

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 dividida 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 estos 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

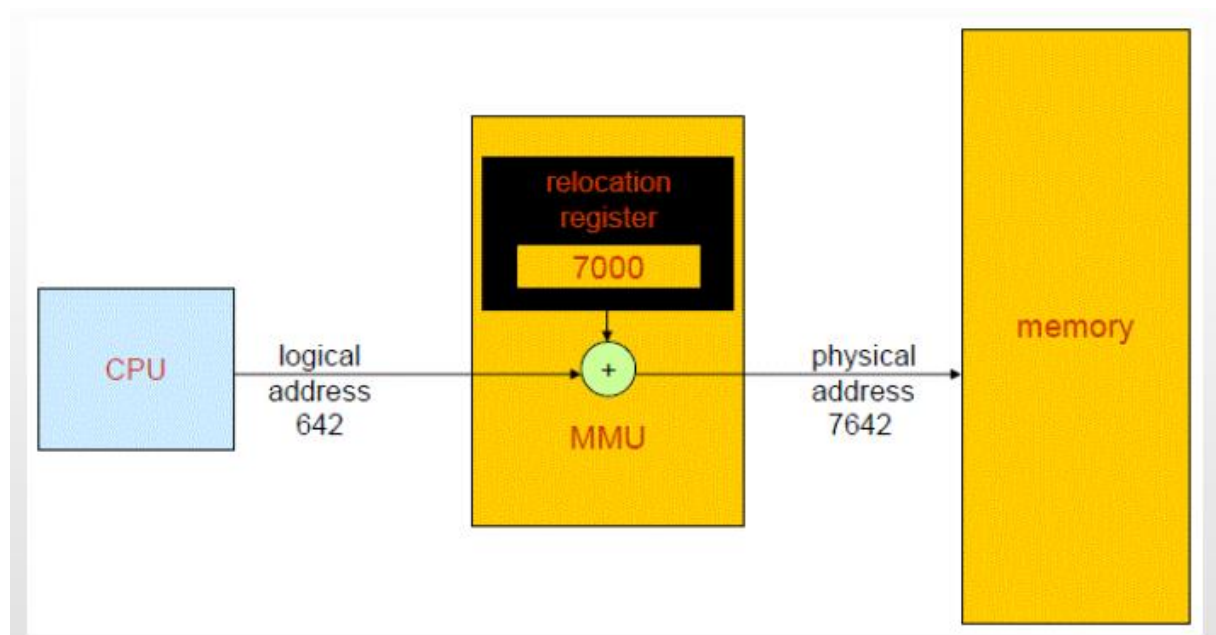
costosa de aplicar en comparación con los beneficios que pueda traer. Por otro lado, de la misma forma que en el método de particiones fijas, resulta difícil conocer con anticipación el espacio que un proceso dado requerirá a lo largo de su historia de ejecución, por lo que pueden surgir problemas en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

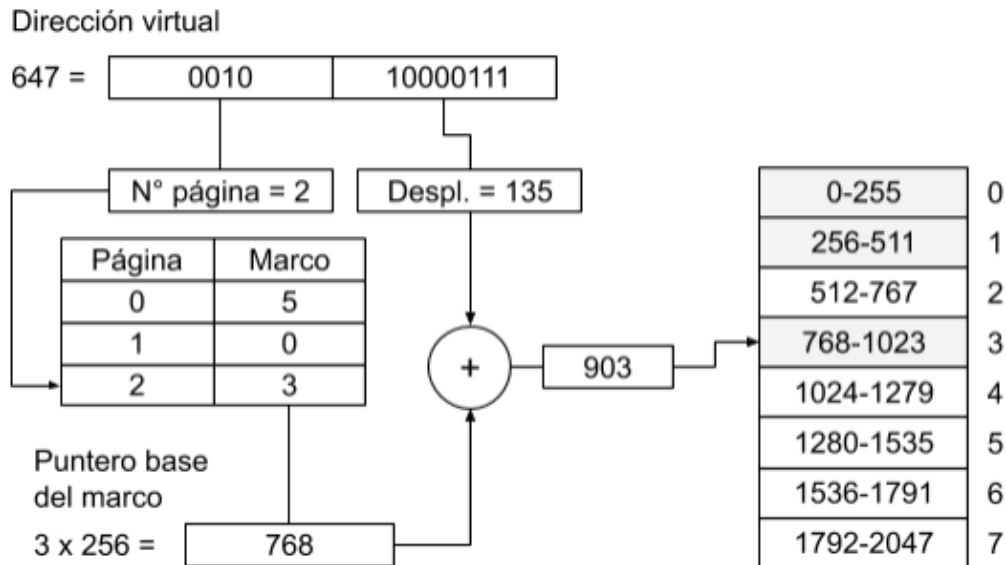
a) Explique cómo trabaja este método de asignación de memoria.

- ➔ 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 sólo 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:

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 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 = 1571	Si
512	1	0	M5 2560	2560 + 0 = 2560	Si

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

c) 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 = (Nº página * tam. página) + desplazamiento

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

d) ¿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) ¿Cuántos bits son necesarios para representar una dirección lógica?

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

b) ¿Cuántos 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.

