

Arquitectura de Computadoras

Trabajo Práctico Espacial

Manual de Usuario



GRUPO 6

Integrantes:

- Tadeo Gorganchian (65507)
- Maria del Pilar Resek (65528)
- Eduardo Tormakh (65155)

Fecha de entrega: 5 de Noviembre de 2025

Índice:

Requisitos.....	3
Acceso.....	3
Comandos.....	4
help.....	4
cls.....	5
print_time.....	5
print_date.....	6
print_saved_registers.....	6
play_tron.....	6
1 Jugador:.....	7
2 Jugadores:.....	7
test_invalid_opcode.....	8
test_divide_by_zero.....	8
play_song.....	9
benchmark_fps.....	9
benchmark_cpu.....	9
benchmark_memory.....	10
benchmark_keyboard.....	10
Incremento y decremento de fuente.....	10

Requisitos

Para ejecutar el kernel desarrollado sobre la arquitectura x64 BareBones, es necesario contar con un entorno Linux. Si se utiliza un equipo con sistema operativo Windows, se puede recurrir a WSL para disponer de dicho entorno. Además, es indispensable tener instalado el emulador QEMU y Docker, ya que ambos son necesarios para compilar y ejecutar el proyecto correctamente.

Acceso

Antes de ejecutar el sistema, primero hay que abrir una terminal en la carpeta raíz del proyecto. Desde allí se ejecutan, en este orden, los archivos `create.sh`, `compile.sh` y `run.sh`.

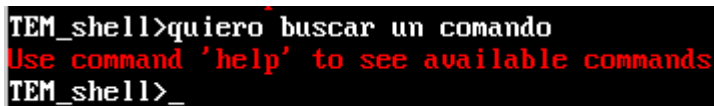
El script `create.sh` prepara el entorno de trabajo: crea un contenedor de Docker llamado `tpe_arqui_2q2025` y descarga la imagen que servirá como base para compilar el proyecto.

El archivo `compile.sh` se encarga de iniciar el contenedor, eliminar los archivos objeto generados en compilaciones anteriores, compilar nuevamente todo el código fuente y cerrar el entorno al finalizar.

Por último, `run.sh` ejecuta el emulador QEMU, cargando el sistema con los argumentos necesarios para su correcto arranque y funcionamiento.

Comandos

A través de la shell, el usuario dispone de una serie de comandos que puede ejecutar para realizar distintas operaciones. Para utilizarlos, debe escribir el nombre del comando respetando las mayúsculas y minúsculas, ya que el sistema distingue entre ambas (es *case sensitive*). En caso de ingresar un comando incorrecto, se mostrará un mensaje de error que indicará la posibilidad de escribir *“help”* para visualizar el menú de ayuda.



```
TEM_shell>quiero buscar un comando
Use command 'help' to see available commands
TEM_shell>_
```

Los comandos son:

- `help`
- `cls`
- `print_time`
- `print_date`
- `print_saved_registers`
- `play_tron`
- `test_invalid_opcode`
- `test_divide_by_zero`

- play_song
- benchmark_fps
- benchmark_cpu
- benchmark_memory
- benchmark_keyboard

