Gestione della memoria centrale

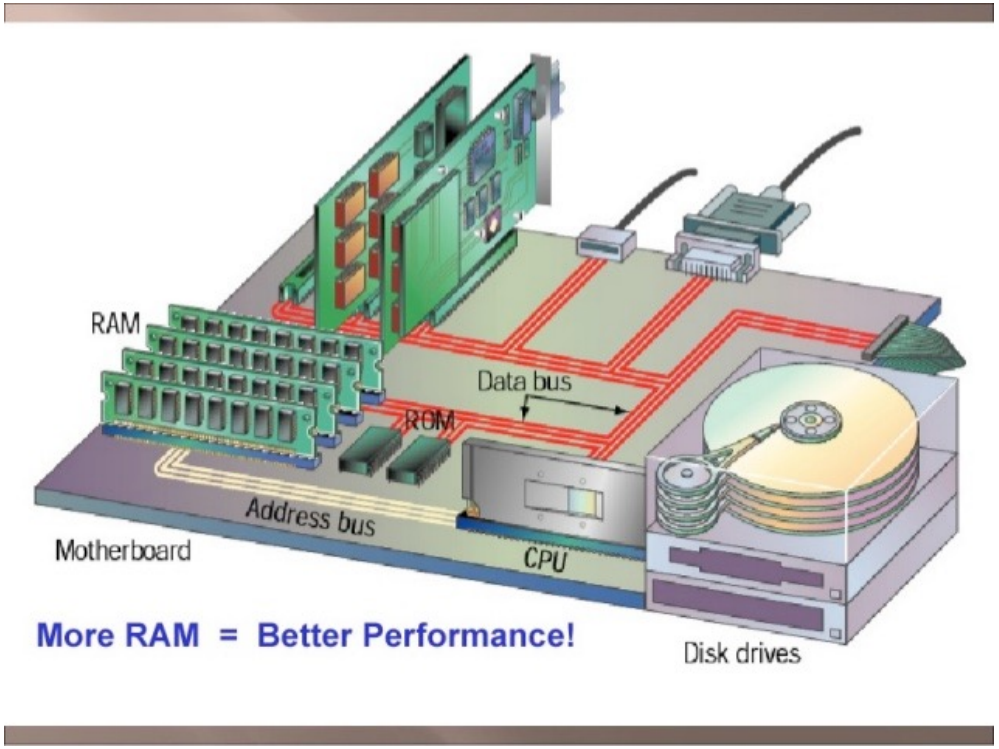
Memoria centrale

- Introduzione
- Overlay
- Avvicendamento dei processi (*swapping*)
- Allocazione contigua della memoria
- Paginazione
  - Tabella delle pagine
- Segmentazione
  - Tabella dei segmenti
- Segmentazione con paginazione
- Esempi

Memoria centrale

Memoria centrale o **principale**, contiene dati e istruzioni prelevati dalla memoria di massa in attesa che siano prelevati ed elaborati dalla CPU.

Solitamente implementata tramite una memoria **RAM** (*Random Access Memory).*



Background

- La gestione della memoria centrale dipende dalle funzionalità che l'hardware del computer mette a disposizione del S.O.
- **Per essere eseguito un programma deve trovarsi (almeno parzialmente) in memoria centrale.**
- 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 essere eseguiti.

Associazione di istruzioni e dati alla memoria

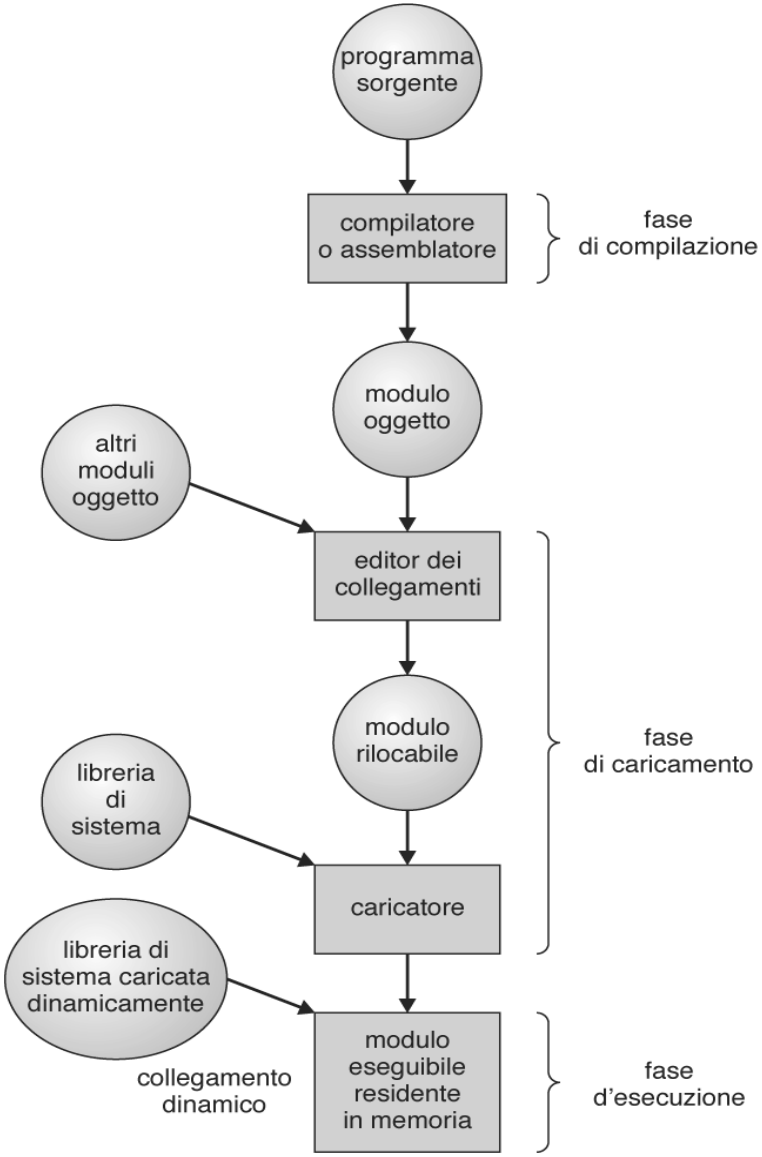
L'associazione di istruzioni e dati alla memoria (***address binding***) serve per eseguire i programmi e può avvenire in momenti diversi:

- **Compilazione**: se la locazione di memoria è conosciuta a priori possono essere generati ***indirizzi assoluti***. Ma in questo caso, la ricompilazione è necessaria quando la locazione di partenza del processo cambia.

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

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

Fasi di elaborazione per un programma utente



Indirizzi fisici e indirizzi logici

- Il concetto di ***spazio di indirizzi logici/virtuali*** 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.

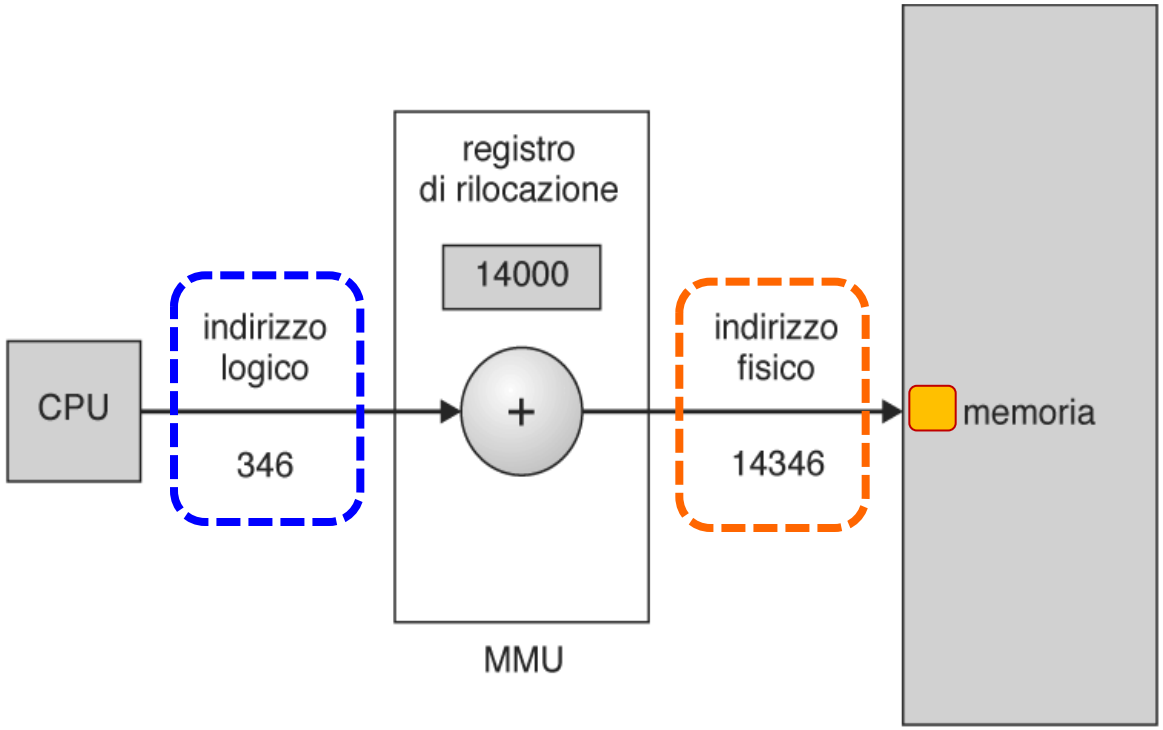
- Gli indirizzi logici e gli indirizzi fisici sono uguali nel caso dell'*address binding* durante la compilazione e nel caricamento.

- Nell'*address binding* durante l'esecuzione gli indirizzi logici sono detti virtuali e differiscono dagli indirizzi fisici.

Memory-Management Unit (MMU)

- C'è bisogno di un'associazione (*mapping*) a tempo di esecuzione.
- **MMU:** *dispositivo hardware che associa indirizzi virtuali a indirizzi fisici.*
- La MMU realizza l'associazione tra indirizzi seguendo lo schema di gestione della memoria centrale.
- Ad esempio, tramite la 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



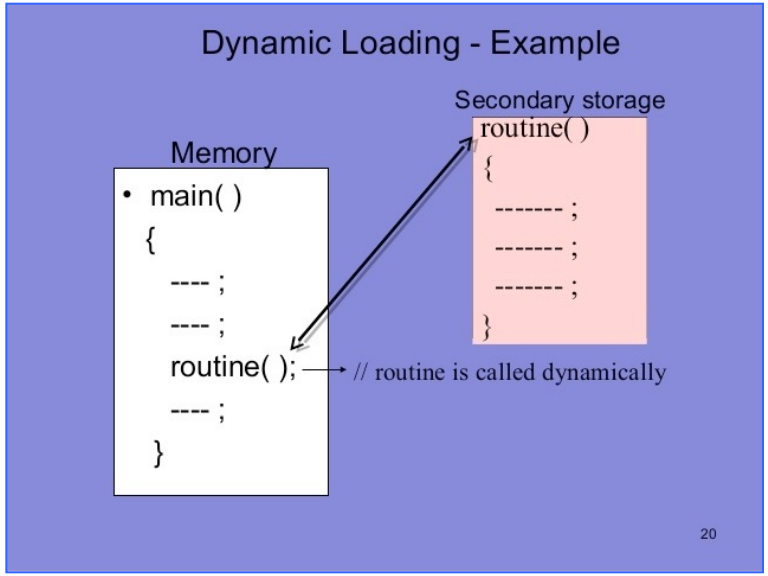
Caricamento dinamico

- Tramite il **caricamento dinamico** le routine (procedure, funzioni, metodi) vengono caricate quando sono chiamate (se non sono già in memoria).

- **Migliore uso dello spazio di memoria**; le routine mai usate non vengono mai caricate in memoria centrale.

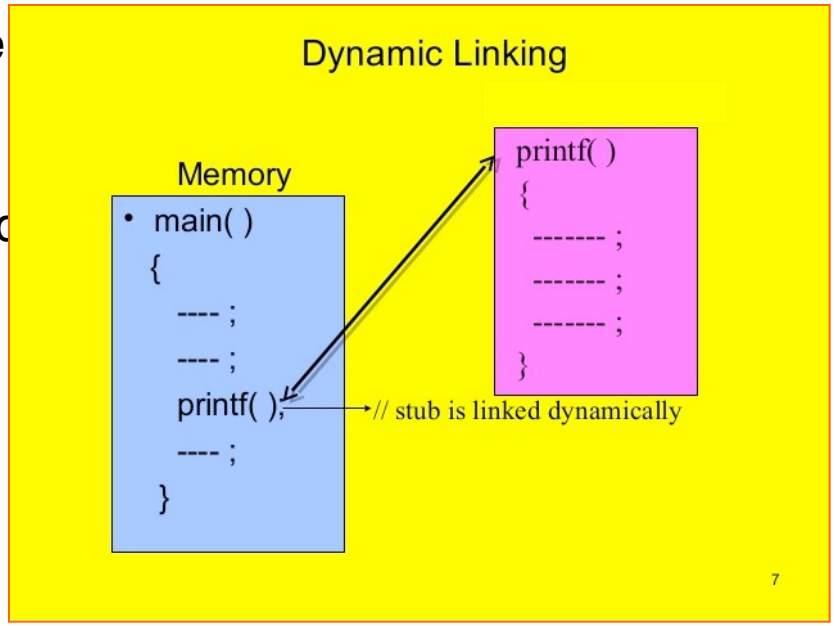
- Utile quando molta parte del codice è usata raramente.

- Nessun speciale supporto del S.O. In alcuni casi esistono librerie per il caricamento dinamico.



