Nº Orden	Apellido y nombre	L.U.	Cantidad de hojas

Organización del Computador 2
Segundo parcial – 12 - 11 - 2013

1 (40)	2 (40)	3 (20)	

Normas generales

- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Entregue esta hoja junto al examen, la misma **no** se incluye en la cantidad total de hojas entregadas.
- Está permitido tener los manuales y los apuntes con las listas de instrucciones en el examen. Está prohibido compartir manuales o apuntes entre alumnos durante el examen.
- Cada ejercicio debe realizarse en hojas separadas y numeradas. Debe identificarse cada hoja con nombre, apellido y LU.
- La devolución de los exámenes corregidos es personal. Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Los parciales tienen tres notas: I (Insuficiente): 0 a 59 pts, A- (Aprobado condicional): 60 a 64 pts y A (Aprobado): 65 a 100 pts. No se puede aprobar con A- ambos parciales. Los recuperatorios tienen dos notas: I: 0 a 64 pts y A: 65 a 100 pts.

Ej. 1. (40 puntos)

Se desea implementar un sistema con paginación y segmentación activas. El mismo, ejecutará código de usuario y kernel en dos niveles de protección. A continuación se definen las características generales del sistema.

El código del usuario estará contenido a partir de la dirección virtual `0x00100000`, y su tamaño será de 8kb. Además, el usuario contará con 1gb de datos que comenzará a partir de la dirección virtual `0x40000000`. El kernel tendrá su código a partir de la dirección virtual `0xC0000000` hasta la dirección `0xC01FFFFF` (inclusive), y sus datos desde la dirección virtual `0x80000000` hasta la `0xBFFFFFFF` (inclusive). Tanto kernel como usuario utilizarán mismo directorio de páginas.

En una instancia particular del sistema se encuentran mapeados los rangos `0x00100000-0x00100FFF` a `0x00000000-0x00000FFF` y `0xBFFF8000-0xC0000FFF` a `0xABCD0000-0xABCD8FFF`. Además, se cuenta con los segmentos detallados a continuación. El usuario tiene segmentos de código (indice 1 gdt) y de datos (indice 3) que se ajustan exactamente a sus respectivos tamaños en la descripción general. El kernel presenta segmentación *flat* (indices 5 y 7 para código y datos respectivamente). Los segmentos de código deben ser de sólo ejecución.

1- (3 puntos) Realizar un esquema de la memoria virtual.

2- (10 puntos) Especificar todas las entradas de las estructuras necesarias para construir el esquema de paginación según lo antes descripto. Solo indicar las entradas que están mapeadas actualmente.

3- (10 puntos) Describir cómo se completarían las entradas de la GDT del sistema. Los valores base y límite deben indicarse en hexadecimal. Justificar por que los atributos y el nivel que estableció para cada segmento son correctos.

4- (12 puntos) Resolver las siguientes direcciones, de lógica a lineal y a física. Utilizar las estructuras definidas en los ítems anteriores y suponer que cualquier otra estructura no está definida. Si se produjera un error de protección, indicar cuál error y en qué unidad (definir EPL según corresponda).

- a) `0x08:0x00000001` - CPL `11` - `lectura`
- b) `0x3B:0x00000400` - CPL `00` - `lectura`
- c) `0x0B:0x00400000` - CPL `11` - `ejecución`
- d) `0x18:0x0000800` - CPL `11` - `ejecucion`
- e) `0x38:0x00001000` - CPL `00` - `escritura`

5- (5 puntos) En el sistema dado se desea extender el segmento de datos del usuario 1 gb más. ¿Existe algún problema de seguridad? ¿En caso de que existiese, que podría cambiarse para solucionarlo? Justifique.

Recordar: `1GB = 0x40000000. 2GB = 0x80000000. 3GB=0xC0000000. 4GB-1 = FFFFFFFF.`

Ej. 2. (40 puntos)

Frigga, enojada porque Thor le rompió la nariz a su hermanastro Loki con su martillo, decide retarlo mandándolo a su cuarto con solo tres porciones de pizza. Thor, enojado, decide continuar con el desarrollo de un primitivo sistema con extrañas funcionalidades dignas de un dios.

