administrador del sistema el que tenga que encargar de detectar el interbloqueo y recuperarlo.

Preguntas de autoevaluación tema 6

6.1. Definir los siguientes conceptos: a) Espacio del núcleo. b) Espacio de usuario.

a).Un SO para ser ejecutado debe estar cargado en la memoria principal. El SO se carga cuando arranca el computador mediante un programa cargador. Al espacio ocupado por el código, las estructuras de datos y la pila (o pilas) del núcleo del SO se le denomina espacio del núcleo.

b).Aparte del SO, en la memoria principal pueden estar cargados (total o parcialmente) uno o varios procesos. Al espacio de memoria principal ocupado por la imagen de un proceso, es decir, por su espacio de direcciones de memoria lógica, se le denomina espacio de usuario.

6.2. ¿Qué es el área de intercambio en memoria secundaria?

Cuando se desea ejecutar un programa en el SO tiene que crear un proceso asociado a un programa en la memoria secundaria y cargarlo en la principal. Para ello debe buscar el archivo ejecutable, crear una copia de la imagen del proceso en la memoria secundaria y cargar dicha imagen en memoria principal. Algunos SO directamente crean y cargan la imagen del proceso en la memoria principal sin crear una copia en la memoria secundaria.

El SO reserva espacio en la memoria secundaria, típicamente un HD, para almacenar las copias de las imágenes de los procesos. A dicho espacio se le denomina área de intercambio. El tamaño de esa zona suele estar predefinido, aunque algunos sistemas permiten que su tamaño máximo sea configurado por el administrador.

6.3. ¿En qué consiste la operación de intercambio de procesos?

También llamado swapping es la operación de cargar la imagen de un proceso desde el área de intercambio a la memoria principal o viceversa.

6.4. ¿En qué casos se suele realizar la operación de intercambio fuera de memoria principal?

Swapping out o intercambio fuera de memoria principal se suele realizar en los siguientes casos:

- ***Cuando se requiere espacio en la memoria principal para cargar otros procesos.***
- ***Cuando un proceso necesita más espacio en memoria principal debido al crecimiento de su región de datos o de su región de pila.***
- ***Cuando es necesario regular el grado de multiprogramación del sistema, es decir, el número de procesos cargados en memoria principal.***

¿Y la operación de intercambio dentro de memoria principal?

Swapping in o intercambio dentro de memoria principal se suele realizar en los siguientes casos:

- ***Se desea aumentar el grado de multiprogramación del sistema para mejorar su rendimiento.***
- ***Existe el espacio en memoria principal para cargar más procesos.***

Existe un proceso de mayor prioridad en el área de intercambio.

6.5. ¿Cuáles son las principales tareas del intercambiador?

- ***Gestionar y asignar el espacio de intercambio.*** Para ello el SO mantiene la estructura de datos con información del espacio libre en el área de intercambio.
- ***Seleccionar procesos para ser intercambiados fuera de memoria principal.*** El intercambiador escoge en primer lugar para ser intercambiados aquellos procesos que se encuentran en estado bloqueado de acuerdo a los siguientes criterios: menor prioridad del proceso y mayor tiempo de residencia en

memoria. En segundo lugar escoge a procesos en el estado preparado, con los mismos criterios que el caso anterior.

- ***Seleccionar procesos para ser intercambiados dentro de memoria principal.*** Se suelen escoger aquellos procesos preparados para ejecución que llevan más tiempo en el área de intercambio o de mayor prioridad.

6.6. ¿Qué dos razones justifican la existencia del área de intercambio?

- ***Una operación de lectura y escritura en el área de intercambio es más rápida que en el sistema de archivos ya que no hay que traducir una ruta de acceso al archivo ni comprobar permisos de accesos.***

- ***La imagen de un proceso es una entidad dinámica cuyo contenido suele diferir del contenido del archivo ejecutable conforme se va ejecutando el proceso.*** La imagen del proceso debe ser guardada para poder continuar con la ejecución del proceso sin tener que reiniciarla desde el principio.

6.7. Describir cómo se realiza la asignación de memoria en sistemas monoprogramados.

Se divide en dos particiones, una reservada para contener permanentemente a aquellas partes del SO que deban estar siempre en memoria principal. Al conjunto de estas partes se le denomina monitor del SO.

La memoria principal no ocupada por el SO conforma otra partición que está disponible para almacenar de forma temporal a un proceso de usuario o a uno de sistema asociado a alguna parte no residente en el SO. Puesto que solo puede estar cargado un proceso en memoria principal, los procesos se ejecutan una a continuación de otro. Cuando un proceso finaliza, el SO puede cargar otro proceso en la única partición disponible para tal efecto.

En función de si la partición de memoria asociada al SO se encuentra en la parte inferior o superior es posible distinguir las siguientes configuraciones:

- ***Exclusivamente con una RAM***. La partición del SO ocupa la parte inferior de la memoria principal.

- ***Con una RAM y una ROM donde el SO ocupa la ROM desde Dir_{f1} hasta Dir_{max}.*** Es una configuración muy poco flexible ya que el código del SO al residir en una ROM no puede ser modificado y actualizado.

- ***Con una RAM y una ROM donde la partición del SO ocupa la parte inferior de la RAM y la memoria ROM contiene la tabla de vectores de interrupciones del hardware y las rutinas de servicio de las interrupciones, a esta memoria se le conoce también por BIOS***.

6.8. ¿En qué consiste la técnica de particionamiento fijo?

