RISC-V: A Free and Open Source ISA

Graham Markall

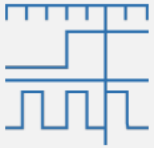
Compiler Engineer, Embecosm

graham.markall@embecosm.com / @gmarkall

Why get excited about RISC-V?

About Embecosm

Toolchain Porting >



Hardware Modeling >

Open Source Support >

Machine Learning Optimization >

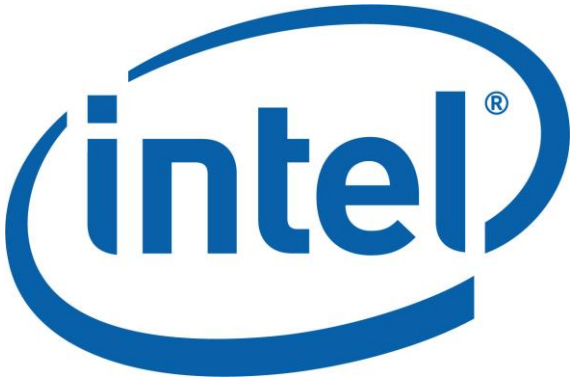
Energy Efficient Compilation >

Superoptimization >

The RISC-V ISA

- **RISC-V** is a Free and Open *Instruction Set Architecture (ISA)*

- Some other ISAs:



What's in a ISA document?

Registers:

Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5	t0	Temporary/alternate link register	Caller
x6–7	t1–2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10–11	a0–1	Function arguments/return values	Caller
x12–17	a2–7	Function arguments	Caller
x18–27	s2–11	Saved registers	Callee
x28–31	t3–6	Temporaries	Caller
f0–7	ft0–7	FP temporaries	Caller
f8–9	fs0–1	FP saved registers	Callee
f10–11	fa0–1	FP arguments/return values	Caller
f12–17	fa2–7	FP arguments	Caller
f18–27	fs2–11	FP saved registers	Callee
f28–31	ft8–11	FP temporaries	Caller

Instructions + Semantics

Pseudoinstruction	Base Instruction(s)	Meaning
`la rd, symbol`	`auipc rd, symbol[31:12]` `addi rd, rd, symbol[11:0]`	Load address
`l{b|h|w|d} rd, symbol`	`auipc rd, symbol[31:12]` `l{b|h|w|d} rd, symbol[11:0](rd)`	Load global
`s{b|h|w|d} rd, symbol, rt`	`auipc rt, symbol[31:12]` `s{b|h|w|d} rd, symbol[11:0](rt)`	Store global
`fl{w|d} rd, symbol, rt`	`auipc rt, symbol[31:12]` `fl{w|d} rd, symbol[11:0](rt)`	Floating-point load global
`fs{w|d} rd, symbol, rt`	`auipc rt, symbol[31:12]` `fs{w|d} rd, symbol[11:0](rt)`	Floating-point store global
`nop`	`addi x0, x0, 0`	No operation
`li rd, immediate`	*Myriad sequences*	Load immediate
`mv rd, rs`	`addi rd, rs, 0`	Copy register

Encodings

RV64I Base Instruction Set (in addition to RV32I)

imm[11:0]		rs1	110	rd	0000011	LWU
imm[11:0]		rs1	011	rd	0000011	LD
imm[11:5]	rs2	rs1	011	imm[4:0]	0100011	SD
000000	shamt	rs1	001	rd	0010011	SLLI
000000	shamt	rs1	101	rd	0010011	SRLI
010000	shamt	rs1	101	rd	0010011	SRAI
imm[11:0]		rs1	000	rd	0011011	ADDIW
0000000	shamt	rs1	001	rd	0011011	SLLIW
0000000	shamt	rs1	101	rd	0011011	SRLIW
0100000	shamt	rs1	101	rd	0011011	SRAIW
0000000	rs2	rs1	000	rd	0111011	ADDW
0100000	rs2	rs1	000	rd	0111011	SUBW
0000000	rs2	rs1	001	rd	0111011	SLLW
0000000	rs2	rs1	101	rd	0111011	SRLW
0100000	rs2	rs1	101	rd	0111011	SRAW

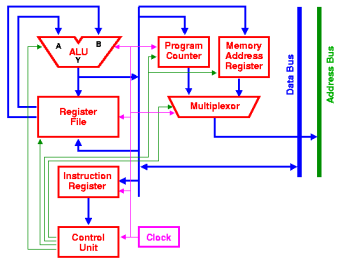
What's implemented around an ISA?

