**Los programas que emplean direcciones relativas a memoria son leídos utilizando carga dinámica en tiempo de ejecución.**

Verdadero

321. Stallings.

“Los programas que emplean direcciones relativas de memoria se cargan utilizando carga dinámica en tiempo de ejecución. Normalmente, todas las referencias de memoria de los procesos cargados son relativas al origen del programa. Por tanto, se necesita un mecanismo hardware para traducir las direcciones relativas a direcciones físicas de la memoria principal, en tiempo de ejecución de la instrucción que contiene dicha referencia.”

**El espacio de direcciones físicas y el espacio de direcciones lógicas, en un esquema de reasignación de direcciones en tiempo de ejecución no difieren.**

Falso.

The compile-time and load-time address-binding methods generate identical logical and physical addresses. However, the execution-time address-binding scheme results in differing logical and physical addresses.

Silberschatz 9th 355

**Si se trabaja con particiones fijas para la administración de la memoria central, el registro base y el registro límite son parte del contexto del proceso.**

Verdadero, es parte del contexto del proceso ya que definen la región de memoria ocupada por el proceso. Puede haber particiones fijas de distintos tamaños, es necesario marcar el límite de las mismas.

70. Stallings.

Los registros base y límite definen la región de memoria ocupada por el proceso: el registro base contiene la dirección inicial de la región de memoria y el registro límite el tamaño de la región (en bytes o palabras). El contador de programa y todas las referencias de datos se interpretan de forma relativa al registro base y no deben exceder el valor almacenado en el registro límite. Esto previene la interferencia entre los procesos.

321. Stallings.

Cuando un proceso se asigna al estado Ejecutando, un registro especial del procesador, algunas veces llamado registro base, carga la dirección inicial del programa en la memoria principal. Existe también un registro «valla», que indica el final de la ubicación del programa; estos valores se establecen cuando el programa se carga en la memoria o cuando la imagen del proceso se lleva a la memoria cada imagen del proceso está aislada mediante los contenidos de los registros base y valla. Además, evita accesos no autorizados por parte de otros procesos.

**La compactación resuelve el problema de la fragmentación interna pero a su vez genera conflictos de performance debido a la pérdida de ciclos de procesamiento invertidos en el proceso mismo de compactación.**

Falso.

One technique for overcoming external fragmentation is compaction.

Stallings 8th 314

**Como en muchas cosas se gana y se pierde: compactando la fragmentación interna, se gana espacio, pero se pierde tiempo.**

Falso.

One technique for overcoming external fragmentation is compaction.

Stallings 8th 314

**Es posible implementar el proceso de compactación para evitar la fragmentación externa si y solo si el método de traducción de las direcciones lógicas en físicas es absoluto.**

Falso.  
La compactación requiere la capacidad de reubicación dinámica. Es decir, debe ser posible mover un programa desde una región a otra en la memoria principal sin invalidar referencias de cada programa. [Stallings, 5ª ed. Esp., Pg. 316].

Esto es imposible si las referencias de la memoria se limitan a direcciones absolutas (O sea direccionamiento absoluto no es lo mismo que reubicación dinámica)

**La compactación para solucionar la fragmentación externa combinada con direccionamiento absoluto es una alternativa posible, pero la desventaja es que desperdicia tiempo de CPU.**

Falso, en direccionamiento absoluto no se pueden mover los segmentos, por lo tanto no se puede compactar.

**En el particionamiento dinámico, utilizando cualquiera de los algoritmos first-fit, next fit, best-fit o worst-fit, el SO mantiene una lista con las particiones libres.**

Verdadero.

This procedure is a particular instance of the general dynamic storage allocation problem, which concerns how to satisfy a request of size n from a list of free holes.

