



DEPARTAMENTO  
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

# Trabajo Práctico II

subtitulo del trabajo

Organización del Computador II  
Segundo Cuatrimestre de 2014

Integrante	LU	Correo electrónico
Nombre	XXX/XX	mail
Nombre	XXX/XX	mail



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 la problemática de ...

## **Índice**

<b>1. Objetivos generales</b>	<b>3</b>
<b>2. Desarrollo</b>	<b>3</b>
2.1. Segmentación y manejo de excepciones . . . . .	3
2.2. Paginación . . . . .	3
<b>3. Conclusiones y trabajo futuro</b>	<b>3</b>

## 1. Objetivos generales

El objetivo de este Trabajo Práctico es ...

## 2. Desarrollo

TODO: agregar introducción a lo que hicimos?

### 2.1. Segmentación y manejo de excepciones

De acuerdo a lo indicado en el enunciado, definimos segmentos de código y datos en la GDT a partir del índice 4, un par con privilegios de kernel y otro con privilegios de usuario. Direccinamos los primeros 878MB de memoria con estos segmentos. Para ello, establecimos la base de cada segmento en la dirección 0x0 y, para poder representar el número en 20 bits, calculamos el límite de cada uno en bloques de 4KB. Adicionalmente, definimos un segmento de datos con privilegios de kernel para la memoria de video, basado en la dirección 0xB8000. Dado que las dimensiones de la pantalla son 80x50 caracteres y que para representar cada uno se necesitan dos bytes, el tamaño de este segmento es de 8000 Bytes. En función de esto definimos el límite del segmento en 7999, su último byte direccionable.

Resolvimos el manejo de excepciones definiendo los primeros veinte índices (de 0 a 19) en la tabla de descriptores de interrupción. Para ello utilizamos un macro en el cual referimos cada entrada  $N$  al segmento de código de kernel, con el offset correspondiente su rutina de atención `_isrN`. En cuanto los atributos de cada descriptor, dejamos en 1 el bit de presencia, asignamos nivel de privilegios de kernel y usamos el tipo *interrupt*.

Para atender excepciones escribimos rutinas que muestran sus mensajes de error por pantalla. Definimos cada `_isrN` con un macro en el cual obtenemos y mostramos el  $N$ -ésimo mensaje de un arreglo donde los guardamos previamente.

### 2.2. Paginación

TODO: Explicar ej. 3 y 4

## 3. Conclusiones y trabajo futuro