Práctica 5

1. Explique a que hacen referencia los siguientes términos:

Dirección Lógica o Virtual

Es una referencia a un espacio de memoria en función del tamaño del área código (texto, datos y pila) de un *programa* (i.e. no es la de un proceso). Es la información con la que se codifican las instrucciones en lenguaje de máquina al momento de compilarse, es decir que son totalmente independientes de las características físicas del sistema. En términos de código, una dirección virtual indica un desplazamiento entre 0 y n-1, donde n es la cantidad de espacios de memoria que ocupa la totalidad de código del programa.

Dirección Física

Es la indicación de una posición concreta en memoria física, en los términos en que el hardware es capaz de comprenderla y actuar sobre esta información para recuperar o escribir en posiciones de memoria. En términos de máquina, es una posición entre 0 y n-1, donde n es la cantidad de espacios de memoria principal disponibles en la totalidad del sistema.

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
- Particiones Dinámicas

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

Particiones fijas: en la técnica de particiones fijas, la memoria se divide lógicamente en sectores que pueden ser de diferente o igual tamaño. Estos sectores serán siempre iguales en el tiempo que esté funcionando la máquina, es decir que no pueden modificarse salvo que se reinicie el sistema. Para sistemas de particiones iguales, la asignación de particiones a procesos entrantes resulta trivial, puesto que se trata simplemente de asignarle la primera partición disponible.
En el caso de particiones de diferentes tamaños, por cada que requiere ser atendido, el sistema operativo le debe tomar una decisión respecto a qué partición asignarle, para lo que puede utilizar diferentes algoritmos.

Ventajas: fácil implementación, baja sobrecarga del CPU en el manejo de memoria y resolución de direcciones

Desventajas: es muy probable que se genere un alto desperdicio de memoria en forma de fragmentación interna, puesto que no se puede saber con certeza el tamaño que ocuparán los procesos antes de que éstos sean ejecutados. Sumado a esto, la cantidad de particiones disponibles impone un límite a la cantidad de procesos que pueden encontrarse en memoria. Por último, la imposibilidad de preveer el tamaño de un proceso también es aplicable a su crecimiento o decrecimiento al momento de ser ejecutado, el segundo caso no trae más problemas que el aumento de la fragmentación interna, pero el primero puede llevar a que la partición asignada no basta ya para almacenar al proceso, lo cual requeriría un bloque del proceso para ser desplazado a otra área de memoria en la que quepa, o su suspensión para ser llevado al área de intercambio.

Particiones dinámicas: a cada proceso entrante se le asigna un espacio en memoria igual al requerido al momento de cargarse (o tal vez un poco más, considerando la opción de que aumente su tamaño requerido). Cuando un proceso libera la memoria, el espacio que ocupaba vuelve a considerarse libre y

es susceptible de ser reasignado, o una fracción del mismo, o la combinación de una de estas opciones en conjunción con un espacio libre adyacente, a un proceso entrante. De la misma forma que en el caso de las particiones fijas de tamaño variable, pueden aplicarse diversas metodologías para decidir, de entre los espacios libres (espacios de tamaño variable entre cada proceso), cuál utilizar para colocar un proceso entrante.
Independientemente del método, siempre que se asigna un espacio, quedará una nueva partición libre de tamaño igual al tamaño del espacio (continuo) disponible originalmente menos el espacio asignado al proceso entrante. Cada uno de esto métodos afecta a la dimensión de estos espacios resultantes de diferente forma:

Primer ajuste: de todos los espacios disponibles, se toma el primero en el que quepa el proceso entrante. *Generará una tendencia a ocupar siempre los primeros espacios de la memoria.*

Siguiente ajuste: se mantiene un puntero en la lista de espacios disponibles, cuando se asigna espacio a un determinado proceso, el puntero queda apuntando al siguiente inmediato, y en la próxima búsqueda se comenzará desde ese punto. *Evita la preferencia por una región específica de la memoria.*

Los dos algoritmos anteriores tienen la dificultad de generar gran cantidad de fragmentación externa, puesto que es muy poco probable que el espacio disponible sea exactamente del mismo tamaño que el requerido, quedando siempre un pequeño espacio sin utilizar.

Mejor ajuste: se recorre la totalidad de espacios disponibles, buscando aquel que más se aproxime al tamaño del proceso entrante (siendo siempre el proceso de igual tamaño o menor que el espacio). *Resuelve a nivel de cada proceso el problema de la fragmentación externa (la reduce, no la elimina), pero implica una búsqueda por todos los espacios libres, que ocupa más tiempo de procesador y lecturas de memoria. Esta complejización puede resolverse en cierta medida utilizando una lista ordenada de espacios disponibles; de esta forma, el primer espacio disponible en el que entre el proceso será siempre el de mejor ajuste, con esta técnica, la búsqueda de espacio tiene la misma eficiencia que el primer ajuste. Con todo, no termina de resolver el problema de la fragmentación externa ocasionado por la sucesiva entrada y salida de procesos a memoria.*

Peor ajuste: intenta resolver el problema de la fragmentación externa por exceso, procurando que las particiones libres resultantes de la asignación de espacio sean suficientemente grandes como para albergar a otros procesos. Pretende lograr esto asignando a cualquier proceso entrante la partición más grande disponible. *En la práctica, no resulta un método que incorpore ventajas significativas sobre la técnica de mejor ajuste.*

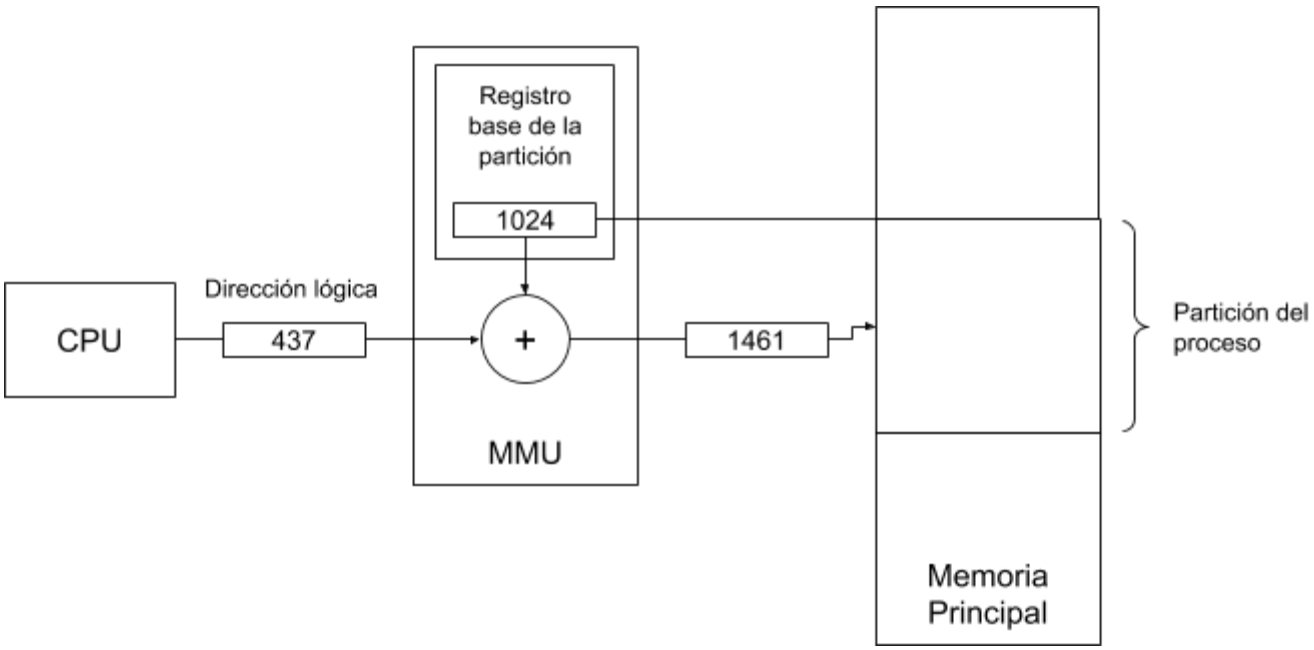
Ventajas: la asignación de memoria es más eficiente por proceso (no hay fragmentación interna) y en general. Aunque es una técnica un poco más difícil de mantener que la asignación fija, su costo continúa siendo relativamente bajo. El grado de multiprogramación puede crecer o decrecer de forma dinámica (aunque hasta cierto límite)

