Tema N° 5: Gestión de Memoria

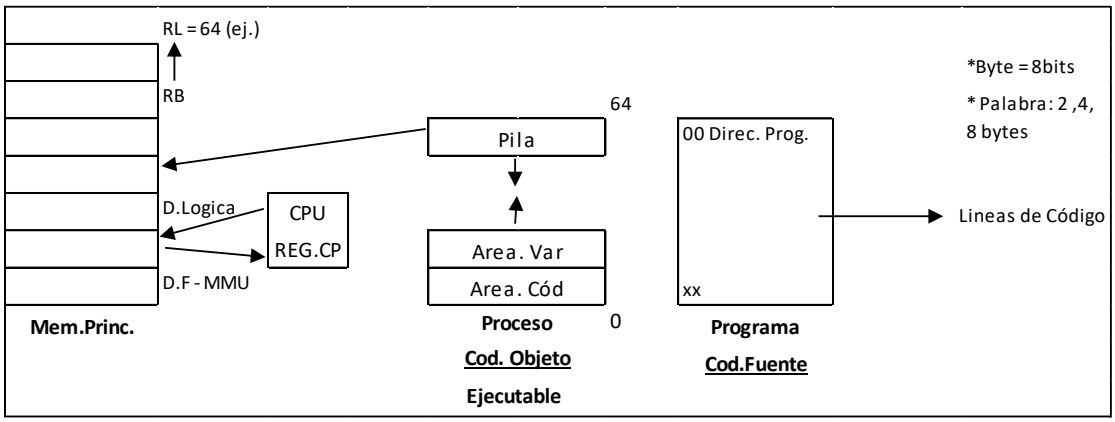
RESUMEN DE CLASE TEORICA

Nombre | Nombre del curso | Fecha

Sistemas Operativos parte II

GESTION DE MEMORIA

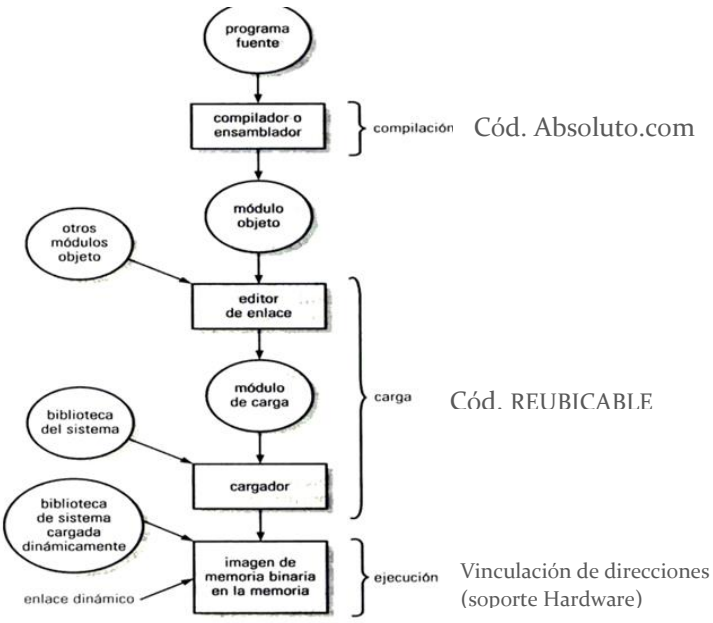
Todo proceso para poder ejecutarse debe estar alojado en MEMORIA PRINCIPAL. Qué es una memoria que está compuesta por un conjunto de bytes o de palabras que tienen una dirección que la registra como única.



El microprocesador tiene el CONTADOR DE PROGRAMA que dice cuál es la dirección, la próxima dirección que debe leer de la memoria, no sabe que hay en esa dirección.

- ✓ MEMORIA PRINCIPAL, es un contenedor de cosas.

Para que un PROGRAMA pase del estado "pasivo" y se convierta en PROCESO primero vamos a tener que pasar por la primera etapa...



1° Etapa de un proceso: COMPILACIÓN

- **Código Fuente**: Es el programa, con líneas de código, esas líneas de código tienen que pasar traducidas por un COMPILADOR O ENSAMBLADOR que va a generar direcciones, pero como no está todavía en la memoria, todo compilador o ensamblador va a generar direcciones de memoria relativa a su tamaño (traducidas en forma binaria).
  Una vez que se compila se dice que genera CODIGO ABSOLUTO, porque ya le da valores pero todavía no se sabe en qué lugar de la memoria se va a ubicar ese código.

- **Unidad de Gestión de Memoria** (M.M.U): Decide como asignar la memoria al proceso según su tamaño.

  Cuando la CPU genera una dirección, normalmente a esa dirección se le dice DIRECCION LOGICA, ahí va a tener que ejecutarse el MMU que es parte de la CPU y va a generar un dirección FISICA, que es la dirección real que tenemos que leer para ese proceso.

