Sistemas Operativos

Administración de Memoria II

Curso 2024

Facultad de Ingeniería, UDELAR

Agenda

1. Direccionamiento físico y virtual
2. Segmentación
3. Paginación
4. Segmentación vs. paginación
5. Segmentación con paginación

Direccionamiento físico y virtual

Direcciones lógicas y físicas

Dirección lógica: dirección generada por la CPU.

Dirección física: dirección manejada por la unidad de memoria (cargada en el registro de direcciones de memoria).

La asignación de direcciones en tiempo de ejecución resulta en direcciones lógicas y físicas diferentes. La dirección lógica pasa a ser una *dirección virtual*. Los espacios de direcciones lógicas y físicas difieren.

Direccionamiento físico y virtual

El mapeo en tiempo de ejecución de direcciones virtuales a físicas lo realiza la Unidad de Administración de Memoria (MMU).

Existen varios métodos para el mapeo de direcciones. Métodos simples generalizan el esquema de registro base y límite, utilizando un registro de reubicación. El valor en el registro de reubicación se agrega a cada dirección generada por un proceso en el momento en que la dirección se envía a la memoria.

Otras técnicas permiten que el espacio de direcciones lógicas de los procesos sea no contiguo: pueden asignar memoria física a un proceso siempre que esté disponible.

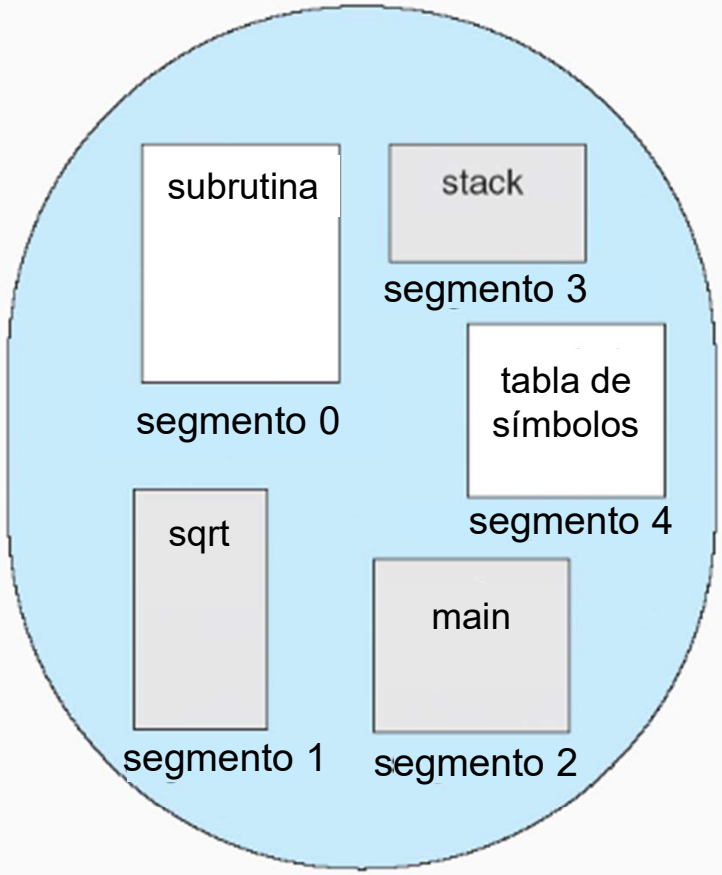
Segmentación

Segmentación

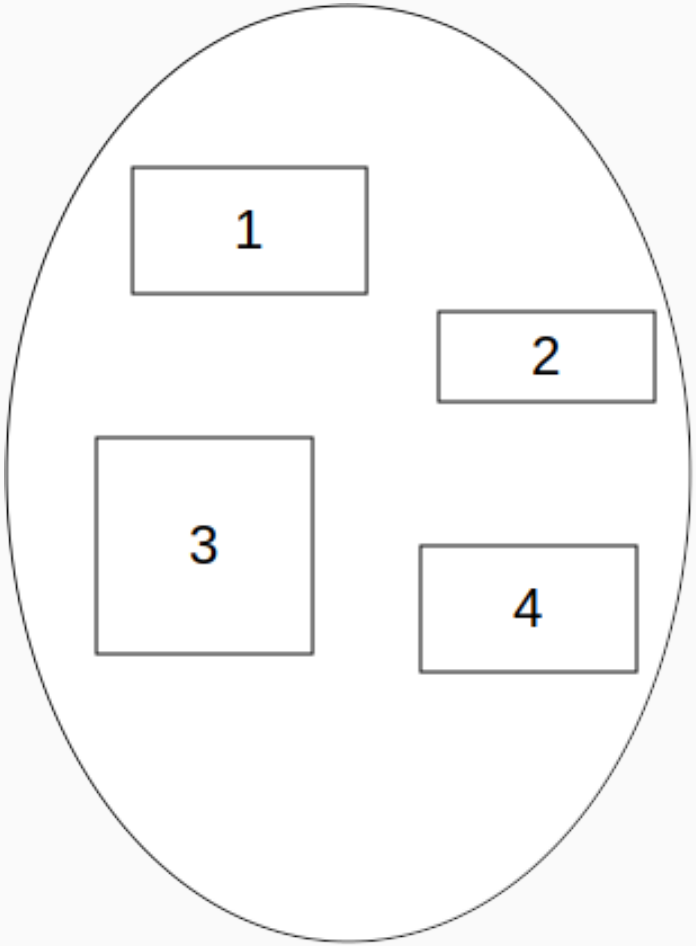
La segmentación asigna segmentos contiguos de memoria para las áreas de memoria de un proceso.

Se adapta a la visión de la memoria que tiene el usuario: el espacio físico se divide en segmentos (no contiguos entre sí).

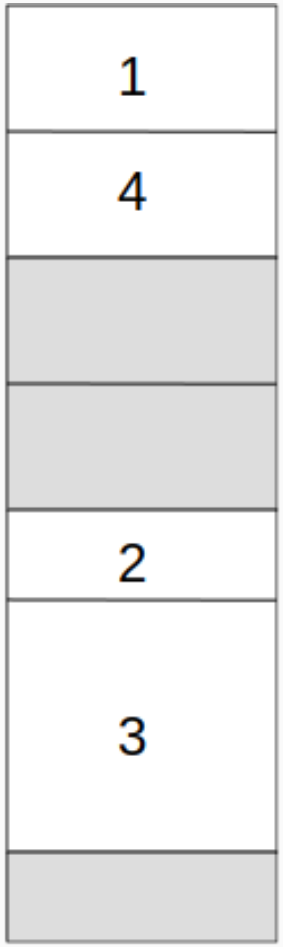
Cada área de un proceso (código, stack, heap, tabla de símbolos, etc.) se asigna a un segmento del tamaño necesario.



Segmentación: visión del usuario y memoria



Visión del usuario

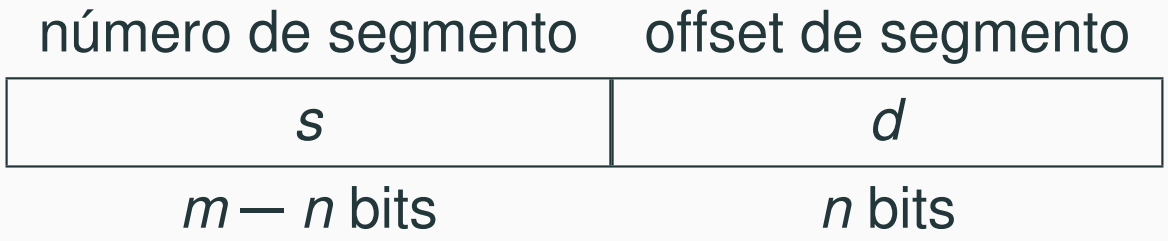


Memoria principal

Segmentación: direccionamiento

Cada segmento tiene un identificador (número) y un largo asociado.

Las direcciones virtuales se componen de un número de segmento y el desplazamiento dentro del segmento.



El desplazamiento debe ser menor que el largo del segmento.

Se requiere un método para mapear direcciones de segmento (bidimesionales) en direcciones físicas (unidimesionales).

