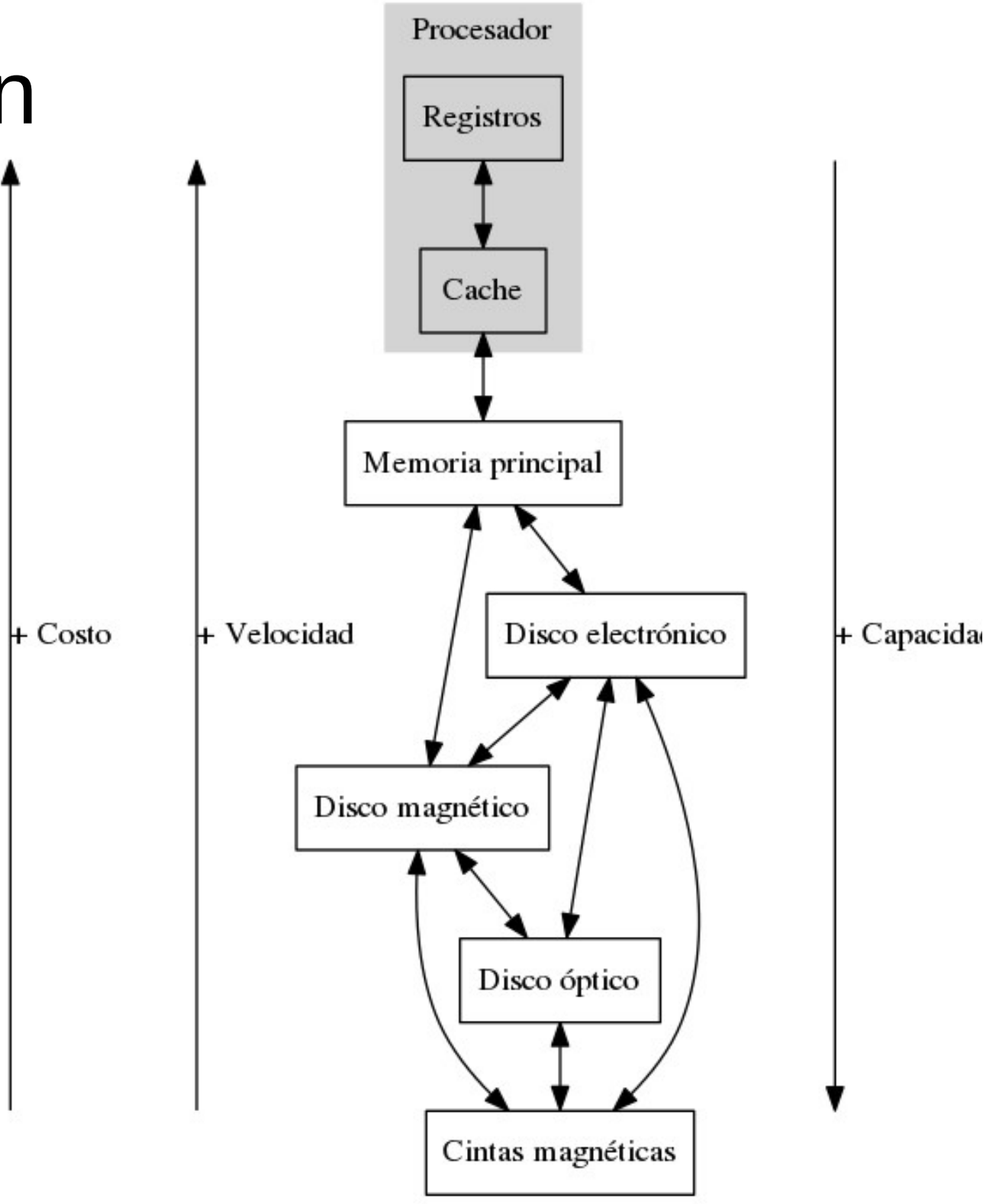
Administración de memoria

Jerarquías de memoria

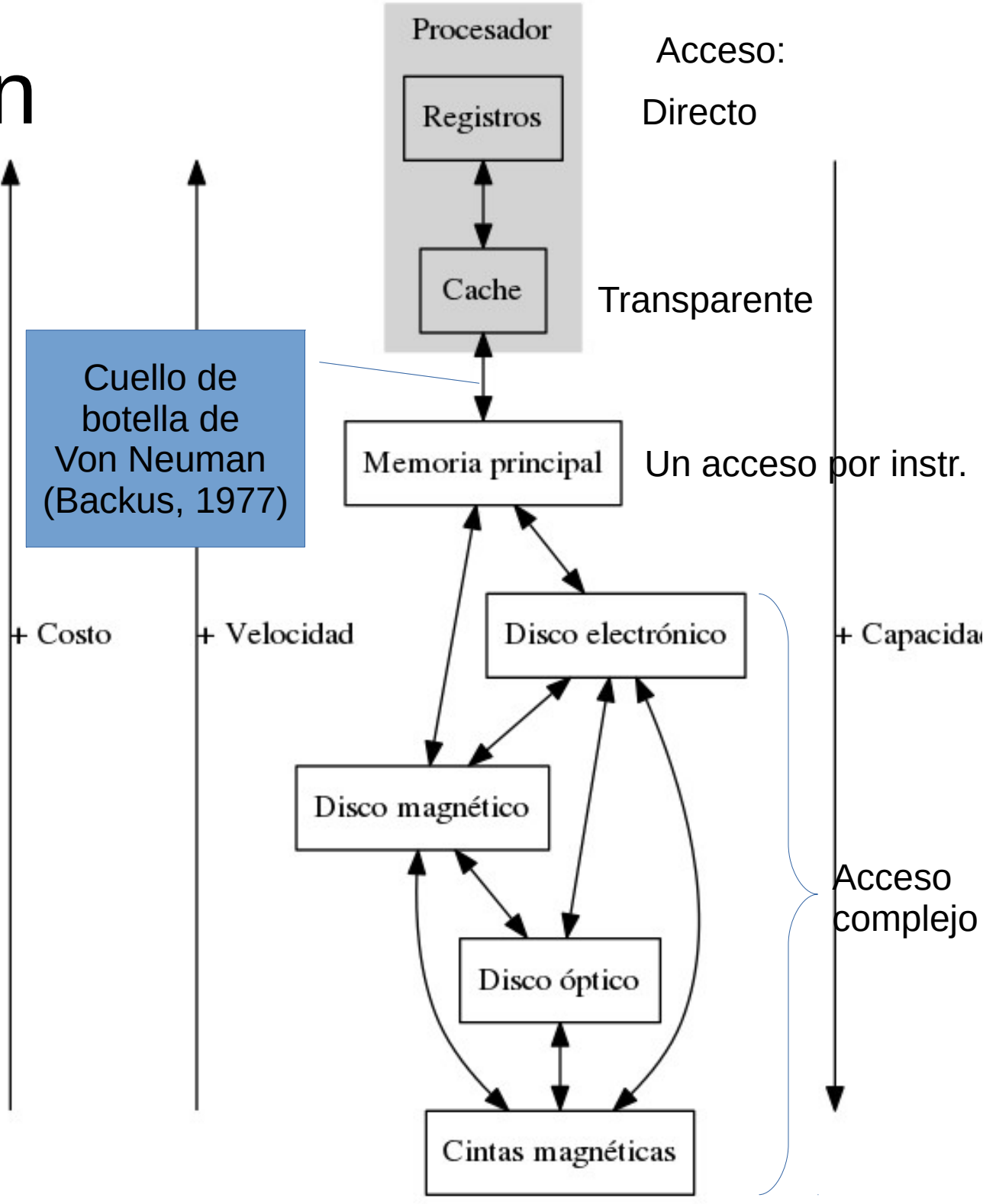
Ref: Fundamentos de Sistemas Operativos



Administración de memoria

Jerarquías de memoria:

¿cómo se accede cada elemento?

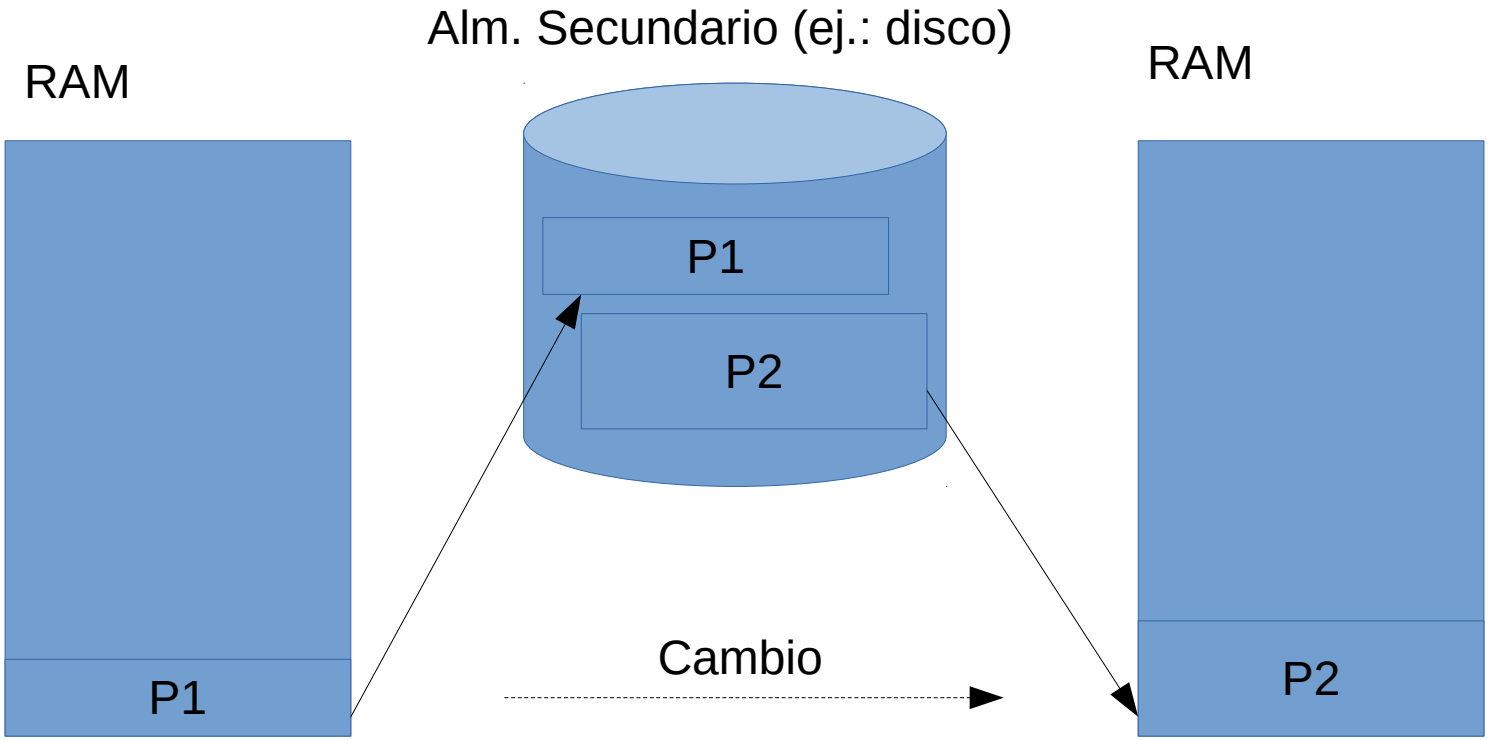


Memoria principal (RAM)

- Arreglo de bytes.
- Originariamente los procesos la usaban directamente.
- ¿cómo soportar la multiarea?

