Memoria Virtuale

Lezione 9-10
Sistemi Operativi

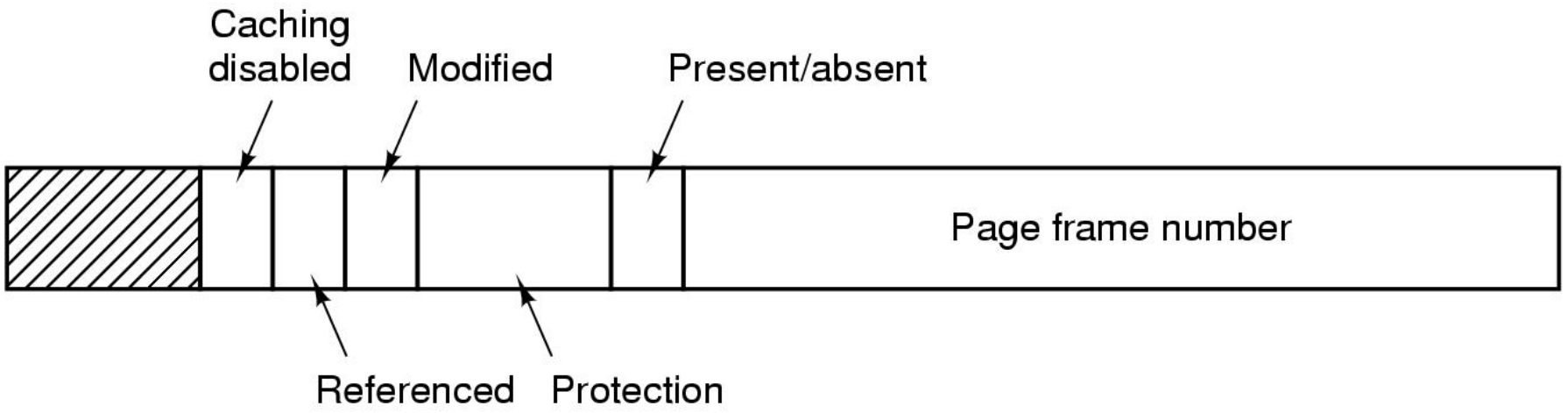
Problemi implementativi

- Dal punto di vista del SO la realizzazione di un sistema di VM richiede la soluzione di una serie di problemi:
  - Individuazione di strutture dati adeguate per le tabelle delle pagine in memoria centrale
  - Procedura di gestione del page fault
  - Back up delle istruzioni
  - Gestione delle tabelle delle pagine in memoria secondaria
  - Interferenze con I/O

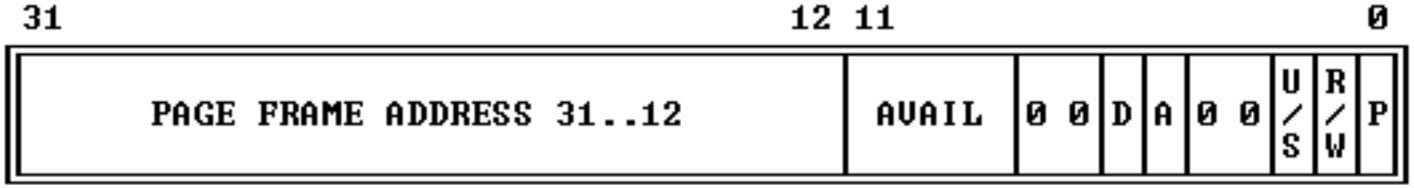
Tabelle delle pagine

- Le principali criticità da considerare sono legati a criteri di
  - Efficienza
  - Spazio
- Un sistema che offre la paginazione deve gestire le seguenti strutture dati
  - Tabella dei frame
  - Tabella delle pagine, che definisce l'associazione pagina → frame
    - una per ogni processo in esecuzione o in attesa di esecuzione

Elemento di una tabella delle pagine



Page table entry (PTE)



P - PRESENT
R/W - READ/WRITE
U/S - USER/SUPERVISOR
D - DIRTY
AVAIL - AVAILABLE FOR SYSTEMS PROGRAMMER USE

NOTE: 0 INDICATES INTEL RESERVED. DO NOT DEFINE.

Paginazione: efficienza

- Procedura di caricamento di un programma in un ambiente multiprogrammato
  1. Long term scheduling verifica disponibilità dei frame in memoria centrale
  2. Il processo viene caricato in MC e la sua tabella delle pagine compilata e agganciata al PCB
  3. Quando il processo va in esecuzione la sua tabella delle pagine viene usata dalla MMU per il binding degli indirizzi, questa operazione compiuta per ogni accesso a memoria, deve essere resa efficiente con il supporto dell'HW

Ottimizzare l'accesso alla tabella delle pagine

- Esistono tre alternative
  - Registri specializzati
    - Molto efficiente in esecuzione ma molto costosa per tabelle di grosse dimensioni, soprattutto in context switch
  - Esiste un registro che indirizza la tabella delle pagine del processo in esecuzione, tra quelle di tutti i processi del sistema; la tabella sta in memoria
    - Lenta richiede più accessi a memoria centrale per recuperare le informazioni
  - Registri associativi
    - Implementano un Translation Lookaside Buffer di 32-1024 elementi

TLBs – Translation Lookaside Buffers

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

TLB

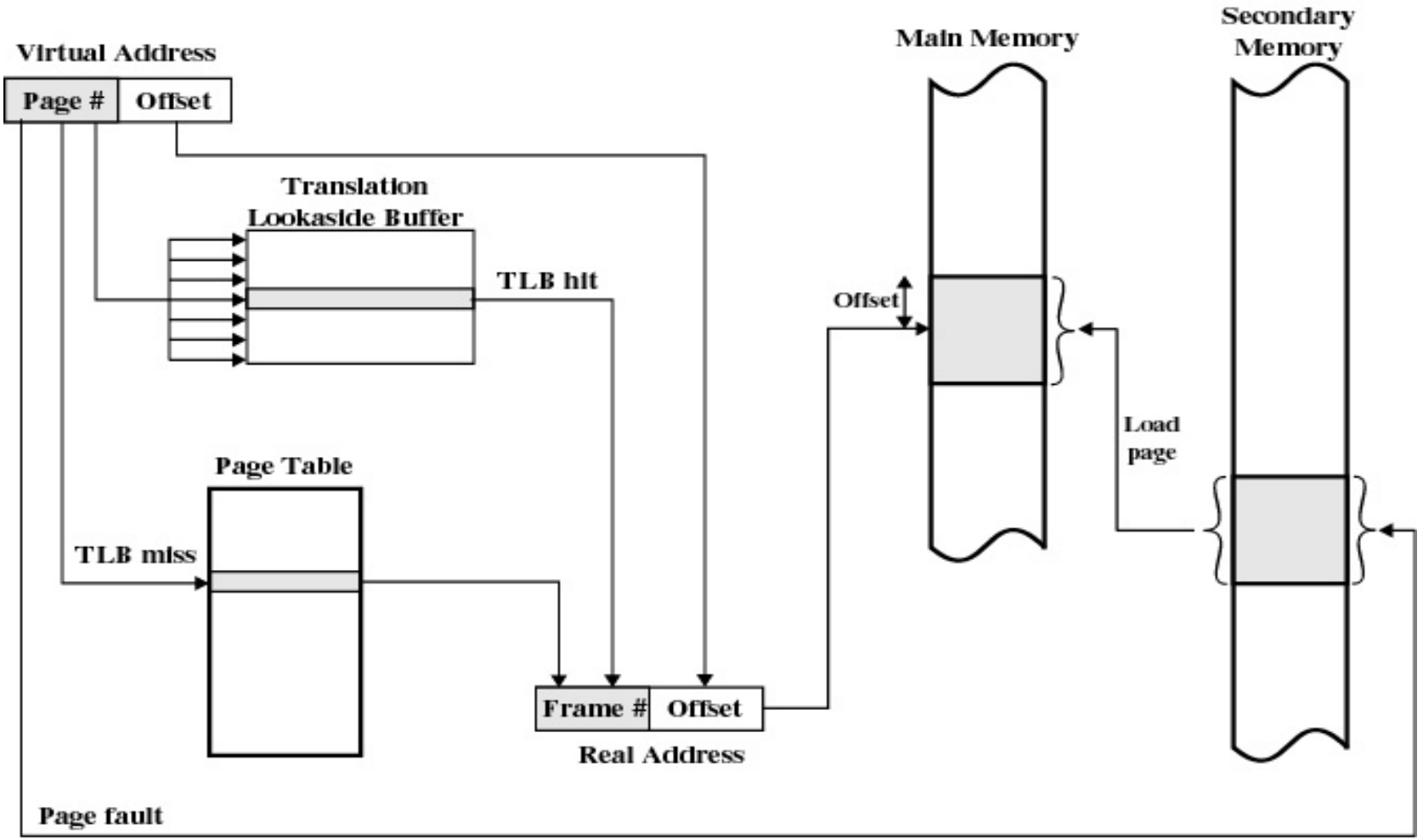


Figure 8.7 Use of a Translation Lookaside Buffer

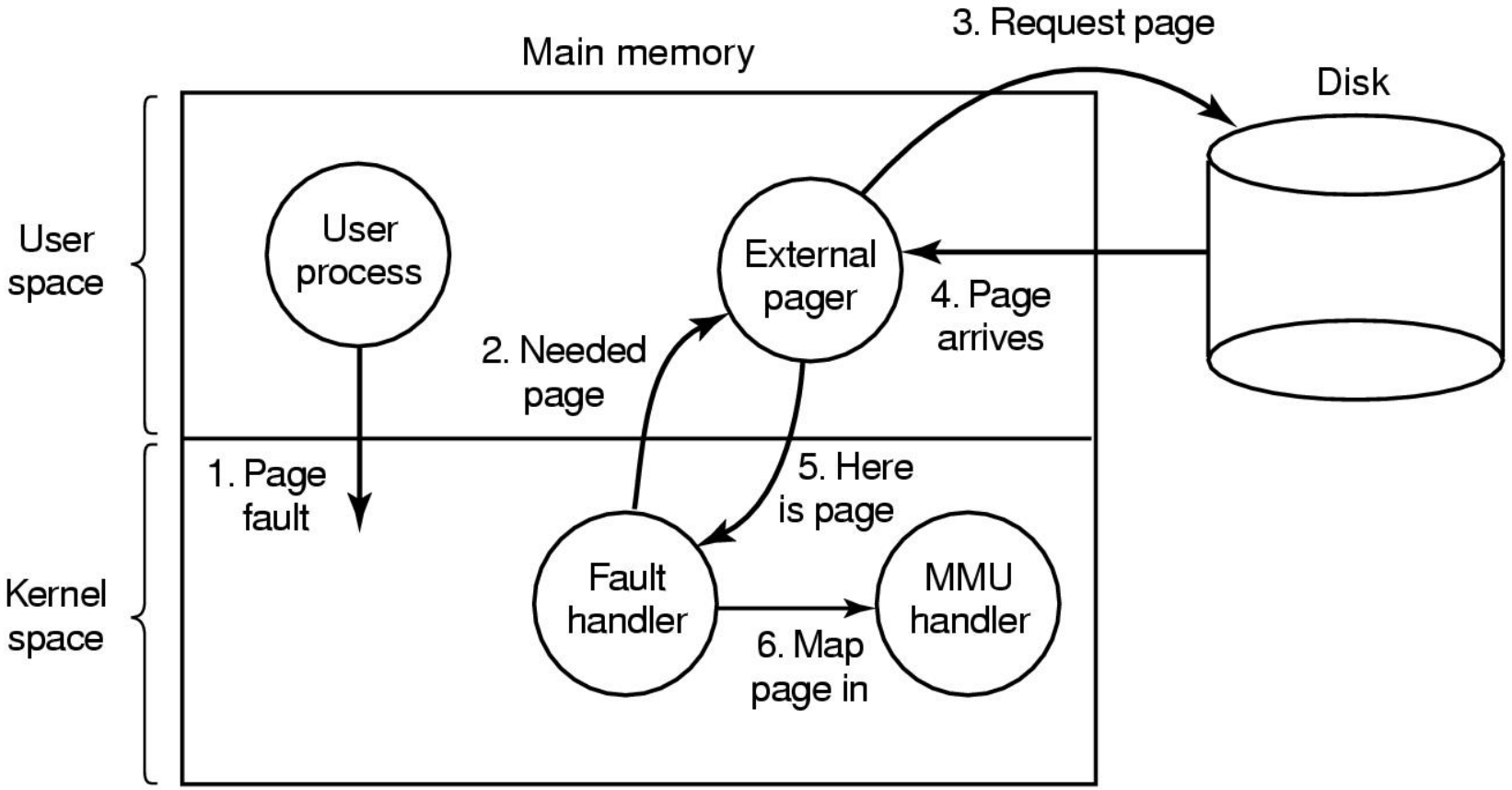
Gestione del Page Fault (1)