En sistemas multiprogramados se cargan múltiples procesos en el estado preparado para ejecución en la memoria principal. En un sistema con un único procesador, solo uno de los procesos podrá encontrarse en el estado ejecutándose. Si el proceso en ejecución se bloquea en espera de la aparición de un evento, como por ejemplo la finalización de una operación de E/S, otro proceso en el estado preparado para ejecución puede ser planificado para ser ejecutado. La multiprogramación permite mejorar el rendimiento del sistema, evitando que sus recursos se encuentren inactivos.

La forma más sencilla de gestionar la memoria en sistemas multiprogramados consiste en dividir la memoria principal en un número fijo N de particiones que pueden ser de igual o de diferente tamaño. Esta división puede realizarse, por ejemplo, manualmente en tiempo de arranque del sistema. A esta técnica se le conoce como particionamiento fijo.

6.9. ¿Qué información contiene la tabla de descripción de particiones?

El SO mantiene una estructura de datos, denominada tabla de descripción de particiones, que contiene la siguiente información por cada partición: dirección física de comienzo (también llamada dirección base), tamaño y estado (si está libre u ocupada). El SO se encarga de gestionar la entrada de cada partición, ya que la dirección base y el tamaño son fijados en tiempo de arranque.

Cuando se asigna una partición a un proceso, el SO guarda el número de partición asignada en el bloque de control de proceso. Además marca como ocupada la entrada de la tabla de descripción de particiones asociada a dicha partición.

Cuando un proceso finaliza su ejecución o es intercambiado al área de intercambio en memoria secundaria su partición de memoria queda libre para ubicar a otro nuevo proceso o a un proceso intercambiado procedente del área de intercambio.

Cuando hay que asignar una partición a un proceso se busca una partición libre en la tabla de descripción de particiones.

6.10. ¿Qué desventajas presenta el particionamiento fijo con particiones de igual tamaño?

A pesar de tener la ventaja de que no importa qué partición libre se asigne a un proceso al ser todas iguales presenta las siguientes desventajas:

- ***Limitación del tamaño máximo de los procesos que se pueden cargar en memoria principal.*** Un proceso cuyo espacio de direcciones lógicas sea de mayor tamaño que el tamaño fijado para las particiones no podrá ser cargado en memoria y en consecuencia no podrá ser ejecutado. Existe una excepción y es si el programa ha sido diseñado con superposiciones (overlay). Una superposición es un conjunto de instrucciones y datos que se pueden y ejecutar y acceder de forma autónoma. Un programa se compone de varias superposiciones que se almacenan en memoria secundaria. En un determinado instante de tiempo solo es necesario, que para que pueda progresar en su ejecución, que esté cargada en memoria principal una única superposición del proceso. Cuando una superposición de un programa termina de ejecutarse invoca a otra superposición que es cargada desde memoria secundaria en el espacio de memoria principal que ocupaba la anterior. Esta técnica permite ahorrar espacio de memoria. Sin embargo, el diseño y la programación de un programa con superposiciones bastante tediosa, sobre todo en el caso de programas largos.

- ***Fragmentación interna.*** Si el tamaño del espacio de direcciones lógicas de un proceso es inferior al tamaño de la partición que se la ha asignado entonces existe un espacio no utilizado dentro de dicha partición. Nótese que este espacio no puede ser utilizado por ningún otro proceso, es decir, es un espacio desperdiciado. Al desaprovechamiento de la memoria dentro de una partición se le denomina fragmentación interna.

6.11. ¿Qué ventajas e inconvenientes presenta el uso de una cola por partición?

La ventaja es que intenta minimizar la fragmentación interna dentro de cada partición. Sin embargo, puede provocar que las particiones de mayor tamaño se queden vacías mientras que las colas asociadas a particiones de menor tamaño están llenas de procesos esperando. Es decir, está infrautilizando la memoria.

Una posible solución consiste en mantener una única cola de procesos para todas las particiones.

6.12. ¿Qué criterios de búsqueda se emplean con las particiones de distinto tamaño y una única cola de procesos?

- ***Criterio del primer ajuste:*** Consiste en buscar el proceso más cercano a la cabecera de la cola cuyo tamaño sea igual o menor al de la partición libre. Este criterio de búsqueda produce búsquedas rápidas, sin embargo no minimiza la fragmentación interna.

- ***Criterio del mejor ajuste*****:** Busca en toda la cola al proceso de mayor tamaño que entre dentro de la partición. Minimiza la fragmentación interna pero produce búsquedas más lentas. Además discrimina a los procesos de menor tamaño, como suelen ser los asociados a tareas interactivas. Obviamente estos procesos deben ejecutarse rápidamente y no ser relegados continuamente por otros procesos de mayor tamaño. Una posible solución sería imponer un valor máximo de veces que un proceso puede ser discriminado.

6.13. ¿Qué desventajas presenta el particionarniento fijo con particiones de distinto tamaño?

Los mismos que con el uso de particiones de igual tamaño: la existencia de fragmentación interna y la limitación del tamaño

máximo del espacio de direcciones lógicas del proceso que puede ser cargado en memoria. Sin embargo ambos problemas son de una magnitud menor que en el caso de particiones de tamaño fijo.

6.14. ¿Cómo se realiza la traducción de direcciones lógicas a físicas y la protección en la técnica de gestión de memoria mediante particionarniento fijo?

Requiere el uso de dos registros denominados registro base y registro límite.

