GESTIONE DELLA

MEMORIA CENTRALE

Gestione della Memoria

- Background
- Spazio di indirizzi
- Swapping
- Allocazione contigua e non contigua
- Segmentazione e Paginazione
- Memoria Virtuale

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.

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.

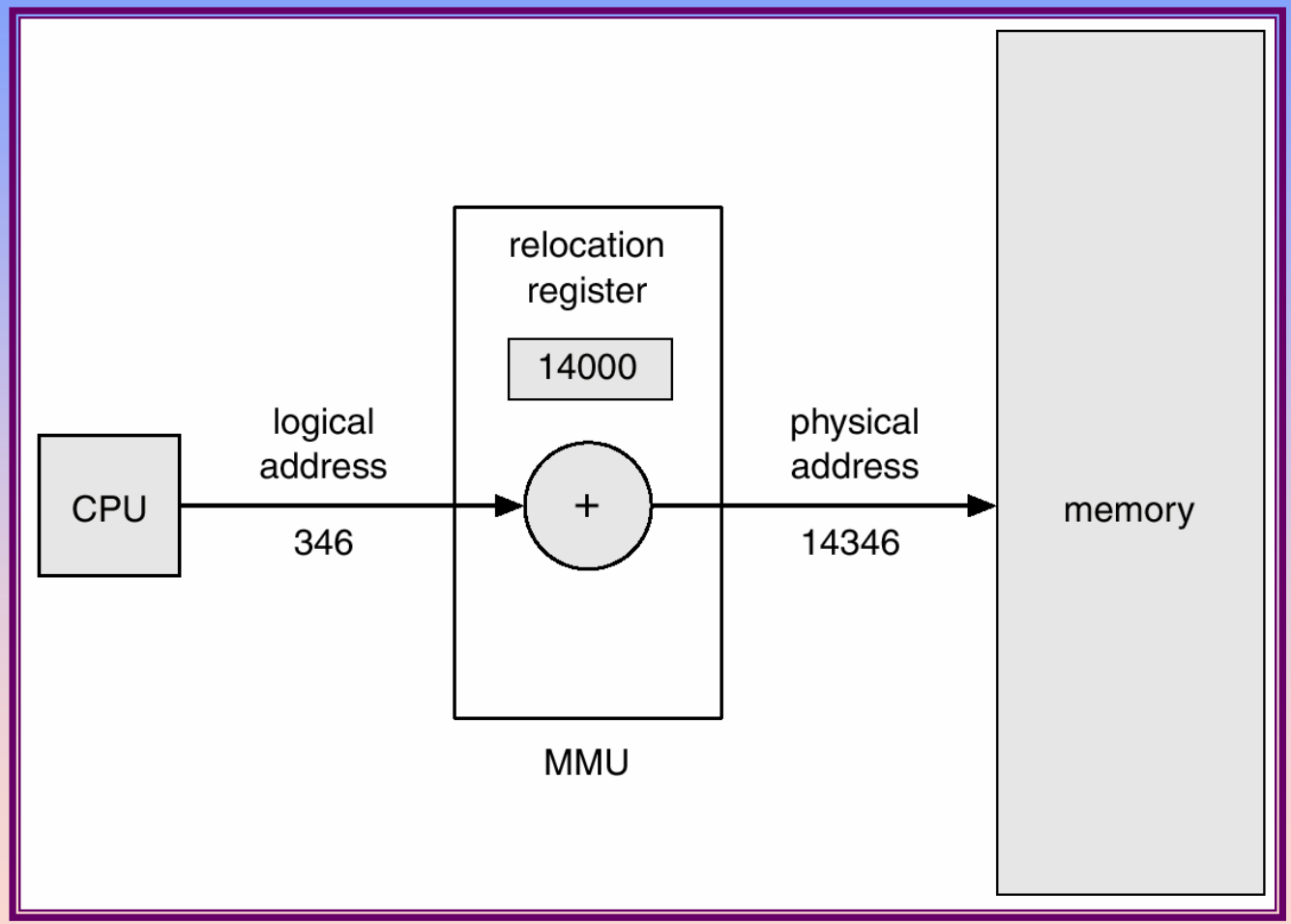
- Gli indirizzi logici e gli indirizzi fisici sono uguali nella compilazione e nel caricamento.

- 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



Swapping

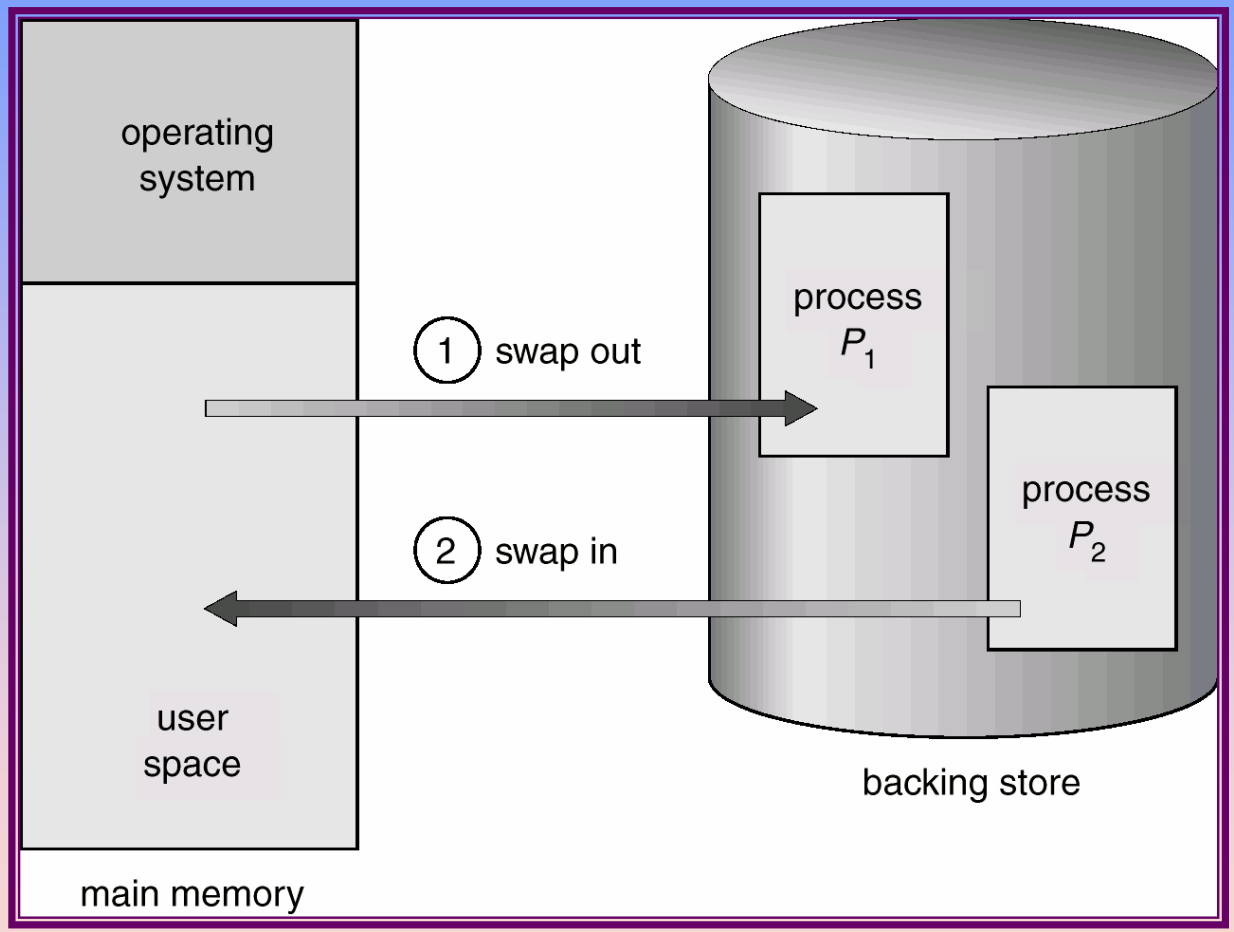
- Un processo può essere temporaneamente riportato (*swapped)* su disco (*backing store*) e quindi riportato in memoria al momento di riprendere l'esecuzione.