1. Viene generata una trap di page fault (da chi?)
2. Salvataggio del contesto interrotto da parte dell'hw
3. Richiamo procedura di gestione dell'eccezione
4. Il SO individua il numero di pagina da caricare
5. Il SO verifica la validita di questo dato e cerca dei frame liberi
6. Se il frame selezionato è dirty, riscrivilo su disco

Page Fault Handling (2)

6. Il SO trasferisce la nuova pagina in memoria
7. La tabella delle pagine viene aggiornata
8. I registri vengono ripristinati, tenendo conto che l'istruzione che ha generato il fault deve essere "ripristinata"
9. Il processo che ha generato il fault viene rimesso in esecuzione

Gestione del Page Fault



Tempo medio di accesso ad un dato/istruzione

- I parametri da considerare:
  - β = pfratio: frequenza di page fault
  - α = Hit ratio: frequenza di successo nell'accesso a TLB
  - T_{mem} = Tempo accesso a memoria
  - T_{TLB} = Tempo accesso a cache
  - T_{pf} = Tempo gestione page fault

La formula

- Effective Access Time (EAT):

$$EAT = (T_{TLB} + T_{mem})\,\alpha + (1 - \alpha)[(2T_{mem} + T_{TLB})(1-\beta) + T_{pf}\beta]$$

Esercizio

T_{mem}= 100 ns

T_{TLB} = 20 ns

T_{pf} = 20ms

hit ratio = 80%

Pfratio = 1%

(0.8)(20+100) + (0.2)[(20+100+100)*0.99 + 20.000.000 * 0,01] =

0.8*120 + 0.2*217,8 + 200.000 = 96 + 43,56 + 200.000 = 200139,56 ns

L'accesso ad un dato in memoria diventa 2000 volte più lento se confrontata con un sistema non paginato !!!

Per avere valori accettabili è necessario avere hit ratio molto più elevato (98%) e pfratio molto più basso (0.0008% cioè 1 ogni 125.000 accessi)

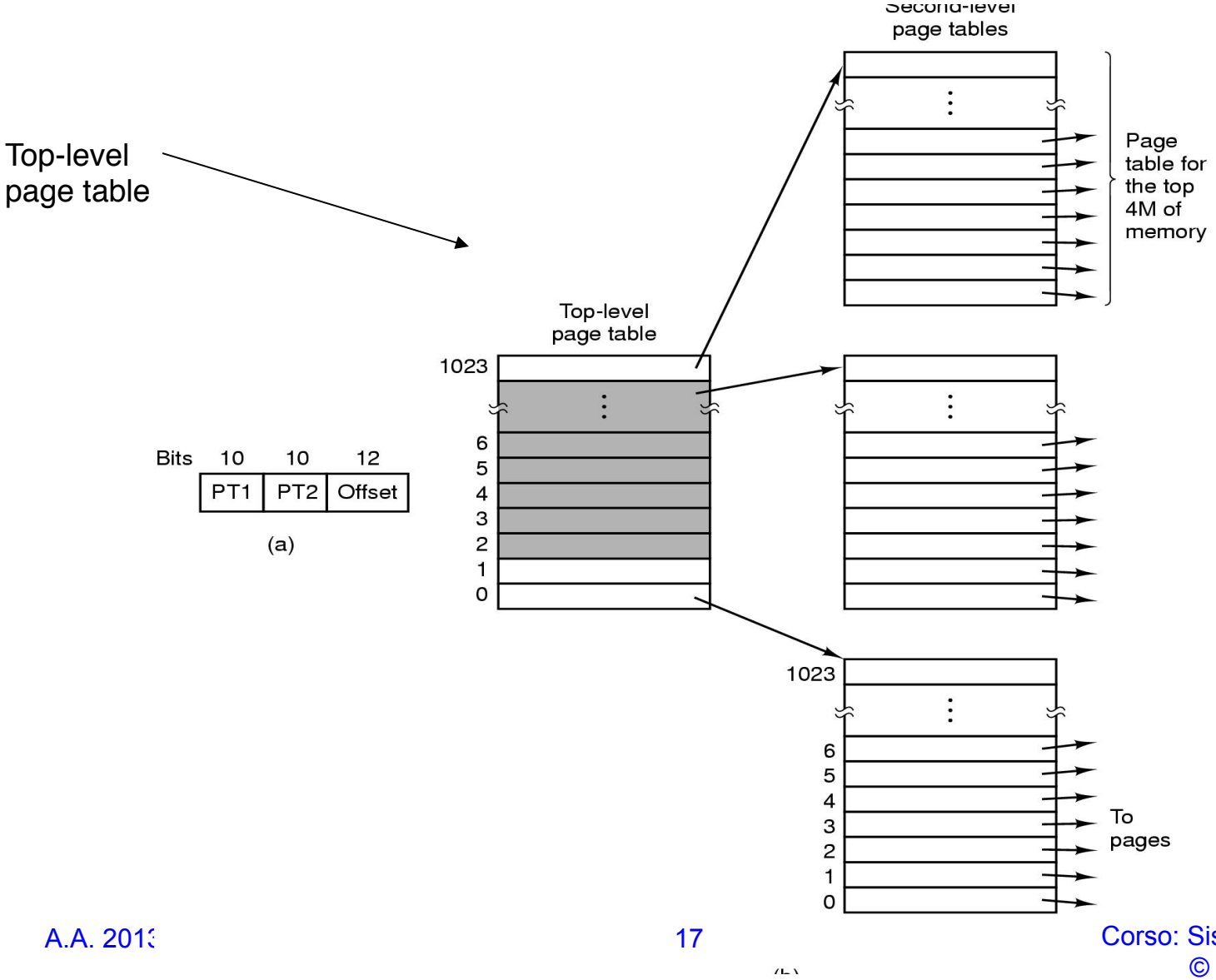
0.98*120 + (0.02)[220*0.999992 + 20.000.000*0.000008] =

= 117,6 + (0.02)*(219,9 + 160) = 117,6 + 7.5 circa 120ns

Tabella delle pagine: dimensioni

- La dimensione della tabella può essere molto elevata
  - Con pagine di 4KB
    - 1 milione di pagine con indirizzi di 32 bit, quindi 1 milione di elementi nella tabella
- Vanno individuate soluzioni per ovviare a questo inconveniente:
  - Tabelle delle pagine multilivello
  - Tabella delle pagine invertite

Tabella delle pagine a due livelli

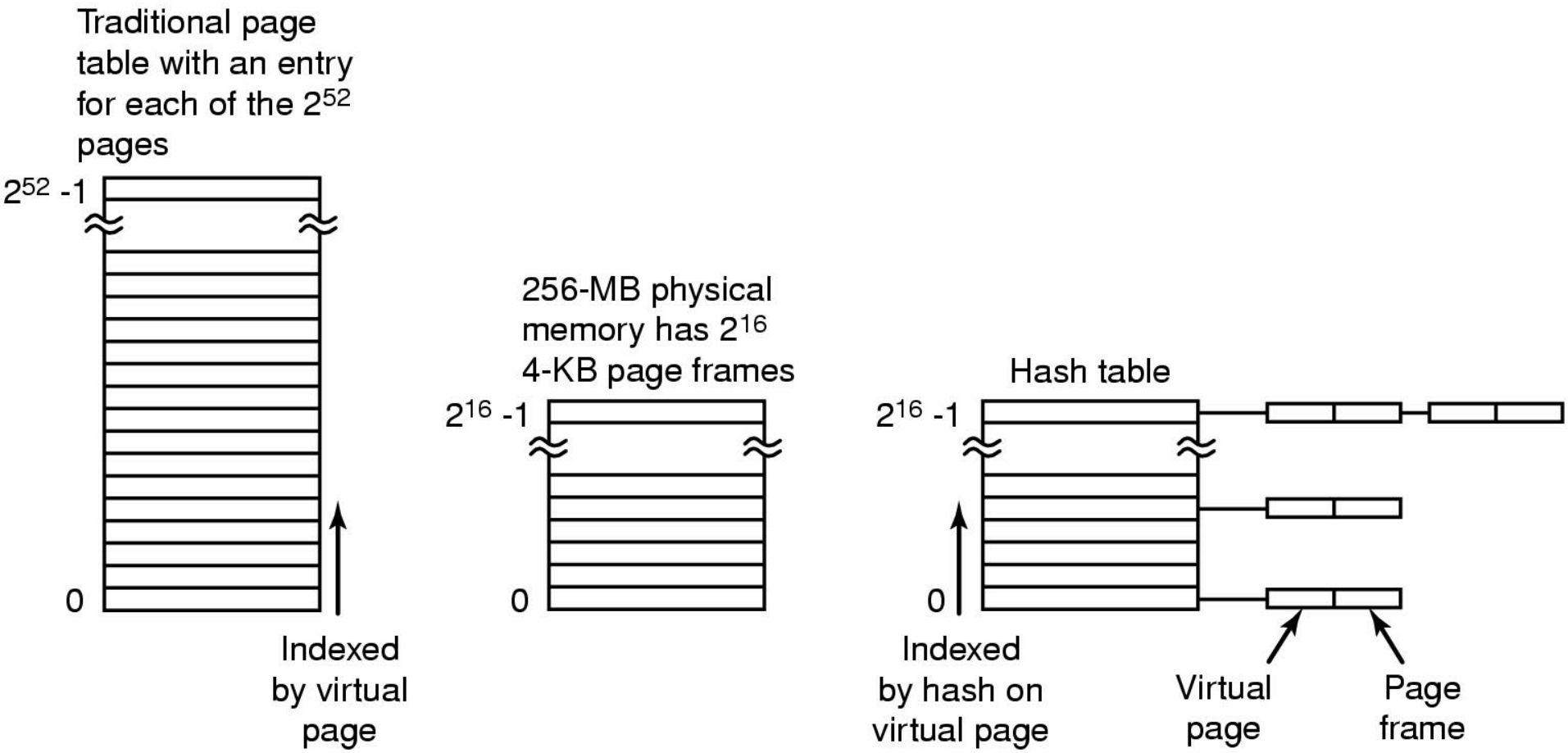


Esercizio

In un sistema paginato con tabella delle pagine a due livelli, tabelle di I e II livello contenti fino a16 elementi e dimensione della pagina di 32 byte, l'indirizzo virtuale 594 come viene "tradotto" dal compilatore?