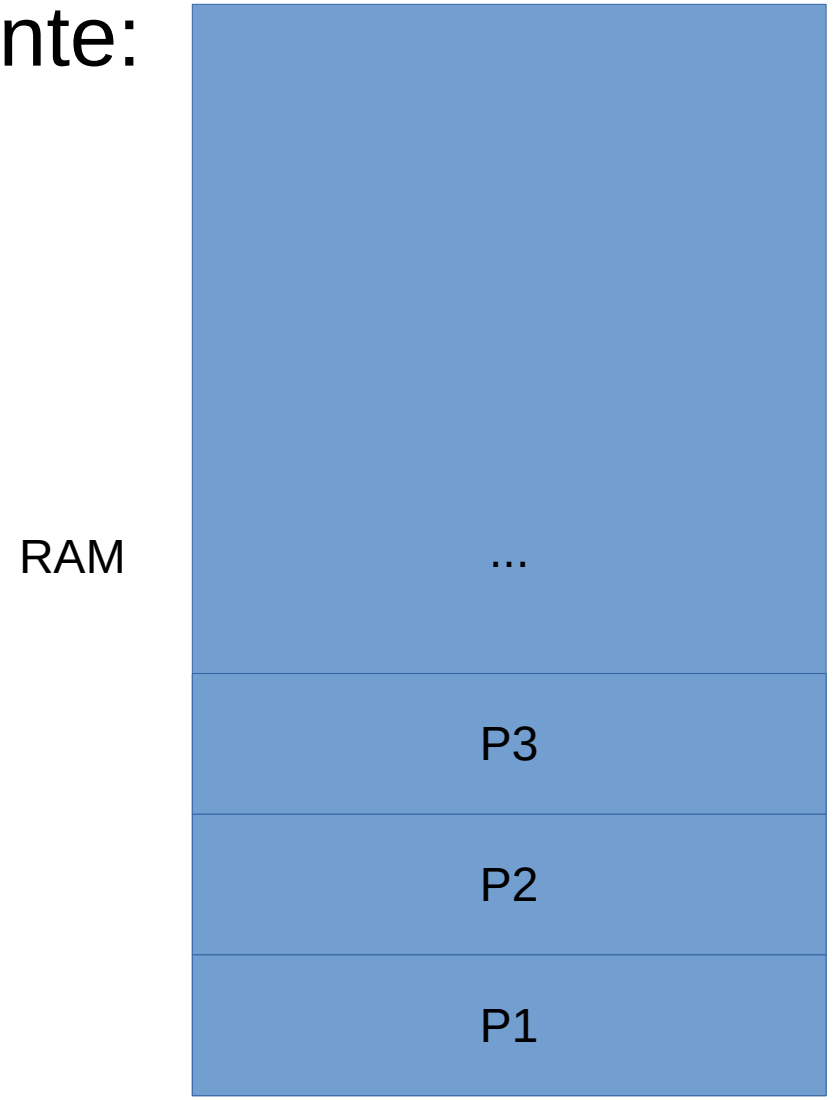
Memoria principal: multitarea, opción 1

- Cuando se necesita darle el procesador a otro proceso, se guarda el proceso actual en disco y se carga el nuevo proceso:



Memoria principal: multitarea, opción 2

- Hacer que cada proceso use una región de memoria diferente:

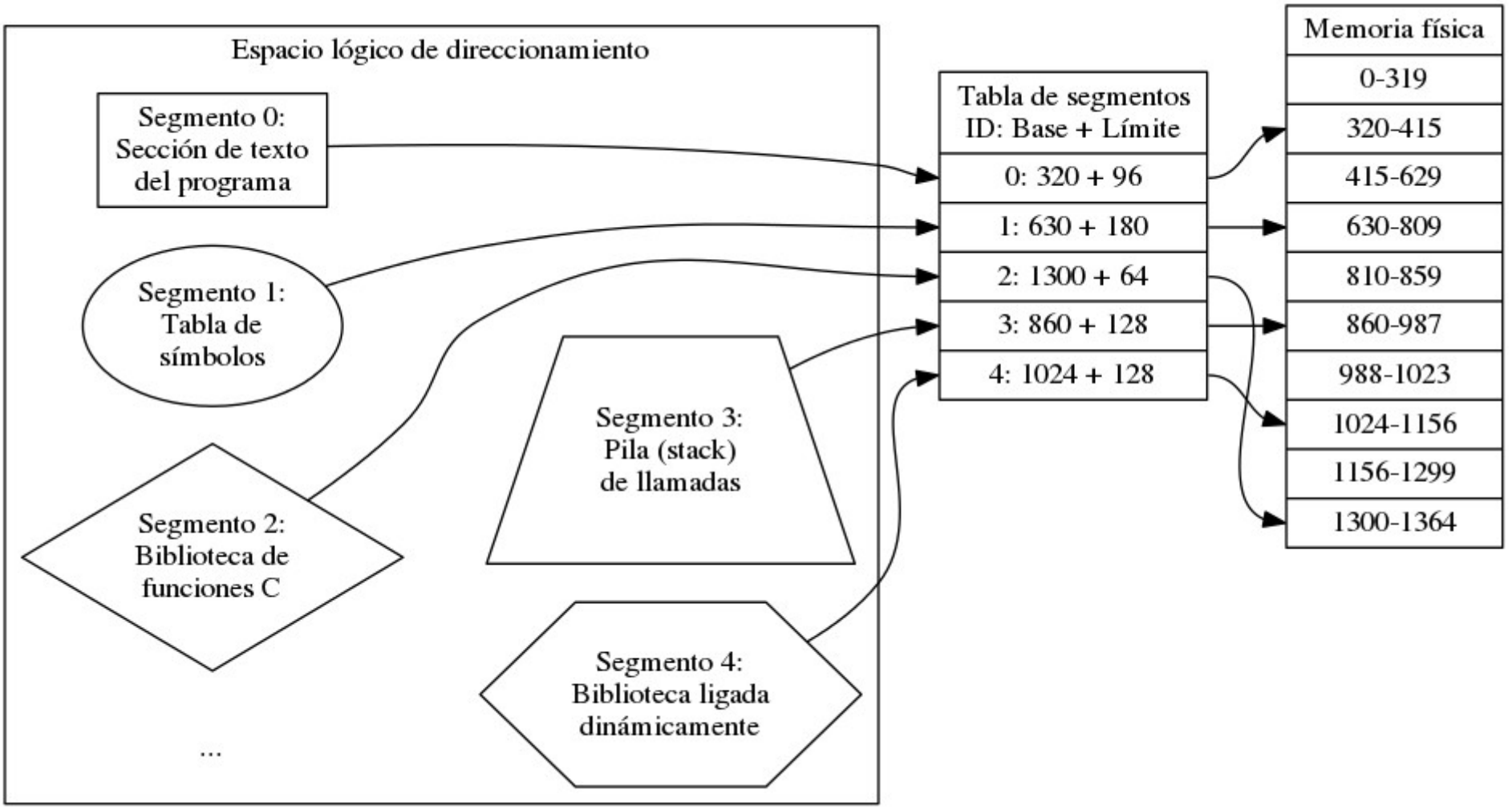


Memoria principal: multitarea Segmentación

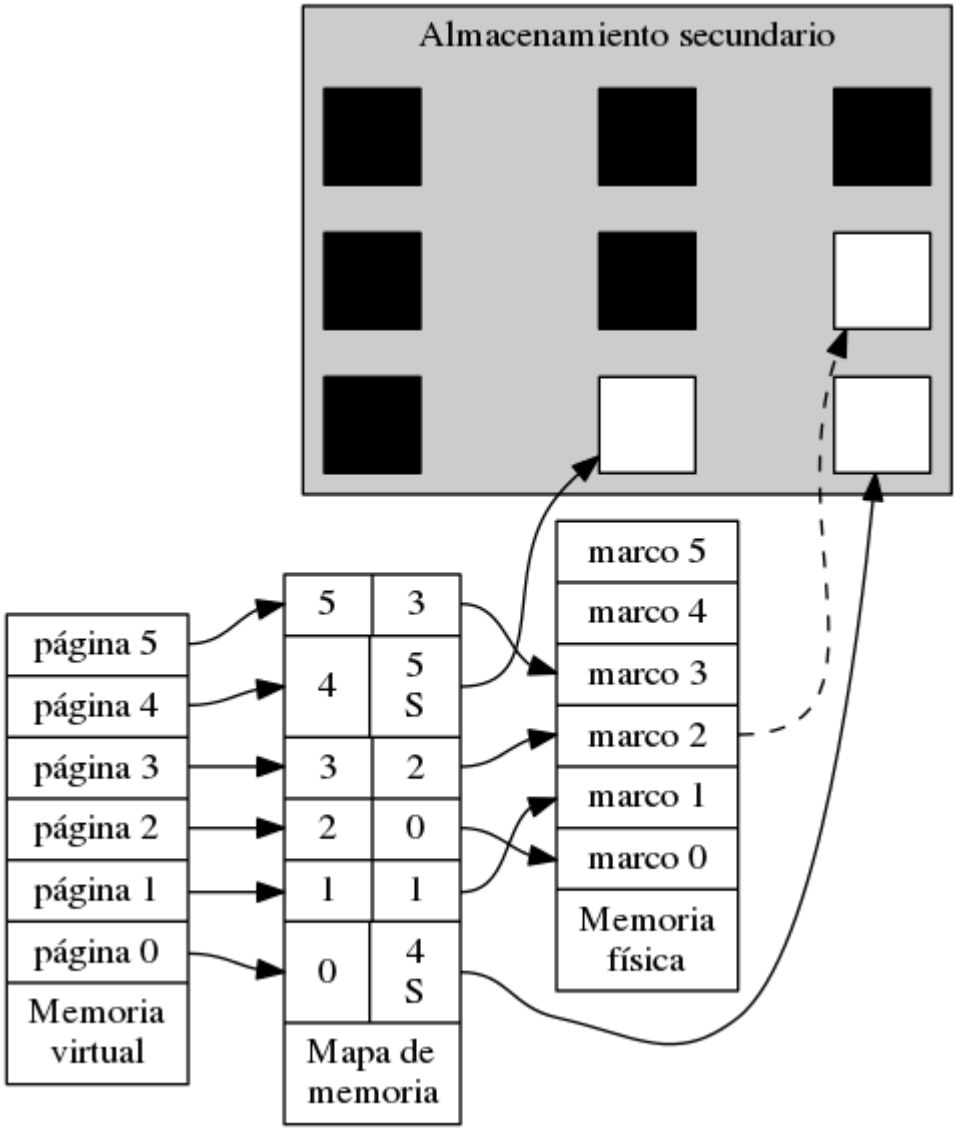
- Agregamos hardware especializado: MMU (unidad de manejo de memoria).
- Los procesos se dividen en distintos segmentos, aparece el concepto de direcciones virtuales.
- Cada segmento comienza en una determinada posición de memoria, tiene un determinado largo y permisos.
- Los segmentos pueden ser reubicados fácilmente en la memoria (sólo hay que cambiar el comienzo del segmento).
- Si un programa no necesita más un segmento lo puede desechar.

