Tema 5 (III)
Jerarquía de Memoria

Grupo ARCOS

Estructura de Computadores

Grado en Ingeniería Informática

Universidad Carlos III de Madrid

Contenidos

1. Memoria virtual
   - Definiciones iniciales
   - Motivación
   - Funcionamiento general
   - Memoria virtual paginada
   - Detalles de gestión

¡ATENCIÓN!

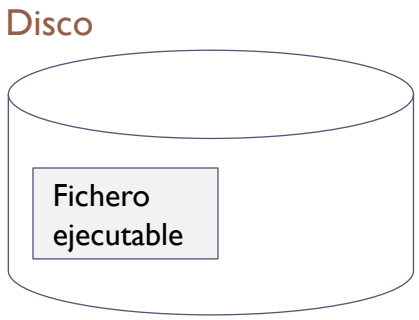
- Estas transparencias son un guión para la clase
- Los libros dados en la bibliografía junto con lo explicado en clase representa el material de estudio para el temario de la asignatura
  - Para la preparación de los exámenes se ha de utilizar todo el material de estudios

Contenidos

1. Memoria virtual
   - **Definiciones iniciales**
   - Motivación
   - Funcionamiento general
   - Memoria virtual paginada
   - Detalles de gestión

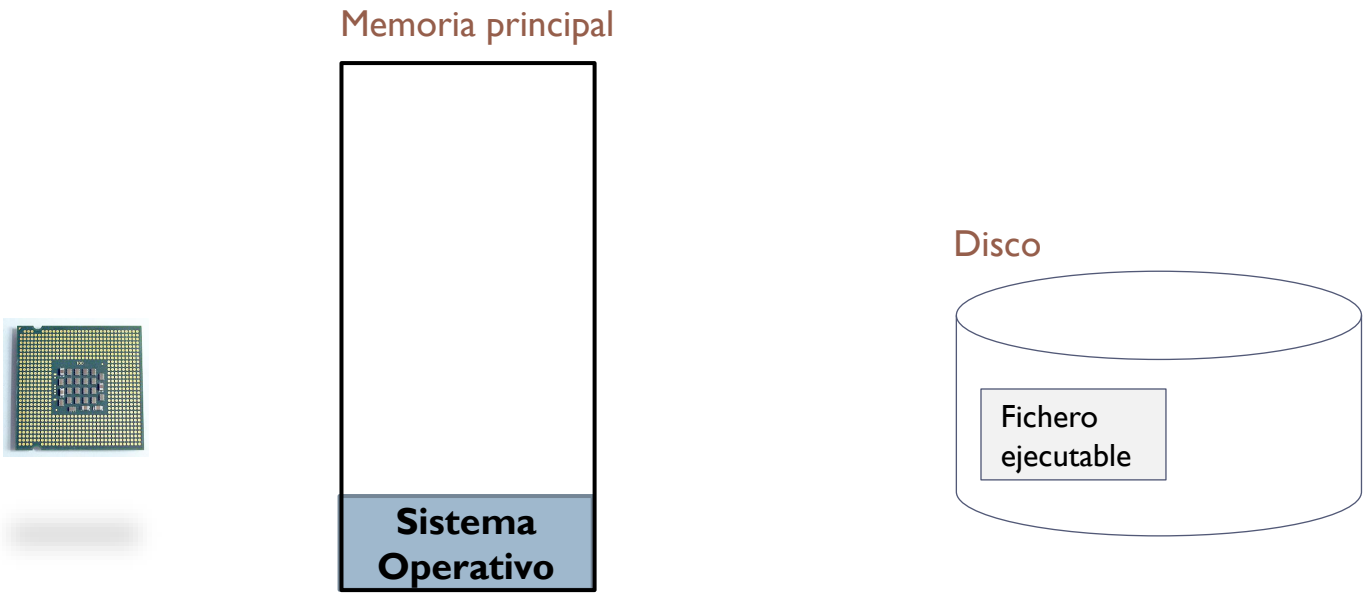
Programa y proceso

- Programa: conjunto de datos e instrucciones ordenadas que permiten realizar una tarea o trabajo específico.



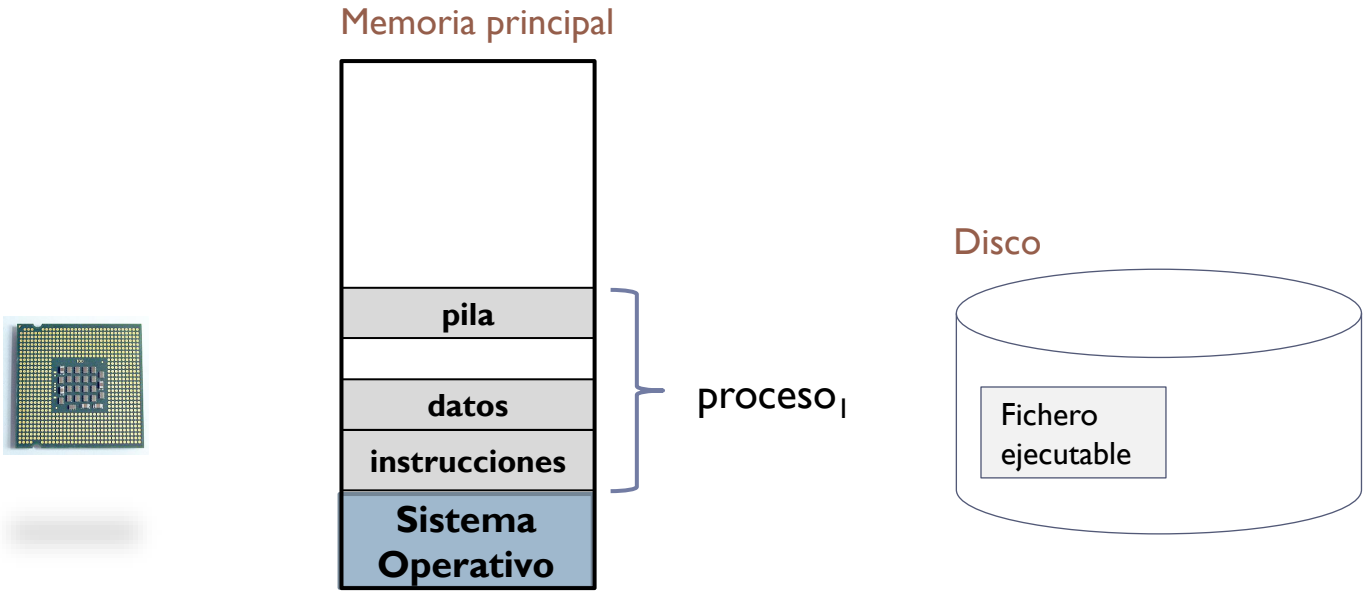
Programa y proceso

- Programa: conjunto de datos e instrucciones ordenadas que permiten realizar una tarea o trabajo específico.
  - Para su ejecución, ha de estar en memoria



Programa y proceso

- Proceso: programa en ejecución.



Programa y proceso

- Proceso: programa en ejecución.
  - Es posible un mismo programa ejecutarlo varias veces (lo que da lugar a varios procesos)

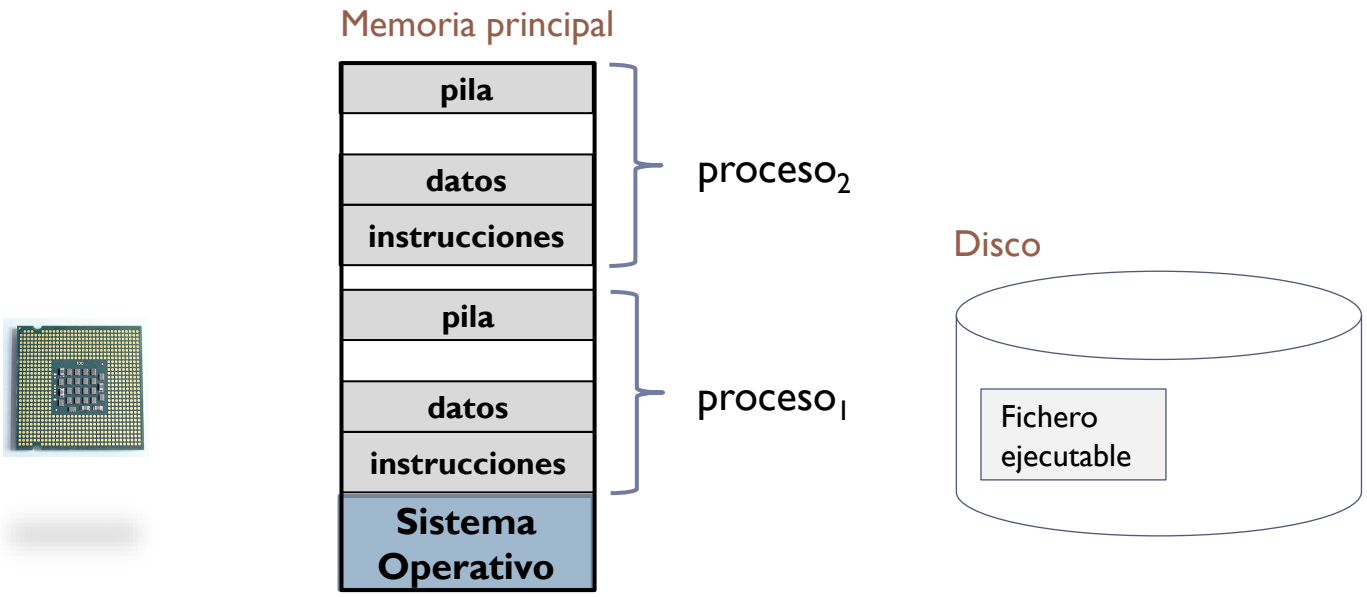
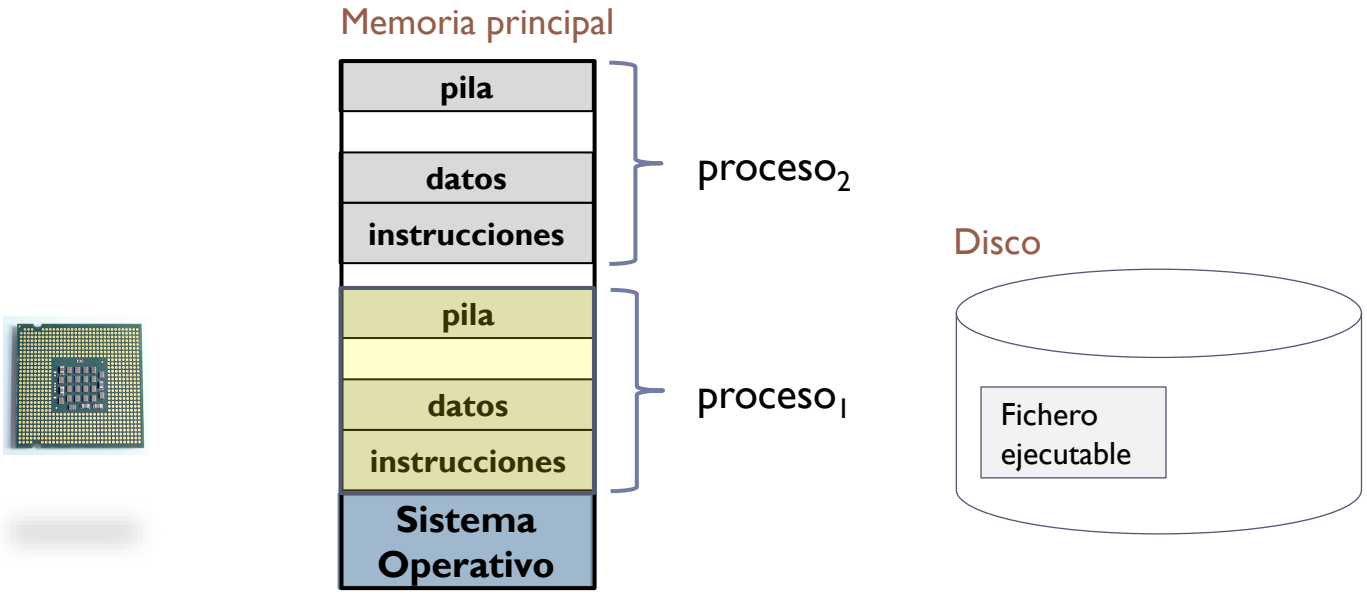


Imagen de un proceso

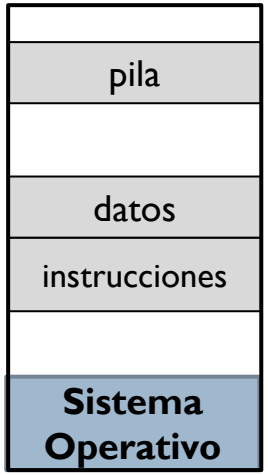
- Imagen de memoria: conjunto de direcciones de memoria asignadas al programa que se está ejecutando (y contenido)



Contenidos

1. Memoria virtual
   - Definiciones iniciales
   - **Motivación**
   - Funcionamiento general
   - Memoria virtual paginada
   - Detalles de gestión

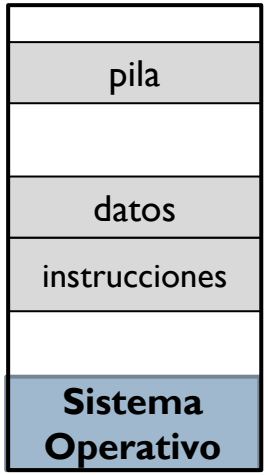
Sistemas **sin** memoria virtual



- En los sistemas sin memoria virtual, el programa se carga completamente en memoria para su ejecución.

- Principales problemas:
  - El tamaño de la imagen puede limitar su ejecución, o la de otros procesos.
  - La protección necesaria para que un proceso no tenga acceso a la imagen de otro proceso.
  - La necesidad de poder reubicar el programa en cualquier zona de memoria.

Sistemas **sin** memoria virtual

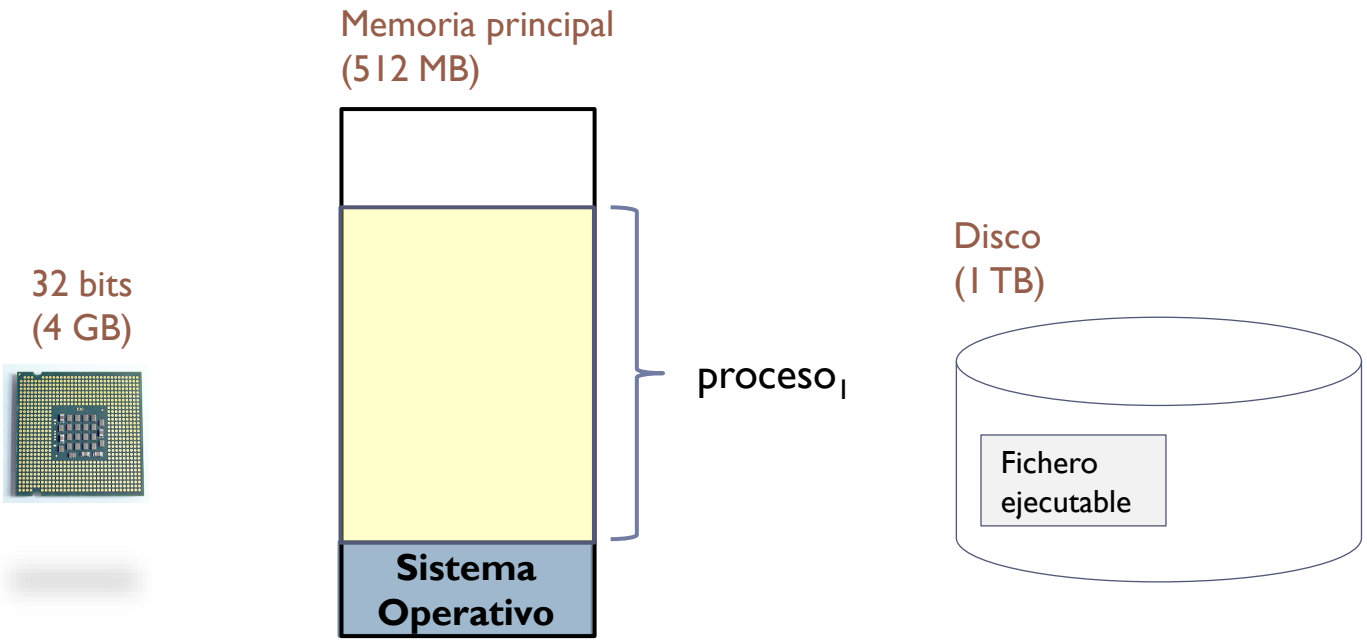


- En los sistemas sin memoria virtual, el programa se carga completamente en memoria para su ejecución.

