ADMINISTRACIÓN DE LA MEMORIA

Introducción

- En los sistemas operativos actuales muchos procesos son ejecutados simultáneamente (multiprocesamiento / multitarea) gracias a la división del tiempo en CPU.

- La memoria principal, al igual que la CPU, es un recurso que se comparte entre el sistema operativo y los procesos

- Para que los *cambios de contexto* entre procesos sean rápidos, los mismos deben estar en la memoria principal, preparados para su ejecución

- El **Administrador de la Memoria** del sistema operativo debe permitir que los procesos convivan en la memoria de manera segura y eficiente, brindando un espacio de memoria independiente para cada uno de ellos

- El Administrador de la Memoria es parte fundamental del Sistema Operativo gestiona la memoria física empleando una serie de mecanismos y técnicas, cargando y descargando procesos hacia y desde la memoria principal

Introducción

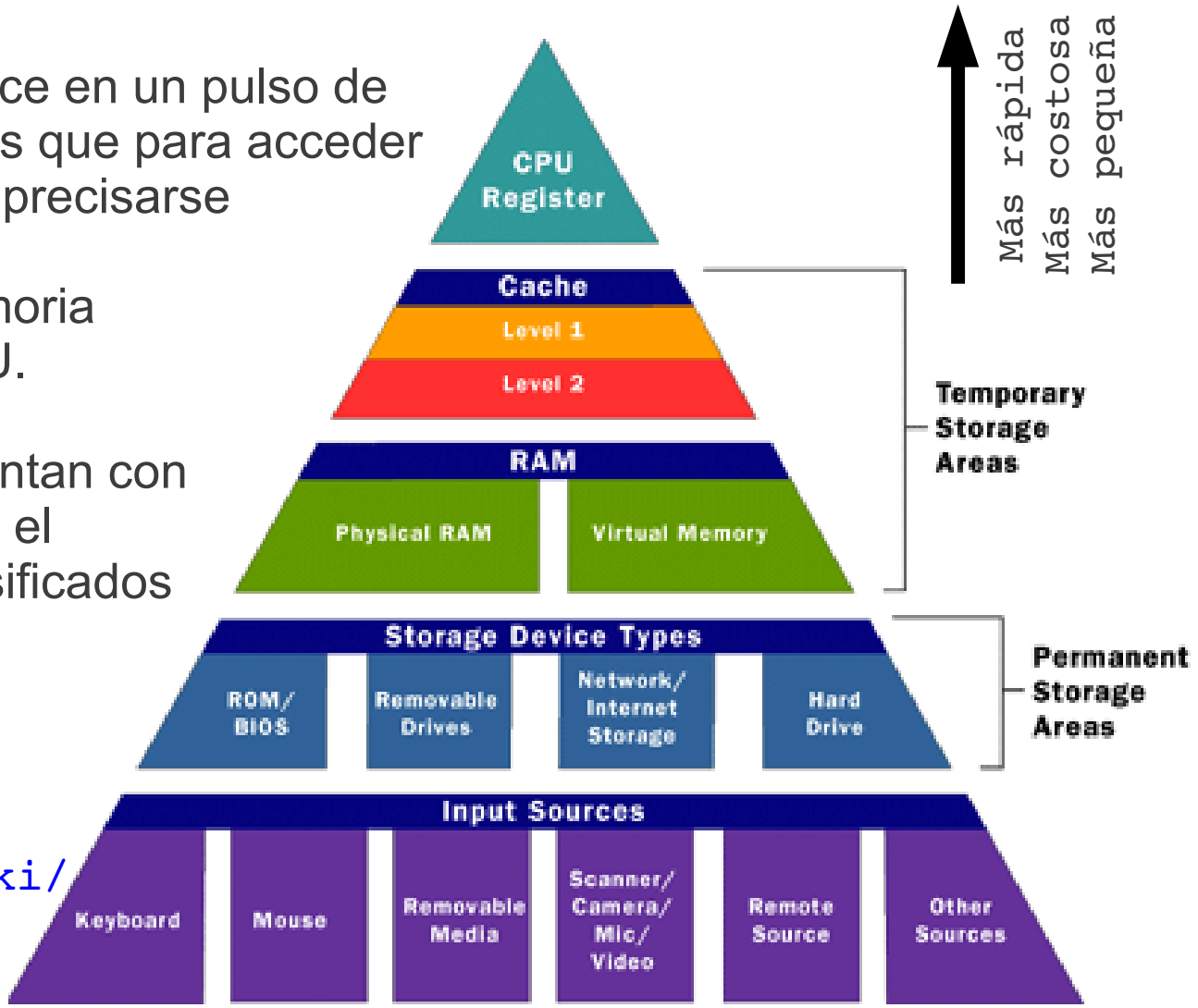
- Para que un programa se ejecute, su código y sus datos necesitan estar cargados en memoria (al menos en parte)
- También las rutinas del sistema operativo deberán residir en memoria, total o parcialmente
- Puede suceder que la memoria principal no tenga la suficiente capacidad para albergar a todos los procesos en ejecución

Jerarquía de Memoria

- La memoria principal y los registros son las únicas unidades de almacenamiento accedidas de manera directa por la CPU

- El acceso a los registros se hace en un pulso de reloj de CPU o menos, mientras que para acceder a la memoria principal podrían precisarse muchos ciclos.
  El caché se ubica entre la memoria principal y los registros de CPU.

- Los sistemas de hardware cuentan con una diversidad de medios para el almacenamiento de datos, clasificados por su velocidad de acceso y su costo/tamaño.



http://en.wikipedia.org/wiki/Memory_hierarchy

Administrador de Memoria

Un Administrador de Memoria debe

- Gestionar las áreas de memoria usada y libre
- Asignar memoria a los procesos cuando lo requieran
- Recuperar la memoria empleada por los procesos cuando finalicen
- Evitar los conflictos de uso entre procesos
- Proteger al sistema operativo
- Aprovechar eficazmente el espacio disponible:
  - Minimizar la memoria desaprovechada, evitar la fragmentación
  - Tener una complejidad temporal mínima (ejecutarse rápido o aplicar poca sobrecarga)
  - Proporcionar una buena protección y una compartición flexible

Requisitos de la Administración de Memoria

Reubicación

En un sistema multiprogramado la memoria se encuentra compartida por los procesos, por lo tanto, los procesos deben ser cargados y descargados de la memoria dado que, generalmente, no es suficiente para albergar a todos en cada momento

Protección

En un sistema con multiprogramación es necesario proteger al sistema operativo y a los otros procesos de posibles accesos que se puedan realizar a sus espacios de direcciones.

Compartición

En ciertas situaciones, bajo la supervisión y control del sistema operativo, puede ser provechoso que los procesos posean la facultad de compartir memoria.

Requisitos de la Administración de Memoria

Organización Lógica

La memoria principal y la secundaria presentan una organización física similar, como un espacio de direcciones lineal y unidimensional.