- *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

Swap in + swap out

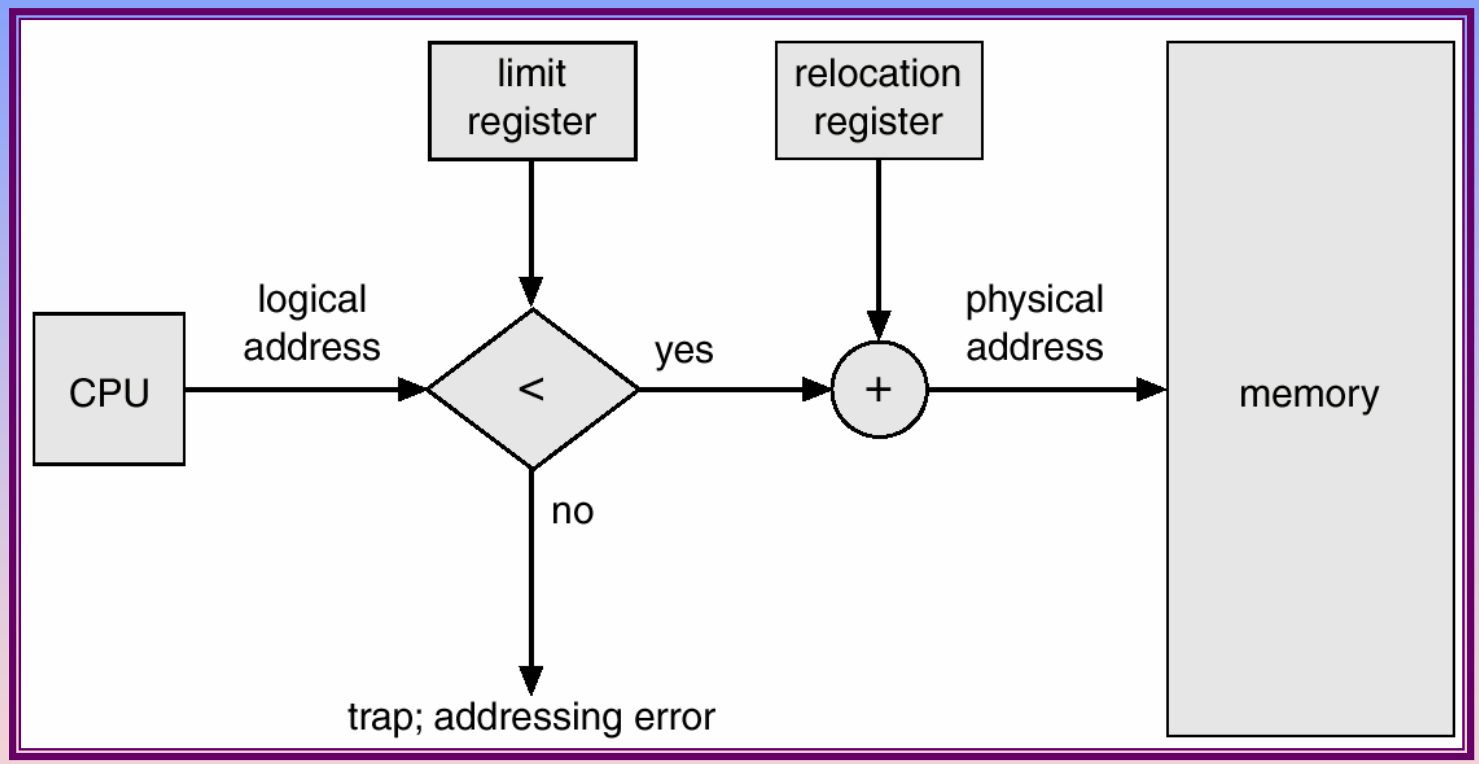
Allocazione Contigua o Segmentazione

- 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.

