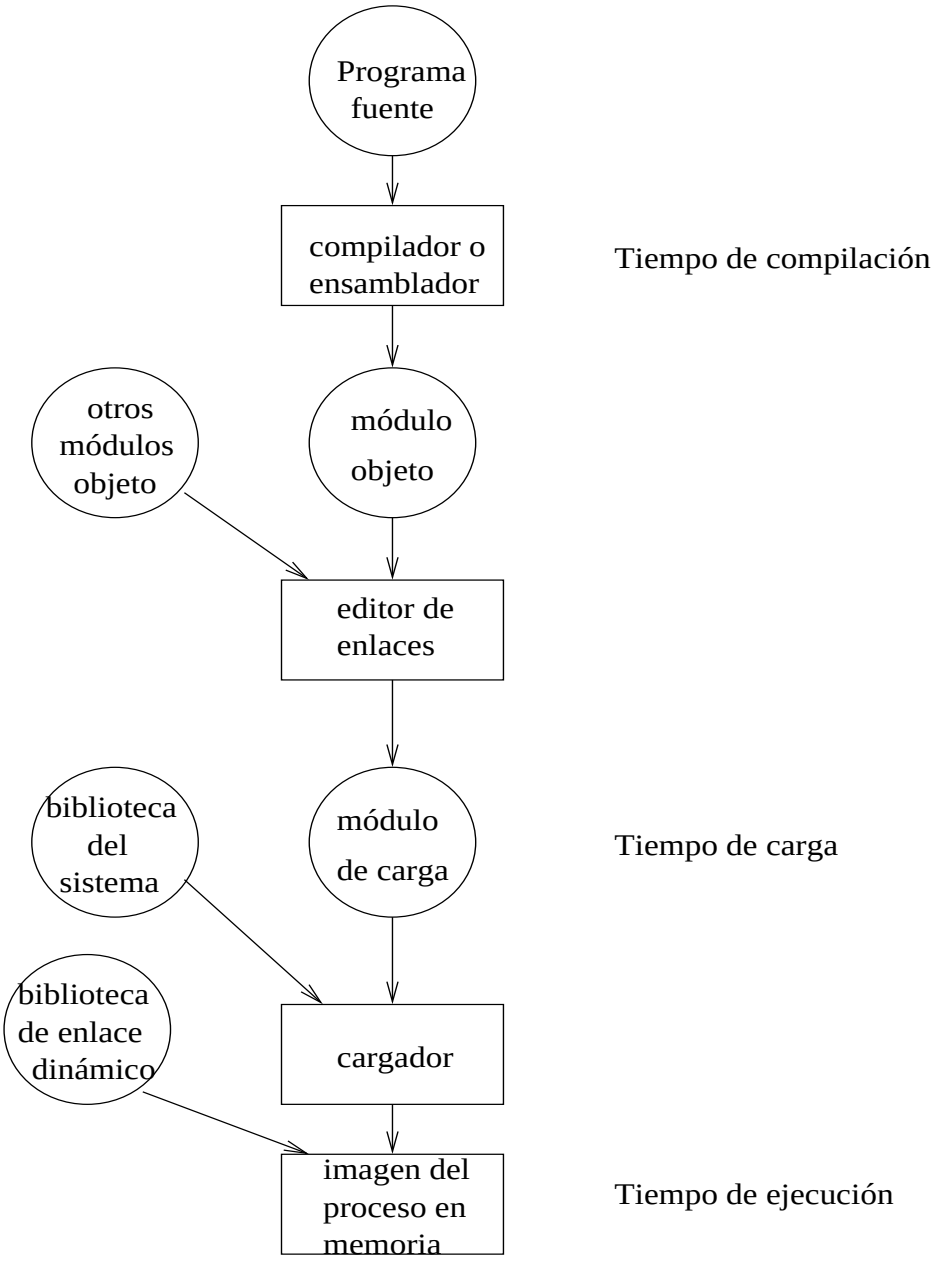
Antecedentes

- La memoria es el componente central para la operación de un sistema de cómputo moderno.
- La memoria puede ser vista como un gran arreglo de palabras o *byte*.
- La CPU acude por instrucciones a la memoria de acuerdo con el valor del contador del programa.
- Ciclo típico de instrucción-ejecución.
- Se ignora *cómo* un programa genera una dirección de memoria. Sólo nos interesa la secuencia de direcciones de memoria generadas por el programa que están en ejecución.

Vinculación de direcciones

- Un programa reside en disco como un archivo binario ejecutable.
- Para ejecutarlo debe ser llevado a memoria y colocarlo dentro de un proceso.
- Primero es puesto en la cola de entrada, manejada por el planificador de largo plazo.
- La principal tarea que debe ser ejecutada antes que un programa pueda ser ejecutado es la asignación de la localización de la memoria física en la cual las instrucciones de máquina y los valores de datos residen durante su ejecución.
- Esta asignación (*vinculación*) no es conocida hasta que el programa es cargado actualmente dentro de la memoria.
- Existen diferentes tipos de vinculación (en tiempo de compilación, en tiempo de carga y en tiempo de ejecución).
- Tipo de direcciones.
    - Absolutas.
    - Relocalizables.

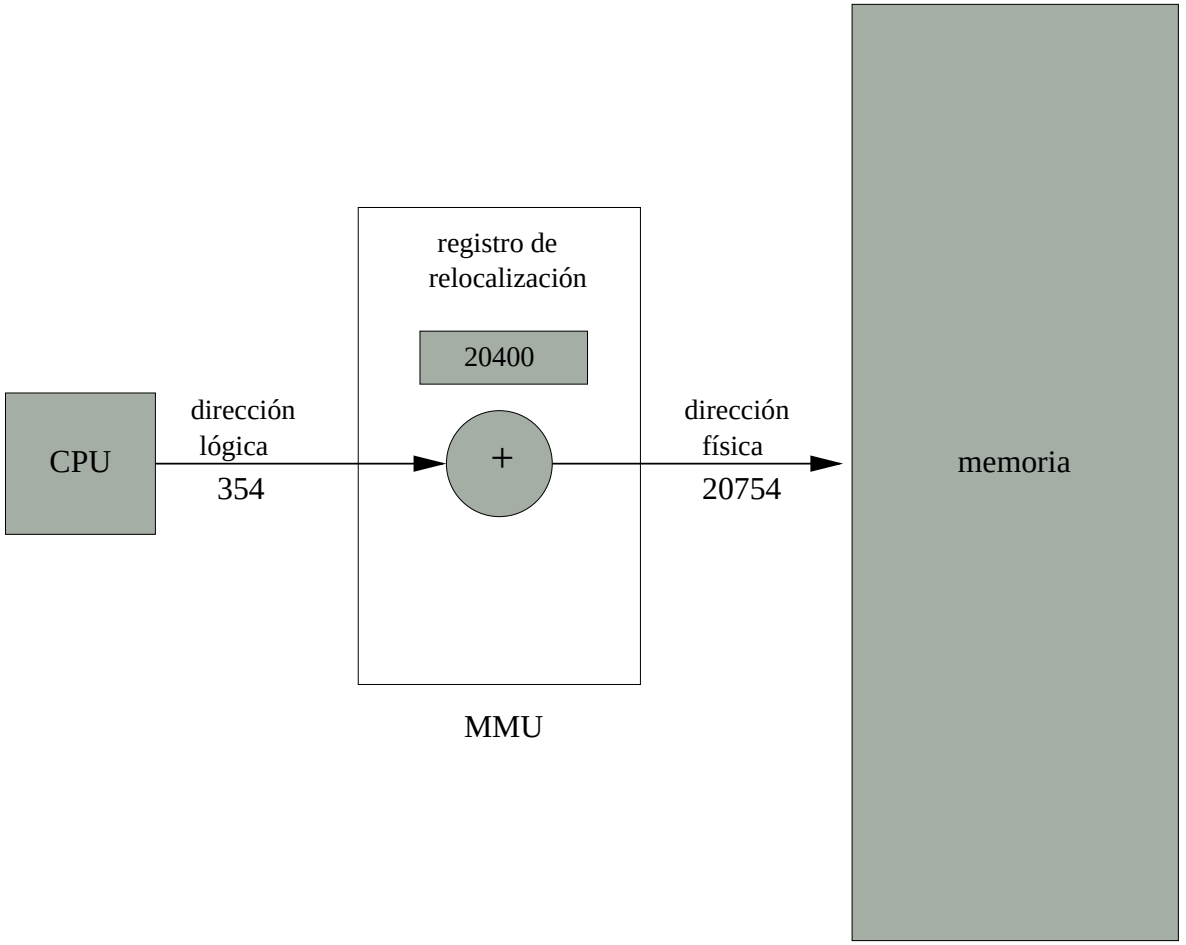
Carga de un programa de usuario



Espacio de direcciones lógicas contra físicas

- Las direcciones generadas por la CPU se conoce como **direcciones lógicas**.
- Una dirección vista por la unidad de memoria –es decir, la que se carga en la memoria en el **registro de direcciones de memoria**– se conoce como **dirección física**.
- Los métodos de vinculación de direcciones en el momento de la compilación y en el de carga dan por resultado un ambiente en el que las direcciones físicas y la direcciones lógicas son las mismas.
- En tiempo de ejecución esta vinculación hace que no sean la misma. La dirección lógica pasa a llamarse **dirección virtual**.
- El conjunto de todas las direcciones lógicas generadas por proceso es un **espacio de direcciones lógicas**.
- El conjunto de todas las direcciones físicas generadas por el *espacio de direcciones lógicas* se conoce como el *espacio de direcciones físicas*.
- El mapeo de los diferentes *espacio de direcciones* es realizado por la **unidad de administración de memoria** (*memory-management unit, MMU*).
- El hardware de mapeo de memoria convierte las direcciones lógicas en direcciones físicas.

Relocalización dinámica



Carga dinámica

- Por el momento para que un programa se ejecute completamente debe estar localizado completamente en memoria principal.
- El tamaño del proceso está limitado por el tamaño de la memoria física.
- Para conseguir una mejor utilización del espacio de memoria, se debe emplear la **carga dinámica**.
- Una rutina no se carga sino hasta que es llamada.
- No se requiere un soporte del sistema operativo.

