

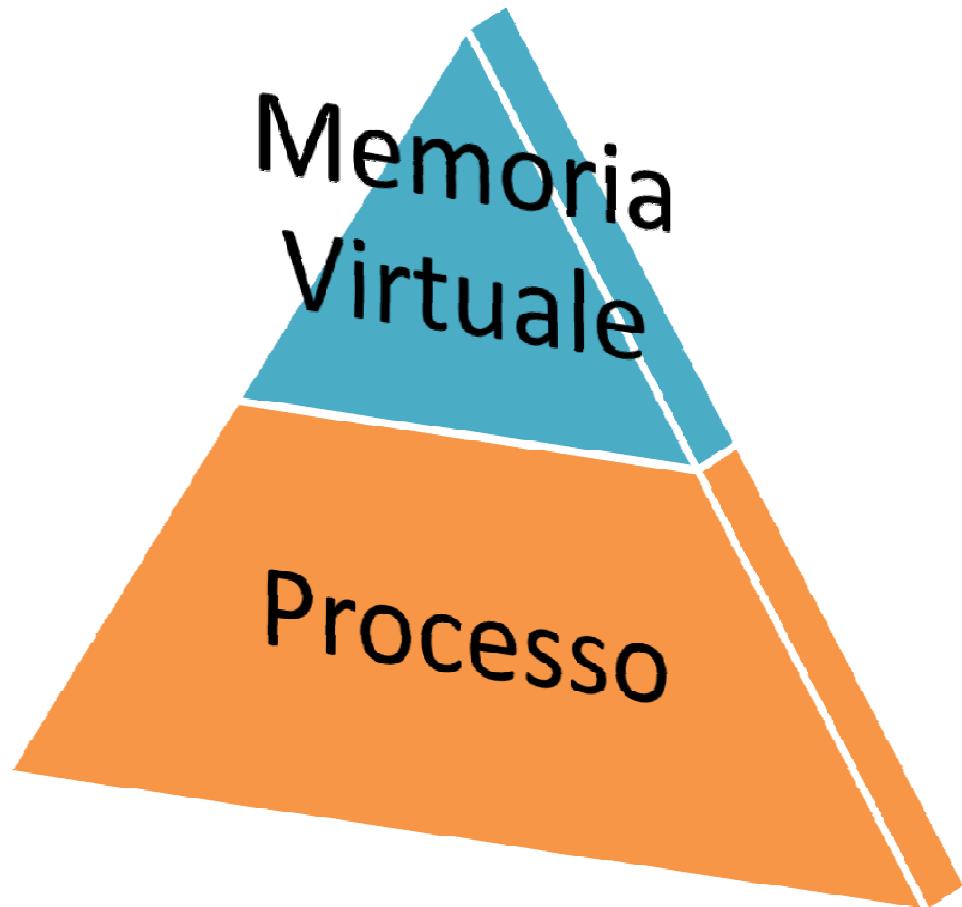
Architettura degli elaboratori

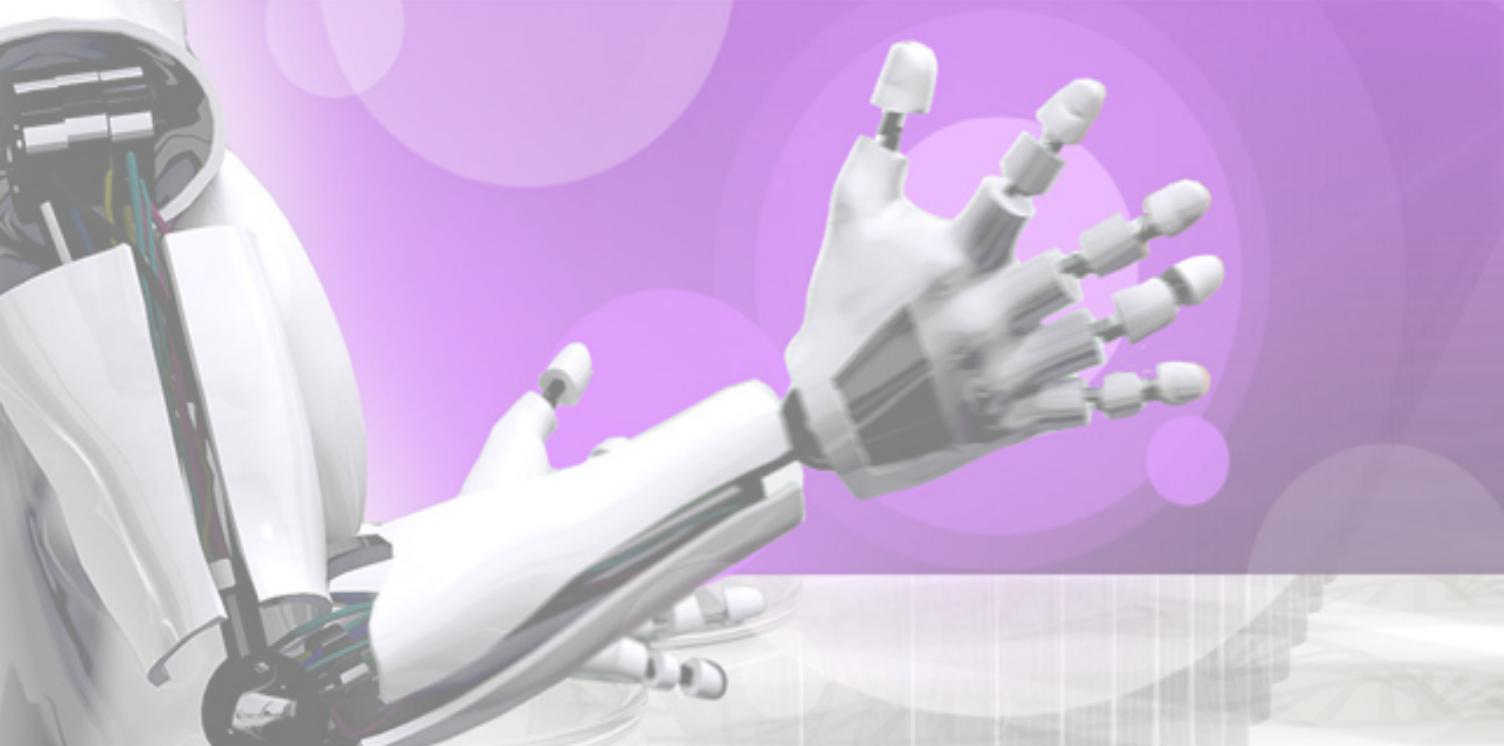
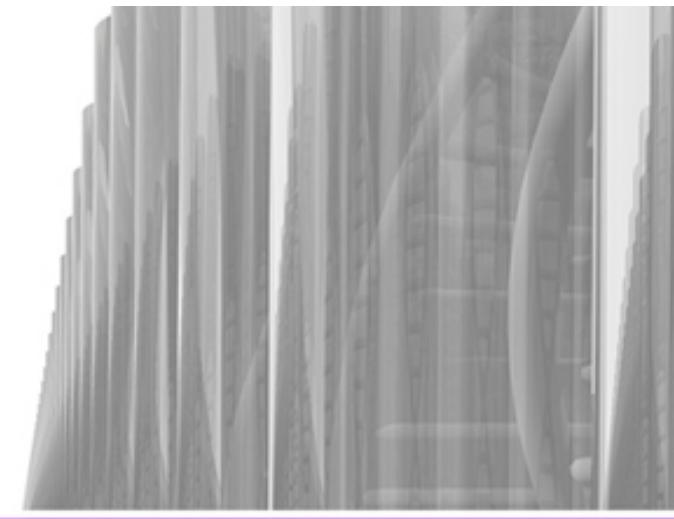
MEMORIA VIRTUALE

Dott. Franco Liberati

ARGOMENTI DELLA LEZIONE

- Memoria Virtuale





Processo



PROCESSO

Definizione

- Un programma presente in memoria principale, ed in corso di esecuzione, è detto **processo**
- Ogni processo ha un blocco di informazioni associate che lo descrivono il **PCB** (*process control block*) costituito da:
 - Stato del processo (new, ready, running, waiting, halted)
 - Program Counter: l'indirizzo in cui punta la prossima istruzione da eseguire)
 - Registri CPU (informazioni salvate all'atto di una interruzione)
 - Informazioni sullo scheduling di CPU (es.: la priorità del processo)
 - **Informazioni utili per la gestione della memoria: i registri base e limite e la tabelle delle pagine**
 - Informazioni di contabilità (es.: il tempo utilizzato, lo slot temporale assegnato,...)
 - Informazione I/O (es.: l'elenco dei file aperti, le risorse I/O richieste)

Identificatore	Stato del processo
Numero del processo	
Program counter	
Registri	
Limiti di memoria	
...	
Elenco file aperti	



PROCESSO

Definizione

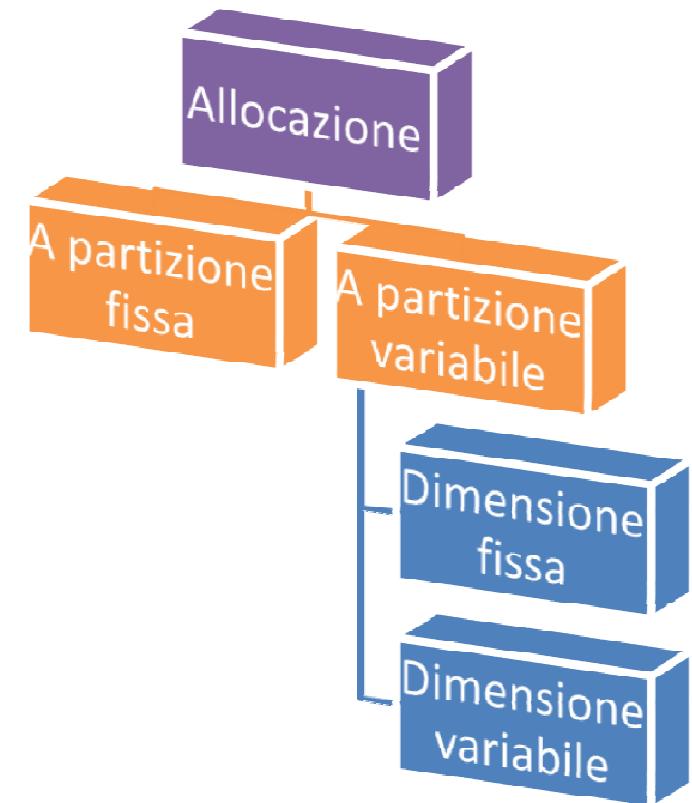
- La **memoria principale** è una risorsa limitata, che deve essere allocata in modo efficiente e adeguata, cioè deve contenere i processi e questi devono essere posizionati in aree sufficientemente grandi
- Deve contenere sia i processi utente che quelli utili al sistema (il sistema operativo) e pertanto è suddivisa in due parti:
 - La parte residente del sistema operativo è generalmente memorizzata nella memoria alta (nelle architetture moderne), insieme al vettore degli interrupt
 - I processi utente sono memorizzati nella memoria bassa



PROCESSO

Allocazione Memoria

- ❑ Allocazione di memoria
 - ❑ A partizione singola
 - ❑ Lo spazio di memoria riservato ai processi è un blocco unico
 - ❑ A partizioni multiple
 - ❑ La memoria è suddivisa in partizioni logiche
 - ❑ A dimensione fissa
 - ❑ A dimensione variabile



