Tema VI

Administración de memoria

Objetivos Docentes

- Los objetivos docentes de este capítulo son los siguientes:
  - Conocer y entender los conceptos de espacio del núcleo, espacio de usuario y área de intercambio.
  - Saber cómo se asigna la memoria principal en sistemas operativos monoprogramados.
  - Conocer y comprender el funcionamiento y las características de las principales técnicas que puede implementar un sistema operativo multiprogramado para la asignación contigua de memoria principal:
    - El particionamiento fijo y el particionamiento dinámico.
  - Conocer y comprender el funcionamiento y las características de las principales técnicas que puede implementar un sistema operativo multiprogramado para la asignación no contigua de memoria principal en sistemas que no soportan memoria virtual:
    - La paginación simple y la segmentación simple.

El subsistema de Administración de la Memoria

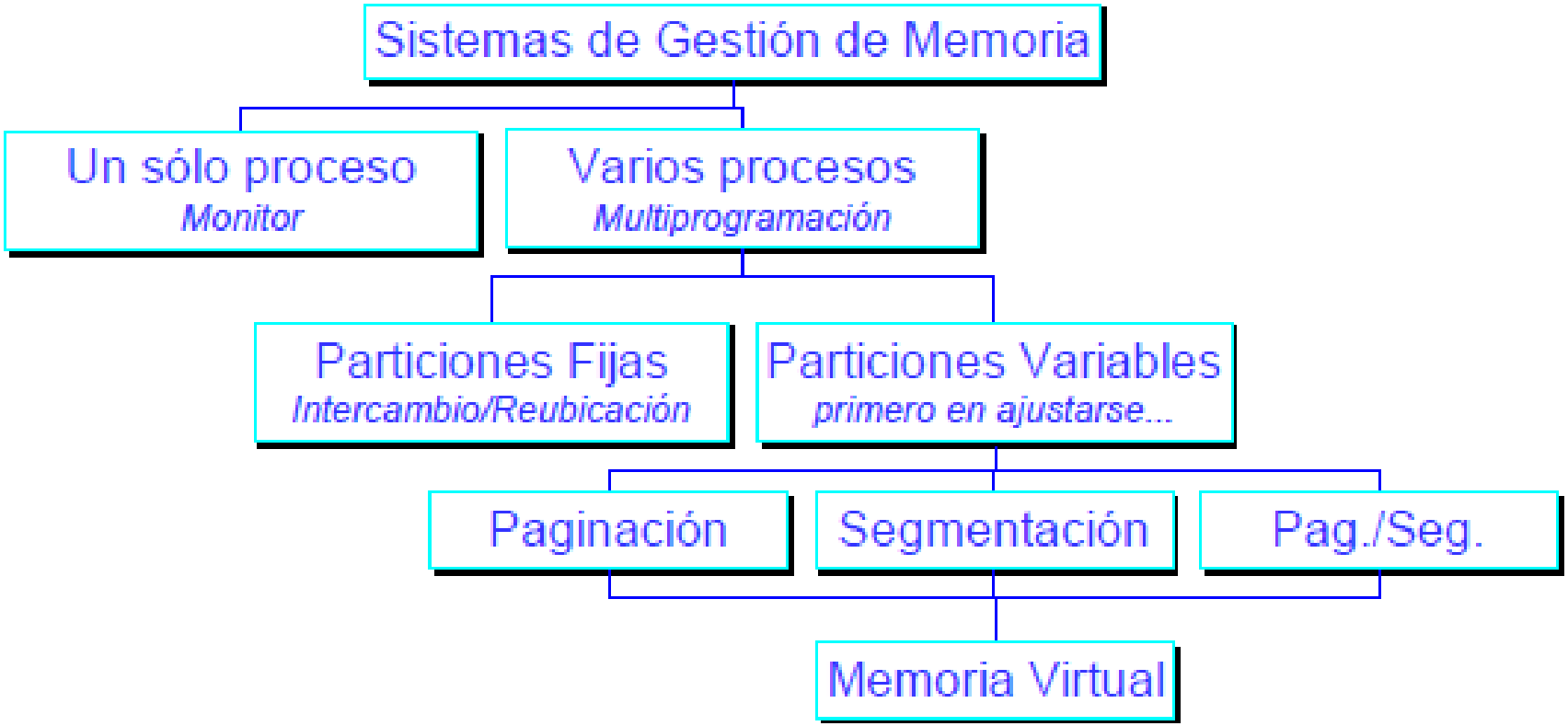
- El subsistema de administración de la memoria principal es el componente del sistema operativo encargado de administrar, en colaboración con el hardware, la memoria principal entre todos los procesos que se ejecutan en el computador
- Para que un proceso esté preparado para ejecución debe estar cargado en memoria principal
- **La misión del Gestor de Memoria** es la asignación de memoria principal a los procesos que la soliciten
  - *Técnicas de asignación contigua*
  - *Técnicas de asignación no contigua.*

Espacio del núcleo y espacio de usuario

- Espacio del núcleo
  - Espacio ocupado por el código, las estructuras de datos y la pila (o pilas) del núcleo del sistema
    - El acceso al espacio del núcleo solo se realizar en modo núcleo.
- Espacio de usuario
  - Espacio de memoria principal ocupado por la imagen de un proceso, es decir, su espacio de direcciones de memoria lógica
    - El acceso al espacio usuario se puede realizar en modo usuario o en modo núcleo.

Área de intercambio en memoria secundaria

- Área de intercambio
  - El sistema operativo reserva espacio en memoria secundaria, típicamente en un disco duro, para almacenar las copias de las imágenes de los procesos.
- *Intercambio* (*swapping*)
  - La operación de cargar la imagen de un proceso desde el área de intercambio a memoria principal o vice-versa.


Sistemas de Gestión de Memoria
Un sólo proceso
Monitor
Varios procesos
Multiprogramación
Particiones Fijas
Intercambio/Reubicación
Particiones Variables
primero en ajustarse...
Paginación
Segmentación
Pag./Seg.
Memoria Virtual

Asignación de memoria en sistemas monoprogramados

- Sin gestión de Memoria:
  - El usuario se encuentra con la máquina desnuda y tiene un control completo sobre el espacio total de la memoria (hasta los 60)
- La memoria esta dividida en dos secciones:
  - Una está asignada permanentemente a la parte del S.O. que debe estar residente en memoria o monitor
  - La otra se asigna a los llamados procesos transitorios, que son cargados y ejecutados uno cada vez, en respuesta a órdenes de usuario
- La memoria esta dividida en tres secciones:
  - La parte inferior (en RAM) está asignada al S.O.
  - La parte central al único programa del usuario
  - La parte superior (en ROM, (BIOS)) a los controladores de dispositivos

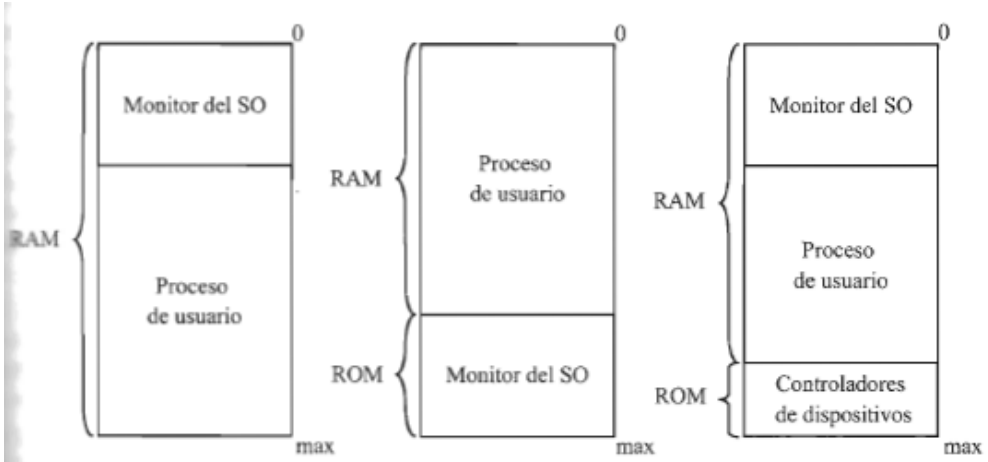


Figura 6.1 – Configuraciones de memoria principal más habituales en sistemas monoprogramados

Particionamiento Fijo

- Multiprogramación:
  - Permite el entrelazado y el solapamiento de la ejecución de más de un programa de forma que se mantenga del modo más ocupado posible todos los recursos
- Consiste en divisiones de memoria que se efectúan en algún momento antes de ejecutar los programas de usuario y permanecen fijas desde entonces
- El número de particiones distintas determina el grado de multiprogramación
  - Problema:
    - La fragmentación interna o memoria desaprovechada dentro de una partición
- Una vez definidas las particiones, el S.O. necesita llevar la cuenta de sus estados, libre o en uso para propósitos de asignación
- El estado y los atributos de las particiones se recogen en una estructura de datos llamada **Tabla de Descripción de Particiones (TDP).**
- Cada partición está descrita por su dirección inicial (base), su tamaño y su estado.
  - Los campos de la base y tamaño son fijos

Particiones de igual tamaño

- Limitación del tamaño máximo de los procesos que se pueden cargar en memoria principal
- Fragmentación interna

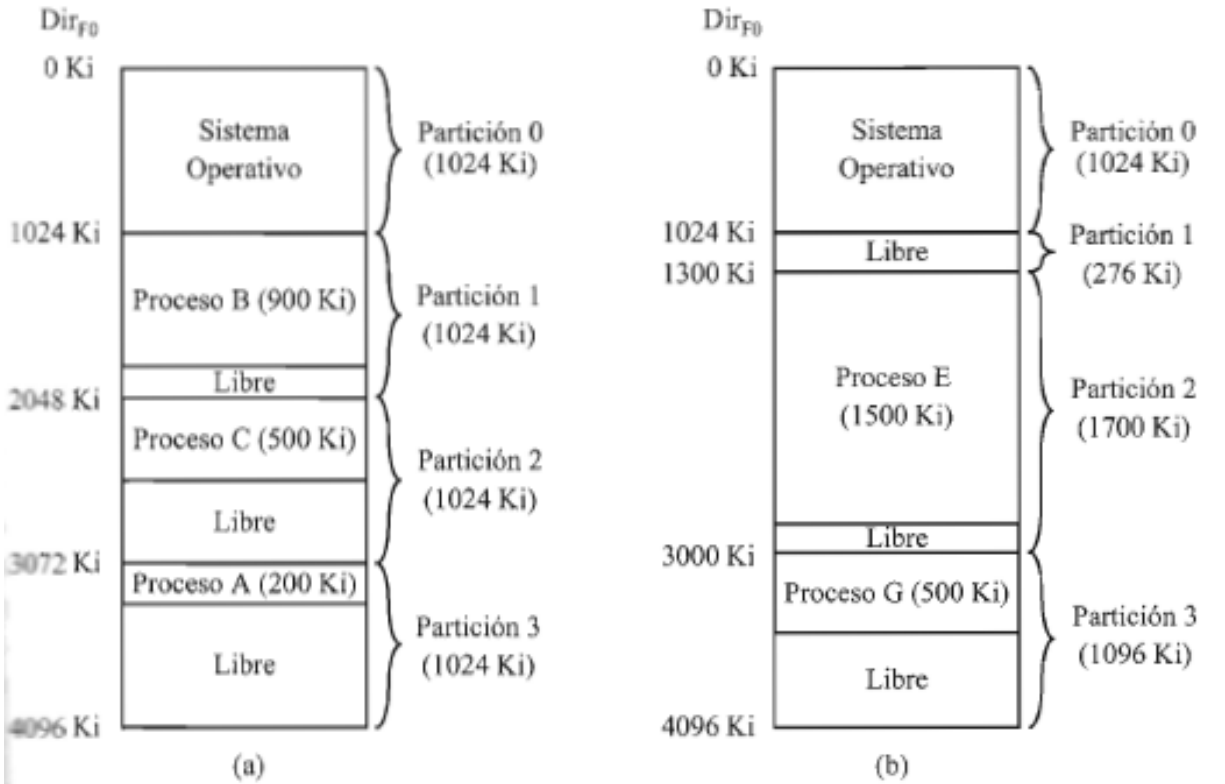


Figura 6.2 – Memoria principal con particionamiento fijo. (a) Particiones de igual tamaño. (b)Particiones de distinto tamaño

Particiones de distinto tamaño

- Otra organización posible es asignar una cola de tareas a cada partición en memoria y las tareas se incluyen en la cola de la partición de memoria correspondiente a sus exigencias de memoria
- Una única cola: estrategias de asignación de particiones:
  - Primer ajuste
  - Mejor ajuste

