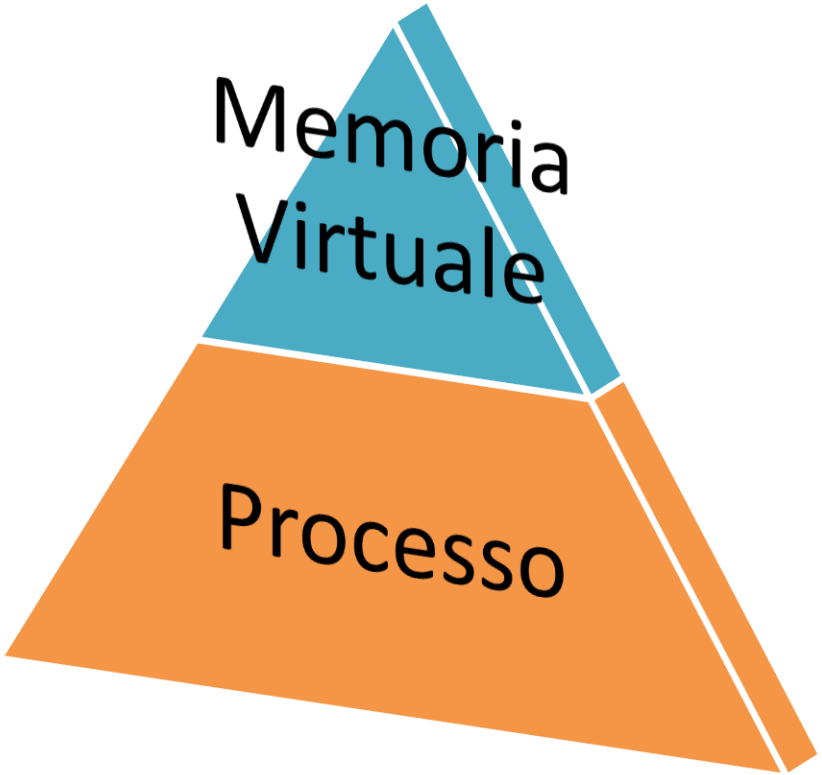
Architettura degli Elaboratori Elettronici

Dott. Franco Liberati
liberati@di.uniroma1.it

ARGOMENTI DELLA LEZIONE

- Memoria Virtuale

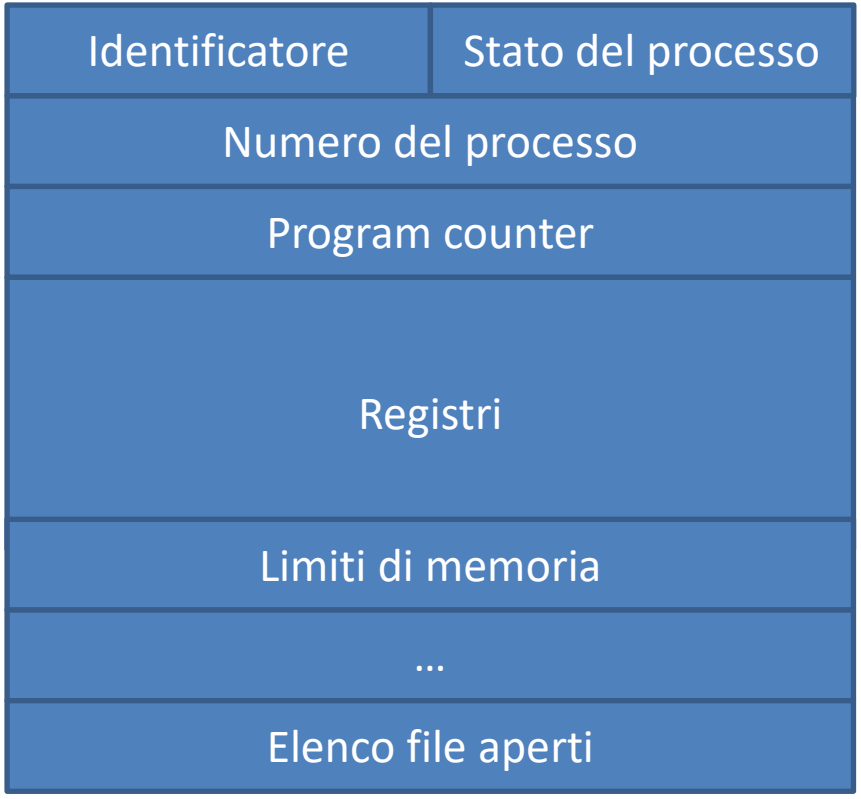


Memoria Virtuale

PROCESSO

Definizione

- Un programma presente in Memoria Centrale, ed in corso di esecuzione, è detto **processo**
- Ogni processo ha una tabella delle 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 eseguire)
  - Registri CPU (informazioni salvate all'atto di una interruzione)
  - Informazioni sullo scheduling di CPU (es.: la priorità del processo)
  - **Informazione sulla 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)



PROCESSO

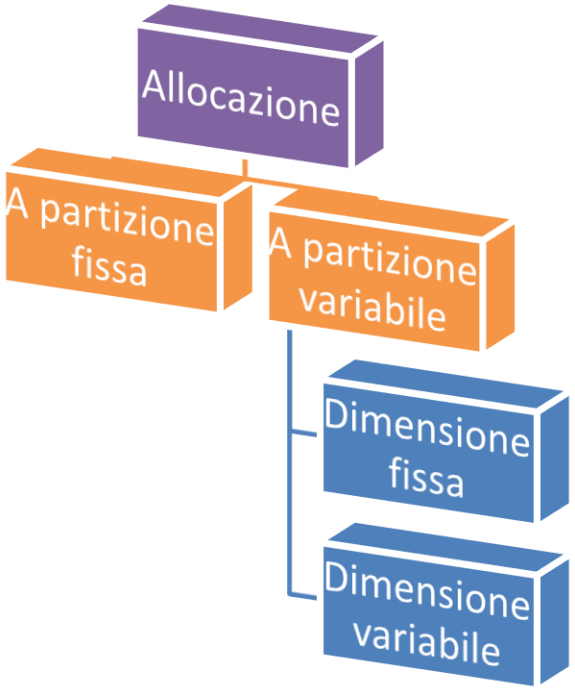
Definizione

- La **memoria principale** è una risorsa limitata, che deve essere allocata in modo efficiente, cioè i processi devono essere posizionati in aree opportune
- Deve contenere sia i processi utente sia 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 alla tabella delle interruzioni o allo handler del polling
  - 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 e suddivisa in partizioni logiche
      - A dimensione fissa
      - A dimensione variabile



PROCESSO

Allocazione Memoria a partizione singola

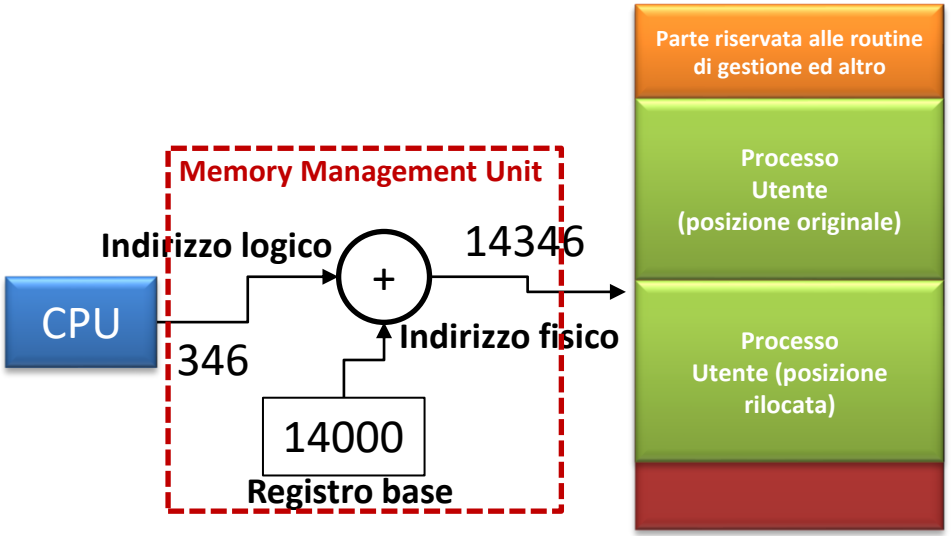
- **Allocazione a partizione singola**
  - La parte di memoria disponibile per l'allocazione dei processi non è partizionata
  - Un solo processo alla volta può essere allocato in memoria
  - Il Sistema Operativo è locato ad una posizione prestabilita (es.: MS-DOS anni '80)

