Guía de Instalación de QEMU

Técnicas Digitales (Piloto)

Cíclo Lectivo 2022

Curso R5054/R4054

Universidad Tecnológica Nacional - Facultad Regional Buenos Aires

Índice

1. Introducción

Para el desarrollo de la guía de trabajos prácticos sobre arquitectura ARMv7 a lo largo del año se requerirán de 2 herramientas:

1. Instalación de QEMU (y entorno de desarrollo): con este emulador comenzaremos a desarrollar el firmware para un Cortex-A8,

2. Placa de desarrollo Beaglebone Black: los 2 últimos TPs serán implementados en dicha placa utilizando una tarjeta SD.

Como premisa para el desarrollo de los T.P., en la materia se promueve el uso de GitLab como *framework* de trabajo.

2. Preparación del entorno de desarrollo

2.1. Herramientas de compilación para ARM

Para el la compilación de los fuentes y generación de binarios ejecutables es necesario descargar herramientas de compilación de ARM que nos permitan *cross-compilar*, es decir, que estándo trabajando en una arquitectura tipo x86 (sea de 32 bits o 64 bits), podamos generar código de máquina para una arquitectura diferente (en nuestro caso, ARM y en particular ARMv7). Para ello deben descargar de la siguiente página el toolchain de ARM; deben descargar el que es para **x86_64_arm** y sólo el de **gcc-linaro**. No descarguen la última versión disponible en la página de ARM Developer, dado que en el 2do cuatrimestre harán uso de este mismo toolchain para armar la imágen de Linux que necesitarán para desarrollar el trabajo práctico del 2do cuatrimestre.

El procedimiento de "instalación" es muy sencillo:

1. Descargan el paquete (comprimido),

2. Lo ubican en una zona de trabajo o lo dejan en la carpeta de descargas,

3. Lo descomprimen,

4. Luego desde consola se ubican en *home* de su usuario (es decir, si hacen "pwd" en la consola, deberían ver solamente: **/home/su_usuario**),

5. Allí ejecutan: vim .bashrc, (o pueden usar su editor de textos favorito),

6. Se posicionan al final del archivo (no modifiquen nada de lo existente), y deben colocar las siguientes líneas:

```
export PATH=$PATH:/path/a/su/carpeta/extraida/de/gcc-linaro-7.5.0-2019.12-x86_64_arm-linux-gnueabihf/bin/
```

7. Hacen desde el terminal (si es que **no abren una nueva**):

```
user@pc:folder$ source .bashrc
```

   Lo cual vuelve a cargar el bashrc en la sesión actual (cada vez que abren una terminal en linux, automáticamente se hace este *source*, y por eso si se modifica .bashrc hay que o cargarlo manualmente de nuevo o abrir otra terminal). Este paso debería salir sin errores; y en caso de haberlos quiere decir que la ruta especificada es errónea.

Como verán en realidad, no lo estamos instalando *per-se*, sino que solo referenciamos su uso. De esta manera no interferimos con el GCC de ARM que la distribución que estemos usando se vea afectada.

2.2. QEMU

Para instalar QEMU en su computadora personal, realice los siguientes pasos (utilice el gestor de paquetes con el que se encuentre familiarizado, a modo de guía aquí se utiliza *apt*):

```
user@pc:folder$ sudo apt install qemu-system-arm bison flex build-
    essential ddd gdb-multiarch
```

Finalizada la instalación de todos los paquetes realice la primer prueba:

```
user@pc:folder$ qemu-system-arm -help
```

Lo cual debería arrojar el menu de ayuda sin ningun tipo de error o librería faltante. Luego ejecuten la siguiente línea:

```
qemu-system-arm -M realview-pb-a8 -m 32M -no-reboot -nographic -monitor
    telnet:127.0.0.1:1234,server,nowait
```

Y no se asusten, pueden pasar 1 de 2 cosas:

1. Reciben el siguiente mensaje:

   ```
   qemu-system-arm: Trying to execute code outside RAM or ROM at 0
       x02000000
   ```

   Lo cual es totalmente esperable dado que no le pasamos un *kernel* para que ejecute,

2. Primero reciben en consola:

   ```
   pulseaudio: set_sink_input_volume() failed
   pulseaudio: Reason: Invalid argument
   pulseaudio: set_sink_input_mute() failed
   pulseaudio: Reason: Invalid argument
   ```

   Que si estaban escuchando música o la radio en la PC, puede que genere ruido en la salida de audio (asique ojo, no se asusten como me pasó a mi, no se rompe nada). Es un tema de incompatibilidad con drivers, pero QEMU está andando bien.

   A continuación solo tienen que presionar *ctrl-c* y ahí deberían recibir la línea del primer punto.

Si todo fue bien, pueden pasar a la siguiente sección.

2.3. GitLab

Para probar que todo "está bien", clone el siguiente repositorio:

```
user@pc:folder$ git clone git@gitlab.frba.utn.edu.ar:td_piloto/
    qemu_primer_test.git
```

Allí encontrará un archivo llamado *startup.s* el cual deberá compilar y ejecutar como se explica en la siguiente sección. En el mientras tanto, revise el archivo y *vea* que hace cada instrucción y de dónde viene esa prueba de "0xAA55AA55" (delen, son 3 líneas de código noma').

3. Ejecutando QEMU y debugging

Primero que nada, tenemos que compilar. Pero para ello vamos a generar varios tipos de archivos que nos serán muy útiles para poder entender que estamos haciendo y poder afrontar las dificultades del desarrollo de firmware. Si, podría haberles pasado el binario ya, pero primero que nada no tiene nada de gracia, y segundo, tenemos que ver que todo el toolchain de trabajo esté funcionando bien.

Asique, primero se meten en la carpeta que acaban de "clonar" y allí hacen:

```
user@pc:folder$ arm-none-eabi-as -o startup.o startup.s
```

Esto genera un objeto (como con C/C++). Luego:

```
user@pc:folder$ arm-none-eabi-ld -o first_test.elf startup.o
```

Acá ya tenemos el binario ejecutable. No se preocupen por el *warning* de "_start", eso es por que nuestro *startup.s* no tiene una "etiqueta" llamada así. Finalmente lo que tenemos que hacer es crear el **binario**:

```
user@pc:folder$ arm-none-eabi-objcopy -O binary first_test.elf
    first_test.bin
```

Ahora si, pasaremos a ejecturar QEMU en una terminal:

```
user@pc:folder$ qemu-system-arm -M realview-pb-a8 -m 32M -no-reboot -
    nographic -monitor telnet:127.0.0.1:1234,server,nowait -kernel
    first_test.bin
```

y en otra terminal, hacemos:

```
user@pc:folder$ telnet localhost 1234
```

Dónde deberíamos observar:

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
QEMU 2.11.1 monitor - type 'help' for more information
```

La cual es nuestra consola de debugging de QEMU y dónde podremos hacer varias consultas, como por ejemplo:

```
(qemu) info registers
00=00000000 R01=00000769 R02=aa55aa55 R03=00000000
R04=00000000 R05=00000000 R06=00000000 R07=00000000
R08=00000000 R09=00000000 R10=00000000 R11=00000000
R12=00000000 R13=00000000 R14=00000000 R15=70010004
.
.
.
```

Dónde vemos que R2 (uno de los registros de propósito general) quedó cargado con el valor que le habíamos puesto en el fuente. Para salir pueden simplemente tipear "quit" (lo cual también cierra a QEMU en la otra terminal).

Si hasta aquí anduvo todo bien, quiere decir que su PC está en condiciones poder encarar los TPs siguientes de la materia.