Arquitectura de Computadoras

Trabajo Práctico Especial

Informe

Instituto Tecnológico de Buenos Aires

Grupo 04

Integrantes:

- Andrioli, Mateo (64042)
- Giambelluca, Santino (64697)
- Navarro, Alan Gabriel (63330)

Fecha de Entrega: 5 de Noviembre de 2025

Índice:

Índice:..... 3
Objetivo.....3
Separación User Space - Kernel Space.....5
System Calls.....5
Manejo de Interrupciones..... 6
Drivers.....6
    Driver de Video.....6
    Driver de Teclado.....7
    Driver de Sonido.....8
Excepciones..... 8
Shell..... 8
Juego - Tron..... 9

Objetivo

El objetivo de este proyecto fue implementar una parte de un Kernel, el cual es booteable mediante software-loader "*Pure 64*", para que administre los recursos de Hardware de una computadora. Además, se le debía brindar a los usuarios una API para que sea posible hacer uso de dichos recursos desde el *User Space*.

Para lograr esto, se debió separar la memoria en una sección destinada a Kernel (*Kernel Space*) y otra para futuros usuarios (*User Space*). La primera interactúa con el hardware mediante el uso de drivers que faciliten la comunicación con los periféricos (pantalla, teclado, etc.). En cambio, la segunda se comunica con estos utilizando las *system calls* brindadas por la primera.

Para poder interactuar de forma adecuada con el sistema y operar con el, se implementó una Shell para así poder interpretar comandos desde una terminal. Dichos comandos implementados hasta el momento son:

- Función de ayuda (help).
- Comando para limpiar la pantalla.
- Comando para desplegar la hora del sistema.
- Comando para desplegar la fecha del sistema.
- Comando para obtener los valores de los registros .
- Funciones que permiten testear la excepción de división por 0 y de código de operación invalido.
- Comando de sonido.
- Comandos para aumentar y disminuir el tamaño del texto de pantalla .
- Comando para jugar al Tron.
- Comando para mostrar las benchmarks.

Separación User Space - Kernel Space

El núcleo o Kernel tiene como objetivo gestionar de manera eficiente los recursos del sistema informático y proporcionar al usuario interfaces para utilizarlos. Dicho de otro modo, el Kernel controla exclusivamente el acceso al hardware y genera una interfaz de programación (API) destinada al usuario, impidiendo que este interactúe directamente con los componentes físicos.

Esta interfaz está compuesta por múltiples llamadas al sistema (System Calls), ubicadas en el espacio del Kernel. Los usuarios pueden acceder a ellas mediante la instrucción "INT 80h", la cual acepta diversos parámetros, incluyendo el "File Descriptor" que especifica qué llamada al sistema se desea ejecutar.

Con dicha base, se crearon diversas librerías que facilitan el uso del hardware para el usuario. Dentro de las cuales se encuentran el driver de teclado, el driver de video, y un driver de sonido, las cuales se explican más adelante.

System Calls

Para las *System Calls*, lo primero que se hizo fue cargar la interrupción en la IDT en la entrada 80h, apuntando a la rutina de atención de interrupción encargada de ejecutar la *System Call* correspondiente. Se desarrollaron las siguientes *System Calls:*

Syscall	Descripción
sys_write	Escribe en pantalla N caracteres de un buffer dado
sys_read	Lee y copia N caracteres del buffer de teclado a un buffer dado
sys_clear	Limpia la pantalla
sys_registers	Copia los valores de los registros
sys_ticks	Devuelve la cantidad de ticks que pasan en un cierto periodo de tiempo
sys_beep	Hace que suene un beep por un periodo de tiempo
sys_speaker_start / sys_speaker_off	Una manda una frecuencia a ejecutar con el parlante de la computadora, la otra apaga el sonido
sys_increase/decrease_fontsize	Incrementa / decrementa el tamaño del texto en la shell
sys_time	Imprime la hora local
sys_date	Imprime la fecha local
sys_screen_height/width	Devuelven el valor de el ancho y el alto de la pantalla
sys_putpixel	Cambia el color de un pixel determinado
sys_fill_rect	Rellena un rectángulo de altura h y ancho w, con un color dado y en una posición indicada.

Manejo de Interrupciones

Para las interrupciones, además del manejo del Timer Tick ya dado por la cátedra en el IDT, se implementó las interrupciones de teclado. Cargando la entrada de la misma en la zona de memoria 21h de la IDT. El valor que establecimos para enmascarar dicha tabla es FCh, ya que con esta queda únicamente habilitado el Timer Tick y la interrupción de teclado.

A raíz del timer tick, decidimos también implementar la función sleep, la cual “frena el código” hasta que no pase una cierta cantidad de tiempo determinada. Para lograr esto, se saca provecho de las interrupciones del timer tick

Drivers

Driver de Video

Para comenzar con la implementación del driver de video se necesitó encontrar una manera de imprimir en pantalla caracteres, emulando la funcionalidad de la naive console presente en el proyecto BareBones.

