## 实验指导

在RISC-V中，由OpenSBI完成外设的扫描，而且将扫描结果放在a1寄存器中，而把HART放在a0寄存器中。

在main函数中增加两个参数，\_hart\_id,dtb\_pa.

然后我们开始解析设备树，调用rCore中的device\_tree库，然后遍历树上节点

**验证的逻辑大概是**：

检测magic属性，判断这段内存是否是设备树，如果是，则从这个节点往下遍历

**virtio:**

virtio 是一种 I/O 半虚拟化解决方案，是一套通用 I/O 设备虚拟化的程序，是对半虚拟化 Hypervisor 中的一组通用 I/O 设备的抽象。提供了一套上层应用与各 Hypervisor 虚拟化设备（KVM，Xen，VMware等）之间的通信框架和编程接口，减少跨平台所带来的兼容性问题，大大提高驱动程序开发效率。

也就是将一部分指令虚拟化，它也是一套通用框架和标准接口，解决设备兼容性问题

**探测virtio节点：**

从reg中读出设备更详细信息的放置位置，这一部分是内存读写MMIO，然后修改entry.asm和memory\_set.rs

修改如下：

.8byte (0x00000 << 10) | 0xcf 将外设映射到0xffff\_ffff\_0000\_0000

range: Range::from(DEVICE\_START\_ADDRESS..DEVICE\_END\_ADDRESS)

增加了一个Segment段

**virtio\_drivers 库**

同样也是利用rCore中的virtio\_drivers库

首先为DMA分配内存，我们需要告诉DMA 设备内存的物理地址，栈上申请一个 请求的结构，这个结构的物理地址也要告诉设备

**思考：为什么物理地址到虚拟地址转换直接线性映射，而虚拟地址到物理地址却要查表？**

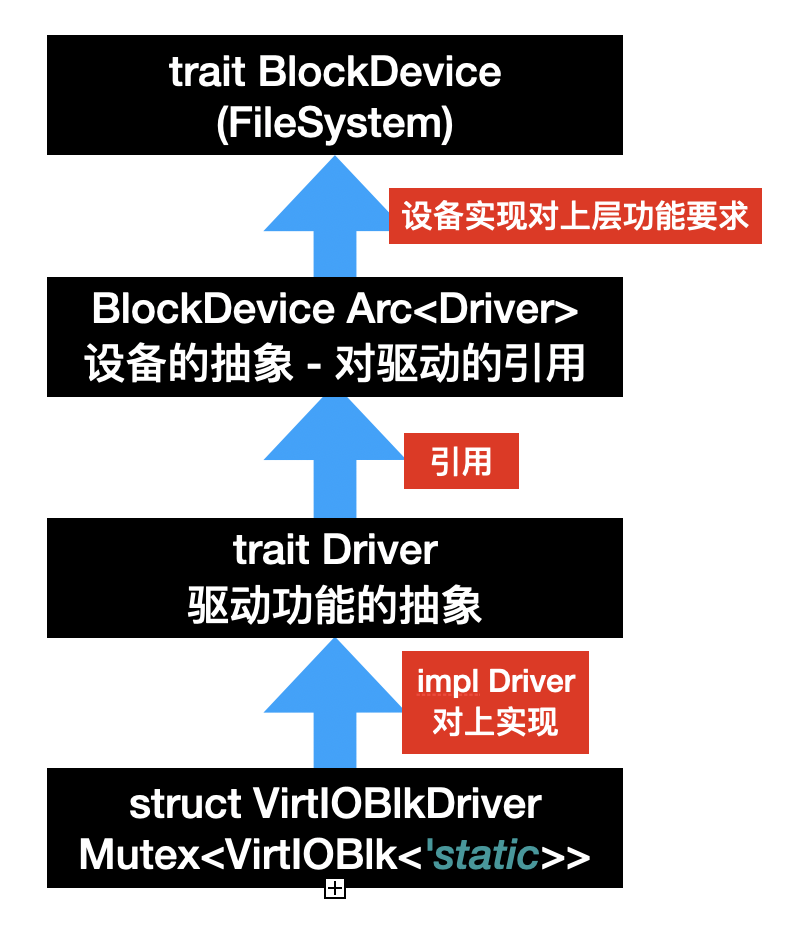
在内核线程中，只要一个物理地址加上偏移得到的虚拟地址肯定是可以访问对应的物理地址的，所以，物理地址转为虚拟地址加个偏移即可。

但是虚拟地址有可能从0开始，通过线性映射不一定能够找到对应的物理地址，所以我们需要通过查表。

**抽象驱动：**

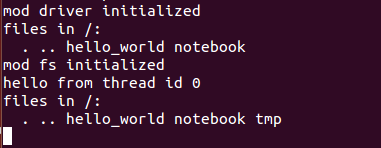
DeviceType enum 也就是设备的类型，定义它是为了更好的接入其他设备

定义读写接口，后面通过抽象块设备与调用接口来实现驱动。



**文件系统：**

调用rCore中的文件系统模块rcoce-fs，选取最简单的文件系统



测试结果如上