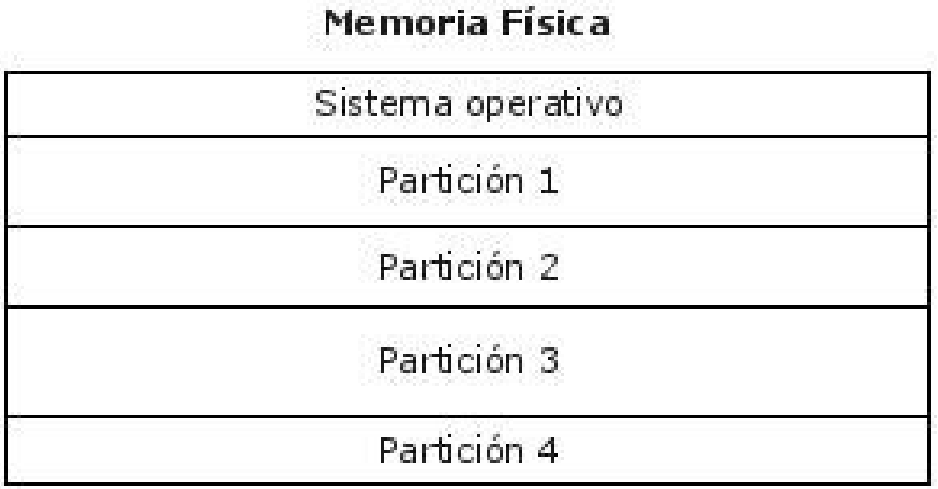
Debe existir cierta correspondencia entre el S.O. y el hardware al tratar los datos y los programas de los usuarios de acuerdo a la estructura lógica que ellos presenten.

Organización Física

Debe ser parte de la gestión de memoria, la organización del flujo de información entre la memoria principal y la memoria secundaria.

Particiones Estáticas o Fijas

- Técnica que persigue la asignación contigua de memoria
- Es la forma más simple de administración de memoria para multiprogramación
- La memoria principal se divide en particiones estáticas, definidas durante el inicio del sistema. Se dedica una partición para el Sistema Operativo y las restantes para los procesos de usuario
- Los procesos se pueden cargar en una partición de igual o mayor tamaño
- Beneficio
  Es simple de implementar e impone poca sobrecarga al sistema operativo



Particiones Estáticas o Fijas

- Las particiones físicas pueden ser implementadas con **registros de límite superior e inferior** o con **registros de base y límite**

- Registros de límite superior e inferior

- Registros de base y límite

Particiones Estáticas: **Problemas**

- Un programa puede ser demasiado grande para caber en una partición, por lo tanto si un programa no se ha diseñado mediante superposición (*overlays*) no se puede ejecutar.
  `Overlay:` http://en.wikipedia.org/wiki/Overlay_(programming)

- El número de particiones limita la cantidad de procesos activos

- Se malgasta el espacio interno de cada partición cuando el bloque cargado es más pequeño, lo que se conoce como **fragmentación interna**.
  Cualquier proceso por pequeño que sea, ocupará una partición completa.

- Se pueden usar particiones de diferentes tamaños para resolver este inconveniente

Particiones Dinámicas

- Técnica que persigue la asignación contigua de memoria
- Particiones variables en número y longitud
- Al cargar un proceso se le asigna exactamente tanta memoria como necesita

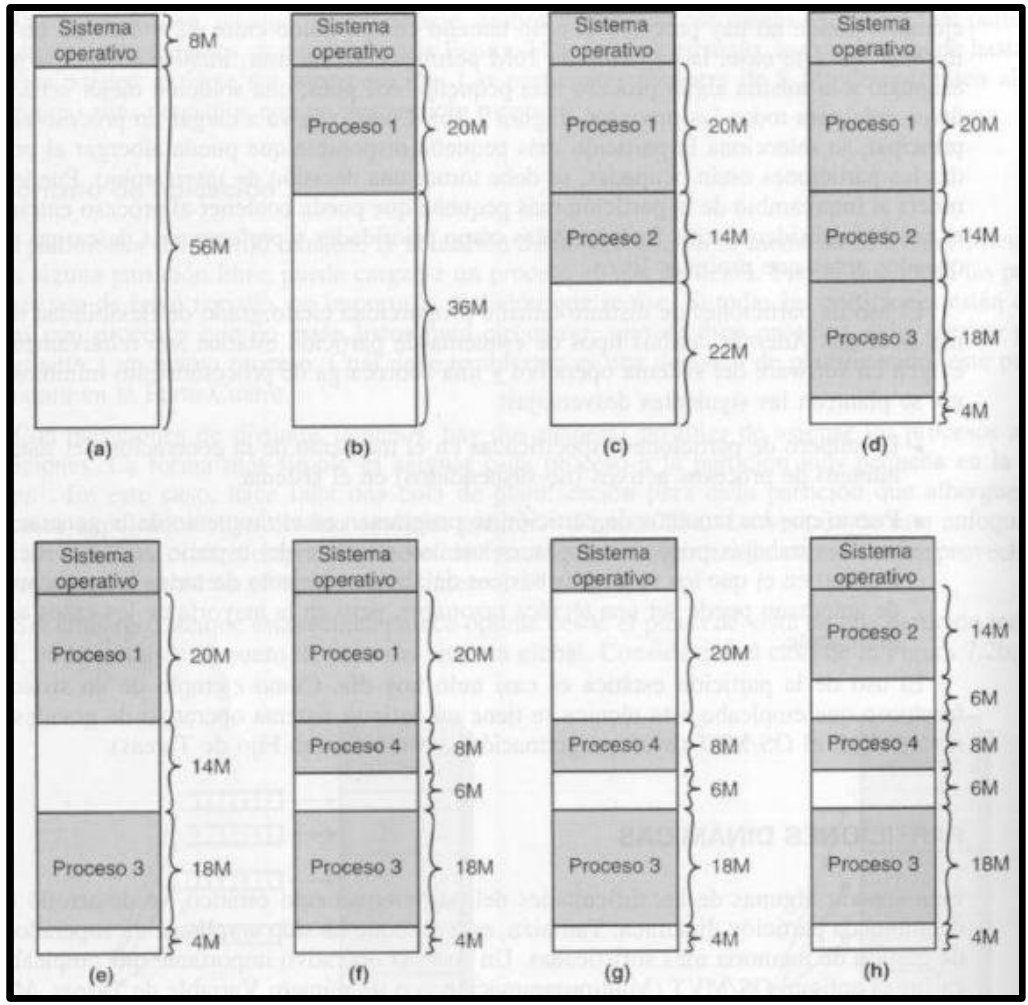
Particiones Dinámicas: Problemas

- Por la entrada y salida de procesos en la memoria, se van generando porciones cada vez más pequeñas de la memoria sin utilizar, lo que se conoce como **fragmentación externa**.

- Para solucionar este problema se debe recurrir a la **compactación** de la memoria de manera de eliminar los espacios (huecos) entre procesos.