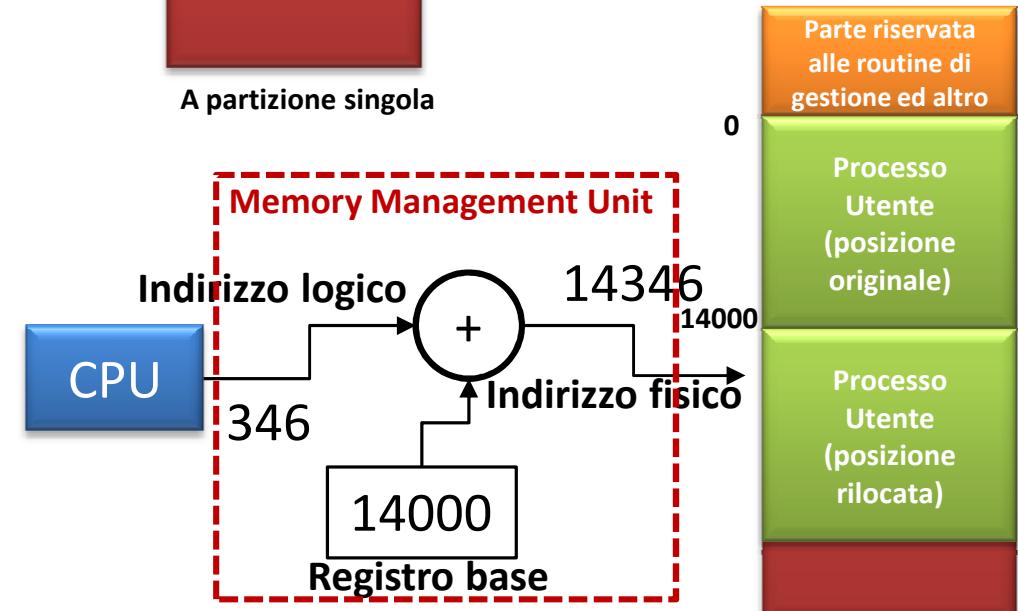


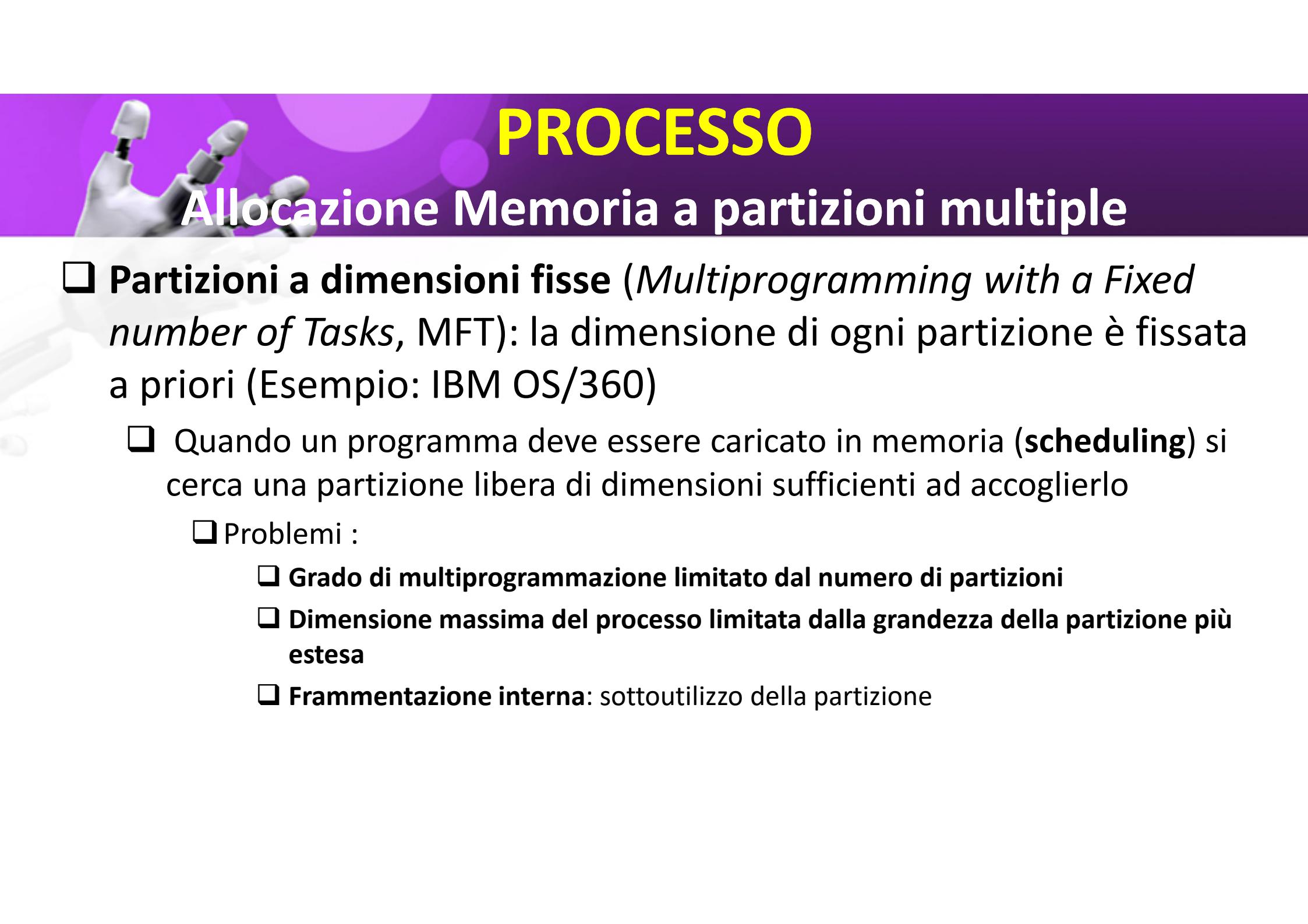
PROCESSO

Allocazione Memoria a partizione singola

□ Allocazione a partizione singola

- La parte di memoria disponibile per l'allocazione dei processi non è partizionata
- Un solo processo alla volta può essere allocato in memoria (questo avveniva nelle prime architetture e con il sistema operativo MS-DOS) ad un punto prestabilito
- Con una modifica hardware (il **registro di rilocazione** contenuto nel Memory Management Unit) si consentì di spostare il processo dal punto prestabilito introducendo la **rilocazione dinamica**





PROCESSO

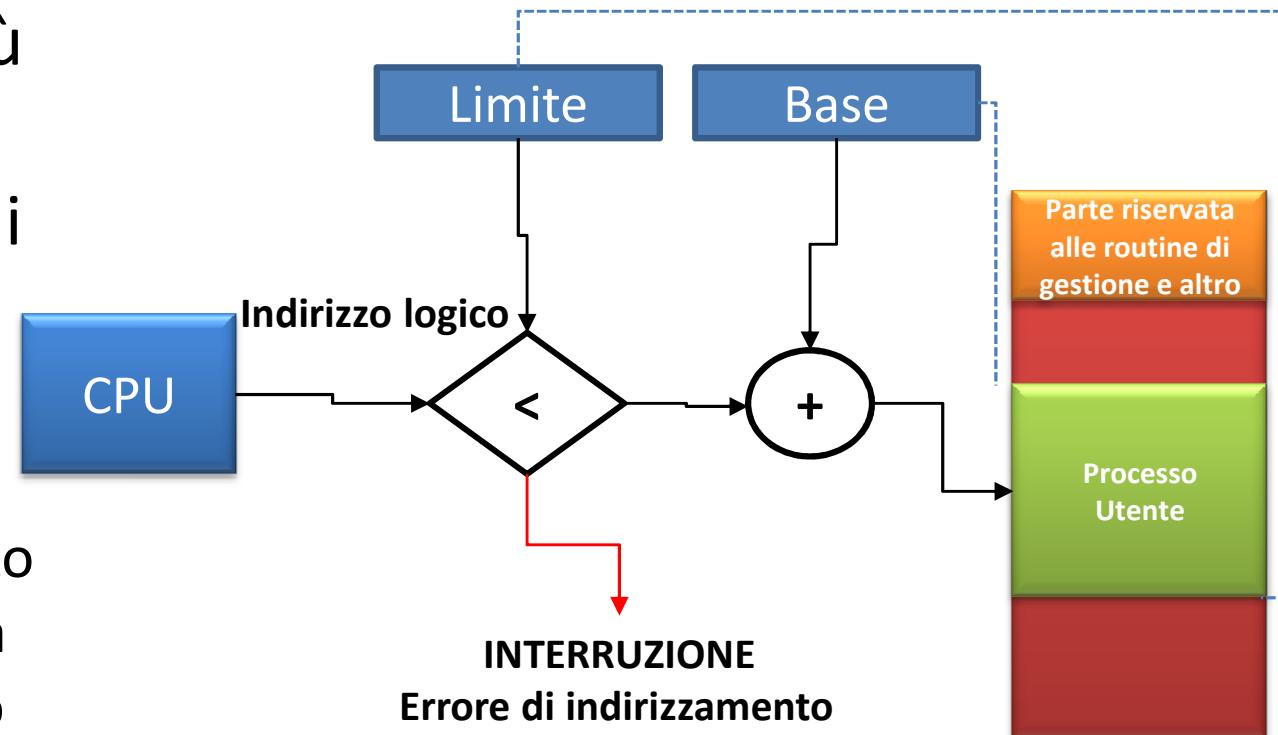
Allocazione Memoria a partizioni multiple

- **Partizioni a dimensioni fisse** (*Multiprogramming with a Fixed number of Tasks*, MFT): la dimensione di ogni partizione è fissata a priori (Esempio: IBM OS/360)
 - Quando un programma deve essere caricato in memoria (**scheduling**) si cerca una partizione libera di dimensioni sufficienti ad accoglierlo
 - Problemi :
 - Grado di multiprogrammazione limitato dal numero di partizioni
 - Dimensione massima del processo limitata dalla grandezza della partizione più estesa
 - **Frammentazione interna**: sottoutilizzo della partizione

PROCESSO

Allocazione Memoria a partizioni multiple (MTF)

- Poiché sono presenti più processi è opportuno disporre di un sistema di protezione
 - **Registro base e registro limite** che indicano rispettivamente l'indirizzo di inizio del processo e la dimensione del processo





PROCESSO

Allocazione Memoria a partizioni multiple

- **Partizioni a dimensioni variabili** (*Multiprogramming with a Variable number of Tasks*, MVT): ogni partizione è allocata dinamicamente in base alla dimensione del processo da allocare (moderni calcolatori)
 - Vantaggi (rispetto a MFT)
 - Il grado di multiprogrammazione è variabile
 - La dimensione massima dei processi è limitata dalla disponibilità di spazio fisico
 - Si elimina la frammentazione interna: ogni partizione è dell'esatta dimensione del processo
 - Problemi
 - Scelta dell'area da allocare
 - **Frammentazione esterna** (risolvibile con la **compattazione**, a svantaggio delle prestazioni)



PROCESSO

Frammentazione

□ **Frammentazione interna**

- La memoria richiesta dal processo è maggiore della dimensione della partizione occupabile: la parte restante richiede l'impiego di una partizione che non è impiegata completamente

