

Memoria Virtuale

- Introduzione
- Paginazione
- Gestione dei page fault
- Rimpiazzamento delle pagine
- Allocazione delle pagine fisiche
- Thrashing

Basato parzialmente su slide originali di Silberschatz, Galvin and Gagne

Memoria Virtuale

1

Introduzione alla Memoria Virtuale

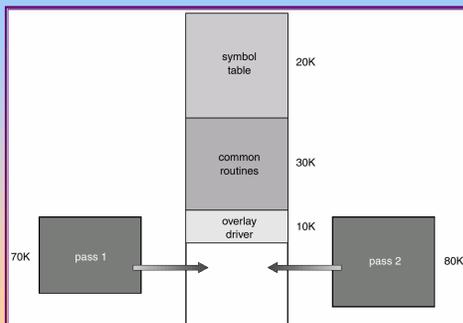
Come gestire situazioni in cui il programma da eseguire è più grande della memoria fisica?

- Overlay
- Memoria virtuale

Memoria Virtuale

2

Overlay per Assemblatore a 2 passate



Memoria Virtuale

3

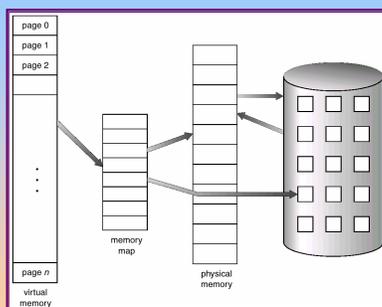
Overlay

- Il programma viene suddiviso in parti (overlay)
- Gli overlay vengono mantenuti su disco e caricati in memoria solo quando devono essere eseguiti
- Il caricamento e scaricamento degli overlay è svolto dal SO
- La suddivisione del programma in overlay è compito del programmatore
 - Difficoltà di scomporre un programma complesso
 - Una scomposizione che va bene per un sistema può non andare bene per un altro sistema con minori capacità di memoria

Memoria virtuale

- Permette l'esecuzione di programmi i cui l'insieme di codice+dati+stack eccede la memoria disponibile
- Si tengono in memoria solo alcune porzioni del programma
- Le altre parti vengono tenute su disco (spazio di swap) e caricate quando necessario
- La gestione è completamente trasparente all'utente
- In sistemi multi-programmati la Memoria Virtuale consente di aumentare il grado di multi-programmazione

Memoria Virtuale (cont.)



Memoria Virtuale (cont.)

- Indirizzo logico (o virtuale)
 - Indirizzo generato dal programma
- Spazio di indirizzamento logico
 - insieme degli indirizzi logici
- Indirizzo fisico
 - Indirizzo visto dalla memoria
- Spazio di indirizzamento fisico
 - insieme degli indirizzi fisici

Spazio di indirizzamento logico \geq Spazio di indirizzamento fisico

Memoria Virtuale 7

Memoria Virtuale (cont.)

- Indirizzo logico == indirizzo fisico
 - In sistemi senza memoria virtuale
 - In sistemi senza rilocazione dinamica
- In sistemi **senza** memoria virtuale l'indirizzo logico viene inviato **direttamente** alla memoria
- In sistemi **con** memoria virtuale l'indirizzo logico viene inviato alla **MMU** per la traduzione
- **MMU (Memory Management Unit)**
 - Esegue la rilocazione dinamica
 - Necessaria perché il programma può essere caricato in punti diversi della memoria durante l'esecuzione

Memoria Virtuale 8

Memoria Virtuale (cont.)

