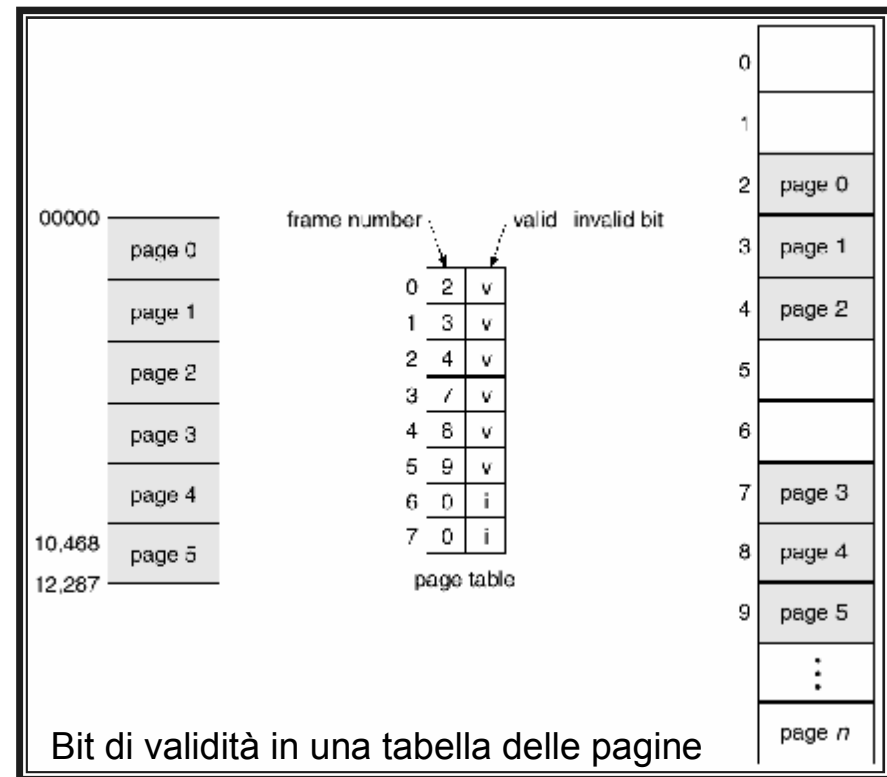
- ◆ Con un hit ratio del 98%, si ha:

$$EAT = (100 + 20) \times 0,98 + (200 + 20) \times 0,02 = 122 \mu\text{s}$$

- All'aumentare del tasso di successi, il rallentamento del tempo d'accesso alla memoria decresce.

Protezione della memoria

- La protezione della memoria è implementata associando un *bit di protezione* a ciascun frame. Vengono impediti accessi non autorizzati (si possono avere bit separati per lettura e scrittura).
- Un ulteriore *bit di validità* viene associato ad ogni elemento della tabella delle pagine:
 - ◆ Un valore “*valido*” indica che la pagina è nello spazio degli indirizzi logici del processo, e quindi è una pagina legale.
 - ◆ Un valore “*non valido*” indica che la pagina non si trova nello spazio di indirizzi logici del processo.



