Sistemas operativos

Gestión de Memoria

Requisitos para la Gestión de Memoria

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

Gestión de Memoria

- Reubicación

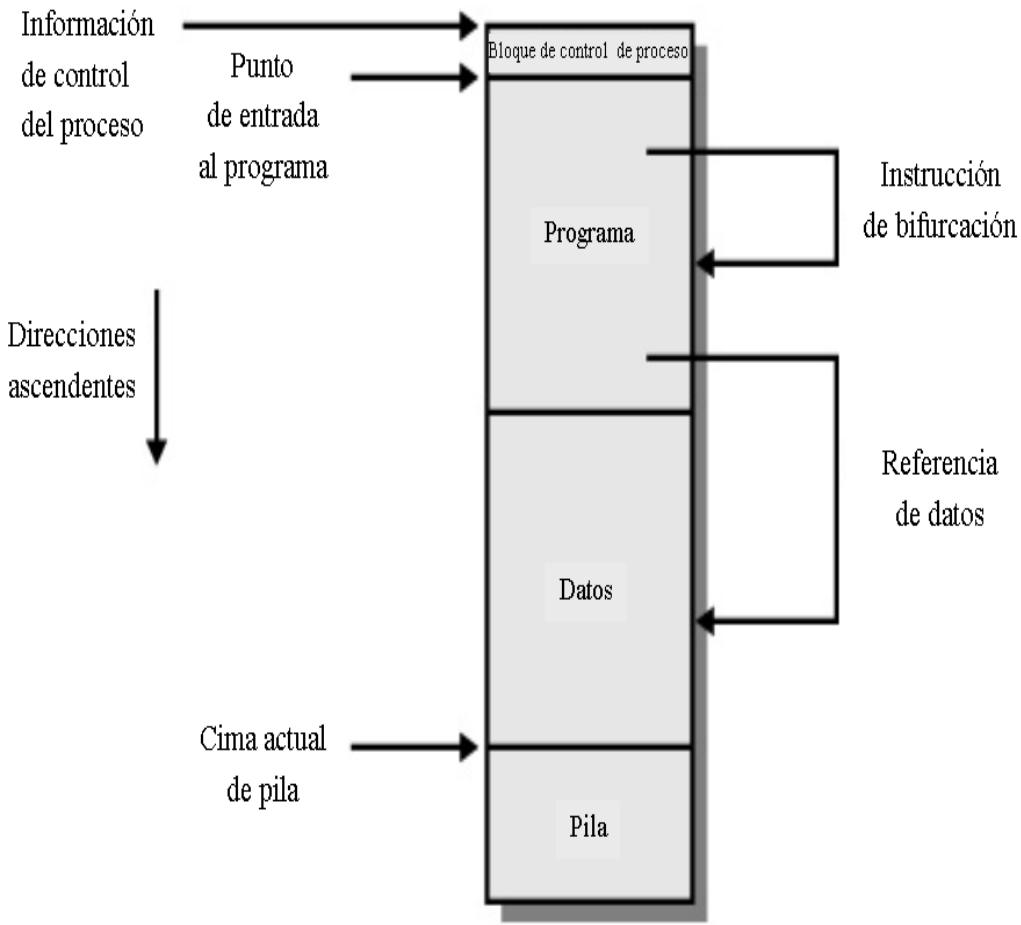
  Los procesos comparten la memoria principal

  El S.O. descarga procesos de memoria a disco. Intercambio.

  Tras la recarga, el proceso en memoria puede no situarse en la misma zona de memoria.

  Asuntos técnicos relativos al direccionamiento.

  - El S.O. debe conocer la ubicación del PCB, el comienzo de programa y la pila. Esto es fácil.
  - El procesador debe ser capaz de determinar la dirección real de las referencias a memoria de las instrucciones de salto y bifurcación asi como las referencias a datos del programa.
  - El Hardware y el S.O. deben ser capaces de traducir las referencias a memoria en direcciones fisicas reales tras la reubicación.



Gestión de Memoria

Protección

Los procesos deben protegerse contra accesos intencionados o no y no permitidos de otros procesos a su código y datos.

La reubicación complica el problema, durante la compilación es imposible determinar si la referencia se hará sobre el espacio de direcciones del proceso o no. Además está el cálculo dinámico de direcciones (Arrays)

Solo se puede comprobar cada referencia a memoria en tiempo de ejecución.

Los mecanismos que soportan la reubicación, también garantizan la protección.

- Un proceso de usuario no puede acceder a código ni datos del Sistema Operativo.
- El código de un proceso no bifurcará a una instrucción de otro proceso.
- Sin consentimiento o acuerdo, no se puede acceder al área de datos de otro proceso. Si esto ocurre el procesador (hardware) debe ser capaz de abortar la ejecución de dichas instrucciones generando un excepción.

Compartición

Es un mecanismo por el que se permite el acceso de varios procesos a la misma zona de memoria.

Se debe compartir código. Es mejor que cada proceso acceda al mismo código de proceso en lugar de hacerlo a una copia individual.

Los mecanismos de reubicación también permitirán la compartición

Gestión de Memoria

- Organización lógica

  La memoria principal se organiza como un espacio lineal de direcciones de byte o palabras.

  La memoria física se organiza de forma similar.

  Los programas se organizan de forma lógica en módulos.

  Si el S.O. y el hardware fueran capaces de gestionar el código y datos de los programas en forma de módulos de algún tipo se conseguirían las siguientes ventajas:

  - Los módulos se pueden escribir y compilar independientemente, ya que el sistema resolverá las referencias entre módulos.
  - Con escaso coste adicional se pueden establecer distintos grados de protección a los módulos.
  - Se pueden introducir mecanismos de compartición de módulos.

  La herramienta que satisface de forma más fácil estas técnicas es la segmentación.

Gestión de Memoria

- Organización física

La memoria se organiza al menos en dos niveles, principapal y secundaria.

- Memoria Principal

  Es volátil, cara, pero de acceso extremadamente rápido

- Memoria Secundaria

  Es más lenta, pero más barata y ofrece almacenamiento permanente.

Podría recaer en el programador la tarea de gestionar este flujo entre memoria principal y secundaria, pero es complicado por:

- En un entorno monoprogramado, si la memoria principal para un programa y sus datos es insuficiente, este debe ser programado con “overlays” o superposiciones.
  Distintos módulos pueden ocupar la misma zona de memoria, y es el programador el que debe saber que módulo esta en memoria, dónde, cómo y cuando cambiarlo.
- En un entorno multiprogramado, no conoce en tiempo de codificación, cuanto espacio tendrá disponible en tiempo de ejecución ni la ubicación de dicho espació.

Por tanto el S.O. debe ser el responsable del traspaso de información entre la memoria principal y secundaria.

Gestión de Memoria

Sin Memoria Virtual

Gestión de Memoria: Sin Memoria Virtual

Gestión de Memoria

La tarea fundamental del gestor de memoria es cargar programas en memoria para su ejecución.

Actualmente esto se realiza mediante un sofisticado sistema de Memoria Virtual.

La memoria virtual se basa en dos técnicas avanzadas: Segmentación y/o Paginación.

Antes de ver estas técnicas estudiaremos mecanismos más simples
que no requieran de Memoria Virtual:

Particionamiento:

- Particiones Estáticas
- Particiones Dinámicas

Proceso de Reubicación

Paginación Simple

Segmentación Simple.

Gestión de Memoria

Sin Memoria Virtual
PARTICIONAMIENTO

Gestión de Memoria: Particiones

Particiones Estáticas

El S.O. ocupa una zona de memoria, el resto está disponible para los procesos.

El esquema más sencillo es dividir esta zona en regiones de tamaño fijo.

Se plantean dos alternativas:

- Particiones de igual tamaño.
- Particiones de distinto tamaño.

- Igual Tamaño

  Cualquier proceso cuyo tamaño sea menor o igual que el tamaño de la partición puede cargarse en cualquier partición libre.

