Capitolo 9: Memoria centrale

- Concetti generali.
- Swapping
- Allocazione contigua di memoria.
- Paginazione.
- Segmentazione.
- Segmentazione con paginazione.

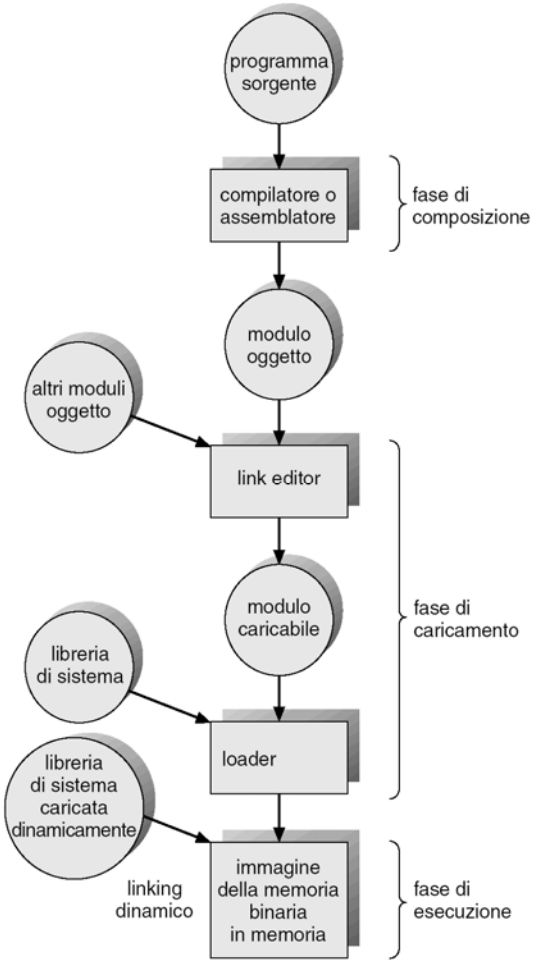
Concetti generali

- Per essere eseguito un programma deve essere portato dalla memoria di massa in memoria centrale ed essere attivato come processo.
- La memoria è sostanzialmente un vettore di word ciascuna delle quali è accessibile singolarmente (dotata di un indirizzo specifico).
- Ciclo istruzione-esecuzione:
  - Prelievo di una istruzione dalla memoria centrale in base al valore del program counter;
  - Decodifica dell'istruzione
  - Prelievo di operandi dalla memoria ed esecuzione
  - Memorizzazione dei risultati in memoria
- Dal punto di vista della memoria, in ingresso/uscita abbiamo solo un flusso di indirizzi.
- *Coda di entrata* – processi su disco che sono in attesa di essere caricati in memoria centrale per l'esecuzione.
- I programmi utente passano attraverso più stadi prima di essere eseguiti (compiling, linking, loading).

Collegamento degli indirizzi

Il collegamento delle istruzioni e dei dati agli indirizzi di memoria viene effettuato mediante la seguente successione di passi:

- **Fase di compilazione**: se in fase di compilazione si conosce la locazione del processo in memoria, allora si può generare un *codice assoluto*; se la locazione di partenza cambia bisogna ricompilare il codice.
- **Fase di caricamento**: se al momento della compilazione non è nota la locazione del processo in memoria, il compilatore deve generare un *codice rilocabile*.
  - Il linking finale è rinviato fino all'istante di caricamento.
  - Modificandosi l'indirizzo di partenza, c'è la necessita di modificare il solo codice utente per incorporare il valore cambiato.
- **Fase di esecuzione**: se il processo può essere spostato, durante l'esecuzione, da un segmento di memoria a un altro, allora il collegamento deve essere ritardato fino al momento dell'esecuzione. Necessita di un hardware specifico.

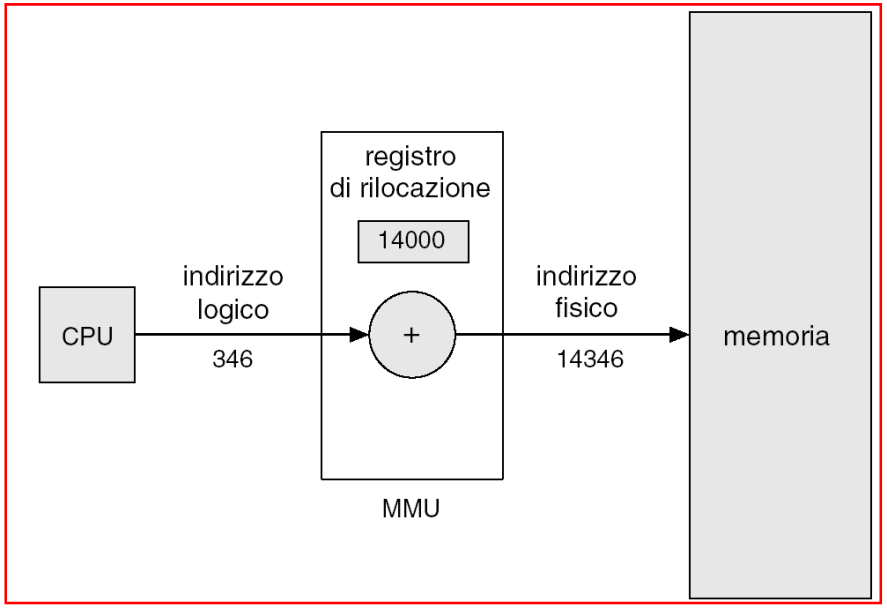


Indirizzamento logico e fisico

- *Spazio di indirizzamento logico* (insieme di tutti gli indirizzi logici generati da un programma).
  - *Indirizzo logico* – indirizzo generato dalla CPU; anche definito come *indirizzo virtuale*.
- *Spazio di indirizzamento fisico* (insieme degli indirizzi fisici corrispondenti a quelli logici).
  - *Indirizzo fisico* – indirizzo visto dalla memoria (caricato nel registro degli indirizzi della memoria).
- Metodi di collegamento degli indirizzi:
  - fase di compilazione e di caricamento:
    - indirizzi logici e fisici identici
  - fase di esecuzione:
    - indirizzi logici e fisici diversi.

Memory Management Unit (MMU)

- Dispositivo hardware che realizza la trasformazione dagli indirizzi virtuali a quelli fisici in fase di esecuzione.
- Nello schema di una MMU, il valore nel registro di rilocazione è aggiunto ad ogni indirizzo generato da un processo nel momento in cui è trasmesso alla memoria.
- Il programma utente lavora con gli indirizzi logici; non avrà mai percezione degli indirizzi fisici reali.



Caricamento dinamico

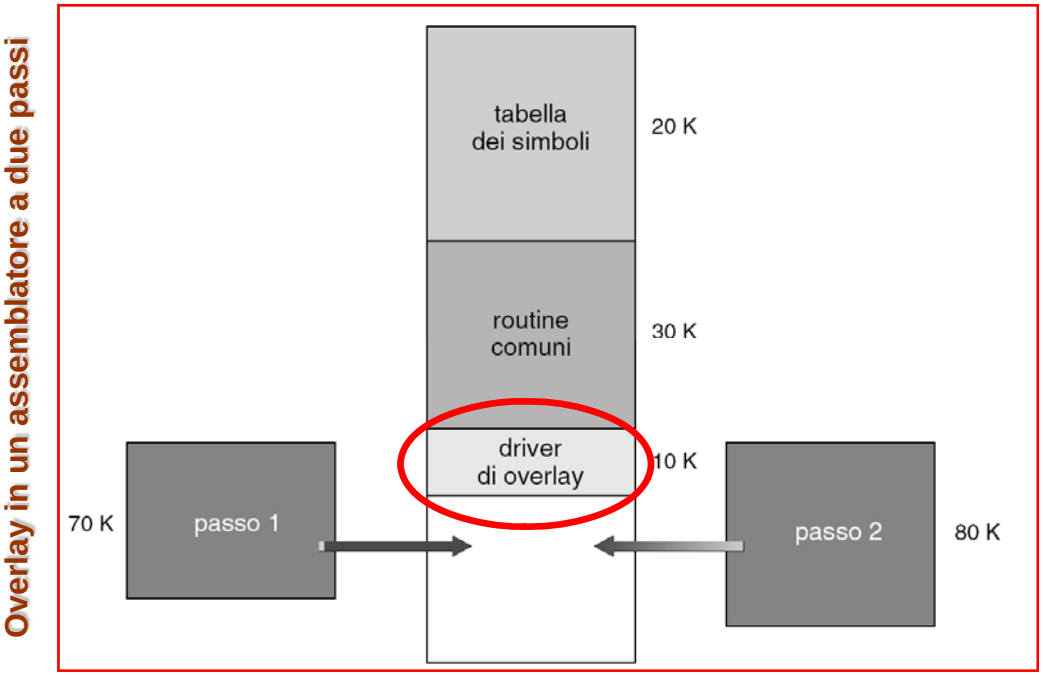
- Affinché un processo possa essere eseguito, l'intero suo codice ed i suoi dati devono essere in memoria.
  - Ne consegue che la dimensione di un processo è vincolata dalla dimensione della memoria fisica.
- Per ottenere un migliore utilizzo dello spazio in memoria si adopera il dynamic loading:
  - una procedura non è caricata finché non viene richiamata;
  - una procedura inutilizzata non viene mai caricata;
  - tutte le procedure trovano posto su disco in un formato di caricamento rilocabile.
- Il programma principale viene caricato in memoria ed eseguito. Quando deve essere richiamata una procedura non in memoria si richiama il loader per effettuare il caricamento rilocabile.
- Utile quando sono necessarie grandi quantità di codice per gestire situazioni che si presentano raramente.
- Non richiede un supporto speciale da parte del sistema operativo.

