クラウドコンピューティングと要素技術
Technologies behind Cloud Computing

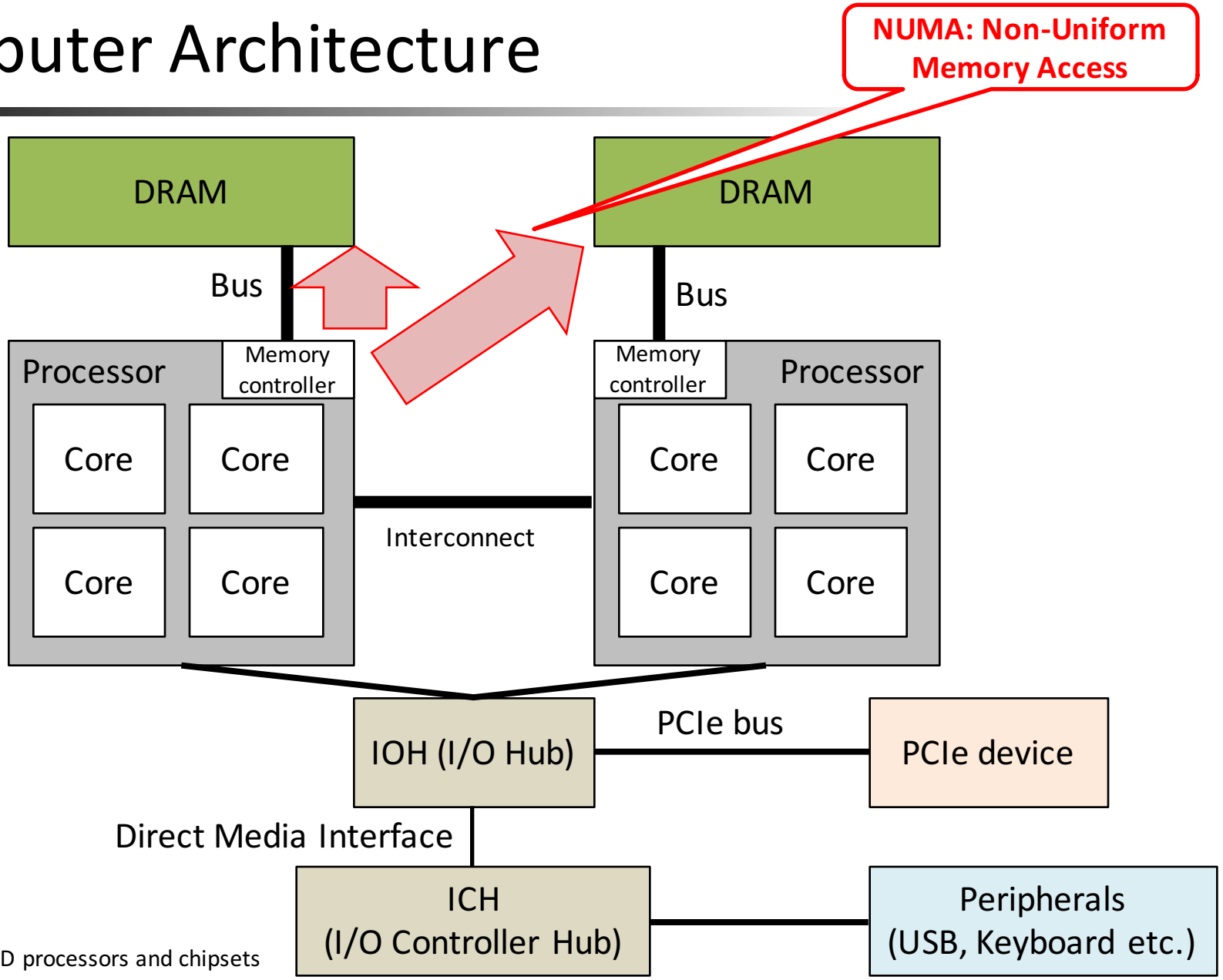
ソフトウェア・クラウド開発プロジェクト実践III

浅井大史

2016年4月22日

COMPUTER ARCHITECTURE AND OPERATING SYSTEM

Computer Architecture



※ Latest Intel and AMD processors and chipsets

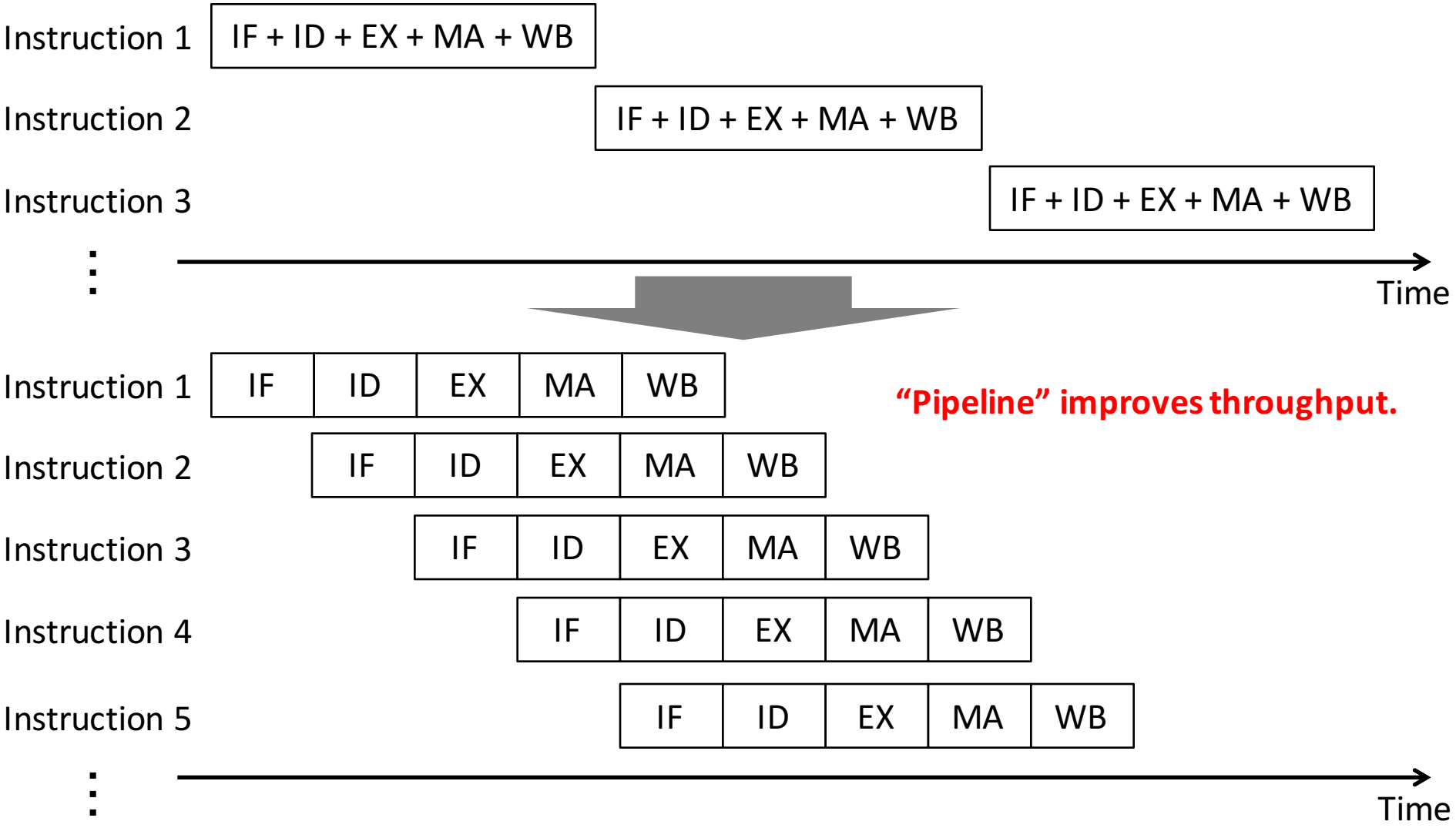
Instruction Set Architecture (ISA)

- Microprocessor instruction set architecture
  - CISC (Complex instruction set computing)
    - x86, x86-64 (Intel® 64, AMD 64)
  - RISC (Reduced instruction set computing)
    - SPARC, MIPS, POWER/PowerPC, ARM

Instruction Execution Cycle (abstracted)

1. Instruction fetch (IF)
   - Fetch instruction from instruction cache memory
2. Instruction decode (ID)
   - Decode the fetched instruction
   - Select registers corresponding to the instruction
3. Execute (EX)
   - Execute the instruction
4. Memory access (MA)
   - Load/Store access to memory
5. Write-back (WB)
   - Write-back the result of execution to registers