Tecniche per realizzare la memoria virtuale

- **Paginazione**
- **Segmentazione**
- **Segmentazione paginata**

Memoria Virtuale 9

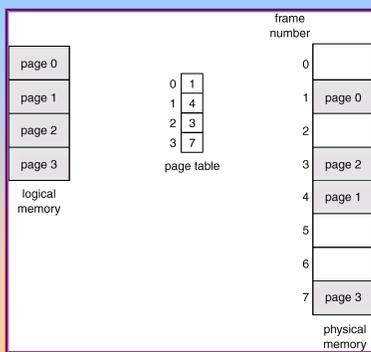
Paginazione

- Memoria fisica suddivisa in **pagine fisiche o frame**
 - Blocchi di memoria di dimensione fissa
 - Dimensione potenza di 2
- Memoria logica suddivisa in **pagine logiche**
 - Stessa dimensione delle pagine fisiche
- Il gestore della memoria tiene traccia dei frame liberi
- Allocazione
 - Per eseguire un programma che richiede N pagine logiche bisogna trovare N frame e caricarci il programma
 - I frame non devono essere necessariamente contigui
- Tabella delle pagine
 - contiene la mappatura fra pagine logiche e fisiche
- Frammentazione interna
 - La dimensione del programma difficilmente è un multiplo esatto della dimensione della pagina

Memoria Virtuale

10

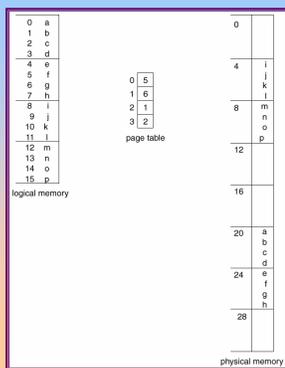
Esempio



Memoria Virtuale

11

Esempio (2)



Memoria Virtuale

12

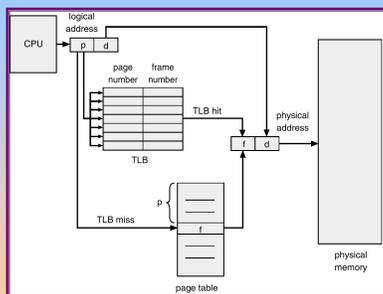
Tabella delle pagine

- Viene tenuta in memoria
- Registro *Page-table base register* (PTBR)
 - Punta all'indirizzo iniziale della TP
- Registro *Page-table length register* (PRLR)
 - Contiene la dimensione della TP
- Per accedere in memoria (per istruzioni o dati)
 - Si accede prima alla TP
 - Quindi si accede a istruzione/dato
- *Translation look-aside buffers* (TLBs)
 - Piccola cache associativa
 - Permette un accesso molto veloce
 - Se si trova l'associazione (pagina logica)/(pagina fisica) si evita di accedere alla TP

Memoria Virtuale

16

Paging Hardware With TLB



Memoria Virtuale

17

Effective Access Time

- Associative Lookup = ϵ time unit
- Assume memory cycle time is 1 microsecond
- Hit ratio
 - percentage of times that a page number is found in the associative registers;
 - ration related to number of associative registers.
- Hit ratio = α
- Effective Access Time (EAT)

$$EAT = (1 + \epsilon) \alpha + (2 + \epsilon)(1 - \alpha)$$

$$= 2 + \epsilon - \alpha$$

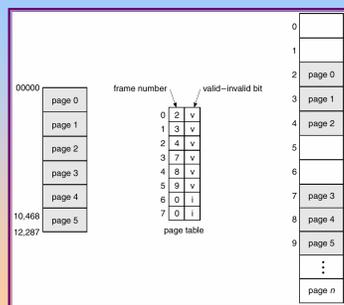
Memoria Virtuale

18

Memory Protection

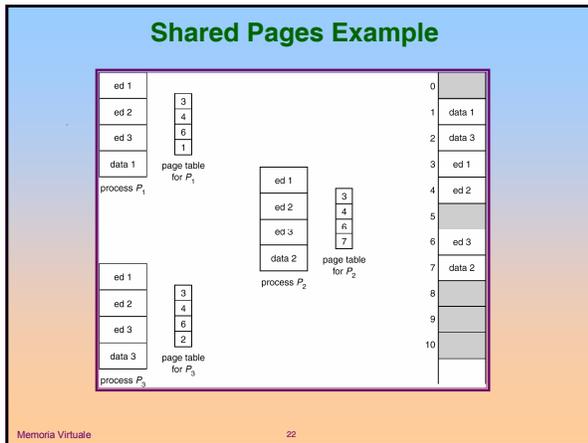
- Memory protection implemented by associating protection bit with each frame.
- *Valid-invalid* bit
 - Bit attached to each entry in the page table:
 - “valid” indicates that the associated page is in the process’ logical address space, and is thus a legal page.
 - “invalid” indicates that the page is not in the process’ logical address space.