□ **Frammentazione esterna**

- Si verifica quando è disponibile lo spazio necessario a soddisfare una richiesta, ma non è contiguo

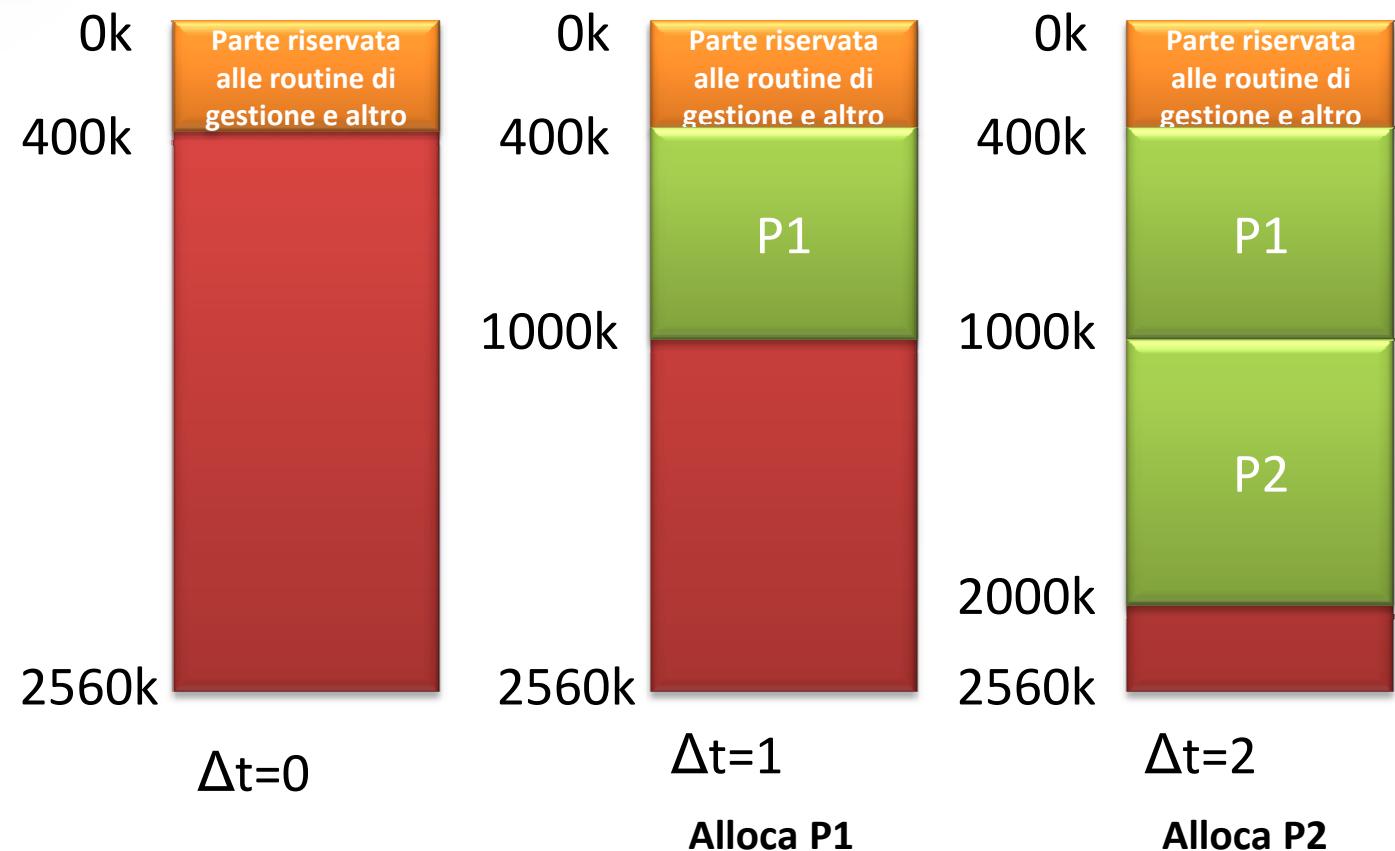
- Si può ridurre la frammentazione esterna con la **compattazione**

- Si spostano i contenuti della memoria per avere tutta la memoria libera contigua a formare un grande blocco
 - La compattazione è possibile solo con la rilocazione dinamica perché viene effettuata a runtime
 - I processi con I/O pendenti non possono essere spostati o si deve garantire che l'I/O avvenga solo nello spazio kernel

PROCESSO

Frammentazione esterna (esempio)

Processo	Dimensione	Tempo di permanenza in memoria
P1	600K	10
P2	1000K	5
P3	300K	20
P4	700K	8
P5	500K	15

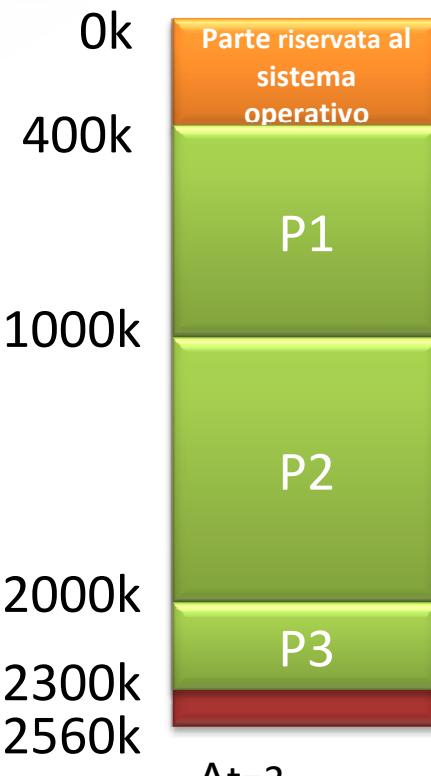


PROCESSO

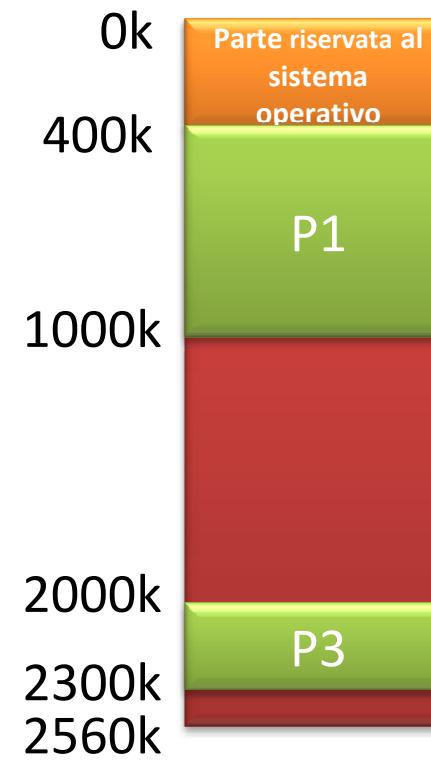
Frammentazione esterna (esempio)

Schema

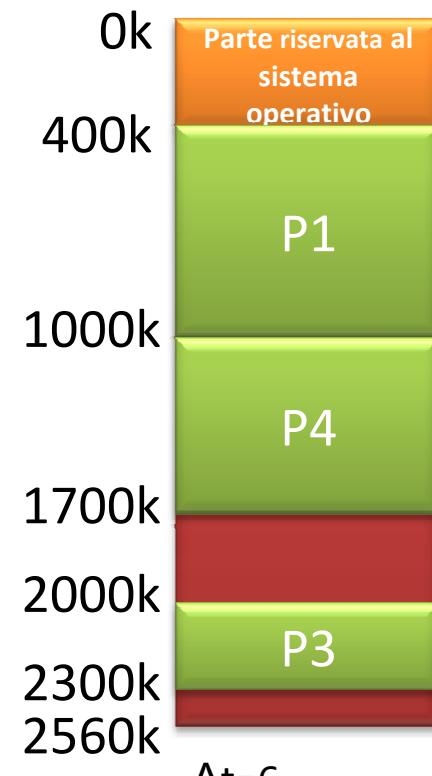
Processo	Dimensione	Tempo di stazionamento in memoria
P1	600K	10
P2	1000K	5
P3	300K	20
P4	700K	8
P5	500K	15



Alloca P3
 $\Delta t=4$
 P4 non può essere caricato



Termina P2



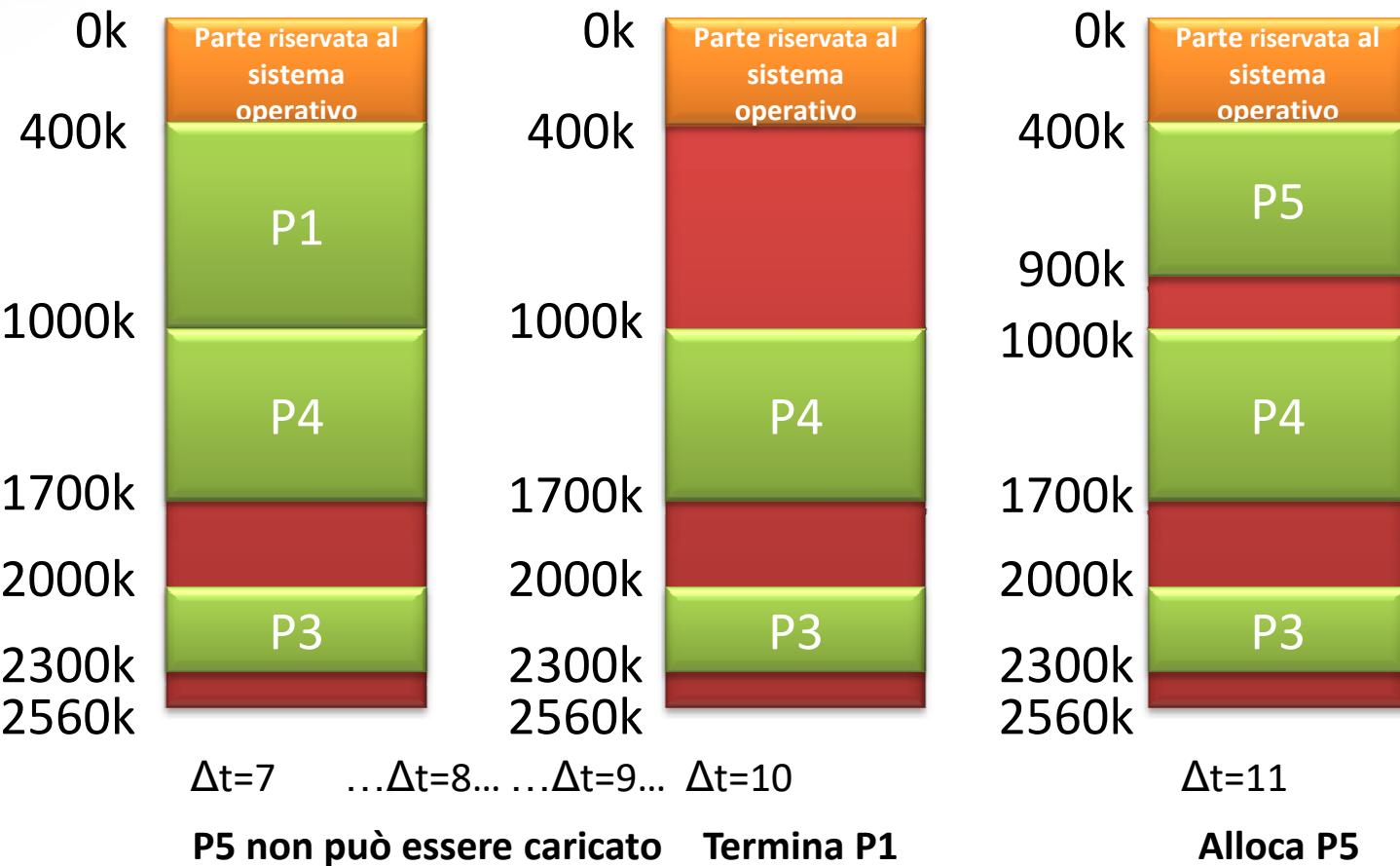
Alloca P4

PROCESSO

Frammentazione esterna (esempio)

