

# Gestione Memoria

## Gestione della memoria

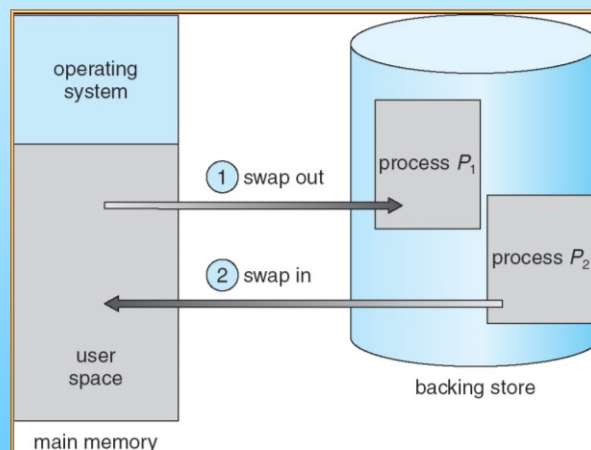
- Il sistema operativo ha il compito di:
  - Assegnare ad ogni processo la memoria di cui ha bisogno per la sua esecuzione
  - Isolare i processi facendo in modo che i processi non possano accidentalmente o in modo malevolo accedere o modificare memoria riservata ad altri processi o al sistema operativo stesso
  - Utilizzare la memoria nel modo più efficiente possibile in modo da poter aumentare il grado di multiprogrammazione
  - Permettere quando questo sia richiesto esplicitamente la condivisione di zone di memoria tra i processi

## Swapping

- Un processo può essere tolto (swapped) temporaneamente dalla memoria e portato su disco (backing store), e quindi riportato in memoria successivamente per continuare l'esecuzione
- **Backing store** – disco veloce abbastanza grande per memorizzare tutte le copie delle immagini di memoria di tutti gli utenti; deve dare accesso diretto a queste immagini di memoria
- **Roll out, roll in** – variante di swapping usata con scheduling a priorità; processi a bassa priorità sono *swapped out* in modo che processi ad alta priorità possano essere caricati e eseguiti
- Maggior parte del tempo di swap è il tempo di trasferimento; direttamente proporzionale alla quantità di memoria trasferita
- Versioni modificate dello swapping sono presenti in molti sistemi operativi (i.e., UNIX, Linux, and Windows)

Sistemi Operativi A.A. 2016/2017

## Visione schematica dello Swapping



Sistemi Operativi A.A. 2016/2017

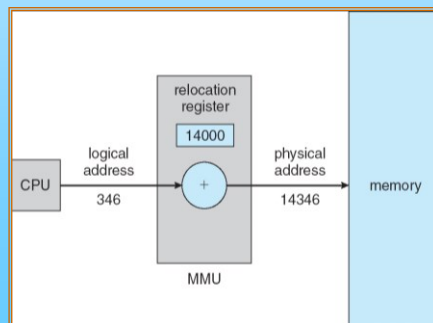
## Spazio indirizzi Logico vs. Fisico

- Il concetto di uno **spazio di indirizzi logico** collegato ad uno **spazio di indirizzi fisico** è fondamentale per una efficace gestione della memoria
  - **Indirizzo logico** – generato dalla CPU; (chiamato anche indirizzo virtuale)
  - **Indirizzo fisico** – indirizzo visto dalla unità di memoria
- Ogni processo ha il proprio spazio di indirizzi logici
- Più spazi di indirizzi logici devono essere associati allo stesso spazio di indirizzi fisico

Sistemi Operativi A.A 2016/2017

## Memory-Management Unit (MMU)

- Dispositivo che mappa indirizzi logici in indirizzi fisici
- Per esempio una **MMU** può usare un *registro di rilocazione* il cui valore viene sommato per ogni indirizzo generato da un processo utente quando questi accede alla memoria
- Il programma utente ha a che fare solo con indirizzi logici non vede mai gli indirizzi fisici reali



Sistemi Operativi A.A 2016/2017

## Gestione Memoria

- Tecniche principali per la gestione della memoria:
  - *Allocazione contigua*
  - *Paginazione*
  - *Segmentazione*
  - *Segmentazione paginata*

