

Lección 5 (a)

La gestión de memoria

Diseño de Sistemas Operativos
Grado en Ingeniería Informática

Objetivos

- ▶ Conocimiento de los métodos de gestión interna de recursos en un sistema operativo.
- ▶ Usar herramientas de monitorización, gestión y ajuste de sistemas operativos.

Lecturas recomendadas

Base



1. Carretero 2007:

1. Cap.4

Recomendada



1. Tanenbaum 2006(en):

1. Cap.4

2. Stallings 2005:

1. Parte tres

3. Silberschatz 2006:

1. Cap. 4

A recordar...

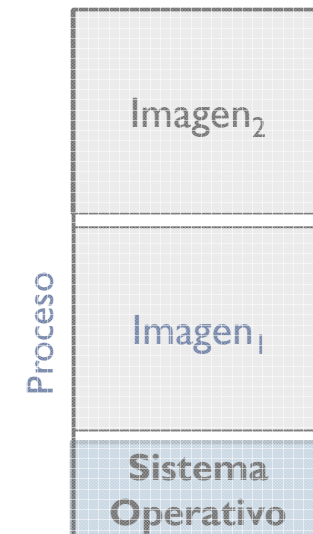
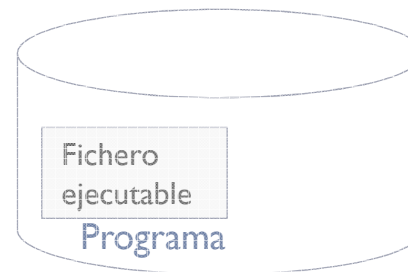
1. Estudiar la teoría asociada.
 - ▶ Estudiar el material asociado a la bibliografía: las transparencias solo no son suficiente.
 - ▶ Crear cuestiones con sus respuestas y justificación.
2. Repasar lo visto en clase.
 - ▶ Realizar el cuaderno de prácticas progresivamente.
3. Ejercitar las competencias.
 - ▶ Realizar las prácticas progresivamente.
 - ▶ Realizar todos los ejercicios posibles.

Contenidos

1. Introducción
2. Regiones de memoria de un proceso
3. Preparación de un ejecutable
4. Soporte para memoria virtual

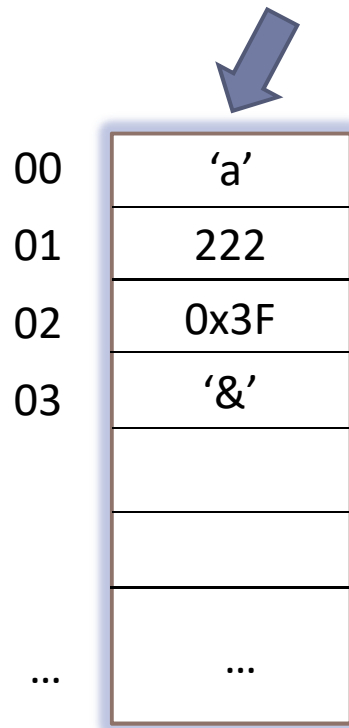
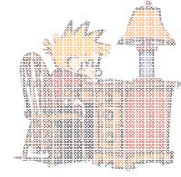
Contenidos

1. **Introducción**
2. Regiones de memoria de un proceso
3. Preparación de un ejecutable
4. Soporte para memoria virtual



Uso básico de la memoria

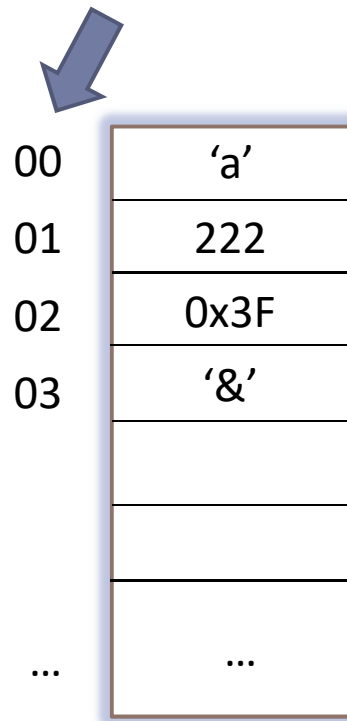
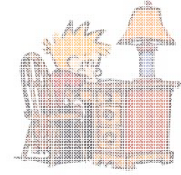
dirección, valor y tamaño



- **Valor**
 - Elemento guardado en memoria.

Uso básico de la memoria

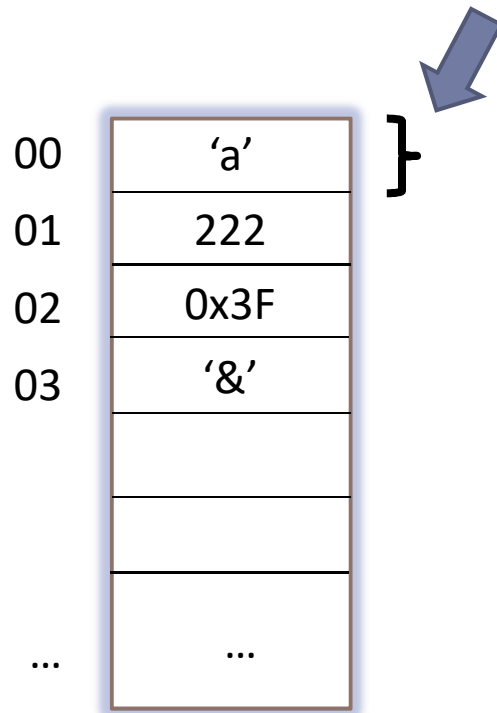
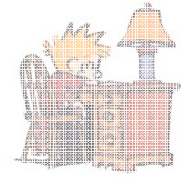
dirección, valor y tamaño



- ▶ **Valor**
 - ▶ Elemento guardado en memoria.
- ▶ **Dirección**
 - ▶ Posición de memoria.

Uso básico de la memoria

dirección, valor y tamaño



► Valor

- Elemento guardado en memoria.

► Dirección

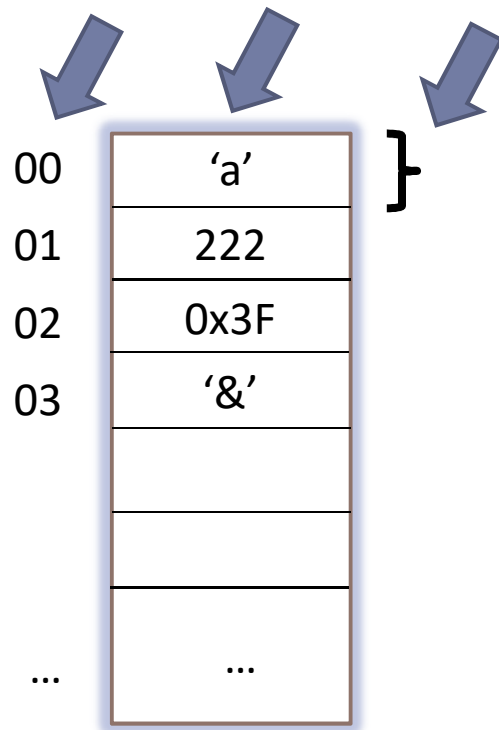
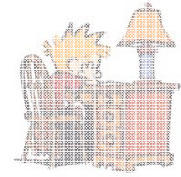
- Posición de memoria.

► Tamaño

- Número de bytes necesarios para almacenar el valor.

Uso básico de la memoria

dirección, valor y tamaño



► Valor

- Elemento guardado en memoria a partir de una dirección, y que ocupa un cierto tamaño para ser almacenada.

► Dirección

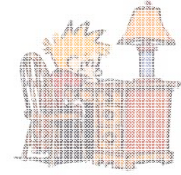
- Número que identifica la posición de memoria (celda) a partir de la cual se almacena el valor de un cierto tamaño.

► Tamaño

- Número de bytes necesarios a partir de la dirección de comienzo para almacenar el valor.

Uso básico de la memoria

interfaz funcional



00	'a'
01	222
02	0x3F
03	'&'
...	...

- ▶ **valor = read (dirección)**
- ▶ **write (dirección, valor)**
- ▶ Antes de acceder a una dirección, tiene que apuntar a una zona de memoria previamente reservada.

Introducción

- ▶ Definiciones

Programa

Imagen de proceso

Proceso

- ▶ Entornos

monoprogramados

multiprogramados

Introducción

- ▶ Definiciones

Programa

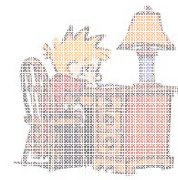
Imagen de proceso

Proceso

- ▶ Entornos

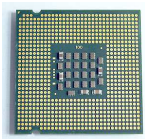
monoprogramados

multiprogramados

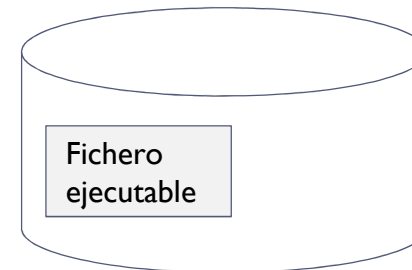


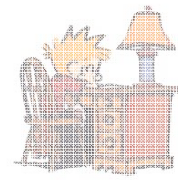
Programa y proceso

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



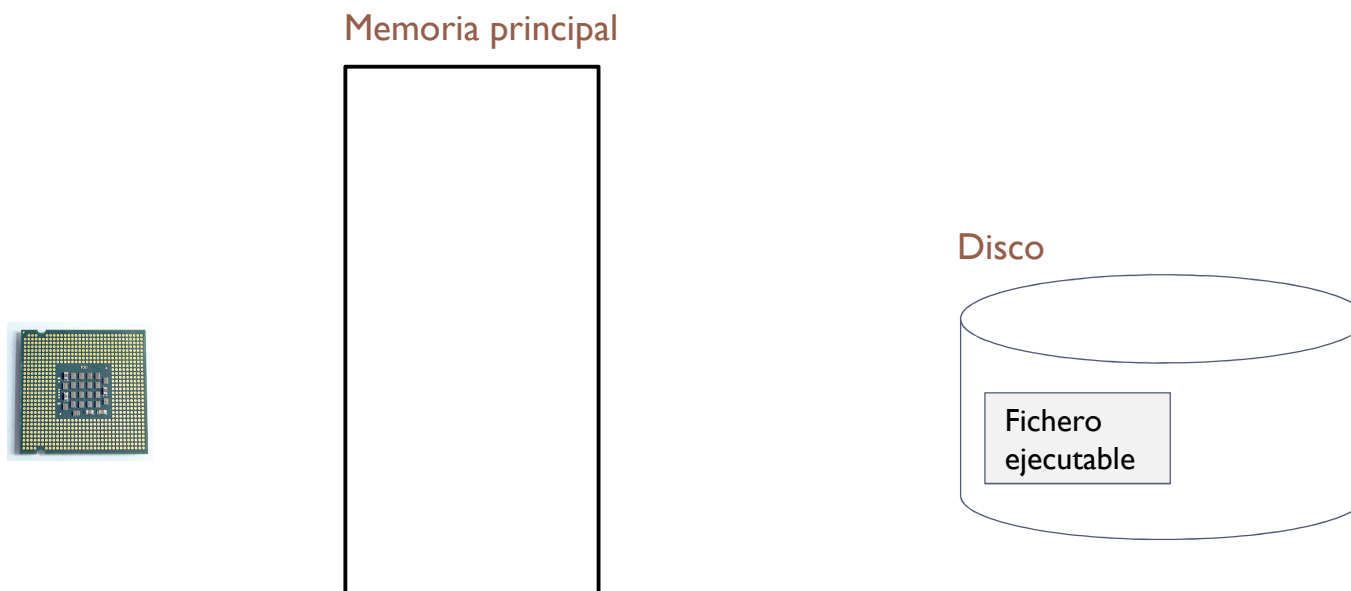
Disco

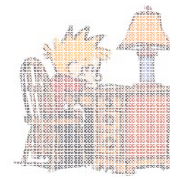




Programa y proceso

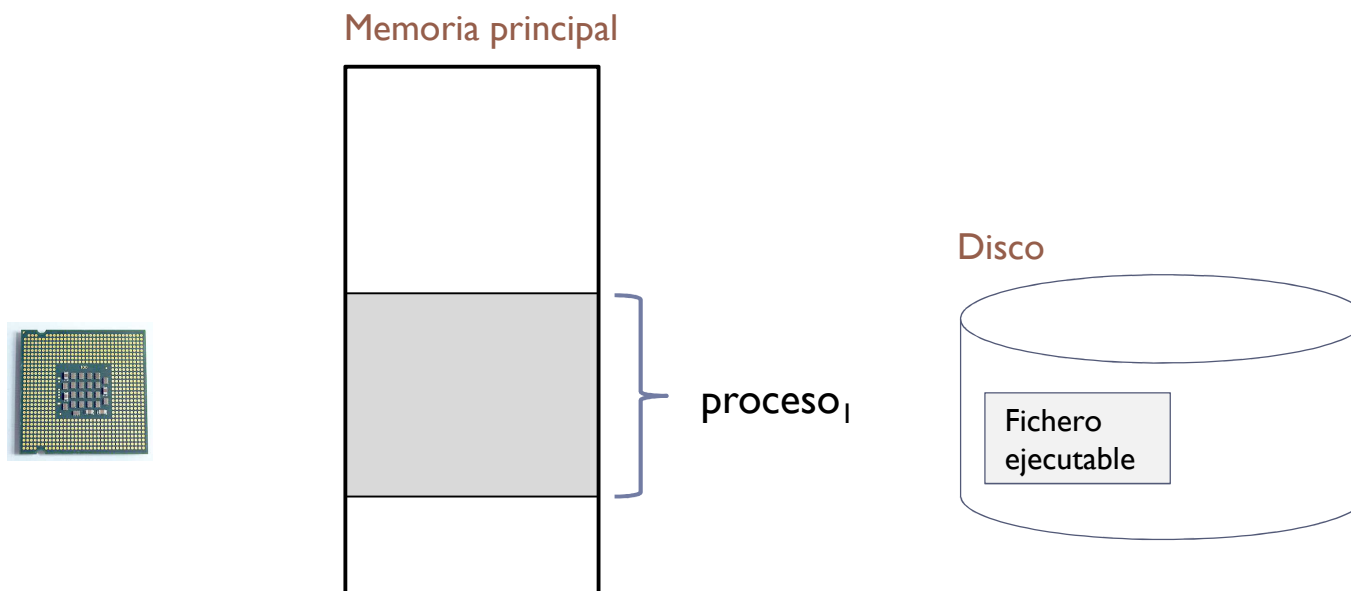
- ▶ **Programa:** conjunto de datos e instrucciones ordenadas que permiten realizar una tarea o trabajo específico.
 - ▶ Para su ejecución, ha de estar en memoria.

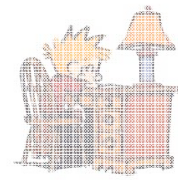




Programa y proceso

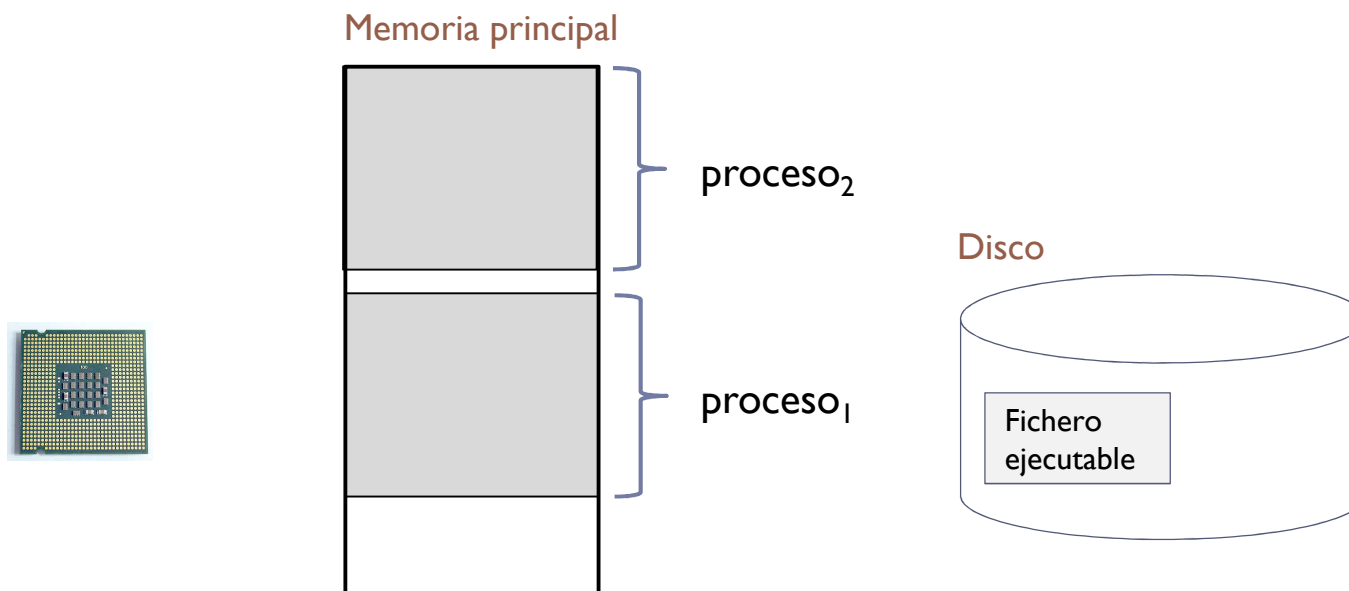
- **Proceso:** programa en ejecución.





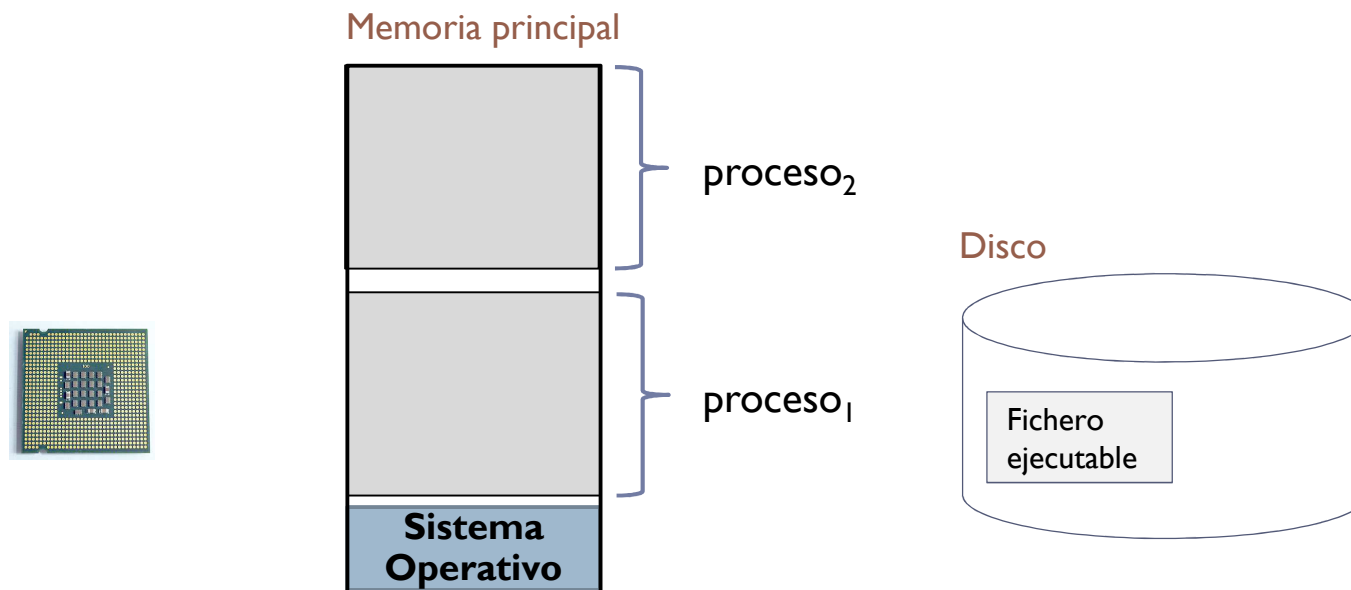
Programa y proceso

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



Programa y proceso

- ▶ **Proceso:** programa en ejecución.
 - ▶ El sistema operativo está presente en memoria y se encargará de la gestión de la memoria para repartir su uso entre todos los procesos (de forma similar a cómo lo hace con la CPU).



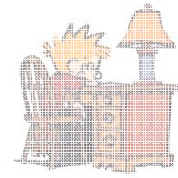
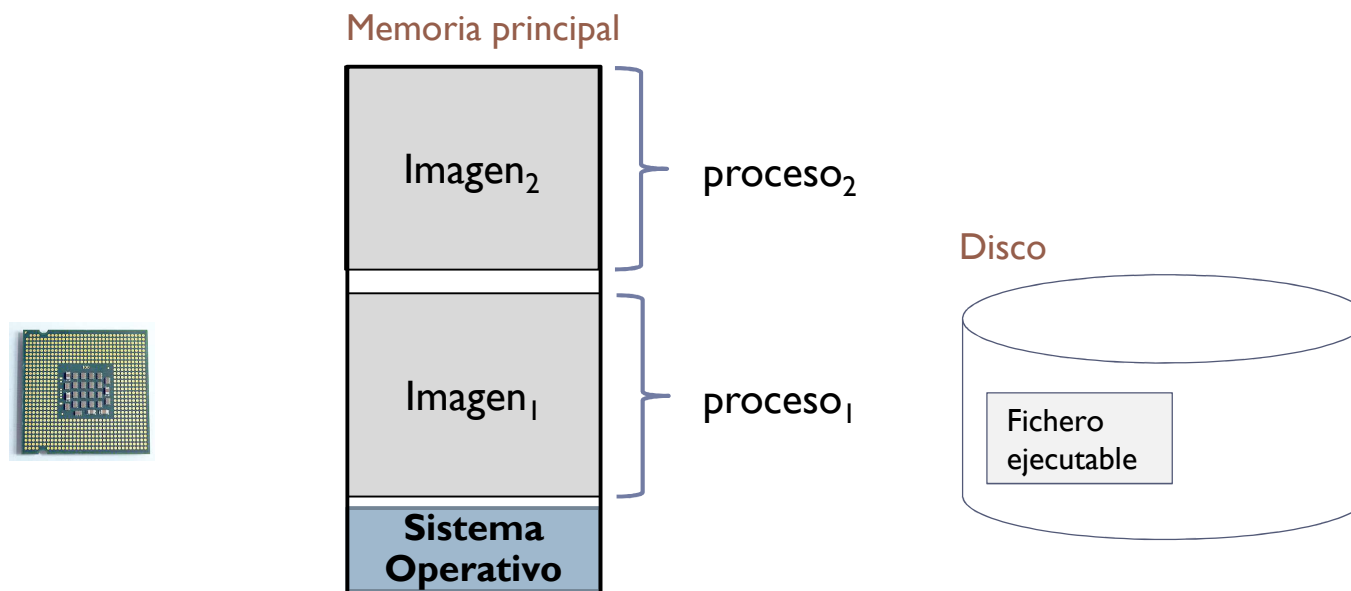


Imagen de un proceso

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



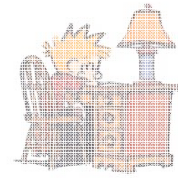
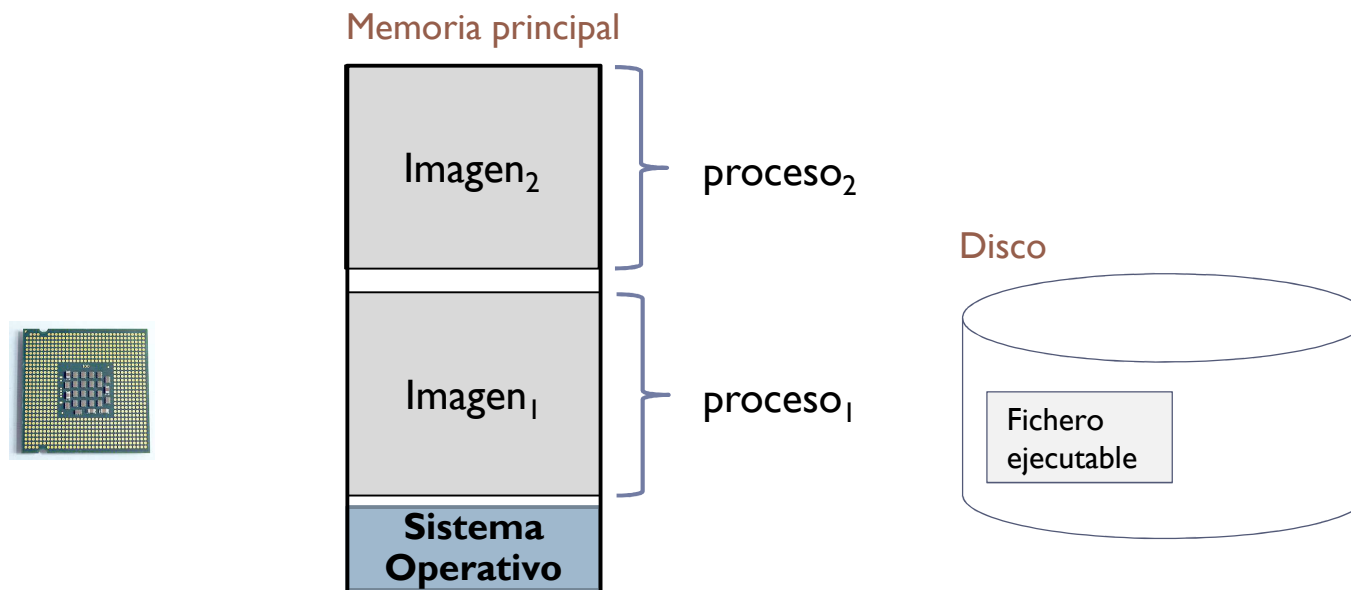


Imagen de un proceso

- ▶ **Imagen de memoria:** conjunto de direcciones de memoria asignadas al programa que se está ejecutando (y contenido).
 - ▶ Parte de la información de gestión no está en el BCP por razones de eficiencia y compartición.



Introducción

- Definiciones

Programa

Imagen de proceso

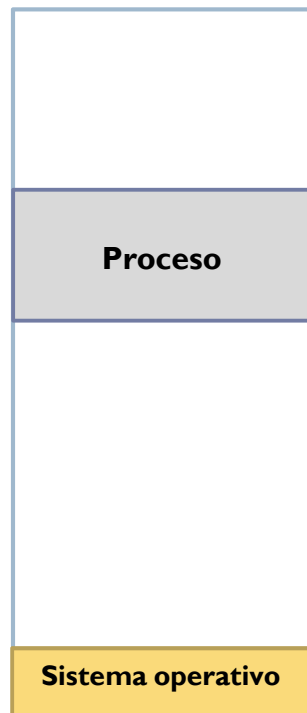
Proceso

- Entornos

monoprogramados

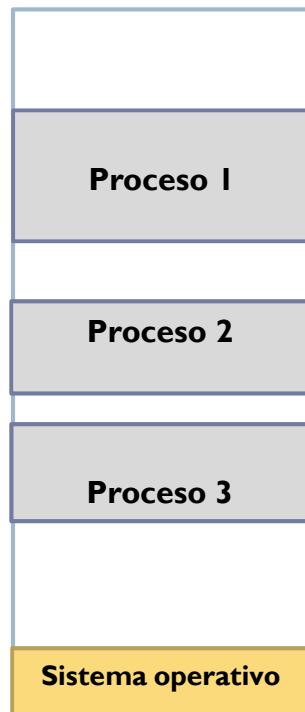
multiprogramados

Sistemas monoprogramados



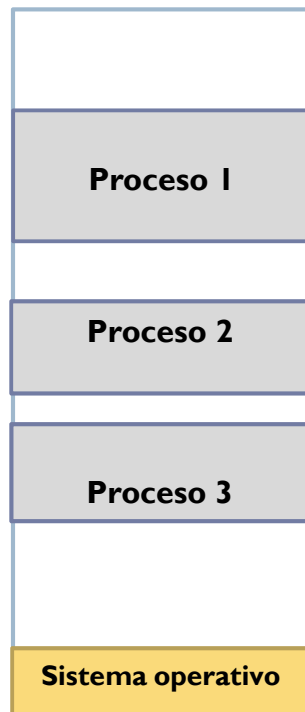
- ▶ Ejecución un proceso como máximo
- ▶ Memoria compartida entre el sistema operativo y el proceso
- ▶ Ej.: MS-DOS, DR-DOS, etc.