ThorSO permite correr una única tarea denominada de prioridad "suprema", y muchas tareas esclavas. El sistema ejecutará a la única tarea suprema todo el tiempo. Mediante interrupciones, se podrán cambiar los registros de la tarea "suprema" informando de eventos. Las tareas esclavas serán despertadas cada un determinado número de *ticks* de reloj indicados por la misma tarea esclava. Las tareas esclavas utilizarán todos los *ticks* de reloj que necesiten hasta terminar su trabajo. Cuando estas terminen generarán una nueva interrupción indicando que finalizaron (int 0x66), dejando en `eax` la cantidad de *ticks* en los que debe ser llamada nuevamente la tarea.

La única interrupción externa que le interesa capturar a la tarea "suprema" es la interrupción de teclado. Cuando esta se produce, de estar corriendo la tarea "suprema" se debe modificar el registro `ecx` por 1.

Considerando que todas las tareas corren en anillo 3, incluyendo a la tarea "suprema".

1- (10 puntos) Indicar que estructuras son necesarias para administrar el sistema.

2- (10 puntos) Indicar el tipo y los campos de las entradas en la IDT de las tres interrupciones, int 0x66, reloj y teclado.

3- (20 puntos) Implementar las tres interrupciones en ASM.

Nota: Si más de una tarea esclava debe ser despertada en el mismo *tick* de reloj, no se debe considerar un orden especial.

Ej. 3. (20 puntos)

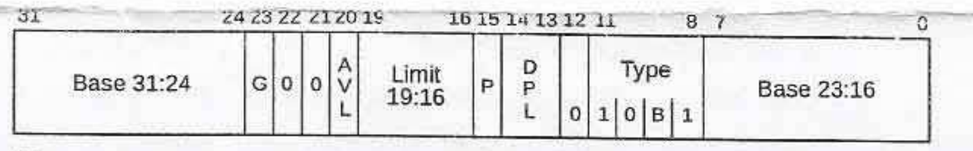
Se cuenta con un sistema denominado *gremlin* en dos niveles sobre el que se desea implementar el llamado a sistema "mojar", que se encarga de duplicar todas las estructuras de administración de la tarea llamadora, creando una nueva tarea. La diferencia entre la vieja y la nueva tarea será el valor del registro `eax`, 0 para el primer caso y 1 para el segundo.

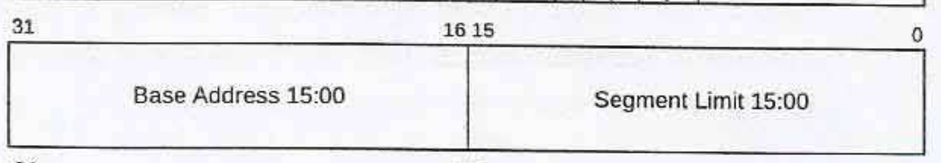
Se posee las funciones de ayuda:

- `gdt* duplicame_todo(indice_gdt)`:
  dado un indice de un descriptor de `tss` en la `gdt` se encarga de duplicar todo el estado de la tarea en cuestión y retornar la nueva entrada en la `gdt`. El `eip` de esta tarea será seteado como `0x0`.
- `indice_gdt dame_mi_indice_en_la_gdt()`:
  retorna el indice en la `gdt` de la tarea actual.

1- Completar la entrada en la `idt` de la interrupción `0x60` denominada "gremlin_mojado".

2- Escribir en ASM el código del *handler* de interrupciones pedido. Si lo requiere, explicar cualquier consideración que tenga en cuenta sobre la memoria a la hora de escribir su solución.





31 15 0		
I/O Map Base Address	Reserved | T	100
Reserved	LDT Segment Selector	96
Reserved	GS	92
Reserved	FS	88
Reserved	DS	84
Reserved	SS	80
Reserved	CS	76
Reserved	ES	72
EDI		68
ESI		64
EBP		60
ESP		56
EBX		52
EDX		48
ECX		44
EAX		40
EFLAGS		36
EIP		32
CR3 (PDBR)		28
Reserved	SS2	24
ESP2		20
Reserved	SS1	16
ESP1		12
Reserved	SS0	8
ESP0		4
Reserved	Previous Task Link	0