Memoria principal: multitarea Segmentación

- Agregamos hardware especializado: MMU (unidad de manejo de memoria)



Un paso más allá: paginación



Un paso más allá: paginación

- Ahora las direcciones virtuales se agrupan de a páginas.
- Cada página lógica (tamaño “estándar” 4Kb) puede corresponderse a una página (o marco) físico en memoria.
- Un proceso ve un rango alto de direcciones virtuales que puede estar mapeado a un cierto número de páginas físicas (nunca mayor al nro. de páginas físicas de la memoria principal)

¿cómo se realiza el mapeo?

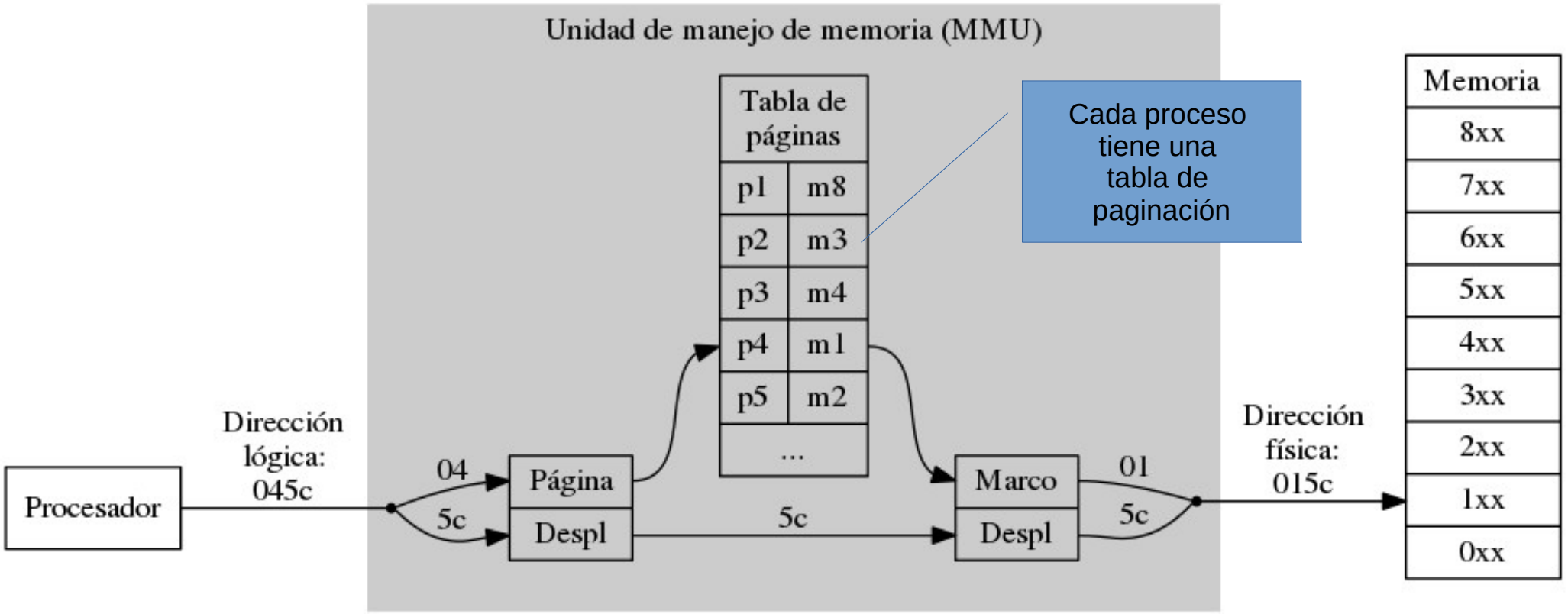
- Las direcciones virtuales se dividen en:

  (nro_de_página_virtual, desplazamiento)
- Se usa una tabla de paginación para mantener la relación:

  (nro_de_página_virtual, nro_de_página_física)

¿cómo se realiza el mapeo?

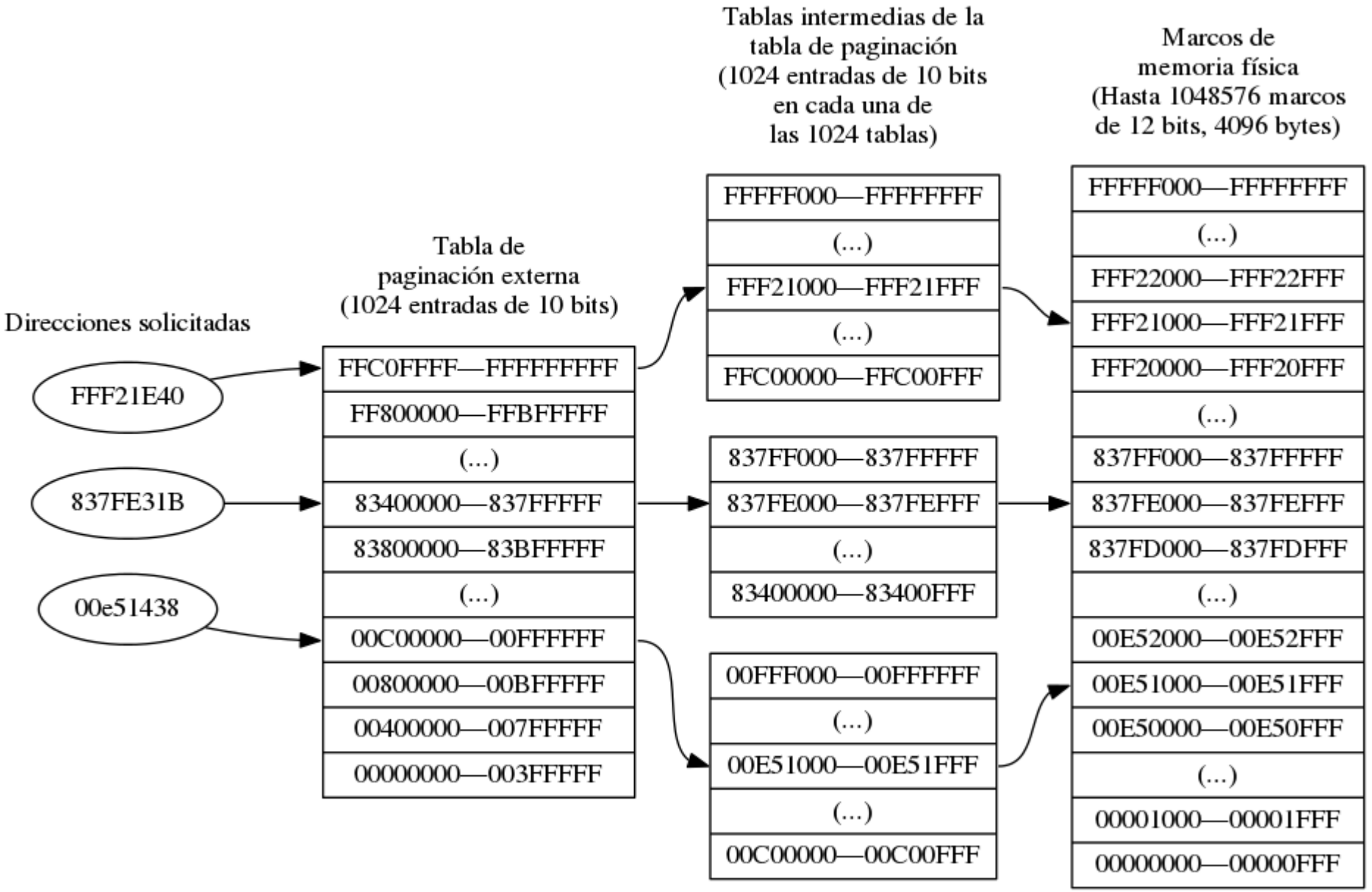
- Generalmente hay hardware especializado:



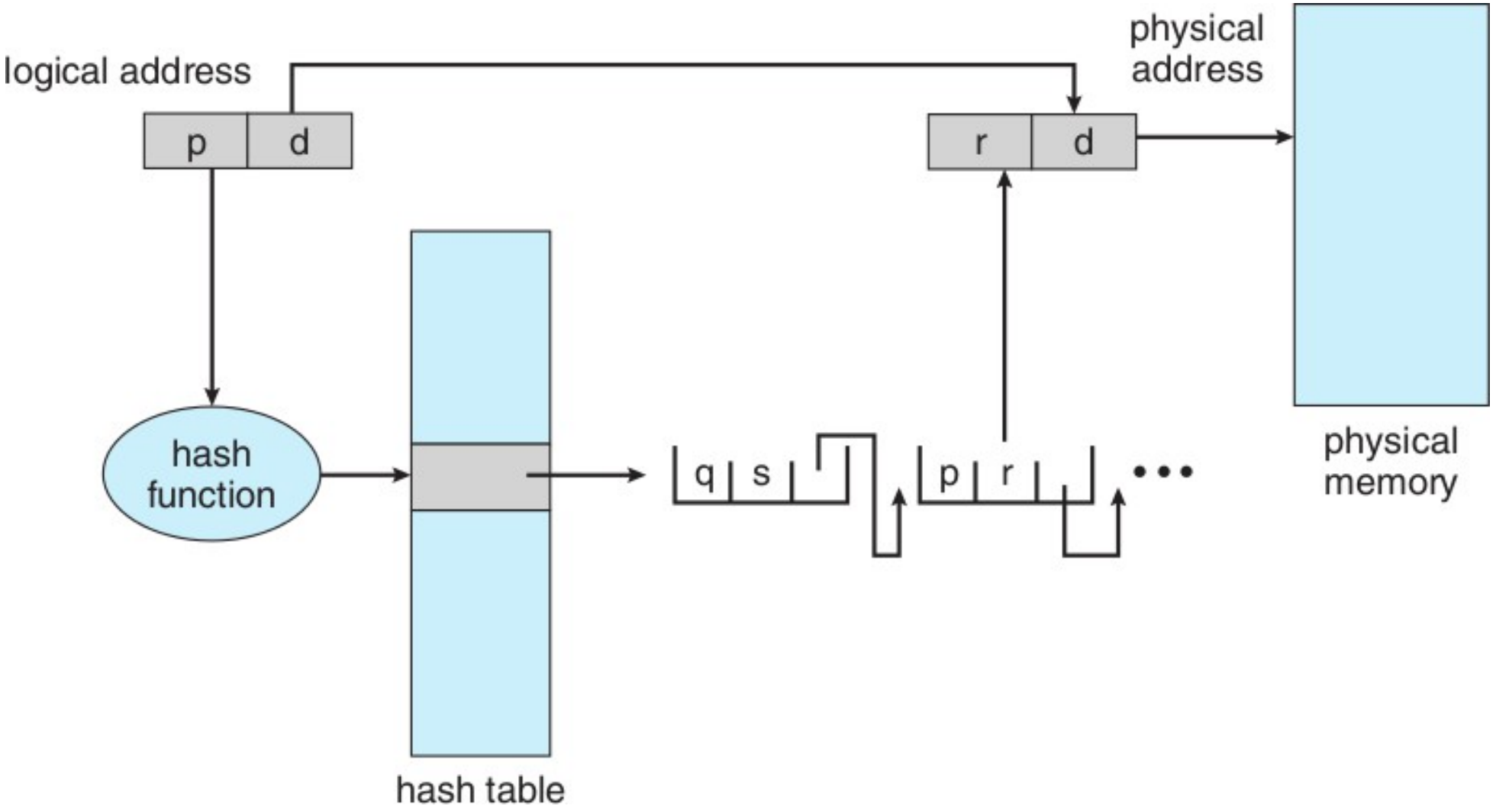
Paginación

- Ahora las direcciones virtuales se agrupan de a páginas.
- Cada página lógica (tamaño “estándar” 4Kb) puede corresponderse a una página (o marco) físico en memoria.
- Un proceso ve un rango alto de direcciones virtuales que puede estar mapeado a un cierto número de página físicas (nunca mayor al nro. de páginas físicas de la memoria principal).
- ¿qué pasa con las restantes?

