

# TEMA 4

## GESTION DE LA MEMORIA

### 4.1 SISTEMAS ELEMENTALES DE GESTIÓN DE LA MEMORIA

El esquema de gestión de memoria más simple es aquel que no tiene ningún gestor.

El usuario se encuentra con la máquina desnuda y tiene un control completo sobre el espacio total de memoria.

Ventajas :

- 1) Da máxima flexibilidad al usuario, ya que puede controlar el uso de la memoria como desee.
- 2) Tiene máxima simplicidad y coste mínimo.
- 3) No es necesario disponer de un computador específico ni tan siquiera del S.O.

Limitaciones:

- 1) No da ningún servicio.
- 2) El S.O. no tiene control sobre las interrupciones.
- 3) No existe un monitor residente para procesar las llamadas al sistema o para el tratamiento de errores.

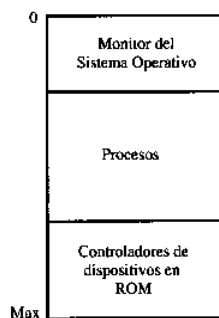
En la actualidad ya no se utiliza, salvo en algunos sistemas dedicados en los que el usuario requiere flexibilidad y simplicidad, y el usuario programa el mismo sus propias rutinas de servicio.

El siguiente esquema en simplicidad se obtiene al dividir la memoria en dos secciones, una para los procesos del usuario y otra para la parte del S.O. que debe estar residente en memoria (monitor).



Monitor de proceso único.

Bastante común, con alguna variante, en los sistemas operativos de los microcomputadores de proceso único. Primeras versiones de MS-DOS, CP/M.



En MS-DOS, la memoria se divide en tres secciones:

- La parte inferior de la RAM, destinada al S.O.
- La parte central dedicada al único proceso de usuario.
- La parte superior, en ROM, para los controladores de dispositivos. BIOS.

Estos esquemas suponen que sólo se ejecuta un proceso a la vez.

Actualmente se ha impuesto la multiprogramación. Razones para usar multiprogramación:

- 1) Facilitar la programación de una aplicación al dividirla en varios procesos.
- 2) Dar servicio interactivo a varios usuarios al mismo tiempo, o el optimizar el tiempo de procesador cuando un proceso está a la espera de una operación de entrada/salida.

## TEMA 4

# GESTION DE LA MEMORIA

### 4.2 GESTIÓN DE LA MEMORIA CON PARTICIONES FIJAS

Consiste en dividir la memoria física disponible en varias particiones y asignar cada una de las partes a un proceso.

Se denomina MFT (Multiprogramming with a Fixed of Tasks).

La gestión de memoria con particiones fijas supone que la división de ésta se ha realizado con anterioridad a la ejecución de los programas, pudiendo hacerla el administrador del sistema de forma manual al iniciar una sesión en la maquina.

El numero y tamaño de las particiones se determinan teniendo en cuenta los factores siguientes:

- 1) Capacidad de la memoria física y el grado de multiprogramación deseado (número máximo de procesos activos en el sistema.)
- 2) Tamaño típico de los procesos ejecutados mas frecuentemente.

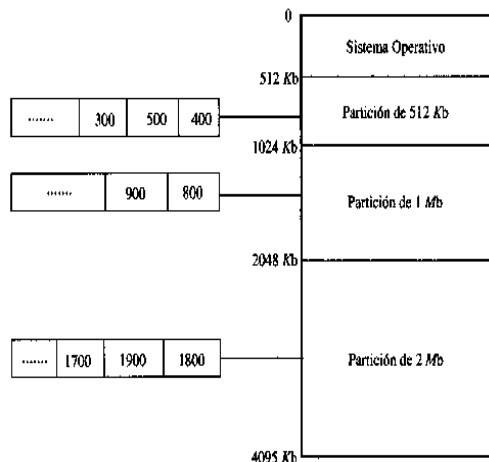
#### 4.2.1 PRINCIPIO DE OPERACIÓN

- a) Cuando llega una tarea ésta se pone en una cola de tareas.
- b) El planificador de tareas tiene en cuenta los requerimientos de memoria de cada una y las particiones de memoria disponibles.
- c) Si una tarea tiene espacio disponible en memoria, se ubica (o reubica) en una partición, y puede competir por el uso de la CPU.
- d) Cuando se termina una tarea, se libera la partición de memoria que ocupa, pudiendo el planificador asignar esta partición a otra tarea de la cola.

Tipos de organizaciones :

- a) **Clasificar todas las tareas según sus requerimientos de espacio.** Cada partición de memoria tendrá una cola de tareas y las tareas se incluyen en la cola de la partición de memoria correspondiente a sus exigencias de memoria.  
Desventaja: La cola de una partición grande puede estar vacía mientras la cola de una partición pequeña está llena.

#### Ejemplo



Sistema de particiones de memoria fija y colas de entrada independientes.

Tamaños de las particiones: 512Kb, 1Mb, 2Mb

Una partición de 512k reservada a la parte residente del S.O.

La cola de 512kb tiene tareas de 400,500 y 300Kb.

La cola de 1Mb de 800 y 900Kb.

La cola de 2Mb de 1800,1900 y 1700Kb.

## TEMA 4

# GESTION DE LA MEMORIA

- b) **Mantener todas las tareas en una sola cola.** El planificador selecciona la próxima tarea para ser ejecutada y espera hasta que se disponga de una región de memoria del tamaño requerido.

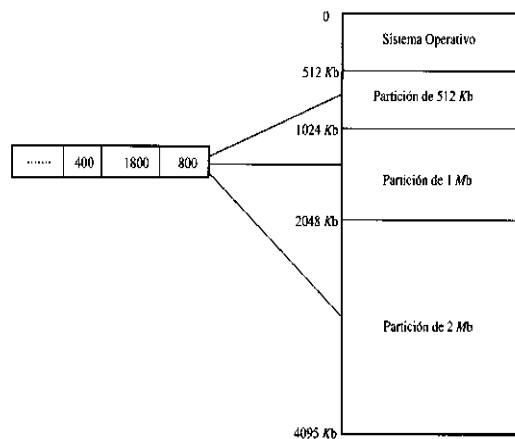
Variantes:

- b.1) Cargar y ejecutar la tarea más cercana al frente de la cola que se ajuste al tamaño de la partición liberada.
- b.2) Buscar en la cola de entradas la tarea más grande que se ajuste a la partición liberada.

Inconvenientes: Discrimina las tareas pequeñas.

Soluciones:

1. Tener una partición de tamaño reducido para la ejecución de tareas pequeñas.
2. Fijar como regla que una tarea no se excluya más de un número de veces fijado a priori por el S.O.



### 4.2.3 INTERCAMBIO

Swapping. Hace referencia a las operaciones de eliminar de la memoria principal procesos suspendidos, llevarlos al disco y cargar del disco a la memoria principal procesos para su ejecución.

Cuando el planificador decide admitir un nuevo proceso para el que no hay memoria, se puede invocar al intercambiador para liberar una partición.

Responsabilidades del intercambiador:

- Selección de los procesos que hay que eliminar de la memoria principal.
- Selección de los procesos que hay que cargar en la memoria principal.
- Asignación y gestión del espacio de intercambio.

La selección de procesos en la memoria principal que hay que intercambiar se hace entre aquellos que ocupan particiones suficientemente grandes para los procesos a los que se les debe asignar espacio en memoria.

Cuestiones a considerar :

- a) Prioridad. Son más aptos para intercambiar aquellos con baja prioridad.
- b) Estado de espera. Procesos que esperan sucesos lentos tienen mayor probabilidad de ser suspendidos durante un tiempo suficientemente grande.
- c) Tiempo de permanencia en memoria y si se ha ejecutado mientras ha estado en memoria.

# TEMA 4

## GESTION DE LA MEMORIA

La elección de los procesos que hay sin cargar en la memoria principal se basan en:

- 1) Tiempo que ha permanecido en memoria secundaria.
- 2) Prioridad.
- 3) Satisfacción del tiempo mínimo de permanencia en disco desde la descarga.

### 4.2.4 ARCHIVO DE INTERCAMBIO

Un proceso ejecutado parcialmente tiene una imagen en tiempo de ejecución diferente de la inicial almacenada en el disco.

El estado dinámico del proceso durante su ejecución al ser intercambiado debe registrarse para su posterior reanudación.

La imagen modificada en tiempo de ejecución no debe sobrescribir a la imagen del proceso estático sobre el disco.

