

# ***Introducción a los Sistemas Operativos***

## **Administración de Memoria - III**



- ✓ Versión: Mayo 2013
- ✓ Palabras Claves: Procesos, Espacio de Direcciones, Memoria, Seguridad, Paginación, Memoria Virtual, Tablas de Páginas

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz (Operating Systems Concepts). También se incluyen diapositivas cedidas por Microsoft S.A.



# Hasta ahora

- ☑ Con paginación vimos que el espacio de direcciones de un proceso no necesariamente debe estar “contiguo” en la memoria para poder ejecutarse
  - ✓ El HW traduce direcciones lógicas a direcciones físicas utilizando las tablas de páginas que el SO administra



# Motivación para MV

- ✓ Podemos pensar también que, no todo el espacio de direcciones del proceso se necesita en todo momento:
  - ✓ Rutinas o Librerías que se ejecutan una única vez (o nunca)
  - ✓ Partes del programa que no vuelven a ejecutarse
  - ✓ Regiones de memoria alocadas dinámicamente y luego liberadas
  - ✓ Etc.



# Como se puede trabajar...

- ✓ El SO puede traer a memoria las “piezas” de un proceso a medida que éste las necesita.
- ✓ Definiremos como “**Conjunto Residente**” a la porción del espacio de direcciones del proceso que se encuentra en memoria.
  - ✓ Alguna bibliografía lo llama “Working Set”
- ✓ Con el apoyo del HW:
  - ✓ Se detecta cuando se necesita una porción del proceso que no está en su Conjunto Residente



# Ventajas

- ☑ Más procesos pueden ser mantenidos en memoria.
  - ✓ Sólo son cargadas algunas secciones de cada proceso.
  - ✓ Con más procesos en memoria principal es más probable que existan más procesos Ready
- ☑ Un proceso puede ser mas grande que la memoria Principal
  - ✓ El usuario no se debe preocupar por el tamaño de sus programas
  - ✓ La limitación la impone el HW y el bus de direcciones.



# *¿Que se necesita para MV?*

- ✓ El hardware debe soportar paginación por demanda (y/o segmentación)
- ✓ Un dispositivo de memoria secundaria (disco) que dé el apoyo para almacenar las secciones del proceso que no están en Memoria Principal (área de intercambio)
- ✓ El SO debe ser capaz de manejar el movimiento de las páginas (o segmentos) entre la memoria principal y la secundaria.



# MV con Paginación

- ✓ Cada proceso tiene su tabla de páginas
- ✓ Cada entrada en la tabla referencia al frame o marco en el que se encuentra la página en la memoria principal
- ✓ Cada entrada en la tabla de páginas tiene bits de control (entre otros):
  - ✓ Bit V: Indica si la página está en memoria
  - ✓ Bit M: Indica si la página fue modificada. Si se modificó, en algún momento, se deben reflejar los cambios en Memoria Secundaria

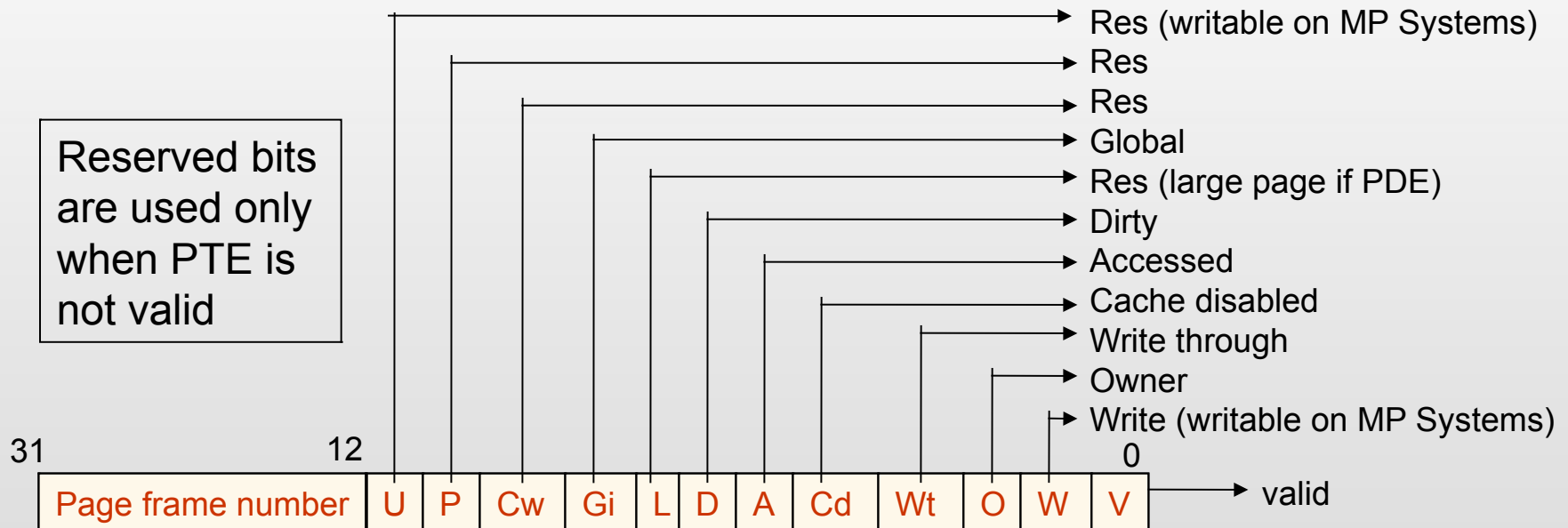




# Entrada en la Tabla de páginas de x86

Una entrada válida tiene:

- ✓ Bit V = 1
- ✓ Page Frame Number (PFN) - Marco de memoria asociado
- ✓ Flags que describen su estado y protección



# Entrada en la Tabla de páginas de x86

Name of Bit

Meaning on x86

Accessed

Page has been read

Cache disabled

Disables caching for that page

Dirty

Page has been written to

Global

Translation applies to all processes  
(a translation buffer flush won't affect this PTE)

Large page

Indicates that PDE maps a 4MB page (used to map kernel)

Owner

Indicates whether user-mode code can access the page or whether the page is limited to kernel mode access

Valid

Indicates whether translation maps to page in phys. Mem.

