MEMORIA VIRTUAL:
PAGINACIÓN Y SEGMENTACIÓN

La **Memoria Virtual** es una técnica que permite la ejecución de procesos que pueden no estar completamente en memoria principal. La ventaja principal de este esquema es que los programas pueden ser mayores que la memoria principal. Esto se debe a que se crea una abstracción de la memoria principal, separando la memoria lógica, tal como la ve el usuario, de la memoria física de la que realmente se dispone.

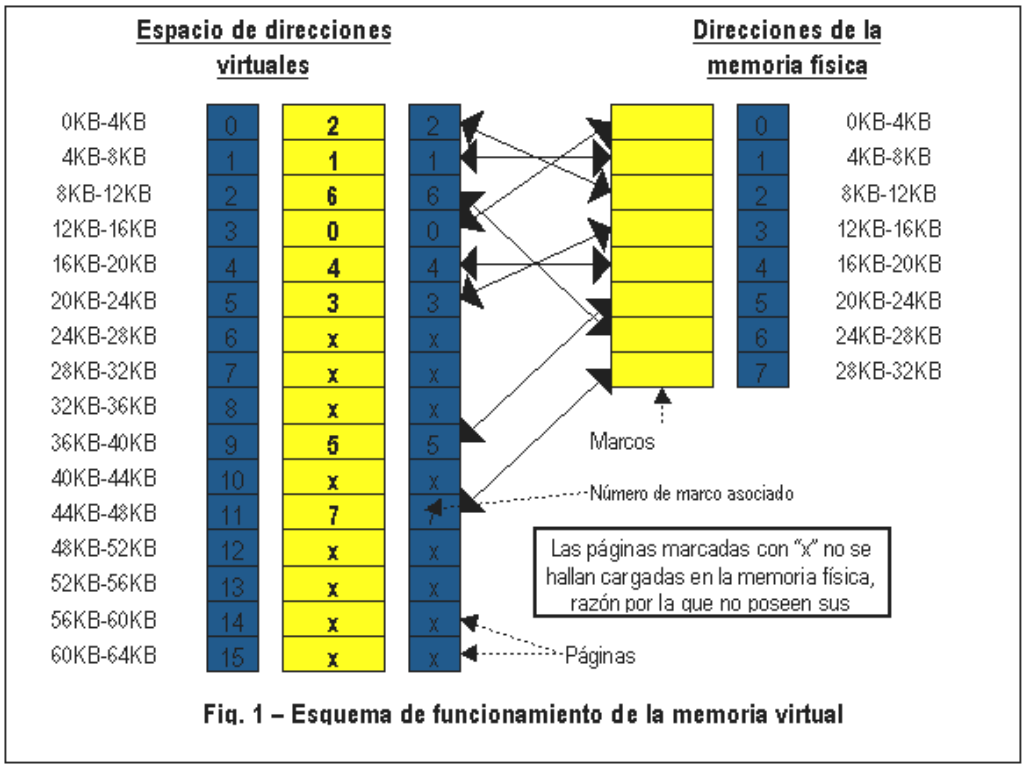
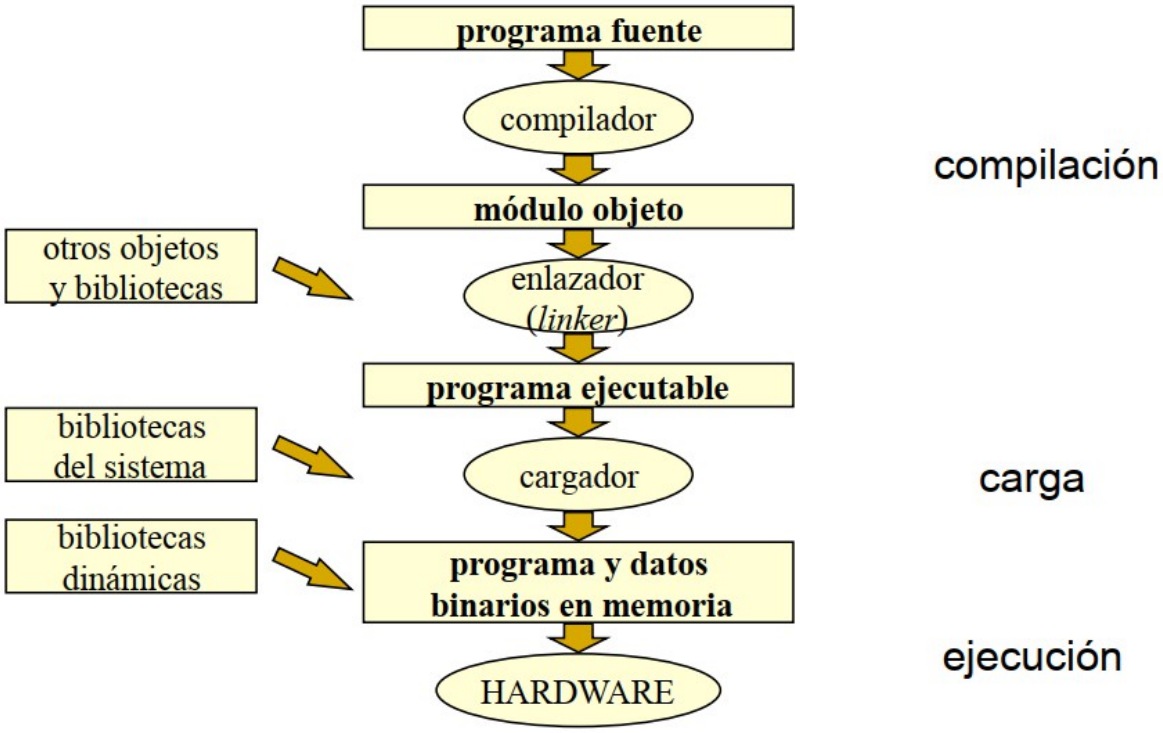


