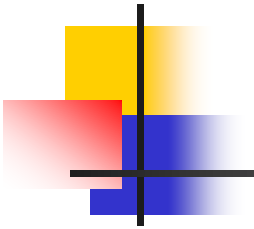
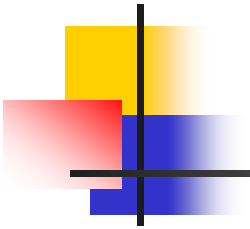


Sistemas Operativos 2021-2022

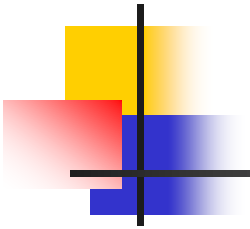
TEMA 4
Gestión de la Memoria Real

4.1. Introducción
4.2. Jerarquías de memoria
4.3. Direccionamiento
4.4 Direccionamiento dinámico
4.5. Mecanismos de gestión de la memoria real



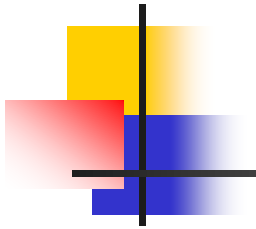
Introducción
Conceptos de gestión de memoria

- Objetivos:
  - Gestionar eficientemente la jerarquía de memoria
  - Lograr transparencia ante los procesos
  - Independizar la ejecución de los procesos de su ubicación física
  - Ofrecer protección contra accesos inválidos
    - Accidentales y/o malintencionados
    - Dentro del proceso, al espacio de otro proceso o del SO
  - Permitir la compartición de memoria entre procesos
    - Código ejecutable o estructuras de datos

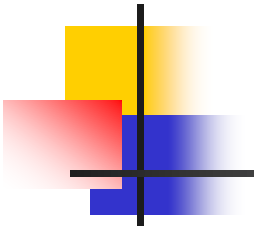


Introducción
Conceptos de gestión de memoria

- **Memoria Principal** (MP)
  - **Para que un programa se ejecute debe encontrarse en memoria principal, al menos, una parte (la instrucción máquina en curso)**
- **Gestor de Memoria**
  - Componente del S.O. que se encarga de las tareas relacionadas con la administración de la MP
    - Asignación de MP a los procesos que la solicitan
    - Localización de espacios libres y ocupados
    - Aprovechamiento máximo de dicha memoria



Jerarquía de memoria
Índice

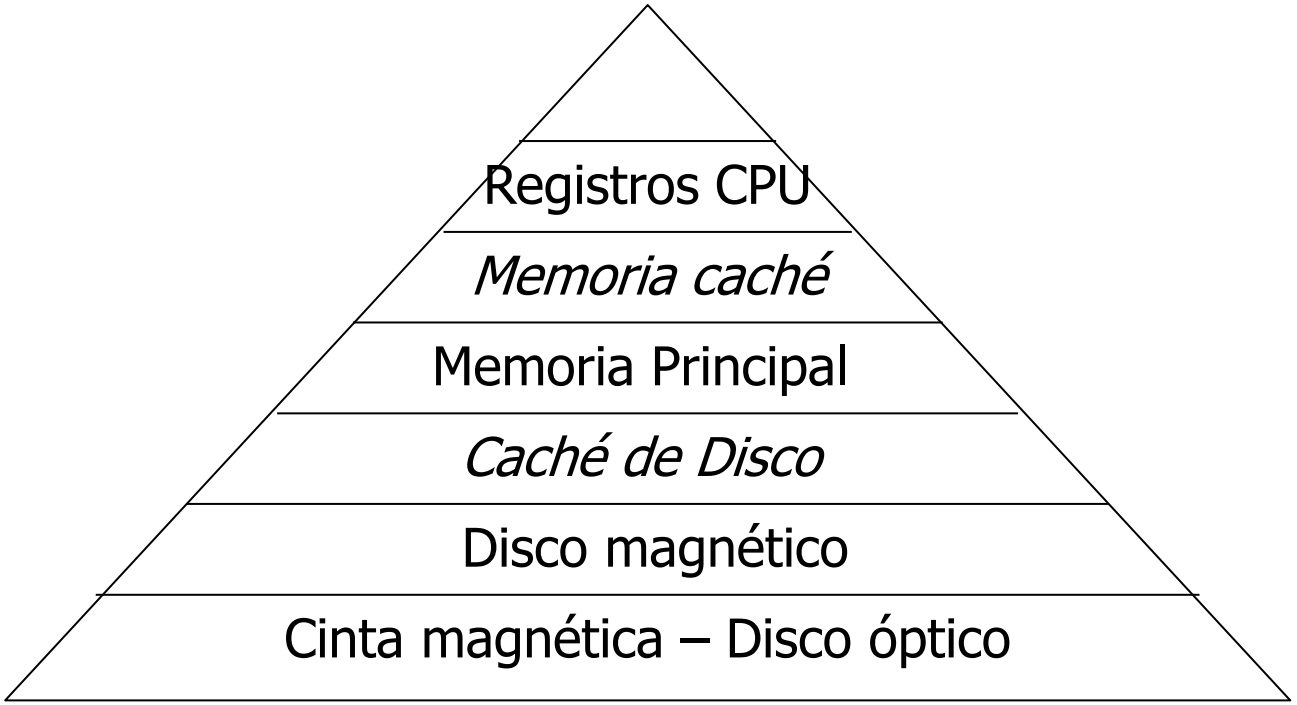


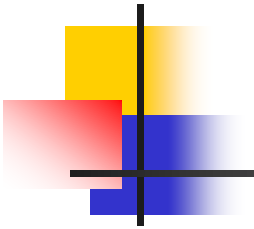
Jerarquía de memoria
Introducción

- Elementos clave en el diseño de la memoria
  - Capacidad, velocidad de acceso y coste
  - Existen diversas tecnologías para diversos tipos de memoria
    - A menor tiempo de acceso, mayor coste por bit
    - A mayor capacidad, menor coste por bit
    - A mayor capacidad, mayor tiempo de acceso

Jerarquía de memoria
Introducción

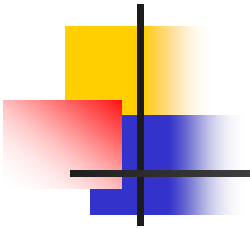
- Jerarquía de memoria





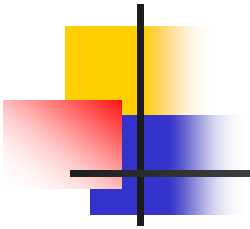
Jerarquía de memoria Introducción

- A medida que se desciende por la jerarquía
  - Disminuye el coste por bit
  - Aumenta la capacidad
  - Aumenta el tiempo de acceso
  - Disminuye la frecuencia de acceso por parte del procesador
    - Debido al principio de localidad y las cachés de memoria



Jerarquía de memoria
Principio de localidad de referencias (1)

- Durante la ejecución de un proceso, las referencias a memoria que éste genera tienden a estar agrupadas en posiciones de memoria muy próximas
  - Programa esencialmente secuencial
  - Estructuras iterativas que repiten un grupo de instrucciones
  - Cálculos con vectores y registros



Jerarquía de memoria
Principio de localidad de referencias (y 2)

- Consecuencias
  - Cuando un proceso genera ciertas direcciones hay una probabilidad muy alta de que vuelva a hacerlo
  - Organizar la memoria en varios niveles mediante un **sistema de cachés**

Jerarquía de memoria
Funcionamiento de la jerarquía (1)

- Memoria principal sin caché (Sistema antiguo)

- Memoria principal CON cache (Sistema actual)

Ultrarrápido
Frecuencia alta

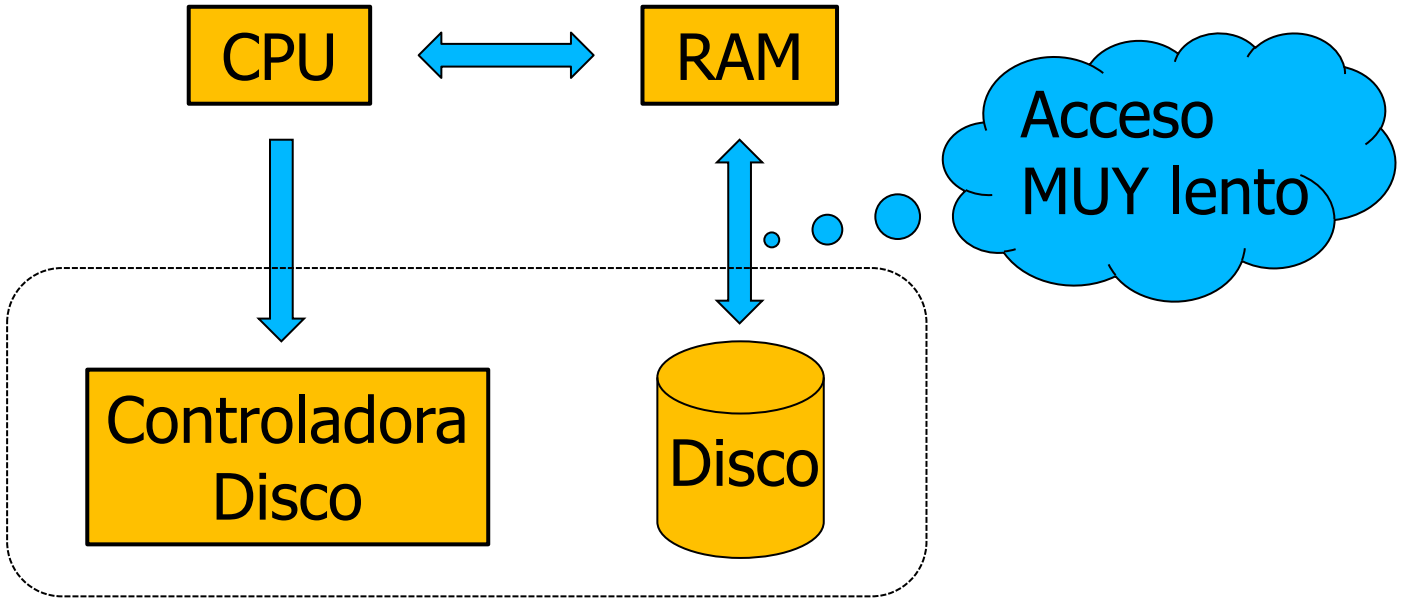
Muy rápido
Frecuencia baja
(Sólo si falla la caché)

Jerarquía de memoria
Funcionamiento de la jerarquía (2)

- Memoria caché
  - La memoria caché es un buffer intermedio entre la memoria principal y los registros del procesador
  - Todos los datos pasan por ella antes de llegar a los registros
  - **Muy cara → Tamaño muy pequeño y gran velocidad de acceso**
  - Compensa su uso por el principio de localidad → Tasa de aciertos ALTA
  - No es visible al procesador, que referencia direcciones de memoria principal como si la cache no existiera

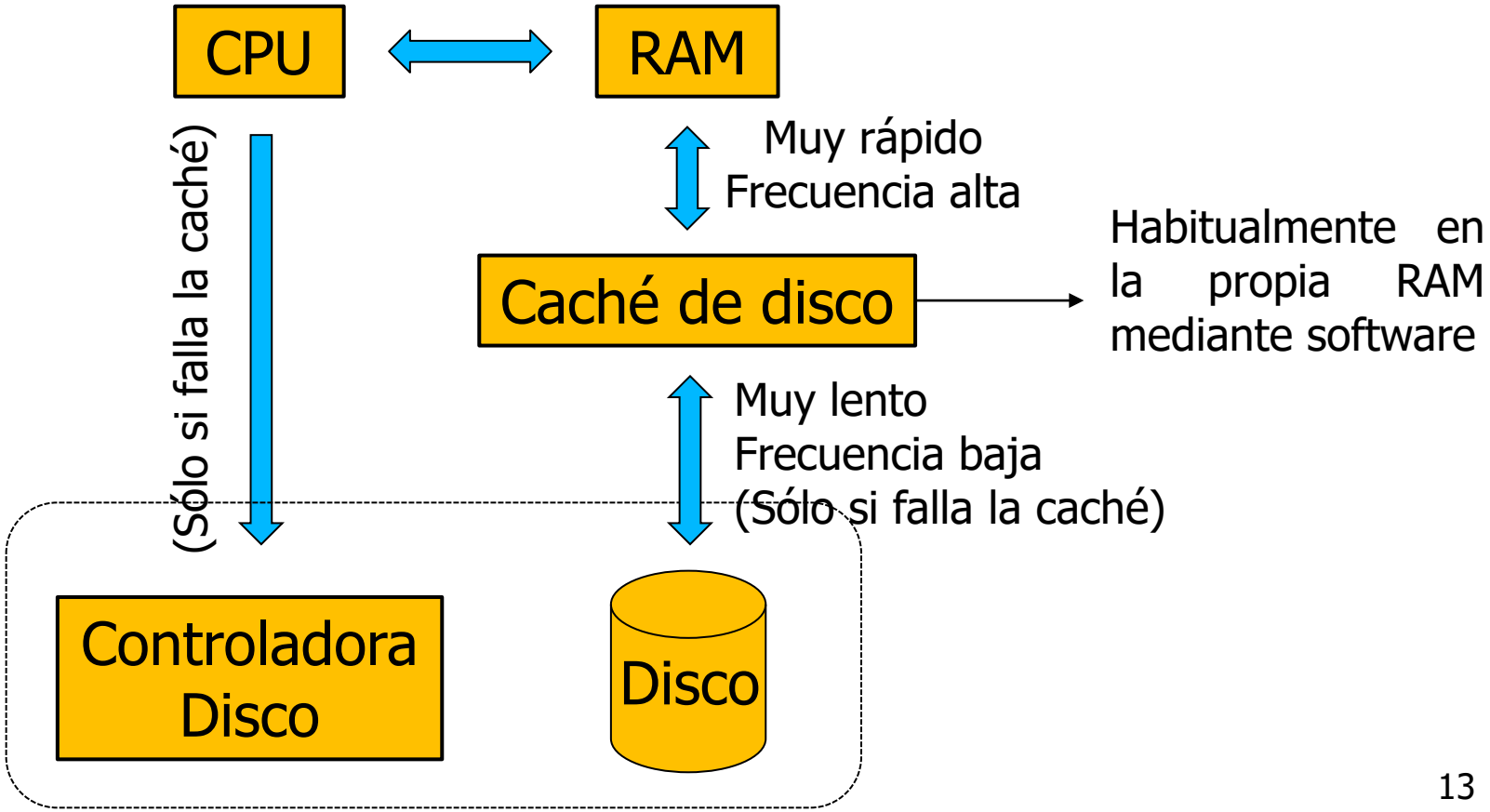
Jerarquía de memoria
Funcionamiento de la jerarquía (3)

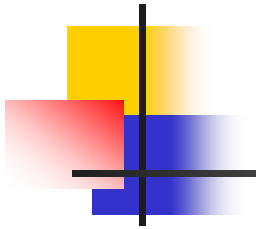
- Acceso a disco sin caché (Sistema antiguo)



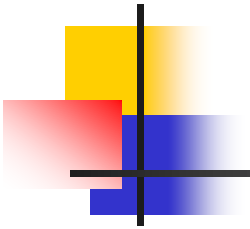
Jerarquía de memoria
Funcionamiento de la jerarquía (4)

- Acceso a disco Con caché (Sistema actual)



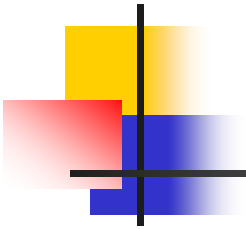


Direccionamiento
Índice



Direccionamiento
Tipos de direcciones

- Hay varios tipos de direcciones:
  1. Direcciones ***simbólicas***
  2. Direcciones ***relativas***
  3. Direcciones ***lógicas***
  4. Direcciones ***físicas***
- La CPU SOLO puede trabajar con direcciones físicas
  - Si no se tienen direcciones físicas hay que llegar hasta ellas antes de ejecutar un proceso
  - A través de un mecanismo de varios pasos



Direccionamiento
Pasos de generación de direcciones físicas

- Paso 0: **Direcciones *símbolicas***
  - Los programas de medio/alto nivel están escritos con direcciones simbólicas → No son numéricas, son nemotécnicas
  - Ejemplos: nombres de variables, nombres de función, etc
  - Un programador recuerda mejor este tipo de direcciones.
  - Está muy lejos de que lo entiende la CPU

