Administración de la Memoria

Roberto Giordano Lerena

Pandemia COVID-19

Figuras tomadas de Stallings y de Silberschatz

Agenda

- ✓ Gestión de memoria e Indicadores
- ✓ Direcciones, direccionamiento y registros
- ✓ Principios de gestión de la memoria
- ✓ Técnicas de gestión de memoria
  - Máquina Desnuda
  - Monitor Residente
  - Intercambio Superpuesto
  - MFT
  - MVT
  - Paginación
  - Segmentación

Gestión de memoria

La memoria es uno de los recursos básicos del Sistema de Computación.

Necesitamos gestionarla para hacer un uso eficiente de ella (que no haya memoria ociosa) y alojar la mayor cantidad de procesos posibles (Grado de multiprogramación).

Eso mejora potencialmente el *% de Uso Efectivo del Procesador.*

Fragmentación

Llamamos **Fragmentación** a la cantidad de memoria ociosa (no utilizada efectivamente), esté disponible o no.

- ✓ Si esa memoria no utilizada está asignada a un proceso, entonces solo él puede usarla. Si no la usa: Fragmentación Interna.
- ✓ Si esa memoria no utilizada no está asignada a ningún proceso, entonces está disponible para su uso: Fragmentación Externa

Indicadores

- ✓ Grado de multiprogramación (N) ⬆
- ✓ Tamaño máximo del proceso (N) ⬆
- ✓ Fragmentación interna (FI) ⬇
- ✓ Fragmentación externa (FE) ⬇

Se deben verificar, además, los mecanismos de garantía de la **Protección**, para evitar que un proceso no acceda a un área de memoria de otro proceso sin el permiso necesario.

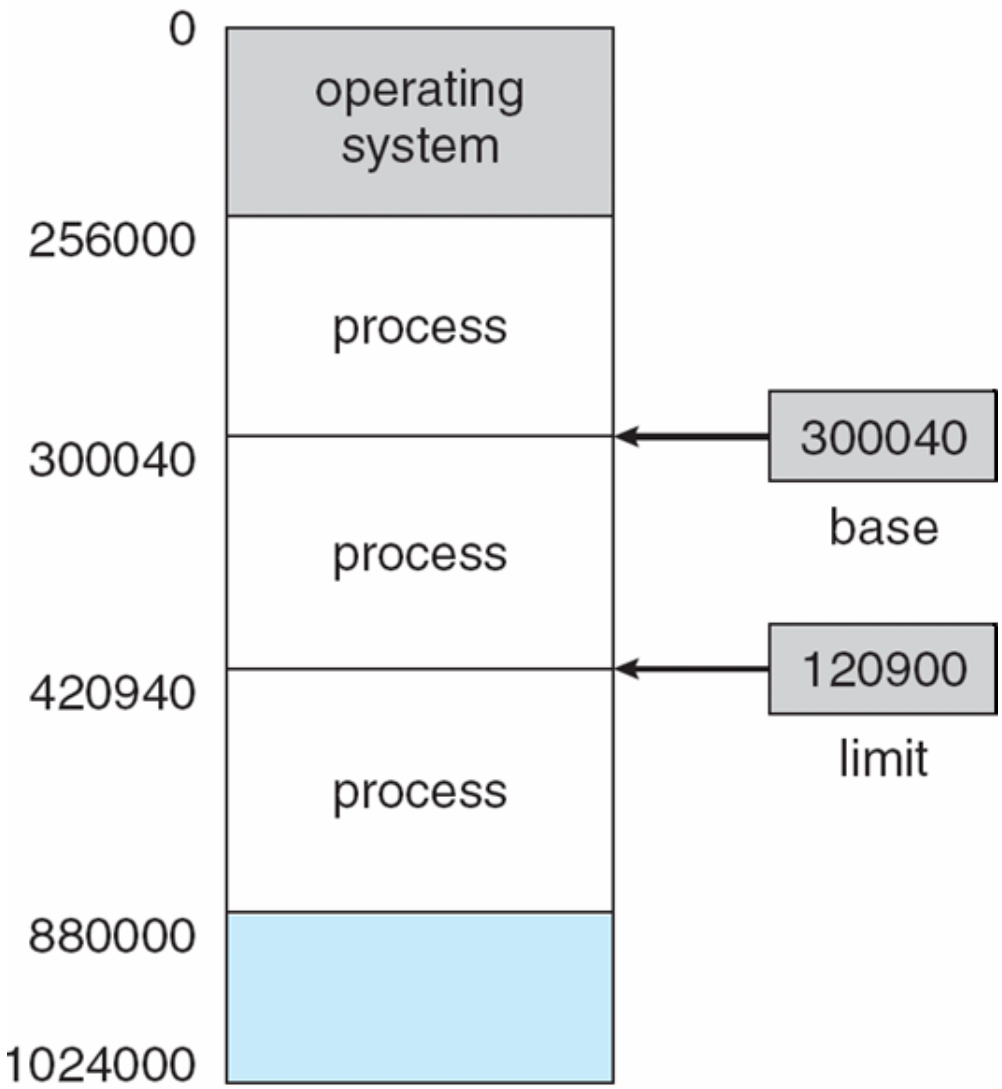
Cuestiones a considerar

- ✓ El programador no sabe dónde se alojará el código de su programa en cada diferente ejecución y tampoco conoce qué otros programas (código) residirán en la memoria en el momento de la ejecución.
- ✓ Es imposible, entonces, comprobar las direcciones absolutas de los programas, puesto que se desconoce la ubicación de ese código en la memoria principal.

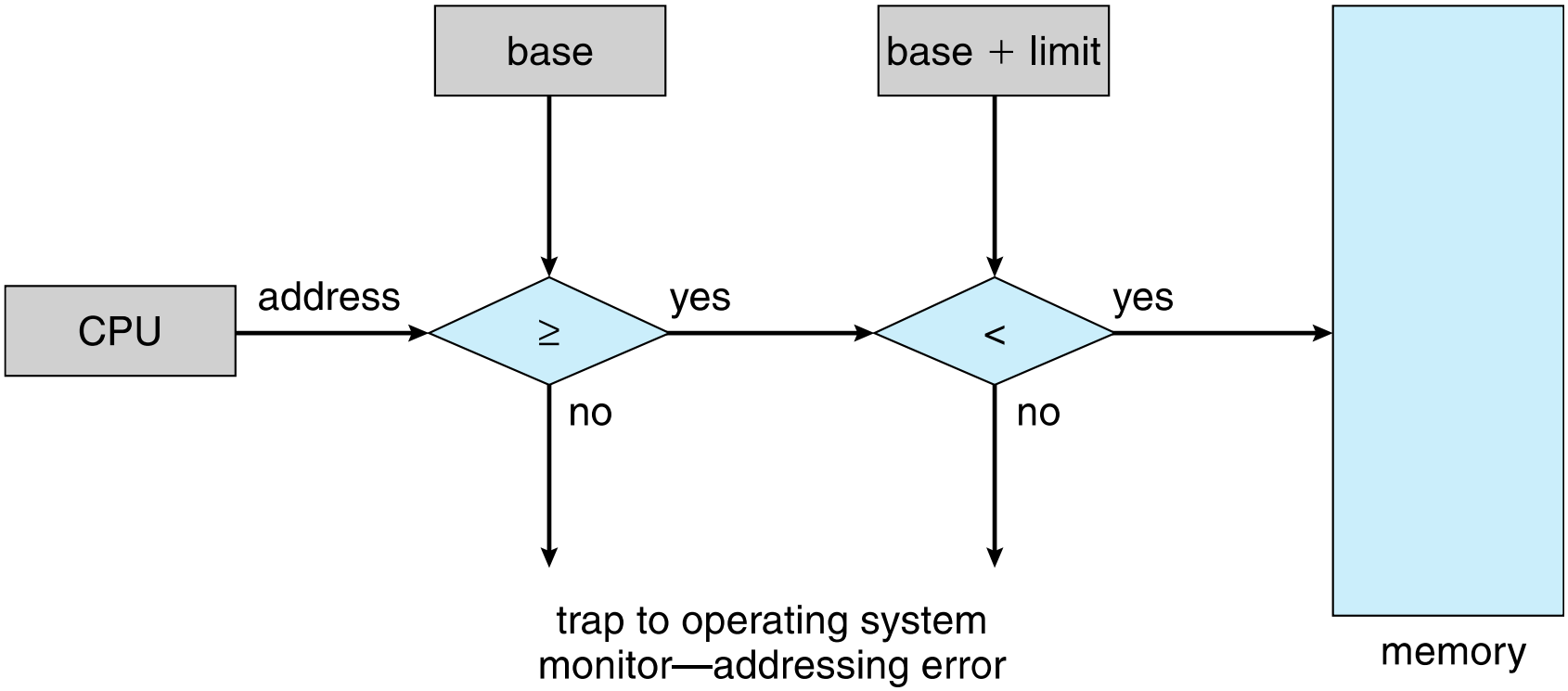
Vinculación de direcciones

La vinculación de direcciones de instrucciones y datos a memoria, puede ocurrir en tres momentos distintos:

- **En tiempo de compilación**: Si la región de memoria se conoce a priori, se puede generar el llamado *código absoluto***.** Si la región cambia, se debe recompilar.
- **En tiempo de carga**: Se debe generar *código relocalizable* si la región de memoria no se conoce en tiempo de compilación.
- **En tiempo de ejecución**: La vinculación se posterga hasta el tiempo de ejecución si el proceso puede moverse de un segmento de memoria a otro. Se necesita soporte de hardware para los mapas de direcciones (registros base y límite)



Espacio de direccionamiento del proceso: entre registros base y límite



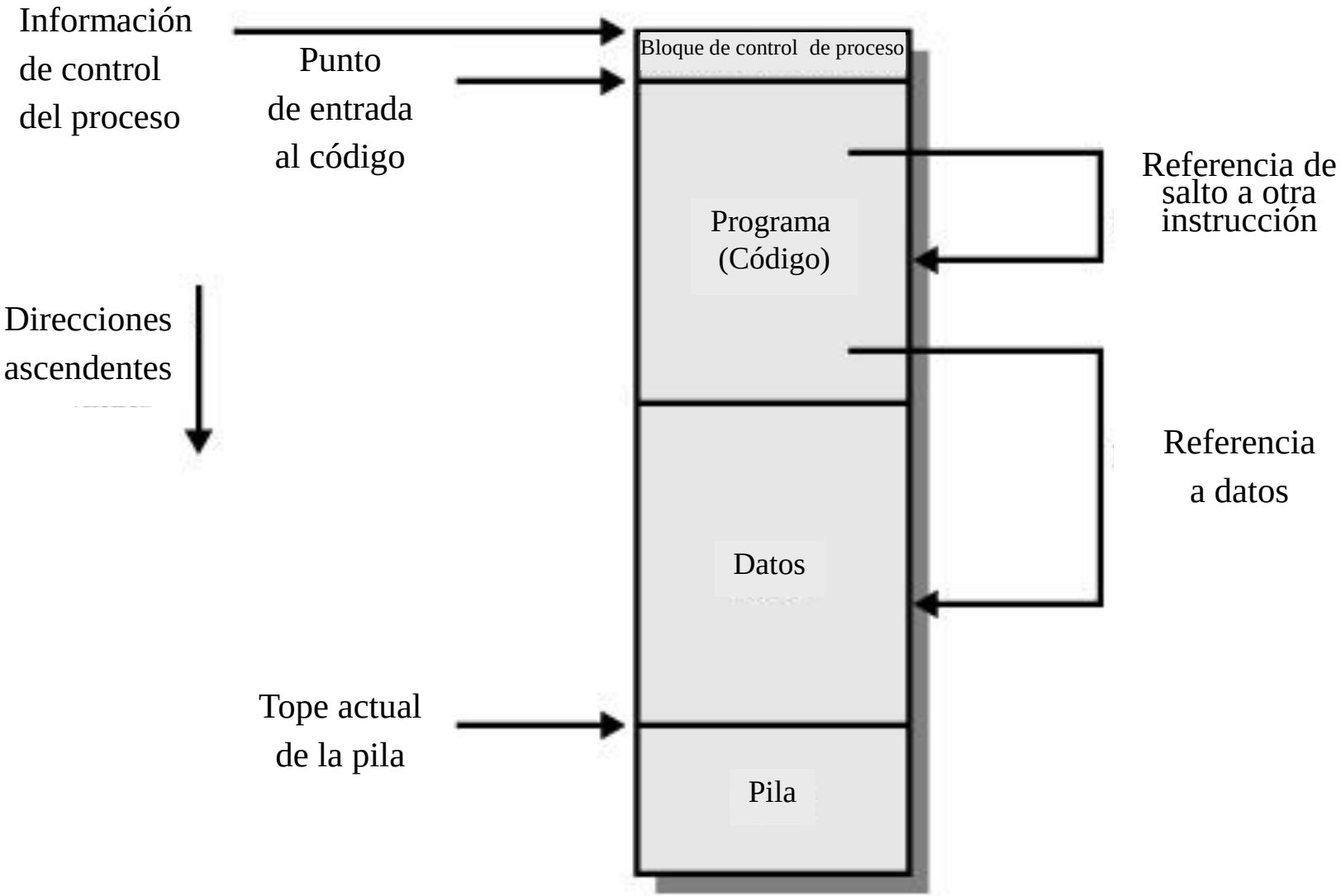
GARANTÍA DE PROTECCIÓN: Verificación de la dirección física

