



Tema 5 (III)

Jerarquía de Memoria



Grupo ARCOS

Estructura de Computadores
Grado en Ingeniería Informática
Universidad Carlos III de Madrid

Contenidos

I. 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

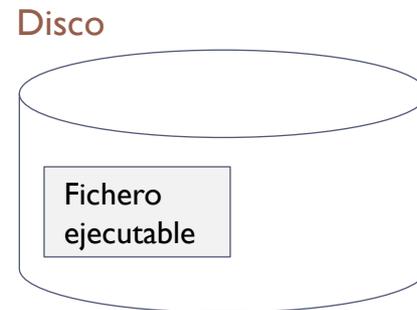
Contenidos

I. Memoria virtual

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

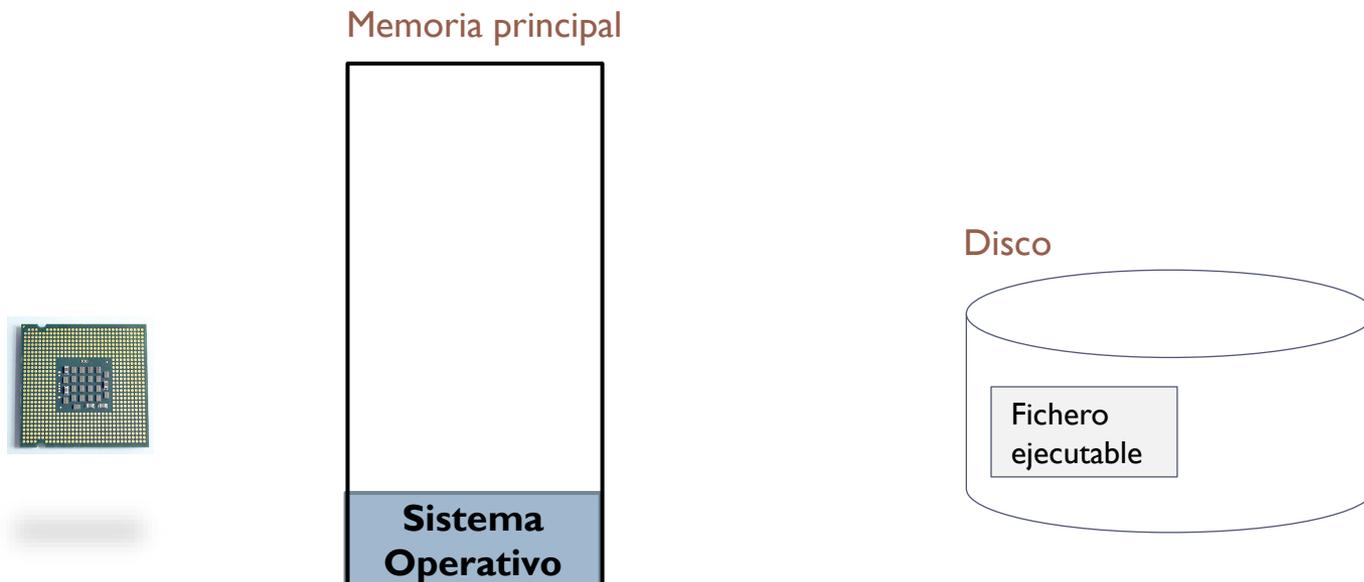
Programa y proceso

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



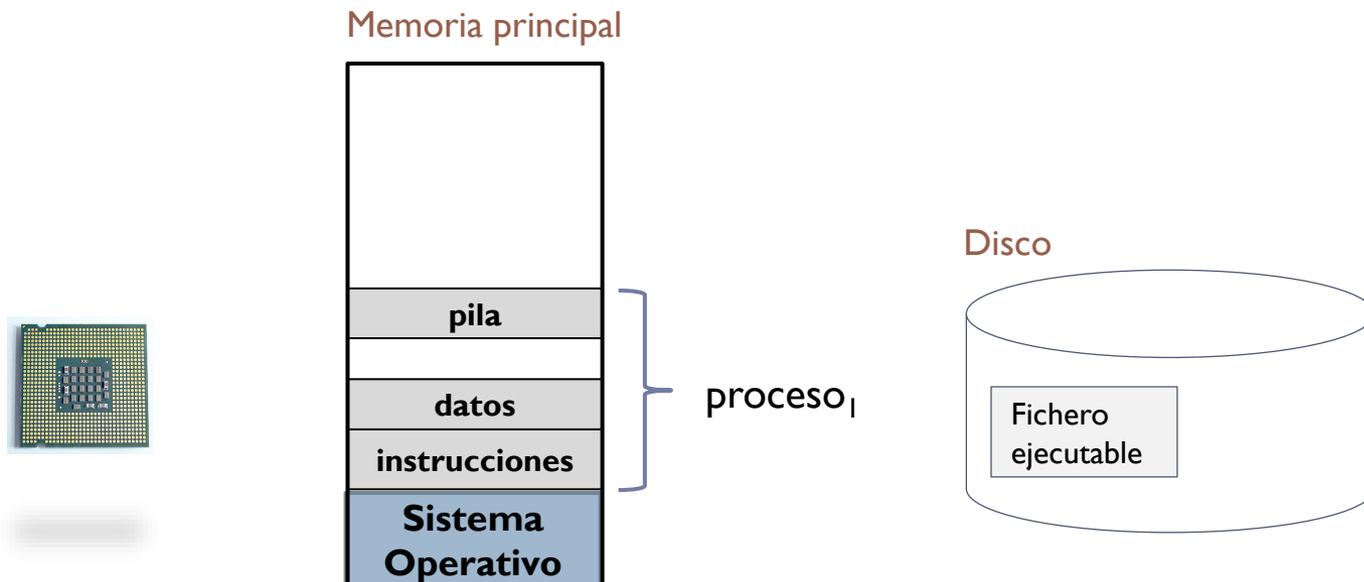
Programa y proceso

- ▶ **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)

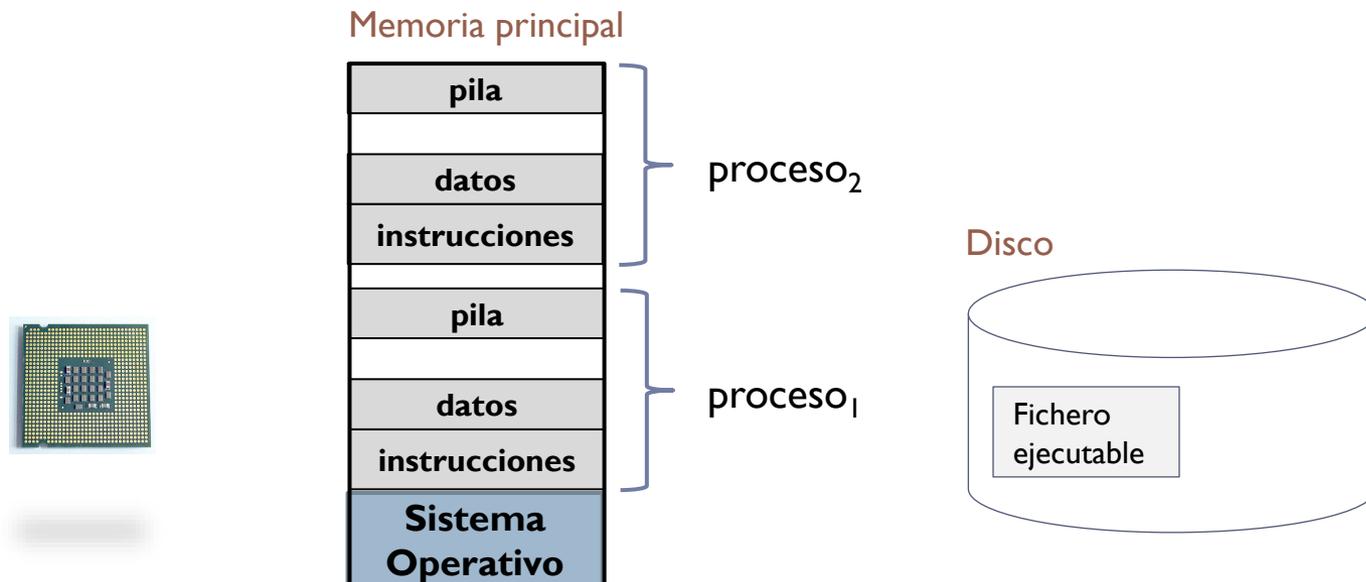
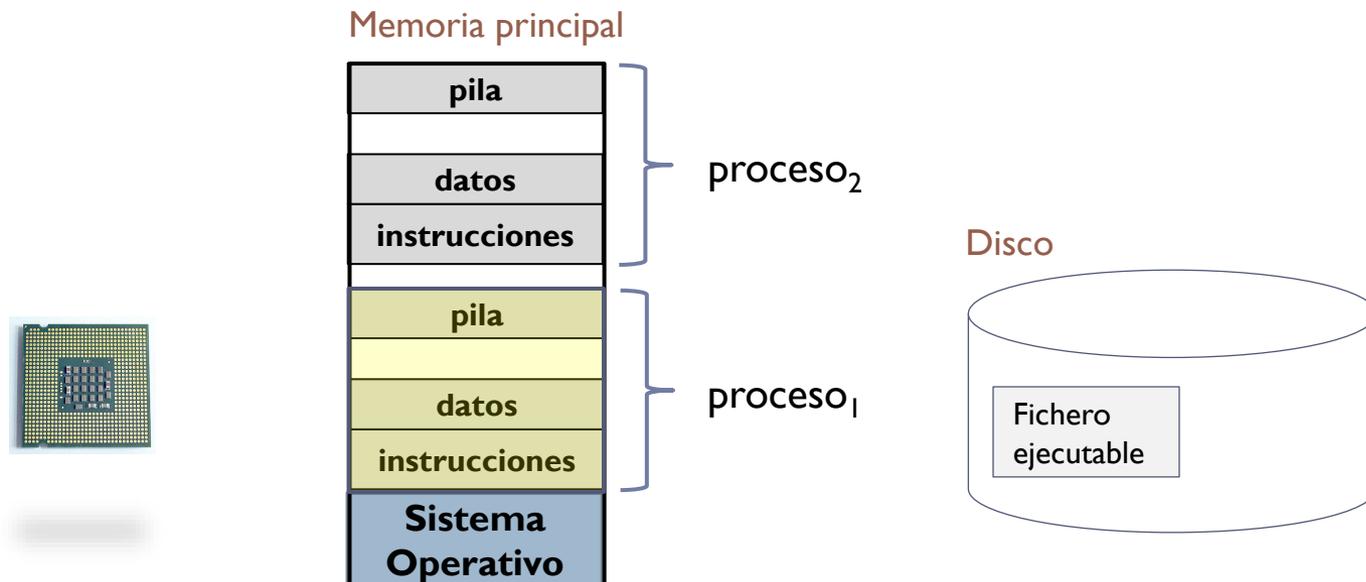


Imagen de un proceso

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

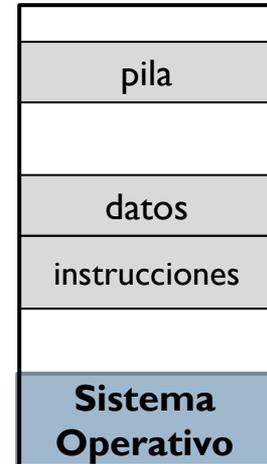


Contenidos

I. Memoria virtual

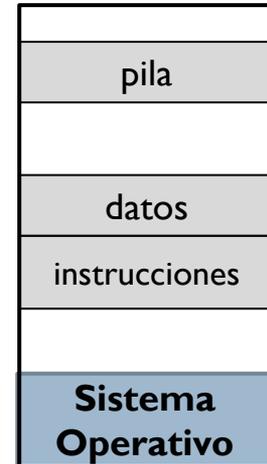
- ▶ Definiciones iniciales
- ▶ **Motivación**
- ▶ Funcionamiento general
- ▶ Memoria virtual paginada
- ▶ Detalles de gestión

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



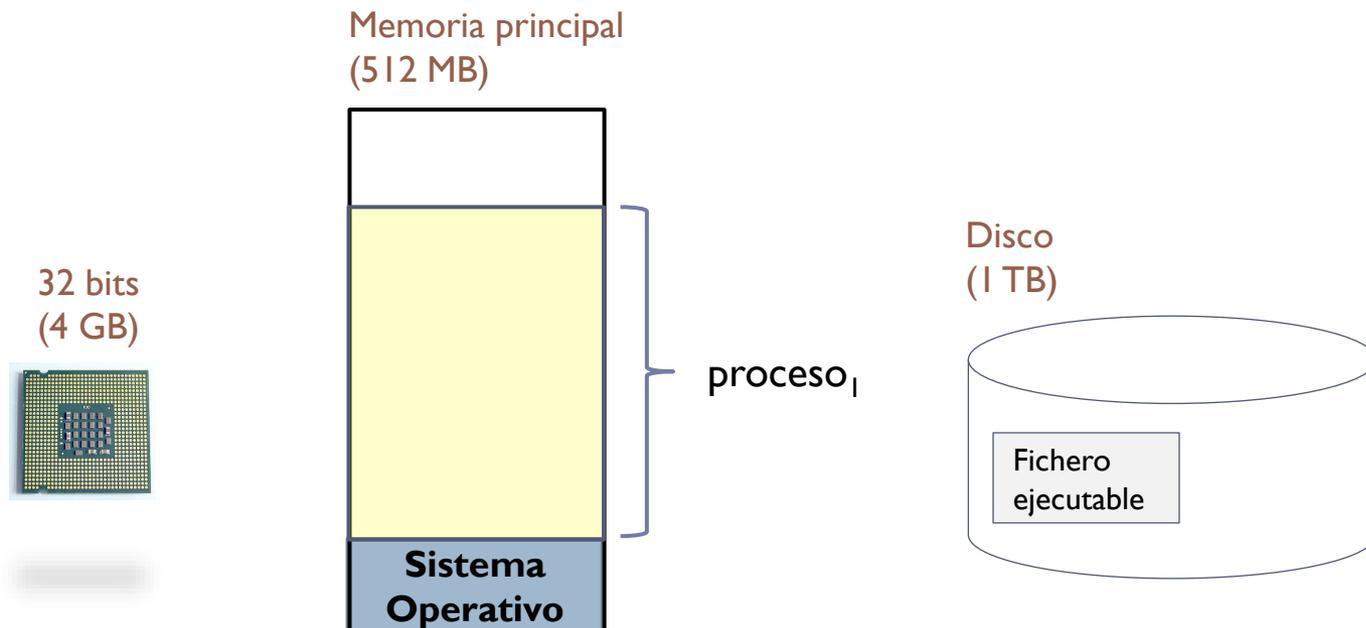
- ▶ En los sistemas sin memoria virtual, el programa se carga completamente en memoria para su ejecución.

- ▶ **Principales problemas:**

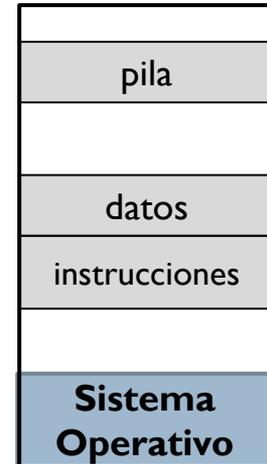
- ▶ El tamaño de la imagen puede limitar su ejecución, o la de otros procesos.
- ▶ La protección necesaria para que un proceso no tenga acceso a la imagen de otro proceso.
- ▶ La necesidad de poder reubicar el programa en cualquier zona de memoria.

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.



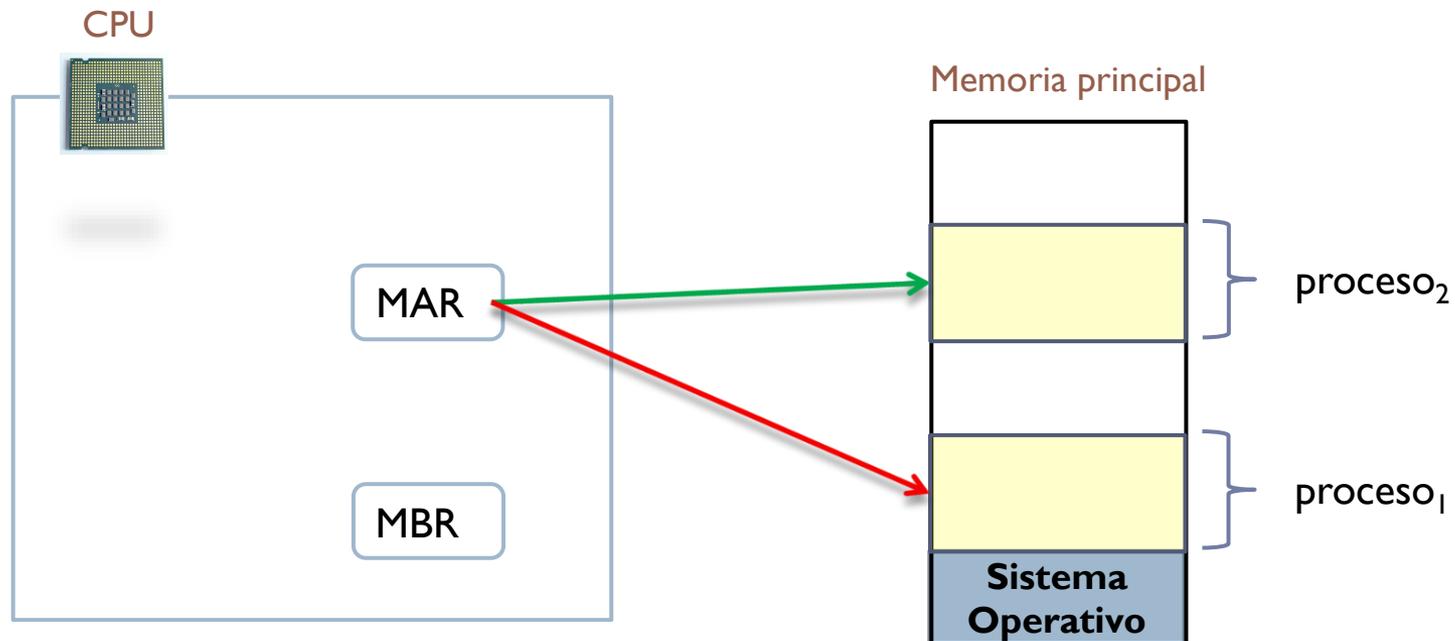
Sistemas **sin** memoria virtual



- ▶ 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.

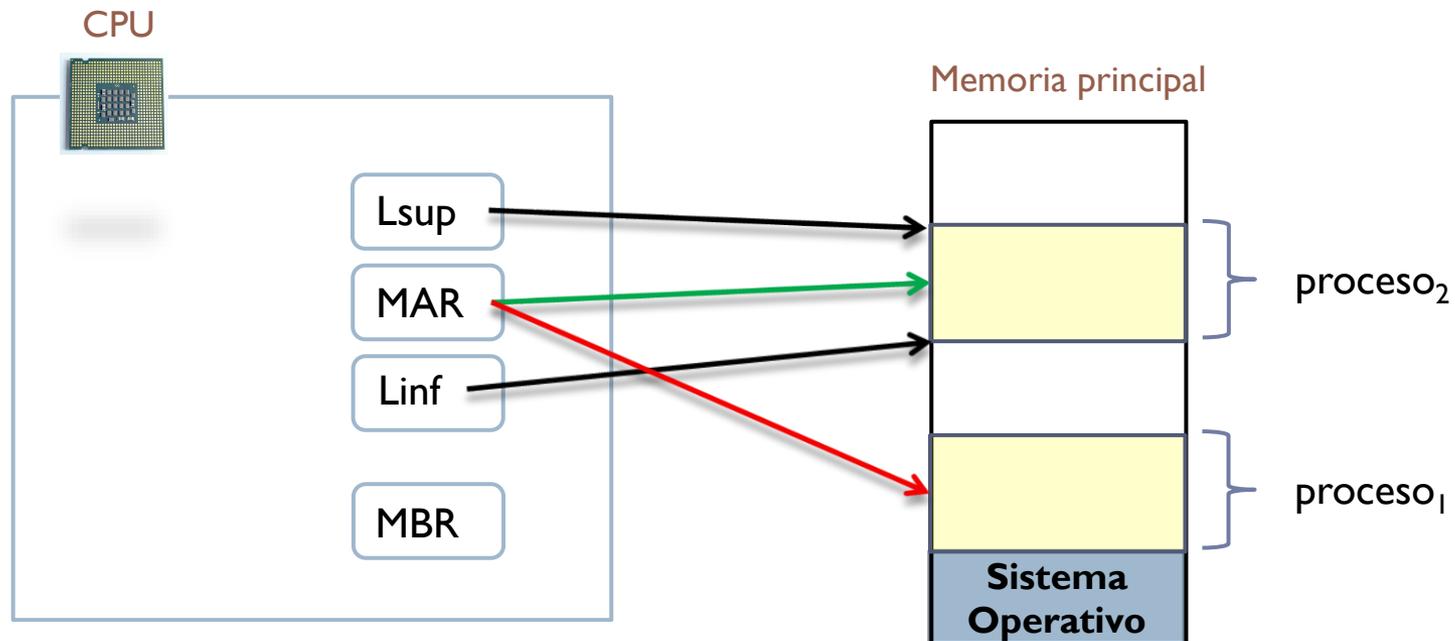
Problema de la protección de memoria

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



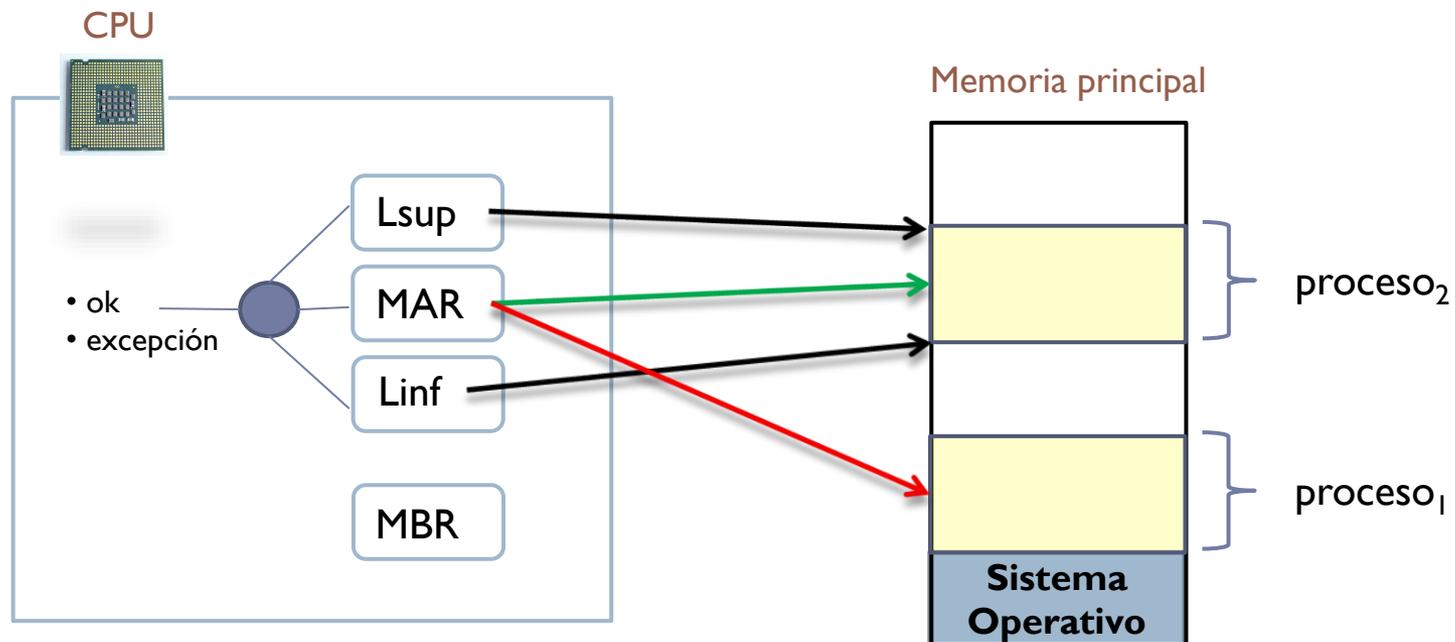
Problema de la protección de memoria

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

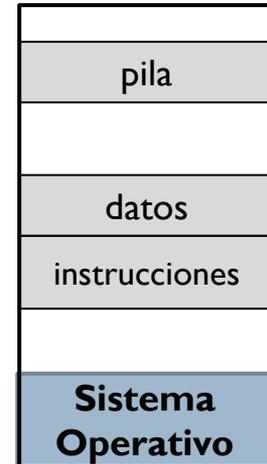


Problema de la protección de memoria

- ▶ 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



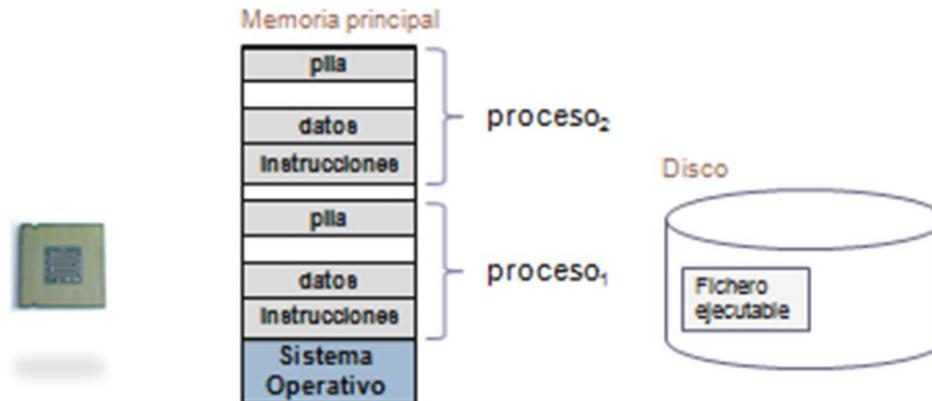
- ▶ En los sistemas sin memoria virtual, el programa se carga completamente en memoria para su ejecución.
- ▶ Principales problemas:
 - ▶ El tamaño de la imagen puede limitar su ejecución, o la de otros procesos.
 - ▶ La protección necesaria para que un proceso no tenga acceso a la imagen de otro proceso.
 - ▶ 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)



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
```

```
int vector[1024];

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



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
```

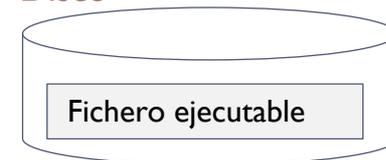


Cabecera

- 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

.	
.	
10000	li \$t0 0
10004	la \$t1 20000+0
10008	bge \$t0 1024 10000+32
10012	mult \$t2 \$t0 4
10016	add \$t2 \$t1 \$t2
10020	sw \$t0 (\$t2)
10024	add \$t0 \$t0 1
10028	b 10000+8
10032	jr \$ra
.	
.	

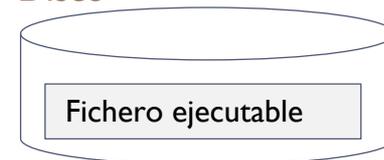


Cabecera

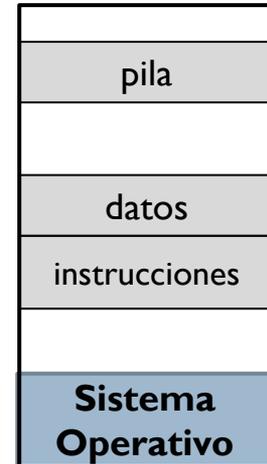
- Tipo de ejecutable
- Tamaño inicial de pila
- Tamaño inicial de datos
- Tamaño inicial de código
- Etc.

```
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



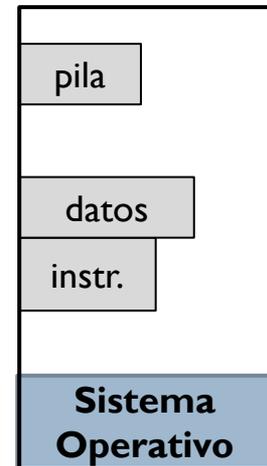
Sistemas **sin** memoria virtual



- ▶ 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.

