

Convivencia

Gestión de la Memoria



Dra. Carolina Mañoso
Dpto. Informática y Automática.UNED

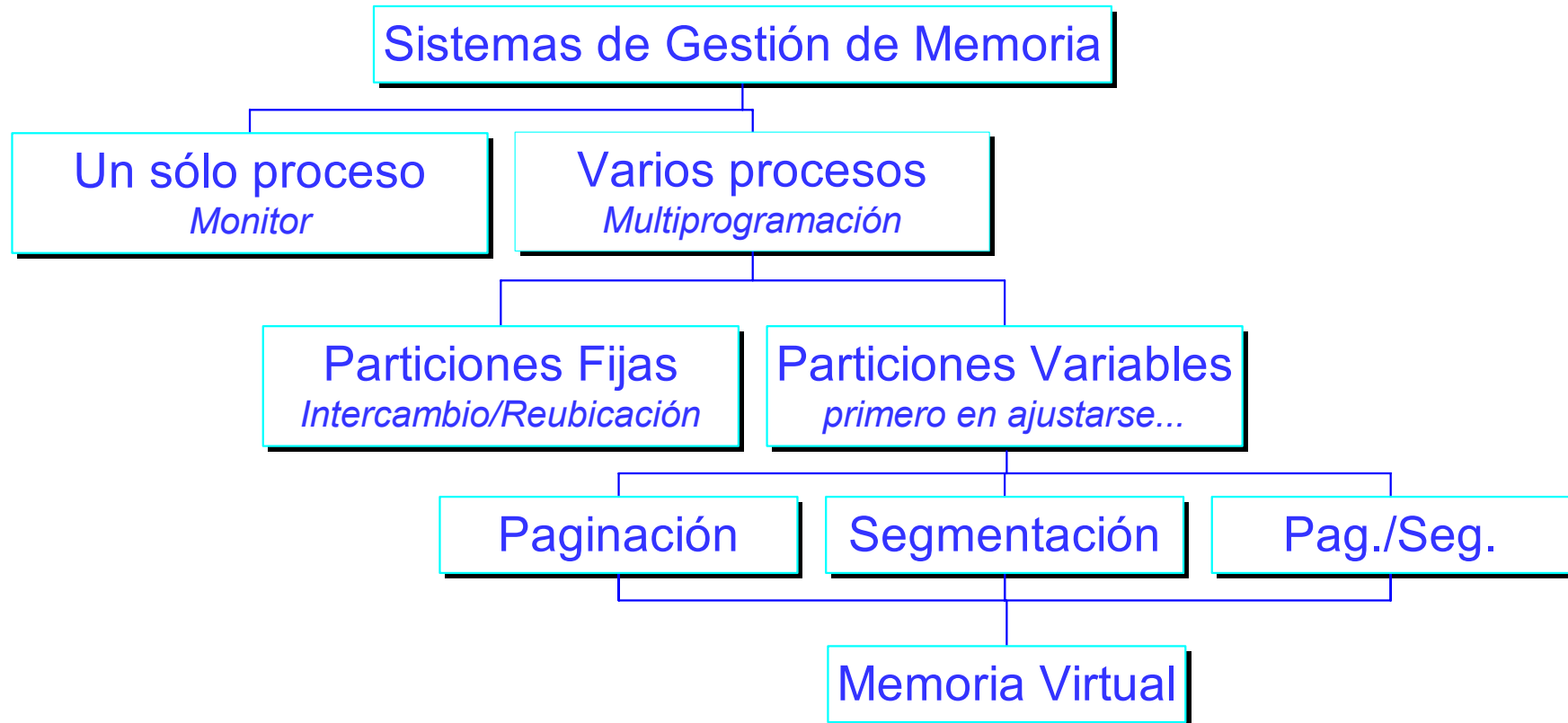
© Carolina Mañoso, 2002

Introducción (1/2)

- Para que un proceso esté preparado para ejecución debe estar cargado en memoria principal
- **La misión del gestor de memoria** es la asignación de memoria principal a los procesos que la soliciten
- El espacio vacante se puede utilizar para cargar procesos que están ya preparados para su ejecución, de forma que el planificador esté en mejores condiciones de preparar las tareas que se van a ejecutar

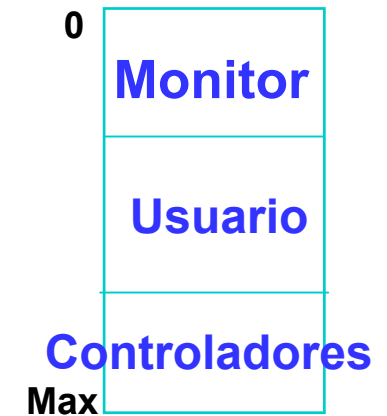
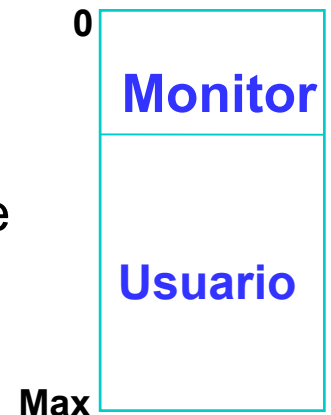


Introducción (2/2)



Gestión de memoria de un solo proceso

- Sin gestión de Memoria: El usuario se encuentra con la máquina desnuda y tiene un control completo sobre el espacio total de la memoria (hasta los 60)
- La memoria esta dividida en dos secciones:
 - ◆ Una está asignada permanentemente a la parte del s.o. que debe estar residente en memoria o monitor
 - ◆ La otra se asigna a los llamados procesos transitorios, que son cargados y ejecutados uno cada vez, en respuesta a órdenes de usuario
- ◆ La memoria esta dividida en tres secciones:
 - ◆ La parte inferior (en RAM) está asignada al s.o.
 - ◆ La parte central al único programa del usuario
 - ◆ La parte superior (en ROM, (BIOS)) a los controladores de dispositivos



Multiprogramación

- **Multiprogramación:** permite el entrelazado y el solapamiento de la ejecución de más de un programa de forma que se mantenga del modo más ocupado posible todos los recursos