Schema

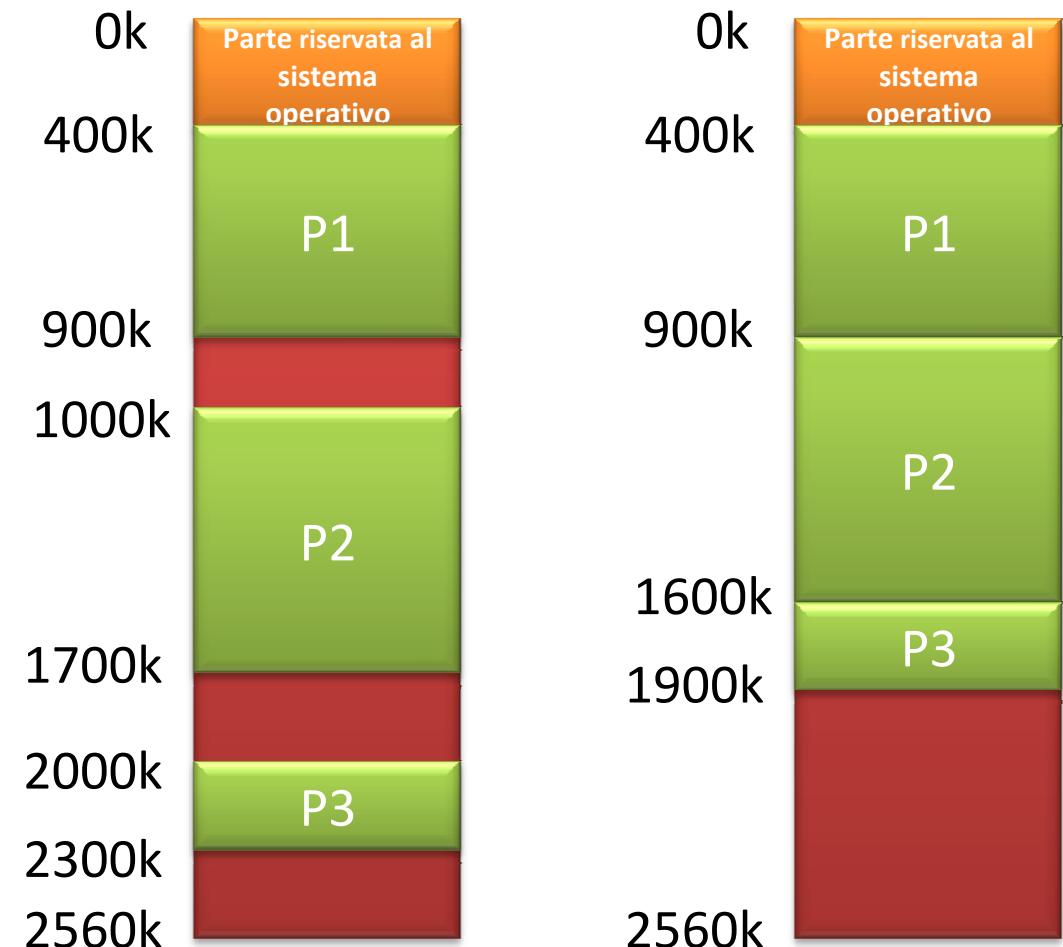
Processo	Dimensione	Tempo di stazionamento in memoria
P1	600K	10
P2	1000K	5
P3	300K	20
P4	700K	8
P5	500K	15





PROCESSO Compattazione

- Si può ridurre la frammentazione esterna con la **compattazione**
 - Si spostano i processi per avere memoria libera contigua (a formare un grande blocco)
 - La compattazione è possibile solo nelle architetture dove è consentita la **rilocazione dinamica** perché è effettuata a runtime





Memoria virtuale



MEMORIA VIRTUALE

Generalità

- La **memoria virtuale (Virtual Memory, VM)** è una tecnica che consente di eseguire programmi in esecuzione (processi) che possono anche non essere contenuti totalmente all'interno della memoria fisica
- Il vantaggio principale che offre questa tecnica è quello di permettere che i programmi siano più grandi della memoria centrale (la RAM) che si ha disposizione; questo lo si fa astraendo la memoria centrale ed estendendola in una memoria virtuale (MV) grande quanto si vuole
- Sebbene esista la tecnica di **overlay**, tuttavia si privilegia una implementazione dell'uso della memoria virtuale perché solleva il programmatore dal gestire quale porzione di programma deve entrare e quale uscire dalla memoria fisica di volta in volta
- Oltre ad aumentare la memoria disponibile per ogni processo, la MV permette di aumentare il numero dei processi caricati in memoria ed eseguiti in intervalli temporali assegnati, la **multiprogrammazione**, e infine (se progettata correttamente) di ridurre il tempo necessario alle operazioni di **trasferimento di processi dalla memoria di massa** (dove sono archiviati i programmi) **alla memoria centrale**



MEMORIA VIRTUALE

Paginazione

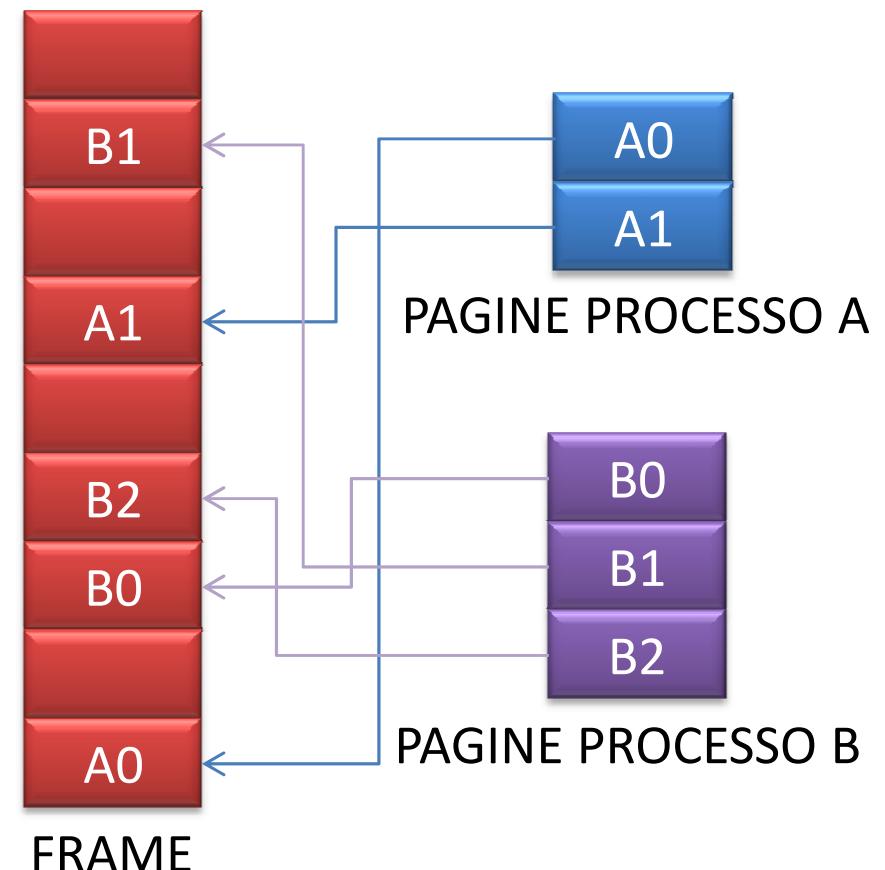
- ❑ Prima di procedere nella spiegazione della MV è necessario fare un riferimento alla tecnica della **paginazione**
- ❑ La paginazione è un metodo di gestione della memoria che fa sì che lo spazio degli indirizzi fisici di un programma in esecuzione **non sia necessariamente contiguo**
- ❑ In questo modo si **elimina il problema della frammentazione esterna** e la dispendiosa attività di **compattazione** dei processi in memoria



MEMORIA VIRTUALE

Paginazione

- La memoria fisica è suddivisa logicamente in porzioni – tutte della stessa dimensione - dette **frame** (o **pagina fisica**)
 - Di solito, un frame corrisponde ad un blocco
- Mentre un programma è suddiviso in parti della stessa dimensione dei frame ciascuno dei quali è chiamato **pagina** (o **pagina logica**)
- Ciascuna pagina può quindi essere caricata in un frame presente nella memoria fisica
- In questo modo si sfrutta la RAM a disposizione nella maniera più flessibile, senza essere obbligati a inserire i programmi in porzioni contigue
- Con questa tecnica pertanto non è necessario ricorrere alla compattazione (c'è frammentazione interna, ma è comunque un problema trascurabile)





MEMORIA VIRTUALE

Paginazione

Osservazione. La dimensione di una pagina, e quindi quella del frame è stabilita dall'architettura del calcolatore ed è, in generale, compresa tra 512byte-16Mb (la dimensione è pari ad una potenze di due in modo da facilitare l'indirizzamento)

Se, ad esempio, le pagine sono di 2048byte, un processo che occupa 72.766 byte necessita di 35 frame e 1086 byte. In questo caso sono assegnati 36 frame, con una **frammentazione interna** (all'ultimo frame) di 962 byte

I moderni calcolatori hanno frame da 4096byte o 8192byte



MEMORIA VIRTUALE

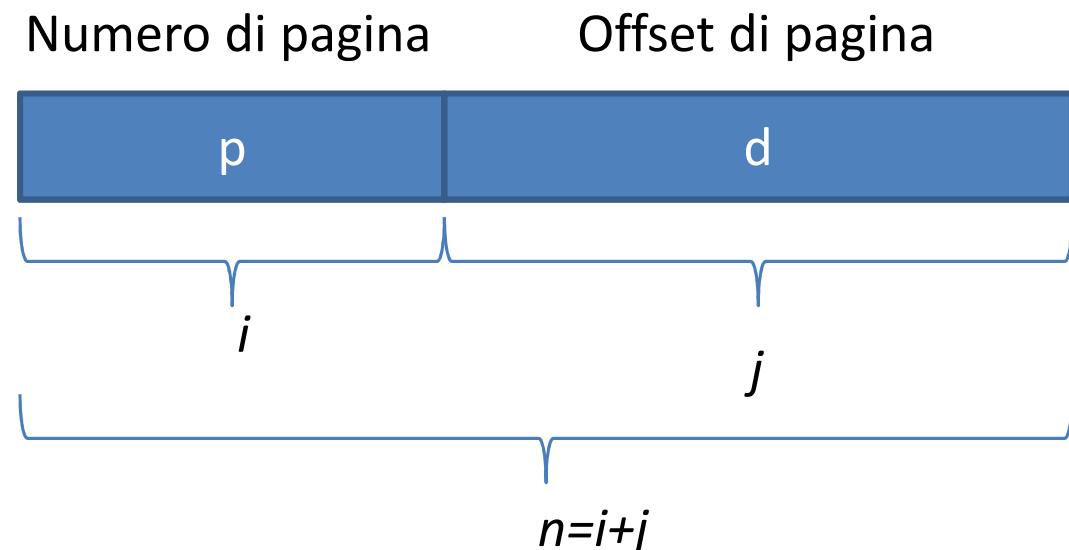
Paginazione

- ❑ Per fare in modo che il programma, anche se non contiguo, possa essere eseguito si determina uno spazio degli **indirizzi logici** totalmente separato dallo spazio degli **indirizzi fisici** (le due tipologie di indirizzi, in generale, hanno dimensione diversa) **[Schema di traduzione degli indirizzi]**
- ❑ Per eseguire un programma di dimensione n pagine, è necessario trovare nel corso della sua esecuzione n frame liberi **[Individuazione dei frame liberi]**