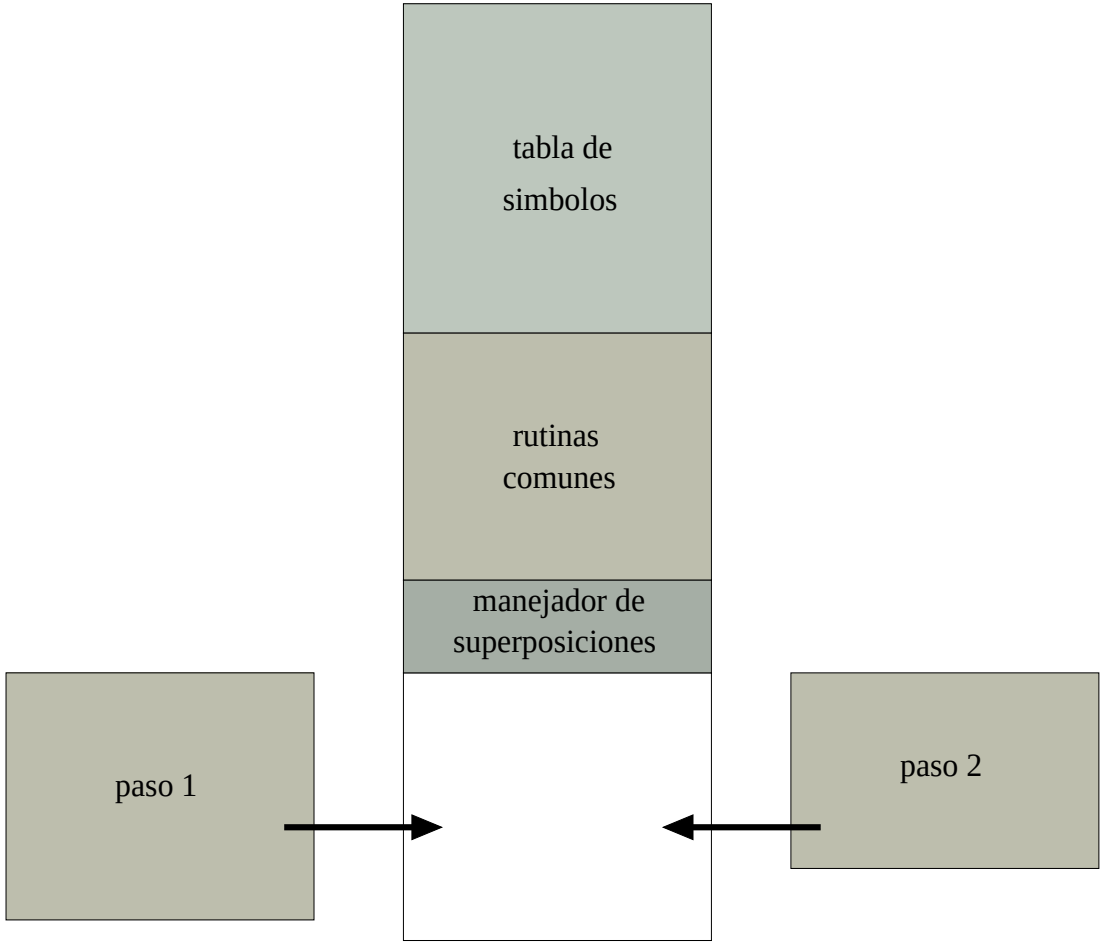
Enlace dinámico y bibliotecas compartidas

- Alguno sistemas operativos sólo soportan el **enlace estático**, en el que las bibliotecas de lenguaje son tratadas como cualquier otro módulo objeto y son combinadas por el cargador en la imagen del programa binario.
- El concepto de enlace dinámico es similar al de la carga dinámica.
- En lugar de posponer la carga hasta el tiempo de ejecución, el enlace es el que se pospone.
- Con el enlace dinámico, se incluye un **stub** en la imagen por cada referencia a la rutina de biblioteca.
- Este *stub* es una pequeña pieza de código que indica cómo localizar la rutina de biblioteca apropiada residente en memoria, o cómo se carga la biblioteca si la rutina no está ya presente.
- Requiere ayuda del sistema operativo.

Superposiciones (*overlays*)

- La idea detrás de las superposiciones consiste en mantener en memoria sólo aquellas instrucciones y datos que se necesitan en un momento dado.
- Cuando se requieren de otras instrucciones éstas se cargan en el espacio que estaba ocupado previamente por instrucciones que ya no se necesitan.

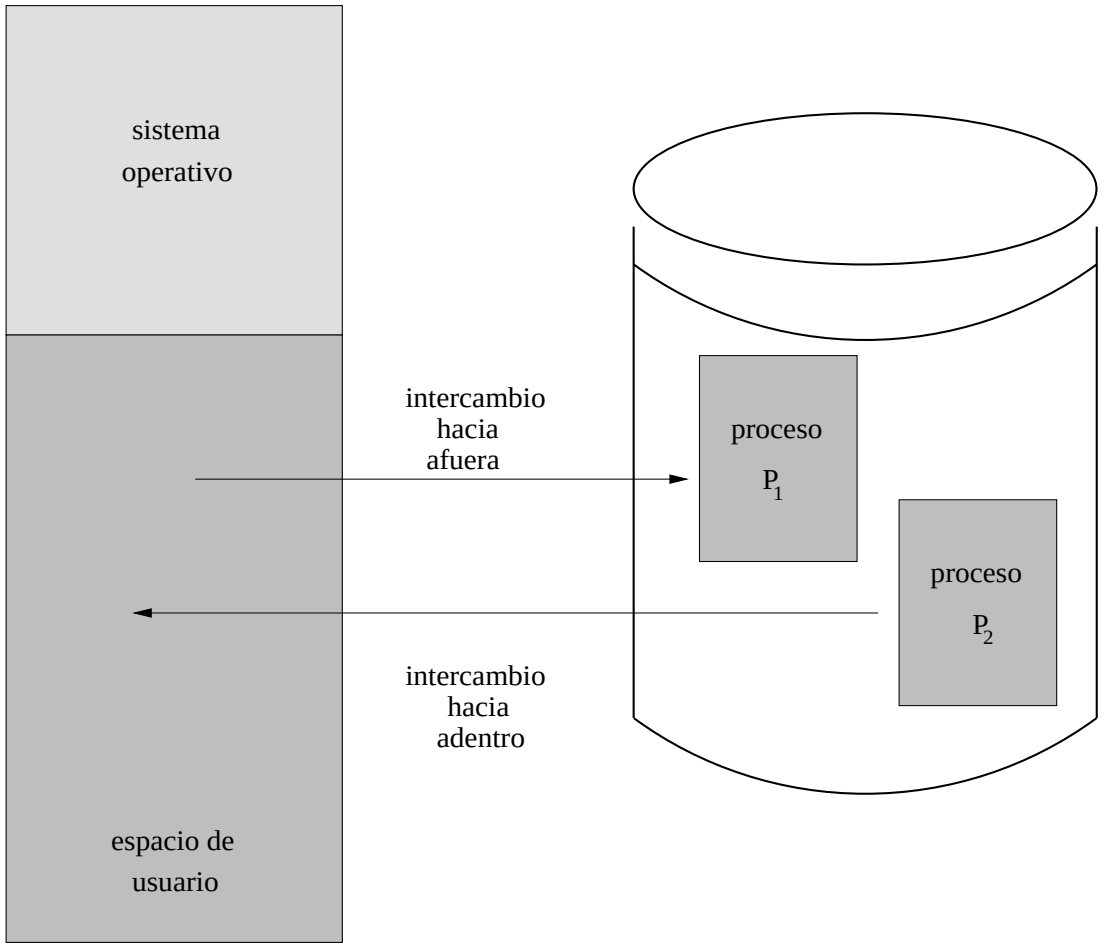
Superposiciones



Intercambio

- Un proceso necesita estar en memoria para ser ejecutado.
- Un proceso puede **intercambiarse** temporalmente de la memoria a un **almacén de respaldo**, y luego llevarse de regreso a la memoria para continuar su ejecución.
- Un proceso que es intercambiado fuera de la memoria será puesto de nuevo en la memoria dependiendo del modo de ensamble o carga que tenga el proceso.
- El almacén de respaldo es comúnmente un disco rápido.

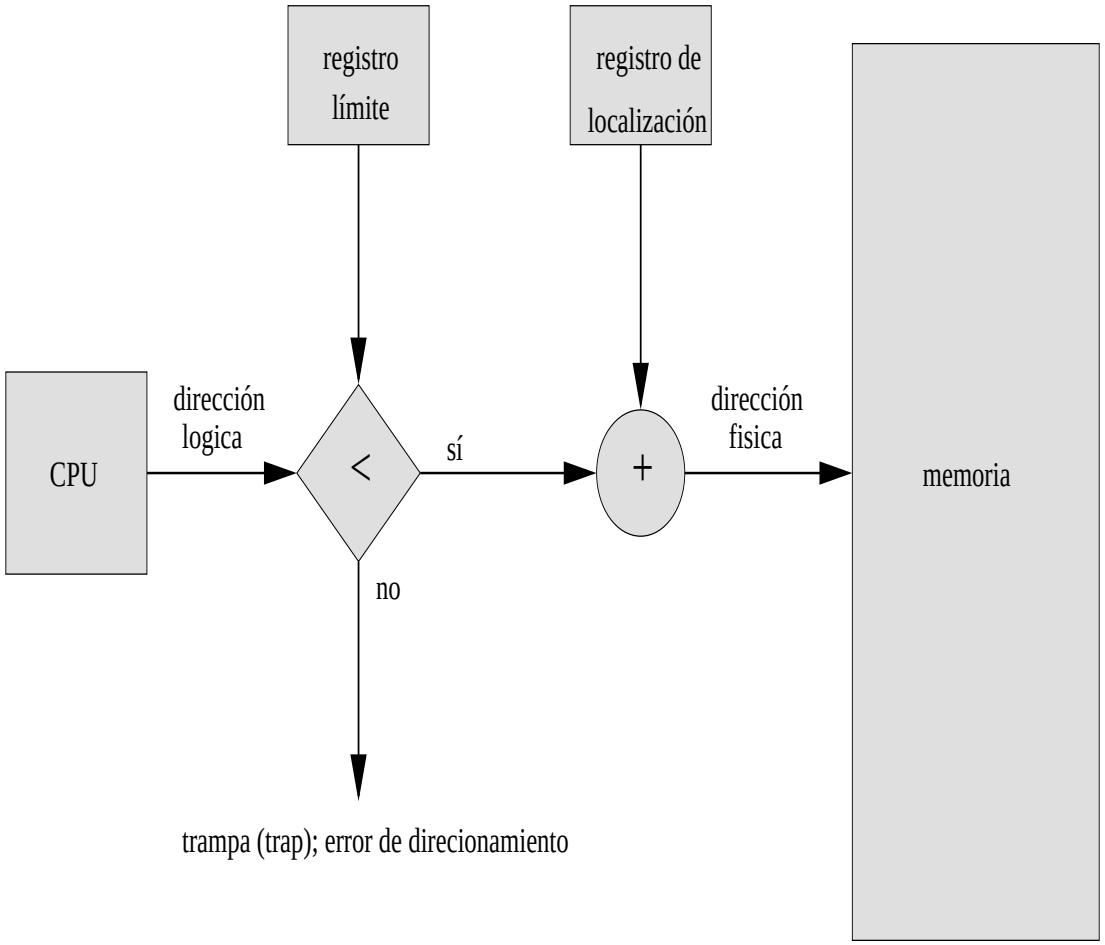
Intercambio



Asignación de memoria continua

- La memoria principal debe acomodar tanto al sistema operativo como a los diversos procesos de usuario.
- La memoria esta divida en dos particiones:
  - Una para el sistema operativo residente. En la memoria baja o alta, esto depende generalmente de donde esta ubicado el vector de interrupciones.
  - La otra para los procesos de usuario.
- Se asigna la restante memoria para los demás procesos, teniendo en cuenta que los procesos se deben proteger los unos a los otros.

