Gestión de memoria

Gestión de memoria. Requisitos

- Reubicación
- Protección
- Compartición
- Organización lógica
- Organización física

Requisitos. Reubicación

- el programador no sabe donde se colocará el programa en memoria cuando se ejecute
- mientras el programa se esté ejecutando, se puede descargar a disco y volver en una dirección de memoria diferente
- las referencias de memoria deben ser traducidas a direcciones reales de memoria

Direccionamiento

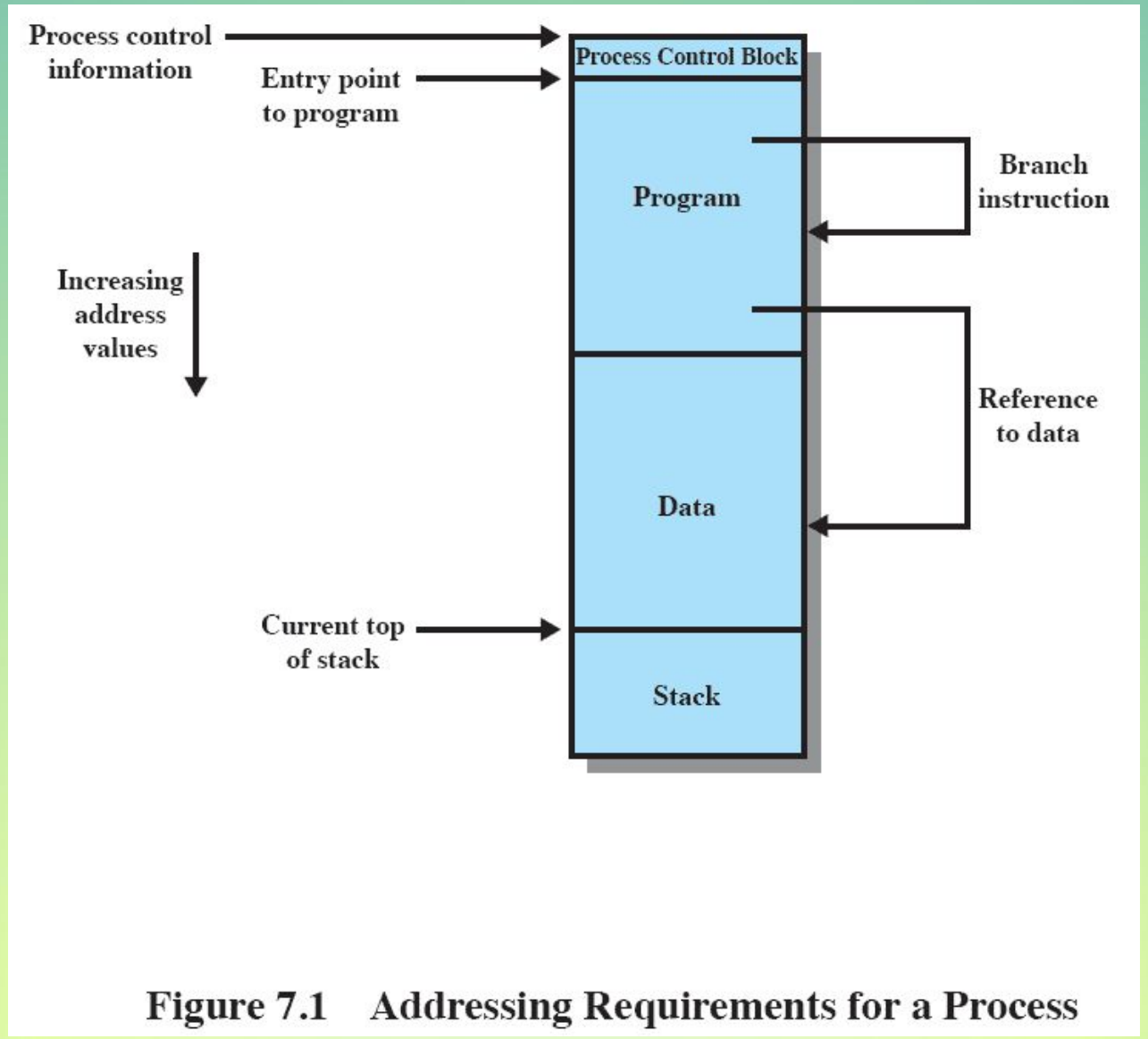


Figure 7.1 Addressing Requirements for a Process

Requisitos. Protección

- los procesos no deberían poder referenciar localidades de memoria de otros procesos sin permiso
- imposible chequear direcciones dentro de los programas puesto que se pueden reubicar
- se debe chequear durante la ejecución

Requisitos. Compartición

- permitir a varios procesos acceder a la misma porción de memoria
- Ejemplos:
  - acceder a la misma copia del programa en lugar de tener su propia copia
  - acceder a una estructura de datos compartida

Requisitos. Organización lógica

- los programas se escriben en módulos
- diferentes grados de protección para los distintos módulos de un programa (solo lectura, solo ejecución)
- compartición de módulos

Requisitos. Organización física

- tarea de mover información entre los por lo menos dos niveles de memoria: principal y secundaria
- no es deseable dejar la responsabilidad al programador
  - La memoria principal disponible para un programa y sus datos puede no ser suficiente
    - Técnica de superposición se malgastaba tiempo del programador
  - En un entorno multiprogramado, el programador no conoce en tiempo de codificación cuánto espacio estará disponible o dónde

Gestión de Memoria

Particiones

Técnicas de gestión de memoria

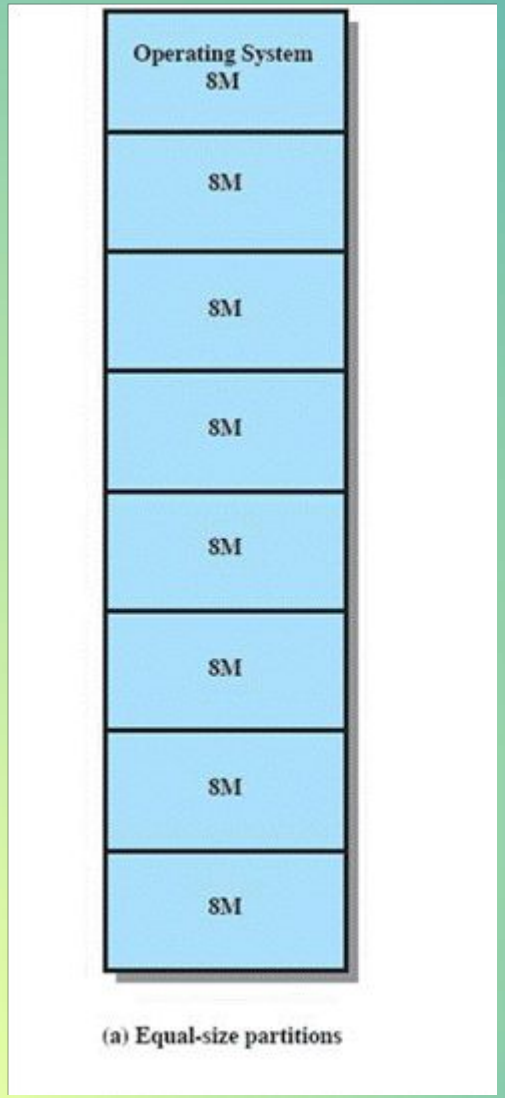
- Particionamiento fijo
- Particionamiento dinámico

- Paginación sencilla
- Segmentación sencilla

- Paginación con memoria virtual
- Segmentación con memoria virtual

Particionamiento fijo

- División de la memoria en particiones de tamaño fijo
  - Se puede colocar cualquier proceso cuyo tamaño sea menor o igual que el de la partición
- Si todas las particiones están llenas el SO puede sacar a un proceso de cualquiera de las particiones (swap) y cargar otro.



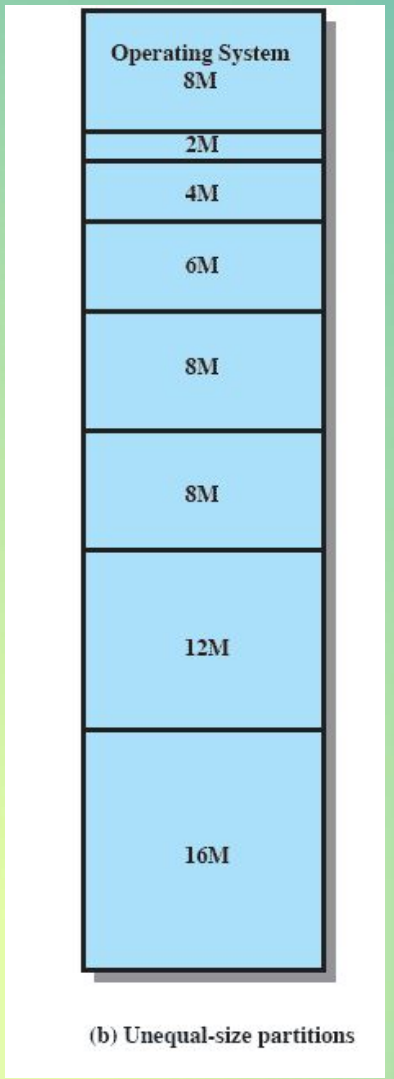
(a) Equal-size partitions

Particionamiento fijo. Problemas

- Un programa más grande que una partición
  - Uso de técnica de superposición
- Uso de la memoria ineficiente. Cualquier programa, sin importar lo pequeño que fuera, ocupa una partición entera. Esto se llama **fragmentación interna.**

Mejora: particiones fijas de tamaño diferente

- No soluciona el problema pero lo mejora un poco



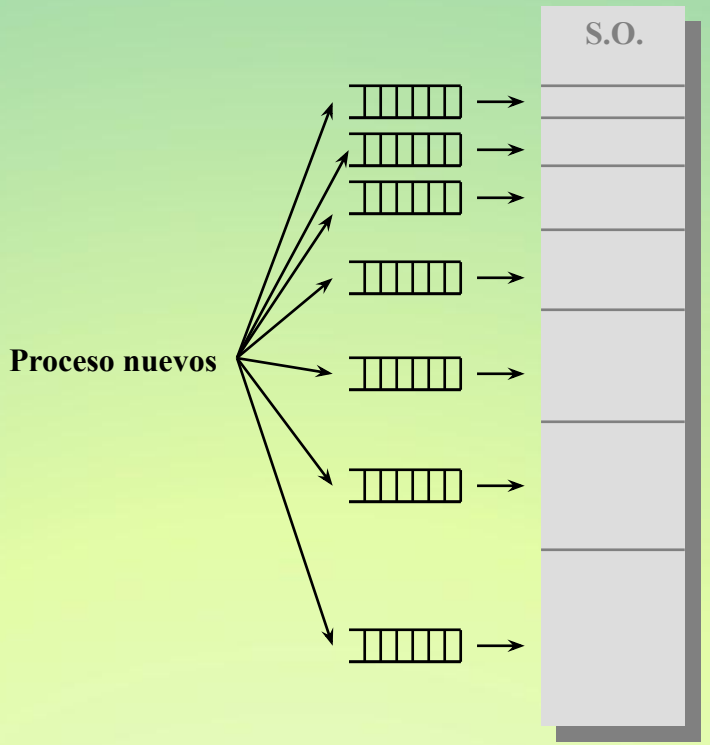
(b) Unequal-size partitions

Algoritmo de ubicación

- Particiones del mismo tamaño
  - Algoritmo trivial
- Particiones de tamaño diferente
  - Asignar a cada proceso a la partición más pequeña dentro de la cual cabe
    - Una cola por partición
    - Minimiza fragmentación interna
  - Cola única
    - Mayor aprovechamiento en uso de todas las particiones

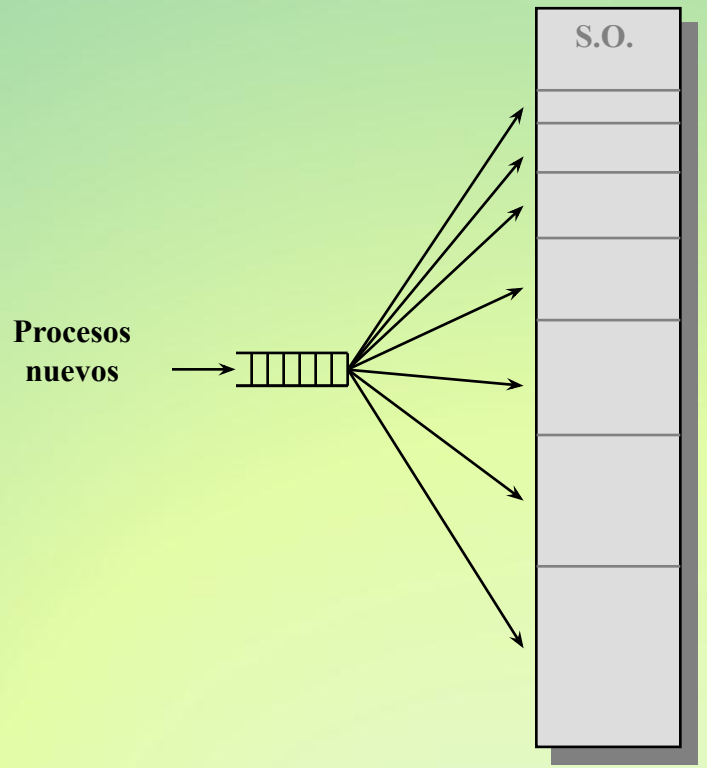
Algoritmo de ubicación

- **Traducción y carga absoluta**
  - Desperdicio del almacenamiento



Algoritmo de ubicación

- **Traducción y carga con reubicación**
  - Los programas se pueden contener en cualquier partición disponible lo suficientemente grande.