Collegamento dinamico

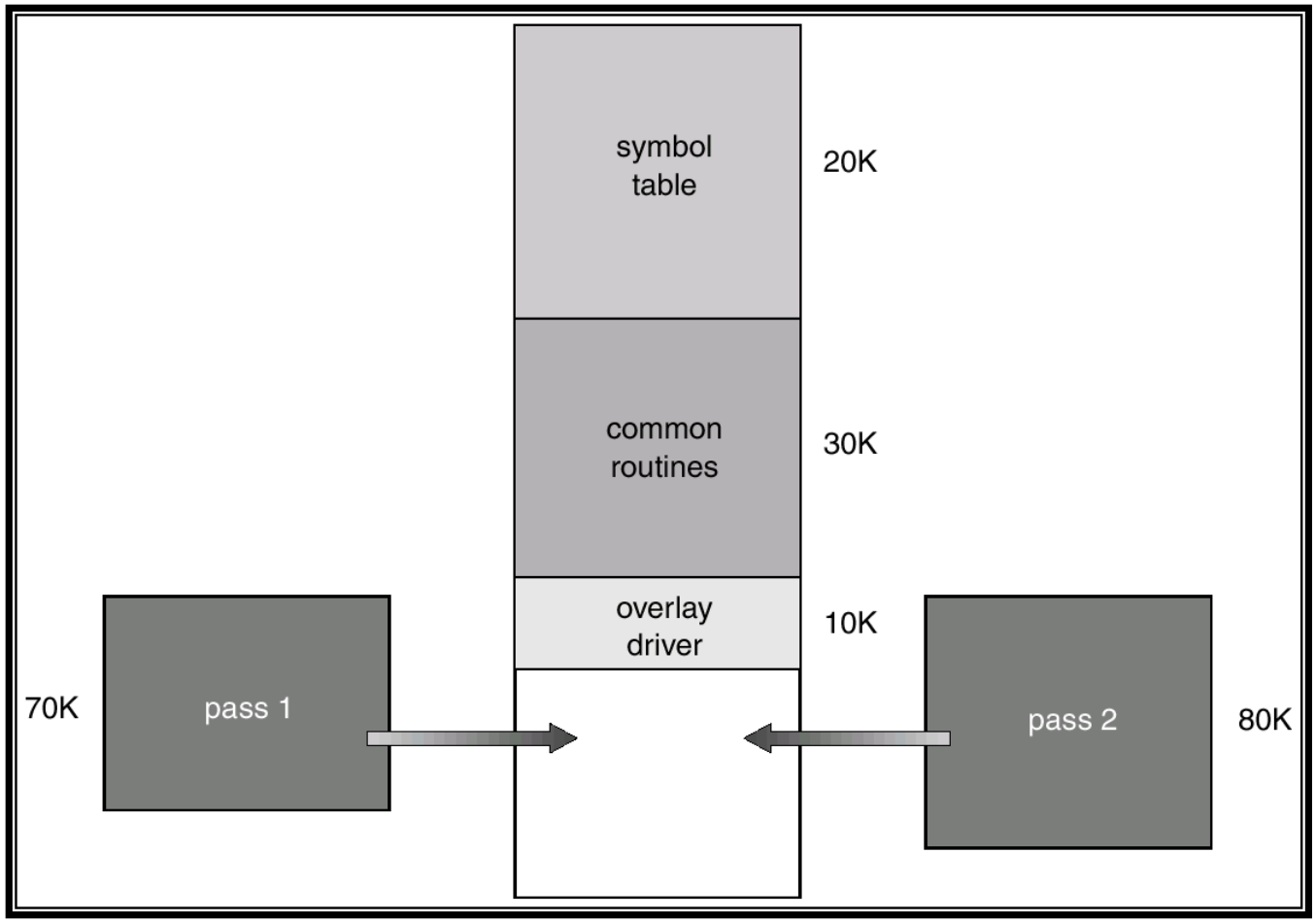
- Il collegamento dinamico serve ad utilizzare parti di codice comuni a più programmi.
- Il collegamento è ritardato fino al momento dell'esecuzione.
- Codice ***stub***, usato per localizzare (in memoria o sul disco) la routine di sistema richiesta.
- Lo ***stub*** si sostituisce con l'indirizzo de eseguita.
- Il sistema operativo permette l'utilizzo d processi.
- Utile per le librerie di sistema.



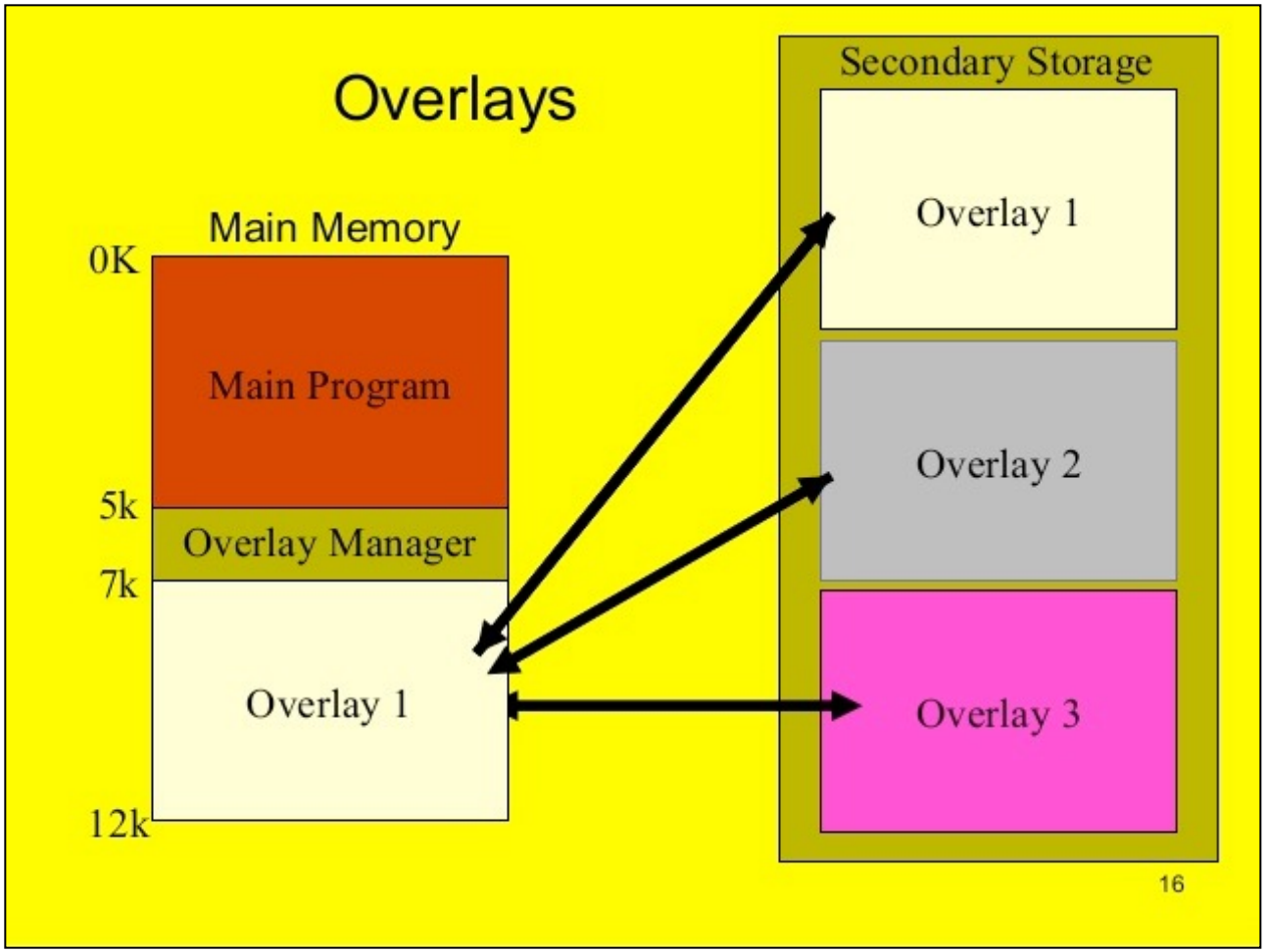
Overlay

- La tecnica dell'**overlay** è usata per eseguire processi che non "entrano" in memoria.
- Le dimensioni dello spazio degli indirizzi del processo sono superiori alla RAM disponibile.
- 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 programma a due passi



Overlay per un assembler a tre passi



Swapping

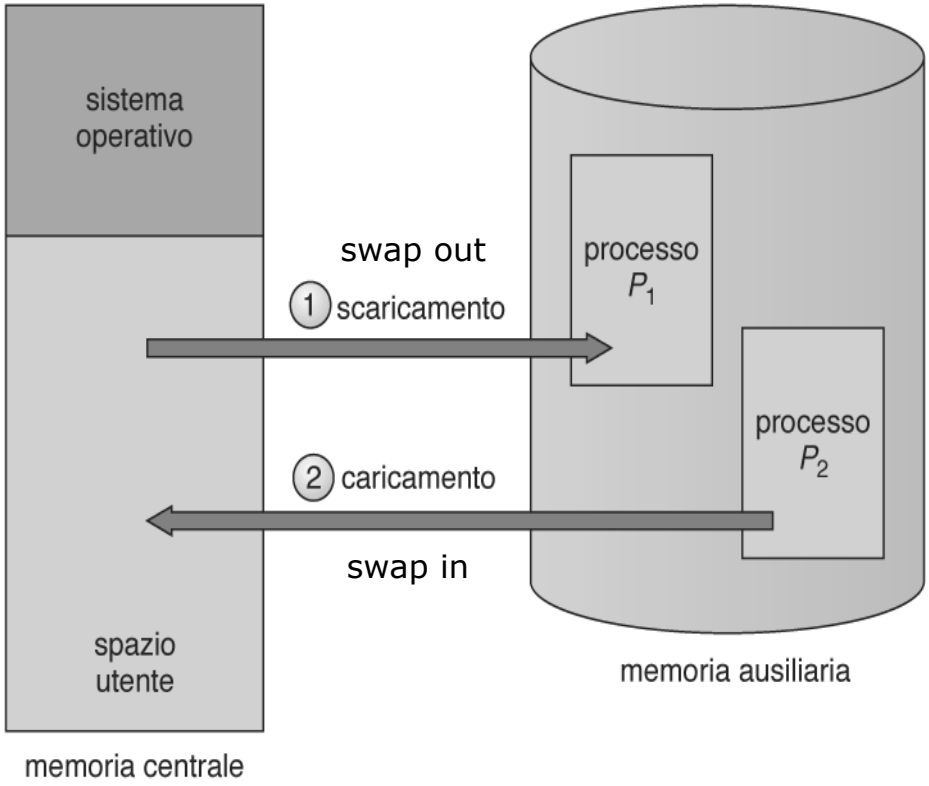
- Un processo può essere temporaneamente riportato (***swapped out***) su disco (*backing store*) e quindi riportato in memoria (***swapped in***) al momento di riprendere l'esecuzione.

- ***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, e Windows.

Swapping tra due processi



Memoria di swap = 1 MB

Velocità = 40 MBs

t = 1 MB/ 40 MBs

= 1/40 sec

= 25 msec

Tempo di swap = 2t = 50 msec

Memoria di swap out = 4 MB

Memoria di swap in = 2 MB

Velocità = 40 MBs

t1 = 4 MB/40 MBs = 4/40 sec = 100 msec

t2 = 2 MB/40 MBs = 2/40 sec = 50 msec

Tempo di swap = t1+t2 =150 msec

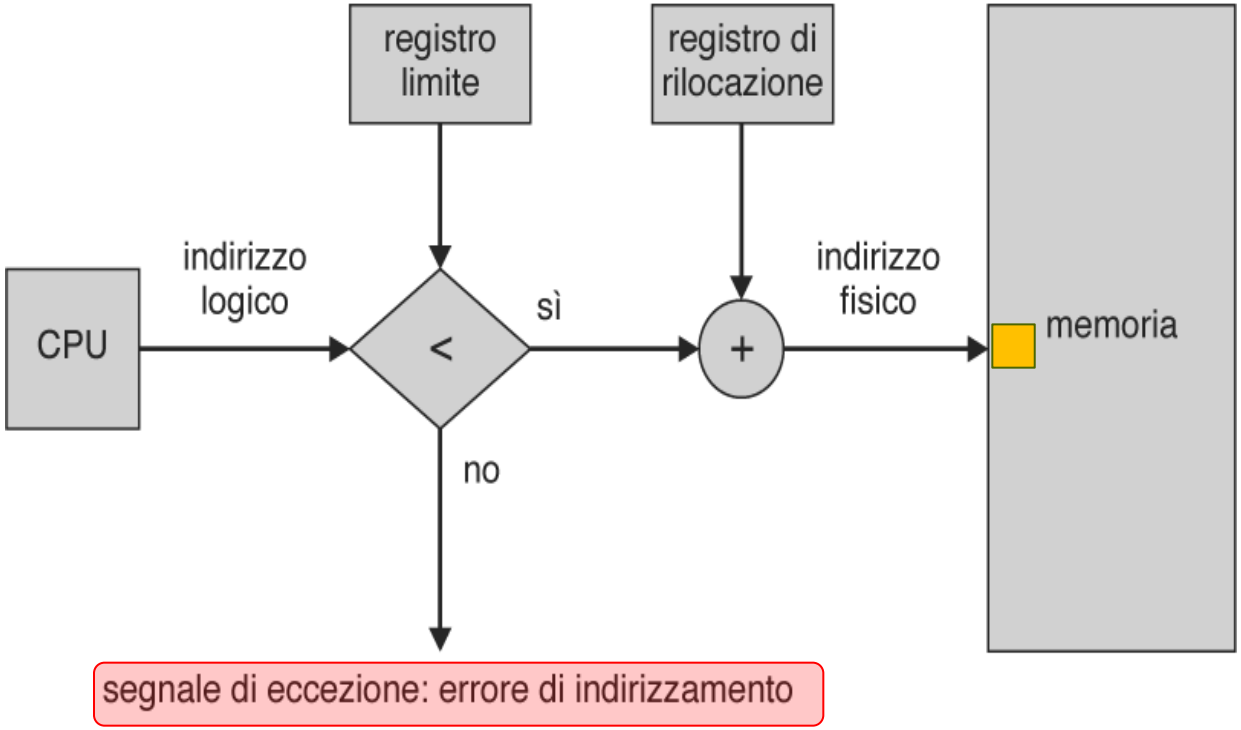
Allocazione contigua

- 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.

