1) ¿Cuáles son las claves más importantes de los esquemas de paginación y segmentación sencilla?

Las claves más importantes en los esquemas de paginación y segmentación sencillas son:

- Todas las referencias a la memoria dentro un proceso se realizan a direcciones lógicas, que se traducen dinámicamente en direcciones físicas durante la ejecución. Esto significa que un proceso puede ser llevado y traído a memoria de forma que ocupe diferentes regiones de la memoria principal en distintos instantes de tiempo durante su ejecución.

- Un proceso puede dividirse en varias porciones (páginas o segmentos) y estas porciones no tienen que estar localizadas en la memoria de forma contigua durante la ejecución. La combinación de la traducción de direcciones dinámicas en ejecución y el uso de una tabla de páginas o segmentos lo permite.

2) ¿Qué es el conjunto residente del proceso? ¿Qué beneficios trae gestionar la carga de procesos de esta manera?

Se denomina conjunto residente del proceso a las porciones que incluyen a la porción inicial del programa y la porción inicial de datos sobre la cual acceden las primeras instrucciones. Algunos de sus beneficios más interesantes son:

- Se pueden mantener un mayor número de procesos en memoria principal. Esto es posible debido a que sólo vamos a cargar algunas de las porciones de los procesos a ejecutar, existe espacio para más procesos.
- Un proceso puede ser mayor que toda la memoria principal.

3) ¿Para qué se utiliza la región de swap? ¿qué es el trashing?

La región Swap se utiliza para guardar las imágenes de los procesos que no se mantienen en la memoria principal.

El Trashing se da cuando el sistema consume la mayor parte del tiempo enviando y trayendo porciones de swap en lugar de ejecutar instrucciones

4) ¿Qué es el principio de proximidad?

El principio de proximidad indica que las referencias al programa y a los datos dentro de un proceso tienden a agruparse. Esto se resume que sólo unas pocas porciones del proceso se necesitarán a lo largo de un periodo de tiempo corto. También, es posible hacer suposiciones inteligentes sobre cuáles son las porciones del proceso que se necesitarán en un futuro próximo, para evitar este trasiego.

5) **En un esquema de memoria virtual con paginación, ¿Cómo es la estructura de una tabla de páginas? ¿Para qué sirven los bits de control P y M?**

En un esquema de memoria virtual cada proceso dispone de su propia tabla de páginas, y todas las páginas se encuentran localizadas en la memoria principal. Cada entrada en la tabla de páginas consiste en un número de marco de la correspondiente página en la memoria principal.

- El bit de control P sirve para indicar si la página está o no en la memoria principal.
- El bit de control M sirve para indicar si la página ha sido modificada desde la última vez que fue cargada a la memoria principal.

6) ¿Dónde y cómo se almacena la tabla de páginas?

Debido a que la tabla de páginas es de longitud variable dependiendo del tamaño del proceso, no podemos suponer que se encuentra almacenada en los registros. En lugar de eso, debe encontrarse en la memoria principal para poder ser accedida. Cuando un proceso en particular se encuentra ejecutando, un registro contiene la dirección de comienzo de la tabla de páginas para dicho proceso. El número de página de la dirección virtual se utiliza para indexar esa tabla y buscar el correspondiente marco de página. Éste, combinado con la parte de desplazamiento de la dirección virtual genera la dirección real deseada.

7) ¿Qué es la paginación multinivel? ¿Por qué es útil?

La paginación multinivel habla de cuando un proceso está en ejecución, al menos parte de su tabla de páginas debe encontrarse en memoria, incluyendo la entrada de tabla de páginas de la página actualmente en ejecución. Algunos procesadores utilizan un esquema de dos niveles para organizar las tablas de páginas de gran tamaño.

En este esquema, existe un directorio de páginas, en el cual cada entrada apuntaba a una tabla de páginas. De esta forma, si la extensión del directorio de páginas es X, y si la longitud máxima de una tabla de páginas es Y, entonces un proceso consistirá en hasta X ≥ Y páginas. Normalmente, la longitud máxima de la tabla de páginas se restringe para que sea igual a una página.

Esta estrategia resulta útil cuando la cantidad de memoria demandada por las tablas de página es increíblemente grande.

8) ¿Cómo funcionan las tablas de páginas invertidas?

