

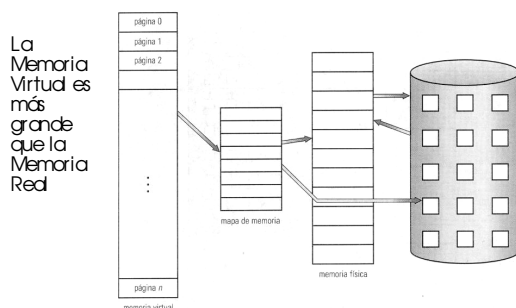
9 MEMORIA VIRTUAL

- 1 ANTECEDENTES
- 2 PAGINACIÓN BAJO DEMANDA
- 3 RENDIMIENTO PAGIN. BAJO DEMANDA
- 4 REEMPLAZO DE PÁGINA
- 5 ALGORITMOS DE REEMPLAZO DE PÁGINA
- 6 ASIGNACIÓN DE FRAMES
- 7 THRASHING
- 8 OTRAS CONSIDERACIONES
- 9 SEGMENTACIÓN BAJO DEMANDA

9.1 ANTECEDENTES

- Memoria Virtual: separación de la memoria lógica del usuario de la memoria física.
 - Sólo parte del programa necesita estar en memoria para su ejecución.
 - El espacio de direcciones lógicas puede por lo tanto ser de un tamaño mucho mayor que el espacio de direcciones físico.
 - Necesita que a las páginas se les permita ser traídas y llevadas (swapped in y out).
- La memoria virtual puede ser implementada vía
 - Paginación bajo Demanda
 - Segmentación bajo Demanda

9.1 ANTECEDENTES

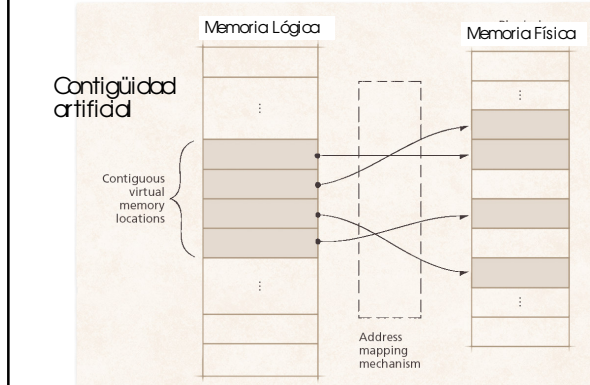


9.1 ANTECEDENTES

Evolución de las Organizaciones de Memoria

Real	Real		Virtual		
Single-user dedicated systems	Real memory multiprogramming systems		Virtual memory multiprogramming systems		
	Fixed-partition multi-programming	Variable-partition multi-programming	Pure paging	Pure segmentation	Combined paging and segmentation
	Absolute	Re-locatable			

9.1 ANTECEDENTES

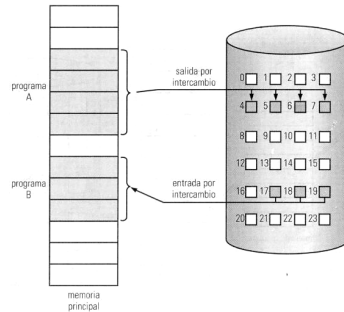


9.2 PAGINACIÓN BAJO DEMANDA

- Una página es traida a memoria sólo cuando es necesitada
 - Se necesita hacer menos E/S
 - Se necesita menos memoria
 - Respuesta más rápida
 - Más usuarios
- Si se necesita una página ⇒ se hace referencia a ella
 - Si la referencia no es válida ⇒ abortar
 - Si no está en memoria ⇒ traerla a memoria

9.2 PAGINACIÓN BAJO DEMANDA

Transferencia de una Memoria Paginada a un Espacio Contiguo en disco (intercambiador perezoso/ paginador: nunca intercambia una página a la memoria si no la va a necesitar)



9.2.1 Bit Válido-Inválido

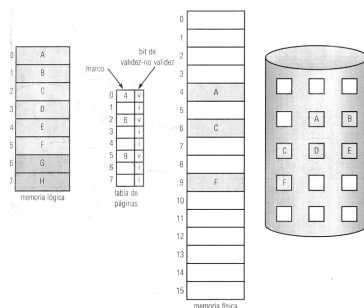
- A cada entrada a la tabla de páginas se asocia un bit válido – no válido (1 en memoria, 0 no en memoria)
- Inicialmente, el bit es puesto en 0 para todas las entradas.
- Ejemplo de tabla de páginas:
- Durante la traducción de las direcciones, si el bit en la entrada de la tabla de páginas es 0 $\Rightarrow$ falta de página (page fault).

