Introducción a los Sistemas Operativos

Administración de Memoria Principal
Práctica 5

Memoria

- La organización y administración de la *"memoria principal* es uno de los factores más importantes en el diseño de los S. O.
- Los programas y datos deben estar en el almacenamiento principal para:
  - Poderlos ejecutar.
  - Referenciarlos directamente.

Memoria (cont.)

- La parte del S. O. que administra la memoria se llama *"administrador de la memoria"*:
  - Lleva un registro de las partes de memoria que se están utilizando y de aquellas que no.
  - Asigna espacio en memoria a los procesos cuando estos la necesitan.
  - Libera espacio de memoria asignada a procesos que han terminado.
- Se espera de un S.O. un uso eficiente de la memoria con el fin de alojar el mayor número de procesos

Memoria (cont.)

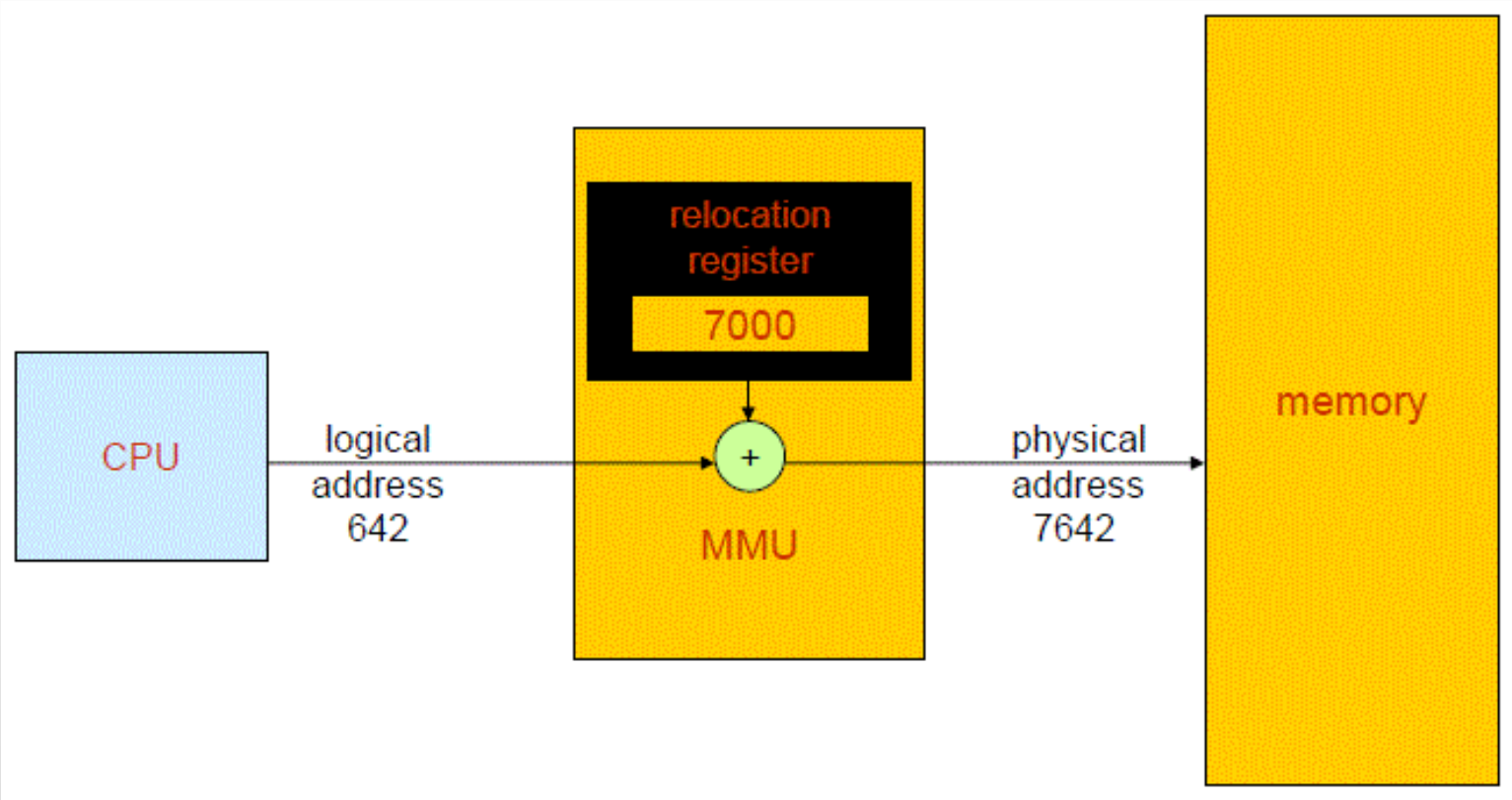
- El S.O. debe:
  - Lograr que el programador se abstraiga de la alocación de los programas
  - Brindar seguridad entre los procesos para que unos no accedan a secciones privadas de otros
  - Brindar la posibilidad de acceso compartido a determinadas secciones de la memoria (librerías, código en común, etc.)
  - Garantizar la performance del sistema

