

2a) git.

b) Lleva un registro de las partes de la 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 que el SO haga uso eficiente de esta memoria con el fin de alojar el mayor número de procesos.

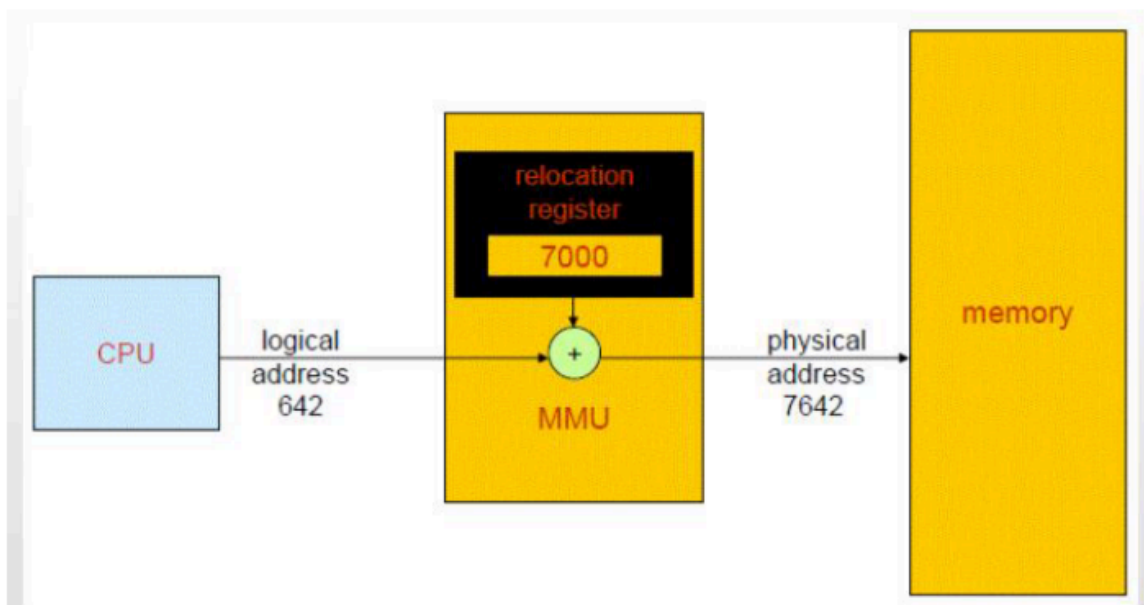
chat gpt:

El sistema operativo debe disponer de la siguiente información para administrar la memoria con estos métodos:

- Tamaño de la memoria física total.
- Tamaños y ubicaciones de las particiones (en el caso de particiones fijas).
- Información sobre particiones disponibles y ocupadas.
- Tamaños de los procesos y sus requisitos de memoria.
- Historial de asignaciones y liberaciones de memoria.

c)

El mapeo entre direcciones virtuales y físicas se realiza mediante hardware → MMU (Memory Management Unit)



3)

Cuando se trabaja con particiones fijas, se pueden considerar dos enfoques para el tamaño de las particiones:

1. ****Particiones de igual tamaño**:**

- ***Ventajas*:**

- Simplicidad: La gestión de particiones de igual tamaño es más simple y predecible.

Cada partición tiene un tamaño constante y es fácil de gestionar.

- Evita la fragmentación externa: Dado que todas las particiones son del mismo tamaño, no se produce fragmentación externa, ya que los procesos se asignan a particiones que coinciden con su tamaño.
- *Desventajas*:
 - Desperdicio de espacio: Puede haber desperdicio de espacio si los procesos no ocupan completamente el tamaño de la partición asignada, lo que genera fragmentación interna.
 - Limita la asignación de procesos: Los procesos deben ajustarse al tamaño de las particiones, lo que puede limitar la capacidad de ejecutar procesos de diferentes tamaños.

2. ****Particiones de diferente tamaño****:

- *Ventajas*:
 - Uso eficiente de la memoria: Las particiones de diferentes tamaños permiten asignar el tamaño adecuado de la partición a cada proceso, lo que reduce la fragmentación interna y mejora la eficiencia en el uso de la memoria.
 - Flexibilidad: Puede acomodar procesos de diferentes tamaños de manera más efectiva, lo que aumenta la capacidad de ejecutar una variedad de aplicaciones.
- *Desventajas*:
 - Mayor complejidad: La gestión de particiones de diferentes tamaños es más compleja que las particiones de igual tamaño, ya que requiere llevar un registro de los tamaños y ubicaciones de las particiones ocupadas y desocupadas.
 - Posible fragmentación externa: Aunque la fragmentación interna se reduce, la fragmentación externa sigue siendo un problema potencial en sistemas con particiones de diferentes tamaños, ya que las particiones ocupadas y desocupadas pueden estar distribuidas en la memoria, lo que podría desperdiciar espacio entre ellas.

En resumen, las particiones de igual tamaño ofrecen simplicidad pero pueden generar fragmentación interna, mientras que las particiones de diferente tamaño permiten un uso más eficiente de la memoria, pero con la complejidad adicional de la gestión y el riesgo de fragmentación externa. La elección entre estos métodos depende de las necesidades específicas del sistema y las aplicaciones que se ejecutarán.

5) git

6)

6.- Cite similitudes y diferencias entre la técnica de paginación y la de particiones fijas.

Similitudes:

- Ambas pueden generar fragmentación interna.
- Ambas dividen a la memoria en fracciones de tamaño fijo.

Diferencias:

- La paginación divide al proceso en varias particiones.
- La partición fija coloca todo un proceso de forma continua en una sola partición.

7)a)

como calculas la direccion virtual?

Tabla de Paginas	
Pagina	Marco
0	3
1	5
2	2
3	6



Memoria Principal (MP)			
#Marco	#Pagina	Direccion Virtual	Direccion Fisica
0	-	-	0..511
1	-	-	512..1023
2	2	1024..1535	1024..1535
3	0	0..511	1536..2047
4	-	-	2048..2559
5	1	512..1023	2560..3071
6	3	1536..1999	3072..3583

b)

Dir. Lógica div Tam. Página = N.º de Página

Dir. Lógica mod Tam. Página = Desplazamiento

Dir. Física = Inicio o base del frame + desplazamiento

15 7-1



Dir Lógica	DIV (512) N° Pag	MOD (512) Desplazamiento	Marco (base)	Dir. Física	Valida
35	0	35	M3 1536	$1536 + 35 = 1571$	Si
512	1	0	M5 2560	$2560 + 0 = 2560$	Si

2051	4	3	-	-	Error
0	0	0	M3 1536	$1536 + 0 = 1536$	Si
1325	2	301	M2 1024	$1024 + 301 = 1325$	Si
602	1	90	M5 2560	$2560 + 90 = 2650$	Si

c) Esta bien la formula de la dir. logica? pq en la teoria directamente hace el tamaño de la pagina + desplazamiento. Ademas aca no redondeo cuando se calculó el marco (como hizo en el inciso b) hay que hacerlo? tipo cuando da 2.4, si es asi entonces va a cambiar el div de la dir. fisica 1500 y 2000.

