Introducción a los Sistemas Operativos
Práctica V

1.-Explique a que hacen referencia los siguientes términos:

Dirección Lógica o Virtual

Una dirección lógica es una referencia a una posición de memoria independiente de la asignación actual de datos a memoria; se debe hacer una traducción a dirección física antes de poder realizar un acceso a memoria.

Dirección Física

Una dirección física o dirección absoluta, es una posición real en memoria principal.

2.- En la técnica de Particiones Múltiples, la memoria es divida en varias particiones y los procesos son ubicados en estas, siempre que el tamaño del mismo sea menor o igual que el tamaño de la partición. Al trabajar con particiones se pueden considerar 2 métodos (independientes entre si): Particiones Fijas o Particiones Dinámicas

a) Explique como trabajan estos 2 métodos. Cite diferencias, ventajas y desventajas.

Particiones Fijas
La memoria se divide en particiones o regiones de tamaño Fijo (pueden ser todas del mismo tamaño o no) . Cada una de estar particiones o regiones aloja un proceso. Los procesos se colocan en las particiones de acuerdo algún criterio (First Fit, Best Fit, Worst Fit, Next Fit.

Particiones dinámicas:
Las particiones varían en tamaño y en número . Cada particion aloja un proceso. A diferencia del método anterior, las particiones se generan en forma dinámica del tamaño justo que necesita el proceso

Problemas
Fragmentación La fragmentación se produce cuando una localidad de memoria no puede ser utilizada por no encontrarse en forma contigua. 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

Fragmentación Interna Se produce en el esquema de particiones Fijas Es la porción de la partición que queda sin utilizar

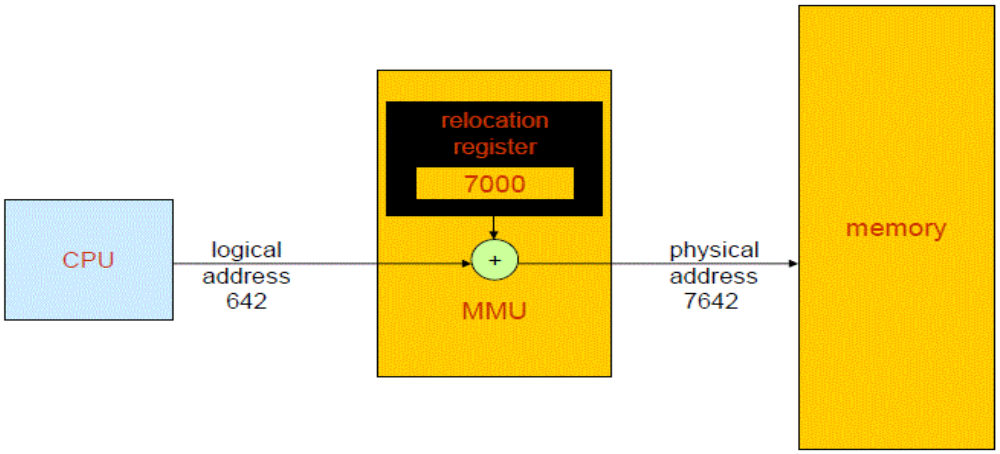
Fragmentación Externa Se produce en el esquema de particiones dinámicas . Son huecos que van quedando en la memoria a medida que los procesos finalizan .

Técnica	Descripción	Ventajas	Desventajas
Partición fija	La memoria principal se divide en un conjunto de particiones fijas durante la generación del sistema. Un proceso se puede cargar en una partición de mayor o igual tamaño.	Sencilla de implementar; poca sobrecarga del sistema operativo.	Empleo ineficiente de la memoria debido a la fragmentación interna; el número de procesos activos es fijo.
Partición Dinámica	Las particiones se crean dinámicamente, de forma que cada proceso se carga en una partición de exactamente el mismo tamaño que el proceso.	No hay fragmentación interna; uso más eficiente de la memoria principal.	Uso ineficiente del procesador debido a la necesidad de compactación para contrarrestar la fragmentación externa.

b) ¿Qué información debe disponer el SO para poder administrar la memoria con estos métodos?

El SO debe matener información sobre el tamaño de los procesos, en que dirección empiezan y terminan para particiones dinámicas. Para las particiones fijas deberá manterner información sobre el sector o particion donde se encuentra un proceso. EL SO también deberà tener noción de cuales particiones o sectores se encuentran ocupados y cuales libres, y cuanta memoria libre queda. En caso de que se utilize la compactación, el SO también deberá llevar cuenta del particionado, para poder tomar la determinación de cuando es necesario compactar para liberar memoria.

c) Realice un grafico indicado como realiza el SO la transformación de direcciones lógicas a direcciones físicas.



3.- Al trabajar con particiones fijas, los tamaños de las mismas se pueden considerar Particiones de igual tamaño y Particiones de diferente tamaño. Cite ventajas y desventajas de estos 2 métodos.

particiones de igual tamaño

En este caso, cualquier proceso cuyo tamaño sea menor o igual que el tamaño de la partición puede cargarse en cualquier partición libre. Si todas las particiones están ocupadas y no hay procesos residentes en estado Listo o Ejecutando, el sistema operativo puede sacar un proceso de alguna de las particiones y cargar otro proceso de forma que haya trabajo para el procesador.

Las particiones fijas de igual tamaño plantean dos dificultades:
•Un programa puede ser demasiado grande para caber en la partición. En este caso, el programador debe diseñar el programa mediante superposiciones, para que sólo una parte del programa esté en memoria principal en cada instante. Cuando se necesita un módulo que no está presente, el programa de usuario debe cargar dicho módulo en la partición del programa, superponiéndose a los programas y datos que se encuentren en ella.
• El uso de memoria principal es extremadamente ineficiente. Cualquier programa, sin importar lo pequeño que sea, ocupará una partición completa. En el ejemplo, puede haber un programa que ocupe menos de 128Kb de memoria y, aún así, ocuparía una partición de 512Kb cada vez que se cargase. Este fenómeno, en el que se malgasta el espacio interno de una partición cuando el bloque de datos cargado sea más pequeño que la partición, se denomina fragmentación interna.

Pueden reducirse, aunque no solventarse, ambos problemas, por medio del empleo de par ticiones de tamaños distinto.

Con particiones de distintos tamaños, hay dos maneras posibles de asignar los procesos a las particiones. La forma más simple es asignar cada proceso a la partición más pequeña en la que quepa'. En este caso, hace falta una cola de planificación para cada partición, que albergue los procesos expulsados cuyo destinado es dicha partición. La ventaja de este enfoque es que los procesos están siempre asignados de forma que se minimiza la memoria desperdiciada dentro de cada partición.
Sin embargo, aunque esta técnica parece óptima desde el punto de vista de una partición individual, no lo es desde el punto de vista del sistema global.