Mapeo de direcciones

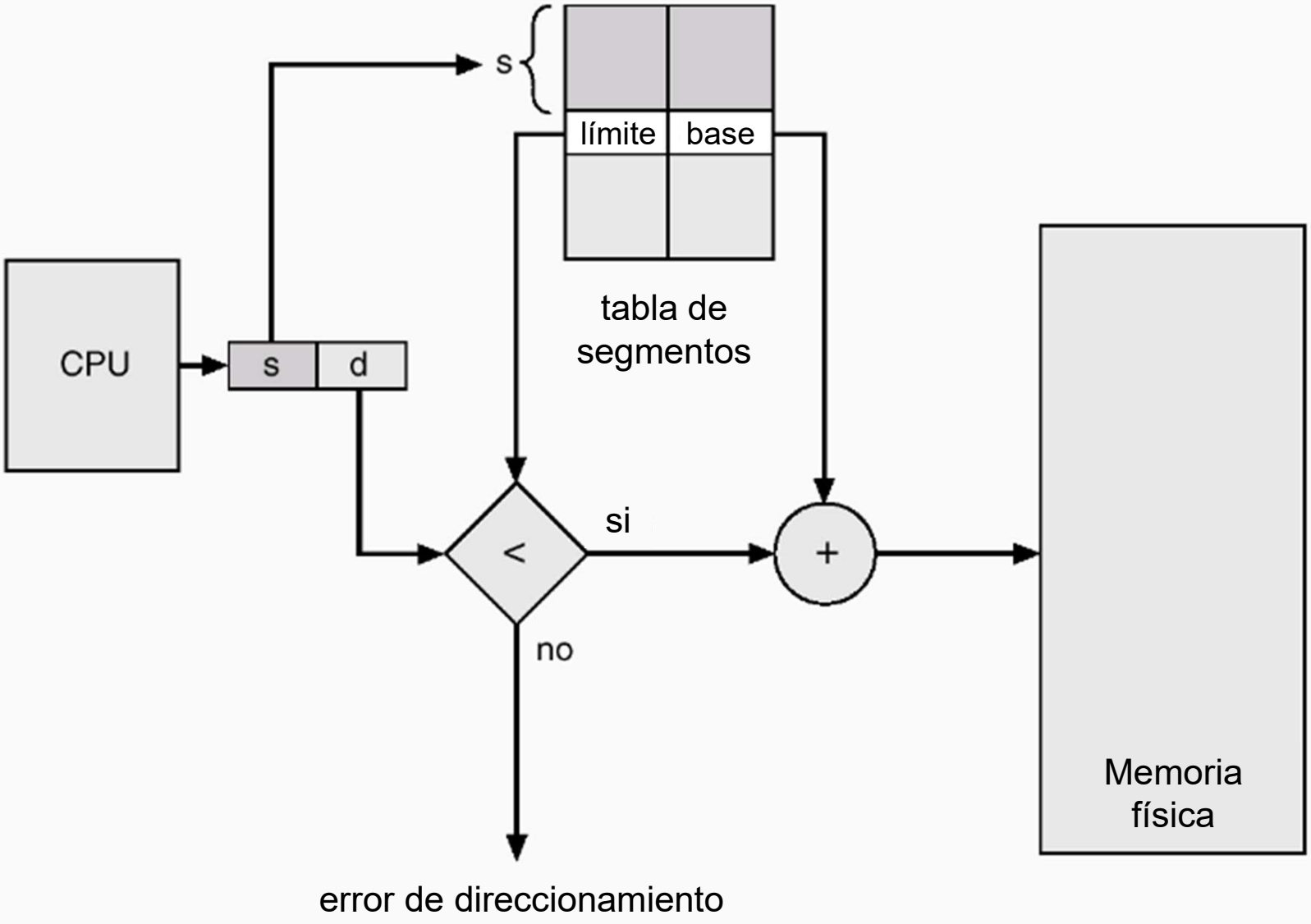
Se utiliza una tabla de segmentos que tiene una entrada por cada segmento, incluyendo la dirección física base del segmento (*base register*) y el largo del segmento (*limit register*).

En la traducción de dirección virtual a física se controla:

1. el número de segmento con la cantidad de segmentos utilizados por el proceso, definido por el registro Segment Table Length Register (STLR), y
2. el desplazamiento con el registro límite del segmento

La tabla de segmentos se mantiene en memoria principal y se asigna un registro que apunta a la dirección base de la tabla (Segment Table Base Register, STBR).

Soporte a nivel de hardware



Ejemplo de segmentación

Cinco segmentos numerados del 0 al 4, almacenados como se presenta en el diagrama.

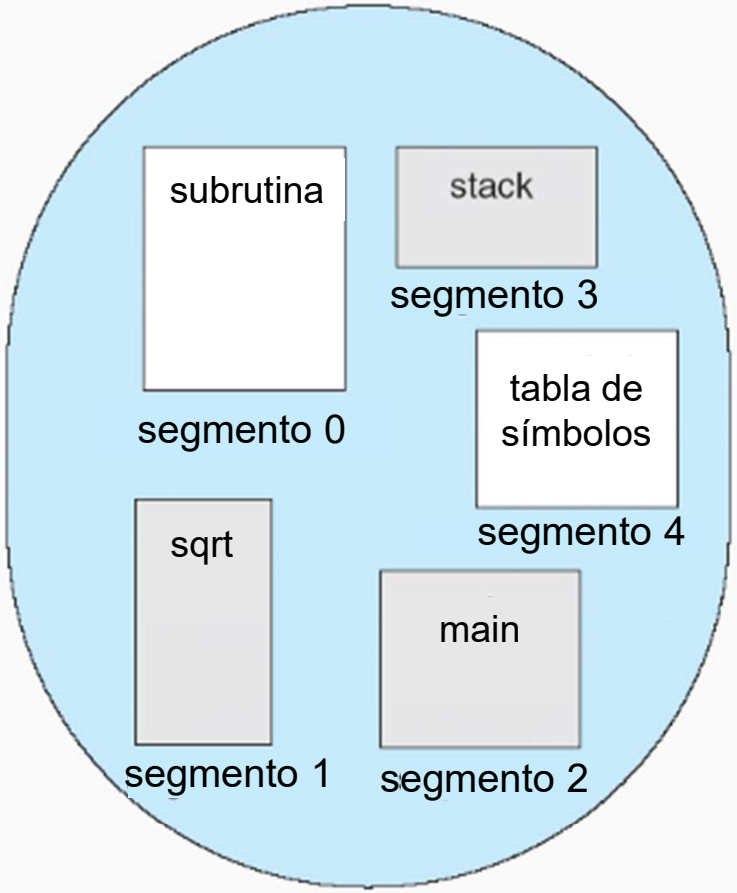
La tabla de segmentos tiene una entrada para cada segmento, que indica la dirección inicial del segmento en la memoria física (base) y la longitud del segmento (límite).

El segmento 2 tiene una longitud de 400 B y comienza en la ubicación 4300. Una referencia al byte 53 del segmento 2 se asigna a la dirección $4300 + 53 = 4353$.

Una referencia al byte 852 del segmento 3 se asigna a la dirección 3200 (base del segmento 3) + 852 = 4052.

Una referencia al byte 1222 del segmento 0 resulta en un error, ya que el segmento 0 tiene solo 1000 B de largo

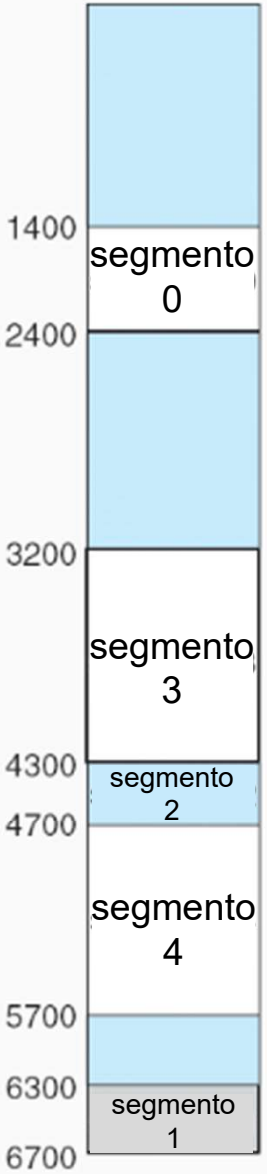
Ejemplo de segmentación



espacio de
direcciones lógicas

	límite	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

tabla de
segmentos



memoria física

Protección de memoria

Dividir la memoria en segmentos permite asociar a cada segmento un conjunto de permisos.