  El MMU controla que esa dirección física no se pase de la asignada a dicho proceso, para que el proceso no use lo que utiliza otro proceso.

- **Código Absoluto**: Es cuando ya sabemos que un proceso siempre va usar esa porción de memoria. Normalmente el código absoluto tiene la extensión **.COM** y pertenece al Sistema Operativo.

  Son rutinas que en tiempo de compilación ya generan la DIRECCION ABSOLUTA donde va a ejecutarse, porque se sabe que ahí se tiene que ejecutar. Son áreas de memoria que solo están destinadas para el S.O.

- **Módulo Objeto**: Es el proceso. Esto es el ejecutable, que quiere decir que ya tiene una imagen binaria, ya está traducido al lenguaje de la computadora.

  Cuando disparamos el proceso, el módulo objeto se convierte en proceso y hay que buscarle un lugar en memoria.

- **Editor de Enlace**: Une a nuestro código con otros módulos objetos (el área de código se hace más grande). Es decir carga el proceso más los módulos que se hayan insertado en el código. Cuando la carga se realiza en este nivel decimos que es ESTATICO porque lo carga, o sea lo hace más grande.

- **Carga de enlaces en forma Dinámica**: Se puede hacer en forma Dinámica, es decir que deja un fragmento dentro del código y en tiempo de ejecución cuando ya está en el microprocesador, si lo llama, verifica si no está la rutina y si no está viene y la trae ocupando entonces menos espacio. Una vez que realizo los enlaces hace la CARGA.

- **Módulo de Carga**: Es el planificador a largo plazo (PLP) que busca el espacio suficiente para que entre ese proceso, lo carga y le asigna el REGISTRO BASE y el REGISTRO LIMITE que se alojan en el PCB (bloque de control de procesos).

2da Etapa de un Proceso: CARGA

En esta etapa se dice que el código es REUBICABLE, porque si tenemos el PMP lo que hace es bajar procesos a disco, cuando tiene que volver a subirlos debe reubicar esos procesos (pueden o no estar en la misma área de memoria). ¿Cómo se reubica?

Se reubica dándole otro REGISTRO BASE (el registro LIMITE siempre es el mismo porque es el tamaño del proceso).

- **Cargador**: Carga la biblioteca del sistema, o sea todas las llamadas al sistema que va a necesitar ese proceso.

  ¿Qué son las llamadas al Sistema?

  Son los servicios que ese proceso va a necesitar del Sistema Operativo, siendo estos servicios otros procesos que también van a ocupar lugar en la memoria del Sistema.

- **La Biblioteca del Sistema**: puede estar en memoria o no, o solo puede estar una parte.

3ra Etapa de un Proceso: Ejecución

- Ya tenemos la IMAGEN DE MEMORIA BINARIA, es decir ya tenemos el proceso puesto en la memoria. Como se está ejecutando, se dice que en ese instante se hace la VINCULACIÓN DE DIRECCIONES, que es el cambio de DIRECCION LÓGICA a DIRECCIÓN FÍSICA, porque el microprocesador siempre me va a estar generando direcciones con respecto a la imagen del proceso, pero el MMU tienen que hacer su correspondencia con la dirección física donde se aloja, por eso está la vinculación de direcciones.

  El micro siempre genera DIRECCIONES LOGICAS, pero para ir a la memoria necesitamos la dirección FÍSICA y esta dirección nos la da el MMU a través de la vinculación.

"***Todo PROCESO pasa por tres etapas para ser ejecutado: La Compilación, La Carga y La Ejecución, con respecto al Gestor de Memoria***"

Técnicas que existen para manejar la Memoria

Existen técnicas para contener el proceso en memoria, siempre pensando que el proceso está en forma completa en memoria:

- **Enlaces Dinámicos:** Para que los procesos no ocupen demasiado espacio, se pueden hacer enlaces dinámicos, es decir, que hay rutinas que no vamos a cargar en memoria porque no sabemos si se llaman o no, como por ejemplo las rutina de errores.

- **Swaping**: Con el PMP lo que se hacía era bajar y subir procesos, esta es una técnica de memoria, con la que podemos bajar procesos al disco, o sea parte de los procesos al disco a una zona especial que se llama **ZONA DE INTERCAMBIO**.

- **La Superposición o Sobrescritura u Overling**: Es limitar a que el proceso, en todos los enlaces que cargue lo haga sobre su zona de intercambio, es decir, que va a sobrescribir.

