GESTIONE DELLA

MEMORIA CENTRALE

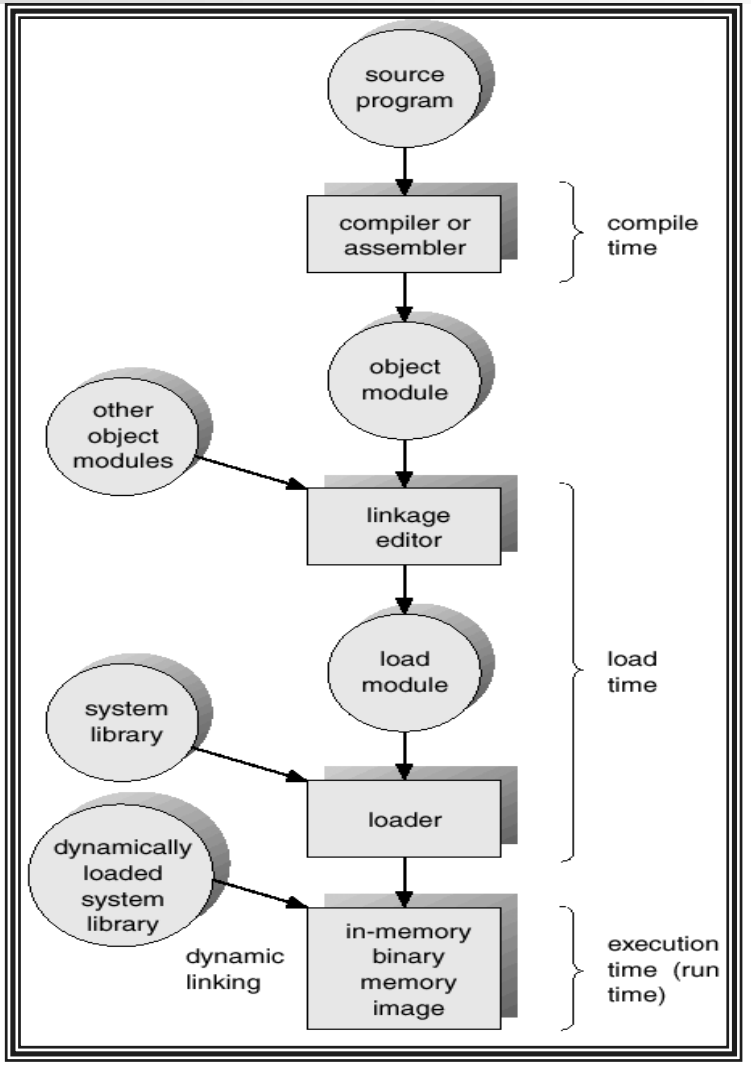
Gestione della Memoria

- Background
- Spazio di indirizzi
- Swapping
- Allocazione Contigua
- Paginazione

Background

- Per essere eseguito un programma deve trovarsi (almeno parzialmente) in memoria centrale.
- La gestione della memoria centrale dipende dalle funzionalità dell'hardware disponibile.
- I programmi che risiedono sul disco devono essere trasferiti in memoria centrale tramite la ***coda di input***.
- ***Coda di input*** : l'insieme dei processi residenti su disco che attendono di essere trasferiti e eseguiti.
- I programmi utente possono attraversare diversi stadi prima di venire eseguiti.

Stadi di un programma utente



Associazione di istruzioni e dati alla memoria

L'associazione (***Address binding***) di istruzioni e dati alla memoria può avvenire in momenti diversi :

- **Compilazione**: se la locazione di memoria è conosciuta a priori possono essere generati indirizzi assoluti. La ricompilazione è necessaria quando la locazione di partenza cambia.

- **Caricamento**: se la locazione di memoria non è conosciuta a priori si genera codice rilocabile (al variare dell'indirizzo iniziale).

- **Esecuzione**: se il processo può essere spostato, l'associazione viene ritardata al momento dell'esecuzione. Necessario hardware specializzato (es: *registri base* e *limite*).

Indirizzi fisici e indirizzi logici

- Il concetto di *spazio di indirizzi logici* che è legato allo *spazio degli indirizzi fisici* è molto importante nella gestione della memoria centrale.
  - *Indirizzo logico* : generato dalla CPU (*indirizzo virtuale)*.
  - *Indirizzo fisico* : visto dall'unità di memoria.

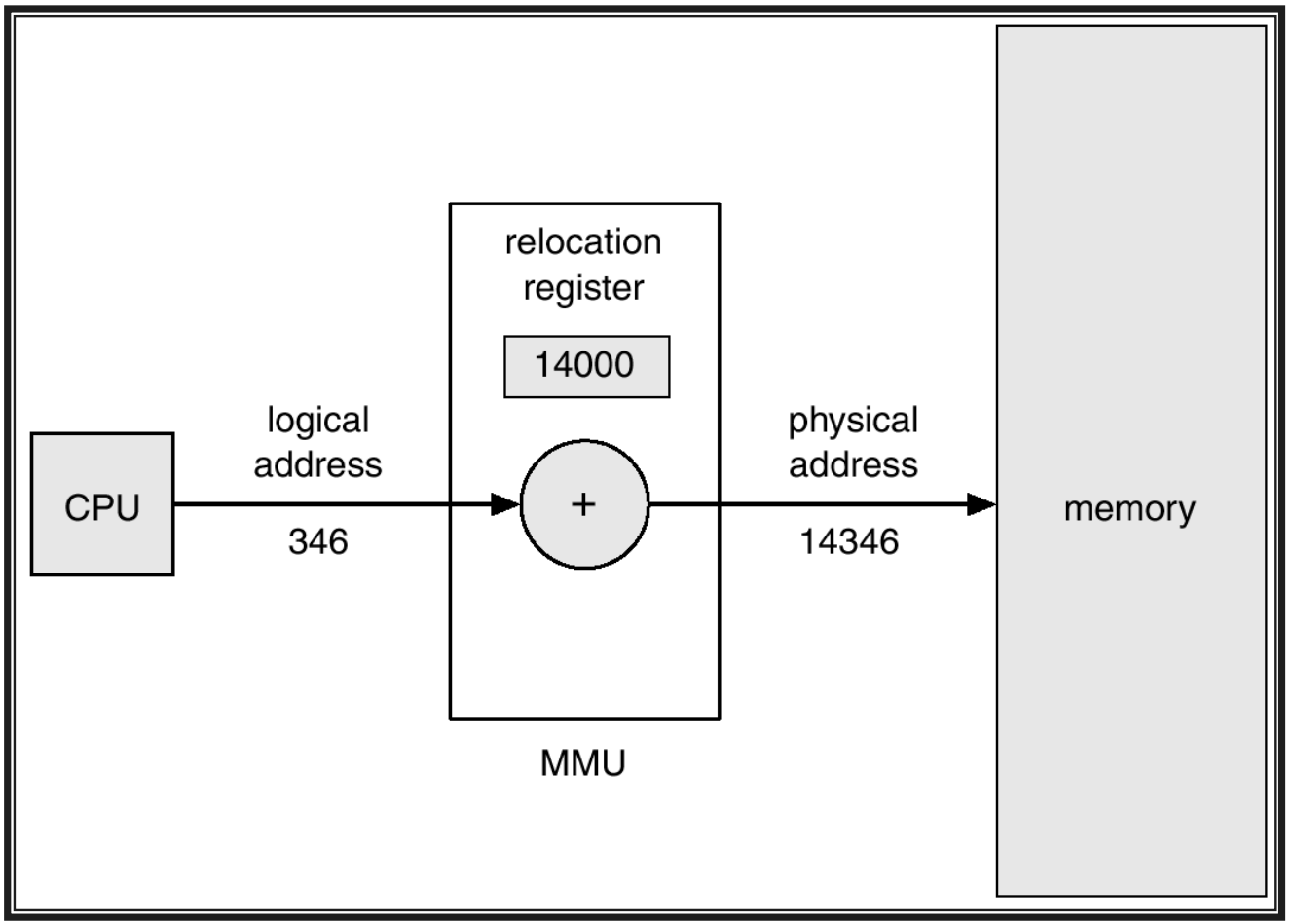
- Gli indirizzi logici e gli indirizzi fisici sono uguali nella compilazione e nel caricamento.

- Durante l'esecuzione gli indirizzi logici sono detti virtuali e differiscono dagli indirizzi fisici.