1. Nella tripla <0, 7, 18>
2. Nella tripla <0, 0, 594>
3. Nella tripla <1, 2, 18>
4. Nessuna risposta è corretta

Tabella delle pagine invertita



La tabella indicizza i frame di memoria fisica

Un TLB permette la ricerca delle pagine più usate

Pagine su memoria secondaria

- Il sistema operativo usa solitamente un'area di swap su cui memorizza l'intero processo
  - Il processo viene interamente caricato in swap a partire dalla prima posizione disponibile
  - le pagine, che avranno un indirizzo ben definito sono poi prelevate/scaricate da quest'area
  - Ogni pagina in memoria ha una sua copia, non necessariamente aggiornata, in swap area

Pagine su memoria secondaria

- Il processo può cambiare dimensioni durante la sua vita
- Si possono allora adottare soluzioni diverse
  - Si allocano in swap area spazi diversi per Codice, dati e stack
  - Non prealloco alcun spazio in area di swap, che gestisco “on-demand”
  - Ci sarà una copia di una pagina in swap area solo a seguito di un’operazione di swap-out

Pagine su memoria secondaria

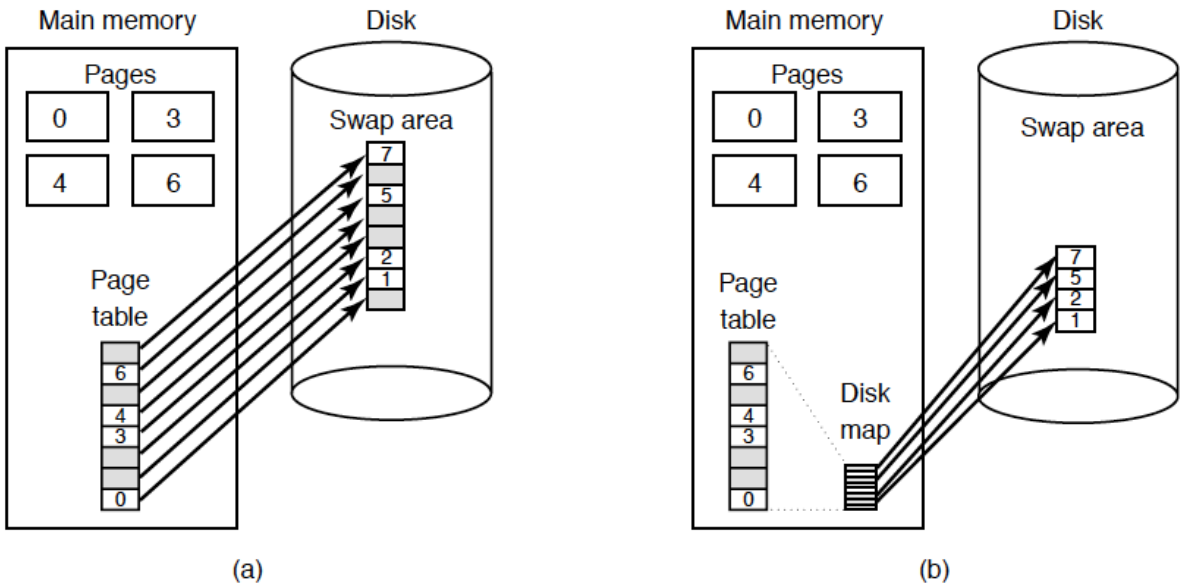
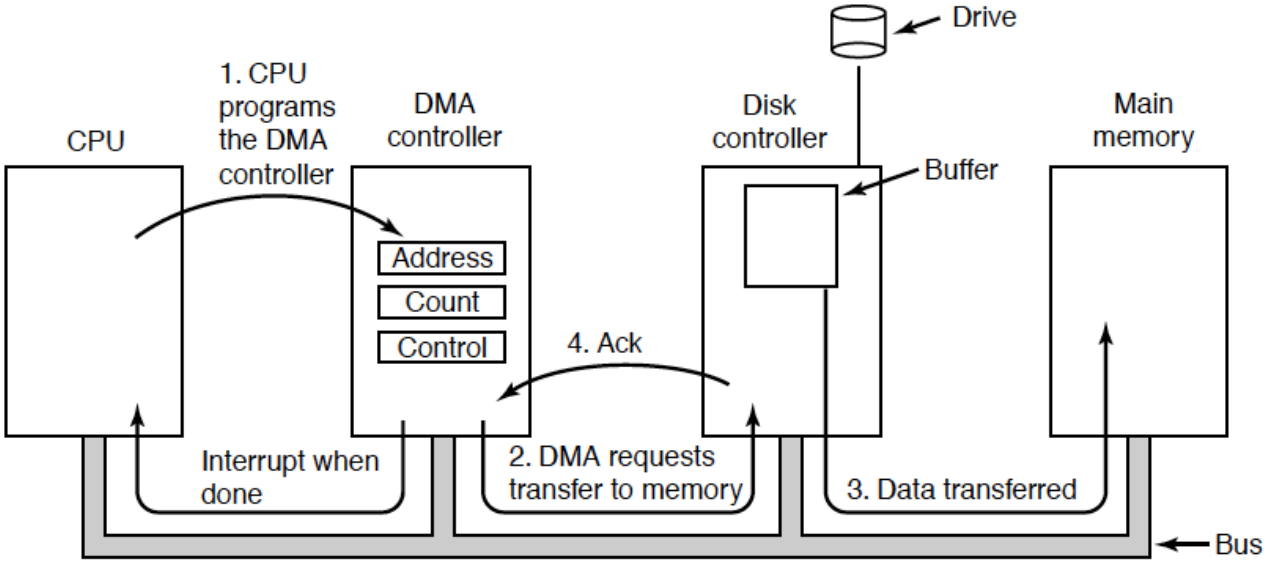


Fig. 4-33. (a) Paging to a static swap area. (b) Backing up pages dynamically.

Blocco pagine (pinning)



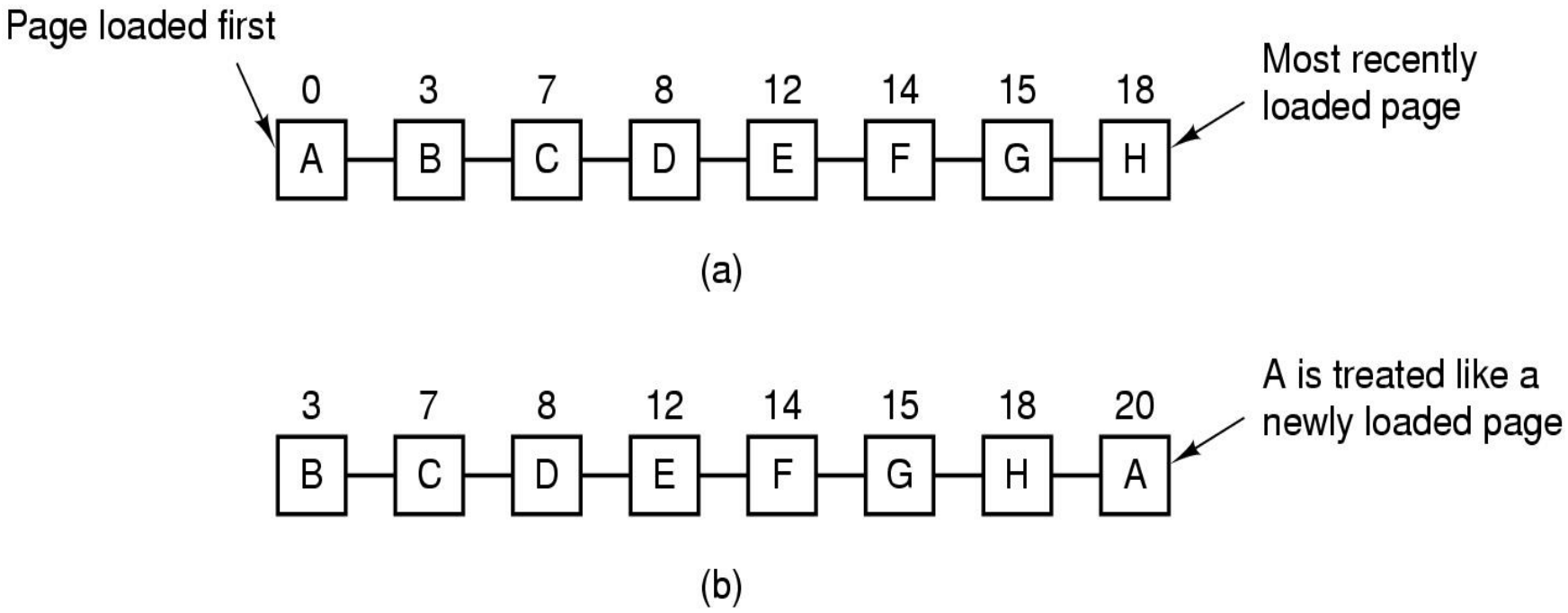
Page Replacement Algorithms

- Nelle fasi di paginazione di processi può verificarsi la seguente situazione:
  - si deve caricare in memoria una nuova pagina ma non ci sono frame disponibili, in questo caso è necessario:
    - Individuare le pagine da scaricare su disco
    - Salvare le pagine modificate durante l'esecuzione, quelle non modificate possono essere sovrascritte

FIFO Page Replacement Algorithm

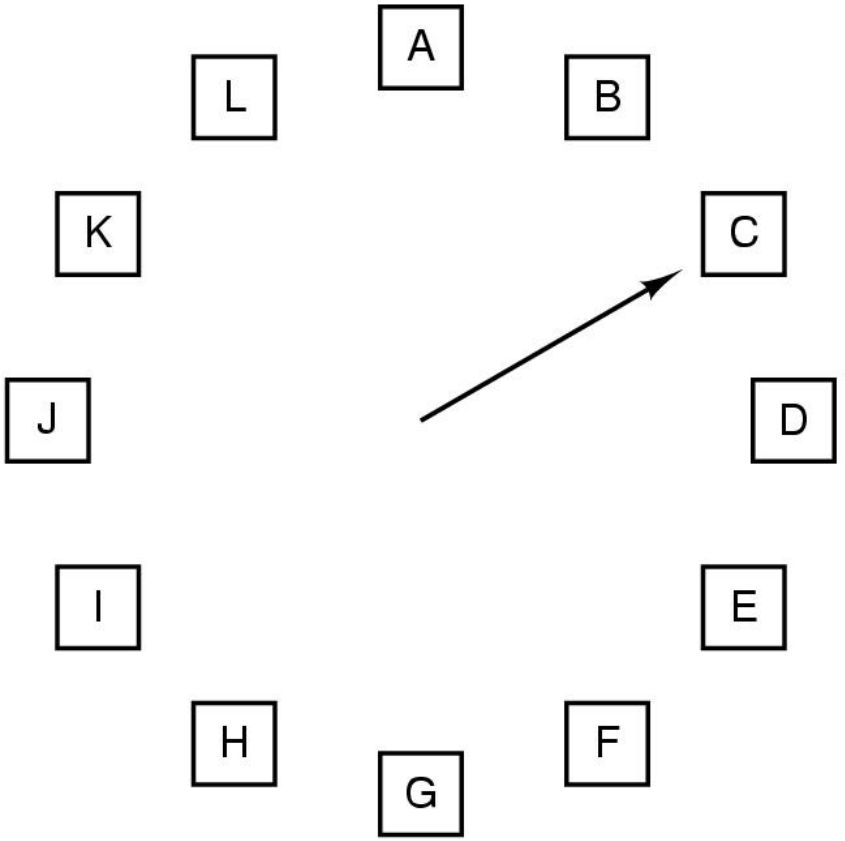
- Mantiene una lista delle pagine che rispetta l'ordine con cui le stesse sono state caricate in memoria (le pagine più "anziane" sono all'inizio della lista)
- Le pagine più anziane sono le prime ad essere sostituite
- Svantaggio
  - Sostituire una pagine che è referenziata spesso

Second Chance Page Replacement Algorithm



- Le pagine sono ordinate FIFO con un bit d'uso R
- La lista delle pagine quando si verifica un page fault all'istante 20, e *A* ha il bit *R* a uno

The Clock Page Replacement Algorithm



When a page fault occurs, the page the hand is pointing to is inspected. The action taken depends on the R bit:

R = 0: Evict the page

R = 1: Clear R and advance hand

- Come seconda chance ma usa una lista circolare, la lancetta punta alla pagina più vecchia

Optimal Page Replacement Algorithm

- Sostituire, tra quelle presenti in memoria, la pagina che sarà referenziata il più tardi possibile
  - Ottimale ma non realizzabile
- Possibili stime
  - Tenere traccia dell'uso passato delle pagine e inferire il futuro dal passato

Least Recently Used (LRU)

- Assume che le pagine usate recentemente saranno usate anche in un futuro prossimo
  - Elimina le pagine che non sono usate da più tempo
- L'implementazione di questo algoritmo richiede la realizzazione di una lista linkata, le pagine usate più recentemente in testa
  - È necessario aggiornare questa lista ad ogni riferimento a pagina !!
- Alternativamente è possibile mantenere un contatore per ciascun elemento della tabella delle pagine
  - La pagina vittima è quella con il contatore più basso
  - Il contatore viene azzerato periodicamente

