ARM体系结构与硬件虚拟化

分享人：莫策

2023年11月6日

ARM体系结构

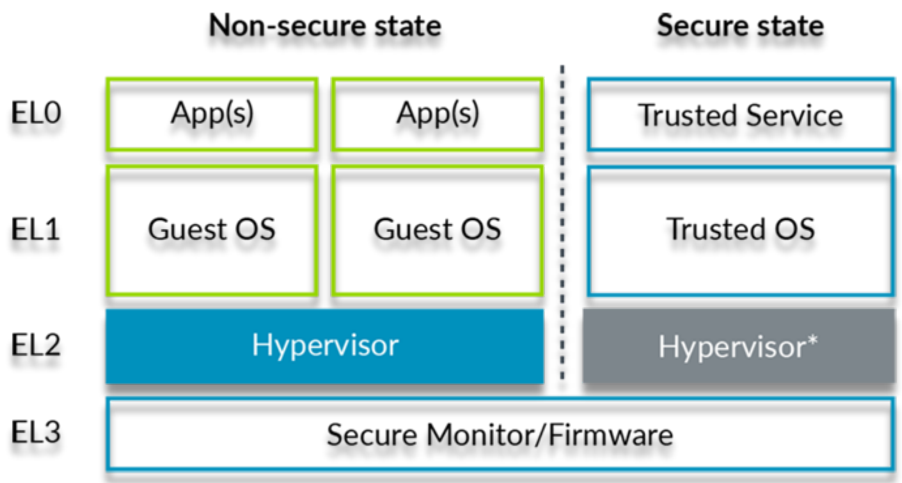
什么是ARM

- ARM，全称“Advanced RISC Machine”
- RISC：Reduced Instruction Set Computer精简指令集
  - 同属RISC的指令集：MIPS，RISC-V等
  - Complex Instruction Set Computing (CISC)：x86等
- ARM处理器的特点：
  - 指令数量少，设计相对简单；
  - 能耗较低，在移动终端应用广泛；

ARM指令集的发展和分类

- 1985~2001：ARMv1~ARMv6
- 2004：ARMv7发布，出现了Cortex-M，Cortex-R和**Cortex-A**三类处理器，支持**硬件虚拟化**
- 2011：**ARMv8**发布，ARM的第一个64位指令集架构
  - ARMv8的64位执行状态也叫aarch64（也可称为aarch64指令集）
- 2021：ARMv9发布，兼容ARMv8

ARMv8的特权等级



	Non-secure state		Secure state
EL0	App(s)	App(s)	Trusted Service
EL1	Guest OS	Guest OS	Trusted OS
EL2	Hypervisor		Hypervisor*
EL3	Secure Monitor/Firmware		

- EL0（用户态）：Applications.
- EL1（内核态）：OS kernel and associated functions that are typically described as privileged.
- **EL2：Hypervisor.**
- EL3：Secure monitor.