Se utiliza el término *multitarea* para referirse a la capacidad que tienen los s.o. de ejecutar de forma simultánea varios procesos. El término *multiprogramación* es más general pues incluye además de la posibilidad de multitarea, la capacidad de gestión de memoria y de los ficheros



Particiones fijas (1/3)

- Consiste en divisiones de memoria que se efectúan en algún momento antes de ejecutar los programas de usuario y permanecen fijas desde entonces
- El número de particiones distintas determina el grado de multiprogramación
- Problema: la **fragmentación interna** o memoria desaprovechada dentro de una partición



Particiones fijas (2/3)

- Una vez definidas las particiones, el s.o. necesita llevar la cuenta de sus estados, libre o en uso para propósitos de asignación
- El estado y los atributos de las particiones se recogen en una estructura de datos llamada **tabla de descripción de particiones (TDP)**.
- Cada partición está descrita por su dirección inicial (**base**), su tamaño y su estado. Los campos de la base y tamaño son fijos



Particiones fijas (3/3)

- Otra organización posible es asignar una cola de tareas a cada partición en memoria y las tareas se incluyen en la cola de la partición de memoria correspondiente a sus exigencias de memoria
- Estrategias de asignación de particiones:
 - ◆ Primer ajuste
 - ◆ Mejor ajuste



Intercambio o sawpping (1/2)

- Corresponde a las operaciones de eliminar de memoria principal los procesos suspendidos, llevarlos al disco y cargar del disco a la memoria principal procesos para su ejecución

- Las funciones a realizar son:
 - ◆ La selección de los procesos que hay que eliminar de la memoria principal
 - ◆ La selección de los procesos que hay que cargar en memoria principal
 - ◆ La asignación y gestión del espacio de intercambio



Intercambio o sawpping (2/2)

- El intercambiador efectúa la mayoría de las funciones del planificador a medio plazo
- Se corre el peligro del **trasiego o thrashing** provocado por la retirada repetida de procesos de memoria casi inmediatamente después de ser cargados en la memoria
- Hay que disponer de un **archivo de intercambio** para almacenar la imagen dinámica del proceso retirado, así se puede tener:
 - ◆ Archivo de intercambio **global** del sistema
 - ◆ Archivos de intercambio **dedicados**, uno por cada proceso



Reubicación

- Se refiere a la capacidad de cargar y ejecutar un programa determinado en una posición arbitraria de memoria, en contraposición a un número fijo de posiciones especificadas en el momento de la traducción del programa
- Dependiendo de cuándo y cómo tenga lugar la traducción del espacio de direcciones virtuales al espacio de direcciones
 - ◆ Estática
 - ◆ Dinámica: se implementa mediante **registros base** especializados



Protección

- En sistemas que utilizan registros base para la reubicación es habitual utilizar **registros límite** para la protección
- Los valores base y límite de cada proceso se guardan en su BCP
- Otro método de protección es registrar los derechos de acceso en la propia memoria, asociando unos cuantos bits de protección con grandes bloques de memoria llamados **clave**



Fragmentación de Memoria

- Existen dos tipos de desaprovechamiento de la memoria:
 - ◆ **Fragmentación interna**: consiste en aquella parte de la memoria que no se está usando porque es interna a una partición asignada a una tarea
 - ◆ **Fragmentación externa**: ocurre cuando una partición disponible no se emplea porque es muy pequeña para cualquiera de las tareas que esperan
- La selección de los tamaños de las particiones es un compromiso entre estos dos casos de fragmentación



Ejercicio 1

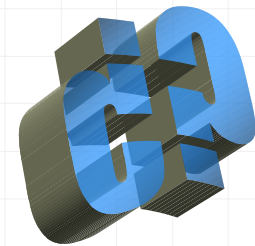
Práctica de particiones fijas

Considerar un sistema con intercambio, en el que la memoria posee particiones libres de tamaño fijo: 1000Kb, 400Kb, 1800Kb, 700Kb, 900Kb, 1200Kb y 1500Kb. Estos huecos están dispuestos en el orden dado. Se tienen tres procesos de tamaños 1200Kb, 1000Kb y 900Kb. Para los algoritmos:

- Primero en ajustarse
- Mejor en ajustarse
- Peor en ajustarse
- Siguiente en ajustarse

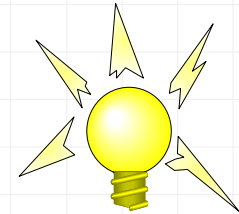
a) ¿Qué huecos serán asignados?

b) ¿Qué algoritmo aprovecha mejor la memoria?



Solución 1 (1/4)

Práctica de particiones fijas



a) Para el algoritmo primero en ajustarse:

<u>Segmento</u>	<u>Hueco asignado</u>	<u>Fragmentación</u>
1200Kb	1800	600
1000Kb	1000	0
900Kb	900	0

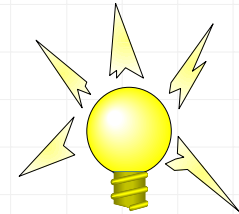
Para el algoritmo mejor ajustarse:

<u>Segmento</u>	<u>Hueco asignado</u>	<u>Fragmentación</u>
1200Kb	1200	0
1000Kb	1000	0
900Kb	900	0



Solución 1 (2/4)

Práctica de particiones fijas



Para el algoritmo peor en ajustarse:

<u>Segmento</u>	<u>Hueco asignado</u>	<u>Fragmentación</u>
1200Kb	1800	600
1000Kb	1500	500
900Kb	1200	300

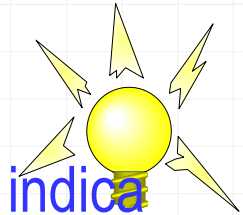
Para el algoritmo siguiente en ajustarse:

<u>Segmento</u>	<u>Hueco asignado</u>	<u>Fragmentación</u>
1200Kb	1800	600
1000Kb	1200	200
900Kb	1500	600



Solución 1 (3/4)

Práctica de particiones fijas

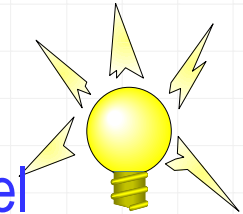


Si el sistema de gestión de memoria es de particiones **fijas** (como indica el enunciado) el problema es la fragmentación interna. En este caso el algoritmo que mejor aprovecha la memoria es el *mejor en ajustarse* dado que la fragmentación interna es nula, después el algoritmo *primero en ajustarse* con fragmentación interna de 600 y finalmente los otros dos que producen una fragmentación interna de 1400



Solución 1 (4/4)

Práctica de particiones fijas



Sin embargo, si las particiones libres dadas se corresponden con el estado de la memoria en un instante dado de una **gestión de memoria con particiones variables** entonces se estará hablando de fragmentación externa que en esta caso no tiene lugar, dado que los nuevos bloques libres son lo suficientemente grandes para alojar a otros procesos que vengan posteriormente.

