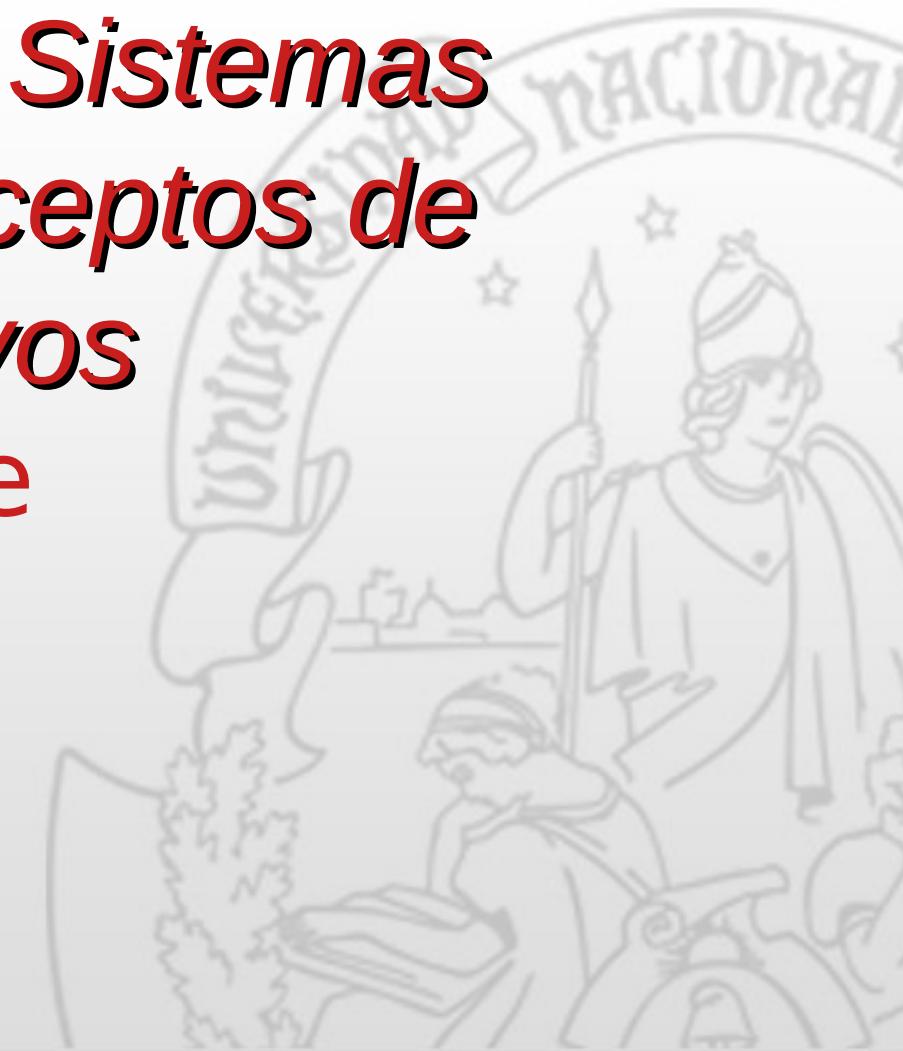


Introducción a los Sistemas Operativos / Conceptos de Sistemas Operativos

Administración de Memoria - I



- Versión: Septiembre 2019
- Palabras Claves: Procesos, Espacio de Direcciones, Memoria, Seguridad, Paginación, Segmentación, Fragmentación

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.



Memoria

- ❑ La organización y administración de la “memoria principal” es uno de los factores más importantes en el diseño de los S. O.
- ❑ Los programas y datos deben estar en el almacenamiento principal para:
 - ❑ Poderlos ejecutar.
 - ❑ Referenciarlos directamente.



Memoria

(cont.)

- El SO debe:
 - Llevar un registro de las partes de memoria que se están utilizando y de aquellas que no.
 - Asignae espacio en memoria principal a los procesos cuando estos la necesitan.
 - Libera espacio de memoria asignada a procesos que han terminado.
- Se espera de un S.O. un uso eficiente de la memoria con el fin de alojar el mayor número de procesos



Memoria

(cont.)

□ El S.O. debe:

- Lograr que el programador se abstraiga de la alocación de los programas
- Brindar seguridad entre los procesos para que unos no accedan a secciones privadas de otros
- Brindar la posibilidad de acceso compartido a determinadas secciones de la memoria (librerías, código en común, etc.)
- Garantizar la performance del sistema



Administración de Memoria

- ✓ División Lógica de la Memoria Física para alojar múltiples procesos
 - Garantizando protección
 - Depende del mecanismo provisto por el HW
- ✓ Asignación eficientemente
 - Contener el mayor numero de procesos para garantizar el mayor uso de la CPU por los mismos



Requisitos

Reubicación

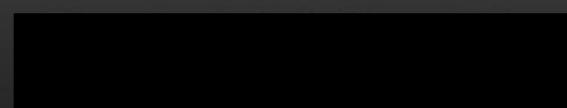
- ✓ El programador no debe ocuparse de conocer donde será colocado en la Memoria RAM
- ✓ Mientras un proceso se ejecuta, puede ser sacado y traído a la memoria (swap) y, posiblemente, colocarse en diferentes direcciones.
- ✓ Las referencias a la memoria se deben “traducir” según ubicación actual del proceso.



Requisitos (cont.).

Protección

- ✓ Los procesos NO deben referenciar – acceder - a direcciones de memoria de otros procesos
 - Salvo que tengan permiso
- ✓ El chequeo se debe realizar durante la ejecución:
 - ◆ NO es posible anticipar todas las referencias a memoria que un proceso puede realizar.



Requisitos (cont.).

Compartición

- ✓ Permitir que varios procesos accedan a la misma porción de memoria.
 - ◆ Ej: Rutinas comunes, librerías, espacios explícitamente compartidos, etc.
- ✓ Permite un mejor uso - aprovechamiento - de la memoria RAM, evitando copias innecesarias (repetidas) de instrucciones

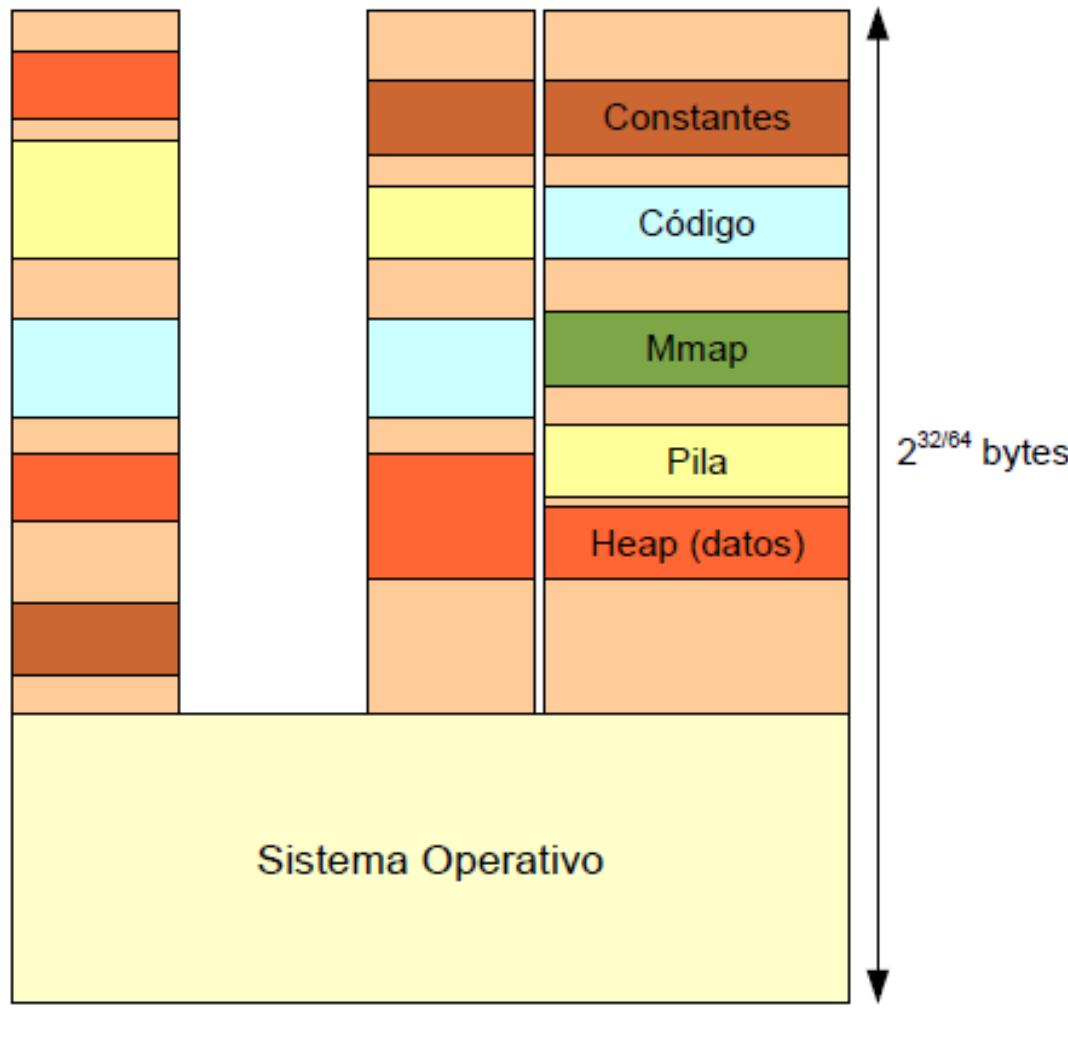


Abstracción - Espacio de Direcciones

- Rango de direcciones (a memoria) posibles que un proceso puede utilizar para direccionar sus instrucciones y datos.
- El tamaño depende de la Arquitectura del Procesador
 - ✓ 32 bits: $0 .. 2^{32} - 1$
 - ✓ 64 bits: $0 .. 2^{64} - 1$
- Es independiente de la ubicación “real” del proceso en la Memoria RAM



Abstracción -Espacio de Direcciones (cont.)



Direcciones

Lógicas

- ✓ Referencia a una localidad de memoria independiente de la asignación actual de los datos en la memoria.
- ✓ Representa una dirección en el “Espacio de Direcciones del Proceso”

Físicas

- ✓ Referencia una localidad en la Memoria Física (RAM)
 - Dirección absoluta

En caso de usar direcciones Lógicas, es necesaria algún tipo de conversión a direcciones Físicas.



Conversión de Direcciones

Una forma simple de hacer esto es utilizando registros auxiliares

Registro Base

- ✓ Dirección de comienzo del Espacio de Direcciones del proceso en la RAM

Registro Límite

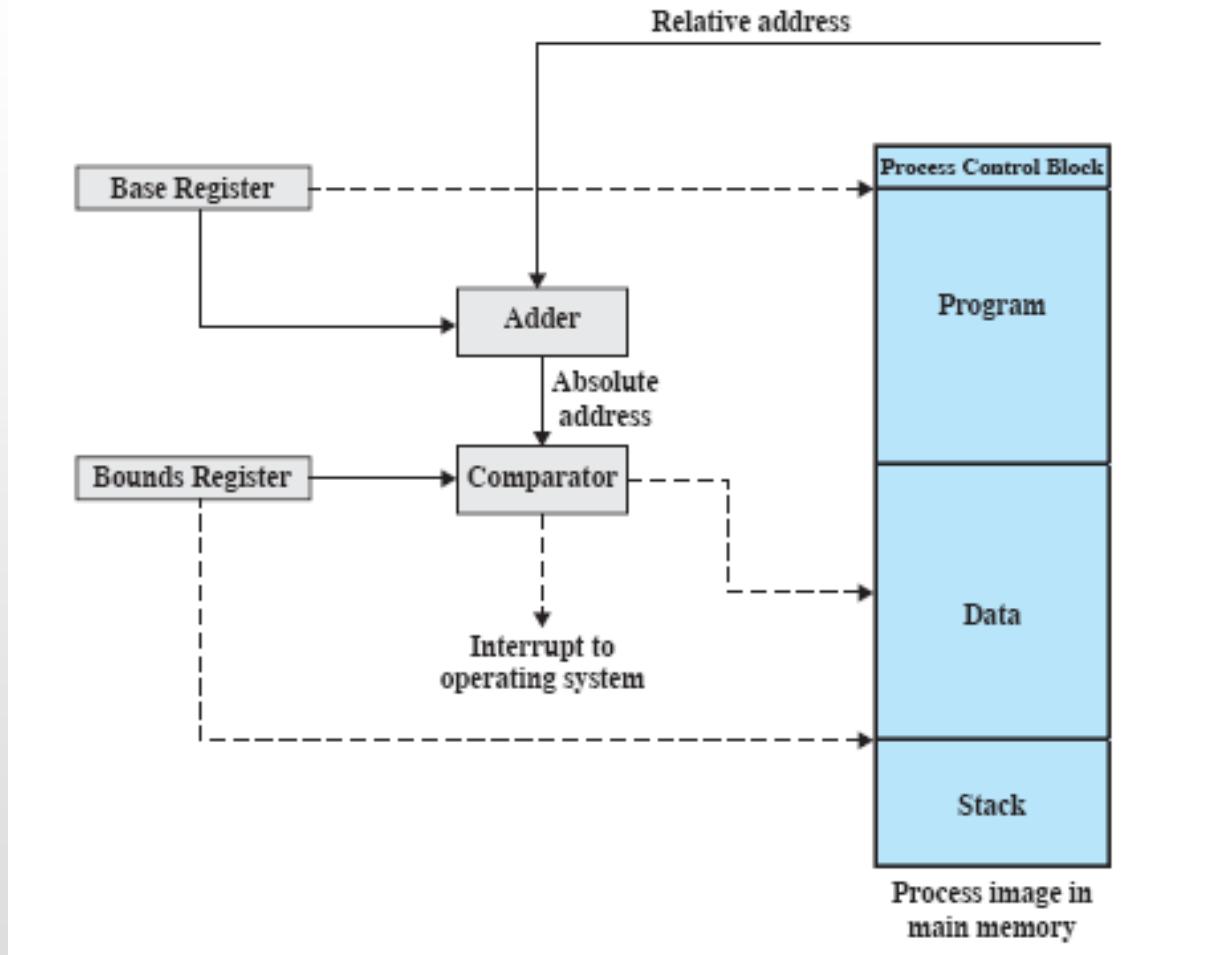
- ✓ Dirección final del proceso o medida del proceso
 - Tamaño de su Espacio de Direcciones

Ambos valores se fijan cuando el espacio de direcciones del proceso es cargado a memoria.

Varían entre procesos (Context Switch)



Direcciones (cont.)



Dir. Lógicas vs. Físicas

- Si la CPU trabaja con direcciones lógicas, para acceder a memoria principal, se deben transformar en direcciones físicas.
 - Resolución de direcciones (address-binding): transformar la dirección lógica en la dirección física correspondiente
- Resolución en momento de compilación (Archivos .com de DOS) y en tiempo de carga
 - ✓ Direcciones Lógicas y Físicas son idénticas
 - ✓ Para reubicar un proceso es necesario recompilarlo o recargarlo.



Dir. Lógicas vs. Físicas

Resolución en tiempo de ejecución

- ✓ Direcciones Lógicas y Físicas son diferentes
- ✓ Direcciones Lógicas son llamadas “Direcciones Virtuales”
- ✓ La reubicación se puede realizar fácilmente
- ✓ El mapeo entre “Virtuales” y “Físicas” es realizado por hardware
 - Memory Management Unit (MMU)



Memory Management Unit (MMU)

Dispositivo de Hardware que mapea direcciones virtuales a físicas

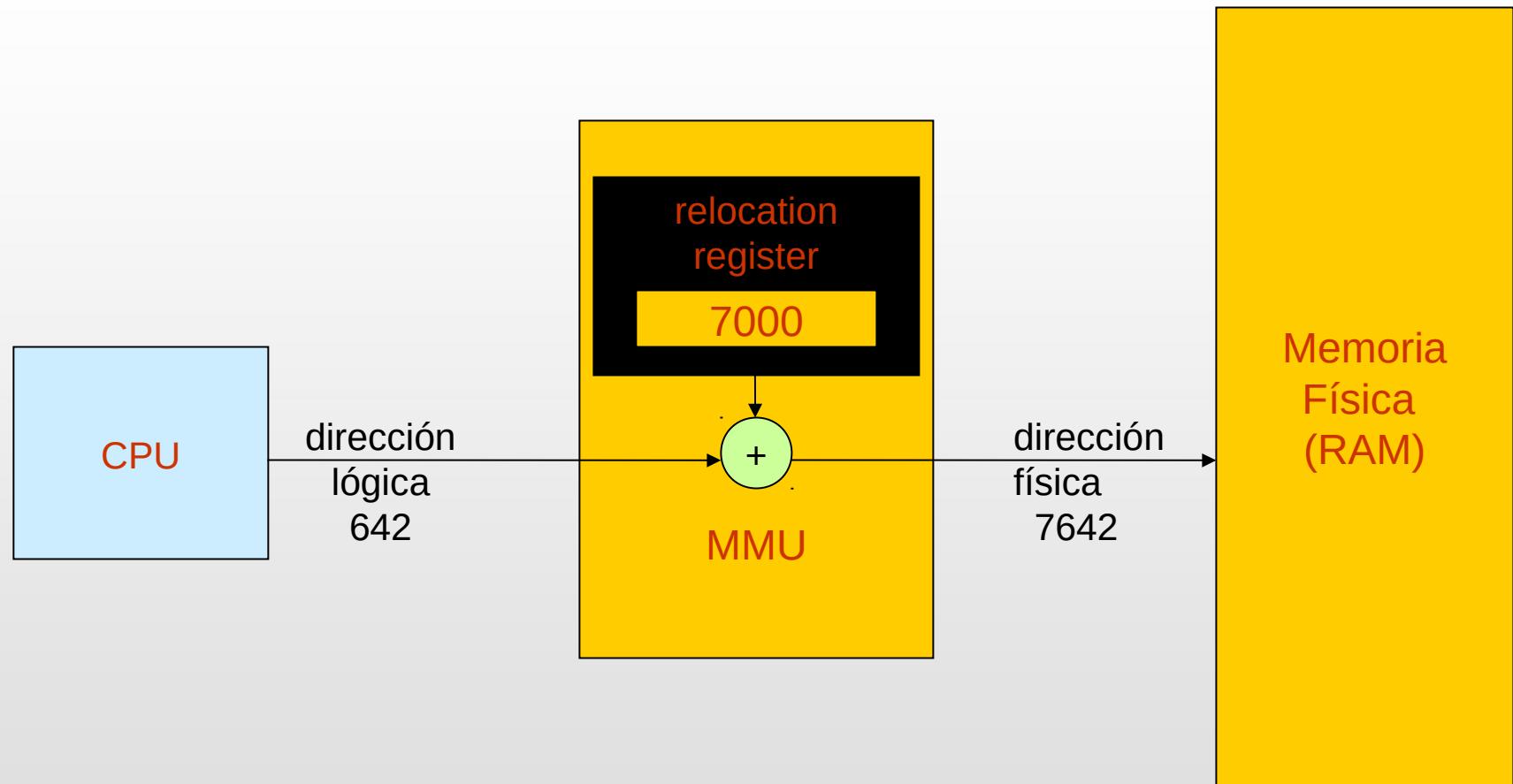
- ✓ Es parte del Procesador
- ✓ Re-programar el MMU es una operación privilegiada
 - solo puede ser realizada en Kernel Mode

El valor en el “registro de realocación” es sumado a cada dirección generada por el proceso de usuario al momento de acceder a la memoria.