Direcciones Lógicas y Físicas

- **Direcciones lógicas**: son generadas por la CPU. También se denomina espacio virtual de direcciones.
- **Direcciones físicas**: son las direcciones vistas por la unidad de memoria.
- Las direcciones físicas y lógicas son las mismas en los esquemas de vinculación de tiempo de compilación y tiempo de carga.
- Las direcciones físicas y lógicas difieren en el esquema de vinculación en tiempo de ejecución

Referencias

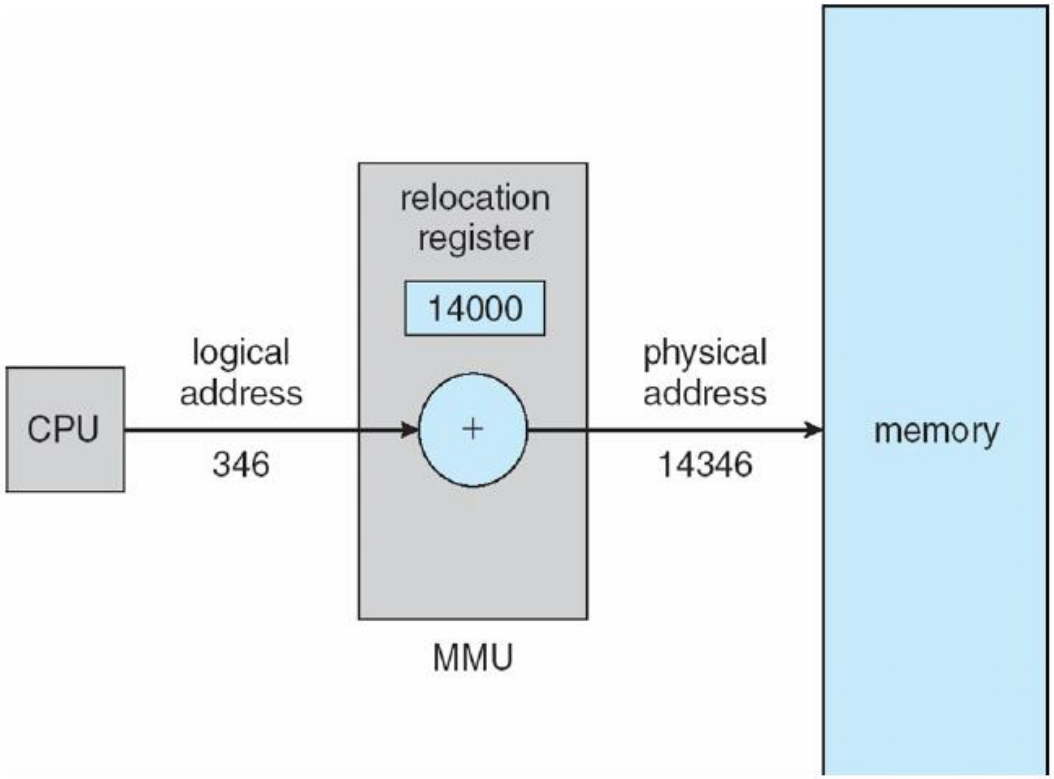
- Se deben traducir las referencias a la memoria encontradas en el código a las direcciones físicas reales
  - Referencias al mismo código
  - Referencias a datos y a pila
- Es conveniente permitir el acceso de varios procesos a una misma zona de la memoria principal (compartida) en lugar de que cada proceso tenga su propia copia.



Direccionamiento desde el proceso

Traducción de lógicas a físicas

- La Memory-Management Unit (MMU) es un dispositivo de HW que transforma direcciones virtuales en direcciones físicas.
- A cada dirección generada por el proceso usuario se le suma el valor del registro de relocalización.
- Se logra que un programa usuario (código) referencie siempre sólo direcciones lógicas.



Memory-Management Unit (MMU)

Direccionamiento con reubicación

- ✓ Dirección lógica:
  - Es una referencia a una posición de memoria independiente de la asignación actual de datos a la memoria.
  - Se debe hacer una traducción a una dirección física.
- ✓ Dirección relativa:
  - La dirección se expresa como una posición relativa a algún punto conocido.
- ✓ Dirección física:
  - La dirección absoluta o la posición real en la memoria principal.

Registros necesarios

- ✓ Registro base:
  - Se carga con la dirección física de inicio del proceso.
- ✓ Registro límite:
  - Indica la dirección física final del proceso.
- ✓ Se suma el valor del registro base a la dirección relativa del código para obtener una dirección absoluta y se compara con el valor del registro límite (protección).
  - Si la dirección no está dentro de los límites, se generará una interrupción en el sistema operativo.

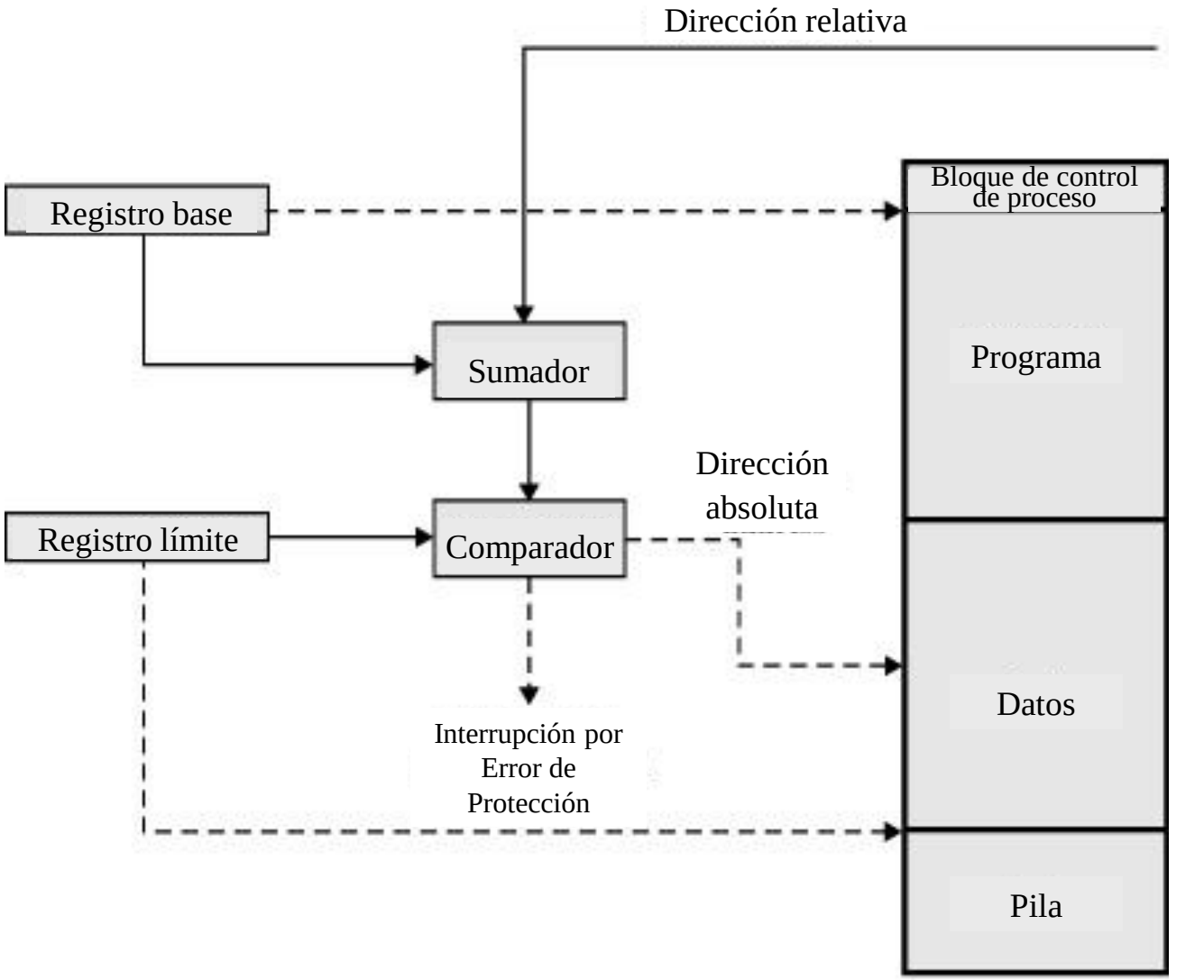
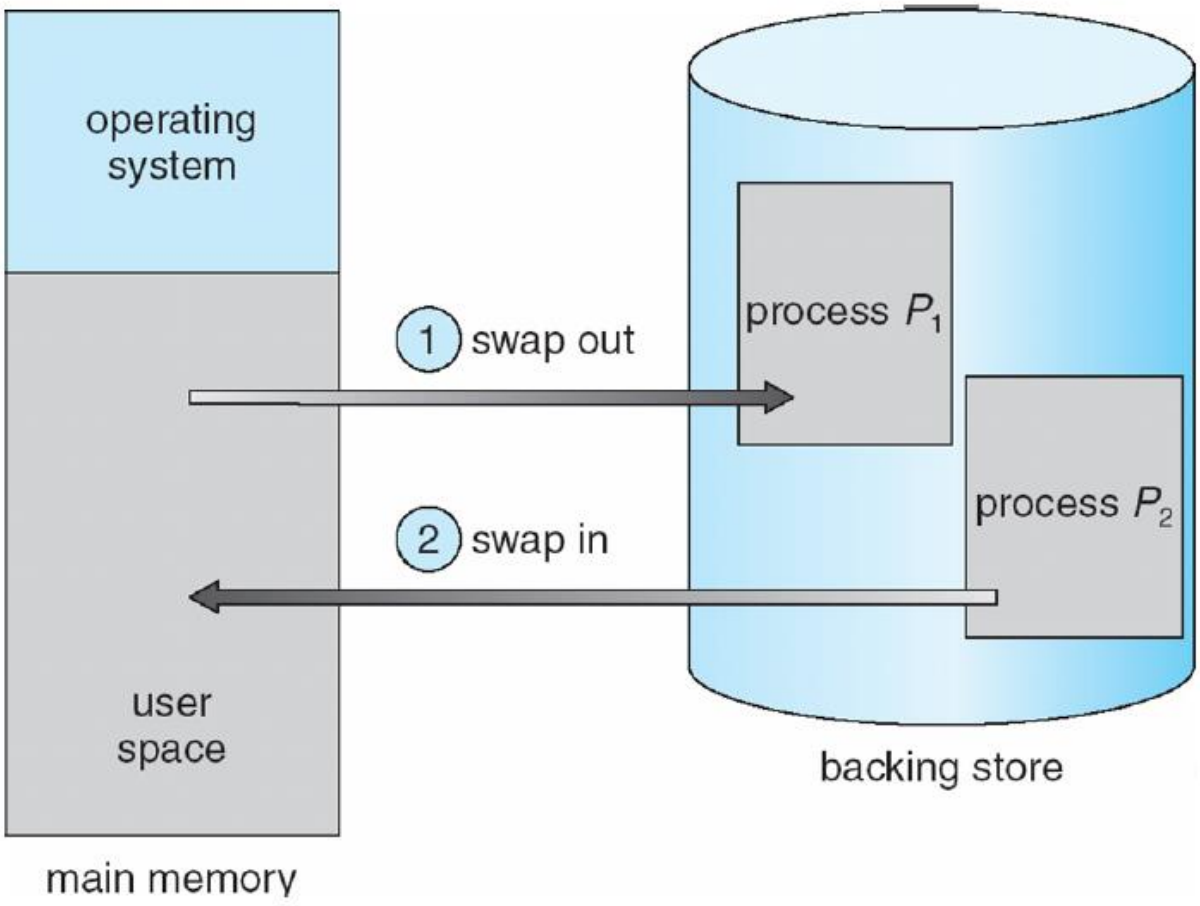