Mecanismo de

- **Dirección Lógica:**
  - En forma genérica es una dirección que enmascara o abstrae una dirección física
  - Referencia a una localidad de memoria
  - Se la debe traducir a una dirección física
- **Dirección física:**
  - Es la dirección real, es con la que se accede efectivamente a memoria.
  - Representa la dirección absoluta en memoria principal
- La CPU trabaja con direcciones lógicas. Para acceder a memoria se deben transformar en direcciones absolutas
- El mapeo entre direcciones virtuales y físicas se realiza en hardware (MMU (Memory Management Unit)

Mecanismo de

☑ **¿Cómo se realiza la traducción?:**



Mecanismos de asignación de memoria

- Particiones Fijas:
  - La memoria se divide en particiones o regiones de tamaño Fijo (pueden ser todas del mismo tamaño o no)
  - Alojan un proceso cada una
  - Cada proceso se coloca de acuerdo a algún criterio (First Fit, Best Fit, Worst Fit, Next Fit) en alguna partición
- Particiones dinámicas:
  - Las particiones varían en tamaño y en número
  - Alojan un proceso cada una
  - Cada partición se genera en forma dinámica del tamaño justo que necesita el proceso

¿Qué problemas se generan en cada caso?

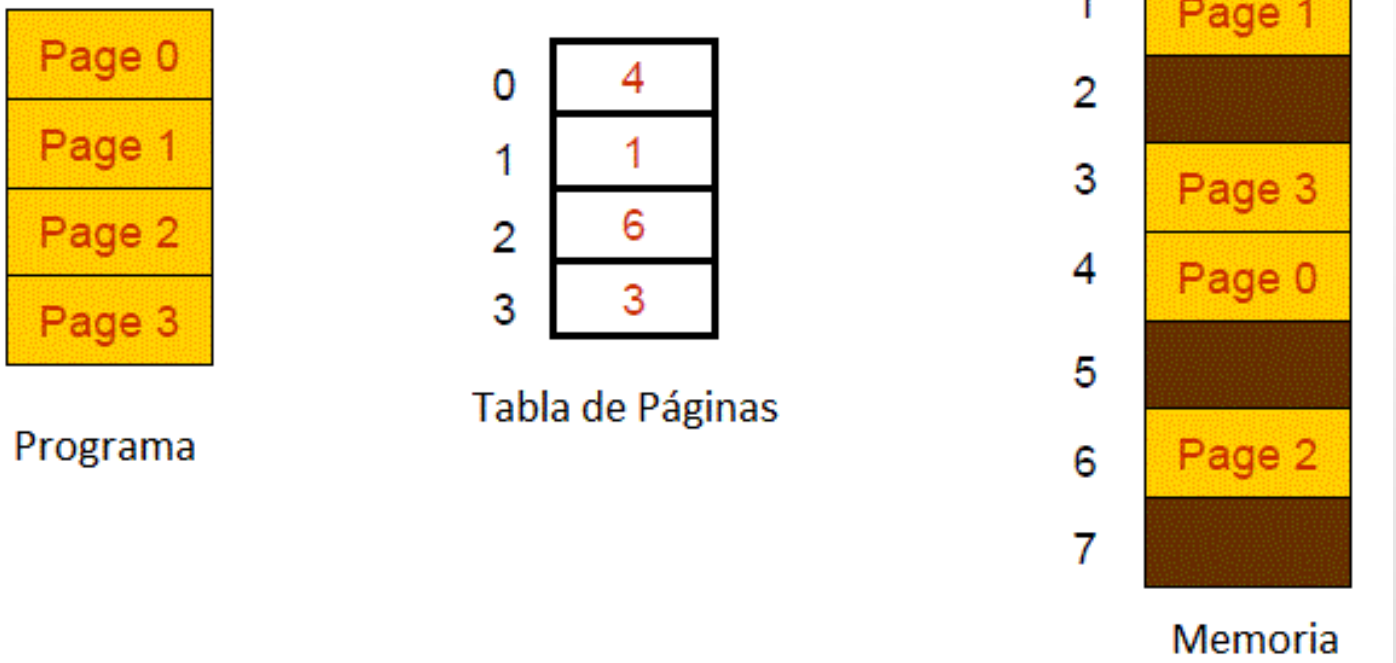
Fragmentación

- La fragmentación se produce cuando una localidad de memoria no puede ser utilizada por no encontrarse en forma contigua
- **Fragmentación Interna:**
  - Se produce en el esquema de particiones Fijas
  - Es la porción de la partición que queda sin utilizar
- **Fragmentación Externa:**
  - Se produce en el esquema de particiones dinámicas
  - Son huecos que van quedando en la memoria a medida que los procesos finalizan
  - Al no encontrarse en forma contigua puede darse el caso de que tengamos memoria libre para alocar un proceso, pero que no la podamos utilizar
