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

Dirección Lógica o Virtual

- Es una dirección que enmascara o abstrae una dirección física
- Referencia a una localidad en memoria
- Se la debe traducir a una dirección física

Dirección Física

- Es la dirección real. Es con la que se accede efectivamente a memoria.
- Representa la dirección absoluta en memoria principal.

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 prever 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

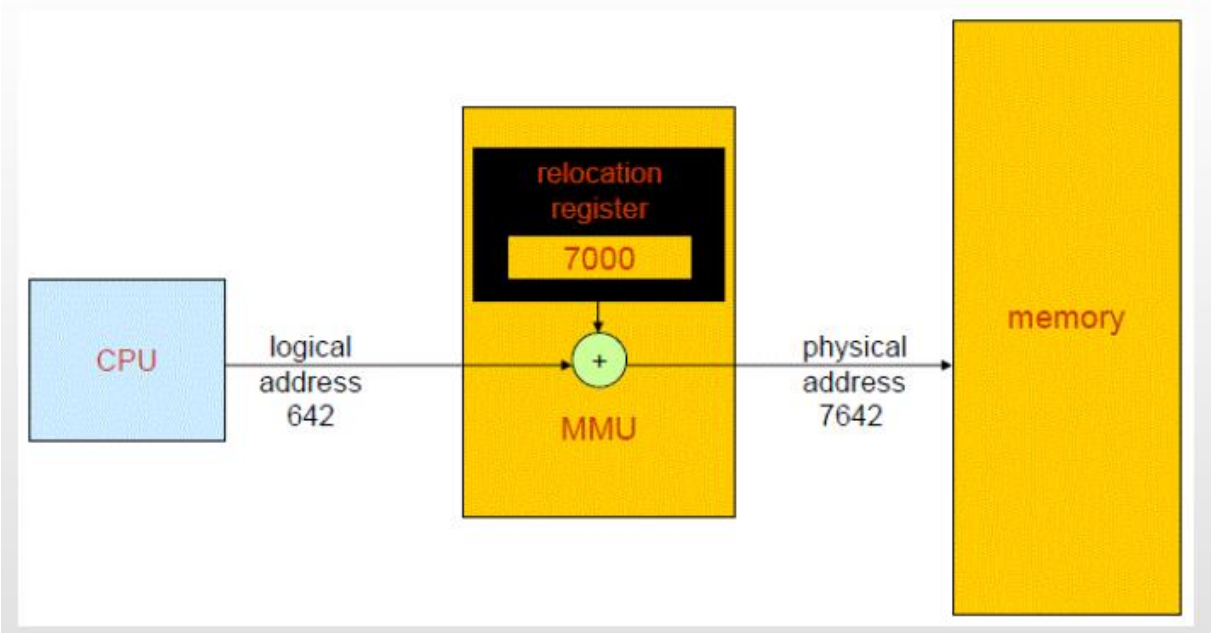
costosa de aplicar en comparación con los beneficios que pueda traer. Por otro lado, de la misma forma que en el método de particiones fijas, resulta difícil conocer con anticipación el espacio que un proceso dado requerirá a lo largo de su historia de ejecución, por lo que pueden surgir problemas en 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?

- Para cada partición el S.O debe conocer el registro base (donde comienza la memoria asignada al proceso) y el registro límite (donde termina la memoria asignada al proceso).
  Esto genera problemas:
  - Necesidad de almacenar el espacio de direcciones de forma continua en memoria física.
  - Se mantienen partes del proceso que pueden ser innecesarias en el momento.

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

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



3.- Al trabajar con particiones fijas, los tamaños de las mismas se pueden considerar:

- **Particiones de igual tamaño.**
  - Fácil de implementar.
  - Pueden generar mucha fragmentación interna
  - El uso de algoritmos implica una carga más para el procesador
  - Límite de espacio máximo aunque haya espacio libre
- **Particiones de diferente tamaño.**
  - Hacen mejor uso del espacio

- Genera fragmentación externa
- El uso de algoritmos implica una carga más para el procesador

Cite ventajas y desventajas de estos 2 métodos.

