Introducción a los Sistemas Operativos / Conceptos de Sistemas Operativos

Administración de Memoria - II

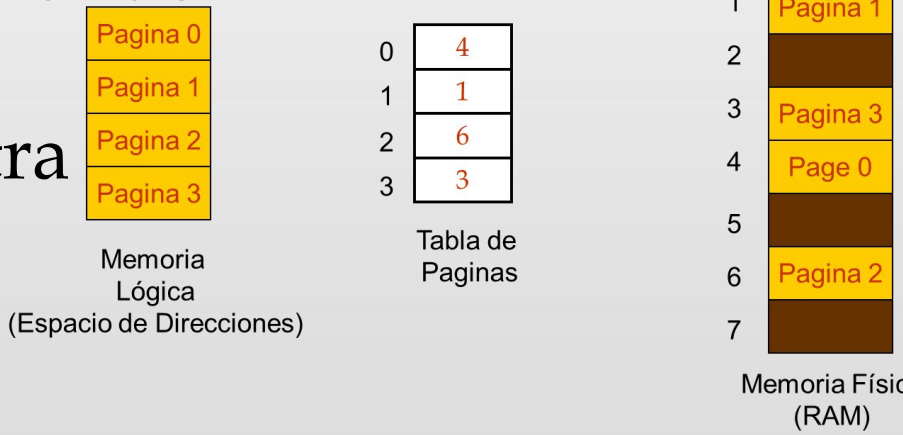
I.S.O.

- Versión: Agosto 2025
- 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 Kernel 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 Kernel 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.

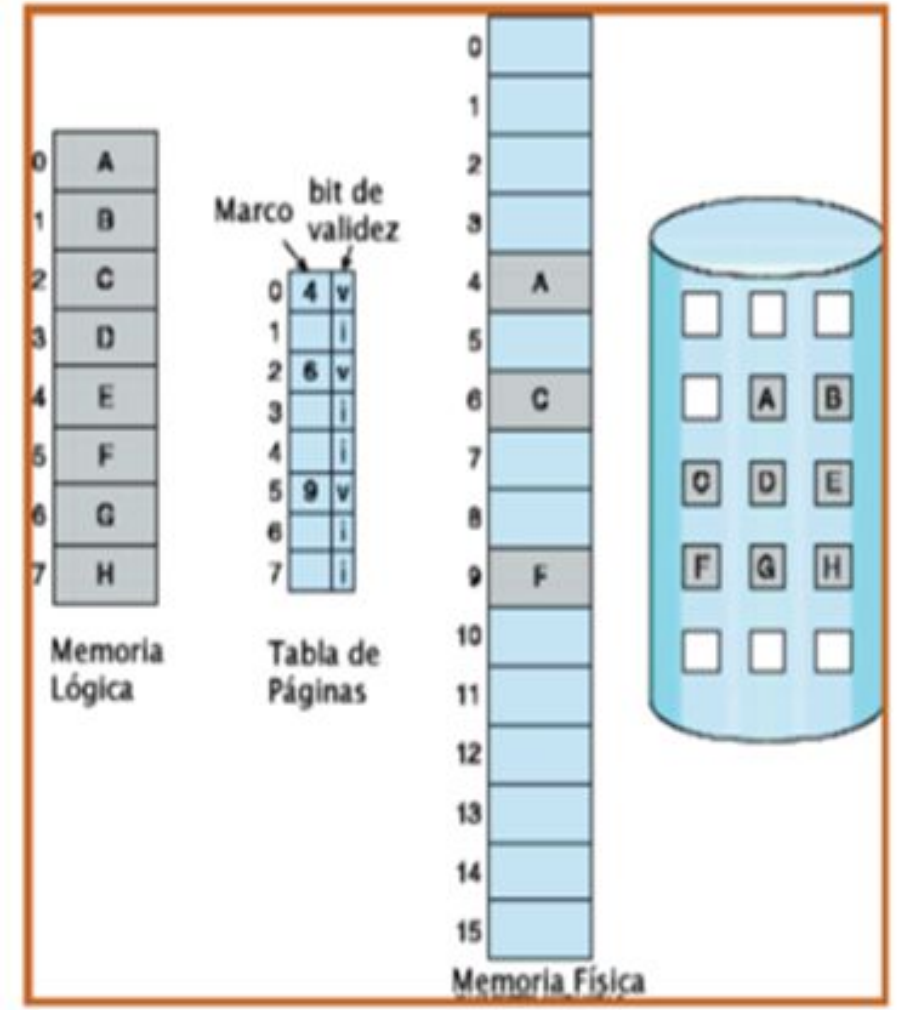
¿Que se necesita para Memoria Virtual?

