



Developing Diosix

A bare-metal RISC-V-Rust hypervisor

Chris Williams

twitter.com/diodesign ... github.com/diodesign

What is it... and why?

- Diosix is a WIP bare-metal (type-1) hypervisor
 - No underlying OS. Small surface area and footprint
- Written in Rust
 - Memory safety, speed, the language of ‘no’
- Written primarily for RISC-V
 - Arm, x86 well-worn path. Undiscovered country
- Run a mix of operating systems on a system
- Rapidly develop and test low-level code
- MIT licensed

What can it do?

- Parses device trees and initializes the system
- Creates, protects, and schedules capsules
 - Provides runtime for system services that provide a UI, file-systems, etc
 - Boots ELF guest OSes, such as Linux
 - Loaded from Diosix Manifest File System (DMFS)
- Provides SBI syscall interface
 - IO, fences, reboot, etc. Implementation ID 5
- Abstracts serial port, timers, RAM, IRQs, etc

```
$ just
--> Building debug-grade Diosix for riscv64gc-unknown-none-elf systems
--> Ensuring Rust can build for riscv64gc-unknown-none-elf
info: component 'rust-std' for target 'riscv64gc-unknown-none-elf' is up to date
--> Building system services
--> Building dmfs image
--> Building hypervisor
--> Diosix built and ready for use
--> Running Diosix in Qemu
[?] CPU 0: Diosix 0.0.1 :: Debug enabled. 4 CPU cores and 1 GiB RAM found
```

[illegible]

<https://diosix.org> :: The Rust-RISC-V bare-metal hypervisor
See README and LICENSE for usage, copyright, and distribution

```
[?] CPU 0: Created capsule for system service gooeey, 931808 bytes in manifest
[?] CPU 0: Created guest OS capsule for riscv64-linux-busybox: 64-bit RISC-V Linux 5.4.58 with Busybox, 13447248 byte
[?] CPU 0: Physical CPU core RV64IMAFDC (Qemu/Unknown) ready to roll
[?] CPU 1: Physical CPU core RV64IMAFDC (Qemu/Unknown) ready to roll
[?] CPU 3: Physical CPU core RV64IMAFDC (Qemu/Unknown) ready to roll
[?] CPU 2: Physical CPU core RV64IMAFDC (Qemu/Unknown) ready to roll
```

```
[ 0.000000] Linux version 5.4.58 (chrisdiosix-dev) (gcc version 9.3.0 (Buildroot 2020.08-642-ga2830f0dad)) #2 SMP
[ 0.000000] initrd not found or empty - disabling initrd
[ 0.000000] Zone ranges:
[ 0.000000]   DMA32    [mem 0x00000000a0000000-0x00000000afffffff]
[ 0.000000]   Normal    empty
[ 0.000000] Movable zone start for each node
```

```
[ 0.852892] Freeing unused kernel memory: 1972K
[ 0.854690] This architecture does not have kernel memory protection.
[ 0.856428] Run /init as init process
```

```
Starting syslogd: OK
```

```
Starting klogd: OK
```

Running sysctl: OK

Saving random seed: OK

```
Starting network: Waiting for interface eth0 to appear..... timeout!
```

```
run-parts: /etc/network/if-pre-up.d/wait_iface: exit status 1
```

FAIL

```
Welcome to Busybox/Linux on Diosix
system-boot-capsule login:
```

Keeping the peace

- RISC-V has a base instruction set and a collection of extensions
 - Eg, A for atomic, M for integer mul & div
 - RV32IMAFDC, RV64GC
- Oh, great. There's a hypervisor extension (H)