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

- ***a.*** Fragmentación interna:
  - ***i.*** Producido en particiones fijas.
  - ***ii.*** Porción de partición que queda sin utilizar.
- ***b.*** Fragmentación externa:
  - ***i.*** Producido en particiones dinámicas.
  - ***ii.*** Son huecos que van quedando en la memoria a medida que los procesos terminan.
  - ***iii.*** Estos huecos si bien son memoria libre, al no estar contiguos entre sí un proceso no puede ser alocado.

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

El problema se soluciona compactando → pero es costoso.

5.- Paginación

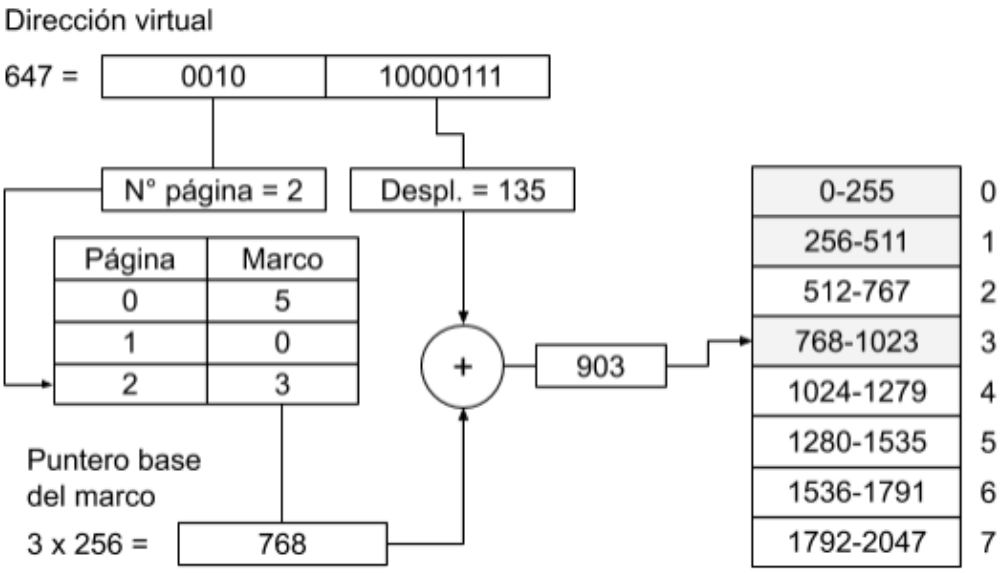
a) Explique cómo trabaja este método de asignación de memoria.

- ➔ La memoria física es dividida lógicamente en pequeños trozos de igual tamaño (marcos).
- ➔ La memoria lógica (de los procesos) es dividida en trozos de igual tamaño que los marcos (esto genera paginas).
- ➔ El SO debe mantener una tabla de páginas por cada proceso, donde cada entrada de dicha tabla contiene el marco en la que se coloca dicha página. Se guarda la base del marco que se cargó en la PCB. Hay puntero a la tabla.
- ➔ La dirección lógica es interpretada como:
  - Un numero de página y un desplazamiento dentro de la misma.
- ➔ Como se puede observar, se rompe la continuidad.
- ➔ Puede causar fragmentación interna (menos rompe bolas que la externa).

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

El SO debe mantener una tabla de páginas por cada proceso, donde cada entrada de dicha tabla contiene el marco en la que se coloca dicha página. Se guarda la base del marco que se cargó en la PCB

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



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

Se puede producir fragmentación interna, pero no tan importante como en las particiones fijas.

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

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

Diferencias:

- La paginación divide al proceso en varias particiones.
- La partición fija coloca todo un proceso de forma continua en una 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 de encuentran desde la dirección física 0.**

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

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

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

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

I. **35**
II. **512**
III. **2051**
IV. **0**
V. **1325**
VI. **602**

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

Dir. Lógica mod Tam. Página = Desplazamiento

Dir. Física = Inicio o base del frame + desplazamiento

Dir Lógica	DIV (512) N° Pag	MOD (512) Desplazamiento	Marco (base)	Dir. Física	Valida
35	0	35	M3 1536	1536 + 35 =1527	Si
512	1	0	M5 2560	2560 + 0 = 2560	Si

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

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

I. **509**
II. **1500**
III. **0**
IV. **3215**
V. **1024**
VI. **2000**

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

Dir. física mod Tam Marco = Desplazamiento

Dir. lógica = (N° página * tam. página) + desplazamiento

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

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

48 de fragmentación interna

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

a) ¿Cuantos bits son necesarios para representar una dirección lógica?

