Mastering Embedded RISC-V Systems: Develop Practical Applications using Simulators

Abdul Qadeer
Research Assistant
Centre for AI & Big Data
Namal University
Mianwali

Contents

- **Spike: The official RISC-V instruction set simulator.**
- QEMU: A generic and open-source machine emulator and virtualizer.
- GEM5: A modular platform for computer-system architecture research.

Introduction to Spike Simulator

Spike is:

- official RISC-V ISA simulator
- developed by the RISC-V Foundation
- used for testing and debugging of RISC-V applicatins
- an Instruction Set Simulator and supports RV32I, RV64I, RV32E etc

Top Level Structure

Boot ROM: Contains the initial boot code for the RISC-V processors. This is the first code executed when the simulation starts, initializing the system.

Debug Module: Provides debugging for the the execution of instructions.

Bus: It connects various components of the RISC-V simulator allowing them to transfer data and instructions.

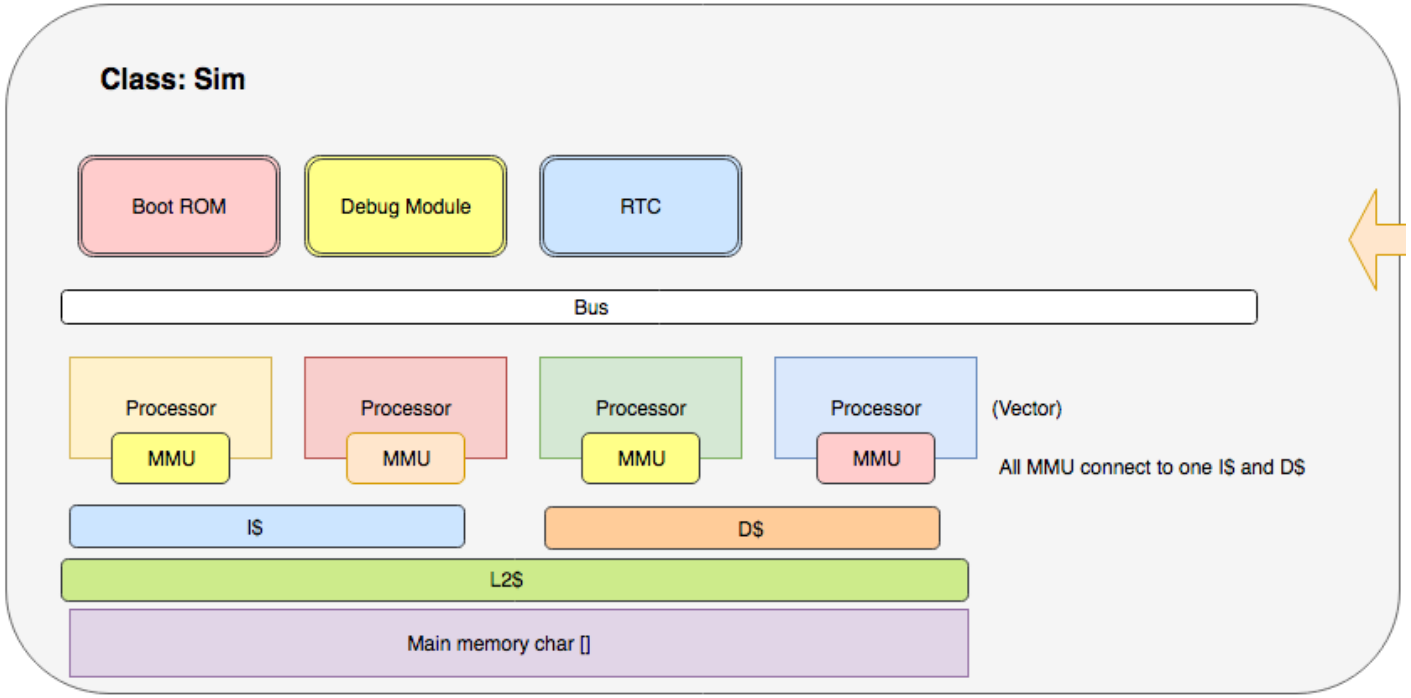

Figure 1: Top level structure of Spike Simulator

[https://github.com/poweihuang17/Documentation_Spike]

Memory System Overview

- **Processor** Generates virtual addresses (VA) and sends virtual addresses to the Memory Management Unit (MMU).
- **MMU** (Memory Management Unit) receives virtual addresses (VA) from the processor.