•El número de particiones especificadas en el momento de la generación del sistema limita el número de procesos activos (no suspendidos) del sistema.
•Puesto que los tamaños de partición se programan en el momento de la generación del sistema, los trabajos pequeños no hacen un uso eficiente del espacio de las particiones. En un entorno en el que los requisitos básicos de almacenamiento de todos los procesos se conocen de antemano, puede ser una técnica razonable, pero, en la mayoría de los casos, ineficiente.

4.- Fragmentación

Ambos métodos de particiones presentan el problema de la fragmentación: Fragmentación Interna (Para el caso de Particiones Fijas) Fragmentación Externa (Para el caso de Particiones Dinámicas)

a) Explique a que hacen referencia estos 2 problemas

EN el metodo de particiones fijas El uso de memoria principal es extremadamente ineficiente. Cualquier programa, sin mi portar lo pequeño que sea, ocupará una partición completa. En el ejemplo, puede haber un programa que ocupe menos de 128Kb de memoria y, aún así, ocuparía una partición de 512Kb cada vez que se cargase. Este fenómeno, en el que se malgasta el espacio interno de una partición cuando el bloque de datos cargado sea más pequeño que la partición, se denomina **fragmentación interna.**

El método de particiones dinámicas comienza bien, pero, finalmente, desemboca en una situación en la que hay un gran número de huecos pequeños en memoria. Conforme pasa el tiempo, la memoria comienza a estar más fragmentada y su rendimiento decae. Este fenómeno se denomina **fragmentación externa** y se refiere al hecho de que la memoria externa a todas las particiones se fragmenta cada vez más, a diferencia de la fragmentación interna, que se comentó anteriormente.

b) El problema de la Fragmentación Externa es posible de subsanar. Explique una técnica que evite este problema.

Una técnica para superar la fragmentación externa es la compactación: De vez en cuando, el sistema operativo desplaza los procesos para que estén contiguos de forma que toda la memoria libre quede junta en un bloque. La dificultad de la compactación está en que es un procedimiento que consume tiempo, por lo que desperdicia tiempo del procesador. La compactación necesita de la capacidad de reubicación dinámica. Es decir, se debe poder mover un programa de una región a otra de memoria principal sin invalidar las referencias a memoria del programa.

5.- Paginación

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

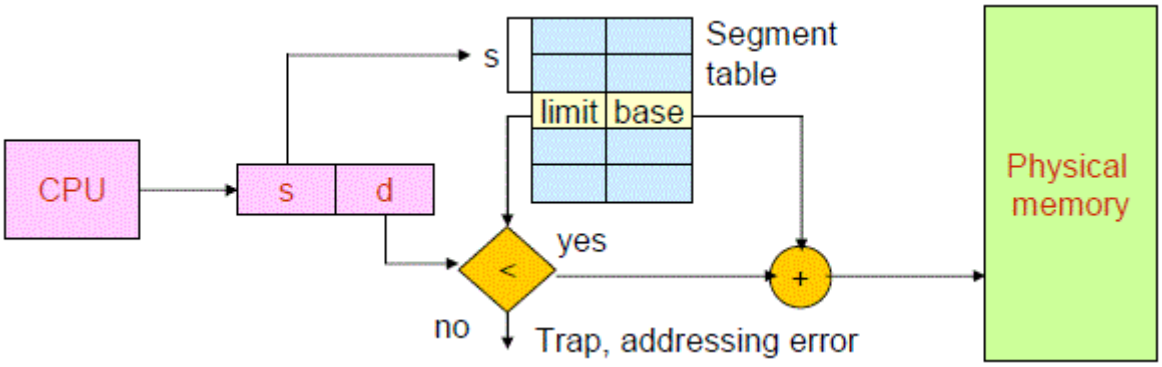
La memoria se divide en porciones de igual tamaño llamadas marcos y el espacio de direcciones de los procesos se divide en porciones de igual tamaño denominados páginas . El tamaño de los marcos es igual al tamaño de las páginas (generalmente 512 Bytes)

Supóngase, no obstante, que la memoria principal se encuentra particionada en trozos iguales de tamaño fijo relativamente pequeños y que cada proceso está dividido también en pequeños trozos de tamaño fijo y del mismo tamaño que los de memoria. En tal caso, los trozos del proceso, conocidos como páginas, pueden asignarse a los trozos libres de memoria, conocidos como marcos o marcos de página. En este apartado se verá que el espacio malgastado en memoria para cada proceso por fragmentación interna consta sólo de una fracción de la última página del proceso. Además, no hay fragmentación externa.
En un instante dado, algunos de los marcos de memoria están en uso y otros están libres. El sistema operativo mantiene una lista de los marcos libres.

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

el sistema operativo mantiene una tabla de páginas para cada proceso. La tabla de páginas muestra la posición del marco de cada página del proceso.
Dentro del programa, cada dirección lógica constará de un número de página y de un desplazamiento dentro de la página. Recuérdese que, en el caso de la partición simple, una dirección lógica era la posición de una palabra relativa al comienzo del programa; el procesador realizaba la traducción a dirección física. Con paginación, el hardware del procesador también realiza la traducción de direcciones lógicas a físicas. Ahora, el procesador debe saber cómo acceder a la tabla de páginas del proceso actual. Dada una dirección lógica (número de página, desplazamiento), el procesador emplea la tabla de páginas para obtener una dirección física (número de marco, desplazamiento).

c) Explique, utilizando gráficos, como son transformadas las direcciones lógicas en físicas.



d) En este esquema: ¿Se puede producir fragmentación (interna y/o externa)?

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

Similitudes: se realiza una división del espacio asignado para cargar los procesos/páginas del mismo tamaño y a su vez, el S.O. posee tablas para acceder a ellos.
La diferencia principal es que, en paginación, el S.O. posee una tabla para cada proceso, en cambio en particiones fijas tiene una tabla para todos los procesos.

7.- Suponga un sistema donde la memoria es administrada mediante la técnica de paginación, y donde:

- **El tamaño de la página es de 512 bytes**
- **Cada dirección de memoria referencia 1 byte.**

•Los marcos en memoria principal de encuentran desde la dirección física 0.

Suponga además un proceso con un tamaño 2000 bytes y con la siguiente tabla de páginas:

Página	Marco
0	3
1	5
2	2
3	6

a) Realice los gráficos necesarios (de la memoria, proceso y tabla de paginas) en el que reflejen el estado descrito.