- Worlds：Normal World and Secure world

ARMv8的寄存器

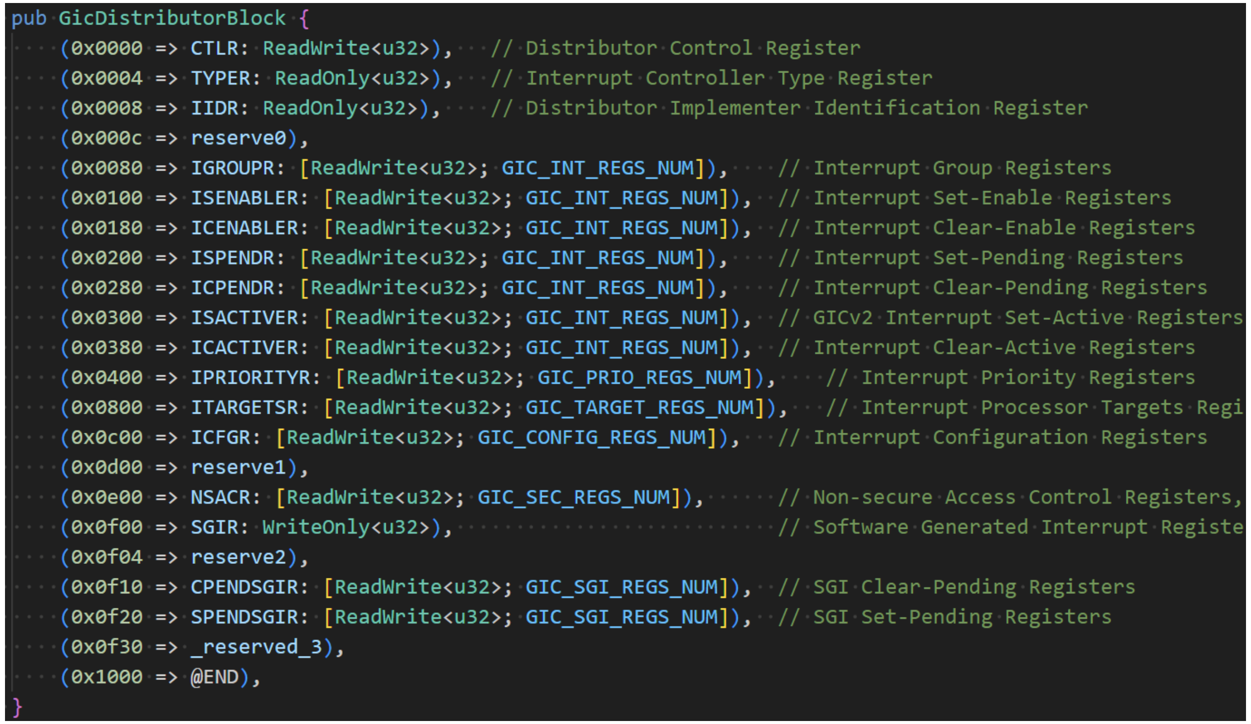
- **通用寄存器（31个，不包含sp）**
- PSTATE（Process state，程序状态信息的集合，是一组寄存器）
  - SPSR，DAIF，SPSel，CurrentEl等
- **系统寄存器（很重要）**
- Float Point寄存器/SIMD寄存器（与虚拟化无关）

ARMv8部分重要的寄存器

- HCR_EL2：Hypervisor Configuration Register
- SCTLR_ELx ：System Control Register
- TTBR0_ELx，TTBR1_EL1：Translation Table Base Register
- TCR_ELx：Translation Control Register
- SPSR_ELx：Saved Program Status Register
- ELR_ELx：Exception Link Register
- ESR_ELx,：Exception Syndrome Register
- VBAR_ELx：Vector Base Address Register
- ……

ARMv8中断控制器：以GICv2为例

- GIC中断分类：
  - Shared Peripheral Interrupts (SPIs)
  - Private Peripheral Interrupts (PPIs)
  - Software Generated Interrupts (SGIs)
- GICv2的硬件组成
  - Distributor：GICD从外设接收中断
  - CPU interface：GICC CPU接收中断
  - GIC虚拟化相关
    - Virtual CPU interface：GICV
    - Virtual interface control：GICH