Write through

Disables caching of writes; immediate flush to disk

Write

Uniproc: Indicates whether page is read/write or read-only;  
Multiproc: Ind. whether page is writeable/write bit in res. bit



# Fallo de páginas (Page Fault)

- ✓ Ocorre cuando el proceso intenta usar una dirección que está en una página que no se encuentra en la memoria principal. Bit  $V=0$ 
  - ✓ La página no se encuentra en su conjunto residente
  - ✓ El bit  $V$  es controlado por el HW
- ✓ El HW detecta la situación y genera un trap al S.O.
- ✓ El S.O. Podrá colocar al proceso en estado de “Blocked” (espera) mientras gestiona que la página que se necesite se cargue.



# Fallo de páginas (cont.)

- ✓ El S.O. busca un “Frame o Marco Libre” en la memoria y genera una operación de E/S al disco para copiar en dicho Frame la página del proceso que se necesita utilizar.
- ✓ El SO puede asignarle la CPU a otro proceso mientras se completa la E/S
  - ✓ La E/S se realizará y avisará mediante interrupción su finalización.

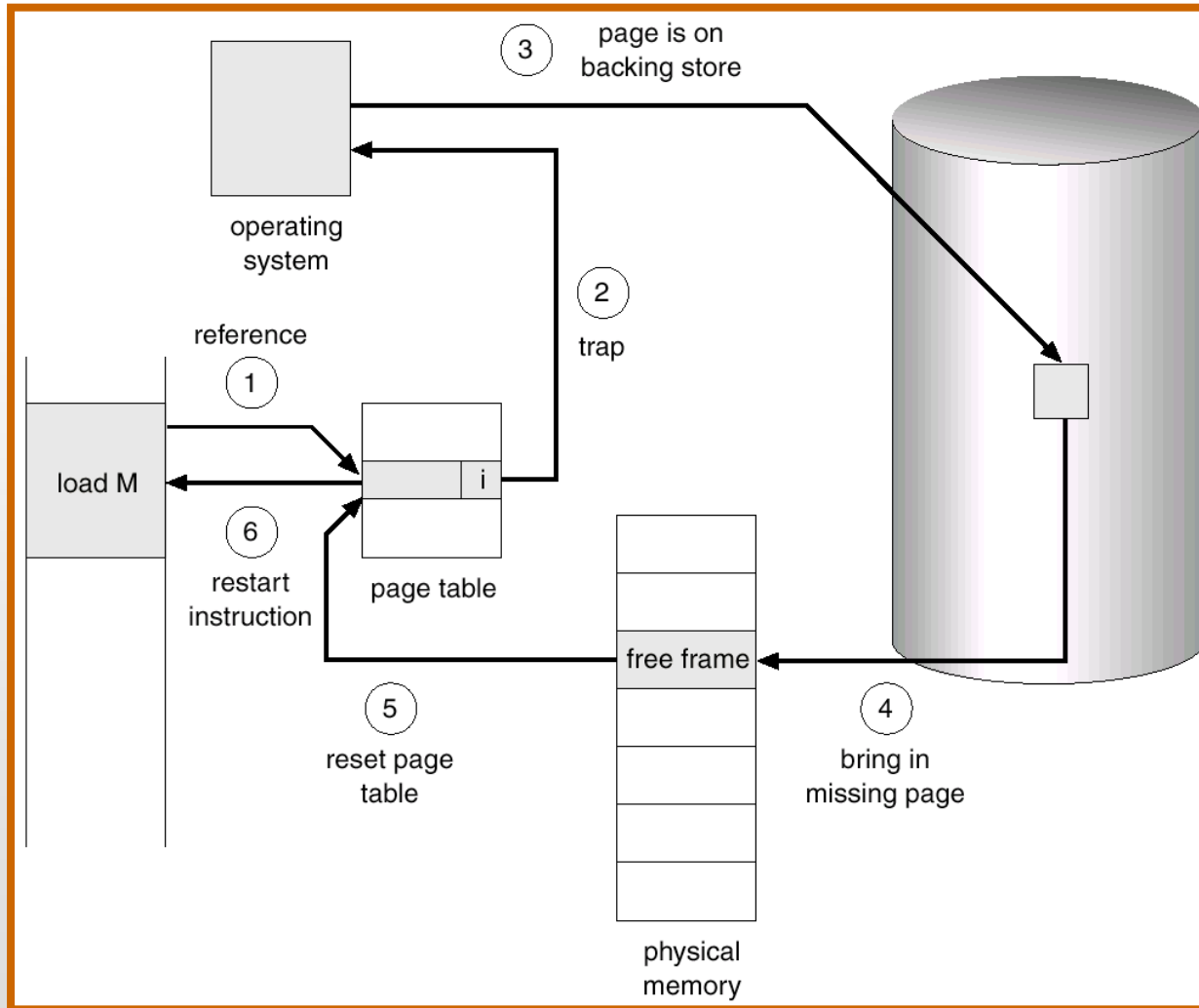


# Fallo de páginas (cont.)

- ☑ Cuando la operación de E/S finaliza, se notifica al SO y este:
  - ✓ Actualiza la tabla de páginas del proceso
    - ◆ Coloca el Bit V en 1 en la página en cuestión
    - ◆ Coloca la dirección base del Frame donde se colocó la página
  - ✓ El proceso que generó el Fallo de Página vuelve a estado de Ready (listo)
  - ✓ Cuando el proceso se ejecute, se volverá a ejecutar la instrucción que antes generó el fallo de página



# Fallo de páginas (cont.)



# Performance

☑ Tasa de Page Faults  $0 \leq p \leq 1$

✓ Si  $p = 0$  no hay page faults

✓ Si  $p = 1$ , cada a memoria genera un page fault

☑ Effective Access Time (EAT)

$$\begin{aligned} \text{EAT} = & (1 - p) \times \text{memory access} \\ & + p \times (\text{page\_fault\_overhead} + \\ & \quad [\text{swap\_page\_out}] + \\ & \quad \text{swap\_page\_in} + \\ & \quad \text{restart\_overhead}) \end{aligned}$$



# Tabla de Páginas

- ✓ Cada proceso tiene su tabla de páginas
- ✓ El tamaño de la tabla de páginas depende del espacio de direcciones del proceso.
- ✓ Puede alcanzar un tamaño considerable





