Segundo Parcial

Primer Cuatrimestre 2021

Normas generales

- El parcial es INDIVIDUAL, de modalidad remota y a libro abierto.
- Las consultas se realizarán por *Zulip*, las mismas serán iniciadas por los docentes a solicitud de los alumnos. Para solicitar consultas, deberán completar su nombre en una planilla compartida y a medida que los docentes podamos ir respondiendo, los contactaremos mediante mensaje privado de *Zulip* en orden de llegada. Las consultas se respondenrán hasta las 22hs.
- Dispondrán de 5hs para resolver el parcial, a las 17hs del día del parcial se publicará un archivo comprimido que deben descargar, el mismo contendrá todo lo necesario para realizar el parcial.
- Una vez cumplido el plazo se deberá completar un formulario con el *hash* del *commit* del repositorio de entrega.
- Se considerará un lapso de 30 minutos (22:00hs a 22:30hs) adicionales para realizar la entrega del parcial, es decir, realizar el commit en el *git*, push al repositorio remoto y el completado del formulario de entrega. Si por algún motivo tienen problemas con *gitlab*, es fundamental que obtengan el hash del commit en su repositorio local y lo envien en el form de entrega. Esto certifica que realizaron la entrega en tiempo y forma. No será corregido ningún parcial cuyo hash no haya sido enviado, o el hash no corresponda con un commit de su repositorio.
- Durante el parcial usaremos el canal de Anuncios en *Zulip* para darles información actualizada y aclaraciones sobre el examen. Les pedimos que lo revisen con regularidad.

Formato de entrega
Se aceptará como solución archivos en formato `TXT` plano, `Markdown` o `PDF`. No se aceptarán archivos `tex` sin compilar o archivos en formatos `doc` o `odt` sin pasar a formato `PDF`. Es válido entregar todo tipo de documentación adicional como imagenes en `jpg` o `png`. El material de la entrega deberá ser separado por ejercicios en carpetas independientes ya creadas en el *template* de la cátedra.

Criterio de aprobación
Cada ejercicio indica una cantidad de créditos. Para aprobar el parcial deberán sumar al menos 65 créditos y haber realizado el ejercicio 2 de forma correcta.

Ejercicio 1 - 10 créditos

Construir un conjunto de segmentos y un mapa de paginación (un solo directorio y varias tablas de página), tal que las siguientes traducciones sean válidas:

Lógica	Lineal	Física	Acción
0x0060:0x00123001	0x00123011	0x01123001	Leer código
0x0060:0x88A94100	0x88A94110	0x00000110	Ejecutar código
0x0030:0x00000000	0xF0000000	0x00000000	Leer Datos
0x0030:0x00399FFF	0xF0399FFF	0x00000FFF	Escribir Datos

Si no es posible completar alguna traducción, justificar, dejando clara cuál es la razón por la cual no es posible realizar dicha traducción. Por simplicidad considerar que todas las traducciones corresponden a acciones realizadas en nivel cero de exactamente 4 bytes

Ejercicio 2 - 50 créditos - Llamen a Moe

Considerar un sistema en modo protegido con segmentación *flat* y paginación activa. Este ejecuta concurrentemente tres tareas de nivel 3 denominadas `Moe`, `Larry` y `Shemp`. Las tareas requieren tan solo 1KB de memoria entre código y datos. Dado el poco espacio que ocupan, estas no resultan muy astutas y suelen realizar accesos a memoria incorrectos. Estos accesos incorrectos deben ser resueltos sin generar ningún error, cualquiera fuere el lugar donde buscan acceder. Como las tareas deben continuar ejecutando a pesar de generar un *page fault*, no se puede continuar desde el mismo lugar y se debe saltear la instrucción que genero el error en su código. A fin de simplificar el mecanismo para saltar una instrucción, las tareas tienen definido el símbolo `ouch` en el offset `0x51` dentro del código de la tarea, desde donde debe continuar la ejecución en el caso generar una excepción de *page fault*. El código en `ouch` requiere que en el estado de los registros sea exactamente el mismo previo a generar la excepción, y que el valor del `EIP` que produjo el error se encuentre cargado en el tope de la pila.