A partizione singola

PROCESSO

Memory Managment Unit

- Con una modifica hardware, cioè tramite un **registro di rilocazione**, si può svolgere la **rilocazione dinamica** effettuata
- Si usa il **Memory Managment Unit** cioè almeno un addizionatore e un registro di rilocazione (o registro base, *base register*) al cui interno è presente l'indirizzo fisico della prima istruzione del programma
  - Cambiando locazione al programma e aggiornando il registro base si può sommare l'indirizzo relativo (o logico) ed ottenere l'indirizzo assoluto (o indirizzo fisico). L'aggiornamento è svolto a seguito di una interruzione che si attiva quando si sposta il programma



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 :
      - **Frammentazione interna**: sottoutilizzo della partizione
      - **Grado di multiprogrammazione limitato dal numero di partizioni**
      - **Dimensione massima dei processi limitata dalla dimensione della partizione più estesa**

PROCESSO

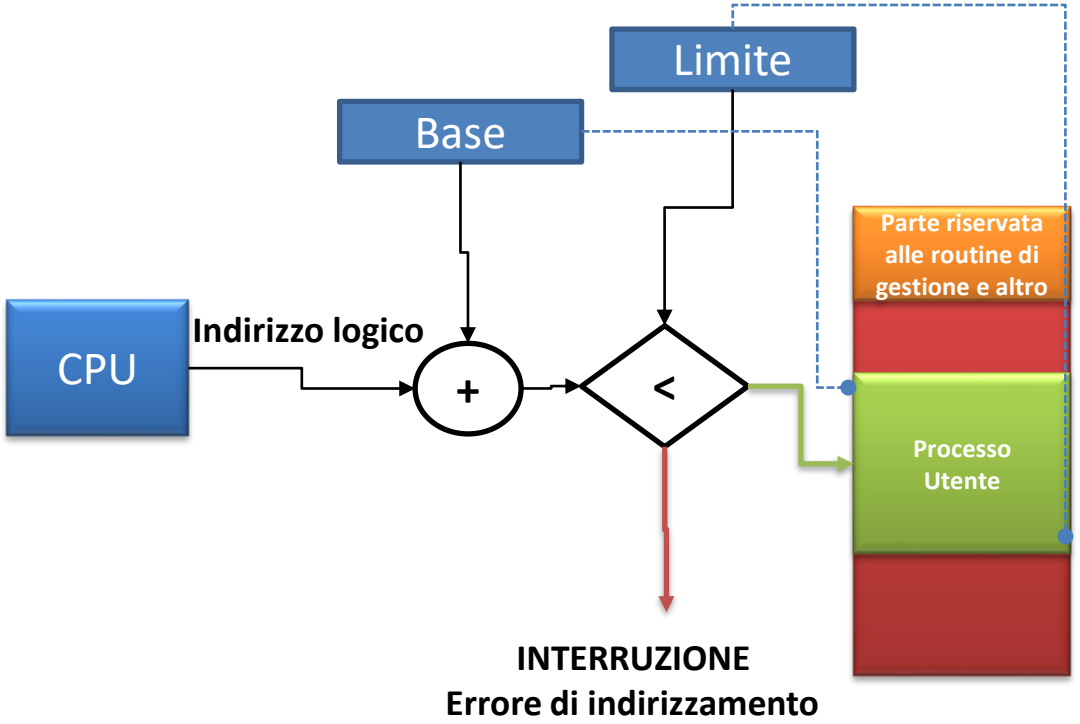
Allocazione Memoria a partizioni multiple

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

PROCESSO

Memoria a partizione multipla: protezione

- Poiché sono presenti più processi è opportuno disporre di un sistema di protezione
  - **Registro base** e **registro limite** che indicano rispettivamente l'indirizzo della prima istruzione del processo e l'indirizzo dell'ultima istruzione del processo (mutando l'ordine di uso dei registri si può avere in base l'indirizzo della prima istruzione e in limite la lunghezza del processo cioè il numero di istruzioni macchina che lo compongono)



PROCESSO

Frammentazione

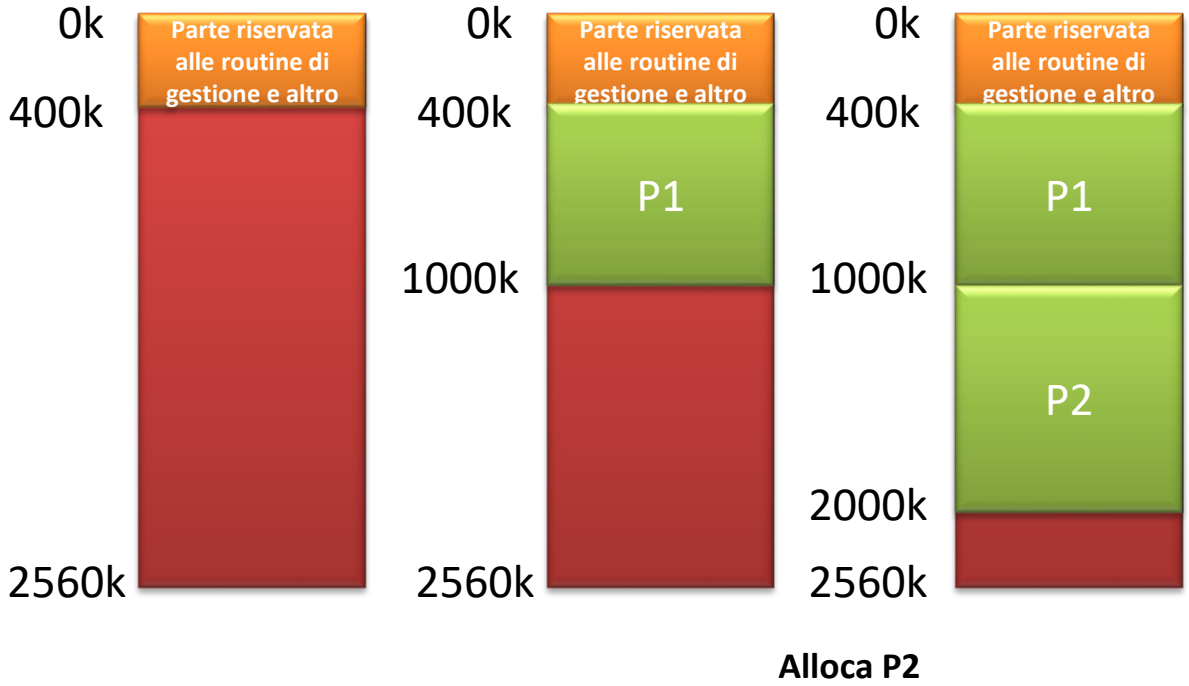
- **Frammentazione interna**
  - La memoria allocata può essere leggermente maggiore della memoria effettivamente richiesta (pochi byte di differenza): la parte restante richiede l'impiego di una partizione che non è impiegata completamente (e non può essere utilizzata per altri processi)
- **Frammentazione esterna**
  - Si verifica quando è disponibile lo spazio necessario a soddisfare una richiesta, cioè ospitare un processo in memoria, 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 è effettuata a runtime (cioè mentre si esegue il programma)
      - 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)

❑ Schema

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



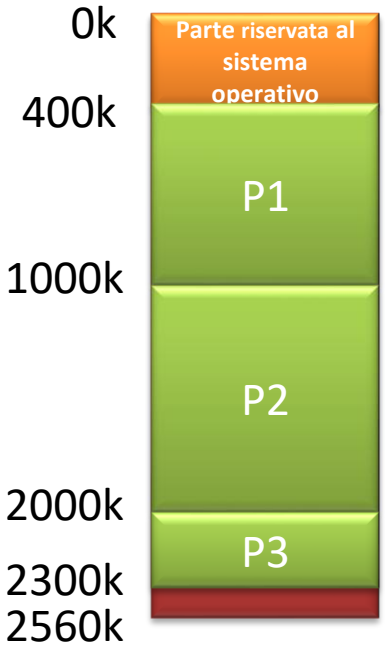
Alloca P2

PROCESSO

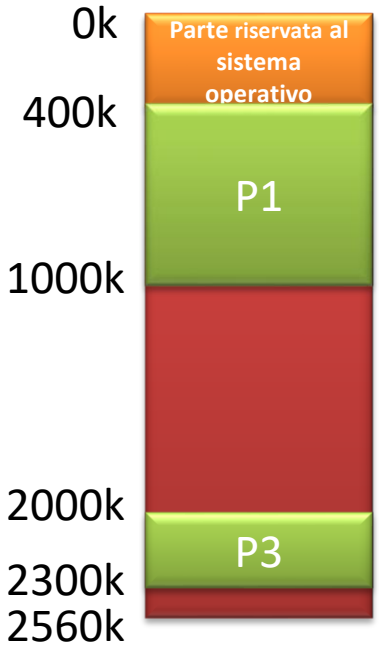
Frammentazione esterna (esempio)