Silberschatz 9th 362

**En relación con el concepto de reubicación y la gestión de memoria es cierto que la reubicación dinámica sólo es necesaria cuando se hace compactación de memoria.**

Falso. Por ejemplo, si se trabaja con particionamiento dinámico también es necesaria la reubicación dinámica. Un proceso puede ocupar diferentes particiones a lo largo de su ciclo de vida. Por tanto, las ubicaciones (de las instrucciones y los datos) referenciadas por un proceso no son fijas. [Stallings, 5ª ed. Esp., Pgs. 319-321].

**La política de limpieza (cleaning policy) en la administración de Memoria Central (M.C.) es utilizada solo para grabar en disco, las páginas que fueron modificadas en M.C.**

Verdadera. La política de limpieza es la opuesta a la de recuperación, se encarga de determinar cuándo una página que está modificada se debe escribir en disco. Las dos opciones son bajo demanda (la página modificada se escribe cuando se selecciona para ser reemplazada) o limpieza adelantada (se escriben las páginas modificadas antes que sus marcos se necesiten, así se pueden escribir en lotes). La mejor estrategia incorpora el buffering de páginas. [Stallings, 5ª ed. Esp., Pg. 376].

**La ventaja de la memoria paginada es que al poseer fragmentación interna siempre el desperdicio de memoria por proceso va a ser menor al tamaño de una página.**

Verdadero. El desperdicio de memoria por proceso va a ser menor al tamaño de una página porque sería solamente la última página del proceso.  
  
**Una estrategia interesante que puede mejorar el rendimiento de la paginación y permitir el reemplazo de páginas más sencilla es el almacenamiento intermedio de páginas.**

Verdadero.

An interesting strategy that can improve paging performance and allow the use of a simpler page replacement policy is page buffering.

Stallings 8th 369

**El método de alocación contigua, en comparación con la paginación, previene la fragmentación.**

Falso, la paginación tiene fragmentación interna pero particiones dinámicas tiene fragmentación externa.

**Para permitir un mayor grado de multiprogramación, es mejor optar por páginas de un tamaño pequeño.**

Verdadero. En caso de utilización de un esquema de memoria virtual, páginas de menor tamaño permitirán que una menor parte de cada proceso esté en memoria, por lo tanto podrá haber más procesos en memoria y así un mayor grado de multiprogramación.   
  
**Al implementar paginación y segmentación cada proceso determina al inicio de su ejecución qué modalidad implementa.**  
Falso, Al elegir un esquema, todos los procesos se encuentran sujetos a éste. Los procesos no pueden decidir la modalidad que implementará.  
  
**La paginación produce fragmentación interna y ofrece ventajas en relación a la protección de memoria.**

Verdadero porque el espacio que se desperdicia en cada página es fragmentación interna.

**La protección de memoria en un entorno paginado se consigue mediante una serie de bits de protección asociados con cada marco. Normalmente, estos bits se mantienen en la tabla de páginas. Uno de los bits puede definir una página como de lectura-escritura o de sólo lectura**

Silberschatz, 261.

**En la paginación al igual que en particiones variables siempre los procesos ocupan un espacio de memoria que es potencia de 2.**   
Falso, en particiones variables las particiones son de longitud y número variable. Cuando se lleva un proceso a la memoria principal, se le asigna exactamente tanta memoria como requiera y no más.

**315.** Stallings**.**

**En los sistemas operativos actuales, la traducción de direcciones lógicas a físicas es una tarea exclusiva del módulo de gestión de memoria del sistema operativo.**

Falso.

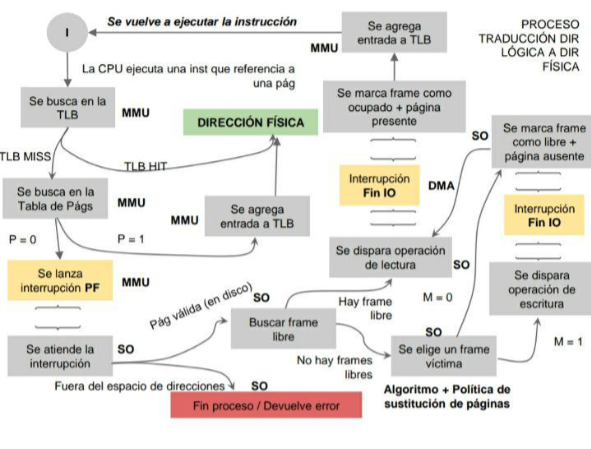
Address translation hardware (memory management unit) is interposed between the processor and memory.