Particionamiento fijo. Problemas remanentes

- El número de procesos activos está limitado por el número de particiones
- Muchos procesos pequeños no utilizan el espacio eficientemente
  - En particiones fijas de tamaño variable o fijo

Particionamiento dinámico

- las particiones son de tamaño y número variable
- los procesos ocupan tanto espacio como necesiten, no hay fragmentación interna
- se van generando huecos, esto se llama **fragmentación externa**
- se debe usar compactación que malgasta tiempo de procesador y requiere capacidad de reubicación dinámica

Particionamiento dinámico. Ejemplo

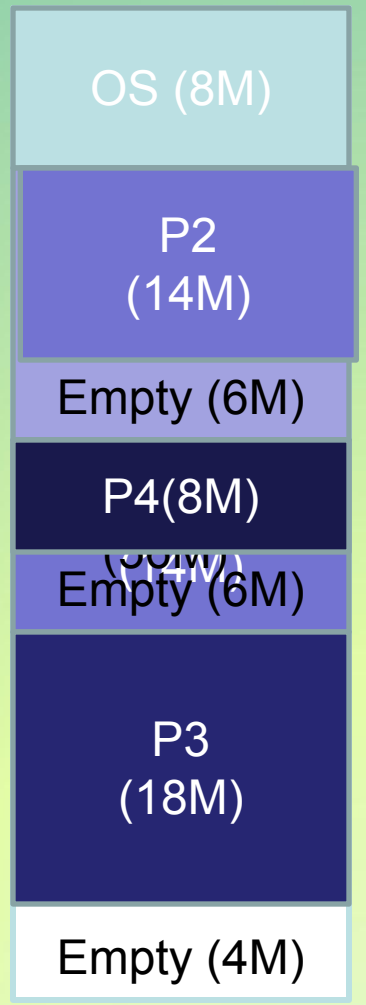


Figura 7.4

Algoritmo de ubicación

- Estrategia del **mejor ajuste**
  - asigna el menor hueco disponible
  - al quedar huecos más pequeños, hay que compactar más frecuentemente
- Estrategia del **peor ajuste**
  - asigna el mayor hueco disponible
  - mejor que la anterior en cuanto a compactación

Algoritmos de ubicación

- Estrategia del **primer ajuste**
  - asigna el primer hueco disponible
  - más rápida
  - puede tener muchos procesos cargados en el extremo inferior de la memoria, los cuales hay que traspasar para llegar al hueco disponible.

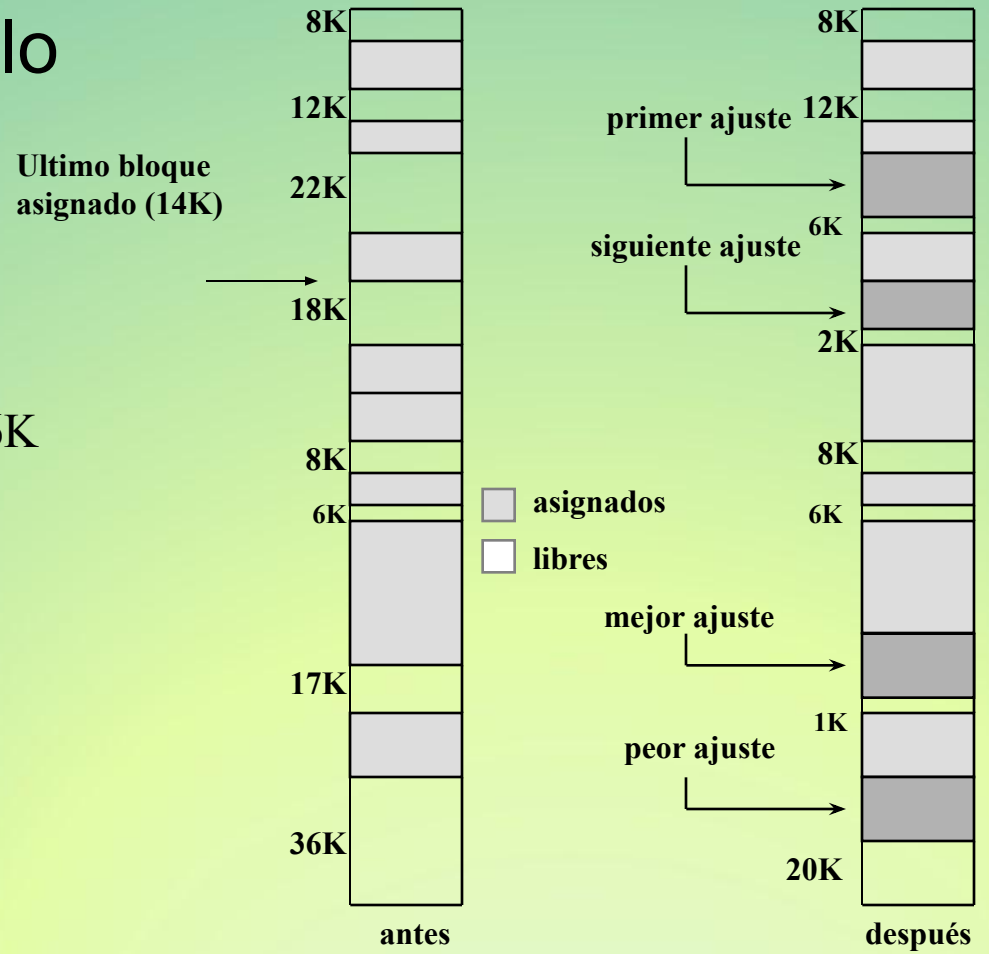
Algoritmos de ubicación

- Estrategia del **siguiente ajuste**
  - variante del primer ajuste, empieza a buscar desde donde terminó la búsqueda anterior
  - se fragmenta más el gran espacio libre del final de la memoria
  - se requiere compactación más frecuente que la anterior

Algoritmos de ubicación

- Ejemplo

Requisito 16K



Sistema buddy (colegas)

- El espacio disponible se trata como un único bloque de 2^U
- Si se realiza una petición de tamaño *s* donde $2^{U-1} < s <= 2^U$
  - Se asigna el bloque entero
- De otro modo el bloque se divide en dos bloques “colegas” iguales
  - Se asigna la petición a uno de los dos
  - El proceso continúa hasta que el bloque más pequeño sea más grande o igual que el s generado y se asigna la petición

Sistema buddy

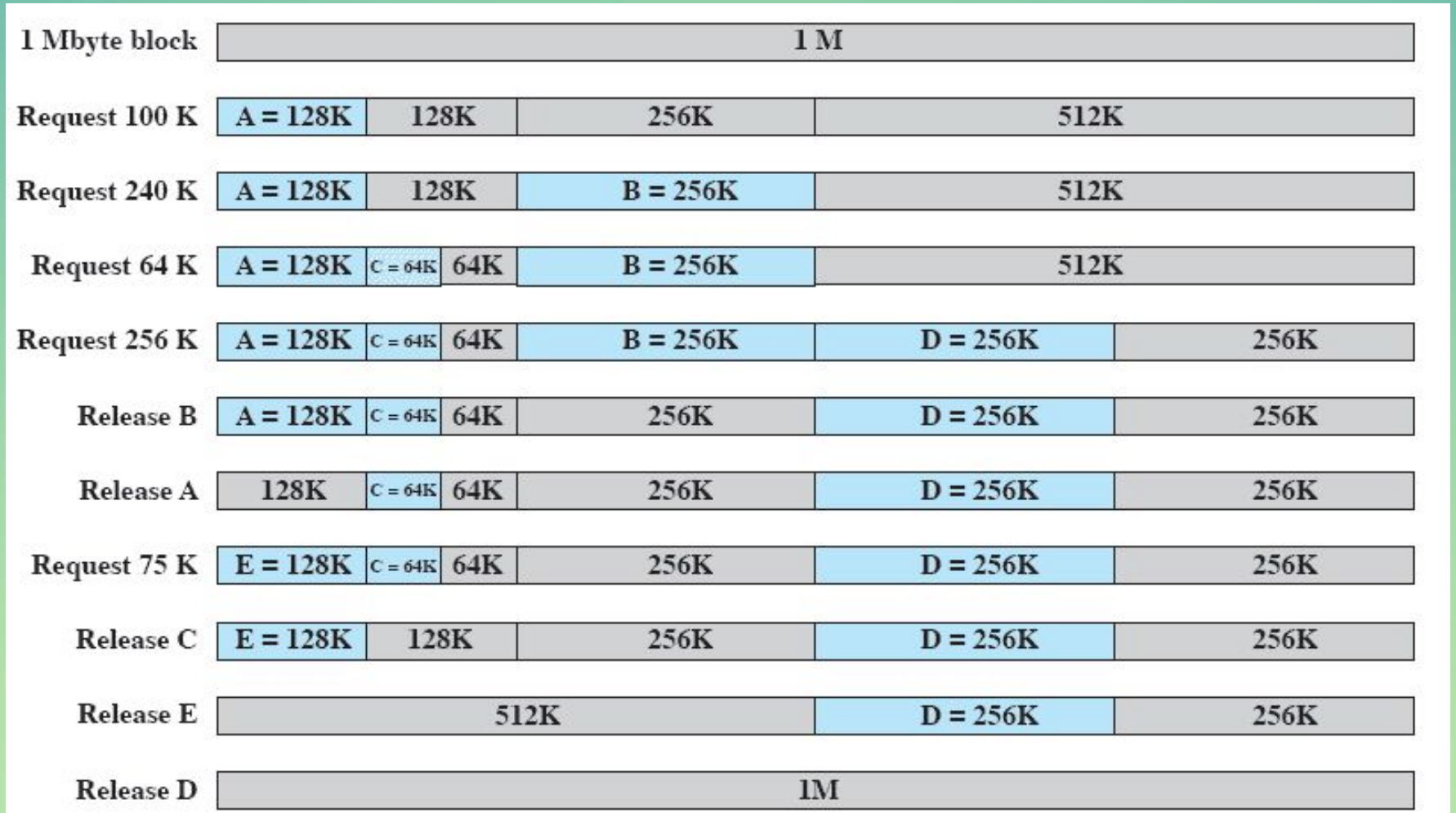


Figure 7.6 Example of Buddy System

Sistema buddy

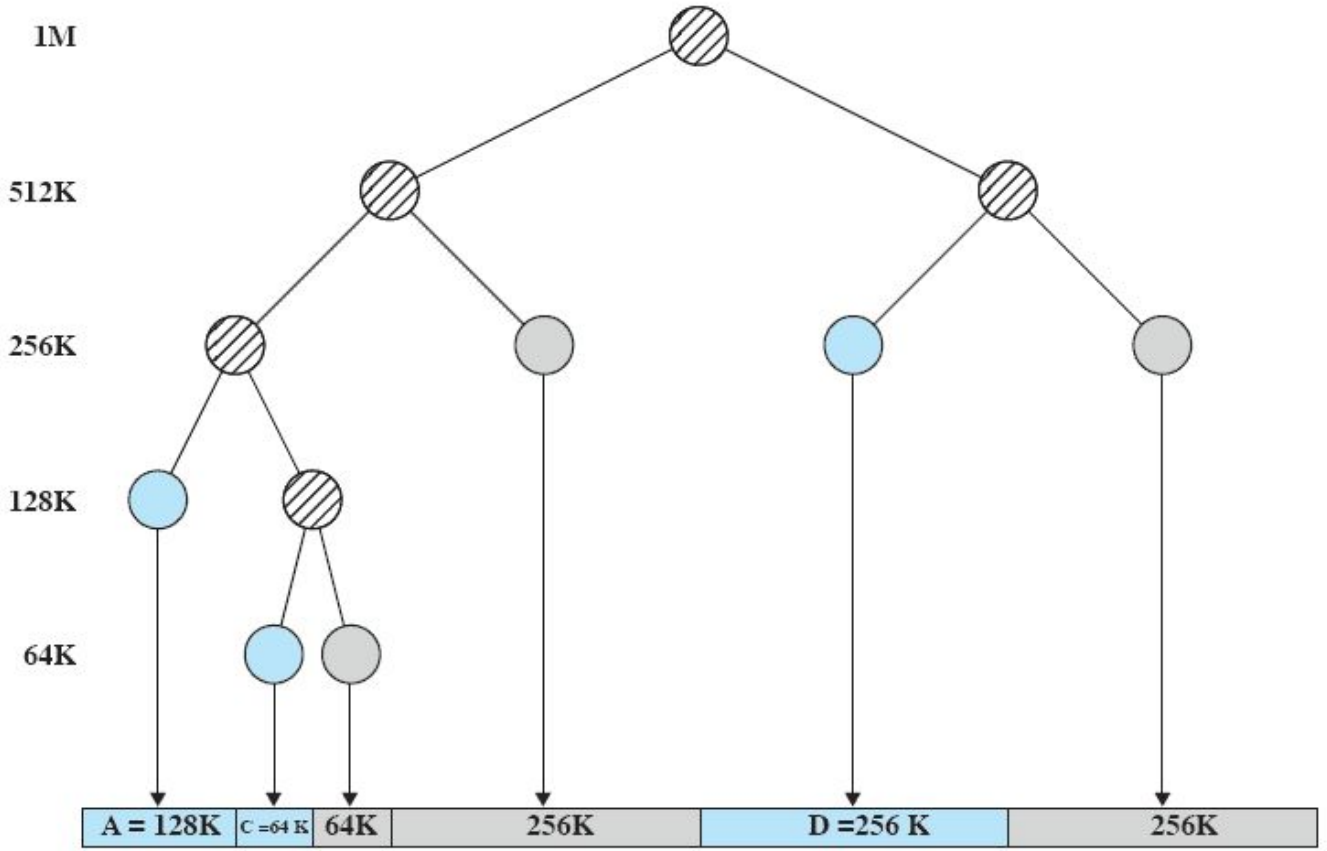


Figure 7.7 Tree Representation of Buddy System

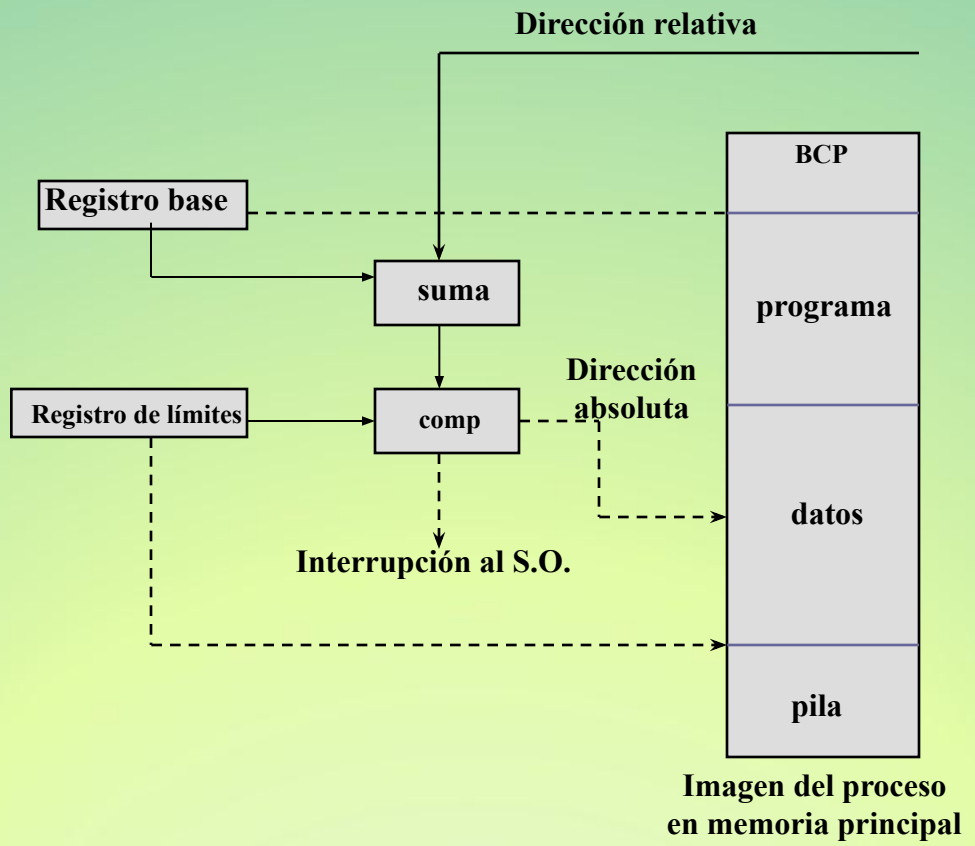
Reubicación

- Cuando un programa se carga en la memoria se determinan las direcciones
- un proceso puede ocupar diferentes particiones (o sea diferentes direcciones absolutas) durante la ejecución (por intercambio)
- La compactación también causa que un programa tenga que ocupar diferentes particiones.