Cuando un proceso es planificado para ser ejecutado, el SO carga en el registro base la dirección física base de la partición de memoria principal donde se encuentra cargado el proceso. Además carga el registro límite con la dirección lógica más alta del espacio de direcciones lógicas del proceso, es decir, con el tamaño de su espacio lógico. El SO obtiene la información que carga en estos registros del bloque de control del proceso.

Cuando el procesador hace una referencia a una dirección lógica se compara con el valor de registro límite. Si la dirección es mayor que dicho valor, entonces el hardware genera una excepción por error de direccionamiento que al ser tratada por el SO típicamente suele producir la terminación del proceso y la notificación de dicha circunstancia al usuario propietario del proceso. Si la dirección lógica es menor que el valor del registro límite entonces se le suma el contenido del registro base para generar la dirección física correspondiente.

Nótese que el registro base impide el acceso de un proceso a direcciones físicas inferiores a las asignadas a su espacio lógico. Por otra parte el registro límite impide el acceso a direcciones físicas superiores a las asignadas a su espacio lógico. De esta forma se protege el espacio de direcciones lógicas del SO o el espacio de otro proceso contra el acceso involuntario o intencionado del proceso en ejecución.

6.15. ¿Cuáles son las principales ventajas e inconvenientes de la técnica de gestión de memoria mediante particionarniento fijo?

La principal ventaja es que es sencilla de implementar para el SO, únicamente requiere una tabla de descripción de particiones y

de una o varias colas para contener a los procesos que deben ser cargados en memoria. La gestión de estas estructuras produce muy poca sobrecarga.

Los principales inconvenientes de esta técnica son que provoca fragmentación interna y que el tamaño máximo de proceso que puede ser cargado en memoria viene limitado por la partición de mayor tamaño que se defina.

Además el número máximo de procesos cargados en memoria principal o grado de multiprogramación está limitado por el número de particiones que se definan en el momento de arrancar el sistema. Ello repercute en una disminución del rendimiento máximo del sistema, es decir, del grado de utilización del procesador y de los canales de E/S.

6.16. ¿En qué consiste la técnica de gestión de memoria denominada particionamiento dinámico?

Se caracteriza en que el número de particiones no es fijo sino que varía a lo largo del tiempo. Así, en la memoria principal se distinguen dos tipos de elementos: particiones y huecos.

Inicialmente al arrancar hay una partición y un gran hueco que se va rellenando con particiones.

6.17. ¿Qué es la fragmentación externa?

Con el paso del tiempo, al crearse nuevos procesos e intercambiarse otros, la memoria se va fragmentando entremezclando particiones asignadas con huecos. En la memoria principal irán apareciendo huecos cada vez más pequeños, algunos de los cuales no pueden ser utilizados por sí solos para crear otras particiones debido a su pequeño tamaño. A este desaprovechamiento de memoria entre particiones debido a la dispersión del espacio libre en una serie de huecos de pequeño tamaño se le denomina fragmentación externa.

¿Cómo se puede solucionar?

La fragmentación interna supone un problema cuando aunque exista tamaño suficiente en memoria este no es contiguo. Una forma de solucionarlo es la compactación de memoria, que consiste en agrupar todos los huecos pequeños en un único hueco. Una

posible implementación de esta técnica consiste en mover todas las particiones hacia un extremo de la memoria de forma que estén contiguas. Así se consigue que los huecos se desplacen hacia el otro extremo y se agrupen para formar un único hueco. El principal inconveniente de la compactación es que consume bastante tiempo de uso de procesador, produciendo una alta sobrecarga.

6.18. ¿Qué estructuras de datos debe mantener el sistema operativo para implementar el particionamiento dinámico?

Debe de contener una o varias estructuras de datos que contengan la información sobre las particiones y los huecos existentes en la memoria principal.

Una posibilidad es mantener una única estructura de datos que mantenga simultáneamente las particiones y los huecos existentes en memoria principal. Puede implementarse con un mapa de bits o una lista enlazada (simple o doble). En el caso de usar una lista ésta puede ser organizada por direcciones físicas o por tamaños.

Otra posibilidad consiste en mantener dos listas independientes: una de particiones y otras de huecos. Así se consigue acelerar el proceso de búsqueda de huecos, sobre todo si dicha lista ésta organizada por tamaños. Sin embargo la operación de desasignación de memoria se ralentiza ya que cuando un proceso finaliza, o es intercambiado al área de intercambio, el SO tiene que actualizar dos estructuras de datos.

6.19. Enumerar y describir brevemente algunos de los algoritmos de búsqueda más empleados en la asignación de memoria en el particionarniento dinámico.

- ***Primer ajuste (first fit*).** Se comienza a buscar desde el principio de la estructura de datos y se asigna el primer hueco con un tamaño igual o mayor a S. Este algoritmo es sencillo de implementar. Además es rápido ya que minimiza el tiempo de búsqueda. Con el paso del tiempo este algoritmo produce pequeños huecos al comienzo de la estructura de datos.

- ***Algoritmo del siguiente ajuste (next fit).*** La búsqueda en la estructura de datos de un hueco de tamaño igual o mayor a S

se inicia desde donde finalizó la búsqueda anterior. Aunque este algoritmo se ideó con la intención de mejorar el aprovechamiento de memoria que producía el algoritmo del primer ajuste, diversas simulaciones han demostrado que con el paso del tiempo produce resultados ligeramente peores, tanto en tiempo de búsqueda como en aprovechamiento de memoria.

