Administración de Memoria.

Sistemas Operativos

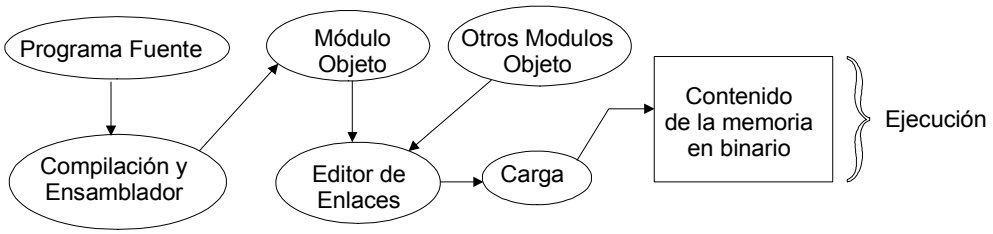
Tema 4.

Administración de memoria.

- Jerarquía de memoria:
  - Registros CPU del procesador
  - Caché (memoria rápida)
  - Memoria principal RAM
  - Almacenamiento secundario (memoria virtual)
- Al bajar en la jerarquía más capacidad pero más lento es el dispositivo y más barato.
- Administrador de memoria:
  - Parte del S.O. que gestiona la memoria:
    - Control de que partes de la memoria están utilizadas o libres.
    - Asignar memoria a procesos y liberarla cuando terminan.
    - Administrar intercambio entre memoria y disco (Memoria Virtual).

Administración de memoria.

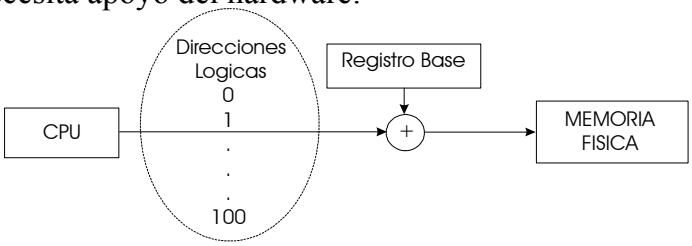
- Proceso de Compilación y Carga de un Programa:



- Ejemplo: (enlace de direcciones) Programa ensamblador con salto a una etiqueta:
  - ETIQ --
    --
    jmp ETIQ

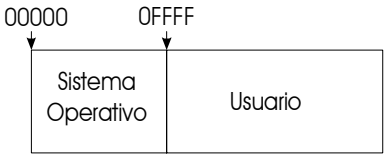
Proceso de Compilación y Carga de Programas.

- **¿En que momento se realiza el enlace o traducción de direcciones?**
  - **Compilación:** Generando código absoluto, en el momento de compilación se sabe donde residirá el programa en memoria.
  - **Carga** (Reubicación estática):
    - El compilador genera código relocalizable.
    - Se crean direcciones de memoria absolutas cuando se carga el programa en memoria.
  - **Ejecución** (Reubicación dinámica) :
    - Durante la ejecución puede moverse el código de un proceso.
    - Necesita apoyo del hardware:



Administración en sistemas Monoprogramados.

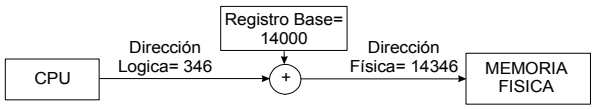
- En sistemas monoprogramados generalmente la memoria principal dividida en dos particiones:
  - Una para el usuario:
    - Un proceso con su código.
    - Dirección a partir de la que se cargan programas de usuario.
  - Otra para el sistema operativo residente (memoria baja).



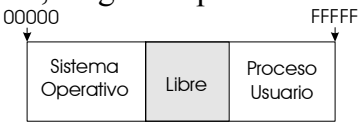
  - Es necesario proteger las particiones entre sí.

Administración en sistemas Monoprogramados.

- A veces el tamaño del S.O. desea variarse:
  - Ej.: Manejadores de dispositivos que no se usan.
    - Se puede realizar una reubicacion dinámica del espacio.



    - También, cargar los procesos de usuario en memoria alta.



Administración en sistemas Multiprogramados.

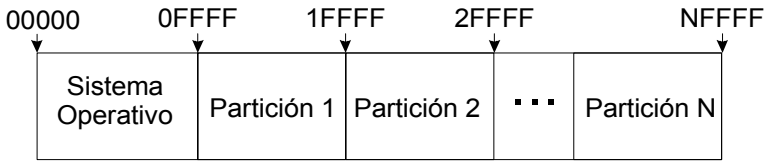
- Es deseable que haya varios procesos en memoria para su ejecución concurrente.
- Se debe compartir la memoria entre varios procesos que esperan asignación de la misma.
- **Esquemas de asignación de memoria:**
  - Contigua: particiones fijas y variables
  - Intercambio (swapping)
  - Paginación
  - Segmentación
  - Segmentación paginada

Administración en sistemas Multiprogramados.

- Primer esquema de asignación de memoria: **Particiones**
  - La memoria está dividida de antemano en espacios (Particiones).
  - Un proceso necesita ejecutarse -> Se le asigna uno de dichos espacios (Partición).
  - Cada partición puede contener un único proceso.
  - Pueden ser:
    - Particiones Fijas:
      - Todas el mismo tamaño.
      - Con diferentes Tamaños.
    - Particiones Variables.

Particiones Fijas.

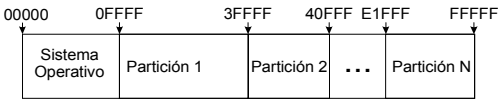
- Particiones de igual tamaño:



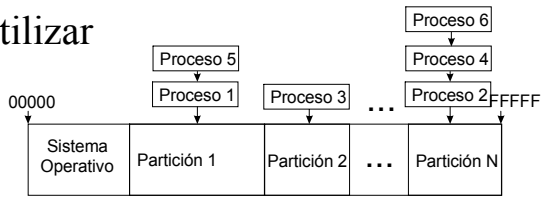
  - Nivel de multiprogramación limitado por número de particiones.
  - Hay una cola con procesos que quieren utilizar memoria y ejecutarse.
  - Hay una tabla para indicar particiones ocupadas y libres.