# Tabla de Páginas (cont.)

## ✓ Ejemplo

20 bits

12 bits

- ✓ Asumir PTE de 4 bytes
- ✓ Ej. Página de 4KB, espacio de direcciones: 32bits
  - ♦ Cantidad de PTEs que tiene un proceso =  $2^{20}$
  - ♦ Cantidad de PTEs que entran en un marco:  $4\text{KB}/4\text{B} = 2^{10}$
  - ♦ Cantidad de marcos necesarios para todas las PTEs de la tabla de páginas de un proceso =  $2^{20}/2^{10} = 2^{10}$
  - ♦ Tamaño de la tabla de páginas del proceso =  $2^{10} * 4\text{KB} = 4\text{MB}$  por proceso
- ✓ Ej. Página de 4KB, espacio de direcciones: 64 bits
  - ♦ Cantidad de PTEs =  $2^{64}/2^{12} = 2^{52}$
  - ♦ Cantidad de marcos necesarios para todas las PTEs del proceso =  $2^{52}/2^{10} = 2^{42}$
  - ♦ Tamaño tabla =  $2^{42} * 4\text{KB} = \textbf{Más de 16.000.000GB por proceso!!!}$



# Tabla de páginas (cont.)

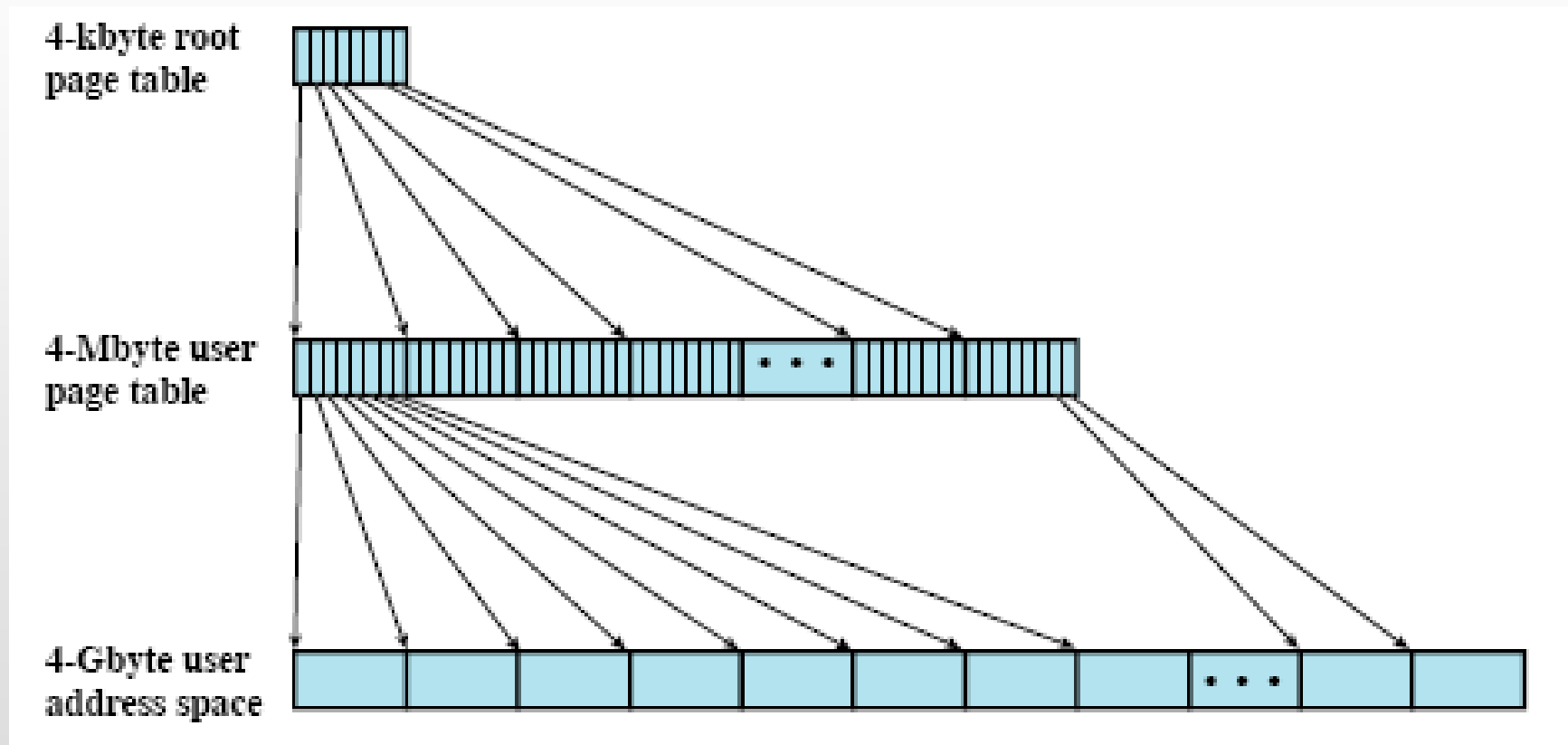
## ☑ Formas de organizar:

- ✓ Tabla de 1 nivel: Tabla única lineal
- ✓ Tabla de 2 niveles (o más, multinivel)
- ✓ Tabla invertida: Hashing

