

# 虚拟化原理和最简Hypervisor

虚拟化概念

虚拟化对象

H\_1\_0 VirtualMode——虚拟化模式切换

执行流程：

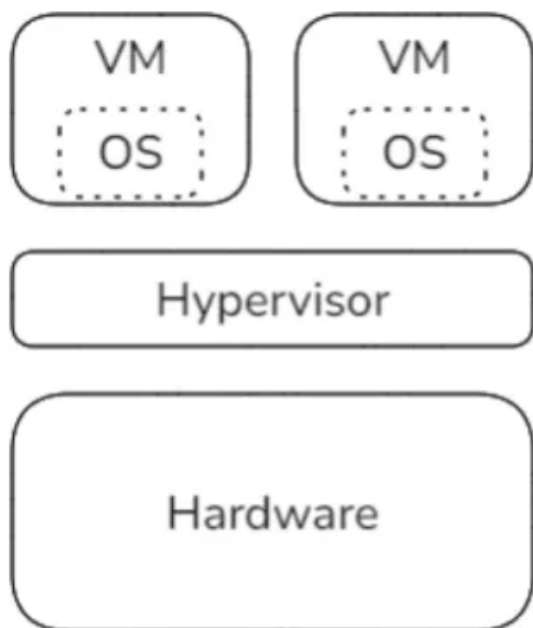
如何进入虚拟化模式

课后练习——VMEXIT

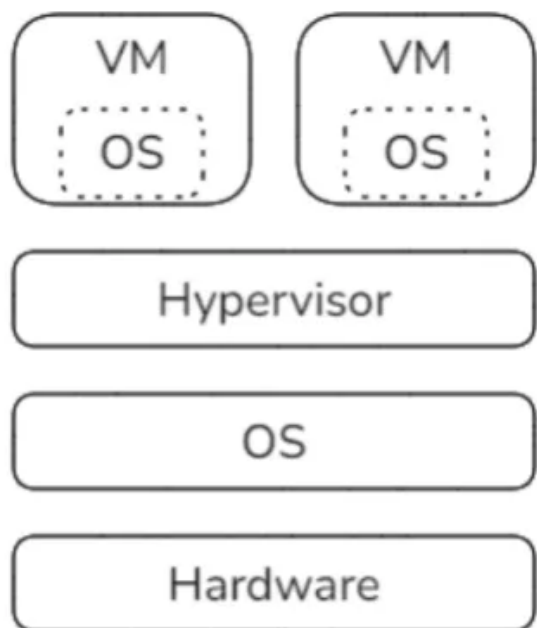
## 虚拟化概念

- 同质：ISA的同构
- 高效：虚拟化消耗可忽略
- 资源受控：中间层对物理资源的完全控制

两类



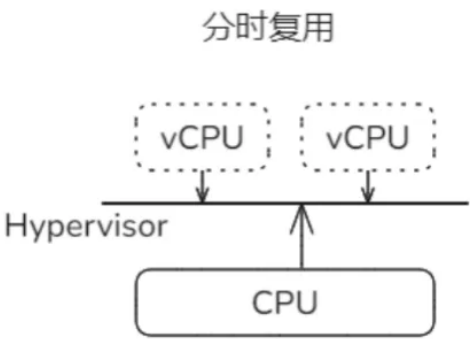
I型：直接在硬件平台运行



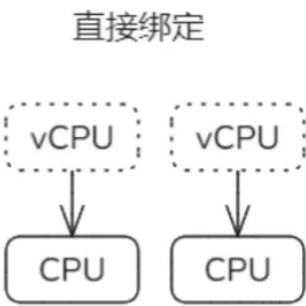
II型：在宿主OS之上运行

# 虚拟化对象

- VM：管理地址空间；同一整合下级各类资源
- vCPU：计算资源虚拟化，VM中执行流



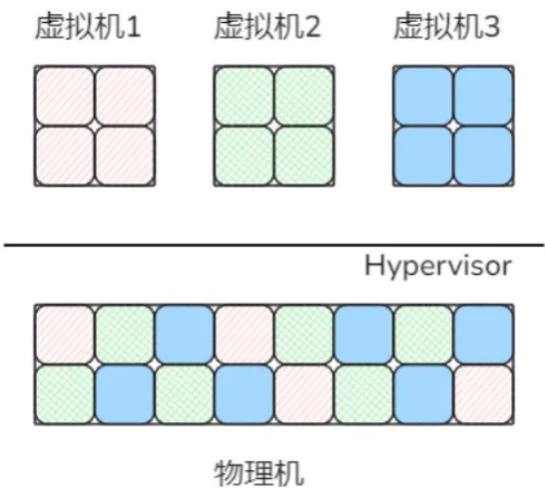
通常方法：效率较低  
Hypervisor建立多任务  
对应多个vCPU



效率较高，vCPU数量受限

本系列实验基于直接绑定模式，因为其实现简单

- vMem：内存虚拟化，按照VM的物理空间布局

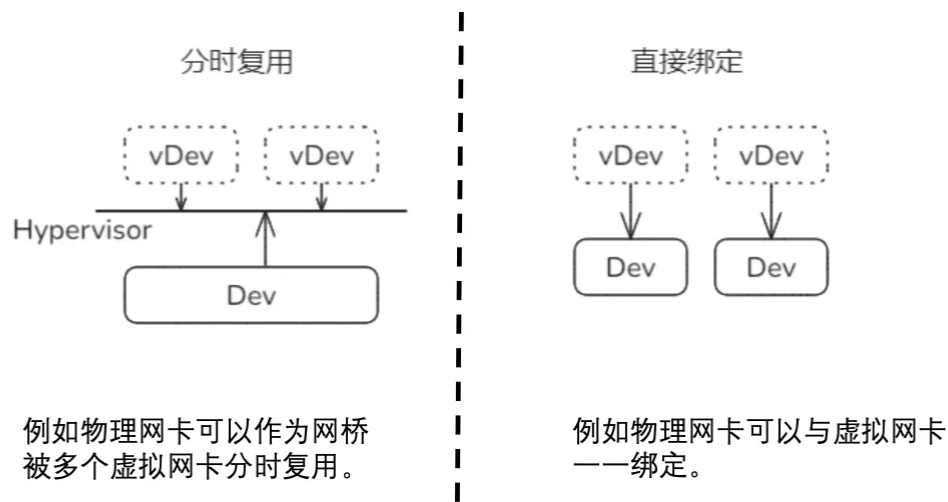


从每个虚拟机的角度，它们的物理内存是连续的。  
实际它们并不拥有实际的物理内存，只是假象。

从物理机的角度，分属各虚拟机的内存页是  
不连续的，间隔零散的。

- vDevice：设备虚拟化：包括直接映射和模拟

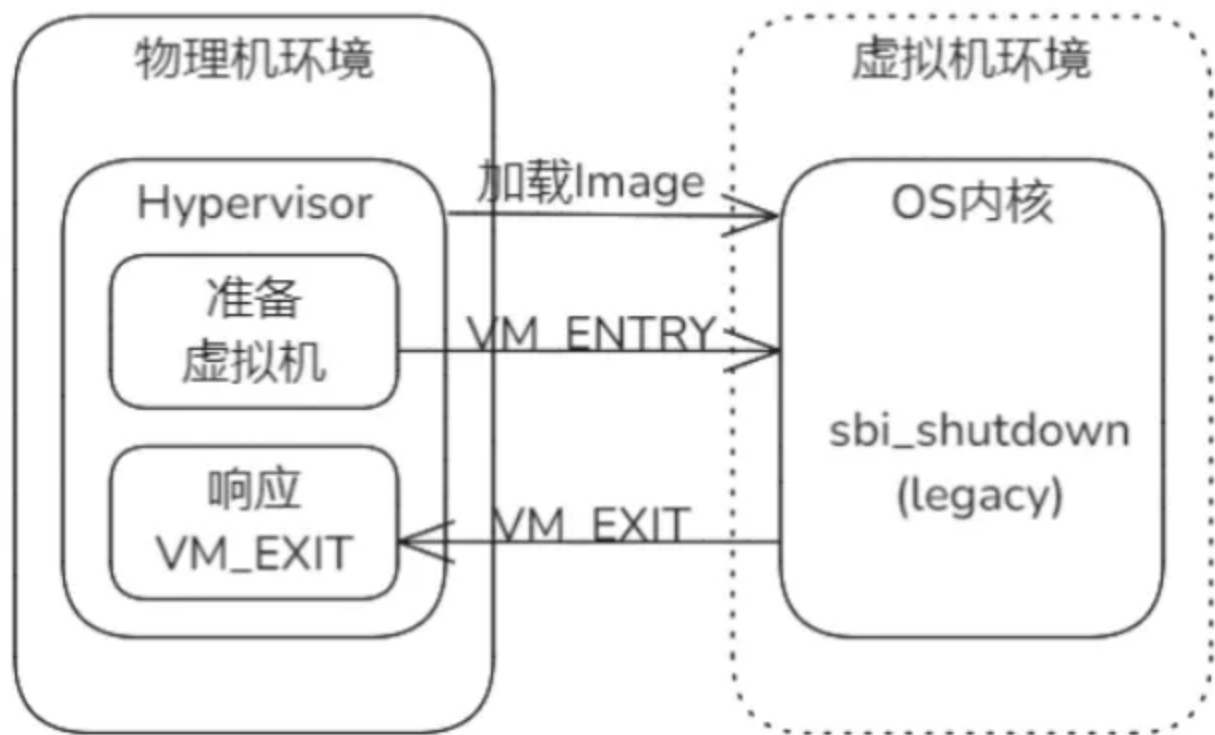
类似于CPU的情况，虚拟设备与物理设备的对应关系也是两种。



- vUtilities：中断虚拟化、总线发现设备等

## H\_1\_0 VirtualMode——虚拟化模式切换

### 实验 h\_1\_0

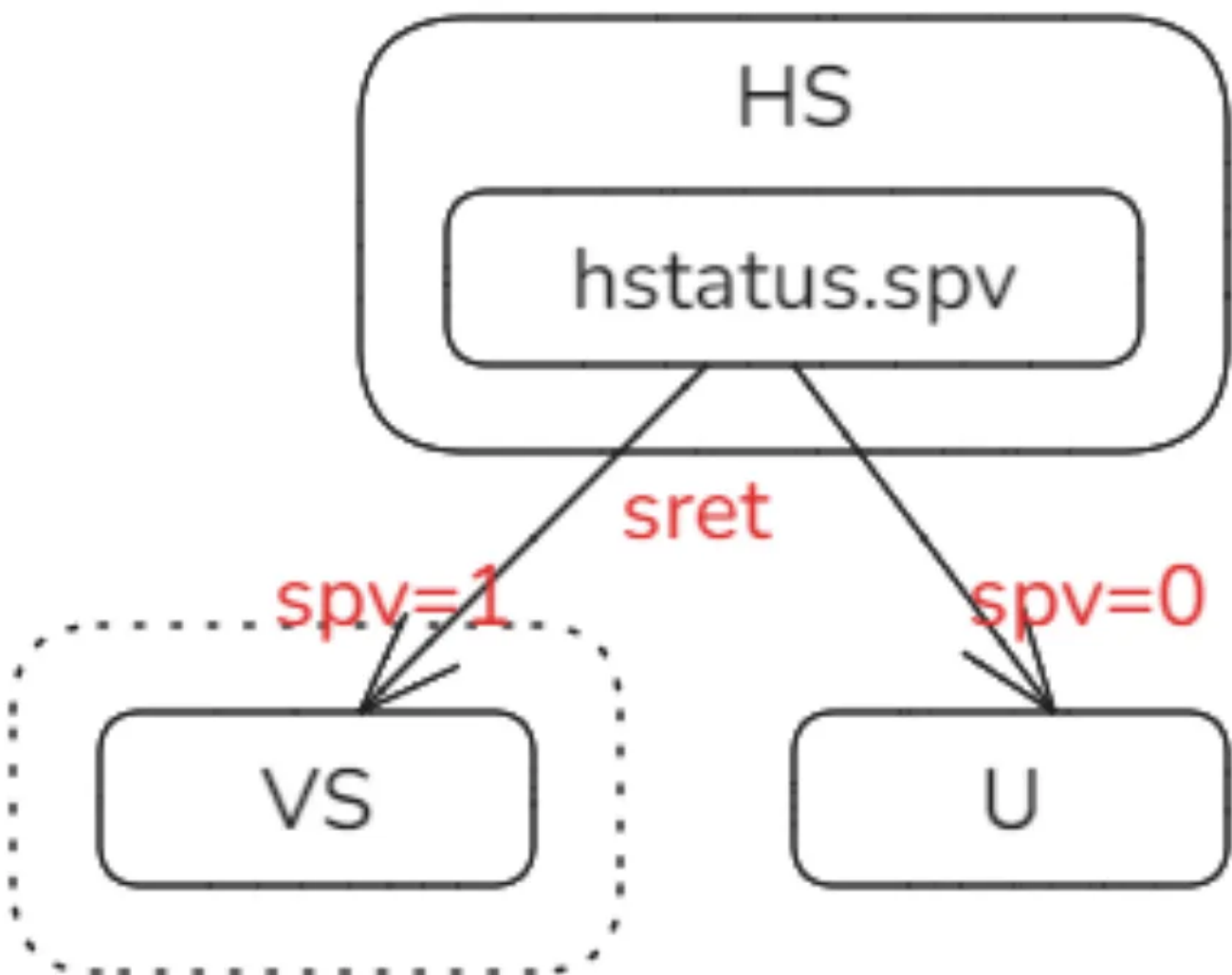