Imagen de un proceso en la memoria principal

Cálculo de la dirección física y verificación de la protección

Intercambio

- El pasaje de procesos de memoria al dispositivo de almacenamiento secundario (backing store) se llama Swap Out.
- El pasaje de procesos del dispositivo de almacenamiento secundario (backing store) a la memoria se llama Swap In.
- Al conjunto de Swap Out + Swap In se lo conoce genéricamente como Swapping


operating system
process P_1
1 swap out
process P_2
2 swap in
user space
backing store
main memory

Swapping

Principios de gestión

- El **Principio de Continuidad** dice que un proceso debe cargarse contiguo en memoria (en direcciones consecutivas)

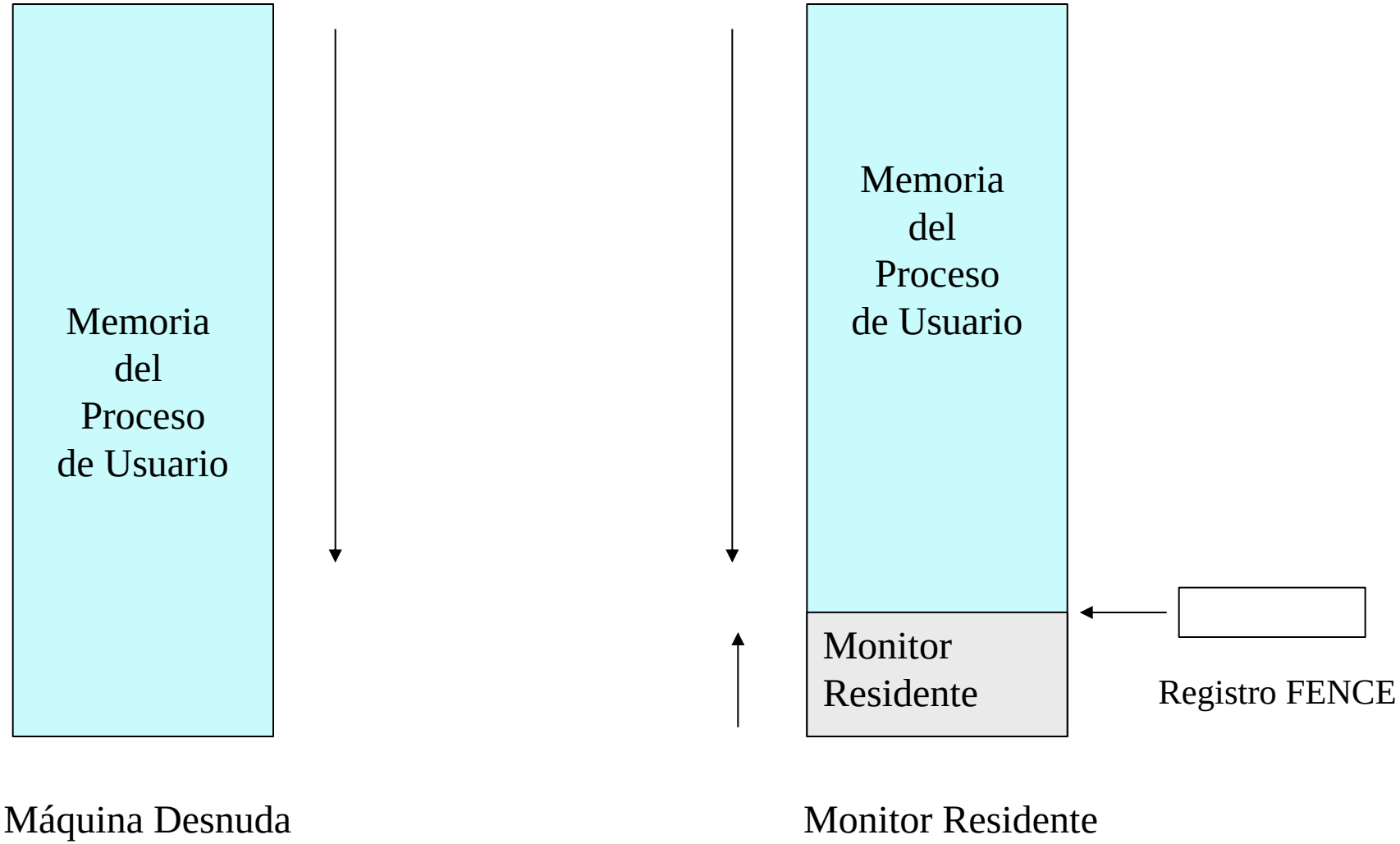
- El **Principio de Completitud** dice que un proceso debe cargarse completo en memoria

Máquina Desnuda – Monitor Residente

- ✓ En los principios de la computación (hasta 1960), no había sistemas operativos. La memoria era una sábana de celdas contiguas sin division física ni lógica.
- ✓ No había soporte para la gestion de la memoria. El proceso debía controlar los dispositivos.
- ✓ El programador, administrador y usuario eran la misma persona y era quien administraba la memoria.

Máquina Desnuda – Monitor Residente

- ✓ El Monitor Residente fue el primer esquema de un “gestor de acciones y recursos”. Se encargaba del secuenciamiento de tareas, la gestion de los dispositivos y algunas rutinas que eran de uso recurrente.
- ✓ Fue necesario “dividir” la memoria para evitar que el proceso de usuario no escribiera sobre el monitor residente: surge la cuestión de la protección y el Registro FENCE.


Memoria
del
Proceso
de Usuario
Memoria
del
Proceso
de Usuario
Monitor
Residente
Registro FENCE
Máquina Desnuda
Monitor Residente

Swapping superpuesto

- ✓ Se divide la memoria en 3 regiones iguales
- ✓ Mientras en una región está el proceso que se está ejecutando, en otra región está el proceso terminado (que se está descargando) y en la tercera región está el proceso nuevo (que se está cargando).
- ✓ Siempre hay solamente un proceso en ejecución (y los registros fences marcan sus fronteras).
- ✓ Hay multiprogramación, pero monotarea.

Swapping superpuesto

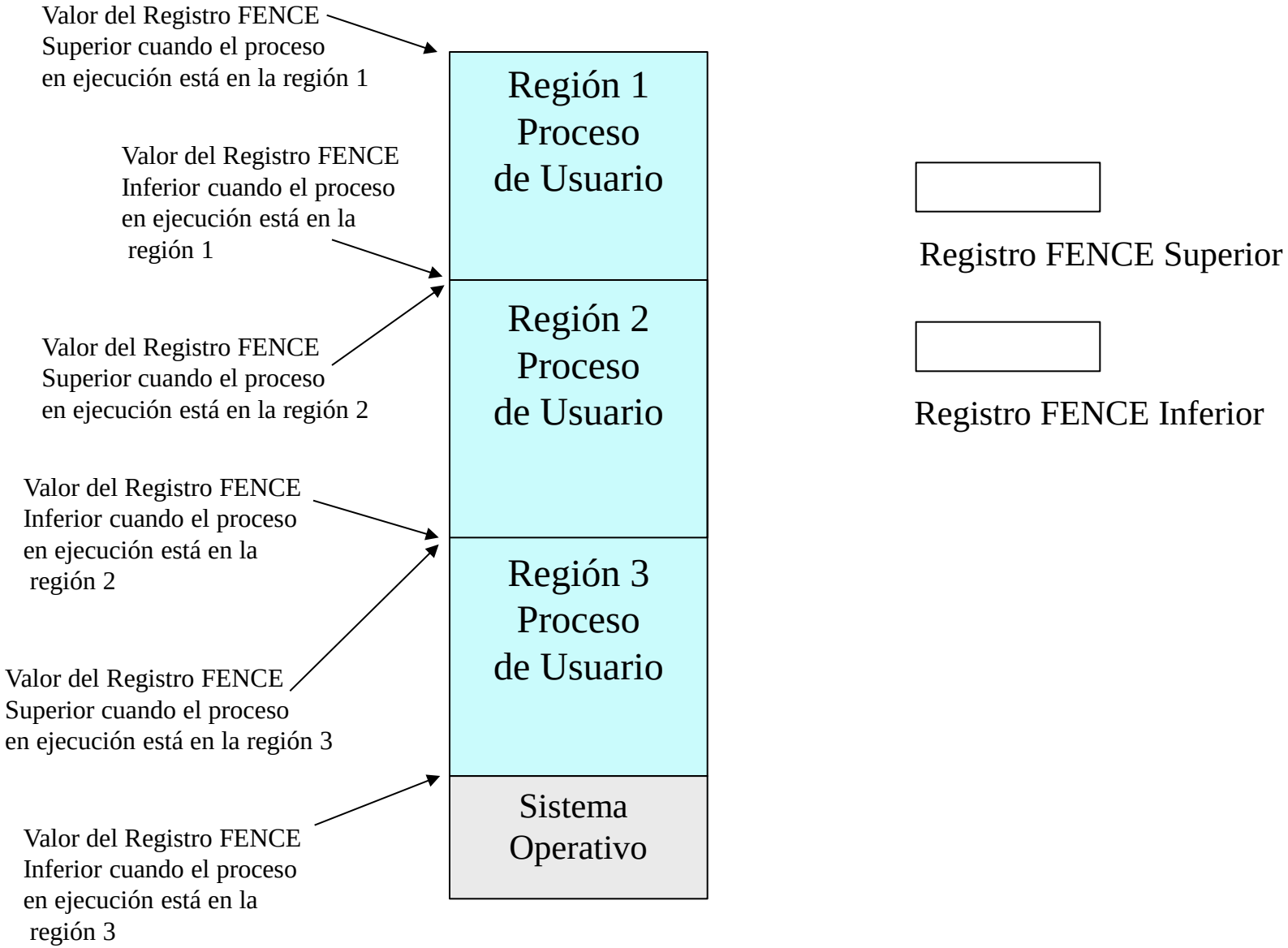
- ✓ Ventaja:

  Se superponen los procesos de carga, descarga y ejecución (se realizan concurrentemente), entonces, se eliminan los tiempos muertos de procesador mientras se cargan y descargan procesos. El procesador siempre está ocupado corriendo un proceso de usuario.

