UNIVERSITÀ DEGLI STUDI DI NAPOLI
PARTHENOPE

Sistemi Operativi

Memoria Virtuale

LEZIONE 22

prof. Antonino Staiano

Corso di Laurea in Informatica – Università di Napoli Parthenope

antonino.staiano@uniparthenope.it

Concetti base

- **Memoria Virtuale**: Una gerarchia di memoria, formata da una memoria e un disco, che permette ad un processo di funzionare con solo alcune porzioni del suo spazio di indirizzamento in memoria
- La MMU traduce gli indirizzi logici in indirizzi fisici
- Il *gestore della memoria* virtuale è una componente software
  - Usa il *caricamento su richiesta*
  - Sfrutta la *località dei riferimenti* per migliorare le prestazioni

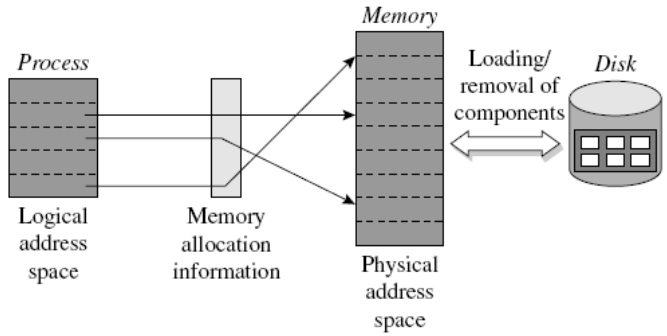


Figure 12.1 Overview of virtual memory.

Concetti base (cont.)

- Il gestore della memoria virtuale carica in memoria solo una componente dello spazio di indirizzamento di un processo per iniziarlo
  - Il componente che contiene l'indirizzo di avvio
- Le altre componenti sono caricate solo quando sono necessarie
  - Caricamento su richiesta
- Per limitare l'impegno di memoria per un processo, il gestore rimuove componenti del processo dalla memoria di volta in volta
  - Le componenti saranno ricaricate quando di nuovo necessario
- Le prestazioni di un processo in memoria virtuale dipendono dal tasso a cui è necessario caricare le sue componenti in memoria
  - Il gestore sfrutta la località dei riferimenti per ottenere tassi bassi

Concetti base (cont.)

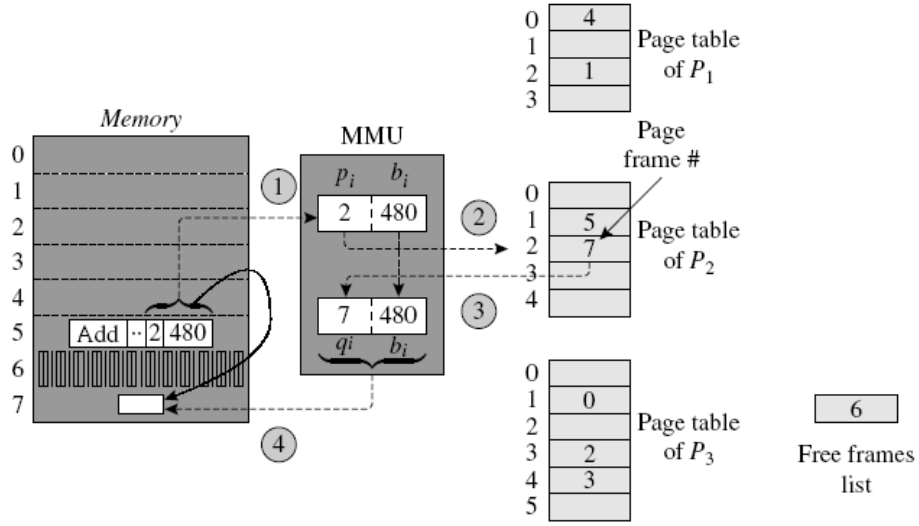
Table 12.1 Comparison of Paging and Segmentation

Issue	Comparison
Concept	A page is a fixed-size portion of a process address space that is identified by the virtual memory hardware. A segment is a logical entity in a program, e.g., a function, a data structure, or an object. Segments are identified by the programmer.
Size of components	All pages are of the same size. Segments may be of different sizes.
External fragmentation	Not found in paging because memory is divided into page frames whose size equals the size of pages. It occurs in segmentation because a free area of memory may be too small to accommodate a segment.
Internal fragmentation	Occurs in the last page of a process in paging. Does not occur in segmentation because a segment is allocated a memory area whose size equals the size of the segment.
Sharing	Sharing of pages is feasible subject to the constraints on sharing of code pages described later in Section 12.6. Sharing of segments is freely possible.

