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

Organización del Computador 2
Recuperatorio del segundo parcial – 13/12/2011

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)

1. (10 puntos) Describa cómo completaría las primeras entradas de la GDT en función de los segmentos que se detallan en la siguiente tabla. Los valores base y límite deben indicarse en hexadecimal.

Índice	Desde	Hasta	Permisos	Tipo
1	0,5 Gb	1,5 Gb	`00`	datos - escritura
2	1,75 Gb	2,25 Gb	`01`	código - lectura
3	2,5 Gb	3,5 Gb	`10`	código - no lectura
4	1 Gb	3 Gb	`11`	datos - no escritura

2. (10 puntos) Especifique todas las entradas de las estructuras necesarias para construir un esquema de paginación según la siguiente tabla. Suponga que todas las entradas no mencionadas son nulas. Los rangos incluyen el último valor. Los permisos deben definirse como supervisor.

Rango virtual	Rango físico
`0xC13FE000` a `0xC1402FFF`	`0xE001D000` a `0xE0021FFF`
`0x3FFFD000` a `0x40000FFF`	`0x00000000` a `0x00003FFF`

3. (14 puntos) Resolver las siguientes direcciones, de lógica a lineal y a física. Utilizar las estructuras definidas en los ítems anteriores. Si se produjera un error de protección, indicar cuál error y en qué unidad.

   - `0x008:0x1FFFF111` - CPL `00` - `escritura`
   - `0x011:0x00011111` - CPL `00` - `lectura`
   - `0x01A:0xA1400222` - CPL `10` - `ejecución`
   - `0x023:0x00000111` - CPL `00` - `lectura`

4. (6 puntos) Se dispone de una máquina con 1Gb de memoria RAM, donde se almacenan tres tareas ádemas del código del Sistema Operativo. El código del O.S. ocupa 23Mb de memoria, mientras que para las tareas se requiere de 8621Kb, 2321kb y 4531kb respectivamente.

   - ¿Cuantas paginas de memoria se requieren mínimamente para almacenar los directorios de paginas y tablas de paginas de cada una de las tareas y el O.S.? (Justificar).

Ej. 2. (40 puntos)

Se cuenta con un sistema operativo básico al cual se le desea agregar una llamada al sistema para poder obtener la tecla presionada por el usuario. Sólo en el momento de ser presionada la tecla será

capturada por el sistema, pasando sólo el caracter ascii al programa que *lo solicite*. La interrupción se la quiere implementar a través de la **interrupt gate** número **90** (`dame_tecla_presionada`). La tarea que la llama obtiene la tecla que presionó el usuario en el registro `al`. En caso de que el usuario no haya presionado ningúna tecla, la tarea se queda a la espera de que lo haga, es decir, la interrupción no retorna hasta que haya una tecla para devolver. Para ello, se marca como 'a la espera de una tecla' y se pone a correr la próxima tarea disponible en el sistema.

1. (5 puntos) Describir la entrada en la IDT para esta nueva interrupción, la misma se debe poder acceder desde el anillo de protección 3.
2. (20 puntos) Programar en Assembler la interrupción `dame_tecla_presionada`.
3. (15 puntos) Programar en Assembler el handler de la interrupción de teclado.

   Se cuenta con las siguientes funciones y variables:

   - `task_wait_key`: Variable que indica el índice en la gdt que corresponde a la tarea a la espera por una tecla, de ser nulo no hay tarea a la espera.
   - `void add_to_scheduler(gdt_index i)`: Agrega la tarea indicada por el índice en la gdt al scheduler.
   - `void remove_from_scheduler(gdt_index i)`: Quita la tarea indicada por el índice en la gdt del scheduler.
   - `int get_key_from_buffer(char* a)`: Toma una tecla del buffer de teclado y la escribe en `a`. Si no hay tecla en el buffer retorna `0` y `1` en caso contrario.
   - `void add_key_to_buffer(char a)`: Agrega una tecla al buffer pasada por parámetro.
   - `unsigned short next_task()`: Retorna el selector de segmento de la TSS de la próxima tarea a ejecutar.
   - `char scancode_a_ascii(char scancode)`: Retorna el ascii del scancode que es pasado como parámetro.

Nota:

- Se garantiza que en ningún momento va a haber más de una tarea esperando por una tecla de manera simultánea.

Ej. 3. (20 puntos)

Se desea proveer un mecanismo por el cual un conjunto de tareas puedan compartir una fracción de su memoria, es decir, que cada una de estas tareas tengan mapeadas en alguna posición virtual de su espacio de direccionamiento la misma posición física de memoria. El mecanismo debe soportar que puedan coexistir varios conjuntos de tareas y cada uno de ellos comparta diferentes porciones de memoria.

1. (10 puntos) Proponer un mecanismo que permita compartir una página de 4kb entre las tareas que la soliciten. Detallar y justificar cada uno de los pasos. Indicar con pseudo-código que acciones debe hacer la tarea y cuales el sistema.
2. (10 puntos) Suponga que hay 3 tareas que quieren compartir una página de memoria: tarea$_X$, tarea$_Y$ y tarea$_Z$. Escriba el código ASM que debería ejecutar cada una de ellas para lograr compartir la página de memoria basándose en los mecanísmos que propuso en el punto anterior.