Nº Frame	bit válido-Inválido
	1
	1
	1
	1
	0
M	
	0
	0

Tabla de páginas

9.2.1 Bit Válido-Inválido

Tabla de Páginas con algunas páginas no cargadas en memoria



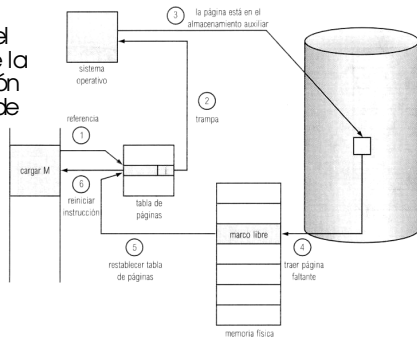
9.2.2 Falta de Página

- Si una página es referenciada alguna vez durante la ejecución del programa, la primera referencia a ésta interrumpirá al Sistema Operativo por *falta de página*
- El Sistema Operativo busca en otra tabla para decidir:
 - Si la referencia es no válida $\Rightarrow$ abortar
 - Si justo no está en memoria
- Ubicar un frame desocupado
- Traer la página solicitada al frame
- Resetear tablas, bit de validación ponerlo en 1.
- Recomenzar la instrucción



9.2.2 Falta de Página

Pasos en el Manejo de la Interrupción por Falta de Página



9.2.2 Falta de Página

- 1 Referencia: Acceso válido o inválido
 - 2 Trampa (Trap) al Sistema Operativo: Para que aborte en caso de invalidez o si es válida dar la instrucción de cargar la página
 - 3 Ubicación de la Página en Memoria de Apoyo
 - 4 Traer Página Faltante y colocarla en un frame/marco libre
 - 5 Resetear Tabla de Páginas: Para que indique que la página ya está en memoria
 - 6 Recomenzar Instrucción, ahora el proceso accede a la página pues está en la memoria
- ¿Qué pasa si se inicia la ejecución de un proceso y no hay páginas de dicho proceso en la memoria?

9.2.3 Reemplazo de Página

- Se ha supuesto que existen marcas libres. ¿Y si no hay frames desocupados?
- Encontrar alguna página en memoria, pero que no esté siendo usada realmente, entonces es sacada (swapped out).
 - Algoritmo de reemplazo de páginas para escoger un frame *Víctima*
 - Rendimiento: Si no hay marcas libres, se requieren dos transferencias de página (la que se saca y la que se carga), esto aumenta el tiempo de servicio de un fallo y del TAE.

9.3 RENDIM. DE LA PAG. BAJO DEMANDA

- Tiempo de acceso a la memoria: varía entre 10 a 100 ns.
- Sin fallos de página-> el tiempo de acceso efectivo = tiempo de acceso a la memoria. Si ocurre un fallo de página, primero se lee del disco y luego acceder al byte deseado.
- Tasa/ probabilidad de ocurrencia de falta de Página : $0 \leq p \leq 1$
 - Si $p = 0$, no hay falta de página
 - Si $p = 1$, cada referencia produce una falta de página.
- Tiempo de Acceso Efectivo:
 $= (1 - p) \cdot \text{acceso a memoria}$
 $+ p \cdot (\text{tiempo de fallo de página}) : \text{overhead de la falta de página} + [\text{tiempo de swap out}] + \text{tiempo de swap in} + \text{overhead de restart}$

9.3 RENDIM. DE LA PAG. BAJO DEMANDA

Ejemplo de Paginación Bajo Demanda:

- Tiempo de Acceso a Memoria = $1 \mu s$
- El 50% de las veces, la página que está siendo reemplazada ha sido modificada y, por lo tanto, necesita ser sacada (swapped out).
- Tiempo de Swap de la Página = $10 \text{ ms} = 10.000 \mu s$
$$\text{TAE} = (1 - p) \times 1 + p(15.000)$$
$$1 + 14.999 P \quad (\text{en } \mu s)$$

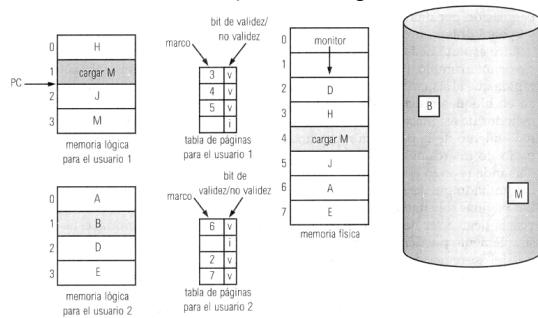
Conclusión: Tiempo de acceso efectivo es directamente proporcional a la frecuencia de fallos de página.