Esercizio

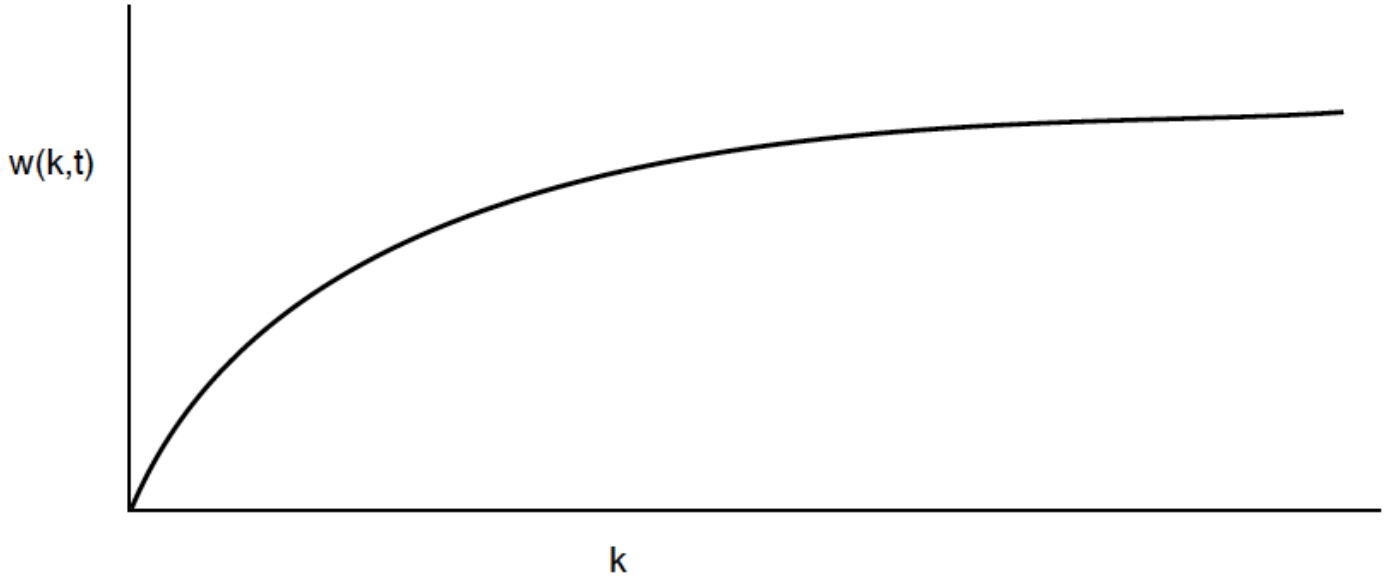
Si consideri un processo di 460 byte, che durante la sua esecuzione referenzia istruzioni ai seguenti indirizzi:

10, 11, 104, 170, 73, 309, 185, 245, 246, 434, 458, 364.

Il processo viene eseguito su un sistema in cui la memoria centrale ha dimensione 200 byte e le pagine hanno dimensione 100 byte. Quanti sono i page fault che l'esecuzione del suddetto processo genera in un sistema in cui vengono rispettivamente adottati gli algoritmi di rimpiazzamento pagine FIFO, LRU e ottimale?

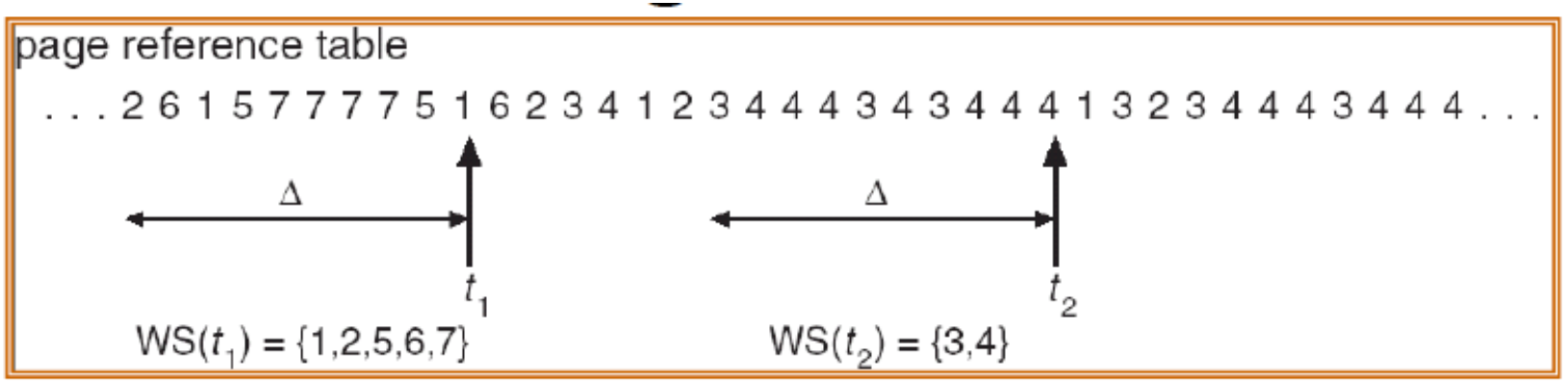
- 6, 7, 5
- 6, 5, 7
- 6, 7, 7
- nessuno dei valori riportati è corretto

Working Set



- Il working set è costituito dall'insieme delle pagine usate negli ultimi k riferimenti a memoria
- w(k,t) è la dimensione del working set all'istante t

Working set



- Δ (k sul libro di testo) è la finestra del WS
  - Se il suo valore è troppo piccolo non coglie per intero la località
  - Se troppo grande includerà più località

Working Set

- Il calcolo del WS è tutt'altro che banale: ogni volta che un processo accede ad una pagina il numero di questa pagina andrebbe memorizzato in un'opportuna struttura (ad es. registro a scorrimento)
- Per rendere più praticabile la determinazione del WS si individuano le pagine utilizzate da un processo negli ultimi T secondi di esecuzione

WS & trashing

- Thrashing → a process is busy swapping pages in and out
- I processi all'interno del sistema necessitano di una quantità di memoria maggiore rispetto a quella posseduta dal sistema
  - Ogni volta che una nuova pagina è portata in memoria, un'altra pagina del WS è scaricata
  - I processi spendono la maggior parte del loro tempo in attesa delle pagine
  - I dischi lavorano al 100%, ma la CPU è scarica

I motivi

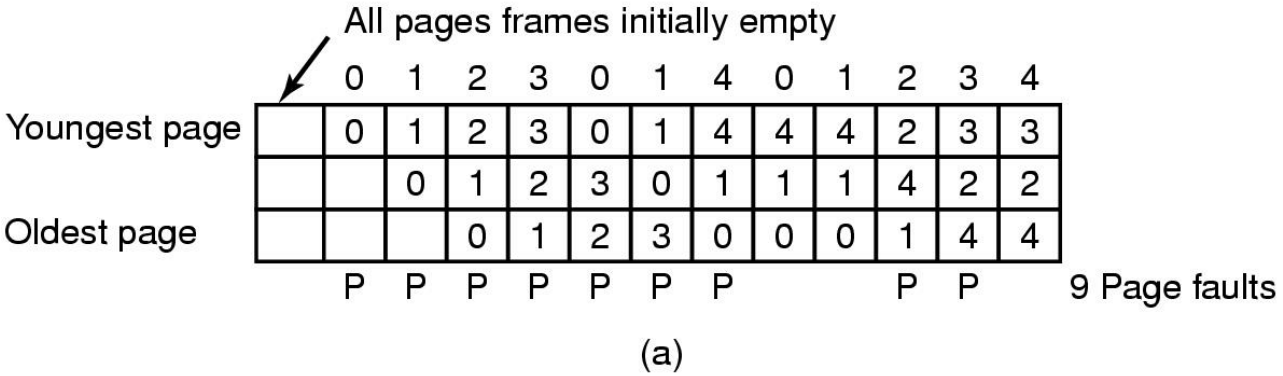
- Il trashing si può verificare per i seguenti motivi:
  - I processi hanno scarsa località e quindi non riutilizzano le stesse pagine (passato != futuro)
  - I processi riusano la memoria ma ne hanno bisogno di grosse quantità
  - I processi sono locali e “piccoli” ma ve ne sono troppi

Una soluzione

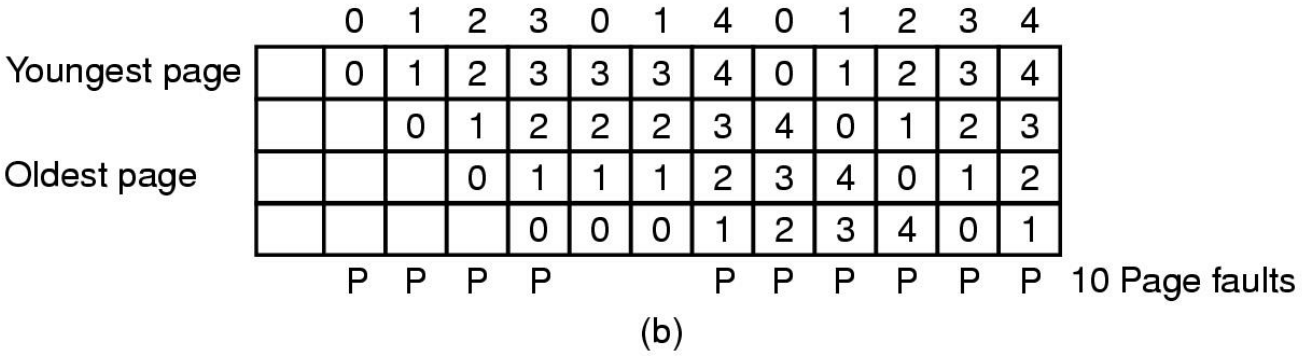
- Per risolvere il problema del trashing si può ricorrere al WS, se indichiamo con WSi il working set dell'i-esimo processo presente in memoria valgono le seguenti considerazioni:
  - $D = \Sigma|WSi| \equiv$ numero totale dei frame richiesti dai processi in memoria
  - Se *D > n.ro frame della memoria centrale* $\Rightarrow$Thrashing
  - In questo caso individua processi critici ed effettua swap out

ALTRE CRITICITÀ

Belady's Anomaly



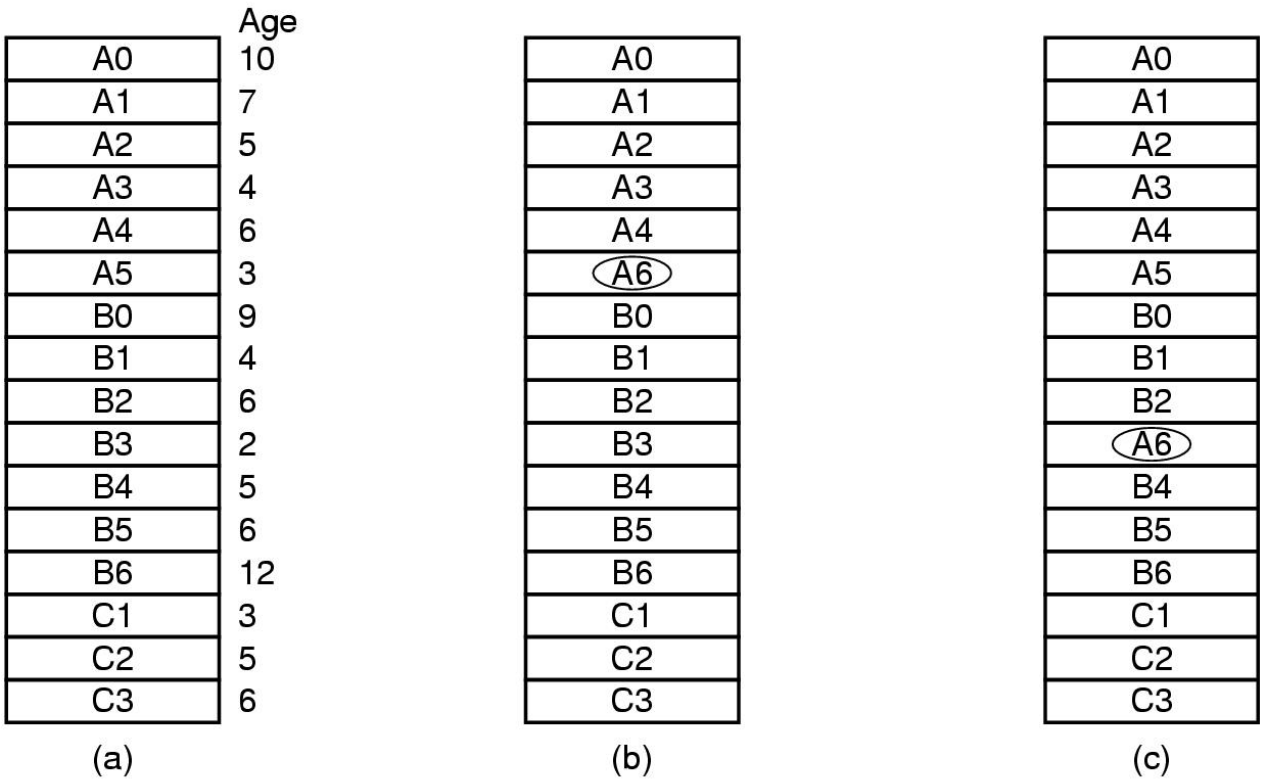
(a)



