

Introducción a los Sistemas Operativos / Conceptos de Sistemas Operativos

Administración de Memoria - II



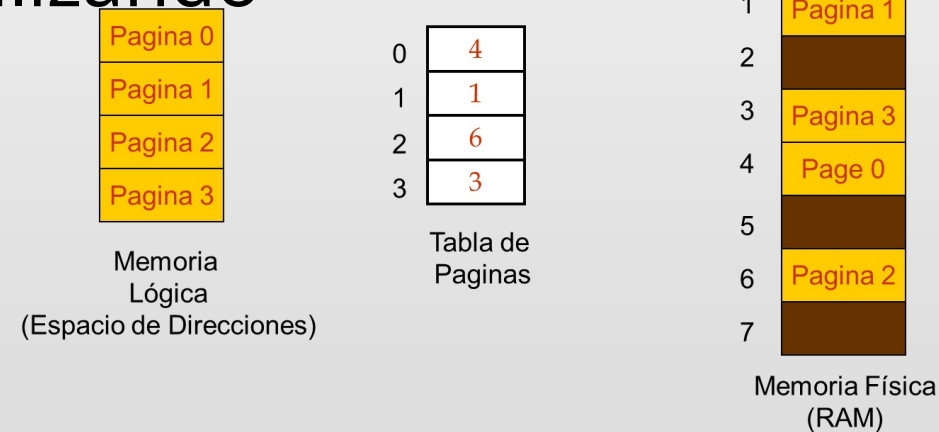
- ✓ Versión: Septiembre 2024
- ✓ Palabras Claves: Procesos, Espacio de Direcciones, Memoria, Seguridad, Paginación, Memoria Virtual, Tablas de Páginas

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 hardware traduce direcciones lógicas a direcciones físicas utilizando las tablas de páginas que el SO administra



Motivación para Memoria Virtual

- ✓ Podemos pensar también que, no todo el espacio de direcciones del proceso se necesitó 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.
- ✓ No hay necesidad que la totalidad la imagen del proceso sea cargada en memoria



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
 - ✓ Se debe cargar en memoria dicha porción para continuar con la ejecución.



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 más 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 Memoria Virtual?

- ✓ El hardware debe soportar paginación por demanda (y/o segmentación por demanda)
- ✓ 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.

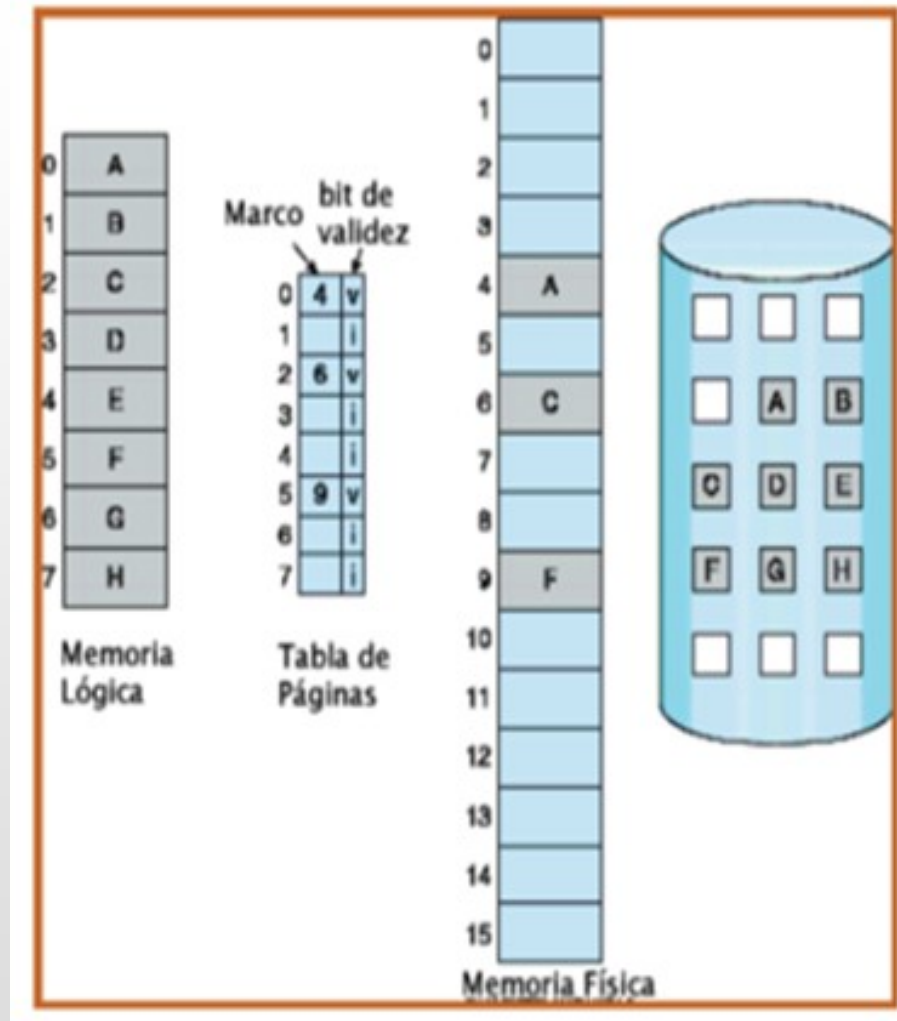


Memoria Virtual con Paginación

- ☑ 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 (lo activa/desactiva el SO, lo consulta el HW)
 - ✓ **Bit M:** Indica si la página fue modificada. Si se modificó, en algún momento, se deben reflejar los cambios en Memoria Secundaria (lo activa el HW, lo consulta y desactiva el SO)



Memoria Virtual con Paginación



Entrada en la tabla de páginas de x86 (32 bits)