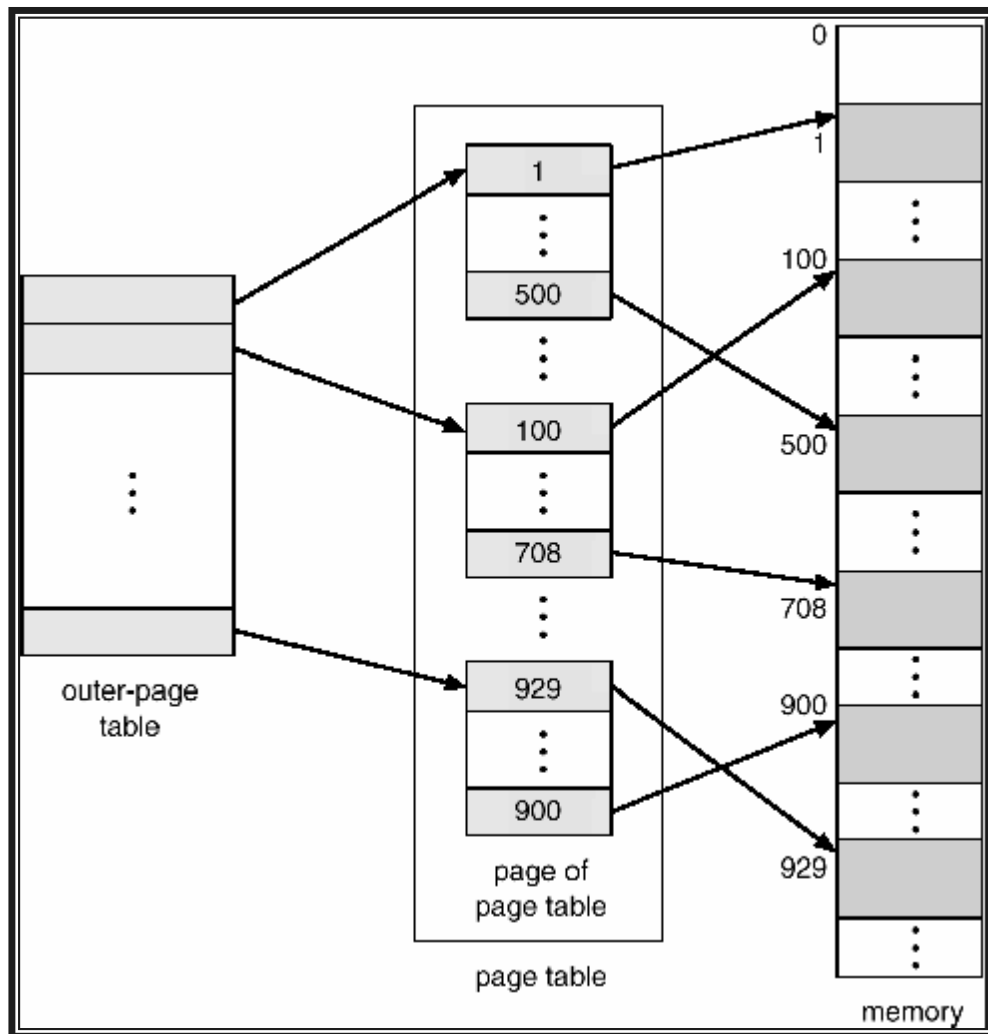
Paginazione a due livelli

- **Paginazione gerarchica:** quando lo spazio degli indirizzi logici è grande, la stessa tabella delle pagine viene paginata. Esempio semplice: **paginazione a due livelli**.
- Un indirizzo logico (in architetture a 32 bit con dimensione della pagina di 4KB) viene suddiviso in:
 - ◆ un numero di pagina di 20 bit;
 - ◆ un offset all'interno della pagina di 12 bit.
- Dato che la tabella di paginazione è paginata, il numero di pagina viene ulteriormente suddiviso in:
 - ◆ un numero di pagina di 10 bit;
 - ◆ un offset di 10 bit.
- Un indirizzo logico è costituito da:

<i>Numero di pagina</i>		<i>offset</i>
p_1	p_2	d
10	10	12

p_1 è un indice nella tabella esterna, e p_2 è lo spostamento all'interno della pagina indicata dalla tabella esterna.

Schema di tabella delle pagine a due livelli



- Poiché ciascun livello viene memorizzato come una tabella separata in memoria, la traduzione di un indirizzo logico in uno fisico può richiedere tre accessi alla memoria.
- Anche se il tempo richiesto per un accesso alla memoria è triplicato, la presenza di cache consente di mantenere prestazioni ragionevoli.

Schema di traduzione degli indirizzi

- Traduzione degli indirizzi per un'architettura a 32 bit con paginazione a due livelli.
- Poiché la traduzione degli indirizzi avviene dalla tabella esterna delle pagine verso l'interno, questo metodo è noto come **tabella delle pagine ad associazione diretta** (Pentium II)

