Basic Concepts of Virtualization

Junming Liu

Agenda

- ❑ Background
- ❑ Aspects in Control Register
- ❑ Aspects in MSR Register
- ❑ Aspects in memory virtualization
- ❑ Aspects in IO
- ❑ Summary

Agenda

- ❑ Background
- ❑ Aspects in Control Register
- ❑ Aspects in MSR Register
- ❑ Aspects in memory virtualization
- ❑ Aspects in IO
- ❑ Summary

Three different context

- Physical context
- Shadow context
- Virtual context

《系统虚拟化》

- 提了“虚拟特权资源”与“影子特权资源”的概念

Agenda

- ❑ Background
- ❑ Aspects in Control Register
- ❑ Aspects in MSR Register
- ❑ Aspects in memory virtualization
- ❑ Aspects in IO
- ❑ Summary

CR4 register

- **Physical context** (the physical value in root mode)

vmcs VMX_HOST_CR4 field.

- **Shadow context** (the physical value in non-root mode)

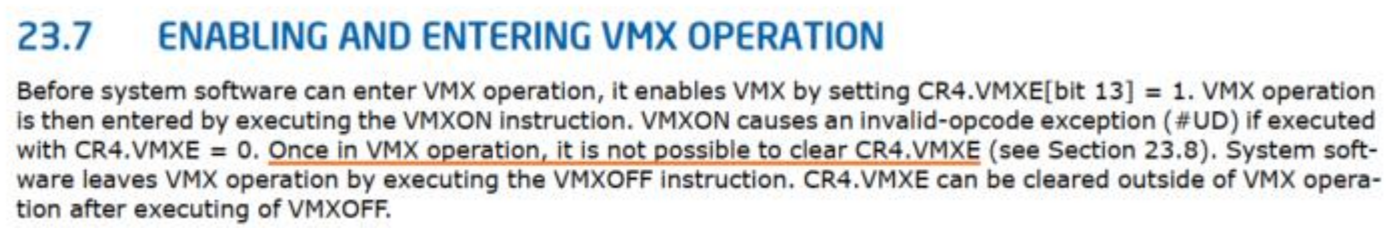
For the bit owned by guest, the value is derived from guest. For the bit owned by host, the value is derived from vmcs VMX_GUEST_CR4 field.

- **Virtual context**

For the bit owned by guest, the value is derived from guest. For the bit owned by host, the value is derived from vmcs VMX_CR4_READ_SHADOW field

Rethinking

- Why CR4.VMXE is TRAP_AND_EMULATE_BITS[1]



23.7 ENABLING AND ENTERING VMX OPERATION

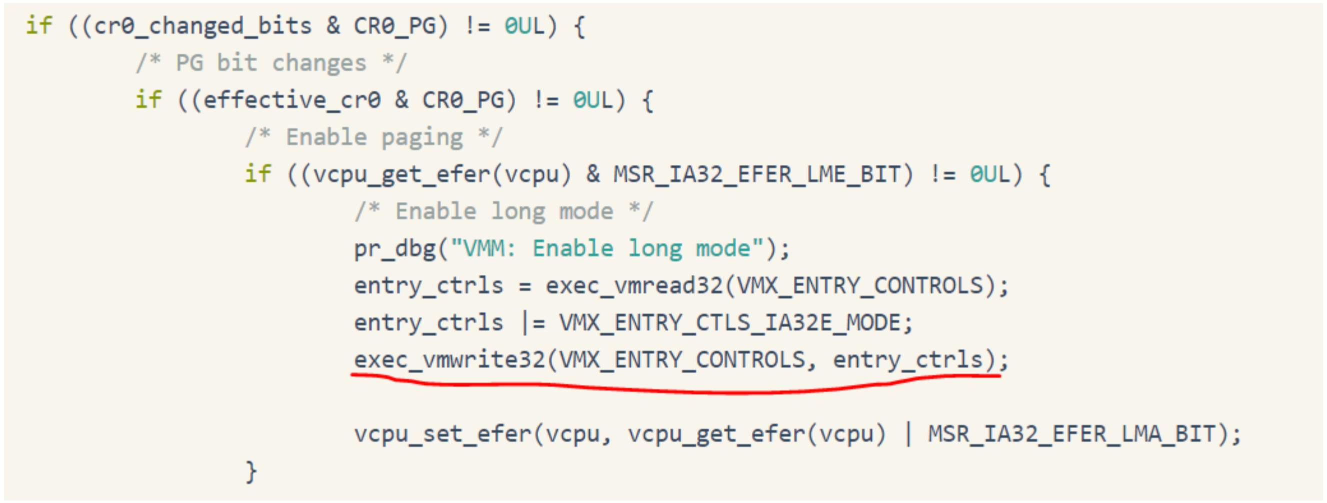
Before system software can enter VMX operation, it enables VMX by setting CR4.VMXE[bit 13] = 1. VMX operation is then entered by executing the VMXON instruction. VMXON causes an invalid-opcode exception (#UD) if executed with CR4.VMXE = 0. Once in VMX operation, it is not possible to clear CR4.VMXE (see Section 23.8). System software leaves VMX operation by executing the VMXOFF instruction. CR4.VMXE can be cleared outside of VMX operation after executing of VMXOFF.