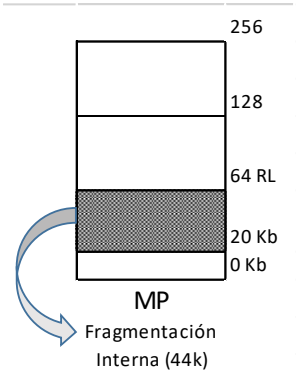
  Si trae un proceso en tiempo de ejecución va a tener que escribir sobre uno que ya está ocupando, sobre la misma porción de memoria, o sea a ningún proceso se le va a dar por el solo hecho de que quiera una porción más de memoria.

Técnica General para usar La Memoria

❖ **Asignación Contigua:** Significa que el proceso tiene que estar todo junto.

¿Cómo partimos a la memoria para contener un proceso?

❖ **Asignación Contigua con Particiones Fijas**:



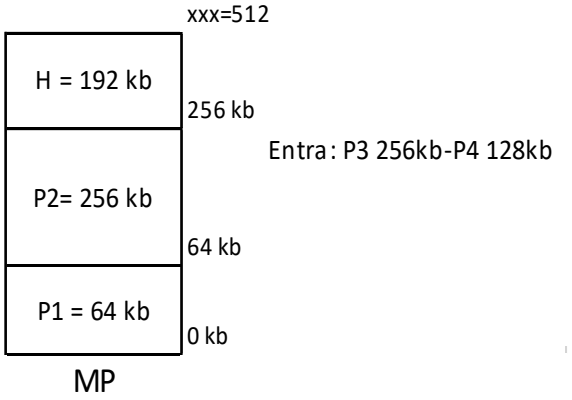
Consiste en particionar la memoria en partes iguales, en un solo tamaño llamado Hueco.

Los procesos indudablemente tenían un problema, que por ejemplo si le daban porciones de 64 kb, los procesos solo podían ser como MÁXIMO de 64 kb o usar técnicas de superposición.

Existe otro problema, supongamos que somos muy buenos programadores entonces escribimos programas pequeños. Por ejemplo: si se escribe un programa que mide 20kb y todas las llamadas se desarrollan en 20 kb y se tiene asignado 64 kb de espacio, los 44 kb restantes se convierte en espacio de memoria desperdiciado porque no puede asignarse a ningún proceso más, entonces esta ZONA se llama FRAGMENTACIÓN INTERNA, que quiere decir que tenemos porciones de memoria que el proceso no utiliza y no puede ser asignado a nadie más porque pertenece a ese proceso. Y cuánto más pequeño sea el proceso más desperdicio de memoria tendremos. La FRAGMENTACIÓN INTERNA no tiene solución.

Existe una Tabla de particiones de Memoria (TPM), dentro del PCB, existe un puntero.

❖ **Asignación Contigua de Espacios Variables**: Comienza teniendo la memoria como un gran bloque o Hueco del total de la memoria.



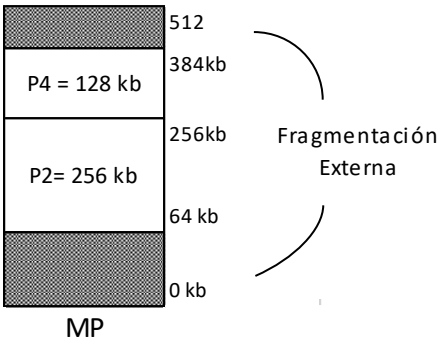
TPM

	RB	RL
P1	0	64
P2	64	256

Esta en el PCB de cada proceso

TABLA DE HUECOS LIBRES

RB	RL
256	192



Como el proceso P3 ocupa 256kb y no hay suficiente memoria espera. Cuando termina de ejecutarse el proceso P1 y libera el espacio que tenía recién hay espacio suficiente de memoria libre para el proceso P3 que entraría justo, pero como el proceso debe estar en forma CONTIGUA, este proceso no entra.

Luego entra el proceso P4 de 128kb, que entrará en el hueco de arriba (192kb), recién llega y puede entrar, mientras que el proceso P3 continua esperando.

Esto de tener varios Huecos de distintos tamaños, que se irán generando a medida que entren los procesos, se llama FRAGMENTACIÓN EXTERNA, que es un problema que tiene la Asignación Contigua de Espacios Variables.

Solución de la Fragmentación Externa – Compactación

La Compactación consiste en hacer un Swaping de los procesos y reacomodarlos en memoria uno tras del otro, dejando todos los **Huecos Unidos** en un solo extremo de la memoria.

Pero la compactación tiene el problema de que cuando hacemos la compactación, No se puede ejecutar ningún proceso, porque se está bajando al disco y obtenemos tiempos muertos de microprocesador al estar haciéndose la Compactación.

Algoritmos para seleccionar el Hueco de Memoria Libre

- **Primer Ajuste:** Selecciona el primer hueco que encuentra con el espacio suficiente para contener ese proceso.
- **Mejor Ajuste:** Va a revisar toda la tabla y va a seleccionar el hueco que mejor se ajuste al proceso, generando de esa manera huecos pequeños.
- **Peor Ajuste:** Política contraria al mejor ajuste, busca el hueco más grande, generando así huecos grandes.

