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

Organización del Computador 2
Segundo parcial – 17/11/16

1 (30)	2 (40)	3 (30)	

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. (30 puntos)

Se tiene la siguiente tabla GDT:

Indice	Base	Límite	DB	S	P	L	G	DPL	Tipo
30	`0x32100123`	`0x21F01`	1	1	1	0	1	11	`0xA`
31	`0x43210012`	`0x10000`	1	1	1	0	1	10	`0x8`
52	`0x22222222`	`0x40F0F`	1	1	1	0	1	00	`0x2`
73	`0x11000000`	`0x3FFFF`	1	1	1	0	1	00	`0x0`

Y el siguiente esquema de paginación:

Rango Lineal	Rango Físico	Atributos
`0x4EA13000 - 0x4EA15FFF`	`0x782AF000 - 0x782B1FFF`	read only, supervisor
`0x4ED7F000 - 0x4ED7FFFF`	`0x702AA000 - 0x702AAFFF`	read/write, supervisor
`0x4EE27000 - 0x4F17EFFF`	`0x8A0A8000 - 0x8A3FFFFF`	read only, user

(10p) a. Especificar todas las entradas de las estructuras necesarias para construir un esquema de paginación. Suponer que todas las entradas no mencionadas son nulas.

(20p) b. Resolver las siguientes direcciones, de lógica a lineal y a física. Utilizar las estructuras definidas y suponer que cualquier otra estructura no lo está. Si se produjera un error de protección, indicar cuál error y en qué unidad. Definir `EPL` en los accesos a datos. El tamaño de todas las operaciones es de 2 bytes.

I - `00F3:1C9130FF` - CPL `11` - lectura

II - `00FA:0BB6F088` - CPL `10` - ejecución

III - `01A0:2CB05000` - CPL `00` - escritura

IV - `0248:3E27FFFF` - CPL `00` - lectura

Ej. 2. (40 puntos)

Se tiene un sistema que ejecuta N tareas concurrentemente en dos niveles de privilegio. Las tareas puede decidir "cansarse de correr" en cuyo caso son desalojadas para no correr mas (se van a "dormir") e inmediatamente es ejecutada la próxima tarea. Además, puede que otra tarea requiera de los servicios de una tarea dormida. Por lo que puede llamar a "te necesito" que la despierta y la agrega al scheduler.

Servicio	Int	Parametros
`cansarse_de_correr`	0x82	Sin Parametros
`te_necesito`	0x87	EAX = Id de la tarea

Se proveen las siguientes funciones:

- `void NextTask(uint16* selector)`
  Retorna en `selector`, el selector de la tarea a ejecutar (puede ser igual a la actual).
- `void GetTaskId(uint16* id)`
  Retorna en `id`, el Id de la tarea actual.
- `void AddToSched(uint16* id)`
  Agrega la tarea identificada por `id` al scheduler.
- `void RemoveFromSched(uint16* id)`
  Borra la tarea identificada por `id` del scheduler.

(10p) 1. Detallar los campos relevantes de todas las estructuras involucradas en el sistema para administrar segmentación, paginación, tareas, interrupciones y privilegios. Instanciar las estructuras con datos y explicar su funcionamiento. Describir tanto el esquema de segmentación como el de paginación si es que lo utiliza.

(15p) 2. Programar en ASM la rutina de atención de interrupciones del reloj y los servicios del sistema `te_necesito` y `cansarse_de_correr`.

(15p) 3. Considerar ahora que si la tarea a "despertar" dejó de correr por una interrupción de reloj, entonces no solo debe ser agregada al scheduler sino también ejecutada inmediatamente. En caso contrario, solamente debe ser agregada al scheduler. Modificar el código anterior para respetar el comportamiento descripto.

Nota: Puede utilzar la siguiente función si es que la requiere:
`tss* getTss(int id)`: Dado un Id de tarea, retorna un puntero a su tss.

Ej. 3. (30 puntos)

Sea un sistema que corre concurrentemente tareas con segmentación flat y paginación activa. En este sistema se busca implementar un servicio que permita a una tarea leer un byte de la memoria de cualquier otra tarea. Para esto el servicio recibe como parámetro el Id de la tarea y la dirección de memoria virtual a leer. El servicio, además de retornar el byte leído, deberá indicar si la lectura fue correcta. Tener en cuenta que el servicio no puede ser usado para leer posiciones de memoria con atributos de supervisor.

(5p) 1. Explicar el funcionamiento del servicio, los parámetros que toma y todo lo que requiera para implementarlo en el sistema.

(10p) 2. Implementar en ASM el código del servicio, considerando que es posible modificar el valor del registro CR3.

(15p) 3. Implementar en ASM el código del servicio, considerando que NO es posible modificar el valor del registro CR3.

Nota: Se pueden hacer funciones auxiliares en C.
Considerar que se cuenta con las siguientes funciones:

- `tss* getTss(int id)`: Dado un Id de tarea, retorna un puntero a su tss.
- `void mapearPagina(uint cr3, uint virtual, uint fisica)`: Mapea en el `cr3` dado, la dirección correspondiente a `virtual` a la dirección `fisica` pasada como parámetro.