

Cryptographic system for  
common cryptographic algorithms based on  
the open-source RISC-V architecture

Hệ thống xử lý dựa trên  
kiến trúc mã nguồn mở RISC-V  
ứng dụng cho một số bộ xử lý mật mã thông dụng

The University of Electro-Communications  
Đại học Điện tử - Viễn thông

Phạm Công-Kha  
Hoàng Trọng-Thức

Tháng 9 năm 2023

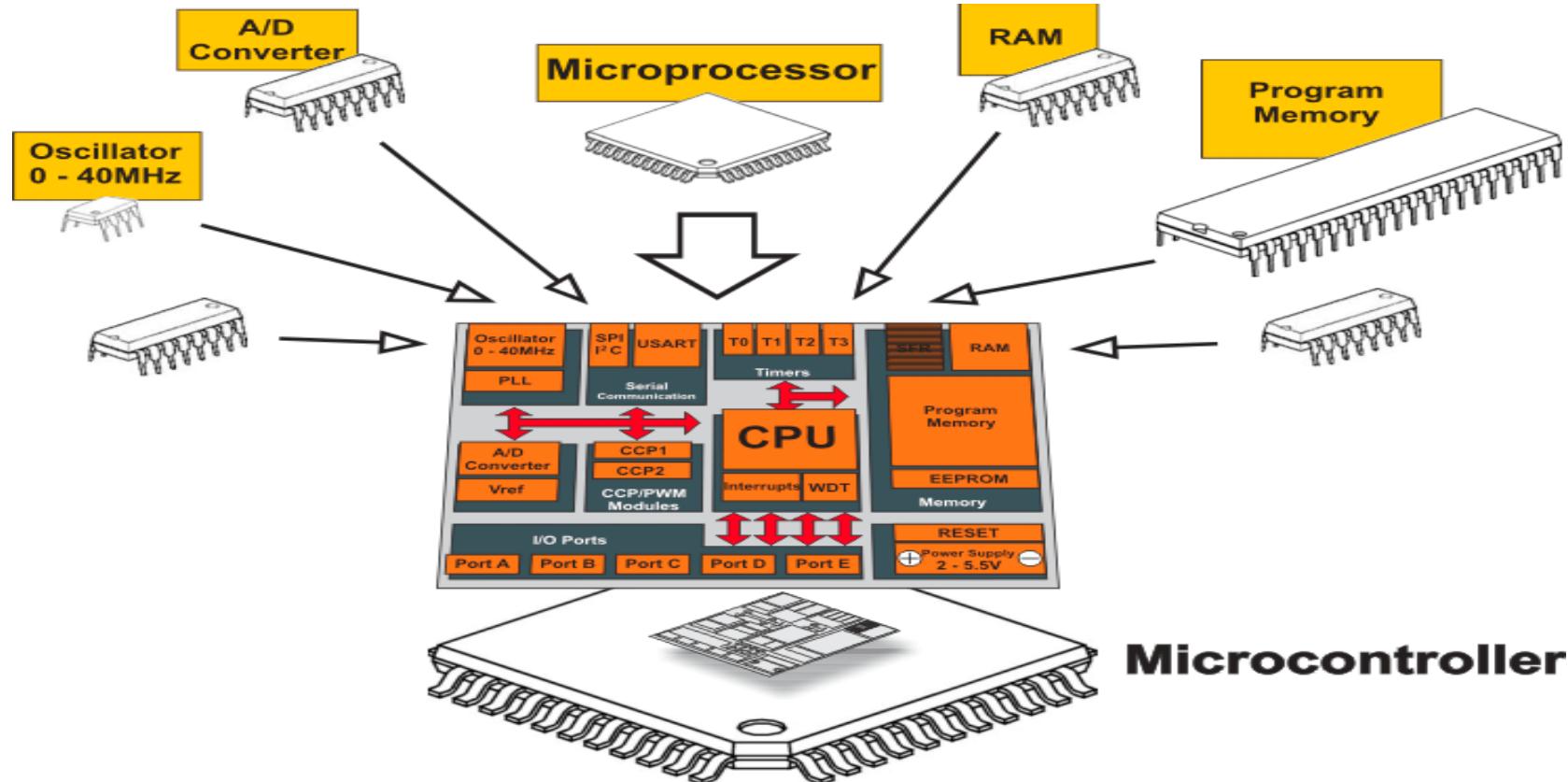
# Introduction to Microcontrollers

- A microcontroller (MCU) is a small computer on a single integrated circuit consisting of a relatively simple central processing unit (CPU) combined with peripheral devices such as memories, I/O devices, and timers.
- By some accounts, more than half of all CPUs sold worldwide are microcontrollers

# Microcontroller VS Microprocessor

- A microcontroller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.
- A microprocessor incorporates the functions of a computer's central processing unit (CPU) on a single integrated circuit.

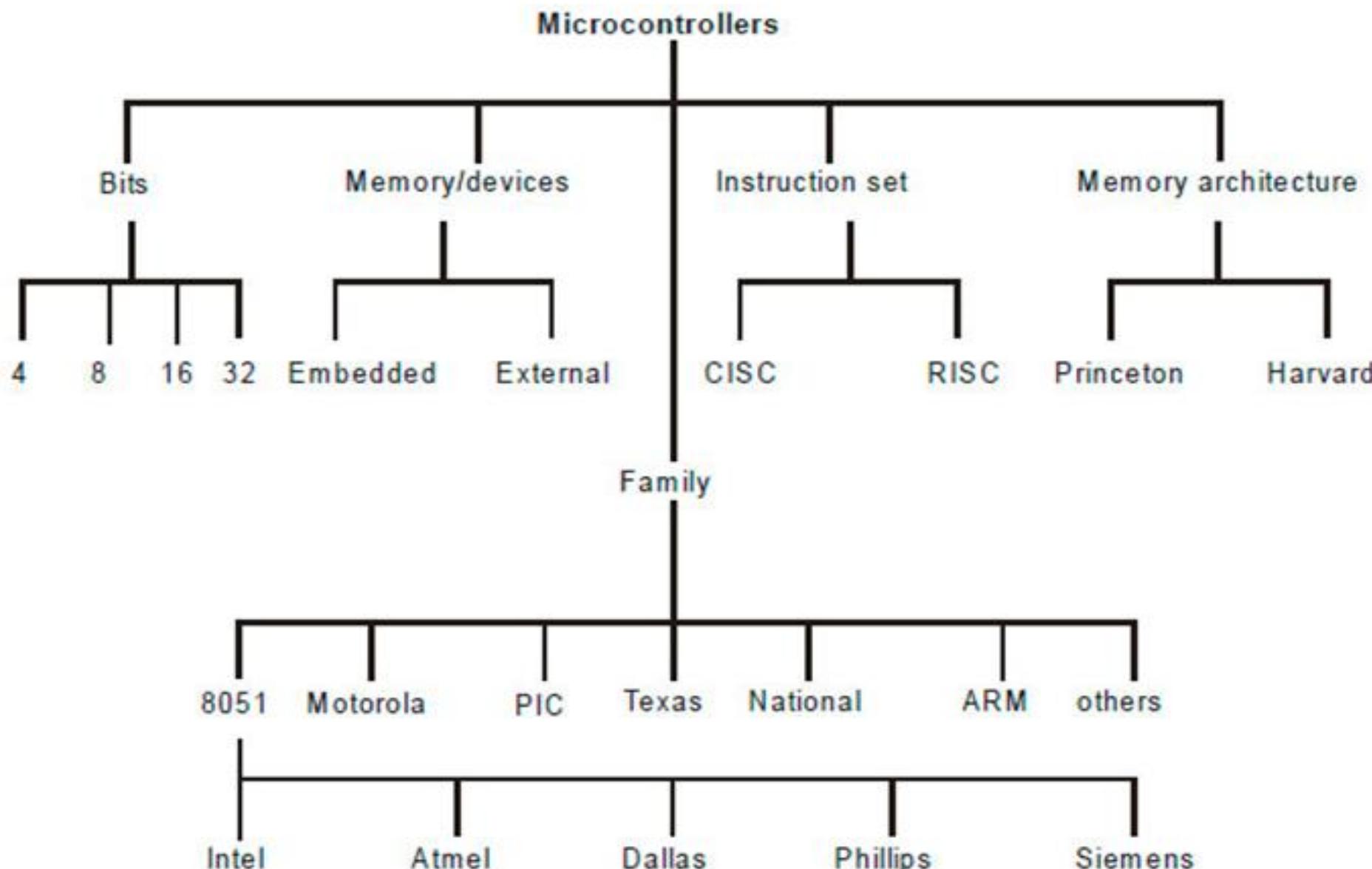
# Microcontroller VS Microprocessor



# Types of Processors

- In general-purpose computing, the variety of instruction set architectures today is limited, with the Intel x86 architecture overwhelmingly dominating all.
- There is no such dominance in embedded computing. On the contrary, the variety of processors can be daunting to a system designer.
- Things that matter
  - Peripherals, Concurrency & Timing, Clock Rates, Memory sizes (SRAM & flash), Package sizes

# Types of Microcontrollers



# How to choose MCU for our project?

- What metrics we need to consider?
  - Power consumption
  - Clock frequency
  - IO pins
  - Memory
  - Internal functions
  - Others

# How to choose MCU for our project?

- What metrics we need to consider?
  - Power consumption
    - We cannot afford mA MCU because the power budget of the system is 3.47mA.
  - Clock frequency (speed that instructions are executed)
    - kHz is too slow...
    - 100MHz is over kill...
  - IO pins
    - Lots of peripherals - Image sensor, UART debugger, SD card, DAC, ADC, microphone, LED

# How to choose MCU for our project?

- What metrics we need to consider?
  - Memory
    - We need to have sufficient memory to store:
      - Program (Non-volatile): Logic to read from sensors, communicate
      - Stack: Function calls are now expensive (no recursion)
      - Data: Constants (time periods), Sensor history, Communication state
    - Internal functions
      - Migrating data from the sensor to the radio (DMA)

# How to choose MCU for our project?

- Memory
  - Store accelerometer history data
    - 12bits each for X,Y,Z acceleration
    - sampled 2 thousand times a second (2 KHz)
    - =  $12 \times 3 \times 2,000$  bits per second (72kbytes or 9 kBytes)
    - How many seconds can we hold if we have only 100 kBytes of storage
  - What types of memory are available on an MCU?
    - Internal memory: RAM, 0.5~128 kBytes
    - External memory: Flash, high power consumption, ~5mA for read and ~10mA for erase

# How to choose MCU for our project?

- Clock frequency
  - kHz is too slow
    - Smartphone camera frame rate is 60fps  
(1 KHz clock would leave only 60 clock cycles per frame)
  - 100MHz is too fast
    - Power consumption is high
    - Several MHz would be ideal

# How to choose MCU for our project?

- IO pins (interface for external peripherals)
  - Interfacing sensors, UART debugger, LEDs, Bluetooth
  - We need **a large number** of IO pins
  - We need **various types** of IO pins
    - Analog pins (input/output analog signals e.g., audio)
    - Digital pins (input/output digital signals e.g., busses, GPIOs)

# Embedded System

- An embedded system is a system that has software embedded into computer-hardware, which makes a system dedicated for an application(s) or specific part of an application or product or part of a larger system.
- An embedded system is one that has a dedicated purpose software embedded in a computer hardware.
- Its software usually embeds into a ROM (Read Only Memory) or flash.



# Microcontroller as the heart of embedded system

- Microcontroller is an IC chip that takes input process data according to program written in its memory and gives output as control signal for controlling other machines and devices.
- A microcontroller (sometimes abbreviated  $\mu C$ ,  $uC$  or MCU) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.
- In today's world of technology, we found Microcontrollers in almost every electronic device we uses. Almost all general purpose devices such as Digital Watches, Mobile Phones works, etc. on the bases of Microcontroller.

# Instruction Set ISA

- The repertoire of instructions of a computer
- Different computers have different instruction sets
  - But with many aspects in common
- Early computers had very simple instruction sets
  - Simplified implementation
- Many modern computers also have simple instruction sets

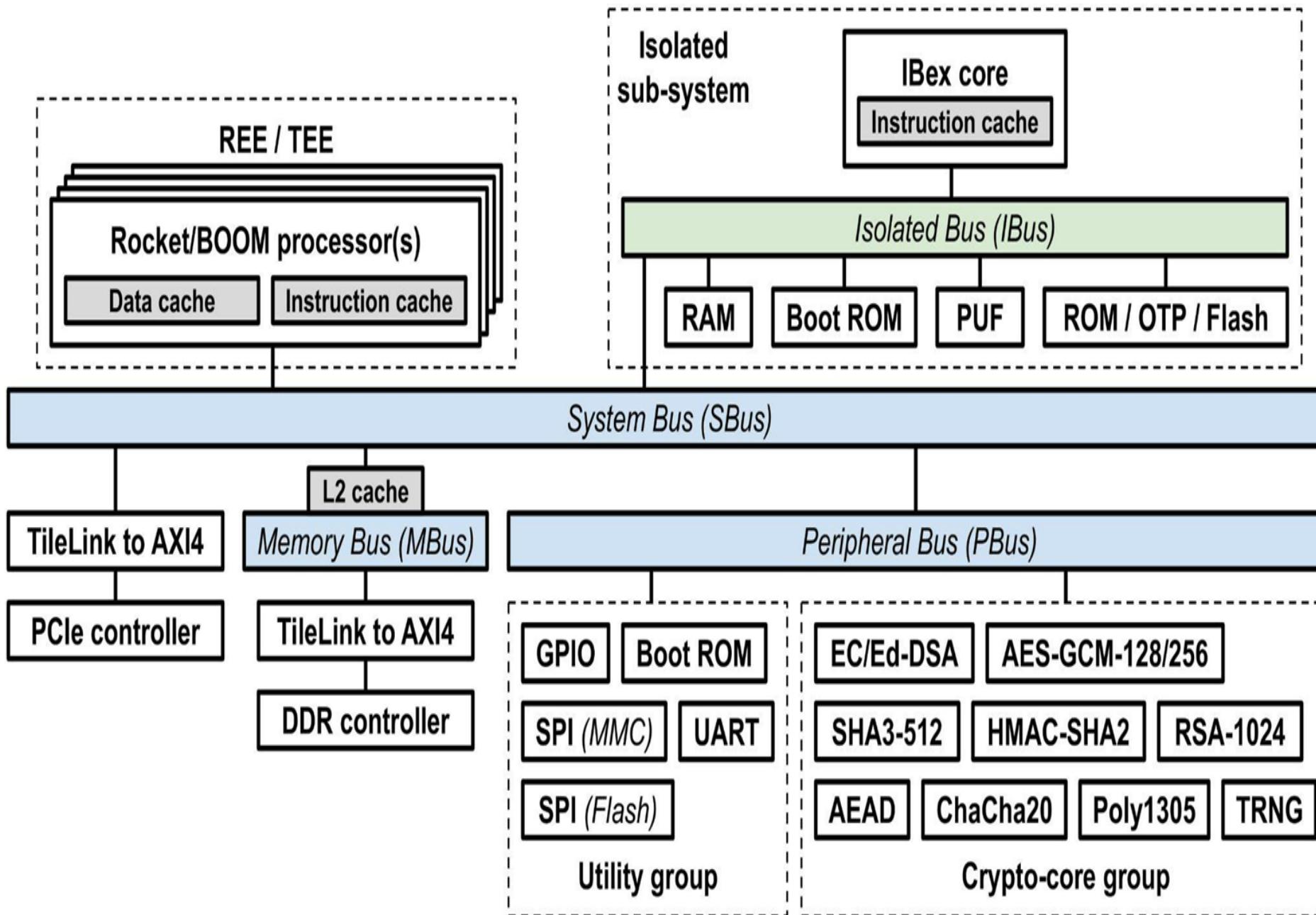
# The RISC-V Instruction Set

- Used as the example throughout the book
- Developed at UC Berkeley as open ISA
- Now managed by the RISC-V Foundation ([riscv.org](http://riscv.org))
- Typical of many modern ISAs
  - See RISC-V Reference Data tear-out card
- Similar ISAs have a large share of embedded core market
  - Applications in consumer electronics, network/storage equipment, cameras, printers, ...

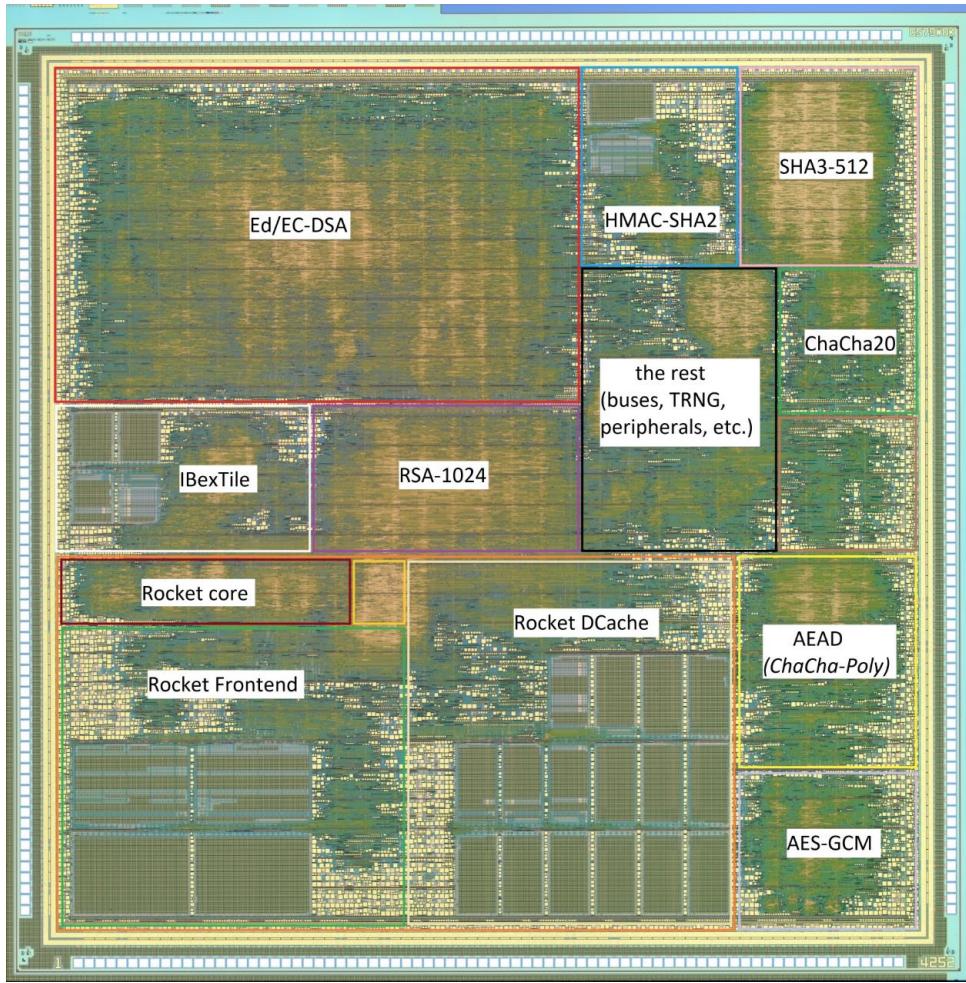
# The RISC-V Instruction Set

- The RISC-V project defines and describes a standardized Instruction Set Architecture (ISA).
- RISC-V is an open-source specification for computer processor architectures, not a particular chip or implementation.
- Several different groups have designed and fabricated silicon implementations of the RISC-V specifications.
- Based on the performance of these implementations and the growing need for interoperability among vendors, it appears that the RISC-V standard will increase in importance.

# Crypto-processor



# Crypto-processor



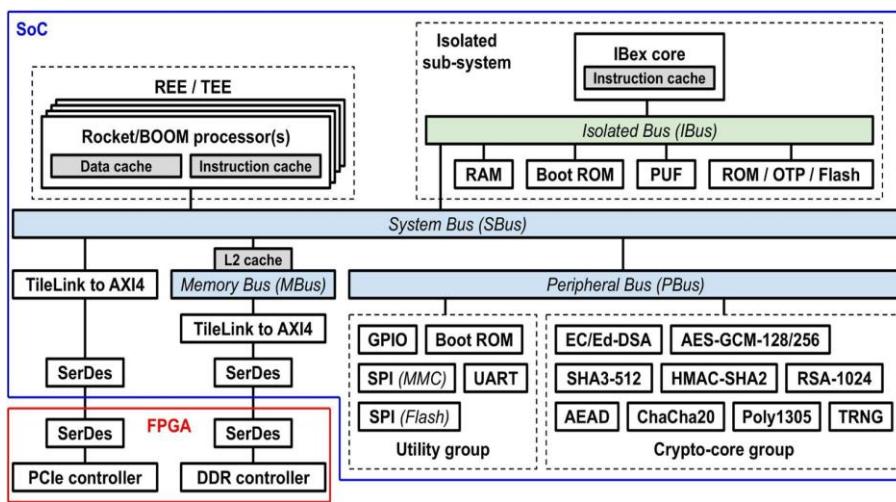
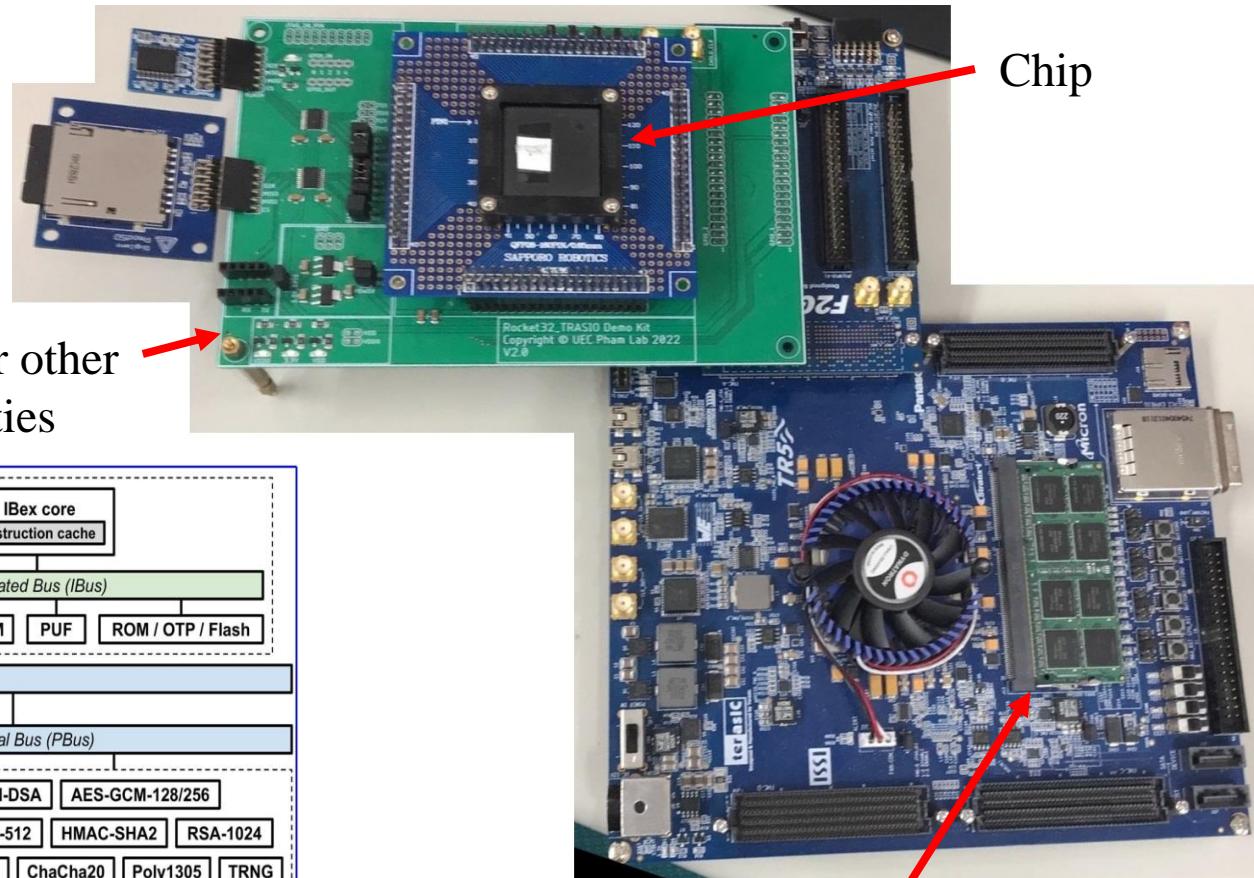
## Key features summarize

<b>Core</b>	Rocket ×1
<b>ISA</b>	RV32IMAC
<b>Cache</b>	\$I = 16KB and \$D = 16KB
<b>Crypto-cores:</b> TRNG, RSA, AES-GCM, SHA3, HMAC-SHA2, ChaCha20, Poly1305, AEAD, & EC/Ed-DSA	
@SYN	<b>#Cell</b>
	460,195
	<b>Area (<math>\mu\text{m}^2</math>)</b>
	14,744,115
	<b>SRAM (<math>\mu\text{m}^2</math>)</b>
	3,386,029
@PNR	<b>(%)</b>
	22.97%
	<b>Power (mW)</b>
	3,075
	<b>Fmax (MHz)</b>
	18
@Gate	<b>#Gate</b>
	1,535,403
	<b>#Cell</b>
	466,882
	<b>Area (<math>\mu\text{m}^2</math>)</b>
	20,799,437
	<b>Density</b>
@Power	<b>71.43%</b>
	<b>Power (mW)</b>
	1,992
@Fmax	<b>Fmax (MHz)</b>
	71
<b>#MOSFET</b>	
7,982,582	

Process: CMOS 180nm  
Feb. 2022

# Crypto-processor

The completed demo  
will look like this:



# 「Course」

# RISC-V Computer System Integration

## 「Lecture 1」 RISC-V Introduction

Phạm Công-Kha  
Hoàng Trọng-Thúc

Tháng 9 năm 2023

# Outline

1. Introduction
2. What makes a processor and computer system?
3. Instruction Set Architecture (ISA) and RISC-V
4. Necessary tools for working with RISC-V
5. RISC-V open-source community and materials
6. Some RISC-V news

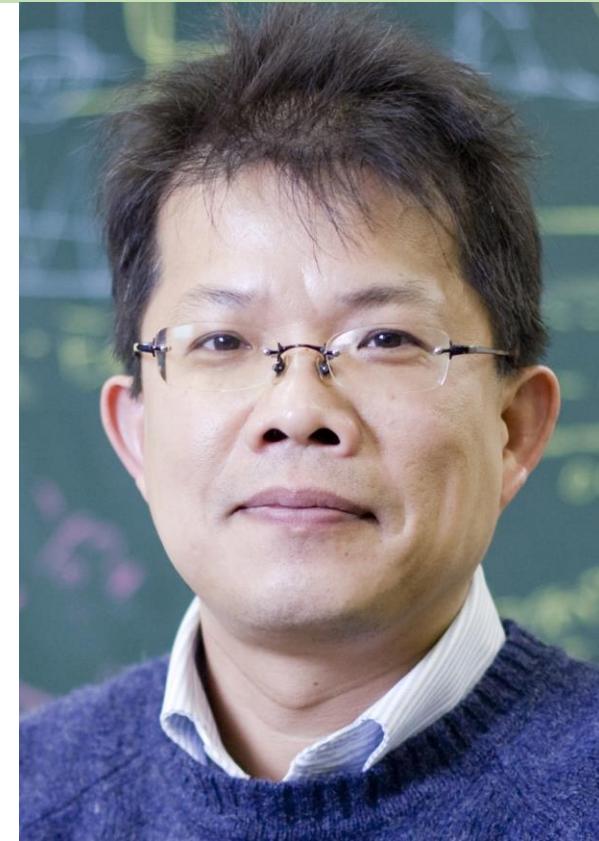
# Outline

1. Introduction
2. What makes a processor and computer system?
3. Instruction Set Architecture (ISA) and RISC-V
4. Necessary tools for working with RISC-V
5. RISC-V open-source community and materials
6. Some RISC-V news

# 1. Introduction (1/3)



Trong-Thuc Hoang  
Assistant Professor (UEC)  
[hoangtt@uec.ac.jp](mailto:hoangtt@uec.ac.jp)



Cong-Kha Pham  
Professor (UEC)  
[phamck@uec.ac.jp](mailto:phamck@uec.ac.jp)

<https://thuchoang90.github.io/>

(you can find tutorials and project sources on the website)

# 1. Introduction (2/3) University



国立大学法人  
**電気通信大学**  
The University of Electro-Communications

<https://www.uec.ac.jp/>

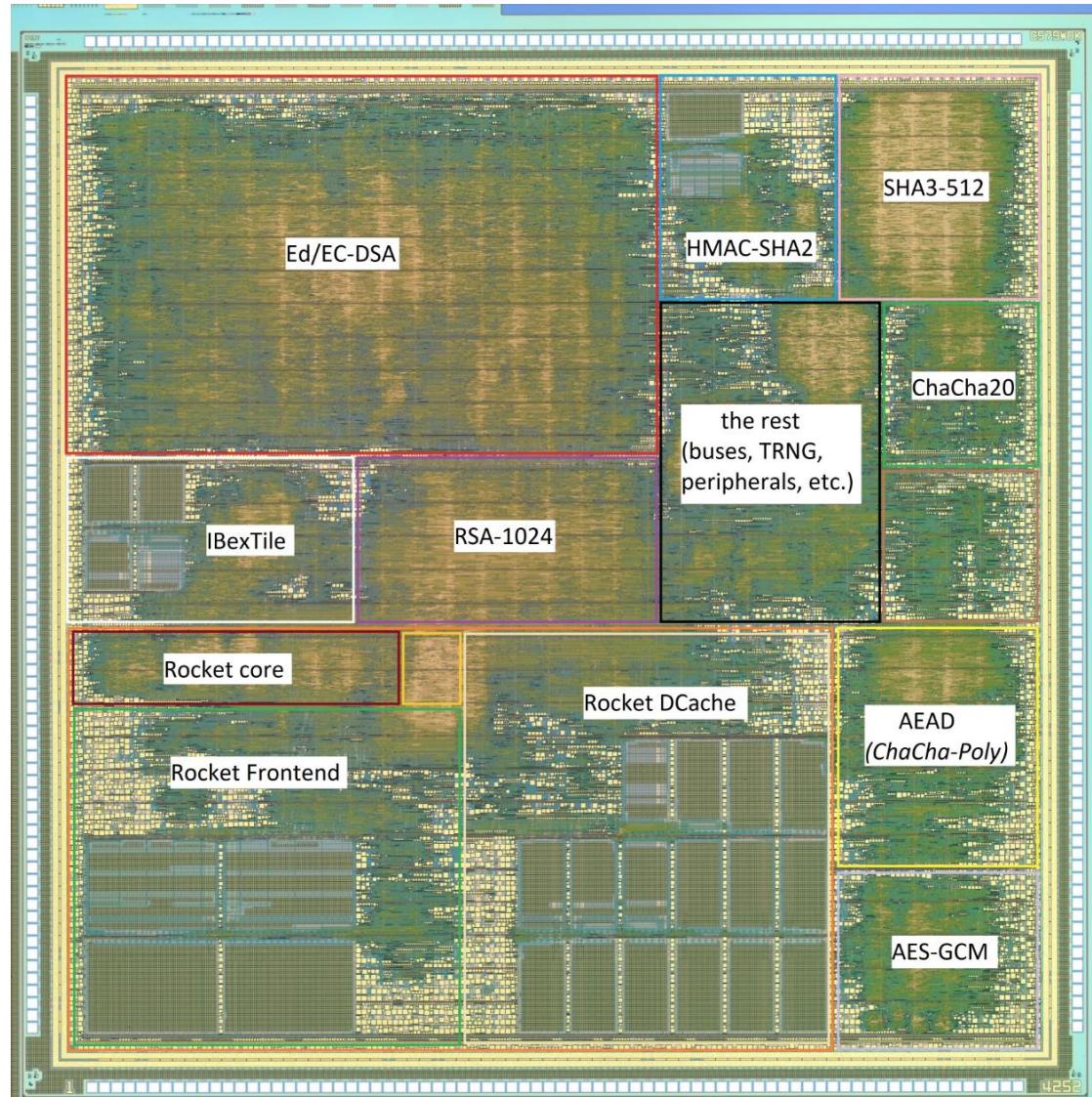


# 1. Introduction (3/3) Laboratory



<http://vlsilab.ee.uec.ac.jp/>

Pham Laboratory  
Integrated circuit design laboratory

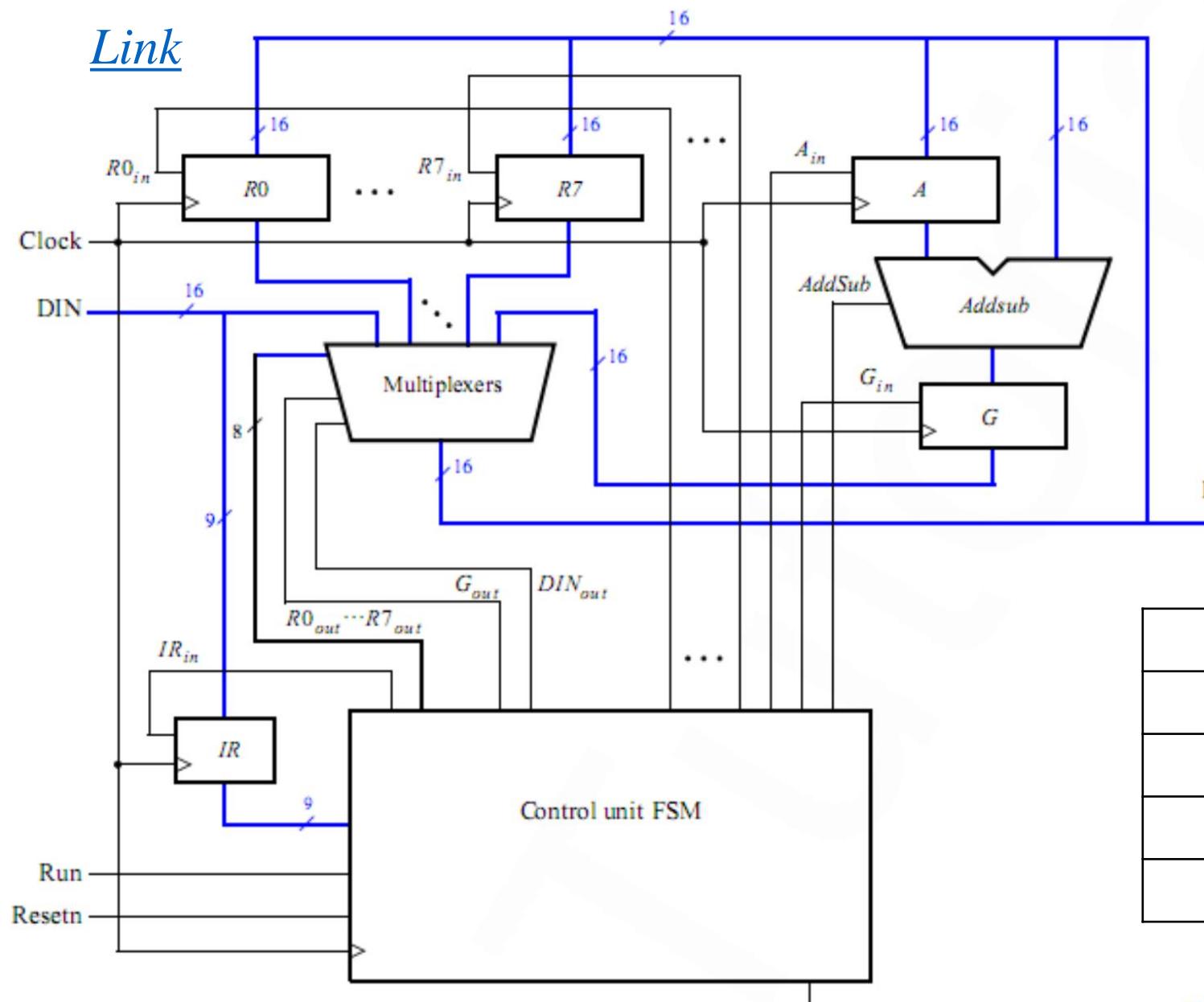


# Outline

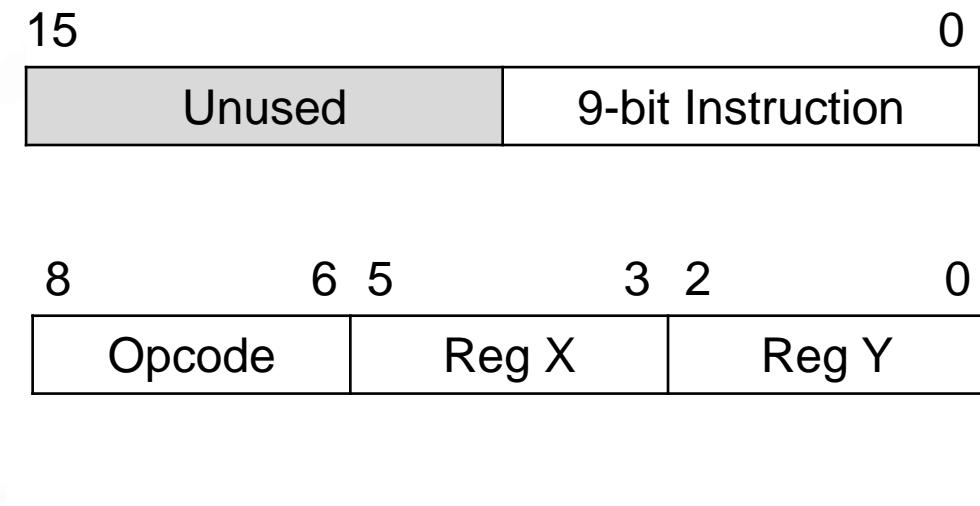
1. Introduction
2. What makes a processor and computer system?
3. Instruction Set Architecture (ISA) and RISC-V
4. Necessary tools for working with RISC-V
5. RISC-V open-source community and materials
6. Some RISC-V news

## 2. Processor and Computer System (1/19) Fundamental

Link



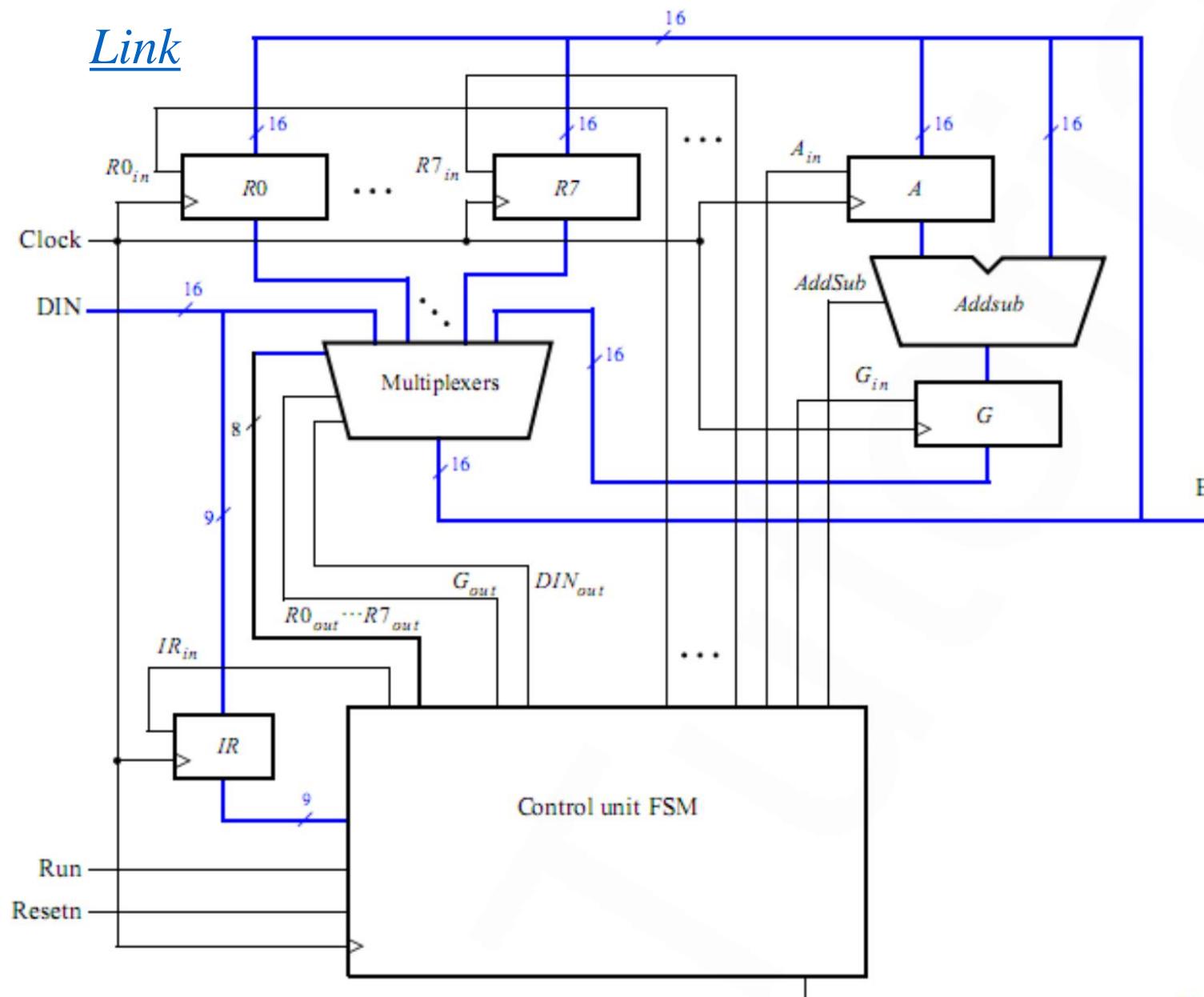
Int



Opcode	Assembly	Function
000	<b>mv X, Y</b>	$X \leftarrow [Y]$
001	<b>mvi X, #D</b>	$X \leftarrow D_{in}$
010	<b>add X, Y</b>	$X \leftarrow [X] + [Y]$
011	<b>sub X, Y</b>	$X \leftarrow [X] - [Y]$

## 2. Processor and Computer System (2/19) Fundamental

## Link



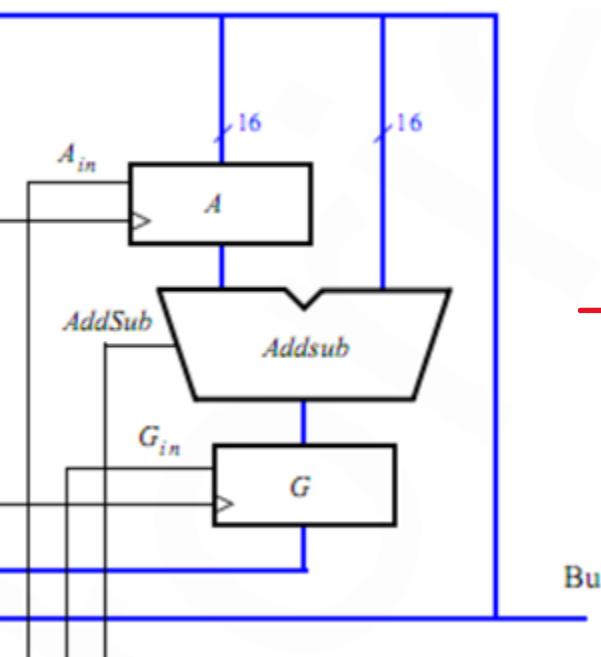
Int

Opcode	Assembly	Function
000	<b>mv</b> $X, Y$	$X \leftarrow [Y]$
001	<b>mvi</b> $X, \#D$	$X \leftarrow D_{in}$
010	<b>add</b> $X, Y$	$X \leftarrow [X] + [Y]$
011	<b>sub</b> $X, Y$	$X \leftarrow [X] - [Y]$

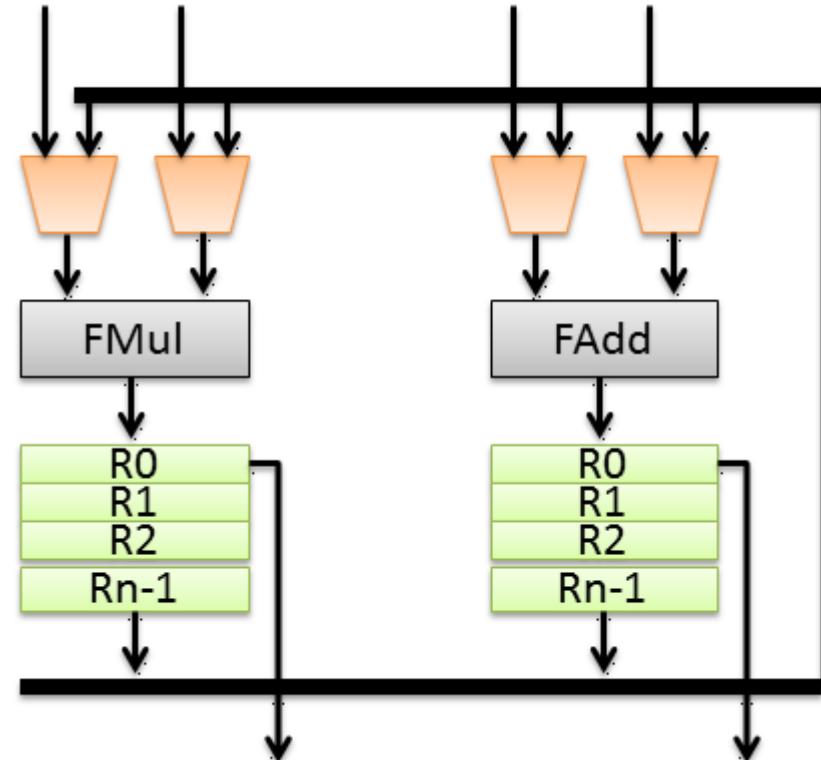
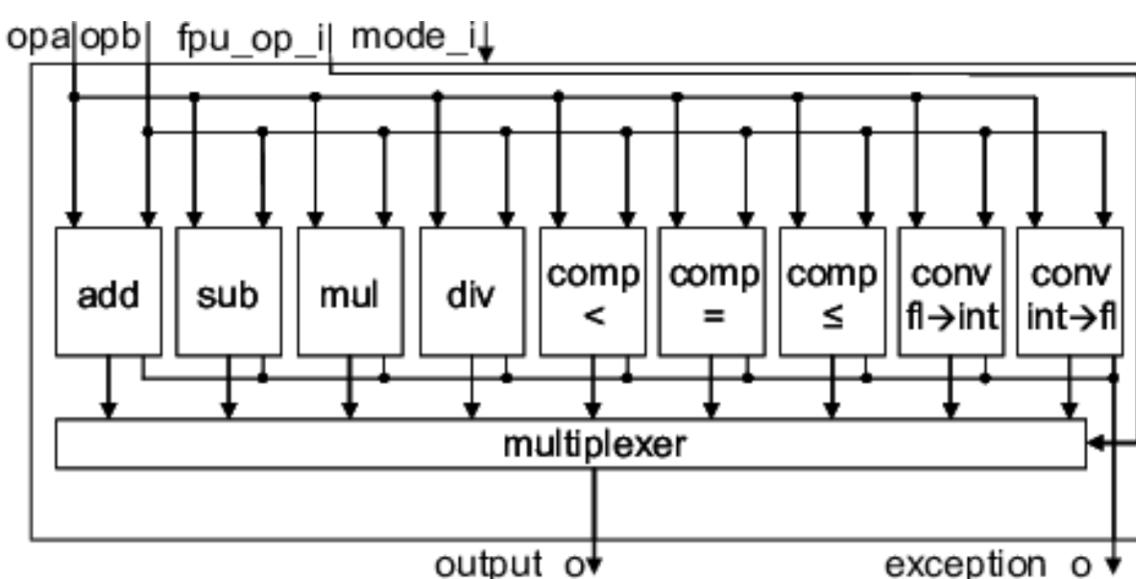
Bus

	T1	T2	T3
<b>mv</b>	Y:out, X:in Done		
<b>mvi</b>	D/N:out, X:in Done		
<b>add</b>	X:out, A:in	Y:out, G:in	G:out, X:in Done
<b>sub</b>	X:out, A:in	Y:out, G:in AddSub	G:out, X:in Done

## 2. Processor and Computer System (3/19) Mul & FPU



**Upgrade:** ALU with multiplication  
**Add:** Floating-Point Unit (FPU)

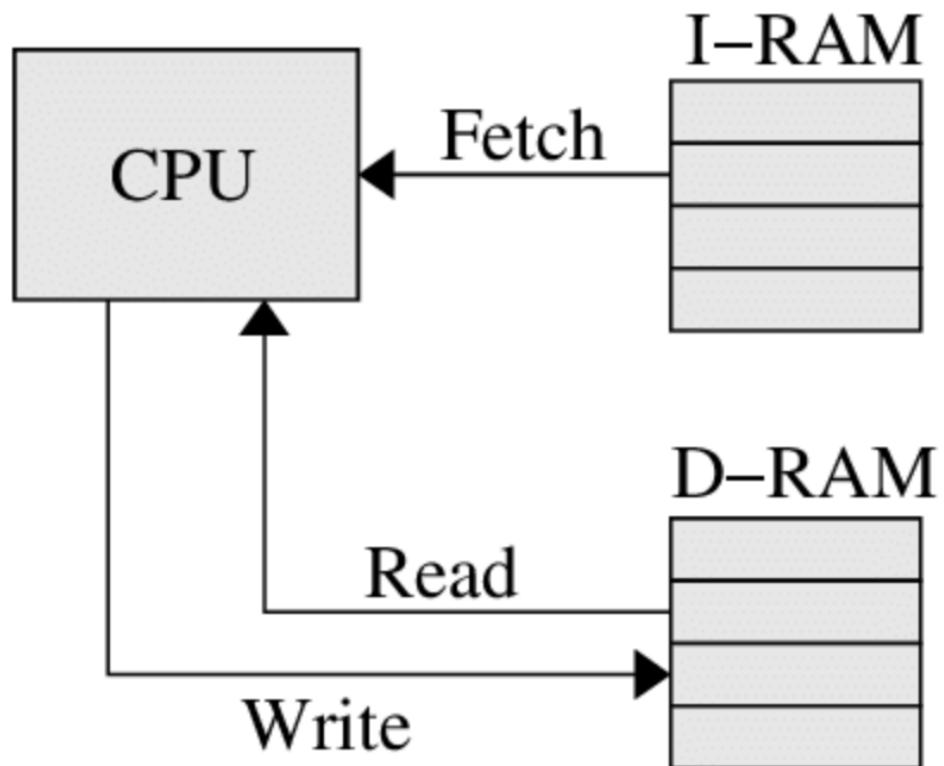


## 2. Processor and Computer System (4/19) Memory

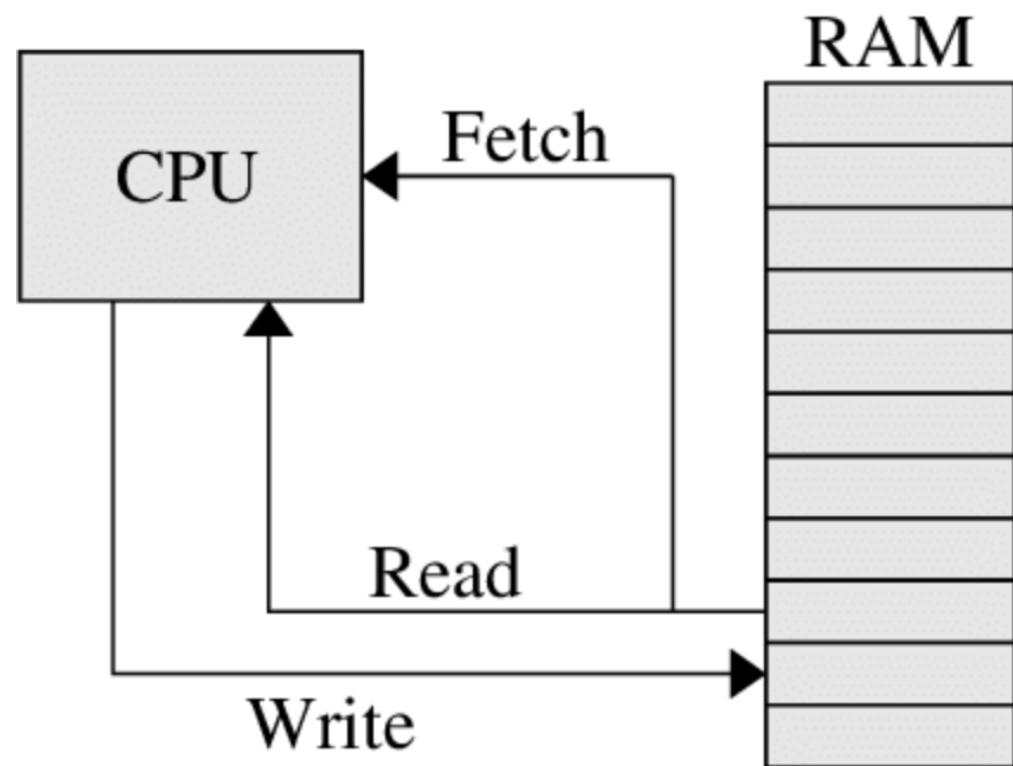
Int Mul

Float Double

Nowadays, they are all **Von Neumann** :  
It is easier to manage the addresses



**Harvard**  
memory architecture

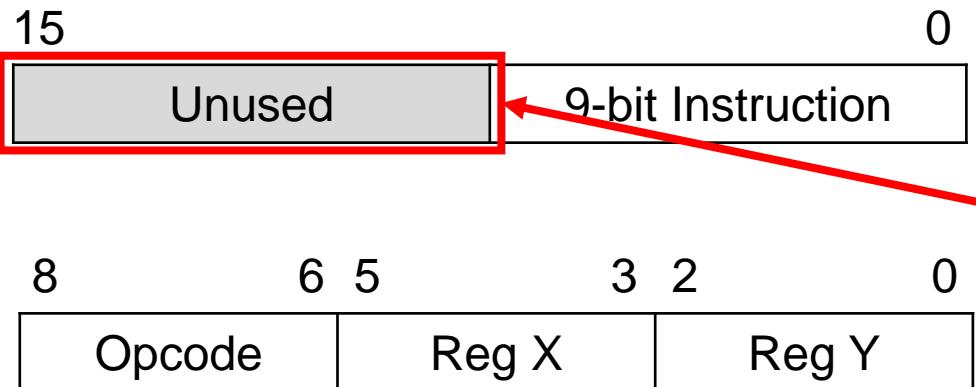


**Von Neumann**  
memory architecture

## 2. Processor and Computer System (5/19) Compressed

Int Mul

Float Double Compress



Sometimes the instruction did not use all the fields.