Dir. física div Tam Marco = N.º de Marco

Dir. física mod Tam Marco = Desplazamiento

Dir. lógica = (Nº página * tam. página) + desplazamiento

Dir Fisica	DIV (512) N° Marco	MOD (512) Desplazamiento	Página (base)	Dir. Logica	Valida
509	0	509	-	-	Error
1500	2	476	P2 1024	1024 + 476 = 1500	Si
0	0	0	-	-	Error
3215	6	143	P5 1536	1536 + 143 = 1679	Si
1024	2	0	P2 1024	1024 + 0 = 1024	Si
2000	3	464	P0 0	0 + 464 = 464	Si

d) ni idea como verlo esto. Solo hay fragmentacion interna creo.

8)

a)

Para calcular la cantidad de bits necesarios para representar direcciones lógicas y físicas en un sistema de paginación, podemos seguir estos pasos:

Cantidad de bits para dirección lógica:

Dado que tienes un espacio lógico de 8 páginas, primero calcula cuántos bits se necesitan para representar cada página:

8 páginas = 2^3 páginas

Por lo tanto, se necesitan 3 bits para representar una página en el espacio lógico.

Dado que cada página tiene 1024 bytes, calcula cuántos bits se necesitan para representar una dirección dentro de una página (offset):

1024 bytes = 2^{10} bytes

Por lo tanto, se necesitan 10 bits para representar un desplazamiento (offset) dentro de una página.

Entonces, la dirección lógica se compone de los bits para la página y los bits para el desplazamiento:

Bits para la página + Bits para el desplazamiento = 3 bits + 10 bits = 13 bits

Por lo tanto, se necesitan 13 bits para representar una dirección lógica.

b)

Tienes una memoria física de 32 marcos, por lo que se necesitan $\log_2(32)$ bits para representar un marco en la memoria física.

$\log_2(32) = 5$ bits

- Bits para el desplazamiento: 10 bits (ya que cada página tiene 1024 bytes, como se mencionó anteriormente).

Por lo tanto, en total, se necesitan 15 bits para representar una dirección física en este sistema de paginación.

9)

- a) Los segmentos no tienen que ser del mismo tamaño como pasaba con las páginas y tienen dirección lógica que se compone por un número de segmento s y un desplazamiento d dentro del segmento. ($0 < d < \text{limite}$). Necesitas el "límite" porque tienes segmentos de distinto tamaño.

chat gpt:

La segmentación es un método de administración de memoria utilizado por los sistemas operativos para organizar y gestionar la memoria de una manera más flexible y eficiente que otros métodos, como la paginación o la memoria continua. En la segmentación, la memoria se divide en segmentos, y cada segmento puede contener una parte lógica o funcional específica de un programa o proceso. Aquí está una explicación de cómo funciona el método de segmentación:

1. ****División en Segmentos****: En lugar de dividir la memoria en páginas o bloques de tamaño uniforme, en la segmentación se divide en segmentos de tamaños diferentes, cada uno diseñado para contener un tipo particular de información o funcionalidad. Por ejemplo, puede haber segmentos para código, datos, pila, bibliotecas, variables globales, etc.
2. ****Tablas de Segmentos****: Para llevar un seguimiento de la ubicación y el tamaño de cada segmento, el sistema operativo mantiene una tabla de segmentos. Cada entrada en esta tabla contiene información sobre un segmento, como su dirección base (dirección de inicio) y su longitud.
3. ****Direcciones Lógicas****: Los programas o procesos hacen referencia a direcciones lógicas que consisten en un número de segmento y un desplazamiento dentro de ese segmento. Por lo tanto, la dirección lógica se compone de dos partes: número de segmento y desplazamiento.
4. ****Traducción de Direcciones****: Cuando se accede a una dirección lógica, el sistema operativo utiliza la tabla de segmentos para buscar la entrada del segmento correspondiente y determinar su dirección base. Luego, agrega el desplazamiento a la dirección base para calcular la dirección física real en la memoria.
5. ****Protección y Permisos****: La segmentación permite una implementación más eficiente de la protección de memoria. Cada segmento puede tener permisos específicos, lo que permite un control más granular sobre quién puede acceder a qué partes de la memoria.
6. ****Fragmentación Interna****: A diferencia de la paginación, la segmentación no sufre de fragmentación interna, ya que los segmentos pueden tener tamaños variables y estar diseñados para contener información específica. Esto mejora la eficiencia en el uso de la memoria.
7. ****Fragmentación Externa****: Sin embargo, la segmentación puede sufrir fragmentación externa si los segmentos no se ubican de manera eficiente en la memoria física.

En resumen, la segmentación es un método que permite una gestión más flexible y eficiente de la memoria al dividirla en segmentos que representan partes lógicas de programas o procesos. Esto facilita la administración de la memoria y proporciona ventajas como la

protección y la eliminación de la fragmentación interna, aunque puede estar sujeto a la fragmentación externa.

git:

a) Explique cómo trabaja este método de asignación de memoria.