## 执行流程：

- 1.hypervisor加载image到新建的地址空间
- 2.hypevisor准备虚拟机环境，准备上下文
- 3.hypervisor设置VM\_ENTRY并通过sret变成v模式，进入虚拟机环境
- 4.虚拟机执行sbi\_shutdown指令，超出了V模式特权级权限，通过VM\_EXIT退出虚拟机环境，进入hypervisor，hypervisor通过检查VM\_EXIT的参数来执行相应的处理函数。

## 如何进入虚拟化模式

- ISA寄存器misa第7位代表Hypervisor扩展的启用/禁止。对这一位写入0，可以禁止H扩展。
- 进入V模式路径的控制：hstatus第7位SPV记录上一次进入HS特权级前的模式，1代表来自虚拟化模式。执行sret时，根据SPV决定是返回用户态，还是返回虚拟化模式。



```
// Set Guest bit in order to return to guest m
hstatus.modify(hstatus::spv::Guest);
// Set SPVP bit in order to accessing VS-mode
hstatus.modify(hstatus::spvp::Supervisor);
```

SPV指示特权级模式的来源;  
SPVP指示HS对V模式下地址空间是否有操作权限,  
1表示有权限操作, 0无权限。

- 伪造上下文
  - 设置Guest的sstatus, 让其初始特权级为Supervisor;
  - 置Guest的sepc为OS启动入口地址VM\_ENTRY

## 课后练习——VMEXIT

做法上就是处理中断, 理解了整个流程的话较为简单

```
main.rs .../simple_hv/... X vcpu.rs .../riscv_vcpu/... 9+ csrs.rs task.rs Makefile payload
exercises > simple_hv > src > main.rs > vmexit_handler