En cualquier caso y a la vista de estos resultados se puede concluir que el algoritmo que mejor aprovecha la memoria es el *mejor en ajustarse* porque se ajusta exactamente a los bloques libres dados y esto impide que se creen bloques pequeños no utilizables como era de esperar en este tipo de algoritmo



Particiones dinámicas (1/2)

- Cuando se pide que se cargue una imagen de un proceso en memoria, el módulo de gestión intenta crear una partición adecuada que asignar al proceso en cuestión, para ello es preciso localizar un área libre de memoria que sea igual o mayor que el proceso, si es así se fabrica la partición
- La gestión de la memoria con particiones libres plantea el problema de mantenimiento de un registro de particiones libres y ocupadas que sea eficiente, tanto en tiempo para la asignación como para el aprovechamiento de la memoria. Formas de mantener este registro:
 - ◆ Mapa de bits
 - ◆ Listas enlazadas para las particiones libres y asignadas
 - ◆ El sistema de los asociados



Particiones dinámicas (2/2)

- Los algoritmos más habituales para la selección de un área libre de memoria a la hora de la creación de una partición son:
 - ◆ Primer ajuste
 - ◆ Siguiente ajuste
 - ◆ Mejor ajuste
 - ◆ Peor ajuste
- Problema la fragmentación externa o memoria desaprovechada entre particiones
- La diferencia esencial con la gestión estática, es que las entradas a la TDP se utilizan sólo para definición de las particiones creadas y asignadas y las áreas libres se describen en la lista de memoria libre



Compactación

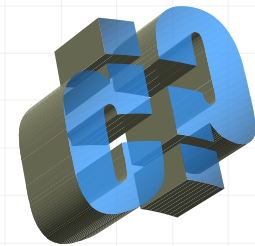
- Si la memoria resulta seriamente fragmentada, la única salida posible es reubicar algunas o todas las particiones en un extremo de la memoria y así combinar los huecos para formar una única área libre grande a este proceso se le llama **compactación**
- Como los procesos afectados deben de ser suspendidos y copiados realmente de un área de memoria a otra, es importante decidir cuando debe realizarse la compactación



Ejercicio 2

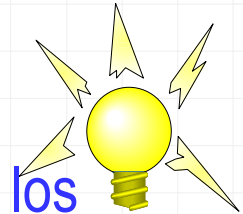
Práctica de compactación

Calcular el porcentaje de tiempo de UCP utilizado para la compactación de la memoria en una máquina de 1Mb de memoria. La compactación se hace cada 0,5 seg. y se tarda 300 nseg. en copiar un byte. Los espacios desaprovechados en promedio son del 50% del tamaño de los bloques



Solución 2 (1/2)

Práctica de compactación

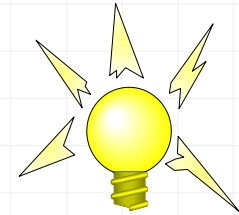


Se entiende por compactación la técnica que permite mover todos los procesos hacia la parte inferior (o superior) de la memoria de forma que es posible combinar todos los bloques libres en uno grande. Con esta técnica se palía el problema de la fragmentación externa de algunos sistemas de gestión de memoria. La cuestión que se resuelve es la siguiente: La memoria que debe ser asignada a un proceso nuevo en particular debe ser contigua, pero la memoria libre disponible se halla dispersa debido a la fragmentación externa por lo que no se puede usar. En cualquier caso la solución de la compactación resulta ser un método de muy alto coste



Solución 2 (2/2)

Práctica de compactación



El espacio desaprovechado es del 50%, ello indica que se deben trasladar a un extremo de la memoria el 50% de la misma, es decir 0,5Mb. Por lo tanto, el número de bytes que se debe trasladar es:

$$M = (1024 \times 1024) \times 0,5 = 524288 \text{ bytes}$$

El tiempo total para el traslado es:

$$T_{CPU} = 524288 \times 300 \cdot 10^{-9} = 0,1573 \text{ seg}$$

Si cada 0,5 seg se pasa 0,1573 seg compactando, entonces cada 1 seg estará 0,3146 seg, por lo que el 31,46% del tiempo de CPU se está desaprovechado en tiempos de compactación. Obsérvese el elevado tiempo que se utiliza para las tareas de compactación como se comentaba al inicio del problema



Paginación (1/4)

- Se suprime el requisito de la asignación contigua de memoria física a un programa
- La memoria física se divide conceptualmente en una serie de porciones de tamaño fijo, llamadas **marcos de página**
- El espacio de direcciones virtuales de un proceso se divide además en bloques de tamaño fijo del mismo tamaño llamados **páginas**
- La asignación de memoria consiste en hallar un número suficiente de marcos de página sin utilizar para cargar en ellos las páginas del proceso solicitante