Particiones Fijas.

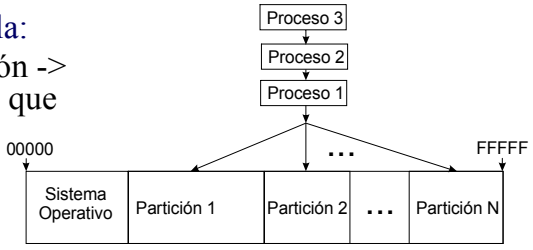
- Particiones con diferentes tamaños:



- Para procesos que quieren utilizar memoria para ejecutarse:
  - Podemos tener varias colas:
  - Cada proceso se asigna a una cola en función de su tamaño.

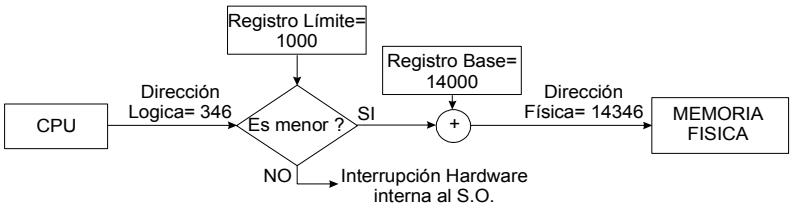


  - Podemos tener una única cola:
  - Cuando se libera una partición -> se asigna al primer proceso que cabe en ella.



Particiones Fijas.

- Problemas que presenta este tipo de asignación de memoria:
  - Debe proporcionarse reubicación:
    - ¿En que partición entrará el proceso?.
  - Existe Fragmentación Interna y Externa:
    - Interna:
      - Una partición asignada y no ocupada totalmente por el proceso.
    - Externa:
      - Un proceso quiere ejecutarse, hay una partición libre, pero de menor tamaño que el proceso.
  - Necesidad de protección: (en sistemas multiprogramados)
    - Un proceso no acceda al área de memoria del otro.
    - Si la reubicación es dinámica puede usarse registros base-límite.



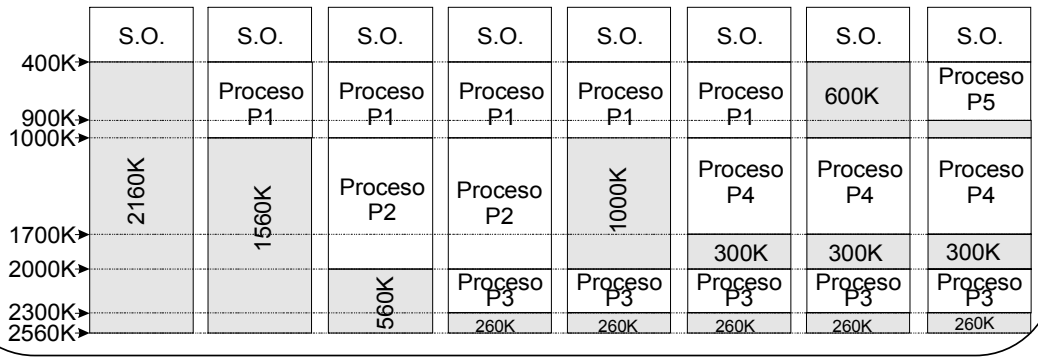
Particiones Variables.

- Funcionamiento:
  - Inicialmente: Toda la memoria (salvo partición del S.O.) disponible para procesos, como si fuese un gran hueco.
  - Llega un proceso:
    - Se introduce en un hueco libre.
    - El espacio no ocupado será un nuevo hueco.
  - Cada zona de memoria ocupada -> una partición.
  - Proceso termina:
    - Libera su zona de memoria.
    - Se convierte en un hueco.
    - Dicho hueco se fusiona con los adyacentes.
  - Se conserva una tabla de partes de memoria ocupadas y libres y la cola de entrada de procesos en memoria.

Particiones Variables.

- Un ejemplo:los procesos se cargan en memoria, compiten por la CPU y al acabar liberan la memoria

Proceso	Memoria Requerida	Proceso	Memoria Requerida
P_1	600 K	P_4	700 K
P_2	1000 K	P_5	500 K
P_3	300 K		



Particiones Variables.

- **Fragmentación** de Particiones Variables:
  - Externa: SI. (memoria dividida en huecos pequeños)
    - Suma del espacio libre en memoria suficiente para el nuevo proceso.
    - Pero no hay huecos suficientemente grandes para él.
    - El nuevo proceso no se carga en memoria.
  - Interna: NO.
    - Las particiones se crean con el tamaño solicitado por el proceso.

Particiones Variables.

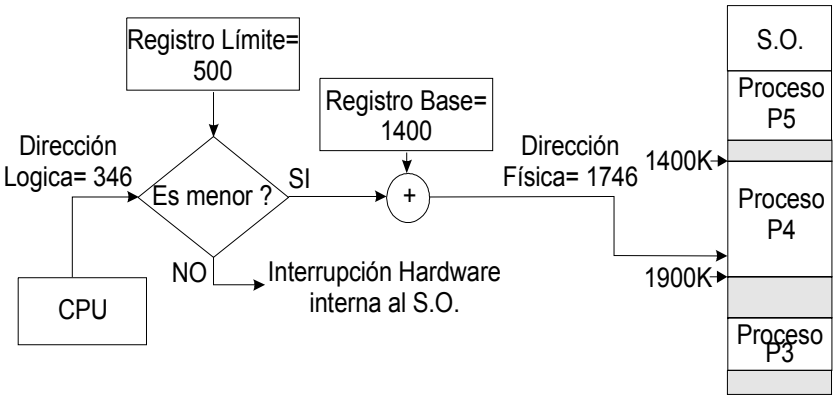
- Esta asignación de memoria se denomina: **Asignación dinámica de almacenamiento**
- ¿Como elegir un hueco cuando llega un nuevo proceso de tamaño N?
- Estrategías:
  - Primer Ajuste:
    - Escoge el primer hueco libre de tamaño suficiente.
  - Mejor Ajuste:
    - Hueco más pequeño con tamaño suficiente (requiere ver toda la lista si no está ordenada).
  - Peor Ajuste:
    - Hueco más grande: Pretende conseguir que los huecos que queden sean grandes (requiere ver toda la lista si no ordenada).