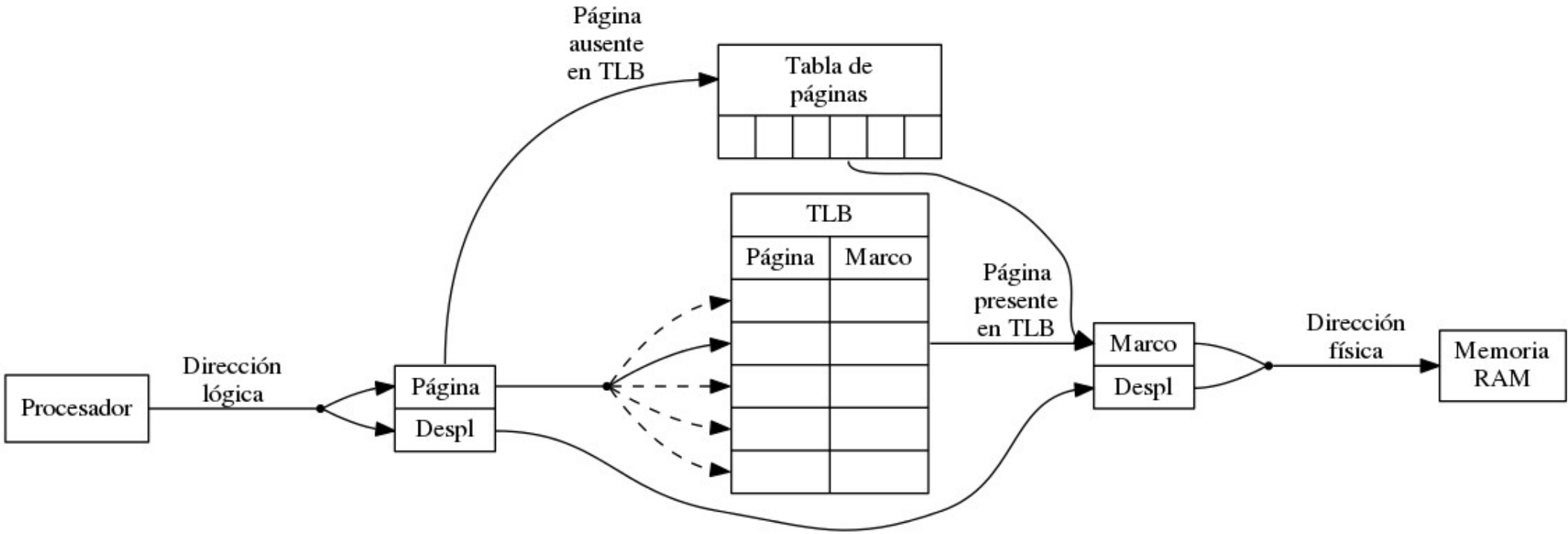
Mejora 1: Paginación Jerárquica



Mejora 2: Tablas hash



Combinando las dos: TLB pequeño



TLB: Translation Lookaside Buffer (buffer de traducción -caché-)

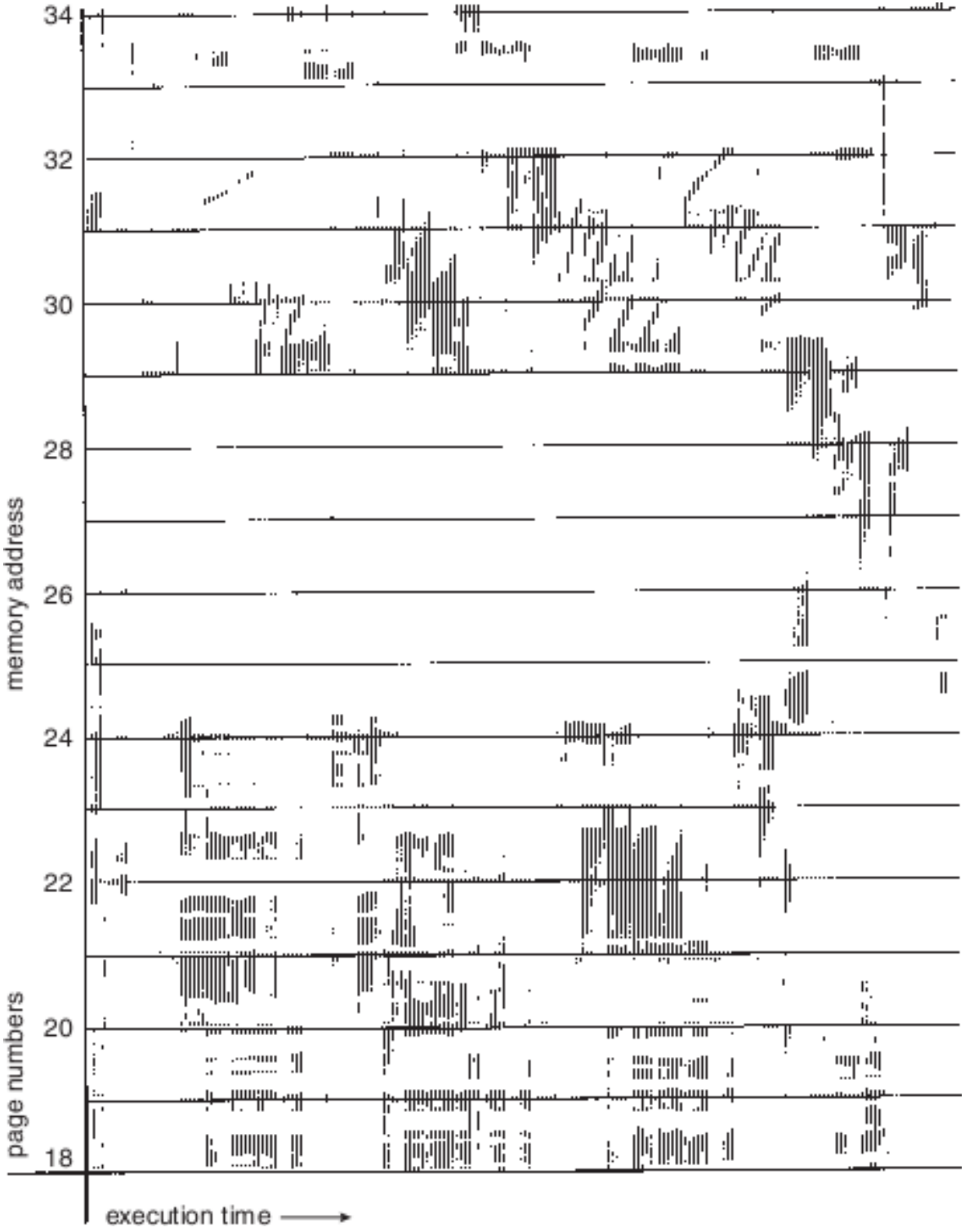
¿paginación + segmentación?

- Se puede en algunas arquitecturas. La mayoría de los SO actuales utilizan sólo paginación.

