Informe

Arnix

Modo protegido con GRUB

31 de mayo de 2011

Autores:

Axel Wassington	Legajo: *50124*
Horacio Miguel Gomez	Legajo: *50825*
Tomás Mehdi	Legajo: *51014*

Índice

1. Decisiones e implementación del sistema

1.1. Código

Se utilizo un mix de assembler con C por decisión de la catedra. Es muy importante aclarar este punto, por que varias instrucciones solo pueden hacerse desde assembler y para simplificar la codificacion de usa C llamando a assembler o assembler llamando a C.

1.2. Compilación, linkedición y ejecución

Reemplazamos el compila por un makefile y el arma por un build. El build puede llamarse con el parametro clean o no. Si se llama con el parametro se hace un make clean y make. Sino solo se hace el make. Luego de estos en los dos casos si el código compila se abre el bochs.

1.3. Pantalla

1.4. Tabla global de descriptores

1.5. Interrupciones

2. Funcionamiento del comando getCPUspeed

Muestra la frecuencia de trabajo del CPU, usando la funcion RDTSC(Read Time-Stamp Counter) de assembler la cual retorna la cantidad de instrucciones realizada hasta el momento desde el inicio del procesador y el PIT(Programable Interval Timer). El PIT es un periferico conectado a la IRQ0 del master PIC. Utilizando estos elementos podemos obtener una cantidad de instrucciones en un intervalo de tiempo. La forma de hacerlo es pidiendo un RDTSC, dejar pasar un tiempo fijo y volver a pedir un RDTSC. El tiempo fijo lo generamos con una cantidad coherente de timer ticks, para ello no debe ser muy pequeña. Ya teniendo estos datos solo falta hacer simples cuentas que nos devolveran la velocidad del CPU en MHz.

3. INT80h

Similar a la INT80h de Unix/Linux; la INT80h de Arnix segun el valor en el registro EAX elige una instruccion.Las instrucciones que puede realizar son las siguientes:

1. Con el valor 3 en EAX hace un read usando los valores de EBX, ECX y EDX. En estos regristros debe estar el tamaño de lo que se va a leer, el source buffer y un file descriptor(de donde leer) respectivamente.
2. Con el valor 4 en EAX hace un write usando los valores de EBX, ECX y EDX. En estos regristros debe estar el tamaño de lo que se va a escribir, el source buffer y un file descriptor(donde esribir) respectivamente.
3. Con el valor 5 en EAX hace un rdtsc guardando el valor en el registro EBX.

4. Referencias

Esta sección detalla las distintas fuentes de información utilizadas para el desarrollo del TP Especial. Es importante destacar que son las mismas fuentes enviadas en un mail previo a la entrega.

4.1. Interrupciones

El manejo de interrupciones es similar al usado en el siguiente tutorial:

http://www.jamesmolloy.co.uk/tutorial_html/4.-The%20GDT%20and%20IDT.html

Nos pareció interesante la opción de crear un wrapper para las idt y luego desde C simplemente asignar handlers a las convenientes, un wrapper se encarga de que a C le lleguen los parametros.También creimos importante tener las entrys para las primeras 31 exceptions del procesador, para evitar resets si por ejemplo dividimos por cero.

4.2. Pantalla

Nos basamos en la implementación de Linux, la pantalla puede recibir scape chars para limpiar la pantalla, o imprimir en colores. Esto era conveniente debido a que al utilizar la int80, no necesitamos parametros extra, para estas funcionalidades extra.

4.3. Reboot

Luego de probar soluciones sucias, como hacer que el procesador triple-faultee y se reinicie la pc. Encontrámos la solucion de enviar la señal de reset desde el controlador de teclado en:

http://wiki.osdev.org/Reboot

4.4. Paginas web utilizadas

En general leimos mucho de:

http://wiki.osdev.org/Main_Page (y sus foros)
http://www.jamesmolloy.co.uk/tutorial_html/index.html
http://www.osdever.net/tutorials/view/brans-kernel-development-tutorial