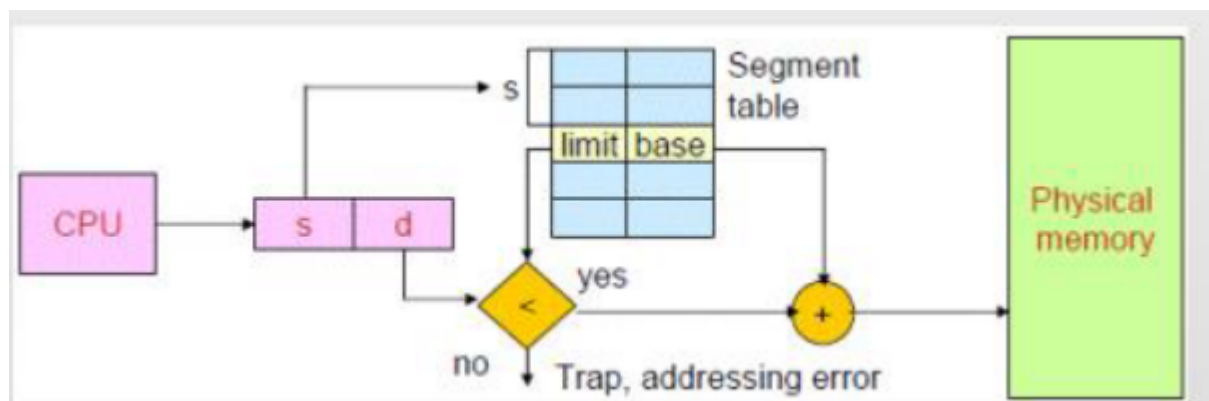
Segmentación

- Esquema que se asemeja a la visión del usuario. El programa es dividido en partes/secciones, donde en cada sección se guardan datos similares.
- Un programa es una colección de segmentos. Un segmento es una unidad lógica como:
 - Programa Ppal, procedures, funciones, variables locales y globales, stack, etc.
 - Cada segmento tiene un registro base y un registro límite.
 - Por lo que se genera una tabla de segmentos con esos dos valores para cada segmento por proceso.
- Puede causar fragmentación EXTERNA.
- Todos los segmentos de un programa pueden NO tener el mismo tamaño (códigos, datos, rutinas). La base y límite del segmento son dinámicos.
- Las direcciones lógicas consisten en 2 partes:
 - Selector de segmento.
 - Desplazamiento dentro del segmento (sobre registro base y límite).
- La segmentación posee ventaja sobre la paginación respecto de: la protección de espacios de memoria y la compartición de bloques de memoria.

b) ¿Qué estructuras adicionales debe poseer el SO para llevar a cabo su implementación?

Una tabla de segmentos, donde por cada segmento hay dos valores: registro base y registro límite.

c)



10)

La segmentación y las particiones dinámicas son dos técnicas de administración de memoria utilizadas en sistemas operativos, y aunque ambas permiten una gestión más flexible de la memoria en comparación con la memoria continua o particiones fijas, tienen similitudes y diferencias clave:

****Similitudes**:**

1. **Gestión Flexible**: Tanto la segmentación como las particiones dinámicas permiten una gestión más flexible de la memoria al dividirla en unidades que se adaptan mejor a las necesidades de los procesos.

2. **Fragmentación Externa**: Ambas técnicas pueden sufrir de fragmentación externa, donde la memoria se divide en segmentos o particiones que no se pueden utilizar de manera eficiente debido a la ubicación de procesos en la memoria.

3. **Protección de Memoria**: Ambos métodos permiten implementar la protección de memoria, ya que es posible asignar permisos específicos a segmentos o particiones individuales para controlar el acceso a la memoria.

Diferencias:

1. **Unidades de Memoria**:

- **Segmentación**: La segmentación divide la memoria en segmentos, que pueden contener partes lógicas o funcionales de un programa. Cada segmento tiene un tamaño y propósito específicos.

- **Particiones Dinámicas**: Las particiones dinámicas dividen la memoria en particiones de tamaños variables que pueden adaptarse a las necesidades de los procesos. Las particiones dinámicas no están vinculadas a ninguna estructura lógica específica.

2. **Fragmentación Interna**:

- **Segmentación**: La segmentación no sufre de fragmentación interna, ya que los segmentos pueden tener tamaños variables y contener información específica.

- **Particiones Dinámicas**: Las particiones dinámicas pueden sufrir de fragmentación interna si un proceso no llena completamente la partición en la que se encuentra, lo que resulta en desperdicio de memoria.

3. **Organización de la Tabla de Asignación**:

- **Segmentación**: Para realizar la traducción de direcciones lógicas a direcciones físicas, se utiliza una tabla de segmentos que mapea los números de segmento a direcciones base y longitudes de segmentos.

- **Particiones Dinámicas**: La administración de particiones dinámicas generalmente se realiza utilizando estructuras de datos como listas enlazadas o tablas que rastrean las particiones ocupadas y desocupadas.

4. **Uso de la Memoria**: En particiones dinámicas, la memoria se asigna a procesos completos, lo que significa que un proceso puede ocupar múltiples particiones o ninguno. En la segmentación, cada segmento se asigna de manera independiente, lo que permite que diferentes procesos compartan segmentos del mismo programa o datos.

En resumen, tanto la segmentación como las particiones dinámicas ofrecen flexibilidad en la gestión de la memoria, pero se utilizan en contextos diferentes y tienen diferencias en la forma en que organizan y gestionan la memoria. La elección entre estas técnicas depende de las necesidades y limitaciones del sistema y las aplicaciones que se ejecutarán.

11)

La paginación y la segmentación son dos técnicas de administración de memoria utilizadas en sistemas operativos, y tienen similitudes y diferencias clave:

****Similitudes**:**

1. ****División de Memoria**:** Ambas técnicas dividen la memoria en unidades más pequeñas para facilitar la gestión y el uso eficiente de la memoria.
2. ****Protección de Memoria**:** Tanto la paginación como la segmentación permiten la implementación de la protección de memoria, lo que significa que se pueden asignar permisos específicos a las unidades de memoria para controlar el acceso.
3. ****Eliminación de Fragmentación Interna**:** Ambos métodos eliminan la fragmentación interna, ya que las unidades de memoria son de tamaño fijo (páginas en la paginación y segmentos en la segmentación) y no hay desperdicio de espacio dentro de estas unidades.

****Diferencias**:**

1. ****Unidades de Memoria**:**

- ****Paginación**:** La paginación divide la memoria en páginas de tamaño uniforme. Cada página tiene un tamaño fijo y es una unidad de administración independiente.
- ****Segmentación**:** La segmentación divide la memoria en segmentos, que pueden tener tamaños variables y están diseñados para contener partes lógicas o funcionales específicas de un programa.

2. ****Fragmentación Externa**:**

- ****Paginación**:** La paginación elimina la fragmentación interna, pero puede sufrir de fragmentación externa, ya que las páginas ocupadas pueden estar dispersas en la memoria y dejar espacios vacíos entre ellas.
- ****Segmentación**:** La segmentación también puede sufrir de fragmentación externa, ya que los segmentos ocupados pueden estar ubicados de manera que generen espacios vacíos entre ellos.

3. ****Traducción de Direcciones**:**

- ****Paginación**:** La traducción de direcciones en la paginación se realiza mediante tablas de páginas, que mapean direcciones lógicas a direcciones físicas. Cada página tiene su propia entrada en la tabla de páginas.
- ****Segmentación**:** La traducción de direcciones en la segmentación se realiza mediante tablas de segmentos, que mapean números de segmento a direcciones base y longitudes de segmentos.

