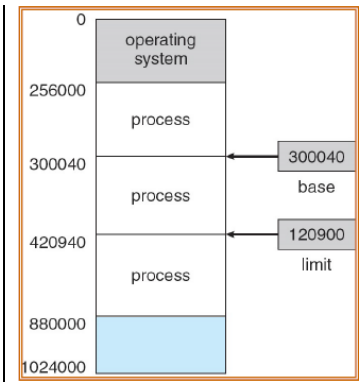
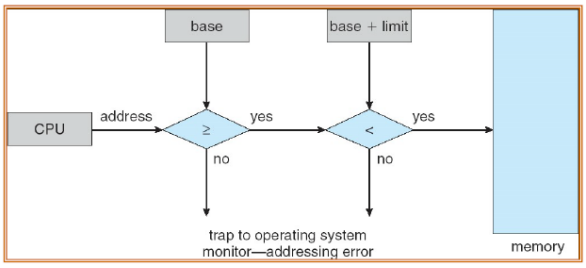
9. GESTIONE DELLA MEMORIA CENTRALE

Un programma in genere risiede su disco in forma di un file binario eseguibile e deve essere portato (dal disco) in memoria e "inserito" all'interno di un processo per essere eseguito e quindi il sistema operativo assegna al processo uno spazio di memoria.

La parte alta della memoria principale viene assegnata al kernel del sistema operativo e in quel momento il kernel viene caricato e poi c'è il resto della memoria principale che libera.
Man mano che si creano i processi a questi viene assegnato un pezzo di memoria principale e all'interno di essi viene immagazzinato tutto il necessario di tale processo.
Al processo in questione gli viene assegnato uno spazio ben definito.
Il sistema operativo deve però mettere un sistema di protezione a questo spazio dedicato al processo, in modo che nessuno invada il suo spazio o che egli invada altri spazi.
Ogni qualvolta che la CPU sta eseguendo un certo processo emette un indirizzo, il SO deve garantire che quell'indirizzo che è stato richiesto è parte dello spazio che è stato assegnato a quel processo specifico.
Per garantire ciò il SO assegna ad ogni processo una coppia di registri (registro **base** e **limit**) che definiscono lo spazio di indirizzamento fisico del processo.



Questi registri **base** e **limit**, vengono utilizzati con questi controlli →

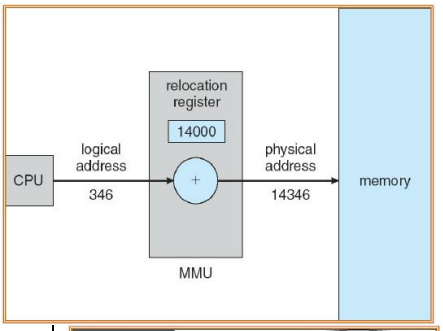


Il collegamento delle istruzioni e dei dati agli indirizzi di memoria può essere effettuato in:

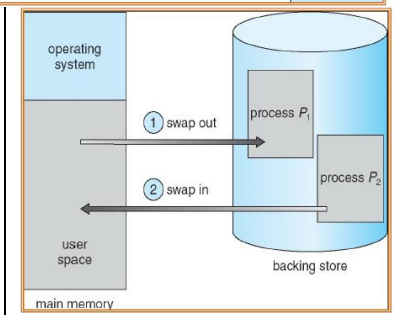
- **Fase (o tempo) di compilazione**: se in fase di compilazione si conosce dove il processo risiederà in memoria, allora si può generare un codice assoluto. Se tale indirizzo cambia per una qualche ragione bisogna ricompilare;
- **Fase di caricamento**: il compilatore genera un codice rilocabile. Il caricatore associa indirizzi rilocabili ad indirizzi di memoria. Se l'indirizzo iniziale cambia è sufficiente ricaricare in memoria il codice;
- **Fase (o tempo) di esecuzione**: se il processo può venire 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 supporto hardware.