- 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 Kernel debe ser capaz de manejar el movimiento de las páginas (o segmentos) entre la memoria principal y la secundaria.

Memoria Virtual con Paginación

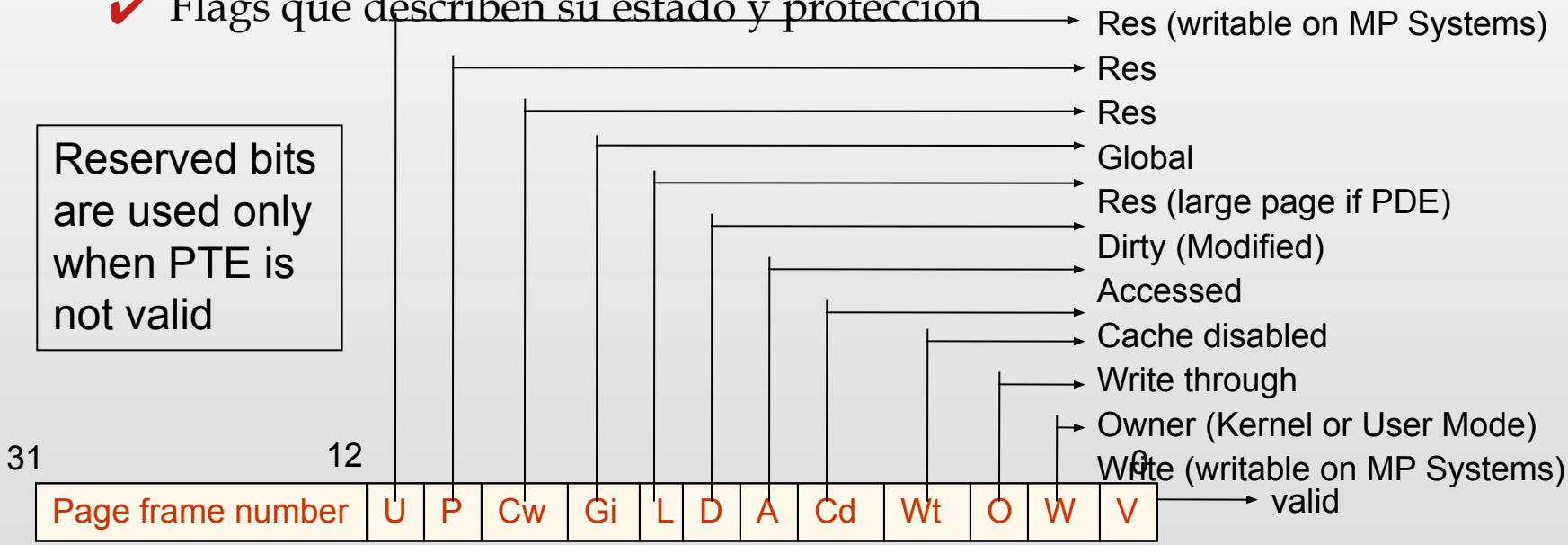
- Cada proceso tiene su tabla de páginas
- Cada entrada en la tabla referencia al frame o marco en el que se encuentra la página en la memoria principal
- Cada entrada en la tabla de páginas tiene bits de control (entre otros):
  - ✔ **Bit V**: Indica si la página está en memoria (lo activa/desactiva el Kernel, 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 Kernel)

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)

- 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.
- El Kernel podrá colocar al proceso en estado de "Blocked" (espera) mientras gestiona que la página que se necesite se cargue.
  - ✔ Para mejor productividad de la CPU