help

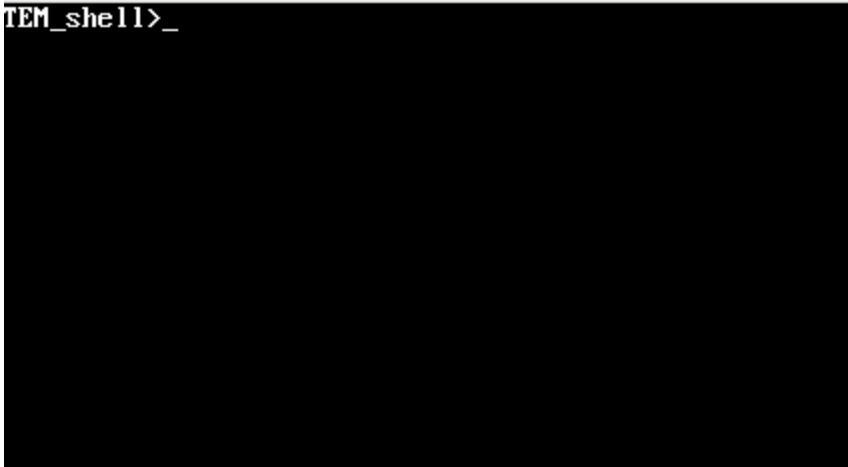
El comando help muestra en pantalla una lista con todos los comandos disponibles y su descripción. Al ejecutarlo, el usuario puede visualizar de forma clara las operaciones que el sistema permite realizar y su correspondiente utilidad. Al introducir este comando, se despliega en la consola el siguiente listado:

```
TEM_shell>help
To change font size press + or -
Available commands:
'help' : Shows available commands
'cls' : Clears the screen
'print_time' : Prints the current time (GMT-3)
'print_date' : Prints the current date (GMT-3)
'print_saved_registers' : Prints the CPU registers
'play_tron' : Plays the Tron game
'test_invalid_opcode' : Tests invalid opcode exception
'test_divide_by_zero' : Tests divide by zero exception
'play_song' : Plays a song
'benchmark_fps' : Benchmarks for average FPS
'benchmark_cpu' : Benchmarks for CPU performance
'benchmark_memory' : Benchmarks for memory accessing
'benchmark_keyboard' : Benchmarks for keyboard latency
```

cls

El comando cls limpia la pantalla de la terminal, eliminando todo el texto visible y dejando el cursor en la parte superior izquierda.

```
TEM_shell>_
```



print_time

Este comando permite consultar directamente desde la consola la hora vigente, mostrando la información según el horario correspondiente a Argentina (UTC-03:00).

```
TEM_shell>print_time  
16:42:46
```

print_date

Este comando permite consultar directamente desde la consola la fecha vigente, mostrando la información según el horario correspondiente a Argentina (UTC-03:00).

```
TEM_shell>print_date  
04/11/25
```

print_saved_registers

Para visualizar los registros, primero es necesario presionar la tecla “Ctrl”, ya que esta acción genera una captura del estado actual de los mismos. Si no se realiza este paso, el sistema mostrará un mensaje de error. Una vez realizada la captura, al ejecutar el comando print_saved_registers, se mostrarán en pantalla los valores almacenados de los registros correspondientes a ese momento.

```
TEM_shell>print_saved_registers  
No se han guardado registros  
Presione Ctrl para guardar un snapshot de los registros
```

```
TEM_shell>print_saved_registers  
RAX: 0x0000000000000000  
RBX: 0x0000000000000000  
RCX: 0x0000000000000000  
RDX: 0x0000000000000000  
RBP: 0x0000000000000000  
RDI: 0x0000000000000000  
RSI: 0x0000000000000000  
R8: 0x0000000000000000  
R9: 0x0000000000000001  
R10: 0x00000000010DF17  
R11: 0x00000000010DF18  
R12: 0x000000000000000A  
R13: 0x00000000010DF04  
R14: 0x0000000000000000  
R15: 0x0000000000000000  
RIP: 0x00000000040470F  
CS: 0x0000000000000008  
RFLAGS: 0x0000000000000246  
RSP: 0x00000000010DF00  
SS: 0x0000000000000000
```

play_tron

luego de ejecutar el comando play_tron se mostrará un menú de inicio con las siguientes opciones:

- 1 jugador
- 2 jugadores
- Presione Enter para salir del juego



1 Jugador:

En esta modalidad, el participante controla una única moto. El objetivo principal es avanzar a través de distintos niveles acumulando la mayor cantidad de puntos posible. El jugador utiliza las teclas de dirección (WASD) para desplazarse por el mapa, mientras evita colisiones con las paredes o su propia estela de luz. Durante la partida, en pantalla se muestra el puntaje obtenido y el nivel alcanzado. En caso de perder, el juego presenta una pantalla de “Game Over” que informa el puntaje final logrado.