Particiones Variables.

- ¿Cuál es el mejor?
  - Simulaciones y Estadísticas:
    - Criterio tiempo (reducción) y utilización de memoria (aprovechamiento):
      - "Primer Ajuste" y "Mejor Ajuste" son mejores que "Peor Ajuste".
    - Regla del 50%: un análisis estadístico refleja que
      - Con Primer Ajuste por cada N bloques de memoria asignados se pierden 0,5 N bloques por fragmentación externa (1/3 memoria inutilizada).

Particiones Variables.

- **Protección de Memoria:** se utiliza código reubicable
  - Si código reubicable -> se pueden usar registros base y límite.

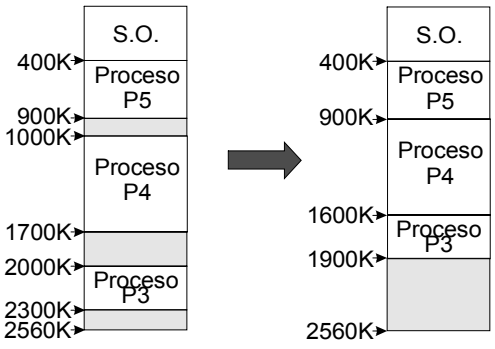


Particiones Variables.

- **Compactación:** intenta solucionar fragmentación ext.
  - Consiste en desplazar las particiones ocupadas para que estén juntas en memoria:
    - Queda un solo hueco libre de mayor tamaño.
  - Es una solución al problema de fragmentación externa.
  - Sólo es posible si la reubicación es dinámica (en ejecución).
  - Ejemplo:

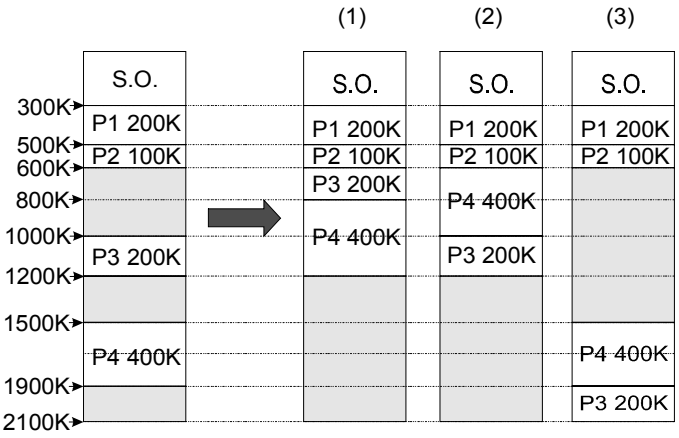
100+300+260=

Hueco de 660k



Particiones Variables.

- Problemas de la Compactación:
  - Consume tiempo: Desplazar zonas de memoria.
  - Difícil seleccionar una estrategia de compactación óptima.



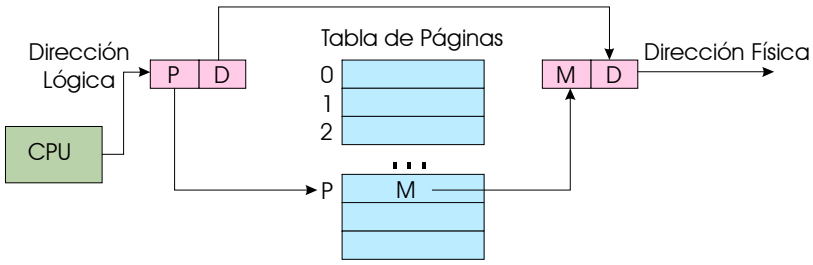
  - ¿Cuál es la mejor?

Paginación.

- **Paginación:** (solución a fragmentación externa)
  - Permite que la memoria de un proceso no sea contigua.
  - Hay una distinción entre direcciones lógicas y físicas.
  - La memoria física la dividimos en bloques de tamaño fijo: ***marcos***.
  - La memoria lógica:
    - La dividimos en bloques llamados: ***páginas***.
    - De igual tamaño que el marco.
  - Las páginas de un proceso se cargan en los marcos de la memoria principal que estén disponibles:
    - Tenemos "trozos" del proceso allí donde la memoria está disponible.

Paginación.

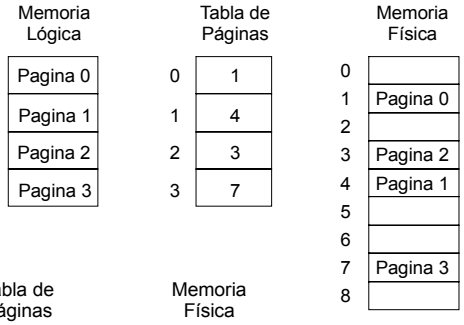
- Hardware de paginación: para traducción de direcciones

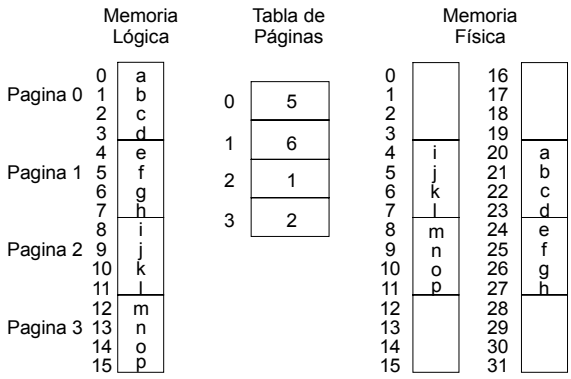


  - La dirección lógica generada consta de dos partes:
    - Número de Pagina (P).
    - Desplazamiento dentro de la página (D).
  - La tabla de páginas: (contiene la dirección base en memoria física)
    - Permite establecer una correspondencia entre el número de página y un número de marco de memoria física.
  - La dirección física es el número de marco y el desplazamiento.