Sistemi Operativi A.A 2016/2017

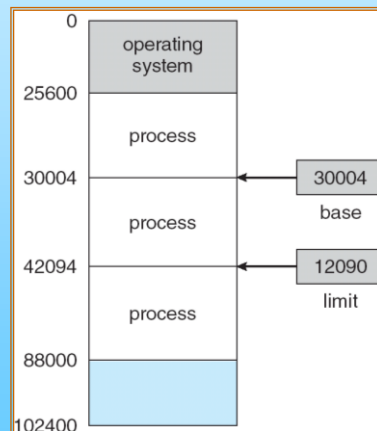
## Allocazione Contigua

- La memoria è solitamente divisa in due parti:
  - Una con codice e dati del sistema operativo solitamente tenuta nella parte bassa della memoria insieme al vettore delle interruzioni.
  - Una usata per tenere i processi degli utenti (solitamente nella parte alta)
- Ogni processo ha associata una sola partizione
  - Registro di rilocalizzazione viene usato per proteggere i processi utenti tra di loro e dal modificare codice e dati del sistema operativo
  - Il registro di rilocalizzazione contiene il valore dell'indirizzo fisico più basso riservato al processo; il registro limite contiene il limite massimo della memoria riservata al processo (ogni indirizzo logico deve essere inferiore a questo valore)

Sistemi Operativi A.A 2016/2017

## Allocazione contigua (Cont.)

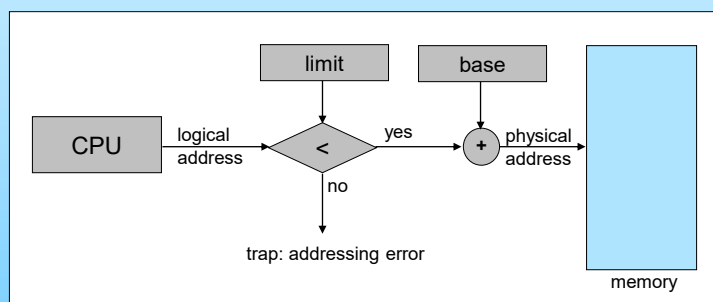
- Registro base e registro limite definiscono lo spazio degli indirizzi del processo



Sistemi Operativi A.A 2016/2017

## Allocazione contigua (Cont.)

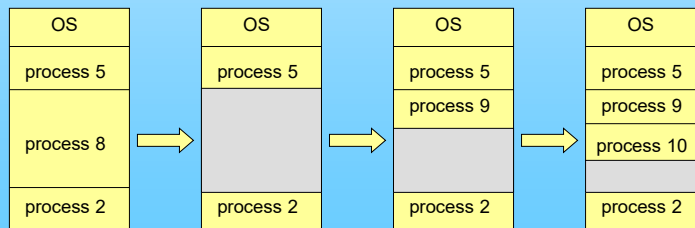
- Protezione dello spazio di memoria



Sistemi Operativi A.A 2016/2017

## Allocazione contigua (Cont.)

- Allocazione di processi multipli
  - *Buco* – blocco di memoria libera; buchi di varie dimensioni sparsi nella memoria
  - Quando un processo arriva è allocato in un buco abbastanza largo per contenerlo
  - Il sistema operativo tiene informazioni su:
    - a) partizioni allocate    b) partizioni libere (buchi)



Sistemi Operativi A.A 2016/2017

## Problema della allocazione dinamica

Come soddisfare una richiesta di n bytes da una lista di buchi liberi?

- **First-fit:** Alloca il primo buco abbastanza grande
- **Best-fit:** Alloca il buco più piccolo abbastanza grande; si deve ricercare nell'intera lista a meno che non sia ordinata per grandezza. Produce il buco libero più piccolo.
- **Worst-fit:** Alloca il buco più grande che può soddisfare la richiesta, deve ricercare l'intera lista. Produce il buco libero più grande.

First-fit e best-fit sono migliori di worst-fit in termini di velocità e utilizzazione di memoria

Sistemi Operativi A.A 2016/2017

## Frammentazione

- **Frammentazione esterna** – esiste spazio di memoria per soddisfare una richiesta ma non è contiguo
- **Frammentazione interna** – la memoria allocata a un processo può essere leggermente superiore di quella richiesta dal processo; questa differenza è interna alla partizione, ma non viene usata
- Si può ridurre la frammentazione esterna tramite la **compattazione**
  - Riordina i contenuti della memoria in modo da posizionare tutta la memoria libera in un unico blocco
  - La compactazione è possibile solo se la rilocalizzazione è dinamica e avviene a tempo di esecuzione
  - Può essere molto oneroso