Valid (v) or Invalid (i) Bit In A Page Table

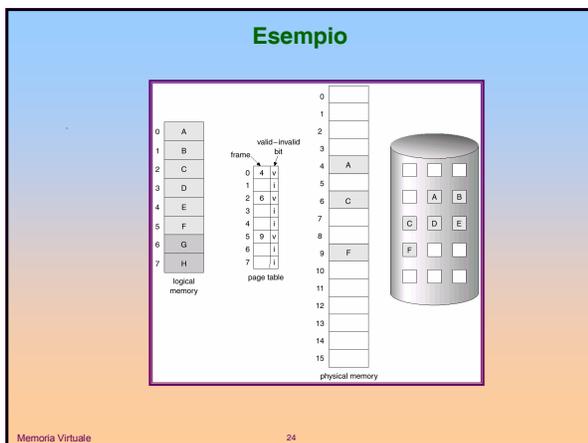


Shared Pages

- Shared code
 - One copy of read-only (reentrant) code shared among processes (i.e., text editors, compilers, window systems).
 - Shared code must appear in same location in the logical address space of all processes.
- Private code and data
 - Each process keeps a separate copy of the code and data.
 - The pages for the private code and data can appear anywhere in the logical address space.



- ### Paginazione su domanda
- Si una pagina in memoria solo quando necessario
 - Meno operazioni di I/O
 - Meno memoria utilizzata
 - Risposte più veloci
 - Maggior numero di processi in memoria
 - La pagina è necessaria quando viene riferita
 - Si guarda il bit di validità
 - Bit di validità = 0:
 - ☞ invalid reference ⇒ abort
 - ☞ not-in-memory ⇒ bring to memory
- Memoria Virtuale 23



Bit di Validità

- Inizialmente i bit di validità sono tutti a zero
- Vengono messi a 1 man mano che le pagine sono caricate in memoria

Frame #	Bit di validità
	1
	1
	1
	1
	0
...	...
	0
	0

- Durante un accesso se il bit di validità si trova uguale a 0 (e il riferimento è valido) si genera un **page fault**

Memoria Virtuale

25

Page Fault

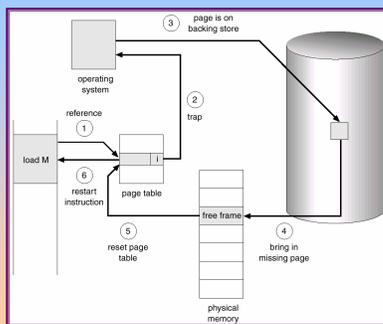
- Il primo riferimento alla pagina genera page fault
- Il SO guarda un'altra tabella e decide:
 - Riferimento non valido ⇒ abort
 - Pagina non in memoria ⇒ caricamento della pagina in memoria
- Nel secondo caso localizza un frame libero
- Carica la pagina dallo spazio di swap nel frame
- Modifica opportunamente la Tabella delle pagine
- Riattiva l'esecuzione dell'istruzione che ha generato page fault

Si veda figura in slide successiva

Memoria Virtuale

26

Gestione di un Page Fault

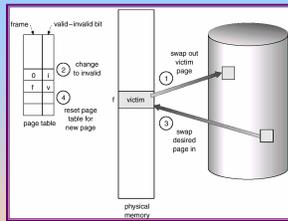


Memoria Virtuale

27

Se non esiste un frame libero?

