

Arquitecturas de Computadoras
Segundo Cuatrimestre 2017



Trabajo Práctico Especial

Fecha de Entrega 14/11/17

Profesores:

Vallés, Santiago

Merovich, Horacio

Integrantes:

Della Sala, Rocío 56507

Santoflaminio, Alejandro 57042

Tay, Lucía 56244

Introducción

El objetivo del trabajo es implementar un kernel booteable por Pure64, a partir de uno provisto por la cátedra para luego modificarlo y agregar distintas funcionalidades.

Se definieron dos espacios separados en el mismo: kernel space y user space. En el kernel se encuentran los drivers que interactúan con el hardware, como el driver de teclado y el de video. En el user space se encuentran distintos módulos para acceder a las distintas funcionalidades. Se decidió optar por módulos en el user space para emproljar el contenido y facilitar el acceso a los códigos de cada una de las funcionalidades a realizar. Se encuentran en el trabajo entregado distintos módulos: Módulos para los gráficos, módulo para la Shell, módulos para probar las excepciones y un módulo para obtener el tiempo local.

Con respecto a las interrupciones de hardware implementadas, en el trabajo se pueden ver:

- Interrupciones de teclado
- Interrupciones de timer tick

Para la interrupción de teclado, mediante el driver se pueden leer las teclas presionadas, interpretarlas e imprimirlas en pantalla.

El Timer Tick se encarga de interrumpir al sistema operativo. Usando el Timer Tick se programó la función sleep que duerme la consola por un período de tiempo.

Por otro lado también tenemos un driver de video¹. Mediante el mismo podemos imprimir en la pantalla pixeles o directamente strings y chars. El modo video se configuró con una resolución 1024x768.

Cabe destacar que todo el sistema se ejecuta en modo vídeo para evitar tener que alternar entre modo texto y modo vídeo.

Funcionalidades

Al iniciar al sistema se observa una pantalla que muestra cuatro opciones (ver figura 1): Shell (ingresa a la consola de comandos), Time (muestra la hora local), Linear Graph (grafica una función lineal) y Parabolic Graph (grafica una función parabólica).

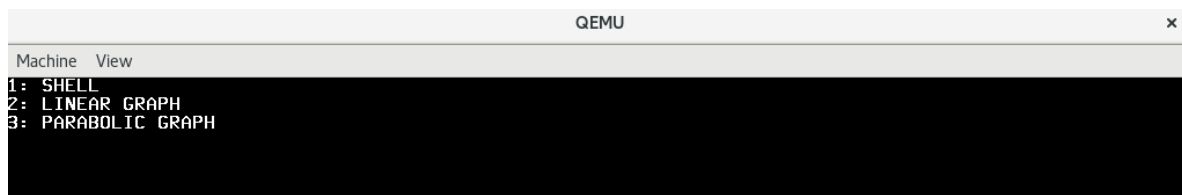
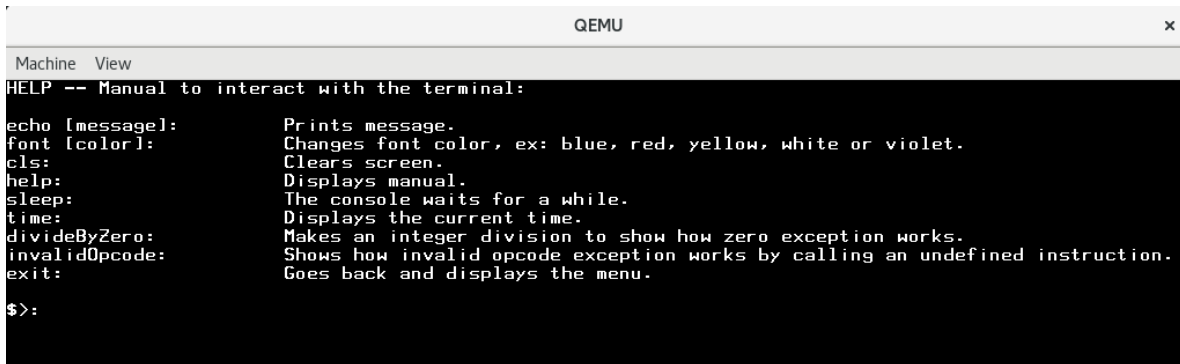


Figura 1: Menú principal

¹ Se consultaron los siguientes links: wiki.osdev.org/VESA_Video_Modes y wiki.osdev.org/Drawing_In_Protected_Mode

Si se selecciona la opción shell se ingresa a una consola de comandos. Escribiendo help seguido de la tecla enter se muestran todos los posibles comandos a ejecutar.