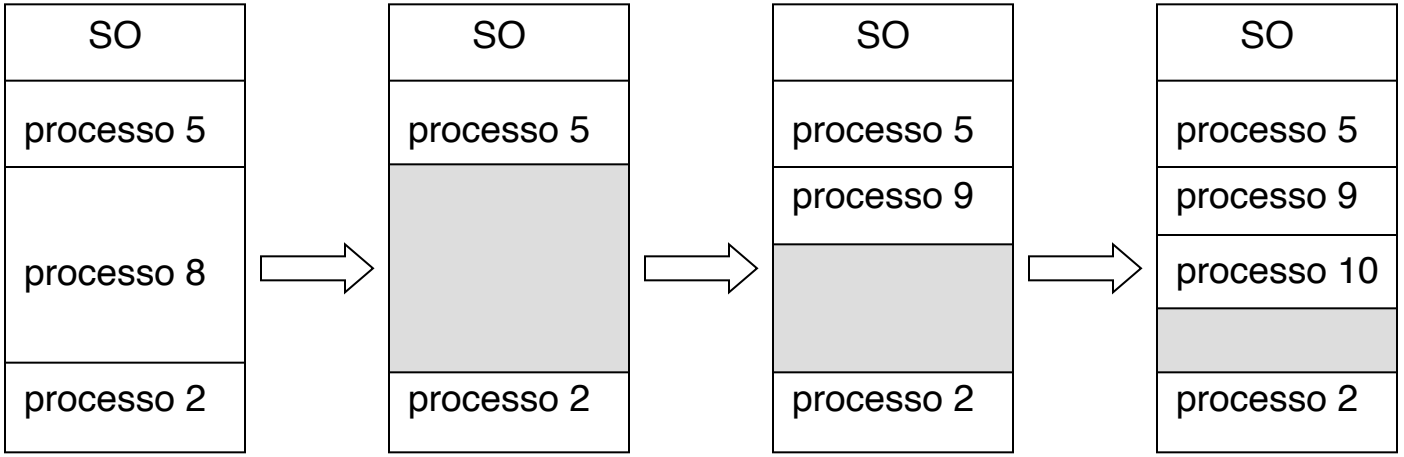
  - Registro di rilocazione contiene l'indirizzo fisico più piccolo e il registro limite contiene l'intervallo degli indirizzi logici: *ogni indirizzo logico deve essere minore del registro limite.*

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

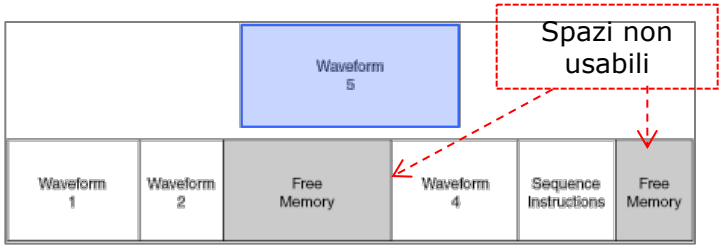
Come soddisfare una richiesta di dimensione N data una lista di buchi liberi ?

- **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*).

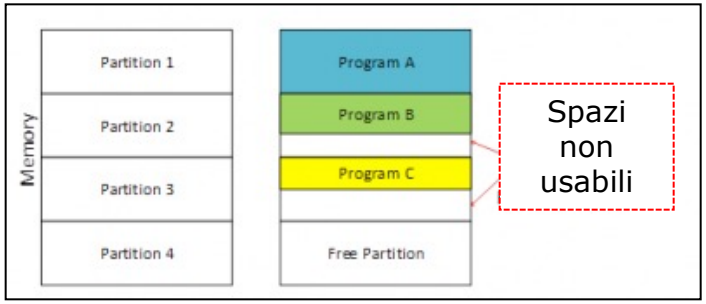
Risultati confermati da Simulazioni: First-fit e Best-fit sono migliori del Worst-fit in termini di velocità e uso di memoria.

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*** della memoria 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.

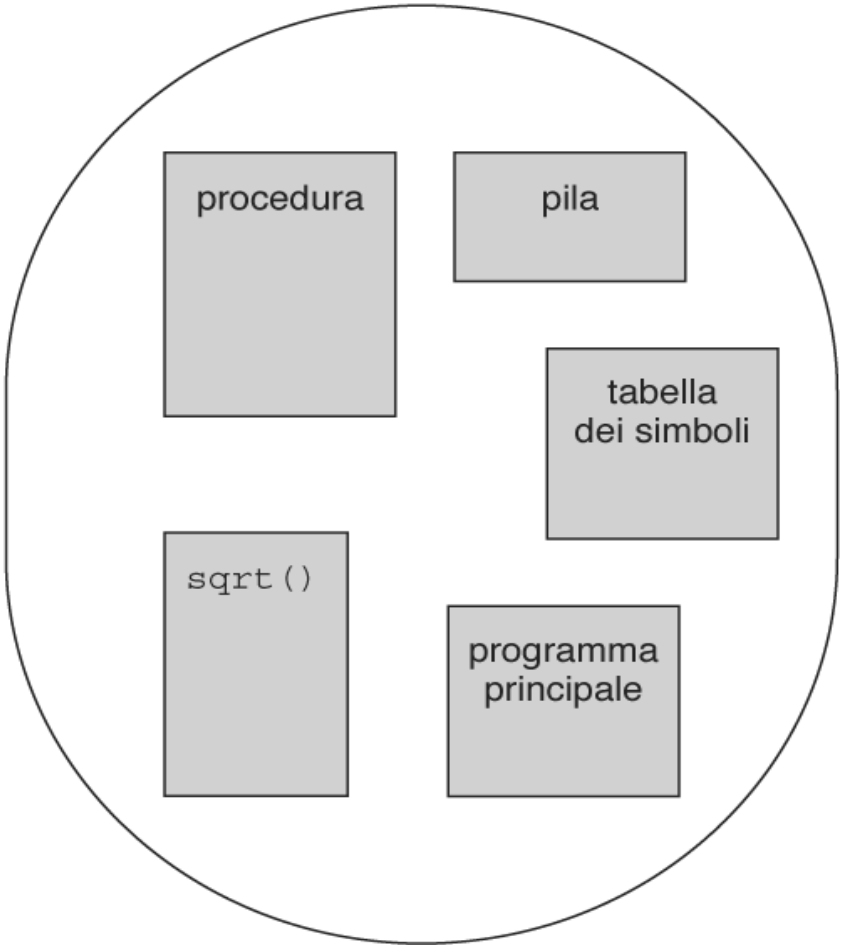
Allocazione non contigua: Segmentazione

- Lo spazio degli indirizzi logici di un processo può essere non contiguo; la memoria da allocare è utilizzata dove essa è disponibile.

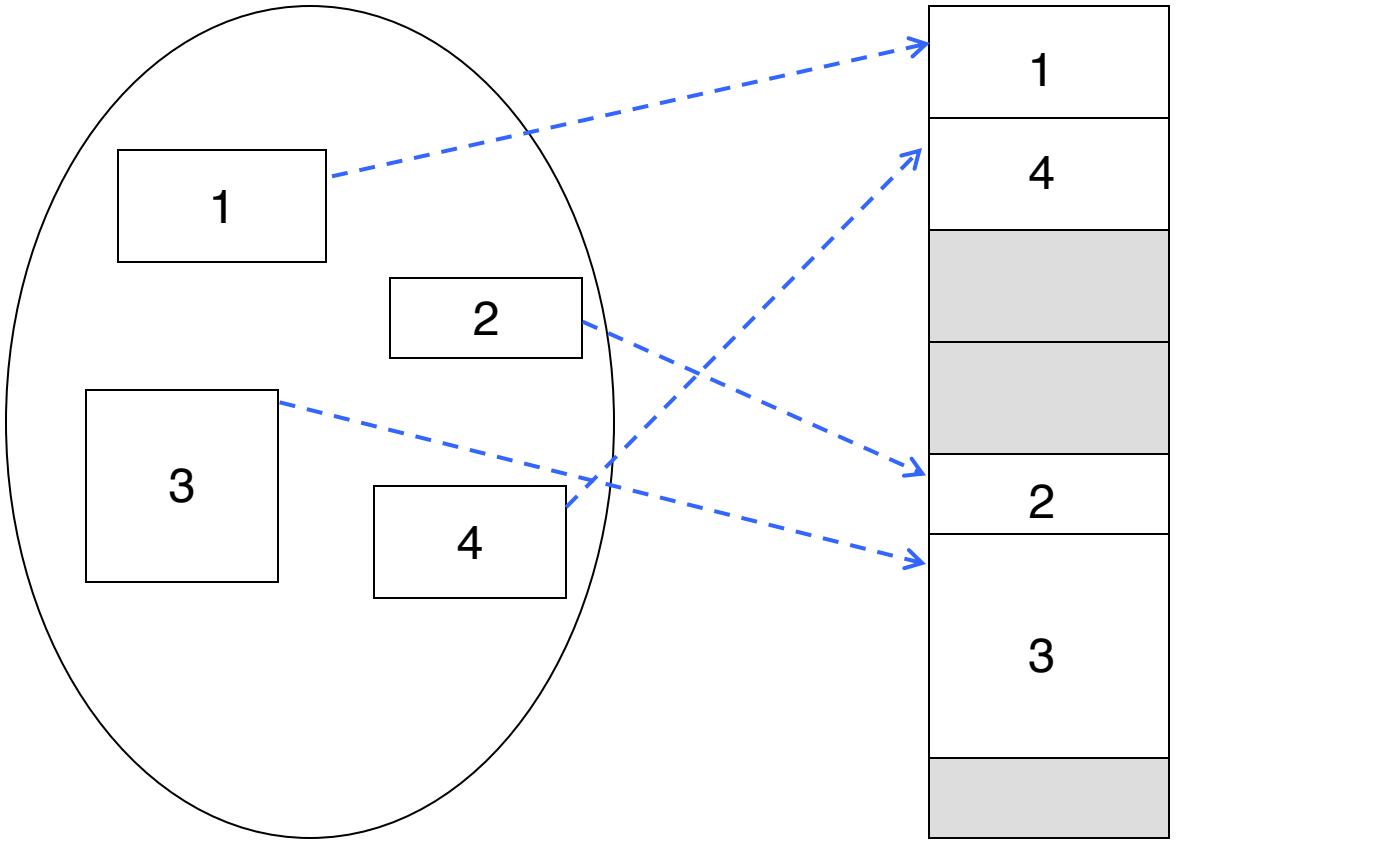
- La segmentazione è uno schema di gestione della memoria che corrisponde alla vista della memoria che in genere ha l'utente : **Un programma è una collezione di segmenti**.

- **Un segmento è una unità logica di memoria** come:
  - programma principale,
  - procedura e/o funzione,
  - metodo,
  - oggetto,
  - variabili locali, variabili globali,
  - stack,
  - tabella dei simboli,
  - array.

Programma dal punto di vista dell'utente



Vista logica della segmentazione



spazio del processo utente spazio della memoria fisica

Architettura di segmentazione

- Un indirizzo logico di una locazione consiste di una coppia di valori :

  <numero-segmento, offset>

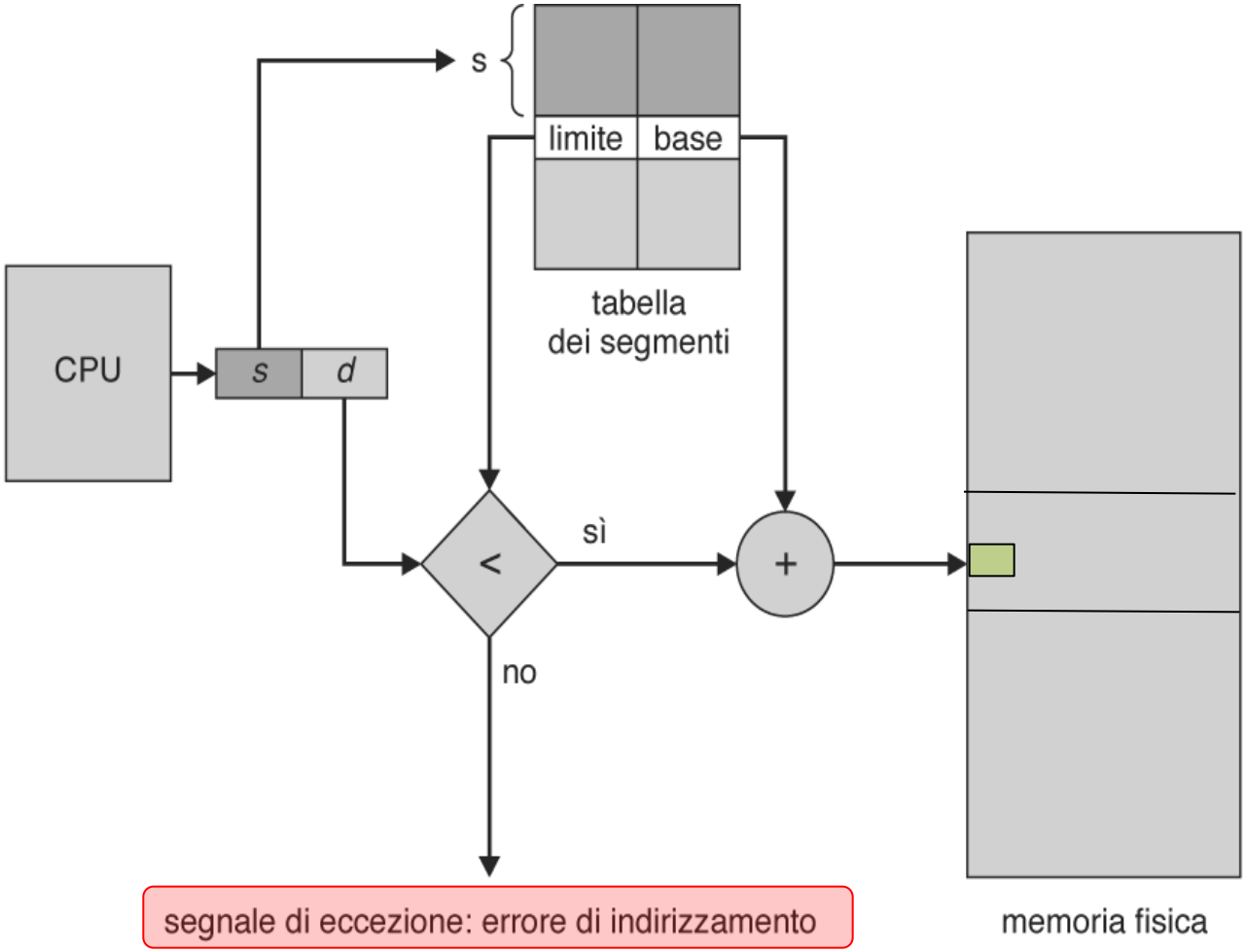
- *Tabella dei segmenti* – assegna indirizzi fisici bi-dimensionali.

