RISC-V

What is RISC-V?

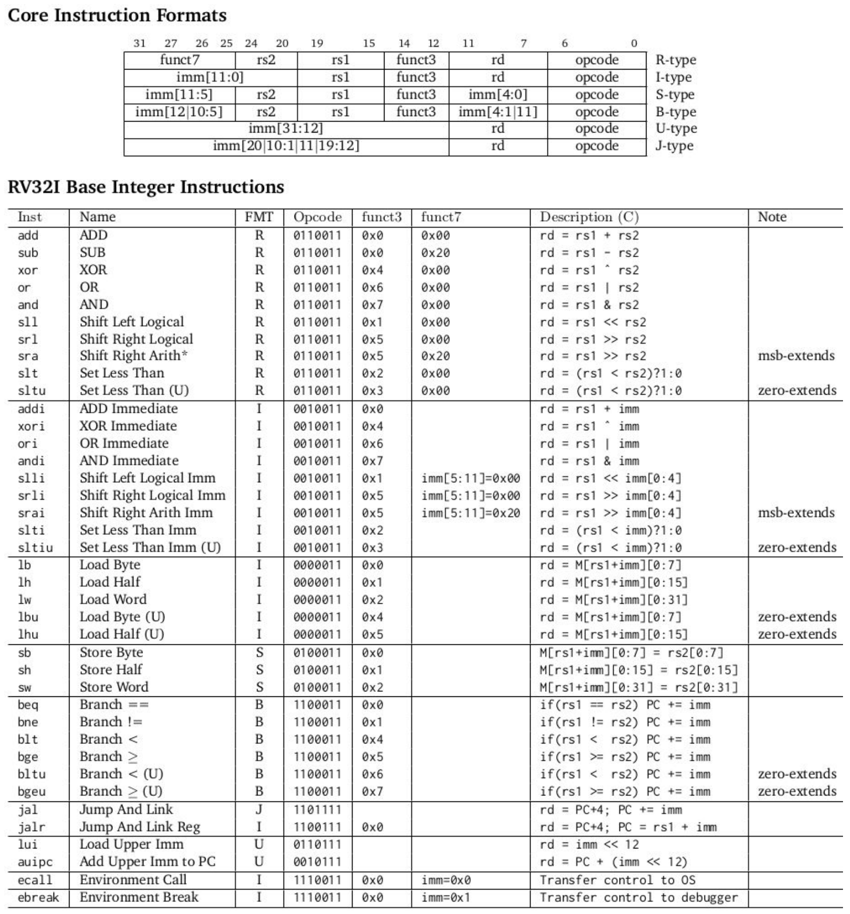
Disclaimer: Apple users may not understand this slide 😈

What is RISC-V?

- Simple and modular Instruction Set Architecture (ISA)
- Research project started at Berkeley in 2010
- In 2014 ISA ratified
- RISC-V (pronounced “risc five”, as it is the fifth generation of RISC ISA at Berkeley)

📙 Waterman, Andrew., Patterson, David A.. **The RISC-V Reader: An Open Architecture Atlas**. United States: Strawberry Canyon LLC, 2017.

📙 Patterson, David A.., Hennessy, John L.. **Computer Organization and Design RISC-V Edition: The Hardware Software Interface**. Netherlands: Elsevier Science, 2017.



Core Instruction Formats

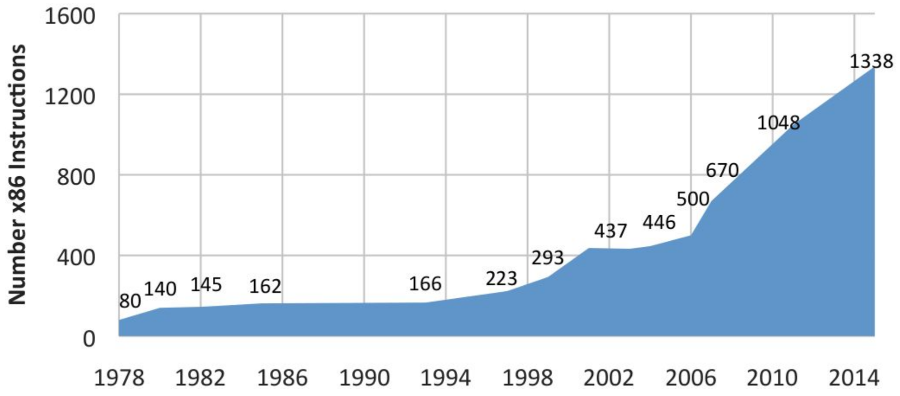
31 27 26 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]		rs1	funct3	rd	opcode	I-type
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode	B-type
imm[31:12]				rd	opcode	U-type
imm[20|10:1|11|19:12]				rd	opcode	J-type

