Memoria virtuale e paginazione

Richiami sulle gerarchie di memoria

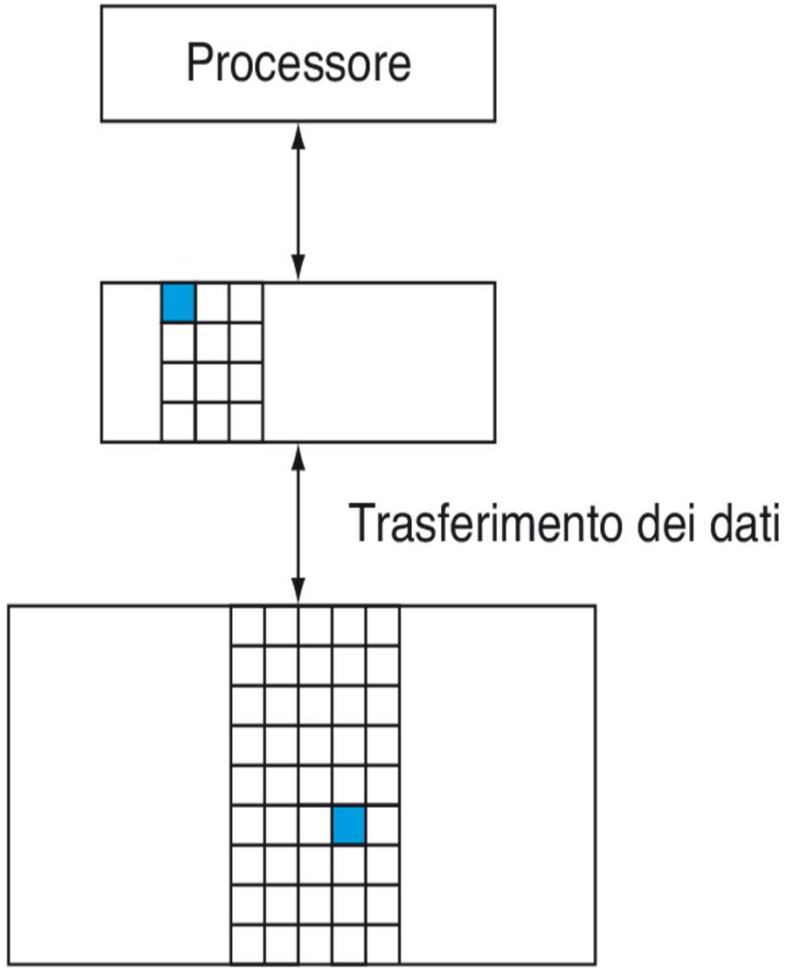
- Esistono attualmente diverse tecnologie per realizzare le memorie, ma nessuna tecnologia ci permette di soddisfare allo stesso tempo ai 2 requisiti fondamentali ipotizzati nella costruzione del processore:
  - accedere a una parola di memoria in un solo ciclo di clock del processore
  - essere così grande da contenere qualsiasi programma (cioè avere uno spazio di indirizzamento fisico uguale allo spazio di indirizzamento virtuale)
- Per superare questo limite si realizza una “gerarchia di memoria”

Idea alla base di una gerarchia di memoria a 2 livelli

- Consideriamo 2 tecnologie di memoria:
  - tecnologia A: permette di realizzare memorie più piccole e veloci
  - tecnologia B: permette di realizzare memorie più grandi ma più lente
- Idea base:
  - tenere il programma nella memoria B e portare nella memoria A solo le porzioni di programma che servono durante una certa fase di esecuzione
- Questa idea funziona solamente se il programma usa relativamente a lungo la porzione di programma contenuta nella memoria veloce prima di richiedere di sostituirla con un'altra

Una gerarchia a 2 livelli

Il processore lavora sulla memoria A, più piccola e veloce
Quando richiede un indirizzo non presente in memoria A, un intero blocco viene trasferito dalla memoria B alla memoria A



Località: comportamento dell'accesso a memoria dei programmi

La distribuzione degli accessi di un programma alla memoria nello spazio e nel tempo non è omogenea, ma presenta la caratteristica di **località**

- **Località temporale**: indica che un programma tende ad accedere agli stessi indirizzi di memoria cui ha già acceduto di recente (cicli di istruzioni)

- **Località spaziale**: indica che un programma accede con maggiore probabilità agli indirizzi vicini a quelli cui ha già acceduto in passato (strutture dati lineari)
- Questo comportamento è esibito praticamente da tutti i programmi e rende efficace l'idea base della gerarchia di memoria

Funzionamento dell'idea base

L'idea base enunciata si concretizza nel comportamento seguente:

- Carica il programma nella memoria B
- Quando il programma richiede una parola a un certo indirizzo, trasferisci un BLOCCO di parole che contengono l'indirizzo richiesto dalla memoria B alla memoria A
- Grazie alla località il programma chiederà di accedere molte volte a parole del blocco caricato in memoria A; questi accessi sono detti HIT
- Prima o poi il programma chiederà di accedere a un indirizzo non presente in memoria A; questo accesso è detto MISS
- Quando si verifica un MISS un nuovo blocco viene caricato dalla memoria B alla memoria A

