Memoria Virtuale

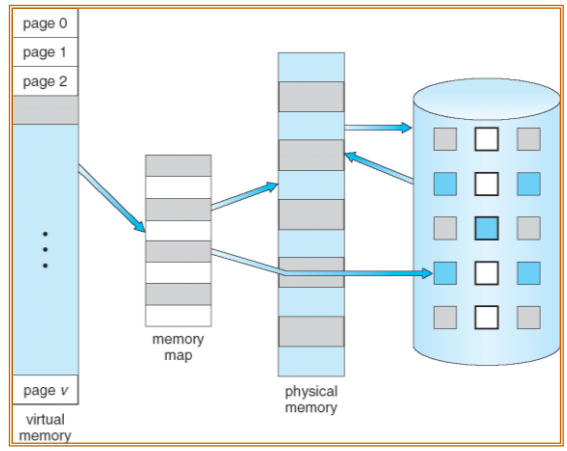
381

Background

- **Memoria virtuale** – separazione della memoria logica dalla memoria fisica.
  - Solo parte del programma ha la necessità di essere in memoria per essere eseguito.
  - Non tutti i dati in memoria sono utilizzati «contemporaneamente»
  - Lo spazio di indirizzi logico può essere molto più grande dello spazio di indirizzi fisico.
  - Serve ad aumentare il grado di multiprogrammazione del sistema
    - Es: 4GB RAM
      - processi max 100MB → max 40 processi
      - processi usano 10MB → max 400 processi
- Memoria virtuale può essere implementa tramite:
  - Paginazione su richiesta

382

Memoria Virtuale è più grande della Memoria Fisica



383

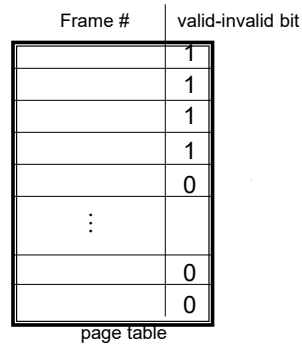
Paginazione su richiesta

- Porta una pagina in memoria solo quando è necessario
  - Necessaria meno memoria
  - Aumenta grado multiprogrammazione
- Pagina necessaria ⇒ si referenzia
  - Riferimento errato ⇒ abort
  - Non in memoria ⇒ porta la pagina in memoria

384

Bit Valido-Invalido

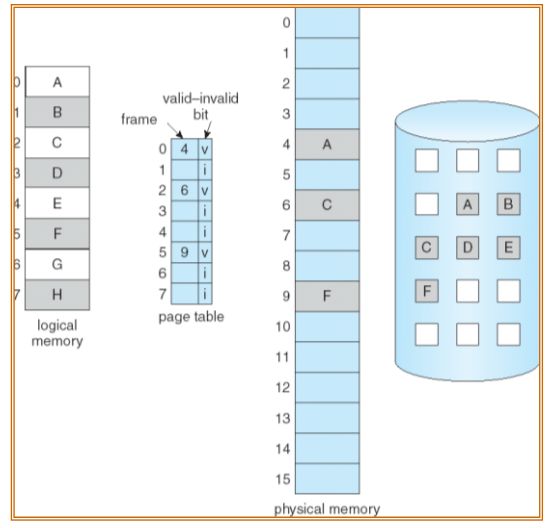
- Ogni riga della tabella delle pagine ha un bit valido-invalido (1 ⇒ in memoria, 0 ⇒ non in memoria)
- Inizialmente il bit è posto a 0 su tutte le righe
- Esempio di tabella delle pagine:



- Durante la traduzione dell'indirizzo se il bit valido/invalido è 0 ⇒ page fault

385

Tabella delle pagine con alcune pagine non in memoria

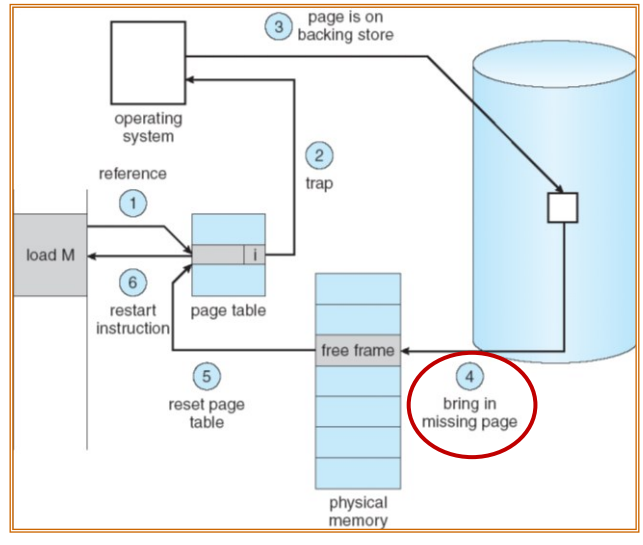


386

Page Fault

- Se viene riferita una pagina il primo riferimento genera una trap nel sistema operativo ⇒ page fault
- SO decide se:
  - Riferimento non valido ⇒ abort.
  - Pagina non in memoria.
- Prende un frame libero.
- Carica la pagina desiderata nel frame.
- Imposta il bit di validità a 1.
- Fa ripartire l'istruzione fallita

Passi nella gestione del page fault



Cosa succede quando non c'è nessun frame libero?