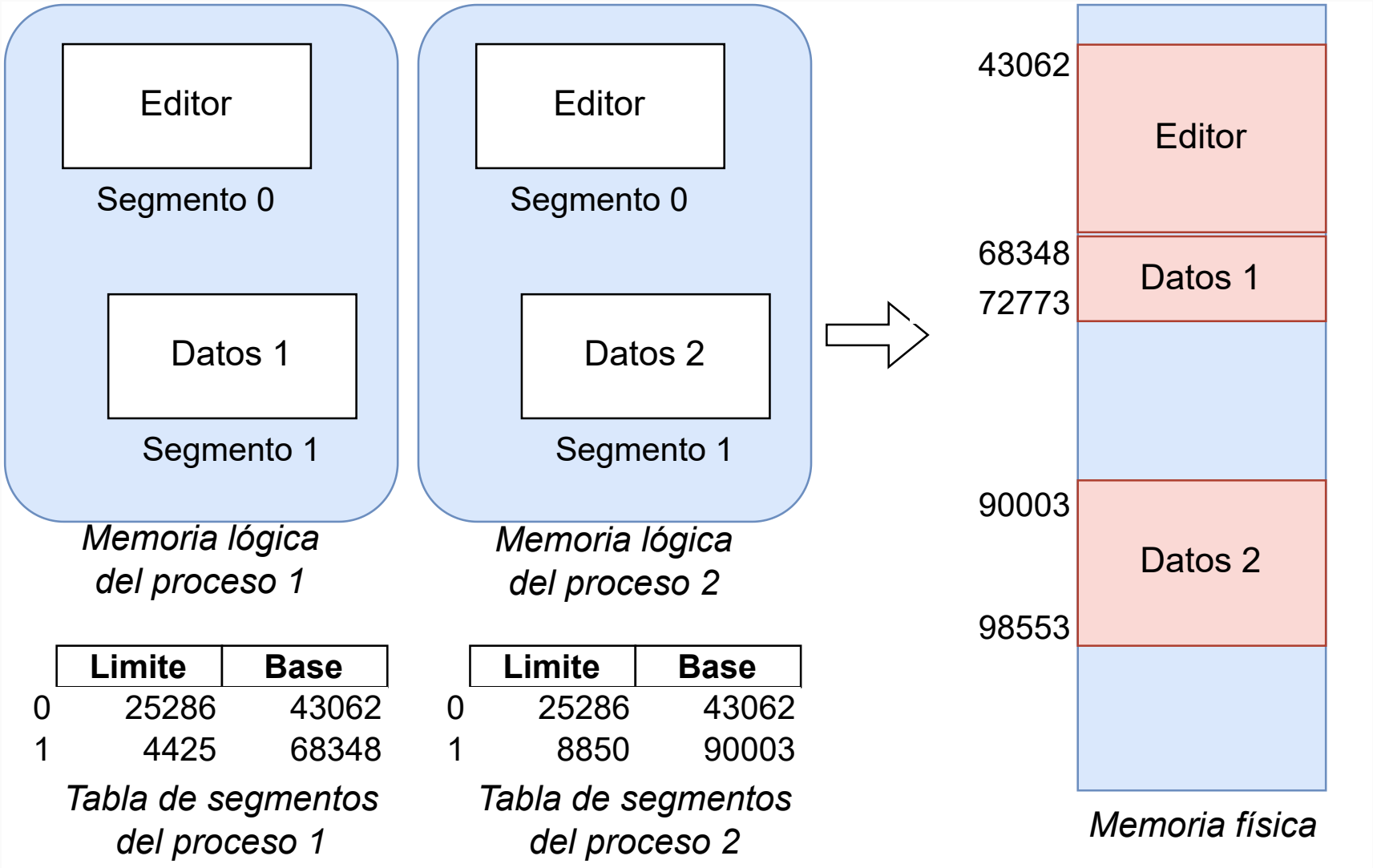
Lo habitual es que un segmento de código tenga permisos de lectura y ejecución y no de escritura.

Un segmento de datos tiene permisos de lectura y escritura.

Se define un conjunto de bits de protección (protection bits) controlados por el hardware.

Compartir memoria/segmentos

Los segmentos proporcionan una forma clara y sencilla para compartir memoria entre varios procesos.



	Limite	Base
0	25286	43062
1	4425	68348

Tabla de segmentos del proceso 1

	Limite	Base
0	25286	43062
1	8850	90003

Tabla de segmentos del proceso 2

Paginación

Paginación

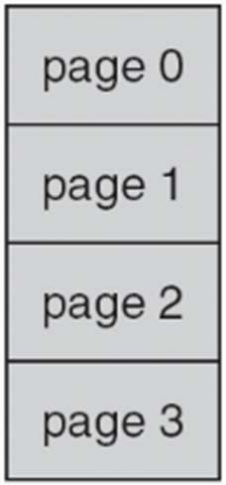
La paginación es una técnica que divide a la memoria física en particiones de tamaño fijo llamados frames.

A su vez, el espacio de direccionamiento virtual se divide en unidades fijas denominadas páginas (*pages*), del mismo tamaño que los frames (*page size*).

Las páginas tienen un tamaño potencia de 2, entre 512 B y 16 MB.

La paginación se utiliza en la mayoría de los sistemas operativos (desde los de mainframe hasta los de teléfonos inteligentes).

Paginación: modelo de memoria



Memoria lógica

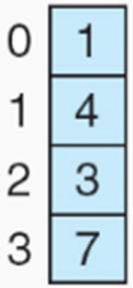
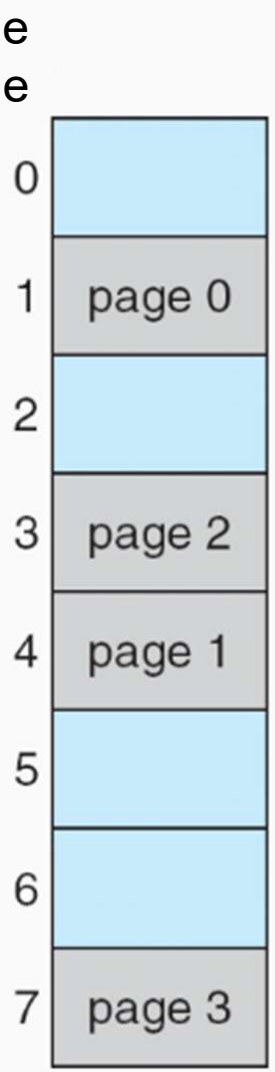


Tabla de páginas



Memoria física

Paginación

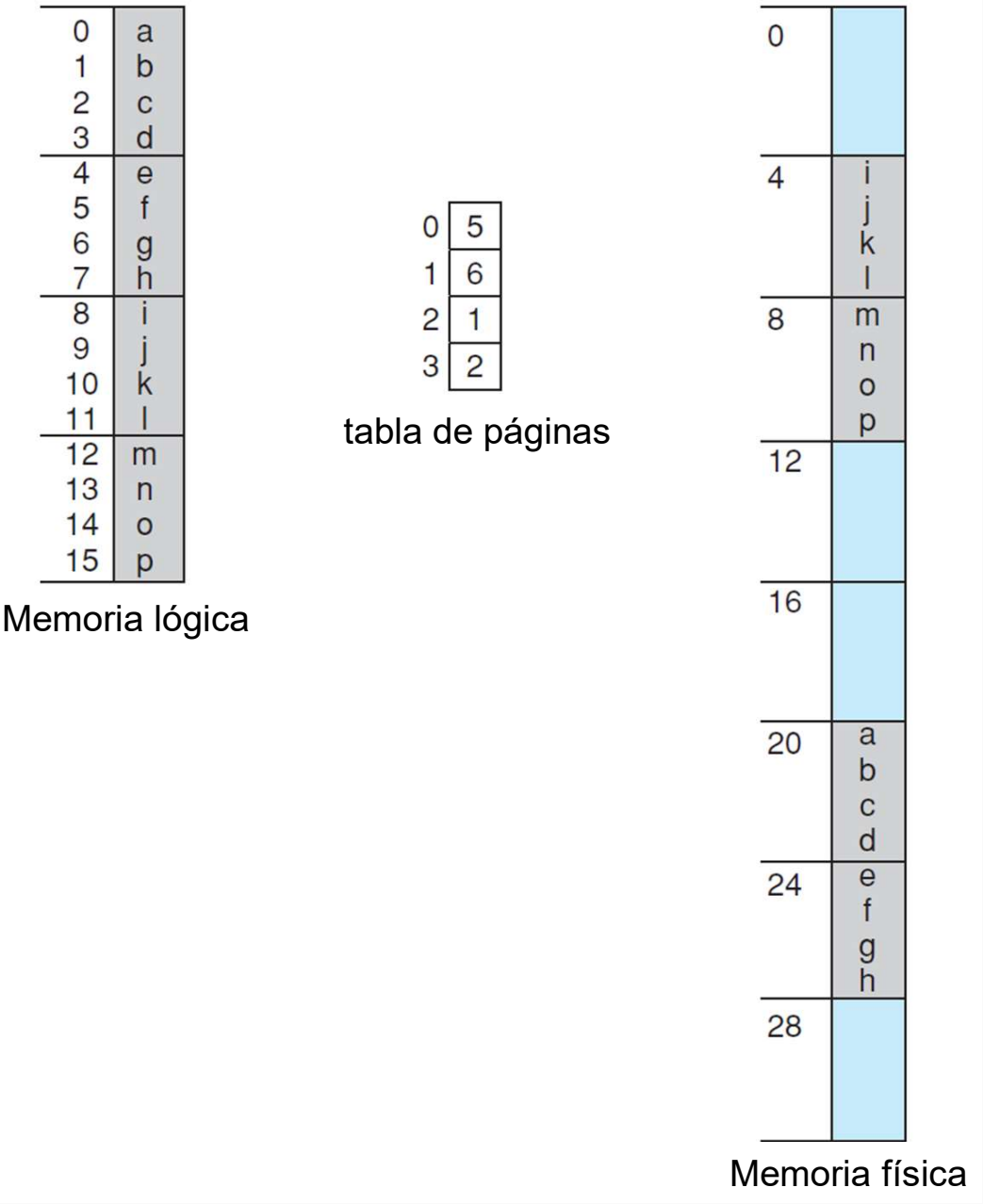
La transferencia entre la memoria principal y memoria secundaria es siempre en unidades de página.