Se debe disponer de un archivo de intercambio separado para almacenar la imagen dinámica del proceso que se está intercambiando.

Dos opciones básicas :

- 1) Un **único archivo de intercambio** para todo el sistema, creado en la inicialización del sistema. Un proceso sólo se activa si se le puede reservar suficiente espacio de intercambio.
- 2) **Diferentes archivos de intercambio** dedicados a cada uno de los procesos. Pueden ser creados dinámicamente en el tiempo de generación del proceso o estáticamente en el tiempo de preparación del programa.

Ventajas :

- a. Elimina el problema del dimensionamiento del archivo de intercambio del sistema.
- b. Elimina los errores de sobrepasar la capacidad del archivo en tiempo de ejecución.
- c. No impone restricciones sobre el número de procesos activos.

Desventajas:

- a. Necesidad de más espacio en disco para intercambio.
- b. Acceso más lento.
- c. Direccionamiento más complicado.

Los algoritmos para la gestión del espacio disponible en el archivo de intercambio son los mismos que los usados para la gestión de la memoria principal.

### 4.2.4 REUBICACIÓN Y PROTECCION

#### REUBICACIÓN

Reubicable. Posibilidad de cargar y ejecutar un programa dado en un lugar arbitrario de memoria, lo contrario a un conjunto fijo de direcciones especificadas en el tiempo de traducción del programa.

Dos tipos de reubicaciones :

- 1) Reubicación estática. Se realiza antes o durante la carga del programa en memoria, mediante un enlazador o cargador reubicable.
- 2) Reubicación dinámica. La asignación del espacio de direcciones virtuales al espacio de direcciones físicas se realiza en tiempo de ejecución, generalmente con ayuda de dispositivos físicos destinados a este fin.

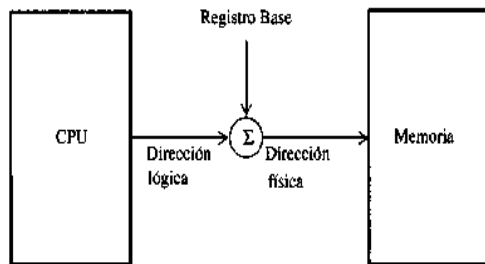
Cuando el proceso se está ejecutando, todas las referencias a direcciones de memoria se recalculan durante la ejecución de la instrucción antes de acceder a la memoria física.

Este proceso se realiza mediante registros base especializados.

Cuando se planifica la ejecución de un proceso, el registro base se carga con la dirección de inicio de la partición imagen del proceso. La dirección física será la dirección lógica dada por el proceso más el valor del registro base.

## TEMA 4

### GESTION DE LA MEMORIA



Reubicación dinámica.

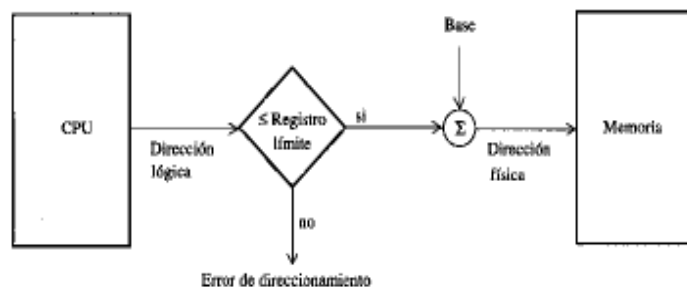
Si el registro base tiene un valor de 1000000, una instrucción CALL 1000 se interpreta como CALL 1000+1000000.

### PROTECCIÓN

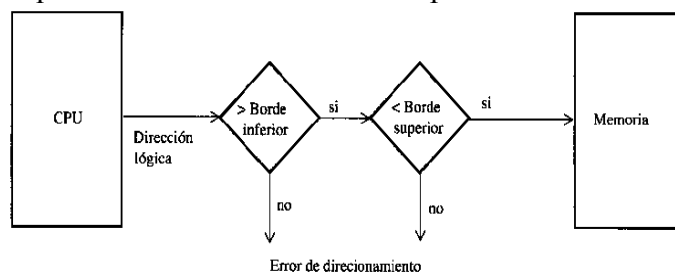
Un programa podría construir una dirección física que se salga fuera de la partición que se le ha asignado.

Soluciones:

- Considerar un registro límite que se cargaría con la longitud de la partición. Para cada referencia de memoria se comprueba en el registro límite si esta permitida. Si la dirección fuera superior, se mandaría una señal al S.O. indicando que se ha producido un error de direccionamiento. Si se usa un registro base un programa se puede trasladar de una partición a otra modificando el valor del registro limite.



- Protección hardware. Se usa principalmente para reubicación estática. Consiste en considerar registros bordes para cada partición, los cuales se cargan con las direcciones físicas de comienzo y final de la partición. Se comprueba que la dirección física esté comprendida entre los bordes.



- Comprobar el acceso en la misma memoria, asignando un código de protección a cada bloque. El computador detendría cualquier intento de un proceso en ejecución si tratase de acceder a una parte de la memoria cuyo código de protección fuera distinto al de la contraseña.

## TEMA 4

### GESTION DE LA MEMORIA

#### 4.2.6 SELECCIÓN DEL TAMAÑO DE LA PARTICIÓN: FRAGMENTACIÓN DE MEMORIA

El principal problema con particiones fijas está en encontrar un buen compromiso entre los tamaños de las particiones y la memoria requerida por los trabajos.

El funcionamiento general del sistema esta relacionado con el nivel de multiprogramación, que a su vez se ve directamente afectado por la eficacia con que se gestiona la memoria.

Existen dos tipos de desaprovechamiento de la memoria:

- Fragmentación interna. Consiste en aquella parte de la memoria que no se está usando pero que es interna a una partición asignada a una tarea. Si una tarea necesita  $m$  palabras de memoria para ejecutarse y se le asigna una partición de  $n$  palabras (con  $n > m$ ), está desaprovechando  $n-m$  palabras.
- Fragmentación externa. Ocurre cuando una partición disponible no se emplea por que es muy pequeña para cualquiera de las tareas que esperan.

### 4.3 GESTIÓN DE LA MEMORIA CON PARTICIONES VARIABLES

#### 4.3.1 PRINCIPIO DE OPERACIÓN

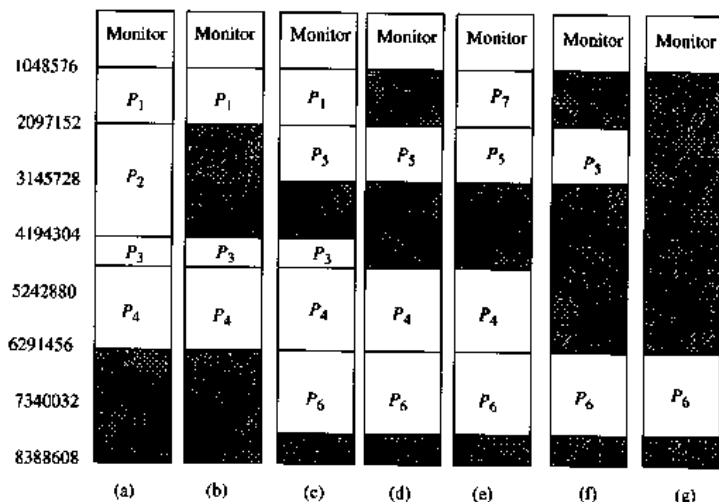
El S.O. mantiene una tabla que indica que partes de la memoria están disponibles y cuales están ocupadas.

Inicialmente toda la memoria está disponible y se considera como un gran bloque. Cuando llega una tarea que requiere memoria, se busca un bloque disponible suficientemente grande. Si se encuentra, se asigna sólo la cantidad que se necesita, manteniendo el resto disponible para satisfacer futuras demandas.

#### EJEMPLO

Se dispone de una memoria de 8Mb. El monitor residente del S.O. ocupa 1Mb.

Se supone que entran 4 procesos, que ocupan regiones consecutivas de memoria.



Cuando se termina o intercambia el proceso P2 se libera su región de memoria. Si a continuación entra el proceso P5(1Mb) puede ocupar parte de la región liberada por P2.

Cuando P3 libere la memoria que ocupa, este bloque se puede unir con el bloque de memoria libre contiguo.

