DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico III

El Kernel contraataca

Organización del Computador II

Segundo Cuatrimestre de 2014

Integrante	LU	Correo electrónico
Alejandro Mignanelli	609/11	minga_titere@hotmail.com
Franco Negri	893/13	franconegri2004@hotmail.com
Federico Suárez	610/11	elgeniofederico@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Resumen

En el presente trabajo se describe el desarrolo del Kernel desarrollado para una arquitectura intel de 32-bits, así como el manejo de paginación, manejo de tareas, interrupciones y todo lo referente al manejo de un pequeño sistema operativo.

Índice

1. Objetivos generales

El objetivo de este trabajo practico, partiendo de un procesador intel de 32-bits, generar un kernel capaz de gestionar memoria entre diferentes tareas, correrlas de manera concurrente, y resolver las diferentes problematicas que puedan surgir al momento de ejecución.

Para ello utilizaremos las diversas herramientas que intel pone a nuestra disposicion en modo protegido: Usaremos segmentacion y paginación para controlar el privilegio con el que las tareas se ejecutarán. Utilizaremos interrupciones del procesador que permitirán, tanto reaccionar de manera apropiada cuando se produzca un error en tiempo de ejecución, obtener input del teclado y gestionar un task manager que nos perimita ejecutar tareas de manera concurrente.

En el presente informe, se detallará de manera mas elaborada todo lo hecho para conseguir el objetivo del trabajo práctico, asi como cualquier decisión que se haya tomado en el código a tales fines. Para su mejor entendimiento, este informe se dividirá en ejercicios, que son pequeñas partes del trabajo, y todos juntos conforman al trabajo práctico en si.

2. Ejercicio 1:

En esta sección se ha completado la gdt con las primeras 7 posiciones (no se como ponerle que basicamente pusimos basura para que si la usamos, rompa(ALE)). Luego, hemos puesto 4 segmentos, dos para codigo de nivel 0 y 3, y dos para datos de nivel 0 y 3. Para mayor entendimiento del código, se han usado defines, con nombres que expresan que representan(por ejemplo, el segmento destinado a codigo de nivel 0, se llama GDT_IDX_CDE_LVL_0). Estos segmentos direccionan los primeros 623 MB de memoria. Tambien se ha declarado un segmento que describe el area de la pantalla en memoria que puede ser utilizado solo por el kernel. Esto se utilizará al principio para imprimir por pantalla, pero más avanzado el trabajo se necesitaran imprimir muchas cosas, y C provee herramientas más comodas para esto. Dado que la convención C nos pide que todos los segmentos de datos apunten al mismo segmento, este segmento termina quedando en desuso. (aca hay que poner que completamos codigo parapasar a modo protegido y setearl la pila del kernel en 0x27000... lo escribimos directamente??... tambien falta el punto d... es necesario?? no es un ejercicio de transicion que despues no va a tener la menor importancia?? (ALE))

3. Ejercicio 2:

4. Ejercicio 3:

5. Ejercicio 4:

6. Ejercicio 5:

7. Ejercicio 6:

8. Ejercicio 7: