TRABAJO PRÁCTICO

MÁQUINA VIRTUAL - PARTE III

Introducción

En esta tercera parte se deberá ampliar la máquina virtual para brindar soporte a trabajar con memoria secundaria (virtual disk drives) y segmentos en tiempo de ejecución por medio de llamadas al sistema.

Máquina virtual

Se debe entregar el código fuente y el ejecutable compilado de la máquina virtual, la cual debe poder utilizarse desde una consola del siguiente modo:

```
vmx.exe filename.vmx [filename.vmi] [m=M] [-d] [disco0.vdd disco1.vdd ...]
```

Donde:
- ***vmx.exe*** es el programa ejecutable del proceso Ejecutor o Máquina Virtual.
- ***filename.vmx*** (obligatorio) es la ruta y nombre del archivo con el programa en lenguaje máquina (puede ser cualquier nombre con extensión ***.vmx***).
- ***filename.vmi*** (opcional) es la ruta y nombre del archivo donde se almacenará la imagen de la máquina virtual (puede ser cualquier nombre con extensión ***.vmi***).
- ***m=M*** (opcional) permite indicar el tamaño de la memoria principal, donde ***M*** es un valor expresado en KiB. Si se omite, el valor por defecto sigue siendo 16 KiB.
- ***-d*** (opcional) es un flag que fuerza a la máquina virtual a mostrar el código *Assembler* correspondiente al código máquina cargado en la memoria principal.
- ***filename.vdd*** (0 o varios archivos) es la ruta y nombre de los archivos donde se almacenarán las imágenes de los discos virtuales (puede ser cualquier nombre con extensión ***.vdd***).

Descripción de la máquina virtual

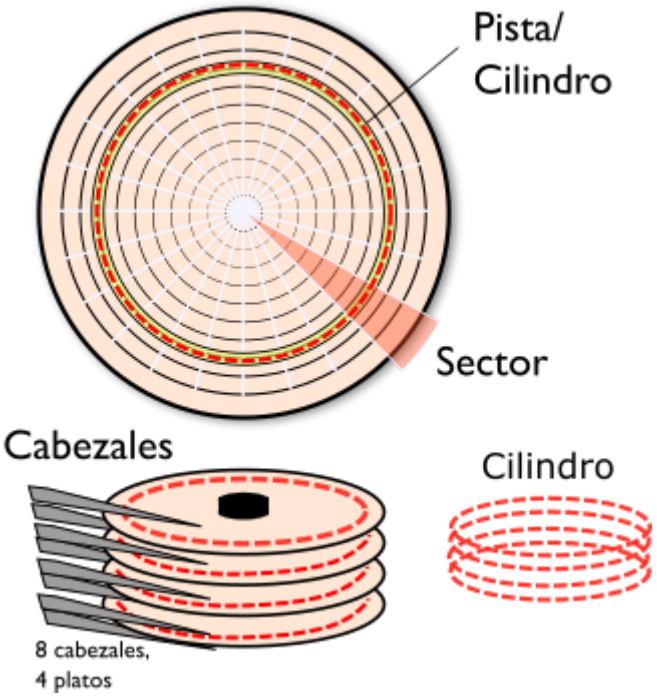
La máquina virtual a implementar en esta tercera parte, debe tener los siguientes componentes:

- Memoria principal (RAM) de tamaño variable.
- Tabla de descriptores de segmentos.
- 16 registros de 4 bytes.
- 1 procesador con:
  - Capacidad para decodificar instrucciones de tamaño variable con 27 códigos de operaciones y 3 tipos de operandos.
  - Capacidad de direccionar a cada byte de la memoria.
- **Descriptores de discos**

Memoria secundaria (trabajar con dispositivos)

La MV recibe por la línea de comandos una lista de archivos ****.vdd*** (virtual disk drive) que representarán unidades de discos. Cada unidad de disco se numera desde **0** hasta **N** según el orden en el listado de archivos que se conforman en la línea de comandos (Máximo 255 archivos, límite teórico).