Keeping the peace

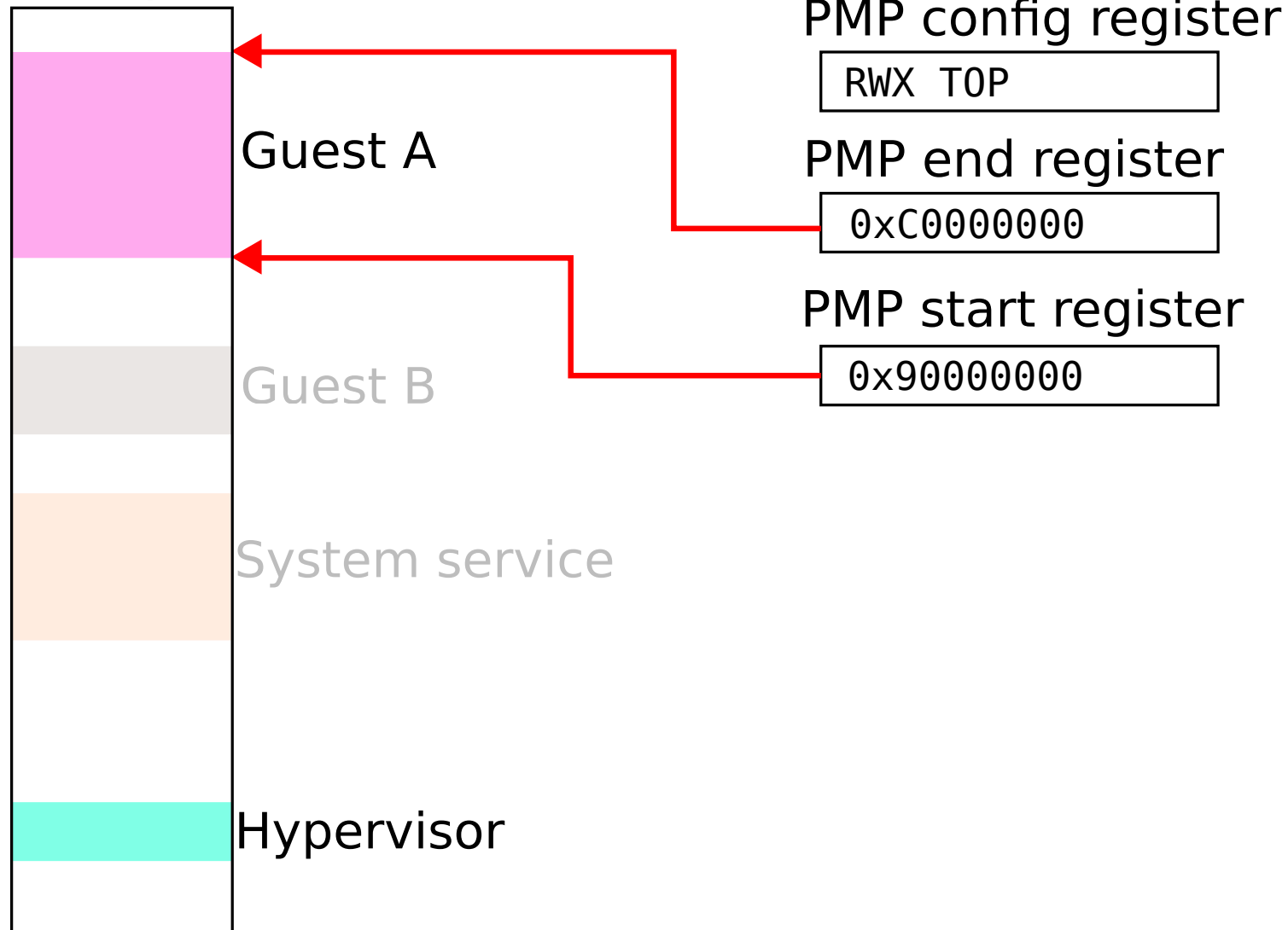
- RISC-V has a base instruction set and a collection of extensions
 - Eg, A for atomic, M for integer mul & div
 - RV32IMAFDC, RV64GC
- Oh, great. There's a hypervisor extension (H)
 - But it's not ratified and still in development

Keeping the peace

- Let's check the specification again
- Physical memory protection (PMP)
 - Available and tested in silicon and Qemu
 - Set of control registers to configure a physical RAM region type with access limits (RWX)
 - Set of address registers that configure the start and end of each region accessible by supervisor and user-level mode
 - Control registers are per CPU core (aka hart)
 - Hypervisor retains full access to all memory
 - Applies to virtual->physical memory accesses

