Proyecto Final
R-222: Arquitectura del Computador

Felipe Andrés Tenaglia Giunta
Legajo: T-2658/1

3 de diciembre de 2012

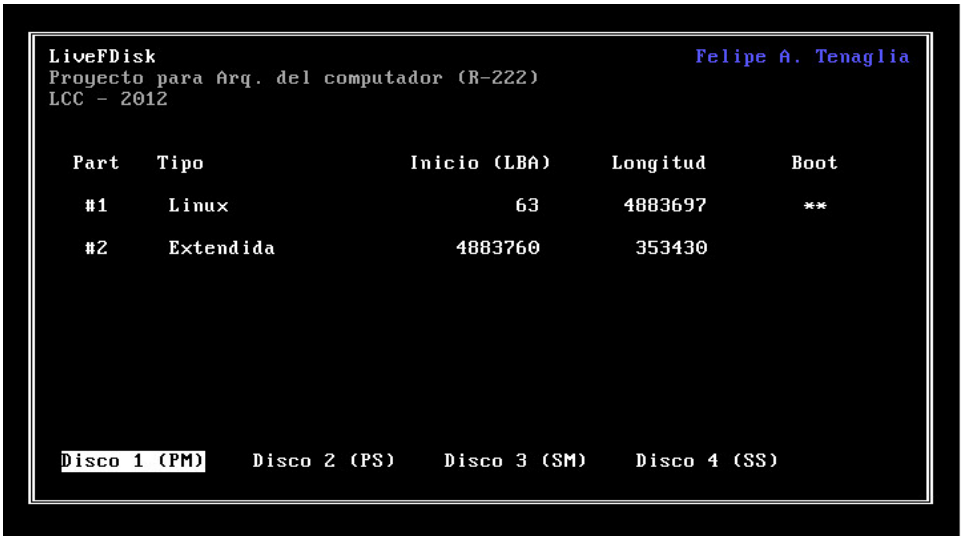

Figura 1: Captura de pantalla del programa

Proyecto elegido: LiveFDisk

El objetivo del trabajo era desarrollar una aplicación booteable que nos permita analizar el contenido del MBR del disco y que nos imprima en pantalla las particiones primarias existentes y así aprender el proceso de inicio de una computadora y el manejo directo de hardware a través de funciones de la BIOS y acceso a puertos.

Características

El programa permite la lectura de la tabla de particiones de los cuatro discos principales del sistema, mostrando en pantalla el tipo de cada partición, el inicio, el fin y la longitud (tanto en formato CHS como en sectores) y si está marcada como partición de booteo. Así también permite eliminar particiones existentes. La Figura 1 se muestra una captura de pantalla del programa, luego de haber iniciado.

Comandos del teclado

- `1 - 4 =` Cambia el disco activo (1 = Primary Master, 2 = Primary Slave, 3 = Secondary Master, 4 = Secondary Slave)
- `E =` Habilita el menú para la eliminación de particiones
- `U =` Alterna las unidades de visualización (Ternas CHS o sectores)

Problemas y dificultades

Durante el desarrollo me surgieron algunas dificultades, relacionadas tanto con decisiones de diseño, como también con impedimentos propios de la BIOS.

Límite de bytes

Como es sabido, para los *bootloaders* existe un límite de 512 bytes. [1] En un principio había decidido diseñar el programa de manera óptima para que el tamaño de su binario no exceda este límite, pero para poder agregarle funciones y para mejorar la interfaz de usuario, decidí separar al programa en 2 etapas.

La primera etapa cumple con el límite de 512 bytes y es la encargada de borrar la pantalla, leer el disco y cargar la segunda etapa. La segunda instancia es el programa considerado en sí mismo. En él, mediante las llamadas a la BIOS y a través de los puertos, podemos leer la tabla de particiones de los 4 discos principales, si estos existen.

Administración de memoria

En parte relacionado con el apartado anterior, la administración de memoria fue un punto a considerar, ya que debiamos establecer para cada etapa, principalmente, donde estaría alojado el código y su respectivo *stack* y en donde almacenaríamos el MBR leído, todo esto considerando la limitación de memoria propia al estar ejecutándose el programa en *modo real*. Finalmente el mapa de memoria quedó establecido como se ve en la Figura 2.

Conflicto entre interrupciones

Durante el desarrollo, al incorporar el manejo de los discos a través del teclado, por alguna razón al ejecutar la interrupción de acceso a disco (`INT 0x13`) luego de una interrupción de teclado (`INT 0x16`), estas colisionaban, bloqueando el acceso a disco a través de esta interrupción, por lo tanto decidí que la lectura del MBR se realice directamente a través de puertos.

Multiboot Specification

Para lograr generar un kernel que pueda ser booteado con algún gestor de arranque compatible con la Multiboot Specification (i.e. GRUB), leí la

[1] En la práctica son 510 bytes ya que los ultimos 2 corresponden a la 'firma': 0x55AA

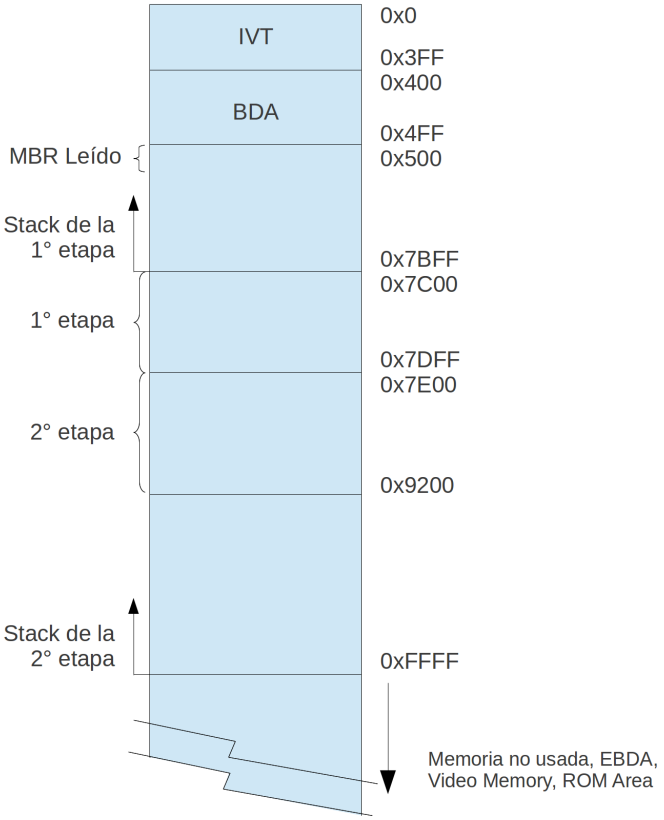

Figura 2: Mapa de memoria del programa

especificación e incluí el header como se explica en ella.

Además hubo que cambiar las interrupciones de BIOS por lectura y escritura de puertos, ya que por estar ejecutándose nuestro programa en modo protegido, estas no son soportadas.

Posibles extensiones

Como fue propuesto se puede extender para que se puedan crear y modificar las particiones primarias como así también incluir la opción de formatear dichas particiones. También se podría incorporar la capacidad de listar las particiones lógicas.

Una extensión más simple puede ser mejorarle el reconocimiento de tipos de particiones, ya que actualmente está limitado a:

- Extendida (`0x5`)
- FAT16 (`0x6`)
- NTFS (`0x7`)
- FAT32 (`0xB`)
- Linux Swap (`0x82`)
- Linux (`0x83`)

Referencias

- Documentación de NASM
- Wikipedia: MBR
- OSDev.org
- MultiBoot Specification