- Schema

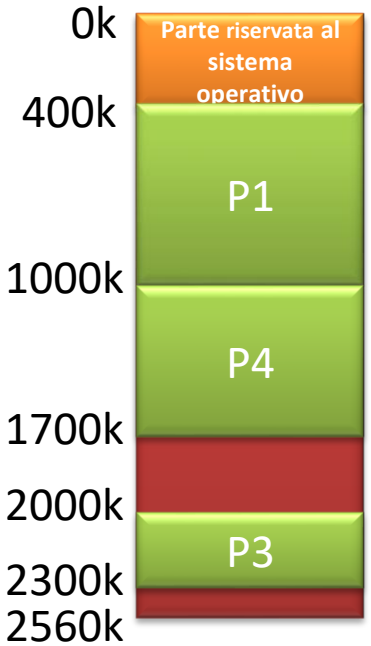
Processo	Dimensione	Tempo di stazionamento in memoria
P1	600K	10
P2	1000K	5
P3	300K	20
P4	700K	8
P5	500K	15



Alloca P3



Termina P2



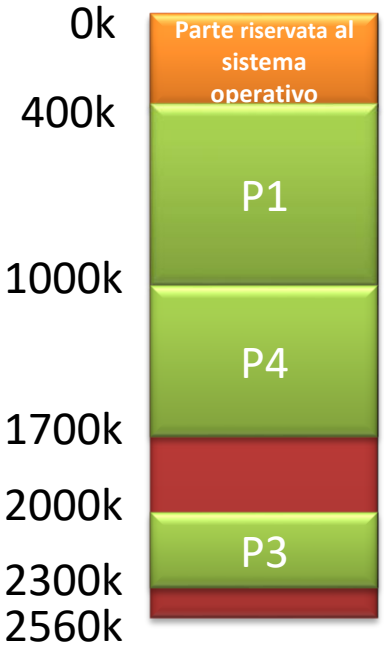
Alloca P4

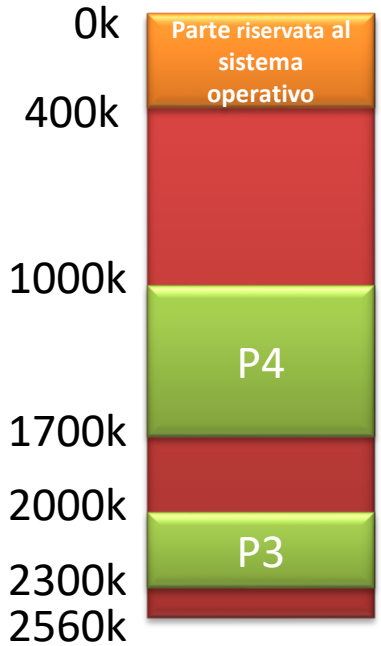
PROCESSO

Frammentazione esterna (esempio)

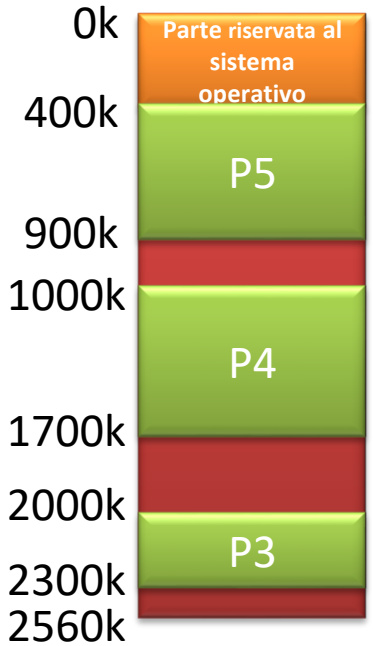
- Schema

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





Termina P1

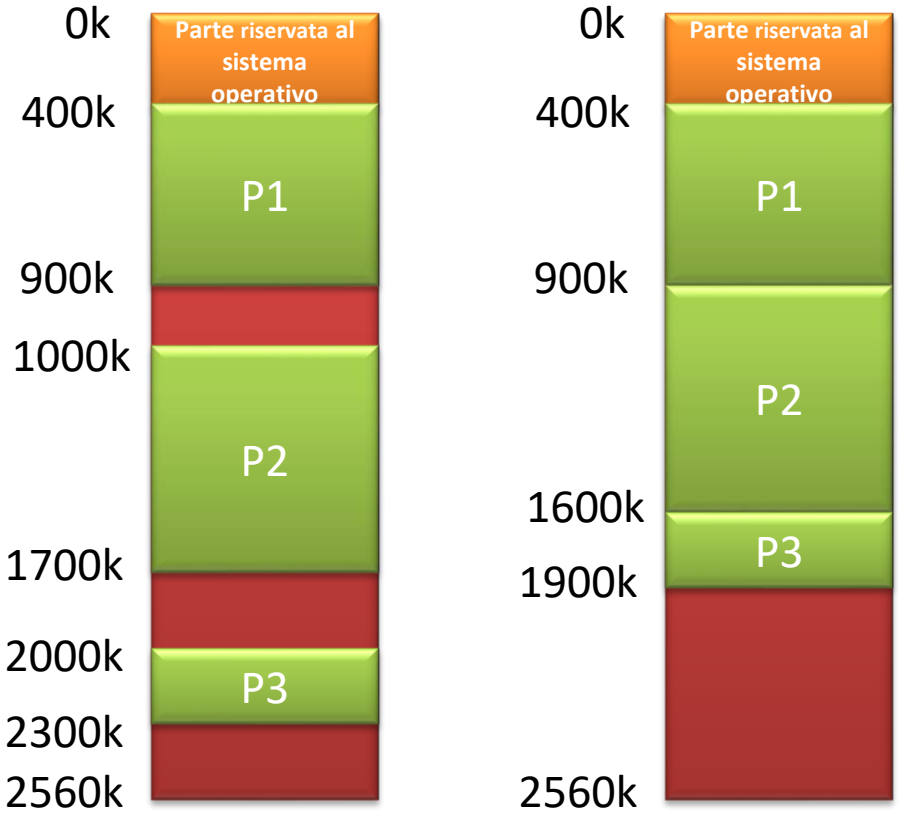


Alloca P5

PROCESSO

Compattazione

- 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 nelle architetture dove è consentita la **rilocazione dinamica** perché è effettuata a runtime



Overlay

OVERLAY

Generalità

- Nel caso in cui si voglia eseguire un programma che richiede una area di memoria più grande della partizione ammissibile o dello spazio fisico disponibile 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 i dati che sono utilizzati 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 trasferire le porzioni del documento eseguibile residente sul disco (**operazioni di swapping**)

OVERLAY

Generalità

- L'overlay è a discrezione e a 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 offre 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 comportare comportamenti del processo imprevedibili. L'utilizzo dell'overlay, infatti, è confinato a sistemi con memoria limitata (primi eleboratori) e che non dispongono un hardware raffinato per supportare altre tecniche