- ***Algoritmo del mejor ajuste (best fit).*** Se busca en toda la estructura de datos un hueco cuyo tamaño sea igual o exceda lo mínimo posible al valor S. Este algoritmo, al buscar en toda la estructura de datos, es más lento que el algoritmo del primer ajuste o del siguiente ajuste. Además produce huecos de memoria muy pequeños que no suelen ser utilizables para posteriores asignaciones, lo que obliga a realizar compactación de memoria más frecuentemente.
- ***Algoritmo del peor ajuste (worst fit).*** Se busca en toda la estructura de datos un hueco cuyo tamaño sea igual o exceda lo máximo posible el valor S, es decir, se busca el hueco más grande posible. Con ello se pretende que los huecos que se generen sean aprovechables para posteriores asignaciones y en consecuencia haya que realizar compactación menos frecuentemente. Diferentes simulaciones han demostrado que con el paso del tiempo este algoritmo no es muy efectivo reduciendo la fragmentación externa y la frecuencia de realización de compactación.

De estos cuatro el más rápido es el algoritmo del primer ajuste, lo cual es lógico ya que en principio, no necesita buscar en toda la estructura de datos. Por otra parte en cuanto a al aprovechamiento de memoria el mejor es el primer ajuste y el peor el peor ajuste.

6.20.. ¿Cómo se realiza la traducción de direcciones lógicas a físicas y la protección en la técnica de gestión de memoria mediante particionamiento dinámico?

Se implementa de igual manera que en la técnica del particionamiento fijo, es decir, se utilizan dos registros dedicados denominados registro base y registro límite. El SO, cuando se va a

ejecutar un proceso, carga en el registro base la dirección física de inicio de la partición de memoria física asociada al proceso, y carga el registro límite el tamaño de la partición.

6.21. Señala las principales ventajas e inconvenientes de la técnica de particionamiento dinámico frente a la técnica de particionarniento fijo.

Las ventajas son:

- ***Minimiza la fragmentación interna*.** Como se ha comentado la fragmentación interna de una partición es prácticamente despreciable, ya que como máximo será cercana a una unidad de asignación.
- ***Permite ejecutar procesos de mayor tamaño.*** En un determinado instante de tiempo el tamaño máximo del proceso que puede cargarse en memoria viene limitado por la existencia de un hueco de tamaño igual o mayor que lo pueda contener. Inicialmente en la memoria solo está cargado el SO, por lo que el resto de la memoria formaría un hueco de tamaño S que podría albergar un proceso de tamaño máximo menor o igual a S.
- ***Permite asignar fácilmente más espacio a procesos cuya región de datos o pila aumenta por encima del tamaño inicialmente asignado.*** Para ello el SO puede proceder de dos formas. Una opción es crear una partición de mayor tamaño y reubicar al proceso en dicha partición. Otra posibilidad es añadir a la partición del proceso el espacio de un hueco adyacente a la partición.

Las desventajas son las siguientes:

- ***Las estructuras de datos que requiere para su implementación y los algoritmos para su gestión son más complejos.*** En consecuencia se produce una mayor sobrecarga del sistema.
- ***Produce fragmentación externa.*** Lo que reduce el aprovechamiento de la memoria.

6.22. ¿En qué consiste la paginación?

Consiste en dividir la memoria principal en bloques del mismo tamaño denominados marcos de página o páginas físicas.

6.23. Cuando se usa paginación ¿en qué campos se descompone una dirección física y una dirección lógica?

El espacio de direcciones de un proceso también se divide en bloques del mismo tamaño Sp denominados páginas o páginas lógicas. Una página de un proceso se carga en un marco de página libre de memoria principal. Además las páginas de un mismo proceso no tienen porque ocupar marcos contiguos.

¿Cómo se puede determinar el tamaño de dichos campos?

Sea Cmp la capacidad de memoria principal, el número Nmp de marcos de página de tamaño Sp en que se divide la memoria principal se obtiene de la siguiente forma:

$$Nmp = floor\left(\frac{Cmp}{Sp}\right)$$

Cada marco de página tiene asignado un identificador numérico entero positivo j que lo identifica de forma unívoca. Al número j se le denomina número de marco de página. Así se habla del marco de página j o marco j de memoria principal.

Por otra parte, si Cx es el tamaño del espacio de direcciones lógicas de un proceso, el número Np de páginas de tamaño Sp en que se descompone el espacio de proceso se obtiene de la siguiente forma:

$$Np = ceil\left(\frac{Cx}{Sp}\right)$$

6.24. Enumerar y describir brevemente las estructuras de datos que utiliza el sistema operativo para implementar la paginación.

- ***Tablas de páginas.*** Cada proceso tiene asignada una tabla de páginas. Cada entrada i de la tabla de páginas de un proceso X contiene, entre otras informaciones, el marco j donde se encuentra almacena la página i del proceso X y los permisos de acceso a la página. Además puede contener información sobre la ubicación de la copia de la página en

memoria secundaria, aunque a veces esta información se implementa en una tabla diferente.

- ***Tabla de marcos de página.*** Esta tabla tiene tantas entradas como marcos de página tiene la memoria principal. Cada entrada j de la tabla de marcos de página contiene, entre otras, las siguientes informaciones relativas al marco: su estado, es decir, si está libre u ocupado, punteros para crear una lista de marcos libres, y la ubicación en memoria secundaria de la página i contenida en el marco.

- ***Lista de marcos libres.*** Es consultada cuando hay que asignar espacio a los procesos que deben ser cargados en memoria principal. Cuando hay que asignar un marco se selecciona al primero de la lista. Si un marco queda libre es colocado al final de la lista.