Sistemi Operativi A.A 2016/2017

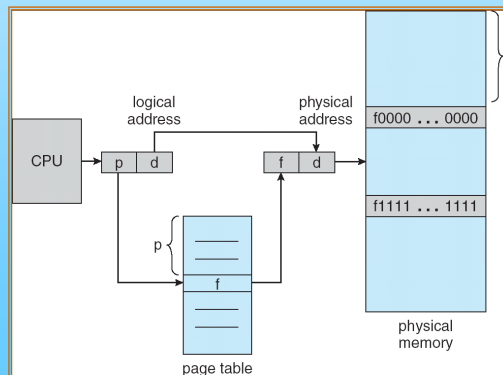
## Paginazione

- Lo spazio degli indirizzi logici di un processo può non essere contiguo; un processo è allocato nella memoria fisica dove questa è libera
- Divide la memoria fisica in blocchi di lunghezza fissa chiamati **frame** (dimensione è potenza del 2, tra 512 byte e 8192 byte)
- Divide la memoria logica di un processo in blocchi di lunghezza uguale chiamate **pagine**.
- Tiene traccia dei frame liberi
- Per eseguire un programma che usa  $n$  pagine, il SO deve trovare  $n$  frame liberi e caricare il programma
- Imposta un **tabella delle pagine** per tradurre un indirizzo logico in un indirizzo fisico
- E' soggetta a *frammentazione interna*

Sistemi Operativi A.A 2016/2017

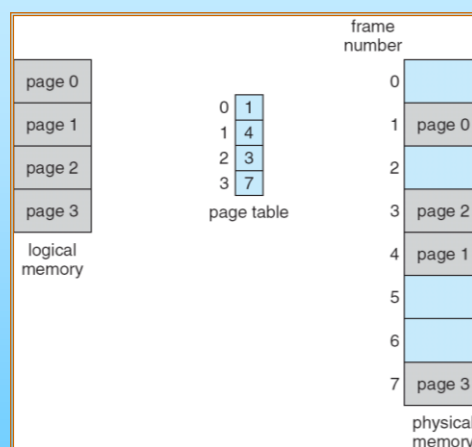
## Schema di traduzione degli indirizzi

- Indirizzo generato dalla CPU è diviso in:
  - *Numero di pagina (p)* – usato come indice nella *tabella delle pagine* che contiene l'indirizzo di base di ogni pagina nella memoria fisica
  - *Scostamento (d)* – combinato con l'indirizzo di base per definire l'indirizzo della memoria fisica che viene inviato alla memoria



Sistemi Operativi A.A 2016/2017

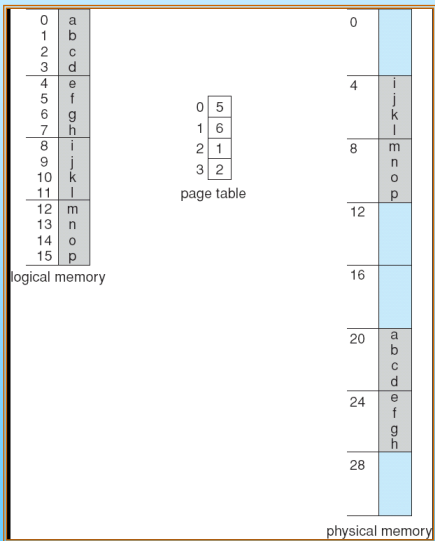
## Esempio



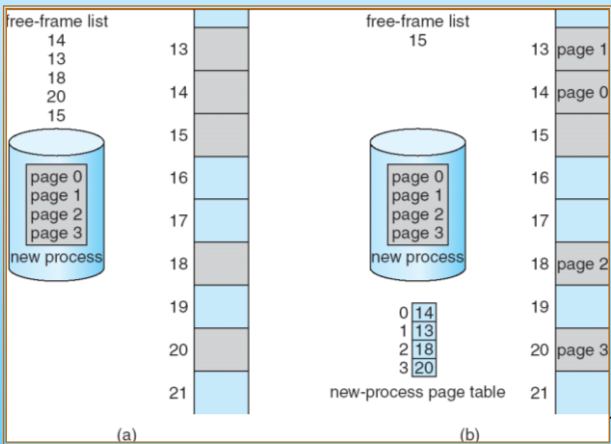
Sistemi Operativi A.A 2016/2017



# Esempio



# Frame liberi



## Implementazione della tabella delle pagine

- La *tabella delle pagine* è tenuta in memoria
- *Registro di base della tabella delle pagine* (*page-table base register* - PTBR) punta alla tabella delle pagine del processo attivo
- In questo schema ogni accesso a dato/istruzione richiede due accessi in memoria. Uno per la tabella delle pagine e uno per il dato/istruzione.
- Il problema del doppio accesso alla memoria può essere risolto usando una piccola cache di ricerca veloce chiamata **memoria associativa** o **TLB** (Translation Look-aside Buffer)

