Instituto Tecnológico de Buenos Aires

Trabajo Práctico Especial

Manual de usuario

Perez de Gracia, Mateo
Quian Blanco, Francisco
Stanfield, Theo
Ves Losada, Tobías

Arquitectura de Computadoras - 72.08

Primer cuatrimestre 2023

1 Requisitos

Los requisitos necesarios para correr el proyecto son los siguientes:

- `docker`: para emular la compilación en una distribución de Ubuntu/Debian
- `qemu`: permite tomar la imagen compilada y emular un sistema

Desde *Arch Linux* el procedimiento de instalación es sencillo:

```
sudo pacman -S qemu-system-x86 docker
```

2 Preparando el entorno

Una vez instalados los requisitos se deben realizar unas tareas antes de poder compilar el proyecto.

2.1 Agregar mi usuario al grupo de docker

Esto permite correr los contenedores, sin necesidad de tener privilegios de administrador.

```
sudo groupadd docker
sudo usermod -aG docker $USER
```

2.2 Iniciar el daemon de docker

Con permisos de administrador correr el siguiente comando:

```
sudo setsid -f dockerd >/dev/null 2>&1
```

Con docker andando ya se pueden correr todos los comandos relacionados con docker. El comando `./compile.sh` permitirá realizar el resto del trabajo.

3 Compilando

Listo! El entorno ya está preparado para compilar el proyecto y correrlo. El comando `./compile.sh` realizará las siguientes tareas:

- Descargará la imágen de `agodio/itba-so:1.0` de docker donde se compilará el proyecto.
- Creará un nuevo contenedor preparado para compilar el proyecto como si fuéramos nosotros porque crea un usuario con nuestro mismo **uid** y **gid**.
- Iniciará el contenedor para que siempre esté corriendo.
- Enviará los comandos necesarios para limpiar y compilar el proyecto completo.

Esa es la funcionalidad base del comando `./compile.sh`. A su vez, cuenta con dos flags que pueden ser enviados:

- `-d`: Compila el proyecto con información de debugging para poder utilizar gdb y ejecutar paso a paso el código.
- `-r`: Una vez compilado el proyecto completo, si no ocurrió ningún error durante el proceso, se ejecuta el comando `./run.sh`, es decir, una vez compilado el proyecto, lo ejecuta.

  Nota: Si se corre el comando con ambos flags en simultáneo, es decir, `./compile.sh -d -r`, el proyecto se compilará y ejecutará en modo debugging. Si solo se utiliza el flag `-r`, se compilará en modo normal y se ejecutará en modo normal, sin información de debugging.

4 Debugging

Para poder ejecutar el código paso a paso debe utilizarse el programa `gdb`.

Para esto hay un archivo de configuración `.gdbinit` que debe copiarse el home del usuario para poder configurarlo apropiadamente.

```
cp .gdbinit $HOME
```

Una vez realizado esto, para poder funcionarlo se utilizarán dos terminales:

1. En una se ejecutará el programa de compilado y ejecución: `./compile.sh -d -r`
2. En la otra se ejecutará el `gdb` que se conectará al `qemu` y podrá debuguearse el código.

El `gdb` estará configurado con dos subcomandos para facilitar el debugueo de código en **ASM** y en **C**:

- `src-prof` configura el `gdb` en modo de debugueo para código **C**.
- `asm-prof` lo mismo pero para código en **ASM**.

  Nota: Se puede intercalar en los modos en cualquier momento.

Una vez que está corriendo el `qemu` este se quedará esperando a que se conecte el `gdb`.

Para eso correr el comando `gdb`. Una vez abierto si se corre `c` o `continue` correrá el programa de forma normal.

Si se quiere hacer un debugueo *step by step* puede ponerse un breakpoint con `b` o `br` en alguna parte del código y ejecutar `c` para que salte hasta ese punto.

```
> b main
> c
```

Si quiere realizarse un debugueo también *step by step* desde el principio, una vez abierto el `gdb` comenzar a ejecutar `s` o `si` (no significan lo mismo **OJO**).