Linking dinamico

- Il collegamento è posposto fino alla fase di esecuzione.
- Senza questa funzione, ogni programma dovrebbe avere una copia delle librerie delle procedure del linguaggio.
  - Spreco di spazio sulla memoria di massa e su quella centrale.
- Una piccola parte di codice eseguibile, detta *immagine* o *stub*, indica come individuare la procedura di libreria desiderata (se residente in memoria) o come caricarla (se non residente).
- L'immagine rimpiazza sè stessa con l'indirizzo della procedura e la esegue.
  - Vantaggio: è sufficiente sostituire una libreria su disco perchè tutti I programmi si riferiscano automaticamente alla versione più recente.
- Il sistema operativo deve controllare se la procedura necessaria è nello spazio di memoria di un altro processo.
  - È l'unica entità in grado di abilitare l'accesso ad un'area sottoposta a meccanismi di protezione.
  - È l'unica entità in grado di permettere che processi multipli accedano alle stesse locazioni di memoria.
- Il collegamento dinamico è particolarmente utile con le librerie di sistema (DLL).

Overlay

- Mantiene in memoria solo le istruzioni e i dati che sono necessari in un dato istante.
- Necessario quando un processo è più grande della quantità di memoria ad esso allocata.
- Gli overlay non richiedono alcun supporto speciale da parte del sistema operativo e possono essere implementati dall'utente. La progettazione della struttura del programma di overlay è complessa.



Swapping (1/3)

- Un processo può essere temporeamente scambiato (swapped) spostandolo dalla memoria centrale ad una memoria temporanea, e poi riportato in memoria centrale per continuarne l'esecuzione.
  - Adoperato tipicamente con schedulazioni round-robin.
- Memoria temporanea (backing storage) – disco veloce abbastanza capiente da accogliere le copie di le immagini della memoria centrale per tutti gli utenti.
  - A tali immagini si deve fornire accesso diretto.
- Roll-in/roll-out – variante dello swapping usata per algoritmi di schedulazione basati sulla priorità;
  - un processo a bassa priorità è scambiato in modo che un processo ad alta priorità possa essere caricato ed eseguito.
- Normalmente un processo scambiato viene ricaricato in memoria sempre nella medesima posizione.
  - Se il linking viene fatto all'atto dell'assemblaggio o del loading, lo spostamento in una posizione differente è assai complesso.
  - Lo swap in uno spazio di memoria differente è invece praticabile con il loading dinamico (gli indirizzi fisici sono calcolati in fase di esecuzione).

Swapping (2/3)

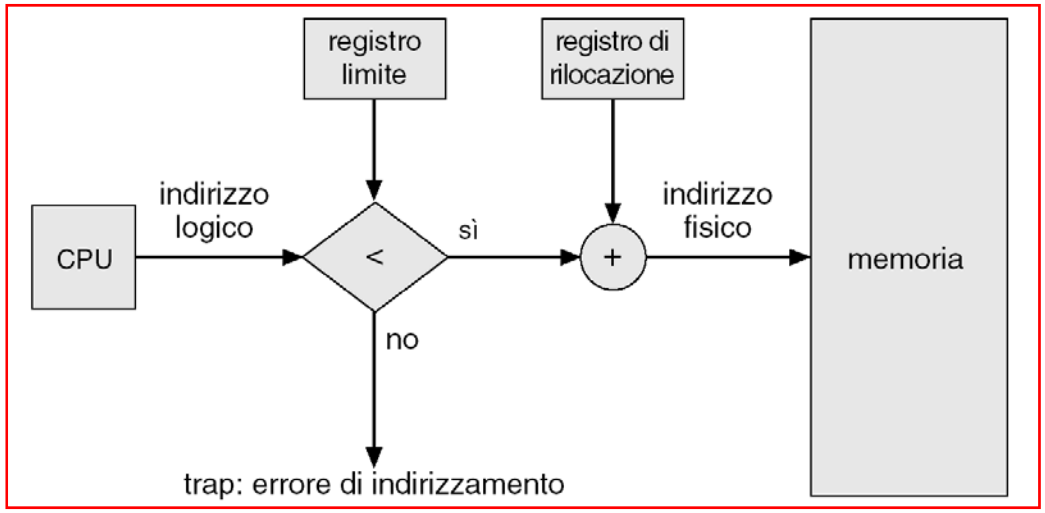
- Il sistema mantiene una **lista dei processi pronti** (le cui immagini sono in memoria temporanea o in memoria centrale).
  - Quando il microscheduler seleziona un processo, richiama il dispatcher che controlla se esso è in memoria centrale:
    - Se si, esso viene eseguito.
    - Se no, esso deve essere caricato da disco:
      - Se non c'è un'adeguata area libera in memoria, viene eseguito lo swap.
      - In tali casi i cambi di contesto sono piuttosto lunghi.
        - » Per un efficiente utilizzo della CPU è desiderabile che il tempo di esecuzione di ciascu processo sia lungo rispetto al tempo di swap.
- La maggior parte del tempo di swap è tempo di trasferimento; il tempo totale di trasferimento è direttamente proporzionale alla quantità di memoria spostata.
  - È utile conoscere quanta memoria sta usando un processo e non quanta potrebbe usarne, facendo swap della sola parte utilizzata.
  - Un processo con richieste di memoria dinamica dovrà adoperare le chiamate di sistema *request memory* e *release memory*.

Swapping (3/3)

- Se si desidera swappare un processo esso deve essere in uno stato di riposo.
  - Particolare attenzione va posta riguardo alle operazioni di I/O:
    - Se durante l'I/O un processo accede in modalità asincrona ai buffer dei dispositivi (in memoria utente) non può essere spostato.
      - Potrebbe accadere che l'operazione di I/O è in attesa sul device e a seguito di swap si potrebbe tentare di usare memoria appartenente ad un processo diverso da quello di partenza. Soluzioni:
        - » Non si esegue mai swap quando c'è un I/O in corso.
        - » Si esegue I/O solo nei buffer del sistema operativo.
- In generale lo spazio di swap è assegnato come porzione a sé stante del disco separata dal file system per velocizzarne gli accessi.
- Versioni modificate di swapping si trovano in molti sistemi operativi (ad esempio UNIX, Linux e Windows).
  - Di norma lo swapping è disabilitato, ma quando l'uso della memoria centrale supera una data soglia, la tecnica viene messa in funzione salvo poi ad essere nuovamente disabilitata quando scende il carico del sistema.

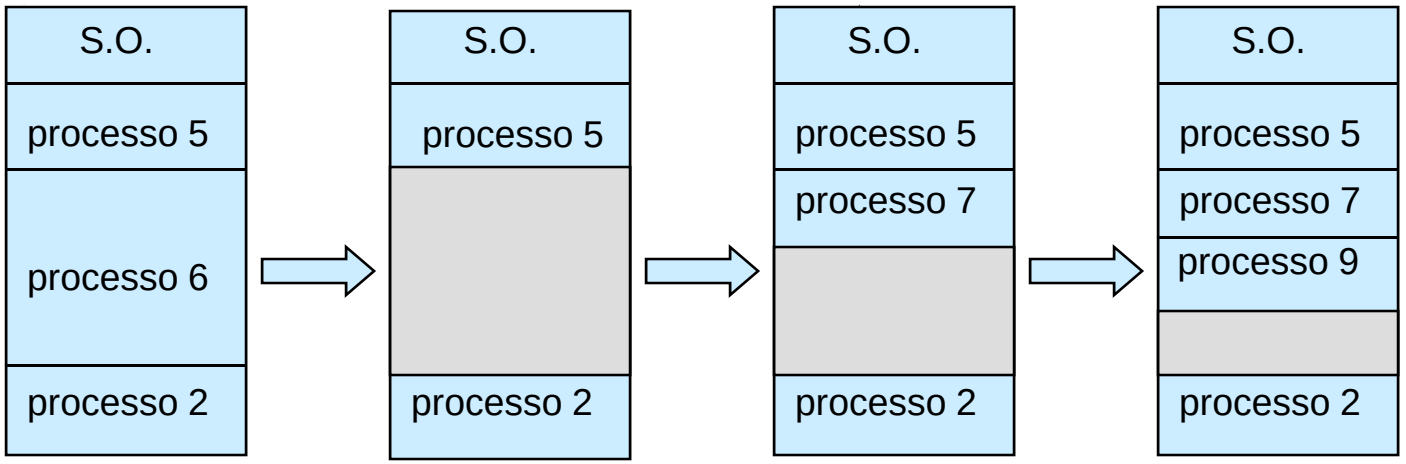
Allocazione contigua di memoria

- La memoria centrale è normalmente divisa in due partizioni:
  - Sistema operativo residente, di solito collocato nella memoria bassa con il vettore di interrupt.
  - Processi dell'utente poi collocati nella memoria alta.
- Allocazione a partizione singola
  - Il registro di rilocazione serve per proteggere il sistema operativo dai processi dell'utente ed i processi tra di loro.
  - Il registro di rilocazione contiene il valore del più piccolo indirizzo fisico; il registro limite contiene l'intervallo degli indirizzi logici – ogni indirizzo logico deve avere un valore inferiore rispetto a quello del registro limite.



