

8.3 PAGINACIÓN

- La Paginación es un esquema de gestión de memoria que permite que el espacio de direcciones físico de un proceso sea no contiguo.
- La paginación evita el problema de hacer calzar el tamaño variable de los programas con la capacidad de la memoria disponible, eliminándose la fragmentación externa.
- Hoy en día, la paginación se hace de manera combinada entre el hardware y el SO.

8.3 PAGINACIÓN

Memoria Lógica

Memoria Física

Contigüidad
artificial

8.3 PAGINACIÓN

- 1 Método Básico
- 2 Estructura de la Tabla de Páginas
- 3 Paginación Multinivel
- 4 Tabla de Páginas Invertida
- 5 Páginas Compartidas

8.3.1 Método Básico

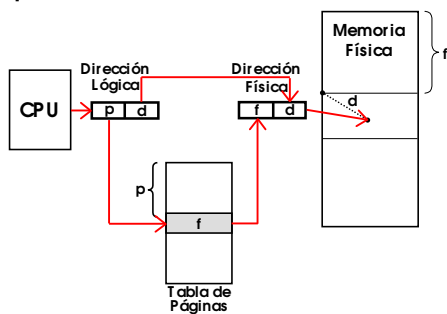
- El espacio lógico de direcciones de un proceso puede ser no contiguo; a un proceso le es asignada memoria física cada vez que ella está disponible.
- Se divide la **memoria física** en bloques de tamaño (capacidad) fijo, llamados **marcos** (frames), su tamaño es una potencia de 2.
- Se divide la **memoria lógica** en bloques del mismo tamaño que los frames, llamados **páginas** (pages).

8.3.1 Método Básico

- Se mantiene información acerca de todos los frames libres.
- Para ejecutar un programa de tamaño N páginas, sólo se necesita encontrar N frames libres y cargar el programa.
- Se necesita una **tabla de páginas** (page table) para traducir las direcciones lógicas a físicas (mapeo).
- La paginación sufre de fragmentación interna.

8.3.1 Método Básico

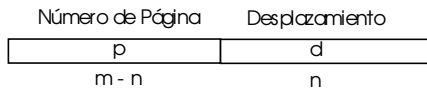
Arquitectura de Traducción de Direcciones



8.3.1 Método Básico

Esquema de Traducción de Direcciones

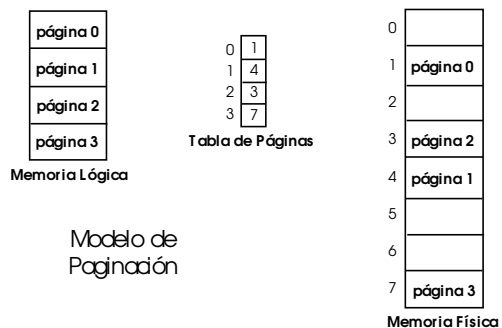
- Una dirección generada por la CPU es dividida en: Tamaño de la página está definido por el HW. Es potencia de 2 y varía entre 512B y 16 MB. Si el tamaño del espacio lógico es de 2^m , y el tamaño de página es de 2^n , entonces la dirección lógica es:
 - Número de página (p)**, usada como índice en la tabla de páginas, la cual contiene la dirección base de cada página en la memoria física.
 - Desplazamiento (d)**, que se combina con la dirección base para definir la dirección en memoria física que es enviada a la unidad de memoria.



8.3.1 Método Básico

- La paginación es una forma de relocalización.
- Es similar a usar un registro de relocalización por cada frame de memoria.
- No existe fragmentación externa.
- Como los frames se asignan en unidades, es posible que el último frame no esté completamente ocupado, y en el peor de los casos se ocupe con sólo 1B. Esto genera fragmentación interna.

8.3.1 Método Básico



8.3.1 Método Básico

Ejemplo

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
13	n
14	o
15	p

Memoria
Lógica

0	5
1	6
2	1
3	2

Tabla de
Páginas

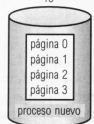
0	
4	i j k l
8	m n o p
12	
16	
20	a b c d
24	e f g h
28	

Memoria
Física

8.3.1 Método Básico

lista de marcos libres

14
13
18
20
15



(a)

lista de marcos libres

15
13
14
15
16
17
18
19
20
21

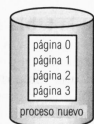


tabla de páginas de proceso nuevo

0	14
1	13
2	18
3	20

(b)

