RISC-V Software Ecosystem

Andrew Waterman

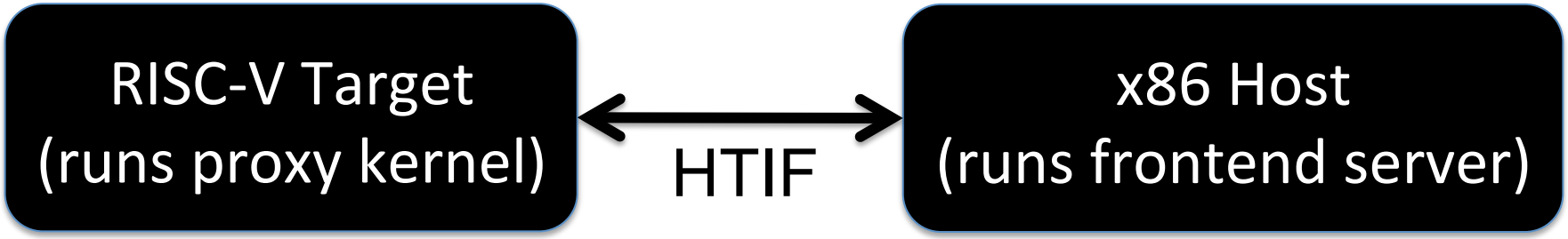
UC Berkeley

waterman@eecs.berkeley.edu

Tethered vs. Standalone Systems

- Tethered systems are those that cannot stand alone
  - They depend on a *host* system to boot
  - They may not have complete I/O facilities
- RISC-V HW implementations at Berkeley are test chips and so are tethered
- Working on software infrastructure for both kinds of systems

Tethered RISC-V Systems



- Proxy Kernel runs on RISC-V target machine
  - Supports a single user process
  - I/O-related system calls are forwarded to host
  - Implements subset of Linux ABI
- Frontend Server runs on host machine
  - Executes I/O syscalls on behalf of proxy kernel
  - Serves as boot loader (loads kernel ELF image into target memory)
- Host-Target Interface (HTIF)
  - Simple protocol for host to interact with target state

Software Implementations of RISC-V ISA

Spike ISA Simulator

- ISA interpreter; models tethered RISC-V systems
- Named for golden spike that marked completion of transcontinental railroad
  - Designed to be the golden model for functional correctness
- Reasonably fast but designed to be very easy to modify

- Github: https://github.com/ucb-bar/riscv-isa-sim

ANGEL JavaScript-based ISA Simulator

- Simulates an RV64IMA system
- Boots Linux
- Implemented purely client-side in JavaScript
- Designed for education & outreach

- Challenging to implement when only numeric type is 64-bit IEEE float!
- 13 MIPS & rising on Chrome 39.0

- riscv.org/angel

QEMU Full-System Simulator

- RV64G port of popular full-system simulator
- Device support
  - 8250 UART console
  - Virtio for network, disks, etc.
  - Can use HTIF protocol to simulate tethered system, but not the primary design goal
- Fast enough for interactive use, software development (>1 BIPS)

- Github: https://github.com/ucb-bar/riscv-qemu

Operating System Support for RISC-V

RISC-V Linux Port

- Kernel + full GNU userland
- Runs on RISC-V silicon, QEMU, Spike
- Based on 3.14 long-term branch

- Ongoing work: implementing optional features for completeness
  - CONFIG_SMP
  - Instrumentation subsystems (ftrace, kprobes, perf_events)

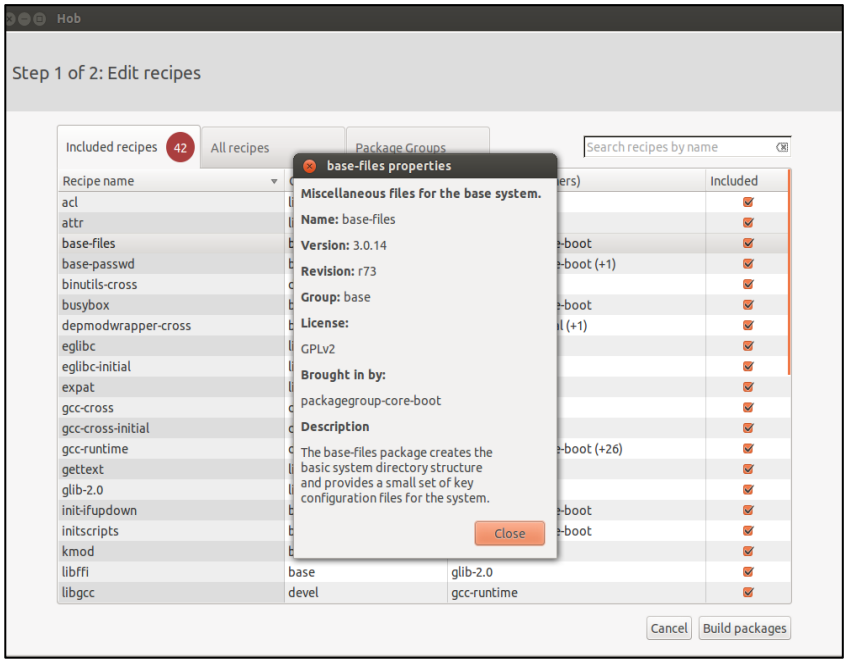
- Will upstream once new privileged spec is frozen

