Gestión de Memoria

(Cap. 6 de Stallings)

SOyD

Que veremos ??

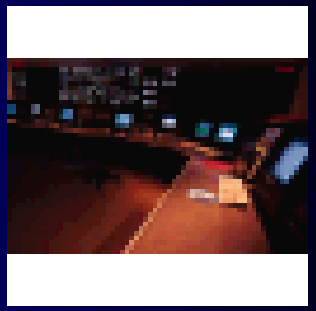
- Definiciones básicas

- Requisitos de la gestión de memoria

- Cargas de programas en memoria principal
  (partición , paginación, segmentación, etc.)

- Memoria Virtual

Definiciones



Memoria Principal

– Área de almacenamiento dividido en unidades a las que se puede referenciar a través de una dirección.
– Recurso básico: Para que un programa se ejecute debe encontrarse en memoria principal, al menos, una parte

Gestor de Memoria

– Componente del Sistema Operativo que se encarga de las tareas relacionadas con la administración de la Memoria Principal:
- Asignación de Memoria Principal a los procesos que la solicitan
- Localización de espacios libres, y ocupados
- Aprovechamiento máximo de dicha memoria

- La memoria es una amplia tabla de datos, cada uno de los cuales con su propia dirección

- Tanto el tamaño de la tabla, como el de los datos incluidos en ella dependen de cada arquitectura concreta

- Para que los programas puedan ser ejecutados es necesario que estén cargados en memoria principal

- La información que es necesario almacenar de modo permanente se guarda en dispositivos de almacenamiento secundarios también conocidos como memoria secundaria

En un Sistema operativo monoprogramado, la memoria principal compartida por el sistema operativo y el programa de usuario que se ejecuta en ese instante.

En un Sistema operativo multiprogramado es:

- necesario subdividir aún más la memoria principal para dar cabida a varios procesos de usuario

- vital una gestión efectiva de la memoria, dado que si caben pocos procesos de usuario es posible encontrarlos bloqueados en una E/S simultáneamente, y el procesador estará desocupado. Por ello, hace falta repartir eficientemente la memoria para meter tanto procesos como sea posible.

Requisitos de un sistema Gestión de Memoria

Al realizar un estudio de los diversos mecanismos y políticas relacionadas con la gestión de memoria, vale la pena tener en mente los requisitos que se intentan satisfacer.

Hay 5 requisitos:

- **Reubicación**
- **Protección**
- **Compartición**
- **Organización Lógica**
- **Organización Física**

1) Reubicación

– El programador desconoce dónde se ubicará el programa en memoria principal y que otros programas residirán en memoria en el momento de la ejecución del programa. El programador debe trabajar al margen de la localización de su código en memoria.

– El gestor de memoria (HW+SW) debe establecer la correspondencia de las direcciones de memoria a las que hace referencia las instrucciones de un programa para que se correspondan, en cada ocasión, con las direcciones de memoria principal asignadas al mismo.

....

Durante su ejecución, el proceso puede ser enviado a disco y cuando vuelva a ser cargado, debe situarse en la misma región de memoria principal que antes, esto es reubicarlo (Re ubicación).

De este modo, se sabe antes de tiempo donde debe situarse un programa y hay que permitir que el programa pueda moverse en memoria principal como resultado de un intercambio.

Esto plantea asuntos técnicos relativos al direccionamiento.

Veamos la sig. imagen, que representa a un proceso en memoria.

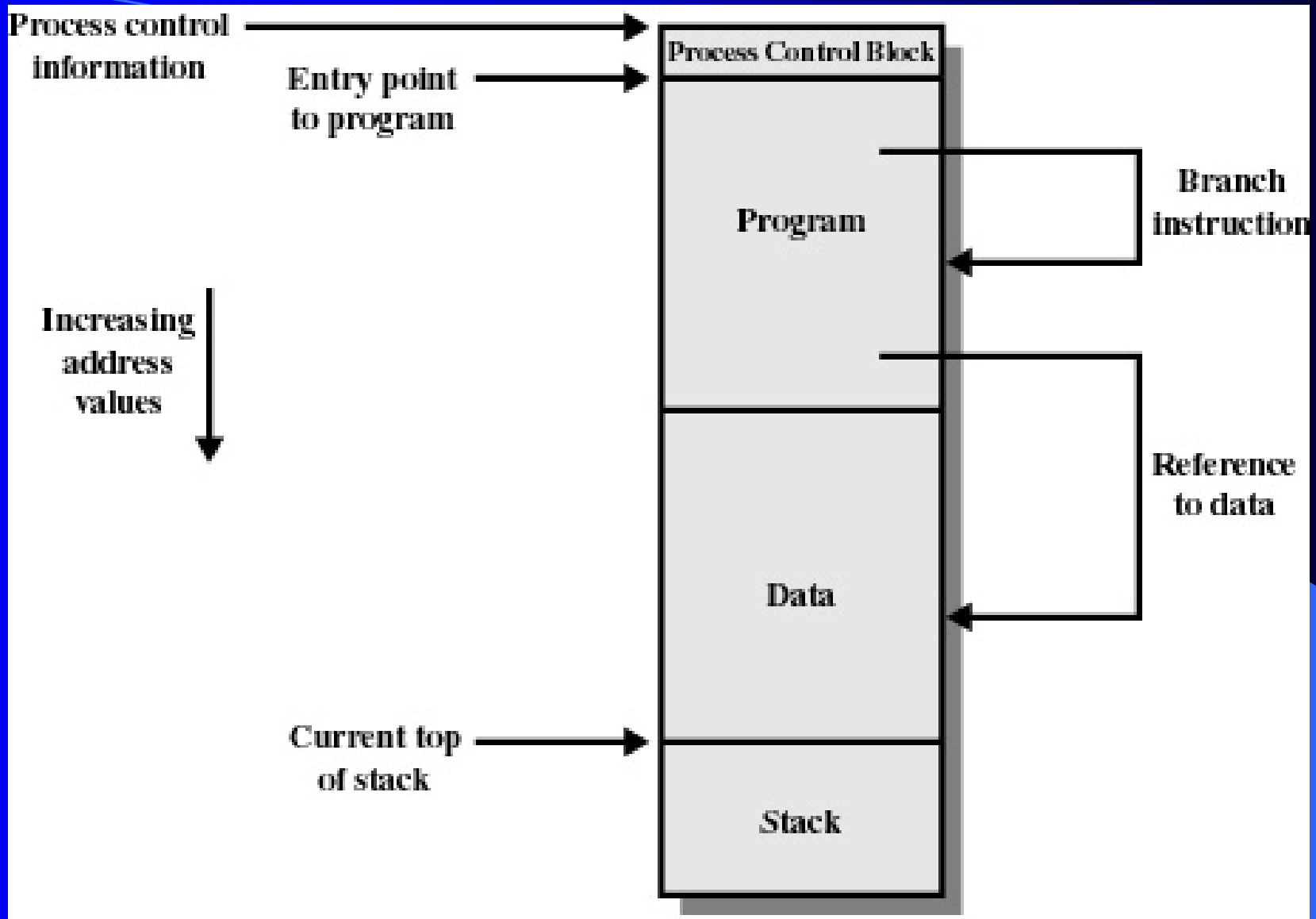


Figure 7.1 Addressing Requirements for a Process

- - - -

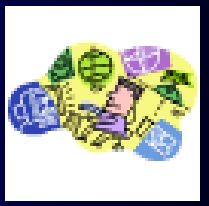
El SO tiene que conocer la ubicación de la información de control del proceso y de la pila de ejecución, así como el punto de partida para comenzar la ejecución del programa para dicho proceso.

El procesador debe ocuparse de las referencias a memoria dentro del programa.

Las instrucciones de bifurcación deben contener la dirección que haga referencia a la instrucción que se vaya a ejecutar a continuación.

Idem las instrucciones de referencia de datos.

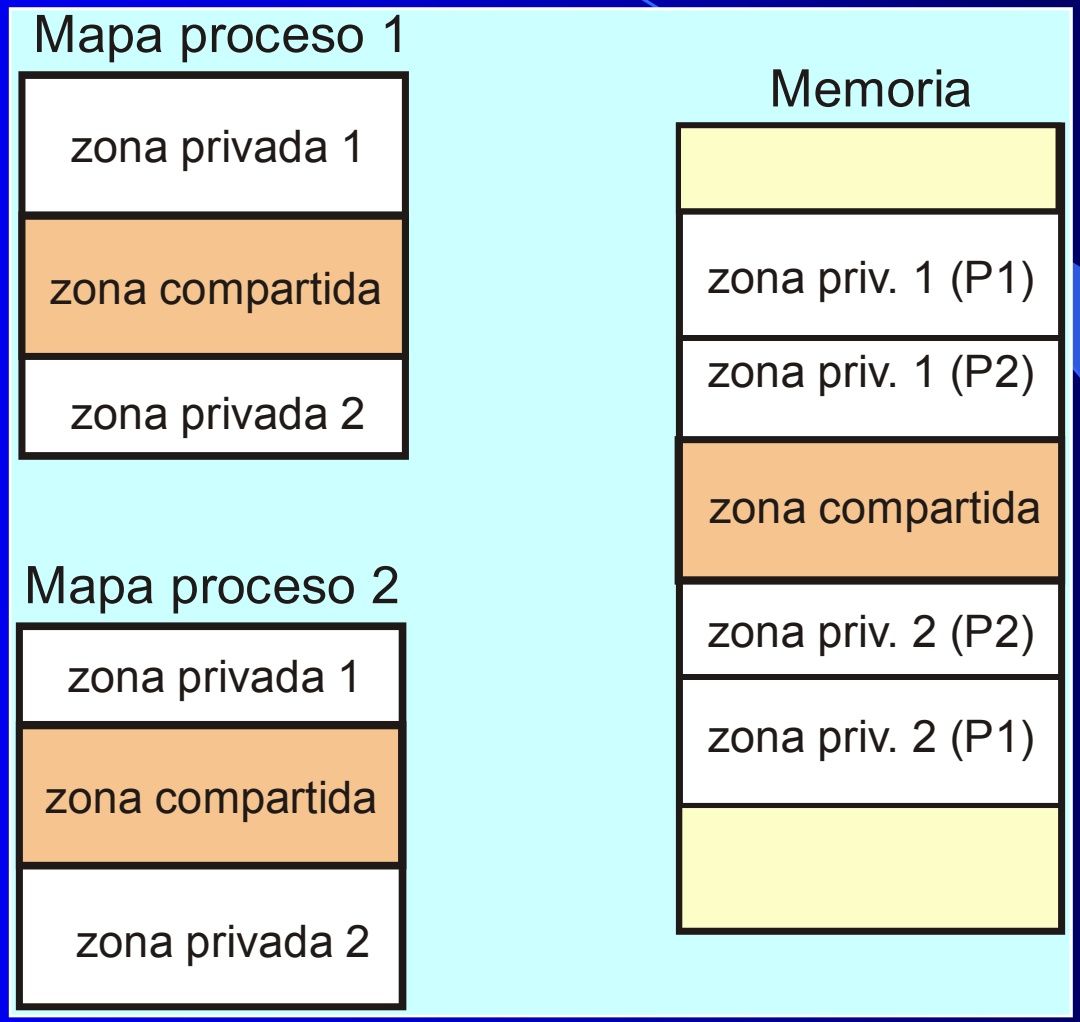
2) Protección



El gestor de memoria debe proteger las zonas asignadas a cada proceso de accesos por parte de terceros; y de haber interferencia debe abandonar tales instrucciones en el momento de la ejecución.

Cada proceso debe protegerse contra interferencias no deseadas de otros procesos, tanto accidentales como intencionadas.

Esta protección debe realizarse dinámicamente (en tiempo de ejecución), ya que todas la referencias a memoria generadas por un proceso deben comprobarse durante la ejecución para asegurar que sólo hacen referencia al espacio de memoria destinado a dicho proceso.


Mapa proceso 1
zona privada 1
zona compartida
zona privada 2
Mapa proceso 2
zona privada 1
zona compartida
zona privada 2
Memoria
zona priv. 1 (P1)
zona priv. 1 (P2)
zona compartida
zona priv. 2 (P2)
zona priv. 2 (P1)

3) Compartición

Cualquier mecanismo de protección que se implemente debe tener la flexibilidad de permitir el acceso de varios procesos a la misma zona de memoria principal.

