RISC-V 多核启动原理介绍

杨德睿 · 启元实验室

2022/08

自我介绍

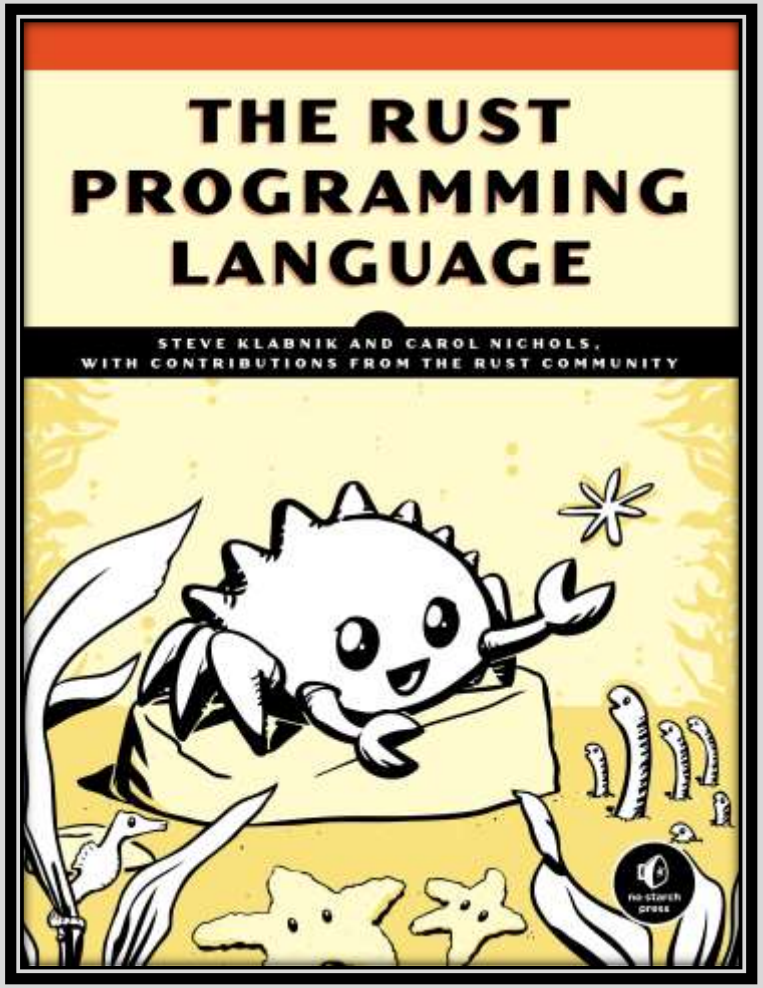
- 北京交通大学通信工程本科
- 多年从事移动机器人软件开发
- 2022 年 3 月入职启元实验室开发 zCore
  - 1 年 Rust 开发经验（做机器人算法实现和驱动开发）
  - 没有操作系统/计算机/RISC-V 经验