Indirizzo logico – indirizzo generato dalla CPU, anche definito come indirizzo virtuale.
Indirizzo fisico – indirizzo visto dalla memoria.

Fase di compilazione e di caricamento: indirizzi logici e fisici identici.
Fase di esecuzione: indirizzi logici e fisici diversi.



CARICAMENTO DINAMICO:
Una procedura non è caricata finché non è chiamata, vengono mantenute in memoria secondaria in un formato rilocabile.
Un problema che si risolve in questo modo è lo **swapping**, ovvero un processo può essere temporaneamente spostato e poi riportato in memoria centrale (swap out, swap in). La maggior parte del tempo di swap è tempo di trasferimento, il tempo totale di trasferimento è direttamente proporzionale alla quantità di memoria interessata, ma anche problemi di I/O.

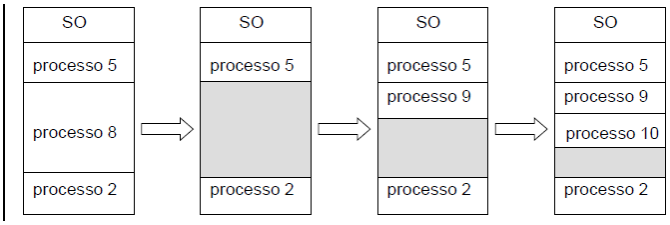


ALLOCAZIONE CONTIGUA DI MEMORIA:
La memoria centrale è normalmente divisa in due partizioni:

- Sistema operativo residente, collocato nella memoria bassa con il vettore degli interrupt.
- Processi dell'utente collocati nella memoria alta.

Con l'allocazione contigua, ciascun processo è contenuto in una singola sezione contigua di memoria

Quando un processo arriva, il sistema cerca nell'insieme un blocco libero abbastanza grande per questo processo.
Supponiamo che il processo 8 termini e lascia un buco (***hole***) nella memoria, arrivano altri processi e vengono allocati in quello spazio, arriva il processo 11 che vorrebbe entrare anche lui ma non può perché non c'è abbastanza spazio, in realtà non c'è spazio abbastanza spazio contiguo (*frammentazione esterna*).



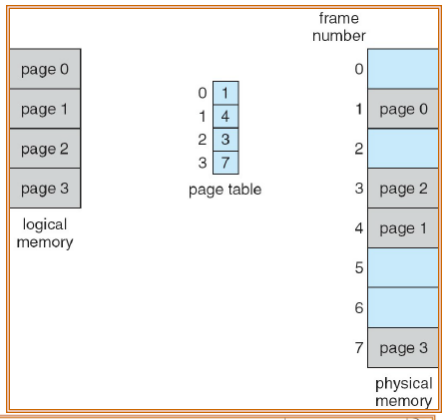
Si prende la memoria principale piuttosto che vederla come una sequenza di Byte la si suddivide in blocchi di dati di potenza di 2, così quando un processo arriva gli si dà un certo numero di blocchi contigui. Esistono varie politiche per assegnare blocchi liberi:

- **First-fit** (più veloce perche non deve esaminale l'intera lista): assegna il primo blocco libero abbastanza grande per contenere lo spazio richiesto.
- **Best-fit**: assegna il più piccolo blocco libero abbastanza grande.
- **Worst-fit** (meno efficiente sull'utilizzo della memoria): assegna il più grande blocco libero.

9.1 PAGINAZIONE

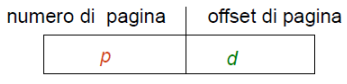
Dobbiamo capire come il sistema operativo assegna spazio ai vari processi quando vengono mandati in esecuzione. I principali approcci sono i First-fit, Best-fit e Worst-fit, ma questi possono provocare frammentazione esterna.

