Sistemas Operativos

Memoria

Administración de Memoria y Memoria Virtual

Dr. Wenceslao Palma M.
<wenceslao.palma@ucv.cl>

La idea fundamental en la adm. de memoria es repartirla eficientemente para que almacene tantos procesos como sea posible.

Enlace de direcciones

Para ejecutar un proceso éste debe cargarse en memoria. Generalmente el archivo se encuentra en disco en un formato **binario ejecutable**.

Un programa de usuario pasa por varias etapas antes de ejecutarse. En estas etapas las direcciones pueden representarse de distintas maneras.

En el código fuente las direcciones generalmente son simbólicas. (int i;). El compilador enlazará estas direcciones simbólicas a direcciones relocalizables, por ejemplo; 4 bytes desde el inicio de este módulo. A su vez, el editor de enlaces o el cargador enlazará estas direcciones relocalizables con direcciones absolutas.

El enlace de instrucciones y datos con las direcciones casi siempre se pueden efectuar en cualquier momento de la vida de un proceso:

> Compilación: es posible generar código absoluto, es decir, a priori se sabe en que dirección de memoria residirá el proceso. Antiguamente los programas de DOS con formato .COM se enlazaban de manera absoluta durante la compilación.

Carga: el compilador genera código relocalizable.

Ejecución: cuando el proceso durante su ejecución puede moverse de un segmento de memoria a otro.

La administración de memoria debe satisfacer cinco requisitos:

- Reubicación.
- Protección.
- Compartición.
- Organización lógica.
- Organización Física.

Reubicación

Ya que no es posible determinar por adelantado qué otro proceso residirá en memoria y, además se busca poder cargar y descargar procesos activos. Seria bastante restrictivo que cuando nuevamente se cargue deba localizarse en la misma posición de memoria. En lugar de ellos se necesita **reubicar** el proceso en un área distinta.

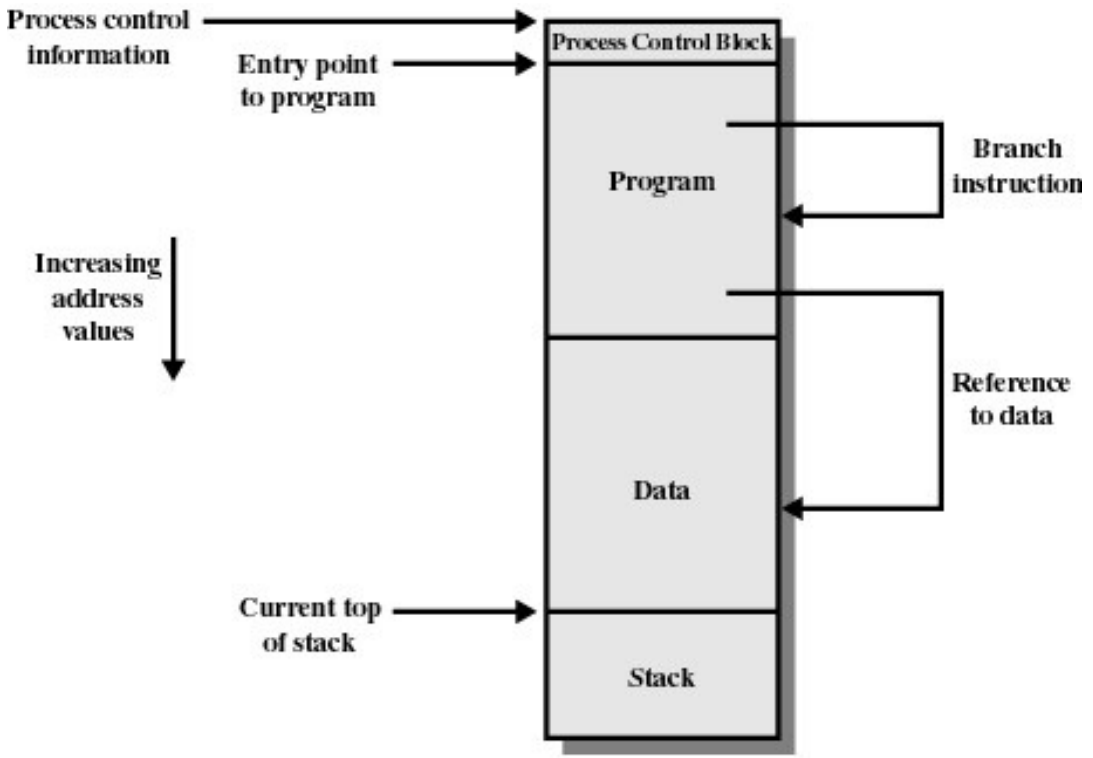


Figure 7.1 Addressing Requirements for a Process

Protección

El código de un proceso no puede hacer referencia (lectura/escritura) a posiciones de memoria a otros procesos sin permiso.

Debido a que se necesita satisfacer la reubicación aumenta la complejidad en la protección.

Además la mayoría de los lenguajes permite el cálculo dinámico de direcciones en tiempo de ejecución (por ejemplo; índices y punteros).

Por lo tanto la protección se debe realizar en tiempo de ejecución.

El procesador debe satisfacer las exigencias de protección de memoria.

Compartimiento

Se debe permitir que varios procesos puedan acceder a la misma zona de memoria sin comprometer la protección básica.

Por ejemplo, procesos que acceden a la misma zona de memoria mediante shmget(),shmat()

Organización Lógica

La memoria principal y la secundaria se organizan como un espacio de direcciones lineal o unidimensional. Esta organización refleja el hw de la máquina pero no la forma en que los usuarios organizan sus programas. Lo cual generalmente es en base a módulos.

Si esa organización puede ser manipulada por el SO y el hw, se consigue una serie de ventajas:

Los módulos ser codificados y compilados en forma independiente, el sistema resuelve durante la ejecución las referencias a otro módulo.

Es posible otorgar grados de protección.

Es posible diseñar mecanismos por medio de los cuales los procesos puedan compartir módulos (muy cercano a la visión del usuario).

Organización física

El esquema que se utiliza es de dos niveles; memoria principal + memoria secundaria.

Por lo tanto la organización del flujo de información entre memoria principal y secundaria es de gran interés.

La responsabilidad no se puede traspasar a la codificación.

Para admistrar la memoria existen distintas técnicas tales como:

- Partición Estática.
- Partición dinámica.
- Paginación simple.
- Segmentación simple.
- Memoria virtual paginada.
- Memoria Virtual segmentada.

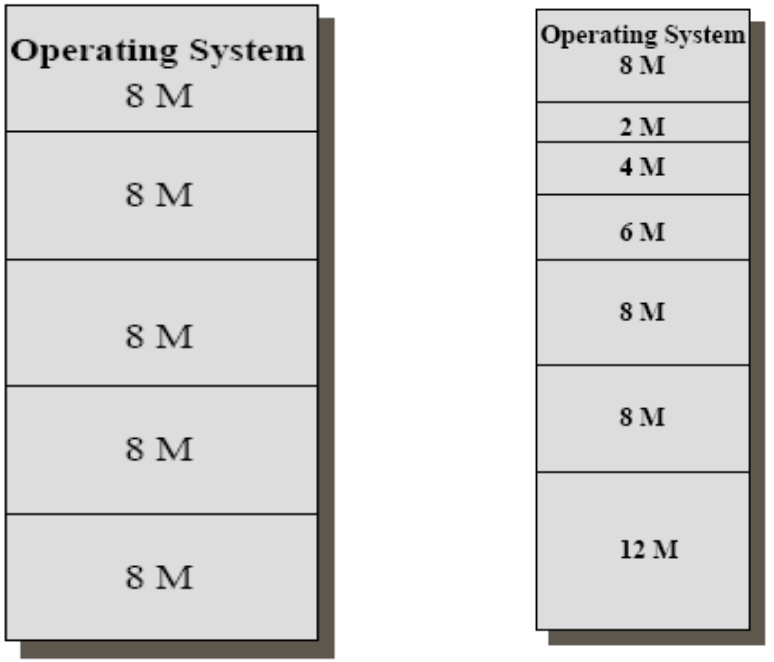
Partición de la memoria

Esta técnica se encuentra obsoleta, sin embargo es útil para introducir ideas importantes.

Particiones estáticas

Es el esquema más sencillo, se asume que el SO ocupa una parte fija de la memoria principal y el resto está disponible para los procesos en regiones con límites fijos.

Algo que se debe determinar es el tamaño de la partición



Las particiones estáticas de igual tamaño generan las siguientes dificultades:

Si el programa es más grande que la partición debe ser diseñado para el uso de overlay. De este modo sólo una parte del programa se encuentra en la memoria.

Si el programa es muy pequeño ocupará una partición completa. Esto lleva a una utilización ineficiente de la memoria. Cuando se malgasta el espacio interno de una partición se denomina **fragmentación interna**.

Algoritmo de ubicación

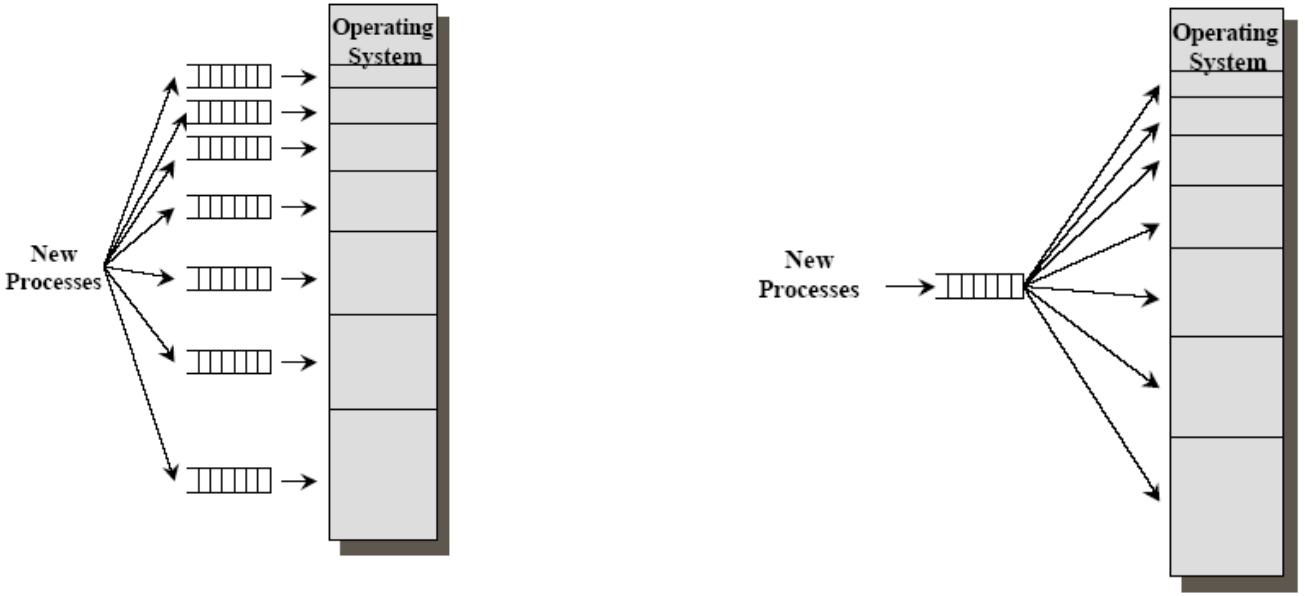
Con particiones del mismo la ubicación de un proceso en memoria no es problema. Si hay una partición libre entonces es posible cargar un proceso.