Desventajas: las sucesivas cargas y descargas de procesos en y desde memoria principal pueden generar espacios de memoria libres entre procesos que sean cada uno demasiado pequeño para ser ocupado por un proceso, pero en conjunto suficientemente grandes como para alojar uno o más procesos adicionales (fragmentación externa). Si bien esto es susceptible de resolverse a través de lo que se denomina desfragmentación, se trata de una técnica muy costosa de aplicar en comparación con los beneficios que pueda traer. Por otro lado, de la misma forma que en el método de particiones fijas, resulta difícil conocer con anticipación el espacio que un proceso dado requerirá a lo largo de su historia de ejecución, por lo que pueden surgir problemas en los que un proceso no puede seguir creciendo por

encontrarse con otro (cada proceso es asignado a un espacio continuo de memoria).

(b) ¿Qué información debe disponer el SO para poder administrar la memoria con estos métodos?
Por cada proceso debe contarse con la posición en memoria de inicio de su partición, así como la dimensión de la partición asignada.
Por otro lado, debe existir alguna estructura de datos adicional que audite el estado de espacios libres de la memoria, de forma tal de facilitar los procesos de asignación de particiones a procesos entrantes.

(c) Realice un gráfico 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.
- Particiones de diferente tamaño.

Cite ventajas y desventajas de estos 2 métodos.

Particiones de igual tamaño: son muy fáciles de implementar, puesto que ni siquiera implican la necesidad de un algoritmo para decidir qué partición asignar a un proceso determinado. Sin embargo, generan desperdicio de memoria y, para peor, implican un límite máximo al tamaño de un proceso, aún cuando pueda haber mucha memoria no utilizada en forma de fragmentación interna.

Particiones de diferente tamaño: hacen un mejor uso del espacio, permitiendo que cada proceso ocupe un espacio en memoria de tamaño más acorde a sus dimensiones, reservando (en teoría) las particiones grandes para los procesos que realmente requerirán ese espacio. Según cómo se administre y las características de los procesos, puede tratarse de una técnica más eficiente, pero igual puede generar mucha fragmentación interna (por ejemplo si todos los procesos son de tamaño menor a la partición más pequeña). Implica por otro lado, la necesidad de administrar de alguna forma la decisión de qué proceso asignar a cada espacio, lo cual incrementa la carga sobre el procesador para resolver la administración

de memoria en lugar de atender procesos.

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
Fragmentación interna: es una situación en la que el espacio asignado a un determinado conjunto de datos (entendiendo datos en el sentido genérico que engloba tanto al texto como las variables y pila de un proceso) es mayor al requerido por dicho conjunto, de forma tal que una fracción de estas direcciones reservadas quedan en efecto desperdiciadas, puesto que no pueden ser aprovechadas por ningún otro conjunto de datos que las requieran.

Fragmentación externa: es el caso en el que, debido a la distribución de los espacios reservados de memoria, entre muchos o todos ellos quedan pequeños grupos de direcciones contiguas, que individualmente no son suficientes para alojar ningún nuevo proceso, pero en conjunto pueden conformar un espacio considerable. Este espacio queda efectivamente desperdiciado no por ser inaccesible, si no por ser imposible de utilizar debido a su tamaño, que no puede ampliarse por estar rodeado por espacios ya reservados.

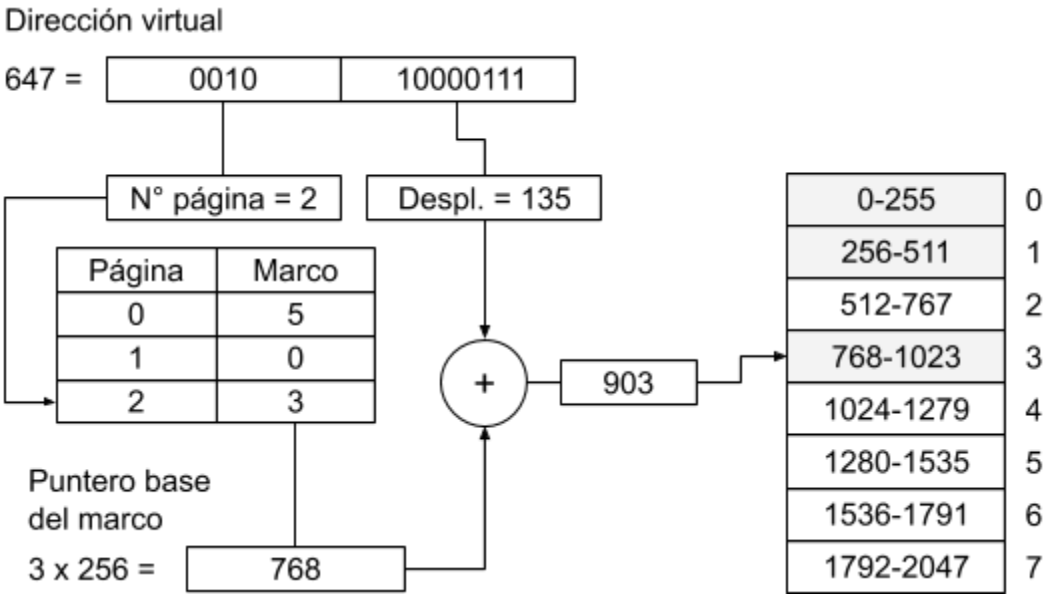
b) El problema de la Fragmentación Externa es posible de subsanar. Explique una técnica que evite este problema.
Se puede resolver haciendo uso de la desfragmentación, que consiste en mover todos los bloques reservados de forma tal que no queden espacios libres entre ellos, generando así un sólo grupo de direcciones libres. Si bien es teóricamente susceptible de ser implementada, los costos en performance la convierten en un método que acarrea más desventajas que aquellas que resuelve.