Por ejemplo, si una serie de procesos están ejecutando el mismo programa, resultaría beneficioso permitir a cada proceso que acceda a la misma copia del programa, en lugar de tener cada uno su propia copia.

El sistema de gestión de memoria debe permitir accesos controlados a las áreas compartidas de la memoria, sin comprometer la protección básica.

4) Organización Lógica

La memoria principal y la secundaria de un sistema se organiza como espacio de direcciones lineal o unidimensional que consta de una secuencia de bytes.

Pero esto no corresponde con la forma que los programas están construidos.

El gestor de memoria debe comprender la organización lógica formada por módulos que tienen los programas (texto, datos, pila...).

La herramienta que satisface esto es la segmentación, que más adelante se verá.

5) Organización Física

El Gestor de memoria debe:

- encargarse de la localización de espacios libres en memoria principal donde cargar los programas.

- mover la información entre la memoria principal y la secundaria

- poder proporcionar un mecanismo para poder ejecutar programas cuyo tamaño supere el de la propia memoria principal (Memoria Virtual).

Carga de Programas en Memoria Principal

La tarea central de cualquier sistema de gestión de memoria es traer los programas a memoria principal para su ejecución en el procesador.

Veremos:

- Partición Fija
- Partición Dinámica
- Paginación Simple
- Segmentación Simple

- **Evolución histórica de métodos de organización hasta llegar a la Memoria Virtual**

- **Esquemas de asignación:**
  - A. Memoria Real
    - Asignación contigua
      - Sistemas de multiprogramación
        - » Con particiones fijas.
        - » Con particiones variables.
    - Asignación no contigua
      - Paginación simple.
      - Segmentación simple.
      - Segmentación + paginación simple.
  - B. Memoria virtual
    - Paginación por demanda.
    - Segmentación por demanda.
    - Segmentación + paginación.

Partición Fija

Se puede suponer que el SO ocupa una parte fija de memoria principal (MP) y que el reto de la memoria está disponible para ser usado por varios procesos.

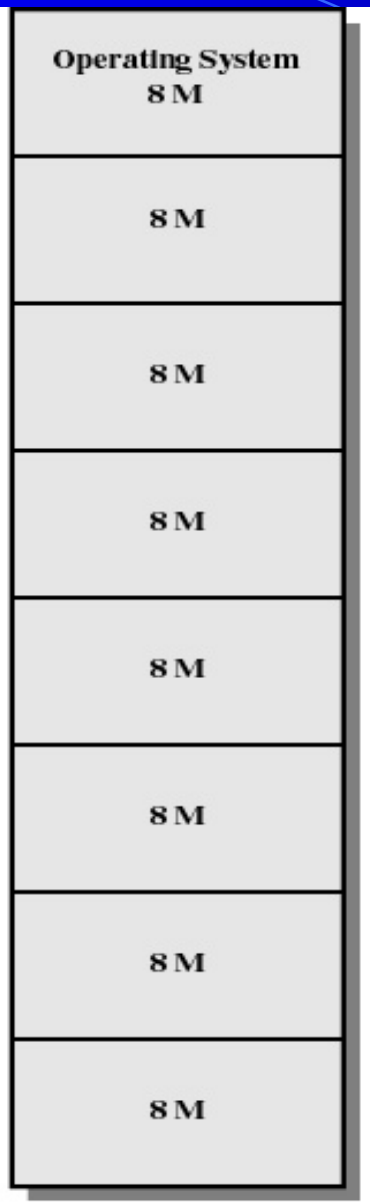
El esquema más sencillo de la gestión de memoria disponible es dividirla en regiones con límites fijos.

Tamaños de Partición Fijas

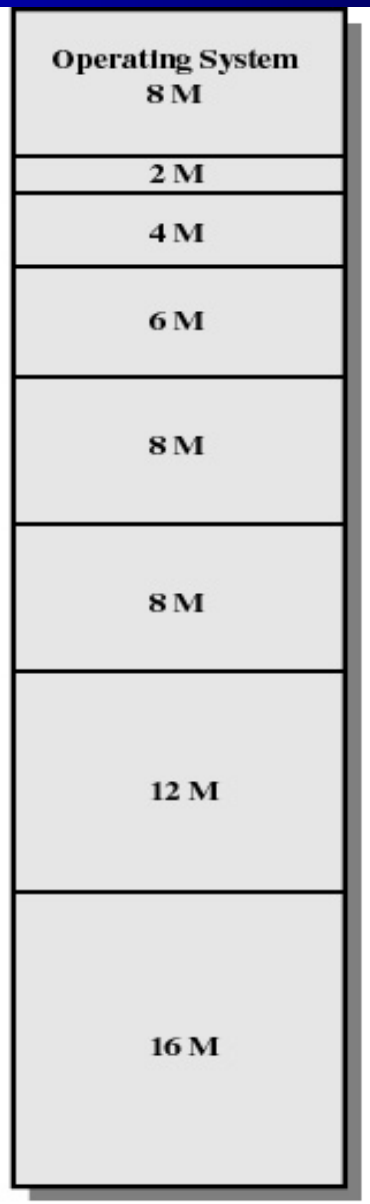
Existen dos alternativas de partición fija:

a) emplear particiones de igual tamaño

b) emplear particiones con distintos tamaños



(a) Equal-size partitions



(b) Unequal-size partitions

Figure 7.2 Example of Fixed Partitioning of a 64-Mbyte Memory

Particiones de igual tamaño

En este caso, cualquier proceso cuyo tamaño sea menor o igual que el tamaño de la partición puede cargarse en cualquier partición libre.

Las particiones de igual tamaño plantean 2 dificultades:

a) un programa puede ser demasiado grande para caber en la partición

Sólo una parte del programa estará en MP en cada instante. Cuando se necesite un módulo no presente, el programa del usuario debe cargar dicho módulo en la partición del programa.

....

b) El uso de MP es ineficiente

Cualquier programa, sin importar lo pequeño que sea, ocupará una partición completa. Este fenómeno se denomina fragmentación interna.

Particiones de distintos tamaños

Reduce ambos problemas, pero no los elimina.
La fragmentación interna es menor, pero existe desperdicio.

Algoritmo de ubicación de particiones fijas

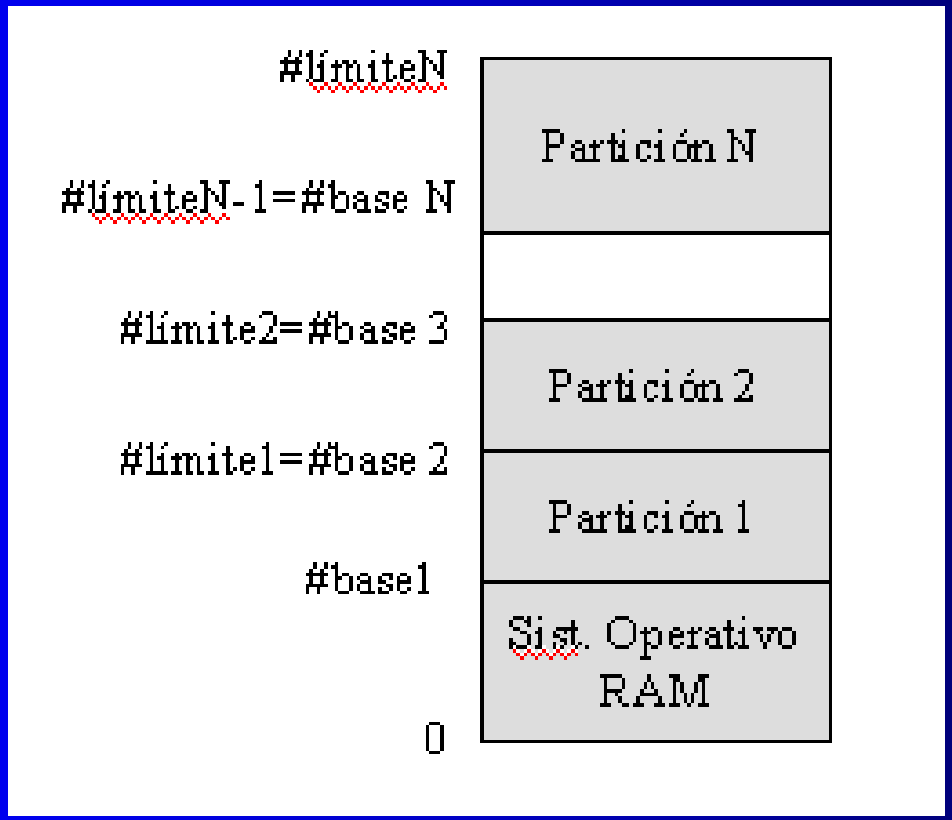
Particiones de igual tamaño

La ubicación de un proceso es trivial. Mientras haya alguna partición libre, puede cargarse un proceso en esa partición, dado que todas las particiones son de igual tamaño.

Si todas las particiones están ocupadas con procesos que no están listos para ejecutar, uno de esos procesos debe sacarse y hacer sitio para el nuevo proceso (esto estará de acuerdo al método de planificación).

Las particiones pueden ser de tamaño fijo, definidas (p.ej.) por el usuario al inicio de la sesión.

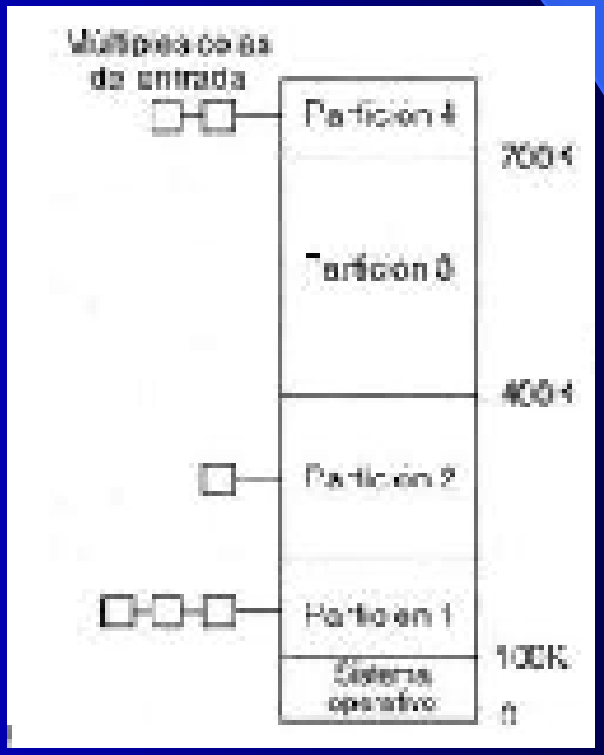
El control de memoria se reduce a asignar una partición a cada proceso entrante, reasignar direcciones (cuando sea necesario) y a proteger unos procesos de los otros.



Particiones de distinto tamaño

Hay 2 maneras posibles de asignar los procesos a las particiones.

La forma más simple es asignar cada proceso a la partición más pequeña en la que quepa. En este caso, hace falta una cola de planificación para cada partición.