  Esto significa que los procesos deben ser reubicados en memoria dinámicamente

  Perjuicio:
  mucha sobrecarga, uso de CPU



Particiones Dinámicas: Asignación de Memoria

- Consiste en determinar en qué hueco ubicar un nuevo proceso, intentando evitar la fragmentación externa

- Para esto existen algoritmos:

  Mejor ajuste (best-fix): consiste en ubicar el proceso en el espacio de memoria que más se ajuste a su tamaño.
  Tiene los peores resultados y genera huecos demasiado pequeños

  Primer ajuste (first-fix): consiste en ubicar el proceso en el primer hueco disponible, recorriendo desde el inicio de la memoria, cuyo tamaño sea suficiente para el proceso.
  El más sencillo y rápido, y con buenos resultados. Genera pequeñas particiones en las primeras posiciones de la memoria

  Próximo ajuste (next-fix): consiste en ubicar el siguiente hueco disponible, que sea suficientemente grande, a partir de la última asignación de memoria.
  Peores resultados que el algoritmo del primer ajuste, necesita compactación más frecuentemente

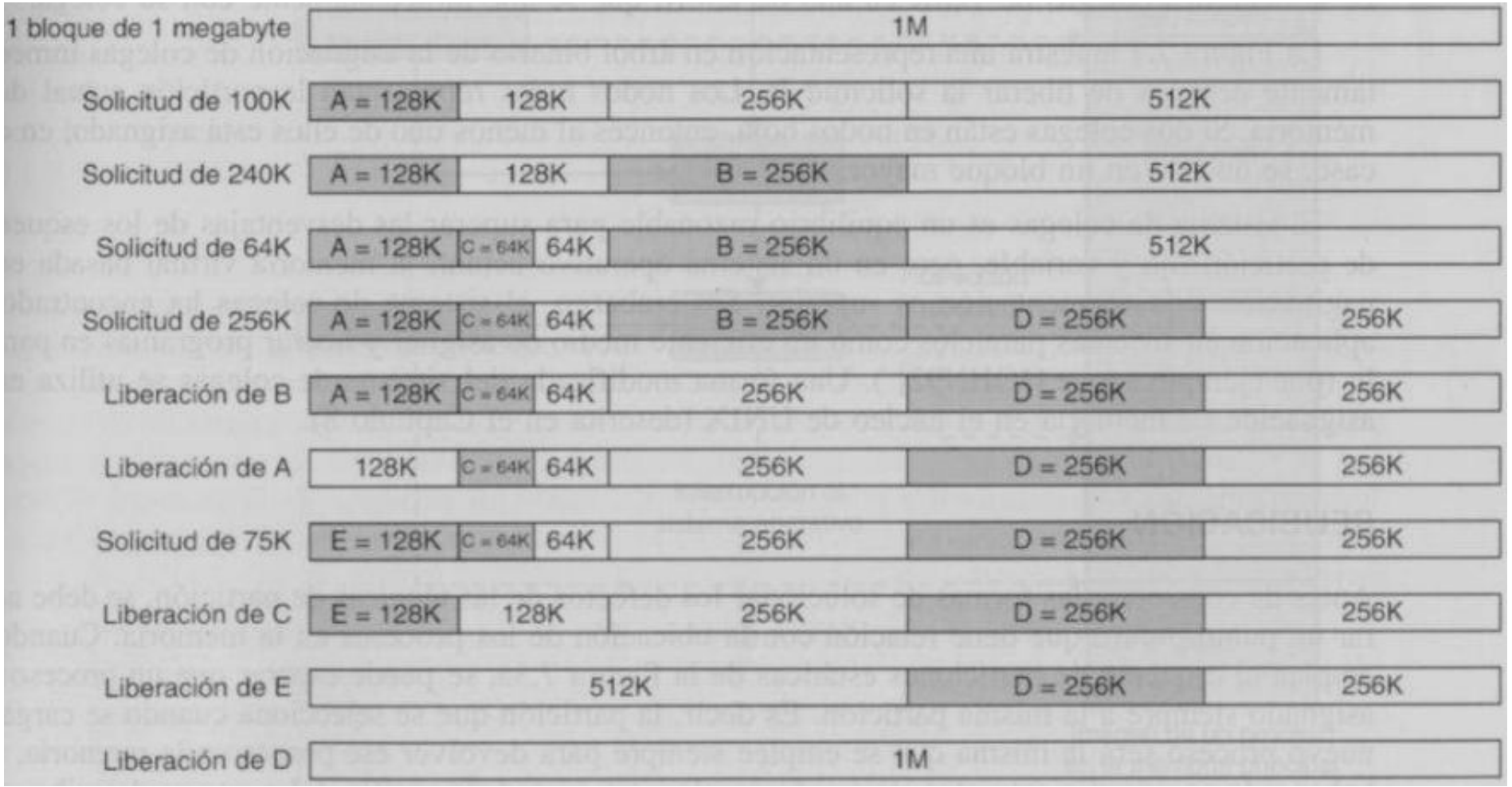
  Peor ajuste (worst-fix): ubicar el proceso generando el mayor hueco posible.
  Curiosamente, ofrece buenos resultados

Sistema de Colegas

- Presenta una alternativa para superar las desventajas de los esquemas de partición fija y variable
- Los bloques de memoria disponibles son de tamaño 2^K , con $L \leq K \leq U$, donde:
  2^L = tamaño de bloque más pequeño asignable
  2^U = tamaño de bloque más grande asignable
- Inicialmente, la memoria disponible se trata como un bloque de tamaño máximo 2^U
- Supongamos que se realiza una solicitud de tamaño T
  Si $2^{U-1} < T \leq 2^U \Rightarrow$ se le asigna el bloque entero 2^U
  En caso contrario:
    Se divide el bloque en dos colegas de igual tamaño 2^{U-1}
    Si $2^{U-2} < T \leq 2^{U-1} \Rightarrow$ se le asigna uno de los colegas 2^{U-1}
    En caso contrario, uno de los colegas se divide por la mitad nuevamente, en mitades de tamaño 2^{U-2}
    El proceso continúa hasta que el bloque más pequeño sea mayor o igual que T
- Cuando dos mitades contiguas (colegas) se liberan, vuelven a unirse en un único hueco con un tamaño igual a la suma de los tamaños de ambas

- http://en.wikipedia.org/wiki/Buddy_system

Sistema de Colegas



UNIX usa una forma modificada del sistema de colegas para la asignación de memoria en el núcleo

Memoria Virtual

- El método diseñado por Fotheringham en 1961 se conoce como Memoria Virtual

- Idea: el tamaño combinado de la pila, programa y datos de los procesos en un sistema puede exceder la memoria física disponible

- El Sistema Operativo mantiene en memoria aquellas partes del programa que deben permanecer en memoria y el resto lo deja en disco (memoria secundaria o swap space), las partes entre el disco y la memoria se intercambian de modo que se vayan necesitando.

- El acceso al swap space se realiza en milisegundos, mientras el acceso a memoria se realiza en microsegundos

Definiciones

- La memoria física es aquella implementada por circuitos integrados, por la electrónica del hardware. Puede apreciarse como un conjunto de celdas referenciables (o direccionables) por medio de un espacio de dirección lineal. Dichas direcciones son denominadas ***direcciones físicas***

- La memoria lógica de un proceso es aquella que un proceso puede referenciar (direccionar) y acceder utilizando sus instrucciones.
  Las direcciones empleadas por los procesos son las ***direcciones lógicas o virtuales***

- Cada proceso posee su propia memoria lógica, que es independiente de la memoria lógica del resto de los procesos.
  Las direcciones virtuales constituyen el ***espacio de direcciones virtual***

