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

# Organización del Computador 2
# Segundo parcial – 19/11/2019

| 1 (30) | 2 (35) | 3 (35) | |
|---|---|---|---|
| | | | |

*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 |
|---|---|---|---|---|---|---|---|---|---|
| 28 | `0x30100123` | `0x23F01` | 1 | 1 | 1 | 0 | 1 | 11 | `0xA` |
| 37 | `0x4A1F0102` | `0x12222` | 1 | 1 | 1 | 0 | 1 | 10 | `0x8` |
| 39 | `0x222F22F2` | `0x40DDD` | 1 | 1 | 1 | 0 | 1 | 00 | `0x2` |
| 43 | `0x1D900D00` | `0x5FFFF` | 1 | 1 | 1 | 0 | 1 | 00 | `0x0` |

Y el siguiente esquema de paginación:

| Rango Lineal | Rango Físico | Atributos |
|---|---|---|
| `0x4CA40000 - 0x4CA55FFF` | `0xD549C000 - 0xD54B1FFF` | read only, supervisor |
| `0x55040000 - 0x5504FFFF` | `0x9A88F000 - 0x9A89EFFF` | read/write, supervisor |
| `0x5EDF2000 - 0x5EFF2FFF` | `0x934F3000 - 0x936F3FFF` | read only, user |

(10p) a. Especificar todas las entradas de las estructuras necesarias para construir un esquema de paginación. Suponer que todas las entradas no mencionadas son nulas.

(20p) 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, en qué unidad, e intentar seguir resolviendo hasta llegar a la dirección física. Definir `EPL` en los accesos a datos. El tamaño de todas las operaciones es de 2 bytes.

I - `00E3:1C9500FF` - CPL `11` - lectura

II - `012A:0AE50088` - CPL `10` - ejecución

III - `013A:3CB00000` - CPL `00` - escritura

IV - `015A:3E25FFFF` - CPL `00` - lectura

## Ej. 2. (35 puntos)

Frigga, enojada porque Thor le rompió la nariz a su hermanastro Loki con su martillo, decide retarlo mandándolo a su cuarto con sólo tres porciones de pizza. Thor, enojado, decide continuar con el desarrollo de un primitivo sistema con extrañas funcionalidades dignas de un dios.

ThorSO permite correr una única tarea denominada de prioridad "suprema", y muchas tareas esclavas. El sistema ejecutará a la única tarea suprema todo el tiempo. Mediante interrupciones, se podrán

cambiar los registros de la tarea "suprema" informando sobre eventos. Las tareas esclavas serán despertadas cada un determinado número de *ticks* de reloj indicados por la misma tarea esclava. Las tareas esclavas utilizarán todos los *ticks* de reloj que necesiten hasta terminar su trabajo. Cuando éstas terminen generarán una nueva interrupción indicando que finalizaron (int 0x66), dejando en `eax` la cantidad de *ticks* en los que debe ser llamada nuevamente la tarea.

La única interrupción externa que le interesa capturar a la tarea "suprema" es la interrupción de teclado. Cuando esta se produce, de estar corriendo la tarea "suprema" se debe modificar el registro `ecx` por 1. Considerando que todas las tareas corren en anillo 3, incluyendo a la tarea "suprema".

(10p) 1. Indicar qué estructuras son necesarias para administrar el sistema y que funciones se requiere para utilizarlas.

(10p) 2. Indicar el tipo y los campos de las entradas en la IDT de las tres interrupciones, int 0x66, reloj y teclado.

(15p) 3. Implementar las tres interrupciones en ASM. Nota: las funciones auxiliares pueden ser realizadas en C.

Nota: Si más de una tarea esclava debe ser despertada en el mismo *tick* de reloj, no se debe considerar un orden especial.

## Ej. 3. (35 puntos)

Se desea implementar una funcionalidad de kernel que cada vez que se desaloje una tarea dentro de la rutina de atención de interrupciones del reloj, se almacene en que función del código de la tarea se produjo la interrupción.

(10p) a. Suponiendo que se cuenta con una función que indica la próxima tarea a ejecutar. Construir una posible rutina de atención de interrupciones de reloj que utilice dicha función para intercambiar tareas. Explicar el funcionamiento de la rutina y de la función que indica la próxima tarea ¿Qué pasa en el caso que el intercambio de tareas sea por la misma tarea?

(25p) b. Modificar la rutina anterior para agregar la funcionalidad pedida. Considerar que se cuenta con la función `void logEIP(uint32_t gdtIndex, void* func)`, que toma el índice en la `GDT` de la tarea que se encontraba ejecutando y el puntero a la función que se estaba ejecutando.

Nota: Asumir que todas las subrutinas dentro del código fueron llamadas mediante la instrucción `call`. Esta ocupa exactamente 6 bytes, siendo los últimos 4 bytes la dirección de la función a llamar. Nota 2: Asumir que la memoria de la tarea ya se encuentra mapeada.