gem5 and RISC-V

Architetture dei Sistemi di Elaborazione 2023-24

P. Bernardi, E. Sanchez
F. Angione, G. Insinga, A. Ruospo

Politecnico di Torino
Dipartimento di Automatica e Informatica

Outline

- **RISC-V**
- **gem5 introduction**
- **Hands-on experience**
- **Useful links**

RISC-V

RISC-V is an open standard Instruction Set Architecture (ISA)

History:

- **RISC-V instruction set in May 2010 as part of the Parallel Computing Laboratory (Par Lab) at UC Berkeley, of which Prof. David Patterson was Director.**
- **The first RISC-V Workshop held in January 2015, and the RISC-V Foundation launch later that year with 36 Founding Members**

RISC-V

What is RISC-V?

- **A high-quality, license-free, royalty-free RISC ISA**
- **Standard maintained by the non-profit RISC-V Foundation**
- **Suitable for all types of computing systems**
  - **From Microcontrollers to Supercomputers**
- **RISC-V is available freely under a permissive license**
- **RISC-V is not…**
  - **A Company**
  - **A CPU implementation**

RISC-V

The worldwide interest in RISC-V is not because it is a great new chip technology, the interest is because it is a common free and open standard to which software can be ported, and which allows anyone to freely develop their own hardware to run the software.

In 2010, when RISC-V was founded, commercial ISAs were too complex and presented IP legal issues.

The standard extensions

- **RV32I – The most basic RISC-V implementation**
- **RV32IMAC – Integer + Multiply + Atomic + Compressed**
- **RV64GC – 64bit IMAFDC**
- **RV64GCXext – IMAFDC + a non-standard extension**

Extension	Description
I	Integer
M	Integer Multiplication and Division
A	Atomics
F	Single-Precision Floating Point
D	Double-Precision Floating Point
G	General Purpose = IMAFD
C	16-bit Compressed Instructions
Non-Standard User-Level Extensions	
Xext	Non-standard extension “ext”

Common RISC-V Standard Extensions
*Not a complete list

Worldwide interest

gem5

The gem5 simulator is a modular platform for computer-system architecture research, encompassing system-level architecture as well as processor microarchitecture.

gem5 is a joint project made by the union of 2 simulation tools:

Michigan m5 + Wisconsin GEMS = gem5

Importance of System Simulation

Complex iteration between the CPU and the memory system should be considered.