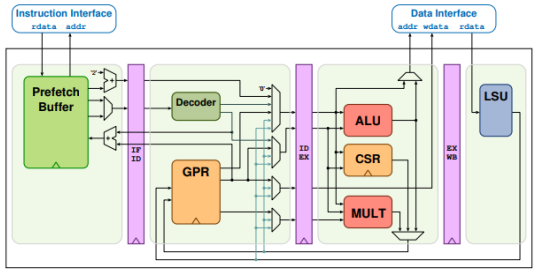
ISA Specification

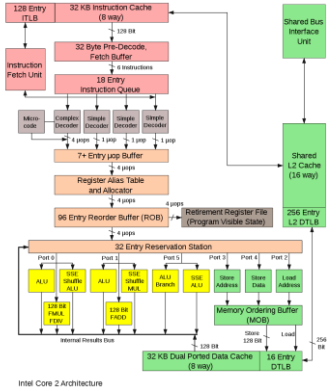
What's implemented around an ISA?

ISA Specification

Hardware





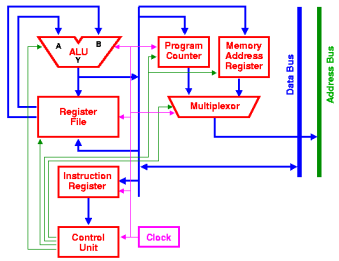


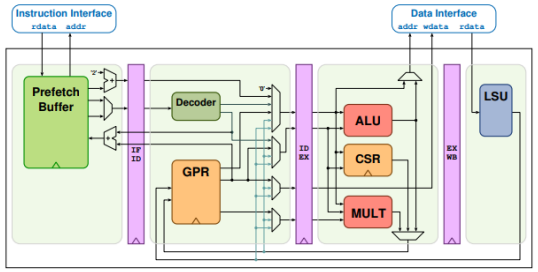
What's implemented around an ISA?

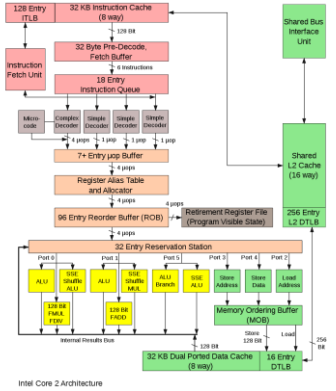
Software

ISA Specification

Hardware



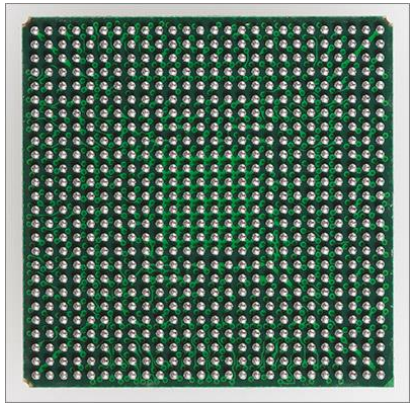




The RISC-V ISA

EMBECOSM®

- **A Boring ISA!**
- Somewhat MIPS-like, but simpler
- Relatively simple instructions and register files
- Wide range of applications
  - Tiny, deeply-embedded systems
  - Multicore, high-performance systems



Base ISAs

Base ISA	Registers	Register width	Instruction width
RV32E	16	32 bit	32 bit
RV32I	32	32 bit	32 bit
RV64I	32	64 bit	32 bit

- I/E – Integer arithmetic, no multiply or divide

Extensions

Integer	I
Integer Multiply / Divide	M
Atomics	A
Single-precision floating-point	F
Double-precision floating-point	D

- G = IMAFD
- General purpose Linux system: RV64GC (RV64IMAFDC)

Quad-precision floating-point	Q
Decimal floating-point	L
16-bit compressed instructions	C
Bit manipulation	B
Dynamic languages	J
Transactional memory	T
Packed-SIMD extensions	P
Vector extensions	V
User-level interrupts	N

RISC-V Ecosystem (a subset!)

RISC-V Ecosystem (a subset!)

EMBECOSM®

Software

Hardware

ISA Specification

RI5CY

PULP
Parallel Ultra Low Power

PicoRV32

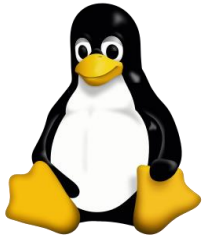
Rocket Chip Generator

Freedom

lowRISC

20+ cores…

RISC-V Ecosystem (a subset!)



Software

Hardware

ISA Specification

RI5CY

PicoRV32

Rocket Chip Generator

20+ cores…

RISC-V Ecosystem (a subset!)