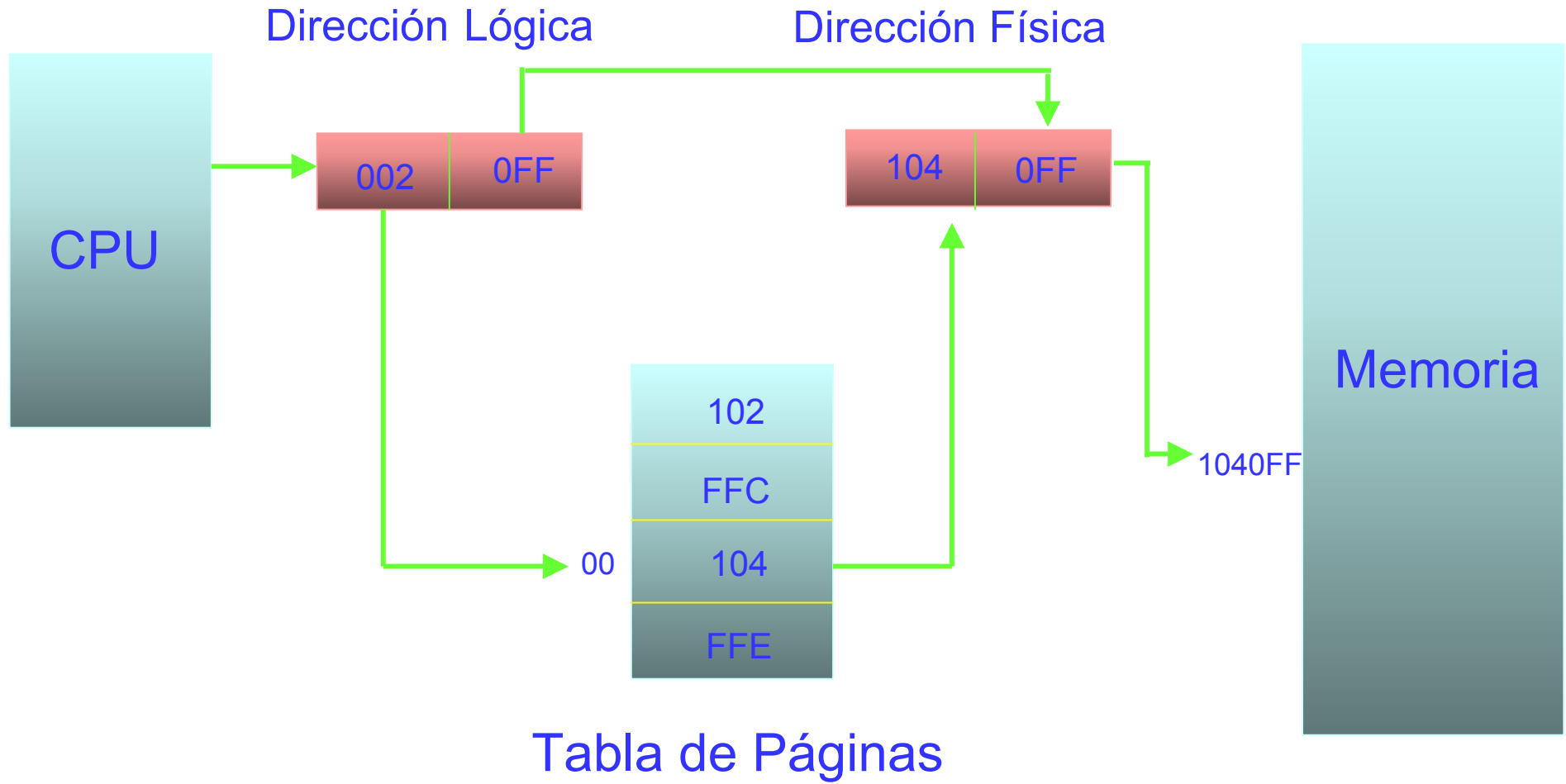


Paginación (2/4)

- Puesto que cada página se hace corresponder separadamente, los diferentes marcos de página asignados a un solo proceso no necesitan ocupar áreas contiguas de memoria física
- Cada dirección virtual esta dividida en dos partes:
 - ◆ El número de página
 - ◆ El desplazamiento dentro de esa página
- Para efectuar la traducción de las direcciones, el número de página se emplea como un índice en la tabla de páginas y la dirección física se



Paginación (3/4)



Paginación (4/4)

- Si el tamaño de un proceso dado no es múltiplo entero del tamaño de la página, el último marco de página puede estar parcialmente inutilizado, esto se llama **fragmentación o ruptura de página**
- Para acelerar el proceso de traducción de direcciones:
 - ◆ La utilización de registros específicos para la tabla de páginas es adecuada si la tabla de páginas es pequeña.
 - ◆ Un método comúnmente utilizado es utilizar una memoria asociativa de alta velocidad para almacenar un subconjunto de las entradas de la tabla del mapa de páginas más frecuentemente utilizadas. Esta memoria se denomina **buffer para apartado de traducciones (BAT)**

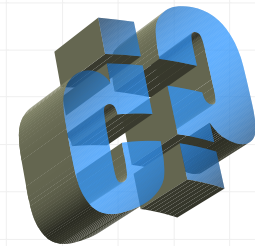


Ejercicio 3 (1/2)

Práctica de paginación

Supóngase que la tabla de páginas de un proceso que se está ejecutando en el procesador es:

Nº de Pág.	Nº de Marco de Pág.	R	M	V
0	4	1	0	1
1	7	1	1	1
2	---	0	0	0
3	2	0	0	1
4	---	0	0	0
5	0	0	1	1



Ejercicio 3 (2/2)

Práctica de paginación

R: es el bit de referencia. $R=1$ (la página ha sido referenciada)

M: es el bit de modificación. $M=1$ (la página ha sido modificada)

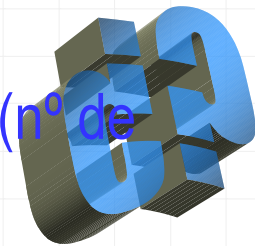
V: es el bit de Presente/ausente. $V=1$ (la página en cuestión está en memoria principal, tiene un marco asociado)

El tamaño de las páginas es de 1kb. El marco 0 está cargado en la dirección física cero y el resto sucesivamente

Se pide: ¿A qué direcciones físicas corresponden las siguientes direcciones virtuales?