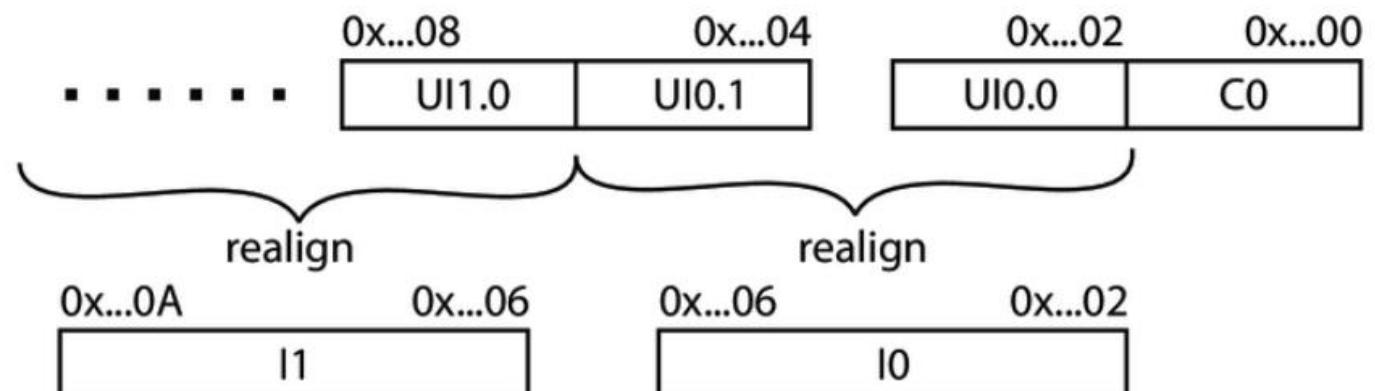
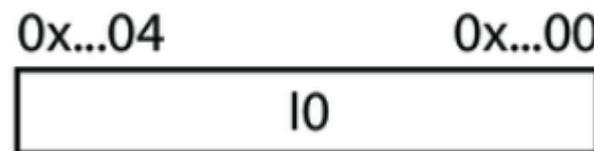
Opcode	Assembly	Function
000	<b>mv X, Y</b>	$X \leftarrow [Y]$
001	<b>mvi X, #D</b>	$X \leftarrow D_{in}$
010	<b>add X, Y</b>	$X \leftarrow [X] + [Y]$
011	<b>sub X, Y</b>	$X \leftarrow [X] - [Y]$

## 2. Processor and Computer System (6/19) Compressed

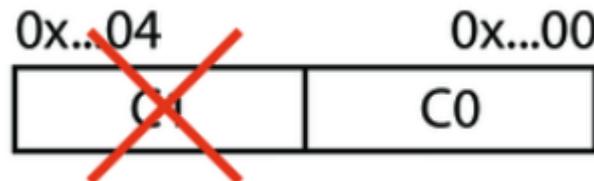
Int Mul

Float Double Compress

Regular instruction:



Compressed instruction:

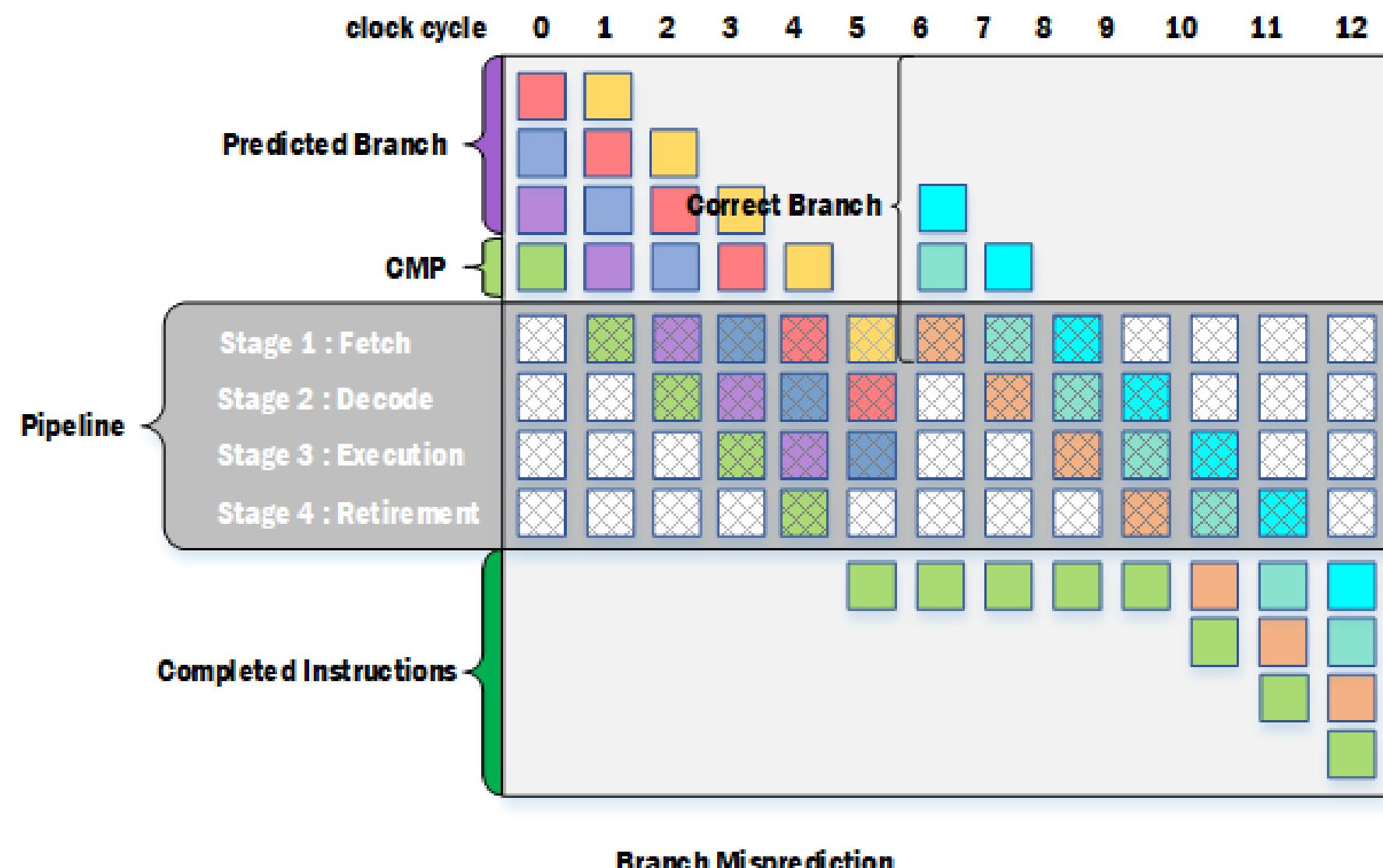


Need an instruction decoder  
(uncompress) to re-align the instructions  
before fetching to the core

## 2. Processor and Computer System (7/19) Branch prediction

Int Mul

Float Double Compress

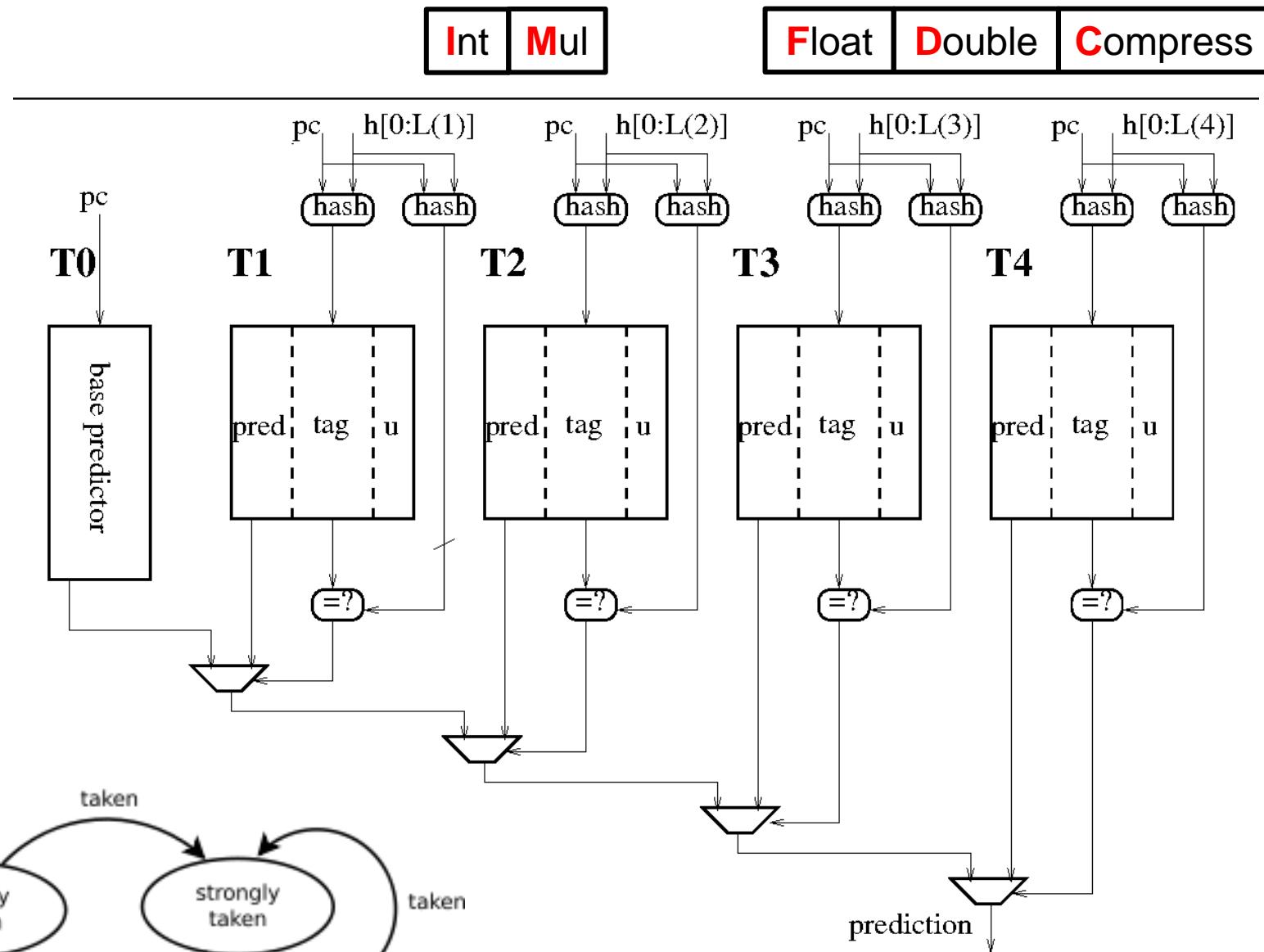
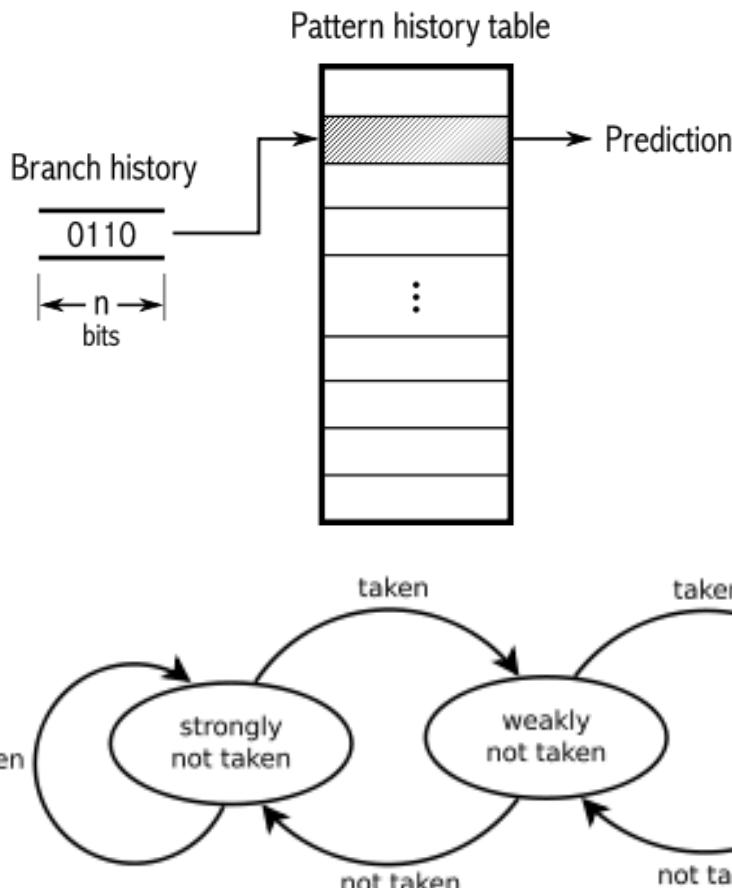


The idea is simple:  
(but the implementation  
is not)

While waiting for the  
correct answer to come  
in, compute ahead.

## 2. Processor and Computer System (8/19) Branch prediction

- With branch prediction:  
**out-of-order** processor
- Without branch prediction:  
**in-of-order** processor



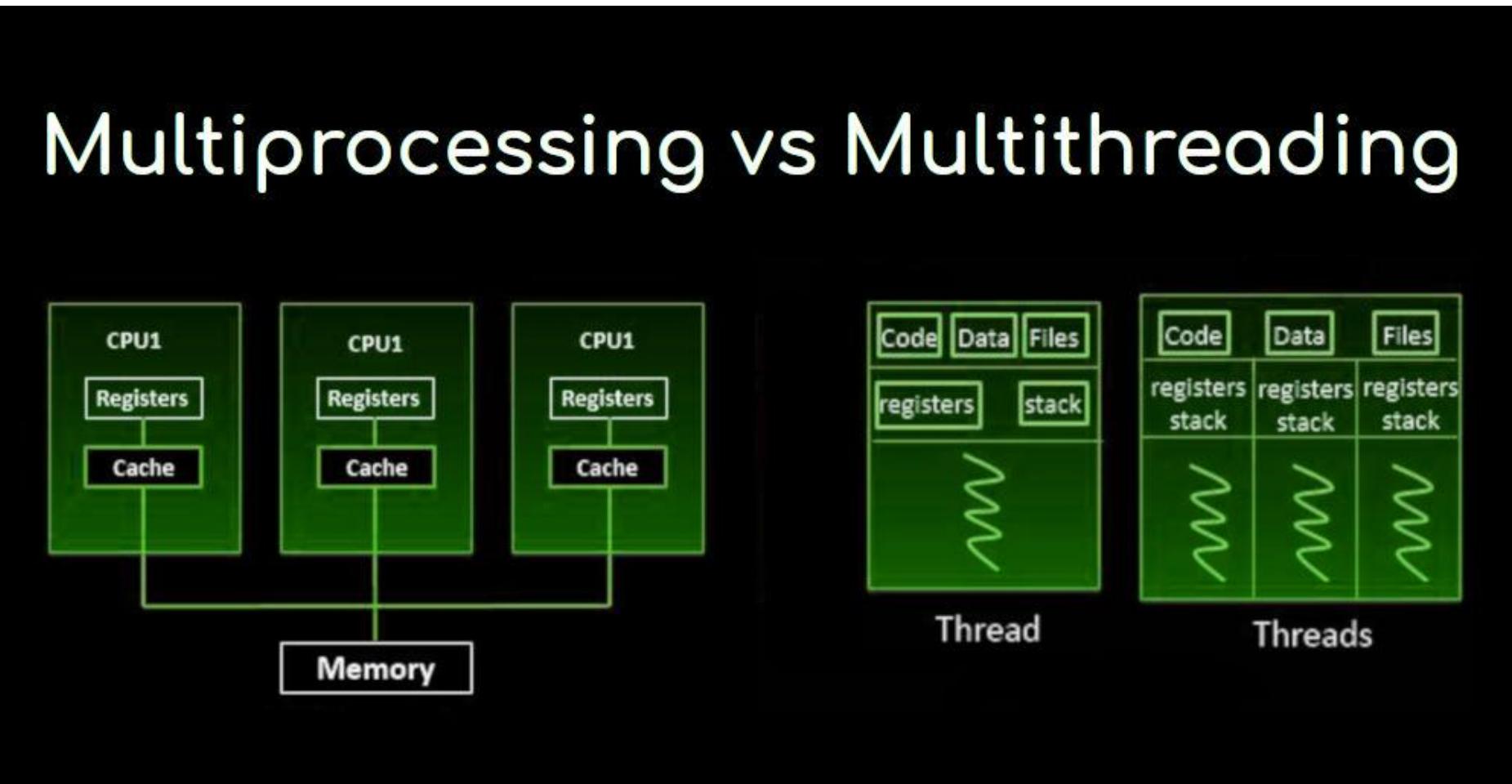
## 2. Processor and Computer System (9/19) Multi-threading

Int Mul

Float Double Compress

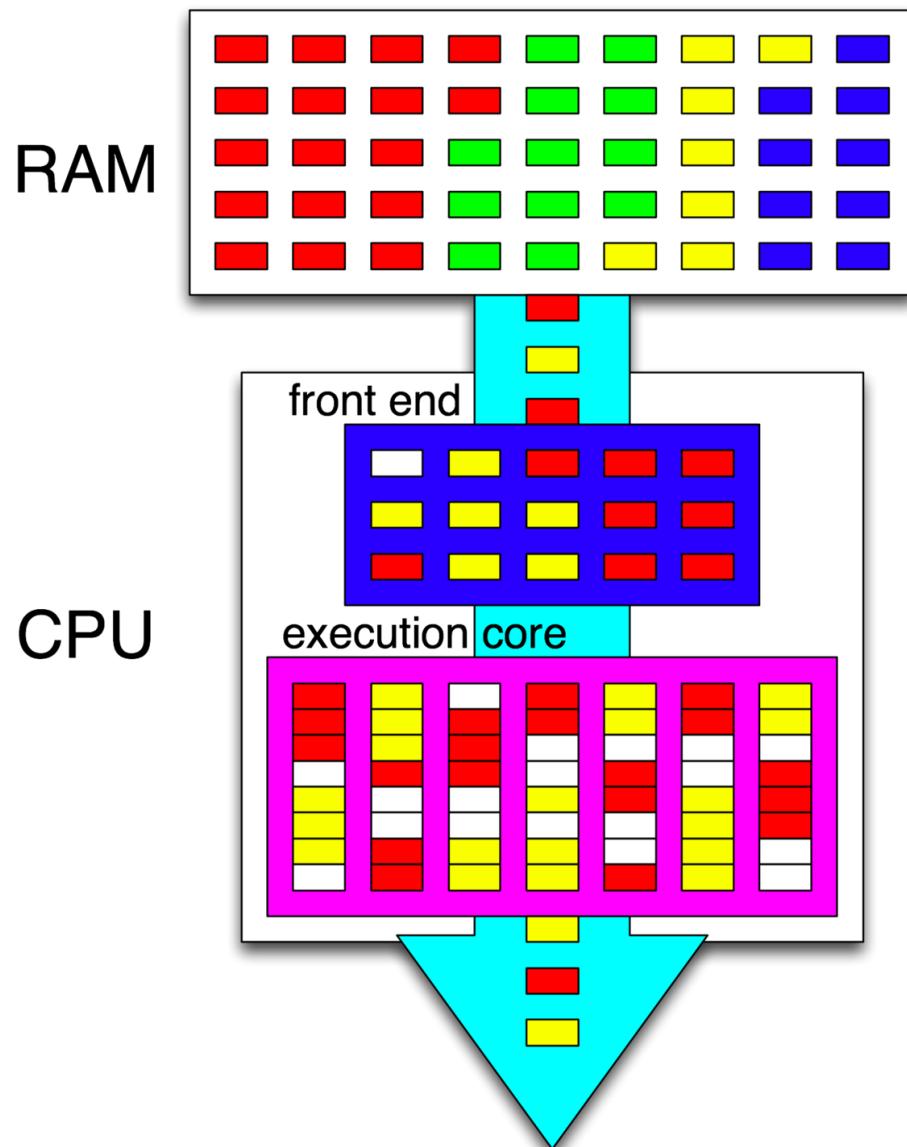
Multiprocessing  
is a hardware  
point-of-view

Multithreading is  
a software point-  
of-view



- One process could set up multiple threads
- One core could run many threads of many processes “at the same time”

