SISTEMAS INFORMÁTICOS UD-3

SISTEMAS OPERATIVOS. GESTION DE ARCHIVOS Y ALMACENAMIENTO

GESTIÓN DE MEMORIA

ÍNDICE

[GESTIÓN DE MEMORIA 3](#_Toc120121123)

[A. ASIGNACIÓN DE MEMORIA 3](#_Toc120121124)

[1. SISTEMAS MONOPROGRAMADOS 3](#_Toc120121125)

[2. SISTEMAS MULTIPROGRAMADOS 3](#_Toc120121126)

[B. PAGINACIÓN 5](#_Toc120121127)

[C. TLB: TRANSLATION LOOKASIDE BUFFER 7](#_Toc120121128)

[D. MEMORIA VIRTUAL 8](#_Toc120121129)

[E. SEGMENTACIÓN 9](#_Toc120121130)

# GESTIÓN DE MEMORIA

## ASIGNACIÓN DE MEMORIA

1. SISTEMAS MONOPROGRAMADOS

Originalmente en los sistemas operativos monoprogramados (por ejemplo, MS-DOS) la memoria se repartía de forma que una parte de la memoria estaba destinada al Sistema Operativo y todo el resto de memoria estaba destinada para el proceso en ejecución.



1. SISTEMAS MULTIPROGRAMADOS

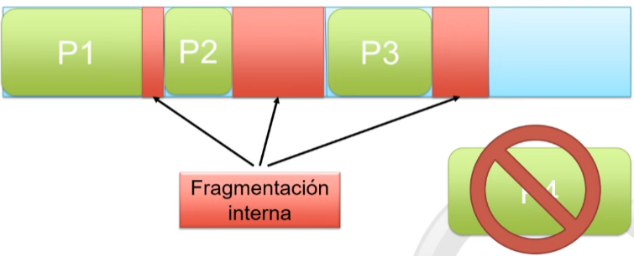
En los sistemas operativos actuales que realizan varias tareas (procesos) al mismo tiempo necesitamos utiliza la memoria para varios procesos que se estén ejecutando al mismo tiempo.

Para ello tenemos varias opciones:

* **Particiones estáticas de tamaño fijo**: Se parte la memoria en n partes llamadas particiones. Estas particiones son estáticas (es decir, no pueden variar su tamaño porque está prefijado) y son de un tamaño fijo (es decir, todas las particiones son del mismo tamaño).

A cada cubículo le vamos asignando un proceso, es decir, nos llega un proceso P1 que se colocará en el primer espacio de memoria, un proceso P2 que se colocará en el segundo…

En este modelo de particiones el tamaño de memoria que ocupa cada proceso es indiferente, cada proceso va a una partición. Salvo si llega un proceso que sea demasiado grande (como es el caso del proceso P4) que obviamente no podría entrar en ninguna de las particiones y no podría ejecutarse.

Este tema de particionamiento sufre de lo que se conoce como **fragmentación interna**, es decir, debido a que pueden entrar procesos de un tamaño inferior al fijado para la partición, puede quedar un espacio dentro de cada partición, el cual no puede utilizar ningún otro proceso porque cada partición está destinada a un único proceso.

* **Particiones estáticas de tamaño variable:** Volvemos a tener una memoria dividida en n particiones. En este caso las particiones vuelven a ser estáticas, pero tienen un tamaño variable (es decir, el tamaño de cada partición no es siempre el mismo, sino que habrá unas más grandes y otras más pequeñas).

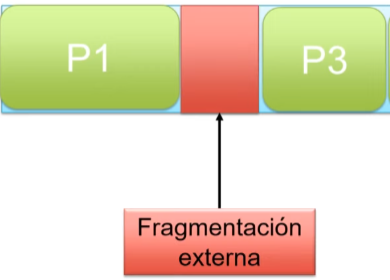
En este caso el tamaño de los procesos sí es importante para decidir qué partición de memoria se le asigna según las necesidades de memoria que tenga.

Podemos volvemos a tener el mismo problema que en las de tamaño fijo si nos llega un P4 que o bien sea demasiado grande o bien el hueco de memoria donde entraría está ocupado, se quedaría en espera o podría no llegar a ejecutarse nunca.

En este modelo volvemos a tener presencia de fragmentación interna.

* **Particiones dinámicas**: En este modelo no tenemos unas particiones estáticas, sino que son dinámicas, es decir, cada proceso irá cogiendo la cantidad de memoria que necesita para llevar a cabo su trabajo.