Allocazione contigua di memoria

- Allocazione a partizioni multiple:
  - *Hole* – blocco di memoria centrale disponibile; blocchi di varie dimensioni sono sparsi nella memoria centrale.
  - Quando un processo arriva, il sistema cerca un blocco libero abbastanza grande da ospitare il processo.
  - Allocato il processo, l'O.S. modifica le informazioni su:
    a) partizioni allocate b) partizioni libere (hole)
  - L'O.S. può ordinare la coda dei processi in ingresso alla memoria in base all'algortimo di schedulazione. La memoria è assegnata fino a che le richieste possano essere soddisfatte. Se non c'è disponibiltà di un hole sufficientemente grande per un processo:
    - l'O.S. può attendere fino al *free* di uno spazio sufficiente
    - l'O.S. può scorrere la coda di entrata alla ricerca di un processo con richieste inferiori.



Allocazione dinamica

Come soddisfare una richiesta di dimensione *n* da una lista di blocchi liberi?

- **First-fit**: assegna il *primo* blocco libero abbastanza grande per contenere lo spazio richiesto.
- **Best-fit**: assegna *il più piccolo* blocco libero abbastanza grande. Bisogna cercare nell'intera lista, a meno che la lista non sia ordinata in base alla dimensione.
- **Worst-fit**: assegna il più grande blocco libero. Si deve nuovamente cercare nell'intera lista, a meno che non sia ordinata in base alla dimensione.

Sia il metodo first-fit sia quello best-fit sono migliori del metodo worst-fit in termini di tempo e di utilizzo della memoria centrale.

- Il metodo first-fit è generalmente più veloce del best-fit mentre essi risultano essere equivalenti dal punto di vista dell'ottimizzazione dello spazio in memoria.

Frammentazione

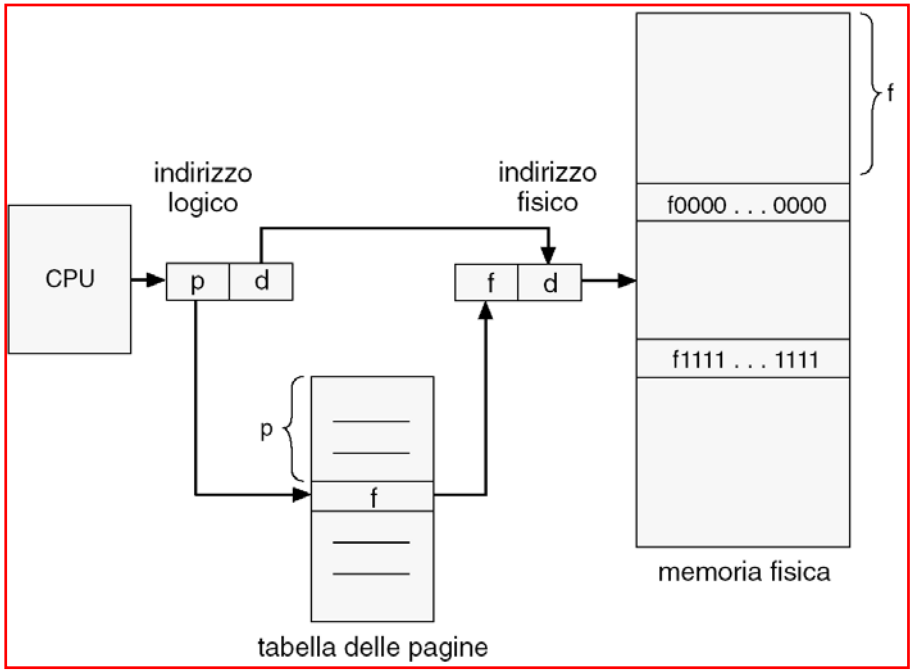
- **Frammentazione esterna** – c'è abbastanza spazio totale di memoria centrale per soddisfare una richiesta, ma gli spazi disponibili non sono contigui.
  - Ne soffrono le strategie first-fit e best-fit.
- Ridurre la frammentazione esterna attraverso la compattazione:
  - Fondere i contenuti della memoria centrale per avere tutta la memoria centrale libera in un grande blocco.
  - La compattazione è possibilie *solo* se la rilocazione è dinamica ed è fatta al momento dell'esecuzione (basta variare il valore del registro base e spostare codice e dati).
- Ridurre la frammentazione permettendo spazi di indirizzamento fisico non contigui mediante due tecniche principali:
  - Paginazione.
  - Segmentazione.
- **Frammentazione interna** – la porzione di memoria centrale allocata ad un processo può essere più grande di quanto richiesto; la differenza è una frazione della memoria centrale **interna** ad una partizione che non viene sfruttata.

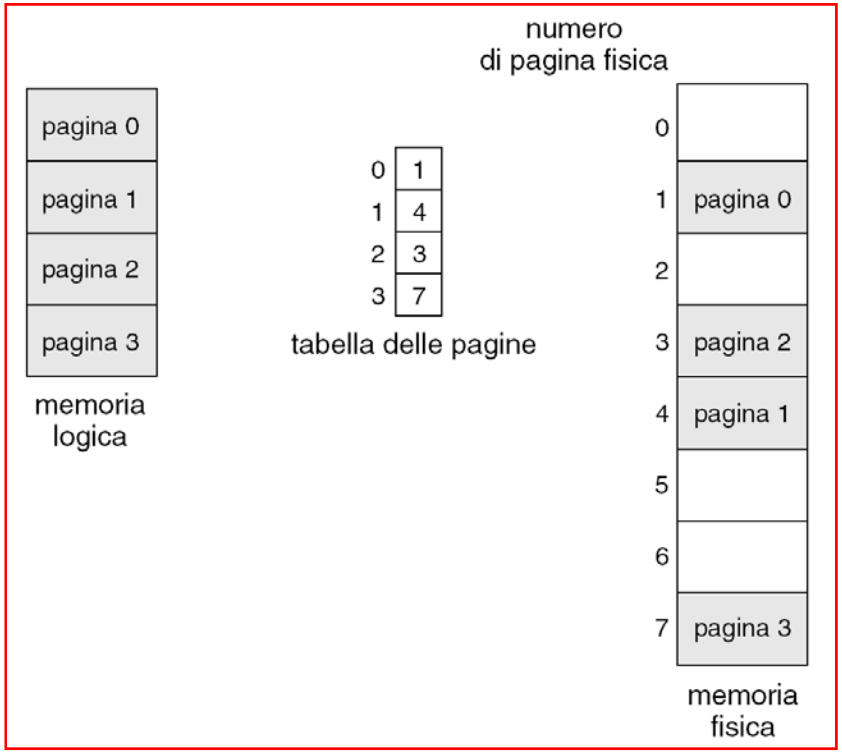
Paginazione

- Lo spazio degli indirizzi fisici può essere non contiguo; il processo è allocato alla memoria fisica tutte le volte che il più recente è disponibile.
- Suddivide la memoria fisica in blocchi di **frame** (la dimensione è una potenza di 2, fra 512B e 16MB) e mantiene traccia di tutti i frame liberi.
- La dimensione della pagina logica (pari a quella della pagina fisica) è definita dall'hardware.
  - Se la dimensione di un indirizzo logico è 2^m e abbiamo a disposizione n bit per il sistema di indirizzamento (dimensione massima della memoria) adopereremo m-n bit per il numero di pagina e n per il displacement.
- Divide la memoria logica in blocchi delle stesse dimensioni chiamati **pagine**.
- Per eseguire un processo di dimensione di *n* pagine, bisogna trovare *n* frame liberi e caricare il programma.
- Occorre impostare una tabella delle pagine per tradurre gli indirizzi logici in indirizzi fisici.
- Non esiste frammentazione esterna, ma frammentazione interna.