Si un proceso ingresa al sistema y todas las particiones estàn ocupadas y algunas de ellas con procesos que no están listos para ejecutarse, entonces uno de esos procesos es sacado para dar lugar al nuevo. Cuál se expulsa es un tema de scheduling.

Cuando las particiones son de distinto tamaño, hay dos formas de asignar procesos a particiones.

Cuando las particiones son de distinto tamaño, hay dos formas de asignar procesos a particiones.

La más simple consiste en asignar cada proceso a la partición más pequeña en donde pueda calzar, lo cual miminiza la fragmentación interna. Se necesita una cola de planificación para cada partición.



El problema se presenta cuando hay procesos que no tiene una partición más pequeña en donde ser almacenados y se desperdicia una más grande.

Si bien al utilizar particiones de distinto tamaño se necesita una sobrecarga de procesamiento mínima, se tienen las siguientes desventajas:

- El número de particiones generadas al iniciarse el sistema limita el número de procesos activos en el sistema.

- Los procesos pequeños no hacen uso eficiente del espacio de las particiones.

El uso de particiones estáticas es casi nulo en la actualidad, se ocupaba en el OS/MFT de IBM.

Particiones Dinámicas

Las particiones varían en número y longitud. Cuando se carga un proceso se le asigna exactamente la memoria que necesita y no más.

Este método comienza bien pero finalmente degenera en una situación en donde la adm. del espacio disponible es cada vez más ineficiente.

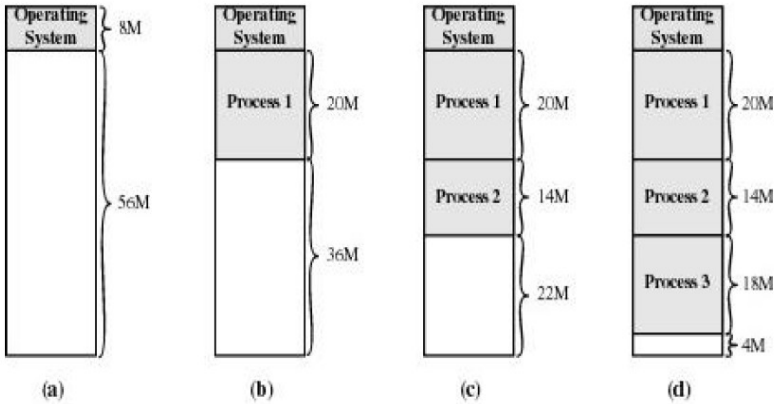


Figure 7.4 The Effect of Dynamic Partitioning

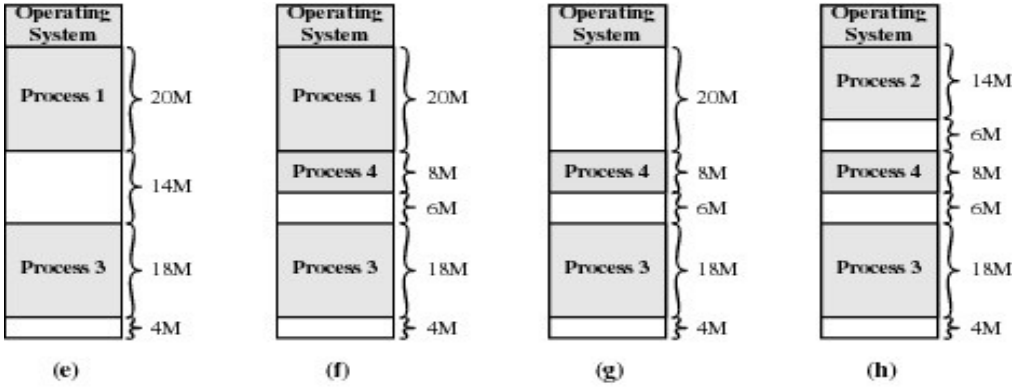


Figure 7.4 The Effect of Dynamic Partitioning

La situación que se genera se conoce como **fragmentación externa**; la memoria externa a cada partición se fragmenta cada vez más.

Una técnica para resolver la fragmentación externa es la compactación. Cada “cierto tiempo” el SO recorre la memoria y desplaza los procesos para que estén contiguos, lo cual genera una zona que puede ser mejor aprovechada.

Algoritmo de ubicación

Básicamente se consideran los algoritmos best-fit, first-fit y next-fit.

best-fit escoge el bloque de tamaño más próximo a lo solicitado.

first-fit recorre la memoria y escoge el primer bloque que sea lo suficientemente grande para almacenar el proceso.

next-fit escoge el siguiente bloque disponible a partir de la última ubicación.

Estos algoritmos son dependientes de la secuencia de intercambio de procesos y del tamaño de estos procesos.

first-fit es el que obtiene un mejor desempeño, next-fit y best-fit necesitan de compactación en forma más frecuente.

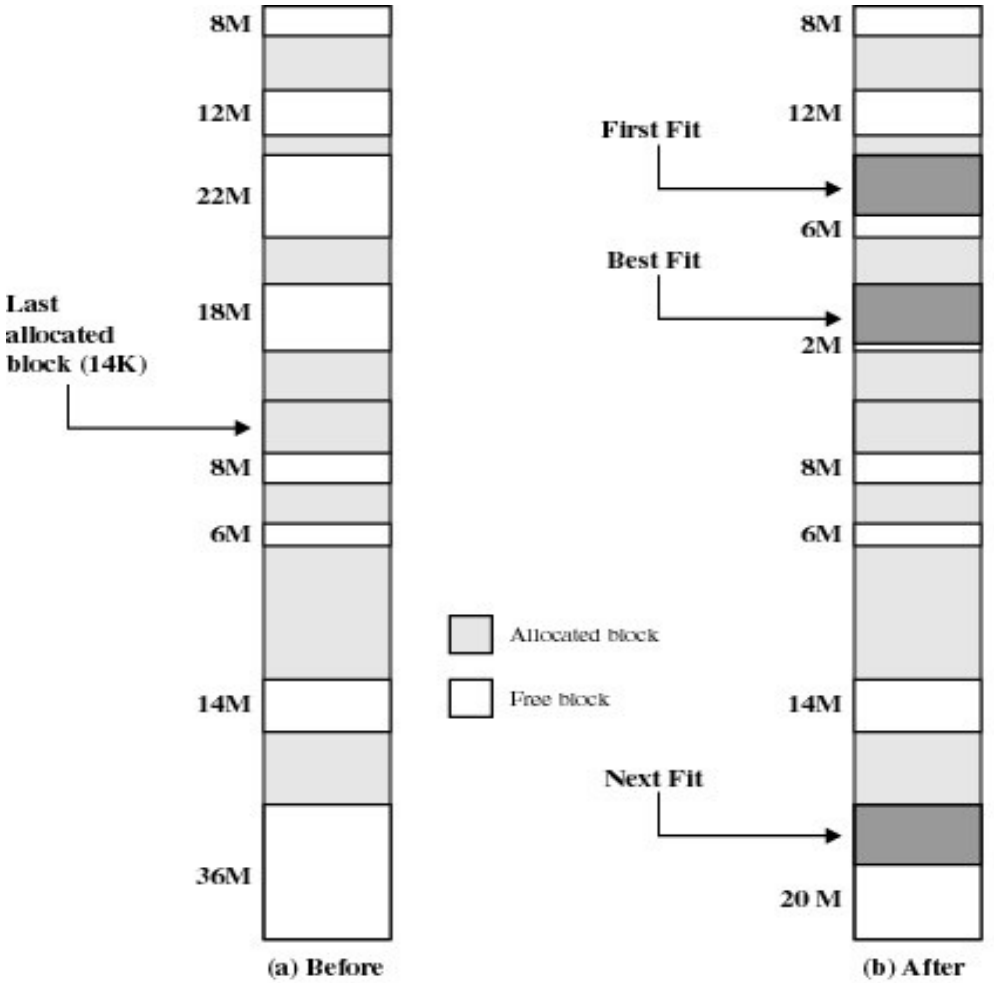


Figure 7.5 Example Memory Configuration Before and After Allocation of 16 Mbyte Block

Sistema de colegas

Una propuesta que trata de equilibrar las desventajas de las particiones estaticas y dinamicas es el Sistema de Colegas.

En esta propuesta los bloques de memoria disponible son de tamaño 2^k para valores K tal que L<=K<=U y donde:

2^L: tamaño de bloque asignable mas pequeño
2^U: tamaño de bloque asignable mas grande; generalmente es el tamaño de la memoria disponible para asignacion.

Al inicio el espacio disponible se trata como un solo bloque de tamaño 2^U. Al llegar una solicitud de tamaño s tal que $2^{U-1}<s<=2^U$, entonces el bloque entero es asignado. En otro caso el bloque se divide en dos colegas de tamaño 2^{U-1}. Si $2^{U-}2<s<2^{U-1}$ la solicitud se asigna a uno de los dos colegas. Si no, nuevamente uno de los colegas se divide por la mitad.

