

Sistemi Operativi

Gestione della Memoria (parte 3)

Docente: Claudio E. Palazzi

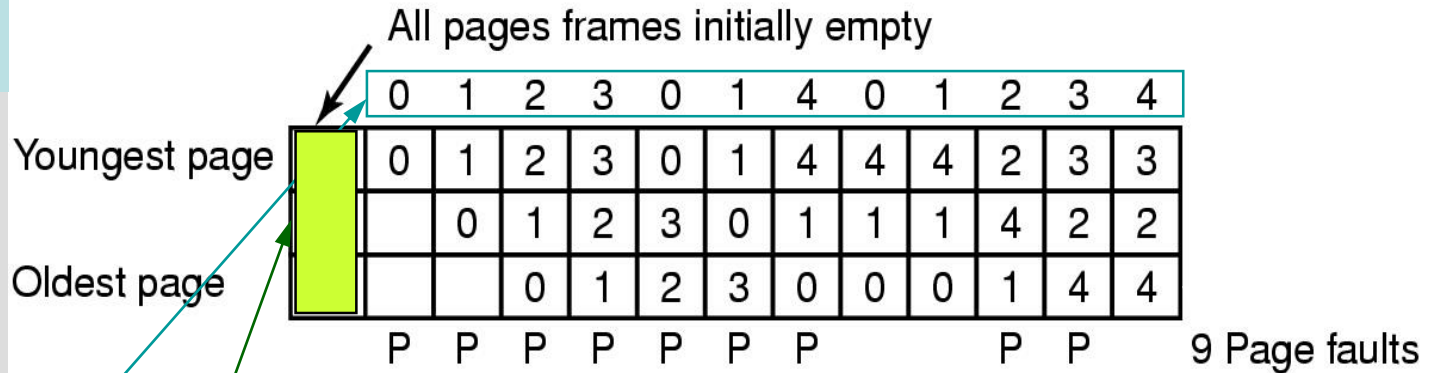
cpalazzi@math.unipd.it

Paginazione: l'anomalia di Belady - 1

- Nel 1969 Lazlo Belady mostrò che la frequenza di *page fault* **non** sempre decresce al crescere dall'ampiezza della RAM
 - Un semplice contro-esempio usando FIFO come strategia di rimpiazzo
 - Sequenza di riferimenti: 0 1 2 3 0 1 4 0 1 2 3 4
 - RAM con 3 *page frame* : 9 *page fault*
 - RAM con 4 *page frame* : 10 *page fault*
- LRU è immune dall'anomalia di Belady
 - Ma la sua forma “pura” è irrealizzabile

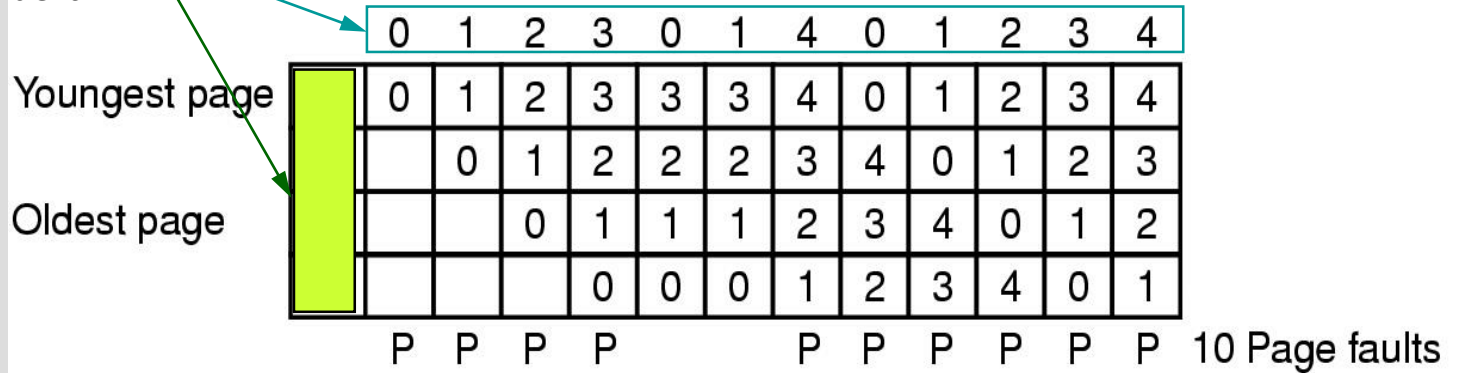
Paginazione: l'anomalia di Belady - 2

FIFO



(a)

- Un sistema di paginazione è caratterizzato da:
1. la sequenza dei riferimenti a pagine generati dai processi
 2. la politica di rimpiazzo
 3. la capienza della RAM



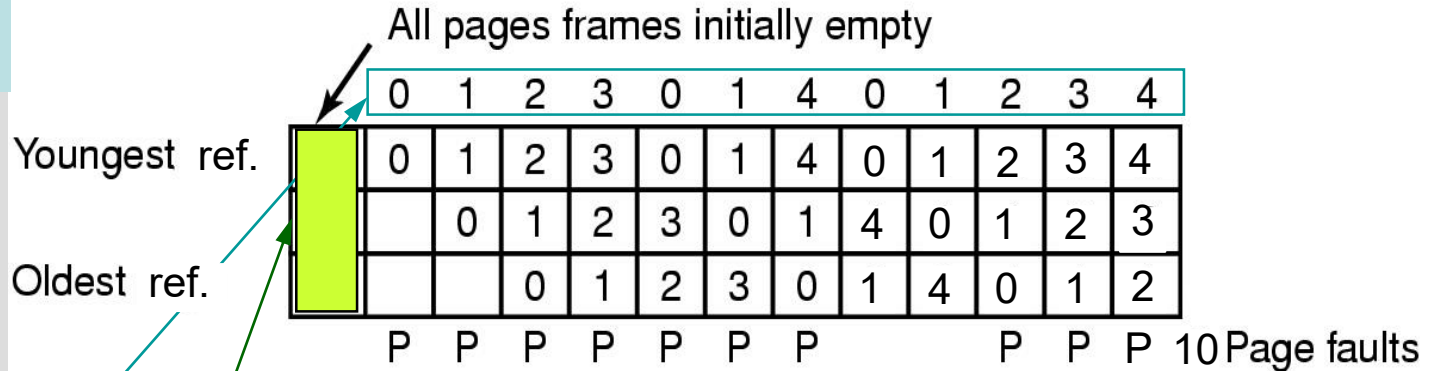
(b)