Sistemas multiprogramados



- ▶ Se mantiene más de un proceso en memoria
- ▶ Se mejoran la ocupación de CPU:
 - ▶ proceso bloqueado -> ejecuta otro
- ▶ La gestión de memoria es una tarea de optimización bajo restricciones
- ▶ Ej.: [Unix](#), [Windows NT](#), etc.

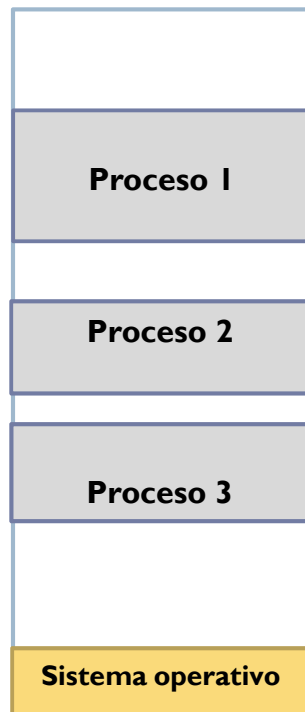
Sistemas multiprogramados



- ▶ Se mantiene más de un proceso en memoria
- ▶ Se mejorar ocupación de CPU:
 - ▶ proceso bloqueado -> ejecuta otro
- ▶ La gestión de memoria es una tarea de optimización bajo restricciones
- ▶ Ej.: [Unix](#), [Windows NT](#), etc.

Sistemas multiprogramados

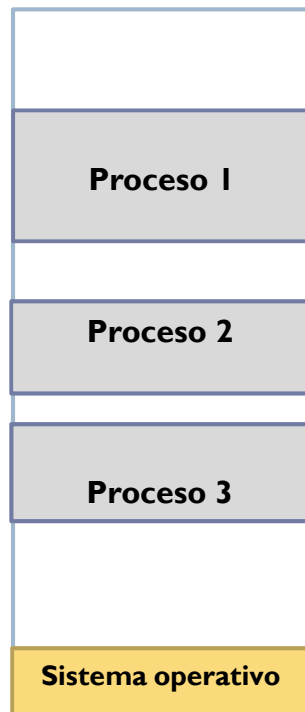
ejemplo de cálculo de utilización



- ▶ $n = 5$ procesos independientes
- ▶ $p = 0,8$ del tiempo bloqueado (20% en CPU)
- ▶ **utilización = $5 * 20\% \rightarrow 100\%$**

Sistemas multiprogramados

ejemplo de cálculo de utilización



- ▶ p = % tiempo un proceso está bloqueado
- ▶ p^n = probabilidad de que n procesos independientes estén todos bloqueados
- ▶ $1 - p^n$ = probabilidad de que la CPU **no** esté ociosa (no *idle*)
- ▶ $n = 5$ procesos independientes
- ▶ $p = 0,8$ del tiempo bloqueado (20% en CPU)
- ▶ **utilización = $5 * 20\% \rightarrow 100\%$**
- ▶ **utilización = $1 - 0,8^5 \rightarrow 67\%$**
 $1 - 0,8^{10} \rightarrow 89\%$

Introducción

resumen

- ▶ Definiciones

Programa

Imagen de proceso

Proceso

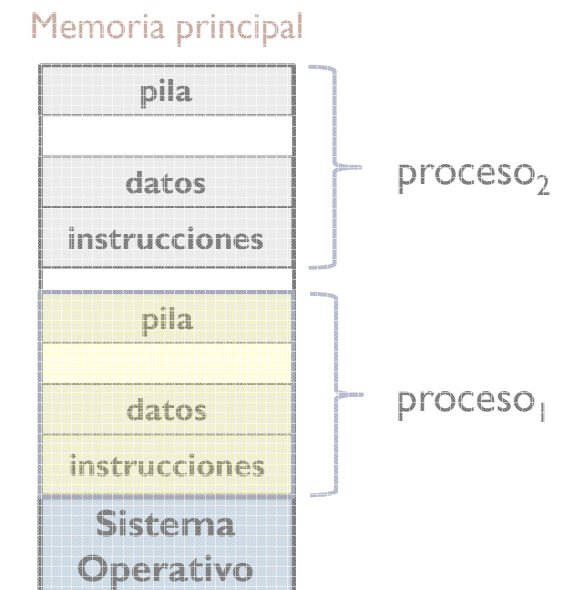
- ▶ Entornos

monoprogramados

multiprogramados

Contenidos

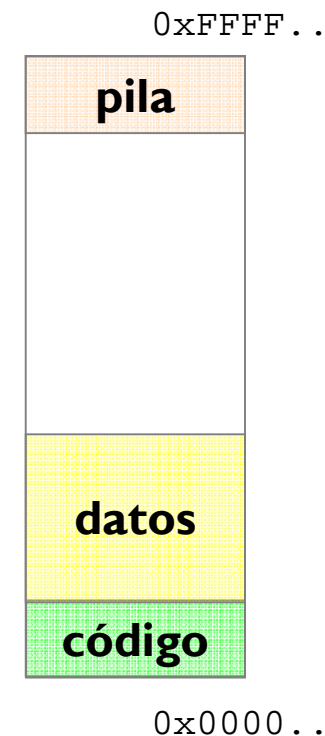
1. Introducción
2. **Regiones de memoria de un proceso**
3. Preparación de un ejecutable
4. Soporte para memoria virtual



Organización lógica (de los programas)

modelo de memoria de un proceso

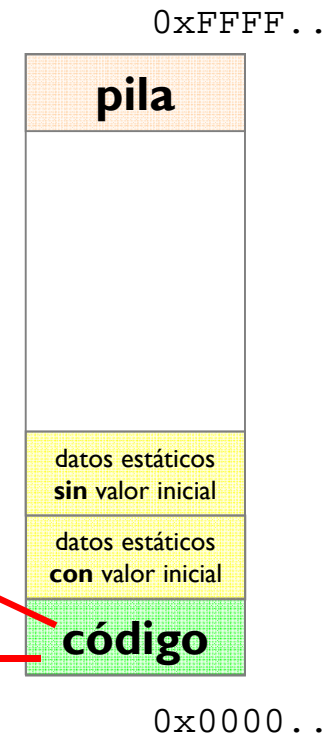
- ▶ Un proceso está formado por una serie de regiones.
- ▶ Una **región** es una **zona contigua** del espacio de direcciones de un **proceso con las mismas propiedades**.
- ▶ Principales propiedades:
 - ▶ Permisos: lectura, escritura y ejecución.
 - ▶ Compartición entre hilos: *private* o *shared*
 - ▶ Tamaño (fijo/variable)
 - ▶ Valor inicial (con/sin soporte)
 - ▶ Creación estática o dinámica
 - ▶ Sentido de crecimiento



Principales regiones de un proceso

código (*text*)

```
int a;  
int b = 5;  
  
void f(int c) {  
    int d;  
    static e = 2;  
  
    b = d + 5;  
    .....  
    return;  
}  
  
main (int argc, char **argv) {  
    char *p;  
    p = (char *) malloc (1024)  
    f(b)  
    .....  
    free (p)  
    ....  
    exit (0)  
}
```



Principales regiones de un proceso

código (*text*)

Código

- Estático
- Se conoce en tiempo de compilación
- Secuencia de instrucciones a ser ejecutadas

```
int a;  
int b = 5;  
  
void f(int c) {  
    int d;  
    static e = 2;  
  
    b = d + 5;  
    .....  
    return;  
}  
  
main (int argc, char **argv) {  
    char *p;  
    p = (char *) malloc (1024)  
    f(b)  
    .....  
    free (p)  
    ....  
    exit (0)  
}
```

0xFFFF..

pila

datos estáticos
sin valor inicial

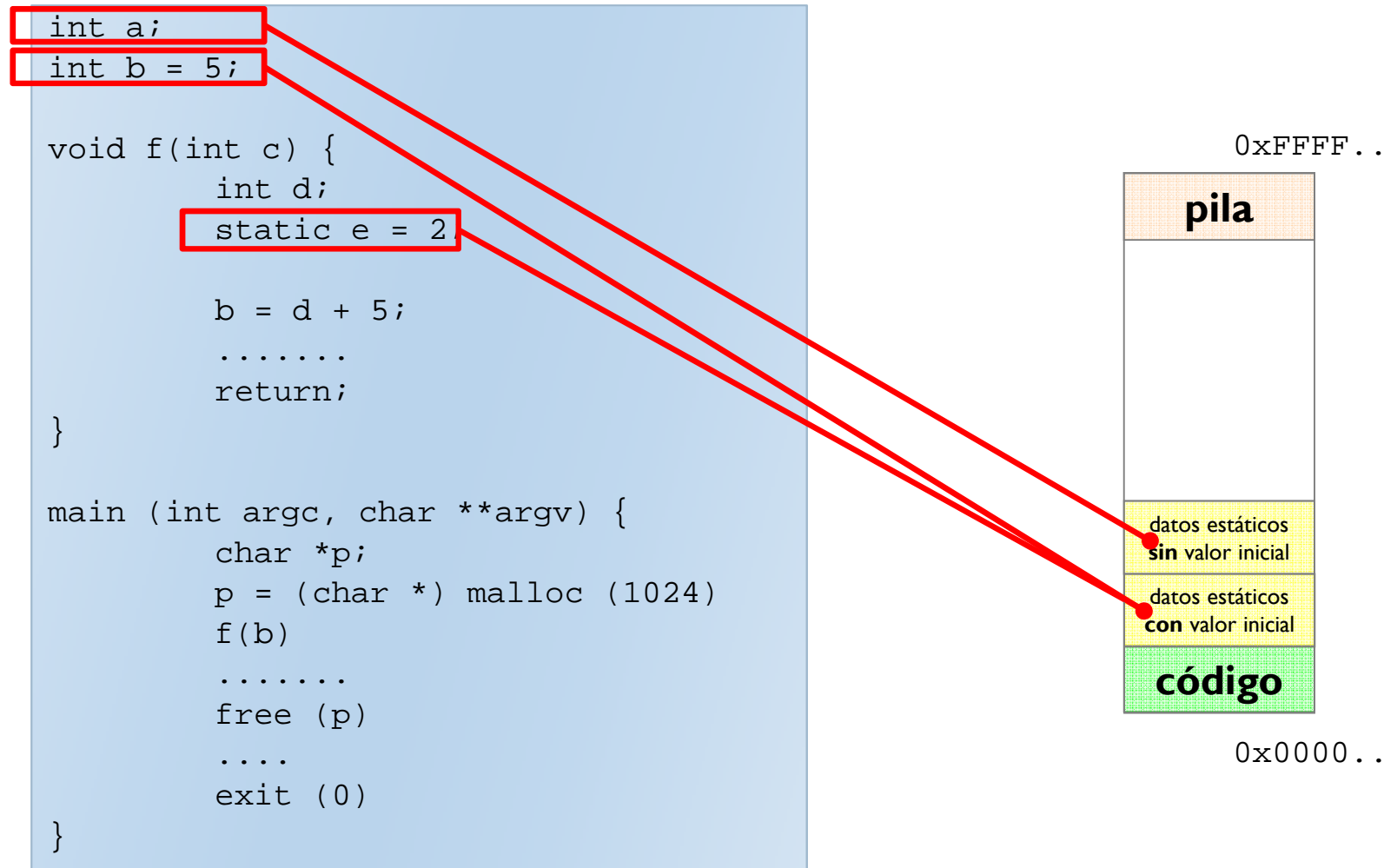
datos estáticos
con valor inicial

código

0x0000..

Principales regiones de un proceso

datos (*data*)



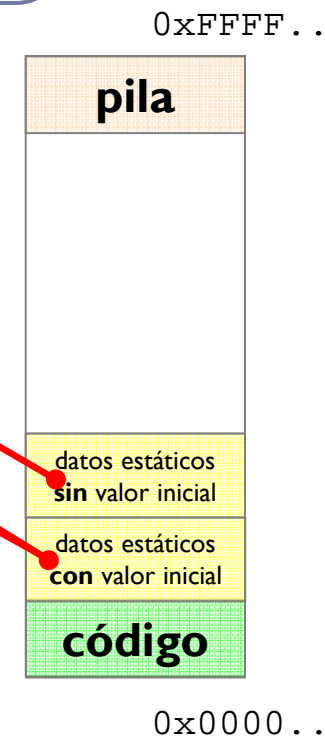
Principales regiones de un proceso

datos (*data*)

Variable globales

- Estáticas
- Se crean al iniciar el programa
- Existen durante ejecución
- Dirección fija en memoria y ejecutable

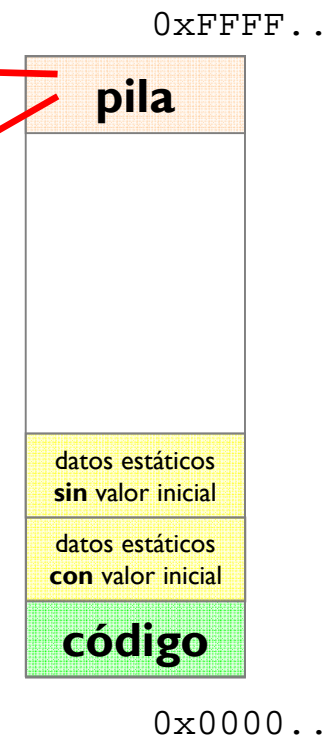
```
int a;  
int b = 5;  
  
void f(int c) {  
    int d;  
    static e = 2;  
  
    b = d + 5;  
    .....  
    return;  
}  
  
main (int argc, char **argv) {  
    char *p;  
    p = (char *) malloc (1024)  
    f(b)  
    .....  
    free (p)  
    ....  
    exit (0)  
}
```



Principales regiones de un proceso

pila (*stack*)

```
int a;  
int b = 5;  
  
void f(int c) {  
    int d;  
    static e = 2;  
  
    b = d + 5;  
    .....  
    return;  
}  
  
main (int argc, char **argv) {  
    char *p;  
    p = (char *) malloc (1024)  
    f(b)  
    .....  
    free (p)  
    ....  
    exit (0)  
}
```



Principales regiones de un proceso

pila (*stack*)

Variable locales y parámetros

- Dinámicas
- Se crean al invocar la función
- Se destruyen al retornar
- Recursividad: varias instancias de una variable

```
int a;  
int b = 5;  
  
void f(int c) {  
    int d;  
    static e = 2;  
  
    b = d + 5;  
    .....  
    return;  
}  
  
main (int argc, char **argv) {  
    char *p;  
    p = (char *) malloc (1024)  
    f(b)  
    .....  
    free (p)  
    ....  
    exit (0)  
}
```

0xFFFF..

pila

datos estáticos
sin valor inicial

datos estáticos
con valor inicial

código

0x0000..

Principales regiones de un proceso

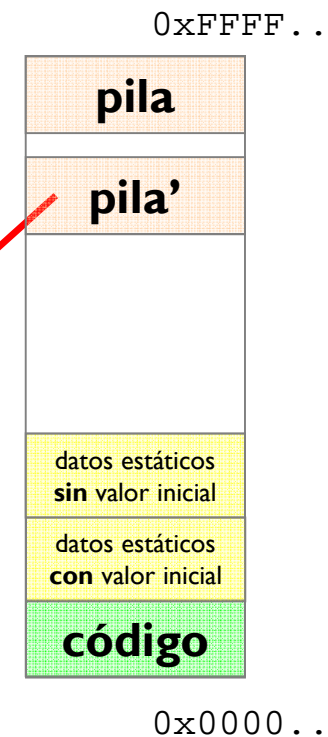
pila (*stack*)

```
int a;
int b = 5;

void f(int c) {
    int d;
    static e = 2;

    b = d + 5;
    .....
    return;
}

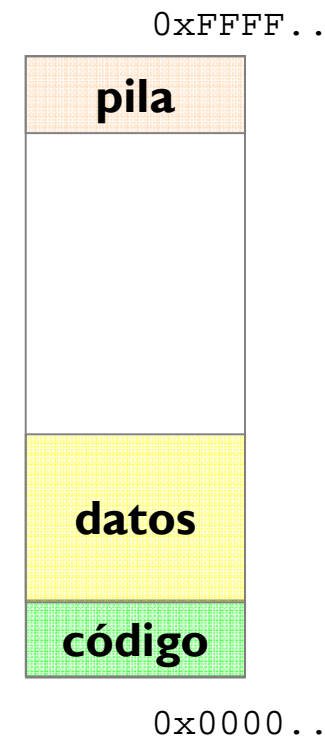
main (int argc, char **argv) {
    char *p;
    p = (char *) malloc (1024)
    f(b)
    ..... pthread_create(f...)
    free (p)
    ....
    exit (0)
}
```



Organización lógica (de los programas)

modelo de memoria de un proceso

- ▶ **Código o Texto**
 - ▶ Compartida, RX, T. Fijo, soporte en ejecutable
- ▶ **Datos**
 - ▶ Con valor inicial
 - ▶ Compartido, RW, T. Fijo, soporte en ejecutable
 - ▶ Sin valor inicial
 - ▶ Compartido, RW, T. Fijo, sin soporte (rellenar 0)
- ▶ **Pila**
 - ▶ Privada, RW, T. Variable, sin soporte (rellenar 0)
 - ▶ Crece hacia direcciones más bajas
 - ▶ Pila inicial: argumentos del programa



Organización lógica (de los programas)

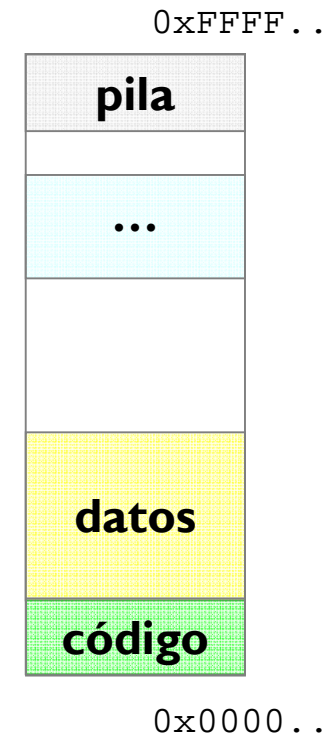
modelo de memoria de un proceso

► Región de *Heap*

- Soporte de memoria dinámica (malloc en C)
- Compartido, RW, T.Variable, sin soporte (rellenar 0)
- Crece hacia direcciones más altas

► Ficheros proyectados

- Región asociada al fichero proyectado
- Privado/Compartido, T.Variable, soporte en fichero
- Protección especificada en proyección



Organización lógica (de los programas)

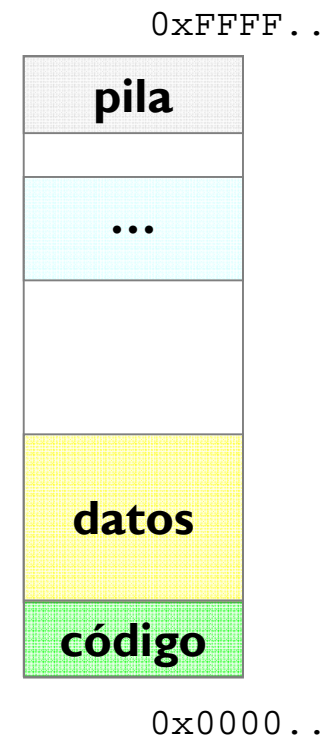
modelo de memoria de un proceso

► Región de *Heap*

- Soporte de memoria dinámica (malloc en C)
- Compartido, RW, T.Variable, sin soporte (rellenar 0)
- Crece hacia direcciones más altas

► Ficheros proyectados

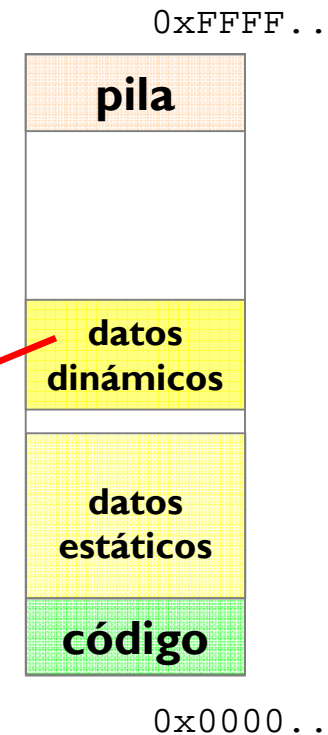
- Región asociada al fichero proyectado
- Privado/Compartido, T.Variable, soporte en fichero
- Protección especificada en proyección



Principales regiones de un proceso

datos dinámicos (*heap*)

```
int a;  
int b = 5;  
  
void f(int c) {  
    int d;  
    static e = 2;  
  
    b = d + 5;  
    .....  
    return;  
}  
  
main (int argc, char **argv) {  
    char *p;  
    p = (char *) malloc (1024)  
    f(b)  
    .....  
    free (p)  
    ....  
    exit (0)  
}
```



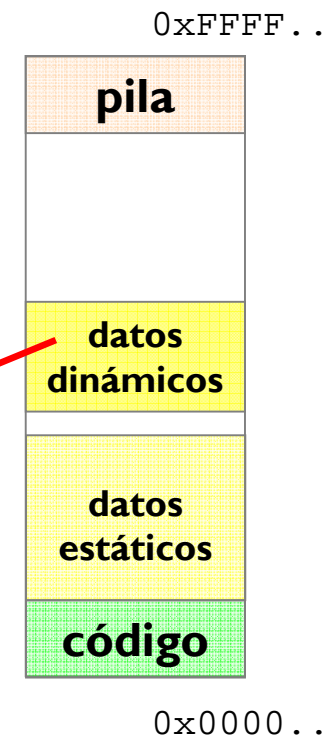
Principales regiones de un proceso

datos dinámicos (*heap*)

Variable dinámicas

- Variables locales o globales sin espacio asignado en tiempo de compilación
- Se reserva (y libera) espacio en tiempo de ejecución

```
int a;  
int b = 5;  
  
void f(int c) {  
    int d;  
    static e = 2;  
  
    b = d + 5;  
    .....  
    return;  
}  
  
main (int argc, char **argv) {  
    char *p;  
    p = (char *) malloc (1024)  
    f(b)  
    .....  
    free (p)  
    ....  
    exit (0)  
}
```



Organización lógica (de los programas)

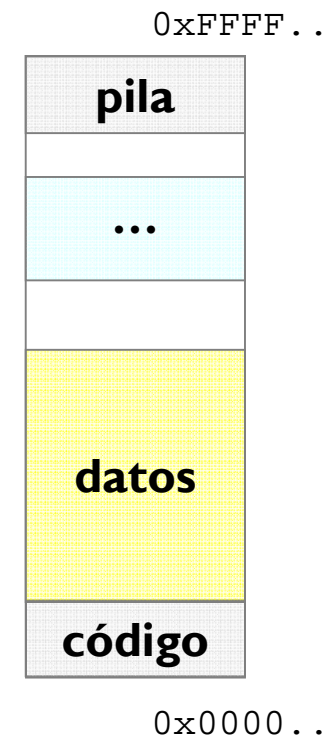
modelo de memoria de un proceso

► Región de *Heap*

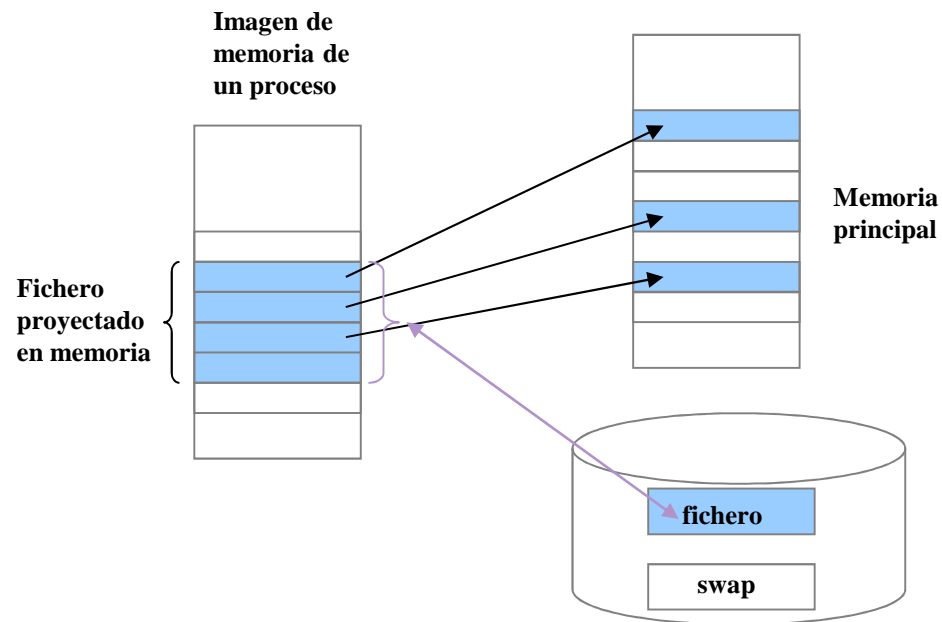
- Soporte de memoria dinámica (malloc en C)
- Compartido, RW, T.Variable, sin soporte (rellenar 0)
- Crece hacia direcciones más altas

► **Ficheros proyectados**

- Región asociada al fichero proyectado
- Privado/Compartido, T.Variable, soporte en fichero
- Protección especificada en proyección



Ficheros proyectados en memoria (1/3)



- ▶ Una región de un proceso se asocia a un fichero
- ▶ Habrá páginas del fichero en memoria principal
- ▶ El proceso direcciona dentro del fichero con instrucciones de acceso a memoria (en lugar de read/write)

Ficheros proyectados en memoria (2/3)

*void *mmap(void *addr, size_t len, int prot, int flags, int fildes, off_t off);*

- ▶ Establece una proyección entre el espacio de direcciones de un proceso y un descriptor de fichero u objeto de memoria compartida.
 - ▶ Devuelve la dirección de memoria donde se ha proyectado el fichero.
 - ▶ *addr* dirección donde proyectar. Si `NULL` el SO elige una.
 - ▶ *len* especifica el número de bytes a proyectar.
 - ▶ *prot* el tipo de acceso (lectura, escritura o ejecución).
 - ▶ *flags* especifica información sobre el manejo de los datos proyectados (compartidos, privado, etc.).
 - ▶ *fildes* representa el descriptor de fichero del fichero o descriptor del objeto de memoria a proyectar.
 - ▶ *off* desplazamiento dentro del fichero a partir del cual se realiza la proyección.

*void munmap(void *addr, size_t len);*

- ▶ Desproyecta parte del espacio de direcciones de un proceso comenzando en la dirección *addr*.