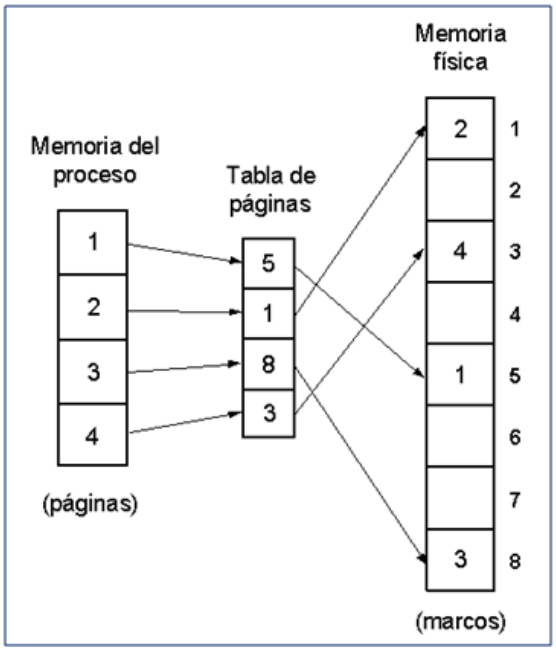
  ¿Cuál de estas 3 Políticas es la mejor?

  Entre el Mejor Ajuste y el Primer Ajuste dan los mismos resultados, pero se dice que mejor es el Primer Ajuste porque no tiene ningún algoritmo. Es más rápido, ahora como resultado de Fragmentación los dos dan lo mismo.

Paginación

La Paginación consiste en dividir la memoria en particiones fijas llamada Marcos y dividir al proceso en particiones fijas llamadas Páginas.

Tabla de Páginas	
Pág.	Marco / RB
1	5
2	1
3	8
4	3



Cada página puede ser contenida en un marco no necesariamente contiguo. La correspondencia es que cada página va asociada a un marco. Para poder hacer la correspondencia entre las páginas y los marcos, cada proceso tiene una TABLA DE PAGINAS.

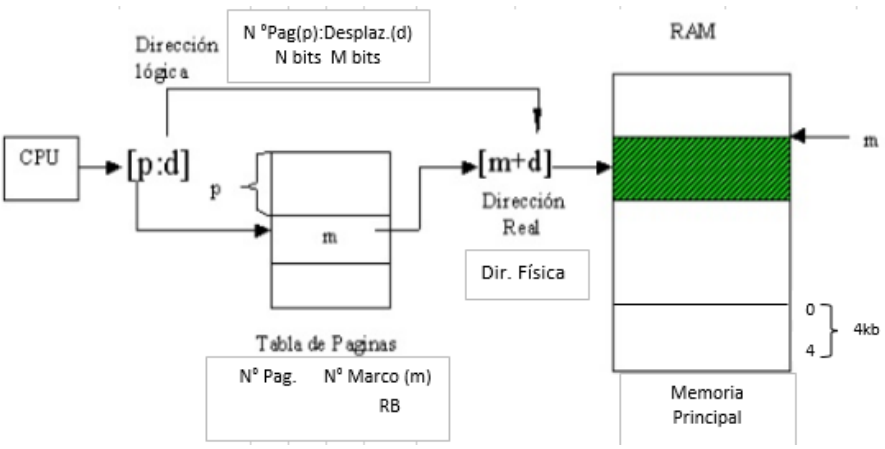
Se tiene Fragmentación Interna pero es muy pequeña, porque el tamaño del marco es pequeño, la partición es de pocas líneas de código y como el proceso también lo estamos partiendo, la única página que tiene Fragmentación Interna será la última página.

O sea que la fragmentación es pequeña, por eso es que si bien la fragmentación interna es un problema, en este caso no lo es.

¿Qué hay dentro de cada página?

Hay direcciones pero no podemos saber si son datos, código, variables, pila; no podemos saber porque nosotros al proceso le estamos particionado en porciones de 4kb (a la imagen binaria lo estamos dividiendo en porciones).

¿Cómo funciona?



Cuando el micro genera la dirección Lógica, que va a constar de dos partes: N° PAGINA y el DESPLAZAMIENTO dentro de esa página.

El desplazamiento, tanto en la dirección Lógica, como en la dirección Física es el mismo. O sea que el Desplazamiento que tengo en Página lo voy a tener en el Marco.

¿Cómo hago la correspondencia del N° de Página al N° de Marco?

Se hace a través de la Tabla de Páginas, con el N° de Pág. entro a la Tabla de Páginas y obtengo el N° de Marco o registro Base.