OVERLAY

Esempio

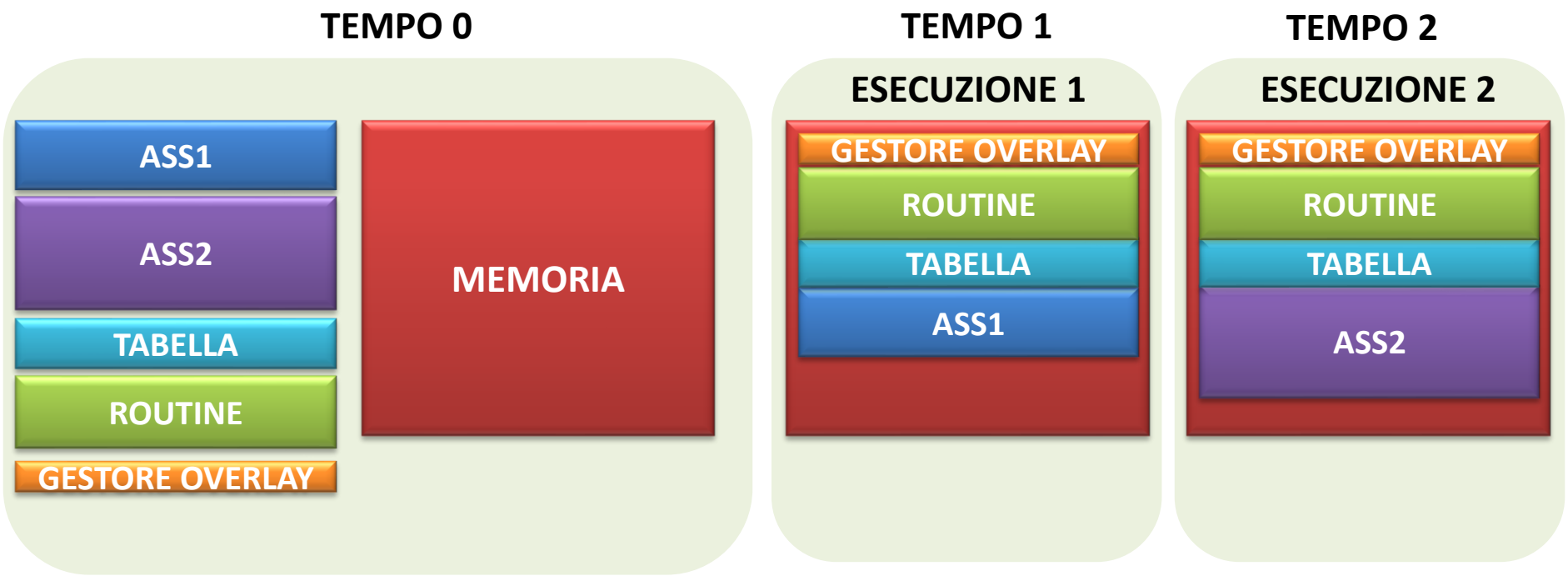
- Si supponga di dover gestire il programma di un assemblatore a due passi. Durante il primo passo Ass1, l'assemblatore costruisce una tabella dei simboli; mentre al secondo passo Ass2 genera il codice in linguaggio macchina. Un programma siffatto può essere rappresentato come segue:

Ass1	**200 Kb**
Ass2	**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'Ass1 (500 Kb) ed eseguire la prima parte; poi si sostituisce la parte Ass1 con la Ass2 (per un totale di 600 Kb) e si conclude l'esecuzione del programma.

OVERLAY

Esempio



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 completamente all'interno della memoria
- 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 RAM di volta in volta (è un compito svolto dal Sistema Operativo)
- 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

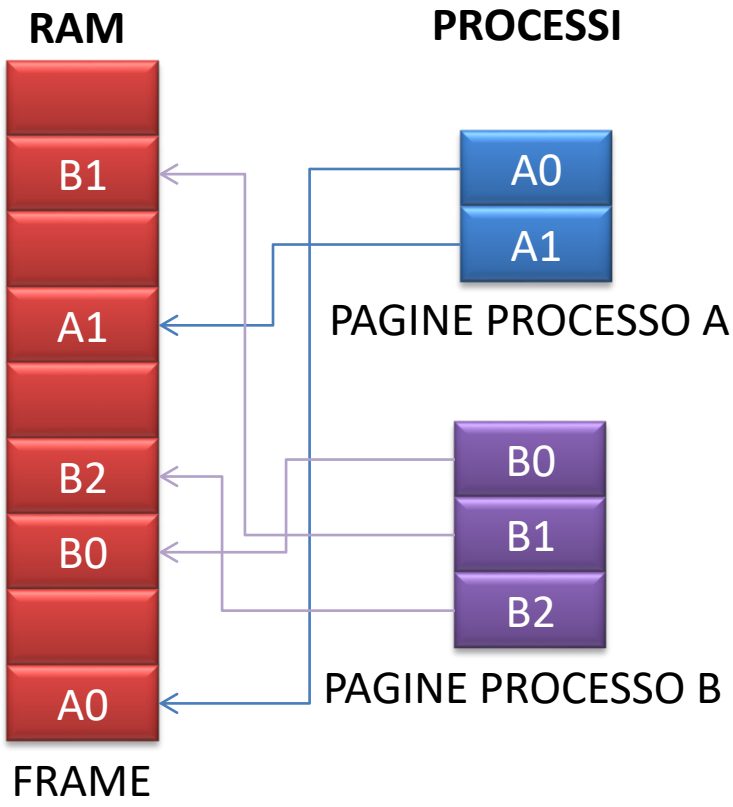
Paginazione

- La Memoria Virtuale richiede la definizione e l'implementazione della tecnica della **paginazione**
- La paginazione è un metodo di gestione della memoria che fa si 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 aventi stessa dimensione dette **frame** (o **pagine fisiche**); dove, di solito, una pagina corrisponde ad un blocco
- Mentre un programma è suddiviso in parti della stessa dimensione del frame ciascuna delle quali è chiamata **pagina** (o **pagina logiche**)
- Ciascuna pagina può essere quindi caricata in un frame della memoria fisica
- In questo modo si sfrutta la RAM a disposizione nella maniera più flessibile e non si è obbligati a inserire i programmi in frame contigui
- 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 genere 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 8192byte, un processo che occupa 74.066 byte necessita di 9 frame e 338byte. In questo caso sono assegnati 10 frame, con una **frammentazione interna** (all'ultimo frame) di 7854byte (cioè circa il 96% di spazio inutilizzabile del singolo frame)

I moderni calcolatori hanno frame da 4096byte o 8192byte

MEMORIA VIRTUALE

Paginazione

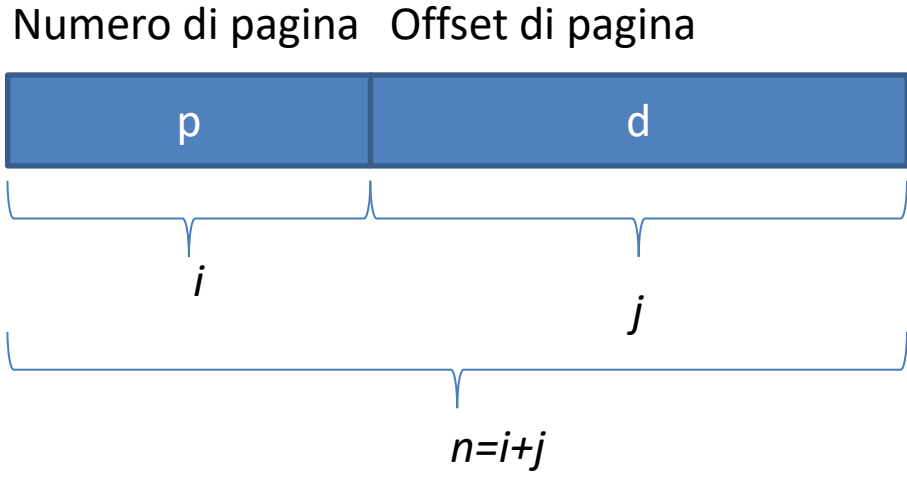
- Si ottiene uno spazio degli **indirizzi logici** totalmente separato dallo spazio degli **indirizzi fisici** (le due tipologie di indirizzi, in generale, hanno dimensione diversa)
  - Per associare ad un indirizzo logico il corrispondente indirizzo fisico si ricorre allo **schema di traduzione degli indirizzi**
- Per eseguire un programma di dimensione *n* pagine, è necessario trovare almeno un frame libero
  - Per risolvere questo problema si introduce l'**individuazione dei frame liberi**

MEMORIA VIRTUALE

Schema di traduzione degli indirizzi

- Nello schema di traduzione degli indirizzi si impiega una **tabella delle pagine** per trasporre 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 di 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 nella tabella delle pagine per definire l'indirizzo fisico da raggiungere