Need to ensure vmxe bit is set in shadow context

[1] ACRN mainline virtual_cr.c

Rethinking

- Why CR0.PG is TRAP_AND_PASSTHRU_BITS [1]



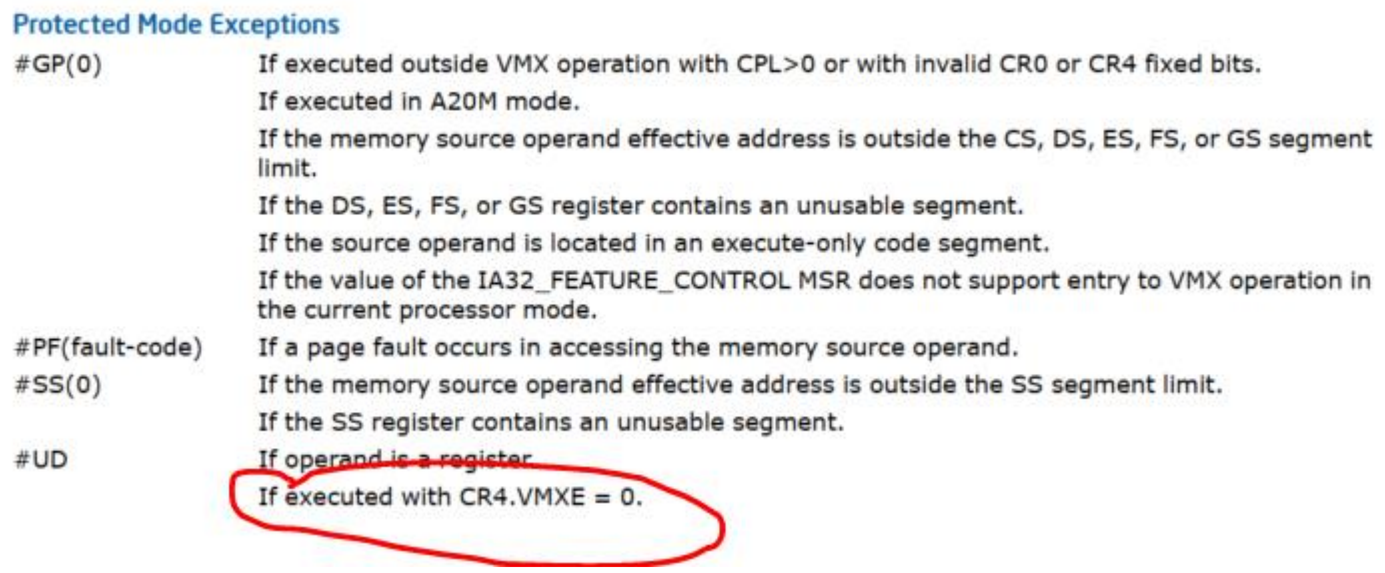
```
if ((cr0_changed_bits & CR0_PG) != 0UL) {
        /* PG bit changes */
        if ((effective_cr0 & CR0_PG) != 0UL) {
                /* Enable paging */
                if ((vcpu_get_efer(vcpu) & MSR_IA32_EFER_LME_BIT) != 0UL) {
                        /* Enable long mode */
                        pr_dbg("VMM: Enable long mode");
                        entry_ctrls = exec_vmread32(VMX_ENTRY_CONTROLS);
                        entry_ctrls |= VMX_ENTRY_CTLS_IA32E_MODE;
                        exec_vmwrite32(VMX_ENTRY_CONTROLS, entry_ctrls);

                        vcpu_set_efer(vcpu, vcpu_get_efer(vcpu) | MSR_IA32_EFER_LMA_BIT);
                }
```