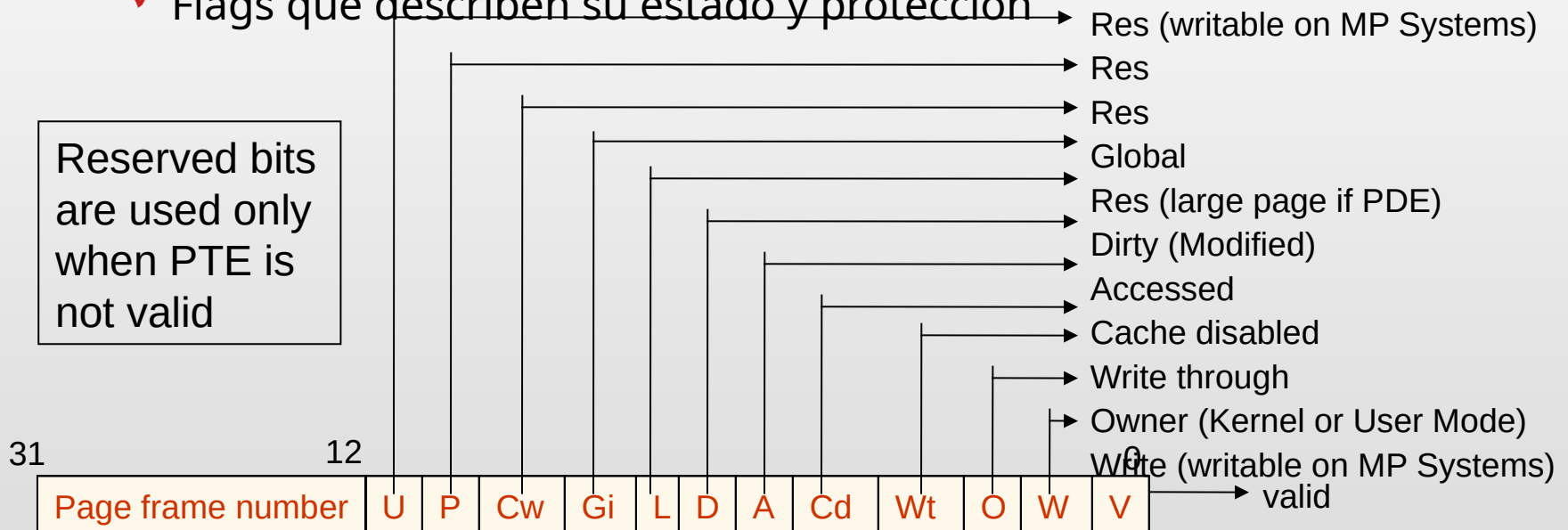
✓ El HW define el formato de la tabla de páginas y el SO se adapta a él

✓ Una entrada válida tiene:

✓ Bit V = 1

✓ Page Frame Number (PFN) – Marco de memoria asociado

✓ Flags que describen su estado y protección



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 (también marcado con i = inválido)
 - ✓ La página no se encuentra en su conjunto residente
- ☑ 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.

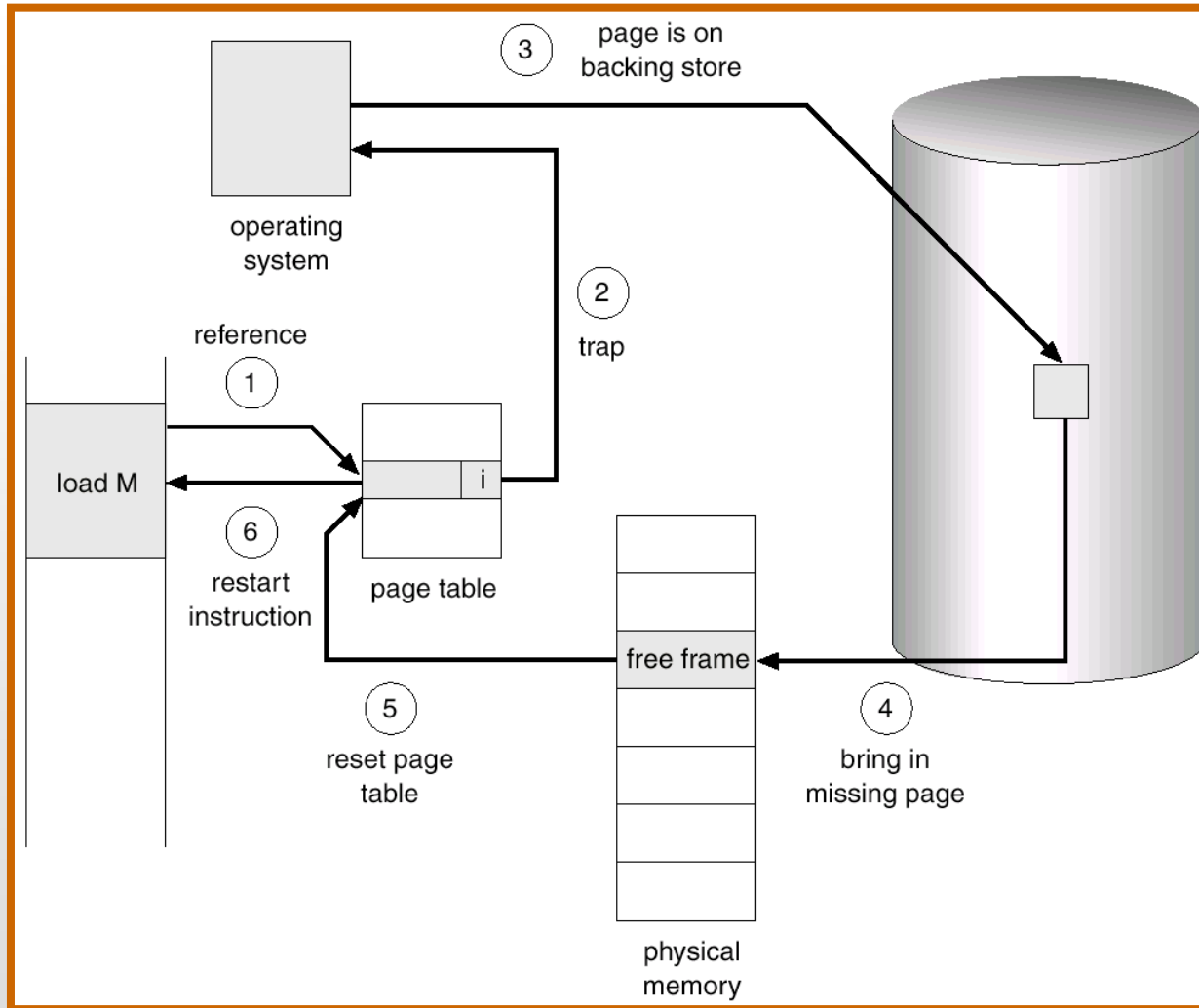


Fallo de páginas (cont.)