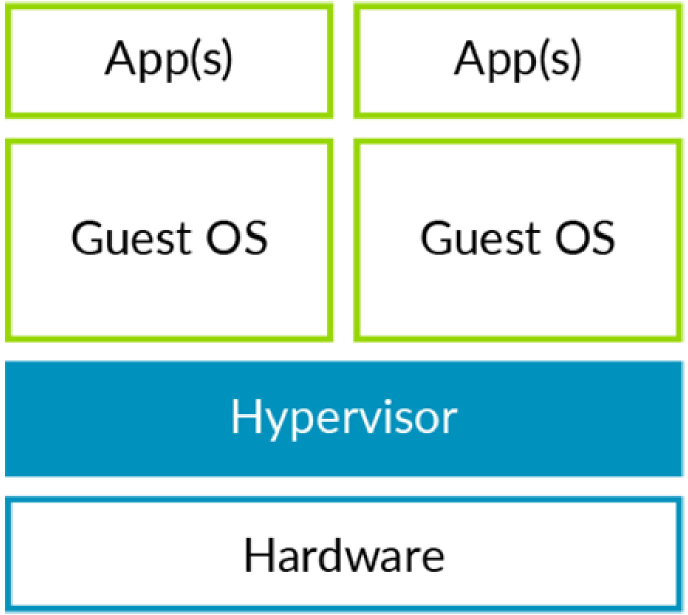


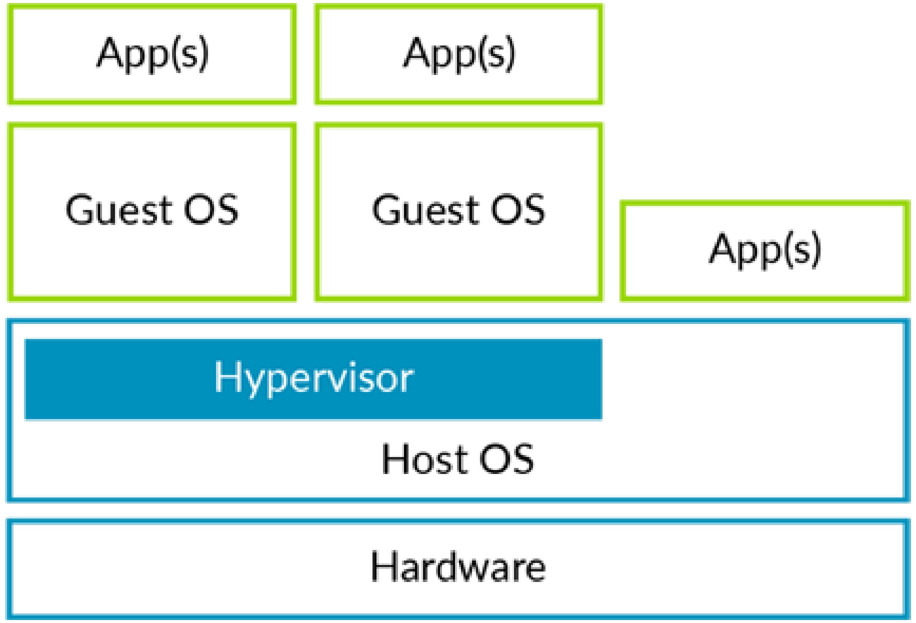
```
pub GicDistributorBlock {
    (0x0000 => CTLR: ReadWrite<u32>),   // Distributor Control Register
    (0x0004 => TYPER: ReadOnly<u32>),   // Interrupt Controller Type Register
    (0x0008 => IIDR: ReadOnly<u32>),    // Distributor Implementer Identification Register
    (0x000c => reserve0),
    (0x0080 => IGROUPR: [ReadWrite<u32>; GIC_INT_REGS_NUM]),    // Interrupt Group Registers
    (0x0100 => ISENABLER: [ReadWrite<u32>; GIC_INT_REGS_NUM]),  // Interrupt Set-Enable Registers
    (0x0180 => ICENABLER: [ReadWrite<u32>; GIC_INT_REGS_NUM]),  // Interrupt Clear-Enable Registers
    (0x0200 => ISPENDR: [ReadWrite<u32>; GIC_INT_REGS_NUM]),    // Interrupt Set-Pending Registers
    (0x0280 => ICPENDR: [ReadWrite<u32>; GIC_INT_REGS_NUM]),    // Interrupt Clear-Pending Registers
    (0x0300 => ISACTIVER: [ReadWrite<u32>; GIC_INT_REGS_NUM]),  // GICv2 Interrupt Set-Active Registers
    (0x0380 => ICACTIVER: [ReadWrite<u32>; GIC_INT_REGS_NUM]),  // Interrupt Clear-Active Registers
    (0x0400 => IPRIORITYR: [ReadWrite<u32>; GIC_PRIO_REGS_NUM]),    // Interrupt Priority Registers
    (0x0800 => ITARGETSR: [ReadWrite<u32>; GIC_TARGET_REGS_NUM]),   // Interrupt Processor Targets Regi
    (0x0c00 => ICFGR: [ReadWrite<u32>; GIC_CONFIG_REGS_NUM]),   // Interrupt Configuration Registers
    (0x0d00 => reserve1),
    (0x0e00 => NSACR: [ReadWrite<u32>; GIC_SEC_REGS_NUM]),      // Non-secure Access Control Registers,
    (0x0f00 => SGIR: WriteOnly<u32>),                           // Software Generated Interrupt Registe
    (0x0f04 => reserve2),
    (0x0f10 => CPENDSGIR: [ReadWrite<u32>; GIC_SGI_REGS_NUM]),  // SGI Clear-Pending Registers
    (0x0f20 => SPENDSGIR: [ReadWrite<u32>; GIC_SGI_REGS_NUM]),  // SGI Set-Pending Registers
    (0x0f30 => _reserved_3),
    (0x1000 => @END),
}
```