Ficheros proyectados en memoria (3/3)

- ▶ Cuántas veces aparece carácter en fichero proyectando en memoria.

```
/* 1) Abrir el fichero */
fd=open(argv[2], O_RDONLY); /* Abre fichero */
fstat(fd, &fs); /* Averigua long. fichero */

/* 2) Proyectar el fichero */
org=mmap((caddr_t)0, fs.st_size, PROT_READ, MAP_SHARED, fd, 0);
close(fd); /* Se cierra el fichero */

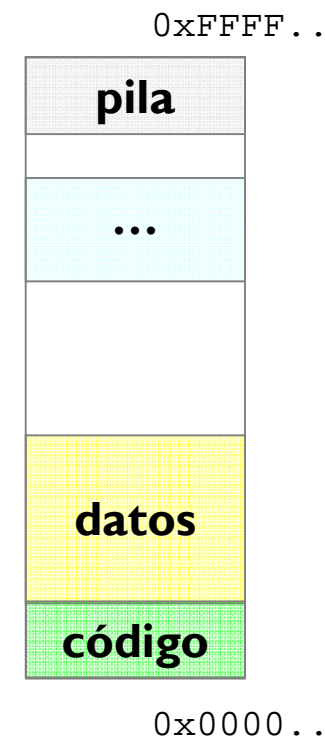
/* 3) Bucle de acceso */
p=org;
for (i=0; i<fs.st_size; i++)
    if (*p++==caracter) contador++;

/* 4) Eliminar la proyección */
munmap(org, fs.st_size);
printf("%d\n", contador);
```

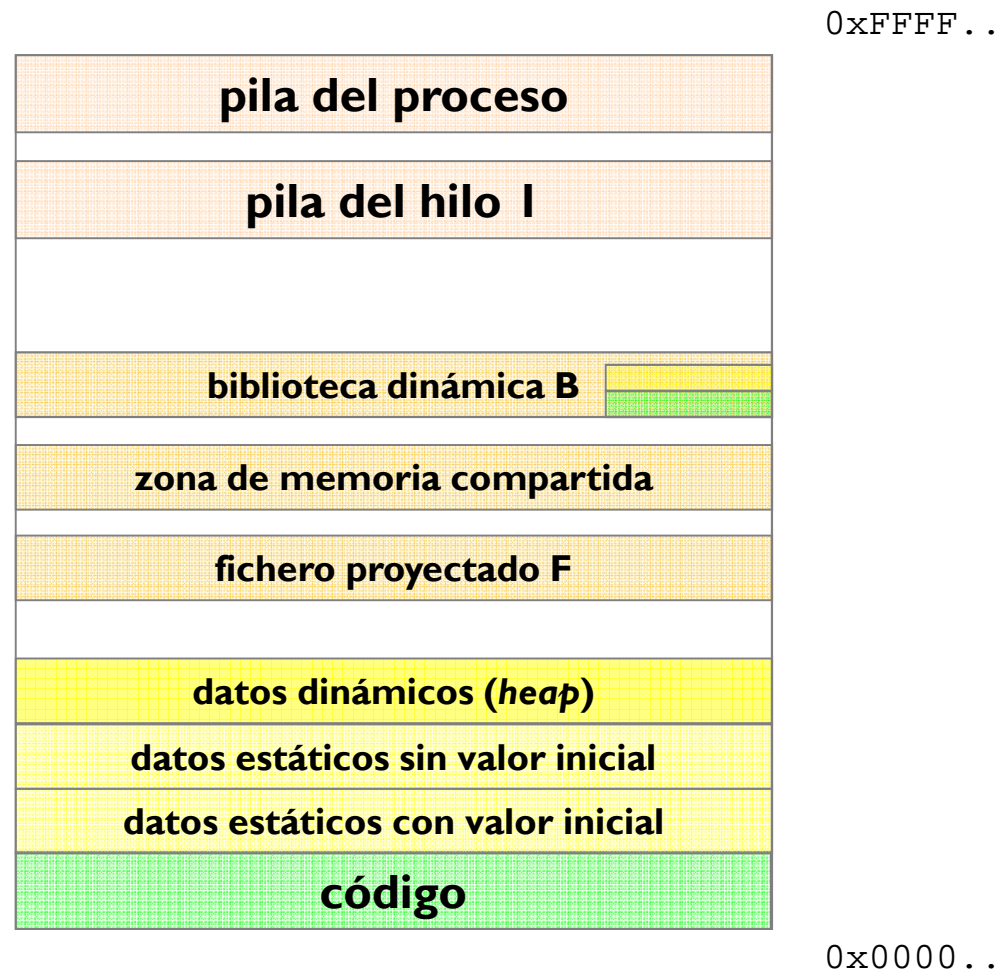
Organización lógica (de los programas)

modelo de memoria de un proceso

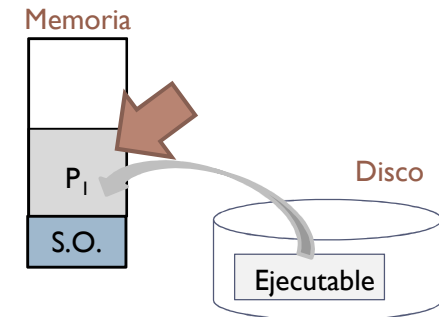
- ▶ **Región de *Heap***
 - ▶ Soporte de memoria dinámica (malloc en C)
 - ▶ Compartido, RW, T.Variable, sin soporte (rellenar 0)
 - ▶ Crece hacia direcciones más altas
- ▶ **Ficheros proyectados**
 - ▶ Región asociada al fichero proyectado
 - ▶ Privado/Compartido, T.Variable, soporte en fichero
 - ▶ Protección especificada en proyección
- ▶ **Bibliotecas dinámicas**
 - ▶ Biblioteca con código y datos **proyectados**
- ▶ **Memoria compartida**
 - ▶ Entre **distintos procesos**



Ejemplo de mapa de memoria



Inspeccionar un proceso



► Detalles de las secciones de un proceso:

```
acaldero@phoenix:~/infodso/$ cat /proc/1/maps
b7688000-b7692000 r-xp 00000000 08:02 1491      /lib/libnss_files-2.12.1.so
b7692000-b7693000 r--p 00009000 08:02 1491      /lib/libnss_files-2.12.1.so
b7693000-b7694000 rw-p 0000a000 08:02 1491      /lib/libnss_files-2.12.1.so
b7694000-b769d000 r-xp 00000000 08:02 3380      /lib/libnss_nis-2.12.1.so
b769d000-b769e000 r--p 00008000 08:02 3380      /lib/libnss_nis-2.12.1.so
b769e000-b769f000 rw-p 00009000 08:02 3380      /lib/libnss_nis-2.12.1.so
b769f000-b76b2000 r-xp 00000000 08:02 1414      /lib/libnsl-2.12.1.so
b76b2000-b76b3000 r--p 00012000 08:02 1414      /lib/libnsl-2.12.1.so
b76b3000-b76b4000 rw-p 00013000 08:02 1414      /lib/libnsl-2.12.1.so
b76b4000-b76b6000 rw-p 00000000 00:00 0
..
b78b7000-b78b8000 r-xp 00000000 00:00 0          [vdso]
b78b8000-b78d4000 r-xp 00000000 08:02 811        /lib/ld-2.12.1.so
b78d4000-b78d5000 r--p 0001b000 08:02 811        /lib/ld-2.12.1.so
b78d5000-b78d6000 rw-p 0001c000 08:02 811        /lib/ld-2.12.1.so
b78d6000-b78ef000 r-xp 00000000 08:02 1699       /sbin/init
b78ef000-b78f0000 r--p 00019000 08:02 1699       /sbin/init
b78f0000-b78f1000 rw-p 0001a000 08:02 1699       /sbin/init
b81e5000-b8247000 rw-p 00000000 00:00 0          [heap]
bf851000-bf872000 rw-p 00000000 00:00 0          [stack]
```

direccion perm. offset dev nodo-i nombre

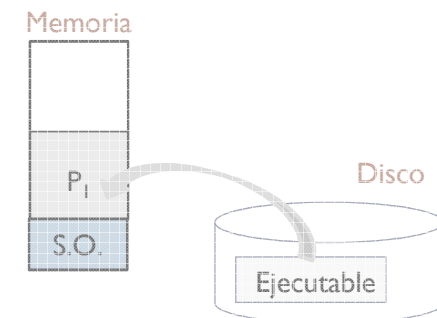
Ejercicio

completar características de regiones

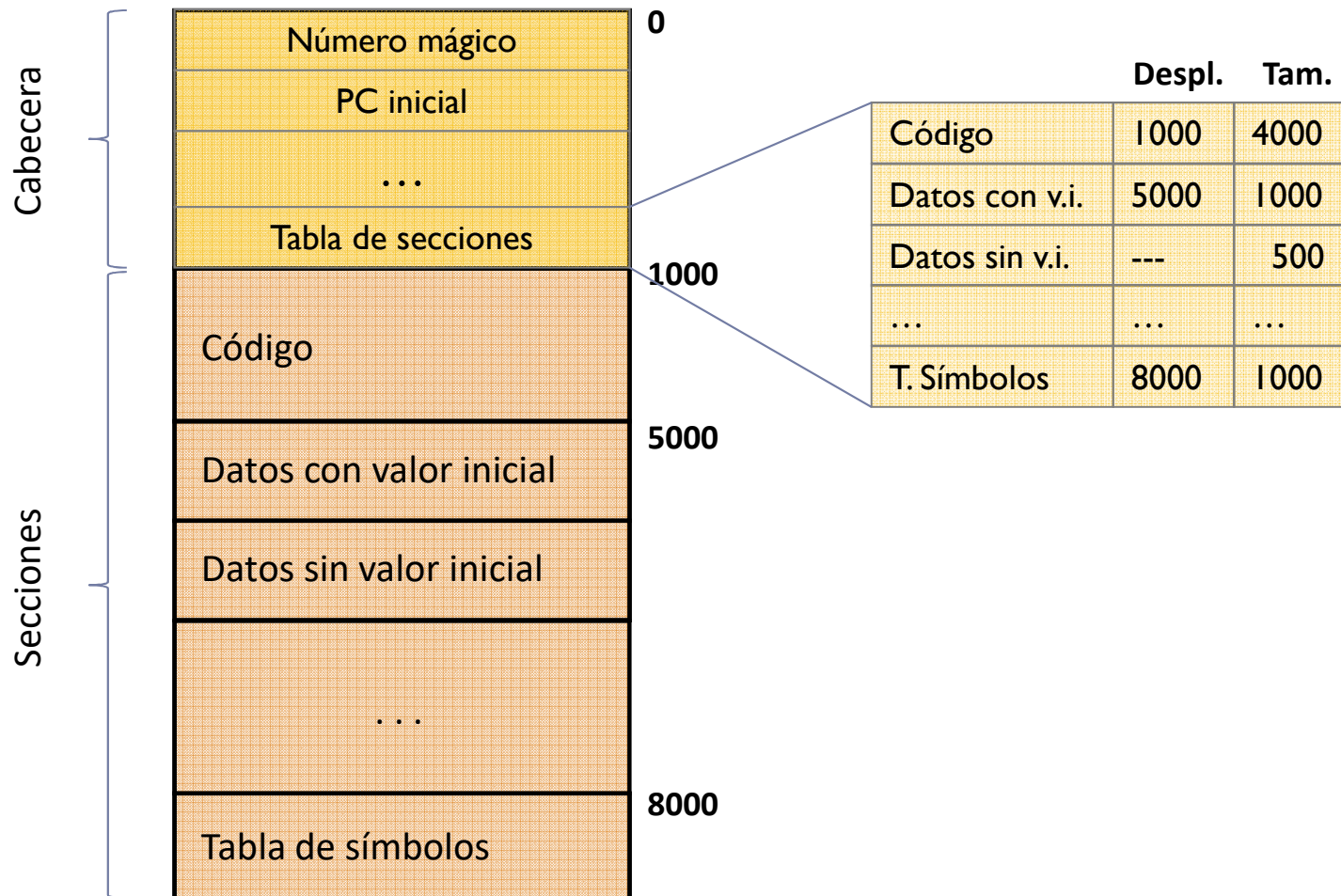
Región	Soporte	Protección	Comp./Priv.	Tamaño
Código	Fichero	RX	Compartida	Fijo
...

Contenidos

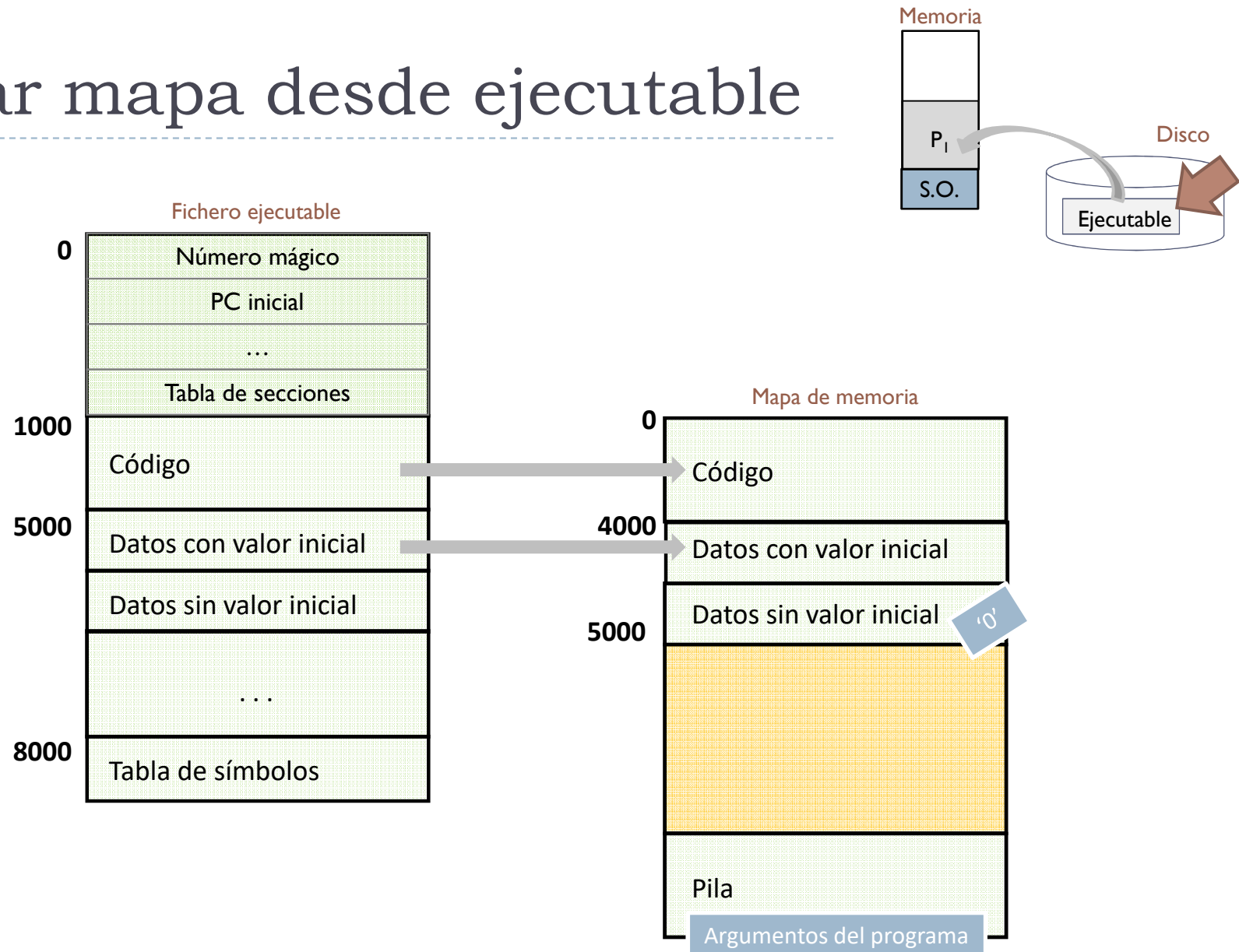
1. Introducción
2. Regiones de memoria de un proceso
3. **Preparación de un ejecutable**
4. Soporte para memoria virtual



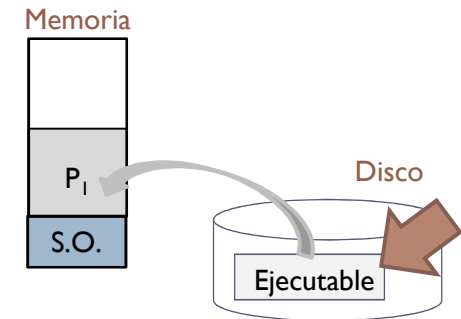
Ejemplo de formato de ejecutable



Crear mapa desde ejecutable



Inspeccionar un ejecutable



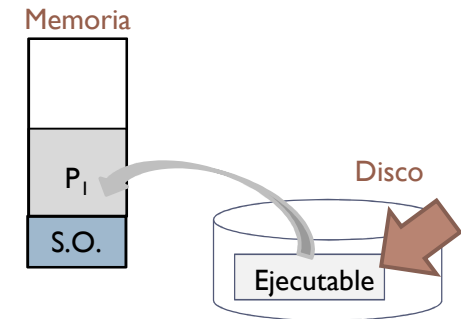
- Dependencias de un ejecutable (lib. dinámicas):

```
acaldero@phoenix:~/infodso/$ ldd main.exe
linux-gate.so.1 => (0xb7797000)
libdinamica.so.1 => not found
libc.so.6 => /lib/libc.so.6 (0xb761c000)
/lib/ld-linux.so.2 (0xb7798000)
```

- Símbolos de un ejecutable:

```
acaldero@phoenix:~/infodso/$ nm main.exe
08049f20 d _DYNAMIC
08049ff4 d _GLOBAL_OFFSET_TABLE_
0804856c R _IO_stdin_used
          w _Jv_RegisterClasses
08049f10 d __CTOR_END__
08049f0c d __CTOR_LIST__
...
```

Inspeccionar un ejecutable



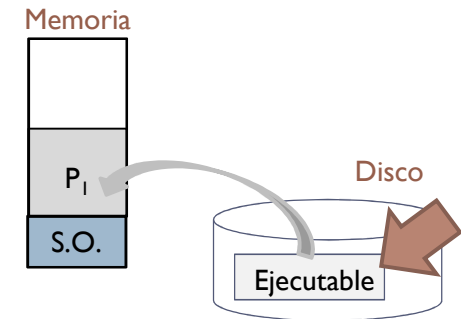
► Detalles de las secciones de un ejecutable:

```
acaldero@phoenix:~/infodso/$ objdump -x main.exe
...

Program Header:
...
    DYNAMIC off      0x00000f20 vaddr 0x08049f20 paddr 0x08049f20 align 2**2
           filesz 0x000000d0 memsz 0x000000d0 flags rw-
...
    STACK off       0x00000000 vaddr 0x00000000 paddr 0x00000000 align 2**2
           filesz 0x00000000 memsz 0x00000000 flags rw-
...

Dynamic Section:
    NEEDED          libdinamica.so
    NEEDED          libc.so.6
    INIT            0x08048368
...
```

Inspeccionar un ejecutable



(continuación)

Sections:

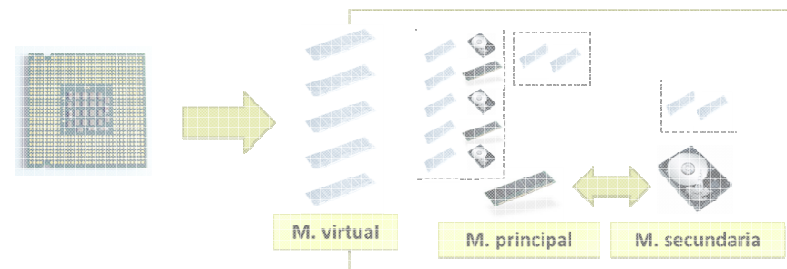
Idx	Name	Size	VMA	LMA	File off	Algn
0	.interp	00000013	08048134	08048134	00000134	2**0
CONTENTS, ALLOC, LOAD, READONLY, DATA						
...						
12	.text	0000016c	080483e0	080483e0	000003e0	2**4
CONTENTS, ALLOC, LOAD, READONLY, CODE						
...						
23	.bss	00000008	0804a014	0804a014	00001014	2**2
ALLOC						
...						

SYMBOL TABLE:

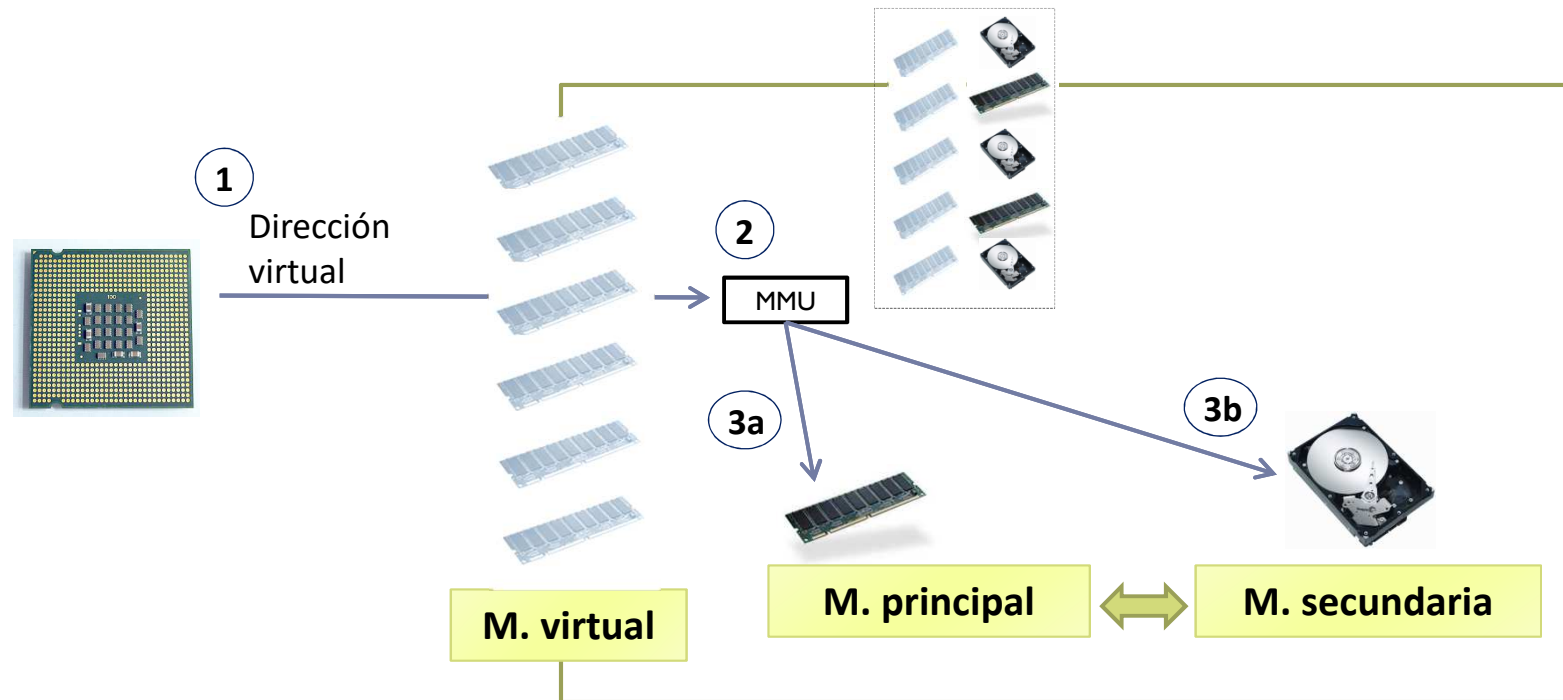
08048134	l	d	.interp	00000000	.interp
08048148	l	d	.note.ABI-tag	00000000	.note.ABI-tag
08048168	l	d	.note.gnu.build-id	00000000	.note.gnu.build-id
...					
0804851a	g	F	.text	00000000	.hidden __i686.get_pc_thunk.bx
08048494	g	F	.text	00000014	main
08048368	g	F	.init	00000000	_init

Contenidos

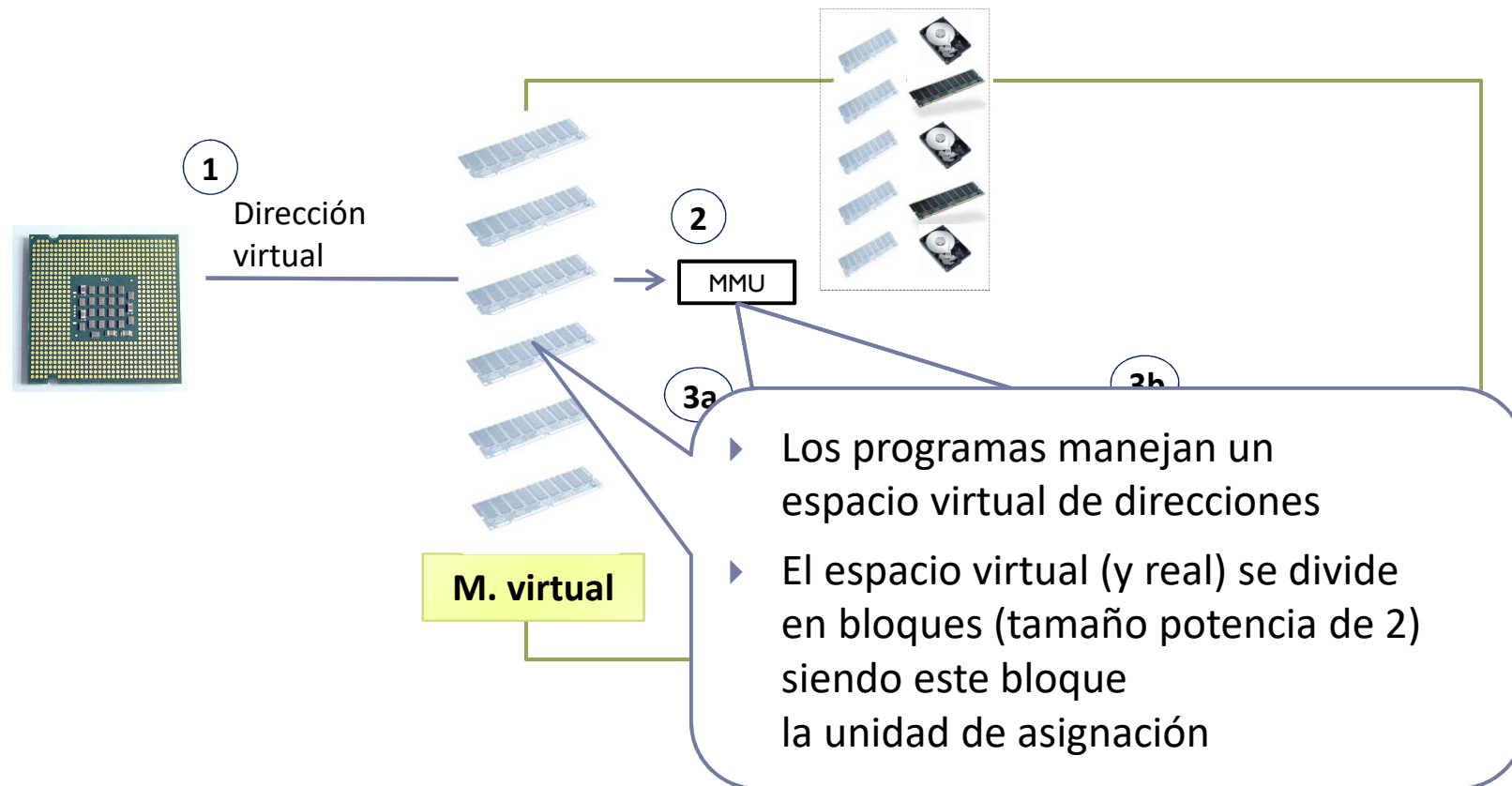
1. Introducción
2. Regiones de memoria de un proceso
3. Preparación de un ejecutable
4. **Soporte para memoria virtual**