A screenshot of a QEMU terminal window. The window has a title bar with 'QEMU' and a close button. Below the title bar is a menu bar with 'Machine' and 'View'. The main area is a black terminal with white text. It shows the output of the 'help' command, which lists various commands and their functions. The prompt '\$>:' is visible at the bottom.

```
Machine View
HELP -- Manual to interact with the terminal:
echo [message]:      Prints message.
font [color]:        Changes font color, ex: blue, red, yellow, white or violet.
cls:                  Clears screen.
help:                 Displays manual.
sleep:                The console waits for a while.
time:                 Displays the current time.
divideByZero:         Makes an integer division to show how zero exception works.
invalidOpcode:        Shows how invalid opcode exception works by calling an undefined instruction.
exit:                 Goes back and displays the menu.
$>:
```

Figura 2: Comando help en la shell

Con el comando time por ejemplo se muestra directamente la hora local.

Si se ingresa a alguno de los módulos para graficar debemos ingresar los valores de las constantes.

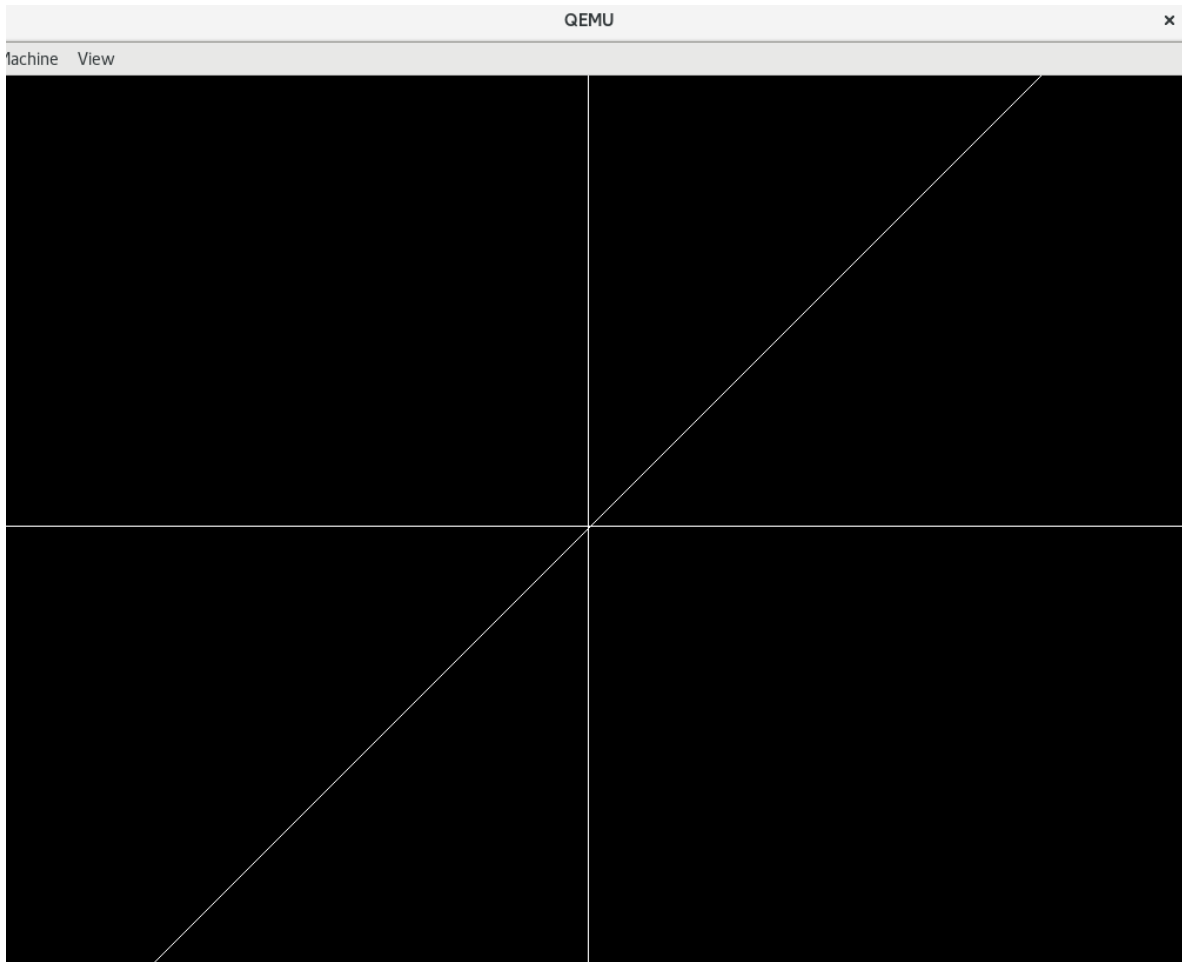


Figura 3: Gráfico lineal de $y = x$

Además el kernel es capaz de manejar tres tipos de excepciones: división por cero, overflow y código de operación inválido. Se pueden probar estas excepciones mediante el uso de la shell (ver figura 4), excepto la excepción por overflow, la cual se encuentra cargada en la IDT pero es posible probarla ya que no fue posible generar un código para lanzarla.

