giovedì 22 giugno 2023

ARCHITETTURE DI CALCOLO LEZIONE 22

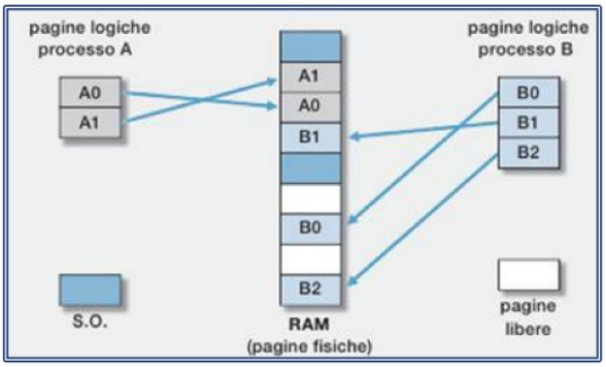
Segmentazione con paginazione, Memoria virtuale e algoritmi

TECNICHE DI ALLOCAZIONE NON CONTIGUA

Segmentazione
Prevede sostanzialmente la divisione di un programma (in termini di memoria) in blocchi di dimensioni variabili, in cui ogni blocco rappresenta una funzione (o una parte del programma). I singoli segmenti possono essere allocati in memoria indipendentemente uno dall'altro. Quindi, permette di inserire il programma in memoria sfruttando al meglio lo spazio disponibile dividendo i processi in parti variabili.

Paginazione
È un altro schema di gestione della memoria, in cui il programma viene diviso in parti con dimensioni uguali (fisse). Quindi gli indirizzi del programma vengono trasformati in indirizzi costituiti da due parti, una parte rappresenta il numero del segmento mentre l'altra rappresenta l'offset all'interno delle pagine. La paginazione è abbastanza simile alla segmentazione però la differenza sta nel fatto che il segmento ha dimensioni variabili mentre la pagina ha dimensioni fisse; usando dimensioni fisse diventa molto più semplice incastrare una pagina con le altre in memoria. Nella paginazione, ogni "parte" in cui viene diviso un programma prende il nome frame.
La paginazione nasce per semplificare l'allocazione della memoria; nei moderni sistemi operativi la paginazione è la tecnica più usata. Con la paginazione, lo spazio degli indirizzi fisici di un processo può essere non contiguo, evitando la frammentazione esterna e la necessità di compattazione.
Frammentazione esterna: la memoria disponibile è frammentata e di dimensione talmente ridotta da non poter essere utilizzata. In effetti, la paginazione costituisce una soluzione al problema della frammentazione esterna perché consente l'allocazione di memoria fisica solo per blocchi di dimensione prefissata.
Frammentazione interna: lo spazio che rimane in un frame è usato solo parzialmente. Esempio: se ho frame fissi di 4k e devo mettere in memoria un programma di 12k devo usare 3 frame (3*4=12). Se ho, invece, un programma di 11k devo sempre usare 3 frame ma l'ultimo viene riempito solo di 3k, quindi l'ultimo k dell'ultimo frame rimane inutilizzato.

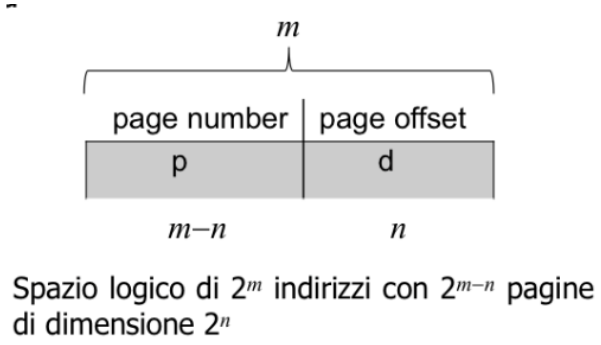
Nota: bisogna distinguere il termine pagina dal termine frame. La pagina è un concetto logico (cioè un concetto scritto dal programma) invece il frame è un concetto fisico riferito ai blocchi di memoria in cui è suddivisa la RAM. Ovviamente i frame e le pagine devono avere la stessa dimensione, perché dobbiamo mettere le pagine nei frame. I frame hanno dimensioni che sono potenze di 2 (es: 1k, 2k, 4k,…).



Osserviamo l'esempio della figura a sinistra: abbiamo due processi (processo A e processo B), il processo A è costituito da due pagine (A0,A1) e il processo B è costituito da tre pagine (B0, B1, B2). Come si nota le dimensioni delle pagine sono tutte uguali. Le pagine bianche sono libere quindi, se viene un terzo processo che ha bisogna di queste due pagine, banalmente le occupa. Dallo schema si capisce che se dobbiamo mettere in esecuzione un programma di dimensione N (numero delle pagine), abbiamo bisogno di N frame. Per gestire l'associazione tra pagine e frame si usa una tabella delle pagine.

Traduzione degli indirizzi

In generale, se la CPU genera un indirizzo di dimensione m (dove m è il numero di bit) posso vedere questo indirizzo come diviso in due parti: parte sinistra che rappresenta il numero della pagina e la parte destra che rappresenta l'offset.

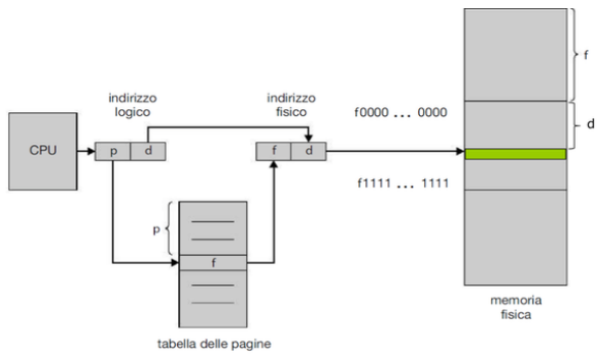


Spazio logico di 2^m indirizzi con 2^{m-n} pagine di dimensione 2^n

•Numero di pagina: impiegato come indice in una tabella delle pagine che contiene l'indirizzo base di ciascun frame nella memoria fisica (o il numero del frame). È costituito da n bit inferiore ad m.
•Offset nella pagina: combinato con l'indirizzo base per definire l'indirizzo fisico che viene inviato all'unità di memoria. La linea verticale determina quanti bit si usano per il numero di pagine e quanti per l'offset.