5. Paginación

a) Explique cómo trabaja este método de asignación de memoria.
La memoria se encuentra físicamente subdividida en marcos, pequeños conjuntos de direcciones contiguas de tamaño fijo. A cada proceso entrante se lo subdivide en tantas secciones como marcos sean necesarios para éste, las cuales son denominadas páginas, y cada página es entonces distribuida entre los marcos disponibles, indistintamente del orden que estas hubieran tenido en el proceso, y de la proximidad o no de los marcos.

b) ¿Qué estructuras adicionales debe poseer el SO para llevar a cabo su implementación?
Por cada proceso debe contarse con una tabla de páginas, que es una estructura de datos en la que se guarda la asociación entre cada página y su marco respectivo, así como información relevante a la página en sí.
A su vez, debe contarse con una lista de marcos disponibles, a fin de facilitar los procesos de asignación de memoria a procesos entrantes y la gestión de procesos salientes.

c) Explique, utilizando gráficos, cómo son transformadas las direcciones lógicas en físicas.
Suponiendo un sistema con marcos de 256 bytes (cada dirección de memoria referencia a 1 byte), y 12 bits de bus de direcciones, un proceso cargado de 957 bytes (fragmentación interna en el último marco de 76 bytes). Si se debe resolver la dirección virtual 647:



d) En este esquema: ¿Se puede producir fragmentación (interna y/o externa)?
Si, se puede producir fragmentación interna. Como de todas formas se está utilizando un espacio de tamaño fijo para alojar procesos de tamaño variable, lo más probable es que la última página del proceso no esté ocupada por completo con información del proceso, si no que una parte de ésta será simplemente espacio reservado sin utilizar. De todas formas, como las subdivisiones son mucho más pequeñas, la fragmentación interna generada es mucho menos significativa que en el caso de las particiones fijas.

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

Ambas técnicas dividen la memoria en fracciones de tamaño fijo, con lo que a su vez generan fragmentación interna. Sin embargo, una de las principales diferencias entre paginación y particionamiento fijo, es que el primero subdivide y distribuye los procesos en varias particiones pequeñas, sin importar su orden o proximidad, mientras que el segundo coloca todo un proceso de forma continua en una sóla partición.

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 se 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 páginas) en el que reflejen el estado descrito.

Tabla de páginas	
#Página	#Marco
0	3
1	5
2	2
3	6

Representación del proceso en memoria			
#Marco	Inicio-Fin		FI
	Dirección física	Dirección virtual	
0	0-511	-	-
1	512-1023	-	-
2	1024-1535	1024-1536	0
3	1536-2047	0-511	0
4	2048-2559	-	-
5	2560-3071	512-1023	0
6	3072-3583	1537-1999	48

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

i) 35: DF = 1571
ii) 512: DF = 2560
iii) 2051: Fuera del límite
iv) 0: DF = 1536
v) 1325: DF = 1325
vi) 602: DF = 2650

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

i) 509: Marco no ocupado (0)
ii) 1500: DL = 1500
iii) 0: Marco no ocupado (0)
iv) 3215: DL = 1679
v) 1024: DL = 1024
vi) 2000: DL = 464

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?

Cant. bytes espacio lógico = 8*1024 = 8192 → $2^3*2^{10} = 2^{13}$
Bits requeridos para representar 8192 direcciones lógicas (0-8191) = log2(8192) = **13**

b) ¿Cuántos bits son necesarios para representar una dirección física?

Cant. bytes espacio físico = 32*1024 = 32768 → $2^5*2^{10} = 2^{15}$
Bits requeridos para representar 32768 direcciones físicas (0-32767) = log2(32767) = **15**

9. Segmentación

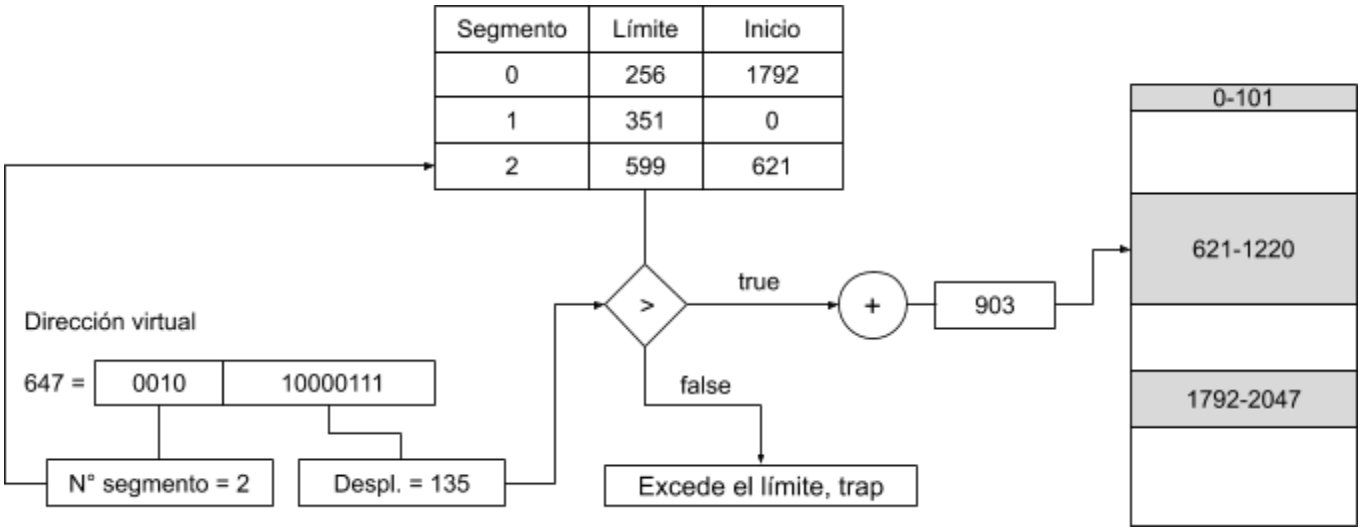
a) Explique cómo trabaja este método de asignación de memoria.

Bajo este esquema, cada proceso se subdivide en porciones, en general con base en su funcionalidad (texto, datos, contratantes, pila, etc.). A cada una de estas porciones, denominada segmento, se le

asigna en memoria un espacio en el que quepa (de manera similar al particionamiento dinámico).
Si bien no aventaja a la paginación en el aspecto de la fragmentación (genera fragmentación externa en mucho menor medida que el método de particiones dinámicas, pero no de forma tan insignificante como la paginación), sí lo hace en el sentido de la facilidad de designar áreas compartidas y protegidas, puesto que mientras en la paginación deben indicarse por lo general una serie de páginas comunes, en la segmentación basta indicar un sólo segmento determinado (el de texto, por ejemplo) para compartir.

b) ¿Qué estructuras adicionales debe poseer el SO para llevar a cabo su implementación?
Se requiere el mantenimiento de una tabla de segmentos, que guarde, por cada uno de estos, un indicador de su puntero base y su tamaño.
A su vez, del mismo modo que para todos los métodos anteriores, se requiere de una estructura con la que se pueda hacer un seguimiento de los espacios libres en memoria.

c) Explique, utilizando gráficos, cómo son transformadas las direcciones lógicas en físicas.
Suponiendo un sistema con marcos de 256 bytes (cada dirección de memoria referencia a 1 byte), y 12 bits de bus de direcciones, un proceso cargado de 957 bytes (fragmentación interna en el último marco de 76 bytes). Si se debe resolver la dirección virtual 647:



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

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

Ambas técnicas se basan en la idea de distribuir espacios de tamaño variable a los procesos en forma dinámica (según la necesidad particular), de esta forma, ambas también generan fragmentación externa, aunque la segmentación en menor medida, y deben contar con algoritmos para seleccionar, entre los espacios libres, aquel al que se asignará un conjunto de datos (segmento o partición según el caso).
La principal diferencia entre ambos es la continuidad, o la falta de ella, de la información de un proceso cargado en memoria; mientras que el particionamiento dinámico requiere que todo el proceso se encuentre cargado de forma continua en memoria, la segmentación lo subdivide y distribuye esas divisiones de forma indiferente a su orden real, aunque sí mantiene la secuencia dentro de cada segmento.