En cualquier instante de tiempo hay libres un conjunto de bloques de memoria, de varios tamaños, repartidos a lo largo de la memoria:

- Cuando un proceso llega y necesita memoria se busca en este conjunto un bloque de memoria libre que sea suficientemente grande para dicho proceso. Si el bloque es muy grande se parte en dos; uno se asigna al proceso que llega y el otro se devuelve al conjunto de bloques libres.
- Cuando un proceso termina libera su partición de memoria, que se coloca en el conjunto de bloques libres. Si el nuevo bloque es adyacente a otro bloque libre, se unen para formar un bloque libre mayor. En este caso se comprueba si existen procesos esperando memoria y si este nuevo bloque recombinado y libre satisface las necesidades de cualquiera de ellos.

# TEMA 4

## GESTION DE LA MEMORIA

Mientras se puedan mover todos los procesos hacia la parte inferior (o superior) de la memoria es posible combinar todos los bloques libres en uno grande. Esta técnica se conoce como **compactación de la memoria**. No se realiza porque consume mucho tiempo de CPU.

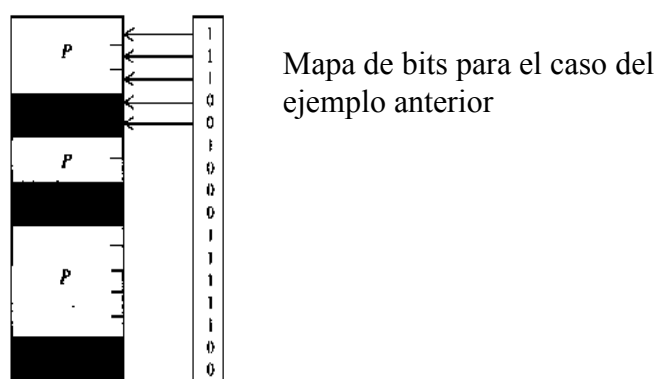
Cuando los segmentos de datos de algunos procesos pueden crecer, si hay alguna partición libre adyacente, se le puede asignar. Pero si el proceso que crece es adyacente a otro proceso, primero debe ser desplazado a una partición libre lo suficientemente grande, o se tendrán que intercambiar uno o mas procesos para crear una partición libre adyacente.

### 4.3.3 SISTEMAS DE REGISTRO DEL USO DE LA MEMORIA

#### MAPAS DE BITS

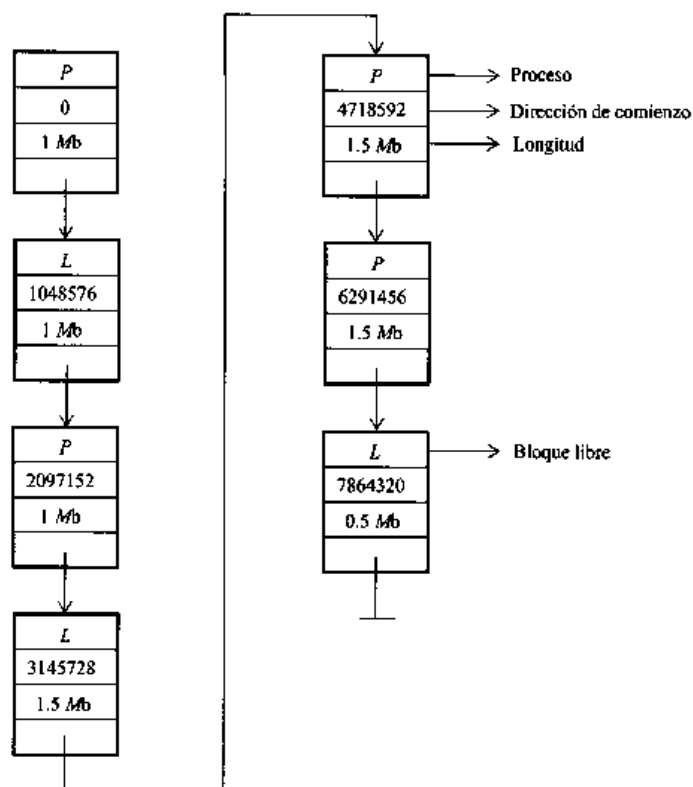
La memoria se divide en unidades de asignación. A cada unidad de asignación le corresponde un bit en el mapa de bits; con el valor cero indica que está libre y con 1 indica que está ocupado.

Inconveniente: Es un sistema de gestión de memoria lento.



#### LISTAS ENLAZADAS

Se mantiene un registro de la memoria, para las particiones libres y asignadas.



## TEMA 4

# GESTION DE LA MEMORIA

### Estrategias de asignación de memoria:

- 1) **Primero en ajustarse.** El gestor de memoria asigna al proceso entrante el primer bloque suficientemente grande que encuentra al recorrer la lista. Una variante es el **siguiente en ajustarse**, que asigna no la primera sino la siguiente partición que se ajusta (si existe).
- 2) **Mejor en ajustarse.** Asigna el bloque más pequeño que sea lo suficientemente grande. Produce restos mas pequeños y es más lento. También produce un mayor desperdicio de memoria, puesto que tiende a rellenar la memoria con particiones libres pequeñas sin utilidad.
- 3) **Peor en ajustarse.** Busca la partición más grande disponible. Produce los restos más grandes, pero se ha demostrado que no tiene un rendimiento superior a los anteriores.

Algunas variantes:

- Considerar dos listas independientes: una para los procesos y otra para las particiones libres.  
Ventaja: Se aumenta la velocidad de asignación de memoria.  
Inconveniente: Mayor complejidad y disminución de la velocidad al liberar memoria.
- Listas independientes para los procesos de tamaños que se solicitan con mas frecuencia. (ajuste rápido)

## SISTEMA DE LOS ASOCIADOS (Knuth, 1973)

Aprovecha el hecho de trabajar con direcciones binarias para hacer más rápida la fusión entre particiones libres adyacentes.

## EJEMPLO

Se mantiene una lista de las particiones libres cuyos tamaños son potencias de 2. Para una memoria de 8Mb, se tendrán 24 listas desde 1 byte hasta 8 Mb.

[illegible]

Al principio toda la memoria está libre. Todas las listas están vacías salvo la de 8Mb, que tiene un único elemento.

Al entrar el proceso P1, que solicita 600Kb, como las listas son potencias de 2, se redondea por exceso solicitándose 1024Kb(1Mb). Como Al principio sólo hay un bloque de 8Mb, éste se divide en dos bloques de 4Mb. Llamados socios; uno de ellos se divide en otros dos socios de 2Mb. Como sólo se necesita 1Mb, el bloque inferior se vuelve a dividir en dos bloques de 1Mb. Asignándole a P1 un bloque de 1Mb.

Quando llega P2 que necesita 300Kb, se solicitan 512Kb. No hay un bloque de 512Kb. en la lista correspondiente, luego se divide un bloque de 1Mb. en dos de 512Kb.

Cuando llega P3 de 700Kb, se solicitan 1024Kb, por la misma razón de antes se divide el bloque de 2Mb. para asignarle 1Mb.

A P4, que necesita 500Kb, se le puede asignar directamente el bloque de 512Kb.

Quando se libera memoria hay que ir fusionando, cuando se pueda, los socios, es decir aquellos bloques que proceden de la división de un mismo bloque.

**Ventaja:** Mayor velocidad para decidir la fusión de dos bloques.

Inconveniente: Fragmentación interna e externa.



## TEMA 4

# GESTION DE LA MEMORIA

### 4.3.4 ANÁLISIS DE LA GESTIÓN DE LA MEMORIA CON PARTICIONES VARIABLES

Los distintos algoritmos de gestión de la memoria producen una fragmentación externa de la memoria. La fusión de las particiones libres adyacentes sólo tiende a diferir el impacto de la fragmentación. La razón primaria de la fragmentación está en la diferencia de vida de los procesos, lo que implica que el patrón para devolver los bloques libres sea distinto del orden de asignación. Después de algún tiempo en operación, los sistemas con asignación dinámica de memoria tienden a un estado de equilibrio en el que la memoria desperdiciada por un determinado algoritmo de asignación se puede estimar y ser usada como un parámetro de comparación.

#### REGLA DEL CINCUENTA POR CIENTO

Para el algoritmo primero en ajustarse con fusión de los bloques libres adyacentes, Knuth demostró que si el promedio de procesos en memoria es  $n$ , el promedio de particiones libres es  $n/2$ . Esta regla tiene su origen en una asimetría fundamental entre las particiones libres y asignadas; cuando las regiones libres están adyacentes, se fusionan en una sola región. Los procesos adyacentes no se fusionan.