EMBECOSM®

GCC

LLVM COMPILER INFRASTRUCTURE

FreeBSD

freeRTOS

fedora

Software

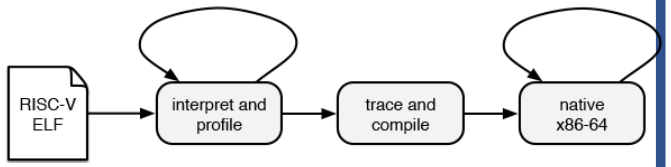
Hardware

Spike ISA simulator

ISA Specification

QEMU

rv8: RISC-V -> x86



RI5CY

PULP Parallel Ultra Low Power

PicoRV32

Rocket Chip Generator

Freedom

lowRISC

20+ cores…

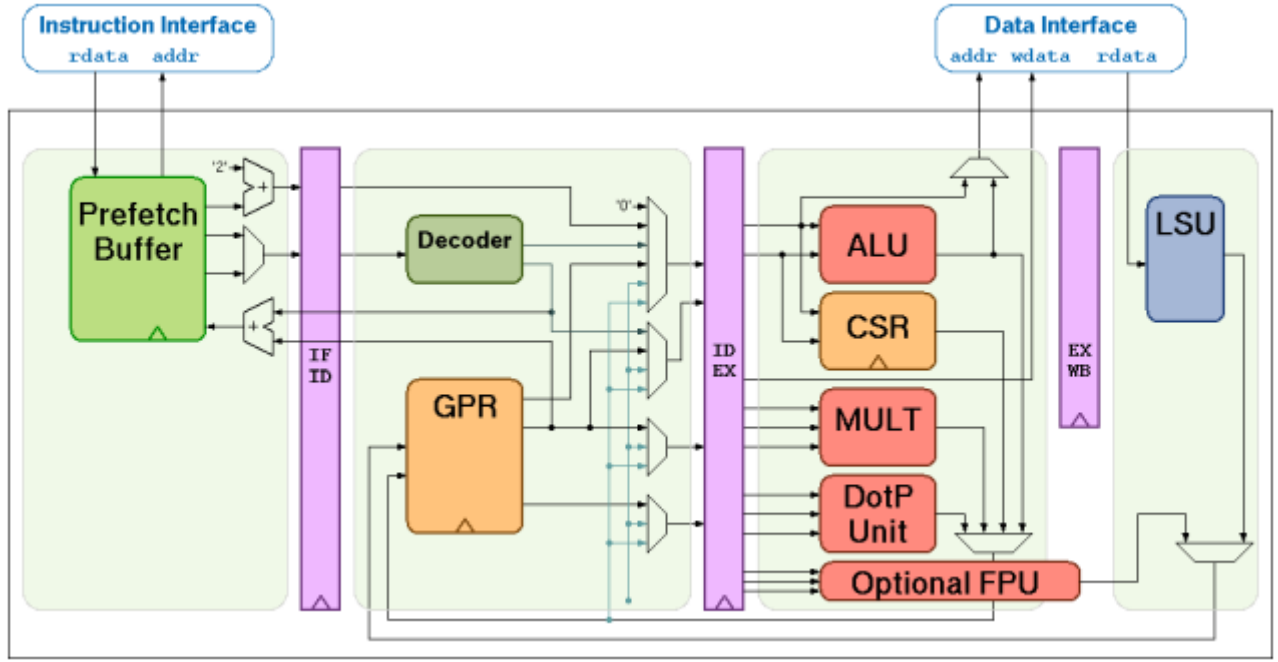
PicoRV32 – Clifford Wolf

- RV32IMC
- Great starting point
  - Small & Simple
- Fast MHz
- Slow CPI
  - (~4 in our experiments)

Device	Device	Speedgrade	Clock Period (Freq.)
Xilinx Kintex-7T	xc7k70t-fbg676-2	-2	2.4 ns (416 MHz)
Xilinx Kintex-7T	xc7k70t-fbg676-3	-3	2.3 ns (434 MHz)
Xilinx Virtex-7T	xc7v585t-ffg1761-2	-2	2.3 ns (434 MHz)
Xilinx Virtex-7T	xc7v585t-ffg1761-3	-3	2.3 ns (434 MHz)
Xilinx Kintex UltraScale	xcku035-fbva676-2-e	-2	2.1 ns (476 MHz)
Xilinx Kintex UltraScale	xcku035-fbva676-3-e	-3	1.8 ns (555 MHz)
Xilinx Virtex UltraScale	xcvu065-ffvc1517-2-e	-2	1.9 ns (526 MHz)
Xilinx Virtex UltraScale	xcvu065-ffvc1517-3-e	-3	1.8 ns (555 MHz)
Xilinx Kintex UltraScale+	xcku3p-ffva676-2-e	-2	1.5 ns (666 MHz)
Xilinx Kintex UltraScale+	xcku3p-ffva676-3-e	-3	1.3 ns (769 MHz)
Xilinx Virtex UltraScale+	xcvu3p-ffvc1517-2-e	-2	1.5 ns (666 MHz)
Xilinx Virtex UltraScale+	xcvu3p-ffvc1517-3-e	-3	1.4 ns (714 MHz)

