SiFive

Igniting the Open Hardware Ecosystem with RISC-V

FOSDEM, February 2018

Palmer Dabbelt, SiFive

Yunsup Lee, SiFive

RISC-V is a free and open ISA standard designed for all computing devices

RISC-V is a free and open ISA standard designed for all computing devices

RISC-V binutils, GCC, Linux, and glibc have all been released by upstream as of February 1, 2018

RISC-V is a free and open ISA standard designed for all computing devices

RISC-V binutils, GCC, Linux, and glibc have all been released by upstream as of February 1, 2018

It is now time to start porting your favorite software project to RISC-V

RISC-V is a free and open ISA standard designed for all computing devices

RISC-V binutils, GCC, Linux, and glibc have all been released by upstream as of February 1, 2018

It is now time to start porting your favorite software project to RISC-V

Join the RISC-V revolution!

Why Instruction Set Architecture matters

- Why can't Intel sell mobile chips?
  - 99%+ of mobile phones/tablets are based on ARM's v7/v8 ISA
- Why can't ARM partners sell servers?
  - 99%+ of laptops/desktops/servers are based on the AMD64 ISA (over 95%+ built by Intel)
- How can IBM still sell mainframes?
  - IBM 360 is the oldest surviving ISA (50+ years)

ISA is the most important interface in a computer system

ISA is where software meets hardware

Open Software/Standards Work!

Field	*Standard*	*Free, Open Impl.*	*Proprietary Impl.*
Networking	Ethernet, TCP/IP	Many	Many
OS	Posix	Linux, FreeBSD	M/S Windows
Compilers	C	gcc, LLVM	Intel icc, ARMcc
Databases	SQL	MySQL, PostgresSQL	Oracle 12C, M/S DB2
Graphics	OpenGL	Mesa3D	M/S DirectX
ISA	??????	-----------	x86, ARM, IBM360

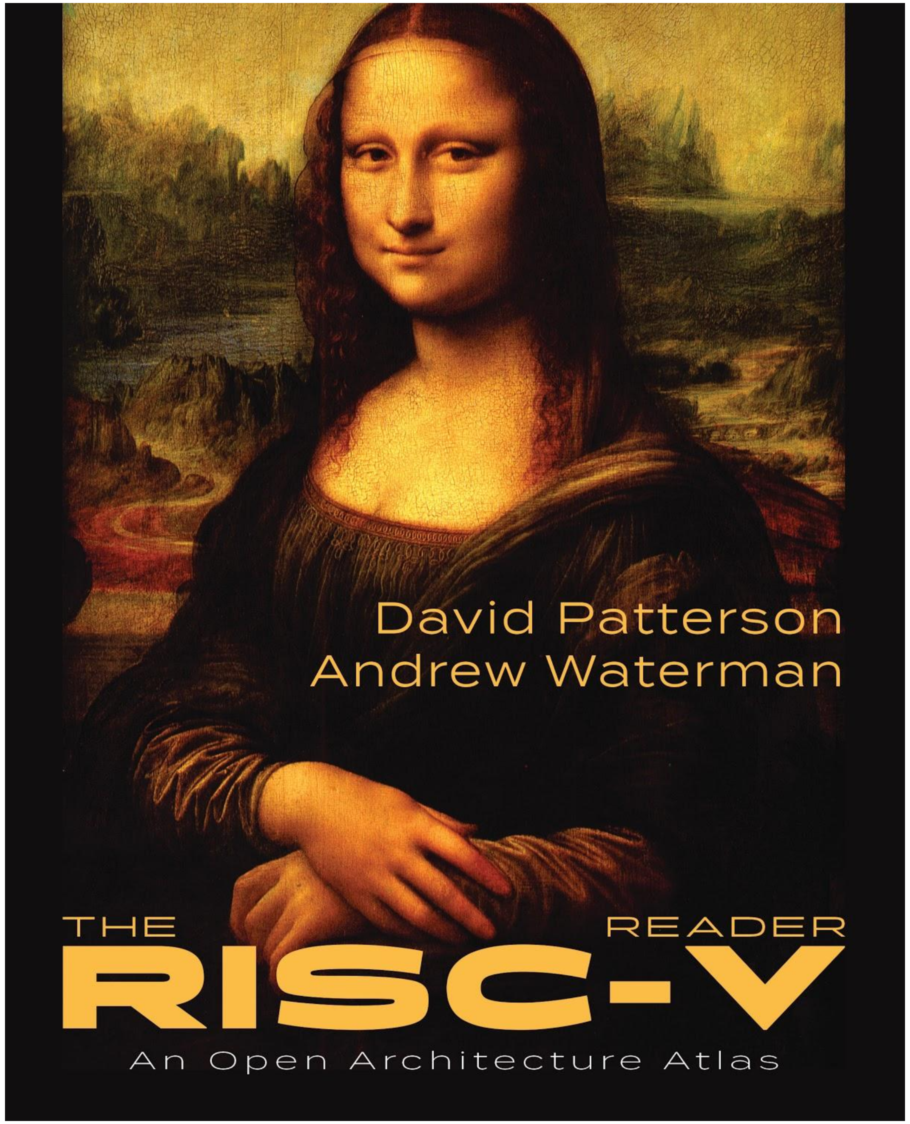
- Why not have successful free & open standards and free & open implementations, like other fields?
- Dominant proprietary ISAs are not great designs

What is RISC-V?

- A high-quality, license-free, royalty-free RISC ISA specification originally designed at UC Berkeley
- Standard maintained by the non-profit RISC-V Foundation
- Suitable for all types of computing system, from microcontrollers to supercomputers
- Numerous proprietary and open-source cores
- Experiencing rapid uptake in industry and academia
- Supported by a growing shared software ecosystem
- A work in progress…

RISC-V Reader Giveaway!

- Authored by Andrew and Dave
  - Andrew Waterman: SiFive co-founder and co-inventor of the RISC-V ISA
  - Dave Patterson: UC Berkeley professor, co-author of "Computer Organization and Design", and co-inventor of RISC-V
- "An Open Architecture Atlas"
  - Concise introduction and reference
  - Aimed at embedded systems programmers, students, and the curious
- Tweet us a photo of talk
  - #HiFiveUnveiled and @SiFiveInc
- Winners selected during the talk



SiFive

Origin of RISC-V

FOSDEM “Igniting the Open Hardware Ecosystem with RISC-V”
February 2018

RISC-V Origins

- In 2010, after many years and many projects using MIPS, SPARC, and x86 as the bases of research at Berkeley, it was time to choose an ISA for next set of projects
- Obvious choices: x86 and ARM

Intel x86 “AAA” Instruction

- ASCII Adjust After Addition
- AL register is default source and destination

- If the low nibble is > 9 decimal, or the auxiliary carry flag AF = 1, then
  - Add 6 to low nibble of AL and discard overflow
  - Increment high byte of AL
  - Set CF and AF
- Else
  - CF = AF = 0

- Single byte instruction

ARM v7 LDMIAEQ Instruction

```
LDMIAEQ SP!, {R4-R7, PC}
```

- LoaD Multiple, Increment-Address
- Writes to 7 registers from 6 loads
- Only executes if EQ condition code is set
- Writes to the PC (a conditional branch)
- Can change instruction sets