## 2. Processor and Computer System (10/19) Multi-threading



Int Mul

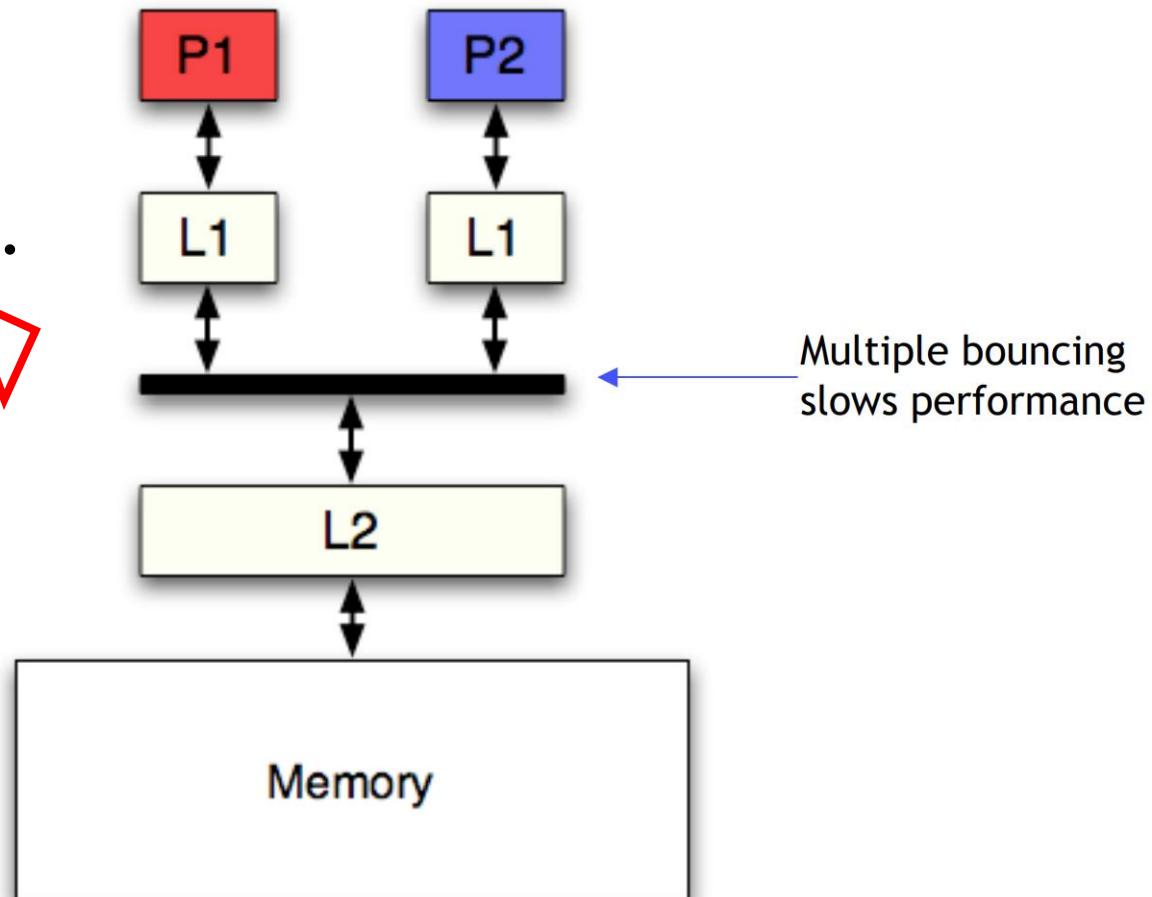
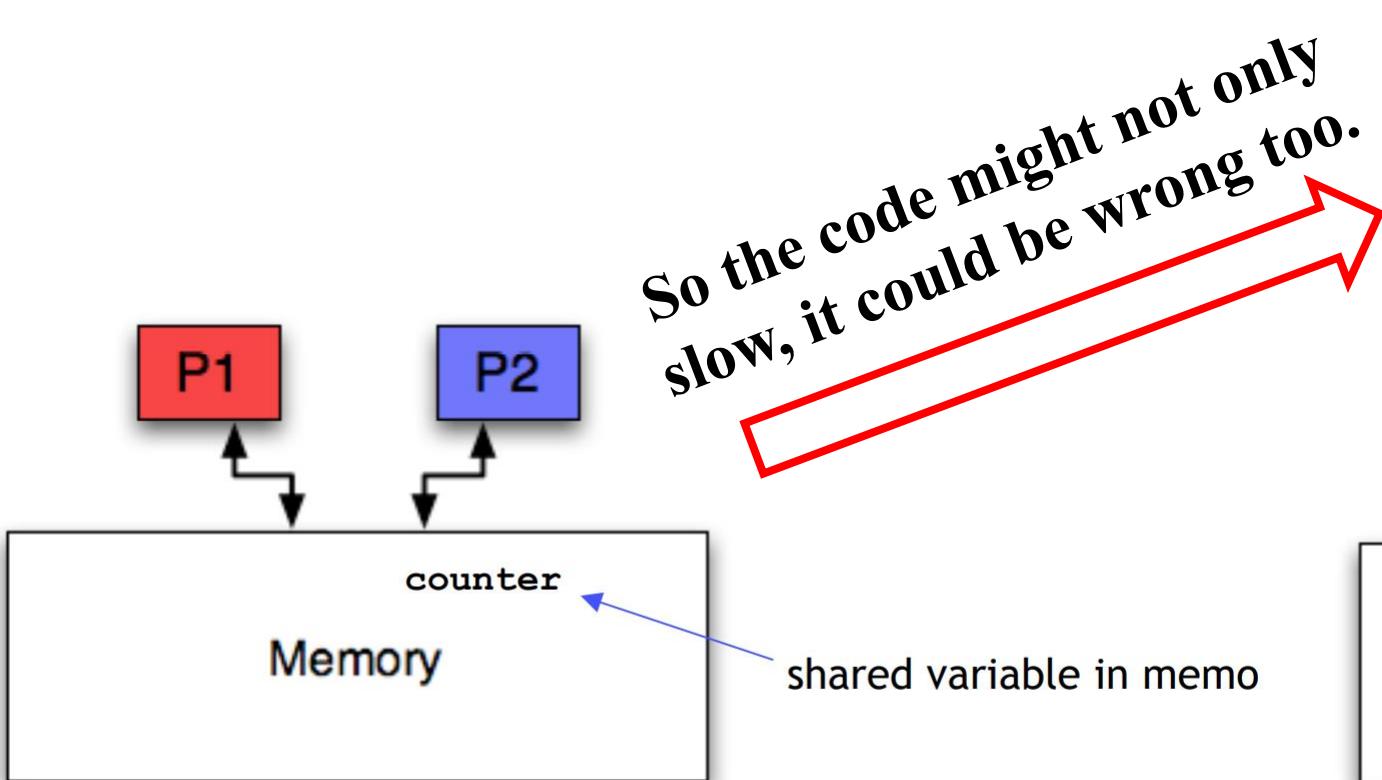
Float Double Compress

- Multi-threading is managed by the Operating System (OS)
- It is the very feature that makes the modern rich OSes like Windows & Linux
- The key hardware modules required for this feature to work are the *Scheduler* and the *Interrupt Handler* (*especially the local interrupt between cores*)

## 2. Processor and Computer System (11/19) Atomic

Int	Mul	Atomic	Float	Double	Compress
-----	-----	--------	-------	--------	----------

Problem with parallelism is the synchronization



What you thinking when coding

What it actually is

## 2. Processor and Computer System (12/19) Atomic

Int Mul Atomic Float Double Compress

Problem with parallelism is the synchronization

Without <i>atomic</i>		
Clock	Thread 1	Thread 2
1	Load	
2	Execute	Load
3	Store	Execute
4		Store

With <i>atomic</i>		
Clock	Thread 1	Thread 2
1	Load	<i>Locked</i>
2	Execute	<i>Locked</i>
3	Store	<i>Locked</i>
4	<i>Locked</i>	Load
5	<i>Locked</i>	Execute
6	<i>Locked</i>	Store

To fix this, we must do the “**load-compute-store**” in a single step.

That calls *atomic* instruction.

→ Meaning: “Do this, and don’t get interrupted while doing this”

So, the nature of *atomic* instructions is the “**lock**” mechanism.

→ One core could keep the other cores at a halt and wait for synchronization.

To do the “**lock**” mechanism: (1) the core must support it, and (2) the system must have some local interrupt between cores.

Nowadays, a processor **without atomic** instructions means **no Windows/Linux support**

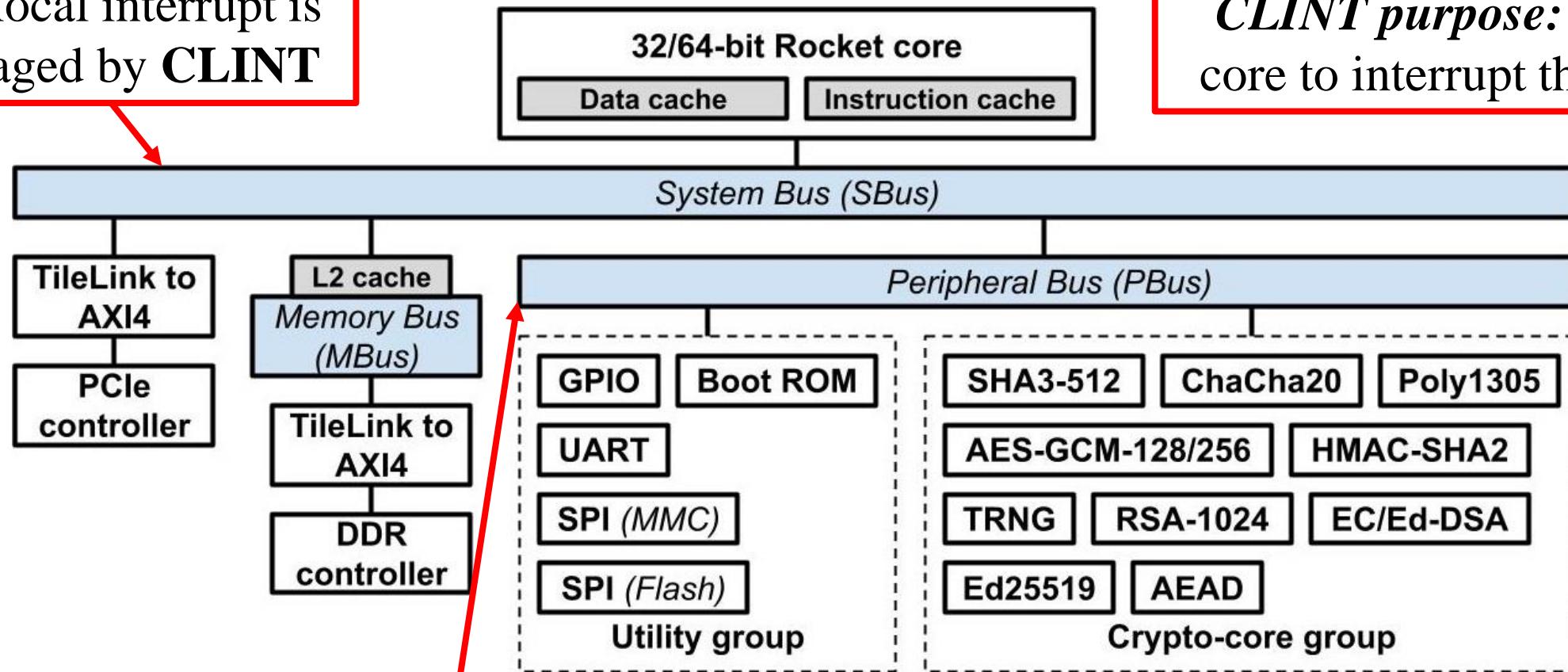
## 2. Processor and Computer System (13/19) Interrupts

Example of a computer system:

Int	Mul	Atomic	Float	Double	Compress
-----	-----	--------	-------	--------	----------

The local interrupt is managed by **CLINT**

**CLINT purpose:** for one core to interrupt the others

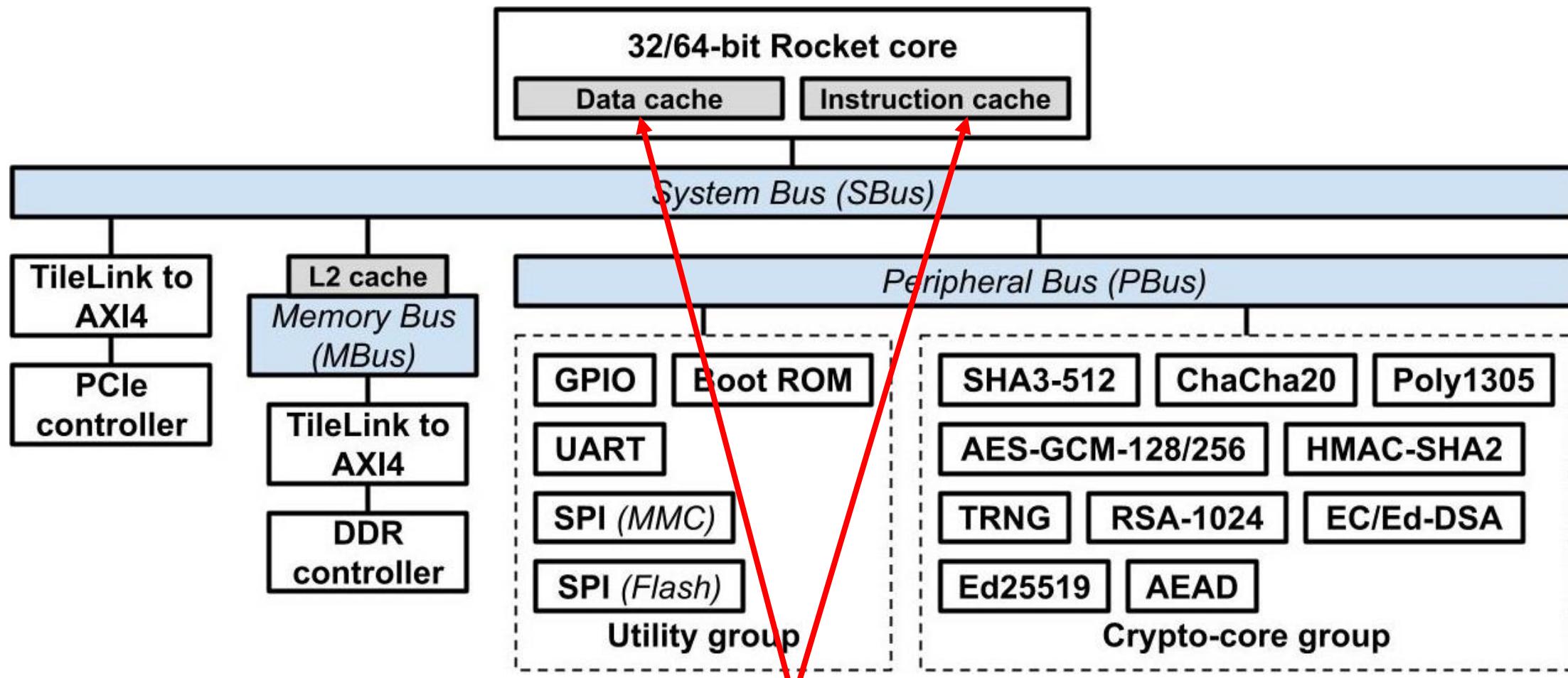


Global interrupt is managed by **PLIC**

**PLIC purpose:** for one module (*usually peripheral*) to interrupt the cores

## 2. Processor and Computer System (14/19) Caches

Int	Mul	Atomic	Float	Double	Compress
-----	-----	--------	-------	--------	----------



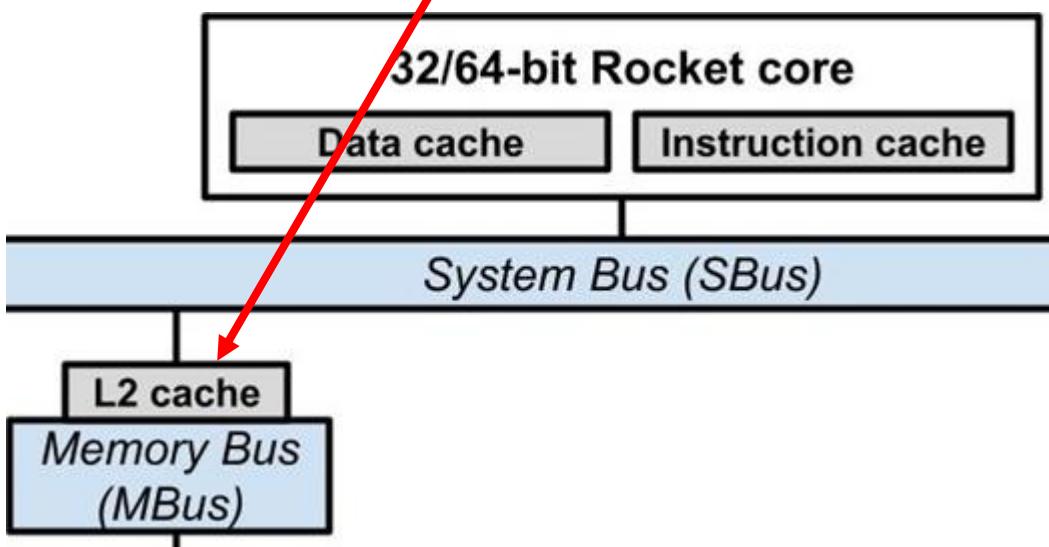
L1 (Level 1) cache always means instruction cache and data cache inside each core

## 2. Processor and Computer System (15/19) Caches

Int	Mul	Atomic	Float	Double	Compress
-----	-----	--------	-------	--------	----------

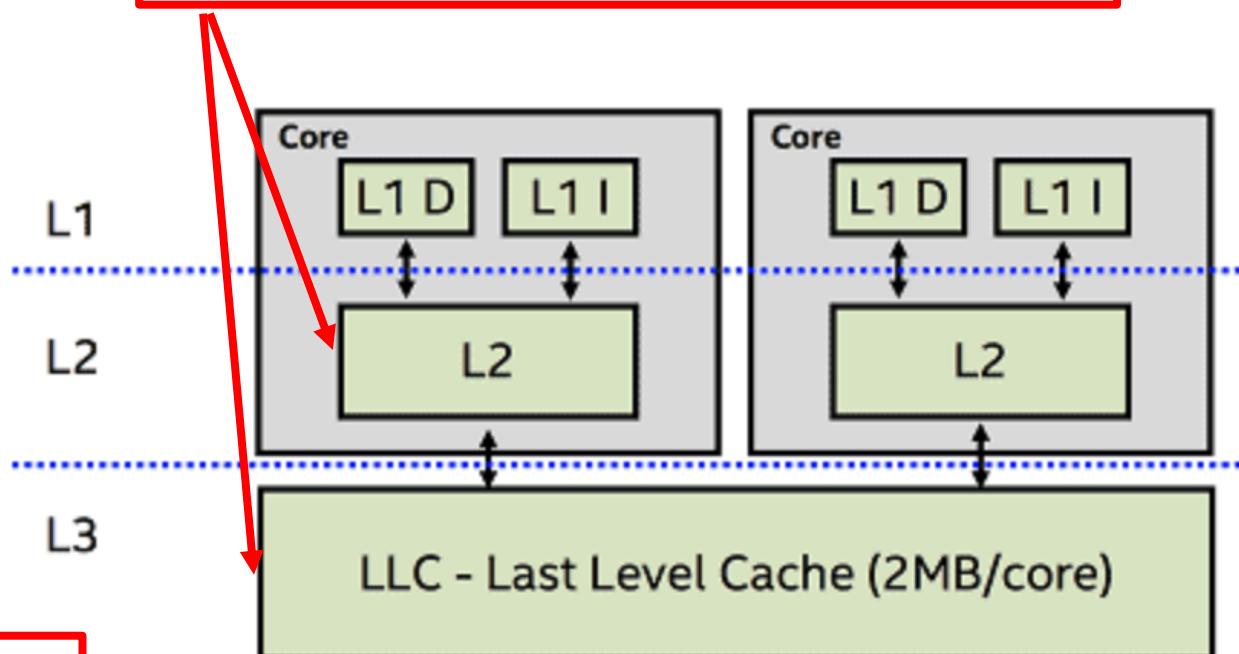
If a system has two-level caches:

- L2 cache means memory cache



If a system has three-level caches:

- L2 cache means shared cache for both data & instruction caches
- L3 means memory cache



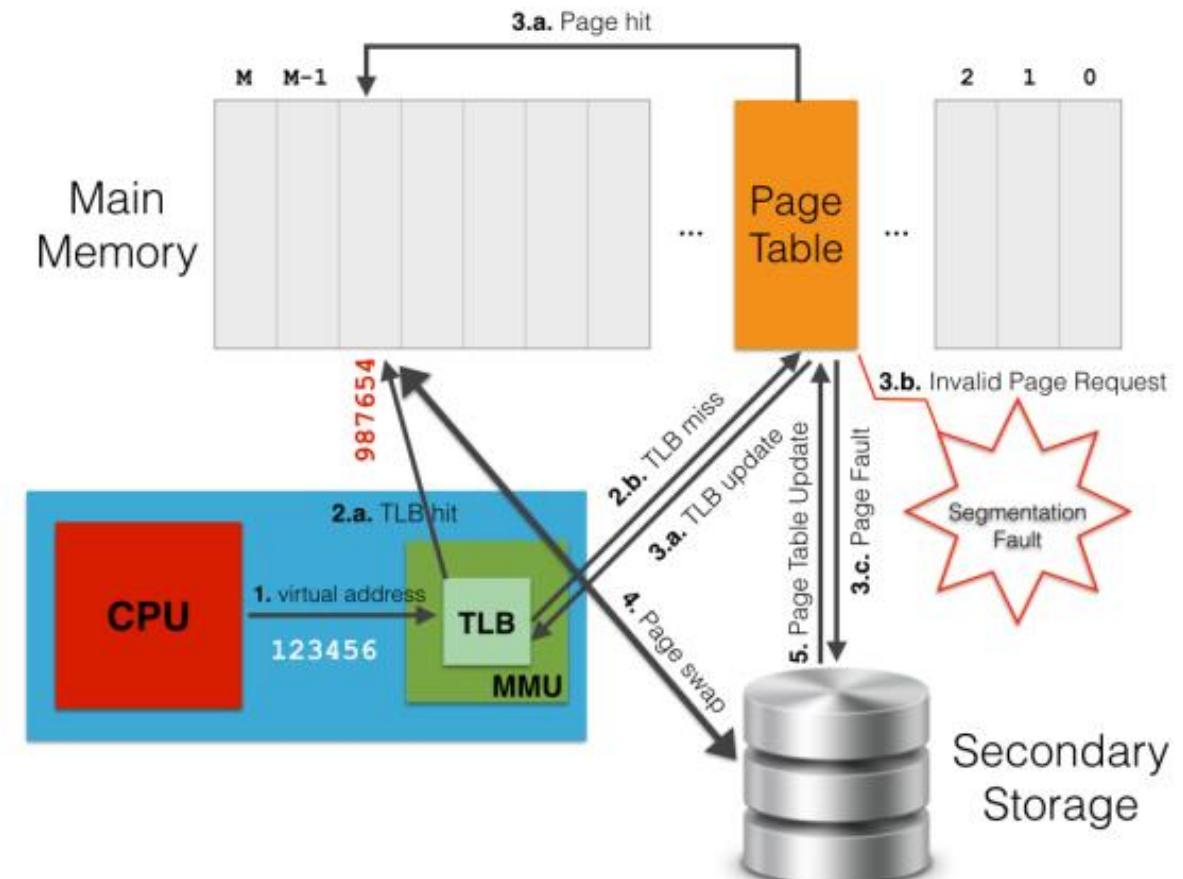
\*Note: peripheral bus must not have any caches, or else the system will fail to control the IOs

## 2. Processor and Computer System (16/19) Virtual memory

Int	Mul	Atomic	Float	Double	Compress
-----	-----	--------	-------	--------	----------

**Virtual memory** is the critical function in Rich OSes.  
Without virtual memory, Windows/Linux cannot function.  
In hardware, it must have the **MMU** and **TLB**.

- Memory Management Unit (**MMU**): translates addresses from virtual to physical.
- Operating System (**OS**): sets up virtual address spaces for each process/program.
- **Page table**: a data structure for the **OS** to manage and store the mapping between virtual and physical addresses.
- Translation Lookaside Buffer (**TLB**): cache for the page table.



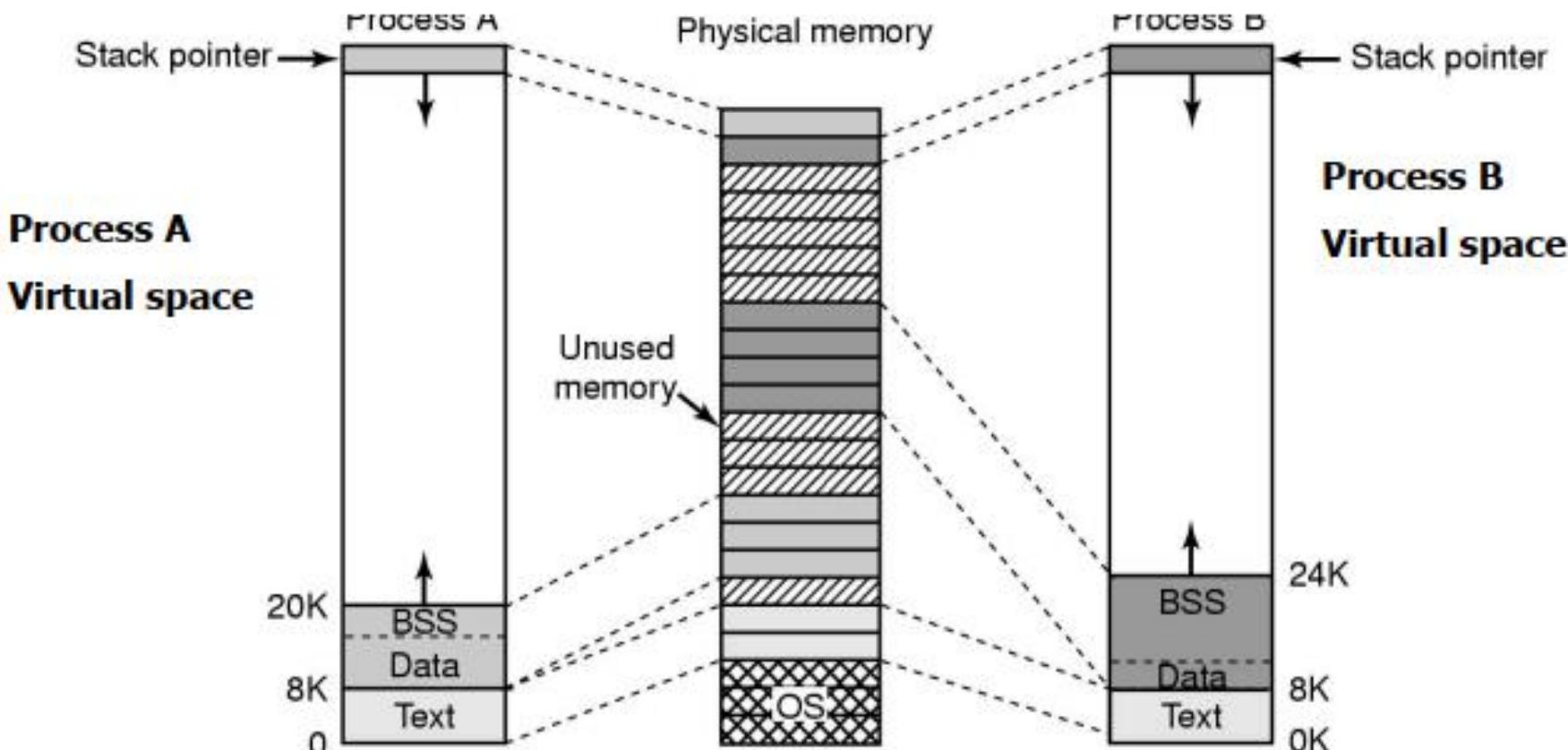
## 2. Processor and Computer System (17/19) Virtual memory

Int Mul Atomic Float Double Compress

**Virtual memory** allows users to use more than the *actual memory capacity*.

It also increases the system's security due to its isolated nature.