#### FRACCIÓN DE MEMORIA NO UTILIZADA

Para su cálculo se definen los siguientes parámetros:

- a)  **$f$** : Fracción de memoria ocupada por las particiones libres.
- b)  **$s$** : Tamaño medio de los  $n$  procesos.
- c)  **$ks$** : Tamaño medio de las particiones libres para algún valor de  $k > 0$ .

Si el tamaño total de la memoria es de  $m$  bytes, las  $n/2$  particiones libres ocuparán  $m - ns$  bytes. Es decir:

$$\frac{n \times k \times s}{2} = m - n \times s$$

Luego :

$$m = n \times s \times \left(1 + \frac{k}{2}\right)$$

La fracción de memoria libre es el número de particiones libres,  $n/2$ , por el tamaño medio de las particiones libres,  $ks$ , dividido por el tamaño de toda la memoria  $m$ :

$$f = \frac{(n \times k \times s)/2}{m} = \frac{(n \times k \times s)/2}{n \times s \times \left(1 + \frac{k}{2}\right)} = \frac{k}{k + 2}$$

Si las particiones libres ocupan la mitad de memoria que los procesos,  $k = 1/2$ , el 20% de la memoria queda libre. Para  $k = 1/4$ , el 11%.

# TEMA 4

## GESTION DE LA MEMORIA

### 4.4 PAGINACIÓN

Uno de los principales inconvenientes que se ha visto en los anteriores sistemas de gestión de la memoria es la fragmentación externa. Este problema tiene dos soluciones generales:

- 1) Compactación de memoria, que no se suele utilizar por su coste de tiempo.
- 2) Paginación, que permite que la memoria de un programa no sea contigua, de forma que siempre que se disponga de espacio, aunque éste no sea adyacente, se pueda asignar a un programa.

#### 4.4.1 PRINCIPIO DE OPERACIÓN

La memoria se divide en un número de bloques de tamaños fijos, denominados **marcos de página**. El espacio de direcciones virtuales de un proceso (todas las posibles direcciones que puede generar el proceso) también se divide en bloques de tamaño fijo, llamados **páginas**, del mismo tamaño que los marcos de página.

La asignación de memoria consiste en encontrar un número suficiente de marcos de página sin usar para cargar las páginas de los procesos solicitadas.

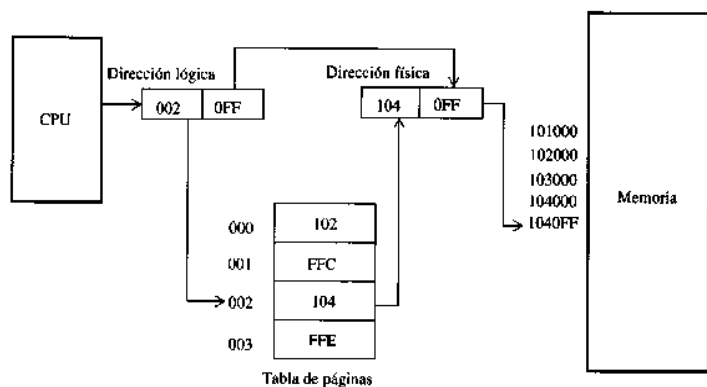
Mediante un mecanismo de traducción de direcciones, se asignan las páginas virtuales a su contrapartida física. Como cada página se asigna por separado, no es necesario que los marcos de página asignados a un único proceso estén situados en zonas contiguas de la memoria física.

Para hacer esta asignación se usa una **tabla de páginas**, que se construye cuando se carga el proceso y que contiene tantas entradas como páginas tenga éste.

Cada dirección lógica se divide en dos partes:

- 1) El número de página que se emplea como un índice en la tabla de páginas.
- 2) Un desplazamiento dentro de la página.

La dirección física se construye sustituyendo en el primer campo en el que se divide la dirección, el número de página por el marco de página correspondiente.



La dirección lógica 0020FF se descompone en dos campos; 002 que es el número de página y 0FF que es el desplazamiento relativo dentro de esta página.

De la tabla de páginas se comprueba que a la página 002 le corresponde el marco de página 104.

El S.O. mantiene el estado de cada marco de página de la memoria mediante un mapa de la memoria estructurado como una tabla estática.

Esta tabla tiene un número fijo de entradas, que es igual al número de marcos de página, que viene dado como la relación entre la capacidad de la memoria física instalada y el tamaño de cada página. Para cada entrada se indica si el marco de página correspondiente está asignado o libre.

No se tiene fragmentación externa, se tiene algo de fragmentación interna, media página por proceso. Si las páginas son pequeñas, ésta sería menor, pero daría lugar a tablas de páginas excesivamente grandes.

#### 4.4.2 IMPLEMENTACIÓN DE LAS TABLAS DE PÁGINAS.

Las tablas de páginas plantean dos cuestiones importantes:

- 1) El tamaño de tabla de páginas, que puede ser demasiado grande.
- 2) El tiempo de asignación, que debe ser rápido.

## TEMA 4

# GESTION DE LA MEMORIA

La realización más sencilla es tener una sola tabla de páginas estructurada como un conjunto de registros dedicados. El distribuidor de la CPU cargará estos registros cada vez que cargue los otros registros del programa.

Éste es un método directo que sólo usa una referencia a la memoria durante la asociación.

Desventajas:

- Coste de la circuitería asociada cuando las tablas de páginas son grandes.
- Tener que cargar la tabla de páginas en cada cambio de contexto.

Como la tabla de páginas puede estar totalmente dentro de la memoria principal, el hardware específico puede limitarse a un registro que apunte al comienzo de la tabla. Un cambio de contexto supone cargar un solo registro.

Desventaja:

- Se requiere más de una referencia a la memoria para leer las entradas de la tabla de páginas durante la ejecución de cada instrucción.

Solución:

- Disponer de un pequeño dispositivo especial llamado memoria asociativa. Su funcionamiento se basa en el hecho de que la mayoría de los programas tienden a hacer un gran número de referencias a un número pequeño de páginas. Sólo se leen unos pocos datos de la tabla de páginas, y el resto se usa muy poco.
- La memoria asociativa consta de un número pequeño de entradas (no más de 32). Cada una tiene la siguiente la siguiente información:
  - Número de página.
  - Marco físico donde se localiza.
  - Código de protección.
  - Bit de modificación.
  - Bit que indica si esta siendo utilizada o no.

Número de página	Marco de página	Código de protección	Bit de modificación
004	104	LE	0
012	FFC	LX	1
003	203	EX	0
024	FFE	LX	1
010	021	EX	0
028	F00	LE	0

Cuando se presenta una dirección lógica, en primer lugar se verifica que el número de página lógica se encuentra en la memoria asociativa, comparando en paralelo el número de página dado en la dirección virtual con la parte de número de página de los registros de la memoria asociativa. Si coincide con alguno y el acceso no viola los bits de protección, el marco de página se toma directamente de la memoria asociativa. Si se viola la protección se genera un fallo de página. Si el número de página no está en la memoria asociativa, se tiene que recurrir a la tabla de páginas. La información obtenida de la tabla de páginas para esta página puede reemplazar a un registro de la memoria asociativa.

El porcentaje de veces que un número de página se encuentra en la memoria asociativa (razón de encuentros) está relacionado con el número de registros de la misma.

## TEMA 4

# GESTION DE LA MEMORIA

Si el sistema es multiprogramado, cada proceso tiene su propia tabla de páginas. Para que al ejecutarse un nuevo proceso no utilice los marcos de página de otro proceso se suelen adoptar dos soluciones:

- 1) Proporcionar una instrucción máquina que invalide la memoria asociativa (que borre todos los bits de validación).
- 2) Realizar la memoria asociativa con un campo de identificación del proceso y añadir un nuevo registro con el identificador del proceso activo.

### 4.4.3 COMPARTICIÓN DE PÁGINAS Y PROTECCIÓN.

Una de las ventajas de la paginación es la posibilidad de compartir código, lo cual es interesante sobre todo en entornos de tiempo compartido.

Para ser compatible, el código debe de ser reentrante, es decir, no automodificable, de forma que no cambia nunca durante la ejecución.

Dos o más procesos pueden ejecutar el mismo código a la vez, manteniendo copia de sus registros y de sus datos, ya que estos cambian de un proceso a otro.

En un sistema con paginación, la protección de la memoria se realiza mediante bits de protección asociados a cada una de las páginas. Generalmente, estos bits se encuentran en la tabla de página y definen la página como de lectura/escritura o de sólo lectura, de forma que al acceder a la página, mientras se calcula la dirección física, se verifica el valor de este bit para no permitir escribir en una página de sólo lectura.