Sistemi Operativi A.A 2016/2017

## Memoria associativa

- Memoria associativa – ricerca parallela

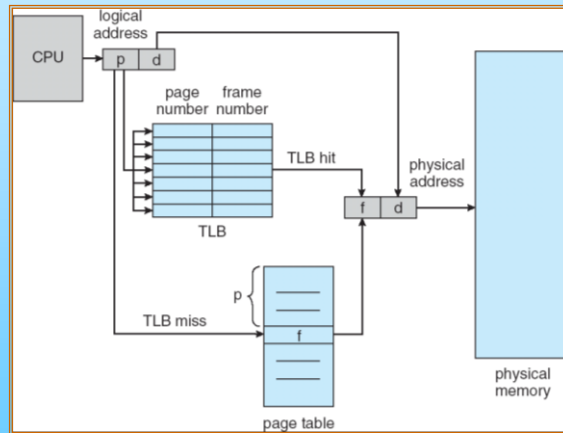
n. pagina	n. frame

Traduzione indirizzo ( $A'$ ,  $A''$ )

- Se  $A'$  è in un registro associativo, preleva il numero di frame
- Altrimenti prende il numero di frame dalla tabella delle pagine in memoria

Sistemi Operativi A.A 2016/2017

## Paginazione con TLB



Sistemi Operativi A.A 2016/2017

## Tempo effettivo di accesso

- Ricerca associativa =  $\varepsilon$  unità di tempo
- Si assume che il ciclo di memoria sia di 1 microsecondo
- Tasso di successi (*Hit ratio*) – percentuale delle volte che un numero di pagina è trovato nei registri associativi; dipende dal numero di registri associativi
- Tasso di successi =  $\alpha$
- **Tempo effettivo di accesso** (EAT)

$$\begin{aligned} \text{EAT} &= (1 + \varepsilon) \alpha + (2 + \varepsilon)(1 - \alpha) \\ &= 2 + \varepsilon - \alpha \end{aligned}$$

- Esempio:

- $\alpha = 80\% = 0,8$
- $\varepsilon = 0,2$  microsecondi
- $\text{EAT} = 2 + 0,2 - 0,8 = 1,4$  microsecondi
- tempo di accesso in memoria aumentato del 40%
- Aumentando  $\alpha$  al 98% si ottiene  $\text{EAT} = 2 + 0,2 - 0,98 = 1,22$  microsecondi quindi un aumento del 22%

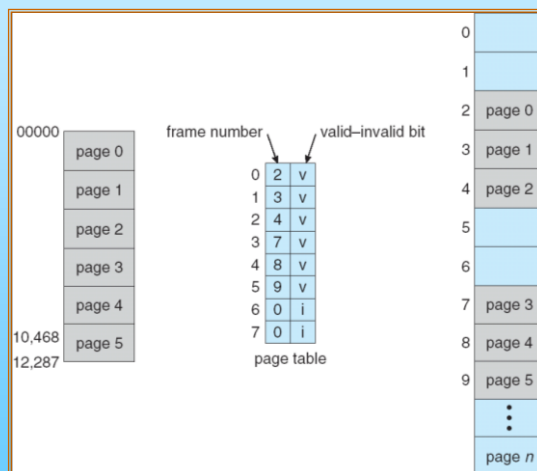
Sistemi Operativi A.A 2016/2017

## Protezione della memoria

- Protezione della memoria implementata associando un bit di protezione a ogni frame
- **Bit valido-invalido** associato a ogni riga nella tabella delle pagine:
  - “valido” indica che la pagina associata è nello spazio degli indirizzi logici del processo, e quindi una pagina valida
  - “invalid” indica che la pagina non è nello spazio degli indirizzi logici del processo.

Sistemi Operativi A.A 2016/2017

## Bit di validità nella tabella delle pagine



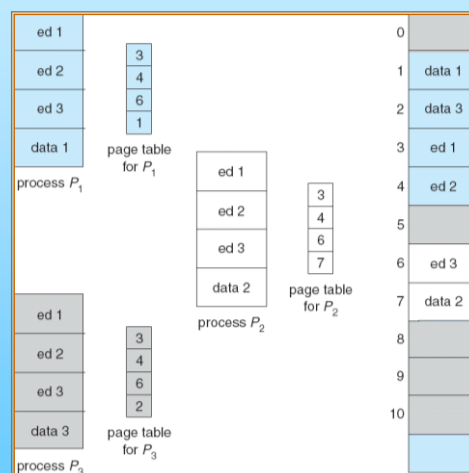
Sistemi Operativi A.A 2016/2017

## Pagine condivise

- La paginazione facilita la condivisione di codice e dati tra i processi
- **Codice condiviso**
  - Una copia di sola lettura del codice viene condivisa tra processi (es. editor di testo, compilatori, shell, ...).
  - Il codice condiviso deve essere nella stessa posizione nello spazio logico di tutti i processi
- **Codice e dati privati**
  - Ogni processo ha una copia separata di codice e dati
  - Le pagine per il codice e dati privati possono essere in qualsiasi posizione nello spazio logico del processo