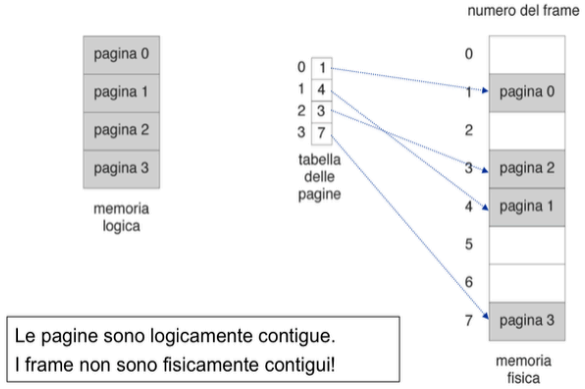
Supporto hardware alla paginazione

Di solito nei computer la paginazione viene supportata da hardware, per motivi di efficienza. Infatti, il SO usa una serie di registri dedicati con cui sostanzialmente si facilita la traduzione indirizzo logico - indirizzo fisico. Gli step sono i seguenti:



1.La CPU genera p e d (numero di pagina e offset); il numero di pagina (p) fa riferimento alla tabella delle pagine.
2.Si va nella riga p-esima e si trova f, indirizzo di base (posizione zero del frame).
3.Si somma d (offset) a f e si ottiene l'indirizzo fisico della cella ricercata. d è la distanza dell'indirizzo cercato rispetto all'inizio del frame (f) e ci indica la cella in verde; quindi, l'indirizzo logico p-d si trasforma in una cella della RAM.

Lo schema è un esempio di tabella delle pagine che mostra la relazione tra le pagine logiche del programma e il frame della memoria fisica. Il programma dal punto di vista logico è fatto da una serie di pagine contigue tra loro. Nella memoria, il programma viene spezzato in più parti e solamente tramite la tabella della pagina riusciamo a ricostruire dove si trovano le varie pagine. Per esempio, la pagina 0 si trova nel frame 1, la pagina 1 sta nel frame 4, la pagina 2 è nel frame 3 e la pagina 3 nel frame 7. Esiste una tabella delle pagine per ogni processo.
La figura che segue indica un esempio di come l'indirizzo binario che viene creato dalla CPU possa essere interpretato come coppia p-d.

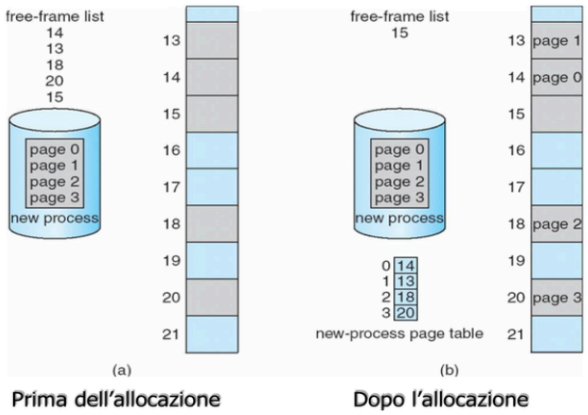


Supponiamo di avere una memoria fisica di 32 byte che è condivisa in frame di 4 byte; se in questo momento solamente il programma indicato a sinistra sta girando in questa memoria, si hanno 4
frame occupati mentre gli altri 4 risultano liberi. La tabella delle pagine ci dirà dove si trovano le varie pagine nei frame.
Esempio: per m=4 (m è il numero di bit di indirizzi), gli indirizzi logici sono al massimo 16 (da 0 fino a 15). Se si ha n=2, dove n è il numero di bit sottoinsieme di m che si stanno usando per rappresentare l'offset, la dimensione della pagina è 4. Quindi, il valore di n all'interno di m determina quanto è grande una pagina.
Quando la CPU genera un numero binario, per esempio 0110, in funzione di quanti bit vale n, una parte la considera come p e una parte come d. Nell'esempio, gli ultimi due bit (10) rappresentano l'offset, mentre le prime due (01) rappresentano il numero della pagina. Convertendo in decimali, 01 diventa 1 (pagina=1), 10 significa 2 (offset=2). Questi due numeri vengono combinati tra loro per ottenere l'indirizzo lineare all'interno della cella tramite la seguente operazione: dimensione della pagina (in questo esempio è pari a 4) per il numero del frame della pagina (in questo caso 6) più l'offset ((6*4)+2=26), ottenendo la cella 26.

Tabella dei frame

Poiché il SO gestisce la memoria fisica, deve essere informato su:
- Quali frame sono assegnati e a chi;
- Quali e quanti frame sono liberi.

Per tenere traccia di queste informazioni è necessaria la combinazione di due strutture di dati:
- Una tabella delle pagine per ogni processo;
- Una struttura dati contenente l'elenco dei frame liberi (free-frame list).



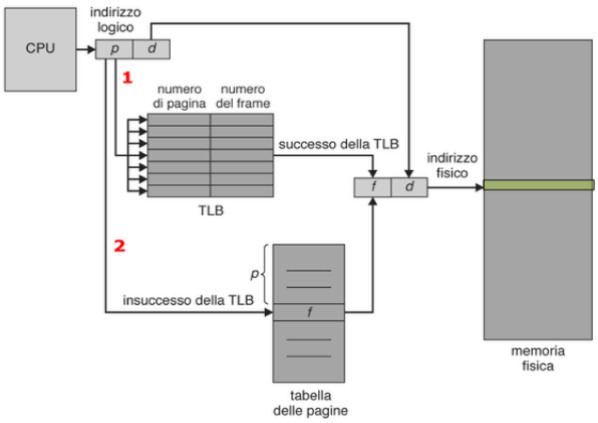
Qui c'è un esempio: supponiamo di avere un nuovo processo che vuole allocare le sue 4 pagine; il SO va ad allocare le pagine nei primi 4 frame indicati nella free-frame list, in successione 14 – 13 – 18 -20; soltanto il frame 15 rimane libero, ottenendo la situazione mostrata a destra della figura.