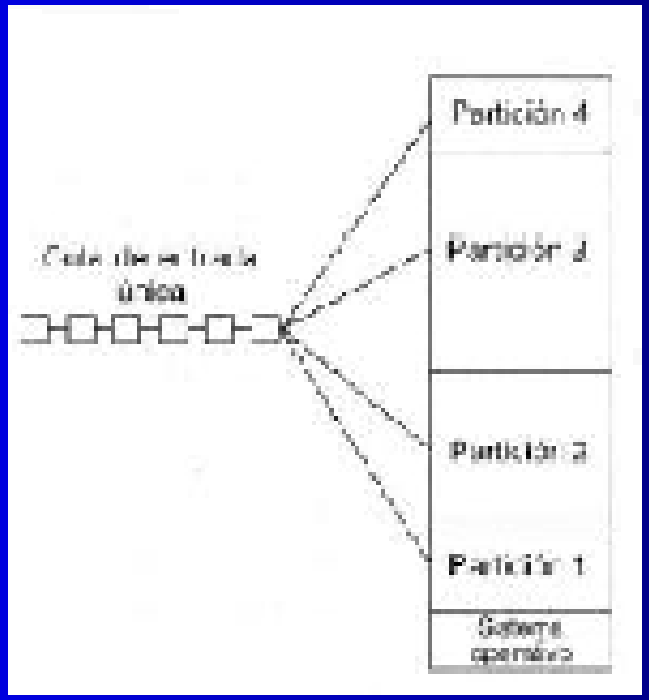


Problema:

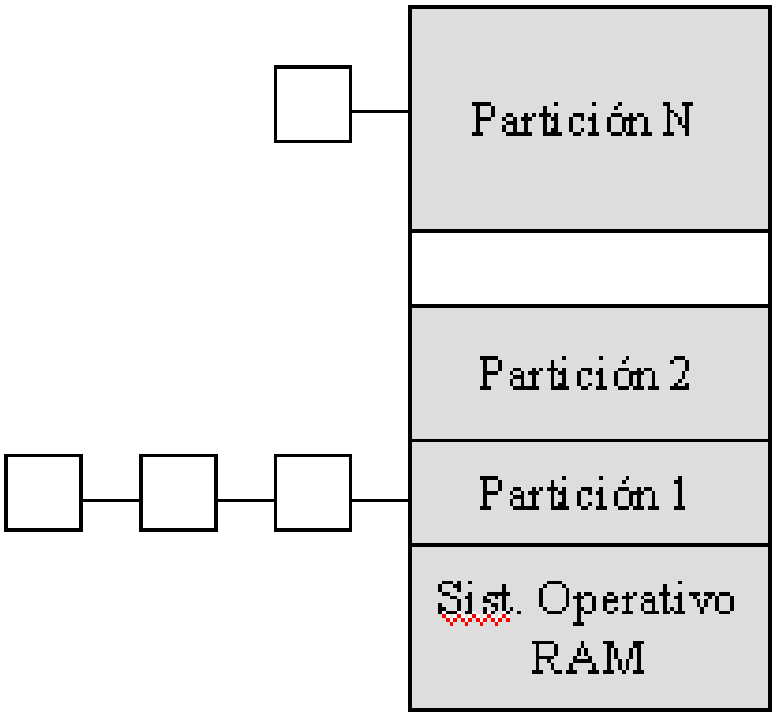
particiones sin usar, incluso con procesos esperando ser asignados (en otra cola de partición)

Una solución mejor sería emplear una única cola para todos los procesos. Cuando se va a cargar un proceso en MP, se selecciona la partición más pequeña disponible que pueda albergar al proceso.

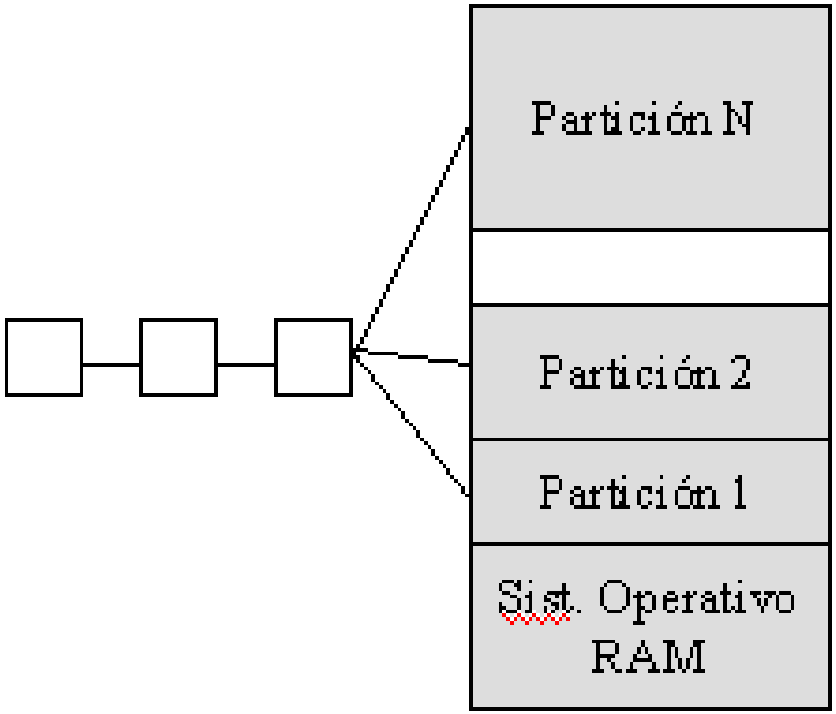
Si todas las particiones están ocupadas, se debe tomar una decisión de intercambio. Considerando otros factores, como prioridad y preferencia para descargar procesos bloqueados antes que procesos listos.



- La asignación pude ser con **múltiples colas** (asignación determinada por el tamaño del proceso entrante) o con una **única cola** (asignación secuencial o planificada -en detrimento de los trabajos más pequeños-)



Asignación con múltiples colas



Asignación con una cola

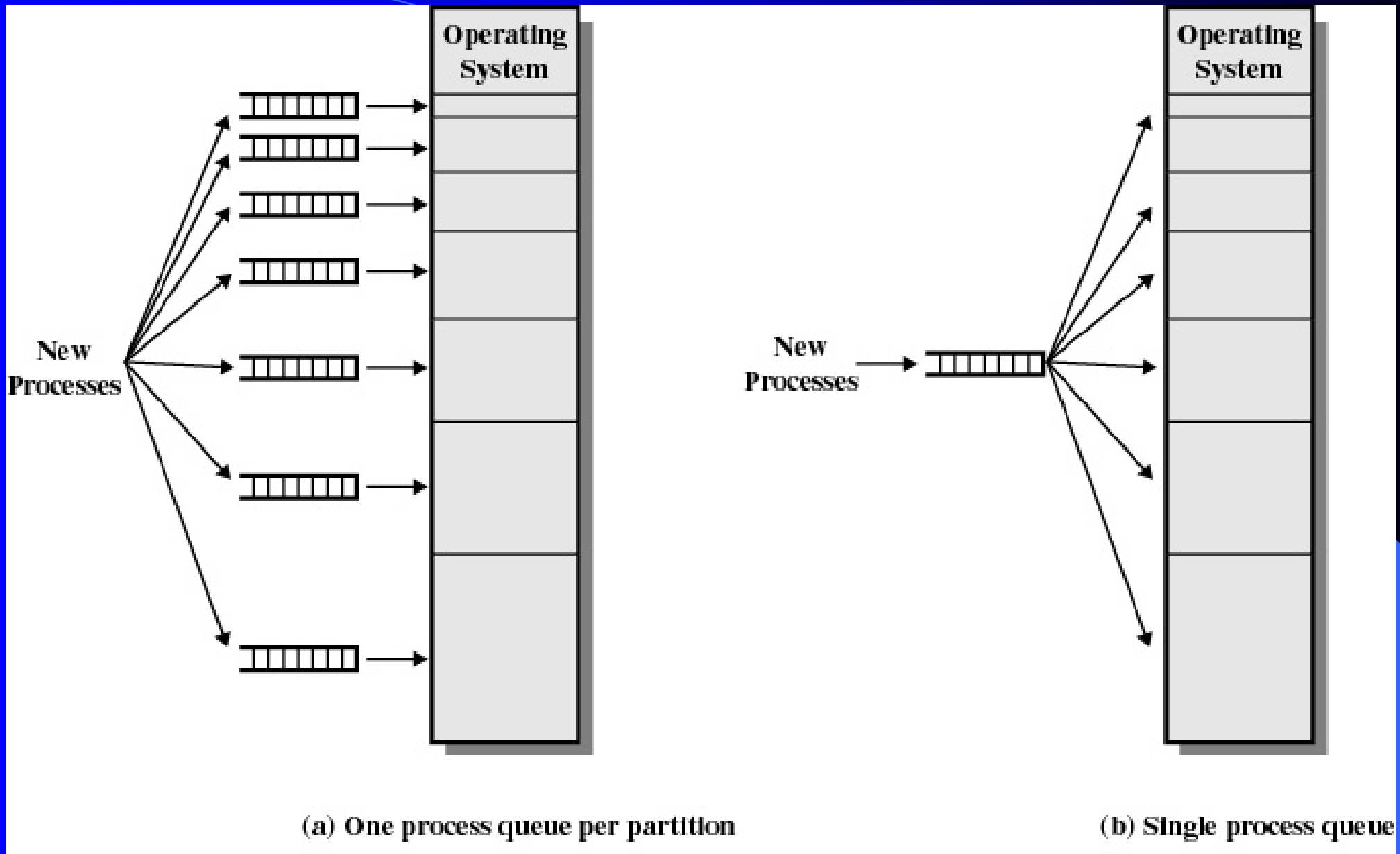


Figure 7.3 Memory Assignment for Fixed Partitioning

El uso de particiones fijas es casi nulo hoy día, se conoce sólo un sistema operativo que la emplea, OS/MFT de IBM.

Esto es porque son limitantes:

- el número de particiones especificadas en el momento de la generación del sistema limita el número de procesos activos del sistema

- los trabajos pequeños no hacen uso eficiente del espacio de las particiones.

Más Inconvenientes:

- Grado de multiprogramación limitado al número de particiones.

- Fragmentación Interna.
  - El proceso es más pequeño que la partición **->** dentro de cada partición queda una zona de memoria no aprovechable.
  - No se puede asignar a ningún otro proceso.
  - Es posible que procesos esperando entrar en memoria no tengan partición a pesar de haber espacio libre para ellos.

Partición Dinámica

Para superar algunas de las dificultades de la partición estática, se desarrolló la partición dinámica.

Este ya también esta obsoleto, y era usado por el SO de IBM, OS/MVT.

Las particiones son variables en número y longitud.

Cuando se trae un proceso a MP, se le asigna exactamente tanta memoria como necesita y nada más.

Veamos un ejemplo que usa 64 Mb de MP:

Al principio , la MP esta vacía, exceptuando el SO (fig.a).

Se cargan los 3 primeros procesos, empezando en donde acaba el SO y ocupando sólo un espacio suficiente para cada proceso (fig. b,c y d).

Esto deja un espacio al final de la MP demasiado pequeño para un cuarto proceso.

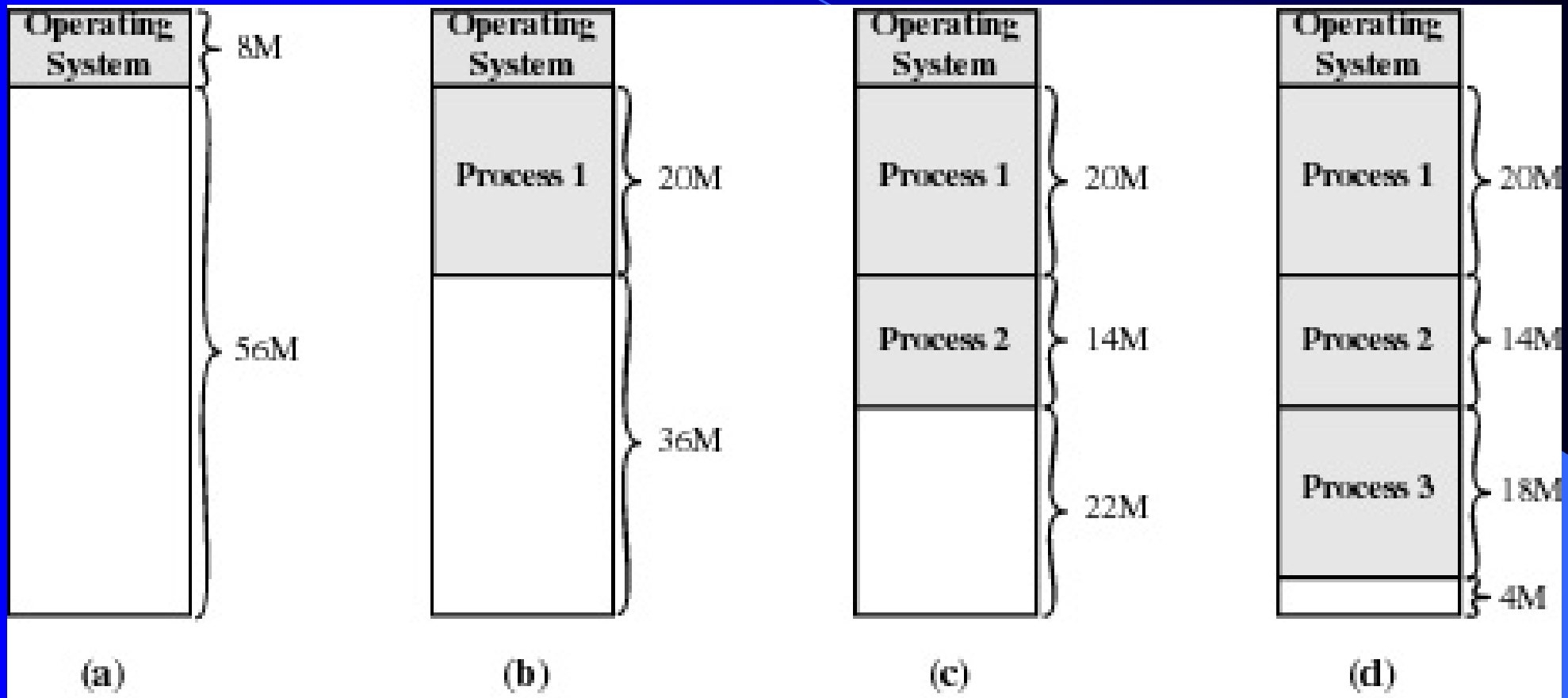


Figure 7.4 The Effect of Dynamic Partitioning

....