Rimpiazzamento di pagina



- Si seleziona una pagina vittima
- Si scarica la pagina selezionata nello spazio di swap
- Si carica la pagina desiderata dallo spazio di swap nel frame lasciato libero dalla pagina vittima

Memoria Virtuale

28

Rimpiazzamento

- Il rimpiazzamento delle pagine completa la separazione fra memoria logica e memoria fisica
- Permette di mappare uno spazio logico su uno spazio fisico più piccolo
- Comporta un notevole overhead
 - La percentuale di page fault dovrebbe essere tenuta bassa
- **Modify (dirty) bit**
 - Ulteriore bit nella tabella delle pagine
 - Inizialmente impostato a 0
 - Viene messo a 1 solo se la pagina viene modificata
 - In fase di swap out la pagina viene copiata solo se dirty=1
 - Nella selezione della pagina vittima conviene privilegiare pagine con dirty=0.

Memoria Virtuale

29

Degradazione delle prestazioni

- Page Fault Rate $0 \leq p \leq 1.0$
 - if $p = 0$ no page faults
 - if $p = 1$, every reference is a fault
- Effective Access Time (EAT)
 - EAT = $(1 - p) \times$ memory access
 - + p (page fault overhead
 - + [swap page out]
 - + swap page in
 - + restart overhead)

Memoria Virtuale

30

Degradazione delle prestazioni: Esempio

- Memory access time = 1 microsecond
- Ipotesi
 - Si assume che nel 50% dei casi la pagina selezionata come vittima non è stata modificata (dirty=0)
 - Non è necessario dare swap out
- Swap Page Time = 10 msec = 10,000 µsec

$$\begin{aligned} \text{EAT} &= (1 - p) \times 1 + p (10,000 + 5000) \\ &= 1 + 14999 \times p \quad (\text{in } \mu\text{sec}) \end{aligned}$$

p dovrebbe essere basso per avere tempi di accesso effettivi bassi

Algoritmi di rimpiazzamento

- Dovrebbero contribuire a tenere bassa la % di page fault
- La % di page fault di un processo dovrebbe diminuire all'aumentare del numero di frame allocati al processo
- Stringa di accessi di riferimento per valutare le prestazioni di vari algoritmi

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

Algoritmi di rimpiazzamento (cont.)

- FIFO
- Ottimale
- LRU
- LRU approssimato
- Seconda chance
- Basati su conteggio

First-In-First-Out (FIFO) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time per process)

1	1	4	5
2	2	1	3
3	3	2	4

9 page faults

- 4 frames

1	1	5	4
2	2	1	5
3	3	2	
4	4	3	

10 page faults

- FIFO Replacement – Belady's Anomaly
 - more frames ⇒ less page faults

Memoria Virtuale 34

Optimal Algorithm

- Replace page that will not be used for longest period of time.
- 4 frames example

1, 2, 3, 4, 1, 1, 2, 5, 1, 2, 3, 4, 5

1	4
2	
3	
4	5

6 page faults

- How do you know this?
- Used for measuring how well your algorithm performs.

Memoria Virtuale 35

Least Recently Used (LRU) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	5
2	
3	5 4
4	3

- Counter implementation
 - Every page entry has a counter; every time page is referenced through this entry, copy the clock into the counter.
 - When a page needs to be changed, look at the counters to determine which are to change.

Memoria Virtuale 36

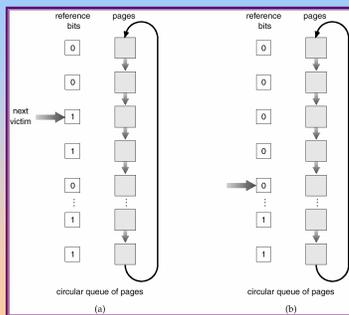
Second chance algorithm

- Need reference bit.
- Clock replacement.
- If page to be replaced (in clock order) has reference bit = 1 then:
 - set reference bit 0.
 - leave page in memory.
 - replace next page (in clock order), subject to same rules.
- Degenerates to FIFO if all pages have reference bit = 0

