

SISTEMAS OPERATIVOS

PED 2

Julio González Caballero
DNI: 77402113Y
Centro asociado: Pontevedra

PRUEBA DE EVALUACIÓN A DISTANCIA 2-1

Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

- I) (1 p) La sobrepaginación aumenta el porcentaje de uso del procesador.*
- II) (1 p) Se denomina buffering de páginas a la estrategia consistente en cargar un cierto número de páginas de un proceso antes de iniciar o continuar su ejecución.*

I) Esta afirmación es VERDADERA. La sobrepaginación se produce cuando el espacio de trabajo de un proceso crece de manera que no todas sus páginas están cargadas en la memoria. Esto provoca fallos en el acceso a las páginas que no están cargadas, por lo que el sistema tiene que preocuparse de ir a buscar la página referenciada, y cargarla en la memoria principal, con el consiguiente uso extra del procesador. Sin sobrepaginación, cuando se producen aciertos en las referencias a páginas, al estar ya cargadas en la MP su lectura es mucho más rápida y requiere menos tiempo de uso y carga del procesador, ya que éste se aprovecha más entrabado útil y no en tareas de intercambio o reemplazamiento de páginas.

II) Esta afirmación es FALSA. El buffering de páginas no consiste en la carga por adelantado de un número de páginas, sino en la reserva por adelantado de marcos libres para introducir (en una lista de páginas que son reemplazables, seleccionadas mediante un algoritmo de reemplazamiento) nuevas páginas no cargadas en memoria, ya que la sobrecarga ante un fallo de página es menor si el sistema ya sabe por adelantado dónde puede ubicar la nueva página a cargar. De hecho el buffering de páginas se denomina también reserva de marcos libres. Por lo tanto no es exactamente lo que dice el enunciado, precarga de páginas, sino preselección de páginas reemplazables.

PRUEBA DE EVALUACIÓN A DISTANCIA 2-2

(2 p) Un sistema con memoria virtual mediante demanda de páginas utiliza el algoritmo LRU para la sustitución de páginas. Un proceso genera la siguiente secuencia de referencias a páginas de memoria:

1 3 2 4 1 5 7 4 3 2 8 9 4 5 4 9 1 8 3 2

- a) Determinar cuántos fallos de página se producen cuando se dispone de 4 o 5 marcos de página para este proceso.
- b) Explicar **razonadamente** si mejoraría la tasa de fallos de página si se aumentase el número de marcos de página a N , siendo $N > 5$.

a) El algoritmo de reemplazamiento de la página usada menos recientemente selecciona para ser reemplazada aquella página del conjunto de páginas candidatas a ser reemplazadas que lleva más tiempo sin ser referenciada, es decir, sin ser utilizada.

Se puede utilizar una lista enlazada (que funciona en realidad como una pila) para guardar los números de página cargados en MP. Cuando se referencia una página, si ésta no está cargada, se añade al principio de la lista desplazando hacia el final a las demás entradas, si las hubiera. Si además la lista está llena, se elimina de ella la situada al final de la lista, es decir, se reemplaza la que lleva más tiempo sin referenciarse. Si la página referenciada sí está en la lista y por tanto cargada en memoria, su entrada en la lista pasa al principio de la misma y las que se hallaban situadas entre el principio y la posición de la referenciada, se mueven hacia el hueco que ha dejado.

Si el proceso tiene 4 marcos de página, el proceso para la secuencia de llamadas dada sería el siguiente (I indica el principio de la lista):

- Se referencia la página 1: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página. Lista: $I \rightarrow 1$.
- Se referencia la página 3: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, la otra entrada se desplaza hacia el final. Lista: $I \rightarrow 3 \rightarrow 1$.
- Se referencia la página 2: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. Lista: $I \rightarrow 2 \rightarrow 3 \rightarrow 1$.

- Se referencia la página 4: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. Lista: $I \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 1$.

- Se referencia la página 1: La página está cargada en memoria, ya que su número aparece en la lista, por lo que se produce un acierto y la página se lee directamente. La página referenciada pasa al principio de la lista y desplazando las entradas que estaban antes que ella hacia el hueco que deja. Lista: $I \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 3$.

- Se referencia la página 5: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 5 \rightarrow 1 \rightarrow 4 \rightarrow 2$.

- Se referencia la página 7: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 7 \rightarrow 5 \rightarrow 1 \rightarrow 4$.

