



**Western Digital<sup>®</sup>**

# **Free RISC-V Systems Benefits and Status of QEMU**

Alistair Francis <[alistair.francis@wdc.com](mailto: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



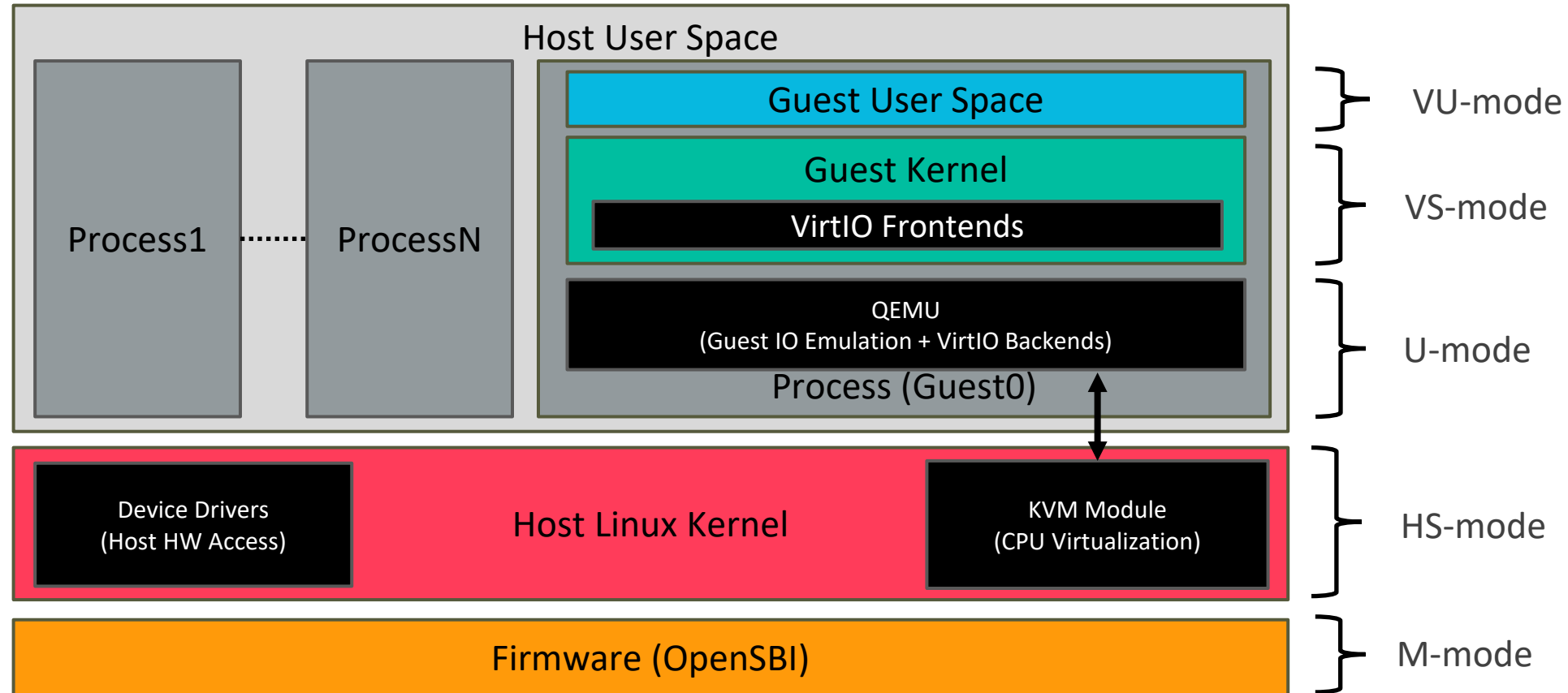
Benoît Canet – [wiki.qemu.org/Logo](https://wiki.qemu.org/Logo) [CC BY-SA](#)

