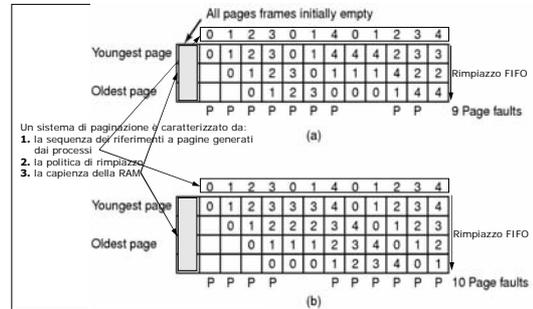


Paginazione: l'anomalia di Belady - 1

- Nel 1969 Lazo 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



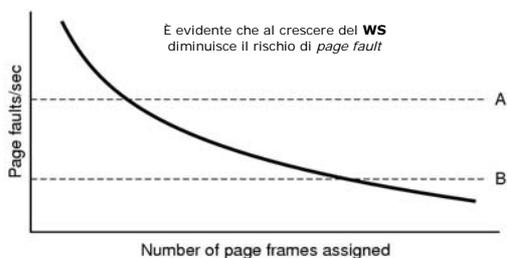
Paginazione: criteri di progetto - 1

- 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 rimpiazzi indesiderati
 - *Thrashing* (trebbiatura ©)
 - 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

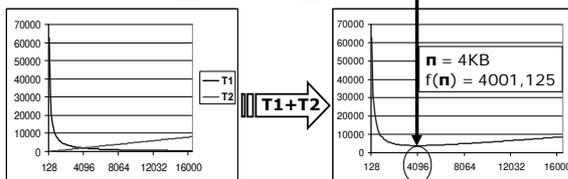


Paginazione: criteri di progetto - 4

- 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 in media in uso per processo
 - n B per pagina
 - ϵ B per riga in tabella delle pagine
 - "Spreco" in funzione di n come $f(n) = (\sigma / n) \times \epsilon + n / 2$
 - Il minimo di $f(n)$ si ha per $n = \sqrt{2 \sigma \times \epsilon}$

Paginazione: criteri di progetto - 5

- Per $\sigma = 1$ MB e $\epsilon = 8$ B si ha $n = 4$ KB = 4096 B
- $f(n) = \underbrace{(\sigma / n) \times \epsilon}_{T1} + \underbrace{n / 2}_{T2}$



Gestione della memoria (parte 3) Sistemi Operativi - Vardanega/Palazzi Pagina 169/183

Paginazione: criteri di progetto - 6

- Come visto, per $\sigma = 1$ MB e $\epsilon = 8$ B si ha $n = 4$ KB
- Per RAM di ampiezza crescente può convenire un valore di n 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)

Gestione della memoria (parte 3) Sistemi Operativi - Vardanega/Palazzi Pagina 170/183

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

Gestione della memoria (parte 3) Sistemi Operativi - Vardanega/Palazzi Pagina 171/183

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

Gestione della memoria (parte 3) Sistemi Operativi - Vardanega/Palazzi Pagina 172/183

Paginazione: realizzazione - 3

- 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

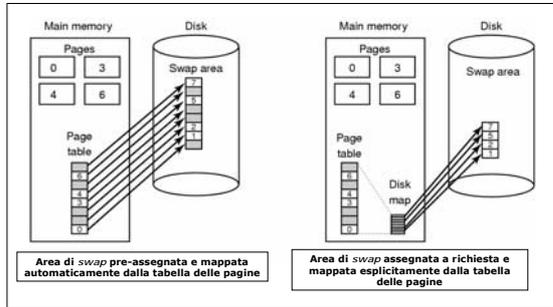
Gestione della memoria (parte 3) Sistemi Operativi - Vardanega/Palazzi Pagina 173/183

Paginazione: realizzazione - 4

- 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

Gestione della memoria (parte 3) Sistemi Operativi - Vardanega/Palazzi Pagina 174/183