Funzionamento dell'idea base - 2

- La memoria veloce non è costituita da un unico blocco, ma da molti blocchi, perchè la località temporale può riguardare dati non contigui in memoria
- Fino a quando la memoria veloce ha blocchi liberi, in risposta a un MISS viene caricato un nuovo blocco nello spazio libero
- Quando però la memoria veloce è tutta occupata, per caricare un blocco è necessario scegliere un blocco già caricato da liberare
- La modalità di scelta del blocco da liberare è detta **politica di sostituzione**

Politica di sostituzione

- È possibile definire diverse politiche di sostituzione:
  - Casuale
  - Least Recently Used (LRU): sostitusce il blocco utilizzato meno di recente, perché, in base al principio di località spaziale, questo ha la probabilità più bassa di essere richiesto nel prossimo futuro
  - FIFO (First In First Out): sostituisce sempre il blocco caricato meno di recente, indipendentemente da quando si sia fatto riferimento a tale pagina
- Statisticamente il comportamento migliore sarebbe ottenuto con LRU, però la realizzazione esatta di LRU è difficile, quindi si utilizzano approssimazioni di diverso tipo in base al livello di gerarchia

Perché l'idea base funziona in pratica

- Definiamo i seguenti parametri relativi all'accesso di un programma alla memoria organizzata su due livelli:
  - h = HIT rate (tasso di hit): è la percentuale di accessi che vengono soddisfatti dalla memoria A
  - m = MISS rate (tasso di miss): è la percentuale di accessi che ha richiesto di caricare un nuovo blocco dalla memoria B alla memoria A
  - chiaramente: m = 1 – h
  - M = MISS penalty, è il tempo richiesto per svolgere l'operazione di caricamento di un blocco dal livello di memoria più lento
- L'idea base funziona in pratica perchè la località dei programmi permette di raggiungere, su memorie veloci, HIT rate sufficientemente elevati da compensare ampiamente le MISS penalty

Gerarchie a più livelli

- Attualmente i programmi utilizzano spazi di indirizzamento (virtuali) tanto grandi e i processori hanno bisogno di memorie così veloci, che una gerarchia a 2 livelli non permette di soddisfare i requisiti
- Si costruiscono quindi gerarchie di memoria con più di 2 livelli; per ogni coppia di livelli si applicano **in linea di principio** le considerazioni fatte per la gerarchia a 2 livelli
- Tuttavia i meccanismi differiscono nei dettagli, perchè tra coppie di livelli più in alto (più veloci) nella gerarchia si devono usare meccanismi più veloci (totalmente HW) mentre a livelli più bassi si utilizzano meccanismi più lenti (che comperndono interventi del Sistema Operativo)

Tecnologie di memoria

- Attualmente le tecnologie utilizzate nel costruire la gerarchia di memoria nei calcolatori sono, in ordine di velocità crescente e dimensione decrescente:
  - SRAM (Static Random Access Memories) - volatile
  - DRAM (Dynamic Random Access Memories) - volatile
  - Flash – non volatile
  - Dischi Magnetici – non volatile
  - nuove tecnologie in arrivo (RAM resistive – non volatili …)

livello	velocità	tipiche dimensioni
SRAM	0,2 – 2,5 ns (nano s)	decine di Kbyte
DRAM	50 – 70 ns (nano s)	alcuni Gbyte
Flash	5 – 50 μs (micro s)	centinaia Gbyte
Dischi	5 – 20 ms (milli s)	molti Tbyte

Livelli delle gerarchie di memoria e tecnologie

Livello	Tecnologia	Nome dell'Unità trasferita	Dimensione tipica dell'unità trasferita	Dimensione tipica della memoria
Cache (1° liv)	SRAM			16 – 64 Kb
Cache (2° liv)	SRAM (+ lente e + grandi)	Blocco	16 - 64	125 – 2000 Kb
Memoria Centrale	DRAM	Blocco	64 - 128	< 4 Gb
Memoria Virtuale	**Flash o Disco**	**Pagina**	**> 4 Kbyte**	

Spazi di indirizzamento e dimensioni

Spazio di indirizzamento

- Lo spazio di indirizzamento è definito dal numero di parole indirizzabili e dipende esclusivamente dal numero di bit dell'indirizzo
- Si deve distinguere lo spazio di indirizzamento dalla **dimensione effettiva**

- La dimensione effettiva della memoria fisica è pari al n° di byte (parole) che la costituiscono materialmente:
  - se N sono i bit di indirizzo, allora 2^N è il numero massimo di parole indirizzabili (2^N è lo spazio di indirizzamento della memoria fisica)
  - poiché la memoria fisica deve essere tutta indirizzabile è sempre minore o uguale al suo spazio di indirizzamento

Indirizzo virtuale

Gli indirizzi virtuali sono quelli generati dal collegatore (linker) (a partire dall'indirizzo 0)