Direcciones

- direcciones virtuales (lógicas)
  - direcciones a las cuales hacen referencia los procesos independientemente de la asignación actual
  - mecanismos de hardware hacen la traducción a direcciones reales (físicas)
- direcciones relativas
  - direcciones expresadas en relación a algún punto conocido
- direcciones reales (físicas)
  - direcciones actuales de memoria

Hardware para reubicación dinámica



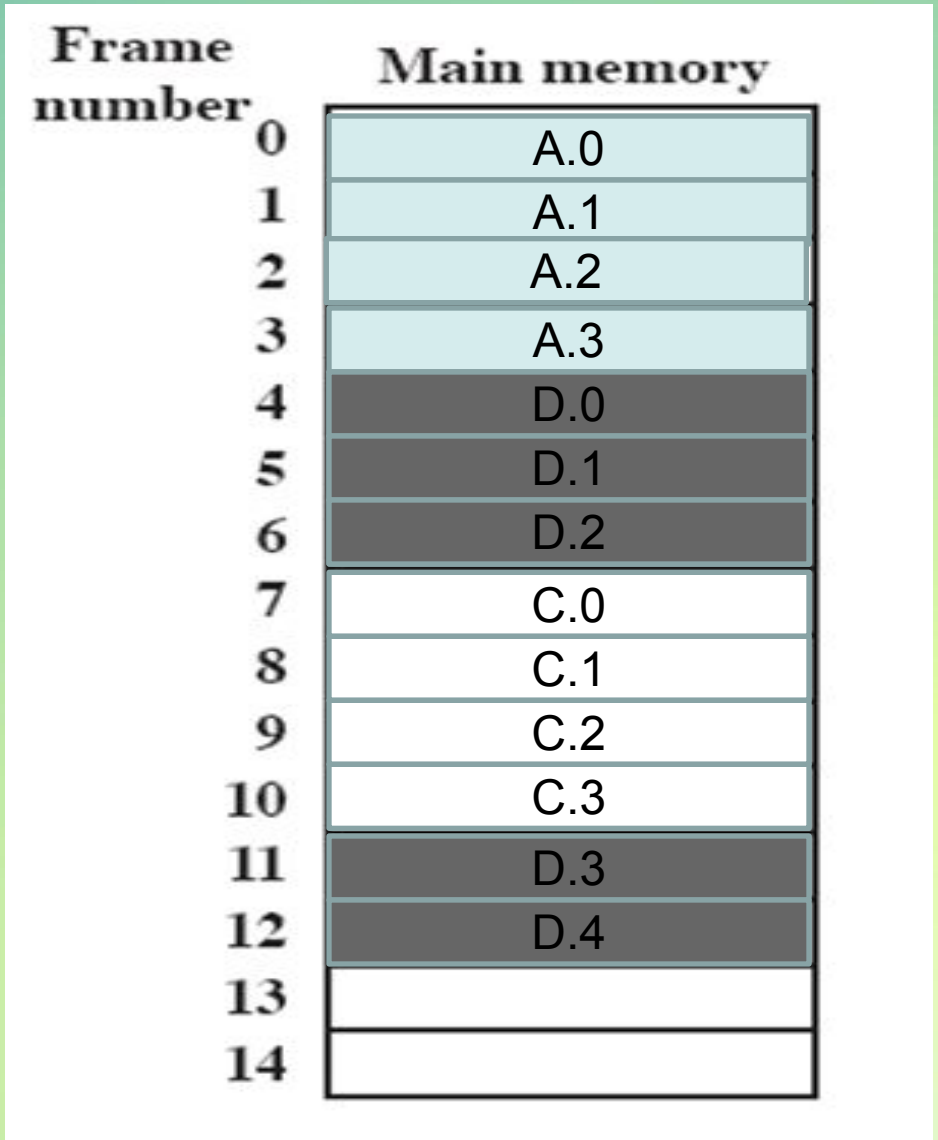
Paginación

- La **memoria principal** se divide en un número de **marcos** de igual tamaño.
- Cada **proceso** se divide en un número de **páginas** del mismo tamaño que los marcos.
- Un proceso se carga colocando todas sus páginas en marcos disponibles, no necesariamente contiguos.

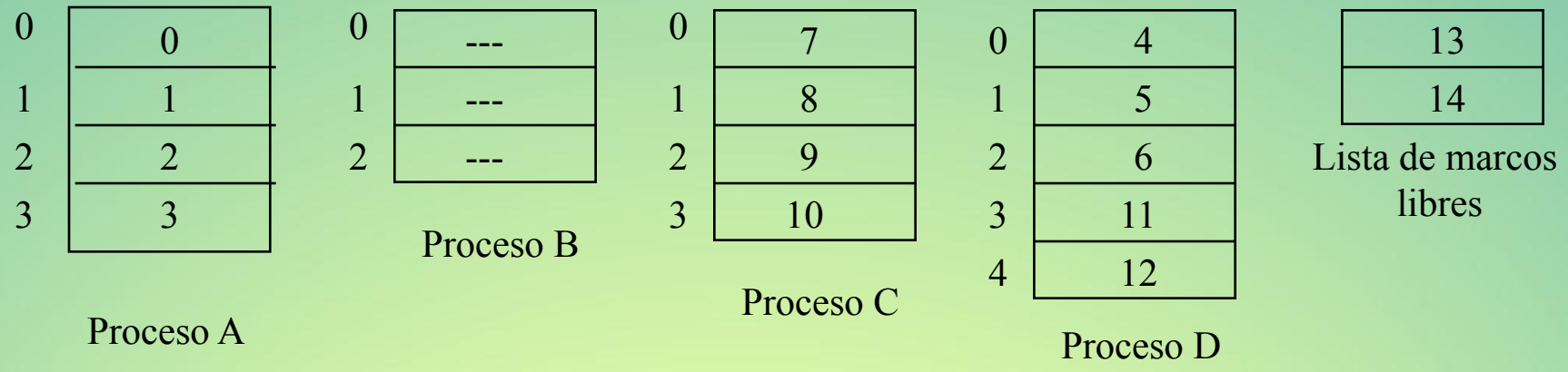
Paginación

- El sistema operativo mantiene una tabla de páginas por cada proceso
  - contiene la dirección del marco correspondiente a cada página en el proceso
  - la dirección de memoria está dada por el número de página y el desplazamiento dentro de la página

Asignación de procesos a marcos



Tablas de páginas



Segmentación

- Cada proceso se divide en un número de segmentos
- la dirección se expresa como número de segmento y desplazamiento
- similar a particiones variables, pero no necesitan estar en forma contigua

Direcciones lógicas

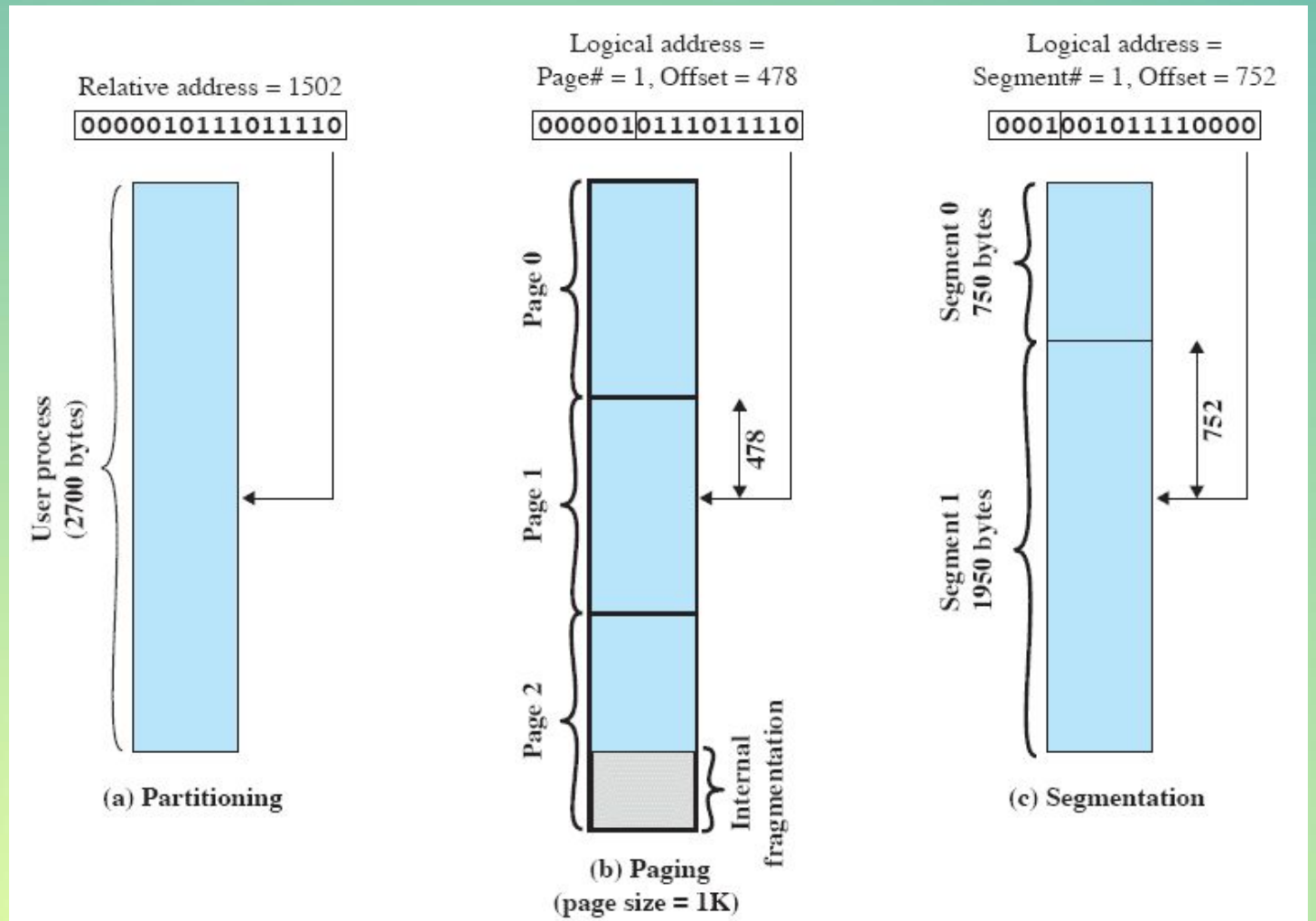
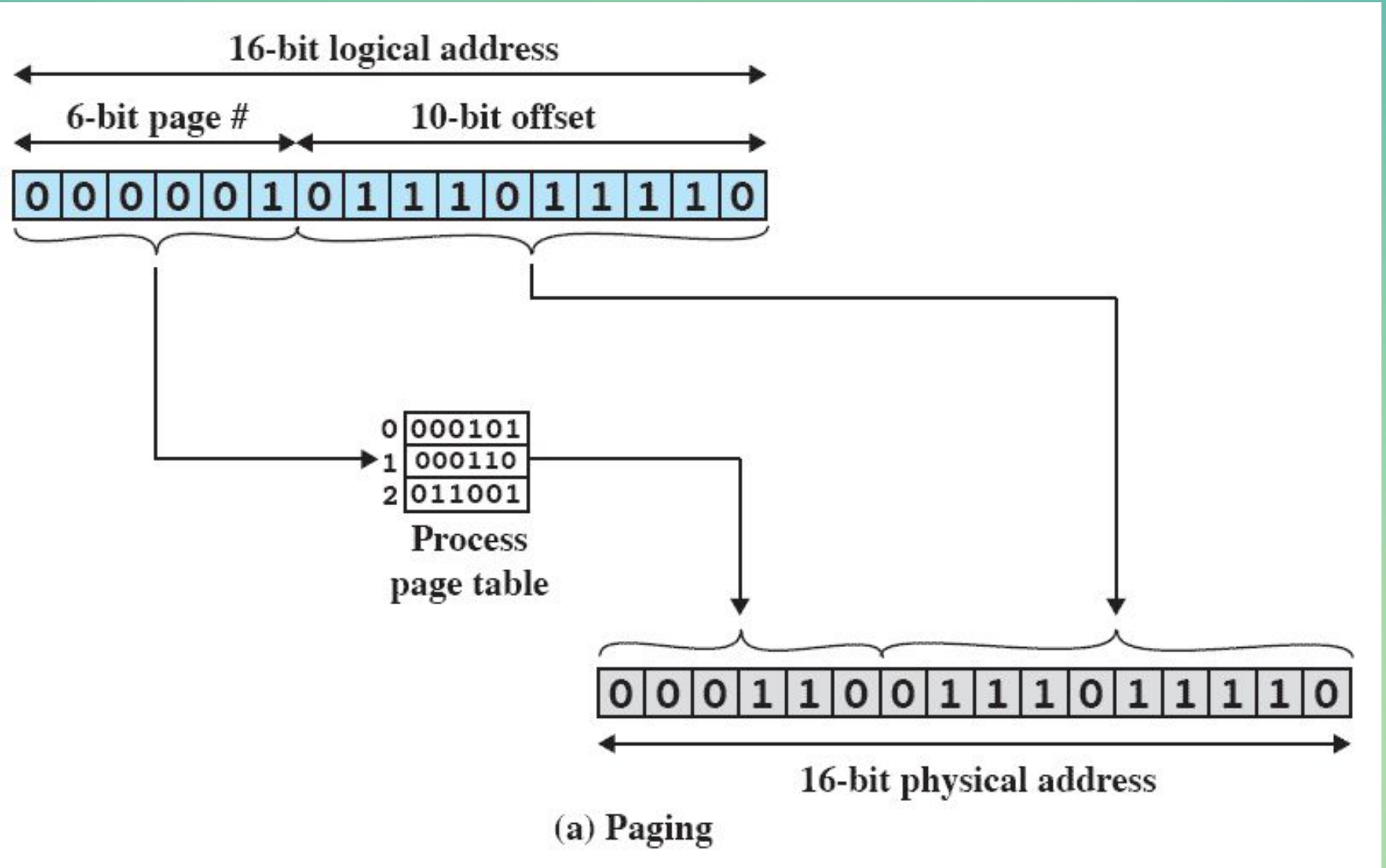