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

Ambas técnicas permiten la distribución de la información de un proceso en sectores no contiguos de memoria, y tampoco requieren que dichos sectores sean asignados en el mismo orden que en el programa principal. A su vez, aunque con sus particularidades, el seguimiento de la ubicación de cada proceso requiere tablas específicas para cada uno de estos que permitan resolver las direcciones lógicas en posiciones físicas en memoria.

Allí se agotan sus similitudes más claras. Si bien por ejemplo ambas técnicas permiten el bloqueo y la compartición de espacios de memoria, la técnica de segmentación es mucho más eficiente que la paginación en este sentido, con base en una de las diferencias claves que consiste en que la segmentación divide al proceso en función de sus utilidades (pila, datos, texto, librerías, etc.), mientras que la paginación lo hace sólo en función del fraccionamiento de la información según el tamaño del marco. A su vez, producto de la forma en que se definen y asignan los espacios libres en una y otra técnicas, la segmentación genera fragmentación externa en niveles relativamente considerables, mientras que la paginación genera fragmentación interna en niveles un poco más despreciables.

12. Dado un S.O. que administra la memoria por medio de segmentación paginada, y teniéndose disponibles las siguientes tablas:

Tabla de segmentos

# Segmento	Dir. base
1	500
2	1500
3	5000

Tabla de páginas

# Segmento	# Página	Dir. 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): DF = 1521
ii) (1,3,15): DF = 575
iii) (3,1,10): DF = 5130
iv) (2,3,5): DF = 1505

13. Memoria Virtual

a) Describa qué beneficios introduce este esquema de administración de la memoria.

Permite tener más procesos en memoria, al mantener sólo aquellos segmentos/páginas (o una combinación de ambas) actualmente requeridos o recientemente utilizados por cada proceso, mientras el resto de la información de estos procesos se encuentra en el área de intercambio. De esta forma, puede aumentar en gran medida el grado de multiprogramación.

A su vez, esta misma distribución de los procesos dentro y fuera de la memoria principal, permite que un proceso cualquiera sea de mayor tamaño que la memoria total, puesto que sólo será cargado en ella en

la medida en que distintas porciones sean requeridas para continuar la ejecución.

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. Aún así, el SO debe dar al hardware la información requerida para llevar a cabo su tarea.

c) Al implementar esta técnica utilizando paginación por demanda, las tablas de páginas de un proceso deben contar con información adicional además del marco donde se encuentra la página. ¿Cuál es esta información? ¿Por qué es necesaria?
Se requiere como mínimo un bit que indique la presencia de la página en memoria, puesto que sólo a partir de ese dato es que el hardware puede generar la interrupción necesaria para resolver el fallo de página. Por otro lado, debe contarse con información sobre la presencia de modificaciones o no sobre las páginas cargadas, puesto que cualquier descarga de una página modificada implica la necesidad de actualizar los datos cargados en disco, para mantener la consistencia.
Otros bits de control pueden resultar útiles para definir el historial de uso del conjunto de páginas o segmentos de un proceso, o de todos los cargados, a fin de poder decidir con mayor grado de precisión cuál porción puede ser la mejor víctima para un reemplazo en caso de presentarse un fallo de página.

14. Fallos de Página (Page Faults):

a) ¿Cuándo se producen?
Cuando una instrucción ejecutada hace referencia a una dirección lógica cuya página no está cargada en memoria.

b) ¿Quién es responsable de detectar un fallo de página?
El hardware. Al momento de resolver la dirección, cuando recupera la entrada en la tabla de páginas correspondiente, debe analizar el bit de control V, que indica si la página se encuentra o no cargada en memoria principal (0 = no cargada). Si el bit V es 0, el hardware genera un trap.

c) Describa las acciones que emprende el SO cuando se produce un fallo de página.
El proceso que genera el fallo de página es puesto en estado bloqueado mientras se resuelve el fallo. En ese tiempo, lo más probable es que el CPU sea asignado a otro proceso a fin de mantenerlo ocupado.
Para resolver el fallo, el SO debe encontrar un marco libre o, en su defecto, seleccionar una página víctima para reemplazarla por la página entrante. Una vez hecho esto, se da inicio a la operación de E/S que carga la página en memoria.
Cuando finaliza la carga del proceso en memoria, el SO es notificado, y en ese momento se actualiza la entrada de la página correspondiente en la tabla, indicando la dirección base del marco, y marcando el bit V = 1.
Luego de esto, el proceso pasa del estado bloqueado a listo para ejecutar (lo hace el planificador de corto plazo). Cuando vuelva a ser asignado al CPU, el proceso comenzará desde la instrucción que generó el fallo en primera instancia (o sea que se deberá resolver nuevamente la dirección, pero esta vez se corresponderá con una página cargada en memoria).

15. Direcciones:

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

i) ¿Cuál es el tamaño máximo de un proceso? (recordar "espacio virtual")

2^{32} direcciones máximas * 1 byte/dirección = 2^{32} bytes máximos

2^{32} bytes / (2^{10} bytes/Kb / 2^{10} Kb/Mb / 2^{10} Mb/Gb) = $2^{32-10-10-10}$ Gb = 2^2 Gb = 4 Gb

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

2^{32} bytes = 2^{32} bytes / 2^{10} bytes/Kb = 2^{22} Kb → tamaño máximo de un proceso en Kb

2^{22} Kb / 2^9 Kb/página = 2^{13} páginas → cantidad máxima de páginas por 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?

2^9 Kb/página = 2^9 Kb/página / 2^{10} Kb/Mb) = 2^{-1} Mb/página = 2^{-1} Mb/marco

256 Mb memoria = 2^8 Mb memoria

2^8 Mb / 2^{-1} Mb/marco = 2^9 marcos = 512 marcos

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

2^{13} entradas/proceso * 2^{10} bytes/entrada = 2^{23} bytes/proceso = 8 Mb/proceso → tamaño máximo de la tabla de páginas de un proceso

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 1 nivel.
Tablas de páginas de 2 niveles.
Tablas de páginas invertidas.

Explique brevemente cómo trabajan estos enfoques e indique cómo se realiza la transformación de la dirección virtual en dirección física.