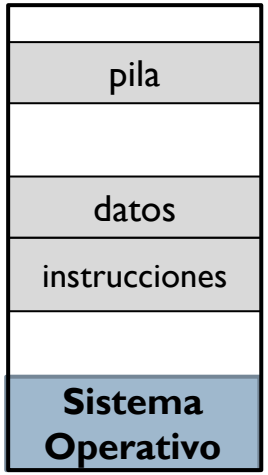
- Principales problemas:
  - El tamaño de la imagen puede limitar su ejecución, o la de otros procesos.
  - La protección necesaria para que un proceso no tenga acceso a la imagen de otro proceso.
  - La necesidad de poder reubicar el programa en cualquier zona de memoria.

Problema del tamaño limitado

- Si la imagen de memoria de un proceso es más grande que la memoria principal, no es posible su ejecución.
- El gran tamaño de la imagen en memoria de un proceso puede impedir ejecutar otros.



Sistemas **sin** memoria virtual

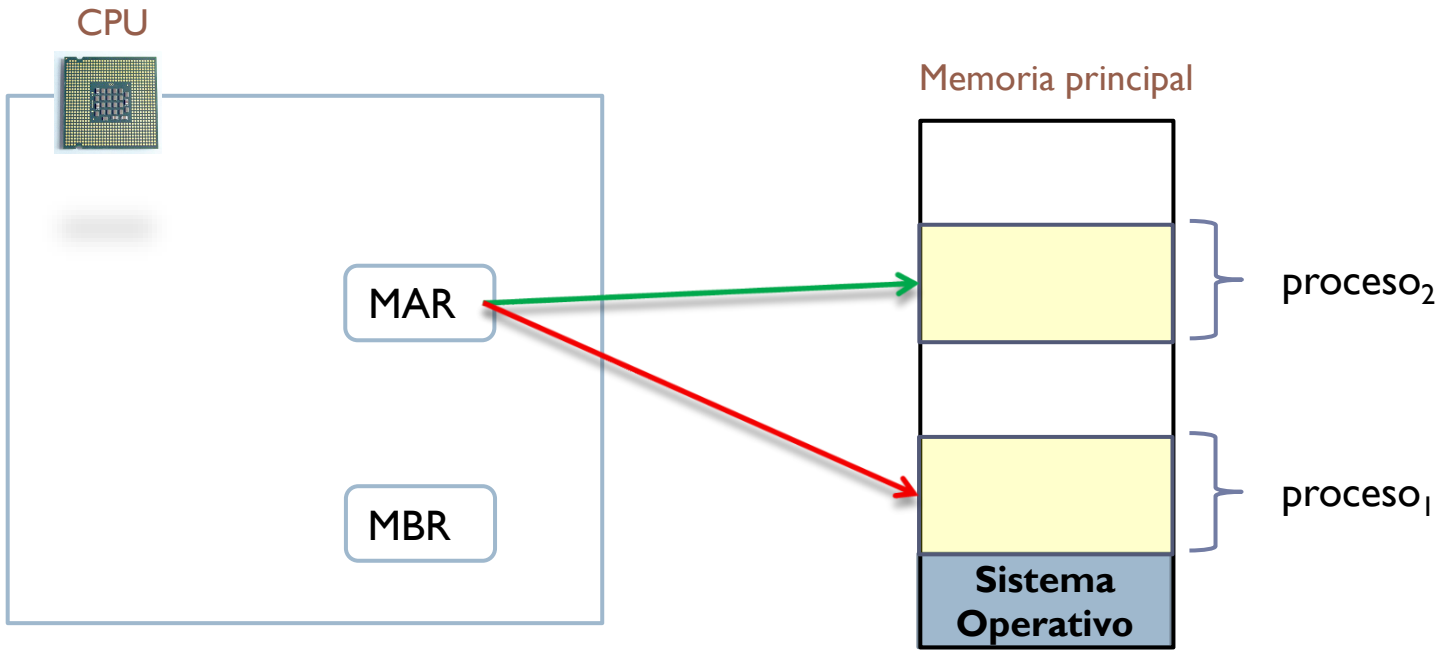


- En los sistemas sin memoria virtual, el programa se carga completamente en memoria para su ejecución.

- Principales problemas:
  - El tamaño de la imagen puede limitar su ejecución, o la de otros procesos.
  - La protección necesaria para que un proceso no tenga acceso a la imagen de otro proceso.
  - La necesidad de poder reubicar el programa en cualquier zona de memoria.

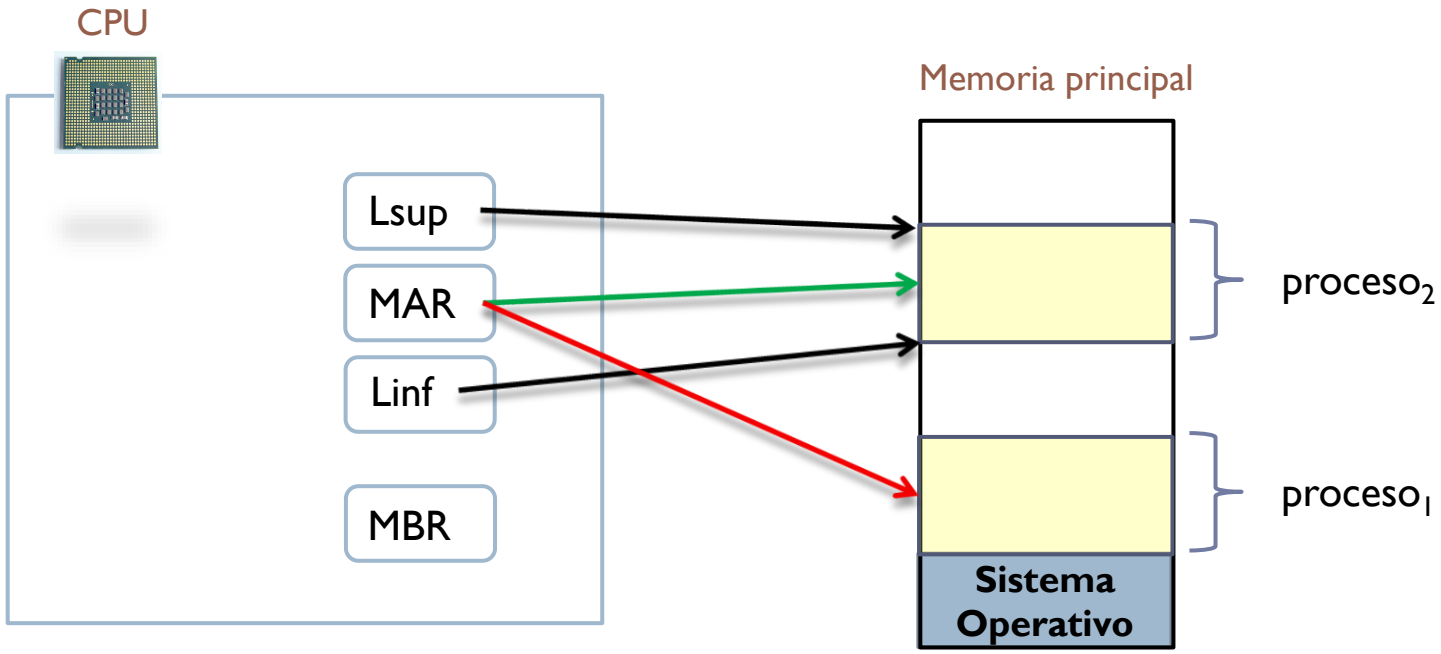
Problema de la protección de memoria

- Si hay varios procesos en memoria, es necesario asegurar que un programa no accede a la zona de memoria que otro tiene asignado.



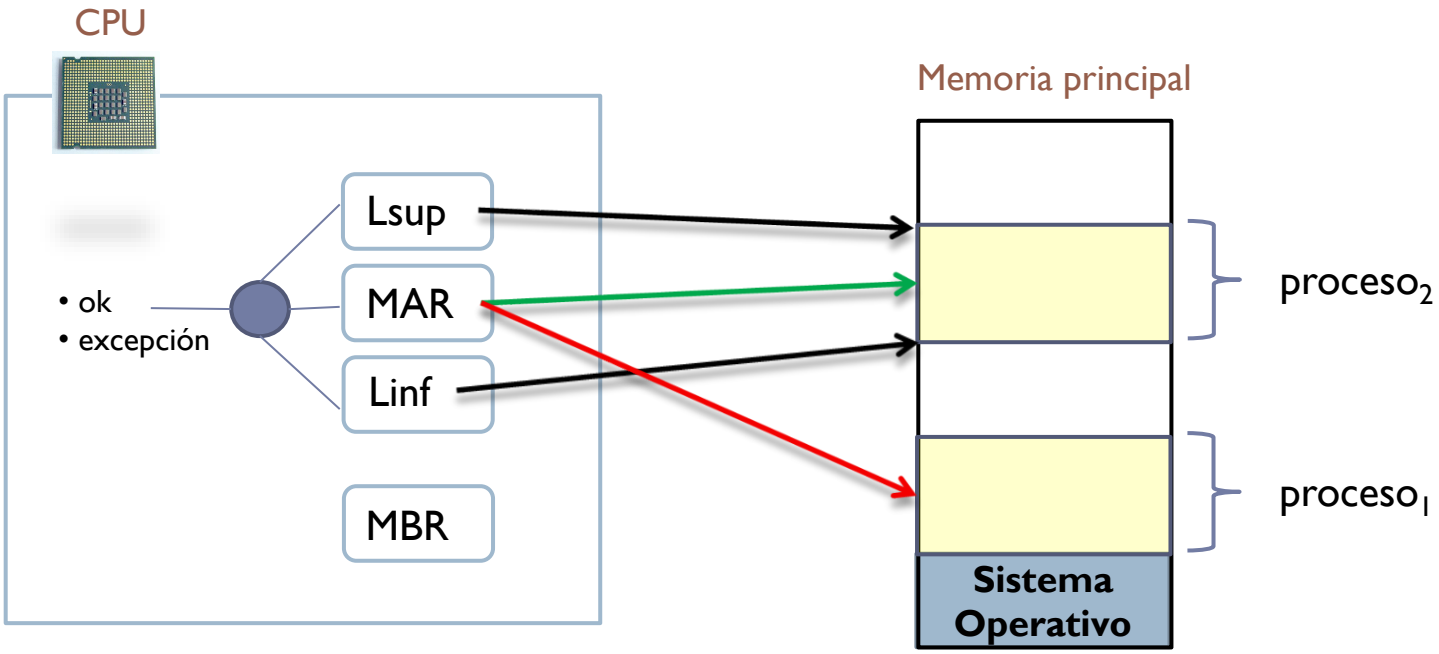
Problema de la protección de memoria

- Si hay varios procesos en memoria, es necesario asegurar que un programa no accede a la zona de memoria que otro tiene asignado.

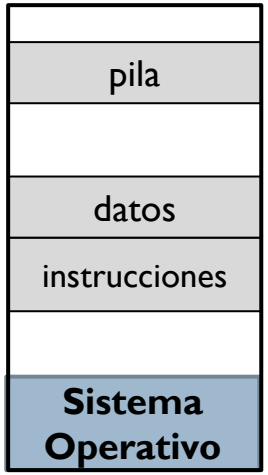


Problema de la protección de memoria

- Si hay varios procesos en memoria, es necesario asegurar que un programa no accede a la zona de memoria que otro tiene asignado.



Sistemas **sin** memoria virtual



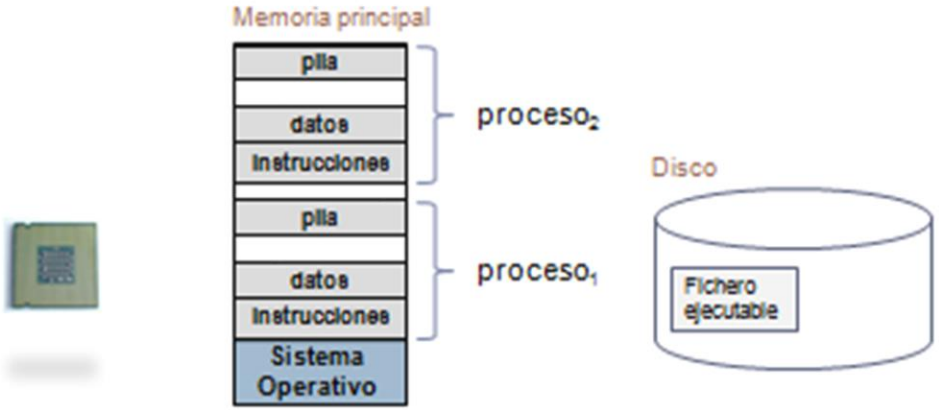
- En los sistemas sin memoria virtual, el programa se carga completamente en memoria para su ejecución.

- Principales problemas:
  - El tamaño de la imagen puede limitar su ejecución, o la de otros procesos.
  - La protección necesaria para que un proceso no tenga acceso a la imagen de otro proceso.
  - La necesidad de poder reubicar el programa en cualquier zona de memoria.

Necesidad de reubicación

Programa y proceso

- Proceso: programa en ejecución.
  - Es posible un mismo programa ejecutarlo varias veces (lo que da lugar a varios procesos)



19

ARCOS @UC3M
Alejandro Calderón Mateos

Necesidad de reubicación

```
int vector[1024];

int main ( void ) {
   int i;
   for (i = 0; i<1024; i++)
        vector[i] = i;
}
```

```
.data
  vector: .space 4*1024

.text
  .globl main

  main: li $t0 0
        la $t1 vector
        b2: bge $t0 1024 finb2
        mult $t2 $t0 4
        add  $t2 $t1 $t2
        sw   $t0 ($t2)
        add  $t0 $t0 1
        b b2
        finb2: jr $ra
```

Necesidad de reubicación

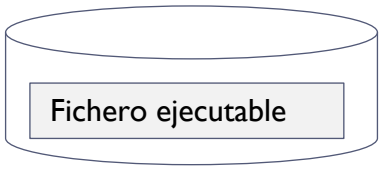
```
.data
  vector: .space 4*1024

.text
  .globl main

  main: li $t0 0
        la $t1 vector
        b2: bge $t0 1024 finb2
        mult $t2 $t0 4
        add  $t2 $t1 $t2
        sw   $t0 ($t2)
        add  $t0 $t0 1
        b b2
        finb2: jr $ra
```

Cabecera

- Tipo de ejecutable
- Tamaño inicial de pila
- Tamaño inicial de datos
- Tamaño inicial de código
- Etc.