Pagina Marco 0 3 1 5 2 2 3 6
Marco 0 3 1 5 2 2 3 6
0 3 1 5 2 2 3 6
3 1 5 2 2 3 6
1 5 2 2 3 6
5 2 2 3 6
2 2 3 6
2 3 6
3 6
6

Marco Inicio/Fin 0 0/511 1 512/1023 2 1024/1535 3 1536/2047 4 2048/2559 5 2560/3071 6 3072/3584
Inicio/Fin 0 0/511 1 512/1023 2 1024/1535 3 1536/2047 4 2048/2559 5 2560/3071
0 0/511 1 512/1023 2 1024/1535 3 1536/2047 4 2048/2559 5 2560/3071 6 3072/3584
0/511 1 512/1023 2 1024/1535 3 1536/2047 4 2048/2559 5 2560/3071 6 3072/3584
1 512/1023 2 1024/1535 3 1536/2047 4 2048/2559 5 2560/3071 6 3072/3584
512/1023 2 1024/1535 3 1536/2047 4 2048/2559 5 2560/3071 6 3072/3584
2 1024/1535 3 1536/2047 4 2048/2559 5 2560/3071 6 3072/3584
1024/1535 3 1536/2047 4 2048/2559 5 2560/3071 6 3072/3584
3 1536/2047 4 2048/2559 5 2560/3071 6 3072/3584
1536/2047 4 2048/2559 5 2560/3071 6 3072/3584
4 2048/2559 5 2560/3071 6 3072/3584
2048/2559 5 2560/3071 6 3072/3584
5 2560/3071 6 3072/3584
2560/3071 6 3072/3584
6 3072/3584
3072/3584

b) Indicar si las siguientes direcciones lógicas son correctas y en caso afirmativo indicar la dirección física a la que corresponden:

35

35 div 512 = 0 -> 35 corresponde a la página 0, en el marco 3.
desplazamiento: 35 mod 512= 35 -> **dirección fìsica: 1536 + 35 : 1571**

0

0 div 512: 0 ? 0 corresponde a la pagina 0 en el marco 3
desplazamiento: 0 mod 512: 0 -> **direccion física: 1536**

512

512 div 512: 1 ? pagina uno: marco 5
desplazamiento 512 mod 512: 0 -> **dirección física: 2560**

1325

1325 div 512: 2 ? pagina dos: marco 2
desplazamiento 1325 mod 512: 301 ? **dirección física: 1024 + 301: 1325**

2051

2051 div 512: 4 -> pagina cuatro. **DIRECCIÓN INVALIDA!**

602

602 div 512: 1 -> página uno: marco cinco
desplazamiento 602 mod 512: 90 -> **direccion física: 2560 + 90: 2650**

c) Indicar, en caso de ser posible, las direcciones lógicas del proceso que se corresponden si las siguientes direcciones físicas:

509

509 div 512 : 0 ? NO VALIDO

3215

3215 div 512: 6 ? marco 6 : pagina 3
desplazamiento 3215 mod 512: 143 ? **direccion logica 512 + 143: 655**

1500
v) 1024
iii) 0
vi) 2000

d) ¿Indique, en caso que se produzca, la fragmentación (interna y/o externa)?

8. Considere un espacio lógico de 8 páginas de 1024 bytes cada una, mapeadas en una memoria física de 32 marcos

a. ¿Cuántos bits son necesarios para representar una dirección lógica?

Las direcciones lógicas van desde 0 hasta 8*1024 bytes, es decir de 2^3 * 2^10 = 2^13. Es decir que se necesitan 13 bits para representar una dirección lógica.

a. ¿Cuántos bits son necesarios para representar una dirección física?

Siguiendo la misma lógica que en el caso anterior: las direcciones van de 0 hasta 32*1024. Entonces 2^5 * 2^10 = 2^15, se necesitan 15 bits para representar una dirección física.

9. Segmentación

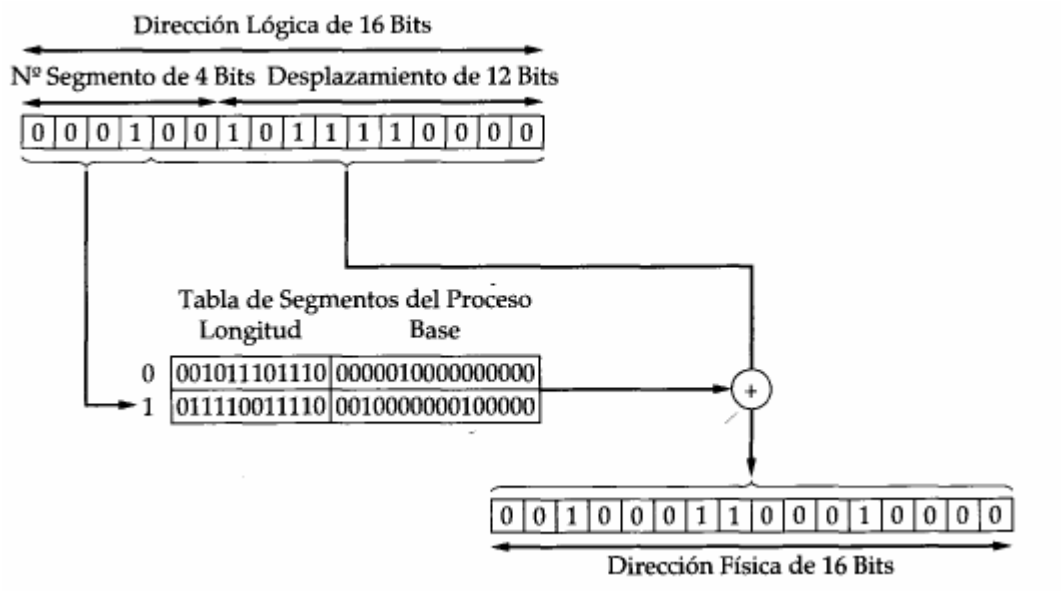
a. Explique cómo trabaja este método de asignación de memoria

Otro modo de subdividir el programa es la segmentación. En este caso, el programa y sus datos asociados se dividen en un conjunto de **segmentos**. No es necesario que todos los segmentos de todos los programas tengan la misma longitud, aunque existe una longitud máxima de segmento. Como en la paginación, una dirección lógica segmentada consta de dos partes, en este caso un número de segmento y un desplazamiento. Como consecuencia del empleo de segmentos de distinto tamaño, la segmentación resulta similar a la partición dinámica.

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

De forma análoga a la paginación, un esquema de segmentación simple hará uso de una tabla de segmentos para cada proceso y una lista de bloques libres en memoria principal. Cada entrada de tabla de segmentos tendría que contener la dirección de comienzo del segmento correspondiente en memoria principal. La entrada deberá proporcionar también la longitud del segmento para asegurar que no se usan direcciones no válidas. Cuando un proceso pasa al estado Ejecutando, se carga la dirección de su tabla de segmentos en un registro especial del hardware de gestión de memoria.

c. Explique, utilizando gráficos, como son transformadas las direcciones lógicas en físicas



d. En este esquema: ¿Se puede producir fragmentación (interna y/o externa)?