Direccionamiento
Pasos de generación de direcciones físicas

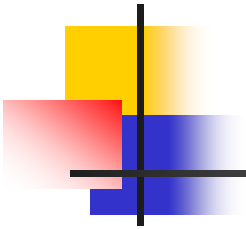
- Ejemplo de programa con **direcciones *simbólicas*** (indicadas en negrita)
  - Tenemos dos ficheros fuentes en C: ***functions.c*** y ***main.c***

```
int my_function(int a)
{
    int local=3;
    return a * local;
}
```

functions.c

```
extern int my_function(int);
int global = 2
int local_function(){
    return 22;
}
int main(){
    int result;
    result = my_function(global)
    printf("%f\n", result)
    local_function();
    return 0;
}
```

main.c



Direccionamiento
Pasos de generación de direcciones físicas

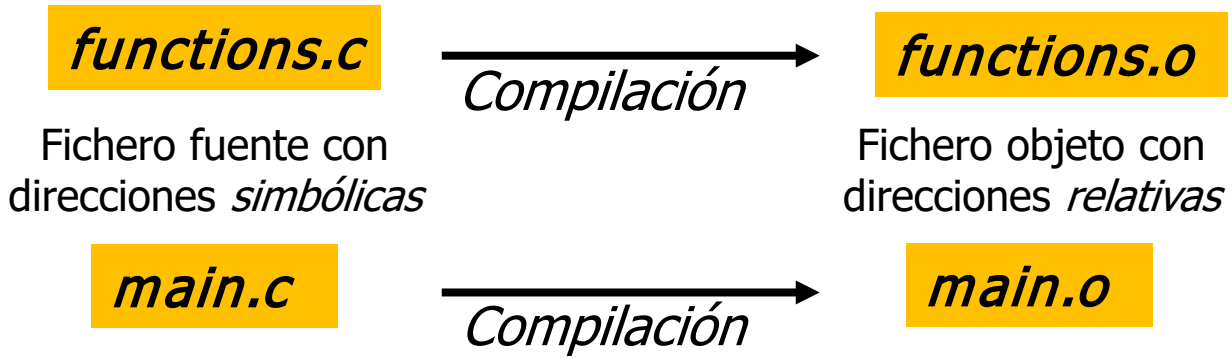
- Paso 1: **Compilación**
  - Pasa a código binario (código máquina) el programa indicado.
  - También se transforman las **direcciones** ***simbólicas*** en direcciones numéricas
  - Estas direcciones pueden ser:
    - **Direcciones** ***relativas*** (lo más habitual)
    - **Direcciones** ***físicas*** (muchos inconvenientes)
  - Se aplica a cada fichero fuente dando lugar a un fichero objeto que contiene lo indicado anteriormente

Pasos de generación de direcciones físicas

- Paso 1: **Compilación** *(continuación)*
  - Si en esta etapa el código objeto tiene **direcciones *físicas***, el código final del fichero ejecutable no sería reubicable → Sólo se puede cargar en una zona concreta de la memoria principal ☹ ☹ ☹
  - **Solucion**: Generar **direcciones *relativas*** → El código final del fichero ejecutable es reubicable → Se puede cargar en cualquier zona de la MP ☺
  - Una **direccion *relativa*** indica un **desplazamiento desde el inicio del fichero** siendo la dirección del primer byte del fichero la dirección cero.

Direccionamiento
Pasos de generación de direcciones físicas

- Ejemplo de compilación generando **direcciones *relativas***
  - Tenemos los dos ficheros fuentes anteriores: ***functions.c*** y ***main.c***
  - Habrá que hacer DOS compilaciones:



Direccionamiento
Pasos de generación de direcciones físicas

- Ejemplo de compilación (*continuación*)
  - El fichero ***functions.o*** (código fuente en transparencia 17) no tiene variables globales → No tiene segmento de datos

functions.o

Dir. Relativa	*Inicio my_function*
0	push 3
2	ld ac, (bs-4) * (bs-16)
6	pop
7	ret
8	

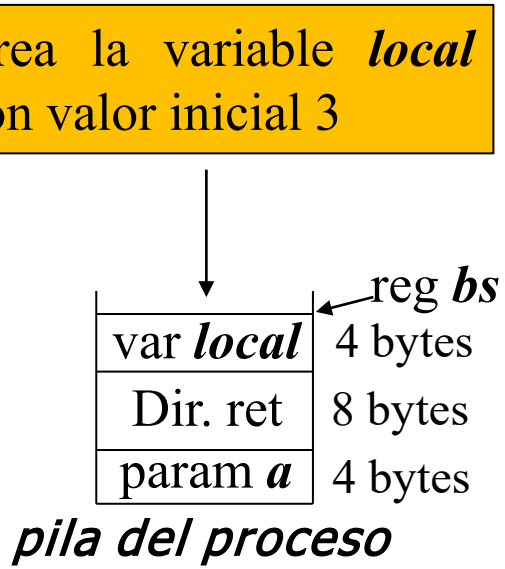
- Código máquina ficticio
- Enteros de 4 bytes
- Bs → base stack → Base de la pila
- Tamaño total del fichero: 8 bytes
- Se dejan los valores de retorno de las funciones en el registro ***ac***

Direccionamiento
Pasos de generación de direcciones físicas

- Ejemplo de compilación (*continuación*)

functions.o (continuación)

Dir. Relativa	*Inicio my_function*
0	push 3
2	ld ac, (bs-4) * (bs-16)
6	pop
7	ret
8	



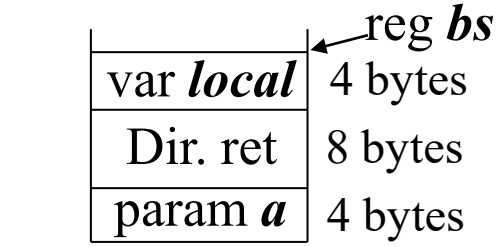
Direccionamiento
Pasos de generación de direcciones físicas

- Ejemplo de compilación (*continuación*)

functions.o (continuación)

Dir. Relativa	*Inicio my_function*
0	push 3
2	ld ac, (bs-4) * (bs-16)
6	pop
7	ret
8	

Multiplica la base de la pila – 4 bytes (variable ***local***) con la base de la pila - 16 bytes (parámetro ***a***) y lo mete en el registro acumulador



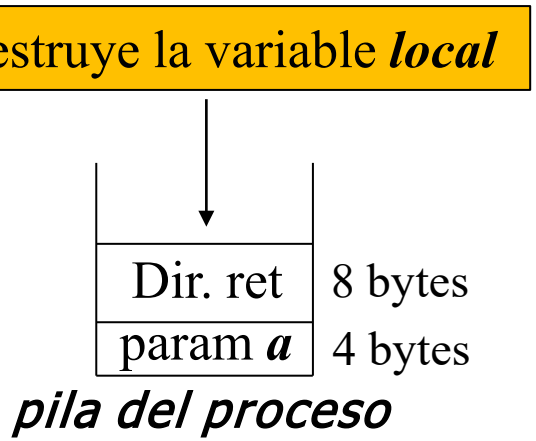
pila del proceso

Direccionamiento
Pasos de generación de direcciones físicas

- Ejemplo de compilación (*continuación*)

functions.o (continuación)

Dir. Relativa	*Inicio my_function*
0	push 3
2	ld ac, (bs-4) * (bs-16)
6	pop
7	ret
8	



Direccionamiento
Pasos de generación de direcciones físicas

- Ejemplo de compilación (*continuación*)
  - El fichero ***main.o*** (código fuente en transparencia 17) tiene variables globales → Tiene código y datos

main.o

Dir. Relativa	***Zona de datos***
0	2
	Zona de código
4	*Inicio de la func. local_function*
4	ld ac,22
6	ret

Variable ***global*** inicializada a 2

Valor de retorno 22 al reg. ***ac***

Direccionamiento
Pasos de generación de direcciones físicas

- Ejemplo de compilación (*continuación*)

main.o (continuación)

	Zona de código
Dir. Relativa	*Inicio de la func. main*
7	push
8	push (**0**)
10	call **????**
14	pop
15	ld (bs-4), ac

Crea la variable local ***result***

Contenido de la **Dir.** ***relativa*** **0** a pila (parámetro ***global***)

Llamada a la función ***my_function*** No podemos saber su **Dir.** ***relativa*** hasta la fase de enlazado → Función externa

Destruimos el parámetro ***global*** al regresar de la función

Cargamos el reg. ***ac*** (valor retornado) en la variable local ***result***

Direccionamiento
Pasos de generación de direcciones físicas

- Ejemplo de compilación (*continuación*)

main.o (continuación)

Dir. Relativa	*Cont. de la func. main*
17	ld reg1,(bs-4)
19	push "%f\n"
21	push reg1
23	call **????**
24	pop 8
28	call **4**
32	xor ac,ac
33	ret
34	

Variable local ***result*** a reg. ***reg1***

Primer parámetro de ***printf*** a pila

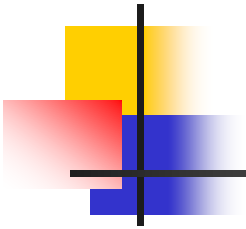
Segundo parámetro de printf a pila

Llamada a la función ***printf*** No podemos saber su **Dir. *relativa*** hasta la fase de enlazado → Función externa

Destruimos los parámetros después de la llamada (8 bytes)

Llamada a la función ***local_function*** (**Dir. *relativa* 4**)

Valor 0 al reg. ***ac*** y retorna

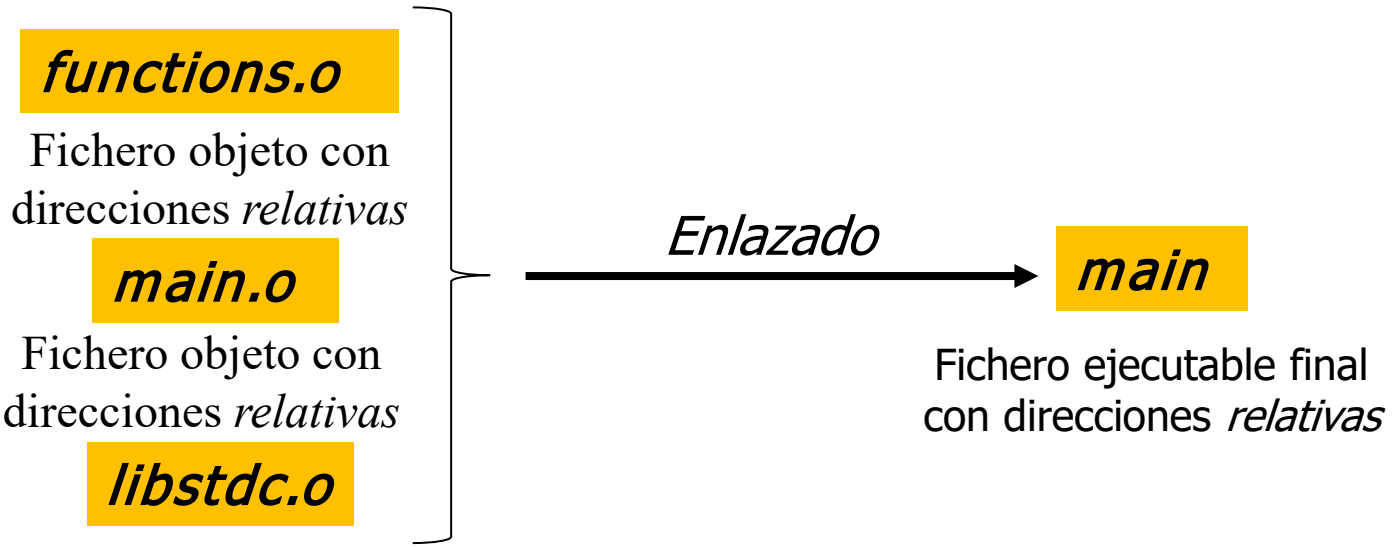


Direccionamiento
Pasos de generación de direcciones físicas

- Paso 2: **Enlazado**
  - En esta etapa se conserva el tipo de direcciones que haya generado el paso de compilación (***relativas*** o ***físicas***)
  - Une (literalmente) todos los ficheros objetos unos detrás de otros para generar el fichero ejecutable final
  - **Tendrá que reajustar las direcciones relativas generadas en la etapa anterior** sumándoles el tamaño en bytes de todos los ficheros objeto anteriores
  - Asigna las direcciones relativas de las funciones externas

Direccionamiento
Pasos de generación de direcciones físicas

- Ejemplo de enlazado generando **direcciones *relativas***
  - Tenemos los dos ficheros fuentes anteriores: ***functions.o*** y ***main.o***
  - Vamos a suponer que se une ***main.o*** detrás de ***functions.o*** y al final el código objeto con la librería de C (para el ***printf***)



Direccionamiento
Pasos de generación de direcciones físicas

- Ejemplo de enlazado (*continuación*)
  - El fichero ejecutable final ***main*** sería:

Fichero ejecutable final *main*

	Zona de código
Dir. Relativa	*Inicio my_function*
0	push 3
2	ld ac, (bs-4) * (bs-16)
6	pop
7	ret

Código procedente de ***functions.o***

Dir. Relativa	***Zona de datos***
8	2
	Zona de código
12	*Inicio de la func. local_function*
12	ld ac,22
14	ret

Código procedente de ***main.o***
+8 a todas las direcciones rel.

Direccionamiento
Pasos de generación de direcciones físicas

- Ejemplo de enlazado (*continuación*)

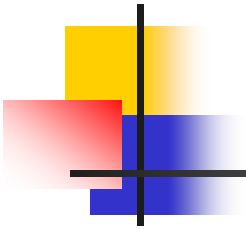
Fichero ejecutable final *main* *(continuación)*

Dir. Relativa	*Inicio de la func. main*
15	push
16	push (**8**)
19	call **0**
22	pop
23	ld (bs-4), ac
25	ld reg1,(bs-4)
27	push "%f\n"

Código procedente de ***main.o***
+8 a todas las direcciones rel.

Dir. Relativa	*Cont. de la func. main*
29	push reg1
31	call **1040 (*)**
32	pop 8
36	call **12**
40	xor ac,ac
41	ret
42	

Código procedente de ***main.o***
+8 a todas las direcciones rel.



Direccionamiento
Pasos de generación de direcciones físicas

- Ejemplo de enlazado (*continuación*)
  - (*) Nótese que la **dirección** ***relativa*** 1040 de la transparencia anterior es el comienzo de la función ***printf*** (inventada porque no sabemos el código de *libstdc.o*)
  - Todas las direcciones del ejecutable final son **direcciones** ***relativas*** → Código reubicable en MP

Direccionamiento
Pasos de generación de direcciones físicas

- Paso 3: **Carga del programa ejecutable en MP → PLP**
  - Recordemos que el programa ejecutable puede tener:
    - **A.** **Direcciones** ***relativas*** (lo más habitual) → Reubicable ☺
    - **B.** **Direcciones** ***físicas*** (muy raro) → NO reubicable ☹

Direccionamiento
Pasos de generación de direcciones físicas

- Paso 3A: **Carga del programa ejecutable en MP → PLP (cuando tiene direcciones *físicas*)**
  - El PLP DEBE cargar el programa ejecutable en la dirección física indicada por el fichero ☹ ☹ ☹
  - Como ya hay **direcciones *físicas*** finales no es necesario realizar más pasos y el fichero ejecutable se cargará tal cual sin ninguna modificación ☺

Direccionamiento
Pasos de generación de direcciones físicas

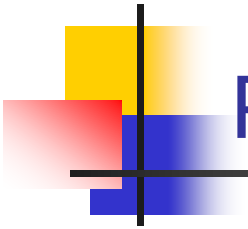
- Paso 3B: **Carga del programa ejecutable en MP → PLP (cuando tiene direcciones *relativas*)**
  - El PLP PUEDE cargar el programa ejecutable donde quiera ☺ ☺
  - El PLP puede optar, en este caso 3B, por cargar el programa ejecutable de dos formas:
    1. Manteniendo las **direcciones *relativas*** (lo más habitual) → Proceso tendrá **direcciones *relativas*** (llamadas ***lógicas***) y será reubicable mientras se está ejecutando ☺ ☺
    2. Sumando a todas las **direcciones *relativas*** la dirección física inicial donde se carga el programa → Proceso tendrá direcciones físicas y NO será reubicable ☹ ☹

Pasos de generación de direcciones físicas

- Paso 4: **Traducción en tiempo de ejecución (MMU)**
  - Esta etapa SOLO existe (y debe existir) si la etapa anterior está en el caso (3B1) → Proceso tiene **direcciones** ***lógicas*** (relativas)
  - Como la CPU solo entiende direcciones lógicas alguien tiene que pasar la **dirección** ***lógica*** a **dirección** ***física***
    - Lo hace un hardware especial → MMU
    - De esta forma se hace MUY rápido