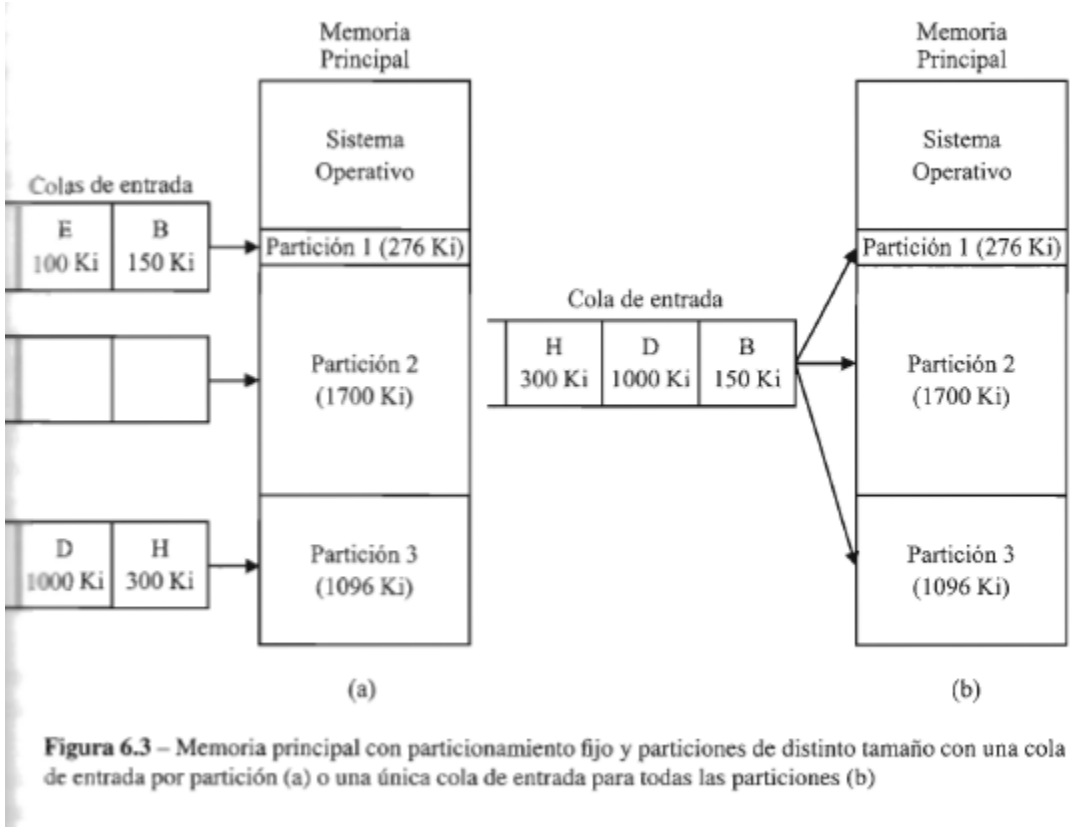


Figura 6.3 – Memoria principal con particionamiento fijo y particiones de distinto tamaño con una cola de entrada por partición (a) o una única cola de entrada para todas las particiones (b)

Traducción de direcciones y protección

- En sistemas que utilizan registros base para la reubicación
- Es habitual utilizar registros límite para la protección
  - Los valores base y límite de cada proceso se guardan en su BCP

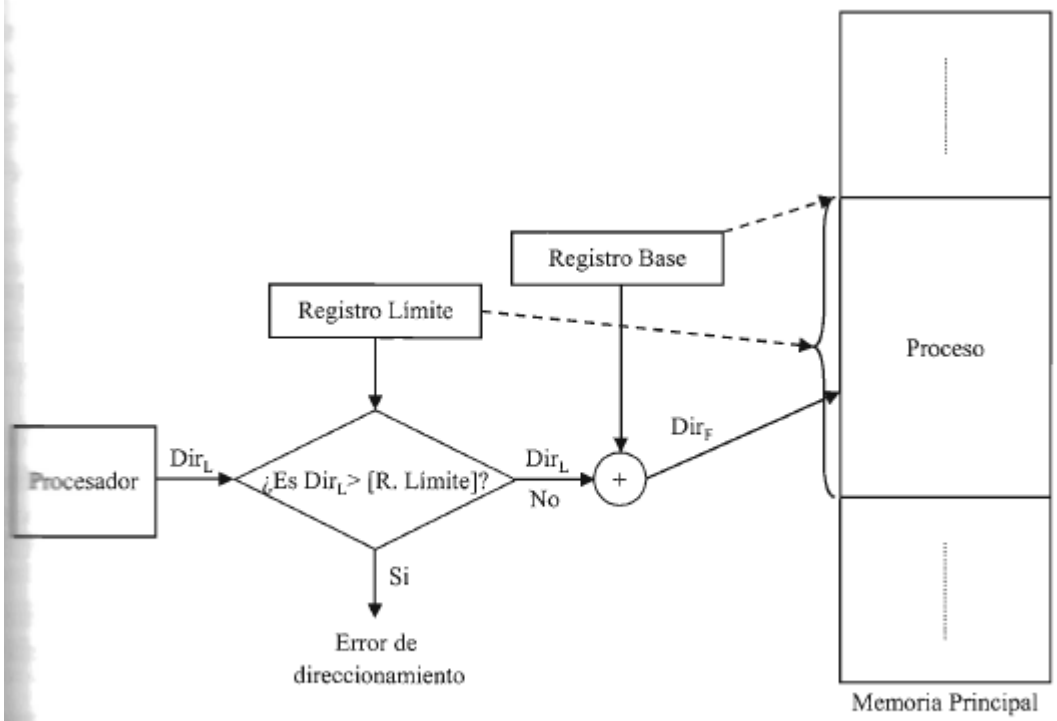


Figura 6.4 – Traducción de direcciones y protección de memoria con particionamiento fijo

Particionamiento Dinámico

- Cuando se pide que se cargue una imagen de un proceso en memoria, el módulo de gestión intenta crear una partición adecuada que asignar al proceso en cuestión, para ello es preciso localizar un área libre de memoria que sea igual o mayor que el proceso, si es así se fabrica la partición

Fragmentación

- Existen dos tipos de desaprovechamiento de la memoria:
- **Fragmentación interna**:
  - Consiste en aquella parte de la memoria que no se está usando porque es interna a una partición asignada a una tarea
- **Fragmentación externa**:
  - Ocurre cuando una partición disponible no se emplea porque es muy pequeña para cualquiera de las tareas que esperan
- La selección de los tamaños de las particiones es un compromiso entre estos dos casos de fragmentación
- Si la memoria resulta seriamente fragmentada, la única salida posible es reubicar algunas o todas las particiones en un extremo de la memoria y así combinar los huecos para formar una única área libre grande a este proceso se le llama **compactación**
- **Compactación de memoria**
  - Consiste en agrupar todos los huecos pequeños en un único hueco
  - Como los procesos afectados deben de ser suspendidos y copiados realmente de un área de memoria a otra, es importante decidir cuando debe realizarse la compactación

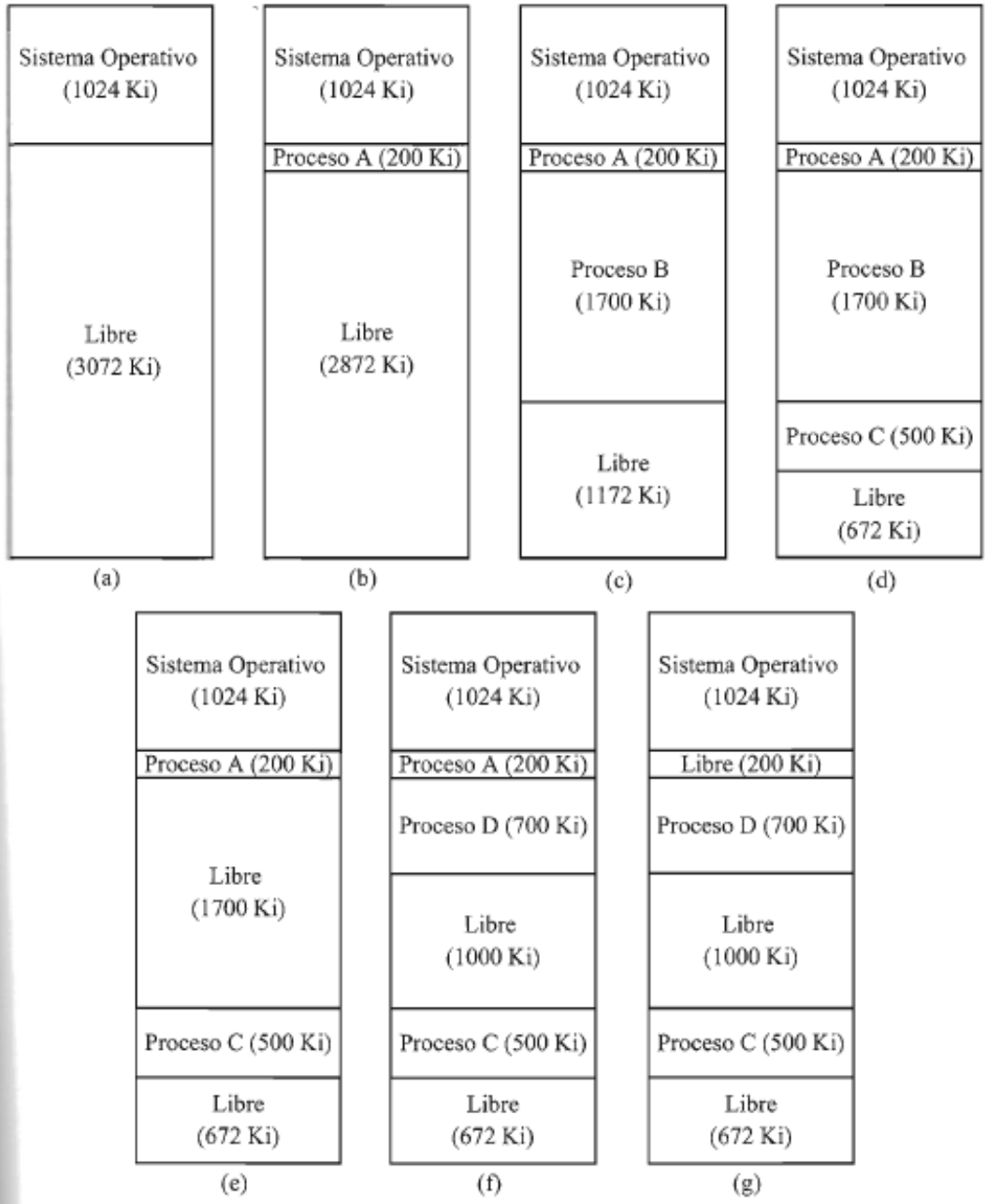


Figura 6.5 – Ejemplo de gestión de memoria con particionamiento dinámico

Control

- La gestión de la memoria con particiones libres plantea el problema de mantenimiento de un registro de particiones libres y ocupadas que sea eficiente, tanto en tiempo para la asignación como para el aprovechamiento de la memoria.
- Formas de mantener este registro:
  - Mapa de bits
  - Listas enlazadas para las particiones libres y asignadas

Mapa de Bit

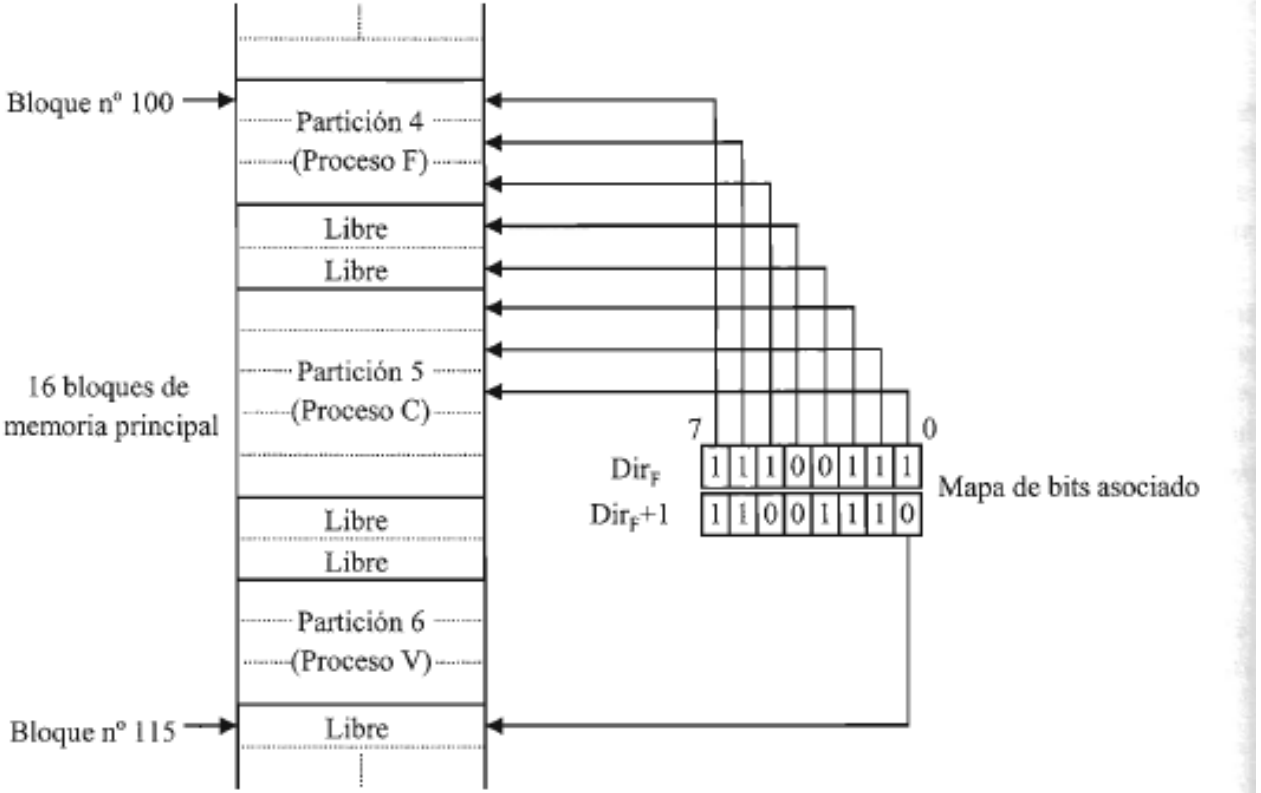


Figura 6.6 – Mapa de bits para establecer el estado libre u ocupado de los bloques de la memoria principal