(b)

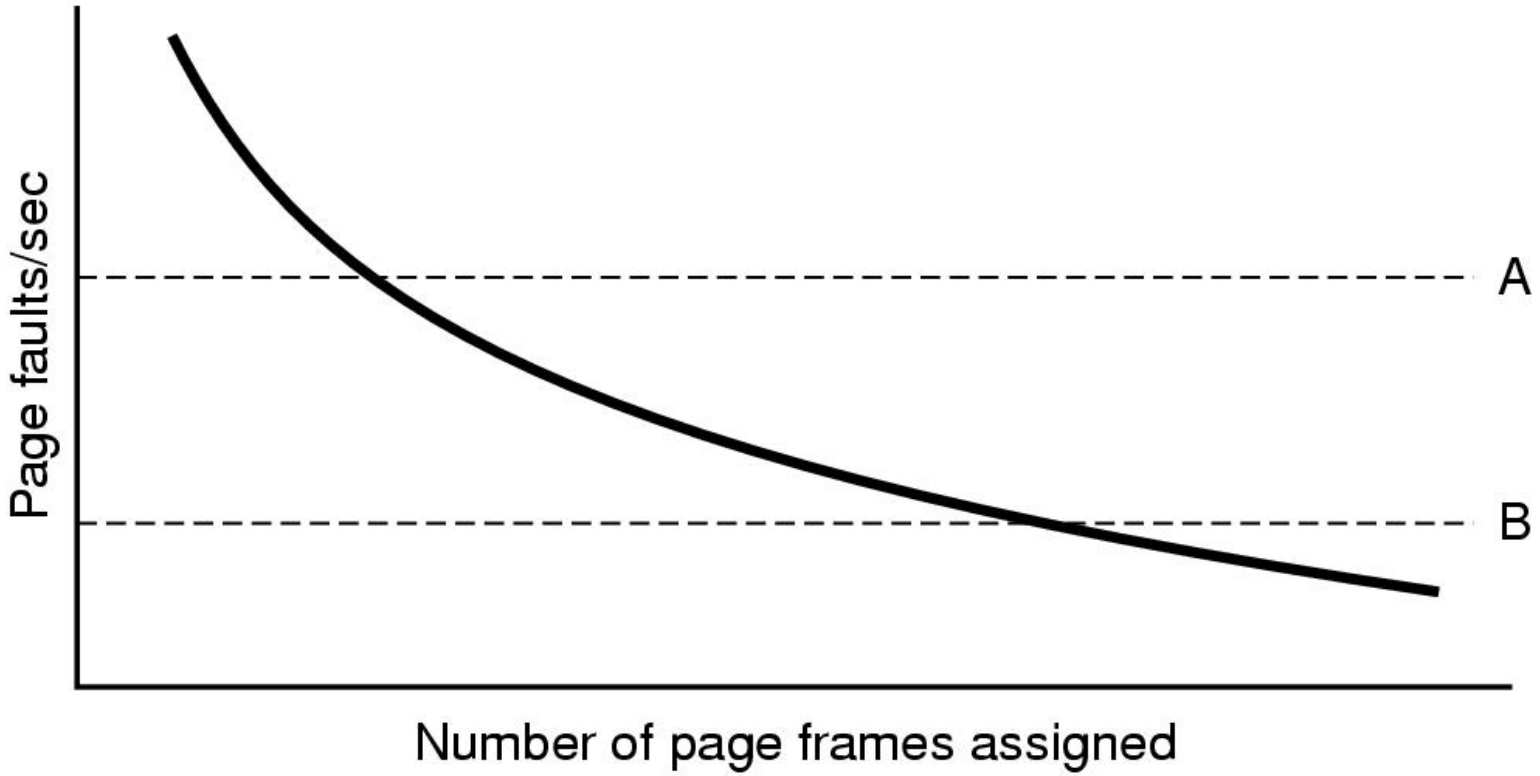
- FIFO con 3 page frames
- FIFO con 4 page frames
- I page fault sono contraddistinti da P

Allocation Policy



a) Configurazione originale
b) Rimpiazzamento locale
c) Rimpiazzamento globale

Allocation Policy (2)



La strategia ottimale è quella di mantenere la frequenze di PF in un valore intermedio tra A e B

Dimensione Pagine

- Vantaggi pagine piccole
  - Minore frammentazione interna
  - Più adattabili alle dimensioni dei programmi
  - Riduzione spazio di memoria inutilizzato
- Svantaggi
  - Grosse tabelle delle pagine
  - Più accessi a disco

Dimensione Pagine

Table 8.2 Example Page Sizes

Computer	Page Size
Atlas	512 48-bit words
Honeywell-Multics	1024 36-bit word
IBM 370/XA and 370/ESA	4 Kbytes
VAX family	512 bytes
IBM AS/400	512 bytes
DEC Alpha	8 Kbytes
MIPS	4 kbyes to 16 Mbytes
UltraSPARC	8 Kbytes to 4 Mbytes
Pentium	4 Kbytes or 4 Mbytes
PowerPc	4 Kbytes

SEGMENTAZIONE

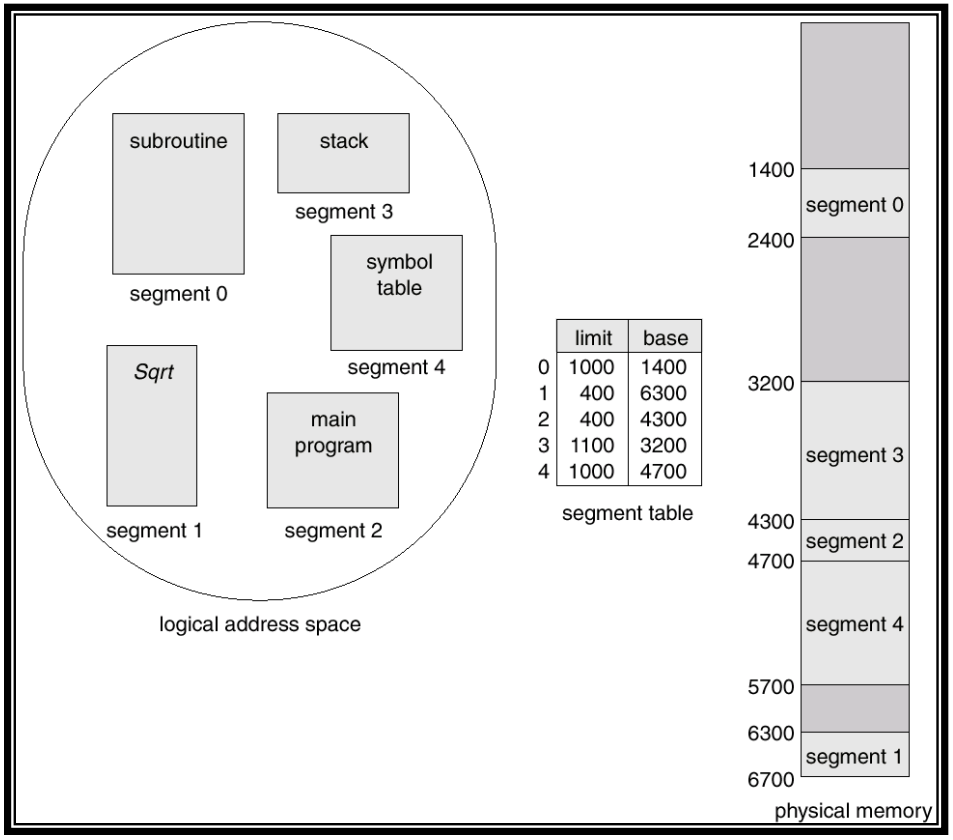
Segmentazione

- La memoria virtuale vista finora è monodimensionale
- Avere più spazi di indirizzi separati può essere vantaggioso in presenza di aree dati distinte che crescono durante l'esecuzione
  - Es. Compilatore che usa aree per
    - Testo sorgente
    - Symbol table
    - Tabella delle costanti intere e floating point
    - Albero di parsing
    - Stack per le chiamate di procedura del compilatore stesso

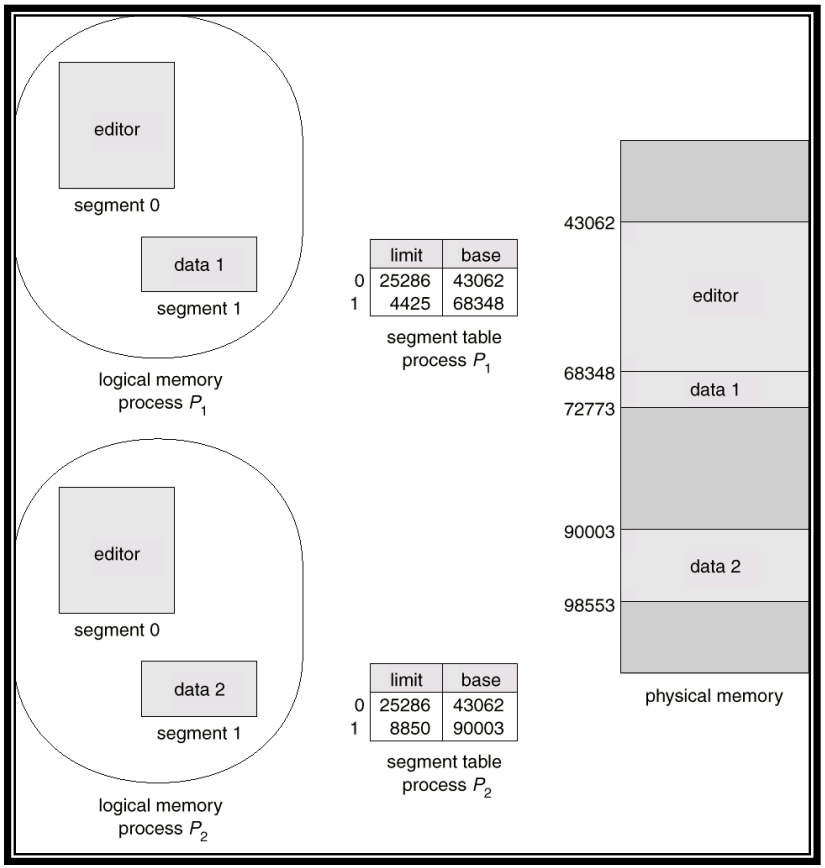
Segmentazione

- Lo spazio di indirizzamento è costituito da un insieme di spazi di indirizzi logici indipendenti detti segmenti
- I segmenti hanno dimensioni diverse e possono cambiare senza interferire gli uni con gli altri
- L'utente definisce i segmenti ma non li gestisce
- Semplifica le operazioni di linking
  - Segmento, offset per indirizzare i moduli
  - I segmenti sono indipendenti
- Facilita la condivisione di librerie
- Soffre di frammentazione esterna