MEMORIA VIRTUALE

Schema di traduzione degli indirizzi

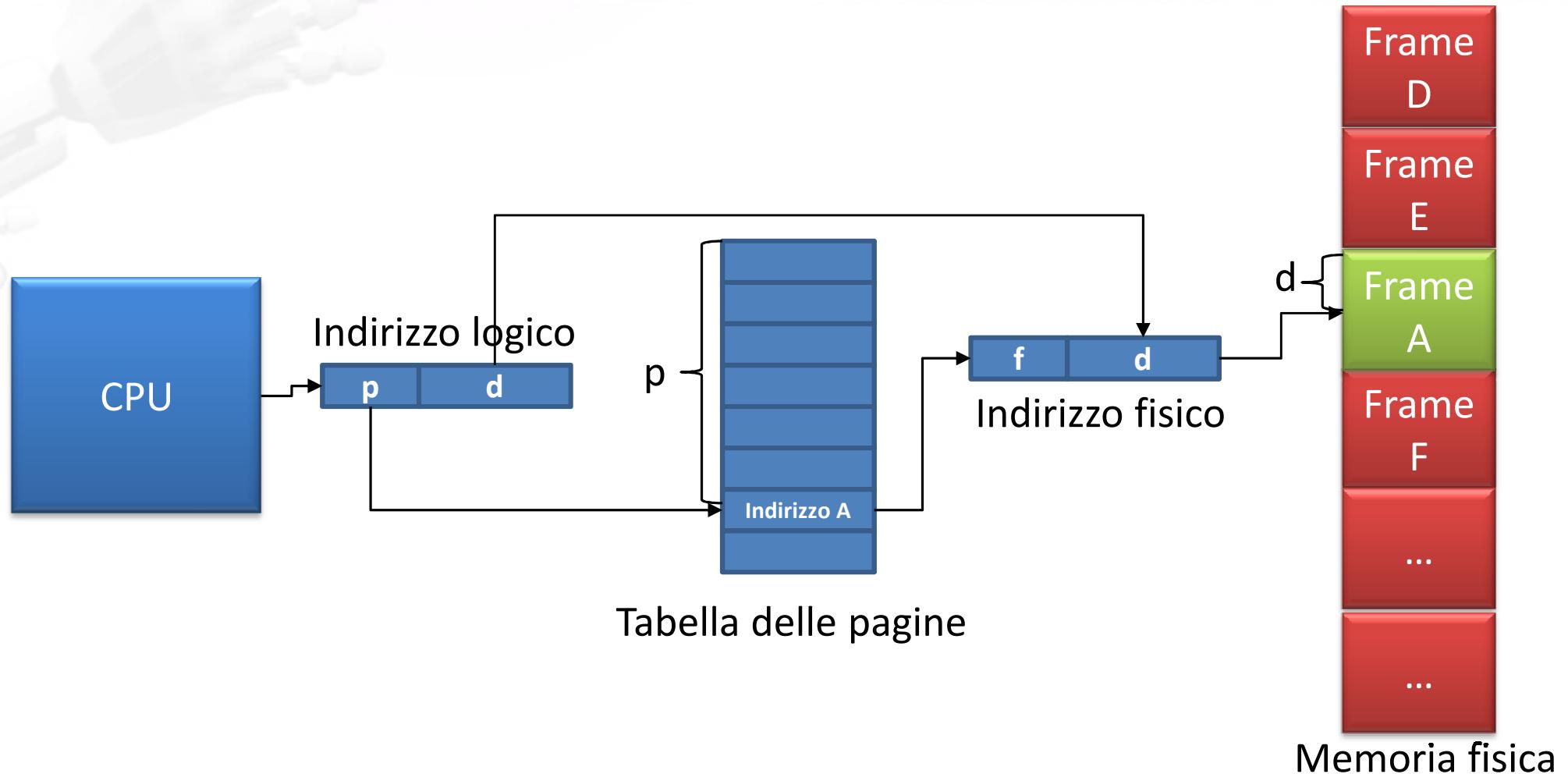
- Nello schema di traduzione degli indirizzi si impiega una **tabella delle pagine** per tradurre gli indirizzi logici negli indirizzi fisici corrispondenti
- L'**indirizzo logico** generato dalla CPU è suddiviso in due campi:
 - **Numero di pagina** (p)
 - impiegato come indice in una tabella delle pagine che contiene l'indirizzo iniziale (**indirizzo base** o *init address*) di ciascun pagina nella memoria fisica
 - **Offset nella pagina** (d)
 - È combinato con l'indirizzo base per definire l'indirizzo fisico da raggiungere