Memoria Virtuale con Paginazione

- La MMU esegue la traduzione degli indirizzi usando la tabella delle pagine

Indirizzo di memoria effettivo di (p_i, b_i) = indirizzo iniziale del frame che contiene la pagina p_i + b_i



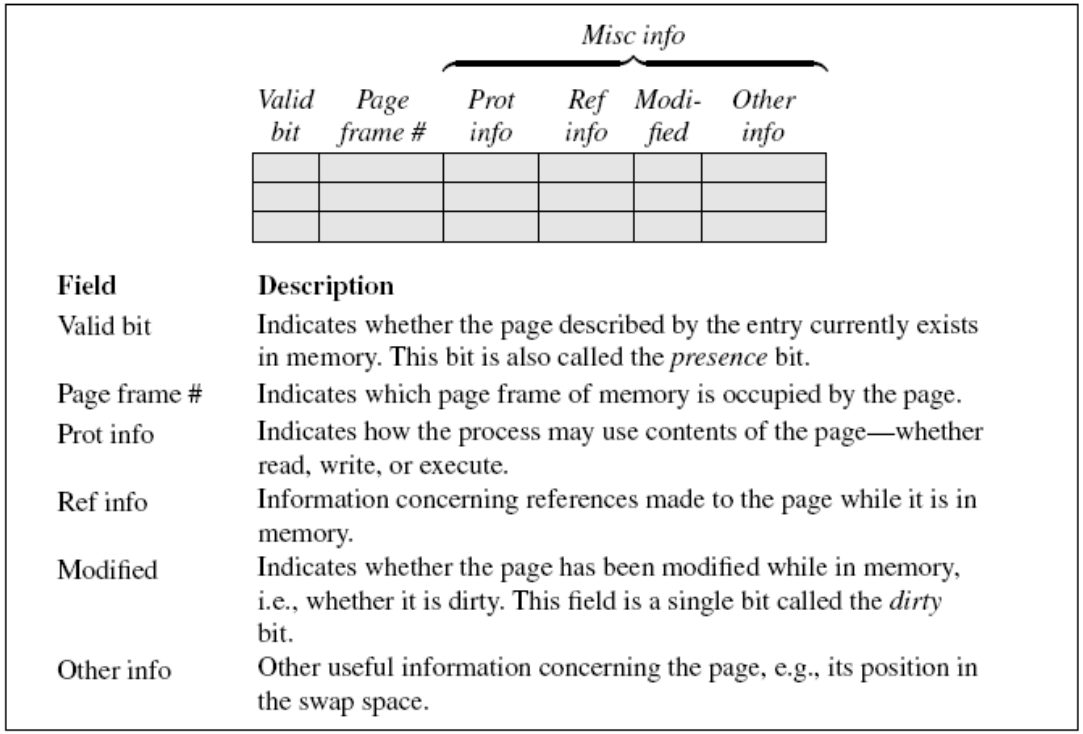
Indirizzo operando
2528 = 2 x 1024 + 480

Figure 12.2 Address translation in virtual memory using paging.

Paginazione su richiesta

- Sul disco è gestita una copia dell'intero spazio di indirizzamento di un processo
  - L'area del disco è chiamata spazio di swap del processo
- All'avvio del processo, il gestore della memoria virtuale alloca lo spazio di swap del processo e vi copia codice e dati
- *Page-in*: caricamento in memoria da disco della pagina del processo che riferisce un'istruzione o dato in una pagina non in memoria
- *Page-out*: rimozione dalla memoria di una pagina e sua memorizzazione su disco se modificata dall'ultimo caricamento
- *Sostituzione della pagina*: caricamento di una pagina in un frame che conteneva un'altra pagina
  - Eventualmente comporta un page-out se la pagina da sostituire è stata modificata
  - Page-in per caricare la nuova pagina

Paginazione su richiesta: tabella delle pagine



Field	Description
Valid bit	Indicates whether the page described by the entry currently exists in memory. This bit is also called the *presence* bit.
Page frame #	Indicates which page frame of memory is occupied by the page.
Prot info	Indicates how the process may use contents of the page—whether read, write, or execute.
Ref info	Information concerning references made to the page while it is in memory.
Modified	Indicates whether the page has been modified while in memory, i.e., whether it is dirty. This field is a single bit called the *dirty* bit.
Other info	Other useful information concerning the page, e.g., its position in the swap space.

Figure 12.3 Fields in a page table entry.

Paginazione su richiesta: fault di pagina

- La MMU genera un interrupt di page fault se la pagina che contiene l'indirizzo logico non è in memoria

Table 12.2 Steps in Address Translation by the MMU

Step	Description
1. Obtain page number and byte number in page	A logical address is viewed as a pair (p_i, b_i), where b_i consists of the lower order n_b bits of the address, and p_i consists of the higher order n_p bits (see Section 11.8).
2. Look up page table	p_i is used to index the page table. A page fault is raised if the *valid bit* of the page table entry contains a 0, i.e., if the page in not present in memory.
3. Form effective memory address	The *page frame* # field of the page table entry contains a frame number represented as an n_f-bit number. It is concatenated with b_i to obtain the effective memory address of the byte.