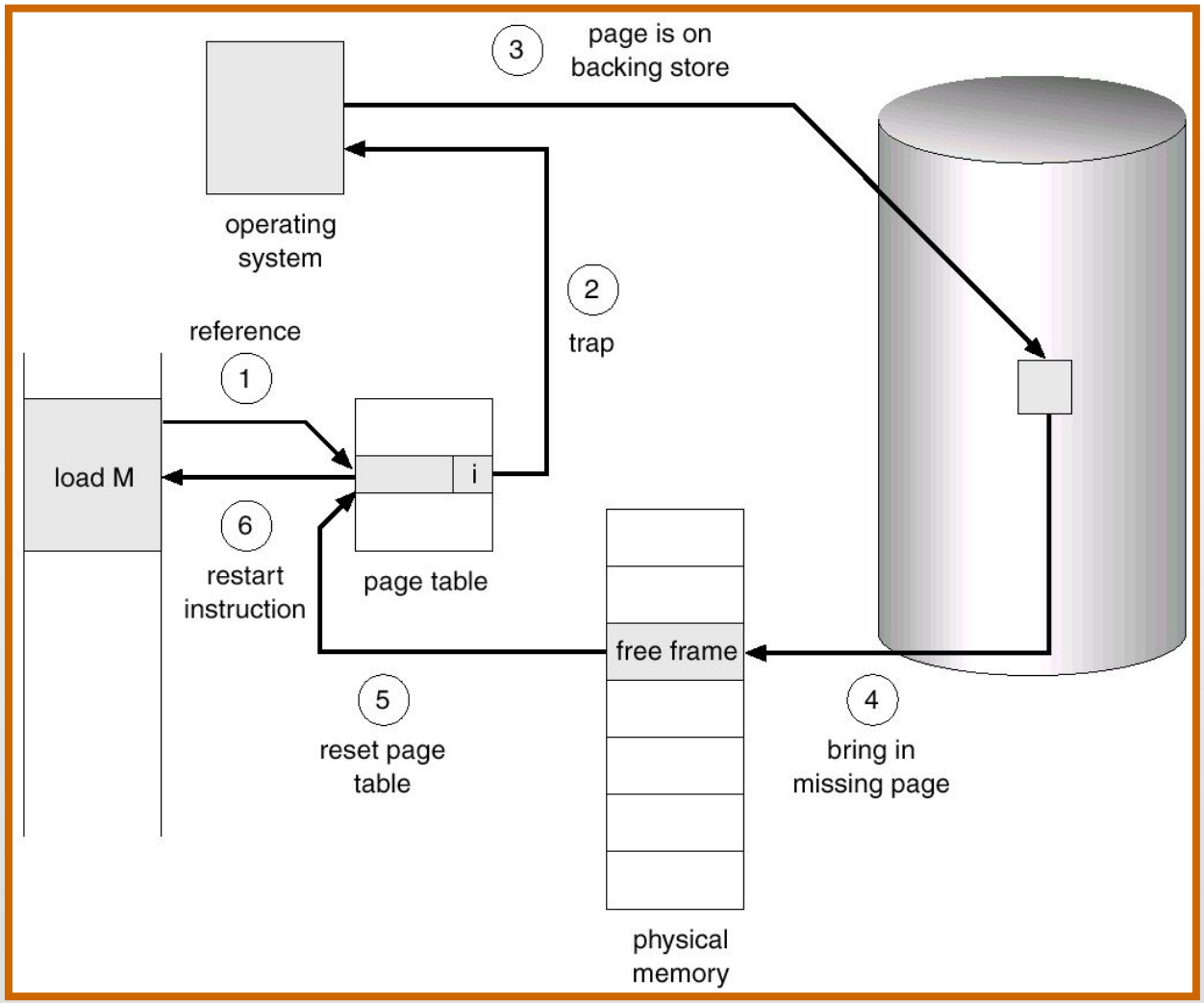
Fallo de páginas (cont.)

- El Kernel 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 Kernel 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 (interrupción) y el Kernel:
  - ✔ 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.)