RV32I Base Integer Instructions

Inst	Name	FMT	Opcode	funct3	funct7	Description (C)	Note
add	ADD	R	0110011	0x0	0x00	rd = rs1 + rs2	
sub	SUB	R	0110011	0x0	0x20	rd = rs1 - rs2	
xor	XOR	R	0110011	0x4	0x00	rd = rs1 ˆ rs2	
or	OR	R	0110011	0x6	0x00	rd = rs1 | rs2	
and	AND	R	0110011	0x7	0x00	rd = rs1 & rs2	
sll	Shift Left Logical	R	0110011	0x1	0x00	rd = rs1 << rs2	
srl	Shift Right Logical	R	0110011	0x5	0x00	rd = rs1 >> rs2	
sra	Shift Right Arith*	R	0110011	0x5	0x20	rd = rs1 >> rs2	msb-extends
slt	Set Less Than	R	0110011	0x2	0x00	rd = (rs1 < rs2)?1:0	
sltu	Set Less Than (U)	R	0110011	0x3	0x00	rd = (rs1 < rs2)?1:0	zero-extends
addi	ADD Immediate	I	0010011	0x0		rd = rs1 + imm	
xori	XOR Immediate	I	0010011	0x4		rd = rs1 ˆ imm	
ori	OR Immediate	I	0010011	0x6		rd = rs1 | imm	
andi	AND Immediate	I	0010011	0x7		rd = rs1 & imm	
slli	Shift Left Logical Imm	I	0010011	0x1	imm[5:11]=0x00	rd = rs1 << imm[0:4]	
srli	Shift Right Logical Imm	I	0010011	0x5	imm[5:11]=0x00	rd = rs1 >> imm[0:4]	
srai	Shift Right Arith Imm	I	0010011	0x5	imm[5:11]=0x20	rd = rs1 >> imm[0:4]	msb-extends
slti	Set Less Than Imm	I	0010011	0x2		rd = (rs1 < imm)?1:0	
sltiu	Set Less Than Imm (U)	I	0010011	0x3		rd = (rs1 < imm)?1:0	zero-extends
lb	Load Byte	I	0000011	0x0		rd = M[rs1+imm][0:7]	
lh	Load Half	I	0000011	0x1		rd = M[rs1+imm][0:15]	
lw	Load Word	I	0000011	0x2		rd = M[rs1+imm][0:31]	
lbu	Load Byte (U)	I	0000011	0x4		rd = M[rs1+imm][0:7]	zero-extends
lhu	Load Half (U)	I	0000011	0x5		rd = M[rs1+imm][0:15]	zero-extends
sb	Store Byte	S	0100011	0x0		M[rs1+imm][0:7] = rs2[0:7]	
sh	Store Half	S	0100011	0x1		M[rs1+imm][0:15] = rs2[0:15]	
sw	Store Word	S	0100011	0x2		M[rs1+imm][0:31] = rs2[0:31]	
beq	Branch ==	B	1100011	0x0		if(rs1 == rs2) PC += imm	
bne	Branch !=	B	1100011	0x1		if(rs1 != rs2) PC += imm	
blt	Branch <	B	1100011	0x4		if(rs1 < rs2) PC += imm	
bge	Branch ≥	B	1100011	0x5		if(rs1 >= rs2) PC += imm	
bltu	Branch < (U)	B	1100011	0x6		if(rs1 < rs2) PC += imm	zero-extends
bgeu	Branch ≥ (U)	B	1100011	0x7		if(rs1 >= rs2) PC += imm	zero-extends
jal	Jump And Link	J	1101111			rd = PC+4; PC += imm	
jalr	Jump And Link Reg	I	1100111	0x0		rd = PC+4; PC = rs1 + imm	
lui	Load Upper Imm	U	0110111			rd = imm << 12	
auipc	Add Upper Imm to PC	U	0010111			rd = PC + (imm << 12)	
ecall	Environment Call	I	1110011	0x0	imm=0x0	Transfer control to OS	
ebreak	Environment Break	I	1110011	0x0	imm=0x1	Transfer control to debugger	

