Tema 6. Gestión de memoria

UNIVERSITAT JAUME·I

Índice

- Introducción
  - ✓ Objetivos del sistema de gestión de memoria
  - ✓ La unidad de gestión de memoria (MMU)
  - ✓ Compartición de memoria
  - ✓ Protección de memoria
  - ✓ Intercambio
- Mapa de memoria de un proceso

Tema 6. Gestión de memoria

UNIVERSITAT JAUME·I

Índice (cont.)

- Gestión de la memoria principal
  - ✓ Particiones
  - ✓ Paginación
  - ✓ Segmentación
  - ✓ Segmentación paginada
- Gestión de la memoria virtual
  - ✓ Motivación
  - ✓ Paginación bajo demanda
  - ✓ Paginación multinivel
  - ✓ Algoritmos de reemplazo de página
  - ✓ Asignación de memoria central a procesos
  - ✓ Hiperpaginación

Tema 6. Gestión de memoria

UNIVERSITAT JAUME·I

Bibliografía

- J. Carretero et al. *Sistemas Operativos: Una Visión Aplicada.* McGraw-Hill. **2ª edición 2007**. Capítulo 5.

- M. A. Castaño, J. Echagüe, R. Mayo, C. Pérez. *Problemas de Sistemas Operativos.* Col.lecciò “Materials”. Servicio de Publicaciones de la UJI, num. 109. 2000. Capítulo 4.

Tema 6. Gestión de memoria

UNIVERSITAT JAUME·I

Índice

Objetivos del gestor de memoria

- SO multiplexa recursos entre procesos
  - Cada proceso cree que tiene una máquina para él solo
  - Gestión de procesos: Reparto de procesador
  - Gestión de memoria: Reparto de memoria

- **¿Qué hace el gestor de memoria?**
  - Ofrecer a cada proceso un espacio lógico propio
  - Proporcionar protección entre procesos
  - Permitir que procesos compartan memoria
  - Maximizar el grado de multiprogramación
  - Proporcionar a los procesos mapas de memoria muy grandes

Espacios lógicos independientes

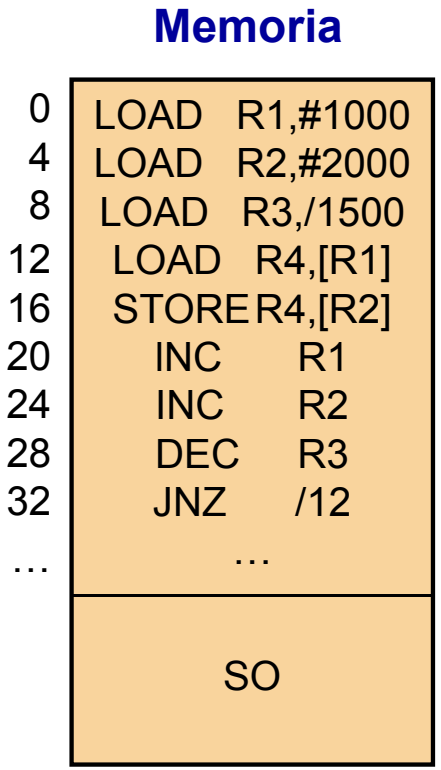
- No se conoce posición de memoria donde un programa se ejecutará
  - Cada proceso cree que tiene una máquina para él solo
  - Código en ejecutable genera referencias desde 0
  - Ejemplo: Programa que copia un vector
    - Vector origen A partir de posición #1000
    - Vector destino: A partir de posición #2000
    - Tamaño vectores: En posición #1500

Fichero ejecutable

0 … 96	Cabecera
100	LOAD R1,#1000
104	LOAD R2,#2000
108	LOAD R3,/1500
112	LOAD R4,[R1]
116	STORE R4,[R2]
120	INC R1
124	INC R2
128	DEC R3
132	JNZ /12
	…
1100	5
1104	3
	…

Ejecución en un SO monoprogramado

- SO, por ejemplo, en posiciones más altas
- Programa se carga a partir de dirección 0
- Se le pasa el control al programa



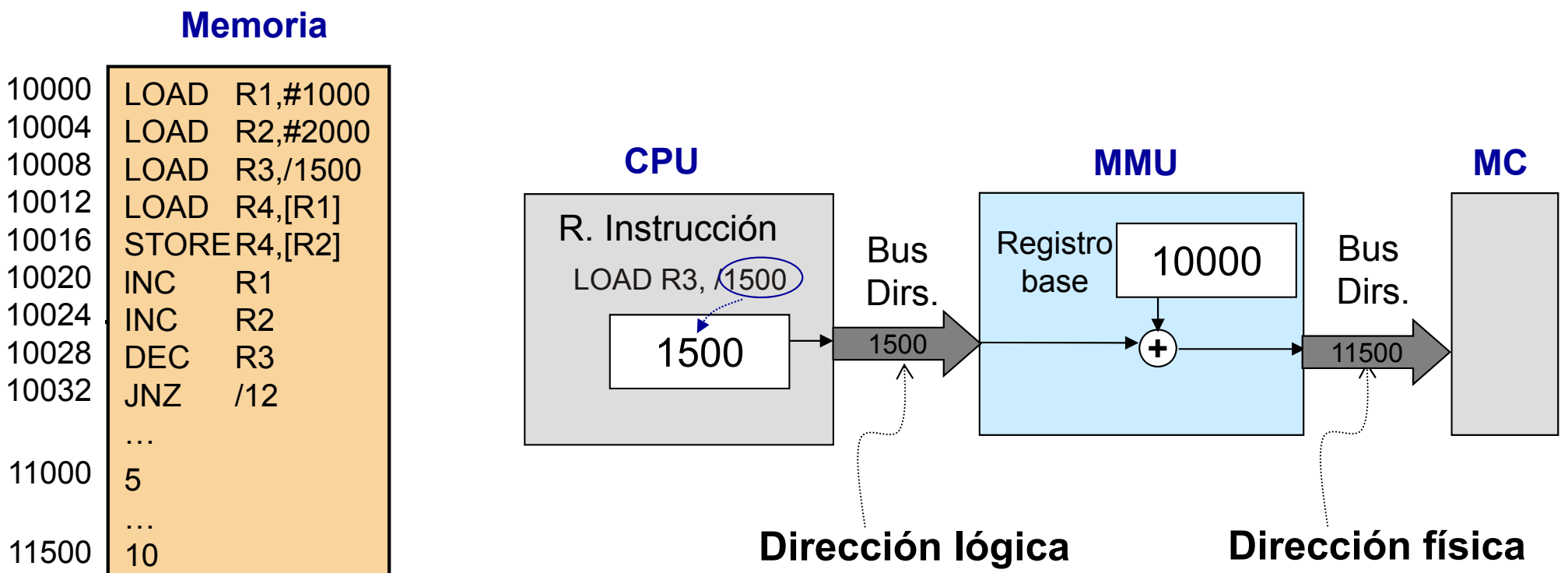
Traducción de direcciones lógicas a físicas

- Reubicación necesaria en SO con multiprogramación
  - Implica traducir direcciones lógicas a físicas
- **Espacio físico y lógico de direcciones:**
  - Dirección lógica: Dirección generada por la CPU
    Dirección física: Dirección real de MC a la que se accede
  - Espacio de direcciones lógicas: Conjunto de direcciones lógicas generadas por un programa
    Espacio de direcciones físicas: Conjunto de direcciones físicas generadas por un programa
- La traducción crea espacios independientes para cada proceso
- Dos alternativas:
  - Traducción HW o SW

Traducción HW

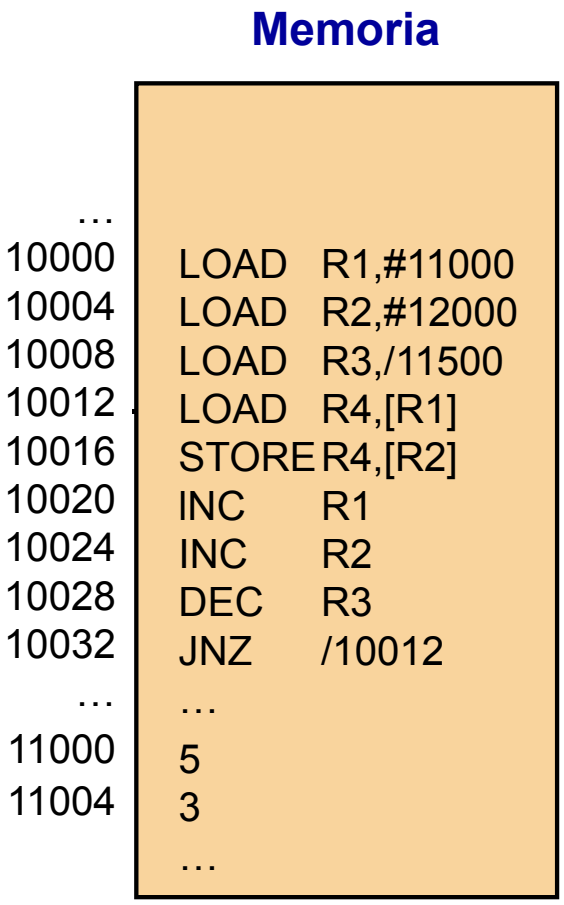
- **Unidad de gestión de memoria ("Memory Management Unit", MMU**):
  - Dispositivo HW que transforma direcciones lógicas en físicas en tiempo de ejecución
  - El programa de usuario maneja direcciones lógicas, nunca ve direcciones físicas reales
  - El programa se carga en memoria sin modificar

Traducción HW



Traducción SW

- Usada en SO antiguos sin MMU
- Traducción de direcciones hecha por el SO durante la carga del programa
- Programa en memoria distinto del ejecutable
- Desventaja:
  - No permite mover el programa en tiempo de ejecución

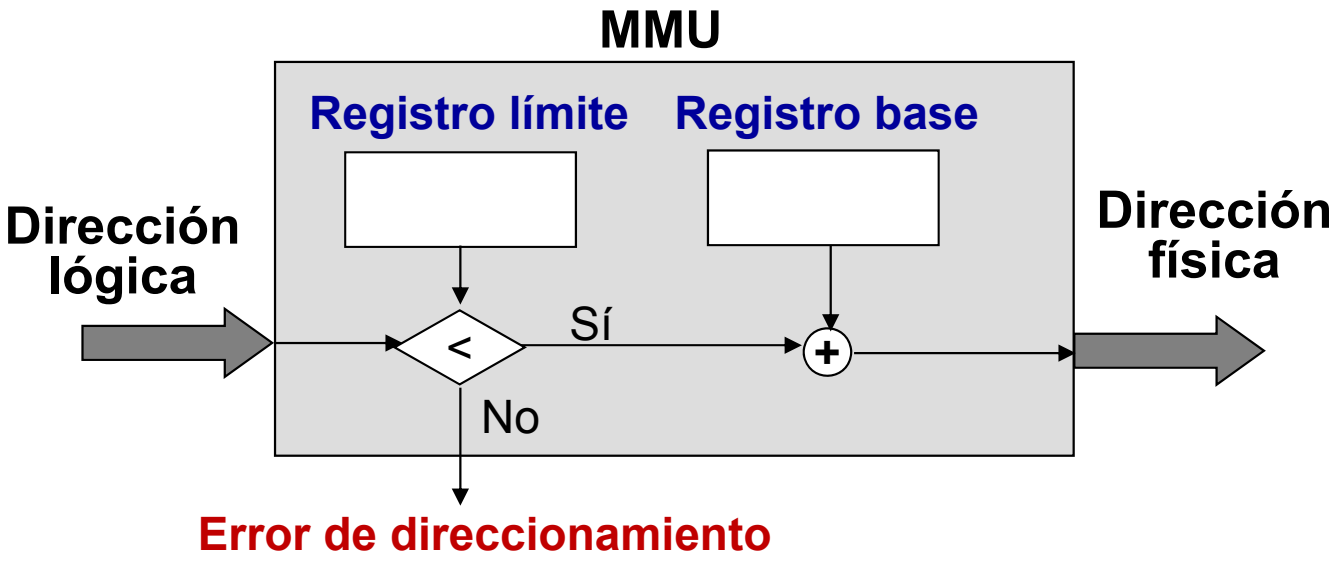


Protección de memoria

- Necesidad de:
  - Protección entre espacio de SO y de usuarios
  - Protección entre espacio de usuarios
  - Validar todas las direcciones que genera el programa
    - Lo realiza el MMU

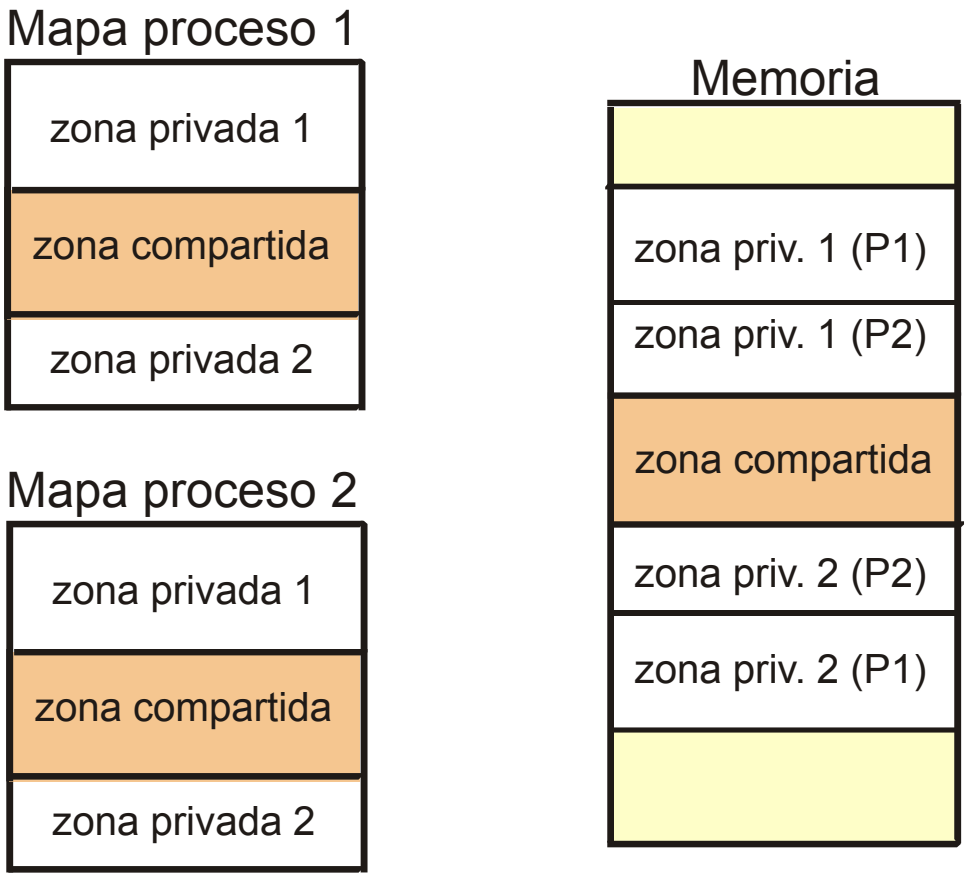
Protección de memoria

- **Mecanismo de protección**:
  - **Registro base**: Dirección inicio partición asignada a un proceso.
  - **Registro límite**: Tamaño de partición asignada a un proceso.



Compartición de memoria

- Direcciones lógicas de dos o más procesos se corresponden con la misma dirección física



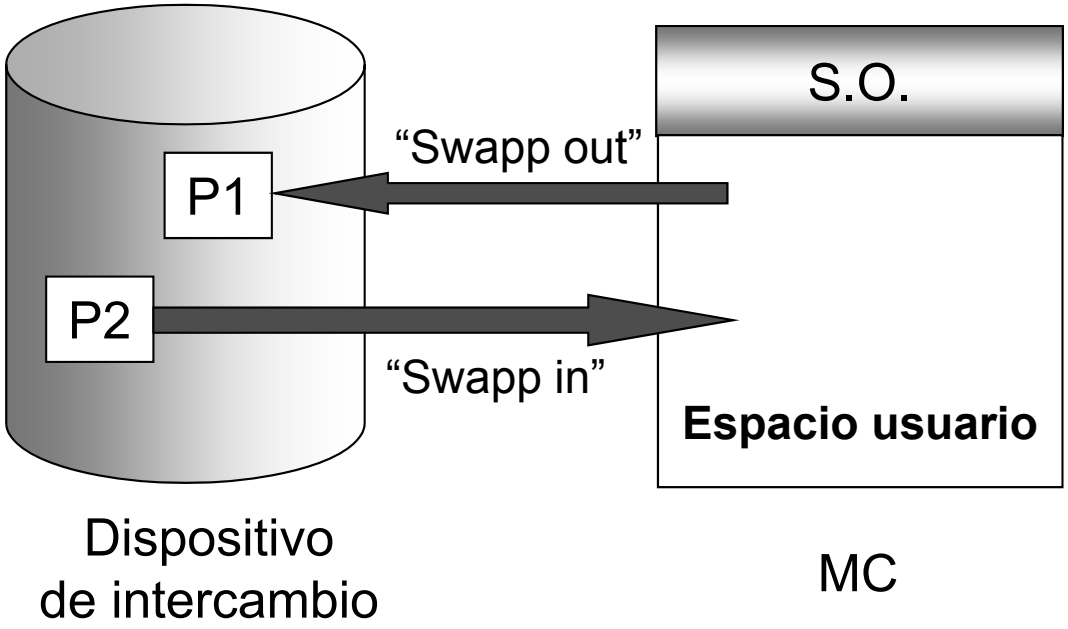
Compartición de memoria

- Compartición de código y/o datos entre procesos
  - Carga única de una librería o un programa solicitado por varios procesos
  - Compartición de memoria entre procesos concurrentes
- Restricciones de acceso:
  - Compartición de código → Acceso de sólo lectura
  - Compartición de datos → Acceso de lectura y/o escritura
- Accesos no permitidos provocan interrupción de violación de protección de memoria

Intercambio ("Swapping")

- Si no hay espacio suficiente en memoria, un proceso puede almacenarse temporalmente en disco

  Cuando se reanude su ejecución, se vuelve a llevar a memoria



Intercambio ("Swapping")

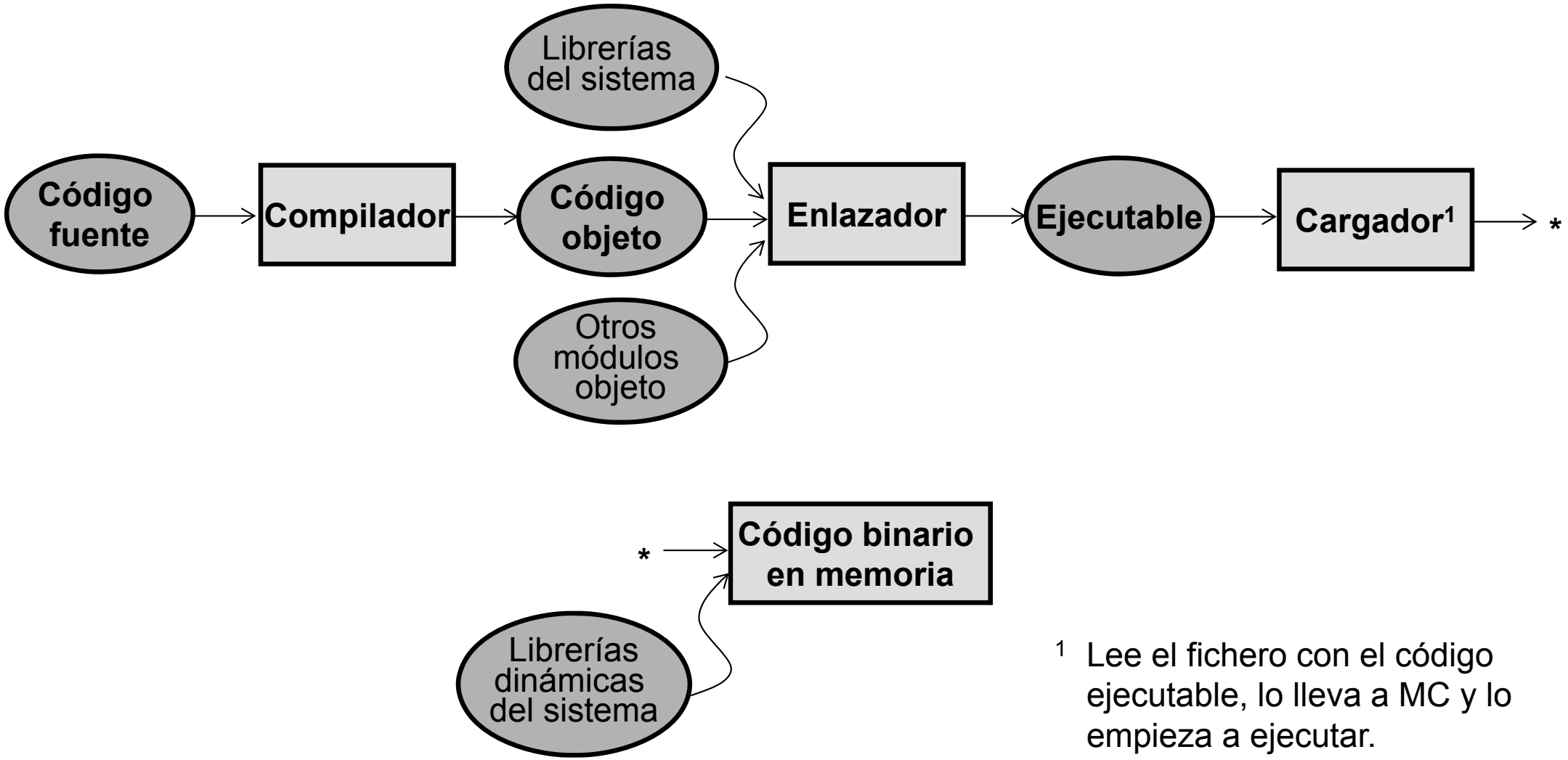
- Tiempo de intercambio:
  - Mayoritariamente es tiempo de transferencia
  - Proporcional a la cantidad de memoria intercambiada
- Diversos criterios de selección del proceso a expulsar:
  - Dependiendo de la prioridad del proceso, el tamaño del mapa de memoria,…
  - Preferiblemente un proceso bloqueado
- No es necesario copiar todo el mapa (ni código ni huecos)

UNIVERSITAT JAUME·I

Índice

- Introducción
  - ✓ Objetivos del sistema de gestión de memoria
  - ✓ La unidad de gestión de memoria (MMU)
  - ✓ Compartición de memoria
  - ✓ Protección de memoria
  - ✓ Intercambio
- ☞ Mapa de memoria de un proceso

Fases en la generación de un ejecutable

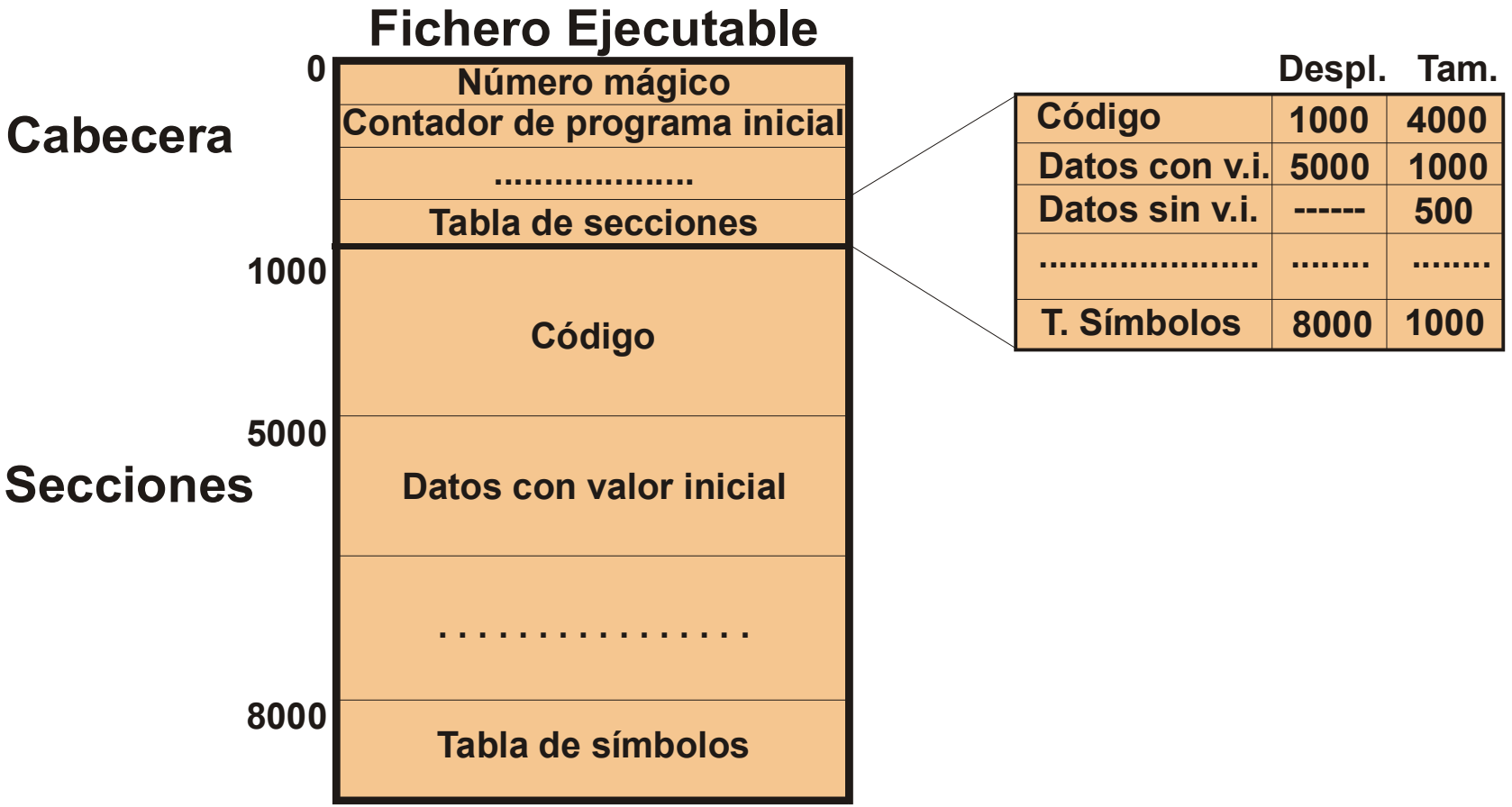


[1] Lee el fichero con el código ejecutable, lo lleva a MC y lo empieza a ejecutar.