Per ovviare a tali problemi si usa la **paginazione**:
Si suddivide la memoria principale in blocchi chiamati **frame** che sono potenze di 2, si prende il pezzo di memoria che il processo necessita e si suddivide anche questo in blocchi chiamate **pagine**.
La size della **pagina** è uguale alla size di un **frame**, così che ciascuna pagina può calzare all'interno di un frame della memoria principale.
Quindi l'idea è di assegnare le varie pagine del processo a frame liberi in memoria principale.
Per non perdere i vari pezzi del processo per la memoria principale, si imposta una **tabella delle pagine**, si va a scrivere dove sono state collocate tutte le pagine del processo.



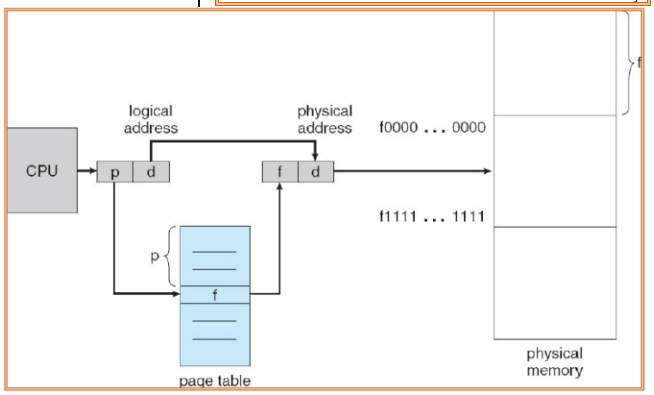
TRADUZIONE INDIRIZZO LOGICO IN FISICO:
Ogni indirizzo generato dalla CPU è diviso in due parti:

- **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.

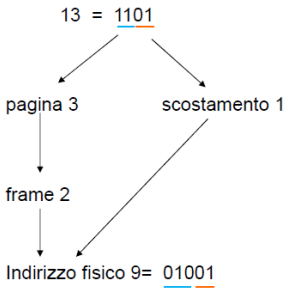


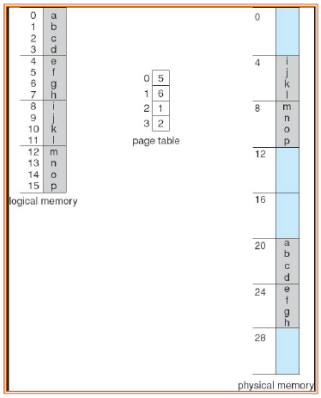
2^m = spazio degli indirizzi logici

2^n = size di una pagina



Esempio di paginazione per una memoria centrale di 32 byte con pagine di 4 byte





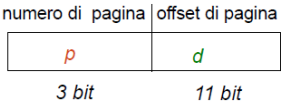
Esempio di paginazione

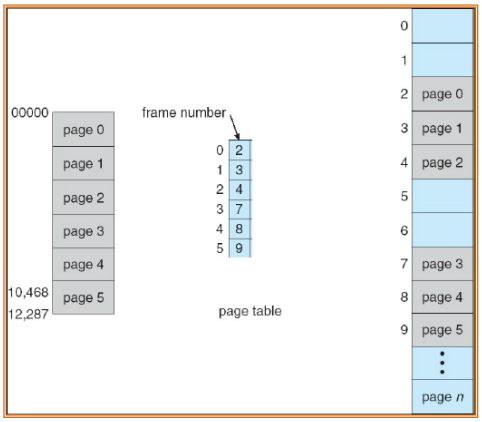
size di un frame=size di una pagina = 2048 = 2^{11}

size del processo = 10469= (2^{11} x 5)+229 => # pagine = 6 (cioè 12288 = 2^{11} x 6 byte)

11 bit = bit necessari per rappresentare una posizione in una pagina (o frame), cioè per *d*