3 bits para página, 10 para desplazamiento → Son necesarios 13 bits.

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

5 bits para marco, 10 para desplazamiento → Son necesarios 15 bits.

9.- Segmentación

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

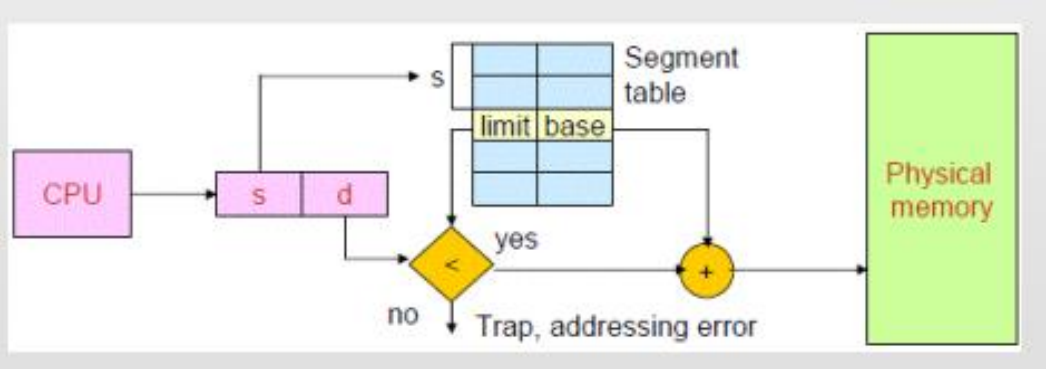
Segmentación

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

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

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

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



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

Se puede producir fragmentación externa

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

Similitudes:

- Ambas pueden generar fragmentación externa
- Ambas se basan en la idea de distribuir espacios de tamaño variable a los procesos en forma dinámica (según la necesidad particular)

Diferencias:

- El particionamiento dinámico requiere que todo el proceso se encuentre cargado de forma continua en memoria.
- La segmentación subdivide y distribuye las 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.

Similitudes:

- Ambas técnicas permiten la distribución en sectores no contiguos de memoria, y tampoco requieren que dichos sectores sean asignados en el mismo orden que en el programa principal
- Se requiere de estructuras adicionales

Diferencias:

- La paginación ttransparente al programador.
- La paginación elimina fragmentación externa.
- La segmentación es visible al programador.
- La segmentación facilita modularidad, estructuras de datos grandes y da mejor soporte a la compartición y protección.

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

Tabla de Segmentos	
Num. 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,pagina,desplazamiento):

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

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

iii) **(3,1,10) =** 500 + 120 + 10 = 5130
iv) **(2,3,5) =** 1500 + 0 + 5 = 1505

13.- Memoria Virtual

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

La memoria virtual permite ejecutar procesos que requieren más memoria que la disponible en el sistema, manteniendo en memoria principal solo aquella memoria que el proceso este utilizando y el resto en disco. De esta forma el usuario ya no debe preocuparse por las limitaciones de memoria física.

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

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

c) Al implementar está técnica utilizando paginación por demanda, las tablas de paginas de un proceso deben contar con información adicional además del marco donde se encuentra la página. ¿Cuál es está información? ¿ Porque 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, generando una interrupción. 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

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

1. Se genera el trap.
2. El SO bloquea al proceso (la CPU toma otro proceso).
3. El SO busca un marco libre en la memoria y genera una operación de E/S que le pide al disco, para copiar en dicho marco la página deseada.
4. La E/S avisa por interrupción cuando finaliza.
5. El SO actualiza la tabla de páginas del proceso.
   - Colocando el bit V en 1, en la página correspondiente.
   - Coloca la dirección base del marco donde se colocó la página.
6. El proceso pasa del estado bloqueado a listo para ejecutar
7. Cuando vuelva a ser asignado al CPU, el proceso comenzará desde la instrucción que generó el fallo en primera instancia

15.- Direcciones:

a) 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 = 2^{32} bytes → 4 GB → 4.294.967.296 bytes

ii) Si el tamaño de pagina es de 512KB. ¿Cuál es el número máximo de paginas que puede tener un proceso?