Las tablas de páginas invertidas, la parte que corresponde al número de página de la dirección virtual se referencia por medio de un valor hash usando una función hash sencilla. El valor hash es un puntero para la tabla de páginas invertida, que contiene las entradas de tablas de página. Hay una entrada en la tabla de páginas invertida por cada marco de página real en lugar de uno por cada página virtual. De esta forma, lo único que se requiere para estas tablas de página siempre es una proporción fija de la memoria real, independientemente del número de procesos o de las páginas virtuales soportadas.

Debido a que más de una dirección virtual puede traducirse en la misma entrada de la tabla hash, una técnica de encadenamiento se utiliza para gestionar el desbordamiento. Las técnicas de hashing proporcionan habitualmente cadenas que no son excesivamente largas (entre una y dos entradas). La estructura de la tabla de páginas se denomina invertida debido a que se indexan sus entradas de la tabla de páginas por el número de marco en lugar de por el número de página virtual.

9) ¿Qué es la TLB? ¿Cuál es su utilidad? ¿Cómo funciona?

El TLB (translation lookaside buffer - buffer de traducción anticipada) es una caché especial de alta velocidad para las entradas de la tabla de página. Sirve para minimizar el tiempo de acceso a la memoria. Esta caché funciona de forma similar a una memoria caché general y contiene aquellas entradas de la tabla de páginas que han sido usadas de forma más reciente.

Dada una dirección virtual, el procesador primero examina la TLB, si la entrada de la tabla de páginas solicitada está presente (acierto en TLB), entonces se recupera el número de marco y se construye la dirección real. Si la entrada de la tabla de páginas solicitada no se encuentra (fallo en la TLB), el procesador utiliza el número de página para indexar la tabla de páginas del proceso y examinar la correspondiente entrada de la tabla de páginas. Si el bit de presente está puesto a 1, entonces la página se encuentra en memoria principal, y el procesador puede recuperar el número de marco desde la entrada de la tabla de páginas para construir la dirección real. El procesador también autorizará la TLB para incluir esta nueva entrada de tabla de páginas. Finalmente, si el bit presente no está puesto a 1, entonces la página solicitada no se encuentra en la memoria principal y se produce un fallo de acceso memoria, llamado fallo de página.

10) ¿Qué implicancias tiene el esquema de segmentación de memoria?

Cuando tenemos que definir el tamaño de una página hay varios factores a considerar.

Por un lado, está la fragmentación interna evidentemente, cuanto mayor es el tamaño de la página, menor cantidad de fragmentación interna. Para optimizar el uso de la memoria principal, sería beneficioso reducir la fragmentación interna.

Por otro lado, cuanto menor es la página, mayor número de páginas son necesarias para cada proceso. Un mayor número de páginas por proceso significa también mayores tablas de páginas.

Para programas grandes en un entorno altamente multiprogramado, esto significa que determinadas partes de las tablas de página de los procesos

activos deben encontrarse en la memoria virtual, no en la memoria principal. Otro factor importante son las características físicas de la mayoría de los dispositivos de la memoria secundaria, que son de tipo giratorio, favoreciendo tamaños de página grandes para mejorar la eficiencia de transferencia de bloques de datos.

11) ¿Qué implicancias tiene el esquema de segmentación de memoria?

La segmentación permite al programador ver la memoria como si se tratase de diferentes espacios de direcciones o segmentos. Los segmentos pueden ser de tamaños diferentes, en realidad de tamaño dinámico. Una referencia a la memoria consiste en un formato de dirección del tipo (número de segmento, desplazamiento).

12) ¿Cómo se implementa?

Lo habitual es que haya una única tabla de segmentos por cada uno de los procesos. En este caso, las entradas en la tabla de segmentos son un poco más complejas. Debido a que sólo algunos de los segmentos del proceso pueden encontrarse en la memoria principal, se necesita un bit en cada entrada de la tabla de segmentos para indicar si el correspondiente segmento se encuentra presente en la memoria principal o no. Si indica que el segmento está en memoria, la entrada también debe incluir la dirección de comienzo y la longitud del mismo.

13) ¿Qué es la Segmentación paginada?

En un sistema combinado de paginación/segmentación, el espacio de direcciones del usuario se divide en un número de segmentos, a discreción del programador. Cada segmento es, por su parte, dividido en un número de páginas de tamaño fijo, que son del tamaño de los marcos de la memoria principal. Si un segmento tiene longitud inferior a una página, el segmento ocupará únicamente una página.