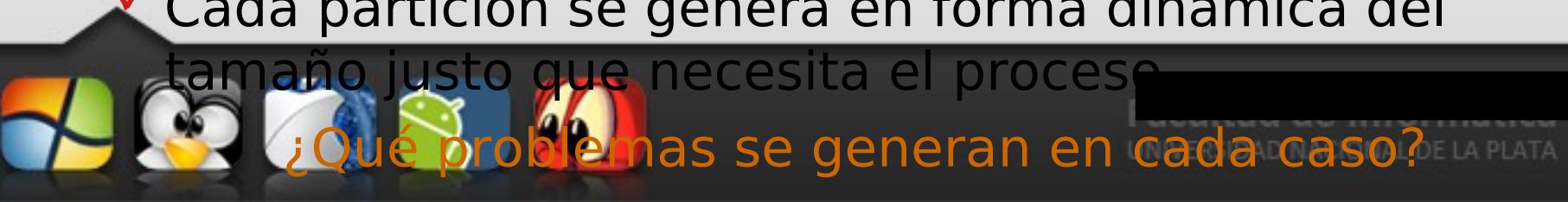
- ✓ Los procesos nunca usan direcciones físicas



MMU



Mecanismos de asignación de memoria

- Particiones Fijas: El primer esquema implementado
 - ✓ La memoria se divide en particiones o regiones de tamaño Fijo (pueden ser todas del mismo tamaño o no)
 - ✓ Alojan un proceso cada una
 - ✓ Cada proceso se coloca de acuerdo a algún criterio (First Fit, Best Fit, Worst Fit, Next Fit) en alguna partición
 - Particiones dinámicas: La evolución del esquema anterior
 - ✓ Las particiones varían en tamaño y en número
 - ✓ Alojan un proceso cada una
 - ✓ Cada partición se genera en forma dinámica del tamaño justo que necesita el proceso
- 
- ¿Qué problemas se generan en cada caso?

Fragmentación

La fragmentación se produce cuando una localidad de memoria no puede ser utilizada por no encontrarse en forma contigua

Fragmentación Interna:

- ✓ Se produce en el esquema de particiones Fijas
- ✓ Es la porción de la partición que queda sin utilizar

Fragmentación Externa:

- ✓ Se produce en el esquema de particiones dinámicas
- ✓ Son huecos que van quedando en la memoria a medida que los procesos finalizan
- ✓ Al no encontrarse en forma contigua puede darse el caso de que tengamos memoria libre para alocar un proceso, pero que no la podamos utilizar
- ✓ Para solucionar el problema se puede acudir a la compactación, pero es muy costosa



Problemas del esquema

El esquema de Registro Base + Límite presenta problemas:

- Necesidad de almacenar el Espacio de Direcciones de forma continua en la Memoria Física
- Los primeros SO definían particiones fijas de memoria, luego evolucionaron a particiones dinámicas
- Fragmentación
- Mantener “partes” del proceso que no son necesarias
- Los esquemas de particiones fijas y dinámicas no se usan hoy en día

Solución:

- Paginación
- Segmentación

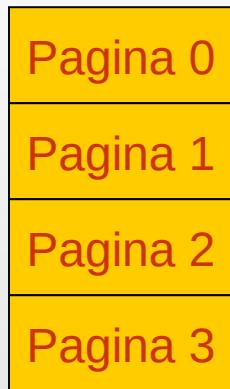


Paginación

- Memoria Física es dividida lógicamente en pequeños trozos de igual tamaño → **Marcos**
- Memoria Lógica (espacio de direcciones) es dividido en trozos de igual tamaño que los marcos → **Paginas**
- El SO debe mantener una tabla de paginas por cada proceso, donde cada entrada contiene (entre otras) el **Marco** en la que se coloca cada pagina.
- La dirección lógica se interpreta como:
 - un numero de pagina y un desplazamiento dentro de la misma.



Paginación - Ejemplo 1

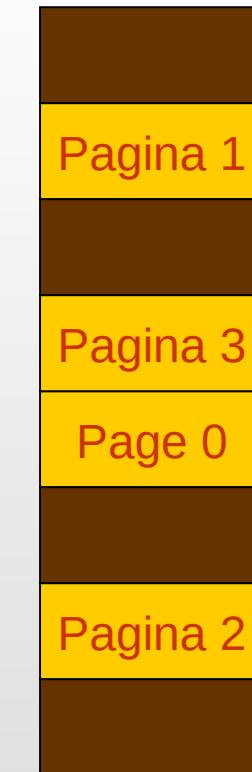


Memoria
Lógica
(Espacio de Direcciones)

0	4
1	1
2	6
3	3

Tabla de
Páginas

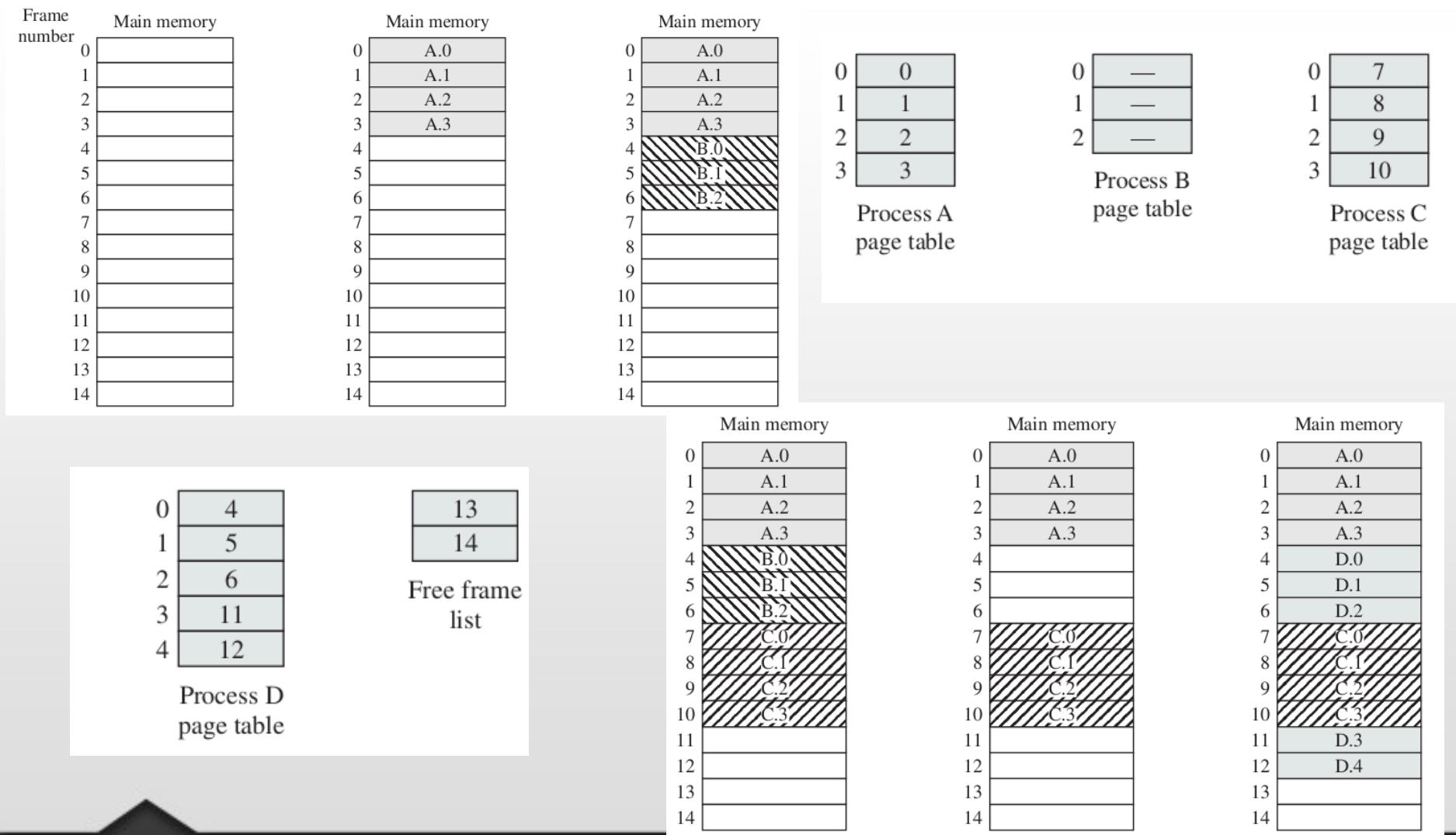
Marco



Memoria Física
(RAM)



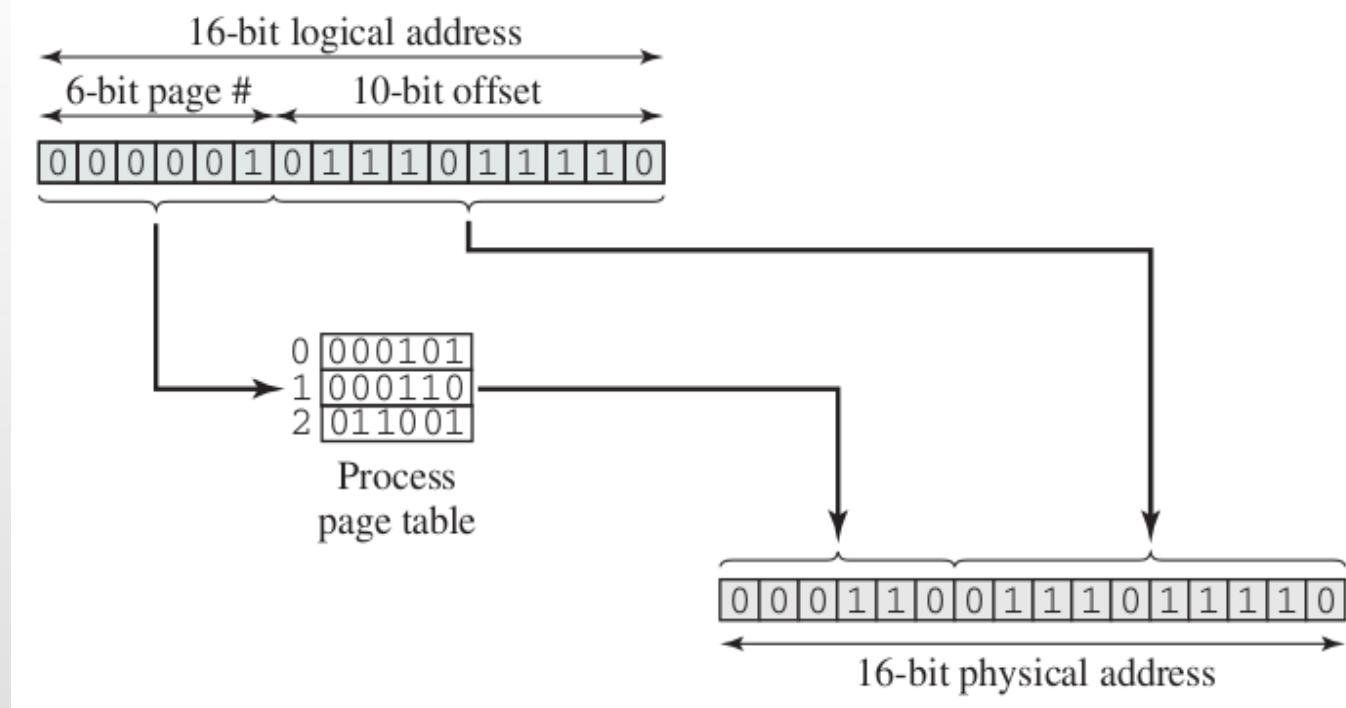
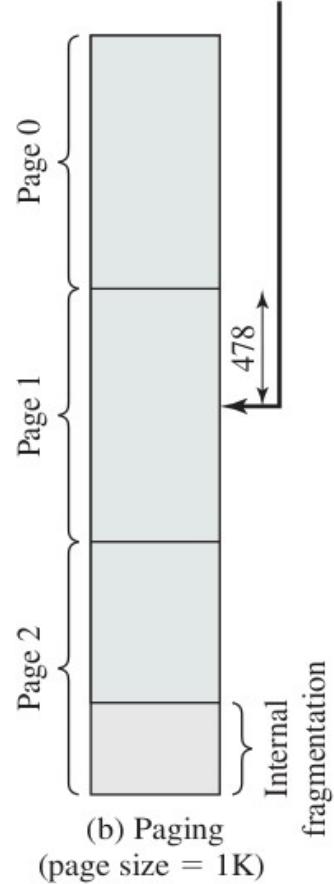
Paginación - Ejemplo II