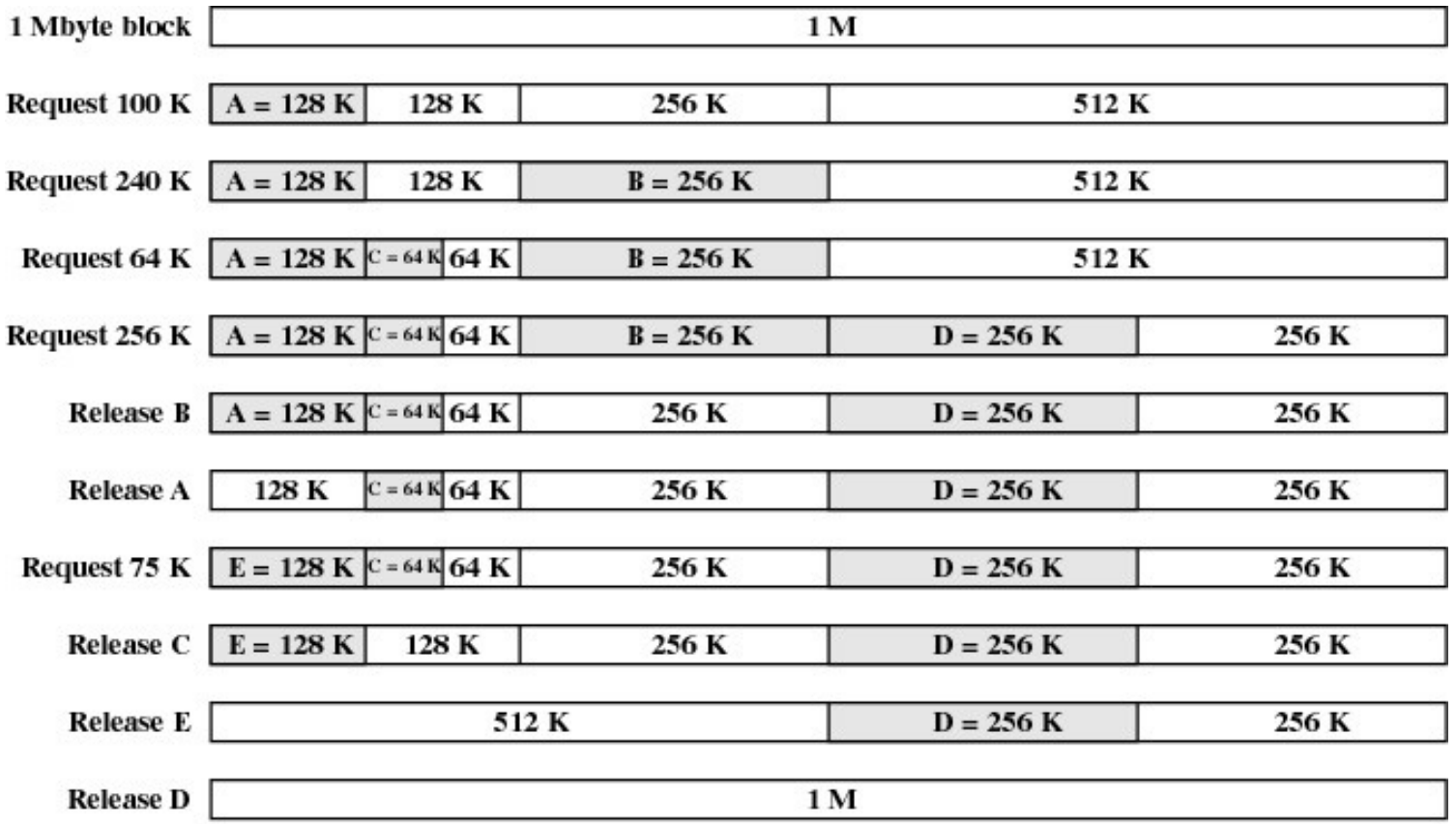


Figure 7.6 Example of Buddy System

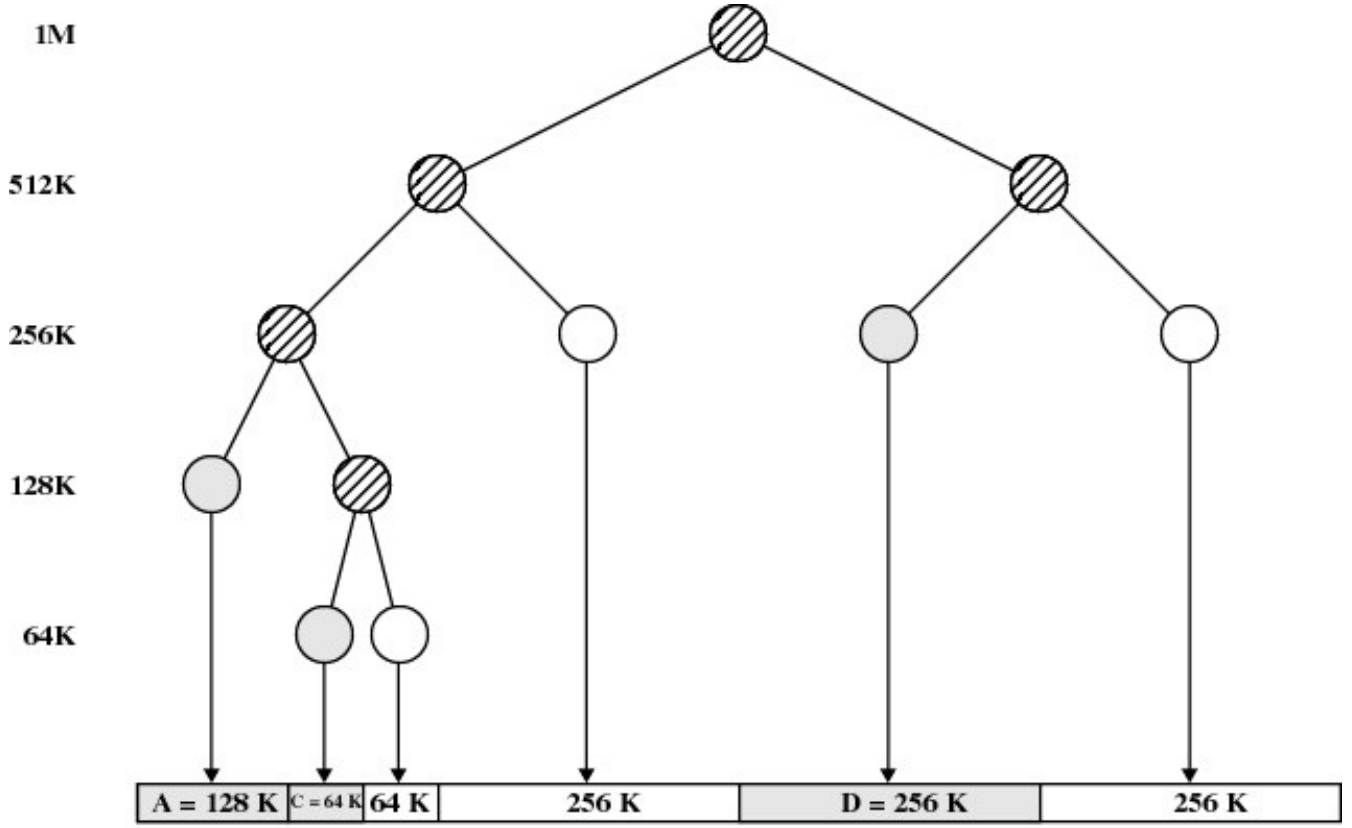


Figure 7.7 Tree Representation of Buddy System

Reubicación

Ya que la ubicación de las instrucciones y datos de un proceso cambian cada vez que el proceso es desplazado (compactación) o se intercambia, se realiza una distinción entre varios tipos de direcciones:

Dirección lógica: es una referencia a una posición de memoria independiente de la asignacion actual, es necesario traducirla a una dirección física.

Dirección relativa: caso particular de dirección lógica, es una posición relativa a algun punto conocido.

Dirección física: posición real en la memoria.

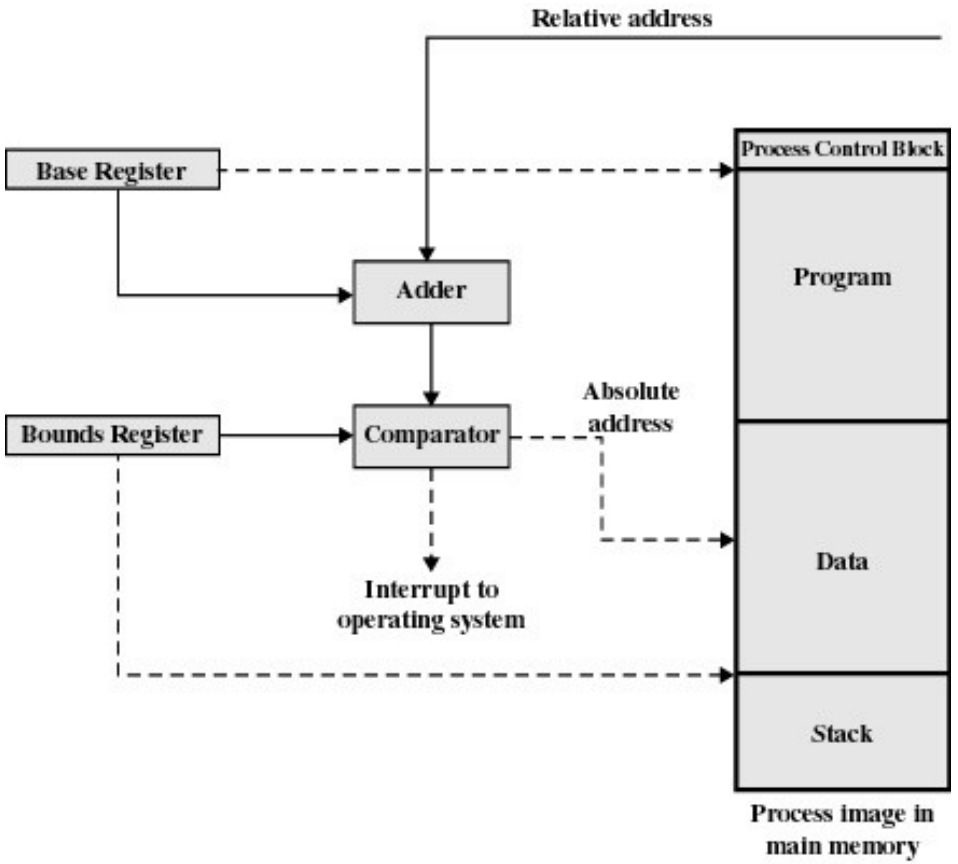


Figure 7.8 Hardware Support for Relocation

Paginación

Recordemos que las particiones de tamaño fijo generan fragmentación interna y que las de tamaño variable generan fragmentación externa.

Supongamos que la memoria se encuentra dividida en trozos relativamente pequeños de igual tamaño y que cada proceso se encuentra dividido en trozos pequeños iguales a los de la memoria.

Los trozos de proceso se llaman páginas y son asignados a trozos libres de memoria llamados frames.

El sistema operativo debe mantener una lista de frames disponibles.

Que no exista memoria disponible en forma contigua no es impedimento para cargar un proceso. La referencia a los frames asignados se almacenan en una tabla de páginas.