Memory-Management Unit (MMU)

- C'è bisogno di un mapping a tempo di esecuzione.
- **MMU:** *dispositivo hardware che associa indirizzi virtuali a indirizzi fisici.*
- Nello schema della MMU, il valore del registro di rilocazione viene aggiunto ad ogni indirizzo generato dal processo utente quando viene portato in memoria.
- Il programma utente lavora con indirizzi logici e non conosce mai gli indirizzi fisici dove sono realmente allocati i propri dati e il proprio codice.

Rilocazione dinamica con registro di rilocazione



Caricamento dinamico

- Le routine (procedure, metodi) vengono caricate quando sono chiamate (se non sono già in memoria).
- Loader di caricamento rilocabile.
- Migliore uso dello spazio di memoria; le routine mai usate non vengono mai caricate in memoria centrale.
- Utile quando molta parte del codice è usata raramente.
- Nessun speciale supporto del S.O. In alcuni casi esistono librerie per il caricamento dinamico.

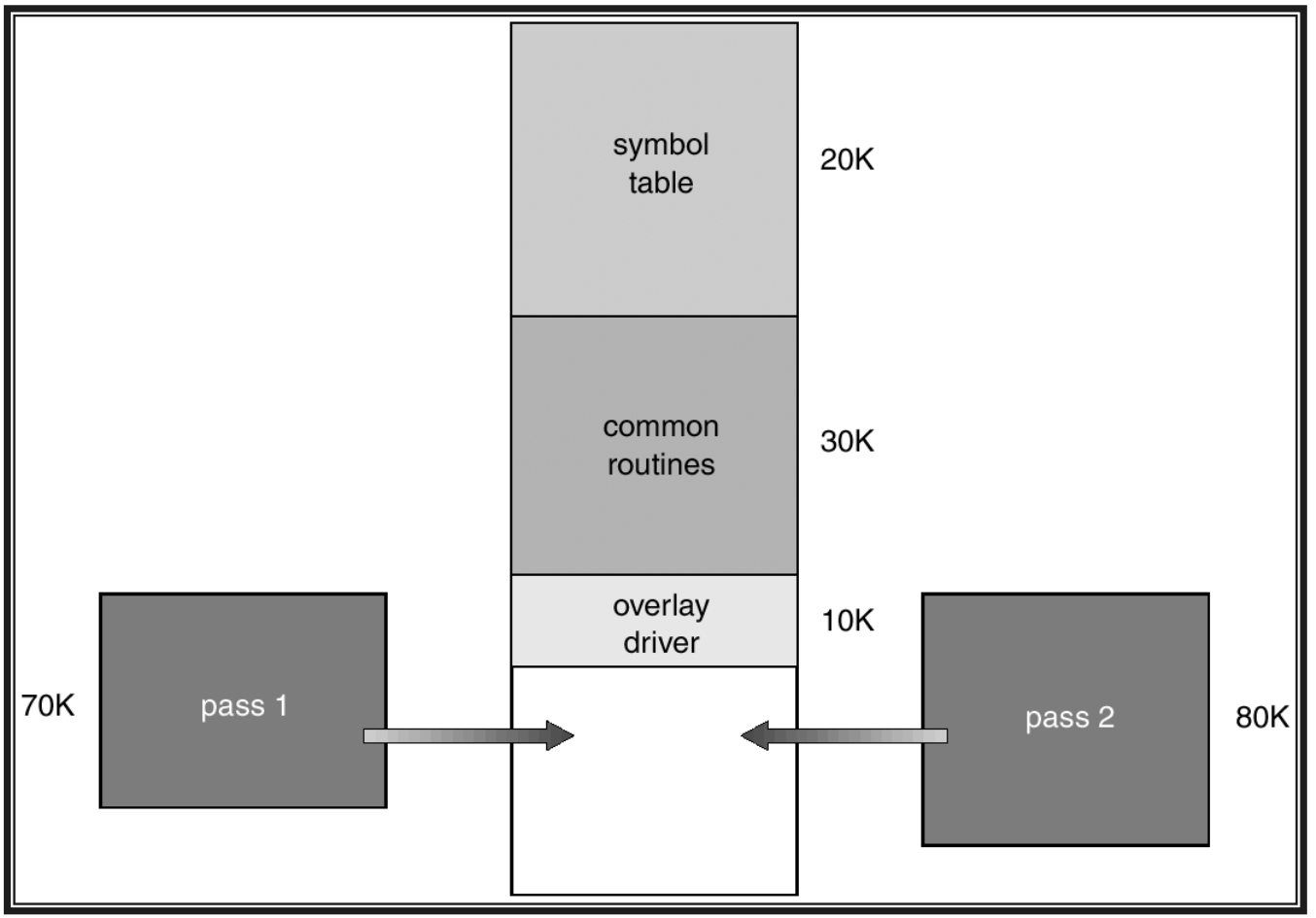
Collegamento dinamico

- Collegamento ritardato fino al momento dell'esecuzione.
- Codice ***stub***, usato per localizzare (in memoria o sul disco) la routine di sistema richiesta.
- Lo ***stub*** si sostituisce con l'indirizzo della routine che verrà eseguita.
- Il sistema operativo permette l'utilizzo delle routine a più processi.
- Utile per le librerie di sistema.

Overlay

- La tecnica dell'overlay è usata per eseguire processi che non "entrano" in memoria.
- Tiene in memoria solo i dati e le istruzioni usati spesso.
- Quando servono altri dati/istruzioni si caricano al posto di quelli meno usati.
- Implementato dagli utenti su sistemi con hardware che non permette di realizzare tecniche migliori.
- Complesso da realizzare.

Overlay per un assembler a due passi



Swapping

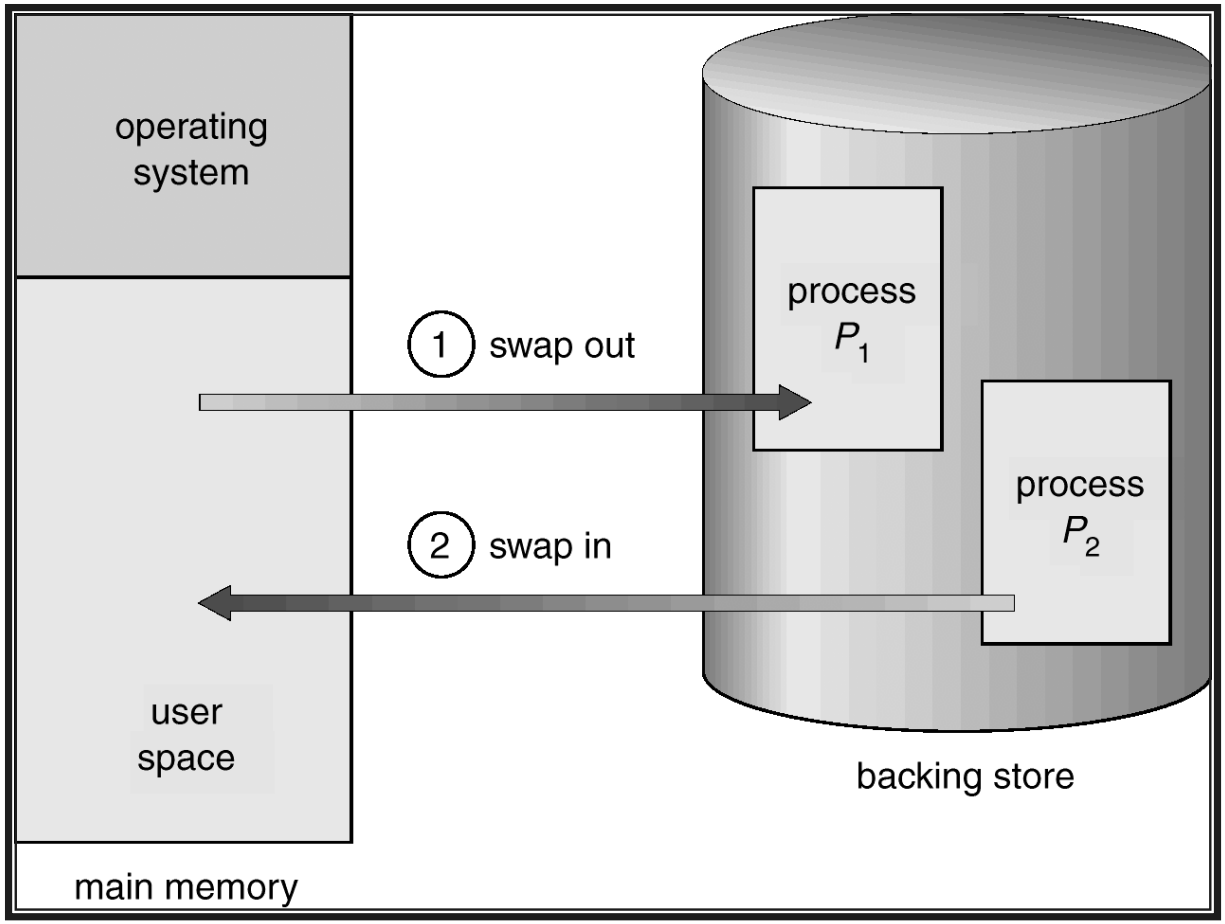
- Un processo può essere temporaneamente riportato (*swapped)* su disco (*backing store*) e quindi riportato in memoria al momento di riprendere l'esecuzione.