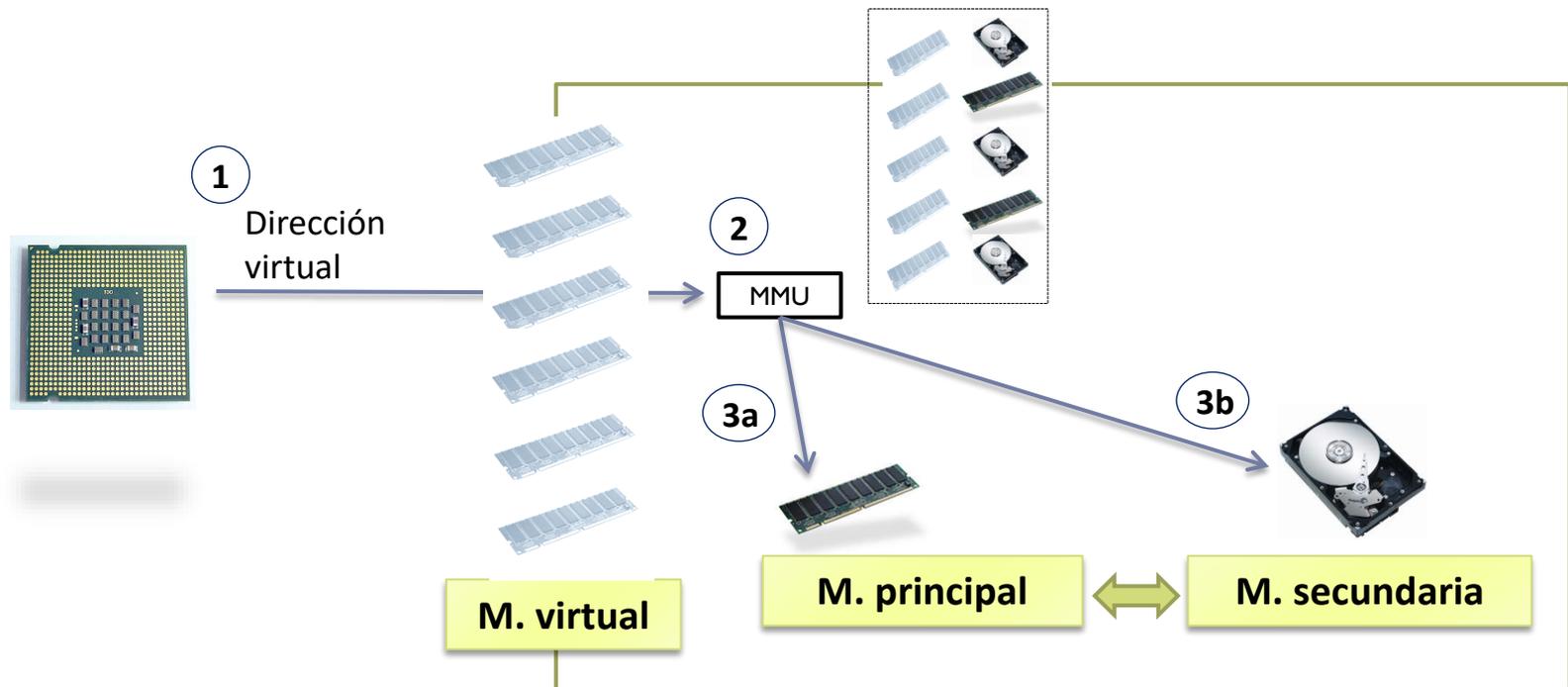
Sistemas **con** memoria virtual

- ▶ 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

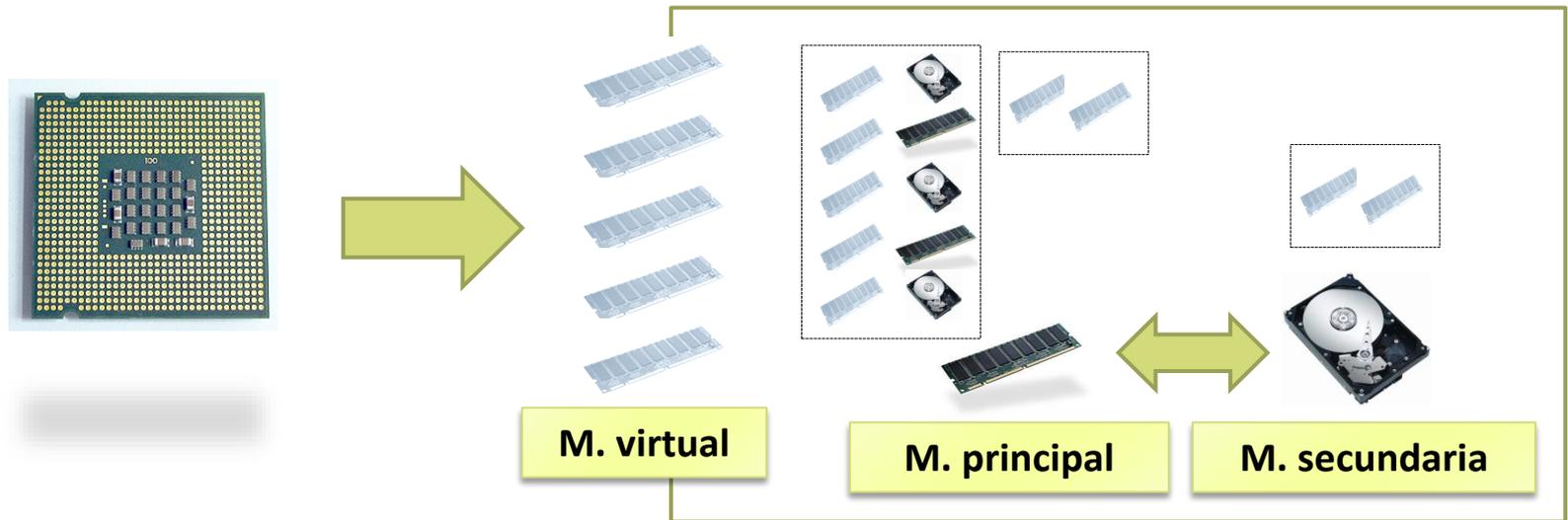


Contenidos

I. Memoria virtual

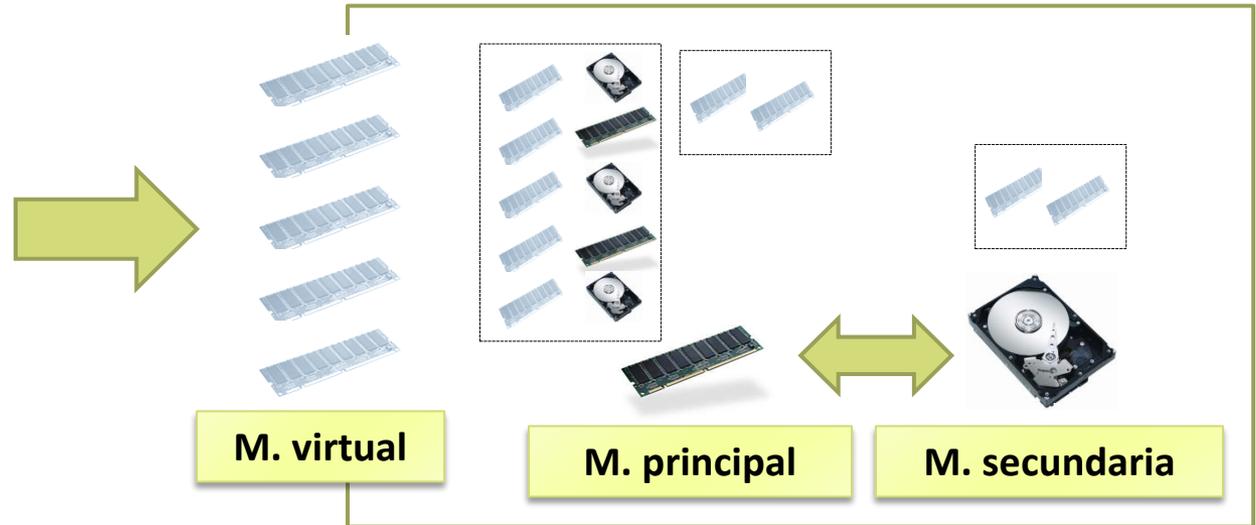
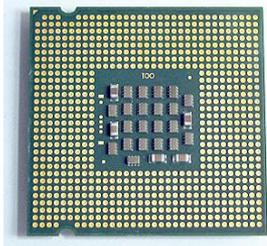
- ▶ Definiciones iniciales
- ▶ Motivación
- ▶ **Funcionamiento general**
- ▶ Memoria virtual paginada
- ▶ Detalles de gestión

Introducción a memoria virtual



Introducción a memoria virtual

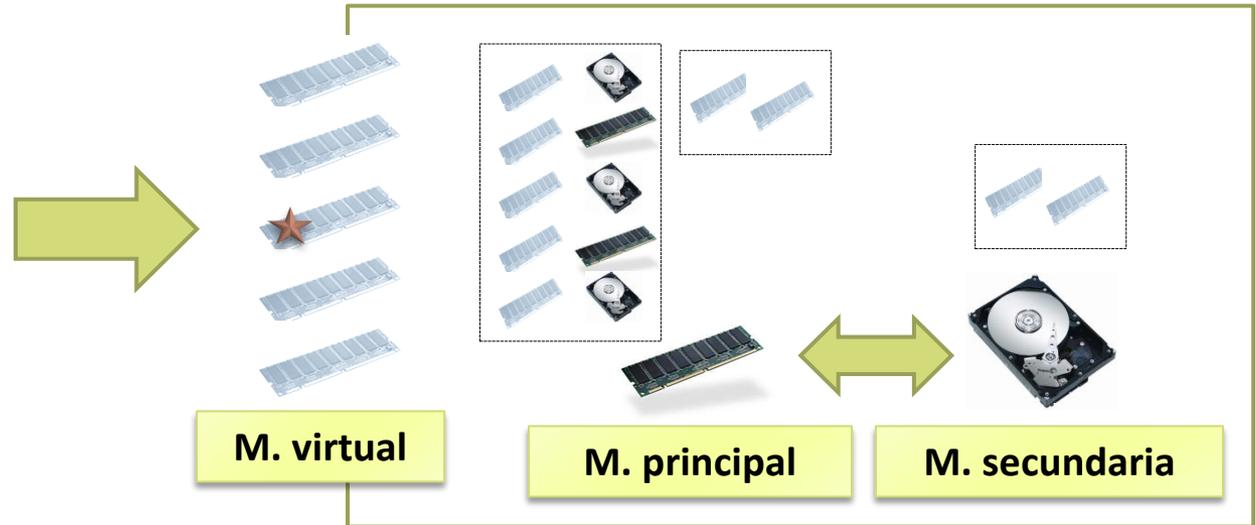
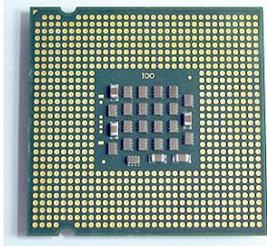
...
lw \$t0 vector
...



Introducción a memoria virtual

...
lw \$t0 vector

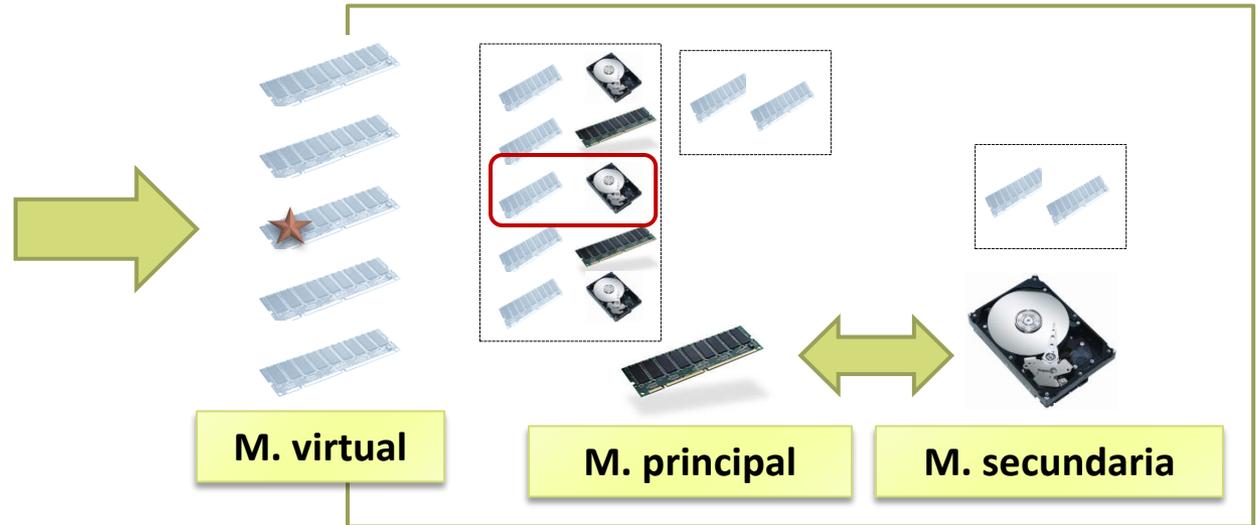
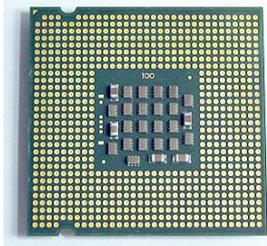
...