- Se referencia la página 4: La página está cargada en memoria, ya que su número aparece en la lista, por lo que se produce un acierto y la página se lee directamente. La página referenciada pasa al principio de la lista y desplazando las entradas que estaban antes que ella hacia el hueco que deja. Lista: $I \rightarrow 4 \rightarrow 7 \rightarrow 5 \rightarrow 1$.

- Se referencia la página 3: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 3 \rightarrow 4 \rightarrow 7 \rightarrow 5$.

- Se referencia la página 2: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 7$.

- Se referencia la página 8: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 8 \rightarrow 2 \rightarrow 3 \rightarrow 4$.

- Se referencia la página 9: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 9 \rightarrow 8 \rightarrow 2 \rightarrow 3$.

- Se referencia la página 4: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 4 \rightarrow 9 \rightarrow 8 \rightarrow 2$.

- Se referencia la página 5: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 5 \rightarrow 4 \rightarrow 9 \rightarrow 8$.

- Se referencia la página 4: La página está cargada en memoria, ya que su número aparece en la lista, por lo que se produce un acierto y la página se lee directamente. La página referenciada pasa al principio de la lista y desplazando las entradas que estaban antes que ella hacia el hueco que deja. Lista: $I \rightarrow 4 \rightarrow 5 \rightarrow 9 \rightarrow 8$.

- Se referencia la página 9: La página está cargada en memoria, ya que su número aparece en la lista, por lo que se produce un acierto y la página se lee directamente. La página referenciada pasa al principio de la lista y desplazando las entradas que estaban antes que ella hacia el hueco que deja. Lista: $I \rightarrow 9 \rightarrow 4 \rightarrow 5 \rightarrow 8$.

- Se referencia la página 1: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 1 \rightarrow 9 \rightarrow 4 \rightarrow 5$.

- Se referencia la página 8: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista $I \rightarrow 8 \rightarrow 1 \rightarrow 9 \rightarrow 4$.

- Se referencia la página 3: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista $I \rightarrow 3 \rightarrow 8 \rightarrow 1 \rightarrow 9$.

- Se referencia la página 2: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista $I \rightarrow 2 \rightarrow 3 \rightarrow 8 \rightarrow 1$.

Como vemos, para este caso de asignar 4 marcos al proceso, se han producido 16 fallos de 20 llamadas (un 80%).

Repetimos el proceso para 5 marcos:

- Se referencia la página 1: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página. Lista: $I \rightarrow 1$.

- Se referencia la página 3: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, la otra entrada se desplaza hacia el final. Lista: $I \rightarrow 3 \rightarrow 1$.

- Se referencia la página 2: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. Lista: $I \rightarrow 2 \rightarrow 3 \rightarrow 1$.

- Se referencia la página 4: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. Lista: $I \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 1$.

- Se referencia la página 1: La página está cargada en memoria, ya que su número aparece en la lista, por lo que se produce un acierto y la página se lee directamente. La página referenciada pasa al principio de la lista y desplazando las entradas que estaban antes que ella hacia el hueco que deja. Lista: $I \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 3$.

- Se referencia la página 5: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. Lista: $I \rightarrow 5 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 3$.

- Se referencia la página 7: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 7 \rightarrow 5 \rightarrow 1 \rightarrow 4 \rightarrow 2$.

- Se referencia la página 4: La página está cargada en memoria, ya que su número aparece en la lista, por lo que se produce un acierto y la página se lee directamente. La página referenciada pasa al principio de la lista y desplazando las entradas que estaban antes que ella hacia el hueco que deja. Lista: $I \rightarrow 4 \rightarrow 7 \rightarrow 5 \rightarrow 1 \rightarrow 2$.

- Se referencia la página 3: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 3 \rightarrow 4 \rightarrow 7 \rightarrow 5 \rightarrow 1$.

- Se referencia la página 2: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 7 \rightarrow 5$.

- Se referencia la página 8: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 8 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 7$.

- Se referencia la página 9: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 9 \rightarrow 8 \rightarrow 2 \rightarrow 3 \rightarrow 4$.

- Se referencia la página 4: La página está cargada en memoria, ya que su número aparece en la lista, por lo que se produce un acierto y la página se lee directamente. La página referenciada pasa al principio de la lista y desplazando las entradas que estaban antes que ella hacia el hueco que deja. Lista: $I \rightarrow 4 \rightarrow 9 \rightarrow 8 \rightarrow 2 \rightarrow 3$.