Figure 7.11 Logical Addresses

Paginación



(a) Paging

Segmentación

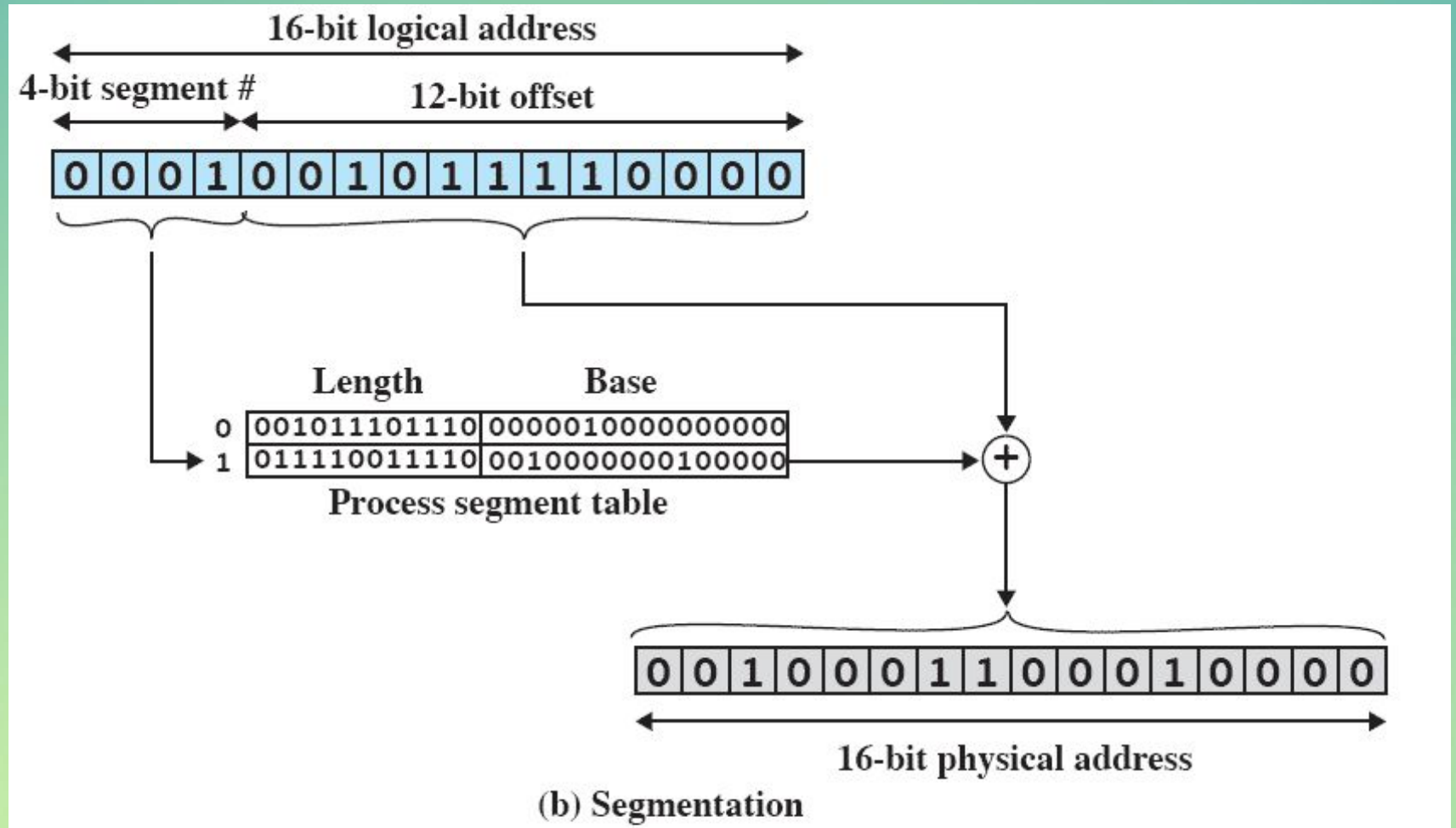


Figure 7.12 Examples of Logical-to-Physical Address Translation

Memoria virtual

Organización

Características de paginación y segmentación

- Traducción dinámica de direcciones en tiempo de ejecución, lo que permite carga en distintas regiones
- Asignación no contigua
- No es necesario que todos los segmentos o páginas estén en memoria durante la ejecución

Ejecución de un programa

- El sistema operativo trae a memoria principal unos pocos fragmentos del proceso incluido el comienzo del programa
- conjunto residente: porción de proceso que está en memoria principal
- Se genera interrupción cuando no se encuentra la dirección lógica en memoria principal. Fallo de acceso a memoria

Ejecución de un programa

- Fragmento de proceso que contiene la dirección lógica se trae a memoria principal
  - el proceso pasa a bloqueo solicitando E/S a disco
  - otro proceso se despacha mientras la E/S toma lugar
  - se genera una interrupción cuando se completa la E/S a disco lo que causa que el SO cambie el estado del proceso a listo

Ventajas de fragmentar procesos

- Más procesos se pueden mantener en memoria principal
  - solamente se cargan algunos fragmentos de cada proceso
- Utilización más eficiente del procesador
- Un proceso puede ser más grande que toda la memoria principal

Ventajas de fragmentar procesos

- El programador trata con memoria del tamaño del disco duro
- Se cargan en memoria los fragmentos que se necesitan
- Se ahorra tiempo porque no se cargan y descargan fragmentos que no se usen

Hiperpaginación (thrashing)

- Intercambio hacia afuera de un bloque de un proceso justo antes de que se necesite
- El procesador gasta la mayor cantidad de su tiempo intercambiando los bloques en lugar de ejecutando las instrucciones.

Principio de cercanía

- Las referencias a datos y programas dentro de un proceso tienden a agruparse.
- Solamente algunas partes de un proceso se necesitarán por un periodo breve de tiempo.
- Es posible adivinar inteligentemente cuáles bloques se necesitarán en el futuro.

Principio de cercanía

Localidad espacial

- tendencia a referenciar localidades cercanas entre sí.
  - Secuencias lineales de código, recorridos de vectores, definiciones de variables afines cercanas unas a otras

Localidad temporal

- tendencia a referenciar la misma posición varias veces durante breves intervalos
  - ciclos, subrutinas, pilas, variables utilizadas para cuenta y totalización

Soporte necesario para memoria virtual

- El hardware debe soportar paginado y/o segmentado.
- El sistema operativo debe ser capaz de manejar los movimientos de páginas y/o segmentos entre memorias primaria y secundaria.

Paginación

- Cada proceso tiene su propia tabla.
- Cada entrada de la tabla de páginas contiene el número de marco de la página correspondiente en memoria principal.
- Se necesita un bit para indicar si la página está o no en memoria principal (**bit de presencia**).
- Se necesita un bit para indicar si la página ha sido modificada desde la última vez que fue cargada en memoria (**bit de modificación**).
- Si no ha habido cambio, la página no necesita escribirse en el disco cuando se quita de memoria principal.

Traducción de direcciones en sistemas de paginado

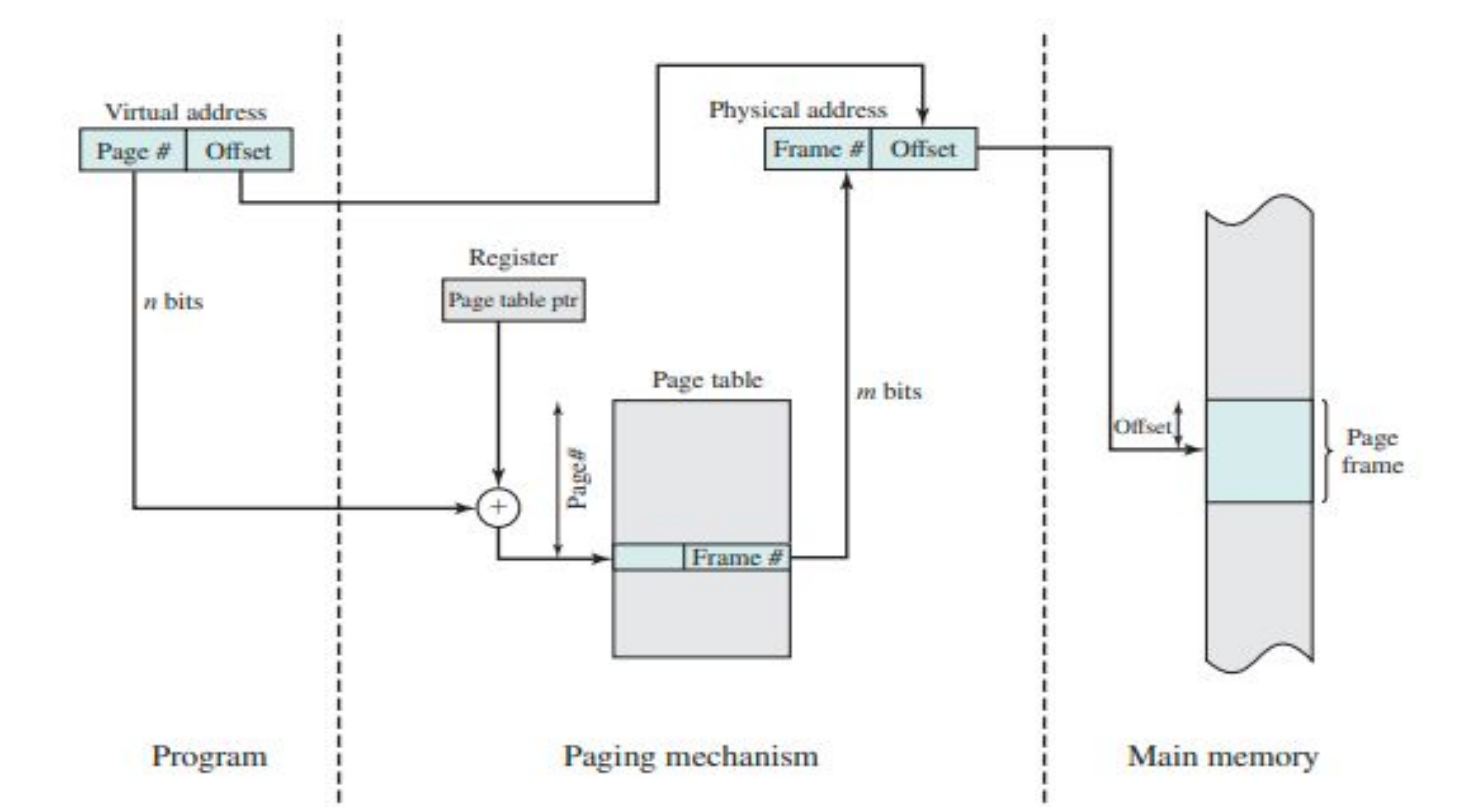


Figure 8.2 Address Translation in a Paging System

Tablas de correspondencia de páginas

- Pueden ser muy grandes y necesitar mucha memoria.
- Las tablas también se almacenan en memoria virtual.
- Cuando un proceso se ejecuta, parte de su tabla está en memoria principal.

Correspondencia

- Cada referencia a una dirección virtual puede causar dos accesos a memoria física.
  - Uno para leer la tabla
  - uno para leer el dato
- para mejorar esto se usa una cache especial para las entradas de las tablas
  - llamado TLB - Translation Lookaside Buffer

TLB

- Contiene las entradas de la tabla de páginas que han sido usadas más recientemente.
- Trabaja parecido a la cache de memoria principal.
- Dada una dirección virtual, el procesador examina el TLB.
- Si se encuentra la entrada de la página (hit), se obtiene el número del marco y se forma la dirección real.
- Si no se encuentra la entrada de la página (miss), se usa el número de página para indexar la tabla de páginas del proceso.

TLB

- Primero comprueba si la página está en memoria principal.
- Si no, se avisa un fallo de página.
- Se actualiza el TLB para incluir la nueva entrada.