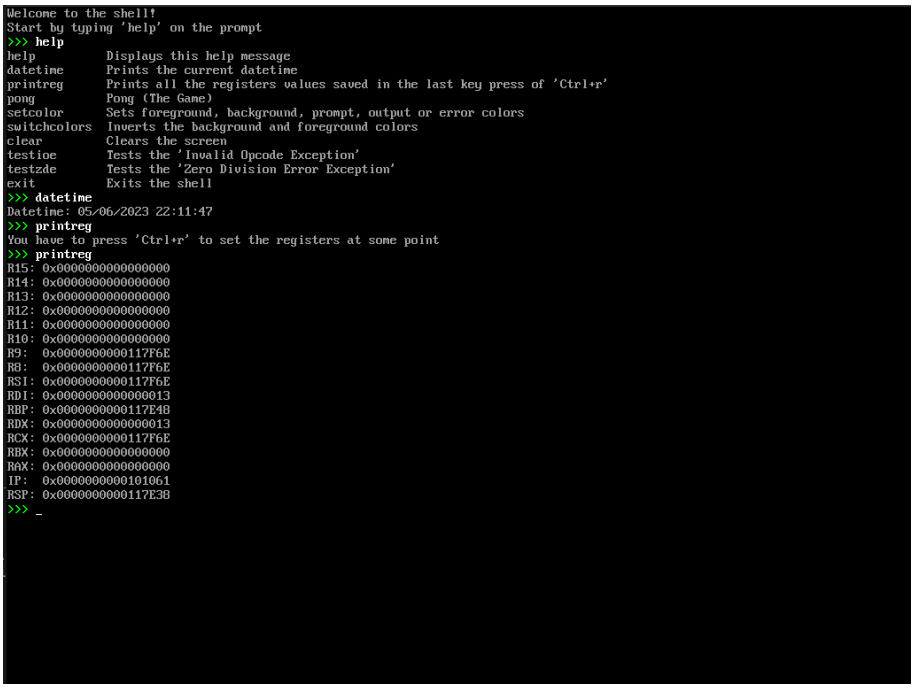
5 Ejecutando

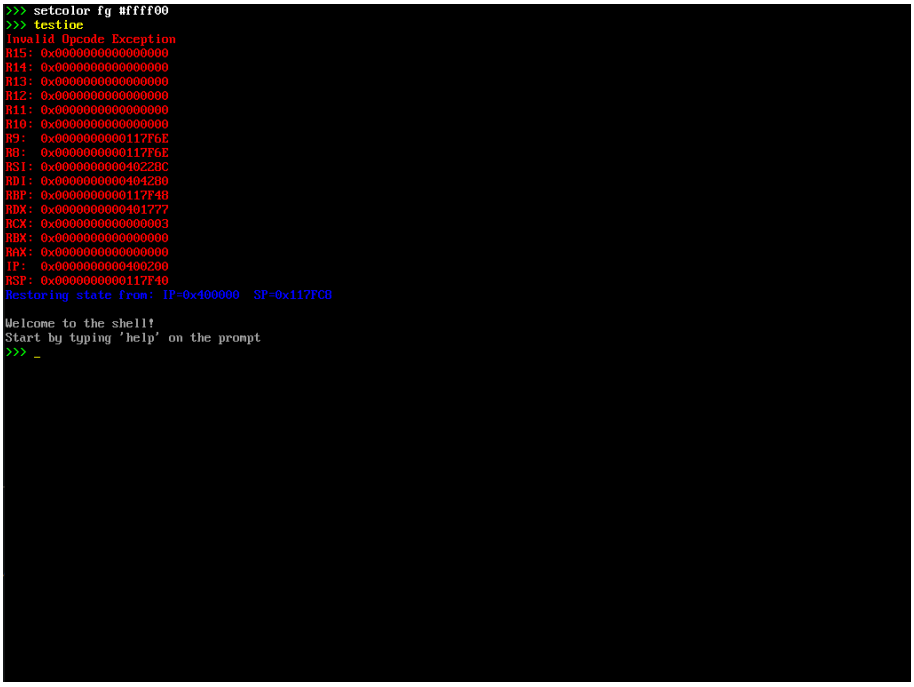
Ejecutando el comando run.sh, se abrira el QEMU y podremos observar la siguiente pantalla:

Unos segundos después, podremos observar la terminal de nuestro sistema operativo, con el mensaje de bienvenida y el prompt de la forma "`>>>`" con el cursor en el siguiente casillero. Este es el lugar donde se escribirán los comandos.