Sistema di paginazione e scheduling della CPU

Il dispatcher è un componente dello scheduler della CPU che si occupa dell'operazione di messa in esecuzione di un processo. Il dispatcher in genere gestisce il cambio di contesto durante il quale bisogna occuparsi dell'allocazione delle pagine dei processi nei frame. Ciò comporta che nella fase di dispatching si perda un po' di tempo per fare l'allocazione. Tuttavia, la paginazione consente di utilizzare al meglio lo spazio di memoria e di indirizzare ad una memoria fisica che è decisamente più grande di quella logica, indirizzabile direttamente dall'architettura hardware.

Implementazione della tabella delle pagine

Ogni sistema operativo ha il proprio metodo per memorizzare le tabelle delle pagine. Alcuni impiegano una tabella delle pagine per ciascun processo.



Il PCB (process control block) contiene, insieme col valore di altri registri, come il registro delle istruzioni, un puntatore alla tabella delle pagine. Queste due informazioni prendono il nome di PTBR e PTLR.

•Il registro Page−Table Base Register (PTBR) punta all'inizio della tabella. Il cambio delle tabelle delle pagine richiede soltanto di modificare questo registro, riducendo considerevolmente il tempo dei cambi di contesto.

•Il registro Page−Table Length Register (PTLR) indica la dimensione della tabella.

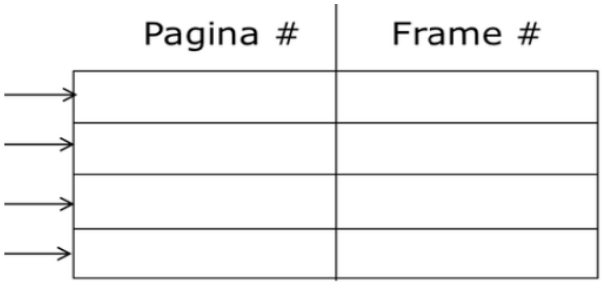
Per esempio ho un processo da 10 pagine (conseguentemente anche una tabella delle pagine con 10 elementi); questi 10 elementi sono memorizzati nella RAM non nel PCB, ma nel PCB ci sono le informazioni per trovare la tabella, ossia le posizioni d'inizio della tabella (PTBR) e la lunghezza della tabella (PTLR).

Ogni volta che il programma deve accedere ad una cella di memoria deve fare un doppio passaggio, prima deve andare nella tabella per la traduzione e poi nella memoria. Quindi, nei sistemi paginati la traduzione dell'indirizzo ha un costo doppo. Per cercare di ridurre tale costo, possiamo usare registri associativi (altrimenti detti translation look–aside buffer, TLB)

attraverso i quali si effettua una rapida ricerca parallela su una parte della tabella delle pagine (64−1024 elementi). Il TLB sostanzialmente è una struttura dati che permette di tradurre velocemente il numero di pagine e il numero di frame senza fare questo doppio passaggio che abbiamo detto prima (un po' come funziona la cache).

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
In pratica la CPU genera sempre una coppia p-d, ma in prima istanza, quando si genera l'indirizzo logico, va a cercare il numero della pagina richiesta nella TLB. Le frecce multiple indicano il concetto di ricerca in parallelo.



Se nella TLB si trova il numero di pagina che cerchiamo, verrà individuato velocemente il numero di frame; tale numero viene sommato all'offset, ottenendo l'indirizzo fisico d'interesse. Se, tuttavia, la ricerca nella TLB fallisce, è necessario cercare p, si va nella tabella delle pagine del processo (che troviamo grazie alle informazioni nel PCB, ossia indirizzo di base e lunghezza), troviamo il numero del frame e lo si somma sempre all'offset.
Ovviamente la ricerca nella TLB sara più veloce della ricerca nella tabella delle pagine, ma dobbiamo anche considerare "quante volte siamo fortunati" (cioè la probabilità che la pagina cercata sia nella TLB); in genere si considerarano vari parametri:

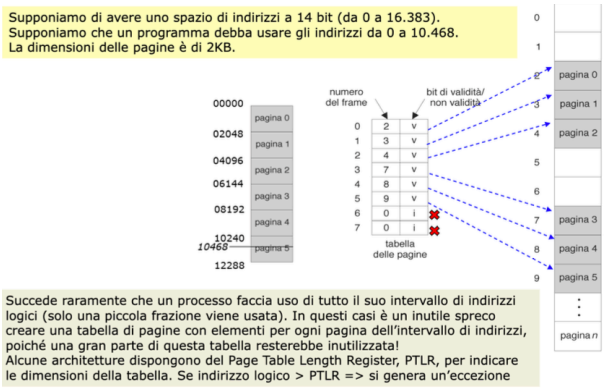
Tempo di accesso effettivo

	$TAE = hr \times (la+tm) + (1-hr) \times (la+2tm)$
(se hr=90%)	$= 0.9 \times (20+100) + 0.1 \times (20+200)$
	$= 108 + 22 = 130$ nsec
	$TAE = hr \times (la+tm) + (1-hr) \times (la+2tm)$
(se hr=70%)	$= 0.7 \times (20+100) + 0.3 \times (20+200)$
	$= 84 + 66 = 150$ nsec

•Lookup Associativo (la) - tempo di accesso alla memoria associativa.
•Hit ratio (hr) – percentuale delle 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).
Accanto un esempio di calcolo del tempo di accesso effettivo al variare dell'hit ratio.

Altri sistemi operativi usano un numero limitato di tabelle delle pagine (oppure una sola di queste) in modo da diminuire l'overhead per i cambi di contesto dei processi.
NOTA: la tabella delle pagine risiede in memoria centrale; in generale, si ha una tabella per ogni processo.