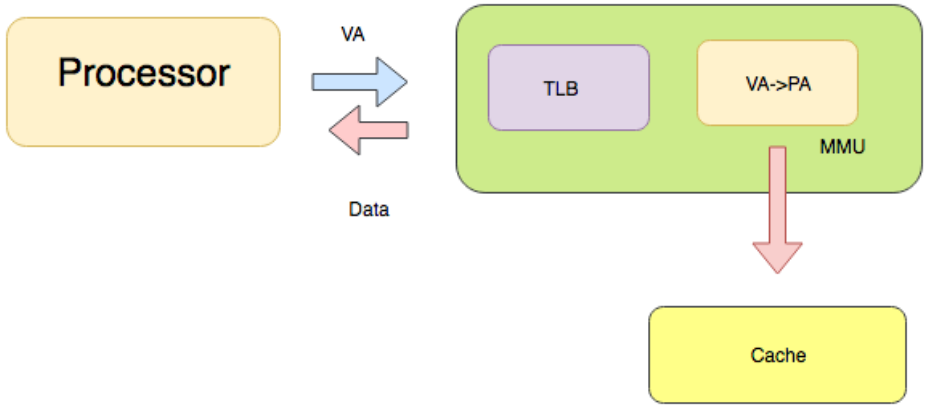

Figure 2: Memory Overview Of spike

- It also contains a **Translation Lookaside Buffer (TLB)** for quick address translation that converts virtual addresses (VA) to physical addresses (PA).
- **Cache** receives physical addresses from the MMU. Retrieves & Sends the requested data back to the processor.

Handson Session on Spike

- **Write a c code of your own choice. i.e *Hello World*.**

  vim yourcode.c

- **Make riscv64 binary of your code.**

  riscv64-unknown-elf-gcc -o exefile yourcode.c

- **Run on spike**

  spike pk ./exefile

- **More Features:**

  spike -d pk ./exefile //debugging feature

  spike –ic=<sets>:<ways>:<cache_size> pk ./exefile //cache sim

  spike -l pk ./exefile //trace of execution of inst

Contents

- Spike: The official RISC-V instruction set simulator.
- **QEMU: A generic and open-source machine emulator and virtualizer.**
- GEM5: A modular platform for computer-system architecture research.

Introduction

What is QEMU?

The QEMU is a virtual machine monitor for the host. It can emulate a Central Processing Unit (CPU) architecture that does not need to match the architecture of the host processor by using the dynamically binary translation [1].

A new operating system can be installed on the emulated processor, which cannot be affected by the host operating system.

The Internals of Qemu

- Hypervisor loads binary machine code from the disk image.
- TCG translates the binary code into executable format for the virtual machine.
- Translated code is stored in the translation cache for efficient retrieval.
- Hypervisor uses the translation cache to manage virtual and real device interactions.
- MMU manages memory addresses, and the hypervisor emulates the operating system from the disk image.

