Ejercicio final de evaluación continua
Sistemas Operativos Avanzados
Parte Práctica

Ejercicio 1

Se dispone de una máquina con procesador IA32 similar al Pentium que utiliza segmentación como mecanismo de gestión de memoria. Este procesador puede direccionar hasta 4Gb y tiene cuatro registros selectores de segmento: CS, DS, SS y ES. La figura 1 muestra con detalle el formato de estos selectores. Se ha instalado en esta máquina 4G DDR-SDRAM que cubre todo el espacio de direccionamiento disponible.

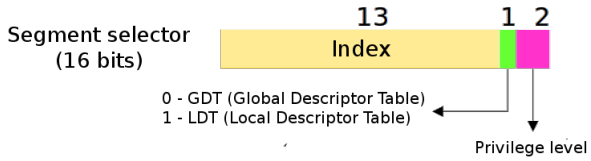

Figura 3: Registros selectores de segmento.

La figura 2 muestra el estado del mapa de memoria física en un instante determinado [2]. En dicho instante de tiempo la memoria estaba siendo ocupada por el kernel y por los segmentos de un único proceso P1.

Esta máquina dispone también de un dispositivo de acceso directo a memoria (DMA) que puede utilizarse para transferir datos entre memoria y entrada/salida, o también entre diferentes zonas de memoria.

1. ¿Cuál es el contenido de la tabla local de descriptores de segmento para el proceso P1? Indique en su respuesta el número de entrada en la tabla, la dirección base en hexadecimal y el límite relativo del segmento (tamaño) en Mb.

(1/10 puntos)

RESPUESTA:

entry#	Base	Size/Limit (Mb)
0	50000000	768
1	90000000	256
2	C8000000	512

2. Describa de forma breve qué acciones tendrían que llevarse a cabo para incrementar el tamaño del segmento de código en 256Mb más.

(2/10 puntos)

RESPUESTA: Puesto que hay 256Mb de espacio sin asignar justo al final del actual segmento de código, lo único que sería necesario hacer es modificar el campo límite de la entrada del segmento de código y establecerlo en 1024.

[2] Con objeto de facilitar la visualización y la realización de este ejercicio, en el mapa todo aparece alineado en direcciones múltiplo de 128Mb. Esto no debe confundirse con ningún tipo de mecanismo de paginación.

Mb	Start	End	
0	00000000	07FFFFFF	Kernel
128	08000000	0FFFFFFF	
256	10000000	17FFFFFF	
384	18000000	1FFFFFFF	
512	20000000	27FFFFFF	
640	28000000	2FFFFFFF	
768	30000000	37FFFFFF	
896	38000000	3FFFFFFF	
1024	40000000	47FFFFFF	
1152	48000000	4FFFFFFF	
1280	50000000	57FFFFFF	P1 Code
1408	58000000	5FFFFFFF	
1536	60000000	67FFFFFF	
1664	68000000	6FFFFFFF	
1792	70000000	77FFFFFF	
1920	78000000	7FFFFFFF	
2048	80000000	87FFFFFF	
2176	88000000	8FFFFFFF	
2304	90000000	97FFFFFF	P1 Stack
2432	98000000	9FFFFFFF	
2560	A0000000	A7FFFFFF	
2688	A8000000	AFFFFFFF	
2816	B0000000	B7FFFFFF	
2944	B8000000	BFFFFFFF	
3072	C0000000	C7FFFFFF	
3200	C8000000	CFFFFFFF	P1 Data
3328	D0000000	D7FFFFFF	
3456	D8000000	DFFFFFFF	
3584	E0000000	E7FFFFFF	
3712	E8000000	EFFFFFFF	
3840	F0000000	F7FFFFFF	
3968	F8000000	FFFFFFFF	

Figura 4: Mapa de memoria física.

3. Se solicita un nuevo segmento de 128Mb para contenidos extra (registro ES). Utilice *worst-fit* como estrategia de ubicación. ¿Cuál será el contenido de la tabla de descriptores de segmento resultante y del registro ES?