Las tablas de página de 1 nivel funcionan como el sistema de paginación sin memoria virtual. Todas las páginas se encuentran en memoria, y las direcciones se resuelven asociando una parte de la dirección virtual con un número de marco, siendo el valor restante el desplazamiento dentro de dicho marco. La entrada en la tabla de páginas indica si la página está presente en memoria, y el número de marco es lo que el hardware utiliza para resolver la base de la dirección física (y luego sumarle el desplazamiento)

Las tablas de página de 2 niveles, funcionan de manera similar, pero agregan un nivel de indirección con tablas paginadas. El primer nivel es equivalente a las tablas de página de 1 nivel, aunque cuenta con menos entradas, ya que el segmento de dirección virtual que antes identificaba a una entrada en la tabla de páginas, se encuentra subdividido en dos. La primera parte de la dirección lógica indica una entrada en la tabla de páginas de primer nivel. Dicha entrada refiere a una tabla paginada que, de la misma manera que cualquier página, puede o no estar cargada en memoria. Si la tabla de segundo nivel está

cargada en memoria, la entrada en la tabla de primer nivel tendrá un indicador de la base del marco en el que se encuentra la tabla de segundo nivel. La segunda parte de la dirección lógica se utiliza para identificar la entrada de página efectivamente del proceso que se está buscando, la cual contiene finalmente la dirección base del marco al que la dirección lógica apunta, a la cual se agregan los bits de desplazamiento para concretar la resolución de la dirección física.

Las tablas de páginas invertidas buscan solucionar el problema invirtiendo el proceso de conversión de direcciones (de ahí su nombre). La tabla de páginas invertida se encuentra siempre cargada en memoria, y contiene una entrada por cada marco de la memoria (en lugar de por cada página de cada proceso). El número de página de cada proceso se vincula con un marco a través de una función de hash relativamente sencilla (de otra forma los costos de procesamiento serían demasiado altos). Como varias páginas pueden dar como resultado de la función de hash un mismo valor, si un marco está asociado con más de una página, las entradas se encadenan, de modo que el punto de entrada siempre es la primera entrada en la tabla de páginas. La entrada de página, a su vez, contiene un identificador del proceso al que corresponde, a fin de poder ser reconocida de manera unívoca. Una vez resuelta la asociación entre página y tabla, se puede resolver la dirección física agregando el desplazamiento como en los demás métodos.

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 (0, 1, 2..) a partir de la dirección real 0.
¿Qué dirección física, si existe, correspondería a cada una de las siguientes direcciones virtuales? (No gestione ningún fallo de página, si se produce)
a)1052: fallo de página (página 2, no está en memoria)
b) 2221: fallo de página (página 4, no está en memoria)
c) 5499: fuera de rango del proceso (apunta a una hipotética página 10)
d) 3101: fuera de rango del proceso (apunta a una hipotética página 6)

18. Tamaño de la 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:**

Un tamaño de página pequeño:

Ventajas: se reduce la fragmentación interna. La posibilidad de asignar más páginas en general

podría permitir mayor grado de multiprogramación. En relación con esto, también es posible seleccionar las páginas de los procesos con mayor granularidad, por lo que podría, teóricamente, aprovecharse de forma más eficiente el espacio en memoria principal - por ejemplo, lo que en páginas grandes podría ser un conjunto de trabajo de tres páginas de 2048 bytes (6 KiB en total), en páginas pequeñas podría estar conformado por 7 páginas de 256 bytes (1,8 KiB en total); es decir aquellas secciones de código que efectivamente se mantienen en referencia frecuente.

Desventajas: el menor tamaño de página incrementa las chances de que una dirección virtual apunte a otra página, y con ello aumentan las posibilidades de que suceda un fallo de página, así como las chances de que la dirección buscada no esté en el TLB. Por último, un mayor número de páginas (para una misma memoria física, menor tamaño de páginas implica mayor cantidad de ellas) implica tablas de páginas más grandes, con lo que cada proceso, independientemente de su tamaño, requerirá más espacio reservado sólo para sus tablas de páginas

Un tamaño de página grande:

Ventajas: menor cantidad de páginas por proceso, lo que incrementa las posibilidades de que una dirección virtual apunte a la misma página o a alguna ya cargada en memoria (con menos páginas se puede mantener el mismo conjunto residente). A su vez, las tablas de página ocupan menos espacio. Por último, los costos de transferencia hacia y desde disco se ven reducidos, puesto que la memoria secundaria funciona de forma más eficiente en bloques de mayor tamaño.

Desventajas: incrementa la fragmentación interna. La subdivisión de la memoria en espacios más grandes, disminuye el potencial de multiprogramación. A partir de cierto punto, se convierte en poco más que un particionado fijo con memoria virtual.

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 cuantas páginas de un proceso puede tener en la memoria principal. Existen 2 políticas que se pueden utilizar:

- Asignación Fija
- Asignación Dinámica.

a) Describa cómo trabajan estas 2 políticas.

Asignación fija: al momento de asignarse espacios de memoria física a los procesos, a cada uno se le permite el uso de un conjunto estático de direcciones, es decir que si todo su espacio está ocupado con páginas cargadas, y sucede un fallo de página, la víctima necesariamente surgirá de su propio conjunto de páginas en memoria. De la misma manera, incluso si aún no ha ocupado la totalidad de su espacio asignado (incluso si no tiene suficientes páginas como para llenarlo), ningún otro proceso podrá “quitarle” marcos para cargar sus páginas.

Asignación dinámica: la memoria es considerada como un recurso compartido por el cual compiten los procesos en ejecución. En la medida en que un proceso dado tiene una alta tasa de fallos de página, y otros no tienen ninguna, se le podría asignar mayor número de marcos de página al primer proceso. A su vez, en los momentos en que aumenta o disminuye el grado de multiprogramación, se genera una redistribución de los marcos de página asignados a cada proceso.

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 páginas usadas
1	15

2	20
3	20
4	8

i) Reparto Equitativo
A cada proceso le corresponde ¼ del total de memoria disponible, lo que en este caso equivale a 10 marcos por proceso.

ii) Reparto Proporcional
Páginas totales = 15+ 20 +20 + 8 = 63
Proporción proceso 1 = 15/63 → marcos proceso 1 = |40*15/63| = 9
Proporción proceso 2 = 20/63 → marcos proceso 2 = |40*20/63| = 13
Proporción proceso 3 = 20/63 → marcos proceso 3 = |40*20/63| = 13
Proporción proceso 4 = 8/63 → marcos proceso 3 = |40*8/63| = 5

b) ¿Cuál de los 2 repartos usados en b) resultó más eficiente? ¿Por qué?
El reparto proporcional, en principio porque el reparto equitativo deja dos marcos sin utilizar (al proceso 4 se le asignan 10 mientras que sólo requiere 8). A su vez, si bien las necesidades de ejecución de cada proceso son desconocidas, puede especularse que, bajo el reparto proporcional, las probabilidades de que cada proceso genere fallos de página, son aproximadamente iguales para todos los procesos, mientras que en el caso del reparto equitativo, se ven penalizados los procesos con mayor cantidad de páginas, y muy beneficiados aquellos con menos.