Listas Enlazadas

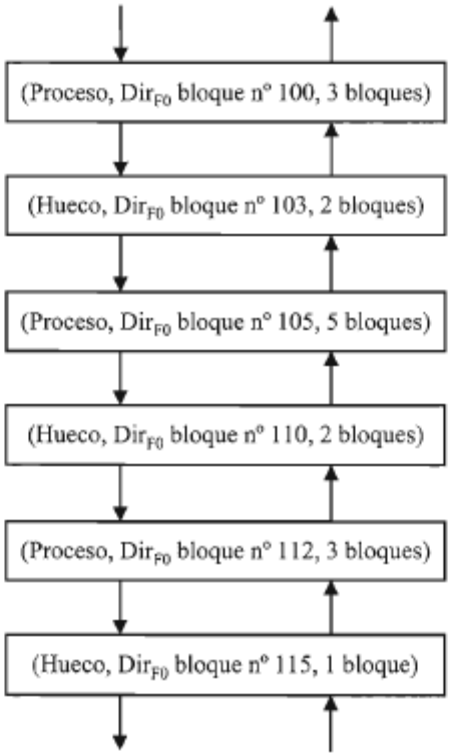


Figura 6.7 – Parte de la lista doblemente enlazada que registra la utilizació la Figura 6.6

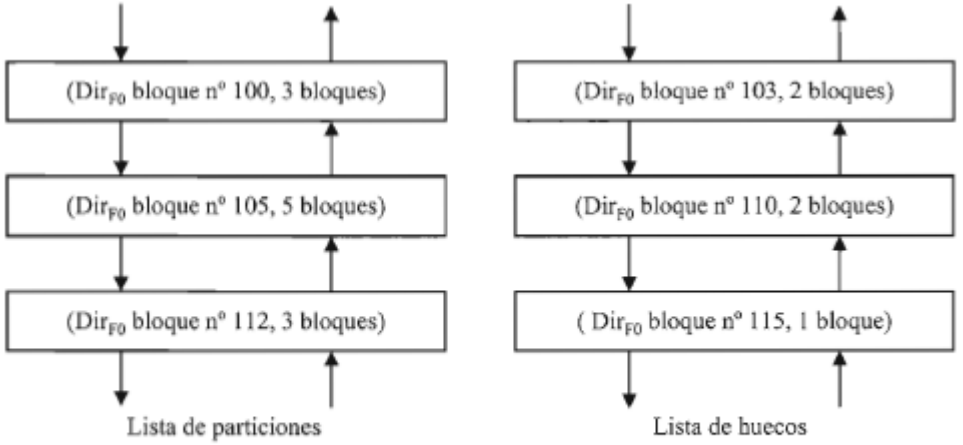


Figura 6.8 – Parte de la lista de particiones y la lista de huecos asociadas a la memoria principal de la Figura 6.6

Asignación de espacio de memoria principal

- Algoritmo del primer ajuste (*first* fit).
- Algoritmo del siguiente ajuste (*next fit*).
  - Se inicia desde donde finalizó la búsqueda anterior
- Algoritmo del mejor ajuste (*best fit*).
- Algoritmo del peor ajuste (*worst fit*)
- **Traducción de direcciones y protección**
  - *Registro base* y *registro límite*

Paginación

- Se suprime el requisito de la asignación contigua de memoria física a un programa
- La memoria física se divide conceptualmente en una serie de porciones de tamaño fijo, llamadas **marcos de página**
- El espacio de direcciones virtuales de un proceso se divide además en bloques de tamaño fijo del mismo tamaño llamados **páginas**
- La asignación de memoria consiste en hallar un número suficiente de **marcos de página** sin utilizar para cargar en ellos las **páginas** del proceso solicitante

$$N_{MP} = \text{floor}\left(\frac{C_{MP}}{S_P}\right)$$

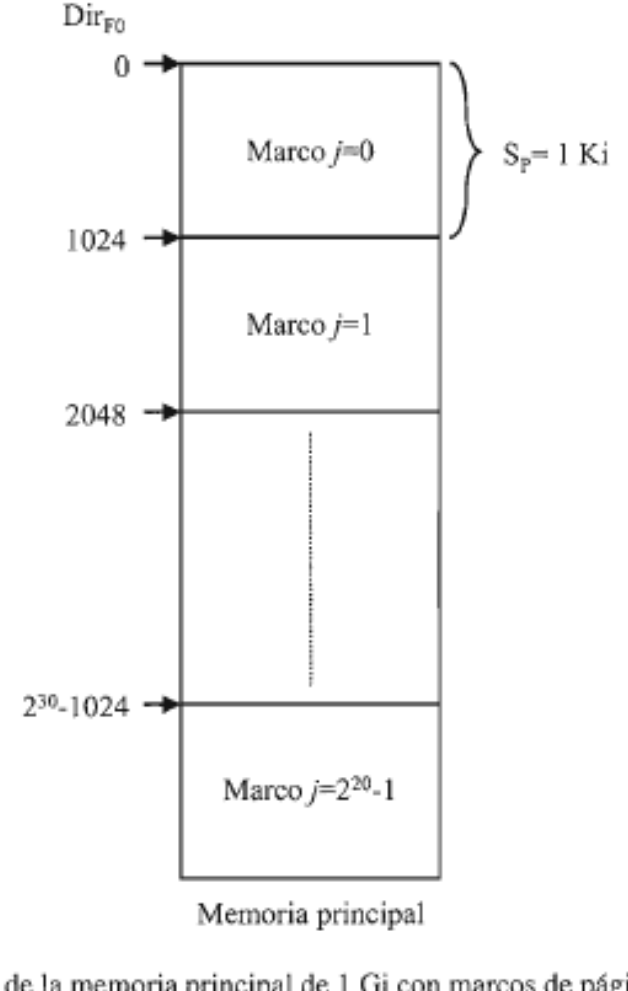


Figura 6.9 – Estructura de la memoria principal de 1 Gi con marcos de página de 1 Ki del Ejemplo 6.9

Espacio de direcciones

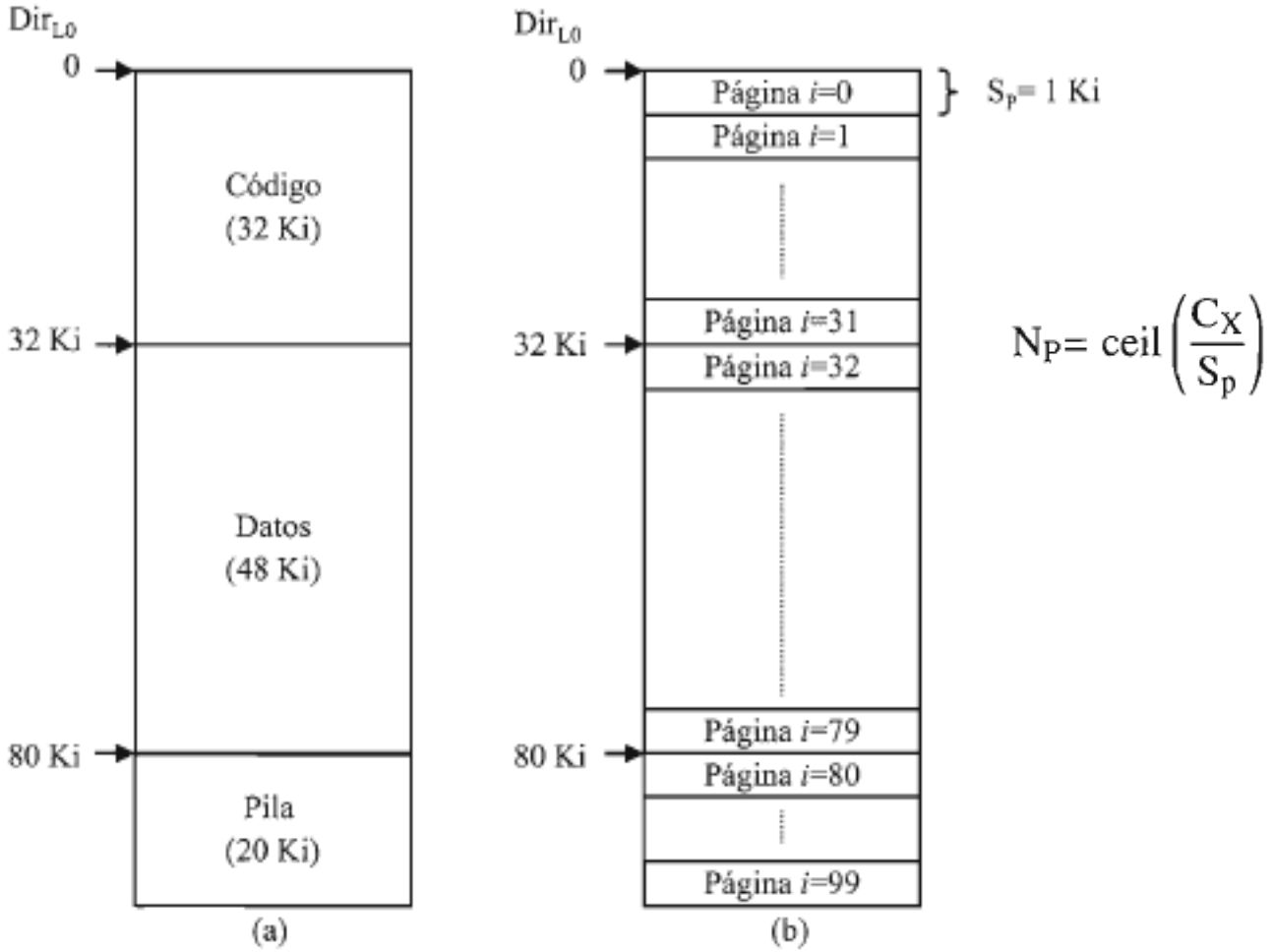
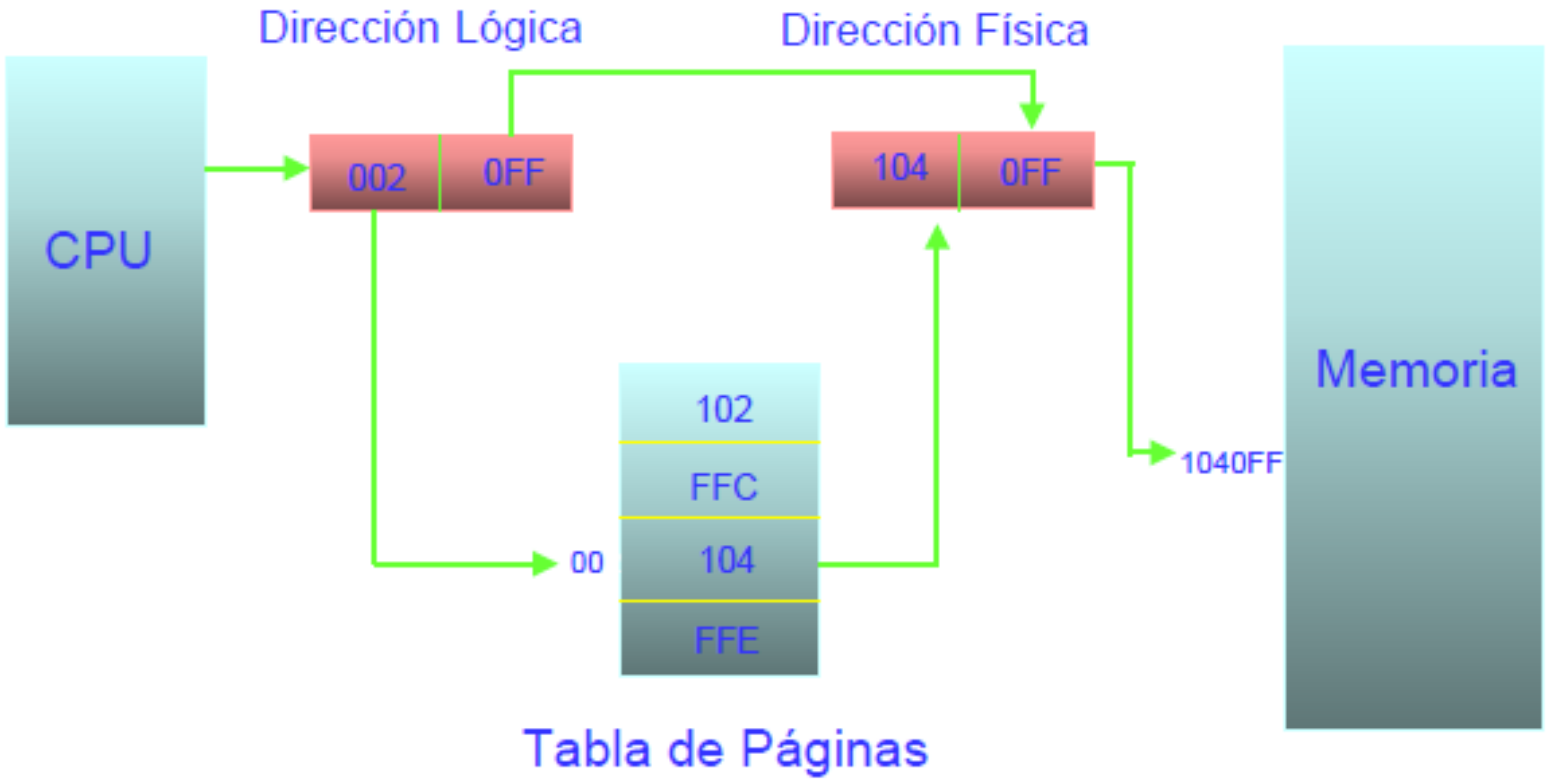