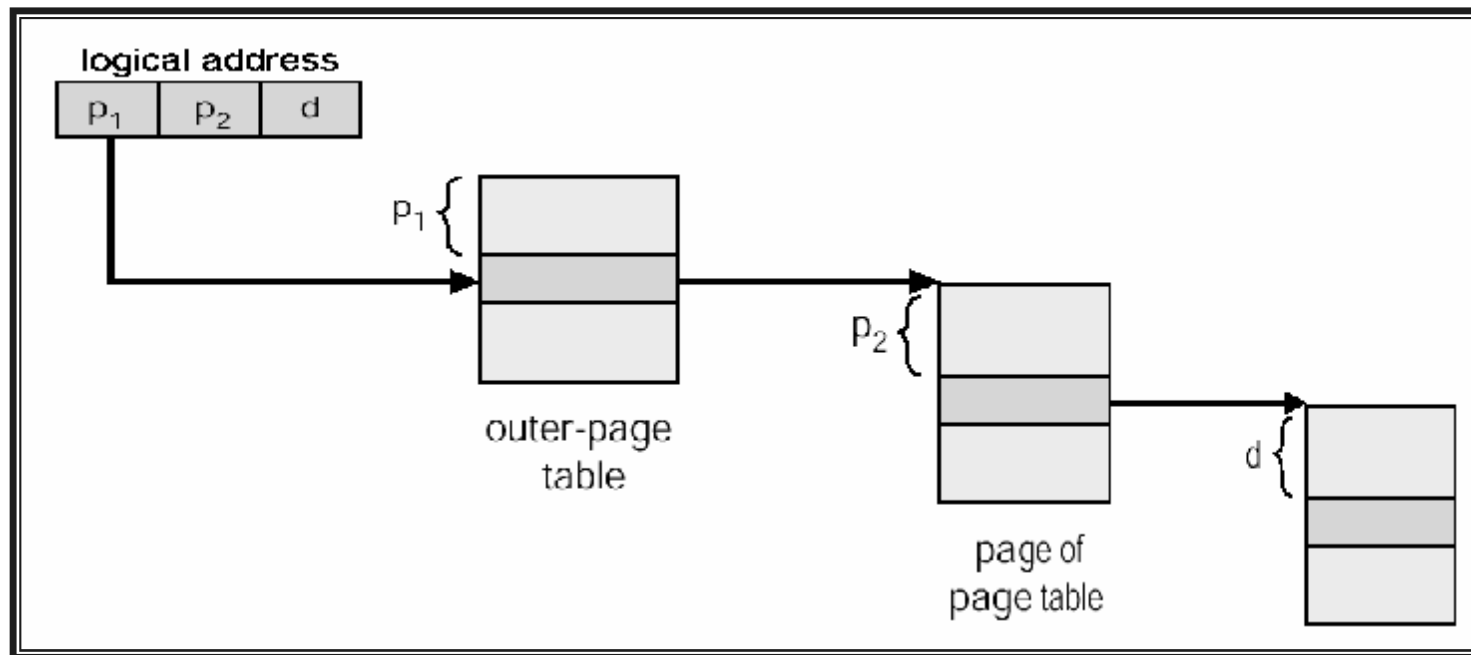


Tabella delle pagine di tipo hash

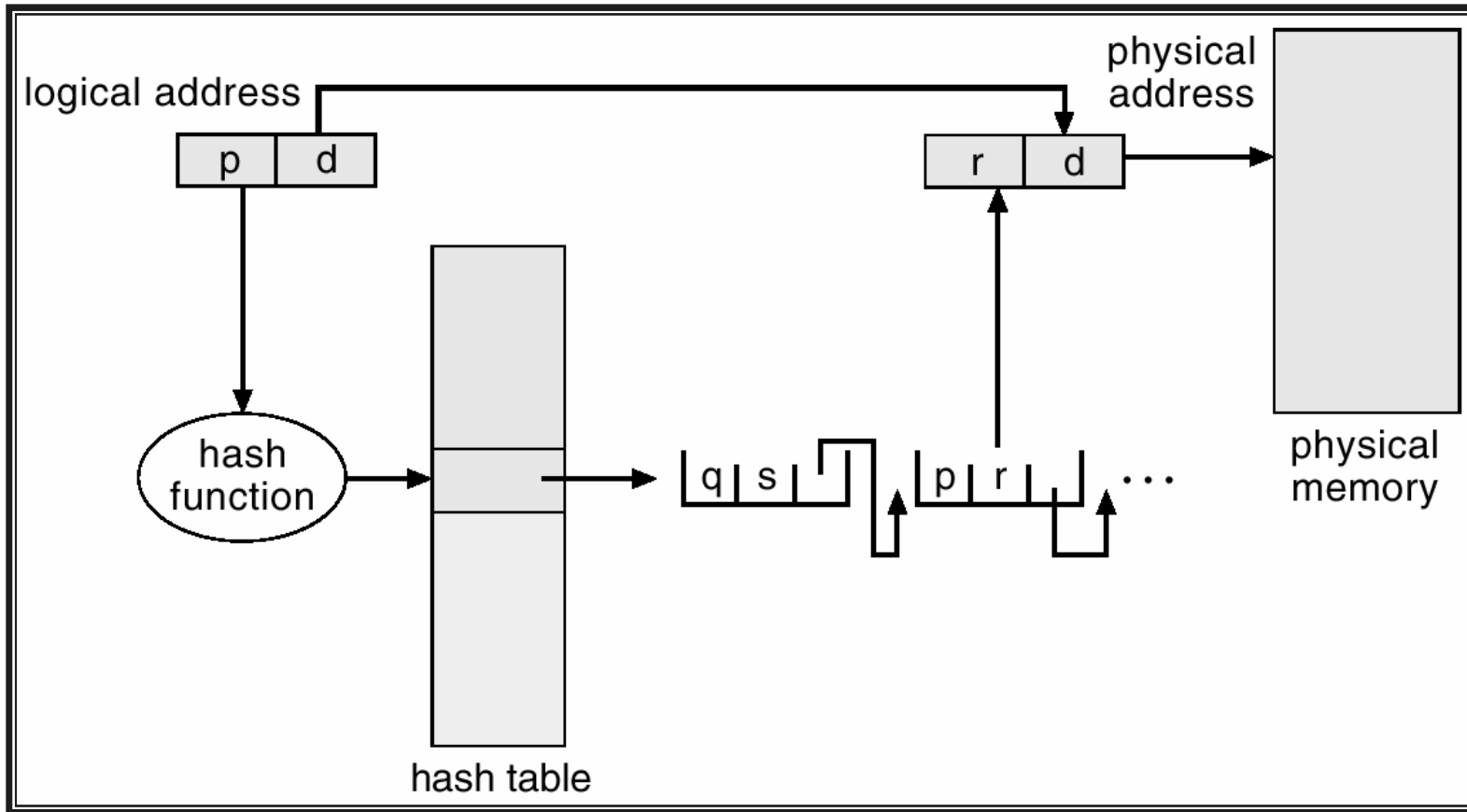
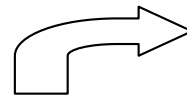
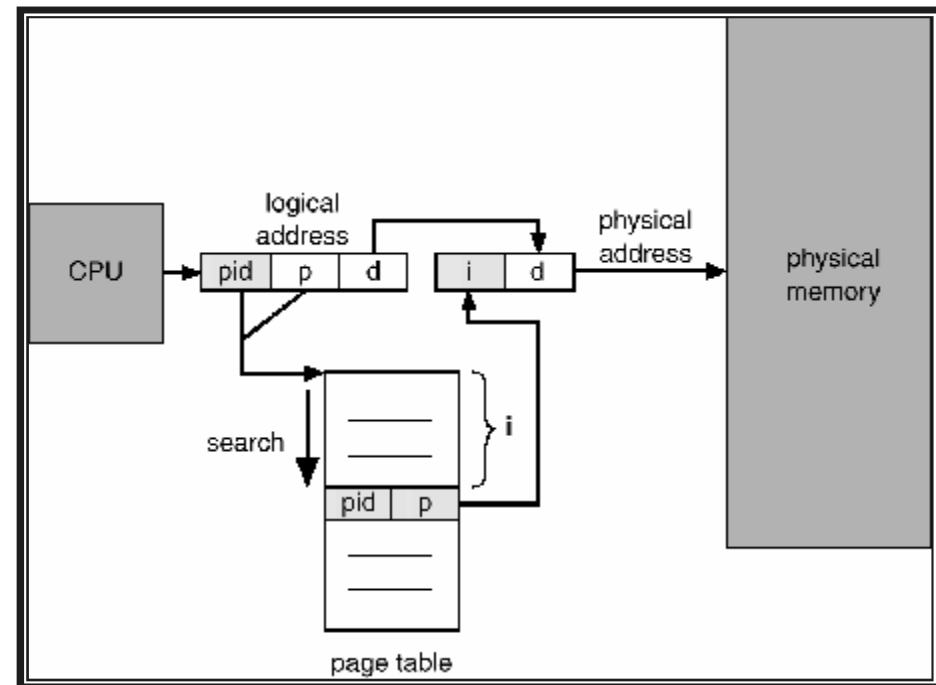