- Se referencia la página 5: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 5 \rightarrow 4 \rightarrow 9 \rightarrow 8 \rightarrow 2$.

- Se referencia la página 4: La página está cargada en memoria, ya que su número aparece en la lista, por lo que se produce un acierto y la página se lee directamente. La página referenciada pasa al principio de la lista y desplazando las entradas que estaban antes que ella hacia el hueco que deja. Lista: $I \rightarrow 4 \rightarrow 5 \rightarrow 9 \rightarrow 8 \rightarrow 2$.

- Se referencia la página 9: La página está cargada en memoria, ya que su número aparece en la lista, por lo que se produce un acierto y la página se lee directamente. La página referenciada pasa al principio de la lista y desplazando las entradas que estaban antes que ella hacia el hueco que deja. Lista: $I \rightarrow 9 \rightarrow 4 \rightarrow 5 \rightarrow 8 \rightarrow 2$.

- Se referencia la página 1: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 1 \rightarrow 9 \rightarrow 4 \rightarrow 5 \rightarrow 8$.

- Se referencia la página 8: La página está cargada en memoria, ya que su número aparece en la lista, por lo que se produce un acierto y la página se lee directamente. La página referenciada pasa al principio de la lista y desplazando las

entradas que estaban antes que ella hacia el hueco que deja.
Lista: $I \rightarrow 8 \rightarrow 1 \rightarrow 9 \rightarrow 4 \rightarrow 5$.

- Se referencia la página 3: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 3 \rightarrow 8 \rightarrow 1 \rightarrow 9 \rightarrow 4$.

- Se referencia la página 2: Como no está en la lista, no está cargada en memoria y se produce un fallo. Se carga la página en memoria y se añade al principio de la lista el número de página, las otras entradas se desplazan hacia el final. La última entrada desaparece indicando que esa página ya no está cargada en memoria. Lista: $I \rightarrow 2 \rightarrow 3 \rightarrow 8 \rightarrow 1 \rightarrow 9$.

Como vemos, para este caso de asignar 5 marcos al proceso, se han producido 14 fallos de 20 llamadas (un 70%).

b) La respuesta a la pregunta del enunciado es SÍ, mejoraría (el número de fallos disminuiría). El algoritmo LRU es un algoritmo de pila y garantiza que a mayor número de marcos, la tasa de aciertos se mantendrá igual o aumentará. Esto se debe a que si aumentamos la capacidad de la lista/pila siempre tendremos en esa lista el subconjunto de páginas que se cargarían para un número menor de marcos. Como decimos, para ciertos casos puede que el número de aciertos no aumente (imaginemos una secuencia de 4 llamadas con una lista de 4 o 5 marcos y la lista vacía, en ambos casos tendríamos 4 fallos), pero por regla general aumentará, de ahí la respuesta afirmativa.

Resumiendo, para cualquier $N > 5$, el número de fallos será igual o menor que para 5 marcos.

PRUEBA DE EVALUACIÓN A DISTANCIA 2-3

(2 p) Explique **razonadamente** las funciones que realizan las capas de software de E/S del núcleo de un sistema operativo.

Las principales capas software de E/S del núcleo de un sistema operativo son tres: el subsistema de E/S, los drivers de dispositivo de E/S y los manejadores de interrupciones:

- Subsistema de E/S

Lleva a cabo todas aquellas tareas relacionadas con las operaciones que son comunes a todos los dispositivos e independientes de éstos, es decir proporciona una interfaz a la siguiente capa (drivers) compuesta por subrutinas de comunicación entre ambas capas. Estas rutinas implementan por un lado los servicios que ofrece esta capa a los drivers, y que éstos pueden invocar, y por otro, las solicitudes que se pueden hacer desde el subsistema de E/S a los drivers.

Hoy en día, esta interfaz es en general común a todos los SO, ya que esto evita el tener que modificar el software de esta capa para adaptarlo a los distintos drivers. Esto implica que la siguiente capa (drivers) debe también adaptarse a esta interfaz, para que ambas puedan comunicarse correctamente.