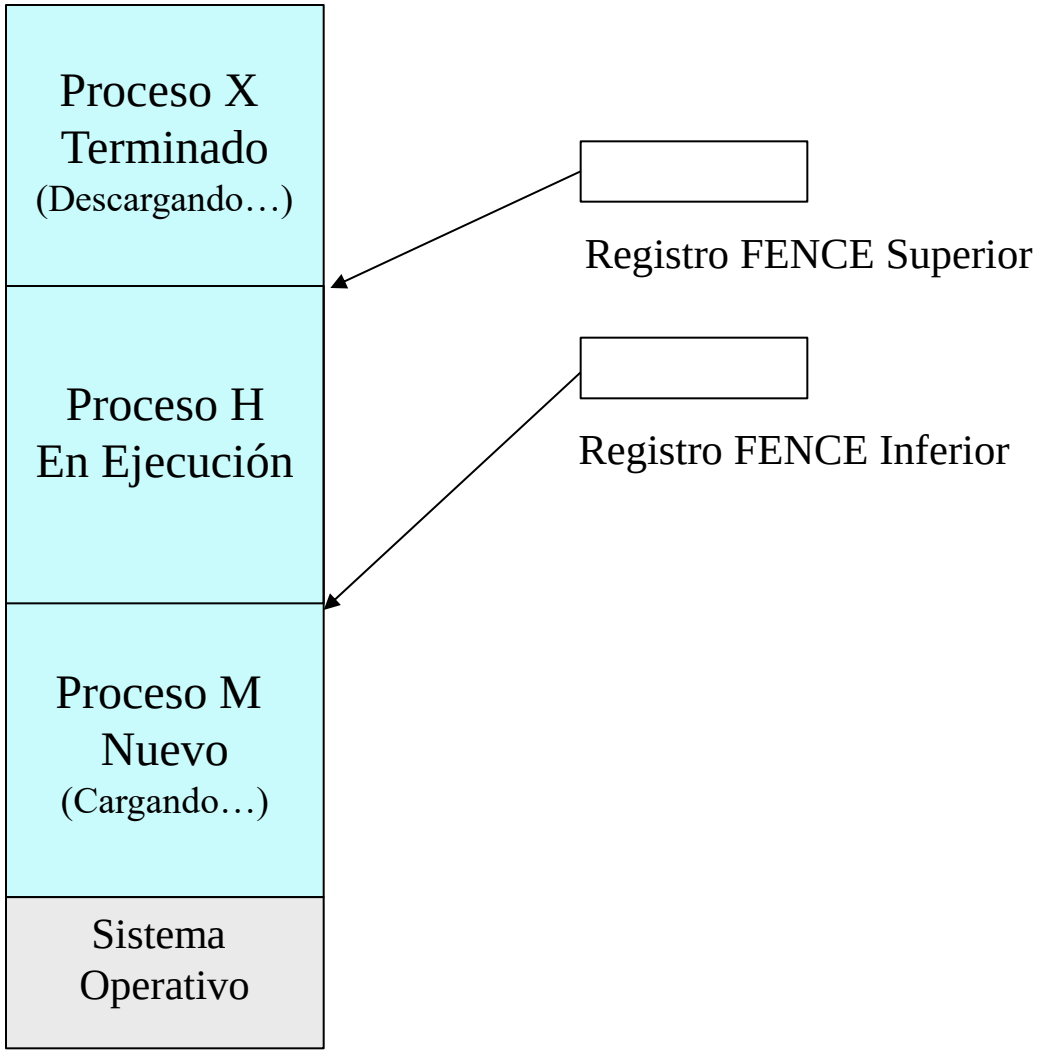
- ✓ Desventajas:

  El tamaño máximo del proceso es menor. Se reduce a un tercio de la memoria física disponible para procesos de usuario.

  Hay fragmentación interna.



Swapping superpuesto (Esquema)



Swapping superpuesto (Ejemplo donde el Proceso H está en ejecución)

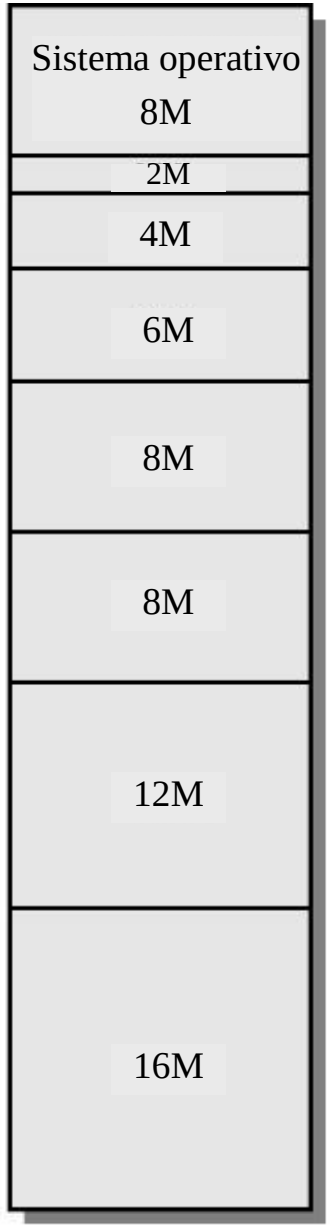
Múltiples particiones de tamaño fijo (MFT)

- ✓ Se divide la memoria en particiones o regiones de igual tamaño (fijo) que alojan diferentes procesos en ejecución (aparece la multitarea).
- ✓ En cualquier partición libre puede cargarse cualquier proceso cuyo tamaño sea menor o igual que el tamaño de la partición.
- ✓ El uso de la memoria es ineficiente. Cualquier programa, sin importar lo pequeño que sea, ocupará una partición completa, generando fragmentación interna.

Múltiples particiones de tamaño fijo (MFT)

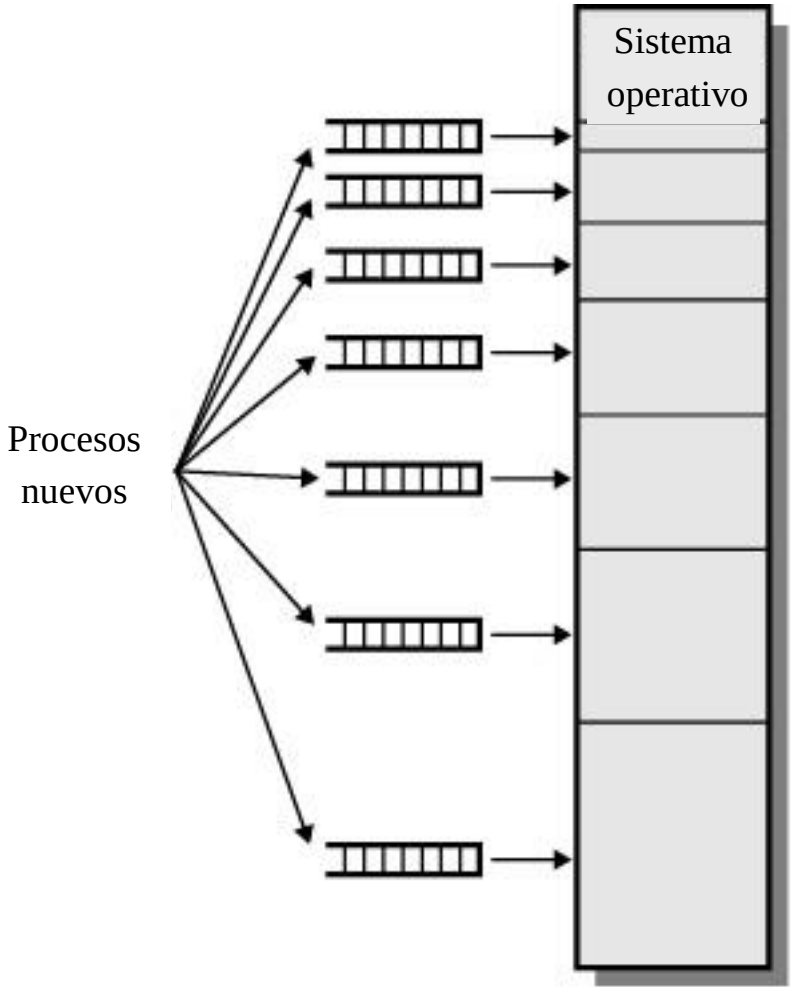
- ✓ El tamaño máximo del proceso es el tamaño de la partición.
- ✓ Las particiones pueden ser todas iguales o diferentes entre sí.
- ✓ Si son todas iguales, con una sola cola es suficiente.
- ✓ Si las particiones son diferentes, conviene implementar múltiples colas para que cada proceso compita por la región en la que mejor ajusta (menos FI) y mayor tamaño máximo.