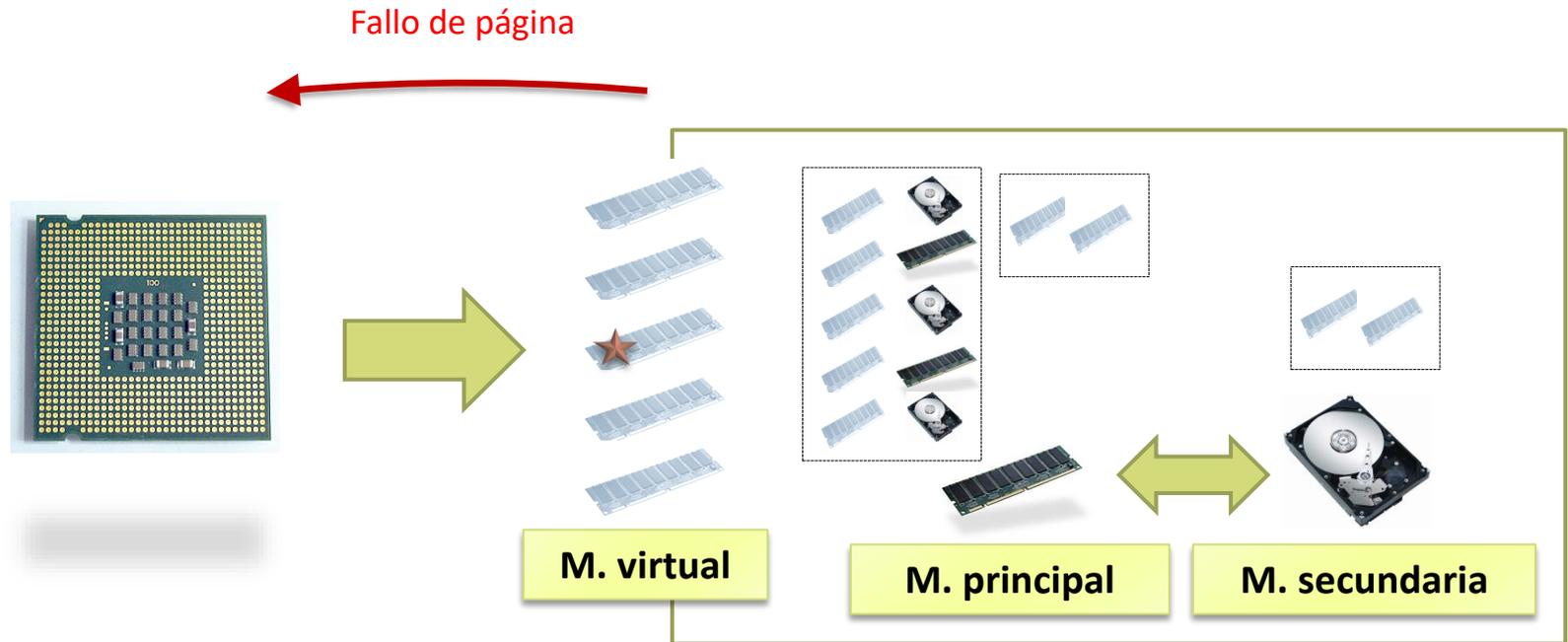
Introducción a memoria virtual

...
lw \$t0 vector

...

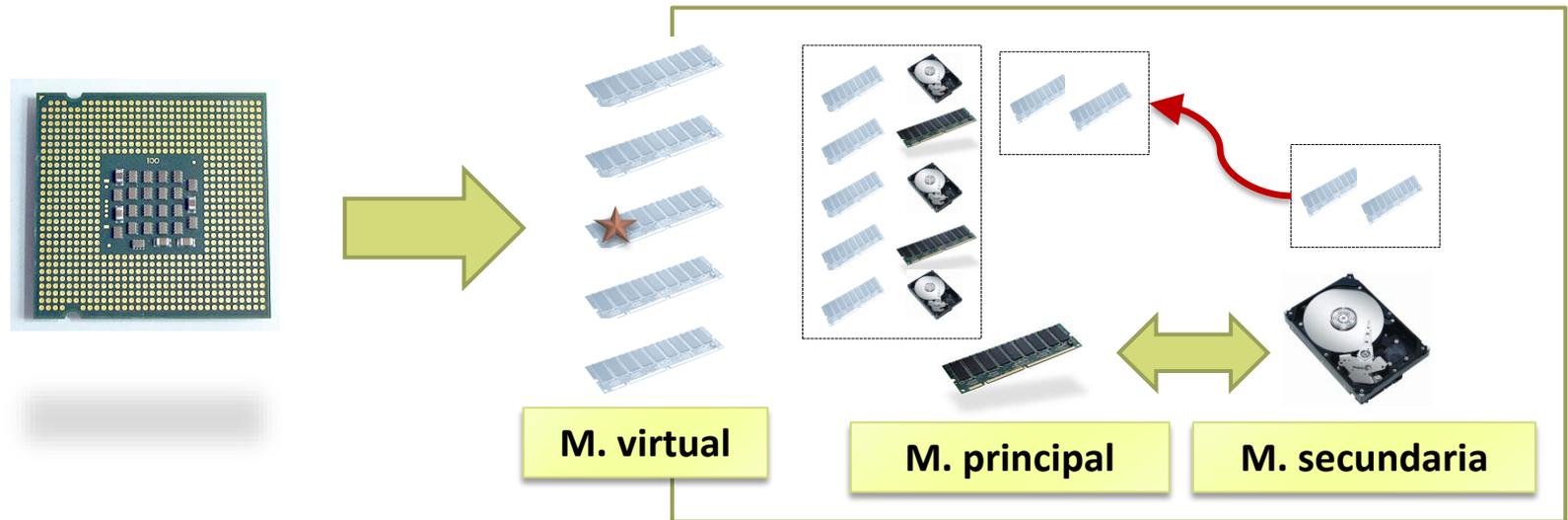


Introducción a memoria virtual



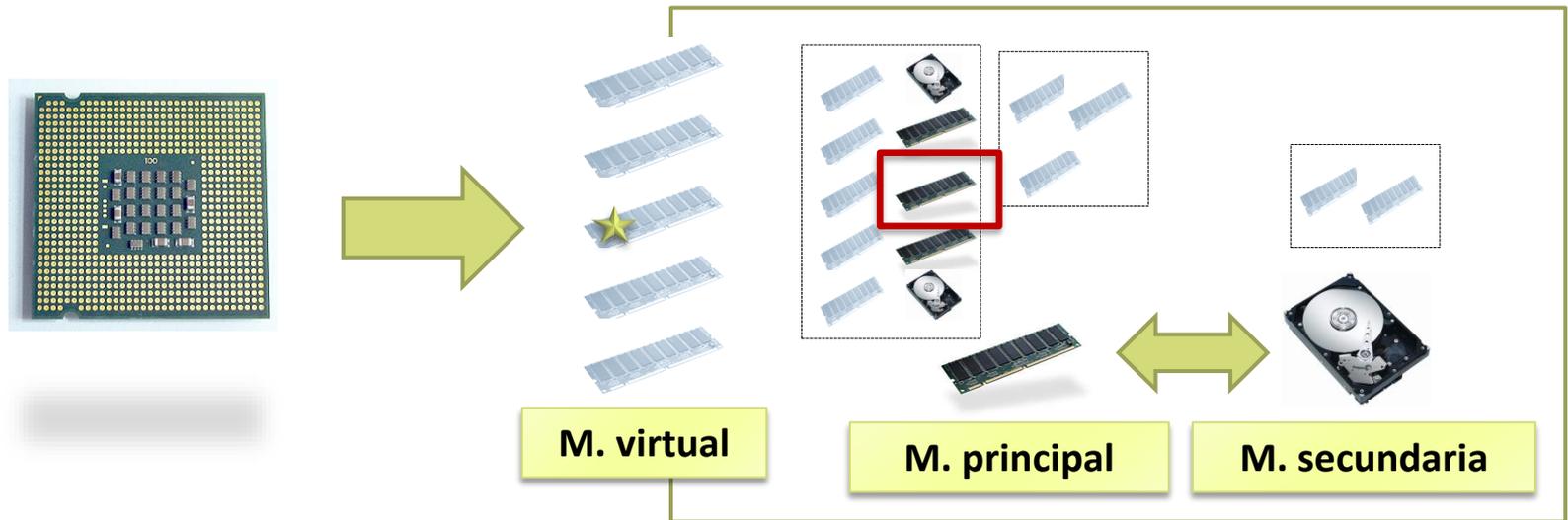
- ▶ El fallo de página es una excepción que provoca que el procesador ejecute la rutina de tratamiento asociada.
- ▶ Está implementada en el sistema operativo.

Introducción a memoria virtual



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

Introducción a memoria virtual

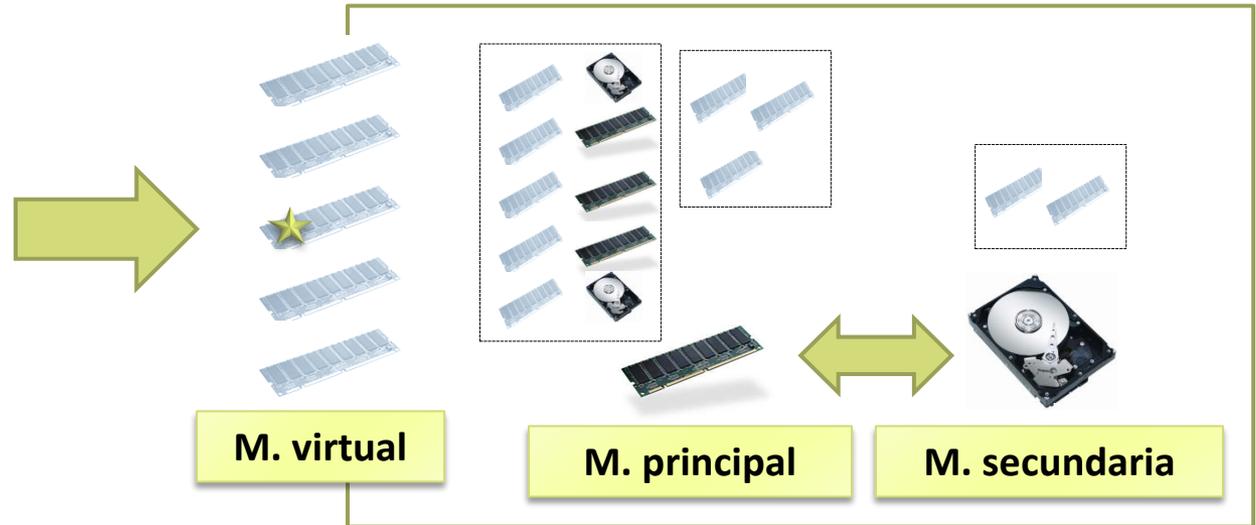
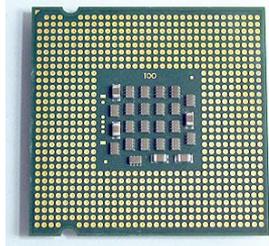


- ▶ El sistema operativo transfiere el 'bloque' solicitado a memoria principal y actualiza la tabla de 'bloques'

Introducción a memoria virtual

...
lw \$t0 vector

...

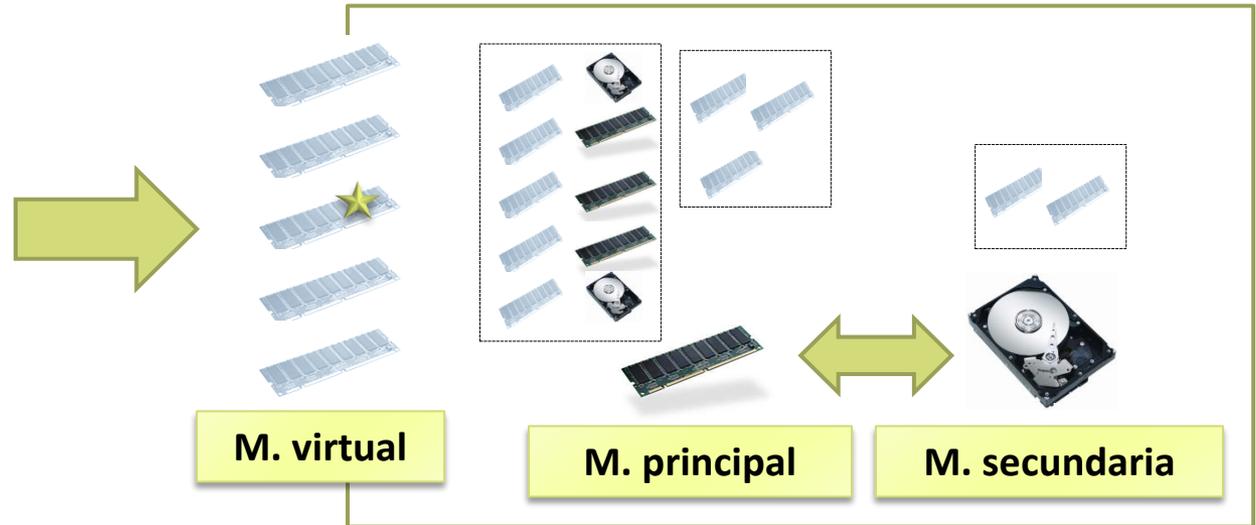


- ▶ Se reanuda la ejecución de la instrucción que provocó el fallo.