Stallings 8th 68

**Cuando un proceso referencia a una página, el sistema operativo primero valida que la misma corresponda al espacio de direcciones del proceso, y si la página es inválida finaliza el proceso.**

SEGUN ESTE GRAFICO LA “CPU” GENERA LA DL, BUSCA EN LA TLB LA PAGINA CORRESPONDIENTE A LA DL, SI NO SE ENCUENTRA, BUSCA EN LA TABLA DE PAGINAS, SI NO SE ENCUENTRA LA MMU GENERA UN PAGE FAULT(INTERRUPCION), EL SO ATIENDE LA INTERRUPCION Y VERIFICA QUE LA DIRECCION SEA VALIDA (corresponda al espacio de direcciones del proceso), y si la página es inválida finaliza el proceso.

FALSO



**Dado que la TLB contiene como entradas las últimas direcciones lógicas referenciadas, se reduce la frecuencia de acceso a la memoria principal para recuperar las entradas de páginas.**

Falso.

Instead, each entry in the TLB must include the page number as well as the complete page table entry.

Stallings 8th 351

**En la TLB es posible identificar a qué proceso pertenece cada página.**

Verdadero.

Some TLBs store address-space identifiers (ASIDs) in each TLB entry. An ASID uniquely identifies each process and is used to provide address-space protection for that process.

Silberschatz 9th 374

**El tamaño de la TLB (Translation Lookaside Buffer) es proporcional al tamaño de la tabla de páginas de los procesos.**

Falso, la TLB tiene un tamaño limitado.

**Al igual que la tabla de páginas, la TLB (Translation Lookaside Buffer) puede estar parcialmente cargada en la memoria principal y las partes restantes ubicadas en memoria virtual.**

Falso, la TLB es una caché implementada en hardware que contiene algunas de las entradas de la tabla de páginas.

**Una de las ventajas del uso de TLB (Translation Lookaside Buffer) en la paginación es que se reduce el número de accesos a memoria.**  
Verdadero, si se produce un acierto en TLB no es necesario buscar en tablas de páginas (que está en memoria).

**La TLB (Translation Lookaside Buffer) sigue siendo útil con paginación simple (sin memoria virtual)**

Verdadero porque acceder a la TLB es más rápido que acceder a la memoria principal.

**La paginación por demanda no puede funcionar sin TLB**

Falso, la TLB es una caché que sirve para aumentar la performance pero no es imprescindible para el funcionamiento de la paginación por demanda.  
  
**Si se trabaja con paginación por demanda, cuando se produce un fallo de página se cambia de proceso, la TLB se salva en el PCB junto con los registros generales del procesador y luego la misma se limpia.**

Falso.

If the TLB does not support separate ASIDs, then every time a new page table is selected (for instance, with each context switch), the TLB must be flushed (or erased) to ensure that the next executing process does not use the wrong translation information. Otherwise, the TLB could include old entries that contain valid virtual addresses but have incorrect or invalid physical addresses left over from the previous process.

Silberschatz 9th 374

**La prepaginación reduce el número de page faults.**

Falso, ya que se pueden traer paginas anticipadamente que luego no sean referenciadas por el proceso, por lo que habrá que sustituirlas provocando page fault.

**En un sistema de administración de memoria que usa Tabla de Páginas Invertida, en el proceso de traducción de direcciones es necesario hacer una búsqueda dentro de la tabla en la que interviene el PID del proceso y el número de página en la que se encuentra la dirección que se pretende traducir.**

Verdadera.

Each inverted page-table entry is a pair where the process-id assumes the role of the address-space identifier. When a memory reference occurs, part of the virtual address, consisting of , is presented to the memory subsystem. The inverted page table is then searched for a match. If a match is found—say, at entry i—then the physical address is generated. If no match is found, then an illegal address access has been attempted.