La segmentación elimina la fragmentación interna, pero, como la partición dinámica, sufre de fragmentación externa. Sin embargo, debido a que los procesos se dividen en un conjunto de partes más pequeñas, la fragmentación externa será menor.

10. Cite similitudes y diferencias entre la técnica de segmentación y la de particiones dinámicas

Como consecuencia del empleo de segmentos de distinto tamaño, la segmentación resulta similar a la partición dinámica. En ausencia de un esquema de superposición o del uso de memoria virtual, sería necesario cargar en memoria todos los segmentos de un programa para su ejecución. La diferencia, en comparación con la partición dinámica, radica en que, con segmentación, un programa puede ocupar más de una partición y éstas no tienen por qué estar contiguas.

Mientras que la paginación es transparente al programador, la segmentación es generalmente visible y se proporciona como una comodidad para la organización de los programas y datos. Normalmente, el programador o el compilador asigna los programas y los datos a diferentes segmentos. En aras de la programación modular, el programa o los datos pueden ser divididos de nuevo en diferentes segmentos. El principal inconveniente de este servicio es que el programador debe ser consciente de la limitación de

tamaño máximo de los segmentos.

11. Cite similitudes y diferencias entre la técnica de paginación y segmentación

Uhm.

12. Dado un So que adminitra la memoria por medio de segmentación pagina y teniéndose disponibles las siguientes tablas:

Tabla de Segmentos

Núm. Seg.	Dir. base
1	500
2	1500
3	5000

Tabla de Paginas

Nro. Segmento	Nro. Pagina	Direc. Base
1	1	40
	2	80
	3	60
2	1	20
	2	25
	3	0
3	1	120
	2	150

Indicar las direcciones físicas correspondientes a las siguientes direcciones lógicas (segmento, página, desplazamiento):

i. (2,1,1) = 1500 + 20 + 1 = 521
ii. (1,3,15) = 500 + 60 + 15 = 575
iii. (3,1,10) = 5000 + 120 + 10 = 5170
iv. (2,3,5) = 1500 + 0 + 5 = 1505

13. Memoria Virtual

a. Describa que beneficios introduce este esquema de administración de la memoria

Se pueden conservar más procesos en memoria principal. Puesto que se van a cargar sólo algunos fragmentos de un proceso particular, habrá sitio para más procesos. Esto conduce a una utilización más eficiente del procesador, puesto que es más probable que, por lo menos, uno de los numerosos procesos esté en estado Listo en un instante determinado.

Es posible que un proceso sea más grande que toda la memoria principal. Se elimina así una de las limitaciones más notorias de la programación. Sin el esquema que se ha expuesto, un programador debe ser consciente de cuánta memoria tiene disponible. Si el programa que está escribiendo es demasiado grande, el programador debe idear formas de estructurar el programa en fragmentos que puedan cargarse de forma separada con algún tipo de estrategia de superposición. Con una memoria virtual basada en paginación o segmentación, este trabajo queda para el sistema operativo y el hardware. En lo que atañe al programador, se las arregla con una memoria enorme, dependiendo del tamaño de almacenamiento en disco. El sistema operativo cargará automáticamente en memoria principal los fragmentos de un proceso cuando los necesita.

La memoria virtual permite una multiprogramación muy efectiva y releva al usuario de las rígidas e innecesarias restricciones de la memoria principal. La tabla 7.1 resume las características de la paginación y la segmentación, con y sin memoria virtual.

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

El hardware debe soportar paginación por demanda (y/o segmentación). También, es necesario un dispositivo de memoria secundaria (disco) que dé el apoyo para almacenar las secciones del proceso que no están en la memoria Principal (área de intercambio). Finalmente, el SO debe ser capaz de manejar el movimiento de las páginas (o segmentos) entre la memoria principal y la secundaria.

c. Al implementar esta técnica utilizando paginación por demanda, las tablas de páginas de una proceso deben contar con información adicional además del marco donde se encuentra la página. ¿Cuál es esta información?¿Por qué es necesaria?

En el estudio de la paginación simple se indicó que cada proceso tiene su propia tabla de páginas y que, cuando carga todas sus páginas en memoria principal, se crea y carga en memoria principal una tabla de páginas. Cada entrada de la tabla de páginas contiene el número de marco de la página correspondiente en memoria principal. Cuando se considera un esquema de memoria virtual basado en la paginación se necesita la misma estructura, una tabla de páginas. Nuevamente, es normal asociar una única tabla de páginas con cada proceso. En este caso, sin embargo, las entradas de la tabla de páginas pasan a ser más complejas (figura 7.2a). Puesto que sólo algunas de las páginas de un proceso pueden estar en memoria principal, se necesita un bit en cada entrada de la tabla para indicar si la página correspondiente está presente (P) en memoria principal o no lo está. Si el bit indica que la página está en memoria, la entrada incluye también el número de marco para esa página.
Otro bit de control necesario en la entrada de la tabla de páginas es el bit de modificación (M), para indicar si el contenido de la página correspondiente se ha alterado desde que la página se cargó en memoria principal. Si no ha habido cambios, no es necesario escribir la página cuando sea sustituida en el marco que ocupa actualmente. Puede haber también otros bits de control. Por ejemplo, si la protección o la compartición se gestionan a nivel de página, se necesitarán más bits con tal propósito.

14. Fallos de página:

a. ¿Cuándo se producen?

Ocurre cuando se intenta utilizar una dirección página que no se encuentra en la memoria principal

b. ¿Quién es el responsable de detectar un fallo de página?

El hardware que detecta esta situación es la unidad de manejo de memoria del procesador. El Software de manejo de excepciones que maneja el fallo de página es parte del sistema operativo. El Sistema operativo intenta manejar el fallo de página haciendo la página accesible en una ubicación en la memoria física o termina el programa en caso que el fallo fuese un acceso ilegal.

c. Describa las acciones que emprende el SO cuando se produce un fallo de página.

1. El HW detecta la situación y genera una excepción al S.O.
2. El S.O. coloca al proceso en estado de “Bloqueado”
3. El S.O. busca un Marco Libre y genera una operación de E/S al disco para subir a dicho marcho la porción (página) del proceso que se necesita.
4. El SO puede asignarle la CPU a otro proceso
5. La E/S se realizará y avisará mediante interrupción su finalización.
6. Cuando la operación de E/S finaliza el SO:
    - Actualiza la tabla de páginas
    - Coloca el Bit V en 1
    - Coloca la dirección base del Marco donde se colocó la página
7. El proceso que generó el Fallo de Página vuelve a estado de Listo

15 Direcciones:

a. Si se dispone de un espacio de direcciones virtuales de 32 bits, donde cada dirección referencia a 1 byte

i. ¿Cuál es el tamaño máximo de un proceso?

