Arquitecturas de las Computadoras
TP5 - Acceso a Hardware

Introducción

Hasta ahora estuvimos trabajando como cliente del Sistema Operativo (SO) a través de su API. La responsabilidad del SO es administrar los recursos de la computadora. Estos recursos son los periféricos y la memoria. En esta guía vamos a hacer un acceso directo y empezar a trabajar de cómo administrarlos.

Para este trabajo vamos a hacer uso de una arquitectura con un *bootloader* modificado por la cátedra que al arrancar el sistema, lo deja configurado en ***long mode*** con un 1 GB de memoria inicializado.

Bajar el código de ejemplo de:

```
https://bitbucket.org/RowDaBoat/x64barebones/
```

El código de ejemplo cuenta con los siguientes componentes

- Bootloader: El código del bootloader
- Image: Las imágenes para utilizar con los emuladores
- Kernel: Código correspondiente al OS
- Toolchain: Herramientas de empaquetación
- Userland: El código de los usuarios o cliente del OS.

Para correr el código de ejemplo, primero se debe compilar el *toolchain,* ir a dicha carpeta y ejecutar ***make,*** luego desde la carpeta raíz del proyecto, y ejecutar **make** nuevamente. Luego, correr el comando

```
$> ./run.sh
```

Ejercicio 1

La pantalla es un periférico que está mapeado en memoria. En modo texto comienza *0xB8000.* Consta de 80 columnas y 25 filas, donde cada posición consta de dos componentes, una para el carácter a dibujar y su atributo, el color de fondo y el color de la letra.

Escriba en la pantalla la leyenda “Arquitecturas de las Computadoras”, con blanco y letras verdes.

Ejercicio 2

Basado en el ejercicio anterior, escribir una función para escribir mensajes por la pantalla. Implemente un mini driver de video.

Ejercicio 3

El **Real Time Clock (RTC)** es el periférico que almacena la hora del sistema. Mientras la Placa Madre tenga pila, éste llevará la hora y es lo que permite conservar la hora luego de que la PC se apague y se le quite la energía.

El RTC permite leer y configurar la el año, el mes, el día, la hora, los minutos y los segundos por separado. También permite configurar una alarma de la misma forma. Para no consumir tantas direcciones de E/S, se accede de manera diferida. En el registro de E/S 70h, se elige lo que se quiere leer/escribir y en el registro 71h se hace la lectura. Por ejemplo, para obtener el los segundos, habría que ejecutar las siguientes líneas:

```
mov al, 0
out 70h, al
in ax, 71h
```

Implemente una función que informe por pantalla la hora actual del sistema. Para más información sobre el RTC puede acceder a la siguiente página:

Doc: http://stanislavs.org/helppc/cmos_ram.html

Ejercicio 4

Hay dos formas de obtener teclas del teclado. La primera que vamos a utilizar es por *pooling,* que consiste en consultar constantemente por la existencia en una tecla.

Lea la documentación del funcionamiento del teclado y escriba una función que se quede esperando que se haya presionado una tecla. Cuando esto suceda, mostrar en pantalla que tecla fue la que se presionó.

Doc: http://stanislavs.org/helppc/8042.html

Ejercicio 5

Ahora vamos a hacer uso de las interrupciones de Hardware. Antes que nada, es necesario configurar la IDT para que cuando una interrupción llegue al procesador, busque en la tabla cuál es el código a ejecutar.

Tener en cuenta que según el manual de ***Pure64*** la IDT quedó configurada a partir de la dirección 0 de la memoria, y las interrupciones han sido corridas 0x20 posiciones. Buscar cómo es el formato de la IDT en 64 bits, escriba una función para modificar sus entradas.

Implemente la interrupción de Timer Tick para que que cada 5 segundos imprima un nuevo mensaje en la pantalla.

Ejercicio 6

Agregue un nuevo handler a la IDT para manejar las teclas del teclado mediante interrupciones. Las teclas deben ir apareciendo en la pantalla a medida que se escriban.

Ejercicio 7

Como parte de la implementación de un Sistema Operativo, es necesario proveer una API al usuario para que use funciones de éste. Agregue a la IDT la entrada 80h, configure un handler adecuado y provea el **System Call write**, que funcione de acuerdo a la implementación que se ha visto en clase. Defina cómo fds:

- STDOUT (1) : la pantalla
- STDERR (2) : la pantalla, pero con texto rojo.

El objetivo es administrar y facilitar el acceso a los distintos recursos del sistema. La pantalla es uno.