  - Para solucionar el problema se puede acudir a la compactación, pero es muy costosa

Paginación

- La memoria se divide en porciones de igual tamaño llamadas **marcos**
- El espacio de direcciones de los procesos se divide en porciones de igual tamaño denominados **páginas**
- El tamaño de los marcos es igual al tamaño de las páginas (generalmente 512 Bytes)
- El S.O. mantiene una tabla de páginas para cada proceso la cual contiene el marco donde se encuentra cada página

Paginación – Direccionamiento (cont)



Paginación - Direccionamiento

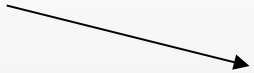
- Un proceso en ejecución hace referencia a una dirección virtual “v = (p,d)”
- El S.O. busca la página “p” en la *“tabla de páginas” del proceso* y determina en que marco se encuentra
- La dirección de almacenamiento real se forma por la concatenación de “p ’ ” y “d”, donde “p” es el número de página y “d” es el desplazamiento.

Paginación – Direccionamiento - Ejemplo

- Memoria administrada por sistema de paginación
- Tamaño de página 512 Bytes
- Cada dirección de memoria referencia 1 byte
- Los marcos en memoria principal se encuentran desde la dirección física 0.
- Tenemos un proceso con un tamaño 2000 bytes y con la siguiente tabla de páginas

Paginación – Direccionamiento - Ejemplo

Página	Marco
0	1
1	**2**
2	3
3	0



Marco	Inicio-Fin
0	0 - 511
1	512 - 1023
2	**1024 - 1535**
3	1536 - **2047**

F.I. de 48 B.

- Si tenemos una dirección **lógica**, por ejemplo 580:
  - Para averiguar el número de página hacemos 580 div 512 = 1. Luego esta dirección corresponde a la página 1 que se encuentra en el marco 2
  - Para averiguar el desplazamiento hacemos 580 mod 512 = 68
  - La dirección física es 1024 + 68 = 1092