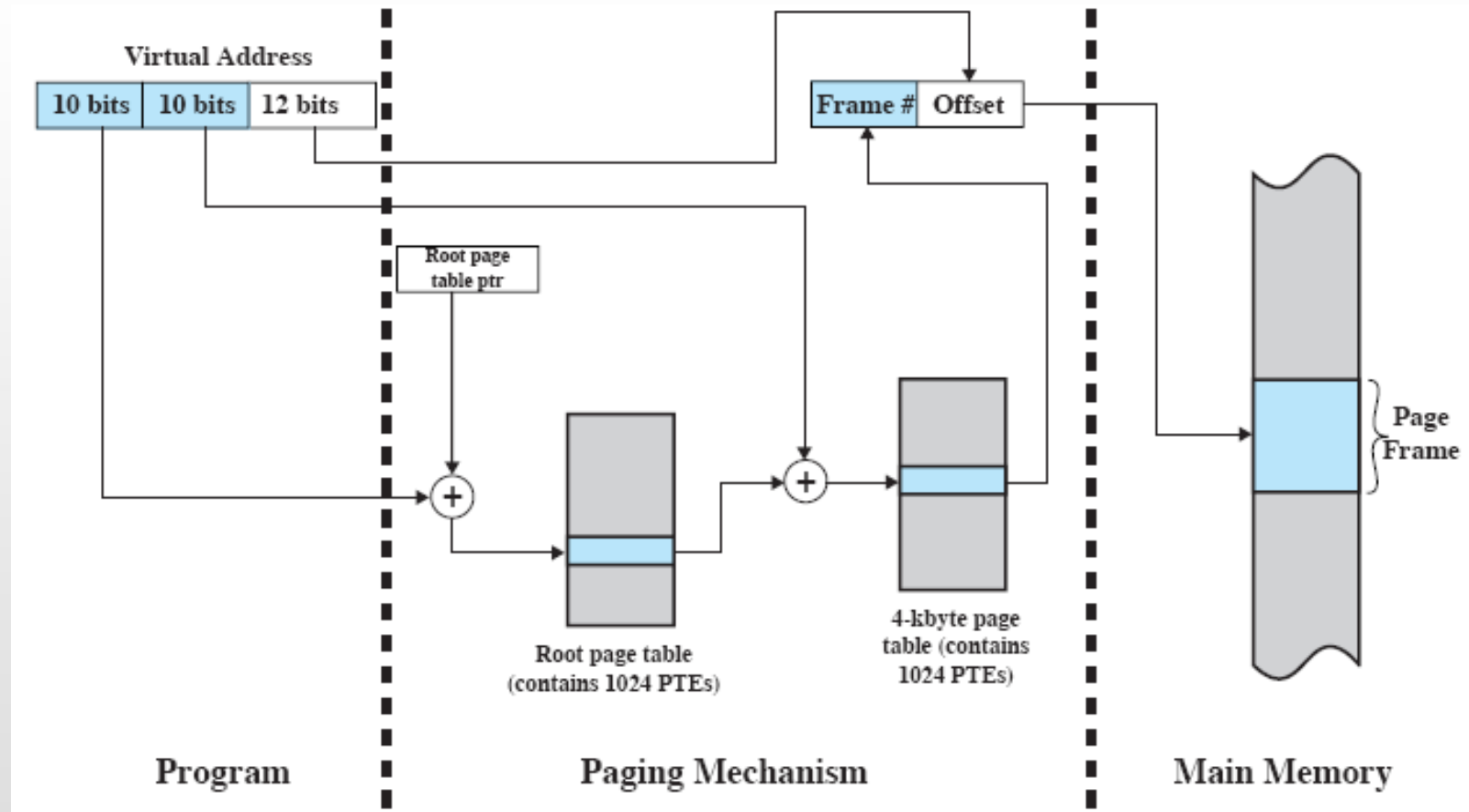
## ☑ La forma de organizarla depende del HW subyacente