2^32 = 4.294.967.296 bytes

ii. Si el tamaño de página es de 512kb ¿Cuál es el número máximo de páginas que puede tener un proceso?

Cantidad de direcciones (Cada dirección referencia a un byte)

2^32 = 4.294.967.296

Tamaño de página:

512 * 1024 = 524288

Entonces:

2^32 / 2^19 = 2^13 = **8192** páginas posibles para un un proceso

iii. Si el tamaño de página es de 512kb y se disponen de 256 MB de memoria real ¿Cuál es el número de marcos que puede haber?

Tamaño de página

512 * 1024 = 5242288 = 2^13

Memoria real:

256 * 1048576 = 2^8 * 2^20 = 2^28

Entonces:

2^20/2^13 = 2^7 = **128** marcos posibles

iv. Si se utilizaran 2kb para cada entrada de la tabla de páginas de un proceso: ¿Cuál sería el tamaño máximo de la tabla de páginas de cada proceso?

8129 * 1024 * 2 = 16648192

16. Como se vio en el ejercicio anterior, la tabla de páginas de un proceso puede alcanzar un tamaño considerablemente grande, que incluso, no podría almacenarse de manera completa en la memoria real. Es por esto que el SO también realiza paginación sobre las tablas de páginas.

Existen varios enfoques para administrar las tablas de páginas: Tablas de páginas de un nivel, tablas de páginas de dos niveles, tablas de páginas invertidas. Explique brevemente como trabajan estos enfoques e indique como se realiza la trasformación de la dirección virtual en dirección física

Tablas de página de un nivel

Tabla común, lineal.

Tablas de página de dos niveles

Algunos procesadores usan un esquema a dos niveles para organizar grandes tablas de páginas. En este esquema, hay un directorio de páginas en el que cada entrada señala a una tabla de páginas. Así pues, si la longitud del directorio de páginas es X y la longitud máxima de una tabla de páginas es Y, un proceso puede estar formado por hasta X x Y páginas. Normalmente, la longitud máxima de una tabla de páginas está limitada a una página.

Tablas de páginas invertidas

Con este método, la parte del número de página en una dirección virtual se contrasta en una tabla de dispersión por medio de una función de dispersión simple'. La tabla de dispersión contiene un puntero a la tabla de páginas invertida, que contiene a su vez las entradas de la tabla de páginas. Con esta estructura, hay una entrada en la tabla de dispersión y la tabla de páginas invertida por cada página de memoria real en lugar de una por cada página virtual. Así pues, se necesita una parte fija de la memoria real para las tablas, sin reparar en el número de procesos o de páginas virtuales soportados. Puesto que más de una dirección de memoria pueden corresponderse con la misma entrada de la tabla de dispersión, para gestionar las colisiones se emplea una técnica de encadenamiento. La técnica de dispersión genera normalmente cadenas cortas, de dos a tres entradas cada una.

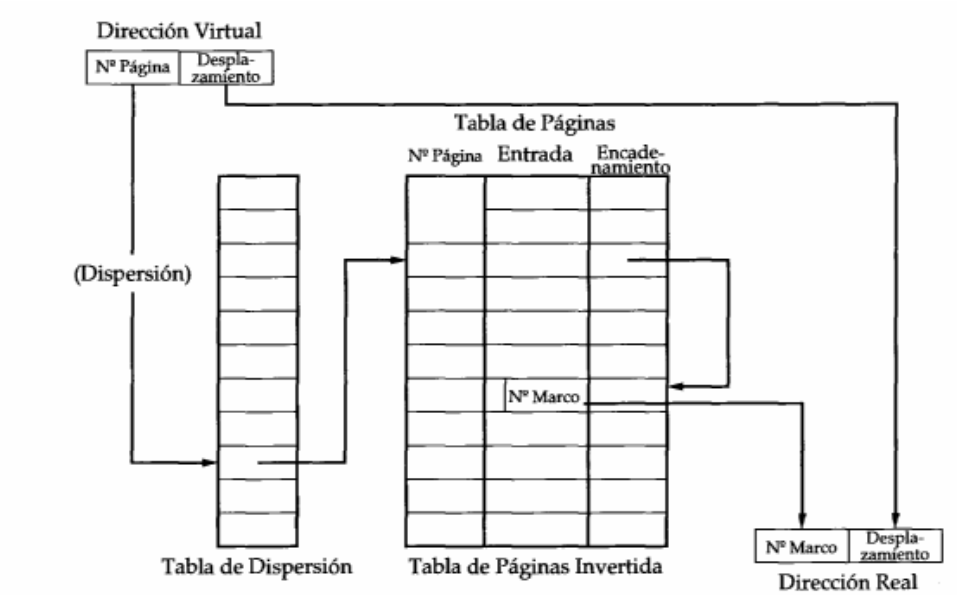


FIGURA 7.4 Estructura de la tabla de páginas invertida

17. Suponga que la tabla de páginas para un proceso que se está ejecutando es la que se muestra a continuación

Página	Bit V	Bit R	Bit M	Marco
0	1	1	0	4
1	1	1	1	7
2	0	0	0	-
3	1	0	0	2
4	0	0	0	-
5	1	0	1	0

Asumiendo que:

- **El tamaño de la página es de 512 bytes**
- **Cada dirección de memoria referencia 1 byte**
- **Los marcos se encuentran contiguos y en orden en memoria a partir de la dirección real 0.**

¿Qué dirección física, si existe, correspondería a cada una de las siguientes direcciones virtuales?

a. 1052 : 1052 div 512 : 2 -> Página 2 : Fallo de página
b. 2221 : 2221 div 512 : 4 -> Página 4 : Fallo de página
c. 5499 : 5499 div 512 : 10 -> Página Errónea
d. 3101 : 3101 div 512 : 6 -> Página Errónea

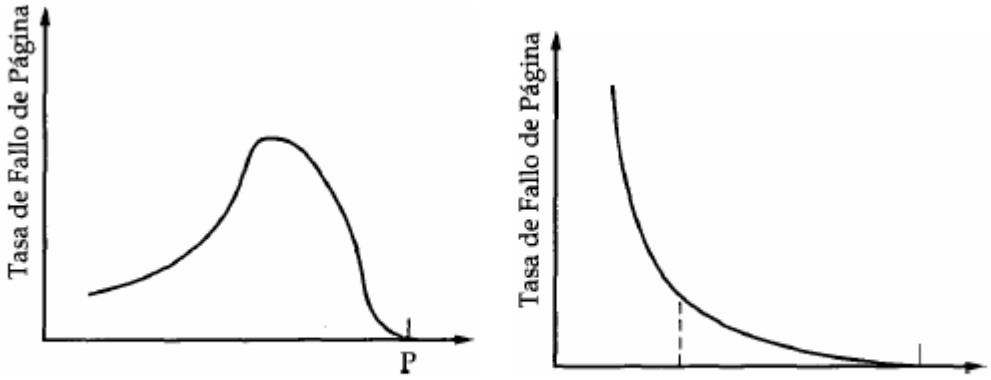
18. Tamaño de página

La selección del tamaño de la página influye de manera directa sobre el funcionamiento de la memoria virtual. Compare las siguientes situaciones con respecto al tamaño de página, indicando ventajas y desventajas Tamaño de página pequeño y Tamaño de página grande