- MMU e gestore memoria virtuale interagiscono per decidere quando la pagina di un processo deve essere caricata in memoria

Paginazione su richiesta (cont.)

E' generato un page fault interrupt perché il Valid bit della pagina 3 è 0

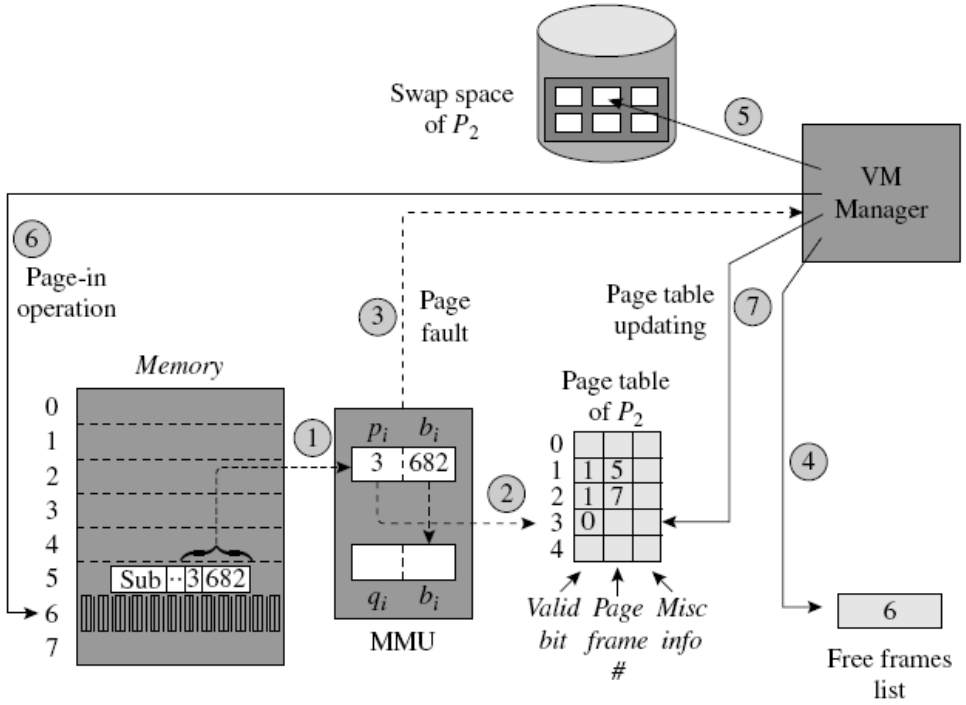


Figure 12.4 Demand loading of a page.

Passo 4: avvia operazione di I/O per caricare la pagina nel frame 6

- Completato I/O il gestore aggiorna l'entrata della tabella delle pagine

Sostituzione di pagina

- Con il page fault, la pagina richiesta è caricata in un frame di pagina libero
- Se nessun frame è libero, il gestore della memoria virtuale esegue un'operazione di sostituzione della pagina
  - Algoritmo di sostituzione della pagina
  - Avviato un page-out se la pagina è dirty (il bit modified è 1)
- Page-in e page-out: I/O di pagina o traffico di pagina (movimento di pagine da e verso la memoria)
- Il processo che provoca un page fault è bloccato (blocked) fino a che la pagina è caricata in memoria

Tempo di accesso in memoria effettivo (EAT)

- Il tempo di accesso effettivo nella paginazione su richiesta

$$\text{Effective memory access time} = pr_1 \times 2 \times t_{\text{mem}} + (1 - pr_1) \times (t_{\text{mem}} + t_{\text{pfh}} + 2 \times t_{\text{mem}}) \quad \textbf{(12.2)}$$

hit ratio memoria →

pr_1	probability that a page exists in memory
t_{mem}	memory access time
t_{pfh}	time overhead of page fault handling

- L'EAT si può abbassare riducendo i page fault
- Un modo consiste nel caricare le pagine prima che siano necessarie ad un processo
  - Windows: carica una pagina all'occorrenza di un page fault ed anche alcune pagine adiacenti
  - Linux: consente ad un processo di specificare quali pagine dovrebbero essere precaricate
    - Il programmatore può usare questa possibilità per migliorare il tempo di accesso effettivo

Sostituzione delle pagine

- Legge (empirica) della località dei riferimenti: gli indirizzi logici usati dai processi, nel breve, tendono a raggrupparsi in specifiche porzioni del loro spazio di indirizzamento logico
  - L'esecuzione delle istruzioni è generalmente sequenziale
    - 10-20% sono istruzioni di branching
  - I processi tendono ad eseguire operazioni simili sugli elementi di dati non scalari (es., array)
- Istruzioni e riferimenti a dati tendono ad essere in prossimità di un'istruzione precedente o di un dato referenziato
- *Località corrente* di un processo: insieme di pagine referenziate in poche istruzioni precedenti
  - Abbassa il numero di page fault
- I page fault sono sempre possibili, infatti se definiamo
  - *Regione di prossimità* di un indirizzo logico a_i: tutti gli indirizzi logici in prossimità dell'indirizzo a_i, allora
    - La regione di prossimità non entra in una pagina
    - Un'istruzione o dato referenziato da un processo può non essere in prossimità dei riferimenti precedenti
      - *Shift della località* di un processo

Sostituzione delle pagine (cont.)