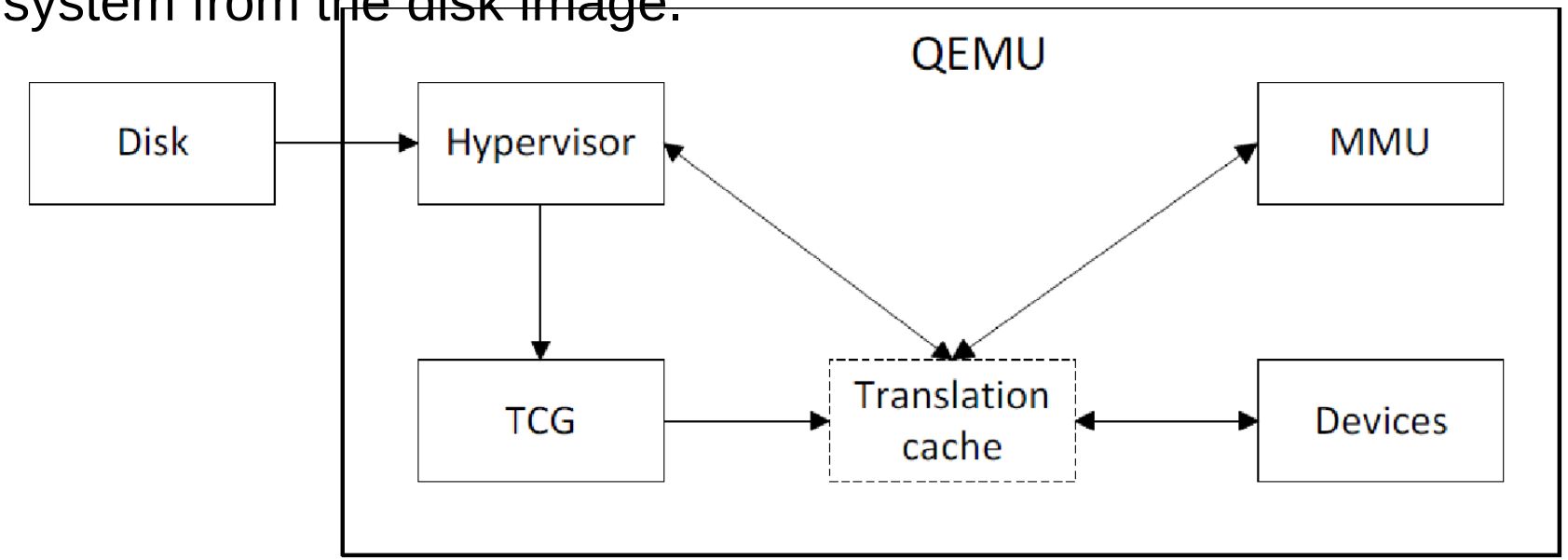

Figure 5: The QEMU Architecture Layer Overview

Emulation Modes of Qemu

User Mode:

Emulates individual applications from a guest system by translating application-level instructions for execution on the host system, without emulating the entire operating system or hardware.

System Mode:

Full System Emulation: Emulates the entire guest system, including CPU, memory, and peripherals, allowing for the execution of a complete operating system and its applications.

Handson Session on Qemu

Contents

- Spike: The official RISC-V instruction set simulator.
- **GEM5: A modular platform for computer-system architecture research.**
- QEMU: A generic and open-source machine emulator and virtualizer.

Introduction

The gem5 is the fusion of two projects

GEMS

- Detailed and flexible memory system model
- Includes support for multiple cache coherence protocols and interconnect models
- Developed by **The University of Wisconsin Madison**

M5

- Highly configurable simulation framework to support multiple ISAs, and diverse CPU models.
- Developed by **The University of Michigan**

What is gem5 used for ?

Architectural exploration

Gem5 provides a fast and easy framework to interconnect hardware components and evaluate them !

Hardware/software performance evaluation ?

Gem5 has a good support of various ISA and allows for realistic HW/SW performance evaluation

Gem5 CPU Models

O3CPU

Six-stage pipelined &

Out of Order Model

MinorCPU:

Four-stage pipelined

In-Order CPU Model

TimingSimpleCPU:

- Single Cycle CPU Model

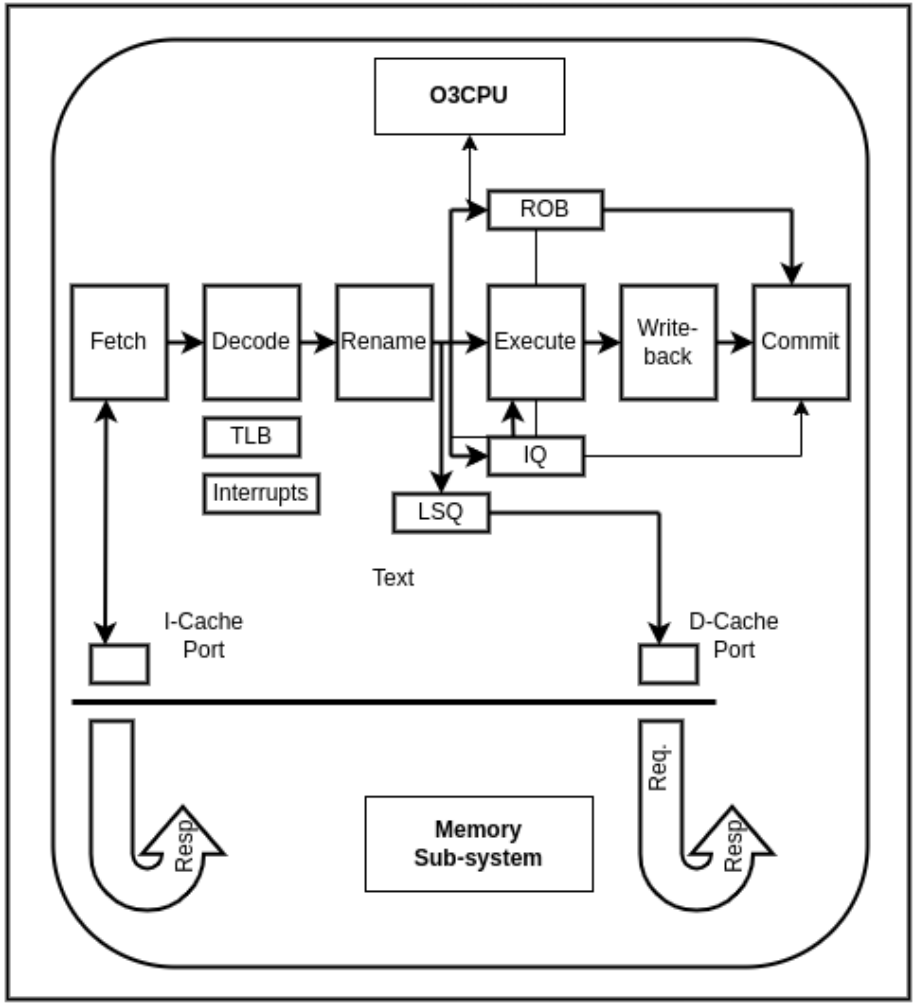

Figure 4 : O3CPU

Gem5 Modes of Operations

Full-system (FS)

- Models bare-metal hardware
  - Includes the various specified devices, caches, …
- Boots an entire OS from scratch

Syscall Emulation (SE)

- Runs a single static application
- System calls are emulated or forwarded to the host OS
- Lot of simplifications (address translation)