La Longitud de la Pág. nos va a determinar la cantidad de bits para contener el N° de Pág.

- ***Conclusión:*** La ventaja que me da la Paginación es que ya no se necesita tener el proceso todo junto, eso quiere decir que marco libre que haya en memoria. Marco que puede ser asignado a una página del proceso. Con esto se elimina la FRAGMENTACIÓN EXTERNA, o sea que optimiza el uso de la memoria porque tampoco es muy significativa la FRAGMENTACIÓN INTERNA, ya que es solo en la última página.
- ***Desventaja:*** La Tabla de Pág. debe estar en memoria para ser accedida, esto quiere decir que quita espacio a la memoria. ¿Cuántas veces tengo que leer la memoria principal para obtener la DIRECCIÓN LÓGICA? Dos veces, una para leer la Tabla de Pág. y otra para encontrar la dirección. Ya no se lee una sola vez la memoria, sino que tengo que tener dos accesos a memoria para poder traer la instrucción, una para convertir haciendo la correspondencia del LÓGICA A FÍSICA y otra para poder ubicar a la instrucción en la dirección real, esto hace que la PAGINACIÓN sea lenta, por eso así sola no sirve por lo que necesita ayuda del Hardware.

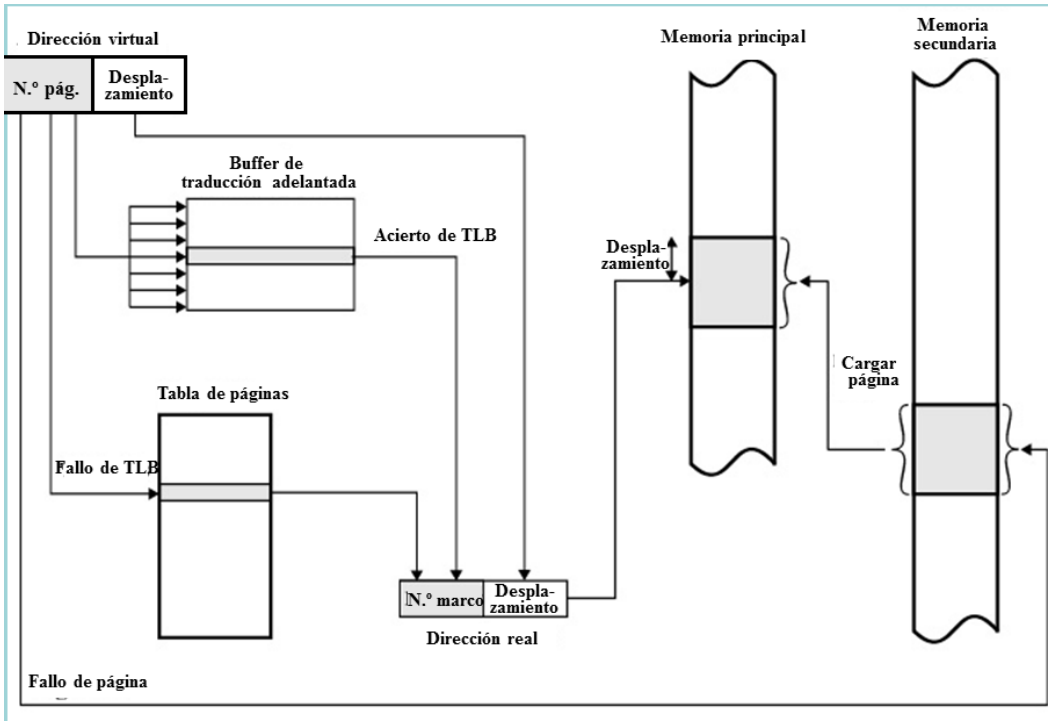
Por esta razón se crearon más registros dentro del microprocesador que se llaman: REGISTROS ASOCIADOS o BUFFER DE TRADUCCIÓN DE MIRADAS RÁPIDAS o simplemente TLB. Como son registros que están en el microprocesador hacen que el acceso sea más rápido.

Entonces, cuando la CPU ve el contador de Programa con el TLB, mira la TLB, si no está se deba a que es la primera vez que vamos a ejecutar la página, luego de esto recién se va a buscar a la Tabla de Pág., lee y la copia en el REGISTRO el N° de página y N° de marco, como las instrucciones de los procesos se ejecutan una tras otra razón por la cual se dice que los programas son secuenciales, si es la primera instrucción y vuelvo a decir que tengo 4 instrucciones ahí, por lo menos 4 veces la voy a encontrar en la TLB y no necesito ir a leer la memoria. La correspondencia se hace dentro del microprocesador gracias a la TLB y recién accede a la memoria principal, eso hace que no tengamos que leer tantas veces la memoria principal. Gracias al hardware (TLB) la paginación es posible, rápida y es lo que se utiliza.