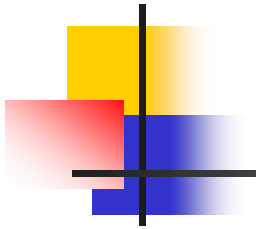
Resumen generación de direcciones físicas

- Las direcciones físicas se pueden generar en UNA de estas etapas:
  - **Tiempo de compilación y enlazado**
    - El programa ejecutable no es reubicable ☹ ☹ ☹
    - Obviamente el proceso tampoco ☹ ☹
  - **Tiempo de carga**
    - El programa ejecutable es reubicable ☺ ☺ ☺
    - El proceso NO es reubicable ☹ ☹
  - **Tiempo de ejecución**
    - El programa ejecutable es reubicable ☺ ☺ ☺
    - El proceso es reubicable mientras se ejecuta ☺ ☺



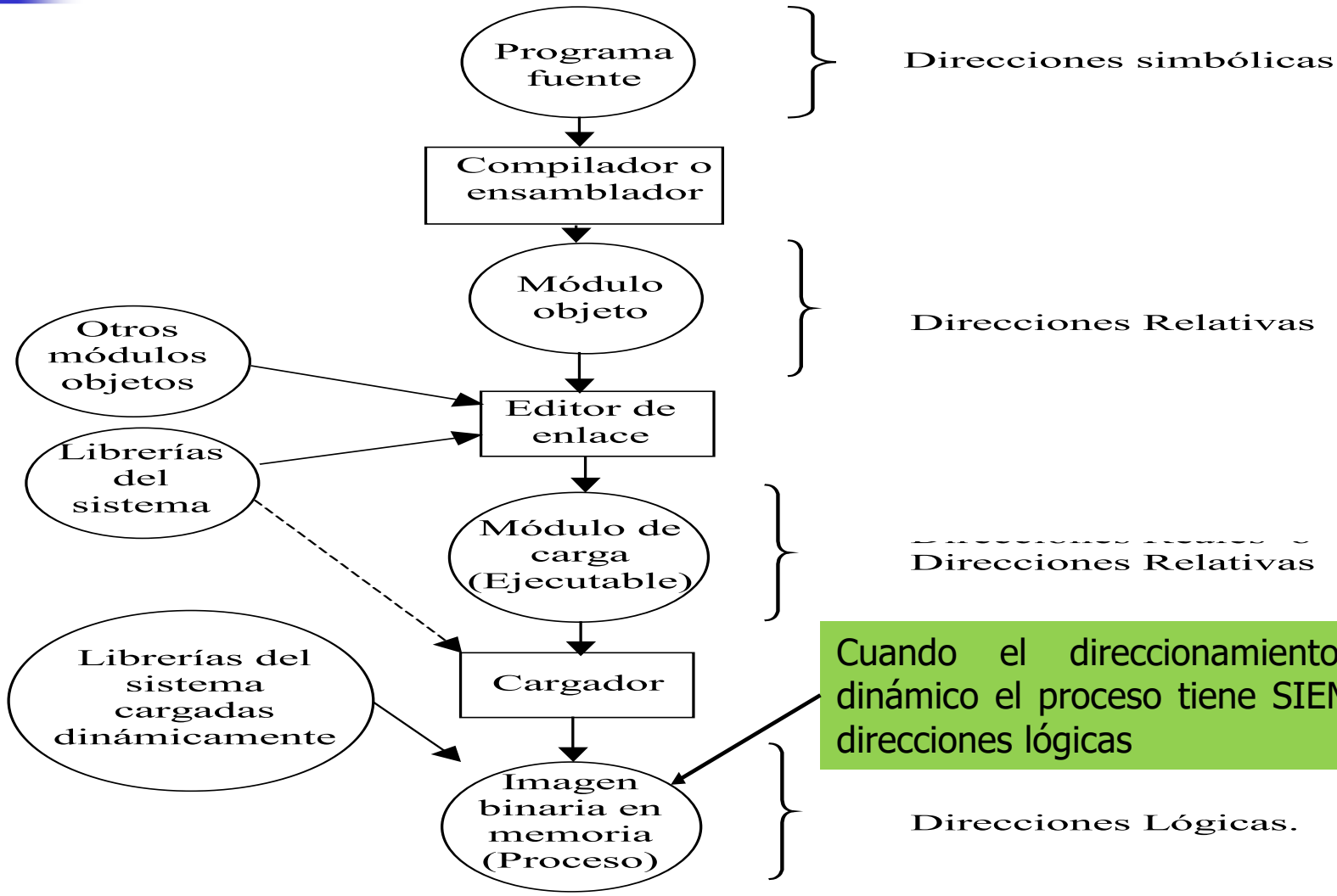
Direccionamiento
Resumen generación de direcciones físicas

- **Correspondencia estática**
  - Las **direcciones** ***físicas*** se generan en tiempo de compilación/enlazado o bien en tiempo de carga
  - Ya no se usa.
- **Correspondencia dinámica**
  - Las **direcciones** ***físicas*** se generan en tiempo de ejecución.
  - Lo que se usa en la actualidad
  - Necesita de un hardware especial → MMU

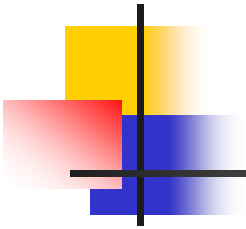


Direccionamiento dinámico
Índice

Direccionamiento dinámico Resumen



Cuando el direccionamiento es dinámico el proceso tiene SIEMPRE direcciones lógicas



Direccionamiento dinámico
Espacio de direcciones lógicas y físicas (1)

- Dirección relativa
  - Referencia a posición de memoria relativa al comienzo de un programa o módulo de un programa
- Dirección lógica
  - Referencia a posición de memoria relativa al comienzo de un proceso
- Espacio de direcciones lógicas de un proceso
  - Conjunto de direcciones lógicas abarcada por un proceso
  - La dirección lógica más baja es la cero y la más alta, el tamaño del proceso-1

Direccionamiento dinámico
Espacio de direcciones lógicas y físicas (y 2)

- Dirección física
  - Designa una posición real de la memoria principal
- Espacio de direcciones físicas de un proceso
  - Conjunto de posiciones de memoria física correspondientes a sus direcciones lógicas
  - Estas direcciones varían dependiendo de dónde se haya ubicado al proceso en memoria principal

Direccionamiento dinámico
Correspondencia entre direcciones lógicas y físicas (1)

- Correspondencia o reubicación
  - Función de traducción entre dos espacios de direcciones
    - f: L $\rightarrow$ F
- *L* (Espacio de direcciones lógicas)
  - El programador trabaja al margen de la localización que su código vaya a tener en memoria principal
    - Desconoce dónde se va a ubicar a la hora de ejecutarse
    - El programa debe poder ejecutarse en diferentes ordenadores con diferente capacidad de memoria
- *F* (Espacio de direcciones físicas)
  - El programa se carga físicamente en unas posiciones de memoria principal determinadas

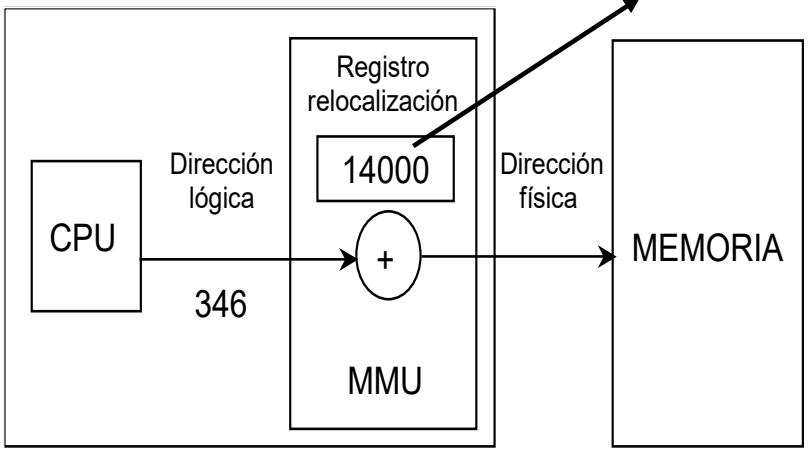
Direccionamiento dinámico
Correspondencia entre direcciones lógicas y físicas (2)

- Correspondencia o reubicación (f)
  - Correspondencia entre las direcciones de memoria a las que hace referencia las instrucciones de un programa y la dirección de memoria principal donde se encuentra físicamente
  - El gestor de memoria (HW + SW) implementa esta función

Direccionamiento dinámico
Unidad de gestión de memoria (MMU)

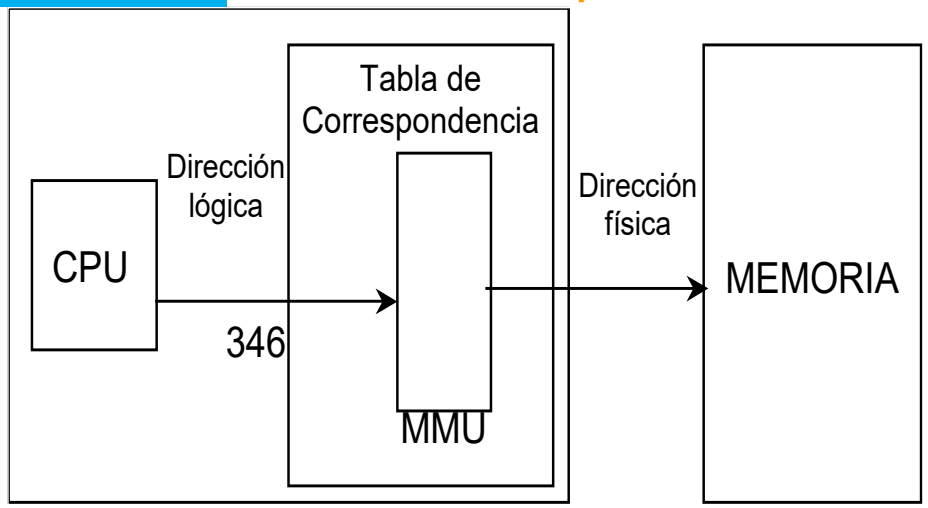
- Dispositivo hardware para traducir direcciones lógicas a físicas en tiempo de ejecución
- Diversos esquemas de traducción
  - Registro base o de relocalización

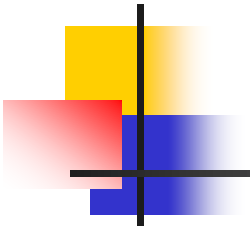
Contiene la dirección física de inicio del **proceso en ejecución**



Tarjeta CPU

  - Tabla de correspondencia





Direccionamiento dinámico
Unidad de gestión de memoria (MMU)

- El esquema de la MMU visto en la transparencia anterior ESTÁ INCOMPLETO
- Le falta la fase de protección que se verá MAS ADELANTE

Mecanismos de gestión de la memoria real
Índice

Mecanismos de gestión de memoria real
Introducción

- Los dos grandes tipos de esquemas de memoria son:
  - ***Memoria real*** ☹
    - Los procesos se cargan SIEMPRE completos en *memoria principal* → Menor grado de multiprogramación ☹ ☹
    - Medianamente complejo de implementar ☺
  - ***Memoria virtual*** ☺
    - Los procesos se cargan PARCIALMENTE en *memoria principal* → Mayor grado de multiprogramación ☺ ☺
    - Hay un respaldo del proceso complete en *memoria secundaria*
    - Bastante complejo de implementar ☹

Mecanismos de gestión de memoria real
Introducción

- En esta primera parte del tema nos vamos a centrar en ***memoria real con:***
  - ***Asignación contigua*** ☹
    - Los procesos se cargan SIEMPRE CONTIGUOS en *memoria principal* → Menor aprovechamiento de las zonas libre de la *memoria principal* ☹ ☹
    - Muy sencillo de implementar ☺
  - ***Asignación no contigua*** ☺
    - Los procesos se pueden cargar de forma NO CONTIGUA en *memoria principal* → Mayor aprovechamiento de las zonas libres de la *memoria principal* ☺ ☺
    - Algo complicado de implementar ☹

Mecanismos de gestión de memoria real
Introducción

- **Esquemas de memoria real con *asignación contigua*** más conocidos:
  - Para sistemas *monoprogramados* ☹☹☹
    1. *Esquema de **un solo proceso***
  - Para sistemas *multiprogramados* ☺
    2. *Esquema de **particiones fijas***
    3. *Esquema de **particiones variables***

Mecanismos de gestión de memoria real
Introducción

- **Esquemas de memoria real con *asignación NO contigua*** más conocidos:
  - Para sistemas *multiprogramados* ☺ ☺
    1. *Esquema de **paginación simple***
    2. *Esquema de **segmentación simple***
    3. *Esquema de **segmentación paginada simple***

Mecanismos de gestión de memoria real
Introducción

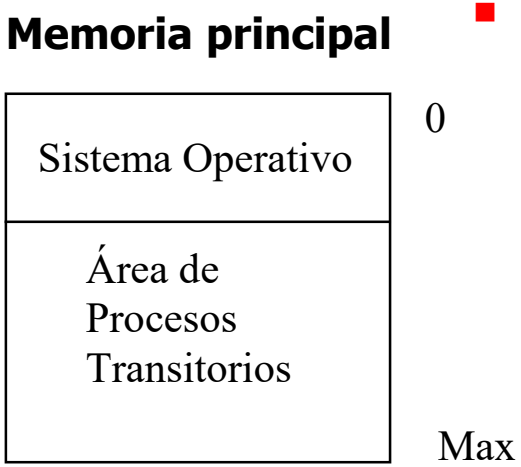
- Resumen de la evolución en los esquemas de la MP
  - Memoria real
    - Asignación contigua
      - **Sistemas de un solo proceso**
      - **Particiones fijas**
      - **Particiones variables**
    - Asignación no contigua
      - **Paginación simple**
      - **Segmentación simple**
      - **Segmentación paginada simple**
  - Memoria virtual
    - **Paginación por demanda**
    - **Segmentación por demanda**
    - **Segmentación paginada por demanda**

Mecanismos de gestión de memoria real
Introducción

- Vamos a ver ahora, con detalle, cada uno de los esquemas de memoria real
- De cada esquema tenemos que saber:
  - **Organización física de la memoria principal**
    - Cómo se divide la memoria principal
  - **Estructuras de datos necesarias** para implementar lo anterior (la organización física de la MP)
  - **Protección y traducción de direcciones lógicas a direcciones físicas**
    - Normalmente lo hará una MMU pero NO NECESARIAMENTE
  - **Estrategias de asignación**
    - Cómo se asigna memoria principal a los procesos
  - **Ventajas e inconvenientes**
    - Los esquemas tienen puntos débiles que motivan la aparición de un nuevo esquema ¿mejorado?

Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Monitor de un solo proceso
  - El primer esquema implementado
  - **Organización física de la memoria principal**
    - Sistema operativo (monitor)
      - Situado habitualmente en la parte inferior
      - Donde residen los vectores de interrupción
    - Área de procesos transitorios
      - Se coloca UN ÚNICO proceso de usuario que se elimina cuando termina y antes de cargar otro



Mecanismos de gestión de memoria real
Asignación contigua de memoria

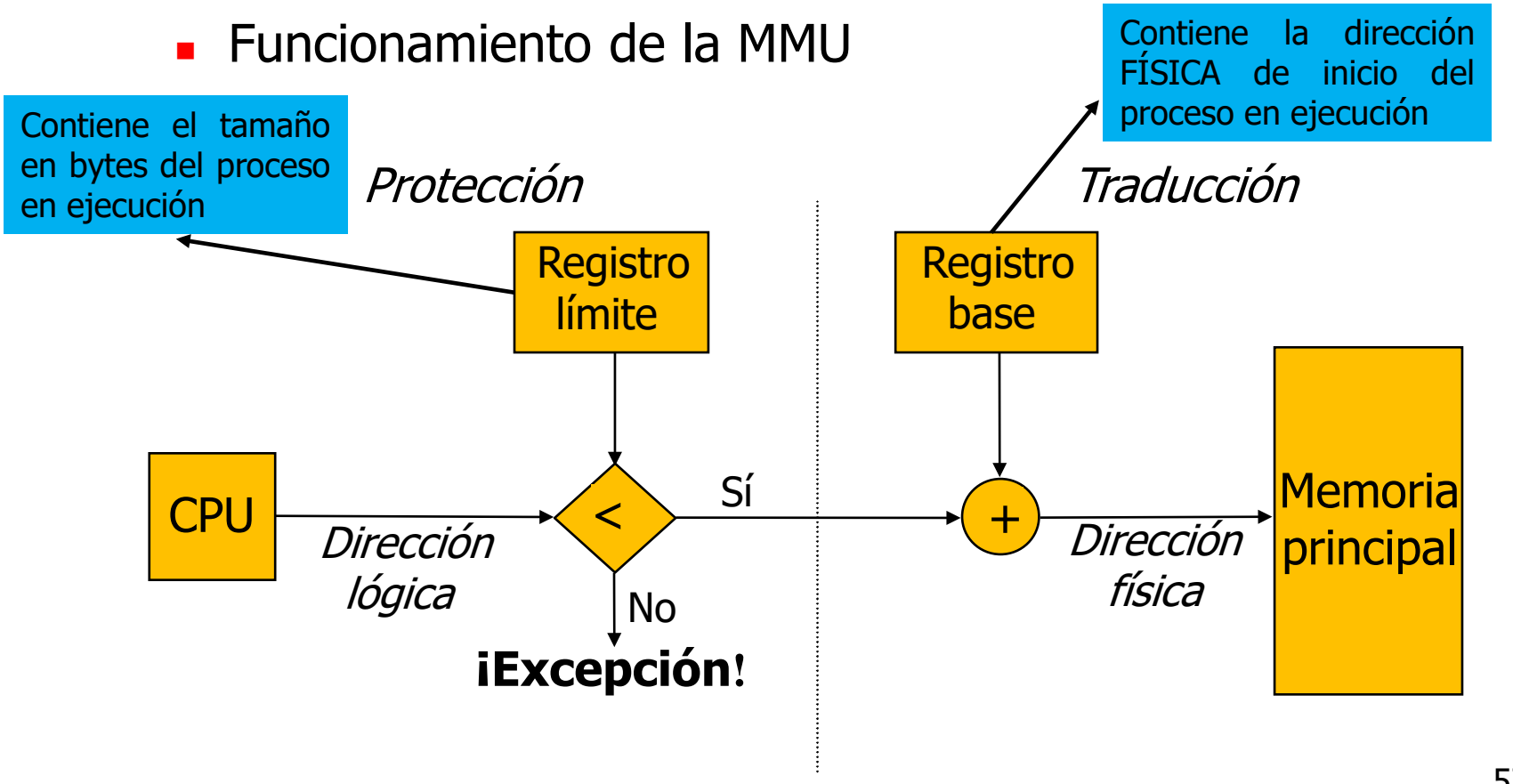
- Monitor de un solo proceso *(continuación)*
  - **No son necesarias estructuras de datos** para reflejar el estado de la MP porque su organización es MUY SIMPLE
  - **Protección**
    - El proceso no deberá NUNCA tener acceso una dirección lógica fuera de su espacio de direcciones lógicas
      - **0 <= dir_logica < Tamaño del proceso**
      - Si garantizo esto JAMÁS podrá acceder al SO ni a otros procesos
    - La protección suele realizarla el hardware (más seguridad) a través de la MMU

Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Monitor de un solo proceso *(continuación)*
  - **Traducción de direcciones**
    - Como el proceso tiene direcciones lógicas hay que traducirlas a direcciones físicas
    - Como ya vimos, solo es possible hacer esto mediante un hardware → La MMU
  - **La MMU tiene por tanto dos fases** que aplica a toda dirección lógica generada por el proceso (por la CPU)
    1. **Fase de *protección***
    2. **Fase de *traducción***

Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Monitor de un solo proceso *(continuación)*
  - Funcionamiento de la MMU

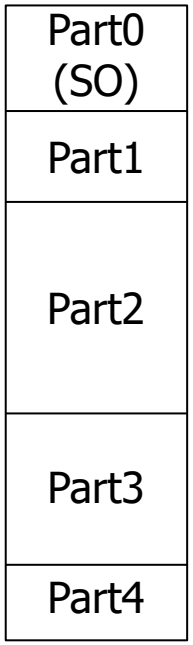


Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Monitor de un solo proceso *(continuación)*
  - **Ventajas e inconvenientes**
    - ✓ Muy simple, poca sobrecarga
    - ✗ Solo admite monoprogramación ☹☹☹☹

Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Particiones fijas
  - **Organización física de la memoria principal**



- División de la memoria en varios trozos (particiones)
  - Del mismo o distinto tamaño
  - Estáticas → No se pueden cambiar una vez arranca SO
- Asignación de particiones a procesos
  - No puede asignarse más de un proceso a una partición
  - Tamaño proceso $\leq$ tamaño partición
  - El sistema operativo ocupa una partición (Normalmente la primera)
- ¿ Y si no hay particiones libres?
  - Esperar a que finalice algún proceso
  - Suspensión de algún otro proceso
  - Creación del proceso en estado “SUSPENDIDO”

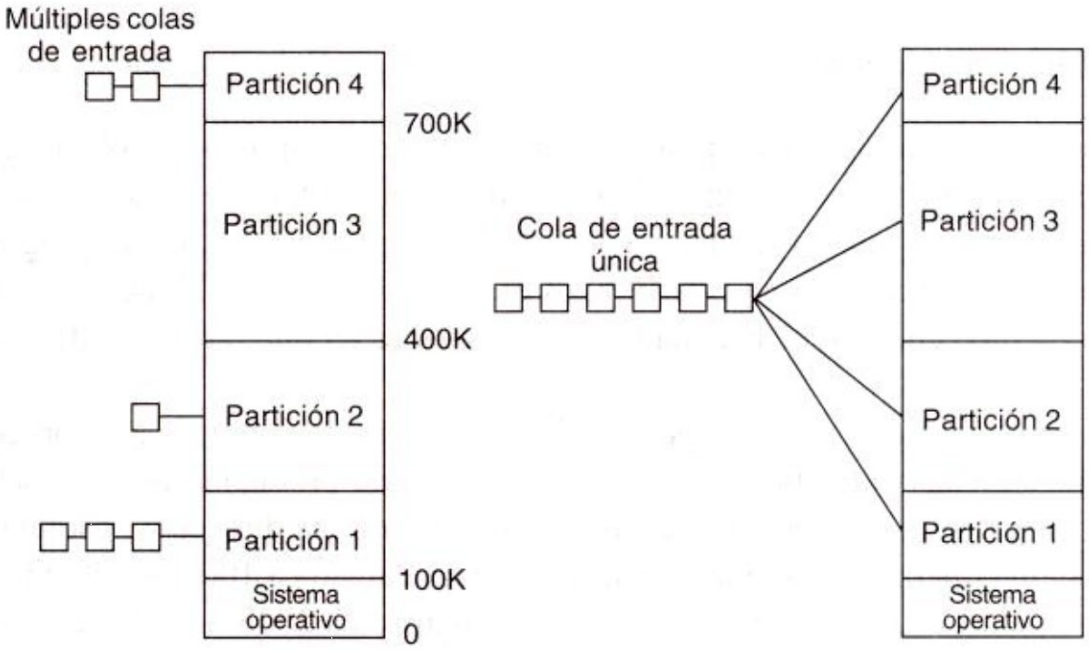
Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Particiones fijas *(continuación)*
  - **Estructuras de datos**
    - *Tabla de descripción de particiones*
      - Una entrada por partición

Estado (libre/asignada)	Base de la partición	Tamaño de la partición

Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Particiones fijas *(continuación)*
  - Cola de entrada de procesos a memoria principal
    - Una única cola para todas las particiones
    - Una cola distinta para cada partición

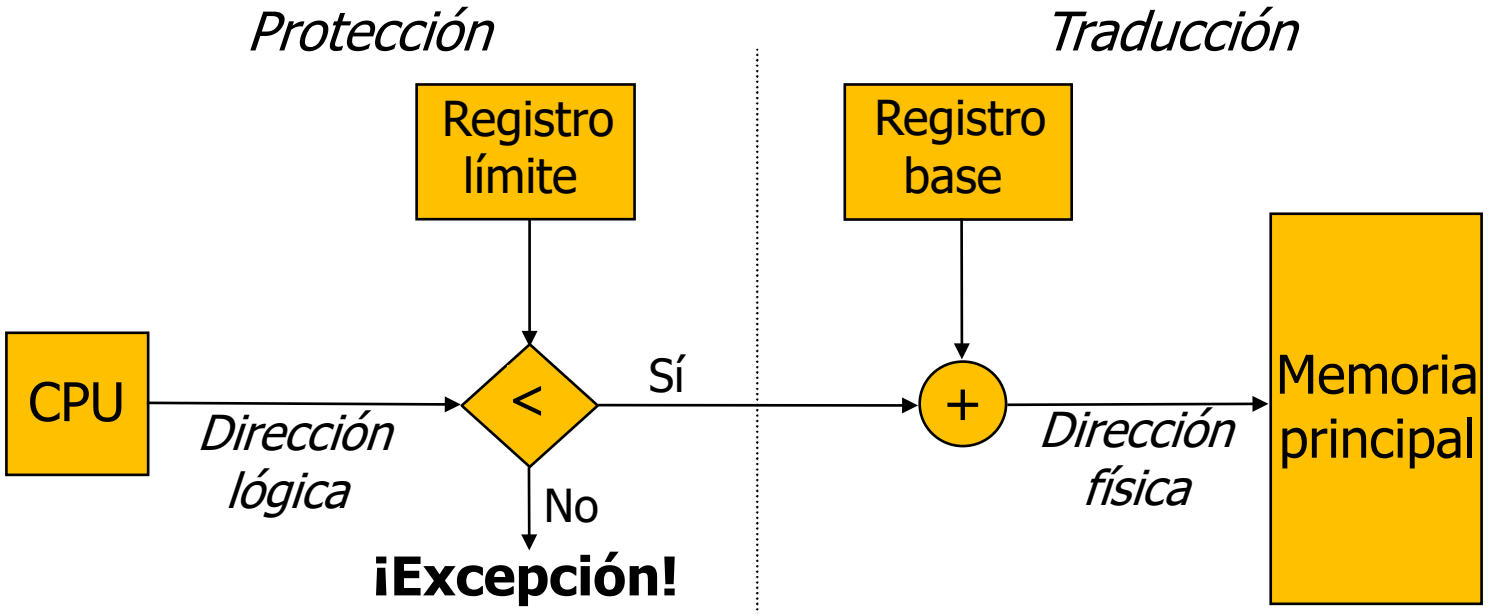


Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Particiones fijas *(continuación)*
  - **Estrategias de asignación**
    - *Primer ajuste*
      - Se asigna la PRIMERA partición libre en la que quepa el proceso (se empieza a buscar SIEMPRE desde la primera partición)
    - *Siguiente ajuste*
      - Se asigna la SIGUIENTE partición libre en la que quepa el proceso (se empieza a buscar donde se dejó la última vez)
    - *Mejor ajuste*
      - Se asigna a la partición libre MAS PEQUEÑA en la que quepa
    - *¿Peor ajuste?*

Mecanismos de gestión de memoria real
Asignación contigua de memoria

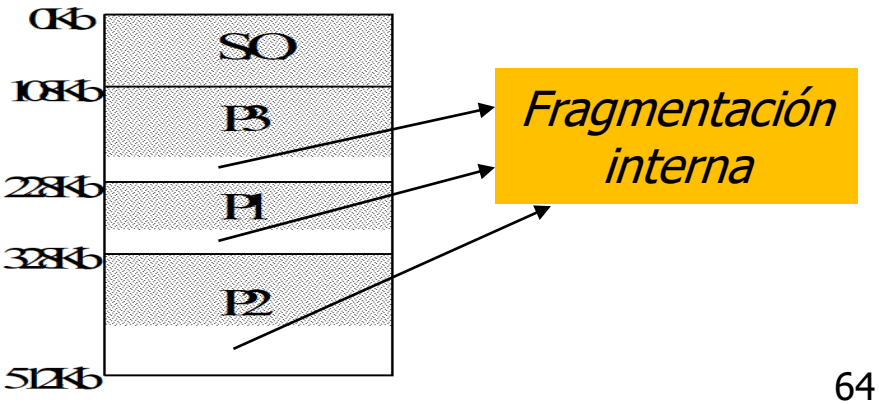
- Particiones fijas *(continuación)*
  - **Protección y Traducción**
    - Se implementa igual en TODOS los esquemas de memoria de asignación contigua



Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Particiones fijas *(continuación)*
  - **Ventajas e inconvenientes**
    - ✓ Muy simple, poca sobrecarga ☺
    - × Tamaño de proceso limitado al de la mayor partición ☹☹
    - × Grado multiprogramación limitado al nº de particiones ☹☹
    - × Fragmentación interna ☹
      - Tamaño proceso < tamaño partición → dentro de cada partición queda una zona de memoria no aprovechable

Nº Part.	Base	Tamaño	Libre
0	0Kb	100Kb	No
1	100Kb	120Kb	No
2	220Kb	100Kb	No
3	320Kb	180Kb	No

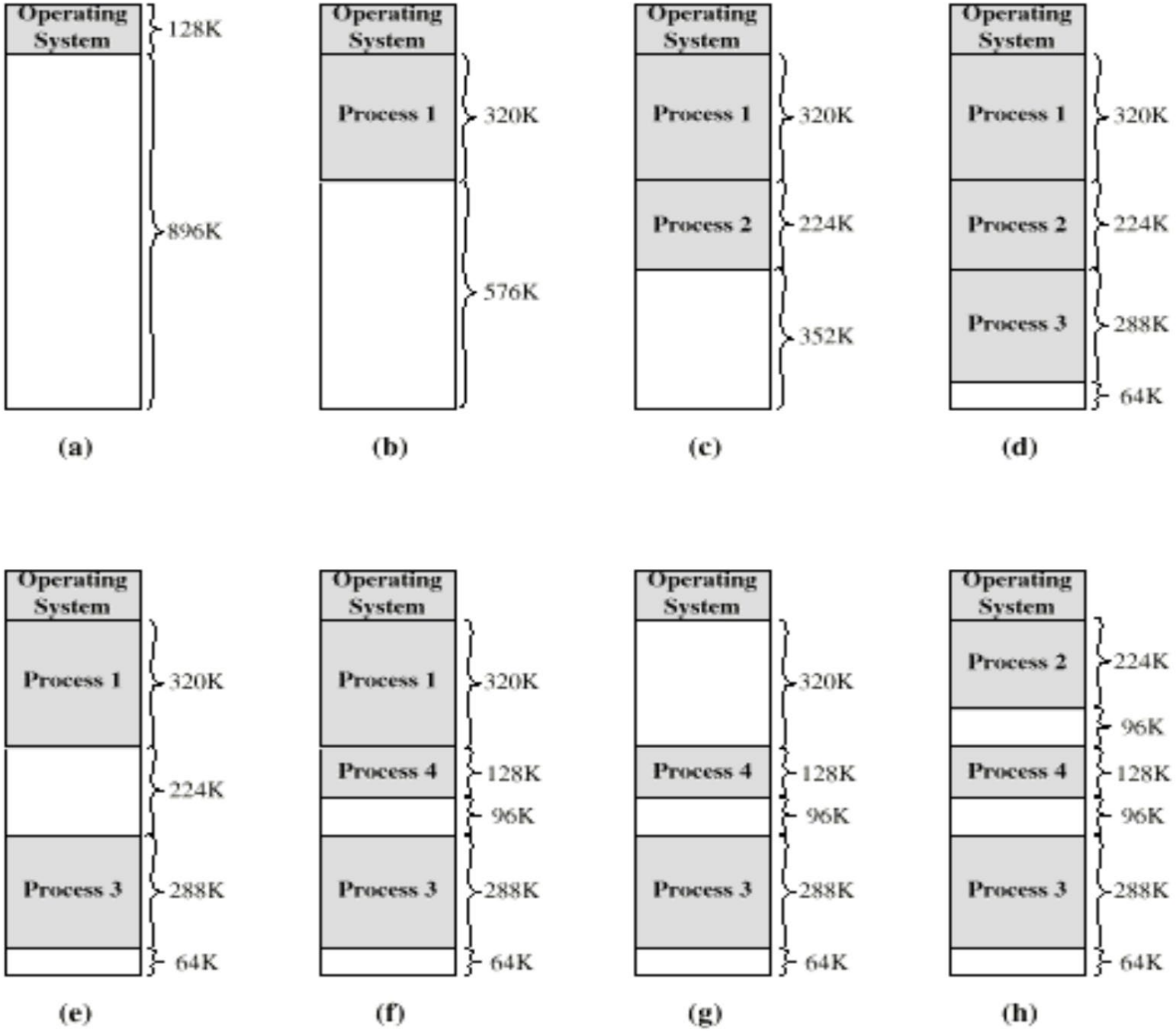


Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Particiones fijas *(continuación)*
  - ¿Cómo afecta la estrategia de asignación a la fragmentación interna?
    - Primer ajuste ☺ ☹
    - Siguiente ajuste ☺ ☹
    - Mejor ajuste ☺
    - Peor ajuste ☹

Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Particiones variables
  - **Organización física de la memoria principal**
    - Memoria Principal dividida en
      - Particiones (ocupadas por procesos o el SO)
      - Huecos (libres)
    - **Inicialmente** la memoria sólo contiene una partición con el S.O. y un gran hueco
    - Particiones son dinámicas
      - Varían en número y longitud
    - Cada proceso obtiene exactamente la MP que necesita
    - ¿Qué hacer si no hay huecos con tamaño suficiente?
      - Ídem particiones fijas
      - Compactación


Operating System
128K
896K
(a)
Operating System
Process 1
320K
576K
(b)
Operating System
Process 1
320K
Process 2
224K
352K
(c)
Operating System
Process 1
320K
Process 2
224K
Process 3
288K
64K
(d)
Operating System
Process 1
320K
224K
Process 3
288K
64K
(e)
Operating System
Process 1
320K
Process 4
128K
96K
Process 3
288K
64K
(f)
Operating System
320K
Process 4
128K
96K
Process 3
288K
64K
(g)
Operating System
Process 2
224K
96K
Process 4
128K
96K
Process 3
288K
64K
(h)

Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Particiones variables *(continuación)*
  - **Estructuras de datos para la gestión de particiones**
    - *Lista de particiones y huecos*
      - Lista única de particiones y huecos o sólo de huecos
      - **Lista de particiones (no huecos) y lista de huecos**
    - Cada elemento de la lista de particiones contiene:
      - Dirección física de inicio; Tamaño en bytes; PID
    - Cada elemento de la lista de huecos contiene:
      - Dirección física de inicio; Tamaño en bytes

Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Particiones variables *(continuación)*
  - **Estrategias de asignación**
    - Igual que en el modelo anterior (particiones fijas)
  - **Protección y Traducción**
    - Igual que en el modelo anterior

Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Particiones variables *(continuación)*
  - **Ventajas e inconvenientes**
    - ✓ Mejora nivel máximo multiprogramación ☺ ☺
    - ✓ Tamaño máximo del proceso solo limitado por capacidad de la memoria ☺ ☺
    - ✓ Elimina la fragmentación interna ☺ ☹
    - × Más complejo que el anterior ☹

    (continúa)

Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Particiones variables *(continuación)*
  - **Ventajas e inconvenientes** *(continuación)*
    - Fragmentación externa ☹ ☹ ☹
      - Hay memoria libre suficiente (aunque no contigua) para cargar procesos
        - Espacio desaprovechado
      - Procesos entran y salen de memoria
        - Memoria libre se va fragmentando en huecos no contiguos de pequeño tamaño (similar a un disco duro)
      - **Soluciones**
        - *Condensación de huecos*: unión de huecos adyacentes al liberar un proceso →**solución parcial** de O(1)
        - *Compactación*: movimiento de procesos en la memoria para unir huecos dispersos y crear un único hueco de gran tamaño → **solución óptima** pero de alto coste computacional

Mecanismos de gestión de memoria real
Asignación contigua de memoria

- Particiones variables *(continuación)*
  - ¿Cómo afecta la estrategia de asignación a la fragmentación externa?
    - Primer ajuste ☺ ☹
    - Siguiente ajuste ☺ ☹
    - Mejor ajuste ☹
    - Peor ajuste ☺

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- El espacio de direcciones lógico de un proceso se reparte entre diferentes zonas (conjuntos de direcciones físicas) de la memoria principal
  - Será necesario organizar de alguna manera a los procesos → "Dividirlos en trozos"
  - Cada "trozo" del proceso podrá ir en cualquier lugar de la memoria y en cualquier orden
  - Complica la gestion de memoria al necesitar estructuras de datos que guarden la información anterior

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- De cada esquema tenemos que saber ahora MÁS COSAS
  - **Organización física de la memoria principal**
  - **Organización lógica de los procesos** *(NUEVO)*
    - Cómo se dividen los procesos en "trozos" y cómo se almacenan en la MP
  - **Estructuras de datos necesarias** para implementar lo anterior (la organización física de la MP)
  - **Estructuras de datos necesarias** para implementar lo anterior (la organización lógica de los procesos) *(NUEVO)*
  - **Estrategias de asignación**
  - **Organización de las direcciones físicas y de las direcciones lógicas** *(NUEVO)*
  - **Protección y traducción de direcciones lógicas a direcciones físicas**
  - **Ventajas e inconvenientes**

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

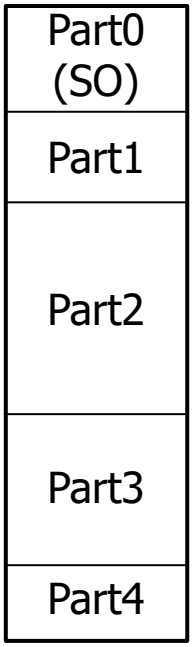
- Esquemas de memoria de asignación no contigua
  - ***Paginación simple***
    - Se basa en las particiones fijas
  - ***Segmentación simple***
    - Se basa en las particiones variables
  - ***Segmentación simple + paginación simple (segmentación paginada simple)***
    - Fusiona las dos anteriores para quedarse con lo mejor de cada una

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

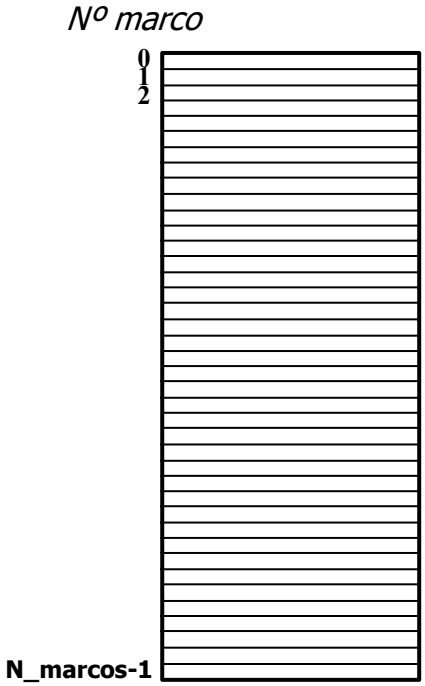
- Paginación simple
  - **Organización física de la memoria principal**
    - Al igual que particiones fijas divide a la MP en trozos pero éstos son **muy pequeños y de igual tamaño**
    - Estas divisiones reciben el nombre de ***marcos de página***
      - El tamaño viene impuesto por el *hardware*
      - Cada *marco de página* se numera comenzando por cero y en orden consecutivo → ***Número de marco de página***

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Organización física de la memoria principal** *(continuación)*



Memoria principal con Particiones Fijas



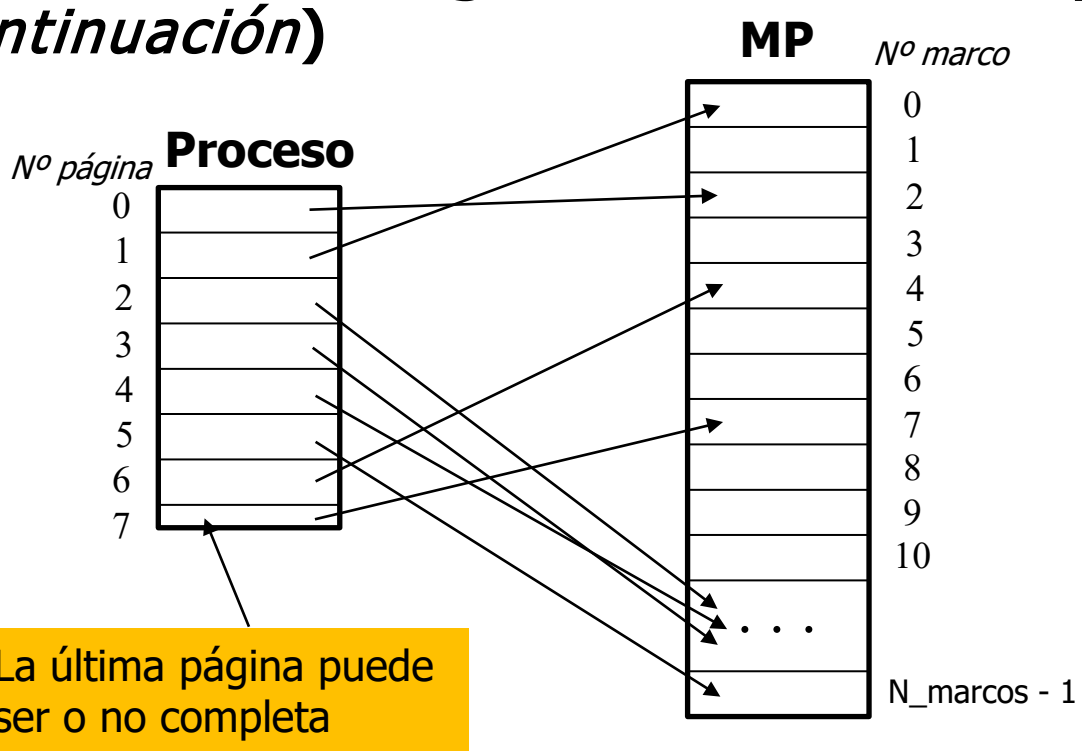
Memoria principal con Paginación simple

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Organización lógica de los procesos**
    - ¿Cómo dividir a los procesos para que se puedan almacenar en la MP?
    - Dividiendo los procesos en "trozos" de IDÉNTICO tamaño a los marcos de página
    - Estas divisiones reciben el nombre de ***páginas del proceso***
      - Cada *página* se numera comenzando por cero y en orden consecutivo → ***Número de página***

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Organización lógica de los procesos (*continuación*)**



Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Estructuras de datos para la organización física**
    - *Tabla de marcos de página*
      - Una única tabla para todo el sistema
      - Con tantas entradas como marcos de memoria física
      - En cada entrada se almacena:
        - Flag indicando si el marco está libre o asignado
        - En el caso de estar asignado el número de página y el PID del proceso al que pertenece
    - Los marcos libres podrían estar organizados en una lista aparte → *Lista de marcos libres*
      - Para acelerar las operaciones de asignación

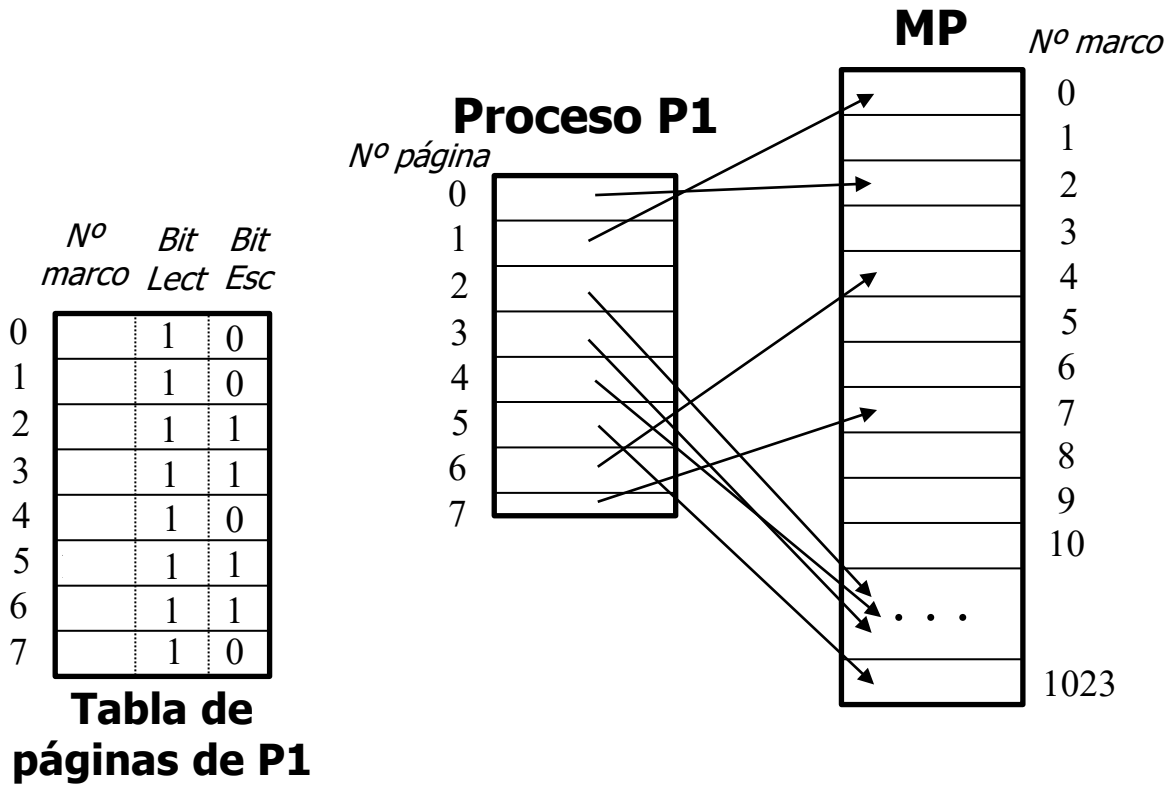
Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Estructuras de datos para la organización lógica**
    - *Tabla de páginas*
      - **Una tabla por proceso**, apuntada/contenida en su PCB
      - Con tantas entradas como páginas tenga el proceso
        - Tablas estáticas o dinámicas
      - En cada entrada se almacena
        - El **número de** ***marco de página*** donde se guarda la *página* del proceso
        - **Bits de protección** (lectura, escritura…)
          - Permiten proteger la página de ciertas operaciones

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Estructuras de datos (*continuación*)**

Falta un dato. ¡Añádelo!



	Nº marco	Bit Lect	Bit Esc
0		1	0
1		1	0
2		1	1
3		1	1
4		1	0
5		1	1
6		1	1
7		1	0

Tabla de páginas de P1

	¿Libre?	Nº pág	PID
0	**0**	**1**	**1**
1	1		
2	**0**	**0**	**1**
3	0	23	2
4	**0**	**6**	**1**
5	1		
6	0	0	2
7	**0**	**7**	**1**
27	**0**	**2**	**1**
28	1		
29	0	7	3
30	**0**		**1**
132	**0**	**3**	**1**
1023	**0**	**5**	**1**

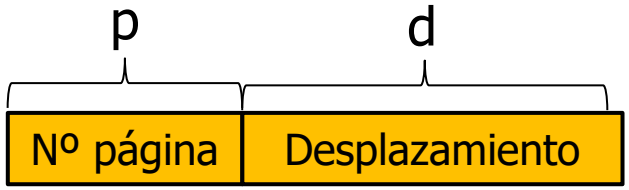
Tabla de marcos

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Estrategias de asignación**
    - Igual que en esquemas/modelos anteriores
    - Nótese que se puede aplicar cualquiera sin afectar al rendimiento de la paginación

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Organización de las direcciones lógicas (L)**

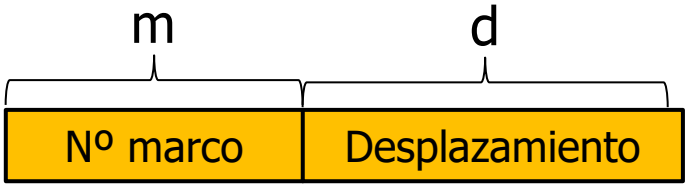


Direcciones lógicas de L bits

- **p** → Nº bits reservados para páginas de los procesos
- 2^p → Máximo de páginas que puede tener un proceso
  - El proceso puede tener menos páginas
- **d** → Nº bits necesarios para almacenar el máximo desplazamiento de una página
- 2^d → Tamaño de las páginas
- L=p+d → Tamaño de las direcciones lógicas en bits

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Organización de las direcciones físicas (R)**

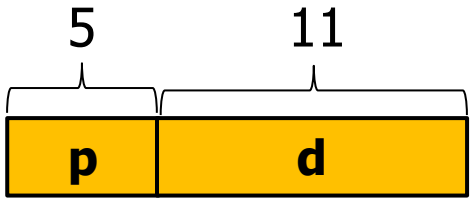


Direcciones físicas de R bits

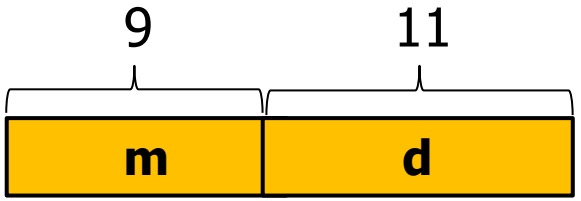
  - **m** → Nº bits reservados para marcos de la MP
  - 2^m → Número total de marcos de página
  - **d** → Nº bits necesarios para almacenar el máximo desplazamiento de un marco
    - ¡¡COINCIDE SIEMPRE CON LA **d** DE LA TRANSPARENCIA ANTERIOR!!
  - 2^d → Tamaño de los marcos (y por tanto de las páginas)
  - R=m+d → Tamaño de las direcciones físicas en bits

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - Ejemplo de organización de direcciones
    - Un sistema informático tiene un bus de direcciones de 20 bits y el máximo tamaño de un proceso es 65536 bytes. Las tablas de páginas tienen como máximo 32 entradas.
    - ¿Organización de las direcciones lógicas y físicas?



