Free RISC-V Systems
Benefits and Status of QEMU

Alistair Francis <alistair.francis@wdc.com>
Spring RISC-V 2022 Week
May 2022
What is QEMU?

Emulator

• QEMU is a very quick open source (mostly GPLv2) emulator and hypervisor

• It is not cycle accurate, but it is functionally accurate

• It uses the Tiny Code Generator (TCG) to translate different guest architecture instructions to host executable code
  – Supports full system (softMMU) emulation
  – Also supports just Linux/BSD user space translation

• Open source project, not written or maintained by a single company
What is QEMU?

Hypervisor

Host User Space

Guest User Space

Guest Kernel

VirtIO Frontends

Process (Guest0)

QEMU
(Guest IO Emulation + VirtIO Backends)

Device Drivers
(Host HW Access)

Host Linux Kernel

KVM Module
(CPU Virtualization)

Firmware (OpenSBI)

VU-mode

VS-mode

U-mode

HS-mode

M-mode
Basics of Tiny Code Generator (TCG)

- TCG began as a backend for a C compiler
- TCG can convert TCG ops to target (host) instructions
  - It also performs some optimisations and liveness analysis to improve performance
- TCG will combine blocks of guest code into a TB blocks
  - The end of a block occurs when a branch/jump instruction is encountered
- TCG currently natively supports these targets (hosts)
  - AArch64, ARMv7, x86, AMD64, MIPS, PPC, PPC64, S390, Sparc and RISC-V

Benefits of QEMU
Free Hardware

• QEMU is faster than FPGAs and completely customisable
• QEMU is available on all major distros
Tock for OpenTitan CI
Using QEMU to Develop Extensions

- QEMU is a valuable tool in prototyping extensions
  - It’s much quicker to add features to QEMU than hardware or full system simulators
  - QEMU is also very quick at running, allowing quick turn around times for tests

- QEMU can dump guest instructions as they are generated
  - Running QEMU with the `-d in_asm` command line argument outputs the generated input instructions
Debugging with QEMU
QEMU Status
Current Mainline QEMU Status

• QEMU supports these extensions:

• Patches on list for
  – IOMMU, crypto extensions and more

• Vendor extensions
  – XVentanacondOps

• 32/64/128-bit CPUs

• Contributions from: Western Digital, SiFive, C-Sky, Windriver, ISCAS and many others

• Getting started information available at: https://wiki.qemu.org/Documentation/Platforms/RISCV
RISC-V KVM on QEMU

QEMU supports KVM on RISC-V systems
Vector Extension Demo
Vendor Extensions in QEMU

• Adding new instructions
  1. Add a .decode file
     • An easy-to-read decoder file that defines the instructions
  2. Write TCG C implementation in trans_* .c .inc file
     • This contains assembly like implementation for instructions
  3. Wire up new files, add CPU config property and expose it to users

• Adding new CSRs still a work in progress

• Follow toolchain conventions
  – https://github.com/riscv-non-isa/riscv-toolchain-conventions/pull/17
How to get involved

• Contribute code to the QEMU mailing list
  – https://wiki.qemu.org/Contribute/MailingLists

• Help review and test extensions you are interested in