4. ****Uso de la Memoria**:**

- ****Paginación**:** En la paginación, la memoria se asigna en unidades de tamaño fijo (páginas completas). Los procesos pueden ocupar múltiples páginas o fracciones de páginas.

- ****Segmentación****: En la segmentación, la memoria se asigna en unidades de tamaños variables (segmentos). Los procesos pueden compartir segmentos o utilizar segmentos específicos para diferentes partes de su funcionalidad.

En resumen, tanto la paginación como la segmentación son técnicas de administración de memoria que dividen la memoria en unidades más pequeñas, pero difieren en la naturaleza de esas unidades, la forma en que se realiza la traducción de direcciones y la forma en que gestionan la fragmentación. La elección entre estas técnicas depende de las necesidades y características específicas del sistema y las aplicaciones.

12)

i) $(2,1,1) = 1500 + 20 + 1 = 1521$

ii) $(1,3,15) = 500 + 60 + 15 = 575$

iii) $(3,1,10) = 500 + 120 + 10 = 5130$

iv) $(2,3,5) = 1500 + 0 + 5 = 1505$

13)

a)

Con particiones dinámicas y fijas cada proceso está cargado completamente en memoria (en una partición) y en locaciones contiguas, es decir todo el proceso dentro de direcciones de memoria contiguas. Mientras que **la paginación** cada proceso está cargado completamente en memoria pero puede ser que esté en locaciones discontinuas de memoria, es decir una página por un lado, otra página por otro lado. Entonces en base a esto surge la pregunta de por qué tengo que tener todo el proceso cargado en memoria si no se ejecutan todas las instrucciones al mismo tiempo. Aquí entra el concepto de **memoria virtual**. Es decir, yo puedo estar ejecutando un proceso que demandaría más memoria de la memoria real que tengo.

Entonces, un proceso tiene que estar cargado en memoria para ejecutarse, si yo no tengo todas sus páginas cargadas en memoria, cuando quiera ejecutar algo que está en una página la cual dicha página no está cargada en memoria la vas a tener que cargar. Dicha detección de que esa página no está cargada en memoria se llama **fallo de página**, entonces el SO tiene que buscar un marco para cargar esa página, sin embargo, si tienes todos los marcos ocupados aparece el concepto de **página víctima** (aquella página q se

encuentra en un marco la cual va a ser descargada de la memoria para poder ingresar la nueva pagina que se quiere ejecutar). Existen distintos algoritmos para seleccionar la pagina victima. (mejor algoritmo es el que produzca menos fallos de pagina).

La memoria virtual ofrece varios beneficios:

- **Uso Eficiente de la Memoria:** Permite que múltiples procesos se ejecuten en una memoria física limitada al permitir que parte del programa y sus datos se almacenen en el disco y se carguen en la memoria física solo cuando sea necesario. Esto maximiza la utilización de la memoria física.
- **Aislamiento de Procesos:** Cada proceso se ejecuta en su propio espacio de direcciones virtuales, lo que brinda aislamiento y seguridad. Los procesos no pueden acceder directamente a la memoria de otros procesos.
- **Programas Grandes:** Permite la ejecución de programas más grandes de lo que la memoria física podría soportar, ya que solo se necesita cargar en memoria la parte del programa que se está utilizando en un momento dado.
- **Mejor Rendimiento:** La memoria virtual mejora el rendimiento al evitar la sobrecarga de cambiar procesos constantemente entre la memoria principal y el disco. También permite la utilización de técnicas de paginación y segmentación para una gestión eficiente de la memoria.

b)

b) ¿En qué se debe apoyar el SO para su implementación?

En el hardware, pues este debe ser capaz de detectar cuándo una instrucción está tratando de acceder a una dirección que no está en el momento cargada en memoria, y a partir de allí recién el SO podría generar las instrucciones necesarias para atender el fallo. Si esta tarea dependiera exclusivamente del SO, se debería emplear mucho tiempo para detectar si una dirección lógica hace referencia a una porción de programa ya cargada. Aun así, el SO debe dar al hardware la información requerida para llevar a cabo su tarea

c) Información Adicional en las Tablas de Páginas de la paginacion POR DEMANDA:

Cuando se implementa la memoria virtual utilizando la paginación por demanda, además de la dirección física del marco, las tablas de páginas de un proceso deben contar con información adicional, que incluye:

- **Bit de Validez o Presente:** Un bit de validez o presente que indica si una página está actualmente en memoria física (presente) o en el disco (no presente). Es necesario para determinar si es necesario cargar una página desde el disco cuando se accede.
- **Bit de Modificado o Dirty:** Un bit de modificado o dirty que indica si una página ha sido modificada desde que se cargó en memoria. Es esencial para decidir si una página debe escribirse de vuelta al disco antes de ser reemplazada.
- **Bits de Protección y Permisos:** Información sobre los permisos de acceso a la página, que puede incluir bits para lectura, escritura y ejecución.

Estos bits e información adicional son necesarios para permitir que la paginación por demanda funcione eficazmente y garantice la coherencia de la memoria virtual con la memoria física y el disco.

14) b) y c)

La detección de un fallo de página es responsabilidad del hardware del sistema, en particular de la unidad de administración de memoria y del mecanismo de administración de páginas. Cuando un programa en ejecución intenta acceder a una dirección de memoria que no está presente en la memoria física (es decir, está en el almacenamiento secundario o ha sido reubicada en otro lugar), se produce un fallo de página. El hardware detecta este intento de acceso a una página no presente y genera una interrupción llamada "excepción de fallo de página" o "page fault exception."

Una vez que se ha detectado el fallo de página, es responsabilidad del sistema operativo tomar medidas para manejarlo. El SO debe realizar lo siguiente:

Determinar la causa del fallo de página, lo que implica identificar la dirección lógica que provocó el fallo y verificar si esa dirección está asociada a una página que actualmente no está en memoria física.

Realizar la lectura o la carga de la página requerida desde el almacenamiento secundario (como el disco duro) a un marco de memoria física disponible.

Actualizar las tablas de páginas para reflejar la nueva ubicación de la página en memoria física.

Reanudar la ejecución del programa que experimentó el fallo de página, de modo que ahora pueda acceder a la página recién cargada en memoria.

En resumen, el hardware es responsable de la detección de un fallo de página, mientras que la responsabilidad de manejarlo recae en el sistema operativo. El SO se encarga de cargar la página requerida desde el almacenamiento secundario y actualizar las estructuras de administración de páginas para que el programa pueda continuar su ejecución sin problemas.

15)

15) Si se dispone de un espacio de direcciones virtuales de 32 bits, donde cada dirección referencia 1 byte:

a) ¿Cuál es el tamaño máximo de un proceso?