- Each process has its own virtual space of memory.
- Physical memory transactions are mostly (*and should be*) done by the **OS** automatically.



## 2. Processor and Computer System (18/19) RISC-V instructions



In RISC-V architecture, we begin with the Integer as the base instruction set.  
Then we have a bunch of extensions.

The most common extensions: **MAFDC**

**When they wrote**

RV64IMAFDC :

RV32IMAFDC :

**What it means**

RISC-V 64-bit I M A F D C

RISC-V 32-bit I M A F D C

**\*Note:**

IMAFD also called G (*stands for General*)

→ So RV64IMAFDC can be written as RV64GC

## 2. Processor and Computer System (19/19) RISC-V instructions

Int	Mul	Atomic	Float	Double	Compress
-----	-----	--------	-------	--------	----------

What makes **RISC-V** different: its modular mindset

*(modular architecture helps fine-tune the performance based on the developer's needs)*

Base instruction set: **Integer**

Extended instruction set: *the rest*

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"

The most common extensions: **IMAFDC**  
*(also known as GC)*

There are also a lot more than just **IMAFDC** :

Base	Version	Status
RVWMO	2.0	Ratified
RV32I	2.1	Ratified
RV64I	2.1	Ratified
RV32E	1.9	Draft
RV128I	1.7	Draft
Extension	Version	Status
M	2.0	Ratified
A	2.1	Ratified
F	2.2	Ratified
D	2.2	Ratified
Q	2.2	Ratified
C	2.0	Ratified
Counters	2.0	Draft
L	0.0	Draft
B	0.0	Draft
J	0.0	Draft
T	0.0	Draft
P	0.2	Draft
V	0.7	Draft
Zicsr	2.0	Ratified
Zifencei	2.0	Ratified
Zam	0.1	Draft
Ztso	0.1	Frozen

# Outline

1. Introduction
2. What makes a processor and computer system?
3. **Instruction Set Architecture (ISA) and RISC-V**
4. Necessary tools for working with RISC-V
5. RISC-V open-source community and materials
6. Some RISC-V news

### 3. ISA and RISC-V (1/9) ISA

ISA means Instruction Set Architecture.

It is the layer between software and hardware developers.

**Software tools:** assembler, compilers, debugger, linker, etc.

---

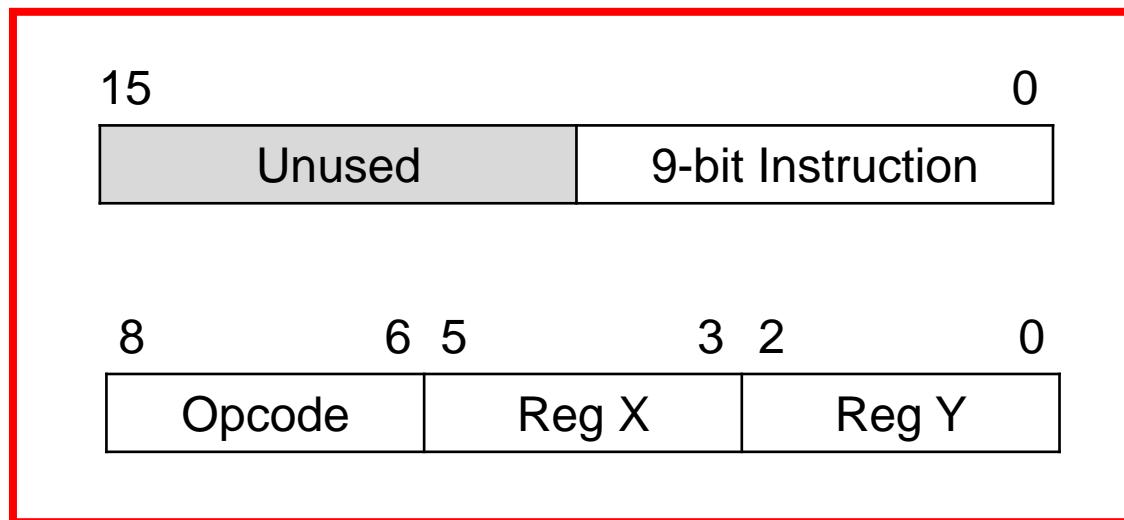
**ISA:** the interface between software & hardware architects

---

**Processor:** ALU, FPU, registers, CSRs, branch predictor, caches, etc.

**ISA has to define all these kinds of stuff:**

- 1) How many instructions, and which is which?
- 2) In an instruction, what field means what?
- 3) Addressing & data-path (8/16/32/64/128-bit)?
- 4) What is supported and what is not?
- 5) etc.



### 3. ISA and RISC-V (2/9) RISC-V ISA

Open-source **RISC-V** means open-source **ISA**, no more, no less.

(*some other common ISAs: i386, amd64, ARM 32/64, AVR, MIPS, NiosII, etc.*)

**RISC-V Foundation:** <https://riscv.org/>

RISC-V Exchange: Available Software

- Simulators
- Object Toolchain
- Debugging
- C Compilers & Libraries
- Bootloaders & Monitors
- Hypervisors
- OS & Kernels
- Non-C Compilers/Runtimes
- IDEs & SDKs
- Security
- Machine Learning & AI
- Configuration
- Verification Tools
- Accelerated Libraries

RISC-V Exchange: Cores & SoCs

Name	Supplier	Links	Capability	Priv. spec	User spec
RV32EC_P2	IQonIC Works	<a href="#">Website</a>	RV32	1.11	RV32E[M]C/RV32I

- Official released ISA specification
- Many cores, SoCs, & software are available for free
- Developers can reuse each other designs & tools  
→ significantly reducing R&D time and effort

- License free:**
- RISC-V ISA
  - RISC-V toolchain

**License depends on authors/developers:**

- RISC-V processors
- RISC-V software applications
- RISC-V-related products

### 3. ISA and RISC-V (3/9) CISC vs. RISC

CISC

## (Complex Instruction Set Computer)

- 1) Emphasis on *hardware*
  - 2) *Multi-clock* complex instructions
  - 3) *Memory-to-memory* mindset
  - 4) *Small* code size, *many* cycles per instruction
  - 5) *Low Fmax* due to complex design
  - 6) Most transistors are used for storing *instructions*
  - 7) *Less memory* for storing data & program

RISC

# (Reduced Instruction Set Computer)

- 1) Emphasis on *software*
  - 2) *Single-clock* simple instructions
  - 3) *Register-to-register* mindset
  - 4) *Large* code size, *few* cycles per instruction
  - 5) *High Fmax* due to simple design
  - 6) Most transistors are used for storing *data*
  - 7) *More memory* for storing data & program

$$\text{Performance} = \frac{\text{time}}{\text{program}} = \frac{\text{time}}{\text{cycle}} \times \frac{\text{cycle}}{\text{instruction}} \times \frac{\text{instruction}}{\text{program}}$$

RISC win      CISC win

**RISC-V** simply means *RISC* architecture version *five*

Nowadays, almost all processors in the market are RISCs.

**Economic reason:** memory  
price is way down ↓ ↓ ↓

### 3. ISA and RISC-V (4/9) OS stack

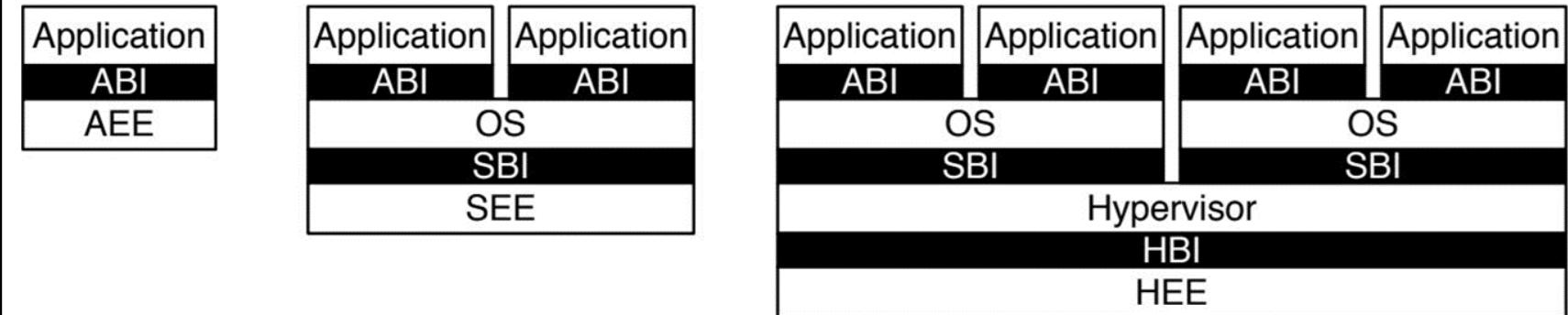
To support an Operating System (OS), the ISA has to support the OS stack or the **M-/S-/U-mode**.

RISC-V privileged architecture:

RISC-V Modes		
Level	Name	Abbr.
0	User/Application	U
1	Supervisor	S
Reserved		
3	Machine	M

Supported Combinations of Modes	
Supported Levels	Modes
1	M
2	M, U
3	M, S, U

Different scenarios of utilizing the OS stack:

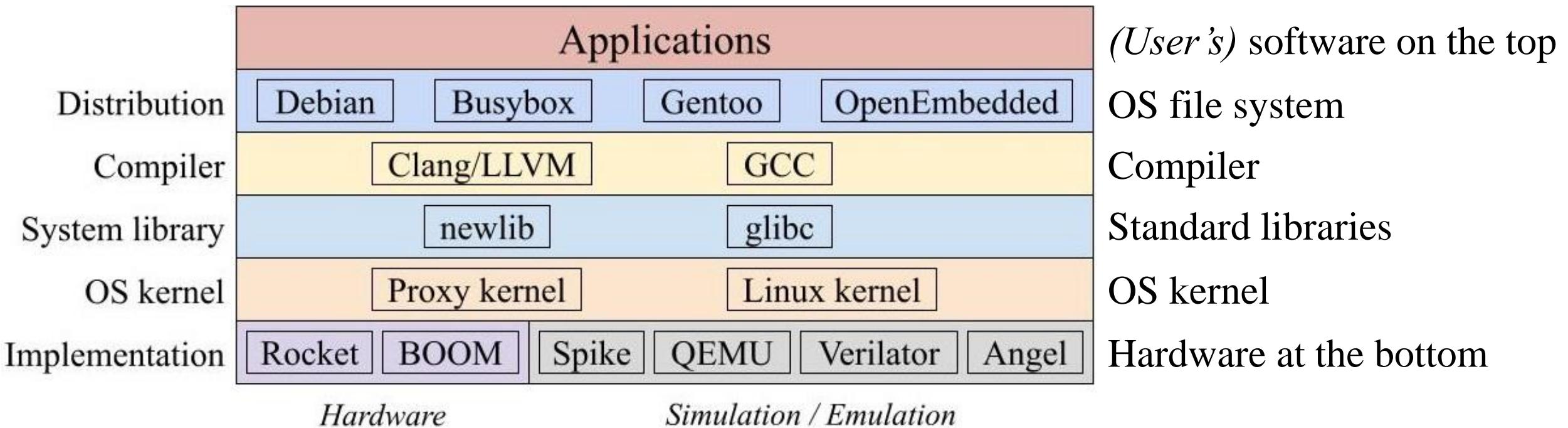


**RISC-V ISA** not only supports the OS stack, but also provides a **privileged architecture**.

→ Better security scheme by having the hardware recognize each code's mode level. (read more at: [Link](#))

### 3. ISA and RISC-V (5/9) RISC-V toolchain

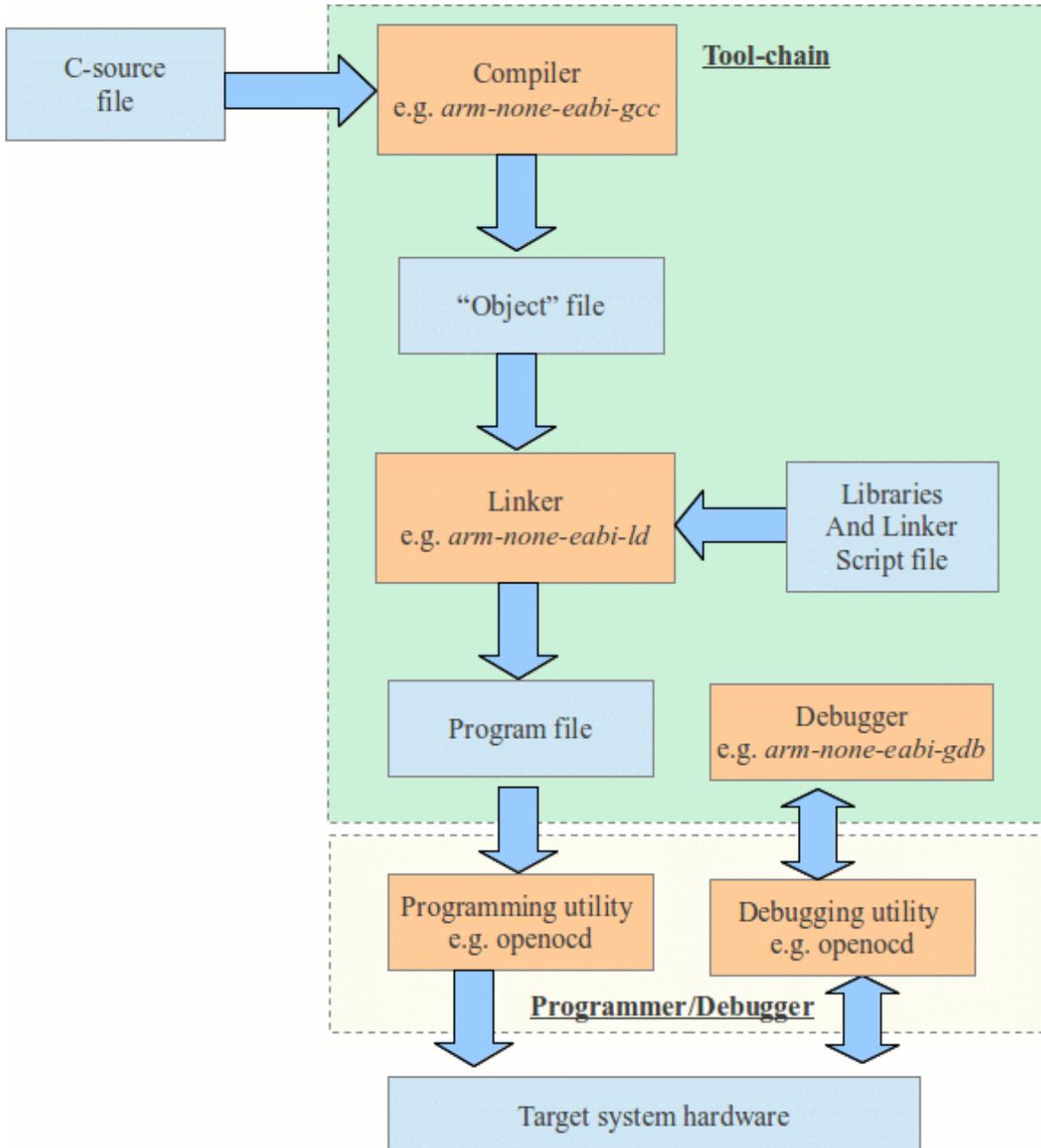
#### RISC-V toolchain and its ecosystem



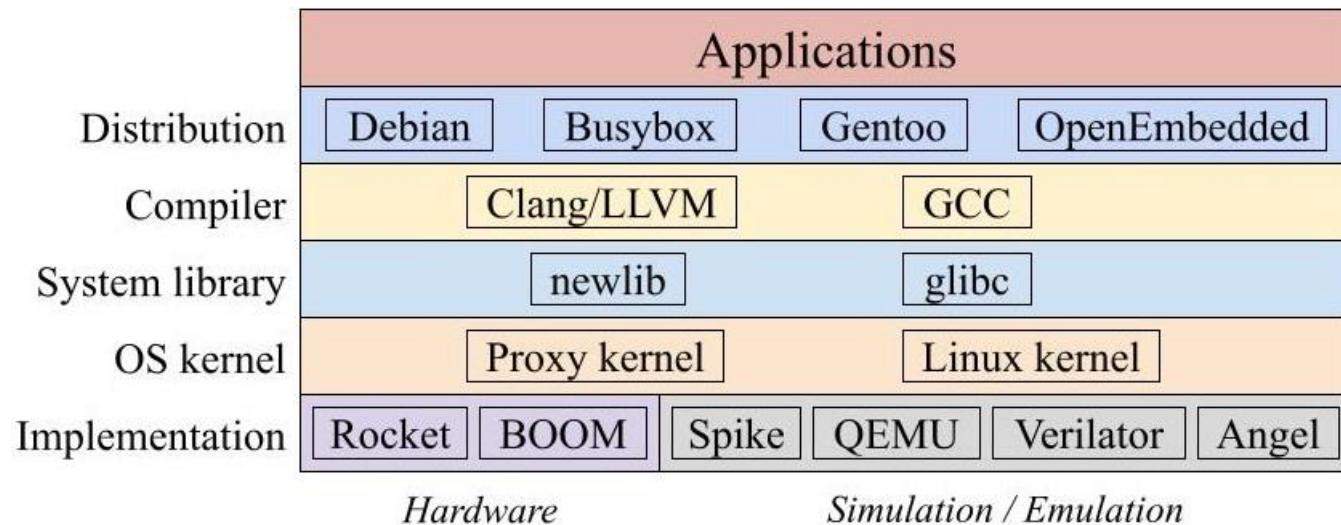
#### Top-down explanation:

User's applications on the top are operated in an OS file system, which then compiled by a compiler based on multiple standard libraries. After compiled, the execution file is run on the OS kernel that manages the hardware at the bottom.

### 3. ISA and RISC-V (6/9) RISC-V toolchain



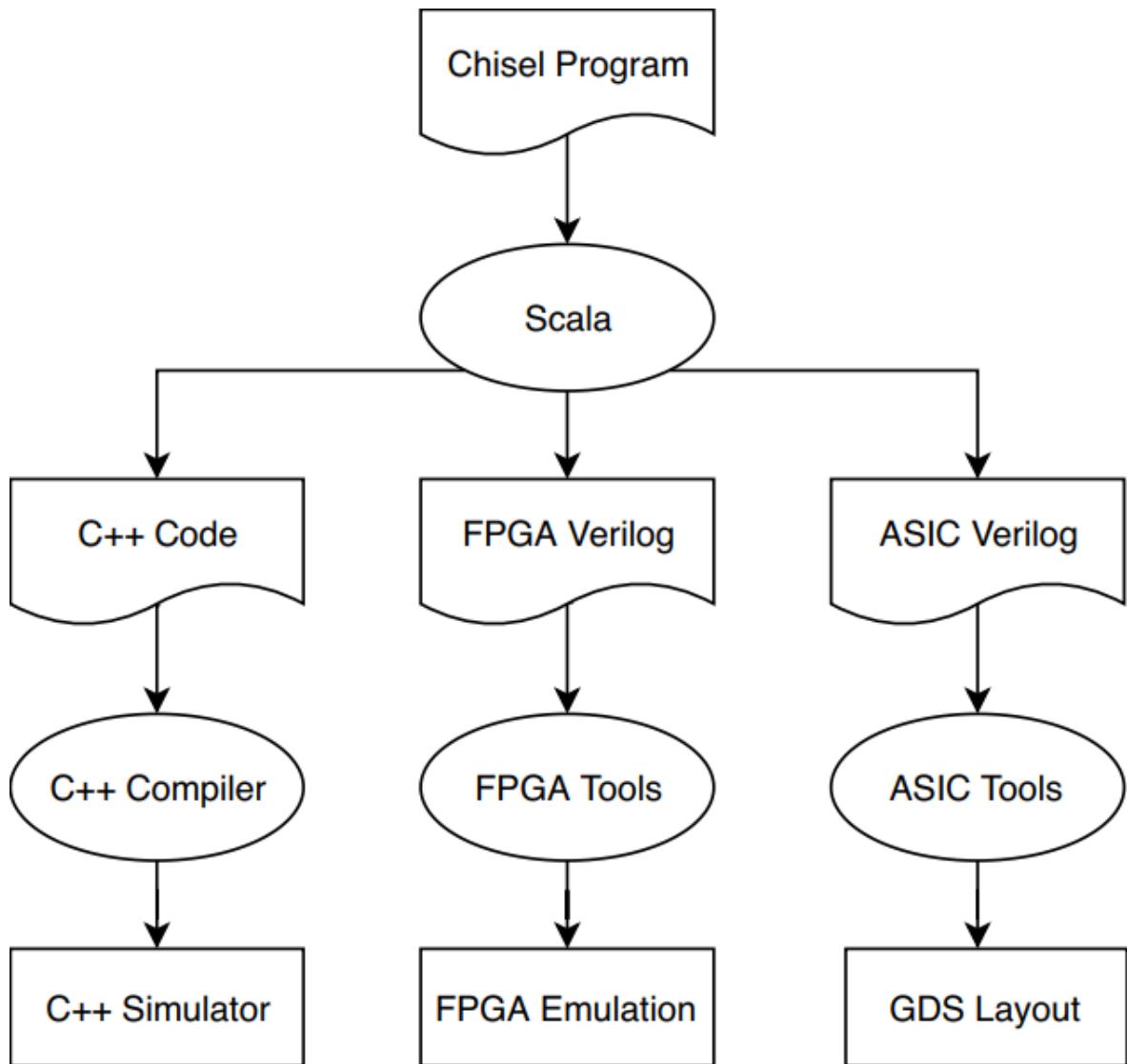
### RISC-V toolchain and its ecosystem



### Three most important tools

- GCC:** (*cross C compiler*) makes a C code into assembly code
- LD:** (*linker*) links standard libraries into the build; also links between multiple C files
- GDB:** (*debugger*) debug the hardware/simulator/emulator

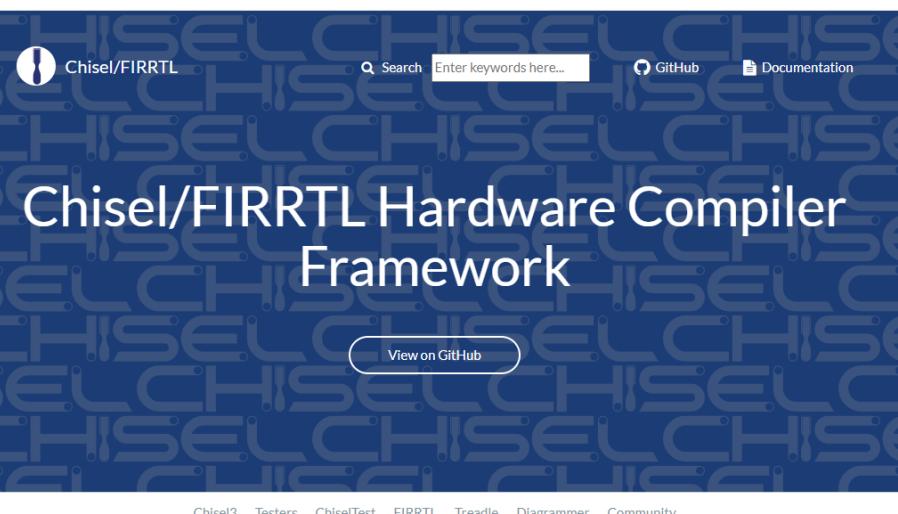
### 3. ISA and RISC-V (7/9) CHISEL



Chisel is a library.  
Scala is a language.