6.25. ¿Qué es la paginación simple?

La técnica de paginación de memoria puede implementarse de diferentes formas. En su implementación más sencilla, denominada paginación simple, para que un proceso X pueda ser ejecutado es necesario que todas sus páginas se encuentren cargadas en memoria principal, aunque no es necesaria que estén cargadas en marcos contiguos.

En la paginación simple cuando un proceso X de tamaño Np páginas tiene que ser cargado en memoria principal, en SO comprueba la existencia de Np marcos libres en la lista de marcos libres. Si no hay marcos suficientes la carga del proceso tendrá que ser pospuesta hasta existan Np marcos libres. Si hay marcos libres suficientes, entonces se les asigna al proceso, los elimina de la lista de marcos libres, los marca como ocupados en la tabla de marcos de página y carga en ellos las páginas del proceso. A continuación, se crea o se actualiza (si había sido intercambiado) la tabla de páginas del proceso indicando el número de marco donde está cargada cada página. Además, si la tabla de páginas no estaba ya creada, almacena en el bloque de control del proceso un puntero a la dirección física de comienzo de la tabla de páginas.

Si un proceso fuese intercambiado fuera de memoria, todas sus páginas desaparecerían de memoria, cuando volviese a ser cargado en memoria sus páginas se cargarán en otros marcos.

6.26. Describir la traducción de direcciones en paginación con un registro base.

Cuando un proceso es planificado para ser ejecutado, el SO carga en dicho registro la dirección física de comienzo de la tabla de páginas del proceso.

Cuando durante un ciclo de instrucción se referencia a una dirección lógica Dir_L, se suma el campo número de página i de la dirección lógica a Dir_{F0} para obtener la dirección física Dir_{F1} de la entrada i de la tabla de páginas. A continuación se realiza una operación de lectura a memoria principal para acceder a Dir_{F1} y obtener el marco de página j donde se encuentra almacenada la página i.

Una vez obtenido el número de marco j donde se aloja la página i ya se tiene toda la información para construir la dirección física DirF a la que equivale la dirección lógica Dir_L, ya que el campo de desplazamiento es común en ambas direcciones.

El mecanismo descrito de traducción de direcciones tiene como principal ventaja que únicamente requiere que el SO cargue el registro base en cada cambio de proceso, lo cual repercute en unos tiempos de cambio de proceso más pequeños.

Su principal inconveniente es que la traducción de una dirección lógica requiere de un acceso a la memoria principal para localizar en la tabla de páginas el marco j que contiene la página i. Este acceso introduce un retardo en la ejecución de la instrucción en curso. Por este motivo este mecanismo de traducción de direcciones se utiliza en raras ocasiones. Sí se suelen utilizar variaciones del mismo.

6.27. Describir la traducción de direcciones en paginación con un banco de registros.

En un banco de registros el SO puede cargar en él, cuando se produce un cambio de proceso, una copia de la tabla de páginas del proceso que se va a ejecutar.

De esta forma cuando se referencia a una dirección lógica Dir_L, se utiliza el campo número e página i de Dir_L para acceder a la entrada i de la copia de la tabla de páginas del proceso almacenada en el banco de registros y obtener el marco de página j donde se encuentra almacenada la página i.

Una vez obtenido el número de marco j donde se aloja la página i ya se puede construir la dirección física Dir_F a la que equivale la dirección lógica, ya que el campo desplazamiento es común en ambas direcciones.

Tiene como principal ventaja que la traducción de una dirección lógica no requiere de ningún acceso a memoria principal. Su principal inconveniente, sin embargo, es que solo se puede utilizar si se asegura que el tamaño de las tablas de páginas de los procesos nunca será mayor que el tamaño del banco de registros. Debe tenerse en cuenta que el banco de registros suele tener entre 64 y 256 registros.

Otro inconveniente de este mecanismo de traducción es que el SO, cuando se produce un cambio de proceso, tiene que copiar en el banco de registros la tabla de páginas del nuevo proceso planificado lo que aumenta el tiempo de cambio de proceso y por tanto la sobrecarga.

6.28. Describir la traducción de direcciones en paginación con un TLB.

Un TLB es una memoria caché especial de alta velocidad. Su funcionamiento es similar a la memoria caché del procesador. Cada entrada del TLB contiene una copia de una entrada de la tabla de páginas del proceso actualmente en ejecución. Conviene matizar que no es necesario copiar en una entrada del TLB todos los campos de una entrada de la tabla de páginas del proceso en ejecución sino únicamente aquellos que son necesarios para poder realizar la traducción de direcciones (número de página i y de marco j), la protección, o determinar si la página ha sido modificada. Un TLB puede tener entre 8 y 4096 entradas.

Cuando el procesador hace una referencia a una dirección lógica Dir_L, el TLB examina en paralelo el campo número de página

i de cada una de sus entradas para ver si alguno coincide con el número de página i al que hace referencia Dir_L. Si el campo nº de página i de alguna entrada del TLB, por ejemplo la entrada h, coincide con el nº de página i de la dirección lógica, se dice que se ha producido un acierto en el TLB. Si nunca coincide se dice que se ha producido un fallo en el TLB.

Si se produce un acierto se lee el campo nº de marco j de dicha entrada h para formar la dirección física Dir_F junto con el campo de desplazamiento Dir_L.

Si se produce un fallo la gestión del mismo requiere de las siguientes acciones.