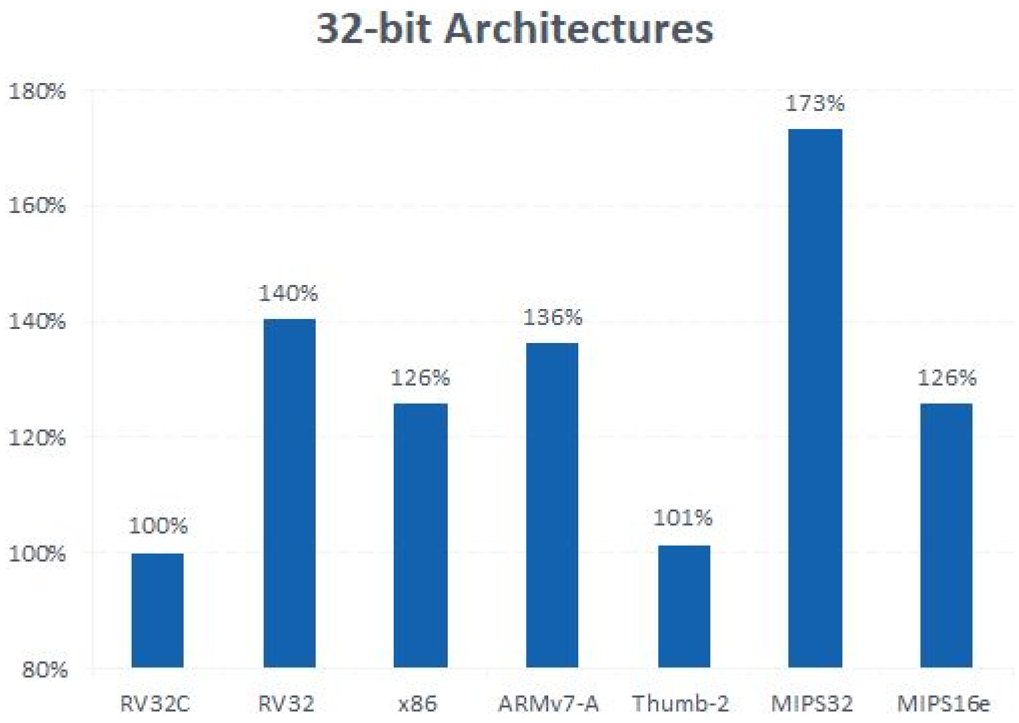
- Idiom for "stack pop and return from a function call"

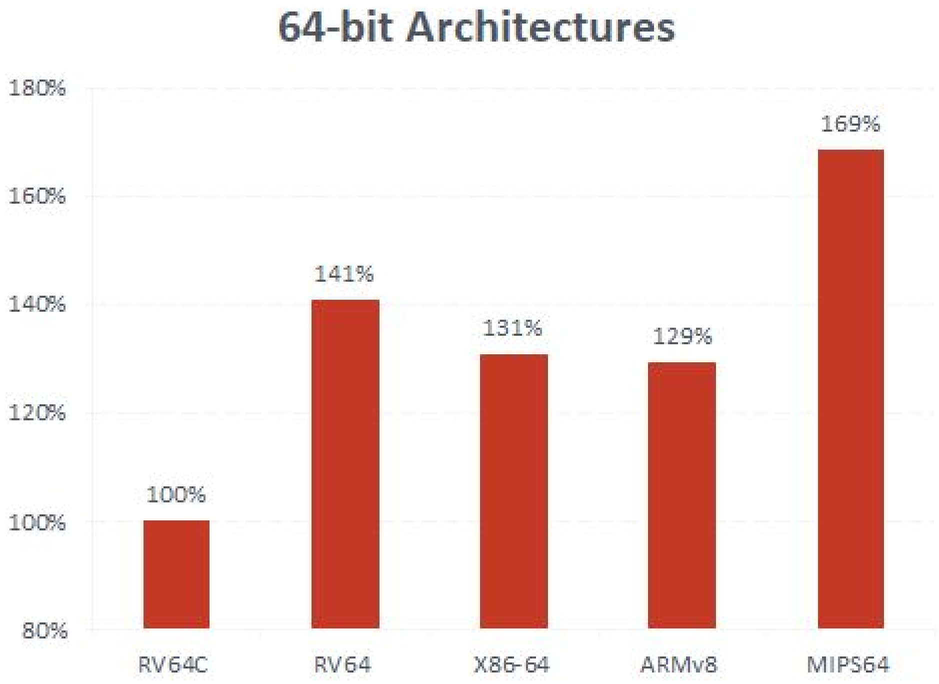
RISC-V Origin Story

- x86 impossible – IP issues, too complex
- ARM mostly impossible – no 64-bit, IP issues, complex
- So we started “3-month project” in summer 2010 to develop our own clean-slate ISA
  - Principal designers: Andrew Waterman, Yunsup Lee, Dave Patterson, Krste Asanovic
- Four years later, we released the frozen base user spec
  - First public specification released in May 2011
  - Several publications, many tapeouts, lots of software along the way

ISA Effort: Static Code Size

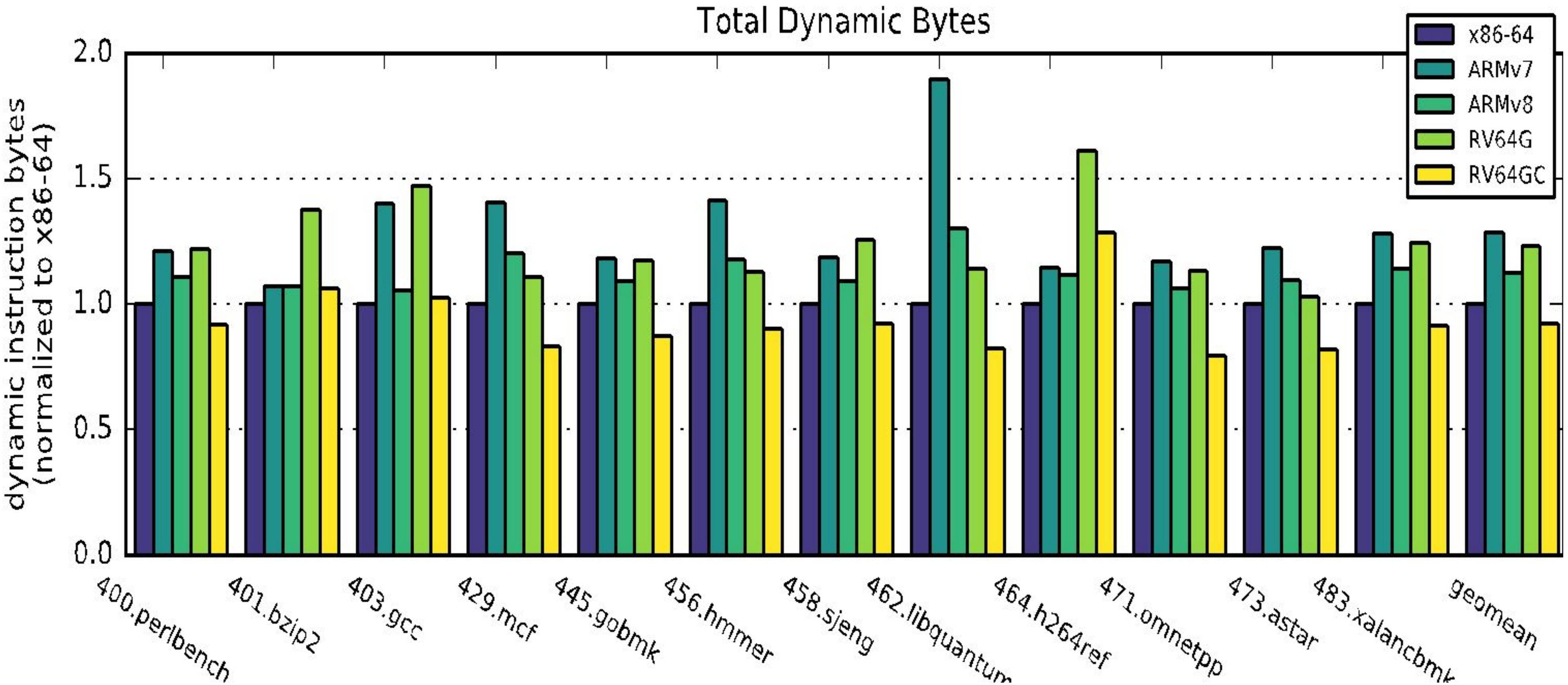
- RISC-V is now the smallest ISA for 32- and 64-bit addresses
- All results are with the same GCC compiler and options