4.294.967.296 / 1024 = 4194304 KB

4194304 / 512 = 8192

iii) Si el tamaño de pagina es de 512KB. y se disponen de 256 MB. De memoria real ¿Cuál es el número de marcos que puede haber?

256 * 1024 = 262144 KB

262144 / 512 = 512

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?

8192 * 2 KB = 16384 KB = 16 MB

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 paginas.

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.**

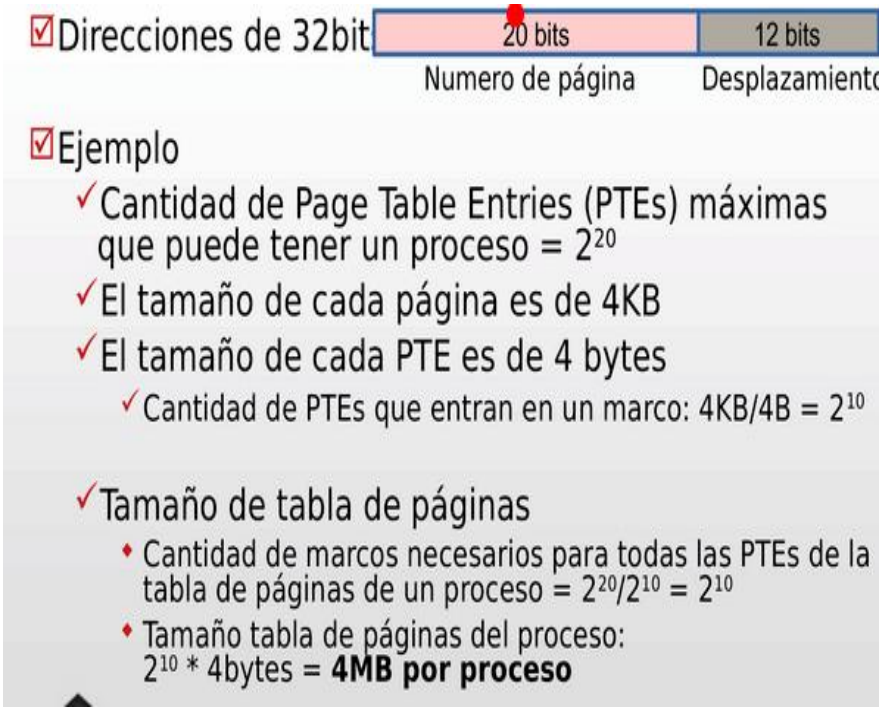
Explique brevemente como trabajan estos enfoques e indique como se realiza la transformación de la dirección virtual en dirección física.