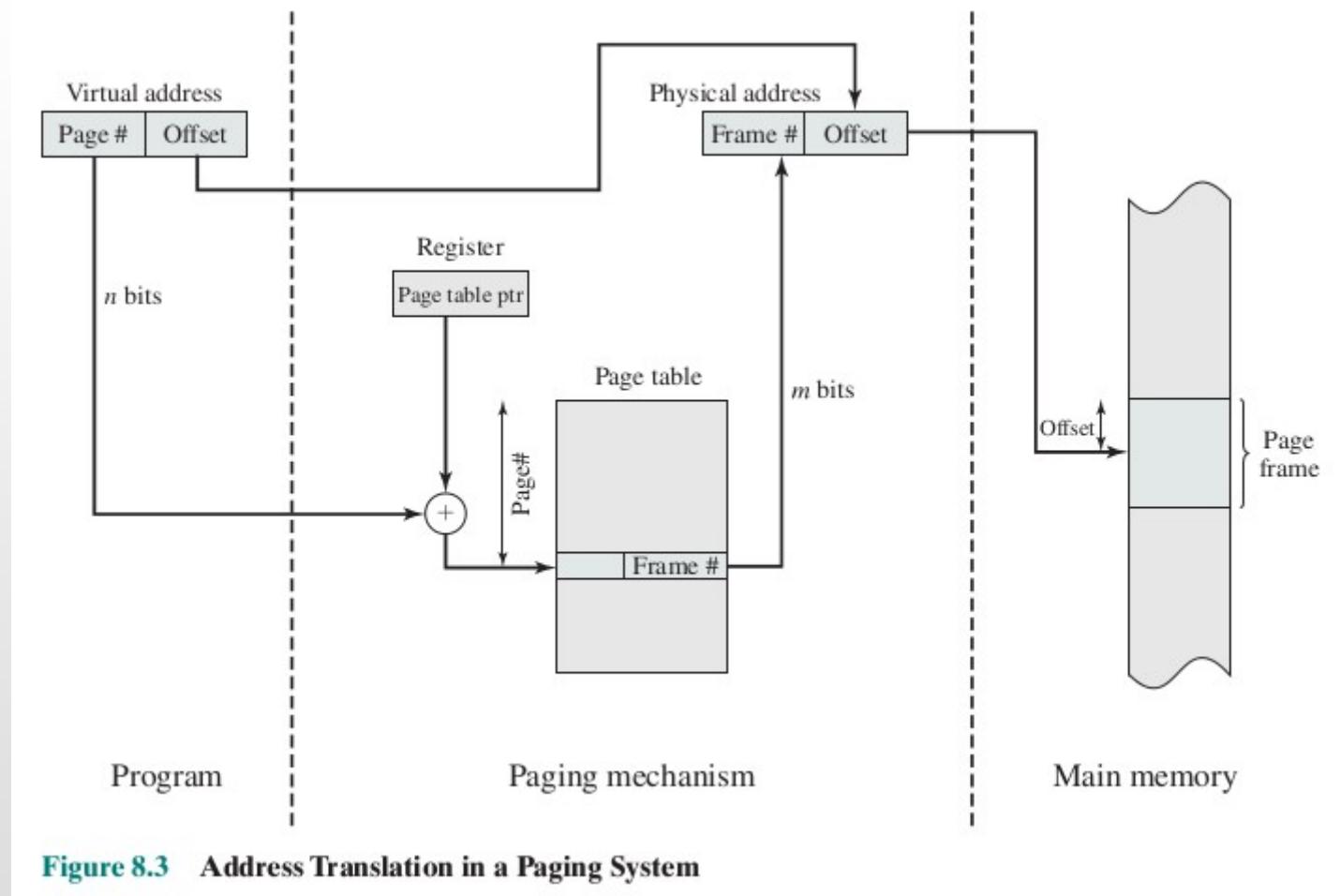
Paginación - Direcciones Lógicas

Logical address =
Page# = 1, Offset = 478

000001|0111011110



Traducción de direcciones

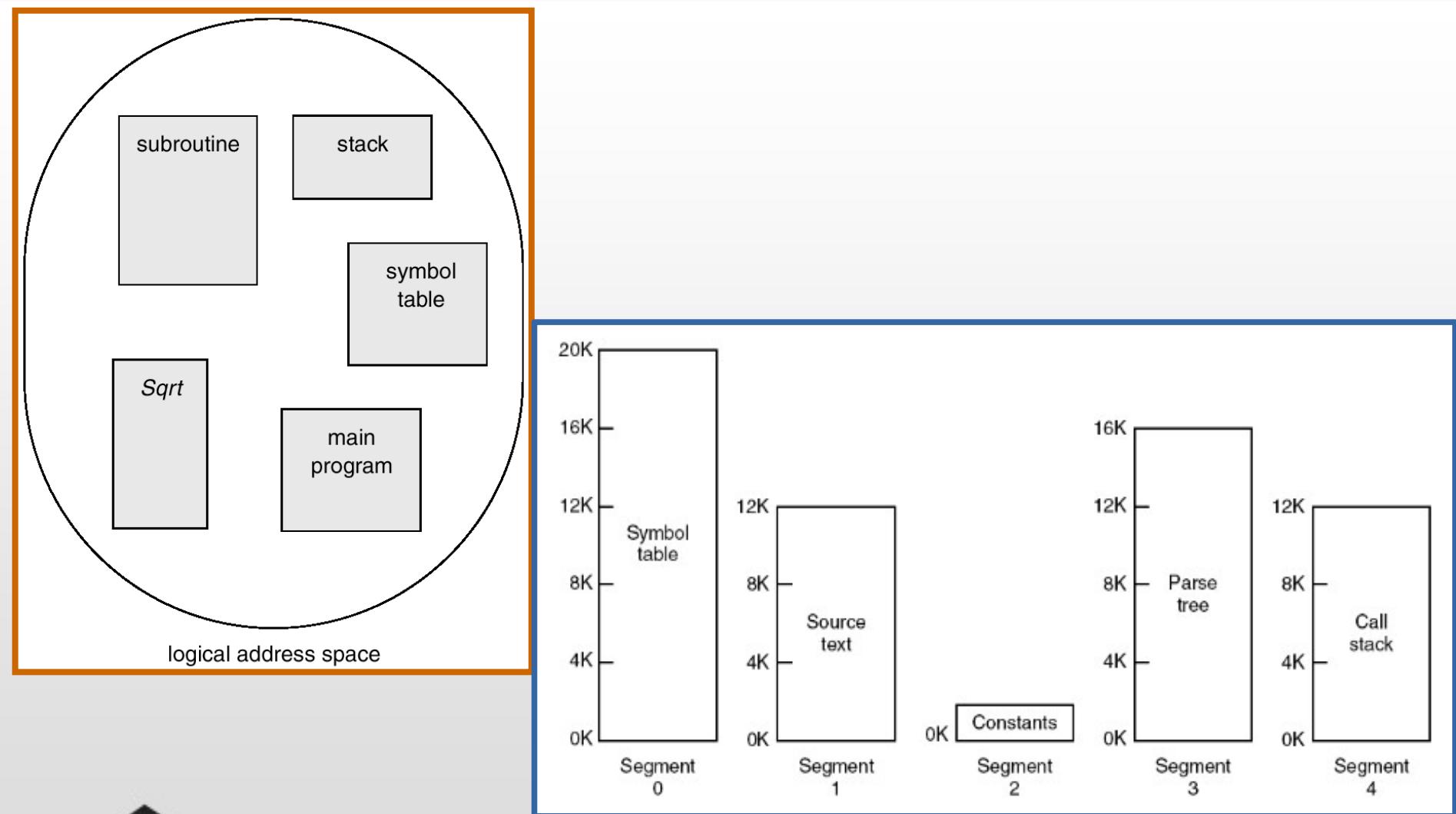


Segmentación

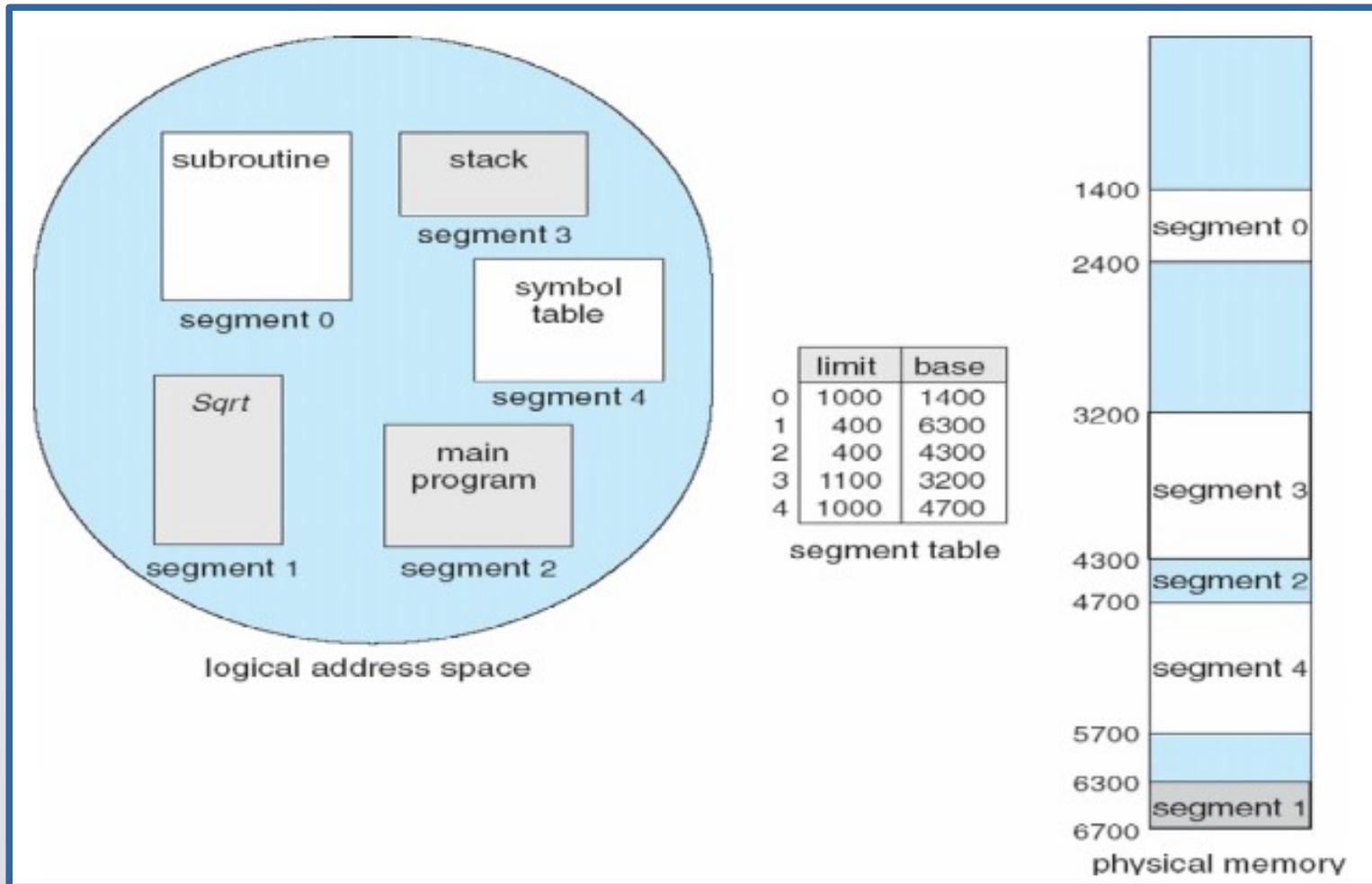
- Esquema que se asemeja a la “visión del usuario”. El programa se divide en partes/secciones
- Un programa es una colección de segmentos. Un segmento es una unidad lógica como:
 - ✓ Programa Principal, Procedimientos y Funciones, variables locales y globales, stack, etc.
- Puede causar Fragmentación



Programa desde la visión del usuario

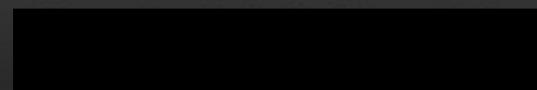


Ejemplo de Segmentación



Segmentación (cont.)

- ✓ Todos los segmentos de un programa pueden no tener el mismo tamaño (código, datos, rutinas).
- ✓ Las direcciones Lógicas consisten en 2 partes:
 - ✓ Selector de Segmento
 - ✓ Desplazamiento dentro del segmento



Segmentación (cont.) - Arquitectura

Tabla de Segmentos

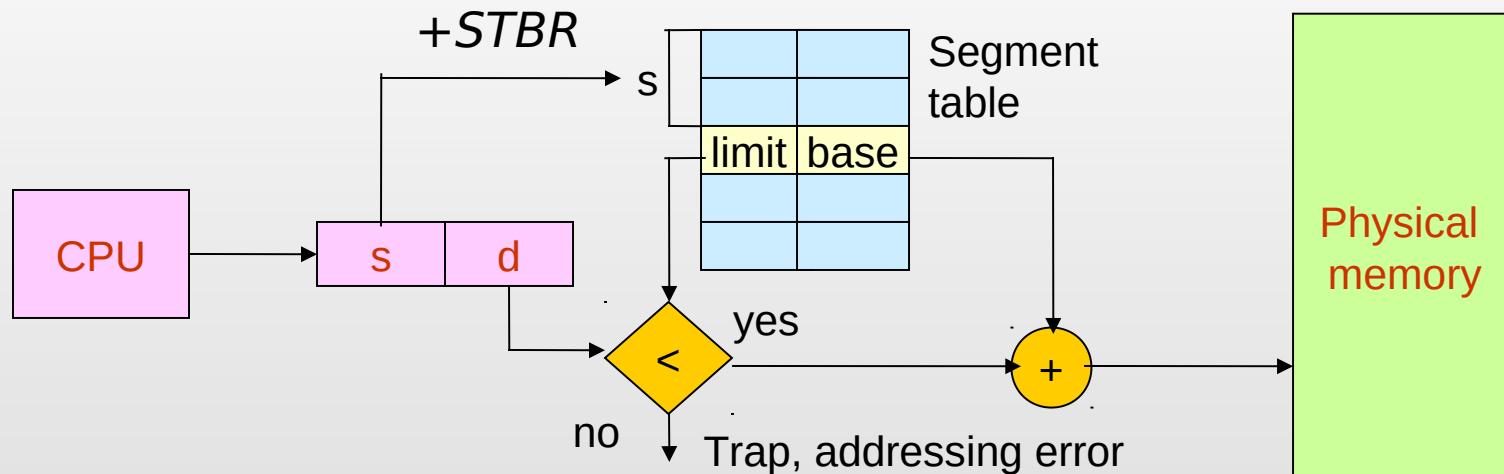
- ✓ Permite mapear la dirección lógica en física. Cada entrada contiene:
 - ◆ *Base: Dirección física de comienzo del segmento*
 - ◆ *Limit: Longitud del Segmento*

Segment-table base register (STBR): apunta a la ubicación de la tabla de segmentos.

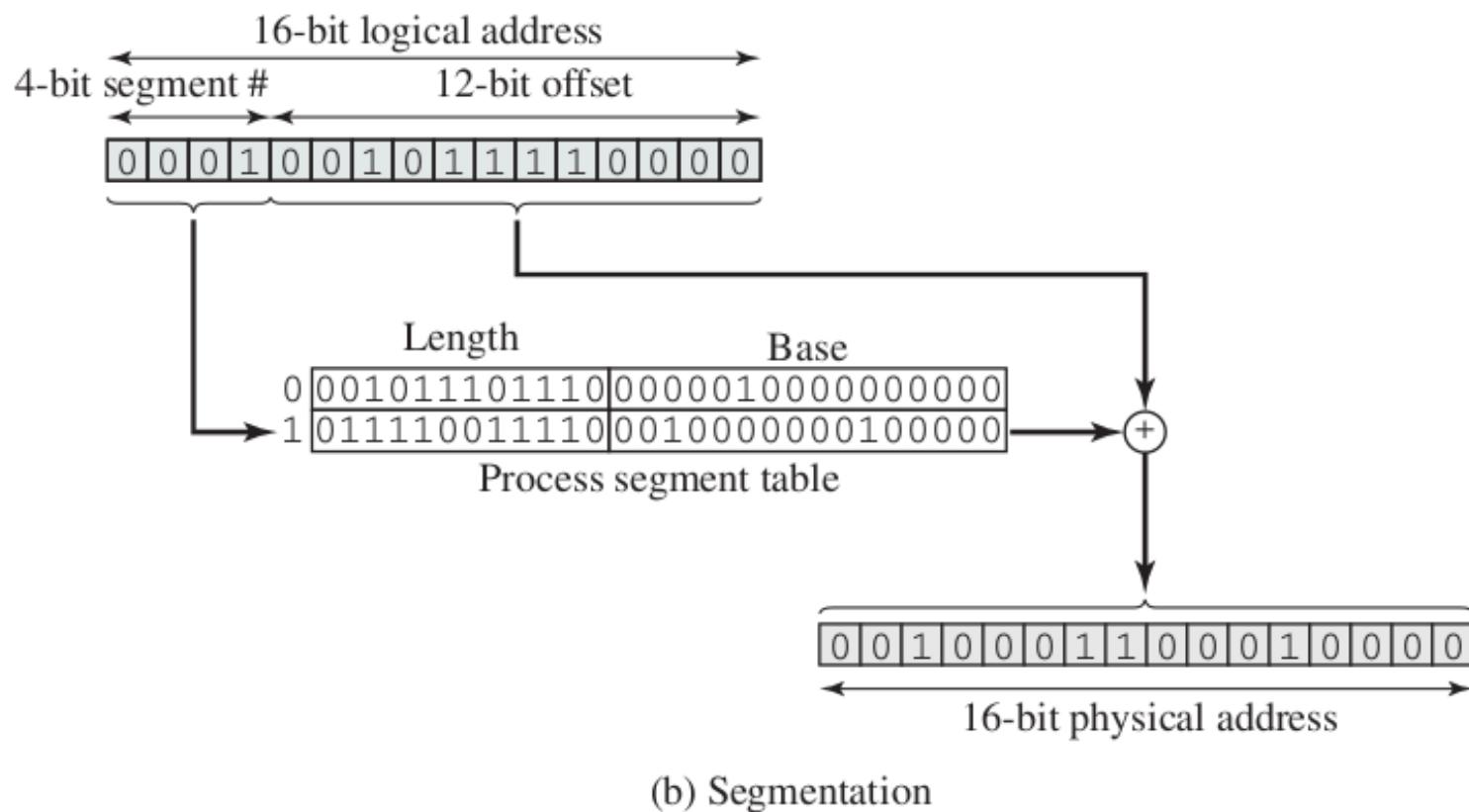
Segment-table length register (STLR) : cantidad de segmentos de un programa



Segmentación (cont.)



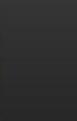
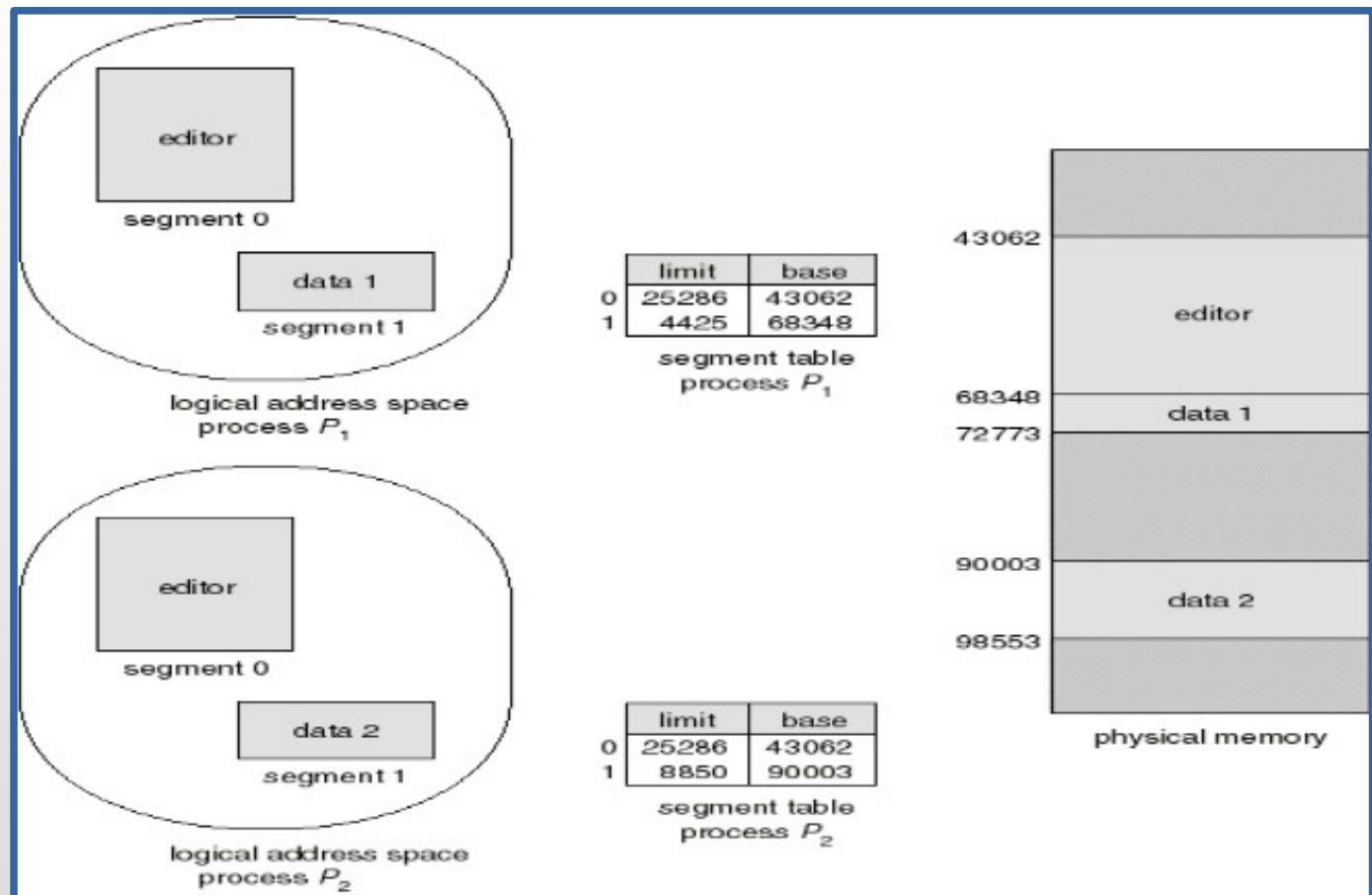
Segmentación - Direcciones (cont.)



Ventajas sobre Página

Compartir

Proteger



Segmentación Paginada

La paginación

- ✓ Transparente al programador
- ✓ Elimina Fragmentación externa.

Segmentación

- ✓ Es visible al programador
- ✓ Facilita modularidad, estructuras de datos grandes y da mejor soporte a la compartición y protección

Segmentación Paginada: Cada segmento es dividido en páginas de tamaño fijo.



Segmentación Paginada (cont.)

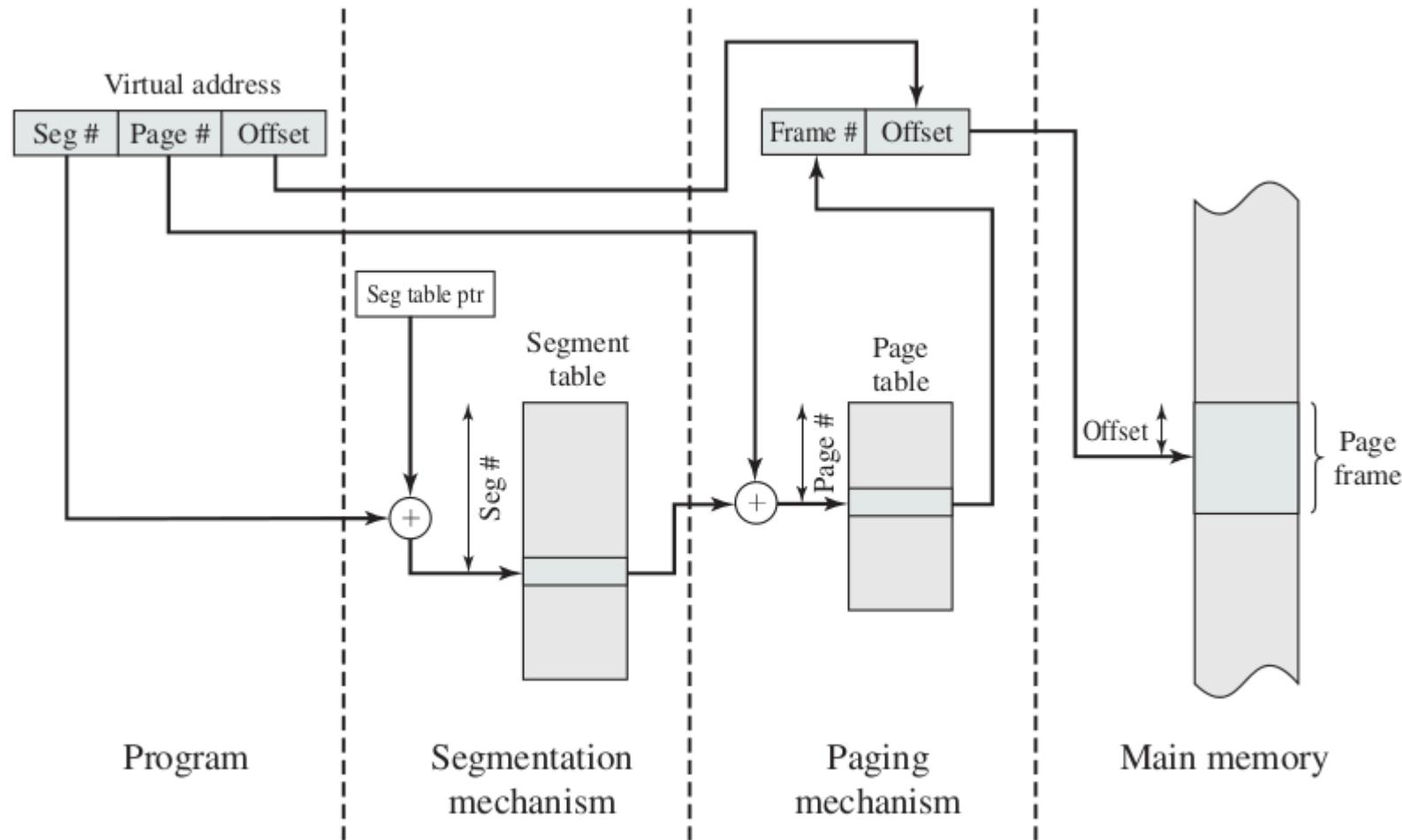
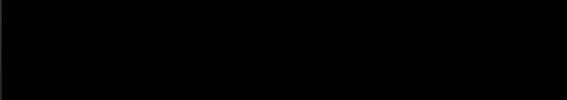
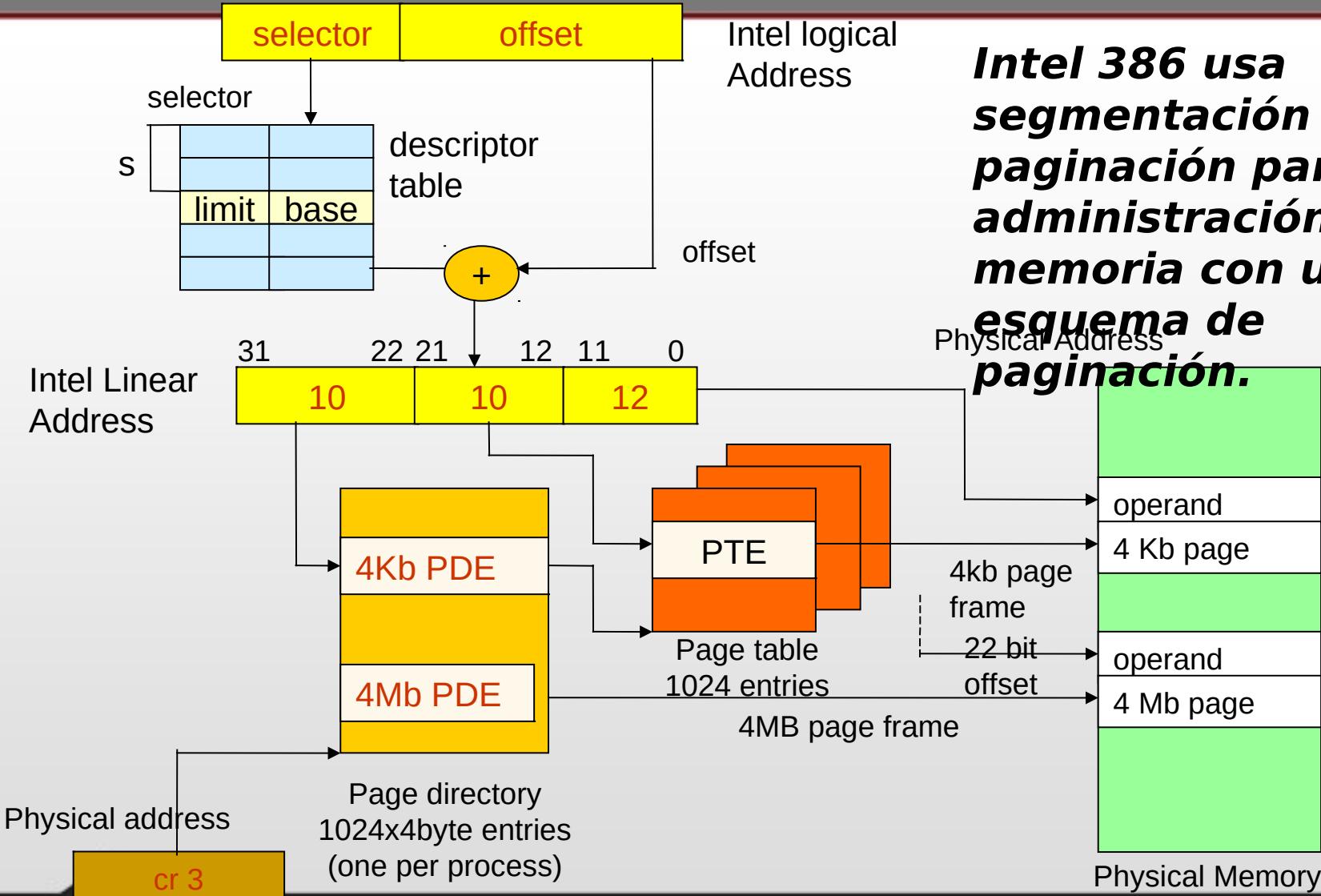