Silberschatz 9th 382

**Una de las ventajas que presenta el esquema de segmentación para los programas de usuario es que los segmentos, que adolecen de restricciones en cuanto a tamaño y cantidad, no presentan fragmentación interna.**

Falso, el tamaño es variable pero tiene a un máximo dado por la longitud del registro límite.

**Cuando se trabaja con segmentación paginada la memoria principal se encuentra dividida en segmentos y dichos segmentos a su vez se dividen en frames.**

Falso, la memoria principal se encuentra dividida en frames, los procesos están divididos en segmentos y estos divididos en páginas.

**En un Sistema Operativo que no utiliza memoria virtual no se implementa ningún planificador de mediano plazo.**

Falso, puede implementarse un planificador de mediano plazo sin memoria virtual, en este caso los procesos deberían intercambiarse entre memoria y disco en forma completa.

**Lo bueno de la memoria virtual es que no se necesitan usar direcciones absolutas, para traducir las direcciones.**

Verdadero. Produce las direcciones reales (absolutas) a partir de las virtuales

**El uso de memoria virtual permite siempre ejecutar un mayor número de procesos simultáneamente de manera más eficiente que si se tiene sólo memoria principal.**

Falso (dice siempre) cuando hay thrashing no se maneja de manera eficiente.

**En un Sistema Operativo de Tiempo Real puede ser contraproducente el uso de memoria virtual.**

Verdadero, ya que sistemas de tiempo real el tiempo es crítico, puede que el tiempo desperdiciado en paginar juegue en contra.

**La organización del flujo de información entre la memoria principal y la secundaria en un ambiente multiprogramado, es uno de los grandes inconvenientes a resolver por los programadores en tiempo de desarrollo.**

Falso, la MV es responsabilidad del SO, el programador de aplicaciones no se tiene que preocupar por la Memoria Virtual.

**En paginación virtual, es preferible reemplazar páginas ya modificadas, que páginas sin modificar.**

Falso, es preferible reemplazar las páginas no modificadas porque éstas no requieren ser actualizadas en disco, lo que implicaría la realización de una entrada/salida con un gasto de tiempo mayor.

**En un esquema con segmentación paginada y memoria virtual, el uso de una asignación fija con alcance global solo es preferible si la mayoría de los procesos son de gran tamaño.**

Falso, no es posible implementar asignación fija con alcance global.

**Para ciertas trazas de páginas, el algoritmo de reemplazo “Clock” podría ocasionar la anomalía de Belady.**

Verdadero. Para ciertas trazas Clock puede degenerar en FIFO.

**Si un algoritmo de sustitución de páginas es óptimo, en el sentido de que para cualquier serie de referencias y cualquier número de frames siempre da lugar al mínimo número de page faults posibles, entonces ese algoritmo no sufre de la anomalía de Belady.**

Verdadero, si para cualquier número de referencias y páginas da menor fallos de página es todo lo contrario a la definición de la anomalía de Belady que cuanto más frames se agregan más PF hay.

**Un estudio sobre políticas de reemplazo en sistemas paginados de memoria expone lo siguiente: “Al observar el efecto del número de frames disponibles para los procesos, para una cadena de referencia dada, se observa que al aplicar la política de Clock es posible que al aumentar el número de frames disponibles aumente el número de fallos de página”. Dicho estudio describe que se produce la anomalía de Belady.**

Verdadero, el algoritmo de clock puede degenerar en fifo y sufrir asi la anomalia de belady

Algoritmos de Reemplazo

**El algoritmo clock mejorado es una versión del clock que agrega un bit de modificación y esto provoca una menor cantidad de fallos de página.**

Falso, le da prioridad a sustituir páginas que no se modificaron para tener que hacer menos operaciones de E/S pero no me asegura una menor cantidad de page faults.