ARM硬件虚拟化

Learn the architecture - AArch64 virtualization

虚拟化分类





- Type-1
- Type-2

ARMv8硬件虚拟化提供的设备或机制

两阶段地址翻译——内存虚拟化

- MMU

虚拟中断——中断控制器的虚拟化

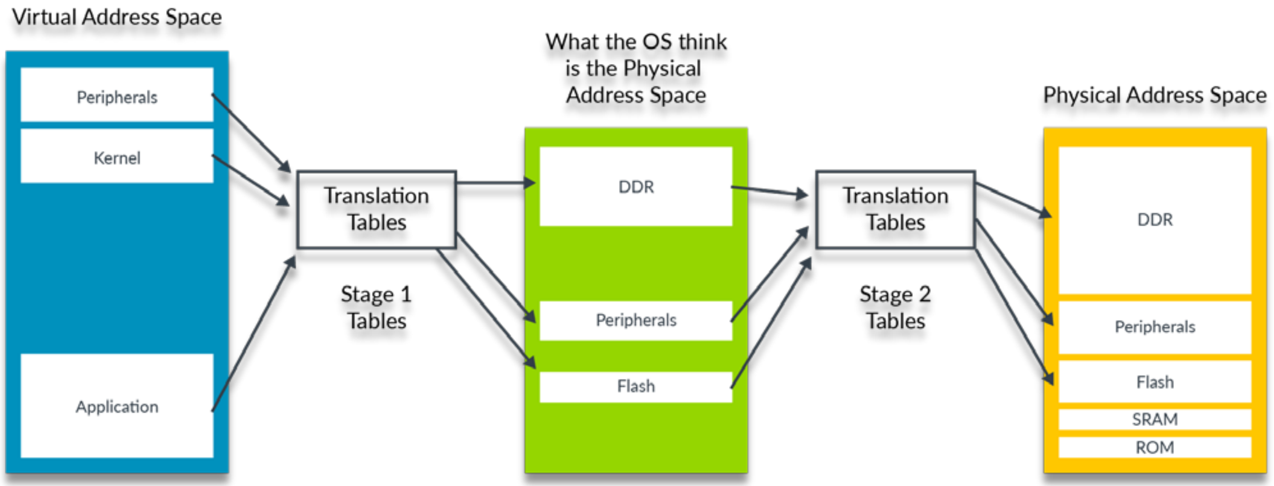
- GICV、GICH

DMA设备的直通——外设的内存视图控制

- System MMU(SMMU)

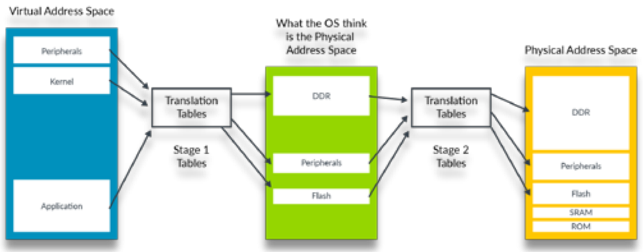
ARMv8内存虚拟化：嵌套页表

- **两阶段地址翻译：VA->IPA->PA**
- VM虚拟地址(VA)地址翻译后得到的地址不再是真实物理地址(PA)，而是中间物理地址(IPA)；
- VM提供一阶段地址翻译的页表，保留了虚拟机管理内存的能力；
- Hypervisor提供二阶段地址翻译的页表，控制虚拟机的内存视图；
- MMIO模拟设备、共享内存......
- 内存超配、内存气球......