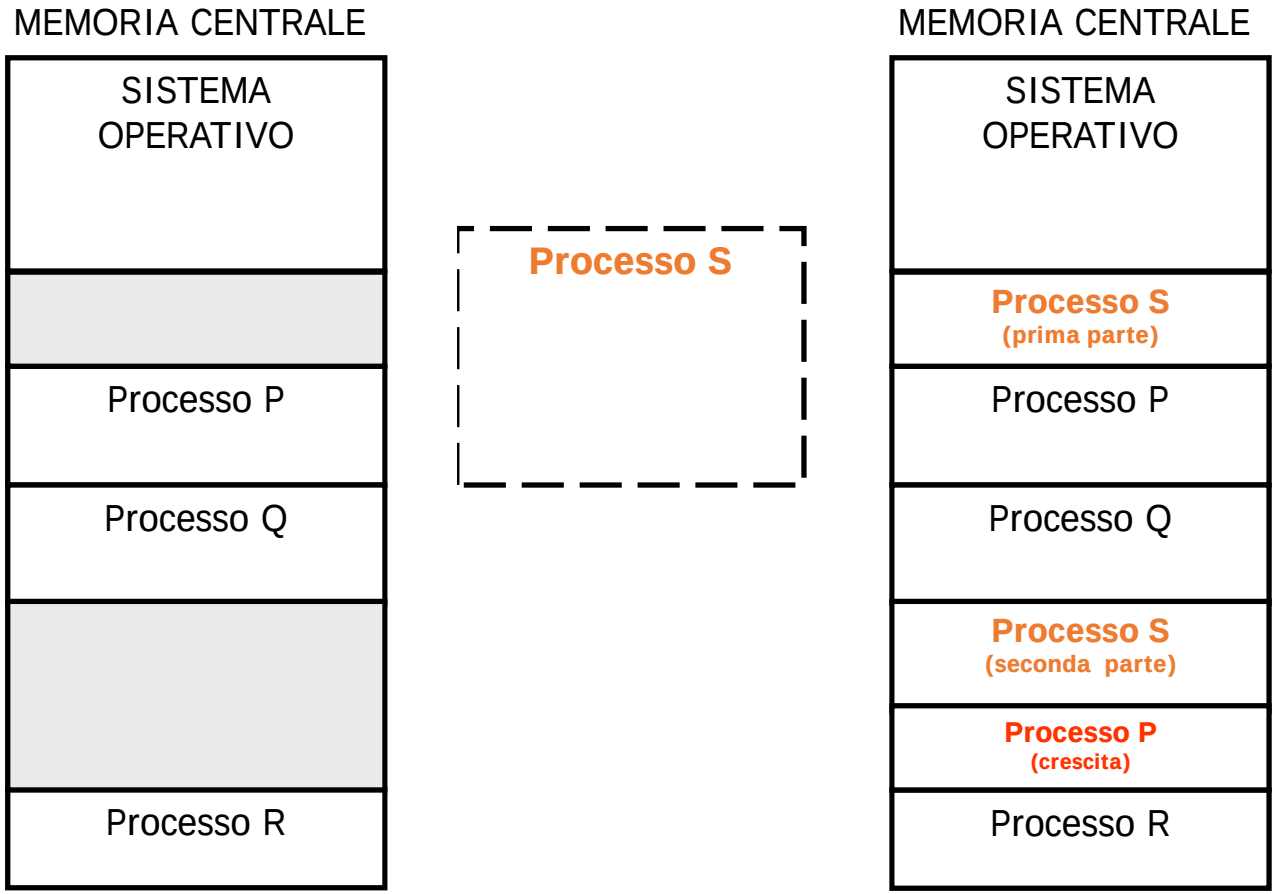
- Lo **spazio di indirizzamento virtuale** è quello definito dagli indirizzi virtuali
- In un programma è possibile definire la **dimensione virtuale** (in termini di indirizzi virtuali):
  - è la somma delle aree di programma effettivamente presenti
  - solo gli indirizzi virtuali appartenenti a queste aree sono validi, gli altri sono non validi
  - dimensione iniziale: è quella calcolata dal linker dopo la generazione del codice eseguibile
  - può variare durante l'esecuzione a causa:
    - della modifica della *pila* per le chiamate a sottoprogrammi
    - a causa della modifica dello *heap* per la gestione delle strutture dati dinamiche
    - per la creazione di aree per la mappatura di librerie dinamiche o aree condivise con altri processi
- Per esempio
  - in x64, la lunghezza degli indirizzi è di 64 bit
  - lo spazio virtuale disponibile è di 2^{48} byte, quindi solo 48 dei 64 bit di indirizzamento sono utilizzati effettivamente per costruire indirizzi virtuali validi

Principali requisiti nella gestione della memoria
1) rilocazione dinamica

- al momento dell'esecuzione il programma viene caricato in memoria e opera con gli indirizzi virtuali, cioè come se fosse caricato a partire dall'indirizzo 0
- La **traduzione automatica degli indirizzi** virtuali in indirizzi fisici (**rilocazione**) è **attuata durante l'esecuzione** del programma a ogni riferimento a memoria **(dinamica):**
  - è svolta da un componente Hardware detto **MMU**
  - è completamente trasparente al programmatore, al compilatore e al collegatore (linker)
- questa traduzione consente il caricamento del programma in una **qualsiasi locazione** della memoria fisica
- Il programmatore può scrivere i suoi programmi pensando di avere a disposizione **l'intero spazio di indirizzamento virtuale**

Principali requisiti nella gestione della memoria
2) esigenza di porzioni non contigue dei processi per evitare la frammentazione della memoria

frammentazione della memoria: presenza di zone di memoria libera piccole e inframmezzate a zone utilizzate



Principali requisiti nella gestione della memoria
3) rendere non residenti alcune parti dei programmi

Dato che:

- Più processi e il Sistema Operativo possono **risiedere contemporaneamente in memoria** indipendentemente dalle dimensioni effettive della memoria centrale
- la dimensione di memoria virtuale di un singolo programma e/o di più programmi in memoria può essere maggiore della dimensione della memoria fisica.