Direcciones lógicas de 16 bits (L)



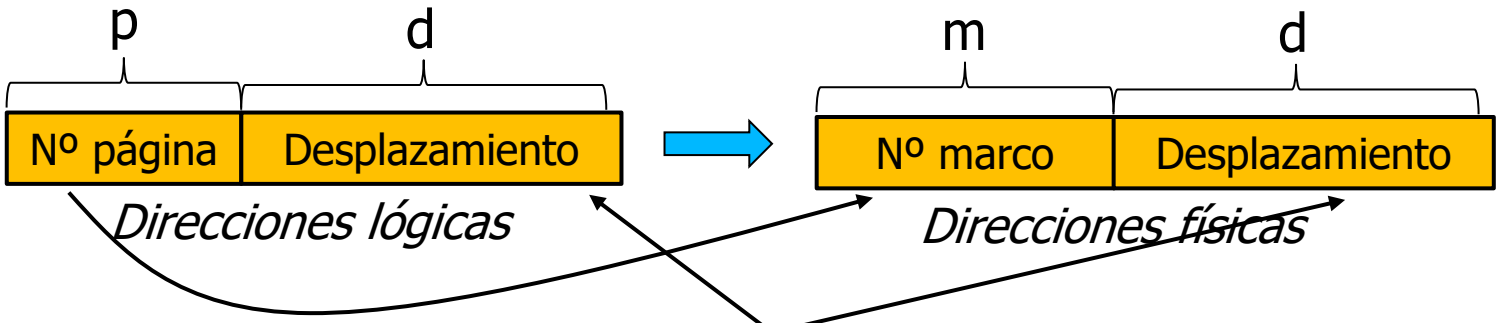
Direcciones físicas de 20 bits (R)

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Protección**
    - ¡¡Igual que en Particiones fijas y Particiones variables!!
    - Hay que tener en cuenta además los bits de protección de la página → Se produce una excepción NUEVA en la MMU si se intenta saltar dicha protección (lectura/escritura)

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Traducción de direcciones lógicas a físicas**



    - Desplazamientos (d) coinciden (tamaño marco = tamaño página) → Esos bits son exactamente iguales en ambas direcciones
    - Solo será necesario encontrar el Nº marco donde está almacenada el Nº de página de la dirección lógica

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

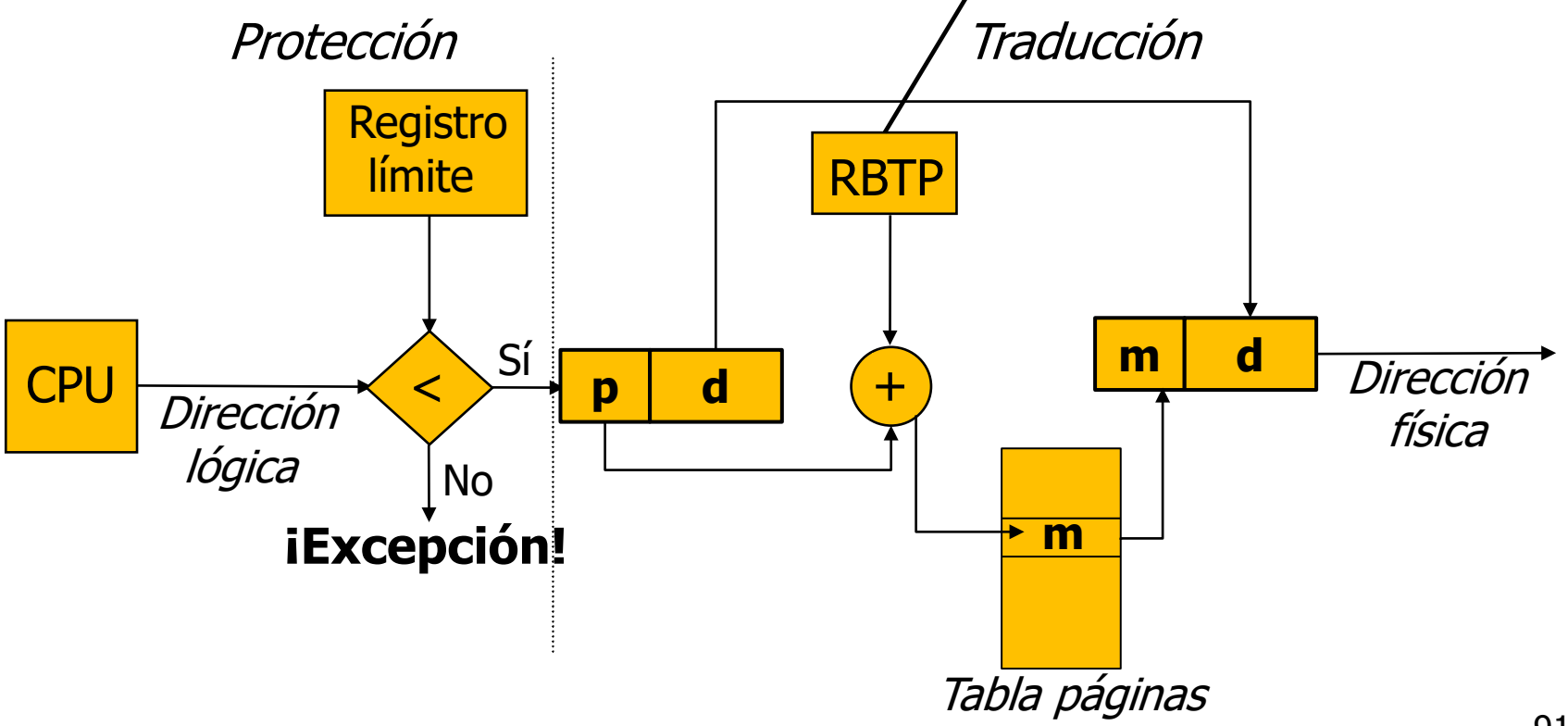
- Paginación simple *(continuación)*
  - **Traducción de direcciones lógicas a físicas** *(continuación)*
    - **Pasos a realizar**
      1. Pasar a binario (si no lo está ya) la dirección lógica **y rellenar con ceros a la izquierda hasta que tenga p+d bits**
      2. Extraer los **p** bits más significativos que contendrán el nº de página de la dirección lógica
      3. Apuntar aparte los **d** bits menos significativos que quedan
      4. Pasar a decimal los **p** bits extraidos en el paso 2 (nº de página en decimal)

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Traducción de direcciones lógicas a físicas** *(continuación)*
    - **Pasos a realizar** *(continuación)*
      5. Acceder a la entrada **p** de la tabla de páginas del proceso para extraer el Nº de marco donde está almacenada la página
      6. Pasar el Nº de marco a binario **y rellenar con ceros a la izquierda hasta que tenga m bits**
      7. Añadir a la derecha de los m bits obtenidos en el paso anterior los **d** bits apuntados en el paso 3 → **¡¡Dirección física final en binario!!**

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - Diseño de la MMU

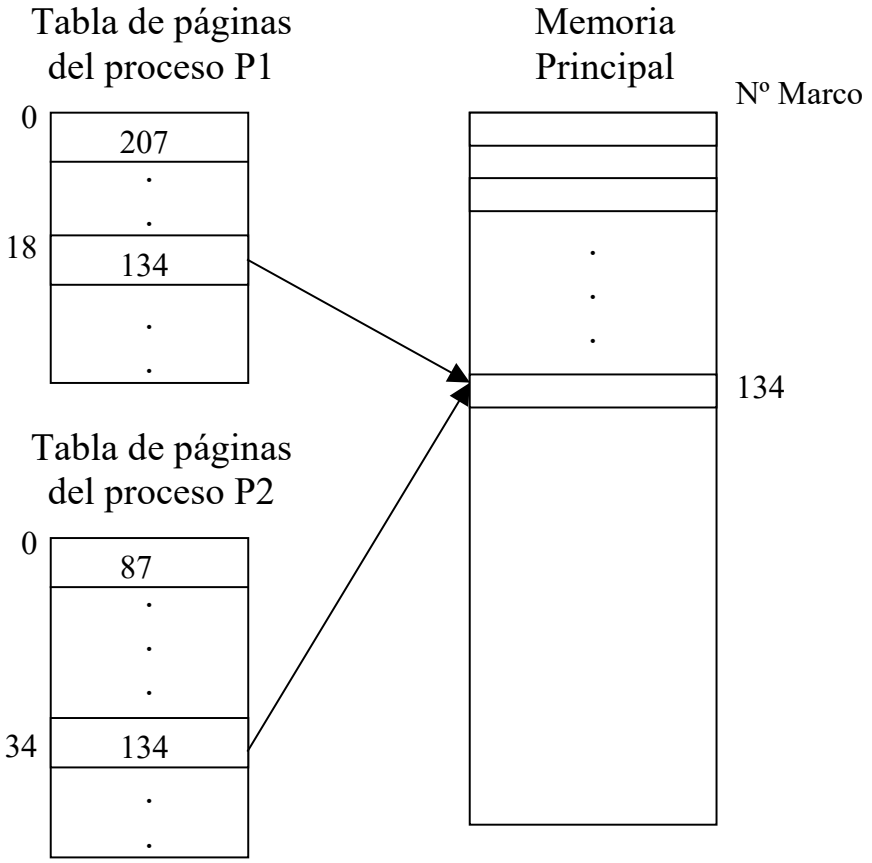


Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Soporte hardware de las tablas de páginas**
    - La tabla de páginas se almacena en memoria principal
    - Un registro de la MMU (RBTP) permite localizar la tabla de páginas del proceso en ejecución
    - Nótese que para acceder a una dirección se necesitan dos accesos a la memoria principal
      1. Acceso a la entrada **p** de la tabla de páginas para obtener **m**
      2. Acceso a la dirección física solicitada
    - ¿Cuando se cambia el valor del registro RBTP?
      - Con cada cambio de proceso
      - El valor del RBTP de cada proceso se almacena en su PCB → Operación de restaurar contexto

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Compartición de memoria entre procesos**
    - Es necesario a veces compartir datos entre procesos
      - Ver tema 3
    - Página como unidad de compartición
    - Varias páginas apuntan al mismo marco



Mecanismos de gestión de memoria real
Asignación no contigua de memoria

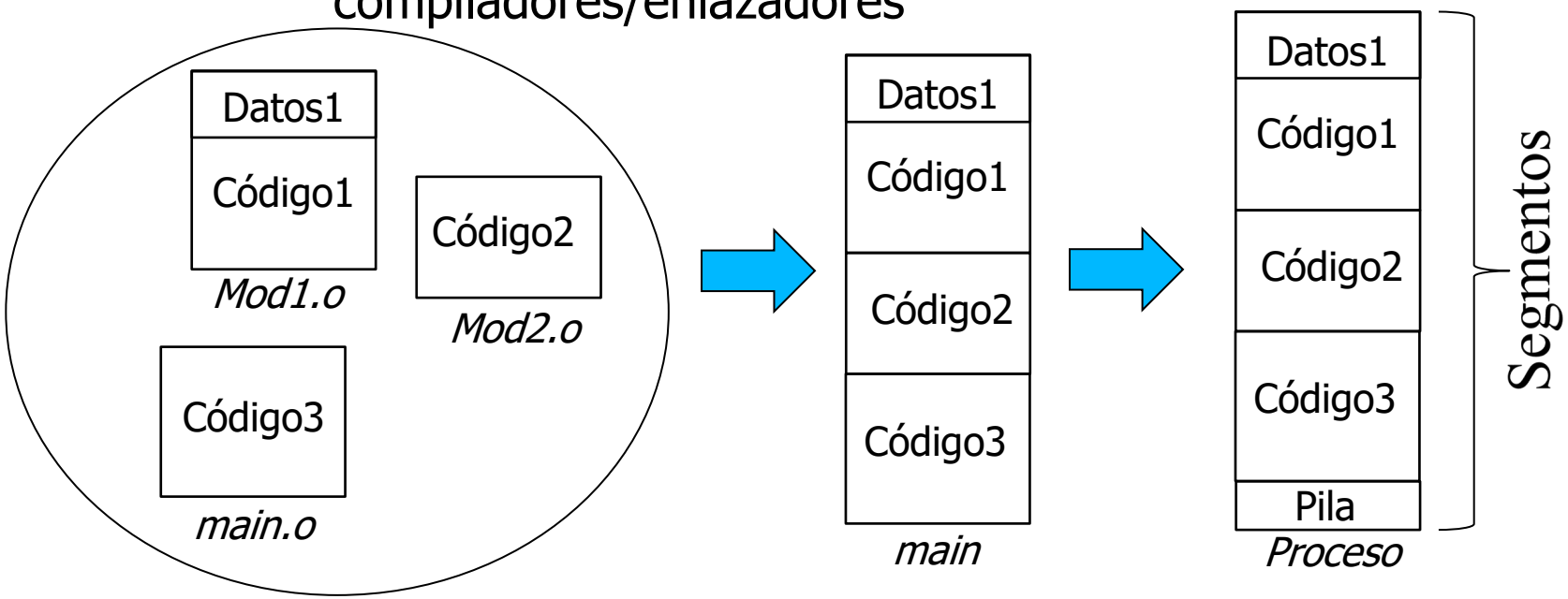
- Paginación simple *(continuación)*
  - **Ventajas e inconvenientes**
    - ✓ La paginación no es visible al usuario del S.O. ☺☺
    - ✓ Se elimina la fragmentación externa ☺☺☺
    - ✓ La fragmentación interna sólo se puede producir en la última página de cada proceso ☹ ☺
    - ✓ Es fácil permitir que procesos compartan memoria ☺
    - ✓ Se pueden proteger las páginas (bits de protección) ☺
    - Si las páginas son pequeñas:
      - ✓ Reducen la fragmentación interna ☺
      - × Aumentan tamaño de tabla de páginas ☹

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Paginación simple *(continuación)*
  - **Ventajas e inconvenientes** *(continuación)*
    - × La manera de dividir a los procesos es MUY artificial ☹☹☹
      - Los compiladores/enlazadores ya dividen a los programas de una manera más natural (Datos+Código)
      - Los procesos heredan dicha estructura a la que se añade la pila (Datos+Código+Pila)
    - × Lo anterior provoca que la memoria no se pueda compartir/proteger de manera razonable y práctica ☹☹
      - Solo se pueden compartir/proteger páginas completas
      - ¿Qué pasa si una página tiene parte de datos y parte de código?