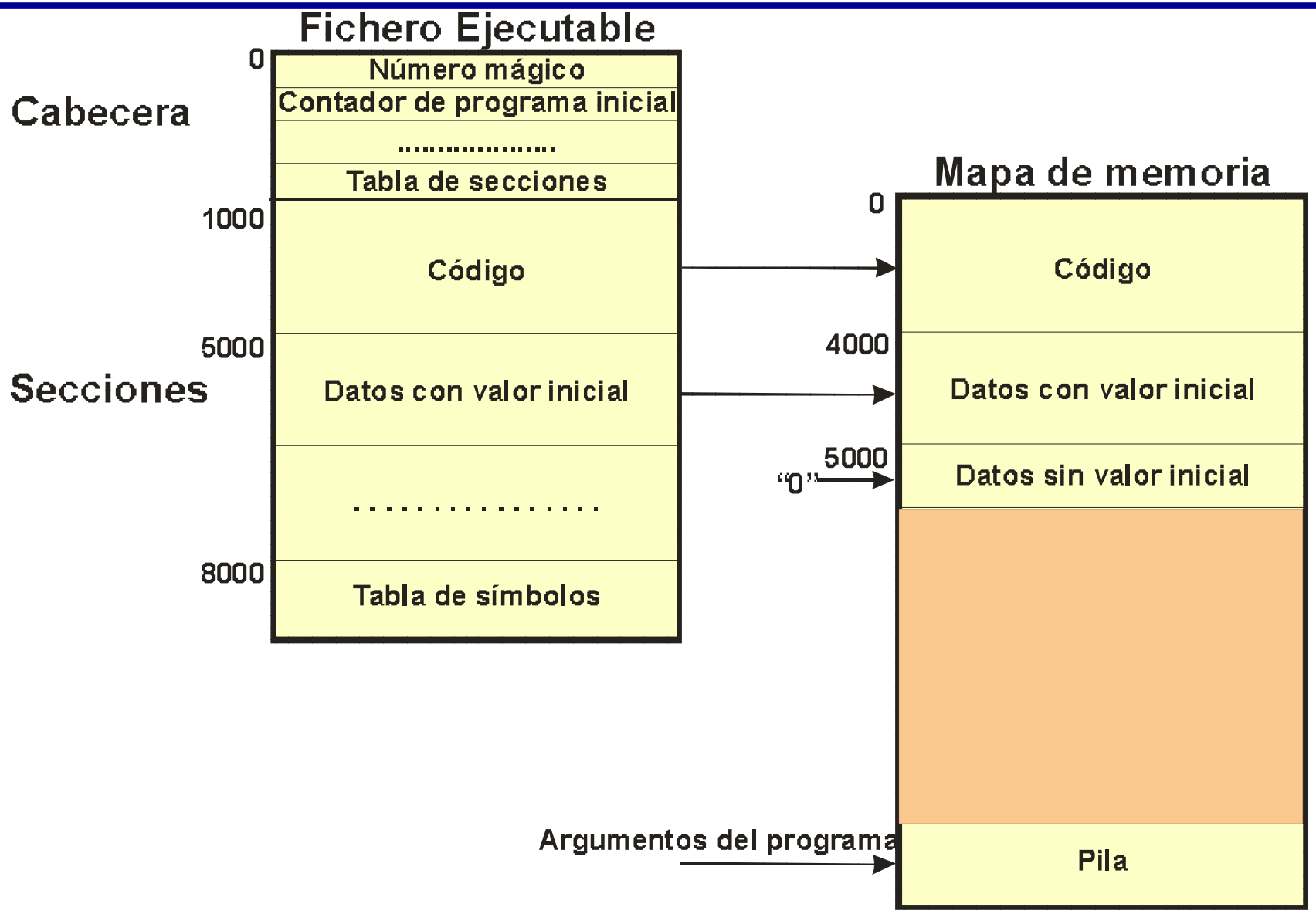
Formato del ejecutable

- Distintos fabricantes usan diferentes formatos
  - En Linux *Executable and Linkable Format* (ELF)
- Estructura: Cabecera y conjunto de secciones
- Cabecera:
  - Número mágico que identifica a ejecutable
  - Punto de entrada del programa
  - Tabla de secciones

Formato del ejecutable



Crear mapa de memoria desde ejecutable



Otras regiones del mapa de memoria

- Durante ejecución de proceso se crean nuevas regiones
  - Mapa de memoria con carácter dinámico
- Región de heap
  - Soporte de memoria dinámica (`malloc` en C)
- Archivo proyectado
  - Región asociada al archivo proyectado

Tema 6. Gestión de memoria

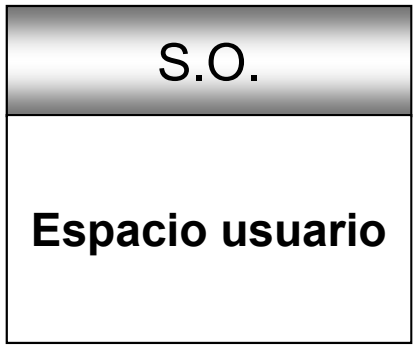
UNIVERSITAT JAUME·I

Índice

- Introducción
- Mapa de memoria de un proceso
- Gestión de la memoria principal
  - ✓ Particiones
  - ✓ Paginación
  - ✓ Segmentación
  - ✓ Segmentación paginada
- Gestión de la memoria virtual

Gestión de memoria mediante particiones

- Memoria dividida en dos particiones: SO y programas de usuario

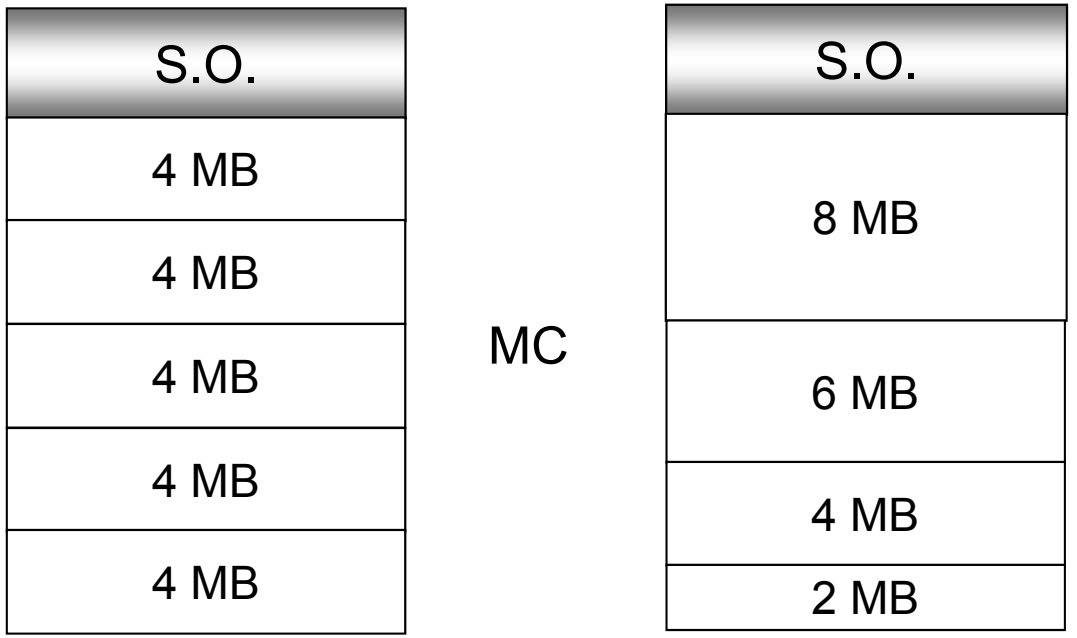


MC

- Programas de usuario en **posiciones contiguas de memoria**
- Compartición de memoria entre diferentes programas de usuario
- **Esquemas de asignación de memoria:**
  - Con particiones fijas
  - Con particiones variables o dinámicas

Gestión de memoria mediante particiones

- **Particiones fijas:**
  - Dos posibilidades:
    - Particiones de igual tamaño
    - Particiones de distintos tamaños



Gestión de memoria mediante particiones

- **Particiones fijas (cont.):**
  - Problemas:
    - Nivel de multiprogramación determinado por el número de particiones
    - Aparece **fragmentación interna**: hay memoria dentro de una partición que no se usa
    - Elección de la partición más adecuada cuando hay particiones de distintos tamaños

      Si, p.e., están ocupadas las particiones de tamaño 4 MB y hay que asignar memoria a un proceso de 3 MB:

      ¿Esperar liberación de una partición de 4MB?

      ¿Asignar partición libre mayor?

Gestión de memoria mediante particiones

- **Particiones dinámicas:**
  - Hueco: Partición libre para procesos de usuario
  - Mecanismo de asignación:
    - Inicialmente existe una única partición del tamaño de la memoria de usuario
    - A un proceso se le asigna un hueco en el que quepa la cantidad de memoria que necesita
      - ✓ Generación de una nueva partición
      - ✓ Creación de un hueco de tamaño diferente

Gestión de memoria mediante particiones

- **Particiones dinámicas (cont.):**
  - Necesidad de mantener listas con:
    - Particiones libres y tamaño de particiones
    - Particiones asignadas y tamaño de particiones

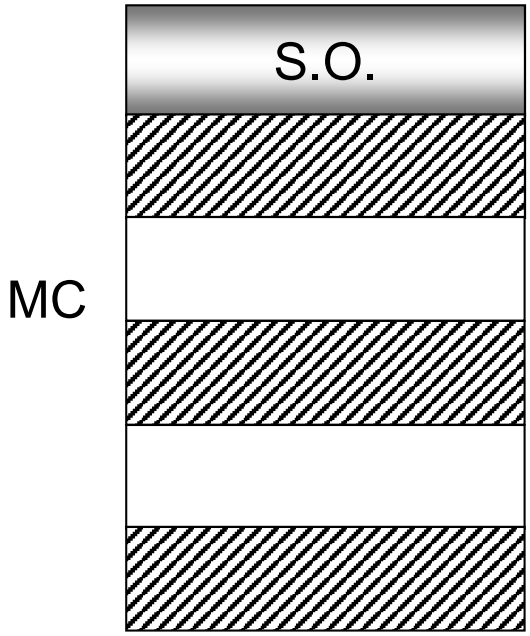
Gestión de memoria mediante particiones

UNIVERSITAT JAUME·I

- **Particiones dinámicas (cont.):**
  - **Políticas de asignación del espacio libre:**
    - Primer hueco (“Firtst Fit”):
      - ✓ Asignar el primer hueco lo suficientemente grande
    - Mejor hueco (“Best Fit”):
      - ✓ Asignar el menor hueco lo suficientemente grande
      - ✓ Requiere buscar en toda la lista de huecos o tener lista ordenada
      - ✓ Genera el menor hueco posible
    - Peor hueco (“Worst Fit”):
      - ✓ Asignar el mayor hueco lo suficientemente grande
      - ✓ Requiere buscar en toda la lista de huecos o tener lista ordenada
      - ✓ Genera el mayor hueco posible

Gestión de memoria mediante particiones

- **Particiones dinámicas (cont.):**
  - Problemas:
    - **Fragmentación interna**: Memoria no usada pero asignada a los procesos para evitar particiones libres demasiado pequeñas
    - **Fragmentación externa**: Hay suficiente espacio libre en memoria para satisfacer una solicitud, pero el espacio no es contiguo



Si hay una petición de 300 KB no sería posible.

Gestión de memoria mediante particiones

- **Particiones dinámicas (cont.):**
  - **Solución a fragmentación externa: Compactación.**
    - Desplazamiento del contenido de memoria hasta tener un único hueco con la memoria libre
    - Problemas:
      - ✓ Operación costosa
      - ✓ Estrategia complicada
    - Sólo posible con reubicación dinámica en tiempo de ejecución

Gestión de memoria mediante particiones

- **Particiones dinámicas (cont.):**
  - **Otras soluciones a la fragmentación externa:**
    - Permitir que la memoria de un proceso no esté en posiciones contiguas de memoria sino asignarle memoria física dondequiera que ésta esté disponible
      - ✓ Paginación
      - ✓ Segmentación
      - ✓ Segmentación paginada

UNIVERSITAT JAUME·I

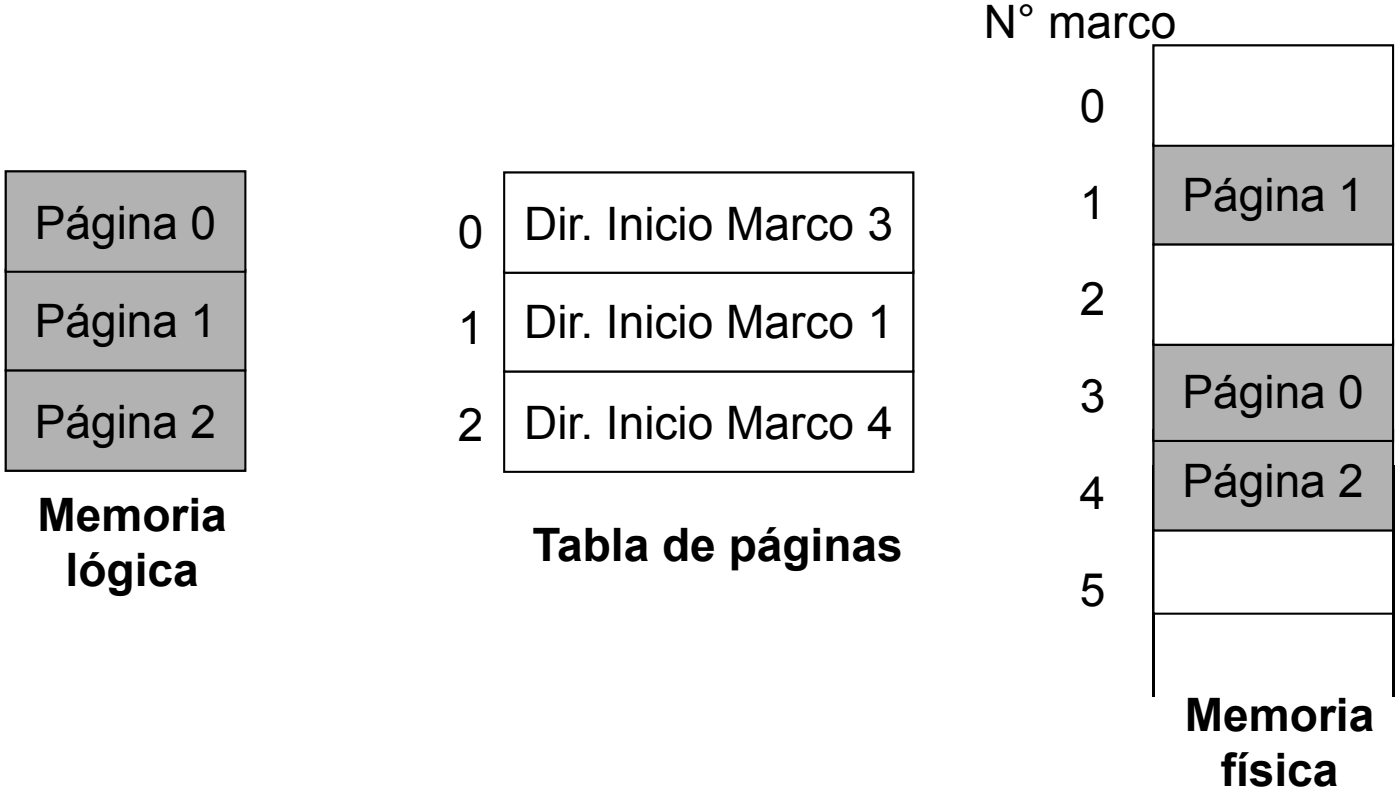
Índice

- Introducción
- Mapa de memoria de un proceso
- Gestión de la memoria principal
  - ✓ Particiones
  - ✓ Paginación
  - ✓ Segmentación
  - ✓ Segmentación paginada
- Gestión de la memoria virtual

Paginación

- División de la memoria física en bloques de tamaño fijo llamados **marcos**

  División de la memoria lógica en bloques del mismo tamaño que los marcos llamados **páginas**



Paginación

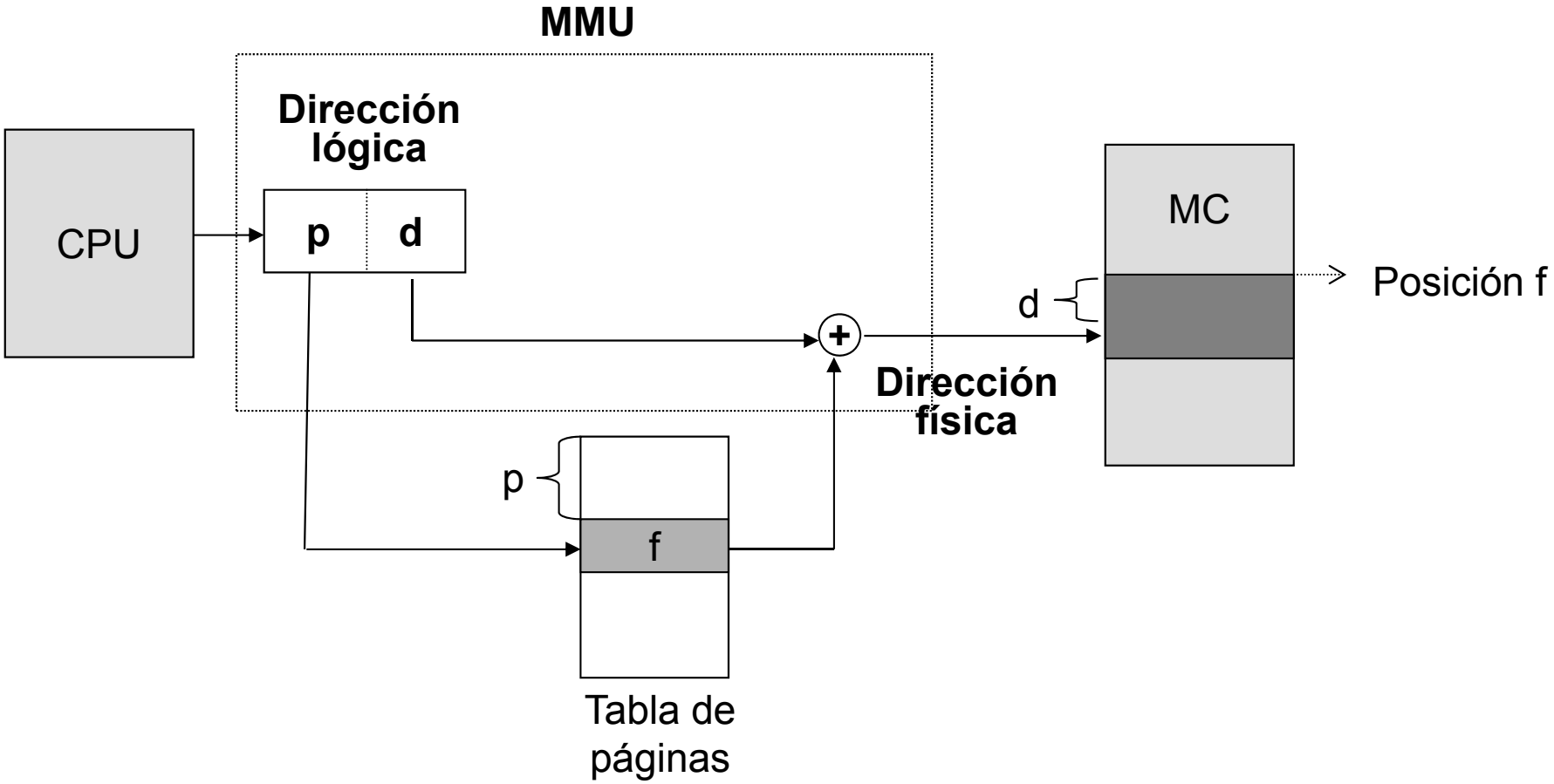
- **Transformación de direcciones lógicas en físicas:**
  - **Tabla de descriptores de páginas**:
    - Un descriptor por página del proceso
    - Cada descriptor contiene la dirección de inicio en MC del marco donde se ha cargado la página
- **Formato dirección lógica:** (p,d) donde:
  - p ≡ Número de página
    - Índice de la tabla de páginas.
    - dirección_lógica DIV tamaño_página
  - d ≡ Desplazamiento dentro de la página
    - dirección_lógica MOD tamaño_página

Paginación

- El SO también utiliza una **tabla de descriptores de marcos**:
  - Una entrada por marco físico
  - Cada entrada indica:
    - Si el marco está libre
    - Si el marco está asignado: página y proceso a la que se asignó
- Cuando se crea un proceso, el SO busca en MC tantos marcos libres como páginas ocupa el proceso
  - Para cada página:
    - Asignar un marco libre
    - Almacenar en la tabla de páginas del proceso la información necesaria sobre el marco asignado

Paginación

- **Mecanismo de traducción de direcciones:**
  - Mecanismo I:

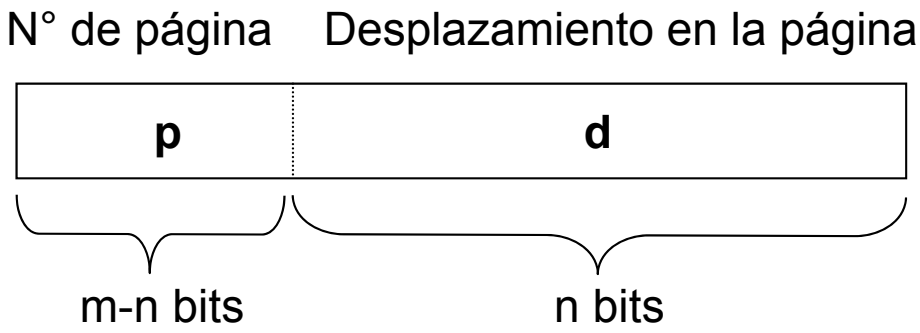


Paginación

- **Mecanismo de traducción de direcciones (cont.):**
  - Mecanismo I:
    - Operación de suma costosa
  - Mecanismo II:
    - Eliminación de la suma en el cálculo de la dirección física
    - Tamaño página:
      - ✓ 2^n unidades de direccionamiento (bytes o palabras)

Paginación

- **Mecanismo de traducción de direcciones (cont.):**
  - Mecanismo II (cont.):
    - Formato dirección lógica:
      - ✓ Tamaño del espacio de direcciones lógicas: 2^m



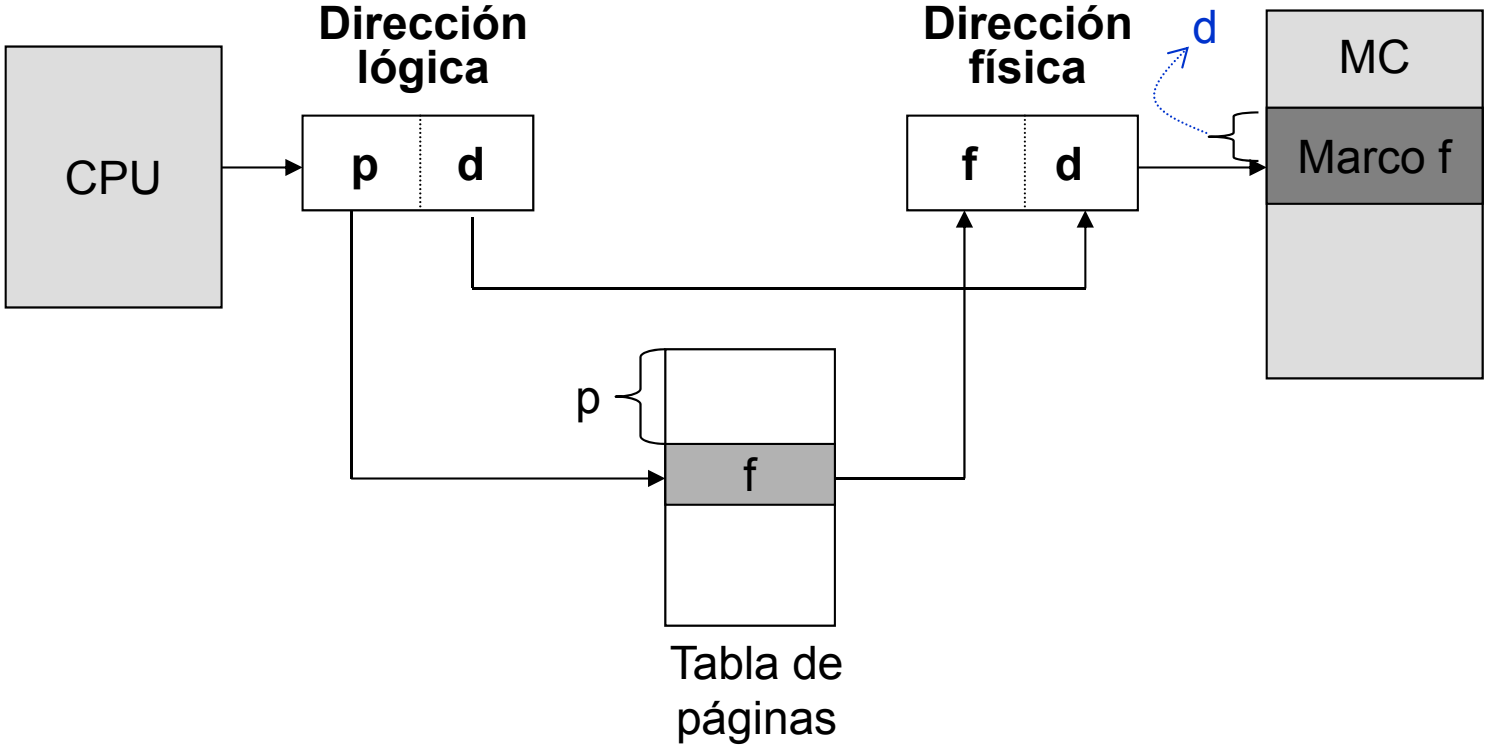
$|página| = 2^n$ bytes

    - Tabla de descriptores de páginas:
      - ✓ Cada descriptor contiene el número de marco asociado a la correspondiente página.