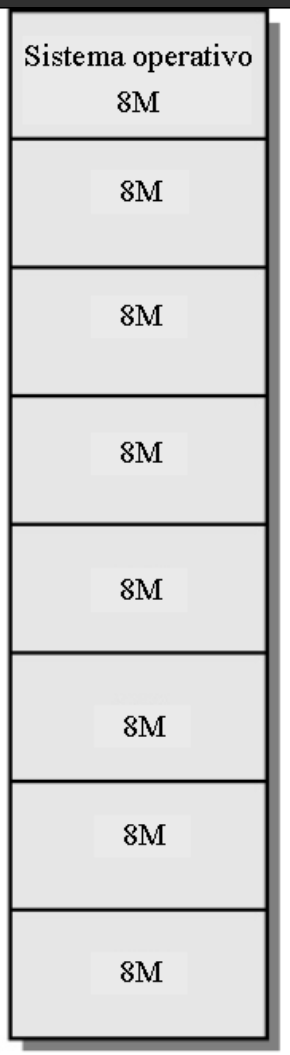
  Si todas las particiones están ocupadas, el sistema operativo puede sacar un proceso de una partición.

  Un programa puede que no se ajuste a una partición. El programador debe diseñar el programa mediante superposiciones que se cargarán en la partición asignada al programa.

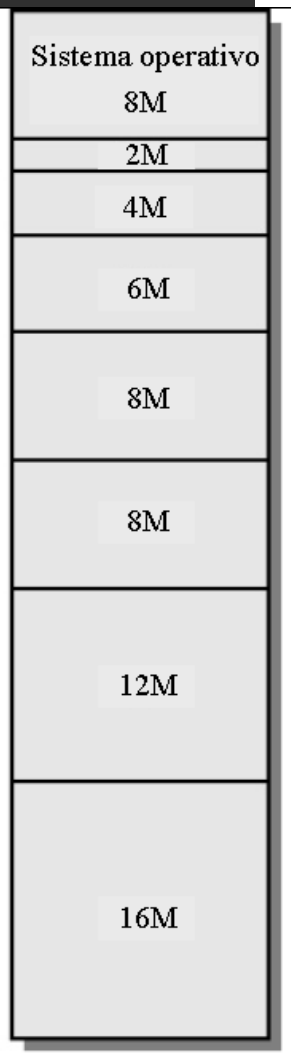
  El uso de la memoria principal es ineficiente. Cualquier programa, sin importar lo pequeño que sea, ocupará una partición completa. Este fenómeno se denomina fragmentación interna

- Distinto Tamaño

  Se pueden solventar en cierta medida dichos problemas, con particiones de distinto tamaño, pero no eliminarlos.



(a) Particiones de igual tamaño



(a) Particiones de distinto tamaño

Gestión de Memoria: Particiones

Particiones Estáticas. Algoritmo de Ubicación

- Igual Tamaño

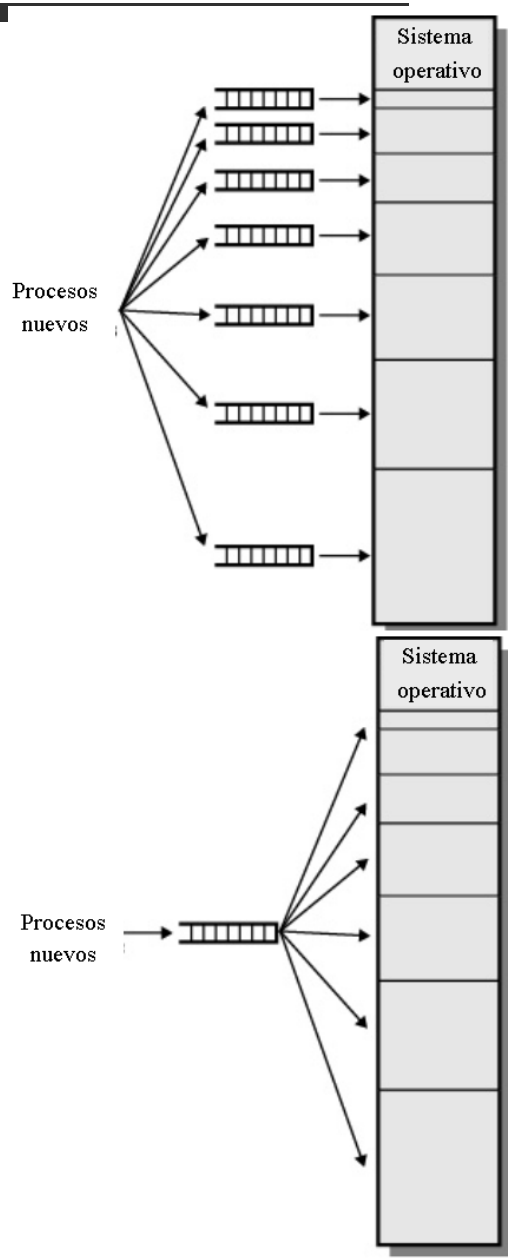
  Puesto que todas las particiones son de igual tamaño no importa cual seleccionar. Si todas están ocupadas el planificador seleccionará una para descargar.

- Distinto Tamaño

  Pueden asignar cada proceso a la partición más pequeña en la que quepa. Para ello hace falta una cola para cada partición.

  Los procesos están asignados de forma que se minimiza la memoria desaprovechada dentro de cada partición.

  Si se usa una sola cola para todas las particiones, si la partición mas pequeña en la que cabe el proceso esta ocupada y existe otra de mayor tamaño libre, esta se selecciona. Si no, se opta por el intercambio, pudiendo dar preferencia al intercambio de la más pequeña, o seleccionar otro criterio de intercambio.



Gestión de Memoria: Particiones

- Particiones Estáticas. Ventajas & Desventajas

Ventajas

Ambos modelos son bastante simples de implementar, y exigen software de S.O. y sobrecarga mínimos.

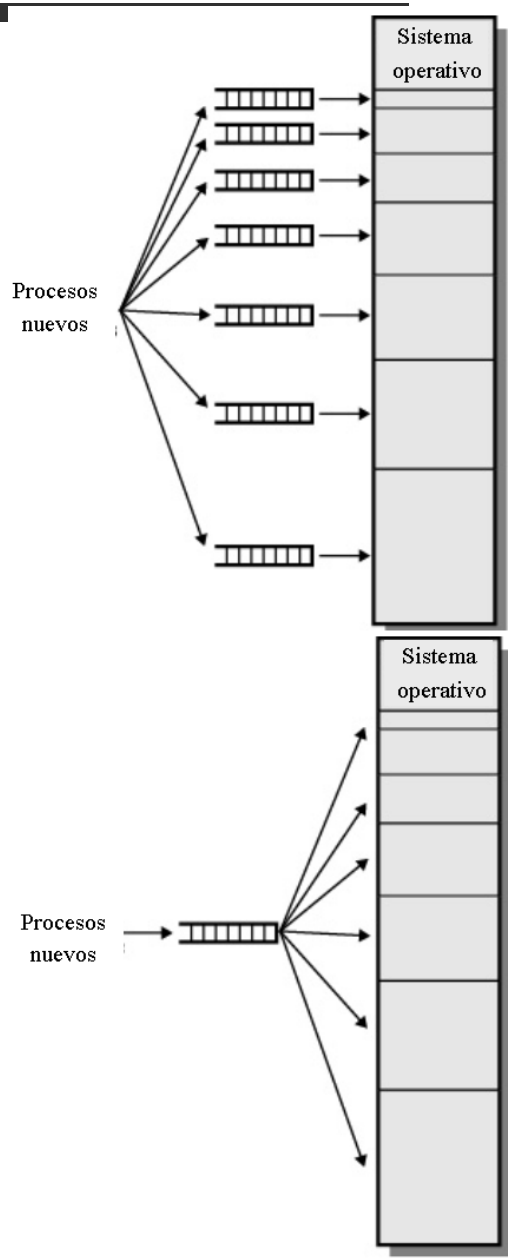
Desventajas

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

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

En la mayoría de los casos no se conoce de antemano la necesidad de memoria de los procesos que correrán en el sistema, por lo que definirlos en el momento de la generación del sistema parece inadecuado.

Hoy en día no se usa. Ejemplo OS/MFT de IBM 360/40 y 360/50 1966



Gestión de Memoria: Particiones

- Particiones Dinámicas.

  Ejemplo OS/MVT IBM 360 y 370 1970

  Las particiones son variables en número y longitud.

  Al proceso se le asigna exactamente tanta memoria como necesite.