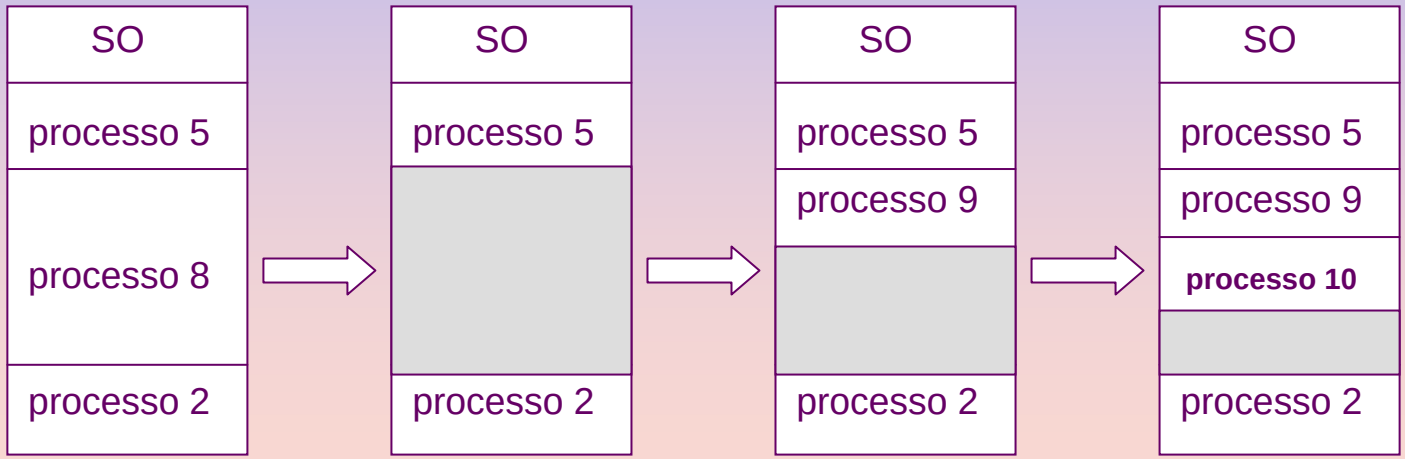
  - Registro di rilocazione contiene l'indirizzo fisico più piccolo e il registro limite contiene l'intervallo degli indirizzi logici: *ogni indirizzo logico deve essere minore del registro limite.*

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.