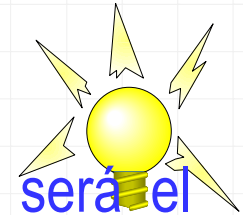
a). (1, 125) b). (2, 324) c). (5, 322) d). (7, 321) e). (3, 1026)

El formato en el que se da la dirección virtual corresponde a (nº de página, desplazamiento)



Solución 3 (1/2)

Práctica de paginación



La dirección física, de acuerdo con el sistema de paginación, será el número de marco multiplicado por el tamaño de marco más el desplazamiento dentro de él

$$\text{Dirección Física} = N^{\circ} \text{ de Marco} \times \text{Tamaño de Marco} + \text{Desplazamiento}$$

Se ha de tener en cuenta, que:

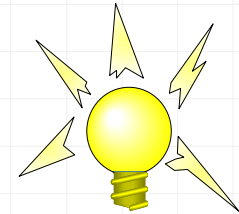
- Exista la página
- El desplazamiento no supere el tamaño del marco

Sino en ambos casos se tendrá un error de direccionamiento; Además la página ha de tener un marco asociado, sino tendrá lugar un fallo de página



Solución 3 (2/2)

Práctica de paginación



- a) Dirección_física=7*1024+125=7293
- b) La página 2 no tiene ningún marco asociado, por lo tanto, tiene lugar un **fallo de página**.
- c) Dirección _física=0*1024+322=311
- d) **Error de direccionamiento** porque no existe la página 7.
- e) **Error de direccionamiento** porque $1026 > 1024$, el desplazamiento es mayor que el tamaño del marco



Segmentación (1/9)

- La idea es dividir el espacio de direcciones de un solo proceso en bloques que puedan ser colocados en áreas no contiguas de memoria
- Los segmentos se forman en tiempo de traducción del programa mediante la agrupación de elementos relacionados lógicamente por ejemplo pila, datos, código
- El programa de usuario se compila y el compilador construye automáticamente los segmentos de acuerdo a la estructura del programa



Segmentación (2/9)

- El cargador tomará todos estos segmentos y le asignará números de segmento. Por propósito de reubicación cada uno de los segmentos se compila comenzando por su propia dirección lógica 0
- Por lo tanto, las direcciones de los procesos segmentados tienen dos componentes:
 - ◆ El **nombre** (número) de segmento
 - ◆ El **desplazamiento** dentro del segmento



Segmentación (3/9)

- Utilizando una lógica similar a la utilizada para las particiones dinámicas, se puede crear una partición separada, ajustada a las necesidades de cada segmento particular
- La **base** (correspondiente a la dirección física en que comienza el segmento, obtenida durante la creación de la partición) y el **tamaño** de cada segmento se anotan en el **descriptor de segmento**
- Todos los descriptores de segmento se recogen en una tabla llamada **tabla de segmentos (TDS)**



Segmentación (4/9)

- El número de segmento, primer parámetro de la dirección lógica, se emplea como índice dentro de la tabla de segmentos. La dirección física se obtiene añadiendo el desplazamiento a la base del segmento
- Este desplazamiento no debe sobrepasar el tamaño máximo del segmento, si se supera el tamaño máximo registrado en la tabla se produciría un error de direccionamiento



Segmentación (5/9)

- Una tabla de segmentos si se mantiene en registros especiales puede ser referenciada muy rápidamente; la suma a la base y la comparación con el tamaño se puede hacer simultáneamente
- Este método se puede llevar a cabo mientras el número de segmentos no sea grande, pero cuando éste crece la tabla de segmentos se deja en memoria
- En este caso el acceso a esta tabla se ve generalmente facilitado por medio de un registro hardware dedicado llamado **registro base de la tabla de segmento (RBTS)** y **registro límite de la tabla de segmentos (RLTS)**

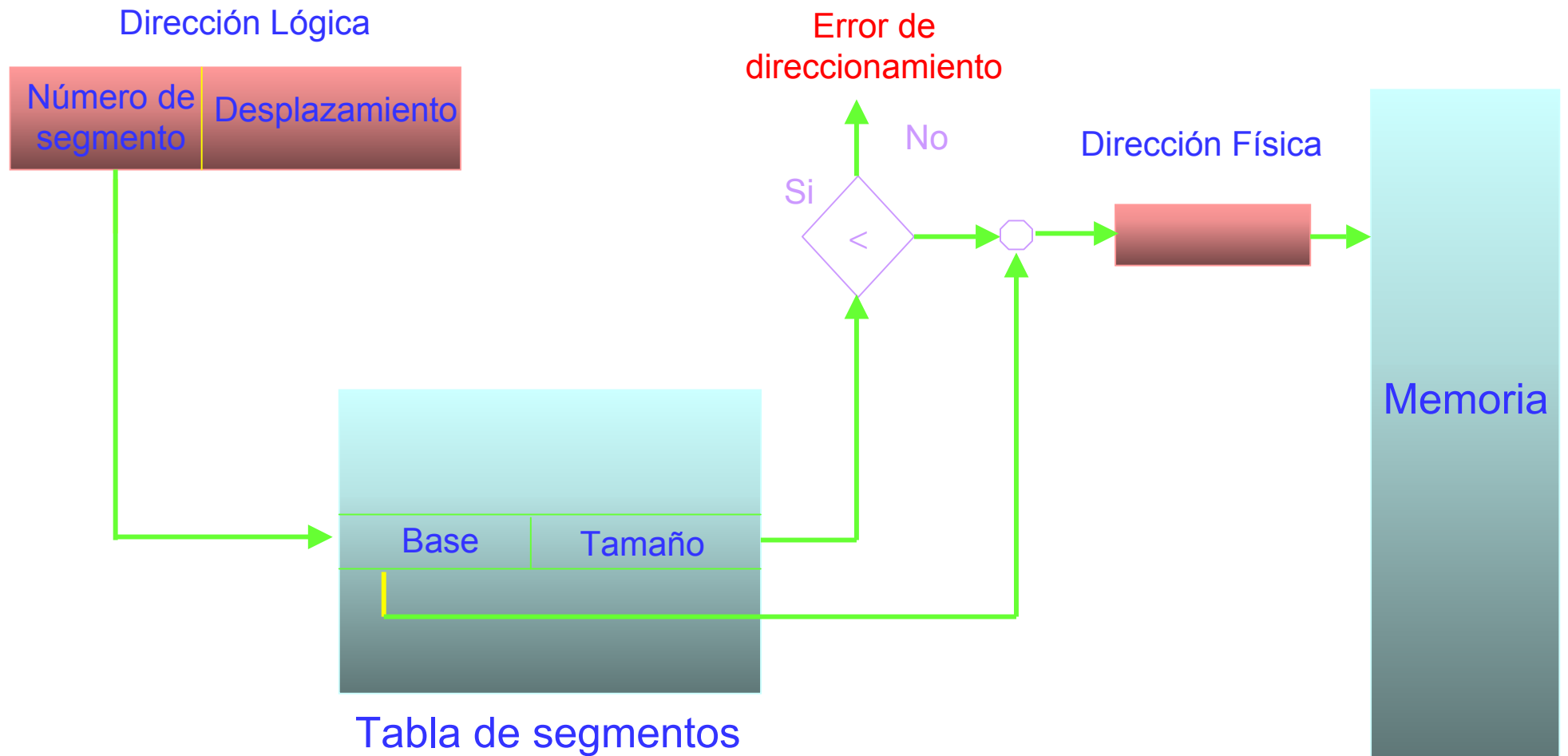


Segmentación (6/9)