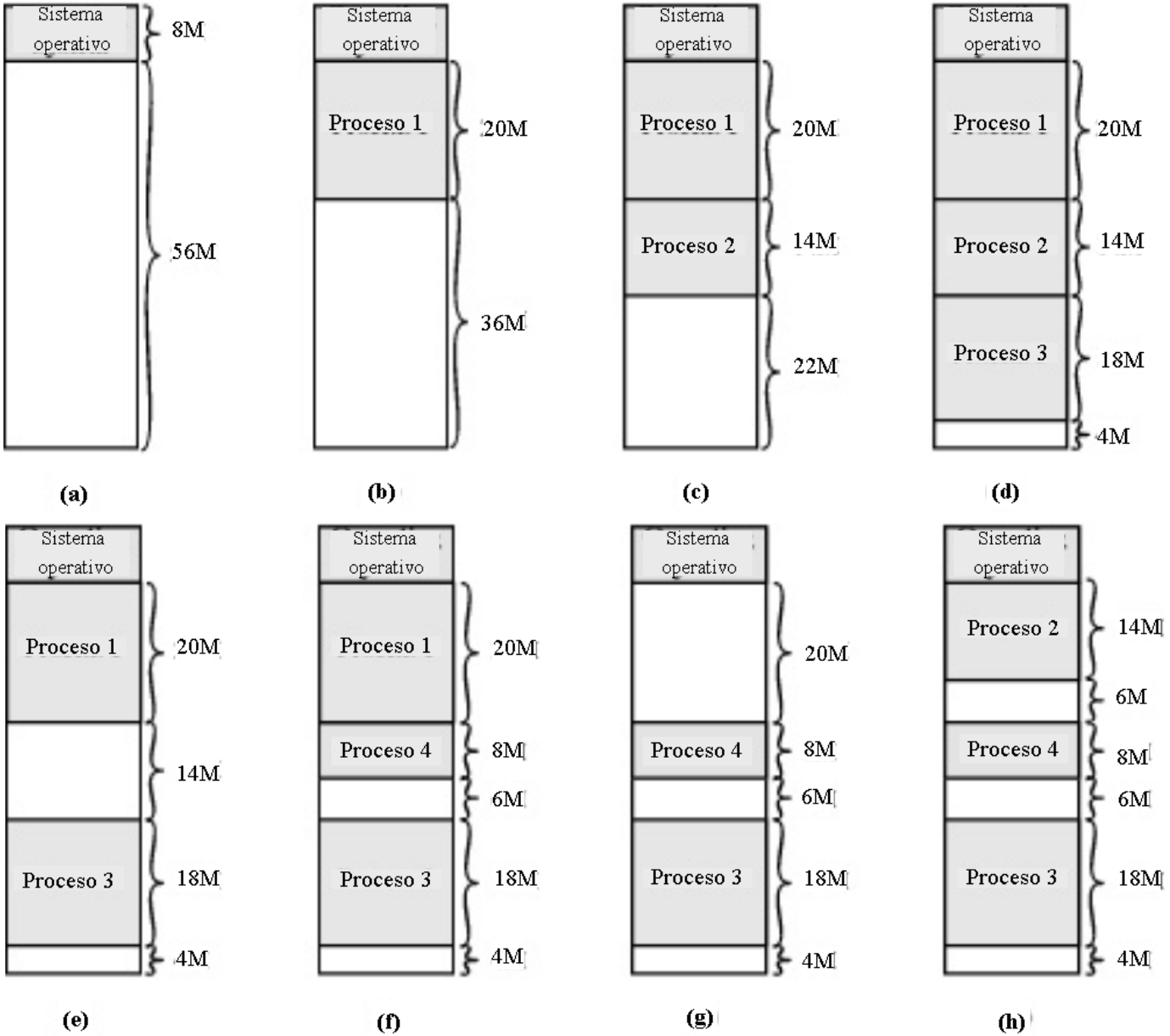
  Finalmente, hay varios huecos en la memoria. Este fenómeno se denomina **fragmentación externa**.

  Se debe usar la **compactación** para desplazar los procesos que estén contiguos, de forma que toda la memoria libre quede junta en un bloque.

  La compactación consume tiempo, por lo que se desperdicia tiempo de procesador, además debe tener capacidad de reubicación dinámica, es decir, mover procesos de una región a otra sin invalidar referencias.

Gestión de Memoria: Particiones

Particiones Dinámicas.



Gestión de Memoria: Particiones

Particiones Dinámicas. Algoritmos de Ubicación.

Como la compactación consume tiempo es deseable un buen algoritmo de asignación de memoria. Si hay más de un bloque libre donde quepa el proceso el S.O. debe decidir cual asignar.

Algoritmos: Se limitan a elegir entre bloques de memoria libre mayores o iguales que el proceso a cargar.

- Best Fit

  Elige el bloque de tamaño más próximo al solicitado

- First Fit

  Elige el primer bloque desde el comienzo de la memoria con espacio suficiente

- Next Fit

  Elige el primer bloque desde la posición de ultima asignación con espacio suficiente.

Gestión de Memoria: Particiones

- Particiones Dinámicas. Algoritmos de Ubicación.

  Cuál es mejor depende de la secuencia exacta de intercambios y tamaños.

  - First Fit

    Es más rápido y el más sencillo de implementar.

    Puede tener varios procesos cargados en el extremo inicial de la memoria que es necesario recorrer cuando se intente encontrar un bloque libre.

  - Next Fit

    Lleva frecuentemente a la asignación de un bloque de memoria de la última ubicación, donde se encuentra el bloque más grande que se divide en fragmentos pequeños.

    Hará falta la compactación para obtener un bloque de memoria grande al final del espacio de memoria.

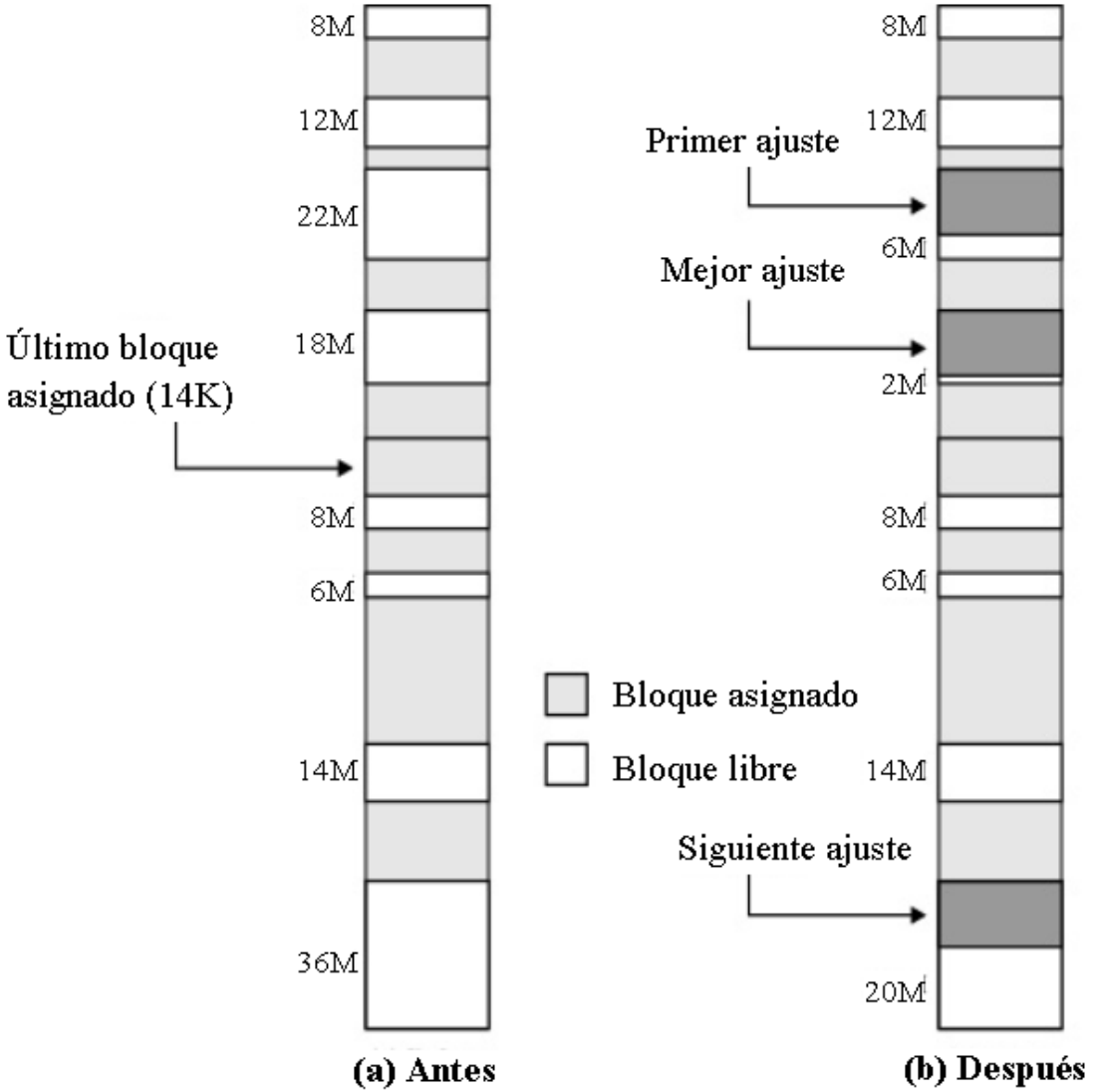
  - Best Fit

    Proporciona en general los peores resultados.