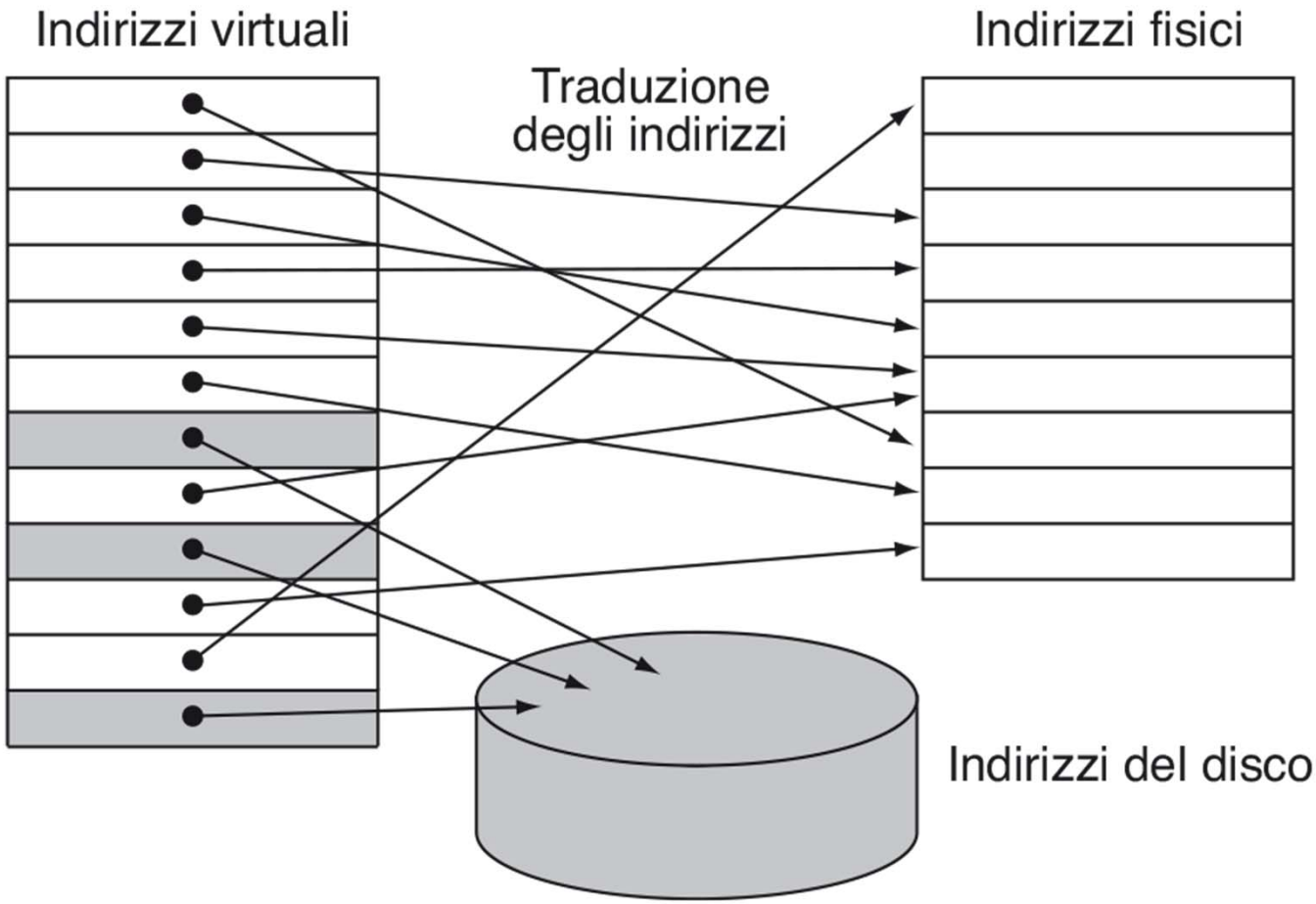
È necessario che:

- un **programma in esecuzione non risieda completamente in memoria**:
  - le parti di programma caricate in memoria sono dette **residenti**
  - in ogni momento vengono rese residenti, cioè sono caricate in memoria, le parti di programma che servono per l'esecuzione
  - il meccanismo di caricamento in memoria delle **parti del programma** e la **traduzione degli indirizzi** siano **trasparenti** al programmatore
- per un buon funzionamento, le parti residenti devono essere sufficienti a produrre uno Hit Rate adeguato