Paginación

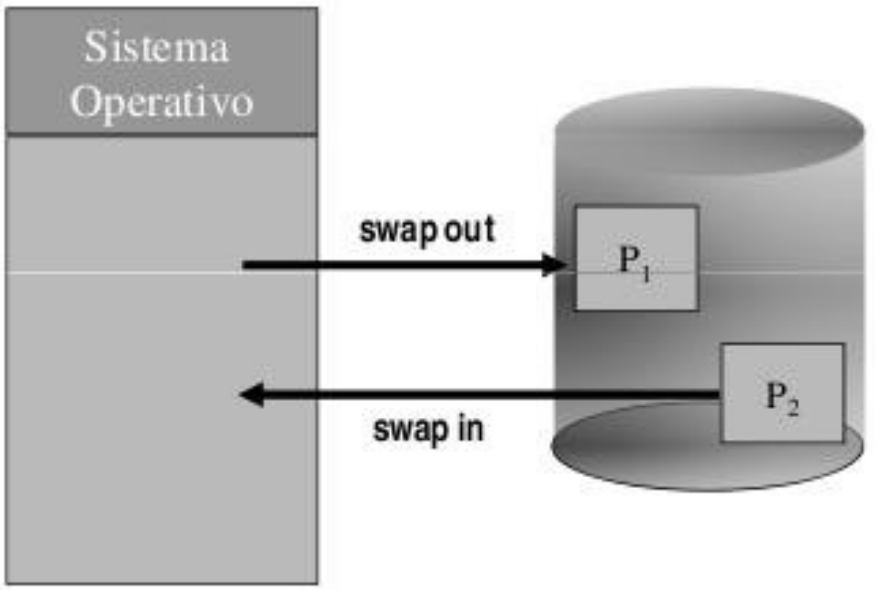
- Es una de las técnicas de memoria virtual más usadas que permite ejecutar programas más grandes que la memoria física disponible de forma transparente al programador
- La memoria principal se divide en un conjunto de marcos de igual tamaño
- Cada proceso se divide en páginas de igual tamaño que los marcos
- Un proceso se carga situando un número de sus páginas en marcos libres *no necesariamente contiguos*
- No genera fragmentación externa
- Existe una pequeña cantidad de fragmentación interna
- http://en.wikipedia.org/wiki/Paging

Intercambio o Swap

- El total de la memoria necesaria para los procesos en general es superior a la memoria física disponible en el sistema, por tal motivo se debe recurrir a almacenamiento secundario (**swap space**) para depositar las páginas de los procesos que no se encuentran en ejecución y obtener las de los procesos que se ejecutan

- **Swap-in**: traer una página desde el swap space a la memoria principal

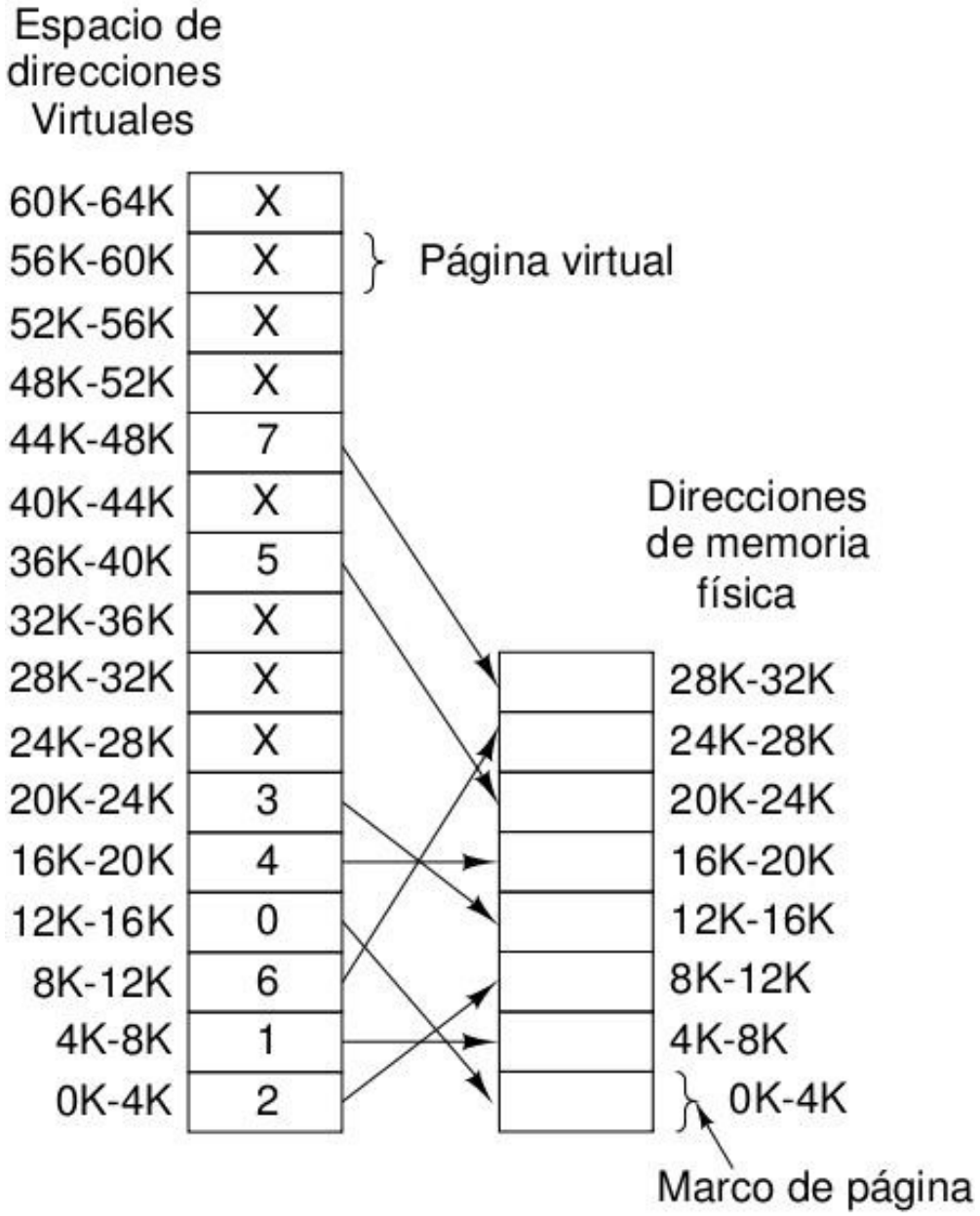
- **Swap-out**: llevar una página desde la memoria principal al swap space