Paginación.

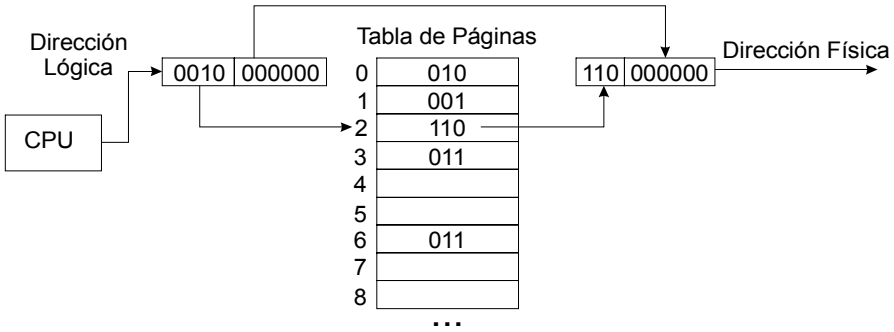
- Ejemplos:





Paginación.

- Tamaño de páginas y marcos definidos por Hardware.
- Normalmente se escoge un tamaño de página potencia de 2:
  - Ya que es más fácil la traducción de direcciones lógicas a físicas.



Tamaño memoria lógica 2^m
tamaño página 2^n (bytes o palabras)
P índice en tabla de páginas
D desplazamiento

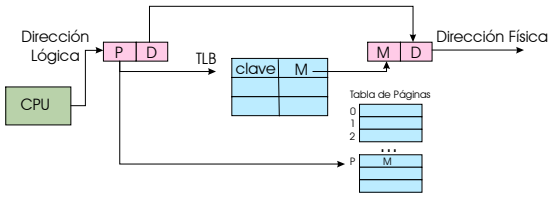
M-n bits altos de la dirección lógica= P
n bits bajos de la dirección lógica = D

Paginación.

- El SO traduce direcciones usando una copia de la tabla páginas en memoria
- **Implementación Hardware de la Tabla de Páginas:**
  1) **Un conjunto de registros** (circuitos lógicos de alta velocidad):
     - Habrá que cargar estos registros en un cambio de contexto.
     - Se usa para pocas entradas (unas 256)
  2) **Tabla en memoria principal y registro base** cuyo contenido apunta a la tabla de páginas:
     - Para cambiar de tabla de páginas -> Basta cambiar de registro base.
     - Menor tiempo de cambio de contexto pero mayor de acceso a memoria
       - Accedemos dos veces a memoria para obtener un dato en memoria.
     - Para tablas grandes (millones de entradas)

Paginación.

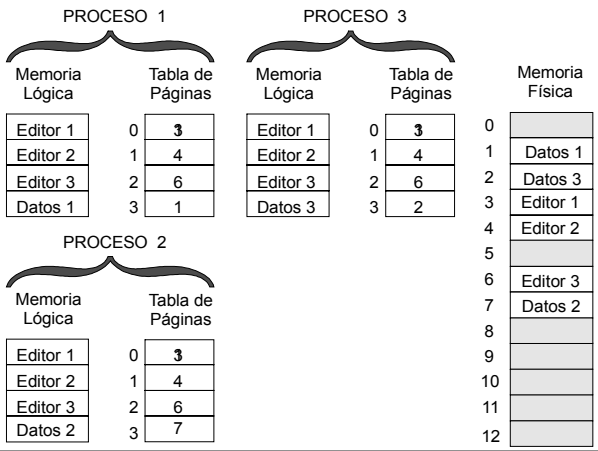
3) **Registros Asociativos (TLB):** (pequeña caché de acceso rápido), (translation look-aside buffers)
   - Los registros contienen solo unas pocas entradas de una T.páginas
   - 2 partes en cada registro:
     - Una clave (número de página).
     - Y un valor (número de Marco).
   - Compara el valor de la página deseada con todas las claves.
     - Si la clave está: Proporciona el número de marco asociado.
     - Si no está: Se accede a la tabla de páginas de memoria.



Paginación.

- **Ventaja:** Páginas Compartidas:
  - La paginación permite compartir código común entre varios procesos:
    - Sólo si el código es reentrante (no se modifica durante ejecución).
    - El área de datos de los procesos sería diferente.
    - Ejemplo: varios procesos ejecutan el mismo editor de textos

Una única copia
Del editor en
Memoria física



Paginación.

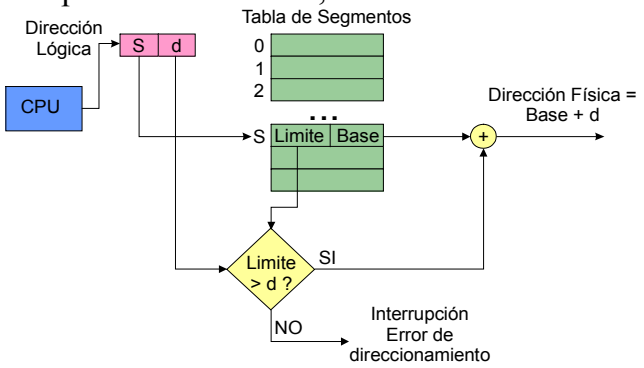
- Protección de memoria en entorno con paginación:
  - En la tabla de páginas pueden encontrarse unos bits de protección asociados a cada marco
  - indican si la página es de sólo lectura o lectura y escritura.
  - Cuando se consulta el número de marco, se consultan además los bits de protección.
  - Se debe controlar que el número de página no supere el total de páginas usadas por el proceso (sería una dirección incorrecta).