Paginazione

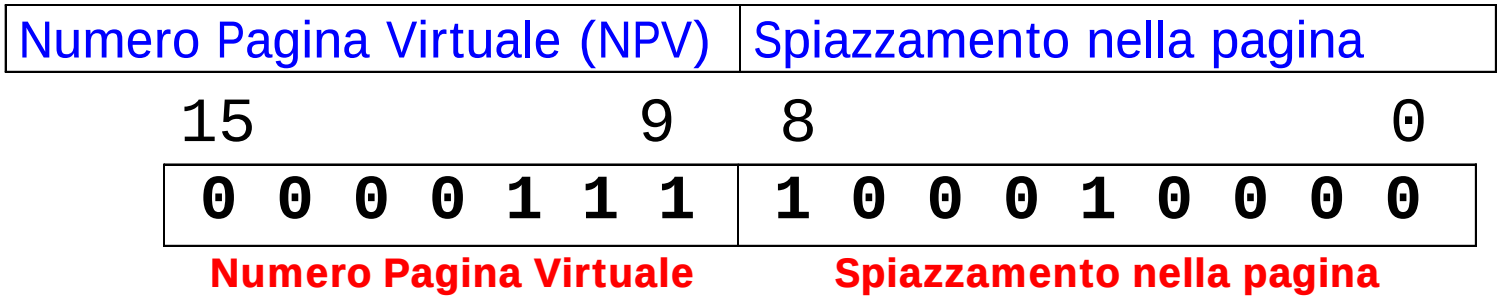
- Il meccanismo fondamentale per ottenere i comportamenti desiderati è costituito dalla **paginazione**
- Tutti i meccanismi di gestione della memoria virtuale rilocano il programma come un insieme di **blocchi di dimensione fissa** (**pagine**), eliminando così la necessità di individuare un blocco di memoria contigua sufficientemente ampio da potervi caricare il programma
- Il sistema operativo ha solo bisogno di trovare un numero sufficiente di pagine (non necessariamente contigue) disponibili nella memoria fisica
- La paginazione ha l'effetto di ridurre il fenomeno della **frammentazione della memoria**
- È possibile gestire facilmente la **crescita di memoria di un processo durante l'esecuzione**
- È possibile rendere **Residenti** o **Non Residenti** singole pagine del processo

Paginazione – rappresentazione intuitiva



Paginazione: scomposizione dell'indirizzo virtuale

- Lo spazio di **indirizzamento virtuale** di ogni programma è suddiviso in un **numero** intero di **pagine** di dimensione fissa e potenza di 2:
  - dimensioni di pagina tipiche: da 512 byte a 64 Kbyte
- L'indirizzo virtuale può essere visto come illustrato dal seguente esempio:
  - spazio di indirizzamento virtuale di un programma 64 K (16 bit di indirizzo virtuale);
  - pagine da 512 byte (9 bit di indirizzo per spiazzamento (o offset) nella pagina)



Esempio di scomposizione degli indirizzi esadecimali

- l'indirizzo 0000 1111 0001 0000 viene rappresentato come 0x 0F10
- se la pagina è di 4K (12 bit) abbiamo la seguente scomposizione in NPV / offset

  0000 /1111 0001 0000, rappresentato come 0x 0 / F10
- attenzione: se la dimensione dello spiazzamento non è un multiplo di 4 la scomposizione in NPV / spiazzamento è più complessa:
- Esempio: con pagina di 512 byte (9 bit di spiazzamento) lo stesso indirizzo diventa

  0000 111 / 1 0001 0000 rappresentato come 0x 07 / 110

Pagine fisiche

- Lo spazio di **indirizzamento fisico** (ossia della memoria centrale) viene suddiviso in un **numero** intero di **pagine** di uguale dimensione di quelle utilizzate per lo spazio di indirizzamento virtuale
- Ogni pagina della memoria può quindi contenere esattamente una pagina dello spazio di indirizzamento virtuale

- l'indirizzo fisico può essere visto come:

Numero Pagina Fisica (NPF)	Spiazzamento nella pagina

Traduzione dell'indirizzo virtuale in indirizzo fisico: è sufficiente trasformare NPV in NPF