- Ogni elemento della tabella ha:
  - *base* – indirizzo di partenza del segmento in memoria.
  - *limite* – lunghezza del segmento.

- *Segment-table base register (**STBR**)* punta alla locazione di memoria dove si trova la tabella.
- *Segment-table length register (**STLR**)* indica il numero dei segmenti appartenenti ad un programma;

  numero di segmento *s* è legale se *s* < *STLR.*

Architettura di segmentazione



Architettura di segmentazione

- **Rilocazione**
  - dinamica
  - tramite la tabella dei segmenti
- **Condivisione**
  - Segmenti condivisi
  - stesso numero di segmento
- **Allocazione**
  - first fit / best fit
  - frammentazione esterna

Architettura di segmentazione

- **Protezione**

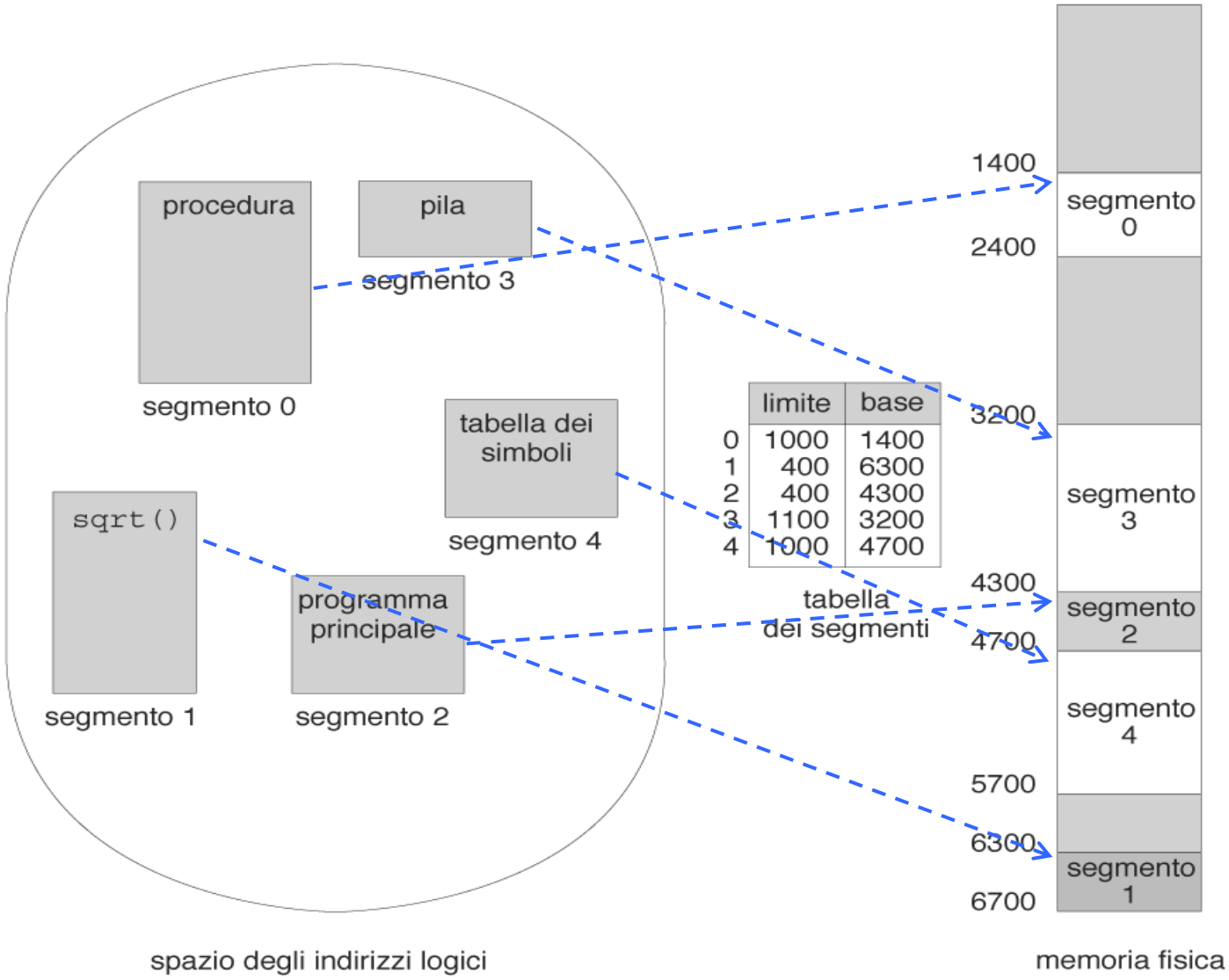
  Con ogni entry nella tabella dei segmenti associata:

  - bit di validazione = 0 $\Rightarrow$ segmento illegale
  - privilegi read/write/execute

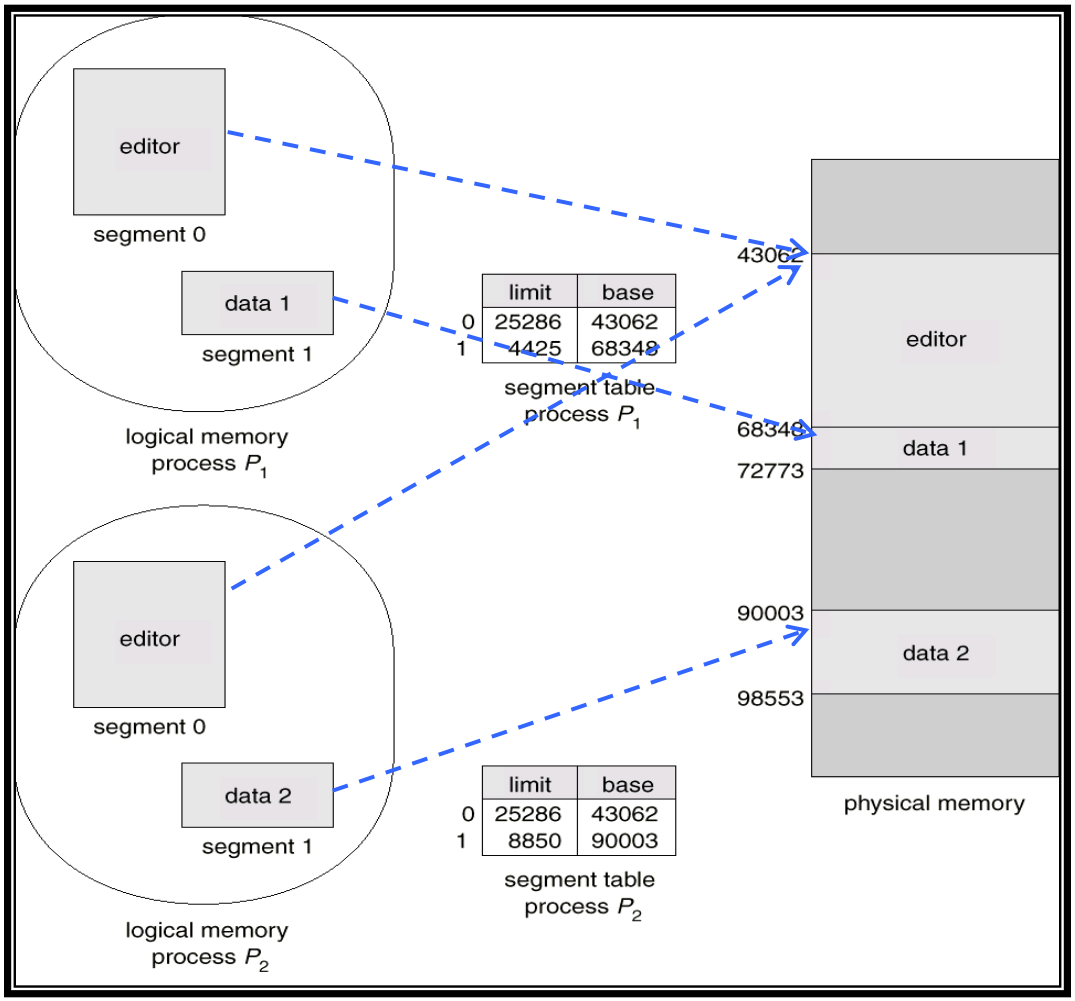
- Bit di protezione associati ai segmenti; condivisione di codice a livello di segmento.

- Poiché i segmenti variano in lunghezza, l'allocazione di memoria è un problema di **allocazione dinamica**.

Esempio di segmentazione



Condivisione di segmenti



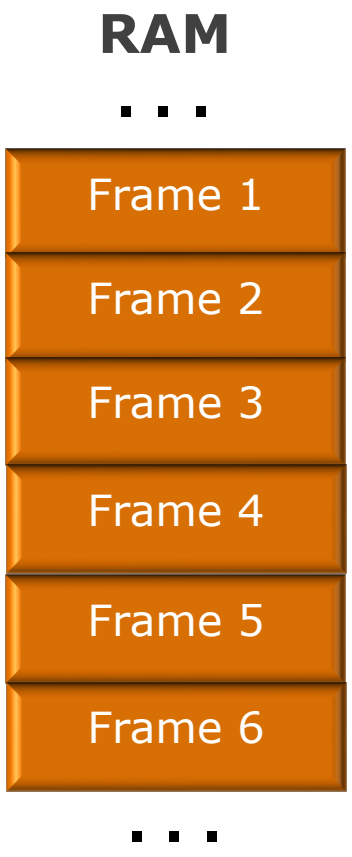
Allocazione non contigua: Paginazione

- Lo spazio degli indirizzi logici di un processo possono essere non contigui; la memoria da allocare è selezionata 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

- Il gestore della memoria tiene traccia di tutti frame liberi.
- Per eseguire un programma che richiede *n* pagine, bisogna trovare *n* frame liberi.
- Per ogni processo 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. Perché?

Schema di traduzione degli indirizzi

- Un indirizzo generato dalla CPU è composto da una coppia di valori ***(p,d)*** che contengono:

  - ***Numero di pagina (p)***

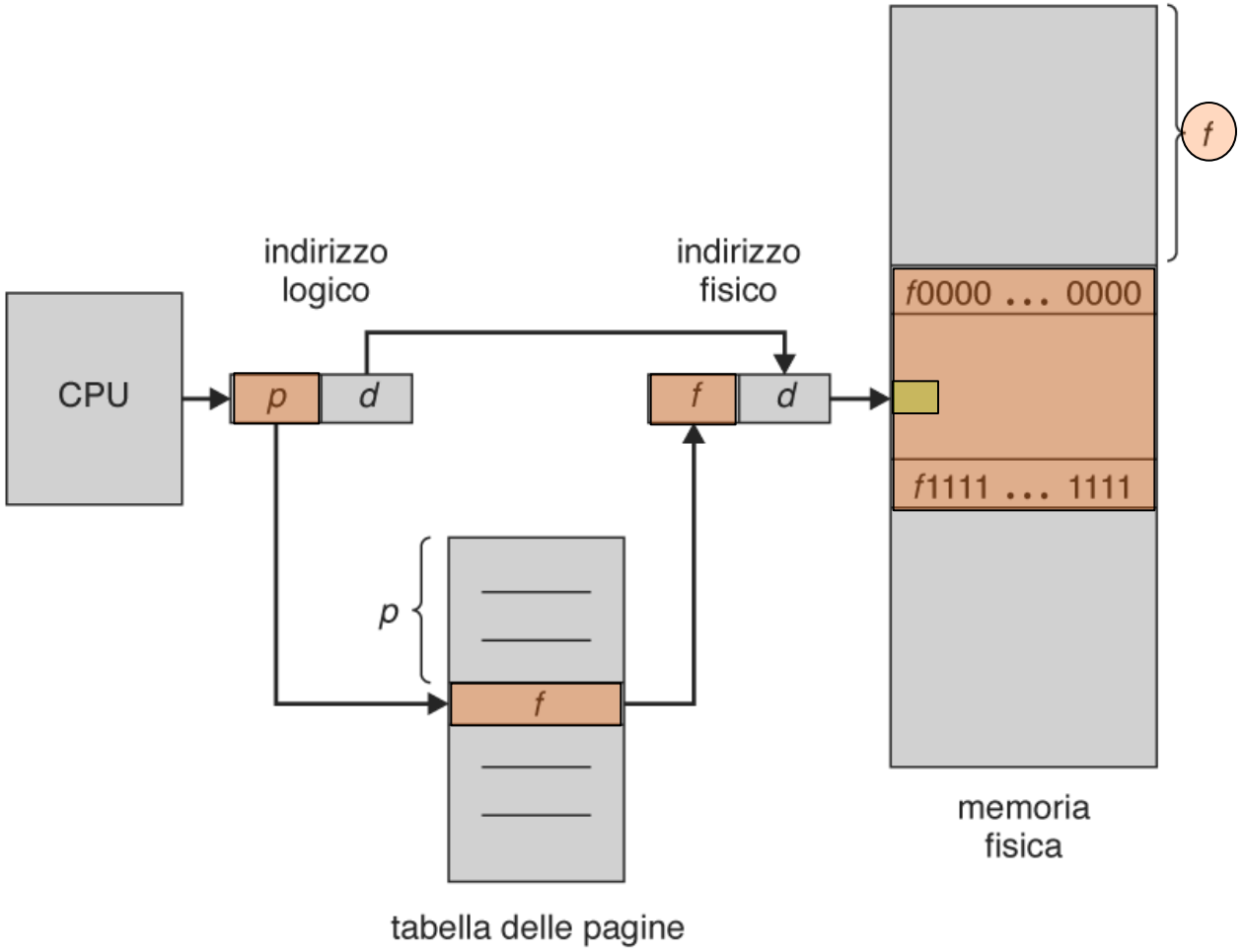
    usato come un indice nella tabella delle pagine che contiene l'indirizzo di base di ogni pagina in memoria fisica.