Paginación

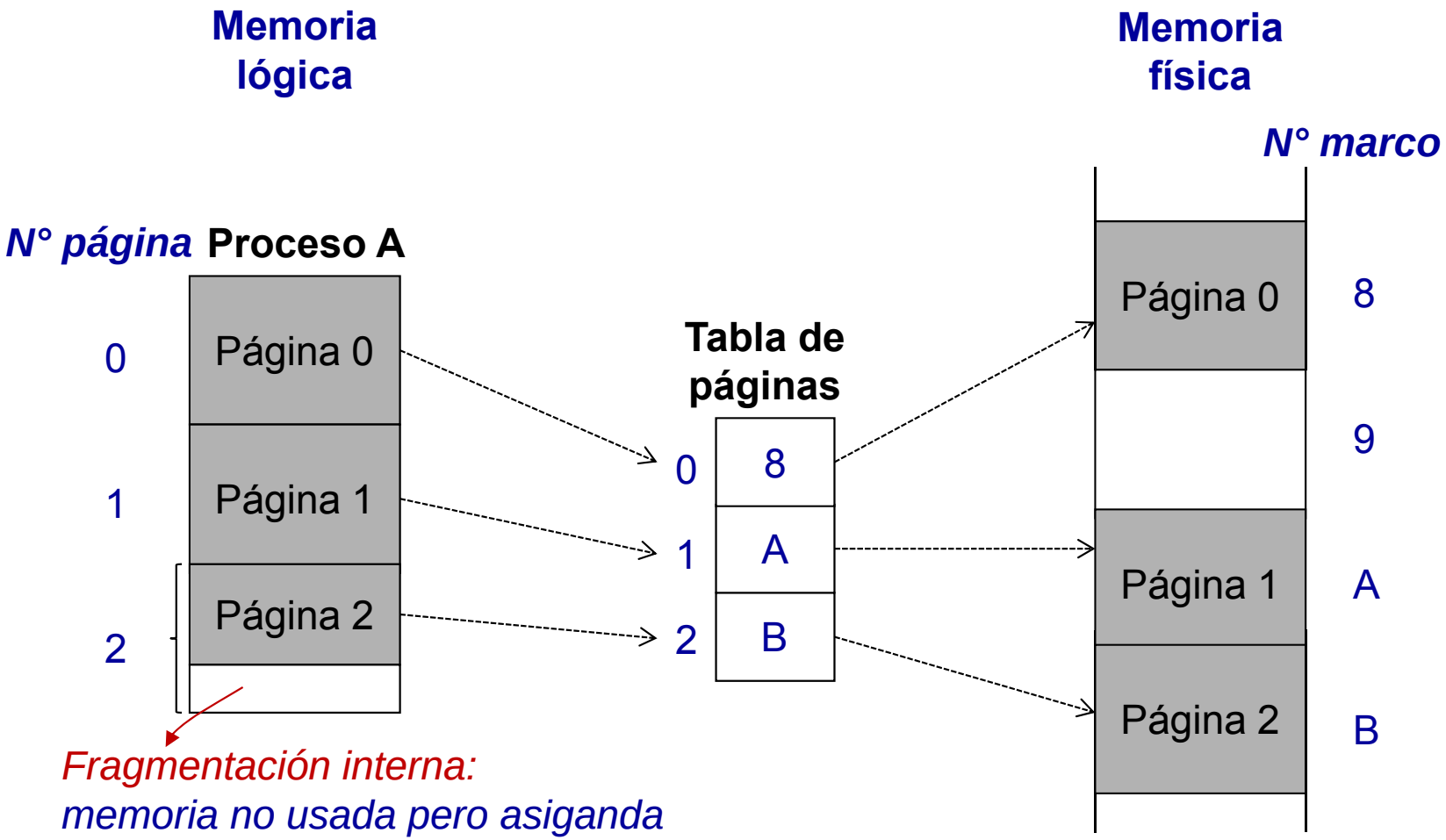
- **Mecanismo de traducción de direcciones (cont.):**
  - Mecanismo II (cont.):

|página| = |marco| = 2^n bytes



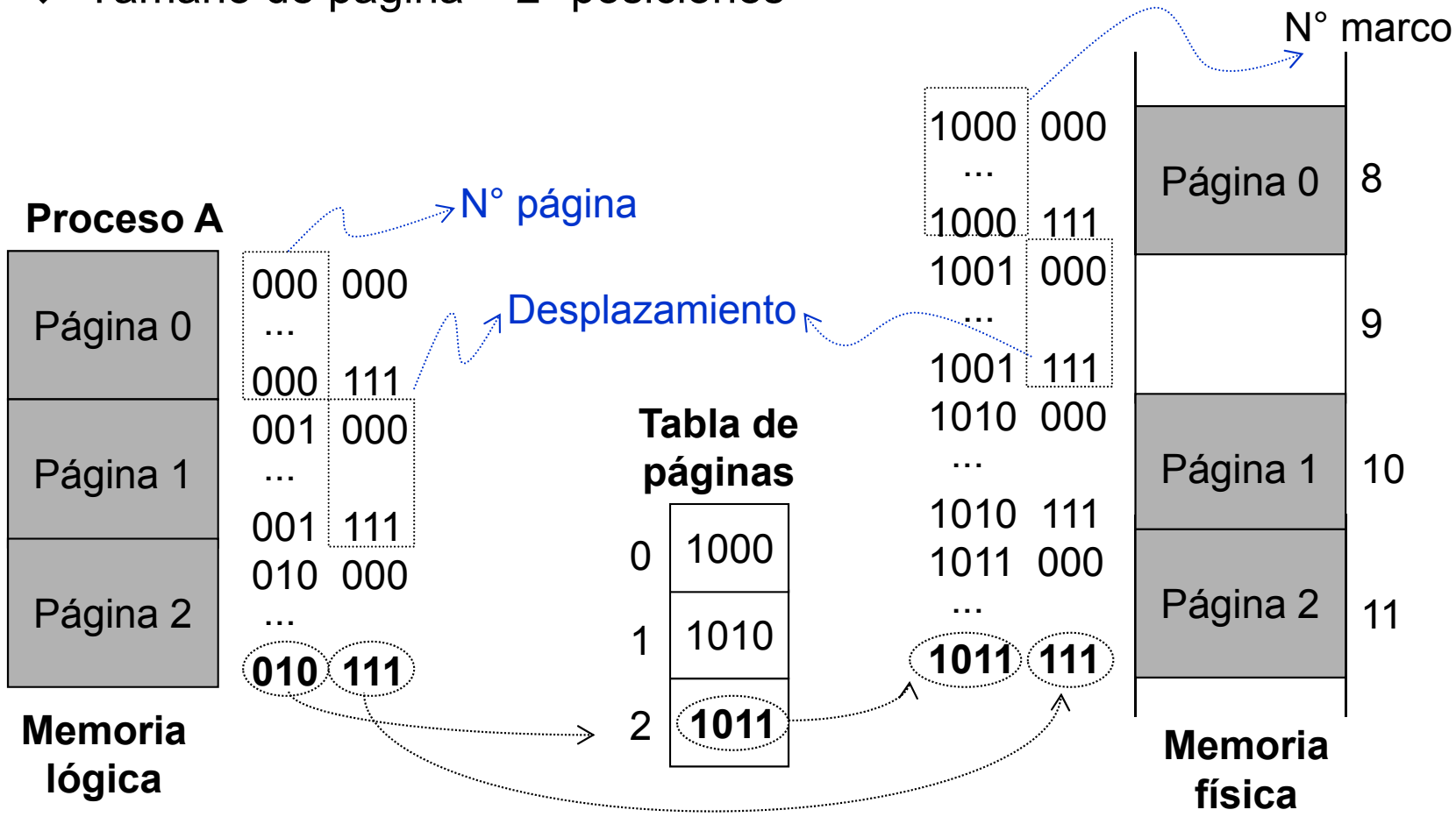
Paginación

- **Mecanismo de traducción de direcciones (cont.):**



Paginación

- **Mecanismo de traducción de direcciones (cont.):**
  - Tamaño de página = 2^3 posiciones



Paginación

- **Tamaño de la página:**
  - Condicionado por diversos factores contrapuestos:
    - Potencia de 2
    - Mejor pequeño por:
      - ✓ Menor fragmentación
    - Mejor grande por:
      - ✓ Tablas de páginas y marcos más pequeñas
    - Compromiso: Entre 2KB y 16KB

Paginación

- **Implementación de la tabla de descriptores de páginas:**
  - La tabla de páginas se almacena en MC
  - **Implementación con registros:**
    - Registro base de la tabla de páginas: Dirección de inicio de la tabla de páginas

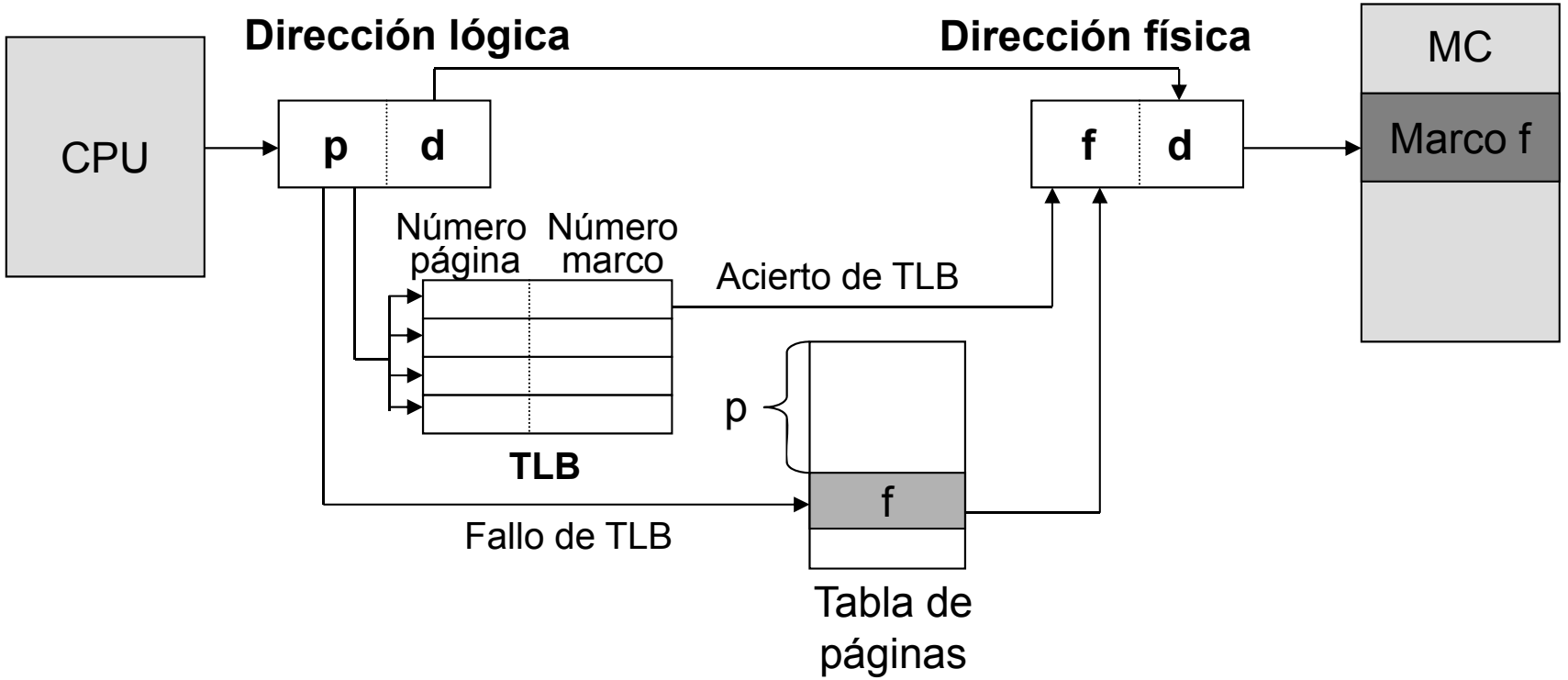
      Registro de longitud de la tabla de páginas: Tamaño de la tabla de páginas
    - Cada acceso a un dato o a una instrucción requiere dos accesos a MC:
      - ✓ Uno a la tabla de páginas
      - ✓ Otro al dato o instrucción

Paginación

- **Implementación de la tabla de descriptores de páginas (cont.):**
  - **Implementación con memoria caché:**
    - Utilización de HW especial (registros asociativos) más rápido que la MC:
      - ✓ Implementación de una **Tabla de asignación rápida** (“Translation look-aside buffer - TLB”)
        - Cada registro contiene una clave y un valor (n° página y n° marco)
    - La TLB contiene sólo unas cuantas entradas de la tabla de páginas.
      - ✓ N° entradas de la TLB: 8..2048
    - Búsqueda de un par (n° página, n° marco) en paralelo en toda la TLB:
      - ✓ Búsqueda rápida pero HW costoso

Paginación

- **Implementación de la tabla de descriptores de páginas (cont.):**
  - **Implementación con memoria caché (cont.):**
    - HW de paginación con TLB:



Paginación

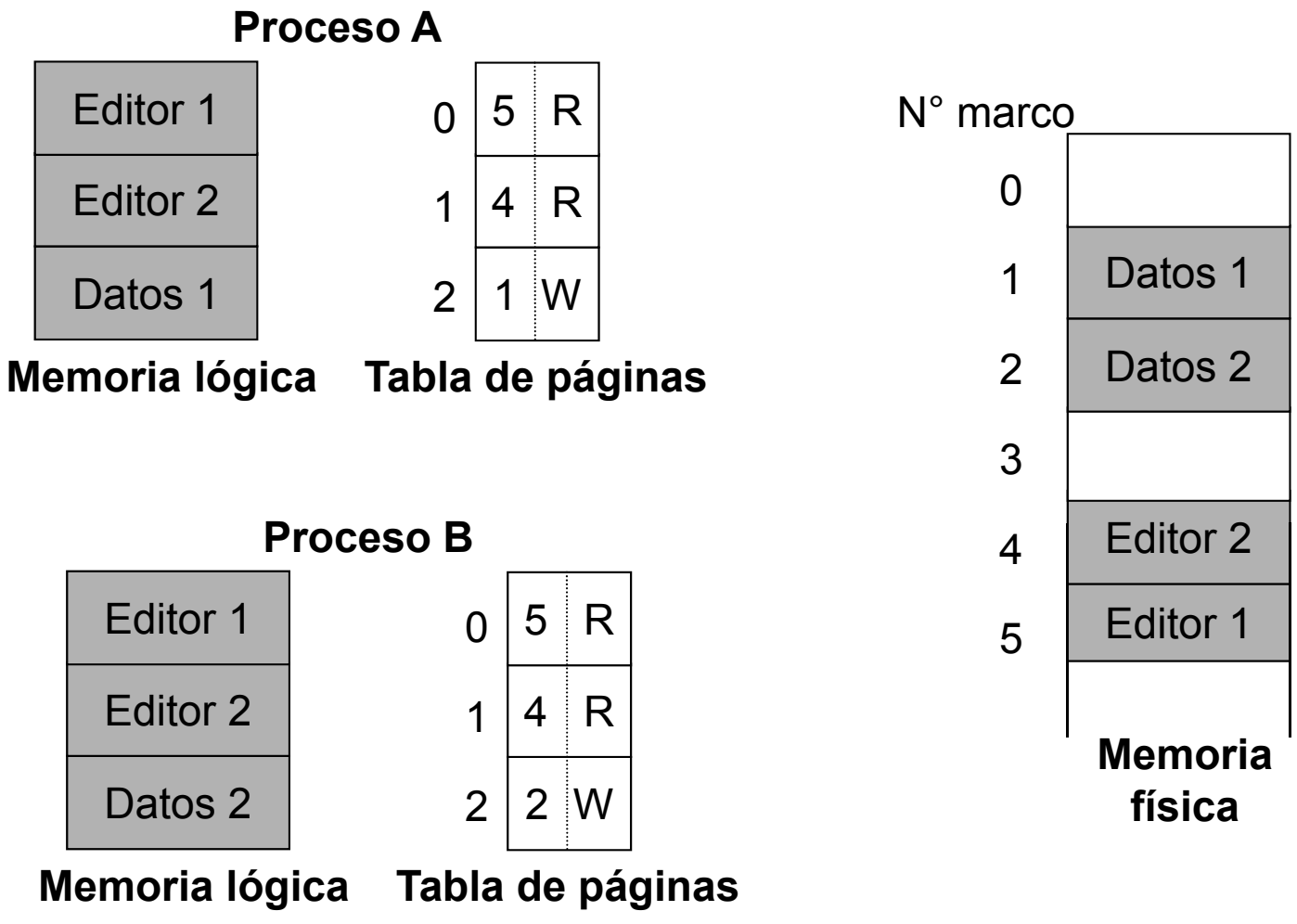
- **Implementación de la tabla de descriptores de páginas (cont.):**
  - **Implementación con memoria caché (cont.):**
    - HW de paginación con TLB (cont.):
      - ✓ Si el número de página está en la TLB, obtener el número de marco
      - ✓ Si no:
        - Acceder a la tabla de páginas en memoria
        - Obtener el número de marco correspondiente.
        - Añadir en la TLB una entrada con la asociación entre el número de página y el número de marco
        - Si la TLB estuviera llena, sustituir un registro

Paginación

- **Compartición de páginas:**
  - Inclusión en la tabla de páginas de:
    - Bits de protección de cada marco (lectura, escritura, ejecución)
  - Inclusión en la tabla de marcos un contador de procesos que comparten cada página:
    - Un marco sólo es liberado de MC cuando no sea utilizado por ningún proceso

Paginación

- **Compartición de páginas (cont.):**



Paginación

- **Problemas de la paginación:**
  - Aparece **fragmentación interna** en el último marco de un proceso
  - Visión de la memoria como una formación lineal de bytes, diferente a la visión del usuario

UNIVERSITAT JAUME·I

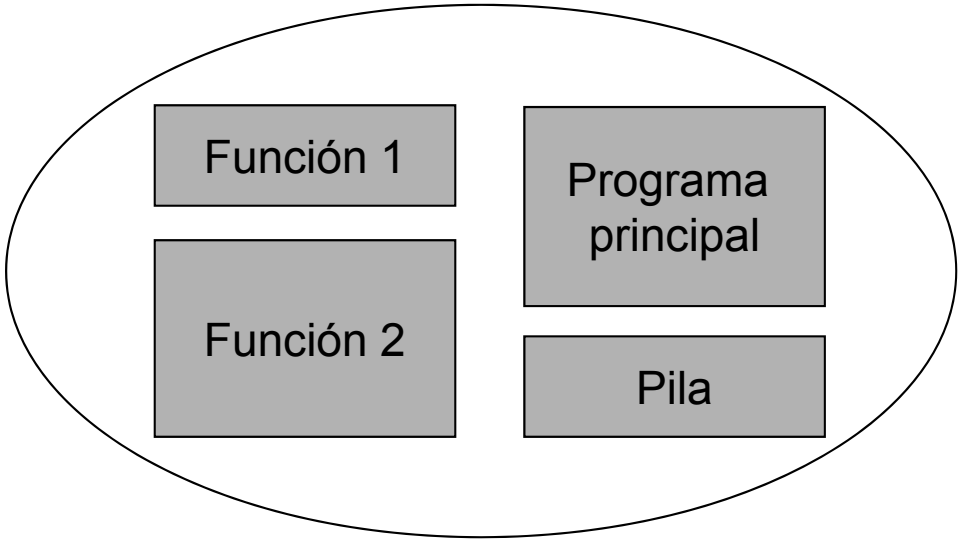
Índice

- Introducción
- Mapa de memoria de un proceso
- Gestión de la memoria principal
  - ✓ Particiones
  - ✓ Paginación
  - ✓ Segmentación
  - ✓ Segmentación paginada
- Gestión de la memoria virtual

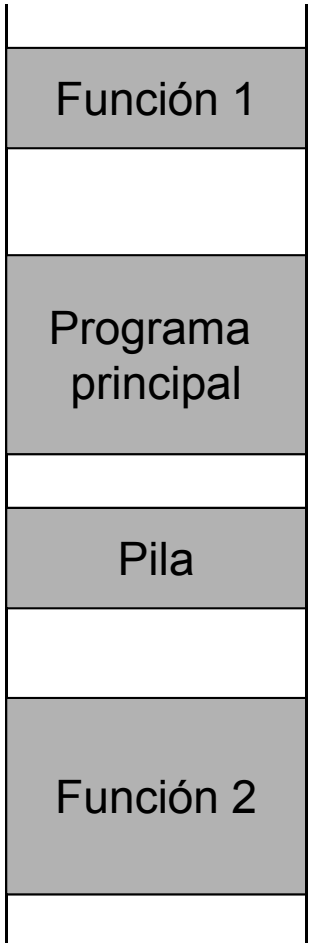
Segmentación

- Esquema de gestión de memoria que comparte la visión de la memoria que posee el usuario:
  - Un programa es un conjunto de módulos -**segmentos**- de tamaño variable
  - Cada segmento es una unidad lógica:
    - Programa principal
    - Función
    - Variables
    - Pila
    - Vectores
    - etc.