Tabella delle pagine invertita

- Un elemento della tabella per ogni frame.
- Gli elementi della tabella contengono l'indirizzo virtuale della pagina memorizzata nel dato frame, con informazioni sul processo che possiede tale pagina.
- Riduzione della memoria richiesta per memorizzare ciascuna tabella delle pagine; incremento del tempo necessario per ricercare nella tabella quando si ha un riferimento a pagina.
- Notare che è necessario ricercare su tutta la tabella! Si impiegano tabelle hash per limitare la ricerca ad uno — o al più pochi — elementi della tabella.



Ciascun indirizzo virtuale è formato dalla tripla:
<# processo, #pagina, offset>



Pagine condivise

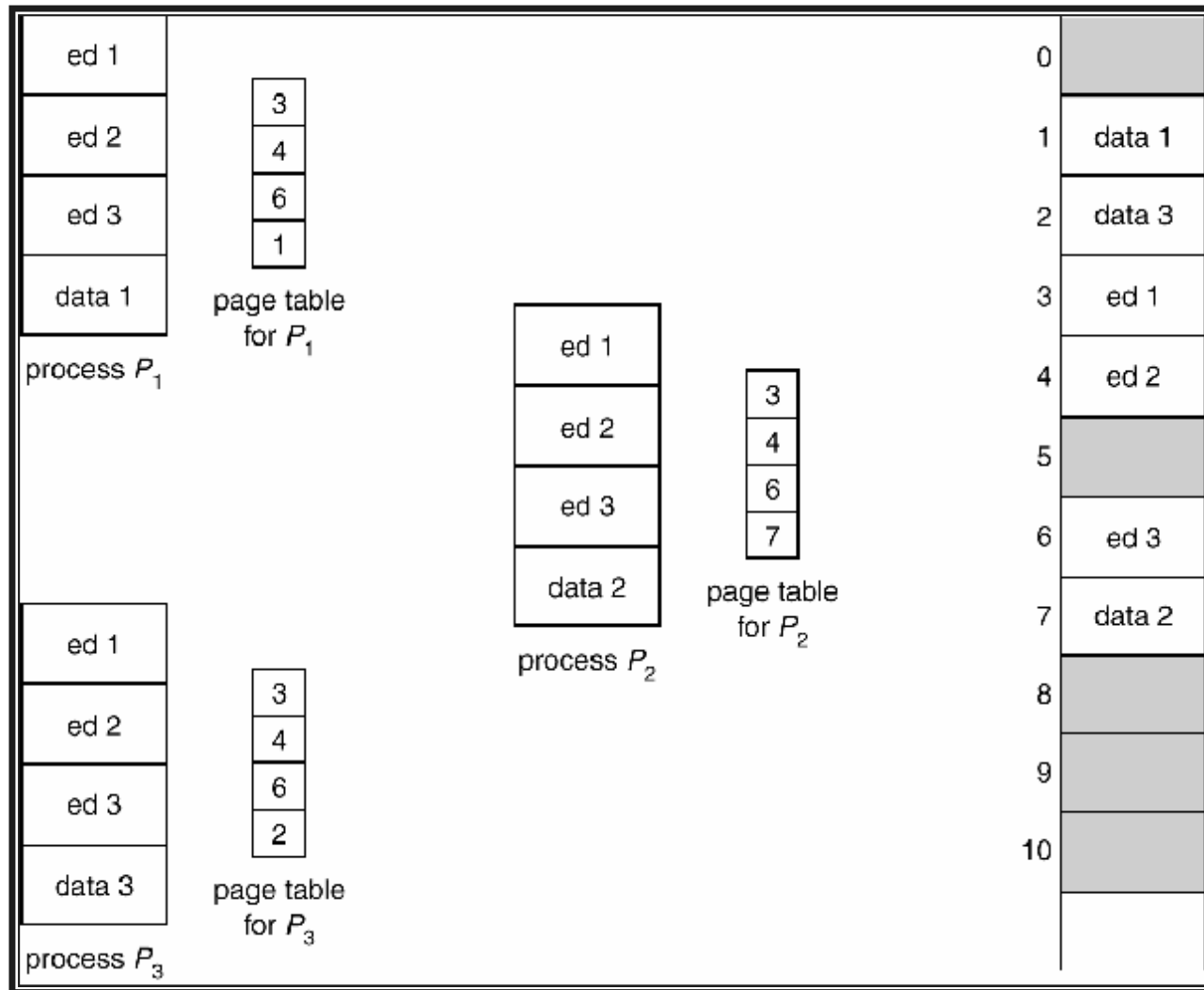
■ Codice condiviso

- ◆ Una copia di codice a sola lettura (*reentrant*) viene condivisa fra processi (ad esempio: text editor, compilatori, sistemi a finestre).
- ◆ Il codice condiviso deve apparire nella stessa locazione nello spazio degli indirizzi logici di tutti i processi.