Uso del TLB

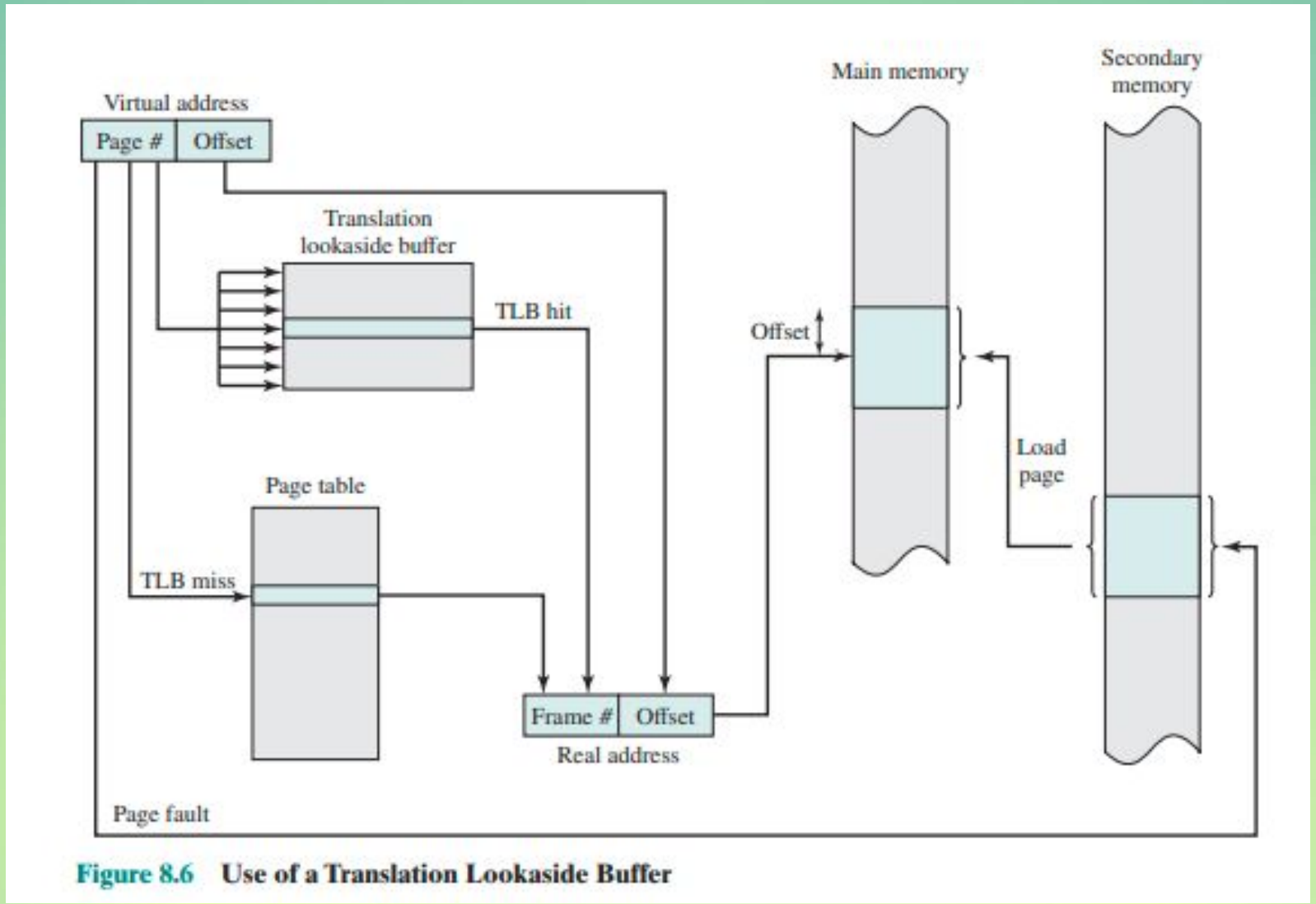
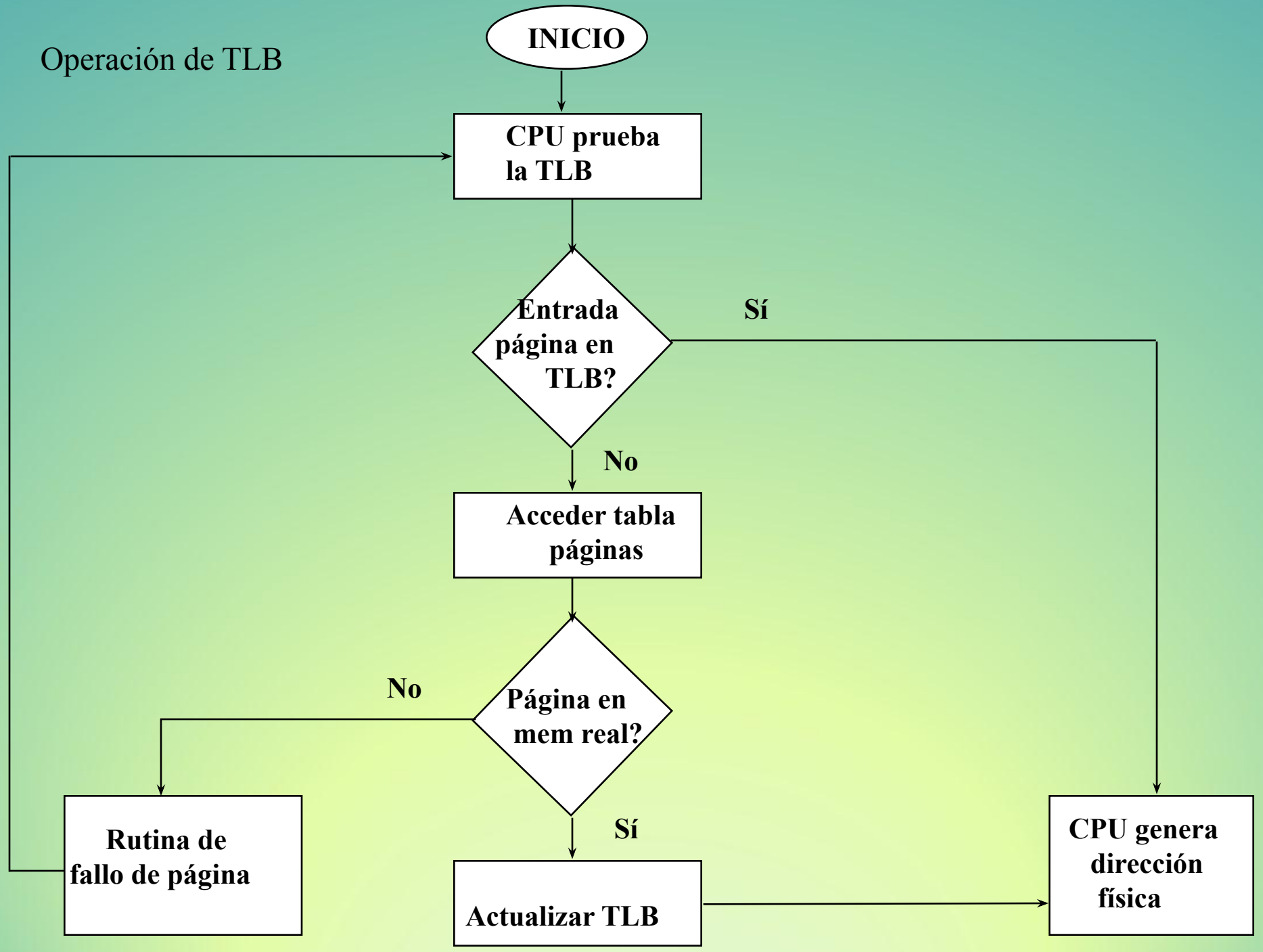
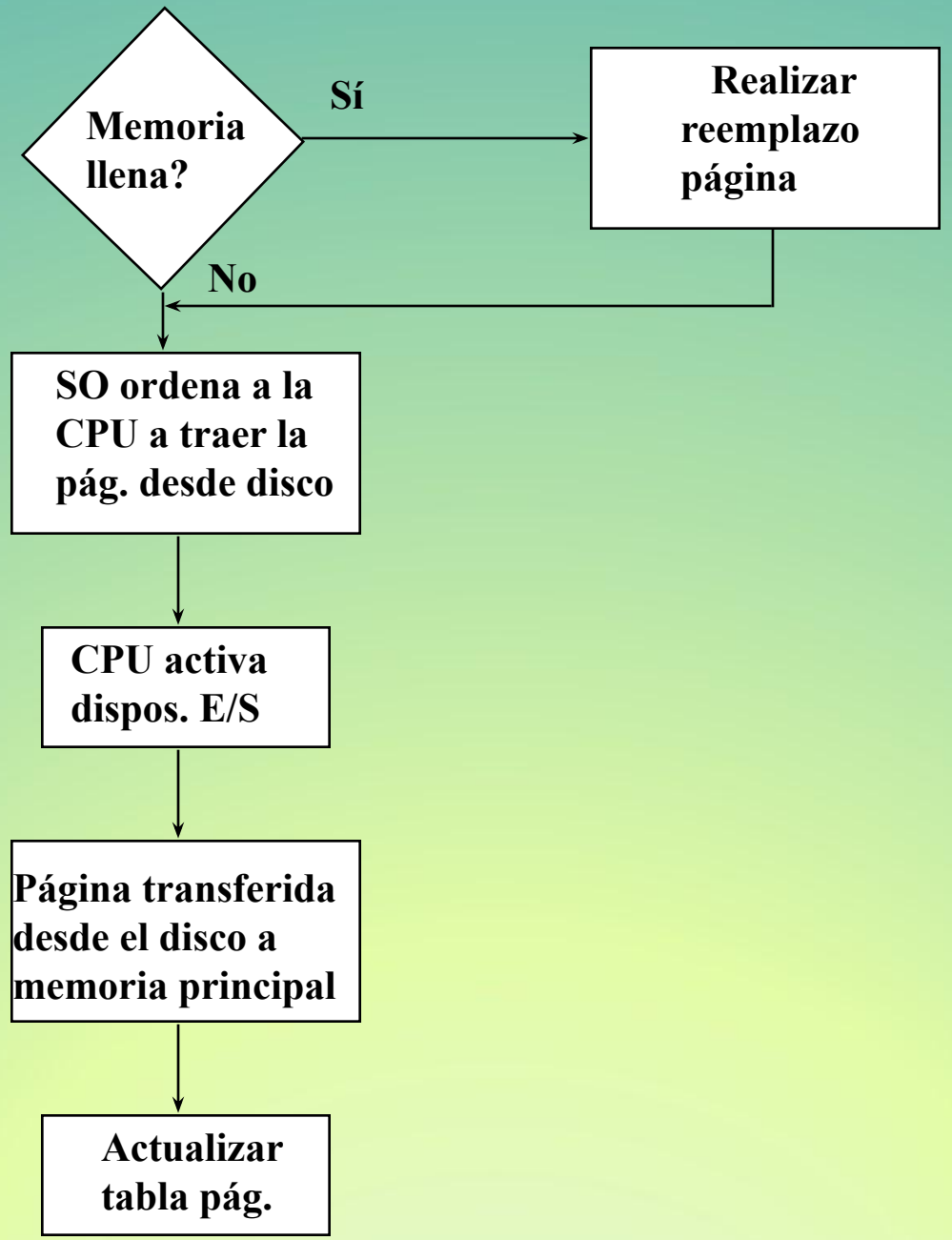


Figure 8.6 Use of a Translation Lookaside Buffer

Operación de TLB



Rutina de fallo de página



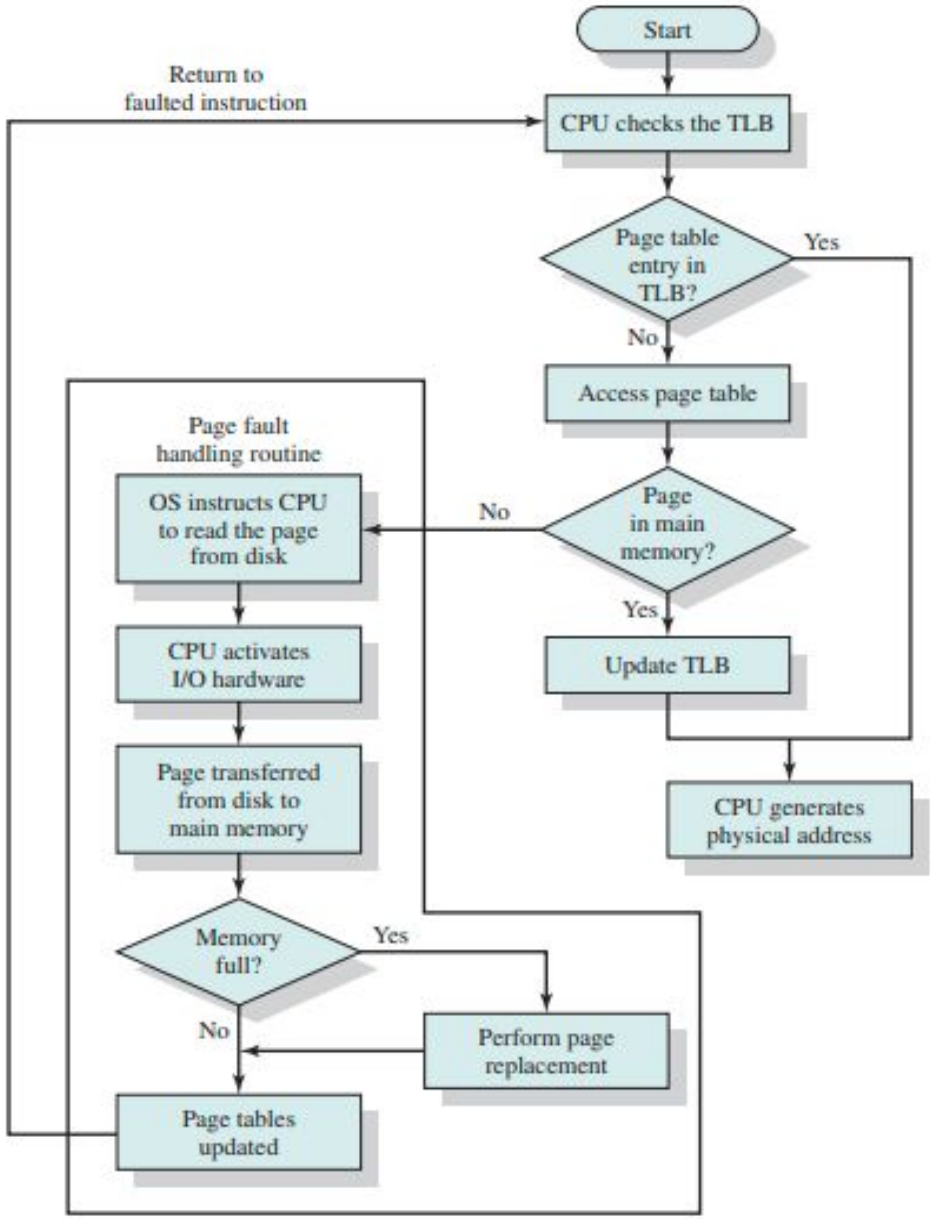


Figure 8.7 **Operation of Paging and Translation Lookaside Buffer (TLB)**

Tamaño de páginas

- A menor tamaño, menor fragmentación interna.
- Con menor tamaño, se requieren más páginas por procesos. Esto quiere decir, tablas mayores.
- Mayores tablas, grandes porciones de tablas en memoria virtual.
- La memoria secundaria está diseñada para transferir grandes bloques de datos eficientemente, por lo tanto mayores tamaños de páginas es mejor.