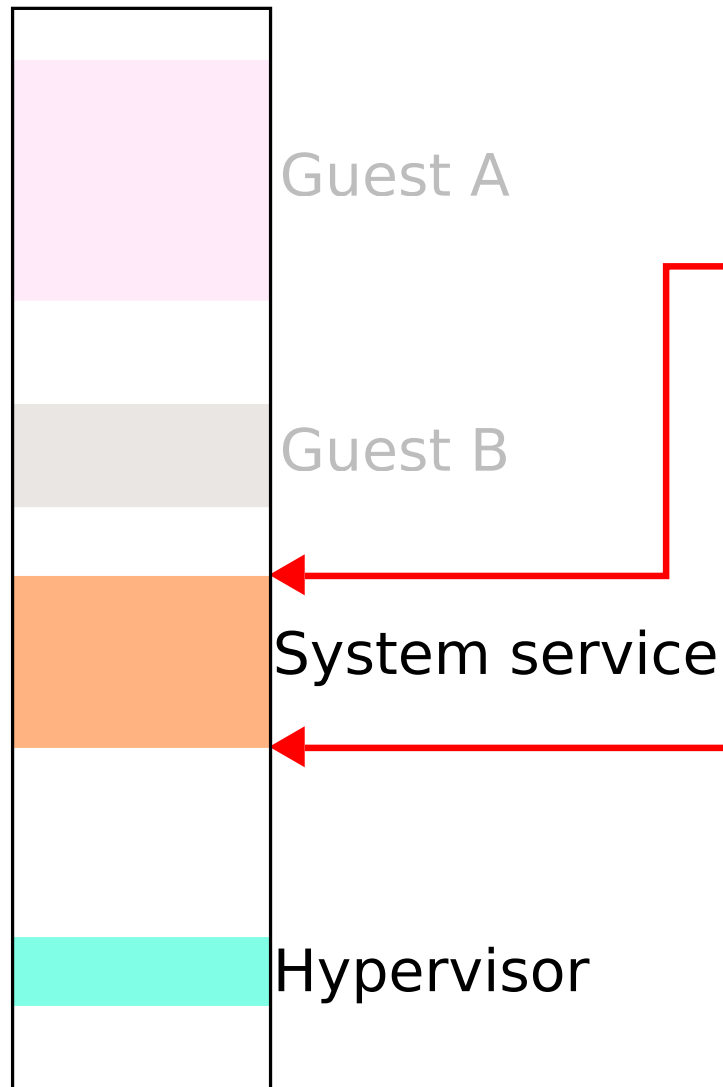
Keeping the peace

Memory
map



Keeping the peace

Memory
map



PMP config register

RWX TOP

PMP end register

0x40000000

PMP start register

0x10000000

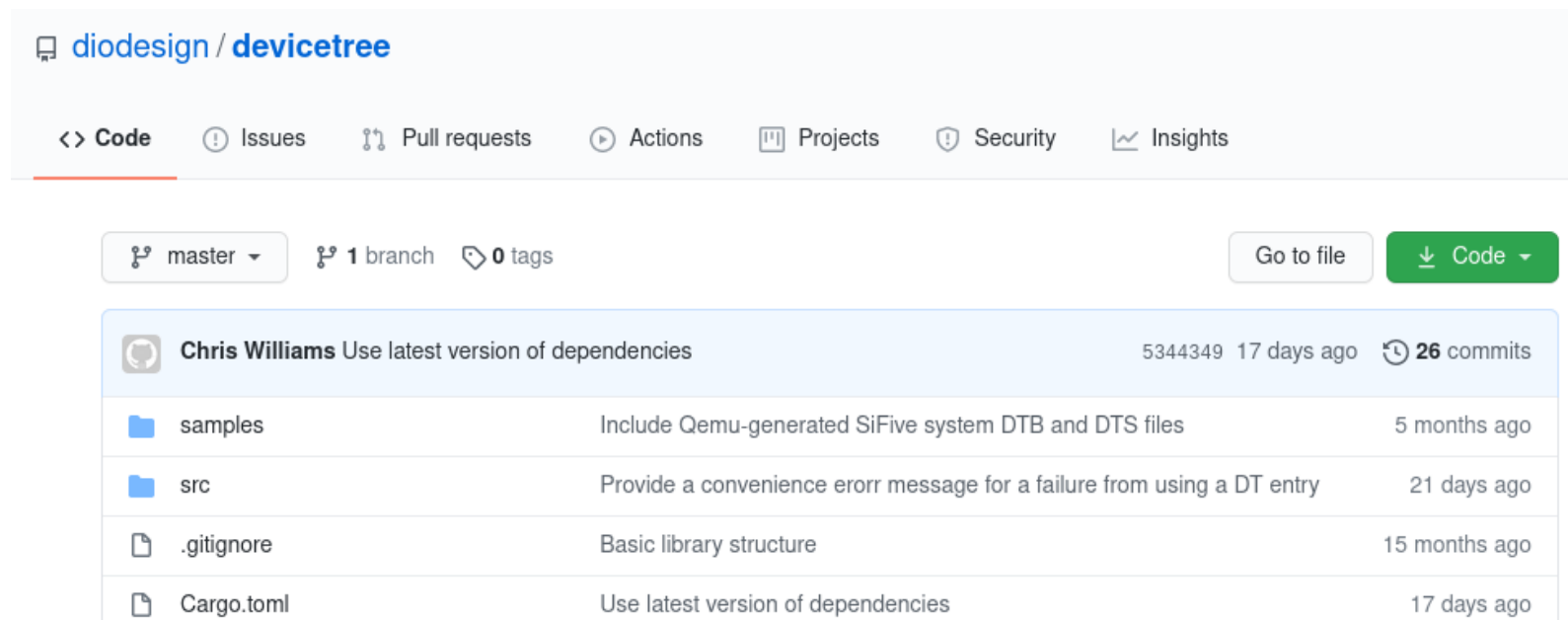


Device Tree

- Automatic discovery of hardware is essential to minimize porting work
- Device Trees are prevalent in RISC-V
- Define CPU cores, memory, peripherals, IRQ routing, boot arguments, etc
- Rust DT parsers exist but no generators I could find, so I wrote my own
- Used to inform capsules of their virtualized environment

Device Tree

- <https://github.com/diodesign/devicetree>
- Not tied to RISC-V
- Tested with real-world DTBs



The screenshot shows the GitHub repository page for `diodesign / devicetree`. The repository is in the `master` branch, has 1 branch, and 0 tags. The commit history shows a recent commit by Chris Williams titled "Use latest version of dependencies" (5344349, 17 days ago) with 26 commits. The file list includes:

File	Description	Commit Date
<code>samples</code>	Include Qemu-generated SiFive system DTB and DTS files	5 months ago
<code>src</code>	Provide a convenience error message for a failure from using a DT entry	21 days ago
<code>.gitignore</code>	Basic library structure	15 months ago
<code>Cargo.toml</code>	Use latest version of dependencies	17 days ago

Diosix manifest file system

- Simple binary format for storing guest OS images, system service executables, boot-time welcome messages, and more
- Loaded into RAM with the hypervisor from storage and unpacked during start-up
- Generated using mkdmfs tool