Protección de la memoria a través del hardware



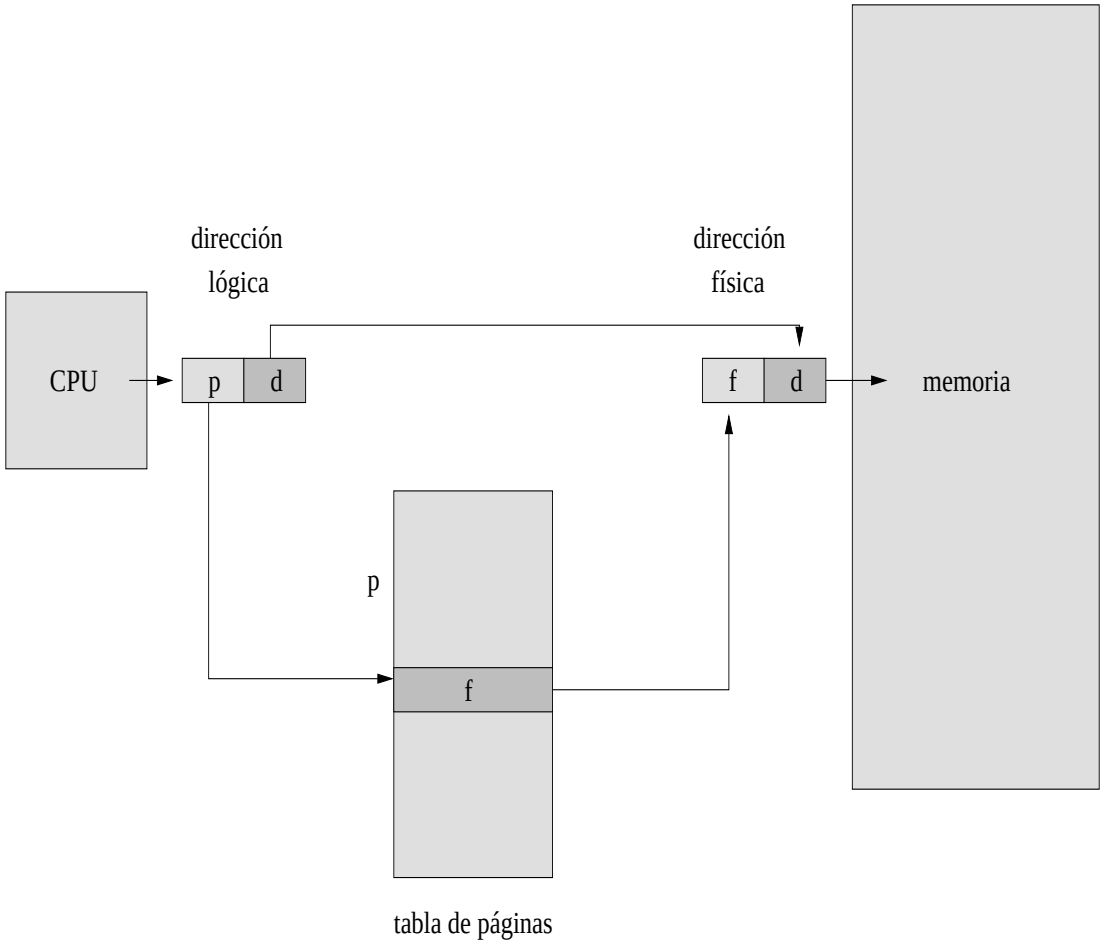
Asignación de memoria continua

- Uno de los métodos más sencillos consiste dividir la memoria en diferentes particiones.
- El sistema operativo mantiene una tabla de la memoria disponible, es decir que no esta ocupada por un proceso.
  - Estática.
    - Una sola cola de entrada de los procesos.
    - Múltiples colas.
  - Dinámica.
    - *Primer ajuste*: asigne el *primer hueco* que sea suficientemente grande.
    - *Mejor ajuste*: asigne el hueco *más pequeño* que sea suficientemente grande.
    - *Peor ajuste*: asigne el hueco *más grande*
- Fragmentación interna.
- Fragmentación externa.
- Compactación.

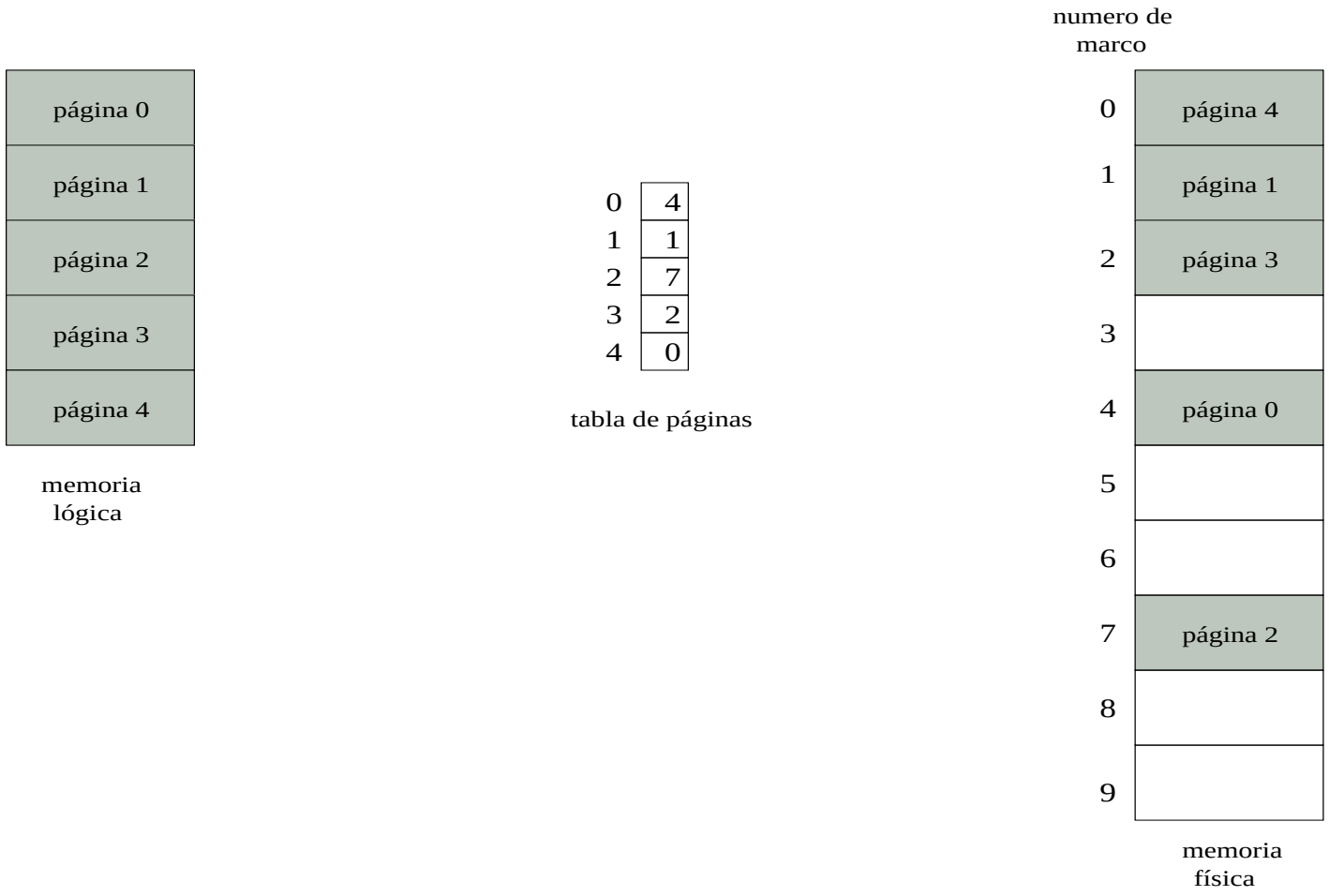
Paginación

- La memoria física se descompone en bloques de tamaño fijo denominados **marcos**.
- La memoria lógica también se descompone en bloques del mismo tamaño denominados **páginas**.
- Cada dirección generada por la CPU se divide en dos partes: un **número de página** (p) y un **desplazamiento de página** (d).
- El número de página se emplea como un índice en una **tabla de páginas**.
- La tabla de páginas contiene la dirección base de cada página en la memoria física. Esta dirección base se combina con el desplazamiento de página para definir la dirección física de la memoria que se envía a la unidad de memoria.
- El tamaño de página es definido por el *hardware*. Es una potencia de 2 y varía entre 512 bytes y 16 megabytes de memoria por página.
- Si el tamaño del espacio de direcciones lógicas es 2^m y el tamaño de una página es 2^n unidades de direccionamiento, entonces los $m - n$ bits de orden elevado de una dirección lógica designan el número de página y los n bits de orden bajo designan el desplazamiento de la página.

Hardware de paginación



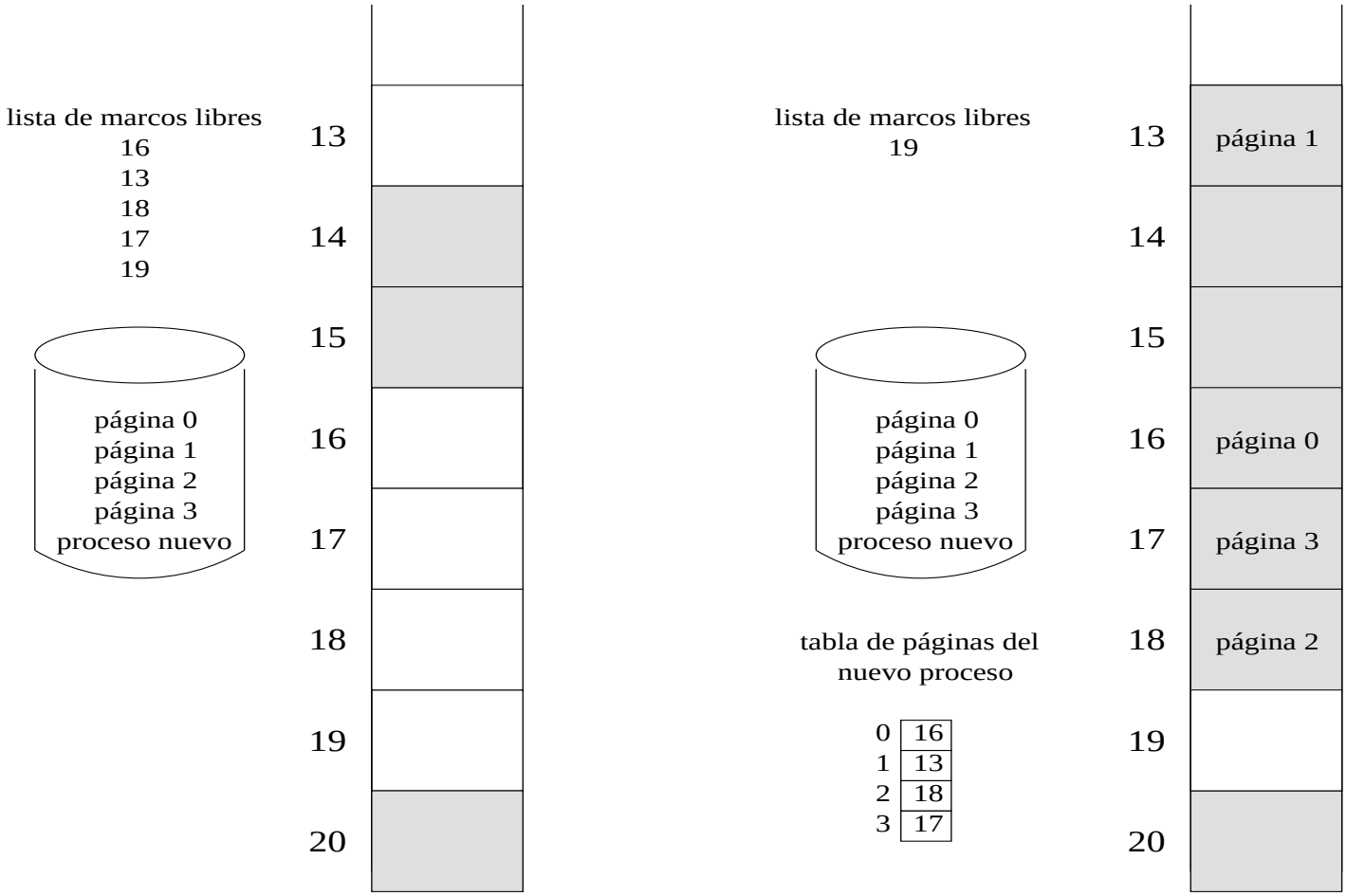
Modelo de paginación



Fragmentación

- Actualmente, las páginas tiene típicamente un tamaño de entre 2 y 8 KB.
- Algunas CPU y kernels soportan incluso tamaños múltiples de páginas.
- Si el espacio lógico de direcciones es de 2^m, el tamaño de una entrada de la tabla de páginas es de m bits.
- Cuando un proceso llega al sistema para ejecutar, si está compuesto de n páginas, debe haber por lo menos n marcos disponibles en la memoria.
- El programa de usuario ve la memoria como un espacio de direcciones contiguo.
- El sistema para gestionar la memoria mantiene una **tabla de marcos**. Esta sirve para indicar que marcos se encuentran libres y cuales están ocupados.