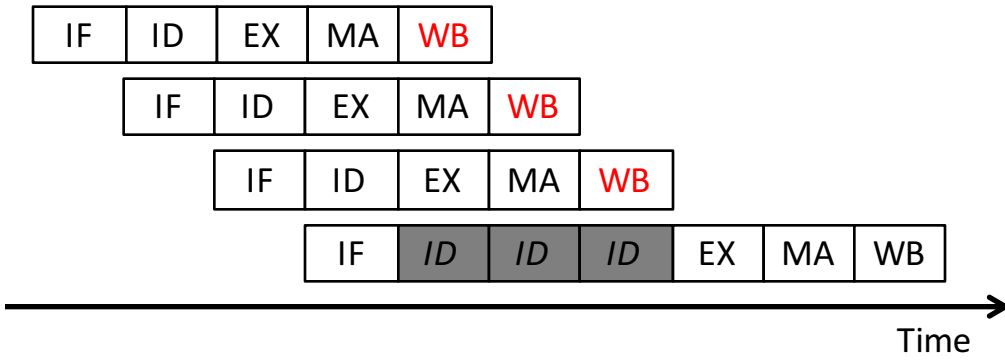
Pipeline



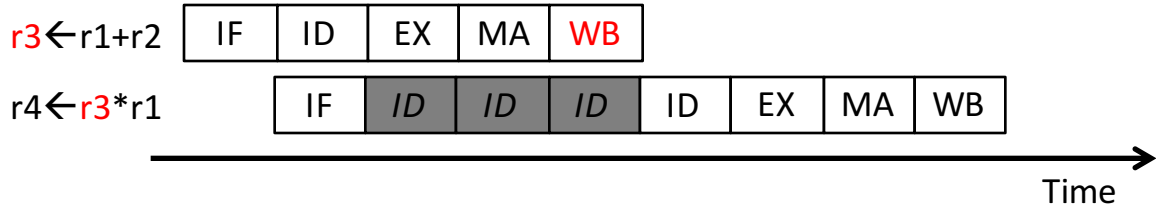
“Pipeline” improves throughput.

Hazards

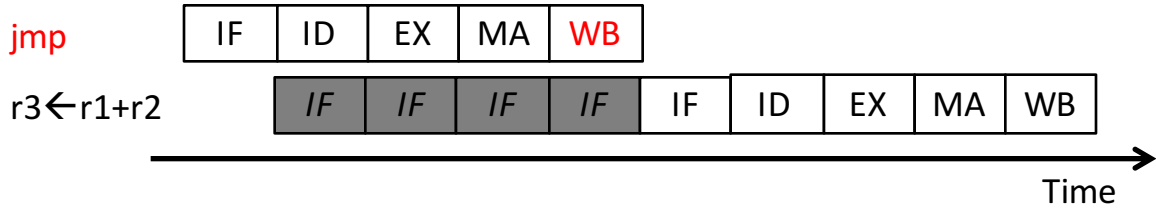
- Structural hazards
  - due to shared hardware resource



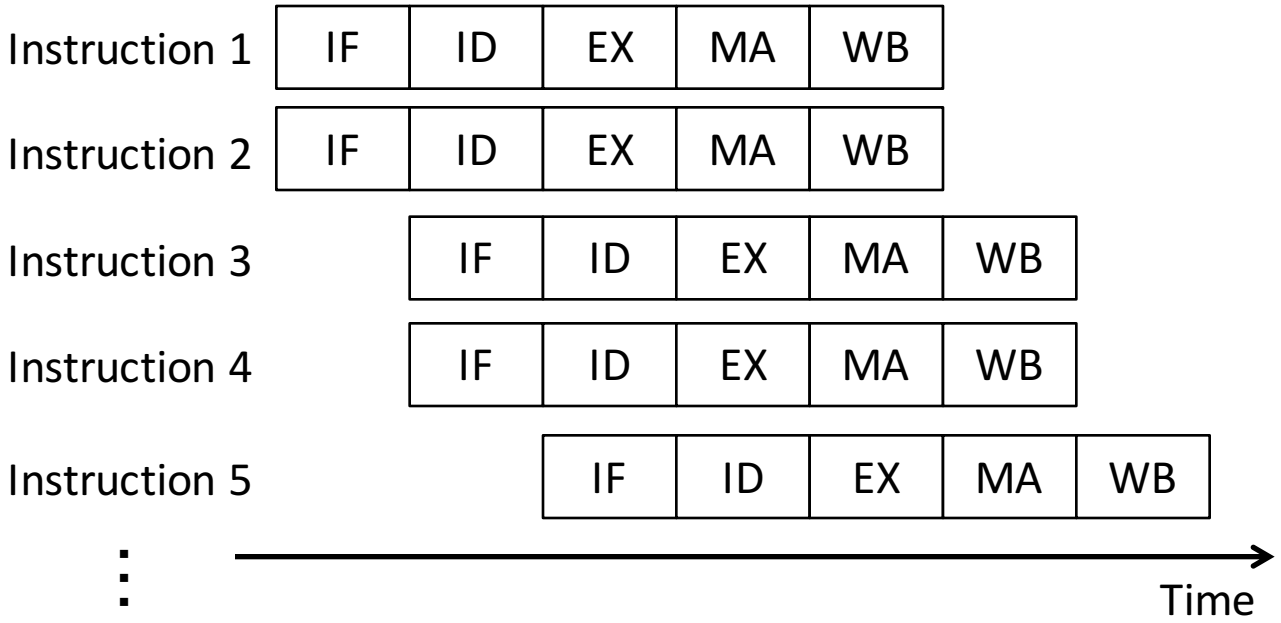
- Data hazards
  - due to data dependencies