■ Codice e dati privati

- ◆ Ciascun processo mantiene una copia separata dei dati e del codice.
- ◆ Le pagine di codice e dati privati possono apparire ovunque nello spazio degli indirizzi logici.

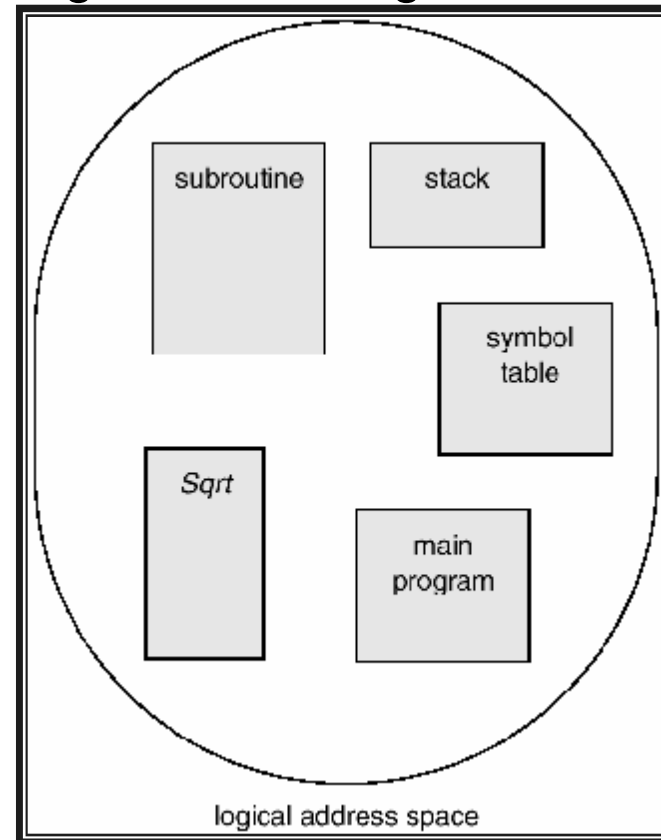
Esempio di pagine condivise



Condivisione di codice in un ambiente di paginazione:
Editor contenuto in tre pagine

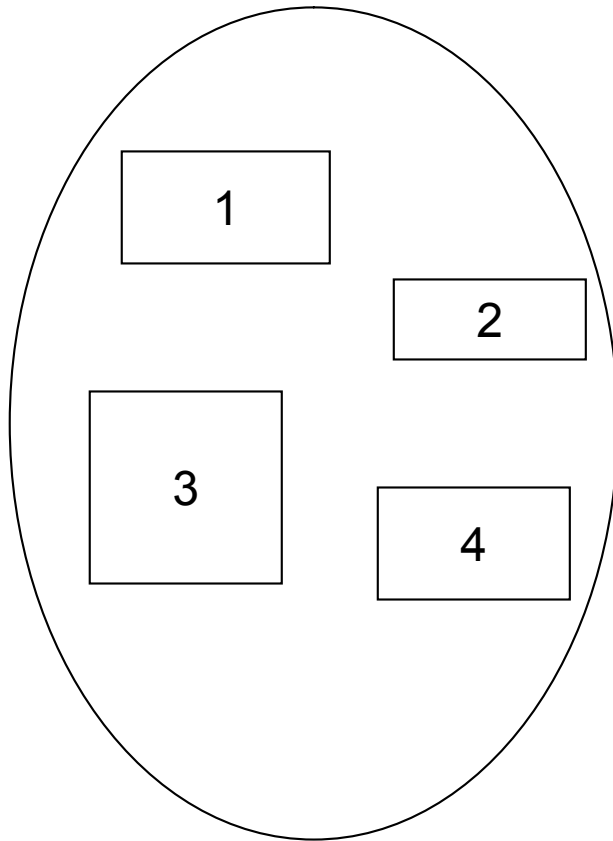
Segmentazione

- Schema di gestione della memoria che asseconda la visione utente (l'utente generalmente non pensa alla memoria come ad un array lineare di byte).
- Un programma è una collezione di segmenti. Un segmento è un'unità logica, come ad esempio:
 - ◆ programma principale (main),
 - ◆ procedura,
 - ◆ funzione,
 - ◆ metodo,
 - ◆ oggetto,
 - ◆ variabili logiche, variabili globali,
 - ◆ blocco common,
 - ◆ stack,
 - ◆ tabella dei simboli, matrici.

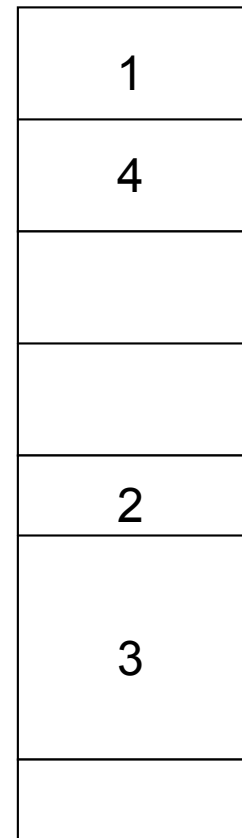


...il mio codice...