- **Scala** itself is a *high-level object-oriented programming language*  
→ It is not designed for “hardware coding.”
- **Chisel** is a library attached to Scala to define a set of coding rules.  
→ It is designed for “hardware coding.”
- From **Scala** to **Verilog**:  
Scala → Java → FIRRTL → Verilog  
1<sup>st</sup> arrow: Scala compiler named SBT  
2<sup>nd</sup> arrow: executing Java  
3<sup>rd</sup> arrow: FIRRTL compiler

# 3. ISA and RISC-V (8/9) CHISEL



CCC 2022

The official CHISEL website:  
<https://www.chisel-lang.org/>

CHISEL tutorials can be found at:  
<https://github.com/ucb-bar/chisel-tutorial>

Search or jump to... Pull requests Issues Marketplace Explore

ucb-bar / chisel-tutorial Public Watch 89

<> Code Issues 29 Pull requests 3 Actions Projects Wiki Security

release 20 branches 12 tags Go to file Add file Code

4 authors Bump for 3.3.1 (#162) e567a5f on Jun 2, 2020 426 commits

- doc Bump for 3.3.1 (#162) 2 years ago
- project Bump for 3.3.1 (#162) 2 years ago
- src Bump for 3.3.1 (#162) 2 years ago
- .gitignore ignore files created when viewing doc/tutorial/\*.te... 4 years ago
- LICENSE.txt Add LICENSE.txt and reference to it in all .scala files. 6 years ago
- README.md Merge branch 'master' into release 4 years ago
- build.sbt Bump for 3.3.1 (#162) 2 years ago
- run-examples.sh Bump Scala compiler versions, sbt version to 1.1.1 ... 5 years ago
- run-problem.sh Bump Scala compiler versions, sbt version to 1.1.1 ... 5 years ago
- run-solution.sh Bump Scala compiler versions, sbt version to 1.1.1 ... 5 years ago

README.md

Chisel Tutorials (Release branch)

Search or jump to... Pull requests Issues Marketplace Explore

ucb-bar / chisel-release Public Watch 24

<> Code Issues 2 Pull requests Actions Projects Security Insights

master 39 branches 91 tags Go to file Add file Code

jackkoenig Remove chisel-template and chisel-tutorial c2cd7e7 on Jan 12 275 commits

- chisel-tester bump.x branches 2 years ago
- chisel3 bump.x branches 2 years ago
- chiseltest Rename chisel-tester2 -> chiseltest 9 months ago
- diagrammer bump.x branches 2 years ago
- doc Update documentation. 2 years ago
- dsptools bump.x branches 2 years ago
- firrtl bump.x branches 2 years ago
- treadle bump.x branches 2 years ago
- .gitignore Update Makefile with individual project dependen... 3 years ago
- .gitmodules Remove chisel-template and chisel-tutorial 9 months ago
- LICENSE Create LICENSE 4 years ago
- README-OLD.md Fixing up 2 years ago
- README.md Rename chisel-tester2 -> chiseltest 9 months ago
- deps.bare Remove chisel-template and chisel-tutorial 9 months ago
- version.yml Remove chisel-template and chisel-tutorial 9 months ago

README.md

The chisel-release Repository

Released CHISEL sources:  
<https://github.com/ucb-bar/chisel-release>

### 3. ISA and RISC-V (9/9) Summary

#### RISC-V revolutionizes Computer System Design

##### 1. Modular at heart:

customizable ISA and customizable hardware  
→ fine-tune the system to your specific needs.

##### 2. Open-source community:

license-free ISA, open cores and SoCs, open-source libraries, open-source software, etc. → reuse other developers' designs → save time and effort for R&D

##### 3. CHISEL (*Constructing Hardware In Scala Embedded Language*):

a new way to “coding” hardware circuits. When compiled, it will generate a true RTL Verilog code.

→ a “meta-programming” language for hardware developers with parameters and sub-designs that can be overridden or extended.

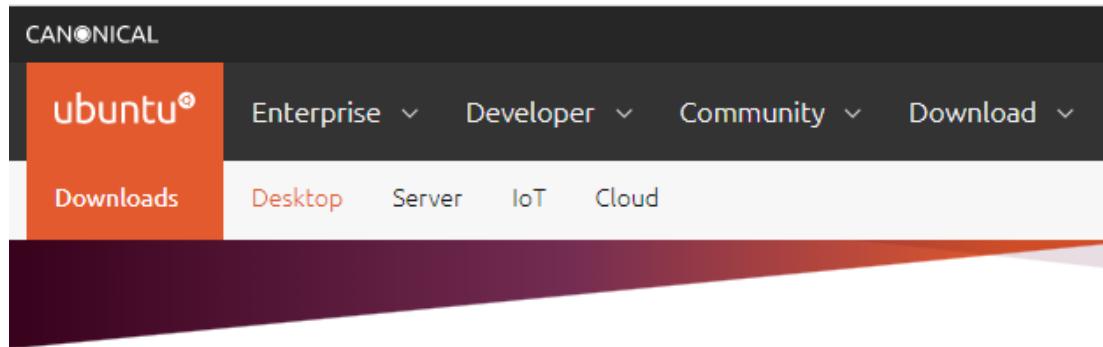
→ easy to develop “object-oriented” hardware library for reuse purpose.

# Outline

1. Introduction
2. What makes a processor and computer system?
3. Instruction Set Architecture (ISA) and RISC-V
4. **Necessary tools for working with RISC-V**
5. RISC-V open-source community and materials
6. Some RISC-V news

# 4. RISC-V working tools (1/8) Ubuntu desktop

First of all, you'll need an Ubuntu machine (recommend version 20.04-LTS)



## Download Ubuntu Desktop

The open-source desktop operating system that powers millions of PCs and laptops around the world. Find out more about Ubuntu's features and how we support developers and organisations below.

[Ubuntu Desktop homepage](#)

[Visit the Ubuntu Desktop blog ›](#)

You can follow my guide to set up a new Ubuntu machine for working with RISC-V:  
<https://thuchoang90.github.io/tutorial/2022/09/30/Fresh-Ubuntu-setup>

Thuctorial      HOME    TUTORIAL    PROJECT    TAGS    ABOUT    ☰

Light   Dark

### Content

- I. Dependencies & Proxy
- II. RISC-V Tools
  - II. a) Github
  - II. b) Scala & sbt:
  - II. c) Verilator
  - II. d) QEMU
  - II. e) Idea IntelliJ
  - II. f) Eclipse
  - II. g) OpenOCD
  - II. h) Vivado
  - II. i) Quartus
- III. RISC-V Toolchain
  - III. a) Git clone
  - III. b) Config & Make

## Fresh Ubuntu machine setup for working with RISC-V

Sep 30, 2022 | About 13 mins

#RISC-V

### I. Dependencies & Proxy

To make `vi` more comfortable:

SHELL

# 4. RISC-V working tools (2/8) RISC-V toolchain

The next step, preparing the RISC-V GNU toolchain, is the essential part.

Official GitHub link for compiling from source:

<https://github.com/riscv-collab/riscv-gnu-toolchain>

A screenshot of a GitHub repository page. At the top, there's a dark header with a GitHub icon, navigation links for 'Product', 'Solutions', 'Open Source', and 'Pricing', and a search bar. Below the header, the repository name 'riscv-collab / riscv-gnu-toolchain' is shown as 'Public'. There are buttons for 'Notifications' and a bell icon. Below this, there are links for 'Code', 'Issues (45)', 'Pull requests (5)', 'Actions', 'Projects', 'Wiki', and 'Security'. A dropdown shows 'master' branch, '5 branches', and '43 tags'. On the right, there are buttons for 'Go to file' and 'Code'. The main content area shows a file named 'README.md'.

## RISC-V GNU Compiler Toolchain

This is the RISC-V C and C++ cross-compiler. It supports two build modes: a generic ELF/Newlib toolchain and a more sophisticated Linux-ELF/glibc toolchain.

You can also download a pre-built toolchain on the SiFive webpage:

[\(however, I recommend compiling from source\)](https://www.sifive.com/software)

Prebuilt RISC-V  
GCC Toolchain and Emulator

Save time by using one of our prebuilt toolchains which contain all the tools necessary to compile and debug programs on SiFive products. No hardware, no problem as the QEMU emulator packages can be used to test software applications without hardware. Our toolchain and emulator distributions have been carefully packaged to support both 32-bit & 64-bit ISAs. All the available packages are built using our [Freedom Tools GitHub repository](#), where previous released versions are available.

GNU Embedded Toolchain –  
v2020.12.8

OpenOCD – v2020.12.1

Windows

Windows

macOS

macOS

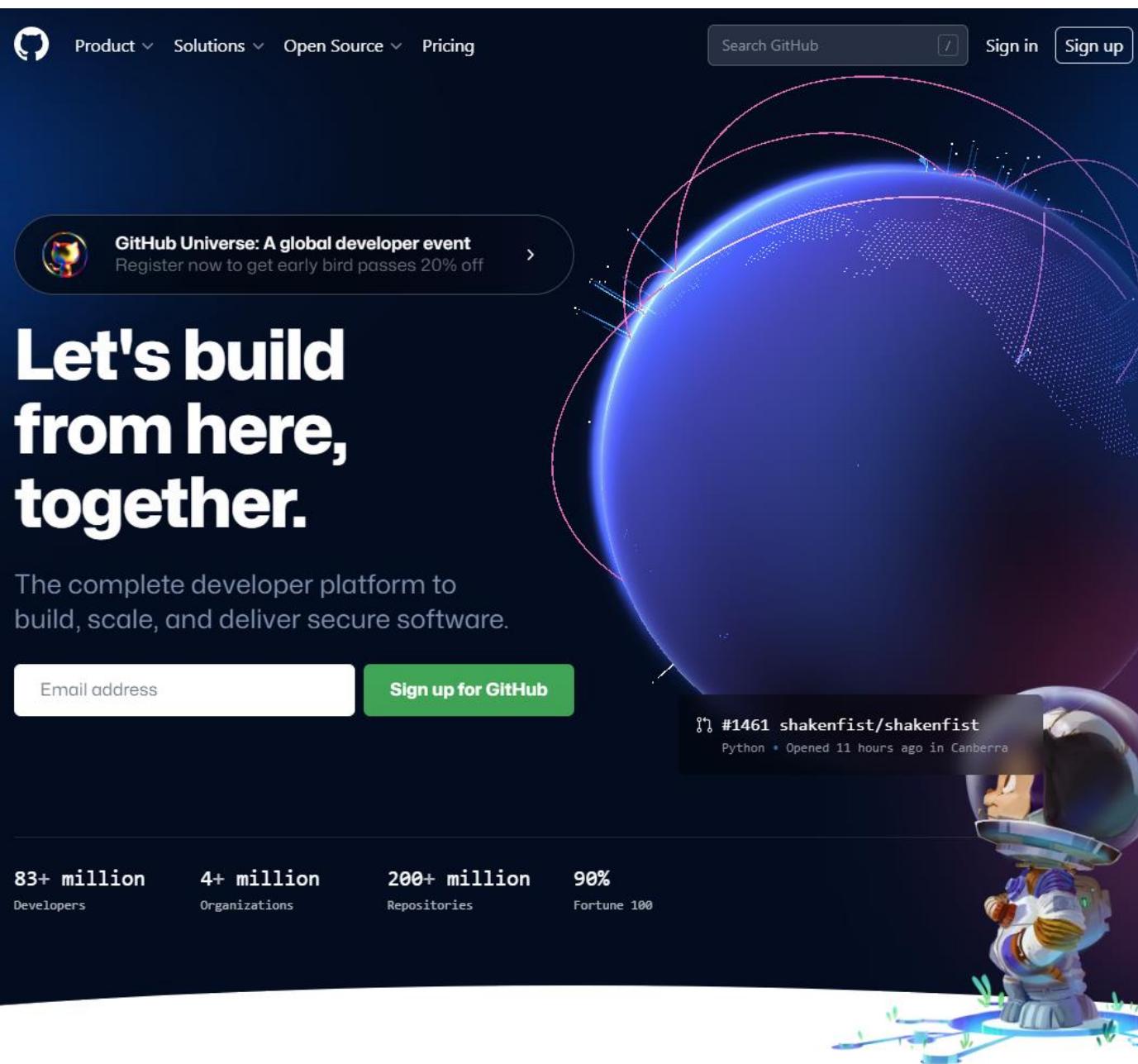
CentOS

CentOS

Ubuntu

Ubuntu

# 4. RISC-V working tools (3/8) GitHub



When working with the RISC-V open-source community, you'll be using GitHub a lot.

So it's better to set up a GitHub account for personal use.

<https://github.com/>

# 4. RISC-V working tools (4/8) Hardware simulation/emulation

## Verilator

- A simulator
- Simulate a Verilog code at each clock cycle
- A tool for hardware developers
- Web: <https://www.veripool.org/verilator/>

The screenshot shows the Verilator homepage. At the top, there's a navigation bar with links for Home, Verilator (which is highlighted), Verilog-Mode, Other Tools, Papers, and About. Below the navigation is a large "Welcome to Verilator" header. To the right of the header is a large blue "V" logo with the word "VERILATOR" below it. Underneath the logo is a bar chart titled "Fast" which compares the performance of Verilator against other simulators. The chart has two sets of bars: one for "Big 3" simulators (Vendor A and Vendor B) and one for Verilator (1, 2, 4, 6, 12 threads). The Verilator bars are significantly taller than the others. To the right of the chart is a list of features: "Accepts Verilog or SystemVerilog", "Performs lint code-quality checks", "Compiles into multithreaded C++ or SystemC", and "Creates XML to front-end your own tools".

## QEMU

- An emulator
- Emulate an ideal RISC-V processor for software to run (*exactly like running a virtual machine; in this case, RISC-V on Linux*)
- A tool for software developers
- Web: <https://www.qemu.org/>

The screenshot shows the QEMU homepage. At the top, there's a dark navigation bar with links for DOWNLOAD, SUPPORT, CONTRIBUTE, DOCS, WIKI, and BLOG. Below the navigation is a large red section containing the word "QEMU" in white. Underneath "QEMU" is the text "A generic and open source machine emulator and virtualizer".

## 4. RISC-V working tools (5/8) Working with Chisel

Remind the Chisel-to-Verilog compile flow:  
Scala (+Chisel) → Java → FIRRTL → Verilog

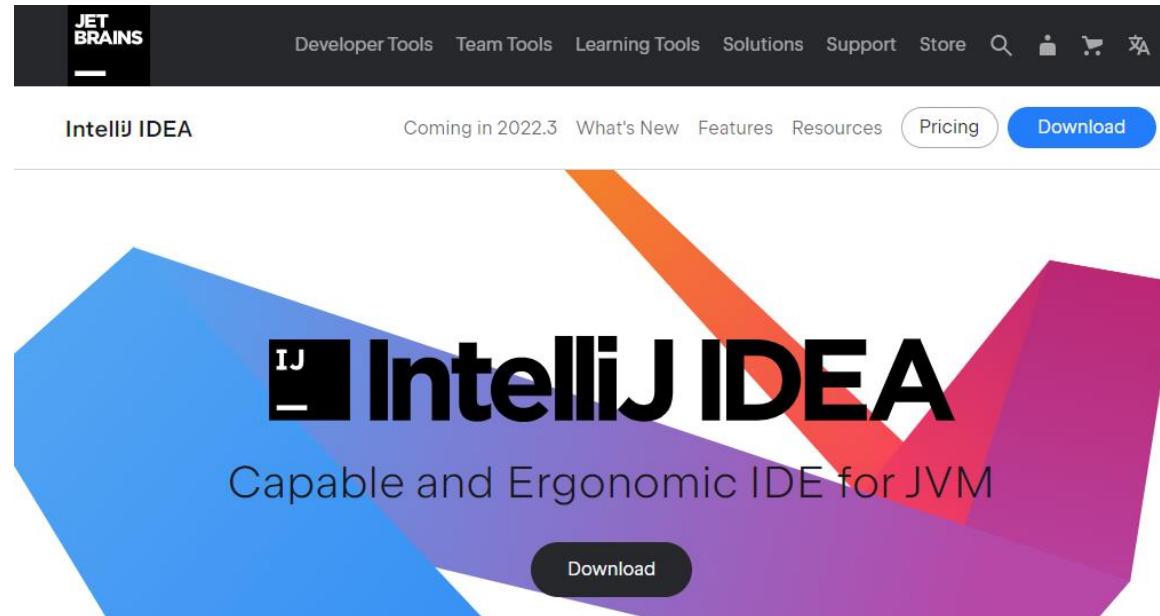
- Scala: a programming language
- Chisel: a library (*attached to Scala*)
- Sbt: a Scala compiler

So, we need to install the **Sbt** tool.

Webpage: <https://www.scala-sbt.org/>

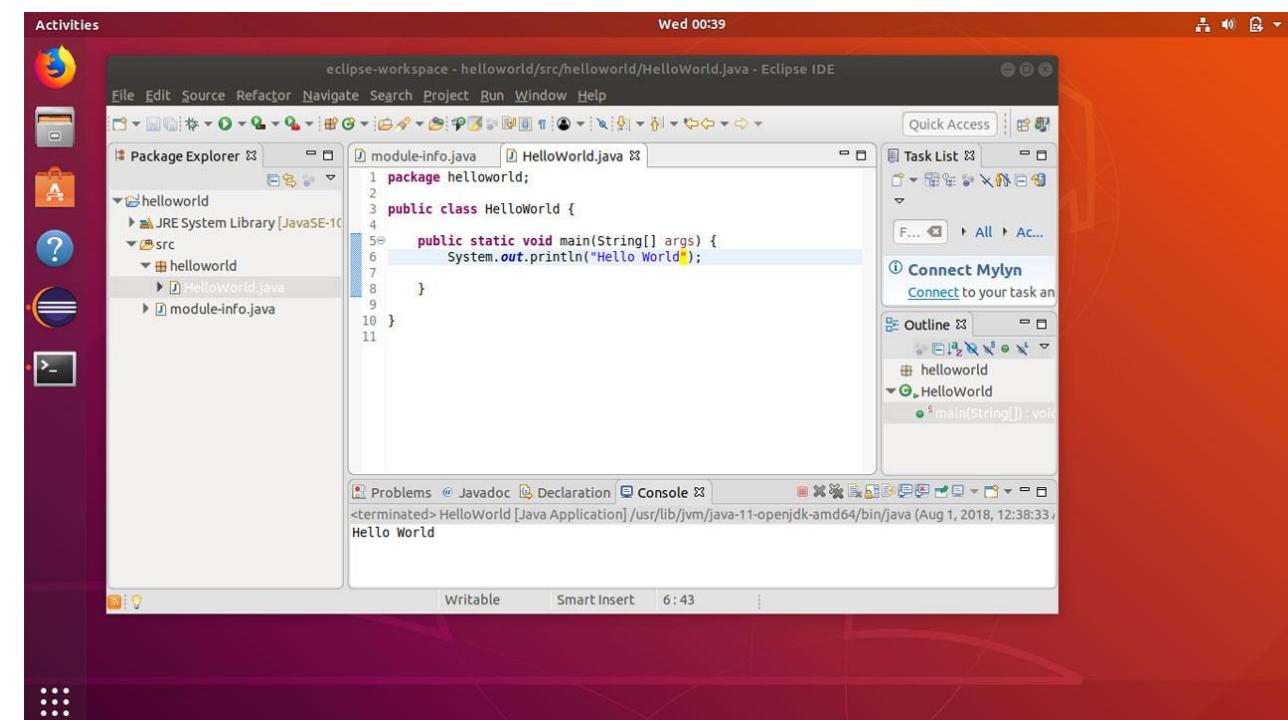


The **Sbt** tool itself is a terminal command line.  
Recommend installing **IntelliJ IDEA**, a GUI for  
Sbt (*like a Visual Studio for C/C++*).  
Webpage: <https://www.jetbrains.com/idea/>



# 4. RISC-V working tools (6/8) Software (C/C++) coding

For software coding, especially C/C++, recommend installing **Eclipse**.  
Webpage: <https://www.eclipse.org/>



# 4. RISC-V working tools (7/8) Debug tool

OpenOCD is the tool used for debugging  
*(together with RISC-V GDB in the toolchain):*

Webpage: <https://openocd.org/>

GitHub: <https://github.com/riscv/riscv-openocd>

## Open On-Chip Debugger

About Bug Tracker Discussion Documentation Donations

Getting OpenOCD IRC Mailing lists Repository Supported JTAG interfaces

### OpenOCD 0.12.0-rc1 release candidate is out

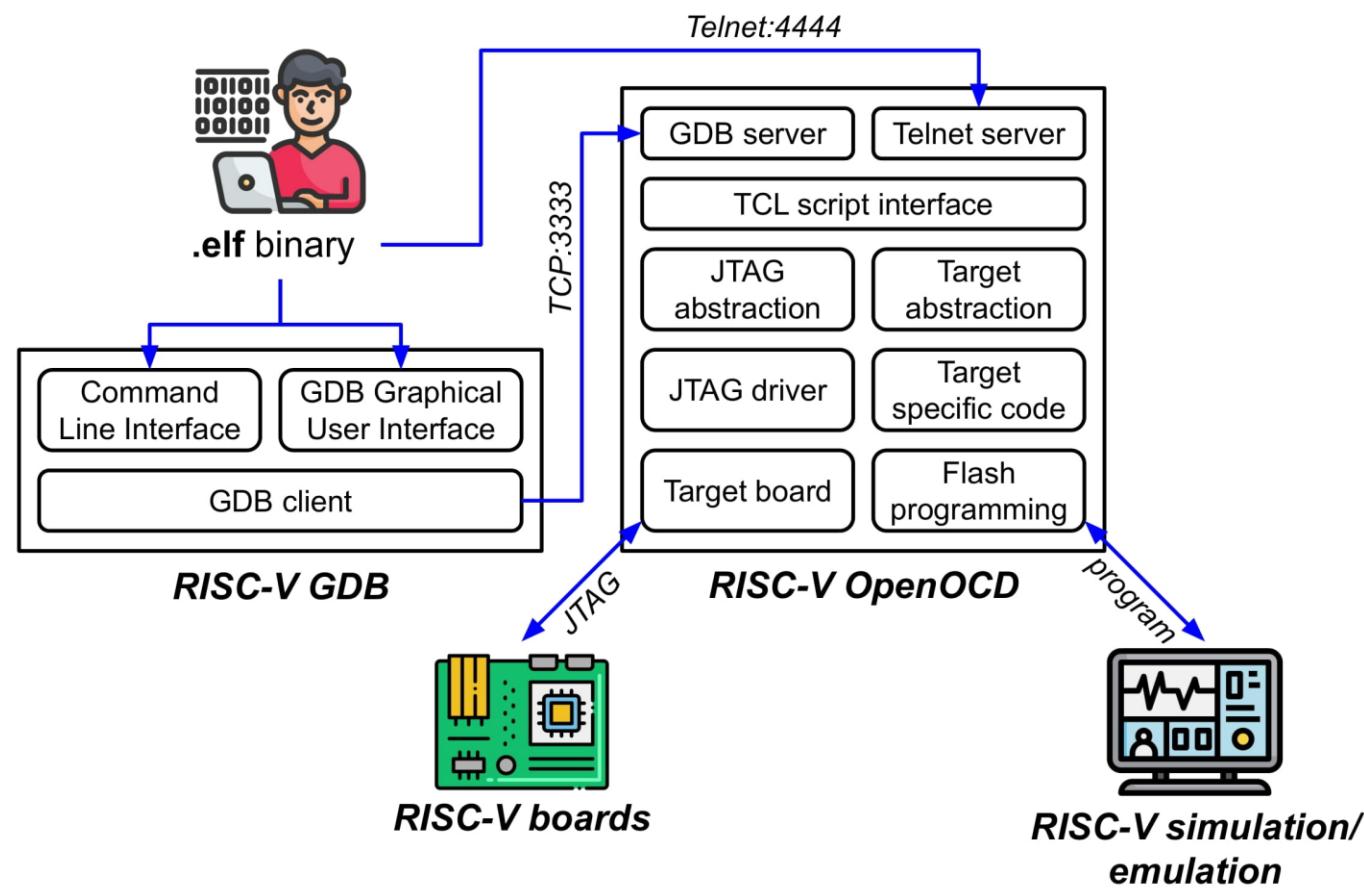
We are pleased to announce the first release candidate of the upcoming OpenOCD version.

Sun 18 September 2022  
By [fercerpav](#)

The source archives and release notes are available from [the usual SF download locations](#).

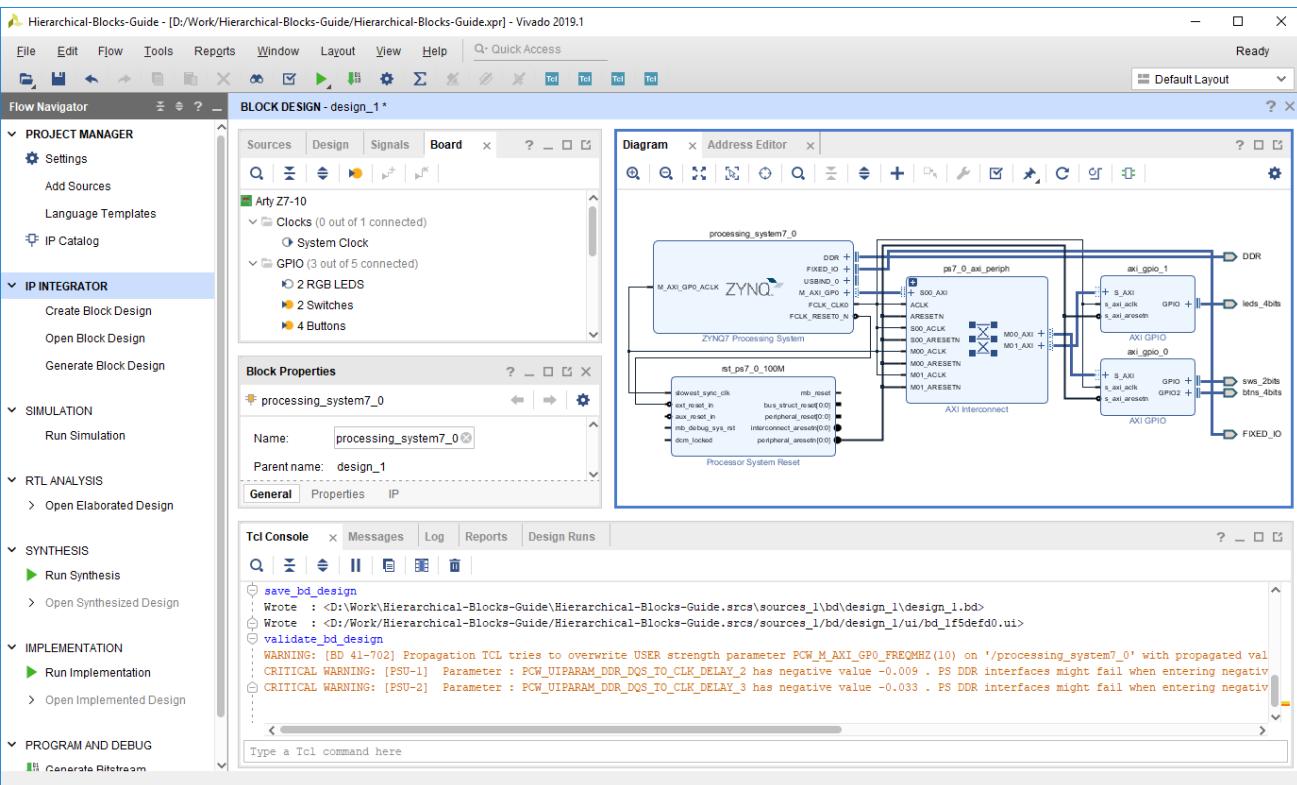
Please post all your feedback to the [openocd-devel mailing list](#).