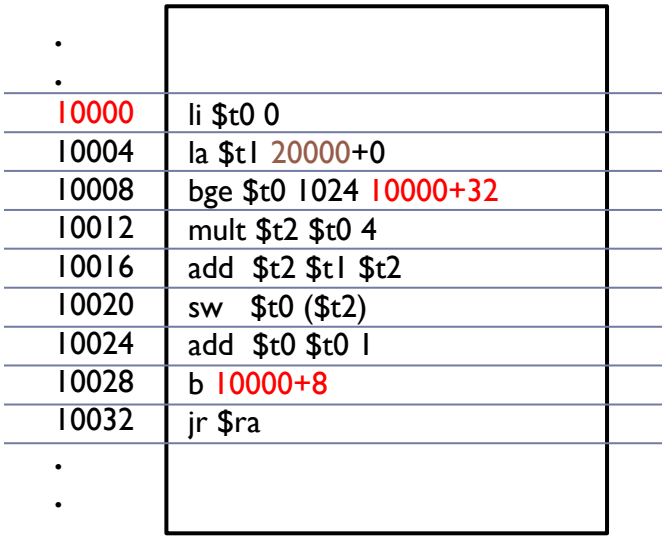
```
00: li $t0 0
04: la $t1 ini_datos+0
08: b2: bge $t0 1024 ini_codigo+32
12: mult $t2 $t0 4
16: add  $t2 $t1 $t2
20: sw   $t0 ($t2)
24: add  $t0 $t0 1
28: b ini_codigo+8
32: jr $ra
```

Disco



Necesidad de reubicación

Memoria principal

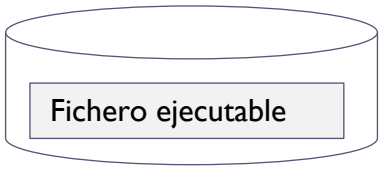


Cabecera

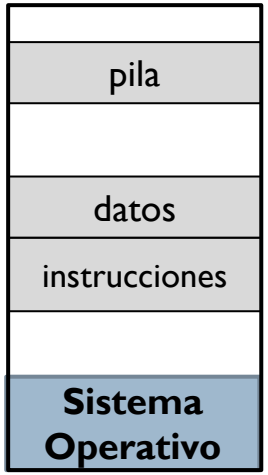
- Tipo de ejecutable
- Tamaño inicial de pila
- Tamaño inicial de datos
- Tamaño inicial de código
- Etc.

```
00: li $t0 0
04: la $t1 ini_datos+0
08: b2: bge $t0 1024 ini_codigo+32
12: mult $t2 $t0 4
16: add  $t2 $t1 $t2
20: sw   $t0 ($t2)
24: add  $t0 $t0 1
28: b ini_codigo+8
32: jr $ra
```

Disco



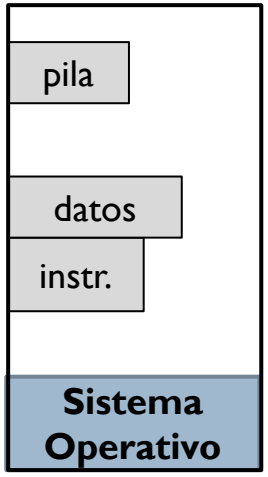
Sistemas **sin** memoria virtual



- En los sistemas sin memoria virtual, el programa se carga completamente en memoria para su ejecución.

- Principales problemas:
  - El tamaño de la imagen puede limitar su ejecución, o la de otros procesos.
  - La protección necesaria para que un proceso no tenga acceso a la imagen de otro proceso.
  - La necesidad de poder reubicar el programa en cualquier zona de memoria.

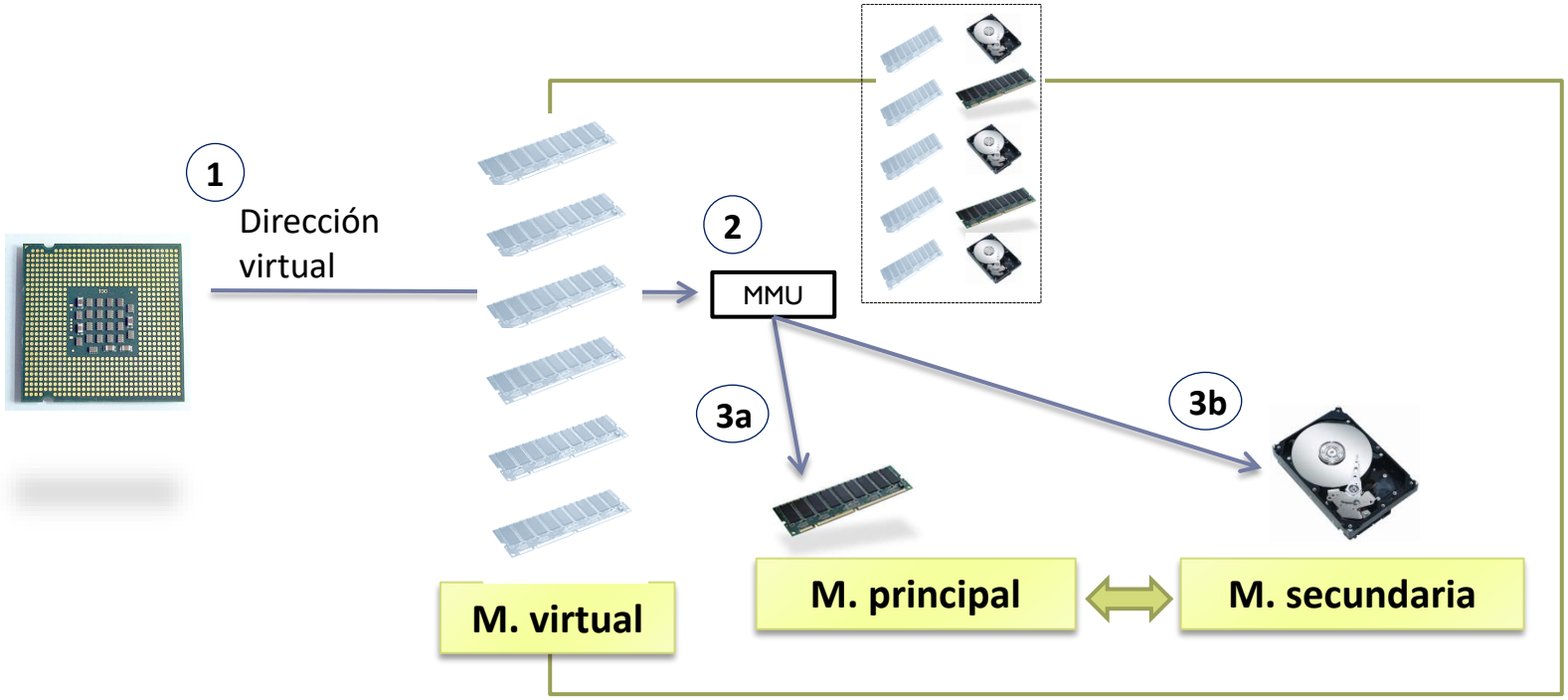
Sistemas **con** memoria virtual



- Los programas se carga parcialmente en memoria principal para su ejecución:
  - Cuando se necesite una parte del mismo, se carga en memoria principal dicha parte
  - Cuando no se necesite, se mueve a memoria secundaria (ej.: disco duro)
- Principales ventajas:
  - Se puede ejecutar programas cuya imagen es mayor que la memoria principal disponible.
  - Se pueden ejecutar más programas a la vez.
  - Cada programa tiene su propio espacio.

Sistemas **con** memoria virtual

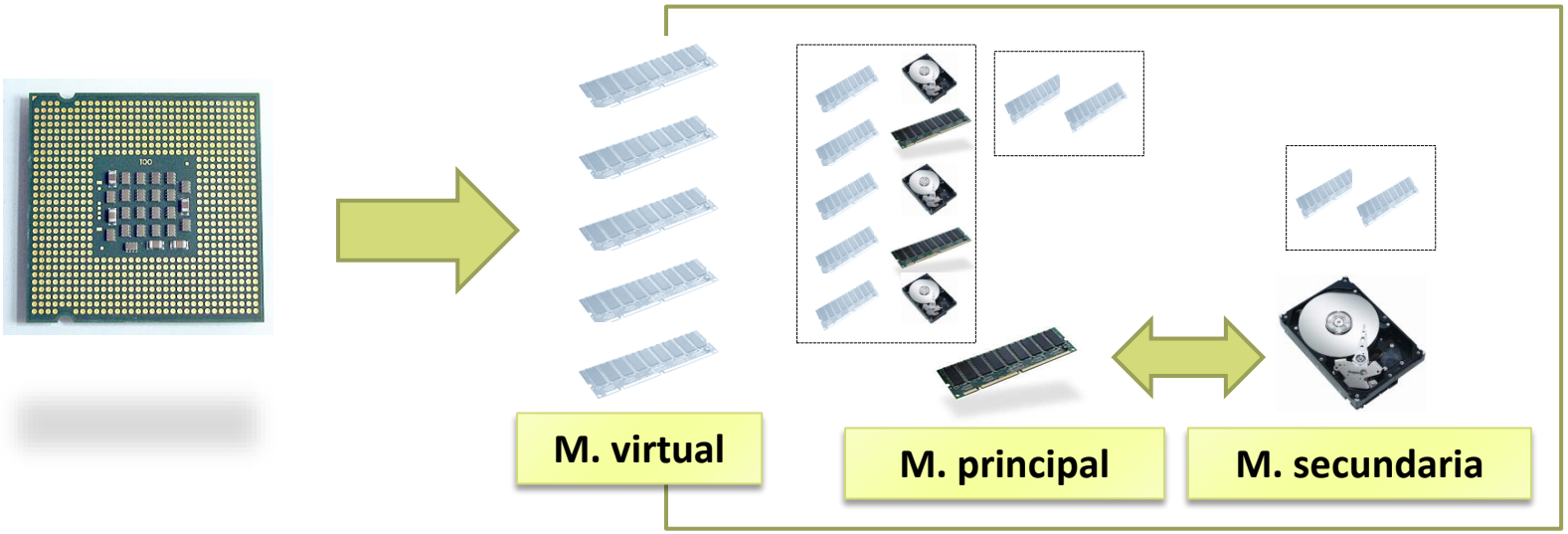
- Espacio virtual de direcciones
  - Los programas manejan un espacio virtual
  - MMU: Unidad de gestión de memoria



Contenidos

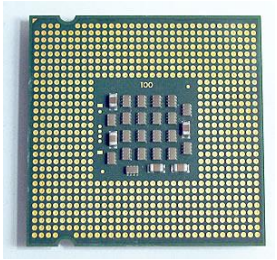
1. Memoria virtual
   - Definiciones iniciales
   - Motivación
   - **Funcionamiento general**
   - Memoria virtual paginada
   - Detalles de gestión

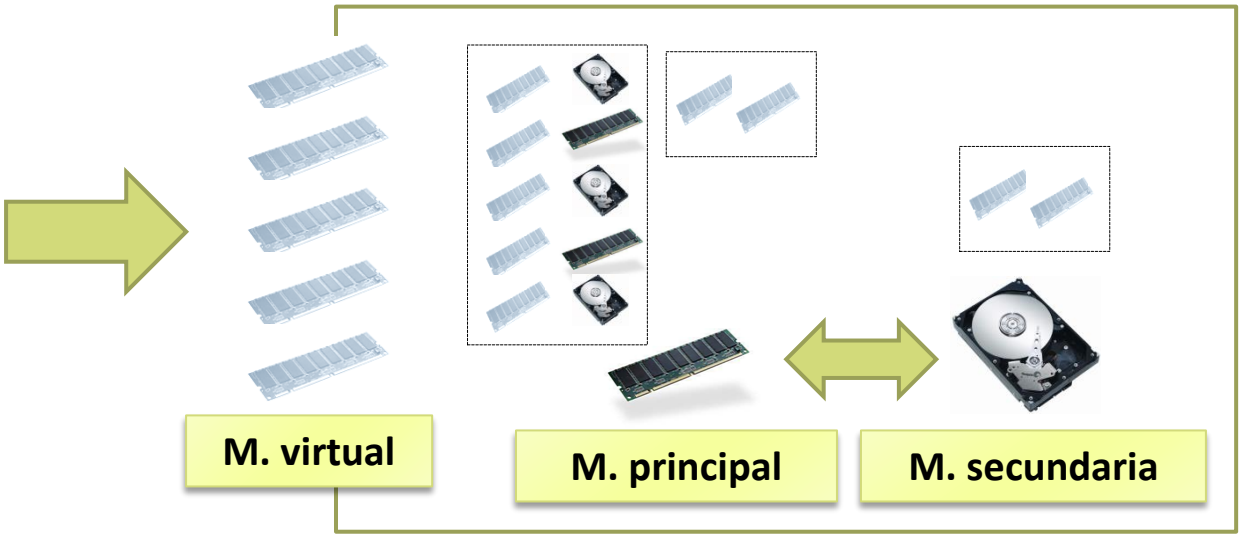
Introducción a memoria virtual



Introducción a memoria virtual

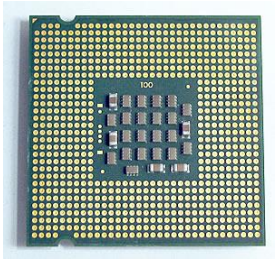
...
lw $t0 vector
...

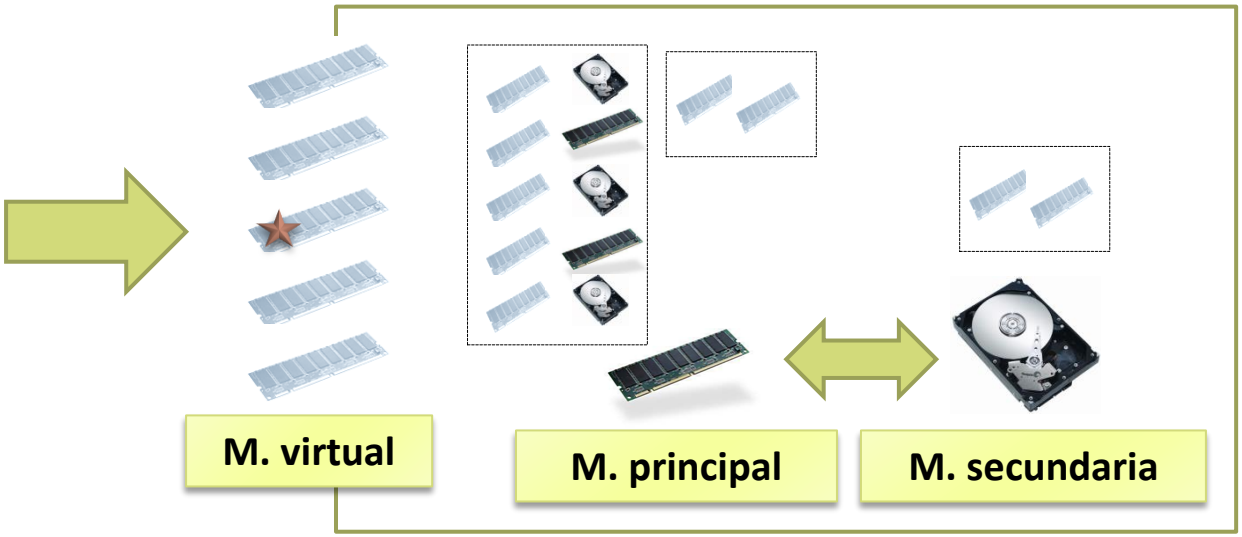




Introducción a memoria virtual

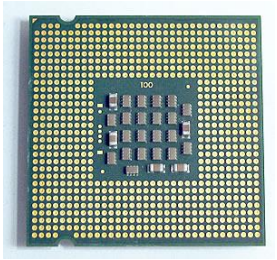
...
lw $t0 vector
...