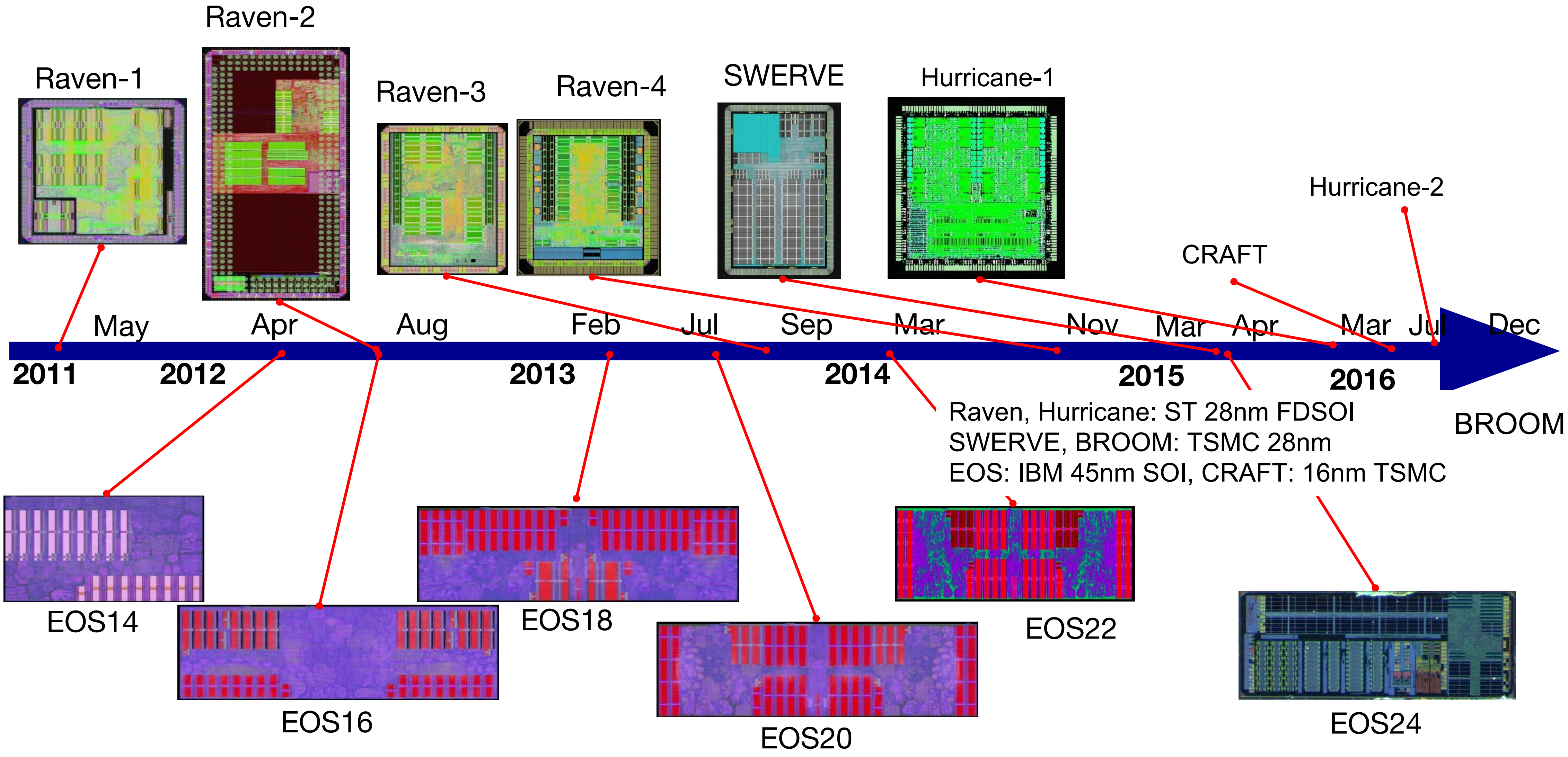


ISA Effort: Dynamic Bytes Fetched

- RV64GC is lowest overall in dynamic bytes fetched

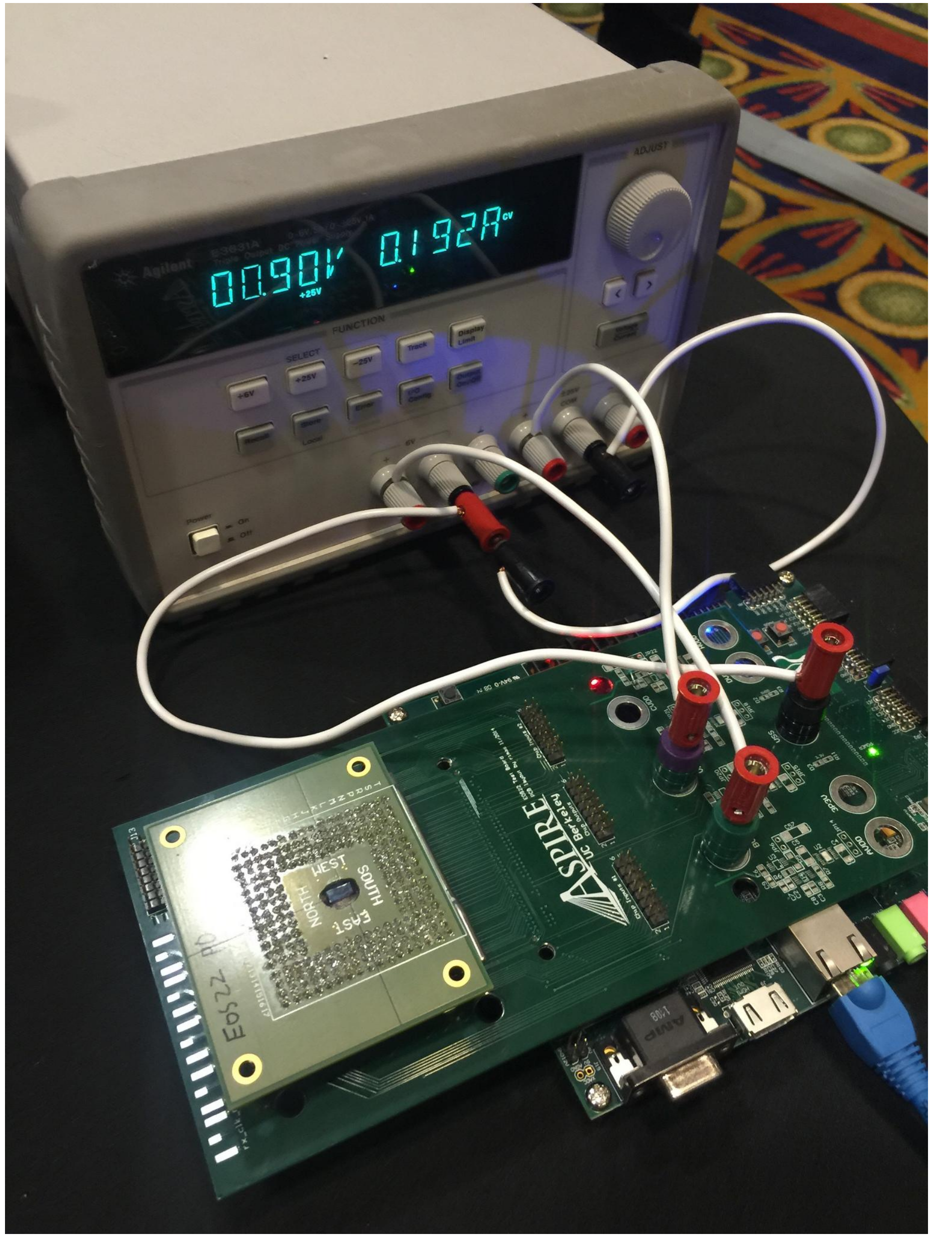


Chip Tapeout Effort: Built 10+ RISC-V Chips



Raven, Hurricane: ST 28nm FDSOI
SWERVE, BROOM: TSMC 28nm
EOS: IBM 45nm SOI, CRAFT: 16nm TSMC