# What is QEMU?

## Hypervisor

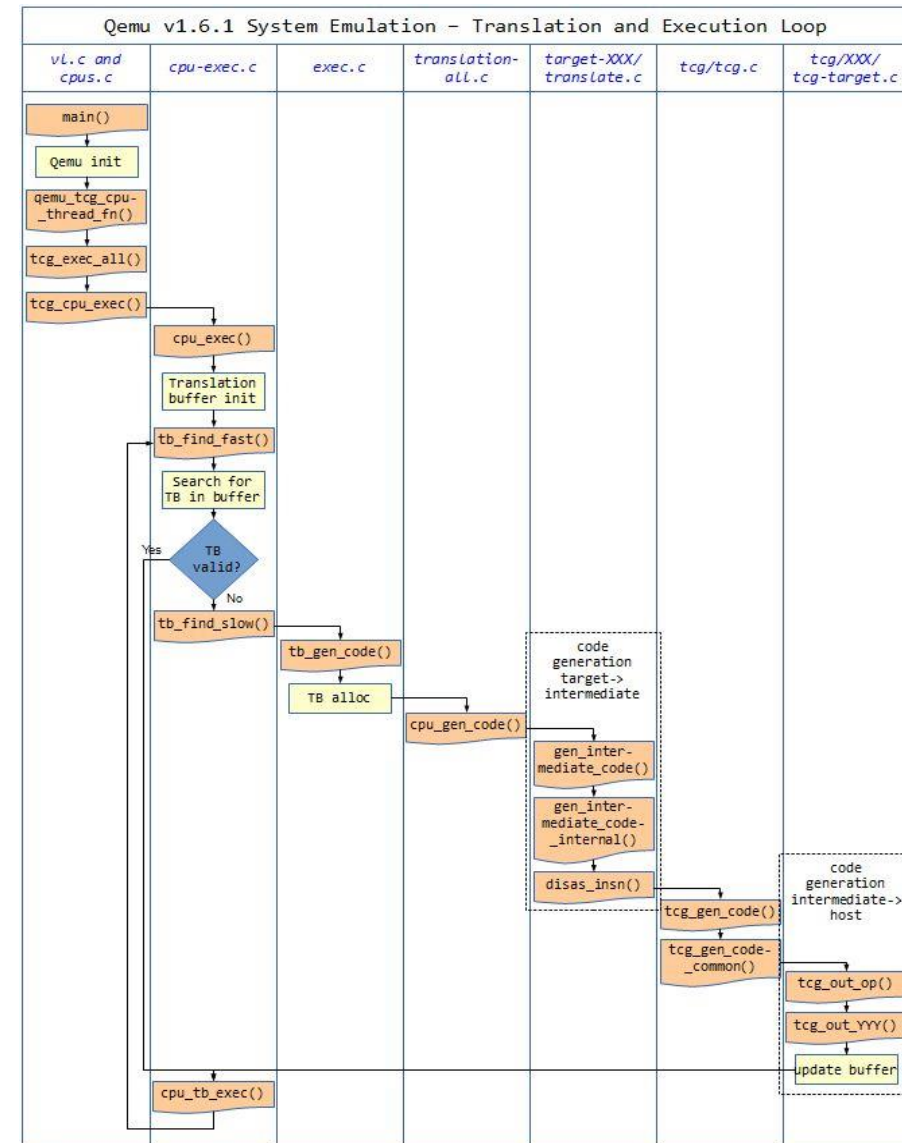


O.T.S.U. – KVM Logo [CC BY-SA](#)