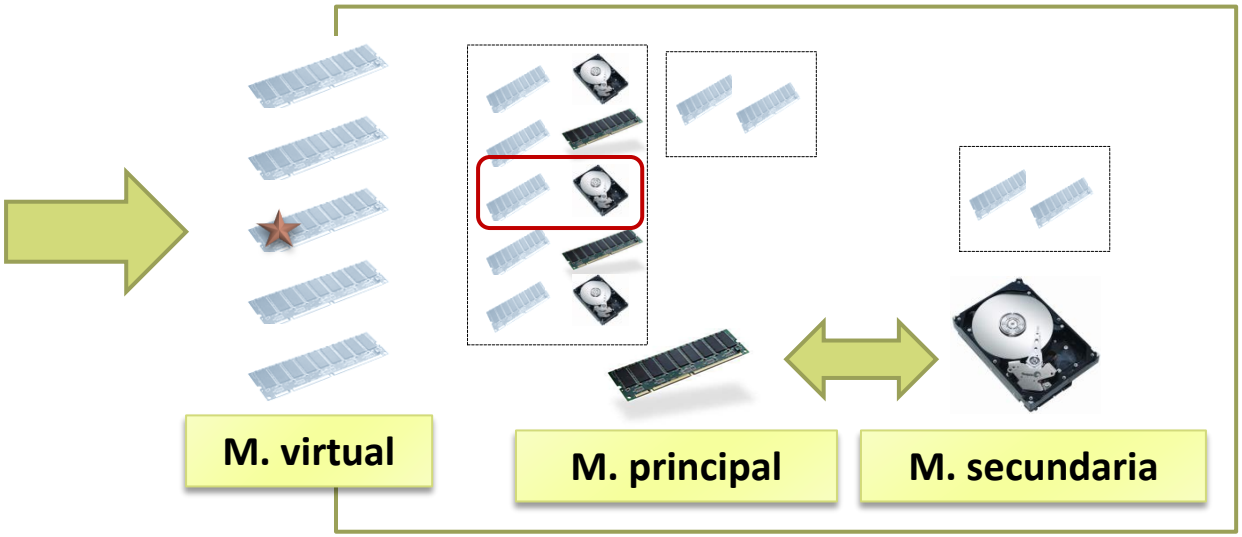




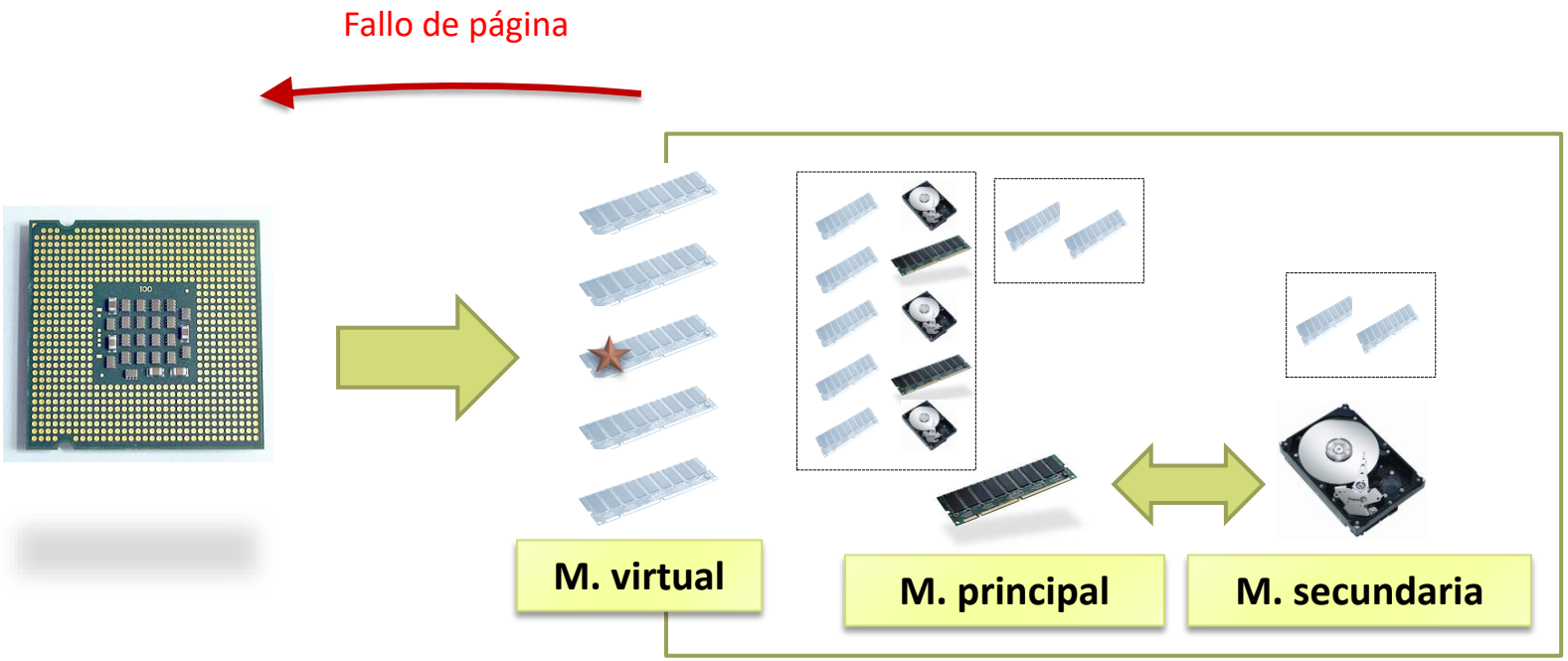
Introducción a memoria virtual

...
lw $t0 vector
...



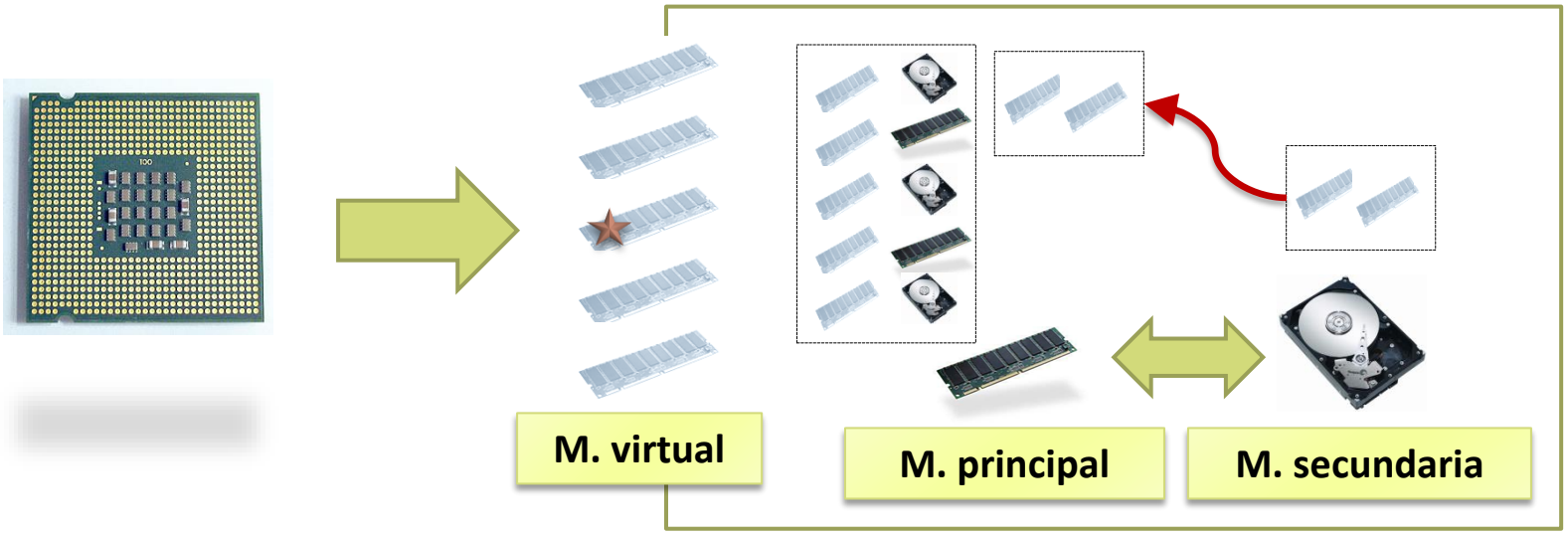


Introducción a memoria virtual



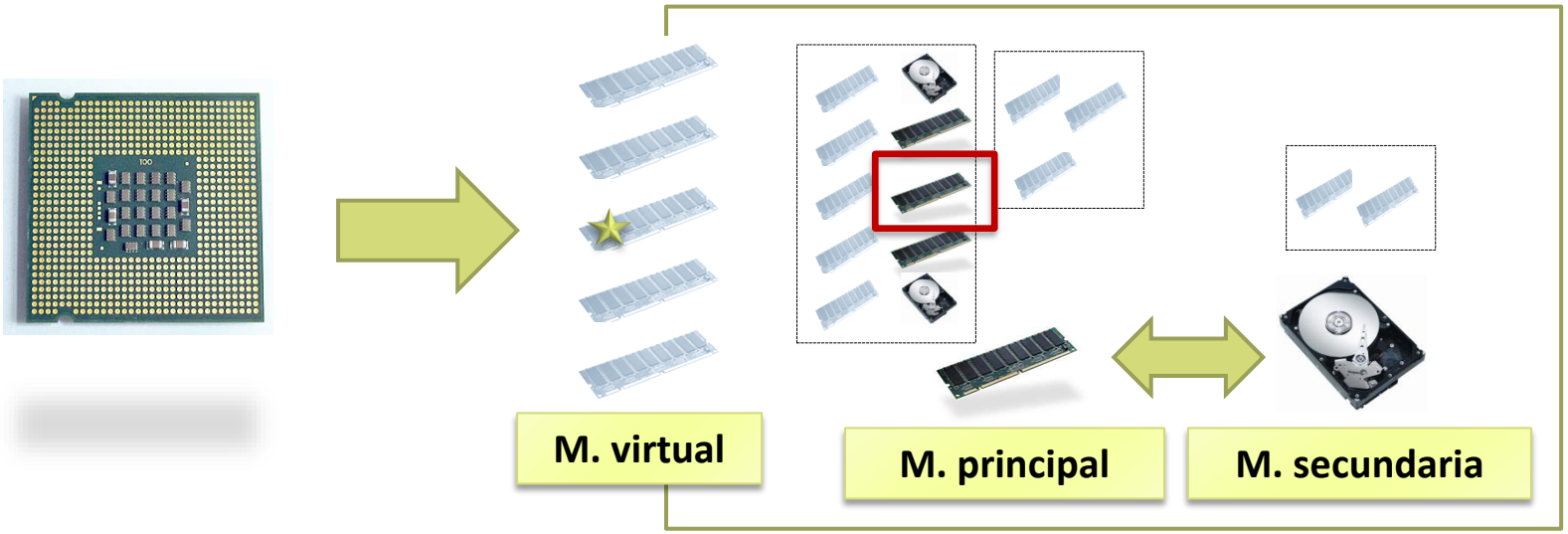
- El fallo de página es una excepción que provoca que el procesador ejecute la rutina de tratamiento asociada.
- Está implementada en el sistema operativo.

Introducción a memoria virtual



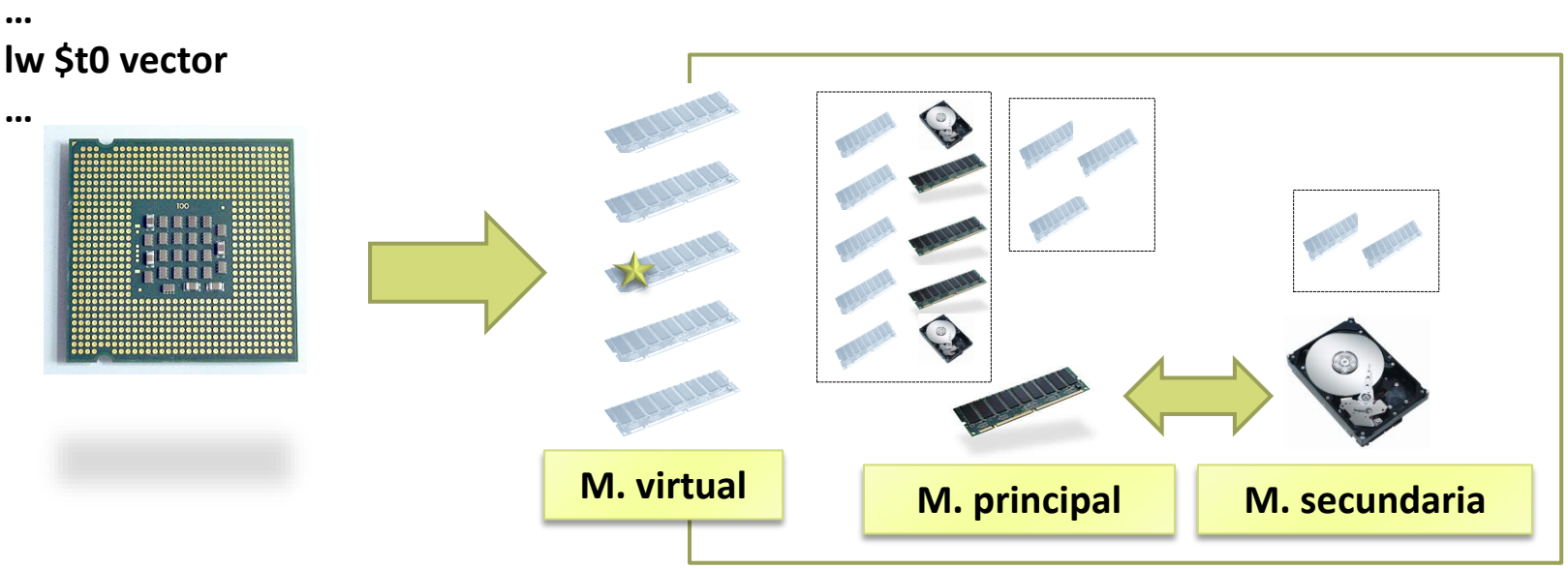
- El sistema operativo transfiere el 'bloque' solicitado a memoria principal y actualiza la tabla de 'bloques'

Introducción a memoria virtual



- El sistema operativo transfiere el 'bloque' solicitado a memoria principal y actualiza la tabla de 'bloques'

Introducción a memoria virtual

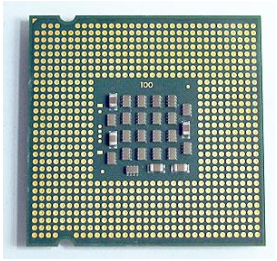


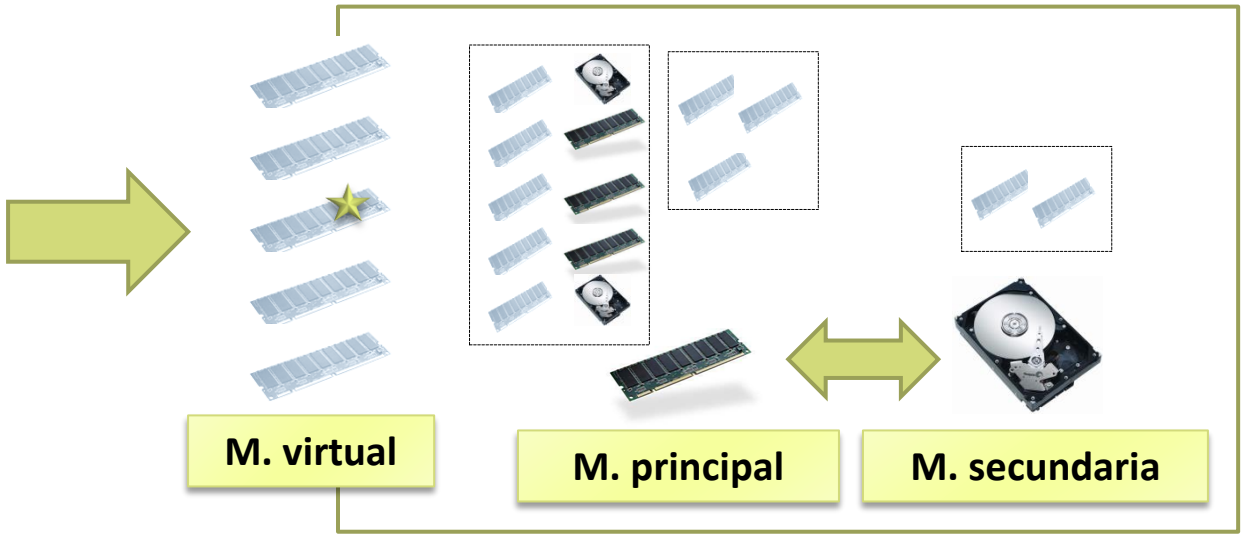
- Se reanuda la ejecución de la instrucción que provocó el fallo.