Figura 6.10 – Espacio de direcciones lógicas de un cierto proceso A de 100 Ki: (a) Regiones del proceso. (b) Páginas del proceso

Paginación

- Puesto que cada página se hace corresponder separadamente, los diferentes marcos de página asignados a un solo proceso no necesitan ocupar áreas contiguas de memoria física
- Cada dirección virtual esta dividida en dos partes:
  - El número de página
  - El desplazamiento dentro de esa página
- Para efectuar la traducción de las direcciones, el número de página se emplea como un índice en la **Tabla de Páginas** y la dirección física se construye

Traducción de direcciones de paginación



Paginación

- Si el tamaño de un proceso dado no es múltiplo entero del tamaño de la página, el último marco de página puede estar parcialmente inutilizado, esto se llama **fragmentación** o ruptura de página
- Para acelerar el proceso de traducción de direcciones:
  - La utilización de registros específicos para la tabla de páginas
    - Es adecuada si la tabla de páginas es pequeña.
  - Un método comúnmente utilizado es utilizar una memoria asociativa de alta velocidad para almacenar un subconjunto de las entradas de la tabla del mapa de páginas más frecuentemente utilizadas.
    - Esta memoria se denomina buffer para apartado de traducciones (BAT)

Marcos en memoria principal

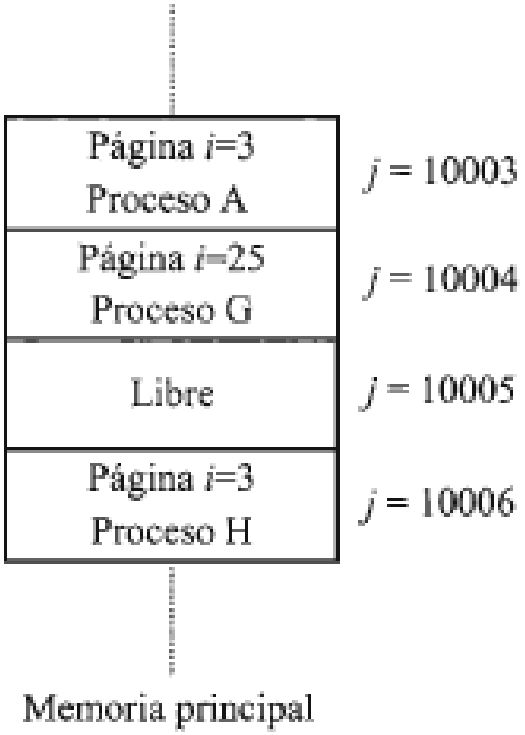


Figura 6.11 – Ejemplo del contenido en un determinado instante de tiempo de cuatro marcos de una memoria principal en un sistema con paginación con páginas de 1 Ki

Formato direcciones

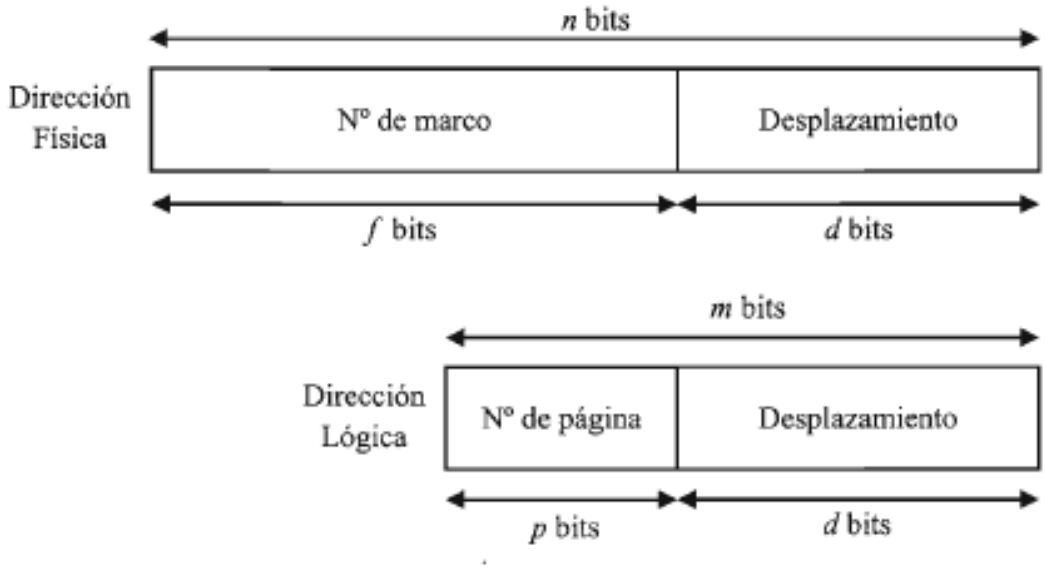


Figura 6.12 – Formato de las direcciones físicas y de las direcciones lógicas en la técnica de paginación simple ($m \leq n$)

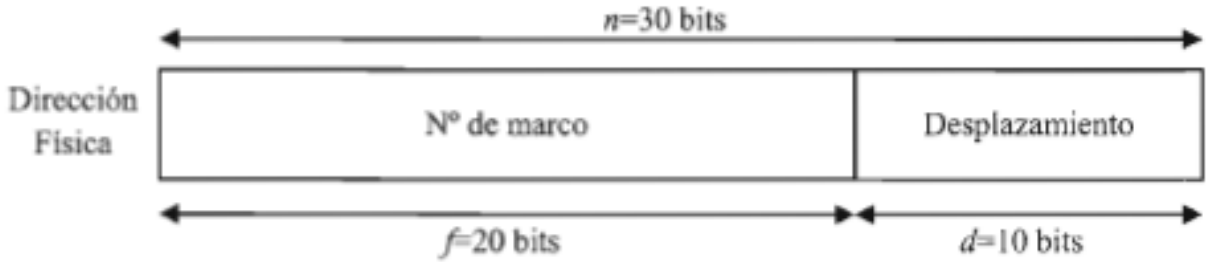


Figura 6.13 – Formato de una dirección física para una memoria de 1 Gi con marcos de página de 1 Ki

Marcos de memoria principal

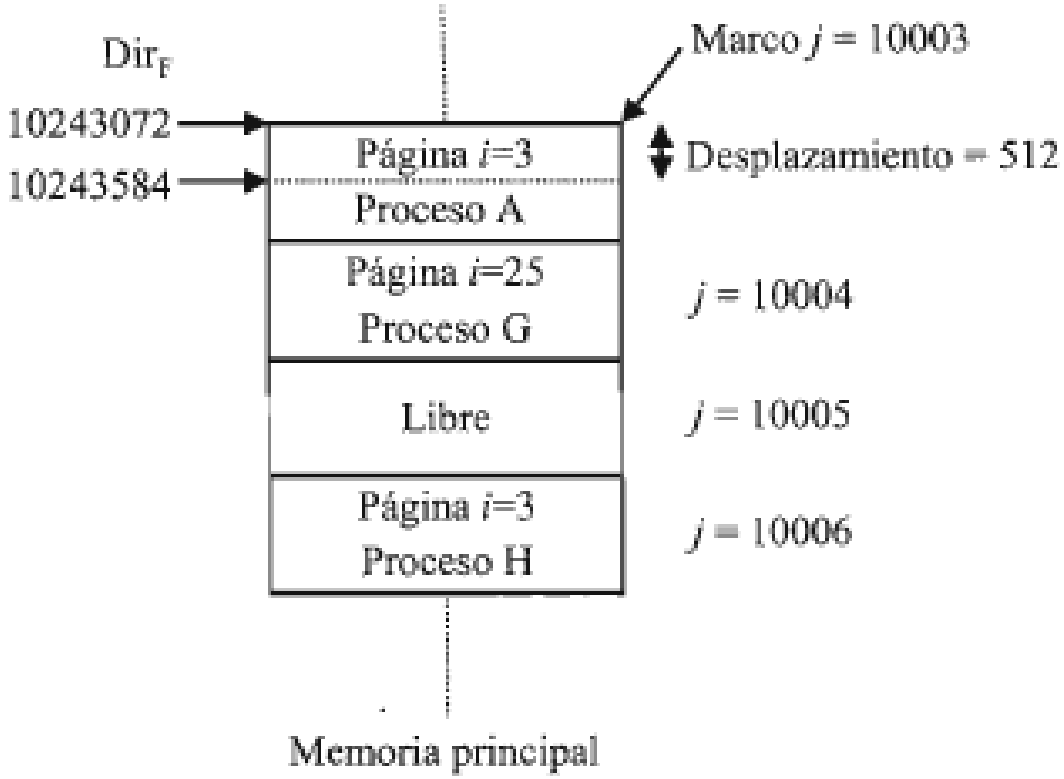


Figura 6.14 – Contenido en un determinado instante de tiempo de cuatro marcos de una memoria principal en un sistema con paginación con páginas de 1 Ki. Ubicación de la dirección física 10243584_{10}

Dirección

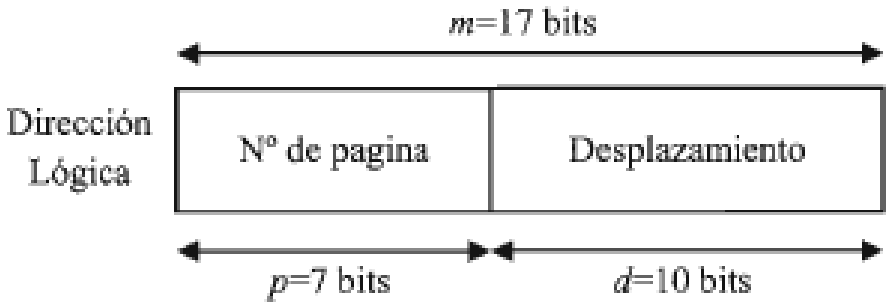


Figura 6.15 – Formato de una dirección lógica de proceso de 100 Ki con un tamaño de página de 1 Ki

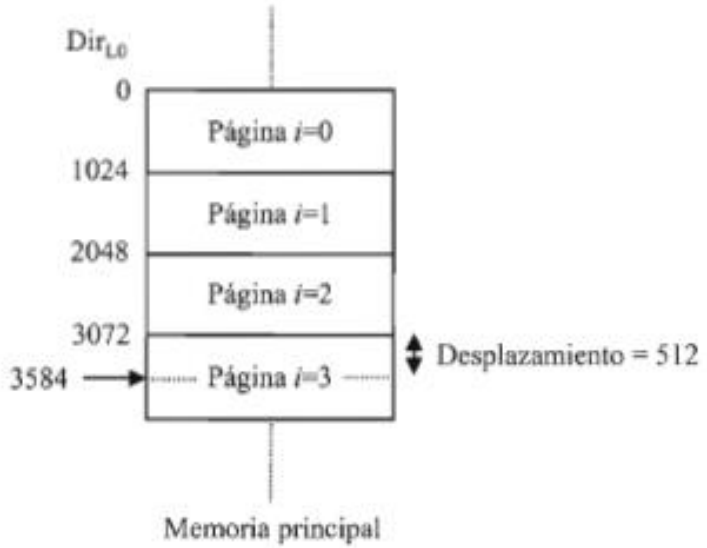


Figura 6.16 – Ubicación de la dirección lógica $Dir_L = 3584_{10}$ dentro del espacio de direcciones lógicas del proceso A