- ☑ 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.)



Fallo de páginas (cont.)

- ✓ La técnica de paginación intenta alocar la mayor cantidad de páginas necesarias posibles
- ✓ Cada vez que hay que alocar una página en un marco, se produce un fallo de página
- ✓ ¿Qué sucede si es necesario alocar una página y ya no hay marcos disponibles?
- ✓ Se debe seleccionar una página víctima, para lo cual existen diversos algoritmos (FIFO, Óptimo, LRU, etc.)
- ✓ ¿Cuál es el mejor algoritmo?:
- ✓ El que seleccione como página víctima aquella que no vaya a ser referenciada en un futuro próximo



Fallo de páginas Ejemplo

- ✓ El algoritmo FIFO (First In First Out) trata a los frames en uso como una cola circular
- ✓ Simple de implementar
- ✓ La página más antigua en la memoria es reemplazada
- ✓ La página puede ser necesitada pronto

| Marco/Página | 1 | 2 | 1 | 3 | 4 | 1 | 4 | 3 | 5 |
|--------------|---|---|---|---|---|---|---|---|---|
| F1 | 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 |
| F2 | | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| F3 | | | | 3 | 3 | 3 | 3 | 3 | 5 |
| PF? | X | X | | X | X | X | | | X |



Performance

- ✓ Si los page faults son excesivos, la performance del sistema decae
- ✓ 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)

$$\begin{aligned} \text{EAT} = & (1 - p) \times \text{memory access} \\ & + p \times (\text{page_fault_overhead} + \\ & \quad [\text{swap_page_out}] \\ & \quad \text{swap_page_in} \\ & \quad \text{restart_overhead}) \end{aligned}$$

Podría ocurrir que no haya marcos disponibles, con lo cual habrá que descargar uno para lograr espacio para la nueva página entrante



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



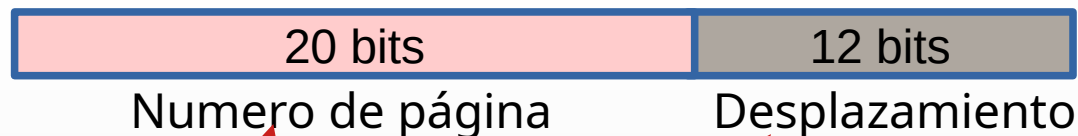
Tabla de páginas (cont.)

- ☑ 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 1 nivel – 32 bits

✓ Direcciones de 32bits



✓ Ejemplo

- ✓ Cantidad de Page Table Entries (PTEs) máximas que puede tener un proceso = 2^{20} (1.048.576)
- ✓ El tamaño de cada página es de 4KB (2^{12})
- ✓ El tamaño de cada PTE es de 4 bytes
 - ✓ Cantidad de PTEs que entran en un marco: $4\text{KB}/4\text{B} = 2^{10}$
- ✓ Tamaño de tabla de páginas
 - ♦ Cantidad de marcos necesarios para todas las PTEs de la tabla de páginas de un proceso = $2^{20}/2^{10} = 2^{10}$
 - ♦ Tamaño tabla de páginas del proceso:
 $2^{10} * 4\text{bytes} = \mathbf{4\text{MB por proceso}}$



Tabla de 1 nivel – 64 bits

☑ Direcciones de 64bits

| | |
|------------------|----------------|
| 52 bits | 12 bits |
| Numero de página | Desplazamiento |

☑ Ejemplo

- ✓ Cantidad de Page Table Entries (PTEs) máximas que puede tener un proceso = 2^{52}
- ✓ El tamaño de cada página es de 4KB
- ✓ El tamaño de cada PTE es de 4 bytes
 - ♦ Cantidad de PTEs que entran en un marco: $4KB/4B = 2^{10}$
- ✓ Tamaño de tabla de páginas
 - ♦ Cantidad de marcos necesarios para todas las PTEs de la tabla de páginas de un proceso = $2^{52}/2^{10} = 2^{42}$
 - ♦ Tamaño tabla de páginas del proceso = $2^{42} * 4bytes = 2^{54}$
Más de 16.000GB por proceso!!!