La base de la Paginación es la asignación contigua con espacios fijos, la diferencia es que en la paginación se hacen más pequeñas las particiones y también se parte el proceso por lo cual la paginación permite que no esté todo junto el proceso.

>> ***Conclusión***: La Paginación consiste en dividir el proceso en páginas y la memoria en marcos, y se hace una correspondencia ya que cada página entra en un marco. Como es

lento a través de la Tabla de Páginas, se utiliza un Registro Asociado o TLB, donde cada vez que se lea una nueva página que no este se cargará en ese registro. Indudablemente como estos registros son la imagen de este proceso ¿dónde va a estar cargado? Estará cargado en el PCB, y como la Tabla de Páginas está en memoria, dentro del PCB tenemos un puntero a la Tabla de Páginas.

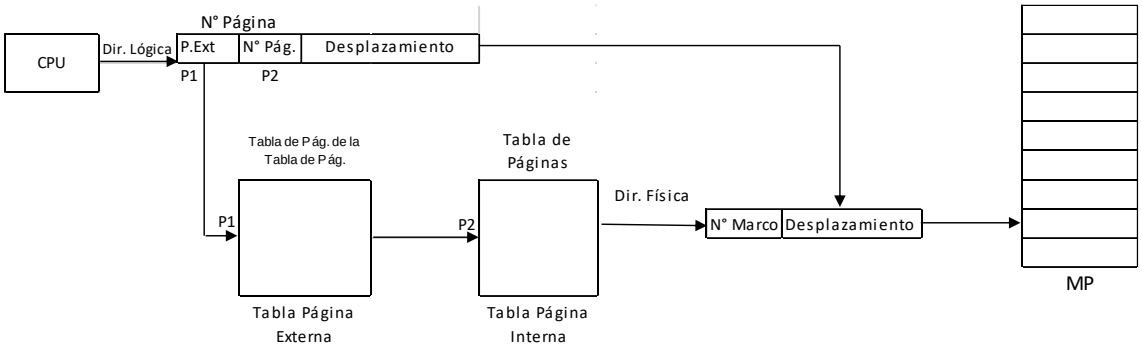


Paginación Multinivel

- **Proceso Pesado:** Es un proceso grande, cuando se trabaja en un solo bloque o sea monolítico, todas las rutinas y tronco principal del programa está todo junto, entonces obtenemos muchas líneas de código lo que provoca que nuestro proceso sea muy grande, si es muy grande lo que se va a agrandar es el tamaño de la tabla de páginas, cuanto más grande sea la tabla de páginas más marcos de memoria va a utilizar.

  La memoria está dividida en marcos, lo óptimo sería que una tabla de página ocupe un marco, pero ¿qué ocurre cuando la tabla de páginas es muy larga?, la vamos a tener que dividir, porque no va a ocupar uno sino varios marcos, entonces vamos a tener que Paginar la Tabla de Páginas, a eso se llama Tabla de Páginas de Multinivel.

  Cuando se va a paginar la Tabla, va a generar ya no solo el N° de Página del proceso sino que también tenemos el N° de Pág. de la Tabla de Páginas

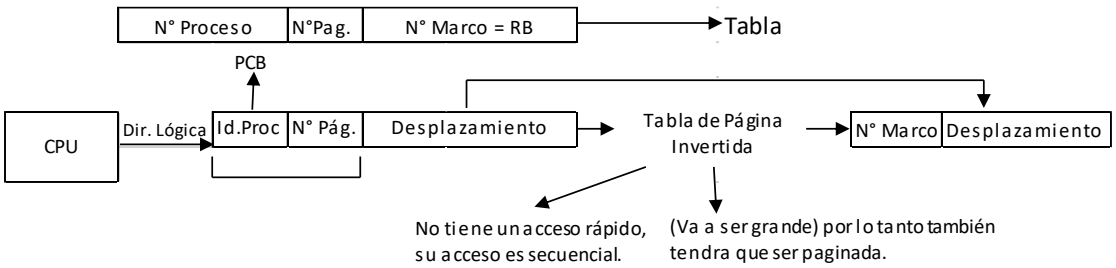


Esta figura muestra la Tabla de Páginas Multinivel, o sea es paginar la Tabla de Páginas. Por eso es que o es conveniente hacer programas demasiados grandes, porque si antes en la paginación nosotros accedíamos dos veces a memoria, ahora accedemos tres veces a memoria: para la Página Externa, para la Página Interna y para la Memoria. Cuánto más grande sean los procesos más lentos son.