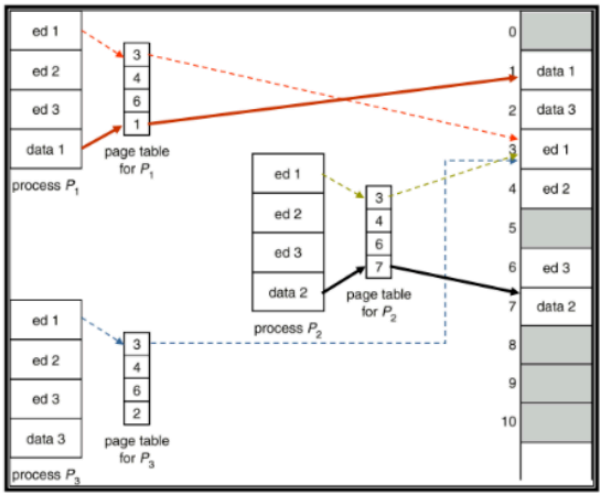
PROTEZIONE DELLA MEMORIA



Nella tabella delle pagine, alcune pagine possono essere marcate come non valide; questo accade quando la pagina associata non è nello spazio degli indirizzi del processo (non è caricata nella RAM e non è attualmente accessibile).
La protezione della memoria è implementata associando dei bit di protezione ad ogni frame. Permettono di indicare i possibili accessi al frame (lettura, scrittura, esecuzione). In alcuni casi si usa anche un bit di validità.
Tale bit può assumere valore 0 se la pagina non è valida ed 1 se la pagina non è valida e viceversa, in base alle condizioni scelte.
Nell'immagine osserviamo la struttura: nella tabella delle pagine viene aggiunta una colonna che corrisponde al bit di validità (qui con valore v per pagina valida ed i per pagina non valida).

Pagine condivise
Un altro aspetto che è simile a quello della segmentazione è relativo alle pagine condivise; la paginazione permette di realizzare semplicemente la condivisione di memoria tra processi su pagine di SOLA LETTURA. Ad essere condiviso, infatti, è il solo codice, non modificabile. Quindi, se si ha un codice che può essere utilizzato da più programmi, si mette in pagine di sola lettura e, in quanto contenenti solo codice (in gergo tecnico codice rientrante), non si modificano. Nel caso di pagine che contengono codici o dati privati non verranno condivisi ma si utilizzerà un approccio diverso.

Esempio

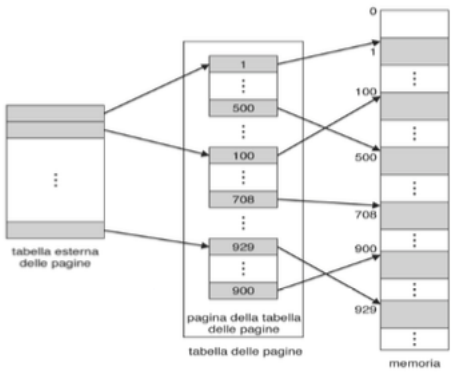


Sono presenti tre processi (p1 p2 p3) e tutti si occupano di dati.
Siccome fanno girare tutti lo stesso codice, la parte operativa (uguale per tutti) viene condivisa, mentre la parte del programma relativa ai dati non può esser condivisa.
Prendendo il processo p1, potrebbe avere 4 pagine dove le prime 3 sono relative al programma di editing e l'ultima ai dati.
Stessa cosa sui processi p2 e p3.
Siccome il codice dell'editor (Ed1. Ed2. Ed3) viene richiamato da tutti e tre i processi, tali pagine vengono messd una volta sola nei frame e poi si fa in modo che i processi puntino agli stessi frame,

confluendo nella stessa tabella delle pagine; l'ultima pagina, relativa ai dati, è diversa per ogni processo e viene salvata in un frame non condiviso.
Tale tecnica è utile per risparmiare memoria in quanto quando, ad esempio, si aprono 100 volte le stesse pagine (da parte di processi diversi) il programma non caricherà più copie dello stesso codice in memoria ma lo caricherà solo una volta.

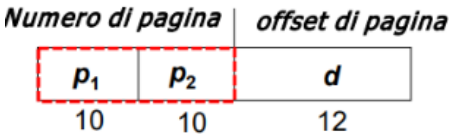
Struttura della tabella

L'allocazione di grandi programmi potrebbe non essere possibile in una tabella di pagine perché, ad esempio, in alcuni programmi che richiedono 100mila righe potrebbe essere un problema trovare uno spazio contiguo per l'allocazione di tali dati.



La soluzione a questo problema si risolve con il partizionamento.
Si usa l'approccio della paginazione gerarchica:
- Si divide lo spazio degli indirizzi logici in più tabelle.
- Una tecnica semplice è basata sull'uso di una tabella a due livelli.
- A volte, si usano anche tabelle a tre livelli.
Altro approccio è la tabella delle pagine invertita dove si ha un'unica struttura dati (tabella) globale che contiene un elemento per ogni frame. La più semplice è la tabella a due livelli in cui la prima pagina punta ad altre pagine che contengono i dati veri della tabella.

Se il numero di pagine viene diviso a sua volta in due parti (esempio 32 bit divisi in 20 per il numero di pagina e 12 per l'offset di pagina), quanto è grande il frame? 2^12 = 4096 bite.

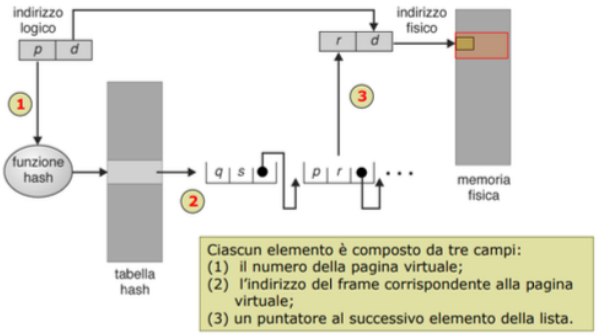


Poiché siamo nella tabella a 2 livelli, la tabella delle pagine è composta a sua volta da più tabelle, il numero di pagina è diviso in:
•Un numero di pagina di 10 bit.
•Un offset di pagina di 10 bit.

p1 è un indice nella tabella esterna (o di primo livello), e p2 è l'offset all'interno della pagina indicata dalla tabella esterna. Il vantaggio è che la tabella delle pagine viene divisa in una tabella esterna, che occupa un frame, e tante altre tabelle interne, sempre di un frame, non necessariamente contigui. La tabella esterna funge, quindi, da indice.