En algún momento, ninguno de los procesos en MP está listo.

Así pues, el SO saca al proceso 2 (fig. e), que deja suficiente sitio para cargar un nuevo proceso, el proceso 4 (fig. f)

Puesto que el proceso 4 es más pequeño que el proceso 2, se crea otro espacio pequeño.

Más tarde , el proceso 2 esta listo para entrar, y el SO expulsa al proceso 1 (no listo) (fig. g) y carga de nuevo el proceso 2 (fig. h)

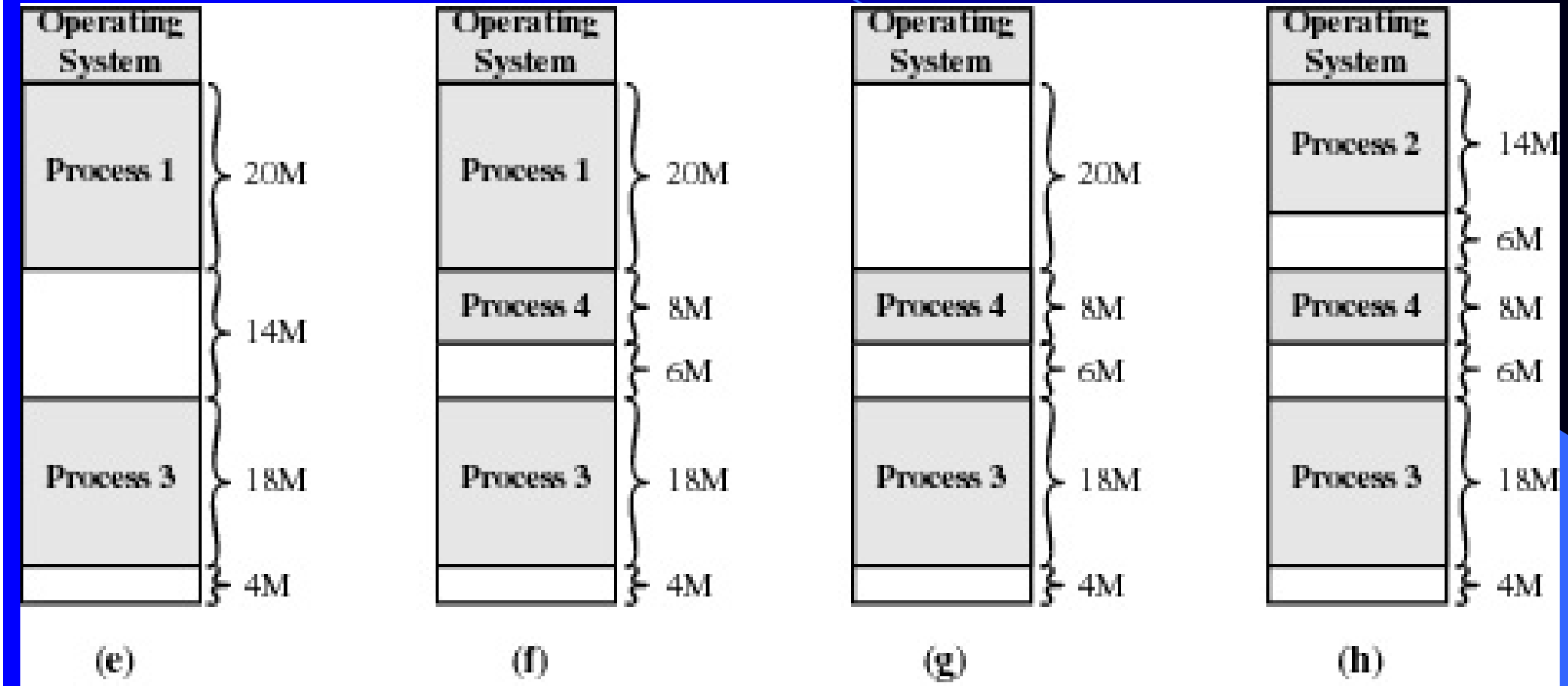


Figure 7.4 The Effect of Dynamic Partitioning

Organización física de la Memoria Principal

– La Memoria Principal se compone de Particiones y Huecos.
– Inicialmente la memoria sólo contiene una partición con el S.O. y un gran hueco/espacio.
– Las particiones son variables en número y longitud (son dinámicas).
– A cada proceso se le asigna la memoria que necesita exactamente.
– ¿Qué hacer si no hay particiones libres?
    a) Esperar a la finalización de algún proceso.
    b) Intercambio.

....

Como se muestra en este ejemplo, el método comienza bien, pero finalmente desemboca en una situación en la que hay un gran número de huecos pequeños en memoria.

A medida que pasa el tiempo, la MP comienza a estar más fragmentada y su rendimiento decae.

Este fenómeno se denomina **Fragmentación Externa**, y se refiere al hecho de que la memoria externa a todas las particiones se fragmenta cada vez más.

....

Una técnica para superar la fragmentación externa es la **compactación.**

De vez en cuando, el SO desplaza los procesos para que estén contiguos de forma que toda la memoria libre quede junta en un bloque.

En el ejemplo visto, se se compacta queda un espacio de 6+6+4 = 16 Mb de memoria libre, lo que puede ser suficiente para cargar un proceso adicional.

La dificultad de la compactación está en que es un procedimiento que consume tiempo, por lo que desperdicia tiempo del procesador.

Algoritmo de Ubicación en partición dinámica

Cuando llega el momento de cargar o traer un proceso a MP y si hay libre más de un bloque de MP de tamaño suficiente, el SO debe decidir cuál asignar.

Hay 3 algoritmos de ubicación, que se limitan a elegir entre los bloques de memoria libres:

- Mejor Ajuste (Best-Fit)
- Primer ajuste (First-Fit)
- Siguiente Ajuste (Next-Fit)

....

El MEJOR AJUSTE elige el bloque de tamaño más parecido al solicitado.

El PRIMER AJUSTE comienza recorriendo la memoria desde el principio y escoge el primer bloque disponible que sea suficientemente grande.

El SIGUIENTE AJUSTE recorre la memoria desde el lugar de la última ubicación y elige el siguiente bloque disponible que sea suficientemente grande.

Ej.:

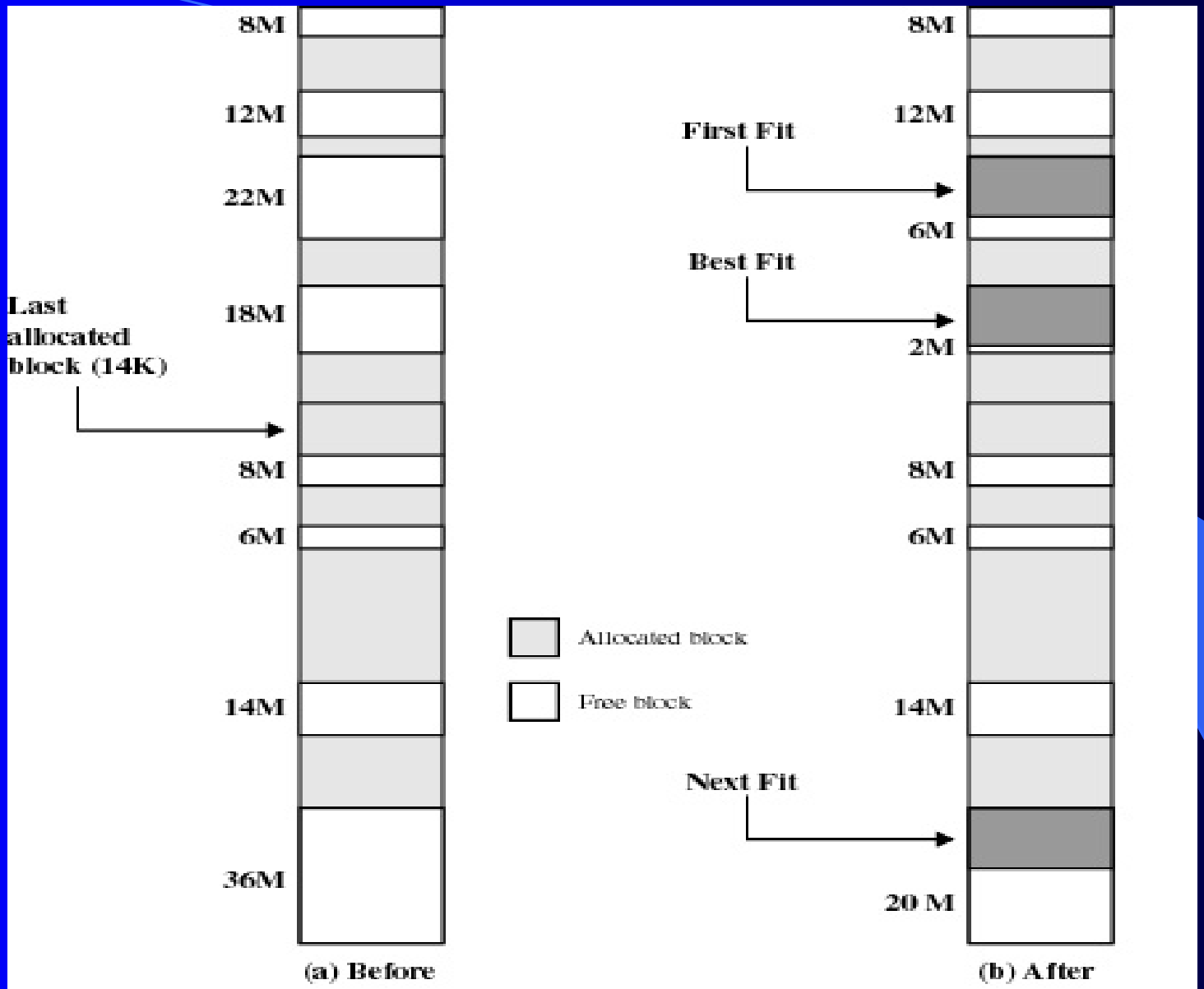


Figure 7.5 Example Memory Configuration Before and After Allocation of 16 Mbyte Block

....

Cuál de estos métodos es mejor dependerá de la secuencia exacta de intercambios de procesos que se den y del tamaño de estos procesos.

El **algoritmo del Primer Ajuste** no sólo es el más sencillo, sino que normalmente es también el mejor y más rápido. Puede poblar el extremo inicial de pequeñas particiones libres que es necesario recorrer en las pasadas siguientes del algoritmo.

....

El **algoritmo del Siguiente Ajuste** llevará frecuentemente a la asignación de bloques libres del final de la memoria. El resultado es que el bloque de memoria libre m´s grande, que suele aparecer al final del espacio de memoria, se divide rápidamente en fragmentos pequeños. Lo que implica compactación más frecuente.

....

El **algoritmo del Mejor Ajuste**, proporciona en general los peores resultados. Puesto que este algoritmo busca el espacio m´s pequeño que cumple con los requisitos, garantiza que el fragmento que se deja es lo más pequeño posible.

Aunque cada solicitud de memoria desperdicia siempre la menor cantidad, el resultado es que la memoria principal se llena rápidamente de bloques demasiados pequeños como para satisfacer las solicitudes de asignación de memoria.