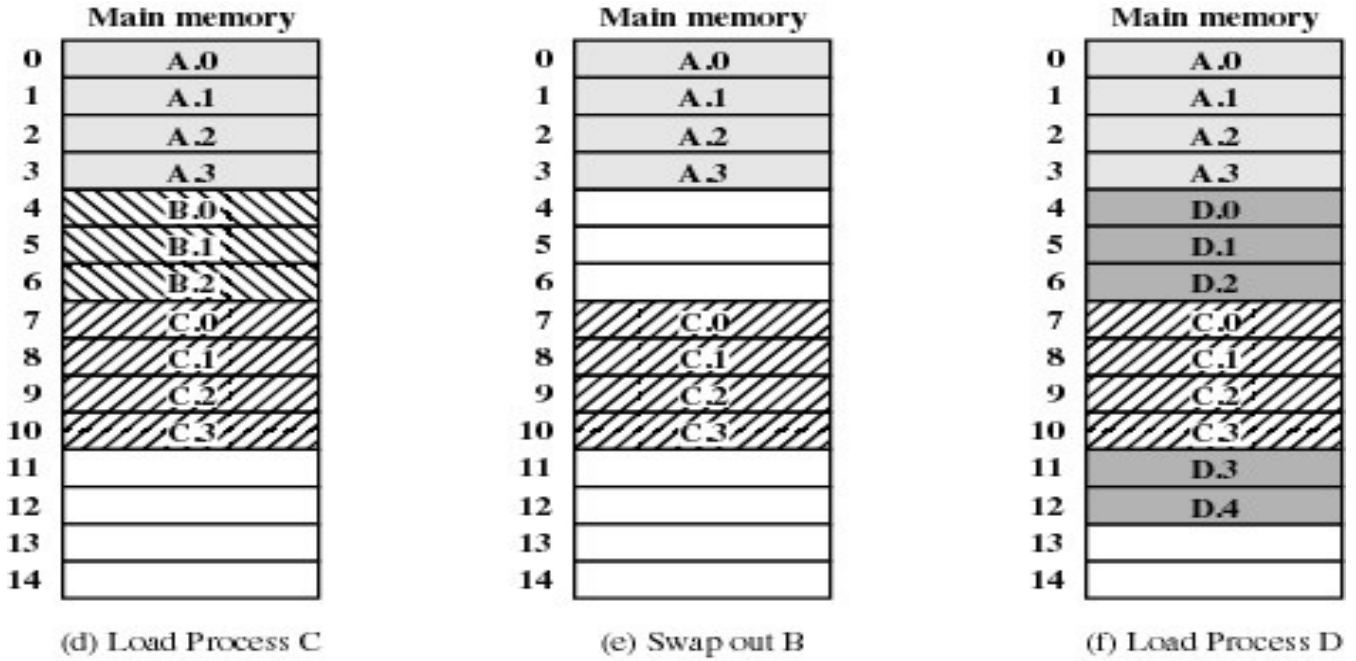


Figure 7.9 Assignment of Process Pages to Free Frames

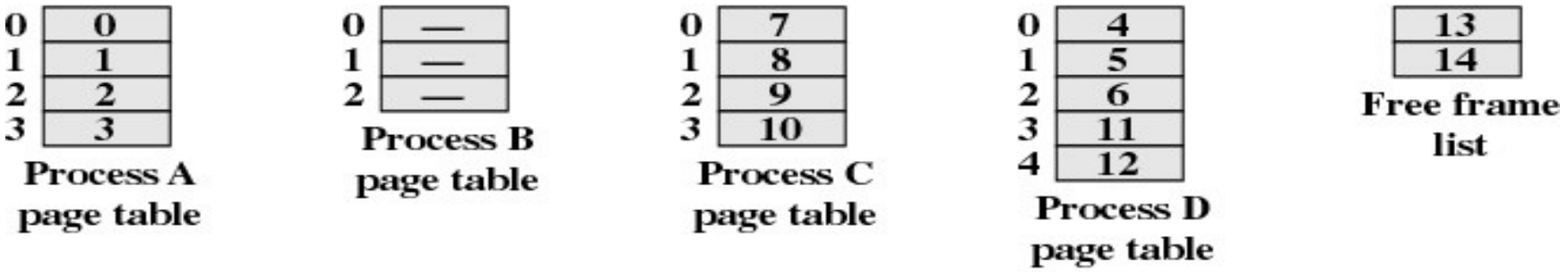


Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)

Resulta conveniente que tanto el tamaño de la página como del frame sean potencias de 2.

Una dirección está compuesta de n+m bits, en donde los n bits más significativos son el número de página y los m bits restantes el desplazamiento. Luego, para traducir una dirección lógica a física hay que realizar lo siguiente:

Extraer el número de página de la dirección lógica.

Emplear el número de página como índice en la tabla de páginas para encontrar el número del frame k.

El comienzo de la dirección del frame es $k*2^m$ y la dirección física del byte referenciado es este número más el desplazamiento. No es necesario calcularla sino que se construye concatenando.

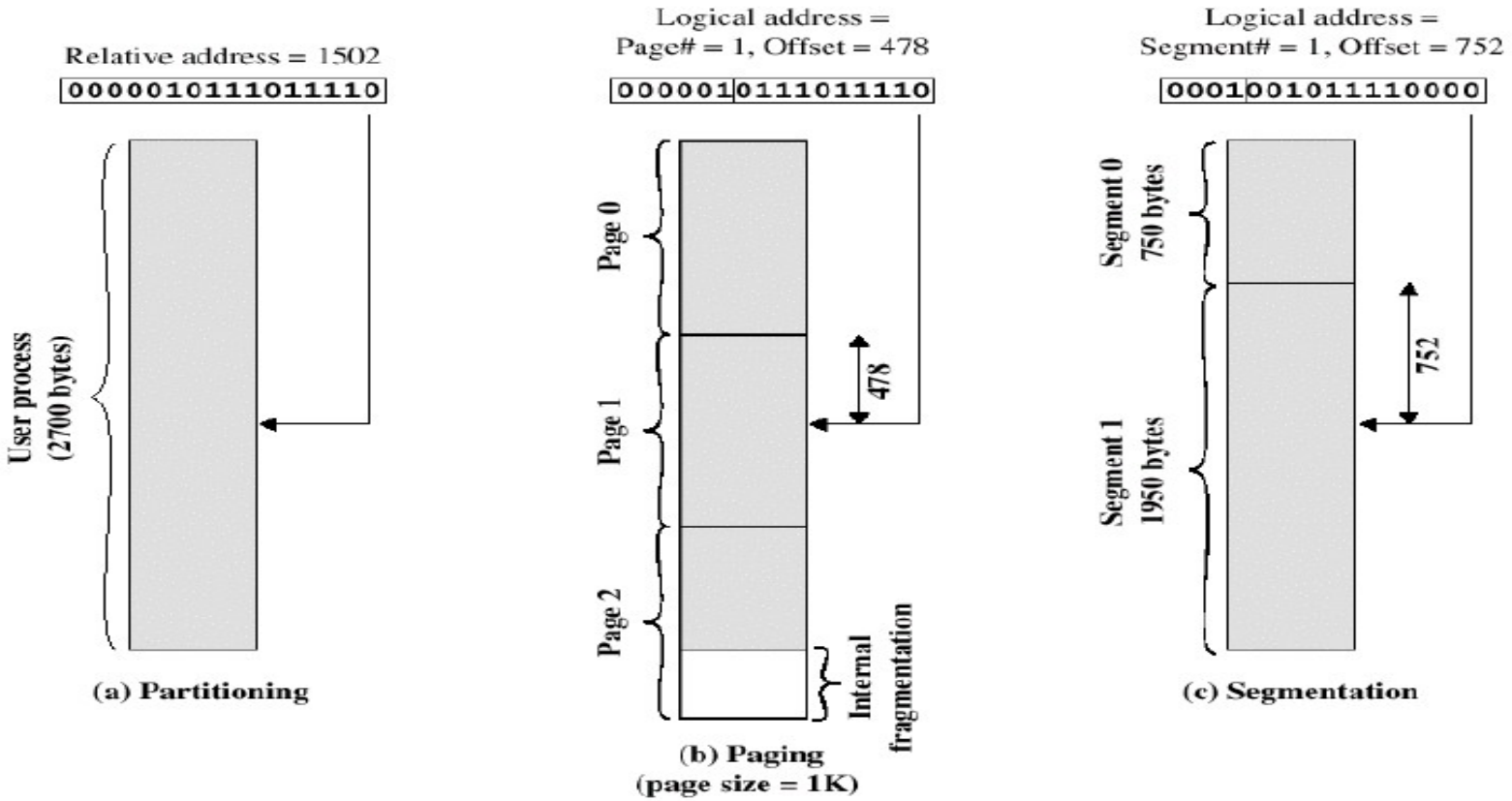


Figure 7.11 Logical addresses

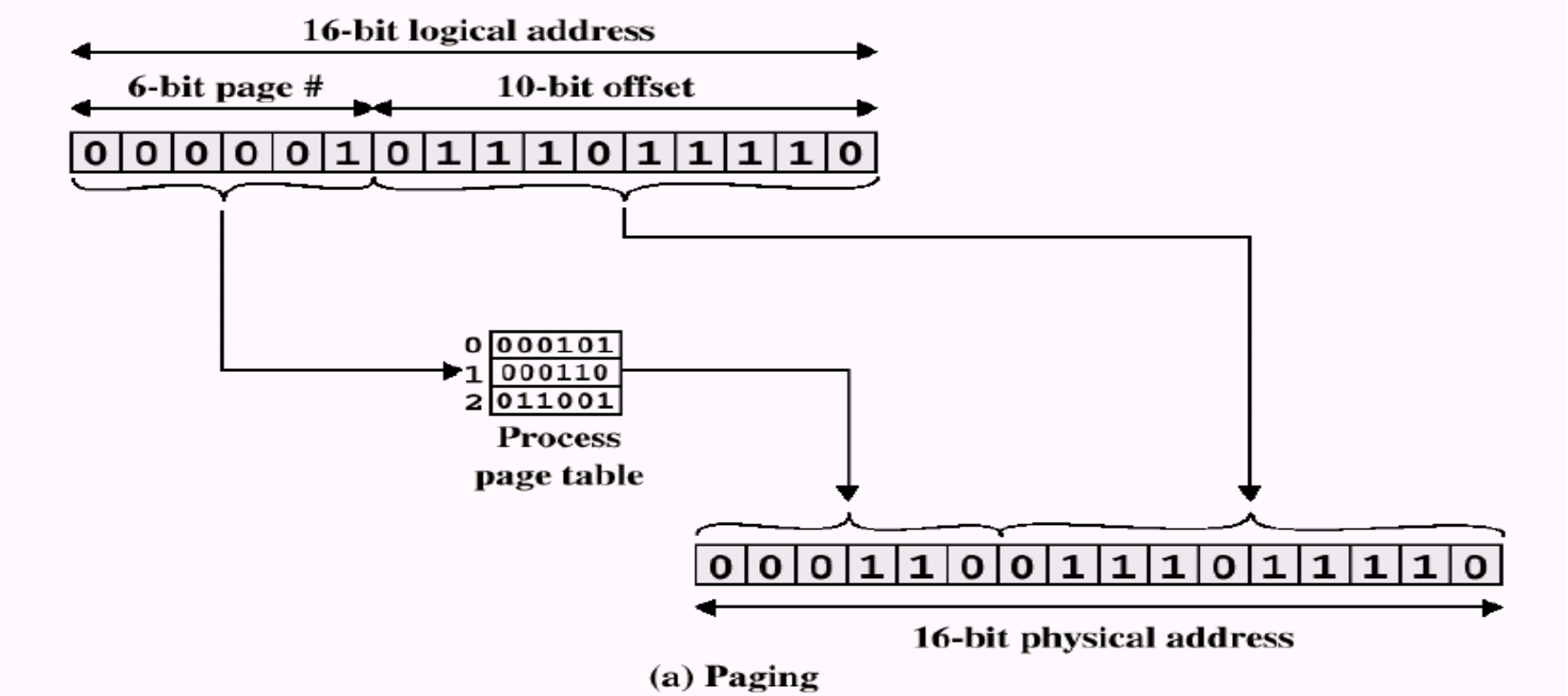


Figure 7.12 Examples of Logical-to-Physical address translation

Segmentación

Otra forma de dividir un proceso es la segmentación. No es necesario que todos los segmentos tengan la misma longitud, aunque existe una longitud máxima de segmento.