  - ***Offset di pagina (d)***

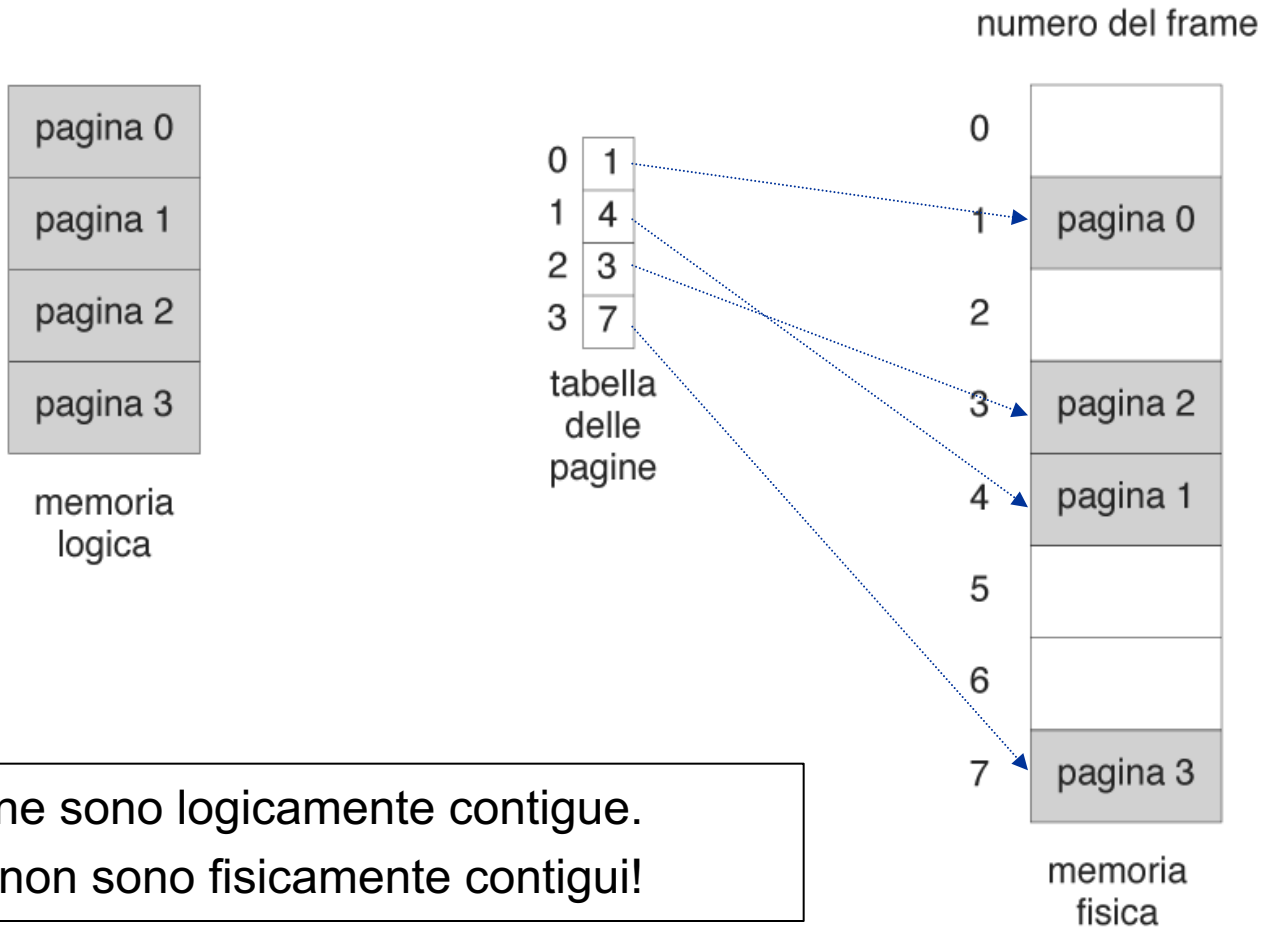
    usato insieme all'indirizzo base per definire l'indirizzo fisico di memoria da inviare alla unità di memoria.

Indirizzo = **[p, d]**

Architettura di paginazione



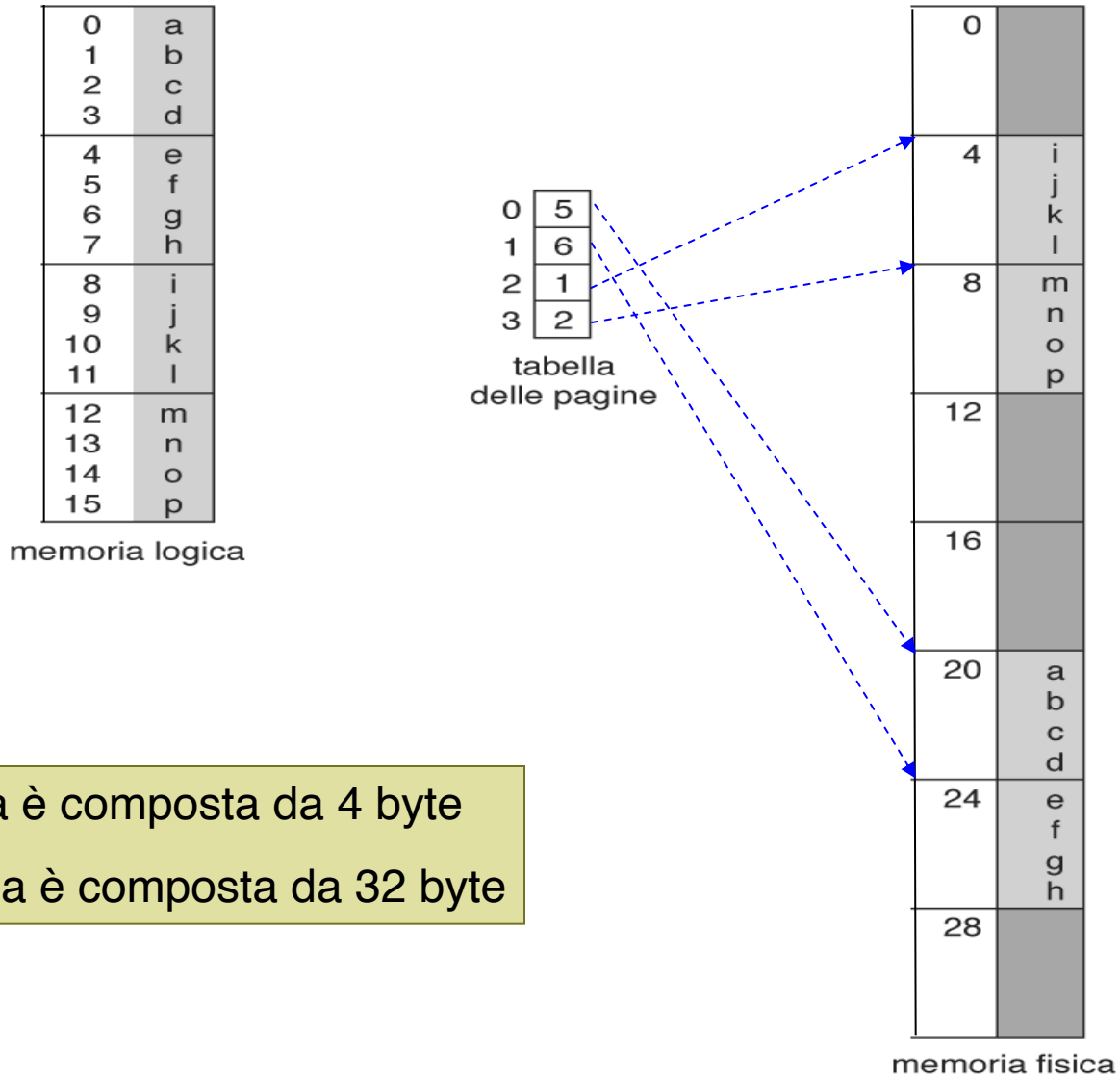
Esempio di paginazione



Le pagine sono logicamente contigue.

I frame non sono fisicamente contigui!

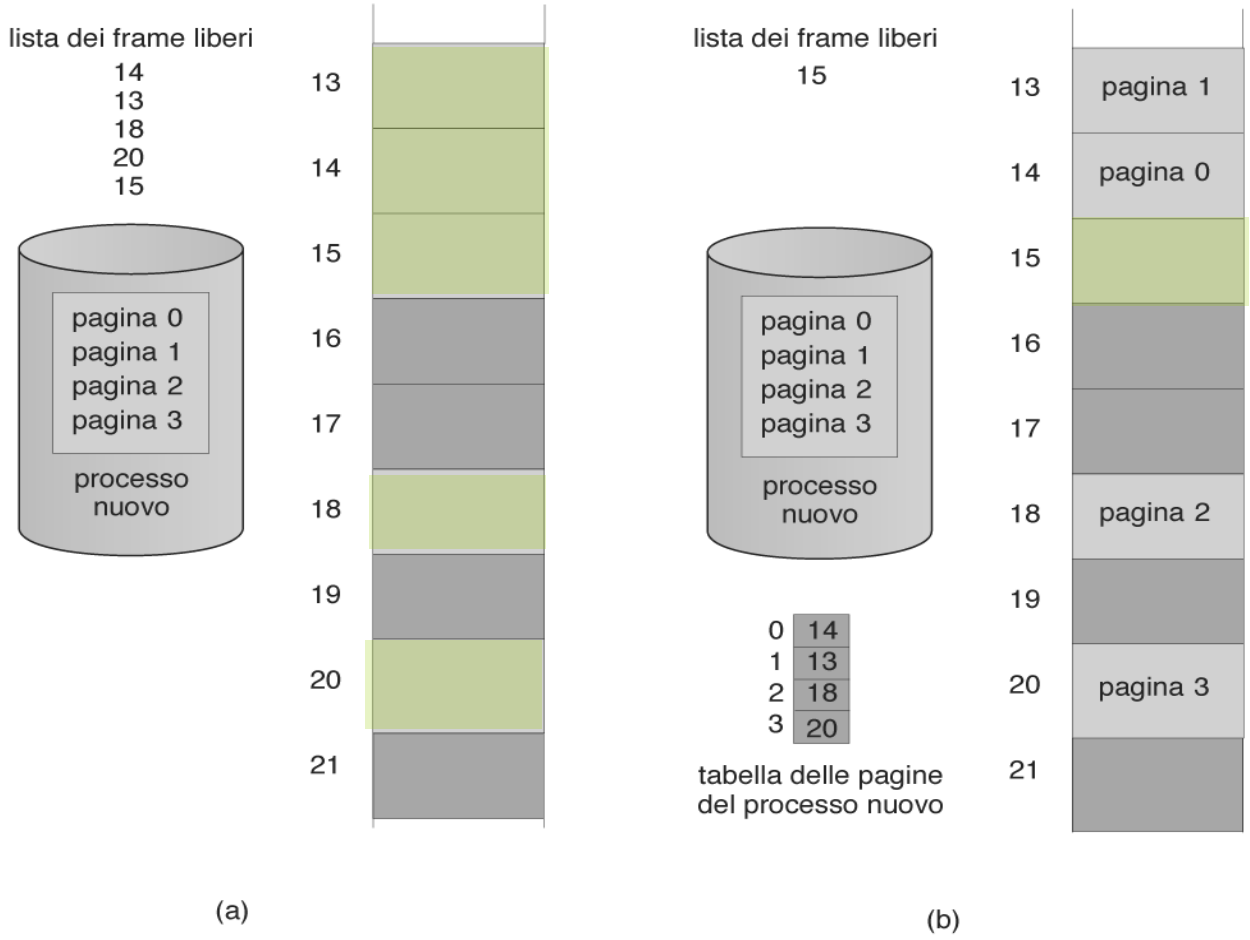
Esempio di paginazione



Ogni pagina è composta da 4 byte

e la memoria è composta da 32 byte

Frame liberi



Prima dell'allocazione

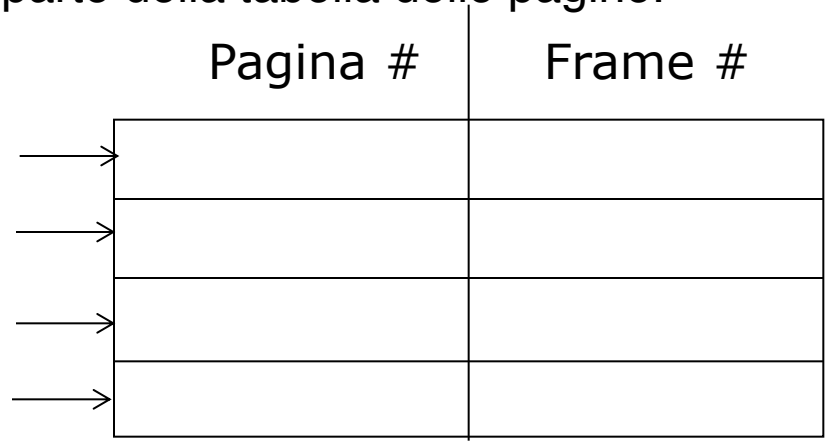
Dopo l'allocazione

Implementazione della tabella delle pagine

- La tabella delle pagine è memorizzata in memoria centrale.
- Per ogni tabella delle pagine:
  - 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.***