Este método se puede ampliar con varios bits, uno para cada clase de acceso, por ejemplo, de sólo lectura, de lectura/escritura, de sólo ejecución.

## 4.5 SEGMENTACIÓN

En este esquema el espacio de direcciones lógicas es un conjunto de segmentos, con diferentes nombres y tamaños. Las direcciones especifican el nombre de segmento y el desplazamiento.

Un segmento es un objeto lógico, conocido por el programador y que lo utiliza como tal. Un segmento puede tener una determinada estructura de datos o un procedimiento o módulo, pero, normalmente, no una mezcla de varios.

### 4.5.1 PRINCIPIO DE OPERACIÓN

El programa de usuario se compila y el compilador construye automáticamente los segmentos de acuerdo a la estructura del programa.

El cargador tomará todos estos segmentos y les asignará un número de segmento. Por propósito de reubicación, cada uno de los segmentos se compila comenzando por su propia dirección lógica 0. Las direcciones lógicas constarán de un número de segmento y un desplazamiento dentro de él.

La conversión de estas direcciones lógicas bidimensionales en su dirección física equivalente unidimensional se realiza mediante una **tabla de segmentos**, en la que se mantienen registrados los descriptores de cada segmento: la base, correspondiente a la dirección física en que comienza el segmento, obtenida durante la creación de la partición, y el tamaño del mismo.

El número de segmento se emplea como un índice dentro de la tabla de segmentos. La dirección física se obtiene añadiendo el desplazamiento a la base del segmento. Si este desplazamiento superase el tamaño máximo del segmento registrado en la tabla, se producirá un error de direccionamiento.

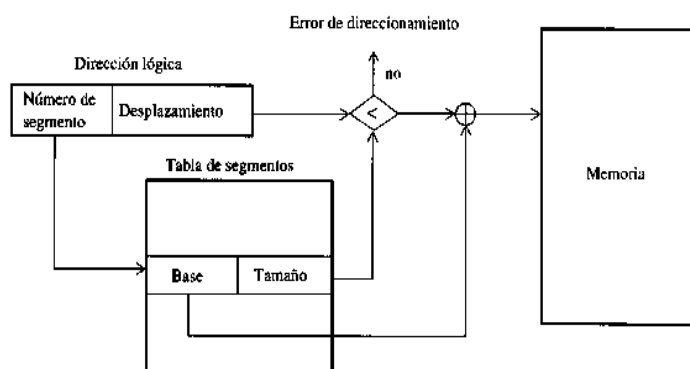
Una tabla de segmentos que se mantiene en registros especiales puede ser referenciada muy rápidamente (la suma a la base y la comparación pueden hacerse simultáneamente). Este método puede llevarse a cabo si el número de segmentos no es grande.

Cuando el número de segmentos es grande, la tabla de segmentos se deja en memoria. En este caso, un **registro base de la tabla de segmentos** apunta a la tabla de segmentos.

# TEMA 4

## GESTION DE LA MEMORIA

Como el número de segmentos usados por un programa puede variar, también se emplea un **registro de la longitud de la tabla de segmentos**.



Uso de la tabla de segmentos.

Para una dirección lógica:

- 1) Se comprueba que el número de segmento es menor que el registro de la longitud de la tabla de segmentos.
- 2) Se suma el número de segmento al valor del registro base de la tabla de segmentos para obtener la dirección de memoria para la entrada de la tabla de segmentos.
- 3) Esta entrada se lee de memoria, e comprueba que el desplazamiento no rebasa la longitud del segmento y se calcula la dirección física como la suma del registro base de la tabla de segmentos y del desplazamiento.

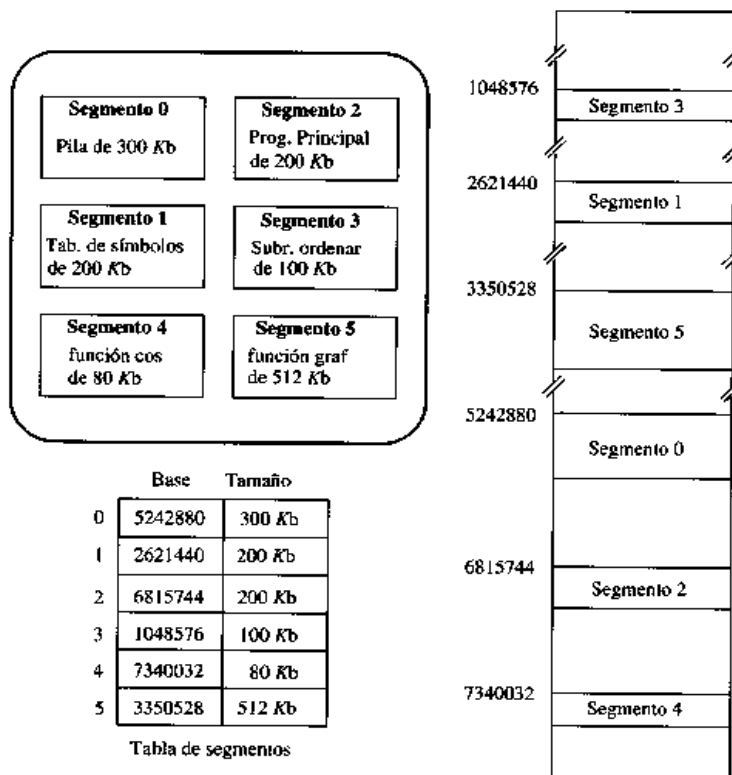
Inconveniente:

- Requiere dos referencias a memoria. (Al igual que la paginación)

Solución:

- Disponer de una memoria asociativa para retener las entradas de la tabla de segmentos usadas más recientemente.

La segmentación NO produce fragmentación interna, pero SI externa.



El espacio de direcciones lógicas está compuesto por seis segmentos situados en memoria.

La tabla de segmentos tiene una entrada para cada segmento, indicándose la dirección de comienzo del segmento en la memoria física (base) y la longitud del mismo (tamaño).

El segmento 3 empieza en la dirección 1048576 y tiene un tamaño de 100Kb; si se hace referencia a la dirección 115000 del segmento 3 se producirá un error de direccionamiento, puesto que esta dirección supera el tamaño del segmento.

## TEMA 4

# GESTION DE LA MEMORIA

### 4.5.2 PROTECCIÓN

Es posible asociar la protección a los segmentos. Así, los segmentos de código se pueden definir de sólo lectura y los datos de lectura/escritura.

También facilita la compartición de código o datos entre varios procesos

Los segmentos son compartidos cuando la entrada a la tabla de segmentos de dos procesos apuntan a la misma posición física.

### 4.5.3 SISTEMAS COMBINADOS DE PAGINACIÓN-SEGMENTACIÓN

Características comunes y diferencias:

- El programador no necesita saber que se está utilizando la paginación pero si la segmentación.
- La paginación sigue un esquema lineal, con único espacio de direcciones. La segmentación proporciona un espacio bidimensional con muchos espacios de direcciones.
- En la paginación no se distingue el código de los datos. En la segmentación sí, pudiéndose proteger de forma independiente.
- La segmentación facilita la compartición de código y datos, así como el uso de estructuras de datos de tamaños fluctuantes.
- Tienen en común que el espacio total de direcciones puede exceder el tamaño de la memoria física, haciendo uso de la memoria virtual.

