**Tutorial 5**

En ensamblador se escucha un termino de “Ring 0” o “Ring 3” y se refiere a los anillos, entornos, o contextos en los que se ejecutan los programas. El kernel trabaja sobre el “Ring 0” mientras que los programas de los usuarios se manejan en el “Ring 3”. Entre menos sea el numero del “Ring” mas control tiene sobre el hardware. Existen 4 Rings : 0, 1, 2, 3. Un procesador puede cambiar el ring sobre el que esta trabajando si ocurre una excepcion, llamada al sistema (syscall), interruptor o hace un far jmp, return, etc.

Para efectos de este tutorial y como no se puede programar toda la logica o programacion para cargar nuestro kernel y preparar nuestro ambiente de 32 bits en un solo archivo de bootloader, se creara un segundo archivo de bootloader que sera como la etapa 2 de el inicio. Lo que se hace es simplemente cargar el bootloader inicial de 512 bytes y lo que hace este es cargar un segundo bootloader que puede ser de 512 bytes o mas tamaño.

Para poder leer la etapa 2 de nuestro bootloader necesitaremos ayuda de un interruptor de la BIOS que nos lea sectores del floppy virtual. El interruptor 0x13 con la funcion 0 se empieza utilizando para hacerle un reset al controlador del floppy. Si este reset falla, no podremos hacer mas llamados al controlador del floppy mediante la BIOS ya que no sabremos en que sector esta leyendo la controladora y de este modo nos aseguramos que regrese al sector 0. Luego utilizaremos el interruptor 0x13 con funcion 2 para leer sectores. De parametros pasamos el numero de la funcion, los sectores a leer, numero de cilindro, numero de sector por comenzar, numero de head o cabeza de disco, numero de dispositivo y por ultimo preparamos el segmento:offset ES:BX para que tenga en buffer lo leido. Una vez hemos hecho la llamada al interruptor, este colocara el status en el registro AH, el numero de sectores leidos en AL y el Carry Flag o CF no tendra nada puesto si todo fue exitoso o si fue puesto entonces significa que hubo un problema. Una vez leido, la segunda etapa del bootloader se debe de haber puesto en el segmento:offset 0x1000:0

De aquí en adelante, nuestra segunda etapa de bootloader ya se encuentra en memoria y podremos hacer un jmp hacia la direccion en memoria de 0x1000 y ejecutarlo. Este nuevo bootloader nos ayudara a preparar el procesador para entrar en modo de 32 bits protegido, habilitar la linea 20 de direccionamiento de memoria para acceder hasta 4GB de RAM, ademas de colocar las tablas GDT, LDT, e IDT respectivamente para mapear la memoria.

El codigo para leer y colocar en memoria nuestra segunda etapa del bootloader seria:

.Reset:

mov ah, 0 ; resetear el floppy drive

mov dl, 0 ; drive 0 is floppy drive

int 0x13 ; llamar a la BIOS interruptor 13

jc .Reset ; Si el CF (Carry Flag) esta seteado, entonces volvemos a intentar resetear.

mov ax, 0x1000 ; vamos a cargar nuestra segunda etapa a la direccion 0x1000:0

mov es, ax

xor bx, bx

.Read:

mov ah, 0x02 ; funcion 2

mov al, 1 ; leer 1 sector

mov ch, 1 ; seguimos en el track 1

mov cl, 2 ; numero de sector a leer

mov dh, 0 ; numero de head

mov dl, 0 ; numero de drive (0 si es floppy drive)

int 0x13 ; llamar a la BIOS interruptor 13

jc .Read ; Si ocurre un error y esta seteado CF, entonces volver a intentar leer

jmp 0x1000:0x0 ; hacer un jump al segmento:offset 0x1000:0 y ejecutar la segunda etapa.sss