HW/SW performance verification

- **Need accurate performance measures**

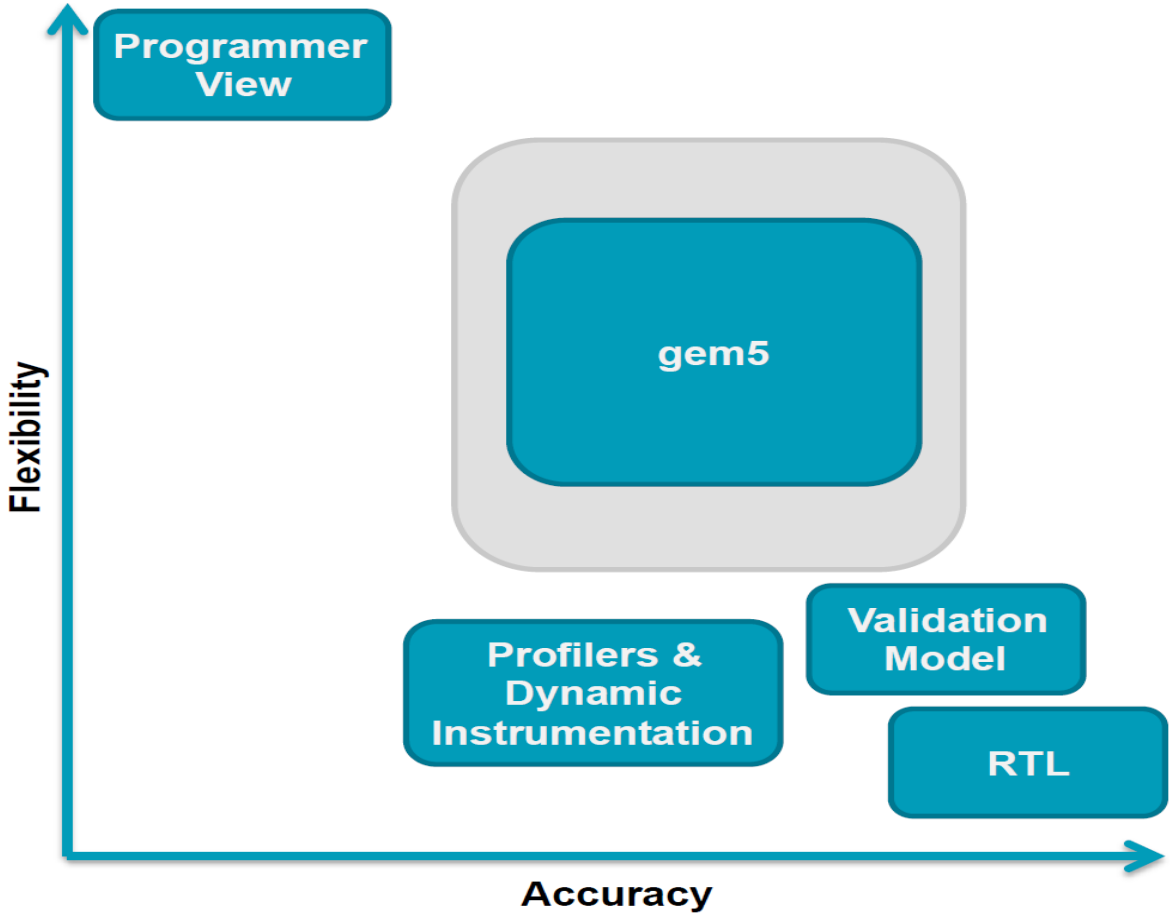
Early Architectural Exploration

- **Need an environment where it is fast and easy to model and connect the key architectural components**

Useful for computing:

- **Benchmark run time, CPU performance, Interconnect latencies, DRAM controller scheduling.**

gem5 is a flexible simulator



gem5 modes of operation

Full system (FS)

- **For booting operating systems**
- **Models bare hardware, including devices**
- **Interrupts, exceptions, privileged instructions, fault handlers**

Syscall emulation (SE)

- **For running individual applications, or set of applications on MP/SMT**
- **Models user-visible ISA plus common system calls**
- **System calls emulated, typ. by calling host OS**
- **Simplified address translation model, no scheduling.**

gem5 timing

Gem5 is an event-driven discrete simulator.

The time is not continuous but discrete.

In this case the smallest unit is the tick.

Therefore, the Tick is a way for discretizing the time (again, normally continuous) on a computer and let gem5 know that the time is running.

Given this, the clock cycle is made of some ticks and this allows for event-driven programming (i.e. things to happen within the clock cycle).

gem5 timing

In our case, 1 tick expires every 1ps

1 tick = 1ps, since we have in the configuration (gem5 output): global frequency set at 1,000,000,000,000 ticks per second

https://www.gem5.org/documentation/learning_gem5/part1/gem5_stats/

gem5 capabilities

- **Execution modes**: System-call Emulation (SE) & Full-System (FS)
- **ISAs**: ALPHA, ARM, MIPS, Power, SPARC, x86, RISCV
- **CPU models**: AtomicSimple, TimingSimple, InOrder, and O3
- **Cache coherence protocols**: broadcast-based, directories, etc.
- **Interconnection networks**: Simple & Garnet
- **Devices**: NICs, IDE controller, etc.
- **Multiple systems**: communicate over TCP/IP.