Segmentación



Espacio de direcciones lógico



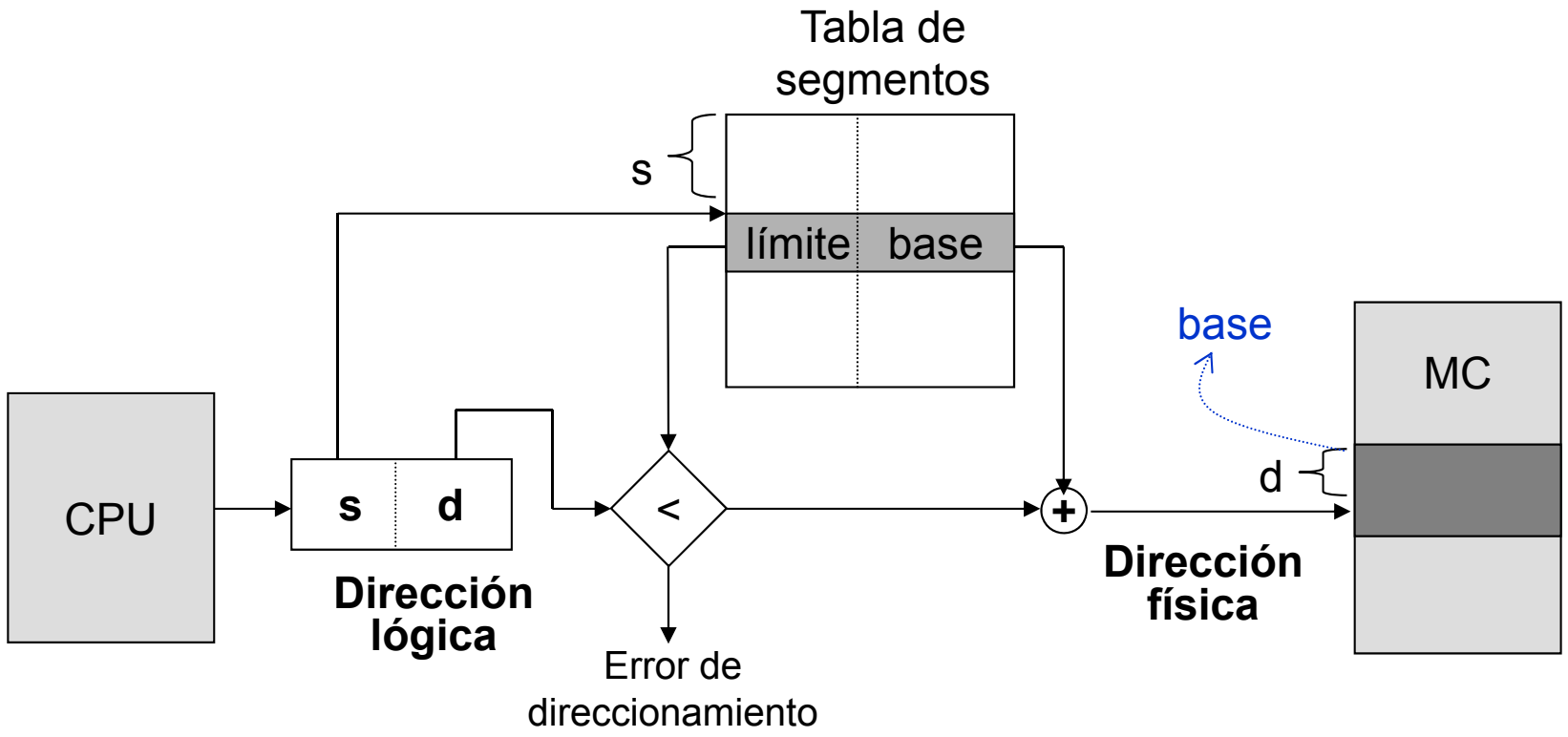
Memoria física

Segmentación

- **Transformación de direcciones lógicas en físicas:**
  - **Tabla de descriptores de segmentos**:
    - Un descriptor por segmento del proceso
    - Cada descriptor contiene:
      - ✓ Base del segmento: Dirección de inicio en MC donde se ha cargado el segmento
      - ✓ Límite del segmento: Longitud del segmento
- **Formato dirección lógica:** (s,d) donde:
  - s ≡ Número de segmento
    - Índice de la tabla de segmentos
  - d ≡ Desplazamiento dentro del segmento

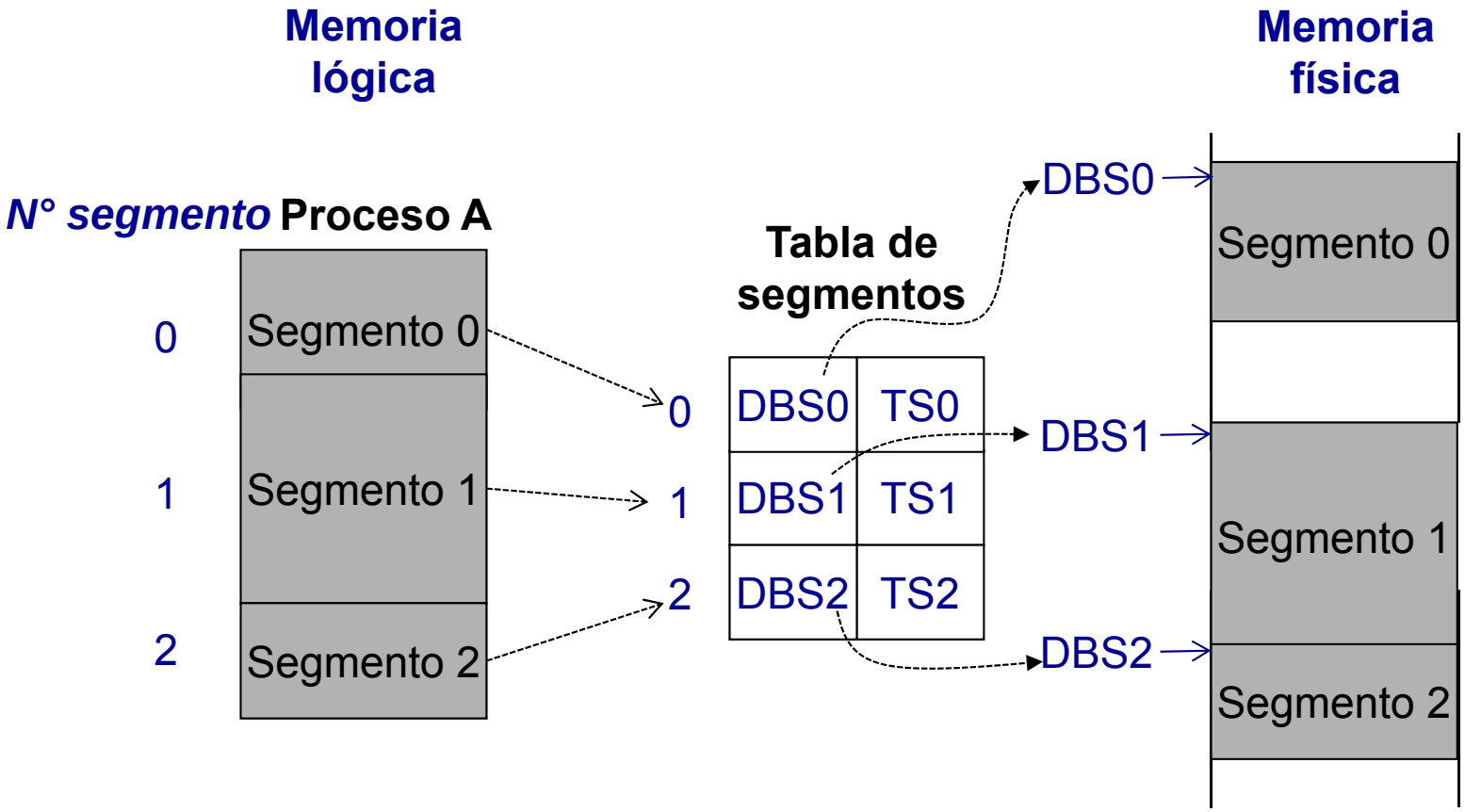
Segmentación

- **Mecanismo de traducción de direcciones:**



Segmentación

- **Mecanismo de traducción de direcciones (cont.):**



DBS0: Dirección base (física) segmento 0
TS0: Tamaño segmento 0

Segmentación

- **Implementación de la tabla de descriptores de segmentos:**
  - **Implementación con registros:**
    - Registro base de la tabla de segmentos: Dirección de inicio de la tabla.
      Registro de longitud de la tabla de segmentos: Tamaño de la tabla
    - Cada acceso a un dato o a una instrucción requiere dos accesos a MC:
      - ✓ Uno a la tabla de segmentos
      - ✓ Otro al dato o instrucción
  - **Implementación con memoria caché:**
    - Conjunto de registros asociativos con las entradas de la tabla de segmentos usadas más recientemente
    - Reducción del tiempo de acceso a memoria

Segmentación

- **Compartición de código y datos:**
  - Compartición a nivel de segmento
  - Los segmentos se comparten cuando entradas de la tabla de segmentos de dos procesos apuntan a las mismas posiciones físicas
  - Compartición de segmentos de código → Todos los procesos que lo comparten deberán definirlo con el mismo número
  - Inclusión en la tabla de segmentos de:
    - Bits de protección de cada segmento (lectura, escritura, ejecución)

Segmentación

- **Problemas de la segmentación:**
  - Fragmentación externa
  - Gestión compleja de la memoria física

Tema 6. Gestión de memoria

UNIVERSITAT JAUME·I

Índice

- Introducción
- Mapa de memoria de un proceso
- Gestión de la memoria principal
  - ✓ Particiones
  - ✓ Paginación
  - ✓ Segmentación
  - ✓ Segmentación paginada
- Gestión de la memoria virtual

Segmentación paginada

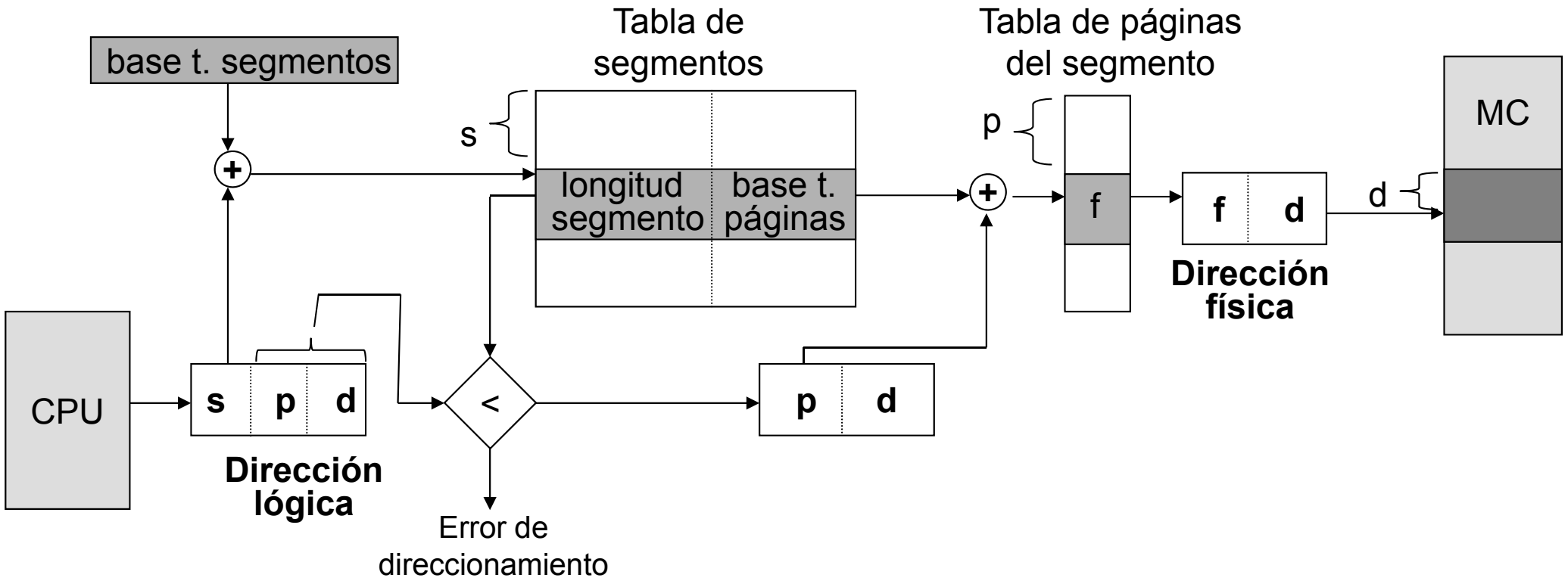
- Recoge las ventajas de los dos esquemas anteriores:
  - Visión de la memoria como la ve el usuario
  - Facilidad de gestión del SO
  - No fragmentación externa
- Paginación de los segmentos:
  - Una tabla de páginas por segmento
- Tabla de segmentos:
  - Dirección base de la tabla de páginas para el segmento
  - Longitud del segmento

Segmentación paginada

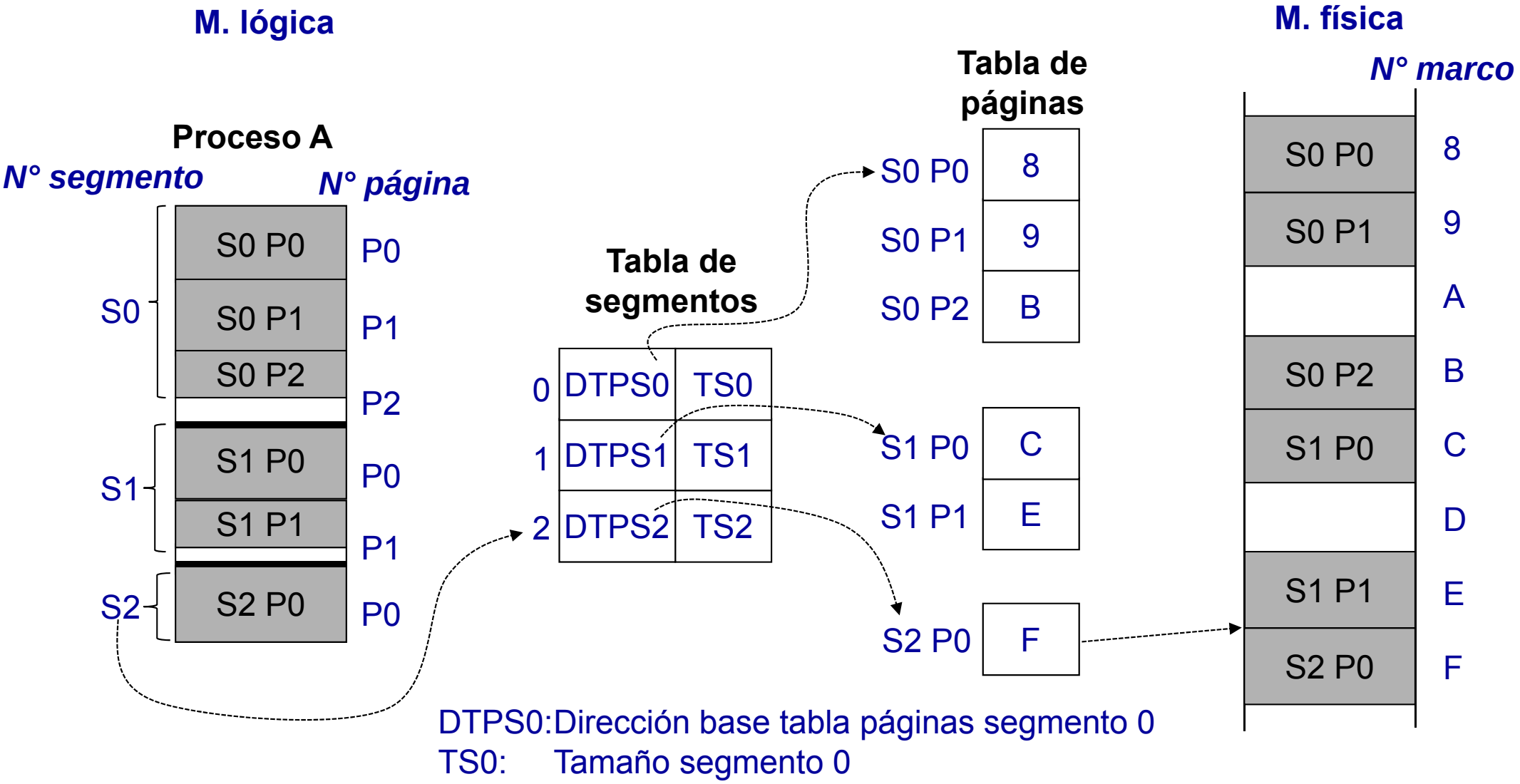
- **Formato dirección lógica:** (s,p,d) donde:
  - s ≡ Número de segmento
    - Índice de la tabla de segmentos
  - p ≡ Número de página dentro del segmento
  - d ≡ Desplazamiento dentro de la página

Segmentación paginada

- **Mecanismo de traducción de direcciones:**



Segmentación paginada



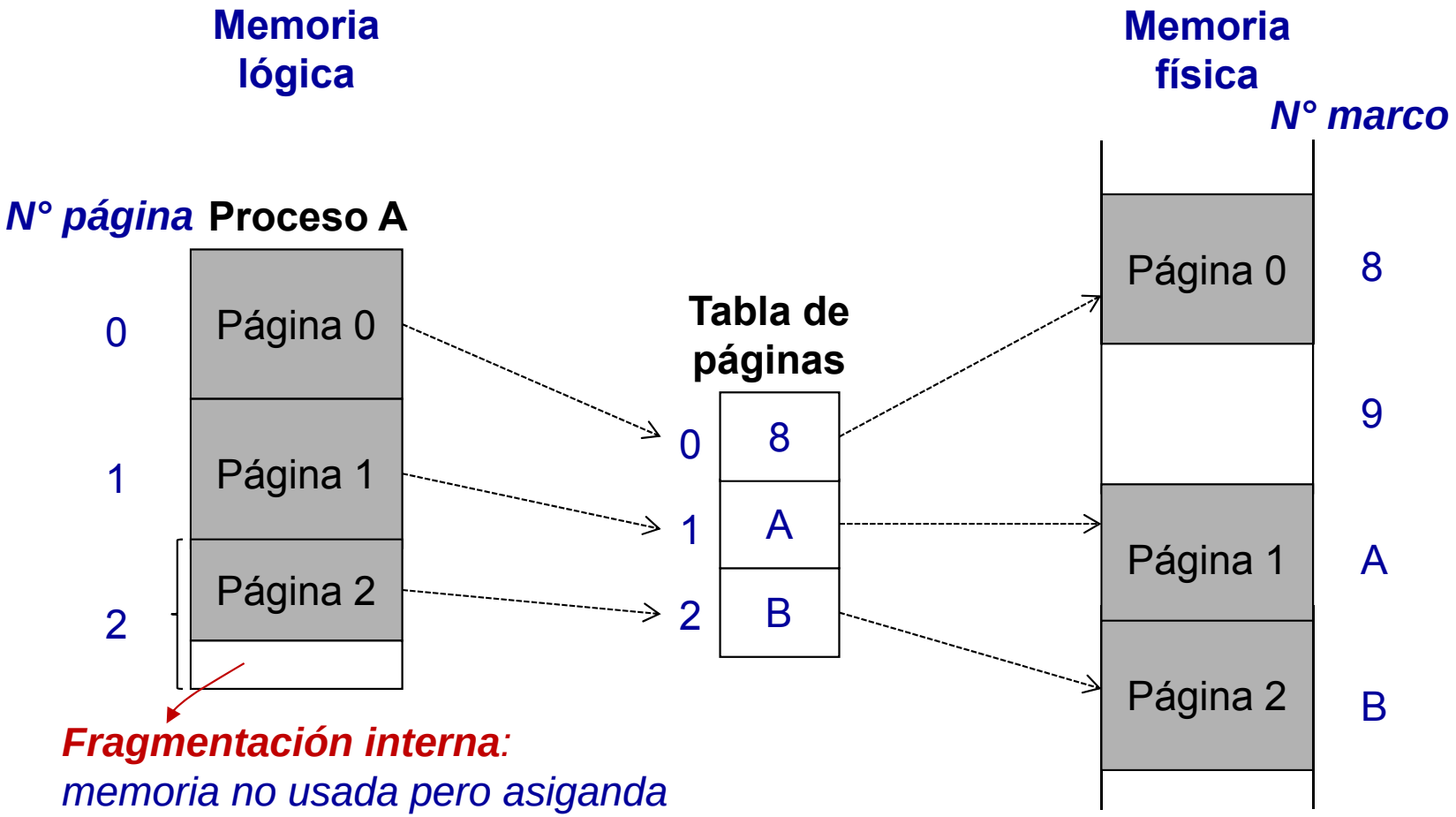
DTPS0:Dirección base tabla páginas segmento 0
TS0: Tamaño segmento 0

Fragmentación interna

- **Paginación:**
  - Solo genera fragmentación interna la última página
- **Segmentación:**
  - No hay fragmentación interna
- **Segmentación paginada:**
  - Solo genera fragmentación interna la última página de cada segmento

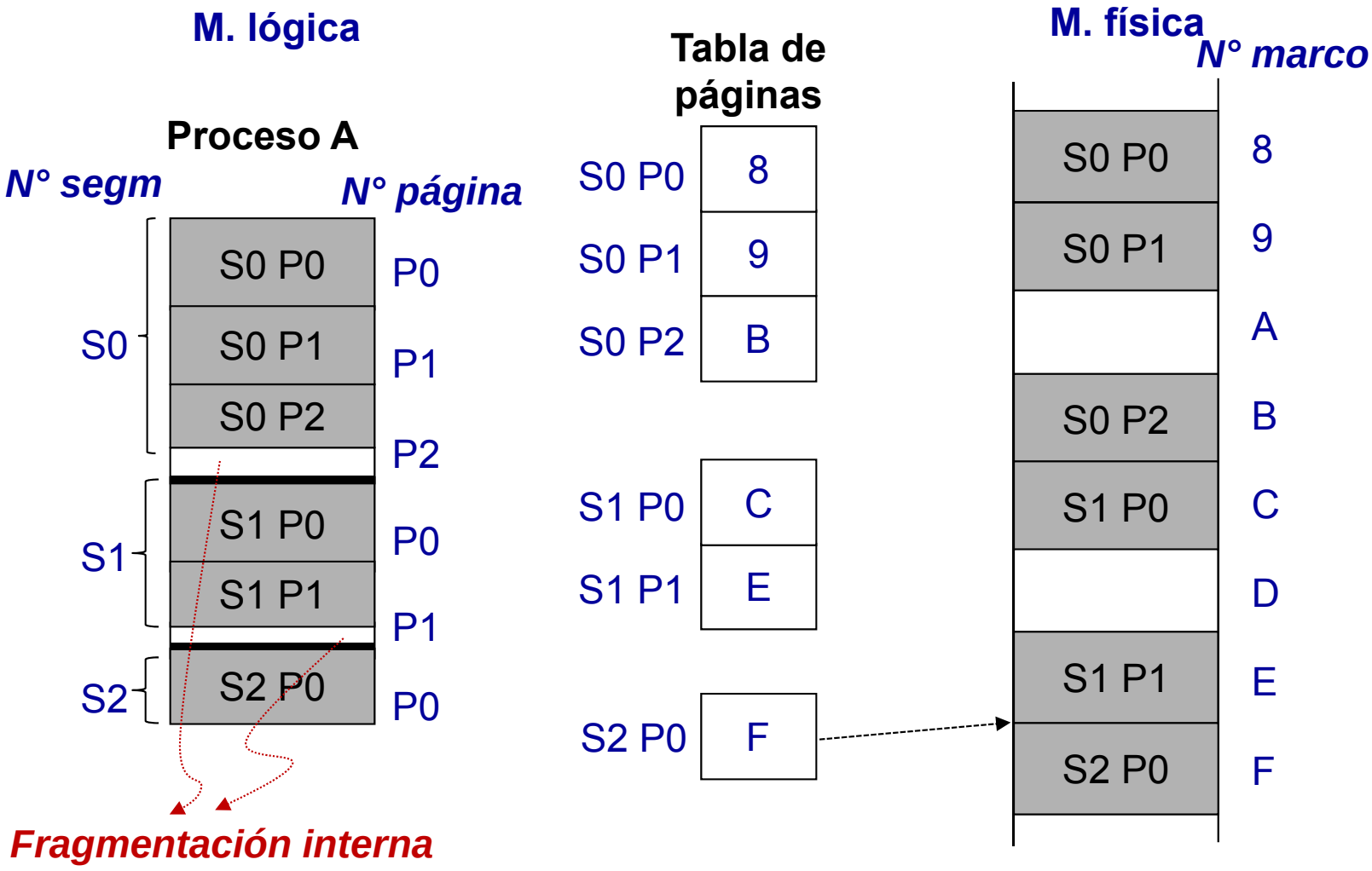
Fragmentación interna

- **Paginación**



Fragmentación interna

- **Segmentación paginada**



*Solo genera **fragmentación interna** la última página de cada segmento*

Tema 6. Gestión de memoria

UNIVERSITAT JAUME·I

Índice

- Introducción
- Mapa de memoria de un proceso
- Gestión de la memoria principal
- Gestión de la memoria virtual
  - ✓ Motivación
  - ✓ Paginación bajo demanda
  - ✓ Paginación multinivel
  - ✓ Algoritmos de reemplazo de página
  - ✓ Asignación de memoria central a procesos
  - ✓ Hiperpaginación

Motivación

- **Memoria virtual:**
  - Separación de la memoria lógica y de la memoria física
    - Sólo es necesario tener en memoria central parte del espacio de direccionamiento lógico de cada proceso
      - ✓ Principio de localidad de referencias
    - El espacio de direccionamiento lógico puede ser mayor que el espacio de direccionamiento físico disponible
    - Los procesos pueden ser más grandes que la memoria física
    - Utilización de parte de un dispositivo de almacenamiento masivo como **dispositivo de intercambio**
    - Se debe gestionar el trasiego de parte del espacio de direccionamiento lógico entre la memoria central y el dispositivo de intercambio

Memoria virtual

- **La memoria virtual se implementa vía:**
  - Paginación bajo demanda

Paginación bajo demanda

- Cada página se lleva a memoria central sólo cuando es necesario
  - Menor número de operaciones de E/S para crear un proceso
  - Menor necesidad de MC por proceso
  - Respuesta más rápida
  - Posibilidad de más usuarios
- ¿Cuándo se necesita una página en MC ? Cuando es referenciada
  - Referencia a una dirección no válida $\Rightarrow$ Terminación anormal
  - Referencia a una dirección válida que no está en memoria $\Rightarrow$ Llevar a MC la página que la contiene

Bit de presencia

- En cada descriptor de página se añade un **bit de presencia** (bit de validez):
  - ✓ 1 ⇒ página cargada en MC.
  - ✓ 0 ⇒ página no cargada en MC.
- Inicialmente todos los bits de presencia toman valor 0
- Se produce una situación de **fallo de página** cuando al intentar traducir una dirección lógica en una física el correspondiente bit de presencia tiene valor 0

# Marco	Bit de presencia
	1
	1
	1
	1
	0
…	
	0
	0

Tabla de páginas

Fallo de página

- Excepción que se produce cuando se intenta transformar una dirección lógica en la correspondiente dirección física y el bit de presencia tiene valor 0 ①
- **Rutina de servicio de fallo de página:**
  - Se obtiene el identificador de un marco de MC libre ②
  - Se copia la página referenciada en dicho marco ③
  - Se actualiza el descriptor de dicha página ④
  - Se reinicia la ejecución de la instrucción que ha generado el fallo de página