Tabla de página

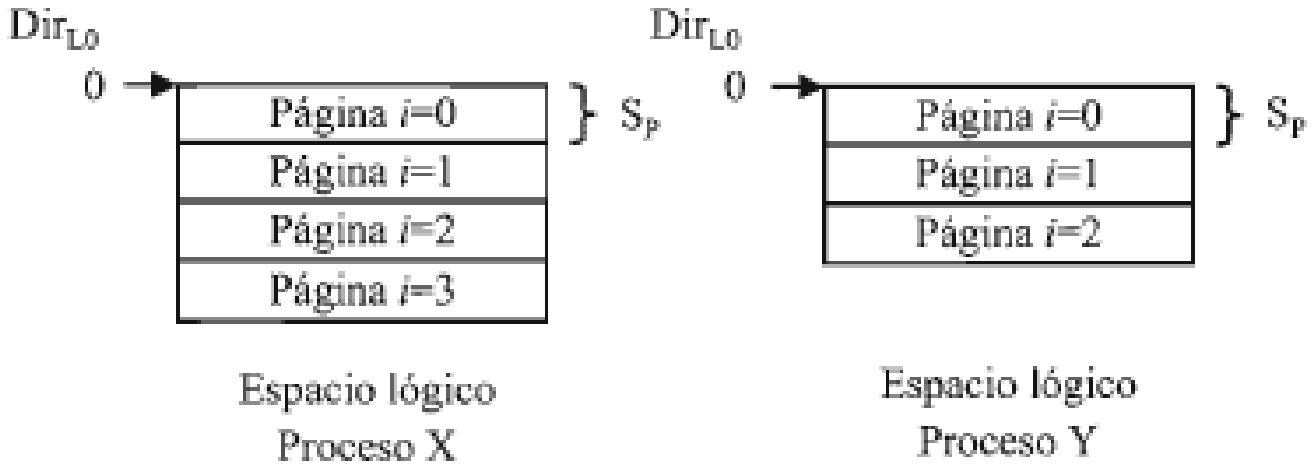


Figura 6.17 – Espacio lógico de los procesos X e Y del Ejemplo 6.14 en el área de intercambio

Estructuras necesarias

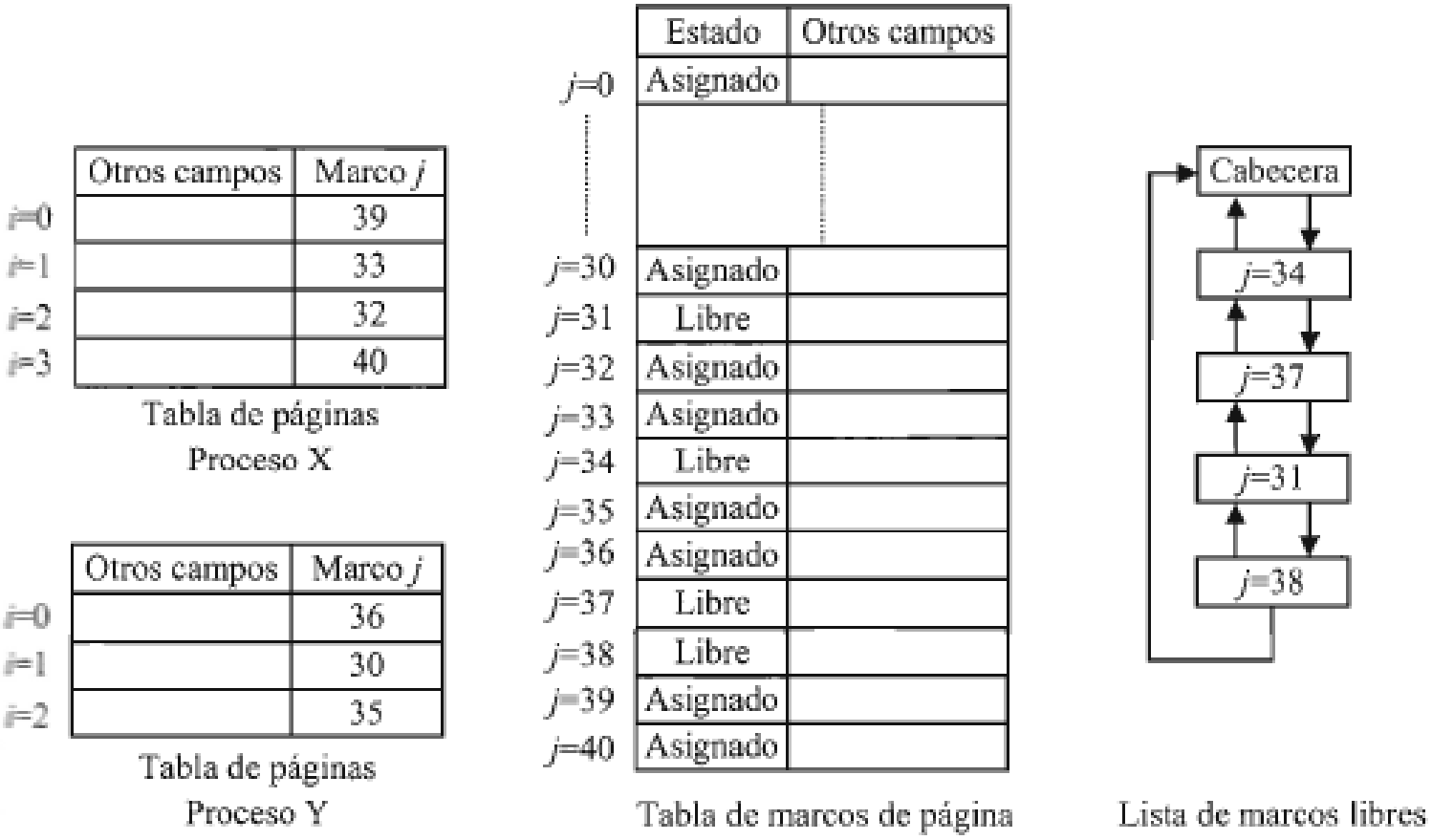


Figura 6.18 – Contenido en el instante *t* de las estructuras de datos en el espacio del núcleo de un determinado sistema operativo para la implementación de la técnica de paginación simple en el Ejemplo 6.14

Memoria principal

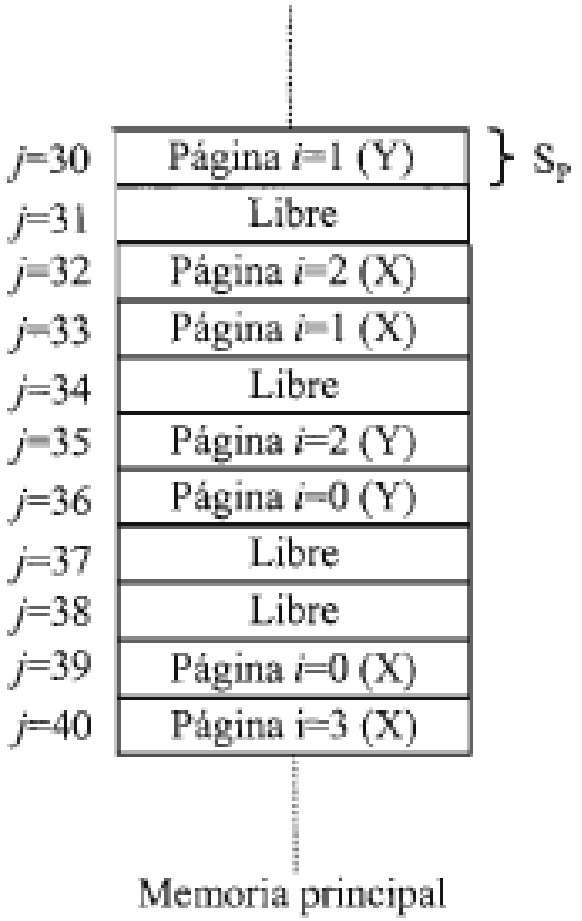


Figura 6.19 – Contenido en el instante t de los marcos $j = 30$ a $j = 40$ de la memoria principal del Ejemplo 6.14

Ejemplo

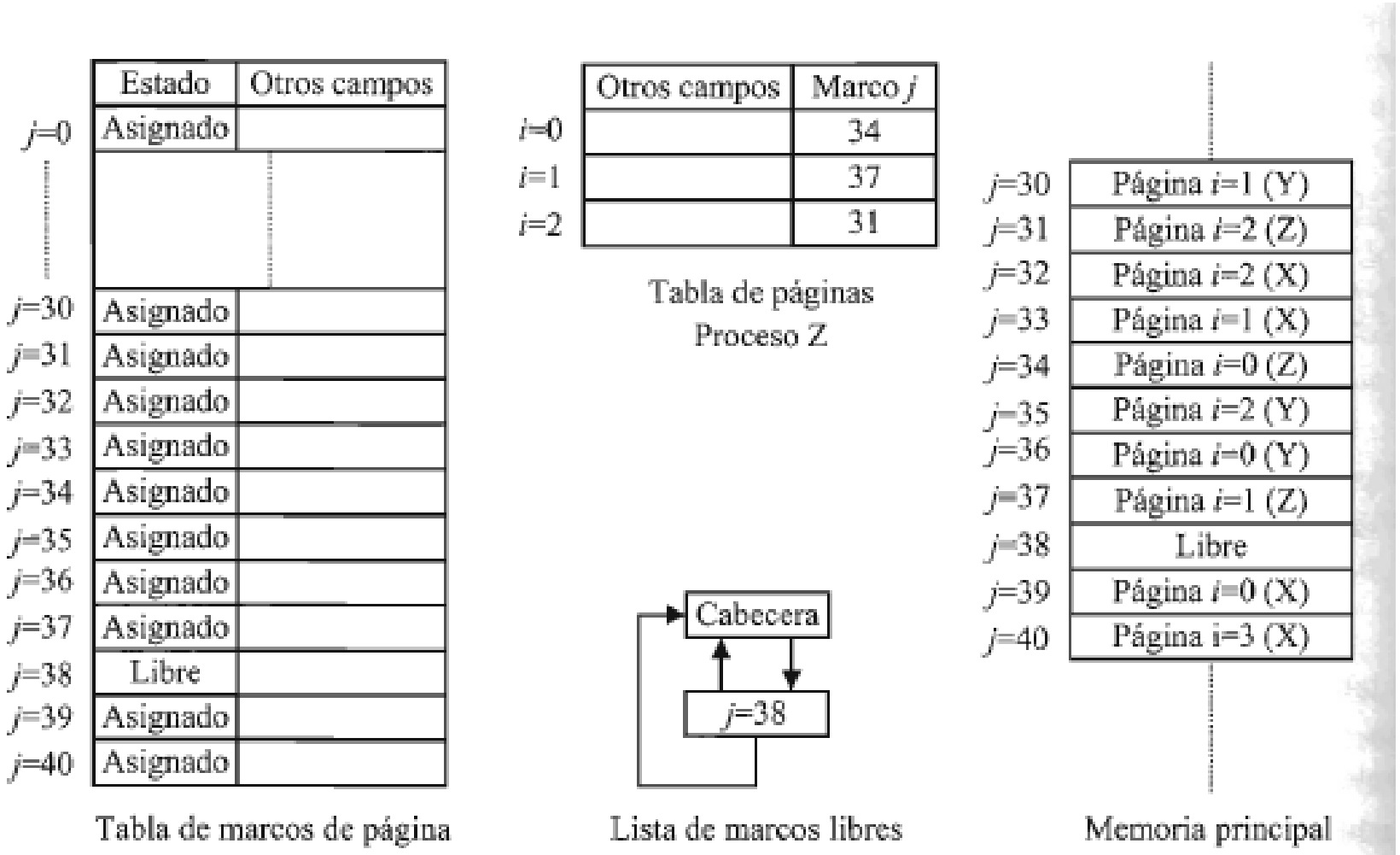


Figura 6.20 – Contenido de las estructuras de datos del sistema operativo y de la memoria principal después de crear al proceso Z del Ejemplo 6.14

Traducción de direcciones

- Registro Base.
  - Se utiliza para almacenar la dirección física de comienzo de la tabla de páginas.
- Banco de Registros.
  - Se utiliza para almacenar una copia completa de la Tabla de Páginas.
- Buffer de traducción de vista lateral (*Translation Lookaside Buffer, TLB*).
  - Se utiliza para almacenar algunas entradas de la tabla de páginas.