Paginazione: l'anomalia di Belady - 3

- Una classe di algoritmi particolarmente interessante è quella che soddisfa la proprietà:
 - $M(m, r) \subseteq M(m+1, r)$
 - Dove m rappresenta il numero di page frame, mentre r sono i riferimenti
 - “assumendo gli stessi riferimenti, le pagine caricate con m page frame sono un sottoinsieme di quelle caricate con $m+1$ page frame”
- Detti **stack algorithms**
 - Sono immuni dall'anomalia di Belady
 - LRU, Optimal Replacement

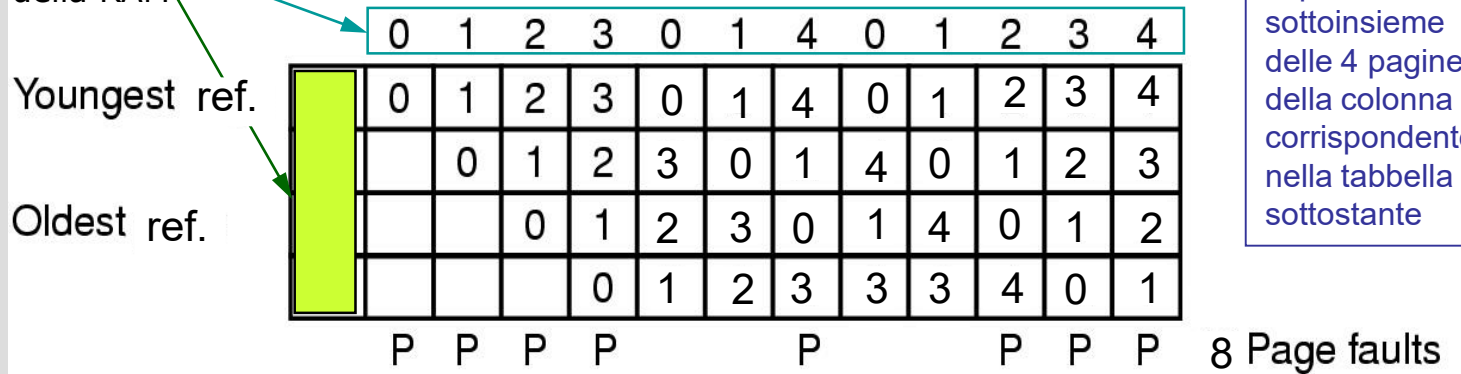
Paginazione: l'anomalia di Belady - 4

LRU



(a)

- Un sistema di paginazione è caratterizzato da:
1. la sequenza dei riferimenti a pagine generati dai processi
 2. la politica di rimpiazzo
 3. la capienza della RAM



(b)

Le 3 pagine in ogni colonna nella tabella sopra sono un sottoinsieme delle 4 pagine della colonna corrispondente nella tabella sottostante