    Puesto que este algoritmo busca el hueco más pequeño para el proceso, garantiza que el fragmento que se deja es lo más pequeño posible y, por lo tanto, se debe compactar más frecuentemente.

Gestión de Memoria: Particiones

- Particiones Dinámicas. Algoritmos de Ubicación.



Diferencia entre los algoritmos de ubicación para intentar asignar un bloque de 16 Mb tras la situación a)

Gestión de Memoria: Particiones

- Particiones Dinámicas: Sistema de Colegas.

  Los bloques de memoria disponibles son de tamaño 2^K para valores de K tales que $L \leq K \leq U$ y donde

  - 2^L = tamaño de bloque más pequeño asignable
  - 2^U = tamaño de bloque más grande asignable

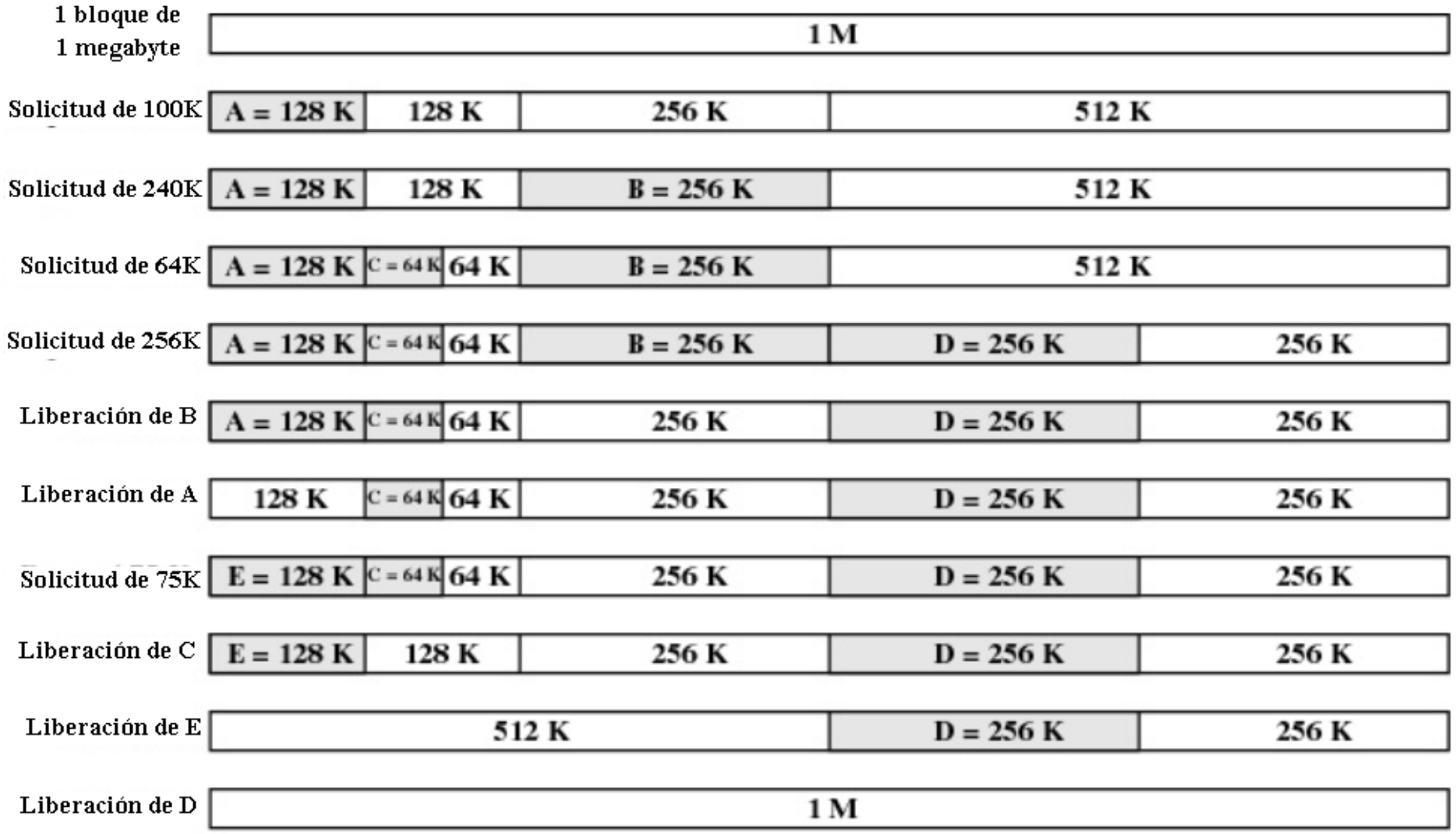
  Inicialmente el espacio entero disponible para la asignación se trata como un solo bloque de tamaño 2^U.

  Si se hace una solicitud de tamaño *s* tal que $2^{U-1} < s <= 2^U$, entonces el bloque entero se asigna:

  - En otro caso, el bloque se divide en dos colegas de igual tamaño.
  - Este proceso continúa hasta que el bloque más pequeño sea mayor o igual que *s* asignándose entonces a éste.

