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

Organización del Computador 2
Recuperatorio del segundo parcial – 10/12/2015

1 (30)	2 (45)	3 (25)	

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
7	0x2800	0x3FFFF	1	1	1	0	1	0	0xA
18	0x5000	0x00400	1	1	1	0	0	3	0xA
21	0xC800	0x001FF	1	1	1	0	0	0	0x2
33	0x4000	0x00000	1	1	1	0	1	3	0x2

Y el siguiente esquema de paginación:

Rango Lineal	Rango Físico	Atributos
0x0000C000 a 0x0000CFFF	0x01010000 a 0x01010FFF	read/write, supervisor
0x00002000 a 0x00005FFF	0x00FFF000 a 0x01002FFF	read/write, user

(12p) a. Especificar todas las entradas de las estructuras necesarias para construir un esquema de paginación. Suponer que todas las entradas no mencionadas son nulas. Los rangos incluyen el último valor.

(18p) 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.

I - `0x0038:0x00000135` - CPL `00` - `ejecución`

II - `0x0FF1:0x60606060` - CPL `10` - `ejecución`

III - `0x0108:0x00000AAA` - CPL `00` - `lectura`

IV - `0x0090:0x00000402` - CPL `00` - `escritura`

recordar:
`4KB=0x1000, 64KB=0x10000, 1MB=0x100000, 4MB=0x400000,`
`1GB=0x40000000, 2GB=0x80000000, 3GB=0xC0000000, 4GB-1=0xFFFFFFFF.`

Ej. 2. (45 puntos)

Se desea implementar un sistema multi-tareas. El mismo cuenta con un kernel que hace correr concurrentemente a un máximo de 8 tareas de usuario independientes. Las mismas no deben poder leer ni escribir la memoria de las demás. El scheduler las ejecuta cíclicamente dandole K ciclos para correr. Además pueden lanzar más tareas mediante el syscall `fork`. Cuando un proceso pide `fork` al sistema, éste último crea una copia *idéntica* del mismo, y la agrega al scheduler. La única diferencia entre ambos será que el valor devuelto en `EAX` será 1 para el nuevo y 0 para el original. Además, el sistema deberá copiar el mapa de memoria del proceso original al proceso nuevo, de manera que sus datos sean independientes. Para tal fin copiará todas las paginas del proceso viejo que estén mapeadas en nivel usuario.

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

(10p) b. Escribir el código de la rutina de atención de interrupciones del reloj.

(10p) c. Escribir el código de una rutina `uint copiar_memoria(uint cr3)`, que se encarga de todo lo referido a la copia de memoria necesaria para el syscall `fork`. Asumir que se cuenta con un area libre del kernel de donde salen las nuevas paginas con `dame_fisica_libre()`.

(15p) d. Escribir el código de la rutina de atención del syscall `fork`.

Para resolver el ejercicio, se cuenta con algunas rutinas:

- `void mapear_pagina(uint virtual, uint cr3, uint fisica, uint attributos)`
- `int copiar_pagina(void *src, void *dst)` que copia 4kb de `src` a `dst`.
- `void* dame_fisica_libre()` que devuelve y asigna una dirección de una página libre en memoria física.

Nota: Cualquier código pedido debe estar escrito en C o ASM. Asumir que se dispone de las definiciones de las estructuras del procesador en C; que hay 1 tarea corriendo inicialmente que las tareas nunca forkearán mas de 7 veces en total. Las tareas nunca se eliminan ni generan excepciones.

Ej. 3. (25 puntos)

En un remoto rincón de alaska un programador se encuentra debugueando un código misterioso de nivel usuario. Para ayudarse, va a programar un mini sistema operativo con segmentación flat e identity mapping. Su funcionalidad será la que se describe a continuación. El código misterioso estará corriendo todo el tiempo. Cuando el usuario presione una tecla se activará el modo debug. El código misterioso seguirá corriendo hasta generar una excepción. Cuando esto suceda, si está en modo debug, el sistema llamará a una función `log_error` y de alguna manera esperará a que se presione otra vez una tecla. Finalmente, al presionar la tecla el sistema deshabilitará el modo debug y volverá a ejecutar el código misterioso, pero no desde donde estaba corriendo sino a partir de la dirección `0x401000`. Si no se estaba en modo debug el reseteo será instantáneo. Al retornar el control al código misterioso, los registros de uso general pueden tener cualquier valor.

La función `void log_error(ulong eip, ulong cs, ulong eflags)` recibe el valor de esos datos al generarse la excepción.

(5p) a. Describir la implementación de un sistema que permita implementar esta funcionalidad. Describir los valores de las estructuras del sistema que son necesarios para que el mismo funcione.

(10p) b. Escribir el código de la rutina de atención de la excepción #GP.

(10p) c. Escribir el código de la rutina de atención del teclado.