62 fn prepare_vm_pgtbl(ept_root: PhysAddr) {
63 }
64
65 fn run_guest(ctx: &mut VmCpuRegisters) -> bool {
66     unsafe {
67         _run_guest(state: ctx);
68     }
69
70     vmexit_handler(ctx)
71 }
72
73 #[allow(unreachable_code)]
74 fn vmexit_handler(ctx: &mut VmCpuRegisters) -> bool {
75     use scause::{Exception, Trap};
76
77     let scause: Scause = scause::read();
78     match scause.cause() {
79         Trap::Exception(Exception::VirtualSupervisorEnvCall) => {
80             let sbi_msg: Option<SbiMessage> = SbiMessage::from_regs(args: ctx.guest_regs.gprs.a_regs()).ok();
81             ax_println!("VmxExit Reason: VSuperEcall: {:?}", sbi_msg);
82             if let Some(msg) = sbi_msg {
83                 match msg {
84                     SbiMessage::Reset(_) => {
85                         let a0: usize = ctx.guest_regs.gprs.reg(reg_index: A0);
86                         let a1: usize = ctx.guest_regs.gprs.reg(reg_index: A1);
87                         ax_println!("a0 = {:#x} a1 = {:#x}", a0, a1);
88                     }
89                 }
90             }
91         }
92     }
93 }
94
95
```

```
问题 63 输出 终端 GITLENS
> > 终端
log_level = warn
[ 1.177677 0 fatfs::dir:139] Is a directory
[ 1.246383 0 fatfs::dir:139] Is a directory
[ 1.320762 0 fatfs::dir:139] Is a directory
[ 1.409595 0 fatfs::dir:139] Is a directory
Hypervisor ...
app: /sbin/skernel2
paddr: PA:0x80642000
Bad instruction: 0xf14025f3 sepc: 0x80200000
LoadGuestPageFault: stval0x40 sepc: 0x80200004
VmxExit Reason: VSuperEcall: Some(Reset(Reset { reset_type: Shutdown, reason: NoReason }))
a0 = 0x6688, a1 = 0x1234
Shutdown vm normally!
[ 1.539477 0:2 axruntime::lang_items:5] panicked at exercises/simple_hv/src/main.rs:59:5:
Hypervisor ok!
ziz@ziz:~/arceos/oscamp/arceos$
```