Asi como en la paginación, una dirección lógica segmentada consta de dos partes; número de segmento y un desplazamiento.

Debido a que los segmentos son de distinto tamaño la segmentación resulta similar a la partición dinámica, sin embargo la fragmentación externa será menor debido a que se utilizan páginas pequeñas.

A diferencia de la paginación, la segmentación es visible para quien programe. Se ofrece como una ventaja para la organización de programas y datos.

Al existir un tamaño desigual no existe una correspondencia simple entre dirección lógica y física.

El sistema operativo mantiene una tabla de segmentos para cada proceso y una lista de segmentos libres. Cada entrada en la tabla contiene la dirección de inicio del segmento en la memoria y su largo.

Cuando un proceso pasa al estado “running” se carga la dirección de su tabla de segmentos en un registro especial del hw de gestión de memoria.

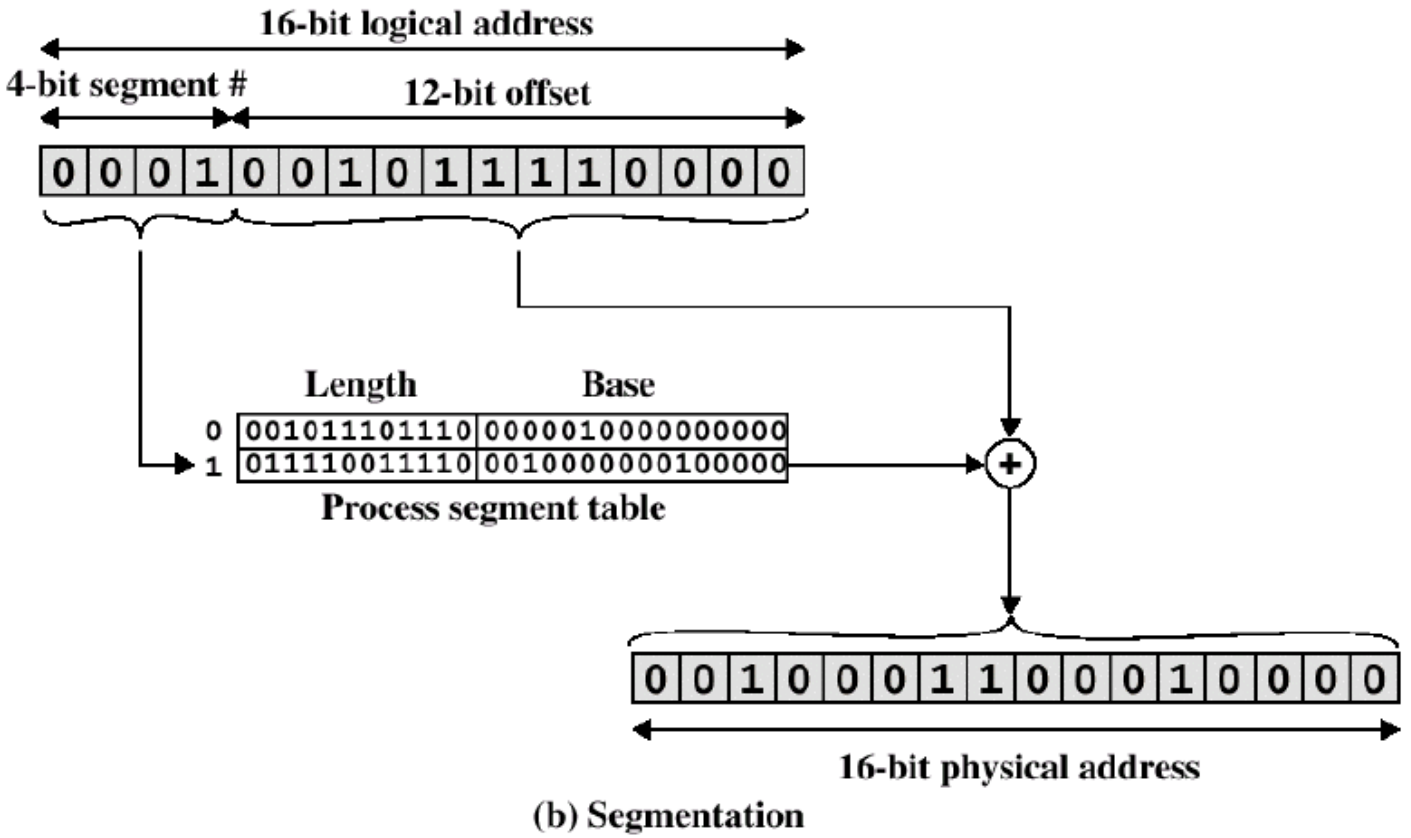
Si tenemos una dirección de n+m bits. El tamaño mas grande de segmento sera 2^m. Para la traducción, se necesitan los siguientes pasos:

Extraer el número de segmento de los n bits.

Tomar el número de página como índice en la tabla de segmentos para encontrar la dirección física del inicio del segmento.

Comparar el desplazamiento de m bits con la longitud del segmento.

La dirección física es la suma de la dirección física del comienzo del segmento mas el desplazamiento.



(b) Segmentation

Figure 7.12 Examples of Logical-to-Physical address translation

Memoria Virtual

No es necesario que durante la ejecución de un procesos todas sus páginas o segmentos se encuentren en la memoria.

La parte de un proceso que se encuentra en memoria se llama conjunto residente. Cuando el procesador detecta una dirección lógica que no está en memoria principal se genera una interrupción que indica un fallo de acceso a la memoria.

El sistema operativo deja al proceso en estado sleep. Para que el proceso pueda continuar es necesario traer a memoria el fragmento (página, segmento) que contiene la dirección lógica que provocó el fallo de acceso.

Para lograr lo anterior el SO emite una solicitud de lectura al disco, se termina la interrupción de E/S, el SO toma el control y deja al proceso en estado ready.

Debido a lo anterior es posible tener mas procesos en memoria principal y el tamaño de los procesos puede ser mas grande que toda la memoria principal .

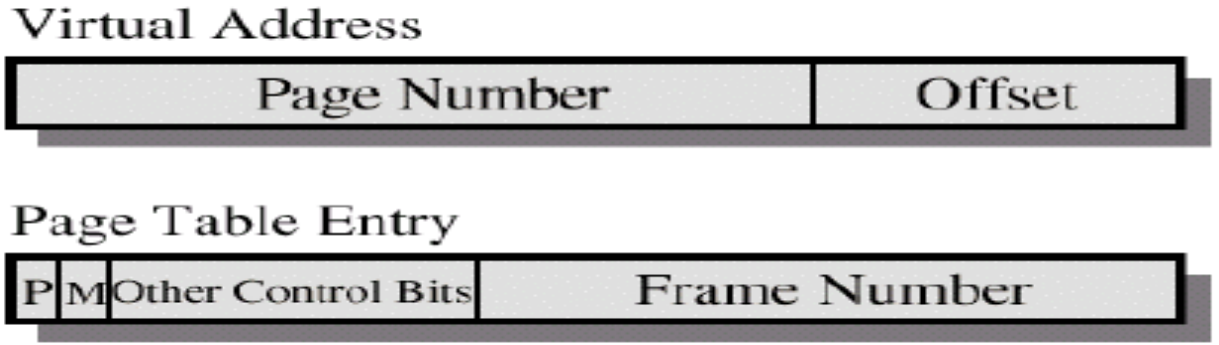
Un gran número de intercambio de fragmentos generan lo que se conoce como hiperpaginación (thrashing); en este fenómeno el procesador pierde tiempo intercambiando fragmentos en vez de ejecutar instrucciones de los procesos.

Para evitar la hiperpaginación el SO de algun modo debe intentar adivinar que fragmentos se usaran con menor probabilidad en un futuro proximo.

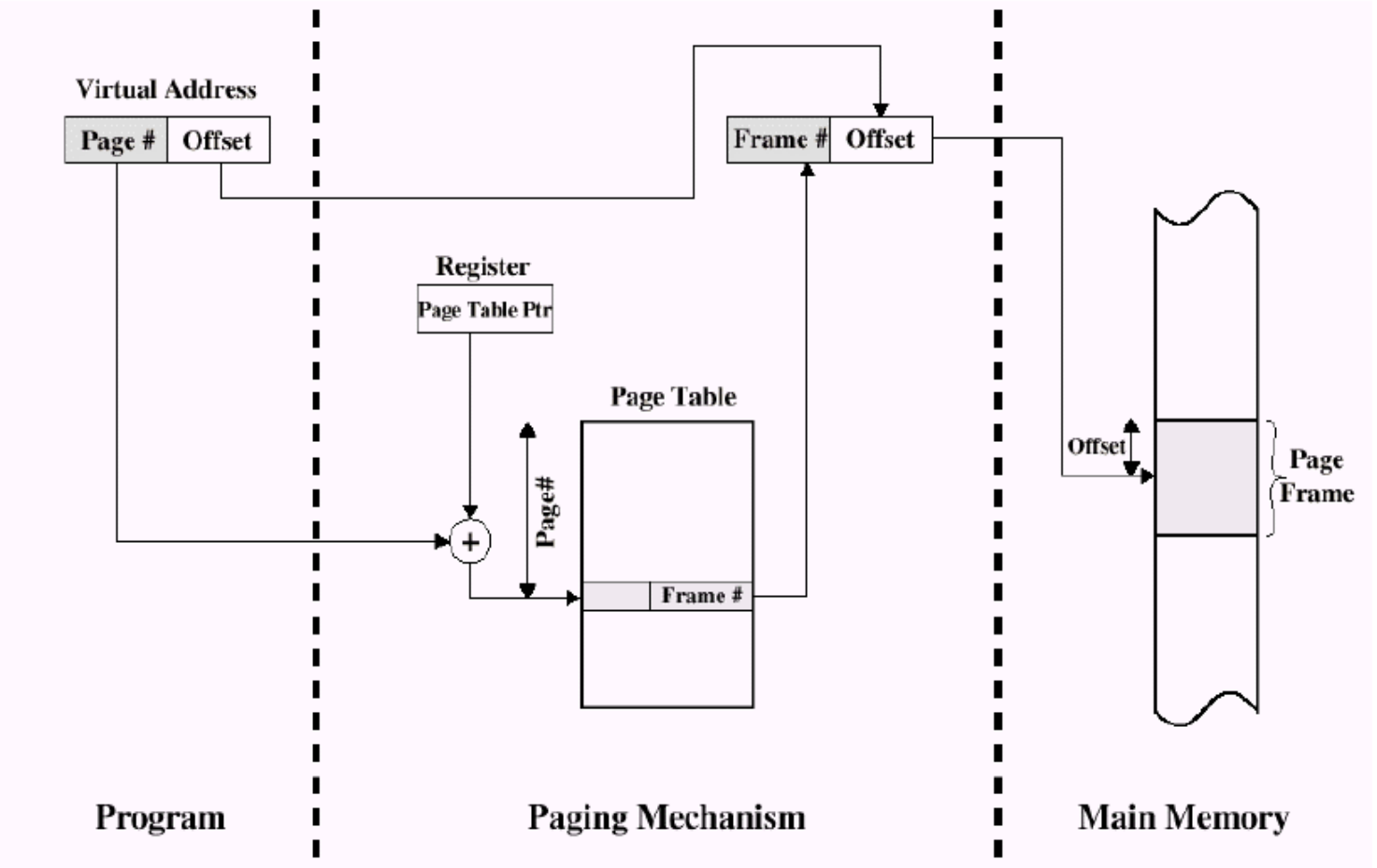
Las referencias a los datos y al programa tienden a agruparse, esto se conoce como principio de cercania.