Marcos libres



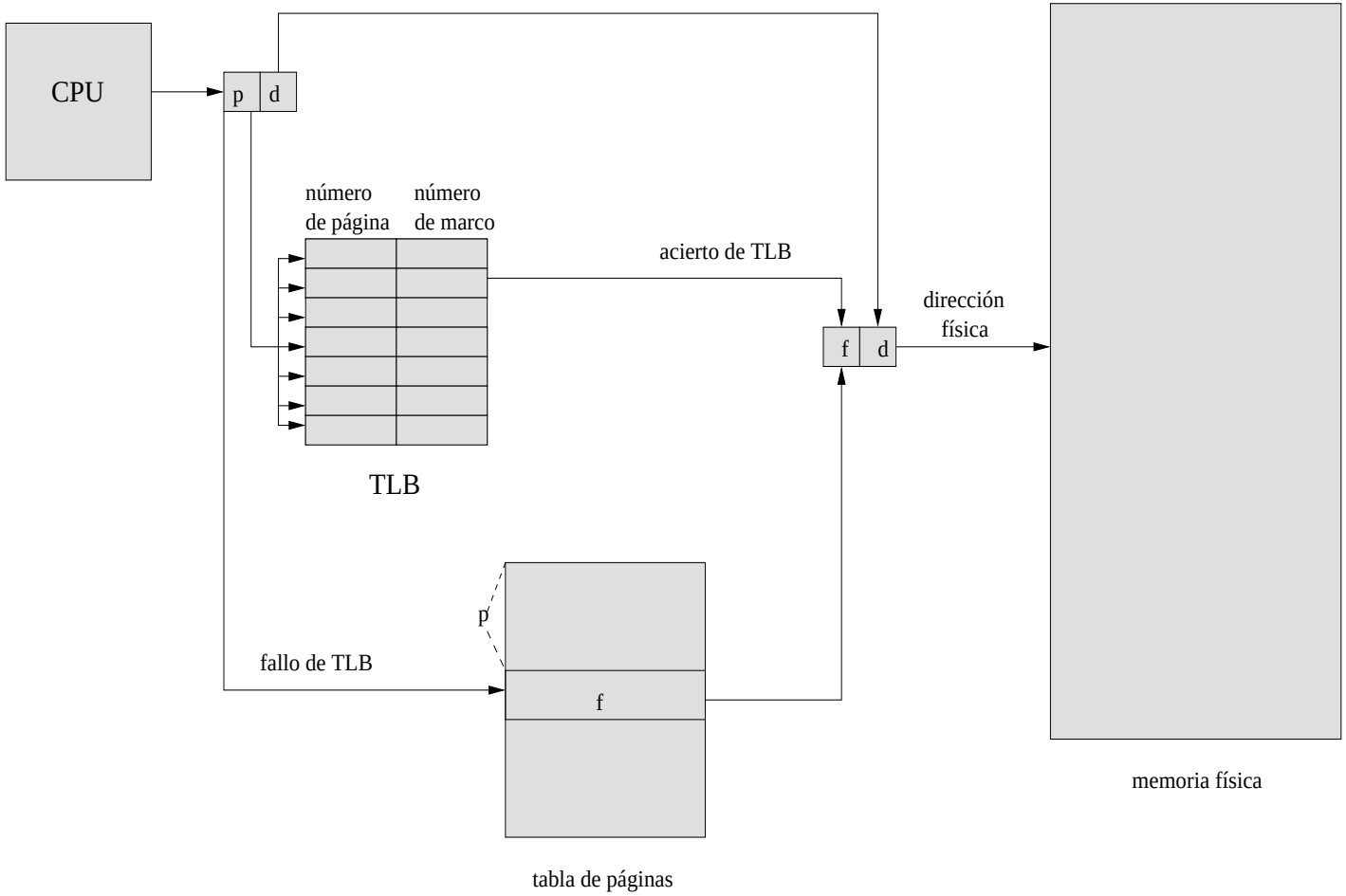
Gestión de la tabla de páginas

- Cada sistema operativo tiene su forma de almacenar las tablas de páginas.
- La mayoría asigna una tabla de páginas por cada proceso. Esta esta almacenada con los demás valores de registros en el bloque de control de procesos.
- El despachador cuando carga un proceso debe cargar su contexto y adicionalmente cargar en el hardware la tabla de páginas del proceso.

Soporte de hardware

- Registros dedicados de alta velocidad. (PDP-11, 2^{16}, Página = 8K).
- La tabla se mantiene en memoria principal y un **registro base de tabla de páginas** *page-table base register, PTBR*.
- **Registros asociativos** o **buffers de traducción de vista lateral (TLB)**.

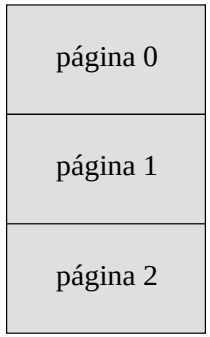
Hardware de paginación con TLB



Protección

- Se realiza mediante *bits de protección* que están asociados a cada marco.
- Se mantienen en la tabla de páginas.
- Un *bit* puede definir que una página sea de lectura y escritura, sólo de lectura.
- El *bit* de verificación sirve para examinar que se está haciendo acceso indebido. Cuando esto ocurre se genera un *fallo de página*.
- Para control del espacio de direcciones del proceso, algunos sistemas proporcionan un *hardware*, bajo la forma de un **registro de longitud de la tabla de páginas (PTLR)**, para indicar el tamaño de dicha tabla de páginas.

Protección



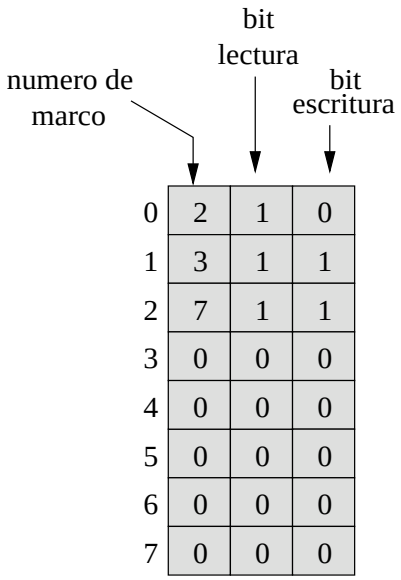
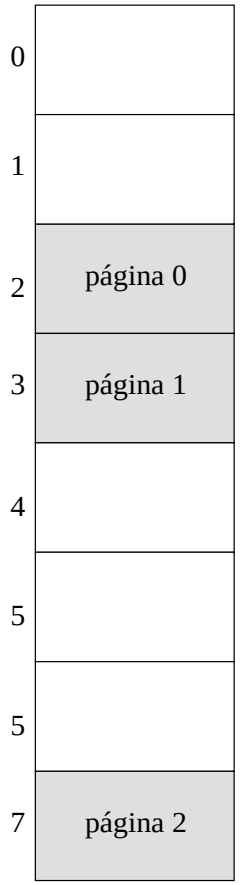


tabla de páginas



memoria física

Paginación con niveles múltiples

- La mayoría de los sistemas de cómputo soportan un espacio grande de direcciones lógicas (2^{32} o 2^{64}).
- La tabla de páginas se vuelve excesivamente grande (Tamaño de tabla de 4K - 2^{12} -).
- Dividir la tabla de páginas en piezas más pequeñas.