(a) Particiones de igual tamaño

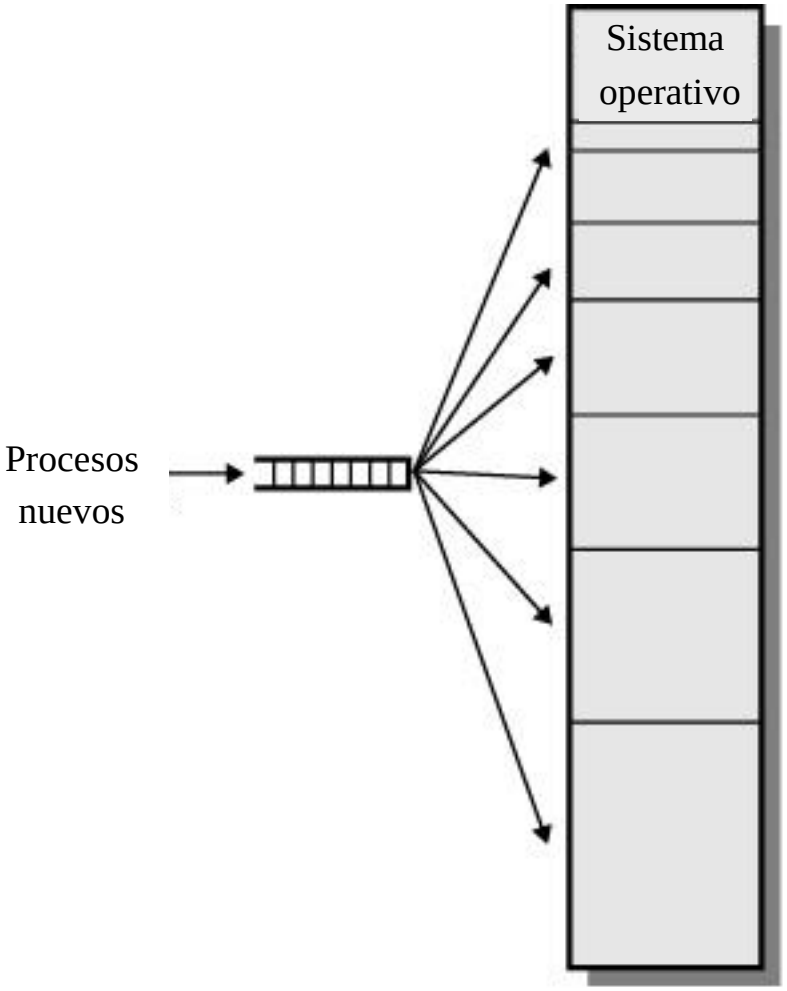


(a) Particiones de distinto tamaño

Múltiples particiones de tamaño fijo



(a) Una cola de procesos por partición

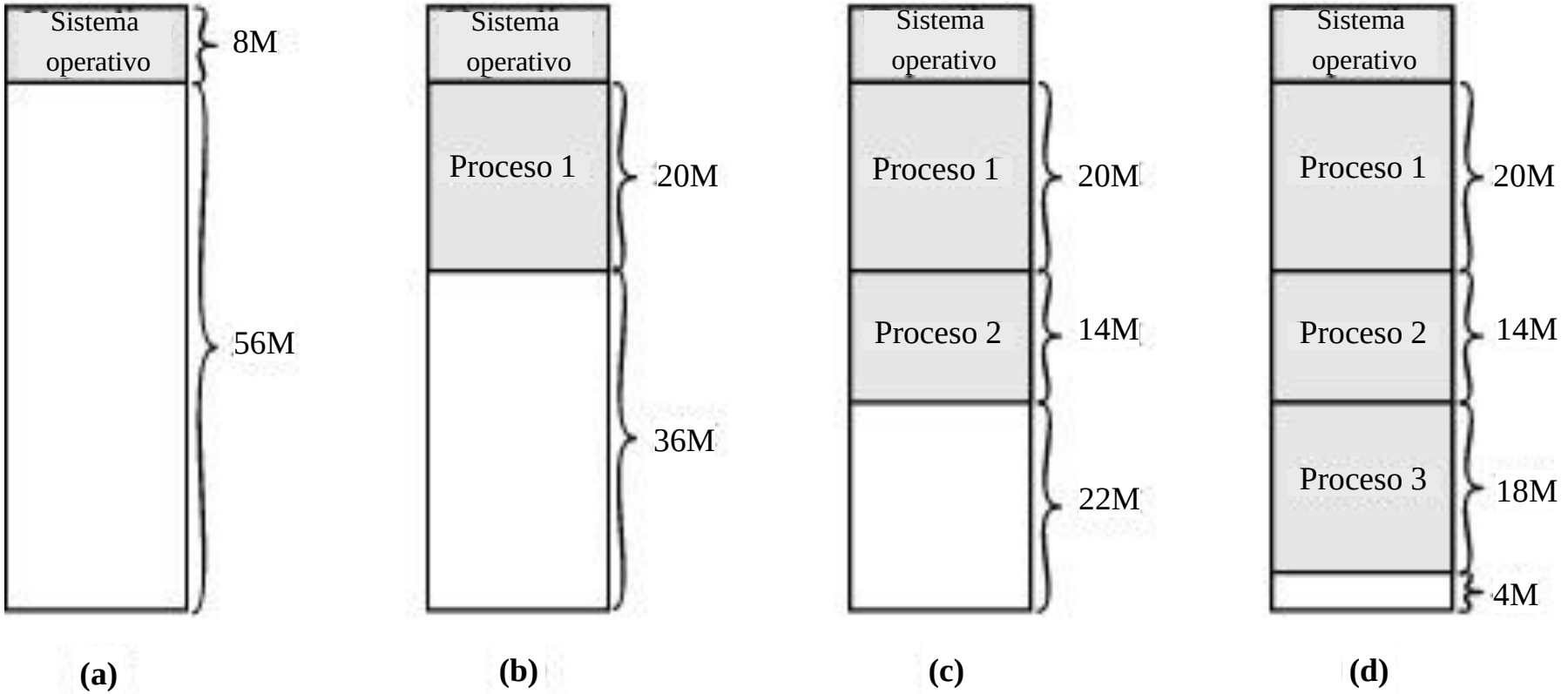


(b) Cola única de procesos

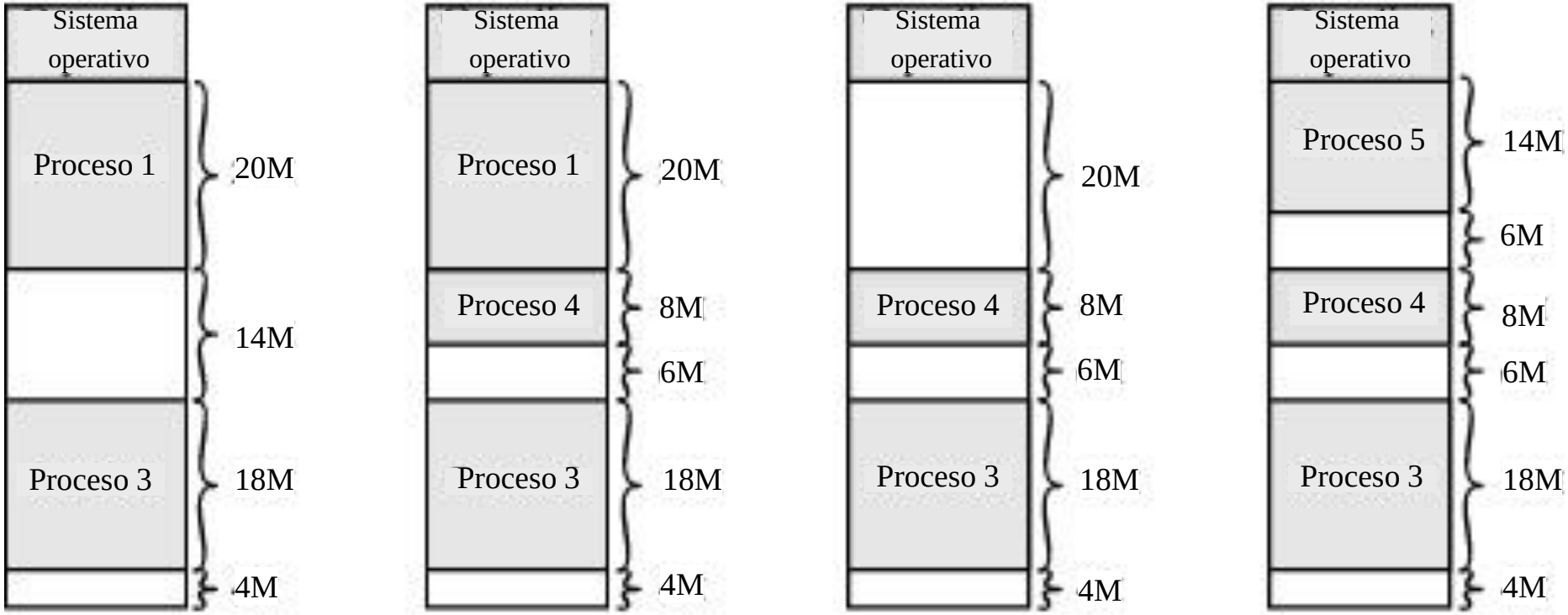
Asignación de memoria con una o múltiples colas

Múltiples particiones de tamaño variable (MVT)

- ✓ Se divide la memoria en particiones variables en número y longitud.
- ✓ Al proceso se le asigna exactamente tanta memoria como necesite.
- ✓ Con el uso, se van produciendo varios huecos en la memoria (Fragmentación Externa).
- ✓ Se debe usar la **Compactación** para desplazar los procesos para que estén contiguos, de forma tal que toda la memoria libre quede junta en un solo bloque. Desventaja para RTOS.



Secuencia de alojamiento de procesos con particiones variables



Secuencia de alojamiento de procesos con particiones variables

Múltiples particiones de tamaño variable (MVT)

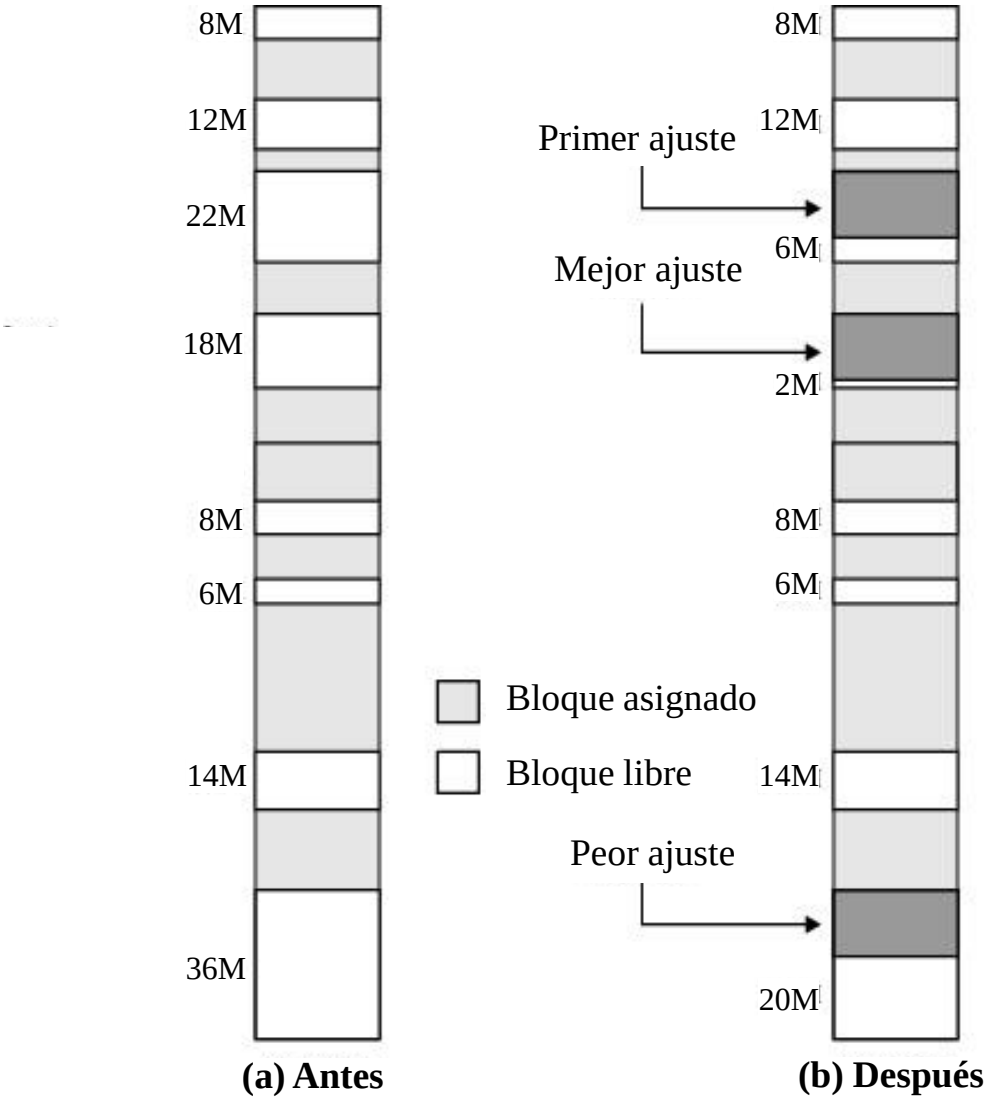
- El sistema operativo debe decidir qué espacio libre (hueco) le asigna al proceso.