Chip Tapeout Effort: DIY



Software Effort: Started from Berkeley

Software Effort: There’s a Lot of Software

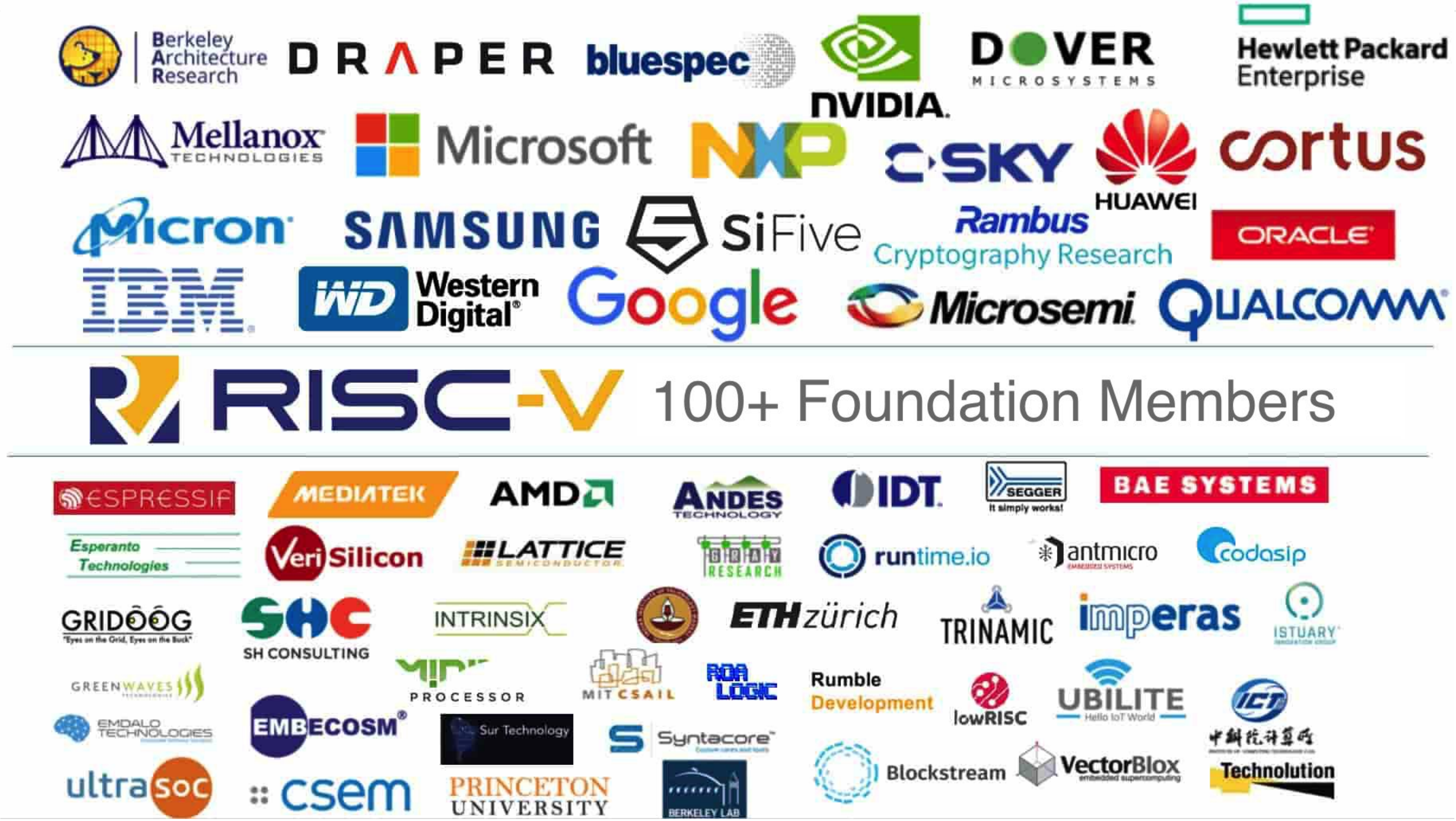
```
From: Palmer Dabbelt <palmer.dabbelt@eecs.berkeley.edu>
To: config-patches@gnu.org
Subject: config.sub patch for RISC-V
Date: Wed, 10 Sep 2014 19:20:31 -0700
Message-Id: <1410402032-9184-1-git-send-email-palmer.dabbelt@eecs.berkeley.edu>
X-Mailer: git-send-email 1.8.5.5

This patch provides support for the RISC-V ISA: http://riscv.org/

Not yet upstreamed ports of the binutils, GCC, LLVM, glibc, and Linux
exist for RISC-V, and a number of hardware implementations exist --
more more information can be seen at http://riscv.org .  We'd like to
start getting RISC-V recognized by configure so it's easier for people
to start porting stuff.

Thanks!
```

SiFive

Current State of RISC-V

FOSDEM “Igniting the Open Hardware Ecosystem with RISC-V”
February 2018

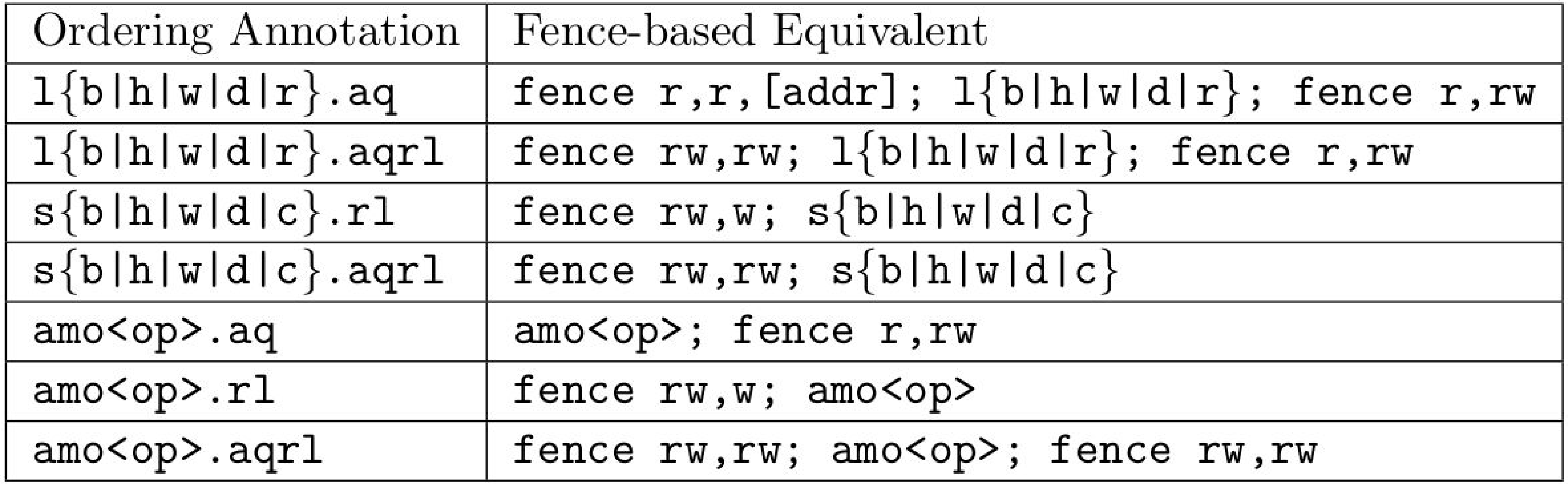
RISC-V Foundation



RISC-V Specifications

- User Mode ISA Specification
  - RV32I/RV64I: ALU, branches, and memory
  - M extension for multiplication
  - A extension for atomics
  - F and D extensions for single and double precision floating-point
  - C extension for compressed instructions (16-bit)
- Privileged Mode ISA Specification
  - Supervisor mode
  - Hypervisor mode
  - Machine mode
- External Debug Specification
  - Debug machine-mode software over JTAG