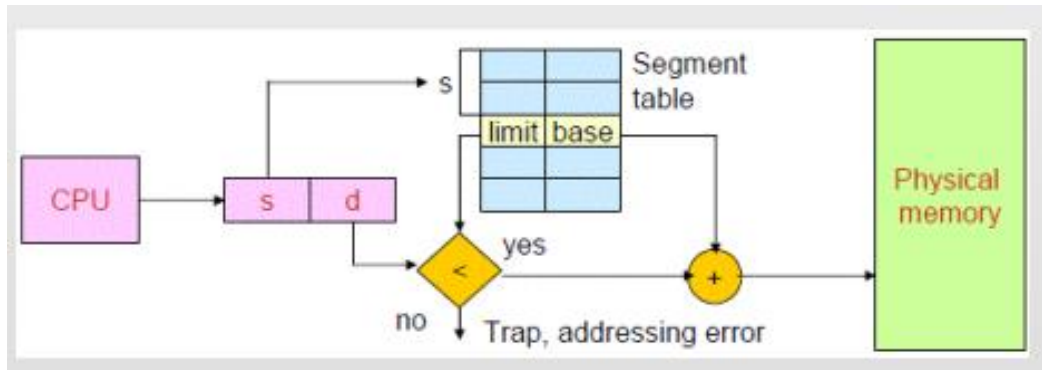
Segmentación

- ➔ 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):

- $(2,1,1) = 1500 + 20 + 1 = 1521$
- $(1,3,15) = 500 + 60 + 15 = 575$

- 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 esta 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 esta 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 cuando 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 un 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 \rightarrow 4 GB \rightarrow 4.294.967.296 bytes

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

$4.294.967.296 / 1024 = 4194304$ Kb

$4194304 / 512 = 8192$

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?

$256 * 1024 = 262144$ 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?

2^{32} direcciones máximas * 2 byte = $2^{32} * 2$ bytes \rightarrow $4.294.967.296 * 2$ bytes \rightarrow 8589934592 bytes \rightarrow 8 GB

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 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.



✓ Ejemplo

- ✓ Cantidad de Page Table Entries (PTEs) máximas que puede tener un proceso = 2^{20}
- ✓ El tamaño de cada página es de 4KB
- ✓ El tamaño de cada PTE es de 4 bytes
 - ✓ Cantidad de PTEs que entran en un marco: $4KB/4B = 2^{10}$
- ✓ Tamaño de tabla de páginas
 - ♦ Cantidad de marcos necesarios para todas las PTEs de la tabla de páginas de un proceso = $2^{20}/2^{10} = 2^{10}$
 - ♦ Tamaño tabla de páginas del proceso:
 $2^{10} * 4bytes = \mathbf{4MB \text{ por proceso}}$

✓ Direcciones de 64bits

52 bits	12 bits
Numero de página	Desplazamiento

✓ Ejemplo

- ✓ Cantidad de Page Table Entries (PTEs) máximas que puede tener un proceso = 2^{52}
- ✓ El tamaño de cada página es de 4KB
- ✓ El tamaño de cada PTE es de 4 bytes
 - ♦ Cantidad de PTEs que entran en un marco: $4KB/4B = 2^{10}$
- ✓ Tamaño de tabla de páginas
 - ♦ Cantidad de marcos necesarios para todas las PTEs de la tabla de páginas de un proceso = $2^{52}/2^{10} = 2^{42}$
 - ♦ Tamaño tabla de páginas del proceso = $2^{42} * 4bytes = 2^{54}$

Más de 16.000GB por proceso!!!

➤ Tabla de 2 niveles

Se usan 3 páginas para referenciar
12MB de espacio en memoria:

- 4MB de datos
- 4MB de texto
- 4MB de stack



- 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:

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)

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

Dir. Lógica mod Tam. Página = Desplazamiento

Dir. Física = Inicio o base del frame + desplazamiento

a) 1052

$$1052 / 512 = 2$$

$$1052 \% 512 = 28$$

Fallo de página.

b) 2221:

$$2221 / 512 = 4$$

$$2221 \% 512 = 173$$

Fallo de página.

c) 5499:

$$5499 / 512 = 10$$

$$5499 \% 512 = 379$$

Dirección inválida.

d) 3101:

$$3101 / 512 = 6$$

$$3101 \% 512 = 29$$

Dirección inválida.

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 página

➔ Pequeño

- Menor fragmentación interna (menos probabilidad de que queden espacios libres que sean muy grandes).
- Más páginas requeridas por proceso → tablas de páginas más grandes.
- Más páginas 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 → $V_p \cdot m / V_t$.
- **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)

V_p = Paginas usadas por proceso

$V_t = 15 + 20 + 20 + 8 = 63$ (sumatoria de V_p (todas las paginas usadas en total))

I. Reparto Equitativo

$m / p \rightarrow 40 / 4 = 10$ marcos a cada proceso

II. Reparto Proporcional

$V_p * m / V_t$

Proceso 1 $\rightarrow 15 * 40 / 63 \approx 10$

Proceso 2 $\rightarrow 20 * 40 / 63 \approx 13$

Proceso 3 $\rightarrow 20 * 40 / 63 \approx 13$

Proceso 4 $\rightarrow 8 * 40 / 63 \approx 5$

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 víctima):

¿Qué sucede cuando todos los marcos en la memoria principal están usados por las páginas de los procesos y se produce un fallo de página? El SO debe seleccionar una de las páginas que se encuentra en memoria como víctima, y 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. ¿Cómo 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 última 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 reemplazada puede estar modificada. ¿Qué acciones debe llevar el SO cuando se encuentra ante esta situación?

El sistema operativo reserva uno o varios marcos 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.