Se debe compactar más frecuentemente que con los otros algoritmos.

....

Best-fit algorithm

- Chooses the block that is closest in size to the request
- Worst performer overall
- Since smallest block is found for process, the smallest amount of fragmentation is left memory compaction must be done more often

First-fit algorithm

- Fastest
- May have many process loaded in the front end of memory that must be searched over when trying to find a free block

Next-fit

- More often allocate a block of memory at the end of memory where the largest block is found
- The largest block of memory is broken up into smaller blocks
- Compaction is required to obtain a large block at the end of memory

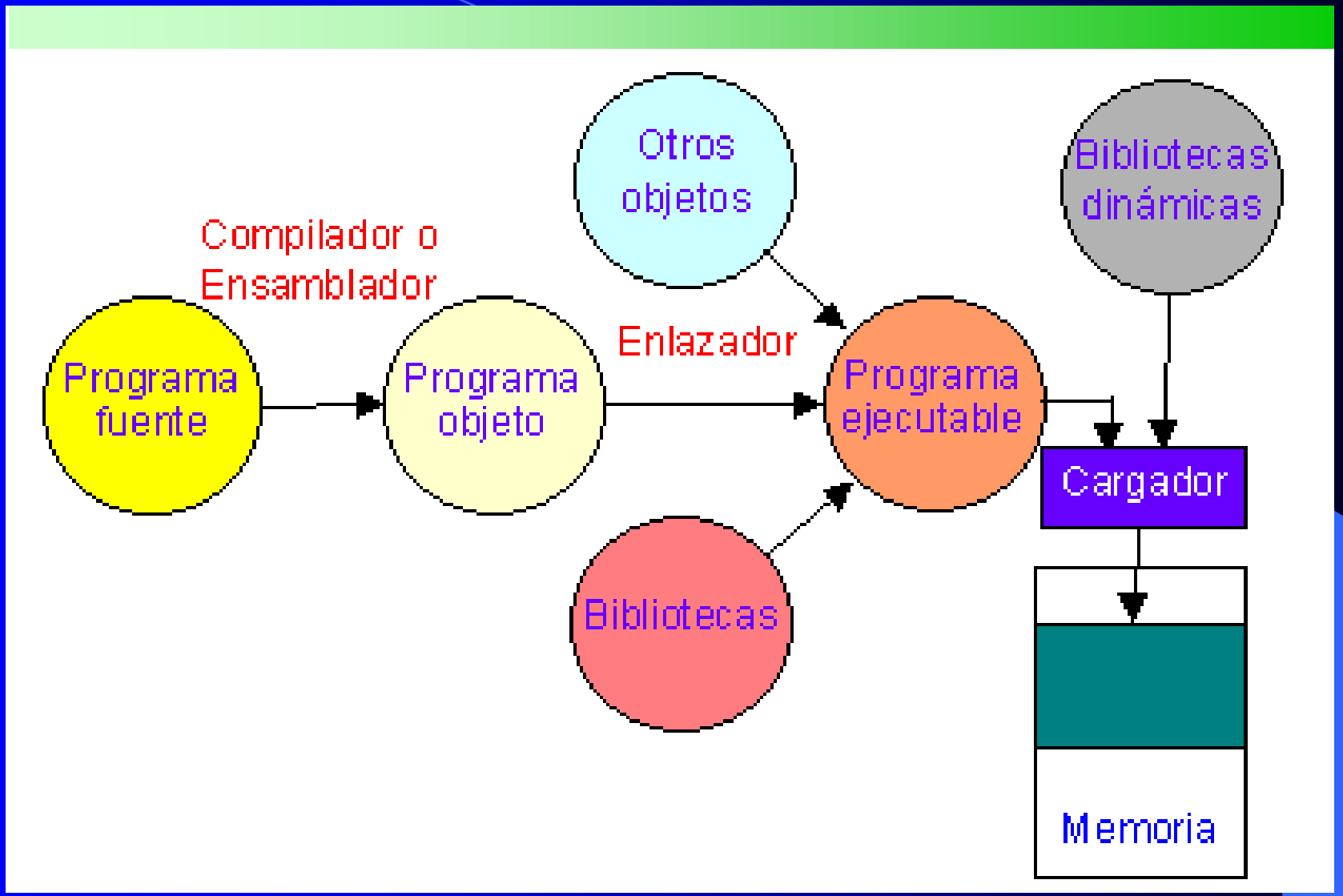
Reubicación

PARTICIONES FIJAS

Cuando se emplea el esquema de particiones fijas se espera que un proceso sea asignado siempre a la misma partición.

Es decir que la partición que se selecciona cuando se carga un nuevo proceso será la misma que se emplee siempre para devolver ese proceso a memoria, tras haber sido sacado.

En este caso se puede emplear un cargador sencillo:
Cuando el proceso se carga por primera vez, todas las referencias relativas a memoria en el código se reemplazan por direcciones absolutas de MP, determinadas por la dirección base del proceso cargado.


Otros objetos
Bibliotecas dinámicas
Compilador o Ensamblador
Programa fuente
Programa objeto
Enlazador
Programa ejecutable
Cargador
Bibliotecas
Memoria

....

PARTICIONES DINAMICAS

Un proceso puede ocupar diferentes particiones a lo largo de su vida.

Cuando al principio se cargará en alguna partición de MP. Posteriormente, el proceso puede ser descargado; cuando, más tarde, vuelva a ser cargado, podrá asignársele una partición distinta de la anterior.

Es más, cuando se usa compactación, los procesos son desplazados durante su estancia en MP.

Consideremos un proceso en memoria que incluya instrucciones y datos.

Las instrucciones contendrán referencias a MP de los dos tipos siguientes:

a) Direcciones de elementos de datos, empleadas en instrucciones de carga, almacenamiento y en algunas instrucciones aritméticas y lógicas.

b) Direcciones de instrucciones, empleadas para bifurcaciones e instrucciones de llamada.

Estas instrucciones no están fijas, sino que cambian cada vez que se intercambia o desplaza un proceso.

Para resolver este problema, se debe realizar una distinción entre varios tipos de direcciones.

....

Una DIRECCION LÓGICA es una referencia a una posición de memoria independiente de la asignación actual de datos a memoria; se debe hacer una traducción a dirección física antes de poder realizar un acceso a memoria.

Una DIRECCION RELATIVA es un caso particular de una dirección lógica, en el cual la dirección se expresa como una posición relativa a algún punto conocido, habitualmente el comienzo del programa

Una DIRECCION FÍSICA o ABSOLUTA es una posición real en MP.

Dirección lógica.

– Referencia a posición de memoria independiente de la asignación actual de datos.

– Generada por la CPU.

- **Espacio de direcciones lógicas.**

– Conjunto de direcciones generadas por un programa o por un proceso. La primera dirección siempre es la cero y la última el tamaño del proceso-1.

- **Dirección física.**

– Designa una posición real de la memoria principal.

– Aquella cargada en el registro de direcciones de la memoria.

- **Espacio de direcciones físicas.**

– Conjunto de posiciones de memoria física correspondientes a las direcciones lógicas.

– Las direcciones dependen de dónde se haya ubicado al proceso.

....

Los programas que emplean direcciones relativas en MP se cargan por medio de cargadores dinámicos durante la ejecución. Esto significa que todas las referencias a memoria en el proceso cargado son relativas al origen del programa.

Por ello, se necesita en el HW un medio para traducir las direcciones relativas a direcciones físicas en MP en el momento de la ejecución de la instrucción que contiene la referencia.

....

Veamos como se realiza normalmente esta traducción de direcciones.

Cuando un proceso pasa a estado Ejecutando (running), se carga con la dirección en MP del proceso registros especiales:

- registro base
- registro límite (indica la posición final del programa)