Tabla de páginas - Tabla de 2 niveles

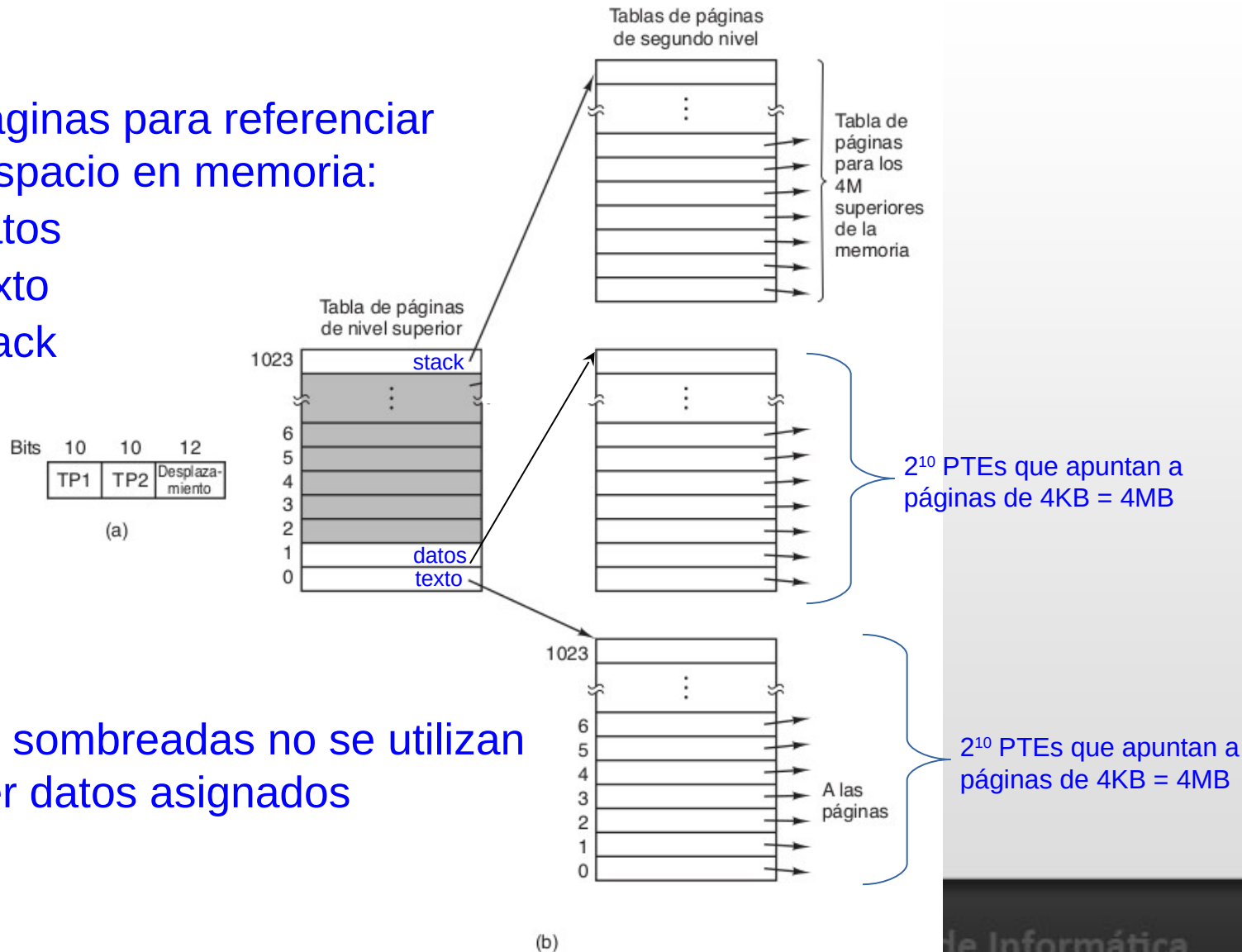
- ✓ El propósito de la tabla de páginas multinivel es dividir la tabla de páginas lineal en múltiples tablas de páginas
- ✓ Cada tabla de páginas suele tener el mismo tamaño pero se busca que tengan un menor número de páginas por tabla
- ✓ La idea general es que cada tabla sea más pequeña
- ✓ Se busca que la tabla de páginas no ocupe demasiada memoria RAM
- ✓ Además solo se carga una parcialidad de la tabla de páginas (solo lo que se necesite resolver)
- ✓ Existe un esquema de direccionamientos indirectos



Ejemplo: mapeo en memoria de tabla de páginas de 2 niveles

Se usan 3 páginas para referenciar
12MB de espacio en memoria:

- 4MB de datos
- 4MB de texto
- 4MB de stack

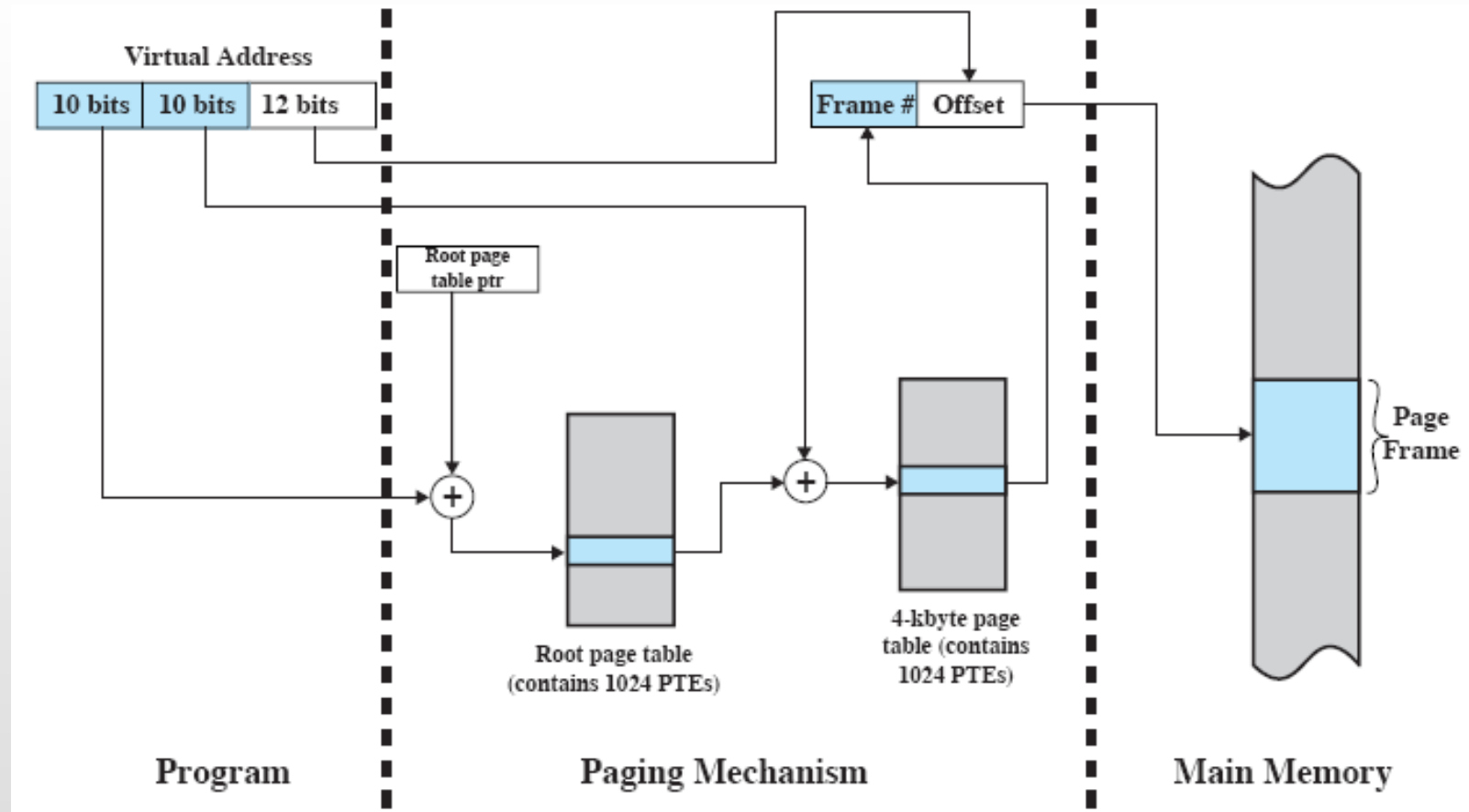


Las entradas sombreadas no se utilizan
por no tener datos asignados

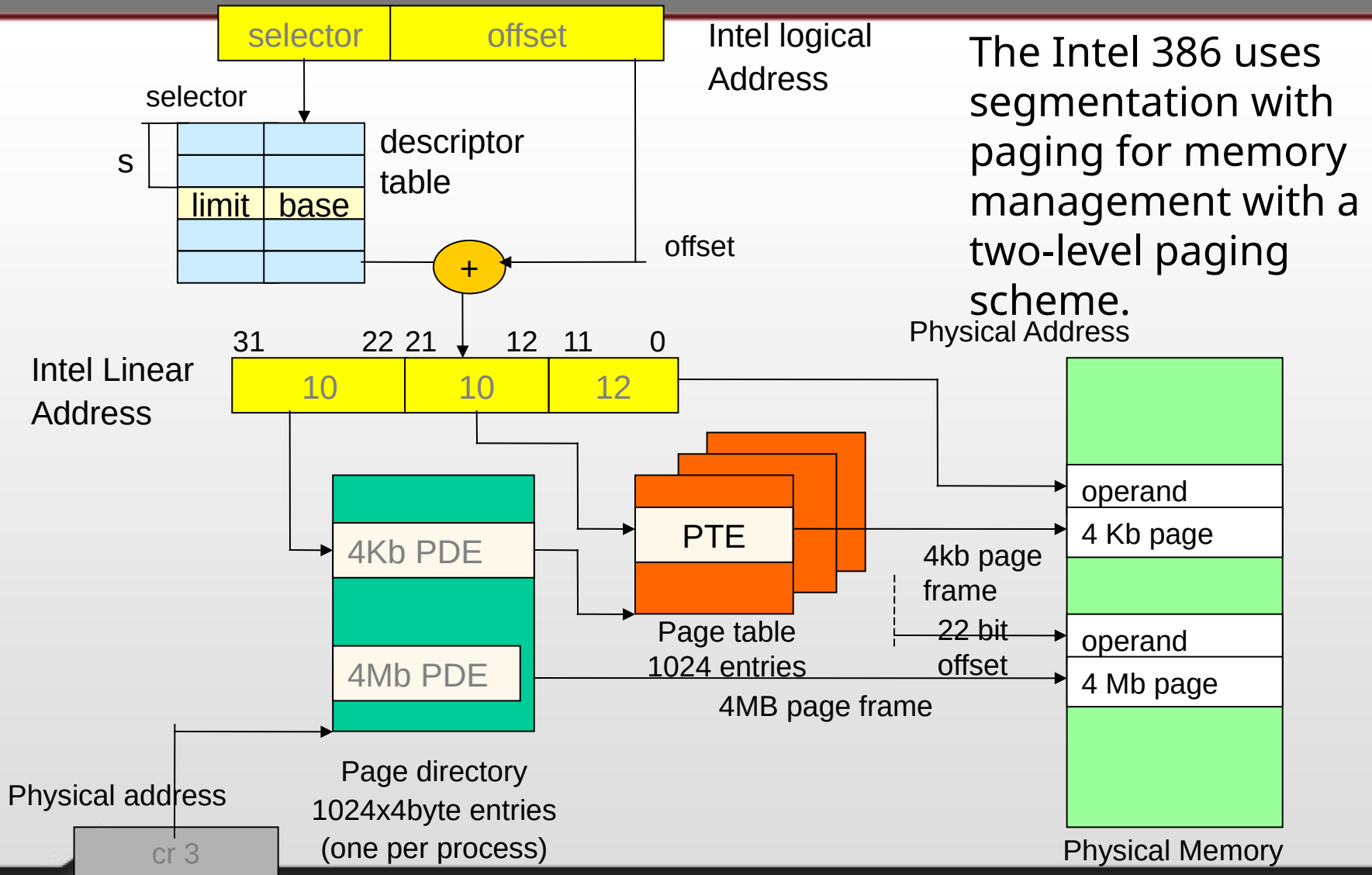
Figura 3-13. (a) Una dirección de 32 bits con dos campos de tablas de páginas.
(b) Tablas de páginas de dos niveles.