- Github: https://github.com/ucb-bar/riscv-linux

Poky: A Linux Distro for RISC-V

- Ported the Yocto Project, an OpenEmbedded-based Linux Distribution
  - Includes build tool that can cross-compile hundreds of packages for different architectures
  - Downloads and builds RISC-V toolchain, riscv-linux, riscv-qemu, etc. automatically (one command!)
  - Produces file system images, RPM/DEB packages, SDKs for different host platforms, etc.
- Very portable: Downloads and builds host tools
- Quality assurance throughout: Checks output files, validates checksums for inputs

Poky: A Linux Distro for RISC-V



- Instead of painfully cross-compiling all RISC-V tools and applications (plus dependencies), you can build them automatically!
- Select packages installed in your file system image, or use a package manager at runtime
- We ask the community to help grow the set of supported packages by contributing patches!

- Github: https://github.com/ucb-bar/riscv-poky

Compiler Support for RISC-V

GCC for RISC-V

- New port of GCC 4.9.2 (latest)
- C, C++, Fortran, OpenMP all working
- Support for ISA variants
  - RV[32/64]IMAFD, I, IM, IA, IMA, IMFD
- Nearing closure on test suite
  - 99.6% of tests passing
  - 0.2% failing/unresolved
  - 0.2% can't run on proxy kernel
- Generated code quality OK but improvable
- Working towards upstreaming this year

- Github: https://github.com/ucb-bar/riscv-gnu-toolchain

Binutils for RISC-V

- New port of Binutils 2.24 (latest)
- Feature-complete, including full support for dynamic linking
- GDB port, too
  - Bluespec contributed remote debugging support
  - As of yesterday, can now dump core & run gdb natively on the core dump
- Working towards upstreaming this year

Linker Relaxation in Binutils

- Expressing 32-bit addresses in RISC-V code takes multiple instructions
- Want to use shorter sequences when possible, but don't know addresses until link time
- Emit conservative code; shorten at link time
- Common cases:
  - Function calls

```
auipc t0, target[31:12]
jalr ra, t0, target[11:0]
```
→
```
jal ra, target[21:0]
```

  - Global variable references

```
auipc t0, global[31:12]
ld t0, global[11:0] (t0)
```
→
```
ld t0, offset(global ptr)
```

- 7% code size reduction for Linux kernel

C Libraries for RISC-V

- GNU libc 2.20
  - Default C library for Linux ELF toolchain
  - Full-featured (dynamic linking, pthreads, …)
  - Working towards upstreaming later this year
- Newlib 1.18
  - Default C library for generic ELF toolchain
  - glibc too bloated for embedded systems
  - Lighter-weight library with fewer OS dependences
  - Much lower code size when statically linked
- Libraries are ABI compatible
  - Newlib uses small subset of Linux system call ABI

RISC-V LLVM

- Provides code generation for user-level software
- Much easier to modify than GCC
  - Our motivation in porting was to write our own passes
- Stable port of LLVM 3.3
- Active development is on fork of LLVM trunk
  - Merged with upstream trunk weekly
  - Contributions/pull requests should be on this branch
- Plans for LLVM port
  - Work towards upstreaming
  - Increase test coverage
  - Improve robustness; clean up code
  - Improve code generation
  - Assembler, disassembler
- Github: https://github.com/ucb-bar/riscv-llvm

OpenCL Support for RISC-V

- Clang frontend + pocl OpenCL library
- LLVM RISC-V backend
- Full support for RISC-V scalar codegen

- Working on vector codegen for UCB vector machine research

RISC-V Verification Suite

- Hand-written tests for each instruction in ISA
  - Each test runs in any of several “test virtual machines”
- Simple test kernels (sorts, matmul, etc.)
  - Also serve as basic performance diagnostics
- Random test program generators, `torture` and `schadenfreude`
  - Tests combinations of instructions/μarch events that compiled code is less likely to exhibit
  - Not yet released; needs cleanup & better coverage
- We would love help from external users in improving testing & verification methodology

- Github: https://github.com/ucb-bar/riscv-tests

Questions?