Figure 8.13 Address Translation in a Segmentation/Paging System

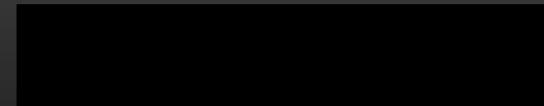


Intel x386



Introducción a los Sistemas Operativos / Conceptos de Sistemas Operativos

Administración de
Memoria - II



Versión: Abril 2020

Palabras Claves: Procesos, Espacio de Direcciones, Memoria, Seguridad, Página, 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 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.
- ✓ 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 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 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.

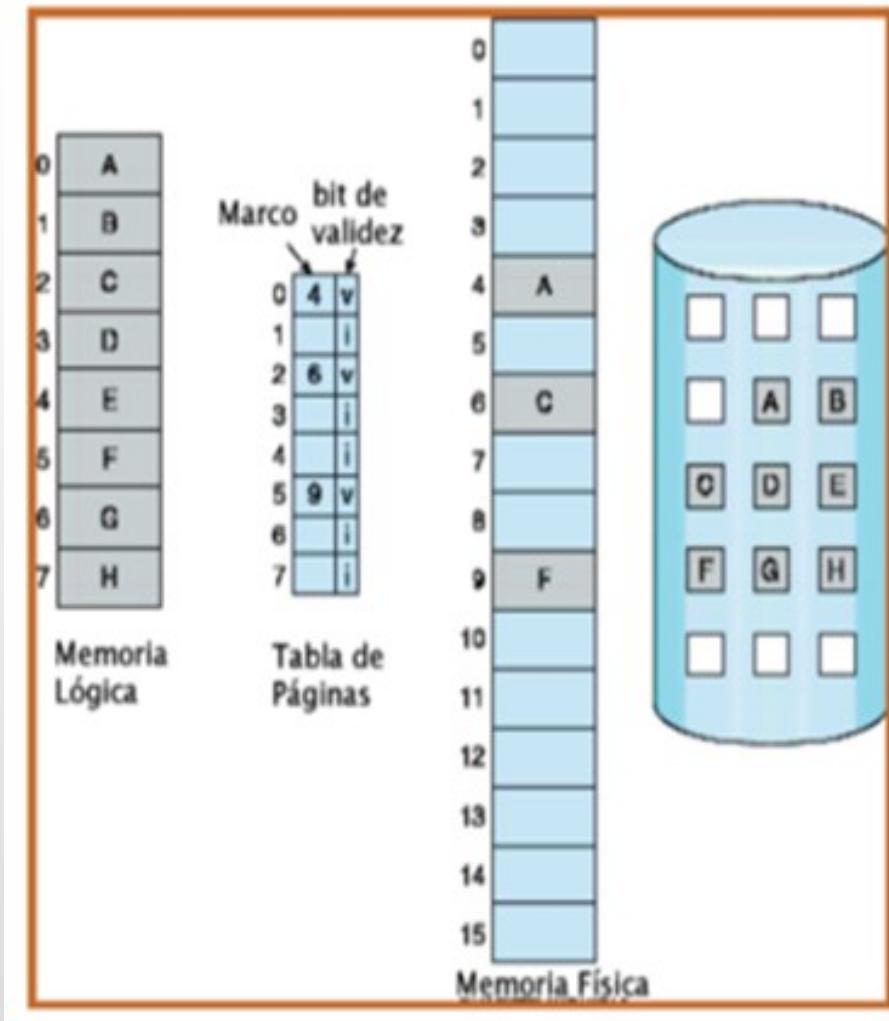


Memoria Virtual con Paginación

- 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
 - ✓ **Bit M:** Indica si la página fue modificada. Si se modificó, en algún momento, se deben reflejar los cambios en Memoria Secundaria



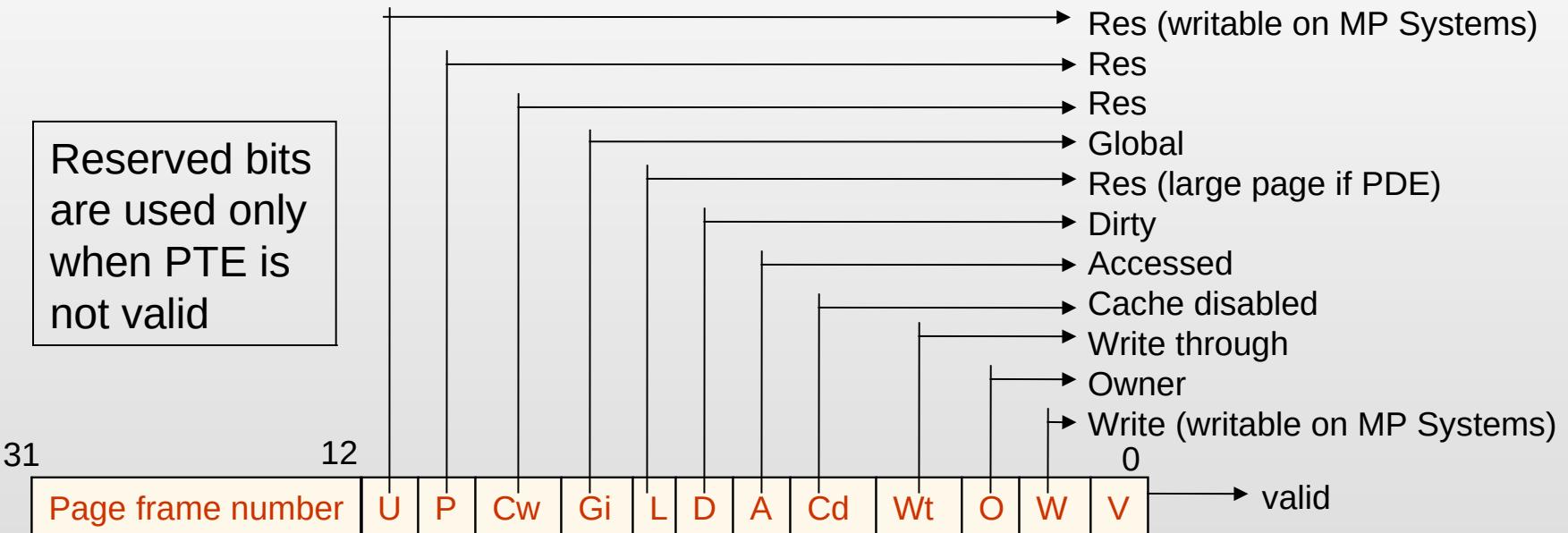
Memoria Virtual con Paginación



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



Fallo de páginas (Page Fault)

- Ocurre 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.

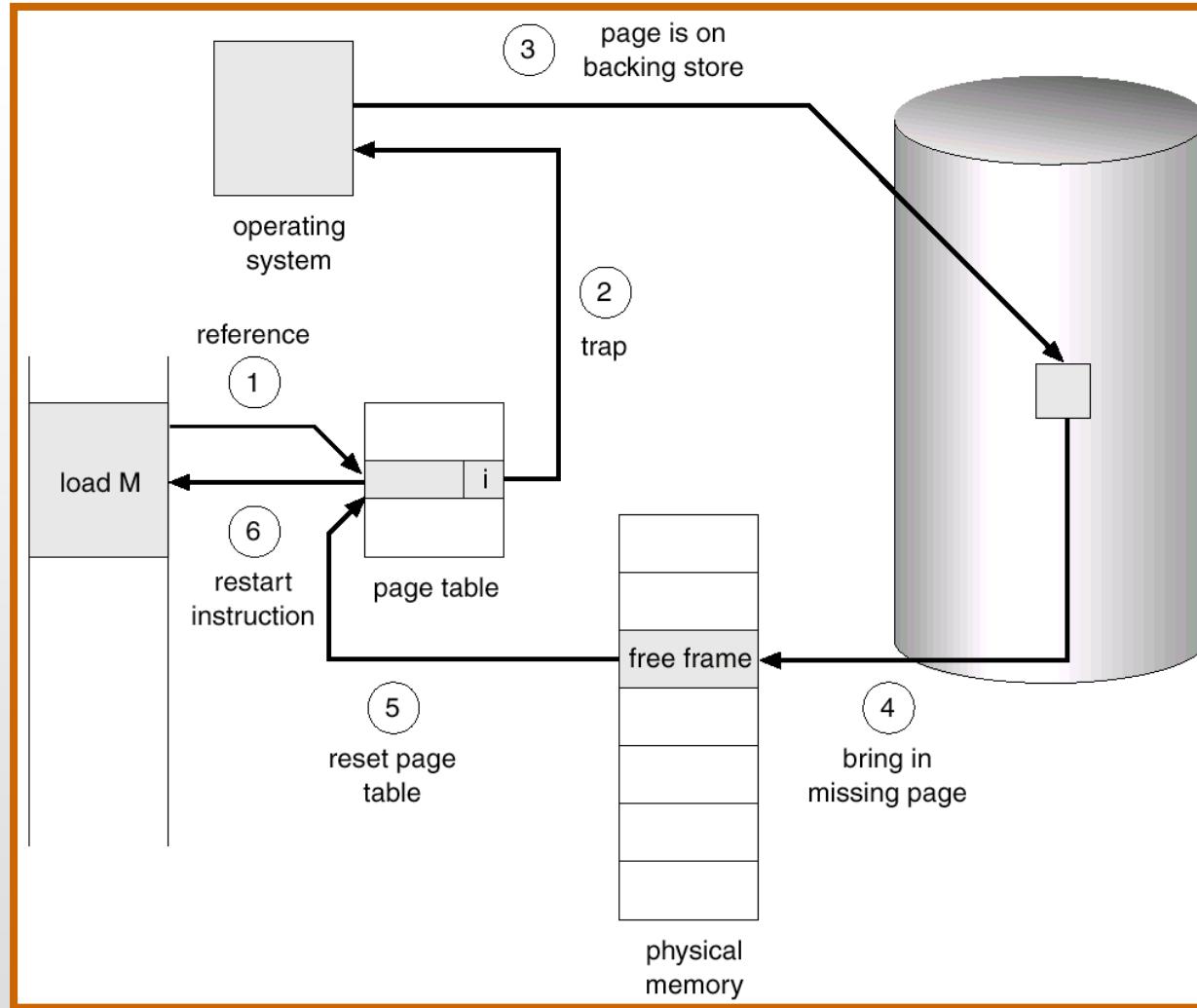


Fallo de páginas (cont.)

- ✓ 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.)



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)

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

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

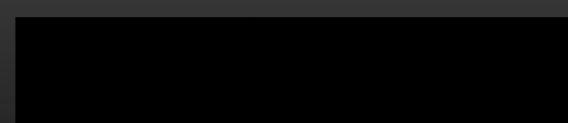


Tabla de 1 nivel - 32 bits

- ✓ Direcciones de 32bit



✓ Ejemplo

- ✓ Cantidad de Page Table Entries (PTEs) máximas que puede tener un proceso = 2^{20}
- ✓ 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 = $2^{20}/2^{10} = 2^{10}$
 - ◆ Tamaño tabla de páginas del proceso:
 $2^{10} * 4\text{bytes} = \textbf{4MB por proceso}$



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 = $2^{42} * 4\text{bytes} = 2^{54}$
- Más de 16.000GB por proceso!!!**



Tabla de páginas - Tabla de 2 niveles

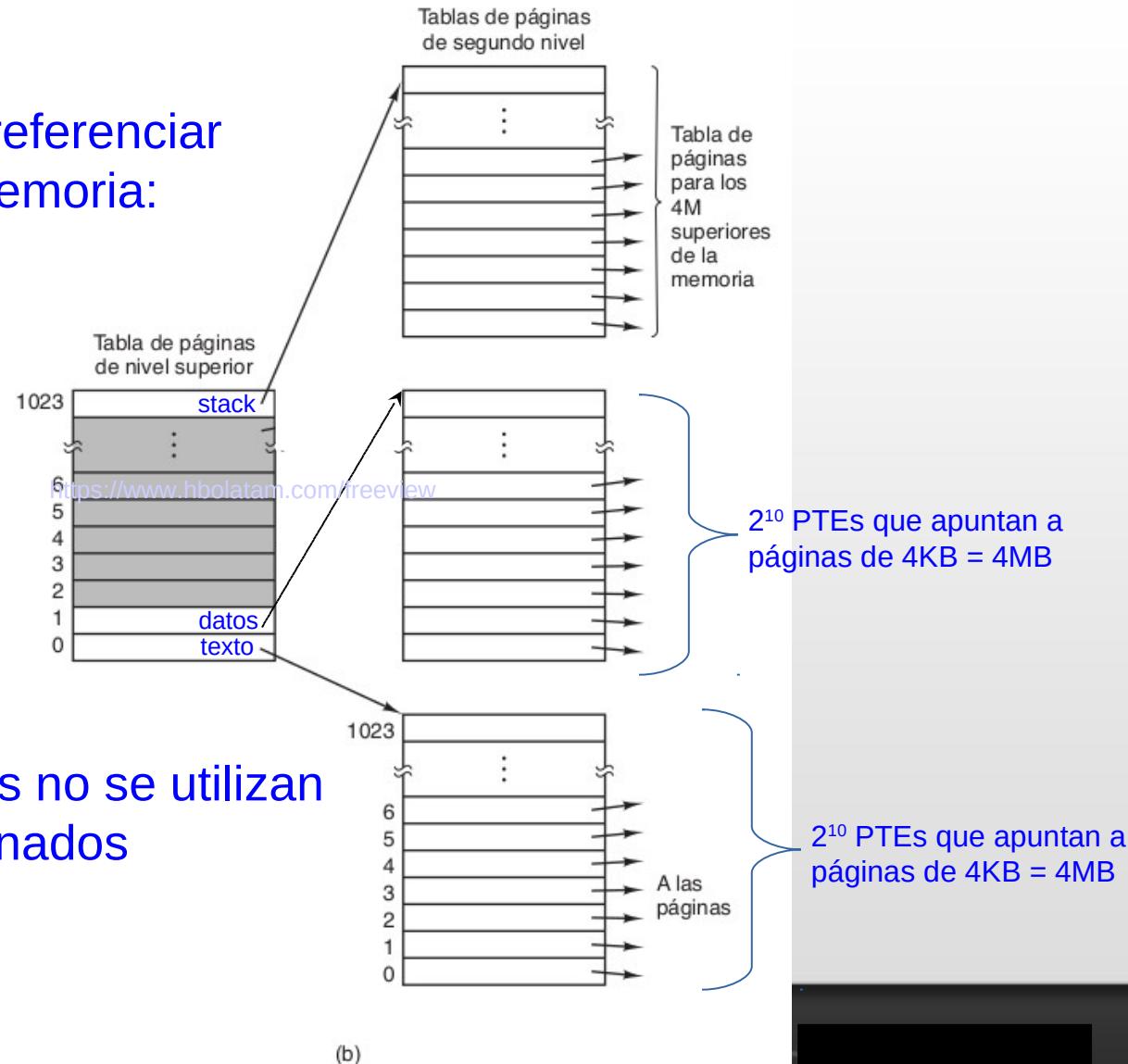
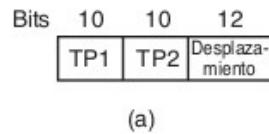
- El propósito de la tabla de páginas multinivel es dividir la tabla de páginas lineal en múltiples tablas de páginas
- Cada tabla de páginas suele tener el mismo tamaño pero se busca que tengan un menor número de páginas por tabla
- La idea general es que cada tabla sea más pequeña
- Se busca que la 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:

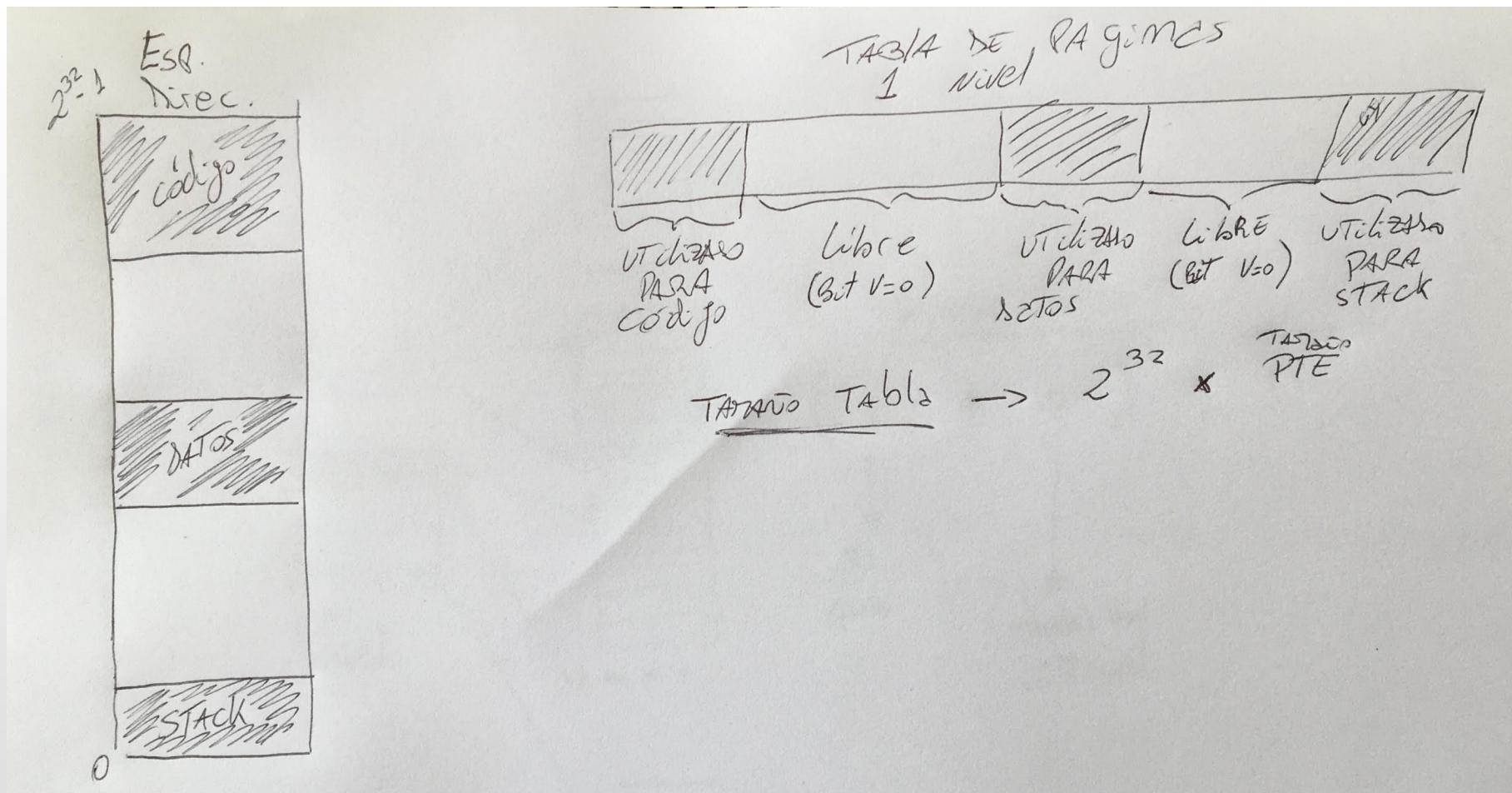
- 4MB de datos
- 4MB de texto
- 4MB de stack



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.