Tabla de Páginas - Tabla de 2 niveles



Intel 30386



Tablas de Páginas - x64

- ✓ Se usan 47 bits para direccionar
- ✓ Usa tabla de páginas de 4 niveles

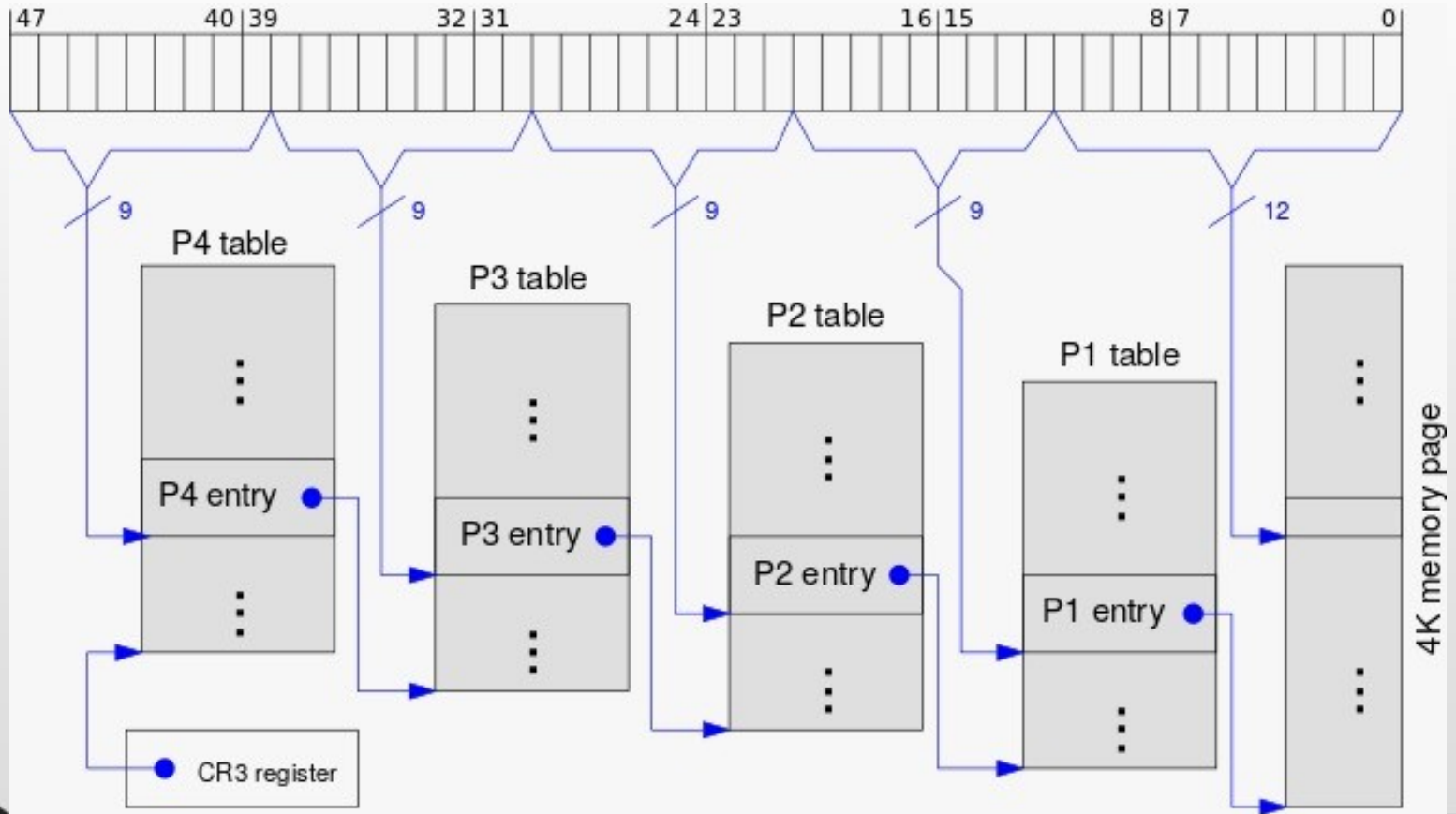


Tabla de Paginas (cont.) - Tabla invertida

- ✓ Utilizada en Arquitecturas donde el espacio de direcciones es muy grande
 - ✓ Las tablas de páginas ocuparían muchos niveles y la traducción sería costosa
 - ✓ Por esta razón se adopta esta técnica
- ✓ Por ejemplo, si el espacio de direcciones es de 2^{64} bytes, con páginas de 4 KB, necesitamos una tabla de páginas con 2^{52} entradas
- ✓ Si cada entrada es de 8 bytes, la tabla es de más de 30 millones de Gigabyes (30 PB)



Tabla de Paginas (cont.) - Tabla invertida

- ✓ Hay una entrada por cada marco de página en la memoria real. Es la visión inversa a la que veníamos viendo
- ✓ Hay una sola tabla para todo el sistema
- ✓ El espacio de direcciones de la tabla se refiere al espacio físico de la RAM, en vez del espacio de direcciones virtuales de un proceso
- ✓ 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
- ✓ Se define un mecanismo de encadenamiento para solucionar colisiones (cuando el hash da igual para 2 direcciones virtuales)