Paginación

Cuando se tiene memoria virtual basada en paginación se necesita una tabla de páginas. Sin embargo sus entradas pasan a ser mas complejas. El bit P indica si la página se encuentra en memoria. Por su parte el bit M indica si la página ha sido modificada desde su carga.



Traducción de direcciones en un sistema de paginación



Estructura de la tabla de páginas

En la mayoría de los sistemas hay una tabla de página por proceso.

En arquitectura VAX cada proceso puede ser de hasta 2GB de memoria. Con páginas de 512Bytes se necesitan 2^{22} entradas en la tabla de páginas!!

Para solucionar lo anterior se almacena la tabla de páginas en memoria virtual. Si un proceso esta en ejecución se carga parte de su tabla en memoria.

Estructura de dos niveles

Considerar direcciones de 32 bits, páginas de 4KB y espacio de 4GB.

La tabla tendría 2^{20} entradas. Si cada entrada en la tabla de páginas está compuesta de 4 bytes, la tabla ocupa 2^{22} Bytes (4MB).

Ésta tabla podría a su vez podría dividirse en páginas de 4KB y almacenarse en disco.

Para acceder a la información de la tabla almacenada en disco se crea una tabla raíz de páginas de 2^{10} entradas residente en memoria.

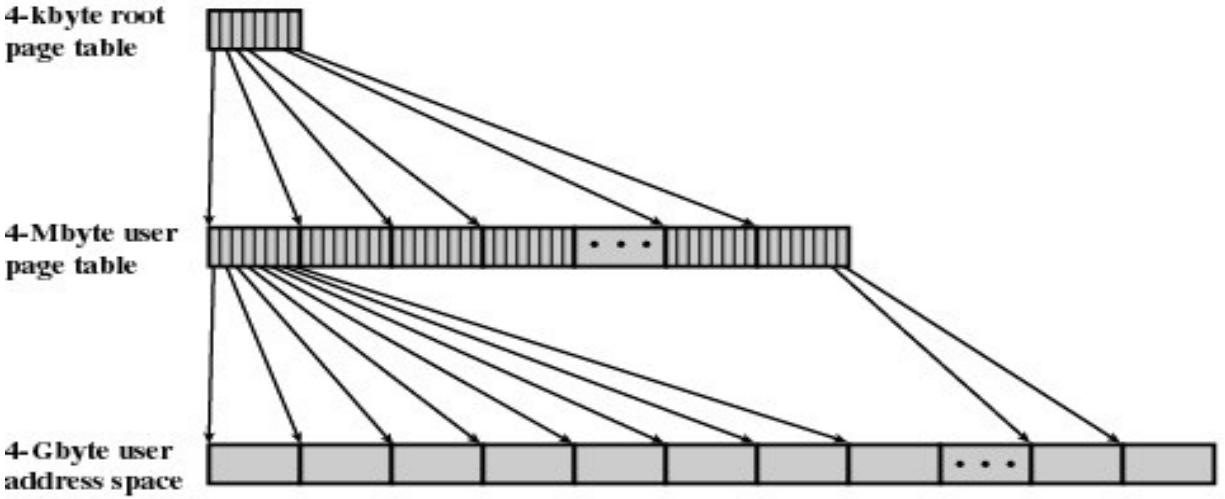
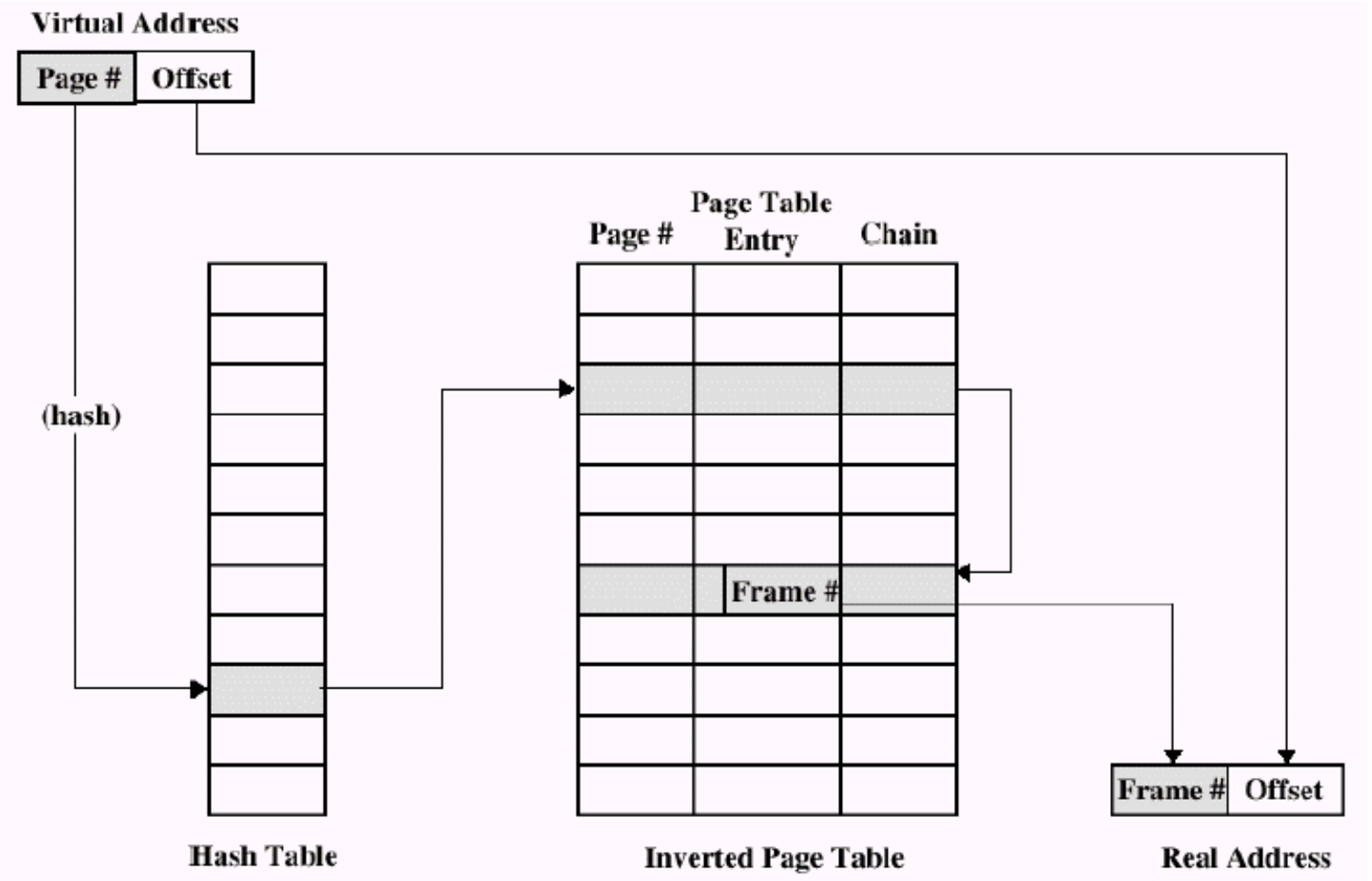


Figure 8.4 A Two-Level Hierarchical Page Table [JACO98a]

Tabla de páginas invertida

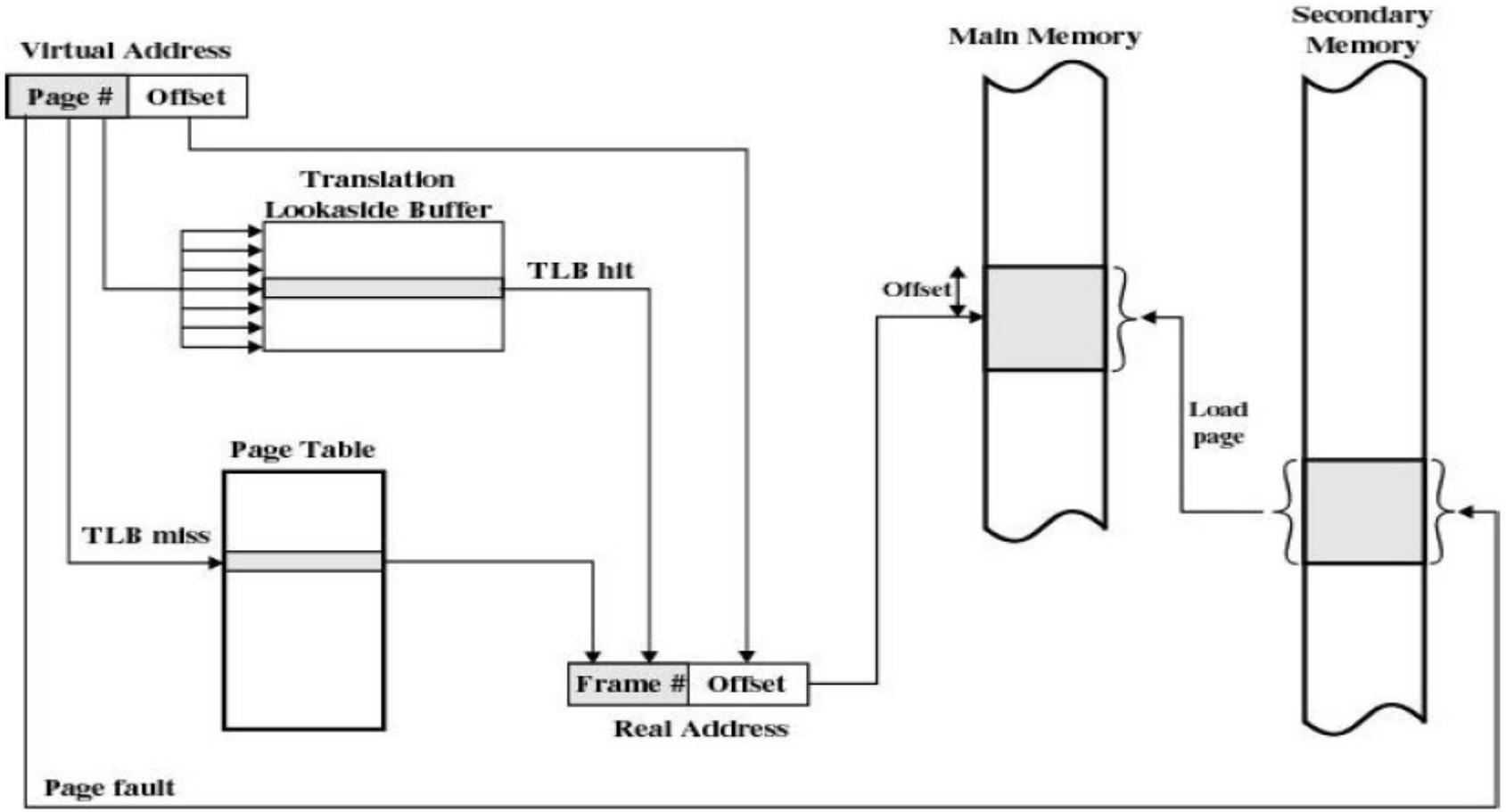
Empleada en PowerPC y AS/400 de IBM.