Introducción a memoria virtual

Acierto

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



Memoria virtual: windows

Windows Task Manager

File Options View Shut Down Help

Applications Processes Performance Networking Users

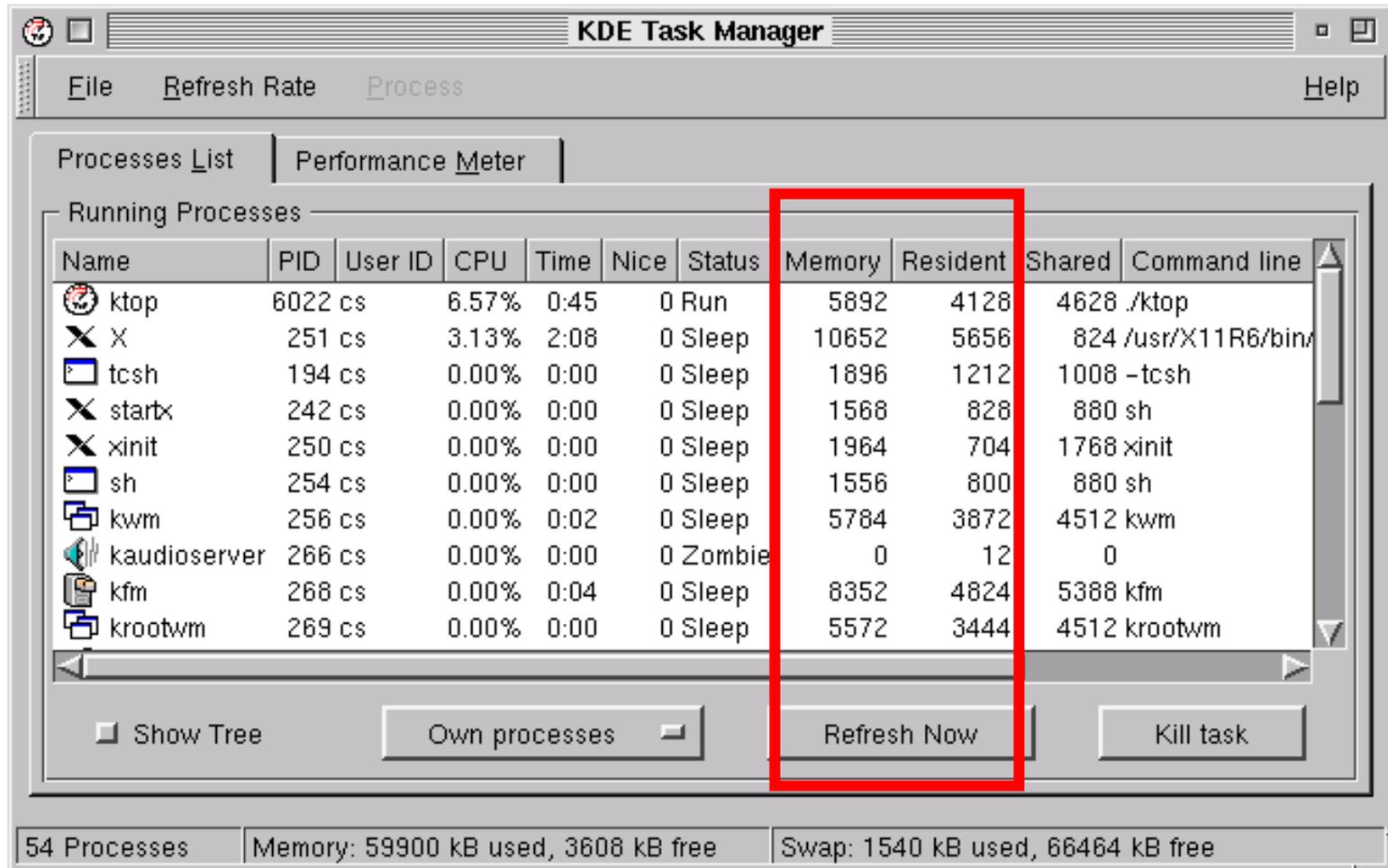
Image Name	User Name	C...	Mem Usage	VM Size	Page Fa...	Base Pri	Threads	I/O Reads	I/O Writes	I/O Other
System	SYSTEM	00	28 K	28 K	1,424,798	Normal	114	114,978	84,264	4,089,482
HDD Thermometer.exe	merlin	00	216 K	5,848 K	880,646	Normal	4	322	1,807	47,728
svchost.exe	SYSTEM	00	1,772 K	2,472 K	4,899	Normal	5	115	118	322
ctfmon.exe	acaldero	00	2,136 K	1,100 K	22,378	Normal	1	1	0	1,044
msmsgs.exe	acaldero	00	1,184 K	3,064 K	3,520	Normal	2	123	128	649
svchost.exe	SYSTEM	00	19,000 K	24,728 K	1,927,569	Normal	81	14,002	165,195	798,840
InCDsrv.exe	SYSTEM	00	1,084 K	1,312 K	1,946	Normal	10	4	145	112,059
Tmntsrv.exe	SYSTEM	00	1,988 K	1,652 K	39,590	Normal	15	838	1,332	291,753
svchost.exe	NETWORK SERVICE	00	1,516 K	1,604 K	13,539	Normal	6	199	87	85,225
vmnetdhcp.exe	SYSTEM	00	160 K	524 K	603	Normal	2	18,033	3	61
GoogleToolbarNotifier...	acaldero	00	196 K	4,596 K	9,727	Normal	6	15	11	1,305
vmware-authd.exe	SYSTEM	00	592 K	1,076 K	1,545	Normal	5	118	6	83,210
vmount2.exe	SYSTEM	00	140 K	1,084 K	1,339	Normal	3	21	13	398
vmnat.exe	SYSTEM	00	184 K	608 K	649	Normal	3	4,570	5	311
MsPMSP5v.exe	SYSTEM	00	48 K	412 K	381	Normal	2	3	3	41
svchost.exe	LOCAL SERVICE	00	156 K	2,184 K	1,099	Normal	3	5	5	221
svchost.exe	LOCAL SERVICE	00	1,304 K	3,128 K	3,020	Normal	18	93	81	1,812
CTSVCCDA.EXE	SYSTEM	00	40 K	408 K	325	Normal	2	5	5	84
wmpnetwk.exe	NETWORK SERVICE	00	1,692 K	6,004 K	3,284	Normal	14	56	6	9,325
smss.exe	SYSTEM	00	44 K	164 K	226	High	3	9	4	152
svchost.exe	SYSTEM	00	188 K	1,576 K	1,094	Normal	8	3	3	217

Show processes from all users

End Process

Processes: 52 CPU Usage: 9% Commit Charge: 564M / 1498M

Memoria virtual: linux



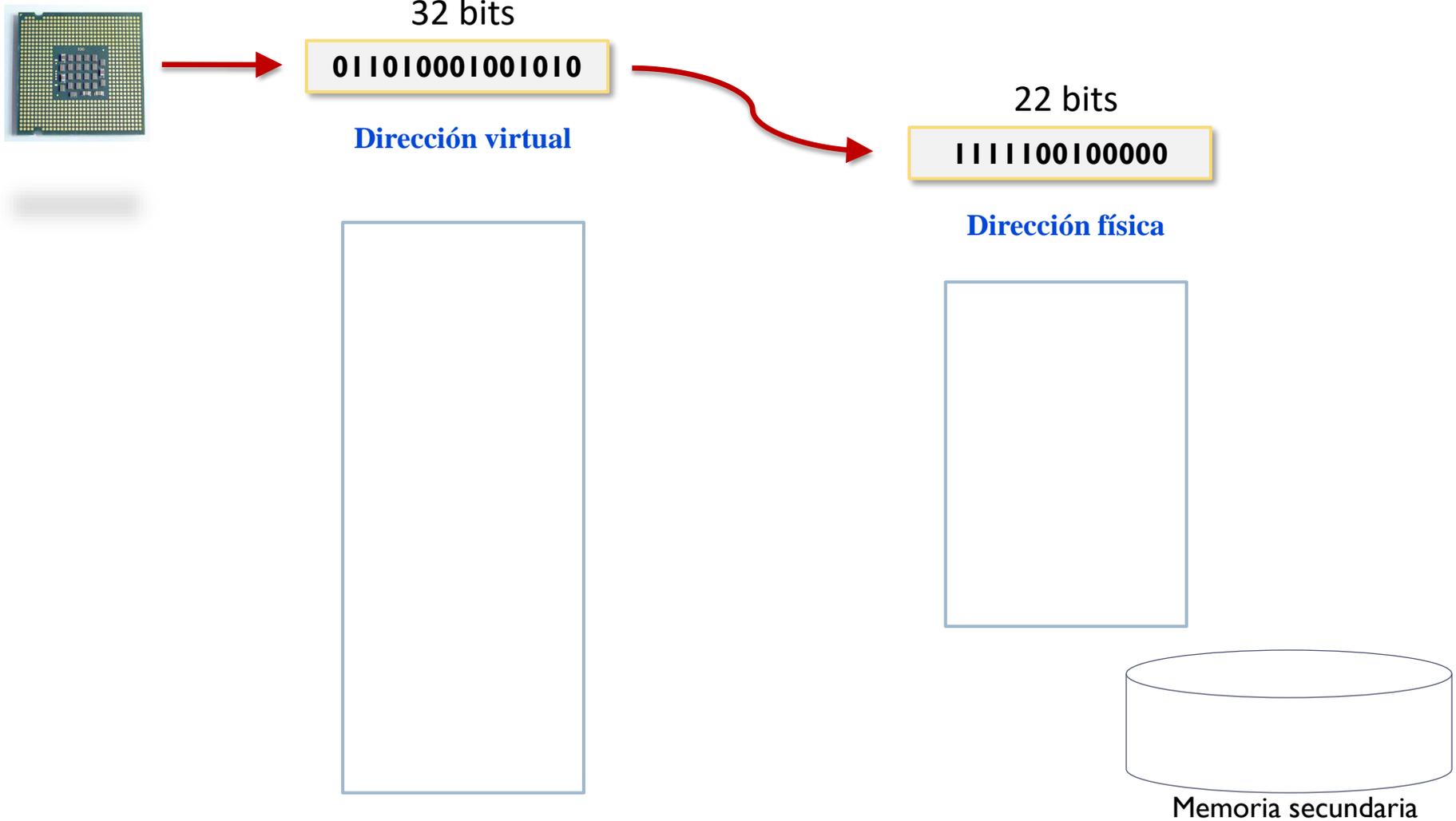
The screenshot shows the KDE Task Manager window. The 'Processes List' tab is active, displaying a table of running processes. A red rectangular box highlights the 'Memory' and 'Resident' columns. The status bar at the bottom indicates 54 processes, 59900 kB of memory used, and 1540 kB of swap used.

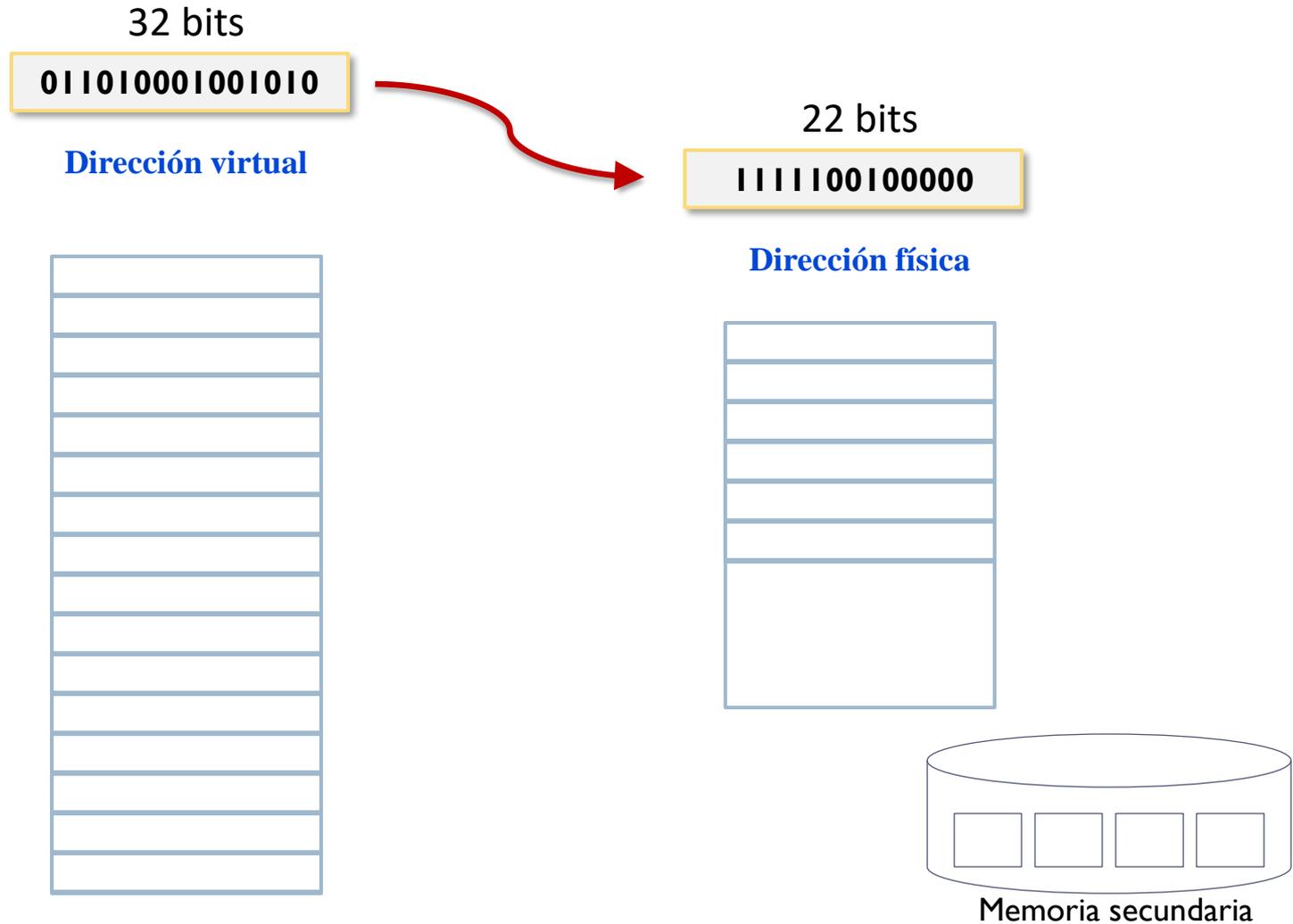
Name	PID	User ID	CPU	Time	Nice	Status	Memory	Resident	Shared	Command line
ktop	6022	cs	6.57%	0:45	0	Run	5892	4128	4628	./ktop
X	251	cs	3.13%	2:08	0	Sleep	10652	5656	824	/usr/X11R6/bin/
tcsh	194	cs	0.00%	0:00	0	Sleep	1896	1212	1008	-tcsh
startx	242	cs	0.00%	0:00	0	Sleep	1568	828	880	sh
xinit	250	cs	0.00%	0:00	0	Sleep	1964	704	1768	xinit
sh	254	cs	0.00%	0:00	0	Sleep	1556	800	880	sh
kwm	256	cs	0.00%	0:02	0	Sleep	5784	3872	4512	kwm
kaudioserver	266	cs	0.00%	0:00	0	Zombie	0	12	0	
kfm	268	cs	0.00%	0:04	0	Sleep	8352	4824	5388	kfm
krootwm	269	cs	0.00%	0:00	0	Sleep	5572	3444	4512	krootwm