Tamaño de páginas

- Pequeño tamaño, mayor número de páginas se encontrarán en memoria principal
- A medida que sigue la ejecución, las páginas en memoria contendrán porciones del proceso traído recientemente a memoria, por lo tanto disminuyen los fallos de página
- Múltiples tamaños de página proveen la flexibilidad necesaria para usar efectivamente el TLB
  - Páginas grandes se pueden usar para instrucciones de programa
  - Páginas pequeñas se pueden usar para hilos

Segmentado

- Puede ser dinámico
- Simplifica el manejo de estructuras que crecen.
- Permite alterar los programas y recompilarlos independientemente.
- Se usa para compartir datos entre procesos.
- Fácil protección.

Tablas de correspondencia de segmentos

- Cada entrada contiene la dirección de comienzo del correspondiente segmento en memoria principal.
- Cada entrada contiene la longitud del segmento.
- Se necesita un bit para determinar si el segmento está o no en memoria principal.
- Se necesita un bit para determinar si el segmento ha sido modificado desde la última vez que se cargó.

Traducción de direcciones en sistemas segmentados

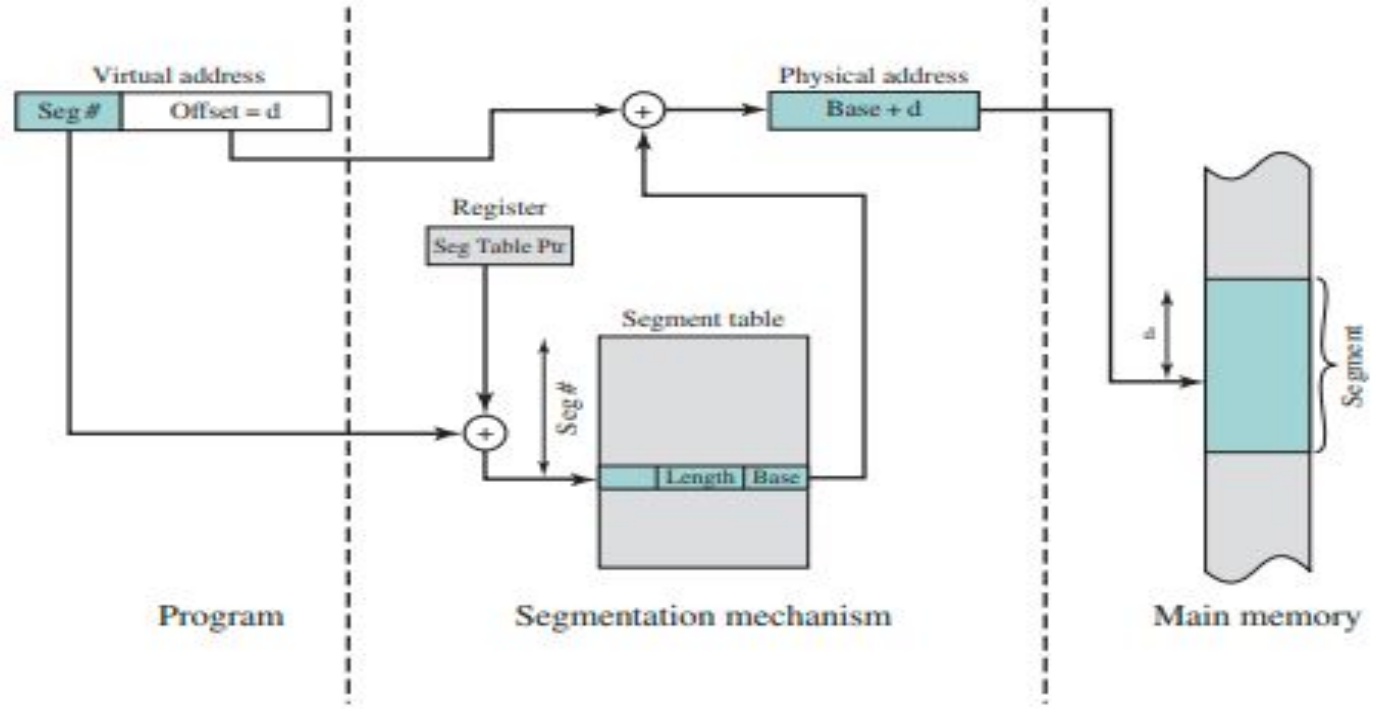


Figure 8.11 **Address Translation in a Segmentation System**

Segment table entry

P	M	Other control bits	Length	Segment base

(b) Segmentation only

Sistemas de paginación/segmentación

- La paginación es transparente al programador.
- La paginación elimina fragmentación externa.
- La segmentación es visible al programador.
- La segmentación permite crecimiento de estructuras, modularidad y soporte para compartir y proteger.
- Cada segmento se parte en páginas de tamaño fijo.

Traducción de direcciones en sistemas segmentados/paginados

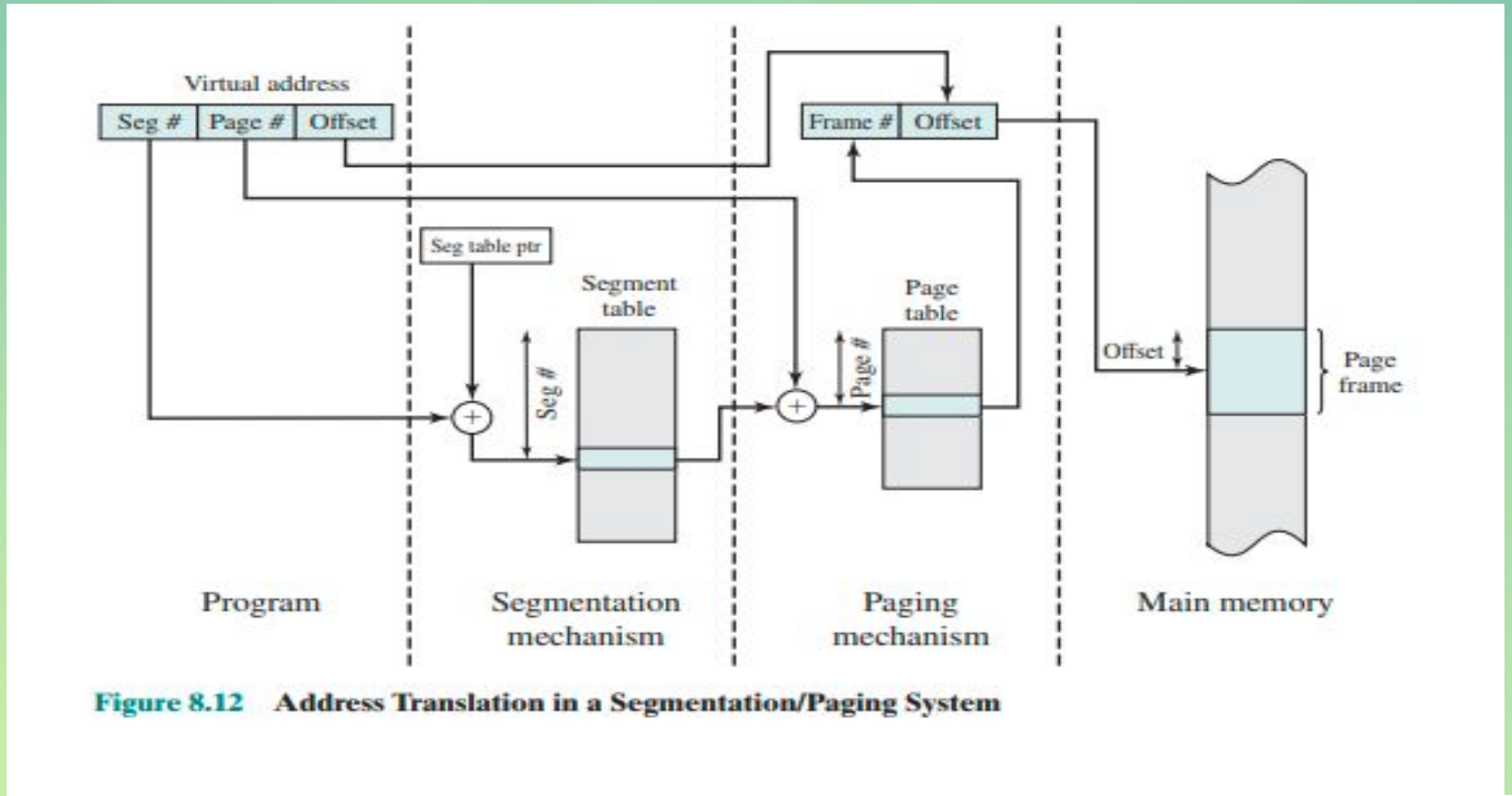
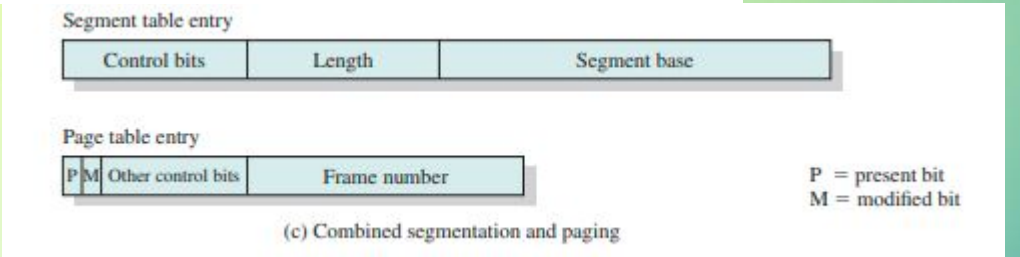


Figure 8.12 **Address Translation in a Segmentation/Paging System**



Paginación segmentación

Tabla 8.1. Características de la paginación y la segmentación.

Paginación sencilla	Paginación con memoria virtual	Segmentación sencilla	Segmentación con memoria virtual
Memoria principal particionada en fragmentos pequeños de un tamaño fijo llamados marcos	Memoria principal particionada en fragmentos pequeños de un tamaño fijo llamados marcos	Memoria principal no particionada	Memoria principal no particionada
Programa dividido en páginas por el compilador o el sistema de gestión de la memoria	Programa dividido en páginas por el compilador o el sistema de gestión de la memoria	Los segmentos de programa se especifican por el programador al compilador (por ejemplo, la decisión se toma por parte el programador)	Los segmentos de programa se especifican por el programador al compilador (por ejemplo, la decisión se toma por parte el programador)
Fragmentación interna dentro de los marcos	Fragmentación interna dentro de los marcos	Sin fragmentación interna	Sin fragmentación interna
Sin fragmentación externa	Sin fragmentación externa	Fragmentación externa	Fragmentación externa
El sistema operativo debe mantener una tabla de páginas por cada proceso mostrando en el marco que se encuentra cada página ocupada	El sistema operativo debe mantener una tabla de páginas por cada proceso mostrando en el marco que se encuentra cada página ocupada	El sistema operativo debe mantener una tabla de segmentos por cada proceso mostrando la dirección de carga y la longitud de cada segmento	El sistema operativo debe mantener una tabla de segmentos por cada proceso mostrando la dirección de carga y la longitud de cada segmento
El sistema operativo debe mantener una lista de marcos libres	El sistema operativo debe mantener una lista de marcos libres	El sistema operativo debe mantener una lista de huecos en la memoria principal	El sistema operativo debe mantener una lista de huecos en la memoria principal
El procesador utiliza el número de página, desplazamiento para calcular direcciones absolutas	El procesador utiliza el número de página, desplazamiento para calcular direcciones absolutas	El procesador utiliza el número de segmento, desplazamiento para calcular direcciones absolutas	El procesador utiliza el número de segmento, desplazamiento para calcular direcciones absolutas
Todas las páginas del proceso deben encontrarse en la memoria principal para que el proceso se pueda ejecutar, salvo que se utilicen solapamientos (*overlays*)	No se necesita mantener todas las páginas del proceso en los marcos de la memoria principal para que el proceso se ejecute. Las páginas se pueden leer bajo demanda	Todos los segmentos del proceso deben encontrarse en la memoria principal para que el proceso se pueda ejecutar, salvo que se utilicen solapamientos (*overlays*)	No se necesitan mantener todos los segmentos del proceso en la memoria principal para que el proceso se ejecute. Los segmentos se pueden leer bajo demanda
	La lectura de una página a memoria principal puede requerir la escritura de una página a disco		La lectura de un segmento a memoria principal puede requerir la escritura de uno o más segmentos a disco

Memoria virtual

Estrategias de Gestión

Políticas

- Lectura (también políticas de vaciado)
  - cuándo se debe transferir una página (segmento)
- Ubicación
  - dónde colocar la nueva página (segmento)
- Reemplazo
  - cuál página (segmento) desalojar de memoria
- Asignación
  - qué cantidad de memoria real se asigna a cada proceso activo
- Control de carga
  - Grado de multiprogramación

Políticas de lectura

- determina cuándo traer una página a memoria
  - paginación por demanda
  - paginación anticipada

Políticas de lectura

- Demanda
  - el paginado por demanda trae solo las páginas cuando se hacen las referencias
  - garantiza que las únicas páginas que se transfieren son las requeridas
  - esperas más costosas

Políticas de lectura

- Paginación anticipada
  - trae más páginas que las que se necesitan
  - acelera tiempos de ejecución de un proceso
  - más eficiente cuando las páginas son contiguas en el disco, se usa en la carga inicial.
  - no se puede predecir el uso de estas páginas

Políticas de ubicación

- determina dónde ubicar un bloque en memoria real
- irrelevante en el caso de paginación
- iguales estrategias que en particiones variables para segmentación
  - primer ajuste
  - mejor ajuste
  - siguiente ajuste

Políticas de reemplazo

- bloqueo de marcos
  - para el kernel, estructuras de control del sistema operativo, buffers I/O
  - bit de bloqueo asociado a cada marco

