-什么是SBI

-引导程序

是RISC-V提供的标准；

SBI是常驻后台的，不是启动操作系统之后就会结束；

——统一的硬件环境

从下往上为操作系统提供很多接口，在运行时，操作系统会调用SBI来获得一些和硬件相关的信息；

统一的接口被称为“监督模式二进制接口”，即SBI，是一套接口的标准

-RISC-V SBI标准

接口很多，包括重启、跨核软中断、设备树（了解硬件基本信息）等；

SBI是一个标准，有很多实现，openSBI是其中一种内置于QEMU中的实现；

-硬件运行时

对具体的硬件来说，硬件会有一些中断和异常，SBI的硬件运行时会处理这些，然后通过统一的方法提供给操作系统，例如委托

-硬件环境接口

操作系统是如何调用SBI的？类似于系统调用，SBI处于M层，操作系统处于S层，通过ecall指令让操作系统陷入到M层，操作完成后，操作系统返回S层，从而实现操作系统向环境的调用；可以屏蔽掉底层硬件的区别；

-引导启动模块

调整寄存器的值，使用mret指令下降到S层，来启动操作系统；

-兼容模块

SBI可以用作硬件的兼容，可以用旧版本指令模拟部分新版本指令，从而能在旧版本硬件上运行新版本操作系统；

-模块化

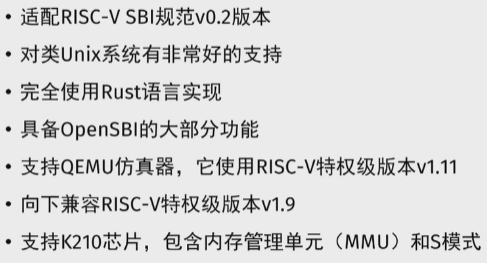
是一个模块化的标准，可以为自己的操作系统搭配SBI，不同的SBI有不同特性；

-兼容性

对于新版本的操作系统，可以用过更新SBI实现来继续使用，无需更换硬件，延长了RISC-V硬件的生命周期；

对于使用了实验性指令集的操作系统，很多时候不会针对该指令集制作出一个硬件，但可以通过修改SBI的软件，添加该指令集；

——RUSTSBI



——组成模块

-trait抽象

把SBI各个功能模块抽象为一个trait，可以理解为是一个接口；

实现trait并将函数注册到RUSTSBI之后，就可以在ecall中调用想要的功能

-ecall实现模块

支持各种二进制接口调用，分发到trait实现

-平台实现

已实现QEMU、K210等芯片，可以直接下载二进制库安装；

如果需要实现新的芯片，可以把RUSTSBI看作一个库，调用里面的函数；

RUSTSBI实现的一部分与RUST嵌入式生态有关

——Embedded-hal

-与标准对接

RUST嵌入式开发标准；

包含大量trait：串口、SBI接口等，会提供一些和嵌入式硬件接口的抽象；

Embedded-hal标准和RUST部分功能的标准比较重合，所以可以通过用Embedded-hal

实现来简化SBI实现的过程，因为社区已经有大量embedded-hal的实现；

-RUST嵌入式生态

要实现对embedded-hal支持，第一步需要生成外设访问库，第二步需要生成硬件中间层库；两个共同构成了BSP，可以完整地支持一个芯片；

外设访问库是可以用寄存器描述文件直接生成的，而硬件中间层库需要自己编写；

——高级SBI设计

-RUSTSBI可以作为库使用

可以用它实现新的芯片等，RUSTSBI会提供简单的功能实现，但如果要开发高级的实现，可以加工RUSTSBI作为一个库，为其添加很多功能，从而实现复杂的设计；

-复杂的引导程序

可以扫描磁盘启动操作系统，可以从扫描网络启动操作系统；

-带额外功能的引导程序

接到上位机，通过输入一些调试指令，用SBI监控系为了方便调试，SBI可以带一个调试接口，可以用一个专门的串口或者专门的调试通道，连统；

可以监控系统的工作行为，作为教学系统使用，可以在SBI里加入一个接口，以验证操作系统的执行是否正确；

——兼容性设计：以特权级1.9到1.11

-1.9和1.11特权级的不同点

Sstatus.sum位（1.11）和ssstatus.pum位（1.9）不同；

Sfence.vma指令（1.11）和sfence.vm指令（1.9）不同

1.9版本没有s特权级的外部中断

1.11版本有页异常；1.9版本没有单独的页异常，被包含在访问异常中

-兼容办法

1-捕获非法指令异常，转换为旧版指令

2-satp寄存器用于保存页表基址，从操作系统认为的1.11版本的satp寄存器取出数据，进行位运算之后，写入到1.9版本的sptbr寄存器

3-1.11版本satp寄存器的页表配置，对应1.9版本的mstatus寄存器

-外部中断兼容

没理解

-跨级内存访问和页异常

S层不能运行U层代码；

针对用户页的访问，需要操作系统区分；

在指令异常中处理页异常；

-运行时与SBI