- ***Una operación de lectura a memoria principal para acceder a la entrada i de la tabla de páginas del proceso y obtener el marco de página j donde se encuentra almacenada la página i***. Con dicha información ya se puede formar Dir_F.
- ***Copiar en una entrada vacía del TLB, el contenido de dicha entrada i.*** Si todas las entradas del TLB están ocupadas entonces alguna debe ser elegida para ser reemplazada de acuerdo con algún algoritmo de reemplazamiento. Por lo tanto, si en un futuro próximo se vuelve a referenciar a una dirección lógica asociada a la página i del proceso en ejecución se producirá un acierto en el TLB ya que su entrada asociada en la tabla de páginas estará cargada en una entrada del TLB.

  La gestión de un fallo en le TLB la puede realizar el hardware, el SO o ambos en colaboración. Si la gestión del fallo la realiza el hardware únicamente o en colaboración con el SO, entonces también se dispone de un registro base donde se almacena la dirección física de comienzo de la tabla de página del proceso en ejecución.

  Cuando se produce un cambio de proceso, el SO tiene que cargar el registro base. Además tiene que borrar todas las entradas del TLB, ya que la información que contiene no sirve

para el próximo proceso que se va a ejecutar. Por lo tanto en los primeros instantes de la ejecución de un proceso se producen fallos en el TLB.

6.29. ¿Qué es una tabla de páginas paginada?

La tabla de páginas de un proceso se ubica en memoria principal ocupando un rango de direcciones físicas contiguas, para facilitar su localización y acceso. Si el proceso posee un espacio de direcciones lógicas grande, entonces su tabla de páginas ocupará una cantidad no despreciable de memoria física contigua.

Una forma de tratar las tablas de páginas grandes es descomponerla también en páginas, se tiene por tanto una tabla de páginas paginada. De esta forma la tabla de páginas se fragmenta y puede ubicarse en memoria principal de forma no contigua. Así se pueden distinguir dos tipos de páginas:

- ***Páginas ordinarias.*** Contienen instrucciones y datos del espacio lógico de un proceso.
- ***Páginas de tablas de páginas*****.** Contienen entradas de una tabla de páginas.

¿Cuántos accesos a memoria requiere?

Para localizar en memoria principal a las páginas de tabla de páginas se utiliza otra tabla de páginas denominada tabla de páginas de primer nivel o primaria. A las entradas de la tabla de páginas original contenidas en una página se les denomina tabla de páginas de segundo nivel o secundaria. Por lo tanto para cada proceso existe una única tabla de primer nivel y varias de segundo nivel, tantas como páginas ocupen la tabla de páginas original del proceso.

6.30. ¿Qué es una tabla de páginas invertida?

Tiene asociada una entrada por cada marco j de memoria principal, en vez de una entrada por cada página i de un proceso. De ahí el nombre de tabla invertida. En consecuencia el número de entradas de la tabla de páginas invertida es fijo y es igual al número de marcos. No depende, por tanto, del número de procesos que haya en el SO o del número de páginas en que se divida el espacio virtual de estos procesos.

Cada entrada j de la tabla de páginas invertida contiene típicamente los siguientes campos: número de página i alojada en el marco j, identificador numérico del proceso al que pertenece dicha página y bits de control (protección, referenciada, etc).

6.31. ¿Cómo se realiza la protección en la técnica de paginación?

En la técnica de gestión de memoria mediante paginación la protección del espacio de direcciones de un proceso se implementa a nivel de página. En cada entrada i de la tabla de páginas de un proceso existe un campo protección con un tamaño de uno o varios bits que en función de su valor permiten establecer el tipo de acceso que se permiten sobre dicha página i: solo lectura, lectura y escritura, ejecución, …

Cada vez que el procesador referencia a una dirección lógica se comprueba, antes de traducir la dirección, que el tipo de acceso que se pretende realizar sobre la página i no viola los permisos establecidos por el campo de protección. Si se detecta un intento de acceso incorrecto entonces el hardware genera una excepción denominada fallo de protección. El tratamiento de dicho fallo por parte del SO, suele ocasionar la finalización del proceso y el envío de un mensaje de aviso al usuario del proceso.

Adicionalmente a este mecanismo de protección a nivel de página, suele existir también un registro límite donde el SO almacena en cada cambio de proceso el número de páginas de que consta el proceso que va a ser ejecutado. Cuando se produce una referencia a una dirección lógica, antes de examinar el campo de protección, se compara el campo número de página de la dirección lógica con el valor almacenado en el registro límite. Si el número de página es mayor que dicho valor, entonces el hardware provoca una excepción que debe ser atendida por el SO. El tratamiento de esa excepción suele producir las mismas acciones comentadas para el caso de un fallo de excepción.

6.32. ¿Cómo se implementa la compartición de páginas?

Cada vez que se ejecuta un programa el SO crea un proceso asociado a su ejecución. El espacio lógico de dicho proceso se

divide típicamente en tres regiones: código, datos y pila. Para poder ser ejecutado en un esquema de paginación simple todas las páginas en que se descompone el espacio lógico del proceso deben estar cargadas en memoria.

En entornos de tiempo compartido varios usuarios pueden estar ejecutando simultáneamente el mismo programa. Supóngase que se ejecutan concurrentemente tres instancias del mismo programa, cada instancia tendrá asociada un proceso. Esto significa que existirán en memoria tres copias idénticas de las páginas del código del programa. Si el programa posee código reentrante, es decir, que no se puede modificar, se ahorraría espacio en memoria si solo se mantuviese cargada en memoria principal una única copia de las páginas de código de dicho programa. Dichas páginas son compartidas en modo solo lectura por todos los procesos asociados a un mismo programa.