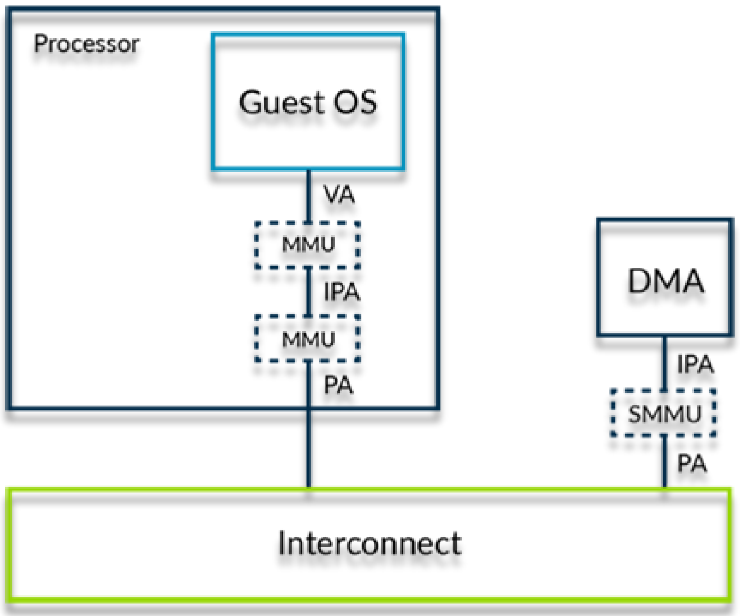


ARMv8内存虚拟化：嵌套页表

- **问题：**如果VM和Hypervisor的两阶段页表均采用**三级页表**的形式，那么用户程序的一次访存，**至多需要访问多少次内存？**



ARMv8 I/O虚拟化：DMA设备直通



- 虚拟化的设备模型：直通设备
- DMA设备具备直接访问内存的能力；
- 没有IOMMU：DMA设备获取读写的独立地址PA；
- 虚拟化环境：DMA设备获取到的是中间物理地址IPA
  - **DMA 设备访存不受第二阶段地址翻译约束，**此时DMA设备按照IPA读写物理内存，会破坏内存隔离；
  - VM和DMA设备对拥有不同的内存视图；
- 解决方案：**SMMU**，是一种IOMMU；
- Hypervisor需要模拟并配置SMMU；