1 nivel vs. 2 niveles (1)



1 nivel vs. 2 niveles (2)

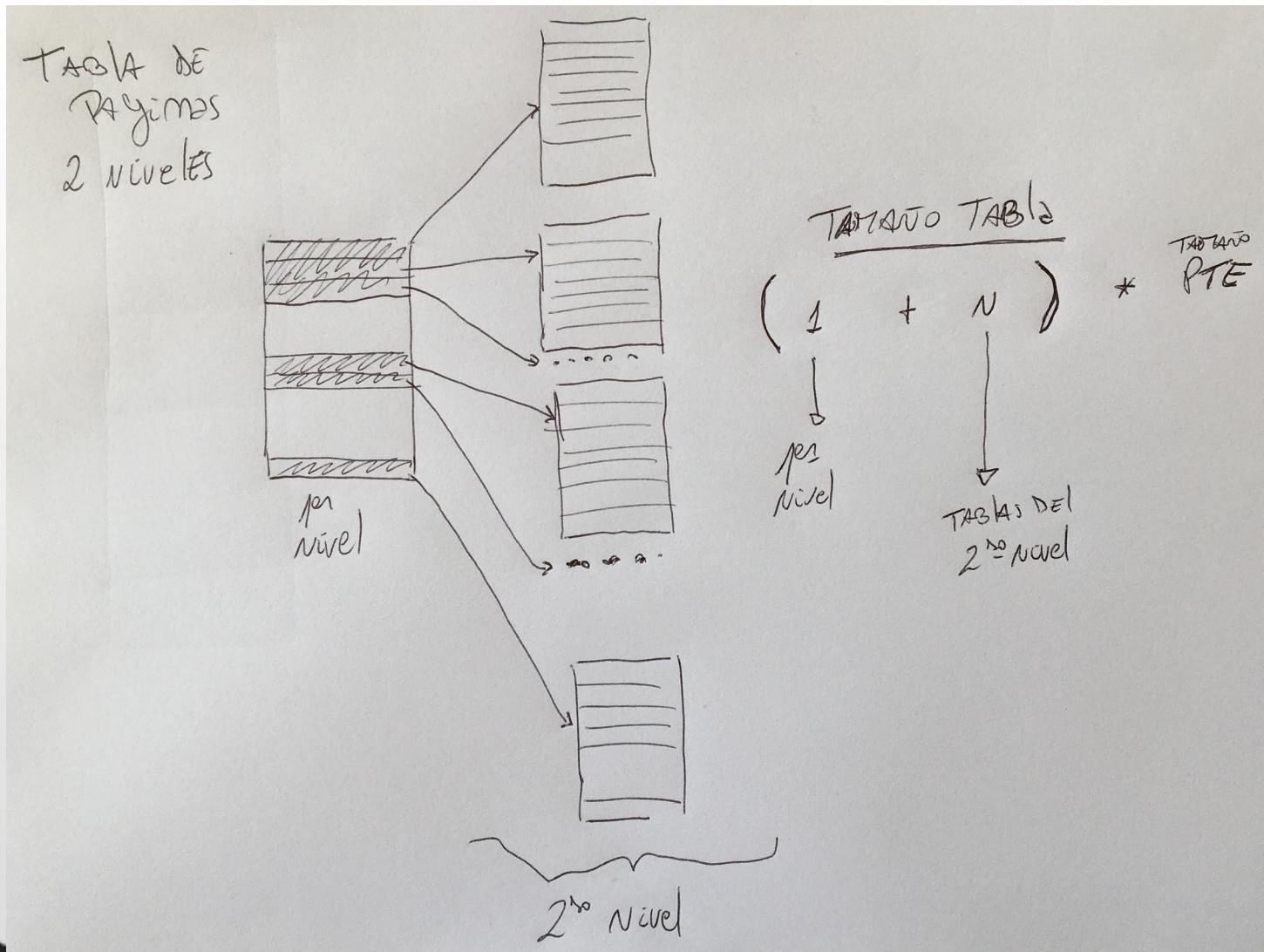
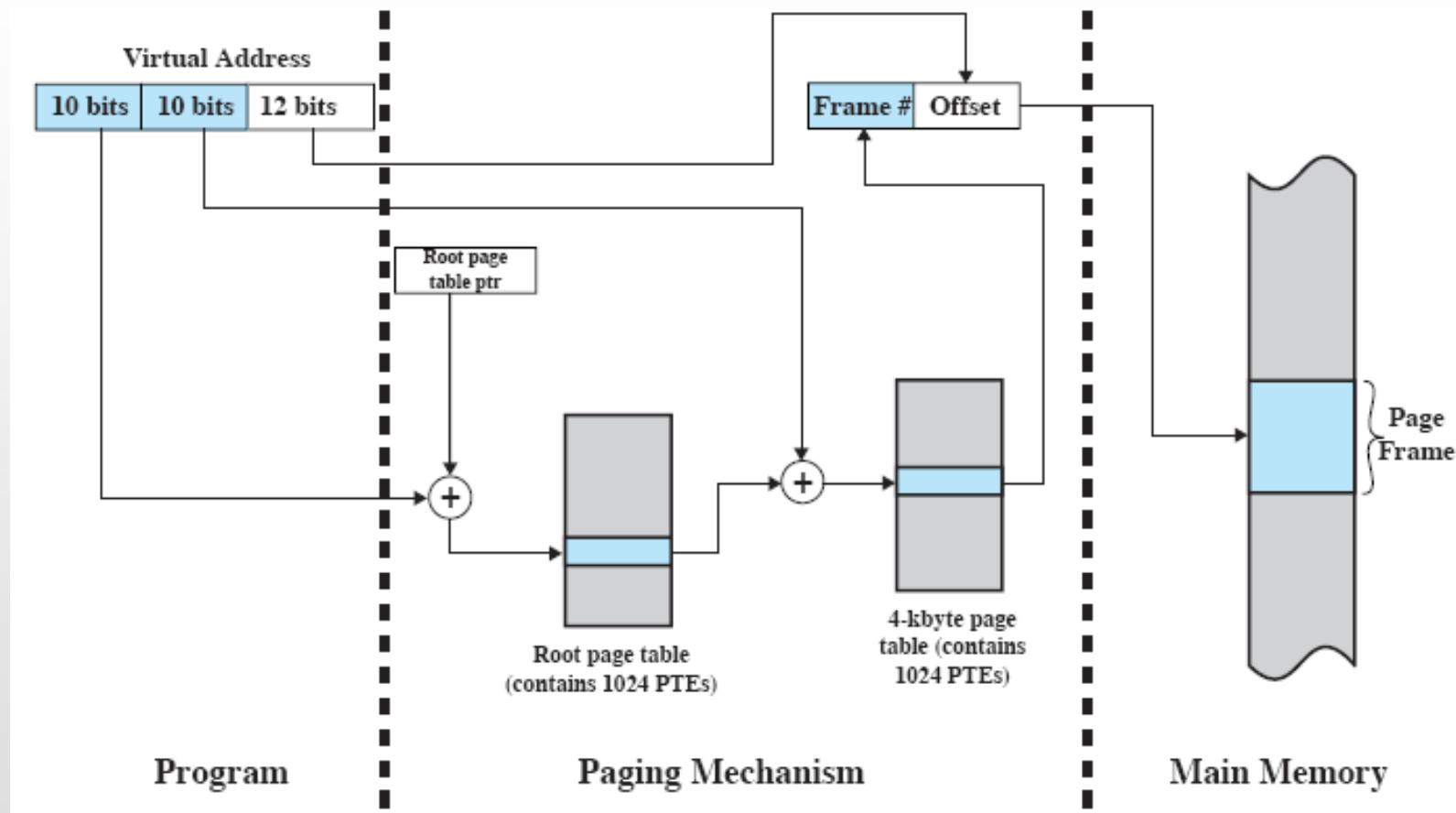
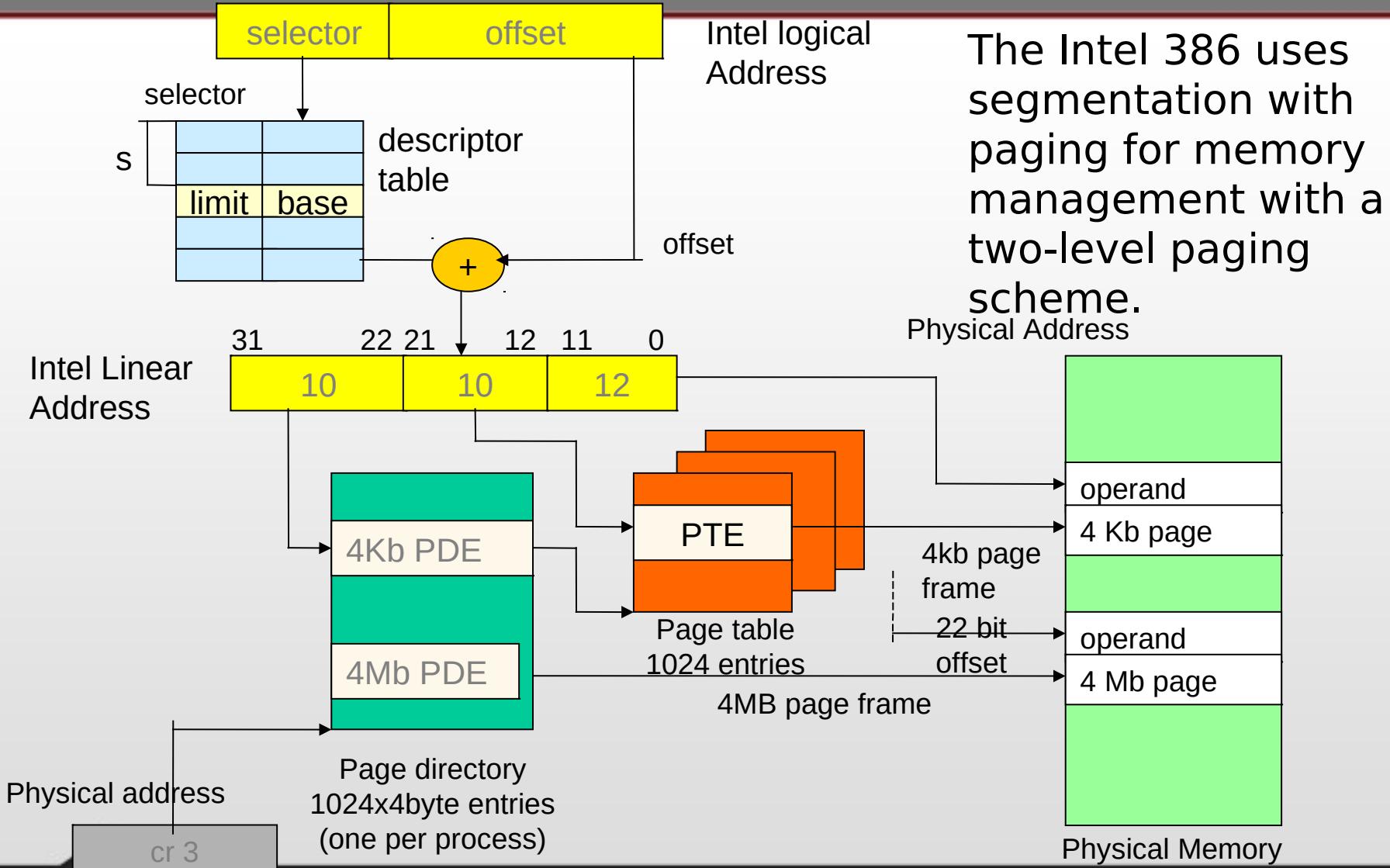


Tabla de Páginas - Tabla de 2 niveles



Intel 30386



Tablas de Páginas - x64

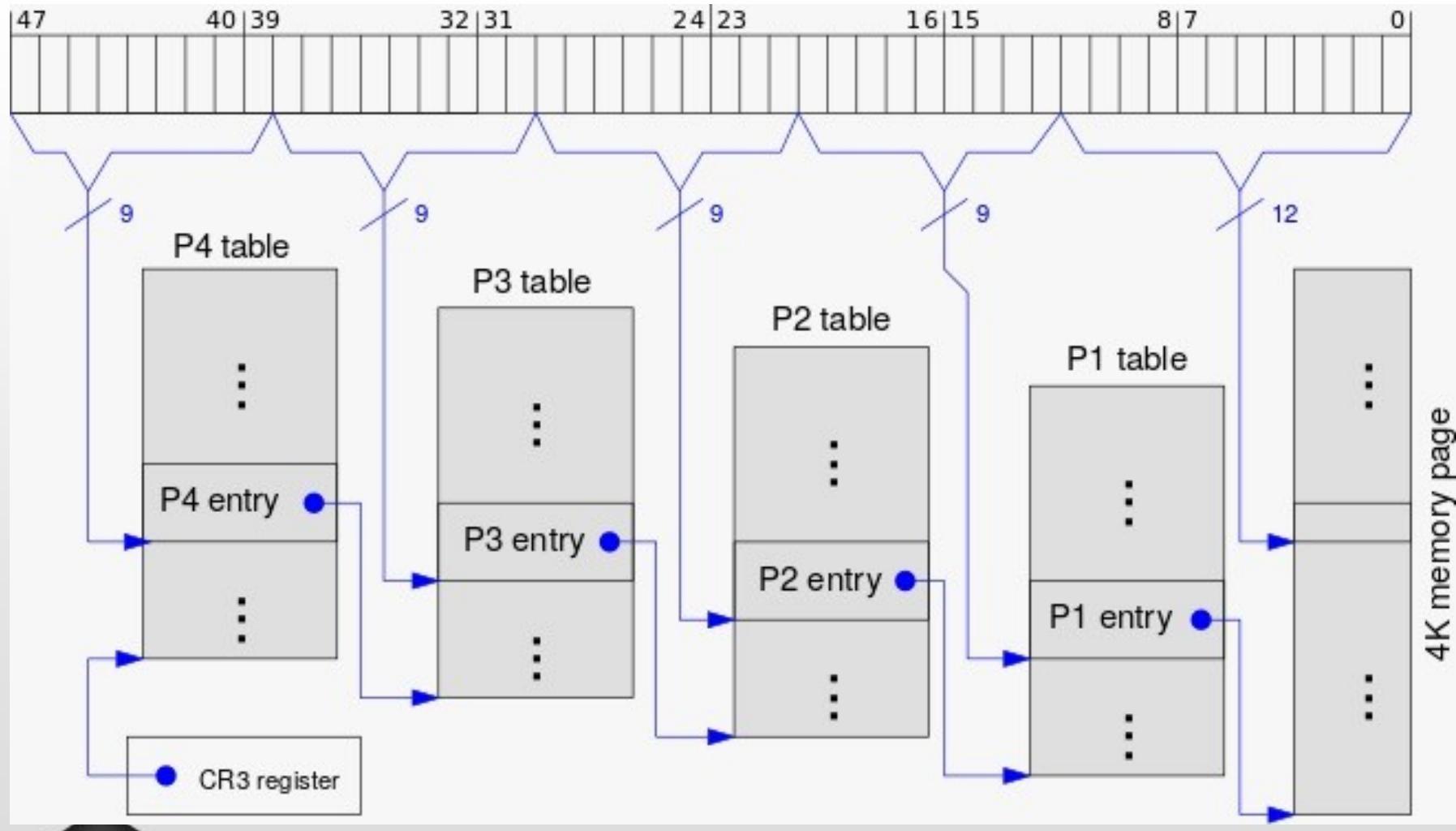


Tabla de Paginas (cont.) - Tabla invertida

- Utilizada en Arquitecturas donde el espacio de direcciones es muy grande
 - ✓ Las tablas de paginas 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 Gigabytes (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 refiera 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

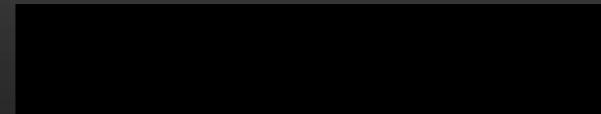
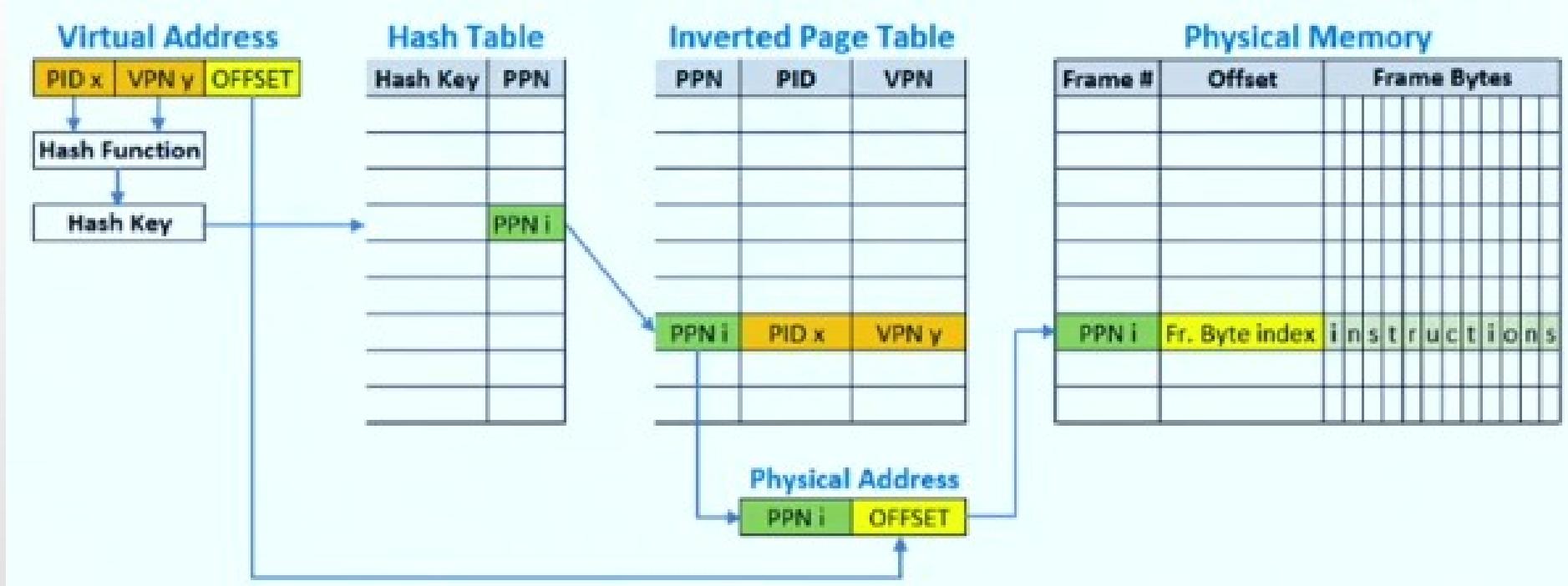


Tabla de Paginas (cont.) - Tabla invertida



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



Tamaño de la Pagina

Pequeño

- ✓ Menor Fragmentación Interna.
- ✓ Más paginas requeridas por proceso → Tablas de páginas 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.)