我们都一样！

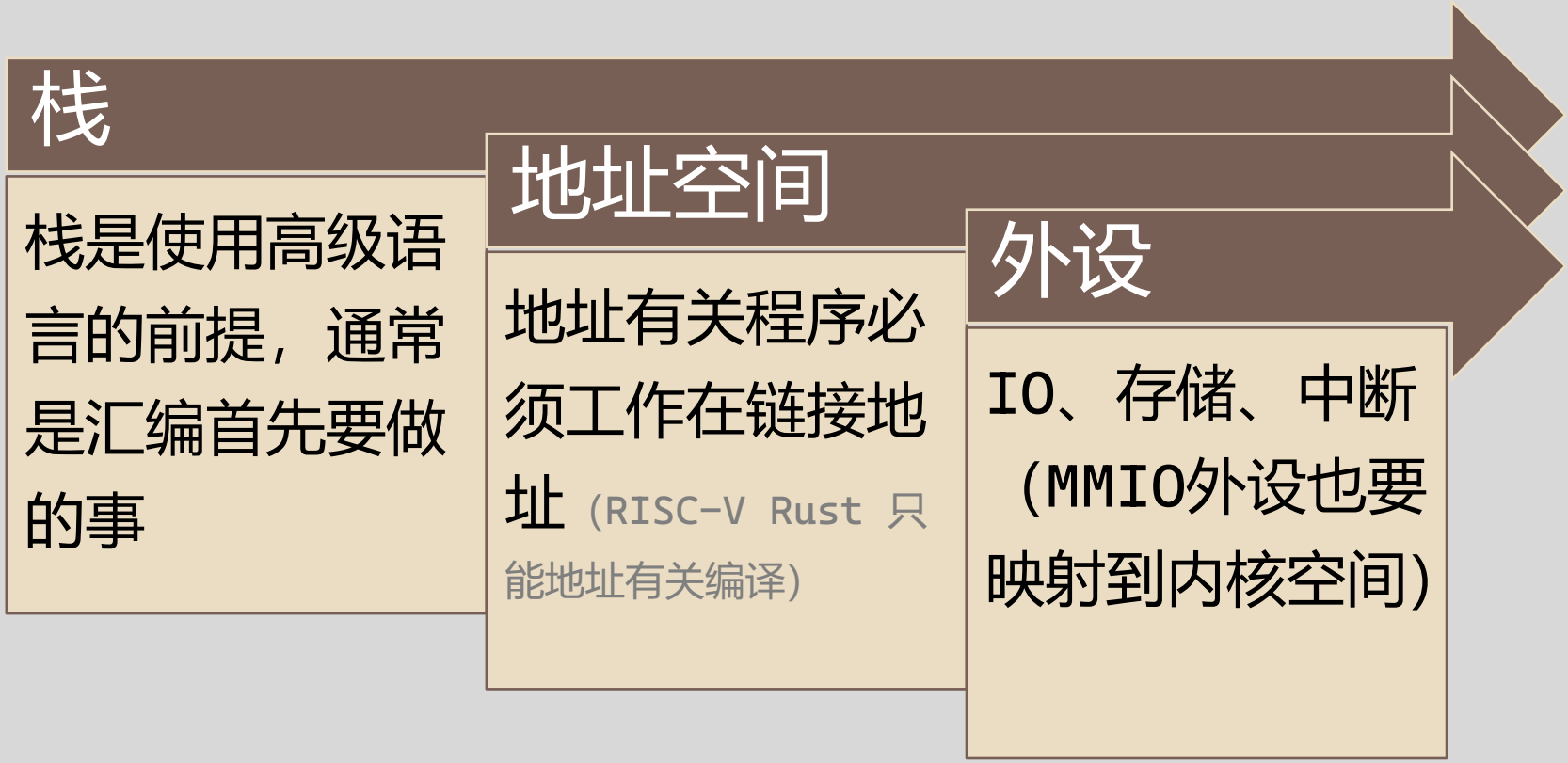
（重在交流）

大纲 · TOC

- zCore 启动流程回顾
  - 分配启动栈
  - 构造地址空间
  - 多核启动
  - 内核硬件线程用法展望
- RustSBI-Qemu 多核环境
  - Qemu-virt ROM 多核环境
  - SBI HSM 扩展支持
  - SBI 多核支持展望



zCore 启动流程


栈
栈是使用高级语言的前提，通常是汇编首先要做的事
地址空间
地址有关程序必须工作在链接地址 (RISC-V Rust 只能地址有关编译)
外设
IO、存储、中断（MMIO外设也要映射到内核空间)