Una decisión importante de diseño del hardware es el tamaño de página que se va a usar. Hay varios factores que considerar. Uno es la fragmentación interna. Sin duda, cuanto menor sea el tamaño de página, menor será la cantidad de fragmentación interna. Para optimizar el uso de la memoria principal, es positivo reducir la fragmentación interna. Por otro lado, cuanto menor sea la página, mayor será el número de páginas que se necesitan por proceso. Un número mayor de páginas por proceso significa que las tablas de páginas serán mayores. Para programas grandes, en un entorno multiprogramado intensivo, esto puede significar que una gran parte de las tablas de páginas de los procesos activos deban estar en memoria virtual, no en memoria principal. Así pues, pueden suceder dos fallos de página para una única referencia a memoria: primero, para traer la parte necesaria de la tabla de páginas y, segundo, para traer la página del

proceso. Otro factor radica en que las características físicas de la mayoría de los dispositivos de memoria secundaria, que son de rotación, son propicias a tamaños de página mayores, puesto que así se obtiene una transferencia por bloques de datos que es más eficiente.

Para complicar la cuestión se tiene el efecto que tiene el tamaño de página en el porcentaje de fallos de página. Este comportamiento se representa, en líneas generales, en la figura y se basa en el principio de cercanía.



(a) Tamaño de pagina W N (b) Número de marcos de página asignados

Si el tamaño de página es muy pequeño, normalmente estarán disponibles en memoria principal un gran número de páginas para cada proceso. Después de un tiempo, todas las páginas en memoria contendrán parte de las referencias más recientes del proceso. Así pues, la tasa de fallos de página será menor. Cuando se incrementa el tamaño de la página, cada página individual contendrá posiciones cada vez más distantes de cualquier referencia reciente. Así pues, se atenúa el efecto del principio de cercanía y comienza a aumentar la tasa de fallos de página. Sin embargo, la tasa de fallos de página comenzará a bajar cuando, finalmente, el tamaño de página se aproxime al tamaño de todo el proceso (punto P del diagrama). Cuando una única página abarca todo el proceso, no hay fallos de página.

Una dificultad más es que la tasa de fallos de página viene determinada también por el número de marcos asignados a un proceso. La figura 7.9b demuestra que, para un tamaño de página fijo, la tasa de fallos baja conforme sube el número de páginas contenidas en memoria principal. Así pues, una política del software (la cantidad de memoria asignada a cada proceso) afectará a una decisión de diseño del hardware.

19. Asignación de marcos a un proceso (Conjunto de trabajo o Working Set):

Con la memoria virtual paginada, no se requiere que todas las páginas de un proceso se encuentren en memoria. El SO debe controlar cuántas páginas de un proceso puede tener en la memoria principal. Existen dos políticas que se pueden utilizar:

- **Asignación Fija**
- **Asignación Dinámica**

a. Describa cómo trabajan estas dos políticas

Asignación Fija

La política de asignación fija otorga a cada proceso un número fijo de páginas en las que ejecutar. Dicho número se decide en el instante de carga inicial (instante de creación del proceso) y puede estar determinado por el tipo de proceso (interactivo, por lotes, tipo de aplicación) o en función de las directrices del programador o del administrador del sistema. Con una política de asignación fija, cada vez que se produce un fallo de página en la ejecución de un proceso, se debe reemplazar una de las páginas de dicho proceso por la página que se necesite.

Asignación Dinámica

La política de asignación variable permite que el número de marcos asignados a un proceso cambie a lo largo de su vida. En el mejor de los casos, un proceso que está sufriendo de forma permanente un alto

número de fallos de página, demostrando que el principio de cercanía apenas se cumple para él, recibirá marcos adicionales para reducir el porcentaje de fallos de página, mientras que un proceso con una tasa de fallos excepcionalmente baja, que demuestra que el proceso se comporta bastante bien desde el punto de vista de la cercanía, verá reducida su asignación, con la esperanza de que esto no aumentará demasiado su tasa de fallos. El uso de una política de asignación variable está relacionado con el concepto de alcance del reemplazo, como se explica más adelante.

La política de asignación variable parece ser la más potente. Sin embargo, la dificultad de este método está en que requiere que el sistema operativo evalúe el comportamiento de los procesos activos. Esto produce inevitablemente la necesidad de más software en el sistema operativo y es dependiente de los mecanismos de hardware ofrecidos por la plataforma del procesador.

b. Dada la siguiente tabla de procesos y las páginas que ellos ocupan, y teniéndose 40 marcos en la memoria principal, cuántos marcos le corresponderían a cada proceso si se usa la técnica de Asignación Fija

Proceso	Total de Paginas Usadas
1	15
2	20
3	20
4	8

i**. Reparto Equitativo**

Se reparten entre los 4 procesos la cantidad total de marcos, por lo tanto cada proceso recibirá 10 marcos.

ii. Reparto Proporcional

Fórmula= total de páginas usadas x proceso / suma de páginas de todos los procesos * cantidad de marcos.

Proceso 1= 15/63 * 40= 9
Proceso 2=20/63 *40= 13
Proceso 3=20/63 *40 =13
Proceso 4=8/63 *40= 5

c. ¿Cuál de los dos repartos usados en b. resulto más eficiente? ¿Por qué?

La política de asignación variable parece ser la más potente, ya que le asigna a los procesos el espacio adecuado según las necesidades. Aunque aumenta el trabajo del Sistema Operativo.