Cuando un proceso ejecuta, sus páginas son cargadas en los frames de memoria principal y secundaria (sección de swap).

Los frames en el swap tienen el mismo tamaño que los frames de memoria principal.

El espacio de direcciones lógicas se separa del espacio de direcciones físicas. Un proceso puede tener un espacio de direcciones lógicas de 64 bits, aunque el sistema no tenga 2^{64} B de memoria física.

Ejemplo de paginación



Paginación: direccionamiento

La paginación se implementa a través de la cooperación entre el sistema operativo y el hardware.

Es un tipo de reubicación dinámica. Cada dirección lógica está vinculada por el hardware de paginación a una dirección física.

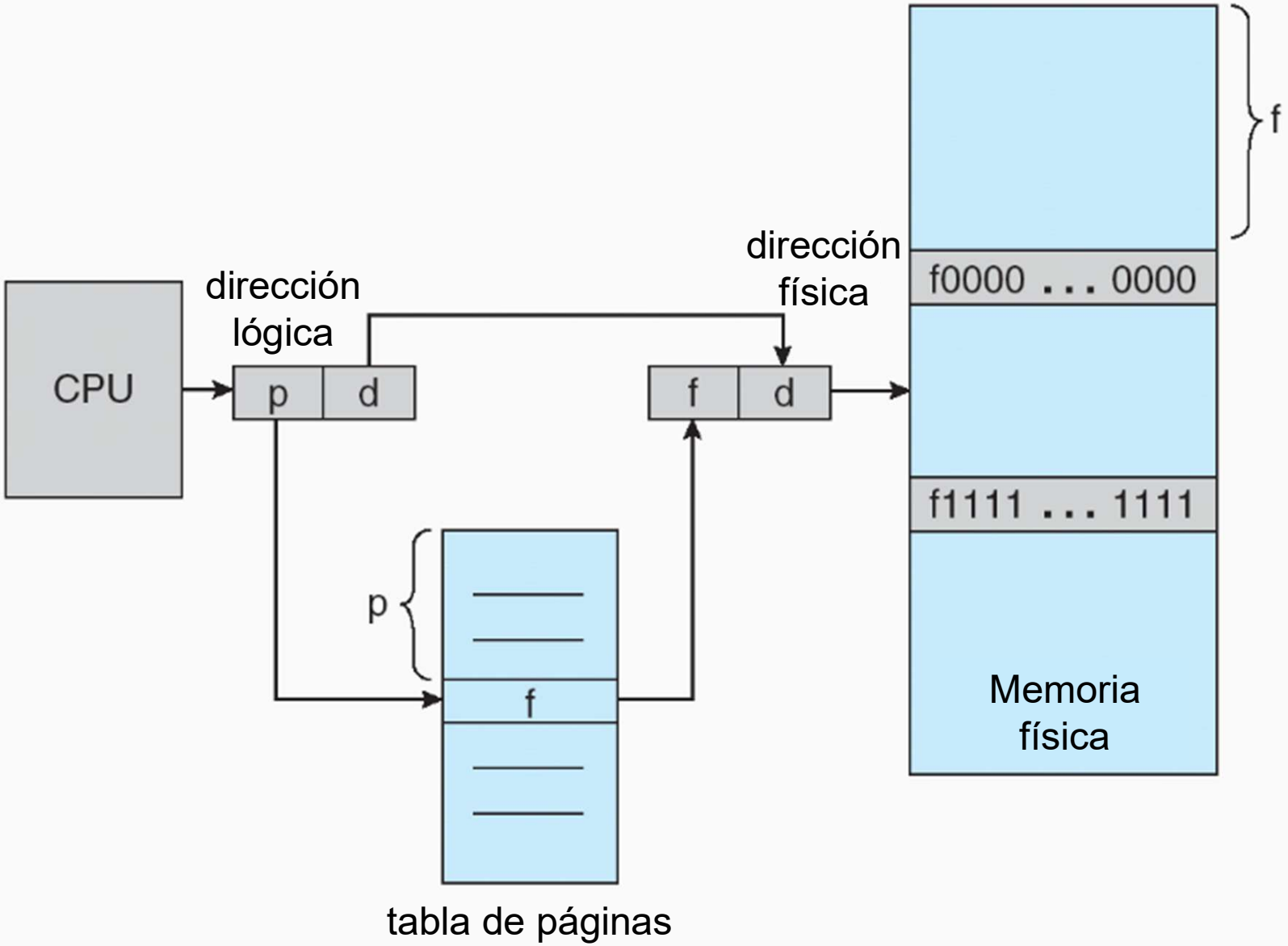
Las direcciones lógicas/virtuales se componen de un número de página (*page number*) y un desplazamiento (*offset*).

El número de página es un índice sobre una tabla de páginas (*page table*) y el desplazamiento es la referencia dentro del frame.

La tabla de página contiene el mapeo del frame correspondiente.

El tamaño de página/frame es 2^n.

Paginación: soporte de hardware y direccionamiento



Paginación: tamaño de paginas y memoria física

En CPUs actuales (32/64 bits), cada entrada de la tabla de páginas tiene una longitud de 4 B/8 B. Una entrada de 32/64 bits puede apuntar a uno de los $2^{32}/2^{64}$ frames en memoria física.

Para un tamaño de frame de 4 KB (2^{12}), un sistema de 32 bits con entradas de 4 B puede direccionar 2^{44} B (16 TB) de memoria física, aunque el espacio de direccionamiento del proceso es de 2^{32} B (4 GB).

La paginación permite usar una memoria física mayor a la que puede ser direccionada por la CPU.

Paginación y fragmentación

La paginación evita la fragmentación externa y elimina la necesidad de compactación.

También resuelve el problema de colocar fragmentos de memoria de diferentes tamaños en el swap (problema de fragmentación del swap). La compactación no es aplicable (en la práctica) en el swap, porque el acceso es muy lento.

Paginación y fragmentación

La paginación no resuelve la fragmentación interna.

Los frames se asignan como unidades. Si la memoria que requiere un proceso no coincide con los límites de la página, el último frame asignado no estará lleno.

Ejemplo: si el tamaño de página es 2048 B, un proceso de 72766 B necesitará 35 páginas más 1086 B. Se le asignarán 36 frames y se tendrá una fragmentación interna de 2048–1086 = 962 B.

En el peor caso, un proceso necesitará n páginas más 1 B. Se le asignarán $n + 1$ frames, resultando en una fragmentación interna de casi un frame completo.

Paginación, fragmentación y tamaño de paginas

Si el tamaño del proceso es independiente del tamaño de página, el valor esperado de la fragmentación interna es de media página por proceso. Los tamaños de página pequeños son deseables.

Sin embargo, existe una sobrecarga asociada a cada entrada de la tabla de páginas, que se reduce a medida que aumenta el tamaño de página. Además, la E/S del disco es más eficiente cuando la cantidad de datos que se transfieren es mayor.