RISC-V Weak Memory Model



Ordering Annotation	Fence-based Equivalent
l{b|h|w|d|r}.aq	fence r,r,[addr]; l{b|h|w|d|r}; fence r,rw
l{b|h|w|d|r}.aqrl	fence rw,rw; l{b|h|w|d|r}; fence r,rw
s{b|h|w|d|c}.rl	fence rw,w; s{b|h|w|d|c}
s{b|h|w|d|c}.aqrl	fence rw,rw; s{b|h|w|d|c}
amo<op>.aq	amo<op>; fence r,rw
amo<op>.rl	fence rw,w; amo<op>
amo<op>.aqrl	fence rw,rw; amo<op>; fence rw,rw

Table 1.3: Mappings from .aq and/or .rl to fence-based equivalents

Upstream!

- binutils is upstream
  - released in 2.28
  - 2.30 is in good shape
- GCC is upstream
  - released in 7.1.0
  - 7.3.0 is in good shape
- Linux is upstream
  - released in 4.15
  - missing device drivers
- glibc is upstream
  - released in 2.27
  - missing RV32I support

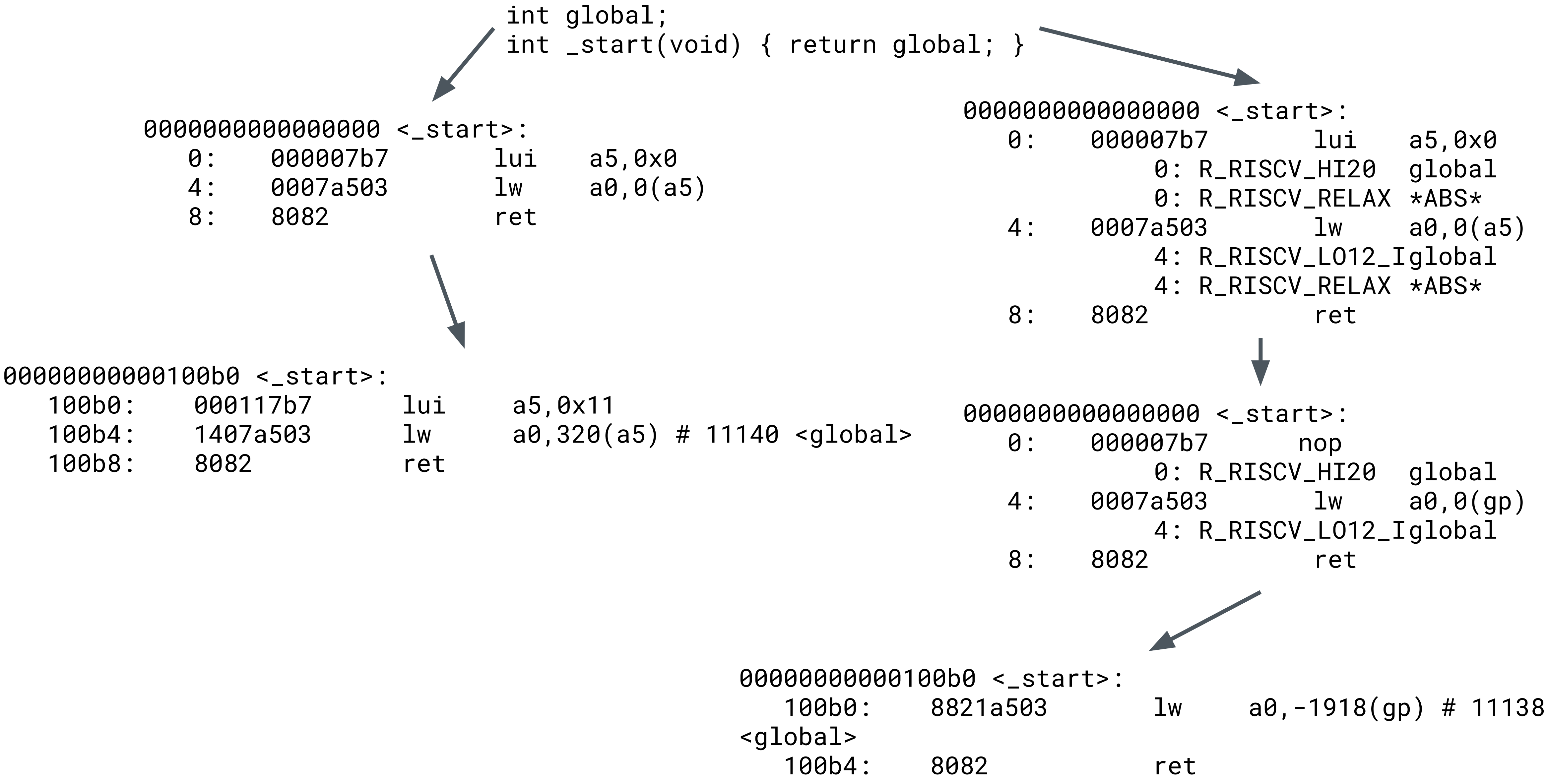
The RISC-V Software Porting Effort

- Kito Cheng (Andes Technology): GCC and newlib
- Jim Wilson (SiFive): binutils and GCC
- Darius Rad (Bluespec): glibc
- Andrew Waterman (SiFive): binutils, GCC, and glibc
- Albert Ou (UC Berkeley): Linux
- Michael Clark (SiFive): QEMU
- DJ Delorie (RedHat): glibc

Future of RISC-V

FOSDEM “Igniting the Open Hardware Ecosystem with RISC-V”
February 2018

Linear-Time Linker Relaxation



```
int global;
int _start(void) { return global; }
```

```
0000000000000000 <_start>:
   0:    000007b7          lui    a5,0x0
   4:    0007a503          lw     a0,0(a5)
   8:    8082              ret
```

```
00000000000100b0 <_start>:
   100b0:    000117b7      lui     a5,0x11
   100b4:    1407a503      lw      a0,320(a5) # 11140 <global>
   100b8:    8082          ret
```

```
0000000000000000 <_start>:
   0:    000007b7          lui    a5,0x0
                 0: R_RISCV_HI20  global
                 0: R_RISCV_RELAX *ABS*
   4:    0007a503          lw     a0,0(a5)
                 4: R_RISCV_LO12_Iglobal
                 4: R_RISCV_RELAX *ABS*
   8:    8082              ret
```

```
0000000000000000 <_start>:
   0:    000007b7        nop
                 0: R_RISCV_HI20  global
   4:    0007a503          lw     a0,0(gp)
                 4: R_RISCV_LO12_Iglobal
   8:    8082              ret
```

```
00000000000100b0 <_start>:
   100b0:    8821a503          lw     a0,-1918(gp) # 11138
<global>
   100b4:    8082              ret
```

Handling auipc Efficiently

```
volatile int global;
int _start(void) { return global + global; }
```

```
00000000000100b0 <_start>:
   100b0:    67c9          lui  a5,0x12
   100b2:    0c07a503      lw   a0,192(a5) # 120c0 <global>
   100b6:    0c07a783      lw   a5,192(a5)
   100ba:    9d3d          addw a0,a0,a5
   100bc:    8082          ret
```

```
00000000000100b0 <_start>:
   100b0:    00002797      auipc a5,0x2
   100b4:    0187a503      lw    a0,24(a5) # 120c8 <global>
   100b8:    00002797      auipc a5,0x2
   100bc:    0107a783      lw    a5,16(a5) # 120c8 <global>
   100c0:    9d3d          addw  a0,a0,a5
   100c2:    8082          ret
```