- Sostituzione di una pagina – trova una pagina in memoria non utilizzata e la manda su disco
  - Come sceglierla?
    - FIFO
    - LRU Least Recently Used
    - Varianti di LRU
  - performance – vogliamo un algoritmo che produca il minor numero di page fault
- La stessa pagina può essere portata in memoria molte volte

Performance della Paginazione su richiesta

- Frequenza Page Fault $0 \leq p \leq 1.0$
  - se $p = 0$ nessun page fault
  - se $p = 1$, ogni riferimento provoca un fault
- Tempo effettivo di accesso (EAT)

EAT = (1 – p) x accesso in memoria
+ p (tempo page fault
+ [tempo swap page out]
+ tempo swap page in
+ tempo restart istruzione)

Esempio Paginazione su richiesta

- Tempo accesso in memoria = 200 nano secondi
- Tempo di gestione page fault = 8 milli secondi

- EAT = (1 – p) x 200 + p x (8 millisecondi) =
  (1 – p) x 200 + p x 8.000.000 =
  200 + 7.999.800 x p (in nanosec)
- **Es.** se 1 riferimento su 1000 genera page fault
  - EAT = 8,2 microsecondi, rallentato di 40 volte!
- **Es.** quale page fault rate per avere EAT ridotto di meno del 10%?
  - EAT < 200 x (1 + 10%)
  - 200 + 7.999.800 x p < 220
  - 7.999.800 x p < 20
  - p < 0,00000025
  - 1 accesso ogni 399.990 deve generare page fault

391

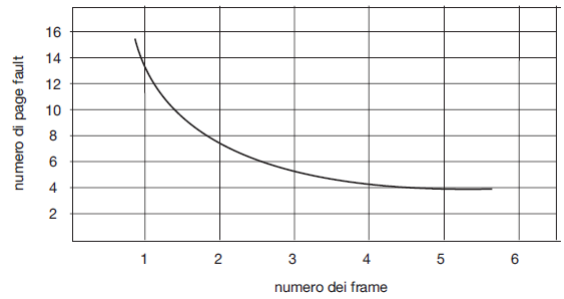
Sostituzione delle pagine

- Può accadere che il sistema stia usando tutta la memoria disponibile, in questo caso il SO trova la lista dei frame liberi vuota e deve scegliere quale frame riusare
- In generale il frame da riusare va prima scritto su disco e poi sostituito con il nuovo frame
- La scrittura può essere omessa se la pagina non era stata modificata, possibile solo se la CPU dispone di un bit di modifica che indica se la pagina era stata modificata

392

Sostituzione delle pagine

- Successione dei riferimenti di un processo
  - sequenza delle pagine utilizzate da un processo non considerando i riferimenti immediatamente successivi alla stessa pagina
- In generale aumentando il numero di frame il numero di page fault diminuisce



393

Sostituzione FIFO

- Si sostituisce la pagina presente da più tempo
- Es. successione riferimenti:
  - 7, 0, 1, 2, 0, 3, 0, 4, 2 ,3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1
- Memoria con 3 frame

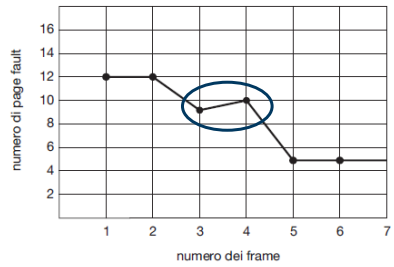
7 → **7** x x	0 → **0** 2 3
0 → 7 **0** x	3 →
1 → 7 0 **1**	2 →
2 → **2** 0 1	1 → 0 **1** 3
0 →	2 → 0 1 **2**
3 → 2 **3** 1	0 →
0 → 2 3 **0**	1 →
4 → **4** 3 0	7 → **7** 1 2
2 → 4 **2** 0	0 → 7 **0** 2
3 → 4 2 **3**	1 → 7 0 **1**

15 page faults

394

Anomalia di Belady

- Con politica FIFO si può avere che aumentando il numero di frame il numero di page faults può aumentare
- Es. 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
  - 3 frame → 9 page faults
    - 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
  - 4 frame → 10 page faults
    - 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