Es el SO el que detecta si el código de un programa es reentrante. En caso afirmativo solo carga en la memoria principal una copia de las páginas del código de un programa, independientemente del número de instancias de dicho programa que se estén ejecutando.

En cada entrada de la tabla de marcos existe un contador de referencias que indica el número de entradas en las tablas de páginas que están referenciando a una página física compartida. De esta forma una página solo puede ser eliminada de la memoria si el contador de referencias es cero.

6.33. Enumerar las ventajas y los inconvenientes de la paginación simple.

Ventajas:

- ***No produce fragmentación externa.***
- ***Produce una sobrecarga pequeña***
- ***No necesita intervención humana.***
- ***Permite compartir entre varios procesos un código común.***

La principal desventaja es que produce fragmentación interna. Suele suceder que se desaprovecha parte del espacio de la última página asignada al proceso.

Puede considerarse como ventaja o inconveniente que la paginación es invisible al programador.

6.34. ¿En qué consiste la segmentación?

Cada segmento tiene asignado un nombre (código principal, subrutinas, datos, pila, etc) y una longitud. El compilador compila cada segmento comenzado por la dirección lógica 0. De esta forma cada segmento tiene su propio espacio de direcciones lógicas.

Por otra parte, cuando el SO crea un proceso asociado a un determinado programa, asigna a cada segmento un identificador numérico entero positivo h. Al número h se le denomina número del segmento. Así se habla del segmento h del proceso X.

6.35. ¿Cuáles son los campos de que consta la dirección lógica de un segmento?

Una dirección lógica consta de dos campos: El número de segmento de s bits y el desplazamiento dentro del segmento de d bits. El tamaño s del campo número de segmento se obtiene a partir del número total de segmentos Ns de que consta un proceso X.

¿Cómo se puede determinar su tamaño?

Por su parte el tamaño mínimo d del campo desplazamiento dentro de un segmento se obtiene a partir del tamaño Ss del segmento expresado en unidades direccionables (usualmente palabras).

Para un proceso que conste de Ns segmentos, el campo número de segmento será del mismo tamaño para todas las direcciones lógicas. Sin embargo, el tamaño mínimo del campo desplazamiento dependerá de la longitud de segmento al que esté asociado dicha dirección lógica. Por otra parte, en la segmentación simple una dirección lógica nunca puede ser de mayor tamaño que una dirección física.

6.36. ¿Qué es la segmentación simple?

La segmentación simple es una técnica de gestión de la memoria que soporta los elementos en los que se descompone el código de un programa. Básicamente el programa es dividido por el compilador en segmentos. Cada segmento es una entidad lógica conocida por el programador asociada a una determinada estructura de datos o a un módulo, pero nunca a una mezcla de ambos.

6.137. ¿Qué información contiene una tabla de segmentos?

Esta tabla está indexada por el número de segmento, es decir, tiene una entrada por cada segmento del proceso. Cada entrada contiene la dirección física base de cada segmento y la longitud del segmento (en unidades de asignación). También contiene información relativa a los permisos de acceso al segmento (lectura, escritura, ejecución). Además puede contener información sobre ubicación de la copia del segmento en memoria secundaria, aunque a veces esta información se implementa en una tabla diferente.

6.38. Describir la traducción de direcciones en segmentación con un registro base.

En la técnica de gestión de memoria mediante segmentación, la traducción de direcciones lógicas a direcciones físicas requiere que el SO cargue en el hardware cierta información sobre la tabla de segmentos del proceso actualmente en ejecución. La información que caga depende de los componentes hardware disponibles para realizar la traducción de direcciones. Si se dispone de un registro base, entonces el SO, en cada cambio de proceso, carga en este registro la dirección física Dirf0 de comienzo de la tabla de segmentos. El SO obtiene esta dirección del bloque de control del proceso.

Cuando se hace una referencia a una dirección lógica Dirl se realizan los siguientes pasos para traducirla a una dirección física:

1. Se suman el campo número de segmento h de la dirección lógica y Dirf0 para obtener la dirección física Dirf1 de la entrada h de la tabla de segmentos.

2. Se accede a Dirf1 para leer la dirección física base y la longitud del segmento h.

3. Se compara el valor del campo desplazamiento de la dirección lógica con la longitud del segmento h. Si el desplazamiento es mayor entonces el hardware genera una excepción por violación del tamaño del segmento.

4. Si el desplazamiento es menor que la longitud del segmento h se suma a la dirección física base del segmento h el desplazamiento (Desp.) indicado en la dirección lógica para obtener la dirección física Dirf2 a la que equivale Dirl.

   Tiene como principal ventaja que únicamente requiere que el SO cargue en cada cambio de contexto un solo registro, lo cual disminuye el tiempo de cambio de proceso. Su principal inconveniente es que requiere un acceso a la memoria principal para leer en la tabla de segmentos la dirección física base y la longitud del segmento. Este acceso introduce un retardo en la ejecución de instrucciones en curso.

Para acelerar la traducción de direcciones, al igual que sucedía con la técnica de paginación, se puede utilizar un banco de registros o un TLB.

6.39. ¿Cómo se implementa la protección en la técnica de segmentación?

Se implementa a nivel de segmento. En cada entrada h de la tabla de segmentos de un proceso existe un campo protección con un tamaño de uno o varios bits que en función de su valor permiten establecer el tipo de acceso que se permiten sobre dicho segmento.