How to Contribute to RISC-V Software

- RISC-V GitHub Organization
  - https://github.com/riscv
- RISC-V Mailing Lists
  - sw-dev@groups.riscv.org
- Upstream!
  - binutils@lists.sourceware.org
  - gcc-patches@lists.gnu.org
  - linux-riscv@lists.infradead.org
  - libc-alpha@lists.sourceware.org

The Future of RISC-V: Linux Distributions

- Debian
  - Bootstrap in progress
- Fedora
  - Bootstrap in progress
- OpenEmbedded
  - Some toolchain support has landed
- OpenWRT
  - Distro maintainer
- Gentoo
  - We have an ARCH name but no maintainer

The Future of RISC-V: New Specifications

- V extension for vectors
  - Cray-style vectors, updated for the modern world
  - Targeted for both temporal and spatial implementations
  - Initial presentation at RISC-V workshop in December 2017
  - Draft expected by end of the year
- J extension for JITs
  - Primary target is advanced JVMs
  - Working group is being founded
- Unix platform specification
  - Interface between the firmware and kernel
  - Required system profile
  - Working group not yet founded

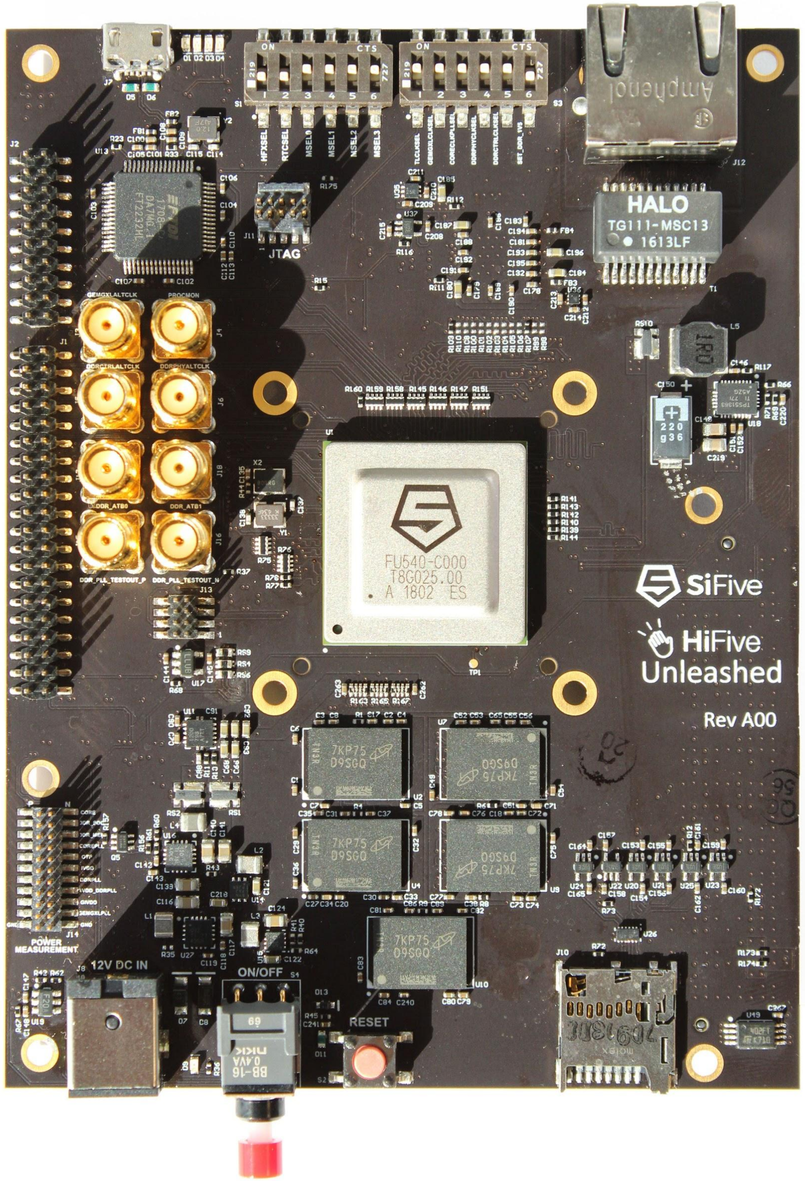
Future of RISC-V: Linux-Capable Silicon

We need Linux-capable silicon to push RISC-V software to the next level

The Future is Here

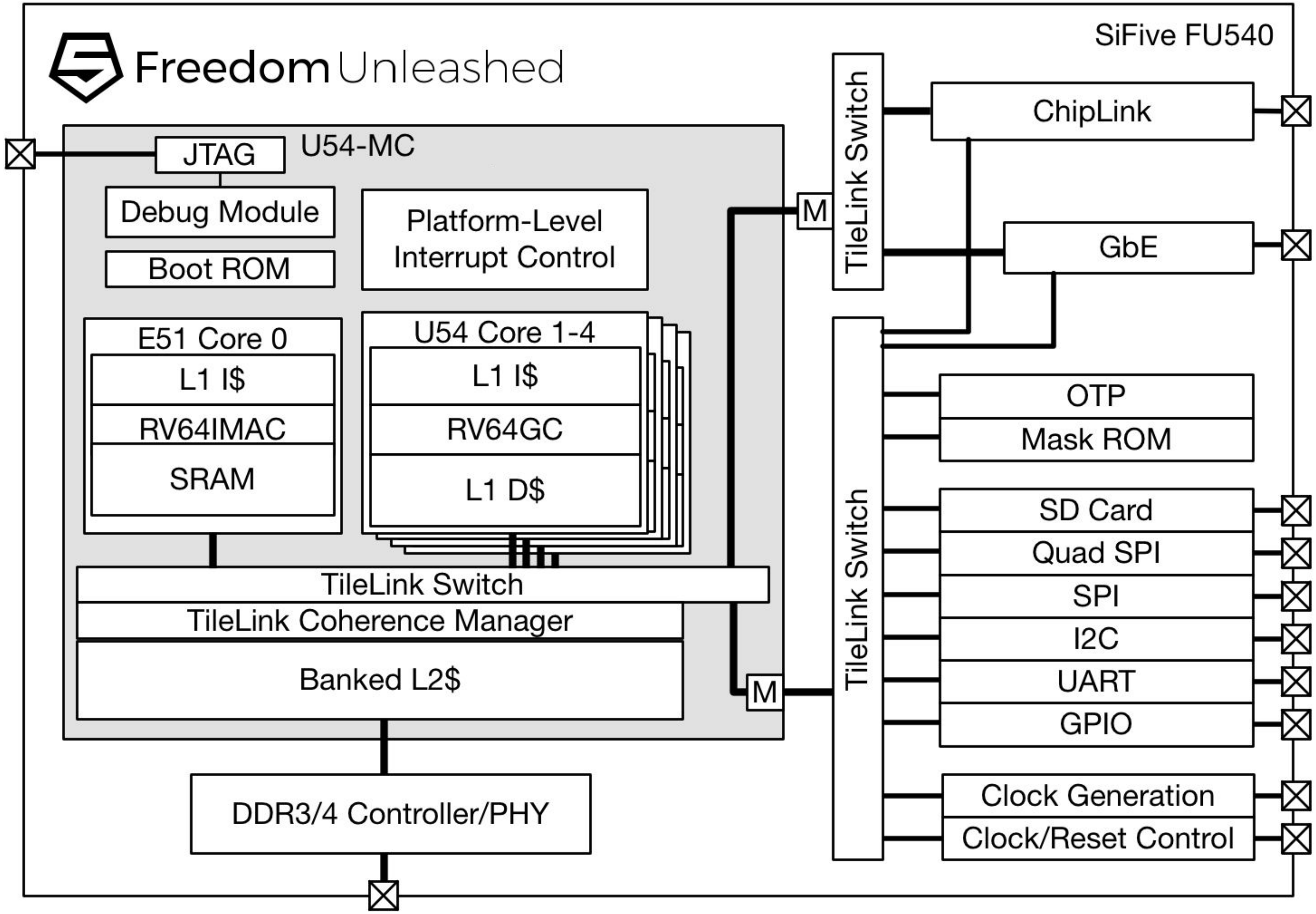
FOSDEM “Igniting the Open Hardware Ecosystem with RISC-V”
February 2018

