Interrupt and Interrupt Virtualization

Junming Liu

1/15/2022

Outline

- Terms
- Introduction to interrupt
- Key point
- Interrupt virtualization

Outline

- Terms
- Introduction to interrupt
- Key point
- Interrupt virtualization

Basic

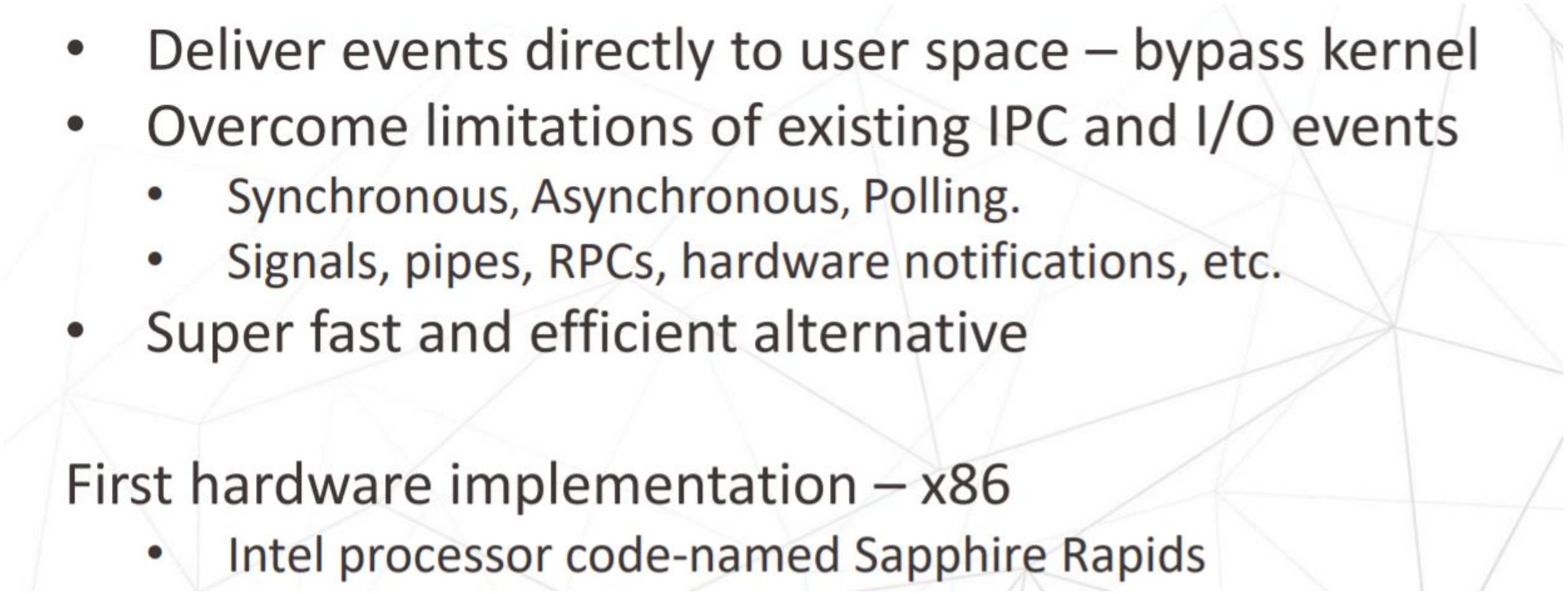
- PIC
- LAPIC
- I/O APIC
  - GSI
- MSI/MSI-x

Virtualization

- FlexPriority
- APICv
- VT-x Posted Interrupt
- VT-d Interrupt Remapping
- VT-d Posted Interrupt
- IPIv
- LAPIC pass-thru

New

- User Interrupt



- Deliver events directly to user space – bypass kernel
- Overcome limitations of existing IPC and I/O events
  - Synchronous, Asynchronous, Polling.
  - Signals, pipes, RPCs, hardware notifications, etc.
- Super fast and efficient alternative