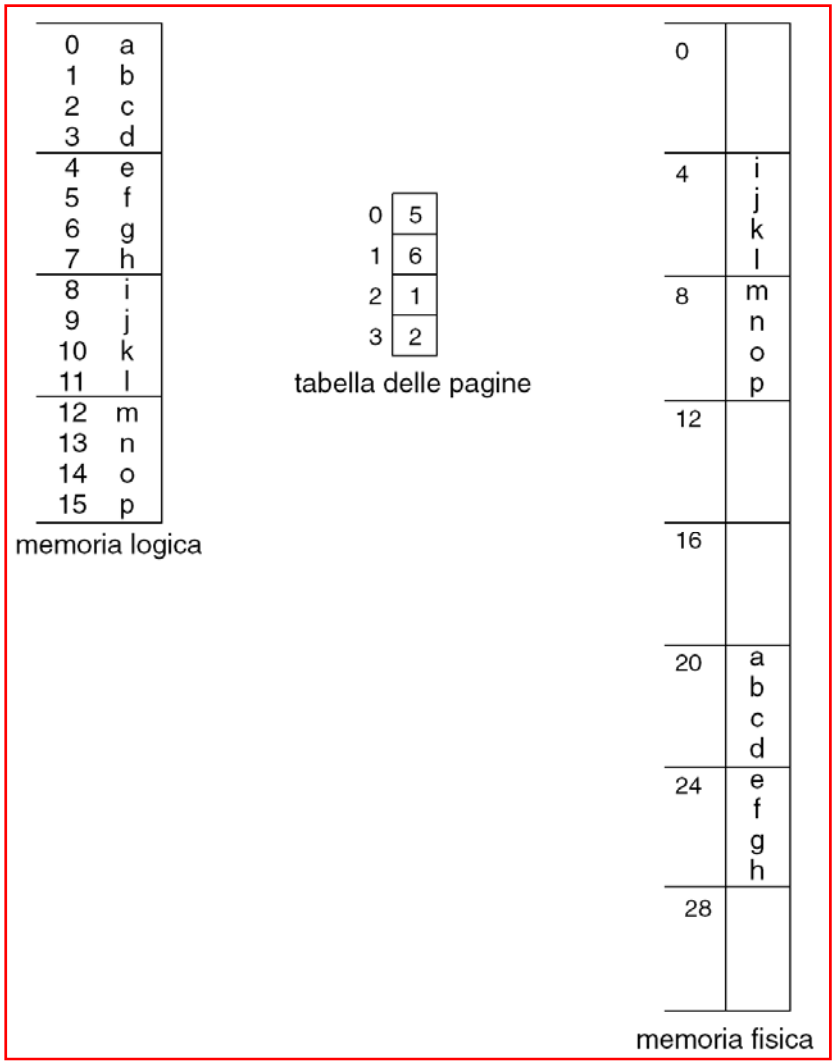
Traduzione di un indirizzo

- Ogni indirizzo generato dalla CPU è diviso in due parti:
  - *Numero di pagina (p)* – usato come indice nella *tabella delle pagine* che contiene l'indirizzo di base di ogni frame nella memoria fisica.
  - *Spiazzamento nella pagina (d)* – combinato con l'indirizzo di base per calcolare l'indirizzo di memoria fisica che viene mandato all'unità di memoria centrale.





Esempio



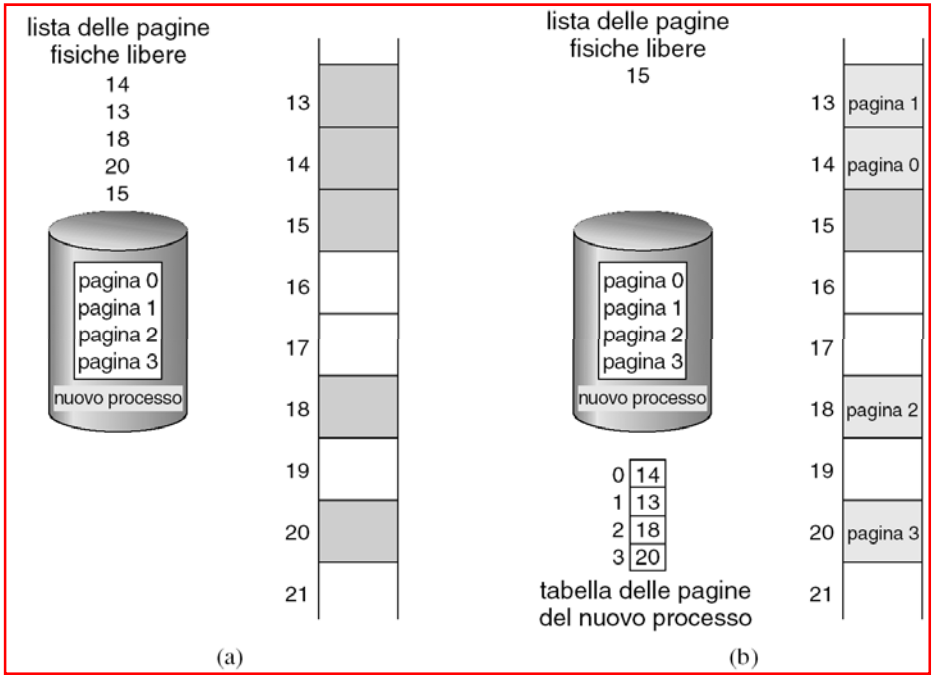
- Memoria centrale a 32 byte con pagine di 4 byte (8 pagine).
- L'indirizzo logico 0 va espresso come p=0, d=0.
- La pagina 0 è secondo la page table nel frame 5.
  - L'indirizzo logico 0 corrisponde a quello fisico 20 (5x4+0).
  - L'indirizzo logico 3 corrisponde a (p,d) = (0,3), quello fisico è 23 (5x4+3).
- L'indirizzo logico 4 va espresso come p=1, d=0. La pagina 1 è nel frame 6.
  - L'indirizzo logico 4 corrisponde a quello fisico 24 (6x4+0).
- L'indirizzo logico 13 va espresso come p=3, d=1. La pagina 3 è nel frame 2.
  - L'indirizzo logico 13 corrisponde a quello fisico 9 (2x4+1).

Frame liberi (1/2)

- La paginazione è una strategia di rilocazione dinamica che utilizza un registro di rilocazione per ogni frame della memoria centrale.
  - La page table può essere intesa come tabella di registri di rilocazione.
- Un processo che abbisogni di n pagine più un byte comporta l'allocazione di n+1 frame con un ultimo frame quasi completamente frammentato (caso peggiore di frammentazione interna).
  - La frammentazione interna in media è di mezza pagina per processo.
  - È auspicabile avere piccole dimensioni di pagina per ridurre la frammentazione interna, tuttavia:
    - con pagine piccole cresce la dimensione della page table;
    - gli accessi a disco sono maggiormente efficienti con grossi quantitativi di dati da trasferire.
- La dimensione tipica attuale delle pagine è compresa tra 4 kB e 8 kB.
- Solitamente ogni elemento della page table è lungo 4 byte (2^{32} frame nella memoria fisica) quindi con frame da 8 kB si può indirizzare una memoria fisica da 2^{45} byte (32 TB).

Frame liberi (2/2)

- L'utente vede la memoria centrale come un unico singolo spazio che contiene un dato programma mentre in realtà questo è suddiviso e sparpagliato nella memoria fisica insieme ad altri programmi.
- L'hardware di traduzione degli indrizzi riconcilia la visione della memoria dell'utente con quella reale.
- Le informazioni sui frame totali, su quelli allocati e su quelli disponibili vengono mantenute dall'O.S. in una **tabella dei frame**.