Por ejemplo, tenemos 3 procesos que han ido cogiendo la cantidad de memoria necesaria para hacer sus funciones

Termina el P2 liberando ese especio en memoria y llega un P4 que no entraría en el espacio liberado por P2 y se colocaría tras P3.

Ese espacio que ha liberado P2 y que no puede utilizar P4 es lo que se conoce como fragmentación externa, la cual la podemos definir como el espacio en memoria resultante de la liberación de memoria por un proceso que ha terminado su ejecución y que no satisface las necesidades de otros procesos entrantes.

## PAGINACIÓN

En la actualidad los sistemas informáticos necesitan que los programas se carguen de forma total y secuencial en memoria, de forma que un programa que necesita de 4 direcciones de memoria, debe ocupar 4 direcciones consecutivas de memoria para que el programa pueda llevarse a cabo.

El problema que tiene esta secuencialidad es que si tenemos 2 direcciones libres, seguidas de 2 posiciones ocupadas y luego n posiciones libres, si dividimos un programa de 4 instrucciones en 2 partes, de forma que las 2 primeras instrucciones ocupen las 2 direcciones libres anteriores a las que se encuentran ocupadas y luego las otras 2 instrucciones ocupen las 2 direcciones libres siguientes a las que se encuentran ocupadas, cuando el programa comience su ejecución, al terminar la instrucción 2 y dar paso a la 3 dará un error.

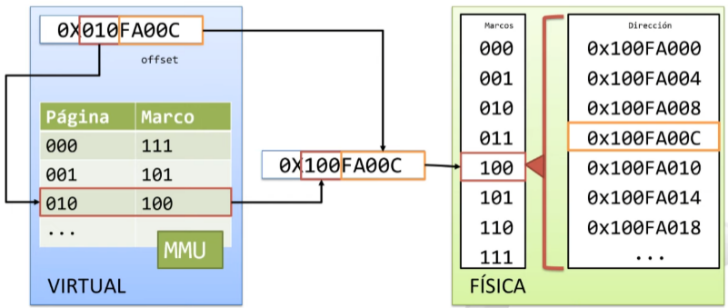
Para solucionar esto, se creó la **paginación**. Si pensamos en una serie de páginas que tengan una numeración y, además, un valor asignado a cada línea de la página (desplazamiento), podríamos acceder fácilmente y rápido a cada una línea específica de una página específica.

Si estas páginas estuvieran desordenadas el acceso a estas líneas seguiría siendo sencillo si tenemos una tabla donde almacenamos ese orden.

En la paginación lo que vamos a hacer es dividir la memoria en páginas que se van a numerar y en las cuales vamos a tener ese desplazamiento.

De esta forma, en la memoria física vamos a poder guardar estas páginas en lo que se conoce como **marcos de páginas** (son bloques de la memoria RAM del mismo tamaño)y van a poder estar desordenados. Cada marco de página se compone de un conjunto de direcciones de memoria física. Por ejemplo, un marco de página puede almacenar 64KB de memoria.

Para encontrar una dirección de memoria a través de la paginación partimos de una dirección virtual (por ejemplo, 0X010FA00C) que se divide en 2 partes: una parte destinada a definir la página donde se encuentra la dirección de memoria (010) y otra parte destinada al desplazamiento (FA00C) la cual se conoce también como ***offset***.

Una vez que sabemos la página la MMU se va a encargar de, a través de una tabla que enlaza páginas con marco, de traducir esa página por un marco (en este caso la página 010 se encuadra dentro del marco 100). Esto nos dará la dirección física a buscar (0X100FA00C) y con este dato se puede ir ya a memoria física, acceder al marco donde se encuentra la dirección de memoria a la que queremos acceder.

Esto va a permitir que podamos tener páginas desordenadas (vamos a poder tener el programa desordenado), ocupando de esa manera cualquier espacio de memoria y no tener problemas de fragmentación.

## TLB: TRANSLATION LOOKASIDE BUFFER

El problema que tiene la paginación es que debemos hacer 2 accesos a memoria:

* El primero para consultar la tabla de páginas que reside en memoria.
* El segundo para consultar realmente la dirección de memoria.

Forma

Descripción generada automáticamente con confianza mediaPara solucionar esto surge el **TLB o búfer de traducción de direcciones** que es una memoria cercana al procesador que almacena todas estas traducciones de manera que no se tengan que hacer siempre 2 accesos a memoria (sólo se harán cuando la traducción no esté en el búfer).

Lo que se hace es comprobar si la página de la traducción (la primera parte de la virtual) se encuentra en el TLB, si lo está, el TLB manda la traducción automáticamente.

En el caso de que no lo estuviera, entonces habría que ir a la tabla de páginas y realizar la traducción normal.

