**Tutorial 4**

En el pasado tutorial se programo un pequeño bootloader que lo único que hacia era limpiar los interruptores y parar el procesador mediante los comandos cli y hlt respectivamente. También se nos dio a conocer que los procesadores corren en varios modos, primeramente en Real Mode de 16 bits. En Real Mode el procesador opera con el modelo de memoria segmento:offset, esta limitado a solo 1MB de RAM, no tiene memoria virtual y protección de memoria.

Con el modelo de segmento:offset, se tiene una limitante de 1MB en cuanto a lo que el procesador puede utilizar. Se utilizo este modelo porque al principio cuando los desarrolladores de Intel estaban por lanzar el procesador 8086 de 16 bits, este solo tenia acceso en un inicio a 64KB de memoria pero las aplicaciones iban a requerir mas. Es por eso que los desarrolladores implementaron este modelo de segmento:offset para poder acceder hasta 1MB de memoria. Existen 4 registros especiales en el CPU que guardan las direcciones bases de los segmentos y son **CS, DS, ES y SS**.

Un offset es un numero que se agrega un numero base para obtener un numero final de una dirección en memoria. Por ejemplo Offset = Numero base + offset, Offset = 2, base 3. Resultado Final 3+2 = 5. Lo mismo aplica para el modelo de memoria de segmento:offset de los procesadores y la formula es la siguiente: Dirección de memoria = (Dirección de Segmento \* 16(decimal)) + Offset. El problema con esta formula y con el procesador en Real Mode es que varias direcciones de formato segmento:offset pueden hacer referencia a una sola dirección de memoria, por lo cual no se puede proteger de sobrescribir los datos en memoria. Ejemplo a) Segmentos sin overlapping, Ejemplo b) Overlapping de segmentos:

A description...

Para resolver este problema, los procesadores posteriores implementaron un modo llamado Protected Mode en el cual mediante una tabla de descriptores se puede hacer un mapeado de la memoria y asi limitar y proteger el acceso a la memoria mediante el kernel. Sin embargo, Protected Mode tiene limitantes en cuanto a lo que tiene Real Mode, por ejemplo : En Protected Mode se hace lo siguiente :

* No se puede usar interruptores (Uso de interruptores crearía una triple falta y ocasionar un reinicio)
* Cualquier error podría ocasionar una triple falta
* Se requiere el uso de tablas de descriptores GDT, LDT e IDT.
* Tenemos acceso hasta 4GB de memoria con protección.
* El uso de direccionamiento segmento:offset se utiliza en conjunto con un acceso lineal de memoria.

En este tutorial se agrega al bootloader lo que es el OEM Parameter Block o una especie de descripción de lo que contiene el sistema de archivos en el disco o tabla FAT12. También se agrega el uso de ciertos interruptores de la BIOS con los cuales podemos imprimir en pantalla. El interruptor 0x10 nos ayuda a imprimir una carácter de la siguiente manera para imprimir el carácter 'A' :

;;;;;;;;;;;;;;;;;;;;;;;;;Código para imprimir un carácter en ensamblador;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

xor bx, bx ; A faster method of clearing BX to 0

mov ah, 0x0e

mov al, 'A'

int 0x10

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

Otro interruptor que podemos utilizar es el interruptor 0x12 que nos da el numero de kilobytes de memoria y se utiliza asi:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

xor ax, ax

int 0x12 ; Ahora el registro AX contiene el numero de Kilobytes de el sistema

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;