2 Jugadores:

En este modo, dos jugadores compiten simultáneamente, cada uno controlando su propia moto. El jugador 1 emplea un conjunto de teclas (WASD) y el jugador 2 utiliza otro (IJKL). El objetivo consiste en mantenerse en juego por más tiempo que el oponente, evitando chocar contra los límites del escenario o la estela generada por cualquiera de las motos. Durante la partida, el sistema muestra el puntaje de ambos jugadores y, al finalizar, la pantalla de “Game Over” indica quién ha resultado ganador junto con los puntajes obtenidos.

test_invalid_opcode

Prueba el sistema de manejo de errores del sistema operativo generando un error especial llamado "instrucción inválida". A su vez, imprime el estado de los registros en el momento de la excepción. Sirve para demostrar cómo el sistema responde ante fallos graves de software.

```
RAX = 0000000000000000
R15 = 0000000000000000
R14 = 0000000000000000
R13 = 0000000000000000
R12 = 0000000000000000
R11 = 0000000000000000
R10 = 0000000000000000
R9 = 0000000000888888
R8 = 00000000004050DD
RSI = 000000000010DF64
RDI = 000000000010DF18
RBP = 0000000000000006
RDX = 000000000010DF24
RCX = 0000000000000000
RBX = 0000000000000000
RIP = 00000000004047AB
CS = 0000000000000008
RFLAGS = 000000000000206
RSP = 000000000010DF10
SS = 0000000000000000
Invalid Opcode Exception
```

test_divide_by_zero

Prueba el sistema de manejo de errores generando una división por cero, que es un error matemático. Tal como la excepción por "instrucción inválida", imprime el estado de los registros en el momento de la excepción. Por su parte, el sistema operativo detecta y muestra un mensaje de error.

```
RAX = 0000000000000000
R15 = 0000000000000000
R14 = 0000000000000000
R13 = 0000000000000000
R12 = 0000000000000000
R11 = 0000000000000000
R10 = 0000000000000000
R9 = 0000000000888888
R8 = 0000000000405117
RSI = 000000000010DF7C
RDI = 000000000010DF30
RBP = 0000000000000000
RDX = 000000000010DF3C
RCX = 0000000000000000
RBX = 0000000000000001
RIP = 0000000000400207
CS = 0000000000000008
RFLAGS = 000000000000202
RSP = 000000000010DF20
SS = 0000000000000000
Zero Division Exception
```

play_song

Reproduce una melodía usando el parlante interno de la computadora.

```
TEM_shell>play_song  
_Playing Y Dale Dale Boca
```

benchmark_fps

Mide cuántas veces por segundo la computadora puede actualizar la pantalla (frames por segundo). Es útil para saber qué tan rápido puede mostrar gráficos el sistema.

```
Frame rendering test...  
FPS promedio: 35  
TEM_shell>_
```

benchmark_cpu

Mide la velocidad del procesador realizando varios cálculos y muestra cuánto tiempo tarda. Sirve para comparar el rendimiento del CPU.

```
TEM_shell>benchmark_cpu  
=== Benchmark CPU ===  
Midiendo tiempo de procesamiento...  
Tiempo de procesamiento: 91 ticks  
Operaciones por tick: 549450  
TEM_shell>_
```

benchmark_memory

Mide qué tan rápido la computadora puede leer y escribir datos en la memoria. Es útil para saber si el sistema responde rápido al trabajar con grandes cantidades de información.

```
TEM_shell>benchmark_memory  
=== Benchmark Memoria ===  
Midiendo velocidad de acceso a memoria...  
Tiempo de acceso a memoria: 10 ticks  
Operaciones por tick: 12288000
```

benchmark_keyboard

Mide el tiempo que tarda el sistema en detectar que presionaste una tecla. Es una prueba de la rapidez de respuesta del teclado.

```
TEM_shell>benchmark_keyboard  
=== Benchmark Teclado ===  
Presiona cualquier tecla para medir latencia...  
  
Tiempo de respuesta: 31 ticks  
TEM_shell>_
```


Incremento y decremento de fuente

Este comando brinda la posibilidad de modificar el tamaño de la fuente utilizada en pantalla. Al presionar la tecla “+”, el texto aumenta su tamaño, facilitando la lectura, mientras que al presionar la tecla “-”, la fuente se reduce, permitiendo visualizar una mayor cantidad de información dentro del mismo espacio.