分配启动栈

```rust
/// 内核入口。…
#[naked]
#[no_mangle]
#[link_section = ".text.entry"]
0 references
unsafe extern "C" fn _start(hartid: usize, device_tree_paddr: usize) -> ! {
    asm!(
        "csrw sie, zero",       // 关中断
        "call {select_stack}", // 设置启动栈
        "j    {main}",          // 进入 rust
        select_stack = sym select_stack,
        main = sym primary_rust_main,
        options(noreturn)
    )
}

/// 副核入口，此前副核被 bootloader/see 阻塞。      You, 8月前 • cleanup … …
#[naked]
1 reference
unsafe extern "C" fn secondary_hart_start(hartid: usize) -> ! {
    asm!(
        "csrw sie, zero",       // 关中断
        "call {select_stack}", // 设置启动栈
        "j    {main}",          // 进入 rust
        select_stack = sym select_stack,
        main = sym secondary_rust_main,
        options(noreturn)
    )
}
```

```rust
/// 根据硬件线程号设置启动栈。
///
/// # Safety
///
/// 裸函数。
#[naked]
0 references
unsafe extern "C" fn select_stack(hartid: usize) {
    const STACK_LEN_PER_HART: usize = 4096 * STACK_PAGES_PER_HART;
    const STACK_LEN_TOTAL: usize = STACK_LEN_PER_HART * MAX_HART_NUM;

    #[link_section = ".bss.bootstack"]
    static mut BOOT_STACK: [u8; STACK_LEN_TOTAL] = [0u8; STACK_LEN_TOTAL];

    asm!(
        "   addi t0, a0,  1",
        "   la   sp, {stack}",
        "   li   t1, {len_per_hart}",
        "1: add  sp, sp, t1",
        "   addi t0, t0, -1",
        "   bnez t0, 1b",
        "   ret",
        stack = sym BOOT_STACK,
        len_per_hart = const STACK_LEN_PER_HART,
        options(noreturn)
    )
}
```