Introducción a memoria virtual

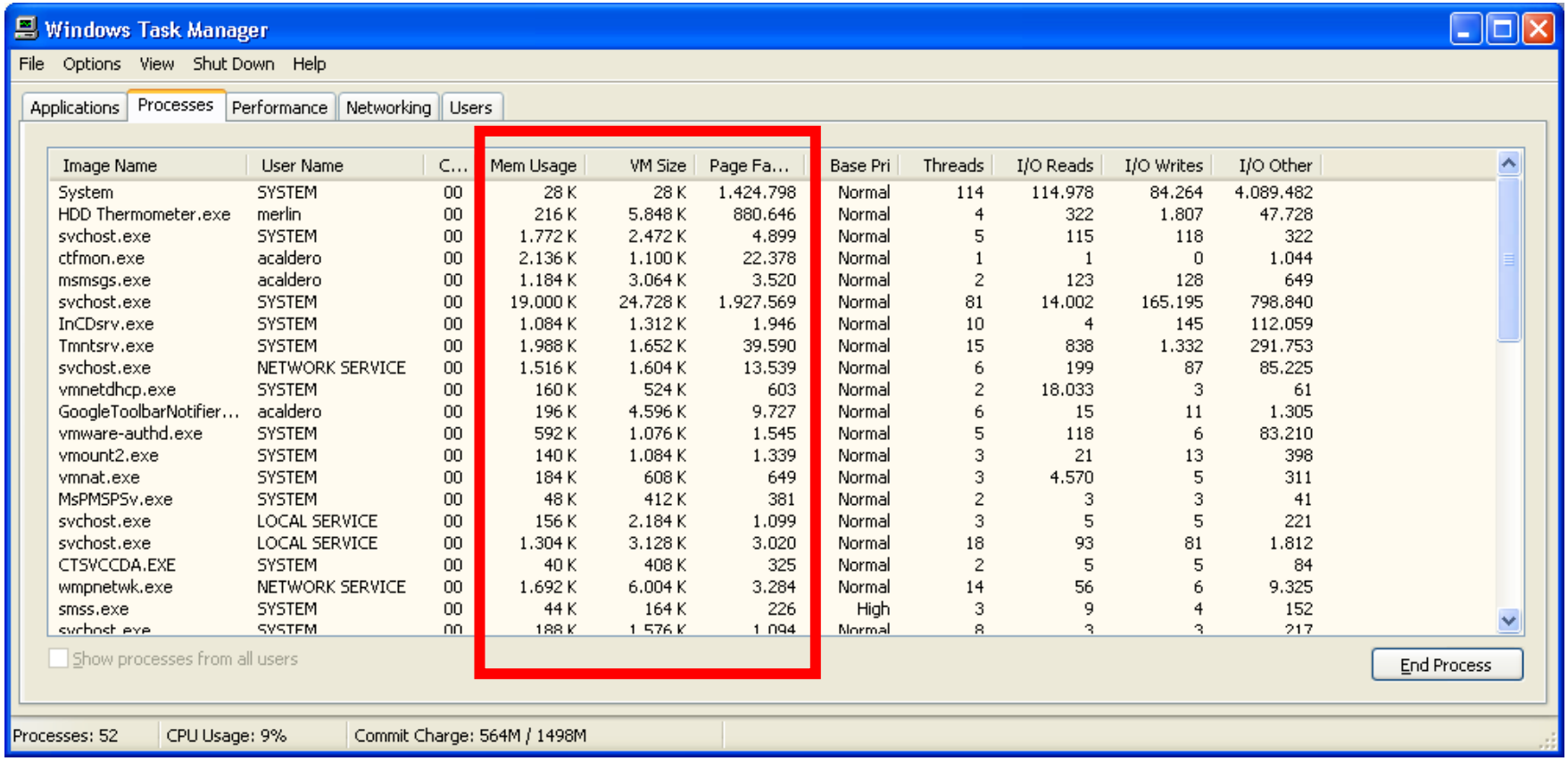
Acierto

...
lw $t0 vector+4
...

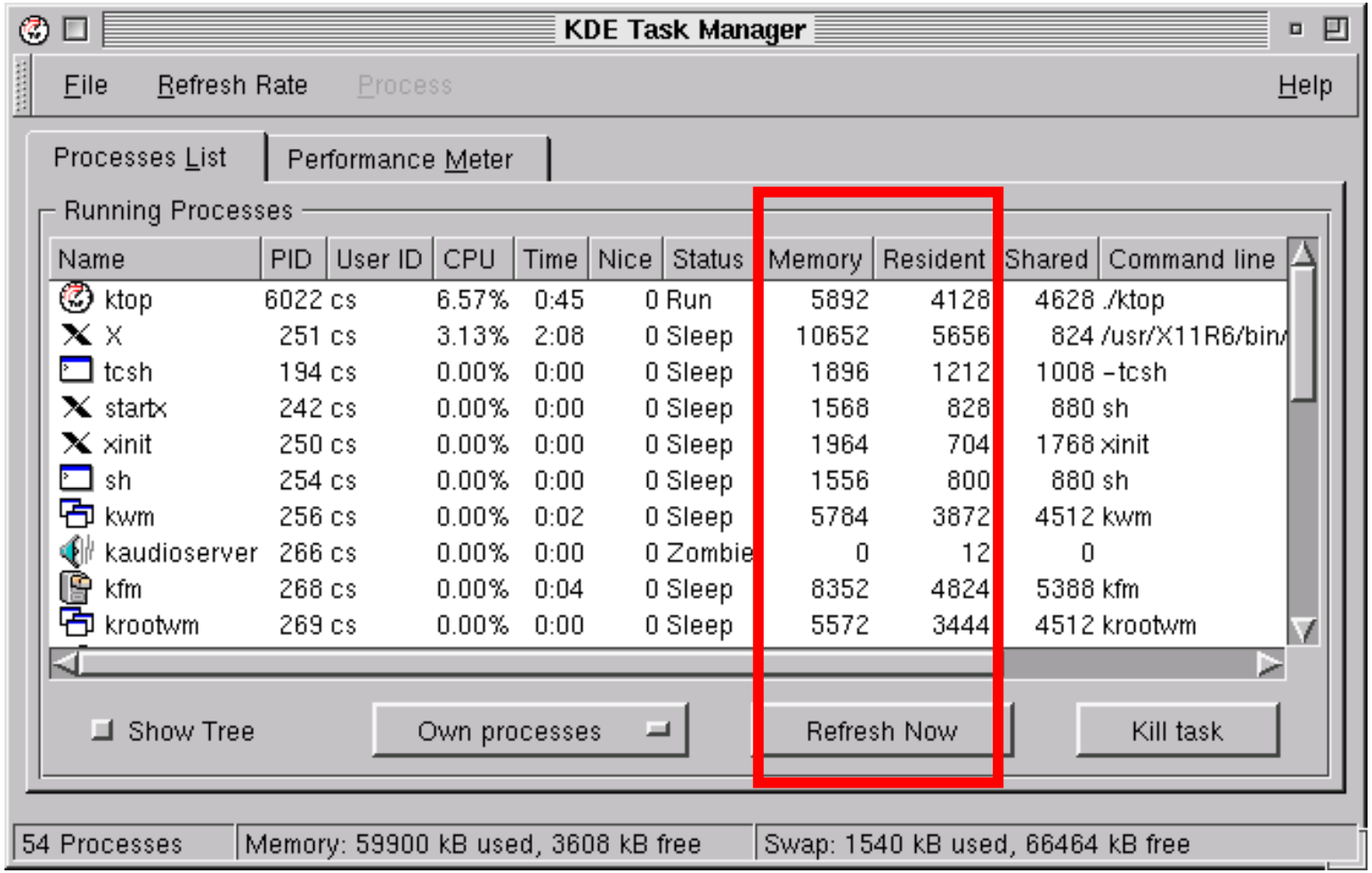




Memoria virtual: windows



Memoria virtual: linux



Contenidos

1. Memoria virtual
   - Definiciones iniciales
   - Motivación
   - Funcionamiento general
   - **Memoria virtual paginada**
   - Detalles de gestión