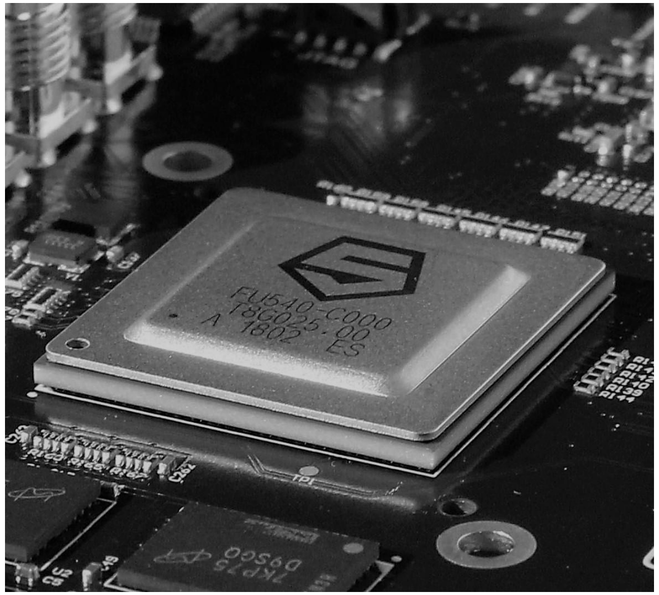
HiFive Unleashed



- World’s First Multi-Core RISC-V Linux Development Board
  - SiFive FU540-C000 (built in 28nm)
    - 4+1 Multi-Core Coherent Configuration, up to 1.5 GHz
    - 4x U54 RV64GC Application Cores with Sv39 Virtual Memory Support
    - 1x E51 RV64IMAC Management Core
    - Coherent 2MB L2 Cache
    - 64-bit DDR4 with ECC
    - 1x Gigabit Ethernet
  - 8 GB 64-bit DDR4 with ECC
  - Gigabit Ethernet Port
  - 32 MB Quad SPI Flash
  - MicroSD card for removable storage
  - FMC connector for future expansion with add-in cards

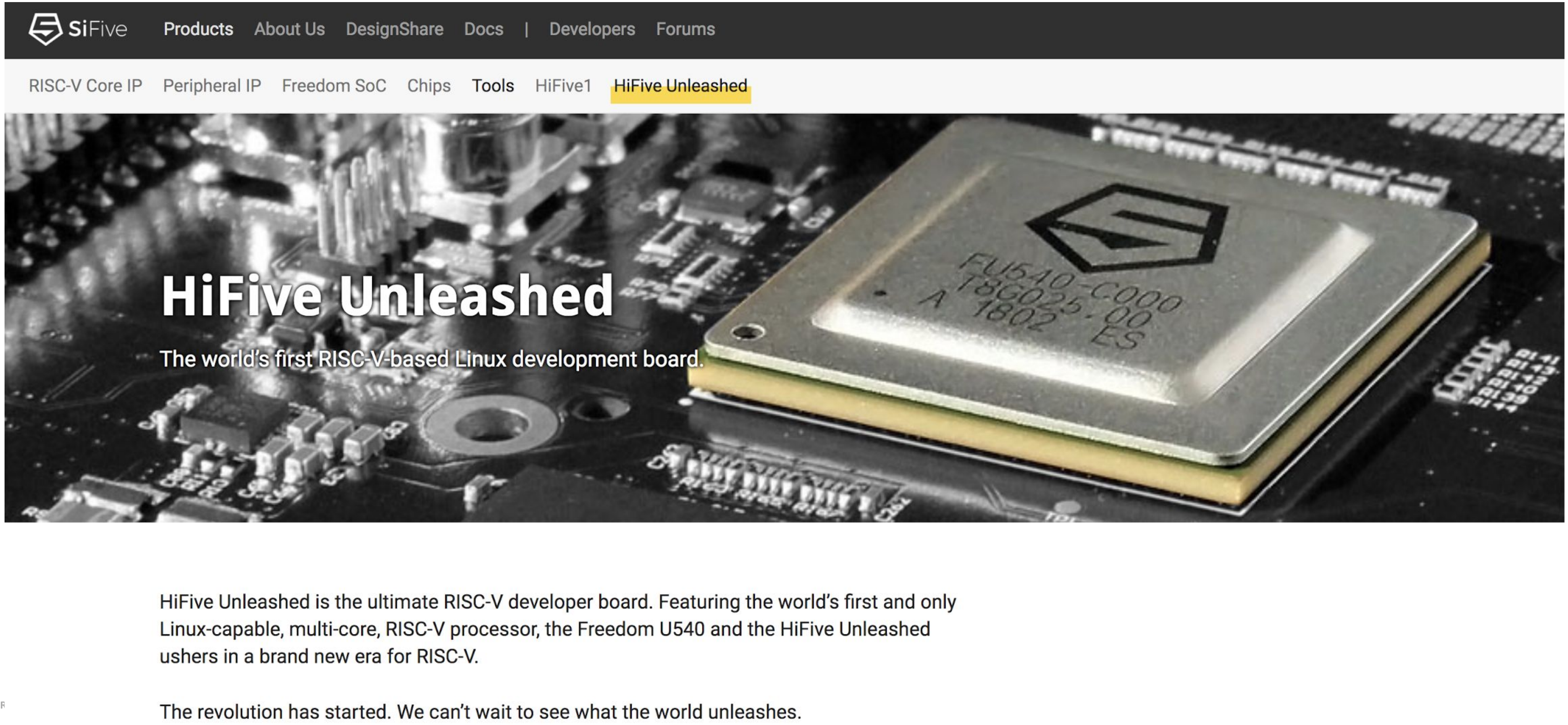
FU540: Penta-Core 64-bit RISC-V Linux SoC





Buy it Now!

https://www.sifive.com/products/hifive-unleashed/



HiFive Unleashed

The world's first RISC-V-based Linux development board.