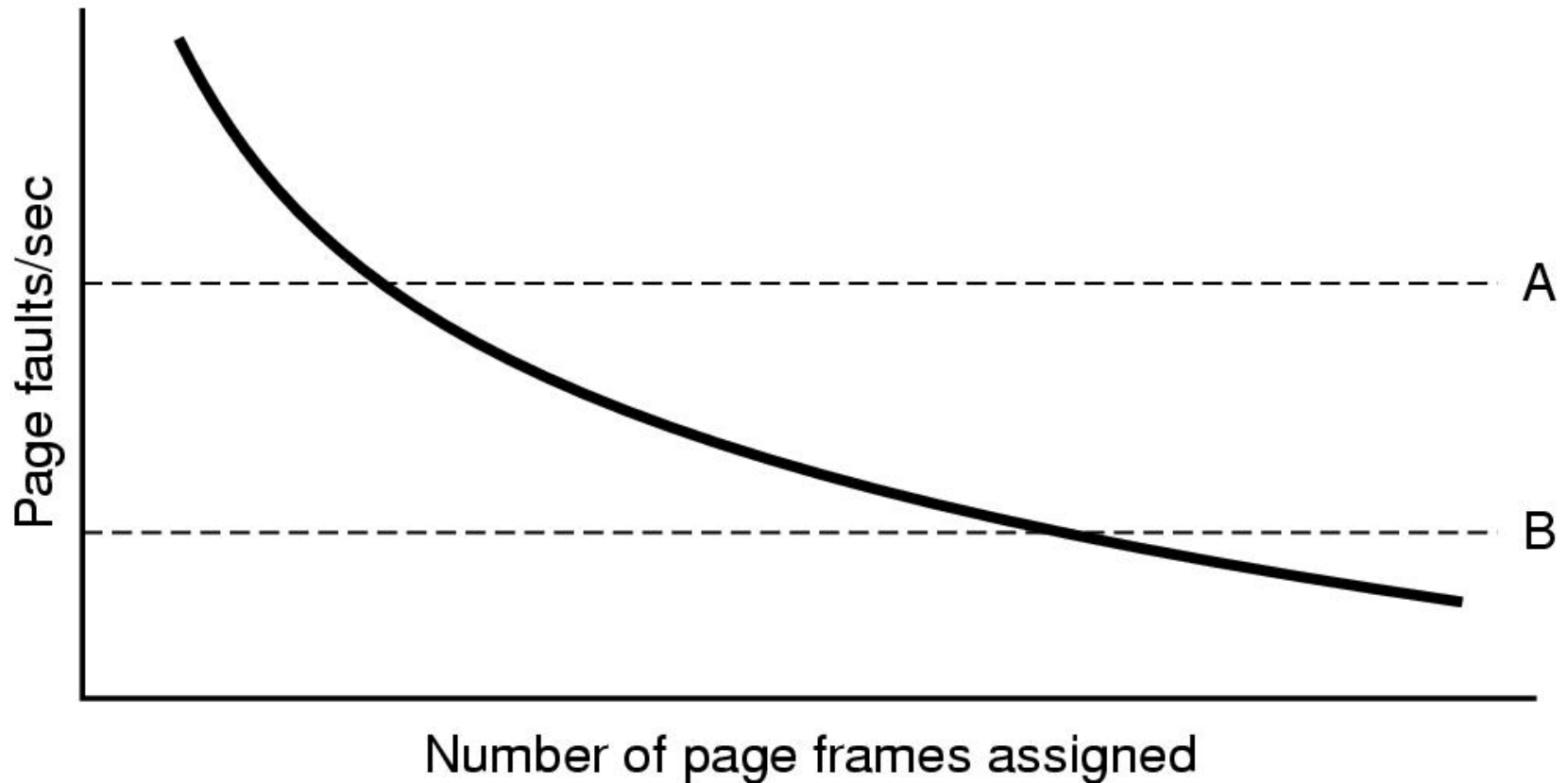
Paginazione: criteri di progetto - 1

- Nel rimpiazzare una pagina occorre scegliere consapevolmente tra
 - Politiche **locali**
 - Rimpiazzo nel WS del processo che ha causato il *page fault*
 - In tal caso ogni processo conserva una quota fissa di RAM
 - Politiche **globali**
 - La scelta avviene tra *page frame* senza distinzione di processo
 - L'allocazione di RAM a disposizione di ogni processo varia dinamicamente nel tempo

Paginazione: criteri di progetto - 2

- Le politiche **globali** sono più **efficaci**
 - Specialmente se l'ampiezza del WS può variare durante l'esecuzione
 - Però bisogna decidere **quanti** *page frame* assegnare a ogni singolo processo
- Le politiche **locali** hanno prestazioni **inferiori**
 - Se il WS di un processo cresce l'allocazione fissa causa rimpiazzamenti indesiderati
 - *Thrashing*
 - Anche con RAM disponibile non usata da altri processi
 - Se il WS si riduce si ha invece spreco di memoria
- Non tutte le politiche si adattano all'uso in entrambe le varianti

Paginazione: criteri di progetto - 3



Paginazione: criteri di progetto - 4

- Controllo del carico
 - Anche con le migliori politiche può accadere che a volte il sistema subisca thrashing
 - Se i WS di tutti i processi eccedono la capacità di memoria
 - **PFF** (Page Fault Frequency) indica che alcuni hanno bisogno di più memoria ma nessuno ha bisogno di meno memoria
 - **SWAP!**
 - Rimuoviamo in successione alcuni processi finché il thrashing si ferma

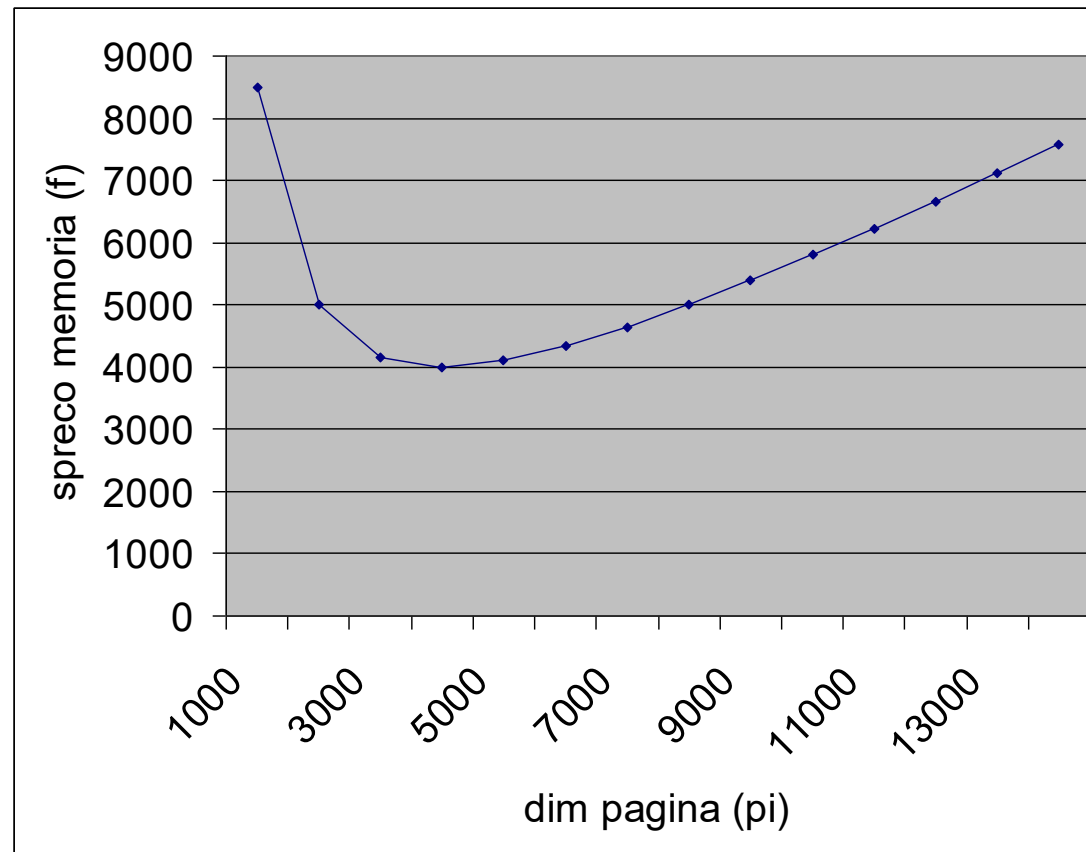
Paginazione: criteri di progetto – 5

- Quale dimensione di pagina?
 - Pagine **ampie**
 - Maggiore rischio di **frammentazione interna**
 - In media ogni processo lascia inutilizzata metà del suo ultimo *page frame*
 - Pagine **piccole**
 - Maggiore ampiezza della tabella delle pagine
- Il valore ottimo può essere definito matematicamente
 - σ B dimensione media di un processo
 - π B dimensione media di una pagina
 - ε B per riga in tabella delle pagine
 - Spreco per processo come $f(\pi) = (\sigma / \pi) \times \varepsilon + \pi / 2$
 - Parte di tabella delle pagine + frammentazione interna
 - Derivata prima è $-\sigma \varepsilon / \pi^2 + 1/2$
 - Ponendo uguale a zero si ha che il minimo di $f(\pi)$ si ha per $\pi = \sqrt{2 \sigma \varepsilon}$