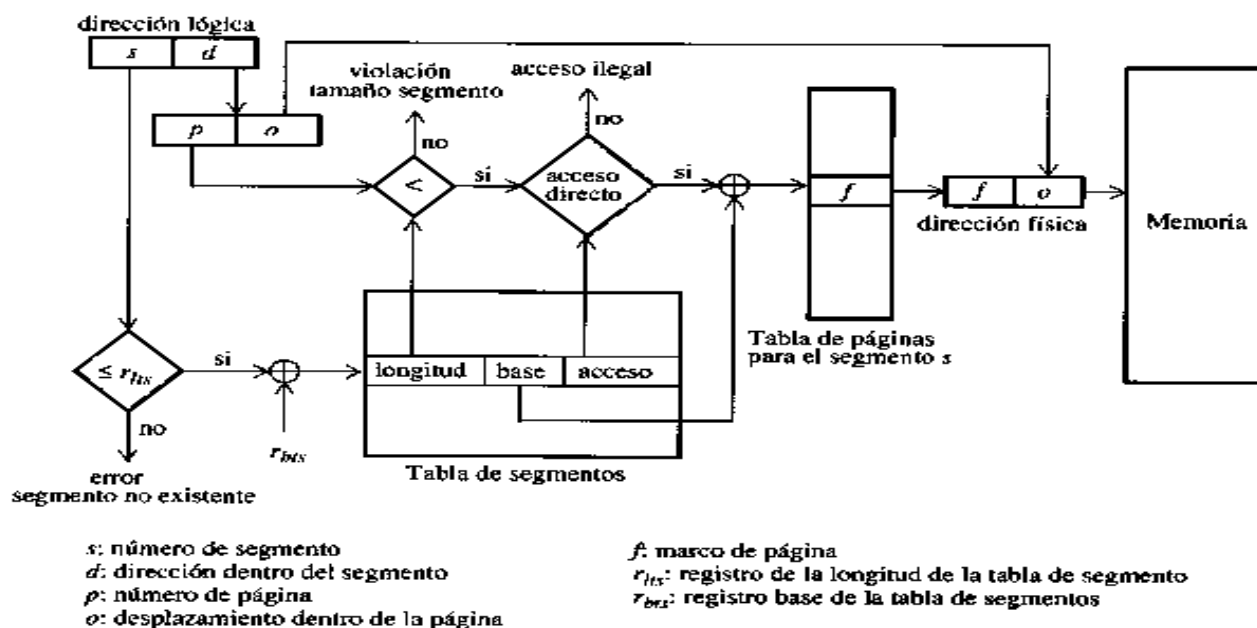
Hay sistemas que combinan los dos métodos.

Una técnica muy extendida es usar la segmentación desde el punto de vista del usuario, pero dividiendo cada segmento en páginas de tamaño fijo. Es adecuada en el caso de que los segmentos sean muy grandes y resulte un grave inconveniente mantenerlos en su totalidad dentro de la memoria.

Se elimina la fragmentación externa y se hace más fácil el problema de la ubicación.

La entrada a la tabla de segmentos no contiene la dirección base del segmento, sino la dirección base de una tabla de páginas para este segmento. El desplazamiento del segmento se divide en un número de página y un desplazamiento dentro de ella. El número de página se usa como índice en una tabla de páginas para obtener el número del marco de página. Este último se combina con el desplazamiento de página para obtener la dirección física.

Cada segmento tiene su propia tabla de páginas, cuyo tamaño es función de la longitud del segmento. En promedio, se tiene una fragmentación interna de media página por segmento. El espacio de tablas aumenta. Se puede incluir un bit de presente/ausente en la tabla de segmentos.



## TEMA 4

# GESTION DE LA MEMORIA

### 4.6 MEMORIA VIRTUAL

Consiste en un sistema de gestión de la memoria que permite que el espacio de direcciones virtuales de un proceso activo sea mayor que la memoria física disponible. El S.O. mantiene en la memoria principal aquella parte del proceso que se utiliza en cada momento, el resto permanece en el disco.

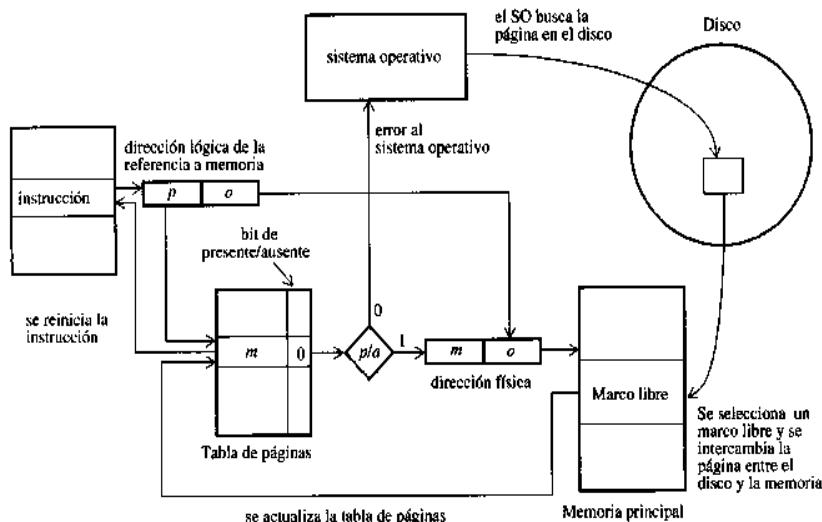
Es normal desarrollarla como una extensión de la gestión de la memoria.

#### 4.6.1 PRINCIPIO DE OPERACIÓN.

Uno de los sistemas más comunes de memoria virtual es el denominado **demanda de página**, que es similar a un sistema de paginación con intercambio. En su forma más pura, los procesos se inician sin ninguna página en memoria; cuando se intenta ejecutar la primera instrucción se produce un fallo de página, que provocará que el S.O. tenga que cargar en la memoria principal la página que contiene esta instrucción; después de un cierto tiempo el proceso tiene la mayoría de las páginas que necesita y la ejecución tiende a desarrollarse con un número relativamente pequeño de fallos páginas.

Cuando se produce un fallo de página el procedimiento a seguir:

- 1) Se verifica si la dirección es válida. Si fuera inválida se enviaría una señal al proceso o se abortaría.
- 2) Si es una referencia válida, pero aún no se ha traído esa página a la memoria principal, el S.O. detecta un fallo de página y determina la página virtual requerida para traerla. En este caso la instrucción queda interrumpida y se guarda el estado (registros, código, contador de programa) del proceso, para poder continuarlo en el mismo lugar y estado.
- 3) Se selecciona un marco libre. Si no existiese, se tendría que ejecutar un algoritmo de sustitución de página.
- 4) Si el marco de página está ocupado, el planificador transfiere la página al disco y se produce un cambio de contexto. Se suspende el proceso causante del fallo y se permite la ejecución de otro hasta que termine la transferencia al disco. El marco queda marcado como ocupado.
- 5) Cuando el marco queda limpio, el S.O. examina la dirección en el disco donde se encuentra la página necesaria y planifica una operación de lectura de la misma.
- 6) Cuando se completa la lectura del disco, la tabla de páginas se actualiza.
- 7) La instrucción que produjo el fallo regresa al estado de comienzo y se planifica su ejecución.



La circuitería para soportar demanda de página consiste en un disco y una tabla de páginas con la posibilidad de marcar una entrada como ausente mediante un bit de presente/ausente o un valor especial de bits de protección. Además de la programación necesaria.

Restricciones sobre la arquitectura: posibilidad de continuar cualquier instrucción después de un fallo de página. Si el fallo de página ocurre cuando se está buscando o trayendo un operando de la instrucción, cuando se tenga que continuar el proceso se debe traer otra vez la instrucción, decodificarla y volver a buscar y traer el operando. No todas las máquinas permiten recuperar el estado cuando se interrumpe en mitad de una instrucción.

# TEMA 4

## GESTION DE LA MEMORIA

### 4.6.2 CONCEPTOS DE MEMORIA VIRTUAL

Tiempo promedio de acceso ( $t_{pa}$ ):

$$t_{pa} = (1 - p) \times a_m + p \times f_p$$

donde :

$a_m$  -> Tiempo medio de acceso a memoria

$p$  -> Probabilidad de que ocurra un fallo de página ( $0 \leq p \leq 1$ )

$f_p$  -> Tiempo que lleva resolver un fallo de página.

Si se quiere una degradación en el rendimiento del computador inferior al 10%, es preciso que  $p < 0,00001$ , es decir, la probabilidad de que ocurra un fallo de página debe ser menor al 0,001%.

Muchos procesos no usan todas sus páginas, por tanto no hay que traerlas todas, pudiéndose ejecutar más procesos. Esto puede provocar una *sobreasignación de memoria*. En este caso, se necesita un sistema de sustitución de páginas que permita mantener un alto nivel de multiprogramación, de forma que el promedio de uso de la memoria sea cercano a la memoria física disponible.

La realización de la memoria virtual mediante paginación supone mantener una tabla de páginas por cada uno de los procesos activos. Estas tablas de páginas pueden ser mucho más grandes en un sistema virtual, ya que el espacio de direcciones virtuales de un proceso puede exceder de la memoria real:

En la gestión de memoria virtual se deben tener políticas de :

- Asignación de memoria real a cada uno de los procesos activos.
- Búsqueda y lectura en el disco de los correspondientes elementos.
- Sustitución de elementos en memoria principal por los nuevos elementos que se traen del disco.
- Ubicación de los nuevos elementos.

La política de ubicación sigue las mismas reglas que la paginación y la segmentación.