Algunas de las tareas que lleva a cabo este subsistema son: la asignación y liberación de dispositivos dedicados, para gestionar el uso de recursos que pueden ser requeridos por distintos procesos; el bloqueo de procesos que solicitan una operación de E/S; la planificación de la E/S, es decir, gestionar los recursos de forma equitativa para disminuir el tiempo medio de espera de los procesos, y tener en cuenta las prioridades de las solicitudes; la invocación del driver de dispositivo apropiado traduciendo mediante tablas de correspondencia las referencias a dispositivos de los procesos a la dirección de inicio del driver correspondiente; el buffering y asignación de buffers, es decir almacenar temporalmente los datos de E/S que van llegando mientras no se puedan procesar; proporcionar un tamaño de bloque uniforme a los niveles superiores de software; la gestión de los errores producidos en una operación de E/S, determinando las acciones a tomar en esos casos (manejo de excepciones).

- Drivers

Los drivers o controladores constituyen el software que se encarga de comunicar a la capa anterior con el hardware específico, es decir, traduce las llamadas de la capa anterior (para ello debe proporcionar una serie de funciones que cumplan la interfaz entre ambas capas) y las convierte en instrucciones específicas que el hardware sabe interpretar, ya que cada dispositivo está realizado físicamente de una forma distinta. Cuando una petición es atendida la siguiente debe ser atendida. La gestión de la atención a llamadas también debe ser realizada por el driver.

Otras tareas que debe realizar un driver son, por un lado, la comprobación del estado del dispositivo y esperar a su disponibilidad para realizar la tarea requerida, y por otro lado, la gestión de los errores que se puedan producir en el proceso de atención y ejecución por parte del dispositivo de una llamada, bien resolviendo dicho error por sí mismo, o bien, en caso de que esto no sea así, despachándolo a la capa superior para que sea el subsistema de E/S quién lleve a cabo las acciones oportunas.

- Manejadores de interrupciones

Cuando se genera una interrupción en una operación de E/S es el manejador de interrupciones el que determina qué hacer ante ese evento. Normalmente esto se realiza mediante una tabla de vectores que contiene entradas para cada tipo de interrupción y la dirección de inicio de la subrutina adecuada para llevar a cabo las acciones oportunas ante ese tipo de interrupción. Así cuando se produce una interrupción se consulta la tabla y se ejecuta la subrutina.

PRUEBA DE EVALUACIÓN A DISTANCIA 2-4

En un computador con una capacidad de memoria principal de 64 kibipalabras se utiliza gestión de memoria mediante segmentación. La tabla de segmentos (todos los datos numéricos están en decimal) es la siguiente:

Nº de segmento	Base	Longitud
0	0	7230
1	16384	8191
2	32768	1024
3	8192	356
4	24576	4200

Se pide:

- a) (1 p) Supuesto que una dirección lógica tiene el mismo tamaño en bits que una dirección física y que consta de los campos [nº de segmento, desplazamiento], determinar el tamaño en bits de cada uno de estos campos.
- b) (1 p) Determinar a qué direcciones físicas expresadas en decimal corresponden las siguientes direcciones lógicas expresadas en hexadecimal:
 - i) $11AE_{16}$
 - ii) 6190_{16}

Antes de resolver cada uno de los apartados, vamos a analizar la tabla de segmentos dada.

Tenemos 5 segmentos, del 0 al 4. Cada uno de ellos, para la base y longitud dadas, se ubicará en la memoria principal de la siguiente manera:

- Segmento 0: empieza en la dirección 0 y su longitud es de 7230 palabras, luego en la memoria principal ocupa las direcciones de la 0 a la 7229, ambas inclusive.
- Segmento 3: empieza en la dirección 8192 y su longitud es de 356 palabras, luego en la memoria principal ocupa las direcciones de la 8192 a la 8547 ($8192 + 356 - 1$), ambas inclusive.
- Segmento 1: empieza en la dirección 16384 y su longitud es de 8191 palabras, luego en la memoria principal ocupa las direcciones de la 16384 a la 24574 ($16384 + 8191 - 1$), ambas inclusive.
- Segmento 4: empieza en la dirección 24576 y su longitud es de 4200 palabras, luego en la memoria principal ocupa las direcciones de la 24576 a la 28775 ($24576 + 4200 - 1$), ambas inclusive.
- Segmento 2: empieza en la dirección 32768 y su longitud es de 1024 palabras, luego en la memoria principal ocupa las

direcciones de la 32768 a la 33791 ($32768 + 1024 - 1$), ambas inclusive.

