Building the RISC-V Software Ecosystem

Arun Thomas, BAE Systems
3rd RISC-V Workshop, January 2016
Where I'm Coming From

• On the hardware side
  
  • Cut teeth on OpenCores, OpenRISC, OpenSPARC
  
  • Designed an ISA that fourth-year CpE students were forced to implement
  
  • Worked on DARPA CRASH/SAFE project
  
  • Now, working on RISC-V security (tagging) extensions
Where I'm Coming From

- On the software side
  - Gentoo Linux developer
  - MINIX 3 developer
  - Low-level kernel hacking on x86/amd64 and ARMv7/v8
  - Now, FreeBSD/RISC-V
2016 is the Year of RISC-V
2016 is the Year of RISC-V Software
Current Software Landscape

• Several OS ports in progress
  • Linux (Yocto/OpenEmbedded, Gentoo), FreeBSD, NetBSD, seL4
• Support primary open source toolchains
  • Binutils, GCC, clang/LLVM
• Multiple software simulators
  • Spike, QEMU, Angel
Landscape as of 12/31/16 (With Your Help)

• Upstreamed GNU toolchain and QEMU
• More mature clang/LLVM support
• Upstreamed OS support
• Debian/RISC-V port
• Start thinking about Android and a real-time OS
How do we get there?
How We Get There (My Take)

• Recruit more RISC-V developers
• Reduce startup costs for new developers
• More docs, more specs
We need more people writing RISC-V software.
# Who is Contributing Now?

48 contributors to RISC-V GitHub from:

<table>
<thead>
<tr>
<th>Universities</th>
<th>Companies</th>
<th>OSS Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>University of California, Berkeley</td>
<td>Bluespec</td>
<td>Gentoo</td>
</tr>
<tr>
<td>University of Cambridge</td>
<td>Google</td>
<td>Debian</td>
</tr>
<tr>
<td>Texas Tech University</td>
<td>LG Electronics</td>
<td>FreeBSD</td>
</tr>
<tr>
<td>ETH Zurich</td>
<td>BAE Systems</td>
<td>NetBSD</td>
</tr>
<tr>
<td>Cornell University</td>
<td>SRI</td>
<td></td>
</tr>
<tr>
<td>North Carolina State University</td>
<td></td>
<td></td>
</tr>
<tr>
<td>University of Erlangen-Nuremberg</td>
<td>VectorBlox</td>
<td></td>
</tr>
</tbody>
</table>

Also, many individual contributors
Attracting Developers

• Present talks/tutorials at developer conferences and local user group meetings
  • Potential targets: OSCon, Linuxcon, BSD events, LLVM Developer Summit
  • My efforts: EuroBSDCon, BLU, ESC (if accepted)
• Encourage current RISC-V users to submit pull requests
  • Including PRs for in-house patches
How many of you have sent a GitHub pull request?
My First Pull Request

Update riscv.ac to set CPPFLAGS with fesvr include path #2

arunthomas commented on Sep 20, 2014

We need to set CPPFLAGS in riscv.ac in addition to configure, since configure is a generated file. This is a followup to commit 4d1c63e by @ubclrl.

This fixes the build on Ubuntu and Mac for me.

arunthomas referenced this pull request on Sep 20, 2014

Include path for fesvr not added to compiler command line #1

sbeamer commented on Sep 21, 2014

Thanka!

This was probably caused by 44793fd. CPPFLAGS is definitely the cleanest way to fix this. Should close issue #1.

sbeamer merged commit eb27f3e into riscv:master on Sep 21, 2014

Pull request successfully merged and closed
You're all set—the arunthomas:build_fix branch can be safely deleted.
Attracting Developers

• Fund developers/projects via the Foundation
  • Possible models: Linaro, Linux Foundation, FreeBSD Foundation

• Apply to be a Google Summer of Code 2016 mentoring organization

• Update the list of open bugs and feature requests in GitHub

• Track contribution statistics (See Linux Kernel survey)
Reducing new developer startup costs
Setting up ARMv8

• Install cross-toolchain and QEMU

$ sudo apt-get install gcc-aarch64-linux-gnu qemu-system-arm qemu-user-static

• Download OpenEmbedded LAMP stack VM image from Linaro

$ wget http://releases.linaro.org/openembedded/aarch64/latest/vexpress64-openembedded_lamp-armv8-gcc-5.2_20151120-735.img.gz

• Boot OpenEmbedded in QEMU (system mode)

$ qemu-system-aarch64 ... -drive if=none,id=image,file=vexpress64-openembedded_lamp-armv8-gcc-5.2_20151120-735.img

• Cross-build and run ARMv8 binary in QEMU (user mode)

$ aarch64-linux-unknown-gnu-gcc -static -o hello hello.c && ./hello
Improving RISC-V Setup

- Debian packages for toolchain and QEMU
- Upstream everything: Toolchain, kernel, QEMU, OpenEmbedded, Gentoo, various packages
- Regular snapshots of OS images
- Near term: Switch GCC, Binutils, and Linux kernel over to git repos tracking upstream
QEMU vs Spike

• Spike is great for prototyping hardware features

• QEMU is a better tool for software development
  • Solid device support (e.g., network, disk)
  • Handy debugging features (e.g., GDB stub, monitor console)
  • More familiar to software folks
  • Faster emulation
  • Advanced features (e.g., snapshots)
More docs, more specs
Defining the RISC-V Platform

- Devices
- Interrupt controller - BERI PIC as start?
- DMA
- IOMMU
- Performance counters
- Debugging (e.g., JTAG, trace)
- Power management
Specifying RISC-V Systems

• Platform specification
  • Critical for OS developers
  • ARMv8 Server Base System Architecture (SBSA) worth a skim
• Boot architecture
  • ARMv8 Server Base Boot Requirements (SBBR) worth a skim
  • Device configuration: Device Tree, ACPI
  • Bootloader/firmware: u-boot, coreboot, UEFI (TianoCore)
• RISC-V ABI
• Hypervisor and Security
Documentation Needed

- RISC-V Assembly Guide
- Something like ARM Cortex-A Programmer's Guide
- New Contributors Guide
Let's make 2016 the year of RISC-V software
Where You Can Help

- Recruiting developers
- Upstreaming
- clang/LLVM
- QEMU
- OS ports
- Docs and platform specs
Questions?

• Contact: arun.thomas@acm.org

• See you in Boston for the 4th RISC-V workshop!