First hardware implementation – x86

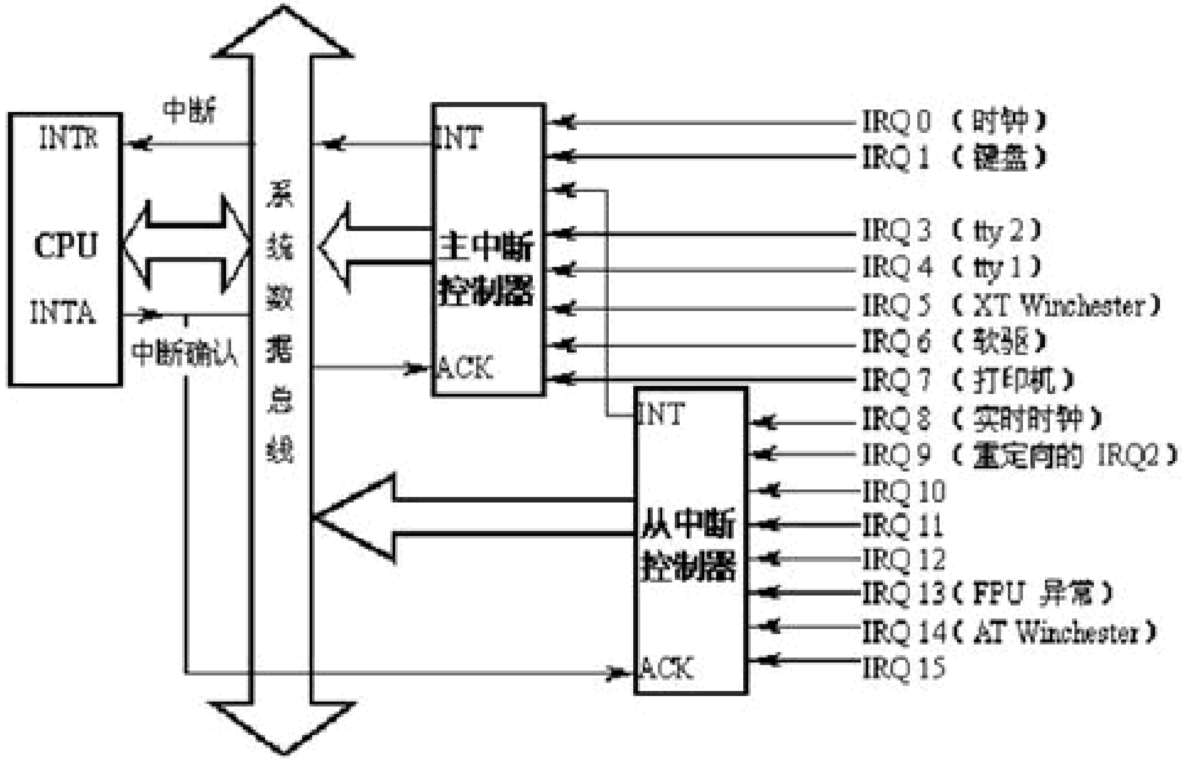
- Intel processor code-named Sapphire Rapids

https://linuxplumbersconf.org/event/11/contributions/985/attachments/756/1417/User_Interrupts_LPC_2021.pdf
https://lore.kernel.org/lkml/20210913200132.3396598-1-sohil.mehta@intel.com/
https://software.intel.com/content/www/us/en/develop/download/intel-architecture-instruction-set-extensions-programming-reference.html

Outline

- Terms
- Introduction to interrupt
- Key point
- Interrupt virtualization

PIC


中断
INTR
CPU
INTA
中断确认
系统数据总线
INT
主中断控制器
ACK
INT
从中断控制器
ACK
IRQ 0 （时钟）
IRQ 1 （键盘）
IRQ 3 （tty 2）
IRQ 4 （tty 1）
IRQ 5 （XT Winchester）
IRQ 6 （软驱）
IRQ 7 （打印机）
IRQ 8 （实时时钟）
IRQ 9 （重定向的 IRQ2）
IRQ 10
IRQ 11
IRQ 12
IRQ 13（FPU 异常）
IRQ 14（AT Winchester）
IRQ 15

APIC

- Compare with PIC, it support multiprocessor
- CPU side: LAPIC
- Discrete chip: I/O APIC

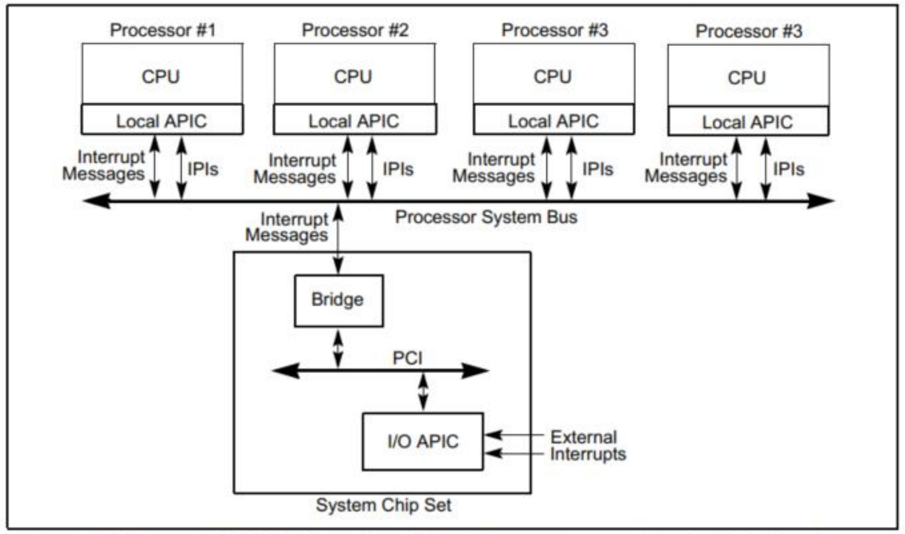


Figure 10-2. Local APICs and I/O APIC When Intel Xeon Processors Are Used in Multiple-Processor Systems