Se aprecia, por un lado, que la mayor dirección de memoria ocupada es menor que el tamaño real de la memoria (64 kibipalabras o 65536 palabras), y por otro, que ninguna de las zonas de memoria ocupadas por cada segmento se solapan. Nótese que los segmentos se han descrito ordenados respecto a su dirección base para facilitar esta última comprobación, y que la última palabra de cada segmento, es menor que la primera del siguiente, con lo que los datos son coherentes.

Una vez realizada esta comprobación, podemos pasar a resolver los apartados.

a) El tamaño de los campos N° de segmento y desplazamiento de una dirección lógica ha de ser suficiente para que puedan, respectivamente, referenciar cada uno de los segmentos y la cantidad de desplazamiento en palabras. Por otro lado, la suma del tamaño en bits de los dos campos debe ser igual al tamaño de la dirección lógica. Vamos a trabajar primero con este dato.

En el enunciado se nos dice que el tamaño de la memoria principal es de 64 kibipalabras, o lo que es lo mismo:

$$64 \times 2^{10} = 64 \times 1024 = 65536 \text{ palabras}$$

Podemos calcular los bits necesarios para direccionar cada una de las 65536 palabras de la memoria:

$$2^{n_{bits}} \geq 65536 \Rightarrow n_{bits} = 16 \text{ bits}$$

Luego el tamaño de una dirección de memoria física será de 16 bits (al menos), y, según el enunciado, lo mismo para una dirección lógica.

Vamos ahora a calcular el tamaño en bits del campo N° de dirección. Será un número de bits suficiente para referenciar cada uno de los segmentos, que son 5. Luego el número de bits de este campo debe cumplir:

$$2^{n_{bits}} \geq 5$$

El valor mínimo que satisface la desigualdad es:

$$n_{bits} = 3 \text{ bits}$$

Si cogiéramos 2 bits sólo podríamos referenciar $2^2=4$ segmentos. Con 3 bits podemos referenciar hasta $2^3=8$.

Vamos a calcular ahora el tamaño mínimo del otro campo, desplazamiento. Pero antes, hay que tener en cuenta que no podemos superar el tamaño total de la dirección, calculado anteriormente y de 16 bits. Como tenemos que usar 3 para el campo Nº de segmento, nos quedan otros 13 bits para el segundo campo, luego el tamaño de éste no debería superar este valor.

Calculamos el tamaño mínimo necesario para el segundo campo de la misma forma que hicimos para el primero y teniendo en cuenta que, si observamos la tabla del enunciado, el máximo desplazamiento que se podría realizar sería dentro del segmento 1, que es el de mayor tamaño, y éste desplazamiento sería como máximo de 8191 palabras. Por lo tanto, se tiene que cumplir:

$$2^{n_{bits}} \geq 8191$$

Y tenemos:

$$n_{bits} = 13bits$$

El mínimo número de bits que cumple la desigualdad es 13, ya que $2^{13}=8192$. Este valor coincide justamente con los bits libres que nos quedaban del tamaño total de una dirección, con lo cual los valores calculados son válidos y una dirección lógica tendría la siguiente forma:



Cabe decir que los resultados obtenidos son válidos para la tabla que se da en el enunciado. Si tuviéramos más segmentos, el tamaño del primer campo podría aumentar (lo haría por primera vez para más de 8 segmentos) y para no violar la longitud total de la dirección, los segmentos deberían ser más cortos, de manera que pudieran referenciarse con menos bits en el segundo campo.

b) Para traducir las direcciones lógicas que se nos dan en este apartado, tenemos primero que descomponerlas y extraer el valor de cada uno de los dos campos. Para ello, es necesario en primer lugar pasarlas a binario para poder “leer” los bits correspondientes a cada campo:

$11AE_{16} \rightarrow 0001|0001|1010|1110_2 \rightarrow 000|1000110101110_2$
 $6190_{16} \rightarrow 0110|0001|1001|0000_2 \rightarrow 011|0000110010000_2$

Los valores anteriores en binario se han separado con | en primer lugar por dígitos hexadecimales y finalmente por campos.

Vemos que los valores binarios para el campo N° de segmento son 000 y 011, que en decimal se corresponden con los valores 0 y 3 respectivamente, luego la primera dirección lógica hace referencia al segmento 0 y la segunda al segmento 3.

Por otro lado, los valores binarios para el campo desplazamiento son 1000110101110 y 0000110010000, que en decimal se corresponden con los valores 4526 y 400 respectivamente.