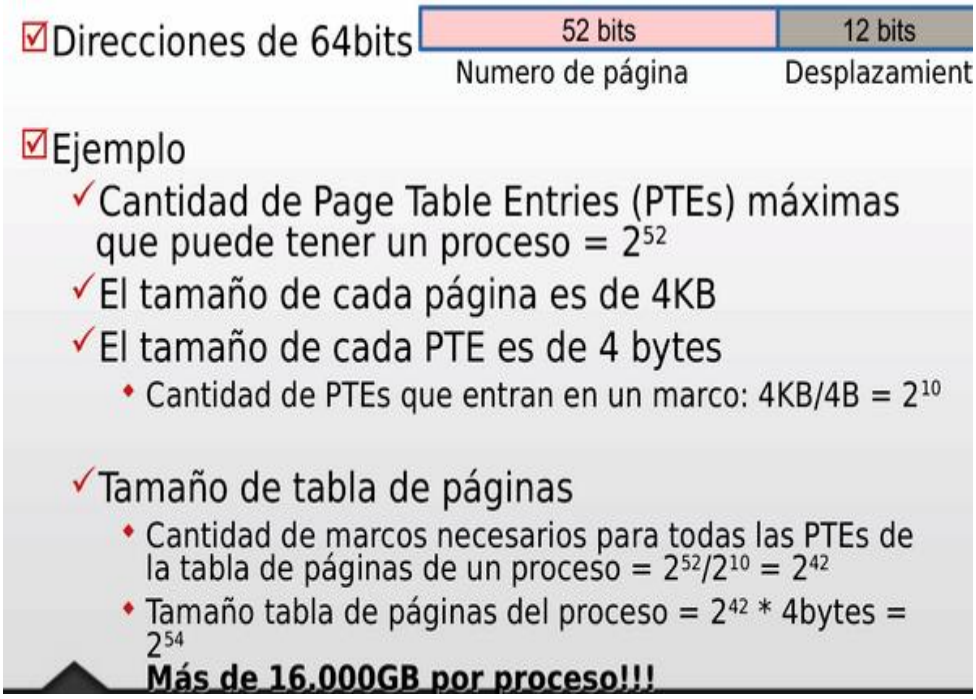
Tabla de paginas

- Cada proceso tiene su tabla de páginas.
- El tamaño de la tabla de páginas depende del espacio de direcciones del proceso.
- Puede alcanzar un tamaño considerable.
- Formas de organizar:
    - Tabla de 1 nivel: tabla única lineal
    - Tabla de 2 niveles: o más, multinivel
    - Tabla invertida: Hashing

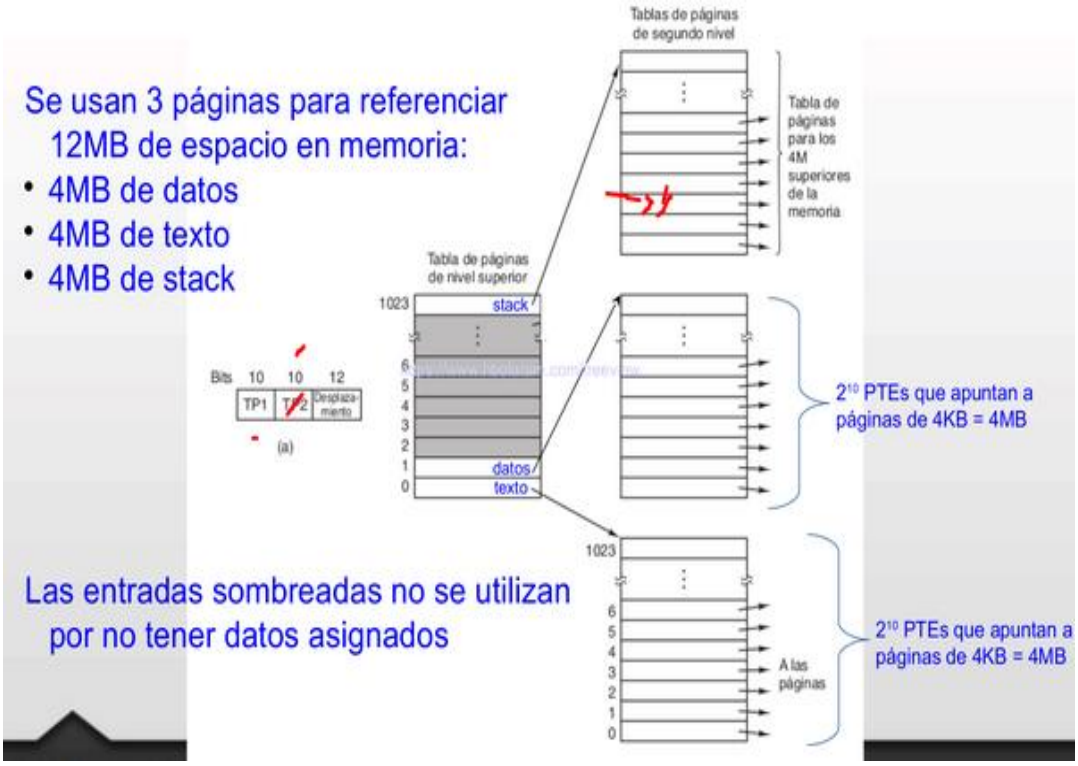
➢ Tabla de 1 nivel

- La forma de organizar la tabla de páginas depende del HW subyacente.