TLB con Paginación por Demanda

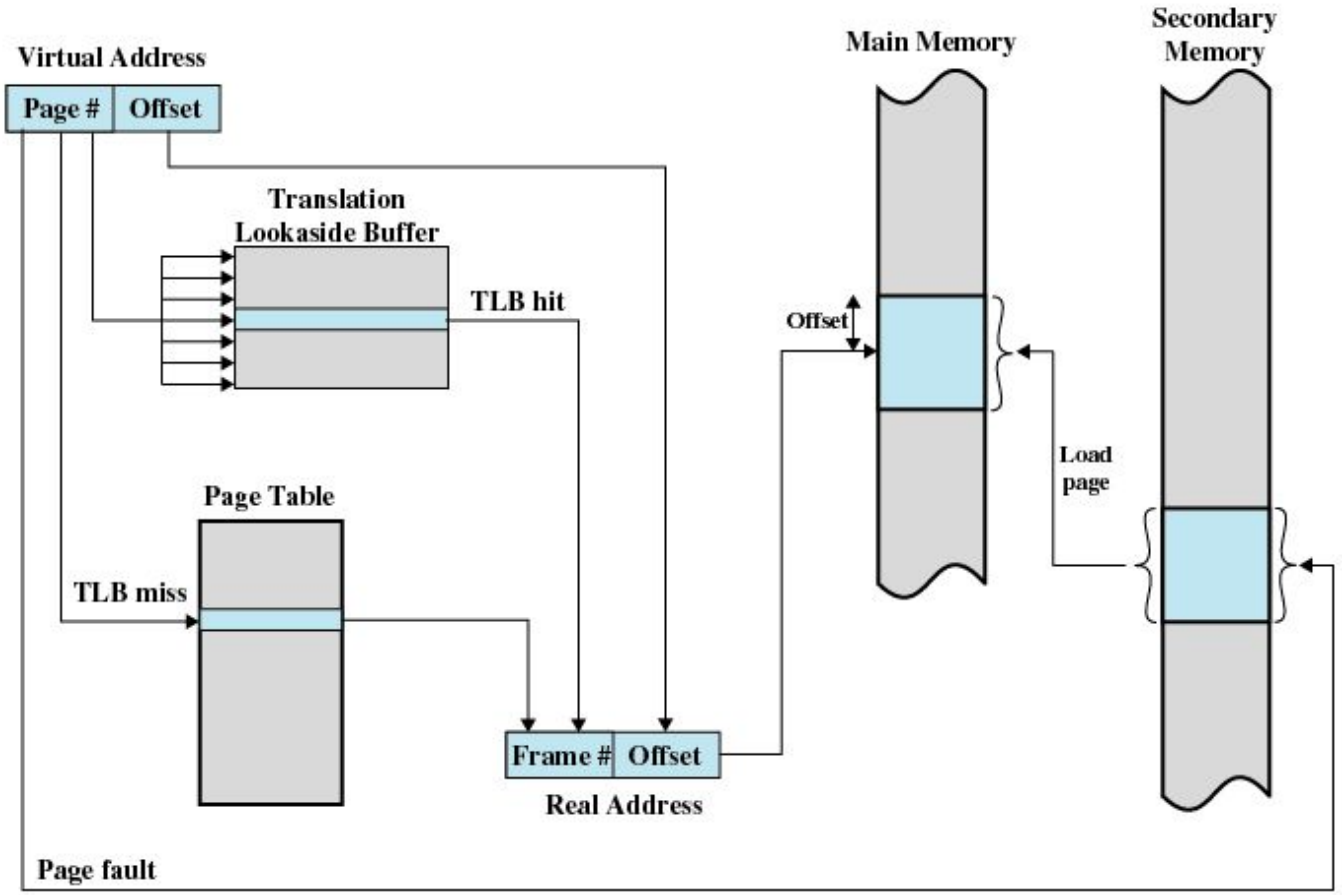


Figure 8.7 Use of a Translation Lookaside Buffer

TLB con Paginación por Demanda

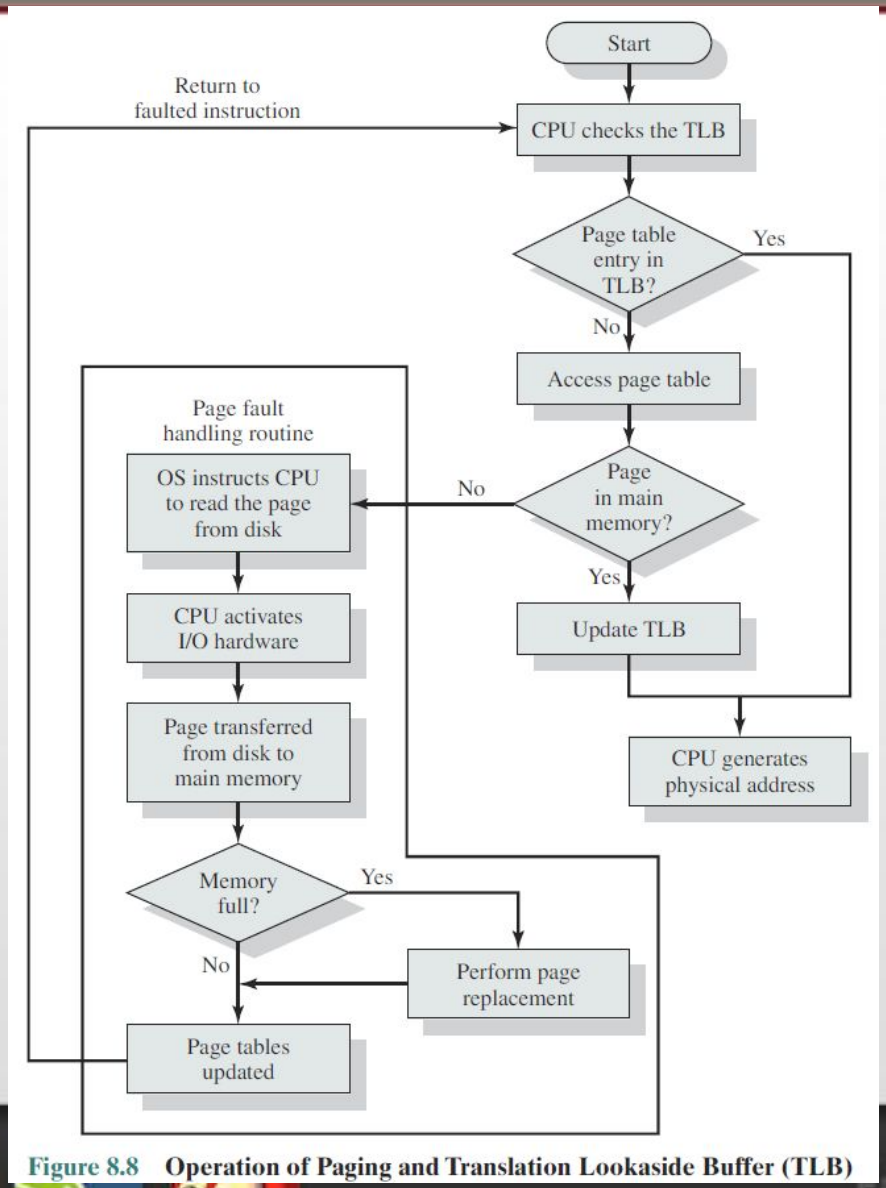


Figure 8.8 Operation of Paging and Translation Lookaside Buffer (TLB)

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 (Fist 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)

EAT = (1 – p) x memory access
+ p x (page_fault_overhead +
[swap_page_out] +
swap_page_in +
restart_overhead)

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

Políticas en el manejo de MV

Table 8.4 Operating System Policies for Virtual Memory

Fetch Policy
- Demand paging
- Prepaging

Cuando una página debe ser llevada a la memoria

Placement Policy

Donde ubicarla (best-fit, first-fit, etc…)

Replacement Policy
- Basic Algorithms
  - Optimal
  - Least recently used (LRU)
  - First-in-first-out (FIFO)
  - Clock
- Page Buffering

Elección de víctima

Resident Set Management
- Resident set size
  - Fixed
  - Variable
- Replacement Scope
  - Global
  - Local

Cuántas páginas se traen a memoria

Cleaning Policy
- Demand
- Precleaning

Cuando una página modificada debe llevarse a disco

Load Control
- Degree of multiprogramming

 # de procesos en memoria

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.

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	• Number of frames allocated to a process is fixed. • Page to be replaced is chosen from among the frames allocated to that process.	• Not possible.
Variable Allocation	• 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.	• 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