- Para una dirección lógica dada de un segmento se procede:
 - ◆ Se comprueba que el número de segmento es correcto (es decir, menor que el registro de la longitud de la tabla de segmentos)
 - ◆ 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
 - ◆ Esta entrada se lee de memoria y se procede como antes, es decir, se 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



Segmentación (7/9)



Segmentación (8/9)

- Para acelerar el proceso de traducción de direcciones, se puede mantener unos pocos descriptores de segmentos, más frecuentemente usados en registros, llamados **registros descriptores de segmentos**
- Desde el punto de vista del s.o. la segmentación es esencialmente una versión con múltiples bases y límites de la memoria particionada dinámicamente
- La posibilidad interesante en los sistemas segmentados es proporcionar protección **dentro** del espacio de direcciones de un único proceso



Segmentación (9/9)

- La segmentación 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
- La segmentación no produce fragmentación interna, pero si externa, la cual ocurre cuando todos los bloques de memoria libre son muy pequeños para acomodar a un segmento. En esta situación, el proceso sólo puede esperar a disponer de más memoria, huecos más grandes o usar estrategias de compactación para crear huecos más grandes



Segmentación y paginación (1/3)

- Corresponde a la gestión de memoria que combina ambos enfoques con el fin de mantener las ventajas de cada uno
- Así se mantiene la segmentación desde el punto de vista del usuario y se divide cada segmento en páginas de tamaño fijo
- La traducción de direcciones requiere tanto tablas de segmento como tablas de páginas, pero ahora 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 de la 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

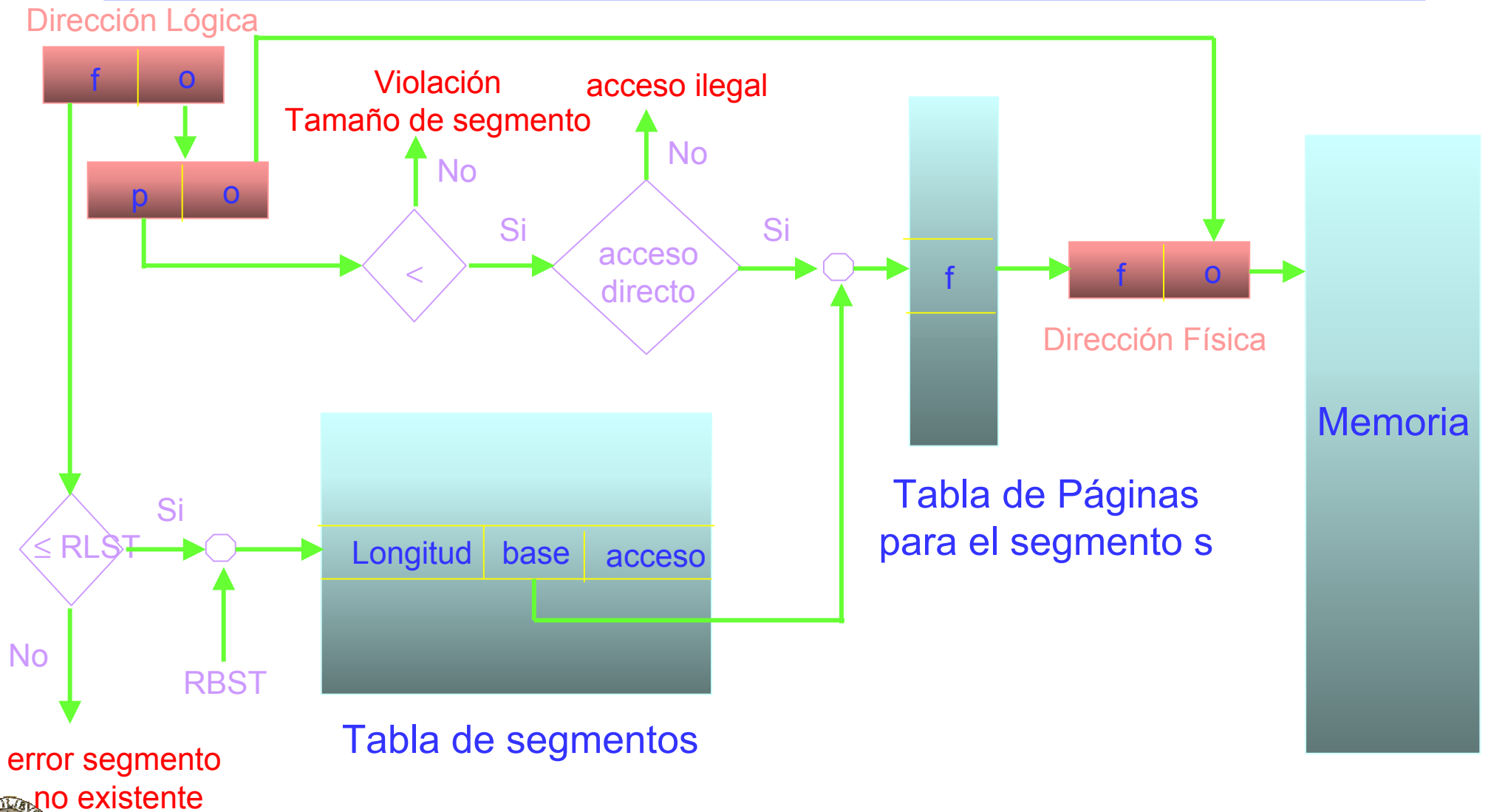


Segmentación y paginación (2/3)