Sistemi Operativi A.A 2016/2017

## Esempio di pagine condivise



Sistemi Operativi A.A 2016/2017

## Struttura della tabella delle pagine

- Es. **Intel IA32** usa:
  - 20 bit per identificare pagina/frame ( $2^{20} = 1\text{M}$  pagine)
  - 12 bit per offset → pagina da 4KB
  - ogni elemento della tabella delle pagine usa 4 byte (20 bit per indicare frame + 1 bit validità)
  - Quindi ogni processo userebbe 4MB per la sua tabella delle pagine! sono troppi!
- Paginazione gerarchica
- Tabella delle pagine di tipo Hash
- Tabella delle pagine invertita

Sistemi Operativi A.A 2016/2017

## Paginazione Gerarchica

- Divide l'indirizzo logico in tabelle delle pagine multiple
- Una tecnica semplice è la tabella della pagine a due livelli

Sistemi Operativi A.A 2016/2017

## Esempio di paginazione a due livelli

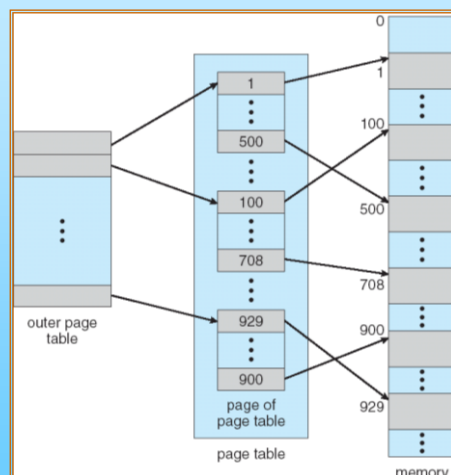
- L'indirizzo logico (32-bit con pagine di 4K) è diviso in:
  - Un numero di pagina di 20 bit
  - Un offset di 12 bit
- La tabella delle pagine è paginata, il numero di pagina è diviso in:
  - 10-bit per il numero di pagina
  - 10-bit per offset nella pagina
- Quindi l'indirizzo logico è come segue:

page number		page offset
$p_1$	$p_2$	$d$
10	10	12

dove  $p_1$  è un indice nella tabella delle pagine esterna (outer page table), e  $p_2$  è lo scostamento nella tabella delle pagine esterna

Sistemi Operativi A.A 2016/2017

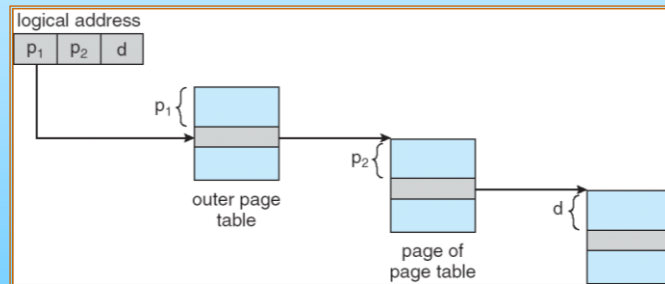
## Schema di tabella delle pagine a due livelli



Sistemi Operativi A.A 2016/2017

## Schema di traduzione degli indirizzi

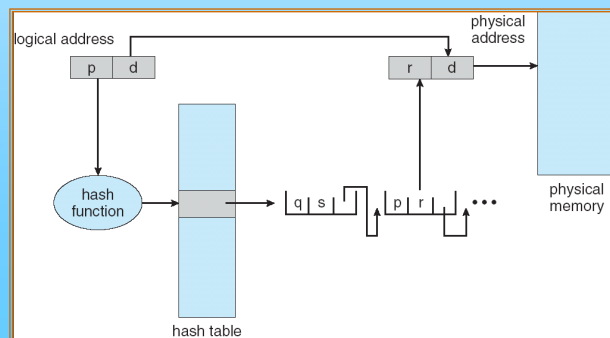
- Schema di traduzione degli indirizzi per paginazione a due livelli



Sistemi Operativi A.A 2016/2017

## Tabella delle pagine di tipo hash

- Comune con indirizzi  $> 32$  bit, dove la tabella delle pagine avrebbe dimensioni proibitive, es: 64 bit e pagine da 4K ( $2^{12}$ ), 52 bit per numero di pagina, tabella di  $4 \cdot 2^{52} = 16384\text{TB}$
- Usa una tabella hash per associare al numero di pagina il frame. Ogni riga della tabella contiene una catena di elementi che hanno lo stesso valore hash.
- Il numero di pagina viene comparato con gli elementi della catena fino a trovare il frame associato alla pagina.