# 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

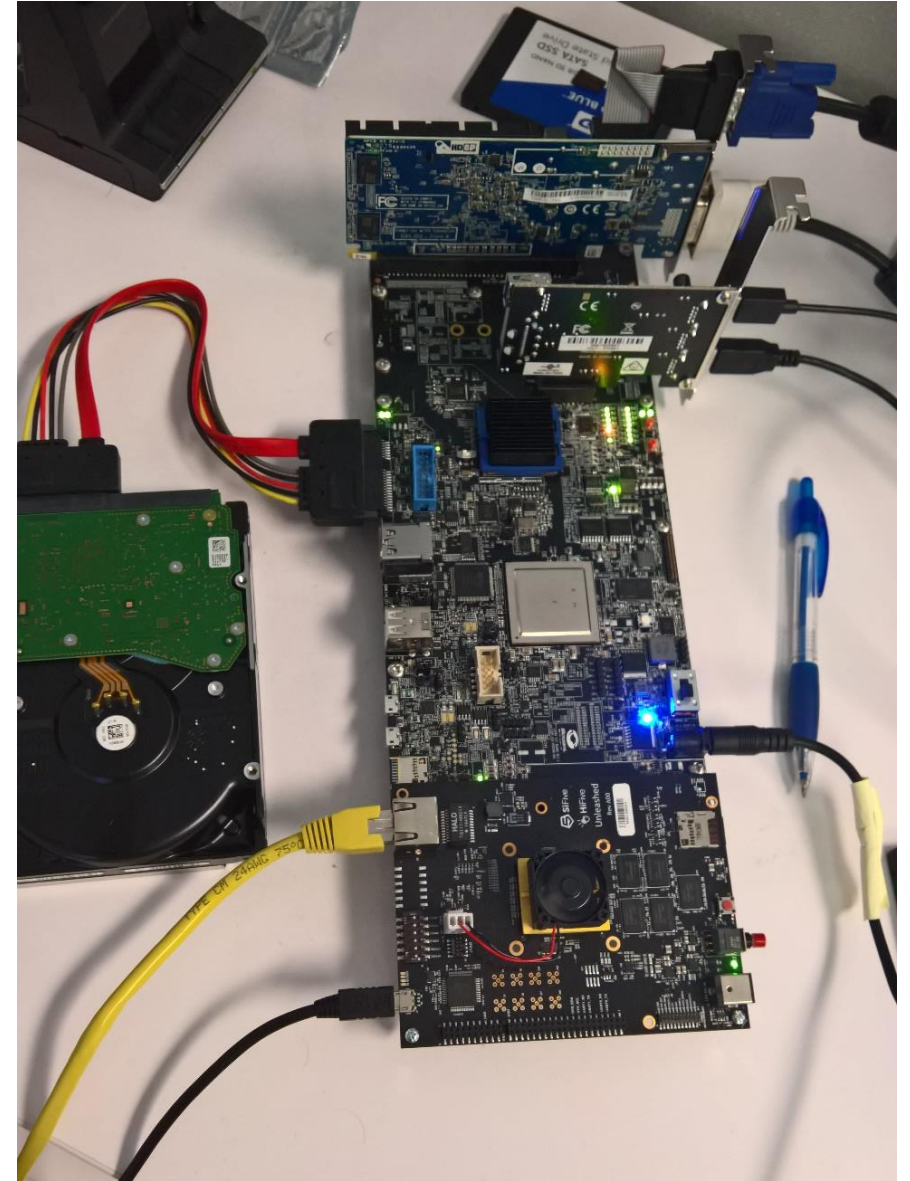


VividD - <https://stackoverflow.com/questions/20675226/qemu-code-flow-instruction-cache-and-tcg>

# Benefits of QEMU

# Free Hardware

- QEMU is faster than FPGAs and completely customisable
- QEMU is available on all major distros





# Tock for OpenTitan CI



The screenshot shows the GitHub Actions interface for the repository `tock / tock`, which is public. The `Actions` tab is selected, showing a workflow named `Add driver for LPM013M126 display` with a commit hash of `tock-ci #6687`. The workflow is marked as successful with a green checkmark. Below the workflow name, there is a `Summary` section and a list of jobs. The jobs listed are:

- `ci-format (ubuntu-latest)`
- `ci-clippy (ubuntu-latest)`
- `ci-build (ubuntu-latest)`
- `ci-build (macos-latest)`
- `ci-tests (ubuntu-latest)`
- `ci-tests (macos-latest)`
- `ci-qemu` (highlighted)

The `ci-qemu` job is expanded, showing its steps and status:

- `ci-qemu` succeeded 22 hours ago in 8m 24s
- `> Set up job`
- `> Update package repositories`
- `> Install dependencies`
- `> Run actions/checkout@v2`
- `> Run actions-rs/toolchain@v1`
- `> ci-job-qemu`
- `> Post Run actions/checkout@v2`
- `> Complete job`