Paginación con niveles múltiples

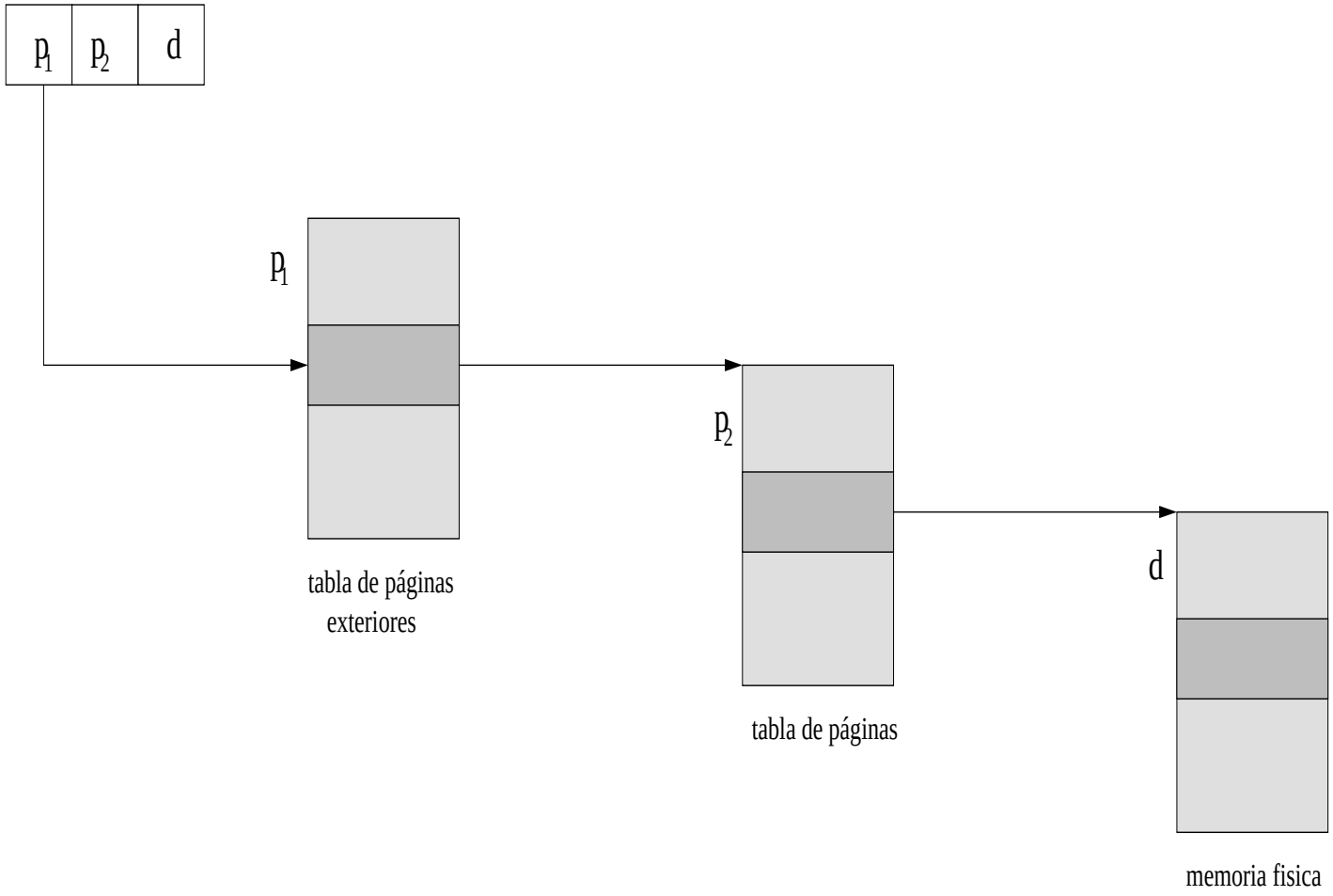
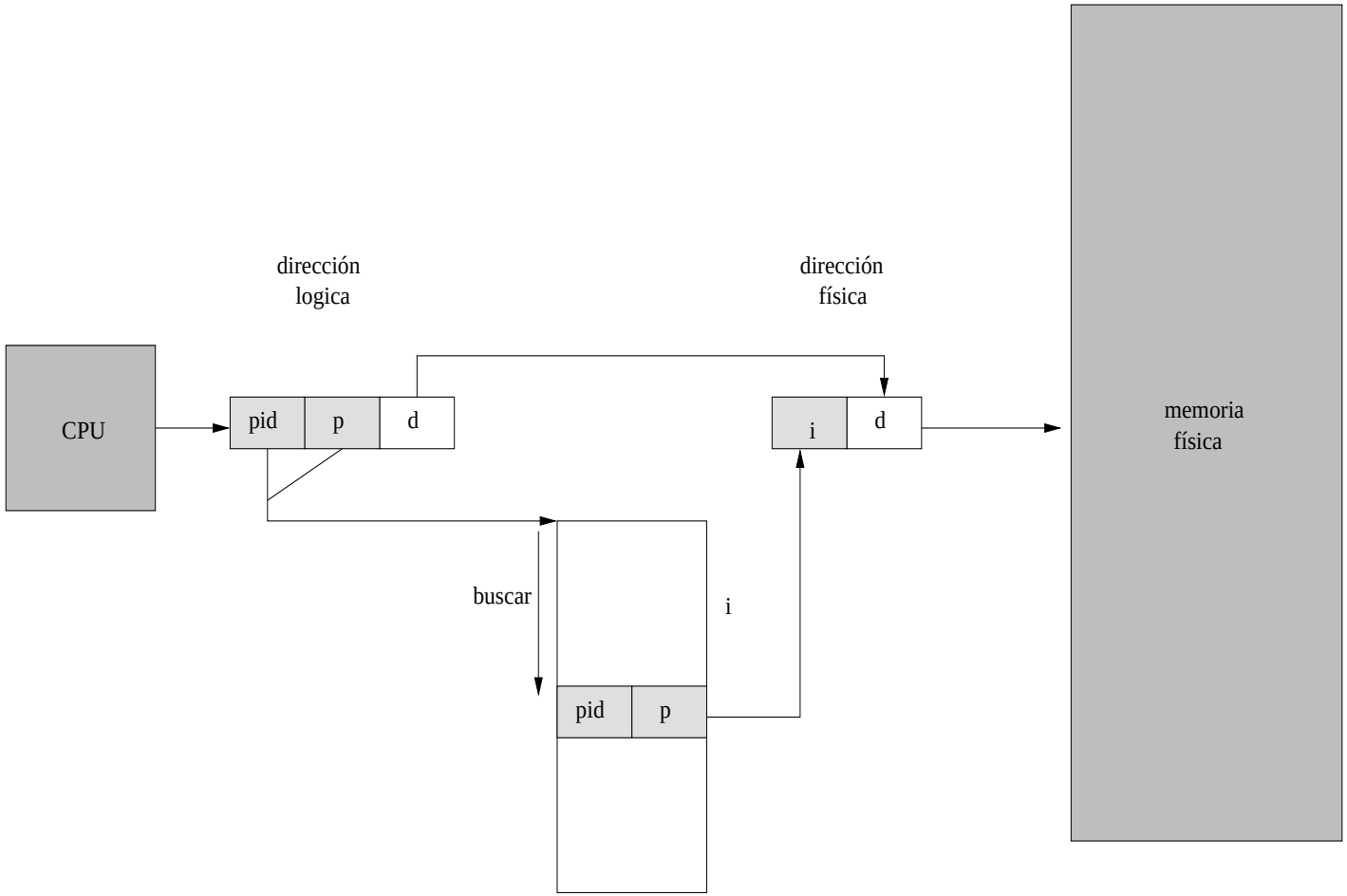


Tabla de páginas invertidas

- Una **tabla de páginas invertidas** tiene una entrada por cada página (marco) real de la memoria.
- Cada entrada consiste de la dirección virtual de la página almacenada en dicha ubicación de memoria real, con información acerca del proceso que posee a dicha página.
- Sólo hay una tabla de páginas en el sistema, y sólo se tiene una entrada por cada página de memoria física.

Tabla de páginas invertidas



Páginas compartidas

- Otra ventaja de la paginación es la posibilidad de *compartir* un código común.
- Permite el uso de código **reentrante**.
- El código reentrante es un código que no puede modificarse a sí mismo. Si el código es reentrante nunca cambia durante la ejecución. Cada proceso tiene su propia copia de registros y almacenamiento de datos para contener los datos para la ejecución del proceso.

Segmentación

- La paginación y la forma como el usuario mira la memoria no son idénticas. El usuario mira la memoria divida en espacios de direcciones lógicas aparte.
- La *segmentación* es un esquema de administración de memoria que soporta esta visión del usuario. Un espacio de direcciones lógicas es un conjunto de segmentos.
- Cada segmento tiene un nombre y una longitud.
- Las direcciones especifican tanto el nombre del segmento como el desplazamiento dentro del mismo.

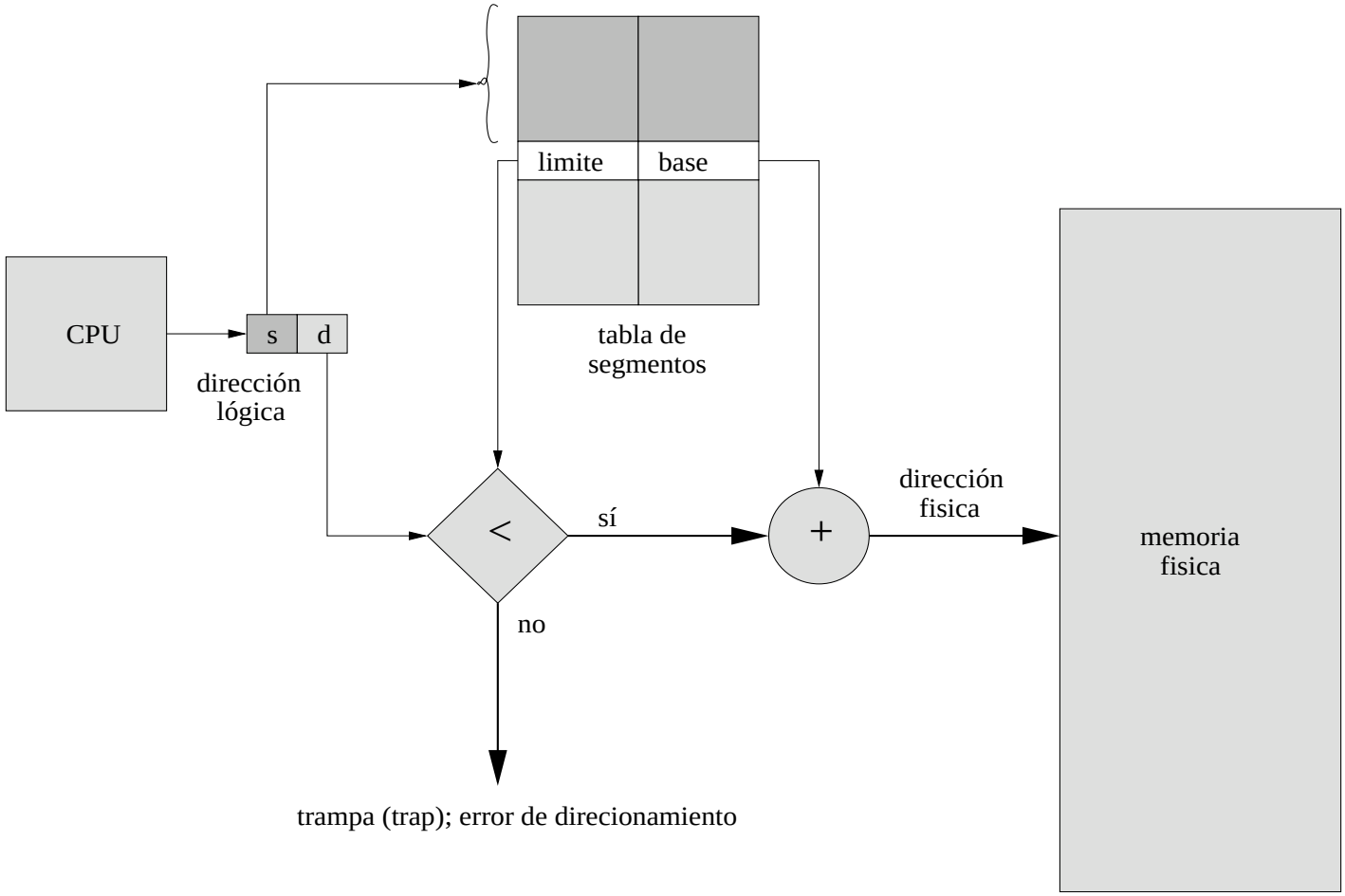
  `<número-de-segmento, desplazamiento>`

- Los segmentos son generados por el compilador.