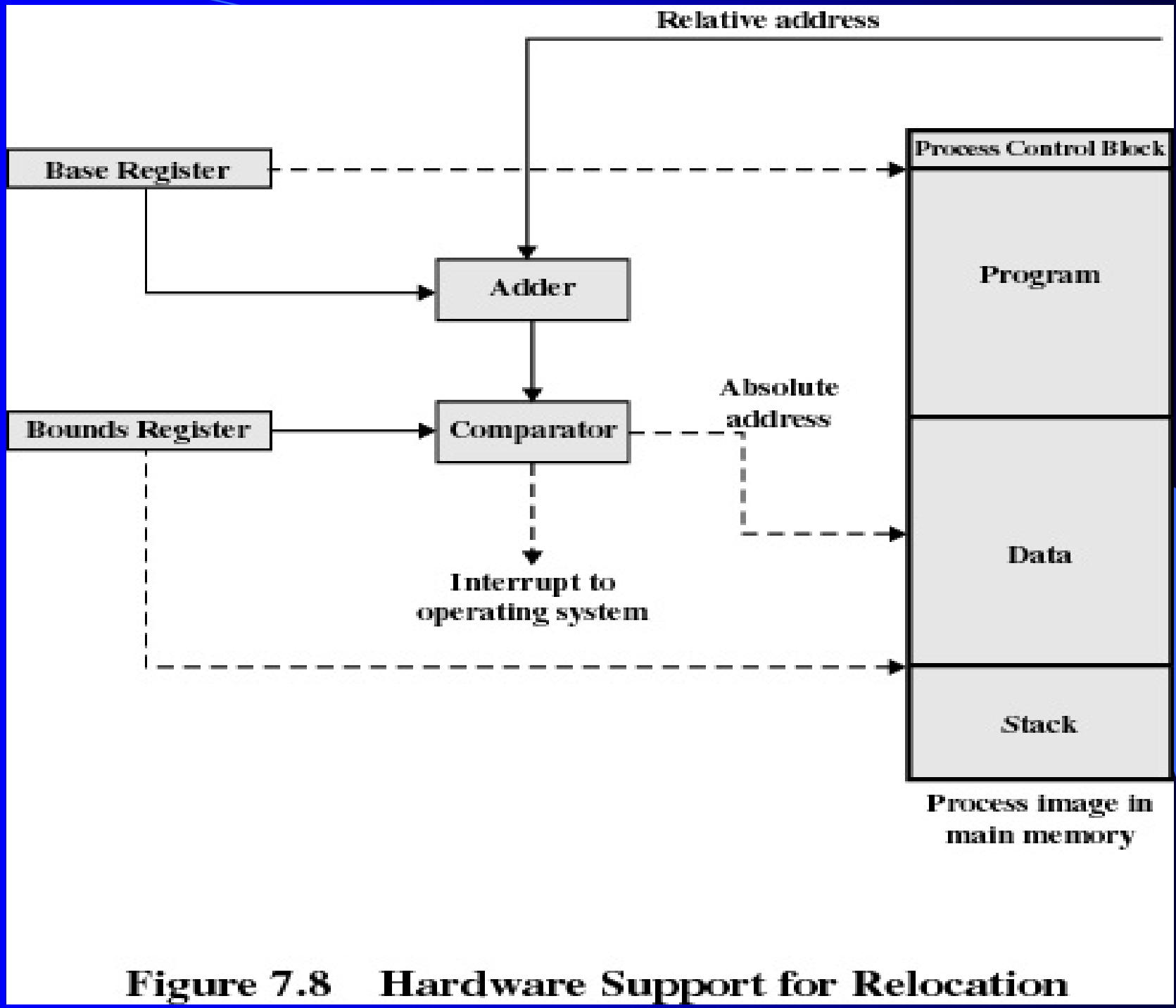


Figure 7.8 Hardware Support for Relocation

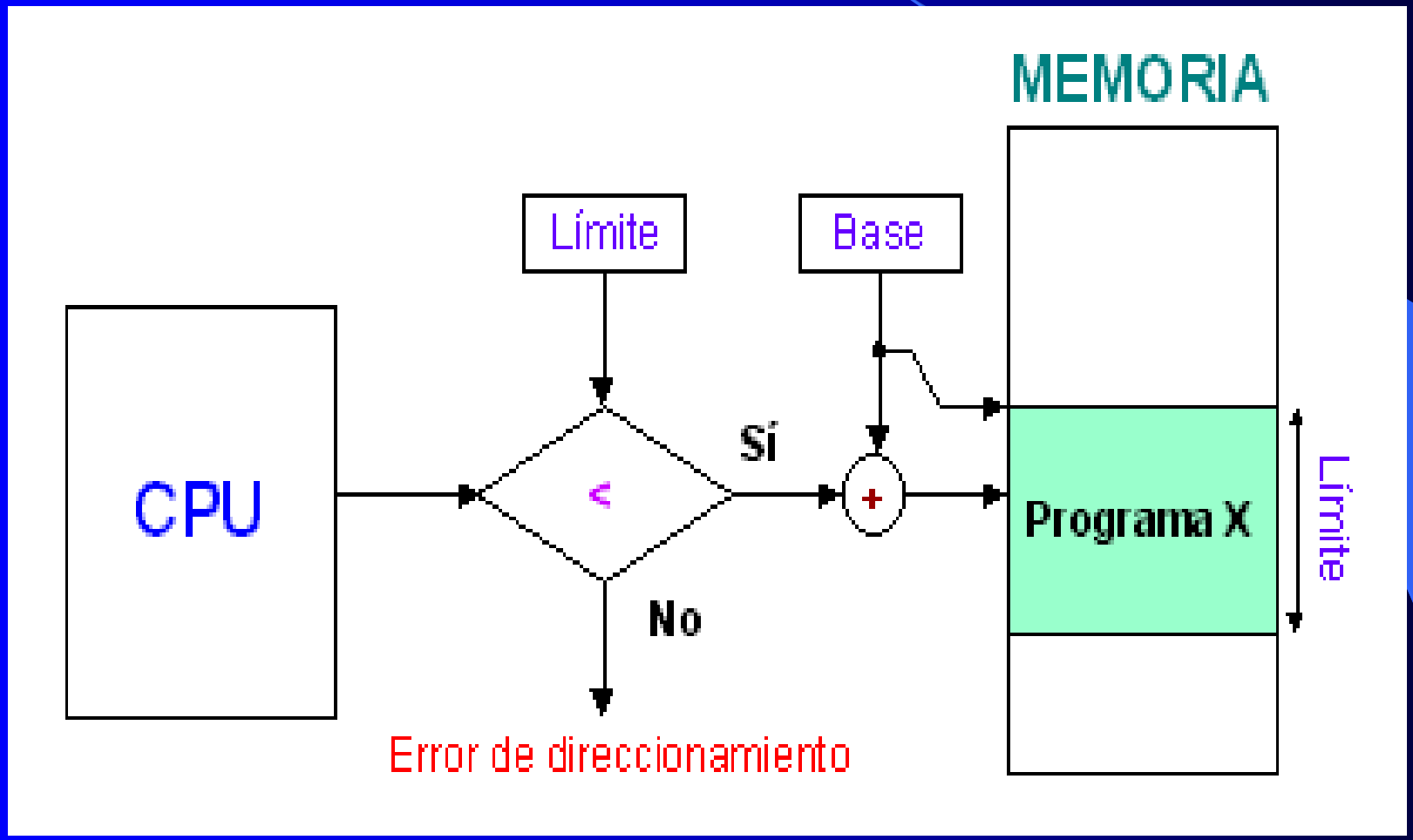
Los valores de los registros se deben asignar cuando se carga el programa/proceso en memoria.

A lo largo de la ejecución del proceso se encuentran direcciones relativas. Estas direcciones incluyen los contenidos del registro de instrucción, las direcciones que aparecen en las instrucciones de bifurcación y llamada, y las direcciones de datos que aparecen en las instrucciones de carga y almacenamiento.

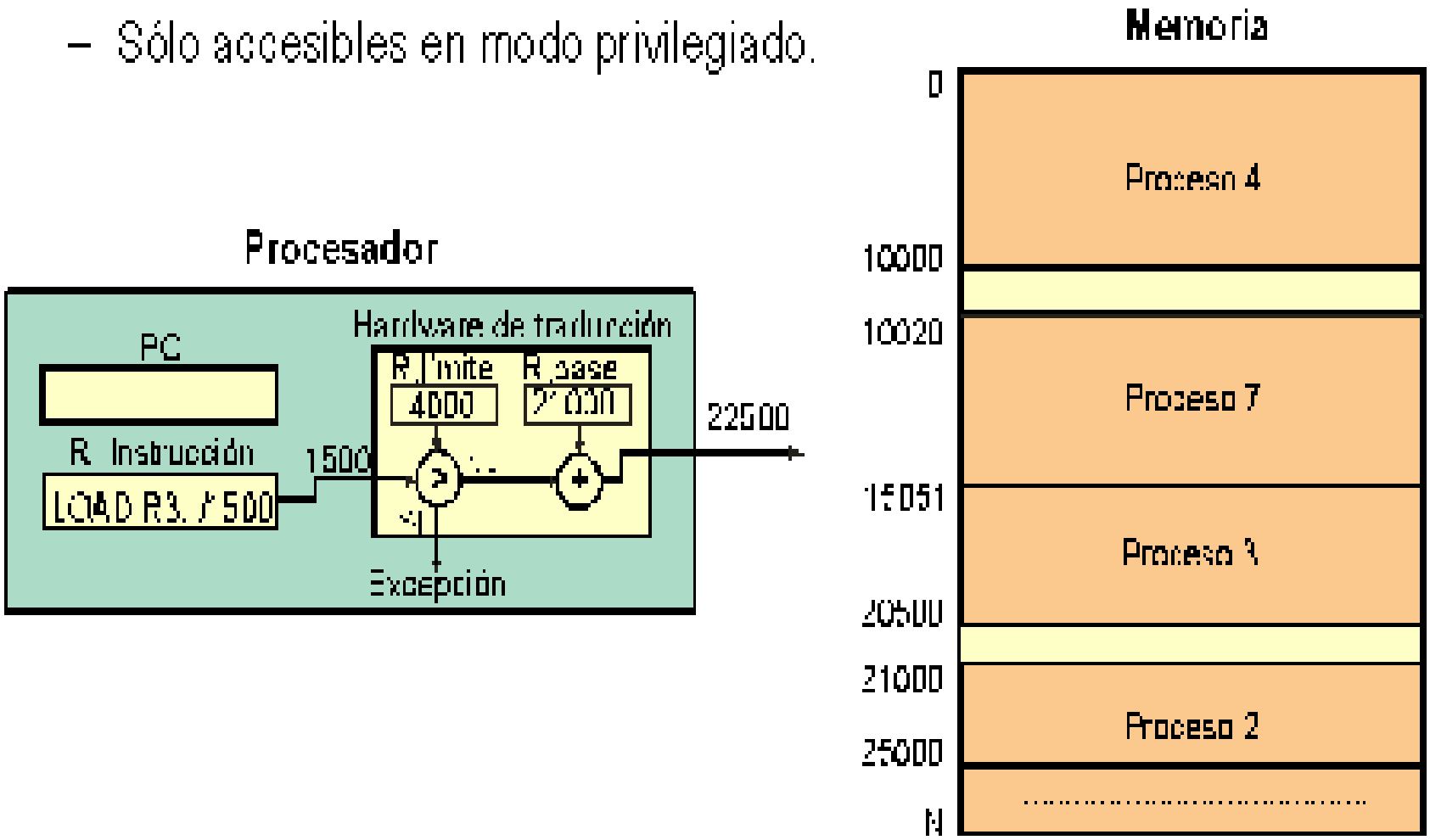
Cada una de estas direcciones relativas pasa por 2 etapas de manipulación en la CPU:

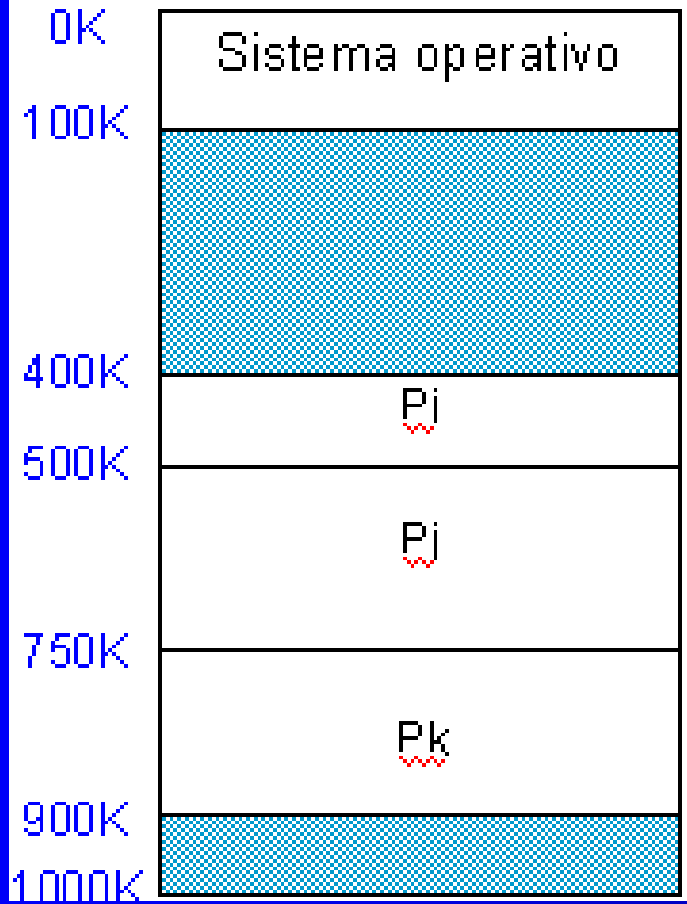
1) se añade el valor del registro base a la dirección relativa para obtener una dirección absoluta.

2) la dirección obtenida se compara con el valor del registro límite. Si la dirección está dentro de los límites, se puede proceder con la ejecución de la instrucción. En otro caso, se genera una interrupción al SO, que debe responder al error del algún modo.


MEMORIA
Límite
Base
CPU
<
Si
+
Programa X
Límite
No
Error de direccionamiento

- Hardware requerido: Registros valla (**Rbase** y **Rlímite**)
  - Sólo accesibles en modo privilegiado.





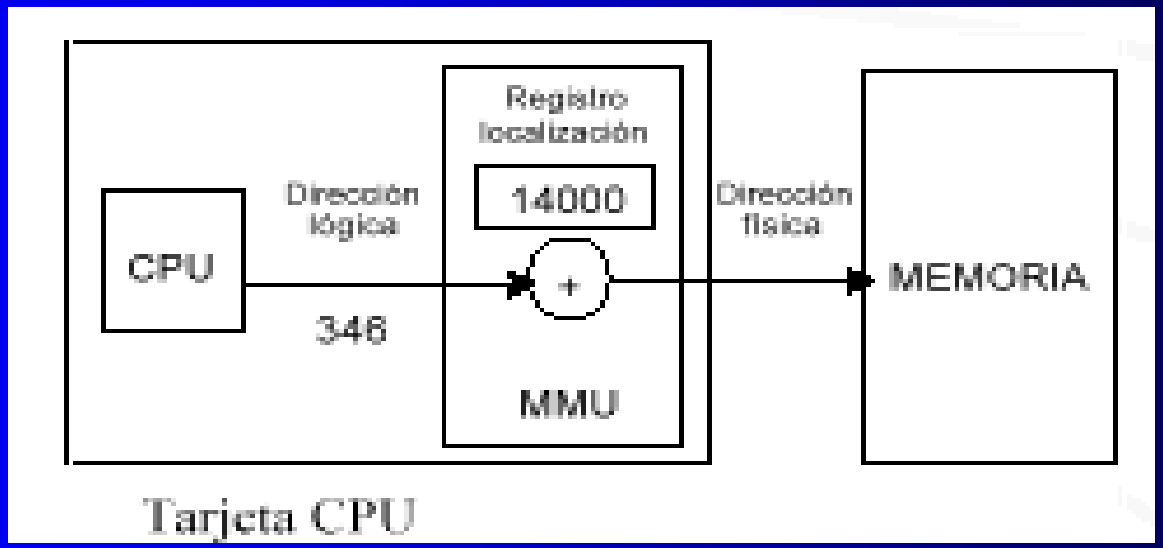
Número de la partición	Base de la partición	Tamaño de la partición	Estado de la partición
0	0K	100K	ASIGNADA
1	100K	300K	LIBRE
2	400K	100K	ASIGNADA
3	500K	250K	ASIGNADA
4	750K	150K	ASIGNADA
5	900K	100K	LIBRE

Unidad de Gestión de Memoria (MMU)