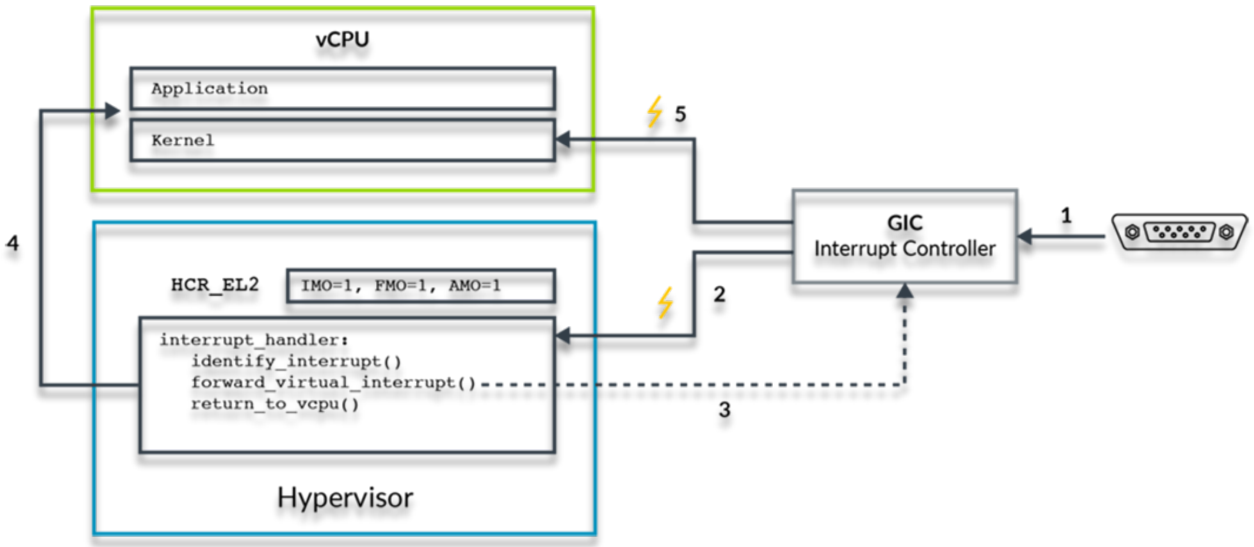
ARMv8 中断虚拟化：以GICv2为例

Hypervisor的工作：

- “接管”GICD和GICC，配置GICH
- 为VM模拟GICD，直通GICV作为GICC
- 设置HCR_EL2寄存器

Hypervisor如何注入虚拟中断：

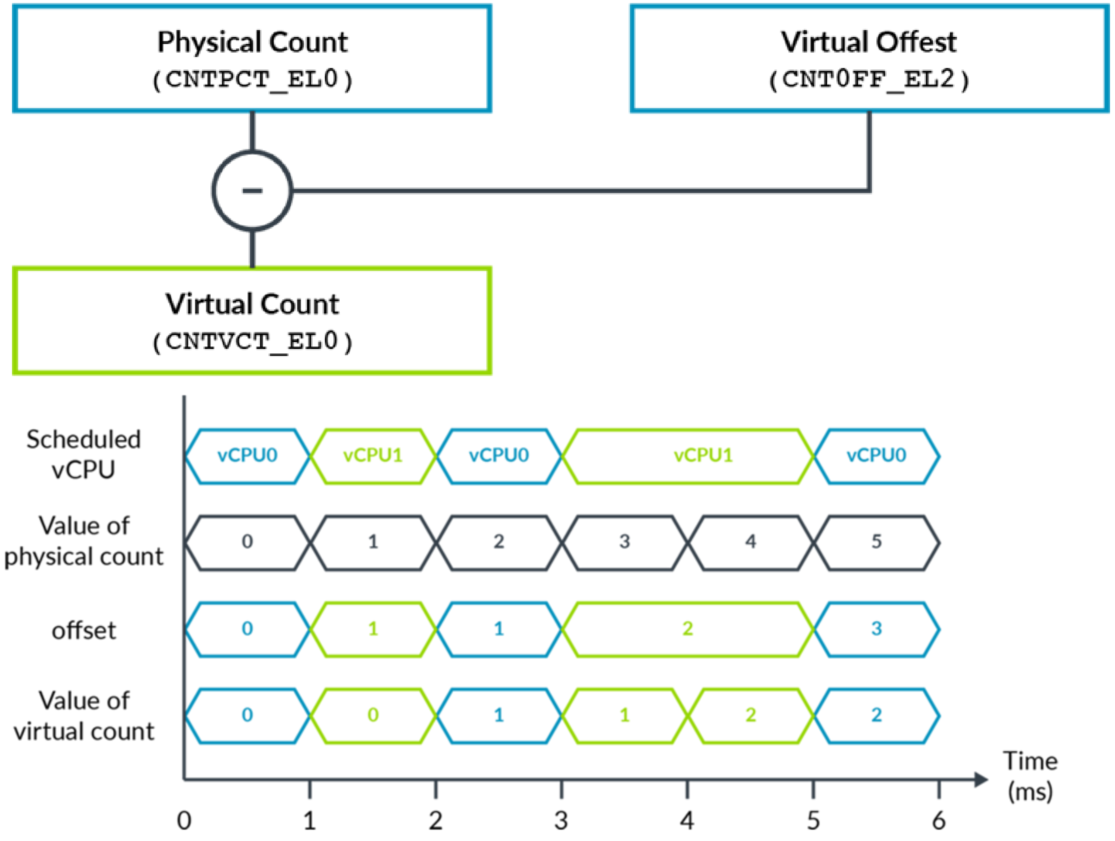
1. 物理外设产生中断；
2. Hypervisor收到该中断，判断中断所属；
3. Hypervisor通过配置GICH，为vCPU注入中断；
4. Hypervisor通过eret返回EL1，vCPU开始执行；
5. GIC向vCPU发出虚拟中断；



```
interrupt_handler:
    identify_interrupt()
    forward_virtual_interrupt()
    return_to_vcpu()
```

ARMv8 时钟虚拟化

- 所有处理器
- 关键寄存器：
  - 物理时钟：CNTPCT_EL0 (wall clock)
  - 虚拟时钟：CNTVCT_EL0
  - **虚拟时钟偏移：CNTVOFF_EL2**
- Hypervisor的工作
  - 维护虚拟机的虚拟时钟；
  - 适时更新虚拟时钟偏移寄存器；



Physical Count
(CNTPCT_EL0)

Virtual Offest
(CNT0FF_EL2)

Virtual Count
(CNTVCT_EL0)

	0–1	1–2	2–3	3–4	4–5	5–6
Scheduled vCPU	vCPU0	vCPU1	vCPU0	vCPU1		vCPU0
Value of physical count	0	1	2	3	4	5
offset	0	1	1	2		3
Value of virtual count	0	0	1	1	2	2

ARMv8 虚拟化的其他内容

- 虚拟化的开销
  - 地址翻译；
  - TLB；
  - 中断延迟；
  - 上下文切换；
- 嵌套虚拟化
- Type-2 Host虚拟化：VHE扩展
- Secure World虚拟化

谢谢大家！

分享人：莫策
2023年11月6日星期一