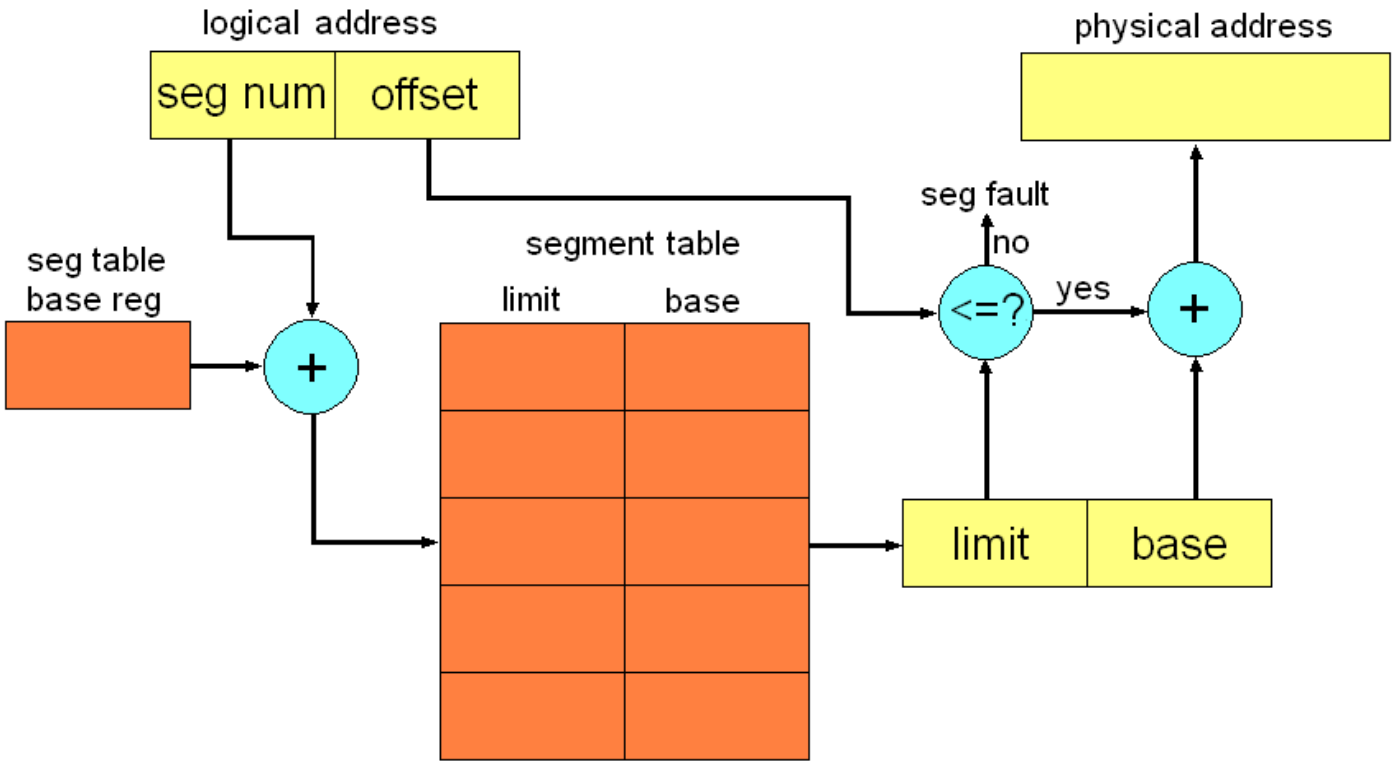
Segmentazione: esempio



Segment Sharing



Mapping indirizzo logico-fisico



Segmentazione

Consideration	Paging	Segmentation
Need the programmer be aware that this technique is being used?	No	Yes
How many linear address spaces are there?	1	Many
Can the total address space exceed the size of physical memory?	Yes	Yes
Can procedures and data be distinguished and separately protected?	No	Yes
Can tables whose size fluctuates be accommodated easily?	No	Yes
Is sharing of procedures between users facilitated?	No	Yes
Why was this technique invented?	To get a large linear address space without having to buy more physical memory	To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection

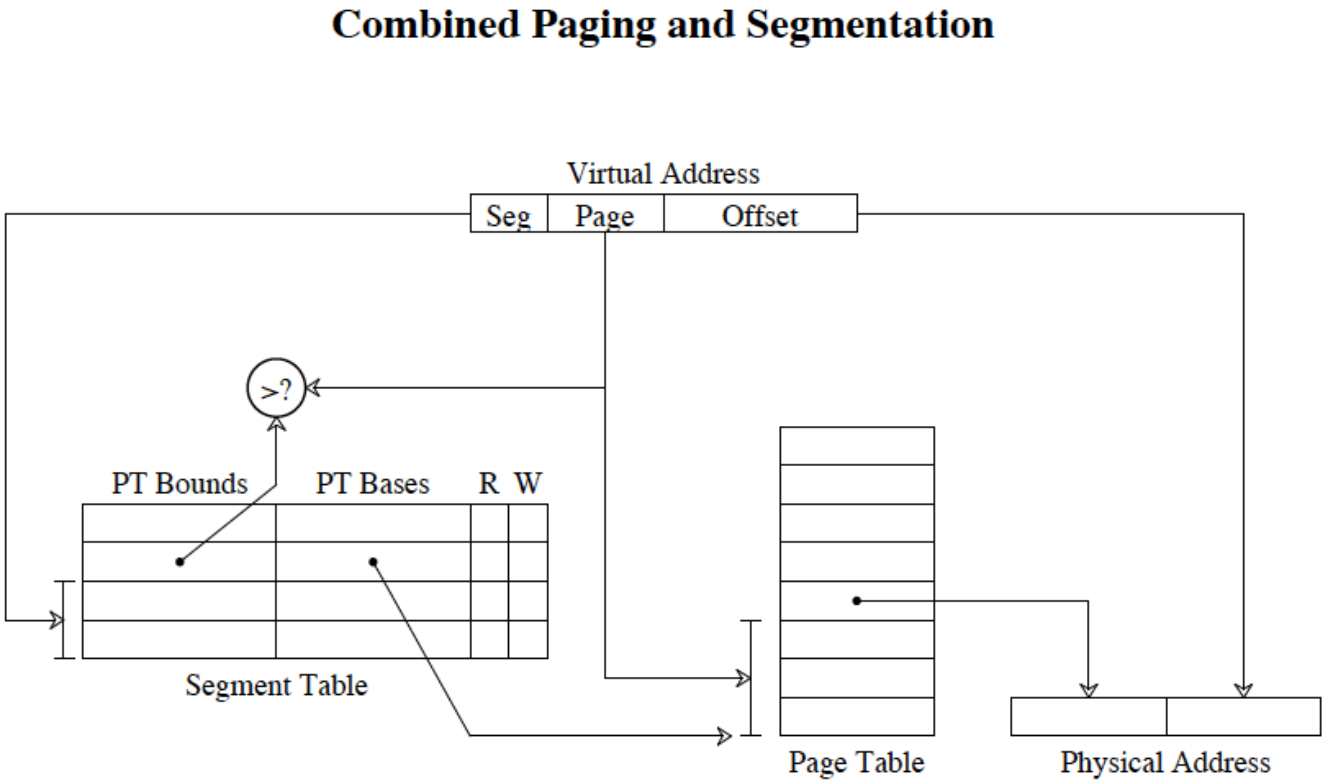
Problemi Segmentazione

- Non è trasparente al programmatore
- Presenta i problemi di frammentazione che caratterizzano gli schemi di gestione della memoria a partizioni

Segmentazione paginata

- Obiettivo: cogliere gli aspetti positivi di entrambe le soluzioni cercando di contenere gli effetti negativi
- Un sistema in cui i processi sono suddivisibili in segmenti che non devono essere necessariamente caricati interamente in memoria perché paginabili

Calcolo indirizzo



Un caso di studio: IA-32

- Ogni processo è suddiviso in almeno tre segmenti: code-segment, data-segment, e uno stack-segment i cui dati di riferimento sono presenti in appositi registri (segment register)
- Ogni segmento può avere dimensioni sino a 4GB
- Il compilatore assegna ad ogni dato/istruzione il suo spiazzamento all'interno del segmento di riferimento
- L'indirizzo logico di un elemento è costituito da:
  - L'indicatore del suo segmento di riferimento (segment selector)
  - L'indirizzo asegnatogli dal compilatore
- In una prima fase di trasformazione l'indirizzo logico viene trasformato in indirizzo lineare

Logico → Lineare

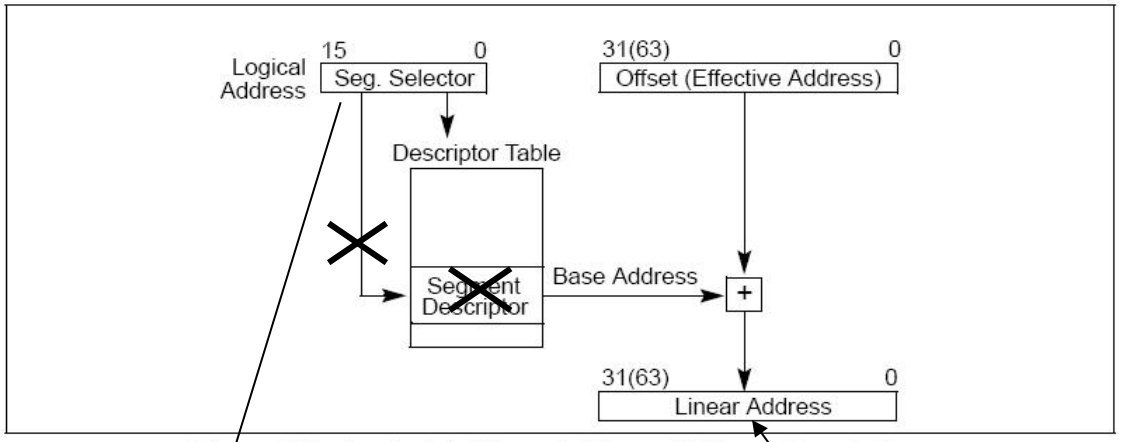


Figure 3-5. Logical Address to Linear Address Translation

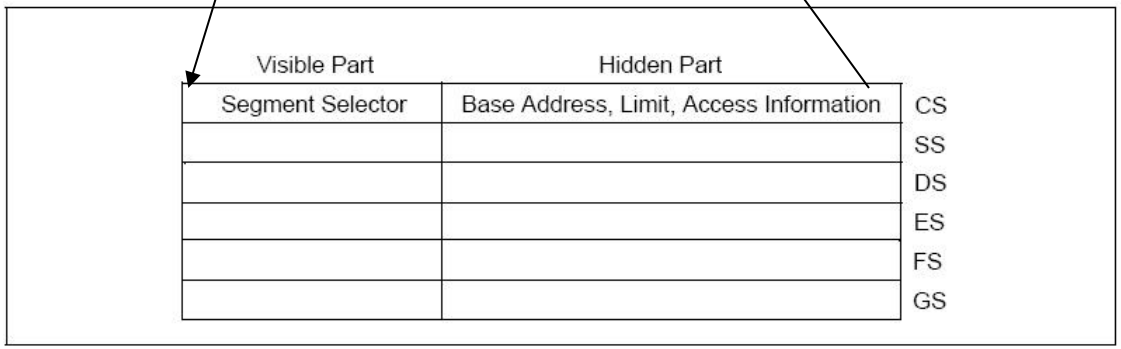
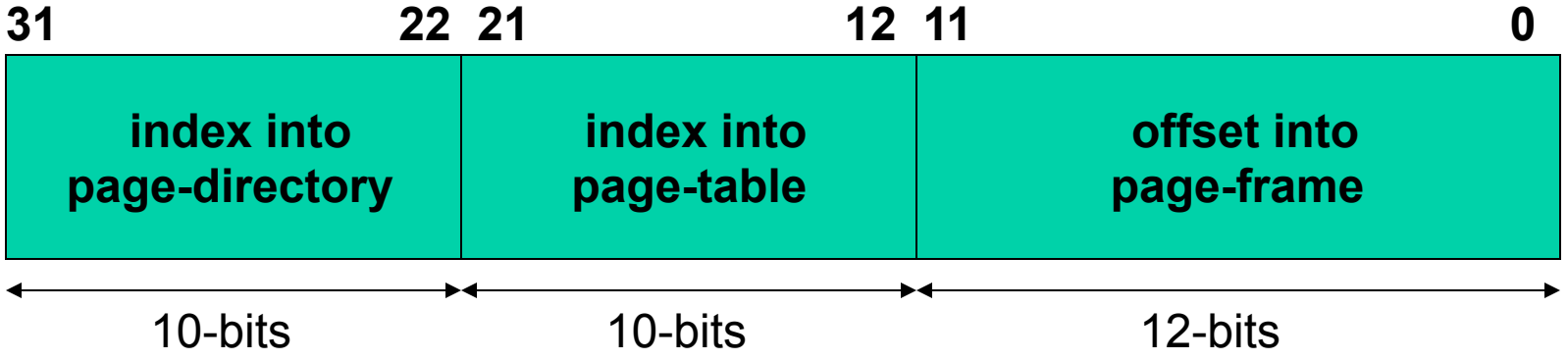