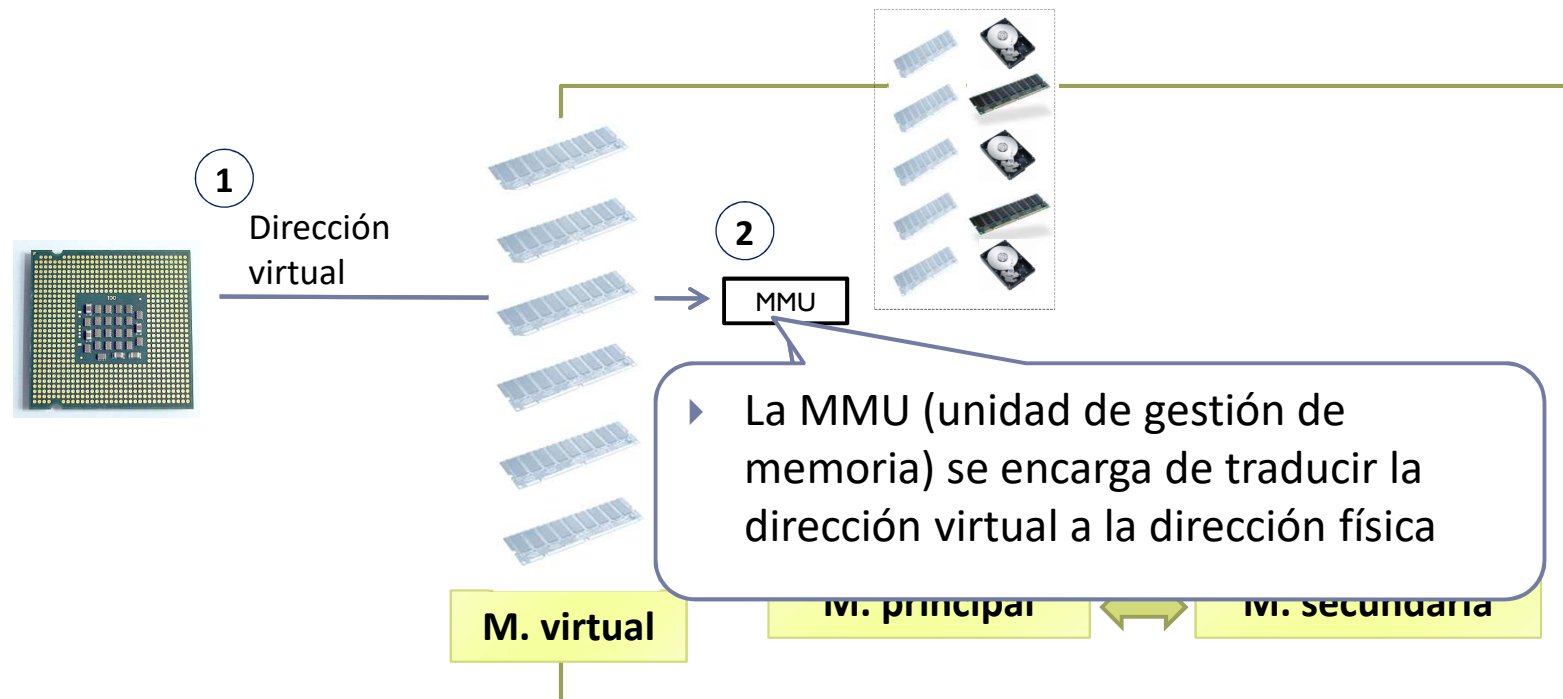
Sistemas con memoria virtual



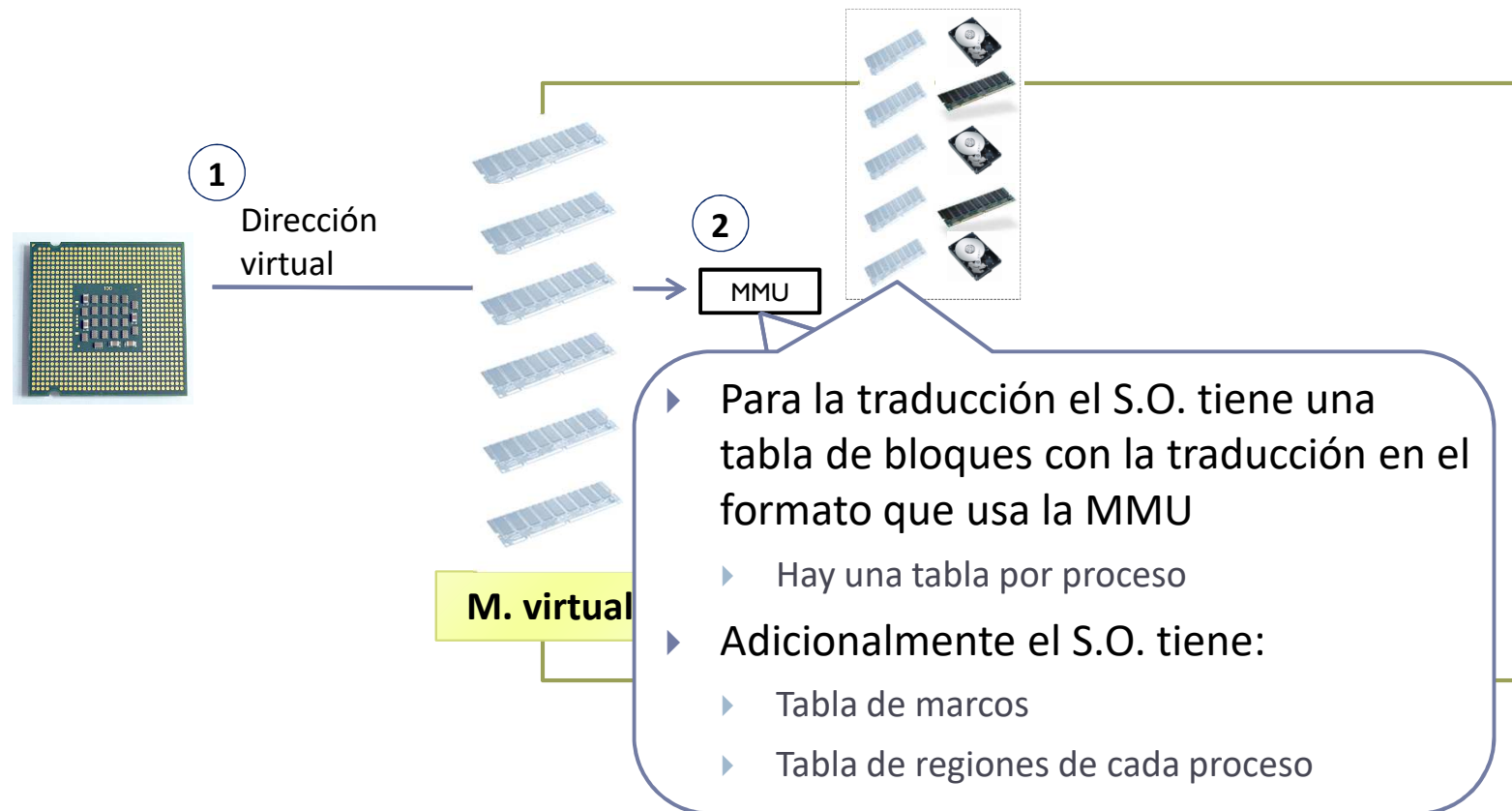
Sistemas con memoria virtual



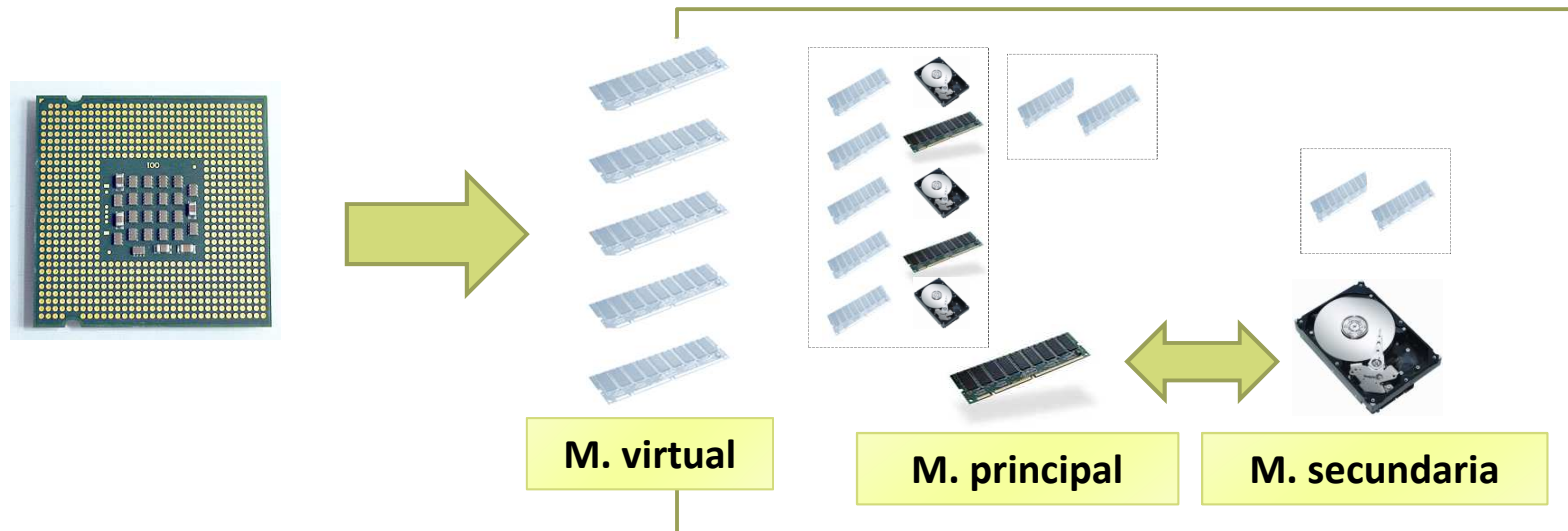
Sistemas con memoria virtual



Sistemas con memoria virtual



Introducción a memoria virtual

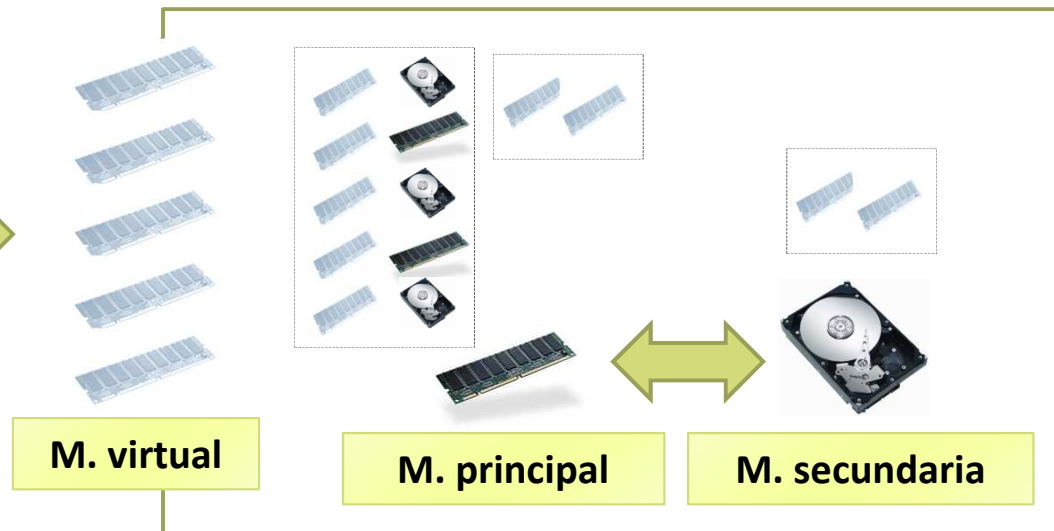
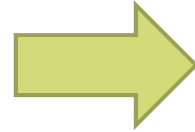
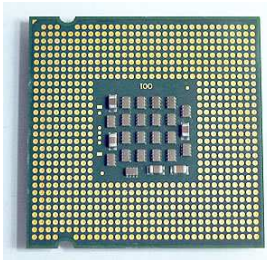


Introducción a memoria virtual

...

lw \$t0 vector

...

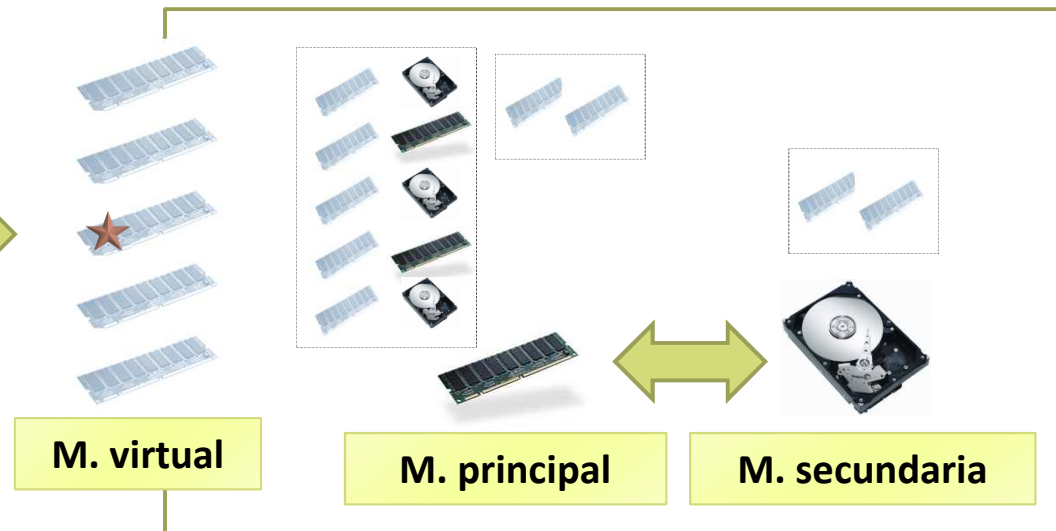
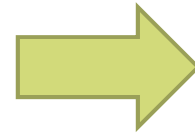
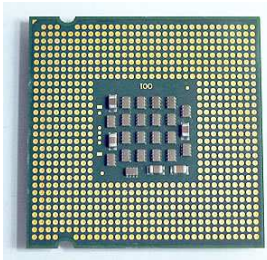


Introducción a memoria virtual

...

lw \$t0 vector

...

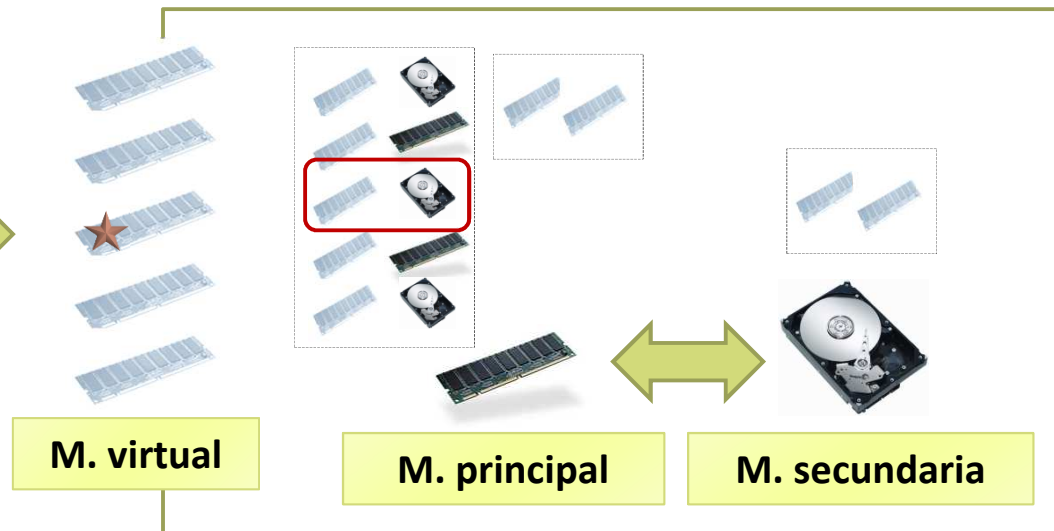
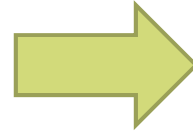
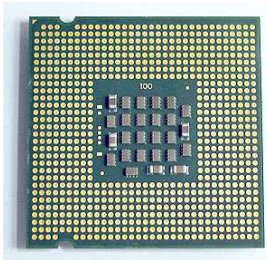


Introducción a memoria virtual

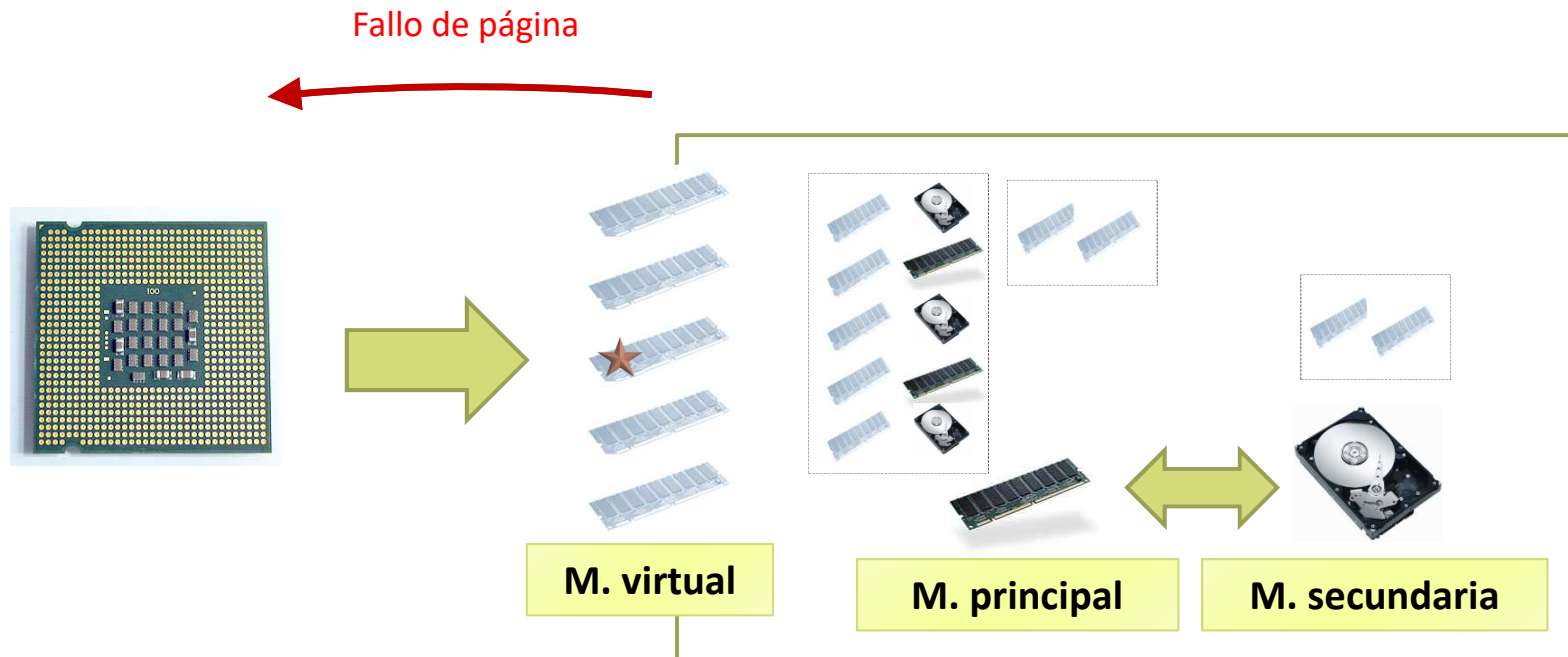
...

lw \$t0 vector

...

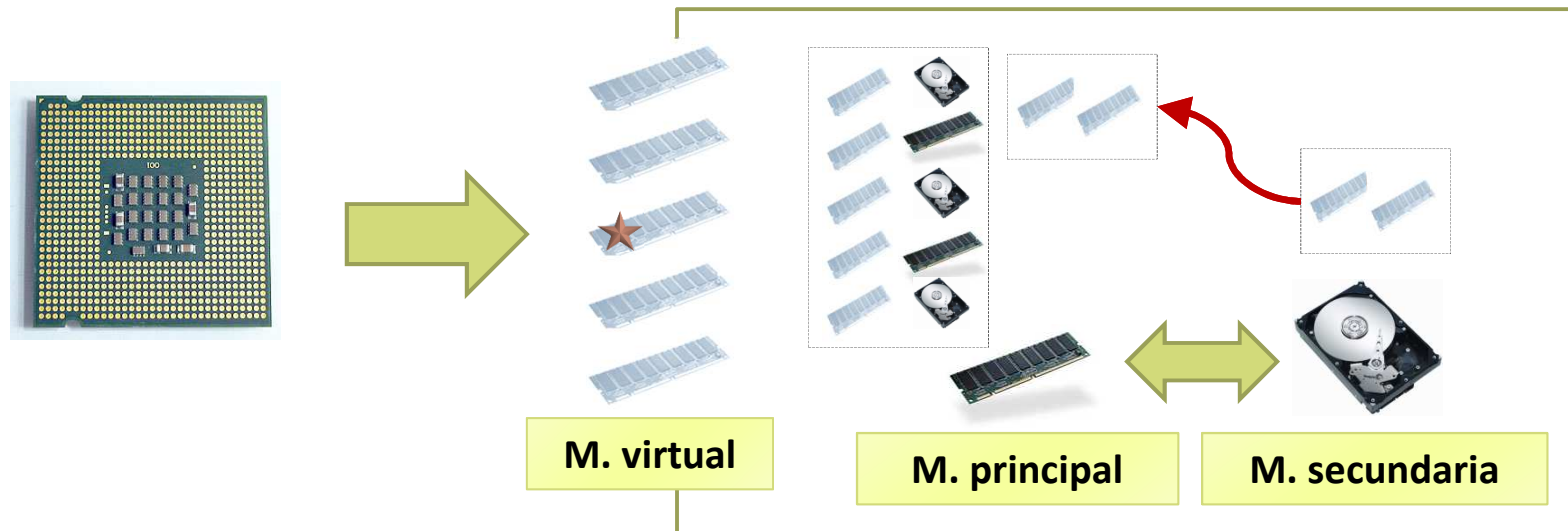


Introducción a memoria virtual



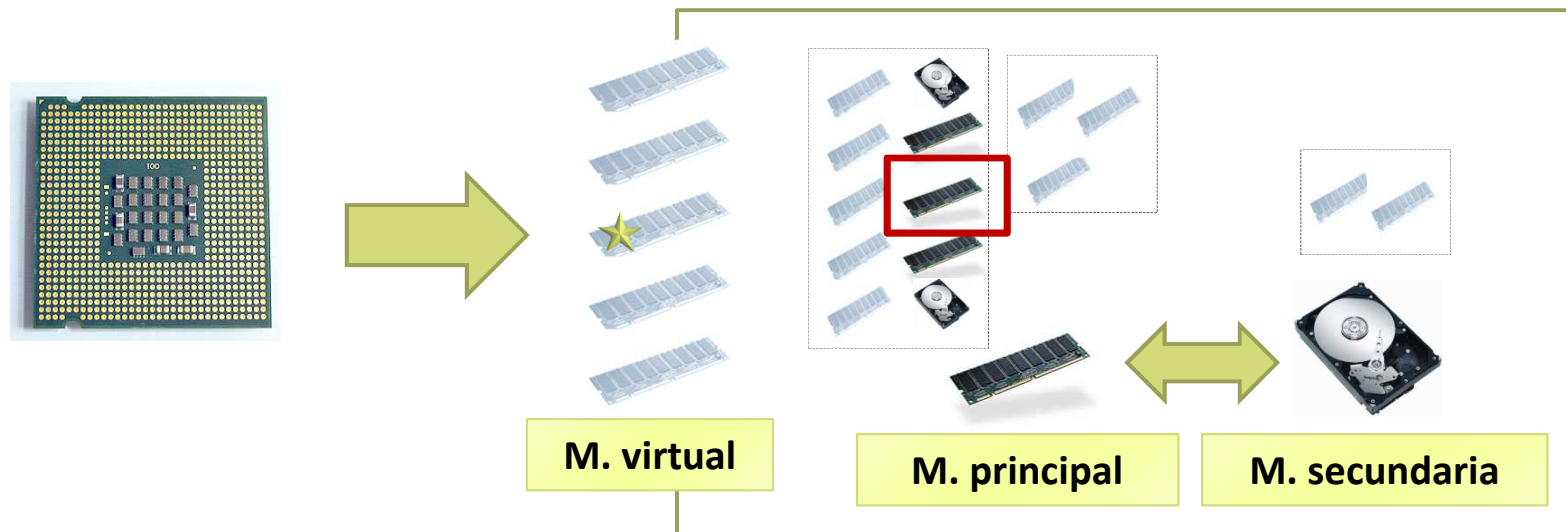
- ▶ El fallo de página es una excepción que provoca que el procesador ejecute la rutina de tratamiento asociada (kernel del s.o.).
- ▶ La rutina pide los bloques de disco asociados y bloquea el proceso.

Introducción a memoria virtual



- ▶ En la interrupción hardware de disco se transfiere el 'bloque' solicitado a memoria principal y programa una interrupción software.

Introducción a memoria virtual

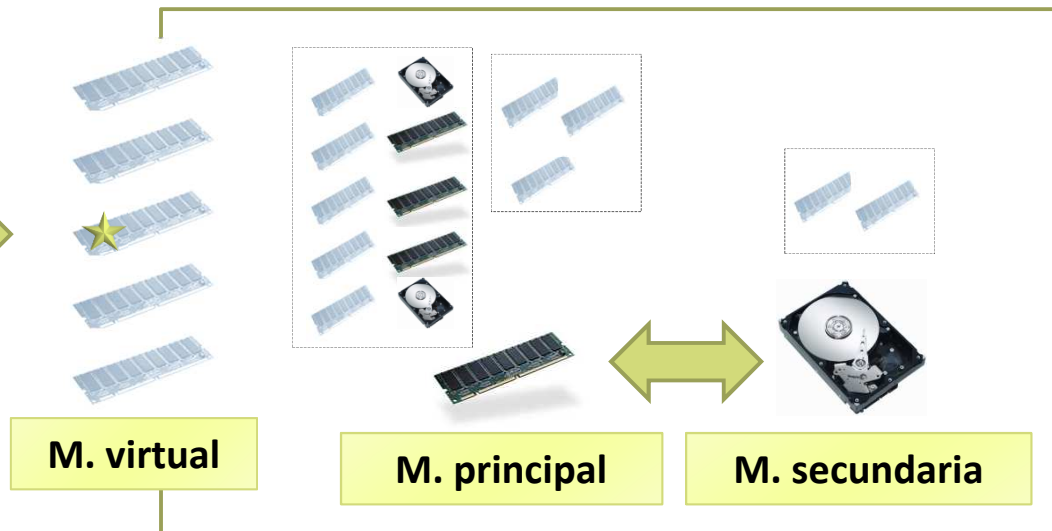
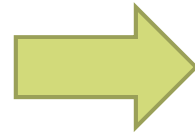
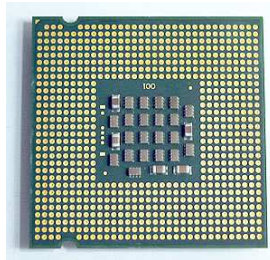


- ▶ En la interrupción software se actualiza la tabla de 'bloques' y se pone el proceso listo para ejecutar.

Introducción a memoria virtual

Acierto

...
lw \$t0 vector
...



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

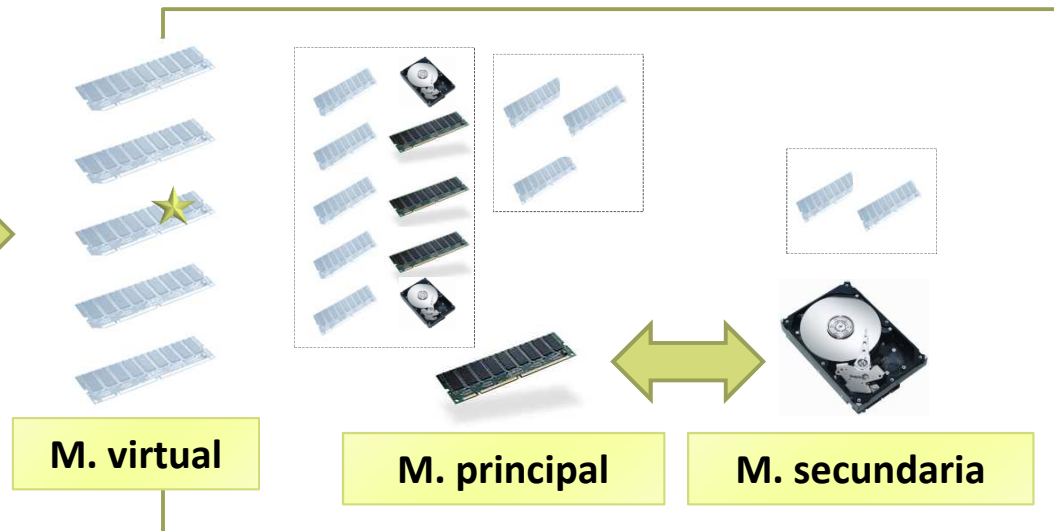
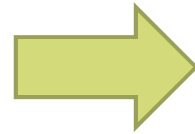
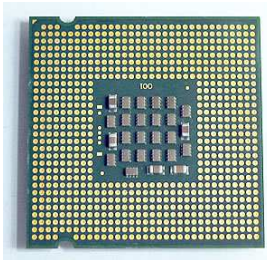
Introducción a memoria virtual

Acierto

...