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 en fallo de página? El SO debe seleccionar una de las páginas que se encuentra en memoria como víctima, y ser reemplazada por la nueva página que produjo el fallo.
Considere los siguientes algoritmos de selección de víctimas básicos:
- LRU
- FIFO
- OPT (Optimo)
- Segunda Chance

a) Clasifique estos algoritmos de malo a bueno de acuerdo a la tasa de fallos de página que se obtienen al utilizarlos.
El algoritmo óptimo es el que menor tasa de fallos de página genera por definición, ya que es más una concepción teórica que permite establecer un criterio de comparación entre los demás algoritmos, de lo que es realmente un algoritmo practicable en la mayoría de los casos.
LRU: Entre aquellos más susceptibles de ser implementados, es el que más se aproxima a la idea de trabajo con el conjunto residente, por lo que podría esperarse una tasa de fallos de página relativamente baja respecto a otros métodos.
Segunda chance: Implica una mejora sustancial sobre el método FIFO, aproximándose en mayor medida que éste a la idea de trabajo con el conjunto residente.
FIFO: Es el peor en cuanto a la tasa de fallos de página, puesto que no hace más que eliminar aquellas páginas que llevan más tiempo en memoria, lo que puede significar eliminar aquellas páginas que más uso han tenido.

b) Analice su funcionamiento. ¿Cómo los implementaría?
FIFO: no necesitaría más que una cola FIFO de punteros hacia las entradas en la tabla de páginas.

Dependiendo del tipo de asignación la cola podría representar a la totalidad de páginas en memoria, o a las de cada proceso. En la medida en que se agregan páginas a memoria, se agregan punteros a las mismas al final de la cola, y en la medida en que se requiere seleccionar páginas víctima, se selecciona siempre al primer elemento en la cola.
Segunda chance: podría implementarse también con una cola de punteros, con la sola diferencia de generar una reubicación en el final de la cola, y marcado del bit R en 0 en la entrada correspondiente, cada vez que, al momento de seleccionar una víctima, el elemento seleccionado tenga su bit R en 1.
LRU: para hacerlo realmente fiel a su concepto, debería poder asociar timestamps a cada página, de ser posible, podría tenerse una min-heap de páginas ordenadas según el tiempo del sistema en el que fueron ejecutados. Si bien esto hace relativamente fácil la selección de una página víctima, implica una búsqueda lineal para encontrar una página al momento de requerir actualizar el tiempo de acceso a una página que ya se encontraba en memoria.
OPT: no lo podría implementar en ejecución, pero, si existe un conjunto de procesos cuyos comportamientos exactos conozco, podría generarse un algoritmo que calcule la distribución óptima de páginas. Intuitivamente asumo que un algoritmo de ese tipo sería de un orden relativamente grande, por lo que sólo podría ser conveniente para conjuntos de procesos que de antemano se sabe que se van a ejecutar en un determinado orden durante un largo período de tiempo (en forma cíclica). De esta forma la optimización de distribución de páginas de los procesos podría compensar el tiempo requerido para calcular la misma.

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

En este caso no puede directamente sobreescribirse o descartarse la página, o se perdería información. Inicialmente debería realizarse el pedido de transferencia a disco de la página. Mientras dicho pedido no sea atendido, debería disponerse de un espacio en memoria en el que pueda residir dicha página hasta ser efectivamente escrita en memoria secundaria. Una vez concluido este proceso, el espacio ocupado por la página quedaría nuevamente libre.
En la práctica, por lo general se cuenta con uno o más marcos de página de intercambio que, en el caso de haber una página víctima que requiera ser escrita en memoria secundaria, será ocupado por la página entrante, una vez que la víctima es escrita en disco, el espacio que ocupaba se considera el nuevo espacio de intercambio.

21. Alcance del reemplazo:

Al momento de tener que seleccionar una página victima, el SO puede optar por 2 políticas a utilizar:
- Reemplazo local
- Reemplazo global

a) Describa cómo trabajan estas 2 políticas.

Reemplazo local: de cada cada proceso sólo se puede seleccionar como víctima a una página que pertenezca a su propio conjunto de páginas.
Reemplazo global: la selección de página víctima se realiza sobre el total de páginas en memoria, independientemente de si pertenecen o no al proceso que generó el fallo de página.

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.

22. Considere la siguiente secuencia de referencias de páginas:

1, 2, 15, 4, 6, 2, 1, 5, 6, 10, 4, 6, 7, 9, 1, 6, 12, 11, 12, 2, 3, 1, 8, 1, 13, 14, 15, 3, 8

a) Si se dispone de 5 marcos. ¿Cuántos fallos de página se producirán si se utilizan las siguientes técnicas de selección de víctima? (Considere una política de Asignación Dinámica y Reemplazo Global)

i) Segunda Chance
ii) FIFO
iii) LRU
iv) OPT

b) Suponiendo que cada atención de un fallo se pagina requiere de 0,1 seg. Calcular el tiempo consumido por atención a los fallos de páginas para los algoritmos de a).

23. Sean los procesos A, B y C tales que necesitan para su ejecución las siguientes páginas:

A: 1, 3, 1, 2, 4, 1, 5, 1, 4, 7, 9, 4
B: 2, 4, 6, 2, 4, 1, 8, 3, 1, 8
C: 1, 2, 4, 8, 6, 1, 4, 1

Si la secuencia de ejecución es tal que los procesos se ejecutan en la siguiente secuencia:

1. B demanda 2 páginas
2. A demanda 3 páginas
3. C demanda 2 páginas
4. B demanda 3 páginas
5. A demanda 3 páginas
6. C demanda 2 páginas
7. B demanda 2 páginas
8. C demanda 4 páginas
9. A demanda 3 páginas
10. B demanda 3 páginas
11. C termina
12. A demanda 3 páginas
13. B termina
14. A termina

a) Considerando una política de Asignación Dinámica y Reemplazo Global y disponiéndose de 7 marcos. ¿Cuántos fallos de página se producirán si se utiliza la técnica de selección de víctimas:

i) LRU
ii) Segunda Chance

b) Considerando una política de Asignación Fija con reparto equitativo y Reemplazo Local y disponiéndose de 9 marcos. ¿Cuántos fallos de página se producirán si se utiliza la técnica de selección de víctimas:

i) LRU
ii) Segunda Chance

24. ISean los procesos A, B y C tales que necesitan para su ejecución las siguientes páginas:

A: 1, 2, 1, 7, 2, 7, 3, 2
B: 1, 2, 5, 2, 1, 4, 5
C: 1, 3, 5, 1, 4, 2, 3

Si la secuencia de ejecución es tal que los procesos se ejecutan en la siguiente secuencia:

1. C demanda 1 página
2. A demanda 2 páginas
3. C demanda 1 página

4. B demanda 1 página
5. A demanda 1 página
6. C modifica la página 1
7. B demanda 2 páginas
8. A demanda 1 página
9. C demanda 1 página
10. B modifica la página 2
11. A modifica la página 2
12. B demanda 2 páginas
13. A demanda 1 página
14. B demanda 2 páginas
15. C demanda 2 páginas
16. C demanda 1 página
17. A demanda 1 página
18. B termina
19. A demanda 2 páginas
20. C demanda 1 página
21. A termina
22. C termina

Considerando una política de Asignación Dinámica y Reemplazo Global y disponiéndose de 7 marcos, debiéndose guardar 1 marco para la gestión de descarga asincrónica de páginas modificadas ¿Cuántos fallos de página se producirán si se utiliza la técnica de selección de víctima:

a) Segunda Chance
b) FIFO
c) LRU

25. Hiperpaginación (Trashing)

a) ¿Qué es?