In non-root mode, hardware couldn't update vmcs VMX_ENTRY_CONTROLS field.
Update this field in root mode.

[1] ACRN mainline virtual_cr.c

VMXON instruction



Protected Mode Exceptions

#GP(0)	If executed outside VMX operation with CPL>0 or with invalid CR0 or CR4 fixed bits.
	If executed in A20M mode.
	If the memory source operand effective address is outside the CS, DS, ES, FS, or GS segment limit.
	If the DS, ES, FS, or GS register contains an unusable segment.
	If the source operand is located in an execute-only code segment.
	If the value of the IA32_FEATURE_CONTROL MSR does not support entry to VMX operation in the current processor mode.
#PF(fault-code)	If a page fault occurs in accessing the memory source operand.
#SS(0)	If the memory source operand effective address is outside the SS segment limit.
	If the SS register contains an unusable segment.
#UD	If operand is a register.
	If executed with CR4.VMXE = 0.

That means the physical value of CR4.VMXE.
It’s **shadow context** in non-root mode.

Conclusion

- If one bit has restriction in VMX operation or needs to do some operations in root mode, It’s better to trap(owned by host) this bit.

Agenda

- ❑ Background
- ❑ Aspects in Control Register
- ❑ Aspects in MSR Register
- ❑ Aspects in memory virtualization
- ❑ Aspects in IO
- ❑ Summary

Hardware-assisted save and restore MSR

- VMCS field MSR
- MSR area
  - A VMM may specify lists of MSRs to be stored and loaded on VM exits
  - A VMM may specify a list of MSRs to be loaded on VM entries

We can get physical context and shadow context from above information block.
virtual context may diff with **shadow context,**
it's MSR specific, **it depends on HV**.

VMCS field MSR(MSR_IA32_EFER)

- VMX_HOST_IA32_EFER_FULL
- VMX_GUEST_IA32_EFER_FULL
- HV has ensured shadow context equals to virtual context, so don’t need to intercept RDMSR

MSR area(MSR_IA32_TSC_AUX)

- In ACRN, virtual context equals to shadow context

Non hardware-assisted save and restore MSR

- shadow context = physical context
- MSR_IA32_EXT_XAPICID
- MSR_IA32_TIME_STAMP_COUNTER
- MSR_IA32_EXT_APIC_VERSION