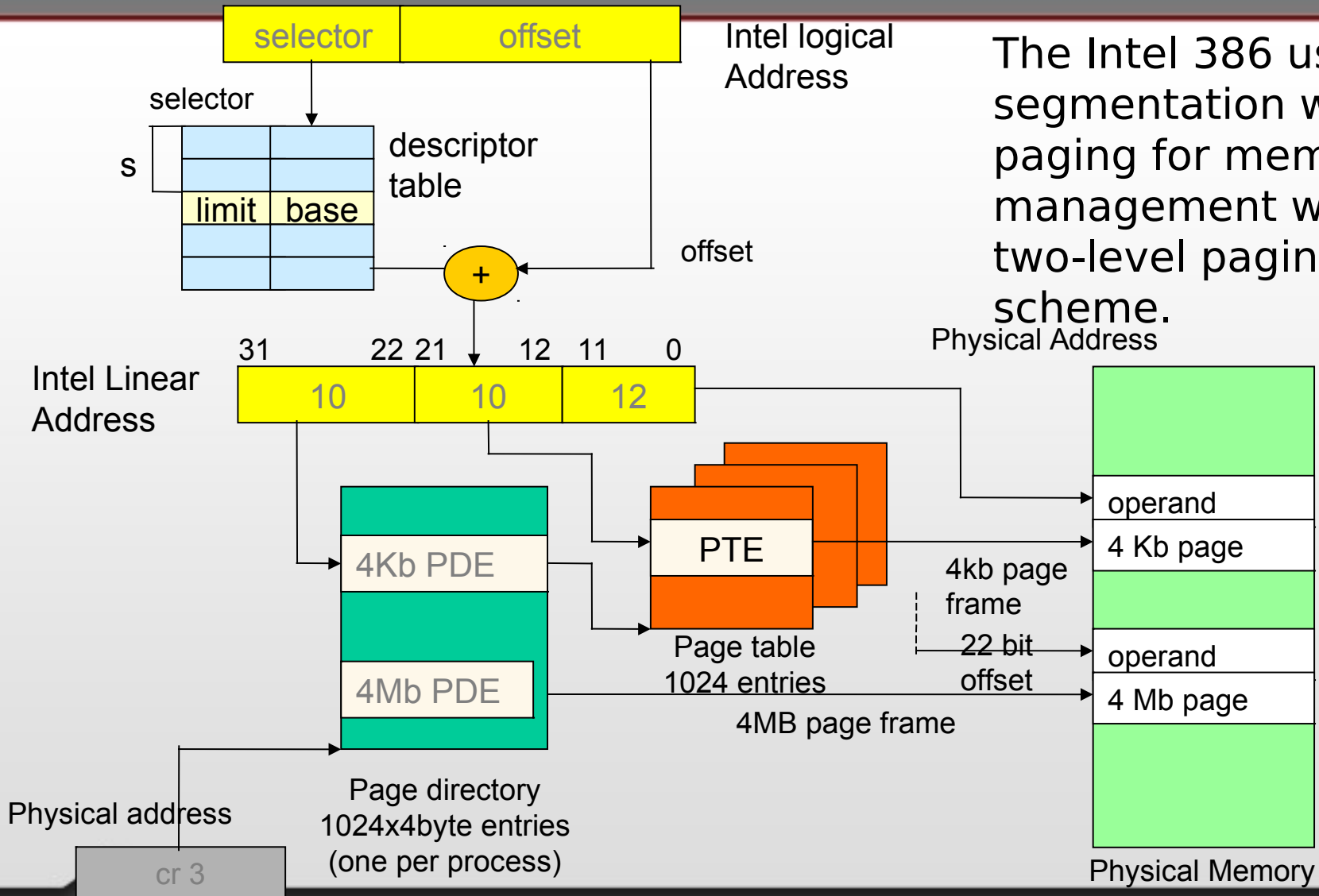
# Tabla de páginas - Tabla de 2 niveles



# Tabla de Páginas - Tabla de 2 niveles



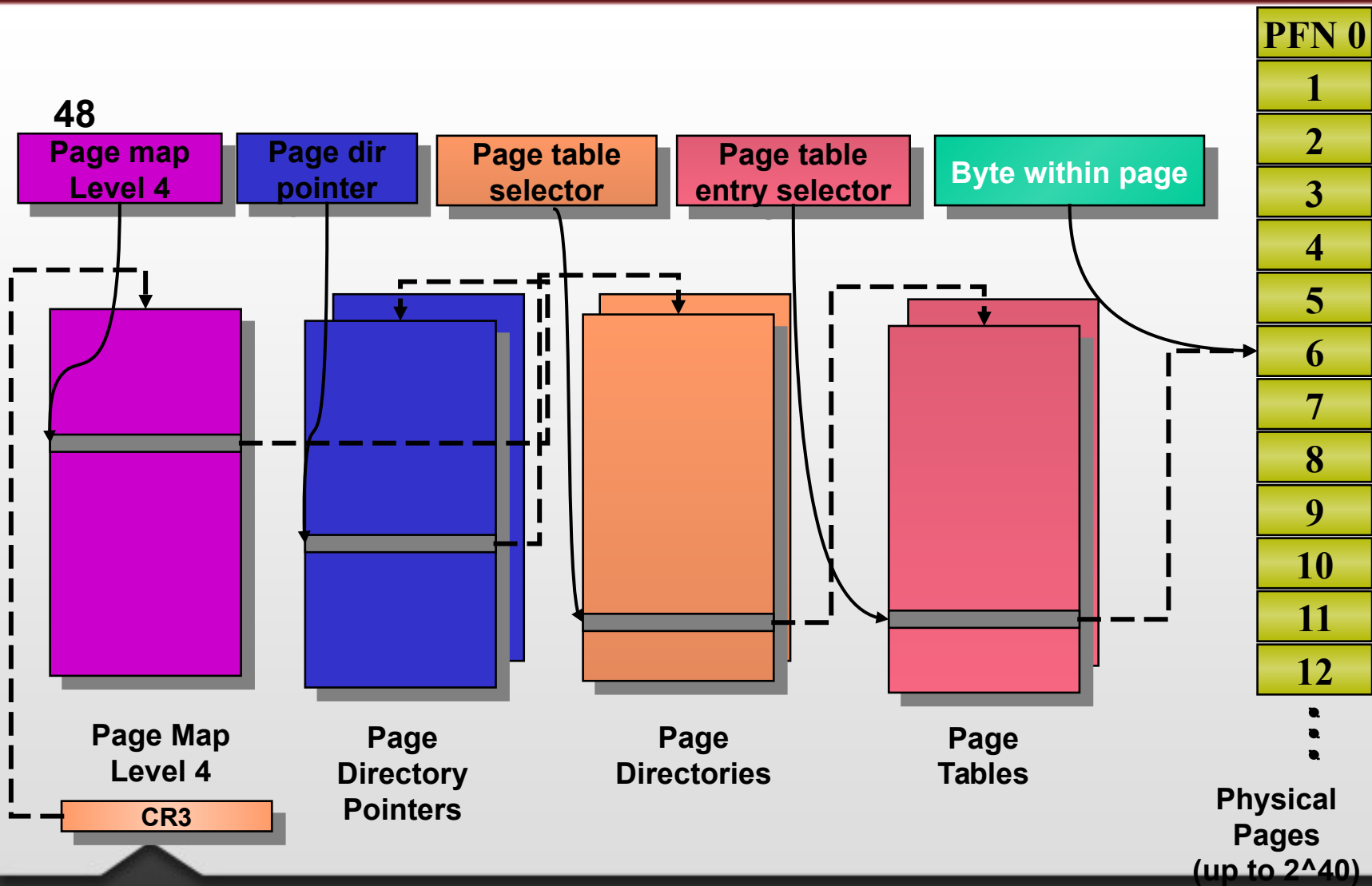
# Intel 30386



The Intel 386 uses segmentation with paging for memory management with a two-level paging scheme.



# Tablas de Páginas - x64



# Tabla de Paginas (cont.) - Tabla invertida

- ✓ Utilizada en Arquitecturas donde el espacio de direcciones es muy grande
  - ✓ Las tablas de paginas ocupan muchos niveles y la traducción es costosa
- ✓ Hay una entrada por cada frame. Hay una sola tabla para todo el sistema
- ✓ Usada en PowerPC, UltraSPARC, y IA-64
- ✓ El número de página es transformado en un valor de HASH
- ✓ El HASH se usa como índice de la tabla invertida para encontrar el marco asociado