- Las direcciones virtuales requieren de tres campos:
 - ◆ El número de segmento
 - ◆ El número de página
 - ◆ El desplazamiento dentro de la página
- En este esquema, cada segmento tiene su propia tabla de páginas. Sin embargo, cada segmento tiene una longitud, dada en la tabla de segmentos, de forma que el tamaño de la tabla de páginas es función de la longitud del segmento
- Se ha eliminado la fragmentación externa, pero introduciendo fragmentación interna y aumentando el espacio de tablas



Segmentación y paginación (3/3)



Ejercicio 4

Práctica de segmentación y paginación

Se tiene un sistema operativo multitarea con las siguientes características:

- Gestión de memoria virtual formada por la combinación de la segmentación y la paginación.
- Direcciones virtuales de 32 bits (igual que las direcciones físicas) con el siguiente formato:

<u>8 bits</u>	<u>12 bits</u>	<u>12 bits</u>
nº de segmento	nº de página	Desplazamiento

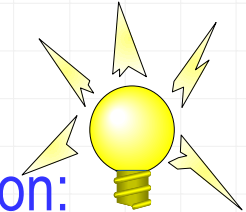
a) ¿Cuántos segmentos distintos puede direccionar cualquier proceso del sistema?

b) ¿Cuántos marcos de página distintos puede haber como máximo?



Solución 4

Práctica de segmentación y paginación



a) Los segmentos distintos que se pueden direccionar con 8 bits son:

$$2^8 = 256$$

se obtiene del campo que indica el número de segmento en la dirección virtual.

b) El marco de página es del mismo tamaño que la página por lo tanto es de igual tamaño que el campo del desplazamiento en la dirección virtual 2^{12} . La memoria real tiene un tamaño de 2^{32} por lo tanto el número de marcos vendrá dado:

$$2^{32} / 2^{12} = 2^{20} \text{ marcos de página diferentes en memoria física.}$$



Memoria virtual (1/7)

- Es un esquema de gestión de memoria en donde puede que sólo una parte del espacio de direcciones virtuales de un proceso residente sea cargada realmente en memoria física
- En otras palabras la memoria virtual permite la ejecución de procesos parcialmente cargados
- Esto se consigue manteniendo una imagen del espacio de direcciones virtuales completo de un proceso en memoria secundaria, y trayendo a memoria principal parte de esa imagen cuando sea necesaria. La elección de que sección traer, cuando y donde es efectuada por el s.o.



Memoria virtual (2/7)

- La memoria virtual puede implementarse como extensión de la gestión de memoria paginada, segmentada o como una combinación de ambas
- La tarea adicional del hardware de traducción de direcciones en sistemas virtuales es detectar si el elemento referenciado esta en memoria real o no. Para ello se añade **un bit de presencia**, a cada entrada de la TP (en el caso de gestión paginada)



Memoria virtual (3/7)

- En el caso de que en el momento de la traducción el bit este a cero, el hardware genera una excepción por elemento ausente para anunciar el hecho al s.o. A esta excepción se denomina **fallo de página**
- Por lo tanto el fallo de página se produce cuando en la tabla de páginas del proceso no hay asociado un marco de página a la página que se quiere acceder



Memoria virtual (4/7)

- Cuando el proceso en ejecución experimenta un fallo de página **queda suspendido** hasta que la página que falta sea incorporada en memoria principal, así el procedimiento a seguir es el siguiente:
 - ◆ Se verifica si la dirección es válida. Si la dirección fuera inválida se enviará una señal al proceso o se abortaría
 - ◆ Si es una referencia válida, pero aún no se ha traído esta página, 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 del proceso, para poder continuarlo en el mismo lugar y estado
 - ◆ Se selecciona un marco libre. Si no existe un marco libre, se tendría que ejecutar



Memoria virtual (5/7)

- ◆ 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. Mientras se carga la página, el proceso sigue suspendido y se permite ejecutar otro proceso
- ◆ Cuando se completa la lectura del disco, la tabla de páginas se actualiza para indicar que ya se dispone de la página en memoria
- ◆ La instrucción que produjo el fallo regresa al estado de comienzo y se planifica su ejecución, pudiéndose acceder a la página como si siempre hubiese estado en memoria



Memoria virtual (6/7)

- La memoria virtual puede producir la interrupción de ciertas instrucciones **durante** su ejecución. Para abordar este problema:
 - ◆ Deshacer los efectos parciales.
 - ◆ Reanudar la instrucción desde el punto exacto de interrupción
 - ◆ Preexaminar las referencias a memoria, antes de comenzar la ejecución de la instrucción



Memoria Virtual (7/7)

- Políticas de la gestión de la memoria virtual:
 - ◆ **Política de asignación:** que cantidad de memoria real se asigna a un proceso
 - ◆ **Política de acceso:** que elementos se incorporan y cuando
 - ◆ **Política de sustitución:** si no existe memoria libre cuando se debe añadir un nuevo elemento, que elemento se desaloja
 - ◆ **Política de ubicación:** donde se coloca el elemento nuevo



Política de sustitución (1/4)

- Governa la elección de la página víctima cuando se va a producir el desalojo
- Dependiendo de si ha sido modificada o no el elemento retirado de memoria puede tener que ser escrito en disco o simplemente descartado. Para ello se utiliza el **bit de modificación o de suciedad** en la TP
- Es posible elegir aleatoriamente una página para sustituirla por la solicitada, pero el rendimiento es mayor si se elige una página que no se usa



Política de sustitución (2/4)

- Se pretende que la razón de fallos de página sea lo más baja posible, por lo que se dispone de los siguientes algoritmos
 - ◆ **FIFO**: se sustituye la página que lleva más tiempo en memoria. En realidad, no es necesario guardar el tiempo de entrada, ya que se puede crear una cola, según el orden de entrada, con todas las páginas de la memoria, 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.
Sufre la **anomalía de Belady** que consiste en que aumentan los fallos de página al aumentar el número de marcos de página para asignación
 - ◆ **OPT o OPTIMO**: se sustituye la página que tardará más en volverse a utilizar. Este algoritmo es irrealizable, ya que el s.o. no sabe a priori que páginas se referenciarán en el futuro