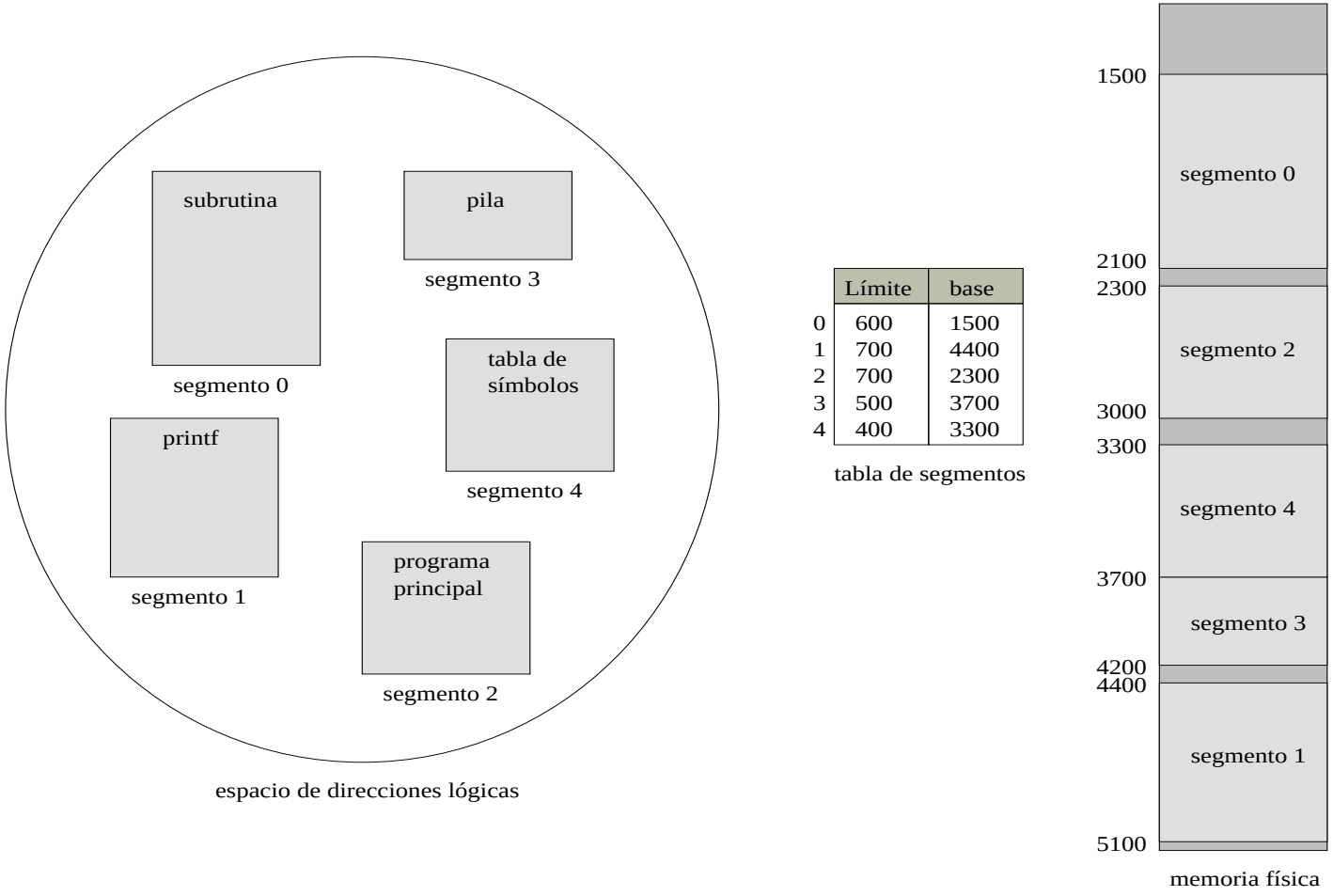
Hardware de segmentación

- Aunque el usuario referencia la memoria utilizando dos partes, la memoria es una secuencia unidimensional de *bytes*.
- Se requiere un esquema de mapeo de direcciones bidimensionales definidas por el usuario en direcciones físicas unidimensionales.
- El mapeo es efectuado por la **tabla de segmentos**.
- Cada entrada tiene una **base** de segmento y un **límite** de segmento.
- La base contiene la dirección física inicial en donde reside el segmento de memoria, en tanto que el límite especifica la longitud del segmento.

Hardware de segmentación



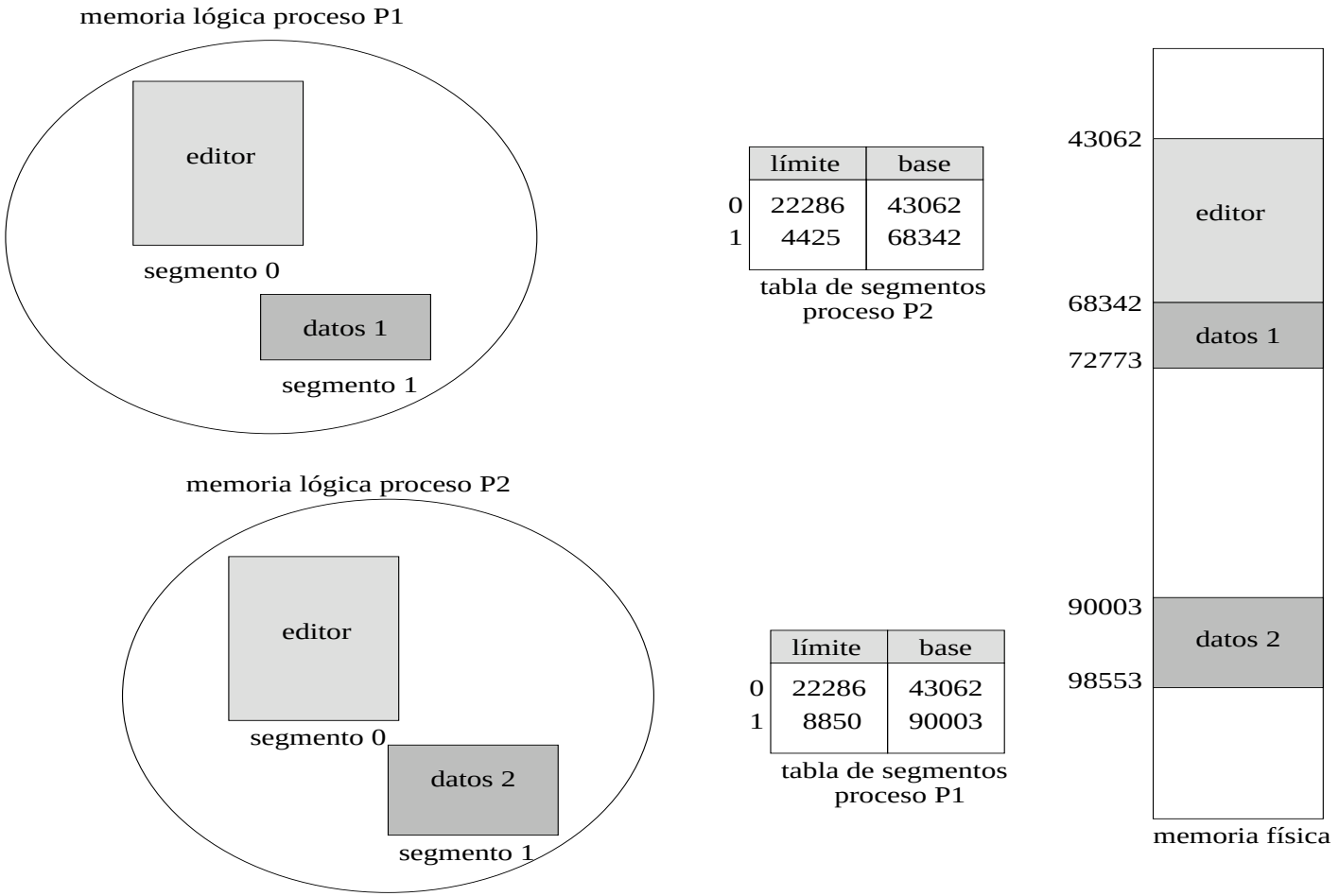
Ejemplo de segmentación



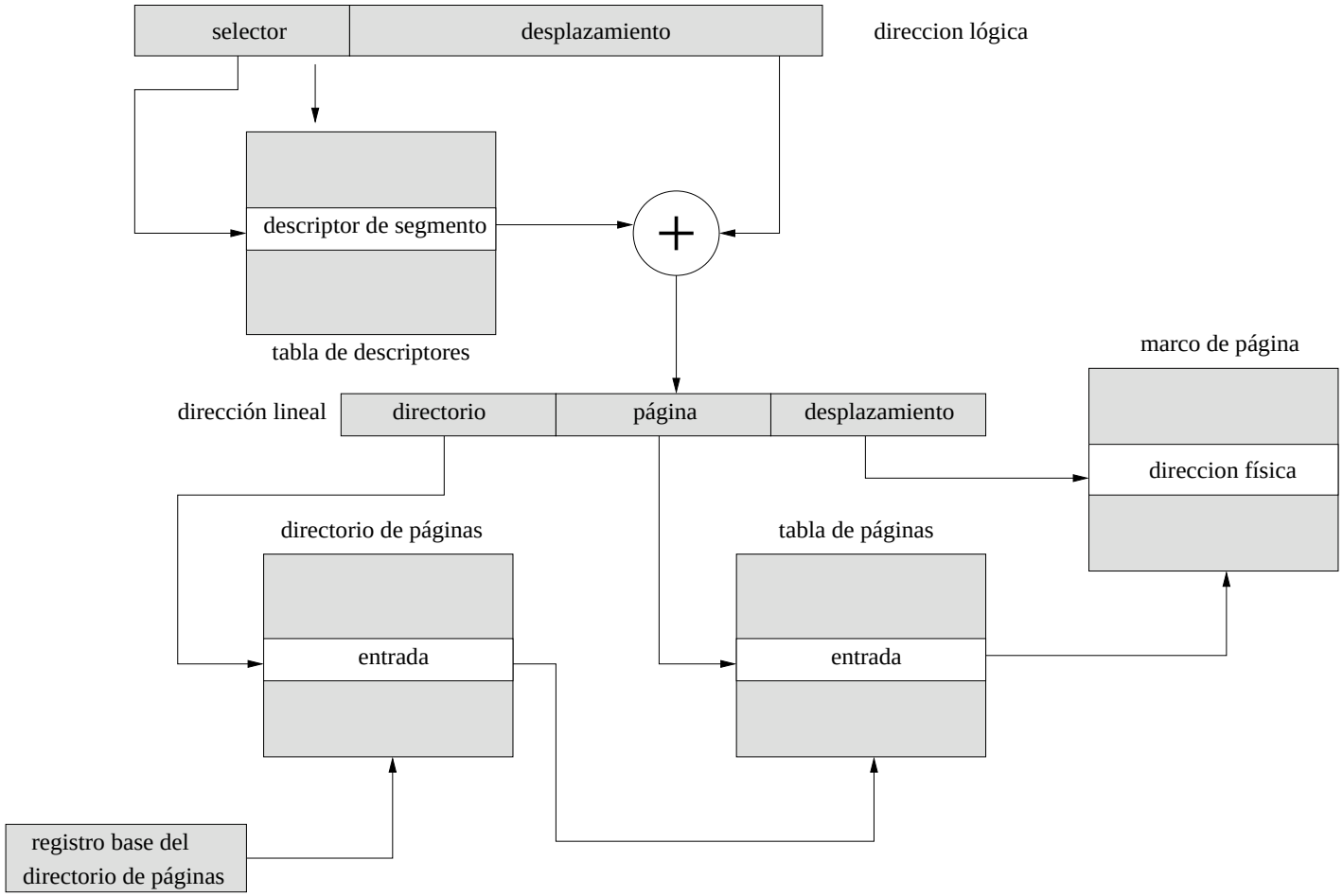
	Límite	base
0	600	1500
1	700	4400
2	700	2300
3	500	3700
4	400	3300

tabla de segmentos

Compartir segmentos



Segmentación con paginación



Memoria virtual

La memoria virtual es una técnica que permite la ejecución de procesos que pueden no estar completamente en memoria.
Los programas pueden ser más grandes que la memoria física.

Antecedentes

- Los programas no utilizan todos su código.
- Los programas a menudo tiene código de excepciones poco comunes.
- Asignación en memoria de más de la utilizada.
- Funciones de programa que rara vez se utilizan.

Memoria virtual

La capacidad para ejecutar un programa que no está en la memoria física tiene mucha ventaja:

- Los programas no estarían limitados por la cantidad de memoria física.
- Puesto que cada programa de usuario ocuparía menos memoria física, se podría ejecutar más programas al mismo tiempo.
- El programador no tiene que preocuparse por la cantidad de memoria física.

La *memoria virtual* es la separación entre la memoria lógica de los usuarios y la memoria física. Esta separación permite ofrecer a los programadores una memoria virtual extremadamente grande aunque solo este disponible una memoria física pequeña.