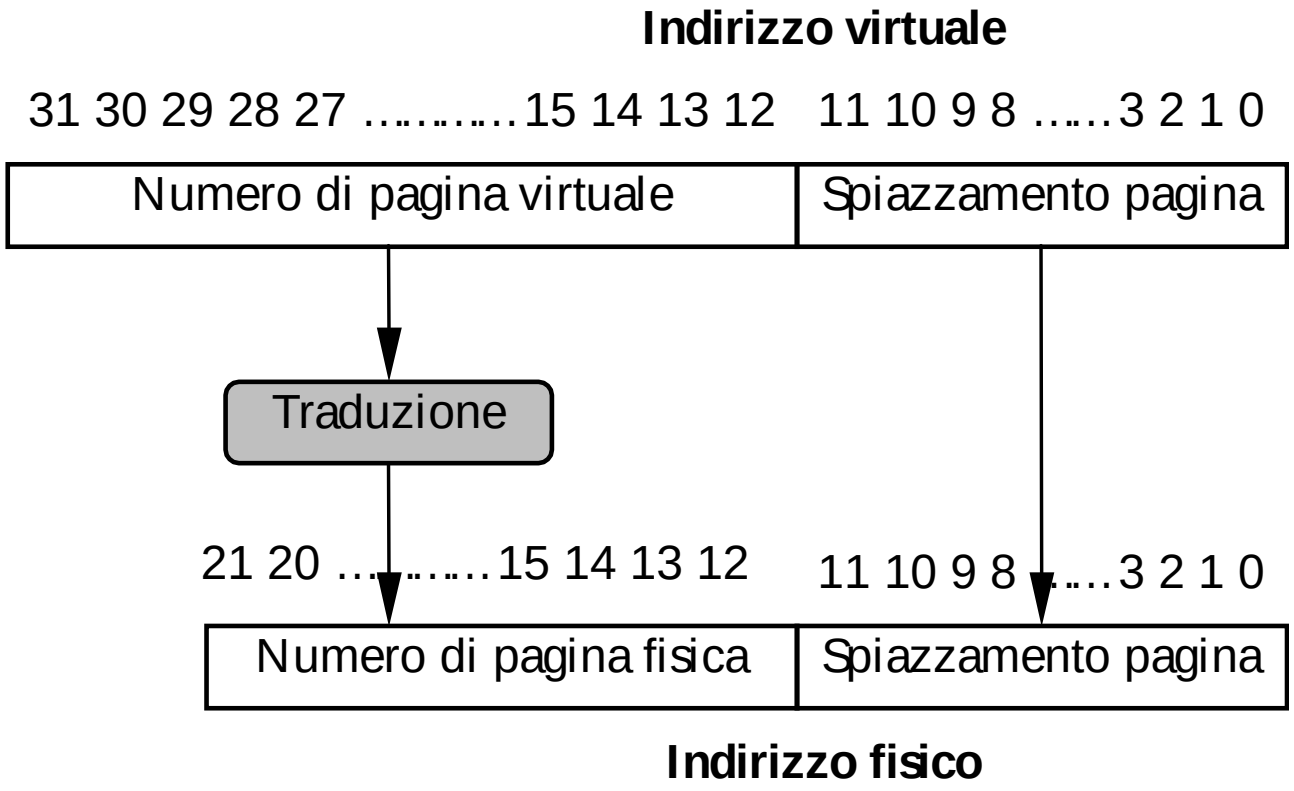
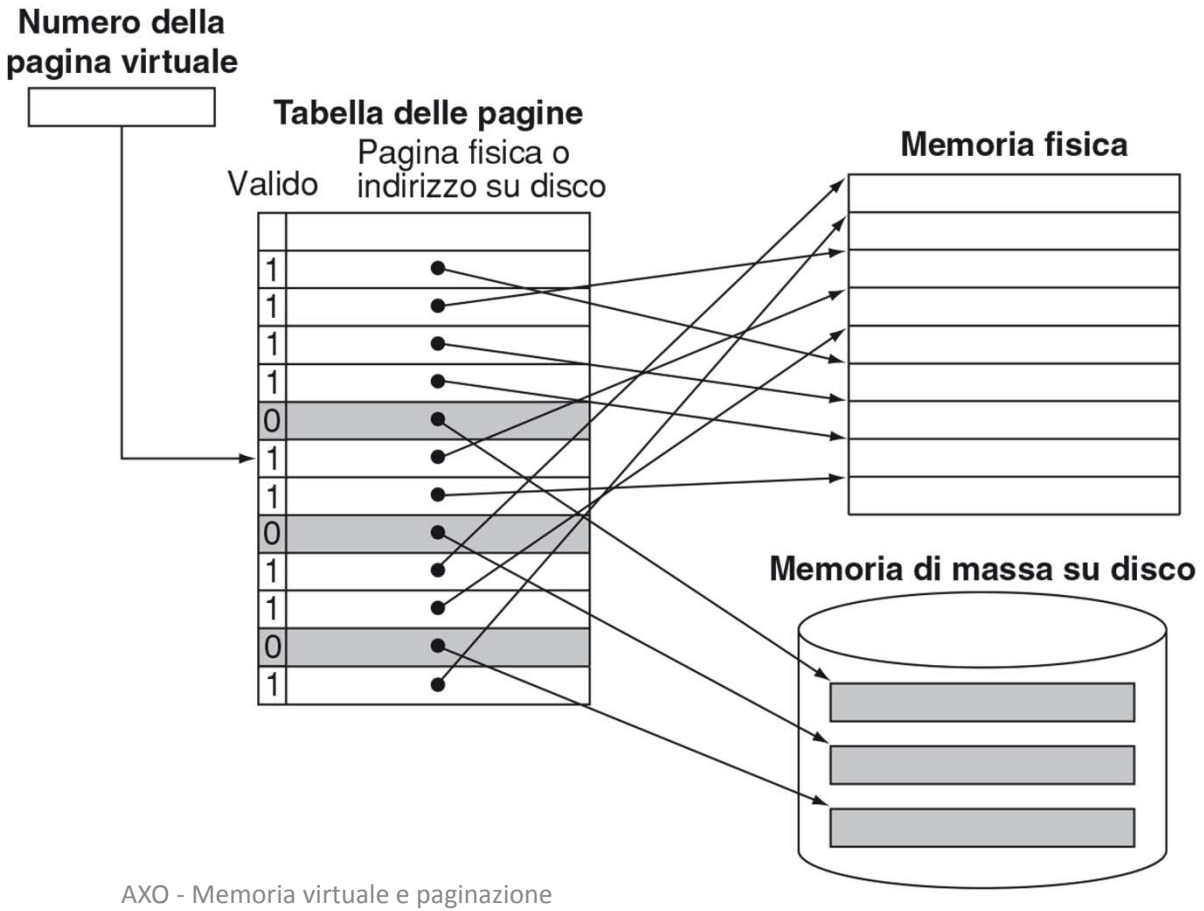


Tabella delle pagine

- La corrispondenza tra pagine virtuali e pagine fisiche è rappresentata da una **tabella delle pagine** associata al processo:
  - **esiste una tabella delle pagine per ogni processo**
  - la tabella delle pagine appartiene concettualmente al descrittore del processo
- La tabella delle pagine deve, in linea di principio, contenere una riga per ogni pagina virtuale dello spazio di indirizzamento del processo
- In questa ipotesi, il numero di pagina virtuale (**NPV**) può essere utilizzato come indice nella tabella delle pagine del processo.
- Se il processo ha una dimensione virtuale inferiore allo spazio di indirizzamento alcune di queste pagine non esistono, cioè **non sono valide**
- La struttura della tabella delle pagine viene in realtà ottimizzata per evitare di rappresentare un numero eccessivo di pagine non valide
- Le pagine valide caricate sono dette **residenti**