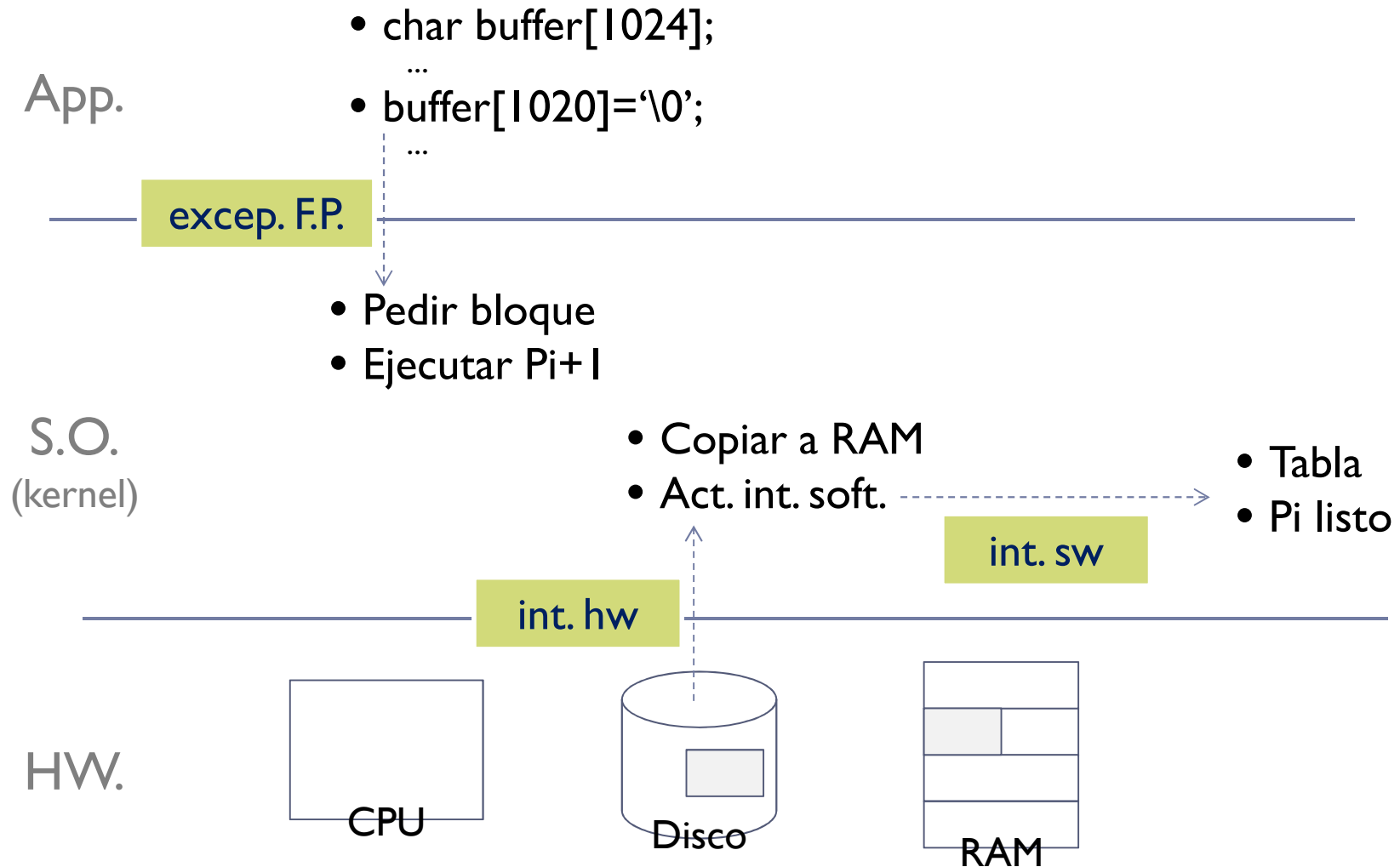
lw \$t0 vector+4

...



- ▶ La siguiente instrucción del mismo bloque no provoca fallo.

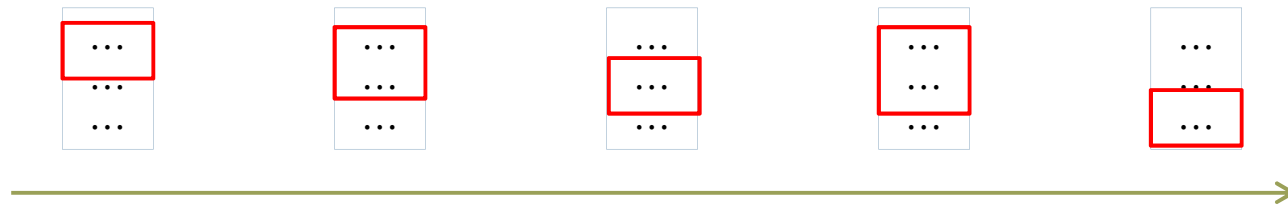
Esquema de la excepción de fallo de página



Introducción a memoria virtual

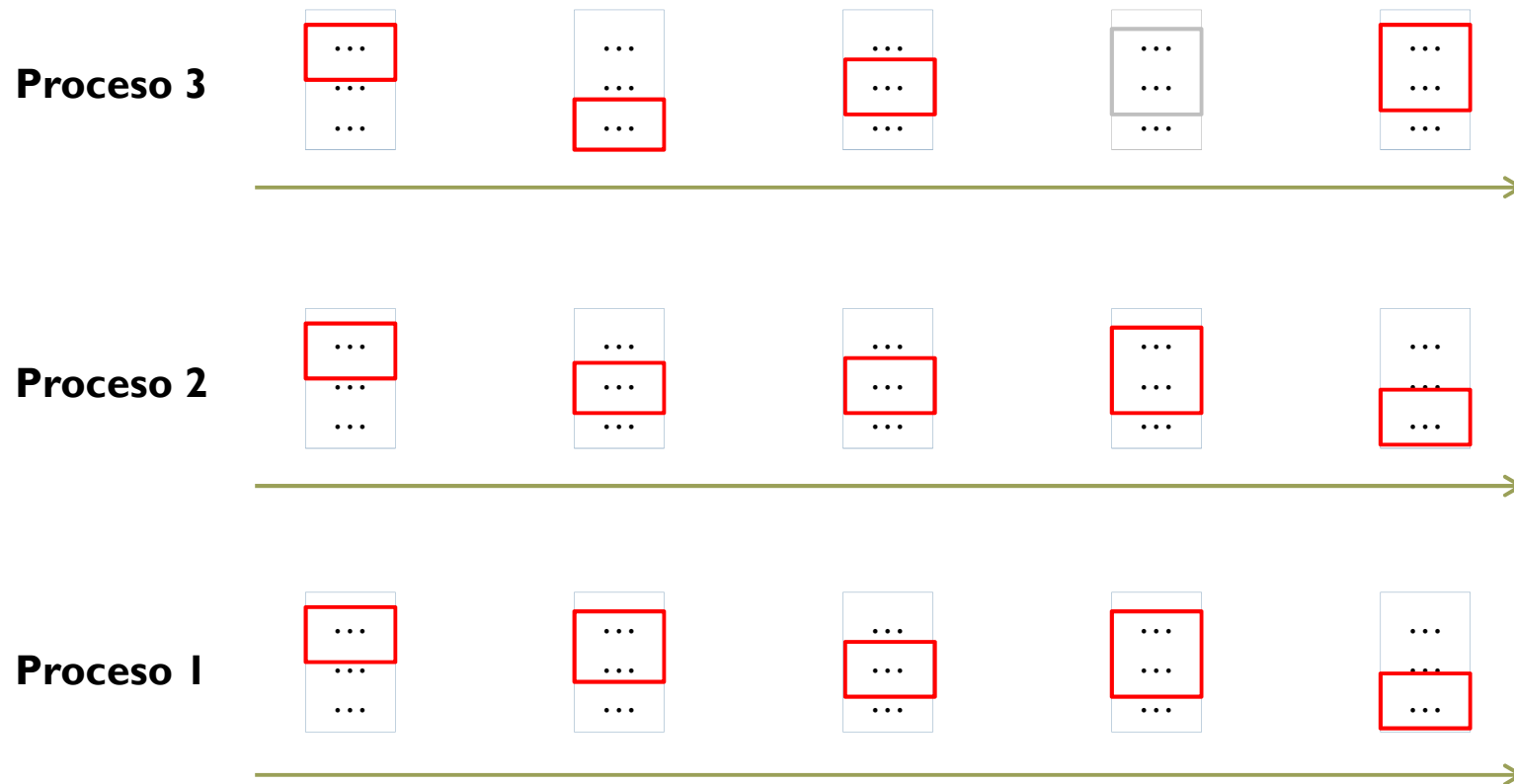
Espacio de trabajo y multiprogramación

Proceso I



Introducción a memoria virtual

Espacio de trabajo y multiprogramación



Introducción a memoria virtual

Espacio de trabajo y multiprogramación

Proceso 3



Proceso 2



Proceso 1



Sistemas multiprogramados ejemplo de cálculo de utilización



- ▶ p = % tiempo un proceso está bloqueado
- ▶ p^n = probabilidad de que n procesos independientes estén todos bloqueados
- ▶ $1 - p^n$ = probabilidad de que la CPU **no** esté ociosa (no idle)
- ▶ $n = 5$ procesos independientes
- ▶ $p = 0,8$ del tiempo bloqueado (20% en CPU)
- ▶ **utilización = $5 \cdot 20\% \rightarrow 100\%$**
- ▶ **utilización = $1 - 0,8^5 \rightarrow 67\%$**
- ▶ $1 - 0,8^{10} \rightarrow 89\%$

▶ 20

http://www.cs.rutgers.edu/~pxk/416/notes/content/09-memory_management-slides-6.pdf

ARCOS @ UC3M
Alejandro Calderón Mateos

Introducción a memoria virtual

Espacio de trabajo y multiprogramación

Proceso 3



Proceso 2



Proceso 1



Principales parámetros (4/4)

► Hay que equilibrar:

► El número de procesos en memoria (grado de multiprogramación)

► Swaping:

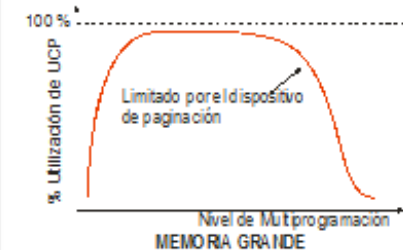
- Trasiego de información entre M. Principal y M. secundaria.

► Hiperpaginación:

- Se produce cuando el número de fallos es muy elevado
- El sistema está más tiempo intercambiando fragmentos que ejecutando instrucciones de usuario.

► ...

Comportamiento típico de la paginación con varios programas.



► 86

ARCOS @ UC3M
Alejandro Calderón Mateos



Memoria virtual: Windows 7

Administrador de tareas de Windows

Archivo Opciones Ver Ayuda

Aplicaciones Procesos Servicios Rendimiento Funciones de red Usuarios

Nombre de imagen	Nombre ...	CPU	Espacio de trabajo (memoria)	Memoria (espacio de trabajo privado)	Errores d...	Bytes de ...	Bytes de escritu...	Otros
AcroRd32.exe *32	merlin	00	163.352 KB	132.384 KB	465.070	9.356.971	121.067	95
AppleMobileDeviceHelper.exe *32	merlin	00	11.084 KB	3.016 KB	4.968	52.812	6.867	5
AppleMobileDeviceService.exe *32	SYSTEM	00	7.940 KB	2.128 KB	2.283	906	5.834	3
audiodg.exe	SERVIC...	00	26.744 KB	19.908 KB	36.062	30.470	0	42
AvastSvc.exe *32	SYSTEM	00	22.644 KB	5.388 KB	1.575.904	610.18...	278.223.219	987.3
AvastUI.exe *32	merlin	00	11.212 KB	3.364 KB	12.577	3.875.267	116	
AVerHIDReceiver.exe *32	merlin	00	5.604 KB	1.376 KB	2.879	604	0	2
AVerRemote.exe *32	SYSTEM	00	9.240 KB	2.680 KB	4.127	30.323	0	4
AVerScheduleService.exe *32	SYSTEM	00	7.896 KB	2.228 KB	54.294	136	0	13
CLMLSvc.exe *32	merlin	00	5.376 KB	3.512 KB	19.291.250	3.031.317	14.084.100	123.9
CNYHKEY.exe *32	merlin	00	11.420 KB	3.052 KB	3.383	33.108	0	3
conhost.exe	merlin	00	4.256 KB	1.640 KB	1.070	20.210	0	
conhost.exe	merlin	00	4.236 KB	1.636 KB	1.065	20.210	0	
csrss.exe	SYSTEM	00	4.836 KB	1.860 KB	1.740	476.159	0	13
csrss.exe	SYSTEM	01	26.992 KB	9.480 KB	76.156	1.878.108	0	17
DeviceDisplayObjectProvider.exe	merlin	00	20.344 KB	9.996 KB	6.245	1.492.596	1.497.407	2.21
distnoted.exe *32	merlin	00	5.780 KB	1.504 KB	1.485	24.974	0	1
dllhost.exe	SYSTEM	00	7.900 KB	2.696 KB	2.203	26.252	0	2
DVDAgent.exe *32	merlin	00	880 KB	564 KB	8.821	50.092	116	12
dwm.exe	merlin	00	45.464 KB	19.960 KB	2.136.433	63	0	10
Dxpserver.exe	merlin	00	20.168 KB	7.084 KB	15.568	155.546	26.737	15
E_S40RPB.EXE	SYSTEM	00	4.152 KB	1.744 KB	1.061	0	0	1
E_S40STB.EXE	SYSTEM	00	4.644 KB	1.840 KB	1.188	1	43	1
EEventManager.exe *32	merlin	00	6.860 KB	2.040 KB	2.392	107.023	492	4.42
explorer.exe	merlin	01	106.264 KB	60.336 KB	974.316	1.176.17...	1.442.888	60.41

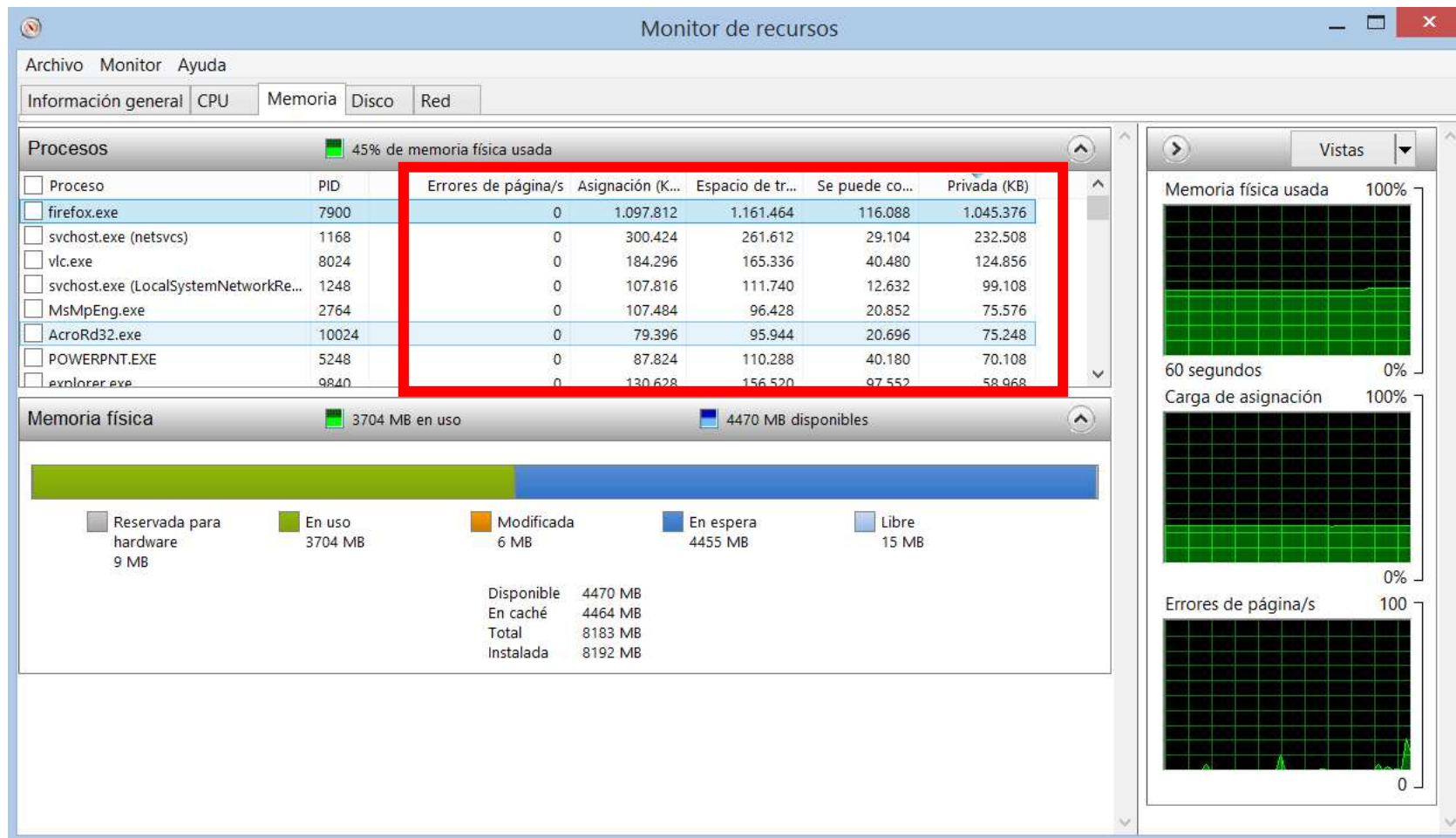
☒ Mostrar procesos de todos los usuarios

Finalizar proceso

Procesos: 118 Uso de CPU: 27% Memoria física: 45%



Memoria virtual: Windows 8.x





Memoria virtual: Linux

KDE Task Manager

File Refresh Rate Process Help

Processes List Performance Meter

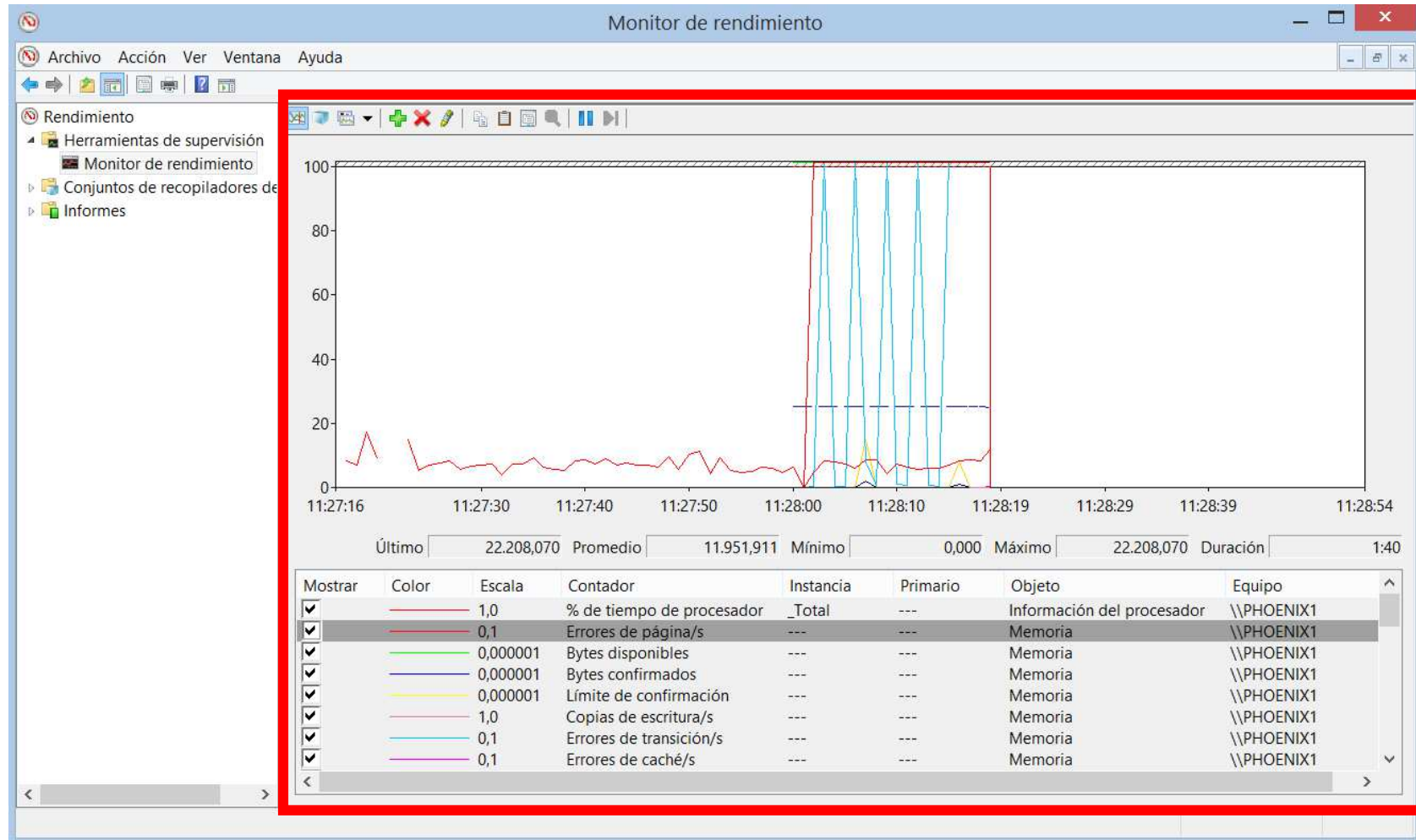
Running Processes

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

Show Tree Own processes Refresh Now Kill task

54 Processes Memory: 59900 kB used, 3608 kB free Swap: 1540 kB used, 66464 kB free

Windows: perfmon





Linux: ps, top, ...

```
arcos:~$ ps -o min_flt,maj_flt 1
MINFL MAJFL
18333  25
```

Minor fault: petición de reserva de página

Major fault: se precisa acceso a disco

Linux: ps, top, ...

```
arcos:~$ vmstat 1 5
procs  -----memory-----  ---swap--  -----io-----  -system--  -----cpu-----
 r  b    swpd   free   buff   cache    si   so    bi    bo    in   cs  us  sy  id  wa
 1  0      140 3092132 1575132 2298820    0    0    12    19    20   32   1   2  97   0
 0  0      140 3092124 1575132 2298820    0    0     0     0   128  250   0   0 100   0
 0  0      140 3092124 1575132 2298820    0    0     0    16   143  281   0   0 100   1
 0  0      140 3092124 1575132 2298820    0    0     0     0   137  247   0   0 100   0
 0  0      140 3092124 1575132 2298820    0    0     0     0   138  270   0   0 100   0
```

Procs

r: The number of processes waiting for run time.
b: The number of processes in uninterruptible sleep.

Memory

swpd: the amount of virtual memory used.
free: the amount of idle memory.
buff: the amount of memory used as buffers.
cache: the amount of memory used as cache.
inact: the amount of inactive memory. (-a option)
active: the amount of active memory. (-a option)

Swap

si: Amount of memory swapped in from disk (/s).
so: Amount of memory swapped to disk (/s).

IO

bi: Blocks received from a block device (blocks/s).
bo: Blocks sent to a block device (blocks/s).

System

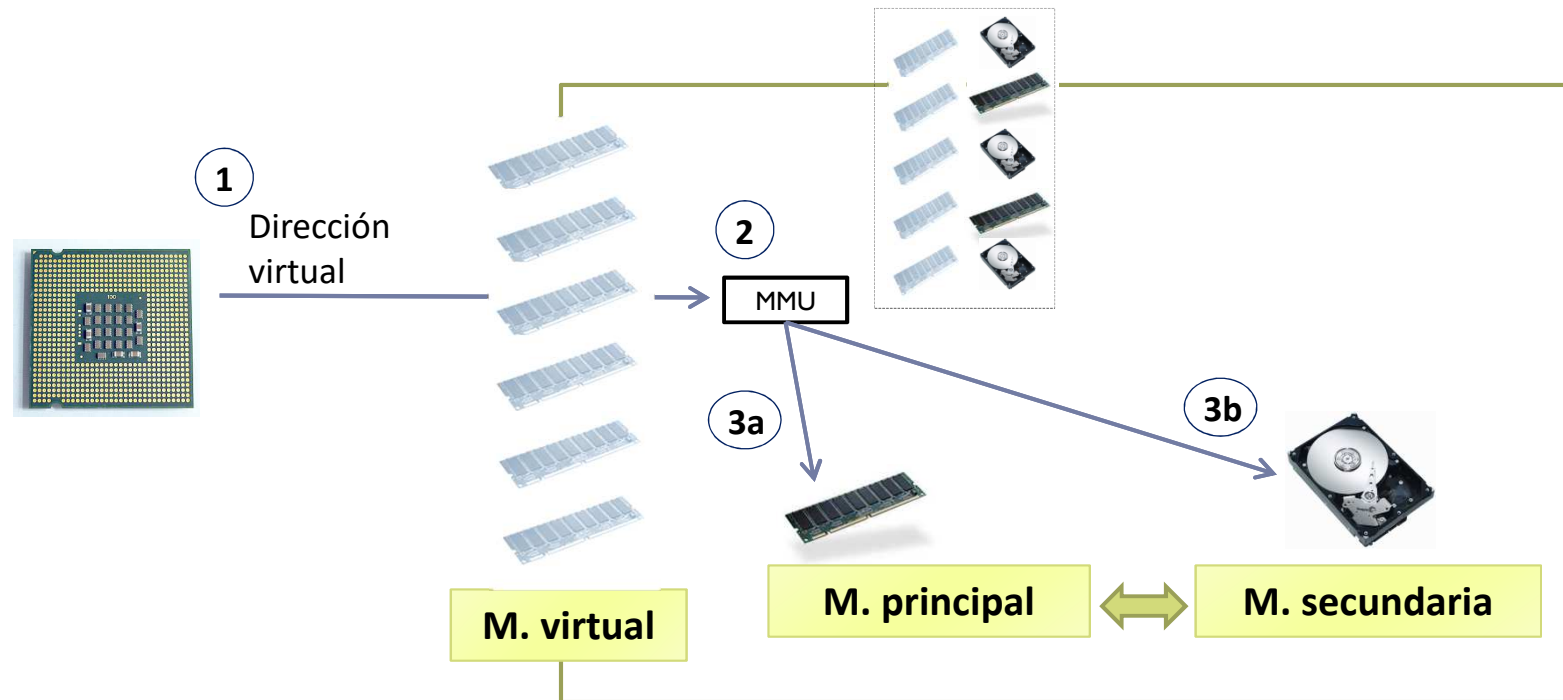
in: The number of interrupts per second, including the clock.
cs: The number of context switches per second.

CPU

These are percentages of total CPU time.

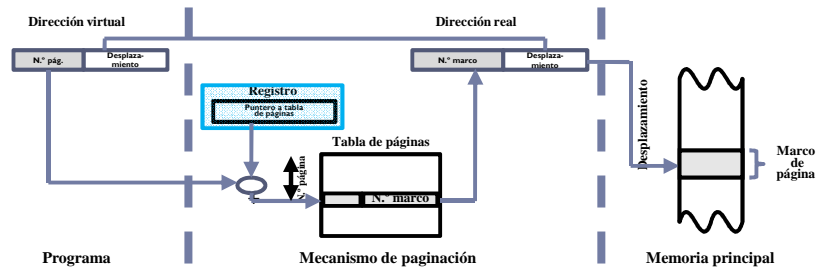
us: Time spent running non-kernel code. (user time, including nice time)
sy: Time spent running kernel code. (system time)
id: Time spent idle. Prior to Linux 2.5.41, this includes IO-wait time.
wa: Time spent waiting for IO. Prior to Linux 2.5.41, included in idle.
st: Time stolen from a virtual machine. Prior to Linux 2.6.11, unknown.