Buffer de traducción adelantada (TLB, Translation Lookaside Buffer)

Cada referencia a memoria virtual puede generar dos accesos a memoria (tabla de página+dato).

Para evitar lo anterior se utiliza un buffer para almacenar los registros de la tabla de páginas recientemente utilizadas.



Tamaño de la página

Con la paginación un fenómeno no deseable es la fragmentación interna.

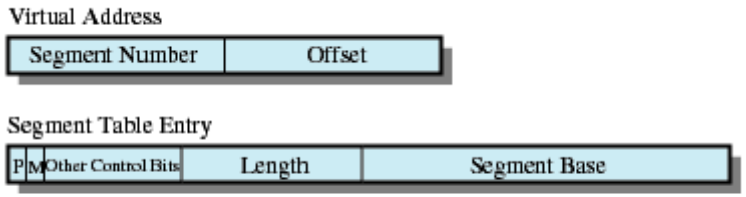
Con un tamaño de página pequeño se reduce la fragmentación interna pero se incrementa la posibilidad de uso de la memoria virtual con dos fallos de página relacionados con un acceso a memoria (tabla de página + página)

Segmentación

Referencias a las memoria son de la forma segmento+desplazamiento.

Los segmentos pueden ser de distinto tamaño.

Las entradas en la tabla de segmentos contienen: largo del segmento, dirección de inicio, bit de presencia en memoria, bit de modificación.



(b) Segmentation only

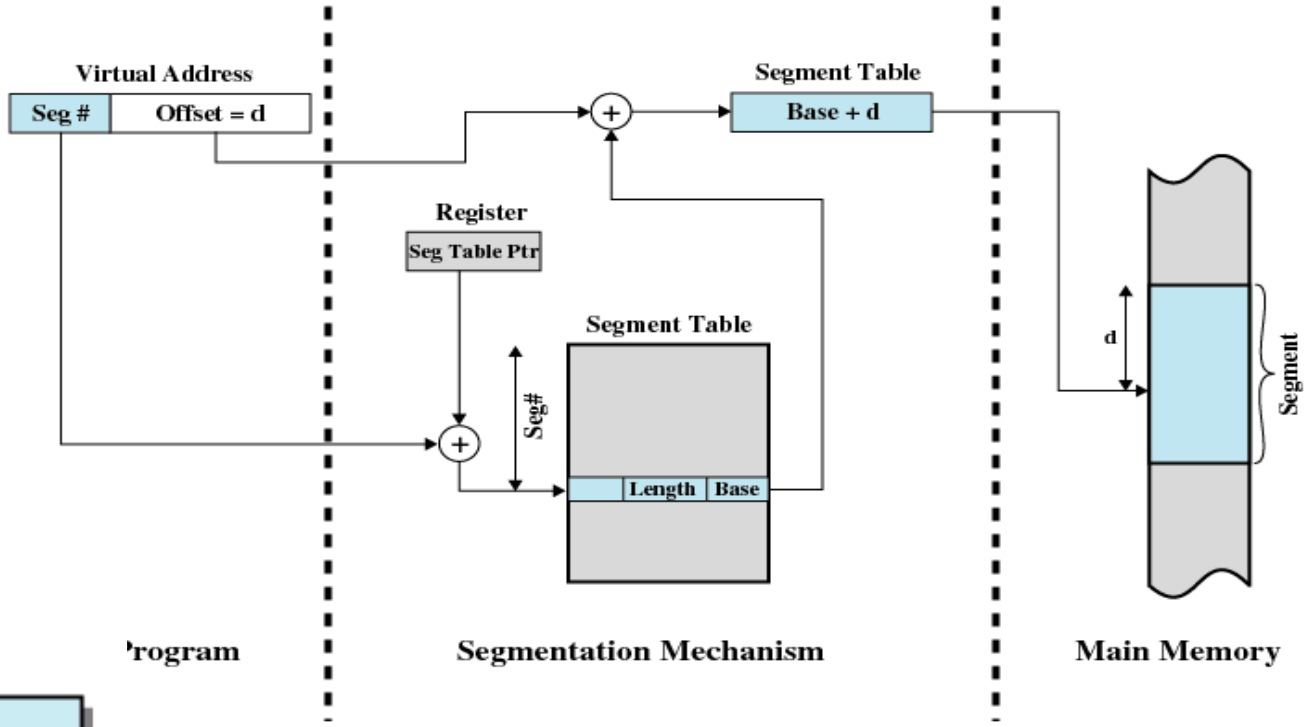


Figure 8.12 Address Translation in a Segmentation System

Paginación y segmentación combinadas

El espacio de direcciones de un usuario se divide en varios segmentos según el criterio del programador.

Cada segmento se divide en páginas de tamaño fijo igual al de un frame de memoria principal.

El segmento ocupará solo una página si su tamaño es menor que el de una página.

Para el programador una dirección esta formada por un segmento+desplazamiento.

Desde el punto de vista del sistema, el desplazamiento del segmento es un número de página dentro del segmento y un desplazamiento dentro de la página.

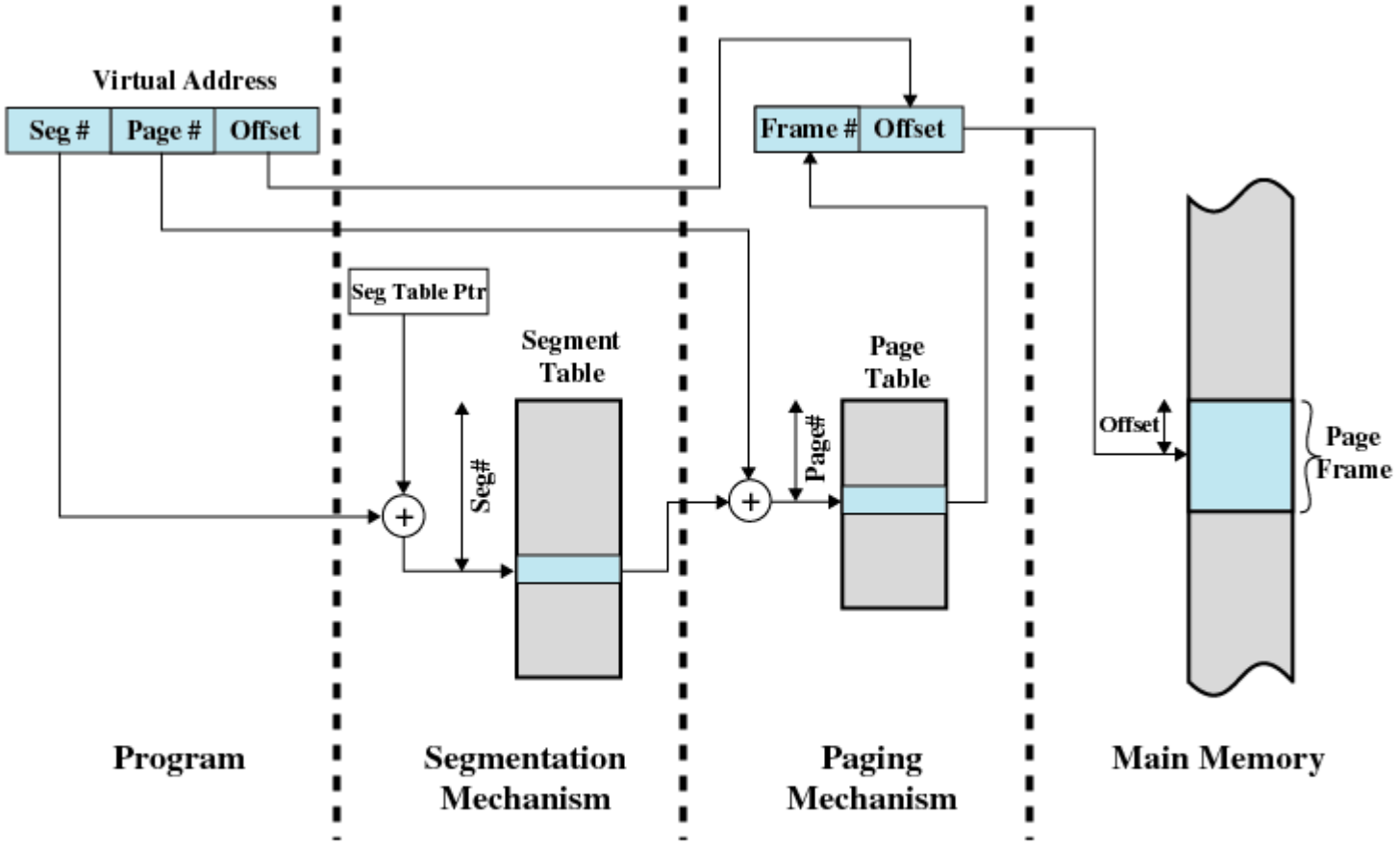


Figure 8.13 Address Translation in a Segmentation/Paging System

Politicas de sistema operativo sobre la memoria virtual

Politica de lectura (fetch)

Determina cuando una página debe ser cargada en memoria. Las dos alternativas son la paginación por demanda y la paginación previa.

En la paginación por demanda se producirán muchos fallos de página cuando el proceso se ejecuta por primera vez.

En la paginación previa se cargan otras páginas además de la que se debe cargar debido al fallo de página. Esto se basa en el hecho que en disco los procesos se almacenan en sectores contiguos.

Política de ubicación

Determina donde se ubicarán las partes del proceso en la memoria principal. En un sistema de segmentación eso es relevante.

En sistemas que usan paginación o paginación con segmentación es irrelevante.

Política de reemplazo

Cual es la página a reemplazar cuando se debe cargar una nueva página.?