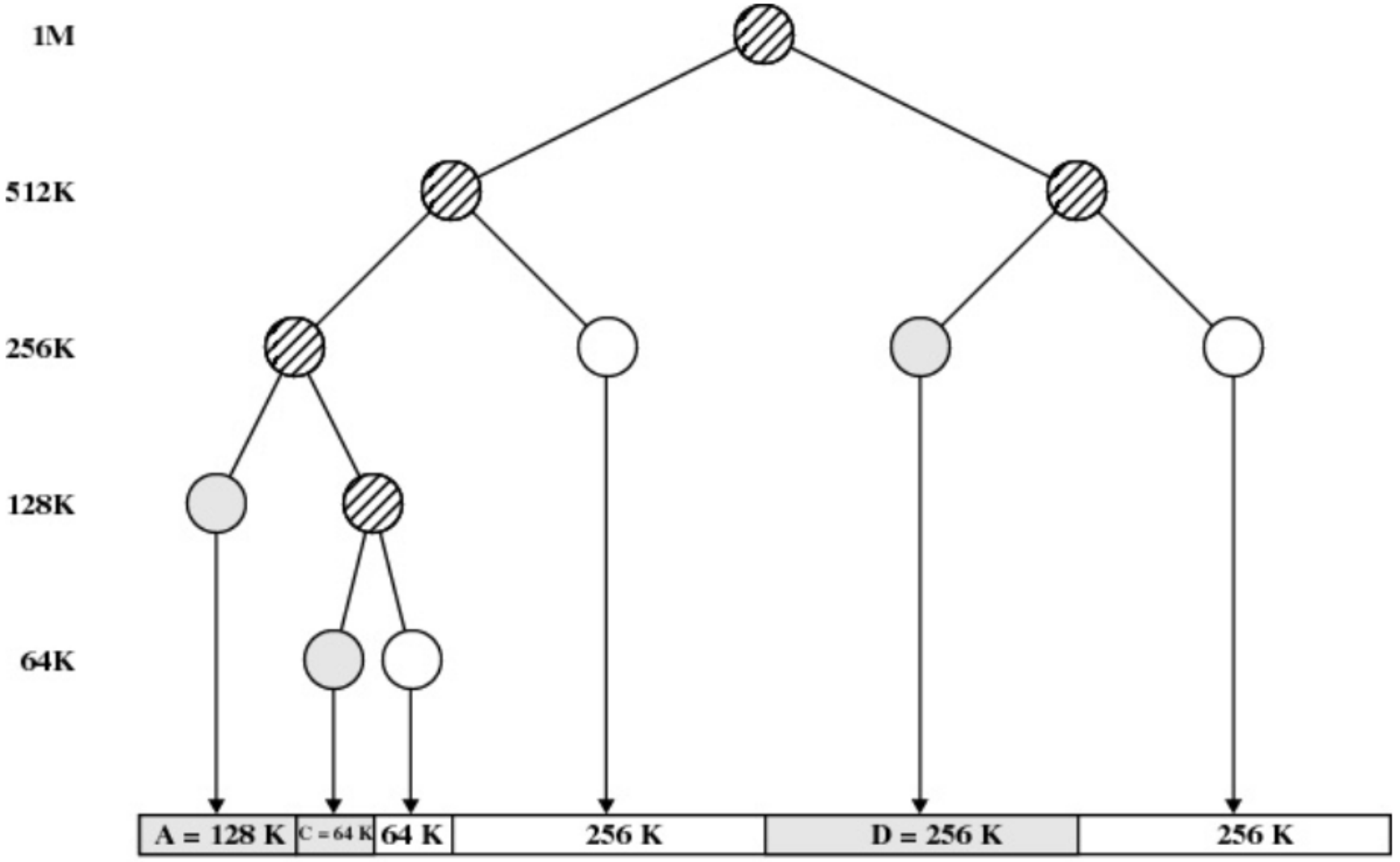
Gestión de Memoria: Particiones

❑ Particiones Dinámias: Sistema de Colegas. Ejemplo.



Gestión de Memoria: Particiones

- Particiones Dinámicas: Sistema de Colegas. Ejemplo.



Gestión de Memoria

(Proceso de Reubicación)

Gestión de Memoria: Proceso de Reubicación

Cuando el proceso se carga en la memoria, se determina la ubicación real (absoluta) de la memoria.

Cuando el sistema ubica siempre en la misma partición al proceso tras el intercambio:

Se resuelven las referencias relativas por absolutas en el momento de la carga basándose en la dirección base del proceso cargado.

En otro caso un proceso puede ocupar diferentes particiones a lo largo de su vida:

Por tanto las ubicaciones de las instrucciones y datos referenciadas no son fijas.

El intercambio (en estos casos) y la compactación provocan la reubicación.

Se diferencia entre varios tipos de direcciones

- Dirección lógica

  Referencia a una posición de memoria independientemente de la asignación actual de datos a la memoria.

  Se deben traducir a direcciones físicas.

- Dirección relativa

  Un caso especial de dirección lógica, que se expresa como una posición relativa a un punto conocido, el comienzo del programa, p.ej.

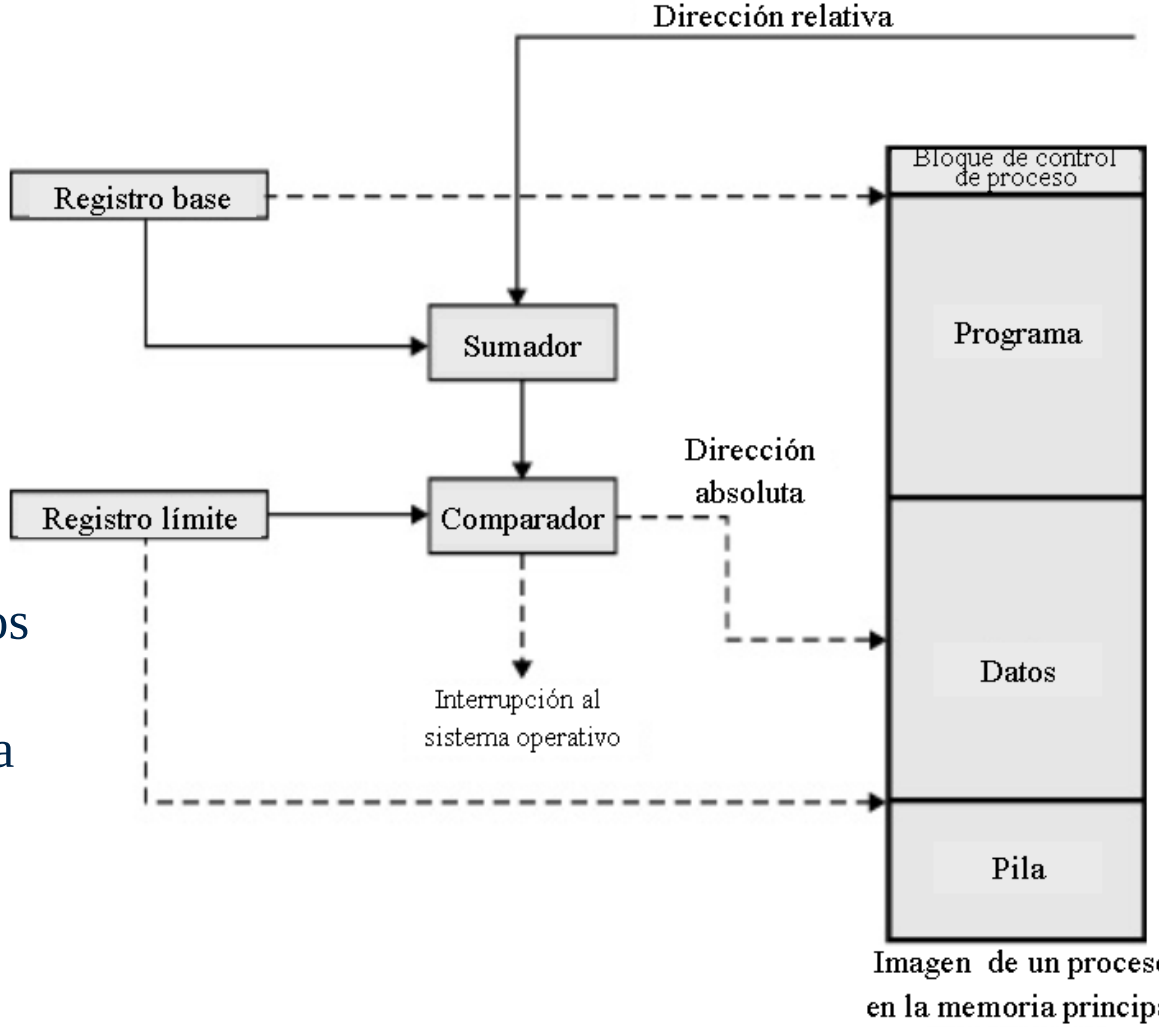
- Dirección física.

  O dirección absoluta es una posición real de la memoria principal.

Gestión de Memoria: Proceso de Reubicación

Los programas que utilizan direcciones relativas se cargan mediante cargadores dinámicos lo que ocasiona que las direcciones sean relativas al origen del programa. Por tanto el hardware necesita de un mecanismo de traducción de las direcciones relativas en direcciones físicas.

Soporte hardware para la reubicación



Este esquema permite a los programas cargarse y descargarse de la memoria proporcionando adicionalmente una medida de seguridad

Gestión de Memoria: Proceso de Reubicación

Registros utilizados durante la ejecución

- Registro base:

  Se carga con la dirección de inicio del proceso en la memoria principal.