Fig. 1 – Esquema de funcionamiento de la memoria virtual

Aunque la memoria virtual podría estar implementada por el software del sistema operativo, en la práctica casi siempre se usa una combinación de hardware y software, dado el esfuerzo extra que implicaría para el procesador.

Ciclo de vida de un programa



El **COMPILADOR** traduce direcciones de memoria simbólicas a direcciones binarias. Si estas direcciones son definitivas (absolutas), el programa sólo se puede ejecutar en una zona fija de la memoria. Si las direcciones generadas son provisionales entonces:

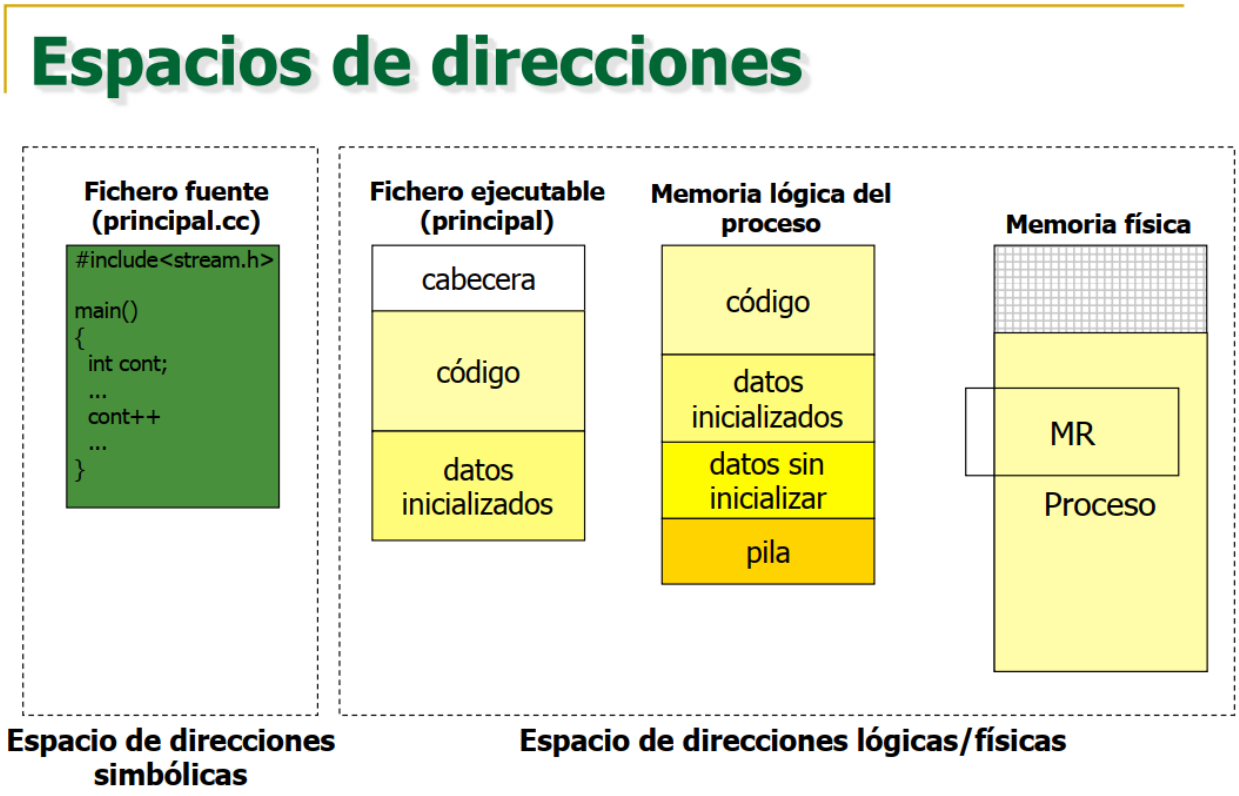
El **ENLAZADOR / CARGADOR** resuelve direcciones binarias provisionales a direcciones binarias definitivas, el gestor de memoria en SO trabaja con reubicaciones estáticas en memoria. Pero si en esta fase seguimos trabajando con direcciones provisionales, entonces:

La fase de **EJECUCIÓN** que implica al HARDWARE SÍ genera direcciones binarias definitivas → direcciones físicas

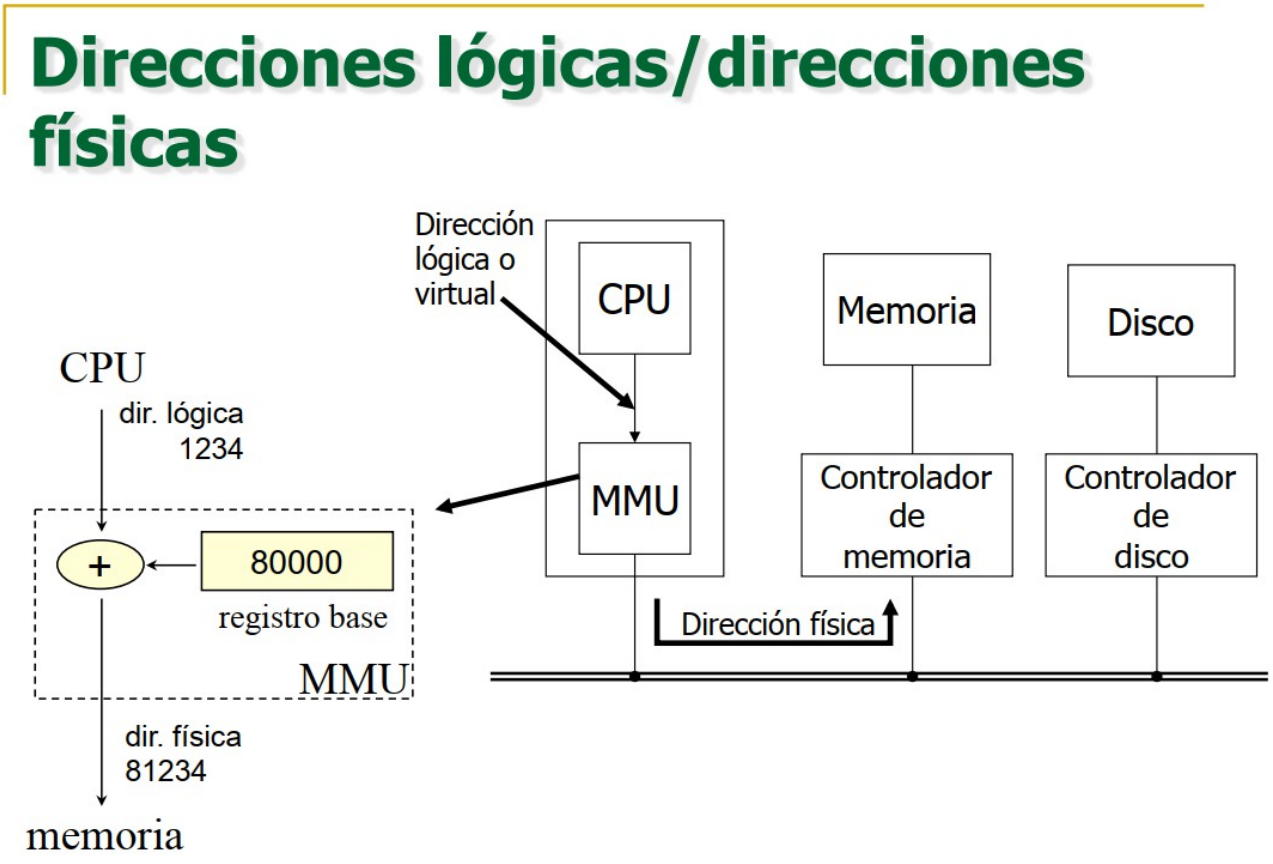
Dirección física: la que llega al *chip* de memoria

Dirección lógica o virtual: la generada por la CPU

El espacio de direcciones lógicas y el espacio de direcciones físicas no tienen por qué coincidir.