6.40. ¿Cómo se implementa la compartición de segmentos en la técnica de segmentación?

El SO mantiene una tabla con los segmentos cargados en memoria principal. Cada entrada de esta tabla contiene, entre otras informaciones relativas a un segmento cargado en memoria, un contador de referencias que mantiene el número de procesos que referencian a dicho segmento a través de alguna entrada de sus tablas de segmentos, y en consecuencia lo comparten. Dicho segmento no podrá ser eliminado de memoria principal hasta que el contador de referencia valga 0.

6.41. Enumerar las ventajas y los inconvenientes de la segmentación simple.

Ventajas:

- ***Produce una fragmentación interna despreciable.*** Cuando se asigna espacio dentro de un hueco de memoria para un segmento de un proceso, dicho espacio se asigna como un múltiplo de una determinada unidad de asignación.

  Por ello la fragmentación interna por segmento será como máximo del orden de una unidad de asignación.

- ***Soporta la visión modular que el programador posee de su programa.***

- ***Permite manejar con facilidad estructuras de datos que crecen.***

- ***Facilita la protección y compartición de las diferentes pares de un programa.***

Inconvenientes:

- ***Produce una fragmentación externa.***

- ***Aumenta la sobrecarga del sistema.***

6.42. ¿En qué consiste la técnica de segmentación con paginación?

Consiste en combinar ambas técnicas. Básicamente consiste en dividir cada segmento de un proceso en un determinado número de páginas.

Una dirección lógica se descompone en tres campos: número de segmento de s bits, número de página de p bits y desplazamiento dentro de la página de k bits. Nótese que la suma de la longitud del campo número de página y desplazamiento dentro de la página, es igual a la longitud del campo desplazamiento dentro de un segmento de una dirección lógica en segmentación.

Cuando se carga un proceso en memoria principal el SO crea la tabla de segmentos del proceso y varias tablas de páginas (una

por cada segmento). Ahora cada entrada de la tabla de segmentos no contiene la dirección física base del segmento sino la dirección física base de la tabla de páginas asociada al segmento. Además el campo longitud del segmento viene expresado en número de páginas.

Si se dispone de un registro base, entonces el SO, en cada cambio de proceso, carga en dicho registro base la dirección física Dirf0 de comienzo de la tabla de segmentos. Cuando el procesador hace una referencia a una dirección lógica Dirl, se realizan los siguientes pasos para traducirla a una dirección física:

1. Se suma el campo número de segmento h de la dirección lógica y Dirf0 para obtener la dirección física DirF1 de la entrada h de la tabla de segmentos.
2. Se accede a Dirf1 para leer la dirección física base de comienzo de la tabla de páginas (Base) y la longitud del segmento h (Longitud).
3. Se compara el valor del campo desplazamiento de Dirl con la longitud del segmento h. Si el desplazamiento es mayor entonces el hardware genera una excepción por violación del tamaño del segmento.
4. Si el desplazamiento es menor que la longitud del segmento h, entonces se suma el campo número de página de Dirl y la dirección física base de la tabla de páginas del segmento para generar la dirección física Dirf2 de la entrada i de la tabla de páginas.
5. Se accede a Dirf2 para leer el número de marco j que contiene la página i referenciada.
6. Se construye la dirección física Dirf con el número de marco j leído y con el campo desplazamiento de Dirl.

   El mecanismo descrito de traducción de direcciones realiza dos accesos a memoria principal: el primero para leer en la entrada h de la tabla de segmentos la dirección física base de la tabla de páginas y la longitud del segmento, el segundo

para leer en la entrada i de la tabla de páginas el marco de página j donde se aloja la página i. Para acelerar la traducción de direcciones se puede utilizar un banco de registros o un TLB.

Preguntas de autoevaluación tema 7

7.1. ¿Qué ventajas e inconvenientes presenta el uso de memoria virtual?

Permite aumentar el grado de multiprogramación del sistema, ya que en memoria principal caben más procesos si solo hay algunas partes de cada proceso que si sus espacios de direcciones virtuales estuviesen cargados por completo. Además permite ejecutar procesos con un espacio de direcciones virtuales de tamaño superior al tamaño de la memoria física disponible para procesos.
Otra ventaja es que reduce el número de operaciones de E/S que son necesarias para intercambiar un proceso bloqueado desde memoria principal a secundaria o viceversa.

Se debe diseñar con cuidado ya que de lo contrario el rendimiento del sistema puede verse seriamente afectado. Si un proceso está causando continuamente excepciones porque requiere cargar en memoria partes del proceso que no están cargadas, su ejecución se ralentiza enormemente y además estará provocando una fuerte sobrecarga al sistema que tendrá que ser atendido por continuas excepciones.

7.2. ¿Qué requisitos debe reunir el hardware de un computador para poder implementar la memoria virtual?

Que soporte el reinicio de las instrucciones de su repertorio. Además cuando se implementa la mv también se suele utilizar un componente hardware llamado unidad de gestión de memoria (MMU) que se encarga de realizar la traducción de una dirección virtual en una dirección física. Una MMU puede implementarse en un circuito integrado independiente o dentro del circuito integrado del procesador. Consta de registros, sumadores y comparadores. También puede tener un banco de registro o un TLB.

7.3. Describir brevemente el funcionamiento de la técnica de paginación por demanda.

Al igual que en la técnica de paginación simple divide el espacio de la memoria principal en bloques de igual tamaño