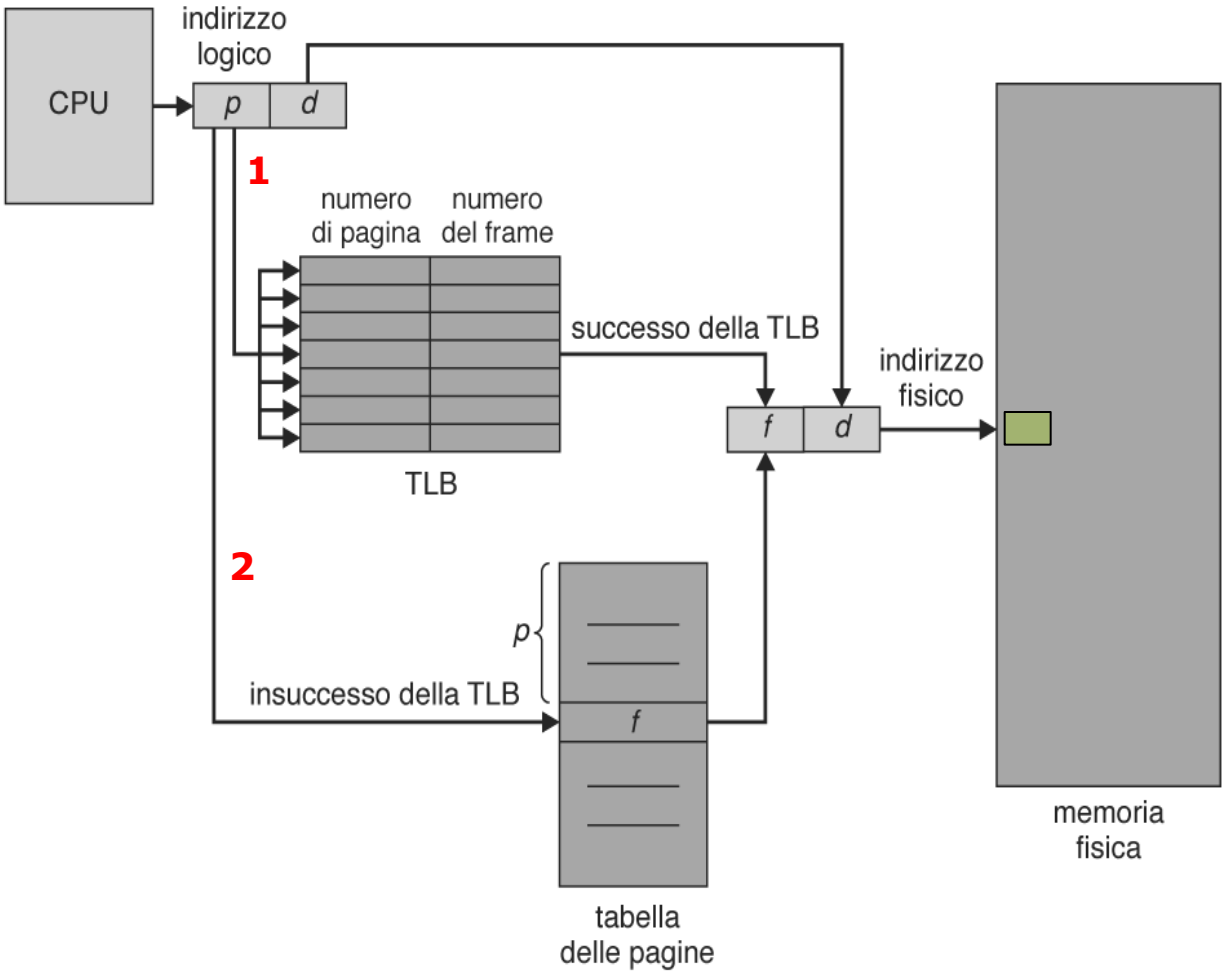
Memoria associativa

- Memoria associativa – permette la ricerca parallela e permette di implementare una **TLB** (translation look-aside buffer) che contiene solo una parte della tabella delle pagine.



- Traduzione di un indirizzo:
  - Se l'indirizzo è in un registro associativo della TLB, si ottiene direttamente il numero di frame.
  - Altrimenti il numero di frame si ottiene dalla tabella delle pagine e occorre fare un accesso alla tabella.

Architettura di paginazione con TLB



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 memoria (*tm*) – tempo di accesso alla memoria (non associativa)
- Tempo di accesso effettivo

TAE = hr x (la+tm) + (1-hr) x (la+2tm)

(se hr=90%) = 0.9 x (20+100) + 0.1 x (20+200)

= 108 + 22 = 130 nsec

TAE = hr x (la+tm) + (1-hr) x (la+2tm)

(se hr=70%) = 0.7 x (20+100) + 0.3 x (20+200)

= 84 + 66 = 150 nsec

Protezione della memoria

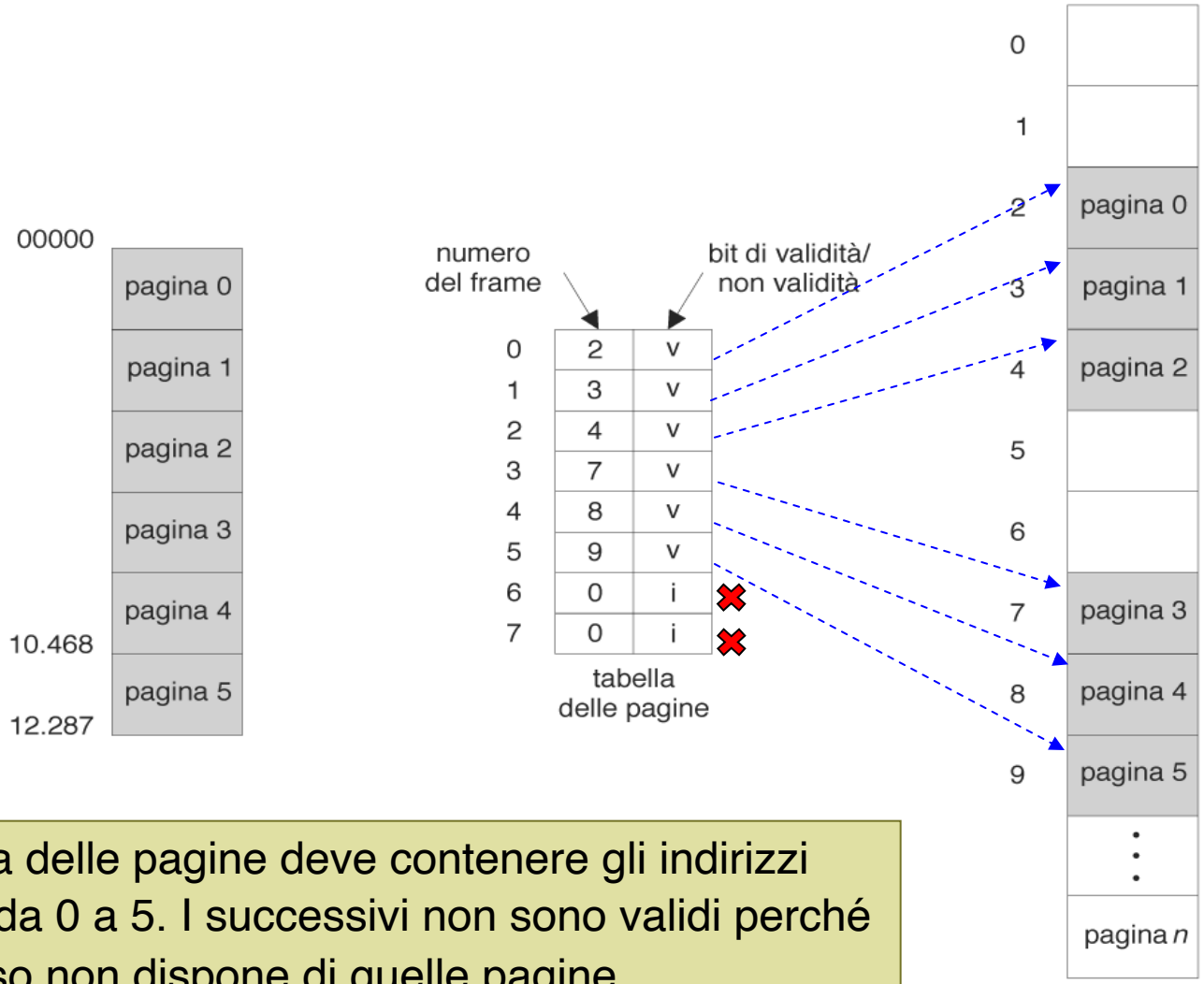
- La protezione della memoria è implementata associando dei *bit di protezione* ad ogni frame. Permetteno di indicare i possibili accessi al frame (lettura, scrittura, esecuzione).

- In alcuni casi si usa anche un ***bit di validità***.

- Il ***bit di validità*** associato ad ogni entry nella tabella delle pagine ha i valori:
  - "**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. Un accesso genera un'eccezione.

Bit di validità (v) o non validità (i)



La tabella delle pagine deve contenere gli indirizzi ai frame da 0 a 5. I successivi non sono validi perché il processo non dispone di quelle pagine

Struttura della tabella

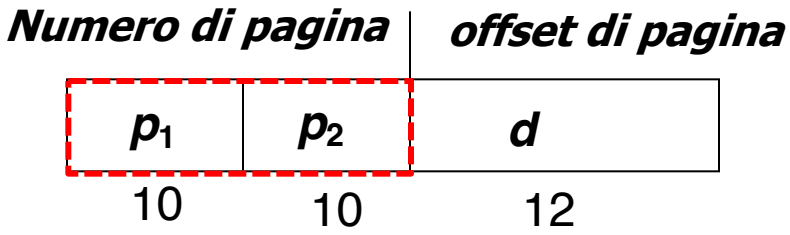
Per gestire tabelle delle pagine molto grandi si fa uso di due tecniche principali:

1. **Paginazione gerarchica**
   - Si divide lo spazio degli indirizzi logici in più tabelle
   - Una tecnica semplice è basata sull'uso una tabella a due livelli.
   - A volte, si usano anche tabelle a tre livelli.

2. **Tabella delle pagine invertita**
   - un'unica struttura dati (tabella) globale che contiene un elemento per ogni frame.

Esempio di tabella a due livelli

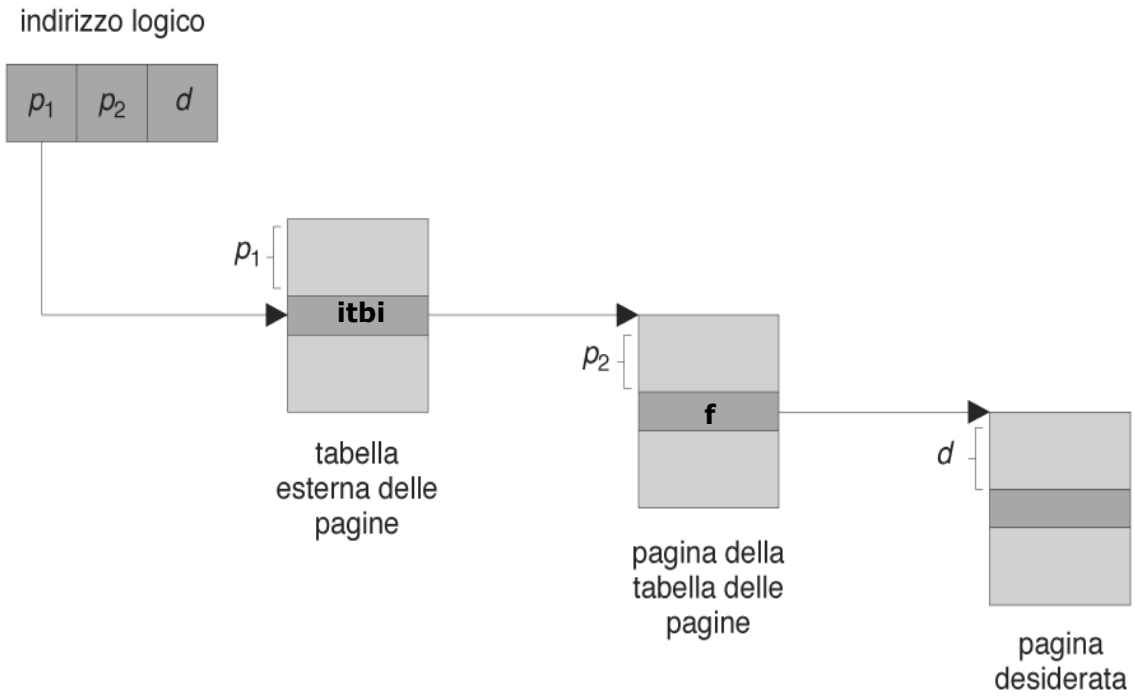
- Un indirizzo logico (su una macchina a 32 bit con pagine di 4KB) è divisa in:
  - un numero di pagina di 20 bit.
  - un offset di pagina di 12 bit.
- Poiché la tabella delle pagine è composta a sua volta da più tabelle, il numero di pagina è diviso da:
    - un numero di pagina di 10 bit.
    - un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