Entonces tenemos:

$11AE_{16}$: segmento 0, desplazamiento 4526
 6190_{16} : segmento 3, desplazamiento 400

El segmento 0 empieza en la dirección física 0, luego un desplazamiento de 4526 palabras implica acceder a la dirección física 4526_{10} . Por lo tanto, la dirección lógica $11AE_{16}$ referencia esta dirección física. Como el desplazamiento (4526) es menor que la longitud del segmento 0 (7230), la referencia es válida.

Por otro lado, el segmento 3 empieza en la dirección física 8192, luego un desplazamiento de 400 palabras implica acceder a la dirección física $8192 + 400 = 8592_{10}$. Por lo tanto, la dirección lógica 6190_{16} referencia esta dirección física. Sin embargo, puesto que el desplazamiento (400) es mayor que la longitud del segmento 3 (356), la referencia es inválida, y el hardware generará una excepción por violación del tamaño del segmento.

PRUEBA DE EVALUACIÓN A DISTANCIA 2-5

La política de gestión de memoria de un cierto sistema es del tipo demanda de página. El tamaño de una página es de 1 KiB, el tamaño máximo de la memoria virtual es de 4 MiB y el tamaño de la memoria física es de 1 MiB. Se pide:

- a) (1 p) Determinar el tamaño de cada uno de los campos de una dirección virtual y de una dirección física.*
- b) (1 p) Determinar la capacidad mínima que debe tener la tabla de páginas del proceso de mayor tamaño que se puede ejecutar en el sistema. ¿Qué tanto por ciento de la memoria principal ocuparía dicha tabla?*

a) En la paginación por demanda no es necesario que un proceso tenga cargadas en memoria principal todas sus páginas, lo que permite la ejecución de procesos con un número de páginas mayor que el disponible en la memoria física. Sin embargo, al igual que en la paginación simple, la memoria física se divide en espacios de igual tamaño denominados marcos de página de igual tamaño que el tamaño de página (con la diferencia de que no tiene por qué contener todas las páginas de un proceso).

Sabiendo lo anterior, y el tamaño de página dado en el enunciado, podemos calcular el número de marcos en que se puede dividir y se divide la memoria física principal mediante la siguiente expresión:

$$N_{MP} = \text{floor} \left(\frac{C_{MP}}{S_p} \right)$$

Donde C_{MP} es la capacidad de la memoria principal y S_p el tamaño de página. Por lo tanto, sustituyendo los datos tenemos:

$$N_{MP} = \text{floor} \left(\frac{1\text{MiB}}{1\text{KiB}} \right) = \text{floor} \left(\frac{2^{20}}{2^{10}} \right) = \text{floor}(2^{10}) = 2^{10} = 1024$$

Con lo que ya sabemos que la memoria física se dividirá en 1024 marcos, y en ella, por lo tanto, se podrán cargar otras tantas páginas.

La memoria virtual funcionará, en cuanto a lo que nos ocupa, igual que la memoria física, sólo que, como no todas las páginas estarán cargadas en la memoria física, cierta cantidad lo estará en memoria secundaria, esperando a ser cargadas en la principal cuando se necesiten. Pero, como decimos, el modo de dividir esta memoria es análogo al de la memoria física. Por ello, si repetimos el cálculo anterior con los datos de la memoria virtual tenemos que:

$$N_{MV} = \text{floor}\left(\frac{4MiB}{1KiB}\right) = \text{floor}\left(\frac{4 \times 2^{20}}{2^{10}}\right) = \text{floor}(4 \times 2^{10}) = 4 \times 2^{10} = 4096$$

Y el número de marcos/páginas de la memoria virtual será 4096.

Las técnicas de paginación funcionan en cuanto al direccionamiento de forma similar a las de segmentación, sólo que con un tamaño de bloque constante, la página. Por lo tanto, en una dirección tendremos dos campos, un campo de direccionamiento de marco, que indica el número de marco de la memoria física o virtual, en el que se halla cierta página y al que queremos acceder, de f bits, y otro de d bits que indica el desplazamiento en palabras dentro de ese marco.

Para calcular el número de bits necesarios para direccionar N marcos (o N palabras) podemos utilizar la siguiente igualdad:

$$2^{n_{bits}} \geq N$$

Si hallamos el mínimo n_{bits} que resuelve la igualdad para $N=1024$ marcos y $N=4096$ marcos tenemos:

$$2^{n_{bits}} \geq 1024 \Rightarrow n_{bits} = 10bits$$

$$2^{n_{bits}} \geq 4096 \Rightarrow n_{bits} = 12bits$$

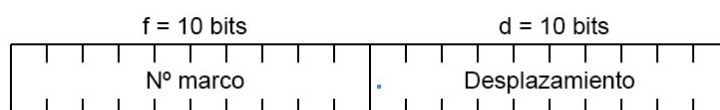
Por lo que el campo f deberá tener 10 bits en el caso de una dirección física y 12 bits en el caso de una dirección virtual.