Políticas de reemplazo

- algoritmos de reemplazo
  - óptimo
  - LRU (menos recientemente utilizada)
    - NUR (no utilizada recientemente)
  - FIFO (primero que entra, primero que sale)
    - reloj
    - segunda oportunidad
- buffering de páginas

Políticas de reemplazo

- Óptima
  - selecciona aquella página que tardará más tiempo en volver a ser utilizada
  - imposible predecir el futuro

Políticas de reemplazo

- **F**irst **I**n, **F**irst **O**ut (FIFO)
  - Reemplaza la página que ha estado más tiempo en almacenamiento primario
  - lista tipo buffer circular (cola FIFO)
  - la más simple de implementar
  - la página más antigua puede ser la más usada

Políticas de reemplazo

- **L**east **R**ecently **U**sed (LRU)
  - Reemplaza la página que no ha sido utilizada por el mayor tiempo
  - Principio de localidad
    - pasado reciente indicador de futuro cercano
  - implementación difícil
    - contador de tiempo utilizado
    - lista ordenada en cada referencia (no en cada fallo de página)

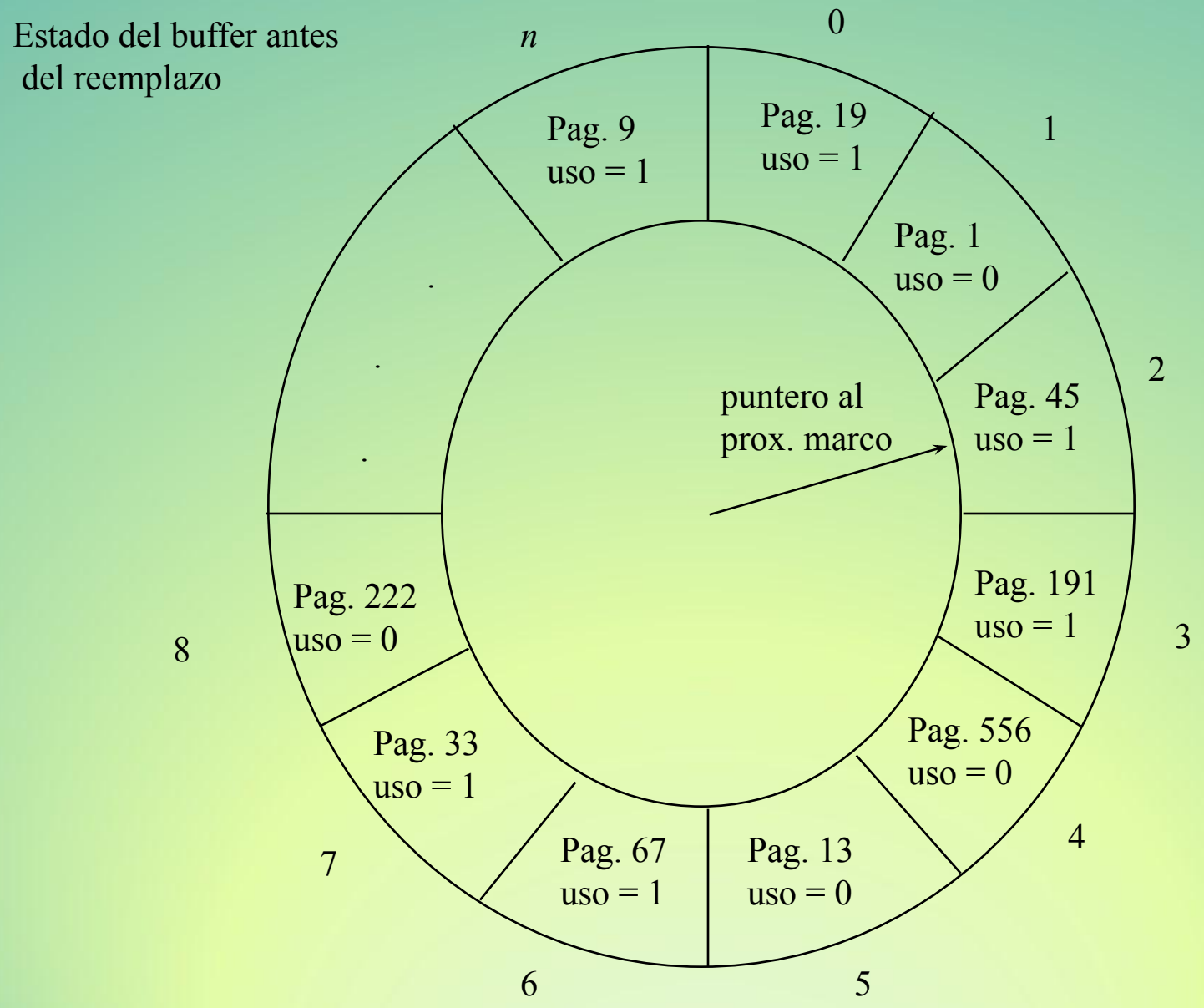
Políticas de reemplazo

- **N**ot **R**ecently **U**sed (NRU)
  - aproximación de LRU
  - utiliza dos bits de hardware
    - uso
    - modificación
  - reemplaza la página no utilizada recientemente basándose en el bit de uso o referencia

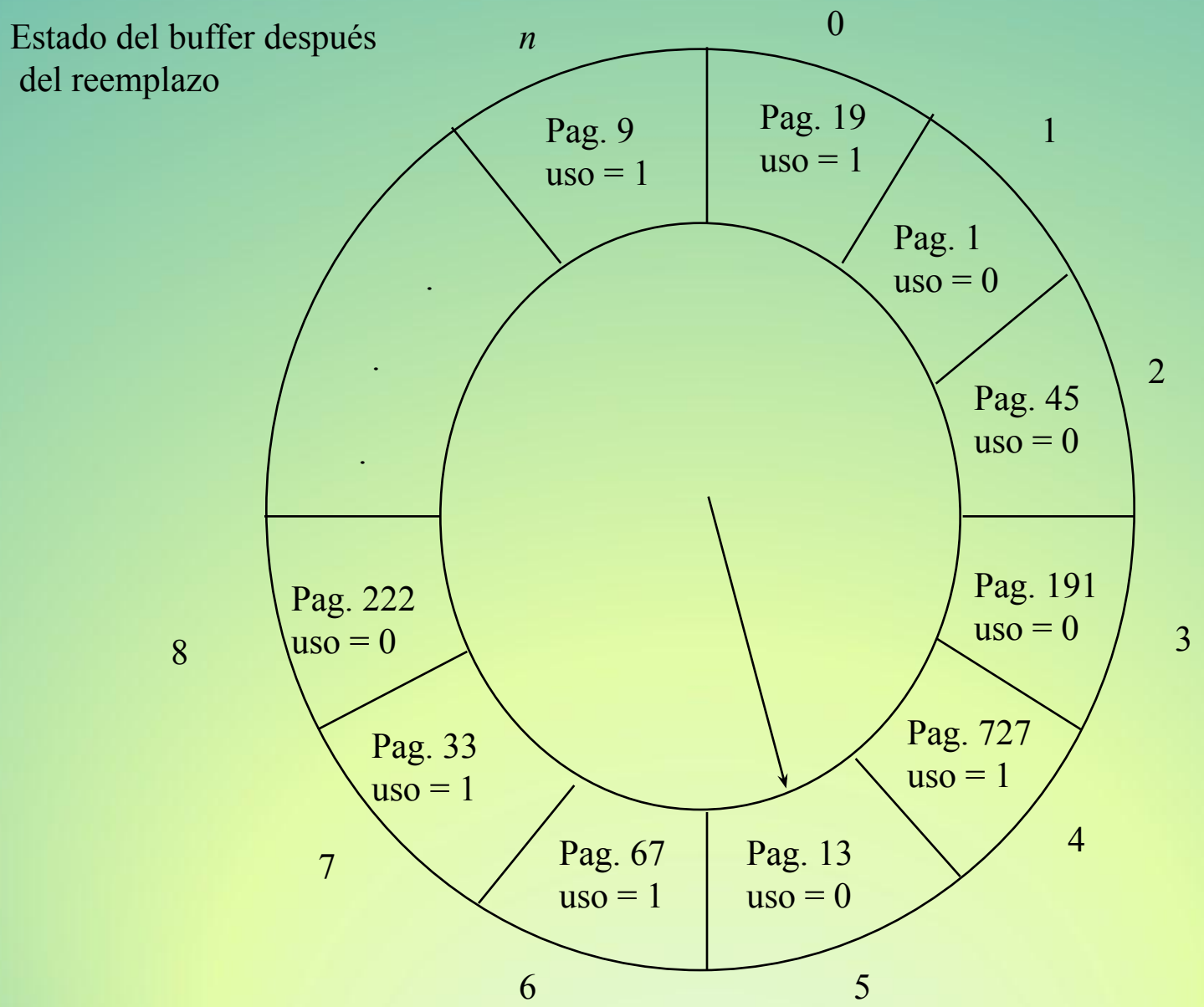
Políticas de reemplazo

- **Reloj o segunda oportunidad**
  - aproximación de FIFO mejorando el rendimiento a través del bit de uso
  - se reemplaza el primer marco cuyo bit de uso sea 0
  - en la búsqueda cada bit de uso en 1 se cambia a 0

Estrategia del reloj



Estrategia del reloj



Políticas de reemplazo.Ejemplo

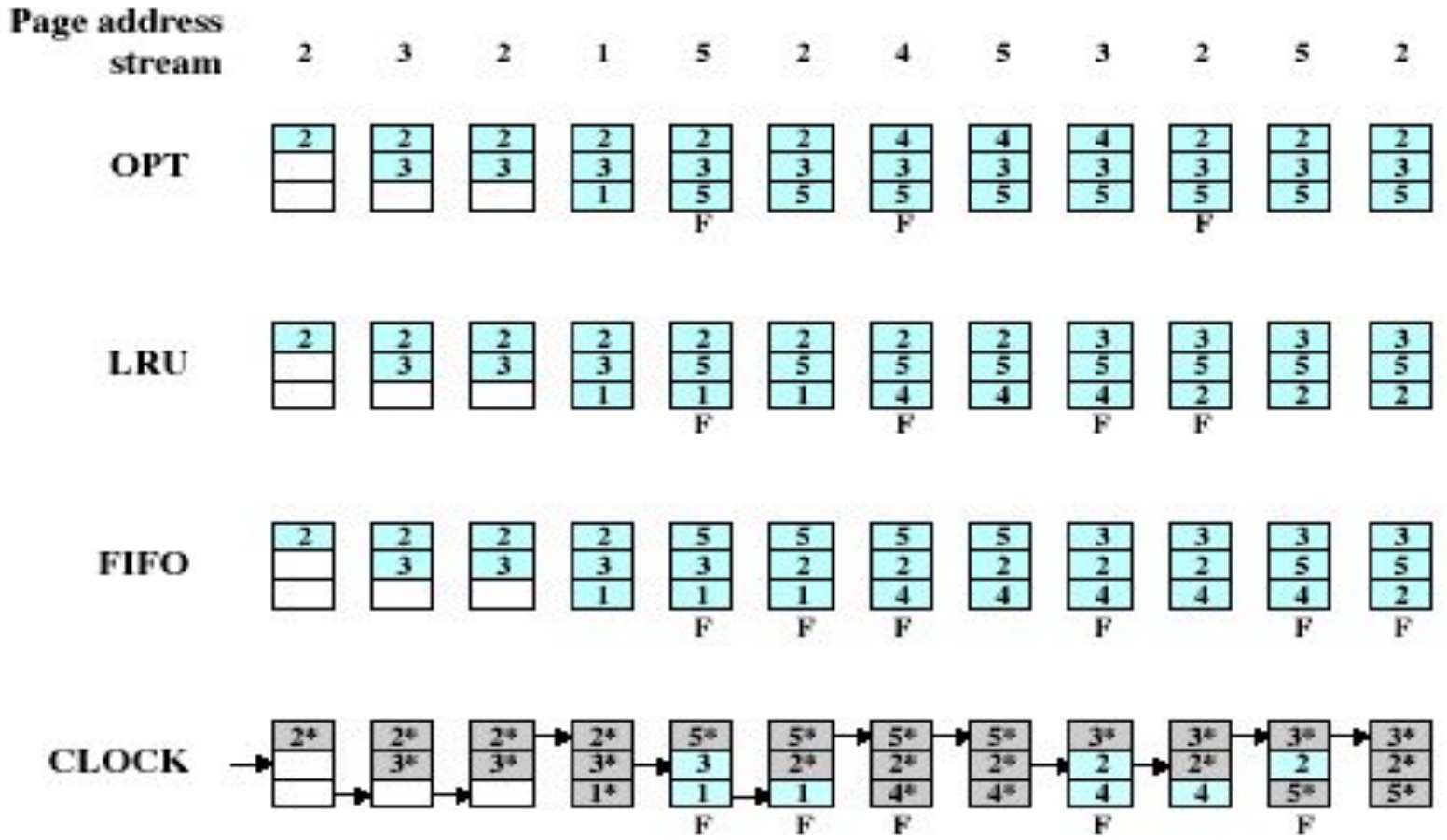


Figure 8.15 Behavior of Four Page-Replacement Algorithms

Buffering de páginas

- Algoritmo FIFO
- la página no se reemplaza inmediatamente, se agrega a una de dos listas
  - lista de marcos libres si la página no ha sido modificada
  - lista de páginas modificadas
- las listas funcionan como cache

Políticas de asignación

- Factores a tener en cuenta
  - menor cantidad de memoria para un proceso, mayor cantidad de procesos residentes. Disminuye el tiempo perdido en intercambios.
  - Pequeña cantidad de páginas residentes, aumento de tasa de fallos de páginas.
  - Después de una cierta cantidad de marcos asignados no hay efecto en la tasa de fallos de página

Políticas de asignación

- Asignación fija
  - número fijo de marcos asignados a un proceso en forma anticipada
  - página a reemplazar debe ser elegida entre los marcos asignados a ese proceso
  - Se usan los algoritmos de reemplazo vistos
  - Desventajas:
    - Asignación muy pequeña: alto grado de fallos de página
    - Asignación muy grande: bajo grado de multiprogramación

Políticas de asignación

- Asignación variable
  - número variable de marcos asignados a un proceso, manteniendo el conjunto de trabajo
  - reemplazo
    - local: se elige la página a reemplazar entre los marcos asignados a ese proceso
    - global: se elige la página a reemplazar entre los marcos disponibles