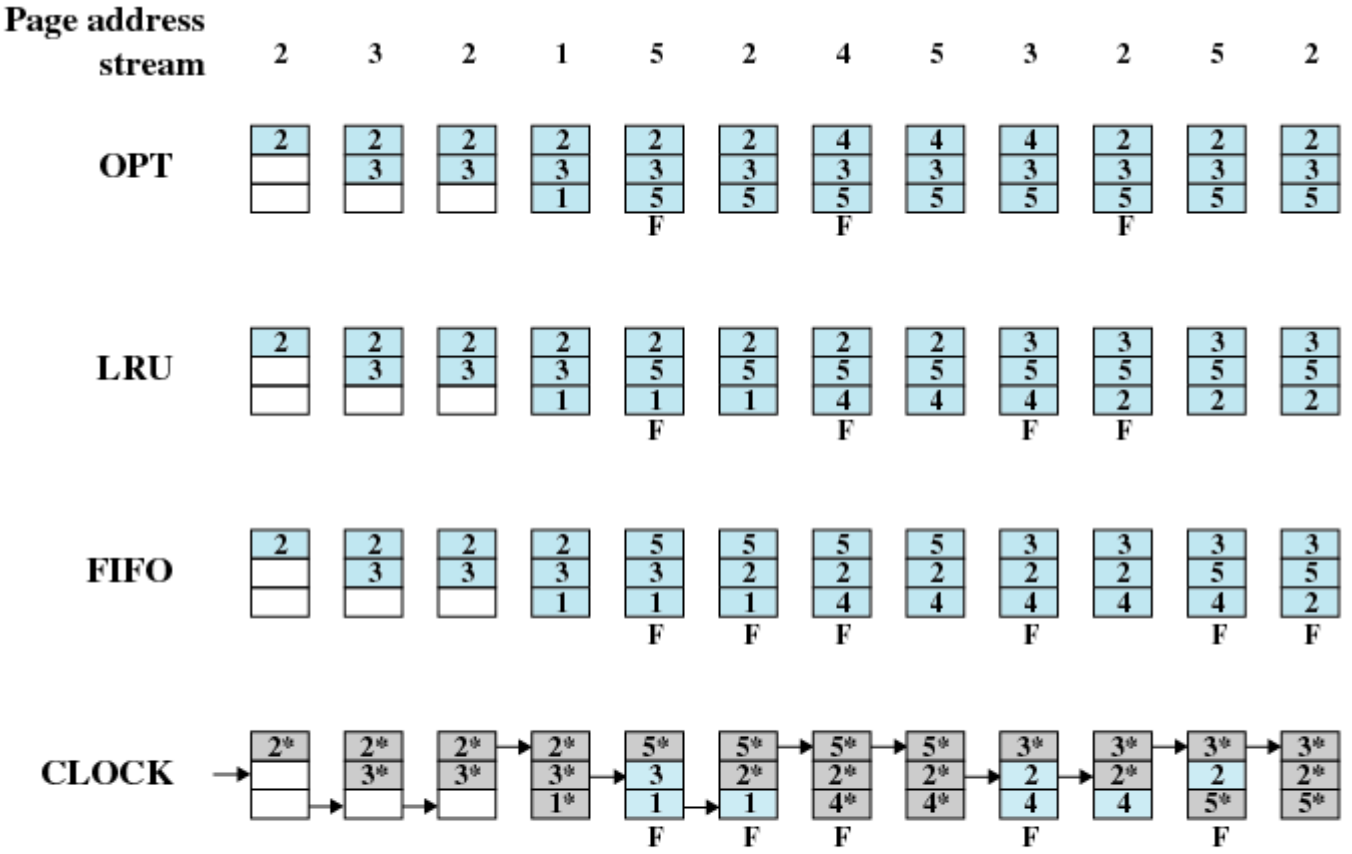
Todas las políticas tiene como objetivo reemplazar la página que tenga menor posibilidad de ser referenciada en un futuro cercano.

Mientras mas sofisticada sea la política de reemplazo, mayor será la sobrecarga de hardware y software utilizada para implementarla.

Una de las cosas que se debe tomar en cuenta es la posibilidad de que un frame se encuentre bloqueado (kernel,estructuras de control, buffers de I/O).

Algoritmos básicos

Óptimo: reemplaza la página que tiene que esperar una mayor cantidad de tiempo hasta que se produzca sobre ella otra referencia. Este algoritmo es imposible de implementar pero sirve para comparar los otros algoritmos.



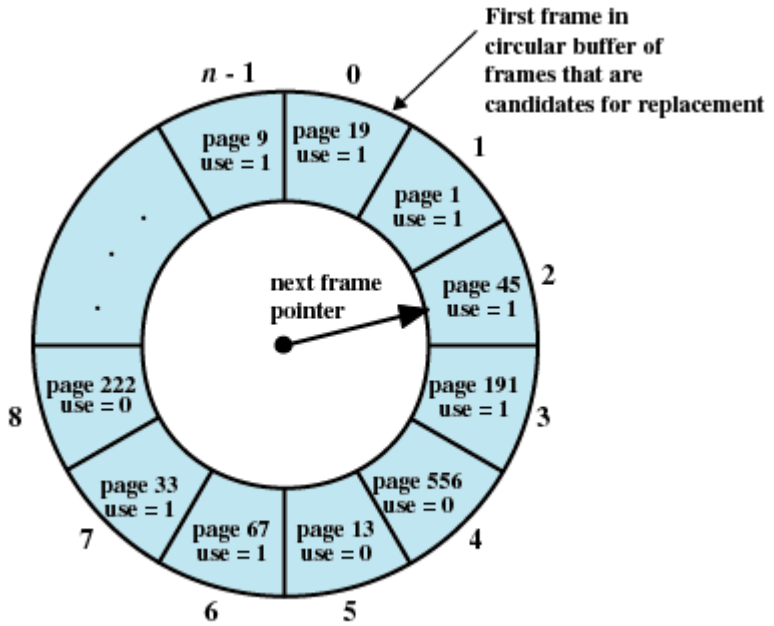
F = page fault occurring after the frame allocation is initially filled

Figure 8.15 Behavior of Four Page-Replacement Algorithms

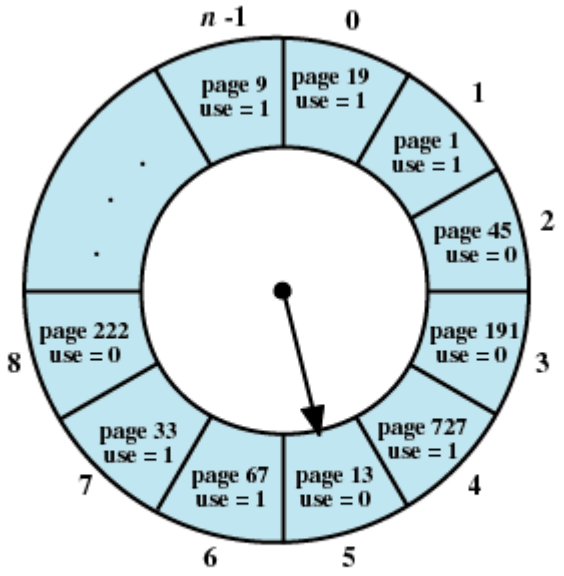
LRU: reemplaza la página de memoria que se ha mantenido por mas tiempo no referenciada. Es un método difícil de implementar, se necesitaria rotular cada pagina al momento de su última referencia, con la consecuente sobrecarga.

FIFO: la política es reemplazar la página que ha estado por mas tiempo en la memoria (puede haber caido en desuso). No considera que hay regiones de un programa o datos que son usados a lo largo de la vida de éste.

Política del Reloj: en su forma más simple requiere agregar un bit a cada frame (bit de uso). Cuando una página se carga por primera vez el bit de uso del frame se setea a uno. Un referencia posterior a ésta página setea el bit a uno. El conjunto de frames candidatos a ser reemplazados se considera como un buffer circular con un puntero. Al reemplazar una página, el puntero quedará en el siguiente frame. Cuando nuevamente llega el momento de reemplazar una página el SO recorre los frames buscando uno con el bit de uso en cero. Cada vez que encuentra un frame con el bit de uso a uno lo deja en cero. El primer frame que se encuentra con bit cero será reemplazado. Si todos los frames tienen el bit de uso en 1, el puntero dará la vuelta completa y reemplazará el inicial.



(a) State of buffer just prior to a page replacement



(b) State of buffer just after the next page replacement

Figure 8.16 Example of Clock Policy Operation

Almacenamiento intermedio de páginas

Su objetivo es mejorar el rendimiento de la paginación permitiendo el uso de una política de reemplazo mas sencilla.

En VAX/VMS el algoritmo es FIFO. Sin embargo, no se pierde la pista de la página reemplazada asignándose a una de las siguientes listas; lista de páginas libres o lista de páginas modificadas.

Lo importante es que la página no abandona la memoria, solo se suprime su entrada en la tabla de páginas.

Las listas antes mencionadas actúan como un caché.

Gestión del conjunto residente

Tamaño del conjunto residente

Considerando que:

- cuanto menor sea la cantidad de memoria asignada a un proceso, mayor es el número de procesos que pueden estar en la memoria principal.

- si el conjunto residente es muy pequeño, el porcentaje de fallos de página sera mayor.

- la asignación de memoria adicional a un proceso no tendrá efectos notables en su porcentaje de fallos de página.

Existen dos tipos de politicas para la gestión del conjunto residente; asignación fija y asignación variable.

Política de vaciado

Es contraria a la política de lectura ya que determina el momento en el cual hay que escribir en memoria secundaria una página modificada.

Existen dos políticas: vaciado por demanda y vaciado previo.

En el vaciado por demanda la página se escribe solo cuando ha sido escogida para reemplazo. Un proceso tiene que esperar dos transferencias de página antes de desbloquearse, lo cual disminuye el aprovechamiento del procesador.

En el vaciado previo, las páginas se escriben antes que se necesite el frame donde se almacenan. Permite escribir las páginas por lotes, lo cual es bueno desde el punto de vista del rendimiento pero las escrituras pueden ser innecesarias.

Una mejor solución es incorporar el almacenamiento intermedio de páginas.

Control de carga

Determina la cantidad de procesos que pueden estar en la memoria (grado de multiprogramación). Su funcionamiento es crítico para una administración exitosa de la memoria.

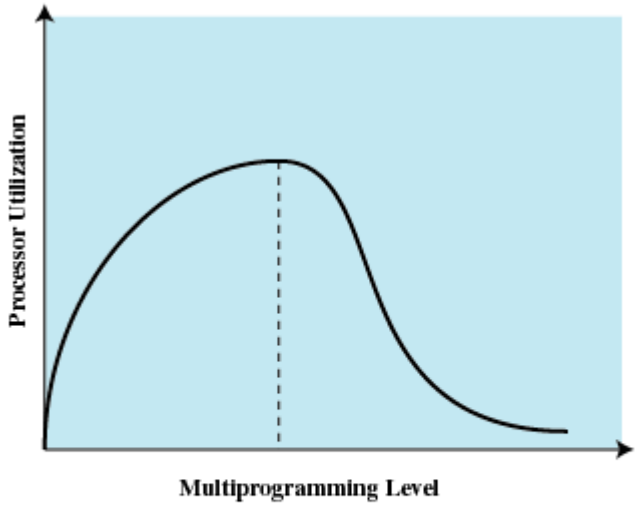


Figure 8.21 Multiprogramming Effects

Existen criterios para evitar la hiperpaginación. Uno de ellos consiste en ajustar el grado de multiprogramación considerando que el tiempo medio entre fallos sea igual al tiempo medio exigido para procesar un fallo de página.

Suspensión de procesos

Una reducción del grado de multiprogramación implica la suspensión de uno o mas procesos residentes. Las criterios para la elección son los siguientes:

- Procesos con la prioridad mas baja.
- Procesos con fallos de página.
- Ultimo proceso activado.
- Proceso con el conjunto residente mas pequeño.
- El proceso mayor.
- Procesos con la mayor ventana de ejecución restante.