A typical debug flow with GDB and OpenOCD:



# 4. RISC-V working tools (8/8) Working with FPGA

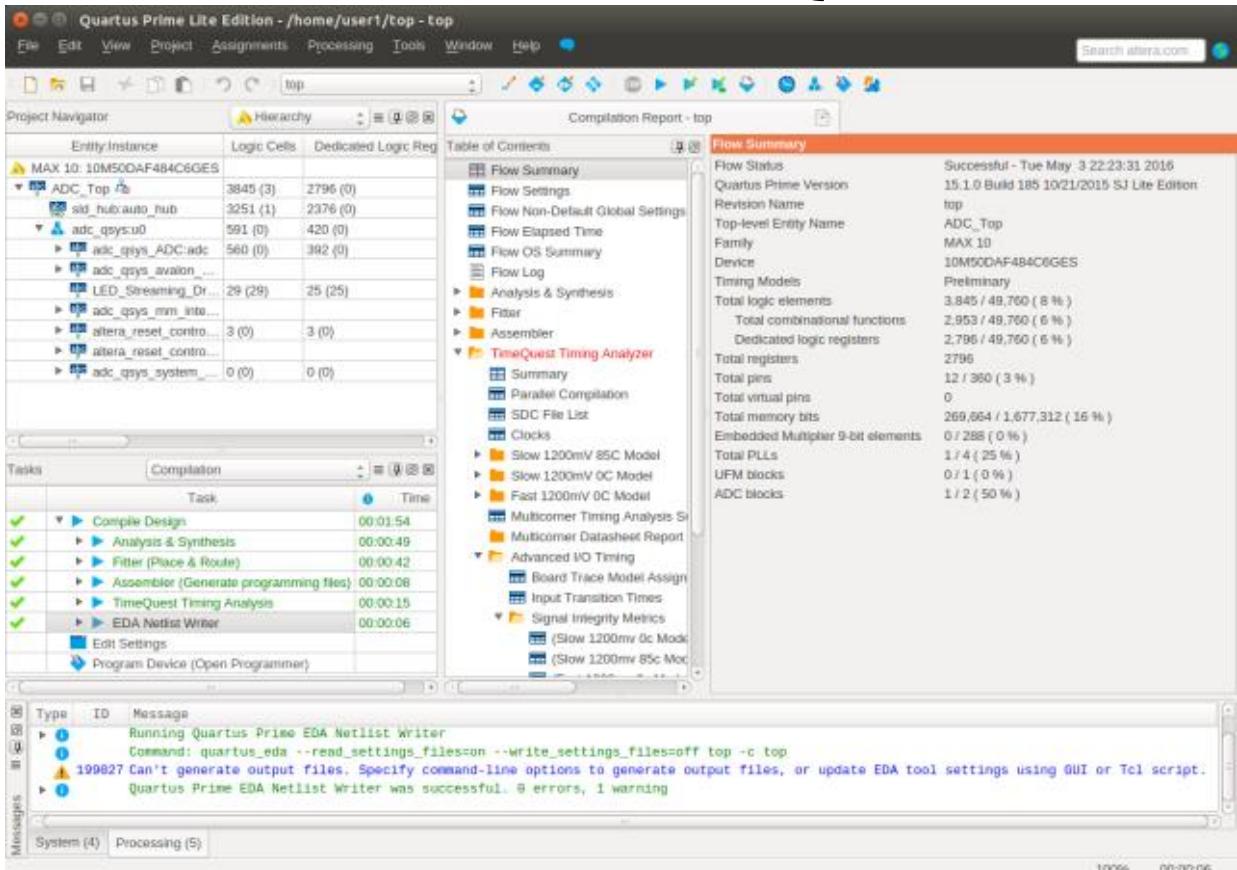
Finally, if you are working with FPGAs, you must install Vivado for Xilinx or Quartus for Altera boards.



Vivado ML standard

Web:

<https://www.xilinx.com/support/download.html>



Quartus Prime Standard

Web:

<https://www.intel.com/content/www/us/en/collections/products/fpga/software/downloads.html>

# Outline

1. Introduction
2. What makes a processor and computer system?
3. Instruction Set Architecture (ISA) and RISC-V
4. Necessary tools for working with RISC-V
5. **RISC-V open-source community and materials**
6. Some RISC-V news

# 5. Open community and materials (1/9) Libraries

The common open RISC-V libraries that you can use

**Chipyard** (*contains many common and frequently used open IPs, including RISC-V processors and other peripherals such as uart, spi, sd-card, etc.):*

<https://github.com/ucb-bar/chipyard>

README.md

## CHIPYARD

Chipyard Framework chipyard-ci-process passing

Quick Links

- Stable Documentation: <https://chipyard.readthedocs.io/>
- User Question Forum: <https://groups.google.com/forum/#!forum/chipyard>
- Bugs and Feature Requests: <https://github.com/ucb-bar/chipyard/issues>

Using Chipyard

To get started using Chipyard, see the stable documentation on the Chipyard documentation site: <https://chipyard.readthedocs.io/>

What is Chipyard

sifive / fpga-shells Public Watch 45

Code Issues 6 Pull requests 16 Actions Projects Security

master 106 branches 0 tags Go to file Add file Code

erikdanie Merge pull request #158 from a... f9fb9fd on Dec 29, 2020 473 commits

File	Description	Time Ago
.github/workflows	Cl: add a scala compilation check	2 years ago
microsemi	newshells checkpoint	3 years ago
src/main/scala	Add utility function for IBUF_LOW_POWER	2 years ago
vsrc/nfmac10g	nfmac10g: fix a power-0 bug	4 years ago
xilinx	refactor tcl code that wasn't executing correctly	2 years ago
.gitignore	Initial commit for fpga-shells	5 years ago
README.md	improved clarity of documentation	3 years ago
build.wake	wake: use variable for package location	2 years ago
wit-manifest.json	bump sifive-blocks (#157)	2 years ago

README.md

fpga-shells

An FPGA shell is a Chisel module designed to wrap any SiFive core configuration. The goal of the fpga-shell system is to reduce the number of wrappers to have only one for each physical device rather than one for every combination of physical device and core configuration.

**fpga-shells** (*contains many common FPGA configurations*):  
<https://github.com/sifive/fpga-shells>

# 5. Open community and materials (2/9) Processors

## Some famous RISC-V processors

**Rocket** is the most popular among RISC-V processors:

<https://github.com/chipsalliance/rocket-chip>

(it is an in-of-order processor)

The screenshot shows the README.md page of the Rocket Chip Generator repository. It features a header with a logo, a continuous integration status badge (passing), and a table of contents. The main text describes the repository's purpose and links to a technical report. A detailed list of instructions and guides follows.

README.md

## Rocket Chip Generator 🚀

Continuous Integration passing

This repository contains the Rocket chip generator necessary to instantiate the RISC-V Rocket Core. For more information on Rocket Chip, please consult our [technical report](#).

## Table of Contents

- Quick instructions for those who want to dive directly into the details without knowing exactly what's in the repository.
- What's in the Rocket chip generator repository?
- How should I use the Rocket chip generator?
  - Using the cycle-accurate Verilator simulation
  - Mapping a Rocket core down to an FPGA
  - Pushing a Rocket core through the VLSI tools
- How can I parameterize my Rocket chip?
- Debugging with GDB
- Building Rocket Chip with an IDE
- Contributors

**BOOM** is an out-of-order processor that can rival ARM:

<https://github.com/riscv-boom/riscv-boom>



☞ **The Berkeley Out-of-Order RISC-V Processor** FAILED

# 5. Open community and materials (3/9) Processors

## Some famous RISC-V processors

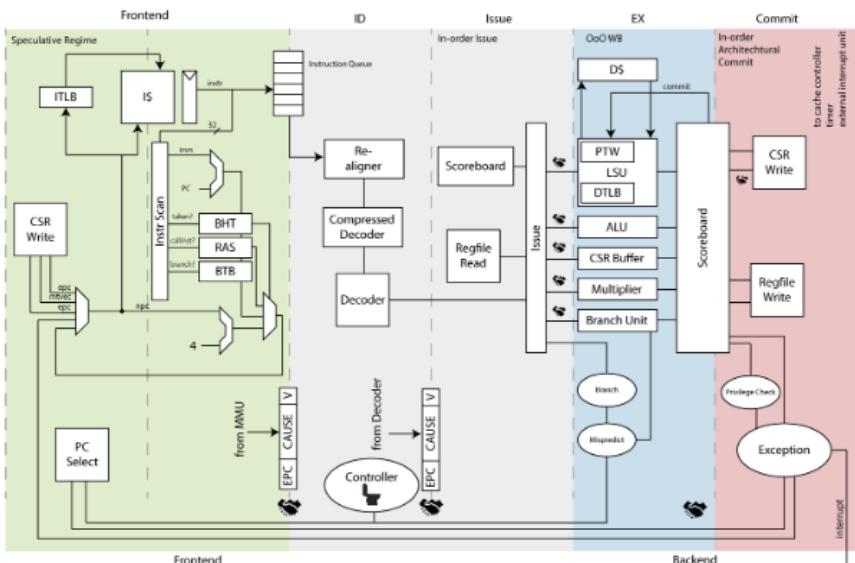
Ariane is a high-performance 64-bit in-of-order processor:

<https://github.com/lowRISC/ariane>

### ❖ Ariane RISC-V CPU

Ariane is a 6-stage, single issue, in-order CPU which implements the 64-bit RISC-V instruction set. It fully implements I, M, A and C extensions as specified in Volume I: User-Level ISA V 2.3 as well as the draft privilege extension 1.10. It implements three privilege levels M, S, U to fully support a Unix-like operating system. Furthermore it is compliant to the draft external debug spec 0.13.

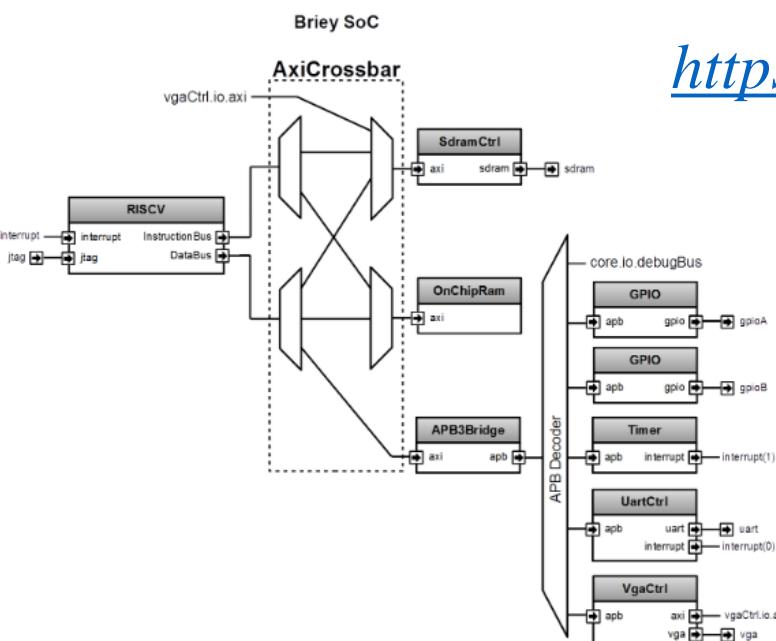
It has configurable size, separate TLBs, a hardware PTW and branch-prediction (branch target buffer and branch history table). The primary design goal was on reducing critical path length.



VexRiscv is a simple in-of-order 32-bit processor that is suited for an MCU:

### ❖ Briey SoC

As a demonstration, a SoC named Briey is implemented in `src/main/scala/vexriscv/demo/Briey.scala`. This SoC is very similar to the Pinsec SoC:



<https://github.com/SpinalHDL/VexRiscv>

# 5. Open community and materials (4/9) Processors

## Some famous RISC-V processors

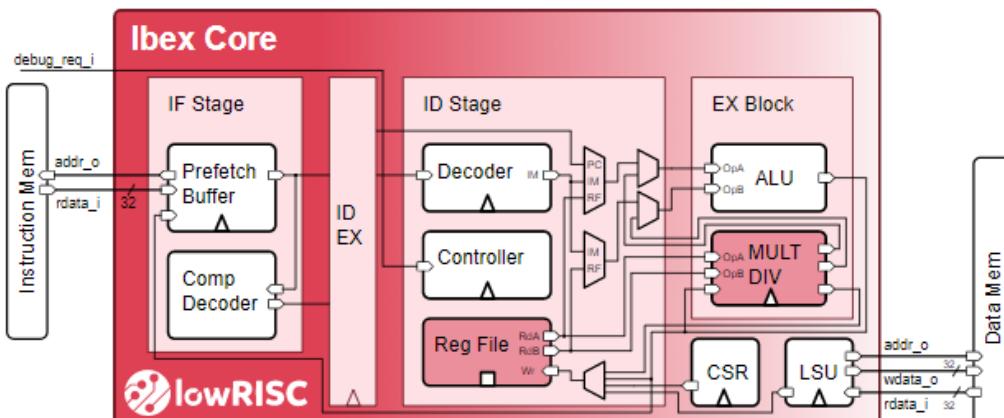
Ibex is an in-of-order 32-bit processor that focuses on security applications:  
<https://github.com/lowRISC/ibex>



### Ibex RISC-V Core

Ibex is a production-quality open source 32-bit RISC-V CPU core written in SystemVerilog. The CPU core is heavily parametrizable and well suited for embedded control applications. Ibex is being extensively verified and has seen multiple tape-outs. Ibex supports the Integer (I) or Embedded (E), Integer Multiplication and Division (M), Compressed (C), and B (Bit Manipulation) extensions.

The block diagram below shows the *small* parametrization with a 2-stage pipeline.

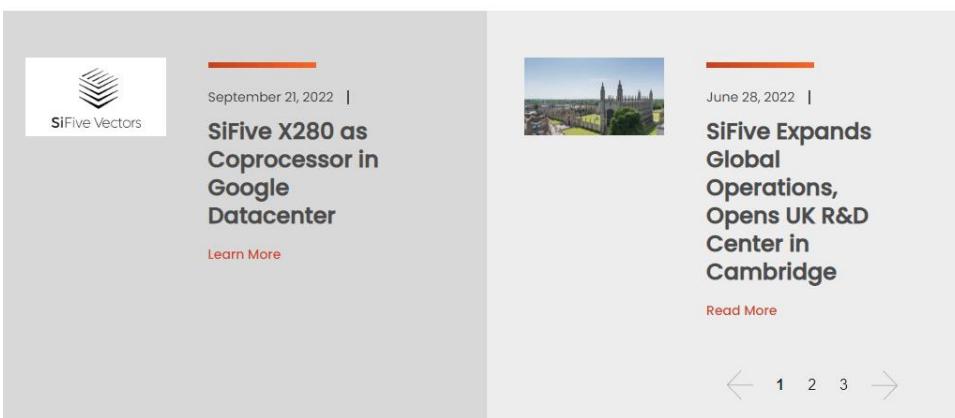
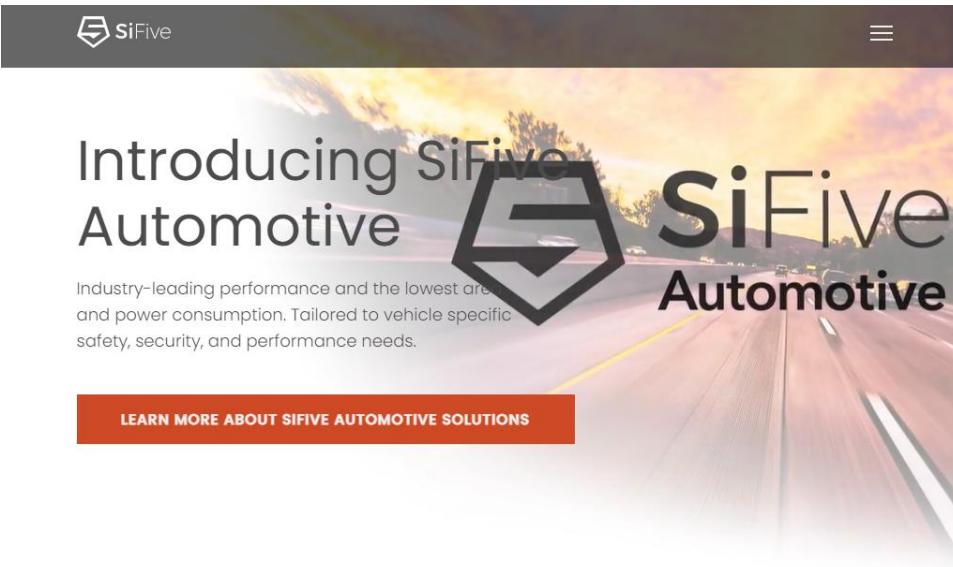


Ibex was initially developed as part of the [PULP platform](#) under the name "Zero-riscy", and has been contributed to [lowRISC](#) who maintains it and develops it further. It is under active development.

# 5. Open community and materials (5/9) Organizations

## Some famous RISC-V groups

**SiFive** is the first and the most famous company that is doing RISC-V-related products:



- Their website: <https://www.sifive.com/>
- Their github: <https://github.com/sifive>

**HexFive** is a company that focuses on RISC-V-based security applications:

- Their website: <https://hex-five.com/>
- Their github: <https://github.com/hex-five>

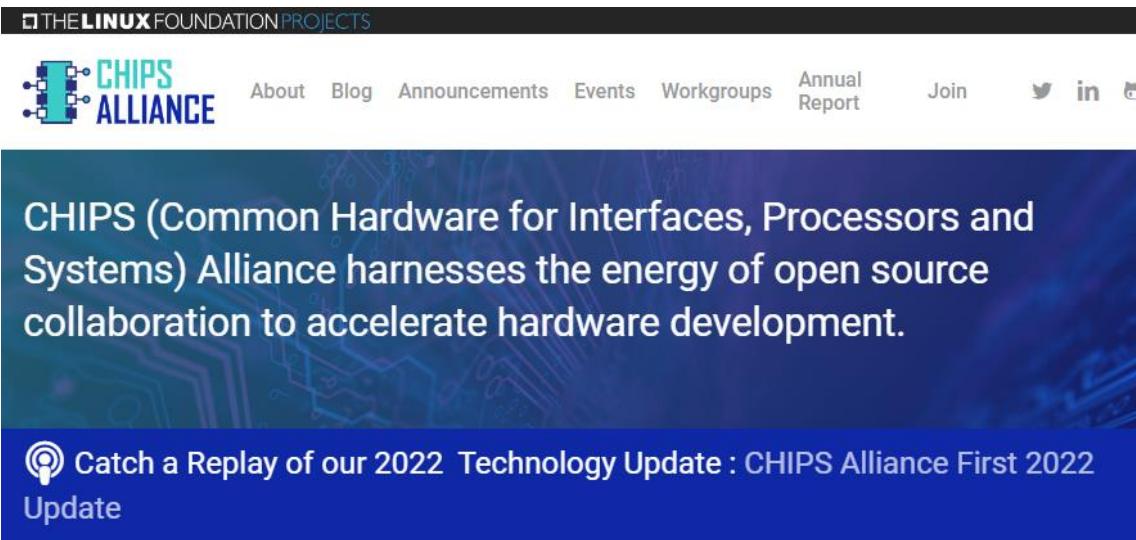
0x5 HEX-Five Security Products Partners Company Downloads Contact FAQ



# 5. Open community and materials (6/9) Organizations

## Some famous RISC-V groups

**Chips Alliance** is an organization that maintains many high-quality open-source IPs used in RISC-V systems:



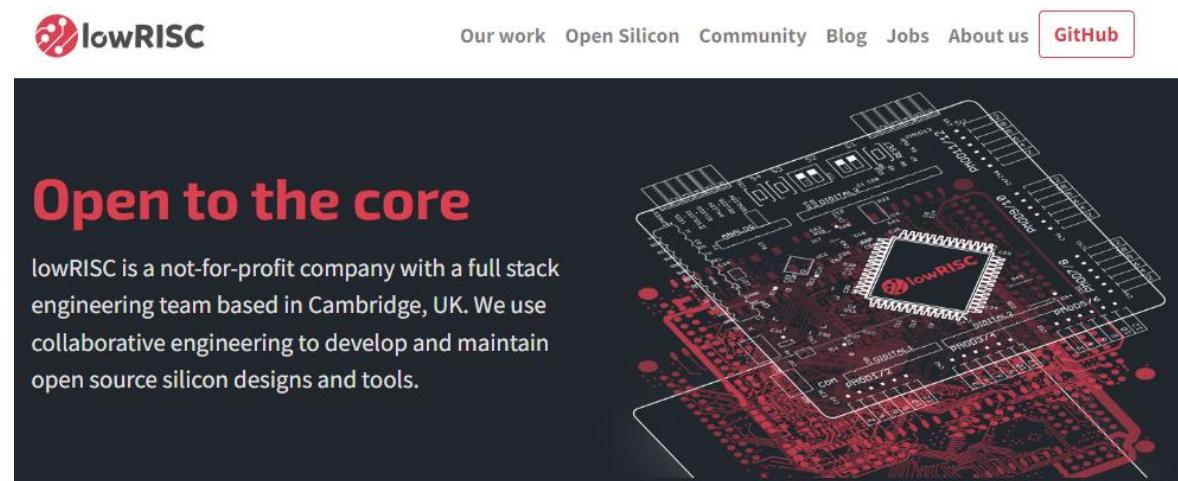
The screenshot shows the CHIPS Alliance website under THE LINUX FOUNDATION PROJECTS banner. It features the CHIPS ALLIANCE logo, navigation links for About, Blog, Announcements, Events, Workgroups, Annual Report, Join, and social media icons. A main banner states: "CHIPS (Common Hardware for Interfaces, Processors and Systems) Alliance harnesses the energy of open source collaboration to accelerate hardware development." Below this is a call-to-action button: "Catch a Replay of our 2022 Technology Update : CHIPS Alliance First 2022 Update".

The CHIPS Alliance develops high-quality, open source hardware designs relevant to silicon devices and FPGAs. By creating an open and collaborative environment, CHIPS Alliance shares resources to lower the cost of development. Companies and individuals can work together to develop open source CPUs, various peripherals, and complex IP blocks. CHIPS Alliance is open to all organizations who are interested in collaborating on open source hardware or software tools to accelerate the creation of more efficient and innovative chip designs.

- Their website: <https://chipsalliance.org/>
- Their github: <https://github.com/chipsalliance>

**lowRISC** is a non-profit company that has published many low-power and high-security IPs for RISC-V systems:

- Their website: <https://lowrisc.org/>
- Their github: <https://github.com/lowRISC>

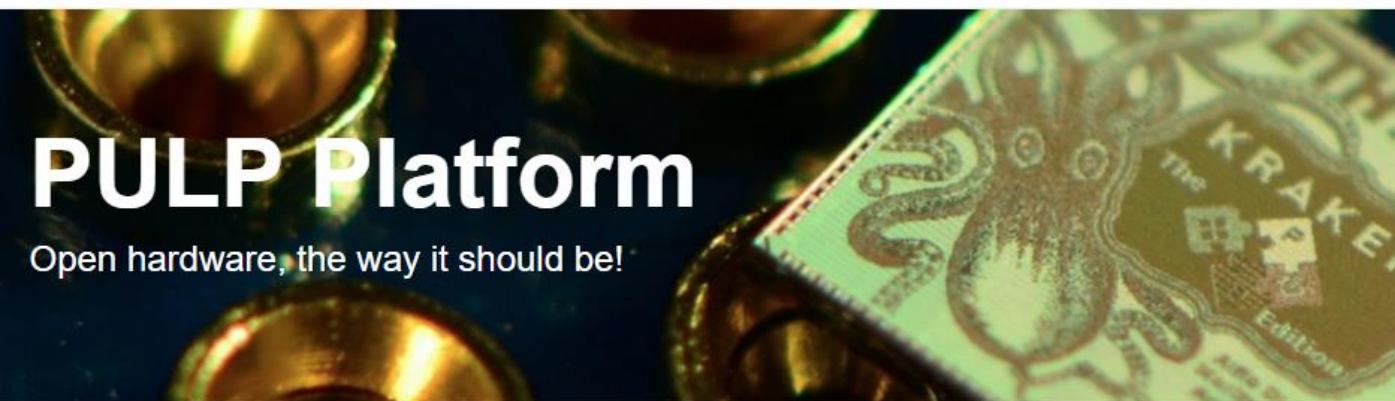


The screenshot shows the lowRISC website. It features the lowRISC logo and navigation links for Our work, Open Silicon, Community, Blog, Jobs, About us, and GitHub. A prominent red banner reads "Open to the core". Below it, a text block says: "lowRISC is a not-for-profit company with a full stack engineering team based in Cambridge, UK. We use collaborative engineering to develop and maintain open source silicon designs and tools." To the right is an image of a printed circuit board (PCB) with a central lowRISC chip.

