RISC-V Update

Jonathan Neuschäfer
ecc 2017

What is RISC-V?

- New general purpose instruction set
- Open standard, no fees
- Lots of interest from the industry
- Hobbyist implementations possible

RISC-V banner © RISC-V Foundation, https://pbs.twimg.com/profile_banners/2694452875/1445472464/1500x500

Software Status

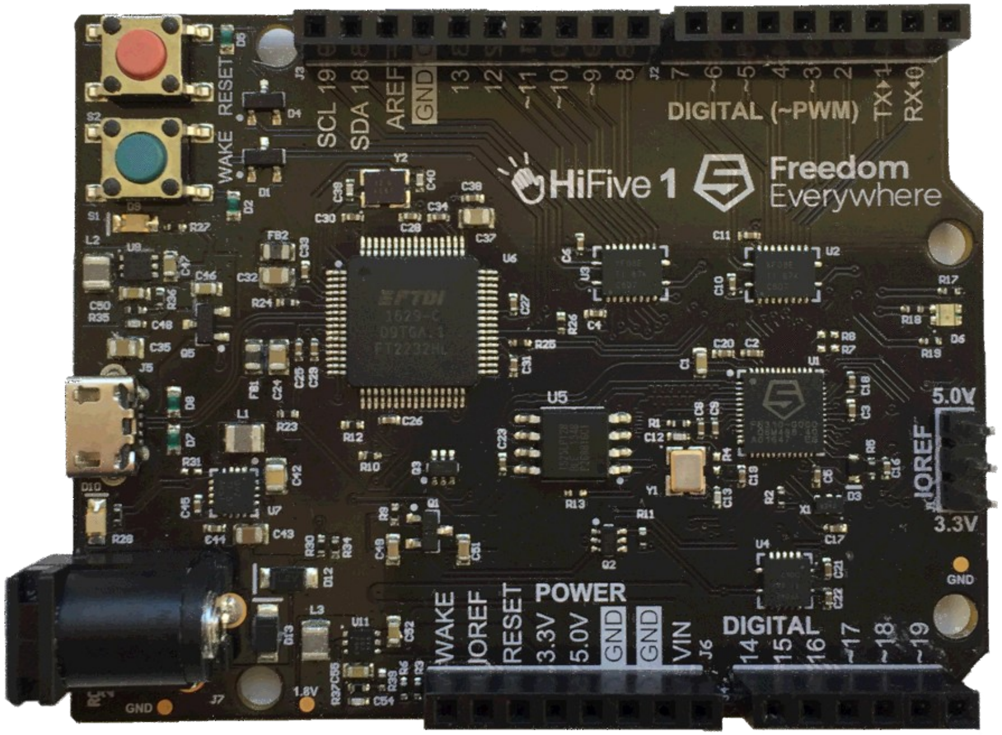
binutils/GCC	upstream
Linux kernel	soon to be upstream
glibc/musl	waiting for Linux
FreeBSD	upstream
LLVM	WIP, partially upstream
coreboot	upstream, needs some work :)

Virtual Platforms

- Spike
  - the golden reference
  - currently the main focus of coreboot/RISC-V
- QEMU
- RISCVEMU
  - lots of Virtio