Spazio logico 2^n
Numero di pagine 2^i
Dimensione della pagina 2^j

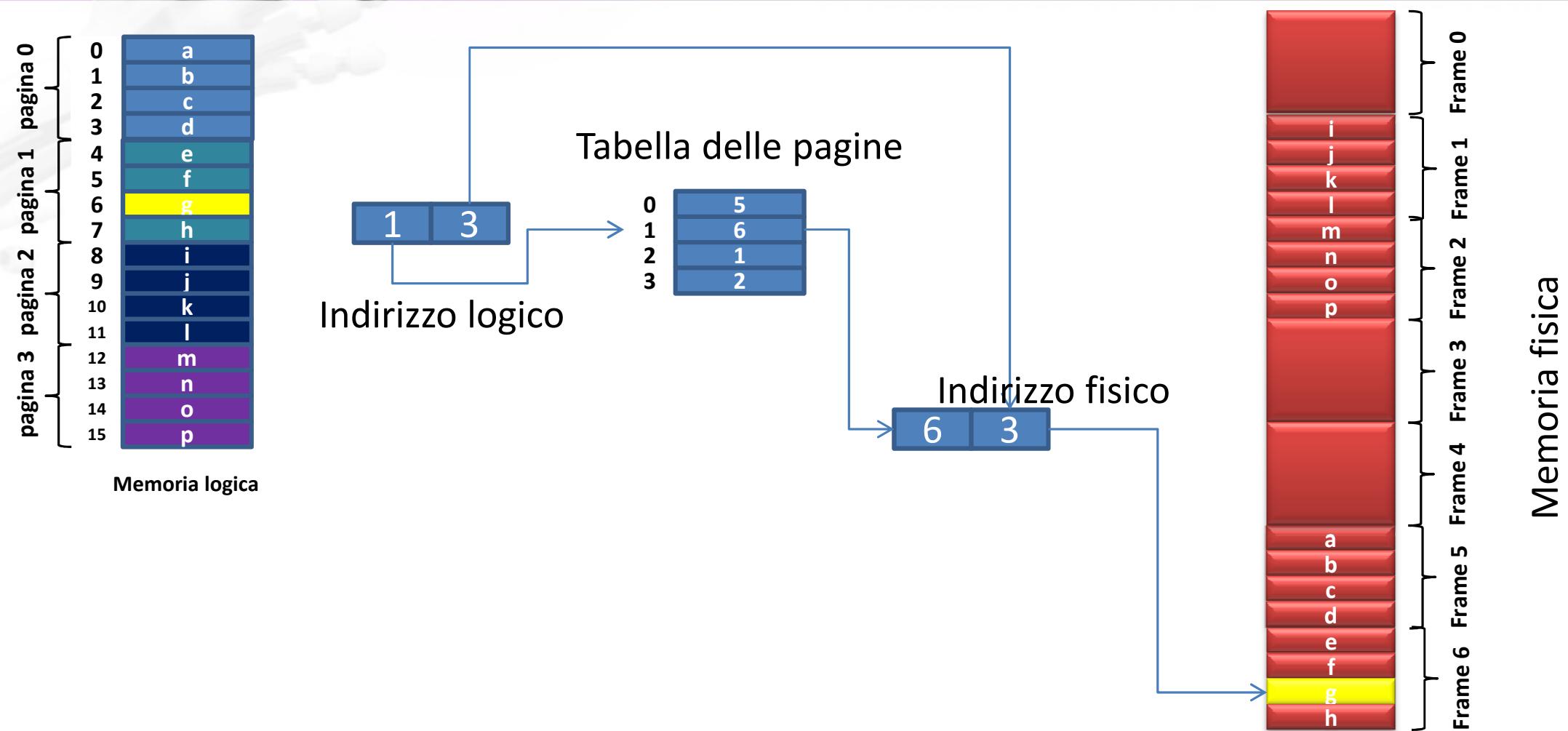
MEMORIA VIRTUALE

Supporto hardware per la paginazione



MEMORIA VIRTUALE

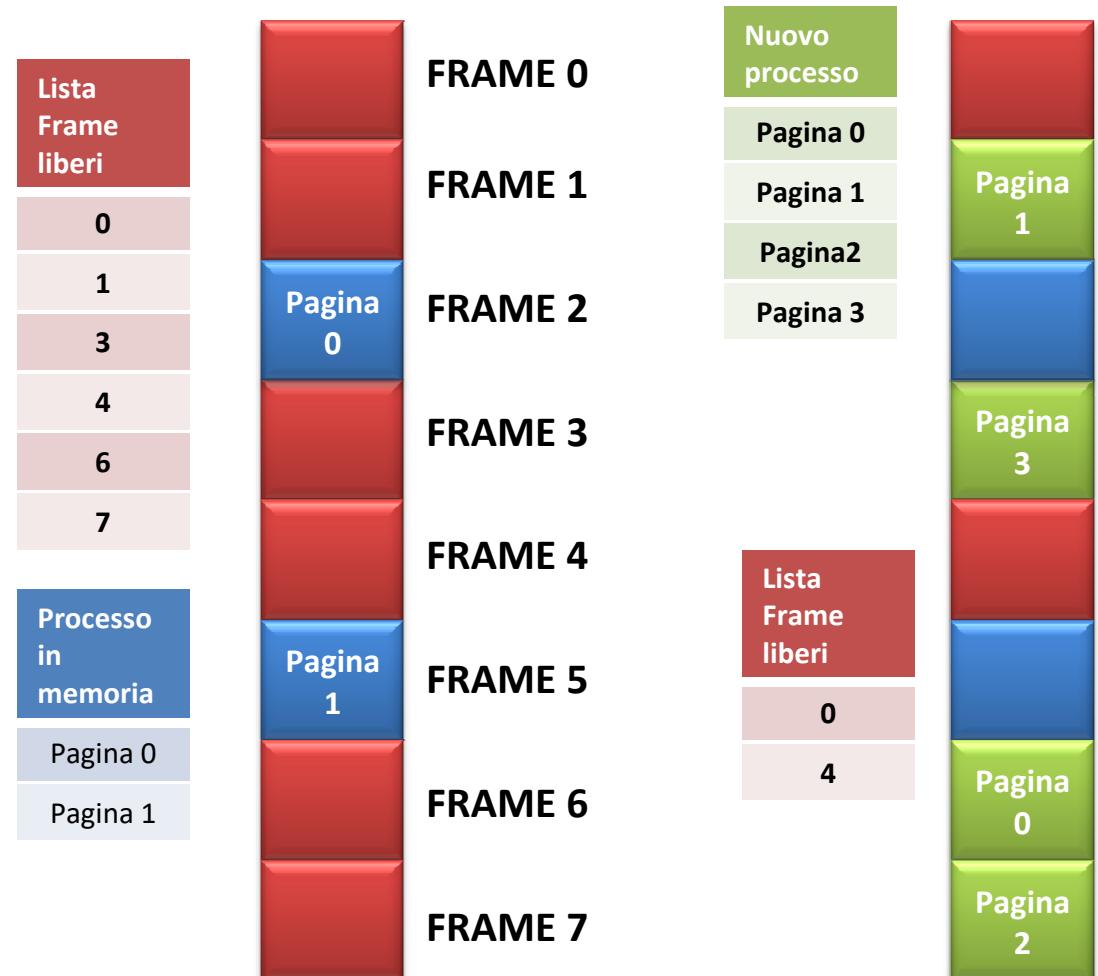
Esempio di paginazione



MEMORIA VIRTUALE

Tabella dei frame

- Per allocare una pagina nel frame è necessario sapere:
 - quali frame sono assegnati e a chi
 - quali e quanti frame sono liberi
- Tali informazioni sono contenute nella **lista dei frame liberi**
 - Implementazione base: una lista in cui ciascun elemento riporta se la pagina fisica è libera o assegnata

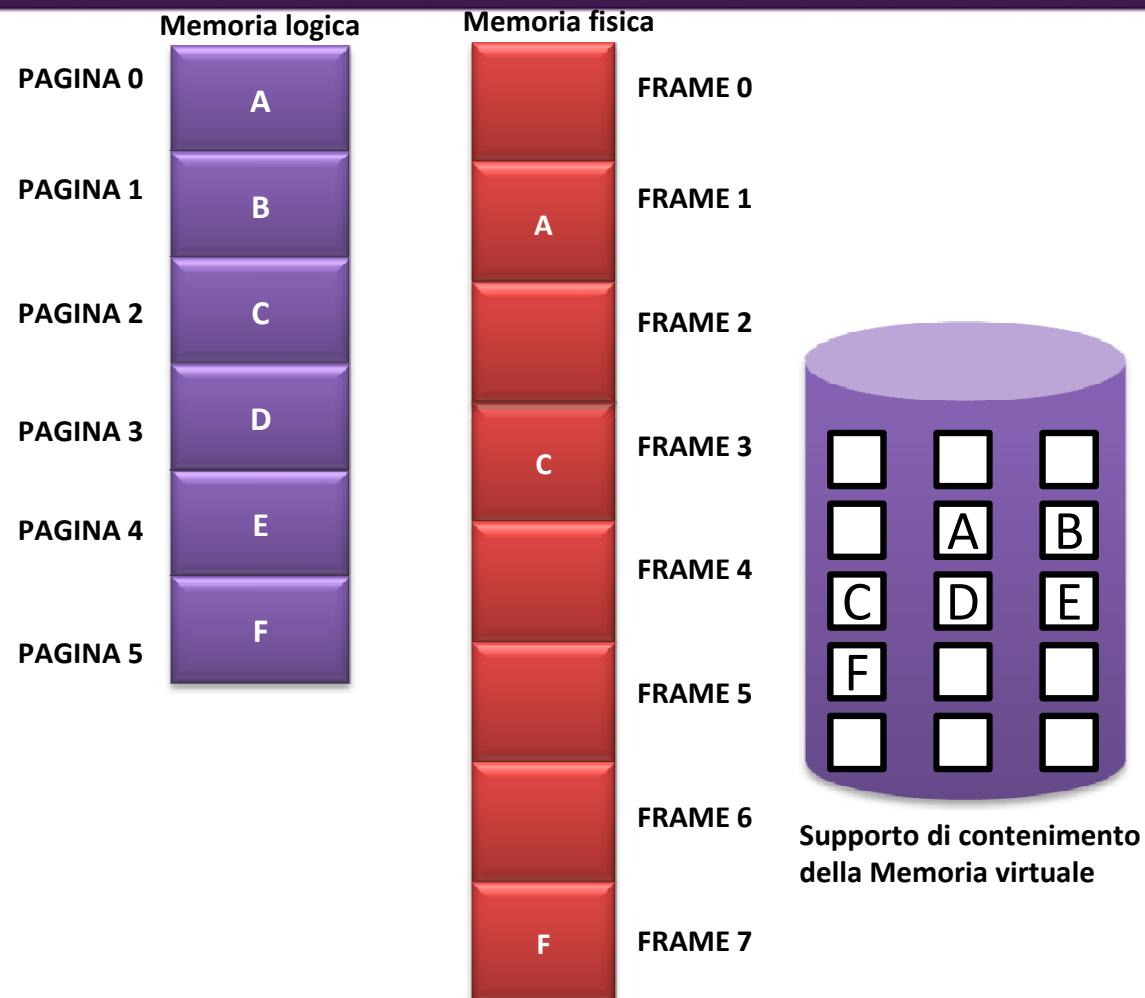




MEMORIA VIRTUALE

Introduzione alla Memoria Virtuale

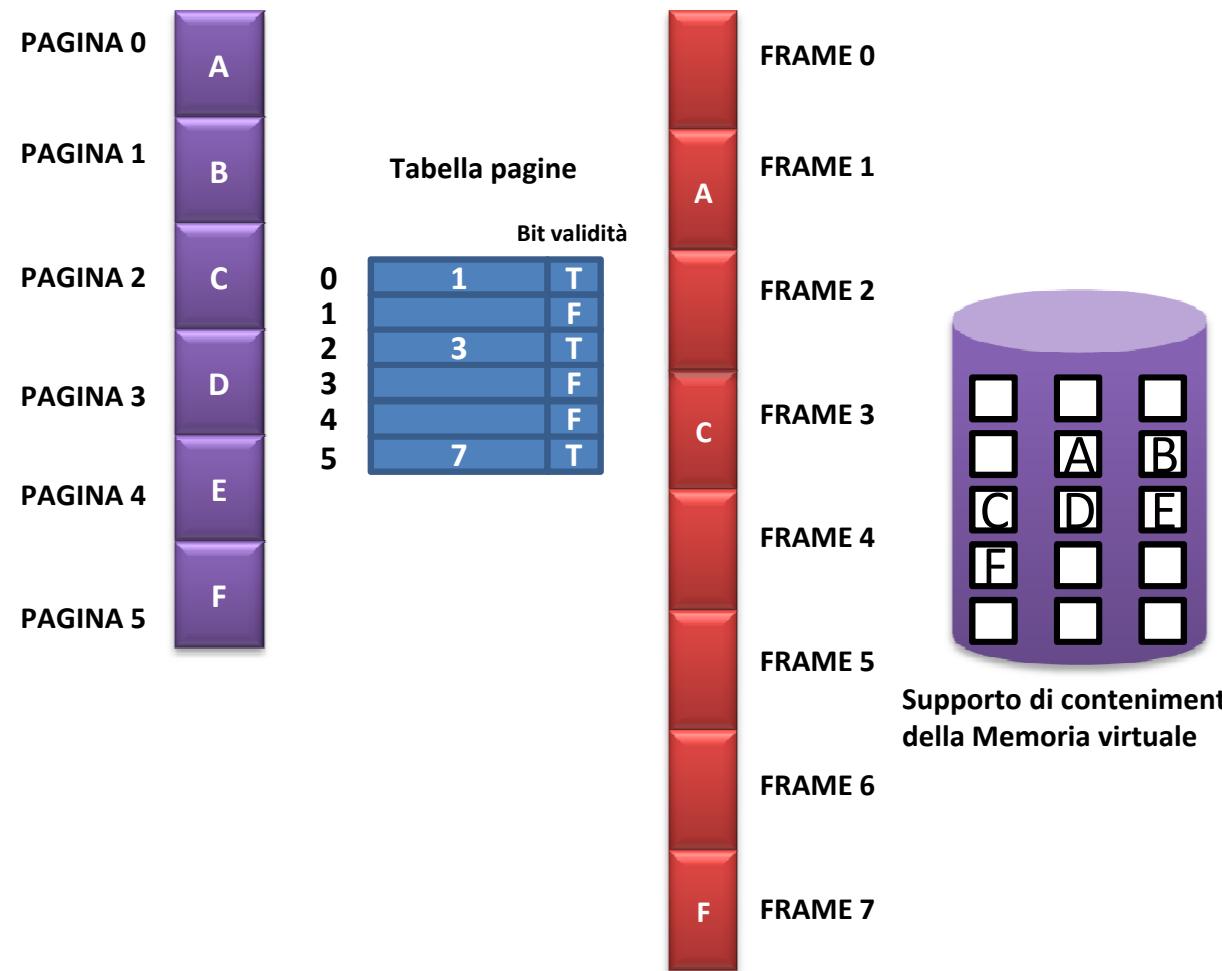
- La MV rappresenta la separazione della memoria logica, vista dall'utente, dalla memoria fisica
- Questa separazione permette di offrire ai programmatori **una memoria molto ampia** (superiore a quella fisica)
- La Memoria Virtuale è una tecnica **trasparente** al programmatore
- Si sfrutta la tecnica di **paginazione**
- I processi risiedono nella **memoria virtuale** generalmente costituita da un disco magnetico (o, negli ultimi anni, da una memoria a stato solido che incrementa la velocità dello **spostamento delle pagine (paging)** da memoria virtuale a memoria fisica perché non include componenti meccaniche)
- Il programma non è caricato per intero in memoria ma si carica una parte e poi solo la pagina necessaria (**PAGE IN** disco→ram e **PAGE OUT** ram → disco)
- Si procede quindi manipolando le singole pagine dei processi: non c'è un avvicendamento dei processi, grazie alla tecnica di **swapping**, ma delle singole parti di ogni processo **pagina**