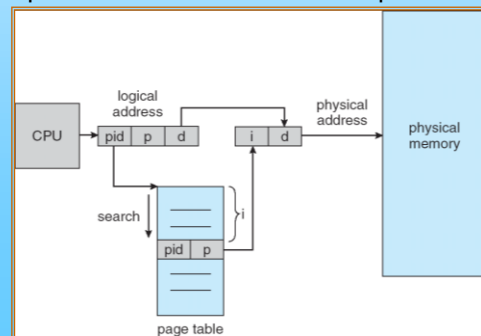


Sistemi Operativi A.A 2016/2017



## Tabella delle pagine invertita

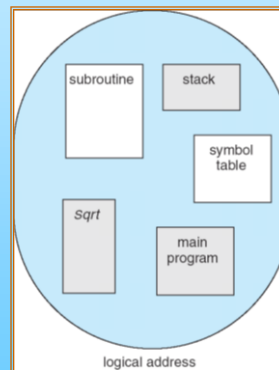
- Una riga per ogni pagina reale in memoria
- Nella riga associata a un frame della memoria viene memorizzato il numero di pagina memorizzata in quel frame e anche informazioni sul processo che possiede quella pagina (pid)
- Diminuisce la memoria necessaria per memorizzare ogni tabella delle pagine ma incrementa il tempo necessario per cercare nella tabella
- Può essere usata una tabella hash per limitare la ricerca a una o poche righe della tabella
- Difficile condividere delle pagine



Sistemi Operativi A.A 2016/2017

## Segmentazione

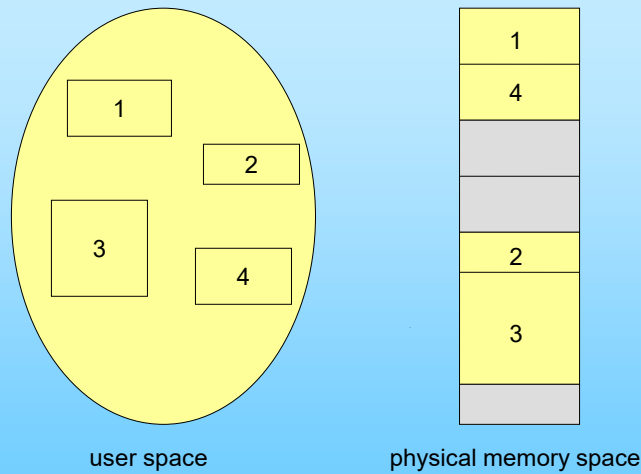
- Schema di gestione della memoria simile a come il programmatore vede la memoria
- Un programma è una collezione di segmenti. Un segmento è una unità logica come:
  - programma principale,
  - procedura,
  - funzione,
  - metodo,
  - oggetto,
  - variabili locali, variabili globali,
  - stack,
  - tabella dei simboli,
  - vettori



- I segmenti non sono tra loro ordinati e possono essere posti in memoria in qualsiasi ordine in modo da utilizzare al meglio la memoria disponibile

Sistemi Operativi A.A 2016/2017

## Visione logica della segmentazione



Sistemi Operativi A.A. 2016/2017

## Architettura della segmentazione

- Indirizzi logici formati da due valori:  
    <numero di segmento, offset>,
- **Tabella dei segmenti** – associa a ogni segmento:
  - *base* – contiene l'indirizzo fisico iniziale dove il segmento è in memoria
  - *limite* – indica la lunghezza del segmento
- Il registro di base della tabella dei segmenti (*Segment-table base register* – *STBR*) punta alla tabella dei segmenti del processo in esecuzione

Sistemi Operativi A.A. 2016/2017

## Architettura della segmentazione (Cont.)

### ■ Rilocalizzazione

- Dinamica
- Usa la tabella dei segmenti

### ■ Condivisione

- Segmenti di memoria possono essere facilmente condivisi
- Ogni processo deve usare lo stesso numero di segmento