20. Reemplazo de páginas (selección de una víctima):

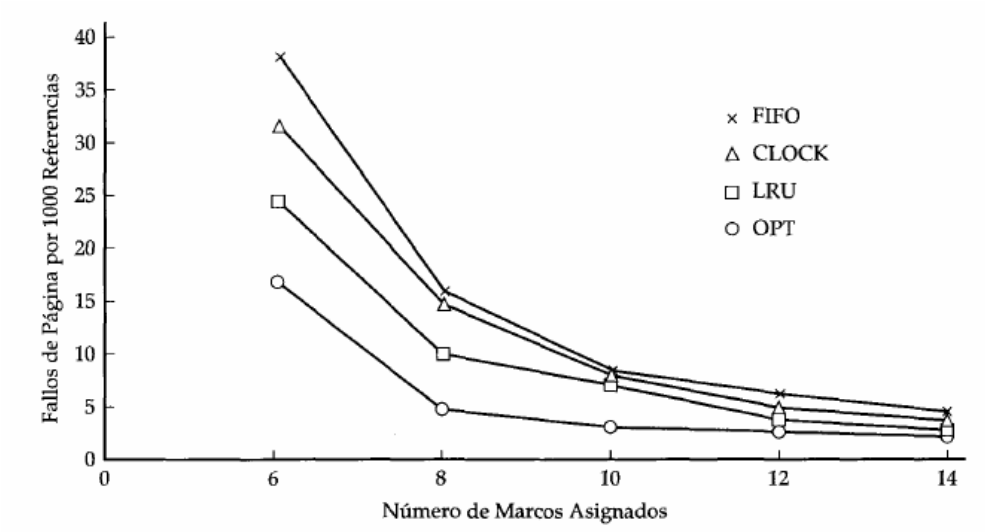
¿Qué sucede cuando todos los marcos en la memoria principal están usados por las páginas de los procesos y se produce un fallo de página? El SO debe seleccionar una de las páginas que se encuentra en memoria como víctima, y reemplazarla por la nueva página que produjo el fallo.

Considere los siguientes algoritmos de selección de victimas básicos:

1. **LRU**
2. **FIFO**
3. **OPT**
4. **Segunda Chance**

a. Clasifique estos algoritmos de malo a bueno de acuerdo a la taza de fallos de página que se obtienen al utilizarlos

OPT->LRU-> Segunda Chance ->FIFO



b. Analice su funcionamiento. ¿Cómo los implementaría?

La política **óptima** selecciona para reemplazar la página que tiene que esperar una mayor cantidad tiempo hasta que se produzca la referencia siguiente. Se puede demostrar que esta política genera el menor número de fallos de página [BELA66]. Sin duda, este algoritmo resulta imposible de implementar, puesto que requiere que el sistema operativo tenga un conocimiento exacto de los sucesos futuros. Sin embargo, sirve como un estándar con el que comparar los otros algoritmos.

La política de la **usada hace más tiempo** (LRU, Least Recently U sed) reemplaza la página de memoria que no ha sido referenciada desde hace más tiempo. Debido al principio de cercanía, ésta debería ser la página con menor probabilidad de ser referenciada en un futuro cercano. De hecho, la política LRU afina casi tanto como la política óptima. El problema de este método es su dificultad de implementación. Una solución sería etiquetar cada página con el instante de su última referencia; esto tendría que hacerse para cada referencia a memoria, tanto para instrucciones como datos. Incluso si el hardware respaldara este esquema, la sobrecarga resultaría tremenda. Como alternativa, se podría mantener una pila de referencias a página, aunque también con un coste elevado.

La política de **primera en entrar, primera en** salir (FIFO) trata los marcos asignados a un proceso como un buffer circular y las páginas se suprimen de memoria según la técnica de espera circular (round-robiri). Todo lo que se necesita es un puntero que circule a través de los marcos del proceso. Esta es, por tanto, una de las políticas de reemplazo más sencillas de im-plementar. La lógica que hay detrás de esta elección, además de su sencillez, es reemplazar la página que ha estado más tiempo en memoria: Una página introducida en memoria hace mucho tiempo puede haber caído en desuso. Este razonamiento será a menudo incorrecto, porque habrá regiones de programa o de datos que son muy usadas a lo largo de la vida de un programa. Con el algoritmo FIFO, estas páginas se cargarán y expulsarán repetidas veces.

En La política de **Segunda Chance** se utiliza un Bit adicional (Bit de referencia). Cuando la página se carga en memoria, el bit R se pone en 0. Cuando la página es referenciada el bit R se coloca en 1. La víctima 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.

c. Sabemos que la página a ser reemplazada puede estar modificada. ¿Qué acciones debe llevar el SO cuando se encuentra ante esta situación?

- 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 marco de descarga asincrónica.
    - El SO envía la orden de descargar asincrónicamente la página modificada mientras continua la ejecución de otro proceso
    - El marco de descarga asincrónica pasa a ser el que contenía a la página victima que ya se descargó correctamente.

21. Alcance del reemplazo

Al momento de tener que seleccionar una página víctima, el SO puede optar pod dos políticas a utilizar : Reemplazo local, o Reemplazo global.

a. Describa como trabajan estas dos políticas

El alcance de un reemplazo puede clasificarse en global o local. Ambos tipos de políticas son activadas por un fallo de página que se produce cuando no hay marcos libres. Para seleccionar la página a reemplazar, una política de **reemplazo local** escoge únicamente de entre las páginas residentes del proceso que originó el fallo de página. Una política de **reemplazo global** considera todas las páginas en memoria como candidatas para reemplazar, independientemente del proceso al que pertenezcan. Aunque las políticas locales son más fáciles de analizar, no hay ninguna evidencia de que se comporten mejor que las políticas globales, las cuales son atrayentes por su simplicidad de implementación y su mínimo coste.

Existe una relación entre el alcance del reemplazo y el tamaño del conjunto residente.

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

No. Un conjunto residente fijo implica una política de reemplazo local: Para mantener constante el tamaño del conjunto residente, una página suprimida de memoria principal debe reemplazarse por otra del mismo proceso. Una política de asignación variable puede, sin duda, emplear una política de reemplazo global: El reemplazo de una página de un proceso en memoria principal por otra provoca que la asignación de un proceso crezca en una página y la de otro disminuya también en una.

22, 23, 24 : En Cuaderno.

25. Hiperpaginación (Trashing)

a. ¿Qué es?

Cuando el grado de multi- programación supera un pequeño valor, se podría esperar que la utilización del procesador aumentara, puesto que hay menos posibilidades de que todos los procesos estén bloqueados. Sin embargo, se alcanza un punto en el que el conjunto residente en promedio no es ade- cuado. En este punto, el número de fallos de página se eleva drásticamente y la utilización del procesador se desploma.

Hay varias formas de abordar este problema. Los algoritmos del conjunto de trabajo o de frecuencia de fallos de página incorporan implícitamente el control de carga. Sólo pueden ejecutar aquellos procesos cuyos conjuntos residentes sean suficientemente grandes. Dado un tamaño exigido para el conjunto residente de cada proceso activo, la política determina automática y dinámicamente el número de programas activos.

b. ¿Cuáles pueden ser sus motivos?

c. ¿Cómo la detecta el SO?

El sistema puede detectar Hiperpaginación evaluando el nivel de utilización de la CPU en comparación con el nivel de multiprogramación.