# Tabla de Paginas (cont.) - Tabla invertida

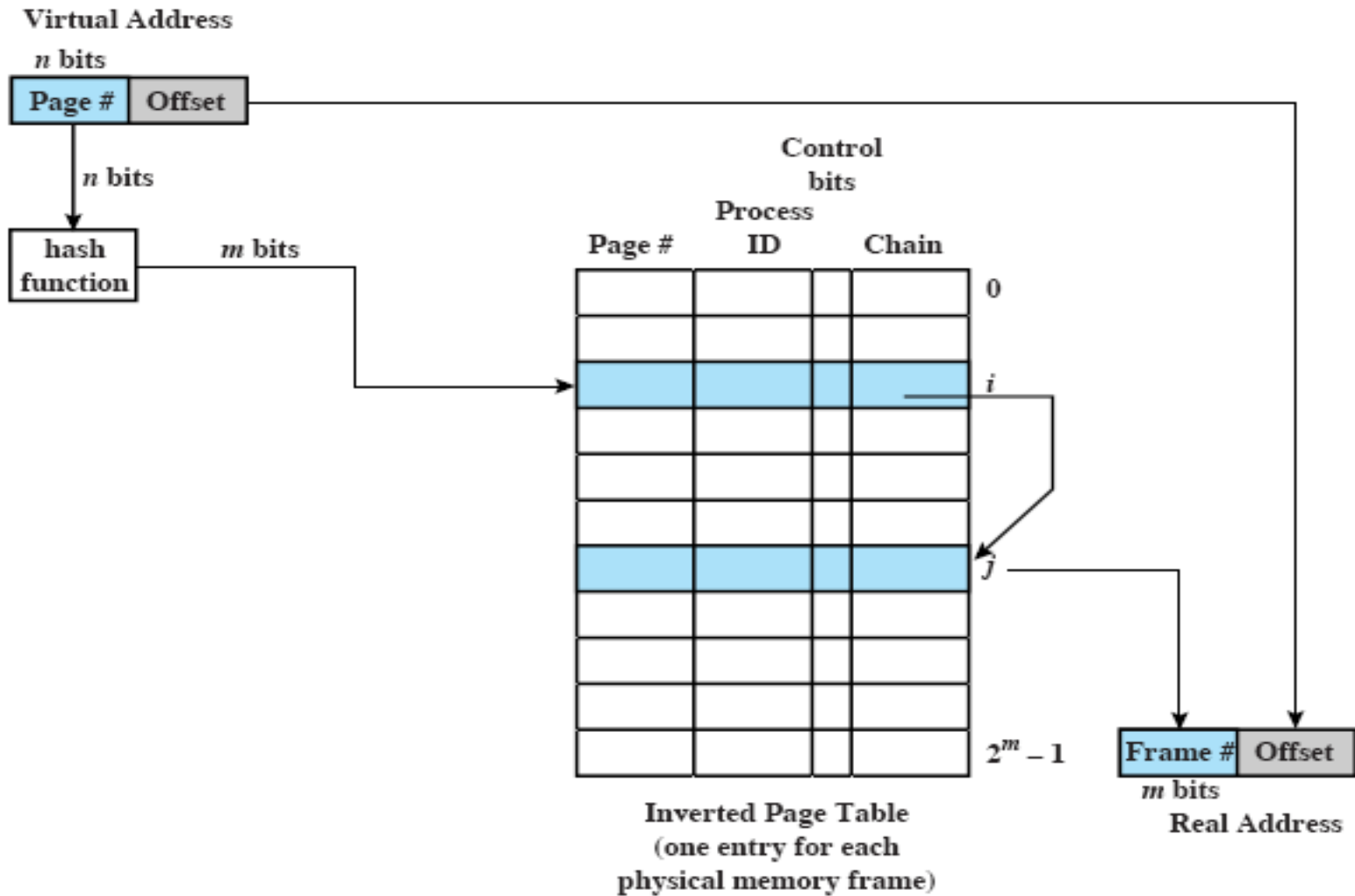
✓ Sólo se mantienen los PTEs de páginas presentes en memoria física

- ✓ Tabla invertida organizada como tabla hash en memoria principal
  - ♦ Se busca indexadamente por número de página virtual
  - ♦ Si está presente en tabla, se extrae el marco de página y sus protecciones
  - ♦ Si no está presente en tabla, corresponde a un fallo de página





## Tabla de Paginas (cont.) - Tabla invertida



# Tamaño de la Pagina

## ✓ Pequeño

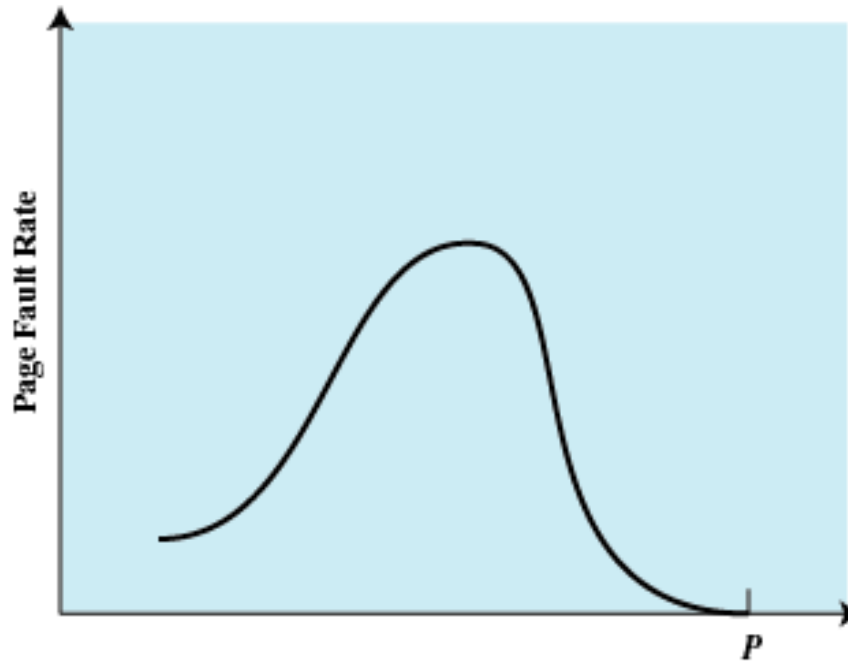
- ✓ Menor Fragmentación Interna.
- ✓ Más paginas requeridas por proceso → Tablas de páginas mas grandes.
- ✓ Más paginas pueden residir en memoria

## ✓ Grande

- ✓ Mayor Fragmentación interna
- ✓ La memoria secundaria esta diseñada para transferir grandes bloques de datos más eficientemente → Mas rápido mover páginas hacia la memoria principal.



# Tamaño de la Pagina (cont)



(a) Page Size

$P$  = size of entire process

$W$  = working set size

$N$  = total number of pages in process



# Tamaño de la Pagina (cont)

**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 kbytes to 16 Mbytes
UltraSPARC	8 Kbytes to 4 Mbytes
Pentium	4 Kbytes or 4 Mbytes
PowerPc	4 Kbytes
Itanium	4 Kbytes to 256 Mbytes



# Translation Lookaside Buffer 1

- ☑ Cada referencia en el espacio virtual puede causar 2 (o más) accesos a la memoria física.
  - ✓ Uno (o más) para obtener la entrada en tabla de paginas
  - ✓ Uno para obtener los datos
- ☑ Para solucionar este problema, una memoria cache de alta velocidad es usada para almacenar entradas de páginas
  - ✓ TLB



# Translation Lookaside Buffer 2

- ✓ Contiene las entradas de la tabla de páginas que fueron usadas mas recientemente.
- ✓ Dada una dirección virtual, el procesador examina la TLB
- ✓ Si la entrada de la tabla de paginas se encuentra en la TLB (hit), es obtenido el frame y armada la dirección física



# Translation Lookaside Buffer 3

- ✓ Si la entrada no es encontrada en la TLB (miss), el número de página es usado como índice en la tabla de paginas del proceso.
- ✓ Se controla si la pagina está en la memoria
  - ✓ Si no está, se genera un Page Fault
- ✓ La TLB es actualizada para incluir la nueva entrada