Traducción de direcciones con un Registro Base

En el Registro Base se almacena la dirección origen de la Tabla de páginas

Hay que acceder a memoria principal para leer la tabla de páginas y otro acceso para leer la memoria (dos acceso a memoria)

Un costo alto en tiempo

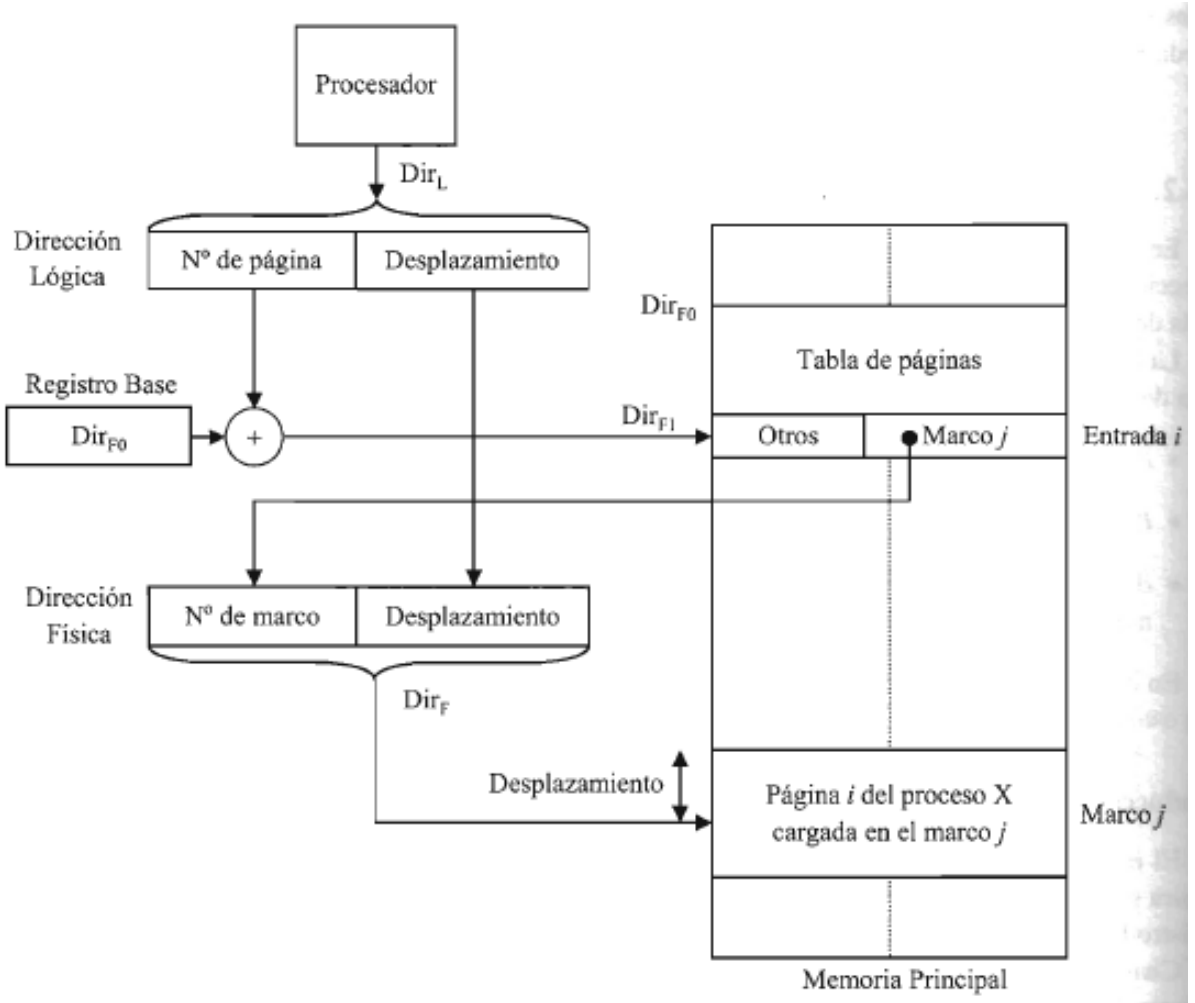


Figura 6.21 – Traducción de direcciones en paginación con un registro base

Traducción de direcciones con un Banco de Registros

Un a copia de la tabla de páginas del proceso que se va a ejecutar se guarda en los registros

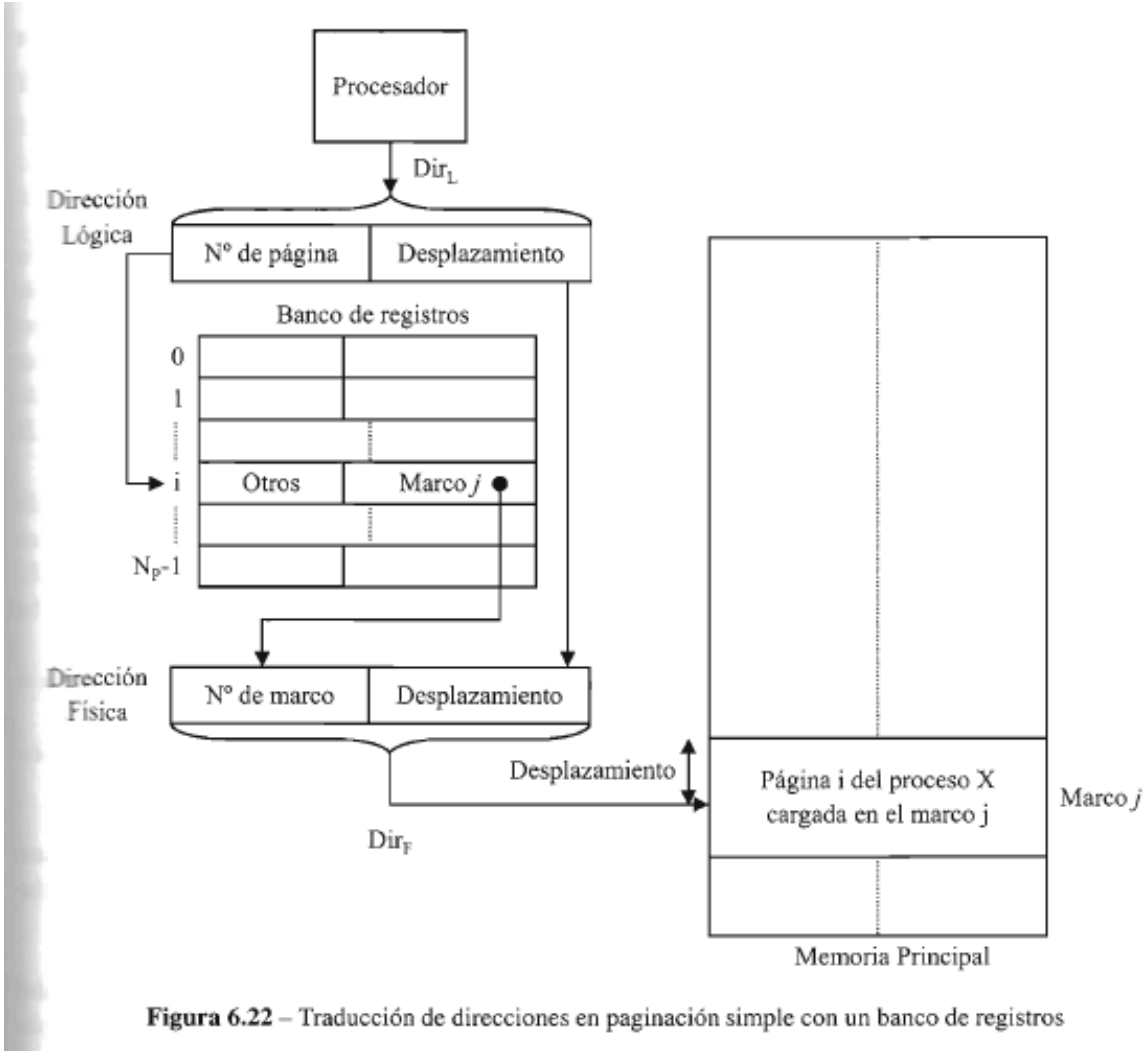


Figura 6.22 – Traducción de direcciones en paginación simple con un banco de registros

Traducción de direcciones con un TLB

- Un **TLB** es una memoria caché especial de alta velocidad
- Su funcionamiento es similar al de memoria caché del procesador.
- Cada entrada del **TLB** contiene una copia de una entrada de la Tabla de Páginas del proceso actualmente en ejecución.
- El uso de un **TLB** es una solución para la traducción de direcciones a medio camino entre el uso de un registro base y el uso de un banco de registros, ya que permite ejecutar procesos de espacios lógico grandes y solo realiza un acceso a memoria principal en la traducción de una dirección si se produce un fallo en el **TLB**.

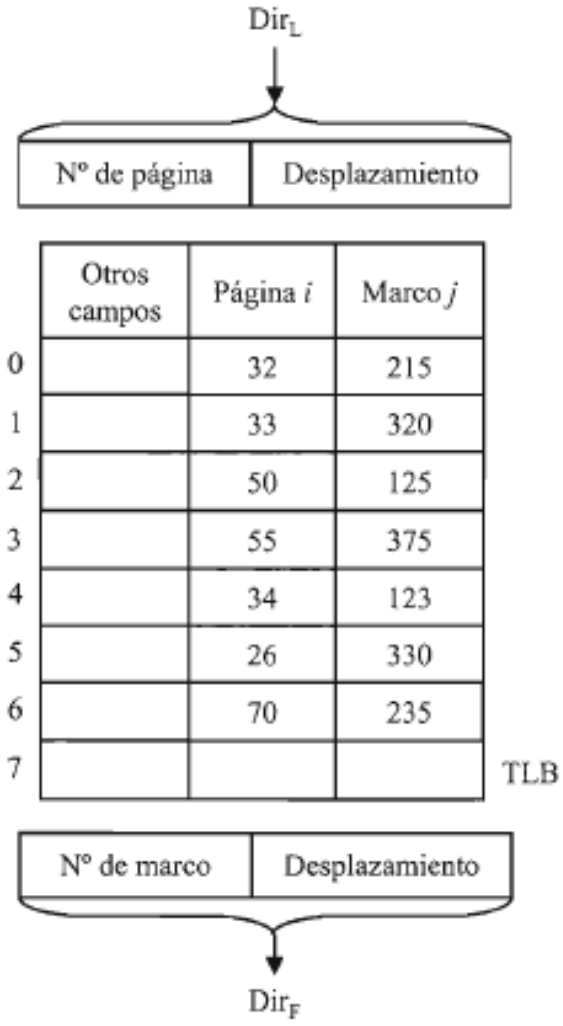


Figura 6.23 – Contenido del TLB del Ejemplo 6.15 en un cierto instante de tiempo

Tablas de páginas paginadas

- Una forma de tratar a las tablas de páginas grandes es descomponerla también en páginas, se tiene una *tabla de páginas paginada.*
- La tabla de páginas se fragmenta y puede ubicarse memoria principal de forma no contigua
- *Páginas ordinarias*
- Contienen instrucciones y datos del espacio lógico de un proceso.
- *Páginas de tablas de páginas*
- Contienen entradas de una tabla de páginas.