- *Roll out, roll in* : indicano le operazioni di swapping usate per algoritmi di scheduling basati su priorità quando un processo a più bassa priorità viene rimosso dalla memoria per far posto al processo con alta priorità.

- La maggior parte del tempo di swap è tempo di trasferimento e il tempo totale è proporzionale alla dimensione dell'area di memoria sottoposta a swap.

- Versioni modificate di tecniche di swapping sono disponibili su molti sistemi operativi: UNIX, Linux, and Windows.

Swapping tra due processi



ms = 1MB

V = 4MBs

t = 1 MB/ 4 MBs

= 1/4 sec

= 250 msec

T = 2t = 500 msec

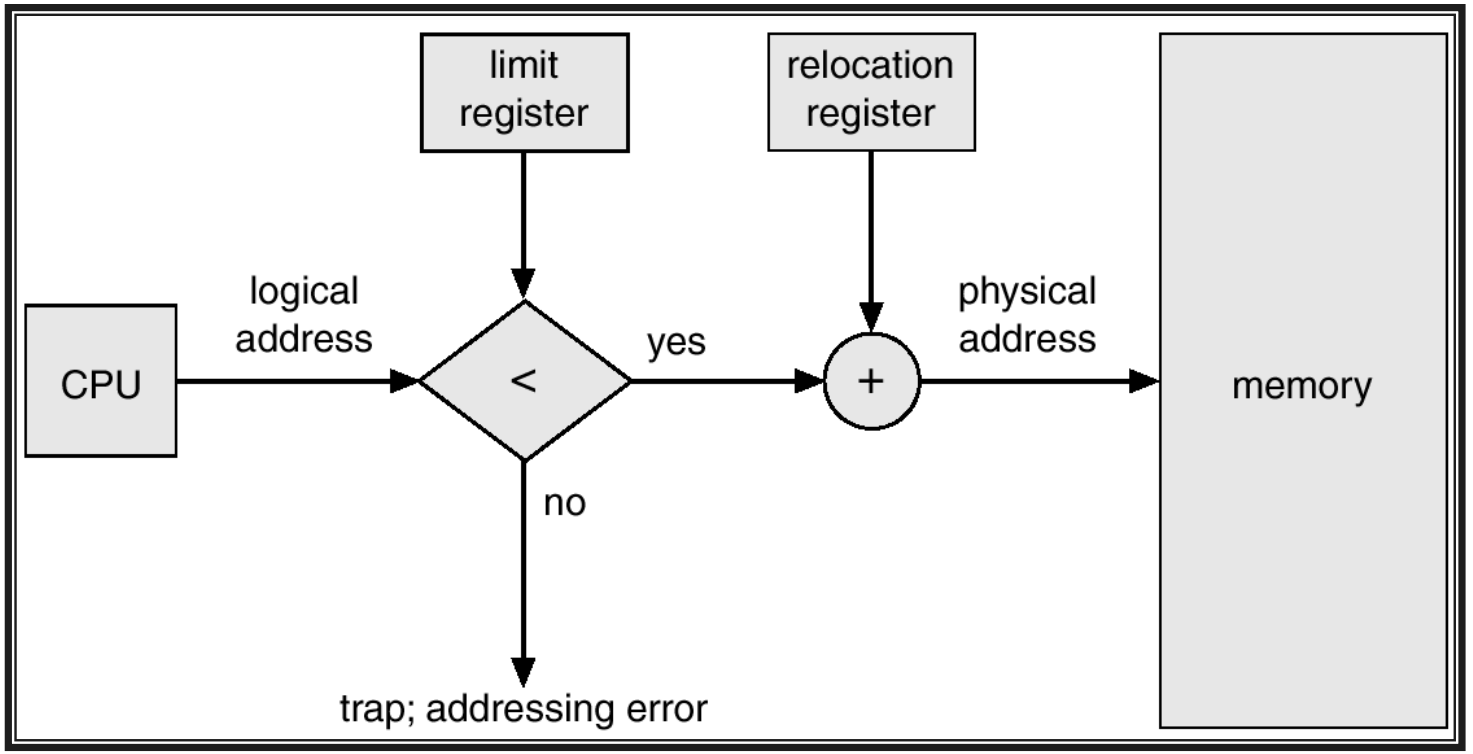
Allocazione Contigua

- La memoria centrale è usualmente divisa in due partizioni:
  - Partizione del sistema operativo.
  - Partizione per i processi utente.

- Allocazione con partizione singola
  - Registro di rilocazione è usato per proteggere i processi utente tra loro e il sistema operativo dai processi utente.

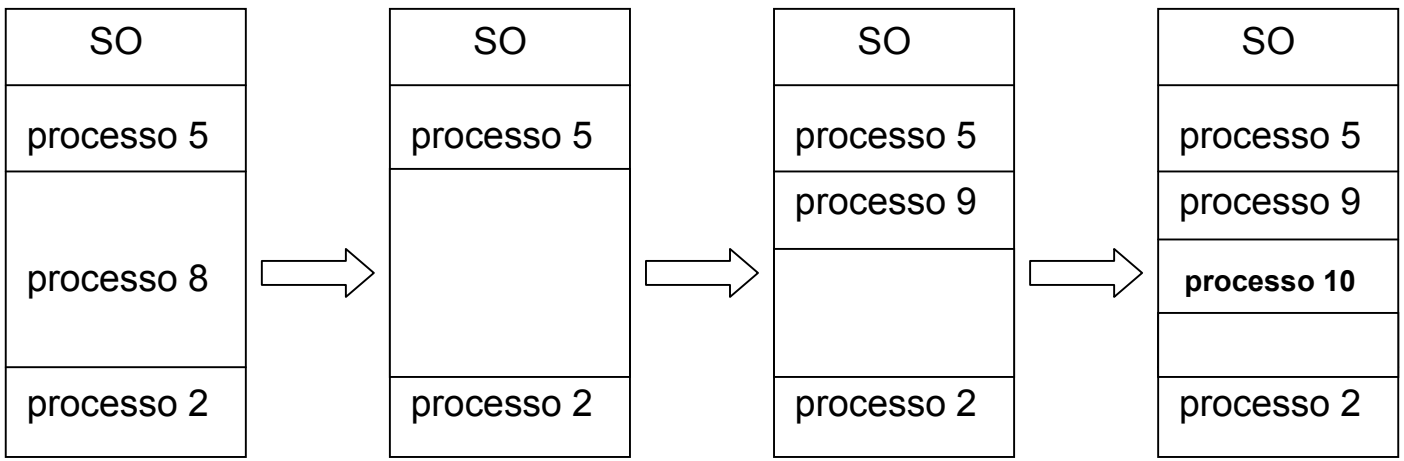
  - Registro di rilocazione contiene l'indirizzo fisico più piccolo e il registro limite contiene l'intervallo degli indirizzi logici: *ogni indirizzo logico deve essere minore del registro limite.*

Supporto hardware per rilocazione e registro limite



Allocazione Contigua

- Allocazione con partizioni multiple
  - *Buco* : blocco di memoria disponibile; buchi di dimensione diverse sono distribuiti nella memoria.
  - Quando un processo arriva viene allocato una partizione di memoria disponibile (*buco*) abbastanza grande per contenerlo.
  - Il sistema operativo mantiene informazioni su: a) partizioni allocate e b) partizioni libere (buchi)



Allocazione dinamica

Come soddisfare una richiesta di dimensione n data una lista di buchi liberi ?