Modular vs Incremental ISA

- Intel x86 is an incremental ISA. Each new release:
  - Maintain backward compatibility
  - Carry on new instructions (also for marketing reasons)



Name	Description	Version	Status	Instruction Count
RV32I	Base Integer Instruction Set - 32-bit	2.1	Frozen	49
RV32E	Base Integer Instruction Set (embedded) - 32-bit, 16 registers	1.9	Open	Same as RV32I
RV64I	Base Integer Instruction Set - 64-bit	2.0	Frozen	14
RV128I	Base Integer Instruction Set - 128-bit	1.7	Open	14
	Extension			
M	Standard Extension for Integer Multiplication and Division	2.0	Frozen	8
A	Standard Extension for Atomic Instructions	2.0	Frozen	11
F	Standard Extension for Single-Precision Floating-Point	2.0	Frozen	25
D	Standard Extension for Double-Precision Floating-Point	2.0	Frozen	25
G	Shorthand for the base and above extensions	n/a	n/a	n/a
Q	Standard Extension for Quad-Precision Floating-Point	2.0	Frozen	27
L	Standard Extension for Decimal Floating-Point	0.0	Open	Undefined Yet
C	Standard Extension for Compressed Instructions	2.0	Frozen	36
B	Standard Extension for Bit Manipulation	0.90	Open	42
J	Standard Extension for Dynamically Translated Languages	0.0	Open	Undefined Yet
T	Standard Extension for Transactional Memory	0.0	Open	Undefined Yet
P	Standard Extension for Packed-SIMD Instructions	0.1	Open	Undefined Yet
V	Standard Extension for Vector Operations	0.7	Open	186
N	Standard Extension for User-Level Interrupts	1.1	Open	3
H	Standard Extension for Hypervisor	1.0	Frozen	2
S	Standard Extension for Supervisor-level Instructions	1.12	Open	7

📙 Waterman, Andrew., Patterson, David A.. **The RISC-V Reader: An Open Architecture Atlas**. United States: Strawberry Canyon LLC, 2017.
🔖 https://en.wikichip.org/wiki/risc-v/standard_extensions

Extension

What is it?

- Optional set of instructions that can be added to the base instruction set to enhance functionality.
    a. base instruction set (e.g., RV32I, RV64I) provides fundamental operations
    b. extensions add features like floating-point arithmetic (F), vector processing (V), or atomic operations (A).

Key Points & Reasons:

- **Modularity:** Allows designers to tailor a processor to specific needs.
- **Scalability:** Reduces complexity in applications that does not need all features.
- **Compatibility:** Extensions follow strict rules to ensure future compatibility.

Comparison with Other Architectures:

- **x86:** Monolithic ISA, where many legacy instructions persist for compatibility.
- **Arm:** Has optional features (SIMD, floating-point, etc.), but its licensing model restricts customization.

Who guarantees that each RISC-V CPU does not supports a 🥗 of instructions?

Profile

What is it?

- Predefined combination of base instructions + extensions
- It defines a standard configuration for a category of RISC-V processors.
- It ensures software compatibility across different implementations.

Key Points & Reasons:

- **Standardization:** Helps software developers by ensuring that all processors within a profile support a consistent instruction set.
- **Optimization:** Targets specific workloads (e.g., embedded vs. high-performance computing).

Comparison with Other Architectures:

- **x86:** N/A. Processor generations introduce new instruction sets (e.g., SSE, AVX).
- **Arm:** Similar concept via architecture profiles
  a. A for application processors, R for real-time, M for microcontrollers

Platform

What is it?

- It defines a complete system environment, including hardware, firmware, and software conventions.
- It specifies how a RISC-V processor interacts with peripherals, boot sequences, operating system expectations, etc.

Key Points & Reasons:

- Ensures interoperability between software and hardware.
- Facilitates OS and firmware development.
- Reduces fragmentation by defining standard system behavior.

Comparison with Other Architectures:

- **x86:** Well-defined platform standards (e.g., UEFI for booting, ACPI for power management).
- **Arm:** Arm SystemReady provides a similar concept.

Modes

What is it?