Otra forma de hacer la Tabla de Páginas es, como cada proceso tiene una tabla de páginas vinculada; lo que significa que nosotros vamos a estar usando un marco por cada proceso que tenemos en memoria, si tenemos 4 procesos tendremos 4 Tabla de Páginas, o sea que vamos a tener 4 marcos menos para utilizar, porque van a estar usadas para las tabla de páginas de esos procesos. Esto es un desperdicio de memoria en tablas de páginas, entonces inventaron lo se conoce como la TABLA DE PAGINA INVERTIDA.

Tabla de Página Invertida

La Tabla de Página Invertida usa el concepto al revés, en vez de que cada proceso tenga una tabla de páginas, hace una única tabla de marcos ocupados. O sea hay una sola tabla de página que es del tamaño de todos los marcos que tengamos disponibles en memoria, esa tabla de páginas está compuesta por:



Segmentación

La Segmentación en realidad depende mucho del compilador, se basa en dividir un proceso según las construcciones lingüísticas que se realice.

- ✓ ***La Paginación***: Se encarga mucho de lo que es físicamente la memoria.
- ✓ ***La Segmentación***: Es otro concepto, se inclina más a como se escriben los PROGRAMAS por eso es que depende de la compilación.

Cada Segmento tiene un tamaño distinto y genera una tabla de Segmentos.

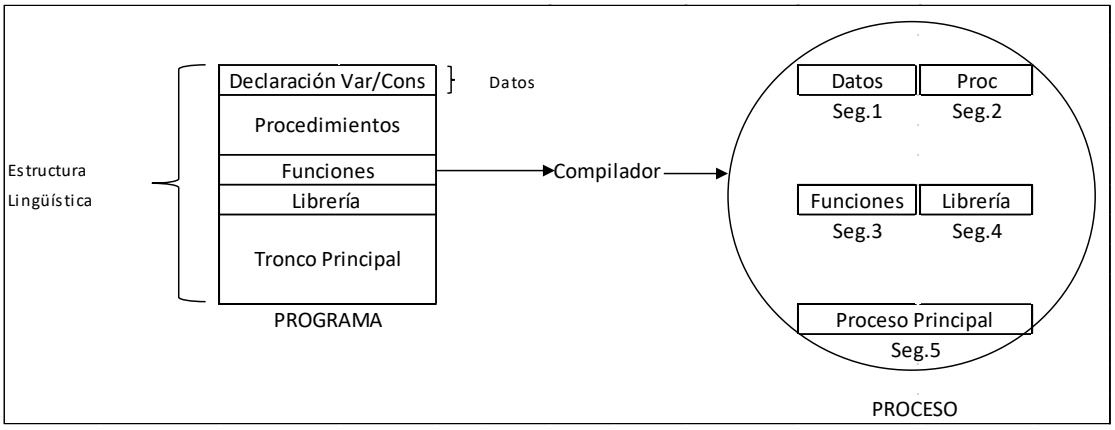
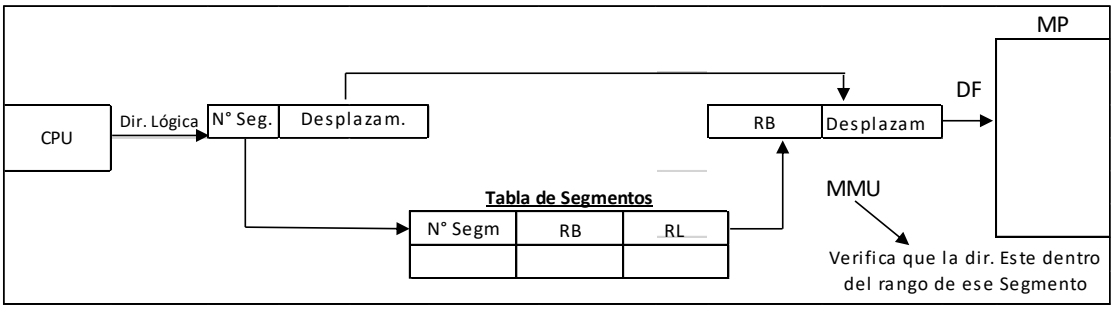