Segmentación.

- Otro esquema de asignación memoria: **Segmentación**
  - El espacio de direcciones lógicas se compone de un conjunto de segmentos: Cada uno tiene un nombre y una longitud.
  - Para el usuario las direcciones especifican el nombre del segmento y el desplazamiento dentro de él.
  - El nombre del segmento se numera (es un número).
    - <número segmento, desplazamiento>
  - El procesador *Intel 8086* usa segmentación, los programas se separan en:
    - Segmento de Código.
    - Segmento de Datos.
    - Segmento de Pila.
  - Hay una división lógica del proceso en diferentes segmentos.

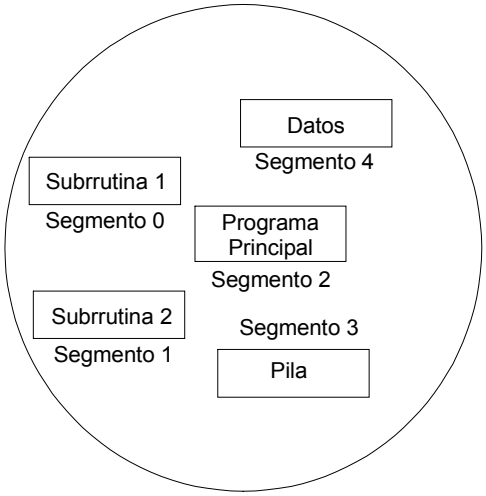
Segmentación.

- Hardware de segmentación mediante Tabla de segmentos:
  - Establece la correspondencia entre direcciones físicas y lógicas.
  - Se busca en la tabla de acuerdo con el número de segmento.
  - Cada entrada 2 registros:
    - base (dir. Física inicial del segmento en memoria)
    - límite de segmento (longitud del segmento)
  - Se compara límite del segmento con desplazamiento.
  - Si desplazamiento válido, se suma a la dirección el registro base.

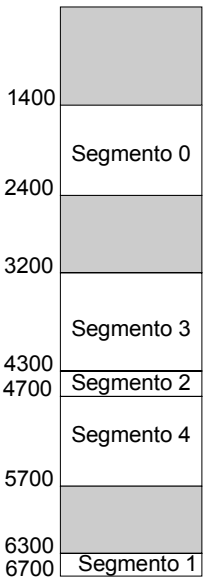


Segmentación.

–Ejemplo: sean 5 segmentos en memoria física



	Limite	Base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700



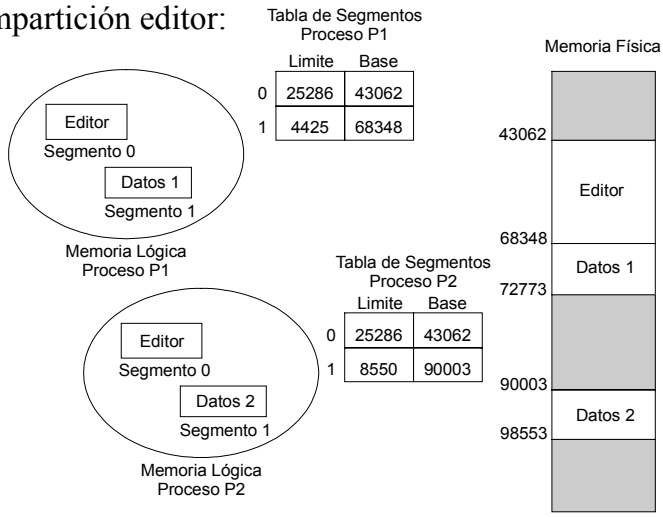
- Acceso a byte 1200 del segmento 0 da error direccionamiento

Segmentación.

- **Implementación Hardware de la tabla de segmentos:**
  - Puede ubicarse en registros rápidos o memoria (como paginación).
  - Si está en memoria:
    - Un registro base STBR (segment table base register) indica inicio de la tabla de segmentos en memoria.
    - Un registro límite indica longitud de la tabla de segmentos.
- **Protección:**
  - Bits de protección: Segmento de sólo lectura o lectura y escritura.
  - Se consultan antes de acceder al segmento.
- **Compartición de código:**
  - Puede realizarse a nivel de segmento (código o datos).
  - Cada proceso tendrá una tabla de segmentos.
  - Compartir un segmento significa que una entrada de la tabla de segmentos coincide en varios procesos (igual posición física).

Segmentación.

- Ejemplo compartición editor:



- Si compartimos un segmento todos los procesos que lo comparten deben definir dicho segmento con el mismo código.

Dirección (S , desplazamiento)

Segmentación.

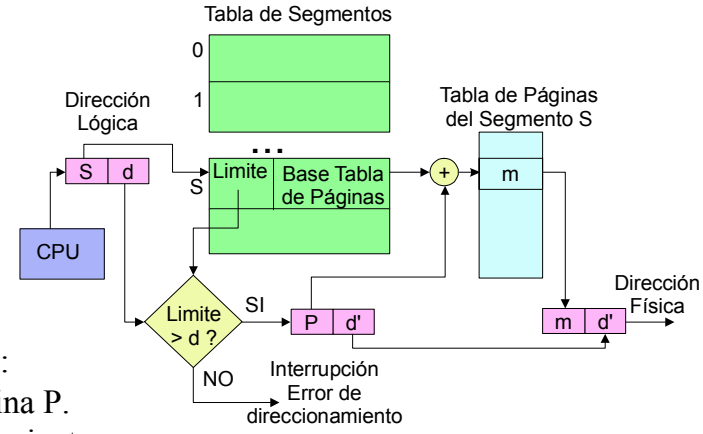
- Fragmentación:
  - Los segmentos son de tamaño variable:
    - Puede haber fragmentación externa.
    - Bloques de memoria demasiado pequeños para contener un segmento.
  - Solución: Se puede compactar la memoria (segmentación usa reubicación dinámica).
  - Problema de fragmentación, casos extremos:
    - Cada proceso un segmento, igual esquema que en particiones variables.
    - Cada palabra (byte) un segmento:
      - No habría fragmentación externa.
      - Necesitamos una tabla de segmentos del tamaño de la memoria.