- Tabla de 2 niveles



- El propósito de la tabla de páginas multinivel es dividir la tabla de páginas lineal en múltiples tablas de páginas.
- Cada tabla de paginas suele tener el mismo tamaño pero se busca que tengan un menor numero de paginas por tabla.
- La idea general es que cada tabla sea más pequeña.
- Se busca que la tabla de paginas no ocupe demasiada RAM.

- Además solo se carga una parcialidad de la tabla de paginas (la que se necesite usar).
- Existe un esquema de direccionamientos indirectos.
    - Beneficios?
        - Las tablas de segundo nivel (o más) se pueden llevar a memoria secundaria, liberando RAM.
    - Desventaja?
        - Más de un acceso a memoria para obtener un dato.

➢ Tabla invertida

- Utilizada en arquitecturas donde el espacio de direcciones es muy grande.
- Las tablas de páginas ocuparían muchos niveles y la traducción es costosa.
- Por esa razón se usa ésta técnica.
    - Por ejemplo, si el espacio de direcciones es de 2^64 bytes, con páginas de 4KB, necesitamos una tabla de páginas con 2^52 entradas.
    - Si cada entrada es de 8 bytes, la tabla es de más de 30 millones de Gigabytes.
- Hay una entrada por cada marco de página en la memoria real. Es la visión inversa a la que veníamos viendo.
- Hay una sola tabla para todo el sistema.
- El espacio de direcciones de la tabla se refiere al espacio físico de la RAM (todo lo que esté cargado), en vez del espacio de direcciones virtuales de un proceso.
- El número de página es transformado en un valor de HASH.
- El HASH se usa como índice de la tabla invertida para encontrar el marco asociado.
- Se define un mecanismo de encadenamiento para solucionar colisiones (el hash da igual para dos direcciones virtuales).
    - Solo se mantienen las entradas de tablas de página (PTE) de páginas presentes en memoria ram.
        - La tabla invertida se organiza como tabla hash en RAM.
- Se busca indexadamente por número de página virtual.
- Si está presente en tabla, se extrae el marco de página y sus protecciones.
- Si no está presente en tabla, corresponde a un page fault.

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:

Pagina	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 encuentras 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)

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

Dir. Lógica mod Tam. Página = Desplazamiento

Dir. Física = Inicio o base del frame + desplazamiento

a) 1052

1052 / 512 = 2

1052 % 512 = 28

Fallo de página.

b) 2221:

2221/ 512 = 4

2221 % 512 = 173

Fallo de página.

c) 5499:

5499 / 512 = 10

5499 % 512 = 379

Dirección invalida.

d) 3101:

3101 / 512 = 6

3101 % 512 = 29

Dirección invalida.

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.**
- **Un tamaño de página grande.**

Tamaño de la pagina

- ➔ Pequeño
  - Menor fragmentación interna (menos probabilidad de que queden espacios libres que sean muy grandes).
  - Más paginas requeridas por proceso → tablas de páginas más grandes.
  - Más paginas pueden residir en memoria.
- ➔ Grande
  - Mayor fragmentación interna.

- La memoria secundaria está diseñada para transferir grandes bloques de datos más eficientemente → es más rápido mover páginas hacia la memoria ram.

➔ Relación con la E/S
- Vel de transferencia: 2 MB/s
- Latencia: 8 ms
- Búsqueda: 20.
- Búsqueda y latencia tienen que ver con el cabezal del disco.

➔ Pagina de 512 bytes.
- 1 pagina → total: 28,2 ms
- Solo 0,2 ms de transferencia (1%) → transferencia real (sin búsqueda ni latencia).
- 2 paginas → 56,4ms

➔ Pagina de 1024 bytes
- Total: 28,4 ms
- Solo 0,4 ms de transferencia.

➔ Vemos que a mayor tamaño de página se transfiere mejor

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 como trabajan estas 2 políticas.