395

Sostituzione Ottimale

- Sostituisce la pagina che non verrà usata per il più lungo periodo di tempo
- Assicura il tasso minimo di page faults
- Difficilmente implementabile si deve conoscere preventivamente la sequenza di accessi, alg. usato solo per fare confronti con altre politiche
- Es. successione riferimenti:
  - 7, 0, 1, 2, 0, 3, 0, 4, 2 ,3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1
- Memoria con 3 frame

7 → **7** x x	4 → 2 **4** 3	2 →
0 → 7 **0** x	2 →	0 →
1 → 7 0 **1**	3 →	1 →
2 → **2** 0 1	0 → 2 **0** 3	7 → **7** 0 1
0 →	3 →	0 →
3 → 2 0 **3**	2 →	1 →
0 →	1 → 2 0 **1**	

9 page faults

396

Sostituzione LRU

- Viene scelta pagina usata meno di recente (Least Recently Used)
- Es. successione riferimenti:
  - 7, 0, 1, 2, 0, 3, 0, 4, 2 ,3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1
- Memoria con 3 frame

7 → **7** x x	4 → **4** 0 3	2 →
0 → 7 **0** x	2 → 4 0 **2**	0 → 1 **0** 2
1 → 7 0 **1**	3 → 4 **3** 2	1 →
2 → **2** 0 1	0 → **0** 3 2	7 → 1 0 **7**
0 →	3 →	0 →
3 → 2 0 **3**	2 →	1 →
0 →	1 → **1** 3 2	

12 page faults

- **Nota:** Sostituzione Ottimale e LRU non soffrono della anomalia di Belady

397

Sostituzione LRU

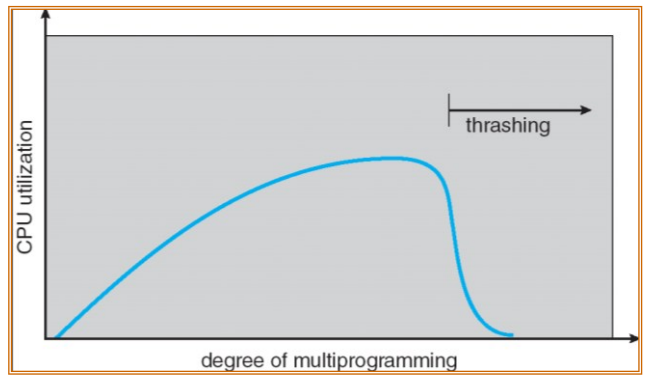
- Implementazione:
  - con un **contatore**:
    - incrementato ad ogni accesso in memoria
    - il valore del contatore associato ad ogni uso di una pagina
    - viene scelta la pagina con il valore più piccolo
  - con uno **stack**
    - ogni volta che una pagina viene usata viene inserita in testa allo stack (spostata se già presente nello stack)
    - in fondo allo stack si trova la pagina usata meno di recente
- LRU deve essere realizzata in hardware ma costosa
- Si possono avere soluzioni approssimate utilizzando bit che indica se pagina riferita

398

Thrashing

- Se un processo non ha abbastanza pagine in memoria, la frequenza dei page-fault è molto alta. Questo porta a:
  - Bassa utilizzazione della CPU
  - Il sistema operativo pensa che può incrementare il grado di multiprogrammazione
  - Un altro processo aggiunto al sistema... che peggiora ulteriormente la situazione...
- **Thrashing** ≡ il sistema è occupato quasi esclusivamente e fare swap delle pagine da/a disco

Thrashing (Cont.)



Trashing

- effetti limitati con **algoritmo di sostituzione locale**,
  - se un processo entra in trashing nel page-fault può riusare solo un frame già posseduto da lui e non rubare frame ad altri processi che possono entrare a loro volta in trashing, comunque un processo in trashing rallenta indirettamente anche l'esecuzione di altri processi aumentando il tempo di gestione del page fault
- Il trashing si evita se ogni processo in esecuzione ha in memoria i frame che gli servono in quel momento **(working-set)** generando pochi page fault

401

Trashing – working set

- **working set** definito come insieme delle pagine riferite dal processo in un certo numero di riferimenti in memoria
- La dimensione del working set cambia durante l'esecuzione del processo, il processo si sposta di "località"
- Il SO tiene traccia della dimensione di ogni WS
- non si ha trashing se
  - la somma delle dimensioni dei working set è inferiore al numero di frame disponibili,
  - se ciò non accade il SO seleziona un processo e lo sospende, lo salva su disco e libera tutti i suoi frame, il processo rientrerà in esecuzione quando ci sarà memoria libera per il suo working-set

402