# Translation Lookaside Buffer 4

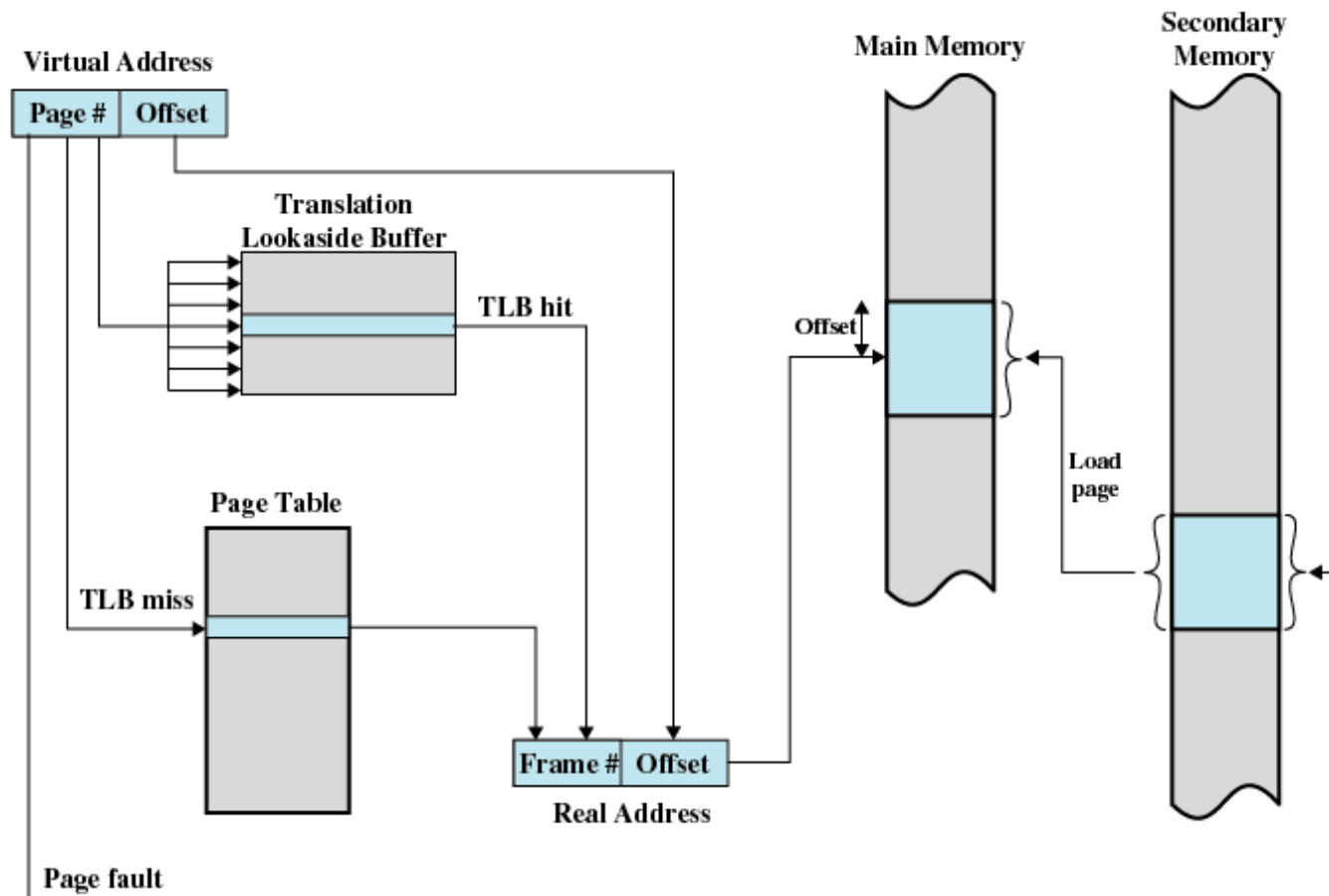


Figure 8.7 Use of a Translation Lookaside Buffer





# Translation Lookaside Buffer 5

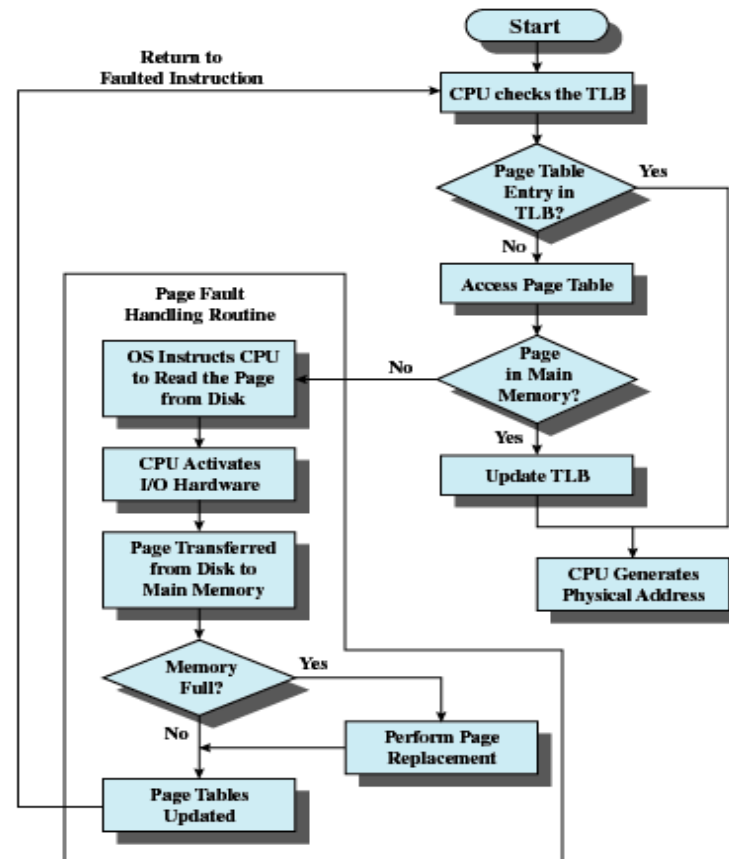


Figure 8.8 Operation of Paging and Translation Lookaside Buffer (TLB) [FURH87]