- Control hazards

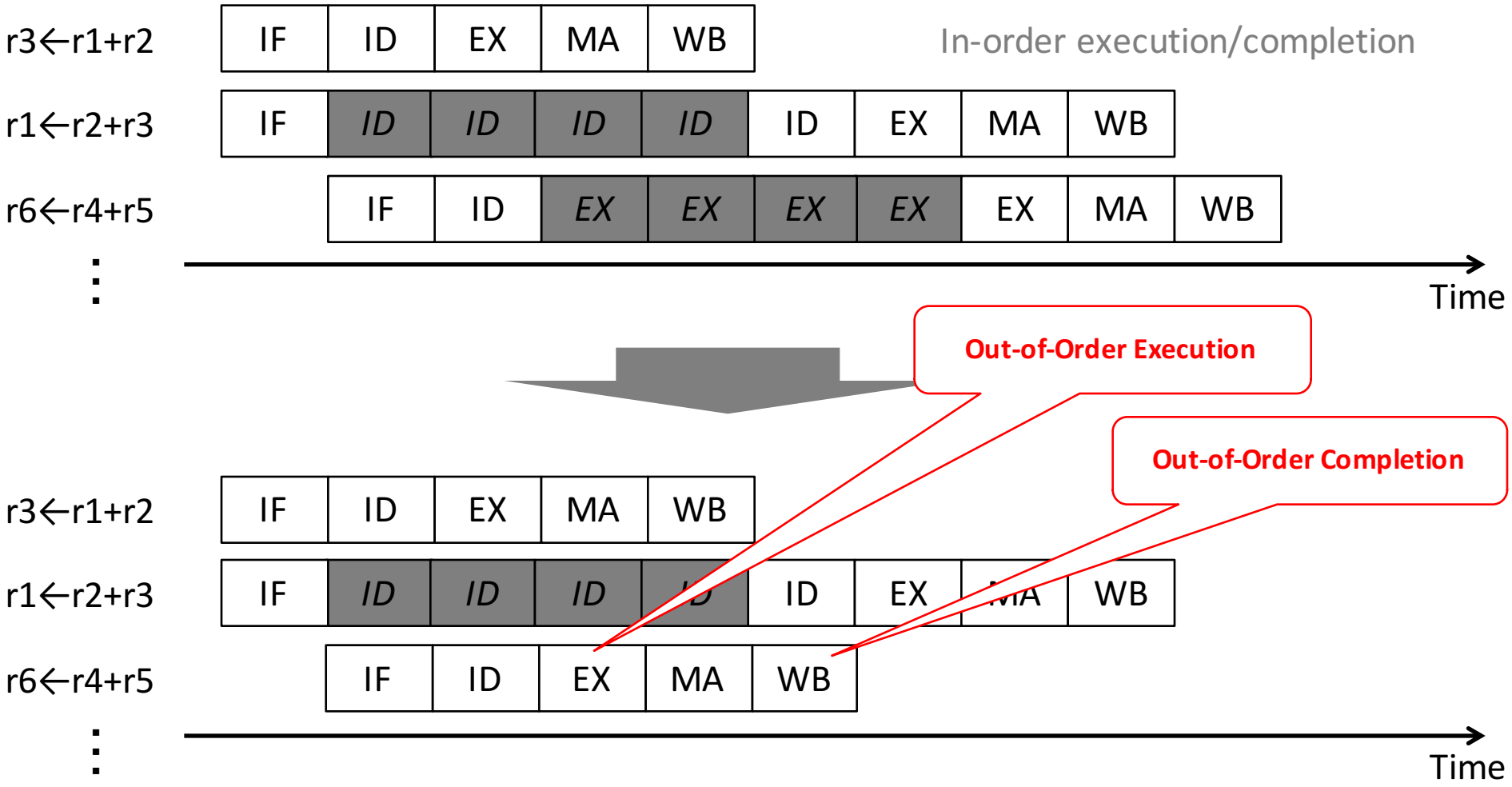


Superscalar



Scalar processor: No instruction-level parallelism
Vector: processor: Parallel data processing
→ Superscalar: Instruction-level parallelism