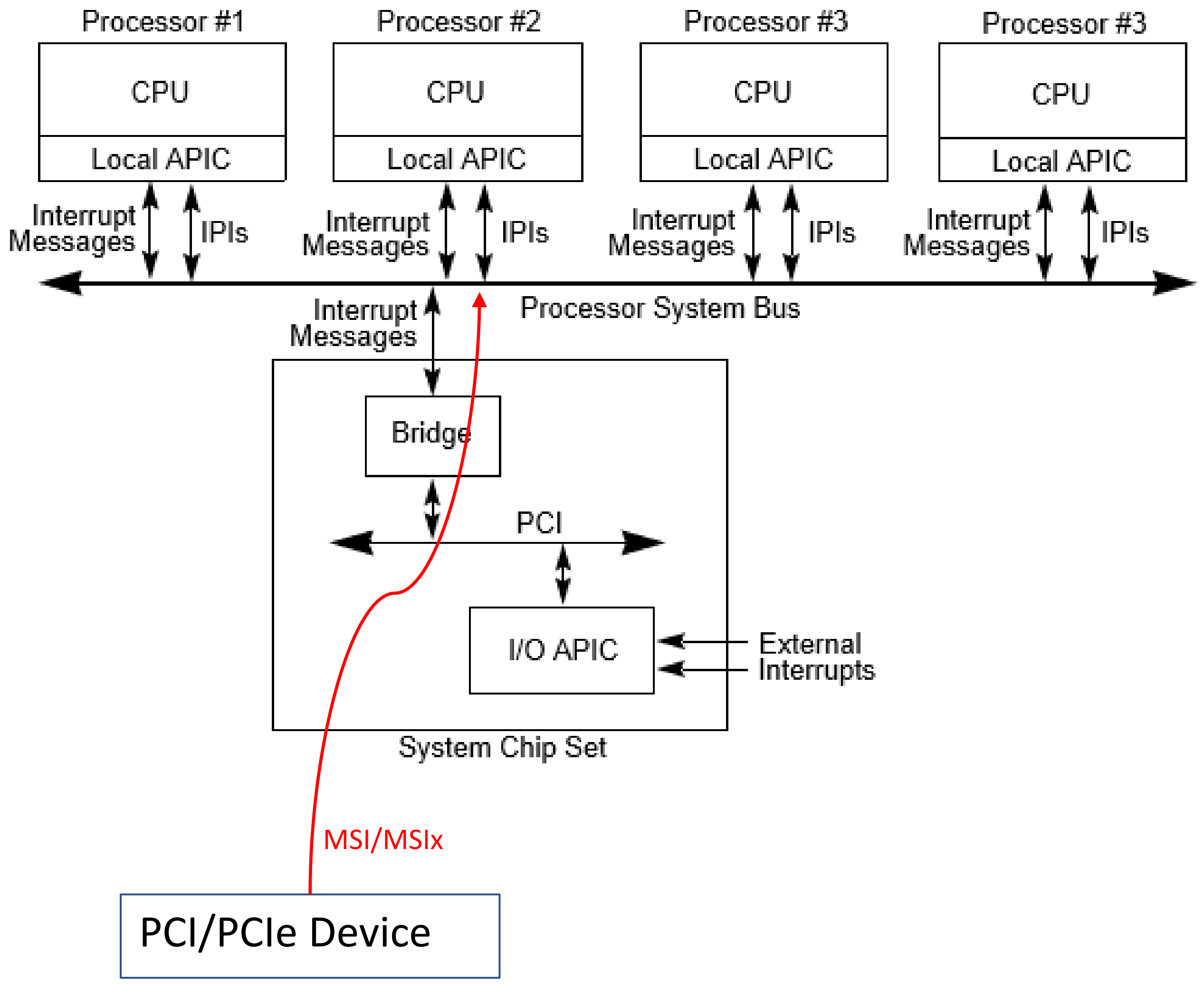
LAPIC

- LVT
- ICR
- Spec
  - https://tcbbd.moe/ref-and-spec/intel-sdm/sdm-basic-ch10/
  - Intel SDM Chapter 10: APIC
- Source code
  - KVM unit test: x86/apic.c
  - KVM: arch/x86/kvm/lapic.c
  - ACRN hypervisor: hypervisor/arch/x86/lapic.c

I/O APIC

- Spec
  - https://pdos.csail.mit.edu/6.828/2016/readings/ia32/ioapic.pdf
- Source code
  - KVM unit test: x86/ioapic.c
  - KVM: arch/x86/kvm/ioapic.c
  - ACRN hypervisor: hypervisor/arch/x86/ioapic.c