Desde el punto de vista del programador, una dirección lógica sigue conteniendo un número de segmento y un desplazamiento dentro de dicho segmento. Desde el punto de vista del sistema, el desplazamiento dentro del segmento es visto como un número de página y un desplazamiento dentro de la página incluida en el segmento.

14) ¿Qué diferencias hay entre paginación bajo demanda y prepaginación?

La diferencia entre paginación de baja demanda y la prepaginacion es que la primera trae una página cuando se hace referencia a una posición a memoria de dicha página en cambio la otra trae también otras páginas, diferentes de la que ha causado el fallo de página. La paginación adelantada tiene en cuenta las características que tienen la mayoría de dispositivos de memoria secundaria, tales como los discos, que tienen tiempos de búsqueda y latencia de rotación.

No es necesaria ya que para los sistemas que usan paginación pura o paginación combinada con segmentación, la ubicación es habitualmente irrelevante debido a que el hardware de traducción de direcciones y el hardware de acceso a la memoria principal pueden realizar sus funciones en cualquier combinación de página-marco con la misma eficiencia.

16) ¿Por qué es necesario el bloqueo de marcos?

Es necesario mencionar una restricción que se aplica a las políticas de reemplazo antes de indagar en los diferentes algoritmos: algunos marcos de la memoria principal pueden encontrarse bloqueados. Cuando un marco está bloqueado, la página actualmente almacenada en dicho marco no puede reemplazarse. Gran parte del núcleo del sistema operativo se almacena en marcos que están bloqueados, así como otras estructuras de control claves.

17) Compare los algoritmos básicos de reemplazo.

Comparación de los algoritmos básicos de reemplazo.

Óptimo: tomará como reemplazo la página para la cual el instante de la siguiente referencia se encuentra más lejos. Esto es imposible de implementar, porque requiere que el sistema operativo tenga un perfecto conocimiento de los eventos futuros. Sin embargo, se utiliza como un estándar a partir del cual contrastar algoritmos reales.

LRU (least recently used- Usado menos recientemente): seleccionará como candidata la página de memoria que no se haya referenciado desde hace más tiempo. Debido al principio de proximidad referenciada, esta página sería la que tiene menos probabilidad de volverá a tener referencias en un futuro próximo. Y, de hecho, la política LRU proporciona unos resultados casi tan buenos como la política óptima. El problema con esta alternativa es la dificultad en su implementación.

FIFO (first-in-first-out – primero entrar primero salir): trata los marcos de página ocupados como si se tratase de un buffer circular, y las páginas se reemplazan mediante una estrategia cíclica de tipo round-robin. Todo lo que se necesita es un puntero que recorra de forma circular los marcos de página del proceso. Por tanto, se trata de una de las políticas de reemplazo más sencilla de implementar.

Reloj: requiere la inclusión de un bit adicional en cada uno de los marcos de página, denominado bit de usado. Cuando una página se trae por primera vez a la memoria, el bit de usado de dicho marco se pone a 1. En cualquier momento que la página vuelva a utilizarse (después de la referencia generada con el fallo de página inicial) su bit de usado se pone a 1. Para el algoritmo de reemplazo de páginas, el conjunto de todas las páginas que son candidatas para reemplazo (de este proceso: ámbito local; toda la memoria principal: ámbito global) se disponen como si se tratase de un buffer circular, al cual se asocia un puntero. Cuando se reemplaza una página, el puntero indica el siguiente marco del buffer justo después del marco que acaba de actualizarse. Cuando llega el momento de reemplazar una página, el sistema operativo recorre el buffer para encontrar un marco con su bit de usado a 0. Cada vez

que encuentra un marco con el bit de usado a 1, se reinicia este bit a 0 y se continúa. Si alguno de los marcos del buffer tiene el bit de usado a 0 al comienzo de este proceso, el primero de estos marcos que se encuentre se seleccionará para reemplazo. Si todos los marcos tienen el bit a 1, el puntero va a completar un ciclo completo a lo largo del buffer, poniendo todos los bits de usado a 0, parándose en la posición original, reemplazando la página en dicho marco. Véase que esta política es similar a FIFO, excepto que, en la política del reloj, el algoritmo saltará todo marco con el bit de usado a 1. La política se denomina política del reloj debido a que se pueden visualizar los marcos de página como si estuviesen distribuidos a lo largo del círculo.