- Asignación Fija: a cada proceso se le asigna una cantidad arbitraria de marco. A su vez para el reparto se puede usar:
  - Reparto equitativo: se asigna la misma cantidad de marcos a cada proceso → m div p .
  - Reparto proporcional: se asignan marco en base a la necesidad que tiene cada proceso → Vp . m / Vt .
- Asignación Dinámica: los procesos se van cargando en forma dinámica de acuerdo a la cantidad de marcos que necesiten

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

I. **Reparto Equitativo**
II. **Reparto Proporcional**

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

Siendo

m = 40 (marcos)

p = 4 (procesos)

Vp = Paginas usadas por proceso

Vt = 15 + 20 + 20 + 8 = 63 (sumatoria de Vp (todas las paginas usadas en total))

I. Reparto Equitativo

m / p ⟶ 40 / 4 = 10 marcos a cada proceso

II. Reparto Proporcional

Vp * m / Vt

Proceso 1 → 15 * 40 / 63 ≈ 10

Proceso 2 → 20 * 40 / 63 ≈ 13

Proceso 3 → 20 * 40 / 63 ≈ 13

Proceso 4 → 8 * 40 / 63 ≈ 5

c) ¿Cual de los 2 repartos usados en b) resulto más eficiente? ¿Por qué?

El proporcional, debido a que el equitativo deja para el proceso 4 dos marcos sin utilizar, al mismo tiempo que hay procesos que requieren más marcos. En el proporcional si un proceso usa la cantidad de marcos que requiere puede tener marcos libres para dárselos a los procesos que necesiten.

20.- Reemplazo de páginas (selección de una victima):

¿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.

- FIFO
- LRU
- FIFO Segunda Chance
- OPT

b) Analice su funcionamiento. ¿Como los implementaría?

- Optimo: es sólo teórico, es el mejor. Selecciona como víctima a la página cuyo próxima referencia se encuentra más lejana a la actual. Es imposible de implementar ya que no se conoce los futuros eventos
- FIFO: el más sencillo, la página más vieja en la memoria es reemplazada y esta puede ser necesitada pronto, debido a esto es el peor en cuanto a la tasa de fallos de página. Trata a los frames en uso como una cola circular
- LRU: requiere soporte del hardware para mantener timestamps de acceso a las páginas. Favorece a las páginas menos recientemente accedidas. 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. Cada página debe tener información del instante de su ´ultima referencia, por lo tanto, el overhead es mayor
- Segunda Chance: un avance del FIFO tradicional que beneficia a las páginas más referenciadas, aproximándose en mayor medida a la idea de trabajo con el conjunto residente.
    - Se utiliza un bit adicional → bit de referencia
    - Cuando la página se carga en memoria, el bit R se pone a 0.
    - Cuando la página es referenciada el bit R se pone en 1
    - La víctima se busca en orden FIFO. Se selecciona la primer página cuyo bit R está en 0
    - Mientras se busca la víctima cada bit R que tiene el valor 1, se cambia a 0

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

El sistema operativo reserva uno o varios marco para la descarga asincrónica de páginas.

Cuando es necesario descargar una página modificada:

1) La página que provoco el fallo se coloca en un frame designado a la descarga asincrónica.
2) El SO envía la orden de descargar asincrónicamente la página modificada mientras continua la ejecución de otro proceso.
3) El frame de descarga asincrónica pasa a ser el que contiene a la página víctima que ya se descargó correctamente.

21.- Alcance del reemplazo

Al momento de tener que seleccionar una página víctima, el SO puede optar por 2 políticas a utilizar:

- **Reemplazo local**
- **Reemplazo global**

a) Describa como trabajan estas 2 políticas.

Reemplazo Global

- El fallo de página de un proceso puede reemplazar la página de cualquier proceso.
- El SO no controla la tasa de page-faults de cada proceso.
- Puede tomar frames de otro proceso aumentando la cantidad de frames asignados a él.
- Un proceso de alta prioridad podría tomar los frames de un proceso de menor prioridad.

Reemplazo Local

- El fallo de página de un proceso solo puede reemplazar sus propias páginas (de su conjunto residente)
- No cambia la cantidad de frames asignados.
- El SO puede determinar cuál es la tasa de page-faults de cada proceso.
- Un proceso puede tener frames asignados que no usa, y no pueden ser usados por otros procesos.

