**Tutorial 11**

Una vez ya comprendida las partes de como entrar a modo protegido, utilizar la memoria, escribir en pantalla y usar los nuevos registros de 32 bits, ya es hora de que podamos empezar a cargar nuestro kernel en memoria y programar los proximos componentes de un sistema operativo. Para esto, es necesario nuevamente revisar como se usa el controlador de diskette y poder cargar un nuevo archivo que sera nuestro kernel final, asi concluyendo el uso de la etapa 2 del bootloader que hemos estado programando.

En este tutorial se debe nuevamente leer sectores de el diskette usando las funciones de la BIOS. Sin embargo, toda esta lectura debe realizarse antes de poder entrar a modo protegido ya que en modo protegido no se pueden utilizar los interruptores de la BIOS porque ocasionarian una triple falta y reiniciaria el sistema. Para este caso se sigue utilizando el interruptor 13 de la BIOS que utiliza la controladora del diskette para leer sectores del diskette.

Sin embargo, no se le puede decir al interruptor que lea mas del sector 18 porque los diskettes tienen solo 18 sectores por pista. Para ese caso se tiene que cambiar el direccionamiento nuevamente de LBA a CHS para que se pueda leer por cilindro-head-sector. De esta manera si se puede utilizar el interruptor y leer los sectores del disco para cargarlos a un buffer en memoria y posteriormente ejecutar el codigo. Para hacer toda esa lectura tambien se debe poder leer el sistema de archivos FAT12 del diskette y poder averiguar en que sectores de la cadena de la FAT Table esta nuestro kernel copiado. Una vez que podamos decodificar la tabla FAT, es simple cuestion de leer el primer sector de la cadena del archivo, llamar al interruptor, colocar en buffer lo leido y luego seguir leyendo la cadena de la tabla FAT hasta llegar al final del archivo.

Una vez cargado el archivo, se puede colocar el procesador en modo protegido y hacer un salto hacia la nueva direccion en descriptor:offset en memoria (0x1000) que es un 1MB y cargaremos el codigo de nuestro kernel.