### ■ Allocazione

- Ogni segmento può essere allocato usando strategia first fit/best fit
- Si può avere *frammentazione esterna* che può essere risolta rilocando e compattando (operazione lenta)

Sistemi Operativi A.A. 2016/2017

## Architettura della segmentazione (Cont.)

- Protezione. In ogni riga della tabella può essere presente:

- Bit validazione = 0  $\Rightarrow$  segmento illegale
- Privilegi lettura/scrittura/esecuzione

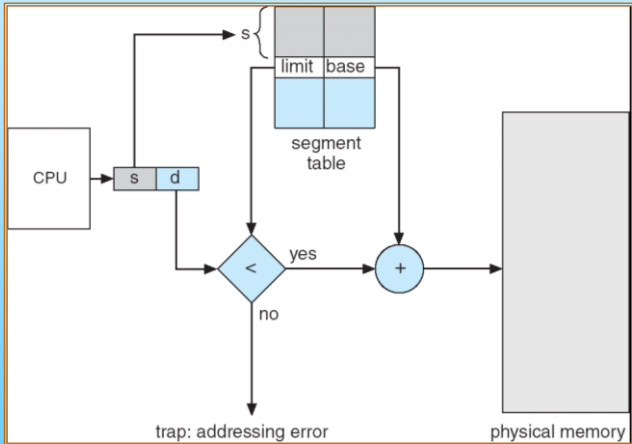
- Bit di protezione associati ai segmenti; condivisione di codice avviene al livello di segmento

- Siccome i segmenti variano di lunghezza l'allocazione della memoria è un problema di allocazione dinamica