MEMORIA VIRTUALE

Paginatore

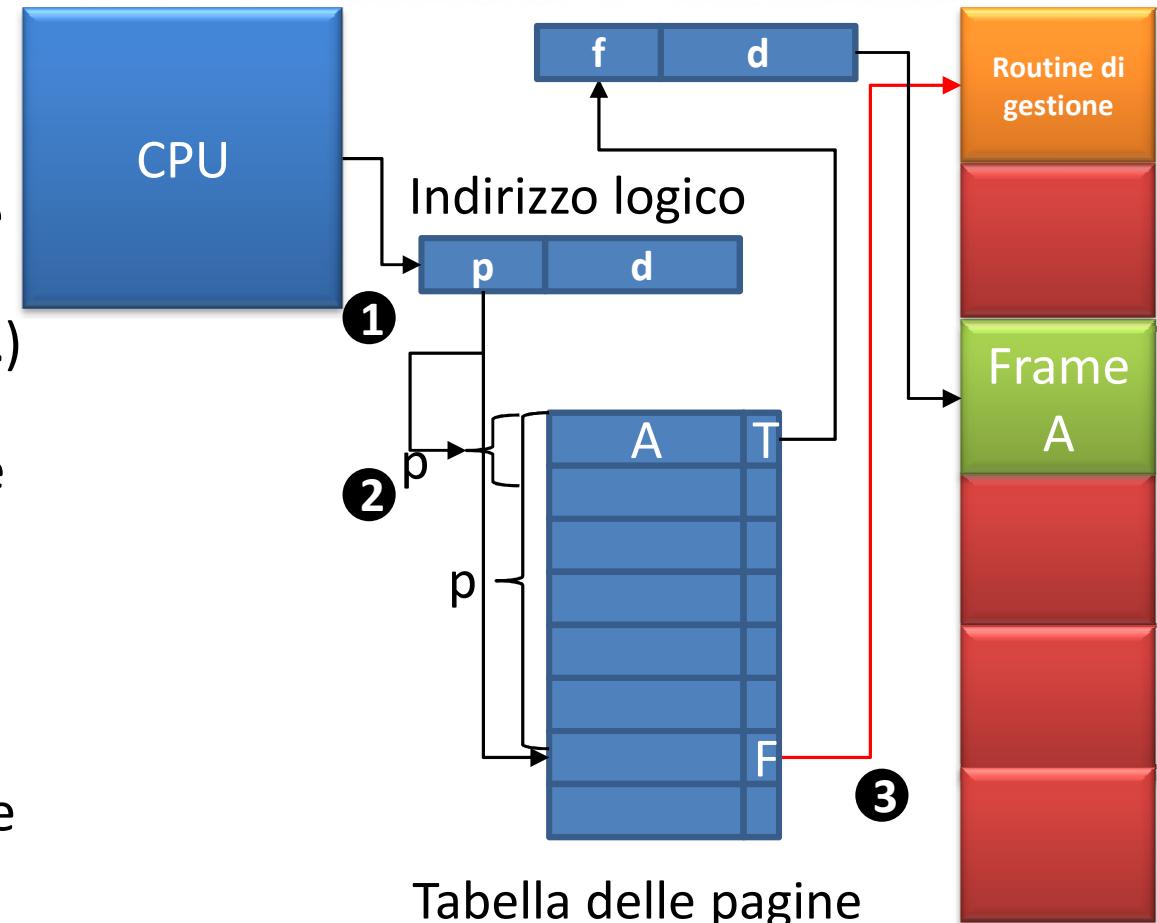
- Il modulo che si occupa dello scambio delle pagine è chiamato **paginatore**
- Una pagina è caricata quando un indirizzo logico all'interno di essa è chiamato (o meglio riferito) dalla CPU cioè quando c'è una interruzione dovuta ad un fallimento (miss) che nella MV si chiama **Page Fault**
- Pertanto, un **pagina** può essere allocata:
 - ❖ in *memoria centrale*
 - ❖ nella *memoria virtuale* (di solito una memoria di massa non volatile)
- Per distinguere i due casi c'è, nella tabella delle pagine, il **bit di validità** (o **bit di presenza**) per indicare se la pagina è nella memoria centrale o in quella virtuale (e quindi va caricato) oppure se se l'indirizzo richiesto non appartiene allo spazio logico del processo
 - ❖ Gli ultimi due casi sono gestiti grazie un Page Fault



MEMORIA VIRTUALE

Funzionamento

- L'architettura di paginazione, si occupa di tradurre l'indirizzo attraverso la tabella delle pagine ed in più gestisce la mancanza di pagine nella memoria centrale
- La CPU richiede un indirizzo logico (1)
- Se il bit è valido lo cerca nel frame individuato dalla tabella delle pagine ricomponendo l'indirizzo fisico (2); altrimenti se non è valido (bit di validità settato ad F: *false*) è inviata una interruzione di **Page Fault** (3): tale condizione è dovuta ad un "insuccesso" nella scelta delle pagine da caricare in memoria

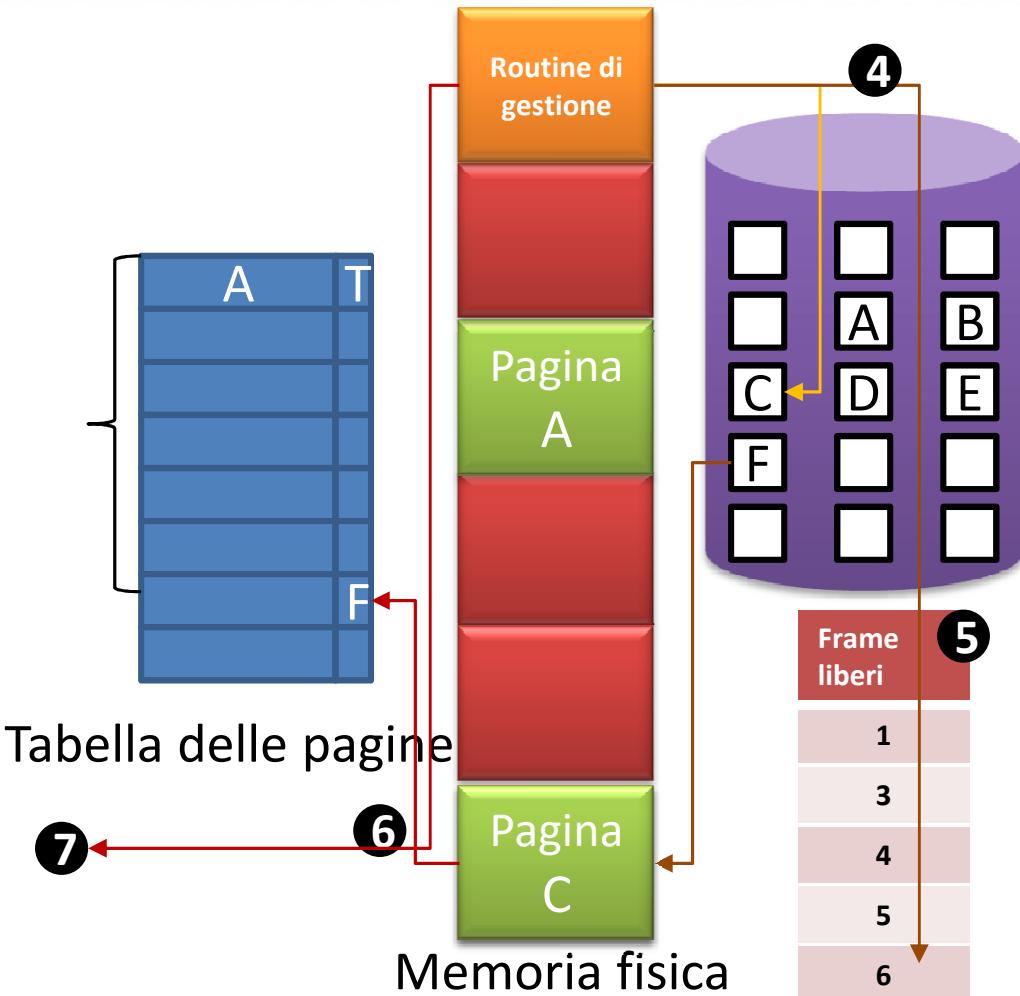


MEMORIA VIRTUALE

Funzionamento

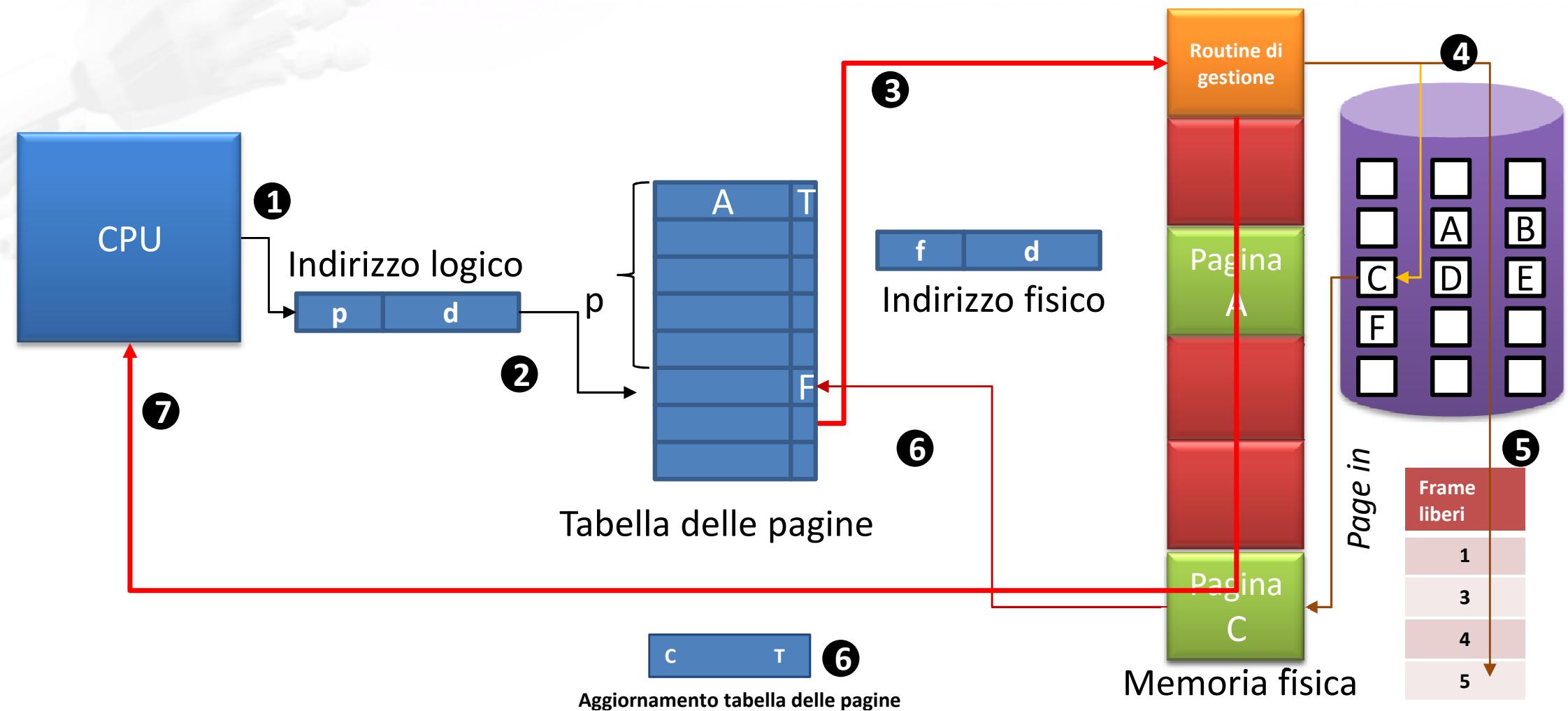
- La gestione dell'interruzione è così espletata:
 - ❖ Si controlla la tabella del processo, la PCB, per verificare che l'indirizzo richiesto sia valido oppure no (e quindi se va fuori dai limiti imposti al programma stesso)
 - ❖ Se il riferimento non è valido si termina il processo (segnalando una condizione di errore, addressing error), se invece è valido lo si carica dalla memoria virtuale (4)
 - ❖ Si individua un frame libero di memoria (5) consultando la relativa lista
 - ❖ Una volta caricato tramite un'operazione di I/O si aggiorna la tabella interna con la nuova pagina caricata (6)
 - ❖ Si riavvia l'istruzione che stava chiedendo la pagina e alla quale - ora - può accedere (7)