- RISC-V defines multiple privileged modes for different levels of access to system resources:
    a. User Mode (U) – For normal applications, limited access.
    b. Supervisor Mode (S) – For OS kernels, can manage memory and processes.
    c. Machine Mode (M) – The highest privilege level, used for low-level firmware and system initialization.

Key Points & Reasons:

- **Security:** Restricts access to sensitive hardware.
- **OS Functionality:** Enables multi-user and multi-tasking systems.
- **Power Efficiency:** Allows fine-grained control over system resources.

Comparison with Other Architectures:

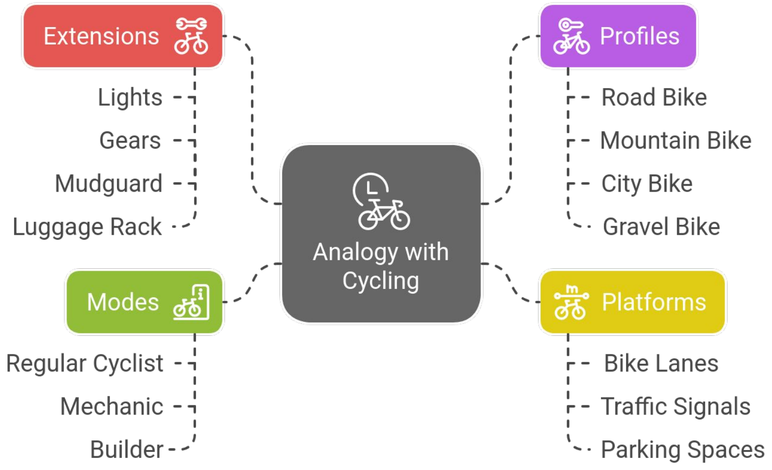
- **x86:** Has four rings (0-3), where Ring 0 is the most privileged (used by the OS kernel).
- **Arm:** Uses EL0-EL3 (Exception Levels), where EL3 is the most privileged.

Relationship among concepts

- **Extensions** define additional features that a processor can have.
- **Profiles** group together a fixed set of extensions to create standard processor configurations.
- **Platforms** define a full system around a processor, ensuring compatibility at the system level.
- **Modes** control access to the processor and system resources, impacting how software interacts with the hardware.

Analogy with 🚲

- **Extensions** = optional accessories
  a. e.g., lights, gears, mudguard, luggage rack, …
- **Profiles** = bike models, where each model comes with a predefined set of features.
  a. Road bike, Mountain bike, City bike, Gravel bike, …
- **Platforms** = cycling infrastructure that defines how bikes interact with the environment
  a. bike lanes, traffic signals, parking spaces, …
- **Modes** = types of access to the bike
  a. **User Mode (U-mode)** → A regular cyclist.
  b. **Supervisor Mode (S-mode)** → A bike mechanic that performs regular maintenance.
  c. **Machine Mode (M-mode)** → A factory technician that assemble or disassemble the bike completely.



Standards are fundamental for technology!

Area	Standard	Implementations
Network	Ethernet	10BaseT, 1000BaseT, …
OS	POSIX	Linux, masOS, …
Wireless	Wi-Fi	Wi-Fi 5 (802.11ac), Wi-Fi 6 (802.11ax)
Graphics	OpenGL	NVIDIA GeForce, Mesa 3D (open source)
File System	FAT	FAT32, exFAT
Web Communication	HTTP	Apache Server, Nginx, …
Interconnect	PCIe	PCIe 3.0, PCIe 5.0, …
Processor Architecture	🦗	🦗

Instruction Set Architecture is the most important interface because it regulates how software interacts with hardware

Short history of ISAs

- Intel today uses x86-64 (formerly AMD64), but in the past…
  - 8086 (i860) ☠️
  - Itanium ☠️
- SPARC (Berkeley)
  - Developed by Sun in the 80's, became SPARC International (Sun Microsystems, Fujitsu, Texas Instruments, Cray, Ross, and others)
  - The architecture of the K-computer
  - ☠️ in 2017
- MIPS (Stanford)
  - Sold to Imagination Technologies (PowerVR graphics processors)
  - Made it open in 2018, then closed again in 2019
  - Now RISC-V ☠️?
- IBM Power
  - Closed ISA that became open in 2019. Nobody adopted it. ☠️?
- Arm
  - Sold to Softbank (Japanese multinational) in 2016
  - Acquisition by NVIDIA announced in 2020, but deal cancelled in 2022

🧐

If you were to start your software project tomorrow, what ISA would you rely on?