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

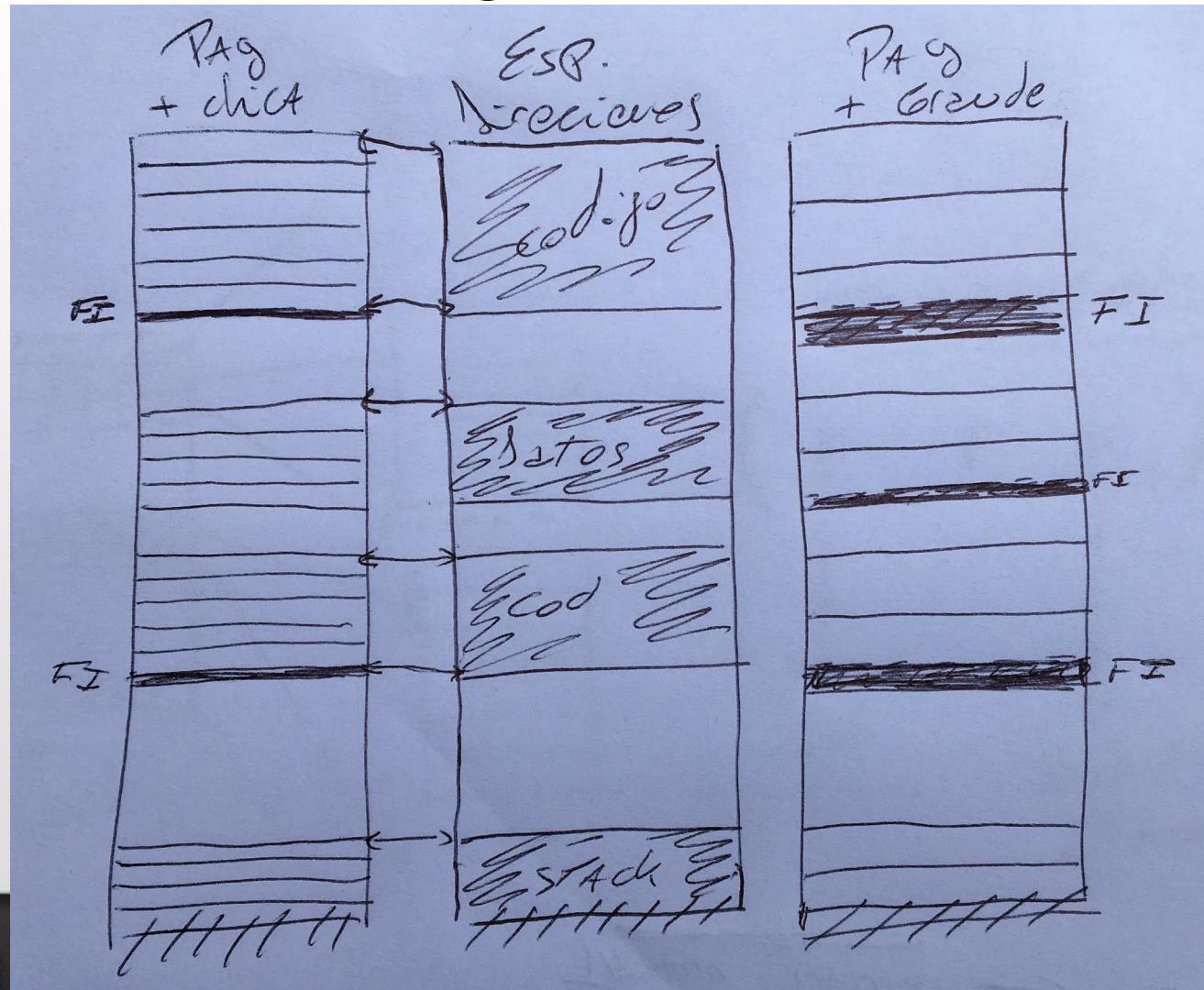
Pagina de 1024 bytes

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



Tamaño de la Pagina (cont.)

Relación con la fragmentación interna



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 páginas
 - ✓ 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.)

- 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 (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

Translation Lookaside Buffer (cont.)

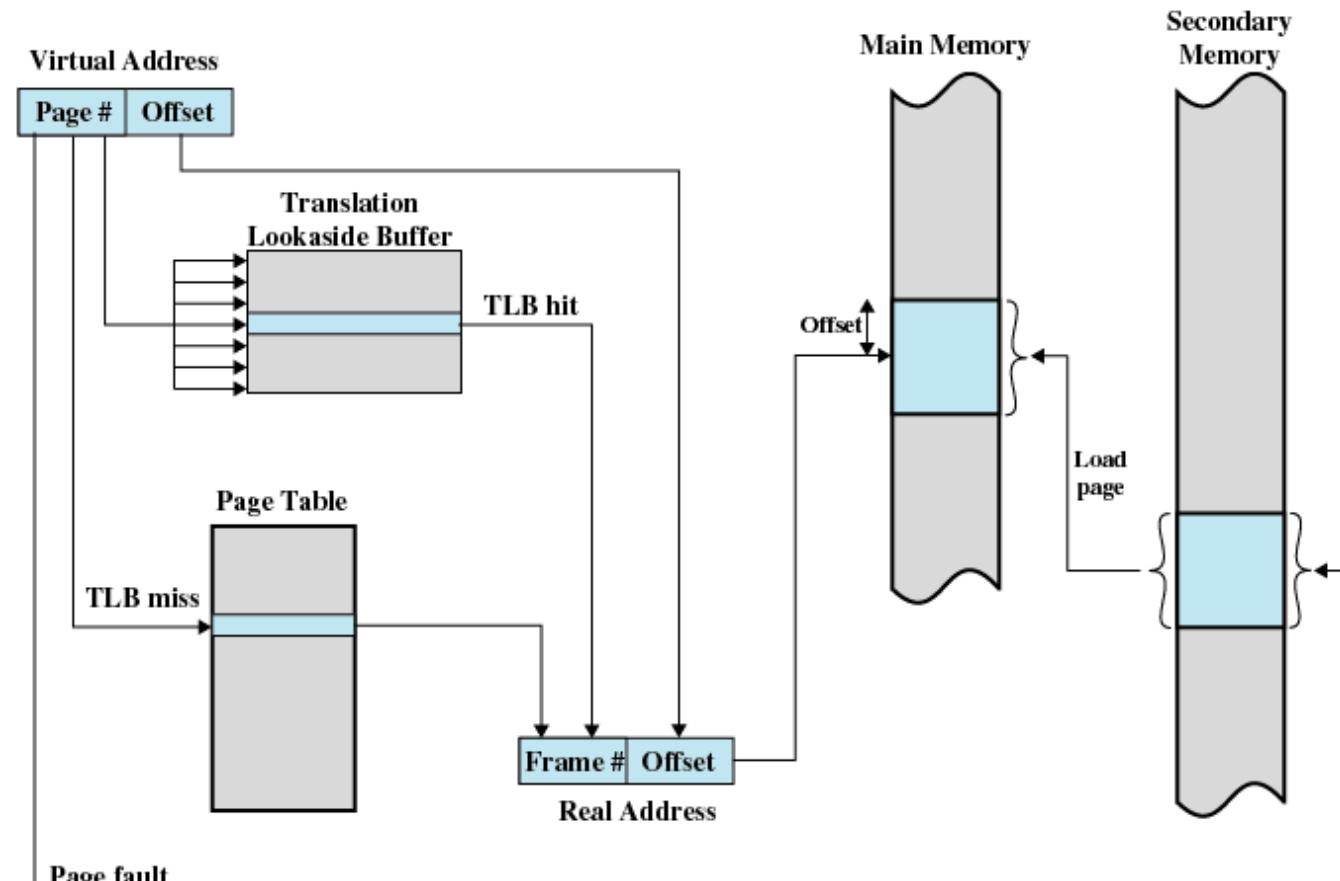


Figure 8.7 Use of a Translation Lookaside Buffer



Translation Lookaside Buffer (cont.)

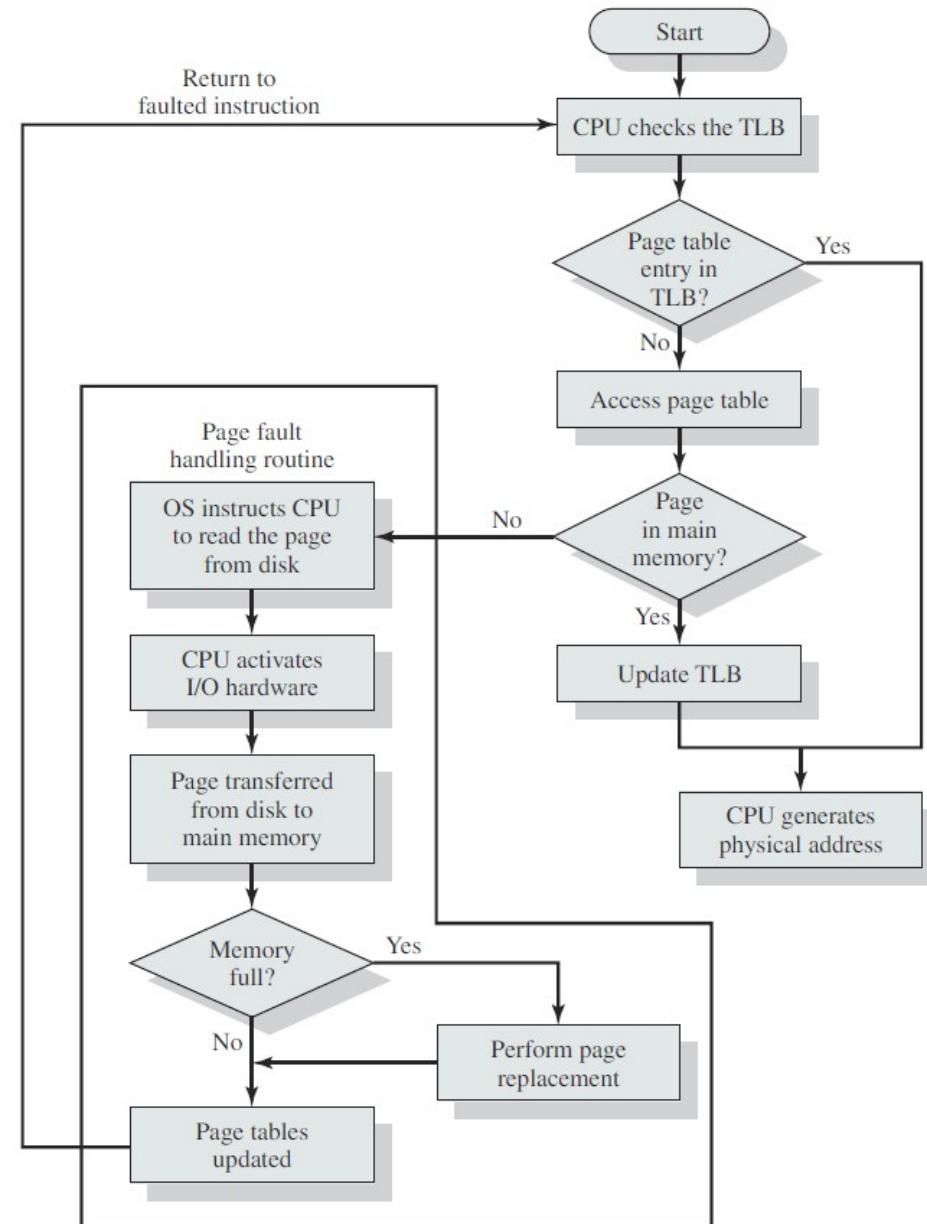


Figure 8.8 Operation of Paging and Translation Lookaside Buffer (TLB)

Políticas en el manejo de MV

Table 8.4 Operating System Policies for Virtual Memory

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



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 = 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”
- ¿Cuál sería Reemplazo Óptimo?
 - ✓ 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.



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 menos recientemente accedidas

2da. Chance: Un avance del FIFO tradicional que beneficia a las páginas mas referenciadas

NRU (Non Recently Used):

- ✓ Utiliza bits R y M

- ✓ Favorece a las páginas que fueron usadas recientemente



Introducción a los Sistemas Operativos / Conceptos de Sistemas Operativos

Administración de
Memoria - III



Versión: Septiembre 2017

Palabras Claves: Procesos, Espacio de Direcciones, Memoria, Paginación, Trashing, Working Set

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.



Thrashing (hiperpaginación)

Concepto: decimos que un sistema está en *thrashing* cuando pasa más tiempo paginando que ejecutando procesos.

Como consecuencia, hay una baja importante de performance en el sistema.

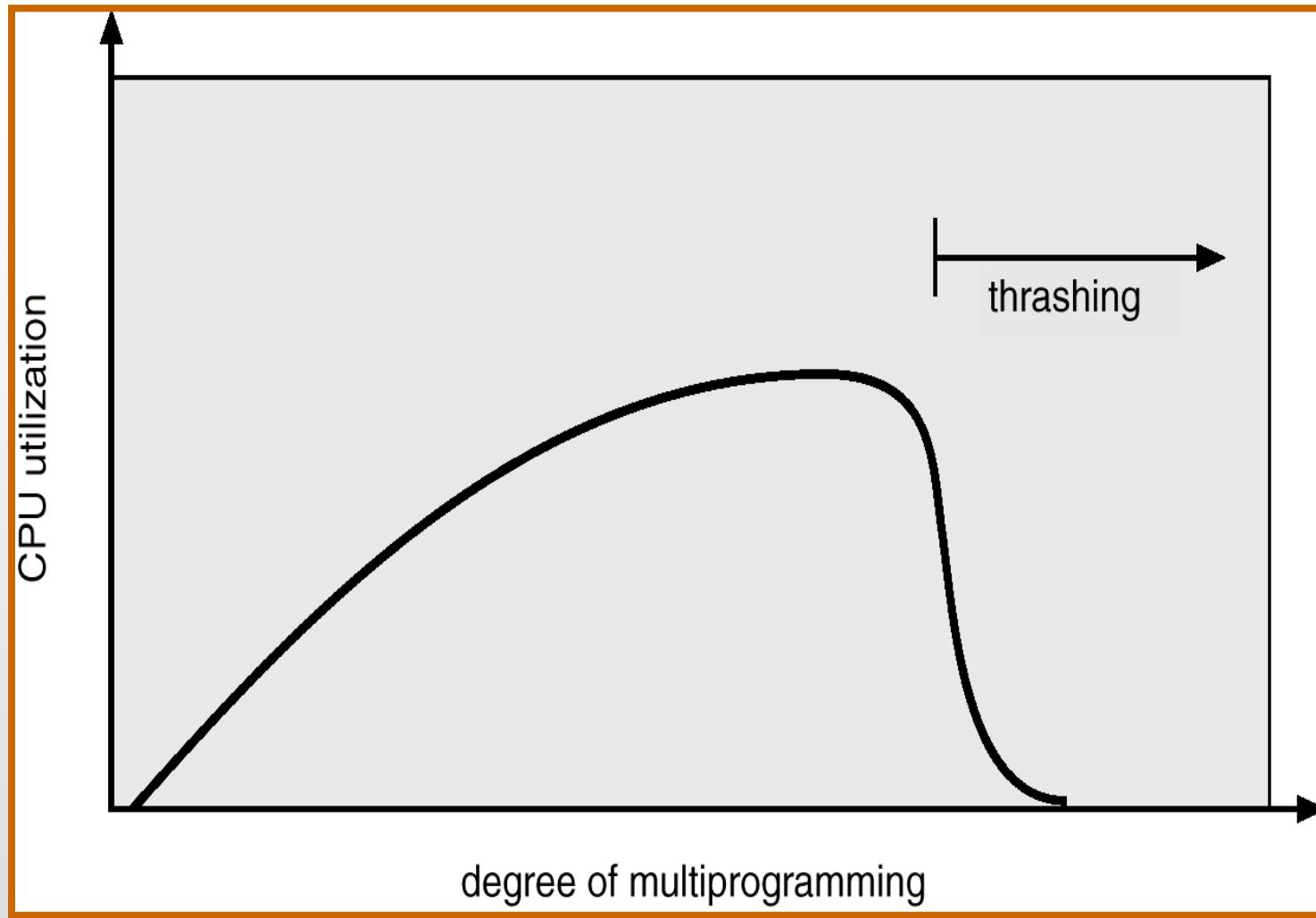


Ciclo del thrashing

- 1) El kernel monitorea el uso de la CPU.
- 2) Si hay baja utilización ⇒ aumenta el grado de multiprogramación.
- 3) Si el algoritmo de reemplazo es global, pueden sacarse frames a otros procesos.
- 4) Un proceso necesita más frames.
Comienzan los page-faults y robo de frames a otros procesos.
- 5) Por swapping de páginas, y encolamiento en dispositivos, baja el uso de la CPU.
- 6) Vuelve a 1).



Thrashing



El scheduler de CPU y el thrashing

- 1) Cuando se decrementa el uso de la CPU, el scheduler long term aumenta el grado de multiprogramación.
- 2) El nuevo proceso inicia nuevos page-faults, y por lo tanto, más actividad de paginado.
- 3) Se decrementa el uso de la CPU
- 4) Vuelve a 1).



Control del thrashing

- Se puede limitar el thrashing usando algoritmos de reemplazo local.
- Con este algoritmo, si un proceso entra en thrashing no roba frames a otros procesos.
- Si bien perjudica la performance del sistema, es controlable.