El dispositivo que traduce direcciones virtuales a físicas se llama **unidad de manejo de memoria** (MMU, en inglés)

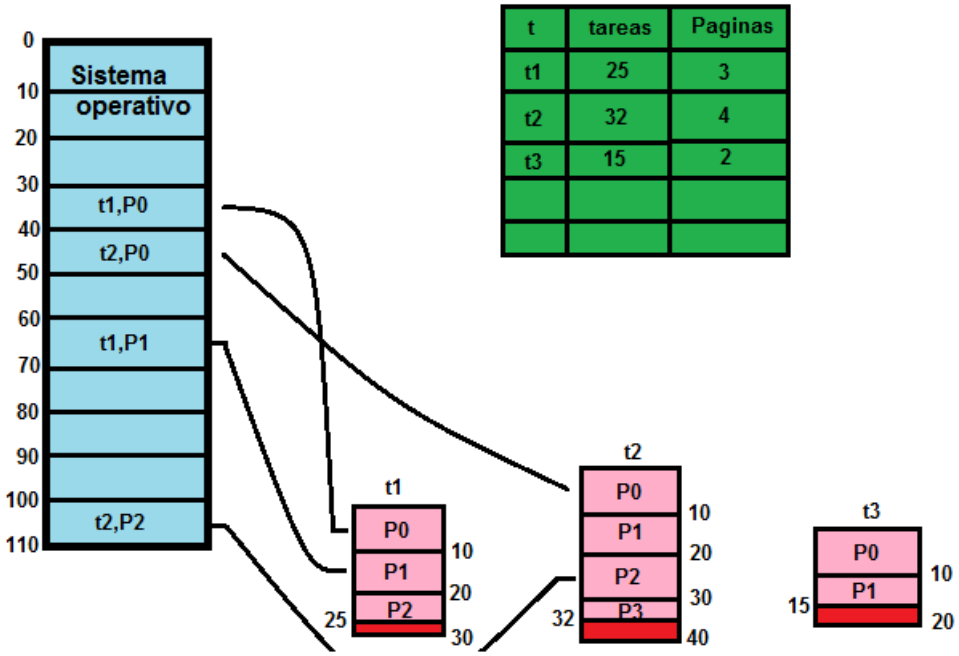


Paginación

Cuando un proceso no entraba en memoria porque el tamaño de los huecos era menor que el tamaño de dicho proceso, podíamos solucionar con una compactación en curso, pero dicha solución requiere tiempo y no siempre es viable.

La gestión de memoria paginada es una gestión basada en la asignación de memoria física no contígua, es decir, que las partes de un proceso lógico puedan estar situadas en áreas no contiguas de la memoria física.

El espacio de direcciones de cada proceso se divide en bloques de tamaño uniforme llamados páginas, los cuales se pueden colocar dentro de cualquier marco de página disponible en memoria.



De esta manera, la fragmentación interna se reduce sólo a una fracción de la última página del proceso . Además, no hay fragmentación externa.

Cuando se solicita cargar un proceso de tamaño T el S.O. debe asignar E marcos, donde Ese obtiene de dividir T entre el tamaño de página más uno. Si por ejemplo, la página es de 32 bytes y tenemos la dirección 200, significa que se necesitan 7 bloques: int(200/32)+1 (donde int() es la función que proporciona la parte entera de la división)

Como se ve, el S.O. asigna un número entero de marcos. Si el tamaño del proceso no es múltiplo entero del tamaño de página, el último bloque no se utiliza en su totalidad, produciéndose una fragmentación, que es siempre menor que el tamaño de página.

Por tanto, una vez calculado el número de bloques, se procede a cargar el proceso sobre los mismos construyendo la tabla de páginas de dicho proceso.