- Algoritmo del mejor ajuste (*best-fit*):
  - Elige el bloque de tamaño más próximo al solicitado.
  - Proporciona en general los peores resultados.
  - Basigna el hueco más pequeño para el proceso, garantizando que el fragmento que se deja es lo más pequeño posible y, por lo tanto, se debe compactar más frecuentemente.

Múltiples particiones de tamaño variable (MVT)

- ✓ Algoritmo del primer ajuste (*first-fit*):
  - Es más rápido.
  - Puede tener varios procesos cargados en el extremo inicial de la memoria que es necesario recorrer cuando se intenta encontrar un bloque libre.

- ✓ Algoritmo del peor ajuste (*worst-fit*):
  - Asigna el bloque libre más grande.
  - Minimiza la Fragmentación Externa.



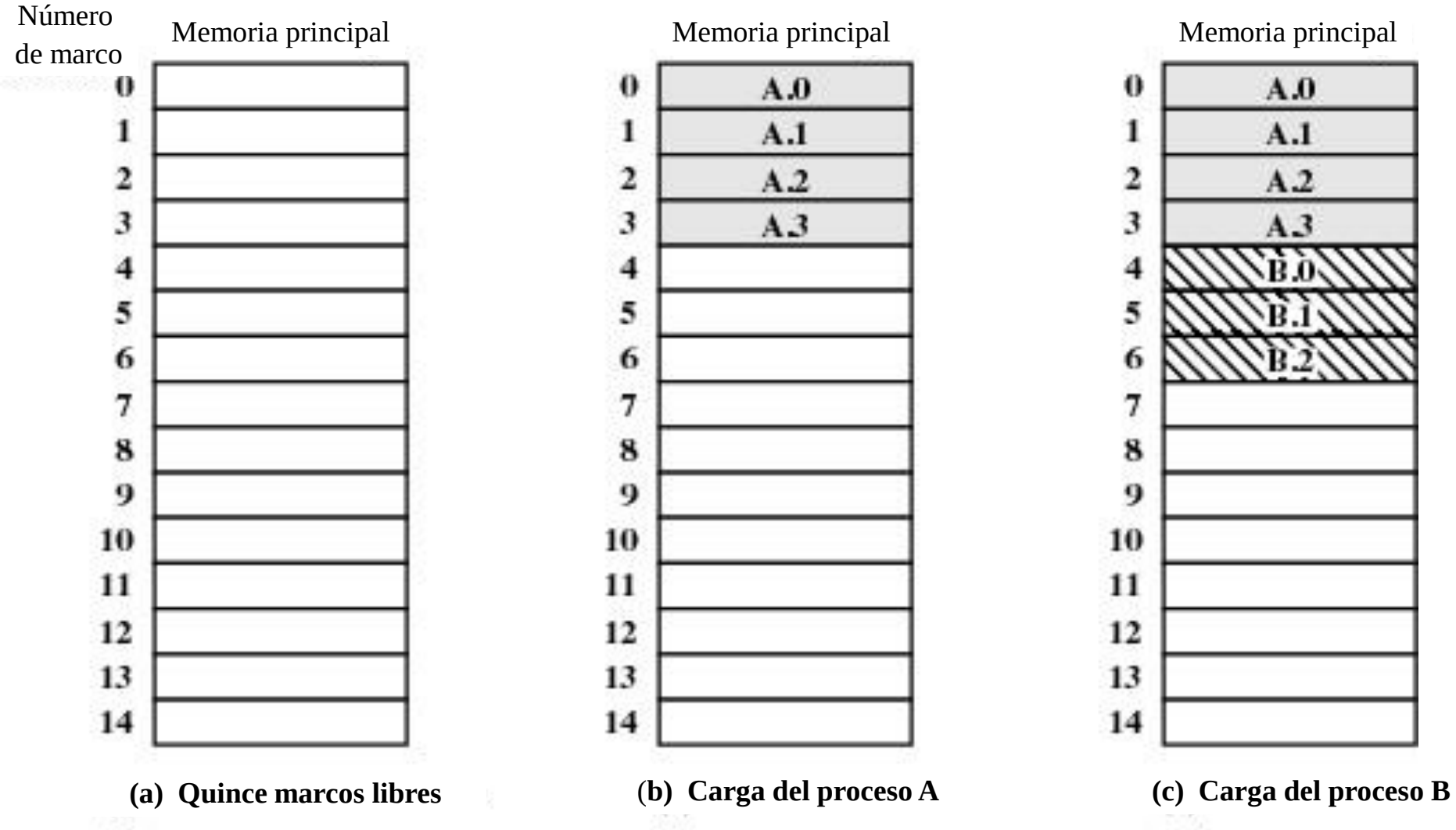
Asignación del bloque según los diferentes algoritmos

Paginación

- ✓ Se rompe el principio de continuidad en MFT !
- ✓ La memoria principal se encuentra dividida en trozos iguales de tamaño fijo **(Marco/Frame)** y cada proceso en pequeños trozos de tamaño fijo **(Página/Page)**.
- ✓ El sistema operativo mantiene una tabla de páginas para cada proceso:
  - Muestra la posición del marco de cada página del proceso.
  - La dirección de la memoria consta de un número de página y de un desplazamiento dentro de la página.

Paginación

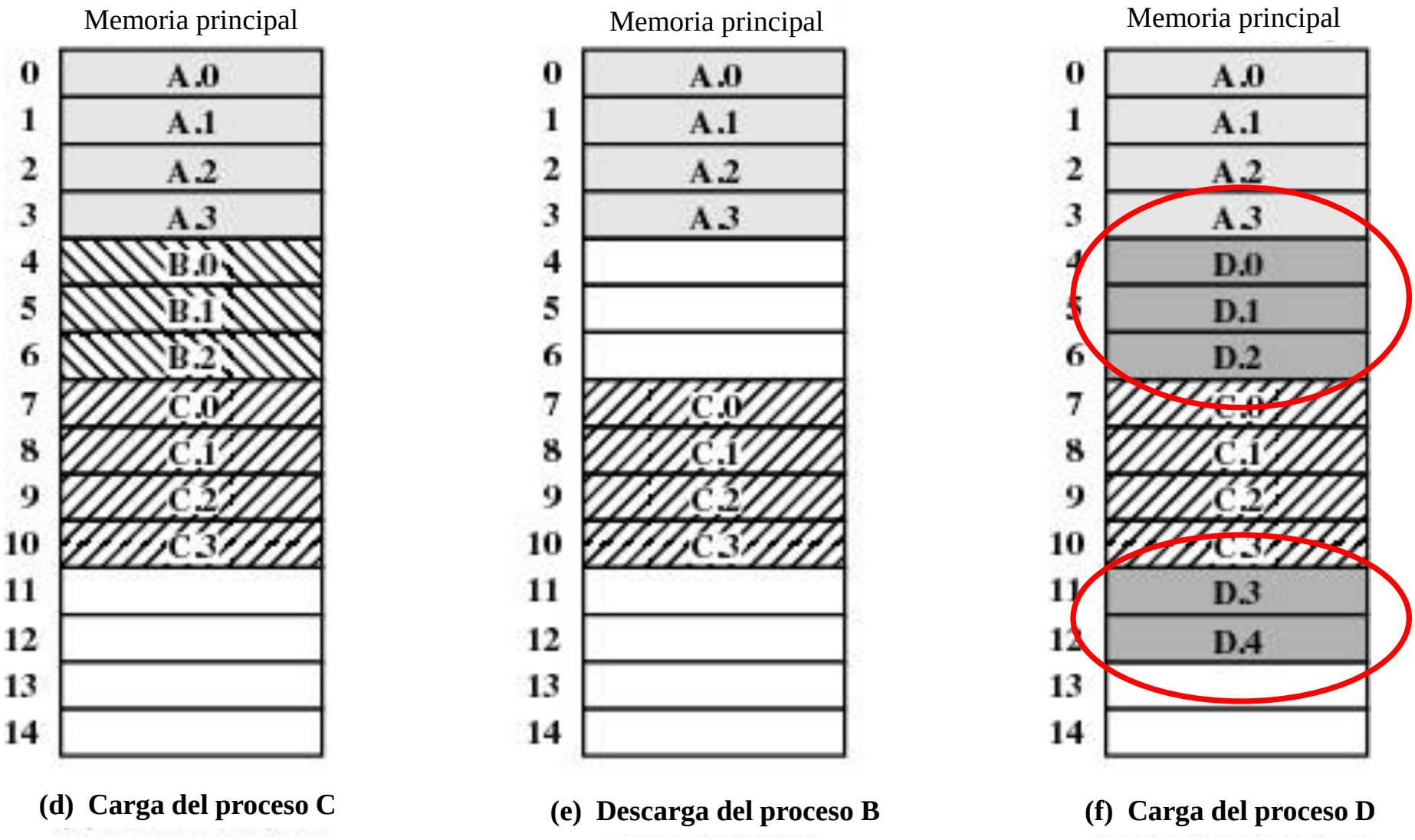
- ✓ Todos los frames libres se pueden ocupar: no hay Fragmentación Externa.
- ✓ Siempre hay Fragmentación Interna en el ultimo frame de cada proceso (en promedio medio frame). La FI Total será: *Tamaño del Frame * N * 0,5*
- ✓ Cuanto más grande sea el tamaño del Frame, habrá mayor FI.
- ✓ El tamaño máximo del proceso que se puede correr es equivalente a toda la memoria física.
- ✓ El N máximo es la cantidad de frames.



(a) Quince marcos libres **(b) Carga del proceso A** **(c) Carga del proceso B**

Paginación: Asignación de páginas de procesos a frames libres.

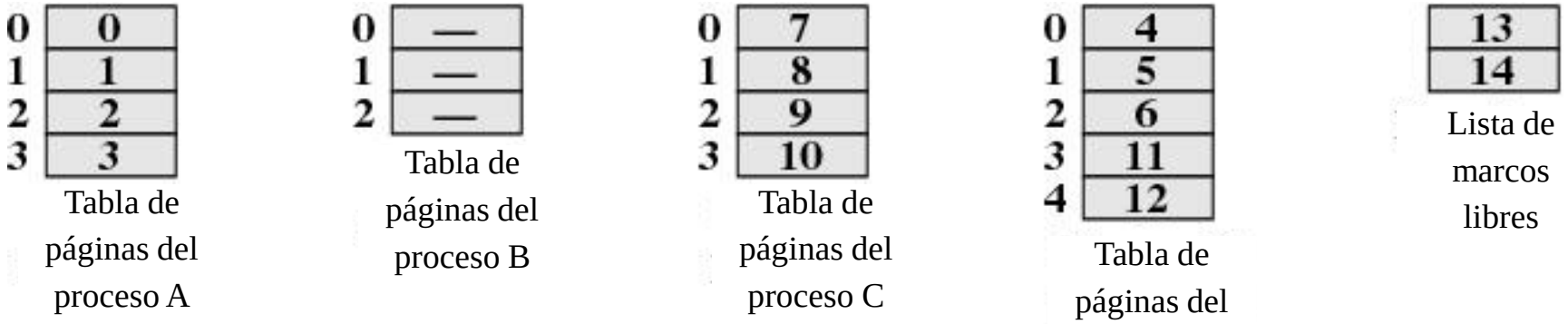
Las páginas de los procesos no se alojan en forma contigua, sino dispersas en diferentes frames



Paginación: Asignación de páginas de procesos a frames libres.