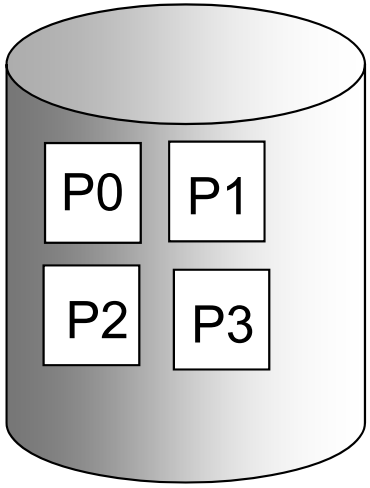
Fallo de página

Suponiendo:
Tamaño de página = 4096
Tamaño del proceso A = 14536

Dirección lógica = 0x0231F

Página 0x02

Desplazamiento 0x31F



Dispositivo de intercambio

0	0x2	0
1	0x2	1
2	0x4	0
3	0x4	1

1

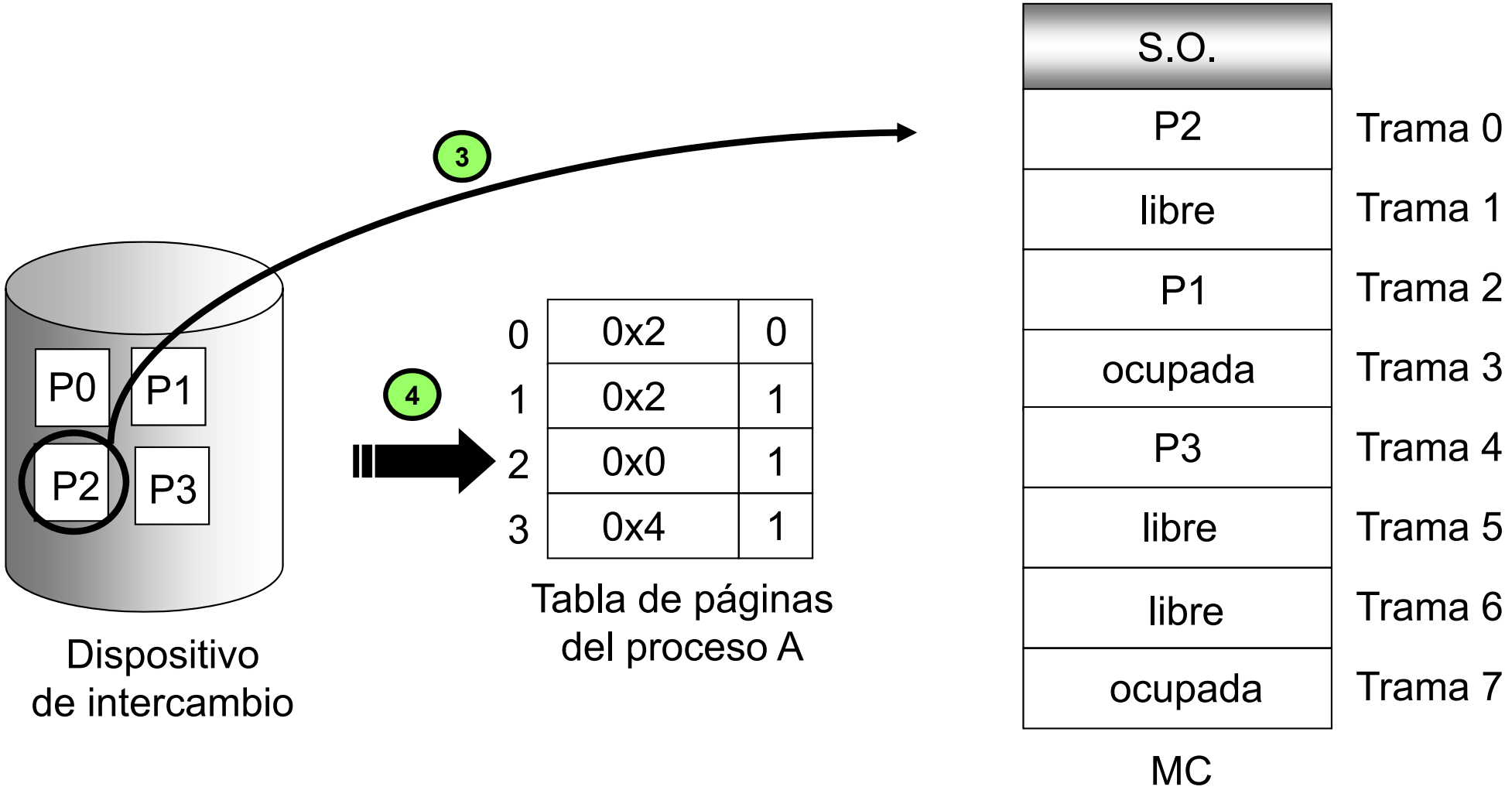
Tabla de páginas del proceso A

S.O.	
libre	Trama 0
libre	Trama 1
P1	Trama 2
ocupada	Trama 3
P3	Trama 4
libre	Trama 5
libre	Trama 6
ocupada	Trama 7

2

MC

Fallo de página



Fallo de página

- **¿Qué ocurre si no existe un marco de MC libre?**
  - Rutina de servicio de fallo de página:
    - Se obtiene el identificador de un marco de MC libre
    - Si no hay ningún marco libre, **reemplazar una página de MC**
    - Se copia la página referenciada en el marco seleccionado
    - Se actualiza el descriptor de dicha página
    - Se reinicia la ejecución de la instrucción que ha generado el fallo de página

Fallo de página

- **¿Qué ocurre si no existe un marco de MC libre (cont.)?**
  - **Reemplazo de página:**
    - Seleccionar una de las páginas que hay cargadas en MC y que no se esté utilizando y devolverla al dispositivo de intercambio (**página víctima**)
      - ✓ Algoritmo de selección de página víctima
      - ✓ Eficiencia: Utilización de un algoritmo que genere el menor número posible de fallos de página
    - Se producirá un trasiego de páginas entre MC y el dispositivo de intercambio
      - ✓ La misma página es llevada a memoria varias veces
      - ✓ Dos transferencias de páginas en un reemplazo: una desde MC y otra hacia MC

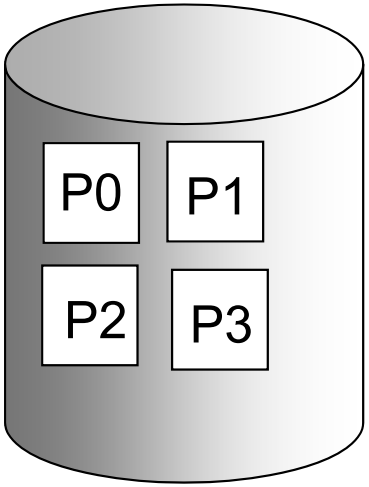
Fallo de página

Suponiendo:
Tamaño de página = 4096
Tamaño del proceso A = 14536

Dirección lógica = 0x0231F

Página 0x02

Desplazamiento 0x31F



Dispositivo
de intercambio

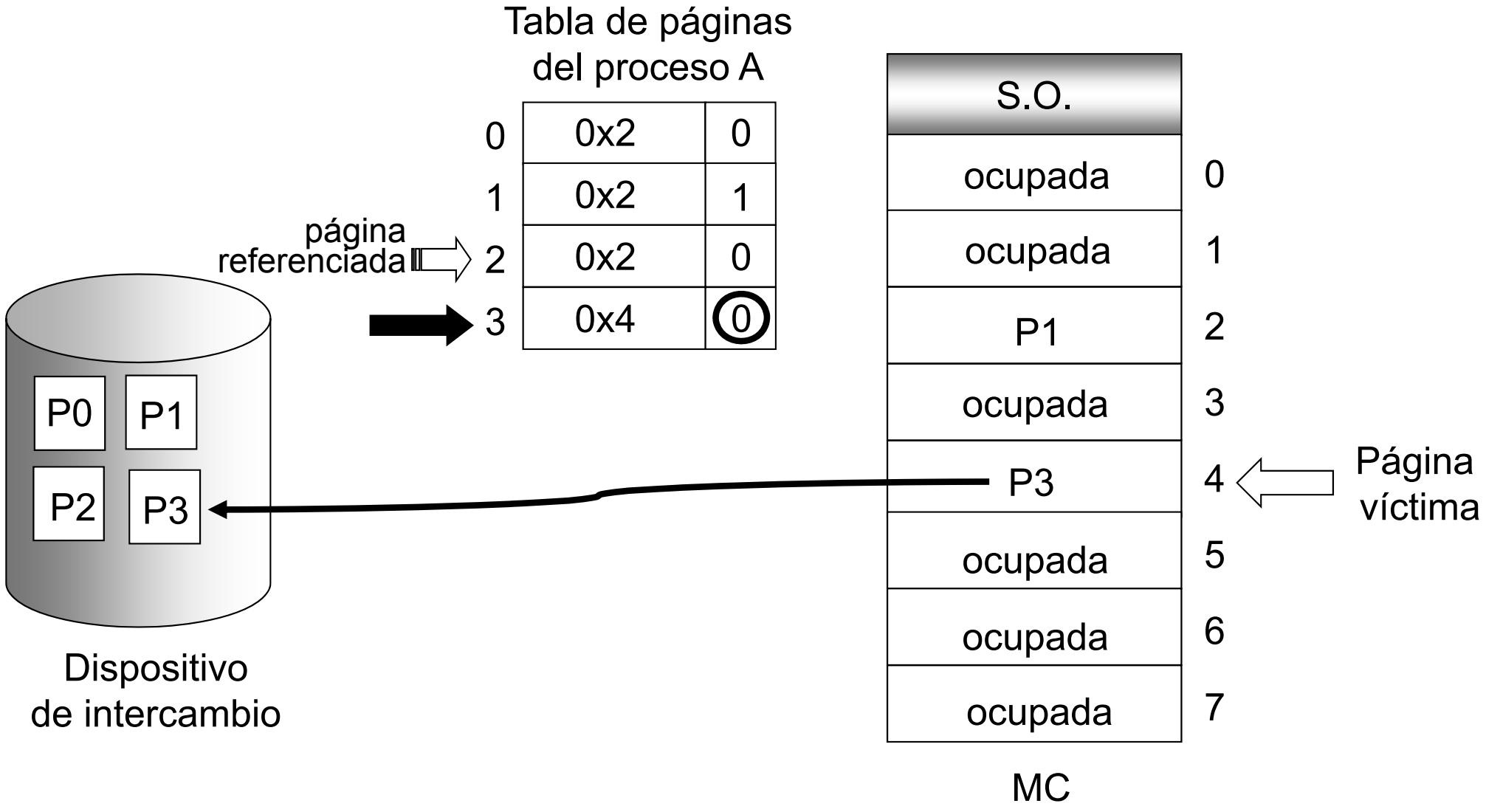
0	0x2	0
1	0x2	1
2	0x2	0
3	0x4	1

Tabla de páginas
del proceso A

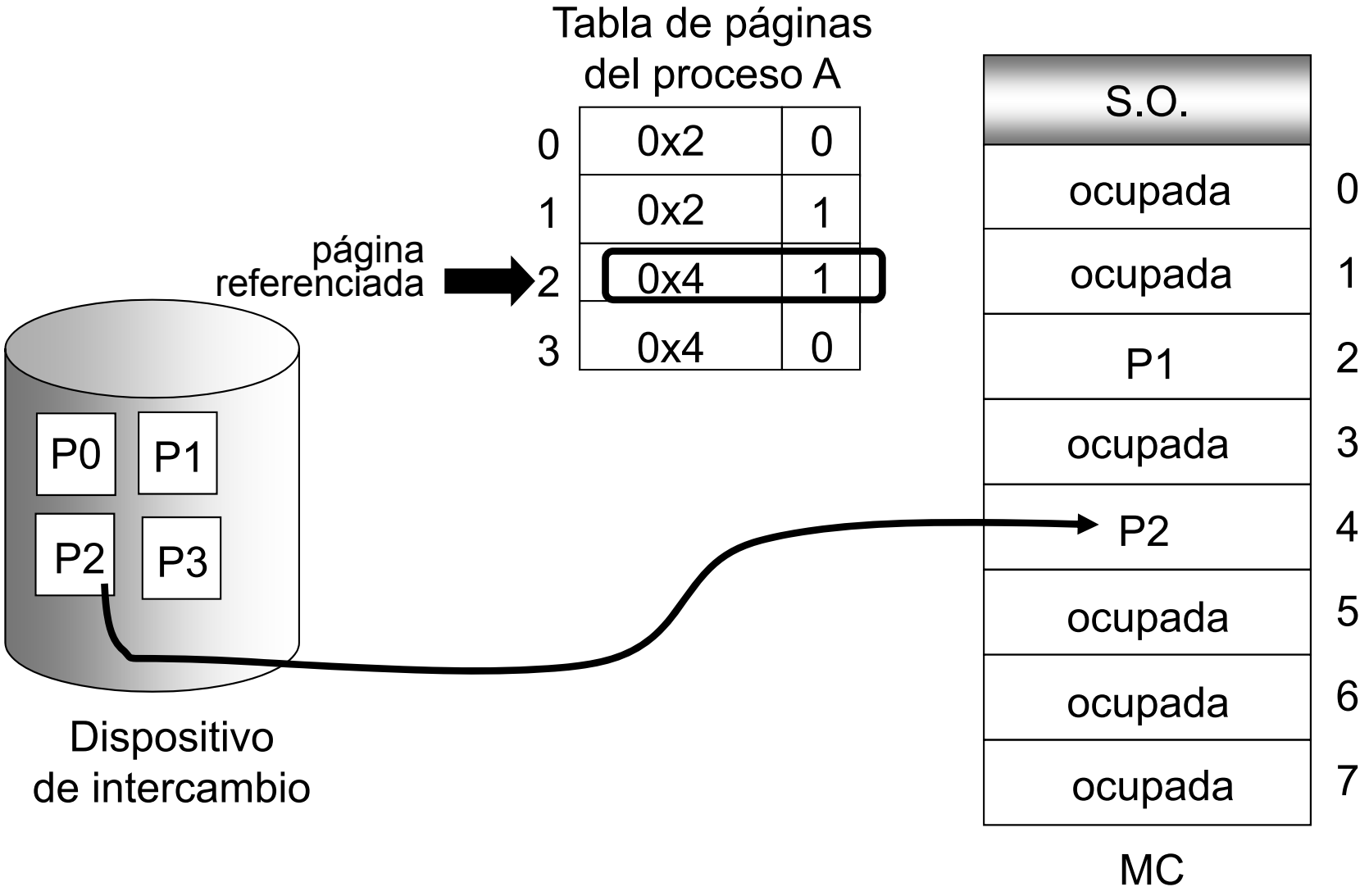
S.O.	
ocupada	Trama 0
ocupada	Trama 1
P1	Trama 2
ocupada	Trama 3
P3	Trama 4
ocupada	Trama 5
ocupada	Trama 6
ocupada	Trama 7

MC

Fallo de página



Fallo de página



Fallo de página

- **Reiniciación de la ejecución de una instrucción:**
  - Necesidad de HW especial que permita reiniciar la ejecución de la instrucción
  - Si el fallo de página ocurre al intentar obtener una instrucción → Reiniciar la obtención de la instrucción
  - Si el fallo de página ocurre al intentar obtener un operando → Volver a obtener la instrucción, decodificarla y obtener operando.
    - Posible problema con modificaciones ya efectuadas

Rendimiento

- **Frecuencia de fallo de página:**
  - p : Probabilidad de que una referencia a memoria provoque un fallo de página ($0 \leq p \leq 1$)
    - Si $p = 0 \Rightarrow$ Jamás se producen fallos de página
    - Si $p = 1 \Rightarrow$ Se produce fallo de página para todas las referencias

- **Tiempo efectivo de acceso a memoria (TEA):**

$$TEA = (1 - p) \times \textbf{tam} + p \times \textbf{tsfp}$$

donde:

tam = tiempo de acceso a MC
tsfp = tiempo de servicio de fallo de página

Rendimiento

- **Tiempo de servicio de fallo de página (tsfp):**
  - En él se contabiliza:
    - Tiempo necesario para la/las transferencia/s entre el dispositivo de intercambio y la MC
    - Tiempo necesario para la actualización de las tablas de página
    - Tiempo necesario para reiniciar la ejecución del proceso
  - El mayor coste se debe a los accesos al dispositivo de intercambio

Reducción del tsfp

⇓

Utilización de un **bit de modificado** (“dirty bit”).

Rendimiento

- **Bit de modificado:**
  - Bit añadido a cada uno de los descriptores de página
  - Sólo tiene significado cuando la página está en memoria
  - Cuando se carga la página en memoria se inicializa dicho bit a 0
  - En cada acceso en modo escritura se modifica dicho bit a 1
  - Cuando la página es elegida como página víctima, se reescribe en el dispositivo de intercambio sólo si el bit de modificado de su descriptor tiene valor 1

Rendimiento

- **Ejemplo de TEA:**
  - Tiempo de acceso a MC = 100 ηseg
  - 50% de veces que se elige una página víctima el bit de modificado se encuentra con valor 1
  - Tiempo necesario para transferir una página entre MC y el dispositivo de intercambio 24 mseg = 24E+6 ηseg

$$TEA = (1 - p) \times 100\ \eta seg +$$

$$+ p\ (\ 0.5 \times 24E+6\ \eta seg\ +\ 0.5 \times 2 \times 24E+6\ \eta seg\ ) =$$

$$= (1 - p) \times 100\ \eta seg + p \times 36E+6\ \eta seg =$$

$$= (\ 100 + p \times 35999900\ )\ \eta seg$$

Si se desea una degradación inferior al 10%:

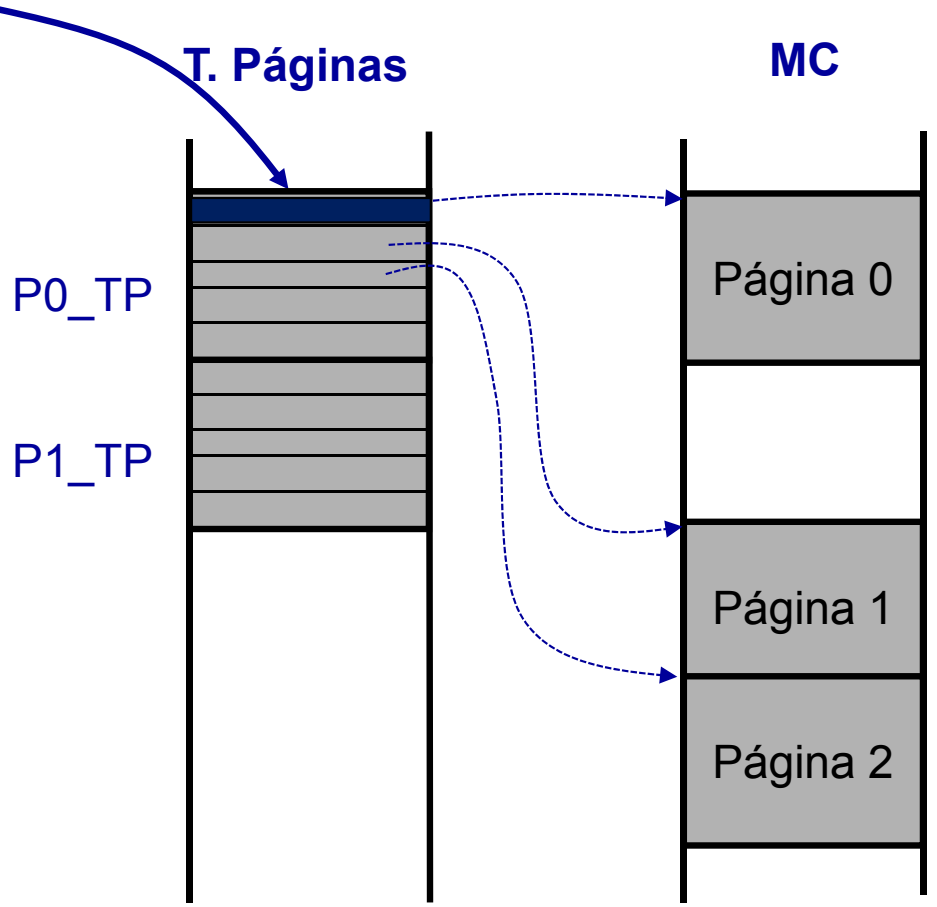
$$TEA \leq 1'10\ tam \Rightarrow p \leq 2'7E-7$$

es decir, se deben producir como mucho 1 fallo de página cada 3'7 millones de accesos a MC

Entradas de la tabla de páginas

¿Qué información hay en una entrada de la TP?

- Número de marco asociado a la página
- Bits de control (estado)
  - ✓ Validez
  - ✓ Modificado
  - ✓ R/W

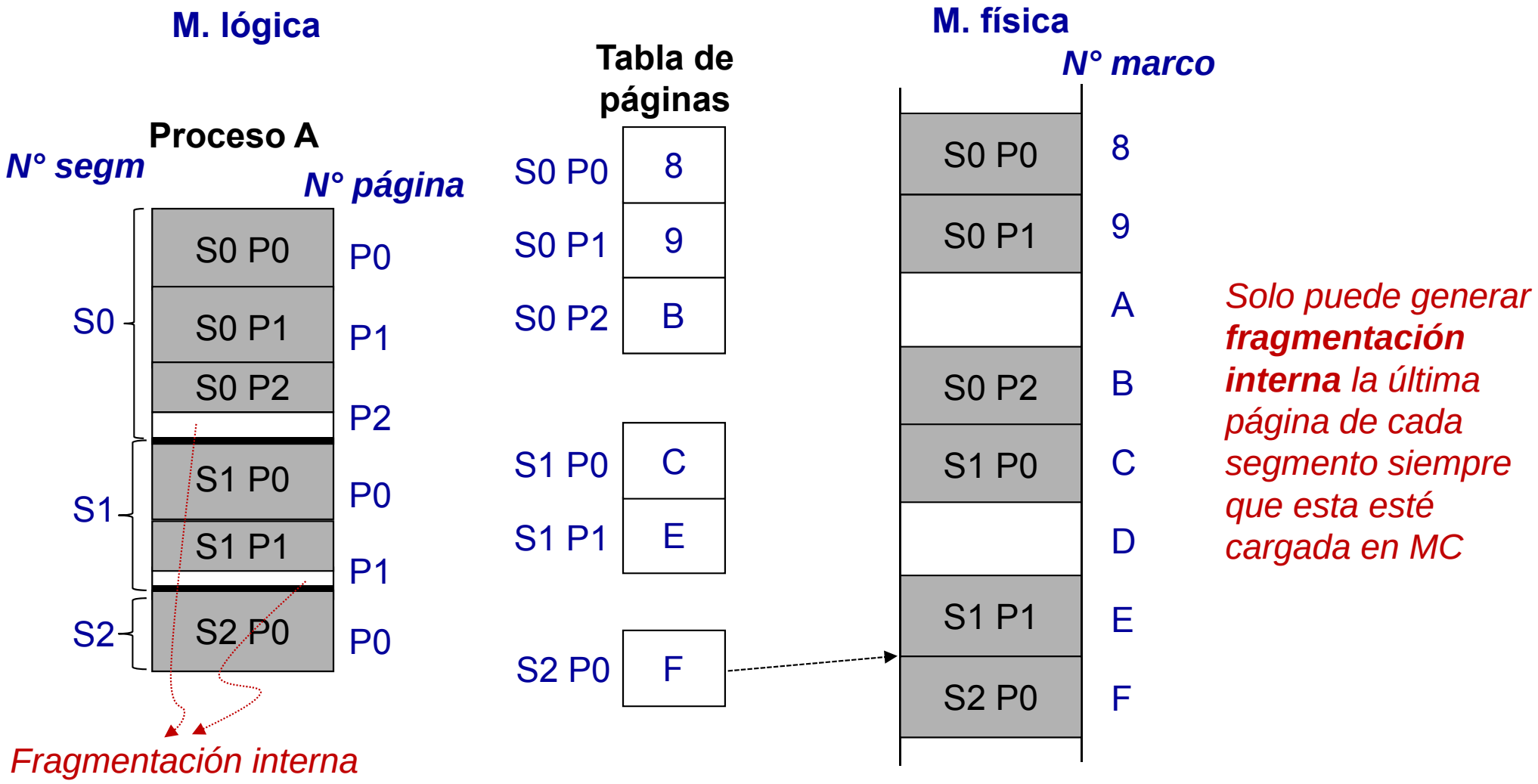


Fragmentación con memoria virutal

- Con **segmentación paginada**:
  - Solo puede generar fragmentación interna
    - la última página de cada segmento siempre que esta esté cargada en MC
  - **Fragmentación interna máxima**:
    - Espacio que se podría desaprovechar como máximo
    - Fragmentación generada si en MC estuviesen todas las últimas páginas de cada segmento
  - **Fragmentación interna generada en ese instante**:
    - Espacio desaprovechado realmente en ese instante
    - Fragmentación generada por las últimas páginas de segmento cargadas en MC en ese instante

Fragmentación con memoria virutal

- Segmentación paginada



*Solo puede generar **fragmentación interna** la última página de cada segmento siempre que esta esté cargada en MC*

UNIVERSITAT JAUME·I

Índice

- Introducción
- Mapa de memoria de un proceso
- Gestión de la memoria principal
- Gestión de la memoria virtual
  - ✓ Motivación
  - ✓ Paginación bajo demanda
  - ✓ Paginación multinivel
  - ✓ Algoritmos de reemplazo de página
  - ✓ Asignación de memoria central a procesos
  - ✓ Hiperpaginación

Paginación multinivel

- **Tabla de páginas con un único nivel:**
  - Ocupan gran espacio (varios MB cada tabla)
    - Gasto inadmisible
    - La mayoría de las páginas de la tabla de páginas de cada proceso estarán vacías
  - Las páginas de una tabla necesitarían estar:
    - En posiciones físicas consecutivas
    - En memoria a la vez