Tabella delle pagine – rappresentazione intuitiva



Esempio: Corrispondenza tra pagine fisiche e pagine logiche

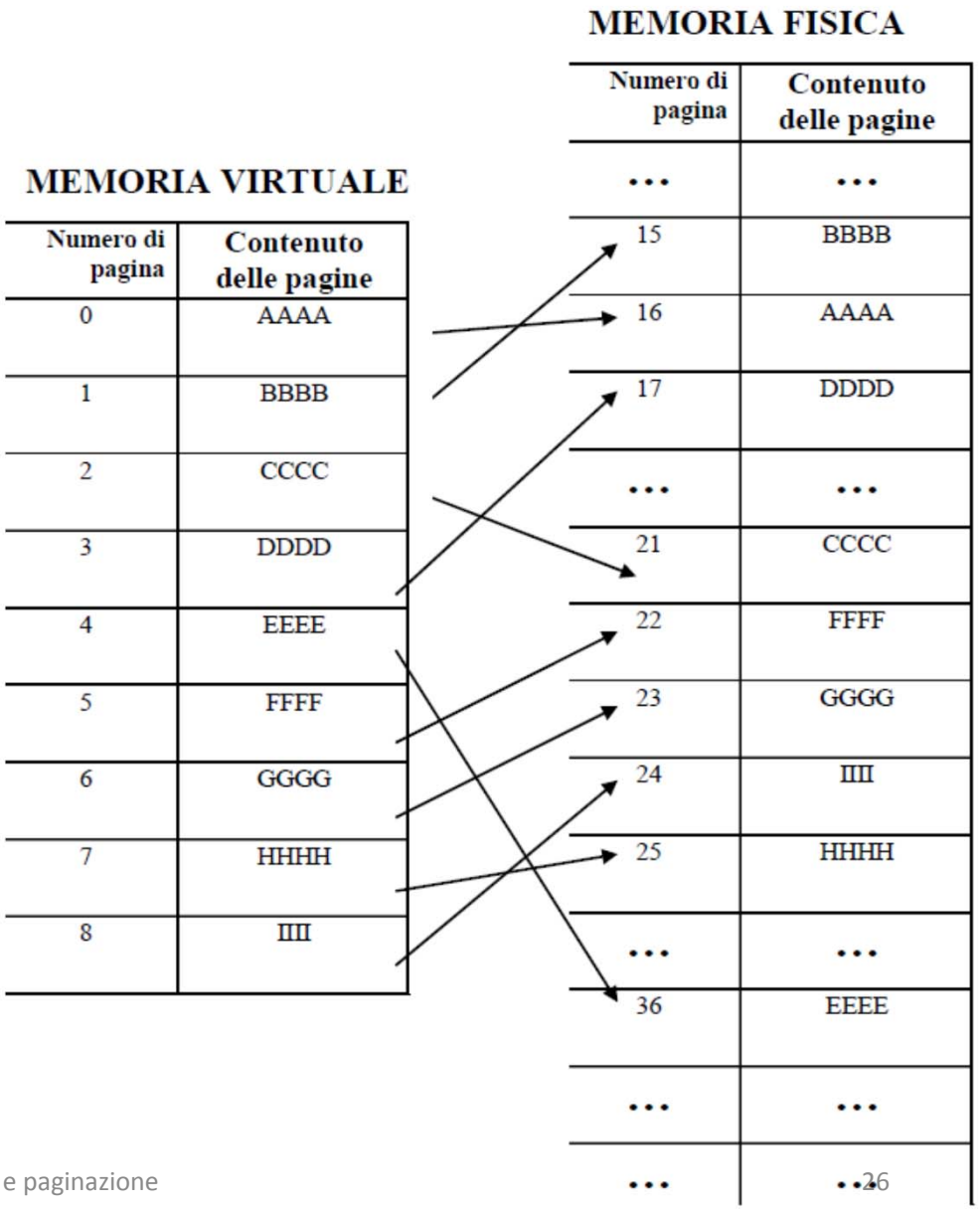


Tabella delle pagine che realizza la corrispondenza (mapping) dell'esempio precedente

MEMORIA VIRTUALE

Numero di pagina	Contenuto delle pagine
0	AAAA
1	BBBB
2	CCCC
3	DDDD
4	EEEE
5	FFFF
6	GGGG
7	HHHH
8	IIII

TABELLA DELLE PAGINE

NPV	NPF
0	16
1	15
2	21
3	17
4	36
5	22
6	23
7	25
8	24

MEMORIA FISICA

Numero di pagina	Contenuto delle pagine
...	...
15	BBBB
16	AAAA
17	DDDD
...	...
21	CCCC
22	FFFF
23	GGGG
24	IIII
25	HHHH
...	...
36	EEEE
...	...
...	...