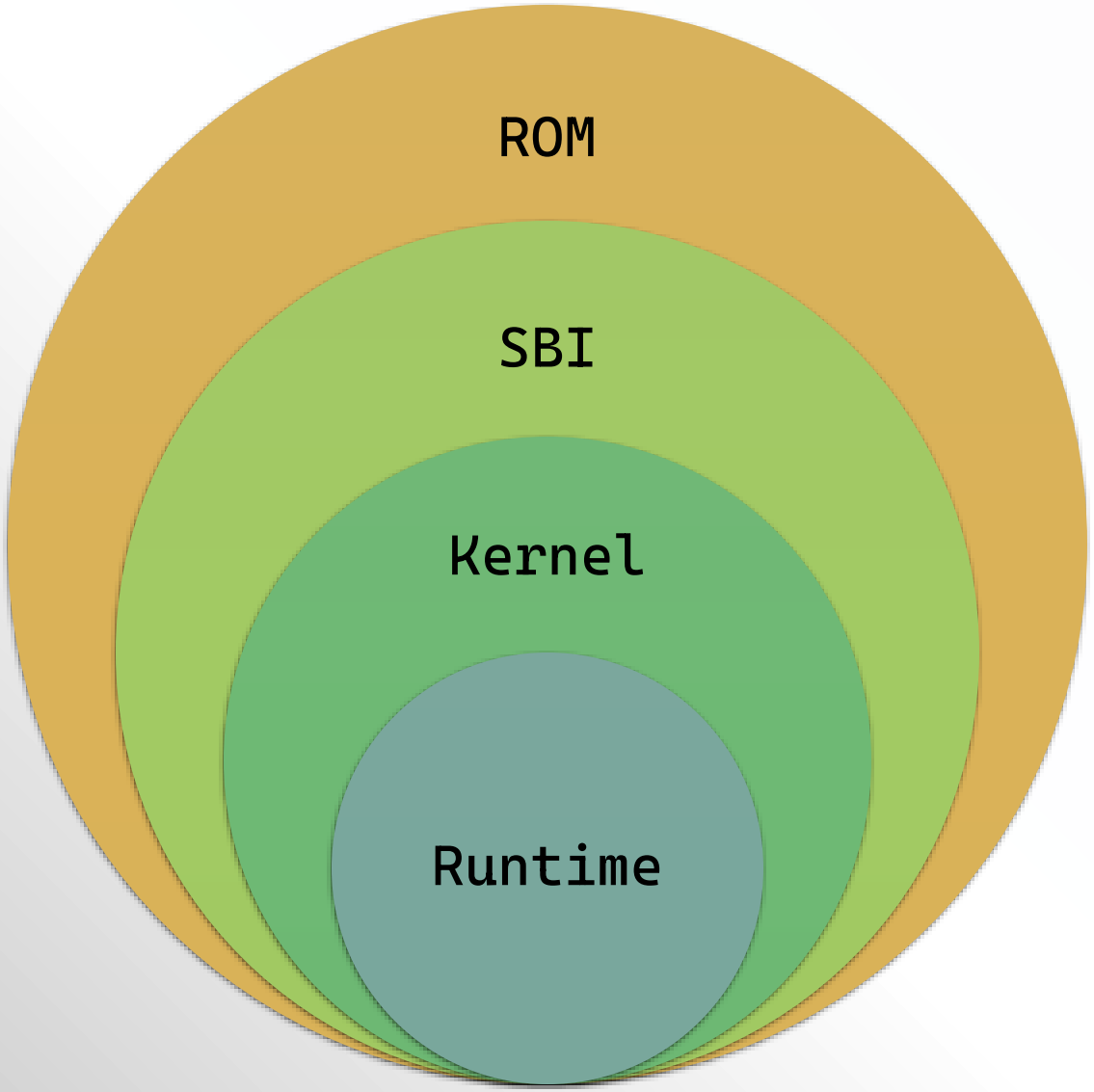
构造地址空间


探测物理空间
•原地查符号表
•找到合适的虚地址
构造启动页表
•构造跳板页
•构造 Sv39 一级
页表映射
跃迁!
•启动地址转换
•跃迁到虚地址

启动副核（Secondary hart）

内核硬件线程用法展望

现在：硬件线程是特殊的资源	期望：硬件线程是普通外设
有特殊的初始化流程	和普通外设一起（动态）初始化
无状态管理	有状态管理（空闲时休眠/停止）
在一个上下文里循环	有多种工作模式
不支持异构（某些 Linux：异构核作为特殊外设执行特殊任务）	异构核统一管理参与正常调度
特殊的调度器	正常的异步运行时（控制硬件线程组成的线程池）