Paginación – Direccionamiento - Ejemplo

Página	Marco
0	1
1	**2**
2	3
3	0

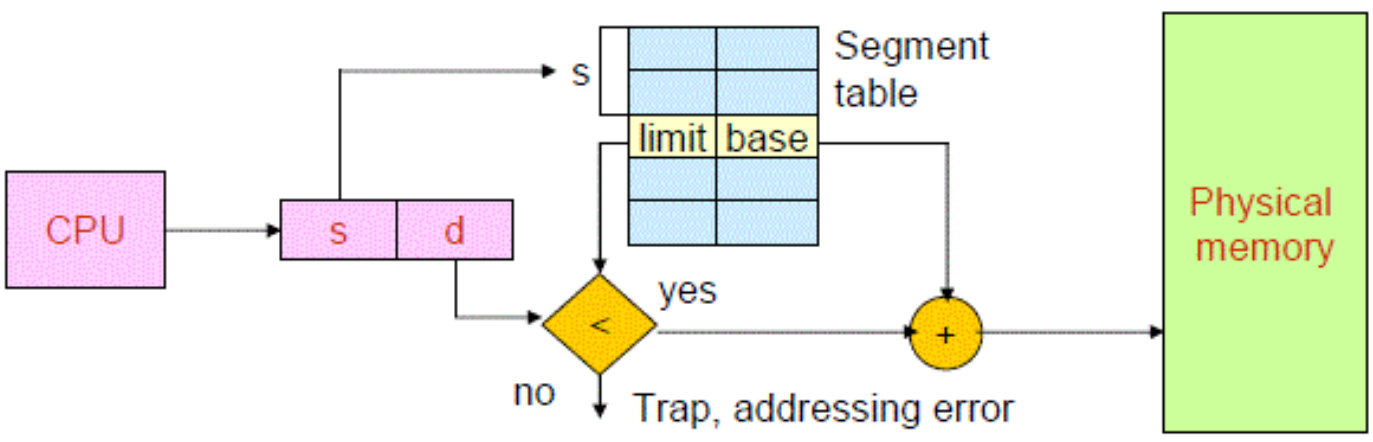
Marco	Inicio-Fin
0	0 - 511
1	512 - 1023
2	**1024 - 1535**
3	1536 - **2047**

F.I. de 48 B.

- Si tenemos una dirección **física**, por ejemplo 1092:
  - Para averiguar el número de marco hacemos 1092 div 512 = 2. En el marco número 2 tenemos la página número 1
  - Para averiguar el desplazamiento hacemos 1092 mod 512 = 68
  - La dirección lógica es 512 + 68 = 580

Segmentación

- La segmentación básicamente la podemos ver como una mejora a la paginación (no hay F. I.)
- Ahora la tabla de segmentos, además de tener la dirección de inicio del mismo, tiene la longitud o límite
- Las direcciones lógicas constan de dos partes (un número de segmento “s” y un desplazamiento “d” dentro del segmento (0 < “d” < límite)



Memoria Virtual con Paginación

- La técnica de paginación intenta alocar la mayor cantidad de páginas necesarias posibles
- Cada vez que hay que alocar una página en un marco, se produce en fallo de página o Page Fault
- ¿Que sucede si es necesario alocar una página y ya no hay espacio disponible?
- Se debe seleccionar una pagina victima, para lo cual existen diversos algoritmos
- ¿Cual seria el mejor algoritmo?
  - El que seleccione como página víctima aquella que no vaya a ser referenciada en un futuro próximo
- La mayoría de los algoritnmos predicen el comportamiento futuro mirando el comportamiento pasado.

Algoritmo Óptimo

- ✓ Selecciona la pagina cuyo próxima referencia se encuentra mas lejana a la actual
- ✓ Imposible de implementar ➔ No se conoce los futuros eventos

Marco/Página	1	2	1	3	4	1	4	3	5
F1	1	1	1	1	1	1	1	1	?
F2		2	2	2	4	4	4	4	?
F3				3	3	3	3	3	?
PF?	X	X		X	X				

Cont. Secuencia: 4 6 3 5 8 1

Algoritmo LRU

- El algoritmo LRU (Least Recently Used) Reemplaza la pagina que no fue referenciada por mas tiempo.
- Cada pagina debe tener información del instante de su ultima referencia ➔ El overhead es mayor

Marco/Página	1	2	1	3	4	1	4	3	5
F1	1	1	1	1	1	1	1	1	5
F2		2	2	2	4	4	4	4	4
F3				3	3	3	3	3	3
PF?	X	X		X	X				X

Algoritmo FIFO

- El Algoritmo FIFO (Fist In First Out) trata a los frames en uso como una cola circular.
- Simple de implementar.
- La pagina mas vieja en la memoria es reemplazada.
- La pagina puede ser necesitada pronto.

Marco/Página	1	2	1	3	4	1	4	3	5
F1	1	1	1	1	4	4	4	4	4
F2		2	2	2	2	1	1	1	1
F3				3	3	3	3	3	5
PF?	X	X		X	X	X			X

Algoritmo FIFO con segunda chance

- Se utiliza un Bit adicional ➔ Bit de referencia
- Cuando la pagina se carga en memoria, el bit R se pone en 0
- Cuando la pagina es referenciada el bit R se coloca en 1.
- La victima se busca en orden FIFO. Se selecciona la primer pagina cuyo bit R esta en 0.
- Mientras se busca la victima cada bit R que tiene el valor 1, se cambia en 0.

