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

Organización del Computador 2
Segundo parcial – 27/06/2013

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.
- Existen tres notas posibles para los parciales: 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. Para los recuperatorios existen sólo dos notas posibles: I: 0 a 64 pts y A: 65 a 100 pts.

Ej. 1. (40 puntos)

1. (10 puntos) Describir cómo se completarían las 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.

Indice	Desde	Tamaño	Permisos	Tipo
3	2MB	5MB	`level 0`	Código - lectura - nonconforming
6	0	2.5GB	`level 1`	Datos - lectura/escritura
9	20KB	2B	`level 2`	Código - solo ejecución - nonconforming
11	3GB	4GB	`level 1`	Datos - lectura

2. (13 puntos) Especificar todas las entradas de las estructuras necesarias para construir un esquema de paginación según la siguiente tabla. Suponer que todas las entradas no mencionadas son nulas. Los rangos incluyen el último valor. Los permisos deben definirse como usuario.

Rango Lineal	Rango físico
`0x013FE000` a `0x013FEFFF`	`0x00000000` a `0x00001FFF`
`0x013FF000` a `0x01400FFF`	`0x00100000` a `0x00102FFF`

3. (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 en todos los casos).

   - a) `0x48:0x00000001` - CPL `10` - `lectura`
   - b) `0x18:0x00400000` - CPL `00` - `lectura`
   - c) `0x5A:0x00000400` - CPL `01` - `lectura`
   - d) `0x30:0x00001000` - CPL `00` - `lectura`
   - e) `0x48:0x00000000` - CPL `00` - `ejecución`
   - f) `0x30:0x013FE321` - CPL `01` - `escritura`

4. (5 puntos) ¿Qué sucede con el item 1.3.e si el segmento al que apunta es de codigo conforming?

Ej. 2. (40 puntos)

Se cuenta con un kernel básico al que se le quiere agregar una funcionalidad para crear tareas de nivel de usuario.

Escriba una función que cree una tarea desde cero y la agregue al scheduler. La tarea creada debe tener mapeado el kernel desde la dirección `KERNEL_START` hasta la dirección `KERNEL_END` (identity mapping). El código de las tareas está preparado para ejecutarse a partir de la dirección virtual `CODE_START` y el área de memoria virtual asignada a su pila es de una página y empieza en la dirección `0xbfff0000` (cualquier otro mapeo necesario debe hacerse mediante *identity mapping*). El prototipo de la función es el siguiente:

`void crear_tarea(unsigned int code, unsigned int code_size)`: `code` indica la posición de memoria dónde se encuentra el código de la tarea a cargar (debe copiarse al área que le corresponda). `code_size` indica el tamaño en bytes del código. Se pide:

a) (5 Puntos) Indique qué estructuras de datos utilizadas por el procesador debe modificar para implementar la función pedida y qué función cumple cada una.
b) (23 puntos) Implemente la función pedida.
c) (12 puntos) Se quiere exportar esta funcionalidad para que pueda ser usada como una `syscall` (`90`) por tareas de kernel que corren en anillo 2. Implemente un handler de llamada al sistema que utilice la función creada. Además, describa la entrada en la IDT (Interrupt Gate) para llamar a la nueva `syscall`. Los parámetros de la misma se pasan por pila. Ejemplo de uso:

```
push 0x5000         ; tamaño del código
push 0x00015000     ; posición de memoria dónde se encuentra el código
int 90
```

Para llevar a cabo la implementación del item `b`, cuenta con las siguiente funciones:

- `unsigned int tss_dame_indice_libre()`: Retorna el índice de una entrada tss libre.
- `unsigned int gdt_dame_indice_libre()`: Retorna el índice de una entrada gdt libre.
- `unsigned int mmu_dame_fisica_libre_kernel()`: Retorna la dirección de una página física libre para datos de kernel.
- `unsigned int mmu_dame_fisica_libre_usuario()`: Retorna la dirección de una página física libre para datos de usuario.
- `unsigned int gdt_base()`: Retorna la dirección dónde comienza la `GDT`.
- `unsigned int tss_base()`: Retorna la dirección dónde comienza el arreglo de `TSSs`.
- `void sched_agregar_tarea(unsigned short indice_tarea)`: Agrega la tarea con índice `indice_tarea` al scheduler.
- `void mmu_mapear_pagina(unsigned int directorio, unsigned int virtual, unsigned int fisica, unsigned int attributos)`: Realiza el mapeo de las direcciones pasadas como parámetros para el directorio indicado.
- `void task_copiar_codigo(unsigned int src, unsigned int dst, unsigned int size)`: Copia el código de tamaño `size` de `src` a `dst`.

Notas:

- Las tareas creadas realizan `syscalls` a servicios de kernel que corren en anillo 2.
- Puede asumir que siempre que llame a las funciones anteriores, estás siempre retornarán un valor válido.
- **Recomendación**: Divida la implementación en funciones convenientes. Puede escribirlas en C y/o ASM según más le parezca.

Ej. 3. (20 puntos)

Para un sistema con poca memoria física, se desea implementar un administrador colaborativo de misma. En este sistema, cada página alocada puede estar *disponible* o *no disponible*. Al recibir un pedido, en el caso de no contar con espacio físico suficiente, el sistema busca una página que esté *disponible* asignada a otra tarea, se la quita y la reasigna a la tarea que hizo el pedido. Antes de usar una página, la tarea avisa al sistema que quiere usarla, para que el sistema la marque como *no disponible*. Si la página sigue en memoria el sistema habilita la utilizacion por un determinado quantum, y una vez agotado la vuelve a marcar como sin usar en cuyo caso devuelve un codigo de error indicando que fue borrada. Al alocarse se marcan como *disponibles*.

a) (6 puntos) Describir que estructuras de datos debe contener el sistema para soportar este sistema y que entradas deben agregarse a las tablas del sistema. No hace falta indicar los bits de las mismas.

b) (14 puntos) Describir que rutinas del sistema operativo deben programarse para soportar este sistema, detallando como sería su protocolo y que haría cada una.