- **First-fit**: Alloca il ***primo*** buco libero sufficiente.
- **Best-fit**: Alloca il ***più piccolo*** buco libero sufficiente; ricerca sull'intera lista e produce i più piccoli buchi inutilizzati.
- **Worst-fit**: Alloca il ***più grande*** buco; ricerca sull'intera lista e produce i più grandi buchi inutilizzati (*ma più utili*).

Simulazione: First-fit e Best-fit sono migliori del Worst-fit in termini di velocità e uso di memoria.

Problema della Frammentazione

- **Frammentazione Esterna** – esiste uno spazio totale di memoria disponibile per soddisfare una richiesta ma non è contiguo.

- **Frammentazione Interna** – la memoria allocata può essere un po' più grande di quella richiesta; la parte in eccesso è interna alla partizione ma non è usata.

- La ***compattazione*** riduce la frammentazione esterna:
  - La memoria libera viene compattata in un unico blocco spostando i blocchi usati.
  - La compattazione è possibile solo se la rilocazione è dinamica a tempo di esecuzione.
  - Metodi semplici richiedono tempi più lunghi.

Allocazione non contigua: Paginazione

- Lo spazio degli indirizzi logici di un processo possono essere non contigui; la memoria da allocare è presa da dove essa è disponibile.

PAGINAZIONE

- *La memoria fisica è divisa in blocchi di dimensione fissa chiamati* **frame** (la dimensione è una potenza di 2, tra 512 e 8192 byte).

- *La memoria logica è divisa in blocchi di dimensione fissa chiamati* **pagine**.

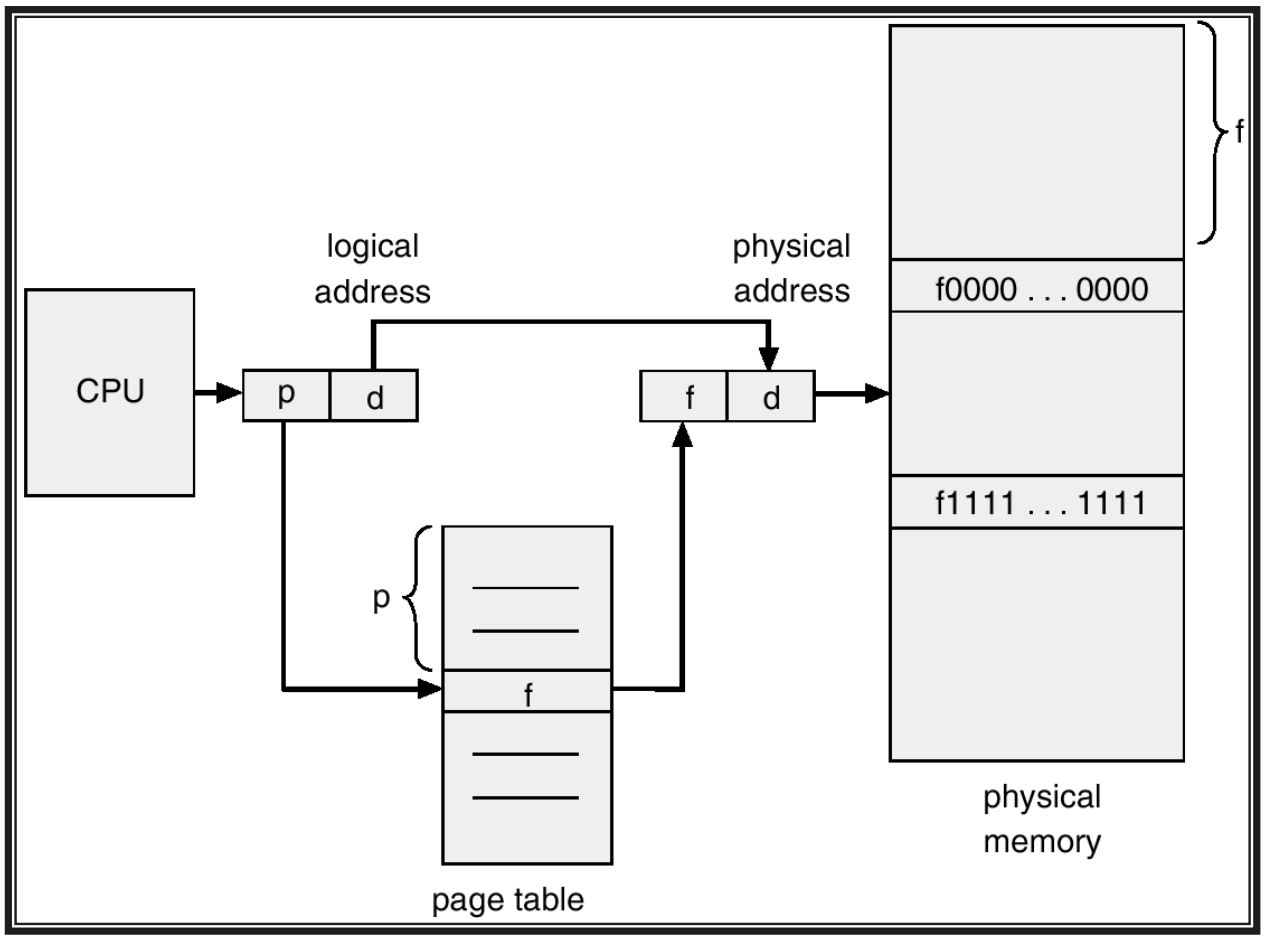
Allocazione non contigua: Paginazione

- Si tiene traccia di tutti frame liberi.
- Per eseguire un programma che richiede *n* pagine, bisogna trovare *n* frame liberi.
- Esiste una tabella delle pagine che contiene l'indirizzo iniziale di ogni pagina nella memoria fisica.
- Si evita la frammentazione esterna.
- Si può avere frammentazione interna.

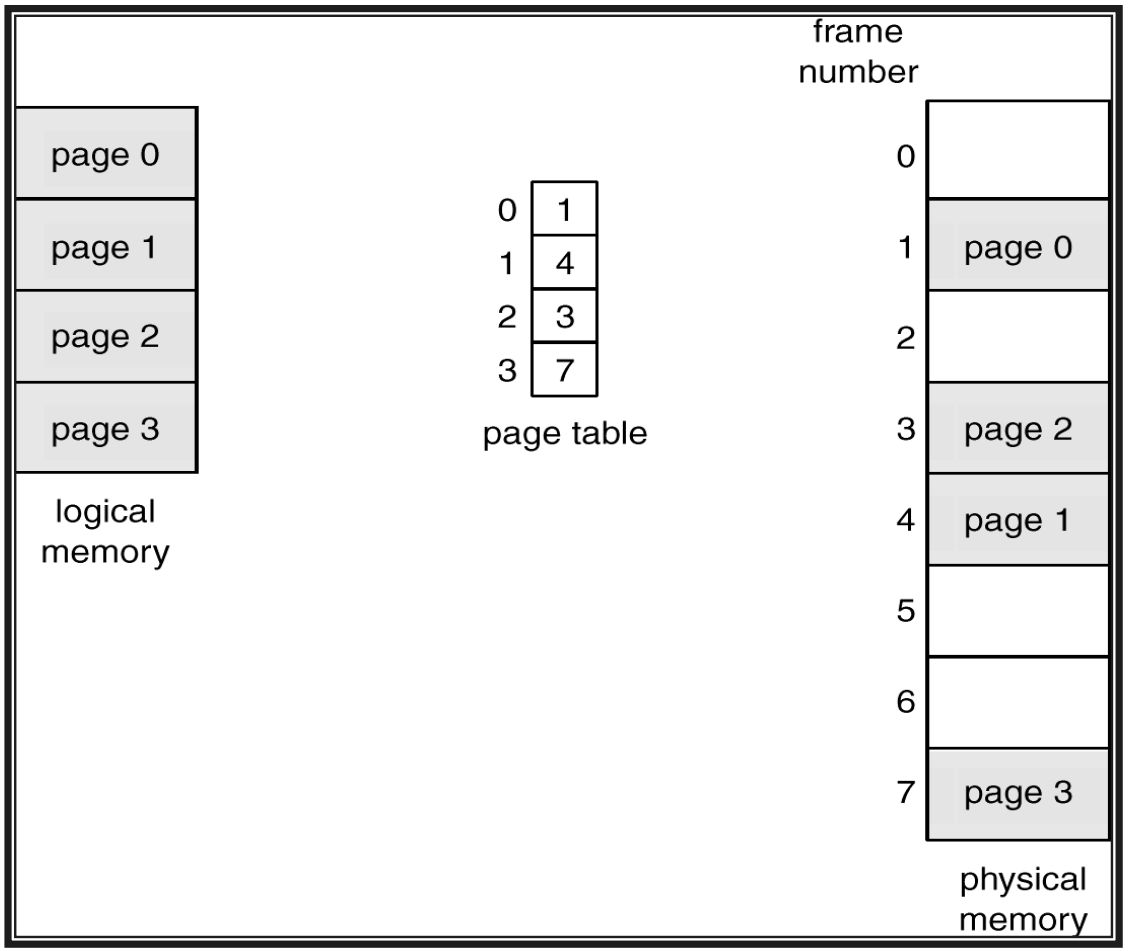
Schema di traduzione degli indirizzi

- Un indirizzo generato dalla CPU è diviso in:
  - *Numero di pagina (p)*
    usato come un indice nella tabella delle pagine che contiene l'indirizzo di base di ogni pagina in memoria fisica.
  - *Offset di pagina (d)*
    usato insieme all'indirizzo base per definire l'indirizzo fisico di memoria da inviare alla unità di memoria.