Schema logico di segmentazione



Spazio utente



Spazio fisico di memoria

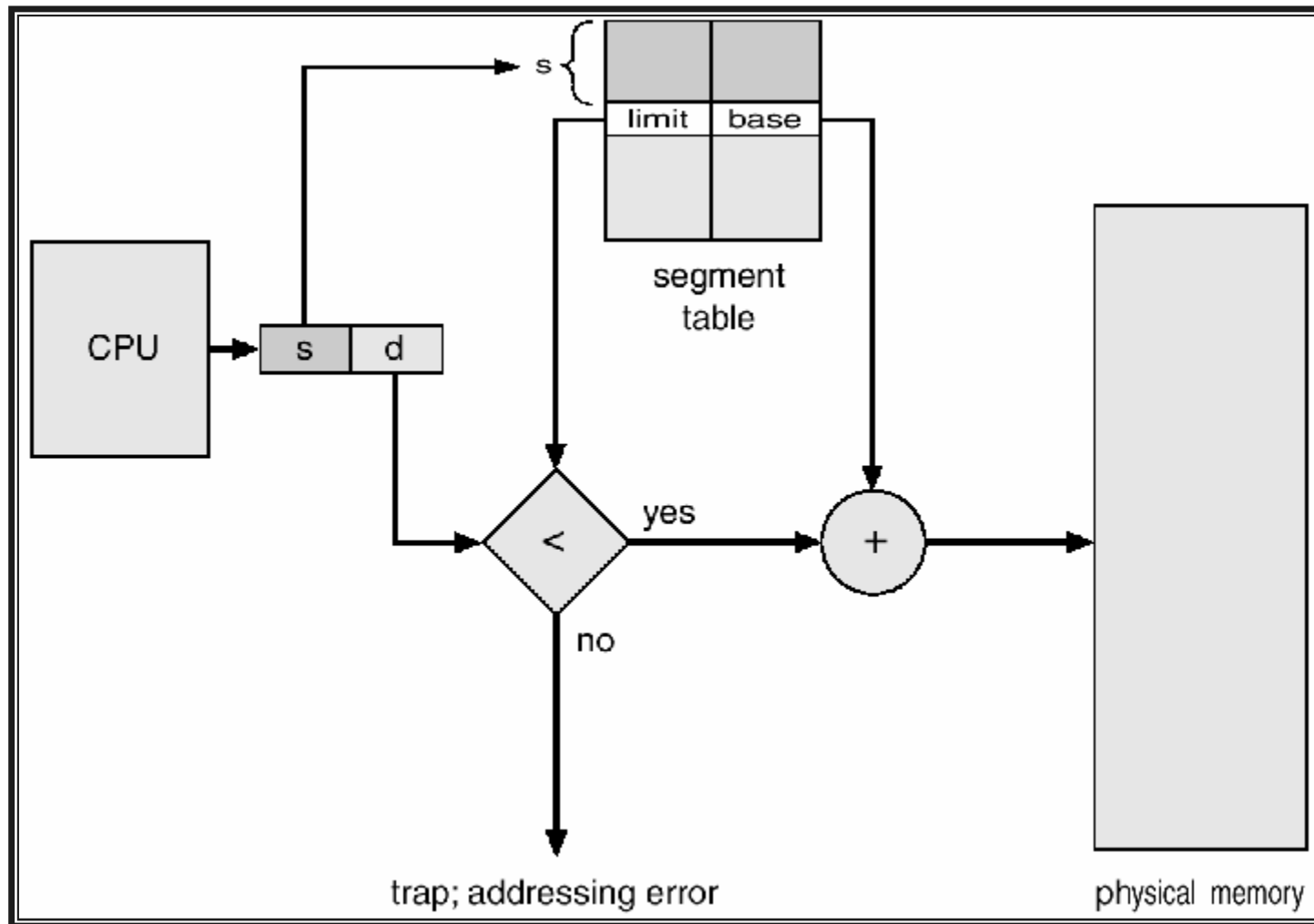
Architettura per la segmentazione

- Gli indirizzi logici sono rappresentati da coppie:
 <#segmento, offset>
- **Tabella dei segmenti** — mappa gli indirizzi fisici bidimensionali; ciascun elemento della tabella contiene:
 - ◆ *base* — indirizzo fisico iniziale della memoria contenente il segmento.
 - ◆ *limite* — specifica la lunghezza del segmento.
- Il registro **Segment-Table Base Register (STBR)** punta alla locazione in memoria della tabella dei segmenti.
- Il registro **Segment-Table Length Register (STLR)** indica il numero di segmenti utilizzati dal programma; un numero di segmento s è legale se $s < \text{STLR}$.