Sistemas con memoria virtual mecanismos de implementación

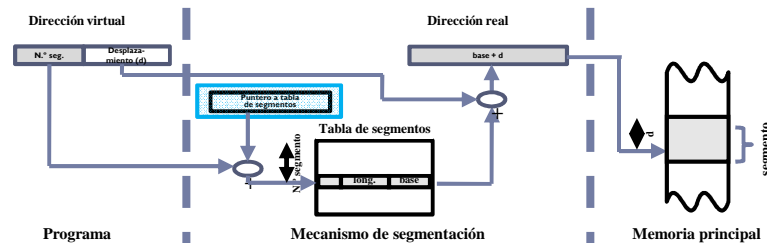


Memoria virtual

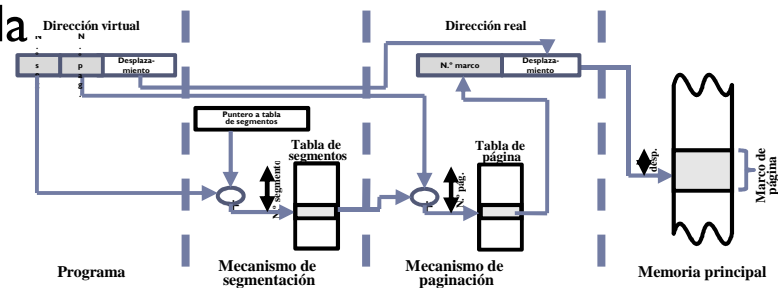
► Paginación



► Segmentación

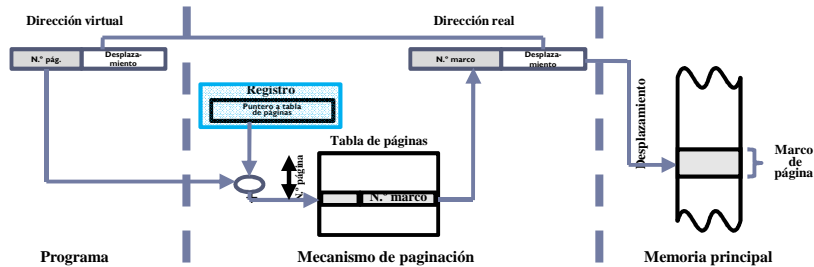


► Segmentación paginada

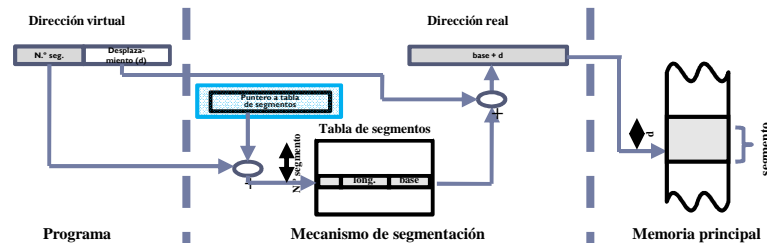


Memoria virtual

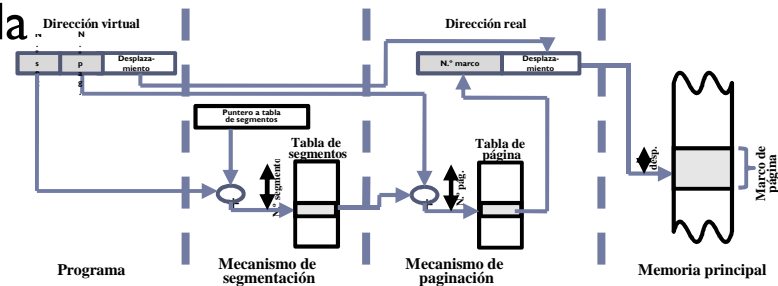
► Paginación



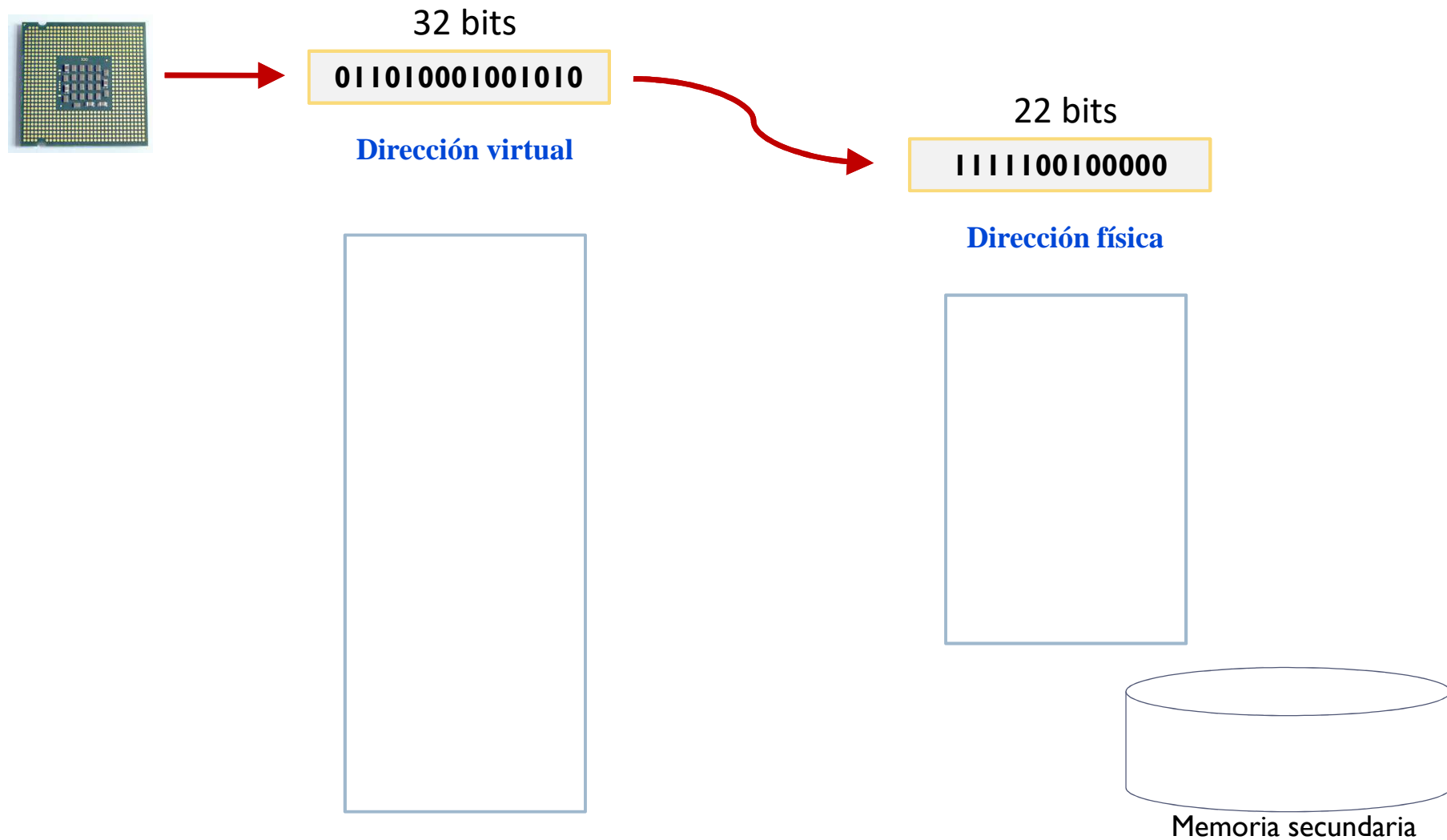
► Segmentación



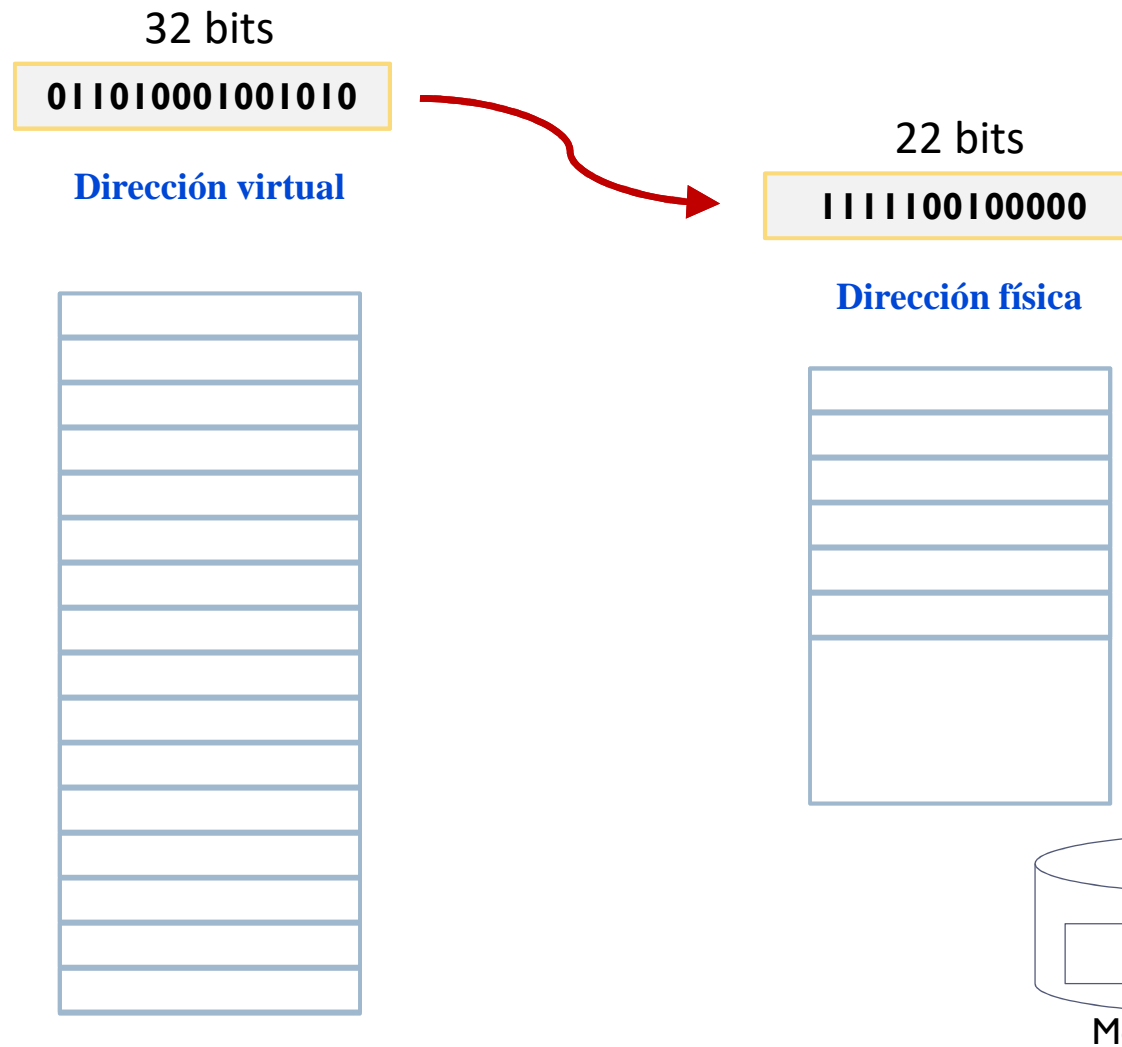
► Segmentación paginada



Ejemplo

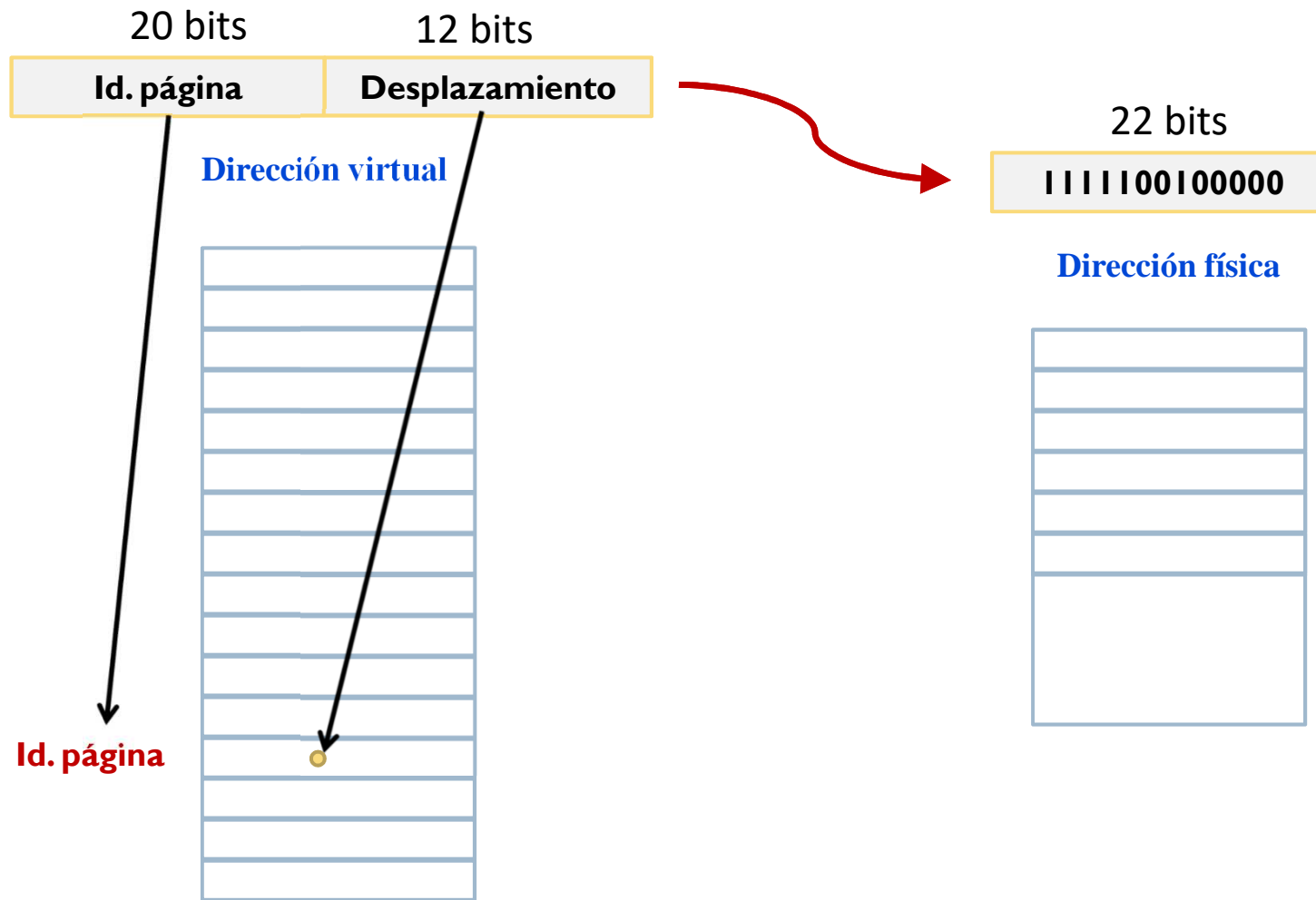


Ejemplo



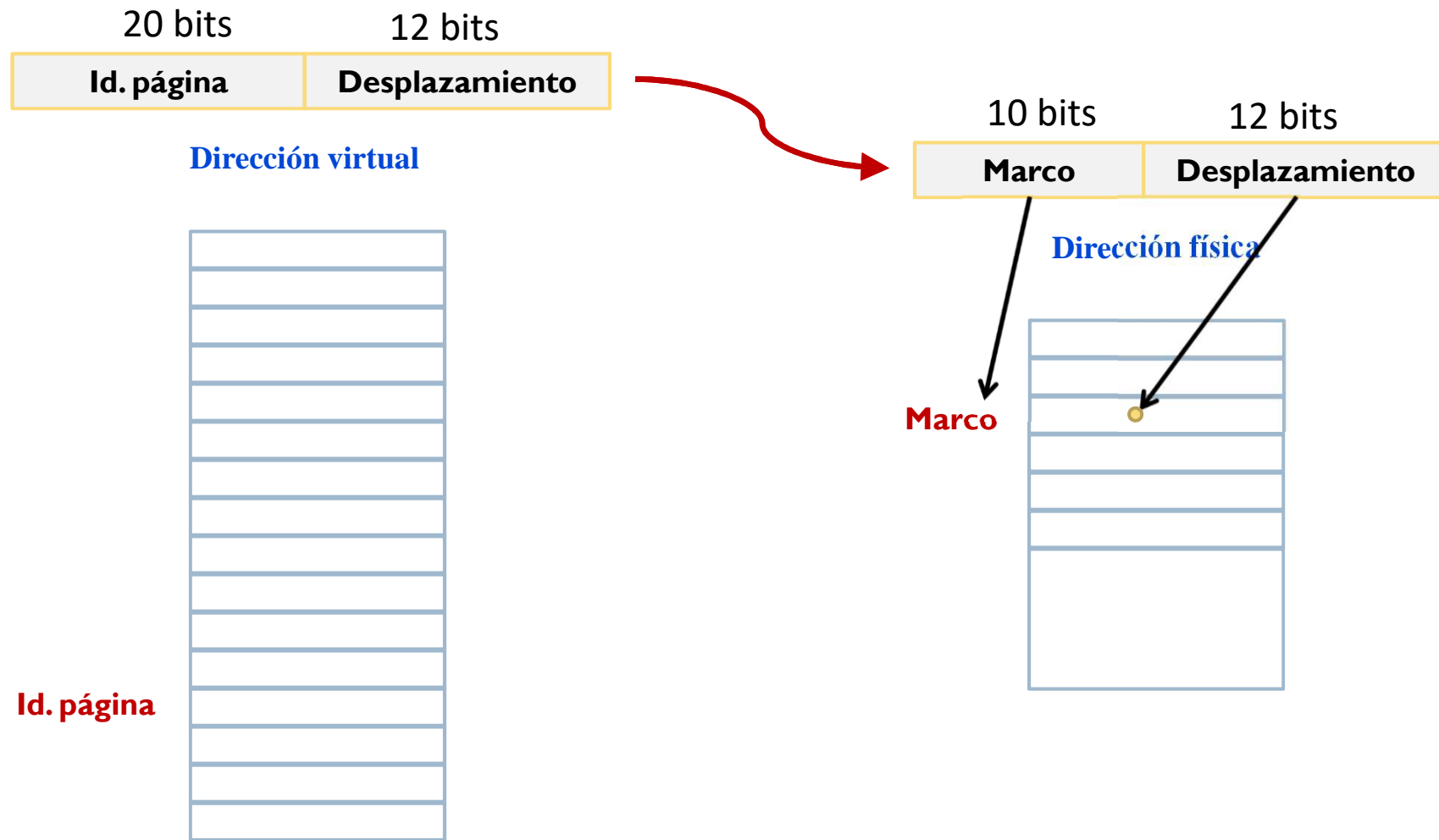
División en bloques del mismo tamaño -> páginas

Ejemplo



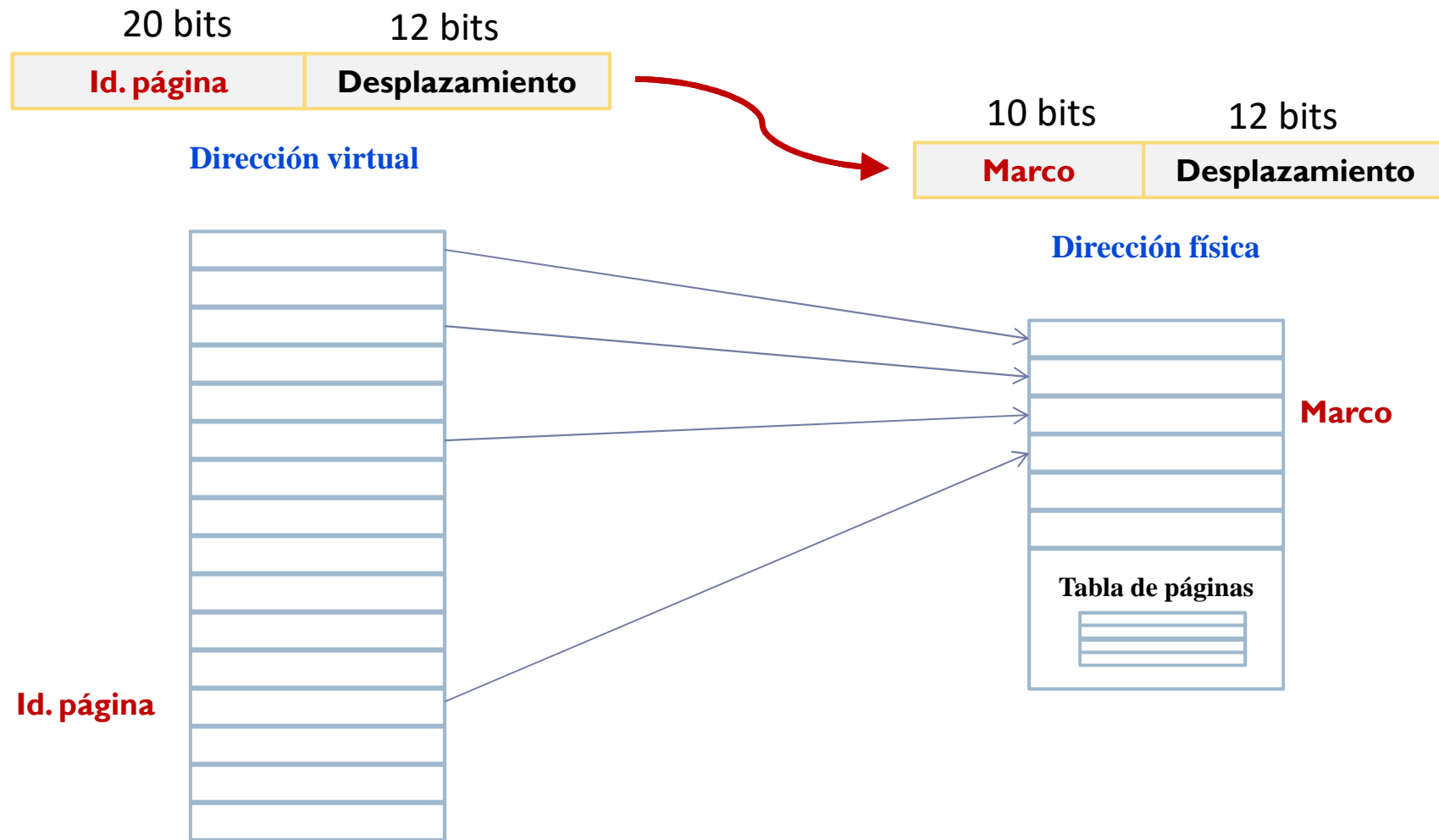
División en bloques del mismo tamaño -> páginas

Ejemplo



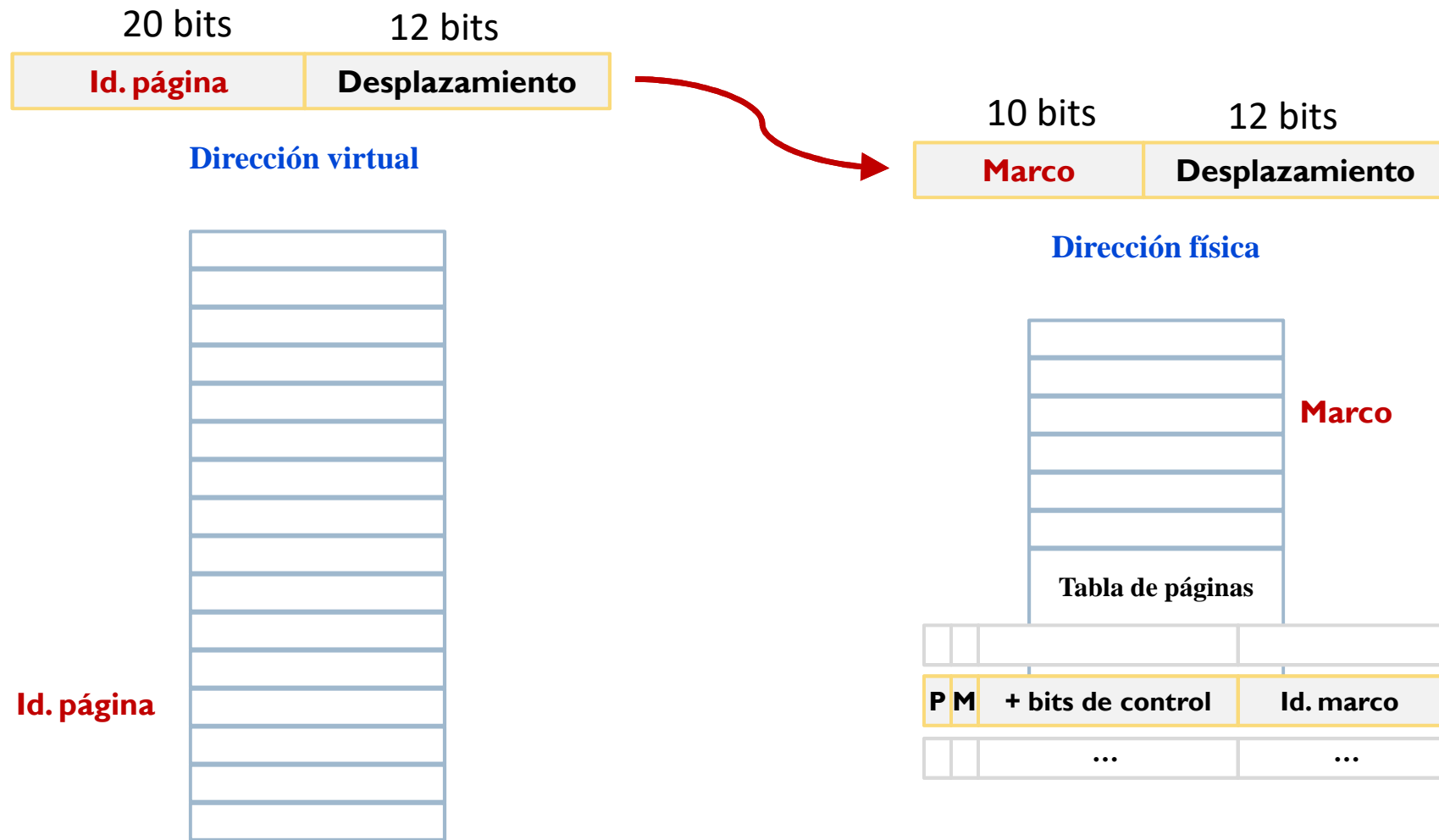
División en bloques del mismo tamaño -> páginas

Ejemplo



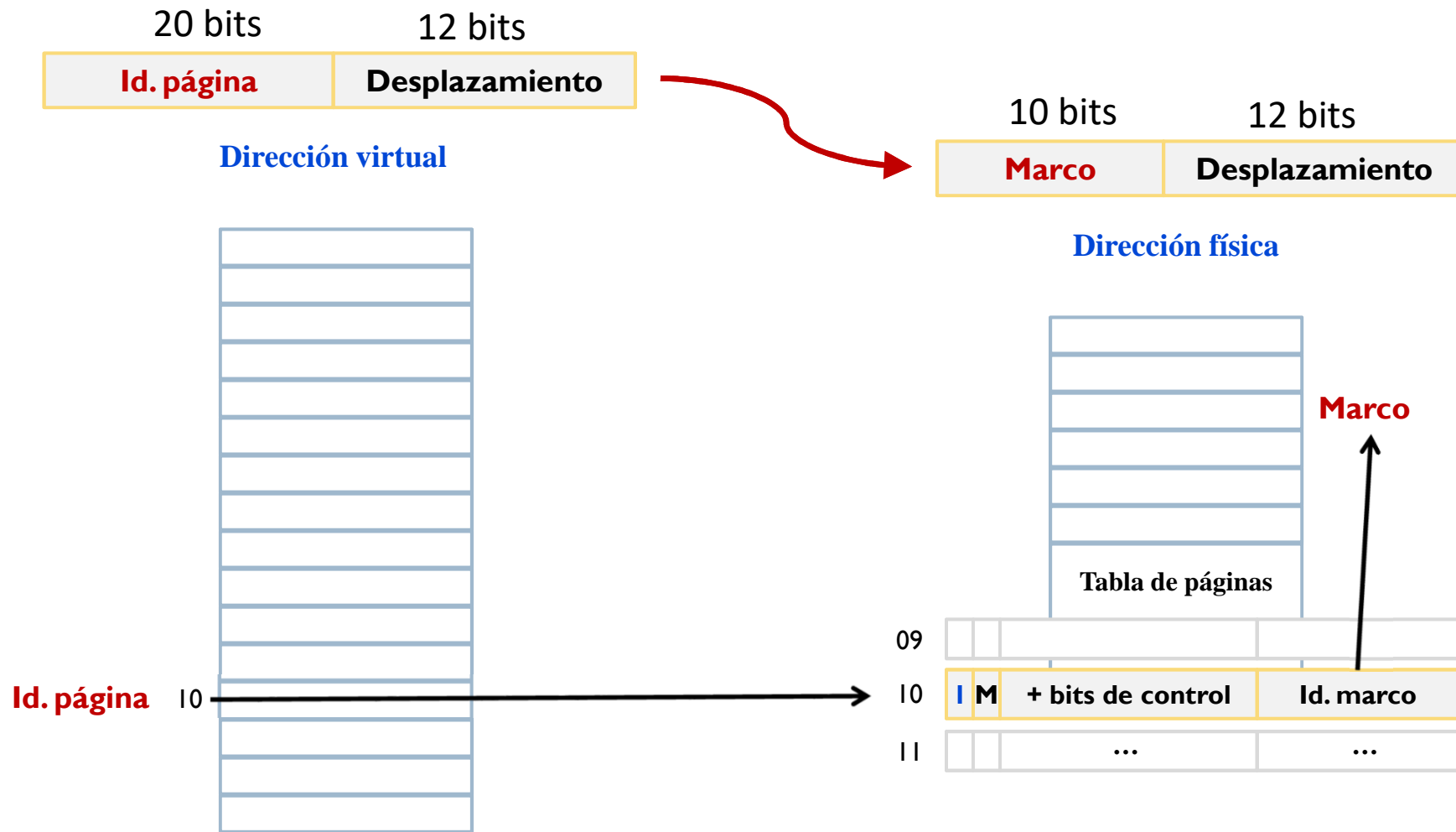
Correspondencia entre Id. página y marco -> T. páginas