(cant de direcciones) * (tam. de referencia) =

$$2^{32} * 1 \text{ byte} = 4\text{GiB}$$

Cantidad de direcciones:
 $2^{32} = 4.294.967.296$

chat gpt:

Para resolver estas preguntas, debemos considerar las conversiones adecuadas entre tamaños de bits y bytes, y aplicar fórmulas relevantes para el cálculo de la cantidad máxima de páginas y marcos, así como el tamaño máximo de la tabla de páginas. Aquí están los cálculos paso a paso:

i) Tamaño máximo de un proceso (espacio virtual): El espacio de direcciones virtuales es de 32 bits, lo que significa que hay

232

2

32

direcciones de memoria posibles. Como cada dirección referencia 1 byte, el tamaño máximo de un proceso es de

232

2

32

bytes, que es igual a 4 GB.

ii)

b) Si el tamaño de página es de 512 KiB. ¿Cuál es el número máximo de páginas que puede tener un proceso?

en el inciso anterior obtuvimos que el tamaño máximo del proceso es:

$$4.294.967.296 \text{ Bytes} / 1024 = 4194304 \text{ KiB}$$

Tamaño máx. del proceso / tamaño de página =

$$4194304 \text{ KiB} / 512 \text{ KiB} = 8192 \text{ páginas}$$

Dividís el tamaño máximo de un proceso por el tamaño de cada página, y obtener la cantidad máxima de páginas que puede tener un proceso.

chat gpt:

Número máximo de páginas que puede tener un proceso: Si el tamaño de página es de 512 KB, primero convertimos este tamaño a bytes:

$$512 \times 1024$$

512×1024 bytes. Luego, para encontrar el número máximo de páginas, dividimos el tamaño máximo del proceso entre el tamaño de la página:

$$4 \text{ GB} / 512 \text{ KB} = 4 \times 1024 \times 1024 \text{ bytes} / 512 \times 1024 \text{ bytes} = 8192 \text{ páginas}$$

$$512 \text{ KB}$$

$$4 \text{ GB}$$

=

$$512 \times 1024 \text{ bytes}$$

$$4 \times 1024 \times 1024 \text{ bytes}$$

$$= 8192 \text{ p}$$

a

,

ginas

iii)

c) Si el tamaño de página es de 512KiB y se dispone de 256 MiB de memoria real ¿Cuál es el número de marcos que puede haber?

Recordar que las páginas y los marcos deben tener el mismo tamaño. Y que la diferencia está en que las páginas hacen referencia a la memoria lógica, y los marcos a la memoria física.

Siempre pasar a una misma unidad para hacer las cuentas

Página = 512 KiB Memoria = 262144 KiB

$262144 \text{ KiB} / 512 \text{ KiB} = 512 \text{ frames}$

Aca os dividis el tamaño de tu memoria real donde vas a tener marcos donde en cada uno almacenas una pagina, y a toda esa memoria real lo dividis por el tamaño de cada pagina y te da la cantidad de freames/marcos maximo que van a haber.

chat gpt:

Número de marcos que puede haber: Si se tienen 256 MB de memoria real y el tamaño de la página es de 512 KB, primero convertimos el tamaño de la memoria a bytes:

$256 \times 1024 \times 1024$

$256 \times 1024 \times 1024$ bytes. Luego, dividimos la memoria real entre el tamaño de la página para obtener el número de marcos:

$256 \times 1024 \times 1024 \text{ bytes} / 512 \times 1024 \text{ bytes} = 512 \text{ marcos}$

$512 \times 1024 \text{ bytes}$

$256 \times 1024 \times 1024 \text{ bytes}$

=512 marcos

iv)

d) ¿Cuál es el tamaño máximo de un proceso si ahora cada dirección apunta (referencia) a 2 bytes?

Recordar Cantidad de direcciones:

$2^{32} = 4.294.967.296$

Tam. de Proceso = can. de dir. * tam. de cada referencia

$4.294.967.296 * 2 \text{ bytes} = 8589934592 \text{ bytes} = 8\text{GiB}$

chat gpt:

Tamaño máximo de la tabla de páginas de cada proceso: Si se utilizan 2 KB para cada entrada en la tabla de páginas de un proceso, simplemente dividimos el tamaño máximo del

proceso entre el tamaño de cada entrada en la tabla de páginas. Primero, convertimos el tamaño máximo del proceso a bytes:

$4\text{ GB}=4\times1024\times1024\times1024\text{ bytes}$

$4\text{ GB}=4\times1024\times1024\times1024\text{ bytes}$

Luego, dividimos el tamaño máximo del proceso entre el tamaño de cada entrada en la tabla de páginas:

$4\times1024\times1024\times1024\text{ bytes}\div2\times1024\text{ bytes/entrada}=2\times1024\times1024\text{ entradas}$

$2\times1024\text{ bytes/entrada}$

$4\times1024\times1024\times1024\text{ bytes}$

$=2\times1024\times1024\text{ entradas}$

Por lo tanto, el tamaño máximo de la tabla de páginas de cada proceso sería de 2,097,152 (aproximadamente 2 millones) de entradas.

Para expresarlo en MB, dividimos esta cantidad entre

1024

1024

2

:

$2,097,152\div1024=2\text{ MB}$

$2,097,152\div1024$

2

$=2\text{ MB}$

Por lo tanto, el tamaño máximo de la tabla de páginas de cada proceso sería de 2 MB.

16)

Los enfoques para administrar las tablas de páginas son técnicas utilizadas en la gestión de memoria virtual para abordar el problema de la administración de las tablas de páginas de procesos que pueden ser bastante grandes. Aquí están los tres enfoques y cómo funcionan:

1. ****Tablas de Páginas de 1 Nivel****:

- ****Funcionamiento****: En este enfoque, se utiliza una única tabla de páginas para cada proceso. Cada entrada en la tabla de páginas de 1 nivel corresponde a una página en el espacio de direcciones virtuales del proceso. La entrada contiene la dirección base del marco de memoria física asociado a la página correspondiente.

- ****Transformación de Dirección****: Cuando se necesita traducir una dirección virtual a una dirección física, se divide la dirección virtual en dos partes: el número de página y el desplazamiento dentro de la página. Luego, se utiliza el número de página como índice en la tabla de páginas de 1 nivel para encontrar la dirección base del marco físico correspondiente.

2. ****Tablas de Páginas de 2 Niveles****:

- ****Funcionamiento****: En este enfoque, se utilizan dos niveles de tablas de páginas. La tabla de páginas de nivel superior contiene entradas que apuntan a tablas de páginas de nivel inferior. Cada entrada en la tabla de páginas de nivel inferior corresponde a una página en el espacio de direcciones virtuales del proceso y contiene la dirección base del marco de memoria física asociado a la página.