Figure 3-7. Segment Registers

Address-translation

- L'indirizzo lineare generato dalla CPU viene suddiviso dalla MMU in 3 campi

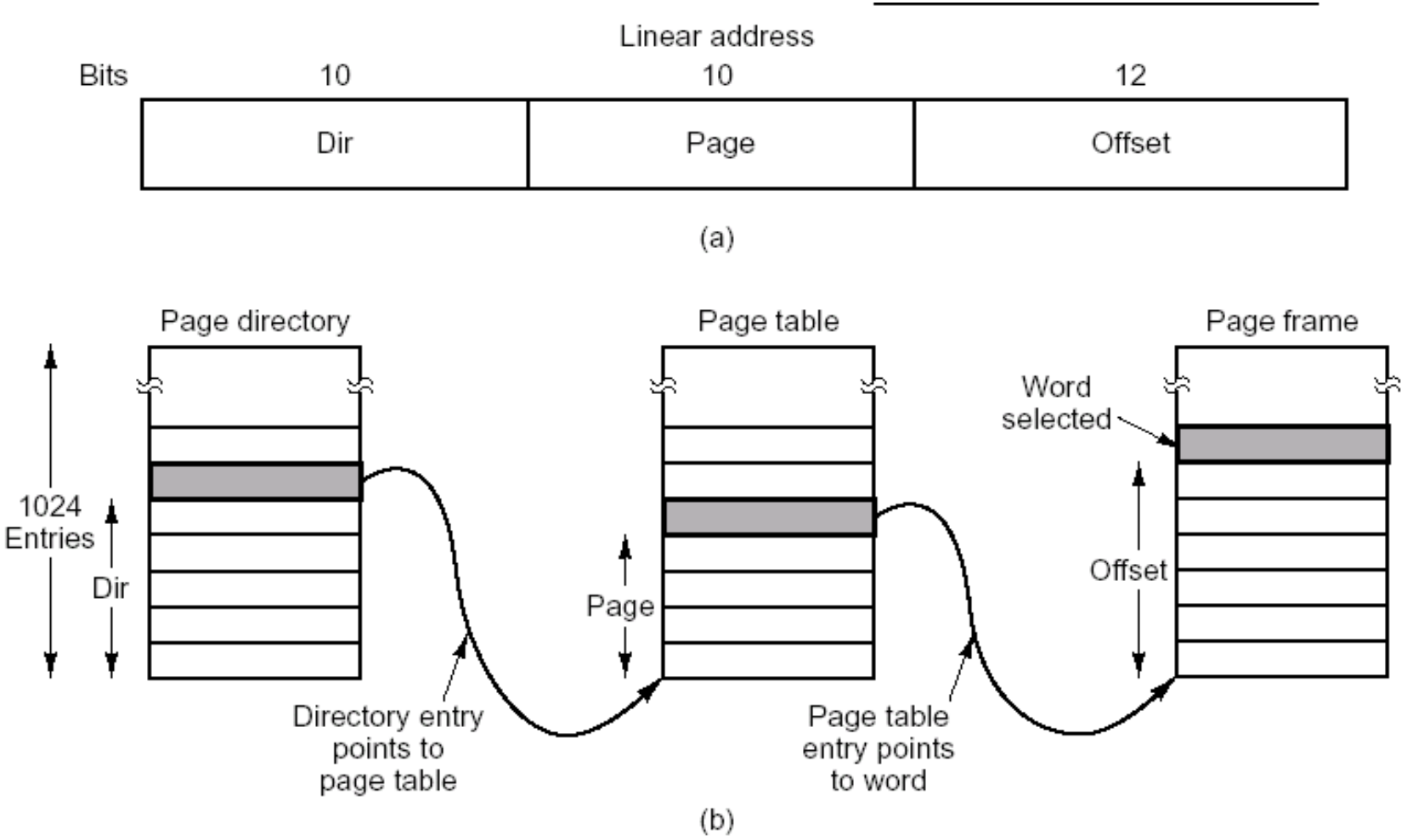


Questo campo seleziona uno dei 1024 record della Page-Directory

Questo campo seleziona uno dei 1024 record della Page-Table

Questo campo è l'offset del byte indirizzato all'interno di una pagine di 4096 byte

Paginazione Intel



Riassumendo

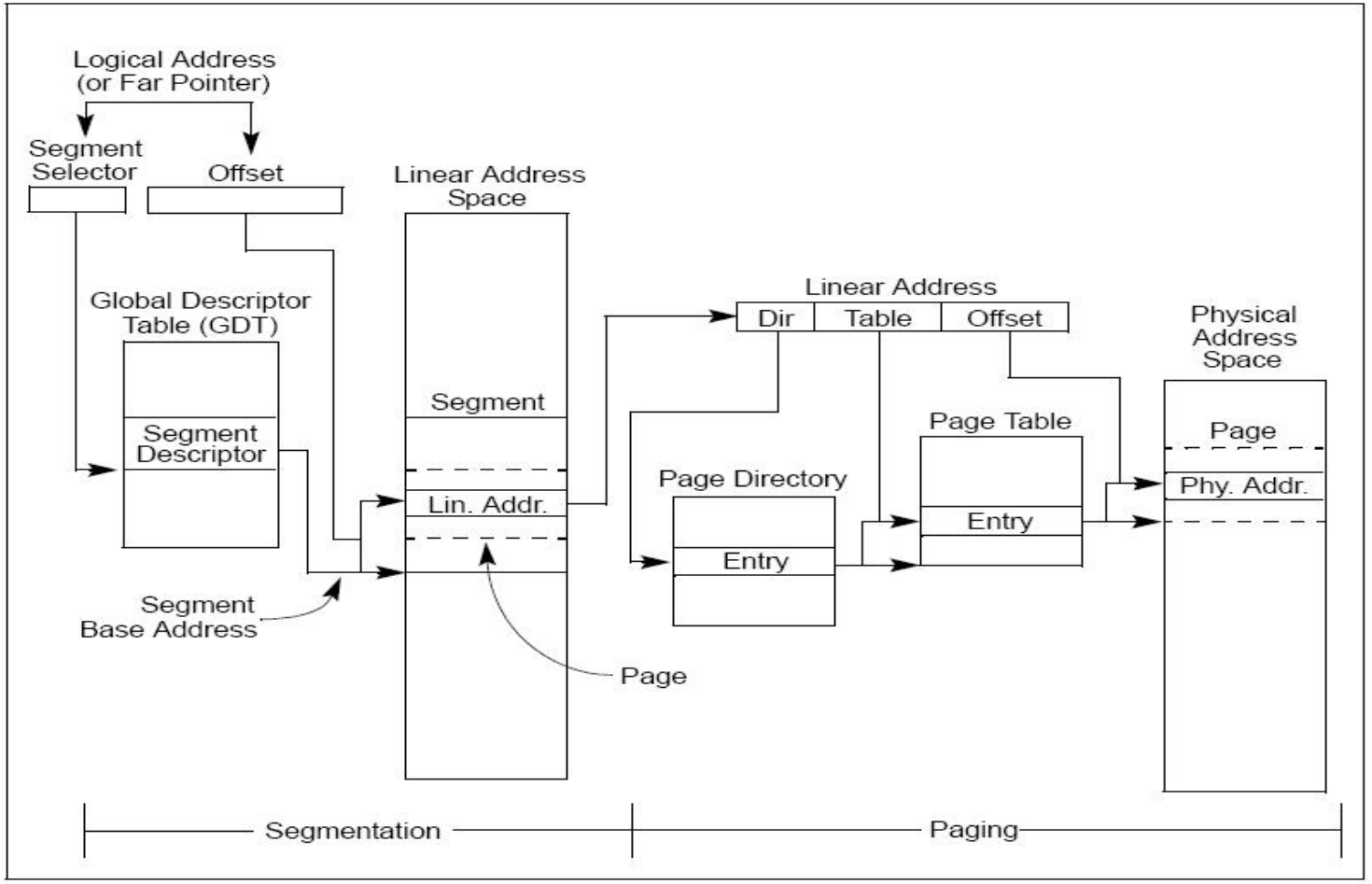


Figure 3-1. Segmentation and Paging

Physical Address Extension

- Abilita un'estensione dello spazio di indirizzamento fisico a 36 bit
- Introdotta a partire dal Pentium Pro, in cui il bus indirizzi è stato esteso da 32 a 36 linee
- Lo spazio virtuale di ciascun processo resta di 32 bit
- Abilitata settando il bit PAE in CR4
- Dimensione consentita delle pagine: 4KB, 2MB, 4MB

PAE: Strutture di supporto

- Page directory e page table restano di dimensione 4K
- Per poter esprimere correttamente indirizzi di 36 bit, gli elementi di page directory e page table sono estesi da 32 a 64 bit
- Questo significa che ogni tabella conterrà al più 2^9 elementi per un totale di 2^{18} pagine indirizzabili e quindi per uno spazio virtuale di 2^{30} bit
- Per garantire il mantenimento dello spazio virtuale a 32 bit si introduce una ulteriore tabella di 4 elementi: page-directory-pointer-table

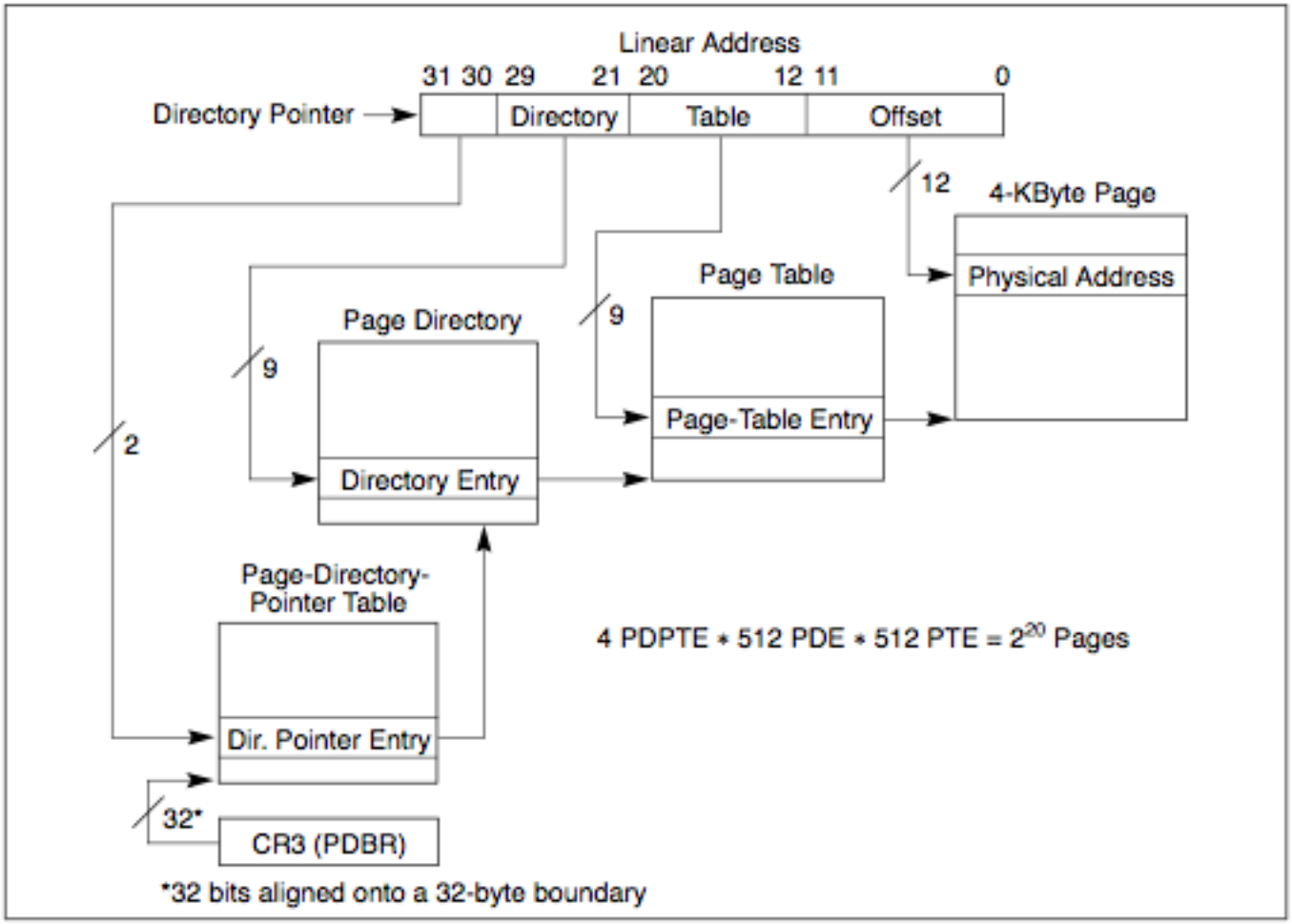


Figure 3-18. Linear Address Translation With Extended Physical Addressing Enabled (4-KByte Pages)