Segmentación Paginada.

- Otro esquema de asignación de memoria es: **Segmentación paginada**
  - La Memoria lógica está dividida en bloque llamados segmentos que contienen las regiones de un proceso.
  - Dirección lógica=<nº segmento, desplazamiento>=<S,d>
  - Los segmento están divididos en páginas de igual tamaño que los marcos (potencias de 2).
  - Las páginas de un proceso se cargan en marcos de la memoria principal.
  - Cada segmento tiene asociada una tabla de páginas
  - Se usa un registro límite y base de la tabla de páginas para cada segmento

Segmentación Paginada.

- Esquema de traducción de direcciones
  - Dirección lógica=<nº segmento, desplazamiento>=<S,d>
  - S= entrada de la tabla de segmentos:
    - Contiene el límite del segmento
    - Contiene la dirección base de una tabla de páginas.
    - Habrá una tabla de páginas por cada segmento.
  - El desplazamiento d es:
    - Un número de página P.
    - Un nuevo desplazamiento dentro de la página d'.



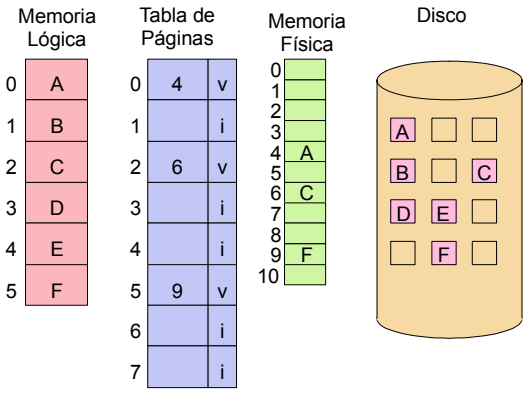
Memoria virtual.

- Recordemos que queremos:
  - Mantener simultáneamente varios procesos en memoria para permitir multiprogramación.
- Memoria Virtual:
  - Permite separar la memoria lógica del usuario de la memoria física.
  - Un proceso en ejecución no tiene porque encontrarse totalmente en memoria principal (sólo parte).
  - Ahora un proceso puede ser mayor que la memoria física.
  - Permite transferencia de información entre memoria principal y secundaria (2 niveles consecutivos de la jerarquía de memoria).
  - Usa un dispositivo de almacenamiento secundario (disco) como dispositivo de intercambio.
- La memoria virtual puede implementarse sobre Paginación o Segmentación paginada: se transfieren páginas.
- La transferencia suele ser bajo demanda.

Paginación por demanda.

- Paginación por demanda:
  - Los procesos están divididos en páginas.
  - Inicialmente: una serie de páginas del proceso cargadas en memoria principal (MP), las que se usan.
  - El resto en almacenamiento secundario.
  - Necesario un bit de presencia en tabla de paginas: ***Bit válido-inválido***
    - 1, página cargada en MP (v).
    - 0, página no cargada (i).

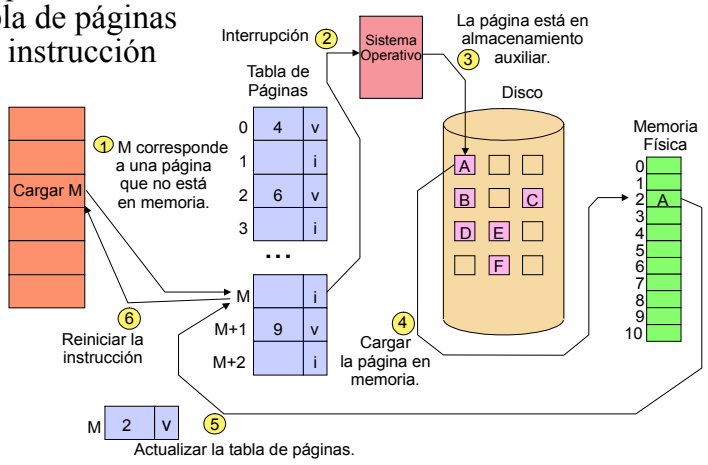
- Si el proceso accede a páginas residentes en memoria (bit de presencia válido):
  - *la ejecución prosigue normalmente*
- Si accede a una página no residente (bit presencia inválido)
  - *Ocurre una interrupción o* ***fallo de página*** **,** Control al SO



Paginación por demanda.

- **Gestión de un Fallo de página**

1. Se detecta que la página no está en memoria
2. Se produce una interrupción
3. Se busca la página en almacenamiento secundario (disco)
4. Se busca un marco libre, se lee la página de almacenamiento secundario y se copia en el marco seleccionado
5. Se actualiza la tabla de páginas
6. Reiniciamos en la instrucción interrumpida



Paginación por demanda.

- Hardware de apoyo a la paginación por demanda:
  - Capacidad de marcar en la tabla de páginas una entrada como válida o invalida (*bit valido-invalido*).
  - Unidad de almacenamiento secundario:
    - La sección de disco empleado para este fin se denomina: espacio de intercambio o almacenamiento auxiliar.
- Paginación por demanda pura:
  - Caso extremo: comenzamos la ejecución de un proceso sin ninguna página cargada en memoria.
  - Se irán produciendo fallos de páginas sucesivamente y cargando las páginas necesarias.

Segmentación Paginada con Paginación por Demanda.

- No todas las páginas de todos los segmentos estarían en memoria.
- Usamos también bits de *valido-invalido* para la tabla de páginas asociada a cada segmento.
- El funcionamiento es igual que paginación por demanda.

Reemplazo de páginas.