Itbi = Indirizzo iniziale tabella interna

Schema di tabella delle pagine a due livelli

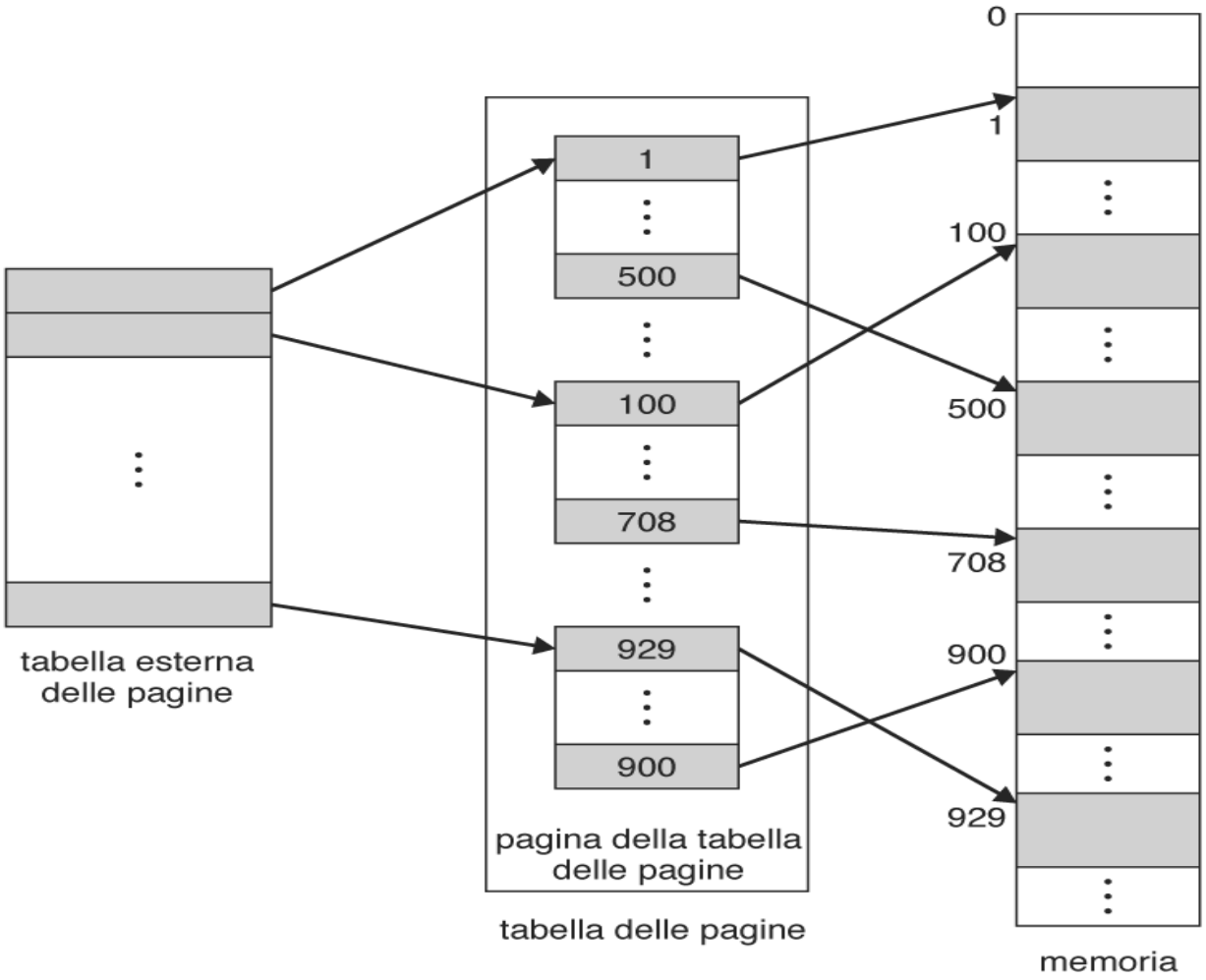
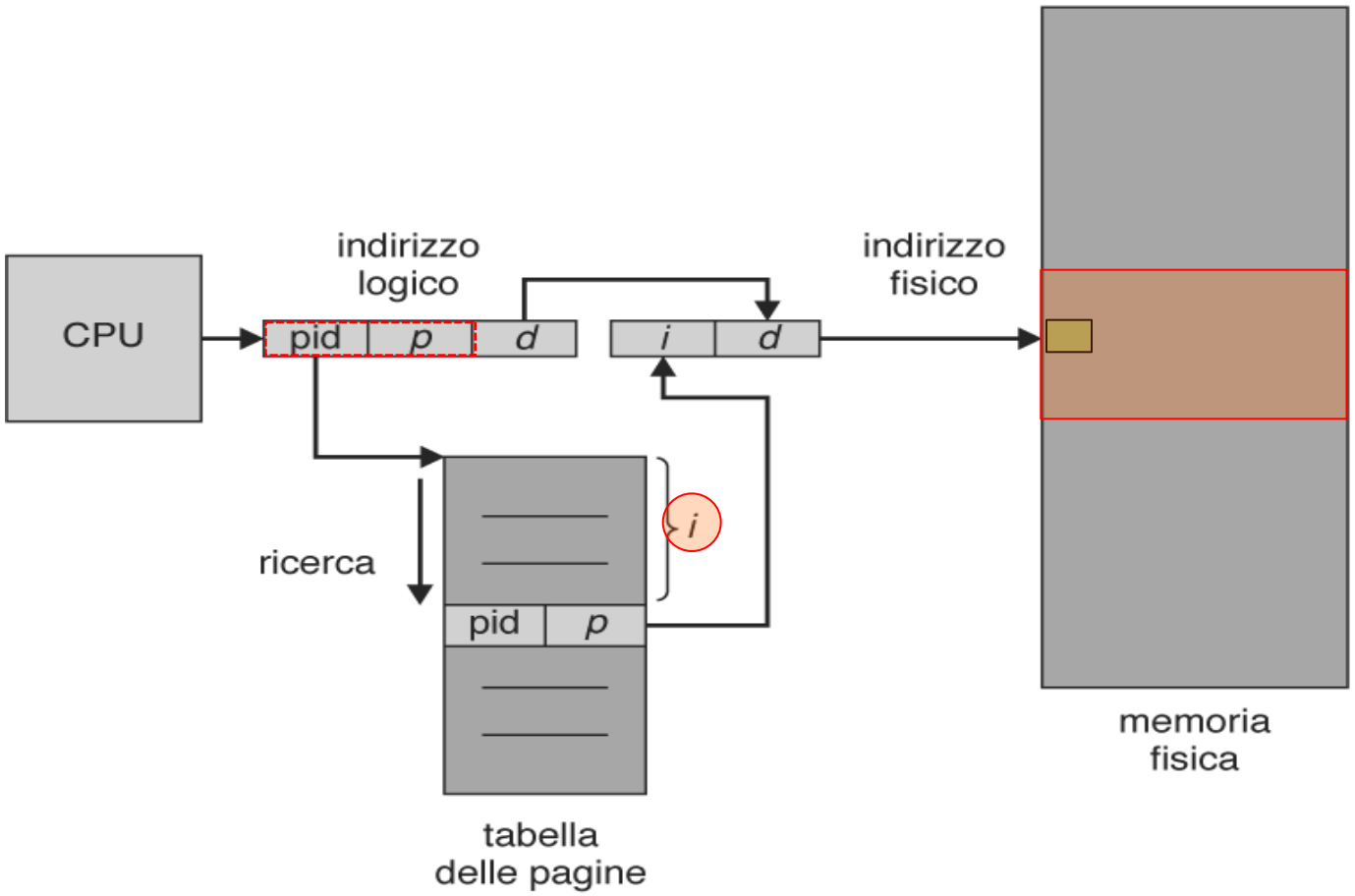


Tabella delle pagine invertita

- Usando questo schema la tabella delle pagine contiene **una sola entry per ogni pagina reale in memoria**.
- Ogni entry di pagina 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



Si ricerca nella tabella l'elemento che contiene la coppia (***pid,p***)

L'indice ***i*** dell'elemento trovato rappresenta il numero del frame allocato alla pagina logica ***p***

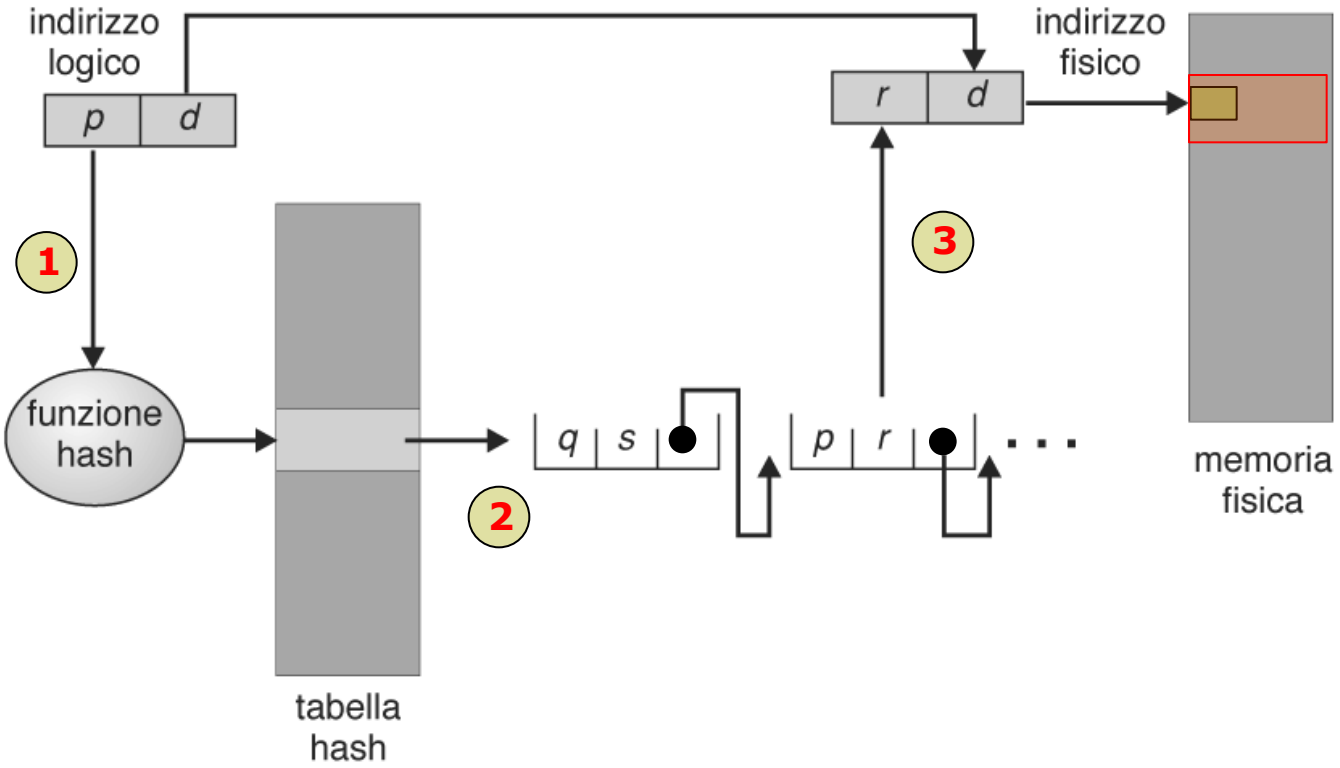
Tabella delle pagine di tipo hash

- Comune nelle architetture con spazio di indirizzi > 32 bit. Si può usare per velocizzare l'accesso alla tabella delle pagine invertite.

- Una funzione *hash* è applicata al numero di pagina virtuale, identificando una riga della tabella
  - La riga contiene una lista di elementi che generano lo stesso valore di hash

- Il numero di pagina virtuale è confrontato con gli elementi nella lista per trovare una corrispondenza
  - Se si trova una corrispondenza, viene estratto il numero di frame corrispondente

Tabella delle pagine di tipo hash



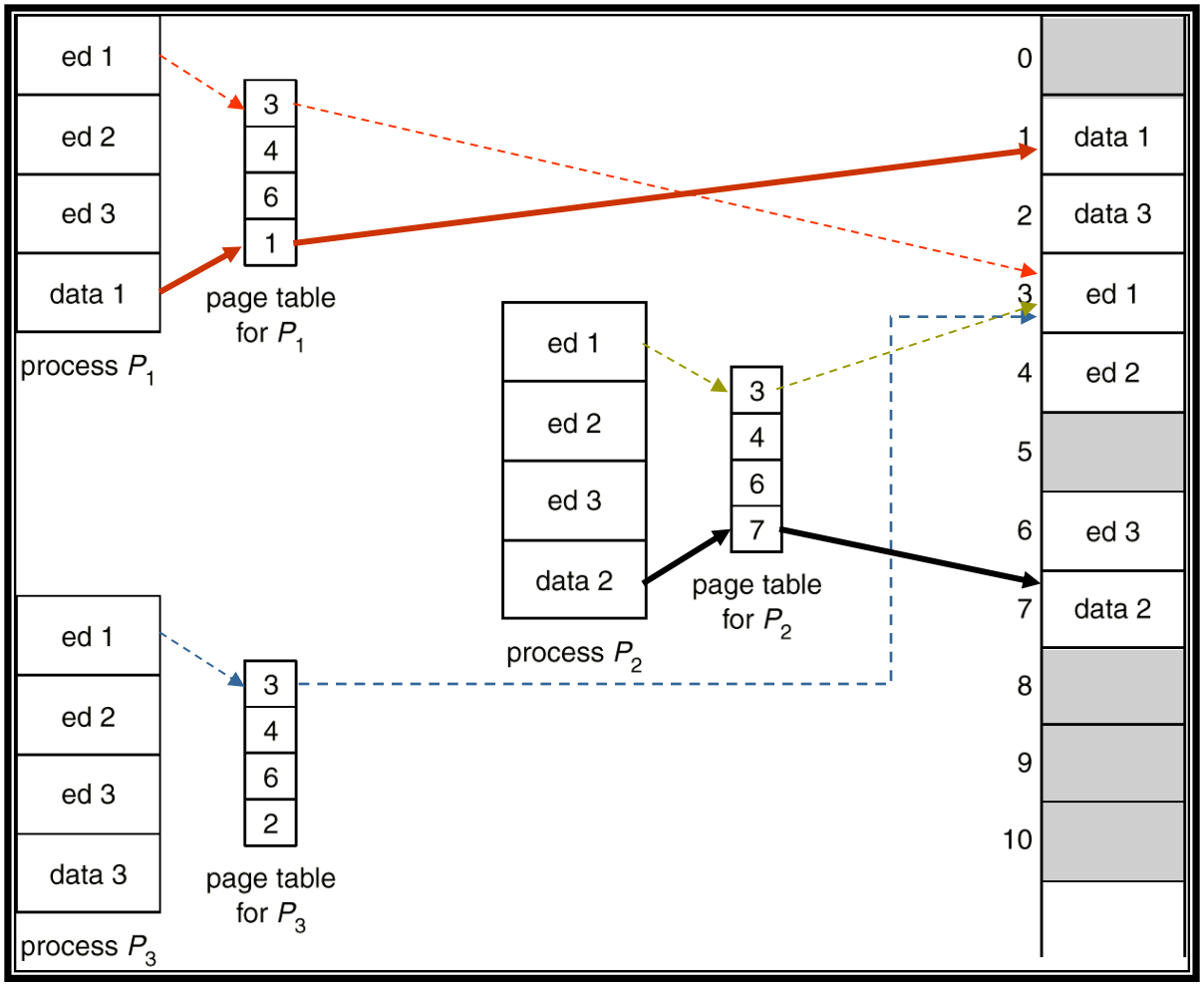
Pagine condivise

- Usando la paginazione si può condividere codice comune.