Prima dell'allocazione

Dopo l'allocazione

Page table

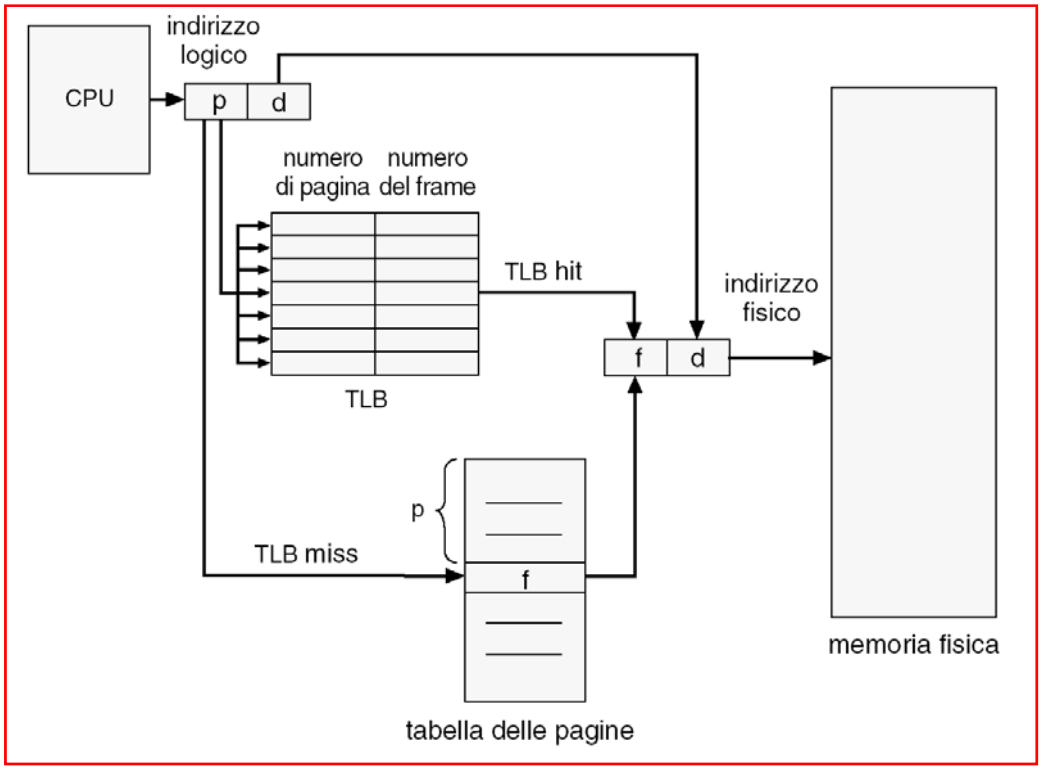
- La maggior parte degli O.S. alloca una page table per processo e un puntatore ad essa è memorizzato nel PCB relativo.
- La tabella delle pagine è mantenuta in registri dedicati (page-map table di piccole dimensioni) o più di frequente nella memoria centrale.
- Il *registro base della tabella delle pagine (*PTBR) punta alla tabella.
  - Il cambiamento di tabella richiede di modificare il solo PTBR riducendo di molto il tempo per il context switching.
- In questo schema ogni accesso a dati/istruzioni richiede due accessi alla memoria. Uno per la tabella delle pagine e uno per i dati/istruzioni.
  - L'accesso alla memoria centrale è rallentato di un fattore 2.
- I due problemi di accesso alla memoria posso essere risolti attraverso l'uso di una speciale piccola cache per l'indicizzazione veloce detta **T*ranslation Look-aside Buffer – TLB*.**
  - Si tratta di un hardware velocissimo ma molto costoso. Le dimensioni tipiche di un TLB tipicamente consentono di mantenere da 64 a 1024 elementi.

TLB

- Memoria associativa – ricerca parallela:

  Traduzione dell'indirizzo (p, d)
  - Se p si trova nella memoria associativa, si ottiene il frame#.
  - Altrimenti si ottiene il frame# dalla page map table in memoria centrale.

page#	frame#



Tempo di accesso effettivo

- L'utilizzo di un TLB può richiedere meno del 10% rispetto al caso di memoria non mappata.
- Se la TLB è già occupata il sistema operativo applica una politica di sostituzione LRU (Least Recently Used) oppure random.
  - Alcuni elementi di TLB possono essere *wired down* e non possono essere rimossi (codice del kernel).
- Per il calcolo del tempo effettivo di accesso alla memoria centrale occorre fare un calcolo pesato di tipo probabilistico. Sia:
  - Associative Lookup = ε unità di tempo.
  - Si assuma che il tempo di ciclo di memoria è ß unità di tempo.
  - Tasso di accesso con successo (*hit ratio)* α – frequenza delle volte che un particolare numero di pagina viene richiesto nella TLB.
  - Tempo di accesso effettivo (EAT):

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

- Esempio: 20 nsec per accedere al TLB, 100 nsec per accedere alla memoria, hit ratio 80%.
  - In caso di Hit: 120 nsec per un accesso mappato.
  - In caso di TLB miss: 20 nsec per l'accesso al TLB (miss), 100 nsec per recuperare la PMT dalla memoria e 100 nsec per l'accesso.

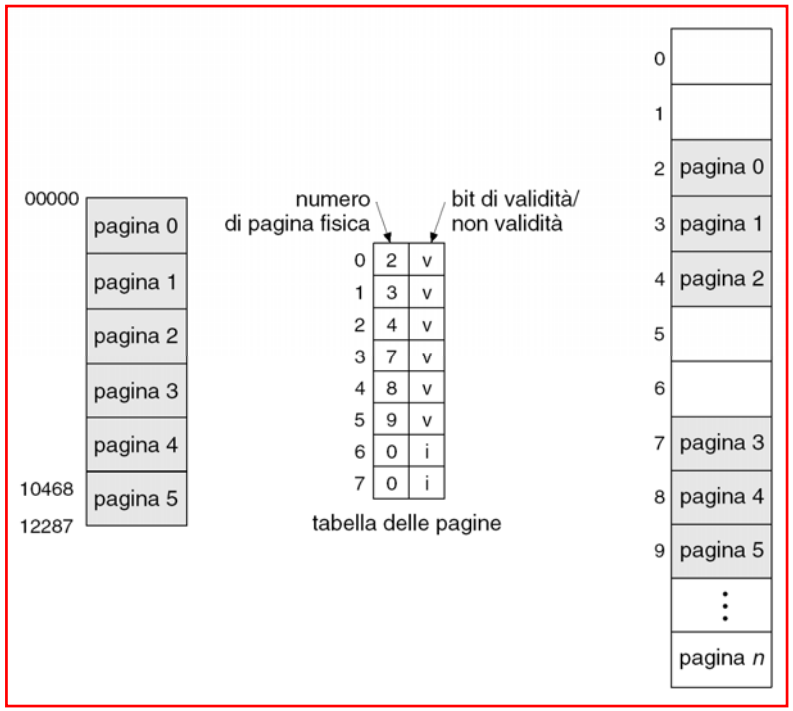
$$EAT = 0.8 \times 120 + 0.2 \times 220 = 140 \text{ nsec}$$

Protezione della memoria centrale

- La protezione della memoria centrale in ambiente paginato è ottenuta mediante **bit di protezione** associati a ciascun frame interni alla page map table.
- Un bit definisce l'attributo di una pagina (sola lettura/lettura-scrittura).
  - Poiché ogni riferimento alla memoria centrale passa attraverso la PMT, mentre viene calcolato l'indirizzo fisico possono essere controllati i bit di protezione.
    - Un tentativo di scrittura su una pagina read-only genera una trap hardware.
- Un bit **valid/invalid** è associato ad ogni elemento della PMT:
  - "valid" indica che la pagina associata è nello spazio degli indirizzi logici del processo ed è quindi una pagina legale (accesso alla pagina consentito).
  - "invalid" indica che la pagina non è nello spazio degli indirizzi logici del processo (accesso alla pagina negato).
- Un processo tipicamente usa solo una parte dello spazio a disposizione.
  - È dispendioso creare una PMT con il riferimento a tutte le pagine nell'intervallo degli indirizzi: la maggior parte di essa resterebbe inutilizzata.
  - Un Page-Table Length Register (PTLR) contiene la dimensione della PMT.
    - Il PTLR viene controllato in corrispondenza di ogni indirizzo logico per verificare che esso sia nell'intervallo di validità del processo.

Protezione della memoria centrale

- In un sistema con spazio di indirizzamento a 14 bit (0-16383), si supponga di avere un programma che sfrutti solo gli indirizzi 0-10468.
- Si supponga una pagina di dimensione 2kB.
  - Le pagine 0,1, .., 5 sono mappate regolarmente.
  - Le pagine 6 e 7 sono non valide.