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 disponen de 5 marcos. ¿Cuántos fallos de página se producirán si se utilizan las siguientes técnicas de selección de victima? (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,**

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.- Sean 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 manera:

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 paginas modificadas ¿Cuántos fallos de página se producirán si se utiliza la técnica de selección de victima:

a) Segunda Chance

b) FIFO

c) LRU

25.- Hiperpaginación (Trashing)

a) ¿Qué es?

Cuando un sistema pasa más tiempo paginando que ejecutando procesos

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.- 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%.

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?**

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%**

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

a) Instalar un procesador mas rápido

b) Instalar un dispositivo de paginación mayor → esta

c) Incrementar el grado de multiprogramación

d) Instalar mas memoria principal → esta

e) Decrementar el quantum para cada proceso

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

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)?

20 ns + (1 – 0,3 ns)* 20ns + 0,3ns * (300ns + 20ns) = 130

+

1 ns + (1 – 0,3 ns)* 20ns + 0,3ns * (300ns + 20ns) = 111

/ 2

= 120.5 ns

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)?

20 ns * 0.4 + 1 * 0.1 + (1 – 0,3 ns)* 20ns + 0,3ns * ((300ns * 0.3 + 500 * 0.7) + 20ns) = 160.1

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?

200ns = 1ns + (1 − p)* 20ns + p * ((300ns * 0.4 + 500 * 0.6) + 20ns) =

p ≈ 0.42

(use photomath)

29.- Anomalía de Belady

a) ¿Qué es?

n Ciencias de la computación, la anomalía de Belady es un fenómeno que ocurre en algunos algoritmos de reemplazo de páginas de memoria, por lo que la frecuencia de fallas de página puede aumentar con el número de fotogramas asignados a los procesos. Los algoritmos FIFO con algunas combinaciones de solicitudes de página sufren de esto.

En 1969, Belady, Nelson y Shedler identificaron secuencias de referencia que, cuando se aplicaban utilizando algoritmos de sustitución FIFO con una cierta cantidad de memoria disponible, producían el doble de fallos de página que con una cantidad menor. También asumen que dos es un límite general. En 2010, bakers e Ivanyi demuestran que se pueden construir cadenas tales que esta proporción puede ser arbitrariamente grande .

Dado que solo los algoritmos FIFO se ven afectados por esta anomalía, es suficiente usar los llamados algoritmos de pila para administrar la paginación. La búsqueda de este tipo de soluciones tuvo un impulso particular justo después del descubrimiento de Nelson y Shadler, inicialmente orientado hacia la definición del algoritmo óptimo (OPT), definido solo en forma teórica dada la imposibilidad de predecir la sucesión de referencias. Dado que no es posible implementar OPT en sistemas reales, la investigación se centró en el algoritmo LRU (menos utilizado recientemente), que en su lugar elige la página "víctima" entre las que se utilizan más tarde en el tiempo. Dado que este último enfoque también es excesivamente caro, se han identificado aproximaciones como el algoritmo de segunda oportunidad y el algoritmo de segunda oportunidad mejorado, ambos basados en el uso de bits de validez.

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 paginas: 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

Ocurre anomalía de Belady

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 pagina 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 paginas 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 64^2

- Se almacenan 4 filas por páginas.

- 1 marco para cada matriz

- Como j es la columna, primero recorro por columna es decir iría A[0,0], A[1,0], A[2,0]

- Se van a generar 32 fallos (ya que ocurren cada 4 operaciones porque 4 filas están en una página) en el primer bucle (el de j) 3 veces (uno por cada matriz).

32 * 32 * 4 = 3072

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.

Si se puede.

```
#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];
```

C[i,j] = A[i,j] + B[i,j] → Se ejecuta 64^2

- Se almacenan 4 filas por páginas.

- Va a haber 3 fallos de página (1 por cada matriz) cada 4 bucles de i (ya que hay 4 filas en 1 pagina).

64 * 3 / 4 = 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):

Proceso A

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
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
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
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

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
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
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 cual seria el conjunto de trabajo de los procesos A y B en el instante 24 (WSA(24) y WSB(24))

WSA(24) = 4, 5, 6

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

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

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

WSB(60) = 3,4,5

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

Con 8 frames no, porque la suma de los WS es 7, menor a la cantidad de frames disponibles. Con 6 si porque se necesitan mas frames de los disponibles.

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

PF=6

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

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