# Using QEMU to Develop Extensions

```
-----
IN: fdt_get_string
Priv: 3; Virt: 0
0x000000008000cc62: 0004c703      lbu      a4,0(s1)
0x000000008000cc66: 0014c603      lbu      a2,1(s1)
0x000000008000cc6a: 0034c803      lbu      a6,3(s1)
0x000000008000cc6e: 0024c683      lbu      a3,2(s1)
0x000000008000cc72: 0106161b      slliw    a2,a2,16
0x000000008000cc76: 0187171b      slliw    a4,a4,24
0x000000008000cc7a: 8f51          or       a4,a4,a2
0x000000008000cc7c: 0086969b      slliw    a3,a3,8
0x000000008000cc80: 01076733      or       a4,a4,a6
0x000000008000cc84: 8f55          or       a4,a4,a5
0x000000008000cc86: 9d1d          subw     a0,a0,a3
0x000000008000cc88: d00e06b7      lui      a3,-80438864
0x000000008000cc8c: 0005061b      sext.w   a2,a0
0x000000008000cc90: 2701          sext.w   a4,a4
0x000000008000cc92: 1502          slli     a0,a0,32
0x000000008000cc94: eed68693      addi     a3,a3,-275
0x000000008000cc98: 9101          srli     a0,a0,32
0x000000008000cc9a: 06d70463      beq      a4,a3,104      # 0x8000cd02

-----
IN: fdt_get_string
Priv: 3; Virt: 0
0x000000008000cd02: 0409cf63      bltz     s3,94          # 0x8000cd60

-----
IN: fdt_get_string
Priv: 3; Virt: 0
0x000000008000cd06: 0144c703      lbu      a4,20(s1)
0x000000008000cd0a: 0154c803      lbu      a6,21(s1)
0x000000008000cd0e: 0174c883      lbu      a7,23(s1)
0x000000008000cd12: 0164c683      lbu      a3,22(s1)
0x000000008000cd16: 0187171b      slliw    a4,a4,24
0x000000008000cd1a: 0108181b      slliw    a6,a6,16
0x000000008000cd1e: 01076733      or       a4,a4,a6
0x000000008000cd22: 0086969b      slliw    a3,a3,8
0x000000008000cd26: 01176733      or       a4,a4,a7
0x000000008000cd2a: 8f55          or       a4,a4,a3
0x000000008000cd2c: 2701          sext.w   a4,a4
0x000000008000cd2e: 46c1          addi     a3,zero,16
0x000000008000cd30: fae6fae3      bleu     a4,a3,-76      # 0x8000cce4

-----
IN: fdt_get_string
Priv: 3; Virt: 0
0x000000008000cd34: 0204c703      lbu      a4,32(s1)
0x000000008000cd38: 0214c803      lbu      a6,33(s1)
0x000000008000cd3c: 0234c883      lbu      a7,35(s1)
0x000000008000cd40: 0224c683      lbu      a3,34(s1)
0x000000008000cd44: 0187171b      slliw    a4,a4,24
0x000000008000cd48: 0108181b      slliw    a6,a6,16
0x000000008000cd4c: 01076733      or       a4,a4,a6
0x000000008000cd50: 01176733      or       a4,a4,a7
0x000000008000cd54: 0086969b      slliw    a3,a3,8
0x000000008000cd58: 8f55          or       a4,a4,a3
0x000000008000cd5a: 2701          sext.w   a4,a4
0x000000008000cd5c: 02e5e263      bgtu     a4,a1,36      # 0x8000cd80

-----
IN: fdt_get_string
Priv: 3; Virt: 0
0x000000008000cd80: 9f0d          subw     a4,a4,a1
0x000000008000cd82: 0007069b      sext.w   a3,a4
0x000000008000cd86: f4c6ffe3      bleu     a2,a3,-162     # 0x8000cce4

-----
IN: fdt_get_string
Priv: 3; Virt: 0
0x000000008000cce4: 1782          slli     a5,a5,32
0x000000008000cce6: 9381          srli     a5,a5,32
0x000000008000cce8: 94be          add      s1,s1,a5
```