CPU models

Simple CPUs

- **Models Single-Thread 1 CPI Machine**
- **Two Types: AtomicSimpleCPU and TimingSimpleCPU**
- **Common Uses:**
  - **Fast, Functional Simulation: 2.9 million and 1.2 million instructions per second**
  - **Studies that do not require detailed CPU modeling**

CPU models - 2

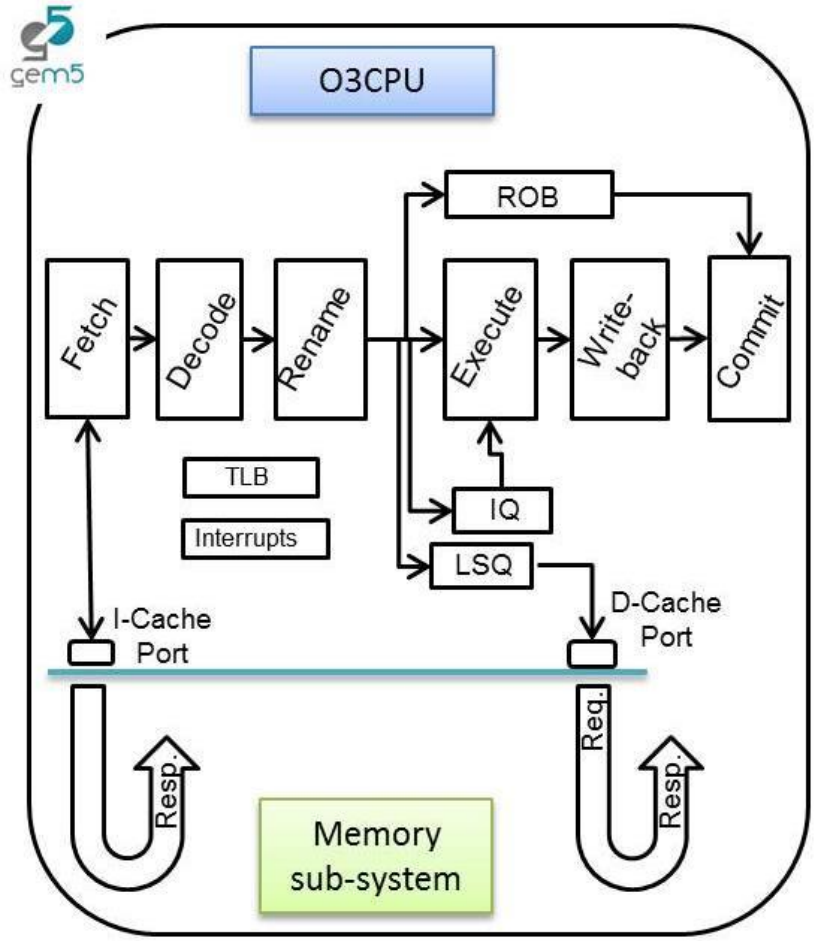
Detailed CPUs

- **Parameterizable Pipeline Models w/SMT support**
- **Two Types: InOrderCPU and O3CPU**
- **Slower than SimpleCPUs: 200K instructions per second**
  - **Models the timing for each pipeline stage**
  - **Forces both timing and execution of simulation to be accurate**
  - **Important for Coherence, I/O, Multiprocessor Studies, etc.**