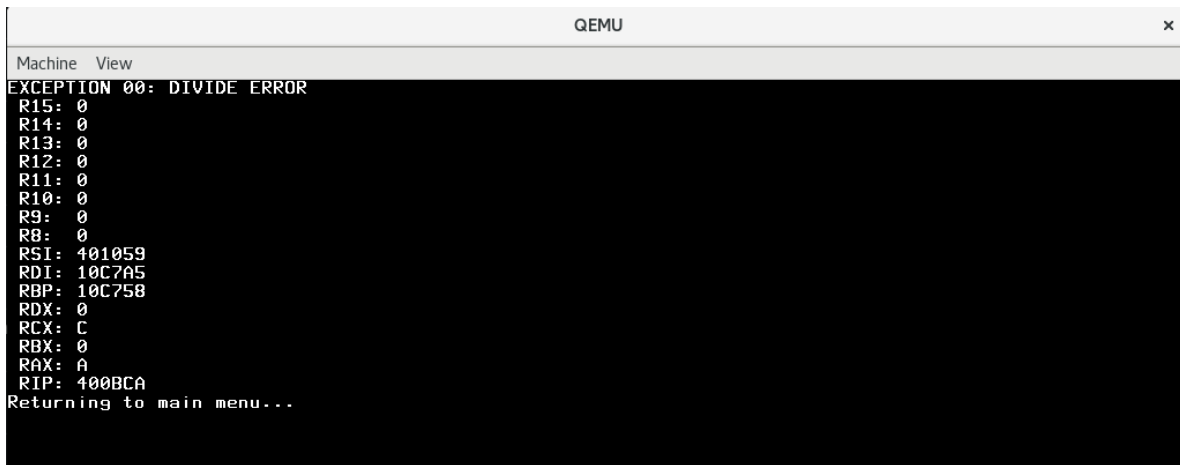


Figura 4: Excepción debido a división por cero

Pros y Contras

- ✓ No hay transición de modo texto a video ya que todo el sistema está hecho en modo video.
- ✓ Organización por módulos facilita lectura de los diferentes archivos.
- ✗ Los gráficos pueden verse alterados dependiendo de los valores, ya que puede que lleguen a quedar fuera de escala. Lo ideal sería en un futuro implementar una función que calcule la escala a utilizar dependiendo de los parámetros que ingrese el usuario.
- ✗ Al utilizar modo video fue necesario utilizar funciones para imprimir letras y pixeles en pantalla.
- ✗ No fue posible probar la excepción por overflow aunque ésta si se encuentra cargada en la IDT.

ANEXO: API de System Calls a los que responde el sistema operativo mediante int 80Write:

Escribe en el siguiente carácter en el descriptor del archivo.

Registros	RSI	RCX	RDX	RAX
Parámetros	Buffer	Size	-	-1 si hay error

Read:

Lee del descriptor del archivo y lo pone en un buffer.

Registros	RSI	RCX	RDX	RAX
Parámetros	File	Buffer	-	Cantidad de bytes leídos.

Clear:

Limpia toda pantalla usada por el modulo (los caracteres que hay). Es decir los reglones reservados para el sistema operativo no los toca

Registros	RSI	RCX	RDX	RAX
Parámetros	-	-	-	-

fontColor:

Cambia el color de lo que se imprime en pantalla.

Registros	RSI	RCX	RDX	RAX
Parámetros	Color	-	-	-

Sleep:

Duerme la consola por un tiempo.

Registros	RSI	RCX	RDX	RAX
Parámetros	Time	-	-	-

Delete:

Permite borrar el último carácter escrito en la consola.

Registros	RSI	RBX	RCX	RDX
Parámetros	-	-	-	-

nextLine:

Salto a la siguiente línea.

Registros	RSI	RBX	RCX	RDX
Parámetros	-	-	-	-

Pixel:

Dibuja un pixel en la pantalla.

Registros	RSI	RBX	RCX	RDX
Parámetros	x	y	-	-