Paginazione: criteri di progetto – 6

Per $\sigma = 1$ MB e $\varepsilon = 8$ B...

$$f(\pi) = (\sigma / \pi) \times \varepsilon + \pi / 2$$



Paginazione: criteri di progetto - 7

- Per $\sigma = 1$ MB e $\varepsilon = 8$ B si ha $\pi = 4$ KB
- Per RAM di ampiezza crescente può convenire un valore di π maggiore
 - Ma di certo non linearmente
- In generale la memoria virtuale **non** è distinta per dati e istruzioni
 - Nella prima metà del '70 vi sono stati elaboratori importanti (PDP-11) che fornivano invece spazi di indirizzamento distinti
 - *Programmed Data Processor* (2 KB *cache*, 2 MB RAM)

Paginazione: realizzazione – 1

- Il S/O compie azioni chiave
 - A ogni creazione di processo
 - Per determinare l'ampiezza della sua allocazione
 - Per creare la tabella delle pagine corrispondente
 - A ogni cambio di contesto
 - Per caricare la MMU e “pulire” la TLB
 - A ogni *page fault*
 - Per analizzare il problema e operare il rimpiazzo
 - A ogni terminazione di processo
 - Per rilasciarne i *page frame*
 - Per rimuoverne la tabella delle pagine

Paginazione: realizzazione – 2

- Per trattare un *page fault* bisogna capire quale riferimento è fallito
 - Per poter completare correttamente l'istruzione interrotta
- Il *Program Counter* dice a quale indirizzo il problema si è verificato
 - Ma non sa distinguere tra istruzione e operando
- Capirlo è compito del S/O
 - Orrendamente complicato dai molti effetti laterali causati dagli “acceleratori” *hardware*
 - Il S/O deve annullare lo stato erroneo e ripetere daccapo l'istruzione fallita

Paginazione: realizzazione – 3

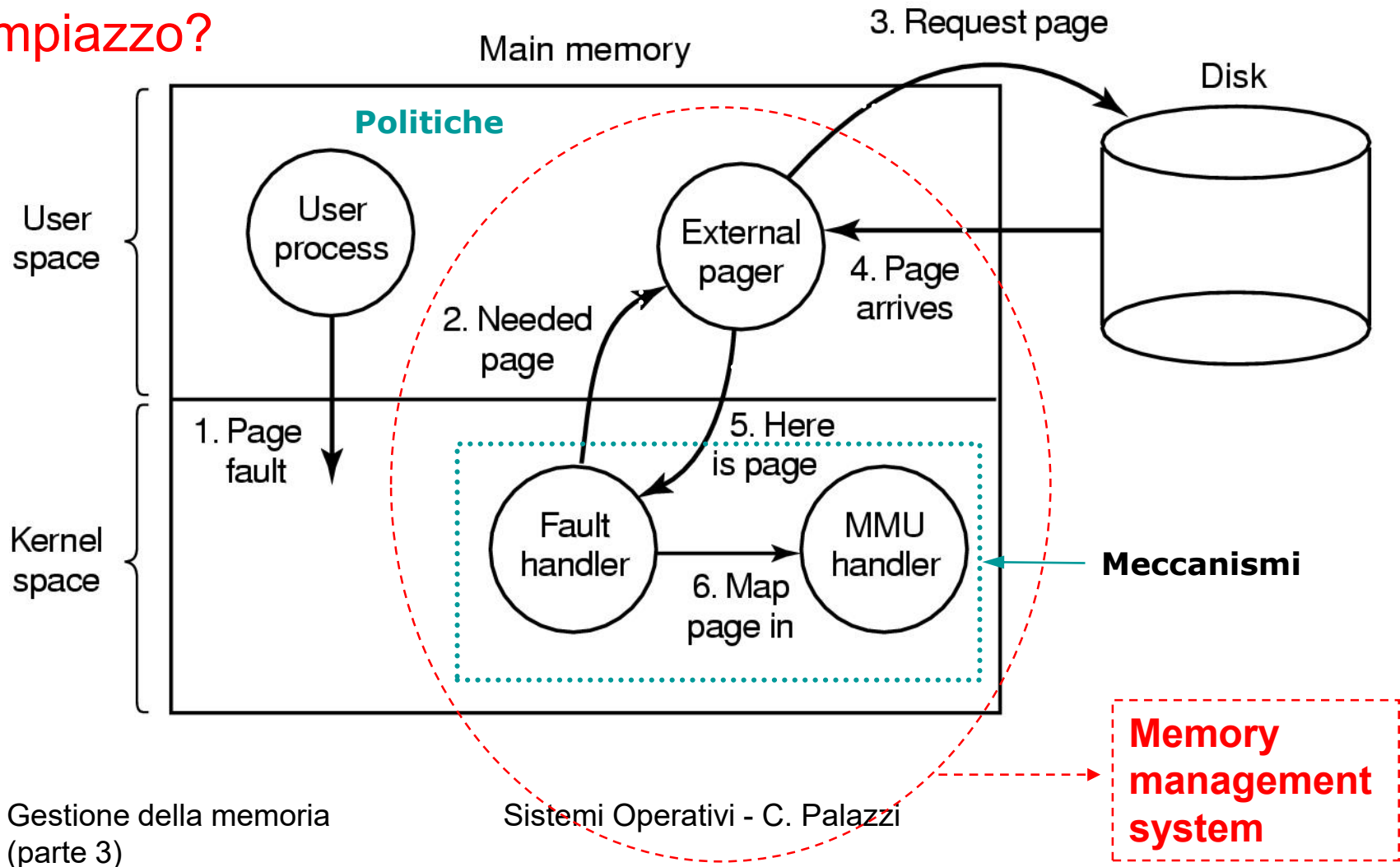
- Page fault: l'hw fa trap al kernel e salva il PC sullo stack
- Un programma assembler salva i dati nei registri e poi chiama il sistema operativo
- Il S.O. scopre il page fault e cerca di capire di quale pagina si tratti (visionando i registri o recuperando il PC e simulando l'istruzione)
- Ottenuto l'indirizzo virtuale causa del page fault, il S.O. verifica che si tratti di indirizzo valido (altrimenti kill del processo) e cerca un page frame vuoto o con pagina rimpiazzabile
- Se la pagina da rimpiazzare è *dirty* ($M=1$), si imposta il suo spostamento su disco (il processo corrente viene sospeso nel frattempo) e il frame viene bloccato