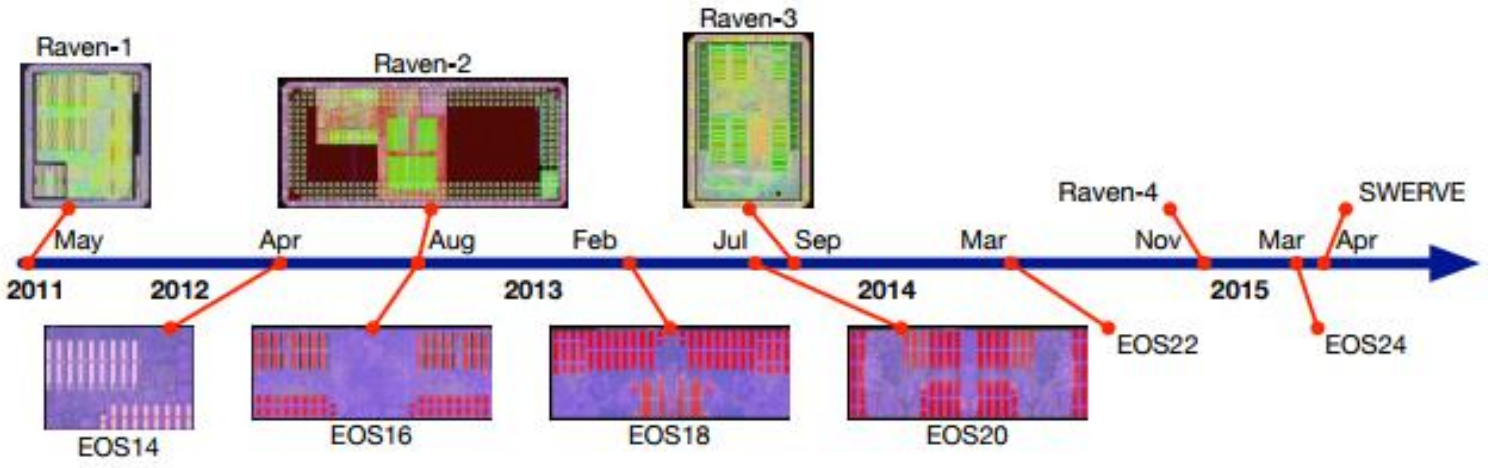
RI5CY - PuLP Platform

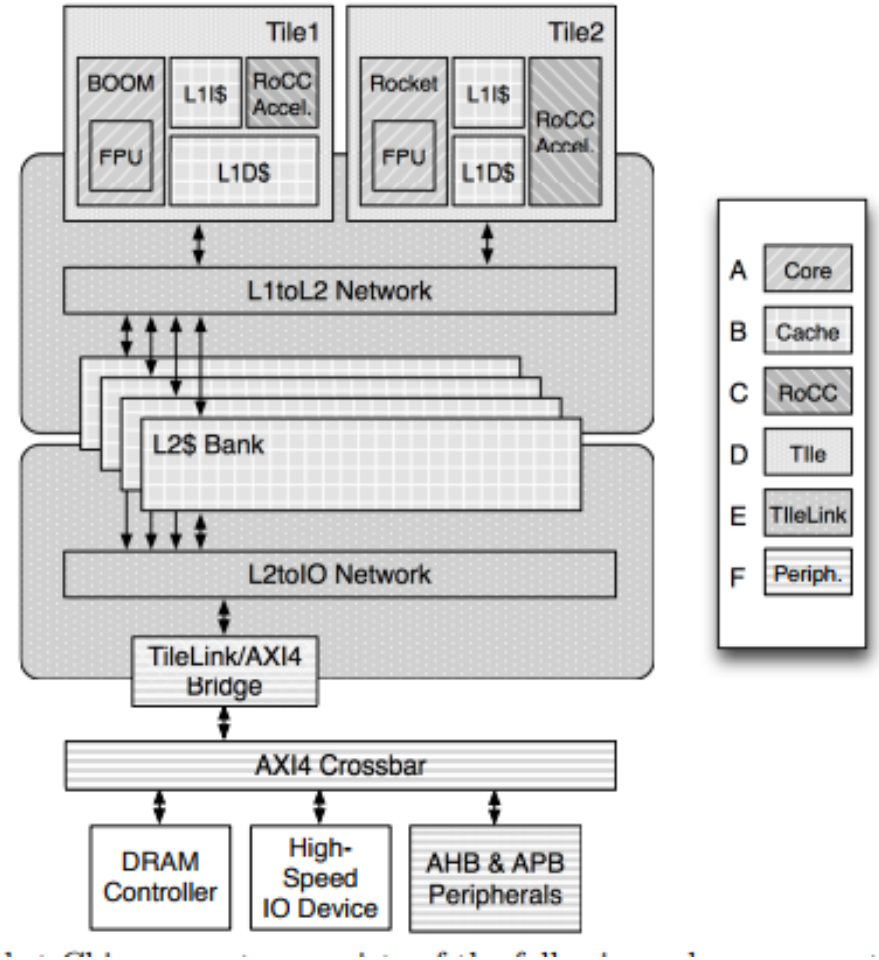
- RV32IMFC
- 4-stage in-order
- Faster CPI (~1)
- More complex than PicoRV32
  - Very tidy source though!