Se partió del driver de video proporcionado por la catedra y se utilizo (https://github.com/hubenchang0515/font8x16/blob/master/font8x16.h) en su tamaños 8x16 a arreglos de caracteres procesables de manera sencilla desde el codigo del driver

Por defecto al convertir la fuente con la aplicación se genera una lista de *defines* que permite que la fuente sea legible y fácilmente modificable desde un editor de texto. Cada char en este arreglo presenta bit a bit la presencia o ausencia de un píxel de una línea de 8 píxeles de ancho. Recorriendo estos arreglos (y haciendo los ajustes que correspondan al cambiar el tamaño) se puede dibujar pixel a pixel cada carácter ASCII necesario para mostrar texto. Se guarda además una posición en x e y del cursor para poder realizar la impresión normalmente, además de casos especiales para caracteres como el newline.

También se implementó una función scroll en el driver de video para mover una línea hacia arriba los contenidos de la pantalla por si se llegara al final de la pantalla.

Driver de Teclado

Para el manejo de las interrupciones de teclado, se decidió implementar una matriz kdb_manager tomando las filas como los valores de las teclas y dos columnas, una con la representación directa y otra en caso de que el *Shift* o *Caps Lock* estuviera accionado, para así contemplar el uso de las letras mayúsculas y algunos caracteres especiales como el + o - .

En lo que respecta al manejo de memoria con la entrada del teclado, el driver funciona utilizando un sistema de First In First Out (FIFO). Para ello cada caracter que leía, se encolaba en este buffer estático de manera circular. Esto significa que si el mismo se llena, en vez de allocar más memoria, pisa lo escrito en el principio.

Cada carácter, además, era procesado a través de una función para así contemplar el uso de algunas teclas especiales. Uno de estos ejemplos es el uso del *Control Izquierdo*, que al momento de ser presionado guarda los valores de los registros en el sistema. En esta misma función también se contempla el uso del *Shift* y el *Caps Lock* para el uso de las mayúsculas, como también el código que manda el teclado al sistema cuando una tecla deja de ser presionada, ya que no queremos que ese código de quiebre aparezca en pantalla.

Driver de Sonido

El audio del SO usa el parlante de PC controlado por el PIT: el kernel ajusta la frecuencia y enciende/apaga el speaker. A userland se expone con tres syscalls: sys_beep(freq, tiempo) para un tono “bloqueante” corto, y sys_speaker_start(freq) / sys_speaker_off() para control “no bloqueante”. La shell demuestra el beep con una pequeña secuencia de notas; en Tron, la música de fondo se maneja sin bloquear: el juego llama periódicamente a una función de “update” que, usando sys_ticks(), cambia a la siguiente nota o silencio en el momento adecuado. Así se logran efectos rápidos y música simple sin frenar la entrada ni la lógica del programa.

Excepciones

Para el manejo de excepciones, se implementaron y contemplaron dos casos según lo pedido por la cátedra, en el caso que el sistema haga una división por cero o por un código de operación inválida. Ante cualquiera de los dos casos, se imprime desde la shell un mensaje explicando el error, en conjunto con la impresión de los valores de cada uno de los registros del procesador.

Shell

La shell del sistema se ejecuta en un bucle permanente que muestra un prompt, captura la entrada del usuario y despacha la acción correspondiente. La entrada y salida se gestionan mediante wrappers de syscalls: la lectura se hace con sys_read de forma no bloqueante y la impresión con sys_write a través de utilidades como shellPrintString y shellPutchar.

Mientras se ingresa la línea, la shell dibuja un cursor parpadeante y soporta retroceso para edición básica. Además, las teclas “+” y “-” funcionan como atajos en tiempo real: no esperan al Enter, sino que invocan sys_increase_fontsize o sys_decrease_fontsize y luego redibujan la pantalla usando un “redraw buffer” que registra todo lo impreso para poder reconstruirlo al cambiar el tamaño de fuente.

Cuando se presiona Enter, la línea completa se pasa a un despachador que compara el texto ingresado contra una tabla estática de comandos (Command {name, function}) y, al hallar coincidencia exacta, ejecuta la rutina asociada. Entre los comandos disponibles están help, clear, printTime, printDate, registers, testDiv0, invOp, playBeep, varios benchmarks (bmFPS, bmCPU, bmMEM, bmKEY) y tron para lanzar el juego.

Si la cadena no coincide con ninguno, la shell informa que el comando no existe y sugiere usar help para listar todas las opciones. Internamente, se usan implementaciones mínimas de strlen/strcmp y utilidades de conversión numérica acorde al contexto bare-metal, y todas las operaciones de I/O y tiempo (sys_ticks) atraviesan int 0x80 mediante los wrappers definidos en userland. En conjunto, esto proporciona un intérprete simple pero robusto, con edición básica, atajos inmediatos para la UI y un mecanismo de despacho claro y extensible.

Juego - Tron

Para el juego Tron se implementó una matriz de estado, que guarda si está ocupada la casilla.

Luego se implementó un menú de opciones, comandos y una “inteligencia” para el modo 1 jugador la cual se guía mediante snapshots de la matriz para tratar de no colisionar. Y en el modo 2 jugadores se habilita otras partes del teclado para jugar .