Tabla de Segmentos

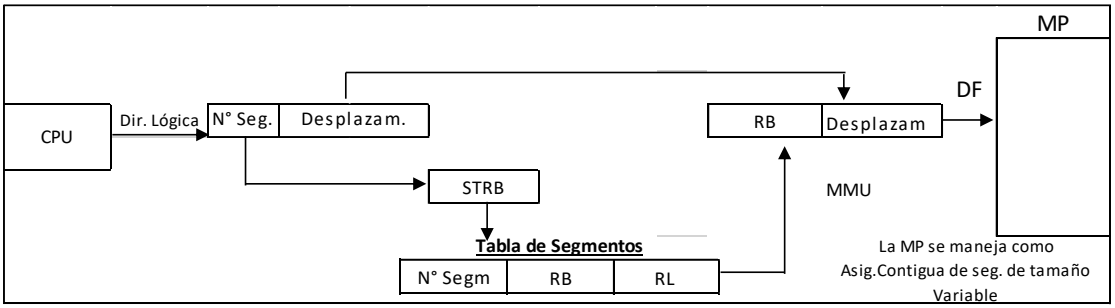
N° Segm	RB	RL

V/I

Como los Segmentos tienen un tamaño distinto la memoria se va a manejar como una asignación contigua de espacios variables. Pero ahora tenemos una tabla y de un proceso tenemos múltiples segmentos. ¿Qué logramos con esto? Logramos que, por ejemplo, si viene otro proceso que usa las mismas funciones, ya no serán necesario cargarlas ya que las podemos COMPARTIR, como son construcciones lingüísticas entonces queda aislado y sabemos que hay en esa construcción.



- Se acceden dos veces a memoria. Para acelerar el acceso a memoria se le agrega la STRB, que seria los registros asociativos o Buffer de Traducción de mirada rápida de segmentos.



- El Segmento esta contiguo, el que está disperso es el proceso.
- El Segmento es un solo bloque y tiene que estar todo junto.
- Tamaño Fijo → Paginación.
- Tamaño Variable → Segmentación.

>> ¿Cuál es la ventaja de la Segmentación con respecto a la Paginación?

Que se puede compartir, es fácil compartir los segmentos y vos sabes que es lo que estas compartiendo en cambio en la paginación no, porque si compartís un marco no sabes lo que contiene.

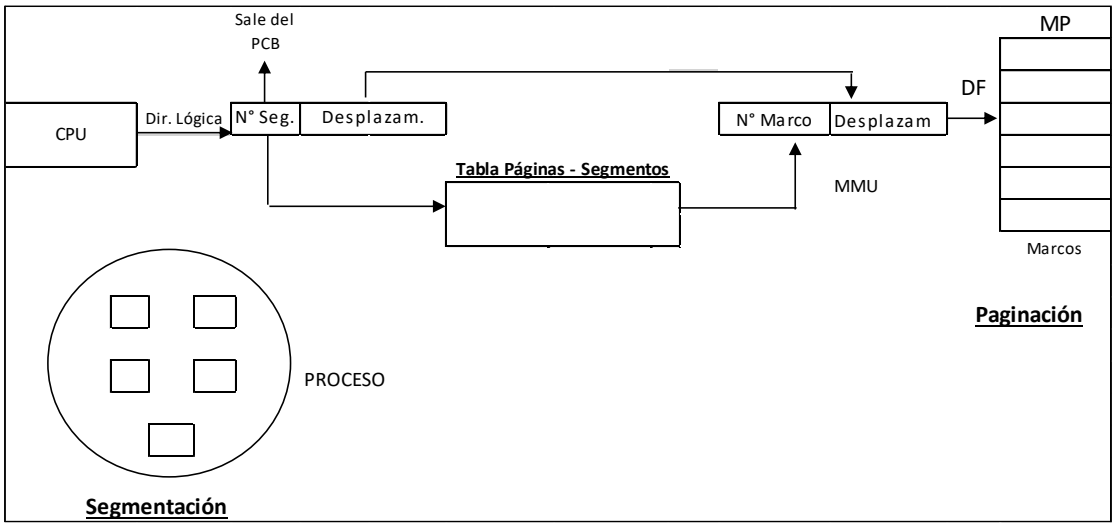
>> ¿Cuál es la ventaja de la Paginación con respecto a la Segmentación?

La FRAGMENTACION EXTERNA, ya que la asignación contigua de tamaño variable tenía el problema de la fragmentación externa y por lo tanto necesitamos de la Compactación.

- La Paginación es lo mejor para el uso físico de la memoria, porque no tenemos desperdicio, o sea no tenemos fragmentación externa que es un problema y la fragmentación interna es mínima.
- La Segmentación tiene la ventaja de que se puede compartir, entonces podes cargar menos cosas a la memoria.
- En las Tablas de Segmentos como en la Tabla de Página, se agrega un bit más que se llama bit de Validez o bit Invalidez (V/I), este bit es el que nos permite acceder a esa página.

Segmentación Paginada

En la Segmentación Paginada o Segmentación con Paginación, el compilador genera los segmentos y cada segmento tiene una tabla de página asociada



Relación proceso-tabla de página, se encuentra en el PCB del proceso: donde tenemos el ID del proceso y dentro de ese vamos a tener el ID del Segmento y el puntero a la tabla de páginas.

PCB	
N° Seg.	T.Pág.

Por eso la memoria se maneja con Paginación, y el proceso se genera con SEGMENTACIÓN, por eso se dice segmentación con paginación.

Como los segmentos son como módulos individuales se los trata como si fueran procesos hijos del proceso padre que es el proceso general.