Algoritmo FIFO con segunda chance

Marco/Página	1	2	1	3	4	1	4	3	5
F1	1	1	1*	1*	1	1*	1*	1*	1
F2		2	2	2	4	4	4*	4*	4
F3				3	3	3	3	3*	5
PF?	X	X		X	X				X

* ⇔ Bit R =1

Paginación – Asignación de Marcos

- La asignación de marcos se puede realizar de 2 modos:
  - Asignación Fija: A cada proceso se le asigna una cantidad arbitraria de Marcos. A su vez para el reparto de puede usar:
    - Reparto equitativo: Se asigna la misma cantidad de marcos a cada proceso
    - Reparto proporcional: Se asignan marcos en base a la necesidad que tiene cada proceso
  - Asignación dinámica: Los procesos se van cargando en forma dinámica de acuerdo a la cantidad de marcos que necesiten

Paginación – Alcance del reemplazo

- Al momento de seleccionar una página víctima entre todos los procesos podemos usar:
  - **Reemplazo Global**
    - El fallo de página de un proceso puede reemplazar la página de cualquier proceso.
  - **Reemplazo Local**
    - El fallo de página de un proceso solo puede reemplazar sus propias páginas.

Descarga asincrónica de Páginas

- El SO reserva uno o varios marcos para la descarga asincrónica de páginas
- Cuando es necesario descargar una página modificada:
  - La página que provoco el fallo se coloca en un frame de descarga asincrónica.
  - El SO envía la orden de descargar asincrónicamente la página modificada mientras continua la ejecución de otro proceso
  - El frame de descarga asincrónica pasa a ser el que contenía a la página victima que ya se descargó correctamente.

Ejemplo de Algoritmo FIFO

Marco/ Página	1	2	1ᴹ	3	4	3ᴹ	5		6	7	
F1	1	1	1ᴹ	1ᴹ	1ᴹ	1ᴹ	1ᴹ			7	7
F2		2	2	2	2	2	2	2	6	6	6
F3				3	3	3ᴹ	3ᴹ	3ᴹ	3ᴹ	3ᴹ	
F4					4	4	4	4	4	4	4
F5							5	5	5	5	5

Performance

- La técnica de paginación por demanda puede generar una degradación de rendimiento del sistema debido a que el reemplazo de páginas es costoso.

- Tasa de Page Faults 0 < p < 1
  - Si p = 0 no hay page faults
  - Si p = 1, cada referencia es un page fault

- Effective Access Time – Medida utilizada para medir este costo

Am = tiempo de acceso a la memoria real

Tf = tiempo de atención de una fallo de pagina

At = tiempo de acceso a la tabla de paginas. Es igual al tiempo de acceso a la memoria (Am) si la entrada de la tabla de páginas no se encuentra en la TLB (cache donde reside la tabla de páginas).

$$TAE = At + (1 - p) * Am + p * (Tf + Am)$$

Thrashing

- Thrashing (hiperpaginación).
  - Decimos que un sistema está en *thrashing* cuando pasa más tiempo paginando que ejecutando procesos.
- Si un proceso cuenta con todos los frames que necesita, no habría thrashing!!.
- Existen técnicas para evitarlo:
  - Estrategia de Working Set basado en el modelo de localidad.

Modelo de Localidad

- Las referencias a datos y programa dentro de un proceso tienden a agruparse.
- La localidad de un proceso en un momento dado se da por el conjunto de páginas que son referenciadas en ese momento.
- En cortos períodos de tiempo, el proceso necesitará pocas “piezas” del proceso (por ejemplo, una página de instrucciones y otra de datos...).
- Entonces se define una ventana de trabajo o Working Set (Δ) que contiene las referencias de memoria más recientes.
- Working set: es el conjunto de páginas que tienen las Δ referencias a páginas más recientes .

La selección del Δ

- Δ chico: no cubrirá la localidad, toma muy pocas referencias.
- Δ grande: puede tomar varias localidades, tomo referencias de la localidad y algunas de mas
- Para determinar la medida del Worknig Set debemos tener en cuenta:
  - m = cantidad frames disponibles
  - D= demanda total de frames.
  - WSS_i = medida del working set del proceso p_i.
  - $\Sigma\ WSS_i=D$;
  - Si D>m, habrá ***thrashing***