9.4 REEMPLAZO DE PÁGINA

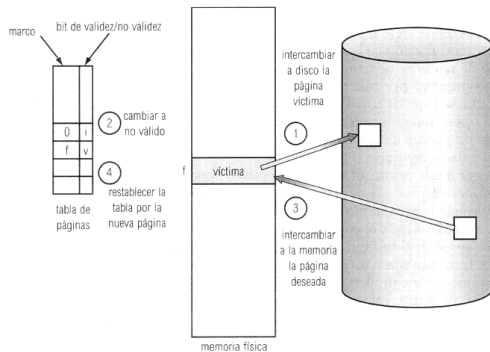
- Prevenir la sobresignación de memoria, modificando la rutina de servicio de falta de página para incluir reemplazo de página.
- Usar bit de modificación (dirty) para así reducir el overhead de transferencia de página, sólo las páginas modificadas son escritas al disco.
- El reemplazo de página completa la separación entre las memorias lógicas y físicas, en cuanto a que se proporciona una gran memoria virtual en una memoria física pequeña.
- Con la paginación que no es por demanda, las direcciones de usuario (lógicas) se transformaban en direcciones físicas, siendo predio que todas las páginas de un proceso estuvieran en la memoria física. Con la paginación por demanda no se limitan las dir. Lógicas con la memoria física.

9.4 REEMPLAZO DE PÁGINA

Necesidad de Reemplazo de Páginas



9.4 REEMPLAZO DE PÁGINA



9.5 ALGORITMOS DE REEMPL. DE PÁGINA

Para implementar la paginación por demanda se necesita: Algoritmo de asignación de marcos y un algoritmo de reemplazo de páginas:

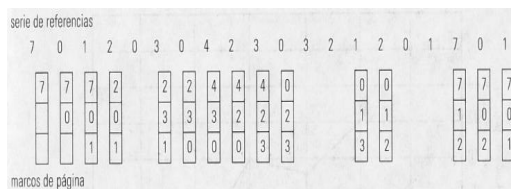
- 1 FIFO
- 2 Óptimo
- 3 LRU
- 4 Aproximación a LRU
- 5 Contabilización
- 6 Colocación de Páginas en Buffers

9.5 ALGORITMOS DE REEMPL. DE PÁGINA

- Se desea la tasa más baja de falta de página.
- Se evalúan algoritmos, ejecutándolos para un string de referencia a memoria particular y calculando el número de faltas de página para ese string.
- En todos los ejemplos siguientes, el string de referencia es 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

9.5.1 Algoritmo FIFO

Si es necesario reemplazar una página, se escoge la más vieja. Total de 15 fallos



9.5.1 Algoritmo FIFO

- String de referencia: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (puede haber 3 páginas en memoria a la vez por proceso)

1	1	4	5
2	2	1	3
3	3	2	4

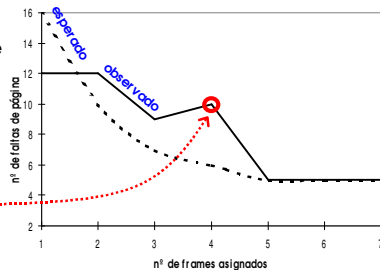
- 4 frames

1	1	5	4
2	2	1	5
3	3	2	
4	4	3	

9.5.1 Algoritmo FIFO

- Se espera que, a más frames, menos faltas de página.
- Pero ello puede no ocurrir siempre.

Se ha descubierto empíricamente que el Algoritmo FIFO sufre de la **Anomalía de Belady**: frecuencia de faltas aumenta al aumentar las frames asignadas



9.5.2 Algoritmo Óptimo

- Objetivo: reemplazar aquella página que no será usada por el mayor período de tiempo.
- No tiene anomalía de Belady.
- Ejemplo con 4 frames (1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5)

1	4
2	
3	
4	5

6 faltas de página

- ¿Cómo saberlo?
- Puede ser usado para medir cuán bien se ejecuta un algoritmo.