- ****Transformación de Dirección****: Cuando se necesita traducir una dirección virtual a una dirección física, se divide la dirección virtual en tres partes: el índice de la tabla de páginas

de nivel superior, el índice de la tabla de páginas de nivel inferior y el desplazamiento dentro de la página. El índice de la tabla de páginas de nivel superior se utiliza para acceder a la tabla de páginas de nivel inferior correspondiente y encontrar la dirección base del marco físico.

3. ****Tablas de Páginas Invertidas****:

- ****Funcionamiento****: En este enfoque, en lugar de tener una tabla de páginas para cada proceso, se utiliza una tabla de páginas invertida única para todo el sistema. La tabla de páginas invertida contiene una entrada para cada marco de memoria física y registra a qué página y proceso pertenece ese marco.

- ****Transformación de Dirección****: Cuando se necesita traducir una dirección virtual a una dirección física, se busca la entrada correspondiente en la tabla de páginas invertida utilizando el número de página de la dirección virtual. La entrada proporciona la información necesaria para determinar a qué proceso y marco de memoria física pertenece la página, lo que permite la traducción de la dirección.

Cada uno de estos enfoques aborda la administración de las tablas de páginas de manera diferente, y la elección del enfoque puede depender de factores como la eficiencia y el rendimiento del sistema, así como de la cantidad de memoria física disponible.

17)

Bit V (Valid/Invalid):

- Significado: Indica si la entrada de la tabla de páginas es válida o no.
- Funcionamiento: Si el bit V está establecido (1), significa que la entrada de la tabla de páginas es válida y apunta a una ubicación de memoria física válida. Si está despejado (0), la entrada no es válida, y cualquier intento de acceder a la página generará un fallo de página. Si el bit V está despejado (0), la entrada no es válida, y cualquier intento de acceder a la página generará un fallo de página.

Bit R (Referenced):

- Significado: Indica si la página ha sido referenciada (leída o escrita) recientemente.
- Funcionamiento: El sistema operativo puede utilizar este bit para implementar algoritmos de reemplazo de página, donde las páginas menos referenciadas tienen una mayor probabilidad de ser reemplazadas.

Bit M (Modified):

- Significado: Indica si la página ha sido modificada (escrita) desde que se cargó en la memoria.
- Funcionamiento: Se utiliza en algoritmos de reemplazo de página para identificar páginas que han sido modificadas y necesitan ser escritas de vuelta al almacenamiento secundario antes de ser reemplazadas. También se conoce como el bit de suciedad.

MARCO	INICIO-FIN
0	0 – 511
1	512 – 1023
2	1024 - 1535
3	1536 - 2047
4	2048 - 2559
5	2560 - 3071
6	3072 - 3583

7	3584 - 4095
---	-------------

Dir. Lógica div Tam. Página = N.º de Página

Dir. Lógica mod Tam. Página = Desplazamiento

Dir. Física = Inicio o base del frame + desplazamiento

a) 1052

$$1052 / 512 = 2$$

$$1052 \% 512 = 28$$

Fallo de página.

b) 2221:

$$2221 / 512 = 4$$

$$2221 \% 512 = 173$$

Fallo de página.

c) 5499:

$$5499 / 512 = 10$$

$$5499 \% 512 = 379$$

Dirección inválida.

d) 3101:

$$3101 / 512 = 6$$

$$3101 \% 512 = 29$$

Dirección inválida.

20 a)

Peor Rendimiento:

- FIFO (First-In-First-Out): FIFO es susceptible a la anomalía de Belady, donde aumentar el tamaño de la memoria puede conducir a un aumento en la tasa de fallos de página. Puede dar lugar a un rendimiento deficiente en comparación con otros algoritmos.

Moderado Rendimiento:

- Segunda Oportunidad (o Clock): La eficacia de la segunda oportunidad está entre FIFO y LRU. Si bien mejora la anomalía de Belady en comparación con FIFO, aún puede no ser tan eficiente como LRU en ciertos casos.

Rendimiento Moderadamente Bueno:

- LRU (Least Recently Used): LRU tiende a proporcionar un rendimiento razonablemente bueno al elegir la página que no ha sido utilizada

durante el tiempo más largo. Sin embargo, su implementación exacta puede ser costosa en términos de tiempo y recursos.

Mejor Rendimiento:

- OPT (Optimo): OPT es teóricamente el óptimo ya que selecciona la página que no será utilizada durante el tiempo más largo en el futuro. Sin embargo, la implementación práctica de OPT es impracticable ya que requiere conocimiento perfecto del futuro, lo cual no es posible en un entorno real.

b)

Claro, con gusto proporcionaré una explicación más detallada del algoritmo de reemplazo de página llamado "Segunda Oportunidad" (o también conocido como "Clock").

Segunda Oportunidad (Clock):

Funcionamiento:

1. **Estructura de Datos Circular:**

- Mantiene una estructura de datos circular que puede ser implementada como una lista enlazada circular o usando un puntero que avanza circularmente por una matriz.

2. **Bits de Referencia:**

- Asocia a cada página un bit de referencia que se establece en 1 cuando se carga en memoria.

3. **Algoritmo:**

- Cuando ocurre un fallo de página:
- Se examina la página apuntada por el puntero.
- Si el bit de referencia es 1, se le da una "segunda oportunidad" y el puntero avanza al siguiente elemento en la estructura.
- Si el bit de referencia es 0, la página es seleccionada para reemplazo.

4. **Actualización de Bits:**

- A medida que el tiempo avanza, los bits de referencia de las páginas se pueden actualizar según las referencias actuales.
- El bit de referencia se puede restablecer a 0 cuando se selecciona una página para reemplazo o se puede mantener en 1 si la página se referencia nuevamente.

Ejemplo:

Supongamos que tenemos una estructura de datos circular con las páginas A, B, C, D y un puntero apuntando a la página actual. Los bits de referencia inicialmente podrían estar todos en 1.

- Referencias: A, B, A, C, A, D

- Fallo de Página:

1. Se examina la página A (referencia, bit de referencia = 1), se le da una segunda oportunidad y el puntero avanza.

2. Se examina la página B (referencia, bit de referencia = 1), se le da una segunda oportunidad y el puntero avanza.
3. Se examina la página C (referencia, bit de referencia = 1), se le da una segunda oportunidad y el puntero avanza.
4. Se examina la página D (sin referencia, bit de referencia = 0), se selecciona para reemplazo y se carga la nueva página.