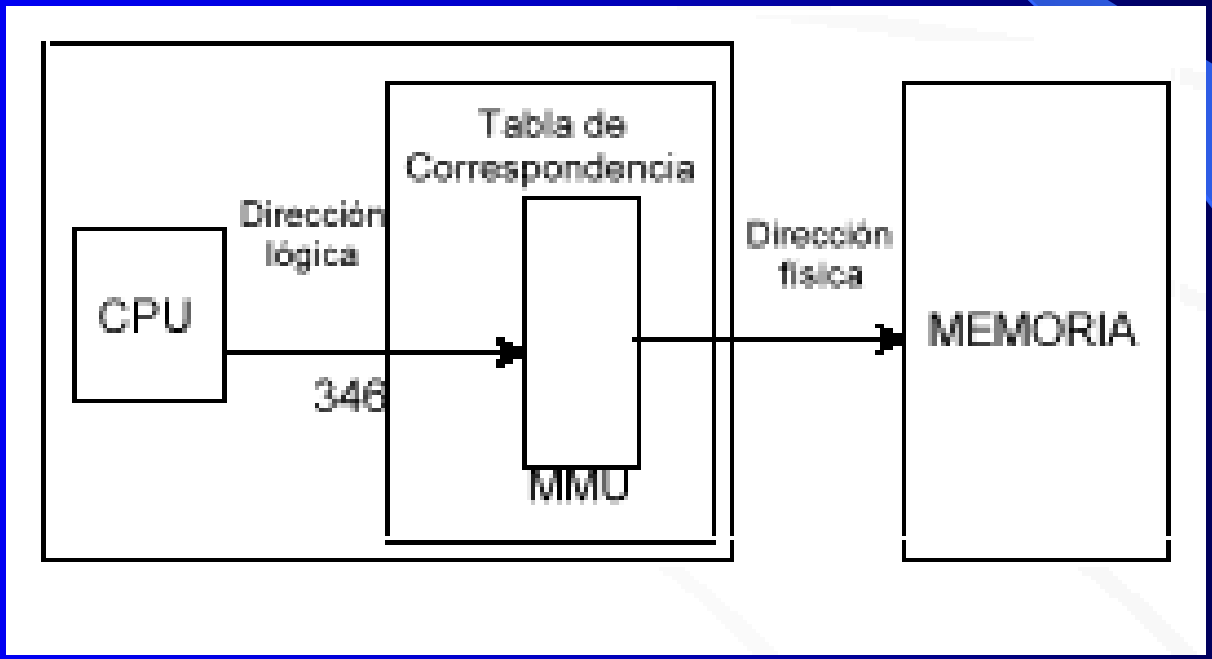
La MMU (Memory Management Unit) es un dispositivo hardware que aplica direcciones virtuales (lógicas) en direcciones físicas en tiempo de ejecución. La MMU suele estar gestionada por el SO.

- Diversos esquemas de traducción:

1. Registro de relocalización.



2. Tabla de Correspondencia



Asignación no contigua de Memoria

- El espacio de direcciones lógico de un proceso se reparte entre diferentes zonas de la memoria principal.

- Métodos de organización:

  - Paginación simple.
  - Segmentación simple.

Paginación Simple

Tanto las particiones de tamaño fijo como las de tamaño variable hacen uso ineficiente de la memoria.

Las primeras generan fragmentación interna; mientras que las segundas originan fragmentación externa.

¿¿ como podemos solucionar esto ??

Con paginación !!

....

• **Organización física de la Memoria Principal**
– División de la memoria principal en bloques de igual tamaño denominados **marcos de página.**

• **Organización del espacio de direcciones lógico de un proceso**
– División del espacio de direcciones de un proceso en bloques de igual tamaño denominados **páginas**.

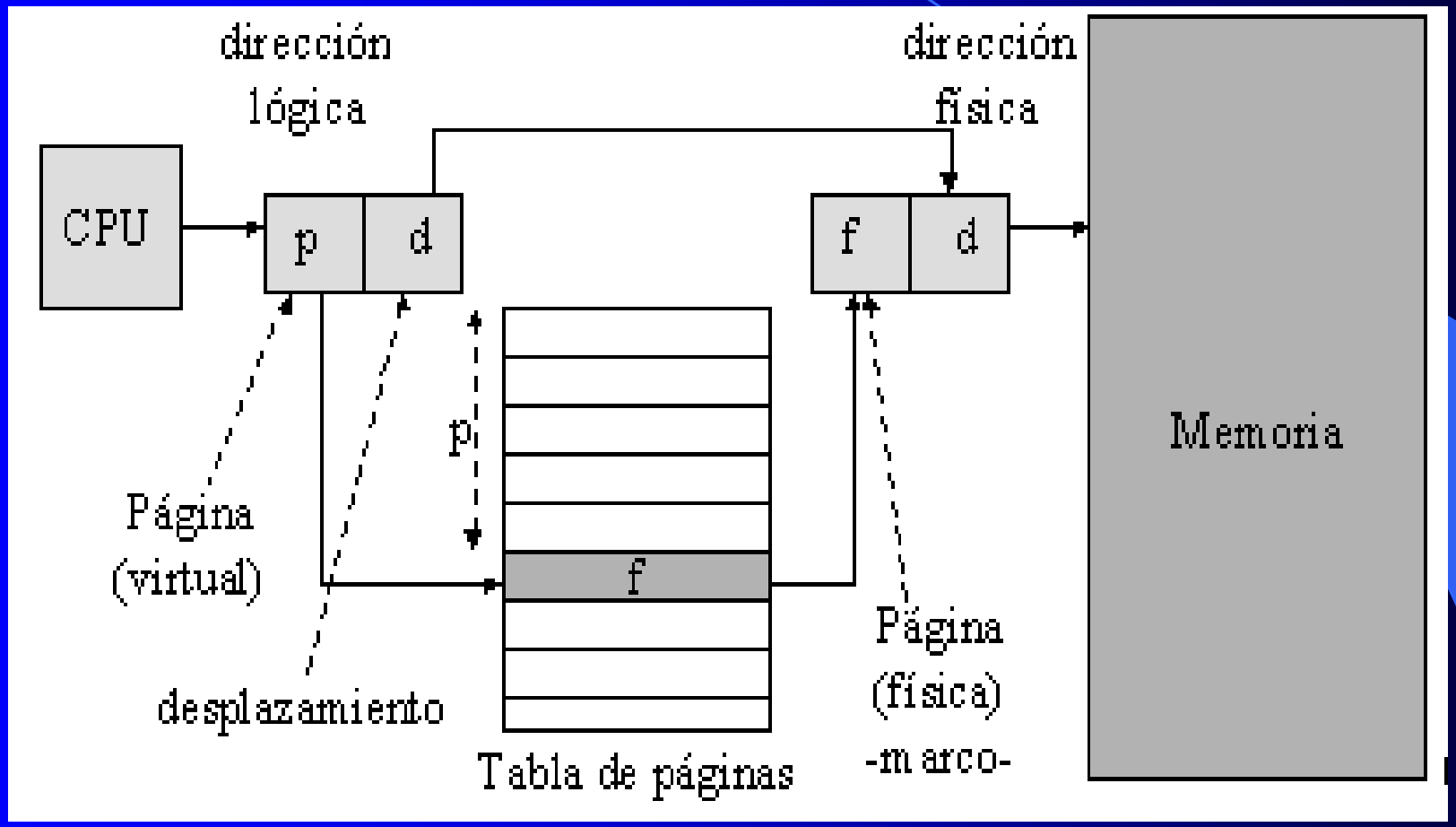
– Marcos y páginas tienen el mismo tamaño.

• **Estrategias de asignación**
– Se van asignando marcos libres a páginas del proceso.
– No tienen por qué ser consecutivos.

....

Suponga que la MP se encuentra particionada en trozos iguales de tamaño fijos relativamente pequeños y que cada proceso está dividido también en pequeños trozos de tamaño fijo y del mismo tamaño que los de la memoria.

En tal caso, los trozos del proceso (conocidos como **páginas o pages**) , pueden asignarse a los trozos libres de memoria (conocidos como **marcos de página o frames**).


dirección
lógica
dirección
física
CPU
p
d
f
d
Memoria
p
Página
(virtual)
f
desplazamiento
Página
(física)
-marco-
Tabla de páginas

Ej. De uso de páginas y marcos

En un instante dado, algunos de los marcos de memoria están en uso y otros están libres.

El SO mantienen una lista de los marcos libres.

El Proceso A, almacenado en disco, consta de 4 páginas, cuando llega el momento de cargar el proceso, el SO busca 4 marcos libres y carga las cuatro páginas del proceso A en los 4 marcos.
El proceso B, que consta de 3 páginas y el proceso C, que consta de 4 páginas, se cargan a continuación.
Más tarde, el proceso B se suspende y es expulsado de MP.
Después el SO trae un nuevo proceso, el proceso D, que consta de 5 páginas.