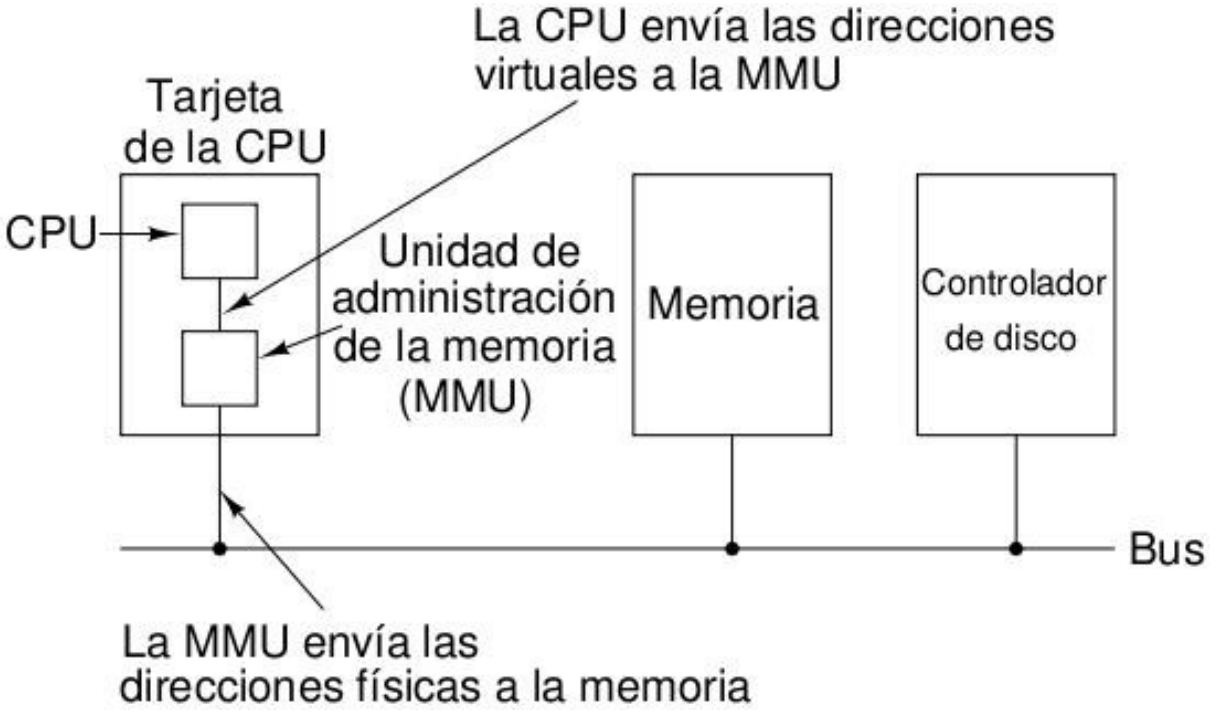
Instituto Politécnico Superior

Páginas y Marcos

- El *espacio de direcciones virtuales* se divide en *páginas*
- El *espacio de direcciones físicas* en *marcos de página*
- **`Tamaño(Marco de página)≡ Tamaño(Página)`**
- La página es la unidad de intercambio de memoria: las transferencias entre la memoria principal y el disco siempre se realizan en páginas



Memory Management Unit (MMU)



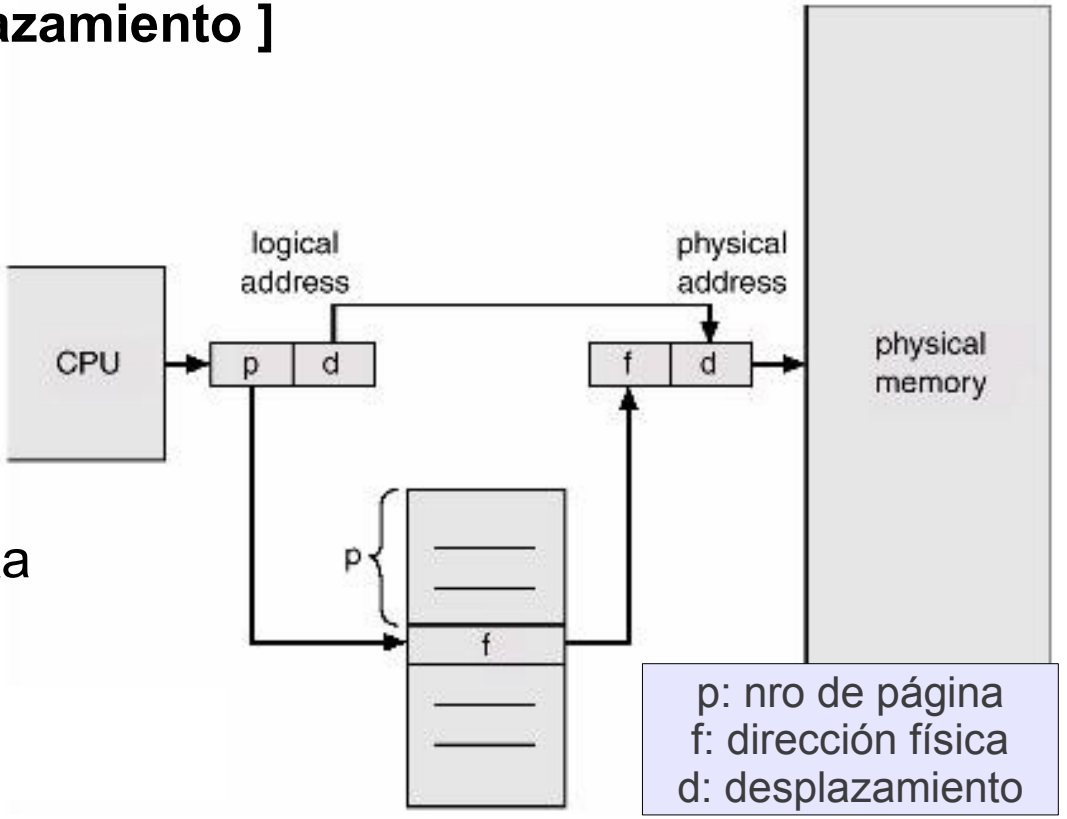
- La **MMU** es un componente de hardware incluido en el procesador que provee los mecanismos básicos que emplea el sistema operativo para la administración de la memoria

- Entre otras cosas la MMU mapea las *direcciones lógicas generadas por los procesos* con las correspondientes *direcciones físicas que se envían a la memoria*

- http://es.wikipedia.org/wiki/MMU

Tabla de páginas

- Establece la correspondencia entre direcciones virtuales (páginas) y direcciones físicas (marcos de página)
- El tamaño de la página y el marco es una potencia de dos
- Las direcciones virtuales se dividen en número de página virtual y desplazamiento
  [número de página virtual | desplazamiento]
- El número de página virtual se emplea como índice en la tabla de páginas
- La entrada correspondiente de la tabla de páginas indica el número de marco de página (si está en memoria)
- El número de marco de página se anexa al desplazamiento para formar una dirección física



Tamaño de las Páginas

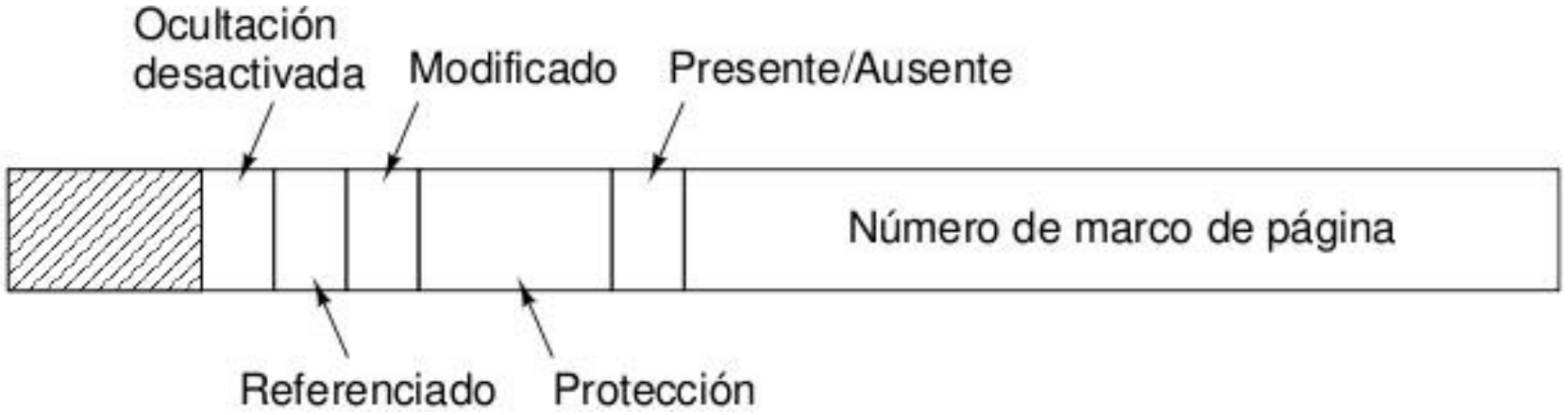
PAGINAS	FRAGMENTACION INTERNA	TAMAÑO DE TABLA DE PAGINAS
Pequeñas	Menor	Más grande
Grandes	Mayor	Más pequeña

