# Segundo Parcial

Primer Cuatrimestre 2023

## Normas generales

- El parcial es INDIVIDUAL
- Puede disponer de la bibliografía de la materia y acceder al repositorio de código del taller de system programming, desarrollado durante la cursada
- Las resoluciones que incluyan código, pueden usar assembly o C. No es necesario que el código compile correctamente, pero debe tener un nivel de detalle adecuado para lo pedido por el ejercicio.
- Numere las hojas entregadas. Complete en la primera hoja la cantidad de hojas entregadas
- Entregue esta hoja junto al examen. La misma no se incluye en el total de hojas entregadas.
- Luego de la entrega habrá una instancia coloquial de defensa del examen

## Régimen de Aprobación

- Para aprobar el examen es necesario obtener cómo mínimo **60 puntos**.
- Para promocionar es condición suficiente y necesaria obtener como mínimo **80 puntos** tanto en este examen como en el primer parcial

**NOTA: Lea el enunciado del parcial hasta el final, antes de comenzar a resolverlo.**

# Enunciado

## Ejercicio 1 - (70 puntos)

En un sistema similar al que implementamos en los talleres del curso (modo protegido con paginación activada), se tienen 5 tareas en ejecución y una sexta que procesa los resultados enviados por las otras. Cualquiera de estas 5 tareas puede en algún momento realizar una cuenta y enviar el resultado de la misma a la sexta tarea para que lo utilice de manera inmediata. Para ello la tarea que realizó la cuenta guardará el resultado de la misma en EAX. A continuación, la tarea que hizo la cuenta le cederá el tiempo de ejecución que le queda a la tarea que va procesar el resultado (lo recibirá en EAX). Tener en cuenta que la tarea que hizo la cuenta no volverá a ser ejecutada hasta que la otra tarea no haya terminado de utilizar el resultado de la operación realizada.

Se desea agregar al sistema una syscall para que la tareas después de realizar la cuenta en cuestión puedan cederle el tiempo de ejecución a la tarea que procesará el resultado.

**Se pide:**

a. Definir o modificar las estructuras de sistema necesarias para que dicho servicio pueda ser invocado.

b. Implementar la syscall que llamarán las tareas.

c. Dar el pseudo-código de la tarea que procesa resultados (no importa como lo procese).

d. Mostrar un pseudo-código de la función `sched_next_task` para que funcione de acuerdo a las necesidades de este sistema.

e. Responder: ¿Qué problemas podrían surgir dadas las modificaciones al sistema? ¿Cómo lo solucionarías?

*Se recomienda organizar la resolución del ejercicio realizando paso a paso los items mencionados anteriormente y explicar las decisiones que toman.*

**Detalles de implementación:**

- Las 5 tareas originales corren en nivel 3.

- La sexta tarea tendrá nivel de privilegio 0.

## Ejercicio 2 - (30 puntos)

Se desea implementar una modificación sobre un kernel como el de los talleres: en el momento de desalojar una página de memoria que fue modificada esta se suele escribir a disco, sin embargo se desea modificar el sistema para que no sea escrita a disco si la pagina fue modificada por una tarea específica.

Se les pide que hagan una función que, dado el CR3 de la tarea mencionada y la dirección física de la página a desalojar, diga si dicha pagina debe ser escrita a disco o no.

La función a implementar es:

```
uint8_t Escribir_a_Disco(int32_t cr3, paddr_t phy);
```

**Detalles de implementación:**

- Si necesitan, pueden asumir que el sistema tiene segmentación *flat*.

- NO DEBEN modificar las estructuras del kernel para llamar a la función que están creando. Solamente deben programar la función que se pide.

## A tener en cuenta para la entrega (para todos los ejercicios):

- Está permitido utilizar las funciones desarrolladas en los talleres.

- Es necesario que se incluya una explicación con sus palabras de la idea general de las soluciones.

- Es necesario escribir todas las asunciones que haga sobre el sistema.

- Es necesaria la entrega de código o pseudocódigo que implemente las soluciones.