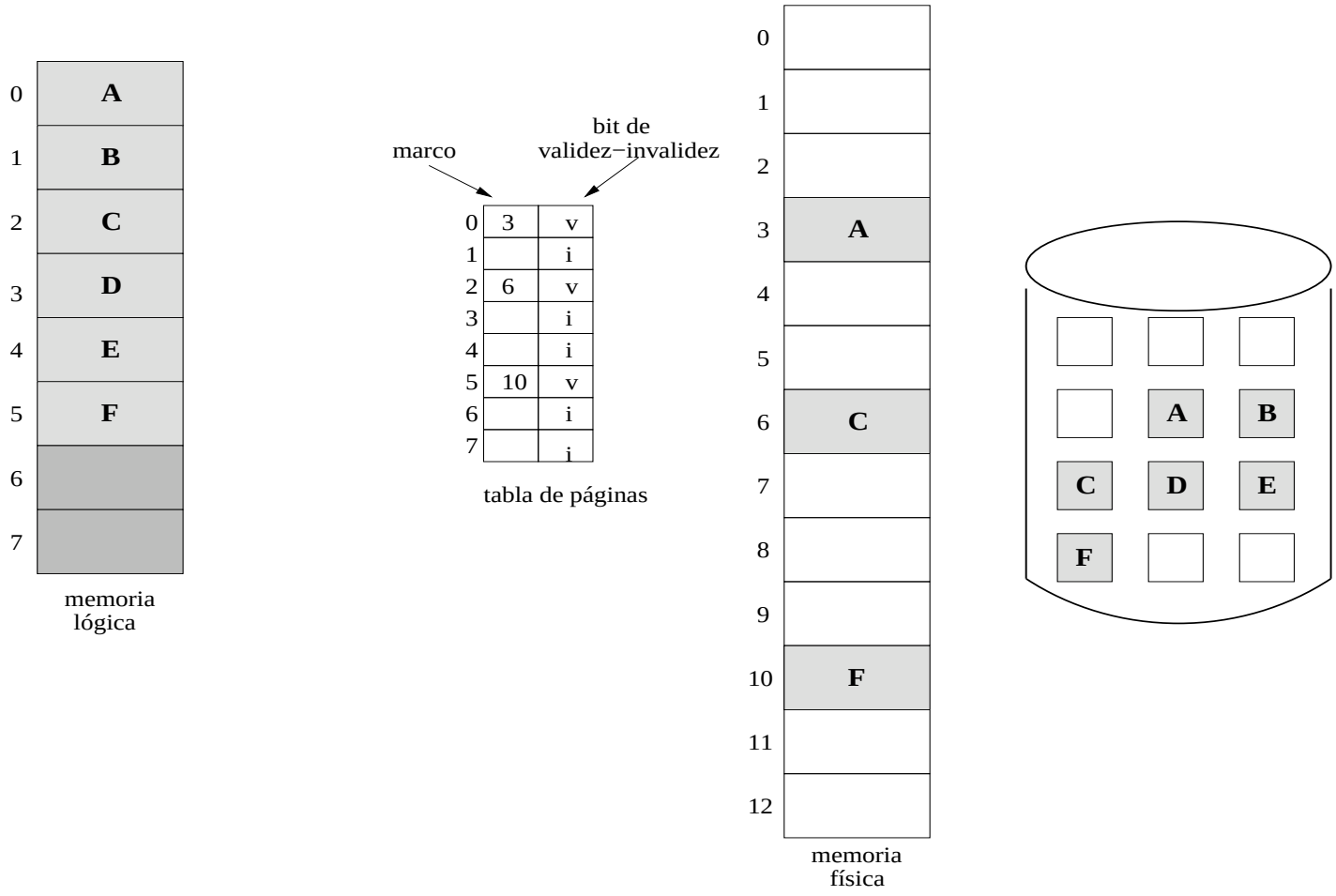
Paginación por demanda

- Es similar al paradigma de intercambio. Pero en el intercambio es todo el proceso.
- Se utiliza un intercambiador perezoso. Nunca intercambia una página a la memoria si no la va a necesitar. Se debe llamar al paginador.
- Cuando un proceso se va a traer a memoria, el paginador adivina cuáles son las páginas que utilizará antes de que el programa se intercambie de vuelta al disco.

Conceptos básicos

- Se requiere un soporte de *hardware* para distinguir entre las páginas que están en memoria y las páginas que están en el disco.
- Se utiliza un esquema de *validez-invalidez*.
  - Cuando el *bit* es "*válido*", indica que la página asociada es legal y se encuentra en memoria.
  - Cuando el *bit* es "*inválido*", este valor indica que la página no es válida (la página no se encuentra en el espacio de direcciones lógicas del proceso), o es válida pero actualmente se encuentra en disco.
- La ejecución continua normalmente si las páginas que accedemos están **residentes en memoria**.
- Si la página no está en las páginas *residentes en memoria* y es del conjunto de direcciones lógicas del proceso, se genera una **trampa de fallo de página**.

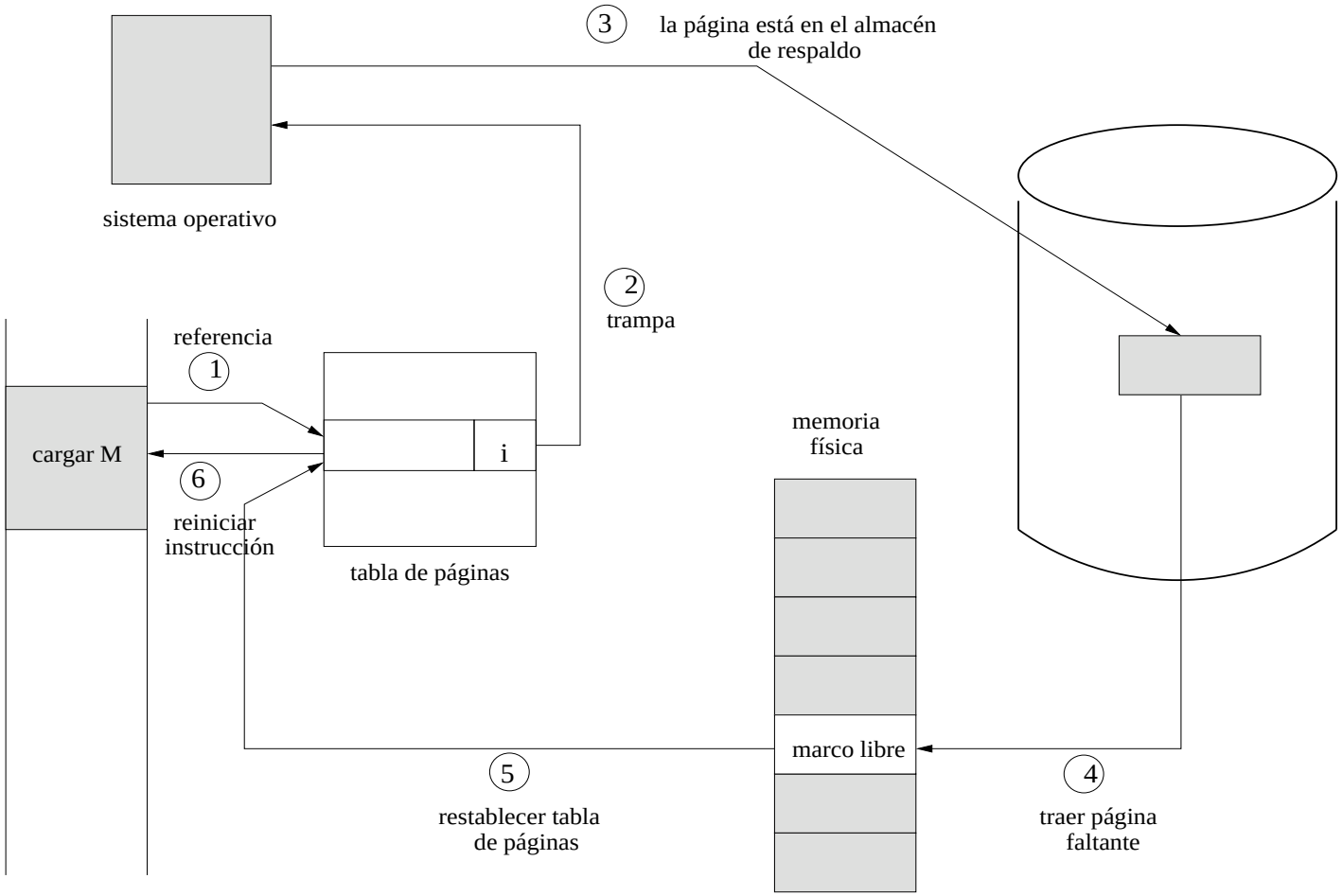
Tabla de páginas en memoria virtual



Pasos del manejo de un fallo de página

1. Consultamos la tabla interna (BCP) de este proceso, para determinar si la referencia fue un acceso válido o no válido a la memoria.
2. Si la referencia no era válida, se termina el proceso, si era válida, pero todavía no se ha traído esa página, procederemos a traerla.
3. Encontramos un marco libre (tomando uno de la lista de marcos libres, por ejemplo).
4. Planificamos una operación de disco para la leer la página deseada y colocarla en el marco recién asignado.
5. Al terminar la lectura del disco, modificamos la tabla interna que se guarda junto con el proceso y la tabla de páginas, de modo que indiquen que la página ya está en memoria.
6. Reiniciamos la instrucción que se interrumpió por la trampa de dirección no válida. Ahora el proceso puede acceder a la página como si siempre hubiera estado en la memoria.

Pasos del manejo de un fallo de página



Desempeño de la paginación por demanda

- ma: Tiempo de acceso a la memoria. (10 y 200 nanosegundos).
- Si no ocurre fallos de página el tiempo de acceso efectivo a la memoria es igual al tiempo de acceso a la memoria.
- Si ocurre un fallo, se debe leer del disco la página correspondiente y luego acceder a la palabra deseada.
- p: Es la probabilidad de un fallo de página ($0 \leq p \leq 1$).
- tfp: Tiempo de fallo de página.
- El tiempo de acceso efectivo.

$$(1 - p) \times ma + p \times tfp$$

Para calcular el tiempo de acceso efectivo, se debe conocer cuanto tiempo se necesita para servicio a un fallo de página.

Manejo de fallas de páginas

1. Trampa al sistema operativo.
2. Guardar los registros del usuario y el estado del proceso.
3. Determinar que la interrupción fue un fallo de página.
4. Verificar que la referencia a la página fue válida y determinar la posición de la página en el disco.
5. Leer del disco un marco libre:
   - *a*) Esperar en una cola para este dispositivo hasta que se atienda la solicitud de lectura.
   - *b*) Esperar durante el tiempo de búsqueda y/o latencia del dispositivo.
   - *c*) Iniciar la transferencia de la página a un marco libre.
6. Durante la espera, asigna la CPU a algún otro usuario (planificación de la CPU; opcional).

Manejo de fallas de páginas

7. Interrupción del disco (E/S terminada).
8. Guardar los registros y el estado de proceso del otro usuario (si se ejecutó el paso 6).
9. Determinar que la interrupción provino del disco.
10. Corregir la tabla de páginas y las demás tablas de modo que indiquen que la página deseada ya está en la memoria.
11. Esperar que la CPU se asigne otra vez a este proceso.
12. Restaurar los registros de usuario, el estado del proceso y la nueva tabla de páginas, y reanudar la instrucción interrumpida.