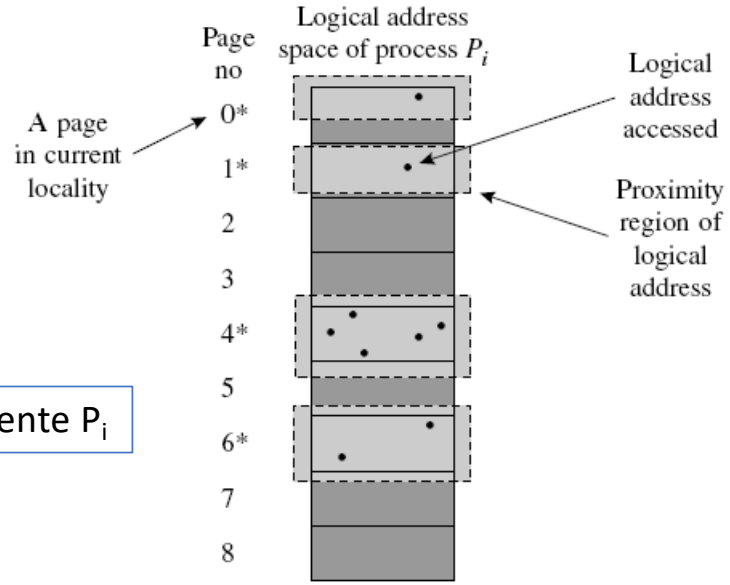


Figure 12.5 Proximity regions of previous references and current locality of a process.

* = pagine località corrente P_i

Allocazione della memoria ad un processo

- Quanta memoria allocare ad un processo
- Trashing

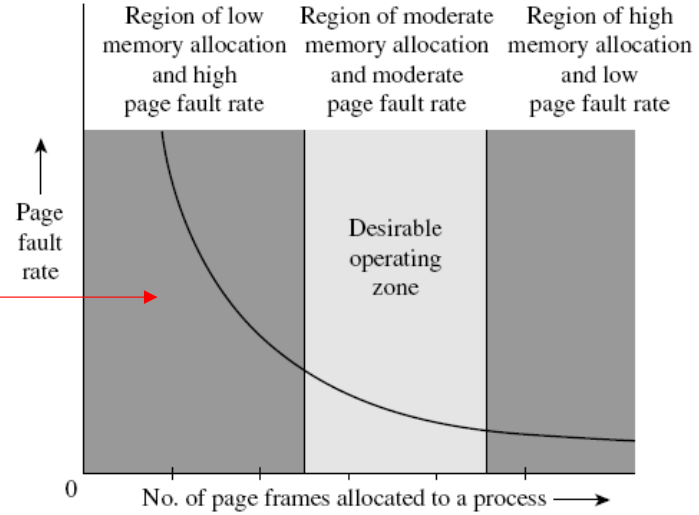


Figure 12.6 Desirable variation of page fault rate with memory allocation.

Dimensione della pagina ottimale

- La dimensione della pagina è definita dallo hardware
- La dimensione della pagina determina
  - Numero di bit richiesti per rappresentare il numero di byte in una pagina
  - Spreco di memoria causata da frammentazione interna
  - Dimensione della tabella delle pagine per un processo
  - I tassi di page fault quando una quantità fissa di memoria è allocata ad un processo
- L'uso di dimensioni di pagina maggiori rispetto al valore ottimo implica page fault maggiori
  - Tradeoff tra costi HW ed operazioni efficienti

Hardware di paginazione

- Il *Registro indirizzo tabella delle pagine* (PTAR) punta all'inizio di una tabella delle pagine

Dato l'indirizzo logico (p_i,b_i), la MMU calcola:
Indirizzo entrata della PT per p_i
<PTAR> + p_i x l_{PT_entry}
Dove
- l_{PT_entry} è la lunghezza di un'entrata della PT
- <PTAR> = contenuto di PTAR