- Mientras más pequeñas sean las páginas más comprenderán los procesos
- Factor a tener en cuenta: velocidad de lectura/escritura de disco
- Tamaños típicos: 4KB u 8KB
- Sistemas operativos modernos emplean múltiples tamaños de página

`http://www.solarisinternals.com/wiki/index.php/Multiple_Page_Size_Support`

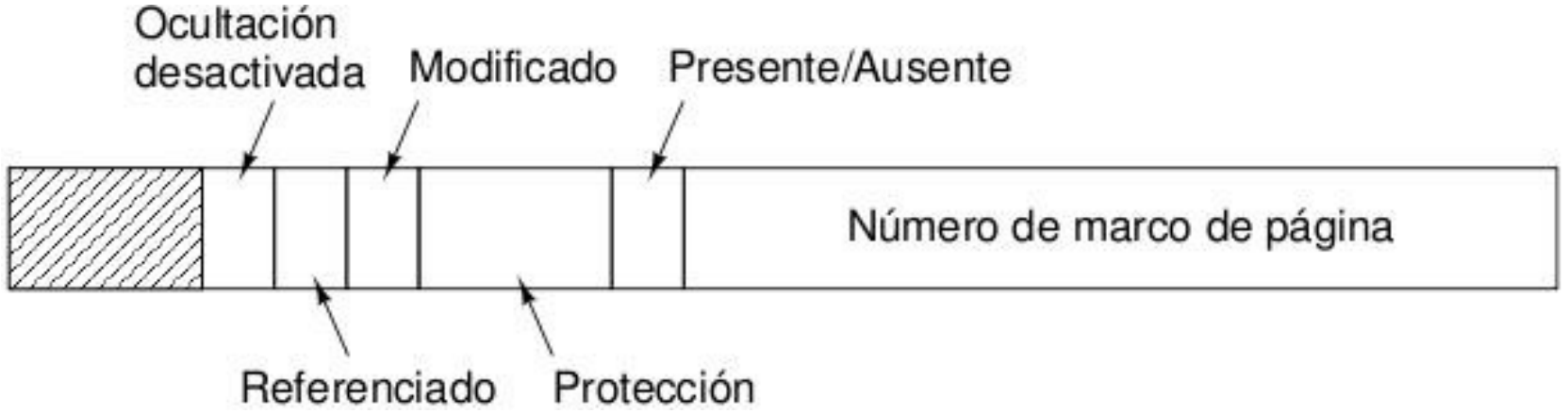
Tabla de páginas

- Estructura de una entrada de la tabla de páginas (posibles campos)
  - Número de marco
  - Bits presente/ausente ⇒ 1 entrada válida, 0 no está en memoria
  - Bits de protección ⇒ tipos de acceso permitido
  - Bit modificado: El bit se setea cuando la página sufre algún cambio en memoria real. Se utiliza cuando la página es seleccionada para salir de la memoria real(fallo de página). Si el bit está seteado la página se escribirá sobre el disco. Si no, sólo se desecha pues la imagen en disco es igual a la existente en la memoria real.
  - Bit referenciado o solicitado ⇒ se ha referenciado a la página



- http://en.wikipedia.org/wiki/Page_table

Bits Referenciado y Modificado



- Estos bits mantienen un registro del uso de las páginas

- El bit Referenciado se establece en 1 cuando una página ha sido referenciada por un proceso, tanto para lectura como por escritura, es útil para que el sistema operativo tome decisiones al momento de hacer swap out. Las páginas no utilizadas son mejores candidatas que las que si.

- El bit Modificado (o bit sucio) se establece en 1 si la página ha cambiado su contenido desde que fue cargada, si hay que emplear el marco que ocupa se debe escribir a disco, sino puede desecharse.

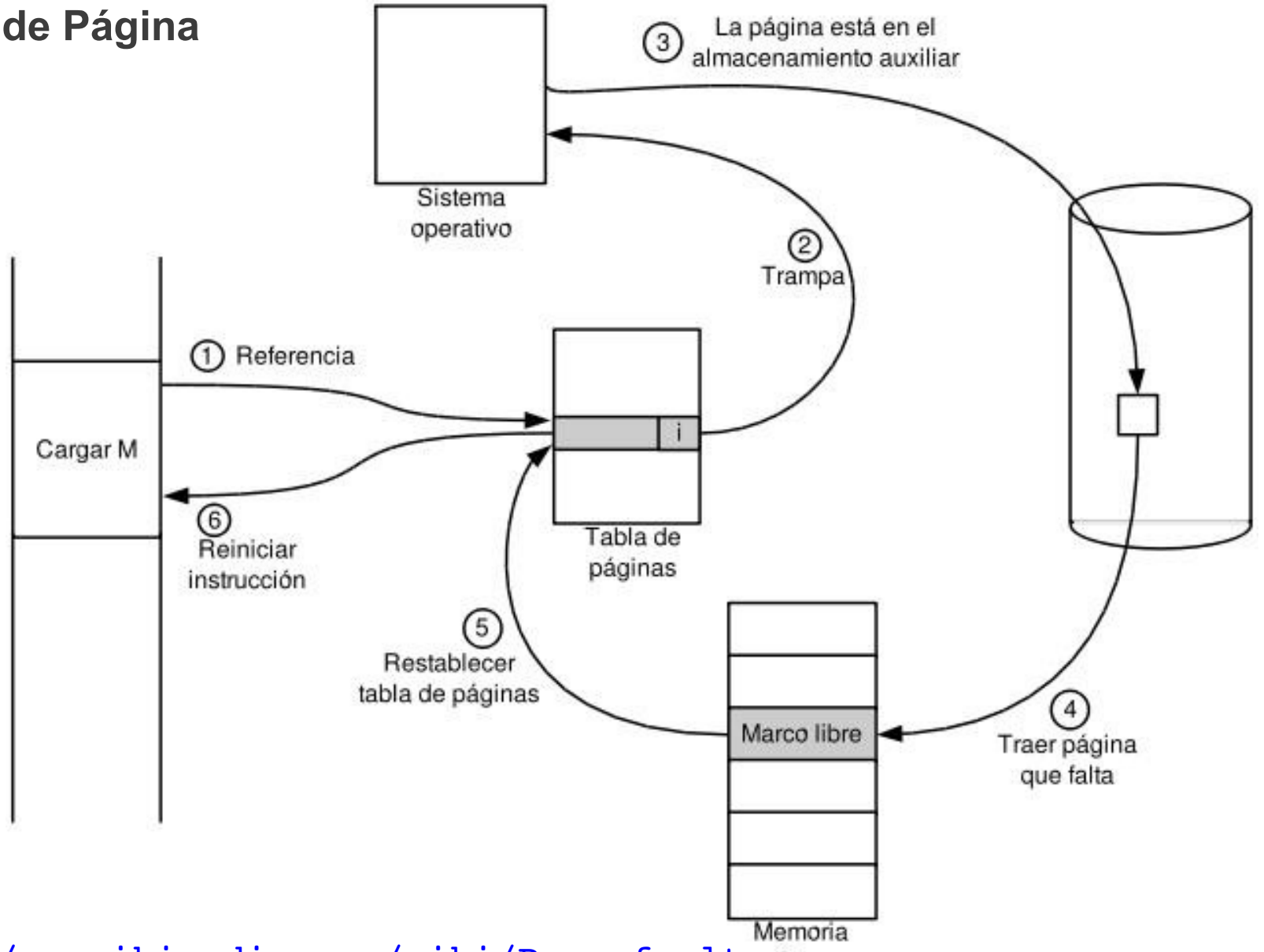
Fallo de página

- Si la página que contiene la próxima instrucción a ejecutar de un proceso y la que contiene los próximos datos a acceder están en memoria principal, dicho proceso podrá continuar ejecutándose normalmente