MSI/MSI-x


Processor #1
Processor #2
Processor #3
Processor #3
CPU
CPU
CPU
CPU
Local APIC
Local APIC
Local APIC
Local APIC
Interrupt Messages
IPIs
Interrupt Messages
IPIs
Interrupt Messages
IPIs
Interrupt Messages
IPIs
Processor System Bus
Interrupt Messages
Bridge
PCI
I/O APIC
External Interrupts
System Chip Set
MSI/MSIx
PCI/PCIe Device

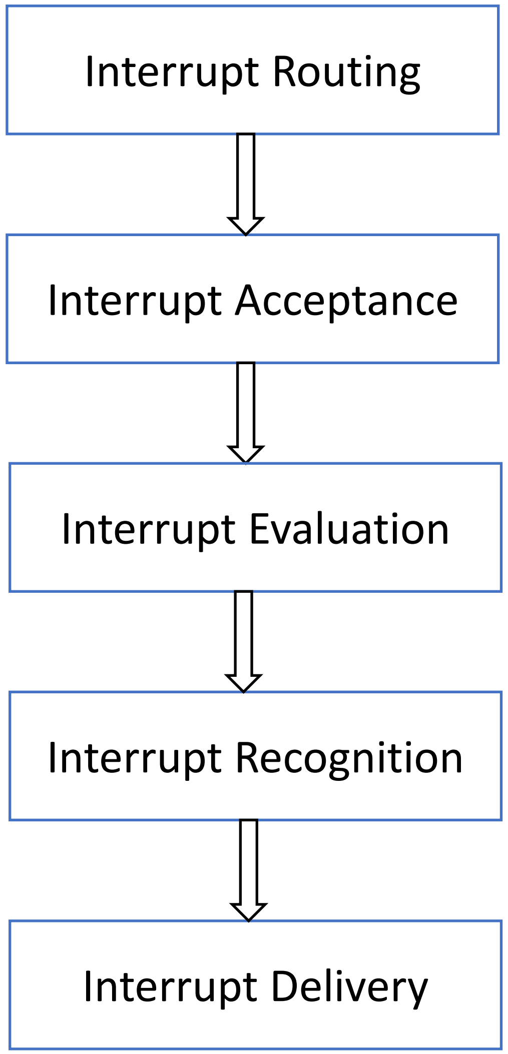
MSI/MSI-x

- Bypass I/O APIC
- Spec
  - Intel SDM Chapter 10: APIC --- 10.11 MESSAGE SIGNALLED INTERRUPTS
  - PCIe SPEC
  - https://tcbbd.moe/ref-and-spec/pci/
- Source code
  - KVM unit test: lib/pci.c