INDIRIZZO GENERATO DA CPU



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

MEMORIA VIRTUALE

Supporto hardware per la paginazione

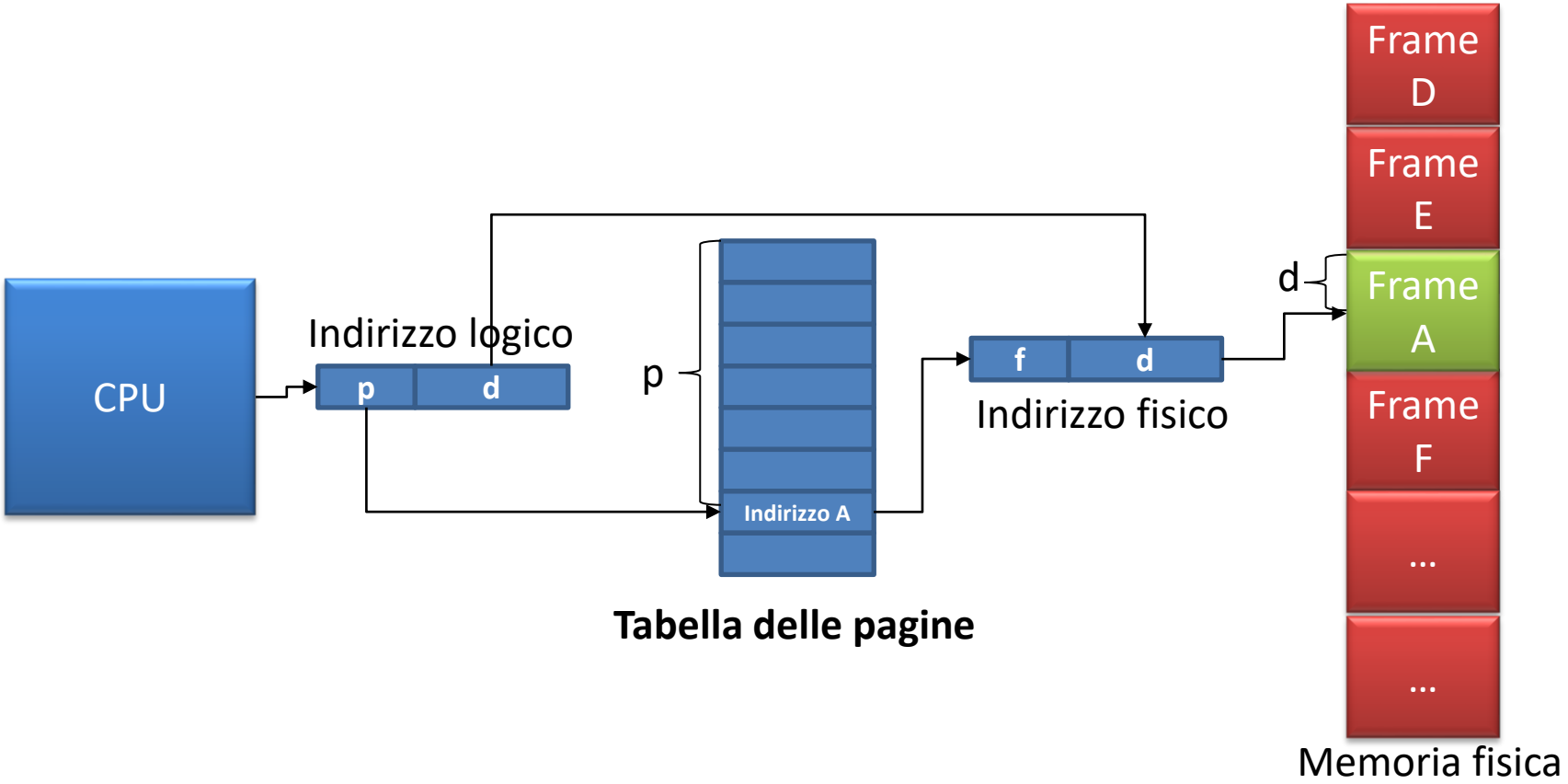
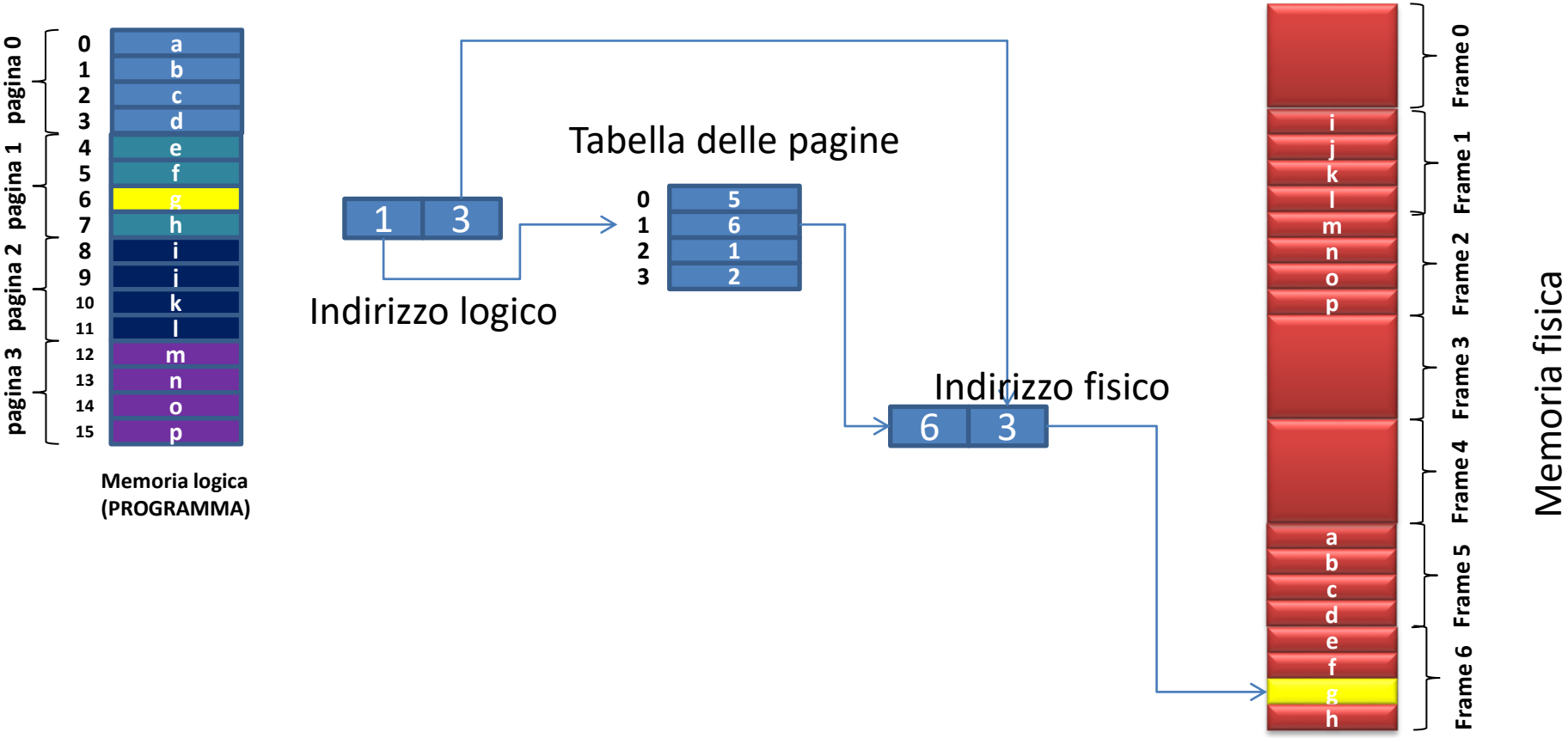