Cada disco se compone de Cilindros, Cabezas y Sectores:



Formato de los archivos VDD

Cada archivo ****.vdd*** contendrá una cabecera (*header*) de 512 bytes con la información necesaria para su interpretación:

Bytes	Contenido
4	**"VDD0" 4 caracteres identifica el tipo de archivo. [FIJO: 0x56444430]**
4	**Número de versión, llevará un 1 entero sin signo. [FIJO: 1]**
16	GUID - Identificador del disco. [ej: 0x7ee914b137774e84b31387ee9bed04a2]
4	Fecha de creación: año(2B), mes(1B), día(1B)
4	Hora creacion: hora(1B), Minutos(1B), Segundos(1B), Décimas(1B)
1	**Tipo, 0: estático, 1: dinámico [FIJO: 1]**
1	Cantidad de cilindros, entero sin signo [default: 128]
1	Cantidad de cabezas, entero sin signo [default: 128]
1	Cantidad de sectores, entero sin signo [default: 128]
4	**Tamaño del sector en bytes, entero sin signo. [FIJO: 512]**
472	Relleno, reservado para uso futuro [default: 0]

Se considerará que el tipo de disco siempre será 1 (dinámico) y cada sector contendrá siempre **512 Bytes de información, que constituyen la mínima cantidad de información que se puede leer/escribir de un disco**.

Si el ***.vdd*** no existe, la MV debe crearlo y lo hará con los valores por defecto. Por lo tanto un disco virtual creado por defecto tendrá 128 * 128 * 128 * 512 bytes = $2^7 * 2^7 * 2^7 * 2^9$ bytes = 2^{30} bytes = 1 GiB.

El identificador de disco GUID se genera de forma aleatoria.

La máquina virtual deberá guardar la información de cada disco virtual: número unidad, archivo físico real, cantidad de cilindros, cantidad de cabezas, cantidad de sectores y tamaño del sector.

Se toma la siguiente convención: los cilindros crecen de la periferia hacia el centro, las cabezas se numeran de arriba hacia abajo, y los sectores en sentido horario.

En el archivo la información se guarda en forma binaria y contigua, a continuación del header en bloques de 512 bytes ordenados por cilindro, cabeza y sector (CHS).

Es decir que si el disco tiene 128 cilindros, 8 cabezas, 128 sectores y se quiere acceder al cilindro 2, cabeza 7, sector 10; se deberá hacer un seek de: **17241600 Bytes** (fseek(disco, 17241600, SEEK_SET);)

Calculado del siguiente modo:

HD + c * H * S * TS + h * S * TS + s * TS

Donde:

HD = Tamaño del header
c = número de cilindro (inicia en 0)
h = número de cabeza (inicia en 0)
s = número de sector (inicia en 0)
C = Cantidad de cilindros
H = Cantidad de cabezas
S = Cantidad de sectores
TS = Tamaño del sector en Bytes

512 Bytes + 2 * 8 * 128 * **512 Bytes** + 7 * 128 * **512 Bytes** + 10 * **512 Bytes** =
512 Bytes + 16777216 **Bytes** + 524288 **Bytes** + 5120 **Bytes** =
17241600 Bytes

Si cuando se accede a leer/escribir en el disco, el tamaño del archivo ***.vdd*** no llega a cubrir el sector, se debe extender el tamaño del archivo para contener los sectores, sin devolver error a la MV.

Por ejemplo: un archivo de un disco vacío podría tener solo el encabezado y ningún dato más. Cuando se lee el sector 0 de la cabeza 0 del cilindro 0, ahí se completan los 512 bytes con el valor 0 y se extiende el archivo. Luego se quiere escribir el sector 1 de la cabeza 0 del cilindro 0 y se vuelve a aumentar el tamaño del archivo ***.vdd***. Es decir que el tamaño del disco virtual es fijo para la máquina virtual pero dinámico para la máquina real.