8.3.2 Estructura de la Tabla de Páginas

- Implementación de la Tabla de Páginas
 - La Tabla es mantenida en memoria principal.
 - El registro base de la tabla de páginas (PTBR) apunta a la tabla de páginas.
 - El registro de longitud de la tabla de páginas (PTLR) indica el tamaño de la tabla de páginas.
 - En este esquema, cada acceso a datos/instrucciones requiere de dos accesos a memoria. Uno para la tabla de páginas y uno para los datos/instrucciones.
 - El problema de tener dos accesos a memoria puede ser resuelto por el uso de un hardware especial de búsqueda (look-up) rápida (cache) llamado registro asociativo o buffer de traducción del tipo look-aside (TLBS).

8.3.2 Estructura de la Tabla de Páginas

- Registros Asociativos: búsqueda por delta

Página	Frame

Traducción de Direcciones (A' , A'')

- Si A' está en un registro asociativo, sacar el frame.
- Sino traer el frame desde la tabla de páginas a memoria

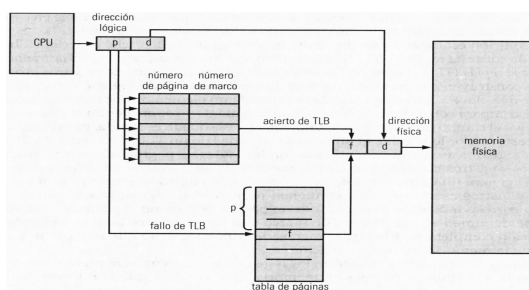
8.3.2 Estructura de la Tabla de Páginas

Tiempo de Acceso Efectivo (EAT)

- Búsqueda asociativa = ϵ unidades de tiempo
- La tasa de éxito (hit ratio) es el porcentaje de veces que un número de página se encuentra en los registros asociativos; la tasa tiene relación con el número de registros asociativos.
- Hit ratio = α
- $EAT = (1 + \epsilon)\alpha + (2 + \epsilon)(1 - \alpha) = 2 + \epsilon - \alpha$
- (ver ejemplo)

8.3.2 Estructura de la Tabla de Páginas

Hardware de paginación con TLB



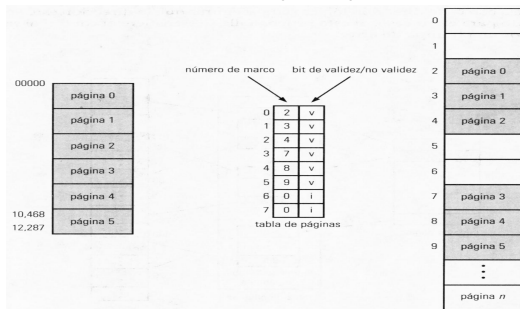
8.3.2 Estructura de la Tabla de Páginas

Protección de Memoria

- La protección de memoria es implementada asociando bits de protección con cada frame.
- El bit válido / no válido asociado a cada entrada en la tabla de páginas indica:
 - Válido (v): que la página asociada está en el espacio de direcciones lógico del proceso y es, por lo tanto, una página legal.
 - No válido o inválido (i): que la página no está en el espacio de direcciones lógico del proceso. Cualquier referencia produce una interrupción.

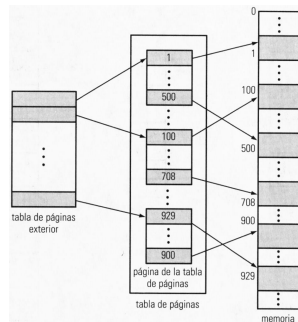
8.3.2 Estructura de la Tabla de Páginas

Protección de Memoria: Espacio de direcciones 0 a 16383, con un programa que usará de 0 a 10468. Página 6 y 7 inválidas. ¿y hasta la dirección 12.287 es válido?. 12.288 a 16.383 se genera fragmentación interna.