Conclusión sobre thrashing

- ✓ Si un proceso cuenta con todos los frames que necesita, no habría thrashing.
- ✓ Una manera de abordar esta problemática es utilizando la estrategia de **Working Set**, la cual se apoya en el modelo de localidad
- ✓ Otra estrategia con el mismo espíritu es la del algoritmo **PFF** (Frecuencia de Fallos de Página)

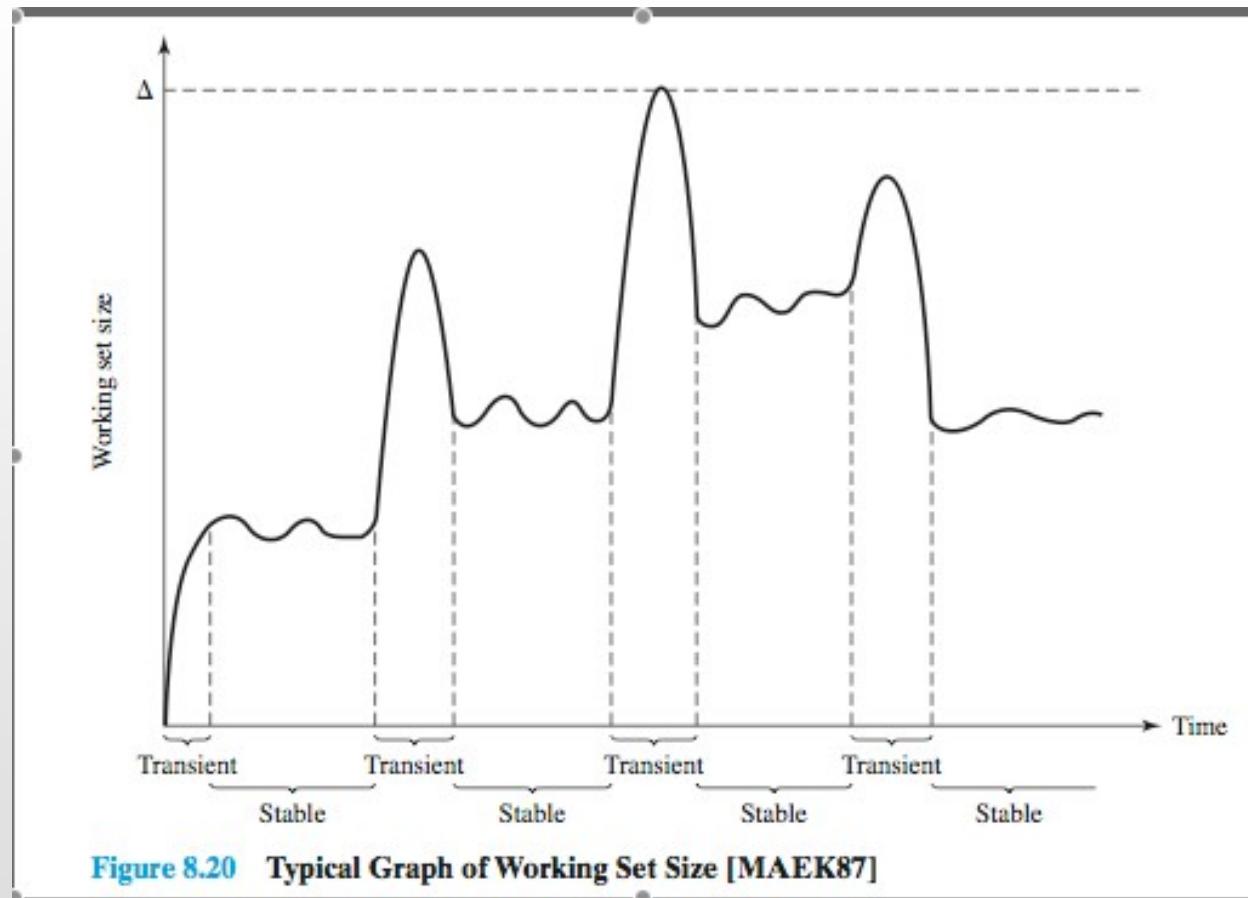


El modelo de localidad

- Cercanía de referencias o principio de cercanía
- Las referencias a datos y programa dentro de un proceso tienden a agruparse
- La localidad de un proceso en un momento dado se da por el conjunto de páginas que tiene en memoria en ese momento.
- En cortos períodos de tiempo, el proceso necesitará pocas “piezas” del proceso (por ejemplo, una página de instrucciones y otra de datos...)



El modelo de localidad



El modelo de localidad (cont.)

- Un programa se compone de varias localidades.
- Ejemplo: Cada rutina será una nueva localidad: se mencionan sus direcciones (cerca) cuando se está ejecutando.
- Para prevenir la hiperactividad, un proceso debe tener en memoria sus páginas más activas (menos page faults).



El modelo de working set

- Se basa en el modelo de localidad.
- Ventana del working set (Δ): las referencias de memoria más recientes.
- Working set: es el conjunto de páginas que tienen las más recientes Δ referencias a páginas.

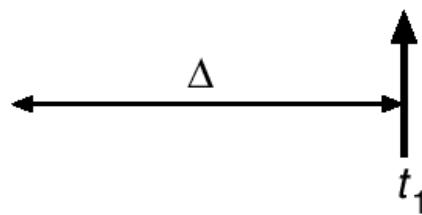


Ejemplo de working set

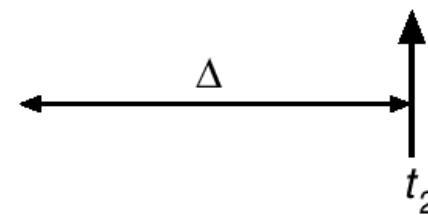
$\Delta=10$

page reference table

... 2 6 1 5 7 7 7 5 1 6 2 3 4 1 2 3 4 4 4 3 4 3 4 4 4 1 3 2 3 4 4 4 3 4 4 4 ...



$$WS(t_1) = \{1, 2, 5, 6, 7\}$$



$$WS(t_2) = \{3, 4\}$$



La selección del Δ

- Δ chico: no cubrirá la localidad
- Δ grande: puede tomar varias localidades



Medida del working set

- $m = \text{cantidad frames disponibles}$
- $WSS_i = \text{medida del working set del proceso } p_i.$
- $\sum WSS_i = D;$
- $D = \text{demanda total de frames.}$
- Si $D > m$, habrá ***thrashing***.



Prevención del thrashing

- SO monitorea c/ proceso, dándole tantos frames hasta su WSS_i (medida del working set del proceso p_i)
- Si quedan frames, puede iniciar otro proceso.
- Si D crece, excediendo m, se elige un proceso para suspender, reasignándose sus frames...

Así, se mantiene alto el grado de multiprogramación optimizando el uso de la CPU.



Problema del modelo del WS

- Mantener un registro de los WSS_i (medida del working set del proceso p_i)
- La ventana es móvil

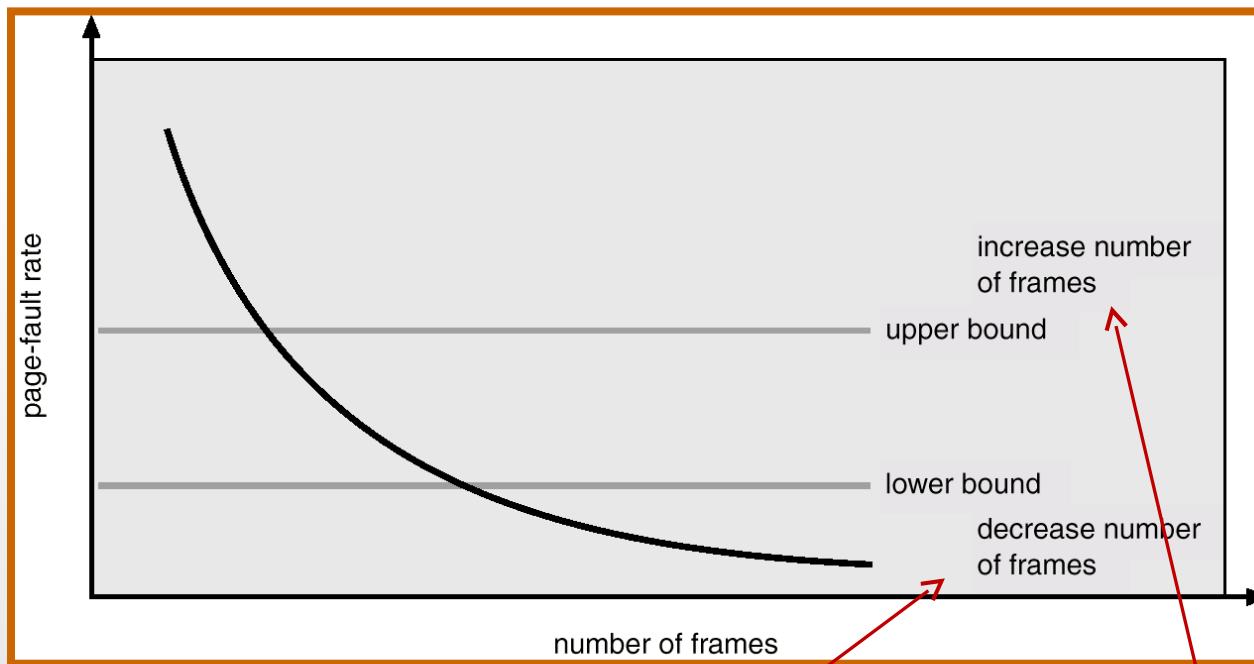


Prevención del thrashing por PFF

- La técnica PFF (Page Fault Frequency o Frecuencia de Fallo de Páginas), en lugar de calcular el WS de los procesos, utiliza la tasa de fallos de página para estimar si el proceso tiene un conjunto residente que representa adecuadamente al WS.
- PFF: Frecuencia de page faults
- PFF alta ⇒ Se necesitan más frames
- PFF baja ⇒ Los procesos tienen muchos frames asignados



Esquema de PFF



- Establecer tasa de PF aceptable
 - ✓ Si la tasa actual es baja, el proceso pierde frames
 - ✓ Si la tasa actual es alta, el proceso gana frames.



PFF (continuación)

- Establecer límites superior e inferior de las PFF's deseadas.
- Si se Excede PFF máx. ⇒ Se le asigna un frame mas al proceso, ya que el mismo genera muchos PF y probablemente lo esté necesitando.
- Si la PFF está por debajo del mínimo ⇒ Se le quita un frame al proceso, asumiendo que tiene de más
- Puede llegar a suspender un proceso si no hay más frames. Sus frames se reasignan a procesos de alta PFF.



PFF y estructura de un programa

✓ Ejemplo:

✓ **int A[][] = new int[1024][1024];**

✓ Cada fila se almacena en una página

✓ Programa 1:

```
for (j = 0; j < A.length; j++)
    for (i = 0; i < A.length; i++)
        A[i,j] = 0;
```

1024 x 1024 page faults

✓ Programa 2:

```
for (i = 0; i < A.length; i++)
    for (j = 0; j < A.length; j++)
        A[i,j] = 0;
```

1024 page faults



Demonio de Paginación

- Proceso creado por el SO durante el arranque que apoya a la administración de la memoria
- Se ejecuta cuando el sistema tiene una baja utilización o algún parámetro de la memoria lo indica
 - ✓ Poca memoria libre
 - ✓ Mucha memoria modificada
- Tareas:
 - ✓ Limpiar páginas modificadas sincronizándolas con el swap
 - ✓ Reducir el tiempo de swap posterior ya que las páginas están “limpias”
 - ✓ Reducir el tiempo de transferencia al sincronizar varias páginas contiguas.
 - ✓ Mantener un cierto número de marcos libres en el sistema.
 - ✓ Demorar la liberación de una página hasta que haga falta realmente



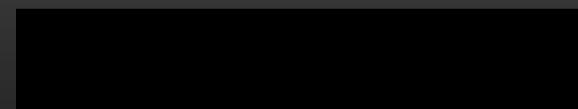
Demonio de Paginación - Ejemplos

En Linux

- ✓ Proceso “**kswapd**”

En Windows

- ✓ Proceso “**system**”



Memoria Compartida

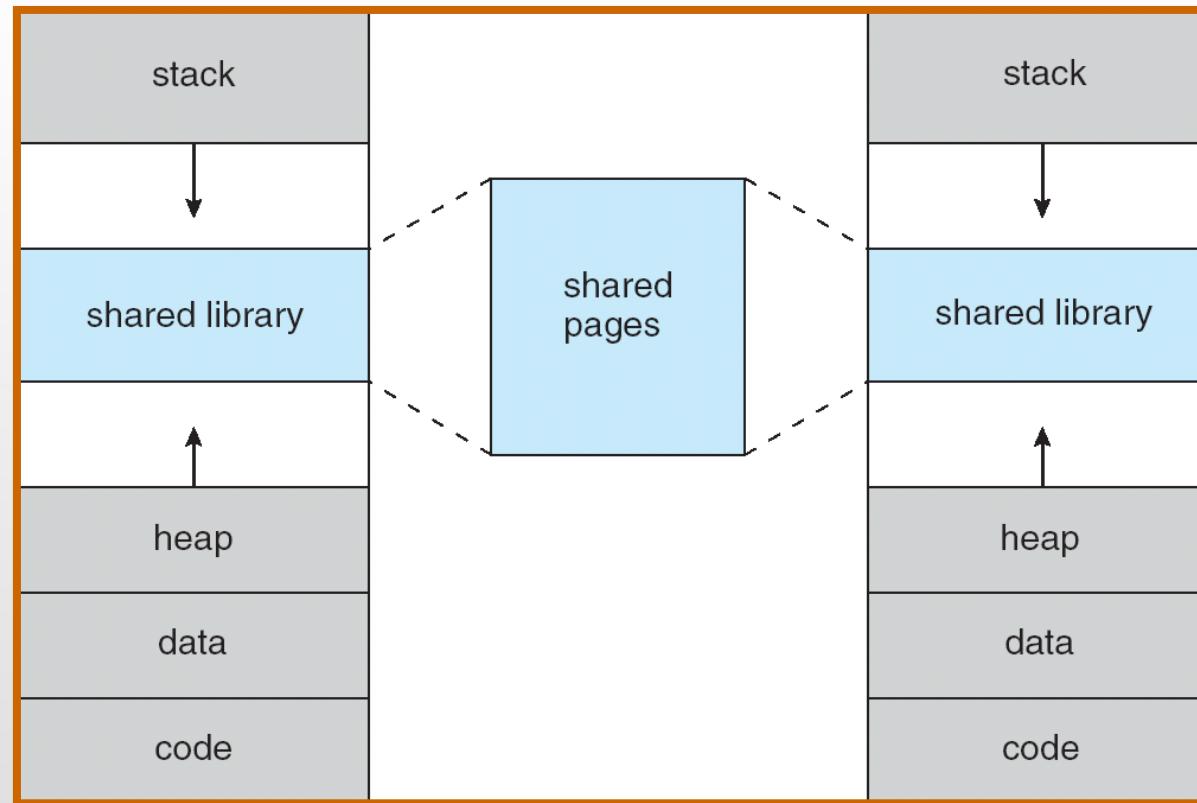
- ✓ Gracias al uso de la tabla de páginas varios procesos pueden compartir un marco de memoria; para ello ese marco debe estar asociado a una página en la tabla de páginas de cada proceso
- ✓ El número de página asociado al marco puede ser diferente en cada proceso

✓ Código compartido

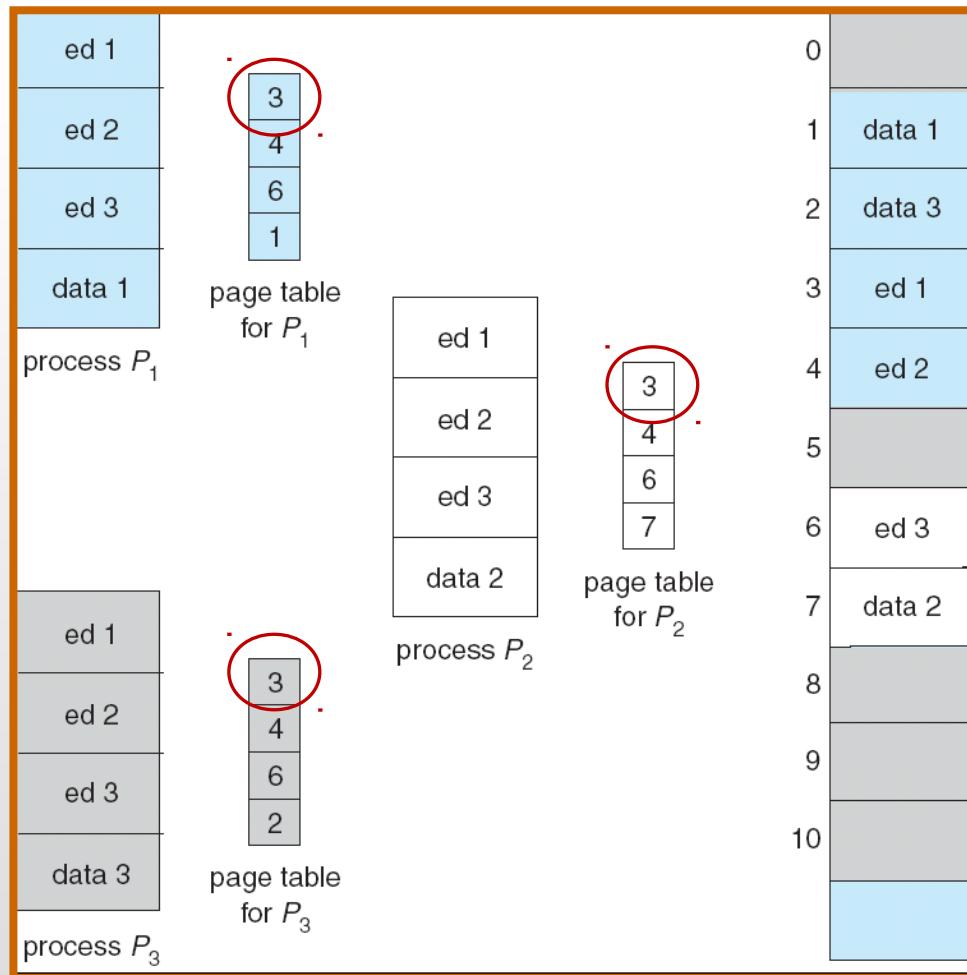
- ✓ Los procesos comparten una copia de código (sólo lectura) por ej. editores de texto, compiladores, etc
- ✓ Los datos son privados a cada proceso y se encuentran en páginas no compartidas



Memoria Compartida (cont.)



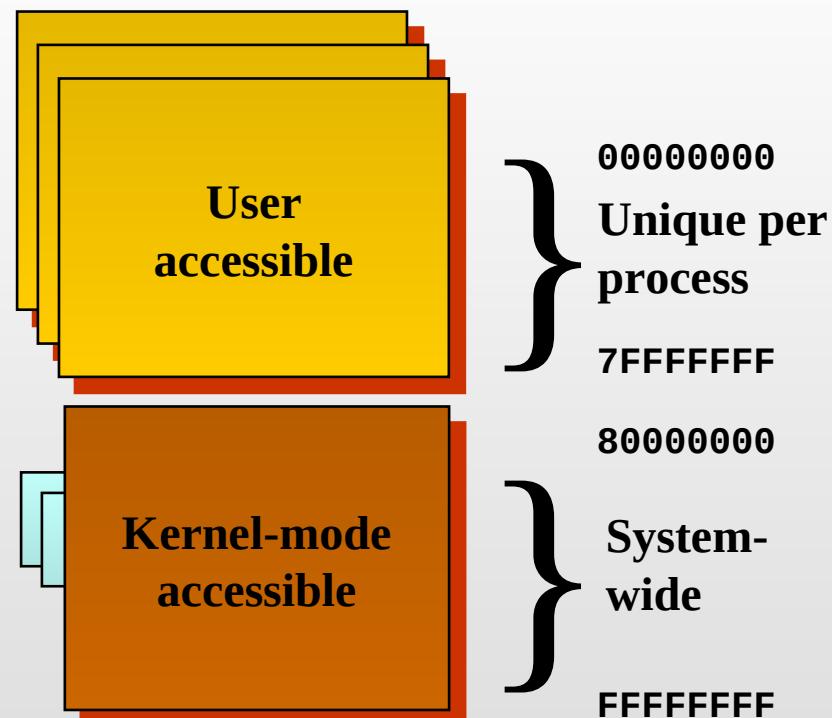
Memoria Compartida (cont.)