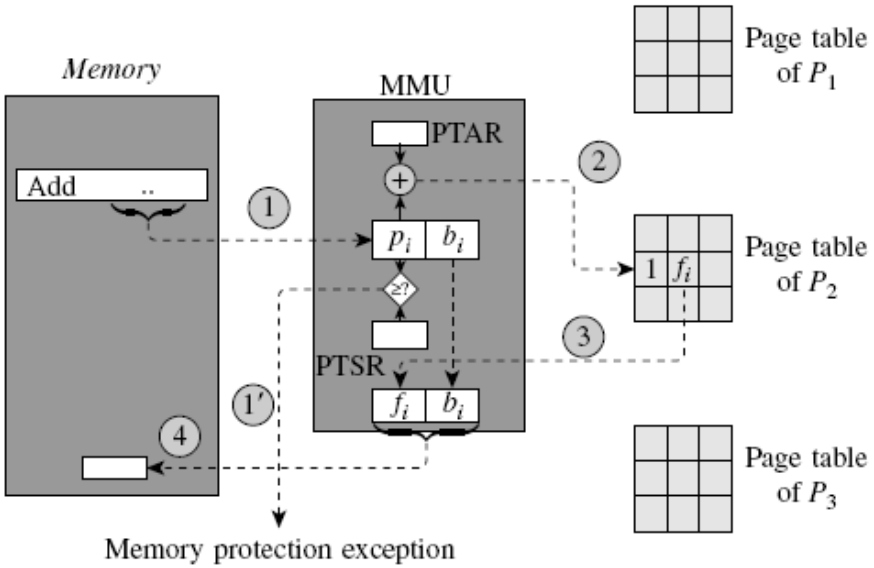


Figure 12.7 Address translation in a multiprogrammed system.

Hardware di paginazione (cont.)

Table 12.3 Functions of the Paging Hardware

Function	Description
Memory protection	Ensure that a process can access only those memory areas that are allocated to it.
Efficient address translation	Provide an arrangement to perform address translation efficiently.
Page replacement support	Collect information concerning references made to pages. The virtual memory manager uses this information to decide which page to replace when a page fault occurs.

HW di paginazione: Protezione della memoria

- E' generata una violazione della protezione della memoria se:
  - Un processo cerca di accedere ad una pagina che non esiste
  - Il processo viola i suoi privilegi di accesso (alla pagina)
- E' implementato attraverso:
  - Il *registro dimensione della tabella delle pagine* (PTSR) della MMU
    - Il kernel memorizza il numero delle pagine contenute in un processo nel suo PCB
      - Carica il numero dal PCB nel PTSR quando il processo è schedulato
      - E' generato un interrupt di violazione se il numero di pagina in (p_i, b_i) è maggiore del contenuto di PTSR
  - Il campo *Prot info* dell'entrata della pagina nella tabella delle pagine
    - Contiene i privilegi di accesso di un processo ad una pagina
      - Codificati come bit. Ogni bit un permesso (es., read, write, ecc.)
    - Durante la traduzione dell'indirizzo, la MMU controlla tali informazioni con il tipo di accesso che sta effettuando
      - Se non combaciano, genera un interrupt di violazione di protezione della memoria

HW di paginazione: Traduzione indirizzo e generazione page fault

- *Translation look-aside buffer* (TLB): memoria associativa piccola e veloce usata per accelerare la traduzione dell'indirizzo

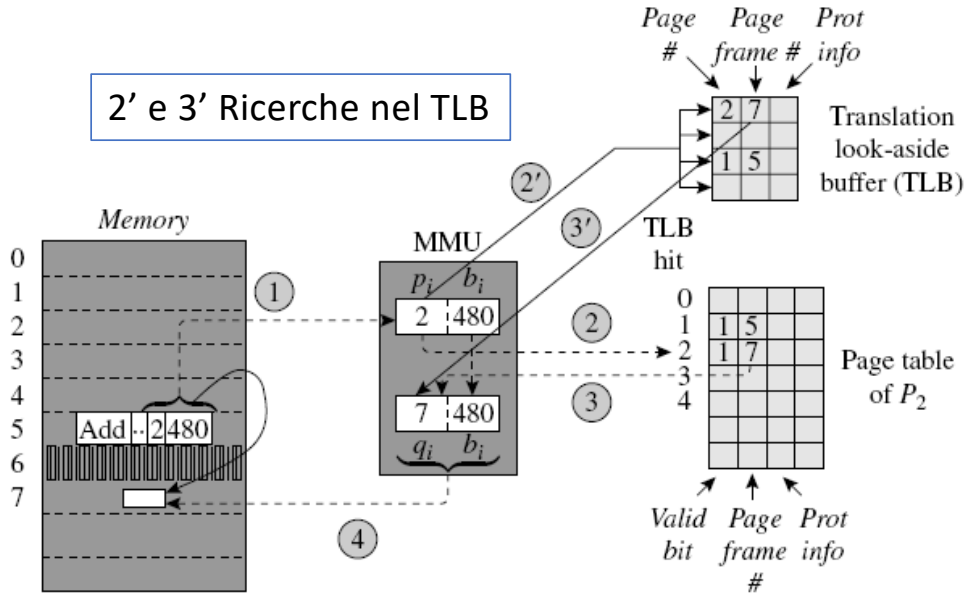


Figure 12.8 Address translation using the translation look-aside buffer and the page table.