Los tamaños de página han crecido a medida que los procesos, los datos y la memoria principal se han incrementado. Actualmente, el tamaño de página es de 4 KB u 8 KB. Algunos sistemas operativos admiten tamaños de página mayores y/o varios tamaños de página.

Implementación

La mayoría de los sistemas operativos asignan una tabla de páginas por proceso: se agrega al PCB del proceso un puntero al comienzo de la tabla de páginas.

En hardware, la opción más simple es usar registros dedicados para alojar la tabla de páginas, con lógica de alta velocidad para que la traducción de direcciones sea eficiente. El despachador recarga estos registros mediante instrucciones privilegiadas. Solo el sistema operativo puede cambiar el mapa de memoria.

El acceso a la tabla de páginas es muy rápido, pero la cantidad de entradas es muy limitada.

Soporte a nivel de hardware: referencia a la tabla de páginas

En los sistemas actuales no es posible guardar todas las entradas en registros de rápido acceso.

La tabla se mantiene en memoria principal y se asigna un registro que apunta a la dirección base de la tabla (Page Table Base Register, PTBR).

Cambiar las tablas de páginas requiere cambiar solo el PTBR, reduciendo sustancialmente el tiempo de cambio de contexto

Sin embargo, el acceso a memoria se duplica: es necesario acceder a la tabla para obtener el número de frame y posteriormente al lugar de memoria solicitado.

Soporte a nivel de hardware: cache de la tabla de páginas

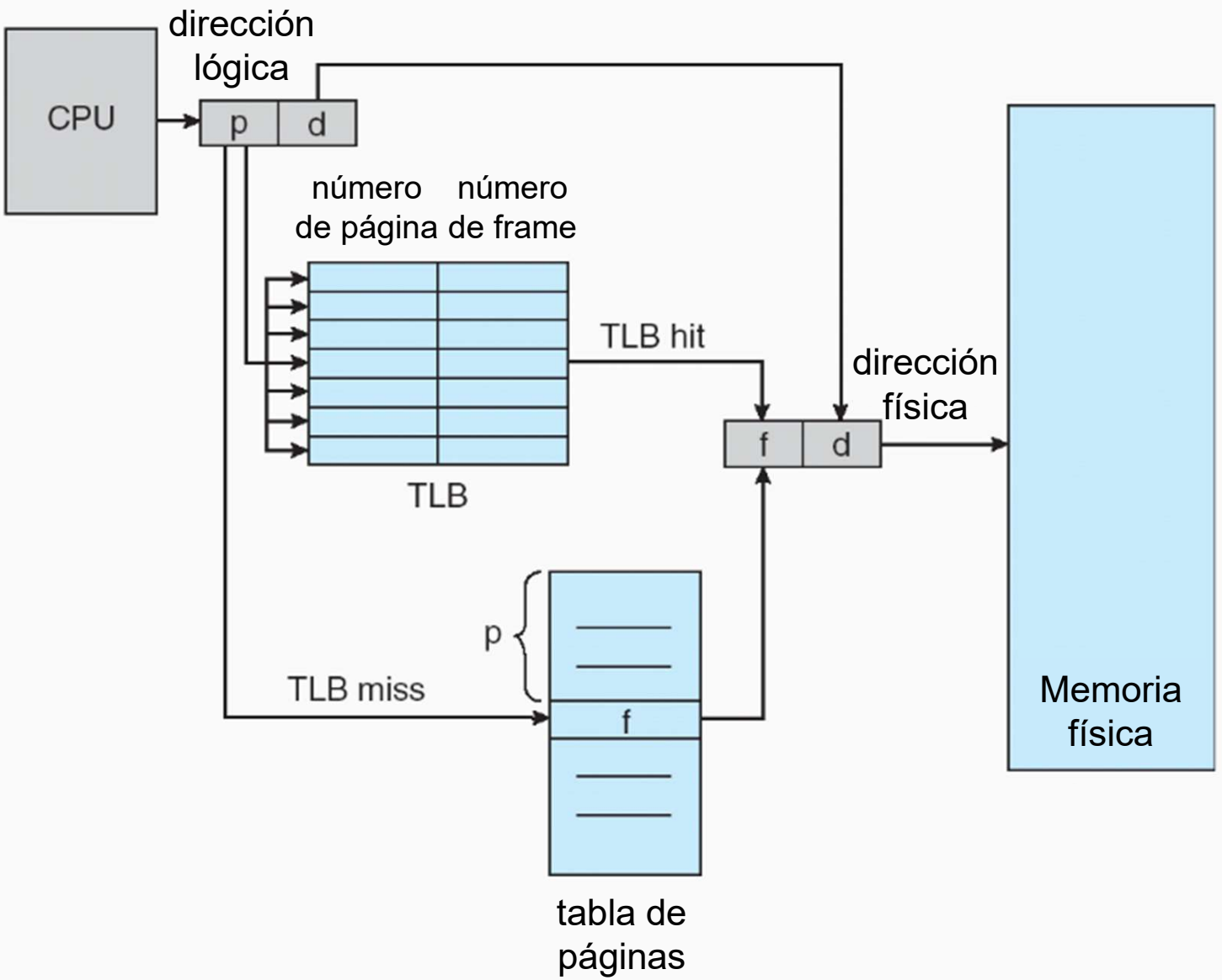
Se utiliza una pequeña cache de la tabla de página: Translation Look-aside Buffer, TLB. Asociativa y de rápido acceso.

Cada entrada consiste de dos partes: una clave (tag) y un valor (el número de frame). La búsqueda de una clave en la cache TLB es simultánea entre todas las tags.

Si se encuentra la clave (TLB hit), inmediatamente se genera la dirección buscada a partir del valor asociado.

En caso contrario (TLB miss), debe accederse a memoria para obtener la entrada. Posteriormente, se guarda el valor obtenido en la cache TLB para posteriores accesos (principio de localidad).

Soporte a nivel de hardware: cache de la tabla de páginas



Soporte a nivel de hardware: cache de la tabla de páginas

Una búsqueda en la TLB en un hardware moderno es parte del pipeline de instrucciones: no penaliza el rendimiento. Sin embargo, para poder ejecutar la búsqueda dentro de una etapa del pipeline, la TLB debe ser pequeña (entre 32/64 y 1024 entradas).

Algunas CPU implementan TLB independientes para direcciones de instrucciones y de datos, y logran duplicar la cantidad de entradas disponibles (las búsquedas ocurren en diferentes etapas del pipeline).

Soporte a nivel de hardware: cache de la tabla de páginas

Algunas caché TLB agregan a cada entrada un identificador de espacio de direccionamiento (Address Space IDentifier, ASID).

En la búsqueda de una clave solo se toman en cuenta las entradas cuyo ASID coincide con el del proceso en ejecución.

El uso del identificador permite que la cache TLB contenga entradas para varios procesos de forma simultánea.

Si no se utiliza el ASID, en cada cambio de contexto se debe limpiar las entradas de la TLB, para no realizar accesos equivocados a memoria.

Eficiencia de la TLB: tiempo efectivo de acceso

Tiempo efectivo de acceso (Effective Access Time, EAT).

El porcentaje de veces que un número de página es encontrado en la cache TLB se denomina hit ratio.

El tiempo efectivo de acceso se define por

$$
\begin{aligned}
EAT = &\text{ tiempo de búsqueda en TLB} \\
&+ (\text{hit ratio} \times \text{tiempo de acceso a memoria}) \\
&+ (1 - \text{hit ratio}) \times (2 \times \text{tiempo de acceso a memoria})
\end{aligned}
$$

EAT determina la ganancia de la utilización de la caché TLB.

Paginación: protección de memoria

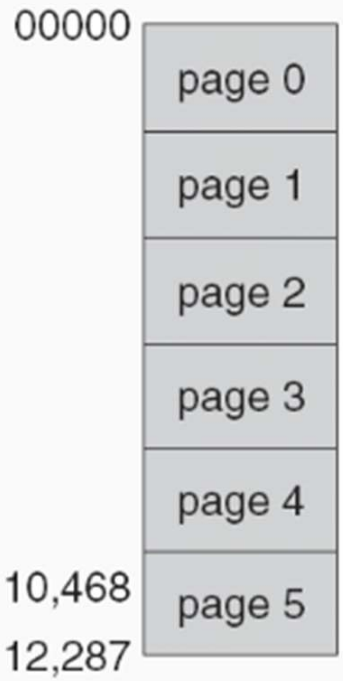
La tabla de páginas tiene una entrada por cada posible página que tenga el proceso.