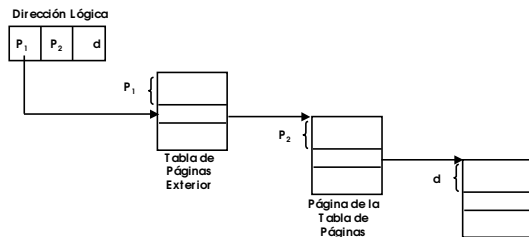
8.3.3 Paginación Multinivel

Esquema de
Tabla de
Páginas de
Dos Niveles:
Tabla de
páginas se
pagina



8.3.3 Paginación Multinivel

Esquema de traducción de direcciones para una arquitectura de paginación de 2 niveles

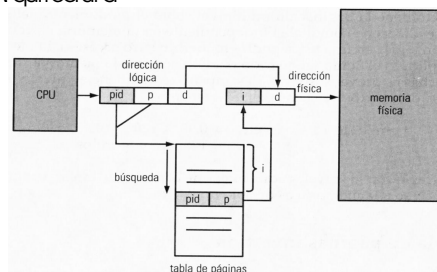


8.3.4 Tabla de Páginas Invertida

- Las tablas podrían consumir grandes cantidades de memoria física, que se ocupa de mantenerse al tanto del uso del resto de la memoria física.
- Solución: Usar una Tabla de Páginas Invertida que tiene una entrada para cada página real de memoria.
- La entrada consiste de la dirección virtual de la página almacenada en dicha ubicación real, con información sobre el proceso dueño de dicha página.
- La memoria necesaria para almacenar la tabla de páginas disminuye, pero aumenta el tiempo necesario para recorrer la tabla cuando hay una referencia.
- Usar tabla de hashing para limitar la búsqueda a una o muy pocas entradas a la tabla de páginas.

8.3.4 Tabla de Páginas Invertida

Arquitectura



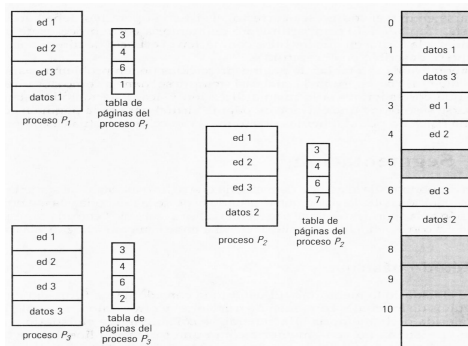
8.3.5 Páginas Compartidas (Shared)

- Código compartido
 - Una copia de código read-only (reentrante o puro) es compartida entre distintos procesos (ej., editores de texto, compiladores, etc).
 - El código compartido debe aparecer en la misma ubicación en el espacio de direcciones lógico de todos procesos.
- Código y datos privados
 - Cada proceso mantiene una copia separada de código y datos
 - Las páginas para código y datos privados pueden aparecer en cualquier parte en el espacio de direcciones lógico.

8.3.5 Páginas Compartidas (Shared)

Ejemplo

8.3.5 Páginas Compartidas (Shared)



8.4 SEGMENTACIÓN

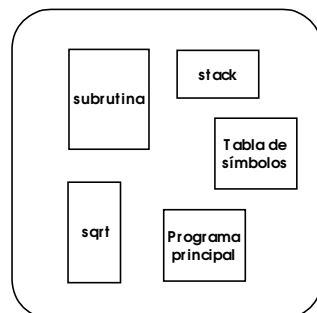
- 1 Método Básico
- 2 Hardware
- 3 Implementación de la Tabla de Segmentos
- 4 Protección y Compartición
- 5 Fragmentación

8.4.1 Método Básico

- Es un esquema de administración de memoria que proporciona una visión de memoria desde el punto de vista del usuario.
- Un programa es una colección de **segmentos**. Un segmento es una unidad lógica tal como:
 - programa principal
 - Procedimiento o función
 - variables locales y globales
 - bloque común
 - stack
 - tabla de símbolos
 - arreglos