Funzione hash

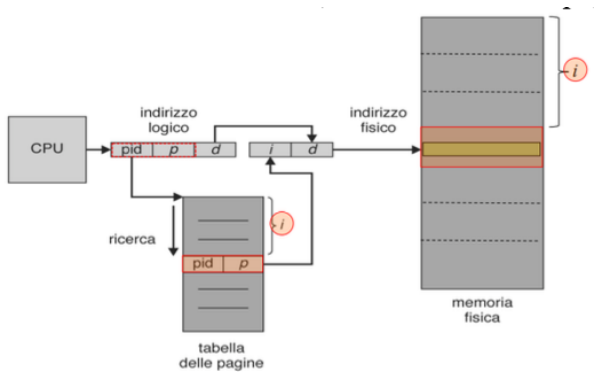
Siccome la tabella delle pagine erano troppo onerose in fase di ricerca e complesse, è stata proposta la funzione hash. Lidea è quella di prendere l'indirizzo logico, calcolare la tabella di hash e prendere tutte le pagine che hanno quel numero di hash, nel caso di più pagine con lo stesso numero di hash si avrà una lista "lista di collisione".



Si prende p l'indirizzo logico e si calcola la funzione di hash; in seguito, il numero ottenuto corrisponderà alla righa nella tabella di hash.
Si suppone che in questa riga ci siano due elementi quindi si comincia a cercare la coppia di elementi p-r (pagina-offset), quindi la r verrà sommata all'offsett "d" dell'indirizzo logico e il risultato ci riporterà all'indirizzo fisico.

Tabella della pagine invertite

L'obiettivo è di ridurre lo spazio in memoria a svantaggio della velocità.
Sostanzialmente tale tabella non contiene il numero di pagina e di frame, ma contiene numero di frame, numero di pagina e numero di processo (pid).
Ciò comporta che, mentre nella tabella delle pagine normale si utilizza il numero di pagina per trovare il numero di frame, nella tabella delle pagine invertite bisogna scandire tutto per trovare la tabella delle pagine corrispondente.



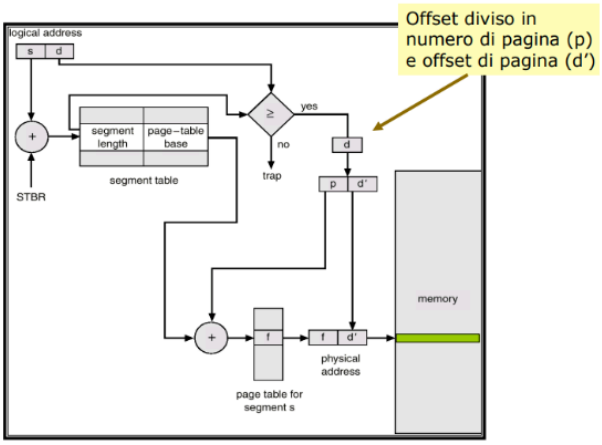
L'uso del pid è legato al fatto che la pagina con numero p si trova in molti processi e quindi è necessario distinguerle utilizzado un id (pid).
Inoltre, si indicizza per numero di frame e non per numero di pagine, ecco perché si chiama invertita. Si ricerca nella tabella l'elemento che contiene la coppia (pid,p), poi si somma i con "d" (offset) dell'indirizzo logico e si avrà l'indirizzo fisico.

Possibile Domanda esame
Differenza tra paginazione e segmentazione?
La paginazione non soffre di frammentazione esterna e l'allocazione dei frame non richiede algoritmi specifici, però come svantaggio ha la separazione tra vista utente e vista fisica della memoria.
La segmentazione ha come vantaggi:

- Consistenza tra vista utente e vista fisica della memoria. - Associazione di protezione/ condivisione ai segmenti . Mentre come svantaggi ha:
- Richiesta allocazione (dinamica) dei segmenti.
- Potenziale frammentazione esterna.

Segmentazione con paginazione

È possibile combinare i due schemi per migliorarli entrambi, la tecnica venne introdotta da multics. La soluzione utilizzata consiste nel «paginare» i segmenti: ogni segmento è suddiviso in pagine e possiede la sua tabella delle pagine (non si ha una tabella delle pagine complessiva del processo, ma ogni processo ha la sua tabella della pagine quanti sono i segmenti che lo costituiscono).
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. Tale 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.



Lo schema inizia con un indirizzo logico "s" che è posto in una tabella indicizzata, STBR (è la tabella di base dei segmenti); quindi si somma s all'indirizzo di base della tabella e si ottiene la riga richiesta.
Nella segment table ho l'indirizzo di base di questa tabella e la lunghezza del segmento. La lunghezza del segmento mi serve solo per verificare che l'offset sia minore o uguale al segmento stesso (se la lunghezza è 3000, ovviamente "d" deve essere minore o uguale a 3000).

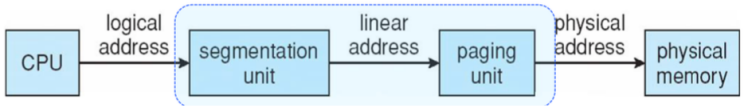
d viene dopo diviso in due parti, uno rappresenta il numero di pagina mentre l'altro rappresenta l'offset.
"d' " corrisponde all'offset finale. La coppia f e d' ci riporterà all'indirizzo fisico in memoria corrispondente.

Esempio in numero
segmento lungo 10000
offset 3500
Lunghezza pagina 1000
Quante pagine mi servono per mettere un segmento?
10000/1000 = 10 pagine L'offset è di 3500, quindi si dovrà andare nella 4 pagina nella posizione 500.
Questo si calcola facendo offset diviso il modulo della lunghezza delle pagine.