- Utilizando Memoria Virtual:
  - Los procesos tienen parte de sus páginas cargadas en memoria.
  - En un instante, la totalidad de los marcos de memoria están ocupados.
- **¿Qué ocurre si ante un fallo de página no existe un marco libre en memoria principal?**
- Posibles soluciones que aplicaría el S.O. :
  - Abortar el proceso de usuario (no es una buena solución).
  - Descargar otro proceso y llevarlo a almacenamiento secundario liberando sus marcos (se puede hacer).
  - *Reemplazar páginas:*
    - Encontramos un marco que no se esté "utilizando" y lo liberamos.

Reemplazo de páginas.

- **Fallo de página con reemplazo de páginas:**
  - Se busca la página deseada en almacenamiento secundario.
  - Se busca un marco libre.
    - LO HAY: lo utilizamos.
    - NO LO HAY: ***reemplazo de página***
      - usar un algoritmo de reemplazo de páginas para seleccionar un marco víctima que genere el menor número de fallos de página
      - Pasamos el contenido del marco a almacenamiento secundario.
      - Actualizamos la tabla de páginas.
  - Ya disponemos de un marco libre. Se lee la página de almacenamiento secundario y se copia en el marco libre.
  - Se actualiza la tabla de páginas.
  - Se reinicia la instrucción interrumpida.

Rendimiento

- **Frecuencia de Fallo de página**
  - Sea **p** la probabilidad de que una referencia a memoria provoque un fallo de página (0<p<1)
    - Si p=0, nunca hay fallos de página
    - Si p=1, hay fallo de página en todas las referencias
  - Sea $\mathbf{t_m}$ el tiempo de acceso a memoria principal
  - Sea $\mathbf{t_{fp}}$ el tiempo para resolver un fallo de página, que depende de:
    - Tiempo de transferencia entre almacenamiento secundario y memoria
    - Tiempo de actualización de tablas de páginas
    - Tiempo de reinicio de la ejecución del proceso
  - El tiempo efectivo de acceso a memoria **TEAM** vendrá dado por:

$$TEAM=(1-p).t_m + p.t_{fp}$$

- Objetivo de cualquier algoritmo de reemplazo:

  Obtener la menor tasa de fallos de página posible

Reemplazo de páginas.

Reducción del tiempo para resolver los fallos de página

- Fallo de página: dos accesos a almacenamiento secundario
  - Uno para guardar la página víctima
  - Otro para cargar la nueva página
- Usar ***bit de modificado*** en la tabla de páginas
  - Al cargar la página, desde almacenamiento secundario a memoria, el bit modificado se pone a 0 (no modificada)
  - Si se escribe en la página el bit pasa a 1 (modificado)
  - Si la página es elegida como víctima se mira su bit de modificado
    - Si la página no ha sido modificada (bit a cero) no habrá que salvarla
    - Si la página ha sido modificada (bit a uno) se salvará
- El uso de bit modificado reduce el tiempo $\mathbf{t_{fp}}$

Reemplazo de páginas.

- La tasa de fallos de página (p) dependerá de:
  - Número de páginas de los procesos
  - Número de procesos en memoria
  - Número de marcos disponibles
  - Del algoritmo de reemplazo de páginas que se utilice
    - Hay que usar aquel que conlleve menor número de fallos
- Para poder implementar un sistema de memoria virtual nos queda por responder a dos preguntas:
  - ***¿Cómo reemplazar las páginas?***
    - Es necesario escoger un algoritmo de reemplazo de páginas
  - ***¿Cómo decidir cuantos marcos de cada proceso tenemos en memoria?***

Algoritmos de reemplazo de página.

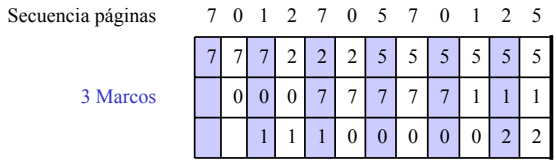
- Clasificación de estrategias de reemplazo:
  - Reemplazo Global
    - Utilizan los algoritmos de reemplazo de páginas actuando sobre las páginas de todos los procesos
  - Reemplazo Local
    - Usa los algoritmos sólo entre las páginas del proceso que necesita un reemplazo de página
- Algoritmos de reemplazo de páginas:
  - ***FIFO***
  - ***Óptimo***
  - ***LRU (Last Recently Used)***
  - ***De la segunda oportunidad o del reloj***
  - ***Con bits referenciado y modificado***

Algoritmos FIFO.

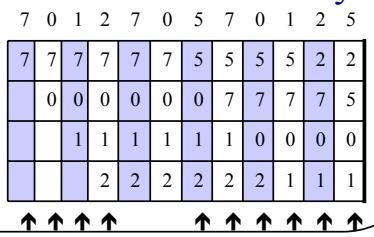
- Se reemplaza la página que lleva más tiempo en memoria
- El SO mantiene una lista de las páginas
  - Se reemplaza la página cabecera de la lista y se inserta al final
- El rendimiento no siempre es bueno, pueden sustituirse páginas muy usadas
- Puede presentarse la anomalía de Belady: más marcos en memoria no implica que hayan menos fallos de página

Ejemplo: Sea la secuencia: 7, 0, 1, 2, 7, 0, 5, 7, 0, 1, 2, 5

Con 3 marcos, 9 fallos de página, con 4 hay 10 fallos

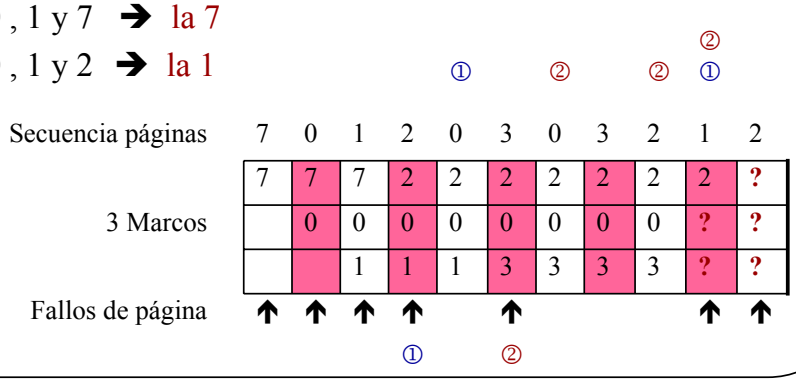


Anomalía de Belady



Algoritmos Óptimo.