3 bit = bit necessari per rappresentare il numero di una pagina, cioè per *p*





2^{14} = spazio degli indirizzi logici

2^{11} = size di una pagina

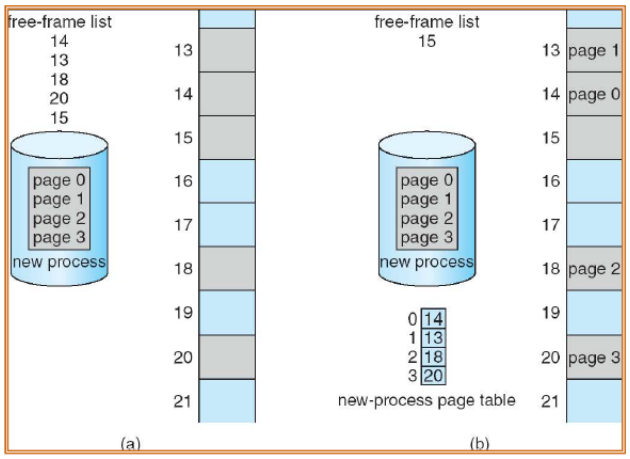
9.2 IMPLEMENTAZIONE DI UNA TABELLA DELLE PAGINE

Il SO conserva una **lista dei frame liberi**, quando un processo parte e richiede un numero di pagine allora il sistema va nella lista e prende tutti i frame necessari.

La tabella delle pagine è mantenuta nella memoria centrale assieme al processo.
Il registro base della tabella delle pagine (**PTBR**) punta alla tabella, quando vi è un cambio di contesto basta cambiare il contenuto del PTBR.

Ogni accesso a dati/istruzioni richiede **due accessi alla memoria**: uno per la tabella delle pagine e uno per dati/istruzioni (indirizzo fisico).

Il problema dei due accessi alla memoria può essere risolto attraverso l'uso di una speciale piccola cache per l'indicizzazione veloce detta memoria associativa (o translation look-aside buffer – **TLB**).



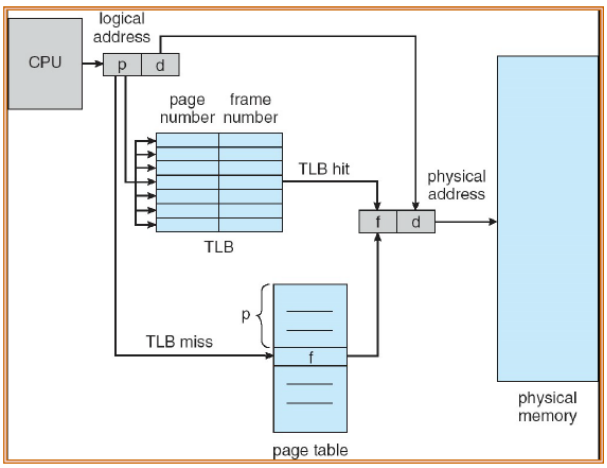
Prima dell' allocazione

Dopo l'allocazione

La **TLB** è una memoria associativa che viene posta vicino alla CPU, e quindi facilmente accessibile e non si trova in memoria principale, che contiene un numero limitato di entry, si segna le coppie accedute frequentemente.

Esiste un ASID che serve a identificare il processo e quindi evitare che un processo faccia riferimento a pagine conservate in tale tabella cache ma non appartenenti al processo in uso, più processi possono essere mantenuti nella TLB.

Se la TLB non consente l'uso dell'ASID, viene cancellato il contenuto di TLB ad ogni cambio di contesto.



TEMPO DI ACCESSO EFFETTIVO:

- *Tempo di accesso alla TLB (associative lookup)* = ε unità di tempo;
- *Tempo di accesso a memoria* = 1 unità di tempo;
- *Tasso di accesso con successo (hit ratio)*, frequenza delle volte in cui un particolare numero di pagina viene trovato nella TLB = α;
- *Tempo di accesso effettivo (**EAT**):*

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

La prima parte rappresenta il caso di successo, in cui con probabilità α trovo l'indirizzo nella TLB e quindi il tempo effettivo sarà (1+ε), la seconda parte rappresenta il caso di insuccesso, con probabilità (1 – α), il tempo effettivo sarà (2 + ε).

