QEMU RISC-V

QEMU for RISC-V supports a special ‘virt’ machine and ‘spike’ machine designed for emulation and virtualization purposes. This document describes how to run U-Boot under it. Both 32-bit and 64-bit targets are supported, running in either machine or supervisor mode.

The QEMU virt machine models a generic RISC-V virtual machine with support for the VirtIO standard networking and block storage devices. It has CLINT, PLIC, 16550A UART devices in addition to VirtIO and it also uses device-tree to pass configuration information to guest software. It implements the latest RISC-V privileged architecture.

See Devicetree in QEMU for information on how to see the devicetree actually generated by QEMU.

The QEMU spike machine models a minimalistic RISC-V virtual machine with only CLINT and HTIF devices. It also uses device-tree to pass configuration information to guest software and implements the latest RISC-V privileged architecture.

Building U-Boot

Set the CROSS_COMPILE environment variable as usual, and run:

- For 32-bit RISC-V:

```
make qemu-riscv32_defconfig
make
```

- For 64-bit RISC-V:

```
make qemu-riscv64_defconfig
make
```


This will compile U-Boot for machine mode. To build supervisor mode binaries, use the configurations qemu-riscv32_smode_defconfig and qemu-riscv64_smode_defconfig

instead. Note that U-Boot running in supervisor mode requires a supervisor binary interface (SBI), such as RISC-V OpenSBI.

To create a U-Boot binary that can be utilized with a pflash device in QEMU apply these addtional settings to qemu-riscv64_smode_defconfig:

```
CONFIG_TEXT_BASE=0x20000000
CONFIG_XIP=y
# CONFIG_AVAILABLE_HARTS is not set
CONFIG_SYS_MONITOR_BASE=0x80200000
```

Truncate the resulting u-boot.bin to 32 MiB. Add the following to your qemu-system-riscv64 command:

```
-drive if=pflash,format=raw,unit=0,file=u-boot.bin
```

Running U-Boot

The minimal QEMU command line to get U-Boot up and running is:

- For 32-bit RISC-V virt machine:

```
qemu-system-riscv32 -nographic -machine virt -bios u-boot.bin
```

- For 64-bit RISC-V virt machine:

```
qemu-system-riscv64 -nographic -machine virt -bios u-boot.bin
```

- For 64-bit RISC-V spike machine:

```
qemu-system-riscv64 -nographic -machine spike -bios u-boot.bin
```

The commands above create targets with 128MiB memory by default. A fr… configurable amount of RAM can be created via the '-m' parameter. For ex… creates 2GiB memory for the target, and the memory node in the embedded DTB created

by QEMU reflects the new setting.

For instructions on how to run U-Boot in supervisor mode on QEMU with OpenSBI, see the documentation available with OpenSBI: https://github.com/riscv/opensbi/blob/master/docs/platform/qemu_virt.md https://github.com/riscv/opensbi/blob/master/docs/platform/spike.md

These have been tested in QEMU 5.0.0.

Running U-Boot SPL

In the default SPL configuration, U-Boot SPL starts in machine mode. U-Boot proper and OpenSBI (FW_DYNAMIC firmware) are bundled as FIT image and made available to U-Boot SPL. Both are then loaded by U-Boot SPL and the location of U-Boot proper is passed to OpenSBI. After initialization, U-Boot proper is started in supervisor mode by OpenSBI.

OpenSBI must be compiled before compiling U-Boot. Version 0.4 and higher is supported by U-Boot. Clone the OpenSBI repository and run the following command.

```
git clone https://github.com/riscv/opensbi.git
cd opensbi
make PLATFORM=generic
```

See the OpenSBI documentation for full details: https://github.com/riscv/opensbi/blob/master/docs/platform/qemu_virt.md https://github.com/riscv/opensbi/blob/master/docs/platform/spike.md

To make the FW_DYNAMIC binary (build/platform/generic/firmware/fw_dynamic.bin) available to U-Boot, either copy it into the U-Boot root directory or specify its location with the OPENSBI environment variable. Afterwards, compile U-Boot with the following commands.

- For 32-bit RISC-V:

```
make qemu-riscv32_spl_defconfig
make
```

- For 64-bit RISC-V:

```
make qemu-riscv64_spl_defconfig
make
```

The minimal QEMU commands to run U-Boot SPL in both 32-bit and 64-bit configurations are:

- For 32-bit RISC-V virt machine:

```
qemu-system-riscv32 -nographic -machine virt -bios spl/u-boot-spl.bin \
-device loader,file=u-boot.itb,addr=0x80200000
```

- For 64-bit RISC-V virt machine:

```
qemu-system-riscv64 -nographic -machine virt -bios spl/u-boot-spl.bin \
-device loader,file=u-boot.itb,addr=0x80200000
```

- For 64-bit RISC-V spike machine:

```
qemu-system-riscv64 -nographic -machine spike -bios spl/u-boot-spl.bin \
-device loader,file=u-boot.itb,addr=0x80200000
```

An attached disk can be emulated in RISC-V virt machine by adding:

```
-device ich9-ahci,id=ahci \
-drive if=none,file=riscv64.img,format=raw,id=mydisk \
-device ide-hd,drive=mydisk,bus=ahci.0
```

or alternatively attach an emulated UFS:

```
-device ufs,id=ufs0 \
-drive if=none,file=test.img,format=raw,id=lun0 \
-device ufs-lu,drive=lun0,bus=ufs0
```


You will have to run ‘scsi scan’ to use them.

A video console can be emulated in RISC-V virt machine by removing “-nographic” and adding:

```
-serial stdio -device VGA
```

In addition, a usb keyboard can be attached to an emulated xHCI controller in RISC-V virt machine as an option of input devices by adding:

```
-device qemu-xhci,id=xhci -device usb-kbd,bus=xhci.0
```

Running with KVM

Running with QEMU using KVM requires an S-mode U-Boot binary as created by qemu-riscv64_smode_defconfig.

Provide the U-Boot S-mode ELF image as *-kernel* parameter and do not add a *-bios* parameter, e.g.

```
qemu-system-riscv64 -accel kvm -nographic -machine virt -kernel u-boot
```

Debug UART

The following settings provide a debug UART for the virt machine:

```
CONFIG_DEBUG_UART=y
CONFIG_DEBUG_UART_NS16550=y
CONFIG_DEBUG_UART_BASE=0x10000000
CONFIG_DEBUG_UART_CLOCK=3686400
```

To provide a debug UART in main U-Boot the SBI DBCN extension can be used instead:

```
CONFIG_DEBUG_SBI_CONSOLE=y
```