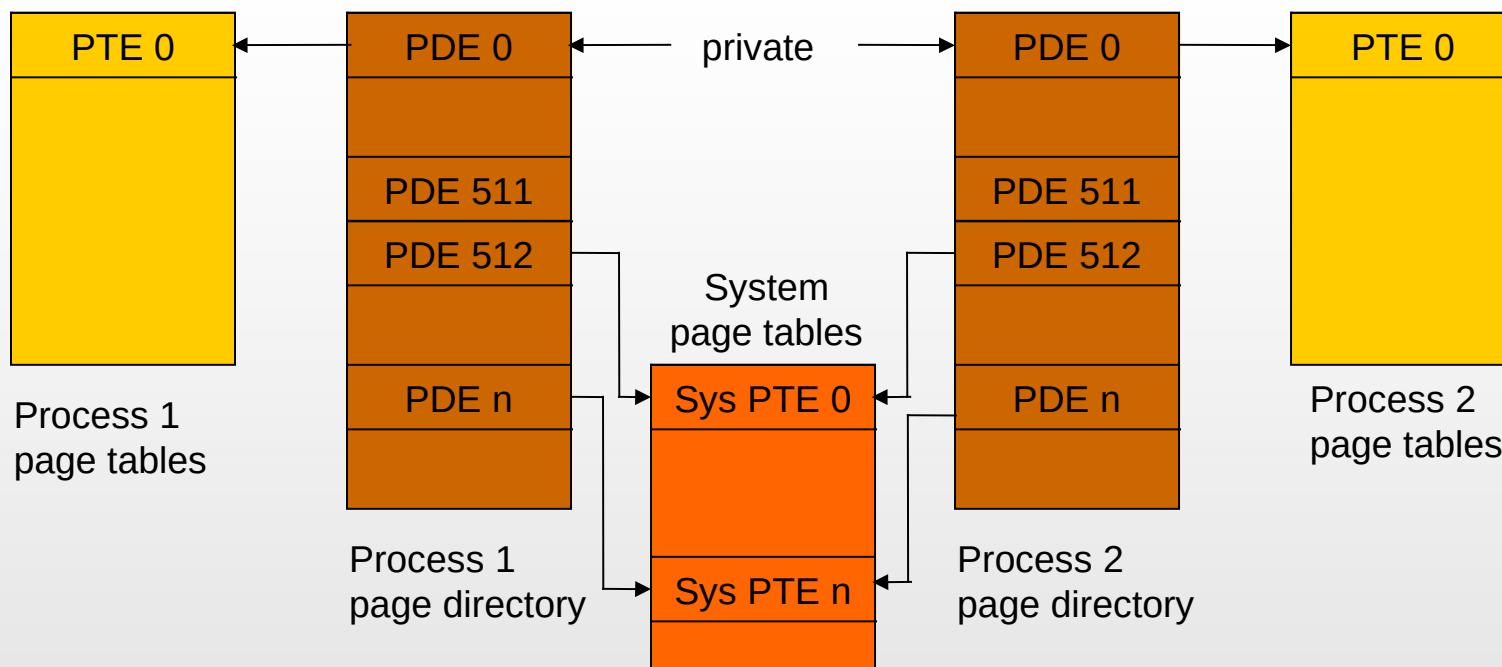
Memoria Compartida - Ej. Windows

Process space contains:

- ✓ The application you're running (.EXE and .DLLs)
- ✓ All static storage defined by the application



Memoria Compartida - Ej. Windows (cont.)



- On process creation, system space page directory entries point to existing system page tables



Mapeo de Archivo en Memoria

- Técnica que permite a un proceso asociar el contenido de un archivo a una región de su espacio de direcciones virtuales
- El contenido del archivo no se sube a memoria hasta que se generan Page Faults
- El contenido de la pagina que genera el PF es obtenido desde el archivo asociado
 - ✓ No del área de intercambio



Mapeo de Archivo en Memoria (cont.)

- Cuando el proceso termina o el archivo se libera, las páginas modificadas son escritas en el archivo correspondiente
- Permite realizar E/S de una manera alternativa a usar operaciones directamente sobre el Sistema de Archivos
- Es utilizado comúnmente para asociar librerías compartidas o DLLs



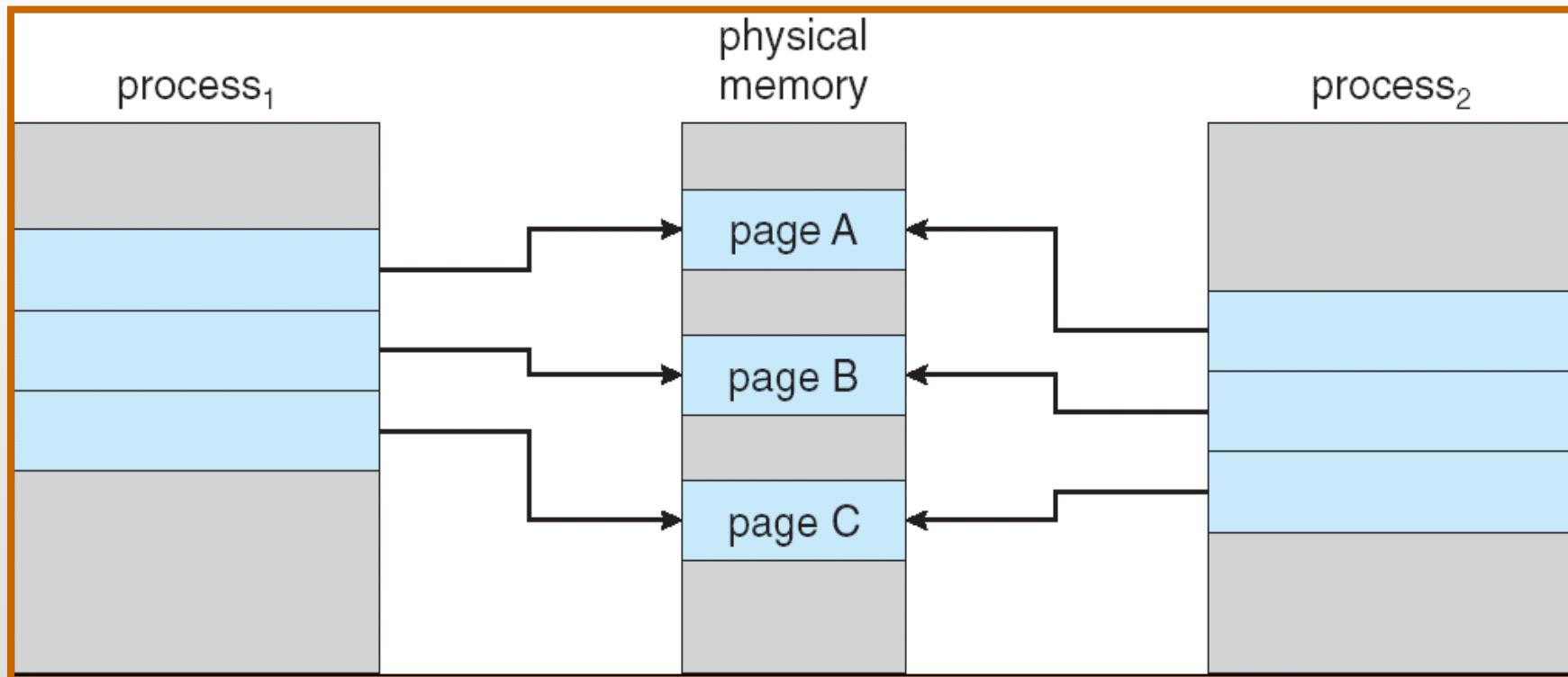
Copia en Escritura

- ✓ La copia en escritura (Copy-on-Write, COW) permite a los procesos padre e hijo compartir inicialmente las mismas páginas de memoria
 - ✓ Si uno de ellos modifica una página compartida la página es copiada
- ✓ COW permite crear procesos de forma más eficiente debido a que sólo las páginas modificadas son duplicadas



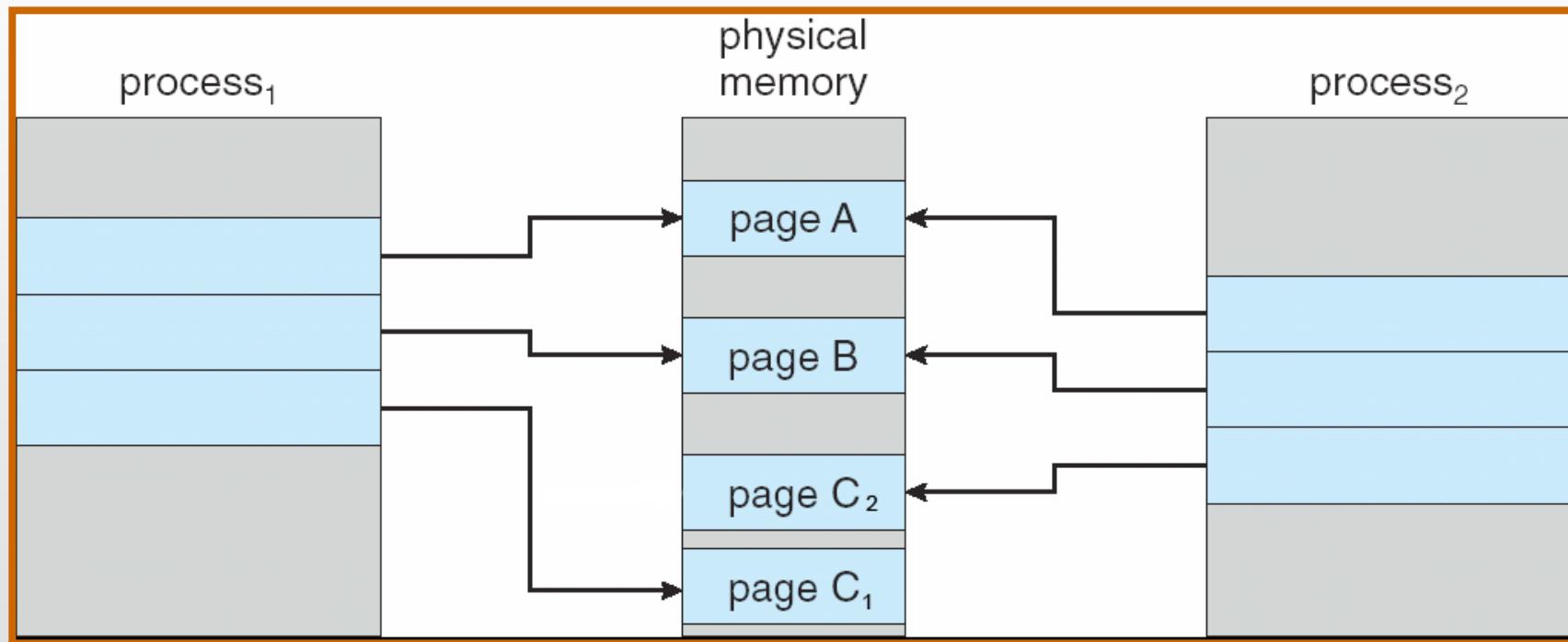
Copia en Escritura (cont.)

El Proceso 1 Modifica la Página C (Antes)



Copia en Escritura (cont.)

El Proceso 1 Modifica la Página C (Después)



Área de Intercambio

Sobre el Área utilizada

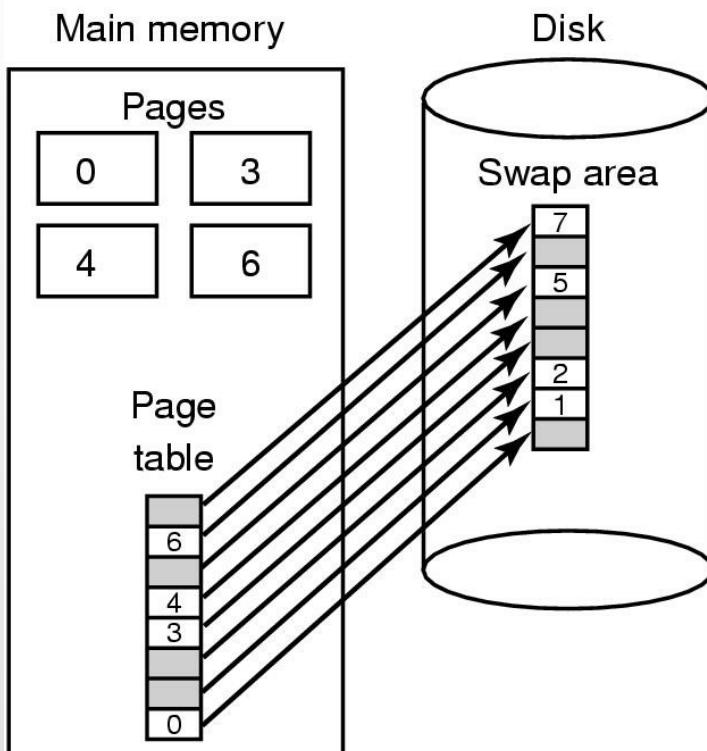
- ✓ Área dedicada, separada del Sistema de Archivos (Por ejemplo, en Linux)
- ✓ Un archivo dentro del Sistema de Archivos (Por ejemplo, Windows)

Técnicas para la Administración:

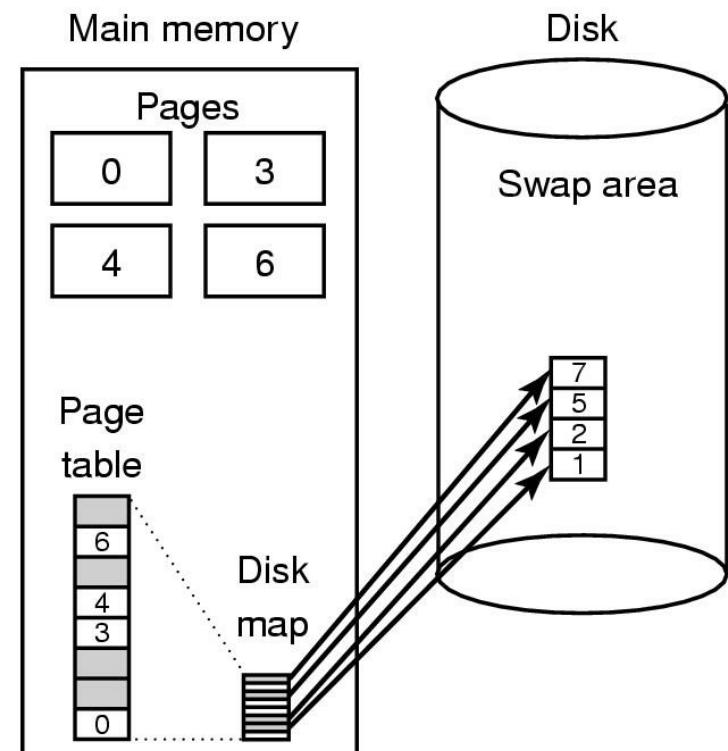
- Cada vez que se crea un proceso **se reserva una zona del área de intercambio** igual al tamaño de imagen del proceso. A cada proceso se le asigna la dirección en disco de su área de intercambio. La lectura se realiza sumando el número de página virtual a la dirección de comienzo del área asignada al proceso.
- No se asigna nada inicialmente.** A cada página se le asigna su espacio en disco cuando se va a intercambiar, y el espacio se libera cuando la página vuelve a memoria. Problema: se debe llevar contabilidad en memoria (página a página) de la localización de las páginas en disco.



Área de Intercambio (cont.)



(a)



(b)



Área de Intercambio (cont.)

- ✓ Cuando una página no esta en memoria, sino en disco, como podemos saber en que parte del área de intercambio está?
 - ✓ Rta: El PTE de dicha pagina tiene el bit V=0 y todos los demás bits sin usar!



Área de Intercambio - Linux

Permite definir un número predefinido de áreas de Swap

swap_info es un arreglo que contiene estas estructuras

<linux/swap.h>

```
64 struct swap_info_struct {  
65     unsigned int flags;  
66     kdev_t swap_device;  
67     spinlock_t sdev_lock;  
68     struct dentry * swap_file;  
69     struct vfsmount *swap_vfsmnt;  
70     unsigned short * swap_map;  
71     unsigned int lowest_bit;  
72     unsigned int highest_bit;  
73     unsigned int cluster_next;  
74     unsigned int cluster_nr;  
75     int prio;  
76     int pages;  
77     unsigned long max;  
78     int next;  
79 };
```



Área de Intercambio - Linux (cont.)

- Cada área es dividida en un número fijo de slots según el tamaño de la página
- Cuando una página es llevada a disco, Linux utiliza el PTE para almacenar 2 valores:
 - ✓ En número de área
 - ✓ El desplazamiento en el área (24 bits, lo que limita el tamaño máximo del área a 64 Gb)



Área de Intercambio - Linux (cont.)

Figure 17-6. Swap area data structures

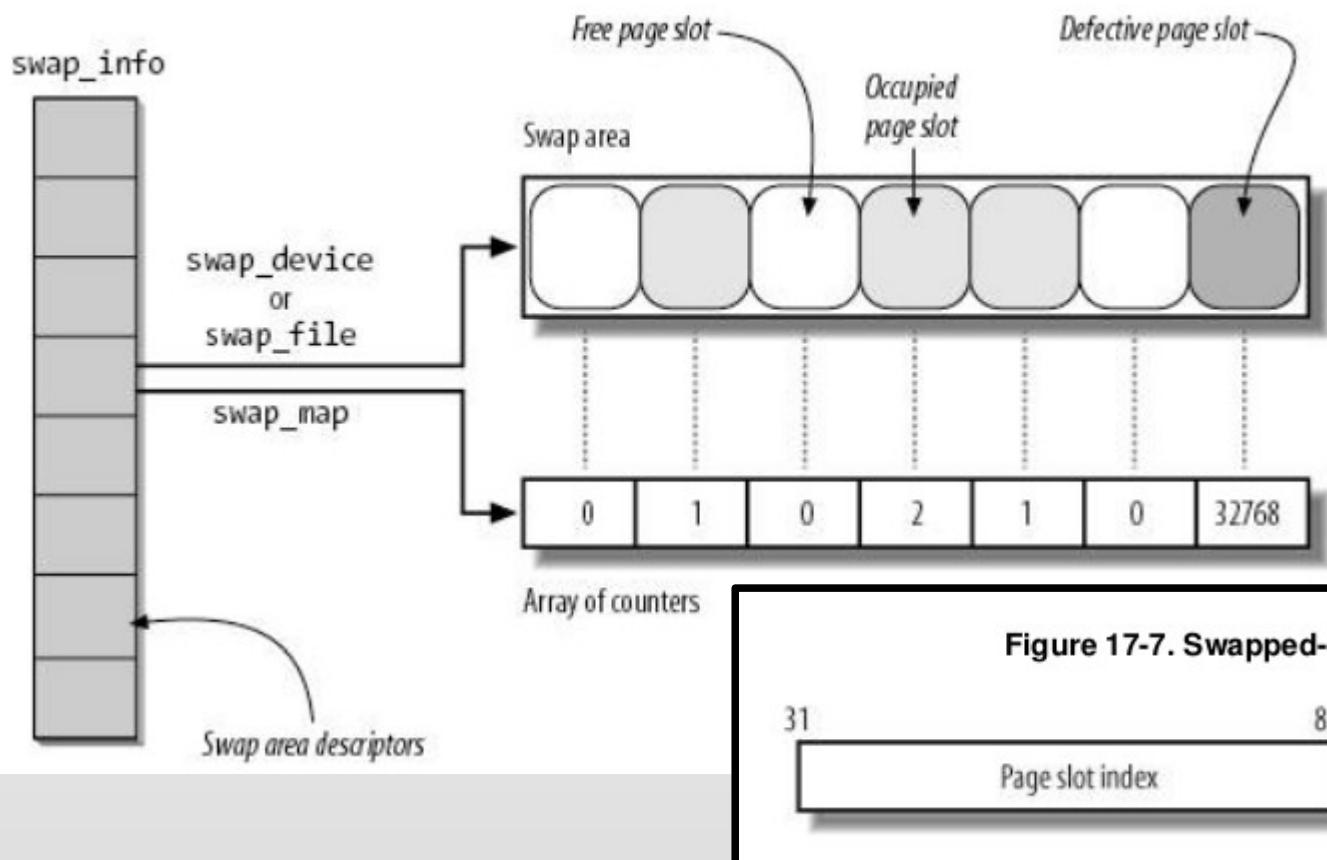
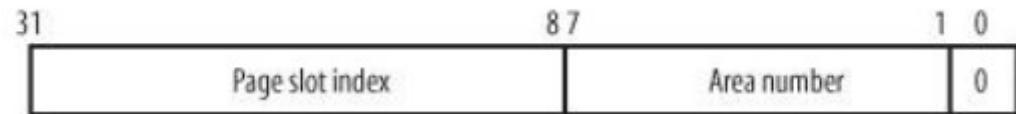


Figure 17-7. Swapped-out page identifier



Ref: Understanding the Linux Kernel <http://ceata.org/~tct/resource/utlk.pdf>