Este proceso continúa de manera circular, dando "segundas oportunidades" a las páginas que se han referenciado recientemente y reemplazando las que no se han referenciado.

****Ventajas y Desventajas:****

- **Ventajas:**

- Relativamente eficiente y fácil de implementar.
- Evita ciertos problemas de la política FIFO (First-In-First-Out).

- **Desventajas:**

- Puede no ser tan preciso como algoritmos más avanzados como LRU (Least Recently Used) en algunos casos.

En resumen, Segunda Oportunidad es un algoritmo de reemplazo de página que ofrece un equilibrio entre eficiencia y simplicidad. Proporciona una "segunda oportunidad" a las páginas que han sido referenciadas recientemente antes de ser consideradas para reemplazo.

Después de seleccionar una página como víctima y realizar el reemplazo, la búsqueda de la próxima página víctima comienza a partir del marco siguiente al que se acaba de reemplazar. En el contexto del algoritmo de Segunda Oportunidad (o Clock), el puntero que avanza circularmente se mueve al siguiente marco después de cada reemplazo.

Supongamos que, después de seleccionar D como víctima, el puntero avanza al siguiente marco. En el siguiente fallo de página, la búsqueda comenzará desde ese nuevo marco en el que el puntero se encuentra.

Este enfoque asegura que todas las páginas tengan la oportunidad de ser consideradas para reemplazo antes de que el puntero vuelva a la página que fue reemplazada anteriormente. El movimiento circular del puntero garantiza que se aplique la política de "segunda oportunidad" a cada página en la estructura de datos.

21)b)

b) ¿Es posible utilizar la política de “Asignación Fija” de marcos junto con la política de “Reemplazo Global? Justifique.

No es posible. La asignación fija determina un conjunto específico de marcos que son exclusivos de cada proceso. Si se implementara un reemplazo global, un fallo de página en el proceso A, podría generar la selección de una página víctima del proceso B, con lo que se le estaría quitando un marco a B y asignándoselo a A, rompiendo por completo con la idea de asignación fija de marcos.

25)

****a) ¿Qué es la hiperpaginación (Trashing)?****

La hiperpaginación, también conocida como trashing, es un fenómeno que ocurre en los sistemas de gestión de memoria virtual cuando hay una excesiva actividad de paginación y despaginación. En otras palabras, el sistema operativo está constantemente intercambiando páginas entre la memoria principal y el almacenamiento secundario, lo que resulta en un rendimiento extremadamente deficiente.

Este fenómeno se produce cuando el grado de multiprogramación es demasiado alto, y el sistema intenta ejecutar más procesos de los que puede manejar eficientemente con los recursos de memoria disponibles. Como resultado, los procesos tienen un conjunto de trabajo que no encaja completamente en la memoria principal, y el sistema operativo se ve obligado a intercambiar páginas continuamente, dejando poco tiempo para la ejecución real de procesos.

****b) ¿Cuáles pueden ser los motivos que la causan?***

Los motivos que pueden causar la hiperpaginación incluyen:

1. ****Demanda Excesiva de Memoria:**** Si los procesos activos en el sistema demandan más memoria de la que está disponible físicamente en la memoria principal.
2. ****Grado de Multiprogramación Demasiado Alto:**** Si el sistema intenta ejecutar más procesos simultáneamente de los que puede manejar eficientemente en términos de recursos de memoria.
3. ****Tamaño Inadecuado del Conjunto de Trabajo:**** Si el "working set" de los procesos es demasiado grande en comparación con la cantidad de memoria disponible.
4. ****Falta de Políticas de Reemplazo de Página Eficientes:**** Si las políticas de reemplazo de página no son eficientes, pueden contribuir a la hiperpaginación.

****c) ¿Cómo la detecta el SO?***

El sistema operativo puede detectar la hiperpaginación mediante el seguimiento de métricas y señales que indican un alto grado de actividad de paginación y despaginación. Algunas señales de que el sistema está experimentando trashing incluyen:

- ****Alto Índice de Fallos de Página:**** Un aumento significativo en el número de fallos de página indica que el sistema está luchando por mantener las páginas necesarias en la memoria principal.

- ****Baja Utilización de la CPU:**** A pesar de un alto grado de multiprogramación, si la CPU está inactiva la mayor parte del tiempo debido a la actividad de paginación, es una señal de trashing.

- ****Interrupciones de E/S Constantes:**** Si el sistema está constantemente realizando intercambios de páginas debido a la demanda de E/S, puede indicar la presencia de trashing.

****d) Una vez que lo detecta, ¿qué acciones puede tomar el SO para eliminar este problema?****

Para resolver el problema de la hiperpaginación, el sistema operativo puede tomar diversas acciones, entre las que se incluyen:

1. ****Ajustar el Grado de Multiprogramación:**** Reducir el número de procesos en ejecución simultánea para que el sistema pueda manejarlos más eficientemente con los recursos de memoria disponibles.

2. ****Optimizar las Políticas de Reemplazo de Página:**** Mejorar las políticas de reemplazo de página para minimizar la necesidad de realizar constantes intercambios de páginas.

3. ****Aumentar la Memoria Principal:**** Si es posible, aumentar la cantidad de memoria física disponible para reducir la necesidad de paginación constante.

4. ****Ajustar el Tamaño del Conjunto de Trabajo:**** Modificar el tamaño del "working set" de los procesos para que se ajuste más eficientemente a la memoria principal disponible.

5. ****Priorizar Procesos Críticos:**** Dar prioridad a la ejecución de procesos críticos para evitar que sean víctimas del trashing.

La resolución de la hiperpaginación implica un equilibrio cuidadoso entre la cantidad de procesos en ejecución, la cantidad de memoria disponible y la eficiencia de las políticas de gestión de memoria.

29)

a)

La anomalía de Belady se refiere a una situación inusual que puede ocurrir en algoritmos de reemplazo de página en sistemas de gestión de memoria virtual. Esta anomalía se caracteriza por un comportamiento aparentemente contrario a la intuición, donde aumentar el tamaño de la memoria principal (la cantidad de marcos de página disponibles) puede resultar en un aumento del número de fallos de página en lugar de una reducción.

En términos más simples, la anomalía de Belady ocurre cuando agregar más memoria a un sistema no mejora el rendimiento y, en cambio, podría empeorar. Esta situación contradice

la expectativa común de que al aumentar la cantidad de memoria disponible, se deberían experimentar menos fallos de página.

La anomalía de Belady es más notable en algoritmos de reemplazo de página que no tienen en cuenta el futuro (por ejemplo, algoritmos óptimos), ya que en estos casos, el comportamiento del algoritmo puede depender fuertemente de la disposición específica de las referencias a las páginas.