- Registro límite:

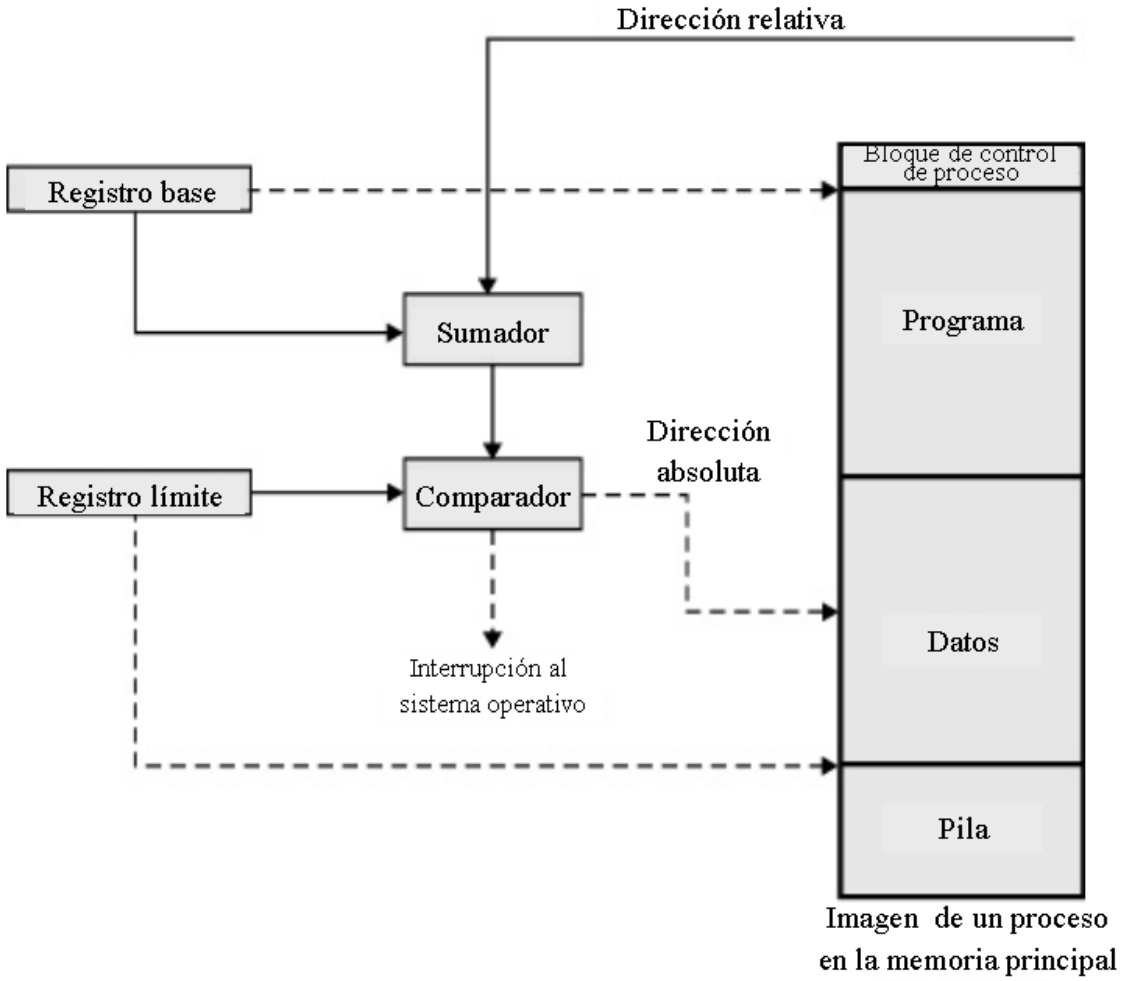
  Indica la posición final del programa.

Estos valores deben asignarse cuando se carga el proceso.

Se añade el valor del registro base a la dirección relativa para obtener una dirección absoluta.

La dirección obtenida se compara con el valor del registro límite.

Si la dirección no está dentro de los límites, se generará una interrupción en el sistema operativo.



Gestión de Memoria

Sin Memoria Virtual
PAGINACIÓN SIMPLE

Gestión de Memoria: Paginación Simple

Con el uso de particiones tenemos un uso ineficiente de la memoria

- Particiones Fijas: Generan Fragmentación Interna
- Particiones Variables: Generan Fragmentación Externa

Con la paginación, la memoria principal se encuentra dividida en trozos iguales de tamaño fijo y cada proceso en pequeños trozos del mismo tamaño fijo.

- Los **trozos del proceso** se denominan **páginas**
- Los **trozos de memoria** se denominan **marcos**.

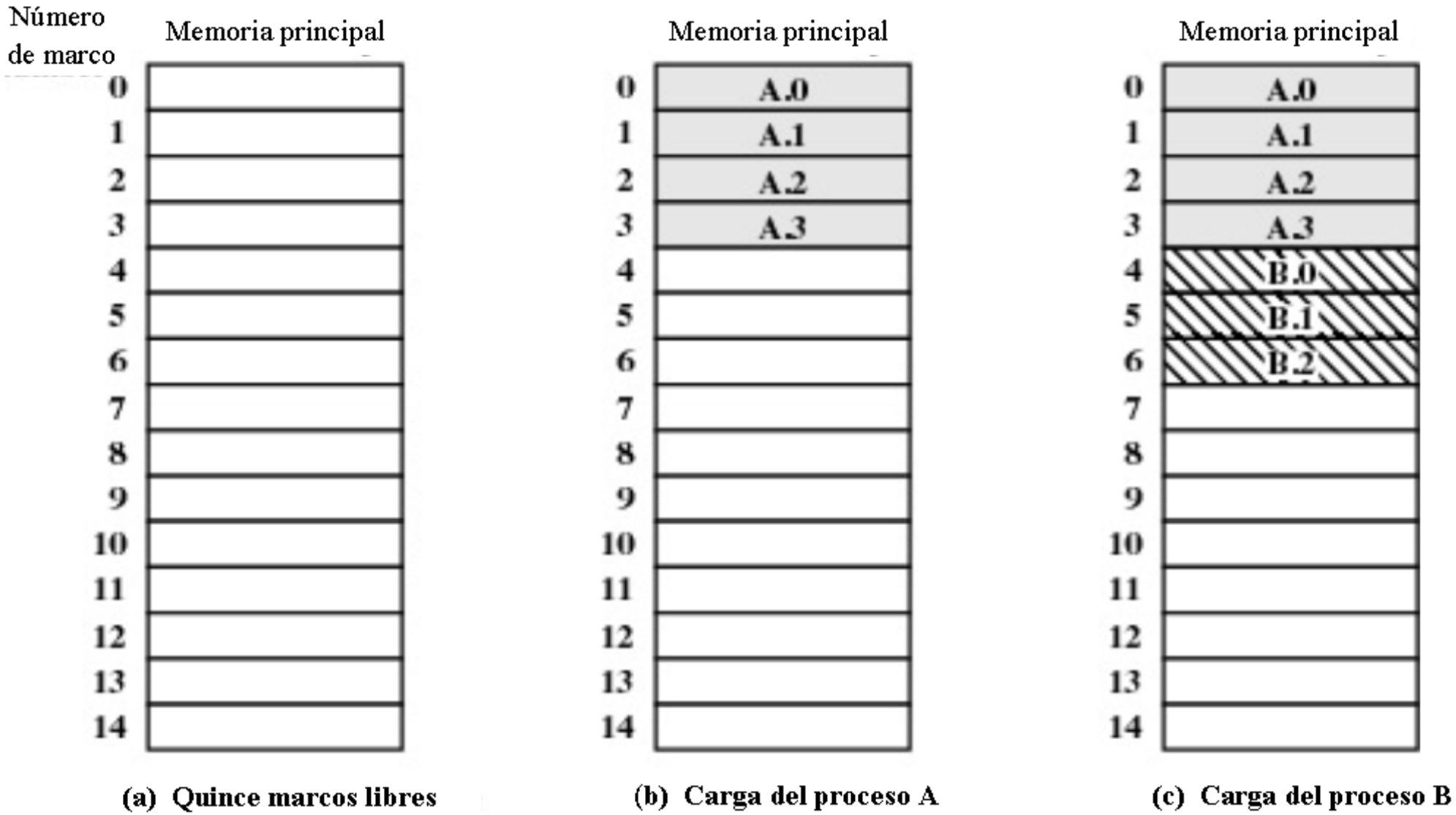
Se asignan marcos de página libres para las páginas del proceso.

Solo se produce fragmentación en el último marco de pagina de un proceso. Además **no hay fragmentación externa**.