Diagrama

Descripción generada automáticamente

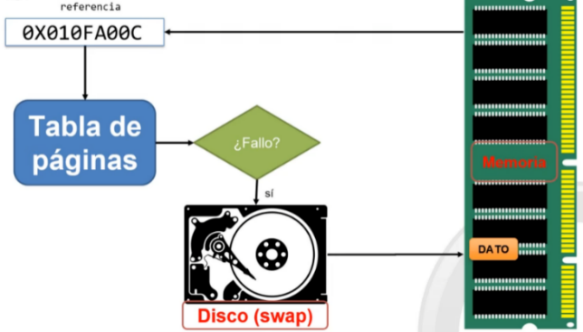
## MEMORIA VIRTUAL

Es una partición de espacio en memoria secundaria que se va a utilizar como una memoria virtual, es decir, que va a apoyar a la memoria principal.

Está íntimamente relacionada con la paginación y permite:

1. No tener todo el programa permanentemente cargado en memoria. Sólo se cargarán en memoria aquellas partes que se necesiten en determinado momento.
2. Tener un mayor espacio de direcciones. Al trabajar con direcciones virtuales que referencian a páginas, esas páginas podrían no estar cargadas en RAM, esas páginas estarán en memoria virtual, por tanto, las direcciones a las que se va a poder acceder son a las direcciones de RAM y de memoria virtual (SSD, HDD).

Esta memoria virtual reside en una parte del almacenamiento secundario y funciona de la siguiente manera:

1. Llega una solicitud de referencia.
2. Se consulta en la tabla de páginas:
   1. Si no da un error, es que esa dirección se encuentra en la memoria física y, por tanto, se devolverá directamente.
   2. Si da un error, es que esa dirección no se encuentra en la memoria física sino en la virtual (almacenamiento secundario). En este caso hablamos de que la dirección se ha **paginado**, es decir, se ha llevado a disco. Esto se conoce como **espacio/fichero de intercambio** (en Windows) o ***swap*** (en Linux). En este caso lo que se va a hacer es llevar el dato desde disco hasta memoria principal y una vez que el dato esté en memoria, se devuelve la referencia.

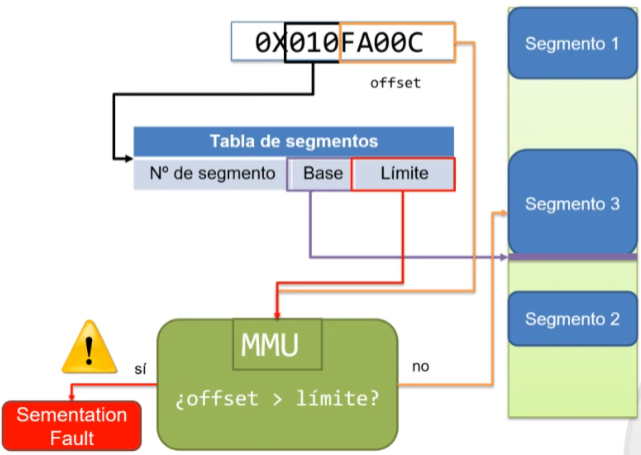
Por tanto, aunque tiene la ventaja de no cargar excesivamente la memoria o de tener mayor espacio, tiene el inconveniente de ser más lento al tener que ir a disco a por el dato.

## SEGMENTACIÓN

Es una alternativa a la paginación. Lo que va a hacer es dividir la memoria en diferentes segmentos que, además, van a ser de distintos tamaños.

Por tanto, la principal diferencia con la paginación es que esta divide a la paginación de forma estática en marcos de página y la segmentación coge partes de memoria de tamaño variable (segmentos).

La traducción va a ser muy similar, la dirección de memoria (0x010FA00C) se divide en 2 partes: la dirección del segmento (010) y el offset (FA00C) que en este caso será el desplazamiento dentro del segmento.

Con el segmento vamos a la tabla de segmentos que nos va a informar de un dato base (que informará de dónde empieza el segmento en memoria, es decir, el límite inferior) y un dato límite superior. Con el límite superior y el desplazamiento la MMU comprueba si el desplazamiento es mayor al límite, es decir, si se quiere acceder más allá del espacio en memoria que se tiene asignado:

* Si es mayor: vamos a recibir un error.
* Si no es mayor: vamos a recibir una dirección física de memoria.

Aunque surge como una alternativa, en los procesadores actuales segmentación y paginación conviven. Por ejemplo, en los procesadores Intel vemos segmentación para los diferentes programas, pero dentro de cada segmento tenemos paginación.