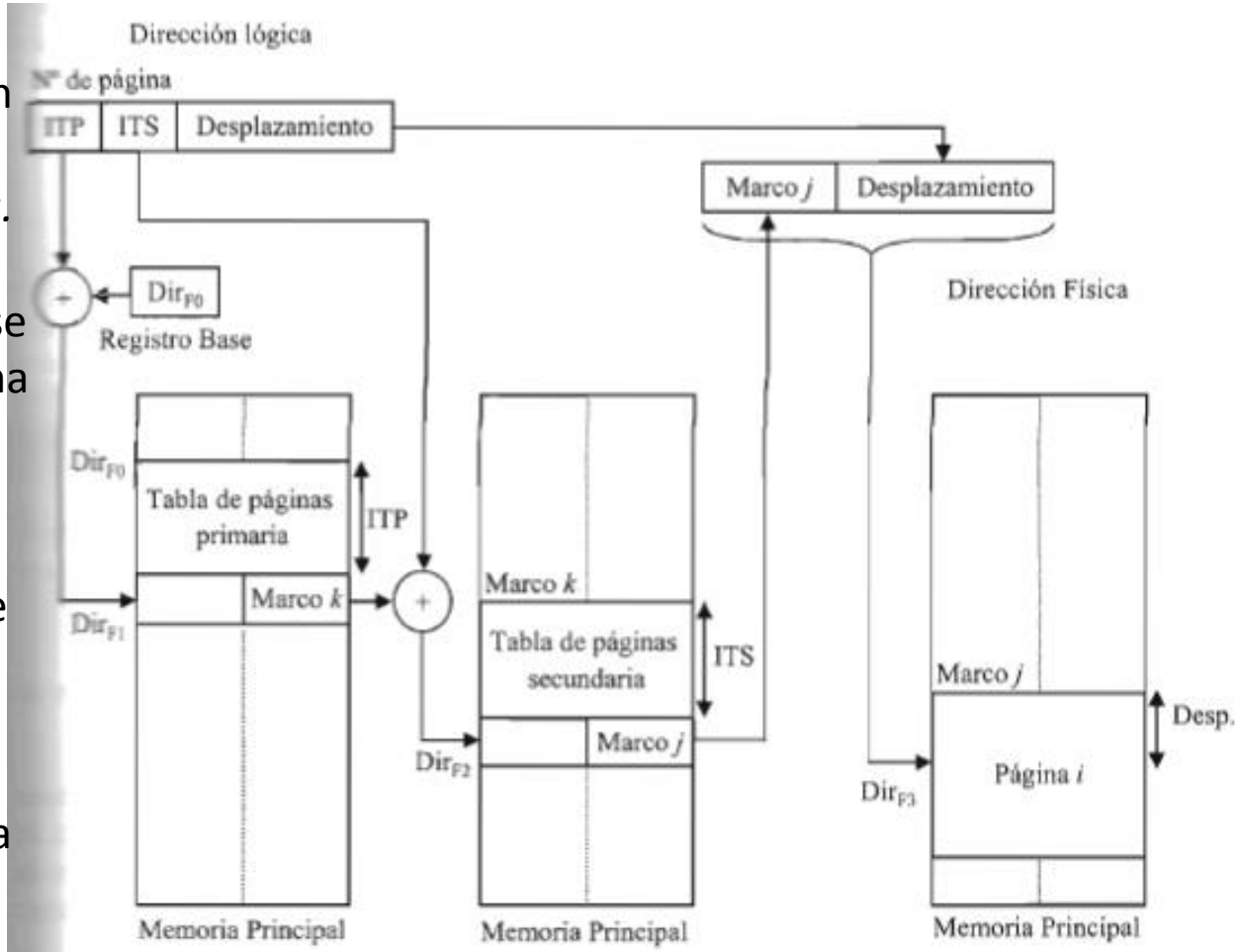


Figura 6.24 – Traducción de direcciones lógicas usando una tabla de páginas paginada

Formato de la dirección Tabla de Página Paginada

Índice Tabla Primaria (ITP)

- Contiene el número de la entrada de la tabla de páginas de primer nivel donde hay que buscar el marco de memoria que contiene a la página que ubica a la tabla de páginas de segundo nivel

Índice Tabla Secundaria (ITS)

- Contiene el número de la entrada de la tabla de páginas de segundo nivel donde hay que buscar el marco de memoria que contiene a la página ordinaria referencia

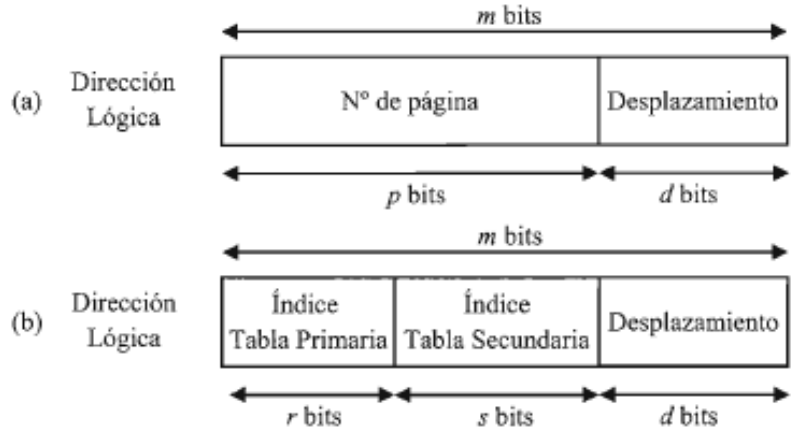


Figura 6.25 – Formato de una dirección lógica: a) Un único nivel de paginación (tabla de páginas sin paginar). b) Dos niveles de paginación (tabla de páginas paginada)

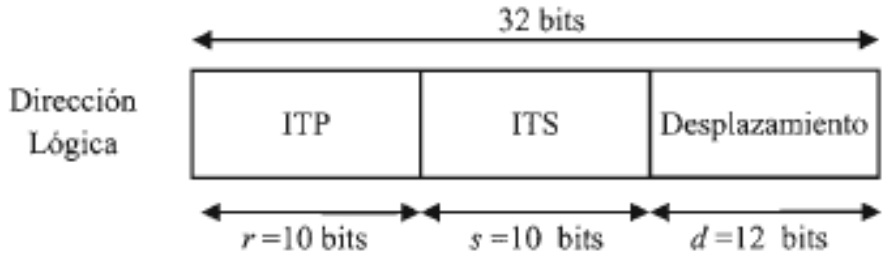


Figura 6.26 – Formato de una dirección lógica para los datos del Ejemplo 6.17

Tablas de Páginas Invertidas

Una tabla de páginas invertida tiene asociada una **entrada por cada marco *j* de memoria principal**, vez de una entrada por cada página *i* de un proceso

El número de entrada es fijo y es igual al número de marcos

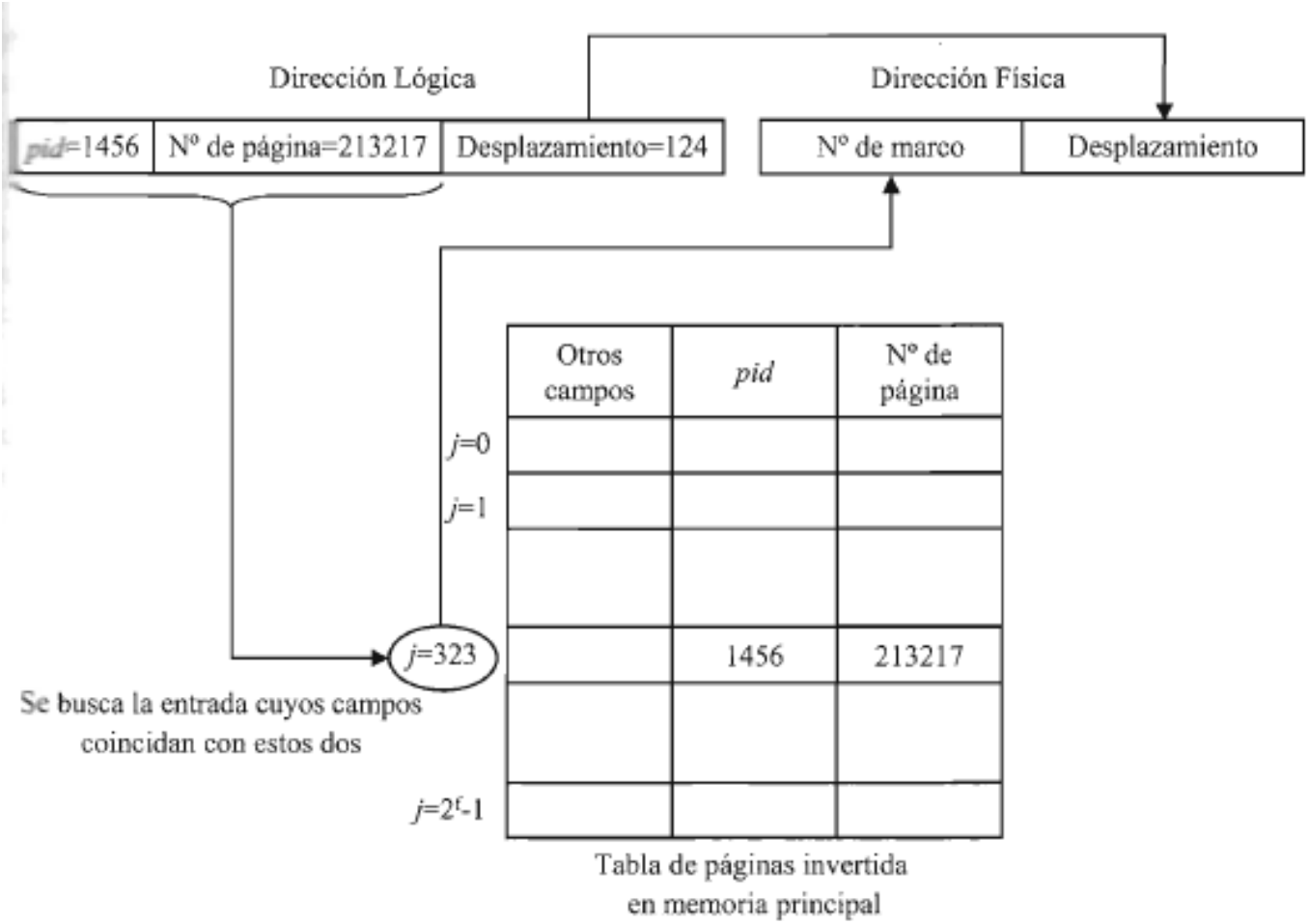


Figura 6.27 – Tabla de páginas invertida del Ejemplo 6.18

Protección y compartición

- Protección
  - Varios bits que en función de su valor permiten establecer el tipo de acceso que se permiten sobre dicha página *i:*
    - Solo lectura, lectura y escritura, ejecución
- Compartición de páginas
  - *Código reentrante* (que no se puede modificar)
    - Se ahorraría espacio en memoria si solo se mantuviese cargada en memoria principal una única copia de las páginas de código de programa.
  - *Contador de referencias*
    - Indica *el* número de entradas en las tablas de páginas que están referenciando a una página física compartida

Ventajas e inconvenientes

- No produce fragmentación externa
- *Produce una sobrecarga pequeña*
- *No necesita intervención humana*
- *Permite compartir entre varios procesos un código común*
- *Inconveniente*
  - *La fragmentación interna*

Segmentación simple

- El programa es dividido por el compilador en ***segmentos***.
  - Cada **segmento** es una entidad lógica conocida por el programador asociada a determinada estructura de datos o a un módulo, pero nunca a una mezcla de ambos
- Cada segmento tiene asignado un nombre (código principal, subrutinas, datos, pila, etc) y una longitud.
- El compilador compila cada segmento comenzando por la dirección lógica 0.
  - Cada segmento tiene su propio espacio de direcciones lógicas.

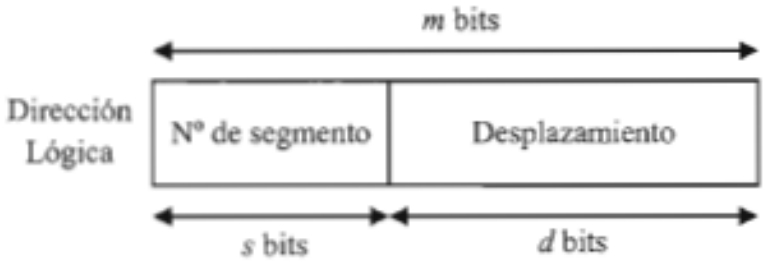


Figura 6.28 – Formato de una dirección lógica en un esquema de gestión de memoria mediante segmentación simple

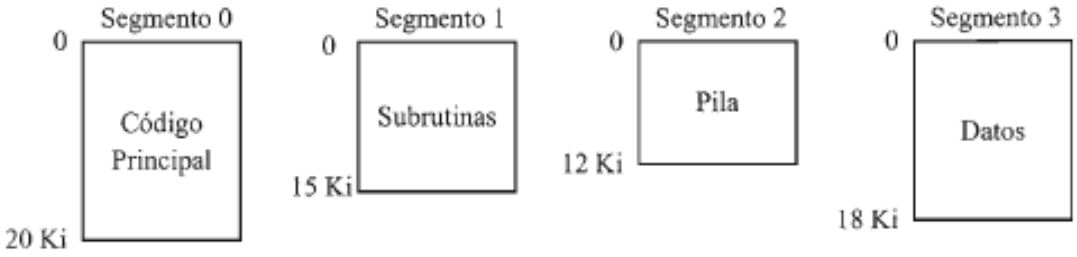


Figura 6.29 – Segmentos de memoria de un cierto proceso

Ejemplo

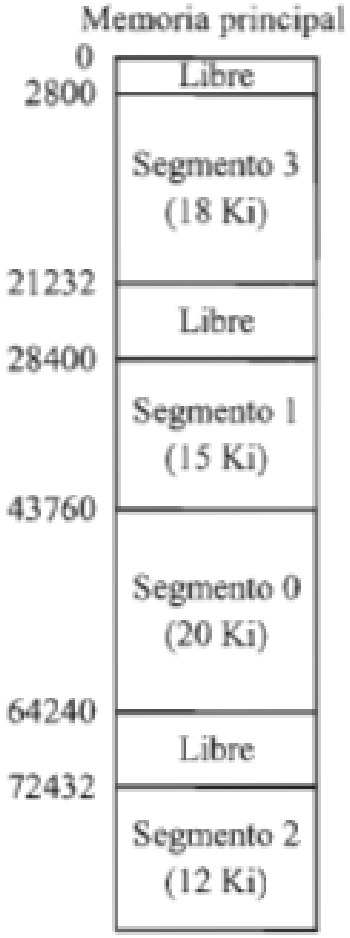


Tabla de segmentos del proceso A

	Base	Longitud	Otros Campos
0	43760	20480	
1	28400	15360	
2	72432	12288	
3	2800	18432	