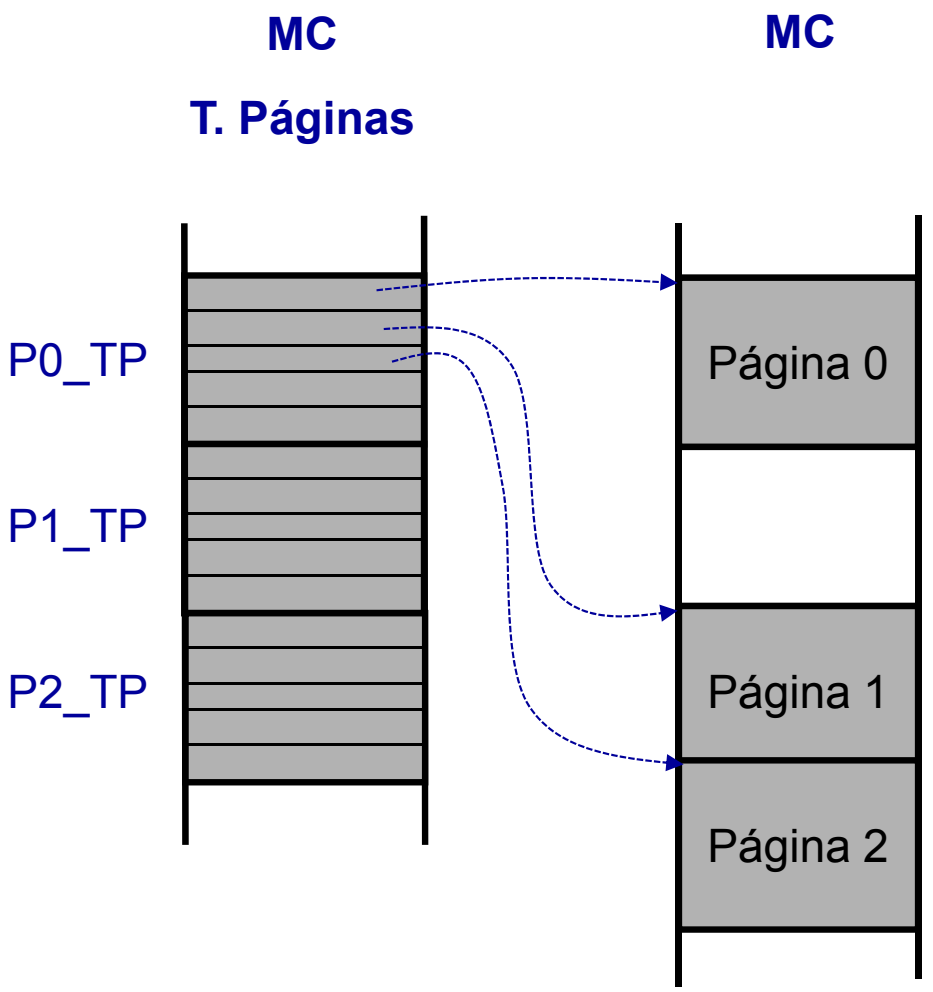
Paginación multinivel

- **Tabla de páginas multinivel:**
  - Paginación de la tabla de páginas
  - Si todas las entradas de una tabla de páginas de un nivel están marcadas como inválidas, no es necesario almacenar esa tabla de páginas
  - Solo se necesita:
    - En MC la tabla de páginas de primer nivel
    - Las páginas de la tabla de páginas de primer nivel en posiciones físicas consecutivas
  - Se pueden compartir las tablas de páginas intermedias

Paginación con un nivel de páginas

Proceso A

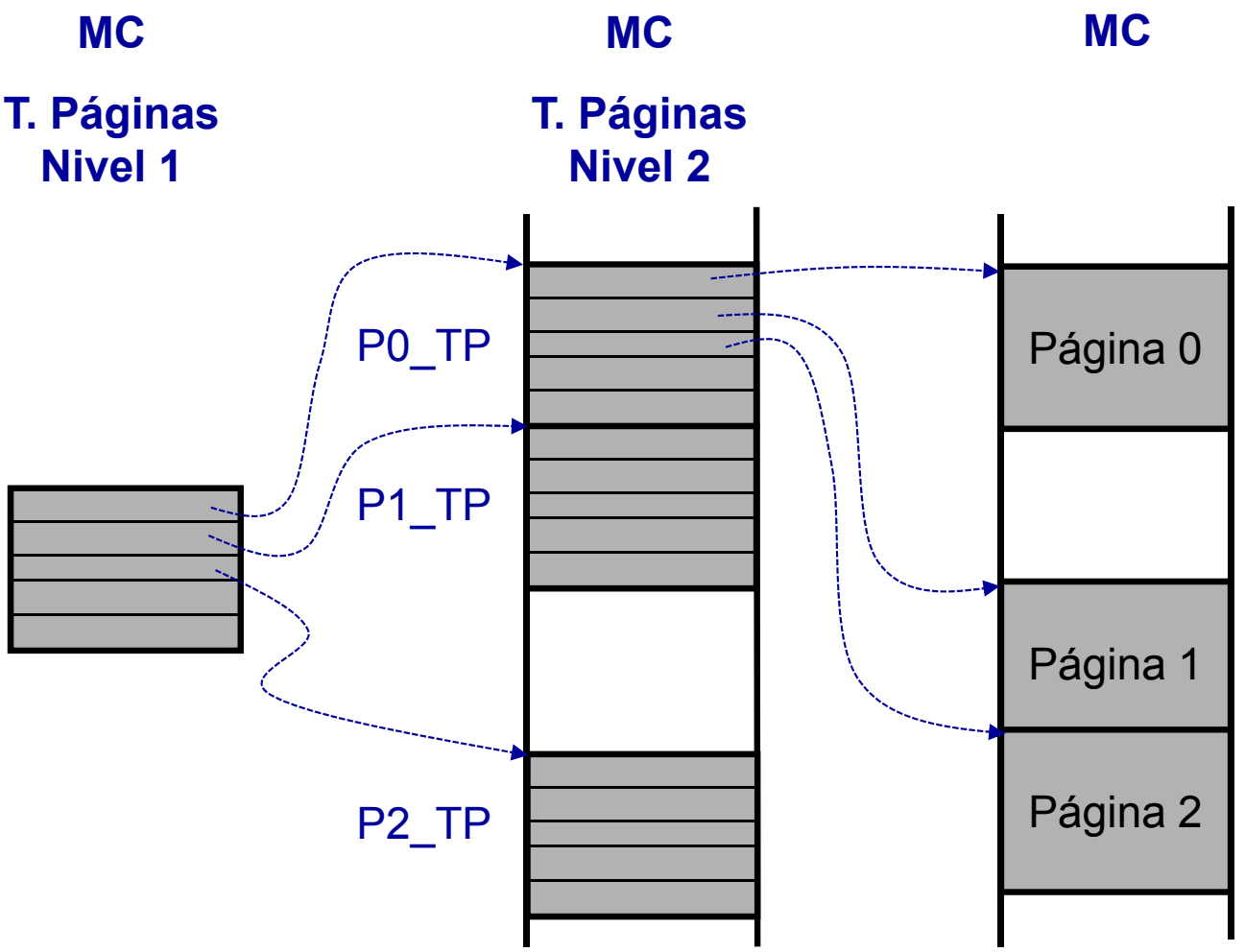
- Las páginas de la TP han de ir en espacio físico contiguo
- Toda la TP en MC



Paginación multinivel

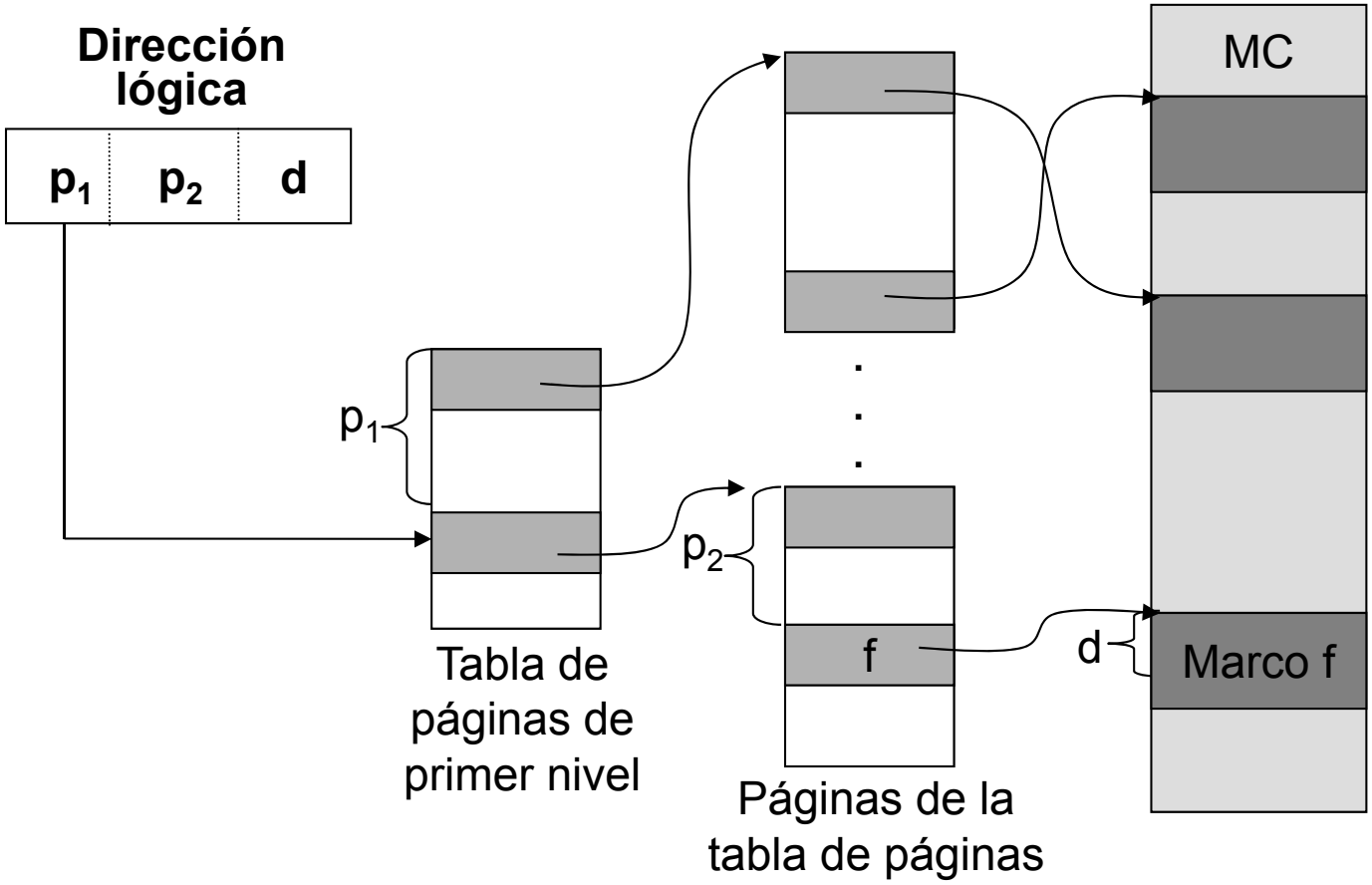
Proceso A

- Las páginas de la TP **no** han de ir en espacio físico contiguo
- **No** toda la TP en MC (sí las páginas de la TP de nivel 1)



Paginación multinivel

- **Mecanismo de traducción de direcciones:**



Paginación multinivel

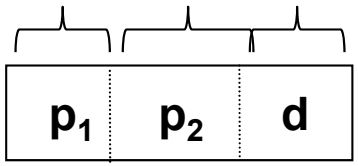
|Dir. Lógica| = 32 bits

|pag| = 4KB = 2^{12} by

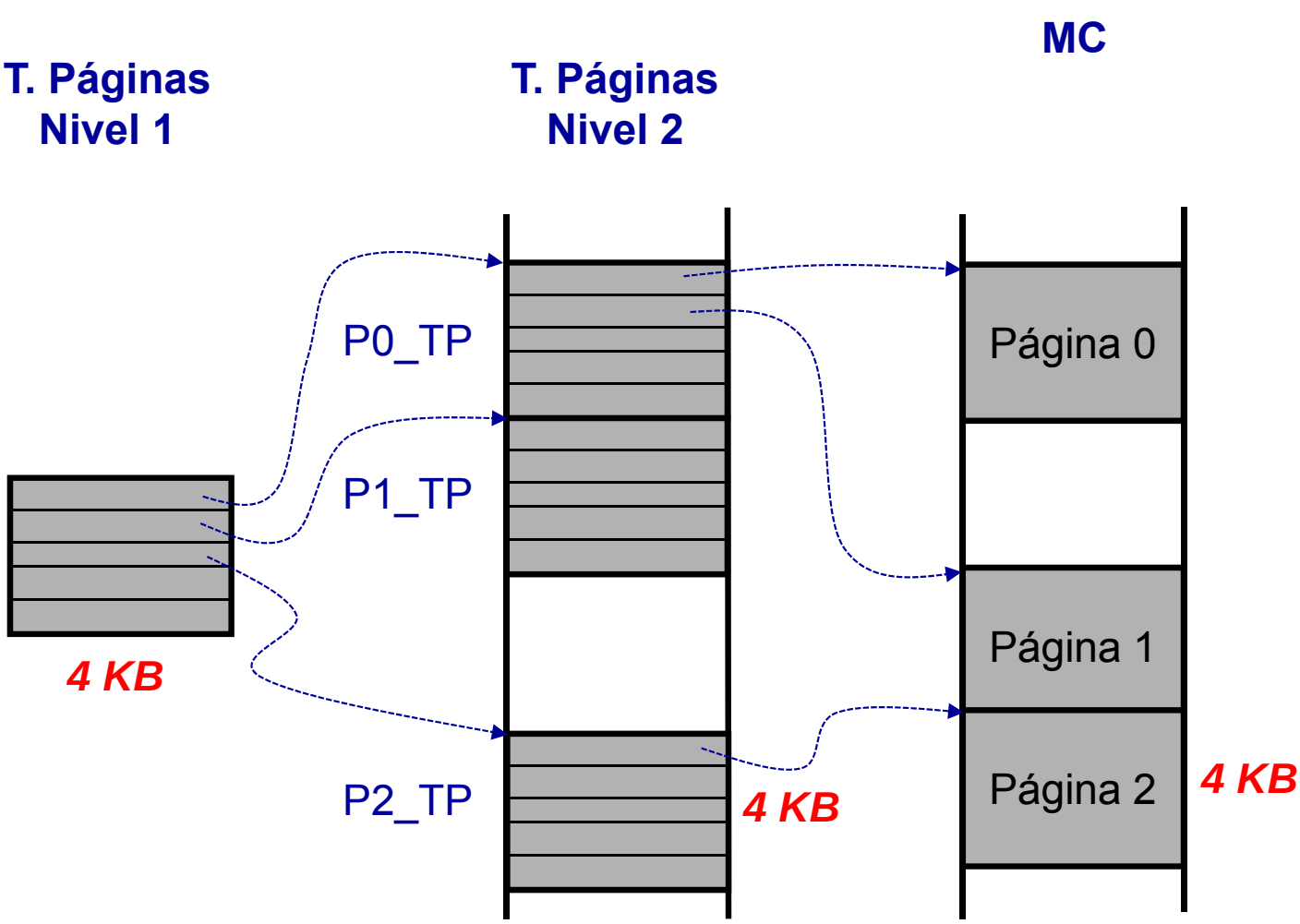
|pag TP|= 4KB

|entrada TP|= 4 by

¿Formato de una dirección lógica?



Dirección lógica



Paginación multinivel

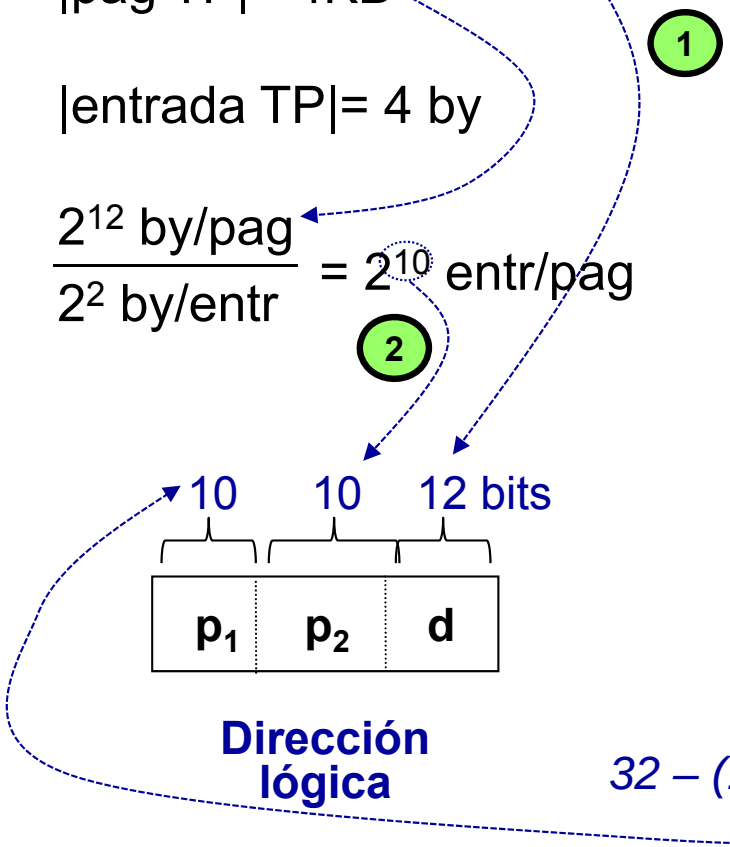
|Dir. Lógica| = 32 bits

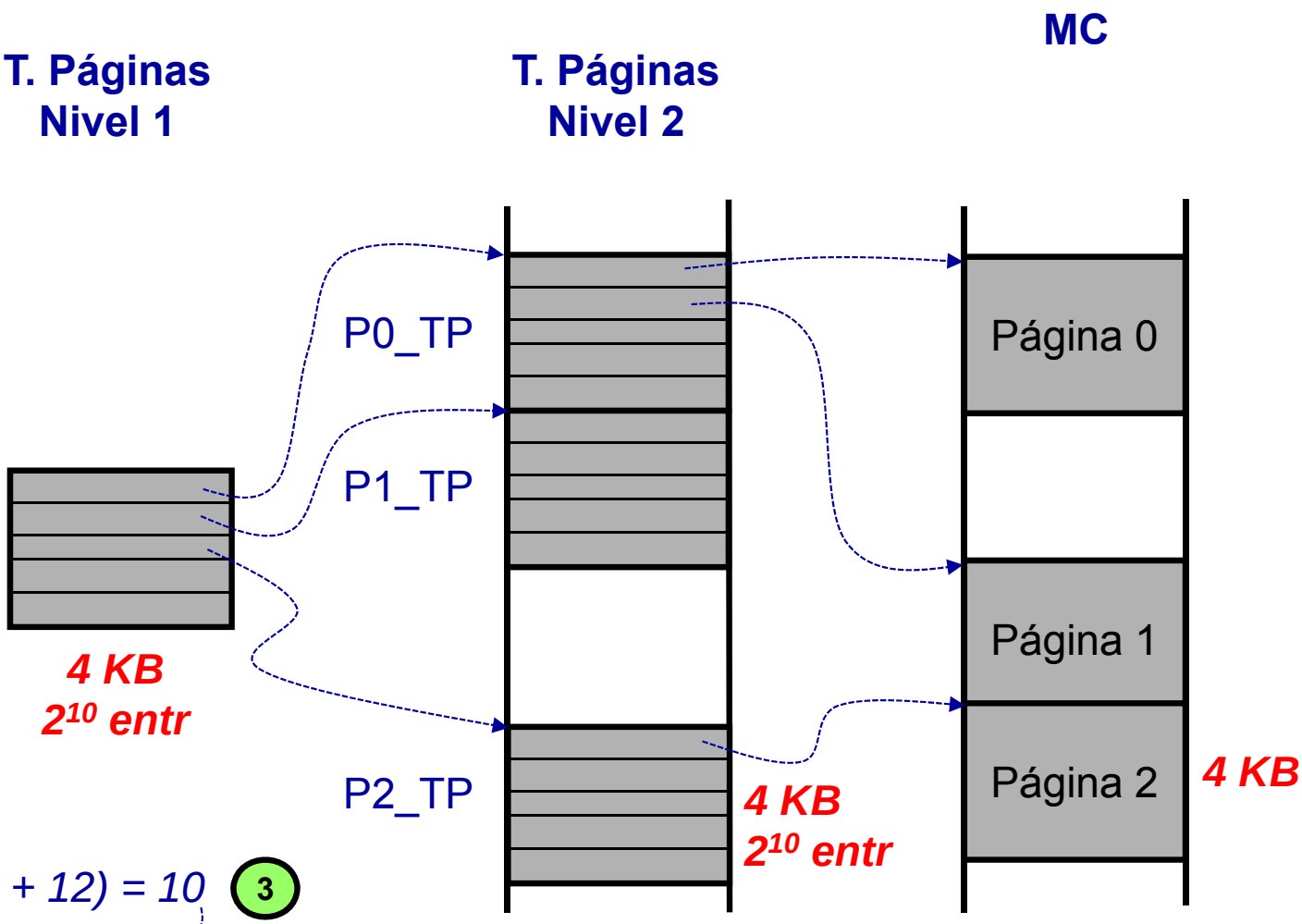
|pag| = 4KB = 2^{12} by

|pag TP|= 4KB

|entrada TP|= 4 by

$$\frac{2^{12}\ \text{by/pag}}{2^{2}\ \text{by/entr}} = 2^{10}\ \text{entr/pag}$$





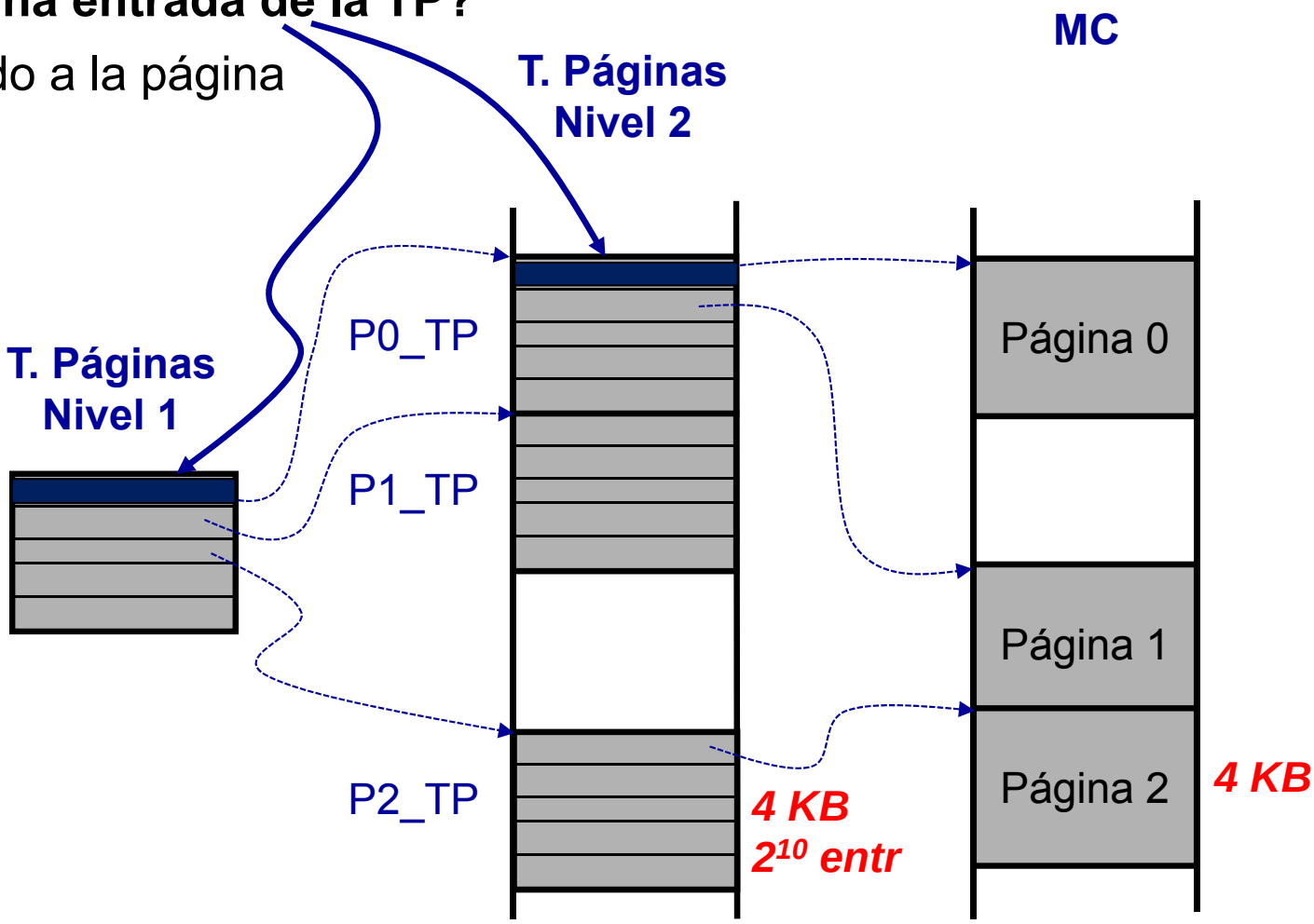
Paginación multinivel

- **Rendimiento:**
  - En el ejemplo anterior, cada acceso a un dato o a una instrucción requiere tres accesos a MC

    No obstante, la utilización de cachés permite una reducción razonable del tiempo de acceso

Paginación multinivel

- **¿Qué información hay en una entrada de la TP?**
  - Número de marco asociado a la página
  - Bits de control (estado)
    - ✓ Validez
    - ✓ Modificado
    - ✓ R/W
    - ✓ Referenciada



UNIVERSITAT JAUME·I

Índice

- Introducción
- Mapa de memoria de un proceso
- Gestión de la memoria principal
- Gestión de la memoria virtual
  - ✓ Motivación
  - ✓ Paginación bajo demanda
  - ✓ Paginación multinivel
  - ✓ Algoritmos de reemplazo de página
  - ✓ Asignación de memoria central a procesos
  - ✓ Hiperpaginación

Selección de página víctima

- Se deben aplicar algoritmos de selección que conlleven el menor índice posible de fallos de página (*p*)
- Se evalúan los algoritmos aplicándolos a unas condiciones dadas: secuencia de referencias y número de tramas de MC
- Algoritmos básicos:
  - FIFO
  - Segunda oportunidad (reloj)
  - Óptimo
  - LRU
- Para ilustrar su comportamiento se va a utilizar la siguiente secuencia de páginas referenciadas:

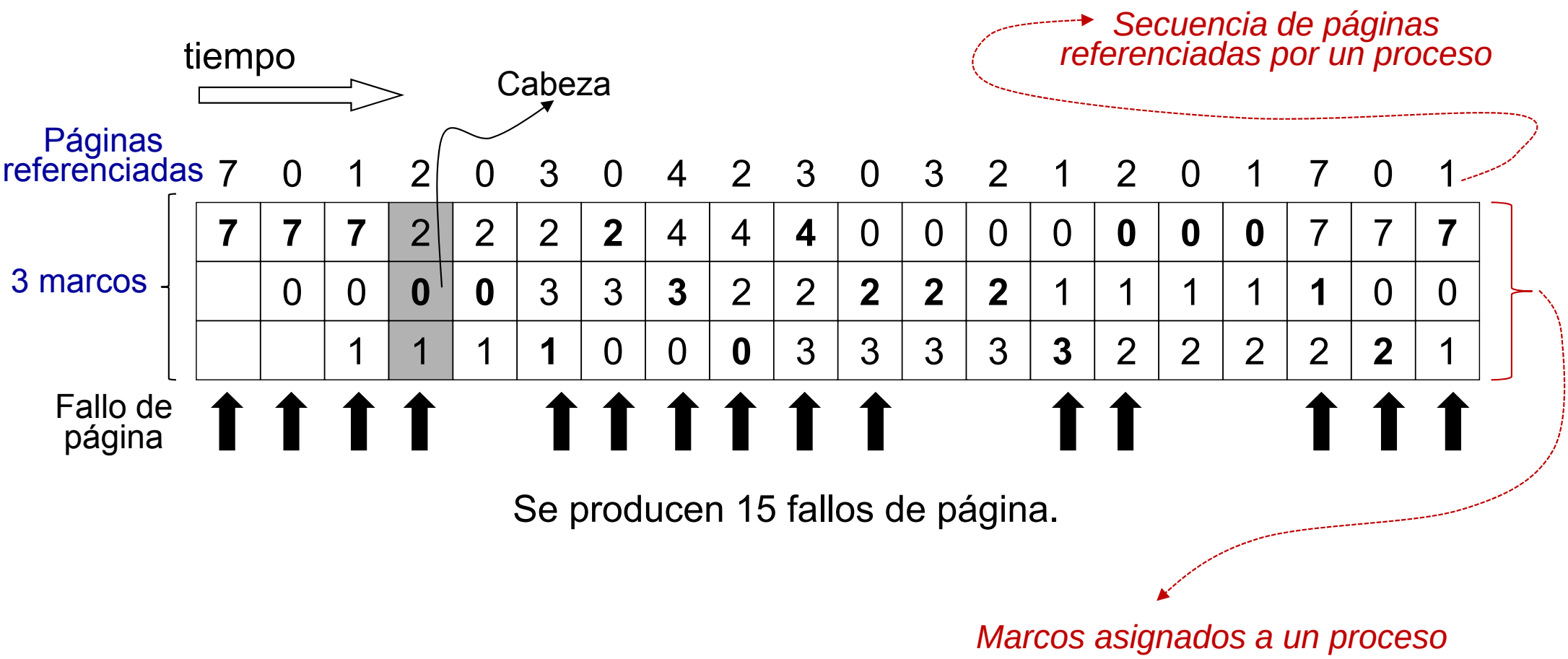
  7 , 0 , 1 , 2 , 0 , 3 , 0 , 4 , 2 , 3 , 0 , 3 , 2 , 1 , 2 , 0 , 1 , 7 , 0 , 1

Algoritmo FIFO

- Se elige como página víctima **aquella que más tiempo lleva cargada en memoria sin tener en cuenta su utilización**
- **Implementación:**
  - Cola FIFO que contenga las páginas que están en MC
  - Reemplazar la página que esté a la cabecera de la cola
  - Al traer una página a MC, insertarla al final de la cola
- **Ventaja:**
  - Fácil implementación
- **Inconveniente:**
  - El rendimiento no siempre es bueno: **Anomalía de Belady**

Más marcos $\not\Rightarrow$ Menos fallos de página.

Algoritmo FIFO



Se producen 15 fallos de página.

Algoritmo FIFO

- **Anomalía de Belady:**
  - ¿Qué ocurre con la secuencia 1,2,3,4,1,2,5,1,2,3,4,5 ?

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

9 fallos de página

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

10 fallos de página

Se producen más fallos de página utilizando 4 marcos que utilizando 3 marcos

Algoritmo segunda oportunidad (reloj)

- Variante FIFO que evita eliminar páginas muy utilizadas que llevan mucho tiempo en MC
- Si el bit de referencia para la primera página de la lista es:
  - 0: Se elige como página víctima
  - 1:
    - ✓ Se lleva al final de la lista (se le da una segunda oportunidad)
    - ✓ Se desactiva el bit de referencia
    - ✓ Se comprueba la siguiente página de la lista
- Implementación:
    - ✓ Lista circular
    - ✓ El puntero apunta al inicio de la lista
    - ✓ Si llega página a MC, se inserta donde el puntero y se avanza este al siguiente elemento de la lista
    - ✓ Imita el comportamiento de un reloj (aguja = puntero)

Algoritmo óptimo

- Se elige como página víctima aquella que al eliminarse de MC menor número de fallos de página va a producir en el futuro. O dicho de otra forma, **aquella que más tiempo va a tardar en volverse a referenciar**

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
↑	↑	↑	↑		↑		↑			↑			↑				↑		

9 fallos de página

- Este algoritmo no se puede aplicar porque se necesita información futura (lo mismo que ocurre cuando se intenta aplicar el algoritmo SJF en la planificación del procesador)
- Sirve como medida del comportamiento de otros algoritmos

Algoritmo LRU

LRU = Least Recently Used

- Se selecciona como página víctima **aquella que más tiempo lleva sin ser referenciada**
- Utilización del pasado reciente como aproximación del futuro cercano

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
↑	↑	↑	↑		↑		↑	↑	↑	↑			↑		↑		↑		

12 fallos de página

- Para su implementación es necesario que se gestione el sistema que permita identificar cuál de las páginas es la que más tiempo lleva sin referenciarse

Algoritmo LRU

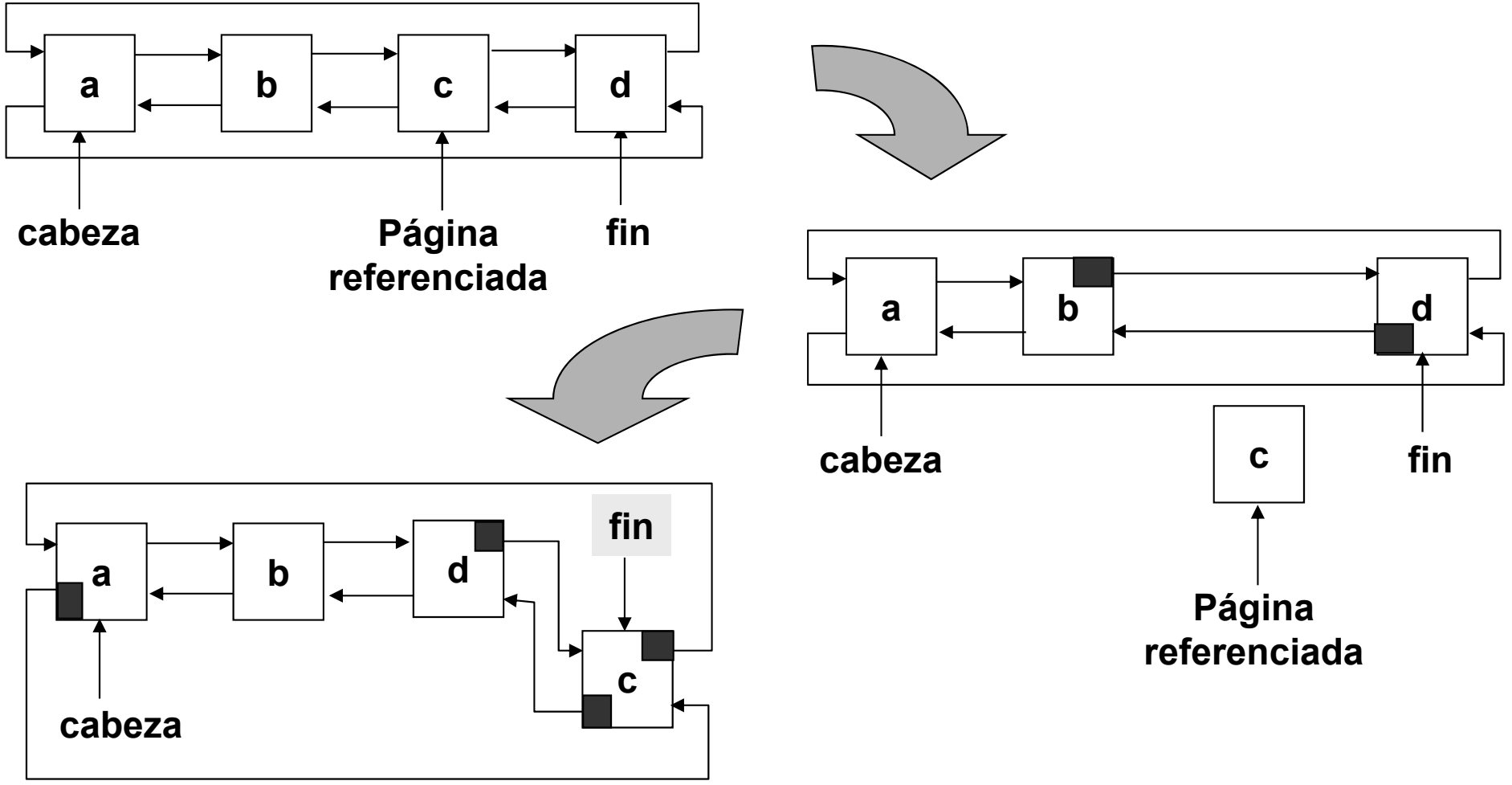
- **Implementación:**
  - **Mediante un contador:**
    - En cada descriptor de página se mantiene un contador que sirve para mantener el instante de tiempo en el que se referenció dicha página
      - ✓ Cada vez que se referencia una página se hace una copia del reloj del sistema en el contador de dicha página
      - ✓ Cuando se tiene que elegir una página víctima se necesita realizar una búsqueda para encontrar la página cuyo valor de contador sea más bajo

Algoritmo LRU

- **Implementación (cont.):**
  - **Mediante una cola de páginas:**
    - Se mantiene una pila en la que se encuentran todas las páginas que hay en memoria
      - ✓ Cada vez que se referencia una página se lleva al final de la cola
        - – Si se implementa con una estructura doblemente enlazada requiere únicamente la actualización de a lo sumo 8 punteros
      - ✓ Cuando hay que elegir una página víctima, se selecciona la que ocupa la cabeza de la cola
        - – No es necesario realizar búsquedas

Algoritmo LRU

- **Implementación mediante una cola de páginas (cont.):**



Algoritmo LRU

- **Implementación (cont.):**
  - **Mediante la utilización de bits de referencia:**
    - En cada descriptor de página se mantienen bits de referencia que indican la utilización de cada página

      Por ejemplo, con 5 bits:
      - Cuando se cargan la página en memoria se inicializan todos los bits a ‘1’
      - Cada X unidades de tiempo se desplazan los bits hacia la derecha y el bit de más peso toma valor ‘0’
      - Cada vez que se referencia una página el bit de más peso (bit de acceso) toma valor ‘1’

01010

Significa que los accesos a esa página se han realizado en los instantes de tiempo comprendidos en los intervalos

[-X , -2X] y [-3X , -4X]

Algoritmos globales o locales

- En cualquiera de las anteriores estrategias:
  - Un **algoritmo de selección de página víctima** es **global** cuando se puede seleccionar como página víctima cualquiera de las que en el momento de su aplicación se encuentran en memoria
  - Un **algoritmo de selección de página víctima** es **local** cuando sólo puede seleccionar páginas del proceso que ha producido el fallo de página

Tema 6. Gestión de memoria

UNIVERSITAT JAUME·I

Índice

- Introducción
- Mapa de memoria de un proceso
- Gestión de la memoria principal
- Gestión de la memoria virtual
  - ✓ Motivación
  - ✓ Paginación bajo demanda
  - ✓ Paginación multinivel
  - ✓ Algoritmos de reemplazo de página
  - ✓ Asignación de memoria central a procesos
  - ✓ Hiperpaginación

Asignación de marcos a procesos

- **¿Cómo repartir la memoria entre los procesos?**
  - Si disminuye el número de marcos asignados a un proceso, la frecuencia de fallos de página aumenta → Ralentización de la ejecución de procesos
  - Cada proceso necesita como **mínimo** tantos marcos (o tramas) de MC como número máximo de páginas pueden referenciarse al ejecutar una instrucción. De lo contrario se puede dar la situación de que no se pueda finalizar jamás la ejecución de una instrucción
  - El número máximo de marcos por proceso está definido por la cantidad de memoria física disponible
  - La asignación de marcos se puede hacer **fija** (un número de páginas fijo por proceso) o **proporcional** (dependiendo de su tamaño, prioridad y cualquier otro parámetro que los diseñadores del sistema operativo consideren importante)

UNIVERSITAT JAUME·I

Índice

- Introducción
- Mapa de memoria de un proceso
- Gestión de la memoria principal
- Gestión de la memoria virtual
  - ✓ Motivación
  - ✓ Paginación bajo demanda
  - ✓ Paginación multinivel
  - ✓ Algoritmos de reemplazo de página
  - ✓ Asignación de memoria central a procesos
  - ✓ Hiperpaginación

Hiperpaginación

- **Proceso hiperpaginado:**
  - Situación que se da cuando un proceso no tiene el suficiente número de páginas en MC

    Su tasa de fallo de página aumenta
- **Hiperpaginación ("trashing"):**
  - Se produce cuando la suma de los tamaños de la localidad de referencias de todos los procesos es mayor que la memoria central disponible

    Por ejemplo, cuando aumenta mucho el grado de multiprogramación

Hiperpaginación

- Provoca un descenso en el rendimiento del sistema, debido a que se utiliza la CPU para servir muchos fallos de página y además el dispositivo de paginación está siendo utilizado al máximo
- Si el algoritmo de selección de página víctima es local se puede localizar el proceso que está provocando la hiperpaginación
- Si el algoritmo de selección de página víctima es global la hiperpaginación de un proceso puede afectar a todos los demás

Hiperpaginación

- **Estrategias para prevenir la hiperpaginación:**
  - Estrategia del conjunto de páginas de trabajo
  - Control de carga y reemplazo global

Hiperpaginación

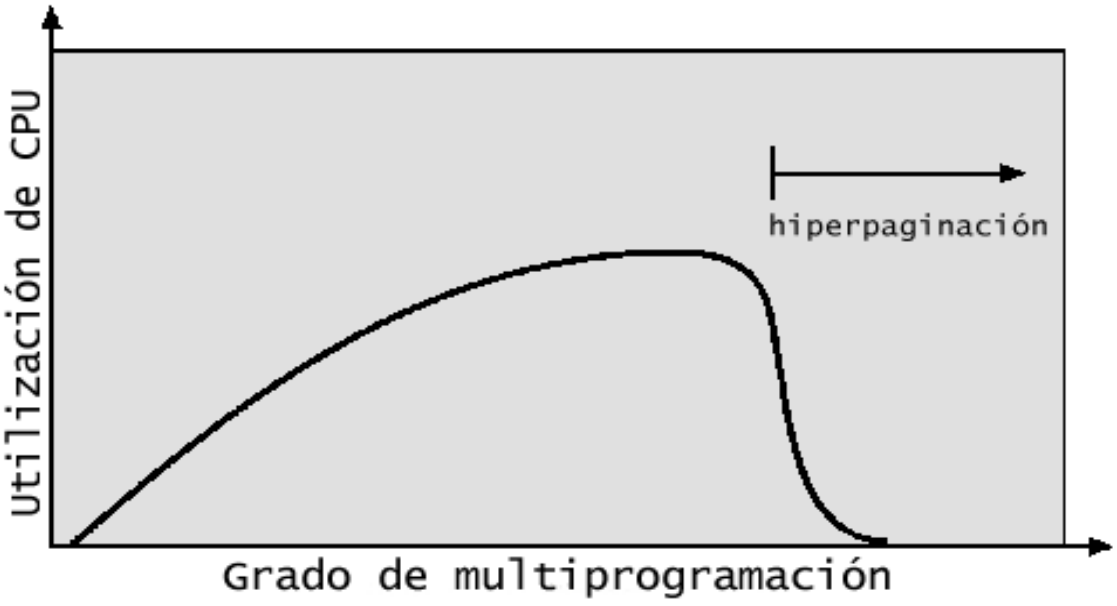
- **Estrategia del conjunto de páginas de trabajo:**
  - Establecer el conjunto de páginas de trabajo para cada proceso (“working set”)
  - Si la suma de los tamaños de los conjuntos de página de todos los procesos supera el número de marcos de MC existentes, detener la ejecución de alguno/s de los procesos
  - Evita la hiperpaginación y mantiene el grado de multiprogramación lo más alto posible → Optimización de la CPU
  - Inconvenientes:
    - Detección difícil del conjunto de trabajo de un proceso
    - Difícil implementación

Hiperpaginación

- **Control de carga y reemplazo global:**
  - Algoritmo de reemplazo global que coopera con un algoritmo de control de carga
  - Ejemplo: UNIX 4.3 BSD:
    - Uso de *buffering* de páginas
      - ✓ Un “demonio de paginación” controla el nº de marcos libres
    - Si número de marcos libres < umbral:
      - ✓ El demonio de paginación aplica reemplazo
    - Si se repite con frecuencia la falta de marcos libres:
      - ✓ Proceso “swapper” suspende procesos

Hiperpaginación

- **Diagrama de hiperpaginación:**



Ejercicios

- **Ejercicio 1:**

  ¿Cuántos accesos se realizarán en el peor de los casos a la memoria caché asociada a la tabla de descriptores de página cuando se traduce una dirección lógica en una dirección física en un sistema de paginación?

Ejercicios

- **Ejercicio 2:**

  En un sistema de gestión de memoria mediante paginación, la tabla de páginas de un proceso tiene el siguiente contenido:

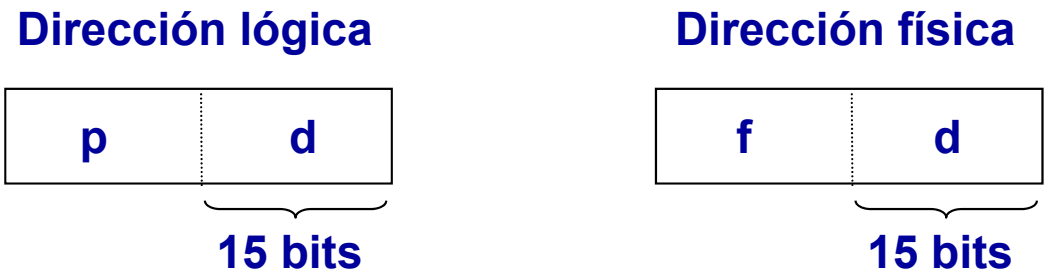
0	0xF00
1	0x50F
2	0xFA0
3	0x16F
4	0xA12
5	0x15F

  Si el tamaño de página es de 32 K posiciones, ¿qué dirección física genera la dirección lógica 0x02AFAF?

Ejercicios

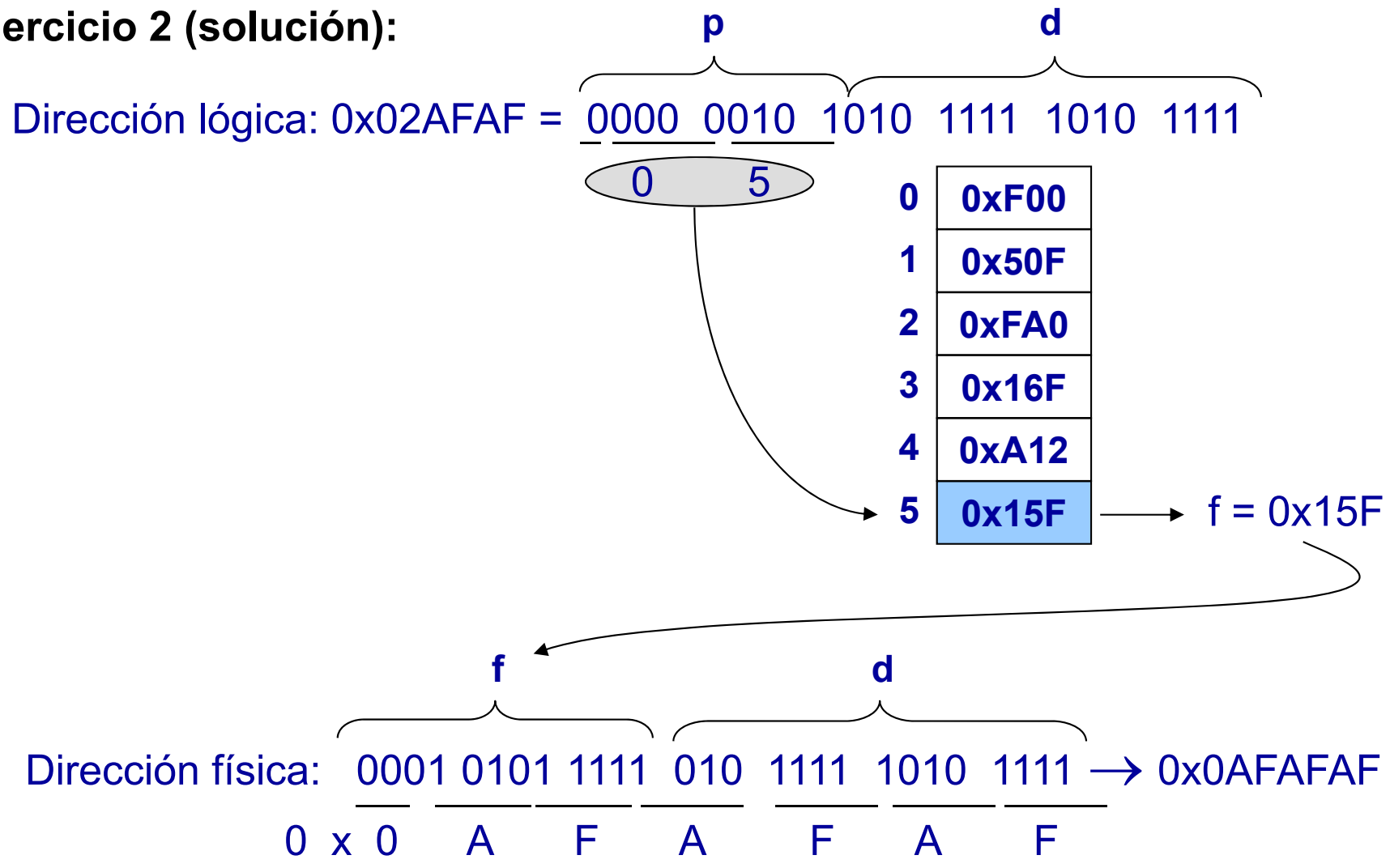
- **Ejercicio 2 (solución):**

  |Página| = 32KB = 2^{15} by $\rightarrow$ |d|=15 bits



Ejercicios

- **Ejercicio 2 (solución):**



Ejercicios

- **Ejercicio 3:**

  En un sistema de gestión de memoria mediante segmentación paginada cada proceso puede dividirse en un máximo de 4 segmentos y cada segmento en un máximo de 16 páginas.

  Un proceso tiene asignados dos segmentos de 90 y 50 KB, respectivamente y el contenido de su tabla de páginas es el siguiente:

0xF00
0x50F
0xFA0
0x16F
0xA12

  Si el tamaño de página es de 32 KB, ¿qué dirección física genera la dirección lógica 0x08AFAF?