8.4.1 Método Básico

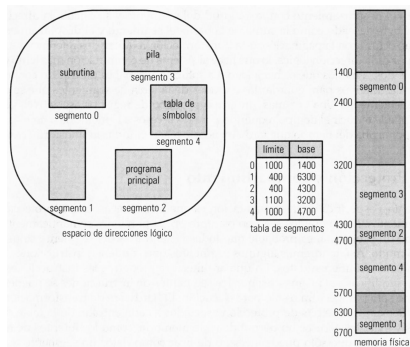
Visión Lógica
de la
Segmentación



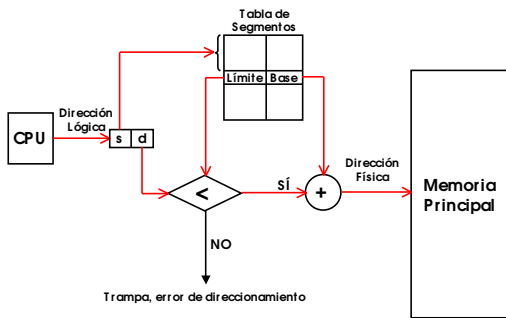
Espacio Lógico de Direcciones

8.4.1 Método Básico

Ejemplo



8.4.2 Hardware



8.4.3 Implementación Tabla de Segmentos

- Una dirección lógica consiste en el par: $\langle \text{numero de segmento, desplazamiento} \rangle$
- Tabla de segmentos: mapea direcciones definidas por el usuario (bidimensionales) en direcciones físicas (unidimensionales); cada entrada a la tabla tiene:
 - base: contiene la dirección física de comienzo donde el segmento reside en memoria
 - límite: especifica la longitud de segmento
- Registro base de la tabla de segmentos (STBR), apunta a la posición de la tabla de la tabla de segmentos en memoria

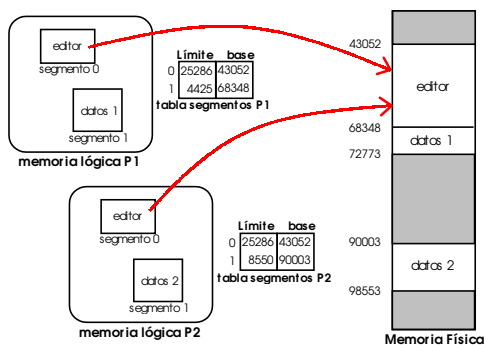
8.4.3 Implementación Tabla de Segmentos

- Registro de la longitud de la tabla de segmentos (STLR) indica número de segmentos usados por un programa; el número de segmentos es legal si $s < \text{STLR}$
- Reubicación:
 - Dinámica
 - Por tabla de segmentos
- Compactación:
 - por segmentos compartidos
 - mismo número de segmentos

8.4.3 Implementación Tabla de Segmentos

- Asignación: First fit / Best fit. Presenta fragmentación externa
- Protección: a cada entrada en la tabla de segmentos se asocia
 - bit de validación = 0 => segmento ilegal
 - privilegio de lectura / escritura / ejecución
- Bits de protección asociados con segmentos; la comparación de código ocurre a nivel de segmento.
- Ya que los segmentos varían en longitud, la asignación de memoria es un problema dinámico

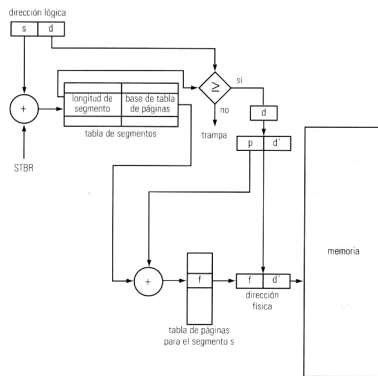
8.4.4 Protección y Compartición



8.4.5 Fragmentación

- La Segmentación provoca Fragmentación Externa, aunque en un grado mucho menor que en MVT.
- Al ser la segmentación un proceso dinámico, es posible aplicar Compactación.

8.5 SEGMENTACIÓN CON PAGINACIÓN



8.7 RESUMEN

- Comparación de las diferentes estrategias de administración de memoria, en base a:
 - **Soporte de hardware:** Registro base simple o un par de registros base y límite es suficiente para una o varias particiones. Paginación y Segmentación requieren tablas de transformación.
 - **Rendimiento:** Si se compara dirección lógica y se suma un valor, es una operación rápida. Paginación y segmentación dependen de cómo se implementen.
 - **Fragmentación:** Sistemas que tienen unidades de asignación de tamaño fijo, como partición única y paginación, favorecen la fragmentación interna. Sistemas con unidades de asignación de tamaño variable como el de múltiples particiones y el de segmentación favorecen la fragmentación externa.

8.7 RESUMEN

- **Reubicación:** Problema de fragmentación externa se soluciona con la compactación. Se requiere reubicar dinámicamente. Si las direcciones sólo se reubican en el momento de la carga ¿es posible compactar la memoria?
- **Swapping / Intercambio:** Con base en las políticas de planificación de la CPU. Memoria o almacenamiento auxiliar, y de éste a memoria ppa. Permite ejecutar más procesos de los que pueden caber en la memoria a la vez.
- **Compartir:** Diferentes usuarios comparten código y datos, aumenta la multiprogramación. Generalmente requiere usar paginación o bien segmentación.
- **Protección:** Al usar paginación o segmentación, se pueden declarar diferentes secciones de un programa como sólo para ejecución, para lectura o de lectura/escritura. Esta restricción es necesaria si se comparten códigos o datos.