Paginazione: realizzazione – 5



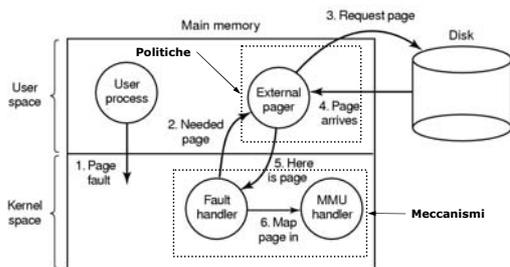
Gestione della memoria (parte 3) Sistemi Operativi - Vardanega/Palazzi Pagina 175/183

Paginazione: realizzazione – 6

- 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*

Gestione della memoria (parte 3) Sistemi Operativi - Vardanega/Palazzi Pagina 176/183

Paginazione: realizzazione – 7



Gestione della memoria (parte 3) Sistemi Operativi - Vardanega/Palazzi Pagina 177/183

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**

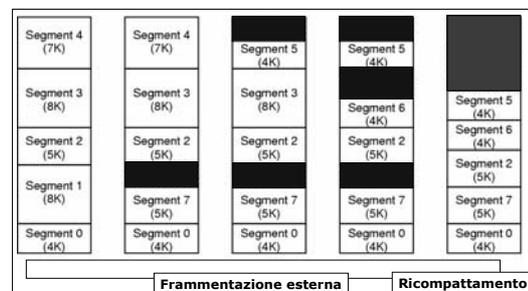
Gestione della memoria (parte 3) Sistemi Operativi - Vardanega/Palazzi Pagina 178/183

Segmentazione: premesse – 2

	Paginazione	Segmentazione
Il programmatore ne deve essere consapevole	No	Si
Consente N spazi di indirizzamento lineari	N = 1	N ≥ 1
La sua ampiezza può eccedere la capacità della RAM	Si	Si
Consente di separare e distinguere tra codice e dati	No	Si
Consente di gestire contenuti a dimensione variabile nel tempo	No	Si
Consente di condividere parti di programmi tra processi	No	Si
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

Gestione della memoria (parte 3) Sistemi Operativi - Vardanega/Palazzi Pagina 179/183

Segmentazione: realizzazione – 1



Gestione della memoria (parte 3) Sistemi Operativi - Vardanega/Palazzi Pagina 180/183

Segmentazione: realizzazione – 2

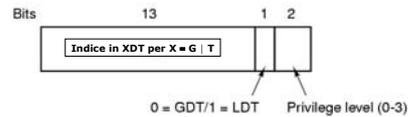
- Vista la grande ampiezza potenziale i segmenti sono spesso **paginati**
- Nel caso del Pentium di Intel
 - Fino a 16 segmenti indipendenti
 - Di ampiezza massima 4 GB (32 *bit*)
 - Una LDT per processo
 - *Local Descriptor Table*
 - Descrive i segmenti del processo
 - Una singola GDT per l'intero sistema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

Gestione della memoria (parte 3)

Sistemi Operativi - Vardanega/Palazzi

Pagina 181/183

Segmentazione: realizzazione – 3



- 6 registri di segmento
 - Di cui 1 denota il segmento corrente
- LDT e GDT contengono $2^{13} = 8\text{ K}$ descrittori di segmento
 - I descrittori di segmento sono espressi su 8 B
 - La **base** del segmento in RAM è espressa su 32 *bit*
 - Il **limite** su 20 *bit* per verificare la legalità dell'*offset* fornito dal processo
 - Consente ampiezza massima a 1 MB (per **granularità** a B)
 - Oppure 1 M pagine da 4 KB ovvero 4 GB (per granularità a pagine)

Gestione della memoria (parte 3)

Sistemi Operativi - Vardanega/Palazzi

Pagina 182/183

Segmentazione: realizzazione – 4

- 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*)
 - 22 *bit* : posizione nella pagina selezionata
 - *Offset* in pagina da 4 KB

Gestione della memoria (parte 3)

Sistemi Operativi - Vardanega/Palazzi

Pagina 183/183