Ejercicios

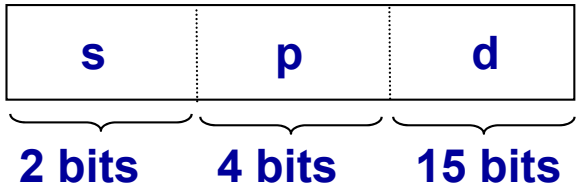
- **Ejercicio 3 (solución):**

  |Página| = 32KB = 2^{15} by $\rightarrow$ |d|=15 bits

  Máximo 16 páginas/segmento = 2^4 páginas/segmento $\rightarrow$ |p|=4 bits

  Máximo 4 segmentos/proceso = 2^2 segmentos/proceso $\rightarrow$ |s|=2 bits

Dirección lógica



Dirección física

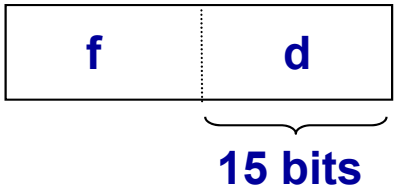
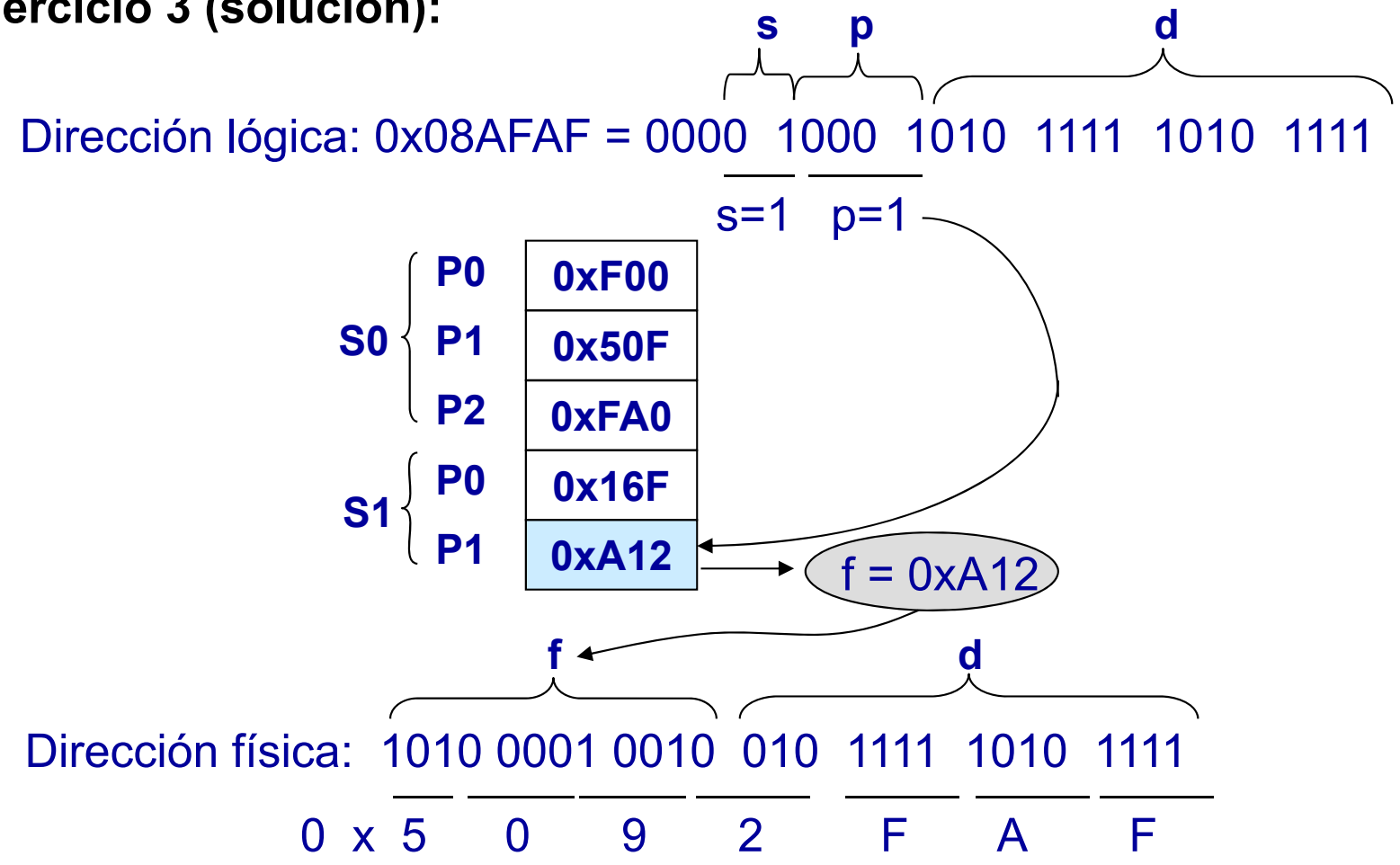


Tabla de páginas:

S0	P0	0xF00
	P1	0x50F
	P2	0xFA0
S1	P0	0x16F
	P1	0xA12

Ejercicios

- **Ejercicio 3 (solución):**



Ejercicios

- **Ejercicio 4:**

  En un sistema de gestión de memoria mediante segmentación paginada cada proceso puede dividirse en un máximo de 8 segmentos y cada segmento en un máximo de 64 páginas.

  Si el tamaño de página es de 32 KB, ¿cuál de las siguientes direcciones lógicas corresponde a la página 0x3A del segmento 0x2?

  a) 0x5AAF92

  b) 0x5D44E7

  c) 0x5546A5

  d) 0x5DD897

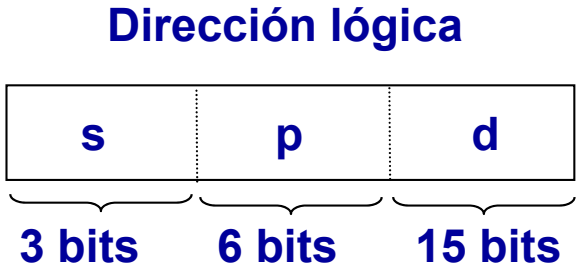
Ejercicios

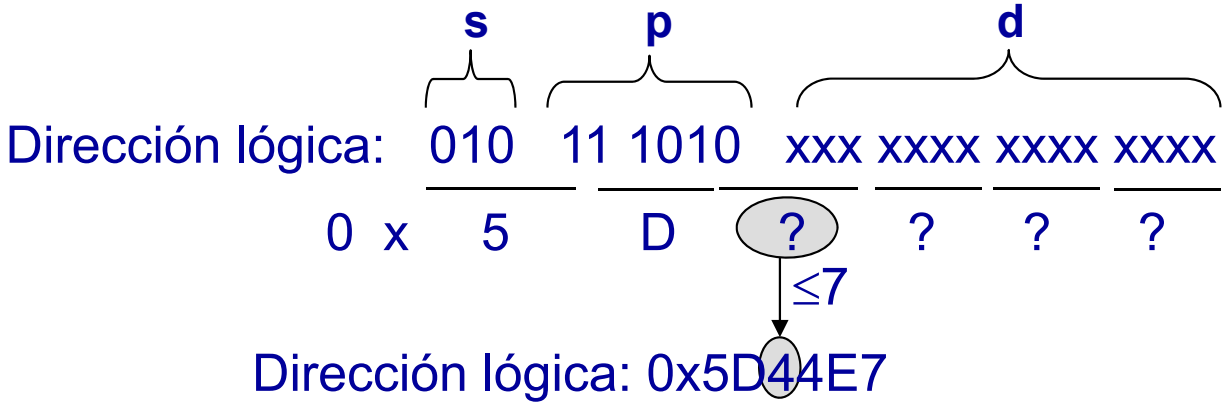
- **Ejercicio 4 (solución):**

  |Página| = 32KB = 2^{15} by $\rightarrow$ |d|=15 bits

  Máximo 64 páginas/segmento = 2^6 páginas/segmento $\rightarrow$ |p|=6 bits

  Máximo 8 segmentos/proceso = 2^3 segmentos/proceso $\rightarrow$ |s|=3 bits





Ejercicios

- **Ejercicio 5:**

  Sistema con paginación bajo demanda.

  Direcciones generadas en la ejecución de un proceso:

Direcciones lógicas	Direcciones físicas
0x3B3AB3	0X07FAB3
0x1FD111	0x0FD111
0X3B2020	0X07E020
0X0B2456	0X07E456
0X3B3345	0X0FF345

Ejercicios

- **Ejercicio 5 (solución):**

  ¿Tamaño máximo de página con el que se está trabajando?

  Este número se obtiene a partir del desplazamiento que se puede utilizar para referenciar cualquier byte dentro de una página o marco; es decir, el número "d" de bits comunes entre una dirección lógica y una dirección física. En el ejercicio, este número es, tal y como puede observarse a continuación, 14 bits.

3B3AB3 ≡ 0011 1011 0011 1010 1011 0011	07FAB3 ≡ 0000 0111 1111 1010 1011 0011
1FD111 ≡ 0001 1111 1101 0001 0001 0001	0FD111 ≡ 0000 1111 1101 0001 0001 0001
3B2020 ≡ 0011 1011 0010 0000 0010 0000	07E020 ≡ 0000 0111 1110 0000 0010 0000
0B2456 ≡ 0000 1011 0010 0100 0101 0110	07E456 ≡ 0000 0111 1110 0100 0101 0110
3B3345 ≡ 0011 1011 0011 0011 0100 0101	0FF345 ≡ 0000 1111 1111 0011 0100 0101

El tamaño máximo de página es, por lo tanto, 2^{14} by= 16KB.

Ejercicios

- **Ejercicio 5 (solución):**

  ¿Tamaño máximo de página con el que se está trabajando?

  Este número se obtiene a partir del desplazamiento que se puede utilizar para referenciar cualquier byte dentro de una página o marco. Concretamente, es el menor número de bits comunes de los pares (dirección lógica , dirección física). En el ejercicio, este número es, tal y como puede observarse a continuación, 14 bits.

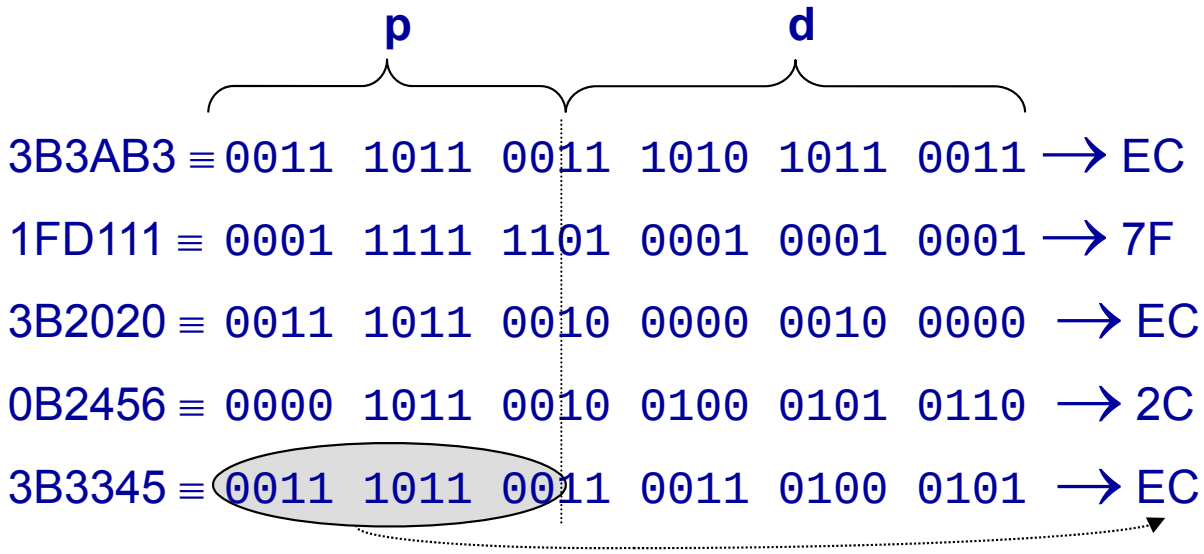
```
3B3AB3 ≡ 0011 1011 0011 1010 1011 0011   07FAB3 ≡ 0000 0111 1111 1010 1011 0011
1FD111 ≡ 0001 1111 1101 0001 0001 0001   0FD111 ≡ 0000 1111 1101 0001 0001 0001
3B2020 ≡ 0011 1011 0010 0000 0010 0000   07E020 ≡ 0000 0111 1110 0000 0010 0000
0B2456 ≡ 0000 1011 0010 0100 0101 0110   07E456 ≡ 0000 0111 1110 0100 0101 0110
3B3345 ≡ 0011 1011 0011 0011 0100 0101   0FF345 ≡ 0000 1111 1111 0011 0100 0101
```

  El tamaño máximo de página es, por lo tanto, 2^{14} by= 16KB.

Ejercicios

- **Ejercicio 5 (solución):**

  ¿Secuencia de páginas a las que se ha accedido?

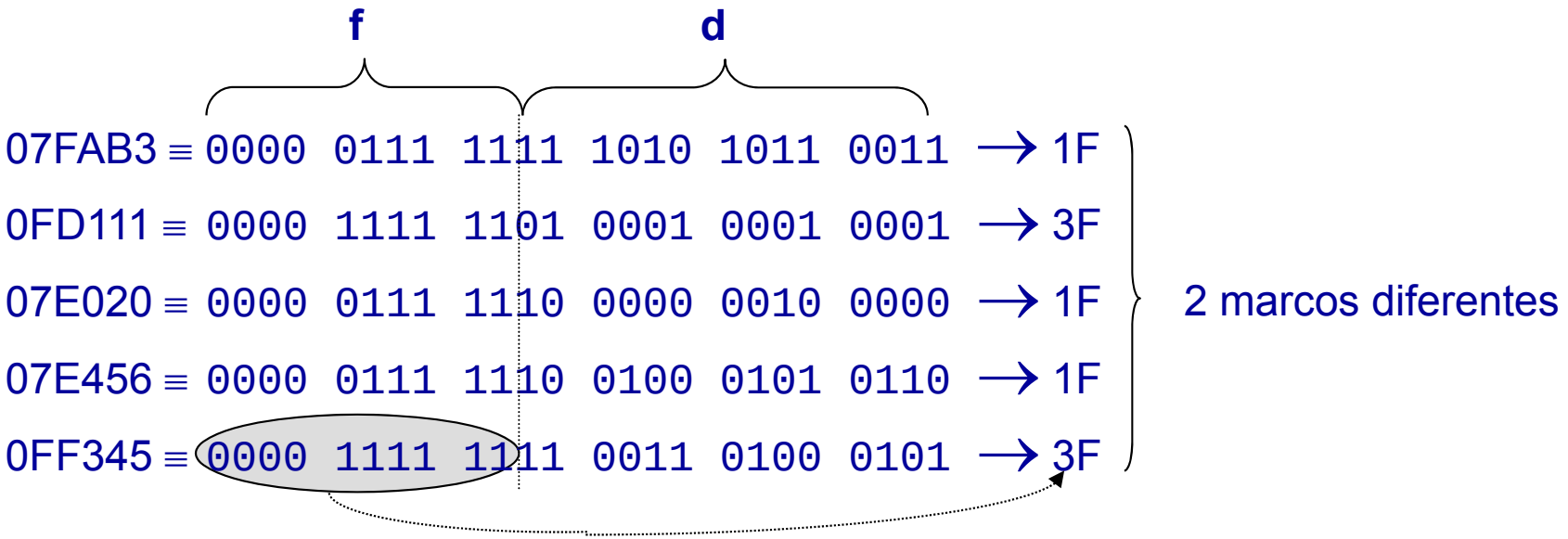


La secuencia de páginas es, por lo tanto, EC, 7F, EC, 2C, EC.

Ejercicios

- **Ejercicio 5 (solución):**

  ¿ Algoritmo (local) de reemplazo de páginas?



Pags. referenciadas		EC	7F	EC	2C	EC
Macos disponibles	1F	EC	EC	EC	2C	2C
	3F		7F	7F	7F	EC

→ Algoritmo FIFO

Ejercicios

- **Ejercicio 6:**

  Sistema con 2MB: 640KB para el SO

  Segmentación paginada

  Máximo 8 segmentos/proceso

  Máximo 64 páginas/segmento

  |Página| = 32KB

Proceso A:	S0: 70KB	Proceso B:	S0: 110KB	Proceso C:	S0: 82KB
	S1: 18KB		S1: 56KB		S1: 40KB
	S2: 60KB				S2: 22KB

Ejercicios

- **Ejercicio 6 (cont.):**

  Secuencia de accesos a páginas:

(Proceso,Segmento,Página)

	(C,1,1)	(C,2,0)	(A,0,1)	(A,0,2)	(B,0,1)	(C,1,0)	(A,0,1)	(A,0,2)	(B,0,2)	(B,0,3)	(B,0,1)	(A,2,0)	(A,0,2)
0x02B			(A,0,1)	(A,0,1)	(A,0,1)	(A,0,1)	(A,0,1)	(A,0,1)	(A,0,1)	(A,0,1)	(A,0,1)	(A,2,0)	(A,2,0)
0x02D				(A,0,2)	(A,0,2)	(A,0,2)	(A,0,2)	(A,0,2)	(A,0,2)	(A,0,2)	(A,0,2)	(A,0,2)	(A,0,2)
0x01C					(B,0,1)	(B,0,1)	(B,0,1)	(B,0,1)	(B,0,1)	(B,0,1)	(B,0,1)	(B,0,1)	(B,0,1)
0x01D									(B,0,2)	(B,0,3)	(B,0,3)	(B,0,3)	(B,0,3)
0x00E	(C,1,1)	(C,1,1)	(C,1,1)	(C,1,1)	(C,1,1)	(C,1,0)	(C,1,0)	(C,1,0)	(C,1,0)	(C,1,0)	(C,1,0)	(C,1,0)	(C,1,0)
0x00F		(C,2,0)	(C,2,0)	(C,2,0)	(C,2,0)	(C,2,0)	(C,2,0)	(C,2,0)	(C,2,0)	(C,2,0)	(C,2,0)	(C,2,0)	(C,2,0)

Ejercicios

- **Ejercicio 6 (solución):**

  ¿Estrategia de sustitución de páginas?

  Proceso A → LRU, FIFO u Óptimo

  Proceso B → ~~LRU~~, ~~FIFO~~ Óptimo

  Proceso C → LRU, FIFO u Óptimo

  Algoritmo de sustitución de páginas óptimo local.

Ejercicios

- **Ejercicio 6 (solución):**

  ¿Estado tabla de páginas al final de la secuencia de accesos?

  Proc. A:
  - S0: 70KB → 3 páginas
  - S1: 18KB → 1 página
  - S2: 60KB → 2 páginas

  Proc. B:
  - S0: 110KB → 4 páginas
  - S1: 56KB → 2 páginas

  Proc. C:
  - S0: 82KB → 3 páginas
  - S1: 40KB → 2 página
  - S2: 22KB → 1 página

(A,0,0)		0
(A,0,1)	02B	0
(A,0,2)	02D	1
(A,1,0)		0
(A,2,0)	02B	1
(A,2,1)		0

(B,0,0)		0
(B,0,1)	01C	1
(B,0,2)	01D	0
(B,0,3)	01D	1
(B,1,0)		0
(B,1,1)		0

(C,0,0)		0
(C,0,1)		0
(C,0,2)		0
(C,1,0)	00E	1
(C,1,1)	00E	0
(C,2,0)	00F	1

Ejercicios

- **Ejercicio 6 (solución):**

  ¿Fragmentación interna máxima que se puede generar?

¿Cuánto espacio se podría desaprovechar como máximo?

El generado por las últimas páginas de cada segmento si todas ellas stuviesen en MC

(A,0,0)		0
(A,0,1)	02B	0
(A,0,2)	02D	1
(A,1,0)		0
(A,2,0)	02B	1
(A,2,1)		0

(B,0,0)		0
(B,0,1)	01C	1
(B,0,2)	01D	0
(B,0,3)	01D	1
(B,1,0)		0
(B,1,1)		0

(C,0,0)		0
(C,0,1)		0
(C,0,2)		0
(C,1,0)	00E	1
(C,1,1)	00E	0
(C,2,0)	00F	1

Ejercicios

- **Ejercicio 6 (solución):**

  ¿Fragmentación interna máxima que se puede generar?

Proceso	Segmento	NPáginas	Fragmentación interna que **puede** producir
A	0	3	3*32K - 70K = 26K
A	1	1	1*32K - 18K = 14K
A	2	2	2*32K - 60K = 4K
B	0	4	4*32K - 110K = 18K
B	1	2	2*32K - 56K = 8K
C	0	3	3*32K - 82K = 14K
C	1	2	2*32K - 40K = 24K
C	2	1	1*32K - 22K = 10K

Total: 118 KB

Ejercicios

- **Ejercicio 6 (solución):**

  ¿Fragmentación interna generada en ese instante?

Realmente solo genera fragmentación interna la última página de cada segmento si esta está cargada en MC

(A,0,0)		0
(A,0,1)	02B	0
(A,0,2)	02D	1
(A,1,0)		0
(A,2,0)	02B	1
(A,2,1)		0

(B,0,0)		0
(B,0,1)	01C	1
(B,0,2)	01D	0
(B,0,3)	01D	1
(B,1,0)		0
(B,1,1)		0

(C,0,0)		0
(C,0,1)		0
(C,0,2)		0
(C,1,0)	00E	1
(C,1,1)	00E	0
(C,2,0)	00F	1

Ejercicios

- **Ejercicio 6 (solución):**

  ¿Fragmentación interna generada en ese instante?

Proceso	Segmento	Última página	¿En MC?	Fragment interna **generada**
A	0	A, 0, 2	Sí	26K
A	1	A, 1, 0	No	0
A	2	A, 2, 1	No	0
B	0	B, 0, 3	Sí	18K
B	1	B, 1, 1	No	0
C	0	C, 0, 2	No	0
C	1	C, 1, 1	No	0
C	2	C, 2, 0	Sí	10K

Páginas cargadas en memoria que provocan fragmentación en ese instante:

(A,0,2) → 26 KB
(B,0,3) → 18 KB
(C,2,0) → 10 KB

} Total: 54 KB

Ejercicios

- **Ejercicio 6 (solución):**

  ¿Formato direcciones lógicas y físicas?

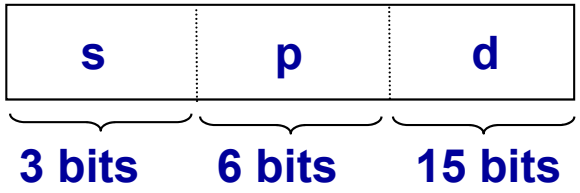
  |Página| = 32KB = 2^{15} by $\rightarrow$ |d|=15 bits

  Máximo 64 páginas/segmento = 2^6 páginas/segmento $\rightarrow$ |p|=6 bits

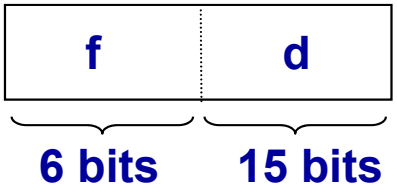
  Máximo 8 segmentos/proceso = 2^3 segmentos/proceso $\rightarrow$ |s|=3 bits

  2MB - 640KB = 1408 KB para procesos de usuario
  1408 KB / 32KB/marco = 44 marcos $\rightarrow$ $2^5<44<2^6$ $\rightarrow$ |f|=6 bits

Dirección lógica



Dirección física



Ejercicios

- **Ejercicio 6 (solución):**

  ¿Dirección física que genera la dirección lógica 0x2015BF en la siguiente situación si se sabe que la siguiente página a ser referenciada es la (A,2,0)?

(A,0,0)	02D	0
(A,0,1)	02B	1
(A,0,2)	02B	0
(A,1,0)	02D	0
(A,2,0)	02D	1
(A,2,1)	02B	0

Ejercicios

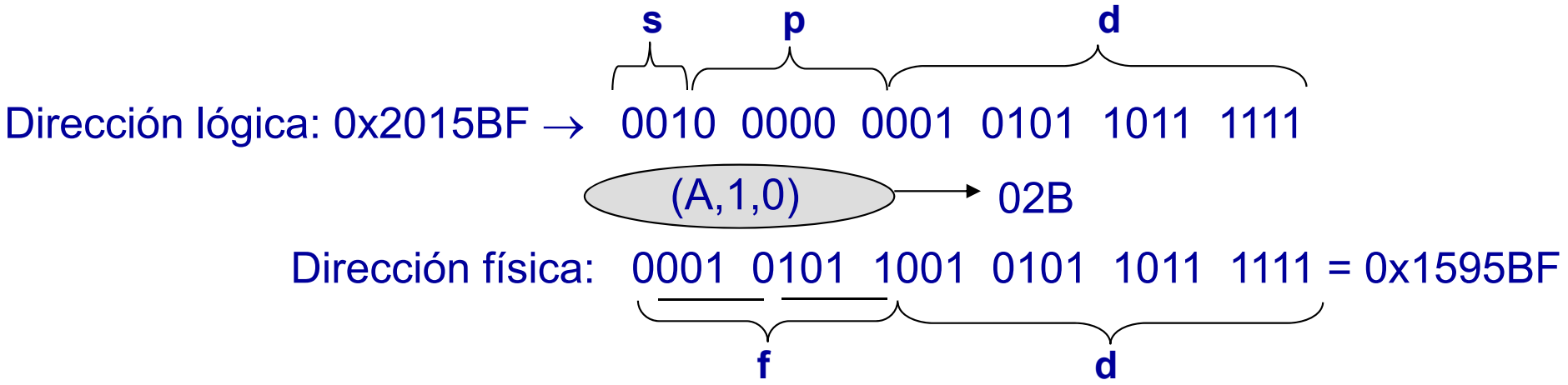
- **Ejercicio 6 (solución):**

  ¿Dirección física que genera la dirección lógica 0x2015BF en la siguiente situación si se sabe que la siguiente página a ser referenciada es la (A,2,0)?

(A,0,0)	02D	0
(A,0,1)	02B	1
(A,0,2)	02B	0
(A,1,0)	02D	0
(A,2,0)	02D	1
(A,2,1)	02B	0

Dirección física 0X1595BF

(A,0,0)	02D	0
(A,0,1)	02B	0
(A,0,2)	02B	0
(A,1,0)	02B	1
(A,2,0)	02D	1
(A,2,1)	02B	0



Ejercicios

- **Ejercicio 6 (solución):**

  ¿Dirección lógica generada en la siguiente situación?

(A,0,0)	02D	0
(A,0,1)	02B	0
(A,0,2)	02B	0
(A,1,0)	02B	1
(A,2,0)	02D	1
(A,2,1)	02B	0

Dirección física
0X1683A3

(A,0,0)	02D	0
(A,0,1)	02B	0
(A,0,2)	02B	0
(A,1,0)	02B	1
(A,2,0)	02D	1
(A,2,1)	02B	0

Dirección física: 0x1683A3 = 0001 0110 1000 0011 1010 0011 (f: 0001 0110; d: 1000 0011 1010 0011)

0 2 D → (A,2,0)

Dirección lógica: 0100 0000 0000 0011 1010 0011 → 0x4003A3 (s: 0100; p: 0000; d: 0000 0011 1010 0011)