Contenidos

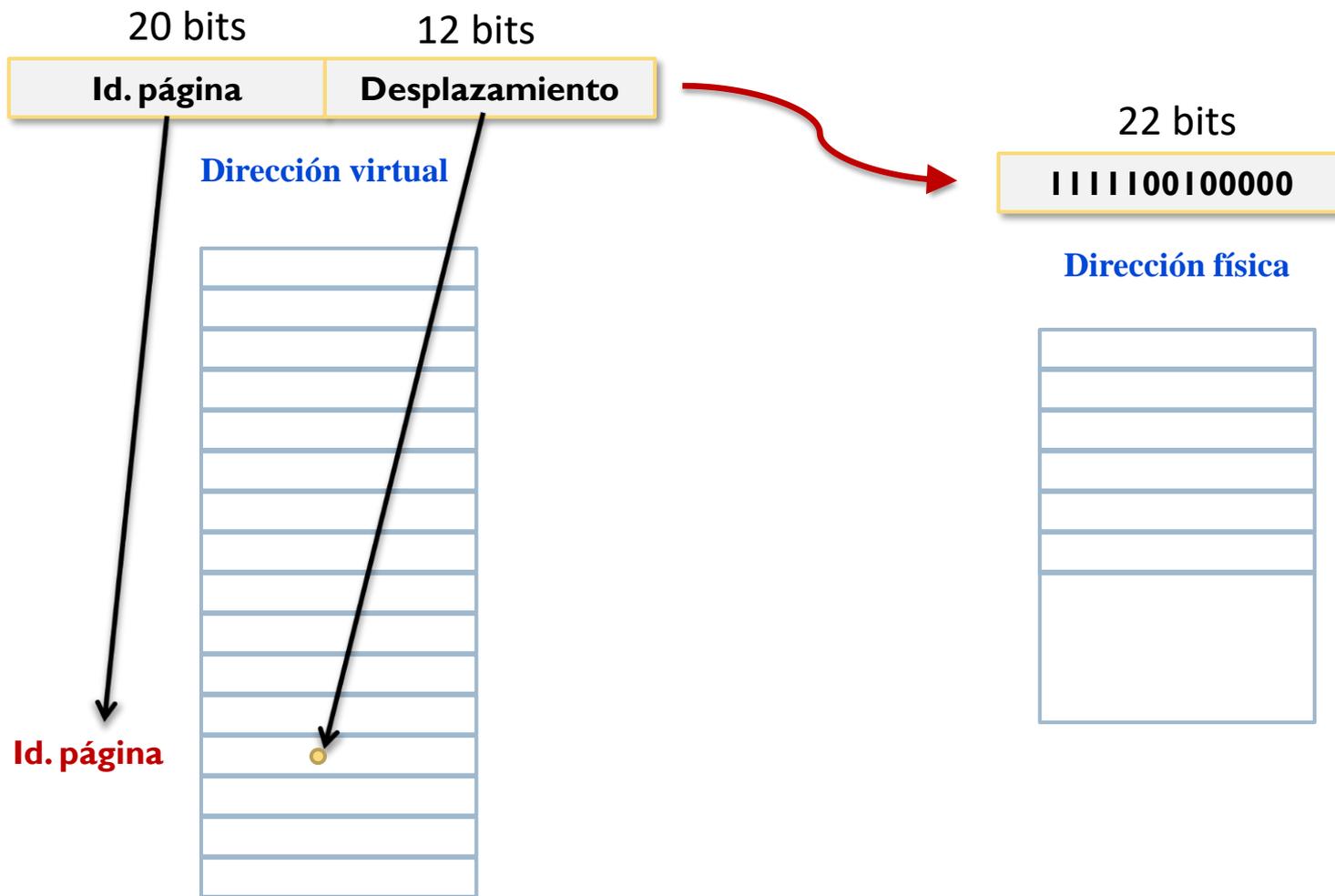
I. Memoria virtual

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

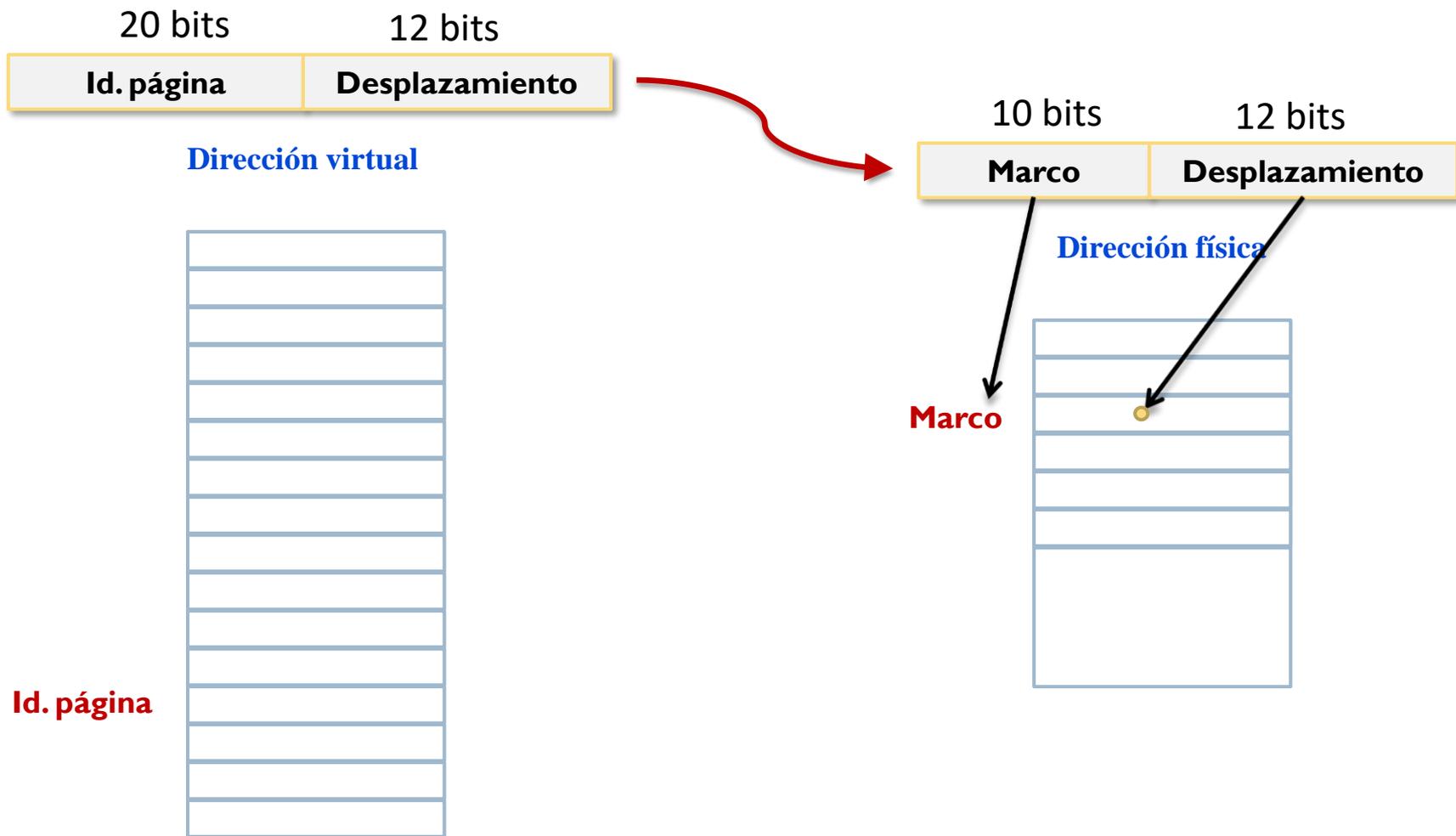




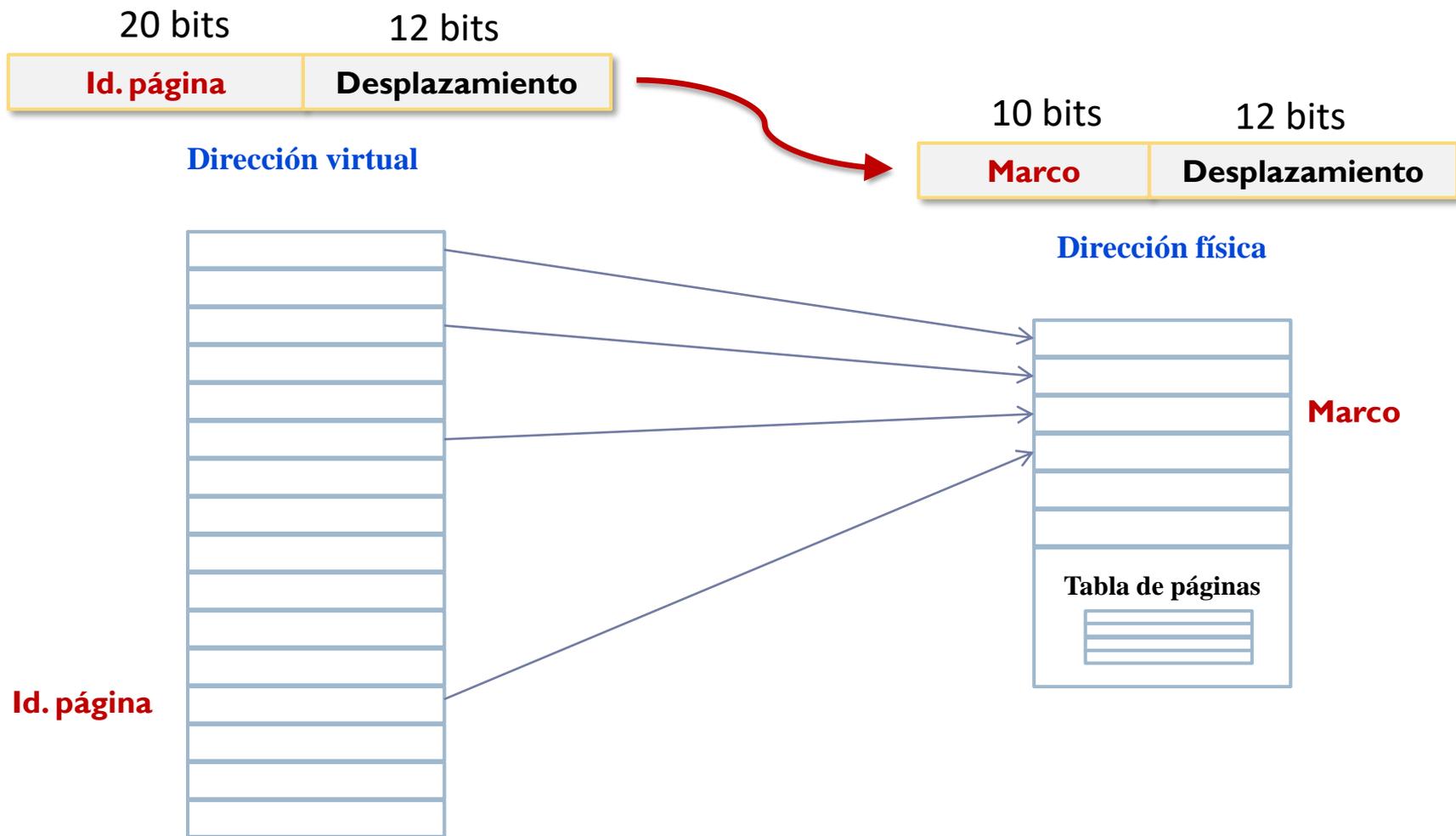
División en bloques del mismo tamaño -> páginas



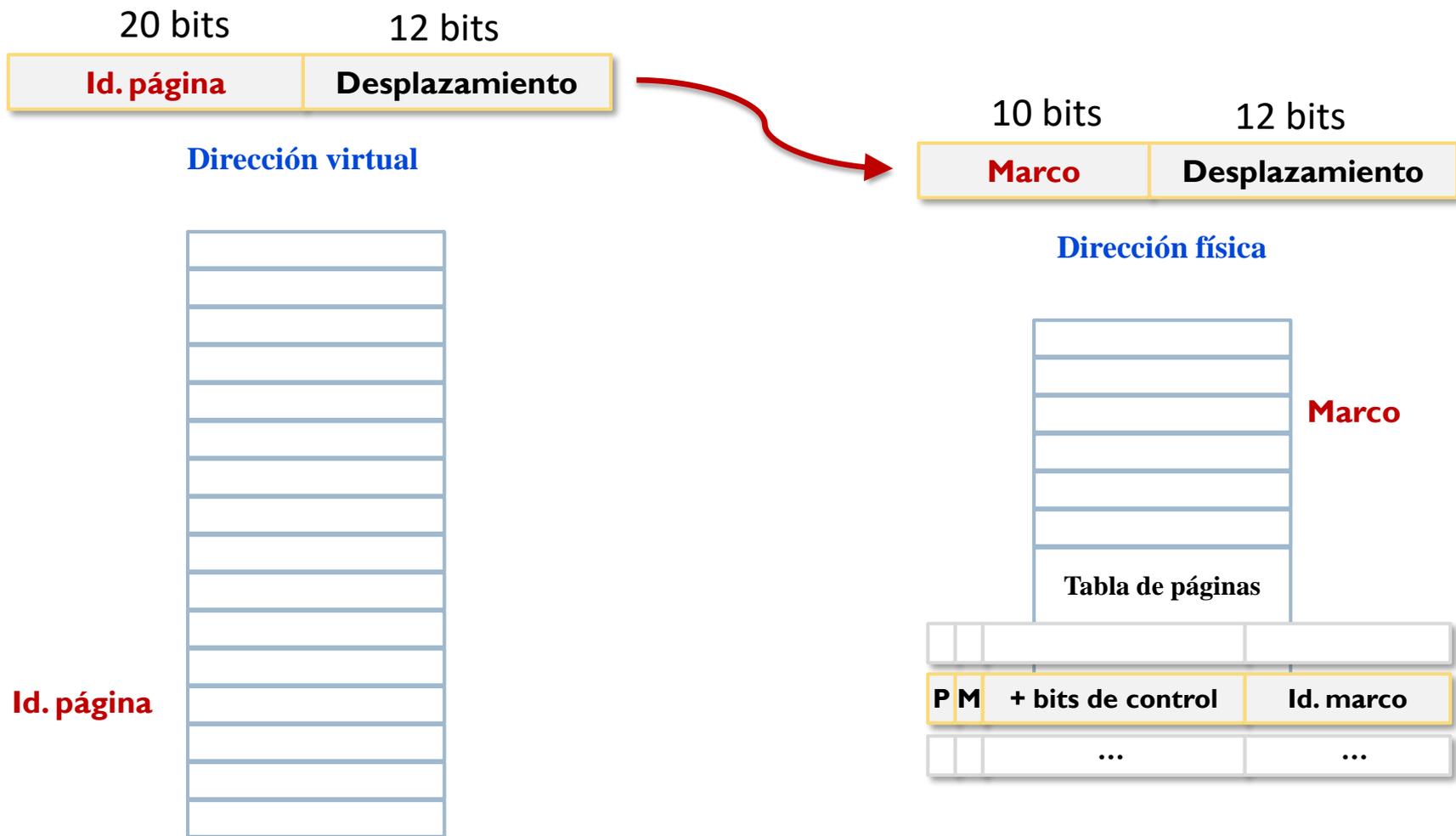
División en bloques del mismo tamaño -> páginas