Políticas de asignación

- Asignación variable con alcance global
  - Ventaja:
    - Fácil de implementar
  - Desventaja
    - Quita marcos de otros procesos activos
  - Se puede usar combinada con buffering de páginas para contrarrestar problemas

Políticas de asignación

- Asignación variable con alcance local
  - Asignación de un cierto número de marcos al proceso
  - Cuando se produce un fallo se selecciona un marco del proceso
  - De vez en cuando se vuelve a evaluar la asignación otorgada

Conjunto de trabajo

- Conjunto de páginas que deben estar en almacenamiento principal para que un proceso se ejecute con eficiencia
- basado en el principio de cercanía
- W(t,w) conj. de páginas a las que hizo referencia el proceso en el intervalo t-w

Conjunto de trabajo

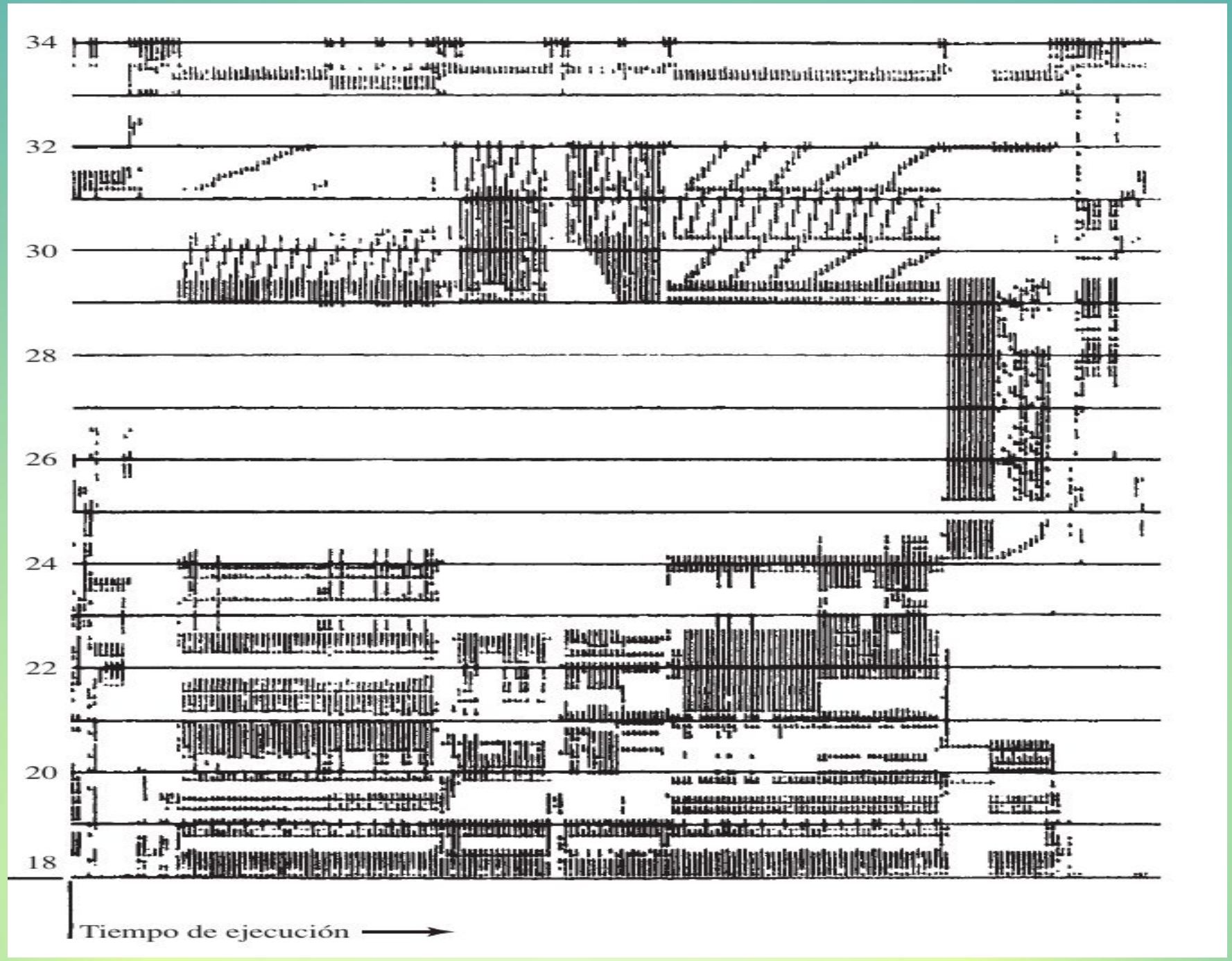
- Políticas de asignación basadas en conjuntos de trabajo
  - intentan mantener el conjunto de trabajo de los procesos activos en memoria real
  - si no se mantiene:
    - sobreutilización de la memoria real (cantidad de marcos > conj. de trabajo)
    - hiperpaginación (cantidad de marcos < conj. de trabajo)

Conjuntos de trabajo

- Problemas:
  - los conjuntos de trabajo de un proceso son transitorios
  - medir el conjunto de trabajo de cada proceso es impráctico
  - el tamaño óptimo de la ventana no se conoce

Algoritmo de frecuencia de fallas de página

- Ajusta el conjunto de páginas residentes de un proceso
  - frecuencia de fallas de página o,
  - tiempo entre fallas
    - > límite superior
      - liberan páginas no referenciadas últimamente
    - < límite inferior
      - página entrante se convierte en miembro del conjunto residente


34
32
30
28
26
24
22
20
18
Tiempo de ejecución

Políticas de vaciado

- Vaciado por demanda
  - una página se escribe en disco solamente cuando ha sido seleccionada para reemplazo
- Prevaciado
  - las páginas se escriben por lotes

Políticas de vaciado

- Buena solución usando buffering de páginas
  - las páginas reemplazadas se colocan en dos listas
    - modificadas o sin modificar
  - las páginas en la lista modificadas se escriben por lotes periódicamente
  - las páginas sin modificar pueden usarse si se referencian nuevamente o perderse si se asigna su marco a otra página

Control de carga

- Determina el número de procesos que estarán residentes en la memoria principal
- Muy pocos procesos, muchas ocasiones en que el procesador estará desocupado porque no hay procesos listos
- Demasiados procesos activos, hiperpaginación

Suspensión de procesos

- Procesos con la prioridad más baja
- Procesos con fallos de páginas
  - este proceso no tiene su conjunto de trabajo en memoria, por lo tanto se bloqueará de cualquier manera
- Último proceso activado
  - este proceso es el que tiene menos posibilidades de tener su conjunto de trabajo residente

Suspensión de procesos

- Procesos con el conjunto residente más chico
  - este proceso requiere el menor esfuerzo futuro para volver a cargarse
- Proceso más grande
  - se obtiene la mayor cantidad de marcos libres
- Proceso con la mayor ventana de ejecución restante

Manejo de memoria en UNIX y Solaris

Sistema de paginado

- Estructuras de datos
  - Tabla de páginas - una por proceso
  - Descriptor de bloque de disco - describe la copia de disco de la página virtual
  - Tabla de datos de marco de página - describe cada marco de memoria real
  - Tabla de uso de swap - una por dispositivo de swap

Manejo de memoria en UNIX y Solaris

- Reemplazo de página
  - refinamiento de la política de reloj conocida como algoritmo de reloj de dos manecillas
- Asignador de memoria para kernel
  - la mayoría de los bloques son más chicos que el tamaño de una página típica

Manejo de memoria en Linux

- Directorio de páginas
- Directorio intermedio de páginas
- Tabla de páginas

Manejo de memoria en Linux

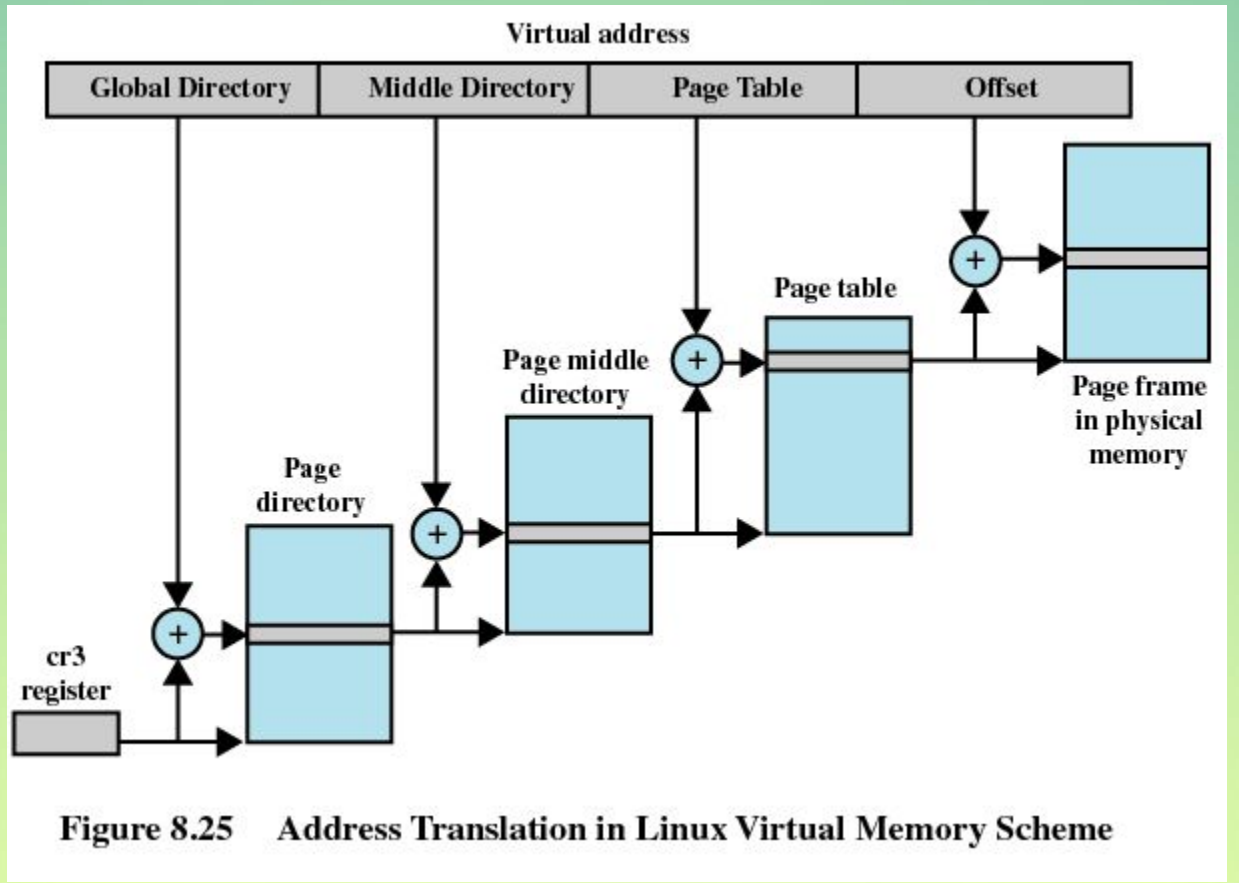


Figure 8.25 Address Translation in Linux Virtual Memory Scheme

Manejo de memoria en Windows

- Asignación de memoria
  - Espacio de direcciones virtual por proceso 2 gigabytes
- Estados de una página
  - disponible
  - reservada
  - comprometida
- Asignación dinámica, alcance local

Manejo de memoria en Windows

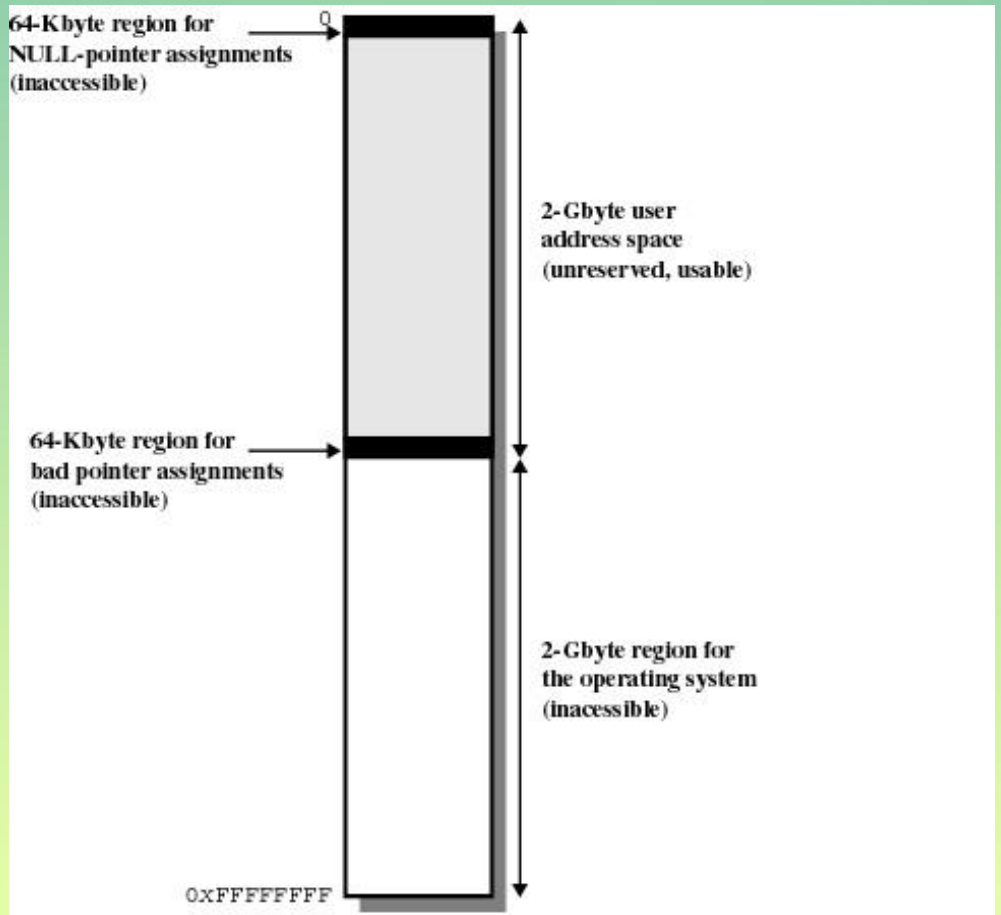


Figure 8.25 Windows 2000 Default Virtual Address Space