Traduzione indirizzo e generazione page fault (cont.)

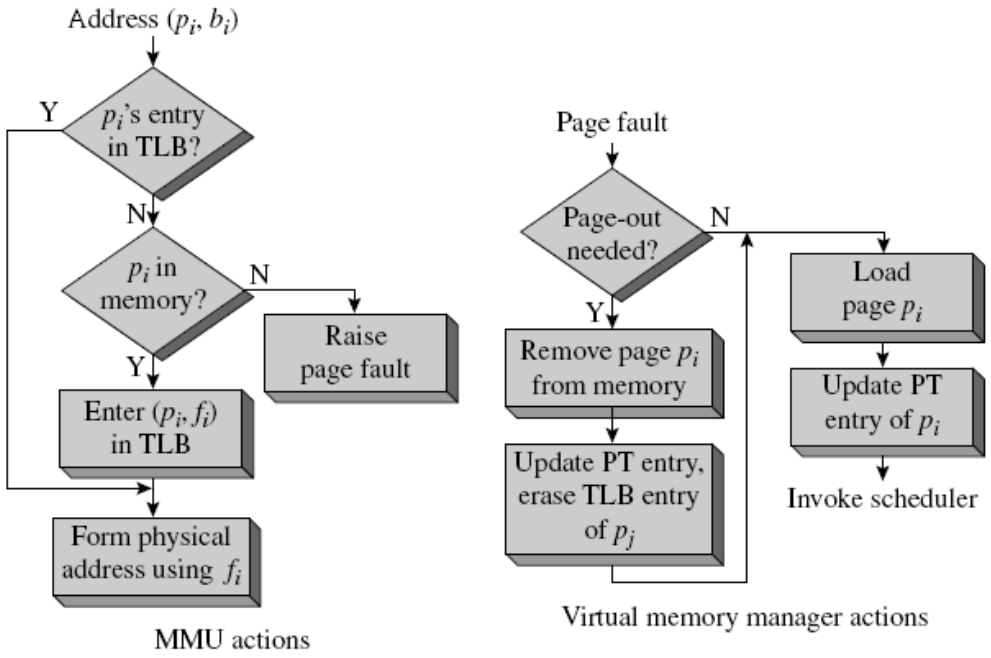


Figure 12.9 Summary of address translation of (p_i, b_i) (note: PT = page table).

- I TLB possono essere gestiti in HW o SW

Traduzione indirizzo e generazione page fault (cont.)

- Alcune caratteristiche comuni ai due approcci
  - È usato un algoritmo di sostituzione per decidere quale entrata del TLB deve essere sovrascritta se c'è una nuova entrata
- L'uso del TLB può minare la protezione se la MMU esegue la traduzione con indirizzi con entrate del TLB create durante l'esecuzione di altri processi, per cui
  - Ogni entrata del TLB può contenere l'id del processo in esecuzione nel momento della creazione dell'entrata
    - la MMU evita di usarla quando altri processi sono in esecuzione
  - alternativamente, il kernel deve scaricare (flush) il TLB mentre esegue la commutazione tra processi

Traduzione indirizzo e generazione page fault (cont.)

$$\text{Effective memory access time} = pr_1 \times 2 \times t_{\text{mem}} + (1 - pr_1) \times (t_{\text{mem}} + t_{\text{pfh}} + 2 \times t_{\text{mem}})$$

$$\begin{aligned}\text{Effective memory access time} = {} & pr_2 \times (t_{\text{TLB}} + t_{\text{mem}}) + (pr_1 - pr_2) \times (t_{\text{TLB}} + 2 \times t_{\text{mem}}) \\ & + (1 - pr_1) \times (t_{\text{TLB}} + t_{\text{mem}} + t_{\text{pfh}} + t_{\text{TLB}} + 2 \times t_{\text{mem}})\end{aligned} \qquad \textbf{(12.3)}$$

pr_1	probability that a page exists in memory
pr_2	probability that a page entry exists in TLB
t_{mem}	memory access time
t_{TLB}	access time of TLB
t_{pfh}	time overhead of page fault handling

TLB hit ratio

- Sono usati alcuni meccanismi per migliorare le prestazioni:
  - Le entrate wired TLB per le pagine del kernel: mai sostituite
  - Le superpagine

Supporto per la sostituzione delle pagine

- Il gestore della memoria virtuale ha bisogno di informazioni per minimizzare i page fault e il numero di operazioni page-in e page-out
  - L'istante in cui una pagina è stata usata l'ultima volta
    - Costoso fornire bit sufficienti per tale scopo
    - Soluzione: usare un singolo bit di riferimento
  - Se una pagina è dirty
    - Una pagina è pulita se non è dirty
    - Soluzione: il bit modified nell'entrata della tabella delle pagine