Paginazione: realizzazione – 4

- Quando il page frame è libero, vi copia la pagina richiesta (il processo viene di nuovo sospeso nel frattempo)
- All'arrivo dell'interrupt del disco, la page table è aggiornata e il frame è indicato come normale
- Il PC viene reimpostato per puntare all'istruzione causa del page fault
- Il processo causa del page fault è pronto per esecuzione e il S.O. ritorna al programma assembler che lo aveva chiamato
- Il programma assembler ricarica i registri e altre info; poi torna in user space per continuare l'esecuzione

Paginazione: realizzazione – 5

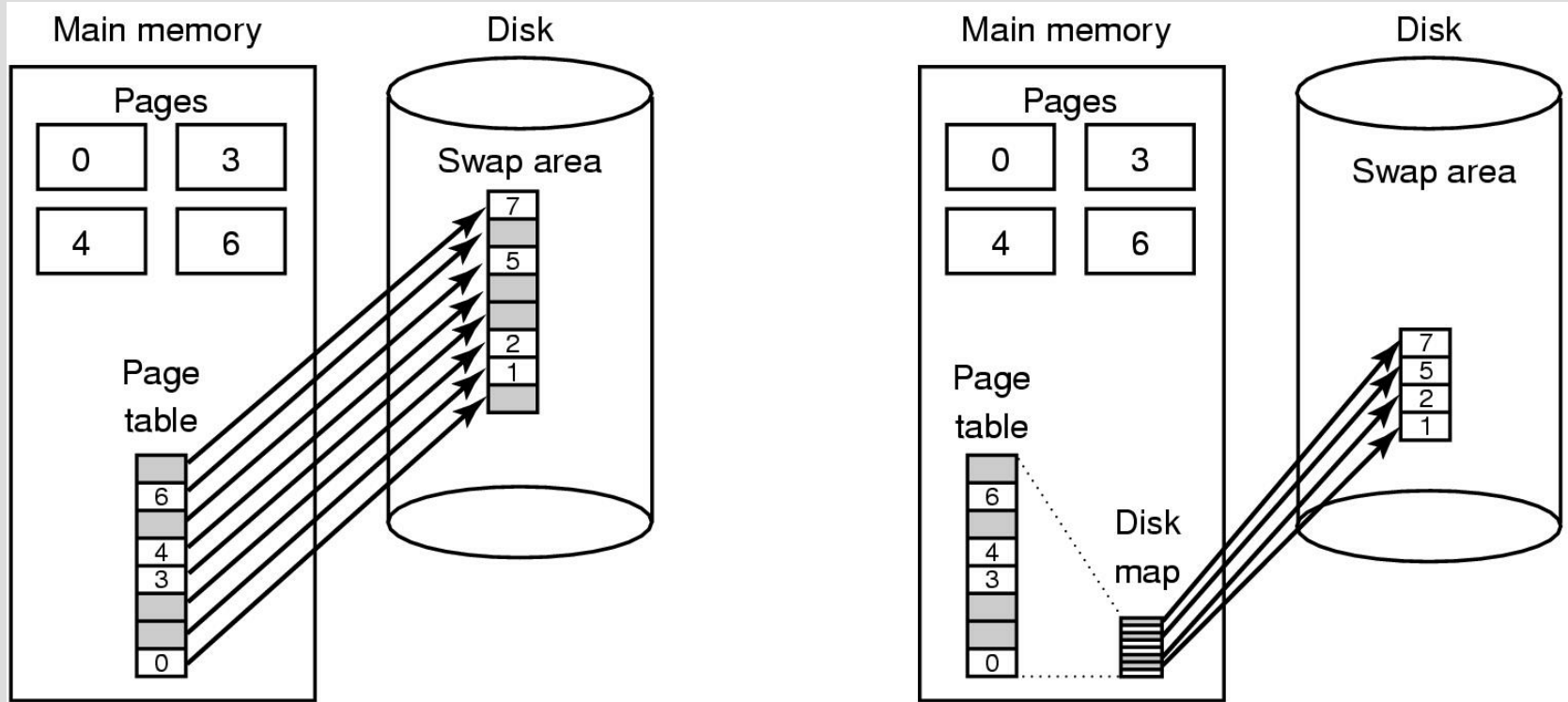
E l'algoritmo di rimpiazzo?



Paginazione: realizzazione – 6

- Un'area del disco può essere riservata per ospitare le pagine temporaneamente rimpiazzate
 - Area di *swap*
- Ogni processo ne riceve in dote una frazione
 - Che rilascia alla sua terminazione
 - I puntatori (base, ampiezza) a questa zona devono essere mantenuti nella tabella delle pagine del processo
 - Ogni indirizzo virtuale mappa nell'area di *swap* direttamente rispetto alla sua base
- Idealmente
 - L'intera immagine del processo potrebbe andare subito nell'area di *swap* alla creazione del processo
 - Altrimenti potrebbe andare tutta in RAM e spostarsi nell'area di *swap* quando necessario
- Però sappiamo che i processi **non** hanno dimensione costante
 - Allora è meglio che l'area di *swap* sia frazionata per codice e dati
- Se l'area di *swap* **non** fosse riservata allora occorrerebbe ricordare in RAM l'indirizzo su disco di **ogni** pagina rimpiazzata
 - Informazione associata alla tabella delle pagine

Paginazione: realizzazione – 7



Area di *swap* pre-assegnata e mappata automaticamente dalla tabella delle pagine