Ejemplo

32 bits

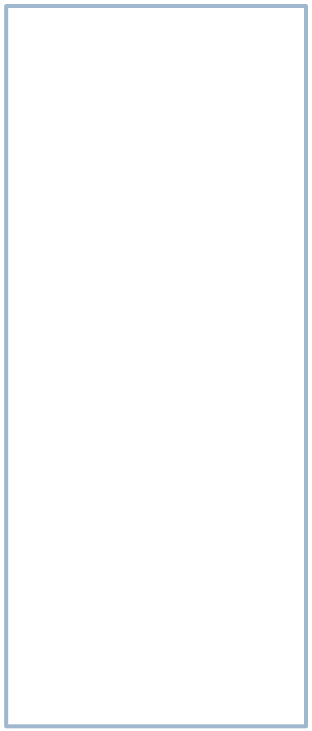
0110100010010l0

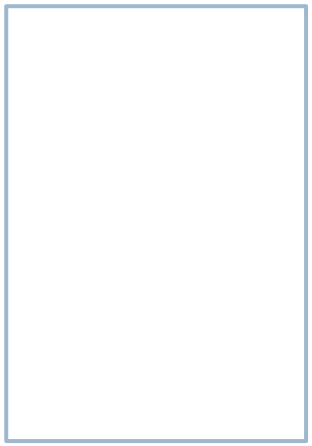
Dirección virtual

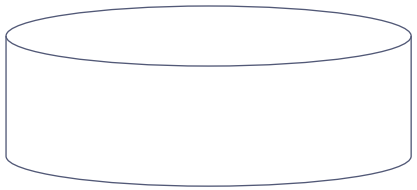
22 bits

1111100100000

Dirección física

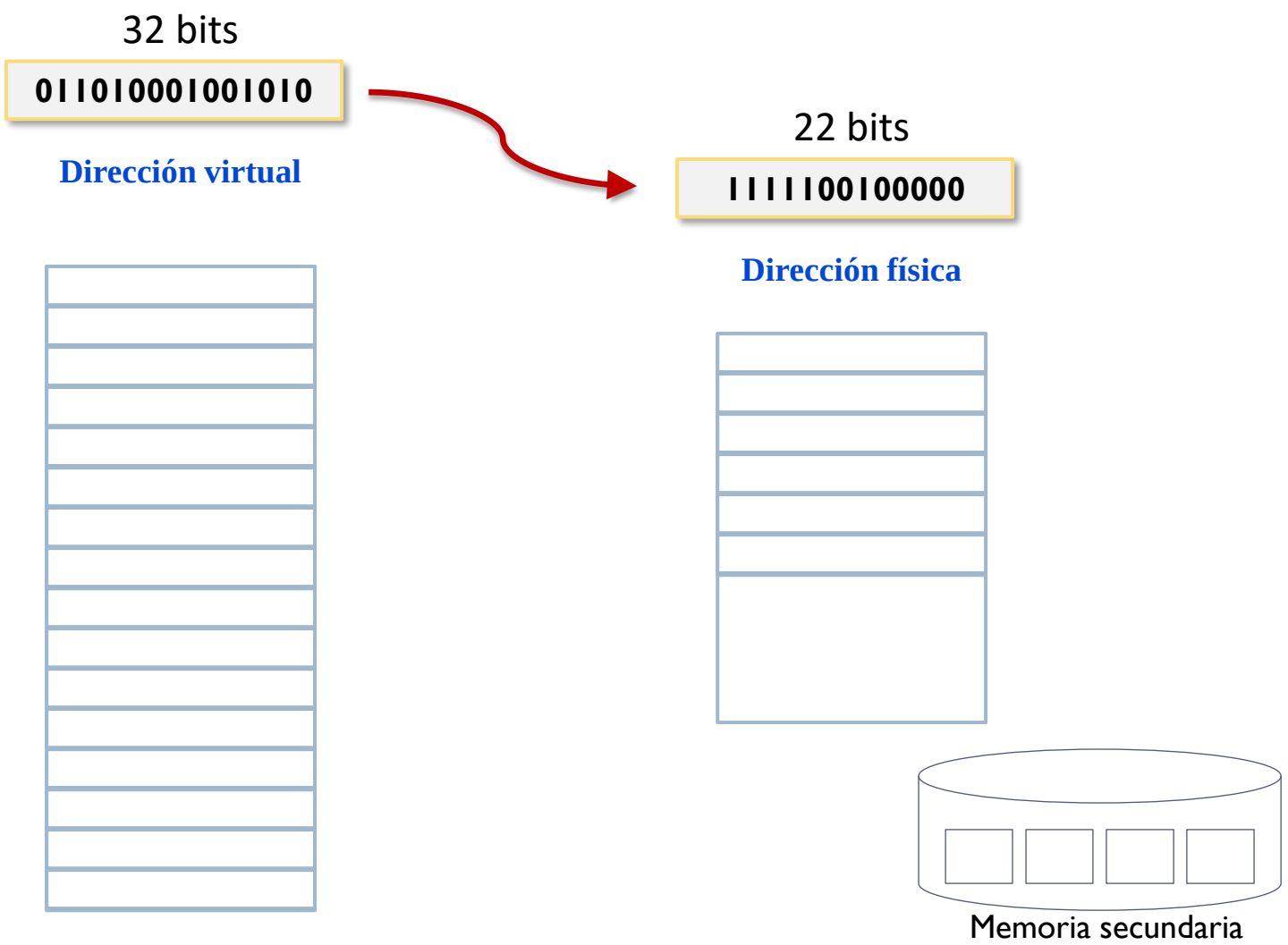






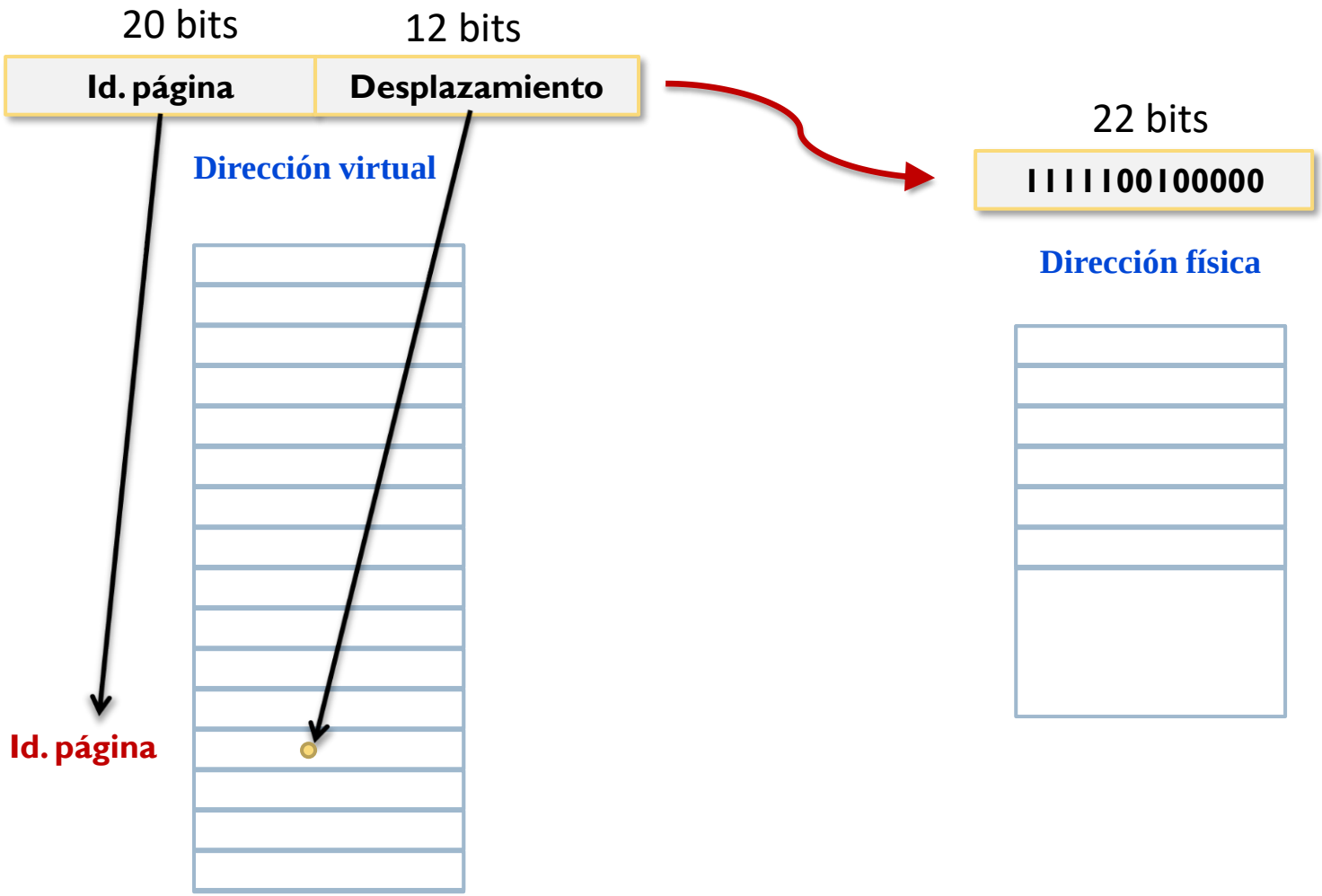
Memoria secundaria

Ejemplo



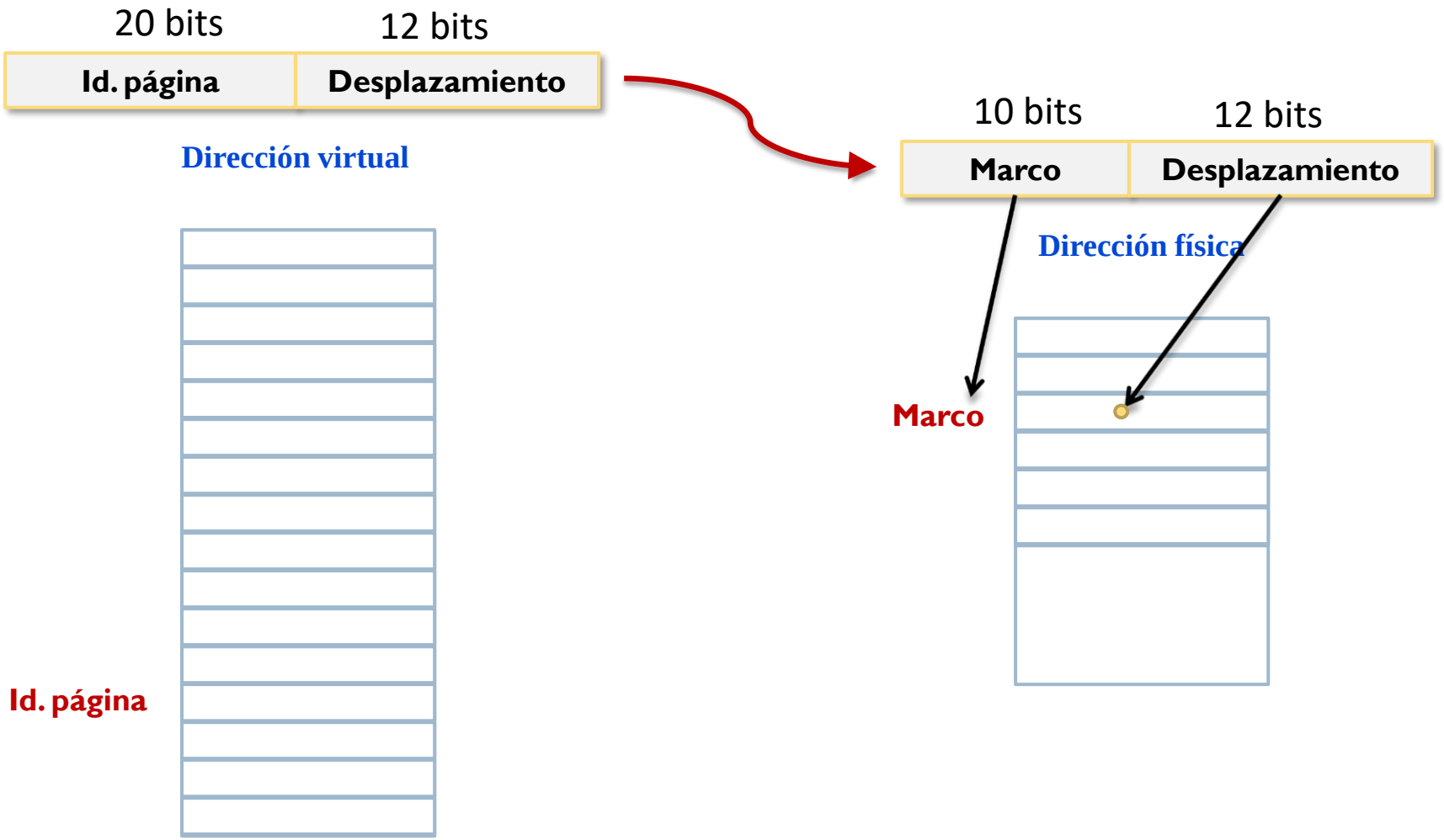
División en bloques del mismo tamaño -> páginas

Ejemplo



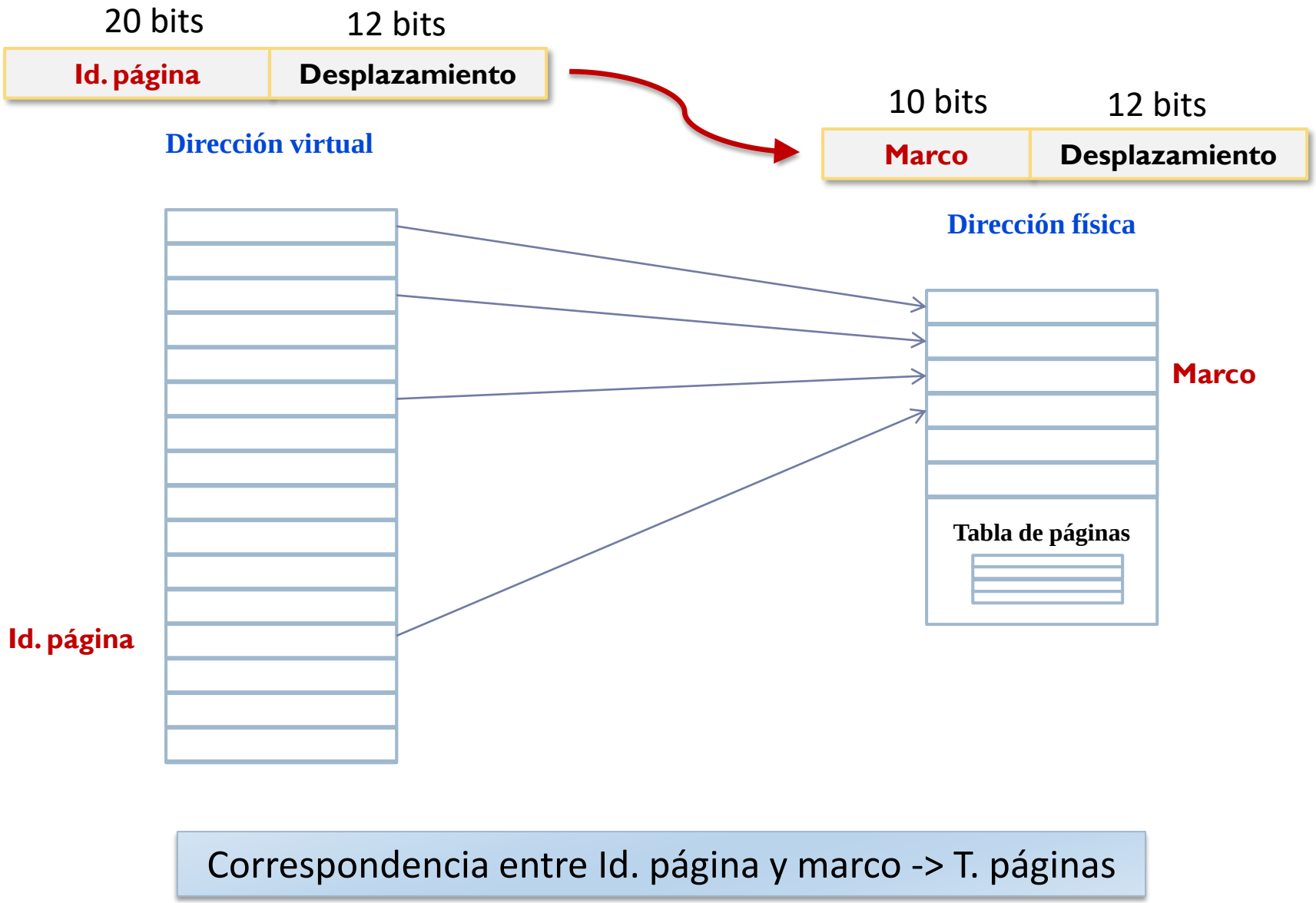
División en bloques del mismo tamaño -> páginas

Ejemplo



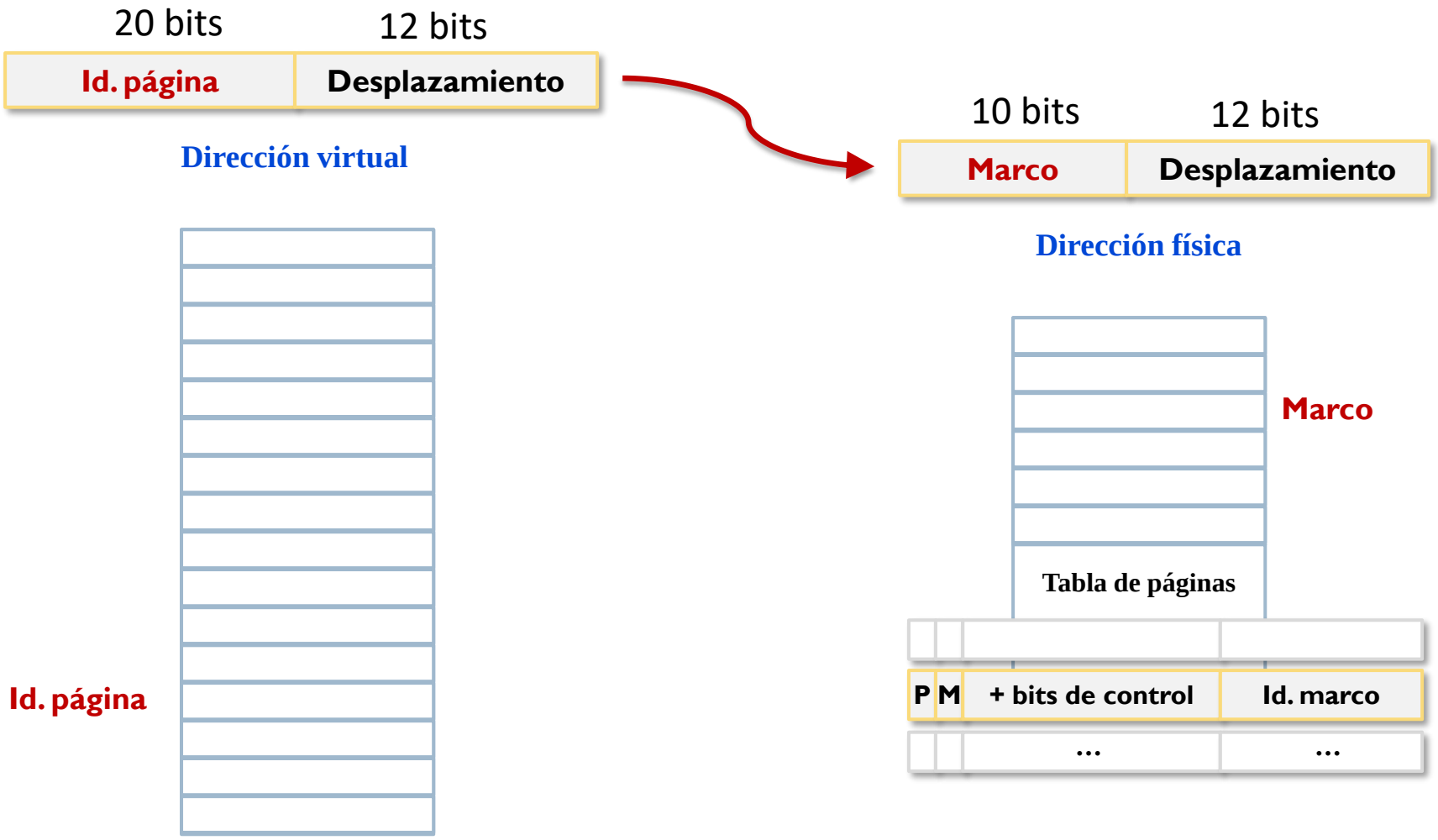
División en bloques del mismo tamaño -> páginas

Ejemplo



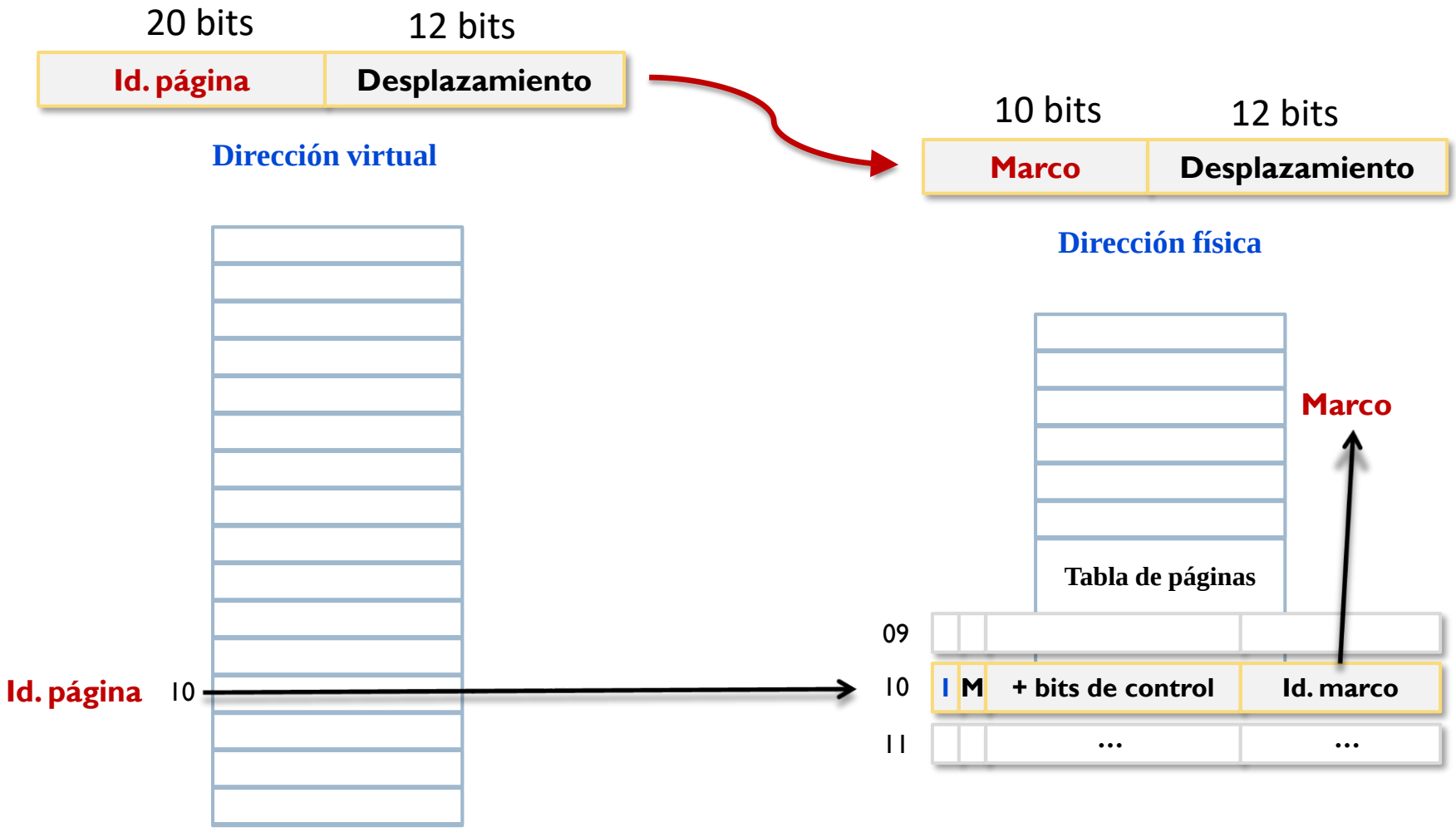
Correspondencia entre Id. página y marco -> T. páginas