Organizzazione della PT in pratica

- Un processo con uno spazio di indirizzamento grande richiede una tabella delle pagine grande che occupa molta memoria
- Soluzioni:
  - Tabella delle pagine invertita
    - Descrive i contenuti di ogni frame di pagina
      - Dimensione regolata dalla dimensione della memoria
      - Indipendente dal numero e dimensioni dei processi
      - Contiene coppie della forma (id programma, # pagina)
    - Limite: le informazioni su una pagina devono essere cercate
  - Tabella delle pagine multilivello
    - La tabella delle pagine di un processo è paginata
    - È usata una PT di livello superiore per accedere alla pagine della PT
      - Se la PT di tale livello è grande, può essere a sua volta paginata …
- In ambo gli approcci è usato il TLB per ridurre i riferimenti alla memoria durante la traduzione degli indirizzi

Tabella delle pagine invertita

- Ogni entrata della IPT è una coppia ordinata (id processo, numero pagina)
- La coppia (P, p_i) nell'entrata f_i-esima indica che il frame f_i è occupato dalla pagina p_i del processo P
- Lo scheduler, selezionato un processo, ne preleva l'id dal PCB e lo invia in un registro della MMU

L'uso di tabelle hash
Velocizza la ricerca

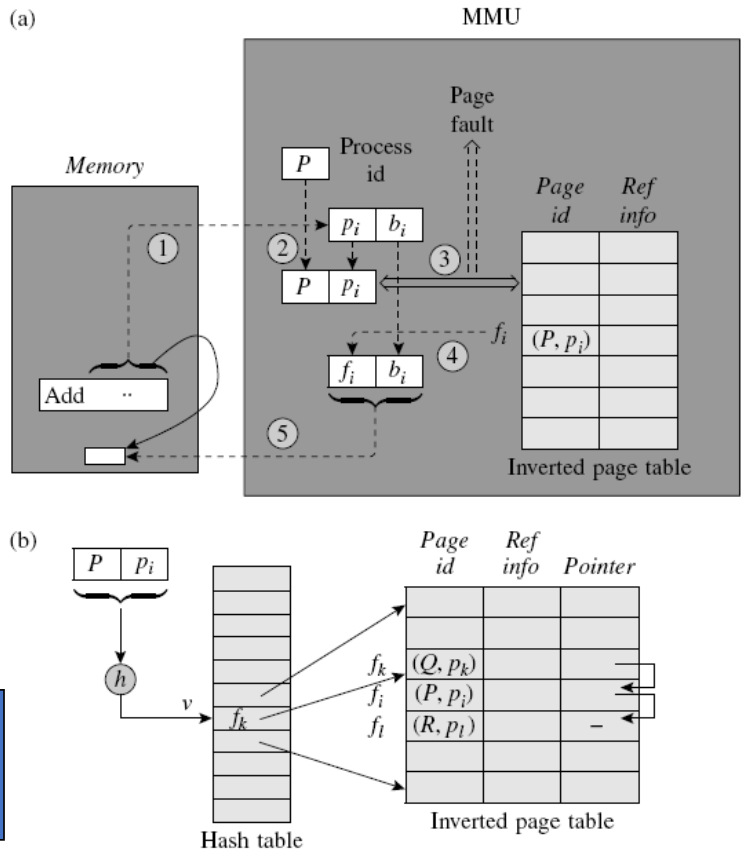


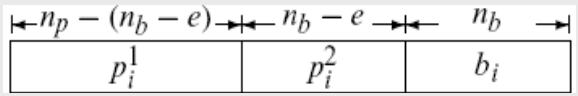
Figure 12.10 Inverted page table: (a) concept; (b) implementation using a hash table.