9.5.3 Algoritmo LRU

- String de referencia 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	5
2	
3	5 4
4	3

- Se considera la mejor opción para implementar, no tiene anomalía de Belady
- Implementación mediante Contadores
 - La entrada de cada página tiene un contador; cada vez que ésta es referenciada a través de esta entrada, copia el contenido del reloj en el contador. Se tiene la "hora" de la última referencia.
 - Cuando se necesita cambiar una página, se examinan los contadores para determinar cual cambiar.

9.5.3 Algoritmo LRU

- Implementación mediante Stacks: mantener un stack de números de página en una lista enlazada doble
 - Si la página es referenciada: moverla al tope
 - No necesita buscar

9.5.4 Aproximaciones a LRU

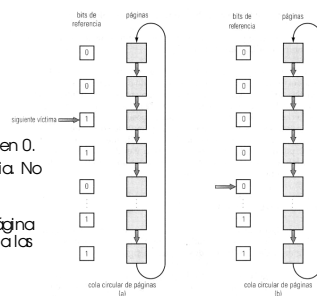
- Bit de Referencia
- Segunda Oportunidad
- Segunda Oportunidad Mejorado

9.5.4.1 Bit de Referencia

- Se asocia un bit con cada página, inicialmente = 0.
- Cuando la página es referenciada. El bit se pone en 1.
- Se puede determinar cuáles se han usado y cuáles no. Sin embargo, no se sabe en que orden.

9.5.4.2 Segunda Oportunidad

- Necesita bit de referencia.
- Reemplazo de reloj.
- Si la página va a ser reemplazada (en orden temporal) tiene bit de referencia en 1, entonces:
 - Se pone el bit de referencia en 0.
 - Se deja la página en memoria. No se reemplaza (segunda oportunidad)
 - Se reemplaza la siguiente página (en orden temporal), sujeto a las mismas reglas.



9.5.4.3 Segunda Oportunidad Mejorado

Bit de referencia + bit de reemplazo: Podría requerir explorar la cola varias veces. Se ocupa en Mac.

Valor	Caracterización	Rank
(0,0)	Ni Recientemente Usada ni Modificada, es la mejor candidata	1
(0,1)	No recientemente Usada pero Modificada, no es la mejor, porque la página debe ser reescrita en disco antes de ser reemplazada	3
(1,0)	Recientemente Usada y No Modificada, puede ser usada nuevamente	2
(1,1)	Recientemente Usada y Modificada, puede ser usada nuevamente y debe ser reescrita en disco antes de ser reemplazada	4

9.5.5 Algoritmos de Contabilización

- Mantener un contador del número de referencias que han sido hechas a cada página.
- Algoritmo **LFU (menos frecuentemente usada)**: reemplazar la página con el contador mas pequeño.
- Algoritmo **MFU (más frecuentemente usada)**: basada en el argumento que la página con el contador mas pequeño fue probablemente traida recién y volverá a ser usada.

9.5.6 Colocación de Páginas en Buffer

- Mantener reserva (pool) de frames libres $\Rightarrow$ más rápido pues se deja en el marco libre antes de escribir a disco.
- Lista de páginas modificadas: Siempre que el dispositivo de paginación esté ocioso, se escoge una página modificada y se escribe a disco y se cambia el bit de modificación.
- Mantener reserva de frames libres con n° de página asociado: Es posible reutilizar la página vieja directamente de la reserva de marcos libres si llega a utilizarse antes de que sea ocupado ese marco.

9.6 ASIGNACIÓN DE FRAMES

- 1 Número Mínimo de Frames.
- 2 Algoritmos de Asignación.
- 3 Asignación Global v/s Local.

9.6.1 Número Mínimo de Frames

- Cada proceso nuevo necesita un número mínimo de frames para funcionar.
- Es necesario tener suficientes marcas para contener todas las páginas a las que una sola instrucción puede hacer referencia.
- Está definido por la arquitectura del conjunto de instrucciones.