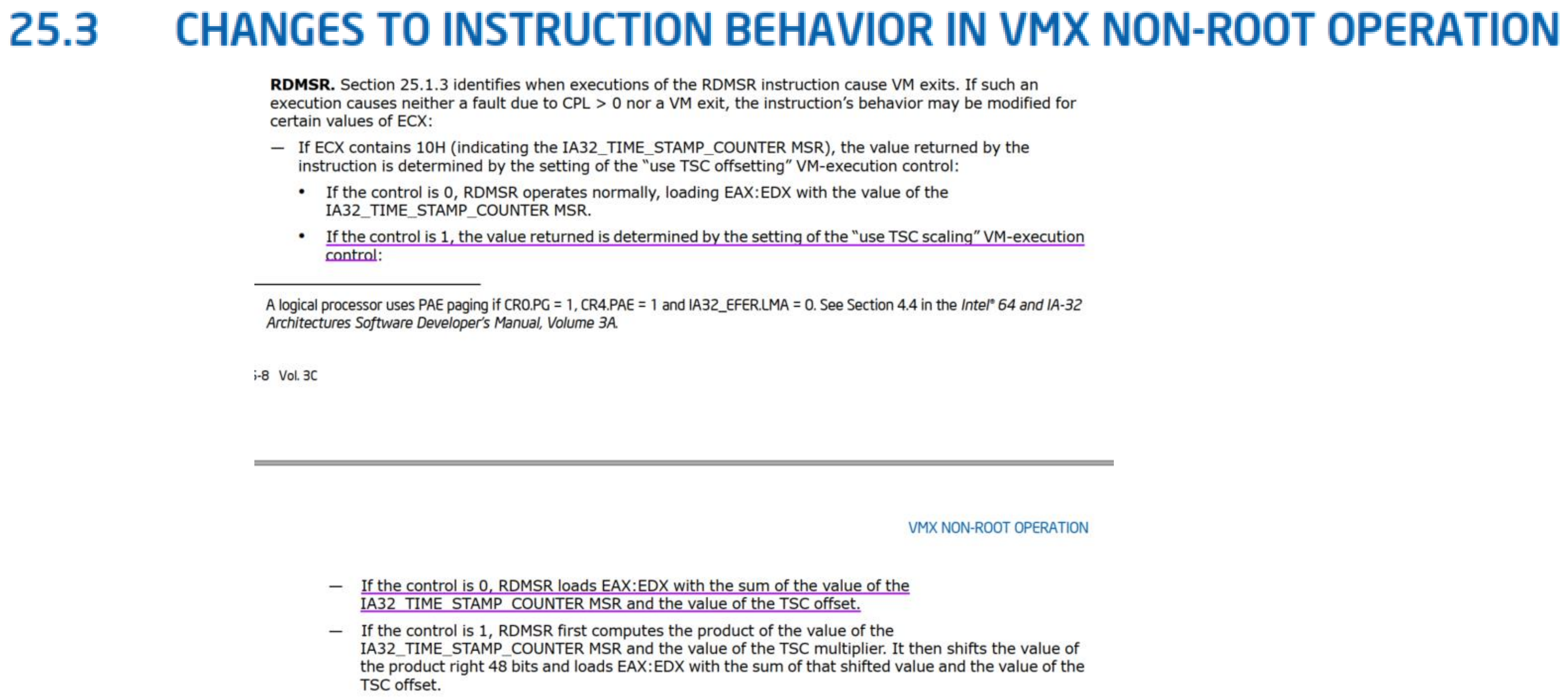
virtual context may diff with **shadow context**,
it's MSR specific.

MSR_IA32_EXT_XAPICID

- For isolation, shadow context diff with virtual context
- Need to intercept RDMSR
- Virtual context value is set by HV

MSR_IA32_TIME_STAMP_COUNTER

- For isolation, shadow context diff with virtual context
- Don't need to intercept RDMSR



25.3 CHANGES TO INSTRUCTION BEHAVIOR IN VMX NON-ROOT OPERATION

RDMSR. Section 25.1.3 identifies when executions of the RDMSR instruction cause VM exits. If such an execution causes neither a fault due to CPL > 0 nor a VM exit, the instruction's behavior may be modified for certain values of ECX:

— If ECX contains 10H (indicating the IA32_TIME_STAMP_COUNTER MSR), the value returned by the instruction is determined by the setting of the "use TSC offsetting" VM-execution control:

- If the control is 0, RDMSR operates normally, loading EAX:EDX with the value of the IA32_TIME_STAMP_COUNTER MSR.
- If the control is 1, the value returned is determined by the setting of the "use TSC scaling" VM-execution control:

A logical processor uses PAE paging if CR0.PG = 1, CR4.PAE = 1 and IA32_EFER.LMA = 0. See Section 4.4 in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A.*

;-8 Vol. 3C

VMX NON-ROOT OPERATION

— If the control is 0, RDMSR loads EAX:EDX with the sum of the value of the IA32_TIME_STAMP_COUNTER MSR and the value of the TSC offset.