Tabella delle Pagine Multilivello

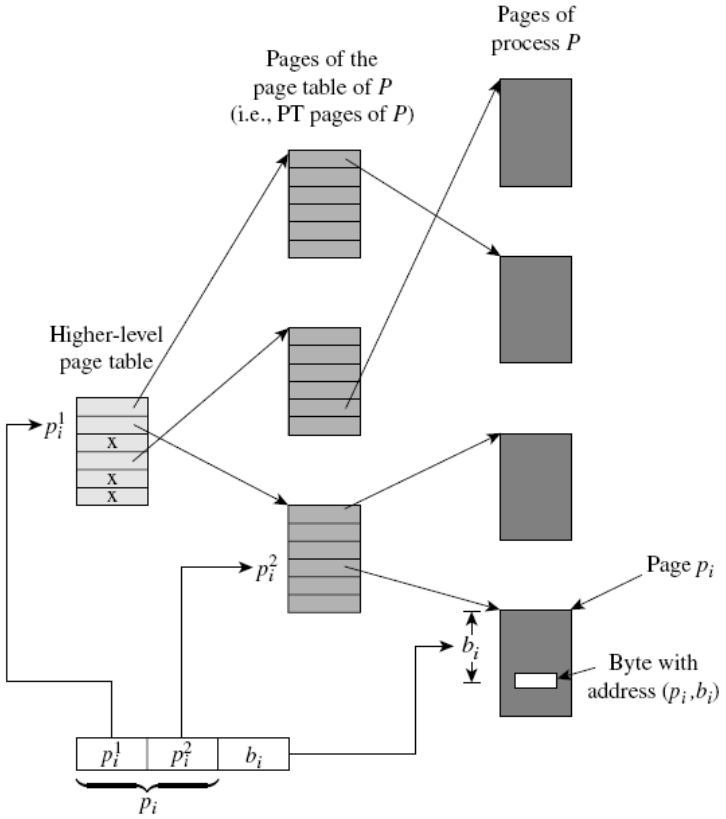
- L'idea è paginare la tabella delle pagine e di caricare le sue pagine su richiesta
- E' richiesto un indirizzamento a due livelli in cui una PT di alto livello contiene le informazioni sulle pagine della PT e la PT contiene info sulle pagine del processo
- La memoria contiene due tipi di pagine: pagine dei processi e pagine delle PT dei processi che chiamiamo pagine PT

Tabelle delle pagine multilivello

- Se la dimensione di un'entrata della tabella è 2^e byte, il numero di entrate della tabella in una pagina PT è $2^{n_b}/2^e$
- L'indirizzo logico (p_i , b_i) è raggruppato in tre campi:



  - La pagina in PT con p_i^1 contiene l'entrata per p_i
  - p_i^2 è il numero di entrata per p_i nella pagina della PT
  - b_i



Esempio PT Multilivello

- Abbiamo qui un indirizzo logico a 32 bit
- Supponiamo un indirizzamento a due livelli e pagine di 4KB (2^12)
- Allora lo spazio virtuale totale di 2^32 (4 GB) è composto di 2^20 pagine
  - Una prima tabella che occupa 2^10 pagine può essere tenuta in memoria virtuale e mappata ad una TP di alto livello con 2^10 entrate che occupa 4 KB (2^12) di memoria principale

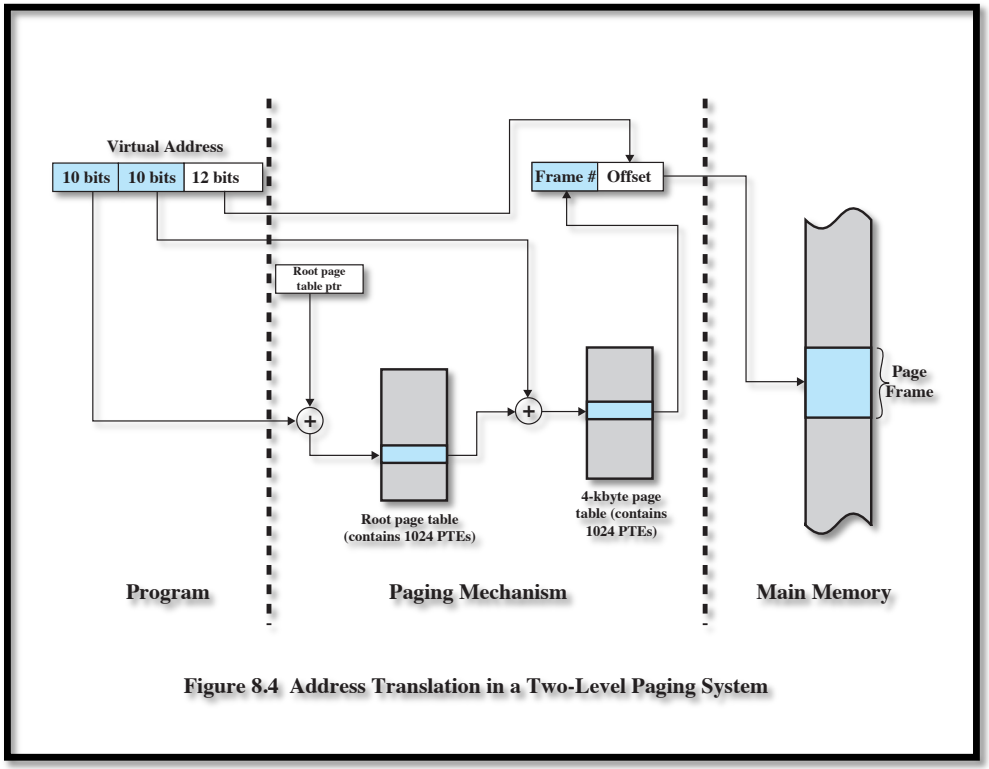


Figure 8.4 Address Translation in a Two-Level Paging System