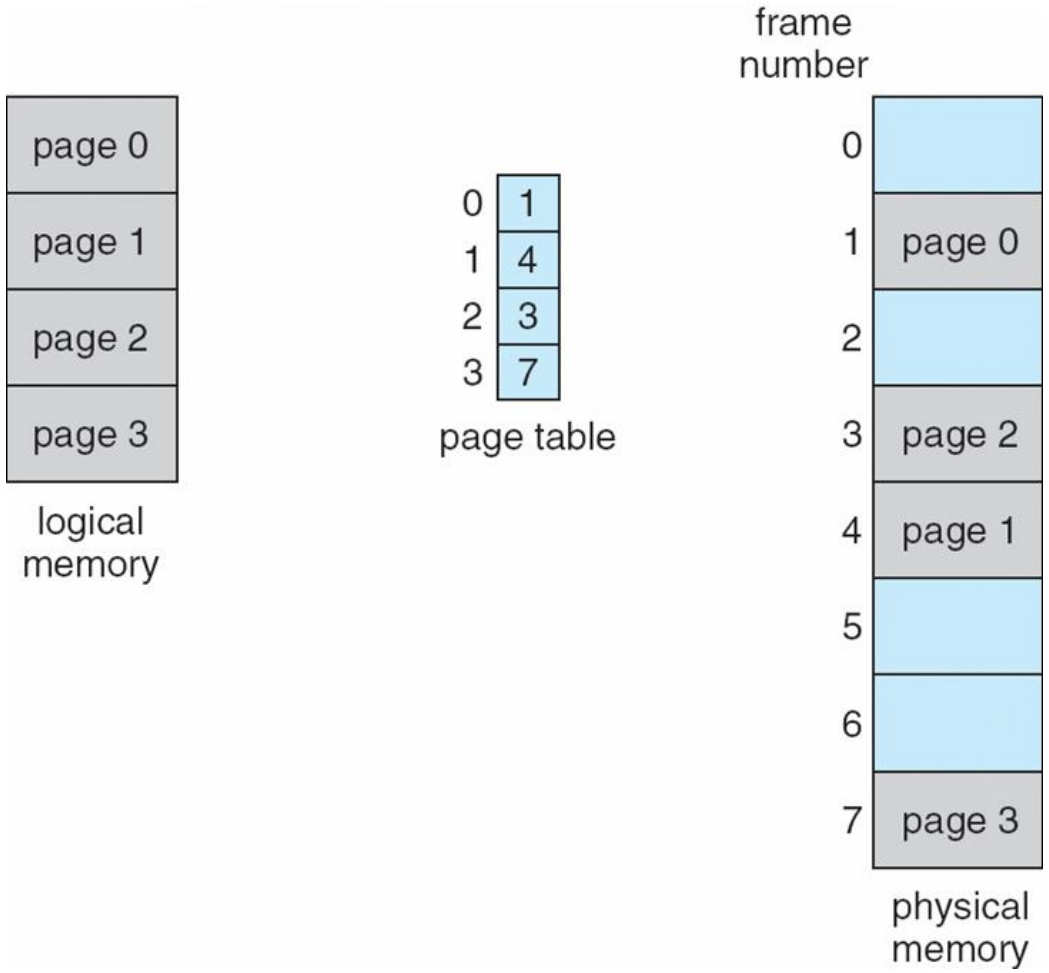
Tabla de páginas

- ✓ Cada proceso tiene su propia tabla de páginas en su PCB. Es un arreglo de tipo integer, donde cada elemento guarda el frame correspondiente a la entrada.
- ✓ El SO tiene una en el contexto de operación del proceso donde se mapea la tabla de páginas del proceso current, en tiempo de context switch, para mayor perfomance.

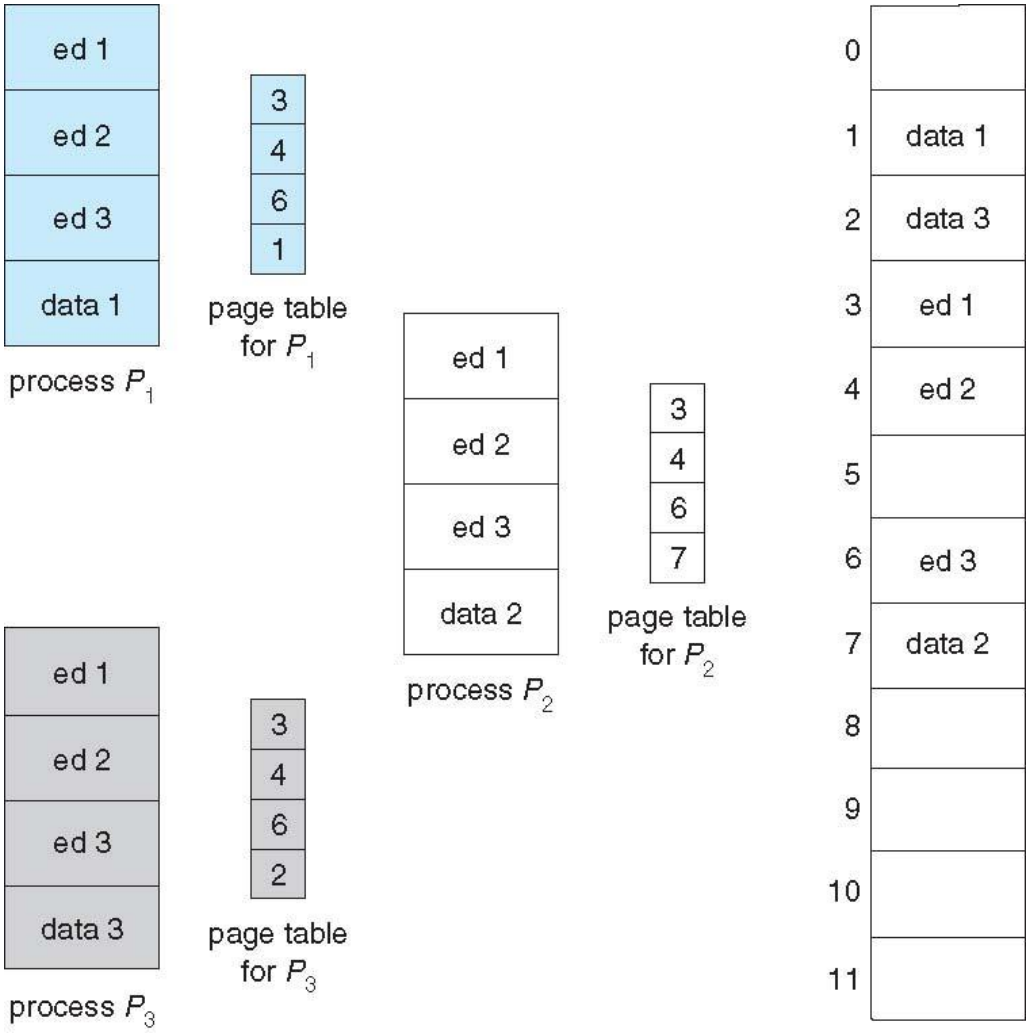


Tablas de páginas

Tabla de páginas

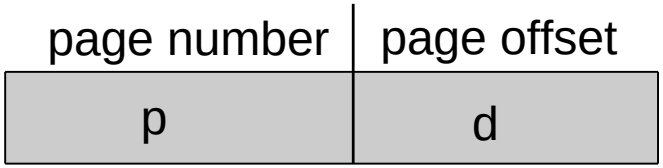


Campartición de páginas



Direcciones

- Cada dirección es expresada como
  - Un **Número de página (p)** que es usada como un índice para ingresar a la tabla de páginas y encontrar el frame donde está alojada esa página en memoria física
  - Un **Desplazamiento de página (d)** que es combinado con la dirección base para definir la dirección física de la referencia.



Cálculo de la dirección

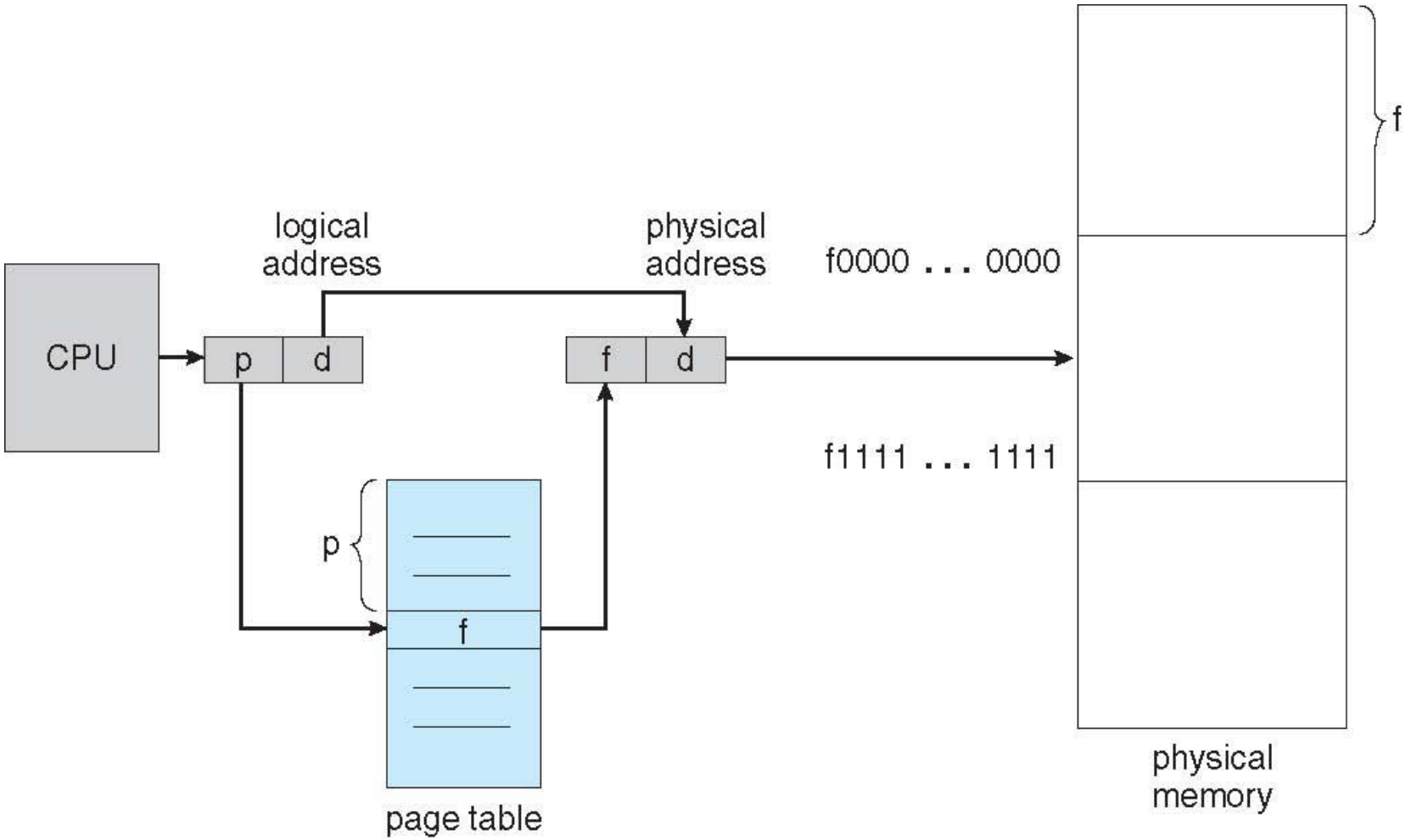
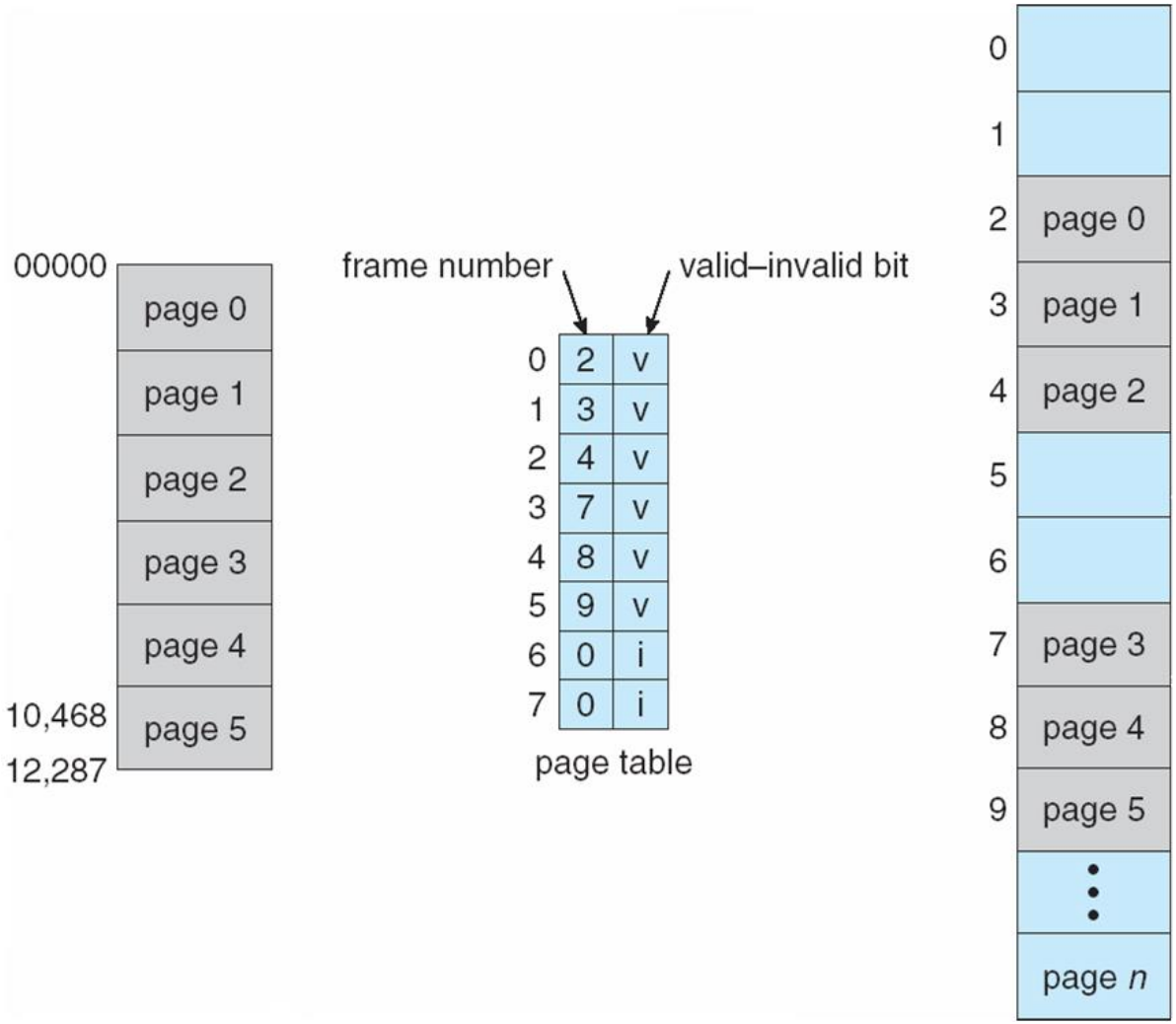