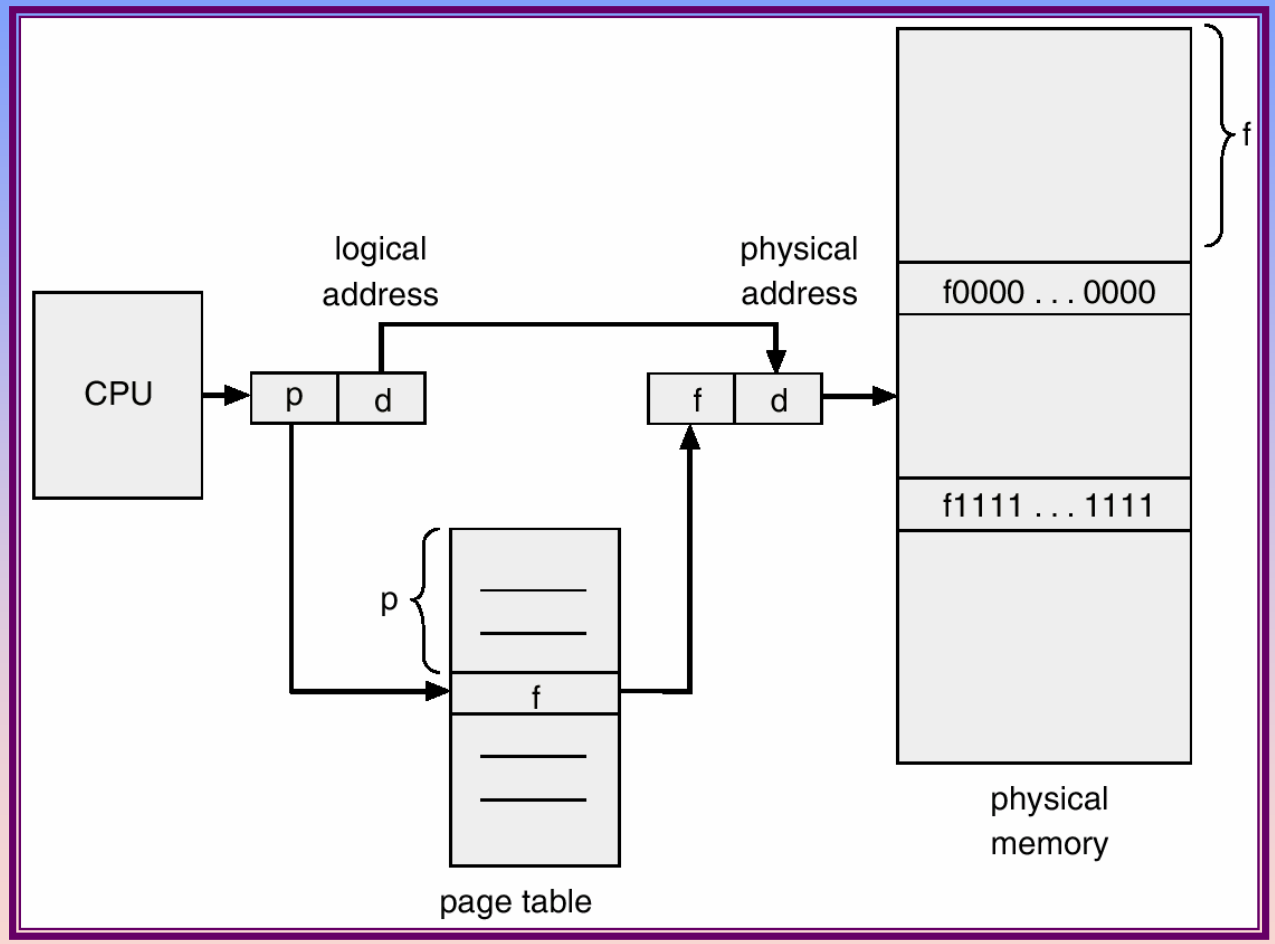
Schema di traduzione degli indirizzi

- Un indirizzo generato dalla CPU è diviso in:
  - *Numero di pagina (p)*

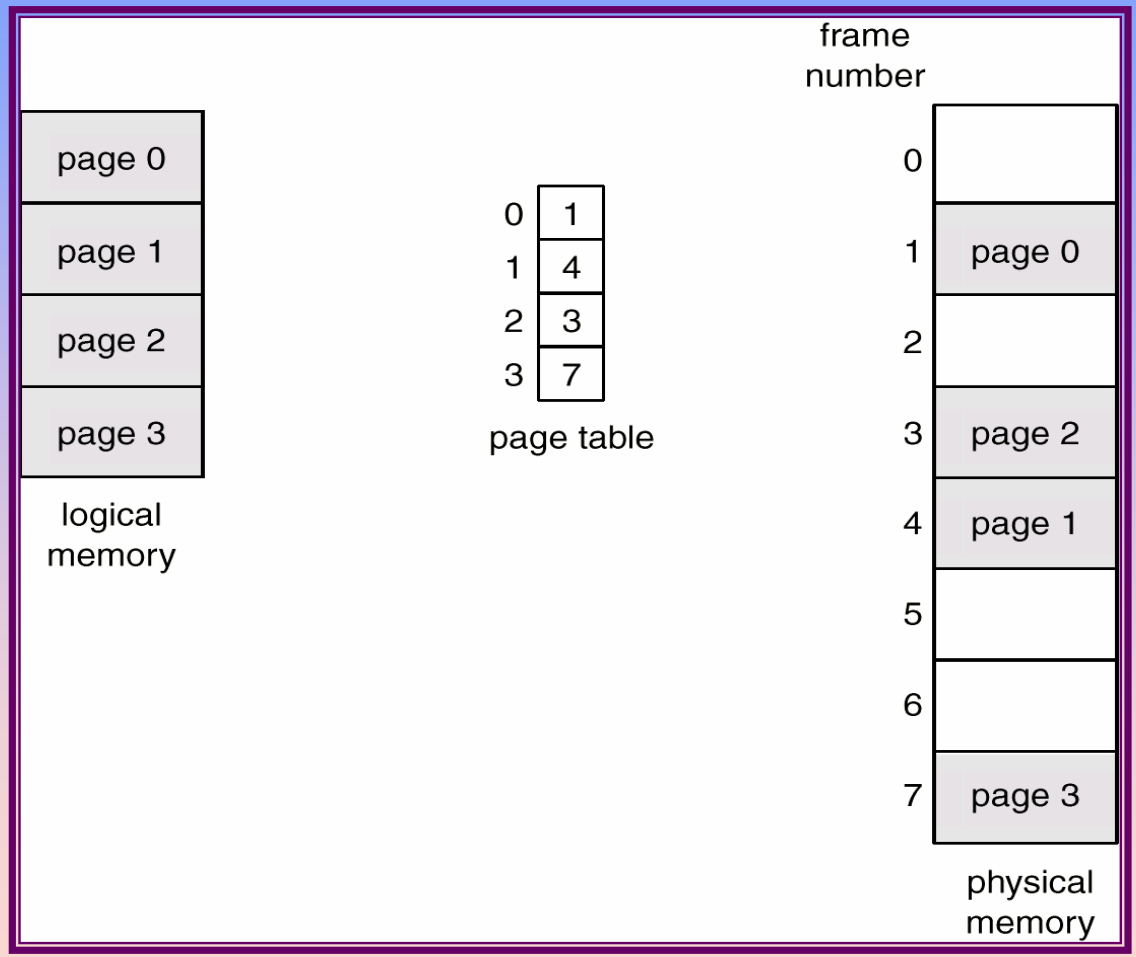
    usato come un indice nella tabella delle pagine che contiene l'indirizzo di base di ogni pagina in memoria fisica.
  - *Offset di pagina (d)*

    usato insieme all'indirizzo base per definire l'indirizzo fisico di memoria da inviare alla unità di memoria.

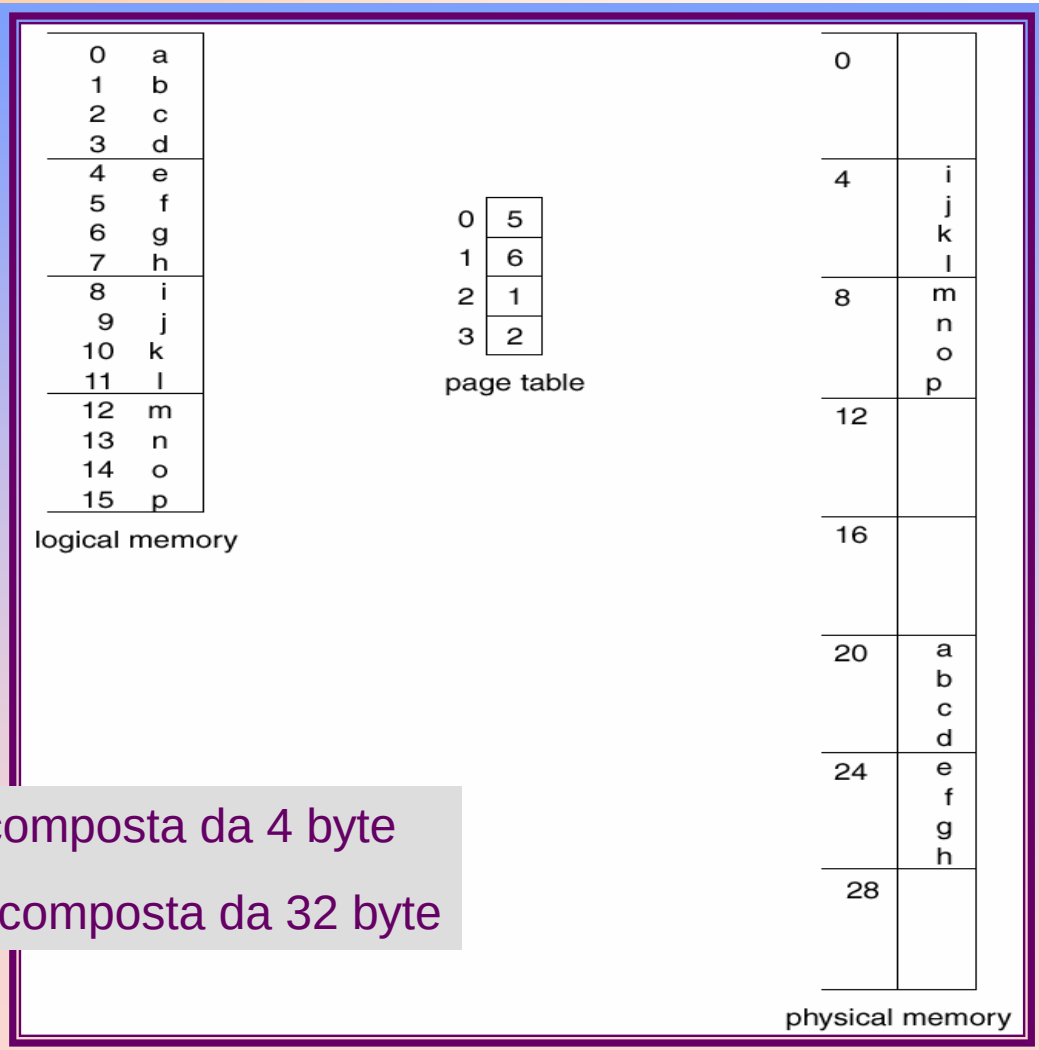
Architettura di traduzione degli indirizzi



Esempio di paginazione



Esempio di paginazione



Ogni pagina è composta da 4 byte

e la memoria è composta da 32 byte

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.