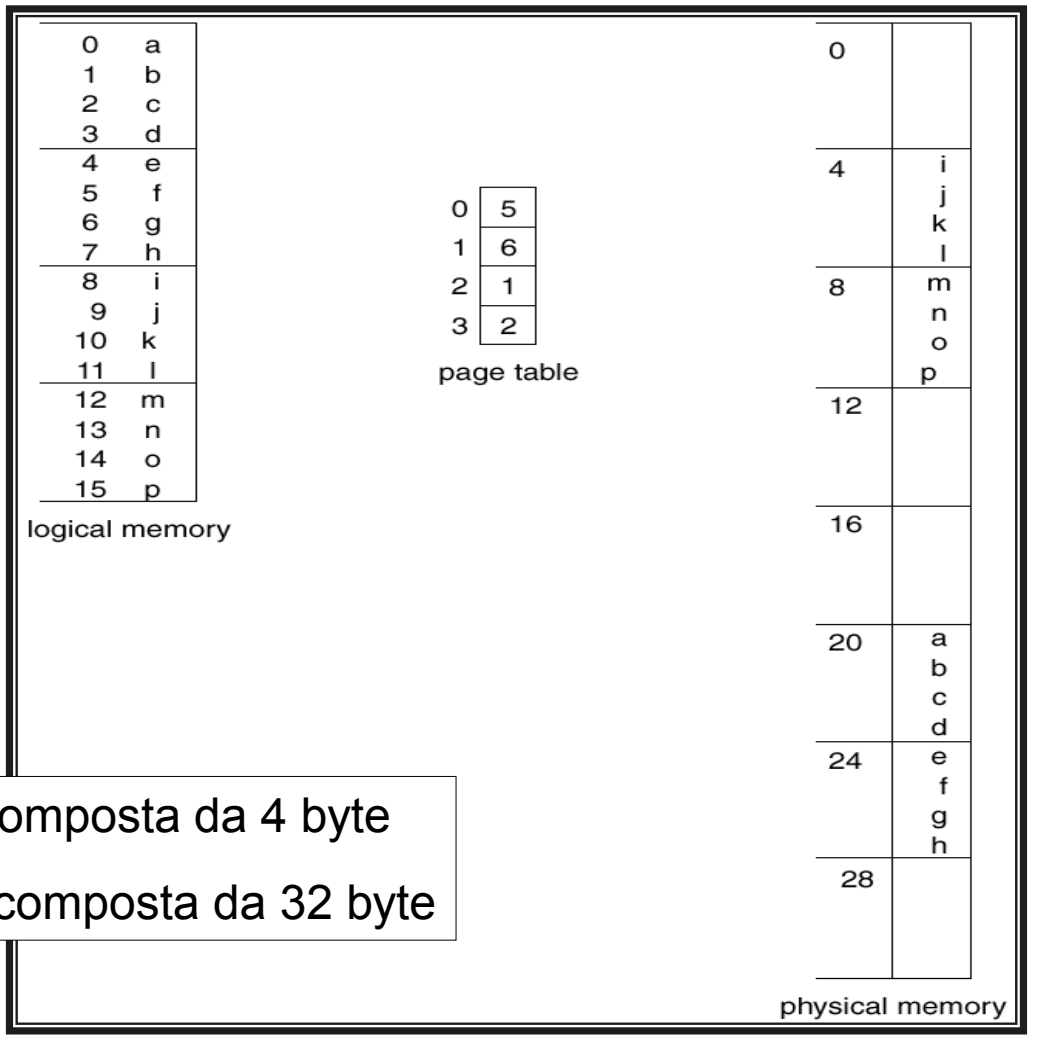
Architettura di traduzione degli indirizzi



Esempio di paginazione



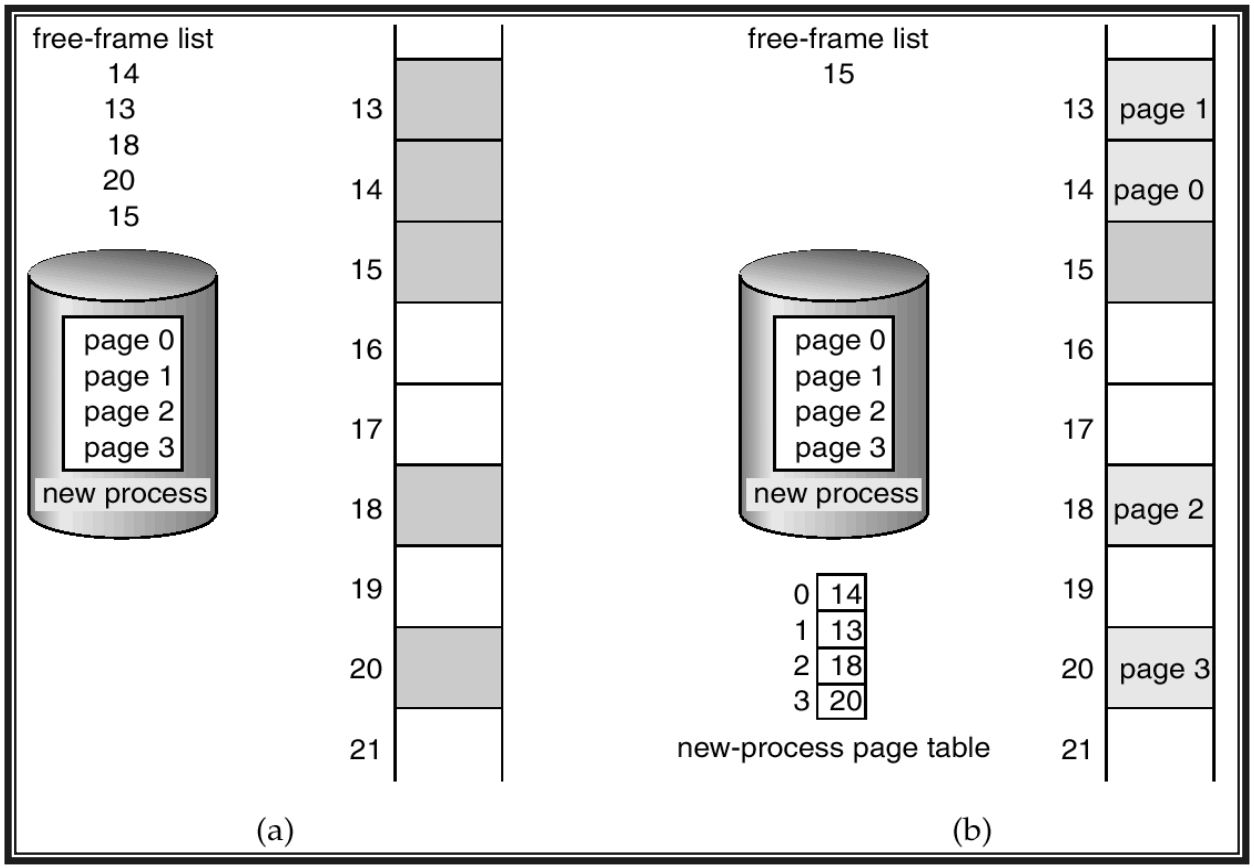
Esempio di paginazione



Ogni pagina è composta da 4 byte

e la memoria è composta da 32 byte

Frame liberi



Prima dell'allocazione

Dopo l'allocazione

Implementazione della tabella delle pagine

- La tabella delle pagine sta in memoria centrale.
- Il *Page-table base register (*PTBR) punta alla tabella.
- Il *Page-table length register* (PRLR) indica la dimensione della tabella.
- Con questo schema l'accesso a dati/istruzioni richiede due accessi alla memoria. Prima alla tabella e poi in memoria.
- Si può risolvere usando una cache hardware detta *memoria associativa.*