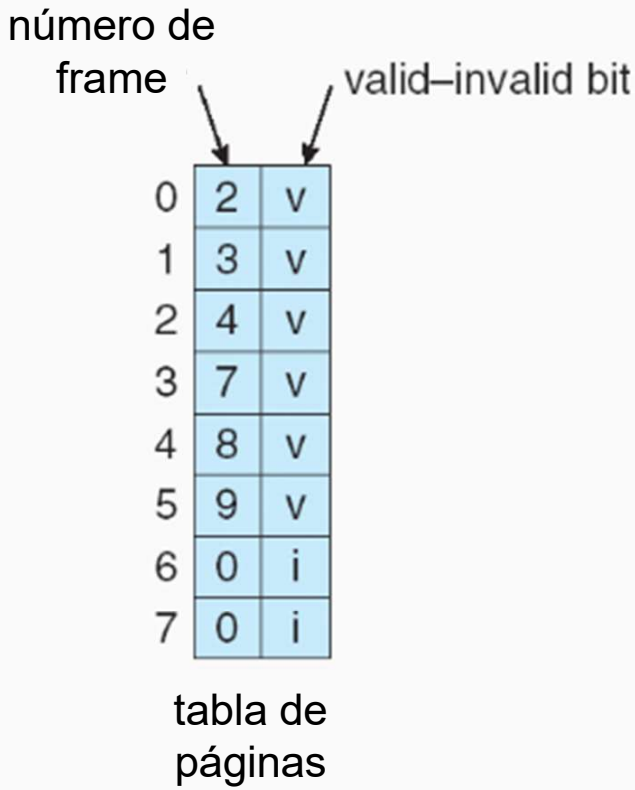
Es necesario identificar cuales son entradas válidas y cuales no.

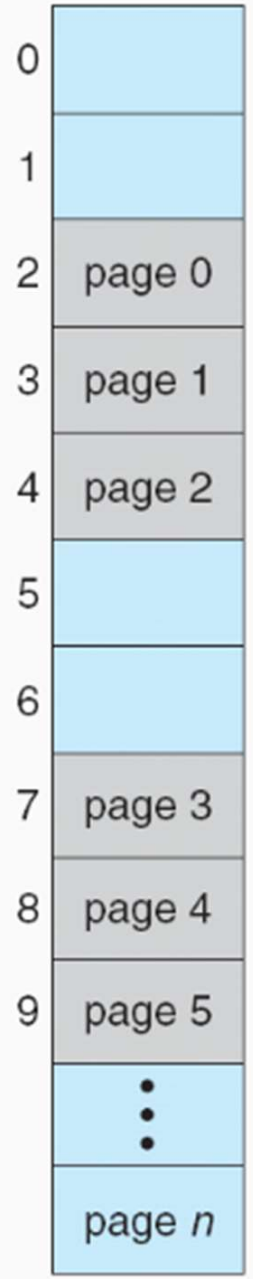
Se utiliza un bit de protección en cada entrada (valid-invalid bit) para determinar si la página es válida o inválida.

El acceso a una página cuyo bit marca que es inválida genera una excepción a nivel del sistema operativo.

Paginación: protección de memoria







Estructura de la tabla de páginas

Los sistemas modernos tiene un gran espacio de direcciones lógicas (2^{32} a 2^{64}).

En un sistema de 32 bits que usa páginas de 4 KB se necesita más de un millón ($2^{32}/2^{12}$) de entradas en la tabla de páginas. Si las entradas son de 4 B, cada proceso necesita 4 MB para almacenar su tabla.

Es necesario utilizar una estructura más eficiente, que ocupe menos espacio para almacenar la tabla de páginas.

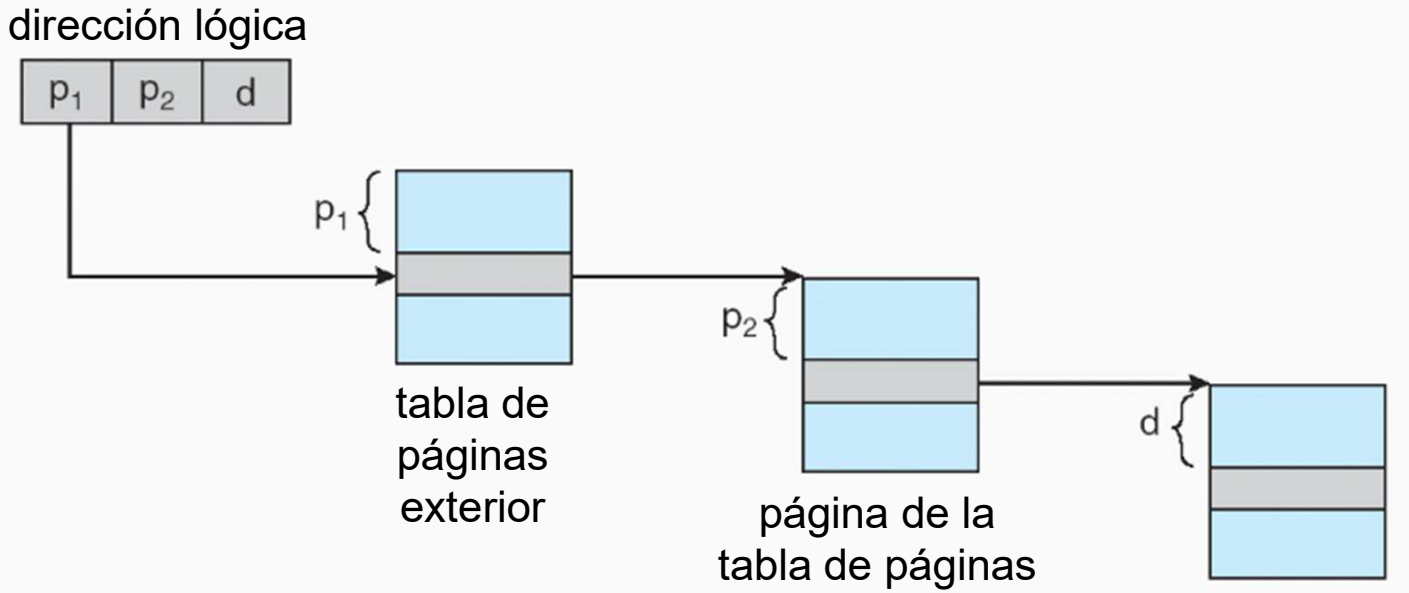
Se han propuesto varias estructuras para la tabla de páginas:

- Jerárquica
- Diccionarios (*hash*)
- Invertida

Estructura de la tabla de páginas: Jerárquica

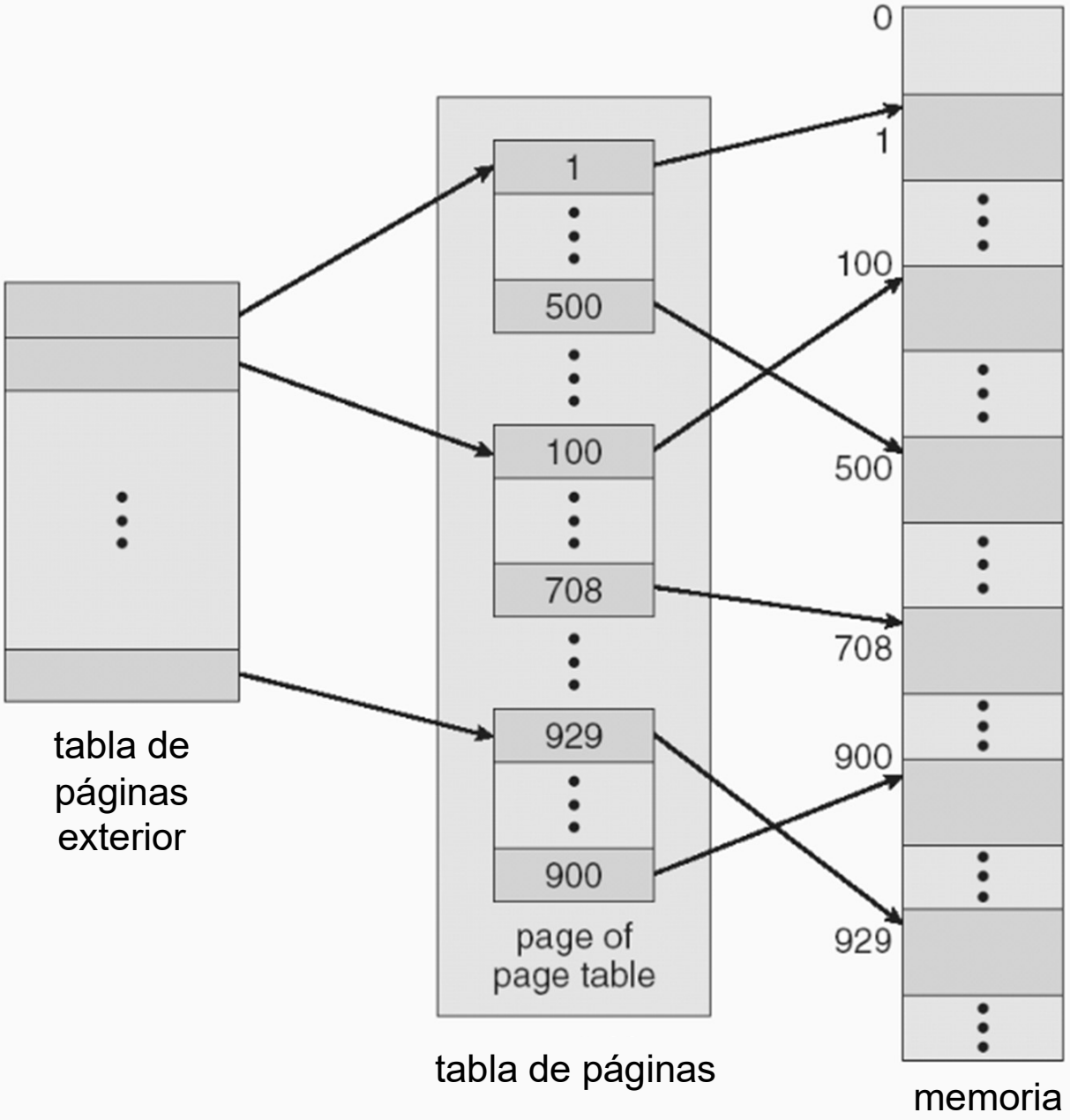
La estructura se basa en dividir la tabla de páginas, definiendo una jerarquía.

Se separa el campo de indexado de la dirección virtual en varios niveles (e.g., sección, página y offset).



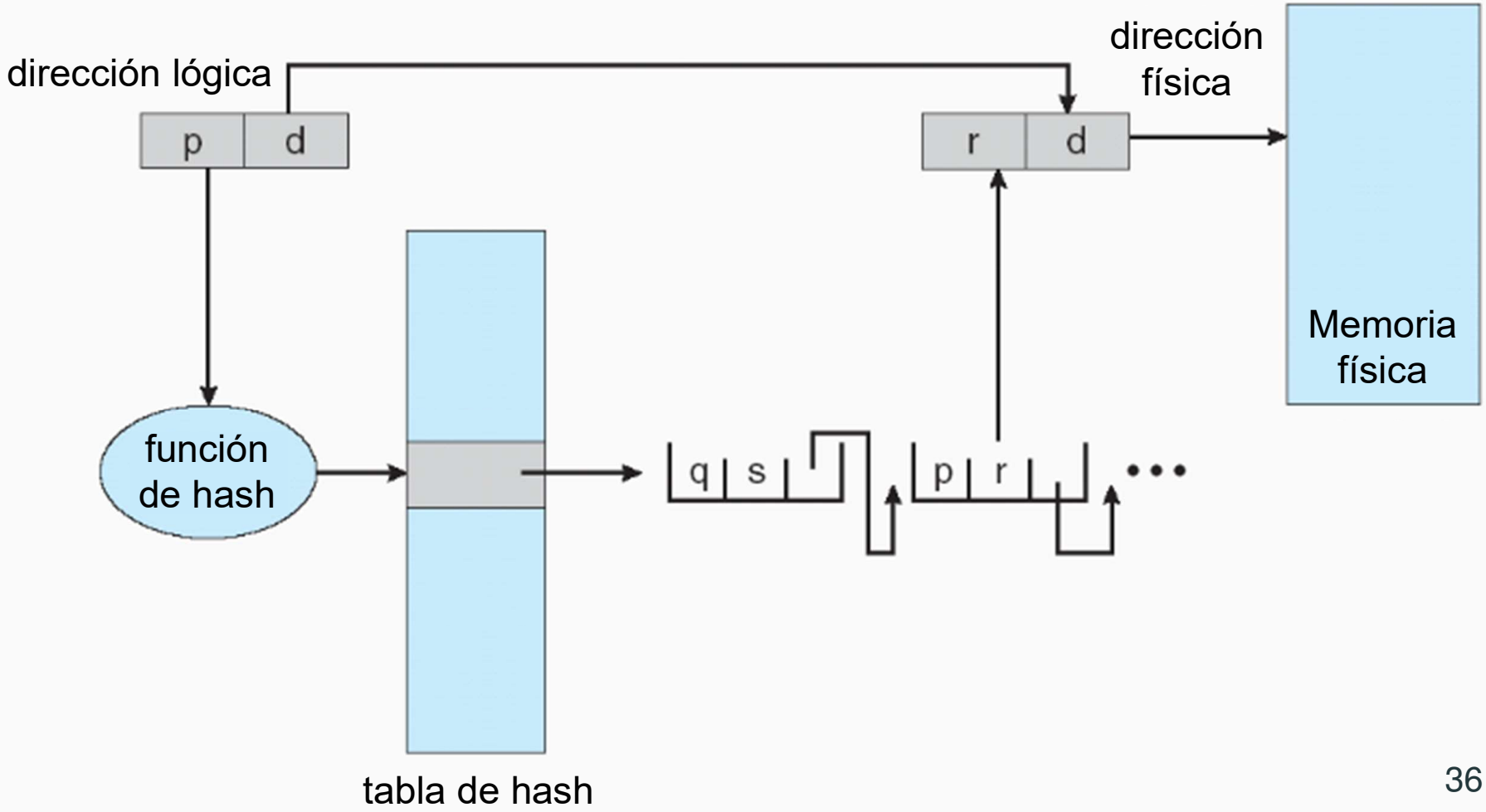
El esquema se denomina forward-mapped page table y se puede aplicar con múltiples niveles de indexación.

Estructura de la tabla de páginas: jerárquica



Estructura de la tabla de páginas: diccionarios

Consiste en implementar una tabla de hash abierto con el valor de la componente de número de página. Es conveniente para arquitecturas de más de 32 bits.



Estructura de la tabla de páginas: diccionarios

El valor hasheado es el número de página.

Cada entrada en la tabla de hash es una lista de elementos que tienen un hash en la misma ubicación (para manejar colisiones).

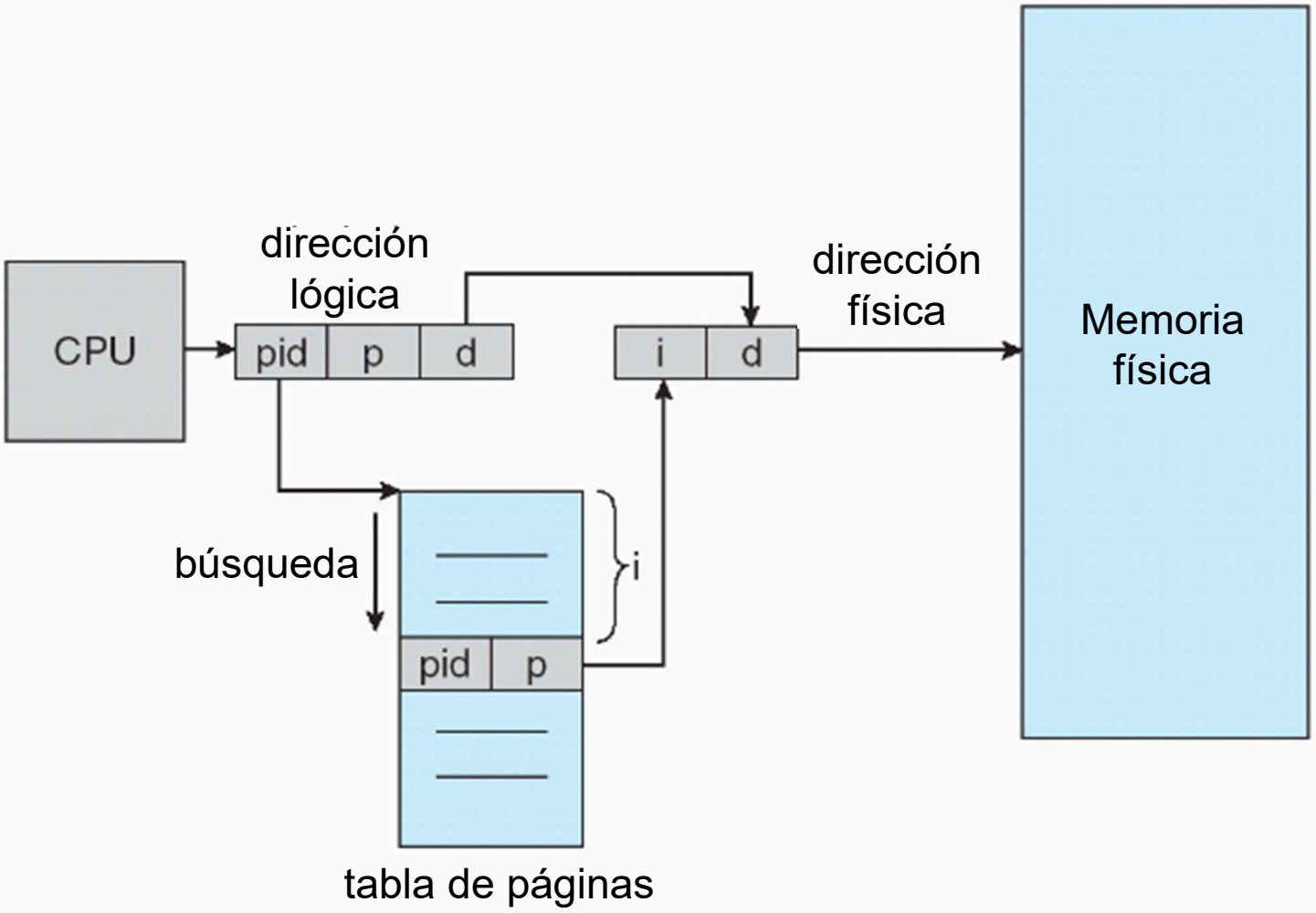
Cada elemento tiene tres campos: (1) el número de página virtual, (2) el valor del frame correspondiente y (3) un puntero al siguiente elemento en la lista.