## 4.7 POLÍTICAS DE SUSTITUCIÓN DE PÁGINAS

### 4.7.1 ALGORITMO DE SUSTITUCIÓN FIFO (Primero en entrar, primero en salir)

Se elige la página que más tiempo lleva en memoria. Se puede crear una cola según el orden de entrada. Cuando hay que sustituir una página, se elige la primera de la cola, y la que se trae se inserta al final de la cola.

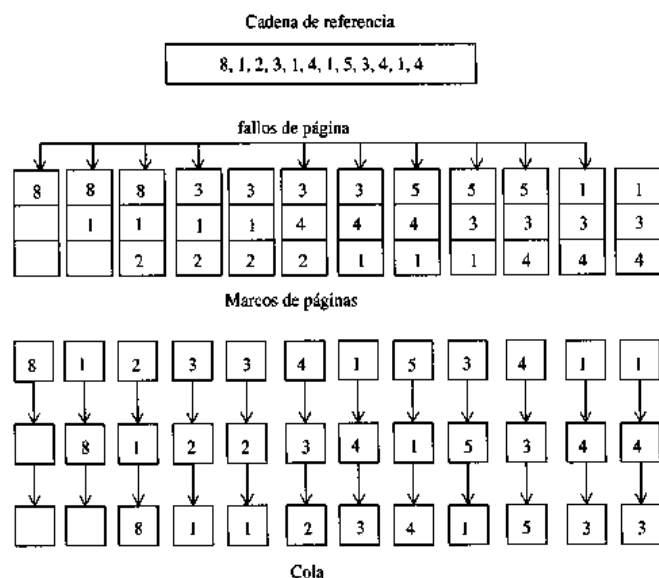
Inconvenientes :

- Aunque una página se use con mucha frecuencia, podría ser seleccionada para ser sustituida.
- Sufre de la **anomalía de Belady**: en algunos casos, al aumentar el número de marcos de página, pueden aumentar los fallos de página.



# TEMA 4

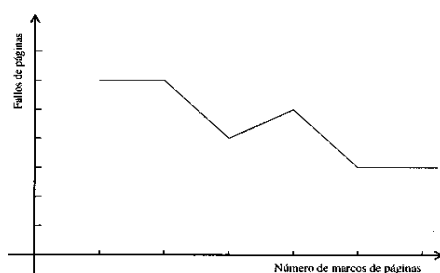
## GESTION DE LA MEMORIA



Se supone que sólo se tienen tres marcos de página. Si inicialmente la memoria está vacía, las primeras tres referencias causarán fallos de página que provocarán que se traigan las tres primeras páginas (8,1,2) a la memoria.

Para la siguiente referencia no se tiene espacio y se tendrá que elegir una página para sustituir, se elige la 8 que fue la primera que se cargó en memoria.

Cuando se accede a la 1 ésta ya está en la memoria y no hay que hacer nada, pero al referenciar a la página 4 se sustituye la 1 que es la que está en la cabeza de la cola.



Anomalía de Belady.

Para la cadena de referencia :

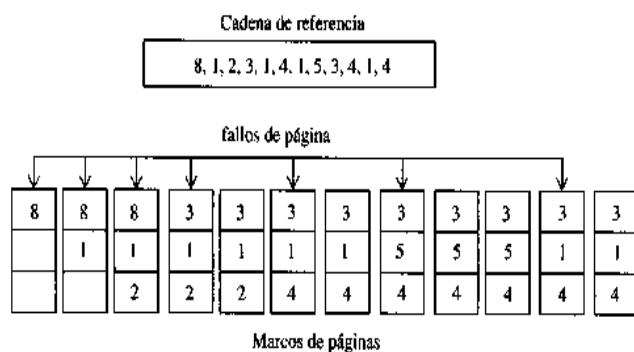
5,6,7,8,5,6,9,5,6,7,8,9

Para cuatro marcos de páginas ocurren más fallos de páginas que para tres.

### 4.7.2. ALGORITMO DE SUSTITUCIÓN ÓPTIMO

Sustituir aquella página que tardará más en volverse a utilizar.

Es irrealizable, pero sirve como nivel de medida para los demás algoritmos de sustitución de páginas, comparándolo mediante simulación o en una segunda ejecución.



Sustituciones de las páginas según el algoritmo óptimo. Se observa que frente a los diez fallos de página del algoritmo FIFO, el algoritmo óptimo presenta siete fallos de páginas. Lo que supone que para la cadena de referencia del ejemplo el algoritmo FIFO es un 43% peor que el óptimo.

## TEMA 4

# GESTION DE LA MEMORIA

### 4.7.3 ALGORITMO DE SUSTITUCIÓN LRU (la menos usada últimamente)

Es una mejora respecto al FIFO. Necesita un considerable apoyo de recursos de la máquina.

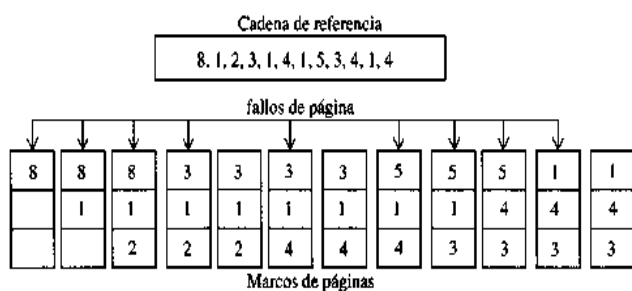
Dos posibles realizaciones :

- Mediante una pila.** Que mantiene los números de las páginas. Cada vez que una página se referencia, su número se elimina de la pila y se coloca en la cumbre de la misma. En el fondo estará la página que hace más tiempo que se usó. Se puede diseñar una lista doblemente enlazada. La eliminación de una página y su colocación en la cumbre de la pila supone tener que modificar seis punteros.
- Con registros contadores.** Cada entrada en la tabla de páginas tiene asociado un registro y en la CPU hay un contador que se incrementa cada vez que se hace referencia a memoria. Cada vez que se hace referencia a una página, se copia el registro contador en el registro. Cuando hay que hacer una sustitución, se busca el registro con el valor más pequeño.

Ambas realizaciones necesitan de una circuitería especial de la que no siempre se dispone.

Algunos sistemas proporcionan un bit de referencia por cada una de las páginas. Inicialmente el S.O. pone a cero estos bits. Cuando se referencia una página, se pone su bit a uno, de forma que se sabe qué páginas han sido referenciadas, pero no se conoce el orden en que han sido usadas.

Se puede tener una tabla con registros de ocho bits por cada una de las páginas. Periódicamente el S.O. desplazará a la derecha los bits.



Se han producido nueve fallos de página, los seis primeros son los mismos que para el algoritmo óptimo.

Aunque presenta más fallos que el algoritmo óptimo es una mejora con respecto al FIFO.

### 4.7.4 ALGORITMO DE SUSTITUCIÓN DE LA SEGUNDA OPORTUNIDAD

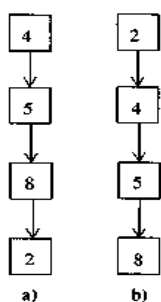
Modificación del algoritmo FIFO. Consiste en examinar el bit de referencia de las páginas que se seleccionan para sustituir. Si el bit de referencia es 0, la página no se ha utilizado, además de ser antigua. Si es 1, entonces se pone a cero y la página se coloca al final de la cola.

Si todas la páginas tienen alguna referencia, el algoritmo degenera en un FIFO puro.

Puede verse como una cola circular, con un puntero que indica cual es la página que se sustituirá a continuación.

Cuando se necesita un marco de página, el puntero avanza hasta que se encuentra una página con un bit de referencia que vale 0, poniendo a 0 los bits de referencia que va pasando.

También se conoce como algoritmo del reloj.



Si ocurre un fallo de página, según el algoritmo FIFO la página que se sustituye es la 2.( ver fig. a.). Si esta página tiene el bit de referencia a 0 se sustituiría sin mas, pero si el bit de referencia es 1, está página se coloca al final de la cola y su bit de referencia se pone a 0 (ver fig. b)

## TEMA 4

# GESTION DE LA MEMORIA

### 4.7.5 OTROS ALGORITMOS DE SUSTITUCIÓN

Aproximaciones del LRU :

- a) Además del bit de referencia, se considera un **bit de modificación** que se activa cuando se escribe una palabra en la página.  
Inicialmente el S.O. asigna un valor 0 a los dos bits. De forma periódica, el bit de referencia se pone a 0 para distinguir las páginas que no tienen referencias recientes de las que sí.  
Se forman así cuatro categorías de páginas según los valores de los bits (00,01,10,11).  
Se sustituye una página de la clase más baja que no esté vacía. Si hay varias, se puede usar FIFO o elegir aleatoriamente.
- b) **Algoritmo de uso frecuente (NFU)**. Necesita una variable de tipo contador asociada a cada página, con valor inicial 0. Periódicamente el S.O. suma el bit de referencia al contador. Se elige para sustituir la página con el valor mínimo del contador.  
Inconveniente : Páginas que fueron usadas con mucha frecuencia hace tiempo, pero que ya no se usan, mantienen un valor alto, mientras que páginas que se han empezado a usar recientemente tienen un valor bajo.  
Solución: Desplazar los bits del contador un lugar antes de sumar el bit de referencia.