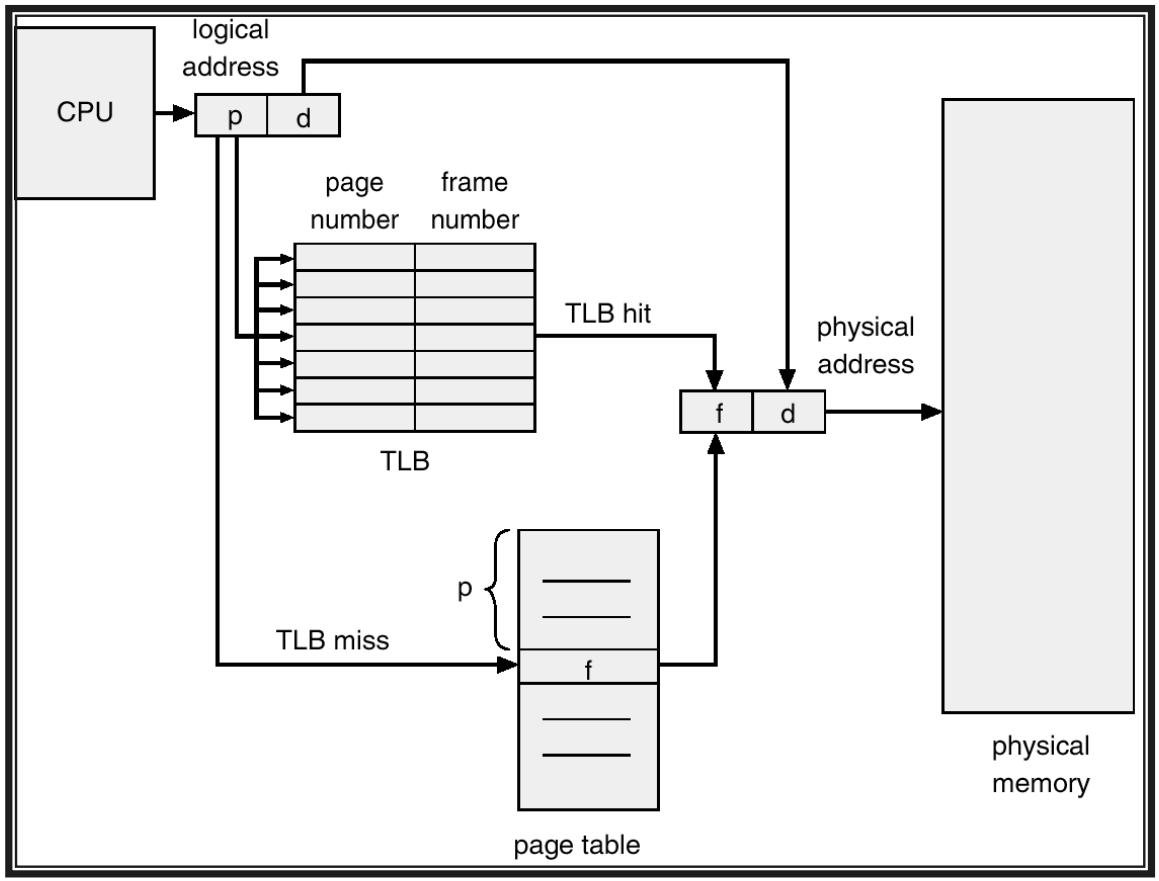
Memoria Associativa

- Memoria associativa – ricerca parallela

Pagina #	Frame #

- Traduzione di indirizzo **(A´, A´´)**
  - Se **A´** è un registro associativo, si ottiene il numero di frame.
  - Altrimenti si ottiene il numero di frame dalla tabella delle pagine e occorre fare un riferimento alla tabella.

Paginazione hardware con memoria associativa



Tempo di accesso effettivo

- *Lookup Associativo* (*la*) - tempo di accesso alla memoria associativa.

- *Hit ratio* (*hr*) – percentuale di volte che un numero di pagina è trovato nei registri associativi; rapporto correlato al numero dei registri.

- Tempo di accesso effettivo

$$TAE = hr \times (la+tm) + (1-hr) \times (la+2tm)$$

$$= 0.9 \times (20+100) + 0.1 \times (20+200)$$

$$= 130 \text{ nsec}$$

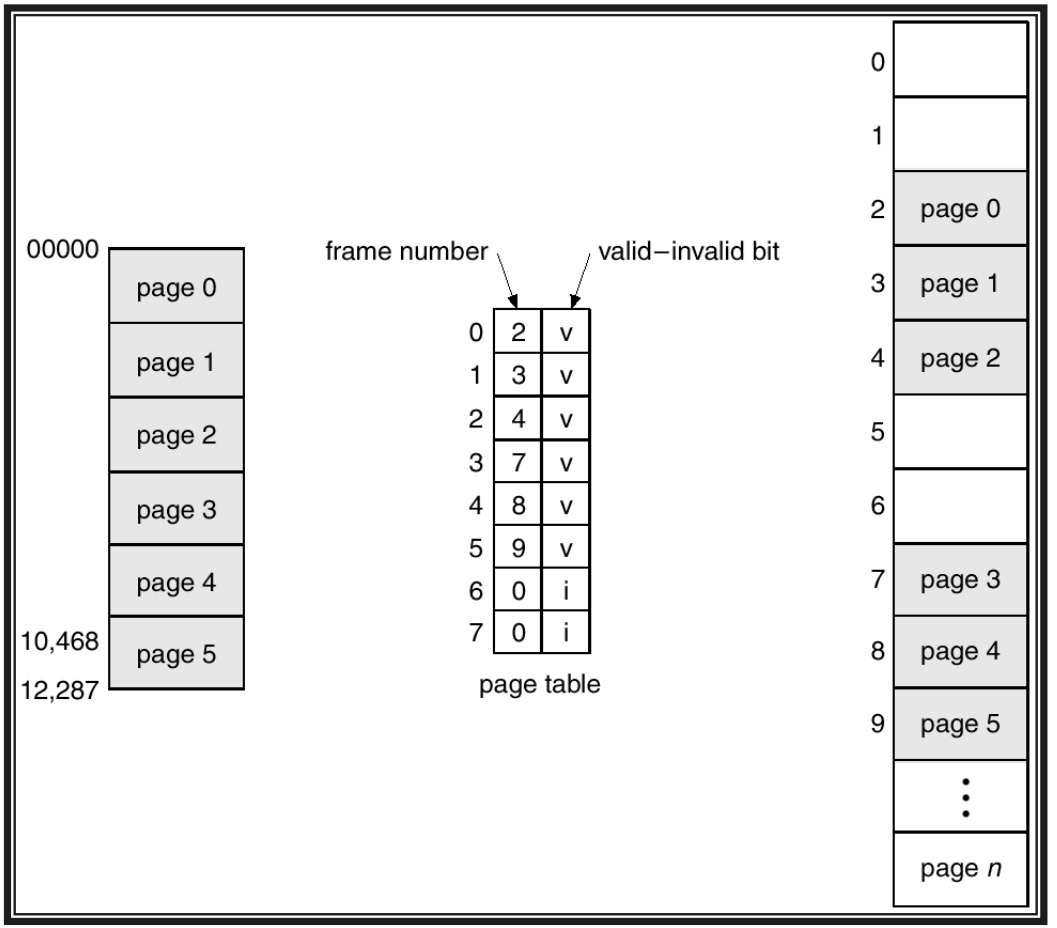
Protezione della memoria

- La protezione della memoria è implementata associando un *bit di protezione* ad ogni frame.

- Il *bit* associato ad ogni entry nella tabella delle pagine:
  - "**valido**" indica che la pagina associata è nello spazio degli indirizzi del processo, e così è una pagina legale.

  - "**non valido**" indica che la pagina associata non è nello spazio degli indirizzi del processo.