Architetture Intel

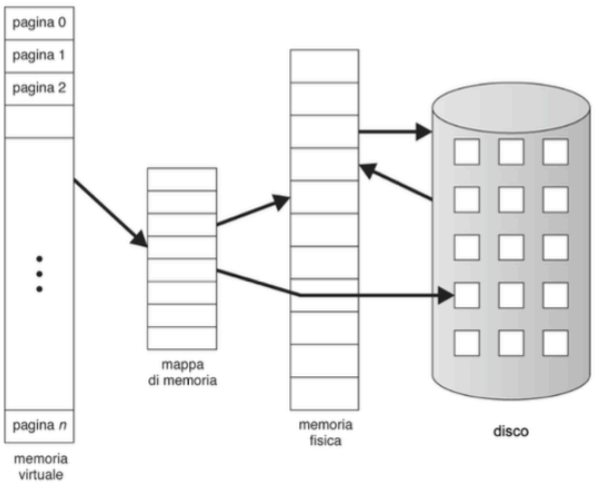
I dispositivi moderni utilizzano una segmentazione a paginazione mista, a partire dall'Intel a 32 bit. In tali architetture:

- La CPU genera indirizzi logici, che confluiscono nell'unità di segmentazione, la quale produce indirizzi lineari (numero unico).
- L'indirizzo lineare passa all'unità di paginazione, che a sua volta genera l'indirizzo fisico (offset) relativo alla memoria centrale.
- Le unità di segmentazione e di paginazione formano un equivalente dell'unità di gestione della memoria (MMU).



MEMORIA VIRTUALE

Tecnica che realizza la separazione della memoria logica dalla memoria fisica al fine di "donare" all'utente della macchina una quantità di memoria maggiore di quella fisicamente posseduta dalla macchina. Essa, infatti, "fa credere" al calcolatore di avere a disposizione più memoria di quella realmente posseduta.
Esistono varie tecniche in questo ambito:



•Solo una parte del programma è caricato 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 meglio tra più processi.
•Permette una più efficiente creazione dei processi.

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

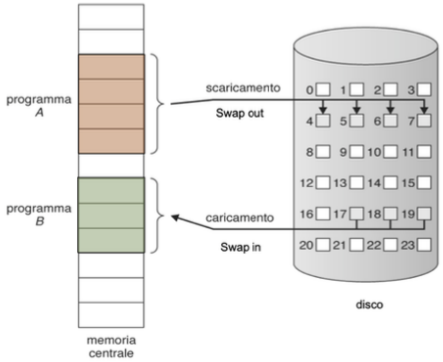
Un programma è caratterizzato da:

- Una serie di pagine virtuali (nel senso che tale memoria non deve obbligatoriamente coincidere con la memoria fisica).
- Una mappa di memoria, che conserva gli indirizzi delle pagine.
- Disco (backing store), luogo in cui finiscono le pagine che è necessario rimuovere (momentaneamente) dalla RAM per far spazio a nuove pagine.

In tale situazione, parte delle sono caricati nella RAM e altri sul disco ed esiste un file (detto file paging) che contiene le pagine che non “entrano” nella RAM.

PAGINAZIONE SU RICHIESTA

Nella paginazione su richiesta si porta una pagina in memoria solo quando serve. Ciò comporta i seguenti vantaggi:



•Riduzione I/O dal disco, perché si evita di copiare le pagine quando non necessario.
•Minore memoria necessaria.
•Risposta più veloce, in quanto si riduce il tempo di content switch.
•Aumento del livello di multiprogrammazione, poiché si ha un maggior numero di processi in esecuzione contemporaneamente nello stesso quantitativo di RAM.

Ogni volta che nel codice si ha la necessità di riferirsi ad una pagina si accede alla tabella delle pagine:

- Se la pagina alla quale si vuole fare riferimento non è valida si verifica un "abort" (non verrà affrontato).
- Se la pagina non è caricata in memoria si manifesta un "page fault" (fallimento non fatale), ossia la pagina non è ancora stata caricata in RAM ed è necessario il suo trasferimento in memoria per risolvere il problema.

Ricorda: le pagine possono essere caricate in memoria tramite eventi di swap in (caricamento) e rimosse tramite swap out (scaricamento). A volte può essere caricato/scaricato un intero programma; spesso, però, vengono caricate/scaricate porzioni di un programma. La politica minimale prevede che ogni volta venga caricata/scaricata una sola pagina per volta, soprattutto il caricamento; ovviamente se un programma ha la necessità di caricare più pagine contemporaneamente può farlo.

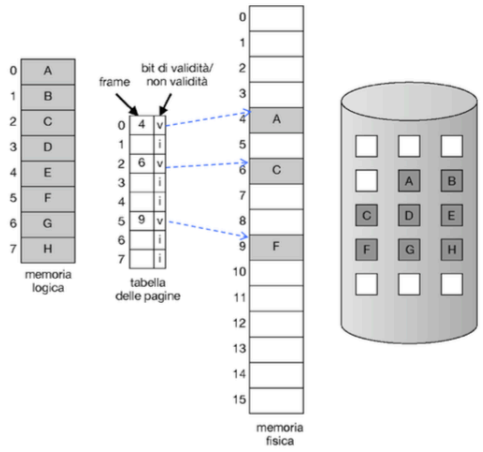
Ad ogni entry della tabella è associato un bit di validità (1 ⇒ in memoria, 0 ⇒ non in memoria ma su disco)
Inizialmente il bit di validità è uguale a 0 per ogni entry.
Esempio di tabella delle pagine:

Frame #	Bit valido- nonvalido
	1
	1
	1
	1
	0
	1
	1
	0
	0

Tabella delle pagine

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

Nella tabella delle pagine abbiamo:
•L'indice di pagina o frame, legato all’indirizzo fisico della pagina.
•Il bit di validità, con il compito di indicare se la pagina è in memoria oppure no.
Ad esempio, in questo caso, se ho una pagina con p=5 si verifica un page fault poiché il bit di memoria è uguale a 0. È necessario caricare la pagina nella memoria per poter validare il bit a 1 e proseguire nell’esecuzione del programma.



Qui vediamo, invece, un caso (tipico di quasi tutti i calcolatori) in cui alcune pagine sono caricate sulla memoria mentre altre sono depositate sul disco, e quindi non valide. In particolare, la memoria logica è costituita da 8 pagine (da 0 a 7) e nella memoria fisica abbiamo 3 pagine caricate in memoria (con bit di validità 1 o v) e 5 su disco (con bit di validità 0 o i).