26.- Considere un sistema cuya memoria principal se administra mediante la técnica de paginación por demanda que utiliza un dispositivo de paginación, algoritmo de reemplazo global LRU y una política de asignación que reparte marcos equitativamente entre los procesos. El nivel de multiprogramación es actualmente, de 4. Ante las siguientes mediciones:

a. **Uso de CPU del 13%, uso del dispositivo de paginación del 97%.**
b. **Uso de CPU del 87%, uso del dispositivo de paginación del 3%.**
c. **Uso de CPU del 13%, uso del dispositivo de paginación del 3%.**

Analizar:

a. ¿Qué sucede en cada caso?

b. ¿Puede incrementarse el nivel de multiprogramación para aumentar el uso de la CPU? ¿La paginación está siendo útil para mejorar el rendimiento del sistema?

27.- Considere un sistema cuya memoria principal se administra mediante la técnica de paginación por demanda. Considere las siguientes medidas de utilización:

- **Utilización del procesador: 20%**
- **Utilización del dispositivo de paginación: 97,7%**
- **Utilización de otros dispositivos de E/S: 5%**

¿Cuáles de las siguientes acciones pueden mejorar la utilización del procesador?

a) **Instalar un procesador más rápido**
b) **Instalar un dispositivo de paginación mayor**
c) **Incrementar el grado de multiprogramación**
d) **Instalar mas memoria principal**
e) **Decrementar el quantum para cada proceso**

d. El sistema está sufriendo de **hiperpaginación**. Incrementar el grado de multiprogramación sólo agregaría más procesos compitiendo por marcos. Instalar un procesador más rápido no sirve de nada, el actual la está poco utilizado. No hay indicación de que el dispositivo de paginación sea inadecuedo (uno más rápido podría ayudar)

28.- La siguiente formula describe el tiempo de acceso efectivo a la memoria al utilizar paginación para la implementación de la memoria virtual:

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

Donde:

TAE = tiempo de acceso efectivo
p = taza de fallo de pagina (0 <= p <=1)
Am = tiempo de acceso a la memoria real
Tf = tiempo se atención de una fallo de pagina
At = tiempo de acceso a la tabla de páginas. Es igual al tiempo de acceso a la memoria (Am) si la entrada de la tabla de páginas no se encuentra en la TLB.

Suponga que tenemos una memoria virtual paginada, con tabla de paginas de 1 nivel, y donde la tabla de páginas se encuentra completamente en la memoria.
Servir una falla de página tarda 300 nanosegundos si hay disponible un marco vacío o si la página reemplazada no se ha modificado, y 500 nanosegundos si se ha modificado. El tiempo de acceso a memoria es de 20 nanosegundos y el de acceso a la TLB es de 1 nanosegundo

a. **Si suponemos una taza de fallos de página de 0,3 y que siempre contamos con un marco libre para atender el fallo ¿Cual será el TAE si el 50% de las veces la entrada de la tabla de páginas se encuentra en la TLB (hit)?**
b. **Si suponemos una taza de fallos de página de 0,3; que el 70% de las ocasiones la pagina a reemplazar se encuentra modificada. ¿Cual será el TAE si el 60% de las veces la entrada de la tabla de páginas se encuentra en la TLB (hit)?**
c. **Si suponemos que el 60% de las veces la pagina a reemplazar esta modificada, el 100% de las veces la entrada de la tabla de páginas requerida se encuentra en la TLB (hit) y se espera un TAE menor a 200 nanosegundos. ¿Cuál es la máxima tasa aceptable de fallas de página?**

29. Anomalía de Belady

a. ¿Qué es?

La **anomalía de Belady** es un efecto descubierto y demostrado en 1969 por el científico de la computación húngaro Laszlo Belady, por el cual es posible tener más fallos de página al aumentar el número de marcos en la memoria física utilizando el método FIFO como algoritmo de reemplazo de páginas en sistemas de gestión de memoria virtual con paginación. Antes de esta fecha, se creía que incrementar el número de marcos físicos siempre llevaría a un descenso del número de fallos de página o, en el peor de los casos, a mantenerlos. Así, pues, antes del descubrimiento de la anomalía de Belady, el algoritmo FIFO era aceptable.

b. Dada la siguiente secuencia de referencias a páginas. 3,2,1,0,3,2,4,3,2,1,0,4

i. Calcule la cantidad de fallos de página si se cuentan con tres marcos y se utiliza el algoritmo de reemplazo FIFO

	3	2	1	0	3	2	4	3	2	1	0	4
1	3	3	3	0	0	0	4	4	4	4	4	4
2		2	2	2	3	3	3	3	3	1	1	1
3			1	1	1	2	2	2	2	2	0	0
PF	X	X	X	X	X	X	X			X	X	

Fallos de Página: 9

ii. Calcule la cantidad de fallos de página si se cuentan con cuatro marcos y se utiliza el algoritmo de reemplazo FIFO

	3	2	1	0	3	2	4	3	2	1	0	4
1	3	3	3	3	3	3	4	4	4	4	0	0
2		2	2	2	2	2	2	3	3	3	3	4
3			1	1	1	1	1	1	2	2	2	2
4				0	0	0	0	0	0	1	1	1
PF	X	X	X				X	X	X	X	X	X

Fallos de página: 9

Analice la situación

Claramente, Belady tenía razón, y el resto se la come.

30.- Considere el siguiente programa:

```
#define Size 64
int A[Size; Size], B[Size; Size], C[Size; Size];
int register i, j;
for (j = 0; j < Size; j ++)
      for (i = 0; i < Size; i++)
            C[i; j] = A[i; j] + B[i; j];
```

Si asumimos que el programa se ejecuta en un sistema que utiliza paginación por demanda para administrar la memoria, donde cada página es de 1Kb. Cada número entero (int) ocupa 4 bytes. Es claro que cada matriz requiere de 16 páginas para almacenarse. Por ejemplo: A[0,0]..A[0,63], A[1,0]..A[1,63], A[2,0]..A[2,63] y A[3,0]..A[3,63] se almacenara en la primer pagina.
Asumamos que el sistema utiliza un working set de 4 marcos para este proceso. Uno de los 4 marcos es utilizado por el programa y los otros 3 se utilizan para datos (las matrices). También asumamos que para los índices "i" y "j" se utilizan 2 registros, por lo que no es necesario el acceso a la memoria para estas 2 variables.

a. Analizar cuantos fallos de páginas ocurren al ejecutar el programa (considere las veces que se ejecuta C[i,j] = A[i,j] + B[i,j])

3 fallos de página por cada cuatro ejecuciones de C[i, j] = A[i, j] +B[i, j].

b. ¿Puede ser modificado el programa para minimizar el número de fallos de páginas. En caso de ser posible indicar la cantidad de fallos de fallos de páginas que ocurren.

Sí, los fallos de página pueden ser reducidos intercambiando los loops, para que i se incremente más lentamente que j. En este caso, habrá 3 fallos de página por cada 256 ejecuciones.