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

Organización del Computador 2
Recuperatorio del segundo parcial – 18/07/17

1 (20)	2 (50)	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. (20 puntos)

Responder detalladamente las siguientes preguntas, ejemplificar de ser posible.

(4p) 1. Con segmentación *flat*, ¿Es posible utilizando solamente paginación, proteger una pagina de memoria para que no sea posible ejecutar código?

(4p) 2. ¿Cuantos bytes de tamaño, tiene un segmento de límite 0 y granularidad 0?

(4p) 3. ¿Qué diferencia hay entre el bit *dirty* y el bit *accessed* en una entrada de tabla de páginas?

(4p) 4. ¿Cómo funciona un segmento *Expand Down*?

(4p) 5. ¿Qué permisos efectivos tiene una pagina si su *Page Directory Entry* es de lectura/escritura con nivel de usuario y su *Page Table Entry* es de solo lectura con nivel supervisor?

Ej. 2. (50 puntos)

En un sistema tipo con segmentación flat, se propone el esquema de paginación que muestra la figura a continuación. Cada tarea ocupa exactamente 20 KB de memoria física, que corresponden a 5 paginas.

El código de las tareas será para todas el mismo, ocupado exactamente 4KB de memoria. El mapeo a memoria virtual de cada tarea corresponderá a mapear el codigo, datos y las dos pilas según corresponda a cada tarea.

Las tareas en el sistema son ejecutadas en orden, una por cada ciclo de reloj. Inicialmente las tareas comienzan con el EIP en la primer dirección de memoria de código y todos los registros en `0xFF`, exepto el registro `EAX`, que contendrá el número de tarea que se esta ejecutando.

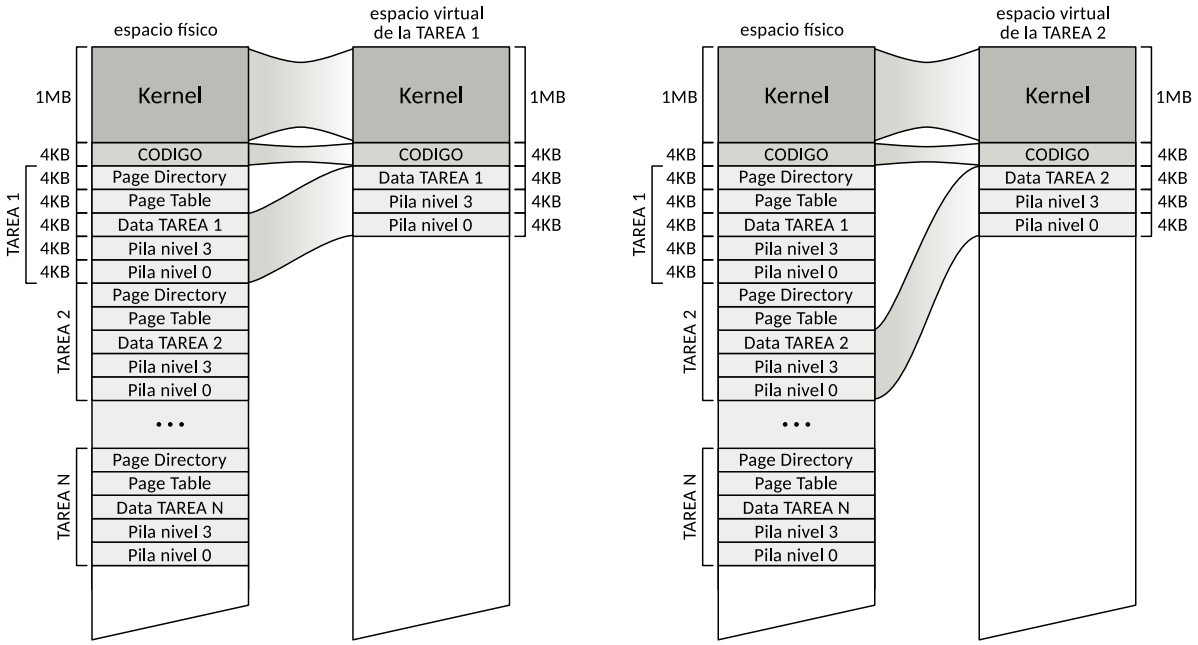
Las tareas pueden cometer cualquier tipo de excepción, en ese caso deben ser reiniciadas y comenzar a ejecutar inmediatamente como si ejecutadas por primera vez.

(10p) 1. Indicar 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.

(15p) 2. Programar en C la función `mapear_tarea`, que dado el puntero al directorio de paginas de una tarea se encarga de construir todo el mapa de paginación de la misma.

(15p) 3. Programar en ASM/C la rutina de atención de interrupciones de alguna excepción del procesador.

(10p) 4. Programar en ASM/C la rutina de atención de interrupciones del reloj.



Ej. 3. (30 puntos)

Suponer un sistema tipo con segmentación y paginación activa. Este sistema ejecuta tareas concurrentemente por cada ciclo de reloj. Las tareas pueden llamar a un servicio denominado `sendData`, que se encarga de copiar un buffer de memoria de una tarea cualquiera a otra diferente, incluso diferente de la tarea que llamo al servicio.

Los parametros de este servicio son los siguientes:

- `TASK_SRC`: nombre de la tarea fuente
- `TASK_DST`: nombre de la tarea destino
- `DIR_SRC`: dirección virtual en el espacio de la tarea fuente
- `DIR_DST`: dirección virtual en el espacio de la tarea destino
- `SIZE`: cantidad de bytes a copiar

Por cuestiones de performance, el servicio no puede modificar el registro `CR3`. Además se cuenta con la función: `uint32_t getCr3ByName(char* nombre)`, que toma el nombre de una tarea y retorna el `CR3` de la misma. El valor de `SIZE` no puede superar los 4000 bytes. Suponer que todas las tablas y directorios de página están mapeados con *identity mapping*.

(15p) 1. Programar en C el código de la función `getFisica`, que dado un `CR3` y una dirección virtual válida, optiene la dirección de memoria física donde resuelve la misma.

(15p) 2. Programar en ASM/C el código de la rutina de servicio `sendData`.

Nota: Considerar que todas las tareas tienen como espacio virtual libre desde la dirección $0x123000$ a $0x234000$.