Ejemplo

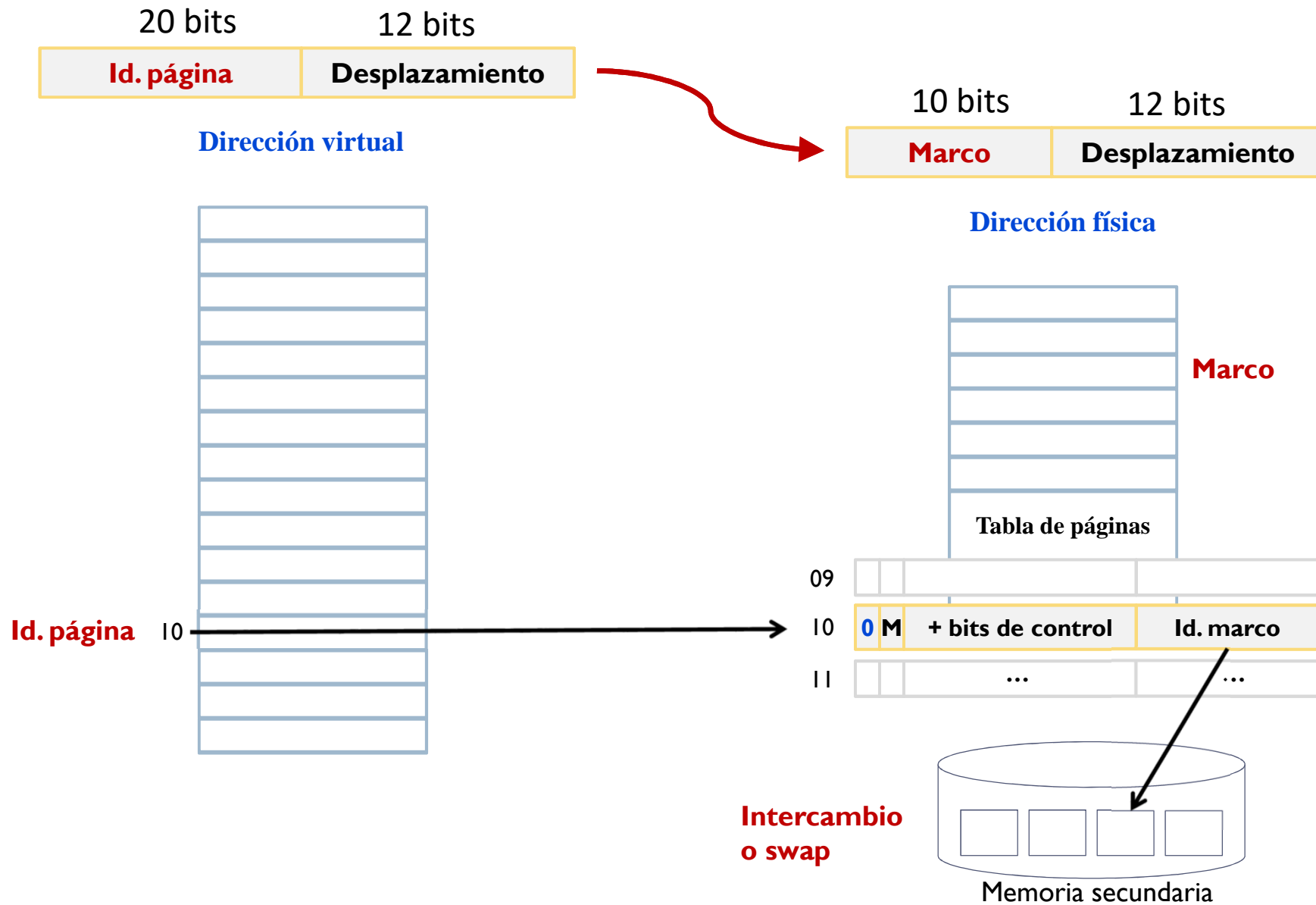


Correspondencia entre Id. página y marco -> T. páginas

Ejemplo

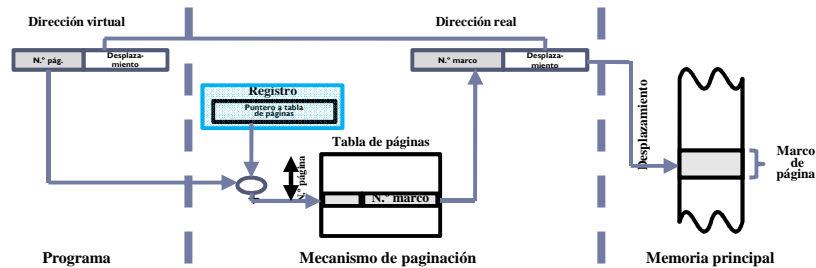


Ejemplo

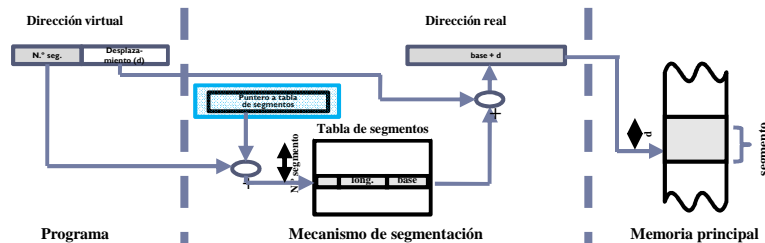


Memoria virtual

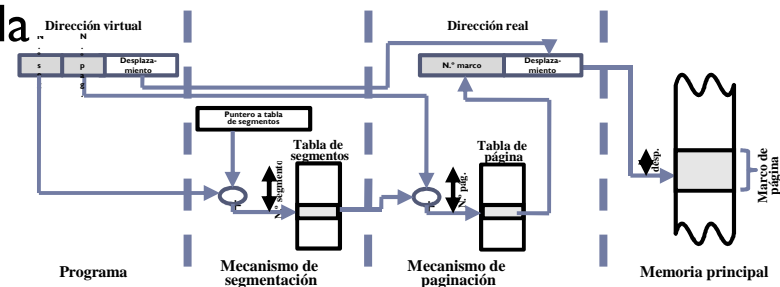
► Paginación



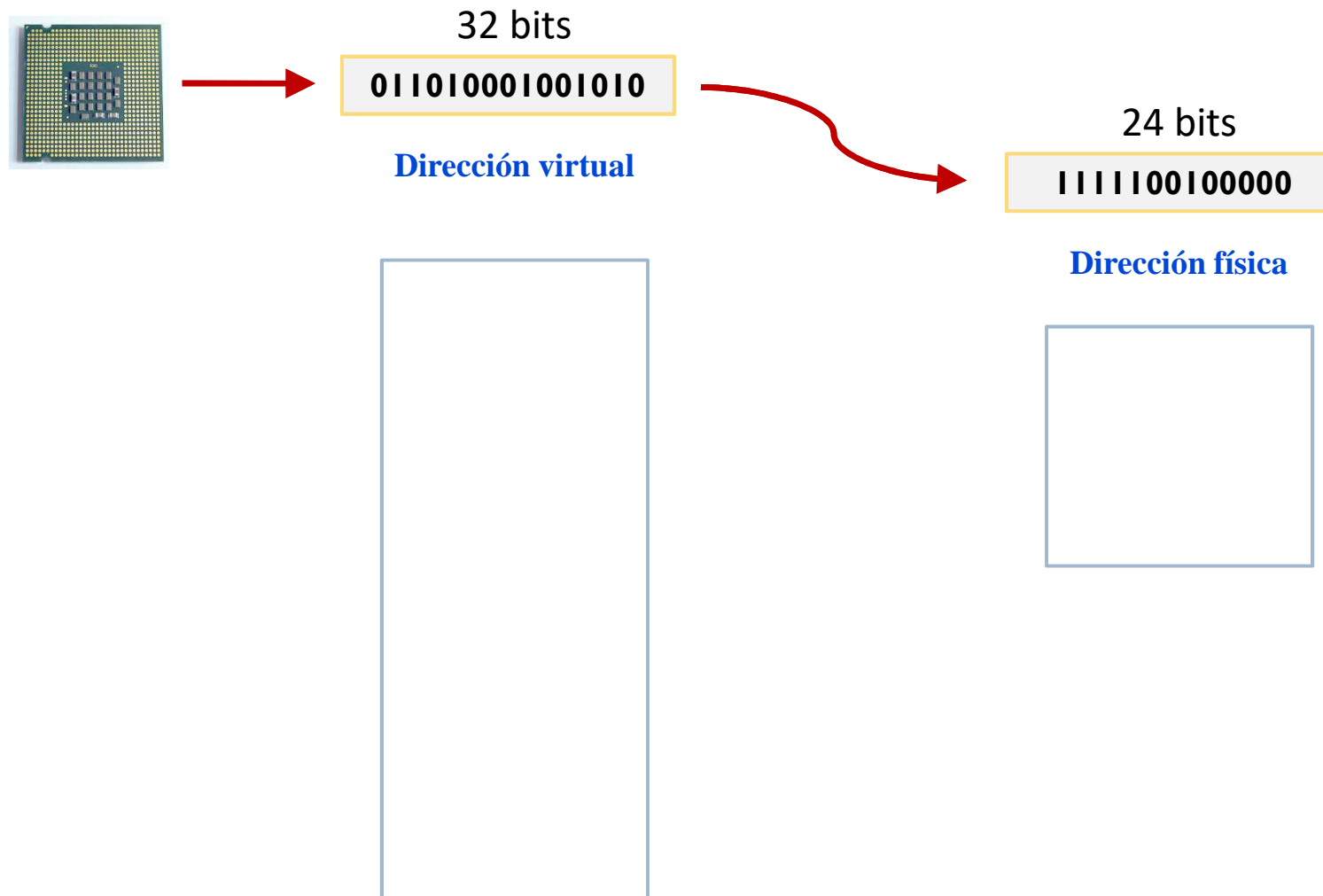
► Segmentación



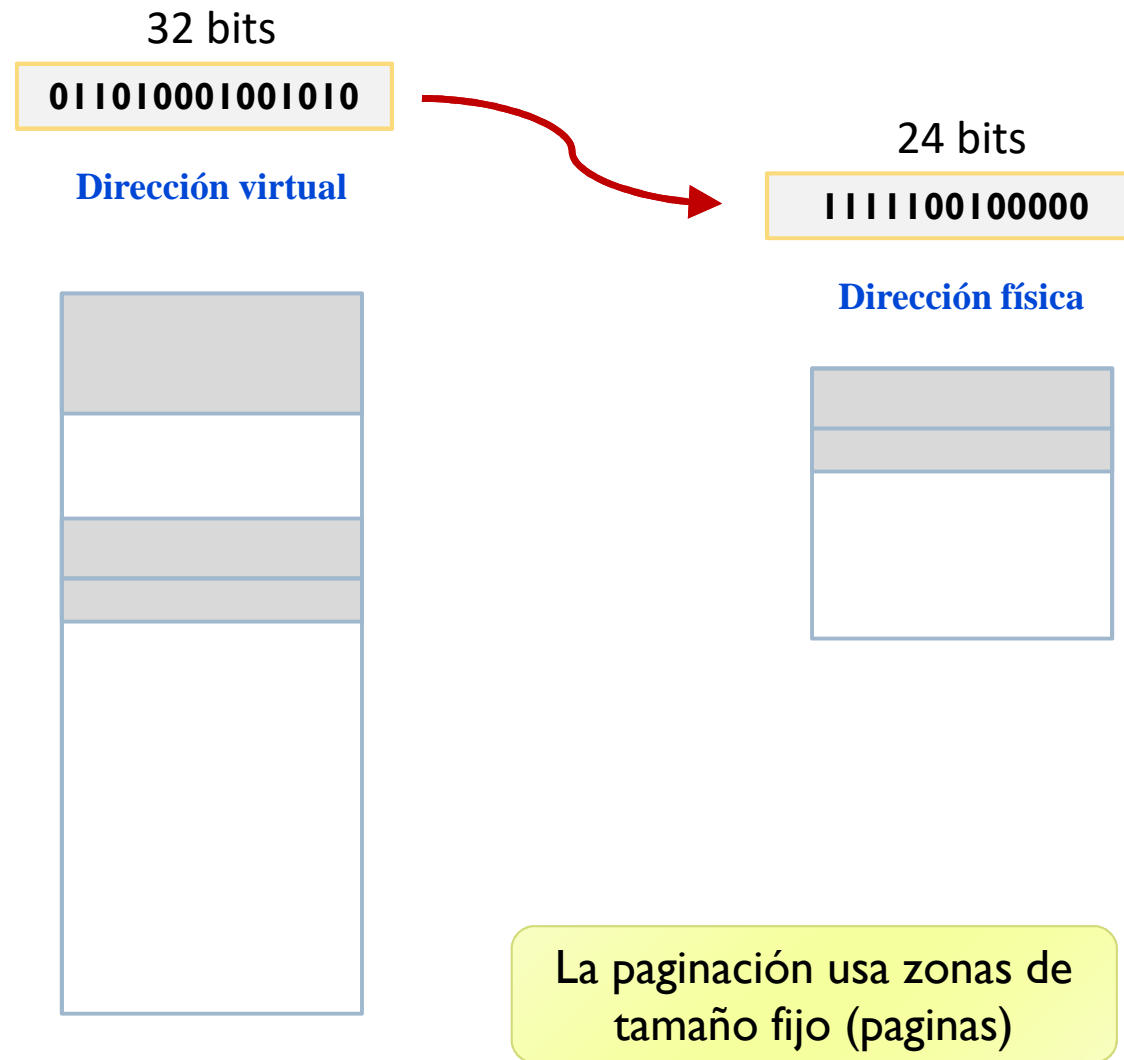
► Segmentación paginada



Ejemplo

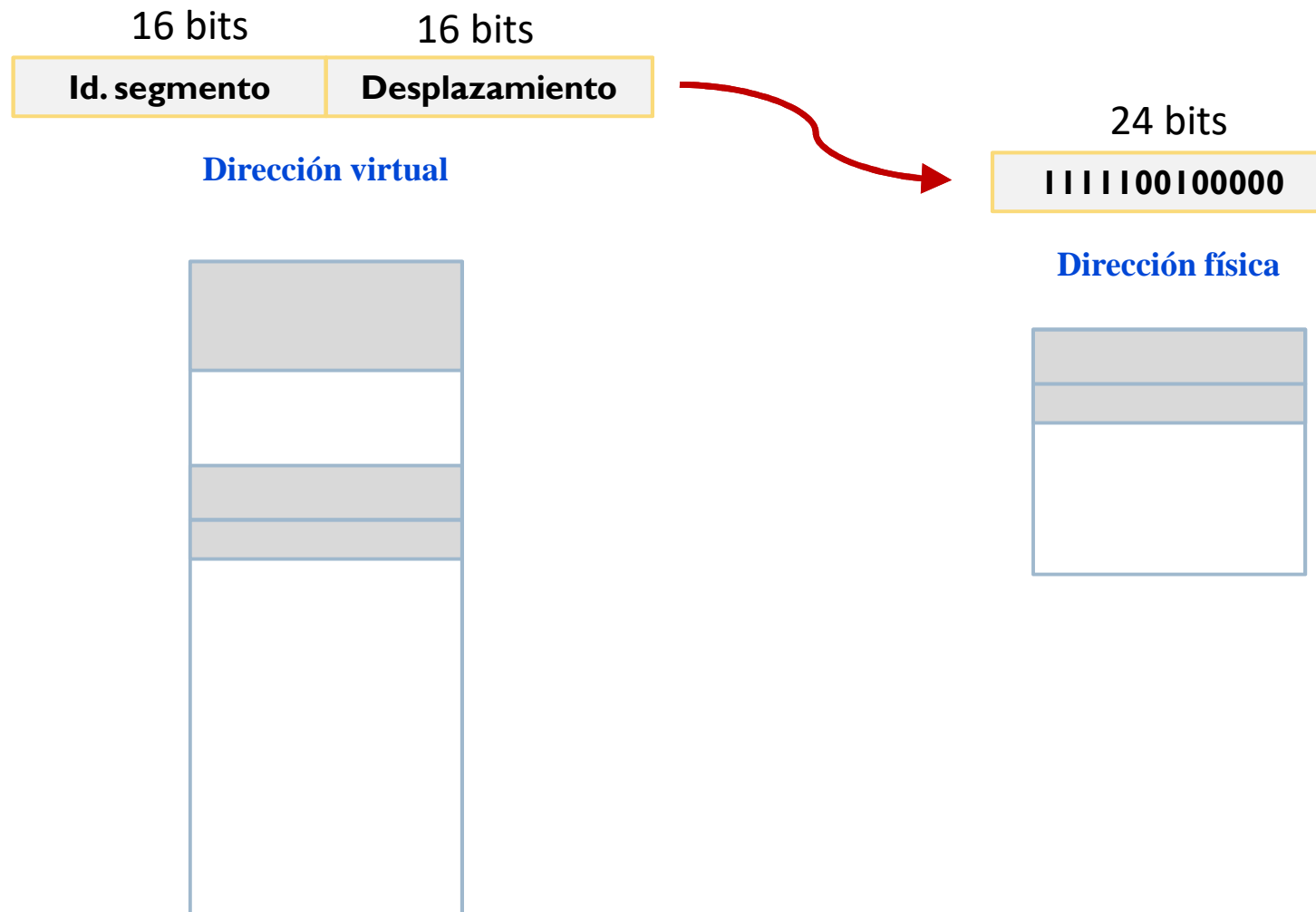


Ejemplo

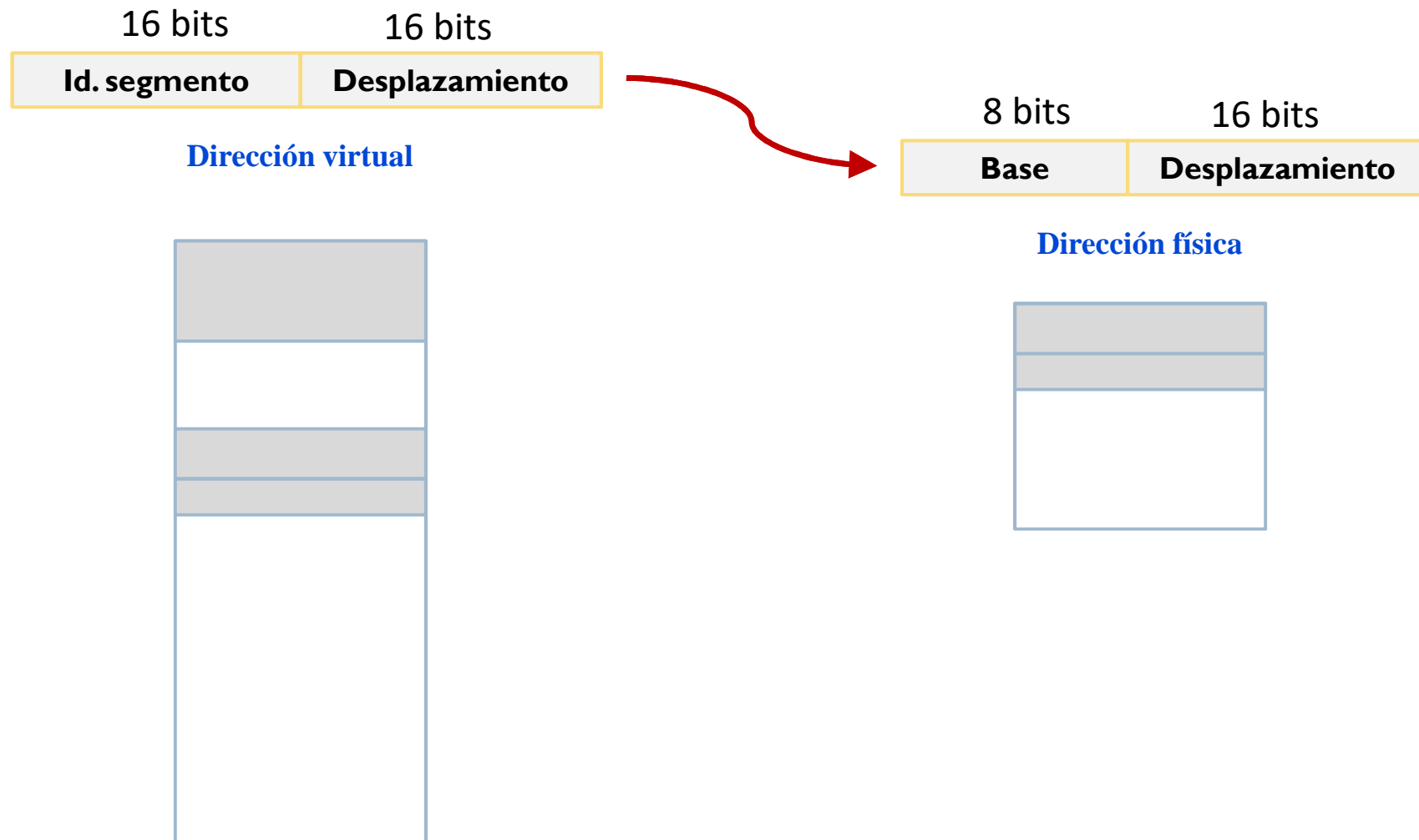


División en zonas de tamaño variable -> segmentos

Ejemplo

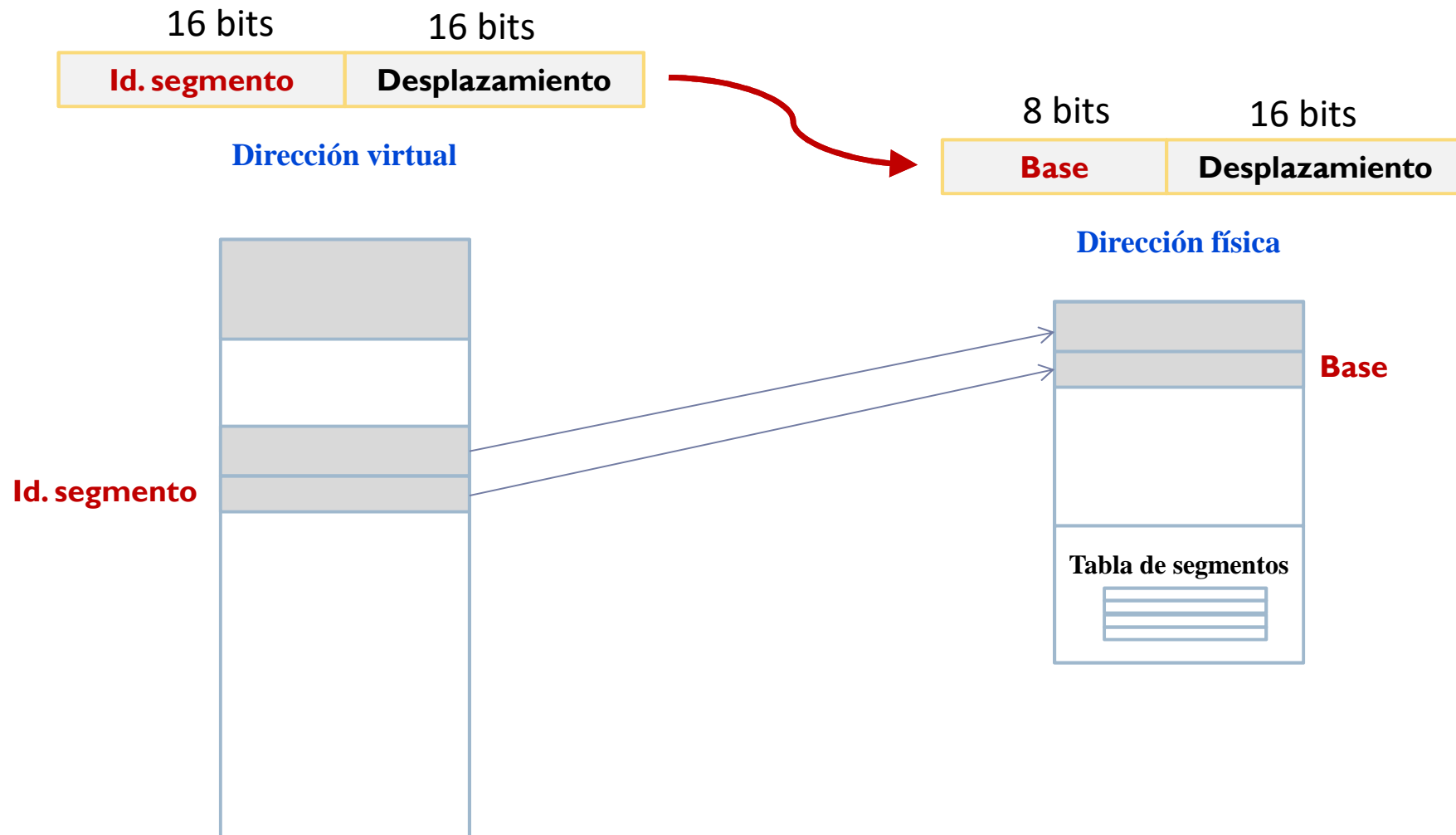


Ejemplo

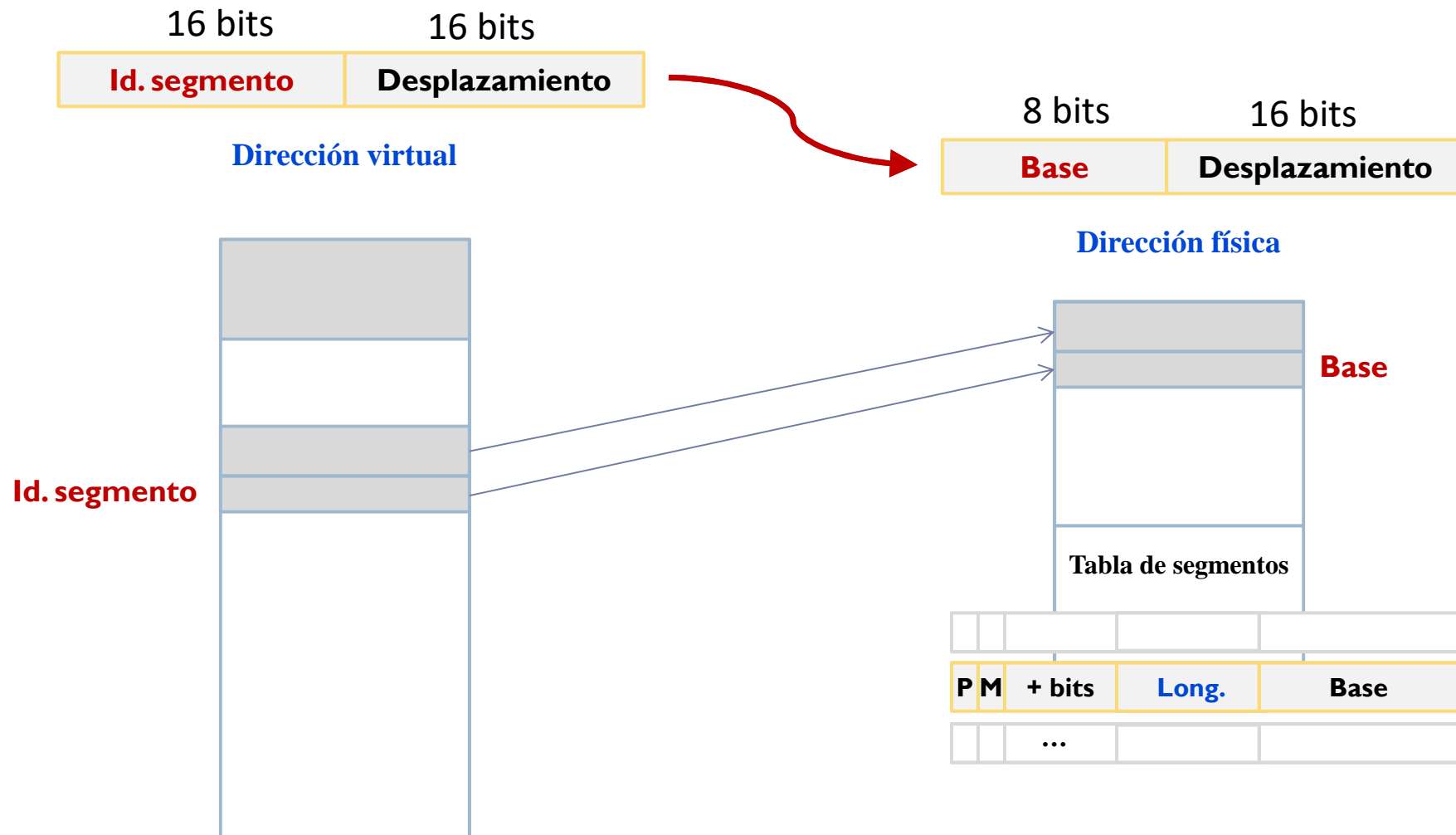


División en zonas de tamaño variable -> segmentos

Ejemplo

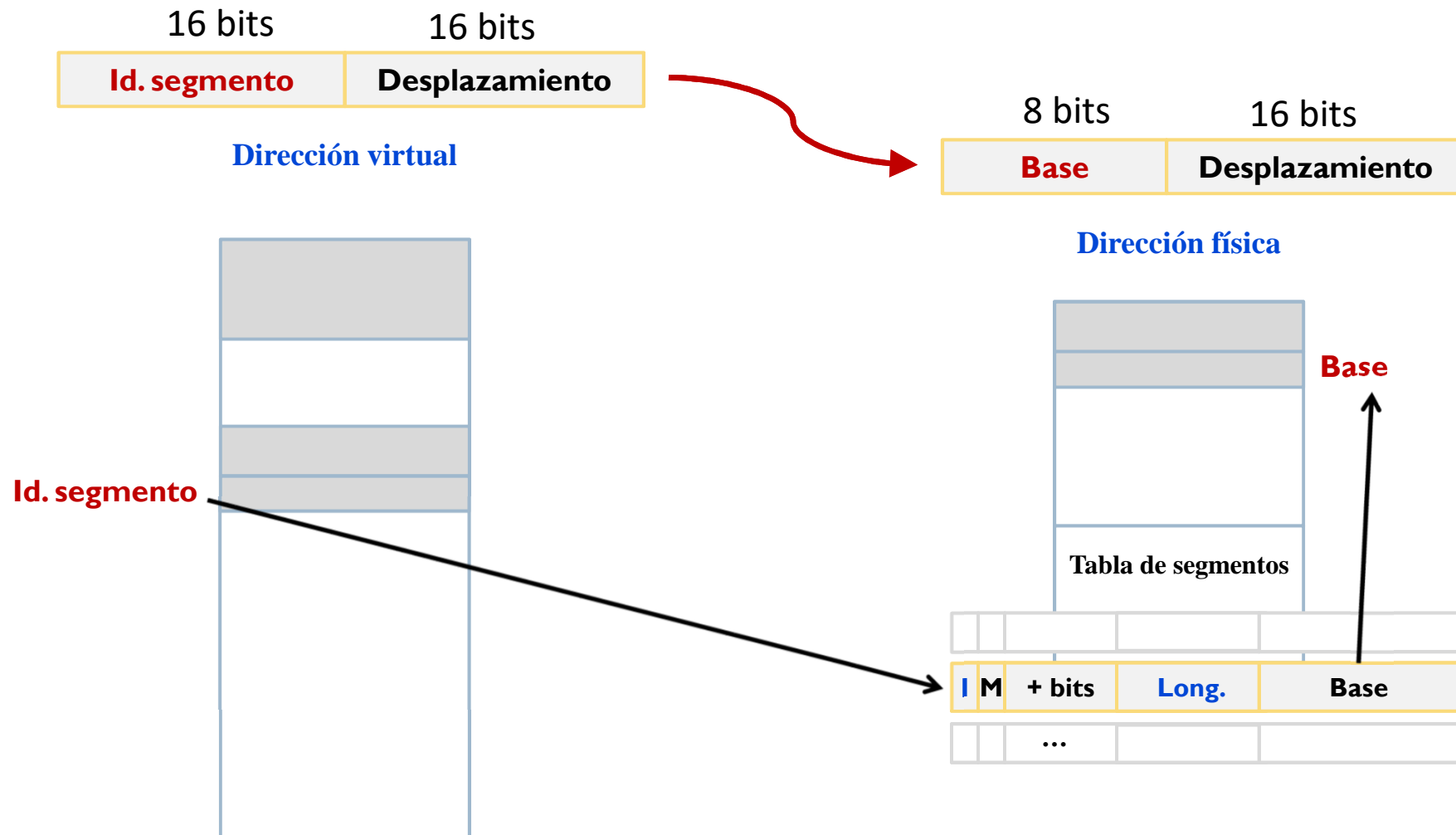


Ejemplo

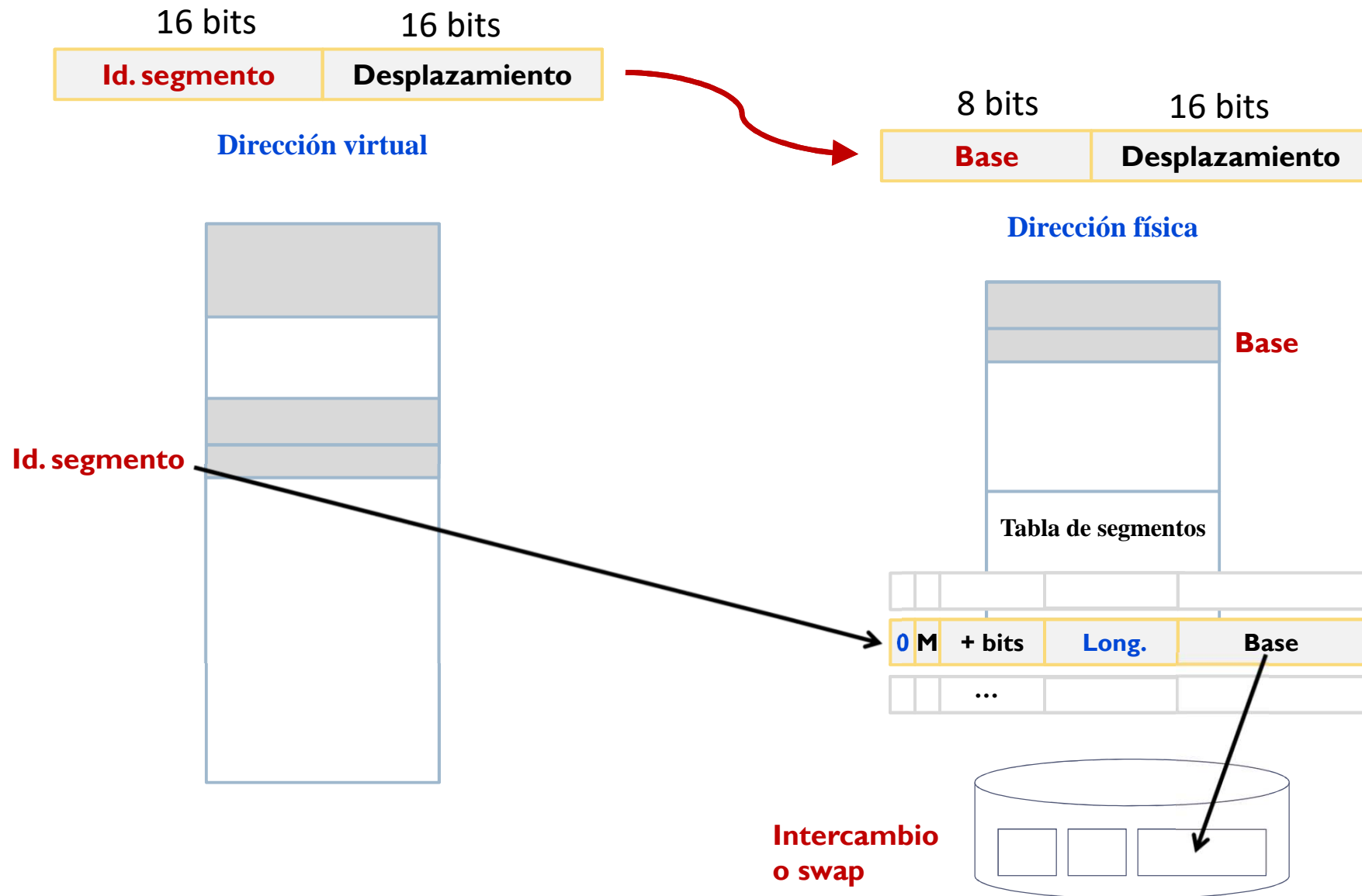


Correspondencia M.V. y M.F. -> tabla de segmentos

Ejemplo

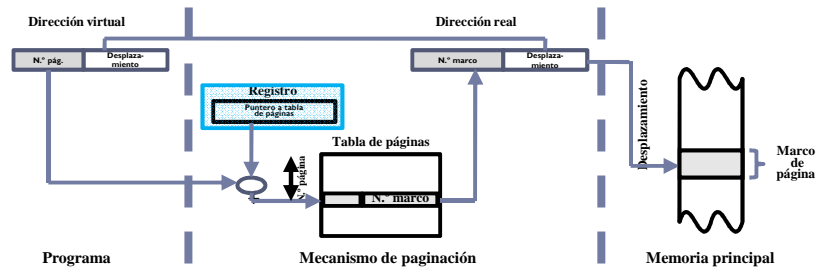


Ejemplo

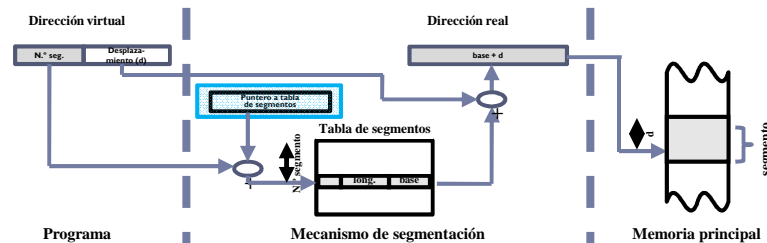


Memoria virtual

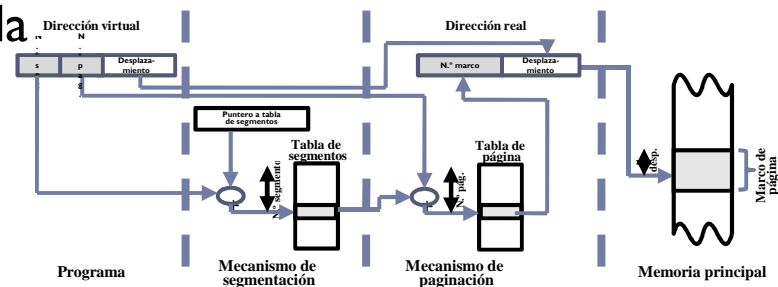
► Paginación



► Segmentación



► Segmentación paginada



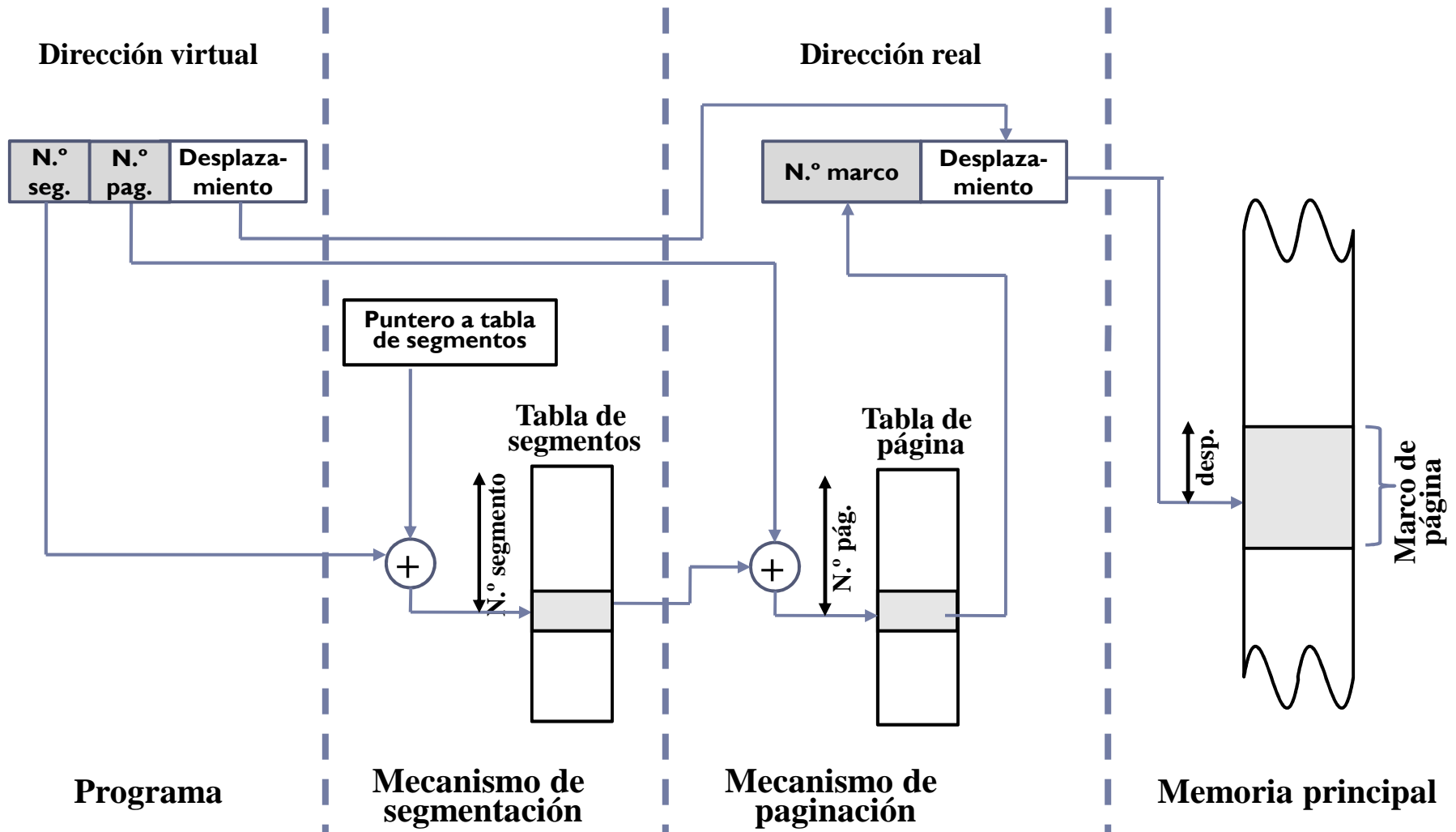
Memoria virtual:

segmentación y paginación

- ▶ Entrada en la tabla de segmentos ‘apunta’ a una tabla de páginas asociada al segmento
 - ▶ Los segmentos de tamaño variable se fragmentan en páginas de tamaño fijo



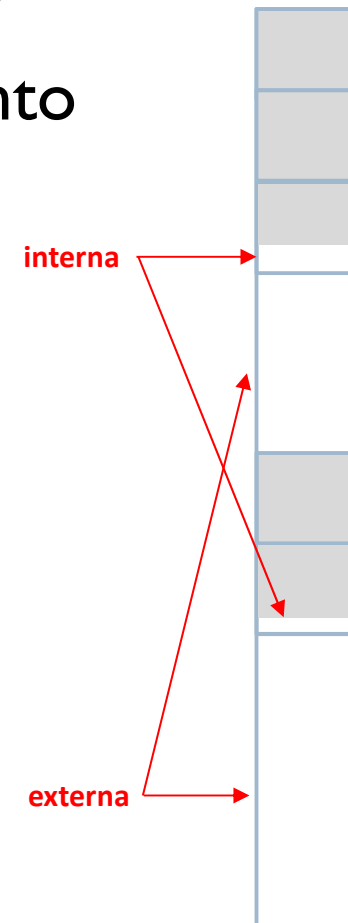
Traducción de direcciones segmentación paginada



Memoria virtual:

segmentación y paginación

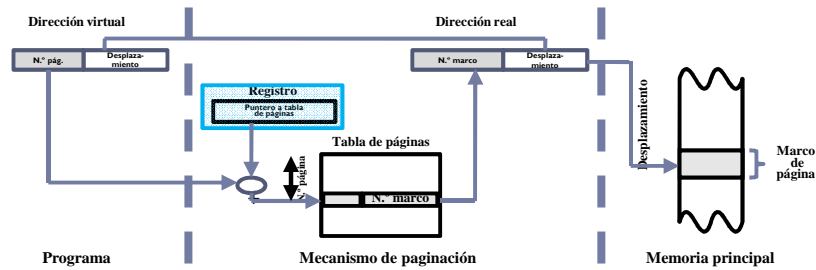
- ▶ Entrada en la tabla de segmentos ‘apunta’ a una tabla de páginas asociada al segmento