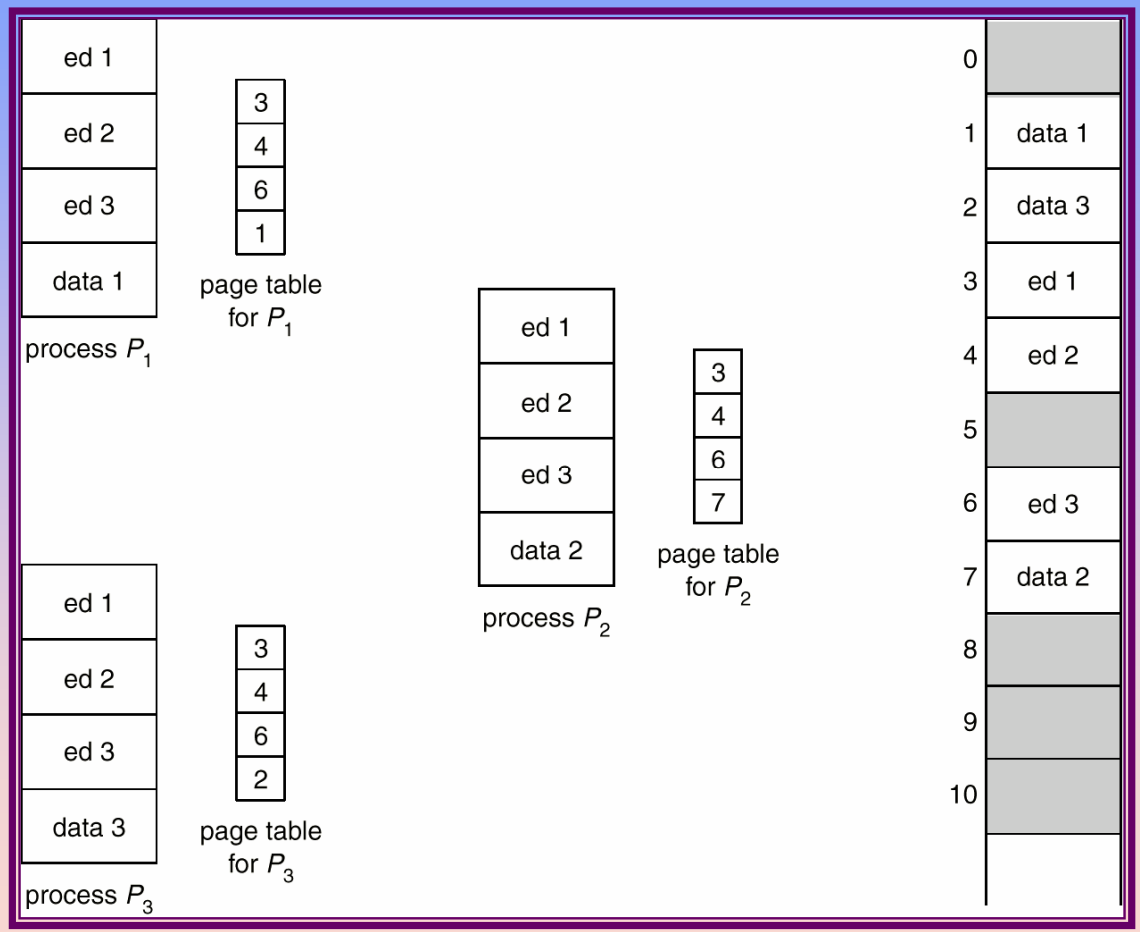
Pagine condivise

- 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



Segmentazione con Paginazione

- Il S.O. MULTICS ha risolto i problemi di frammentazione esterna e dei tempi lunghi di ricerca tramite la paginazione dei segmenti.

- **Un segmento viene realizzato tramite un insieme di pagine**.

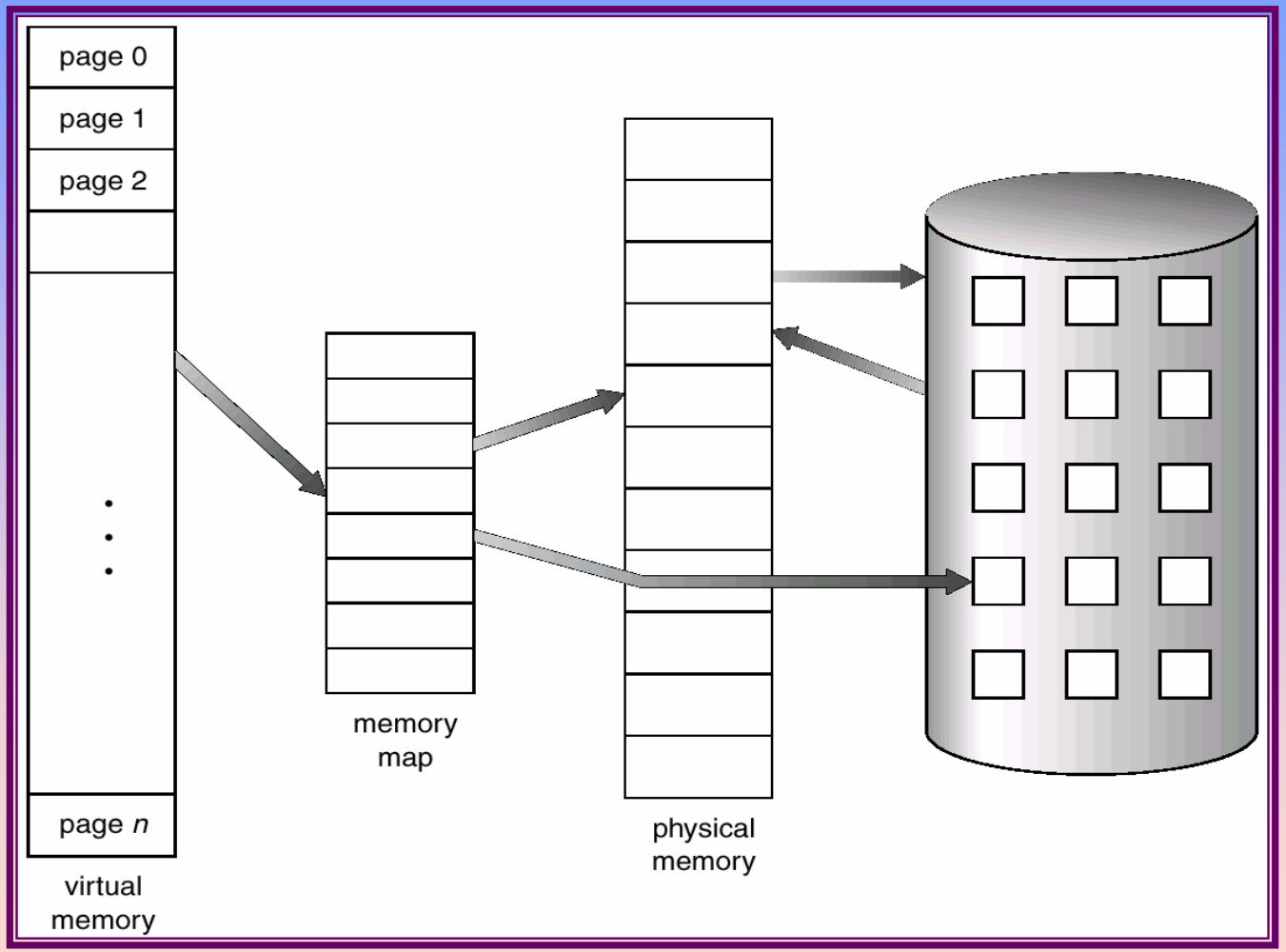
- Soluzione differisce dalla segmentazione “pura” poiché una entry nella tabella dei segmenti non contiene l’indirizzo base di un segmento, ma l’indirizzo base della tabella delle pagine di quel segmento.