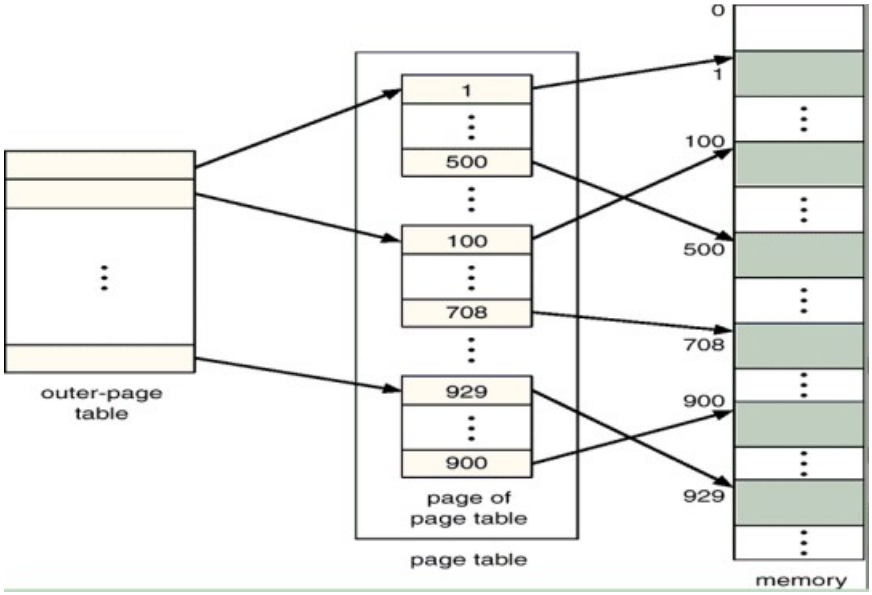
El tamaño de las páginas viene definido por el hardware y suele darse como potencias de 2.

TABLAS DE PAGINACION

Las tablas de paginación o tablas de páginas son una parte integral del Sistema de Memoria Virtual en sistemas operativos, cuando se utiliza paginación.

Son usadas para realizar las traducciones de direcciones de memoria virtual (o lógica) a memoria real (o física) y en general el sistema operativo mantiene una por cada proceso corriendo en el sistema.

La Tabla de Páginas muestra la posición del marco de cada página del proceso. Dentro del programa, cada dirección lógica constará de un número de página y de un desplazamiento dentro de la página.



Cuando las tablas de páginas son muy grandes se puede utilizar un esquema de paginación de varios niveles para que las páginas se paginen a sí mismas.

VENTAJAS DE LA PAGINACION

Es posible comenzar a ejecutar un programa, cargando solo una parte del mismo en memoria, y el resto se cargara bajo la solicitud.
No es necesario que las paginas estén contiguas en memoria, por lo que no se necesitan procesos de compactación cuando existen marcos de páginas libres dispersos en la memoria.
Es fácil controlar todas las páginas, ya que tienen el mismo tamaño.

DESVENTAJAS DE LA PAGINACION

El costo de hardware y software se incrementa, por la nueva información que debe manejarse y el mecanismo de traducción de direcciones necesario.
Se consume mucho más recursos de memoria, tiempo en el CPU para su implantación. Se deben reservar áreas de memoria para las PMT de los procesos.
Al no ser fija el tamaño de estas, se crea un problema semejante al de los programas (como asignar un tamaño óptimo sin desperdicio de memoria, u "ovearhead" del procesador).

Segmentación

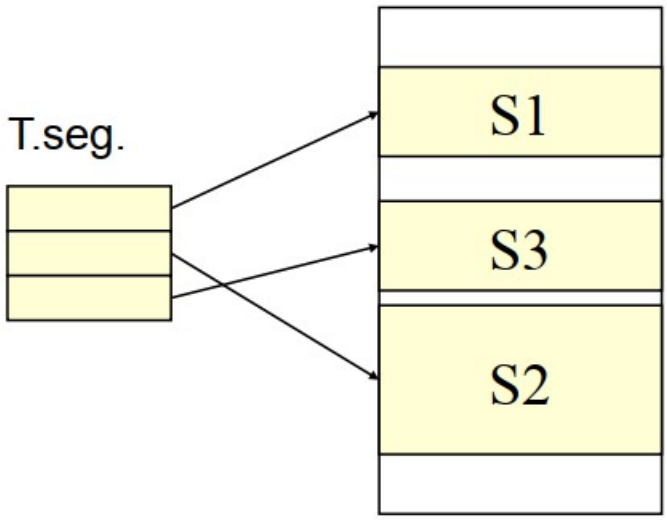
La segmentación de memoria es un esquema de manejo de memoria mediante el cual la estructura del programa refleja su división lógica.