Se si tenta di accedere ad una pagina non in memoria si verifica un page fault (pagina mancante). Il processo di risoluzione di tale errore prevede:

1. II S.O. controlla in una tabella interna del processo:
   - Se il riferimento non è corretto => abort.
   - Se il riferimento è corretto occorre caricare la pagina.
2. Si trova un frame libero nella frame-free list.
3. Si carica la pagina nel frame libero.
4. Si aggiorna la tabella e il bit di validazione = 1.
5. Viene riavviata l'esecuzione: la pagina diventa quella più recente.

Prestazioni della paginazione su richiesta

- Probabilità di page fault: $0 \le p \le 1$
  - se $p = 0$ nessun page fault
  - se $p = 1$ ogni accesso provoca un page fault
  - se $p = 0.5$ metà degli accessi provocano un page fault.
- Tempo di accesso effettivo (***TAE***)

$$TAE = ((1 - p) * tma) + (p * tpf)$$

tma = tempo di memory access
tpf = tempo di gestione del page fault

- Tempo di accesso in memoria = 100 nanosecondi
- Tempo di gestione di un page fault = 25 millisec = 25.000.000 nanosec

TAE = ((1 – p) * 100) + (p * (25.000.000))
= 100 + (25.000.000-100) x p *(nanosecondi)*

Se p= 0.4

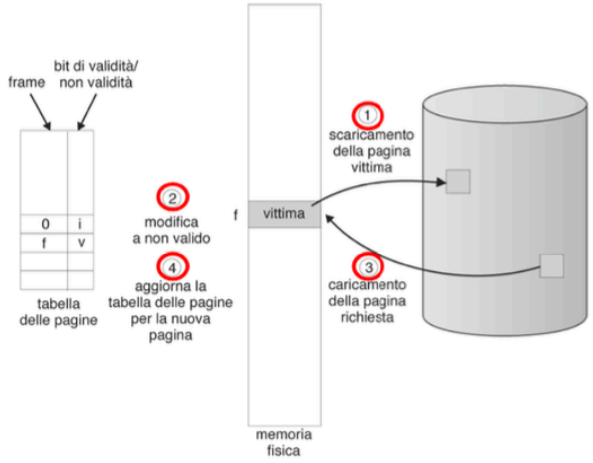
TAE = (1 – 0.4) x 100 + 0.4 x (25.000.000)
= 100 + (25.000.000-100) x 0.4
= 10.000.060 *(nanosecondi)* → 10 millisec

E' importante tenere basso il tasso di page fault, altrimenti il tempo effettivo d'accesso aumenta, rallentando molto l'esecuzione del processo.

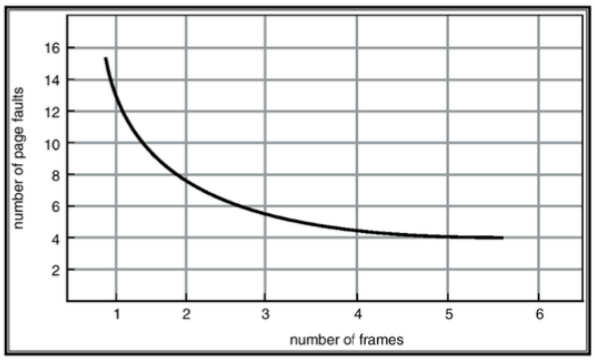
Per calcolare le prestazioni della paginazione su richiesta si fa riferimento alla probabilità di page fault (p). Definiti il tempo di memory access (tma) e il tempo di gestione del page fault (tpf), possiamo dire che il tempo di accesso effettivo medio (TAE) è dato dalla somma del tempo di memory access nel caso in cui non c'è alcun page fault [(1- p)*tma] con il tempo di gestione del page fault per la probabilità di page fault [p*tpf].
Logicamente, più page fault si verifica più il TAE aumenta, con conseguente decadimento delle prestazioni. Da tale equazione, inoltre, si dimostra che la RAM è in grado di gestire più risorse rispetto allo spazio occupabile ma al prezzo di un rallentamento generale delle prestazioni.
Osservare l'esempio nella slide:

SOSTITUZIONE DELLE PAGINE

Quando occorre caricare una pagina e non c'è un frame libero si può usare la sostituzione delle pagine. Sono necessari due trasferimenti di pagine, uno fuori e uno dentro la memoria È molto importante trovare una pagina in memoria che non è usata per portarla sul disco (swap out).
Occorre usare un algoritmo apposito in quanto lo swap out di una pagina che deve essere riusata e, quindi, ricaricata a breve termine sovraccarica la CPU fino, in caso di un ripetuto swap out e swap in, a portare ad un suo rallentamento (tempo di esecuzione dell'ordine dei secondi).



Nel caso si ha la necessità di caricare in memoria una pagina in sola lettura, questo sovraccarico si può ridurre usando un modify (dirty) bit per ridurre il costo del trasferimento delle pagine - solo le pagine modificate vengono (ri)scritte sul disco; tale bit cambia valore (generalmente passa da 0 a 1) quando la pagina salvata sul disco è stata modificata rispetto all'ultima volta in cui è stata caricata in memoria, così da indicare la necessità di ricaricare in memoria la nuova versione.



È auspicabile evitare che una pagina che servirà a breve sia portata sul disco prima di essere richiesta nuovamente; tuttavia, spesso non si conosce la pagina che sarà necessario caricare successivamente. Risulta essenziale la scelta di un algoritmo che dia il numero minimo di page fault.

Bisogna trovare:
- La locazione della pagina richiesta sul disco.
- Trovare un frame libero:
    - Se esiste viene usato;
    - Se non c'è un frame libero seleziona un frame vittima secondo un algoritmo di sostituzione. Scrive la pagina vittima sul disco e aggiorna le tabelle.
- La pagina richiesta nel frame liberato viene letta e le tabelle vengono aggiornate.
- Si riavvia il processo

Questa operazione di sostituzione del frame può essere (in riferimento al processo di appartenenza delle pagine coinvolte):
- Locale, quando la pagina sostituita appartiene allo stesso processo della nuova pagina inserita.