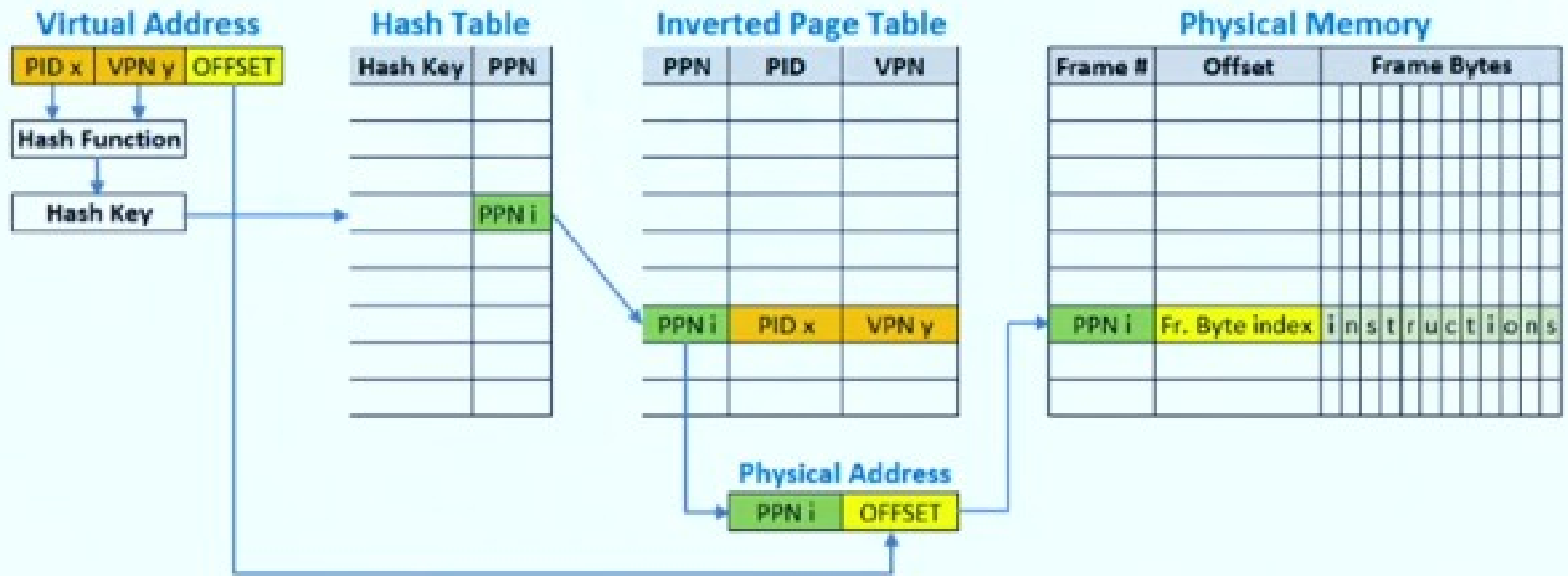


Tabla de Paginas (cont.) - Tabla invertida

- ☑ Sólo se mantienen los PTEs de páginas presentes en memoria física
 - ✓ La tabla invertida es 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



<https://www.youtube.com/watch?v=2zEGiZga04g>



Tamaño de la Pagina

☑ Pequeño

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

☑ Grande

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



Tamaño de la Pagina (cont.)

☑ Relación con la E/S

- ✓ Vel. De transferencia: 2 Mb/s
- ✓ Latencia: 8 ms
- ✓ Búsqueda: 20 ms

☑ Pagina de 512 bytes

- 1 pagina → total: 28,2 ms
- Solo 0,2 ms de transferencia (1%)
- 2 paginas → 56,4 ms

☑ Pagina de 1024 bytes

- total: 28,4 ms
- Solo 0,4 ms de transferencia



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

- ☑ 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 (cont.)

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



Translation Lookaside Buffer (cont.)

- ✓ Si la entrada no es encontrada en la TLB (miss), el número de página es usado como índice en la tabla de páginas del proceso.
- ✓ Se controla si la página está en la memoria
 - ✓ Si no está, se genera un Page Fault
- ✓ La TLB es actualizada para incluir la nueva entrada
- ✓ El cambio de contexto genera la invalidación de las entradas de la TLB □ Analizar. ¿Qué ocurre si el Quantum en Round Robin es chico? ¿y al contrario?



Translation Lookaside Buffer (cont.)

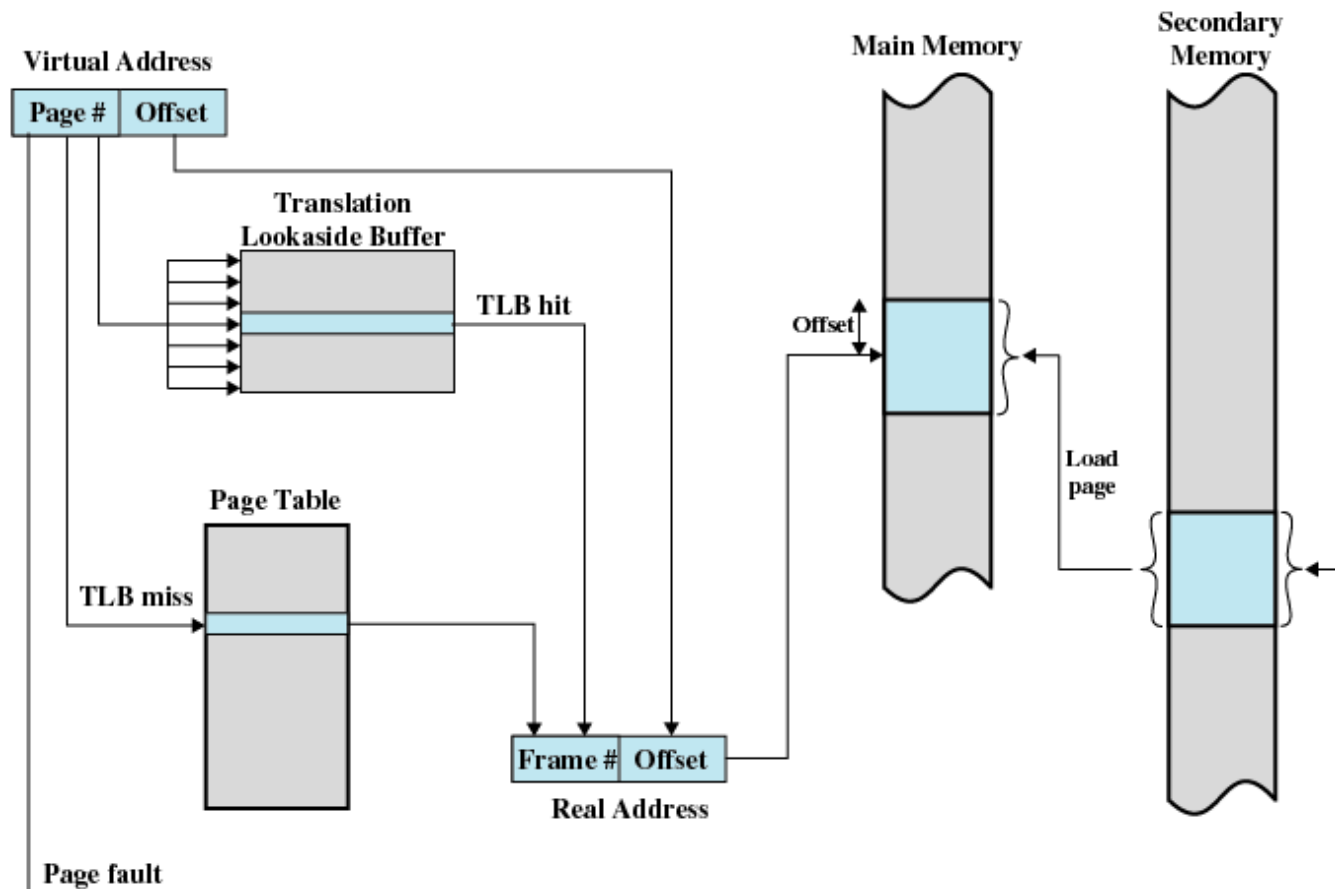


Figure 8.7 Use of a Translation Lookaside Buffer



Translation Lookaside Buffer (cont.)