Area di *swap* assegnata a richiesta e mappata esplicitamente dalla tabella delle pagine

Paginazione: realizzazione – 8

- LINUX
 - Partizione dedicata allo SWAP, con file system apposito
 - Consumo una delle possibili partizioni del disco
 - Dimensione impostabile dall'utente in fase di installazione
 - Almeno la stessa dimensione della RAM se si vuole gestire l'ibernazione (copia di tutto il contenuto della RAM nell'area di SWAP e ricaricamento in fase di riattivazione)
- Windows (2000, XP, ...)
 - Uso di file di swap
 - hiberfil.sys (usato per copiare la RAM in caso di ibernazione del sistema)
 - pagefile.sys (usato quando la memoria RAM non è sufficiente)
 - Se il file viene frammentato, le prestazioni calano

Paginazione: realizzazione – 9

- Per separare le politiche dai meccanismi
 - Conviene svolgere nel nucleo del S/O **solo** le azioni più delicate
 - Gestione della MMU
 - Specifica dell'architettura *hardware*
 - Trattamento **immediato** del *page fault*
 - Largamente indipendente dall'*hardware*
 - Demandando il resto della gestione a un processo esterno al nucleo
 - Scelta delle pagine e loro trasferimento
 - Trattamento **differito** del *page fault*

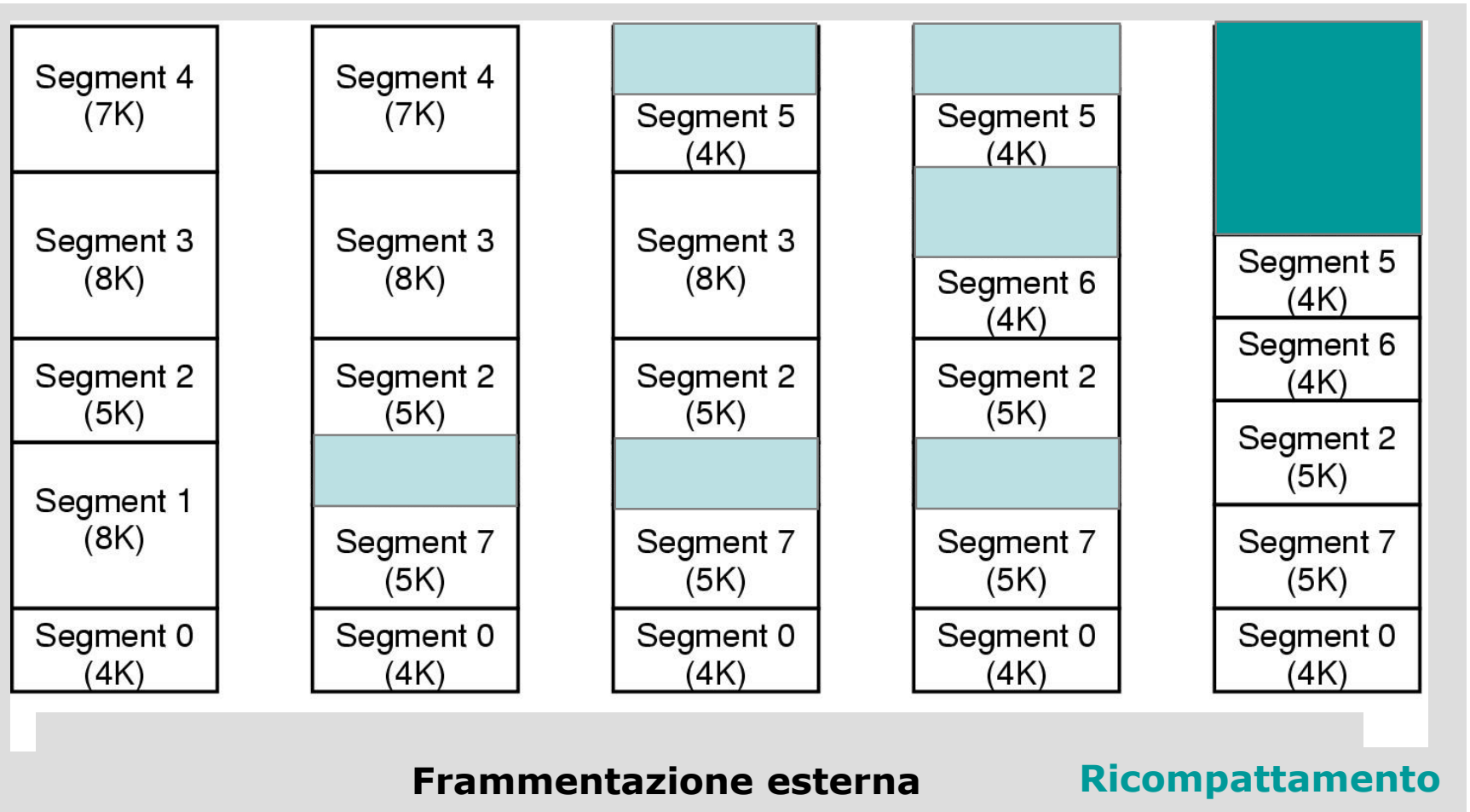
Segmentazione: premesse – 1

- Spazi di indirizzamento completamente **indipendenti** gli uni dagli altri
 - Per dimensione e posizione in RAM
 - Entrambe possono variare dinamicamente
- Entità **logica** nota al programmatore e destinata a contenere informazioni **coese**
 - Codice di una procedura
 - Dati di inizializzazione di un processo
 - *Stack* di processo
- Si presta a schemi di **protezione** specifica
 - Perché il **tipo** del suo contenuto può essere stabilito a priori
 - Ciò che **non** si può fare con la paginazione
- Causa frammentazione **esterna**

Segmentazione: premesse – 2

| | Paginazione | Segmentazione |
|--|---|---|
| Il programmatore ne deve essere consapevole | No | Sì |
| Consente N spazi di indirizzamento lineari | $N = 1$ | $N \geq 1$ |
| La sua ampiezza può eccedere la capacità della RAM | Sì | Sì |
| Consente di separare e distinguere tra codice e dati | No | Sì |
| Consente di gestire contenuti a dimensione variabile nel tempo | No | Sì |
| Consente di condividere parti di programmi tra processi | No | Sì |
| A quale obiettivo risponde | Consentire spazi di indirizzamento più grandi della RAM | Consentire la separazione logica tra aree dei processi e la loro protezione specifica |