Cada acceso de lectura/escritura en un disco virtual, deberá corresponderse con la lectura/escritura en el archivo real al que está vinculado.

Acceso al disco desde al MV

El acceso al disco se realizará mediante la **llamada al sistema 13 (%D)**. Por medio de esta interrupción se podrá consultar el estado del disco, cargar ***n*** sectores a un buffer de memoria o descargar ***n*** sectores desde un buffer de memoria.

El buffer de memoria no es más que un rango de celdas (bytes) de memoria (en cualquier segmento destinado a datos, *Data Segment*, *Extra segment* u otro segmento creado dinámicamente) en donde se cargará, o descargará, la información del disco.

La interrupción requiere el seteo de los siguientes registros/subregistros:

reg	*Parámetros de entrada*	*Resultados de salida*

AH	Se indica el número de operación:
	`%00 Consultar último estado`
	`%02 Leer del disco`
	`%03 Escribir en el disco`
	`%08 Obtener los parámetros del disco`
		Código de estado
		`%00 Operación exitosa`
		`%01 Función inválida (parám AH)`
		`%04 Error de transferencia en lectura`
		`%0B Número inválido de cilindro`
		`%0C Número inválido de cabeza`
		`%0D Numero invalido de sector`
		`%31 No existe el disco`
		`%CC Error de transferencia en escritura`
		`%FF Falla en la operación`

AL	Se indica la cantidad de sectores a leer/escribir	Cantidad de sectores transferidos
CH	Se indica el número de cilindro	Cantidad de cilindros (caso de %08)
CL	Se indica el número cabeza	Cantidad de cabezas(caso de %08)
DH	Se indica la sector	Cantidad de sectores(caso de %08)
DL	Se indica el número de disco (*.vdd)	-
EBX	Se indica la primer celda del buffer de lectura/escritura	-

Ejemplo: se solicitan leer 2 sectores de la unidad de disco 0 a partir del cilindro 8, cabeza 2, sector 10 y se debe almacenar en el *Extra Segment* a partir de la celda 64.

```
mov     EBX, ES
add     EBX, 64    ; Dirección del buffer de lectura
mov      AH, %02   ; Leer del disco
mov      AL, 2     ; 2 sectores a leer
mov      CH, 2     ; Cilindro 2
mov      CL, 8     ; Cabeza 8
mov      DH, 10    ; Sector 10
mov      DL, 0     ; Unidad de disco 0
sys      %D        ; Indica que se realiza la lectura
```

Una vez finalizada la lectura, se rellenaran 1024 bytes, a partir de la celda 64 del *Extra Segment*:

512 bytes/sector * 2 sectores = 1024 bytes

Lo mínimo que se puede leer o escribir del disco es 1 sector, por lo tanto son 512 bytes de la memoria principal. Tanto para la lectura como la escritura, los bytes deben estar contiguas en la memoria y debe haber espacio en el segmento en el que se escriben, en caso contrario se devolverá error **%04 "Error de lectura"** si estaba leyendo, o **%CC "Falla de escritura"** si estaba escribiendo.

Siempre que se ejecute una de las funciones de la interrupción 13 (%D), a excepción de función "Obtener los parámetros del disco" (%08), deberá chequear que:

- la unidad de disco exista (DL),
- el número de cilindro no supere el máximo (CH),
- la cantidad de cabezas no supere el máximo (CL),
- el número de sector no supere el máximo (DH)

En caso de error debe devolver el código correspondiente en el registro AH: **%31, %0B, %0C, %0D**. Si se invoca una función fuera de las 4 permitidas (**%00, %02, %03, %08**) devolverá error **%01 "Función inválida"**.

En el caso de la función "Obtener los parámetros del disco" (%08), solo devolverá el error %31 **"No existe el disco"** en AH, cuando el disco en cuestión no exista.