En este simplificado sistema, el scheduler se encarga de ejecutar una a una las tareas intercambiandolas por cada interrupción del reloj. Cada tarea además, tiene asignada un área de pila de 4MB (tener en cuenta que estas tareas no son muy astutas y no saben usar bien la pila).

Por último, las tareas pueden solicitar al sistema la cantidad de veces que este los salvo de ejecutar incorrectamente un acceso. Este valor debe ser un número entero sin signo de 32bits que jamas es reiniciado.

Se pide:

1. Describir cómo se ubicarían en memoria las tareas. Enumerar para ello todos los rangos de memoria física que ocuparía cada parte de cada tarea, y del sistema. Además explicar como se realiza el mapeo sobre direcciones virtuales, considerando que todas las tareas están compiladas para ejecutar desde la dirección virtual `0xCC000000`. Indicar la dirección virtual donde comienza la pila de nivel 3. Detallar donde se ubicarían las páginas que correspondientes a los Page Directory, Page Table y Pilas de nivel cero.

2. Implementar en ASM/C la rutina de atención de interrupciones del reloj. Explicar su funcionamiento.

3. Implementar en ASM/C las rutinas de atención de interrupciones de *page faul¨t*. Explicar detalladamente el funcionamiento de la implementación propuesta. Recordar indicar como es el mecanismo para restaurar los registros pedidos y cargar el `EIP` en la pila.

4. Diseñar e implementar en ASM/C, funcionamiento del servicio para obtener la cantidad de veces que se salvo a la tarea (genero un *page fault*). Indicar en que registro se retorna el resultado.

5. Recordar que las tareas de modo usuario tienen control sobre los registros de propósito general. Un mal diseño de la funcionalidad pedida podría ocasionar un problema de seguridad que le permita a una tarea escribir en una posición arbitraria de memoria. Explicar brevemente cuál sería este problema y cómo se podría remediar.

Nota: Indicar cualquier dato, estructura, variable o función auxiliar referente al sistema que sea necesaria para lograr la implementación propuesta. Por ejemplo: La posición de los descriptores de TSS dentro de la GDT. En caso de requerir algún valor que no se encuentre definido de forma explícita por el enunciado, proponer un valor que considere razonable.

Ejercicio 3 - 40 créditos - Larry esta en cualquiera

Considerar un sistema con segmentación y paginación activa, que ejecuta *n* tareas utilizando dos niveles de protección. Las tareas cada vez que son desalojadas quedan en un estado tal, que al volver a ejecutar, pueden ejecutar las instrucciones `popad` y `iret` para volver a nivel 3.
Se desea implementar las siguientes funciones que serán ejecutadas en nivel 0:

`int getGdtIndex(tss* task)`
La función debe retornar el indice en la GDT del descriptor de la tss pasada por parámetro. En el caso de no existir, la función debe retornar -1.

`int isRunning(tss* task)`
La función debe retornar un 0 si la tarea no se encuentra corriendo, y 1 si la misma se encuentra siendo ejecutada por el procesador.

`void lastThreeReturnPointers(tss* task, void** pointers)`
La función debe buscar en la pila de nivel 3 y retornar cuales fueron las últimas tres direcciones de retorno de funciones por las cuales paso la tarea. El resultado será cargado en `pointers`, el cual es un vector que tiene espacio para tres punteros. Suponer que en todos los casos existen al menos tres llamados a funciones validos para leer en la pila. Además suponer que todas las funciones de la tarea respetan convención C y construyen un stackframe.

Se pide:

1. Implementar en ASM/C la función `int getGdtIndex(tss* task)`.

2. Implementar en ASM/C la función `int isRunning(tss* task)`, puede asumir que la tarea existe en la GDT.

3. Implementar en ASM/C la función `void lastThreeReturnPointers(tss* task, void** pointers)`, puede asumir que la tarea existe y que NO esta siendo ejecutada.

Nota: No se puede asumir tener ninguna estructura adicional para resolver el ejercicio. Las funciones deben ser resultas usando solo las instrucciones que proporciona el procesador. Ej: No se puede suponer que se tiene una tabla con punteros desde a todas las pilas de nivel 3 de todas las tareas.