- Codice condiviso
  - Una singola copia di codice a sola lettura (rientrante) 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



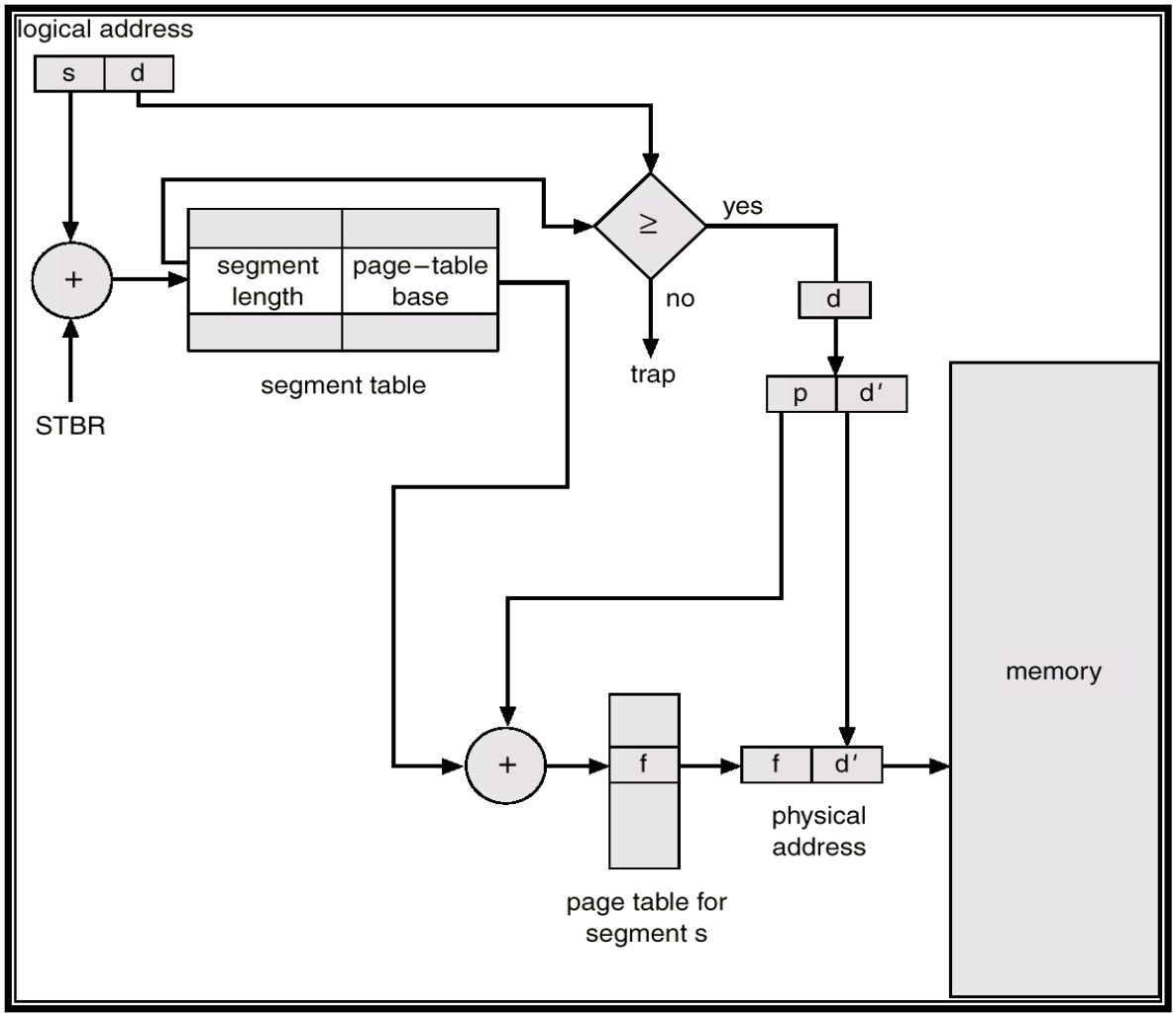
Segmentazione con paginazione

- 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.

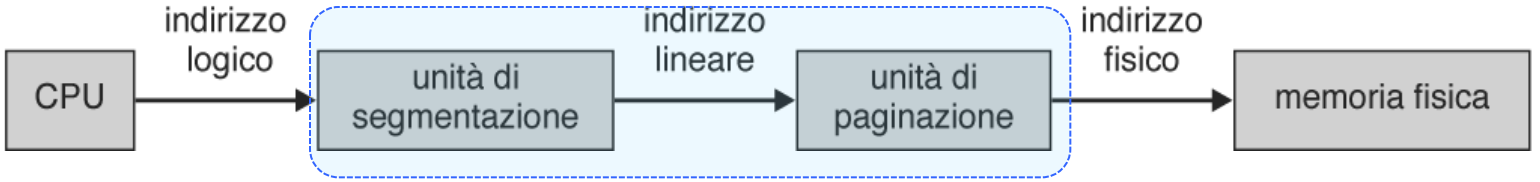
Segmentazione con paginazione in MULTICS



Esempio: Intel Pentium

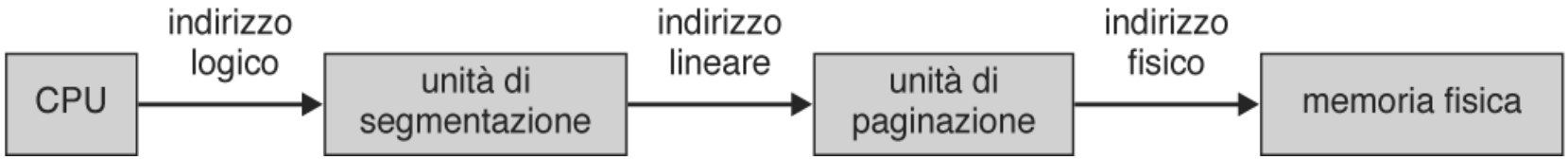
- Il Pentium supporta sia segmentazione pura (con segmenti grandi fino a 4 GB), sia segmentazione con paginazione (le pagine sono di 4KB o di 4MB).

- La CPU genera indirizzi logici
  - Passati all'unità di segmentazione
    - Che produce indirizzi lineari
  - Gli indirizzi lineari sono forniti all'unità di paginazione
    - La quale genera indirizzi fisici in memoria centrale



- Le unità di segmentazione e paginazione formano l'equivalente della **MMU**.

Traduzione degli indirizzi logici in indirizzi fisici nell'Intel Pentium



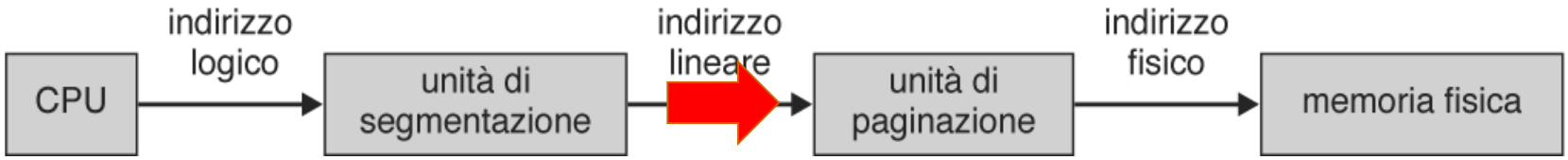
- ✓ Vi sono due partizioni per lo spazio degli indirizzi di un processo (max 16K segmenti – 8K per partizione): la partizione dei segmenti privati e la partizione dei segmenti condivisi.
- ✓ Due tabelle dei descrittori – una per partizione. Tabella locale e tabella globale.

✓ **Indirizzo logico = (selettore, offset)**

- ***Selettore*** (16 bit)

- ***Offset*** (32 bit)

Traduzione degli indirizzi logici in indirizzi fisici nell'Intel Pentium

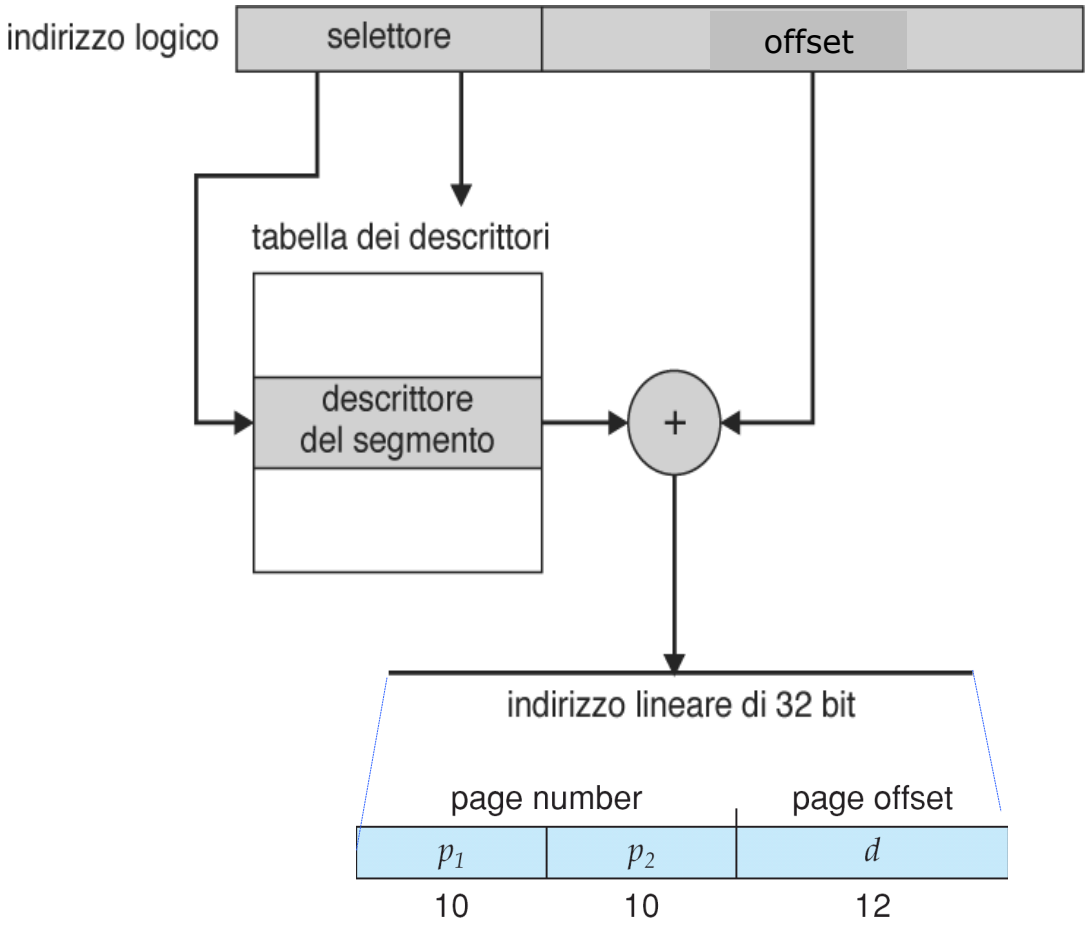


- **Indirizzo logico = (selettore, offset)**
  - ***Selettore*** (16 bit)

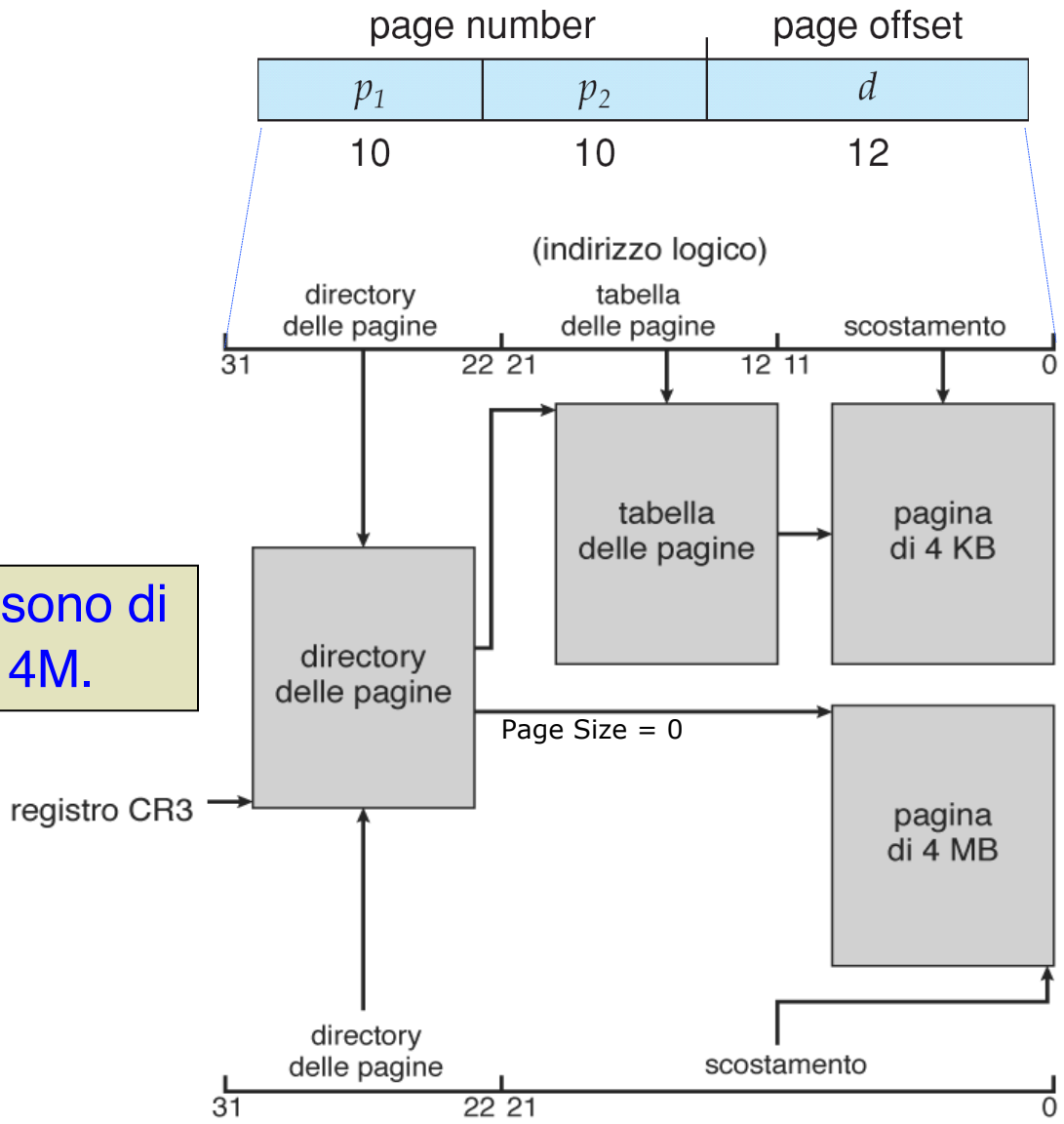
  - ***Offset*** (32 bit)

- **Indirizzo lineare** (32 bit) (costruito con il valore dell'offset + base del segmento)

Segmentazione in Intel Pentium



Paginazione nell'architettura Intel Pentium



Le pagine sono di 4K o di 4M.