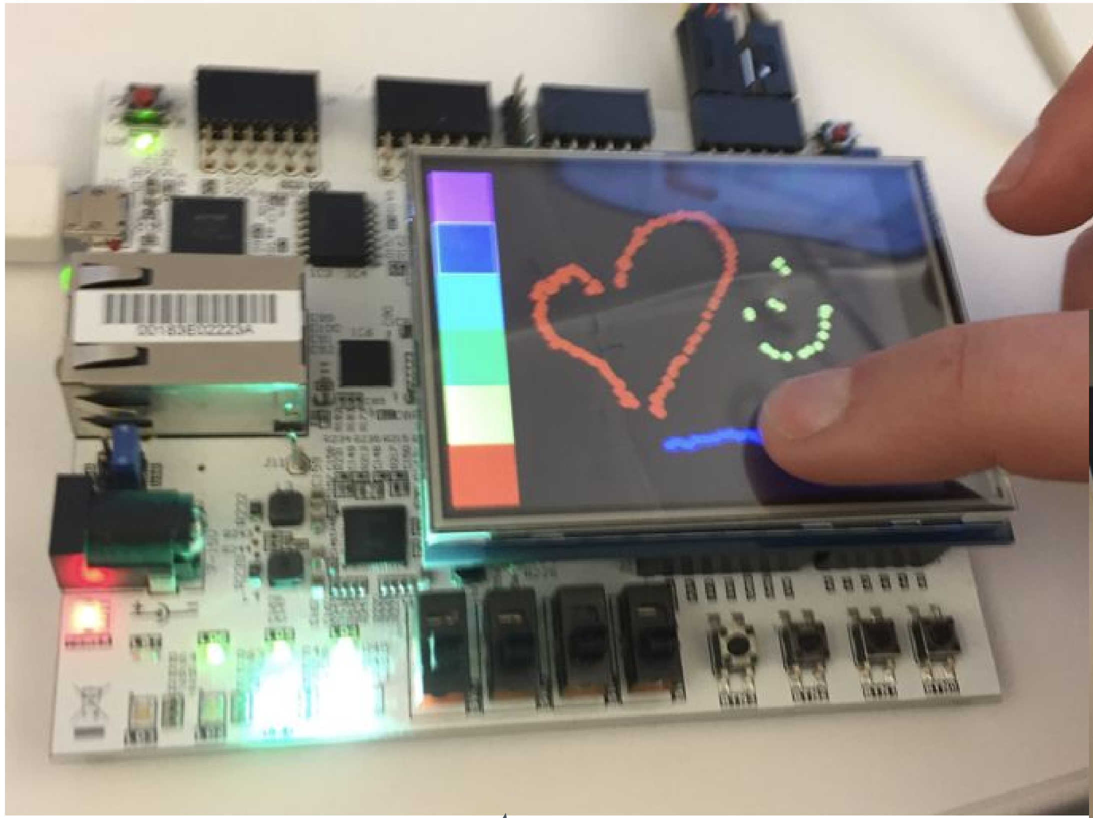
HiFive Unleashed is the ultimate RISC-V developer board. Featuring the world's first and only Linux-capable, multi-core, RISC-V processor, the Freedom U540 and the HiFive Unleashed ushers in a brand new era for RISC-V.

The revolution has started. We can't wait to see what the world unleashes.

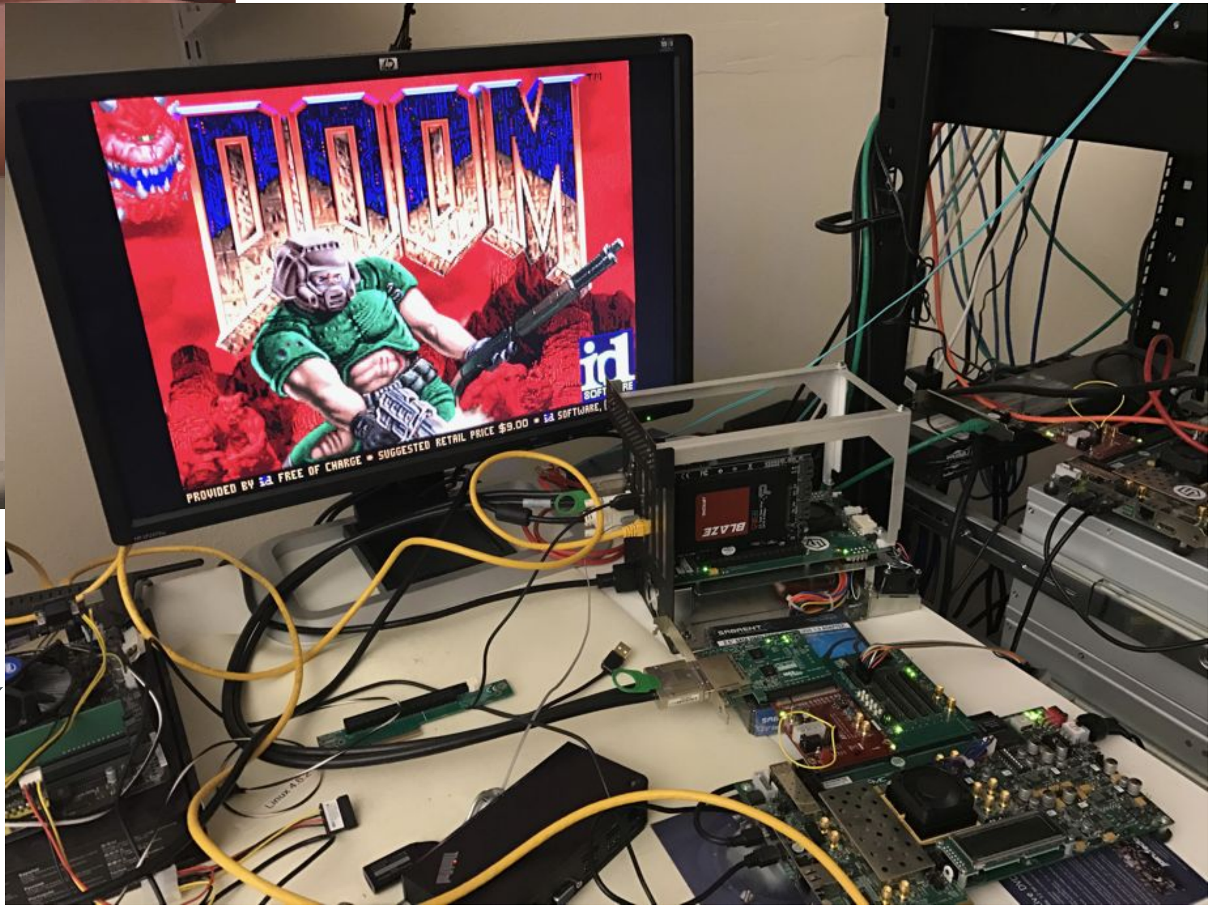
We Open-Sourced the Freedom Platform!

- Freedom Platform is an open-source RISC-V-based SoC platform maintained by SiFive, consisting of:
  - RISC-V Rocket CPU
  - TileLink, a free and open coherent SoC interconnect
  - Low-speed Peripherals: SPI, UART, PWM, GPIO, I2C
  - High-speed Xilinx FPGA Peripheral Wrappers: DDR, PCIe blocks
  - L2$ (will be open-sourced with the HiFive Unleashed Launch)
- Freedom U540 chip is based on the Freedom Unleashed platform
  - Alas, we can't open-source 3rd-party IP: cells, pads, PLL, OTP, DDR, GbE, ROM
  - We'd love to work together to build a completely open chip!
- Check out
  - https://github.com/sifive/freedom
  - https://dev.sifive.com

Freedom FPGA Dev Kits



Freedom E300 Arty FPGA Dev Kit


DOOM
PROVIDED BY id FREE OF CHARGE • SUGGESTED RETAIL PRICE $9.00 • id SOFTWARE,
BLAZE

Freedom U500 VC707 FPGA Dev Kit

Why Open-Source the Freedom Platform?

- Open-source has revolutionized SW: Now it's hardware's turn
- Open-source platform allows for more innovation, promotes reuse, and attracts developers
  - Developers and IP providers can focus on their value-added innovation
  - Leverage the collective effort of the community and industry
- Enables both open-source developers and for-profit IP companies to work with Freedom Platform
- Makes it easier for system designers to work with SiFive to customize their chip and software

Join the RISC-V Revolution!

- RISC-V software ecosystem has been growing rapidly thanks to your help
- RISC-V hardware is here
- Start your favorite software project with RISC-V!

- Buy your own HiFive Unleashed dev board at https://www.sifive.com/products/hifive-unleashed/
- Sign up at https://dev.sifive.com for updates

- Demo
- Q&A, RISC-V BoF 6-7pm @ Room J1.106

SiFive

End

FOSDEM “Igniting the Open Hardware Ecosystem with RISC-V”
February 2018