 - Configurable via TOML file
- Preferable to hard linking guest binaries and other files to the hypervisor

Just build it, Mason

- Rust's Cargo tool is awesome for managing Rust dependencies and compilation
- Mason is a Cargo `build.rs` script that assembles and links hardware-specific assembly code with the Rust-level hypervisor and the services runtime
 - Configurable via a TOML file
 - Rust has improved its inline assembly macro
- Wrapped in a just based build environment
 - Automates Mason, mkdmfs, Cargo



What's next?

- System services
 - User interface to control the host and its guests
 - Networking and storage access
 - Cluster orchestration
- Secure the DMFS
 - Cryptographic signatures to verify contents
 - Encrypted contents
- Support H extension and other OSes

Thank you

Source and docs
<https://diosix.org/>

Chris Williams
<https://diodesign.co.uk/>

```
--> Building hypervisor
--> Diosix built and ready for use
--> Running Diosix in Qemu
[?] CPU 0: Diosix 0.0.1 :: Debug enabled, 4 CPU cores and 1 GiB RAM found

[?] CPU 0: Created capsule for system service goodey, 931808 bytes in manifest
[?] CPU 1: Created guest OS capsule for riscv64-linux-busybox: 64-bit RISC-V Linux 5.4.58 with Busybox, 13447248 bytes in manifest
[?] CPU 1: Created guest OS capsule for riscv64-linux-busybox: 64-bit RISC-V Linux 5.4.58 with Busybox, 13447248 bytes in manifest
[?] CPU 1: Physical CPU core RV64IMAFDC (Qemu/Unknown) ready to roll
[?] CPU 3: Physical CPU core RV64IMAFDC (Qemu/Unknown) ready to roll
[?] CPU 0: Physical CPU core RV64IMAFDC (Qemu/Unknown) ready to roll
```