Ejemplo

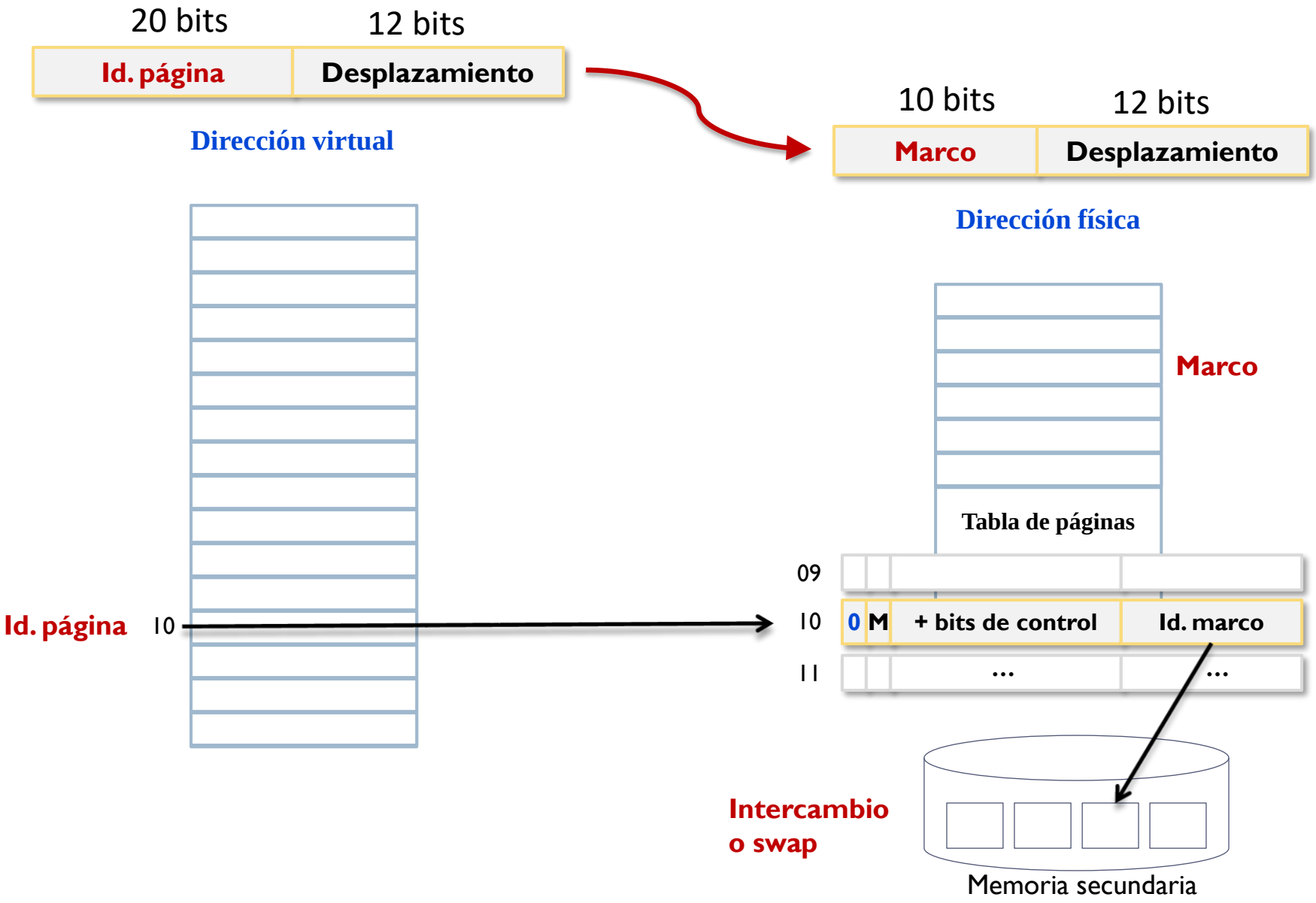


Correspondencia entre Id. página y marco -> T. páginas

Ejemplo


20 bits
12 bits
Id. página
Desplazamiento
Dirección virtual
10 bits
12 bits
Marco
Desplazamiento
Dirección física
Marco
Tabla de páginas
Id. página
10
09
10
11
I
M
+ bits de control
Id. marco
...
...

Ejemplo


20 bits
12 bits
Id. página
Desplazamiento
Dirección virtual
10 bits
12 bits
Marco
Desplazamiento
Dirección física
Marco
Tabla de páginas
Id. página
10
09
10
11
0
M
+ bits de control
Id. marco
...
...
Intercambio o swap
Memoria secundaria

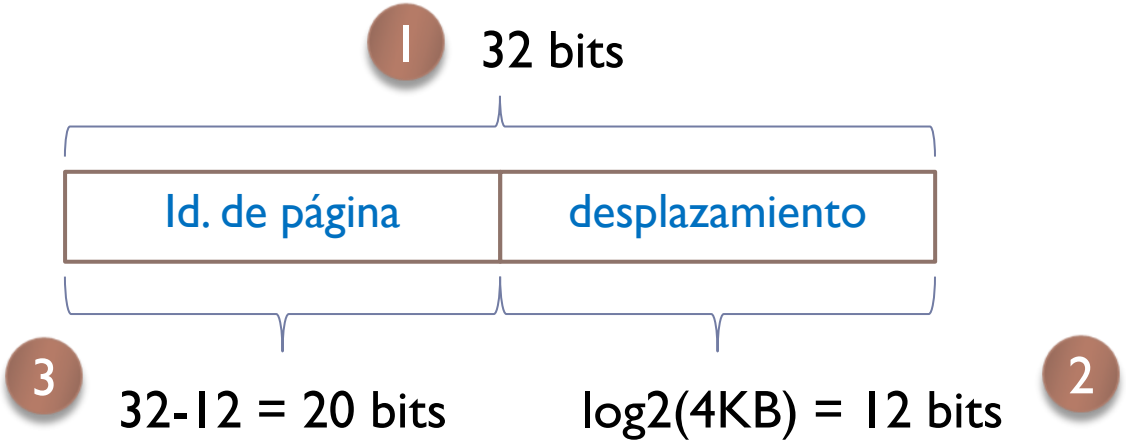
Ejercicio

Sea un computador con direcciones virtuales de 32 bits y una memoria principal de 512 MB, que emplea páginas de 4 KB.

- Se pide:
  a) Indique el formato de la dirección virtual y el número de marcos de página.

Ejercicio (sol.)

- Formato de la dirección virtual:



- Número de marcos de página:

$$\text{Tamaño de M.P.} \over \text{Tamaño de página} = \frac{512\ MB}{4\ KB} = \frac{512 * 2^{20}}{4 * 2^{10}} = 128 * 2^{10}$$

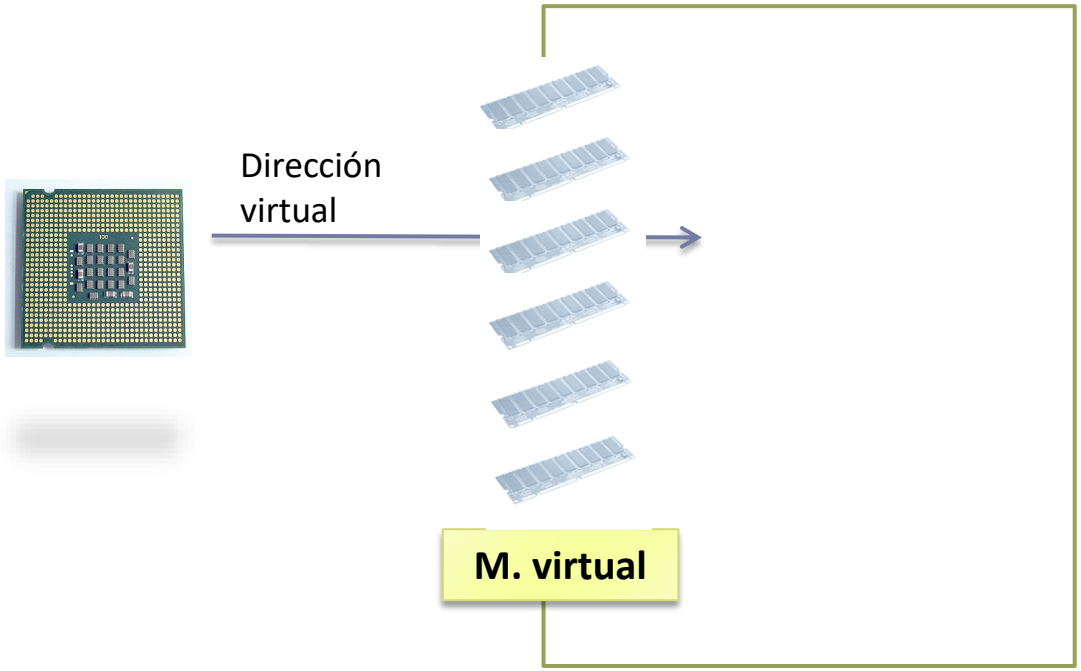
Ejercicio

Un computador que direcciona la memoria por byte emplea direcciones virtuales de 32 bits.
Cada entrada de la tabla de páginas requiere de 32 bits.
El sistema emplea páginas de 4 KB.

- Se pide:
  a) ¿Cuál es el espacio de memoria direccionable por un programa en ejecución?
  b) ¿Cuál es el máximo tamaño de la tabla de páginas en este computador?

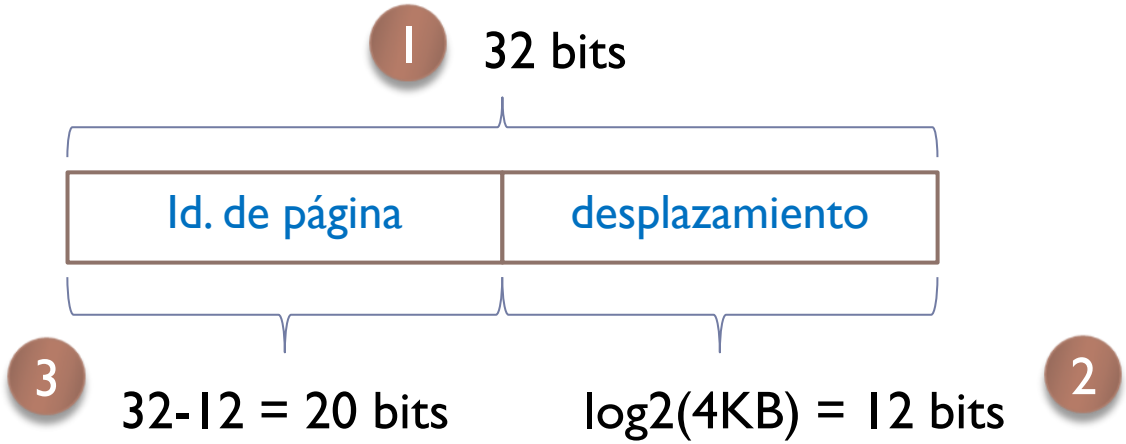
Ejercicio (sol.)

- El espacio de memoria direccionable por un programa en ejecución está determinado por el número de bits de la dirección virtual:
  - 2^{32} = 4 GB



Ejercicio (sol.)

- El tamaño de la tabla de páginas dependerá del máximo número de marcos de páginas y del tamaño de cada entrada de la tabla:
  - 2^{20} * 4 bytes (32 bits) = 4 MB



4 Si hay tanta memoria principal como memoria virtual, los identificadores de marco de página tendrán también 20 bits

Ejercicio

Sea un computador con direcciones virtuales de 32 bits y páginas de 4 KB. En este computador se ejecuta un programa cuya tabla de páginas es:

P	M	Perm.	Marco/ Bloque
0	0	R	1036
1	0	R	4097
0	0	W	3000
0	0	W	7190
0	0	W	3200
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	W	2400
0	0	W	3000

- Se pide:
  a) Tamaño que ocupa la imagen de memoria del programa
  b) Si la primera dirección virtual del programa es 0x00000000, indique la última
  c) Dadas las siguientes direcciones virtuales, indique si generan fallo de página o no:
     - 0x00001000
     - 0x0000101C
     - 0x00004000

Ejercicio (sol.)

P	M	Perm.	Marco/ Bloque
0	0	R	1036
1	0	R	4097
0	0	W	3000
0	0	W	7190
0	0	W	3200
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	W	2400
0	0	W	3000

- El tamaño que ocupa la imagen de memoria del programa dependerá del número de páginas total que tenga asignado y el tamaño de la página:
  - 7 * 4 KB = 28 KB

Ejercicio (sol.)

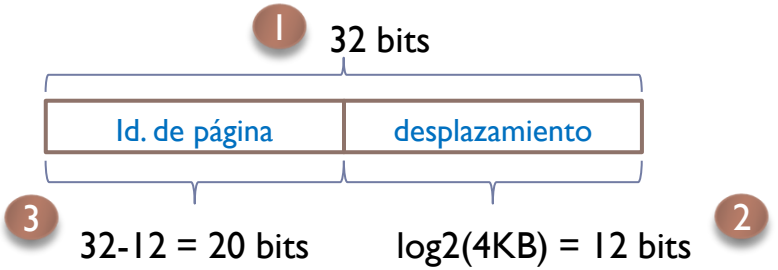
P	M	Perm.	Marco/ Bloque
0	0	R	1036
1	0	R	4097
0	0	W	3000
0	0	W	7190
0	0	W	3200
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	W	2400
0	0	W	3000

- Si el tamaño total del programa es de 28 KB y la primera dirección virtual es la 0x00000000,
  la última dirección será:
  - 28 * 1024 - 1

Ejercicio (sol.)

P	M	Perm.	Marco/ Bloque
0	0	R	1036
1	**0**	**R**	**4097**
0	0	W	3000
0	0	W	7190
0	**0**	**W**	**3200**
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	W	2400
0	0	W	3000

- Lo primero es conocer el formato de la dirección virtual

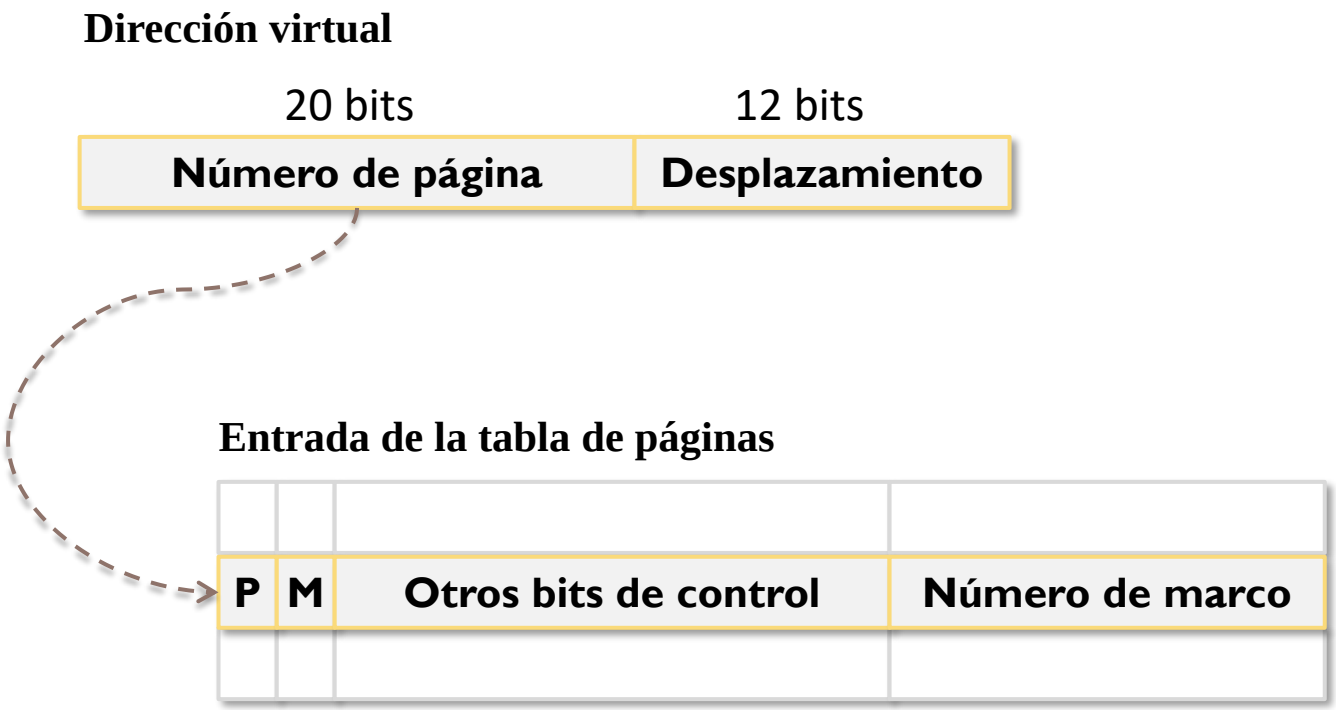


- Para cada dirección virtual, se extrae el identificador de página, se busca en la Tabla de páginas su entrada, y se ve si el bit de presente (P) está a 1:
  - 0x**00001**000 -> no
  - 0x**00001**01C -> no
  - 0x**00004**000 -> si

Contenidos

1. Memoria virtual
   - Definiciones iniciales
   - Motivación
   - Funcionamiento general
   - Memoria virtual paginada
   - **Detalles de gestión**
     - **Tabla de páginas**
     - TLB
     - Memoria virtual y memoria caché