Figura 6.30 – Ubicación en memoria principal de los segmentos del proceso A del Ejemplo 6.20

Traducción de direcciones con Registro Base

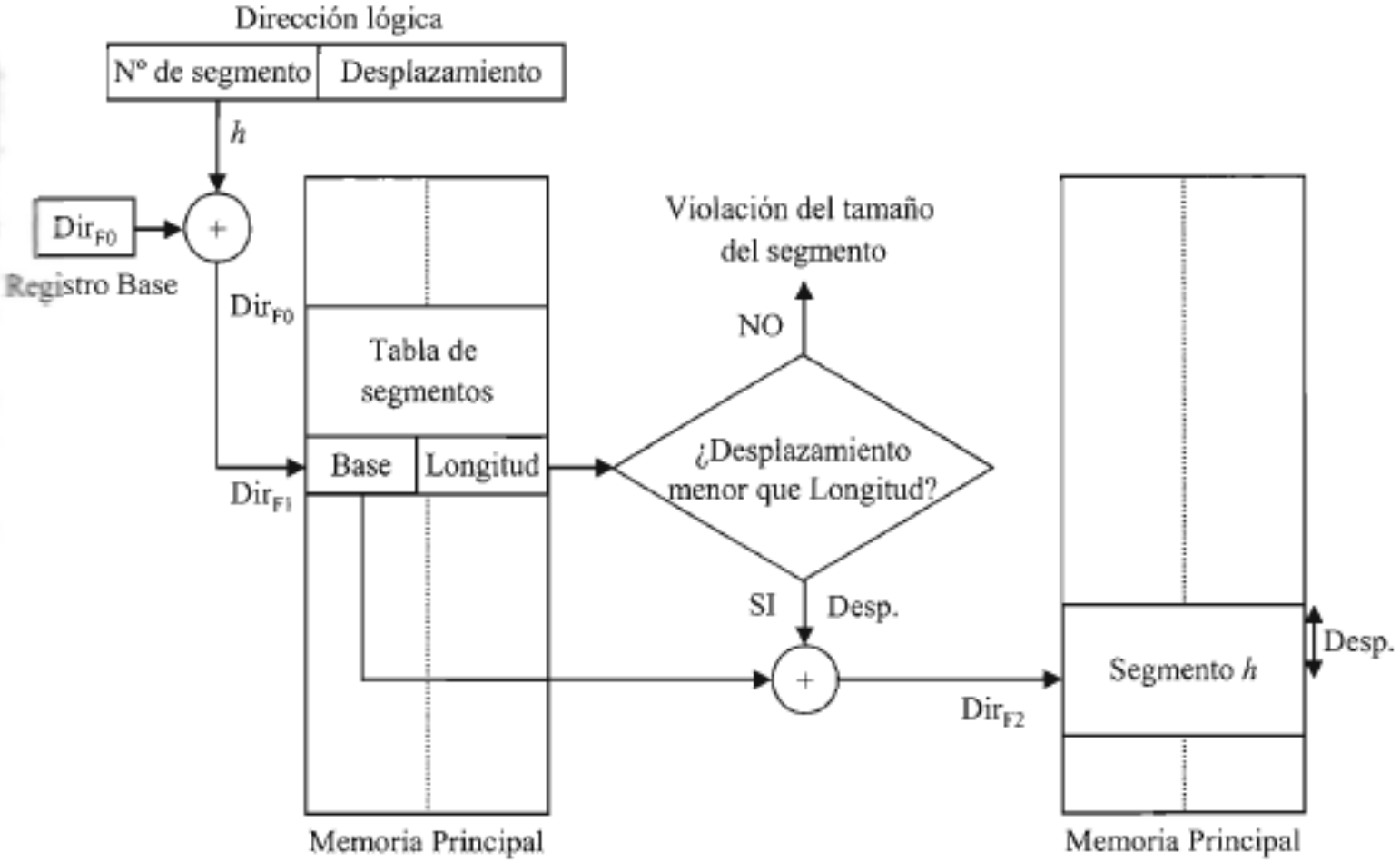


Figura 6.31 – Traducción de direcciones en un esquema de segmentación simple con un registro base

Protección, ventajas e inconvenientes

- Protección
  - Campos en la tabla de segmentos
  - Registro límite
- Compartición
  - Código compartido por segmentos
- Ventajas
  - Produce una fragmentación interna despreciable
  - Soporta la visión modular que el programador posee de su programa
  - Permite manejar con facilidad estructuras de datos que crecen
  - Facilita la protección y compartición de las diferentes partes de un programa
- Inconvenientes
  - Produce fragmentación externa
  - Aumenta la sobrecarga del sistema

Segmentación con paginación simple

- La segmentación y la paginación se pueden combinar para aprovecharse de las ventajas de cada técnica y reducir sus inconvenientes

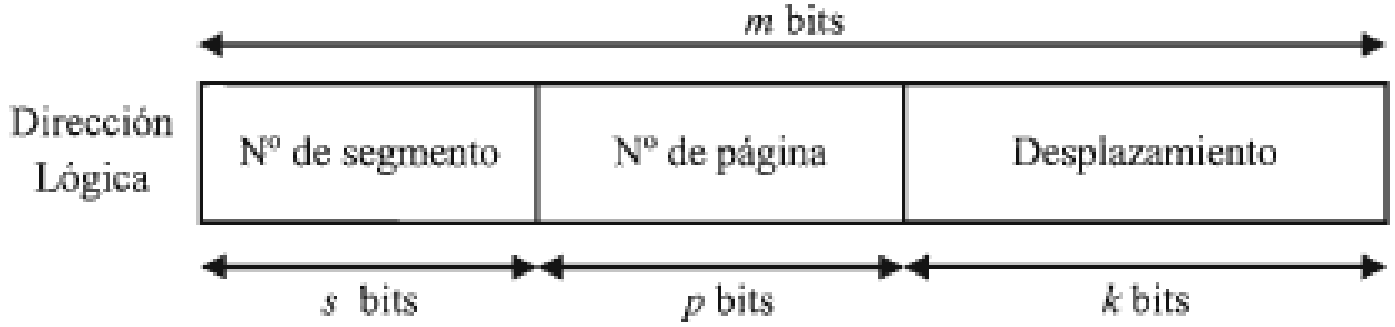
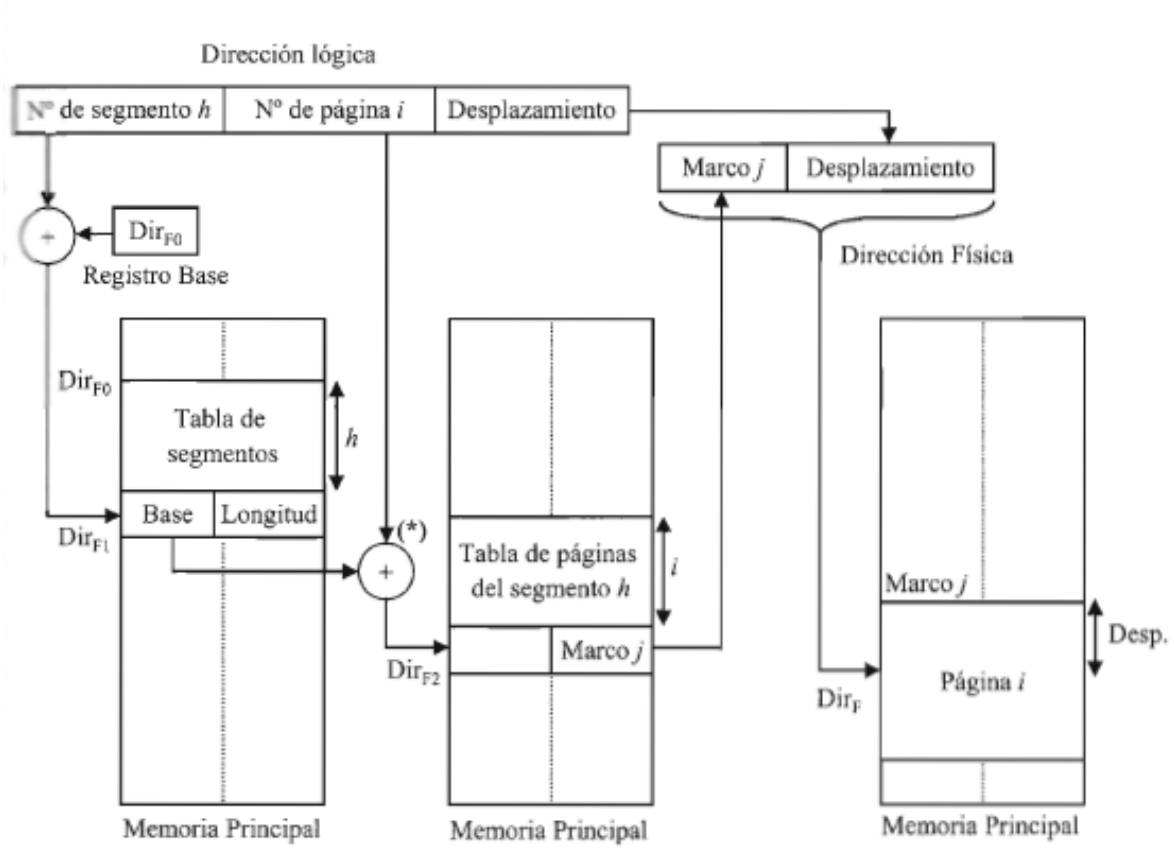


Figura 6.32 – Estructura de una dirección lógica en la técnica de segmentación con paginación

Traducción de direcciones con Segmentación Paginada



(*) Por simplificar en este esquema se ha omitido el paso de comprobar si el desplazamiento es menor que la longitud.

Figura 6.33 – Esquema de traducción de direcciones en un sistema de segmentación con paginación

4. *(2 p)* Un cierto sistema operativo gestiona la memoria principal mediante paginación simple. El tamaño de página utilizado es de 2048 bytes. La memoria física disponible para los procesos es de 8 MiB. Al sistema llegan dos procesos A y B cuya carga en la memoria principal consume 31566 bytes y 18432 bytes, respectivamente. Determinar la fragmentación interna y la fragmentación externa que provoca la carga de cada proceso.

La técnica de paginación simple **no produce fragmentación externa**, solo produce *fragmentación interna* debida al posible espacio libre que puede existir en la última página del espacio de direcciones lógicas de un proceso.

1) Cálculo de la fragmentación interna del proceso A:

Para calcular la fragmentación interna en primer lugar se va a calcular el número de páginas N_P en que se descompone el espacio de direcciones lógicas del proceso A:

$$N_P = \text{ceil}\left(\frac{C_X}{S_P}\right)$$

Donde C_X es el tamaño del espacio de direcciones lógicas de proceso A y S_P es el tamaño de una página. Sustituyendo valores y operando se obtiene que:

$$N_P = \text{ceil}\left(\frac{31566}{2048}\right) = \text{ceil}\,(15{,}413) = 16\ \text{páginas}$$

El espacio ocupado por estas 16 páginas ($i = 0, 1, 2, ..., 15$) es $16 \cdot 2048 = 32768$ bytes. Puesto que el espacio de direcciones lógicas del proceso A tiene un tamaño de 31566 bytes, el espacio libre existente en la última página asignada al proceso (página $i = 15$) es igual a:

$$32768 - 31566 = 1202\ \text{bytes}$$

Por lo tanto, la fragmentación interna provocada por la carga del proceso A es de **1202 bytes** o si se expresa en porcentaje el 58,69 % del tamaño de una página.

2) Cálculo de la fragmentación interna del proceso B:

$$N_P = \text{ceil}\left(\frac{18432}{2048}\right) = \text{ceil}\,(9) = 9\,\text{páginas}$$

El espacio ocupado por estas 9 páginas ($i = 0, 1, 2, ..., 8$) es $9 \cdot 2048 = 18432$ bytes. Puesto que el espacio de direcciones lógicas del proceso A tiene un tamaño de 18432 bytes, el espacio libre existente en la última página asignada al proceso (página $i = 8$) es igual a:

$$18432 - 18432 = 0 \text{ bytes}$$

Por lo tanto, la fragmentación interna provocada por la carga del proceso A es de **0 bytes**. Es decir, la carga de del proceso A no produce fragmentación interna.