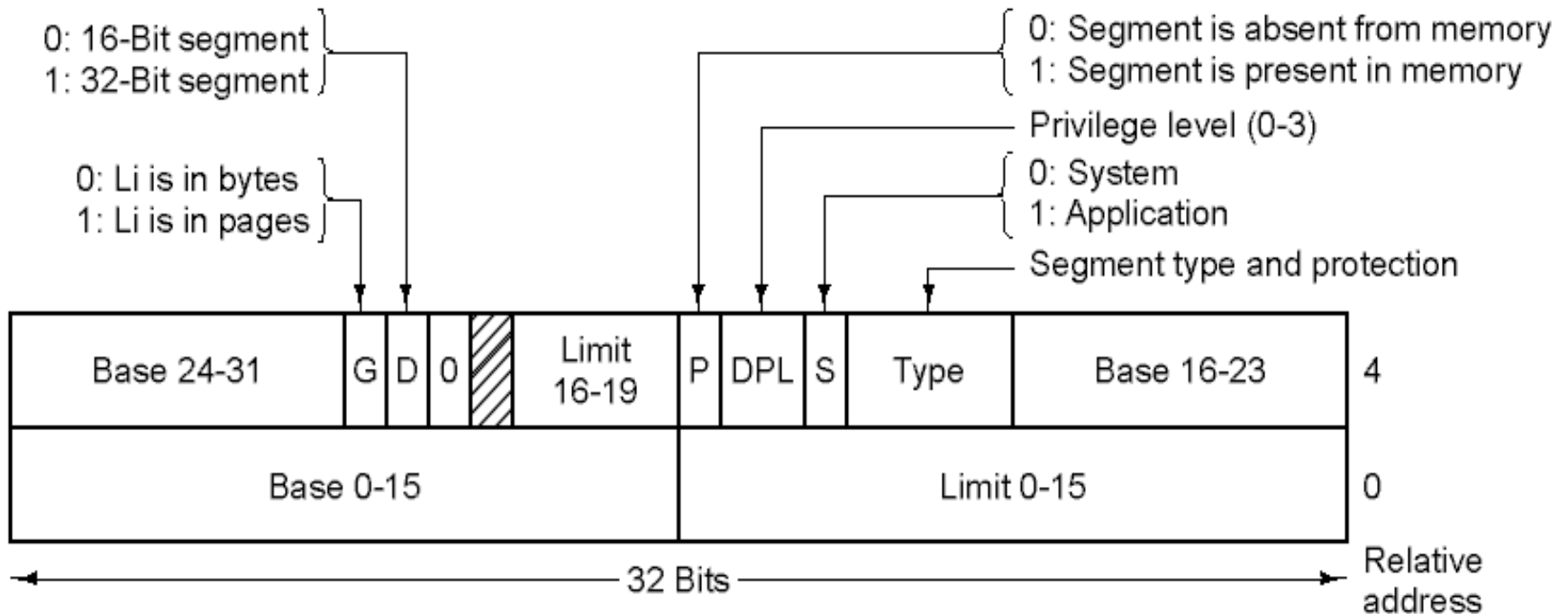
Segmentazione: realizzazione – 1



Segmentazione: realizzazione – 2

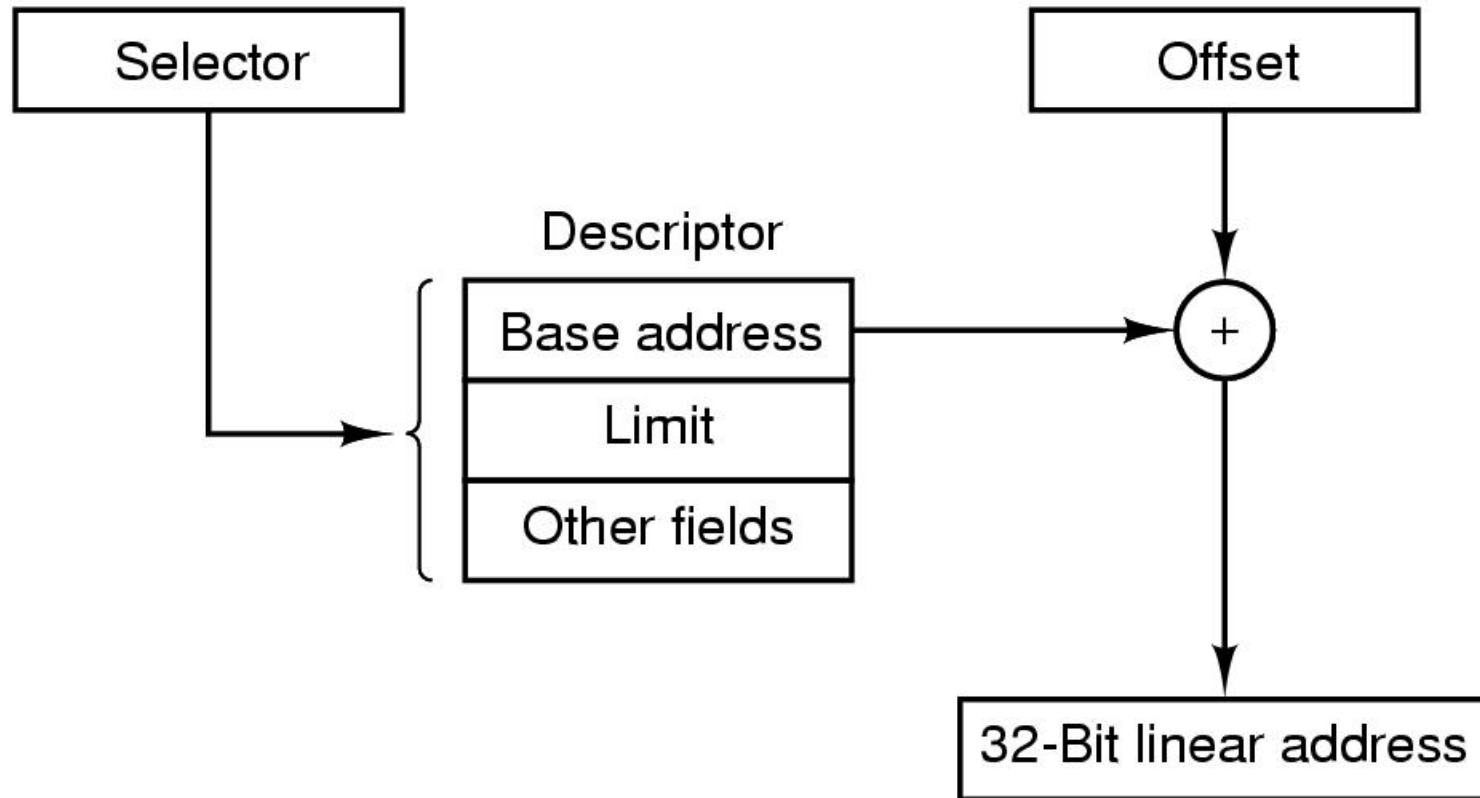
- Vista la grande ampiezza potenziale i segmenti sono spesso **paginati**
- Nel caso del Pentium di Intel
 - Fino a 16 K segmenti indipendenti
 - Di ampiezza massima 4 GB (32 *bit*)
 - Una LDT per processo
 - *Local Descriptor Table*
 - Descrive i segmenti del processo
 - Una singola GDT per l'intero sistema
 - *Global Descriptor Table*
 - Descrive i segmenti del S/O

