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

# Organización del Computador 2
## Segundo parcial – 23-06-1016

| 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 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0x0A000000 | 0x00000 | 1 | 1 | 1 | 0 | 1 | 0 | 0xA |
| 2 | 0xFF000000 | 0xFFFFF | 1 | 1 | 1 | 0 | 0 | 2 | 0x8 |
| 3 | 0x00330000 | 0x00010 | 1 | 1 | 1 | 0 | 1 | 3 | 0x2 |
| 4 | 0x80000000 | 0x7FFFF | 1 | 1 | 1 | 0 | 1 | 3 | 0x0 |

Y el siguiente esquema de paginación:

| Rango Lineal | Rango Físico | Atributos |
|---|---|---|
| 0x0A000000 - 0x0A000FFF | 0x01B30000 - 0x01B30FFF | read only, supervisor |
| 0xFF000000 - 0xFFFFFFFF | 0x00000000 - 0x00FFFFFF | read/write, supervisor |
| 0x00331000 - 0x00332FFF | 0xF5001000 - 0xF5002FFF | 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.

(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. El tamaño de todas las operaciones es de 2 bytes.

I - 0x12:0x00100080 - CPL 00 - ejecución

II - 0x08:0x00000005 - CPL 00 - lectura

III - 0x18:0x00000050 - CPL 00 - escritura

IV - 0x21:0x00F55555 - CPL 02 - escritura

V - 0x20:0x7F004200 - CPL 03 - lectura

VI - 0x1B:0x00001015 - CPL 02 - escritura

## Ej. 2. (40 puntos)

Se desea construir un sistema en dos niveles de protección que debe ejecutar concurrentemente 5 tareas independientes. Las tareas de este sistema utilizarán el registro `ecx` como reservado. El mismo contendrá en todo momento un numero de ticks del reloj denominado UTC (Unreal Time Clock). Al ser un registro de la tarea, este puede ser modificado y seteado en cualquier momento. El sistema solamente se encargará de incrementarlo cada vez que la tarea vuelva a ser ejecutada luego de una interrupción de reloj.

Por otro lado, el sistema cuenta además con un servicio que permite mapear cualquier marco de página en el rango virtual 128MB-136MB a cualquiera del rango físico 8MB-16MB.

(10p) (a) 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. Explicar como implementar el servicio del sistema.

(15p) (b) Escribir en ASM el código de la rutina de atención de interrupciones de reloj.

(15p) (c) Escribir en ASM el código de la rutina de atención del servicio del sistema.

Nota: Asumir que se dispone de las definiciones de las estructuras del procesador en C. Se pueden utilizar funciones auxiliares en C responsablemente. Se cuenta además con la función `freePage` que retorna una pagina libre ya mapeada con *identity mapping*

## Ej. 3. (30 puntos)

Se desea implementar en un sistema tipo, un mecanismo que permita informar a las tareas cuando estas son interrumpidas por el sistema. Para esto, cada tarea implementa su propia función `void informar(short flags, void* ret)` que es llamada cada vez que vuelve a correr una tarea previamente interrumpida. Esta función es la encargada de capturar el estado del procesador, guardar que se produjo una interrupción, y seguir como si nada hubiera pasado. Tener en cuenta la función `informar` es ejecutada en el contexto de la tarea interrumpida y en nivel de usuario, además la misma no respeta convención C ya que mínimamente debe salvar todo el contexto de ejecución. Por otro lado, el parámetro `flags` corresponde a los flags en el momento de ser interrumpida y `ret` a la dirección de retorno para continuar con la ejecución de la tarea.

Cuando se produce cualquier interrupción, se debe llamar adecuadamente a la función `informar` que luego retornará al contexto de ejecución de la tarea interrumpida utilizando para esto los datos almacenados en los parámetros `ret` y `flags`.

(5p) (a) Explicar como funciona el mecanismo descripto anteriormente, tomar las decisiones que haga falta para su implementación evitando cualquier ambigüedad. Considerar la forma de acceder a la función `informar` implementada en cada tarea.

(15p) (b) Implementar en ASM la rutina de atención de interrupciones del reloj. Considerar que existe la función `proxima_tarea` que retorna el selector de tss de la proxima tarea a ejecutar. De ser nulo, entonces no se debe realizar ningun intercambio.

(10p) (c) Implementar en ASM la función `informar`, la misma debe llamar a la funcion `void log()` y continuar con la ejecución.

Notas: Las tareas no tienen ninguna modificación especial, solo cumplen con implementar la función `informar`. Sistema Tipo: Segmentación flat, paginación activa y dos niveles de privilegio.