Osservazione. è possibile caricare un processo tutto all'interno della memoria virtuale e quindi senza pagine valide (**paginazione pura**) ma a costo di perdita di efficienza



MEMORIA VIRTUALE

Funzionamento





MEMORIA VIRTUALE

Politiche di caricamento

- ❑ Quando c'è un riferimento ad una pagina non presente in memoria si innesca il meccanismo di **prelievo (fetch from virtual memory)** e, se la memoria fisica è completamente occupata, di **rimpiazzo (replacement)**
- ❑ Ci sono due tipi di **politiche di prelievo**:
 - ❖ **On-Demand Paging**: carica la pagina appena viene referenziata (inizialmente si generano molti page fault)
 - ❖ **Prepaging**: carica diverse pagine contigue (principio di località)
 - ❖ Spesso queste pagine risiedono in maniera contigua anche nella memoria virtuale (es.: nel disco magnetico) in modo tale da avvantaggiare i tempi di recupero e trasferimento

MEMORIA VIRTUALE

Contiguità delle pagine nella MV

Piriform Defraggler - Professional Edition

Action Settings Help

Drive	Media Type	Capacity	Used	Free Space	Fragmentation	Status
Windows (C:)	HD (NTFS)	523,1 GB	133,8 GB (26%)	389,4 GB (74%)	16%	Analysis Complete
RECOVERY (D:)	HD (NTFS)	16,5 GB	14,6 GB (88%)	1,9 GB (12%)	Unknown	Ready
DATA (F:)	HD (NTFS)	390,6 GB	205,1 GB (53%)	185,5 GB (47%)	Unknown	Ready

Drive C: File list Search Drive map Statistics

Color

- Empty
- Not Fragmented (Low Occupancy)
- Not Fragmented
- Fragmented (Low Occupancy)
- Fragmented
- Reserved MFT Space
- Page File
- Files being read
- Files being written

Edit Settings...

Piriform Defraggler - Professional Edition

Action Settings Help

Drive	Media Type	Capacity	Used	Free Space	Fragmentation	Status
Windows (C:)	SSD (NTFS)	523,1 GB	133,8 GB (26%)	389,4 GB (74%)	16%	Analysis Complete
RECOVERY (D:)	SSD (NTFS)	16,5 GB	14,6 GB (88%)	1,9 GB (12%)	Unknown	Ready
DATA (F:)	SSD (NTFS)	390,6 GB	205,1 GB (53%)	185,5 GB (47%)	Unknown	Ready

Drive C: File list Search Drive map Statistics Highlighted

Filename	Fragments	Size	Type	Last modified	Path
pagefile.sys	1	1.310.72...	File di sistema	11/05/2021 08:06	C:\

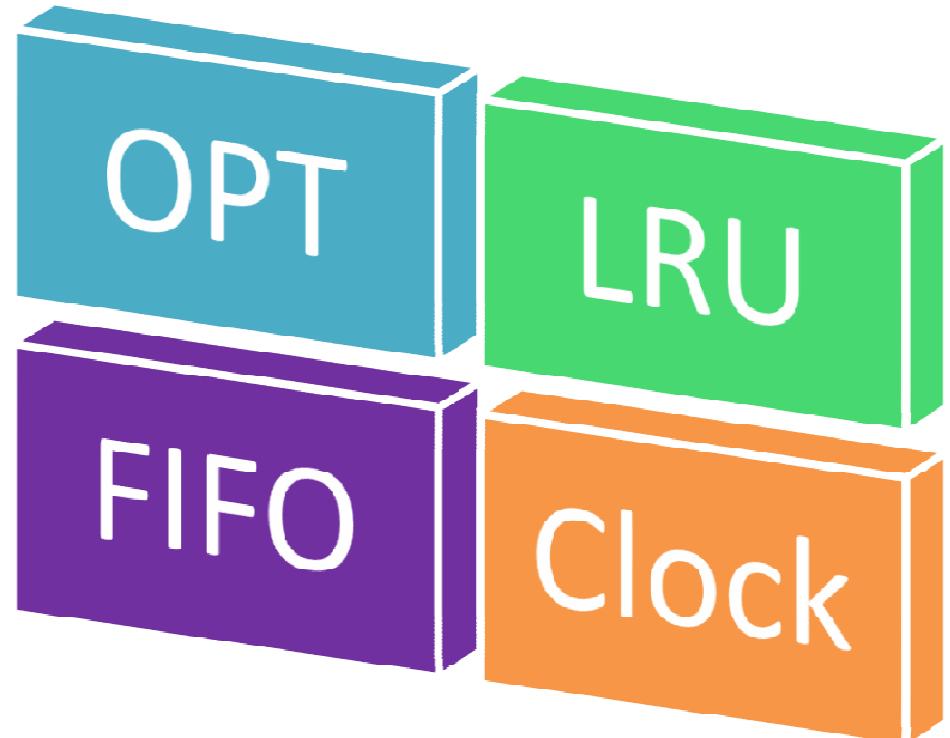
Analyze Optimize Pause



MEMORIA VIRTUALE

Politiche di rimpiazzo

- Decidere quale **pagina rimpiazzare** sono politiche importanti e per questo si può procedere con diversi algoritmi per realizzarle
- La bontà di un algoritmo di **rimpiazzo** è determinata dal numero di page fault che innesca Minore è il numero di page fault migliore è la strategia





MEMORIA VIRTUALE

Politiche di rimpiazzo

OPT (Sostituzione ottimale)

È un'utopistica **sostituzione ottimale** delle pagine.

Non è realizzabile, perché prevede la sostituzione della pagina che verrà usata più tardi (aspetto che non si può determinare a priori)

LRU (Least Recently Used)

Sostituisce la **pagina meno usata di recente**

Ad ogni pagina si associa il tempo in cui è stata utilizzata l'ultima volta; quando c'è un page fault si sostituisce la pagina che non è stata utilizzata per il periodo di tempo più lungo

FIFO (First In – First Out)

Sostituisce la **pagina presente in memoria da più tempo**.

Ad ogni pagina è associato un tempo in cui è stata portata in memoria; quando c'è un page fault si sostituisce la pagina più vecchia.

Semplice ma non sfrutta il principio di località

CLOCK

Ogni pagina, organizzata in una coda circolare, ha un **bit di riferimento** settato ad 1 quando si accede a quella pagina. Quando c'è un page fault un puntatore scandaglia il bit di riferimento di ciascuna pagina: se è 1 è settato a zero e si passa avanti; se è 0 invece avviene la sostituzione

MEMORIA VIRTUALE

Politiche di rimpiazzo

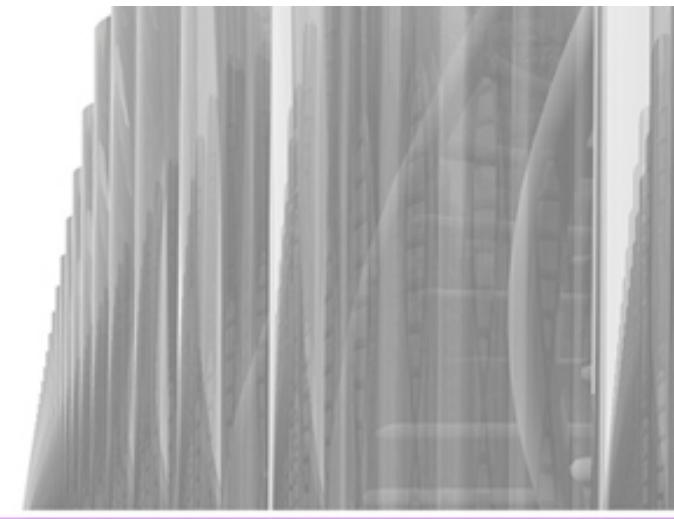
Pagine	2	3	2	1	5	2	4	5	3	2	5	2
OPT	2	2	2	2	2	2	4	4	4	2	2	2
FIFO	2	2	2	2	5	5	5	5	3	3	3	3
LRU	2	2	2	2	2	2	2	2	3	3	3	3
CLOCK (coda circolare con puntatore)	2	2	2	2	5	5	5	5	5	5	5	5

6 Page Fault

9 Page Fault

7 Page Fault

8 Page Fault



Fine



OVERLAY

Generalità

- Nel caso in cui si voglia eseguire un programma che richiede una area di memoria più grande della partizione ammissibile o della stessa memoria a disposizione si può decidere di ricorrere ad una tecnica chiamata **overlay**
- Il concetto di overlay è basato sulla possibilità di mantenere in memoria solo le istruzioni e dati che sono utilizzate in un determinato momento (o con maggior frequenza)
- Quando sono necessarie altre istruzioni, queste sono caricate nello spazio che era precedentemente occupato dalle istruzioni che in quel momento non sono più utilizzate
- Tuttavia le prestazioni non sono ottime a causa delle operazioni di I/O necessarie per sovrascrivere il codice residente sul disco (**operazioni di swapping**)



OVERLAY

Generalità

- L'overlay è a discrezione e completa implementazione del programmatore (è una tecnica non trasparente); si tratta di caricare e scaricare dalla memoria soltanto le parti del programma che sono necessarie sostituendole ai dati/moduli caricati precedentemente nella zona dedicata.
- L'overlay deve essere gestito da un **gestore di overlay** implementato dal programmatore stesso dell'applicazione e soltanto i dati comuni e usati globalmente per l'applicazione devono rimanere in memoria. Il gestore dell'overlay è quindi la porzione di software che da il controllo al modulo chiamato caricando la porzione di memoria necessaria (mappa i riferimenti alle routine e ai piazzamenti)
- Questa tecnica è complicata perché scaricamenti non necessari o errati possono portare a comportamenti del processo imprevedibili. L'utilizzo dell'overlay, infatti, è confinato a sistemi con memoria limitata (cellulari, dispositivi mobili,...) e che non dispongono di un hardware raffinato per supportare altre tecniche (anche se attualmente è una tecnica dismessa)



OVERLAY

Esempio

- Si supponga di dover gestire il programma di un assemblatore a due passi. Durante il primo passo Asm1, l'assemblatore costruisce una tabella dei simboli; mentre al secondo passo Asm2 genera il codice in linguaggio macchina. Un programma siffatto può essere rappresentato come segue:

Asm1	200 Kb
Asm2	300 Kb
Tabella Simboli	100 Kb
Routine comuni	150 Kb

Per caricare tutto il programma occorrerebbero 750 Kb di memoria. Nel caso in cui ne fossero disponibili solamente 600 Kb si può decidere di effettuare un overlay perché, per come opera, il programma non richiede necessariamente la presenza contemporanea della prima e della seconda parte. Pertanto una volta implementato un gestore di overlay (50 Kb); si può pensare di caricare in memoria la Tabella Simboli, il gestore overlay, le Routine comuni e l'Asm1 (500 Kb) ed eseguire la prima parte; poi si sostituisce il modulo oggetto Asm1 con Asm2 (per un totale di 600 Kb) e si conclude l'esecuzione del programma.



OVERLAY

Esempio