Ejemplo: se solicita la información del disco 1.

```
        mov     AH, %08   ; Obtener los parámetros del disco
        mov     DL, 1     ; Unidad de disco 0
        sys     %D        ; Indica que se realiza la lectura
```

Finalizada la interrupción completará los registros CH, CL y DH, con la cantidad de cilindros, cabezas, y sectores del disco (respectivamente) obteniendo esa información de la MV y en el registro AH dejará %00, indicando operación exitosa; en caso contrario los registros CH. CL, DH no se modifican y en AH dejará %31 indicando que no existe el disco.

Cuando se solicita escribir o leer ***N*** sectores y la cantidad de sectores supera a las del cilindro/pista, continua por el sector 0 de la siguiente cabeza (se evita mover los cabezales a otro cilindro); así mismo si se supera la cantidad de cabezas, continuará leyendo desde el siguiente cilindro con la cabeza número 0 y sector 0.

Si una lectura supera el tamaño del disco, se especificará la cantidad de sectores transferidos en el **registro AL** y se considerará que la operación fue exitosa cargando en el registro AH el valor **%00 Operación exitosa.** Si se intenta leer un sector inexistente en el archivo ****.vdd***, pero dentro del rango de los parámetros del disco, se devuelve 0 y se extiende el archivo.

Si una escritura supera el tamaño del disco, se cargará el registro AH con el error **%CC "Error de transferencia en escritura".** Ej: Si el disco tiene 128 cilindros, 4 cabezas y 64 sectores, si se escriben 2 sectores a partir del cilindro 127, cabeza 3 sector 63, el primer sector se puede escribir, pero el siguiente no existe, y por lo tanto excede la capacidad del disco.

Para cualquier error arrojado por el sistema real al intentar leer o escribir, se utiliza el código %FF.

Gestión dinámica de segmentos

Se tiene la capacidad de manejar los segmentos en forma dinámica por medio de la llamada al sistema %E, La cual permite tanto crear un nuevo segmento, eliminar un segmento o consultar la información de un segmento.

La interrupción requiere el seteo de los siguientes registros/subregistros:

reg	*Parámetros de entrada*	*Resultados de salida*

AX	Se indica el número de operación:
	`%0000 Consultar segmento`
	`%0001 Crear un segmento`
		Código de estado
		`%0000 Operación exitosa`
		`%0001 Función inválida (parám AX)`
		`%0031 No existe el segmento`
		`%00CC No hay suficiente memoria`
		`%FFFF Falla en la operación (-1)`

CX	Se indica el tamaño en bytes del segmento	Tamaño en bytes del segmento (caso de %00)
EBX	Se indica el puntero a la primera celda del segmento.	Devuelve el puntero a la primera celda del segmento.

Consultar los datos de un segmento

Para consultar los datos de un segmento se debe conocer el puntero a la primera celda setearlo en EBX y en AX el código de operación %0000.

Si el segmento existe, rellenará en CX el tamaño del mismo y dejará en AX %0000 (operación exitosa)
Si el segmento no existe, en CX pondrá 0. En AX pondrá %31 “No existe el segmento” .

Crear un nuevo segmento

Para crear un nuevo segmento se debe indicar el tamaño en bytes del mismo en CX, y el código de operación %0001 en AX.

Ejemplo:

```
mov      AX, %01   ; Op. Crear segmento
mov      CX, 1024  ; 1024 bytes
sys      %E        ; Invocación
; en EBX se recibe el puntero al nuevo segmento
```

Si existe memoria disponible para crear el segmento, y tiene lugar en la tabla de segmentos, la MV crea la entrada en la tabla de segmentos y ubica el segmento a continuación del último segmento.
Si no existe memoria suficiente, en EBX devolverá -1 (null) y en AH el código será %00CC.
Si no hay espacio en la tabla de segmentos devolverá en EBX -1 y en AX %FFFF (-1).