- 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

```
alistair@toolbox: /scratch/alistair/software/automat — Konsole <2>
>>> si
Output/messages
0x00000000800006b2 292 in /scratch/alistair/yocto/oe-master/build/tmp-glibc/work/riscv64-oe-linux/opensbi/0.8-r0/git/lib/sbi/sbi_init.c
Assembly
-
0x00000000800006ac sbi_init+0 addi sp,sp,-224
0x00000000800006ae sbi_init+2 sd s0,208(sp)
0x00000000800006b0 sbi_init+4 sd ra,216(sp)
0x00000000800006b2 sbi_init+6 sd s1,200(sp)
0x00000000800006b4 sbi_init+8 sd s2,192(sp)
0x00000000800006b6 sbi_init+10 sd s3,184(sp)
0x00000000800006b8 sbi_init+12 sd s4,176(sp)
0x00000000800006ba sbi_init+14 sd s5,168(sp)
Breakpoints
Expressions
History
Memory
Registers
ra 0x00000000800004d2 sp 0x00000000800016f20 gp 0x0000000000000000 tp 0x0000000080017000 t0 0x0000000000000000 t1 0x0000000000000000 t2 0x0000000000000000
fp 0x0000000000000000 s1 0x0000000000000000 a0 0x0000000080017000 a1 0x000000008f000000 a2 0x0000000000001028 a3 0x0000000000000000 a4 0x0000000080000540
a5 0x0000000000001000 a6 0x0000000000000000 a7 0x0000000000000000 s2 0x0000000000000000 s3 0x0000000000000000 s4 0x0000000000000000 s5 0x0000000000000000
s6 0x0000000000000000 s7 0x0000000000000004 s8 0x0000000000002000 s9 0x000000008001370c s10 0x0000000000000000 s11 0x0000000000000000 t3 0x0000000000000000
t4 0x0000000000000000 t5 0x0000000000000000 t6 0x0000000000000000 pc 0x00000000800006b2
Source
Cannot display "sbi_init.c"
Stack
[0] from 0x00000000800006b2 in sbi_init+6 at /scratch/alistair/yocto/oe-master/build/tmp-glibc/work/riscv64-oe-linux/opensbi/0.8-r0/git/lib/sbi/sbi_init.c:292
[1] from 0x00000000800004d2 in _start_warm at /scratch/alistair/yocto/oe-master/build/tmp-glibc/work/riscv64-oe-linux/opensbi/0.8-r0/git/firmware/fw_base.S:424
Threads
[4] id 4 from 0x00000000800006b2 in sbi_init+6 at /scratch/alistair/yocto/oe-master/build/tmp-glibc/work/riscv64-oe-linux/opensbi/0.8-r0/git/lib/sbi/sbi_init.c:292
[3] id 3 from 0x00000000800006dc in sbi_init+48 at /scratch/alistair/yocto/oe-master/build/tmp-glibc/work/riscv64-oe-linux/opensbi/0.8-r0/git/lib/sbi/sbi_init.c:293
[2] id 2 from 0x00000000800006b6 in sbi_init+10 at /scratch/alistair/yocto/oe-master/build/tmp-glibc/work/riscv64-oe-linux/opensbi/0.8-r0/git/lib/sbi/sbi_init.c:292
[1] id 1 from 0x00000000800006e8 in sbi_init+60 at /scratch/alistair/yocto/oe-master/build/tmp-glibc/work/riscv64-oe-linux/opensbi/0.8-r0/git/lib/sbi/sbi_init.c:293
Variables
arg scratch = 0x80017000: {fw_start = 2147483648,fw_size = 122880,next_arg1 = 2183135232,next_addr = 21
loc coldboot = 0, hartid = <optimized out>, plat = <optimized out>
>>>
```

An abstract, colorful pattern of swirling lines in shades of blue, orange, and red, resembling a stylized flame or a digital signal, located on the far left edge of the slide.

# QEMU Status

# Current Mainline QEMU Status

- QEMU supports these extensions:
  - I, E, G, M, A, F, D, C, S, U, V, H, Counters, Zifencei, Zicsr, Zfh, Zfhmin, Zve32f, Zve64f, MMU, PMP, debug, svinval, svnapot, svpbmt, Zba Zbb, Zbc, Zbs, Zdinx, Zfinx, Zhinx, Zhinxmin, J, ePMP and AIA
- 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

A terminal window with a dark background. The title bar at the top reads "alistair@toolbox: /scratch/alistair/software/qemu/build — Konsole". The main area of the terminal shows a green prompt "alistair@toolbox:" followed by a blue path "/scratch/alistair/software/qemu/build" and a grey prompt "(master)\*\$". A white cursor is positioned at the end of the line.

```
alistair@toolbox: /scratch/alistair/software/qemu/build (master)*$
```

# Vector Extension Demo



A terminal window titled "alistair@toolbox: /scratch/alistair/software/qemu/build — Konsole". The prompt is "alistair@toolbox: /scratch/alistair/software/qemu/build (master)\*\$" with a white cursor. The terminal background is dark gray.

# 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



# Western Digital®