Gem5 Modes of Operations

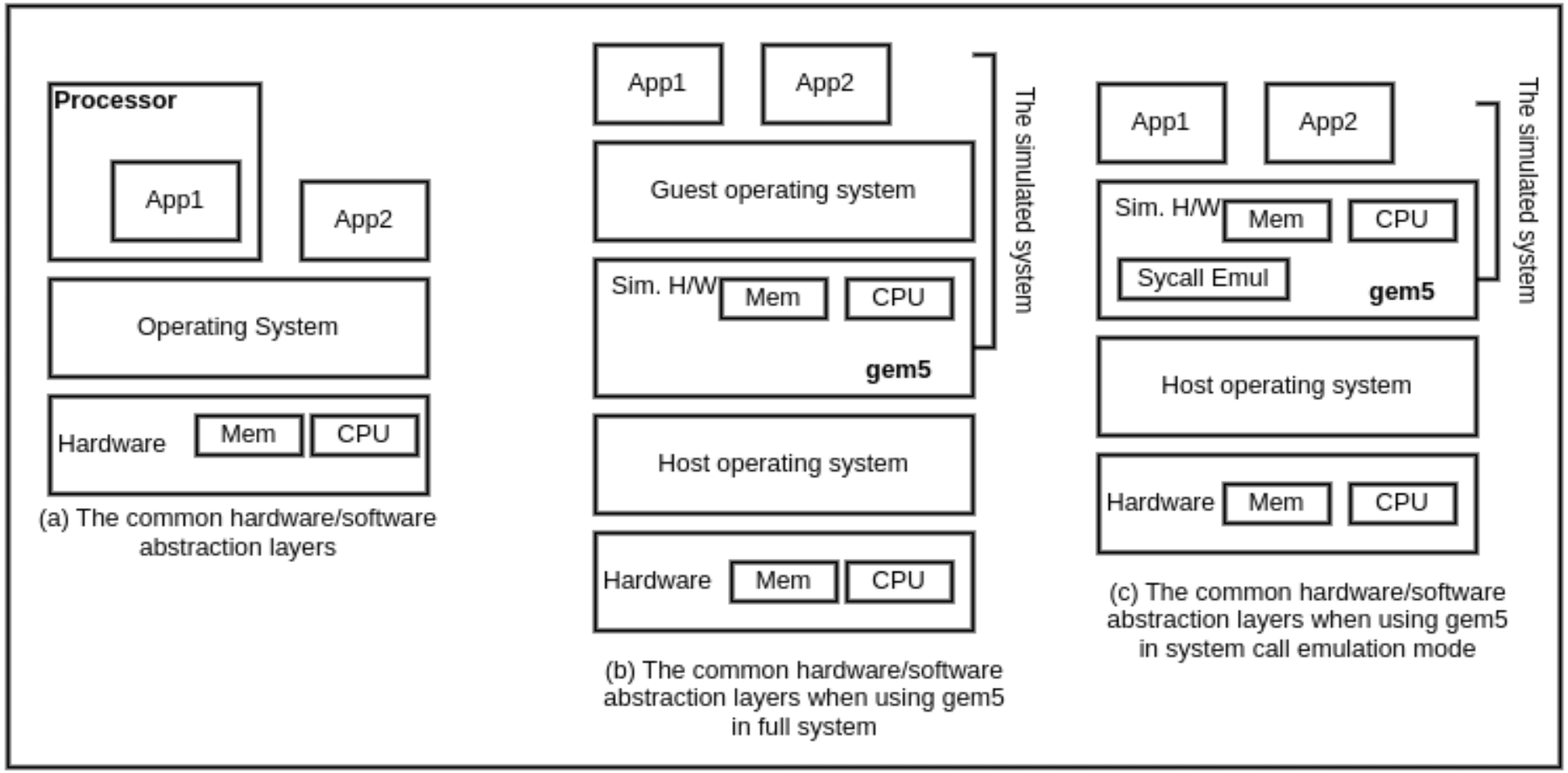

Figure 3 : Modes of operations of gem5

Design and Development Principle

Single Core with Cache and Main Memory Support

- Configure Python script called system.py
- Integrate benchmark script
- Simulate the system using gem5
- Analyze performance statistics

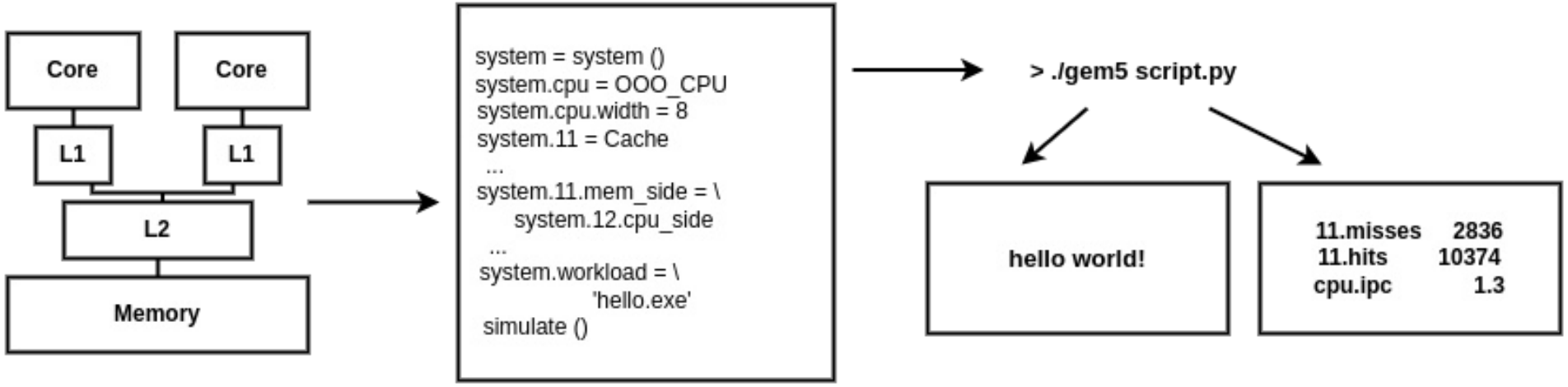

Figure 4 : The gem5 usage overview

Handson Session on gem5

- **Write a c code of your own choice. i.e *Hello World.***

  vim yourcode.c
- **Make riscv64 binary of your code.**

  riscv64-unknown-elf-gcc -o exefile yourcode.c
- **Create python script provided in the HPC directory**
- **Run on gem5**

  gem5.opt -d m5out script.py

Thank You!