— If the control is 1, RDMSR first computes the product of the value of the IA32_TIME_STAMP_COUNTER MSR and the value of the TSC multiplier. It then shifts the value of the product right 48 bits and loads EAX:EDX with the sum of that shifted value and the value of the TSC offset.

MSR_IA32_EXT_APIC_VERSION

- shadow context = virtual context

The motivation of MSR area mechanism

- Let’s take MSR_IA32_TSC_AUX as an example. We need its virtual context is different from physical context. If shadow context equals physical context, Each time guest wants to read this MSR, VM Exit needs to happen(read MSR maybe a frequent operation). This method may impact system performance.
- So it is better to use guest/host field in the MSR area for performance. BTW, use guest/host field in the MSR area, the shadow context can be different from the physical context. So we don’t need to trap MSR_IA32_TSC_AUX. So, MSR area mechanism is to provide host/guest field to the MSR, the hypervisor can use this host/guest field to separate shadow context from physical context.

Agenda

- ❑ Background
- ❑ Aspects in Control Register
- ❑ Aspects in MSR Register
- ❑ Aspects in memory virtualization
- ❑ Aspects in IO
- ❑ Summary

EPT

- **Shadow context**

HPA

- **Virtual context**

GPA

- **Physical context**

HPA, but may need the relationship between GPA,HVA and HPA to build EPT table

Agenda

- ❑ Background
- ❑ Aspects in Control Register
- ❑ Aspects in MSR Register
- ❑ Aspects in memory virtualization
- ❑ Aspects in IO
- ❑ Summary

MSI interrupt W/O interrupt remapping

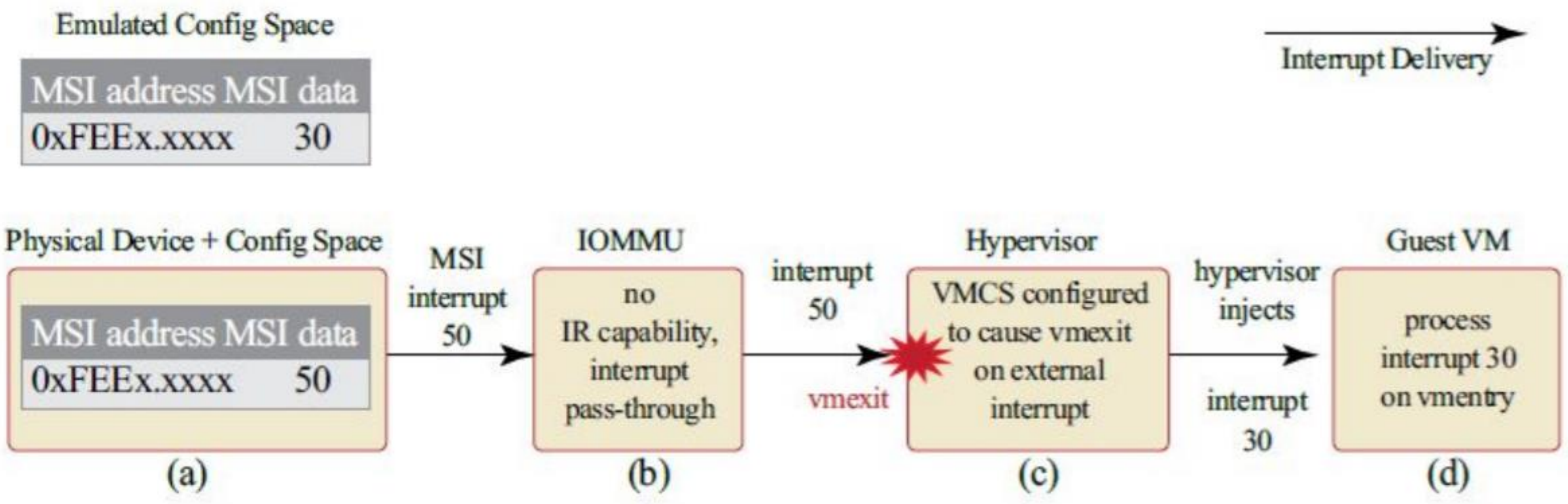


Figure 6.16: MSI interrupt delivery without interrupt remapping support.