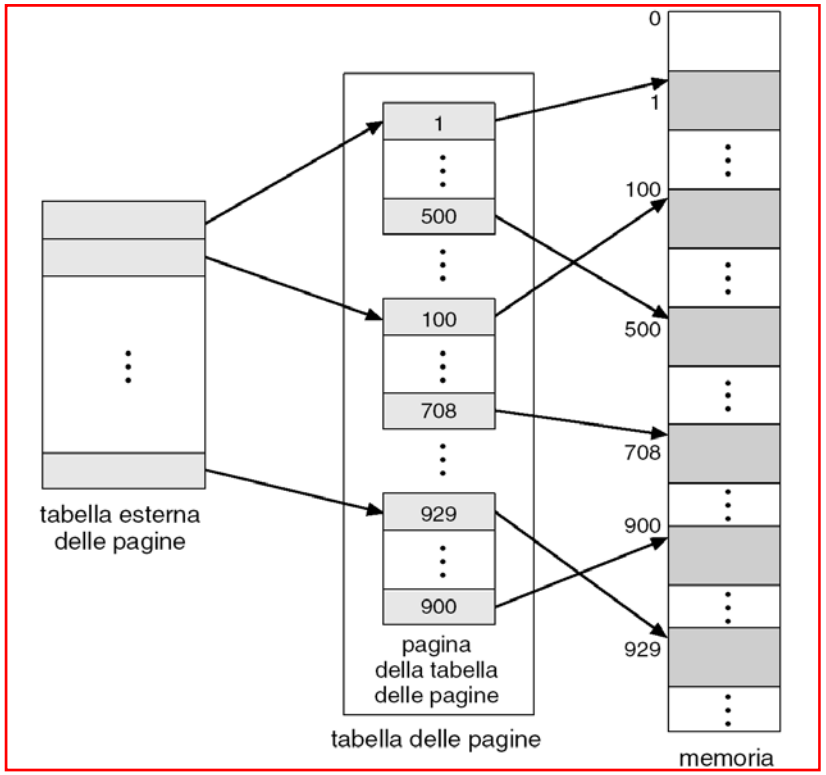
- NOTA: con 6 pagine (0-5) da 2kB si ha:
  - 6 x 2048 = 12288
  - Il programma si estende fino all'indirizzo 10468 e ogni riferimento al di là è classificato come illegale
  - I riferimenti alla pagina 5 (10240-12287) sono ritenuti validi.
  - L'incongruenza è frutto del problema della frammentazione interna.

Struttura della tabella delle pagine

- La maggioranza dei PC supporta vasti spazi di indirizzamento logico (da 2^{32} a 2^{64}) con un conseguente forte impatto della Page Table.
  - Con uno spazio a 32 bit e pagine di 4kB (2^{12}), la PMT potrebbe avere fino a 1 milione di elementi.
    - Se ciascun elemento occupa 4 byte la PMT occuperà 4MB.
- Occorrono tecniche efficienti per la strutturazione della tabella delle pagine,
  - Paginazione gerarchica.
  - Tabelle delle pagine con hashing.
  - Tabella delle pagine invertita.

Paginazione gerarchica

- Suddivide lo spazio degli indirizzi logici in più tabelle di pagine.
- Una tecnica semplice è la tabella delle pagine a due livelli.
  - La stessa PMT è paginata.
  - Occorrerà una External Page Map Table (EPMT) contenente il riferimento alle pagine della PMT.



- Poiché la traduzione di un indirizzo procede dall'esterno all'interno, questo schema prende anche il nome di **tabella delle pagine mappata in avanti**.

Paginazione a due livelli

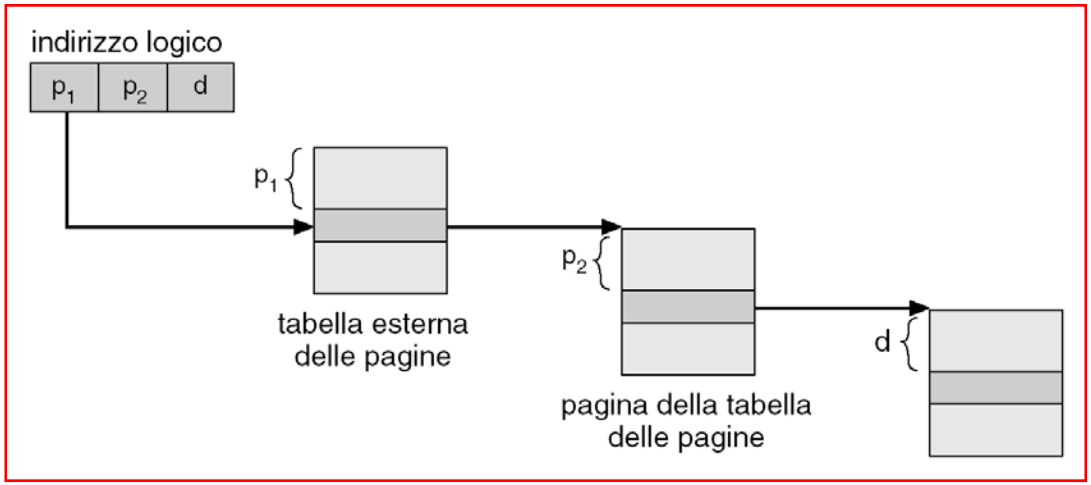
- Un indirizzo logico (su una macchina a 32 bit con pagine di 4kB) è diviso in:
  - un numero di pagina che consiste in 20 bit;
  - uno spiazzamento della pagina da 12 bit.
- Poichè paginiamo la tabella, il numero di pagina è ulteriormente diviso in:
  - un numero di pagina da 10 bit;
  - uno spiazzamento nella pagina da 10 bit.
- Pertanto un indirizzo logico è diviso come segue:

numero di pagina

p_1	p_2	d
10	10	12

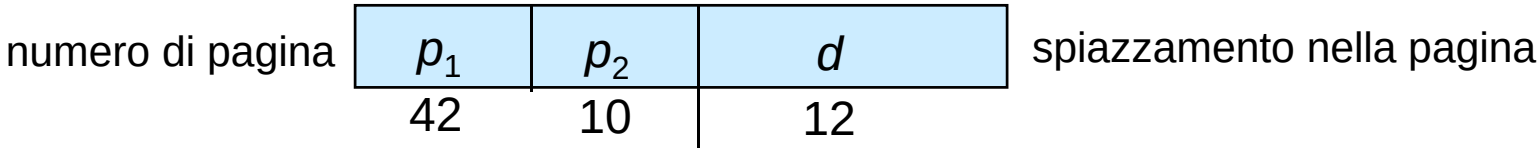
spiazzamento nella pagina

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



Schemi correttivi

- Traduzione dell'indirizzo per un architettura di paginazione a due livelli a 64 bit.
  - Supponiamo pagine da 4kB (2^{12} byte).
  - La tabella potrà ospitare fino a 2^{52} elementi.
  - Si potrebbe immaginare il seguente schema per gli indirizzi:



  - La EPMT sarebbe composta da 2^{42} elementi. Con elementi da 4 byte occuperebbe 2^{44} byte (16 TB).
  - Si può suddividere la EPMT in parti più piccole:
    - schema di paginazione a tre livelli.

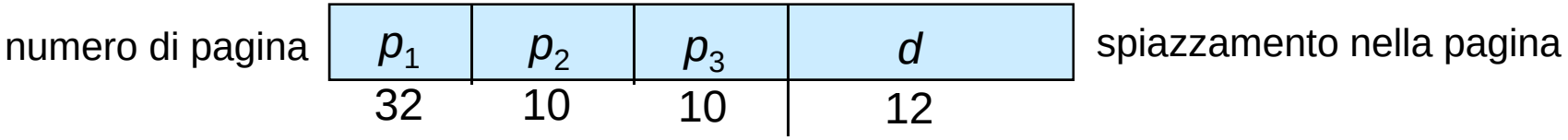


Tabelle delle pagine con hashing

- Comune per trattare gli spazi di indirizzamento più grandi di 32 bit.
- Il numero di pagina virtuale nell'indirizzo virtuale è inserito nella tabella come funzione di hashing. Ogni elemento della tabella contiene una lista di pagine che hanno lo stesso valore della funzione di hashing. Contiene:
  - Numero di pagina virtuale.
  - Numero del frame corrispondente alla pagina mappata.
  - Puntatore all'elemento successivo nella lista collegata.
- I numeri di pagina virtuali sono confrontati con questa lista cerando una corrispondenza. Se ne siste una, il relativo frame fisico viene estratto.
- Variante: clustered page table (ogni elemento della tabella di hashing si riferisce a 16 pagine).