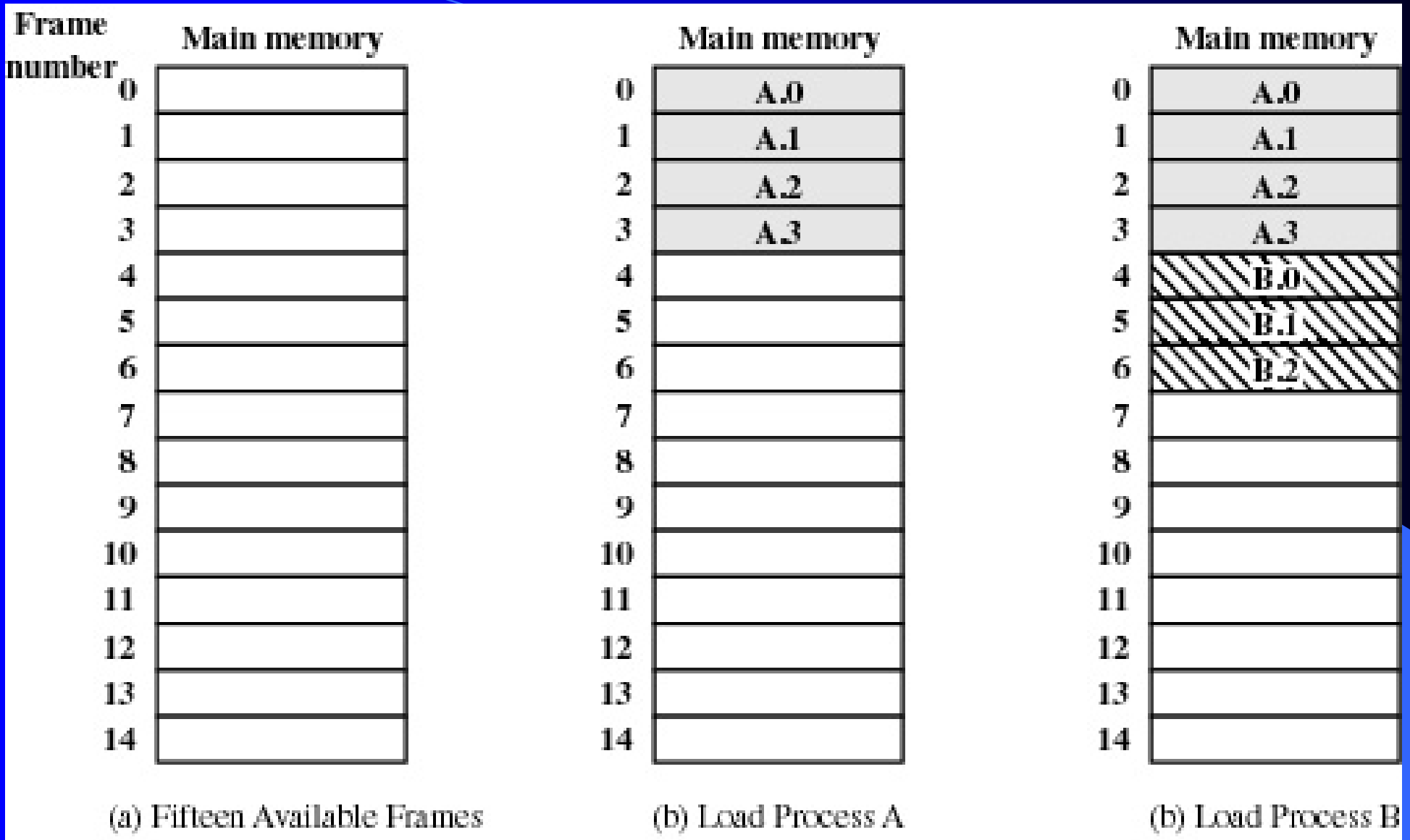


Figure 7.9 Assignment of Process Pages to Free Frames

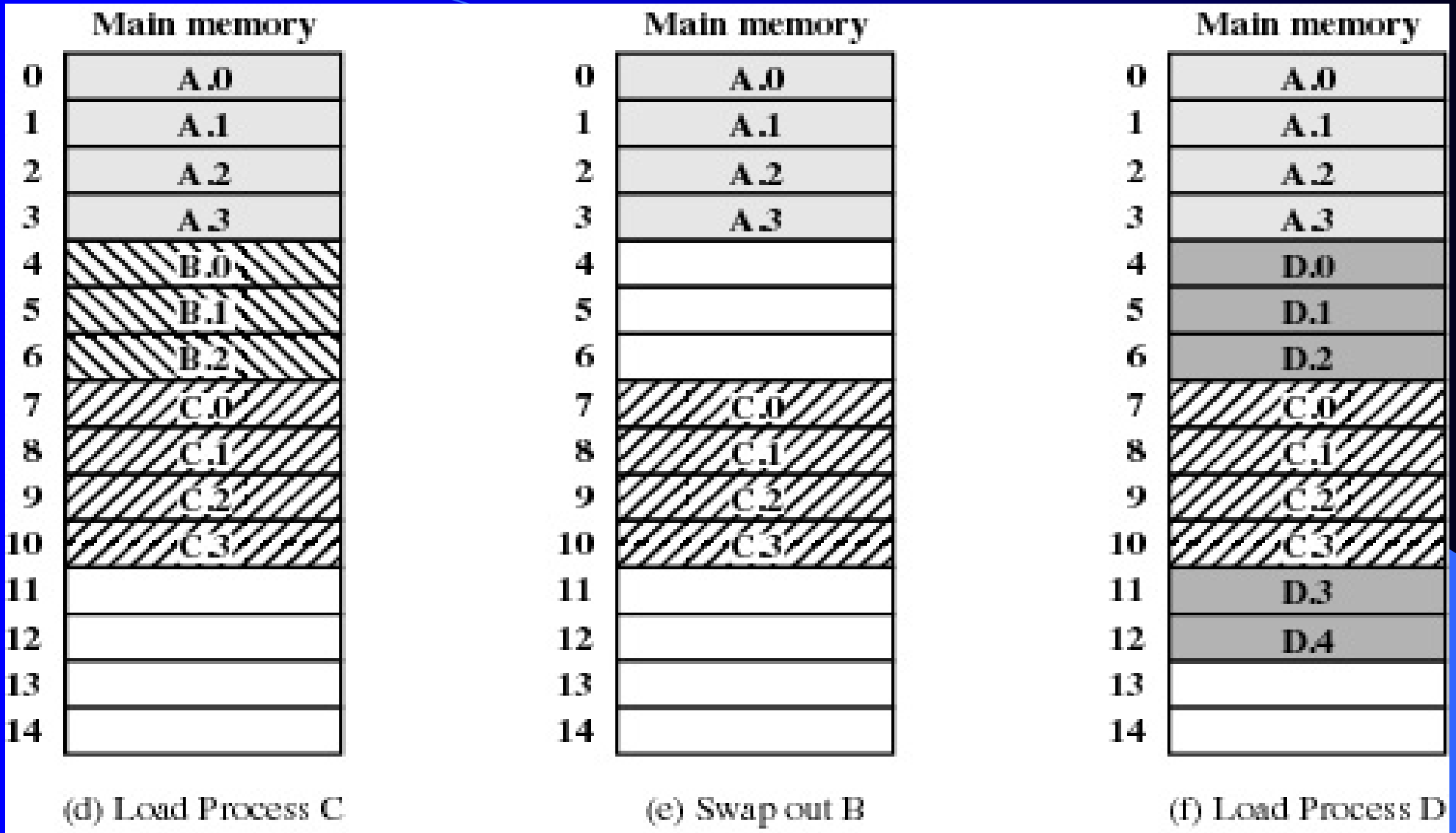


Figure 7.9 Assignment of Process Pages to Free Frames

....

El SO mantiene una **tabla de páginas** para cada proceso.

La tabla de páginas muestra la posición del marco de cada página del proceso.

Dentro del programa, cada dirección lógica constará de un número de página y de un desplazamiento dentro de la página.

El HW del procesador realiza la traducción de direcciones lógicas a físicas.

- - - -

El procesador debe saber cómo acceder a la tabla de páginas del proceso actual.

Dada una dirección lógica (número de página, desplazamiento) el procesador emplea la tabla de páginas para obtener una dirección física (número de marco, desplazamiento).

Siguiendo con el ejemplo anterior, se muestran las distintas tablas de páginas en ese instante

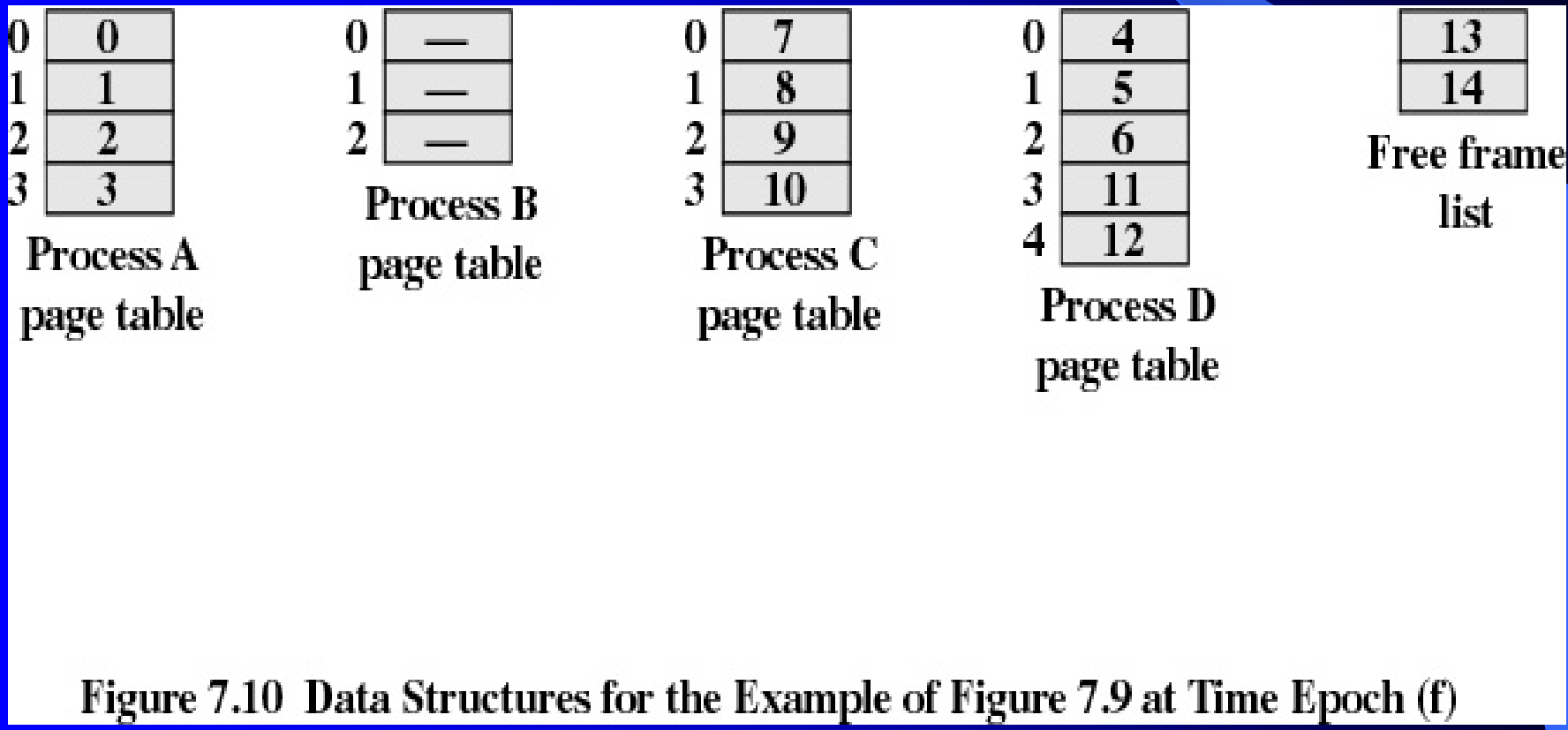


Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)

....

Cada tabla de páginas contiene una entrada por cada página del proceso, por lo que la tabla se indexa fácilmente por número de página.

En cada entrada de la tabla de páginas se encuentra el número de marco de memoria que alberga la página correspondiente.

Además el SO mantiene una lista de marcos libres con todos los marcos de memoria que actualmente están vacíos y disponibles para las páginas.

Generalmente el tamaño de la página y del marzo es potencia de 2.

Estructura de datos para la Paginación

1. Tabla de Páginas.

- Una tabla por cada proceso.
- La tabla tiene tantas entradas como páginas tenga el proceso.
- En cada entrada se almacena:

– El número de marco donde se guarda la página cuyo número coincide con el índice de la tabla.

– Bits de protección (lectura, escritura…).

2. Tabla de marcos.

- Una única tabla para todo el sistema.
- La tabla tiene tantas entradas como marcos de memoria física.
- En cada entrada se almacena:
  - Flag indicando si el marco est á libre o asignado.
  - En el caso de estar asignado el número de página y el PID del proceso al que pertenece.

- **Modos de Traducción**

Con operaciones aritméticas:

i) pagina=dir_logica div tamanio_pagina

ii) desplazamiento=dir_logica mod tamanio_pagina

iii) Acceder a la posición ‘pagina’ de la tabla de páginas del proceso para obtener así el número de marco.

iv) dir_fisica=marco*tamanio_pagina+desplazamiento

.... Ventajas e Inconvenientes de la Paginación

- La paginación no es visible al usuario del S.O.

- Se elimina la fragmentación externa.

- La fragmentación interna sólo se produce en la última página.

- 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.

Segmentación Simple

Otro modo de subdividir el programa es la segmentación.

En este caso, el programa y sus datos asociados se dividen en un conjunto de **segmentos**.

No es necesario que todos los segmentos de todos los programas tengan la misma longitud, aunque existe una longitud máxima de segmento.

Una dirección lógica segmentada consta de 2 partes: un número de segmento y un desplazamiento.

Un esquema de segmentación simple hará uso de una tabla de segmentos para cada proceso y una lista de bloques libres en MP.

Cada entrada de tabla de segmentos tendría que contener la dirección de comienzo del segmento correspondiente en MP.

La entrada deberá proporcionar también la longitud del segmento para asegurar que no se usan direcciones no válidas.

- **Organización física de la Memoria Principal**

– La Memoria Principal se compone de Particiones y Huecos.
– Inicialmente la memoria sólo contiene una partición con el S.O. y un gran hueco.
– Las particiones son variables en número y longitud (son dinámicas).

- **Organización del espacio de direcciones lógico de un proceso**

– 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 zonas de memoria contigua.

- **Estructuras de datos para la gestión de la Segmentación**

– Lista de huecos.
– Tabla de Segmentos.
    - Una tabla por cada proceso.
    - La tabla tiene tantas entradas como segmentos tenga el proceso.
    - En cada entrada se almacena:
        – Dirección base (valor del registro base) y límite o longitud del segmento (valor del registro límite).
        – Bits de protección (lectura, escritura…).

.... Ventajas e Inconvenientes de Segmentación

- Facilita la compartición: se comparten unidades lógicas o segmentos (datos, texto).

- Se pueden proteger los segmentos (bits de protección).

- Facilita la ampliación de las estructuras de datos (solo 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.

Resumen

Una de las tareas más importantes y complejas de un SO es la gestión de memoria.

La gestión de memoria implica tratar la MP como un recurso que asignar y compartir entre varios procesos activos.

Para un uso eficiente del procesador y de los servicios de E/S, resulta interesante mantener en MP tantos procesos como sea posible.

Las herramientas básicas de la gestión de la memoria son la paginación y la segmentación.