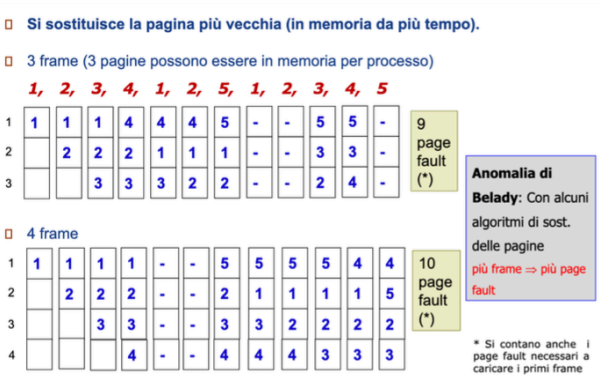
- Globale, quando la nuova pagina inserita e quella sostituita appartengono a due processi qualsiasi.

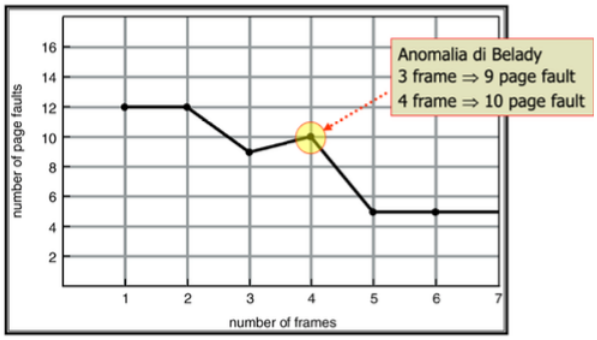
Algoritmi di sostituzione delle pagine

Il criterio di scelta è quello di minimizzare il numero di page fault, al fine di ridurre il tempo di accesso medio. La scelta ricade su tre algoritmi:
- FIFO (First In First Out)
- Ottimale
- LRU (Least Recently Used)

Si valuta gli algoritmi eseguendoli su una particolare sequenza (stringa) di riferimenti alla memoria e calcolando il numero di page fault che si verificano.
Idealmente, all'aumentare del numero di frame si ottiene una diminuzione del numero di page fault, come mostrato nel grafico. Nonostante tale andamento, può verificarsi una situazione in cui l'aumento del numero di frame porta all'aumento di page fault (anomalia di Belady).

ALGORITMO FIFO

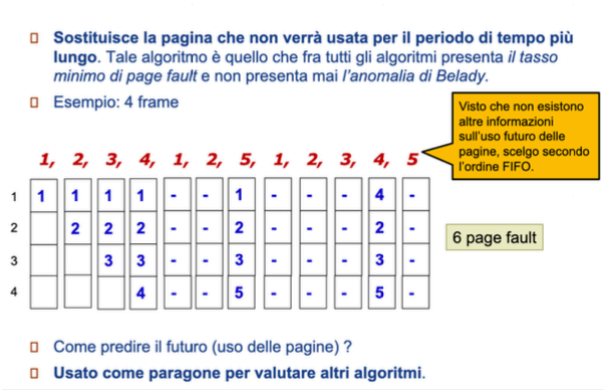




Questo algoritmo prevede che ad essere sostituita sia la pagina "più vecchia", cioè che si trova in memoria da più tempo (la prima caricata). Nell'esempio abbiamo prima il caso 3 frame, in cui 3 pagine possono essere messe in memoria per processo. Si vede che vengono prima inserite in successione le pagine 1, 2 e 3 nei tre free-frame a disposizione.
Nota: tra i page fault si contano anche gli inserimenti (caricamenti) iniziali. I trattini indicano che la pagina è già caricata e non avviene alcuna modifica nella memoria.
Arrivati alla quarta pagina (pagina 4), i frame risultano tutti occupati; viene così sostituita la pagina 1 (la prima pagina ad essere caricata) con la 4, risolvendo un page fault. In modo analogo si esegue per tutte le altre pagine da inserire fino alla terminazione della sequenza. Il numero di page fault totali è 9.
Aumentando il numero di frame a 4 ci aspetteremmo una diminuzione del numero di page fault; tuttavia, osserviamo nell'esempio che si ha un aumento (10 page fault), proprio dovuto all'anomalia di Belady.

ALGORITMO OTTIMALE

Tale algoritmo sostituisce la pagina che non verrà usata per il periodo più lungo. È l'algoritmo che prevede il numero minimo di page fault e non presenta mai l'anomalia di Belady. Si basa sulla predizione delle pagine successive alla corrente, così da sostituire (swap out) la pagina che verrà utilizzata "nel tempo più lontano" tra tutte quelle caricate al momento.



Nell'esempio, si ha il caricamento delle prime 4 pagine nei frame (4 page fault); arrivati alla parte della sequenza che prevede il caricamento della pagina 5, l'algoritmo osserva le pagine successive della sequenza e, poiché tra tutte il 4 è quella che sarà richiamata più tardi, andrà a sostituire la pagina 4 con la 5, così fino a terminare la sequenza. I fault page che si sono verificati sono 6. Il limite di tale algoritmo è proprio la necessità di conoscere a priori l'ordine di caricamento delle pagine, spesso non noto prima del momento del caricamento stesso; visto che non esistono altre informazioni sull'uso futuro delle pagine, si sceglie secondo ordine FIFO. Per tale motivo, l'algoritmo ottimale viene usato come paragone per valutare altri algoritmi.

ALGORITMO LEAST RECENTLY USED (LRU)

Questo algoritmo prevede la sostituzione della pagina che non è stata usata per il periodo più lungo. Fa, così, una predizione "statistica" considerando la pagina meno usata recentemente come quella "meno probabile" da caricare a breve termine. È un'approssimazione dell'algoritmo ottimale (futuro
in base al passato recente).
Nell'esempio, si ha il caricamento delle 4 pagine frame disponibili; quando si arriva al caricamento della pagina 5, l'algoritmo guarda a ritroso la sequenza delle pagine precedentemente caricate e, poiché la sequenza ottenuta è 2-1-4-3, comprende che la pagina caricata meno di recente è la 3, che viene così sostituita dalla 5. In modo analogo si ha la sostituzione della pagina 4 con la 3 (sequenza 2- 1-5-4). Così fino a terminare la sequenza di caricamento. I page fault verificatisi sono 8. Il principio di tale algoritmo è, quindi, quello di lasciare cariche in memoria le pagine richiamate di frequente.