Otra alternativa

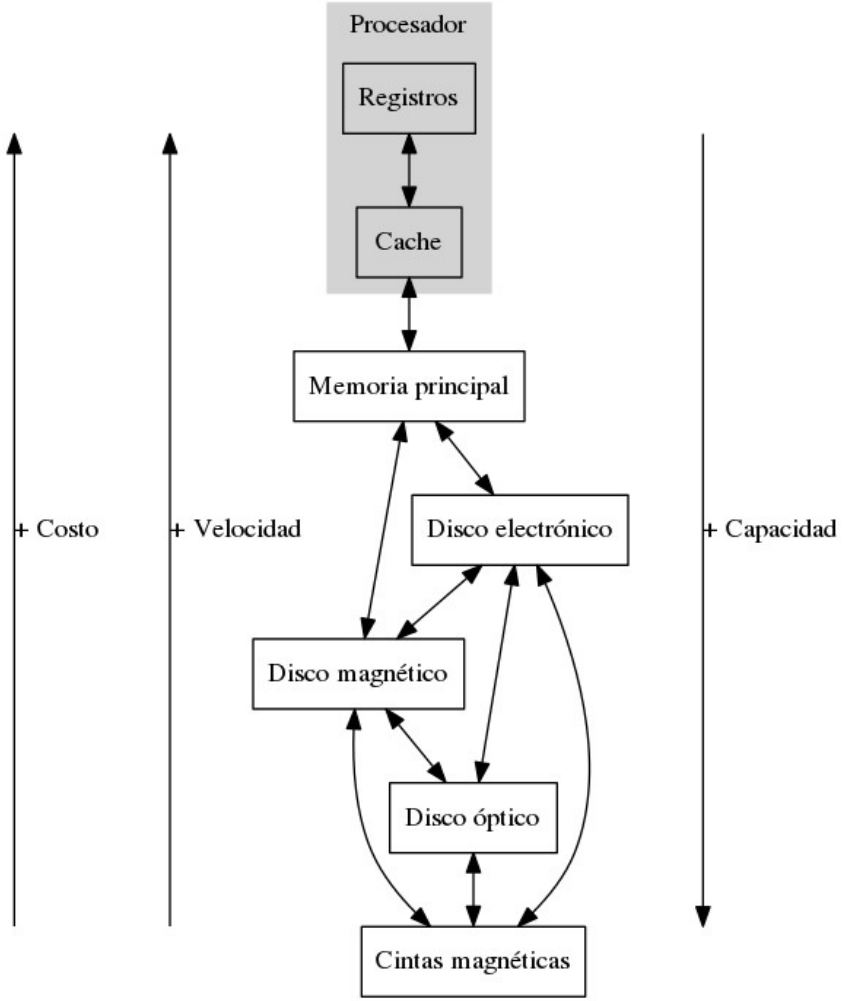
- Tabla de paginación invertida

Localidad de referencias

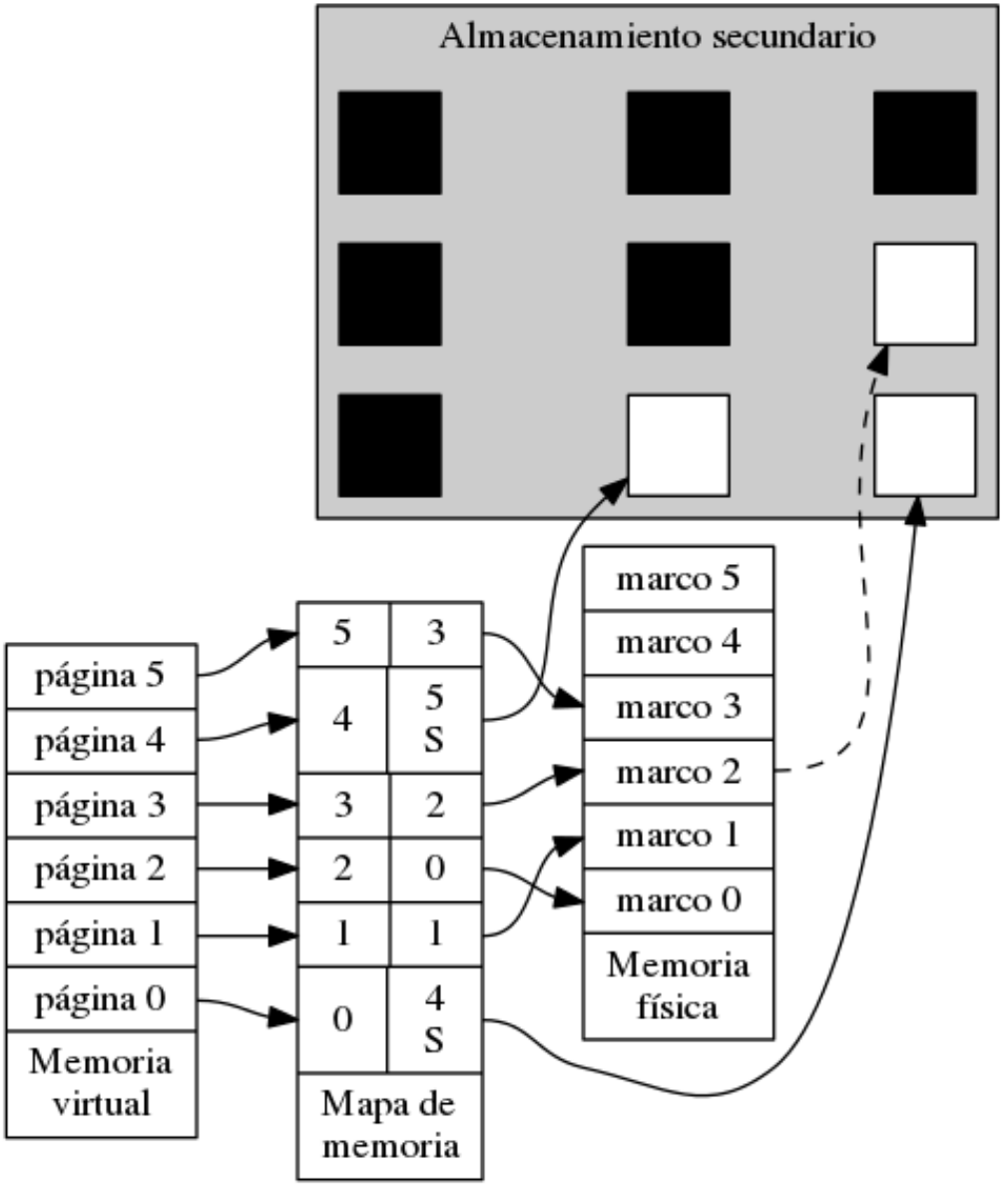


Repasemos entonces

- Jerarquía de memoria

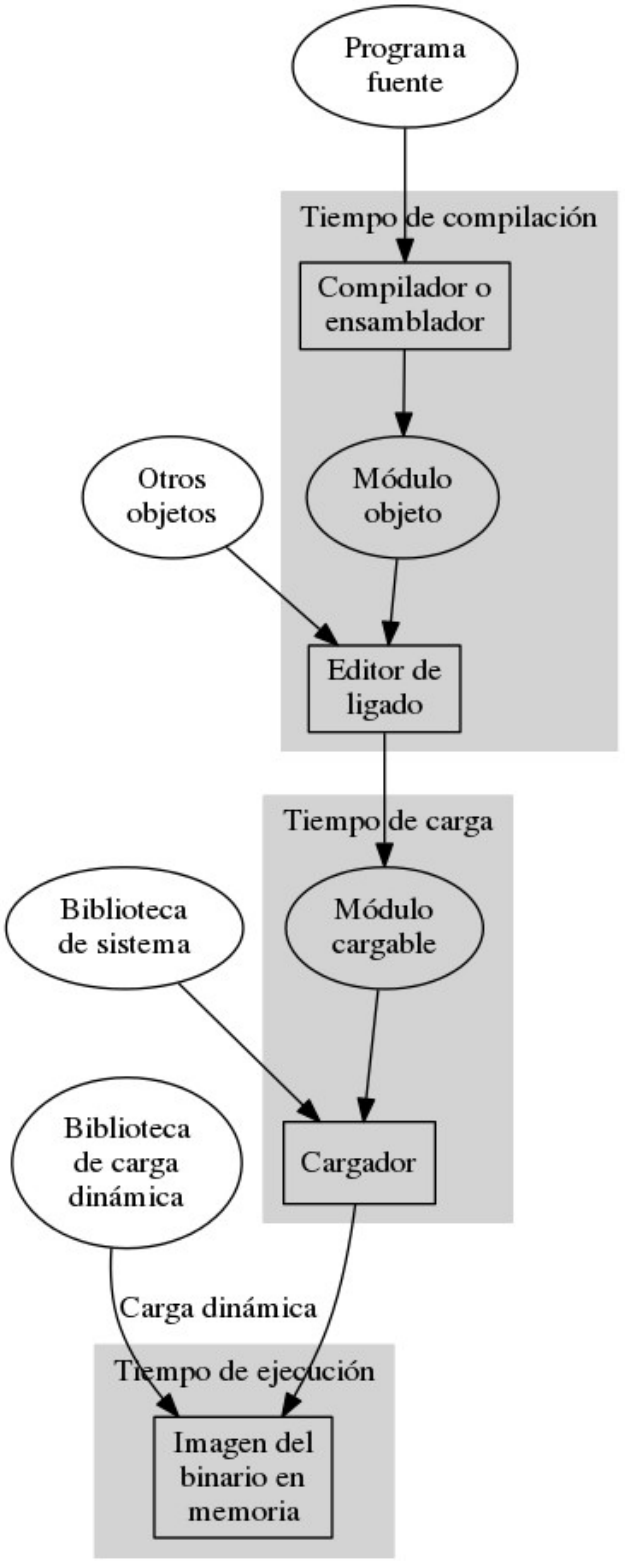


- Esquema de memoria virtual



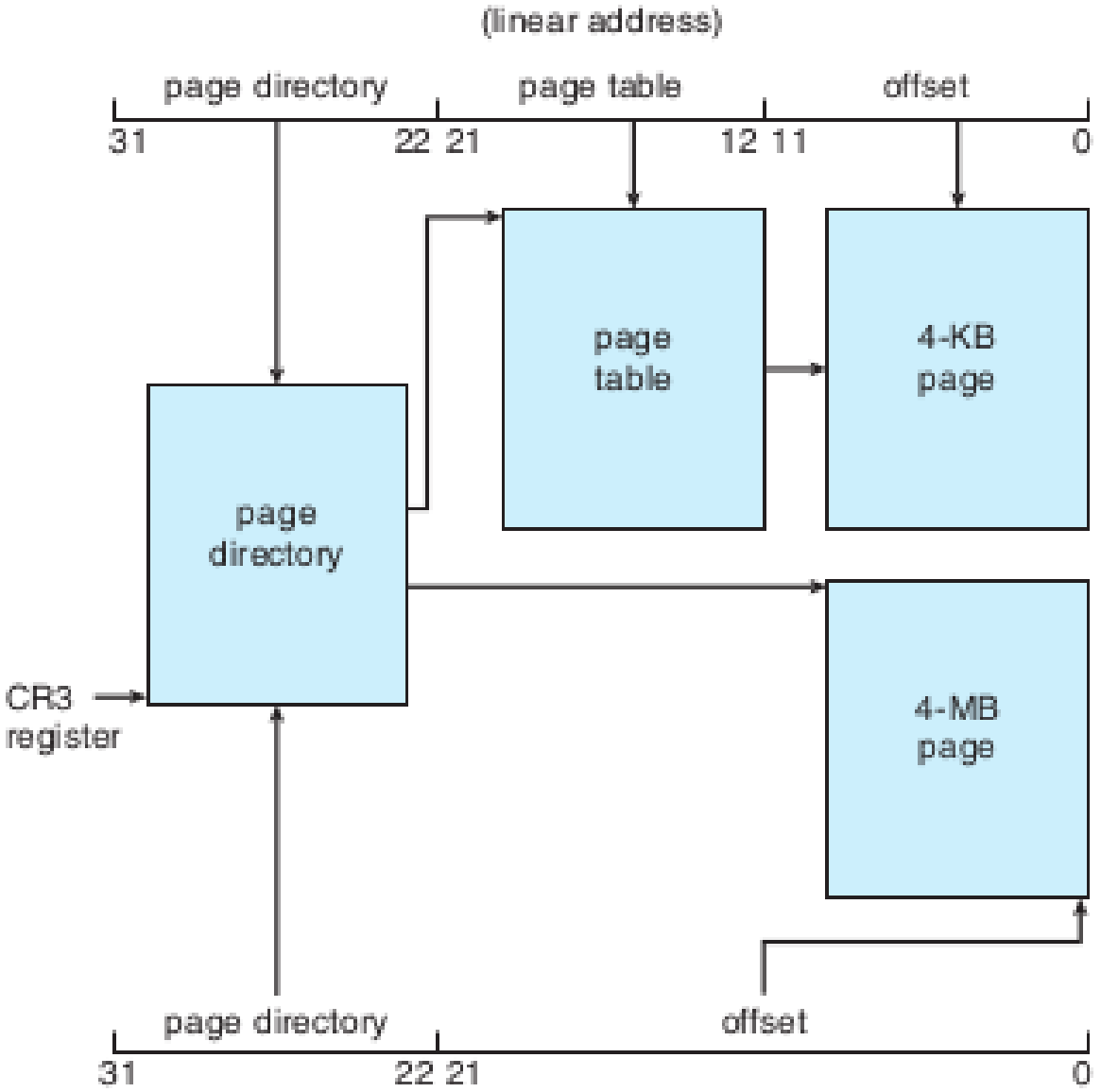
¿Quién/Cuando se organizan los segmentos y los rangos de direcciones de un proceso?