PROTEZIONE DELLA MEMORIA PRINCIPALE:

La protezione della memoria centrale in ambiente paginato è ottenuta mediante **bit di protezione** associati ai frame.

Un bit di validità/invalidità è associato ad ogni elemento della tabella:

- "**valido**" indica che la pagina associata è nello spazio degli indirizzi logici del processo ed è quindi una pagina legale.
- "**non valido**" indica che la pagina non è nello spazio degli indirizzi logici del processo.

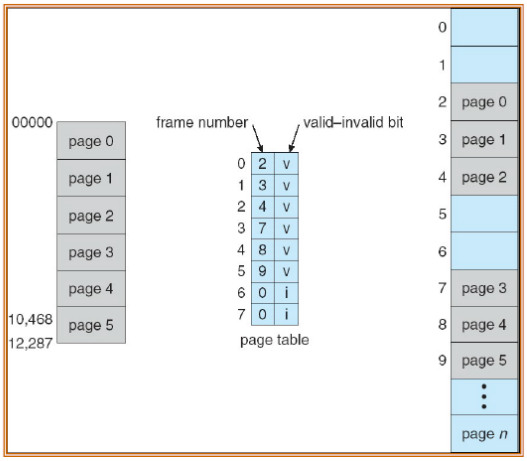
Capita raramente che un processo faccia uso di tutto il suo intervallo di indirizzi, gran parte della tabella rimane inutilizzata. Il registro della lunghezza della tabella delle pagine (PTLR) indica la dimensione della tabella delle pagine del processo. Permette di evitare grosse tabelle delle pagine con molte entrate associate a pagine invalide.

14 bit = lunghezza dell' indirizzo logico
2^{14} = spazio logico

size di un frame=size di una pagina = 2048 = 2^{11}

size del processo = 10469= (2^{11} x 5)+229 => # pagine = 6 (cioè 12288 = 2^{11} x 6 byte)

2^{14} / 2^{11} = 2^3 = 8 pagine => nella PT ci sono 8 entry; di queste ne vengono usate solo 6 (quelle che possono contenere 12288 byte)



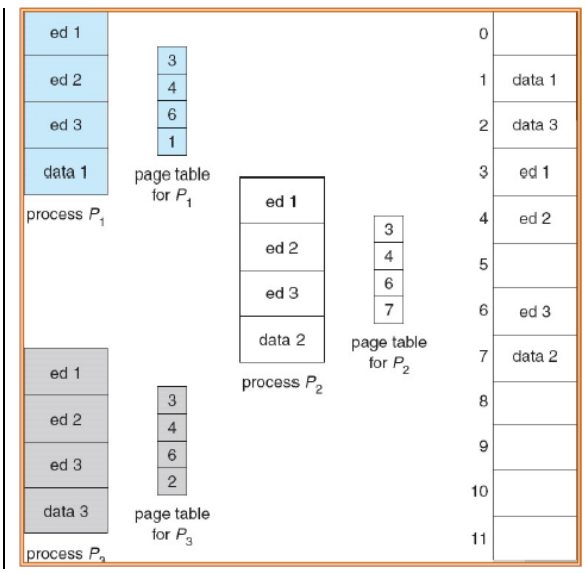
PAGINE CONDIVISE DAI PROCESSI:

Codice condiviso:

- Una copia di sola lettura di codice rientrante (i.e., non automodificante = non cambia durante l'esecuzione) condiviso fra processi (ad esempio editor, basi di dati).
- Le tabelle delle pagine dei processi associano le pagine logiche del codice agli stessi frame fisici.

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



9.3 STRUTTURA DELLA TABELLA DELLE PAGINE

La tabella delle pagine ha un problema, ovvero che queste tabelle possono essere molto grandi, relative agli indirizzi logici.

Con pagine da 4 KB=2^{12} ci sarebbero PT da $2^{32}/2^{12}=2^{20}$ entry, chiaramente sarebbe meglio non collocare in maniera contigua in memoria tale tabella.