MSI interrupt With interrupt remapping

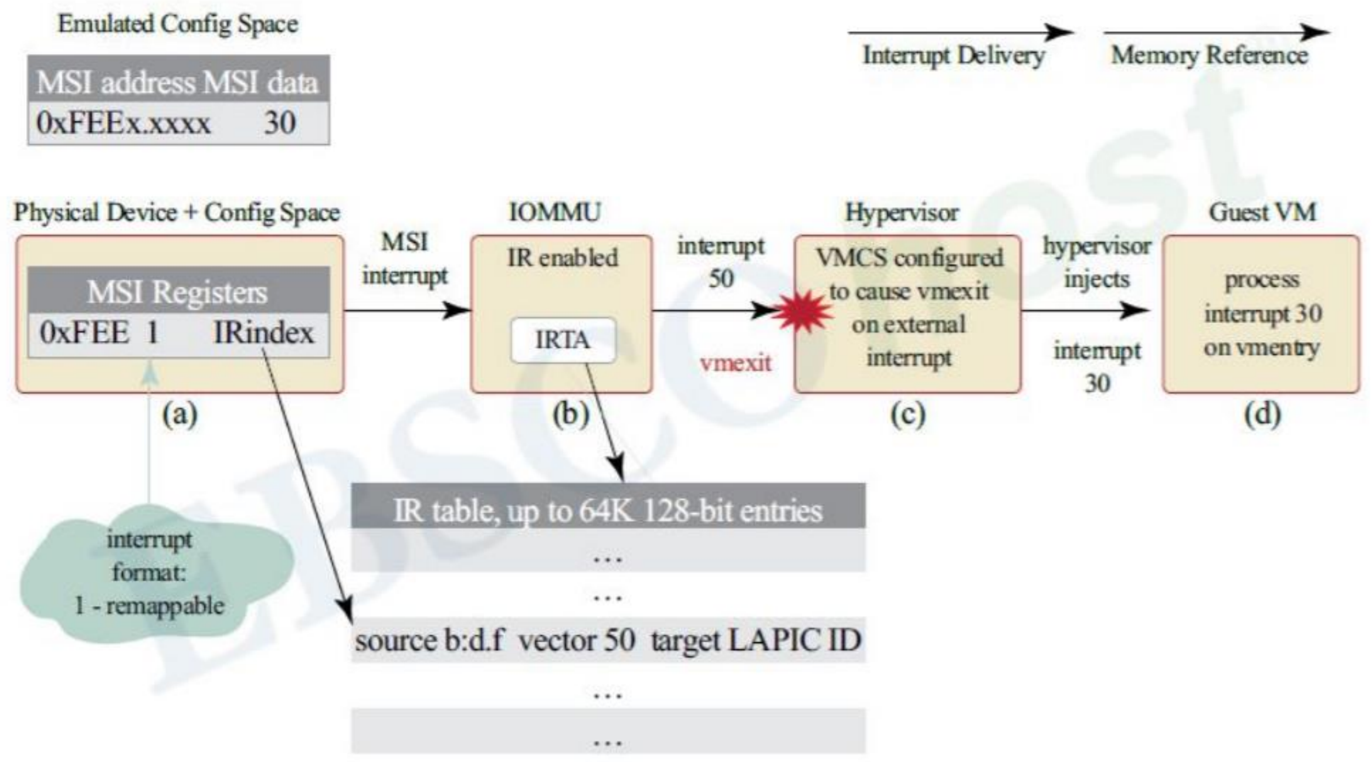


Figure 6.17: MSI interrupt delivery with interrupt remapping support. (IRindex is denoted "interrupt_index" in the VT-d specification.)

MSI interrupt

- In the previous two cases, shadow context equals to physical context
- virtual context diffs with shadow context

Agenda

- ❑ Background
- ❑ Aspects in Control Register
- ❑ Aspects in MSR Register
- ❑ Aspects in memory virtualization
- ❑ Aspects in IO
- ❑ Summary

Summary

- Must be aware of the 3 different context for privileged resource
- Know what is done by hardware, what is done by software
- Know what is done in root mode, what is done in non-root mode