Memoria Virtuale di P

Numero di pagina	Contenuto delle pagine
0x00000	AAAA
0x00001	BBBB
0x00002	CCCC
0x00003	DDDD

Tabella Pagine di P

NPV	NPF
0x00000	0x00004
0x00001	0x00005
0x00002	0x00006
0x00003	0x00007

MEMORIA FISICA

Numero di pagina	Contenuto delle pagine
0x00000	S.O.
0x00001	S.O.
0x00002	S.O.
0x00003	S.O.
0x00004	AAAA
0x00005	BBBB
0x00006	CCCC
0x00007	DDDD
0x00008	RRRR
0x00009	SSSS
0x0000A	TTTT
0x0000B	UUUU
0x0000C	VVVV
0x0000D	non usata
0x0000E	non usata
0x0000F	non usata

Memoria Virtuale di Q

Numero di pagina	Contenuto delle pagine
0x00000	RRRR
0x00001	SSSS
0x00002	TTTT
0x00003	UUUU
0x00004	VVVV

Tabella Pagine di Q

NPV	NPF
0x00000	0x00008
0x00001	0x00009
0x00002	0x0000A
0x00003	0x0000B
0x00004	0x0000C

Tabelle delle pagine di 2 processi P e Q

Condivisione delle pagine

- I diversi processi possono **condividere delle pagine** (tipicamente di codice o di libreria, ma anche di dati, tramite opportuni meccanismi messi a disposizione dal S.O.):
  - le pagine condivise sono **presenti una sola volta nella memoria fisica**.
- Per **ogni pagina condivisa, esiste una riga** (cioè un NPV) nella corrispondente tabella delle pagine di **ogni processo che la condivide** e i valori di NPF relativi alle pagine condivise sono identici
- Esempio: i 2 processi P e Q condividono le pagine virtuali 200 e 150 nella pagina fisica 13:
  - TP di P contiene < 200, **13**>
  - TP di Q contiene < 150, **13**>
- Nota bene: le pagine virtuali possono essere diverse

Condivisione delle pagine – esempio

MEMORIA VIRTUALE di P

Numero di pagina	Contenuto delle pagine
0x00000	AAAA
0x00001	BBBB
0x00002	CCCC
0x00003	DDDD

TABELLA delle PAGINE di P

NPV	NPF
0x00000	0x00004
0x00001	0x00005
0x00002	0x00006
0x00003	0x00007

MEMORIA VIRTUALE di Q

Numero di pagina	Contenuto delle pagine
0x00000	AAAA
0x00001	BBBB
0x00002	TTTT
0x00003	UUUU
0x00004	VVVV

TABELLA delle PAGINE di Q

NPV	NPF
0x00000	0x00004
0x00001	0x00005
0x00002	0x0000A
0x00003	0x0000B
0x00004	0x0000C

MEMORIA FISICA

Numero di pagina	Contenuto delle pagine
0x00000	S.O.
0x00001	S.O.
0x00002	S.O.
0x00003	S.O.
0x00004	AAAA
0x00005	BBBB
0x00006	CCCC
0x00007	DDDD
0x00008	non usata
0x00009	non usata
0x0000A	TTTT
0x0000B	UUUU
0x0000C	VVVV
0x0000D	non usata
0x0000E	non usata
0x0000F	non usata

Protezione delle pagine

- È possibile associare a ogni pagina virtuale di un processo alcuni **bit di protezione**, che definiscono le modalità di accesso (diritti) consentite per quella pagina; normalmente si usano i seguenti tipi di protezione
  - Solo Lettura (**R** o RO)
  - Lettura e Scrittura (**W** o RW)
  - Esecuzione (**X**) (per pagine di codice)

Gestione delle pagine virtuali non residenti in memoria

al momento dell'accesso a memoria, la mancanza della pagina nella memoria è detta **Page Fault**; in tal caso la pagina va caricata da disco

- in caso di **Page Fault**, tramite un interrupt apposito il controllo passa al **sistema operativo**
- il processo in esecuzione viene sospeso e il sistema operativo deve:
  - rintracciare su disco la pagina virtuale richiesta: il S.O. utilizza per questo le tabelle delle pagine che contengono, per le pagine fuori memoria, il riferimento alla loro posizione su disco;
  - trovare spazio in memoria per caricare la pagina richiesta. Quest'operazione può implicare di dover scaricare da memoria altre pagine. Le tabelle delle pagine dei processi coinvolti devono essere aggiornate;
  - caricare la pagina da disco;
  - far rieseguire l'istruzione macchina che aveva generato l'errore di pagina (richiede restartable instructions)

Caricamento iniziale di un programma: Demand Paging

- Esecuzione di un nuovo programma:
  - nessuna pagina del processo si trova in memoria.
  - Quando la CPU cerca di accedere alla prima istruzione si verifica un errore di pagina, e la prima pagina viene portata in memoria e registrata nella tabella delle pagine.
  - Ogni volta che si identifica un indirizzo in una pagina non ancora in memoria, si verifica un errore di pagina e la pagina corrispondente viene caricata in memoria senza scaricare le pagine già residenti finché non si è raggiunto il numero R di pagine residenti
  - In genere, oltre alla pagina che contiene l'istruzione iniziale del programma viene richiesta la pagina di pila, perché i parametri sono passati al main sulla pila

Sostituzione e file di swap

- Nel caso di scrittura, la modifica avviene solo nelle pagine in memoria e non riportata immediatamente sulle pagine su disco
- In caso di sostituzione di una pagina è necessario individuare se la pagina da scaricare sia stata o meno modificata:
  - Se la pagina è stata modificata è necessario copiarla su disco prima di sostituirla.
  - per contenere tali pagine esiste il **file di swap.**
- Nella MMU si associa un bit (**dirty bit**) a ogni pagina fisica per indicare se la pagina che contiene sia stata o meno modificata:
  - quando una pagina deve essere sostituita il S.O. verifica il bit di modifica per stabilire se la pagina debba essere copiata su disco.