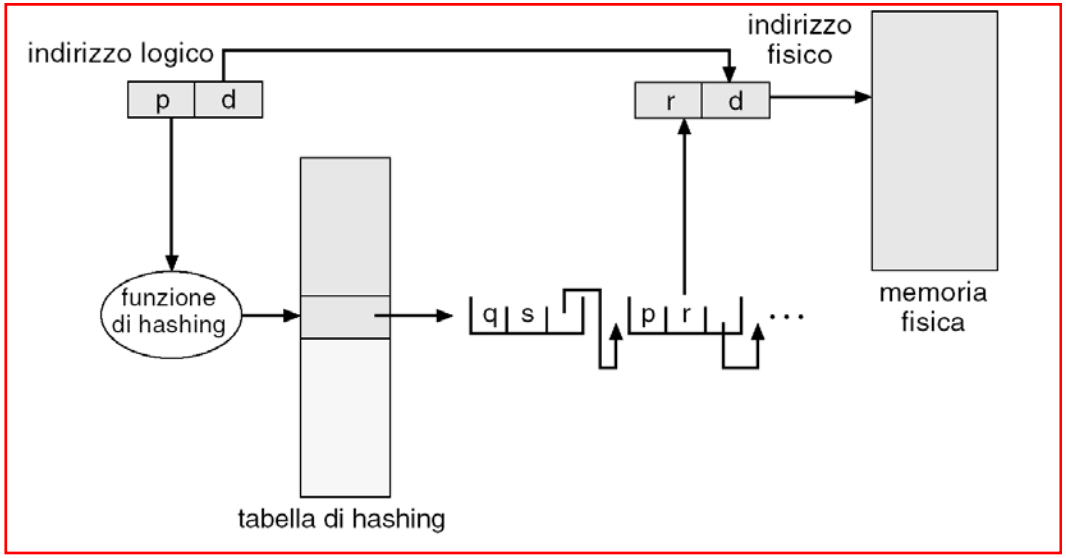
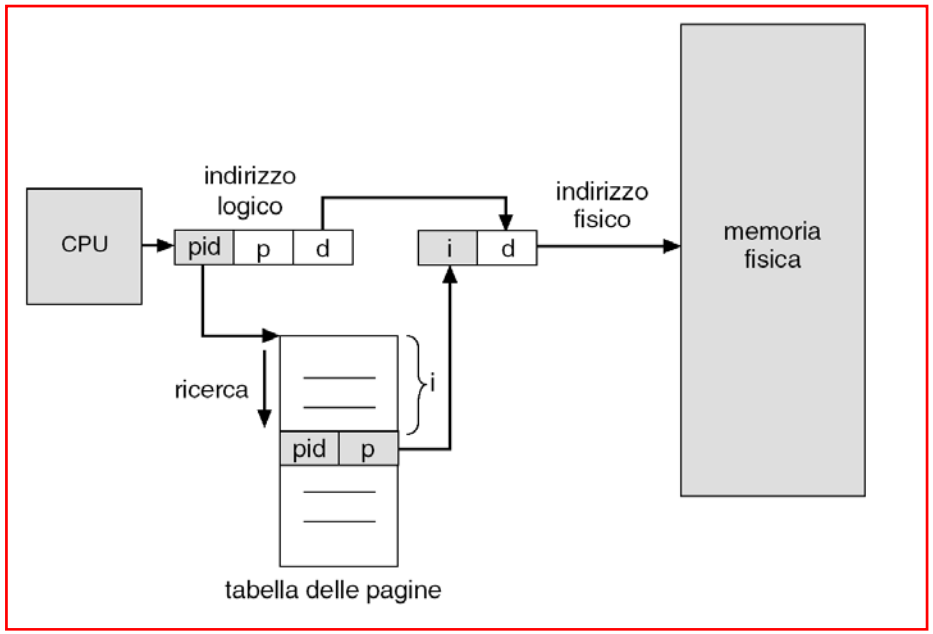


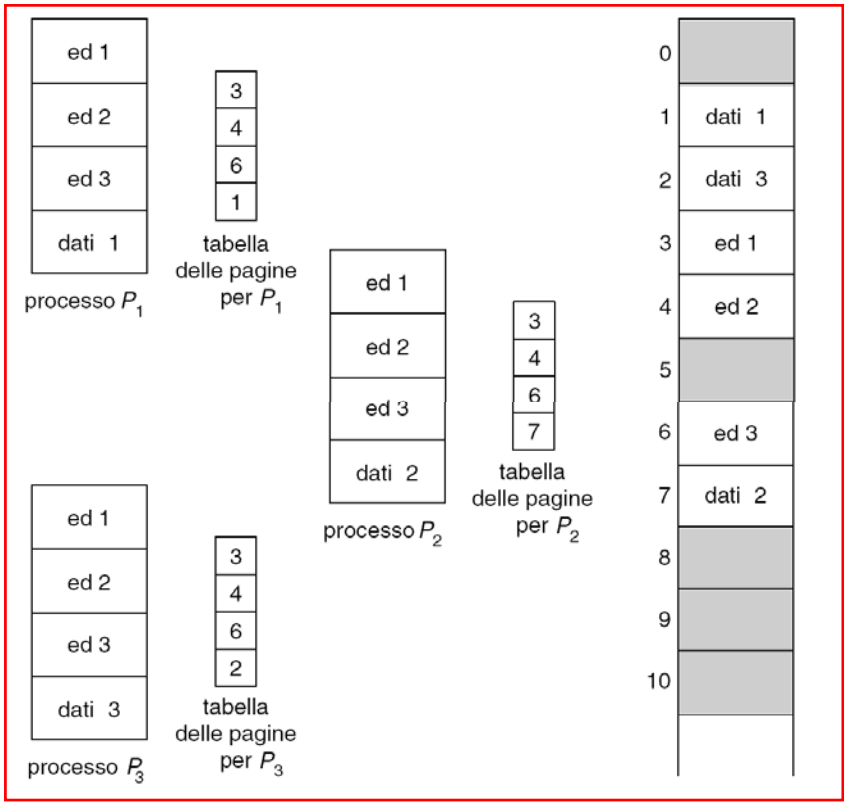
Tabella delle pagine invertita

- PROBLEMA: ogni processo ha la sua PMT con un elemento per ogni pagina che il processo sta usando: le PMT possono occupare grandi quantità di memoria.
- Si adopera lo schema della **tabella delle pagine invertita**.
  - Ciascun elemento consiste nell'indirizzo virtuale della pagina logica memorizzata in quella posizione di memoria fisica con informazioni sul processo cui appartiene quella pagina.
  - Nel sistema c'è una sola PMT con un solo elemento per ogni pagina fisica della memoria centrale.
    - Diminuzione della quantità di memoria centrale necessaria per memorizzare una tabella e aumento del tempo per cercare nella tabella quando si verifica un riferimento alla pagina.
      - Uso di una hashing table per limitare la ricerca a uno o a pochi elementi nella PMT.



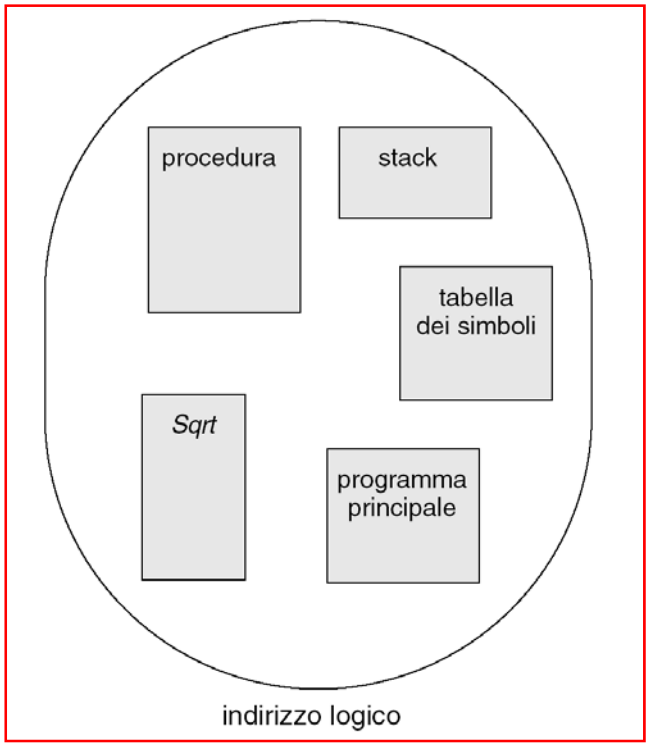
Pagine condivise

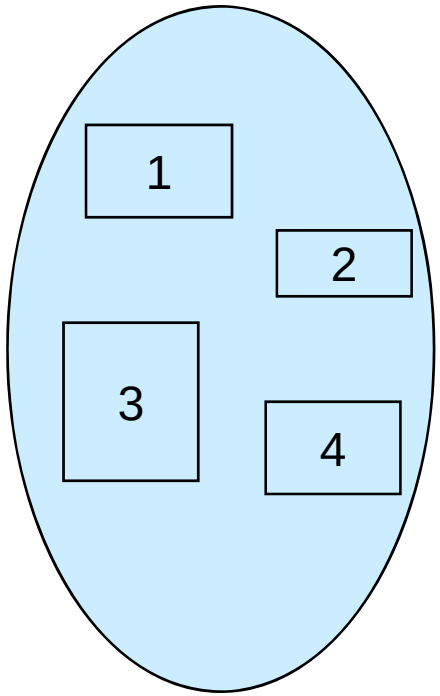
- Un vantaggio della paginazione è la possibilità di condivisione del codice comune.
- Codice condiviso.
  - Una copia di sola lettura del cosiddetto **codice rientrante** condiviso fra processi (ad esempio compilatori, librerie, basi di dati).
  - Il codice condiviso deve apparire nella stessa posizione dello spazio di indirizzo logico di tutti i processi.
- Codice privato e dati.
  - Ogni processo possiede una copia separata del codice e dei dati.
  - Le pagine per il codice privato ed i dati possono apparire ovunque nello spazio di indirizzo logico.
- Generalmente la condivisione si realizza mediante indirizzi virtuali multipli che mappano un unico indirizzo fisico.
  - I sistemi che adoperano tabelle delle pagine invertite hanno una sola pagina virtuale per pagina fisica.
    - La condivisione non è realizzabile (si avrebbero pagine fisiche con due o più indirizzi virtuali condivisi).