RustSBI-Qemu
多核环境

ROM 准备 SBI 环境

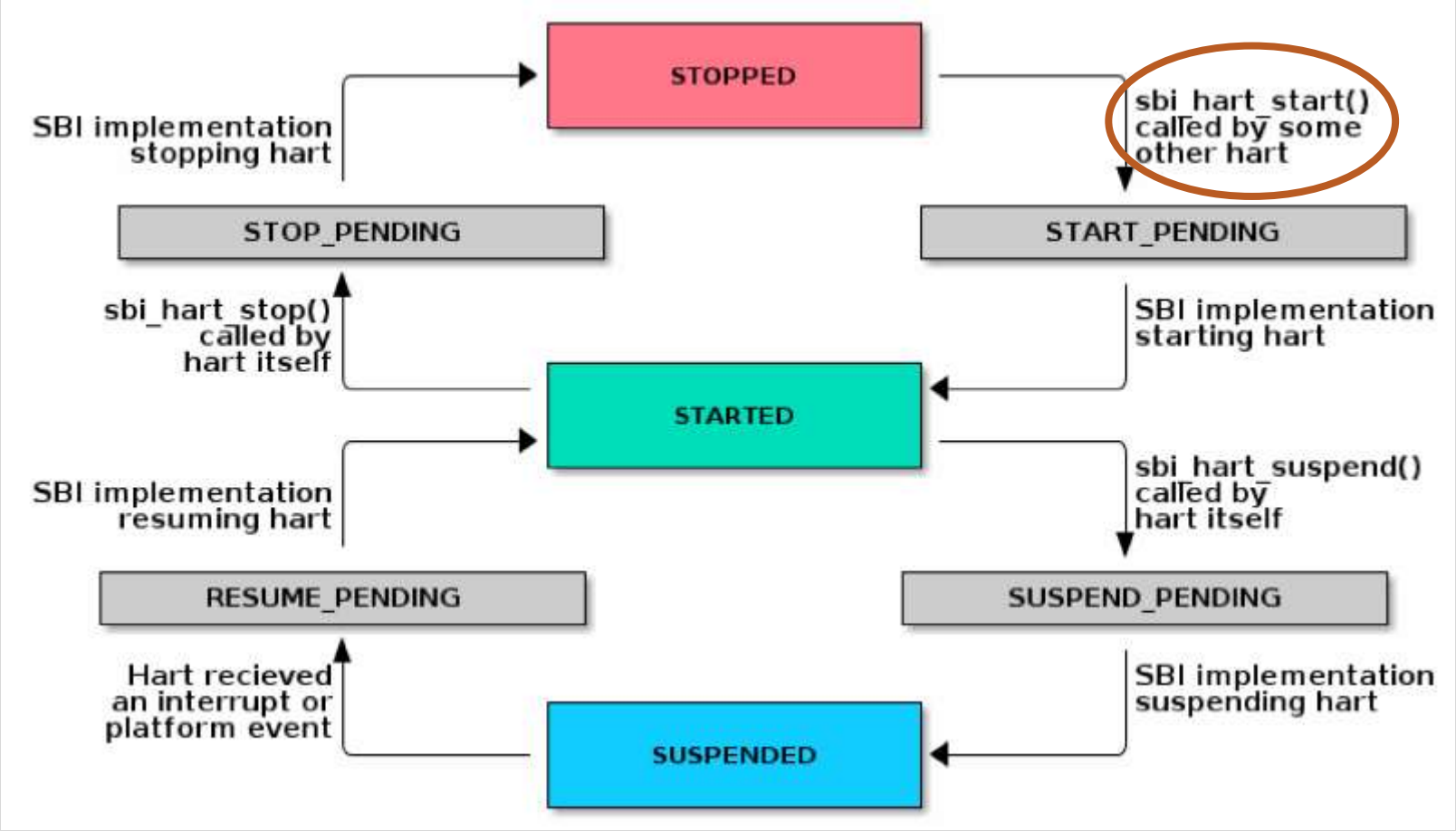
SBI 准备内核环境

内核准备 Runtime 环境

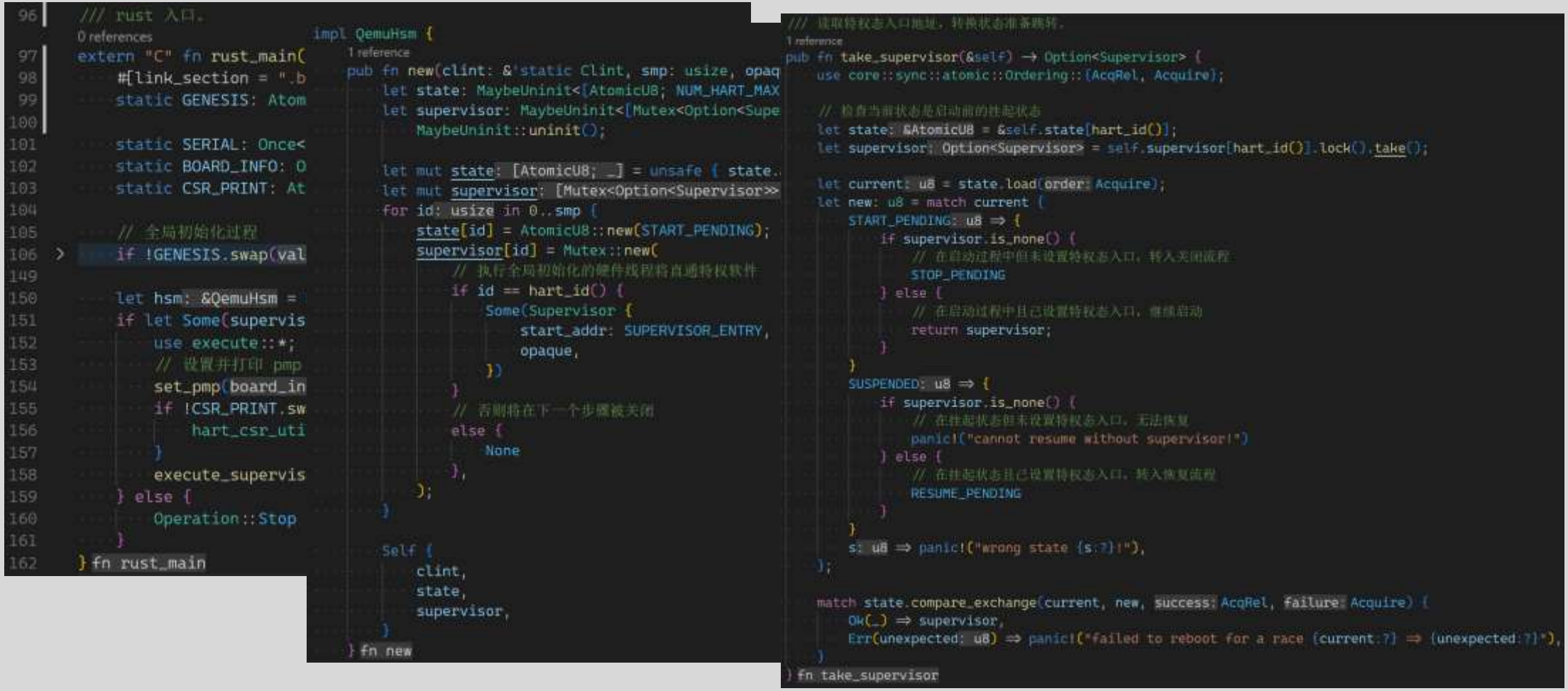
ROM/SBI 环境对比

ROM（不一定）	SBI Legacy (without HSM)	SBI Modern (with HSM)
所有硬件线程独立	一个硬件线程执行全局初始化	一个硬件线程执行全局初始化
所有硬件线程跳到固定地址	所有硬件线程跳到固定地址	每个硬件线程可以跳到不同地址
所有硬件线程并发跳转	所有硬件线程并发跳转	不执行全局初始化的硬件线程关闭
功能不统一，兼容性差	功能统一，要求内核必须接住	功能标准化，对内核要求低

SBI HSM 扩展


STOPPED
sbi_hart_start()
called by some
other hart
SBI implementation
stopping hart
STOP_PENDING
START_PENDING
sbi_hart_stop()
called by
hart itself
SBI implementation
starting hart
STARTED
SBI implementation
resuming hart
sbi_hart_suspend()
called by
hart itself
RESUME_PENDING
SUSPEND_PENDING
Hart recieved
an interrupt or
platform event
SBI implementation
suspending hart
SUSPENDED

RustSBI-Qemu(virt) 实现



SBI 多核支持展望

- SBI（Supervisor Binary Interface）只能给 Supervisor 提供接口吗?
- SBI 必须作为运行环境占住 M 态吗?

不一定!

- SBI 也可以支持异构核（小核）的管理
- SBI 也可以不占满 M 态（模拟一般 Bootloader）

Enclave-side SBI extension for Penglai enclave

- S+U enclave
  - Extension id: 0x100101
  - Use the sbi trap of

翻译 复制 搜索 全选 分享

- U enclave
  - Extension id: 0x100101
  - Use the sbi trap of CAUSE_USER_ECALL, which is undefined in the sbi specification.
- Functions:

Macro	Function
SBI_EXIT_ENCLAVE	99
SBI_ENCLAVE_OCALL	98
SBI_ACQUIRE_ENCLAVE	97
SBI_CALL_ENCLAVE	96
SBI_ENCLAVE_RETURN	95
SBI_GET_REPORT	94


欢迎提问！