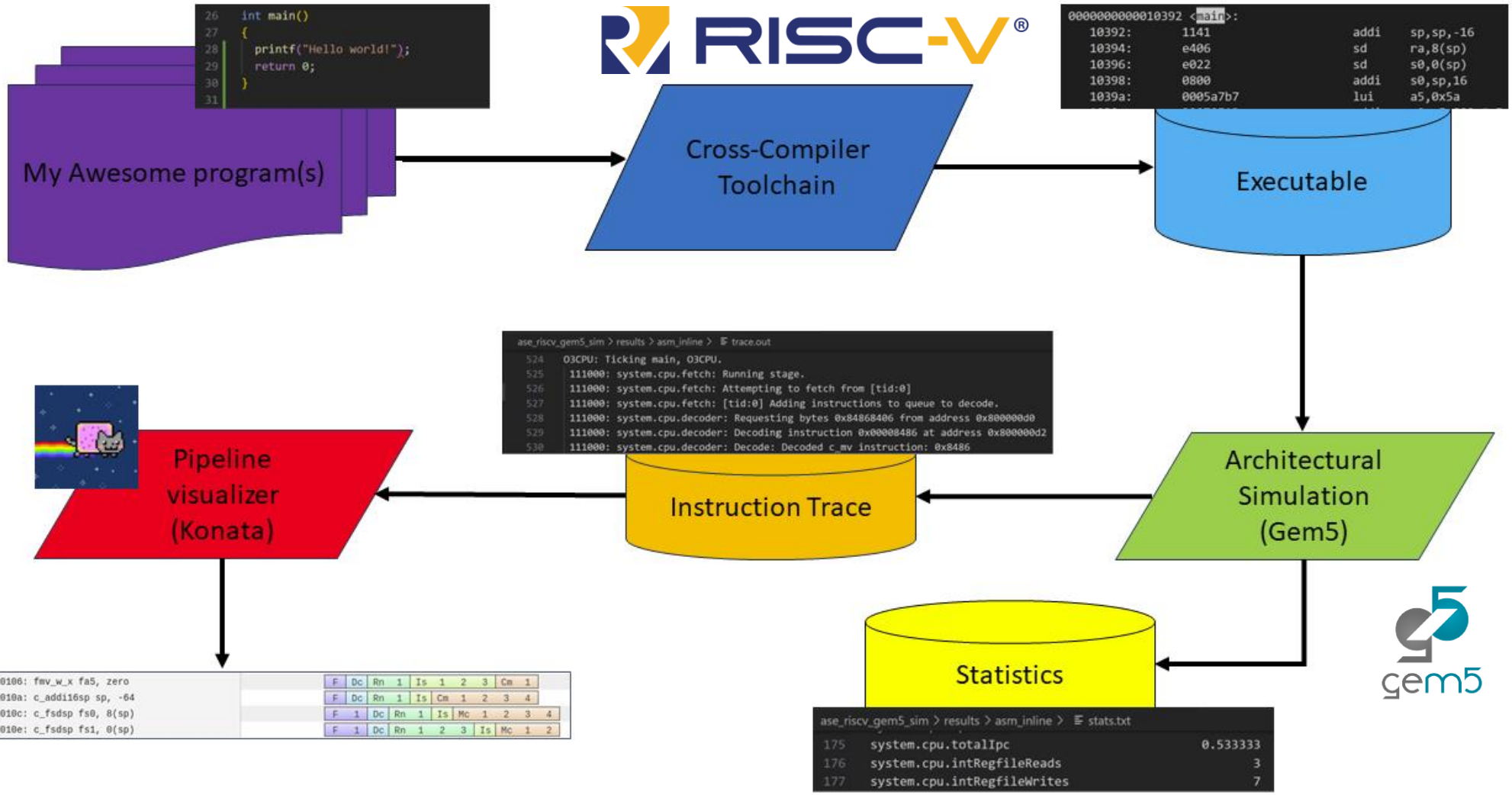
O3 CPU

Detailed Out-Of-Order CPU

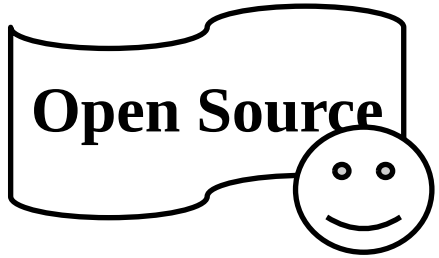
- **Default 7-stage pipeline**
  - **Fetch, Decode, Rename, IEW, Commit**
  - **IEW <=>Issue, Execute, and Writeback**
- **Functional units with varying latencies**
- **Branch prediction**
- **Memory dependence prediction.**
- **AKA: DerivO3CPU.**



Hands-on experience



Useful Links



Gem5:

https://www.gem5.org/

https://www.gem5.org/documentation/

RISC-V

https://riscv.org/

https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf

KONATA

https://github.com/shioyadan/Konata