Memoria virtuale

- **Memoria Virtuale** – *separazione della memoria logica dalla memoria fisica*.
  - Solo una parte del programma sta in memoria per l'esecuzione.
  - Lo spazio degli indirizzi logici è quindi più grande dello spazio degli indirizzi fisici.
  - Lo spazio degli indirizzi può essere diviso tra più processi.
  - Permette una più efficiente creazione dei processi.

- La memoria virtuale può essere implementata tramite:
  - Paginazione su richiesta
  - Segmentazione su richiesta

Memoria Virtuale maggiore della Memoria Fisica

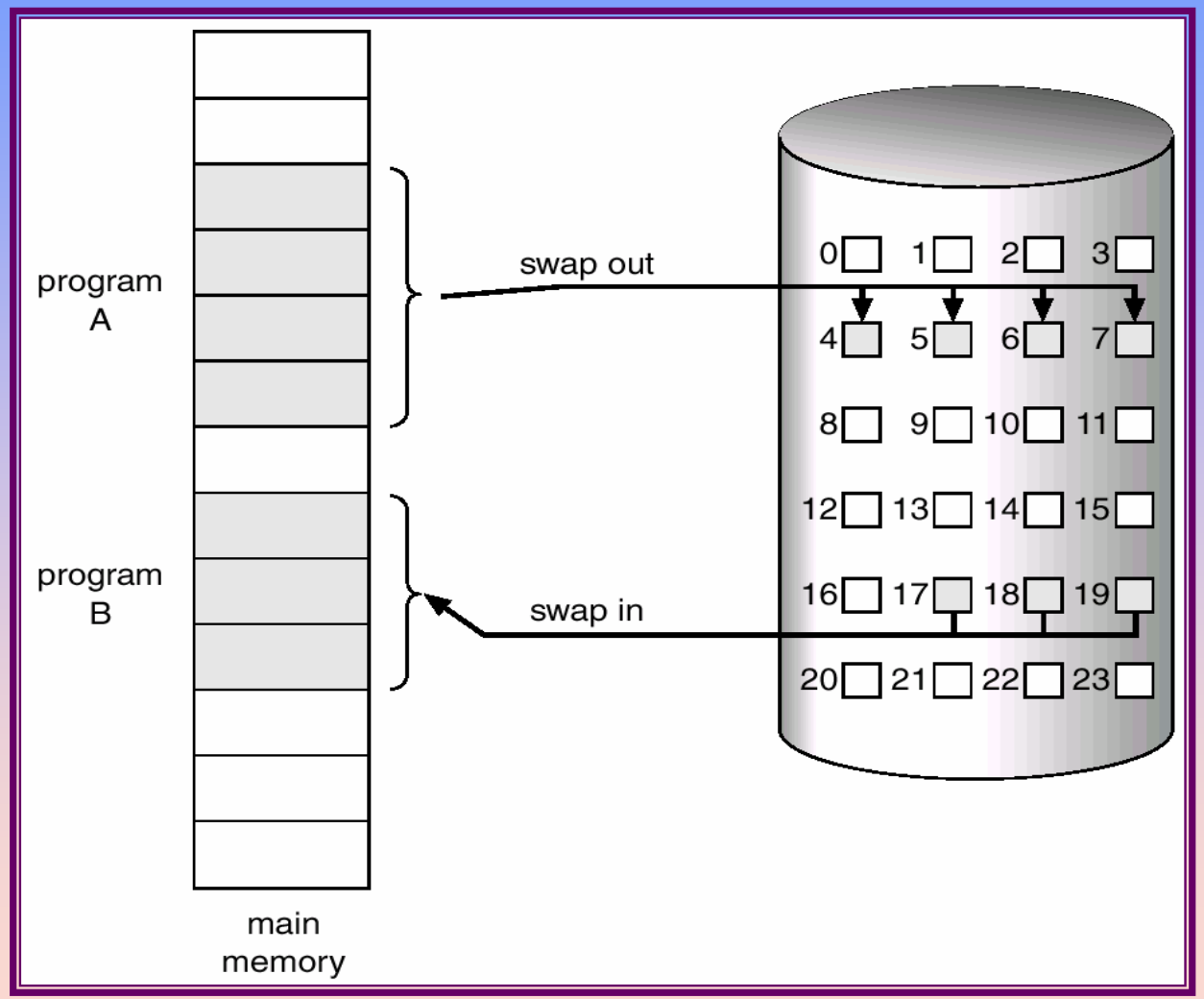


Paginazione su richiesta

- Si porta una pagina in memoria solo quando serve.
  - minore I/O
  - minore memoria necessaria
  - risposta più veloce
  - Maggior numero di utenti