More in general, why a company should trust a closed source ISA for its software?

More in general, why don't we have an open standard ISA with multiple (open/closed) implementations?

Non-technical differences: business models

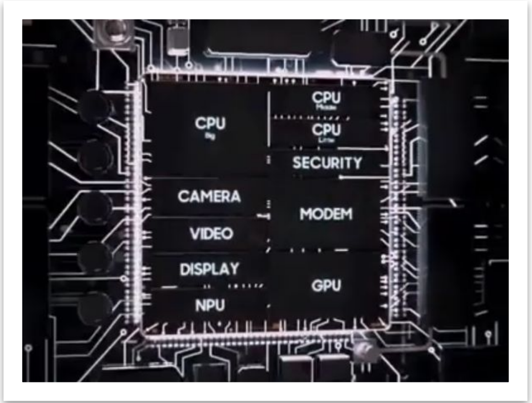
	intel	arm	RISC-V
Open ISA	□	□	□
Adopting the ISA is free	□ [1]	□ [2]	□
It allows development of commercial IPs	□	□	□
Everybody can develop commercial IPs	□	□ [3]	□
It allows access to extension	□	□	□
It allows development of open-source IPs	□	□	□

[1] Adoption of the ISA “de facto” not possible (unless you are AMD)
[2] Adoption of the ISA is possible (under a fee)
[3] Only partners can develop commercial IPs (under a fee)

SoC fragmentation: two sides of the same 🪙

System on Chip ~= Several hardware components within the same chip

- Very likely, each component with its own ISA
  - Therefore, with a different software stack

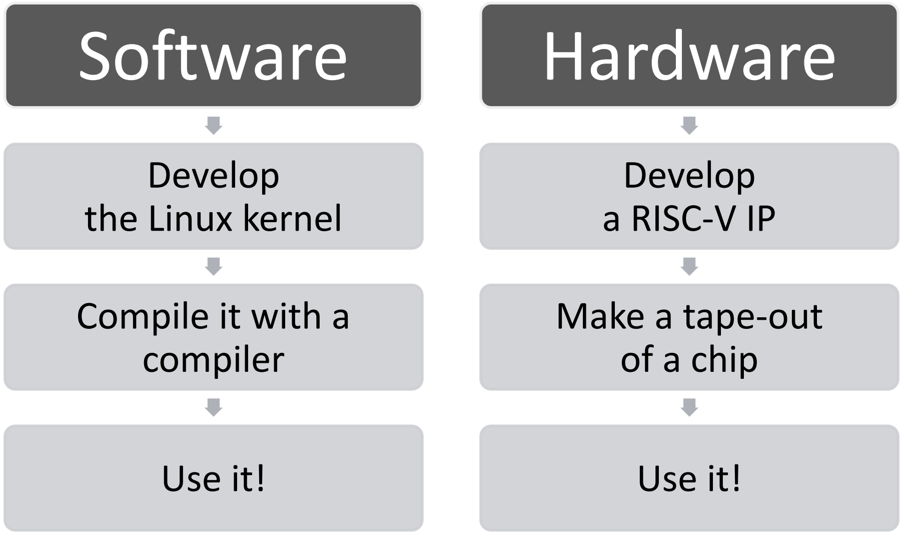


Why?

- ISAs are too big / too rigid for specific components
- Often SoC provider buys IPs from third parties
  - Forced to buy IP + ISA (both proprietary)
- Sometimes there are home-made IP with home-made ISAs
  - Bad practice!