Tabella delle pagine

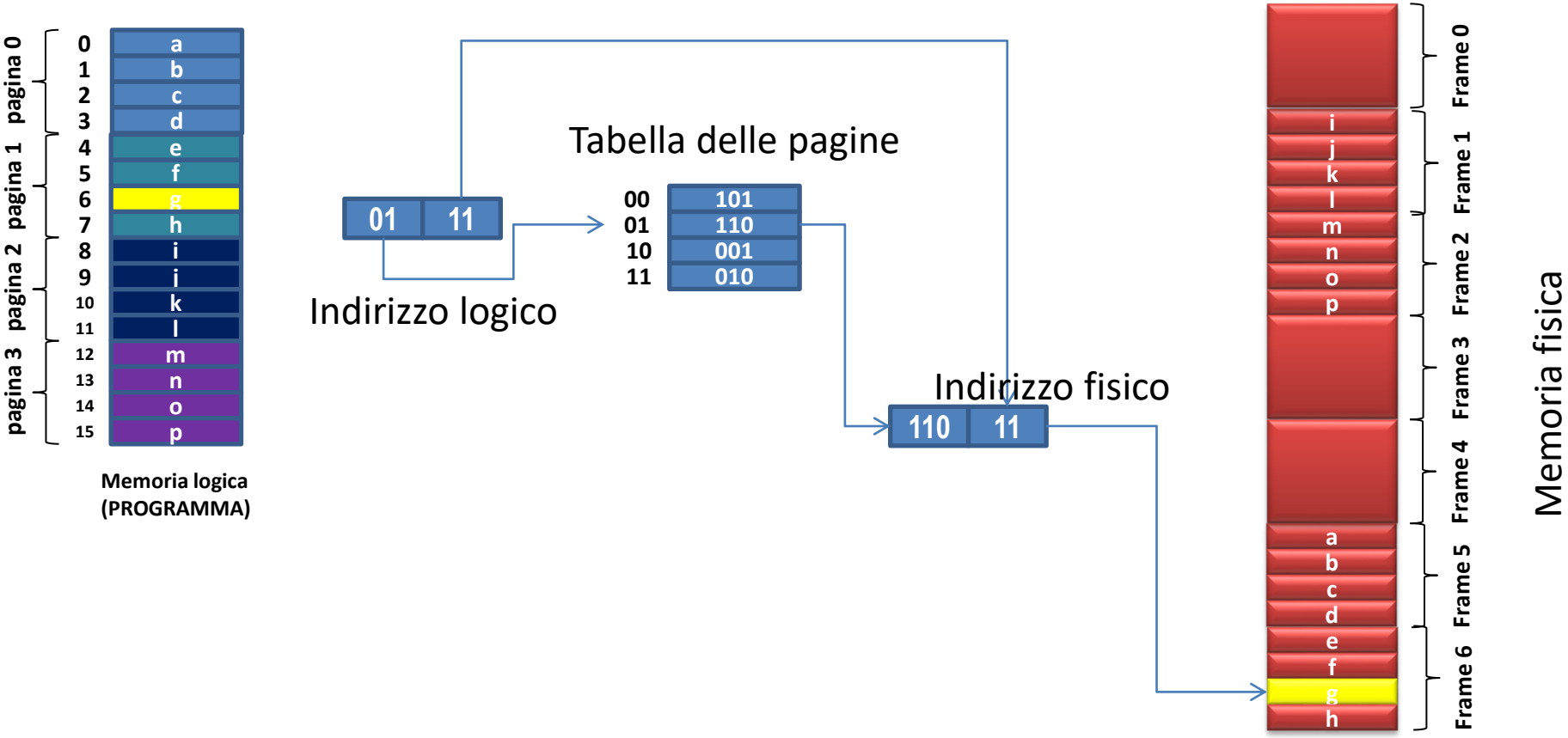
MEMORIA VIRTUALE

Esempio di paginazione (1 frame 4 parola)



MEMORIA VIRTUALE

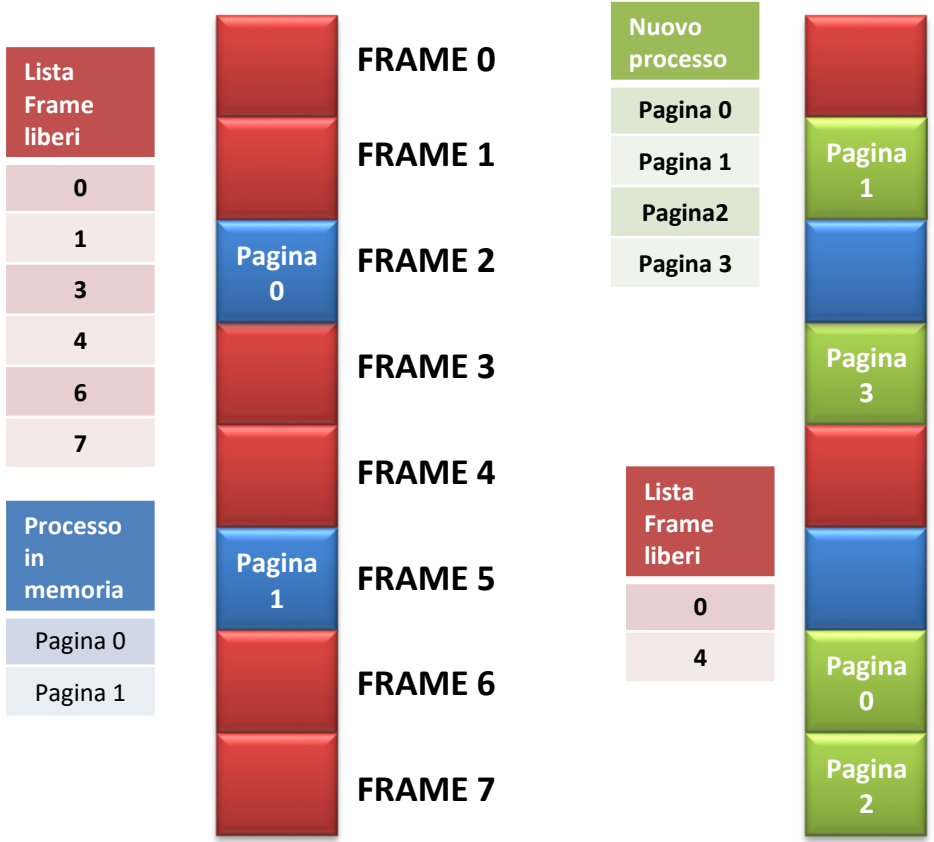
Esempio di paginazione (1 frame 4 parole)



MEMORIA VIRTUALE

Tabella dei frame

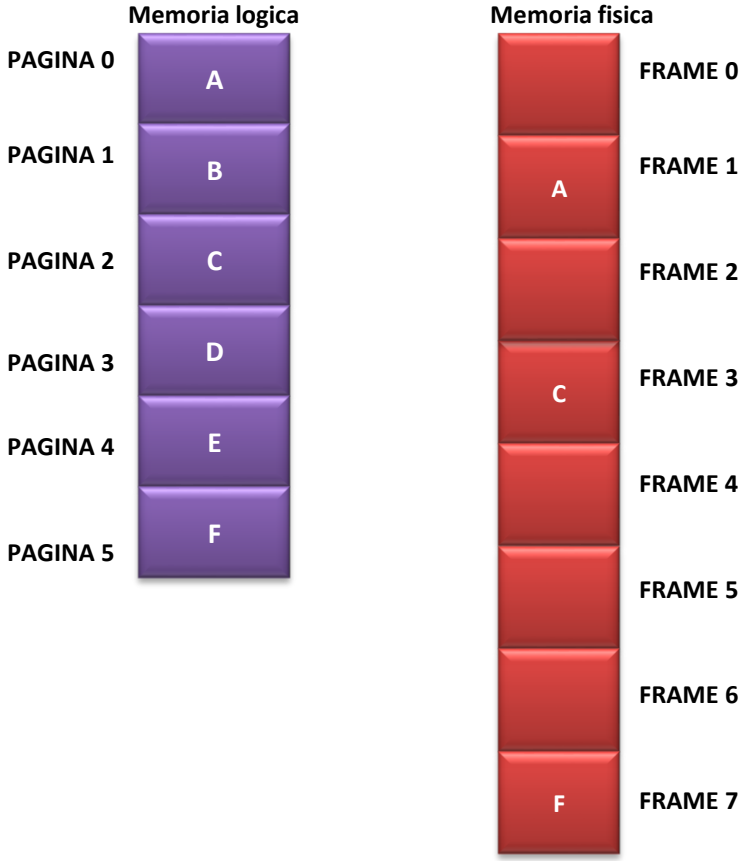
- Per allocare le pagine nei frame è necessario sapere:
  - quali frame sono assegnati e a quale pagina
  - quali e quanti frame sono liberi
- Tali informazioni sono contenute nella **lista dei frame liberi**
  - Implementazione base: una lista in cui a ciascun elemento è associato un frame che consente di definire se la pagina fisica è libera o assegnata



MEMORIA VIRTUALE

Introduzione

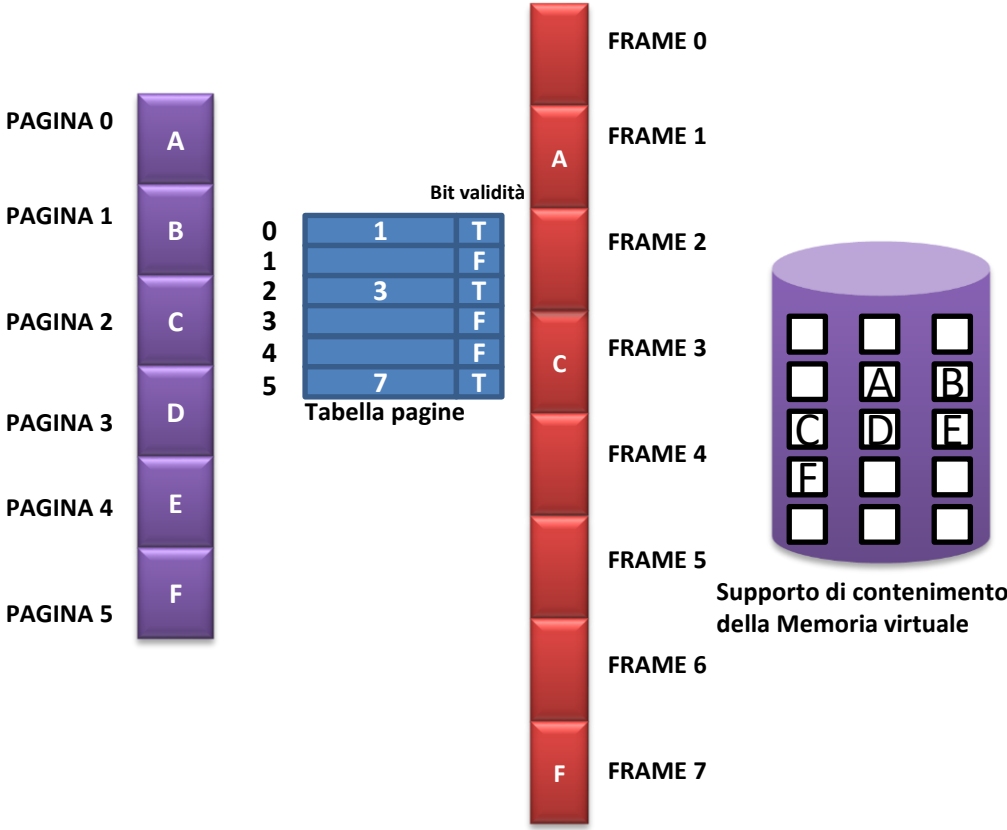
- La **Memoria Virtuale** rappresenta la separazione della memoria logica, vista dall'utente, dalla memoria fisica presente nell'elaboratore
- Questa separazione permette di offrire al programmatore **una memoria molto ampia** (superiore a quella fisica)
- I processi risiedono nella **Memoria Virtuale** 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 trasferisce una parte e poi solo la pagina necessaria (**PAGE IN** disco->ram **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: il **paging**