# 5. Open community and materials (7/9) Organizations

## Some famous RISC-V groups

PULP Platform Resources ▾ About ▾ FAQ Privacy Policy Contact



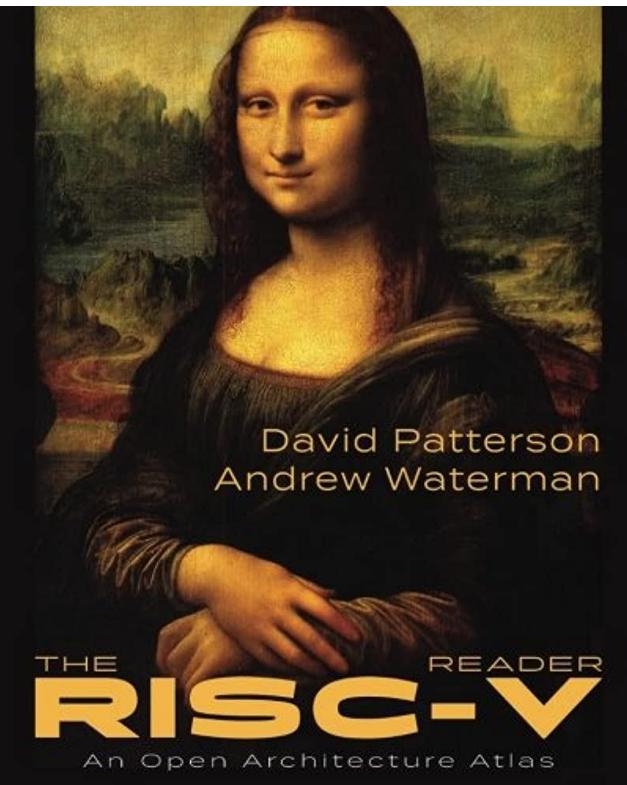
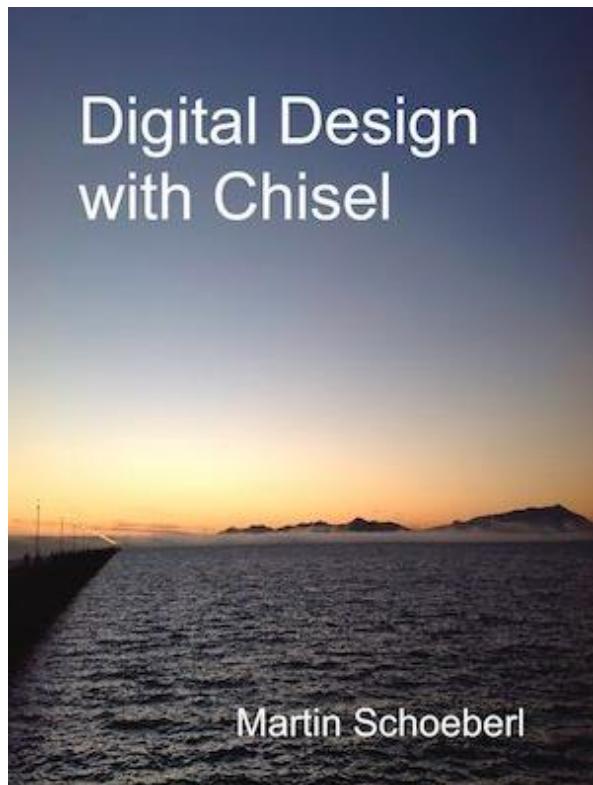
**PULP** is an open group started from a RISC-V project published by two universities.

It has many open-source processors and fabricated many chips:

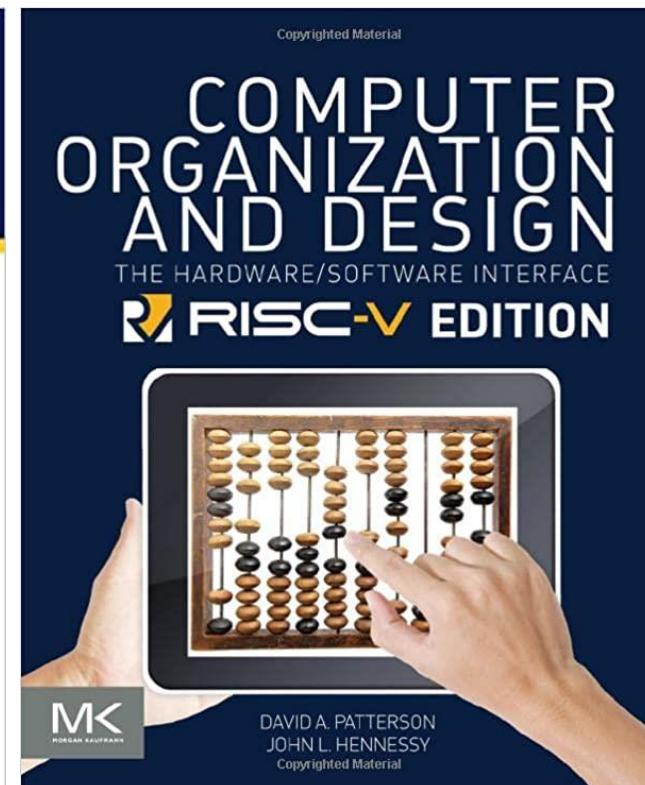
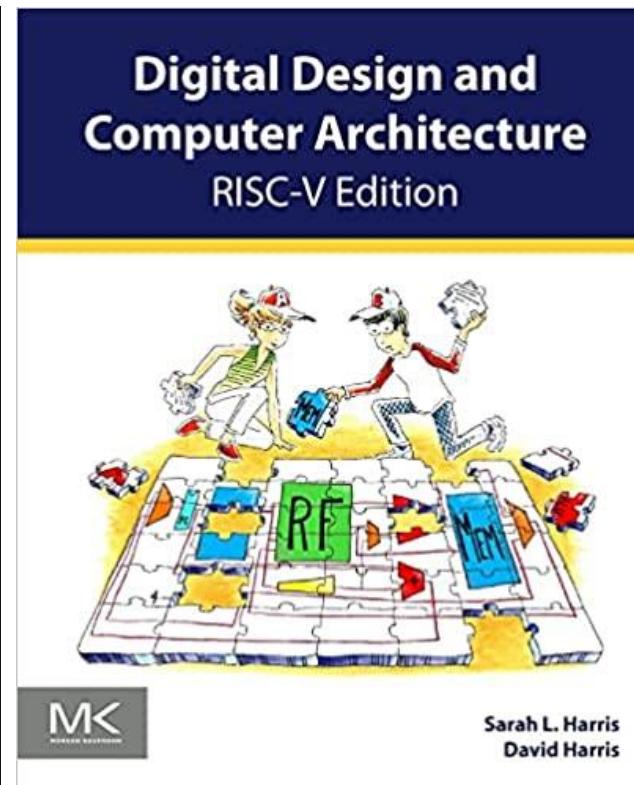
- Their website: <https://pulp-platform.org/>
- Their github: <https://github.com/pulp-platform>

## 5. Open community and materials (8/9) Books

Two “must-have” books for RISC-V developers, from beginners to experts



RISC-V books that often used in universities for teaching



I have a book/material collection on google drive:

<https://drive.google.com/drive/folders/1bAfqALmKGJLOcOlqiE51WB-SudU6-EY?usp=sharing>

# 5. Open community and materials (9/9) Beginner recommend

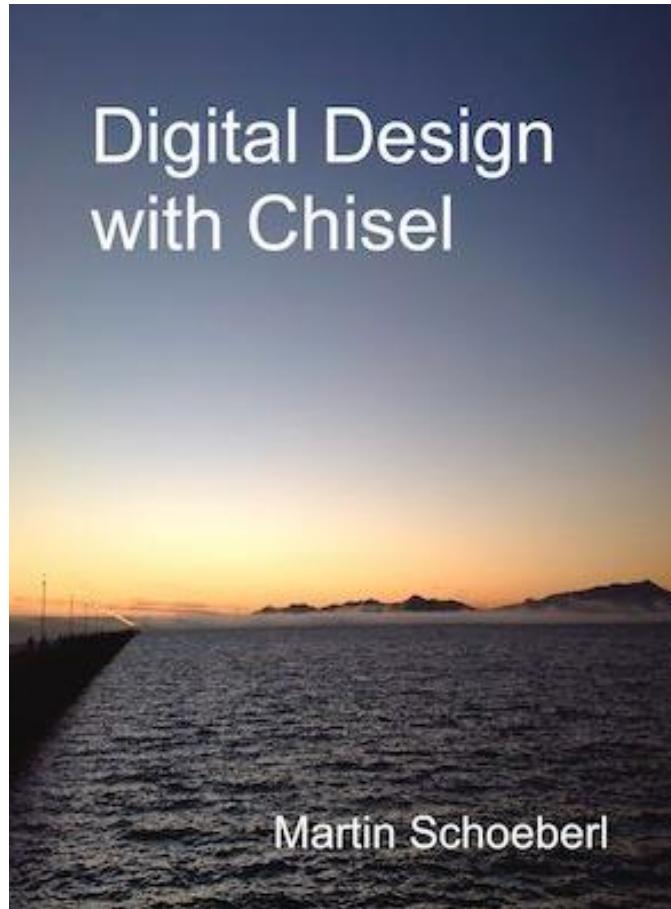
Beginners often started with CHISEL tutorial and specification.

CHISEL tutorials:

A screenshot of a GitHub repository page for 'chisel-tutorial'. The repository is public and has 89 stars. It contains 20 branches and 12 tags. The code tab is selected. A list of recent commits shows changes to files like doc, project, src, .gitignore, LICENSE.txt, README.md, build.sbt, run-examples.sh, run-problem.sh, and run-solution.sh. The commits are dated from June 2, 2020, to over 5 years ago. The repository also includes a README.md file and a Chisel Tutorials (Release branch) section.

<https://github.com/ucb-bar/chisel-tutorial>

CHISEL coding:



RISC-V specification:

A screenshot of the RISC-V Specifications website. The URL is riscv.org/technical/specifications/. The page features the RISC-V logo and a large 'Specifications' heading. Below the heading, there is a brief description of the RISC-V instruction set architecture (ISA) and related specifications. It mentions that work on the specification is performed on GitHub and lists the current ratified releases. There is also a section for the ISA Specification and a note about past ratified releases.

<https://riscv.org/technical/specifications/>

# Outline

1. Introduction
2. What makes a processor and computer system?
3. Instruction Set Architecture (ISA) and RISC-V
4. Necessary tools for working with RISC-V
5. RISC-V open-source community and materials
6. Some RISC-V news

# 6. Some RISC-V news (1/8) NASA



HOME SERVICES NEWS EDUCATION ABOUT US

Search

## NASA Selects SiFive and Makes RISC-V the Go-to Ecosystem for Future Space Missions

*SiFive X280 delivers 100x increase in computational capability with leading power efficiency, fault tolerance, and compute flexibility to propel next-generation planetary and surface missions*

September 06, 2022 12:00 PM Eastern Daylight Time

[link](#)

The Register®

## SiFive RISC-V CPU cores to power NASA's next spaceflight computer

After more than two decades, the space agency's PowerPC love affair appears to be at an end

Tobias Mann

[link](#)

Tue 6 Sep 2022 // 23:14 UTC

EE Times 50  
1972-2022

HOME NEWS ▾ PERSPECTIVES DESIGNLINES ▾ PODCASTS EDUCATION ▾ STORE

DESIGNLINES | OPEN SOURCE DESIGNLINE

## NASA Uses RISC-V Vector Spec to Soup Up Space Computers

Guest Blog  
By Chenny Wang 11.15.2022 □ 0

[link](#)

# 6. Some RISC-V news (2/8) Google

ars TECHNICA

BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

AMPUTATING ARM —

## Google wants RISC-V to be a “tier-1” Android architecture

Google's keynote at the RISC-V Summit promises official, polished support.

RON AMADEO - 1/4/2023, 4:14 AM

[link](#)



Android Open Source Project ports to RISC-V

Technology News | November 1, 2022

By Nick Flaherty

SOFTWARE & EMBEDDED TOOLS

RISC-V

The Register®

The Android Open Source Project (AOSP) has been ported to the RISC-V processor architecture in a key move for the technology.

## SiFive RISC-V cores picked for Google AI compute nodes

[link](#)

Cor, that's a shot in the arm for this upstart CPU ISA

 Dan Robinson

[link](#)

Fri 23 Sep 2022 // 21:25 UTC

# 6. Some RISC-V news (3/8) Ubuntu

**phoronix**

ARTICLES & REVIEWS NEWS ARCHIVE FORUMS PREMIUM CONTACT CATEGORIES



**Show Your Support:** Did you know that the hundreds of articles written on Phoronix each month are mostly authored by one individual? Phoronix.com doesn't have a whole news room with unlimited resources and relies upon people reading our content without blocking ads and alternatively by people subscribing to Phoronix Premium for our ad-free service with other extra features.

Ubuntu 22.10 Up And Running On The LicheeRV ~\$19 RISC-V Board

[link](#)

**The Register®**

kaspersky APAC & Global outlook: Threat landscape and predictions 18 January 2023, 1:00pm SGT (GMT+8)

Ubuntu continues expanding RISC-V support now, the \$17 Sipeed LicheeRV

As progress revealed on Android port to the open ISA

Tobias Mann

[link](#)

Projects ▾ Channels ▾ News Contests Events Videos

## Sipeed Teases a RISC-V System-on-Module That "Beats the Raspberry Pi 4" in Performance

With four RISC-V cores running at 2.5GHz and up to 16GB of RAM, the compact Lichee Module 4 A aims to outperform Raspberry Pi's best board.

 Gareth Halfacree [Follow](#)  
a month ago • HW101

[link](#)

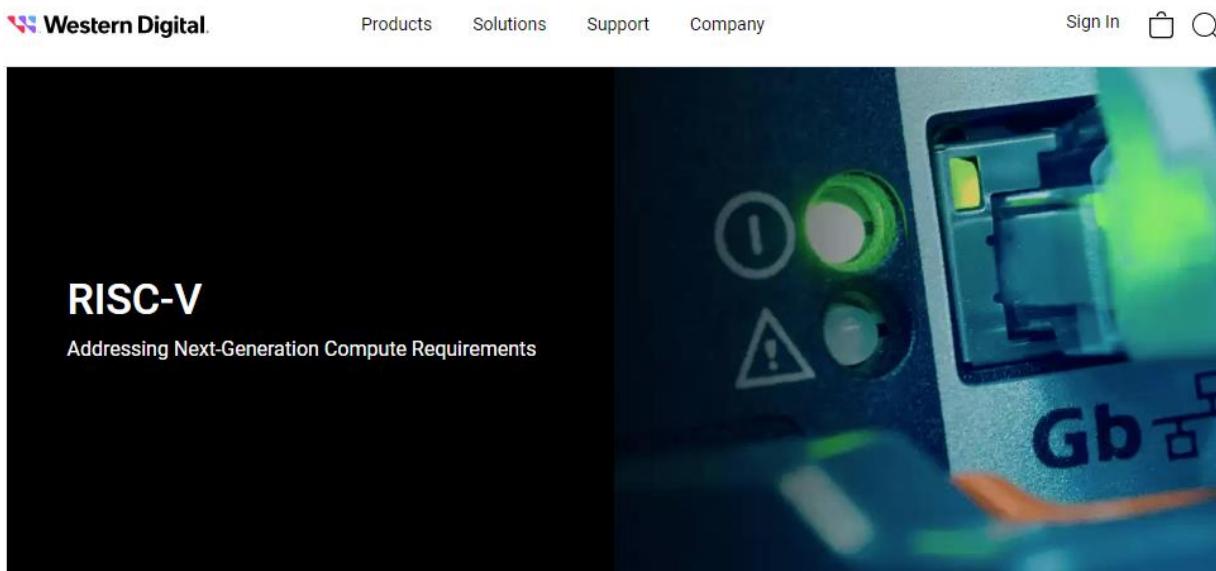
**Inputing** Reviews Deals Mini PCs Amazon Devices ▾ Linux Smartphones ▾ Shop About ▾

## Now you can run Ubuntu on a RISC-V computer that costs less than \$20

 by BRAD LINDER  
10/25/2022  
5 Comments

[link](#)

# 6. Some RISC-V news (4/8) WD & Seagate



The screenshot shows the Western Digital RISC-V landing page. At the top, there's a navigation bar with links for Products, Solutions, Support, Company, Sign In, and a search icon. Below the navigation is a large image of a computer component with glowing green lights and a blue PCB. To the left of the image, the text "RISC-V" is displayed in large letters, followed by "Addressing Next-Generation Compute Requirements".

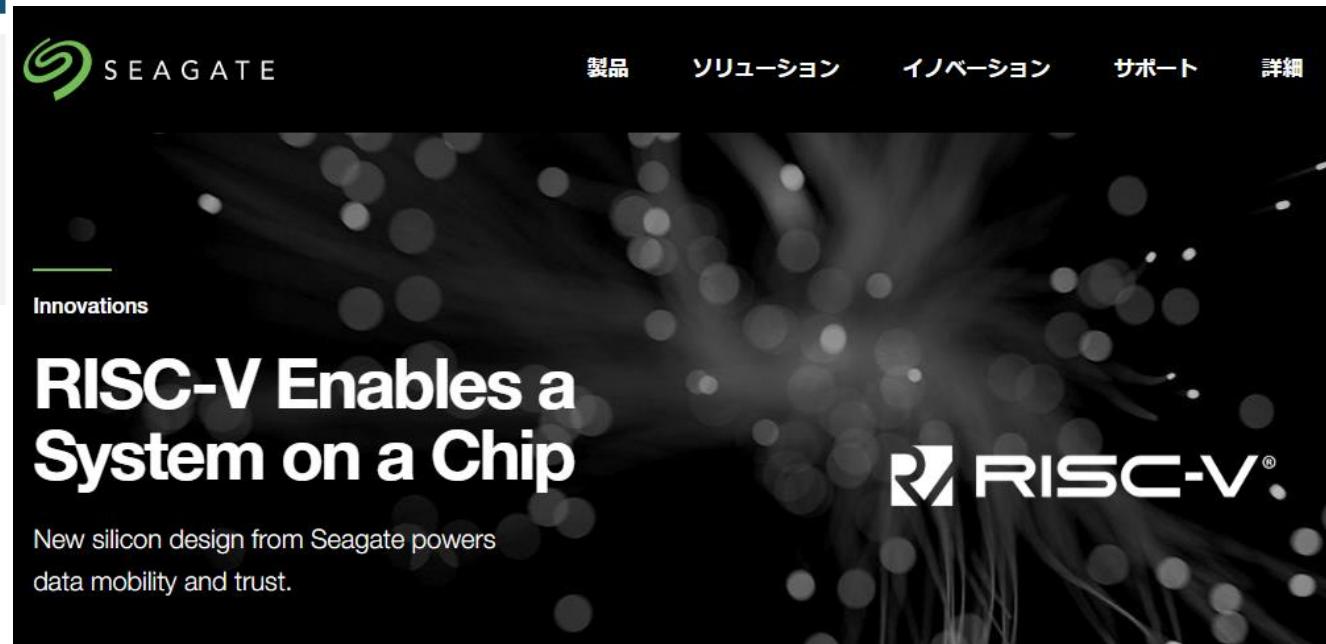
## Unleashing the power of data through RISC-V initiatives

The open source model, proven by the success of Linux®, now has a hardware platform in RISC-V to enable the next generation of innovation. Another reason is that the configurability RISC-V provides is unparalleled. By leveraging the open collaboration and flexibility of RISC-V, Western Digital can create processors that are purpose-built for data-centric applications.

[link](#)

[link](#)

Promising upcoming products in  
**Western Digital and Seagate** [link](#)



The screenshot shows the Seagate RISC-V landing page. At the top, there's a navigation bar with links for 製品 (Products), ソリューション (Solutions), イノベーション (Innovation), サポート (Support), and 詳細 (Details). The Seagate logo is on the left. The main headline reads "RISC-V Enables a System on a Chip". Below the headline, a subtext states "New silicon design from Seagate powers data mobility and trust." The RISC-V logo is visible in the bottom right corner.

## 6. Some RISC-V news (5/8) Renesas

**embedded**

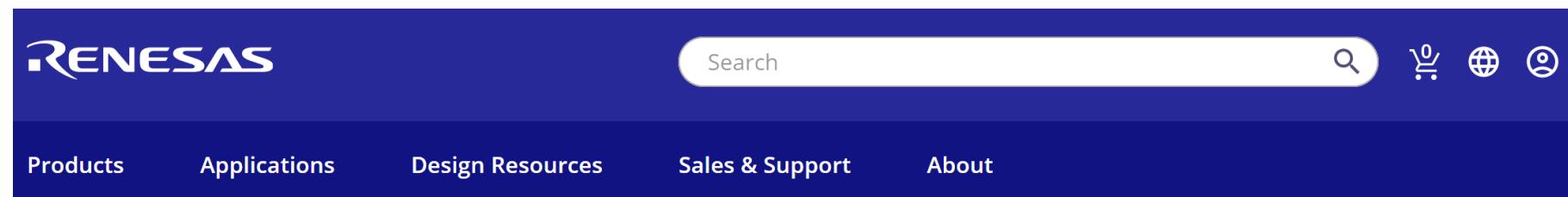
Embedded Focus ▾ News Technical Articles About us ▾

Products

# Renesas delivers RISC-V based motor control ASSP

⌚ September 13, 2022 Nitin Dahad

[link](#)



The header features the Renesas logo in white on a dark blue background. To the right is a search bar with a magnifying glass icon and three small circular icons for user account, globe, and shopping cart. Below the header are five navigation links: Products, Applications, Design Resources, Sales & Support, and About.

Press Room ▶ News ▶ Renesas Extends Leading RISC-V Embedded Processing Portfolio with New Motor Control ASSP Solution

[link](#) Renesas Extends Leading RISC-V Embedded Processing Portfolio with New Motor Control ASSP Solution

# 6. Some RISC-V news (6/8) Alibaba & Microchip

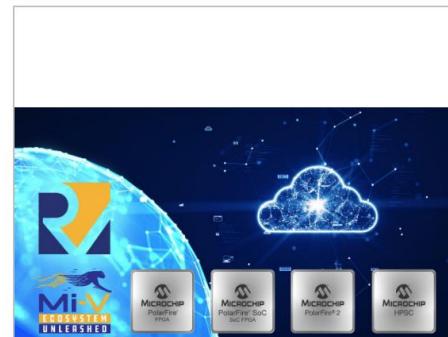
[LOG IN](#)[JOIN](#)[ARTICLES](#)   [FORUMS](#)   [EDUCATION](#)   [TOOLS](#)   [VIDEOS](#)   [DATASHEETS](#)   [GIVEAWAYS](#)   [TECH COMMUNITIES](#)

Push the Boundaries of Your Medical Designs | Advance, Expand and Differentiate Your

[Home](#) > [News](#) > [Alibaba RISC-V SoC Revealed as Processor for First RISC-V Laptop](#)[NEWS](#)

## Alibaba RISC-V SoC Revealed as Processor for First RISC-V Laptop

October 10, 2022 by [Jake Hertz](#)

[link](#)[Products](#)   [Solutions](#)   [Tools and Resources](#)   [Support](#)   [Education](#)   [About](#)   [Order Now](#)[About / Microchip Showcases RISC-V-Based FPGA and Space-Compute Solutio...](#)[link](#)

## Microchip Showcases RISC-V-Based FPGA and Space-Compute Solutions at RISC-V Summit

PolarFire® devices lead in delivering 2X power efficiency, military grade security and highest reliability, which will be extended by the PolarFire 2 FPGA roadmap

# 6. Some RISC-V news (7/8) Intel

**SCIENTIFIC COMPUTING WORLD**

For scientists, researchers and engineers who use computing in their work.

Search

**in** For the latest news direct from the editorial team **CLICK HERE TO CONNECT** SCIENTIFIC COMPUTING WORLD

Home News Analysis & Opinion Features Issues Events Resources Press Releases Suppliers

Your guide for the digital transformation journey

NEWS / HPC / INTEL PATHFINDER ...

Intel Pathfinder for risc-v delivers new capabilities silicon development

1 September 2022

[link](#)



intel®  
PATHFINDER  
FOR RISC-V

A development environment to start your RISC-V journey

GET STARTED

[pathfinder.intel.com](http://pathfinder.intel.com)

ALL ABOUT CIRCUITS

ARTICLES FORUMS EDUCATION TOOLS VIDEOS DATASHEETS GIVEAWAYS TECH COMMUNITIES

Push the Boundaries of Your Medical Designs | Advance, Expand and Differentiate Your

Home > News > Intel Shows Support for RISC-V Chip Design With Intel Pathfinder

NEWS

## Intel Shows Support for RISC-V Chip Design With Intel Pathfinder

September 08, 2022 by Jake Hertz

f t in y

[link](#)

# 6. Some RISC-V news (8/8) Tencent & EU

Tech / Big Tech

## Tencent joins open-source chip design community RISC-V as China seeks to mitigate impact from US sanctions

- Amid tighter US restrictions on chip technology exports to China, Beijing is pinning hopes on RISC-V as an alternative to the Intel and Arm standards
- Analysts say RISC-V alone cannot remedy China's weakness in chip tech, as the country still relies on imported tools and software for design

[link](#)



Why you can trust SCMP

TABOR NETWORK:

DATANAMI

ENTERPRISEAI

HPCWIRE JAPAN

QCWIRE

HPC & AI WALL STREET

**HPC** wire

Since 1987 - Covering the Fastest Computers  
in the World and the People Who Run Them

- ❖ Home
- ❖ Topics
- ❖ Sectors
- ❖ Exascale
- ❖ Specials

[link](#)



December 16, 2022



国立大学法人

電気通信大学

The University of Electro-Communications

Pham Laboratory  
Integrated circuit design laboratory

THANK YOU