Bit Valid (v) o Non valido (i)

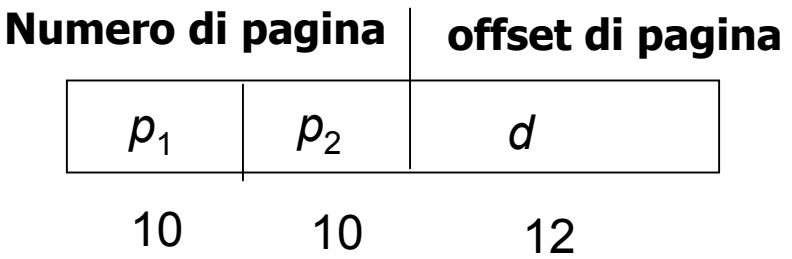


Struttura della tabella

- Paginazione Gerarchica
  - Si divide lo spazio degli indirizzi logici in più tabelle
  - Una tecnica semplice è basata sull'uso una tabella a due livelli.
- Tabella delle pagine invertita

Esempio di tabella a due livelli

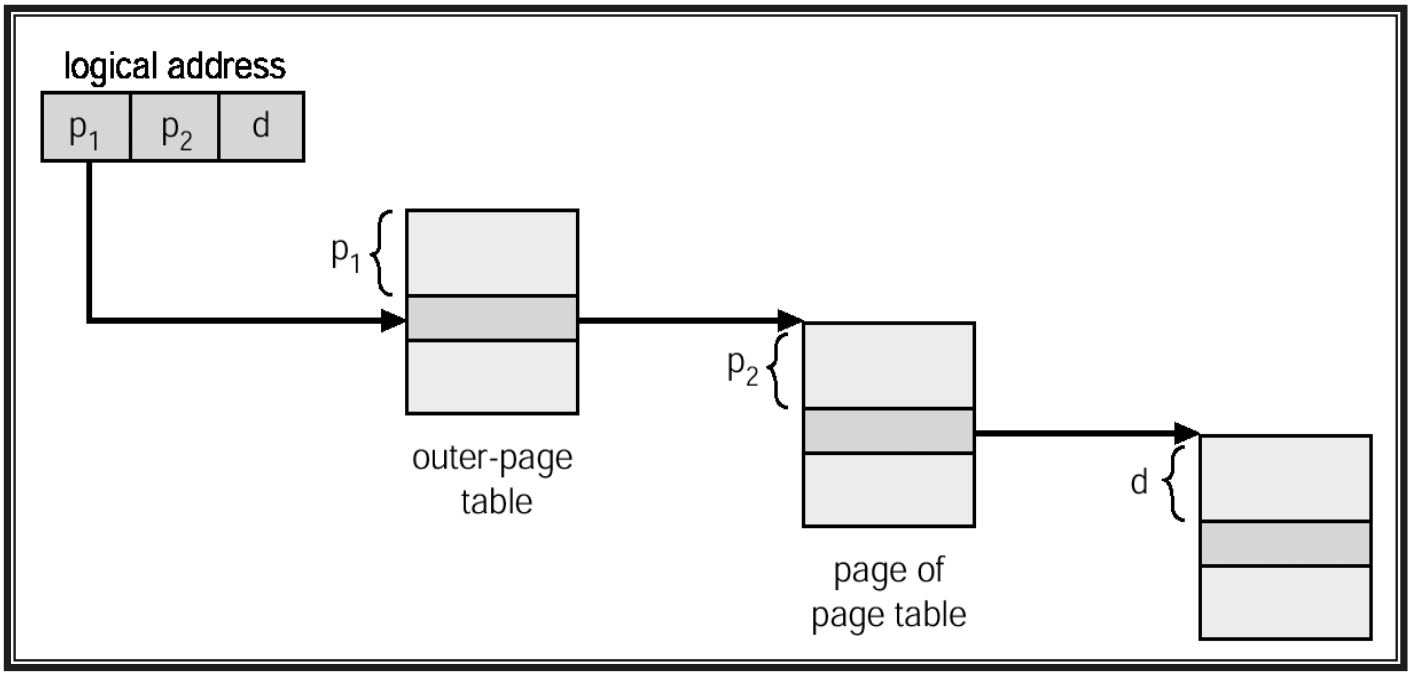
- Un indirizzo logico (su una macchina a 32 bit con pagine di 4KB) è divisa in:
  - un numero di pagina di 20 bit.
  - un offset di pagina di 12 bit.
- Poiché la tabella delle pagine è composta a sua volta da più pagine, il numero di pagina è diviso da:
  - un numero di pagina di 10 bit.
  - un offset di pagina di 10 bit.
- Così un indirizzo logico sarà composto come :



dove p_1 è un indice nella tabella esterna, e p_2 è lo spiazzamento nella tabella interna.

Schema di traduzione degli indirizzi

Schema di traduzione degli indirizzi per un'architettura di paginazione a due livelli.



Schema di tabella a due livelli

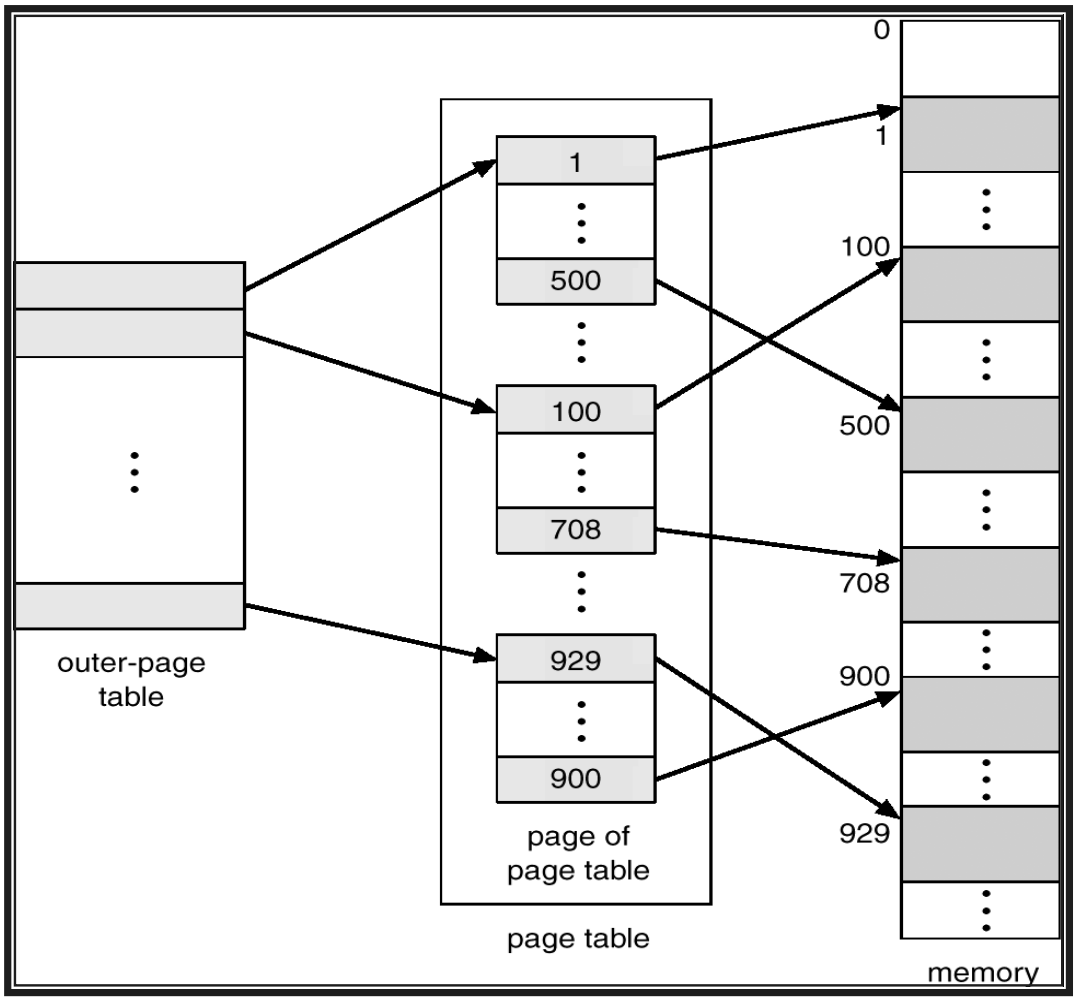
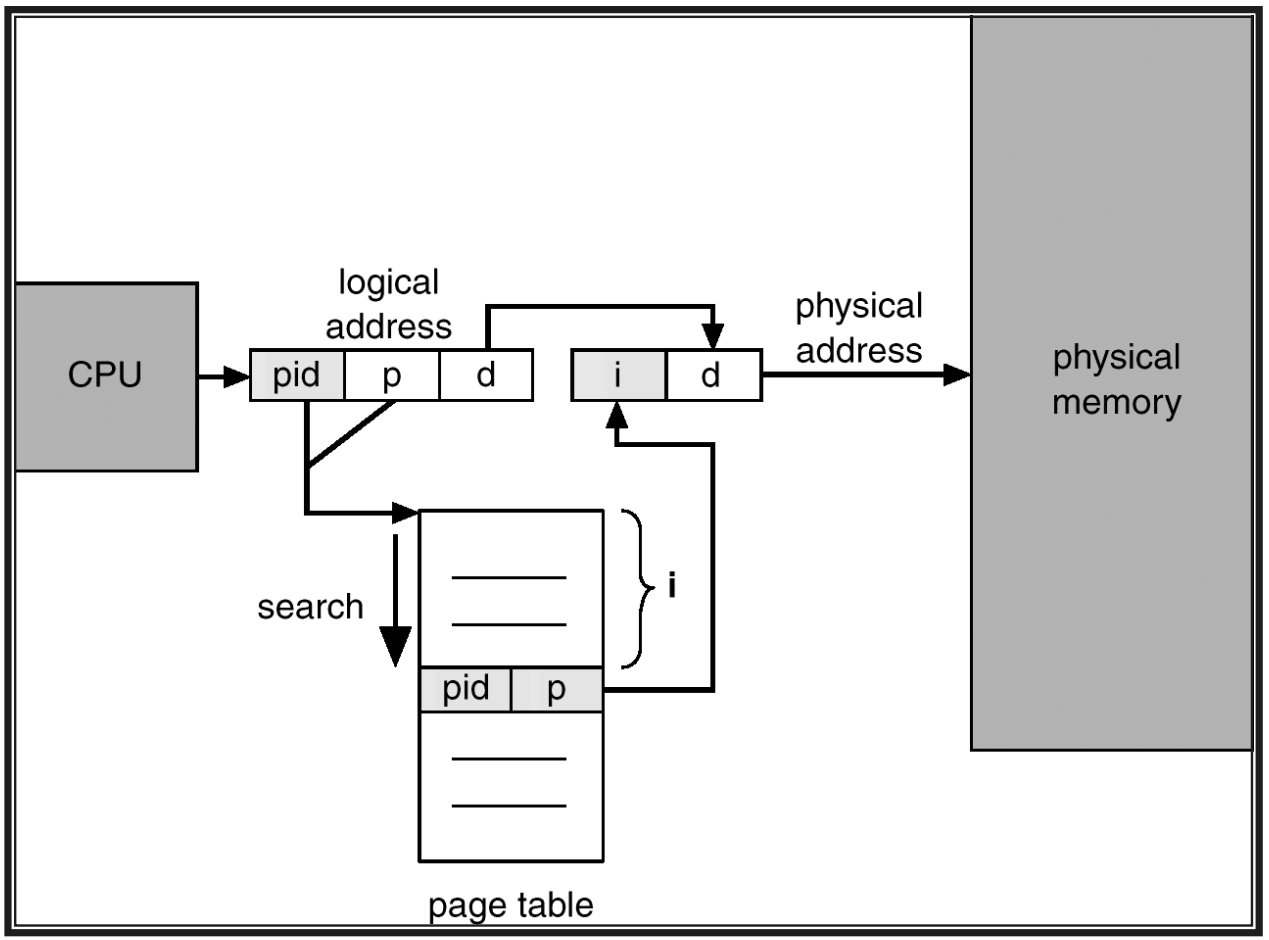