MEMORIA VIRTUALE

Paginatore

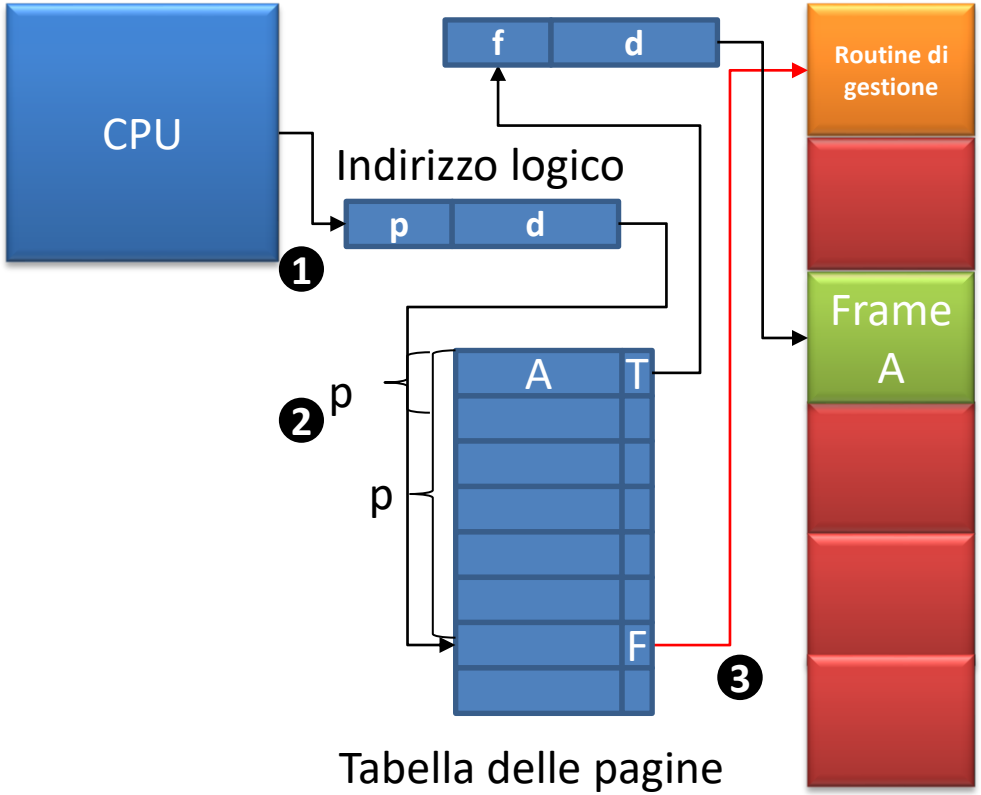
- Il modulo che si occupa dello scambio delle pagine è chiamato **paginatore**
- Una pagina è caricata quando un indirizzo logico all'interno di essa è chiamato (o meglio riferito) dalla CPU cioè quando c'è una interruzione dovuta ad un fallimento (miss) che nella MV si chiama ***Page Fault***
- Pertanto, un **pagina** può essere allocata:
  - in *memoria centrale*
  - nella *memoria virtuale* (di solito una memoria di massa)
- 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, in quella virtuale (e quindi va caricato) oppure è un indirizzo non lecito (va oltre la dimensione del programma)
  - In questi ultimi due casi il sistema genera il **Page Fault Trap**



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 interruzione è dovuta ad un "insuccesso" nella scelta delle pagine da caricare in memoria