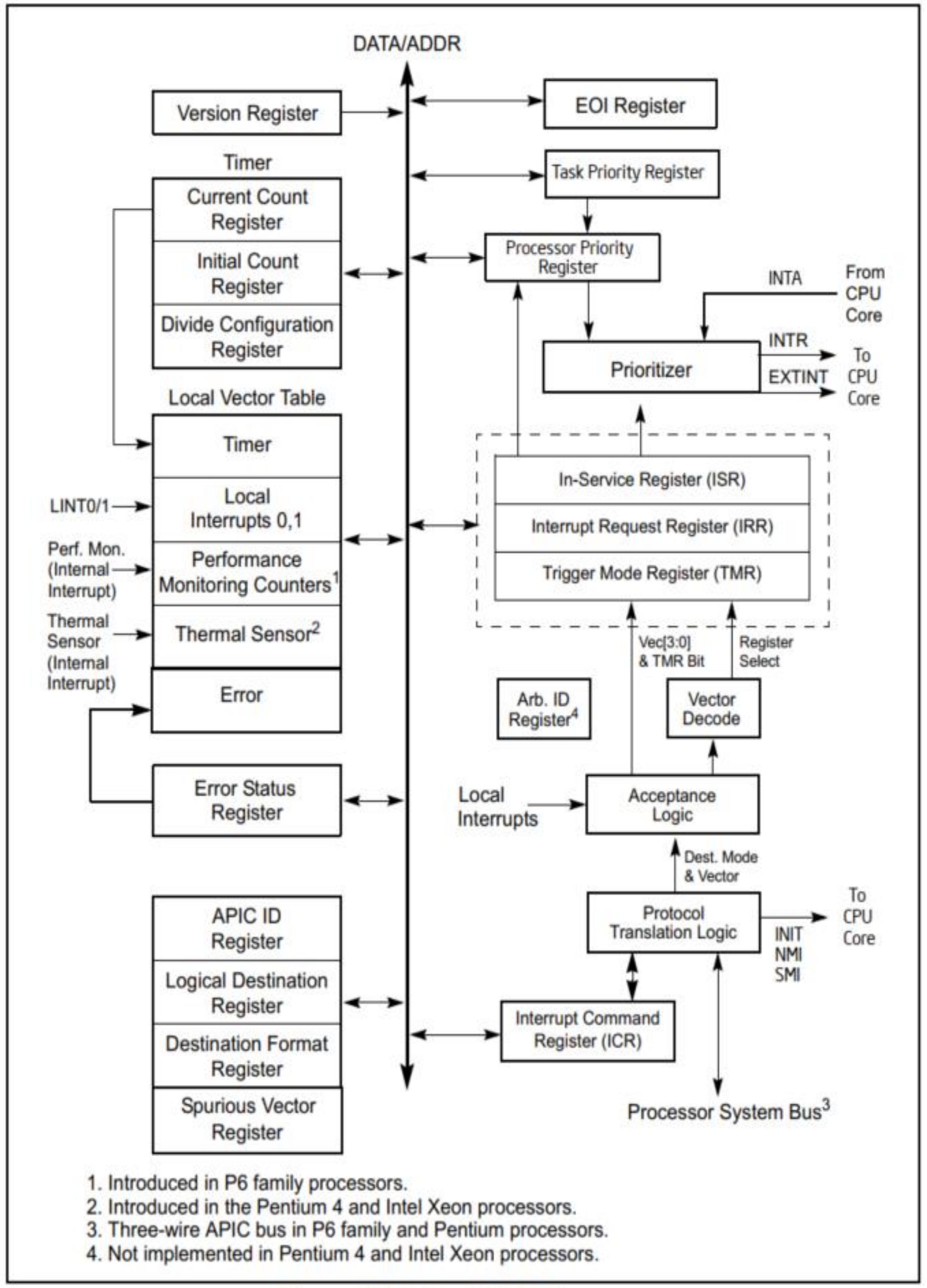
Outline

- Terms
- Introduction to interrupt
- Key point
- Interrupt virtualization

Steps


Interrupt Routing
Interrupt Acceptance
Interrupt Evaluation
Interrupt Recognition
Interrupt Delivery

Overview



1. Introduced in P6 family processors.
2. Introduced in the Pentium 4 and Intel Xeon processors.
3. Three-wire APIC bus in P6 family and Pentium processors.
4. Not implemented in Pentium 4 and Intel Xeon processors.

Figure 10-4. Local APIC Structure

Terms

- IRR(Interrupt Request Register)
- ISR(Interrupt Service Register)
- IRRV(Max Vector in IRR)
- ISRV(Max Vector in ISR)
- EOI(End Of Interrupt)
- PPR(Processor Priority Register)

Interrupt Routing

- Who?
  - IPI
    - source CPU writes ICR register
  - External interrupt
    - IOAPIC
    - MSI/MSI-x
- Where?
  - To which LAPIC ID
- What?
  - Vector
- How?

Interrupt Acceptance

- Set the IRR reg
- Pseudocode details
    - For the received Vector, IRR[Vector] $\leftarrow$ 1
    - IRRV $\leftarrow$ highest index of bit in IRR

Interrupt Evaluation

- IRRV > PPR[7:4], then recognize an interrupt

Interrupt Recognition

- Once an interrupt is recognized, the processor may deliver it

Interrupt Delivery

- Condition: RFLAGS.IF = 1
- Pseudocode details
  - Vector ← IRRV
  - ISR[Vector] ← 1
  - ISRV ← Vector
  - PPR ← Vector & F0H
  - IRR[Vector] ← 0
  - If any bits set in IRR, then IRRV ← highest index of bit in IRR, otherwise, IRRV ← 0
  - Deliver interrupt with Vector through IDT
  - Cease recognition of any pending interrupt

Interrupt Handler

- Do something you want
- Write EOI register
  - Vector ← ISRV
  - ISR[Vector] ← 0
  - If any bits set in ISR, then ISRV ← highest index of bit in ISR, otherwise, ISRV ← 0
  - Update PPR register
- Execute **IRET** instruction