División en bloques del mismo tamaño -> páginas

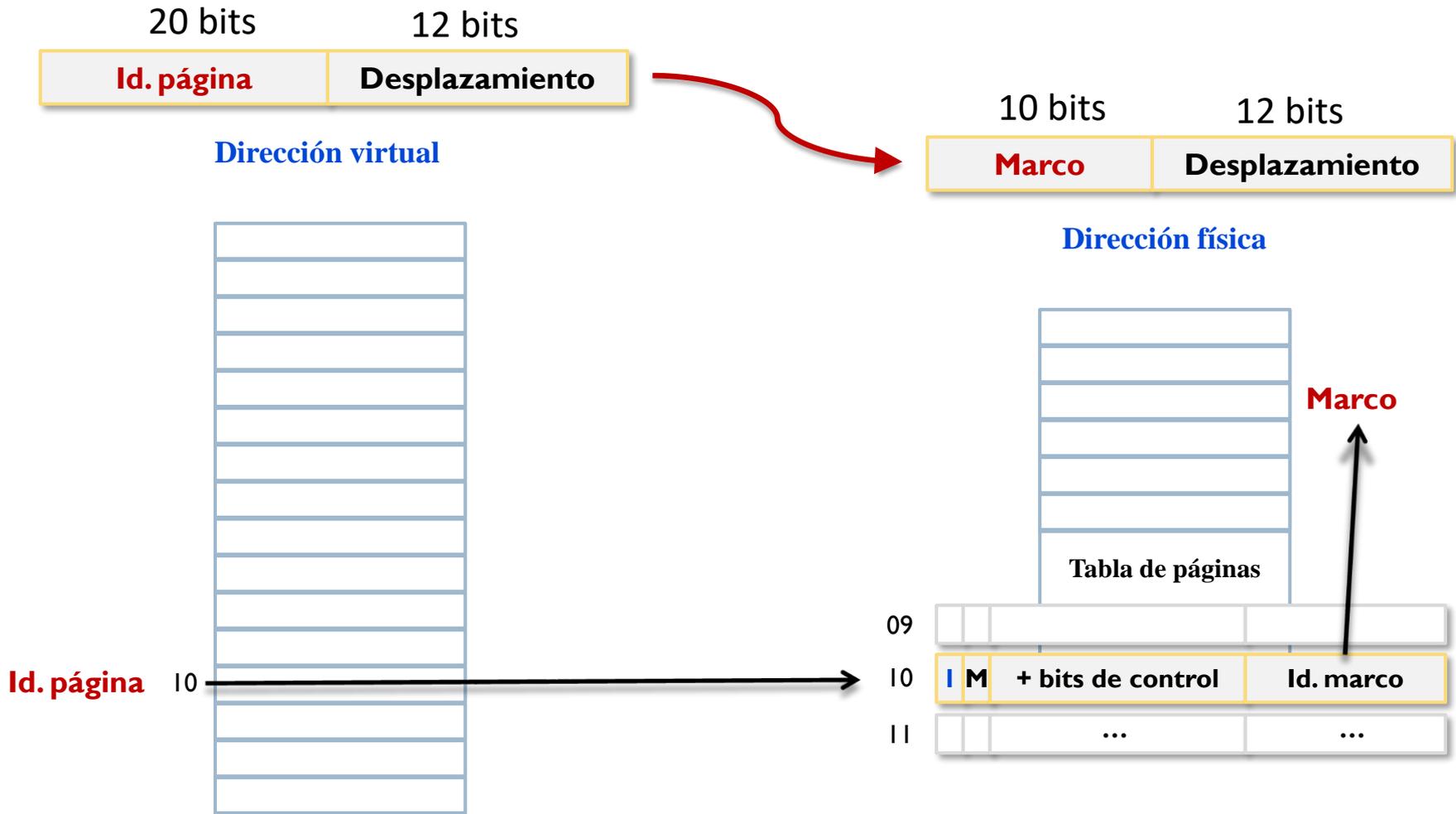


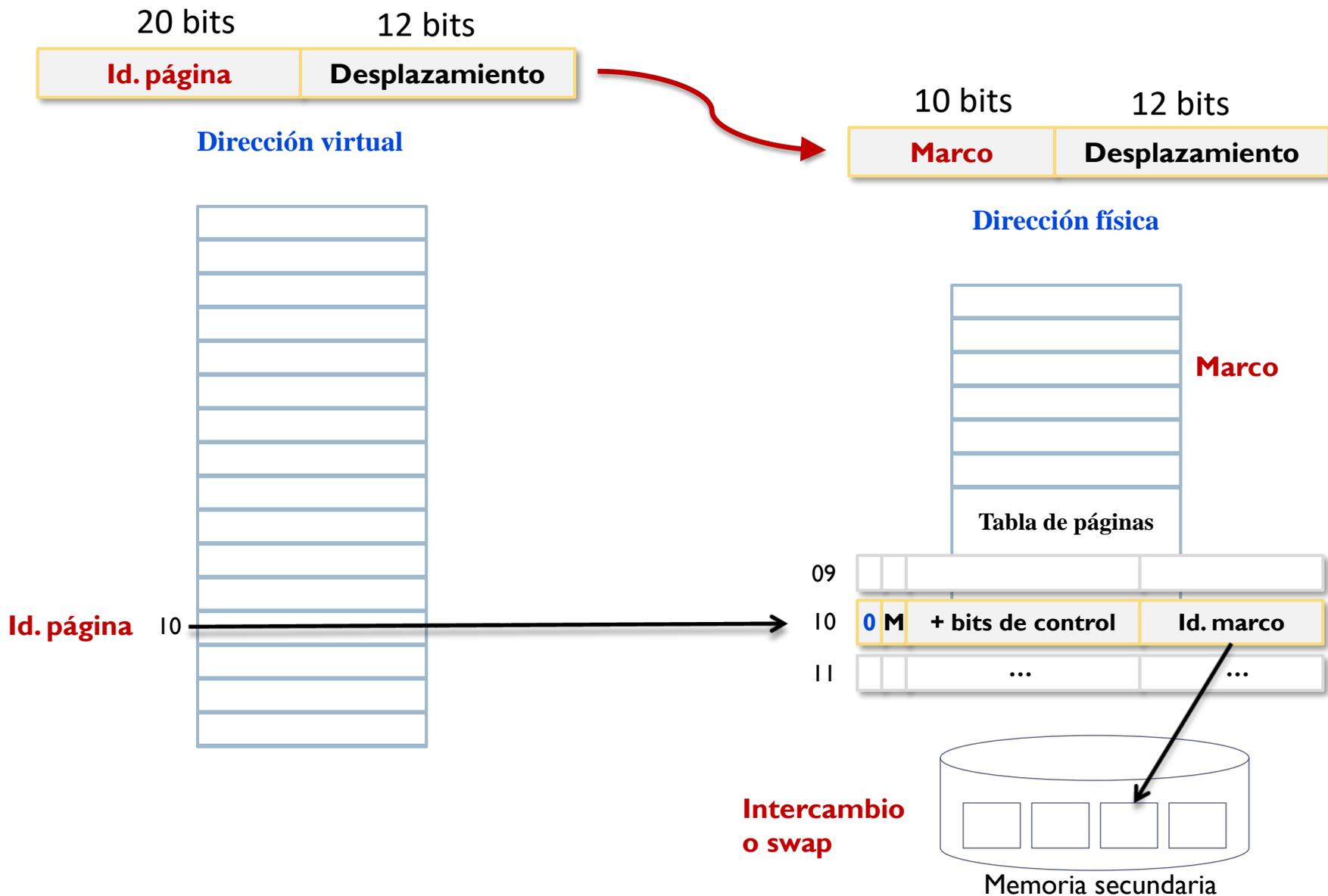
Correspondencia entre Id. página y marco -> T. páginas



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

Ejemplo





Ejercicio



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

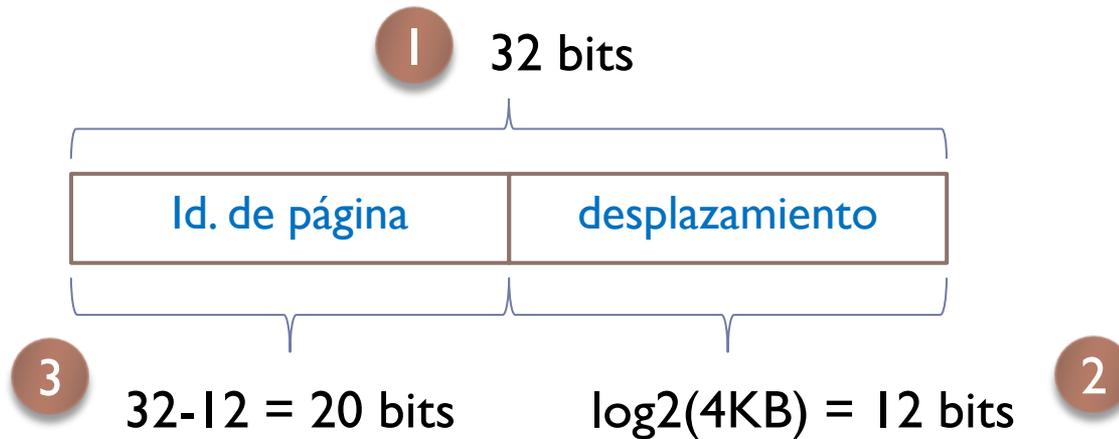
- ▶ Se pide:
 - a) Indique el formato de la dirección virtual y el número de marcos de página.



Ejercicio (sol.)



- ▶ Formato de la dirección virtual:



- ▶ Número de marcos de página:

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

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.

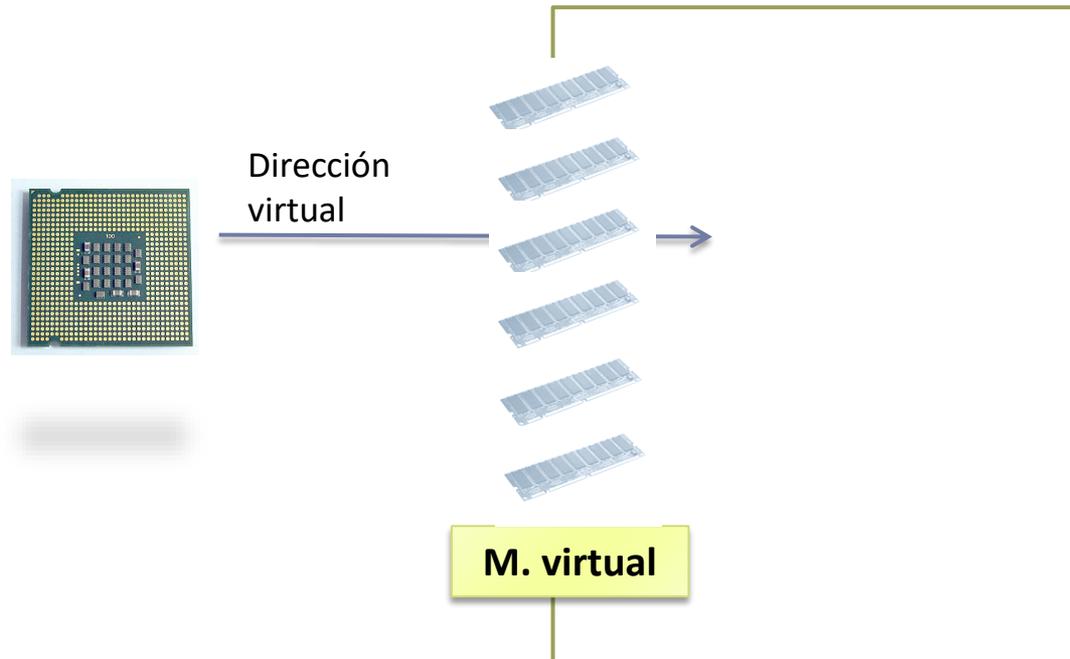
- ▶ Se pide:
 - a) ¿Cuál es el espacio de memoria direccionable por un programa en ejecución?
 - b) ¿Cuál es el máximo tamaño de la tabla de páginas en este computador?



Ejercicio (sol.)



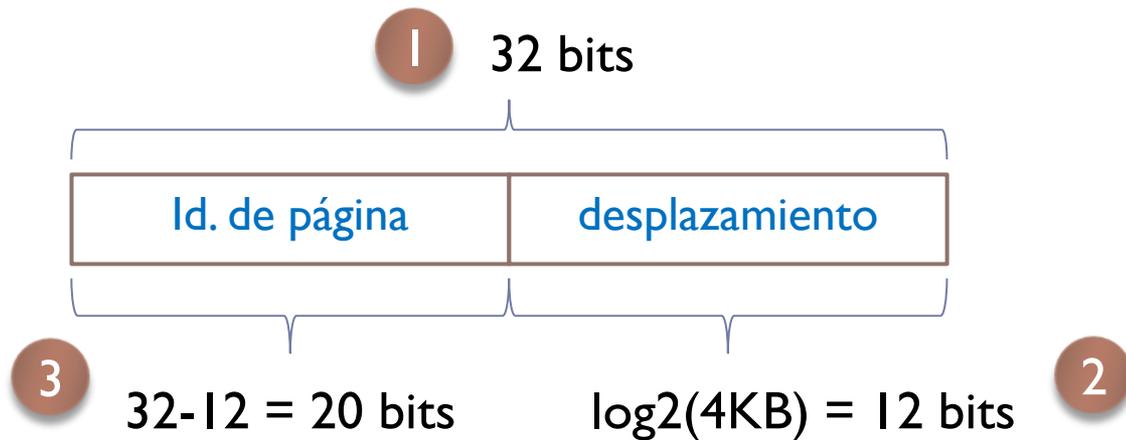
- ▶ El espacio de memoria direccionable por un programa en ejecución está determinado por el número de bits de la dirección virtual:
 - ▶ $2^{32} = 4 \text{ GB}$



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:
 - ▶ $2^{20} * 4 \text{ bytes (32 bits)} = 4 \text{ MB}$



- 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	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:
 - ▶ $7 * 4 \text{ KB} = 28 \text{ KB}$



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	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:
 - ▶ $28 * 1024 - 1$

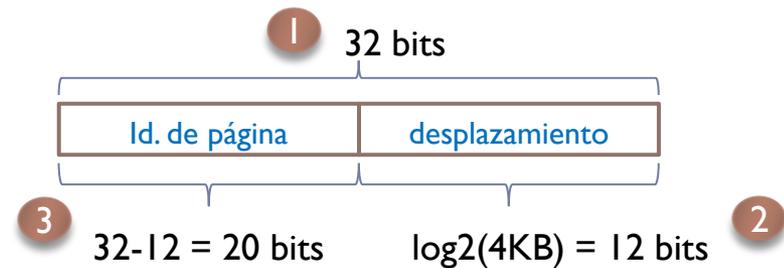


Ejercicio (sol.)