- Se denomina conjunto residente o **resident set** del proceso al conjunto de páginas que se encuentran en memoria principal.
  Mientras se haga referencia a páginas del resident set la ejecución se desarrollará normalmente.
  Mediante la tabla de páginas el procesador verifica si la página solicitada está en el resident set

- Si no existe una relación **página** ↔ **marco de página** ocurre un fallo de página
  1. La MMU detecta que la página no tiene correspondencia
  2. La MMU provoca un **fallo de página** (interrupción)
  3. El S.O. escoge un marco que no se esté usando mucho
     (si es necesario vuelve a escribir su contenido en disco)
  4. Trae la nueva página a ese marco
  5. Modifica la tabla de páginas
  6. Reinicia la instrucción

- Fallo de Página



http://en.wikipedia.org/wiki/Page_fault

Trashing o Hiperpaginación

- Si el número de marcos asignados a un proceso desciende por debajo del número mínimo requerido por la arquitectura del computador, debe suspenderse la ejecución de ese proceso.
  Luego deben descargarse sus páginas restantes, liberando los marcos asignados.
  En general, cualquier proceso que no cuente con marcos suficientes provocará fallos de página muy frecuentemente. Si se reemplazan páginas que a su vez están activas, se estará sustituyendo una página que casi de inmediato se volverá a necesitar. Por tanto, pronto vuelve a generarse otro fallo de página, ocurriendo esto una y otra vez.
- Cuando un proceso consume más tiempo paginando que ejecutando se dice que el proceso esta en situación de trash.
- Una alta tasa de fallos de página provoca
  - Uso de CPU para tareas de sistema
  - Acceso a swap space (dispositivo menos eficiente que la memoria)
  - Mayor uso de disco, posible encolamiento de solicitudes de I/O

  Conllevando a una caída general del rendimiento del sistema
- http://en.wikipedia.org/wiki/Thrash_(computer_science)

Trashing o Hiperpaginación: Anomalía de Belady

- Puede suponerse que mientras más marcos existan en la memoria, menos fallos de página tendrían lugar.

- Belady (1969) demostró mediante un contraejemplo que el algoritmo de paginación FIFO (ver más adelante) puede provocar más fallos con 4 marcos que con 3.

- Para un proceso con 5 páginas virtuales, numeradas de 0 a 4, si la páginas se referencian en el siguiente orden 0 1 2 3 0 1 4 0 1 2 3 4

Con 3 marcos – 9 fallos

	0	1	2	3	0	1	4	0	1	2	3	4
Página reciente	0	1	2	3	0	1	4	4	4	2	3	3
		0	1	2	3	0	1	1	1	4	2	2
Página más antigua			0	1	2	3	0	0	0	1	4	4
	x	x	x	x	x	x	x			x	x	

Con 4 marcos – 10 fallos

	0	1	2	3	0	1	4	0	1	2	3	4
Página reciente	0	1	2	3	3	3	4	0	1	2	3	4
		0	1	2	2	2	3	4	0	1	2	3
			0	1	1	1	2	3	4	0	1	2
Página más antigua				0	0	0	1	2	3	4	0	1
	x	x	x	x			x	x	x	x	x	x

- http://en.wikipedia.org/wiki/Belady's_anomaly

Asignación global o local

- El sistema operativo asigna marcos de página a los procesos de manera global o local

- En el esquema global cualquier proceso puede seleccionar un marco de reemplazo del conjunto de casi todos los marcos, los propios y los marcos de los procesos que tengan menor prioridad que éste
  De esta forma un proceso puede aumentar dinámicamente el número de marcos que posee.
  Como contra un proceso no puede controlar su propia frecuencia de reemplazo de páginas pues no depende exclusivamente de él
  Es el esquema más usado pues aprovecha marcos de poco uso, aprovechando más la memoria física, y por ende, aumentando el rendimiento del sistema

- En el esquema local, el proceso cuenta con un número fijo de marcos, pudiendo elegir únicamente entre sus marcos para reemplazo.
  Al tener un número acotado de marcos a emplear, un proceso puede verse limitado innecesariamente, aunque existan marcos con bajo uso

Algoritmos de reemplazo de Página

- Al surgir la necesidad de ubicar una página en memoria principal puede no existir un marco de página libre
  Se debe llevar una página a swap space *¿Pero cual?*

- Se emplean algoritmos para seleccionar las páginas a reemplazar

- Partamos del ***Algoritmo Optimo***, el mismo
  - Reemplaza la página que más tiempo va a tardar en necesitarse nuevamente
  - Es decir se presentará la menor cantidad posible de fallos de página

  No puede implementarse ya que el orden de las referencias a memoria no se puede conocer de antemano

- Existen muchos algoritmos, veremos los más típicos
  http://es.wikipedia.org/wiki/Algoritmos_de_reemplazo_de_páginas

Algoritmos de reemplazo de Página: NRU (No Usada Recientemente)

- Se utilizan los bits de R (referencia) y M (modificado)

- De forma periódica el bit R se establece a 0 ⇒ distinguir las páginas que se han solicitado recientemente

- Se establecen 4 categorías en base a los bits R y M:
  CLASE 0: R = 0, M = 0
  CLASE 1: R = 0, M = 1
  CLASE 2: R = 1, M = 0
  CLASE 3: R = 1, M = 1

- Desaloja aleatoriamente una página de la clase de número más bajo que no esté vacía

- Este algoritmo funciona bien, es rápido y es sencillo de implementar

Algoritmos de reemplazo de Página: FIFO

- Reemplaza la página que entró hace más tiempo en memoria (la primera que entró)

- Es muy malo, al no tener en cuenta el uso de las páginas
  Raramente es utilizado en su forma no modificada