- Quando serve una pagina $\Rightarrow$ riferimento ad essa
  - riferimento non valido $\Rightarrow$ ***abort***
  - non in memoria $\Rightarrow$ trasferire in memoria

Trasferimento di memoria paginata su disco contiguo



Bit Valido/Non valido

- Ad ogni entry della tabella è associato un bit di validità (1 ⇒ in memoria, 0 ⇒ non in memoria ma su disco)
- Inizialmente il bit di validità è uguale a 0 per ogni entry.
- Esempio di tabella delle pagine:

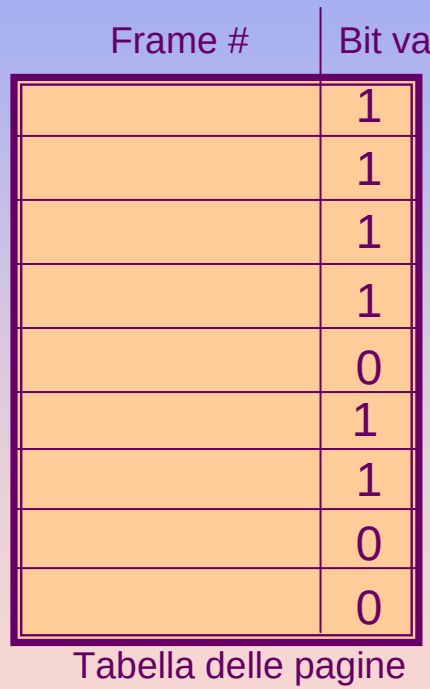
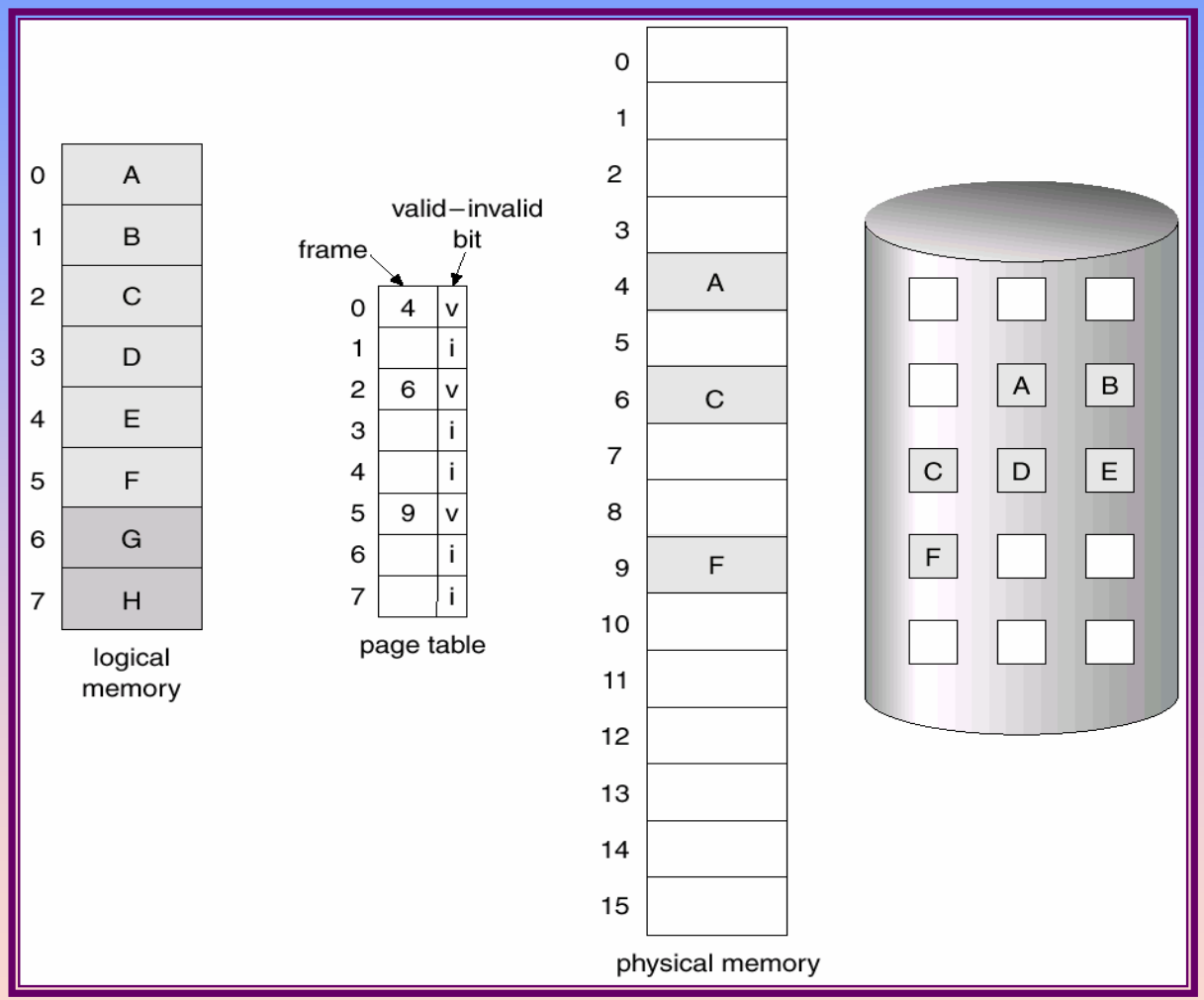