Rocket Chip Generator

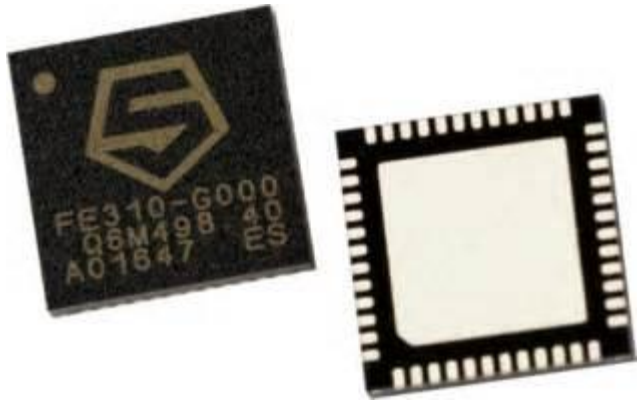
- Generates RV32/64
  - Various extensions
- Implementation: Scala / Chisel
  - Verilator, FPGA, VLSI
- High complexity





SiFive HiFive1 (Freedom E310 SoC)

- RV32IMAC
- Arduino IDE + form factor
- 16K RAM, 16MB flash, up to 320MHz



- Great for starting SW dev on real HW

- Costs $59 on Crowdsupply

Ecosystem drivers

- Free and open nature of the ISA

- Reduced risk and initial investment ($ and time)
  - No architecture licenses
  - Don’t *need* to license hardware designs
  - No charge for documentation

- Selling a commercial end-user RISC-V product:
  - Implementation must conform to the ISA
  - Join the RISC-V Foundation

Ecosystem challenges (1)

- Limited documentation, e.g.

RISC-V Assembly Programmer's Handbook

This chapter is a placeholder for an assembly programmer's manual.

(+3 pages listing registers and instruction names)

- Github repos / implementations are docs

Ecosystem challenges (2)

- Upstreaming in progress for:
  - Linux
  - Newlib
  - GDB
- Need to keep track of changes during development

Ecosystem challenges (3)

- ISA Test suite a little basic
- Our solution: use GCC regression suite to detect issues
- Long-term: Testing Working Group

Ecosystem future

- Stabilisation of toolchains
- Increased maturity of implementations
- Proliferation of hardware (my crystal ball at work here):
  - Short term: Embedded systems
    - Network hardware, GPUs, hard disks, etc. - many places there's already an ARM
  - Medium term: Low power / low speed general purpose computing
    - RISC-Vberry Pi? Phone / small computer OSes?
  - Long term: big iron?
    - Much more conservative system builders / integrators

Practical starting points

- Software only with instruction set simulator:
  - Official RISC-V toolchain: https://github.com/riscv/riscv-tools
- Software toolchain, with HiFive1 hardware:
  - Sifive Freedom E SDK: https://github.com/sifive/freedom-e-sdk
- Verilator cycle-accurate models of PicoRV32 and RI5CY:
  - https://github.com/embecosm/ri5cy and https://github.com/embecosm/picorv32
  - More details following my ORConf talk
- Rocket Chip:
  - https://github.com/freechipsproject/rocket-chip