- Varios



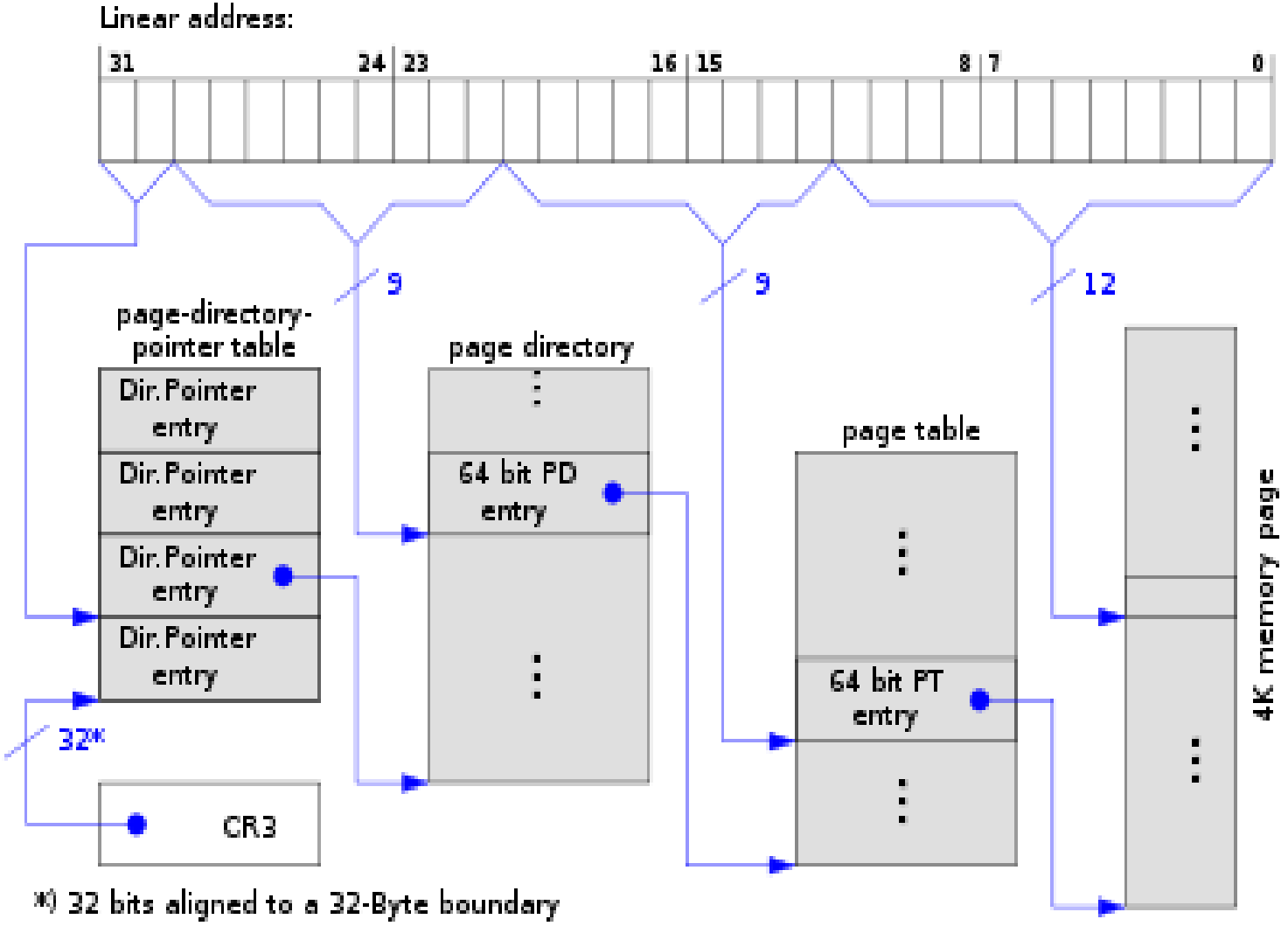
¿Qué pasa en i386?

- Modelo usado por el Pentium (dir. físicas 32 bits):



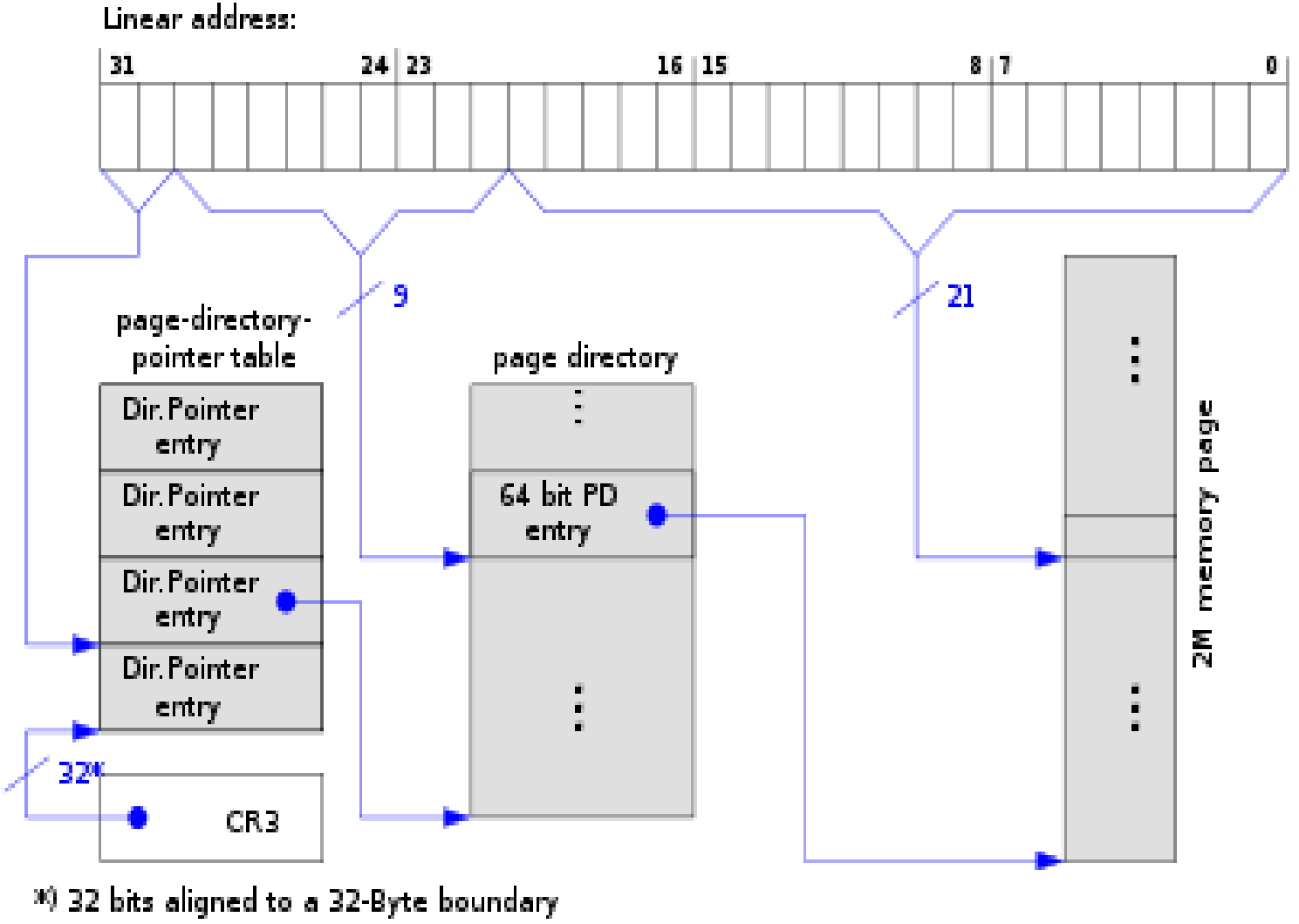
¿Más de 4Gb de RAM?: PAE (Physical Address Extension)

Con páginas de 4 Kb



PAE (Physical Address Extension)

Con páginas de 2 Mb



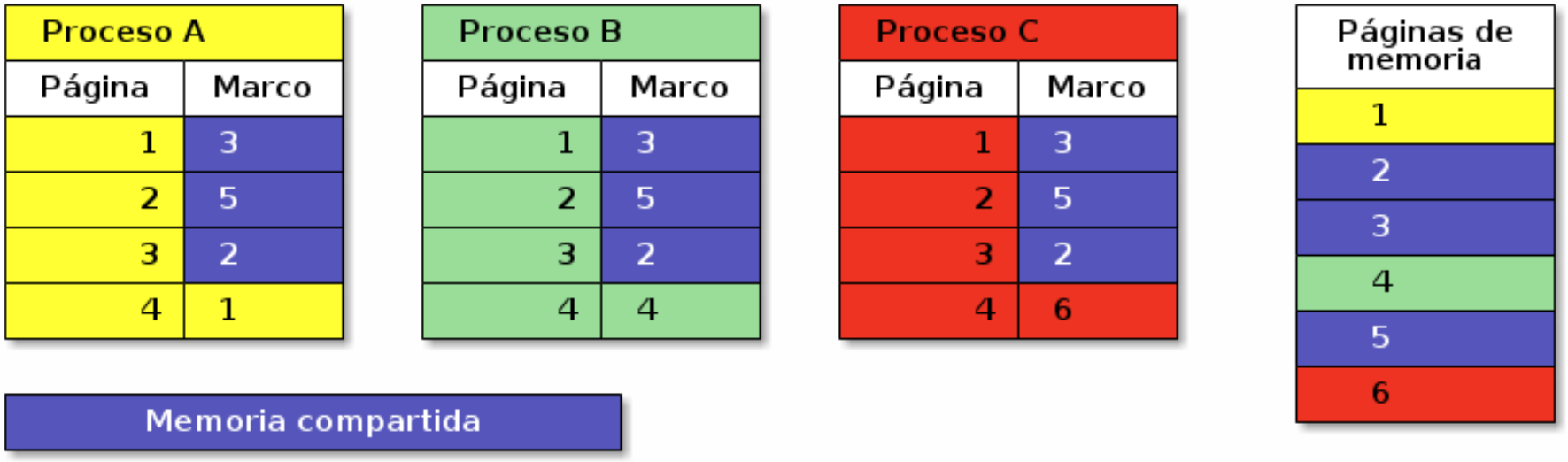
x86_64

- Usa un superset de PAE.
- Añade un nivel más de entradas
- Permite páginas de 1Gb
- ¿cuánto ocupa una tabla de paginación completa para un espacio de dirs. virtuales de 48 bits?

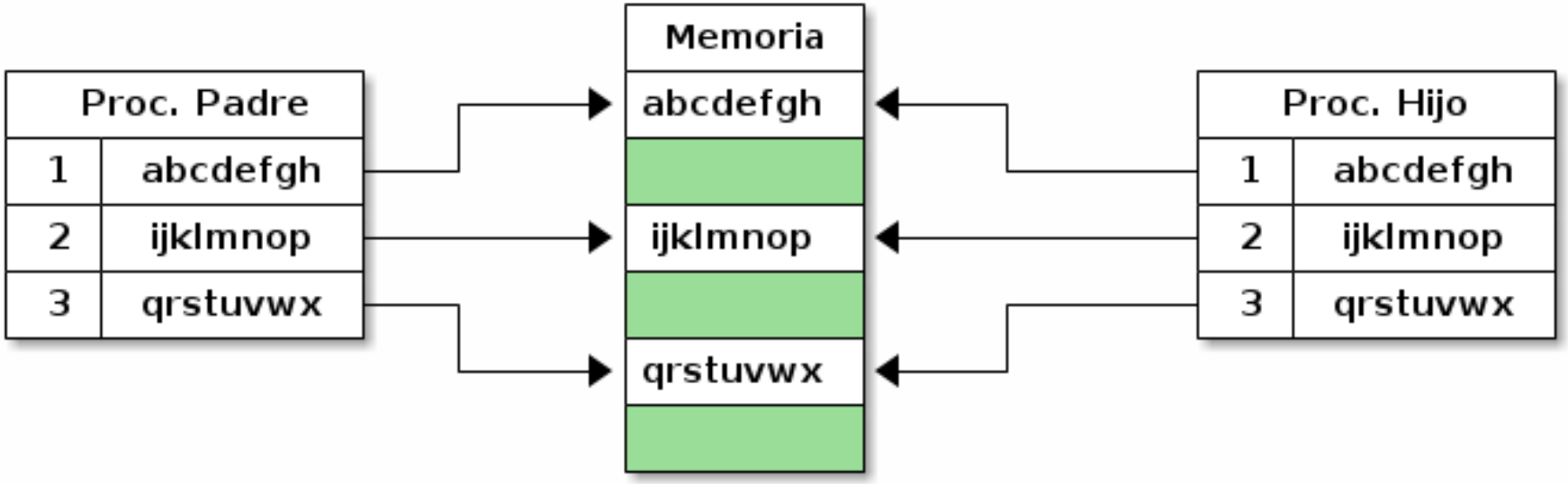
  Algo más de 512 Gb de RAM (wikipedia)

  ¿cómo se soluciona? Sencillamente, no se crean las tablas que no son necesarias (huecos).