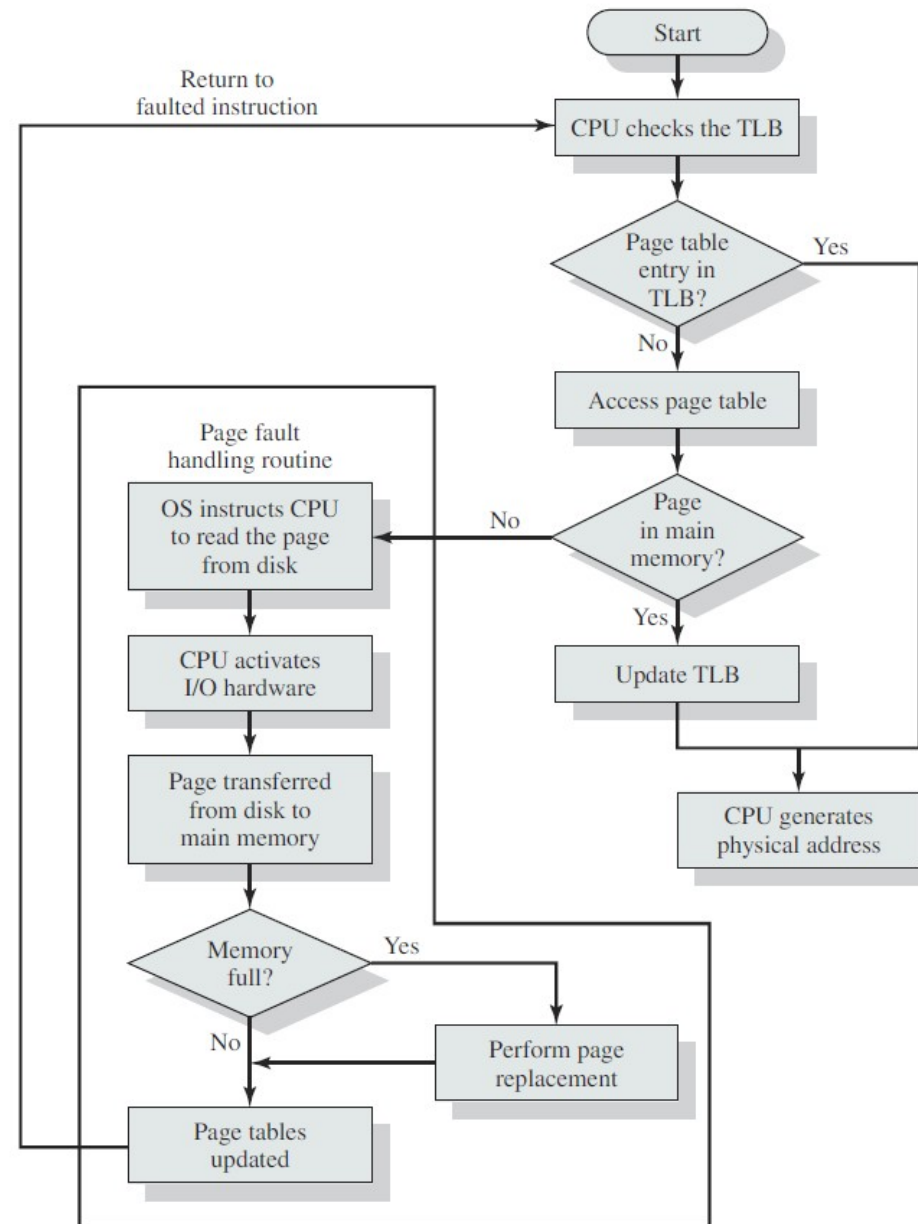


Figure 8.8 Operation of Paging and Translation Lookaside Buffer (TLB)



Políticas en el manejo de MV

Table 8.4 Operating System Policies for Virtual Memory

| | | | |
|---|--|--|---|
| Fetch Policy Demand paging Prepaging | <i>Cuando una página debe ser llevada a la memoria</i> | Resident Set Management Resident set size Fixed Variable | <i>Cuántas páginas se traen a memoria</i> |
| Placement Policy | <i>Donde ubicarla (best-fit, first-fit, etc...)</i> | Replacement Scope Global Local | |
| Replacement Policy Basic Algorithms Optimal Least recently used (LRU) First-in-first-out (FIFO) Clock Page Buffering | <i>Elección de víctima</i> | Cleaning Policy Demand Precleaning | <i>Cuando una página modificada debe llevarse a disco</i> |
| | | Load Control Degree of multiprogramming | <i># de procesos en memoria</i> |



Asignación de Marcos

- ✓ ¿Cuántas páginas 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.

$$s_i = \text{size of process } p_i$$
$$S = \sum_i s_i$$

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”*
- ✓ ¿Cuál 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 cuál 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.



Asignación y Alcance

Table 8.5 Resident Set Management

| | Local Replacement | Global Replacement |
|----------------------------|---|--|
| Fixed Allocation | <ul style="list-style-type: none">• Number of frames allocated to a process is fixed.• Page to be replaced is chosen from among the frames allocated to that process. | <ul style="list-style-type: none">• Not possible. |
| Variable Allocation | <ul style="list-style-type: none">• The number of frames allocated to a process may be changed from time to time to maintain the working set of the process.• Page to be replaced is chosen from among the frames allocated to that process. | <ul style="list-style-type: none">• Page to be replaced is chosen from all available frames in main memory; this causes the size of the resident set of processes to vary. |



Algoritmos de Reemplazo

- ✓ **OPTIMO:** Es solo teórico
- ✓ **FIFO:** Es el más sencillo
- ✓ **LRU (Least Recently Used):** Requiere soporte del hardware para mantener timestamps de acceso a las páginas. Favorece a las páginas más recientemente accedidas
- ✓ **2da. Chance:** Un avance del FIFO tradicional que beneficia a las páginas más referenciadas
- ✓ **NRU (Non Recently Used):**
 - ✓ Utiliza bits R y M
 - ✓ Favorece a las páginas que fueron usadas recientemente