Entradas de la tabla de páginas (formato típico)

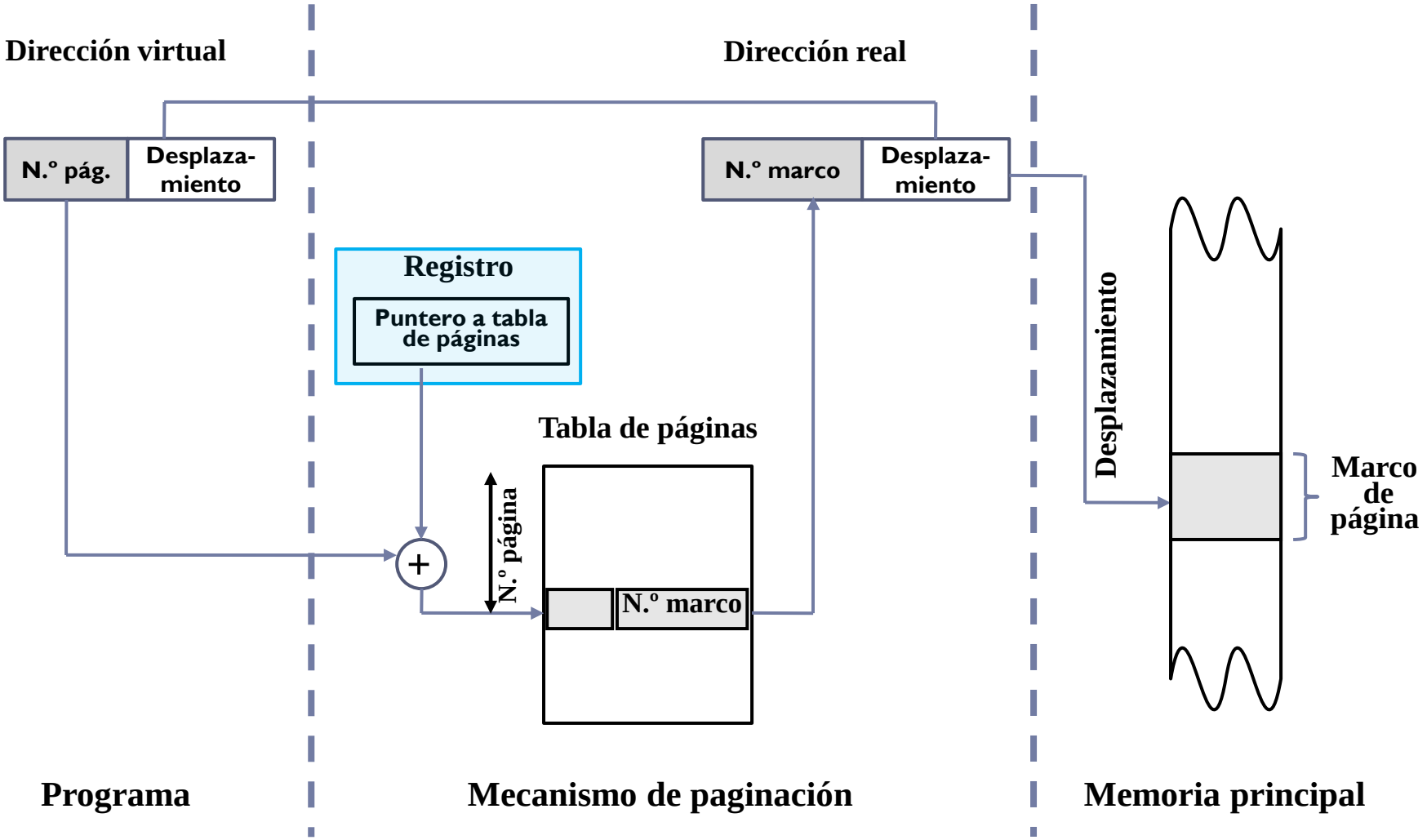


- Bit P: indica si está presente la página en M.P.
- Bit M: indica si ha sido modificada la página en M.P.
- Otros bits: protección (lectura, escritura, ejecución, etc.), gestión (cow, etc.)

Gestión de la tabla de páginas

- Inicialmente:
  - La crea el sistema operativo cuando se va a ejecutar el programa.
- Uso:
  - La consulta la MMU en la traducción.
- Actualización:
  - La modifica el sistema operativo en los fallos de página.

Traducción de direcciones (paginación)

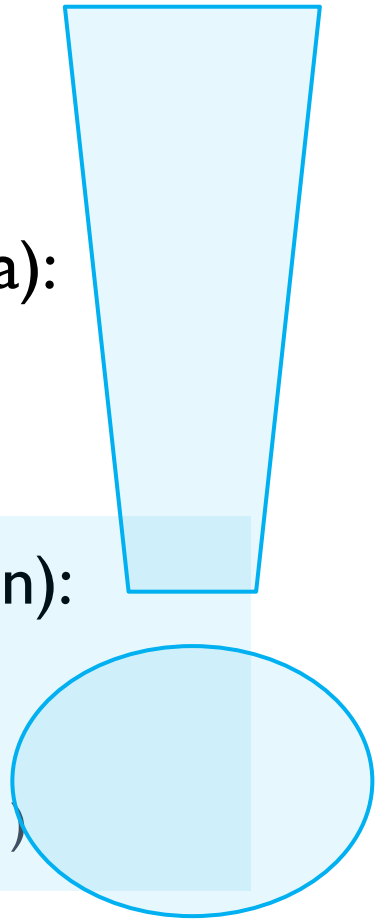


Movimiento de las páginas

- Inicialmente:
  - Página no residente se marca ausente
  - Se guarda dirección del bloque de *swap* que la contiene
- De M. secundaria a M. principal (por demanda):
  - Acceso a pág. no residente: Fallo de página
  - S.O. lee página de M. secundaria y la lleva a M. principal
- De M. principal a M. secundaria (por expulsión):
  - No hay espacio en M. principal para traer página
  - Se expulsa (reemplaza) una página residente
  - S.O. escribe página expulsada a M. secundaria (si bit M=1)

Movimiento de las páginas

- Inicialmente:
  - Página no residente se marca ausente
  - Se guarda dirección del bloque de *swap* que la contiene
- De M. secundaria a M. principal (por demanda):
  - Acceso a pág. no residente: Fallo de página
  - S.O. lee página de M. secundaria y la lleva a M. principal
- De M. principal a M. secundaria (por expulsión):
  - No hay espacio en M. principal para traer página
  - Se expulsa (reemplaza) una página residente
  - S.O. escribe página expulsada a M. secundaria (si bit M=1)



Políticas de **no** reemplazo

- Bloqueo de marcos:
  - Cuando un marco está bloqueado, la página cargada en ese marco no puede ser reemplazada.
- Ejemplos de cuándo se bloquea un marco:
  - La mayoría del núcleo del sistema operativo.
  - Estructuras de control.
  - Buffers de E/S.
- **El bloqueo se consigue asociando un bit de bloqueo a cada marco.**

Políticas de reemplazo

- Qué página se va a reemplazar.
- La página que se va a reemplazar tiene que ser la que tenga una menor posibilidad de ser referenciada en un futuro cercano.
- La mayoría de las políticas intentan predecir el comportamiento futuro en función del comportamiento pasado.
- Ejemplo de políticas: **LRU, FIFO, etc.**

Contenidos

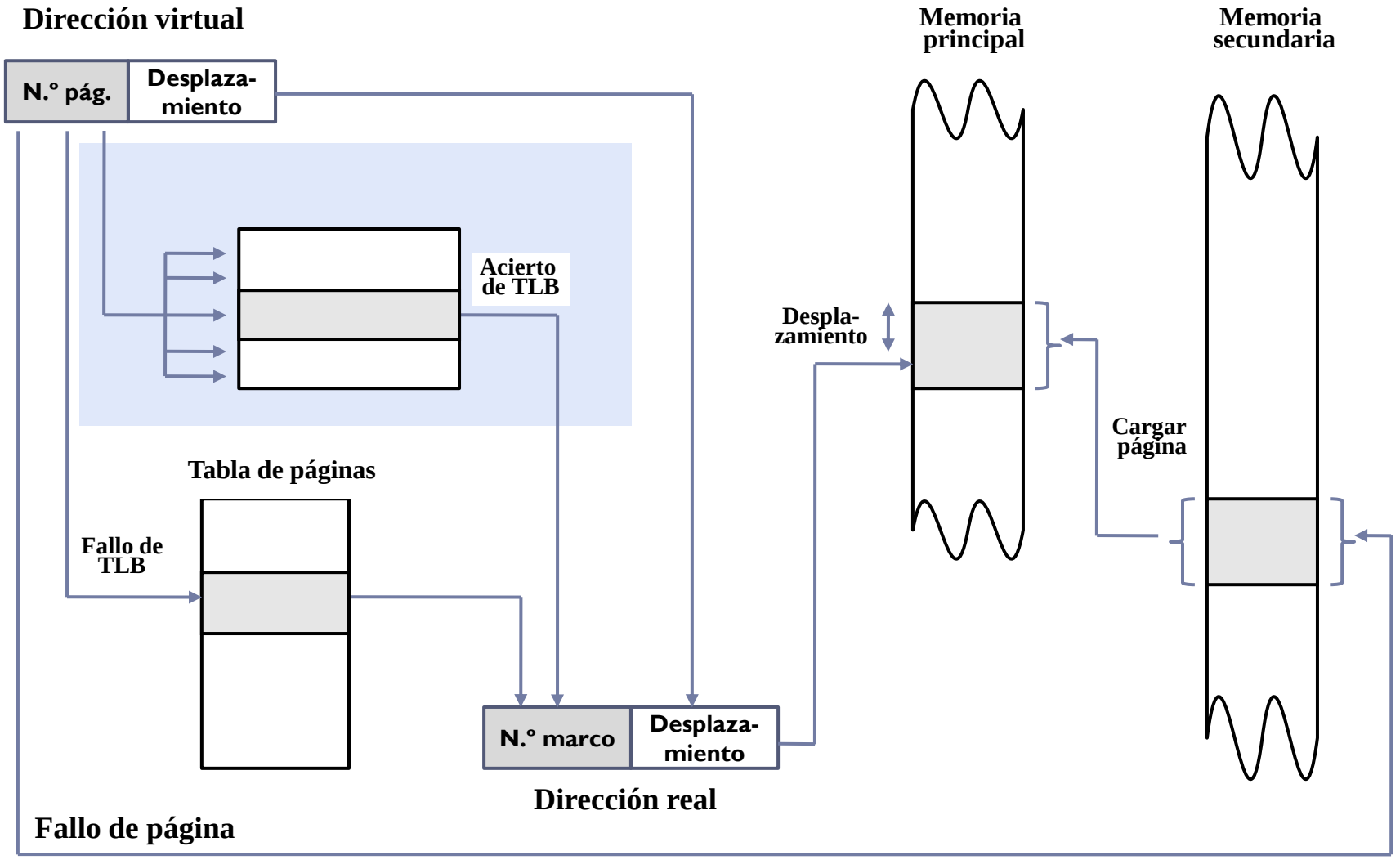
1. Memoria virtual
   - Definiciones iniciales
   - Motivación
   - Funcionamiento general
   - Memoria virtual paginada
   - **Detalles de gestión**
     - Tabla de páginas
     - **TLB**
     - Memoria virtual y memoria caché

Cache de traducciones

- Memoria virtual basado en tablas de páginas:
  - Problema: sobrecarga de acceso a memoria (2 accesos).
  - Solución: TLB.

- TLB: buffer de traducción adelantada:
  - Memoria caché asociativa que almacena las entradas de la tabla de página usadas más recientemente.
  - Permite acelerar el proceso de búsqueda del marco.

Traducción de direcciones (con TLB)



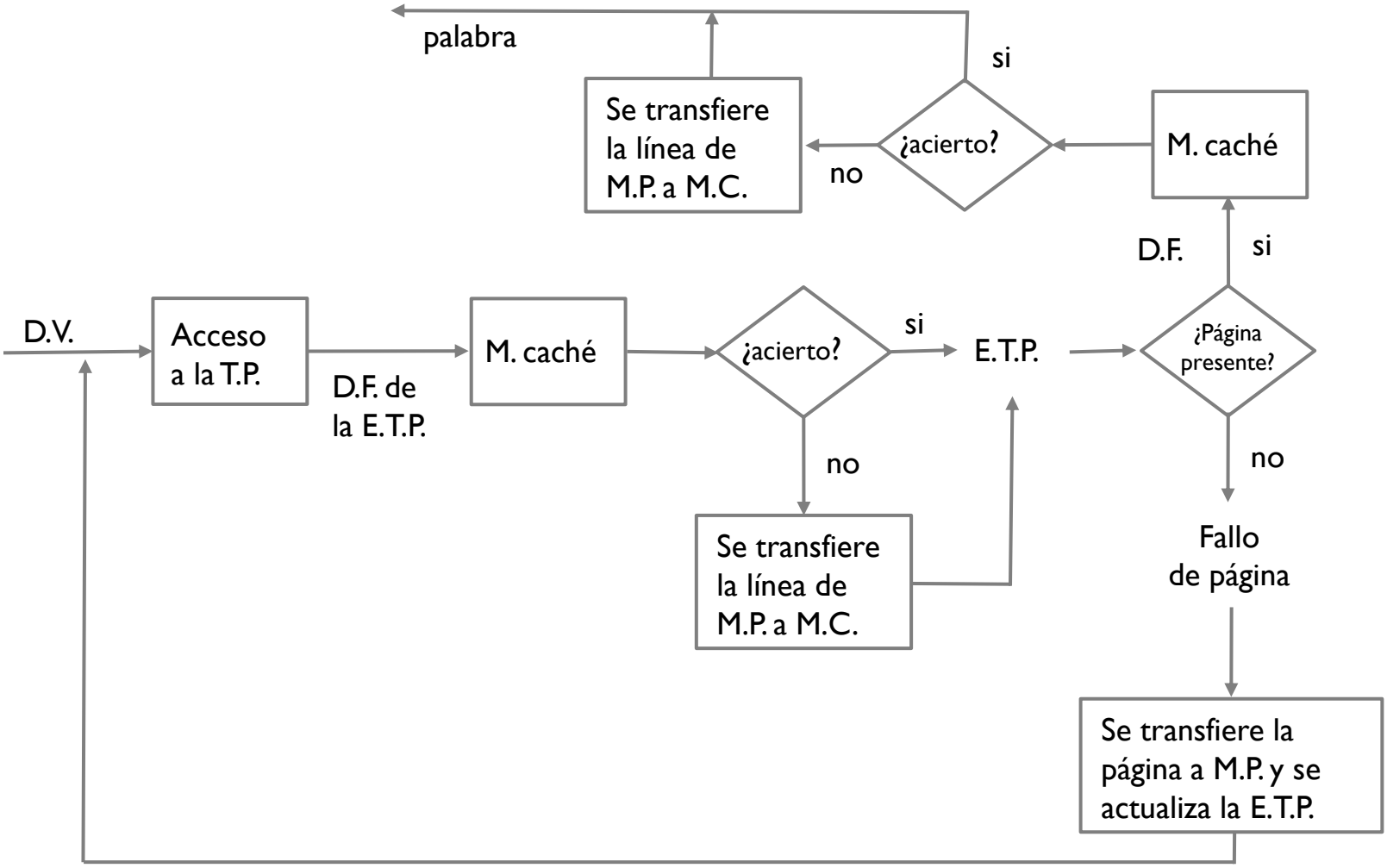
Contenidos

1. Memoria virtual
   - Definiciones iniciales
   - Motivación
   - Funcionamiento general
   - Memoria virtual paginada
   - **Detalles de gestión**
     - Tabla de páginas
     - TLB
     - **Memoria virtual y memoria caché**

Caché y memoria virtual

Caché	Memoria virtual
Acelerar el acceso	Incrementar el espacio direccionable
Transferencia por bloques o líneas.	Transferencia por páginas.
Bloques: 32-64B.	Páginas: 4-8 KB.
Traducción: Algoritmo de correspondencia.	Traducción: Totalmente asociativa.
Escritura inmediata o diferida.	Escritura diferida.

Caché y memoria virtual



Tema 5 (III)
Jerarquía de Memoria

Grupo ARCOS

Estructura de Computadores

Grado en Ingeniería Informática

Universidad Carlos III de Madrid