9.6.2 Algoritmos de Asignación

Asignación fija

- Asignación Equitativa: ej, Si hay 100 frames y 5 procesos, asignar 20 a cada uno. ¿qué pasa si un proceso necesita menos de 20 marcas?
- Asignación Proporcional: Asignar de acuerdo al tamaño del proceso. (ejemplo)

9.6.2 Algoritmos de Asignación

Asignación por prioridades

- Usar un esquema proporcional de asignación usando prioridades en vez del tamaño.
- Si el proceso P_i genera una falta de página,
 - seleccionar uno de sus frames para reemplazo.
 - seleccionar para reemplazo un frame de un proceso con número de prioridad bajo.

9.6.3 Asignación Global v/s Local

- Reemplazo **Local**: cada proceso selecciona solo de su propio conjunto de frames asignados.
- Reemplazo **Global**: el proceso selecciona un frame del conjunto de todos los frames. Mejora el rendimiento y es más usado. ¿por qué?

9.7 HIPERPAGINACIÓN (Thrashing)

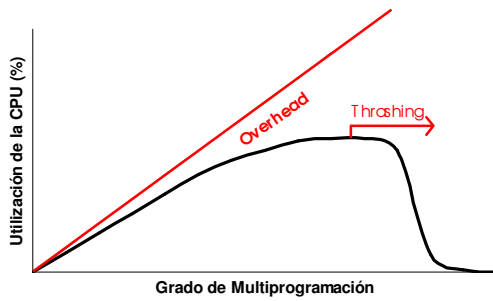
1 Causas

3 Frecuencia de Falta de Página

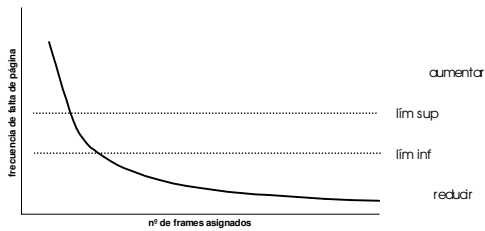
9.7.1 Causas del Thrashing

- Si un proceso no tiene suficientes páginas, la tasa de falta de página será muy alta. Esto conduce a:
 - Baja utilización de la CPU.
 - El Sistema Operativo piensa que se necesita aumentar el grado de multiprogramación.
 - Se traen nuevos procesos al sistema.
- **Thrashing** ⇒ un proceso está ocupado haciendo sólo paginación (swapping)

9.7.1 Causas del Thrashing



9.7.3 Frecuencia de Falta de Página



- Controlar la frecuencia para evitar la hiperpaginación
- Permite establecer una tasa aceptable de faltas de página
 - Si la tasa real es muy baja o sea menos del límite inferior, el proceso pierde frames.
 - Si la tasa real es muy alta o sea excede el límite superior, el proceso gana frames.

9.8 OTRAS CONSIDERACIONES

- 1 Prepaginación
- 2 Selección del Tamaño de página
- 4 Estructura del Programa
- 5 Interbloqueo de E/S

9.8.1 Prepaginación

- Trae a Memoria Principal todas las páginas que se necesitan, de una vez.
- Utilizada para evitar el alto nivel de paginación inicial.
- Puede ocurrir que algunas páginas no se necesiten.

9.8.2 Selección del Tamaño de página

- Los tamaños de página siempre son potencias de 2, varían entre 2^9 (512) y 2^{14} (16384) bytes.
- Páginas pequeñas minimizan la Fragmentación interna
- Tamaño de la tabla de páginas: cada proceso activo tiene su propia copia de tablas, se favorecen las páginas grandes
- Overhead por el tiempo de E/S: tiempos de búsqueda, latencia y transferencia, se requiere minimizar -> páginas grandes
- La tendencia histórica es el uso de páginas grandes (4k por ej.)

9.8.5 Interbloqueo de E/S

- Al haber E/S, es necesario "fijar" o bloquear las páginas comprometidas.
- Se asocia un bit de bloqueo a cada frame.
- Si el frame está bloqueado, la página contenida en éste no puede ser removida para reemplazo

9.9 SEGMENTACIÓN BAJO DEMANDA

- Usada cuando el hardware es insuficiente para implementar la paginación bajo demanda.
- OS/2 asigna la memoria en segmentos, de los cuales lleva un registro, a través de descriptores de segmentos.
- Un descriptor de segmento contiene un bit de validez para indicar si el segmento está actualmente o no en memoria.
 - Si está en memoria principal, el acceso continúa.
 - Si no está, hay interrupción por falta de segmento.