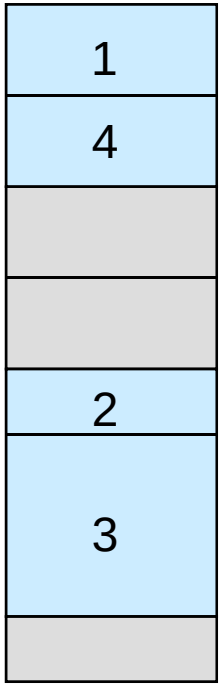
Segmentazione

- Schema di gestione della memoria centrale che supporta il punto di vista dell'utente della memoria centrale.
- Lo spazio di indirizzamento logico è un insieme di segmenti. Un segmento è un'unità logica (programma principale, procedura, funzione, metodo, oggetto, variabili locali, stack, array).





Spazio dell'utente



Spazio della memoria fisica

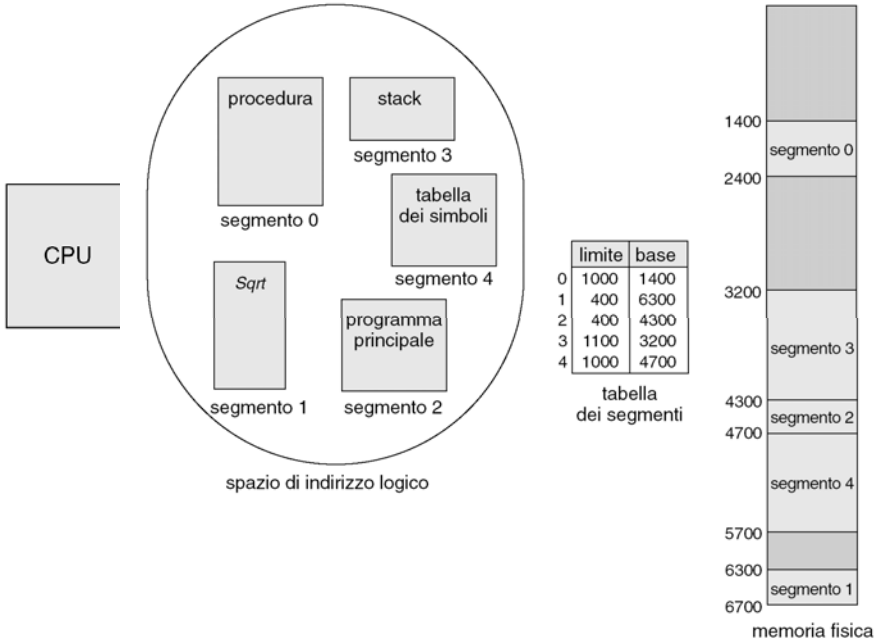
Architettura della segmentazione

- Un indirizzo logico è composto da due parti:

  <numero del segmento, spiazzamento>

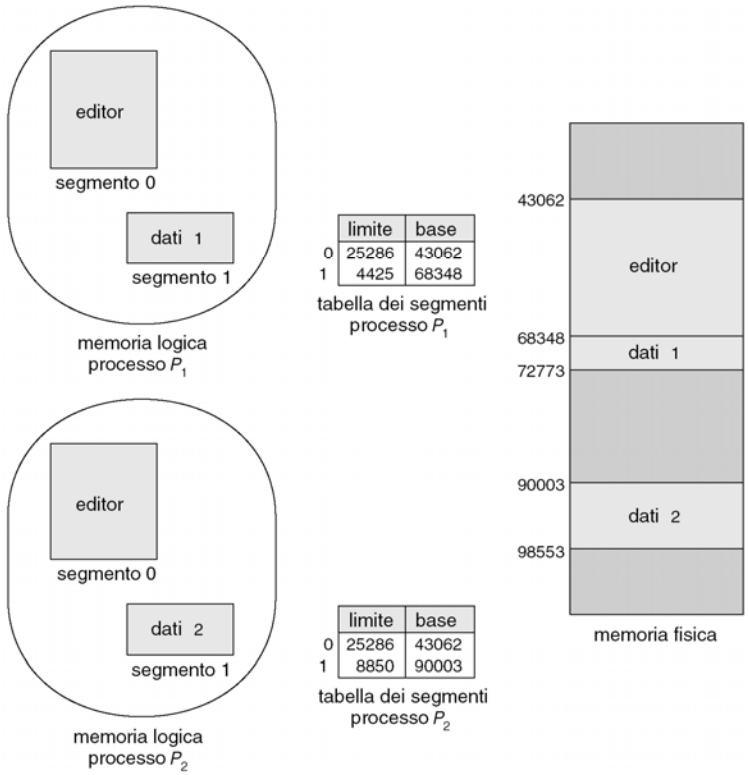
- *Tabella dei segmenti* – mappa gli indirizzi bidimensionali in indirizzi fisici; ogni elemento della tabella ha:
  - Una *base* – contiene l'indirizzo fisico di partenza in cui il segmento risiede in memoria centrale.
  - Un *limite* – specifica la lunghezza del segmento stesso.

- *Registro base del segmento dalla pagina (STBR)* punta alla tabella del segmento che si trova in memoria.
- *Registro della lunghezza del segmento della pagina (STLR)* indica il numero dei segmenti usati dal programma; il numero del segmento *s* è legale se *s* < STLR.



Protezione e condivisione

- Protezione. Associazione di un bit di validità a ciascun segmento:
  - bit di validità = 0 $\Rightarrow$ segmento illegale.
  - privilegi di lettura/scrittura/esecuzione.
- Un segmento è una porzione semanticamente definita di un programma.
  - È altamente probabile che tutte le parti del segmento siano adoperate allo stesso modo.
- Ogni processo ha una Segment Table.
  - Un segmento è condiviso quando gli ingressi di due o più processi nella ST puntano alla stessa locazione fisica.
- La condivisione del codice o dei dati è particolarmente agevole nel caso di segmentazione.
  - Essa si verifica a livello di segmento.
  - Qualsiasi informazione può essere condivisa se definita come segmento.



Frammentazione

- Allocazione. Poichè i segmenti sono di varia lunghezza, l'allocazione in memoria è un problema di allocazione dinamica.
  - First fit/best fit.
  - Frammentazione esterna.
    - Quando tutti i blocchi in memoria sono troppo piccoli per ospitare un segmento occorrerà attendere per la liberazione di blocchi o eseguire la compattazione
- La segmentazione è per natura un algoritmo di rilocazione dinamica.
  - In linea teorica si potrebbe disporre ogni byte in un proprio segmento e rilocarlo separatamente.
    - Eliminazione totale della frammentazione esterna.
    - Avremmo bisogno di un registro base per la rilocazione di ogni byte (raddoppio dell'occupazione di memoria!)
  - Se la dimensione media dei segmenti è piccola, sarà basso il livello di frammentazione.

Segmentazione paginata

- MULTICS risolve i problemi della frammentazione esterna e della lunghezza dei tempi di ricerca attraverso la paginazione dei segmenti.
- Nella segmentazione con paginazione le informazioni della tabella dei segmenti non contengono l'indirizzo di un segmento quanto piuttosto l'indirizzo della tabella delle pagine per quel segmento.
- Lo spazio di indirizzamento logico di un processo è diviso in due partizioni:
  - Segmenti riservati (le cui informazioni sono contenute nella Local Descriptor Table (LDT)).
  - Segmenti condivisi (le cui informazioni sono contenute nella Global Descriptor Table (GDT)).
- Ogni elemento della LDT e della GDT consiste in un descrittore del segmento con informazioni dettagliate su quel segmento (tra le altre anche base e limite del segmento).
- Un indirizzo logico è dato da una coppia <selettore, spiazzamento> ove:
  - Selettore = <s, g, p> a 16 bit con:
    - s: numero di segmento (13 bit);
    - g: segmento nella GDT o nella LDT (1 bit);
    - p: protezione (2 bit).
  - Lo spiazzamento è un valore a 32 bit che specifica la posizione della word nel segmento.

Segmentazione paginata

- Le informazioni BASE e LIMITE contenute nel descrittore di segmento sono adoperate per costruire un indirizzo lineare:
  - Se l'indirizzo è valido (controllo sul valore LIMITE) si somma lo spiazzamento al valore BASE;
  - L'indirizzo lineare adotta uno schema di paginazione a due livelli comprendente:
    - Puntatore all'indice di pagina, puntatore alla PMT e spiazzamento.