Se llaman **algoritmos de pila** a aquellos que nunca presentan la anomalía de Belady, para los que se puede demostrar que el conjunto de páginas en memoria para  $n$  marcos es un subconjunto del conjunto de páginas que se tendrían con  $n+1$  marcos.

LRU y el algoritmo óptimo no presentan la anomalía de Belady.

# TEMA 4

## GESTION DE LA MEMORIA

### 4.8 POLÍTICAS DE ASIGNACIÓN

#### DEFINICIÓN

Cuando se dan más marcos de página a un proceso, se debe reducir la frecuencia de fallos de página, pero esto supone tener un menor número de procesos activos.

Por el contrario, si se asignan muy pocos marcos de página por proceso, se aumenta la multiprogramación, pero también la frecuencia de fallos de página.

El número mínimo de marcos por proceso viene fijado por la propia arquitectura del computador, mientras que el número máximo es definido por la cantidad de memoria física disponible.

La elección del número de marcos por proceso se hará entre estos dos límites.

La sustitución de páginas puede ser :

- Local:** Para cada proceso se seleccionan sólo los marcos asignados a él. El número de marcos de un proceso no cambia y depende sólo del comportamiento del mismo.
- Global:** Permite que para un proceso se seleccione un marco para sustitución del conjunto de todos los marcos. El conjunto de páginas en memoria de un proceso puede variar y depende del comportamiento del mismo y del de los otros procesos.

Proc. A pag 0	12	Proc. A pag 0	Proc. A pag 0
Proc. A pag 1	9	Proc. A pag 1	Proc. A pag 1
Proc. A pag 2	11	Proc. A pag 2	Proc. A pag 2
Proc. A pag 3	5	Proc. A pag 3	Proc. A pag 3
Proc. A pag 4	2	Proc. A pag 6	Proc. A pag 4
Proc. A pag 5	6	Proc. A pag 5	Proc. A pag 5
Proc. B pag 0	7	Proc. B pag 0	Proc. B pag 0
Proc. B pag 1	4	Proc. B pag 1	Proc. B pag 1
Proc. B pag 2	1	Proc. B pag 2	Proc. A pag 6
Proc. B pag 3	10	Proc. B pag 3	Proc. B pag 3
Proc. B pag 4	3	Proc. B pag 4	Proc. B pag 4
(a)		(b)	(c)

La fig. A muestra la situación inicial.

Si al entrar la página 6 del proceso A se usa una estrategia de sustitución local, se tendría que sustituir la página 4 del proceso A, pero si se usa una estrategia global, se podría elegir entre las páginas del cualquier proceso, en este caso se sustituiría la página 2 del proceso B.

El problema para la sustitución global es que los programas no pueden controlar su propia razón de fallos de páginas. El conjunto de páginas en memoria de un proceso no sólo depende de la conducta particular de este proceso, sino también de la de otros procesos.

Para políticas locales, el conjunto de páginas de un proceso depende sólo del comportamiento del mismo.

#### 4.8.1 EL MODELO DEL CONJUNTO DE TRABAJO

Con **algoritmos de sustitución global**, al aumentar el grado de multiprogramación, la utilización del procesador aumenta mientras el grado de multiprogramación se mantenga por debajo de un cierto nivel, que depende del tamaño de la memoria disponible. Si se excede ese límite, se provoca un marcado aumento en el intercambio de páginas, acompañado por una disminución en la utilización del procesador (catástrofe o trashing).

Con algoritmos de sustitución local, si un proceso comienza un fenómeno de catástrofe, no puede coger marcos de otro proceso y extender así la catástrofe. Sin embargo, si varios procesos están ya en catástrofe, permanecerán más tiempo en la cola para los dispositivos de páginas, aumentando así el tiempo promedio de servicio para un fallo de página, con lo que el tiempo de acceso efectivo aumenta para un proceso aunque no este en catástrofe.

## TEMA 4

# GESTION DE LA MEMORIA

El conjunto de páginas utilizadas en un momento determinado por un proceso se denomina **conjunto de trabajo**.

Para evitar la catástrofe, el grado de multiprogramación no debe ser mayor que el que permite que los conjuntos de trabajo para todos los procesos se puedan tener simultáneamente en memoria.

Para determinar que páginas constituyen el conjunto de trabajo de un proceso, se usa el **principio de localidad**: “las referencias a los programas tienden a agruparse en pequeñas zonas del espacio de direcciones, y estas localizaciones tienden a cambiar sólo intermitentemente”

Conjunto de trabajo en un instante  $t$  :

$$w(t,h)=\{\text{pág. } i \mid \text{pág. } i \in N \text{ y la pág. } i \text{ aparece en las últimas } h \text{ referencias}\}$$

El conjunto de trabajo es el conjunto de páginas que han sido referenciadas recientemente. Se puede obtener un valor de  $h$  ( $h_0$ ) a partir del cual aumentar  $h$  no significa aumentar apreciablemente el tamaño del conjunto de trabajo.

Regla de sustitución y asignación :

Ejecutar un proceso solamente si el conjunto de trabajo está por completo en memoria principal, y no eliminar una página que es parte del conjunto de trabajo de un proceso.

## 4.9 ASPECTOS DE DISEÑO PARA LOS SISTEMAS DE PAGINACIÓN

### 4.9.1 TAMAÑO DE PÁGINA

Factores a tener en cuenta en la elección del tamaño de página :

- Un tamaño de página pequeño supone un menor desperdicio de memoria, pero implica un tamaño grande de tablas de páginas. Un tamaño de página grande da lugar a tablas de páginas más pequeñas, pero provoca un mayor desperdicio de memoria.
- El tiempo necesario para leer y escribir una página está compuesto por el **periodo de latencia** (tiempo de búsqueda y rotación del disco, que no depende del tamaño de la página) y el **tiempo de transferencia** ( que es proporcional al tamaño de la página). Como el periodo de latencia es el mismo en ambos casos, se consiguen menores tiempos de lectura/escritura con páginas grandes.
- Otros factores: relación entre el tamaño de la página y el de los sectores del disco.

### 4.9.2 PREPAGINACIÓN

Cuando un proceso se arranca inicialmente o después de haber sido suspendido, si usa un sistema puro de demanda de página, se producirán tantos fallos de páginas como páginas necesita para ejecutarse.

Solución: Traer de una sola vez a memoria todas las paginas que se necesitan, **prepaginación**.

Para un sistema que usa el modelo de conjunto de trabajo, cuando se arranca de nuevo se traerían todas las páginas de su conjunto de trabajo. Pero puede ocurrir que muchas de ellas no sean utilizadas.

Si se supone que  $p$  páginas van a ser prepaginadas y que sólo se van a usar una fracción  $\alpha$  ( $0 \leq \alpha \leq 1$ ) de ellas, la cuestión es si el coste de prepaginar  $(1-\alpha)p$  páginas no usadas es menor que el coste de evitar  $\alpha p$  fallos de páginas. Si  $\alpha$  es cercano a 1, la prepaginación es rentable; si es cercano a 0 no.

## **TEMA 4**

# **GESTION DE LA MEMORIA**

### **4.9.3 FRECUENCIA DE FALLOS DE PÁGINAS**

El modelo del conjunto de trabajo es útil para realizar la prepaginación, pero no es el más adecuado para manejar el control de la catástrofe. Este se puede efectuar mejor mediante estrategias que consideren la frecuencia de los fallos de página.

La razón de fallos de página frente al número de marcos asignados al proceso tiene un comportamiento decreciente en todos los algoritmos de sustitución que no presentan la anomalía de Belady.

Se pueden establecer límites entre los cuales se desea mantener el porcentaje de fallos de páginas

Si en algún momento existe un número elevado de procesos en memoria que no permiten mantener la razón de fallos de página por debajo del límite superior, se retira un proceso y se reparten sus marcos de página entre los otros.