Outline

- Terms
- Introduction to interrupt
- Key point
- Interrupt virtualization

FlexPriority

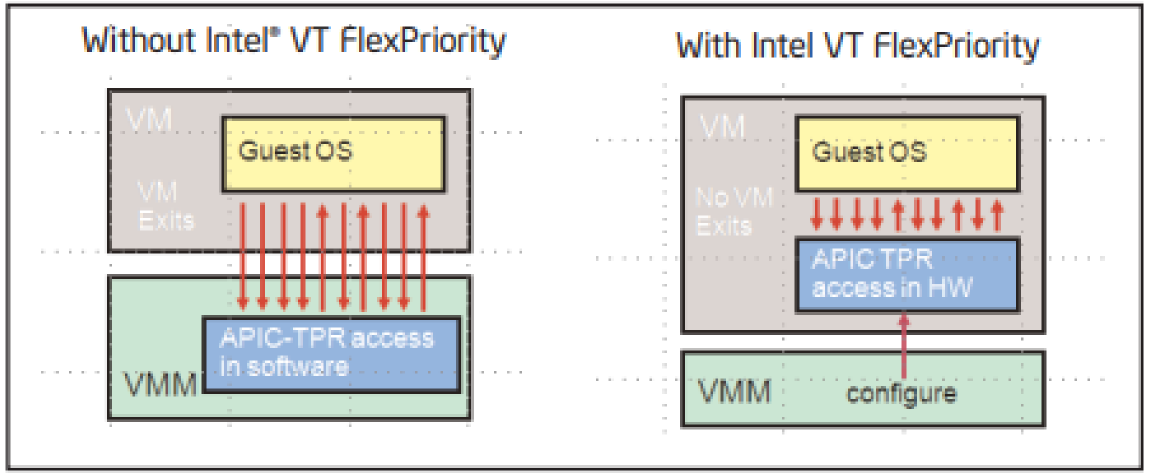
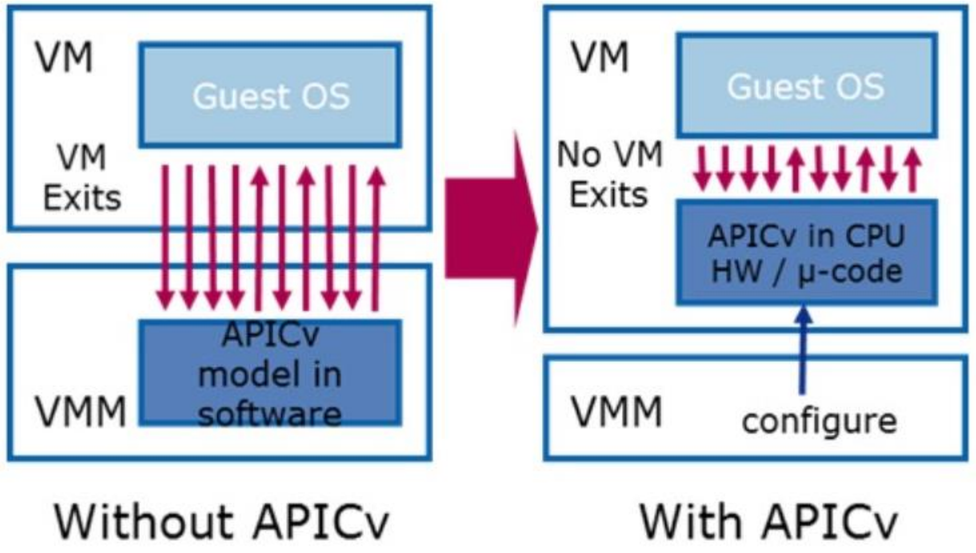

Without Intel® VT FlexPriority
VM
Guest OS
VM Exits
APIC-TPR access in software
VMM
With Intel VT FlexPriority
VM
Guest OS
No VM Exits
APIC TPR access in HW
VMM
configure

Figure 1. Reduction of EXITs with Intel® VT FlexPriority

https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/virtualization-enabling-intel-virtualization-technology-features-and-benefits-paper.pdf
http://liujunming.top/2022/01/12/Introduction-to-VT-x-FlexPriority/

APICv


VM
Guest OS
VM Exits
APICv model in software
VMM
Without APICv
VM
Guest OS
No VM Exits
APICv in CPU HW / μ-code
VMM
configure
With APICv

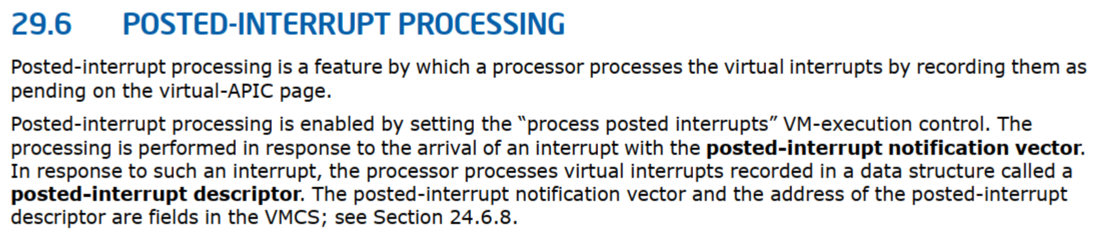
APICv

- Writes to the local APIC have side effects
  - APIC-write emulation
  - APIC-write VM exit
- Avoid Interrupt-Window Exiting
- Source code
  - https://lore.kernel.org/kvm/1359080331-22872-1-git-send-email-yang.z.zhang@intel.com/

Intel SDM Chapter 29: APIC Virtualizaton & Virtual Interrupts
https://tcbbd.moe/ref-and-spec/intel-sdm/sdm-vmx-ch29/
http://liujunming.top/2020/10/08/The-motivation-of-APICv/

VT-x Posted Interrupt

- Avoid VM Exit when Interrupt Acceptance
- Source: CPU
- Source code
    - https://lore.kernel.org/kvm/1365679516-13125-1-git-send-email-yang.z.zhang@intel.com/



29.6 POSTED-INTERRUPT PROCESSING

Posted-interrupt processing is a feature by which a processor processes the virtual interrupts by recording them as pending on the virtual-APIC page.