Hardware: SiFive

- HiFive 1
  - microcontroller
  - non-trivial clock tree

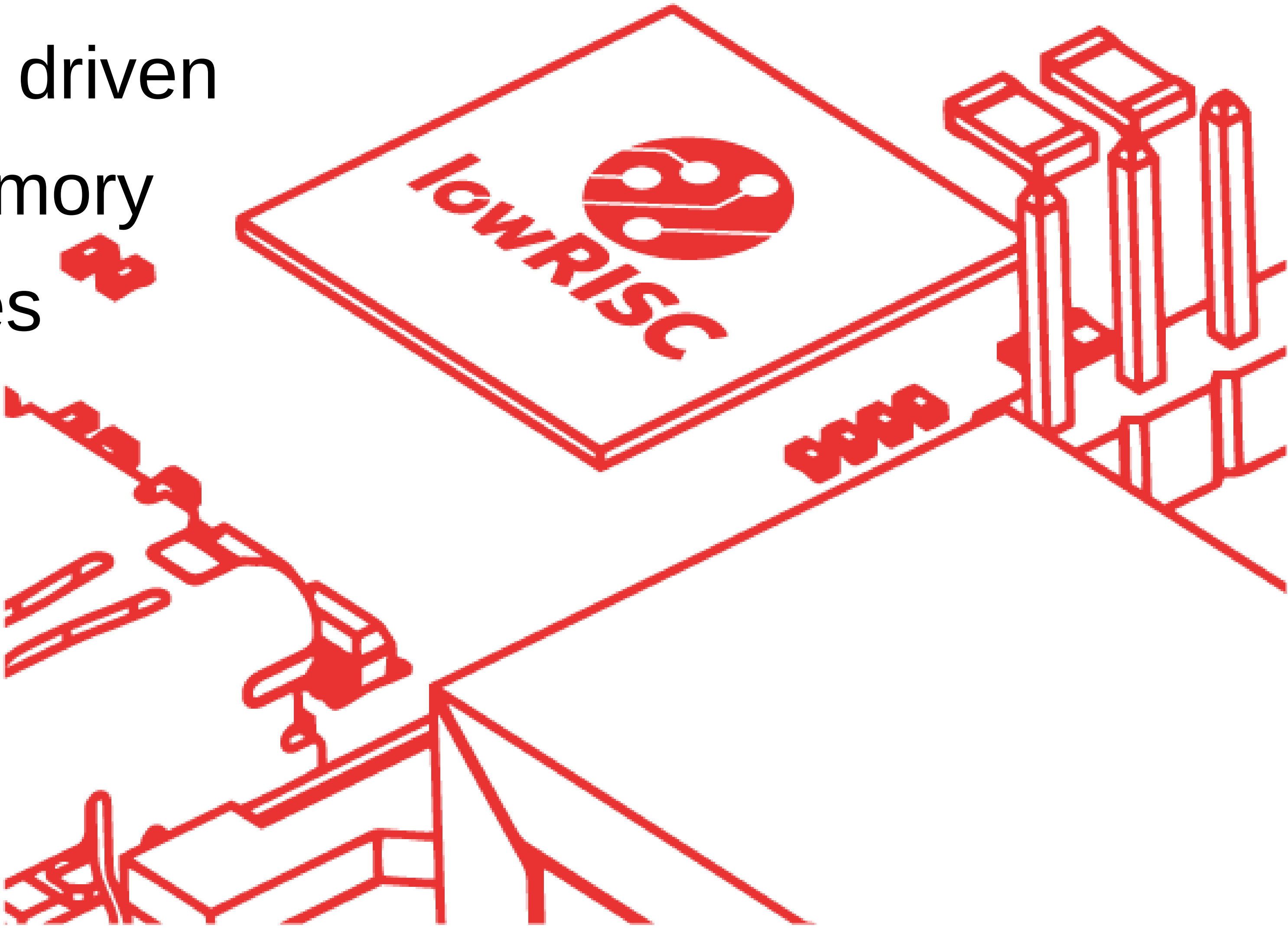


- U54-MC devboard announced for 2018 Q1
  - DDR memory, multicore
  - “RISC-V Meets Linux”

HiFive1 top view © SiFive Inc., https://www.sifive.com/assets/images/products/hifive1/hifive1-top-view.png

Hardware: lowRISC

- Currently runs on FPGA (Nexys4DDR)
- Community driven
- Tagged memory
- Minion cores



config string → devicetree

```
platform {
  vendor ucb;
  arch spike;
};
ram {
  0 {
    addr 0x80000000;
    size 0x40000000;
  };
};
rtc {
  addr 0x40000000;
};
core {
  0 {
    0 {
      isa rv64imafdc;
      timecmp 0x40000008;
      ipi 0x40002000;
    };
  };
};
```

```
/dts-v1/;
/ {
    compatible = "ucbbar,spike-bare-dev";
    memory@80000000 {
        device_type = "memory";
        reg = <0 0x80000000 0 0x40000000>;
    };
    soc {
        compatible = "ucbbar,spike-bare-soc", "simple-bus";
        ranges;
        clint@2000000 {
            compatible = "riscv,clint0";
            interrupts-extended = <&intc 3 &intc 7>;
            reg = <0 0x2000000 0 0xc0000>;
        };
    };
    cpus {
        cpu@0 {
            compatible = "riscv";
            riscv,isa = "rv64imafdc";
            clock-frequency = <1000000000>;
            intc: interrupt-controller {
                #interrupt-cells = <1>;
                interrupt-controller;
                compatible = "riscv,cpu-intc";
            };
        };
    };
};
```

Supervisor Binary Interface

```
sbi_putc = -2000

li   a0,   'H'
jalr zero, sbi_putc


li   a0,   'i'
jalr zero, sbi_putc
```

```
SBI_PUTC = 1

li   a0, 'H'
li   a7, SBI_PUTC
ecall

li   a0, 'i'
li   a7, SBI_PUTC
ecall
```

Future Work

- `COLLECT_TIMESTAMPS`
  - other architectures have it, and it’s a cool feature
- RISC-V Physical Memory Protection
  - prevent supervisor from overwriting our code
- optional resident code and SBI
  - Linux (currently) needs it
  - but I think Ron is right[1]

[1]) cf. “Let’s move SMM out of firmware and into the kernel”, Ron Minnich, ECC2017

That’s it!

- Thank you for listening
- Questions?