Para hallar el tamaño del campo d , podemos usar la misma relación, teniendo en cuenta que n_{bits} debe ser suficiente para representar un desplazamiento de entre 0 y 1023 correspondiente a las palabras que tiene una página del tamaño indicado ($1Ki=1024$). Así pues:

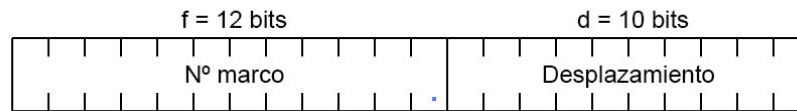
$$2^{n_{bits}} \geq 1024 \Rightarrow n_{bits} = 10bits$$

Con lo que necesitaremos 10 bits para representar el desplazamiento, y éste será el tamaño del campo d , tanto para una dirección física como una virtual, ya que el tamaño de página es el mismo para ambas memorias física y virtual.

Resumiendo, el formato de una dirección física será:



Y el formato de una dirección virtual:



Como podemos ver, el tamaño total en bits de cada dirección coincide con el número de bits necesarios para direccionar los 2^{20} Bytes (1MiB) de la memoria física y los 2^{22} Bytes (4MiB) de la memoria virtual.

b) El proceso de mayor tamaño ejecutable en el sistema será aquel que ocupe toda la memoria virtual, es decir, su tamaño será 4MiB divididos en 4096 páginas (como se calculó en el apartado a). Cada una de estas páginas puede ubicarse en uno de los 1024 marcos disponibles en la memoria física, luego cada entrada de la tabla de páginas, que también se encuentra en la memoria física, deberá tener 10 bits para indicar el marco de memoria principal en el que se encuentra cargada cierta página del proceso, más un bit de validez/invalidéz que indique si dicha página se encuentra cargada realmente en memoria principal. Es decir, cada entrada de la tabla deberá tener, como mínimo, 11 bits. Puesto que una palabra equivale a 1 Byte, y suponiendo que no podemos usar la misma palabra para distintas entradas (aunque queden bits libres), necesitaríamos 2 Bytes (16 bits > 11 bits) para cada entrada de la tabla.

Por otro lado, la tabla de páginas debe tener una entrada por cada una de las 4096 páginas en las que se divide el espacio del proceso en cuestión, y como cada una ocuparía 2 Bytes, el espacio que ocuparía la tabla en memoria principal sería:

$$2 \times 4096 = 8192 = 2^{13} \text{ Bytes}$$

Como el tamaño de la memoria principal era de 1MiB, para calcular el porcentaje que ocuparía la susodicha tabla hacemos:

$$\frac{2^{13} B}{1 \text{ MiB}} = \frac{2^{13}}{2^{20}} = 2^{-7} = 0,0078125$$

Es decir, el porcentaje ocupado es:

$$\boxed{0,78\%}$$

Cabe decir que, puesto que se ha determinado el tamaño de entrada de tabla como 11 bits, si se mezclaran entradas dentro de una misma

palabra de la memoria principal, poniendo el primer bit de cada entrada justamente a continuación del último bit de la anterior, sin forzar que coincidiese con el inicio de una palabra de la memoria, el porcentaje ocupado sería todavía menor. No obstante esto no tendría mucho sentido, ya que la unidad mínima de direccionamiento del sistema es la palabra, y acceder a un dato (entrada de la tabla) que comienza en un punto arbitrario de una palabra de memoria sería poco práctico. En cualquier caso, a continuación se muestran los cálculos para ese supuesto, a modo ilustrativo.

Tamaño total en bits de la tabla:

$$11 \times 4096 = 45056 \text{ bits}$$

Y en Bytes:

$$\frac{45056}{8} = 5632 \text{ Bytes}$$

Ratio de ocupación:

$$\frac{5632B}{1MiB} = \frac{5632B}{1048576B} = 0,005371093$$

Es decir, el porcentaje ocupado sería:

0,54%

Como dijimos, menor aunque poco práctico.