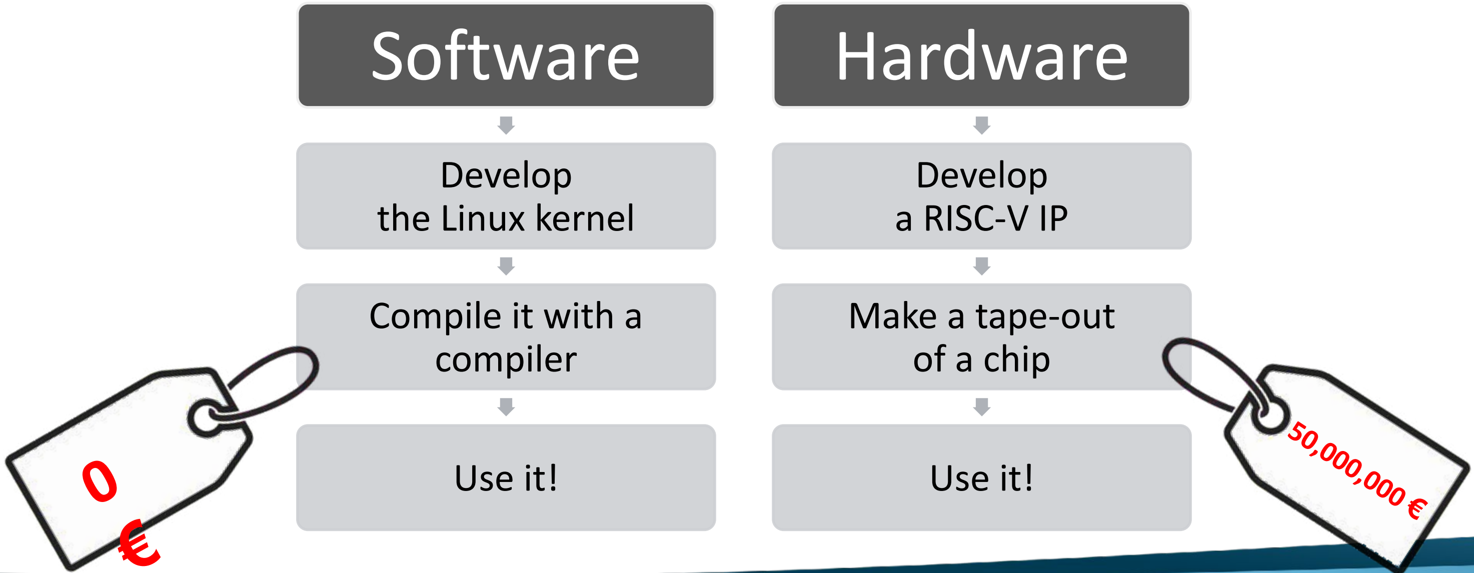
It is not like Linux for software

- A shallow analysis often uses the analogy
  - “It is the same idea of Linux but in hardware”
  - “RISC-V will do to the hardware what Linux did to software”



It is not like Linux for software

- A shallow analysis often uses the analogy
  - "It is the same idea of Linux but in hardware"
  - "RISC-V will do to the hardware what Linux did to software"



Resilient in a delicate geopolitical context

✅ No single country/company controls RISC-V

✅ ISA is free from embargoes and restrictions

✅ Innovation can start without licensing barriers

Control
No single entity governs RISC-V.
Freedom
ISA is free from embargoes and restrictions.
Innovation
Development can begin without licensing barriers.

RISC-V CPUs / Platforms

- Sophgo 🇨🇳
    - a. SG2042: 64-core RISC-V processor (available on Milk-V Pioneer)
    - b. SG2044: Enhanced version with RISC-V vector version 1.0 and PCIe Gen5 support (announced)
    - c. SG2380: 16-core processor based on SiFive P670 cores with a 20 TOPS AI accelerator (announced)
- SiFive 🇺🇸
    - a. Freedom U740: available on HiFive Unmatched
    - b. ESWIN EIC7700X: available on Premier P550
    - c. P870-D: Out of order, supports RVA23 profile (announced)
- SpacemiT 🇨🇳
    - a. K1: 8-core 64-bit with profile RVA22 Profile and 256-bit RVV 1.0 (available on BananaPi F3)

Accelerators:

- Esperanto 🇺🇸
    - ET-SoC-1: 1088-cores ET Minion 64-bit RISC-V in-order cores + 4 high-performance ET-Maxion 64-bit RISC-V out of-order cores (announced)
- Tenstorrent 🇺🇸
    - Wormhole n150: 72 Tensix cores (available)

Take-home message

📽 Krste Asanović, SiFive, **Advancing HPC with RISC-V** (Supercomputing 2022, invited talk) https://www.youtube.com/watch?v=iFlcJFcOJKk

- RISC-V defines an open, free and standard ISA
  - Simple and modular (as opposed to incremental)
- It allows to homogenize hardware designs (SoC)
  - Simplifying software stack of final products
- It defines a new business model
  - Pick a profile, then find a vendor or build your own IP
  - Add your extensions, if needed
  - Collaborate with academia, companies and open-source communities
- It is not dependent on market fluctuations (war, bans, …)
- Some CPU already in the market