5.1 Comandos del shell

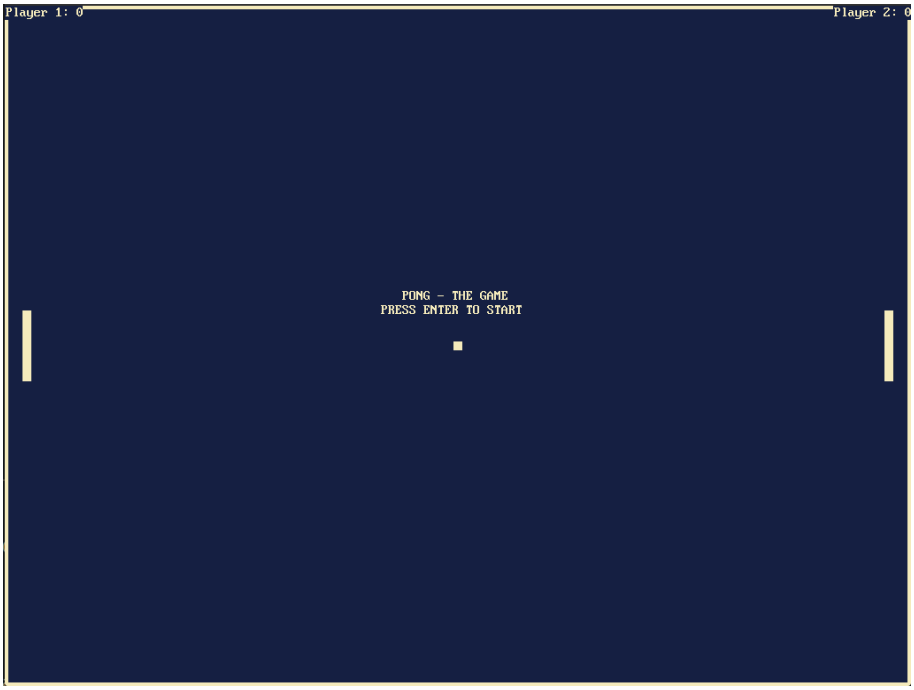
- `help`: Imprime por salida estándar un menú con los distintos comandos disponibles para el usuario.
- `datetime`: Imprime por salida estándar el horario local, en formato `DD/MM/AA HH:MM:SS`.
- `printreg`: Imprime los registros que se guardaron en la última pulsación de las teclas Ctrl+r.
- `pong`: Abre el juego "pong".
- `setcolor`: Permite especificar el color del fondo, de fuente, prompt,output o error recibiendo como primer argumento fg,bg,prompt,output o error respectivamente y como segundo argumento el color en hexadecimal en formato `#XXXXXX` o `0xXXXXXX`.
- `switchcolors`: Intercambia los colores de fuente y fondo entre sí.
- `clear`: Limpia la pantalla de la terminal.
- `testioe`: Desencadena una invalid operation code exception que imprime en pantalla el mensaje de error y el valor de los registros al momento que ocurrió la misma, incluidos entre ellos el Instruction Pointer.
- `testzde`: Desencadena una zero division exception que imprime por pantalla información del error como en la invalid operation code exception.
- `exit`: Abandona la terminal.





5.2 Pong

Una vez ejecutado el comando del pong obtendremos la siguiente pantalla:



Para comenzar a jugar se debe presionar la tecla **Enter**, e inmediatamente comenzará el juego. El jugador del lado derecho de la pantalla se moverá utilizando la tecla **I** (desplazamiento hacia arriba) y la tecla **J** (desplazamiento hacia abajo). En cambio, el jugador del lado izquierdo utiliza las teclas **W** y **S** (desplazamiento hacia arriba y abajo respectivamente). El objetivo del juego es conseguir más puntos que el rival, y estos puntos se consiguen cuando el rival no es capaz de devolver la pelota. El juego concluye cuando algún jugador llega a los 5 puntos, y al haber concluido el juego al apretar la tecla **Enter** se volverá a la terminal.

Es posible salir del juego en cualquier instante presionando la tecla Q