Desempeño del fallo de página

Los tres componentes principales del tiempo de servicio de fallo de página son:

1. Atender la interrupción de fallo de página. (1 a 100 μ-segundos)
2. Leer la página. (Latencia promedio: 8 milisegunos, una busqueda de 15 milisegundos, 1 milisegundo: 25 milisegundos).
3. Reiniciar el proceso. (1 a 100 μ-segundos)

$$\begin{aligned} \text{tiempo de acceso efectivo} &= (1-p) \times (100) + p \times (25milisegundos) \\ &= (1-p) \times 100 + p \times 25,000,000 \\ &= 100 + 24,999,900 \times p \end{aligned}$$

$$110 > 100 + 25,000,000 \times p$$

$$10 > 25,000,000 \times p$$

$$p < 0{,}0000004$$

Reemplazo de páginas

- Si se aumenta el grado de multiprogramación, se asigna más memoria, se aumenta los fallos de páginas.
- Si no hay páginas disponibles, el sistema tiene varias opciones:
  1. Terminar el proceso que no obtiene memoria (no es muy factible).
  2. Intercambiar a disco un proceso para liberar todos sus marcos y reducir el nivel de multiprogramación. (mejor alternativa).
- Reemplazo de páginas:

Reemplazo de páginas

1. Encontrar la página deseada en el disco.
2. Hallar un marco libre:
   a) Si hay un marco libre, usarlo.
   b) Si no, usar un algoritmo de reemplazo de páginas para escoger un marco victima.
   c) Escribir la página victima en el disco; modificar de manera acorde las tablas de páginas y de marcos.
3. Leer una página deseada y colocarla en el marco recién liberado; modificar las tablas de páginas y de marcos.
4. Reiniciar el proceso de usuario.

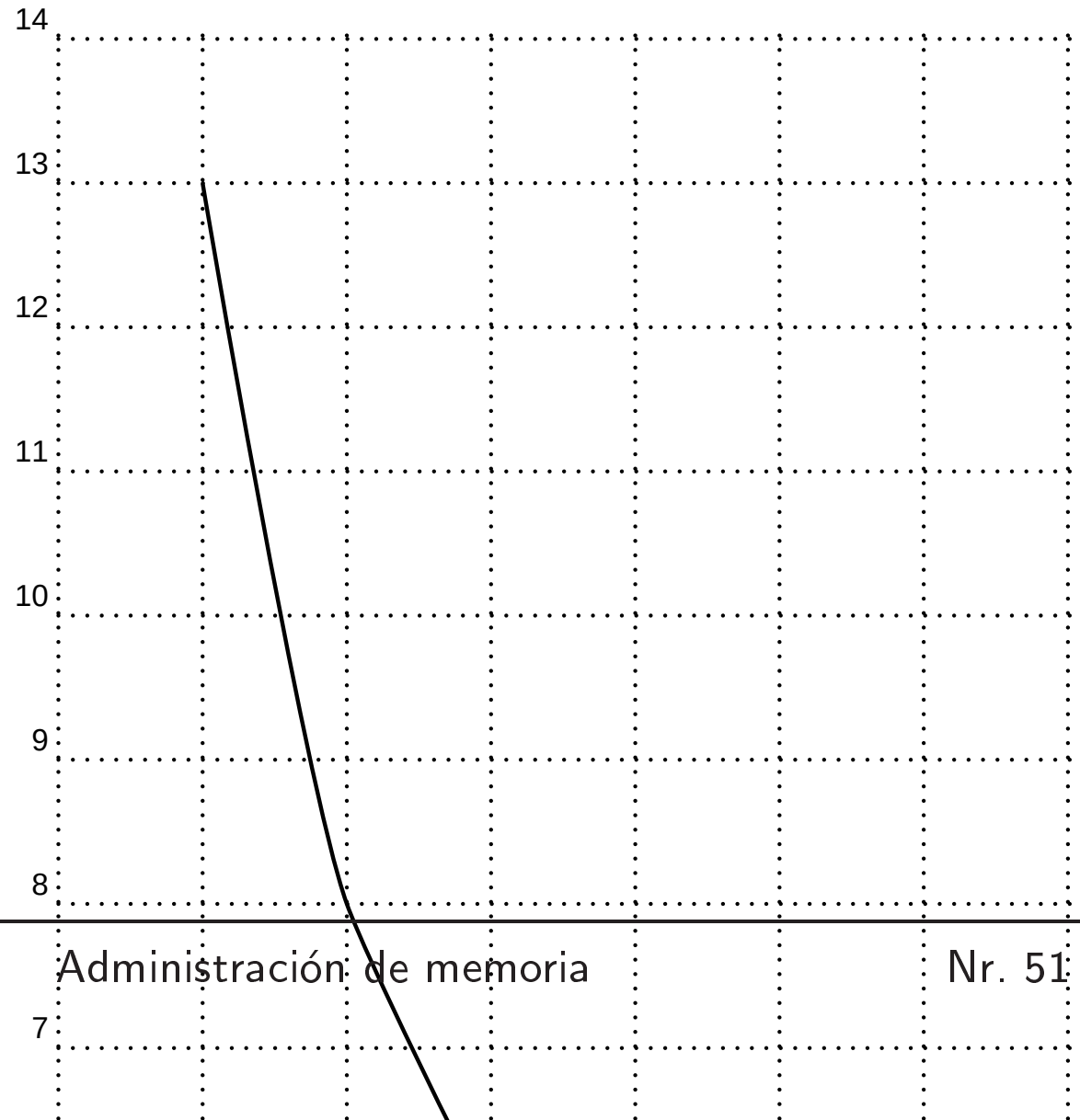
Algoritmos de reemplazo de páginas

- Algoritmo FIFO.
- Algoritmo óptimo.
- Algoritmo LRU.
- Algoritmo de aproximación a LRU
  - Algoritmo con bits de referencia adicionales
  - Algoritmo de segunda oportunidad
  - Algoritmo de segunda oportunidad mejorado
- Algoritmo de conteo
  - Algoritmo LFU (least frequently used)
  - Algoritmo MFU (most frequently used)
- Algoritmo de colocación de páginas en buffers

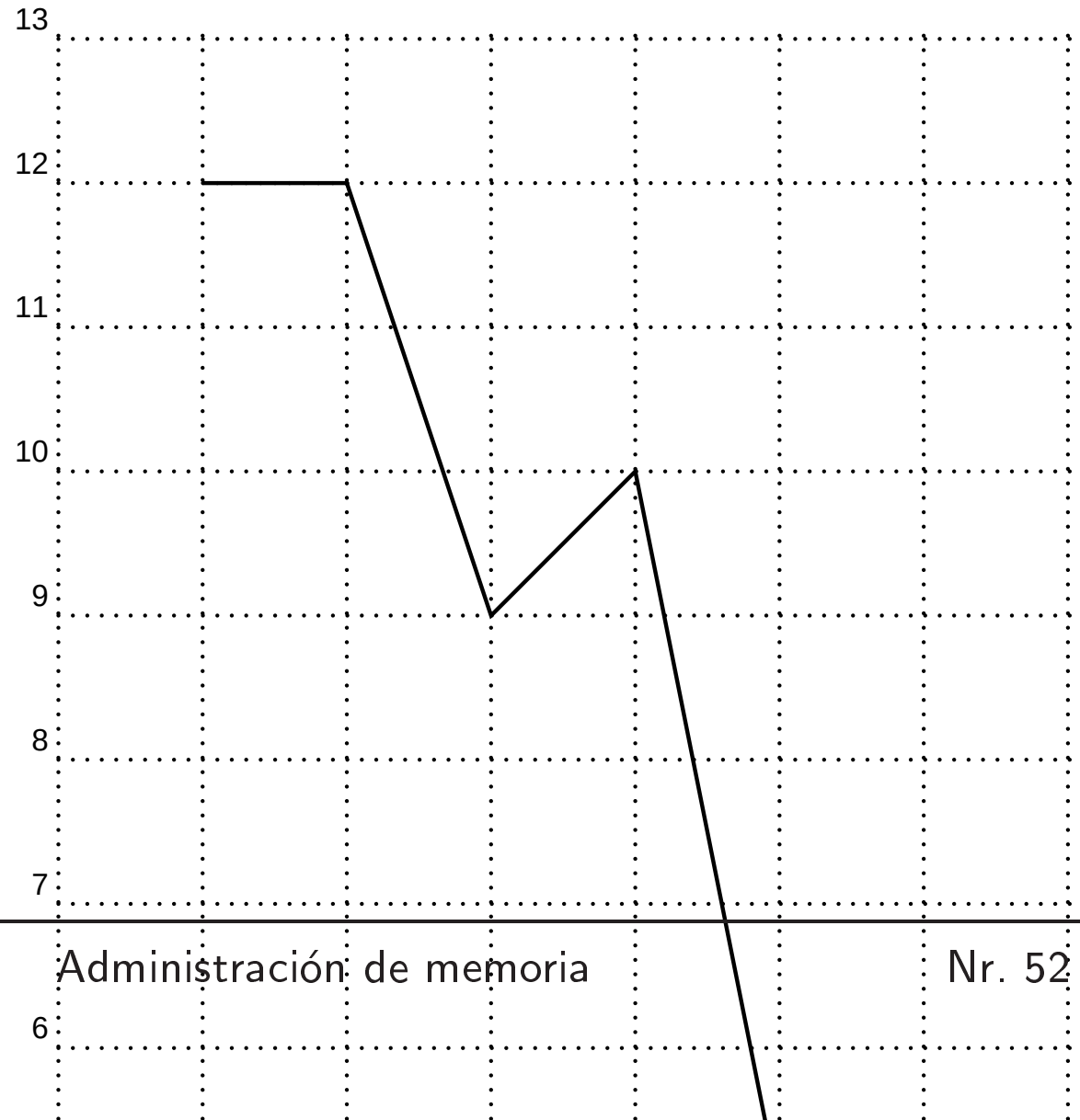
Reemplazo de páginas

- Hay que resolver dos problemas importantes para implementar la paginación por demanada:
  - Desarrollar un algoritmo de asignación de marcos.
  - Un algoritmo de reemplazo de página.

Curva de fallos de página *vs.* número de marcos



Curva de fallos de página para reemplazo FIFO



Asignación de marcos

¿Cómo se reparte la cantidad fija de memoria libre entre los distintos procesos?

- Número mínimo de marcos. Definido por la arquitectura por el conjunto de instrucciones.
- Asignación.
  - Asignación equitativa. Dividir m marcos entre n procesos. (m/n)
  - Asignación proporcional. Asignamos memoria disponible a cada proceso p_i y definimos:

    $$S = \sum s_i$$

    Entonces, si el total de marcos disponibles es m, asignamos a_i marcos al proceso p_i donde a_i es aproximadamente:

    $$a_i = s_i/S \times m$$

  - Asignación global o local.

Hiperpaginación (Thrashing)

- Causas de la hiperpaginación.
- Modelo de localidad.
  1. Una localidad es un conjunto de páginas que se están usando activamente juntas.
  2. Un programa generalmente se compone de varias localidades distintas, que podrían translaparse.

Modelo de conjunto de trabajo

- Emplea un parámetro D, para definir la ventana del conjunto de trabajo.
- Examina las D referencias a páginas más recientes.
- Si una página está en uso activo, estará en el conjunto de trabajo; si ya no se está usando, saldrá del conjunto de trabajo D unidades de tiempo después de su última referencia.
- La propiedad más importante del conjunto de trabajo es su tamaño.
- $D = \sum TCT_i$. Donde D es la demana de marcos.
- Así cada proceso i necesita TCT_i marcos.
- Si la demanda total es mayor que el número total de marcos disponibles habrá hiperpaginación.

Frecuencia de fallos de página

- Adopta un enfoque más directo.
- La hiperpaginación tiene una frecuencia de fallos de página elevada; por ello nos interesa controlar la frecuencia de fallos de página.
- Si la frecuencia es demasiado alta, sabemos que el proceso necesita más marcos; si es demasiado baja, podría ser que el proceso tiene demasiados marcos.
- Se puede establecer límites superior e inferior para la frecuencia de fallos de página deseada.