 - ▶ Los segmentos de tamaño variable se fragmentan en páginas de tamaño fijo
- ▶ Lo mejor de las dos soluciones:
 - ▶ Segmentación:
 - ▶ Facilita operaciones con regiones de memoria
 - ▶ Evita la fragmentación interna (tiene externa)
 - ▶ Paginación:
 - ▶ Optimiza el acceso a la memoria secundaria
 - ▶ Evita la fragmentación externa (tiene interna)



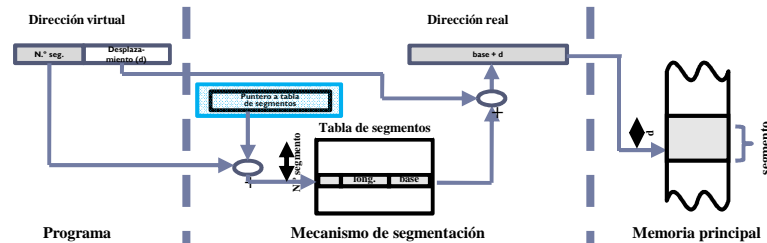
Memoria virtual

resumen

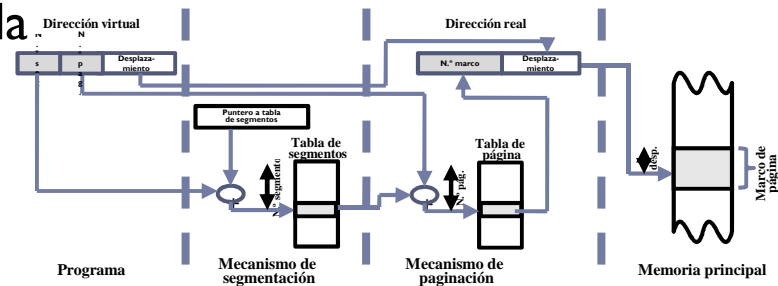
► Paginación



► Segmentación

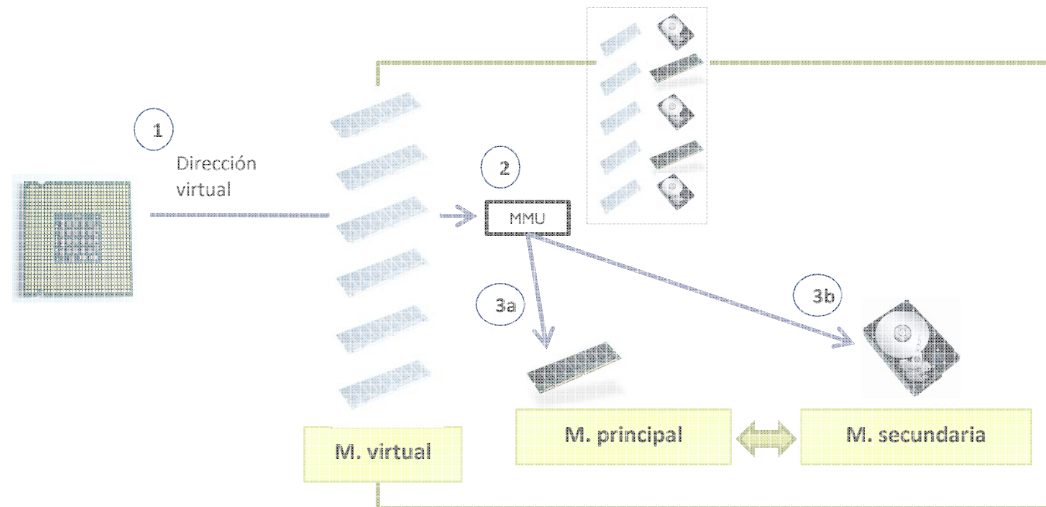


► Segmentación paginada



Gestión de memoria

aspectos avanzados

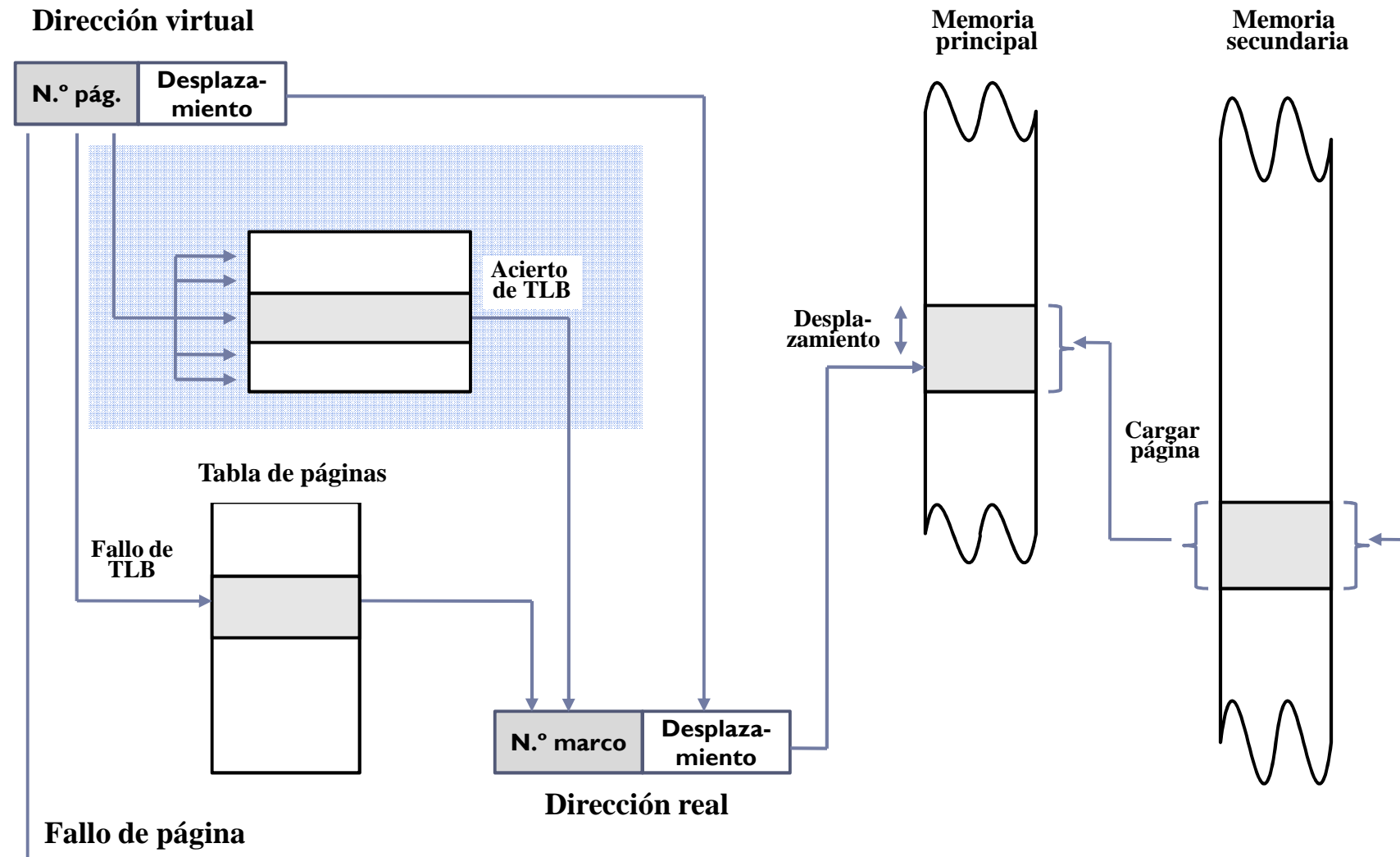


- ▶ TLB
- ▶ Tablas multinivel

Cache de traducciones

- ▶ Memoria virtual basado en tablas de páginas:
 - ▶ Problema: sobrecarga de acceso a memoria (2 accesos)
 - ▶ A la tabla de páginas/segmentos + al propio dato o instrucción
 - ▶ Solución: **TLB**
 - ▶ **Caché de traducciones**
- ▶ **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

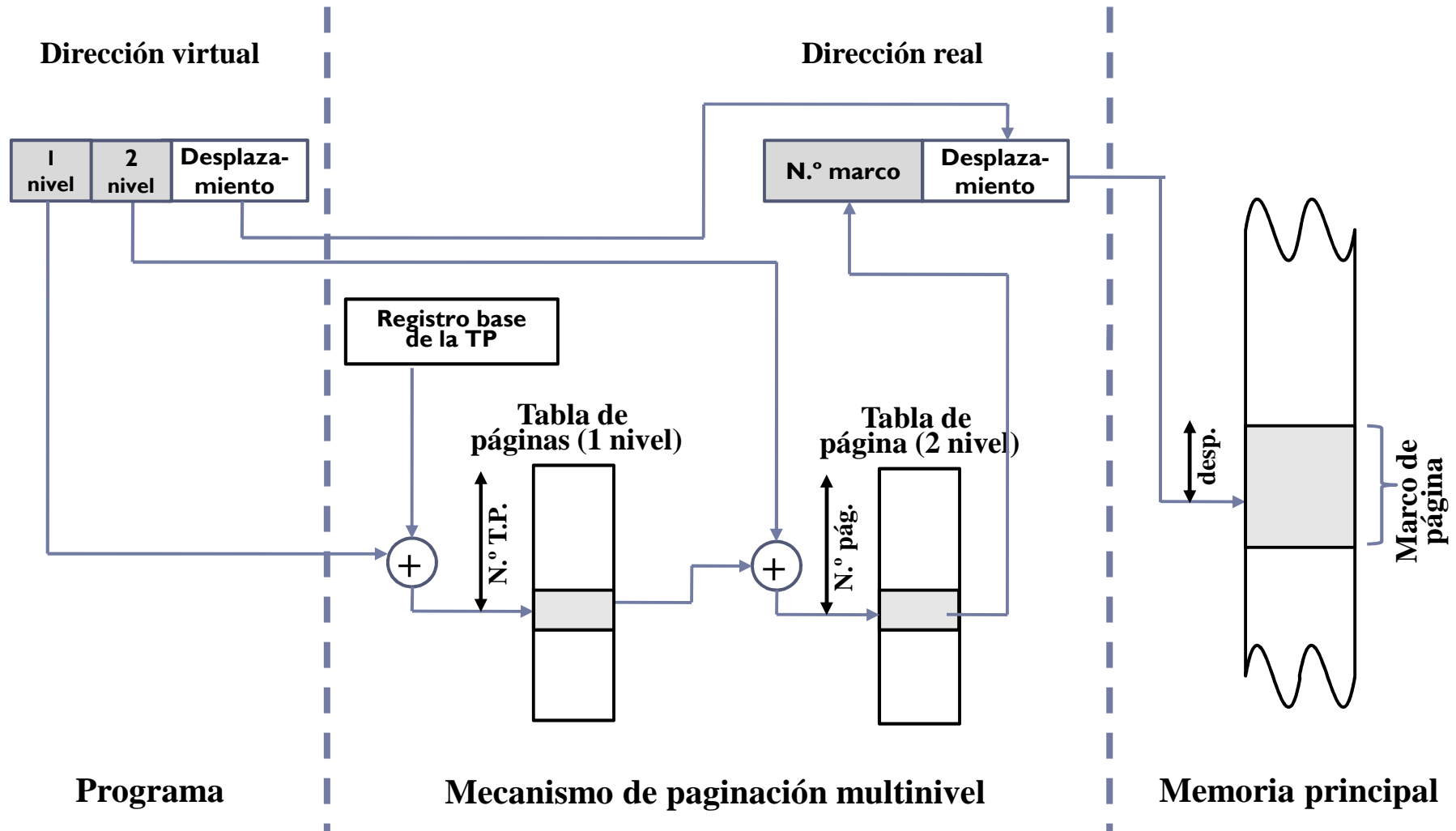
Traducción de direcciones (con TLB)



Tablas multinivel

- ▶ Memoria virtual basado en tablas de páginas:
 - ▶ Problema: consumo de memoria para las tablas
 - ▶ Ej.: páginas 4KB, dir. lógica 32 bits y 4 bytes por entrada:
 $2^{20} * 4 = 4\text{MB/proceso}$
 - ▶ Solución: **tablas multinivel**
- ▶ **Tabla multinivel**
 - ▶ Esquema de traducción en dos niveles:
 - ▶ En memoria la tabla de primer nivel
 - ▶ Solo en memoria las tablas de segundo nivel que se necesiten
 - ▶ Tablas de cada nivel mucho más compactas: $2^{10} * 4 = 4\text{KB/tabla}$

Tablas multinivel



Lección 5 (a)

La gestión de memoria

Diseño de Sistemas Operativos
Grado en Ingeniería Informática