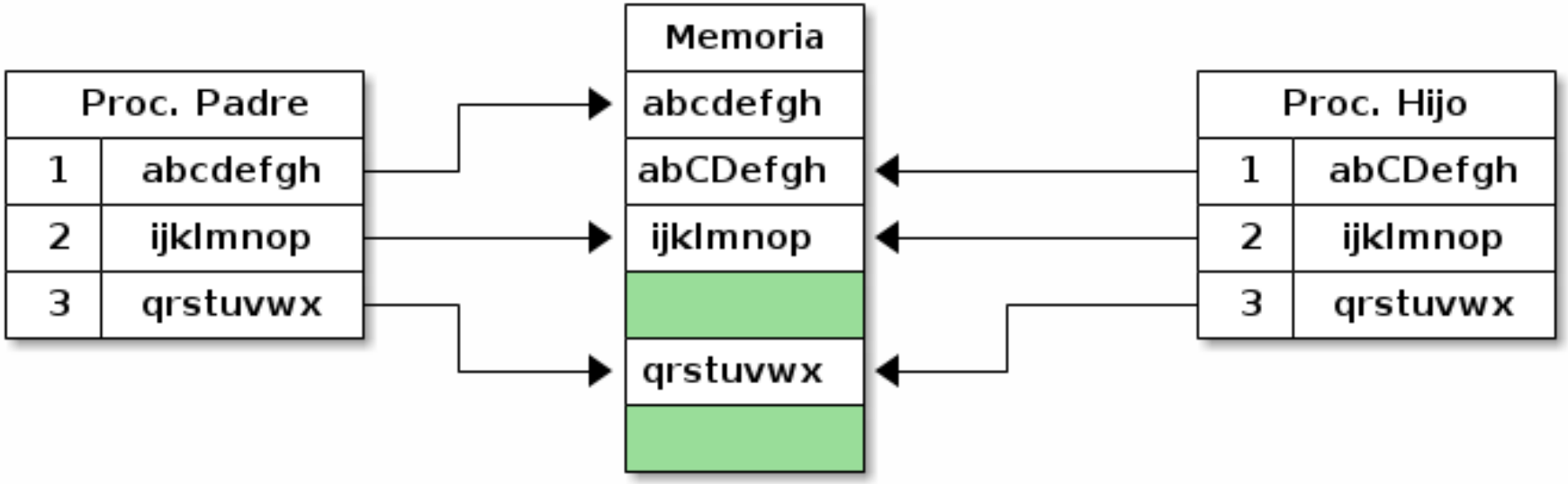
Memoria compartida



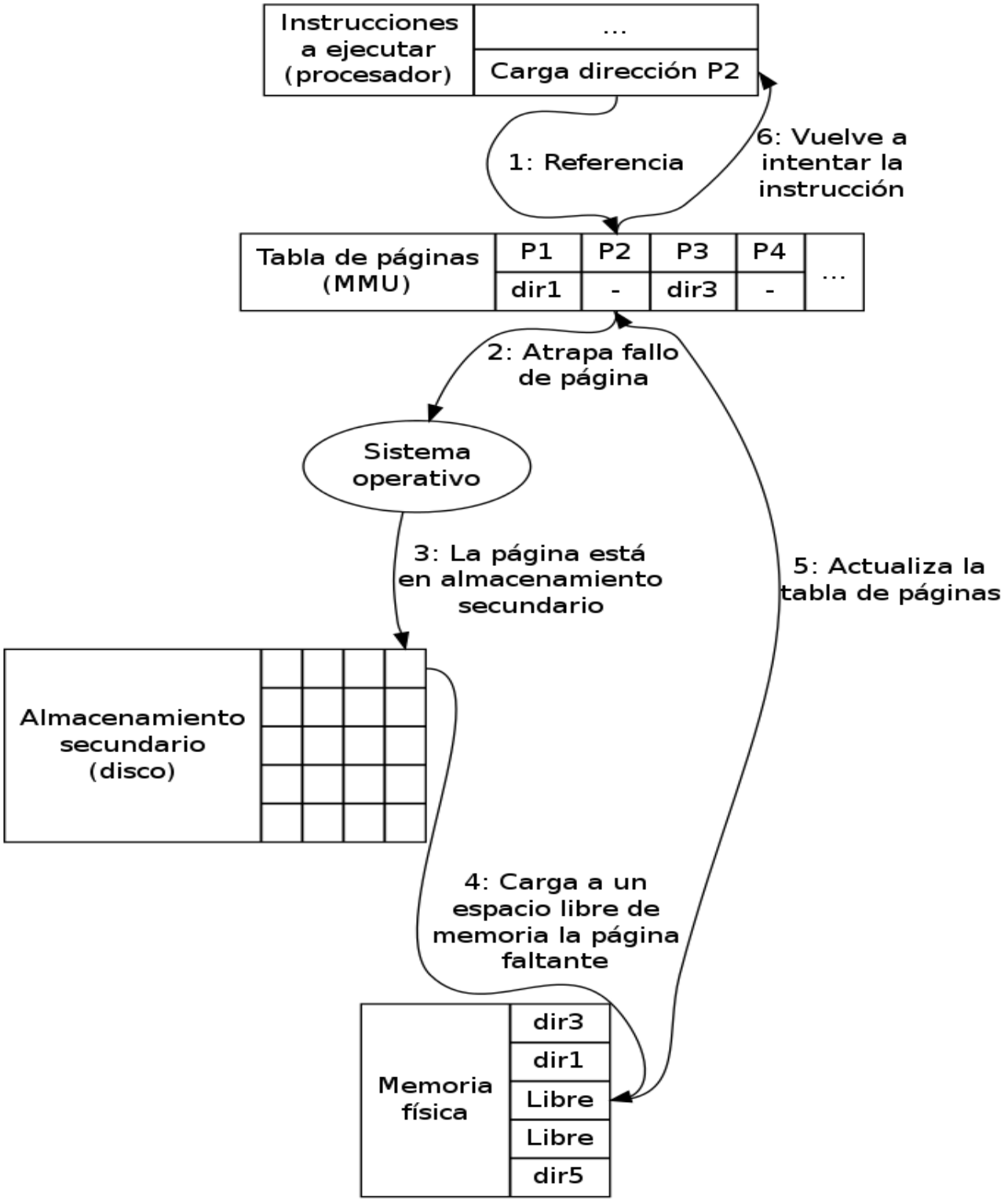
Copy on Write



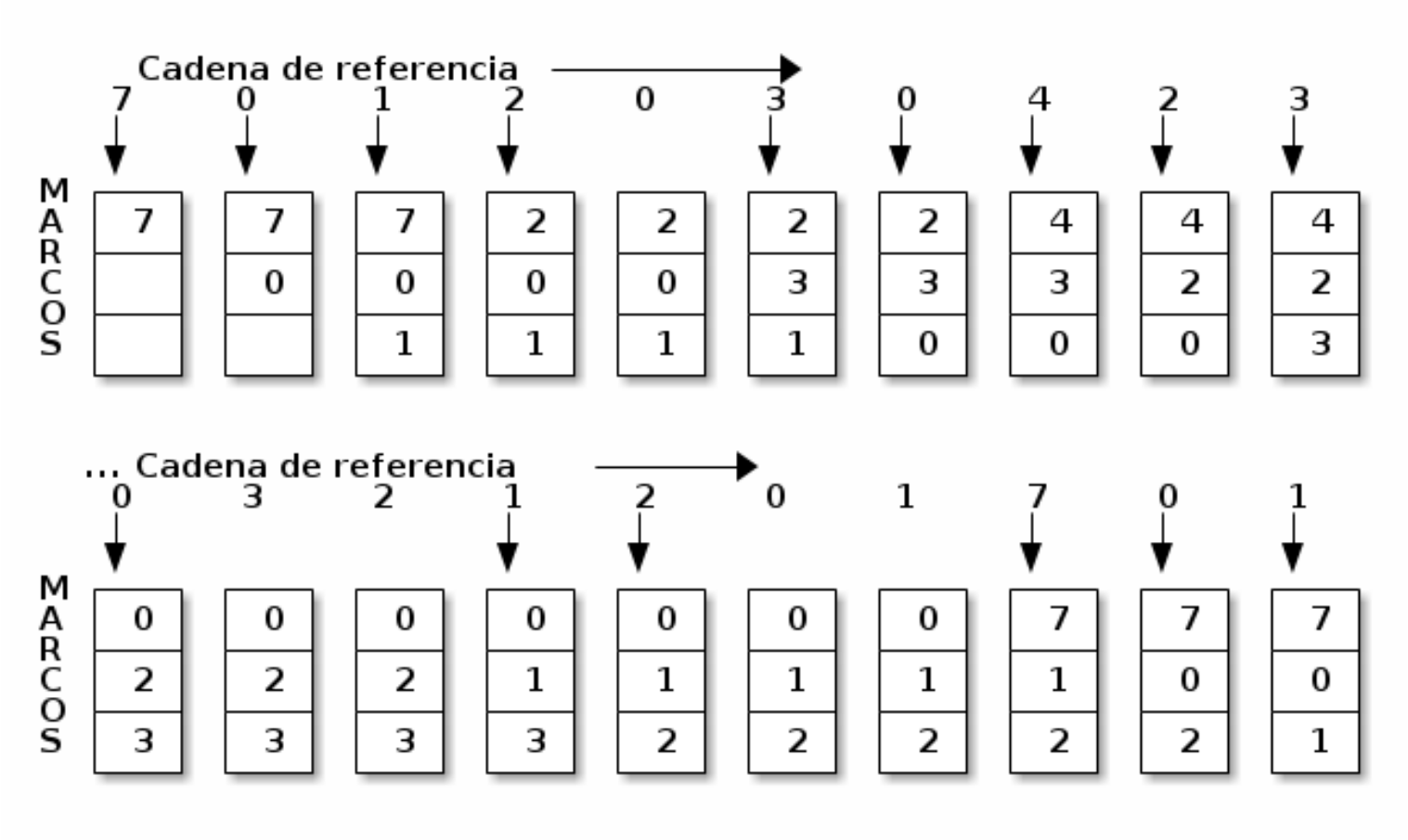
Copy on Write



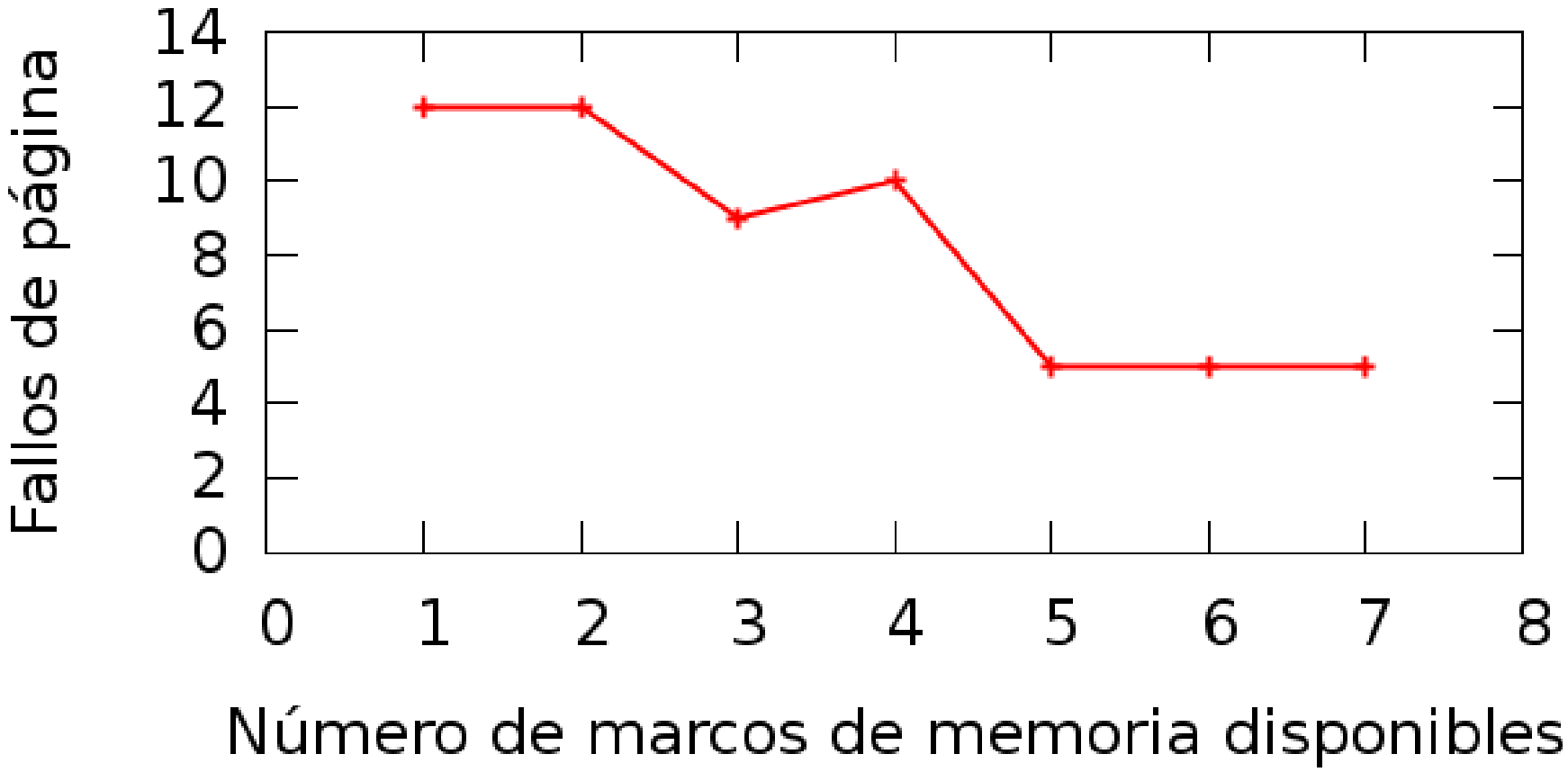
Repaso del mecanismo de swap



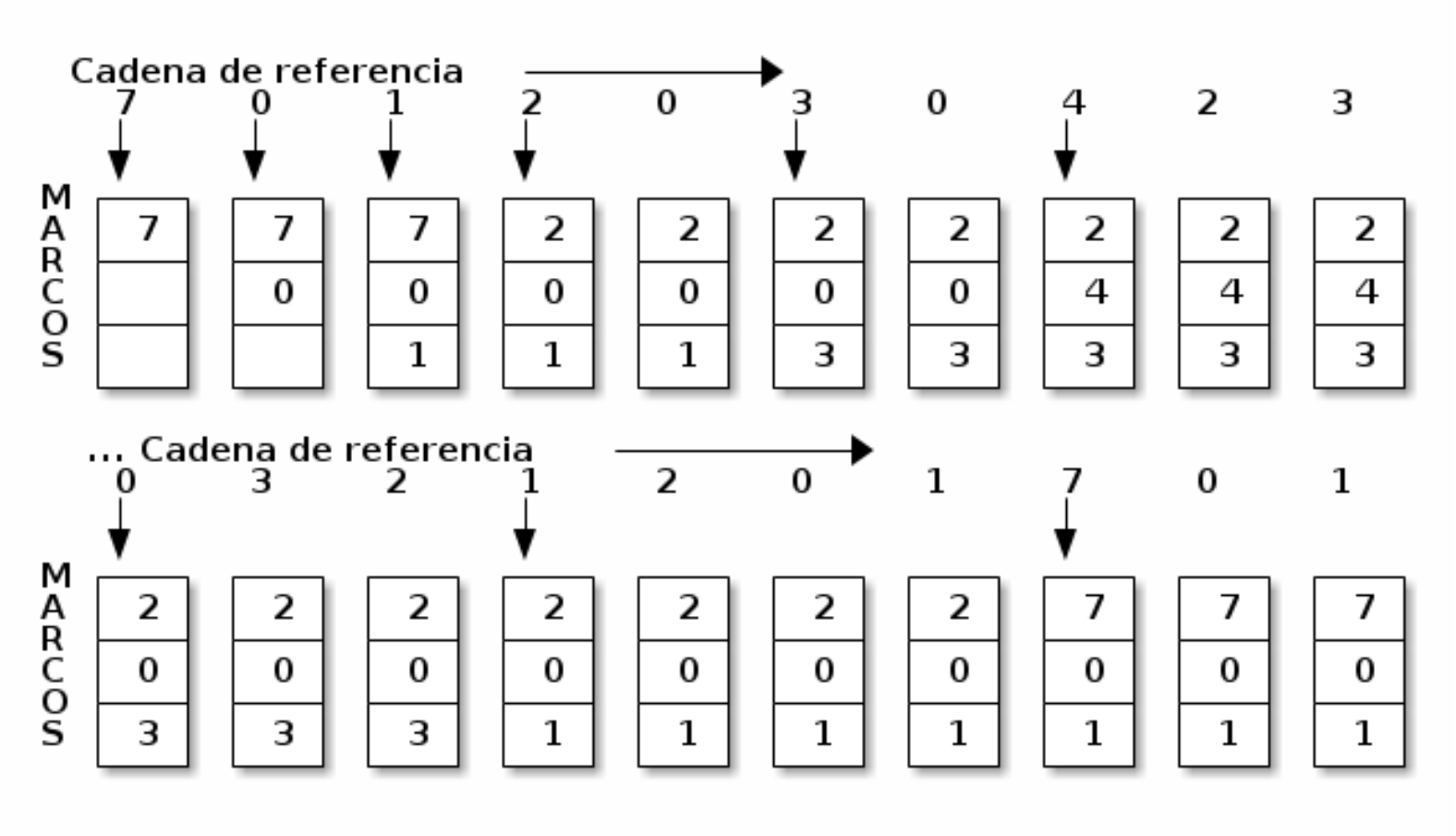
¿Qué reemplazar? Política FIFO