El número de página en la dirección lógica se convierte con la función de hash y se usa para acceder a la tabla de hash.
Se compara el número de página con el primer campo en cada elemento de la lista. Cuando hay una coincidencia, se usa el número de frame (campo 2) para formar la dirección física.

Estructura de la tabla de páginas: invertida

Se usa una única tabla de páginas global a todos los procesos.

Se aumentan los tiempos de búsqueda, para reducir el espacio de almacenamiento en memoria principal.



Compartir memoria entre procesos

Los procesos se componen de una parte privada de código y datos y de otras partes que pueden ser compartidas.

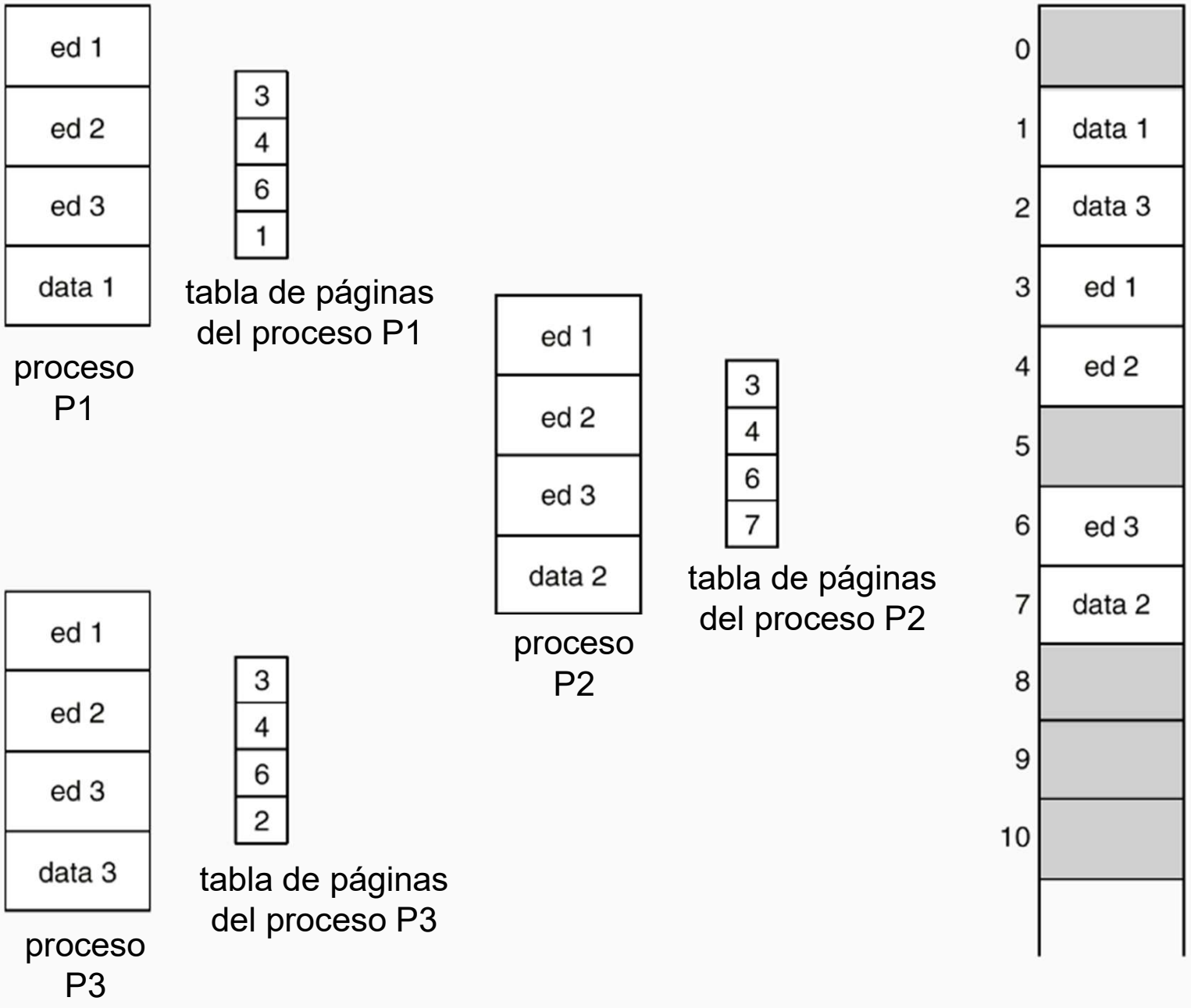
La posibilidad de dividir el espacio de direccionamiento en páginas permite a los procesos compartir de forma eficiente las páginas comunes en memoria.

Ejemplos:

- La sección de código de un mismo proceso.
- El código de una biblioteca dinámica.
- Memoria compartida.

Esta estrategia permite un uso más eficiente de la memoria.

Compartir memoria entre procesos



Segmentación vs. paginación

Segmentación vs. paginación

La segmentación logra implementar la visión que el usuario tiene de la memoria.

La segmentación sufre de fragmentación externa. Los segmentos se asignan de forma contigua, pero a medida que se liberan generan huecos que, si no se completan totalmente, generan huecos menores no utilizables. Para solucionar el problema es necesario reorganizar/compactar la memoria (tarea costosa).

En la paginación, el espacio de direccionamiento de un proceso se distribuye de forma no contigua.

La paginación genera fragmentación interna. Dado que la granularidad es a nivel de página, se generan espacios de memoria dentro de las páginas que quedan sin uso.

Segmentación vs. paginación

La segmentación logra compartir y proteger memoria entre procesos de forma más directa que la paginación.

En la paginación, compartir un espacio de direccionamiento de un proceso implica mantener una gran cantidad de referencias de páginas compartidas, mientras que en la segmentación solo se comparte el segmento.

Con la paginación la asignación de una página en memoria es más rápida. Utilizando un mapa de bits se obtiene de forma sencilla un frame libre de memoria donde puede ser asignada la página. En segmentación es necesario mantener una lista y la búsqueda se hace más costosa.

Segmentación con paginación

Segmentación con paginación

La paginación y la segmentación se pueden combinar para potenciar las ventajas de cada técnica. Un ejemplo es la arquitectura Intel (IA).

La memoria es segmentada y los segmentos se dividen en páginas. La MMU se compone de una unidad de segmentación y una unidad de paginación.

Las direcciones virtuales contienen un identificador de segmento y un desplazamiento. A partir de ellos se genera una dirección lineal (de 32 bits en IA32). Luego, la dirección es traducida a una dirección física por parte de una unidad de paginación

Segmentación con paginación