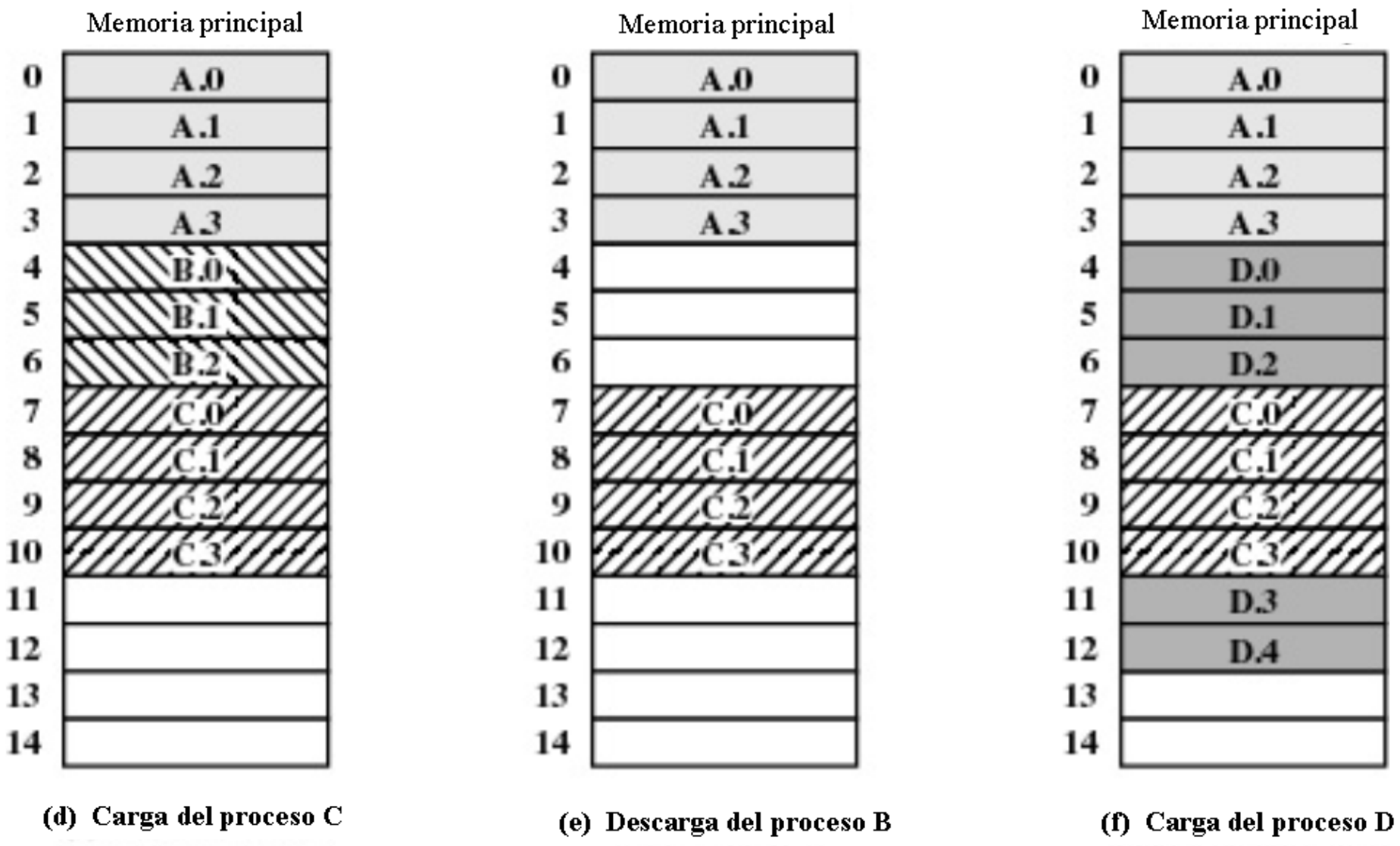
El sistema operativo mantiene una **tabla de páginas** para cada proceso:

- La tabla **contiene la posición del marco** de cada página del proceso.
- La **dirección lógica** de la memoria consta de un **número de página y de un desplazamiento** dentro de la página.

Gestión de Memoria: Paginación Simple



Gestión de Memoria: Paginación Simple



Gestión de Memoria: Paginación Simple

Ya no es suficiente con un único registro base. En su lugar el S.O. debe mantener una **Tablas de Páginas** para cada proceso

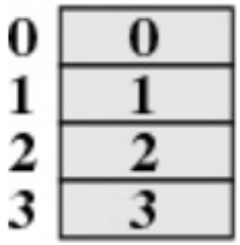


Tabla de páginas del proceso A

Tabla de páginas del proceso B

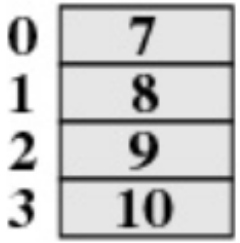


Tabla de páginas del proceso C

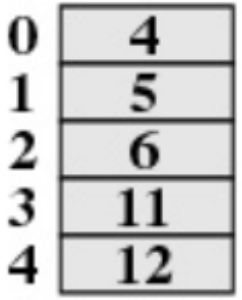
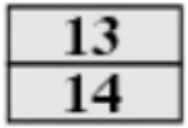
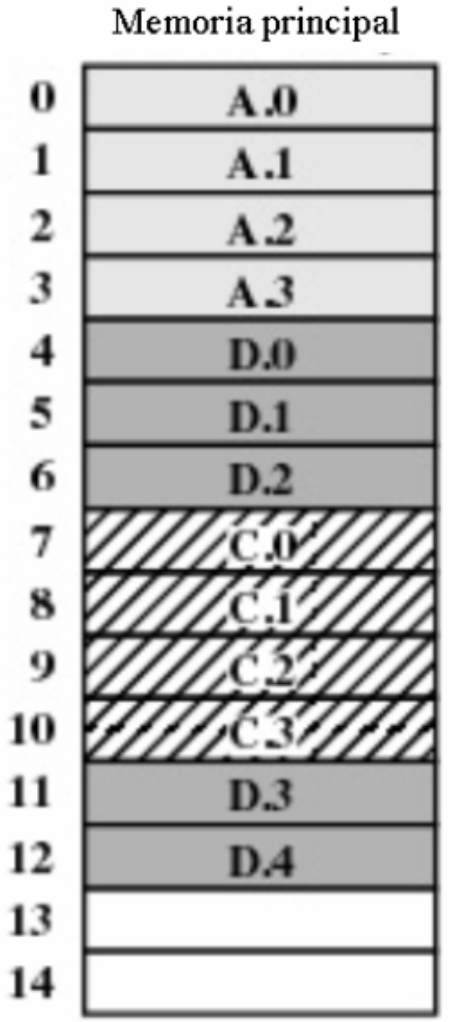


Tabla de páginas del proceso D



Lista de marcos libres

El hardware del procesador también traduce de direcciones lógicas a físicas, pero debe saber como acceder a la tabla de páginas para obtener el marco de página de la página, para añadir el desplazamiento y obtener la dirección física



(f) Carga del proceso D

Gestión de Memoria: Paginación Simple

¿Paginación Simple

vs.

Partición Estática igual tamaño?

Los marcos páginas en la paginación son más pequeños que las particiones estáticas de tamaño fijo.

En la paginación simple se puede ocupar más de un marco de página

En la paginación simple los marcos de página no tienen porque estar consecutivos.

Gestión de Memoria: Paginación Simple

Tamaño del marco de página y páginas

Debe ser potencia de 2, pues de esta forma la dirección lógica relativa al inicio del programa es la misma que la dirección lógica formada por número de página y desplazamiento.