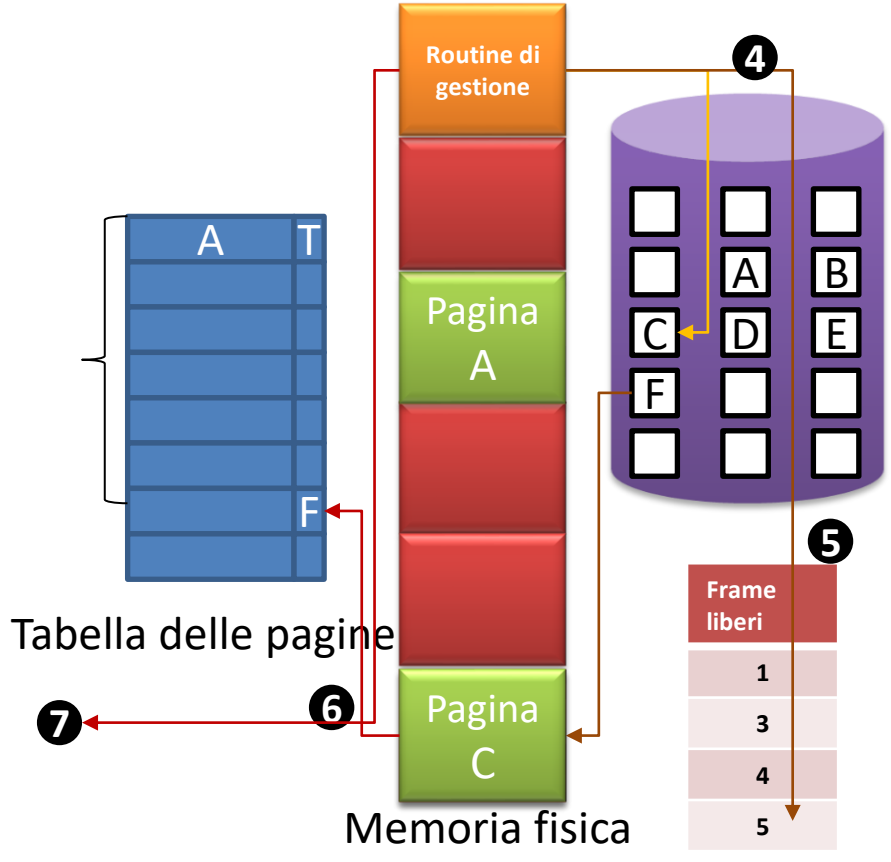


MEMORIA VIRTUALE

Funzionamento

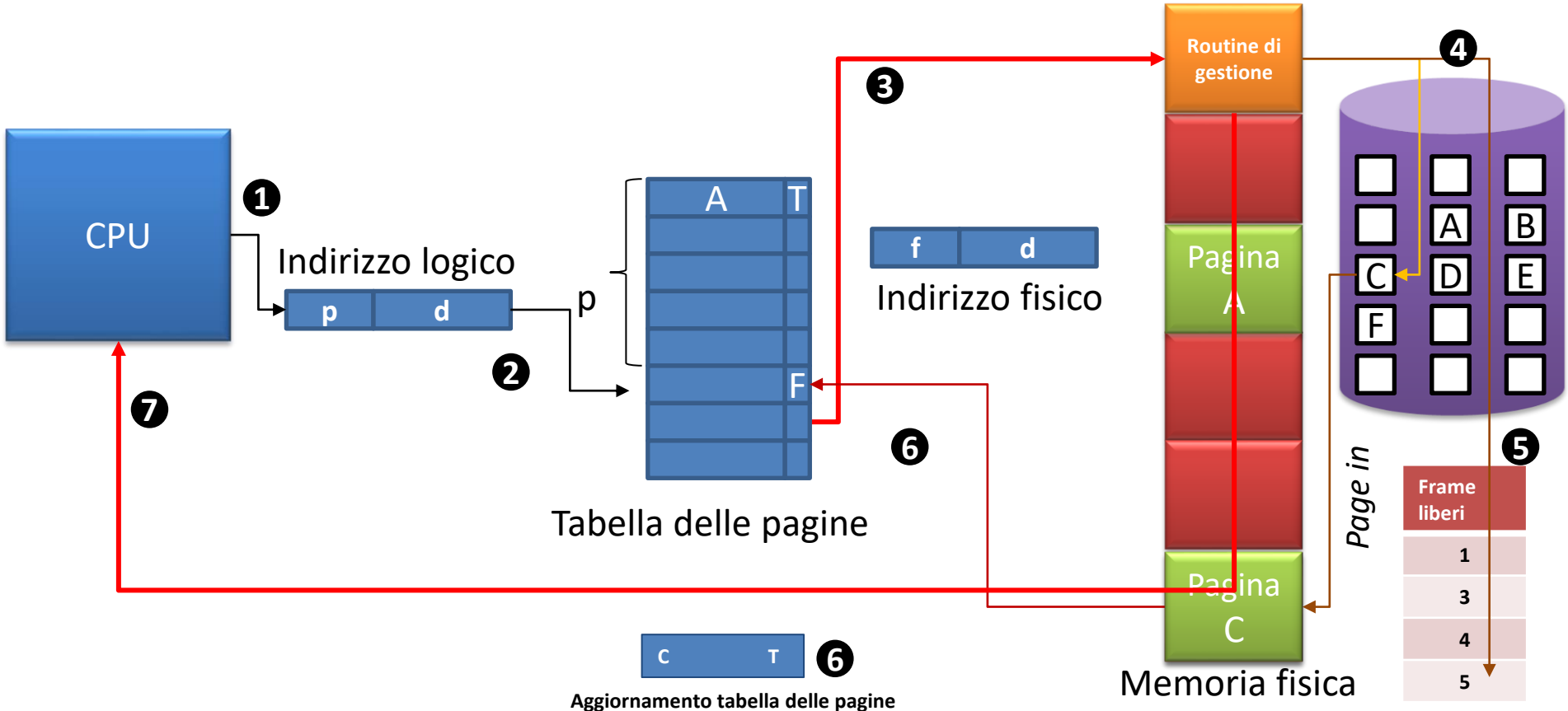
- La gestione dell'eccezione è così espletata:
  - Si controlla la tabella del processo, la PCB, per verificare che l'indirizzo richiesto sia valido oppure no (e quindi se va fuori dai limiti imposti al programma stesso)
  - Se il riferimento non è valido si termina il processo, se invece è valido lo 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 può ora accedere (**7**)

Osservazione: è possibile caricare un processo tutto all'interno della Memoria Virtuale e quindi senza pagine valide (**paginazione pura**) ma si ha una perdita di efficienza



MEMORIA VIRTUALE

Funzionamento



MEMORIA VIRTUALE

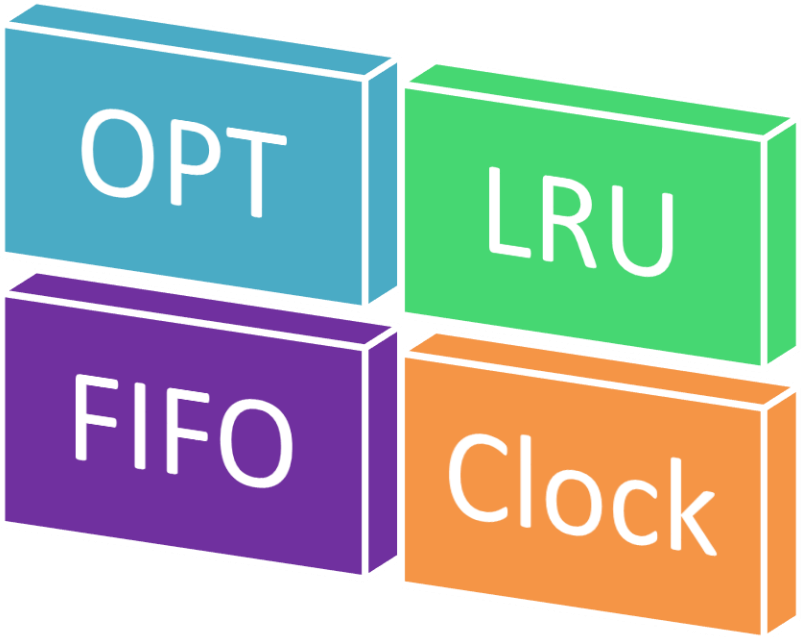
Politiche di caricamento

- Quando c'è un riferimento ad una pagina non presente in Memoria Centrale si innesca il meccanismo di **prelievo (fetch from virtual memory)**, oppure di **rimpiazzo** (**replacement**)
- Ci sono due tipi di **politiche di prelievo**:
  - **On-Demand Paging**: carica la pagina appena se ne ne richiede; inizialmente provoca molti page fault;
  - **Prepaging**: carica diverse pagine contigue (principio di località)
    - Spesso le pagine risiedono in maniera contigua anche nella Memoria Virtuale (es.: **tracce contigue nel disco magnetico**) in modo tale da avvantaggiare i tempi di recupero e di trasferimento

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)

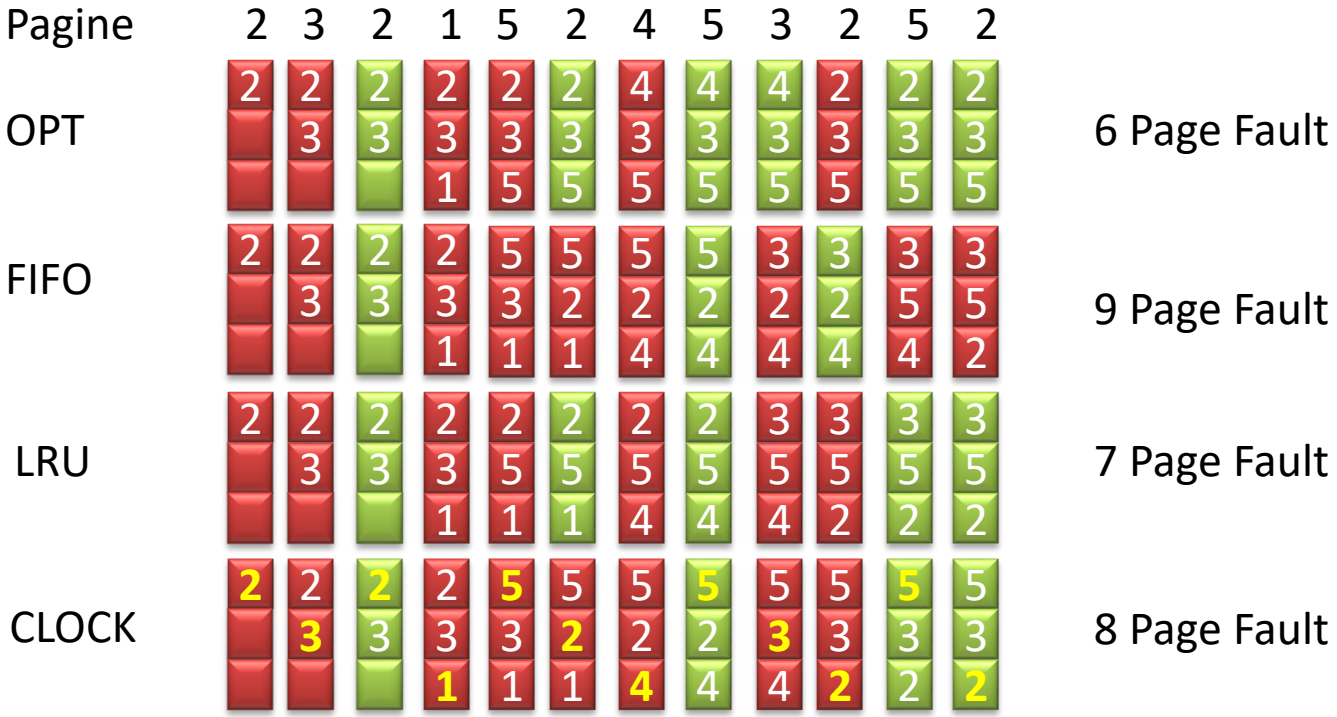
LRU (Least Recently Used)
Sostituisce la **pagina meno usata di recente**
Ad ogni pagina si associa il tempo in cui è stata utilizzata l'ultima volta; quando c'è un page fault si sostituisce la pagina che non è stata utilizzata per il periodo di tempo più lungo. Si mette un contatore di uso ad ogni riga nella tabella delle pagine

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. Si mette un time-stamp ad ogni riga della tabella delle pagine
Semplice ma non sfrutta il principio di località

CLOCK
Ogni pagina, organizzata in una coda circolare, ha un **bit di rifermento** 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



Fine