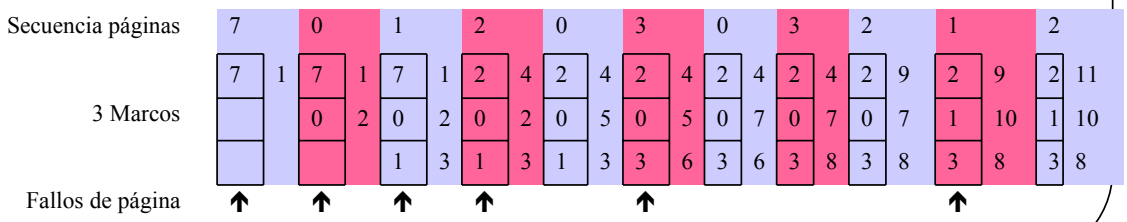
- El algoritmo óptimo tiene la menor tasa de fallos
- Reemplazar la página que no se va a usar durante más tiempo
- Es irrealizable ya que no se conoce a priori la utilización de memoria de instrucciones futuras
- **Ejemplo:** En los fallos ① y ② hay que decidir la página:
  - ① entre la 0 , 1 y 7 ➔ la 7
  - ② entre la 0 , 1 y 2 ➔ la 1



Algoritmos LRU (Last Recently Used).

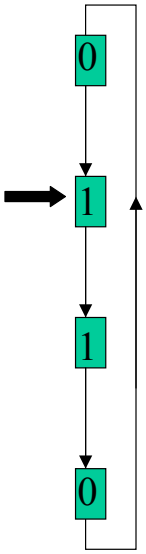
- Sustituye la página que más tiempo lleva sin ser usada
- Implantación mediante un contador:
  - Cada vez que accedemos a memoria se incrementa su valor
  - Se copia el valor del contador en la tabla de páginas asociado a la página a la que hemos accedido
    - Se elimina la página que tiene el valor del contador más bajo
- Implementación mediante pila:
  - En la base, la página que lleva más tiempo, en la cima la más nueva

Ejemplo:



Algoritmos de la segunda oportunidad o del reloj.

- Utiliza un bit de referencia asociado a cada página
  - Inicialmente están a cero
  - Cambia a 1 cuando se accede a la página para leer o escribir
  - El SO pone periódicamente todos a cero
- Funciona como FIFO pero:
  - Selecciona una página y examina su bit referenciado
    - Está a cero, reemplazamos la página
    - Está a uno, se da una segunda oportunidad a la página
      - Se pone el bit referenciado a cero
      - Buscamos la siguiente víctima
- **¿Cómo implementarlo?**
  - Se usa una cola circular de las páginas residentes en memoria
  - Un puntero indica la siguiente página a reemplazar de la cola
    - Si bit referenciado = 1, lo ponemos a cero y avanzamos el puntero
    - Si bit referenciado =0, sustituimos esa página



Algoritmos con bits referenciado y modificado.

- Utiliza en cada página *un bit referenciado* (1 indica página accedida) y *un bit modificado* (1 indica acceso de escritura)
- Las páginas agrupadas en cuatro clases:

  ① Referenciado = 0, Modificado = 0
  ② Referenciado = 0, Modificado = 1
  ③ Referenciado = 1, Modificado = 0
  ④ Referenciado = 1, Modificado = 1

- Se reemplazará la página de la clase inferior (número más bajo) no vacía
  - Si hay varias en esa clase se utilizará FIFO para la selección
- El SO pone periódicamente a cero el bit referenciado

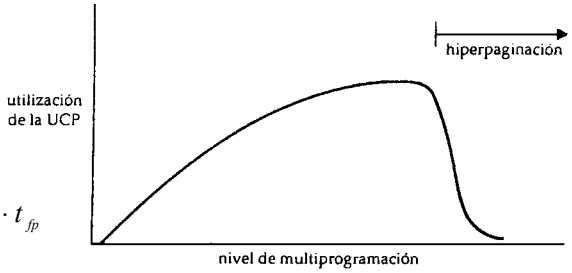
Políticas de asignación de marcos.

¿Cuántos marcos se asignan a cada proceso en un sistema multiprogramado?

- Estrategias de asignación
  - **Asignación fija**
    - Los marcos de memoria existentes se dividen:
      - Equitativamente: nº marcos/nº procesos
      - Proporcionalmente: se asignan más marcos a procesos más grandes
    - Suele estar asociada a una estrategia de reemplazo local
  - **Asignación dinámica**
    - La cantidad de marcos de cada proceso varía dinámicamente según la necesidades del mismo
    - Puede aplicarse a:
      - reemplazo local, cada proceso varia la cantidad de marcos que utiliza
      - reemplazo global, los procesos compiten por el uso de memoria

Hiperpaginación.

- Hiperpaginación (thrashing)
  - Número de fallos de página elevado debido a que el nº de marcos asignados al proceso no es suficiente para almacenar las páginas referenciadas por el mismo
  - Más tiempo en la cola de servicio de paginación que en CPU
  - El rendimiento de la CPU decrece
- Gráfica de utilización de CPU en función del grado de multiprogramación
- Solución:
  - Descargar las páginas de uno o varios procesos a almacenamiento secundario liberando marcos de memoria



$$Tiempo\ \ accesoefec\ \ t = (1 - p)\cdot t_m + p \cdot t_{fp}$$

$t_m = tiempo\ \ de\ acceso\ \ a\ memoria\ \ .$

$t_{fp} = tiempo\ \ de\ \ fallo\ \ de\ pagina\ \ .$

$p = probabilid\ \ ad\ de\ \ fallo\ \ de\ pagina\ \ .$