Segmentazione: realizzazione – 4



Descrittore di segmento di Pentium relativo al codice (lievi differenze con quello relativo ai dati)

Segmentazione: realizzazione – 5

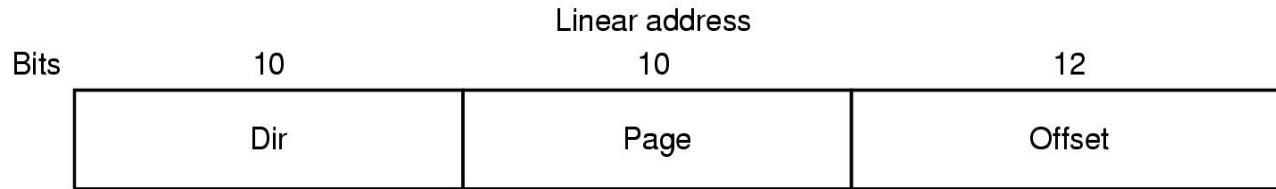


Conversion di una coppia (selettore, offset) in un indirizzo lineare

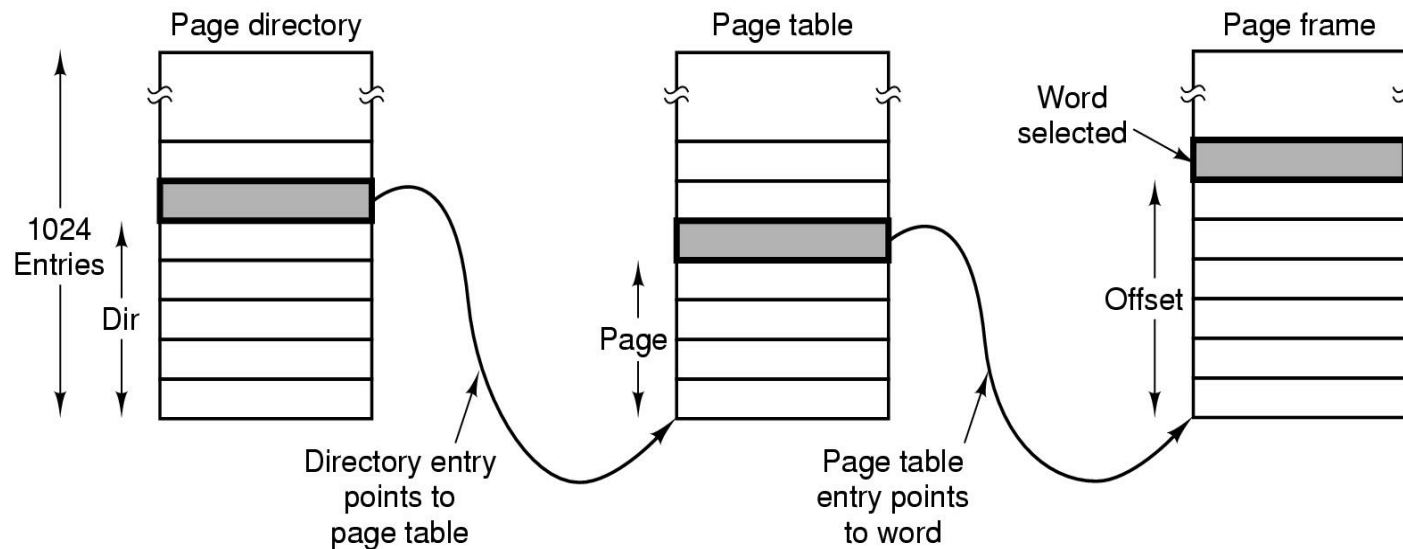
Segmentazione: realizzazione – 6

- L'indirizzo **lineare** ottenuto da (base di segmento + *offset*) può essere interpretato come
 - Indirizzo **fisico** se il segmento considerato non è paginato
 - Indirizzo **logico** altrimenti
 - Nel qual caso il segmento viene visto come una memoria virtuale paginata e l'indirizzo come virtuale in essa
 - 10 *bit* : indice in catalogo di tabelle delle pagine
 - » 2^{10} righe da 32 *bit* ciascuna (base di tabella denotata)
 - 10 *bit* : indice in tabella delle pagine selezionata
 - » 2^{10} righe da 32 bit ciascuna (base di *page frame*)
 - 12 *bit* : posizione nella pagina selezionata
 - » *Offset* in pagina da 4 KB

Segmentazione: realizzazione – 7



(a)



(b)

- L'indirizzo **lineare** mappato sullo spazio virtuale