Posted-interrupt processing is enabled by setting the "process posted interrupts" VM-execution control. The processing is performed in response to the arrival of an interrupt with the **posted-interrupt notification vector**. In response to such an interrupt, the processor processes virtual interrupts recorded in a data structure called a **posted-interrupt descriptor**. The posted-interrupt notification vector and the address of the posted-interrupt descriptor are fields in the VMCS; see Section 24.6.8.

http://liujunming.top/2020/10/07/Introduction-to-Posted-interrupt/
http://liujunming.top/2021/11/14/VT-x-Posted-Interrupt-Code-Analysis/
http://liujunming.top/2022/01/11/How-to-verify-the-difference-between-posted-interrupt-and-event-injection/

Verification of VT-x Posted Interrupt

- http://liujunming.top/2022/01/11/How-to-verify-the-difference-between-posted-interrupt-and-event-injection/

VT-d Interrupt Remapping

W/O Interrupt Remapping

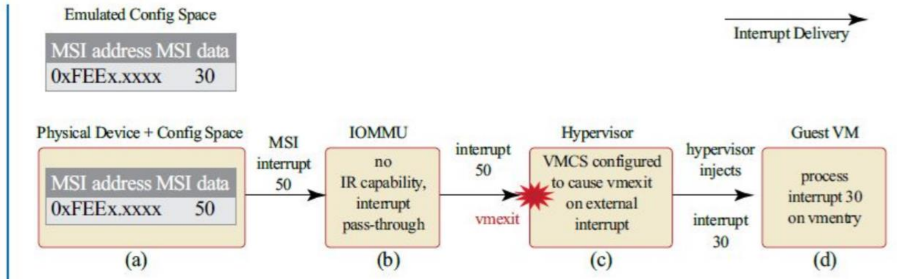


Figure 6.16: MSI interrupt delivery without interrupt remapping support.

With Interrupt Remapping

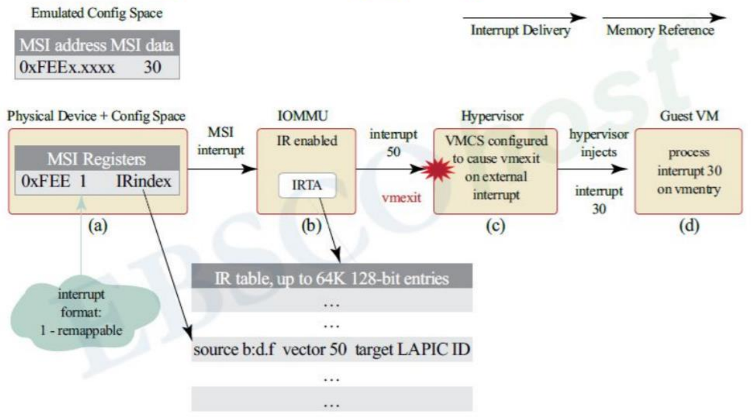


Figure 6.17: MSI interrupt delivery with interrupt remapping support. (IRindex is denoted "interrupt_index" in the VT-d specification.)

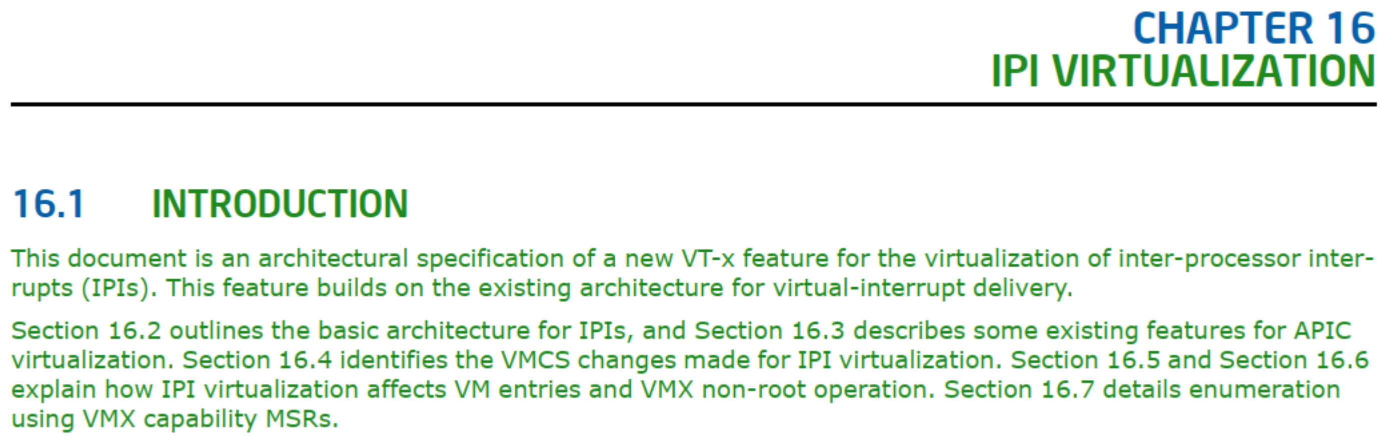
http://liujunming.top/2020/10/10/VT-d-Interrupt-Remapping/