# Translation Lookaside Buffer 6

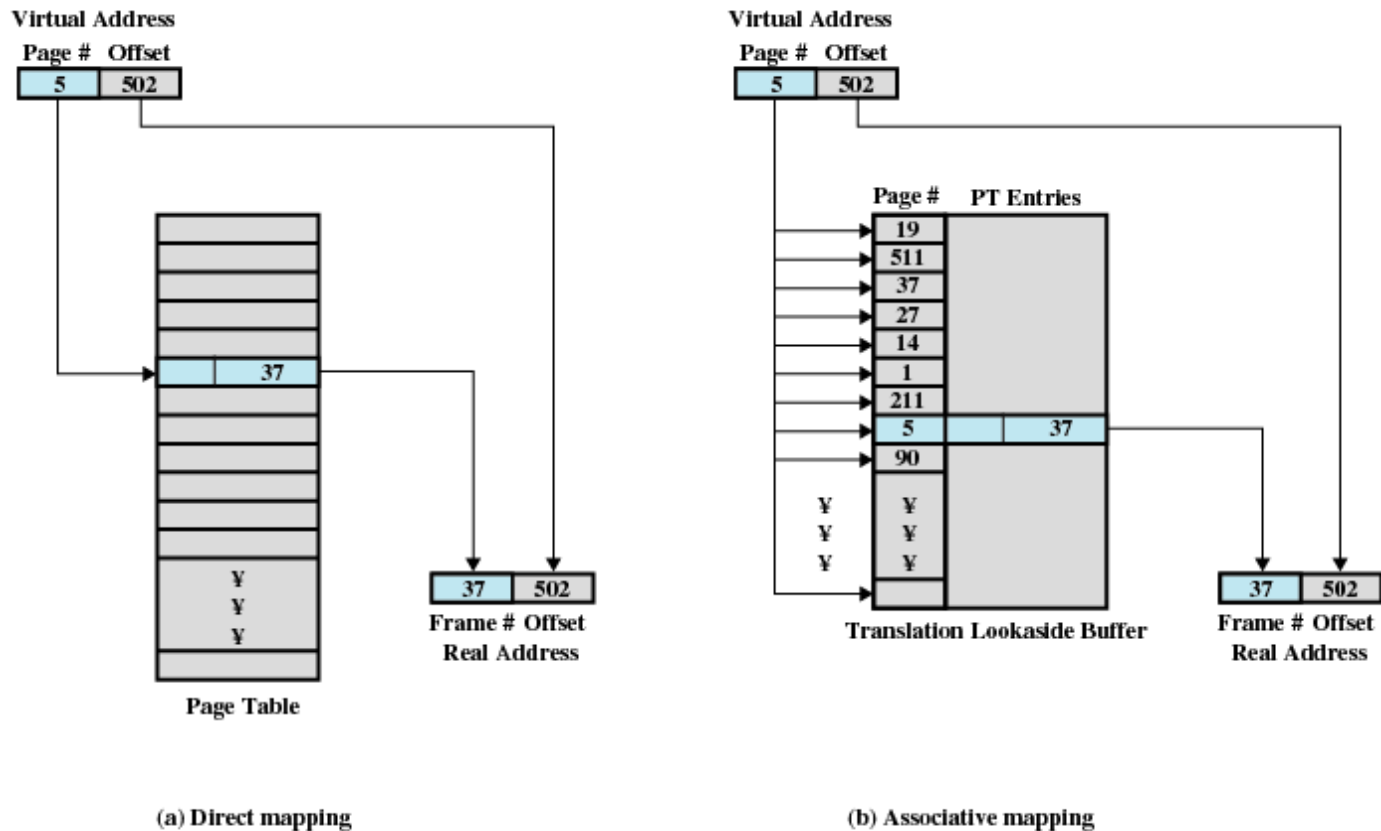


Figure 8.9 Direct Versus Associative Lookup for Page Table Entries



# Asignación de Marcos

- ✓ Cuántas paginas de un proceso se pueden encontrar en memoria?
  - ✓ Tamaño del Conjunto Residente
- ✓ Asignación Dinámica
  - ✓ El número de marcos para cada proceso varía
- ✓ Asignación Fija
  - ✓ Número fijo de marcos para cada proceso



# Asignación de Marcos - Asignación Fija

- ✓ Asignación equitativa – Ejemplo: si tengo 100 frames y 5 procesos, 20 frames para cada proceso
- ✓ Asignación Proporcional: Se asigna acorde al tamaño del proceso.

$s_i$  = size of process  $p_i$

$$S = \sum s_i$$

$m$  = total number of frames

$$a_i = \text{allocation for } p_i = \frac{s_i}{S} \times m$$

$$m = 64$$

$$s_1 = 10$$

$$s_2 = 127$$

$$a_1 = \frac{10}{137} \times 64 \approx 5$$

$$a_2 = \frac{127}{137} \times 64 \approx 59$$



# Reemplazo de páginas

- ☑ Qué sucede si ocurre un fallo de página y todos los marcos están ocupados → *“Se debe seleccionar una página víctima”*
- ☑ ¿Cual sería Reemplazo Optimo?
  - ✓ Que la página a ser removida no sea referenciada en un futuro próximo
- ☑ La mayoría de los reemplazos predicen el comportamiento futuro mirando el comportamiento pasado.



# Alcance del Reemplazo

## ☑ Reemplazo Global

- ✓ El fallo de página de un proceso puede reemplazar la página de cualquier proceso.
- ✓ El SO no controla la tasa de page-faults de cada proceso
- ✓ Puede tomar frames de otro proceso aumentando la cantidad de frames asignados a él.
- ✓ Un proceso de alta prioridad podría tomar los frames de un proceso de menor prioridad.



# *Alcance del Reemplazo (cont.)*

## ☑ Reemplazo Local

- ✓ El fallo de página de un proceso solo puede reemplazar sus propias páginas – De su Conjunto Residente
- ✓ No cambia la cantidad de frames asignados
- ✓ El SO puede determinar cual es la tasa de page-faults de cada proceso
- ✓ Un proceso puede tener frames asignados que no usa, y no pueden ser usados por otros procesos.



# Algoritmos de Reemplazo

✓ OPTIMO

✓ FIFO

✓ LRU (Least Recently Used)

✓ 2da. Chance

✓ NRU (Non Recently Used)

✓ Utiliza bits R y M

✓  $\sim R, \sim M > \sim R, M > R, \sim M > R, M$