En resumen, la anomalía de Belady es una observación inusual en la gestión de memoria virtual, donde añadir más memoria a un sistema puede resultar en un aumento, en lugar de una disminución, en la tasa de fallos de página, según el algoritmo de reemplazo de página utilizado

30)

Para analizar cuántos fallos de página ocurren al ejecutar el programa, es necesario tener en cuenta cómo se accede a las matrices y cómo se manejan las páginas en la memoria. En este caso, se asume que cada matriz (A, B, y C) requiere 16 páginas y que el sistema utiliza un working set de 4 marcos.

El código dado realiza operaciones en las matrices A, B y C mediante dos bucles anidados. Vamos a analizar la ejecución del programa:

1. ****Inicialización:****

- Cada matriz ocupa 16 páginas, y el sistema tiene un working set de 4 marcos.
- El programa ocupa un marco y los datos de las matrices ocupan los otros 3 marcos.

2. ****Acceso a las Matrices:****

- En cada iteración del bucle externo (controlado por `j`), se accede a una columna completa de las matrices A, B y C.
- Cada vez que se accede a una nueva columna, puede resultar en un fallo de página porque las páginas de esa columna pueden no estar en memoria.

3. ****Cambio de Columna:****

- Como el working set es de 4 marcos y cada matriz ocupa 16 páginas, al cambiar de columna en una matriz, es probable que una nueva página (o varias) se cargue en memoria, lo que puede causar fallos de página.

4. ****Resumen:****

- Dado que cada matriz ocupa 16 páginas y el working set es de 4 marcos, al cambiar entre columnas de las matrices es probable que ocurran fallos de página.
- En cada iteración del bucle externo (controlado por `j`), hay posibilidad de que ocurran fallos de página al acceder a las nuevas columnas de las matrices.

5. ****Total de Fallos de Página:****

- En cada iteración del bucle externo, se accede a nuevas columnas, lo que puede resultar en múltiples fallos de página.
- En total, el número de fallos de página será afectado por la cantidad de iteraciones del bucle externo (tamaño de la matriz en la dimensión `j`).

Teniendo en cuenta estos puntos, el número total de fallos de página dependerá del tamaño de la matriz (`Size`). Cada vez que se accede a una nueva columna, hay posibilidad de

fallos de página, y como hay 64 columnas (tamaño de la matriz en la dimensión `j`), el número total de fallos de página será proporcional a `Size`. Específicamente, el número de fallos de página podría ser aproximadamente `Size * 16` si cada cambio de columna resulta en un fallo de página.

b)

Para minimizar el número de fallos de páginas, puedes utilizar la técnica de "localidad espacial" para acceder a los datos de manera más eficiente. Dado que estás utilizando paginación por demanda y trabajas con matrices, modificar el orden de acceso a las matrices puede ayudar a reducir los fallos de página.

En el código proporcionado, las matrices A, B y C se acceden por columnas en el bucle externo controlado por `j`. Para minimizar los fallos de página, puedes cambiar el orden de acceso para que las matrices se accedan por filas, ya que las páginas contiguas de una fila estarán más propensas a estar en el mismo marco.

Aquí hay una versión modificada del código:

```
```c
#define Size 64
int A[Size][Size], B[Size][Size], C[Size][Size];
int register i, j;

for (i = 0; i < Size; i++)
 for (j = 0; j < Size; j++)
 C[i][j] = A[i][j] + B[i][j];
...
```
```

Con este cambio, el bucle externo ahora controla la variable `i` en lugar de `j`, y el bucle interno controla `j`. Esto implica que ahora accedes a las matrices por filas en lugar de por columnas.

Este cambio debería reducir los fallos de página porque accederás a datos de manera más contigua en la memoria, lo que mejora la localidad espacial y disminuye la probabilidad de fallos de página al cambiar entre columnas.

Sin embargo, la cantidad exacta de fallos de página dependerá de la implementación específica del sistema operativo y su política de reemplazo de página. La idea es reducir la necesidad de cargar nuevas páginas al acceder a datos contiguos en la memoria.

31)

a) Considerando una ventana $\Delta=5$, indique cual seria el conjunto de trabajo de los procesos A y B en el instante 24 (WSA(24) y WSB(24))

WSA(24) = 4, 5, 6

WSB(24) = 2, 3, 5, 6

b) Considerando una ventana $\Delta=5$, indique cual seria el conjunto de trabajo de los procesos A y B en el instante 60 (WSA(60) y WSB(60))

WSA(60) = 1, 2, 3, 4, 5

WSB(60) = 3, 4, 5

c) Para el los WS obtenidos en el inciso a), si contamos con 8 frames en el sistema ¿Se puede indicar que estamos ante una situación de trashing? ¿Y si contáramos con 6 frames?

Con 8 frames no, porque la suma de los WS es 7, menor a la cantidad de frames disponibles.

Con 6 si porque se necesitan mas frames de los disponibles.

d) Considerando únicamente el proceso A, y suponiendo que al mismo se le asignaron inicialmente 4 marcos, donde el de reemplazo de paginas es realizado considerando el algoritmo FIFO. ¿Cuál será la tasa de fallos en el instante 38 de páginas suponiendo que la misma se calcula contando los fallos de páginas que ocurrieron en las últimas 10 unidades de tiempo?

PF=6

e) Para el valor obtenido en el inciso d), si suponemos que el S.O. utiliza como limites superior e inferior de tasa de fallos de paginas los valores 2 y 5 respectivamente ¿Qué acción podría tomar el S.O. respecto a la cantidad de marcos asignados al proceso?

Asignarle un marco mas (no seria muy útil ya que es FIFO y puede tener anomalia de Belady) o bajar el grado de multiprogramación, ya que se está generando hiperpaginación.

PARCIAL 2019:

4a)

2 elevado a la 32 = 4.294.967.296 bytes. cant direcciones

cantidad total que referencian 1 byte.

4.294.967.296 bytes * 1 byte = 4.096 Megabytes.

Rta: El tamaño maximo de un proceso esta dado por la cantidad de direcciones posibles multiplicado por la cantidad de bytes q referencian.

b) Si yo tengo un proceso, en el cual su tamaño maximo es de 4.096 Megabytes, lo tenes que dividir por el tamaño de una pagina y obtenes cuantas paginas puede llegar a tener como maximo.

4.294.967.296 / 1024 = 4.194.304 paginas.

c) 4.194.304paginas * 2 bytes = 8.388.608 bytes = 8 Megabytes.

d) $5450 \text{ bytes} / 1024 \text{ bytes} = 6 \text{ paginas ocuparia.}$