Tabla de páginas del SO

- ✓ Es de acceso rápido.
- ✓ Tiene una cantidad finita de entradas que se setean en el Context Switch (IN) con la tabla de páginas del proceso que se carga en el procesador.
  - Cuando el proceso tiene menos páginas que las entradas de esta TP, se deben marcar la entradas inválidas.
  - Cuando el proceso tiene mas páginas que las entradas de esta TP, se mantiene una parte de la tabla en memoria (rebalse) y se incrementa el tiempo de acceso efectivo a memoria

Tabla de páginas del SO



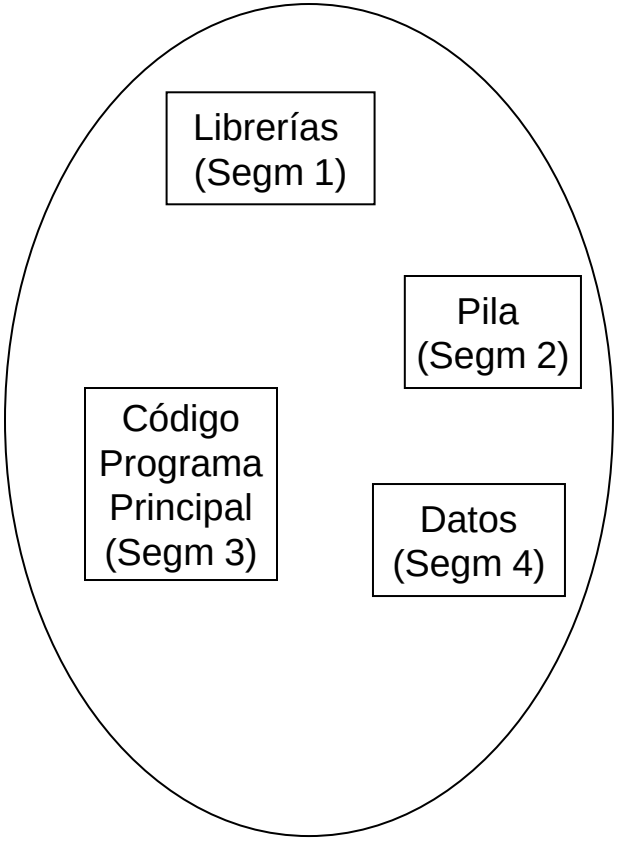
Segmentación

- ✓ Se rompe el principio de continuidad en MVT !
- ✓ La memoria principal se la considera una sábana sin divisiones preestablecidas.
- ✓ Los procesos son divididos en “segmentos” según un **criterio de unidad lógica**, cada uno con su tamaño (existe una longitude máxima de segmento).
- ✓ Los segmentos se van asignando en memoria en los huecos libres que haya, dispersos entre si.

Segmentación

- El esquema de administración de memoria es mas cercano a la vista de memoria del usuario
- Un programa es una colección de segmentos:
  - Programa principal, Procedimiento, Función, Librería, Método, Objeto, Variables locales, Variables globales, Bloque común, Pila, Tabla de símbolos, Matrices, etc.

Segmentación



Espacio de imagen del Usuario

(Criterio de Unidad lógica)

Espacio de memoria física

Segmentación

- ✓ No hay Fragmentación Interna en los segmentos.
- ✓ Todos los huecos libres se pueden ocupar: hay Fragmentación Externa.
- ✓ Se puede generar un solo hueco si se recurre a la Compactación
- ✓ La Compactación es menos probable que en MVT (es más fácil cargar un proceso dividido en varios segmentos que en un solo bloque contiguo)

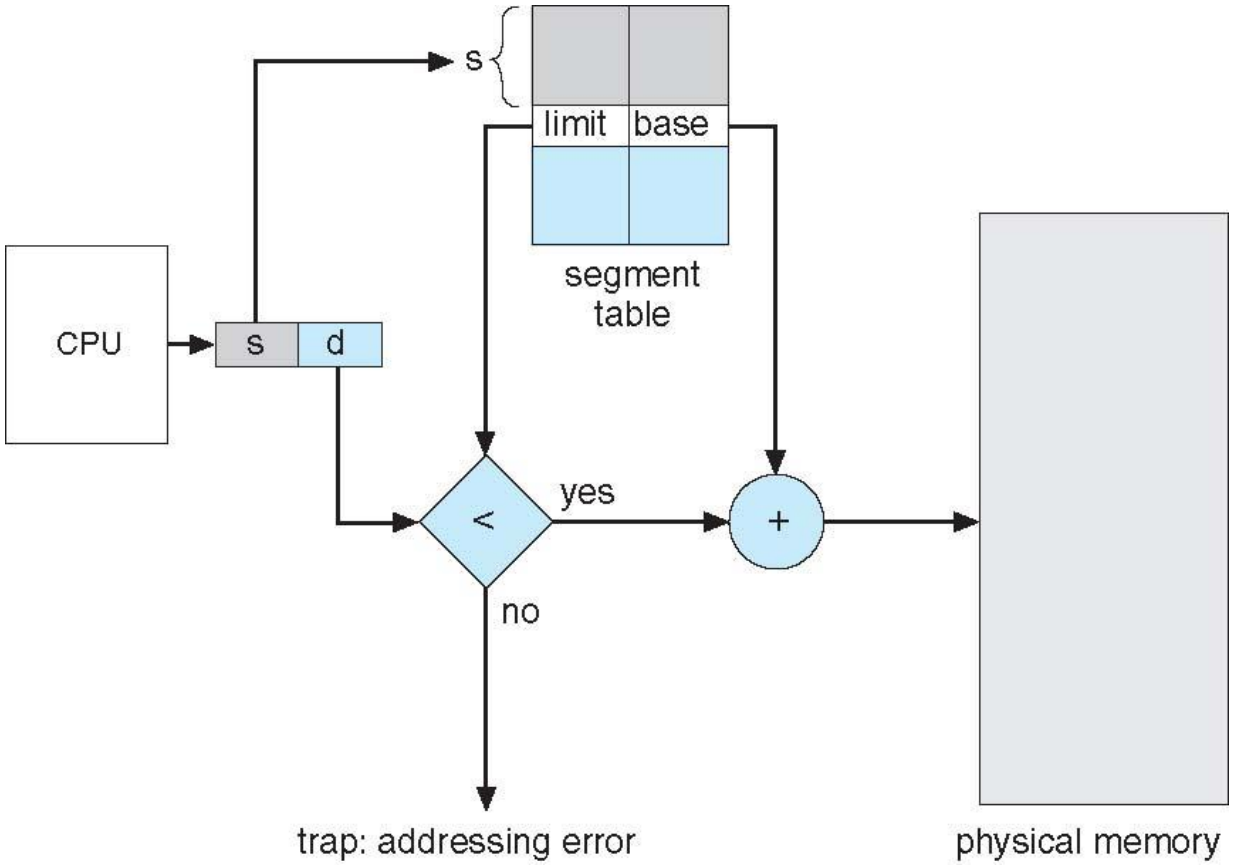
Tabla de segmentos

- ✓ Cada proceso tiene su propia tabla de segmentos en su PCB. Es un arreglo con dos campos (base, límite), donde cada elemento guarda la dirección base del segmento y su tamaño para cada entrada. Es mas “cara” que la Tabla de páginas (por su tamaño).
- ✓ El SO tiene una en el contexto de operación del proceso donde se mapea la tabla de segmentos del proceso current, en tiempo de context switch, para mayor perfomance.
- ✓ Todas las consideraciones hechas para Paginación son válidas para Segmentación en cuanto a la administración de la Tabla de segmentos del SO, bits de validez, rebalse, etc.

Direcciones

- ✓ Cada dirección es expresada como
  - Un **Número de segmento (s)** que es usada como un índice para ingresar a la tabla de segmentos y encontrar la dirección base del segment en la memoria física
  - Un **Desplazamiento de página (d)** que es combinado con la dirección base para definir la dirección física de la referencia.

Cálculo de la dirección



Gracias !