Tabella delle pagine

- Durante la traduzione degli indirizzi, se il bit è 0 ⇒ **page fault.**

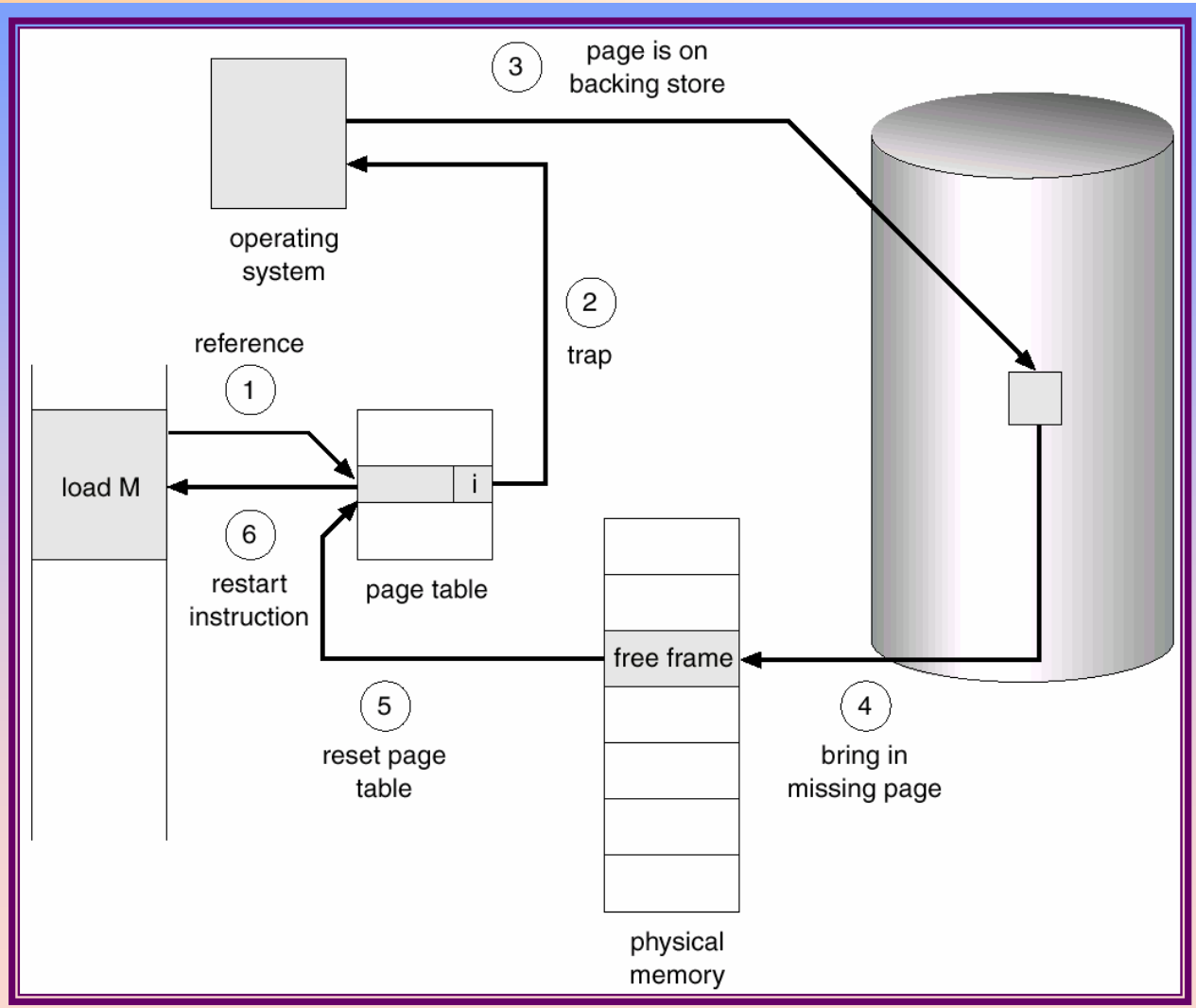
Tabella delle pagine con alcune pagine non in memoria



Page Fault

- Se si tenta di accedere una pagina non in memoria ⇒ **page fault**.
- Il S.O. controlla in una tabella interna del processo:
  - se il riferimento non è valido ⇒ **abort**.
  - Se il riferimento è valido occorre caricare la pagina.
- Si trova un frame libero.
- Si carica la pagina nel frame.
- Si aggiorna le tabelle, bit di validazione = 1.
- Viene riavviata l'esecuzione: la pagina diventa quella più recente.

Passi della gestione di un Page Fault



Cosa accade quando non c’è un frame libero ?

- **Sostituzione delle pagine**

  si trova una pagina in memoria che non è usata e si porta sul disco (*swap out*).

  - algoritmo
  - prestazioni

    si vuole un algoritmo che dia il numero minimo di page fault.

- Alcune pagine possono essere portate in memoria varie volte.

Sostituzione delle pagine

- Quando occorre caricare una pagina e non c'è un frame libero si può usare la **sostituzione delle pagine**.
- Il meccanismo di gestione dei page fault deve essere modificato per gestire questa possibilità.
- Uso di un *modify* (*dirty*) *bit* per ridurre il costo del trasferimento delle pagine – solo le pagine modificate vengono scritte sul disco.
- La sostituzione delle pagine completa la separazione tra memoria logica e memoria fisica: un grande memoria virtuale si può realizzare su una piccola memoria fisica.

Sostituzione delle pagine

Operazioni per la sostituzione:

- Trova la locazione della pagina richiesta sul disco.
- Trova un frame libero
  - Se esiste usalo;
  - Se non c'è un frame libero seleziona un frame ***vittima*** secondo un algoritmo di sostituzione;
  - Scrivi la pagina vittima sul disco e aggiorna le tabelle;
- Leggi la pagina richiesta nel frame liberato e aggiorna le tabelle
- Riavvia il processo.

Algoritmo First-In-First-Out (FIFO)

- Si sostituisce la pagina più vecchia.
- Stringa: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frame (3 pagine possono essere in memoria per processo)

1	1	4	5	9 page fault
2	2	1	3	
3	3	2	4	

- 4 frame

1	1	5	4	10 page fault
2	2	1	5	
3	3	2		
4	4	3		

Algoritmo Ottimale

- **Sostituisce la pagina che non verrà usata per il periodo di tempo più lungo**.
- Esempio: 4 frame

stringa: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	4
2	
3	
4	5

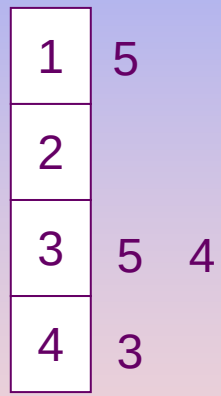
6 page fault

- Come predire il futuro (uso delle pagine) ?
- Usato come paragone per valutare altri algoritmi.

Algoritmo Least Recently Used (LRU)

- **Sostituisce la pagina che non è stata usata per il periodo di tempo più lungo**.

- Stringa: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



Domande

- Descrivere quando accade di avere frammentazione interna e/o frammentazione esterna.
- Discutere i pro e i contro del metodo worst-fit anche rispetto agli altri metodi.
- Discutere le differenze principali tra paginazione e segmentazione.
- Quali sono i benefici di usare la segmentazione paginata ?
- Spiegare le operazioni da eseguire per la gestione di un page fault.
- Discutere le differenze tra gli algoritmi di sostituzione FIFO e LRU.