Tabella delle pagine invertita

- Usando questo schema la tabella delle pagine contiene una entry per ogni pagina reale in memoria.
- Ogni entry consiste dell'indirizzo virtuale della pagina e dell'identificatore del processo che possiede quella pagina.
- Diminuisce la memoria necessaria per memorizzare le tabelle delle pagine, **ma** aumenta il tempo per cercare la tabella quando viene fatto un riferimento ad una pagina.
- Si può usare una tabella *hash* per limitare la ricerca a poche entry della tabella delle pagine.

Tabella delle pagine invertita



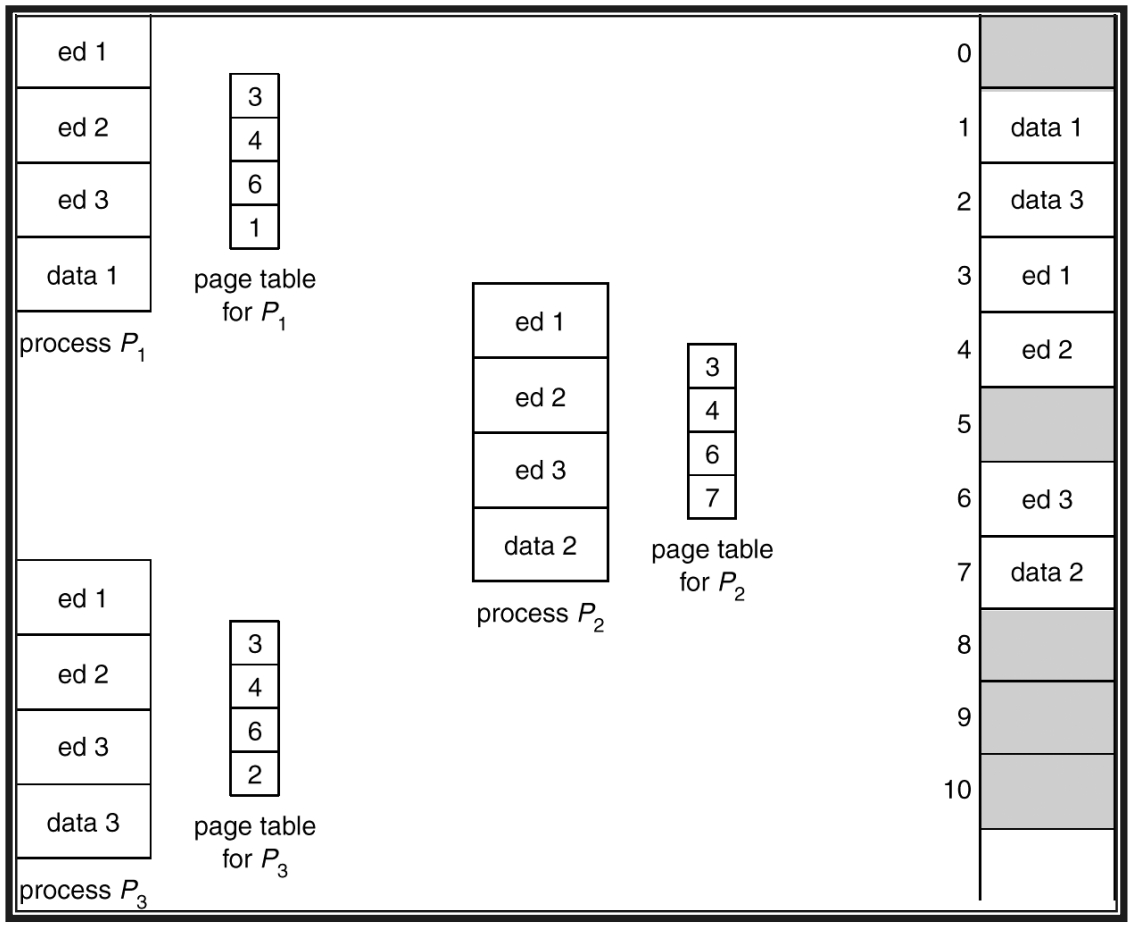
Pagine condivise

- Usando la paginazione si può condividere codice comune.

- Codice condiviso
  - Una singola copia di codice a sola lettura (rientrante) code condivisa tra i processi (i.e., text editor, compilatore, browser).
  - Il codice condiviso deve apparire nella stessa locazione nello spazio degli indirizzi logici di tutti i processi.

- Codice privato e dati
  - Ogni processo mantiene una copia del codice privato e dei dati.
  - Le pagine possono stare in uno qualunque degli indirizzi logici.

Esempio di pagine condivise



Domande

- A cosa serve la tecnica dell'overlay ?
- Descrivere quando accade di avere frammentazione interna e/o frammentazione esterna.
- Quali sono le motivazioni principali per la paginazione ?
- Discutere i pro e i contro del metodo worst-fit anche rispetto agli altri metodi.
- Quando serve usare la tabella delle pagine invertita ?