Un programa se puede descomponer en varios **segmentos** de memoria (código, datos, pila…)

De la misma forma la memoria en estos segmentos, cada uno de los cuales tiene una longitud variable, definidos intrínsecamente por el tamaño de ese segmento del programa.

Los elementos dentro de un segmento se identifican por su desplazamiento, esto con respecto al inicio del segmento.

Con el *hardware* adecuado, podemos ubicar esos segmentos en zonas de memoria no contiguas.

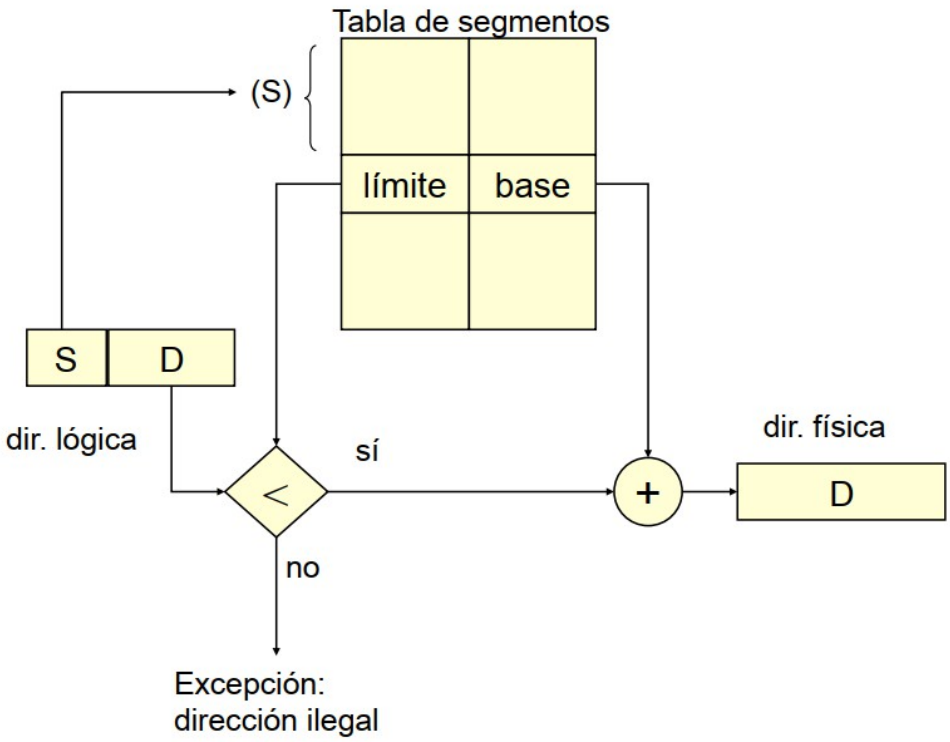


El compilador tiene que generar código que haga referencias a direcciones de memoria con dos componentes: <segmento,desplazamiento>

El S.O. ubica cada segmento en un hueco contiguo de memoria

El *hardware* se encarga de la reubicación dinámica mediante una **tabla de segmentos:**

Hardware de segmentación



VENTAJAS DE LA SEGMENTACION

El programador puede conocer las unidades lógicas de su programa, dándoles un tratamiento particular.

Es posible compilar módulos separados como segmentos, el enlace entre los segmentos puede realizarse mediante una referencia entre segmentos.

Es fácil el compartir segmentos.

Es posible que los segmentos crezcan dinámicamente según las necesidades del programa en ejecución.

DESVENTAJAS DE LA SEGMENTACION

Hay un incremento en los costos de hardware y de software para llevar a cabo la implantación, así como un mayor consumo de recursos: memoria, tiempo de CPU, etc.

Debido a que los segmentos tienen un tamaño variable se pueden presentar problemas de fragmentación externas, lo que puede requerir un plan de reubicación de segmentos en memoria principal.

Registro Base y Limite

El registro base y el registro límite pueden servir para localizar direcciones de memoria.

El registro base sirve como referencia para ubicar una dirección en particular y el registro límite ayuda a determinar si el desplazamiento de una dirección está por encima del área asignada. Este mecanismo sirve de protección para la memoria.

- ***Registro límite :*** tamaño máximo del programa y los datos
- ***Registro base :*** posición de inicio del programa en memoria