DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico III

System Programming

Organización del Computador II

Segundo Cuatrimestre de 2020

Integrante	LU	Correo electrónico
Ivo Pajor	460/19	ivo_pajor@hotmail.com
Laureano Muñiz	498/19	lau2000m@hotmail.com
Luciana Gorosito	577/18	lugorosito0@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

Índice

1. Introducción

2. Desarrrollo

2.1. Ejercicio 1

Para la realización de este ejercicio analizamos las estructuras **gdt_entry_t** y **gdt_descriptor_t** definidas por la cátedra. En esta implementación, la Tabla de Descriptores Globales(GDT) es un arreglo de **gdt_entry_t** y su descriptor, que luego cargaremos en GDTR, es **gdt_descriptor_t**.

Siguiendo lo indicado en el primer item, definimos a partir del índice 10, 4 descriptores de segmento en la GDT utilizando la estructura **gdt_entry_t** atendiendo a las propiedades particulares de cada segmento. Puesto que estos segmentos deben direccionar los primeros 201 MB de memoria establecimos en todos el bit de G en 1, y definimos su base en 0x00000000h y su limite en 0x00C8FFh. Además, estos 4 son segmentos de código o datos de 32 bits por lo que el bit S se encuentra seteado en 1, el de D/B se encuentra seteado en 1 y el bit L en 0. El bit de DPL de cada uno está seteado en 0 o en 3 de acuerdo con el nivel de privilegio que le corresponda. Por último, los bits de tipo están seteados como 0xAh en caso de tratarse de un segmento de código y como 0x2h en caso de tratarse de un segmento de datos.

Para poder pasar a modo protegido, en el kernel deshablitamos las interrupciones, cargamos en el registro GDTR la estructura **gdt_descriptor_t** y modificamos el ultimo bit del registro de control CR0, es decir, seteamos en 1 el bit de *Protection Enable*. Posteriormente, escribimos el código necesario para saltar efectivamente a modo protegido. Debido a que este salto se consigue haciendo un *far jump* a la próxima instrucción, designamos una etiqueta llamada **modo_protegido** a partir de la cual obtendremos el offset, mientras que como selector utilizamos el correspondiente al segmento de código de nivel 0.

2.2. Ejercicio 2

2.3. Ejercicio 3

2.4. Ejercicio 4

2.5. Ejercicio 5

2.6. Ejercicio 6

2.7. Ejercicio 7

2.8. Ejercicio 8

2.9. Ejercicio 9