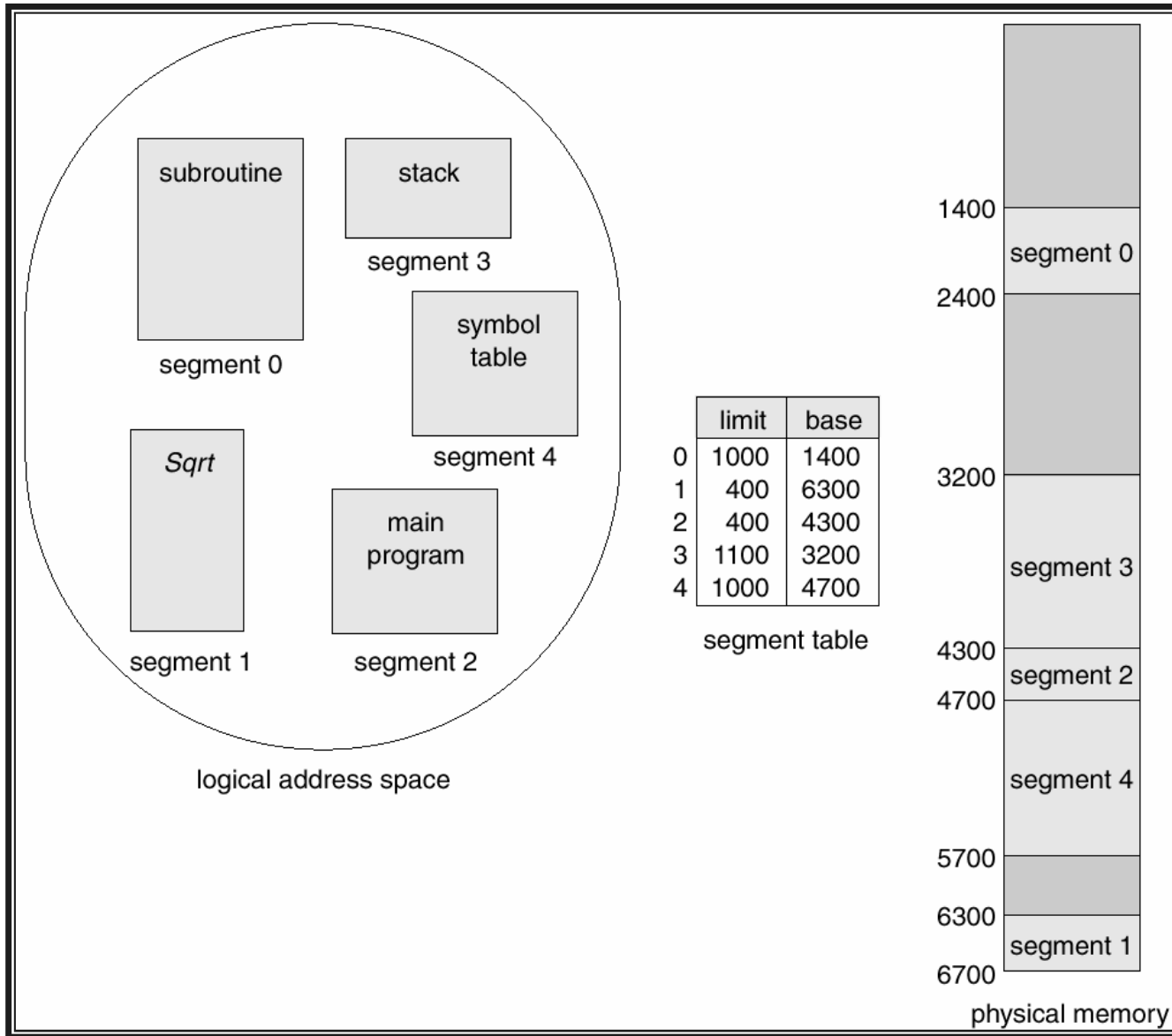
Architettura per la segmentazione

- **Rilocazione:** Dinamica, con tabella dei segmenti.
- **Condivisione:** Si ottengono segmenti condivisi, puntando allo stesso numero di segmento.
- **Allocazione:** First fit/Best fit. Possibile frammentazione esterna.
- **Protezione:** Si associa a ciascun elemento della tabella dei segmenti:
 - ◆ bit di validità = 0 \Rightarrow segmento illegale
 - ◆ privilegi read/write/execute
- Bit di protezione associati ai segmenti; la condivisione di codice viene implementata a livello di segmento.
- Dato che i segmenti variano di dimensione, l'allocazione della memoria è dinamica.

Hardware di segmentazione

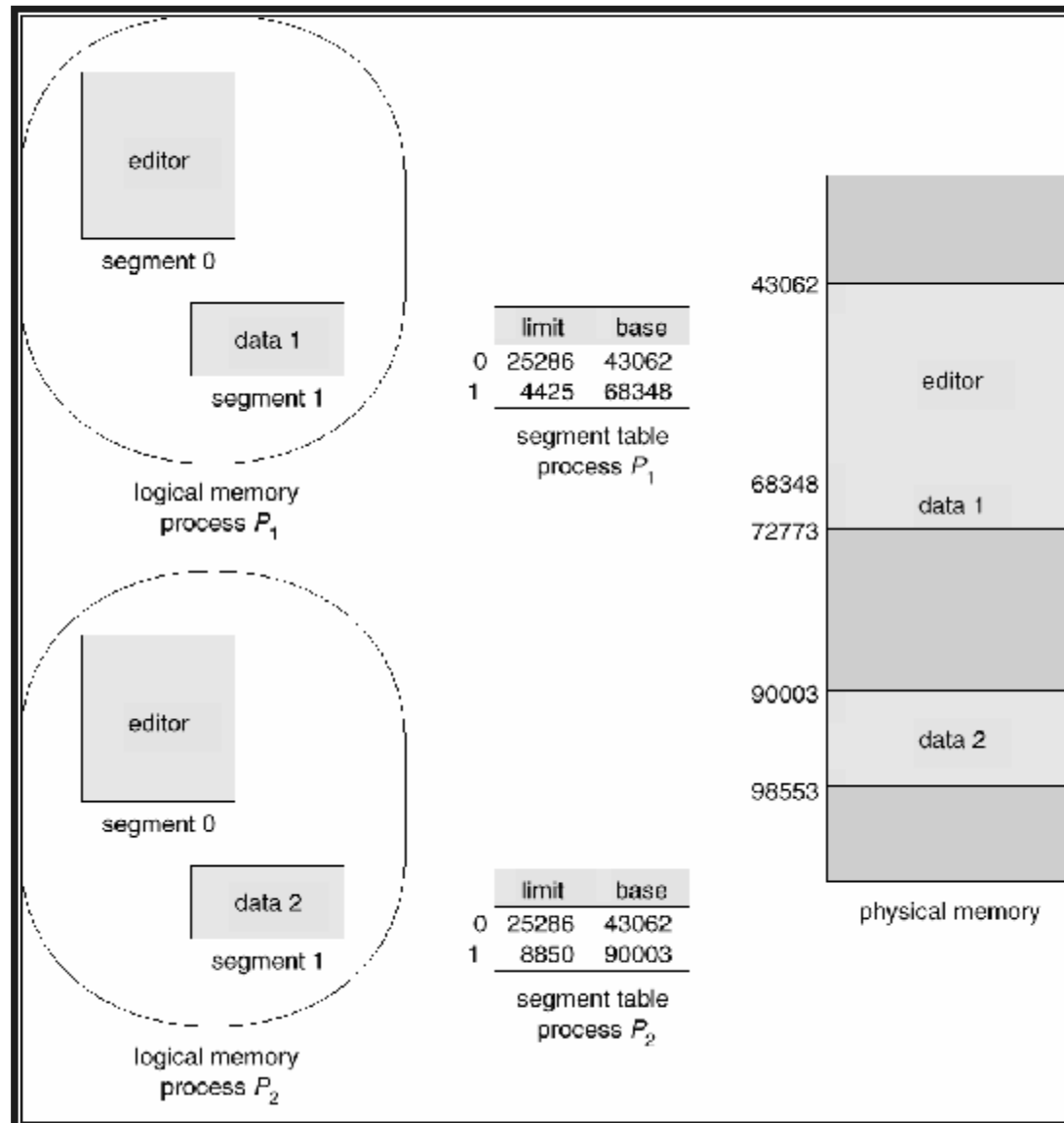


Segmentazione



Esempio di segmentazione

Segmentazione

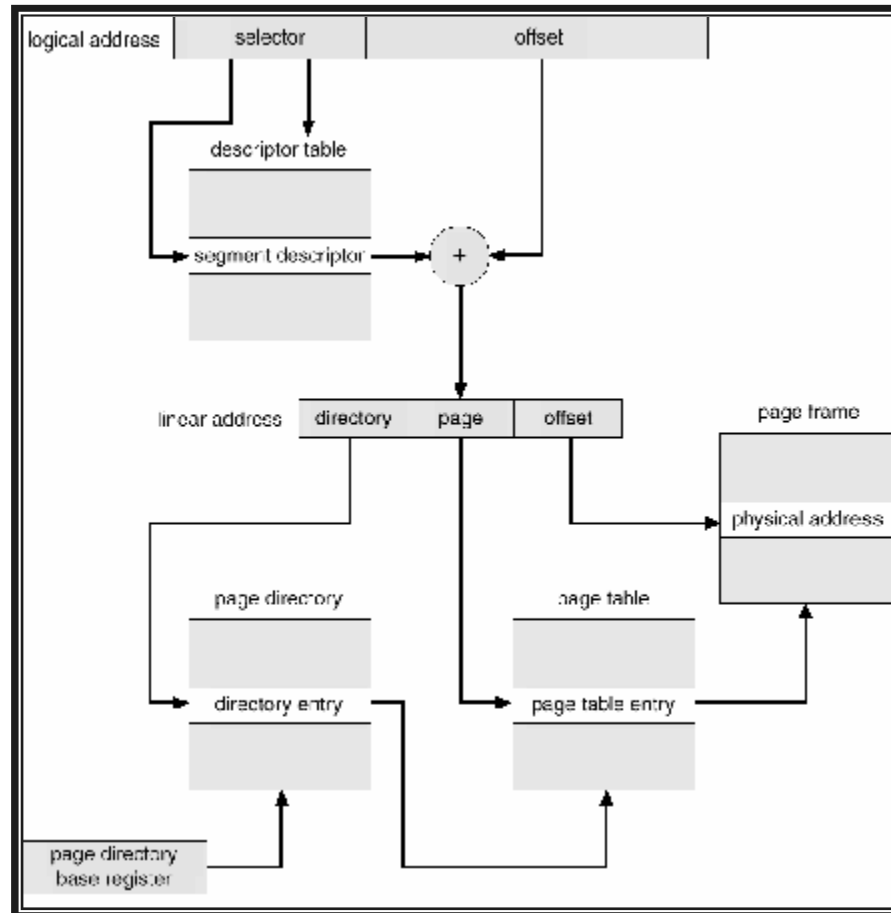
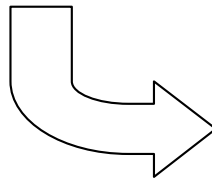


Condivisione di segmenti in un sistema con memoria segmentata

Segmentazione con paginazione — Intel 386

- L'Intel 386 impiega una segmentazione con paginazione per la gestione della memoria con uno schema di paginazione a due livelli.

Schema di traduzione degli indirizzi.



Sommario

- Gli algoritmi di gestione della memoria per sistemi operativi multiprogrammati variano da un metodo semplice, per sistema con un singolo utente, fino alla segmentazione paginata.
- L'architettura determina la scelta del metodo da utilizzare.
- Gli algoritmi di gestione analizzati (assegnazione contigua, paginazione, segmentazione con paginazione) differiscono per aspetti riguardanti:
 - ◆ l'architettura che usano,
 - ◆ le prestazioni che producono,
 - ◆ la frammentazione che introducono,
 - ◆ la rilocazione che impiegano,
 - ◆ l'eventuale utilizzo dello swapping,
 - ◆ la possibilità di condivisione del codice e dei dati,
 - ◆ la protezione che offrono.