**El algoritmo reloj busca y desaloja páginas no recientemente utilizadas con el fin de liberar marcos de página para asignarlos a los procesos que los demandan.**

Falso. Se utiliza para reemplazar páginas, no es que libera marcos con cierta periodicidad.

**La política empleada en el algoritmo del Reloj y en general, un algoritmo de sustitución global procesa todas las páginas residentes con independencia de la identidad del propietario cuando se selecciona una víctima.**

Verdadero, se consideran todas las páginas en la memoria principal que no se encuentren bloquedas.

**Si se tiene un Sistema Operativo que utiliza el algoritmo Clock para el reemplazo de páginas, entonces una página recién ingresada (por ejemplo, recién traída de memoria virtual) no será reemplazada antes que el puntero realice dos rotaciones completas dentro del buffer circular de los frames de página candidatos.**

Verdadero porque la primera vez le va a poner el bit de uso en 1.

**El algoritmo de sustitución de páginas LFU (Least Frecuently Used – menos frecuentemente utilizada) requiere sustituir la página que tenga el valor más alto del contador.**

Falso, el que tenga el valor más bajo del contador que representa la cantidad de veces que se usó.

**En un sistema de memoria virtual las políticas de LRU y RELOJ son superiores a la de FIFO. Pero son más complejas y sufren mayor overhead que la de FIFO.**

Falso. Puede ser que en la mayoría de los casos sean superiores, pero hay casos en los que no. Inclusive en el caso de que hay misma cantidad de page faults el FIFO sería superior, por ser mas simple y generar menos overhead.

**En las políticas de gestión de memoria, la política de reemplazo hace referencia a qué información se va a desplazar cuando no caben en memoria los datos que se necesitan.**

Falso. No tiene que ver con que si cabe en memoria o no porque depende también de la gestión del conjunto residente, o sea, la asignación podría ser fija y ámbito local por ejemplo.

Thrashing

**Un sistema operativo puede detectar el thrashing mediante el análisis de la relación entre los porcentajes de utilización de la CPU y el grado de multiprogramación del sistema.**

Falso, para detectar que hay thrashing el so debe conocer tanto el % de uso del cpu, como el % de utilizacion de la unidad de paginacion (swap device)

**Si hay demasiados procesos residentes en memoria, entonces el tamaño del conjunto residente es poco adecuado ya que se producirá trashing.**

Verdadero. “Si hay demasiados procesos residentes en memoria, entonces, de media, el tamaño del conjunto residente de cada proceso será poco adecuado y se producirán frecuentes fallos de página. El resultado es trashing.” [Stallings, 5ª ed. Esp., Pg. 376].

**Cuantos más procesos hay activos en memoria, más fallos de página existen y mejor es el uso del procesador ya que aumenta la multiprogramación.**

Falso. En thrashing, hay mas fallos de pagina y el cpu disminuye su % de utilizacion porque el sistema está mas tiempo paginando que procesando

**El trashing es independiente del tamaño de los frames de memoria, solo depende de la cantidad de procesos y el tamaño de la memoria.**   
Falso, frames más grandes --> páginas más grandes --> una mayor parte de cada proceso en memoria --> menos procesos en memoria --> menor grado de multiprogramación --> menos posibilidad de trashing.

**En una política de asignación global, la hiperpaginación (trashing) se resuelve fácilmente si se aumenta el tamaño de las páginas.**  
Falso. Si el tamaño de las paginas es muy grande tambien puede ocurrir thrashing

**Incrementar la memoria principal reduce la probabilidad de que ocurra trashing.**

Verdadero. Una solución podría ser bajar el grado de multiprogramación, otra alternativa (más costosa) sería incrementar la cantidad de memoria principal.

**Si se aumenta el tamaño de la memoria de la TLB se consiguen más aciertos ya que puede contener más páginas, esto sería una manera de resolver el trashing.**

Falso. Una solución podría ser bajar el grado de multiprogramación, otra alternativa (más costosa) sería incrementar la cantidad de memoria principal.

.