Política de sustitución (3/4)

- ◆ **LRU** se sustituye la página que no ha sido utilizada durante un periodo mayor de tiempo. Es equivalente a aplicar el algoritmo óptimo hacia atrás en el tiempo. 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. De esta forma, en la parte superior de la pila se tiene siempre el número de la última página usada y en el fondo el de la página que hace más tiempo que se usó
 - ◆ Mediante registros contadores
- ◆ **Segunda oportunidad o Reloj** es 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 encuentra una página con un bit de referencia a cero, según avanza el puntero se ponen a cero los bits de referencia



Política de sustitución (4/4)

- La política de sustitución puede ser:
 - ◆ **Local:** la página desalojada es del proceso en cuestión
 - ◆ **Global:** la página a desalojar puede ser de cualquier proceso en memoria



Ejercicio 5

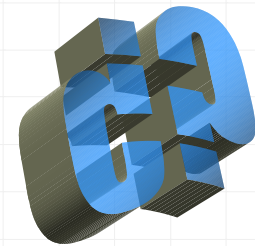
Práctica de política de sustitución

Dada la siguiente cadena de referencia:

0 1 2 3 0 1 4 0 1 2 3 4

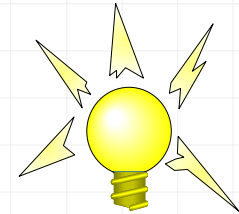
¿Cuántos fallos de página tendrán lugar si se dispone de 4 marcos de página inicialmente vacíos para el algoritmo FIFO?

¿Y si el algoritmo de sustitución fuese el LRU?



Solución 5 (1/4)

Práctica de política de sustitución



Para el algoritmo de sustitución FIFO:

0	1	2	3	3	3	4	0	1	2	3	4
x	0	1	2	2	2	3	4	0	1	2	3
x	x	0	1	1	1	2	3	4	0	1	2
x	x	x	0	0	0	1	2	3	4	0	1

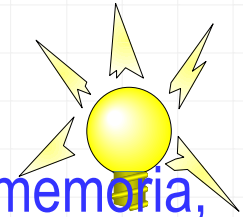
Como se deduce de la figura, inicialmente la memoria está vacía luego las cuatro primeras referencias causan fallos de página que provocan que se traigan las cuatro primeras páginas



Solución 5 (2/4)

Práctica de política de sustitución

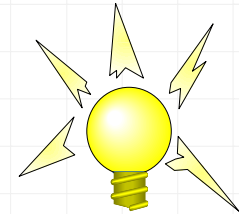
Las dos siguientes páginas que se solicitan, 0 y 1, si se hallan en memoria, por lo que no hay que hacer nada. A continuación, se solicita la página 4, que no está en memoria, produciéndose, de nuevo, un fallo de página, para traer dicha página a memoria hay que buscar, inicialmente, la página a sustituir, dicha página es la de la cabeza de la cola, por ser la más antigua, la página 0. Del mismo modo, las siguientes referencias a páginas también producen fallos de página, porque no se hallan en memoria, las páginas a sustituir serán las de la cabeza de la cola, por ser las más antiguas en memoria. Finalmente, tiene lugar 10 fallos de página



Solución 5 (3/4)

Práctica de política de sustitución

Para el algoritmo de sustitución LRU:



<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>0</u>	<u>1</u>	<u>4</u>	<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>
x	0	1	2	3	0	1	4	0	1	2	3
x	x	0	1	2	3	0	1	4	0	1	2
x	x	x	0	1	2	3	3	3	4	0	1

Igual que en el caso anterior, inicialmente la memoria está vacía, luego las cuatro primeras referencias producen fallos de página, por lo que hay que traer esas páginas de memoria



Solución 5 (4/4)

Práctica de política de sustitución



A continuación, se referencian las páginas, 0 y 1, que se hallan en memoria, luego no se producen fallos de páginas, pero se debe mantener el orden en la pila, por lo que cada vez que se referencia una página su número se elimina de la pila y se coloca en la cumbre. De esta forma, en la parte superior de la pila, se tiene siempre el número de la última página referenciada y en la parte inferior la página que hace más tiempo que se usó. Así cuando se referencia la página 4 que no se halla en memoria y por lo tanto se tiene un fallo de página, la página que se sustituye es la de la parte inferior de la pila, en este caso, la página 2. Continuando con este procedimiento finalmente tiene lugar 8 fallos de página



Política de asignación (1/3)

- La relación entre la frecuencia de fallos de página y la cantidad de memoria real asignada a un programa no es lineal corresponde a una **curva de Parocoro**
- Un problema a tener en cuenta si se asignan pocas páginas a los procesos es el de la **hiperpaginación, catástrofe o (thrashing)**
 - ◆ La explicación de esto es que con el alto grado de multiprogramación es imposible que todos los procesos mantengan suficientes páginas en memoria para que no se generen un gran número de fallos de páginas. Esto implica que se satura el canal de transferencia con el disco, que se bloquean muchos procesos esperando un intercambio de páginas y que el procesador queda infrutilizado



Política de asignación (2/3)

- Como política de asignación tenemos **la teoría del conjunto operativo**. Dicho conjunto corresponde al conjunto de páginas referenciadas por el programa durante un intervalo reciente de tiempo, dicho intervalo es una ventana cuyo tamaño depende del diseño. Además, también se utiliza como política de sustitución:

" 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"



Política de asignación (3/3)

- El conjunto operativo esta basado en **el principio de localidad**

Del estudio del comportamiento de los programas, se observa que existe una fuerte tendencia de los programas a favorecer ciertos subconjuntos de sus espacios de direcciones durante la ejecución. A este fenómeno se le conoce como **localidad de referencia**, y puede ser:

- ◆ **Localidad de referencia espacial**: tendencia a referenciar posiciones agrupadas (arrays)
 - ◆ **Localidad de referencia temporal** tendencia a referenciar la misma posición o grupo de posiciones varias veces durante breves intervalos de tiempo (bucles)
- Una **localidad** es un pequeño grupo de páginas no necesariamente adyacentes a las cuales aluden la mayoría de las referencias a memoria durante un periodo de tiempo