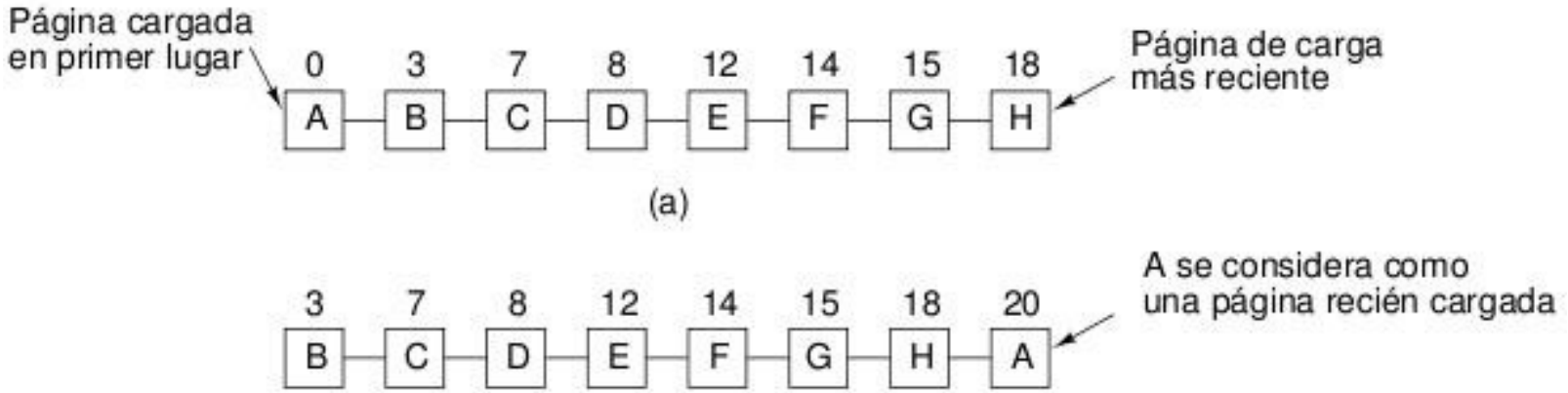
- Este algoritmo sufre de la denominada *Anomalía de Belady*

Algoritmos de reemplazo de Página: Segunda oportunidad

- Modificación del FIFO, evitando los problemas de desalojar una página que se use mucho, teniendo en cuenta su bit R

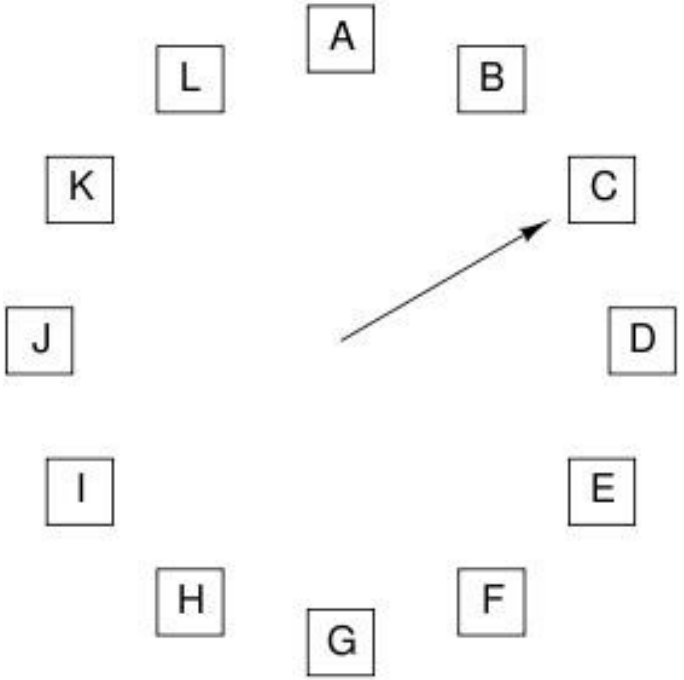
- Funcionamiento:
  Si el bit R de la página a quitar es 0 se elimina
  Si su bit R es 1, se pasa al final y se anula el valor del bit R (como si fuera nueva)
  Si todos los bit R son 1, su comportamiento es el de un FIFO

Busca una página antigua a la que no se le haya hecho referencia desde el último reemplazo



Algoritmos de reemplazo de Página: Algoritmo del Reloj

- Difiere del anterior sólo en la implementación
- Utiliza una lista circular y un puntero a la página a considerar
- Es más eficiente que el anterior, pues vita tener que mover las páginas en la lista



Cuando ocurre un fallo de página y no hay marcos libres, se inspecciona la página a la que apunta la manecilla.

La acción a realizar depende del bit R:
R = 0: Retira la página de la memoria
R = 1: Limpia R y avanza la manecilla

Algoritmos de reemplazo de Página: LRU (Last Recent Used)

• Buena aproximación al algoritmo óptimo
Seleccionar la página que no ha sido utilizada hace más tiempo

Problema: Es ineficiente si se implementa con una lista enlazada
⇒ actualización de la lista en cada referencia a memoria

Algoritmos de reemplazo de Página: Reemplazo al azar

- Reemplaza páginas de memoria al azar
- Eficiente, elimina la sobrecarga de verificar las referencias en cada página
- Generalmente es mejor que FIFO, y para referencias cíclicas de memoria es mejor que LRU, aunque en general LRU lo supera

Working Set (Grupo de Trabajo)

- Según el **Principio de la proximidad o localidad** (Denning 1968) se sabe que en un entorno a la referencia actual, el programa tiende a tener referencias próximas, lo que significa una pequeña porción del total de espacio direccionable.
- Esta proximidad o localidad
  - es una propiedad empírica (observada mediante pruebas)
  - nunca está garantizada pero es altamente probable.
- Esta proximidad puede ser de dos formas:.
  - *Proximidad temporal*: En un futuro cercano, un programa tiende a usar datos que han sido usados en un pasado próximo.
    Se apoya en el uso de, bucles, subrutinas, ciertas estructuras de datos y variables contadoras y acumuladoras
  - *Proximidad espacial*: Implica una cierta secuencialidad en la ejecución de un proceso. Normalmente existe cierta autocorrelación con los lugares de memoria
    Se apoya en: el recorrido de arreglos, la ejecución secuencial de código, la tendencia de los programadores a colocar definiciones de variables relacionadas, próximas entre sí

Working Set (Grupo de Trabajo)

- El Working Set de un proceso es *dinámico*
- Principio de localidad
  http://en.wikipedia.org/wiki/Locality_of_reference
- Working Set
  http://en.wikipedia.org/wiki/Working_set

Prepaginación

- El efecto de paginar bajo demanda, es decir cargar páginas a memoria principal sólo cuando son solicitadas, se denomina paginación pura
  Esto presenta el inconveniente de que las páginas se cargan de a una en memoria principal, provocando considerables tiempos de espera debido al acceso continuo a swap space (disco, lento)

- La **prepaginación o paginación anticipada**, apoyándose en el concepto de Working Set, intentado predecir el comportamiento de los procesos, va cargando un conjunto de páginas de un mismo proceso antes de que este las solicite, por tanto, eventualmente, se evitan fallos de páginas inmediatos.

- Es muy útil para la carga inicial de los procesos

Memoria Asociativa / TLB (Translation Lookaside Buffer)

- Dispositivo de hardware que traduce direcciones virtuales a físicas sin acceder a la tabla de páginas

- Es parte interna de la MMU

- Contiene un número pequeño de entradas

- Cada entrada contiene información sobre una página
  - Número de página
  - Bit de modificado
  - Bit de protección
  - Número de marco de página
  - Bit de validez

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

- http://es.wikipedia.org/wiki/Translation_Lookaside_Buffer

Memoria Asociativa / TLB (Translation Lookaside Buffer)

- *Funcionamiento*
  Dirección virtual ⇒ se comprueba, en paralelo, si su número de página está presente en el TLB

  ESTA
  Si no viola los bits de protección ⇒ el número de marco se toma del TLB
  Si viola los bits de protección ⇒ Fallo de protección

  NO ESTA ⇒ Fallo del TLB
  La MMU consulta la tabla de páginas
  Desaloja una entrada del TLB
  La sustituye por la nueva entrada

TLB y Memoria Física