- Un esempio di segmentazione nel prossimo diagramma

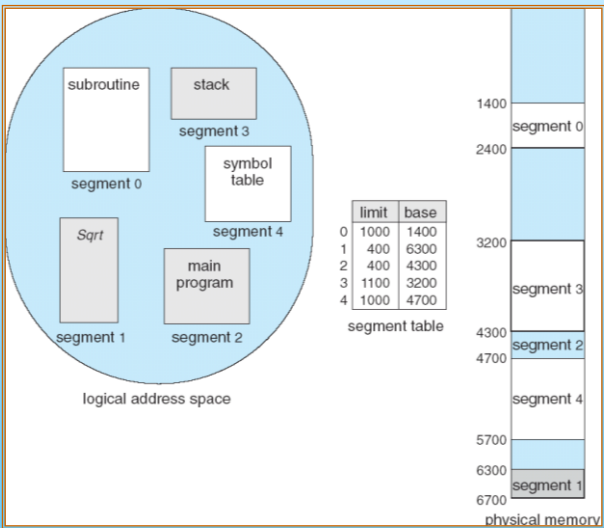
Sistemi Operativi A.A. 2016/2017

# Architettura di traduzione degli indirizzi



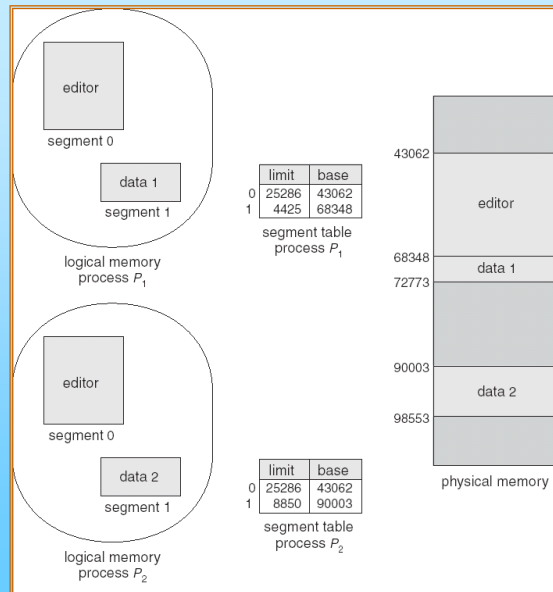
Sistemi Operativi A.A 2016/2017

# Esempio di segmentazione



Sistemi Operativi A.A 2016/2017

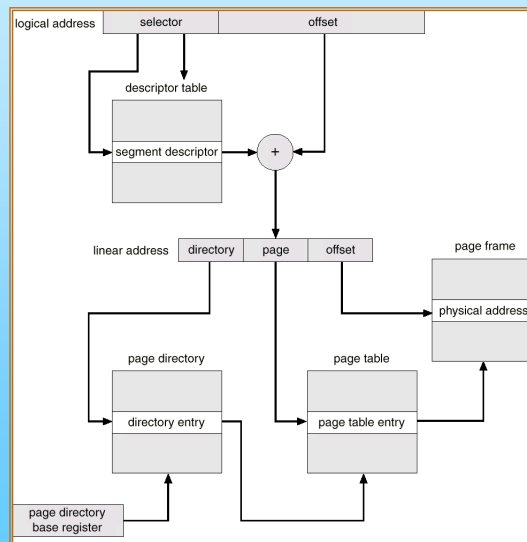
## Condivisione di segmenti



Sistemi Operativi A.A 2016/2017

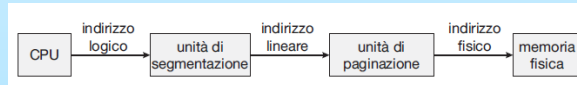
## Segmentazione paginata – Intel Pentium

- Intel Pentium usa la segmentazione con la paginazione a due livelli



Sistemi Operativi A.A 2016/2017

## Architettura Intel – IA 32



### ■ Segmentazione

- 16K segmenti
  - ▶ 8K locali del processo (Local Descriptor Table – LDT)
  - ▶ 8K condivisi tra tutti i processi (Global Descriptor Table – GDT)
- ciascun elemento della LDT e GDT usa 8 byte (ind. base, limite etc.)
- Indirizzo logico:
  - ▶ **selettore segmento** (16 bit)
    - s: 13 bit (id segmento)
    - g: 1 bit (indica se LDT o GDT)
    - p: 2 bit (indica livello privilegio accesso segmento)
  - ▶ **offset nel segmento** (32 bit)

Sistemi Operativi A.A. 2016/2017

## Architettura Intel – IA 32

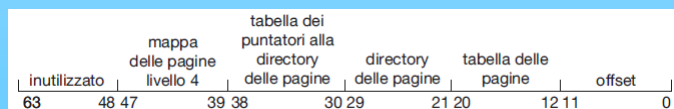
### ■ Paginazione

- Paginazione a due livelli
  - ▶ 10 bit per **directory delle pagine**
  - ▶ 10 bit per **offset nella directory delle pagine**
  - ▶ 12 bit **offset nella pagina**
  - ▶ registro CR3 indica indirizzo della directory delle pagine
  - ▶ Si possono avere pagine da 4KB o da 4MB indicato da flag PageSize nel elemento directory delle pagine (con 4MB offset è di 22 bit)
- 4GB di memoria massima non abbastanza, introdotta PAE Page Address Extension
  - ▶ paginazione a 3 livelli (2 bit, 9 bit, 9bit, 12 bit)
  - ▶ cambiata tabella delle pagine con id frame a 24 bit invece di 20 bit →  $24+12 = 36$  bit di indirizzo max 64GB (ma ogni processo max 4GB)

Sistemi Operativi A.A. 2016/2017

## Architettura x86 - 64

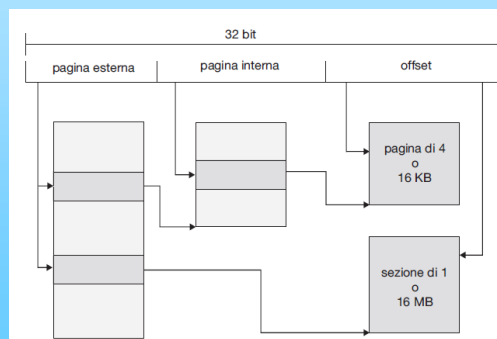
- Sviluppata da AMD (adottata poi anche da Intel) per processori a 64 bit
  - 64 bit indirizzo 16 exabyte sono troppi...
  - sono usati 48 bit (256 TB) con paginazione a 4 livelli
    - ▶ 9 bit **mappa delle pagine livello 4**
    - ▶ 9 bit **tabella dei puntatori alla dir. delle pagine**
    - ▶ 9 bit **directory delle pagine**
    - ▶ 9 bit **tabella delle pagine**
    - ▶ 12 bit **offset**
  - Si posso avere pagine da 4KB, 2MB, 1GB
  - Supporta PAE che usa 52 bit (4096 TB)



Sistemi Operativi A.A 2016/2017

## Architettura ARM

- CPU ARM usate in sistemi mobili (iPhone, iPad, Android)
- Indirizzo a 32 bit
  - Pagine da 4KB o 16 KB (paginazione a 2 livelli)
  - Pagine da 1MB o 16MB (dette sezioni) (paginazione a 1 livello)



Sistemi Operativi A.A 2016/2017