Es un estado del sistema en el que éste se encuentra permanentemente atendiendo fallos de página, sin que los procesos puedan avanzar en su ejecución.

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

En general, sucede cuando el total de marcos disponibles en el sistema es inferior a la suma de los conjuntos de trabajo de cada proceso en ejecución. Ésto es en términos de análisis teórico, en sí, la hiperpaginación sucede cuando los procesos en ejecución no tienen suficientes marcos asignados como para mantener su conjunto de trabajo. Ante esta situación, cuando una referencia a memoria genera un fallo de página, se seleccionaría una víctima que forma parte del conjunto de trabajo (propio o de otro proceso, según la política de asignación y reemplazo), la ausencia de esta víctima genera otro fallo de página, que al ser atendido selecciona a otra página del conjunto de trabajo, y así sucesivamente.

c) ¿Cómo la detecta el SO?

En la práctica, la técnica más utilizada para detectar la hiperpaginación es la del control de frecuencia de fallos de página. Se establecen a priori para el sistema ciertas cotas (máxima y mínima) a partir de las cuales se define el estado de un proceso y, ante la evaluación de la situación general de todos los procesos, el estado del sistema.
La tasa de fallos de página (p) es, de la cantidad de accesos a memoria de un determinado proceso, la proporción que representan los fallos de página. Es decir que $0 < p <= 1$.
Si $p \rightarrow 1$, el proceso no tiene suficientes marcos asignados como para mantener su localidad, y se está en riesgo de hiperpaginación, mientras que si $p \rightarrow 0$, el proceso está muy próximo a mantener su localidad (nunca puede suceder que p=0, durante toda la vida de un proceso ya que se perdería por completo el propósito de la memoria virtual con paginación).
Las cotas máxima y mínima, entonces, son precisamente los indicadores de si $p \rightarrow 1$ o $p \rightarrow 0$ para un determinado proceso, o la totalidad de ellos.

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

Dependiendo del algoritmo de asignación y reemplazo, se puede reducir la cantidad de marcos asignados a un proceso con $p \rightarrow 0$, para reasignarlos a los procesos con $p \rightarrow 1$. En caso de tratarse de un

reemplazo local, o de no haber procesos con p debajo de la cota mínima, se pueden seleccionar uno o más procesos para suspender, a fin de redistribuir sus marcos a los procesos que lo necesiten (es decir, reducir el grado de multiprogramación).
La última técnica es, muy probablemente, la que más garantías tenga de resolver la hiperpaginación, ya que hay un límite a cuántos marcos se pueden quitar a un proceso antes que este también empiece a generar excesivos fallos de página.

26. Paginación por demanda:

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%.

Se está ante una situación de hiperpaginación, los procesos pasan más tiempo esperando la resolución de fallos de página de lo que pueden ocupar el CPU para continuar su ejecución. No es recomendable incrementar el nivel de multiprogramación, puesto que empeoraría la situación.
La paginación no está siendo útil para la mejoría de rendimiento del sistema.

b) Uso de CPU del 87%, uso del dispositivo de paginación del 3%.

Los procesos están en este momento muy próximos a mantener su localidad, pudiendo hacer un buen aprovechamiento de su tiempo de procesador para avanzar con sus tareas. Si la situación persiste podría incrementarse el grado de multiprogramación, aunque dado el uso de CPU, es posible que aumentar el grado de multiprogramación incremente en mayor medida la tasa de fallos de página sin afectar en gran medida al uso del CPU.
La paginación en este caso está siendo útil para la mejoría de rendimiento del sistema.

c) Uso de CPU del 13%, uso del dispositivo de paginación del 3%.

El CPU se encuentra mayormente ocioso y los procesos no están generando muchos fallos de página, lo recomendable en este caso sería incrementar el nivel de multiprogramación, ya que bajo el esquema actual no parece estarse haciendo un buen aprovechamiento de los recursos, y hay un amplio margen todavía para que aumente el uso del dispositivo de paginación sin entrar en hiperpaginación, pero sí obteniendo un mayor uso de la CPU.
La paginación no está siendo útil para la mejoría de rendimiento del sistema.

Analizar:
¿Qué sucede en cada caso?
¿Puede incrementarse el nivel de multiprogramación para aumentar el uso de la CPU?
¿La paginación está siendo útil para mejorar el rendimiento del sistema?

Si las medidas no son estadísticas de uso, si no resultados de mediciones directas en momentos determinados, la última pregunta es difícil de contestar con tan poca información, en general parecería que el uso de CPU (el recurso más relevante) es bastante bajo, por lo que uno podría intuitivamente decir que es poco útil. Sin embargo, no conociendo la naturaleza de las tareas (si se trata de procesos intensivos I/O bound, por ejemplo, la carga sobre el CPU será baja independientemente de la paginación), y con tan pocas mediciones, considero que cualquier inferencia sería apresurada.

27. Paginación por demanda:

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%
Cuales de las siguientes acciones pueden mejorar la utilización del procesador:

a) Instalar un procesador más rápido

No sería útil, ya que el bajo uso del procesador parece estar más asociado a la incapacidad de los procesos de continuar con su ejecución por la generación de fallos de página, con lo que un aumento en la capacidad del procesador de atenderlos no produce un gran cambio.

b) Instalar un dispositivo de paginación mayor

Dado que el uso de E/S es relativamente bajo, el cuello de botella parece ser principalmente el propio dispositivo de paginación, antes que el requerimiento de páginas de disco. Es posible que aumentando la capacidad del dispositivo de paginación, se obtenga un mejor rendimiento.

c) Incrementar el grado de multiprogramación

Esta decisión definitivamente no ayudaría a mejorar la utilización del procesador. Con un uso tan elevado del dispositivo de paginación, es evidente que el grado de multiprogramación actual está muy cerca del límite de procesos que se pueden atender a un mismo tiempo.

d) Instalar mas memoria principal

Esto ayudaría en gran medida a mantener la localidad de los procesos, con lo que se podría reducir en una medida similar la carga sobre el dispositivo de paginación. De esta forma, los procesos a su vez se verían menos afectados por los fallos de página, pudiendo aprovechar en mayor medida la disponibilidad del procesador.

e) Decrementar el quantum para cada proceso

No parece ser una decisión que ayudaría en gran medida a mejorar el aprovechamiento del CPU, aunque bajo ciertas condiciones sí podría suceder. Debe tenerse en cuenta en primera medida que a partir de cierta reducción el tamaño de quantum genera demasiada sobrecarga, bajando el aprovechamiento del procesador. Con esto aclarado, podría estipularse que, con un menor tiempo de uso de CPU por parte de cada proceso, existen menores probabilidades de que cada uno de estos deba ser expulsado del procesador por un fallo de página (puesto que necesariamente ejecutan menor cantidad de instrucciones). De esta forma, sería posible que algunos procesos completen su quantum mientras se atienden los fallos de página de otros, reduciendo la carga sobre el dispositivo de paginación y a su vez incrementando el aprovechamiento del procesador.

