Trabajo Practico Especial  
Arquitectura de las Computadoras

Facundo Gonzalez Fernandez 55746  
Alexis Moragues 55038 (Participo el cuatrimestre anterior)

**Consigna:**  
 Crear un “Sistema Operativo” de 64 bits con una clara separación entre kernel­space  
y user­space sin el uso de permisos. Se podrá utilizar una terminal del tipo shell con acceso  
a la pantalla, teclado, timerTick, PcSpeaker y poder cambiar a modo video manteniendo la separación previamente mencionada

**SysCalls:**  
 Las systemCalls implementadas son:  
● READ:​ RAX = 1  
●WRITE:​ RAX = 2  
●SLOW\_WRITE:​ RAX = 3  
●PLAY\_SOUND:​ RAX = 42  
●PLAY\_SONG:​ RAX = 9  
●DEATH:​ RAX = 4  
●COLOUR\_SWAP:​ RAX = 6

●VIDEO: RAX = 10

**Parámetros:**  
●READ, WRITE, SLOW WRITE,PLAY\_SONG: Char\* buffer, int size  
●COLOUR\_SWAP, KILL, PLAY\_SOUND, VIDEO: No utilizan parametros

**Implementacion:**

●READ:​ Recibe un Char\* a cambiar. Luego queda colgado en un while mantenido por  
el keyFlag. El keyFlag es un flag que se desactiva al oprimir una tecla del teclado y  
se activa al soltarla o al tocar una tecla invalida. Cuando se toca una tecla que  
desactive el keyFlag y salga del while, se guardara el char correspondiente a esta  
flecha en un buffer de teclado del kernel a través de la interrupcion de teclado. INT  
80Handler usa el buffer con la ultima tecla presionada y lo pone en el buffer puesto.  
Como va de una tecla a la vez siempre recibe el size 1. Sin embargo  
dependiendo de la tecla presionada se toman las medidas correspondientes(vease  
enter).  
  
● WRITE:​ Recibe un Char\* que imprime y que tan largo es el string. Luego itera sobre  
el buffer dado, imprimiendo char por char hasta terminar el size  
  
●SLOW\_WRITE: ​ Igual que WRITE pero espera un tiempo entre letra y letra  
  
●PLAY\_SOUND: ​ Lee el buffer de teclado constantemente tocando la nota  
correspondiente a la tecla oprimida, hasta que modifique el flag pianoKey. Una vez  
hecho terminará.  
● PLAY\_SONG: ​ Recibe un string desde un archivo con un formato específico del cual lee. Dependiendo del carácter tocará una nota o esperará un tiempo.  
  
●DEATH:​ Desactiva las interrupciones y cuelga la terminal  
  
●COLOUR\_SWAP:​ Cambia de color la terminal

●VIDEO: Se cambia el modo a modo video con una resolusion de 1024x768 y una profundidad de bits por pixel de 32. Luego comienza a imprimir un fractal en pantalla. El mismo recibe como parametro la cantidad de iteraciones que realizara, y lo recibe desde un archivo.

**Keyboard Driver**  
 Se realiza la interrupción de hardware por teclado, el PIC advierte la misma y manda  
a correr la rutina de atención 0x21. La cual entra al driver de teclado y funciona de la  
siguiente manera.  
 El keyboard posee un buffer de una sola tecla, el cual se va actualizando cada vez  
que uno ingresa una nueva. Leyendo el puerto 0x64, detecta que una tecla fue presionada,  
y luego desde el puerto 0x60 se recibe el scan code de la tecla presionada. Se busca la  
tecla correspondiente de este scan code dentro de un vector que las mapea. Una vez conseguida la misma, entra a una funcion que decide que hacer con cada caracter. En el caso del backspace y el enter, borra un carácter o va a la siguiente línearespectivamente.  
 En caso de estar en modo piano, funciona utilizando un switch, que en cada case o se toca la nota correspondiente,se sube o baja una octava.

***Fuentes Consultadas:***  
http://wiki.osdev.org/PS/2\_Keyboard  
<http://wiki.osdev.org/Interrupts>

**Pc Speaker**  
 Para utilizar el pcSpeaker primero tuvimos que modificar el “./run.sh” de manera  
acorde para que se habilite el mismo para su uso en el qemu. Luego lo manipulamos  
atraves de el PIT (Programable Interval Timer). Primero seteamos la onda que se va a  
transmitir por el pcSpeaker modificando el puerto 0x43, seteando la onda en squarewave y  
el canal 2 del PIT(pcspkr). Una vez seteado, le pasamos la frequencia deseada a el puerto  
0x42 donde se encuentra el pcSpeaker. Se hace una cuenta para que al manipular la  
frequencia, el parametro a pasar sea el numero de hertz. Una vez preparado todo se  
prenden los 2 bits menos significativos del puerto 0x63 que se encarga de prender el  
pcSpeaker. A la hora de apagarlo se ponen dichos bits en 0.

***Fuentes Consultadas:***  
http://wiki.osdev.org/Sound (Las funciones playSound, noSound utilizan codigo directo de  
este link)  
<http://wiki.osdev.org/Interrupts>

**Modo Video**

Para utilizar el modo video en primer lugar, se tuvo que modificar el archivo “./run.sh” para poder usar la extensión que emula el mismo. A partir de ahí, se escribe en los distintos registros para, en primer lugar, deshabilitar la extensión VBE, esto permite modificar los registros 1-3 que se encargan de la resolución y de la profundidad de bits. También se habilita el Linear Frame Buffer y el Clear Display Memory (Settea todos los bytes en 0). Luego, para poder escribir en la nueva pantalla, se debe conseguir la dirección de comienzo del Linear Frame Buffer desde el PIC. Y finalmente para pintar los pixeles, se creo una función que recibe valores de blue, green y red, cada uno ocupando 1 byte. Y la posición x e y que sea desea pintar.

***Fuentes Consultadas:***

<http://wiki.osdev.org/Bochs_VBE_Extensions>

<http://lodev.org/cgtutor/juliamandelbrot.html> (Codigo del fractal de Julia)

**Bugs**  
 El único bug notorio que se pudo observar, fue que presionar al mismo tiempo dos teclas en el teclado, haría que el shell muestre correctamente las dos teclas presionadas, pero el buffer de teclado almacene solo una. Generando errores de input de la shell.

Manual de Usuario  
Para ingresar un comando escribir el nombre y luego apretar la tecla enter.  
● help​ : Imprime en pantalla las funciones de la consola  
● piano:​ Activa el modo piano, (A=Do;S=Re;D=Mi;F=Fa;G=Sol;H=La;J=Si). Para salir  
presionar e.  
● music:​ Reproduce un archivo de sonido  
● badtime:​ Huevo de pascua  
● ­c:​ Cambia el color de la ventana

●video: Cambia a modo video e imprime un fractal en pantalla  
● close:​ Cierra la consola