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación simple
  - **Organización lógica de los procesos**
    - Los procesos se van a dividir de forma "natural" tal y como ya los dividen los propios compiladores/enlazadores



Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación simple *(continuación)*
  - **Organización lógica de los procesos** ***(continuación)***
    - División del espacio de direcciones de un proceso en segmentos (de distinto tamaño)
      - Visión del proceso igual que la de un usuario
        - Segmentos de texto, datos y pila
    - Cada segmento utiliza una zona de memoria contigua
    - Sus segmentos pueden encontrarse dispersos en MP y en cualquier orden
    - Los segmentos se numeran comenzando por cero y de forma consecutiva (s0; s1, s2, …)

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación simple *(continuación)*
  - **Organización física de la memoria principal**
    - No se puede dividir la MP como en paginación porque los segmentos son de distinto tamaño
    - **Solución**: Organizar la MP como en particiones variables
    - En los huecos disponibles se cargarán los segmentos del proceso

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación simple *(continuación)*
  - **Estructuras de datos para la organización física**
    - Igual que en particiones variables
      - *Listas (o lista única) de particiones y huecos*

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

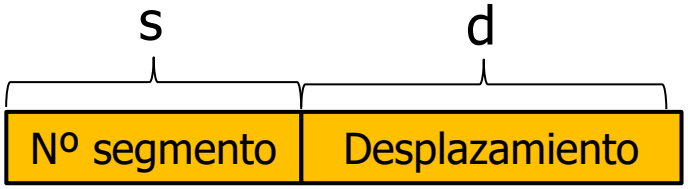
- Segmentación simple *(continuación)*
  - **Estructuras de datos para la organización lógica**
    - *Tabla de segmentos*
      - **Una tabla por proceso**, apuntada/contenida en su PCB
      - Con tantas entradas como segmentos tenga el proceso
        - Tablas estáticas o dinámicas
      - En cada entrada se almacena
        - **Dirección base del segmento** (Dirección física de inicio donde está cargado el segmento en MP)
        - **Límite del segmento** (tamaño en bytes)
        - **Bits de protección** (lectura, escritura…)
          - Permiten proteger el segmento de ciertas operaciones

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación simple *(continuación)*
  - **Estrategias de asignación**
    - Igual que en esquemas/modelos anteriores
    - Nótese que las estrategias afectan a la segmentación igual que lo hacían en particiones variables

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación simple *(continuación)*
  - **Organización de las direcciones lógicas (L)**



Direcciones lógicas de L bits

    - **s** → Nº bits reservados para segmentos de los procesos
    - 2^s → Máximo de segmentos que puede tener un proceso
      - El proceso puede tener menos segmentos
    - **d** → Nº bits necesarios para almacenar el máximo desplazamiento de un segmento
    - 2^d → Tamaño máximo de los segmentos
      - El segmento puede tener un tamaño inferior
    - L=s+d → Tamaño de las direcciones lógicas en bits

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

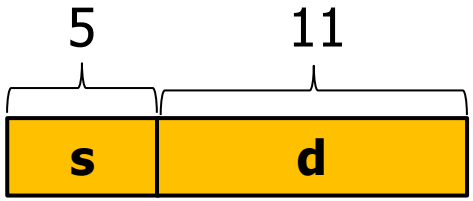
- Segmentación simple *(continuación)*
  - **Organización de las direcciones físicas (R)**

Direcciones físicas de R bits

  - **Las direcciones físicas no se organizan de ninguna manera**

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación simple *(continuación)*
  - Ejemplo de organización de direcciones
    - Un sistema informático tiene un bus de direcciones de 20 bits y el máximo tamaño de un proceso es 65536 bytes. Las tablas de segmentos tienen como máximo 32 entradas.
    - ¿Organización de las direcciones lógicas y físicas?

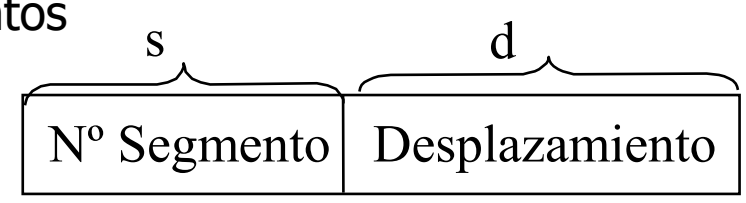


Direcciones lógicas de 16 bits (L)

Direcciones físicas de 20 bits (R)

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

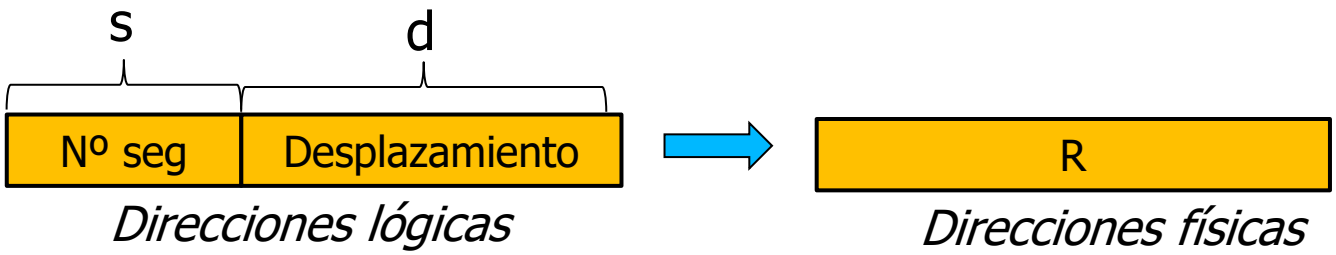
- Segmentación simple *(continuación)*
  - **Protección**
    - **No se hace igual que en el resto de esquemas**
      - No vale con comparar si *dir_logica < tamaño_proceso*
      - En segmentación hay direcciones lógicas válidas que superan el tamaño del proceso
      - Se tiene que comprobar que ***d < tamaño del segmento*** y que ***s < nº segmentos del proceso***
      - El tamaño del segmento se obtiene de la tabla de segmentos



Direcciones Lógicas de s+d bits

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación simple *(continuación)*
  - **Traducción de direcciones lógicas a físicas**



Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación simple *(continuación)*
  - **Traducción de direcciones lógicas a físicas** *(continuación)*
    - **Pasos a realizar**
      1. Pasar a binario (si no lo está ya) la dirección lógica **y rellenar con ceros a la izquierda hasta que tenga s+d bits**
      2. Extraer los **s** bits más significativos que contendrán el nº de segmento de la dirección lógica
      3. Pasar a decimal los **s** bits extraidos en el paso 2 (nº de segmento en decimal)

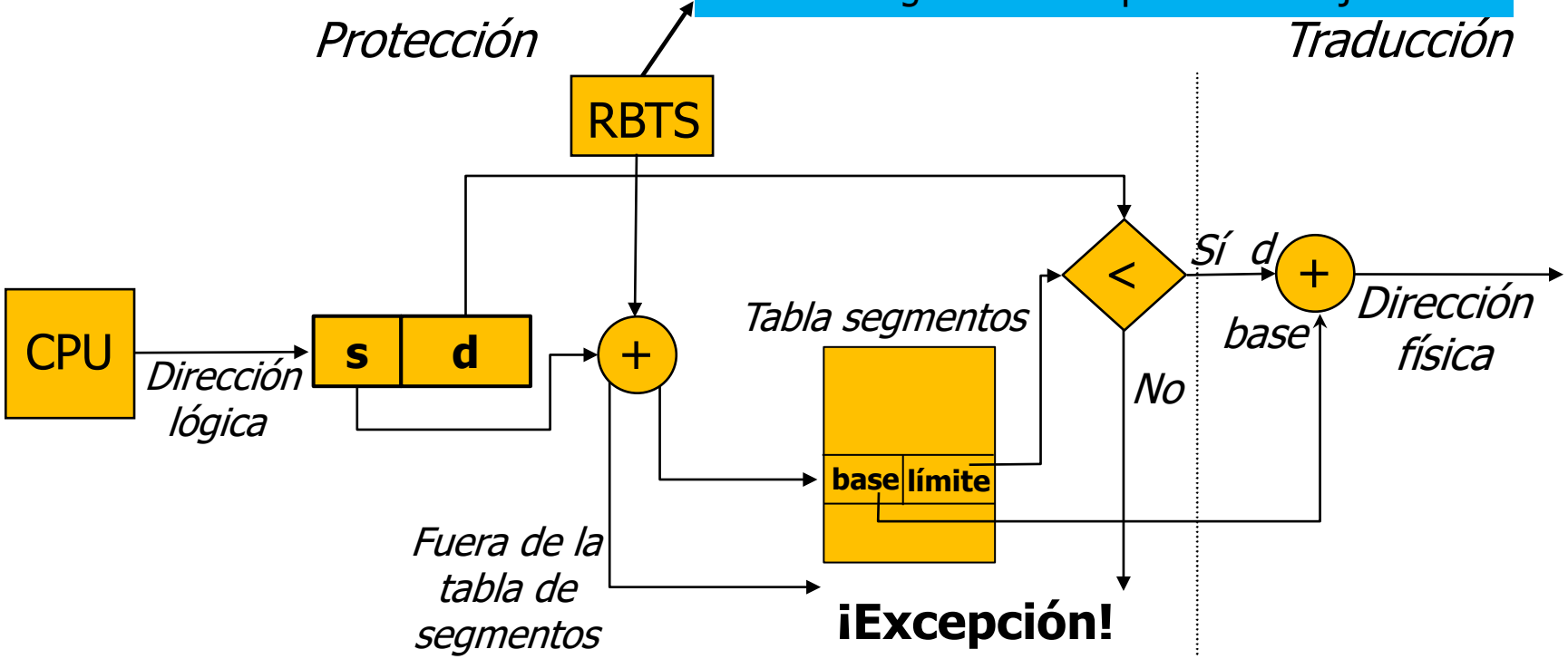
Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación simple *(continuación)*
  - **Traducción de direcciones lógicas a físicas** *(continuación)*
    - **Pasos a realizar** *(continuación)*
      4. Acceder a la entrada correspondiente de la tabla de segmentos del proceso para extraer la dirección física donde está cargado el segmento
      5. Extraer los **d** bits menos significativos que contendrán el desplazamiento dentro del segmento
      6. Pasar a decimal los **d** bits extraidos en el paso anterior
      7. Sumar a la dirección física obtenida en el paso 4 el desplazamiento obtenido en el paso 6 y se obtendrá la dirección física final en decimal

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación simple *(continuación)*
  - Diseño de la MMU

Contiene la dirección FÍSICA donde está la tabla de segmentos del proceso en ejecución

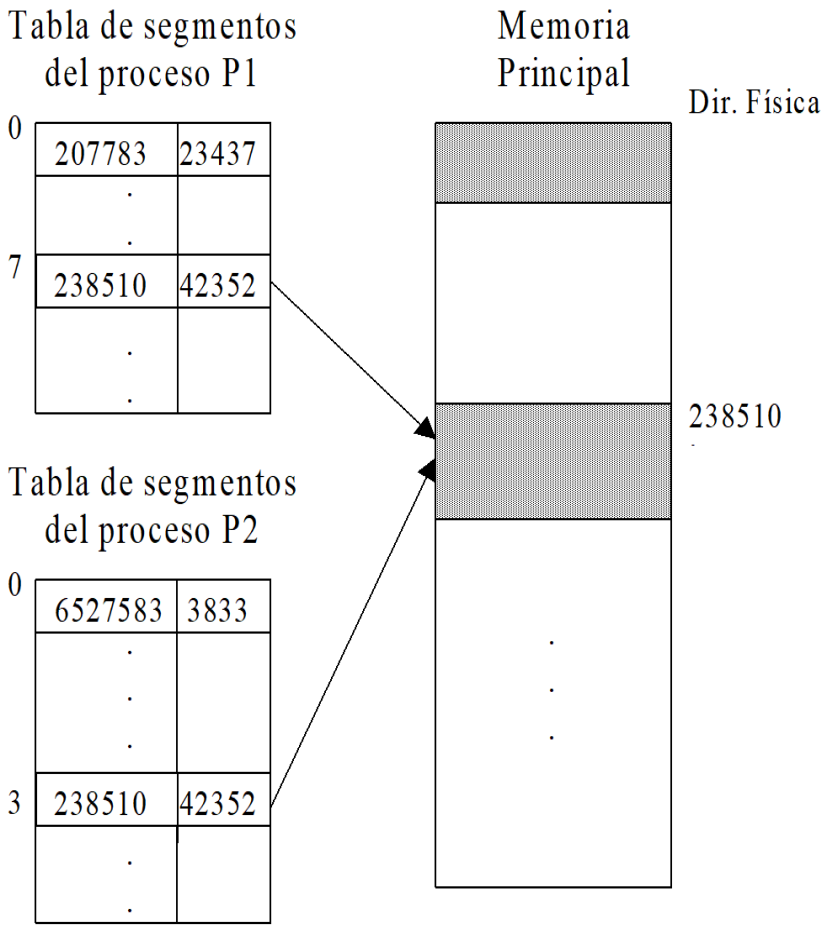


Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación simple *(continuación)*
  - **Soporte hardware de las tablas de segmentos**
    - Idéntico a la de las tablas de páginas (transparencia 89)

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación simple *(continuación)*
  - Compartición de memoria entre procesos
    - Es necesario a veces compartir datos entre procesos
    - Segmento como unidad de compartición
    - Varios segmentos tienen la misma base
    - Ejemplo: dos procesos compartiendo



Mecanismos de gestión de memoria real
Asignación no contigua de memoria

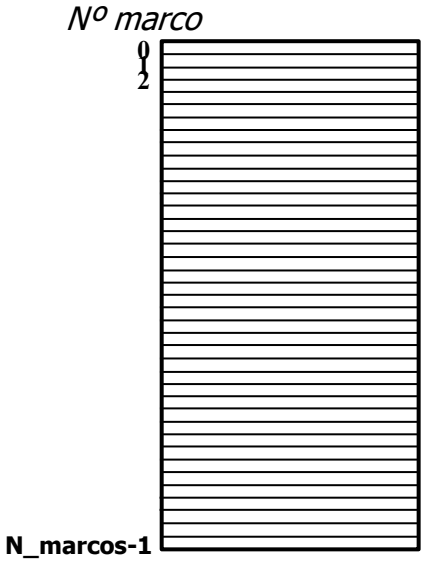
- Segmentación simple *(continuación)*
  - Ventajas e inconvenientes
    - ✓ Facilita la compartición ☺☺
      - Se comparten unidades lógicas o segmentos (datos, texto)
    - ✓ Se puede proteger solo datos o solo código (bits de protección) ☺ ☺
    - ✓ Facilita la ampliación de las estructuras de datos ☺
      - Sólo se ampliaría el segmento correspondiente
    - ✓ Visión del proceso tal y como lo ve el usuario ☺
    - × Fragmentación externa ☹ ☹ ☹
      - Se pueden aplicar las mismas soluciones vistas en Particiones Variables

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple
  - Combina las ventajas de Paginación Simple y Segmentación simple
    - La paginación simple es buena organizando la memoria principal → No hay fragmentación externa
    - La segmentación simple es buena organizando los procesos → Fácil compartir y proteger memoria
  - Toma la ***organización física de la memoria principal*** de la *paginación* y la ***organización lógica de los procesos*** de la *segmentación*

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

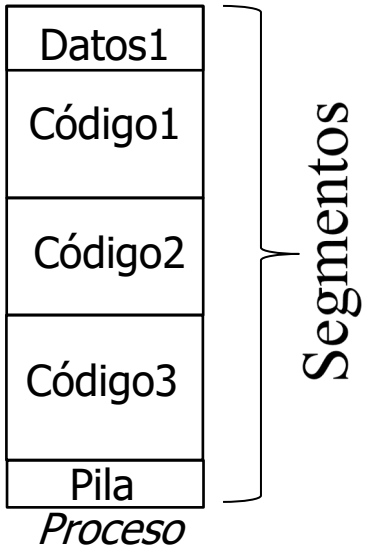
- Segmentación paginada simple *(continuación)*
  - **Organización física de la memoria principal**
    - Al igual que *paginación* divide a la MP en ***marcos de página***



Memoria principal con Segmentación página simple

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Organización lógica de los procesos**
    - Los procesos se van a dividir en "trozos" en DOS fases.
      1. En la primera fase se dividen como lo hacía la *Segmentación → En segmentos*

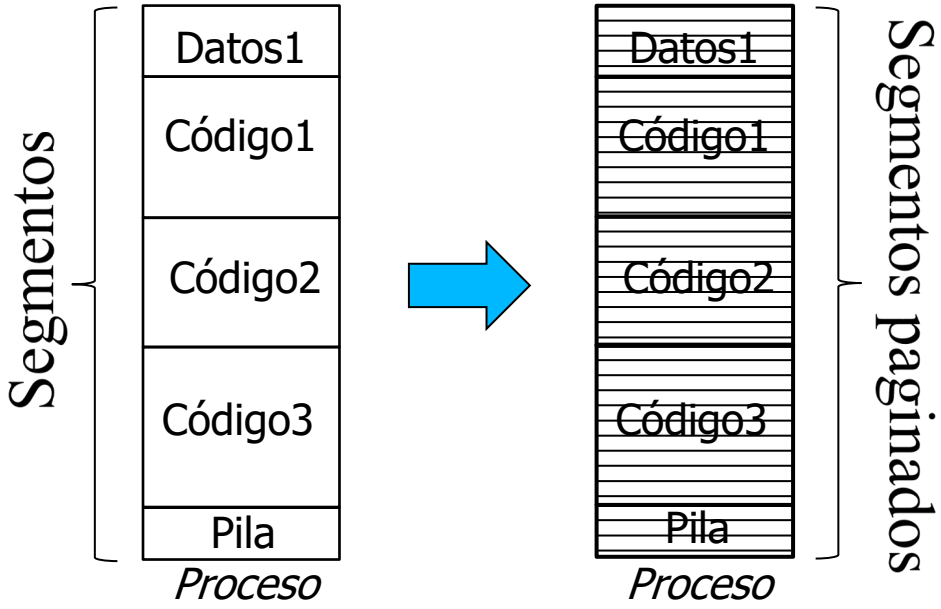


Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Organización lógica de los procesos** ***(continuación)***
    - Los *segmentos* obtenidos en la primera fase NO se pueden cargar en la MP porque está dividida en *marcos de página* → Necesitamos que los procesos tengan páginas.
    - **SOLUCIÓN**: Aplicar una segunda division:
      2. En la segunda fase se divide ahora cada *segmento* en ***páginas***