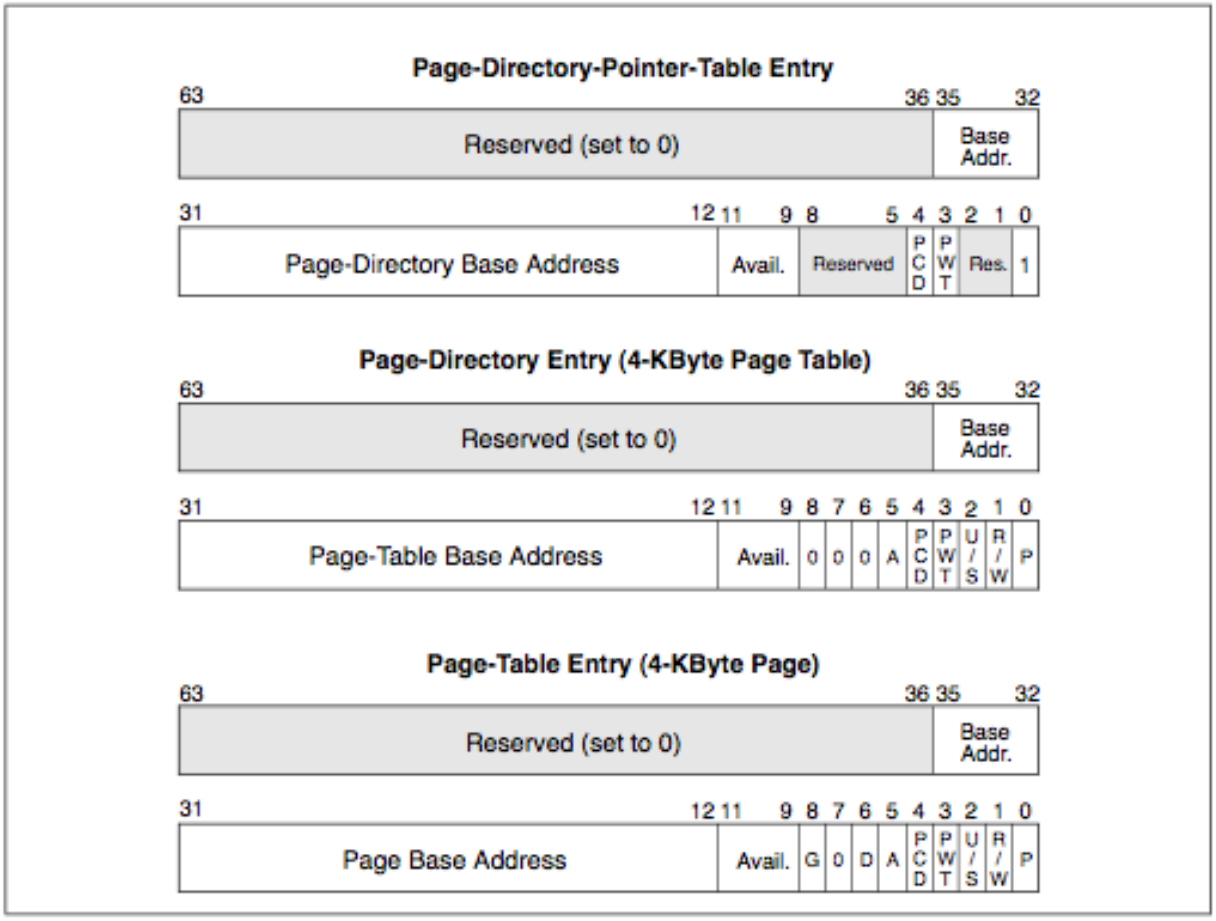


Figure 3-20. Format of Page-Directory-Pointer-Table, Page-Directory, and Page-Table Entries for 4-KByte Pages and 36-Bit Extended Physical Addresses

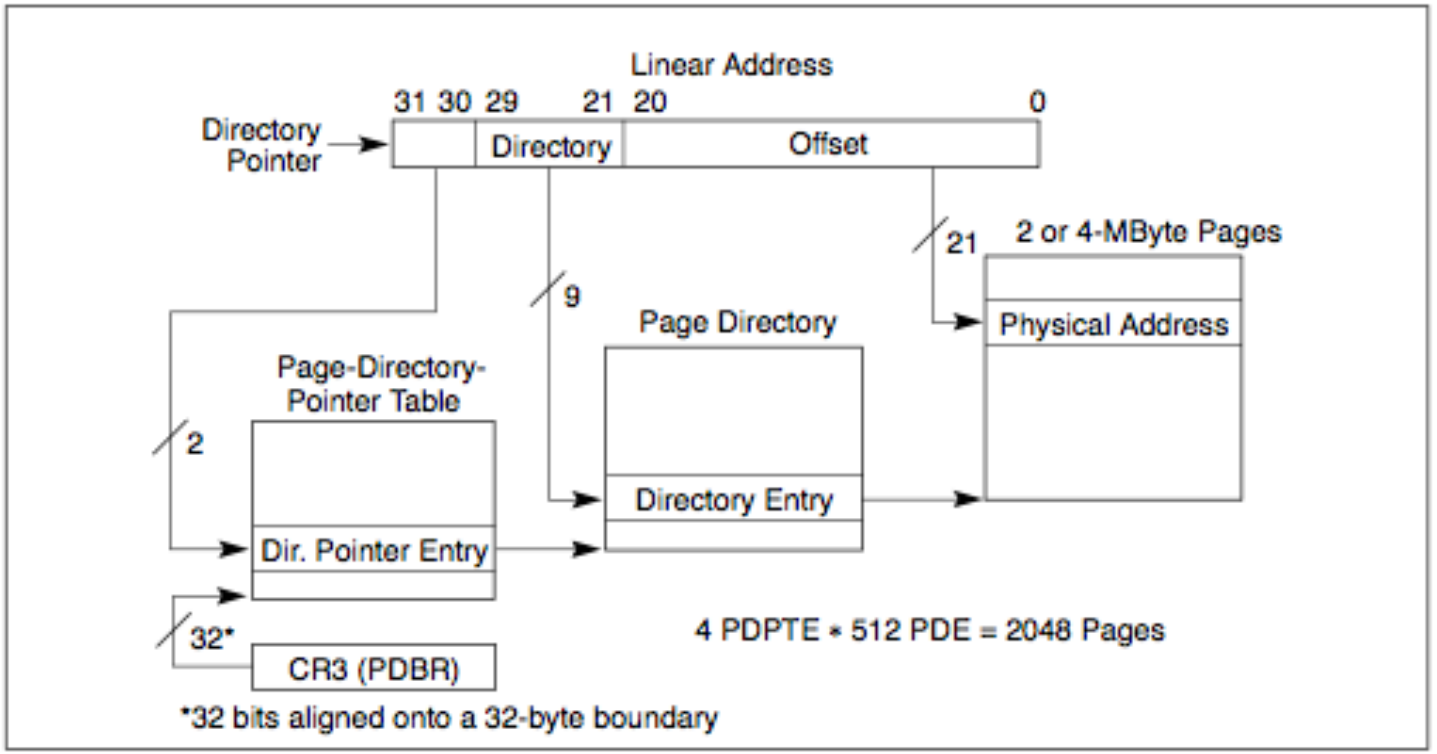


Figure 3-19. Linear Address Translation With Extended Physical Addressing Enabled (2-MByte or 4-MByte Pages)

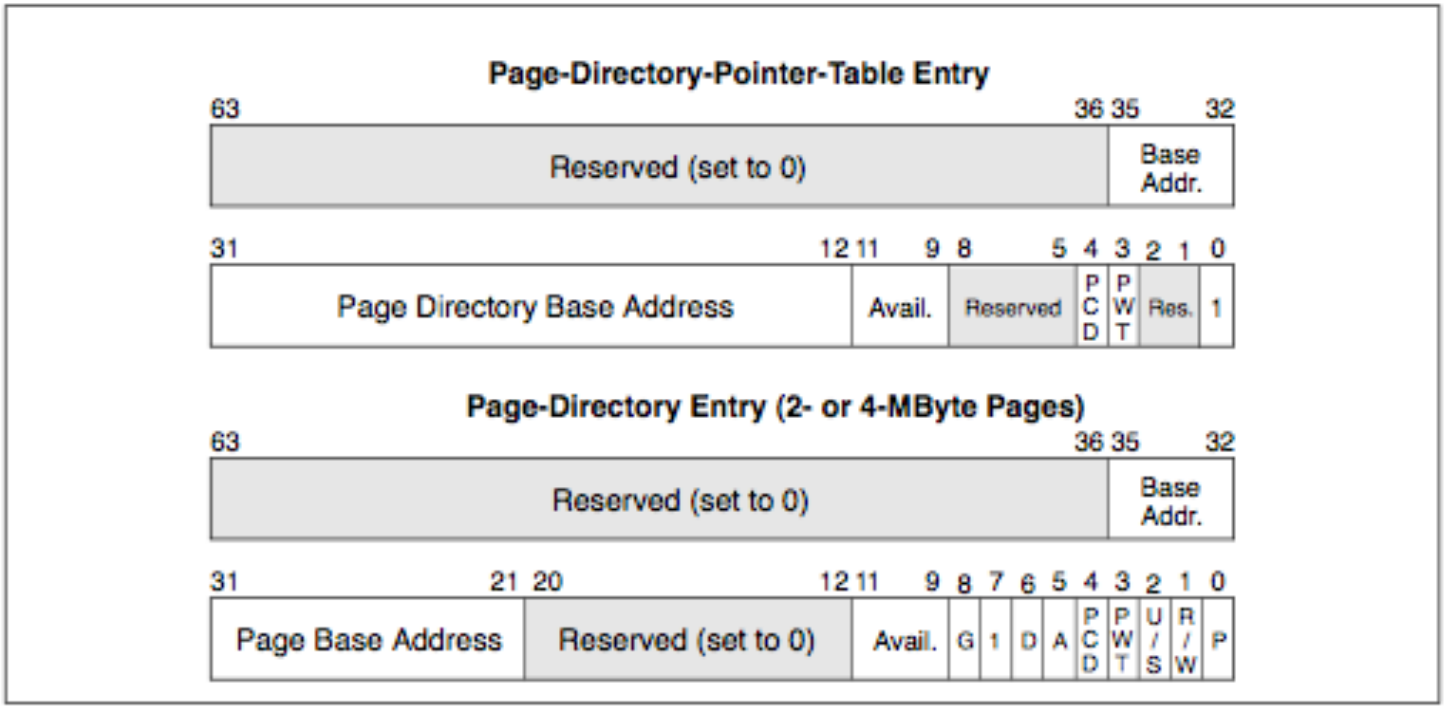


Figure 3-21. Format of Page-Directory-Pointer-Table and Page-Directory Entries for 2- or 4-MByte Pages and 36-Bit Extended Physical Addresses