Direcciones de 16 bits: 2^4

Tamaño de página 1Kb = 1024 bytes

La memoria se direcciona por bytes

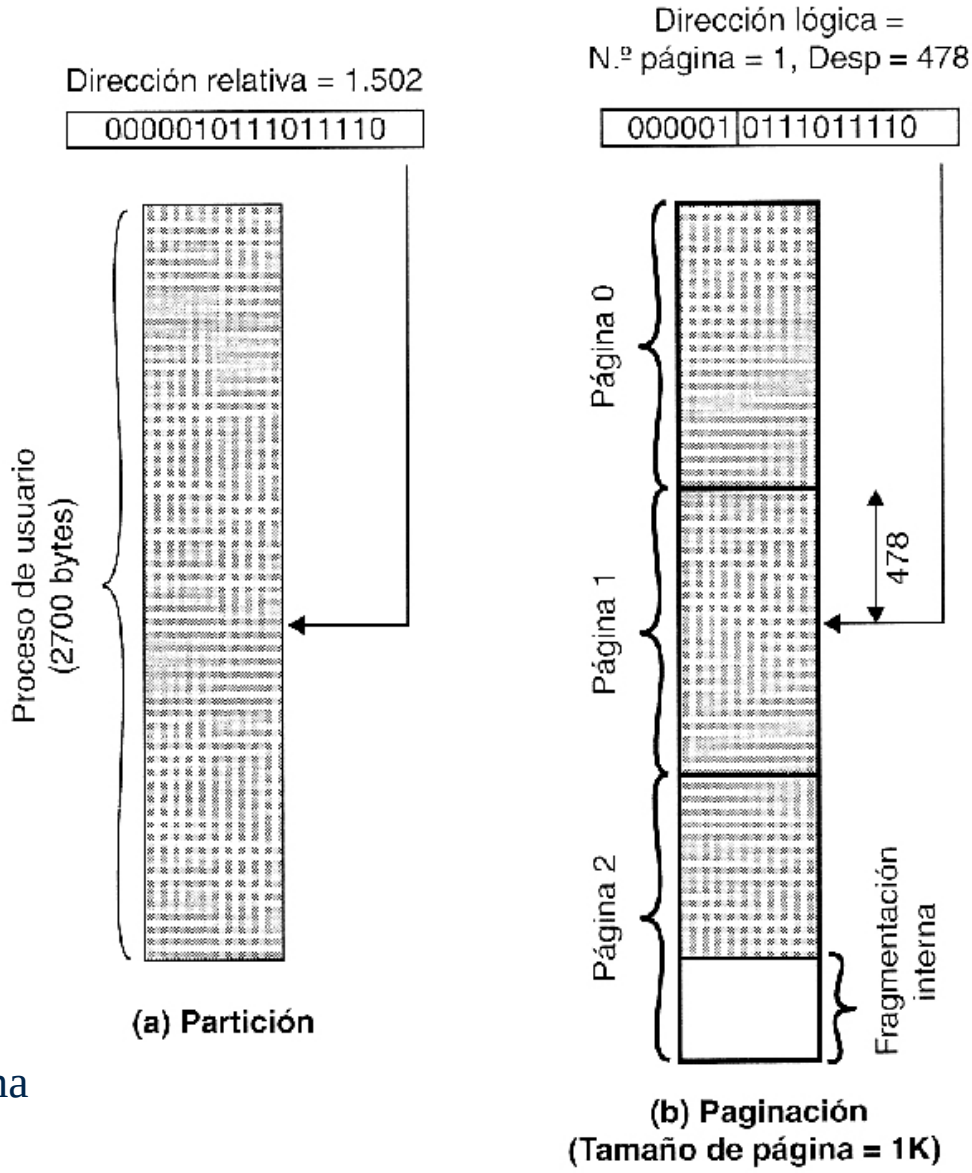
Para direccionar todos los bytes de una página (1024) necesito 10 bits.

Sobran 6 btis para el número de página.

Con 6 bits puedo direccionar 2^6 = 64 páginas de 1Kb.

Supongamos un salto a la dirección 1502 (relativa al inicio), en binario es la 0000010111011110

La dirección lógica pagina + desplazamiento es la misma que la relativa al inicio



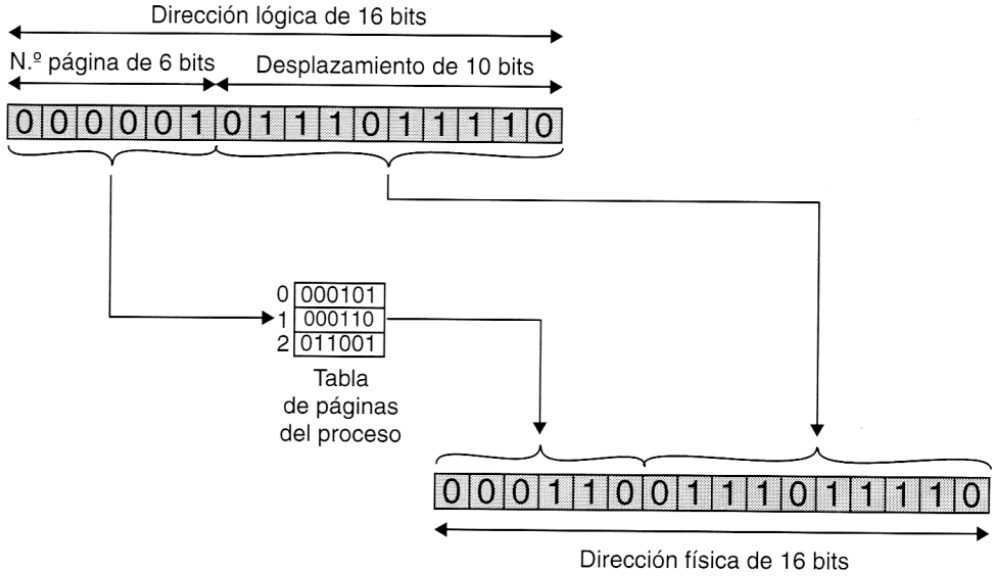
Gestión de Memoria: Paginación Simple

- Consecuencias de usar 2^N como tamaño del marco de página:

  El esquema de dirección lógica es transparente al programador, al ensamblador y al montador.

  Es sencillo realizar una función hardware para traducir las direcciones dinámicas durante la ejecución. Dirección lógica *n-m*

  1. Extraer el número de página de los *n* bits más significativos de la dirección lógica
  2. Emplear el numero de página como indice en la tabla de páginas del proceso para encontrar el número de marco *k*
  3. La dirección del marco comienza en k x 2^m a la que se le suma el desplazamiento *m*. No es necesario calcular, solo concatenar número de marco y desplazamiento.



Gestión de Memoria

Sin Memoria Virtual
SEGMENTACIÓN SIMPLE

Gestión de Memoria: Paginación Simple

- El programa y sus datos se dividen en Segmentos
- No es necesario que todos los segmentos de todos los programas, ni de un programa en particular, tengan la misma longitud.
- Existe una longitud máxima de segmento definida por un registro del MMU (Memory Managment Unit)
- Un dirección lógica segmentada consta de dos partes, un número de segmento y un desplazamiento.
- Como consecuencia del empleo de segmentos de distinto tamaño, la segmentación resulta similar a la partición dinámica, pero pudiendo ocupar más de un segmento y además no necesariamente contiguos.
- Elimina la fragmentación interna, pero crea fragmentación externa como la partición dinámica.
- Si no se adopta superposición ni memoria virtual, todos los segmentos de un programa deben cargarse en memoria principal.
- No es transparente al programador ni al compilador. El compilador suele asignar segmentos distintos a los datos, código, PCB y pila. Además el programador puede dividir el código y datos a su vez en segmentos.

Gestión de Memoria: Paginación Simple

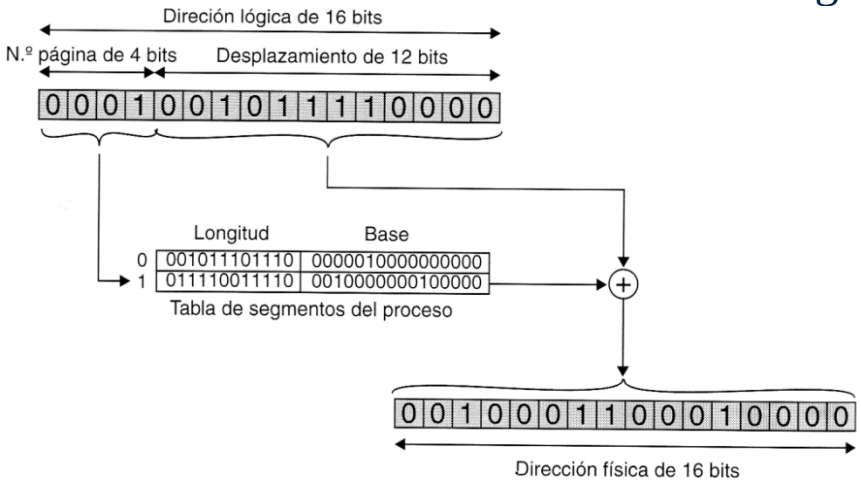
- No hay una correspondencia simple entre direcciones lógicas y físicas.
- Se usa una tabla de bloques libres de memoria y de la tabla de segmentos de cada proceso para localizar la dirección base del segmento y la longitud del segmento.

Considerar una dirección de $n+m$, con $n=4$ y $m=12$

Esto da un tamaño máximo de segmento de $2^{12}=4096$ bytes.

Para la traducción de direcciones:

1. Extraer el número de los **n** bits mas significativos
2. Emplear el número de segmento como índice en la tabla de segmento para localizar la dirección base del segmento
3. Comparar el desplazamiento m, con la longitud del segmento. Si el desplazamiento es mayor que **m** la dirección es invalidada.
4. La dirección buscada es la suma de la dirección física de comienzo del segmento y el desplazamiento.



Fin

Gestión de Memoria