VT-d Posted Interrupt

- Avoid VM Exit when Interrupt Acceptance
- Source: Device
- Source code
  - https://lore.kernel.org/kvm/1442586596-5920-1-git-send-email-feng.wu@intel.com/

IPIv

- Hardware virtualization for Interrupt Routing
- Source: CPU
- Source code
  - https://lore.kernel.org/kvm/20211231142849.611-1-guang.zeng@intel.com/



CHAPTER 16
IPI VIRTUALIZATION

16.1 INTRODUCTION

This document is an architectural specification of a new VT-x feature for the virtualization of inter-processor interrupts (IPIs). This feature builds on the existing architecture for virtual-interrupt delivery.

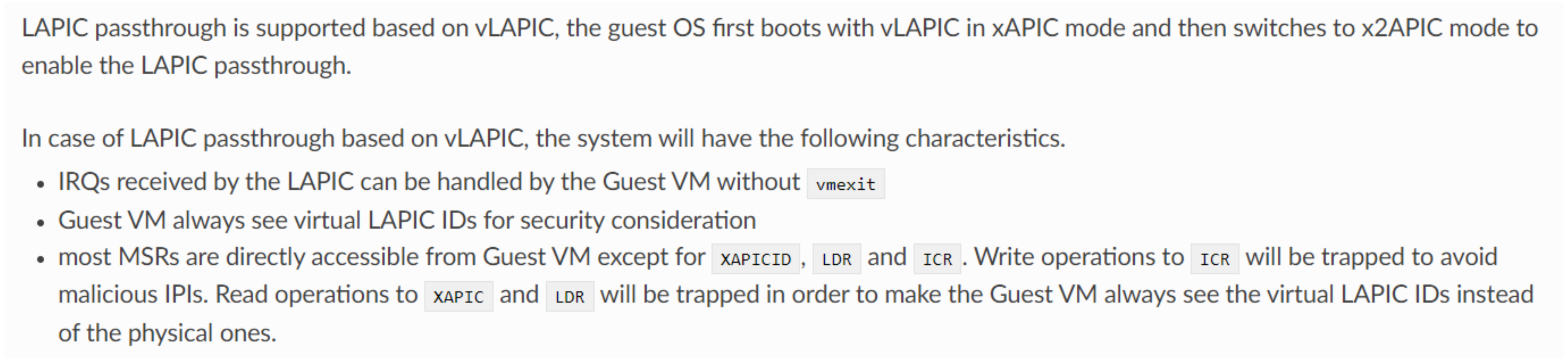
Section 16.2 outlines the basic architecture for IPIs, and Section 16.3 describes some existing features for APIC virtualization. Section 16.4 identifies the VMCS changes made for IPI virtualization. Section 16.5 and Section 16.6 explain how IPI virtualization affects VM entries and VMX non-root operation. Section 16.7 details enumeration using VMX capability MSRs.

---

https://software.intel.com/content/www/us/en/develop/download/intel-architecture-instruction-set-extensions-programming-reference.html

LAPIC pass-thru

- Pass-thru LAPIC to guest



LAPIC passthrough is supported based on vLAPIC, the guest OS first boots with vLAPIC in xAPIC mode and then switches to x2APIC mode to enable the LAPIC passthrough.

In case of LAPIC passthrough based on vLAPIC, the system will have the following characteristics.

- IRQs received by the LAPIC can be handled by the Guest VM without `vmexit`
- Guest VM always see virtual LAPIC IDs for security consideration
- most MSRs are directly accessible from Guest VM except for `XAPICID`, `LDR` and `ICR`. Write operations to `ICR` will be trapped to avoid malicious IPIs. Read operations to `XAPIC` and `LDR` will be trapped in order to make the Guest VM always see the virtual LAPIC IDs instead of the physical ones.

https://projectacrn.github.io/latest/developer-guides/hld/hv-virt-interrupt.html#lapic-passthrough-based-on-vlapic
http://liujunming.top/2022/01/10/LAPIC-pass-thru-in-ACRN-hypervisor/