Sistemas Operativos II

Integrantes:

- Di Doménico, Fausto.
- Stizza, Federico.

Práctica 1:

Ejercicio 1:

La memoria de la máquina virtual está dada por la constante MEMORY_SIZE en mmu.hh.

En disk.hh const unsigned SECTOR_SIZE = 128;

En mmu.hh

- const unsigned PAGE_SIZE = SECTOR_SIZE; // En bytes
- const unsigned NUM_PHYS_PAGES = 32;
- const unsigned MEMORY_SIZE = NUM_PHYS_PAGES * PAGE_SIZE;
- MEMORY_SIZE = 32 * 128 bytes
- MEMORY_SIZE = 4096 bytes

Por lo tanto el tamaño de la memoria es **4096 bytes o 4 kB**.

Ejercicio 2:

Para cambiar ese valor, habría que modificar el número de páginas (NUM_PHYS_PAGES) o el tamaño de cada página (PAGE_SIZE), que en este caso por simplicidad el tamaño de la página es del tamaño de un sector del disco.

Ejercicio 3:

El tamaño de un disco está dado por la constante *DISK_SIZE en disk.cc.*

En disk.hh:

- const unsigned SECTOR_SIZE = 128; // En bytes
- const unsigned SECTORS_PER_TRACK = 32;
- const unsigned NUM_TRACKS = 32;
- const unsigned NUM_SECTORS = SECTORS_PER_TRACK * NUM_TRACKS;
- NUM_SECTORS = 32 * 32
- NUM_SECTORS = 1024

En disk.cc:

- static const unsigned MAGIC_SIZE = sizeof (int);
- static const unsigned DISK_SIZE = MAGIC_SIZE + NUM_SECTORS * SECTOR_SIZE;
- MAGIC_SIZE = 4 bytes
- DISK_SIZE = 4 + 1024 * 128

Por lo tanto el tamaño del disco es de **131072 bytes o 128 kB + 4 bytes**.

Ejercicio 4:

En total hay **59 instrucciones**, las cuales se encuentran en el archivo mips_sim.cc, en ExecInstruccions de la clase Machine.

Ejercicio 5:

1) Primero realiza la suma entre los registros Rs y Rt.
2) Realiza un XOR de los registros operandos para chequear que: si tienen el mismo signo -> el resultado tambien lo tenga, ya que en caso contrario hubo overflow de la suma.
3) Si no hubo overflow, guarda el resultado en uno de los registros.

Ejercicio 6:

PRIMER NIVEL (main):

- Initialize - /code/threads/system.cc
- DEBUG - code/lib/utility.hh
- strcmp - string.h
- PrintVersion - /code/threads/main.cc
- ThreadTest - /code/threads/thread_test.cc
- Thread::Finish - /code/threads/thread.cc

SEGUNDO NIVEL:

En Initialize:

- ASSERT - /code/lib/utility.hh
- strcmp - string.h
- RandomInit - /code/machine/system_dep.cc
- atoi - stdlib.h
- Debug::SetFlags - /code/lib/debug.cc
- Timer::Timer - /code/machine/timer.cc
- Thread::Thread - /code/threads/thread.cc
- Thread::SetStatus - /code/threads/thread.cc

- Interrupt::Enable - /code/machine/interrupt.cc
- CallOnUserAbort - /code/machine/system_dep.cc
- Cleanup - /code/threads/system.cc
- PreemptiveScheduler::SetUp - /code/threads/preemptive.cc

En DEBUG:

- debug.Print - /code/lib/debug.cc

En PrintVersion:

- printf - stdio.h

En ThreadTest:

- DEBUG - /code/lib/utility.hh
- snprintf - stdio.h
- Thread::Thread - /code/threads/thread.cc

En Thread::Finish:

- Interrupt::SetLevel - /code/machine/interrupt.cc
- ASSERT - /code/lib/utility.hh
- DEBUG - /code/lib/utility.hh
- Thread::GetName - /code/threads/thread.cc
- Thread::Sleep - /code/threads/thread.cc

Ejercicio 7:

Se emula una CPU para poder abstraernos de los detalles de bajo nivel (hardware), de esta manera es sencillo realizar cambios en los distintos modulos de nuestro sistema operativo.

Ejercicio 8:

La macro **DEBUG** utiliza una instancia global (para poder ser llamada desde cualquier punto del código) definida de la clase Debug y llama a su método Print. El cual chequea la *flag* encendida (en 1) y *printea* segun el formato y los párametros pasados.

Y la macro ASSERT chequea la condición dada y en caso de no cumplirse, printea en el log, limpia el buffer y aborta la ejecución.

Util para documentar supuestos en el codigo, es decir, lo que se espera que el programa cumpla.

Ejercicio 9:

Las flags de depuración tienen los siguientes usos:

- '+' – activa todos los mensajes de debug

El resto de las banderas depuran...

- 't' – el sistema de hilos. thread system.
- 's' – semaforos, locks y condiciones.
- 'i' – la emulacion de interrupciones.
- 'm' – la emulacion de maquina. (requiere *USER_PROGRAM*)
- 'd' – la emulacion de disco. (requiere *FILESYS*).
- 'f' – el sistema de archivos. (requiere *FILESYS*).
- 'a' – los espacios de direcciones (requiere *USER_PROGRAM*).
- 'n' – la emulacion de red. (requiere *NETWORK*).

Ejercicio 10: (VER)

Las constantes **USER_PROGRAM, FILESYS_NEEDED,FILESYS_STUB y NETWORK** estan definidas en los **makefiles**.

Ejercicio 11:

SynchList y List son dos clases que implementan listas del estilo Lisp. Con la diferencia que SynchList implementa bloqueos para cumplir con las siguientes invariantes:

- Los hilos que deseen eliminar un elemento de la lista deberan esperar a que haya al menos un elemento en la misma.
- Solo un hilo al mismo tiempo tiene acceso a la lista al mismo tiempo.

Ejercicio 12:

Buscando con el comando grep o con el buscador del editor Atom, se puede ver que las definiciones de main están en los siguientes archivos:

- bin/coff2flat.c
- bin/coff2noff.c
- bin/disasm.c
- bin/fuse/nachosfuse.c
- bin/main.c
- bin/out.c
- bin/out.c
- bin/readnoff.c
- threads/main.cc
- userland/filetest.c
- userland/halt.c
- userland/matmult.c

- userland/shell.c
- userland/sort.c
- userland/tiny_shell.c

La función main del ejecutable nachos del directorio userprog se encuentra en el archivo /threads/main.cc

Esto se puede ver si utilizamos el comando grep -r main en el directorio userprog y se obtiene el siguiente resultado: Makefile.depends:main.o: ../threads/main.cc ../threads/copyright.h ../lib/utility.hh

Ejercicio 13:

Nachos soporta la línea de comando:

```
nachos [-d <debugflags>]  [-p]  [-rs <random seed #>]  [-z]
/// [-s]  [-x <nachos file>]  [-tc <consoleIn> <consoleOut>]
/// [-f]  [-cp <unix file> <nachos file>]  [-pr <nachos file>]
/// [-rm <nachos file>]  [-ls]  [-D]  [-tf]
/// [-n <network reliability>]  [-id <machine id>]
/// [-tn <other machine id>]
```

`-rs` – causes `Yield` to occur at random (but repeatable) spots. Es decir, si ejecutamos nachos con esta opción se le puede pasar una semilla para generar los números aleatorios.

Ejercicio 14, 15 y 16:

Ver los archivos threads_test.cc y make file, del directorio de code/threads. Para el 16, la bandera de depuración correspondiente es 's'.