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

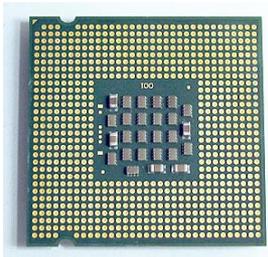


Contenidos

I. 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)



Dirección virtual

20 bits

12 bits

Número de página

Desplazamiento

Entrada de la tabla de páginas

P	M	Otros bits de control	Número de marco

- 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.

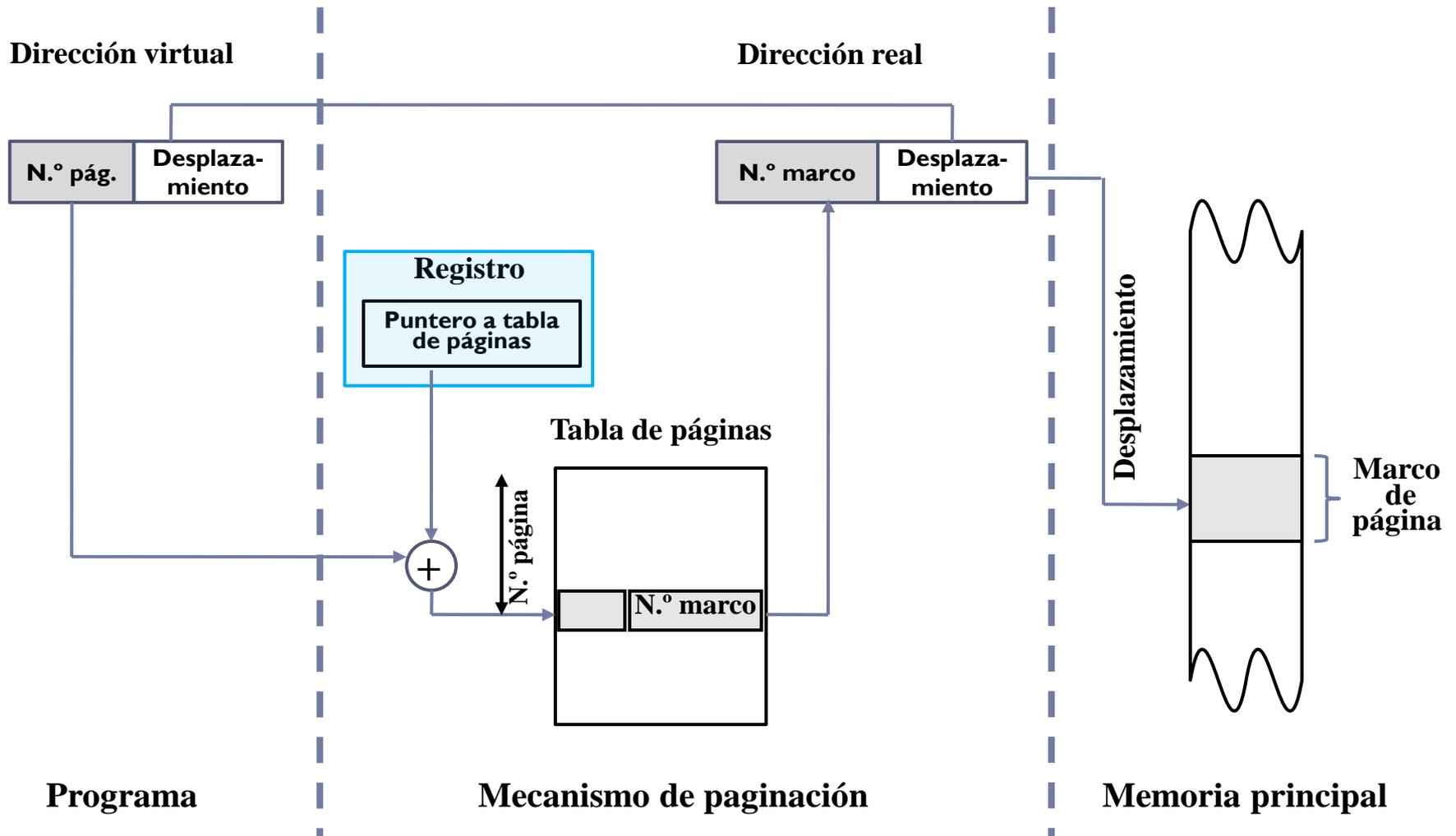
▶ Uso:

- ▶ 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:

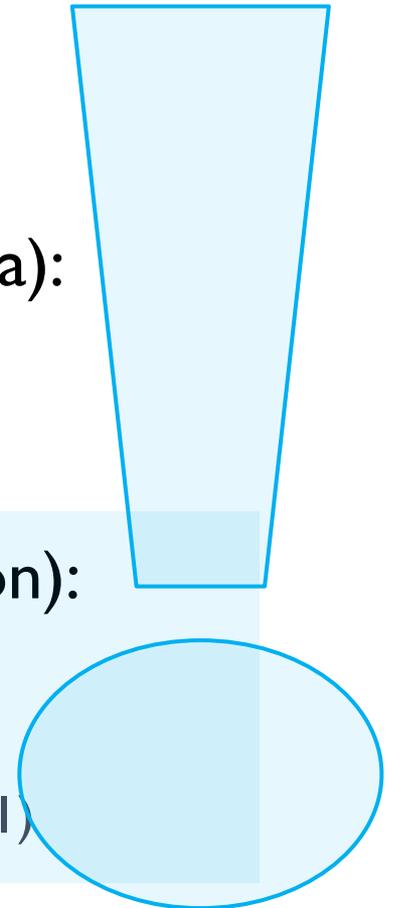
- ▶ Página no residente se marca ausente
- ▶ Se guarda dirección del bloque de *swap* que la contiene

▶ 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

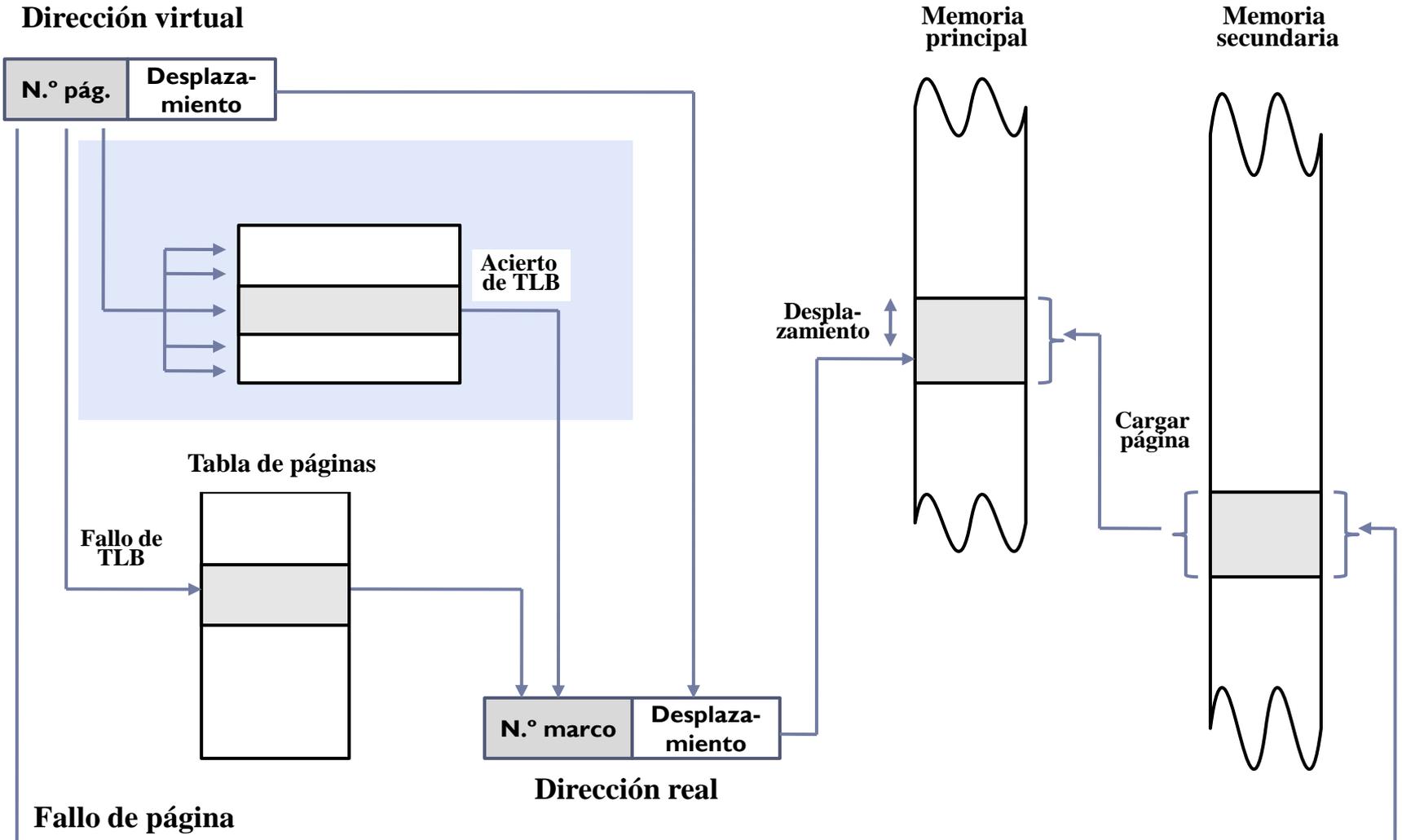
I. Memoria virtual

- ▶ 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

I. 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

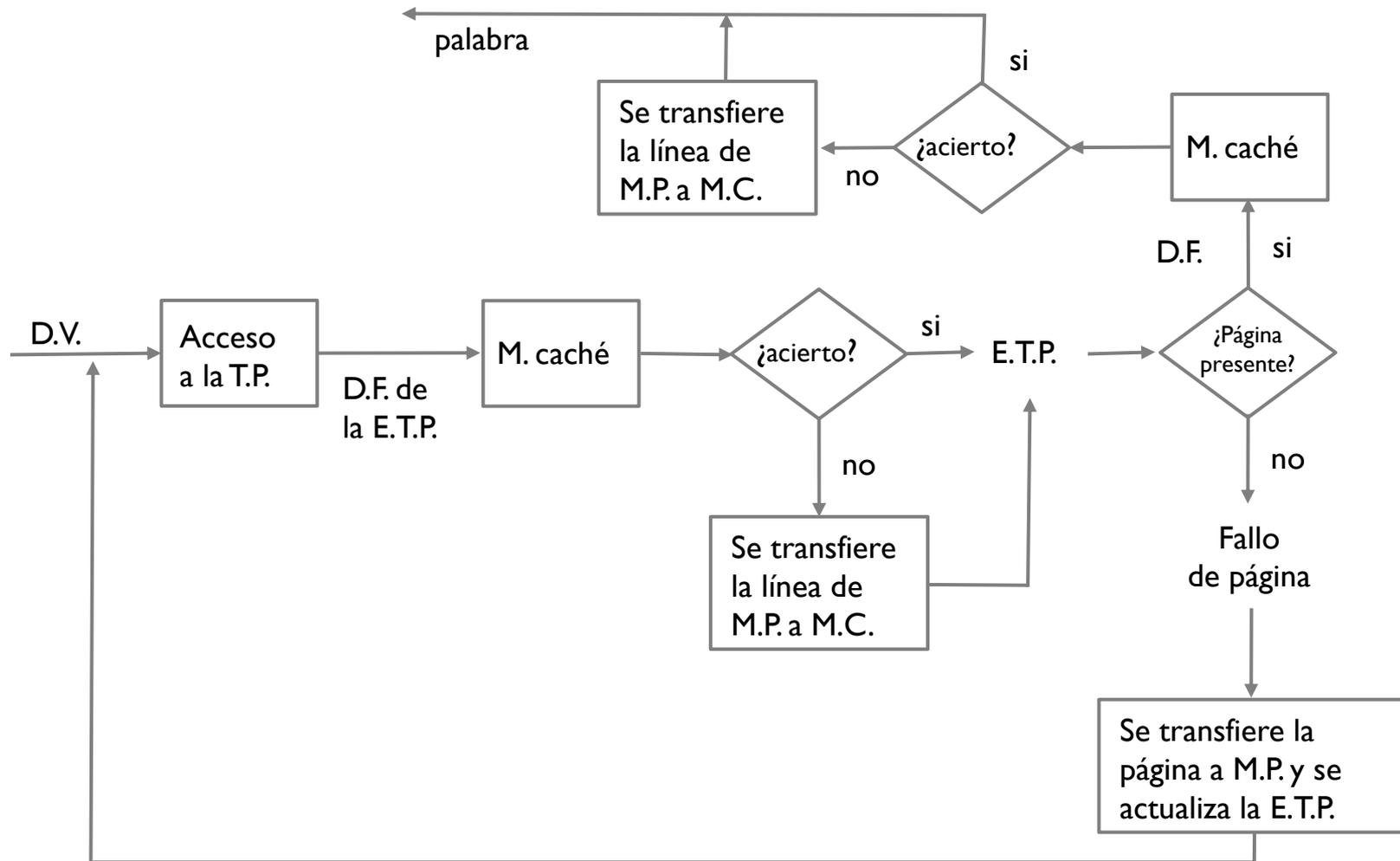
Caché

- ▶ Acelerar el acceso
- ▶ Transferencia por bloques o líneas.
- ▶ Bloques: 32-64B.
- ▶ Traducción: Algoritmo de correspondencia.
- ▶ Escritura inmediata o diferida.

Memoria virtual

- ▶ Incrementar el espacio direccionable
- ▶ Transferencia por páginas.
- ▶ Páginas: 4-8 KB.
- ▶ Traducción: Totalmente asociativa.
- ▶ 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