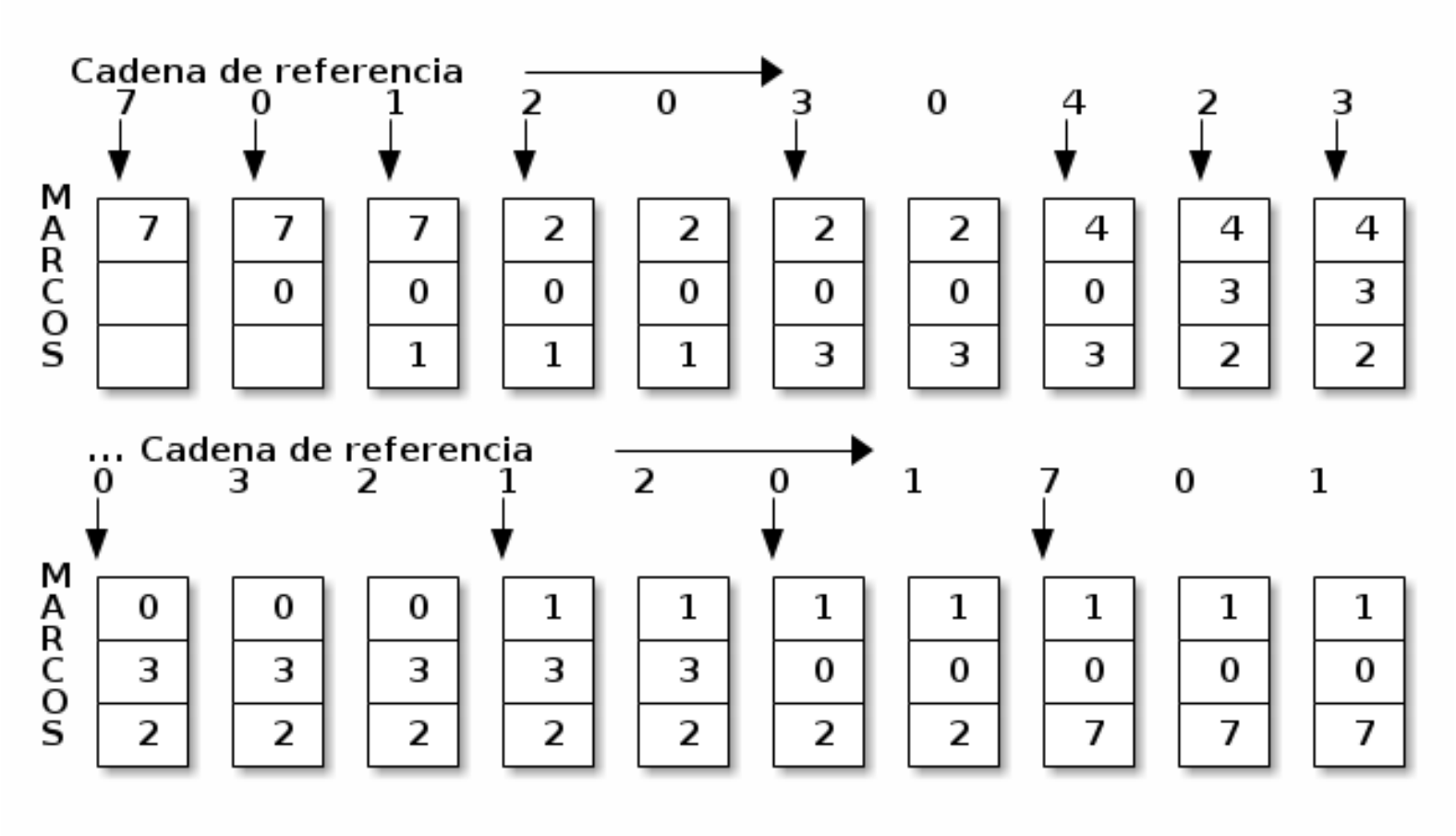
Anomalia de Belady



Algoritmo “óptimo”



LRU


Cadena de referencia
7 0 1 2 0 3 0 4 2 3
MARCOS
7 7 7 2 2 2 2 4 4 4
0 0 0 0 0 0 0 3 3
1 1 1 3 3 3 2 2
... Cadena de referencia
0 3 2 1 2 0 1 7 0 1
MARCOS
0 0 0 1 1 1 1 1 1 1
3 3 3 3 3 0 0 0 0 0
2 2 2 2 2 2 2 7 7 7