Memoria Virtuale

40

Second-Chance (clock) Page-Replacement Algorithm



Memoria Virtuale

41

Counting Algorithms

- Keep a counter of the number of references that have been made to each page.
- LFU Algorithm: replaces page with smallest count.
- MFU Algorithm: based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

Memoria Virtuale

42

Allocation of Frames

- Each process needs **minimum** number of pages.
- Two major allocation schemes.
 - fixed allocation
 - priority allocation

Memoria Virtuale 43

Fixed Allocation

- Equal allocation – e.g., if 100 frames and 5 processes, give each 20 pages.
- Proportional allocation – Allocate according to the size of process.

s_i = size of process p_i $S = \sum s_i$ m = total number of frames a_i = allocation for $p_i = \frac{s_i}{S} \times m$	$m = 64$ $s_1 = 10$ $s_2 = 127$ $a_1 = \frac{10}{137} \times 64 \approx 5$ $a_2 = \frac{127}{137} \times 64 \approx 59$
--	---

Memoria Virtuale 44

Priority Allocation

- Use a proportional allocation scheme using priorities rather than size.
- If process P_i generates a page fault,
 - select for replacement one of its frames.
 - select for replacement a frame from a process with lower priority number.

Memoria Virtuale 45

Global vs. Local Allocation

- **Global** replacement – process selects a replacement frame from the set of all frames; one process can take a frame from another.
- **Local** replacement – each process selects from only its own set of allocated frames.

Memoria Virtuale

46

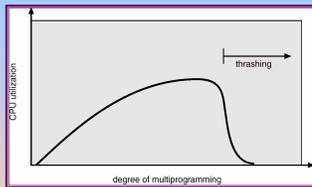
Thrashing

- If a process does not have “enough” pages, the page-fault rate is very high. This leads to:
 - low CPU utilization.
 - operating system thinks that it needs to increase the degree of multiprogramming.
 - another process added to the system.
- **Thrashing** ≡ a process is busy swapping pages in and out.

Memoria Virtuale

47

Thrashing



- Why does paging work?
Locality model
 - Process migrates from one locality to another.
 - Localities may overlap.
- Why does thrashing occur?
 Σ size of locality > total memory size

Memoria Virtuale

48

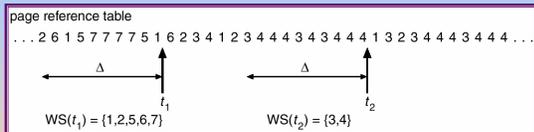
Working-Set Model

- Δ \equiv working-set window \equiv a fixed number of page references
Example: 10,000 instruction
- WSS_i (working set of Process P_i) = total number of pages referenced in the most recent Δ (varies in time)
 - if Δ too small will not encompass entire locality.
 - if Δ too large will encompass several localities.
 - if $\Delta = \infty \Rightarrow$ will encompass entire program.
- $D = \sum WSS_i \equiv$ total demand frames
- if $D > m \Rightarrow$ Thrashing
- Policy if $D > m$, then suspend one of the processes.

Memoria Virtuale

49

Working-set model



Memoria Virtuale

50

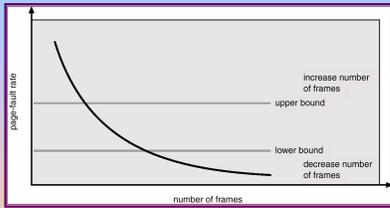
Keeping Track of the Working Set

- Approximate with interval timer + a reference bit
- Example: $\Delta = 10,000$
 - Timer interrupts after every 5000 time units.
 - Keep in memory 2 bits for each page.
 - Whenever a timer interrupts copy and sets the values of all reference bits to 0.
 - If one of the bits in memory = 1 \Rightarrow page in working set.
- Why is this not completely accurate?
- Improvement = 10 bits and interrupt every 1000 time units.

Memoria Virtuale

51

Page-Fault Frequency Scheme



- Establish “acceptable” page-fault rate.
 - If actual rate too low, process loses frame.
 - If actual rate too high, process gains frame.