Soluzioni: Tabella gerarchica, Tabella delle pagine con hashing e Tabella delle pagine invertita.

TABELLA GERARCHICA:

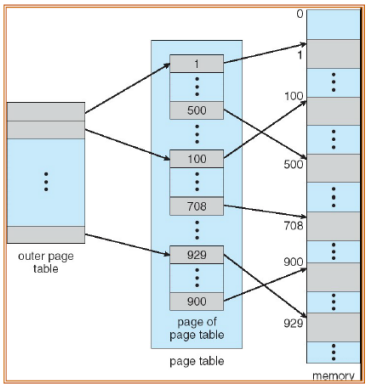
Suddivide lo spazio degli indirizzi logici in più tabelle di pagine, ad esempio la tabella delle pagine a due livelli.

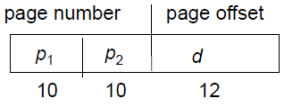
Un indirizzo logico (su una macchina a 32-bit con pagine di 4K) è diviso in:

- un numero di pagina di 20 bit;
- uno spiazzamento della pagina di 12 bit.

Il numero di pagina è ulteriormente diviso in :

- un numero di pagina da 10 bit;
- uno spiazzamento nella pagina da 10 bit.





dove p1 è un indice per la tabella esterna, e p2 rappresenta lo spostamento nella pagina della tabella interna.

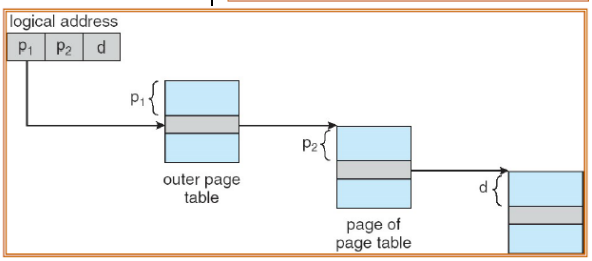


TABELLA DELLE PAGINE CON HASHING:

Comune per trattare gli spazi di indirizzamento più grandi di 32 bit. Ottima per spazi "sparsi".

Il numero di pagina virtuale p è l'input di una funzione hash. Il valore hash prodotto viene usato come indice nella tabella. Ogni elemento della tabella contiene una ***lista*** di coppie (numero pagina, frame) ***concatenata***.

Il numero di pagina p viene confrontato con i numeri di pagina di questa lista, cercando una corrispondenza. Se viene trovata, il numero di frame fisico associato al numero di pagina viene estratto.

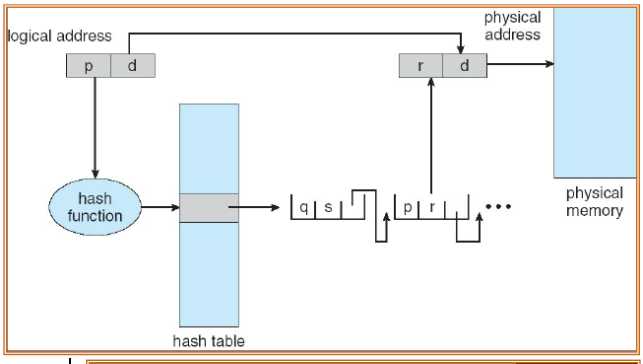


TABELLA DELLE PAGINE INVERTITA:

Un elemento per ogni frame della memoria centrale.

Ciascun elemento contiene il numero della pagina logica memorizzata in quel frame fisico, con informazioni sul processo cui appartiene quella pagina.

Questo schema permette di diminuire la quantità di memoria centrale necessaria per memorizzare le tabelle ma aumenta il tempo necessario per cercare nella tabella quando si verifica un riferimento alla pagina.

Usare una tabella con hashing per limitare la ricerca a uno o al più a pochi elementi nella tabella delle pagine.