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Organización lógica de los procesos** *(continuación)*



Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Estructuras de datos para la organización física**
    - Igual que en *paginación*
      - *Tabla de marcos de página*

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

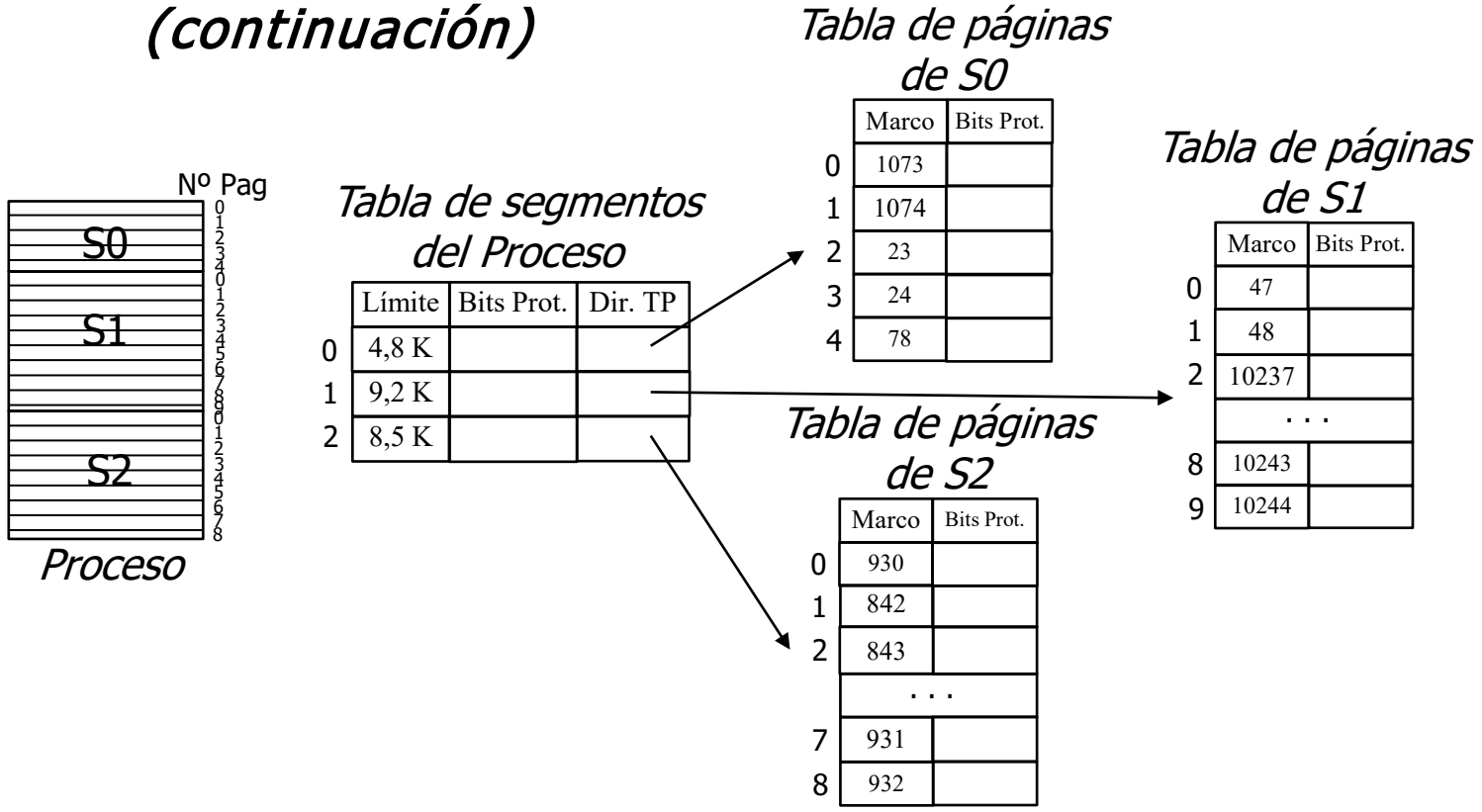
- Segmentación paginada simple *(continuación)*
  - **Estructuras de datos para la organización lógica**
    - *Tabla de segmentos*
      - **Una tabla por proceso**, apuntada/contenida en su PCB
      - Con tantas entradas como *segmentos* tenga el proceso
      - En cada entrada se almacena
        - **Dirección base de la tabla de páginas del segmento**
        - **Límite del segmento** (tamaño en bytes)
        - **Bits de protección** (lectura, escritura…)
          - Permiten proteger el segmento de ciertas operaciones

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Estructuras de datos para la organización lógica** *(continuación)*
    - *Tabla de páginas*
      - **Una tabla por cada segmento**
      - Con tantas entradas como *páginas* tenga el *segmento*
      - En cada entrada se almacena
        - El **número de** ***marco de página*** donde se guarda la *página* del segmento del proceso
        - **Bits de protección** (lectura, escritura…)
          - Permiten proteger la página de ciertas operaciones

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Estructuras de datos para la organización lógica** ***(continuación)***

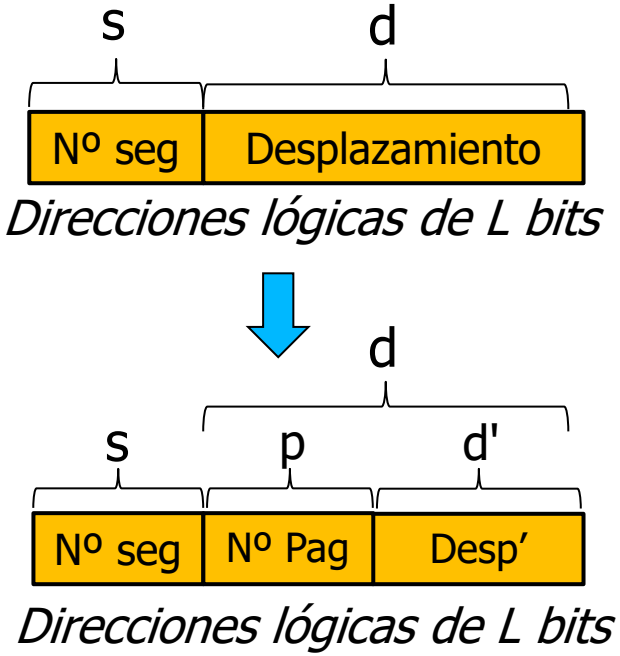


Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Estrategias de asignación**
    - Igual que en páginación → No afecta en nada

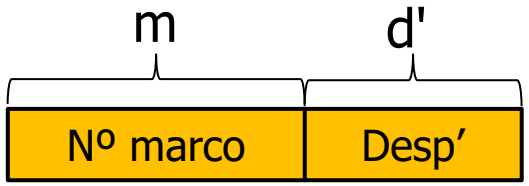
Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Organización de las direcciones lógicas (L)**



Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Organización de las direcciones físicas (R)**

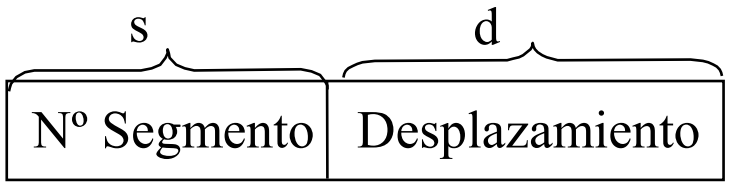


Direcciones físicas de R bits

  - Nótese que las direcciónes físicas trabajan con d' y no con d

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

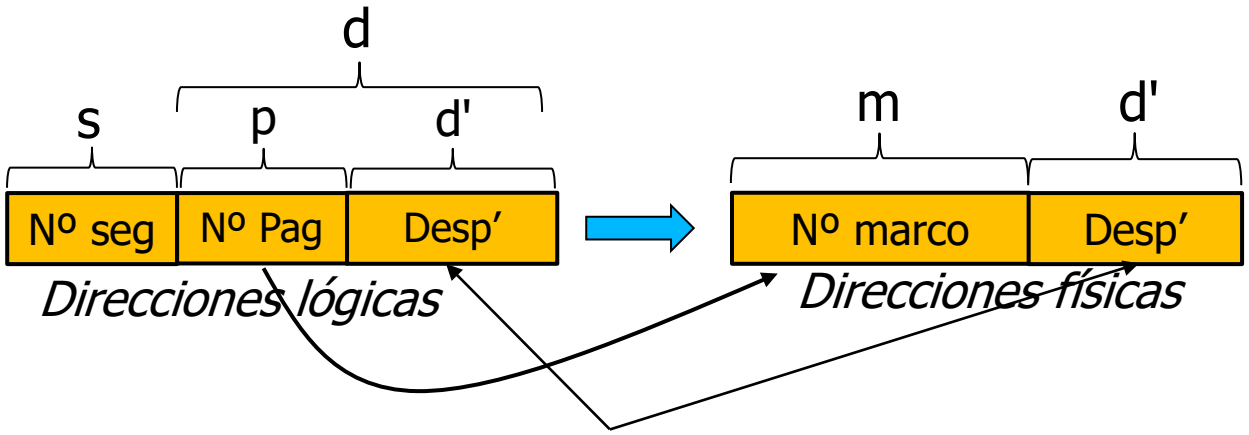
- Segmentación paginada simple *(continuación)*
  - **Protección**
    - **Se hace igual que en *Segmentación***
      - No vale con comparar si *dir_logica < tamaño_proceso*
      - Se tiene que comprobar que ***d < tamaño del segmento*** y que ***s < nº segmentos del proceso***
      - La comparación anterior se hace con d NO CON d'
      - El tamaño del segmento se obtiene de la tabla de segmentos



Direcciones Lógicas de s+d bits

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Traducción de direcciones lógicas a físicas**



    - Desplazamientos (d') coinciden → Esos bits son exactamente iguales en ambas direcciones
    - Solo será necesario encontrar el Nº marco donde está almacenada el Nº de página de la dirección lógica
      - Usando la tabla de páginas del segmento **s**

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Traducción de direcciones lógicas a físicas** *(continuación)*
    - **Pasos a realizar**
      1. Pasar a binario (si no lo está ya) la dirección lógica **y rellenar con ceros a la izquierda hasta que tenga s+d bits**
      2. Extraer los **s** bits más significativos que contendrán el nº de segmento de la dirección lógica
      3. Pasar a decimal los **s** bits extraidos en el paso anterior (nº de segmento en decimal)

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Traducción de direcciones lógicas a físicas** *(continuación)*
    - **Pasos a realizar** *(continuación)*
      4. Seleccionar, para su posterior uso, la tabla de páginas del segmento **s** extraido en el paso anterior
      5. Extraer los **d** bits menos significativos que contendrán el desplazamiento dentro del segmento
      6. Con los bits extraidos en el paso anterior extraer ahora los **p** bits más significativos, que contendrán el nº de página del segmento **s**
      7. Apuntar aparte los **d'** bits menos significativos que quedan

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

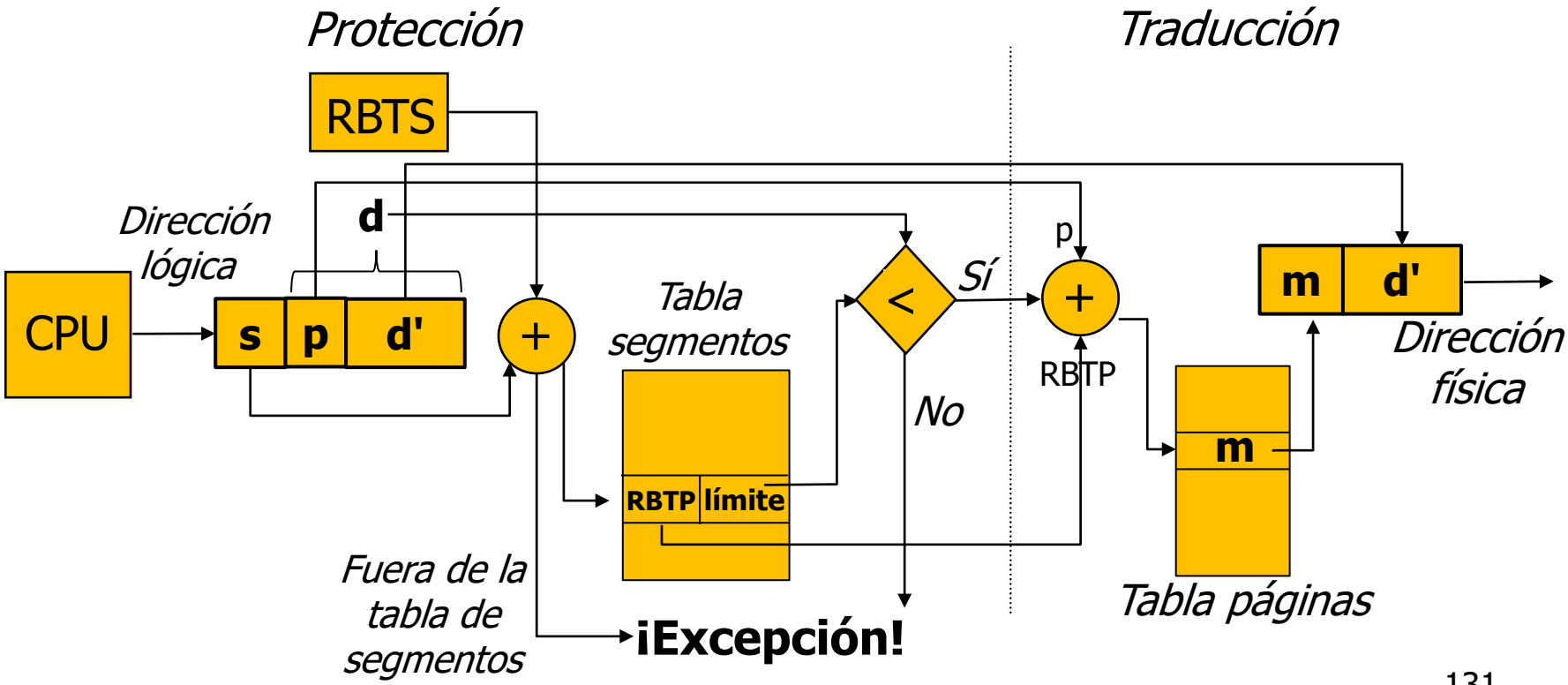
- Segmentación paginada simple *(continuación)*
  - **Traducción de direcciones lógicas a físicas** *(continuación)*
    - **Pasos a realizar** *(continuación)*
      8. Pasar a decimal los **p** bits extraidos en el paso 6 (nº de página en decimal)
      9. En la tabla de páginas extraida en el paso 4, acceder a la entrada **p** para extraer el Nº de marco donde está almacenada la página
      10. Pasar el Nº de marco a binario **y rellenar con ceros a la izquierda hasta que tenga m bits**
      11. Añadir a la derecha de los **m** bits obtenidos en el paso anterior los **d'** bits apuntados en el paso 7 → **¡¡Dirección física final en binario!!**

Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Compartición de memoria entre procesos**
    - Se pueden compartir segmentos y se pueden compartir páginas indistintamente
    - ¿Cuál es más apropiado de los dos?
      - Compartir segmentos → Razonar por qué

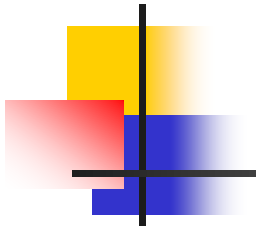
Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - Diseño de la MMU



Mecanismos de gestión de memoria real
Asignación no contigua de memoria

- Segmentación paginada simple *(continuación)*
  - **Ventajas e inconvenientes**
    - ✓ Facilita la compartición ☺☺
      - Se comparten unidades lógicas o segmentos (datos, texto)
    - ✓ Se puede proteger solo datos o solo código (bits de protección) ☺ ☺
    - ✓ Facilita la ampliación de las estructuras de datos ☺
      - Sólo se ampliaría el segmento correspondiente
    - ✓ Visión del proceso tal y como lo ve el usuario ☺
    - × Solo hay fragmentación interna en la última página de cada segmento ☹ ☺
    - × Relativamente complejo ☹



Lecturas recomendadas

- Stallings, “Sistemas Operativos”, 5ª edición
  - Capítulo 7, “Gestión de memoria” (también apéndice 7A)
- Silberschatz, “Fundamentos de Sistemas Operativos”, 7ª edición
  - Capítulo 8, “Memoria principal”