28. La siguiente fórmula 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 = tasa de fallo de página (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 páginas 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 tasa de fallos de página de 0,3 y que siempre contamos con un marco libre para atender el fallo ¿Cuál será el TAE si el 50% de las veces la entrada de la tabla de páginas se encuentra en la TLB (hit)?

TAE = (20 ns + 1 ns)/2 + 20 ns * (1-0.3) + 0.3 * (300 ns + 20 ns) = 10.5 ns +14 ns + 96 ns = 120.5 ns

b) Si suponemos una tasa de fallos de página de 0,3; que el 70% de las ocasiones la página a reemplazar se encuentra modificada. ¿Cuál será el TAE si el 60% de las veces la entrada de la tabla de páginas se encuentra en la TLB (hit)?

TAE = (20 ns * 0.4 + 1 ns * 0.6) + 20 ns * (1-0.3) + 0.3 * ((300 ns * 0.3 + 500 ns * 0.7) + 20 ns) =
= 8.6 ns + 14 ns + 138 ns = 160.5 ns

c) Si suponemos que el 60% de las veces la página a reemplazar está 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?

200 ns = 1ns + 20 ns * (1-p) + p * (300 ns * 0.4 + 500 ns * 0.6 + 20 ns)
200 ns - 1ns = 20 ns - p * 20 ns + p * 440 ns
199 ns - 20 ns = p * (440 ns - 20 ns)
179 ns = p * 420 ns
p = 179 ns/420ns ≈ 0.426
La tasa máxima aceptable de fallos de página es aproximadamente 0.426

29. Anomalía de Bélády

a) ¿Qué es?

Dados dos conjuntos de marcos L y S, teniendo L más marcos, tras aplicar un algoritmo de reemplazo de páginas a una secuencia dada de páginas (siendo la misma secuencia para ambos conjuntos), si el conjunto L genera más fallos de página que el conjunto S, entonces la anomalía existe. Es decir, dados dos espacios de almacenamiento de distinto tamaño (ambos con páginas de igual tamaño, pero distinta cantidad de marcos), una determinada secuencia de requisitos de página conduce a una situación en la que el almacenamiento de mayor tamaño genera más fallos de página que el otro.
A su vez, dados dos conjuntos L y S con las mismas características enunciadas anteriormente, un determinado algoritmo de reemplazo de páginas, y una secuencia de páginas requeridas, luego de seguida la secuencia hasta determinado punto, si existe dentro de la secuencia un subconjunto de requisitos de página que genera más fallos de página en el almacenamiento L que en S, y a su vez deja a ambos espacios en el mismo estado que aquel en el que inicia el subconjunto, entonces también existe la anomalía. Este último caso es porque, si para una parte de la secuencia pueden darse mayor cantidad de fallos de página en L que en S, y a su vez esta parte de la secuencia deja a los dos almacenamientos en el mismo estado que antes de iniciar la secuencia, entonces la misma puede ser repetida indefinidamente hasta que se compense la diferencia de fallos de página entre L y S en el resto de la secuencia de requisitos de página.

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 páginas si se cuentan con 3 marcos y se utiliza el algoritmo de reemplazo FIFO
II. Calcule la cantidad de fallos de páginas si se cuentan con 4 marcos y se utiliza el algoritmo de reemplazo FIFO
Analice la situación

Se produce la anomalía de Bélády, ya que se generan 9 fallos de página en el caso de contar con 3 marcos, y 10 fallos cuando se cuenta con 4.

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 almacenará en la primera página.
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 página ocurren al ejecutar el programa (considere las veces que se ejecuta C[i,j] = A[i,j] + B[i,j])
C[i,j] = A[i,j] + B[i,j] → se ejecuta 64 * 64 = 64^2 veces
4 filas de una matriz pueden almacenarse en 1 página, habiendo 3 páginas en total disponibles para las matrices, eso significa que en todo momento cada matriz sólo tiene cargadas sus filas n a n+4, donde n es un múltiplo de 4 entre 0 y 60.
Dado que el cambio en el índice i (indice de fila) sucede en el bucle más interno, cada 4 operaciones (cuando se haga el salto de la fila 3 a la 4, de la 7 a la 8, etc.) se generarán 3 fallos de página (uno por cada matriz).
Por lo tanto la cantidad de fallos de página será:
64 * 64 operaciones / 4 operaciones/salto * 3 fallos/salto = 3072 fallos

b) Puede ser modificado el programa para minimizar el número de fallos de página. En caso de ser posible indicar la cantidad de fallos de fallos de páginas que ocurren.
Si se cambia el orden en que se anidan los bucles (o en el que se llama a los índices en la matriz):

```
#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];
```

La cantidad de operaciones es la misma, pero ahora se operará sobre cada página completa (desde X[n,0] hasta X[n+4, 63], con las misma interpretación de n) antes de cada “salto”. Es decir que cada trío de fallos de página sucederá cada 4*64 (la cantidad de celdas de cada matriz en una sóla página) operaciones.
Por lo tanto la cantidad de fallos de página será:
64*64 operaciones / 4*64 operaciones/salto * 3 fallos/salto = 48 fallos

31. Considere las siguientes secuencias de referencias a páginas de los procesos A y B, donde se muestra en instante de tiempo en el que ocurrió cada una (1 a 78):

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Proceso A	1	2	1	3	4	3	3	2	1	1	3	4	5	6	1	2	1	3	3	4	5	6	6	6	5	4	3	1	1	2	3	4	5	4	3	2	1	1	1
Proceso B	1	2	3	3	3	4	4	5	1	1	1	2	3	4	4	4	4	1	1	2	3	6	5	6	5	4	6	1	1	1	1	4	5	1	3	2	1	1	2
	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78
Proceso A	2	2	1	2	1	1	2	3	4	5	6	7	8	6	5	3	1	2	3	4	5	6	5	1	1	2	3	4	5	4	2	1	1	2	3	4	5	1	1
Proceso B	2	2	3	3	4	4	4	3	4	5	4	6	1	1	1	2	3	3	3	4	5	5	5	1	6	2	3	1	2	3	1	1	1	2	2	2	2	3	3

a) Considerando una ventana Δ=5, indique cuál sería 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, 6, 5}

b) Considerando una ventana Δ=5, indique cuál sería 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 se está en situación de thrashing, ya que la suma de los tamaños de conjunto de trabajo es 7 (es decir menor que la cantidad de frames disponibles), con lo que puede aventurarse que no hay trashing. Con 6 frames disponibles, la situación es la opuesta: no hay suficientes marcos de página disponibles para contener el conjunto de trabajo de todos los procesos, por lo que se entrará en hiperpaginación.

d) Considerando únicamente el proceso A, y suponiendo que al mismo se le asignaron inicialmente 4 marcos, donde el reemplazo de páginas 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?
p = 5

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

El valor alcanza la cota superior de tasa aceptable, por lo que el SO aún podría no hacer nada. Ahora, si la cota implica un valor a partir del cual el SO debe responder, lo que debería hacer es: o asignarle un marco adicional (lo cual como se vio según la anomalía de Bélády puede empeorar la situación) o, en caso no haber marcos disponibles, bloquear el proceso y efectivamente des-asignarle todos los marcos para distribuirlos entre todos los procesos.