(2/10 puntos)

RESPUESTA: El bloque libre que peor se adapta comienza en `0xA0000000`. La tabla de descriptores de segmento quedaría de la siguiente forma: El valor del registro ES será 3.

entry#	Base	Size/Limit (Mb)
0	50000000	768
1	90000000	256
2	C8000000	512
3	A0000000	128

4. Se crea un nuevo proceso P2 en este sistema y solicita 512Mb para el segmento de código, 512Mb para el de datos y 512Mb para el de pila. Describa todas las operaciones que serían necesarias para ubicar las necesidades de memoria de dicho proceso (sólo las acciones relacionadas con el sistema de gestión de memoria). Dibuje un mapa de memoria similar al de la figura 2 que muestre de forma clara el mapa resultante. Incluya así mismo el estado final de las tablas de descriptores de segmento. Si necesita utilizar el DMA indique con detalle qué operaciones tendría que realizar. Para la realización de este ejercicio considere que el estado inicial de la memoria es mostrado en la figura 2.

   (5/10 puntos)

   Existen varias soluciones posibles, pero todas ellas pasan por llevar a cabo un proceso de compactación de la memoria, ya que inicialmente no existen huecos suficientemente grandes como para contener ninguno de los segmentos del nuevo proceso. Para la solución propuesta, la secuencia de compactación y las órdenes al DMA serían las siguientes:

   a) DMA: Mover desde `0x90000000`, `256Mb`, hacia `0x40000000`.

   b) Reubicación: Modificar segmento de pila de P1. Nuevo comienzo en `0x40000000`.

   c) Asignación: Nuevo segmento de código para P2, comienzo en `0x80000000`, tamaño `512Mb`.

   d) Asignación: Nuevo segmento de datos para P2, comienzo en `0xA0000000`, tamaño `512Mb`.

   e) DMA: Mover desde `0xC8000000`, `512Mb`, hacia `0xC0000000`.

   f) Reubicación: Modificar segmento de datos de P1. Nuevo comienzo en `0xC0000000`.

   g) Asignación: Nuevo segmento de pila para P2, comienzo en `0xE0000000`, tamaño `512Mb`.

   El estado final de las tablas de descriptores de segmento para P1 y P2 respectivamente quedarían de la siguiente forma:

entry#	Base	Size/Limit (Mb)
0	50000000	768
1	40000000	256
2	C0000000	512

entry#	Base	Size/Limit (Mb)
0	80000000	512
1	A0000000	512
2	E0000000	512

   Así mismo en P2, CS apuntaría al segmento 0, DS al segmento 1 y SS al segmento 2. La siguiente figura muestra el estado final de la memoria tras las operaciones de compactación realizadas.:

Mb	Start	End	Original	1st Compact.	2nd Compact.
0	00000000	07FFFFFF			
128	08000000	0FFFFFFF			
256	10000000	17FFFFFF			
384	18000000	1FFFFFFF			
512	20000000	27FFFFFF			
640	28000000	2FFFFFFF			
768	30000000	37FFFFFF			
896	38000000	3FFFFFFF	Kernel	Kernel	Kernel
1024	40000000	47FFFFFF			
1152	48000000	4FFFFFFF		P1 Stack	P1 Stack
1280	50000000	57FFFFFF			
1408	58000000	5FFFFFFF			
1536	60000000	67FFFFFF			
1664	68000000	6FFFFFFF			
1792	70000000	77FFFFFF			
1920	78000000	7FFFFFFF	P1 Code	P1 Code	P1 Code
2048	80000000	87FFFFFF			
2176	88000000	8FFFFFFF			
2304	90000000	97FFFFFF			
2432	98000000	9FFFFFFF	P1 Stack	P2 Code	P2 Code
2560	A0000000	A7FFFFFF			
2688	A8000000	AFFFFFFF			
2816	B0000000	B7FFFFFF			
2944	B8000000	BFFFFFFF		P2 Data	P2 Data
3072	C0000000	C7FFFFFF			
3200	C8000000	CFFFFFFF			
3328	D0000000	D7FFFFFF			
3456	D8000000	DFFFFFFF			P1 Data
3584	E0000000	E7FFFFFF	P1 Data	P1 Data	
3712	E8000000	EFFFFFFF			
3840	F0000000	F7FFFFFF			
3968	F8000000	FFFFFFFFFF			P2 Stack