Out-of-Order Execution/Completion



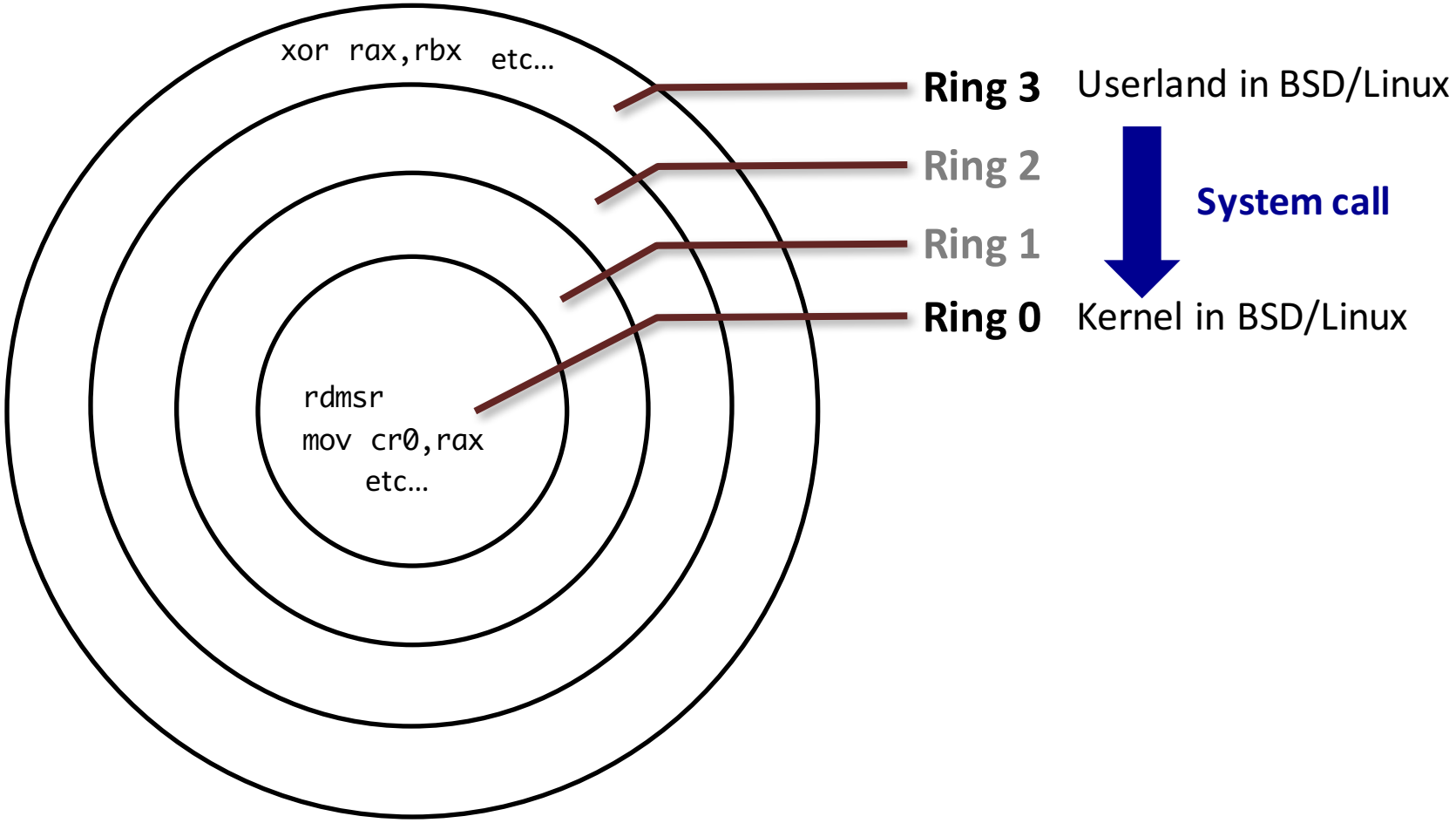
Other Technologies

- Cache
  - Write-Through vs. Write-Back
  - Fully Associative vs. Set Associative
  - Coherency (Multicore/Multiprocessor)
- Virtual memory
  - Page table
    - TLB (Translation Lookaside Buffer)
- Simultaneous Multithreading (Hyper-Threading Technology)

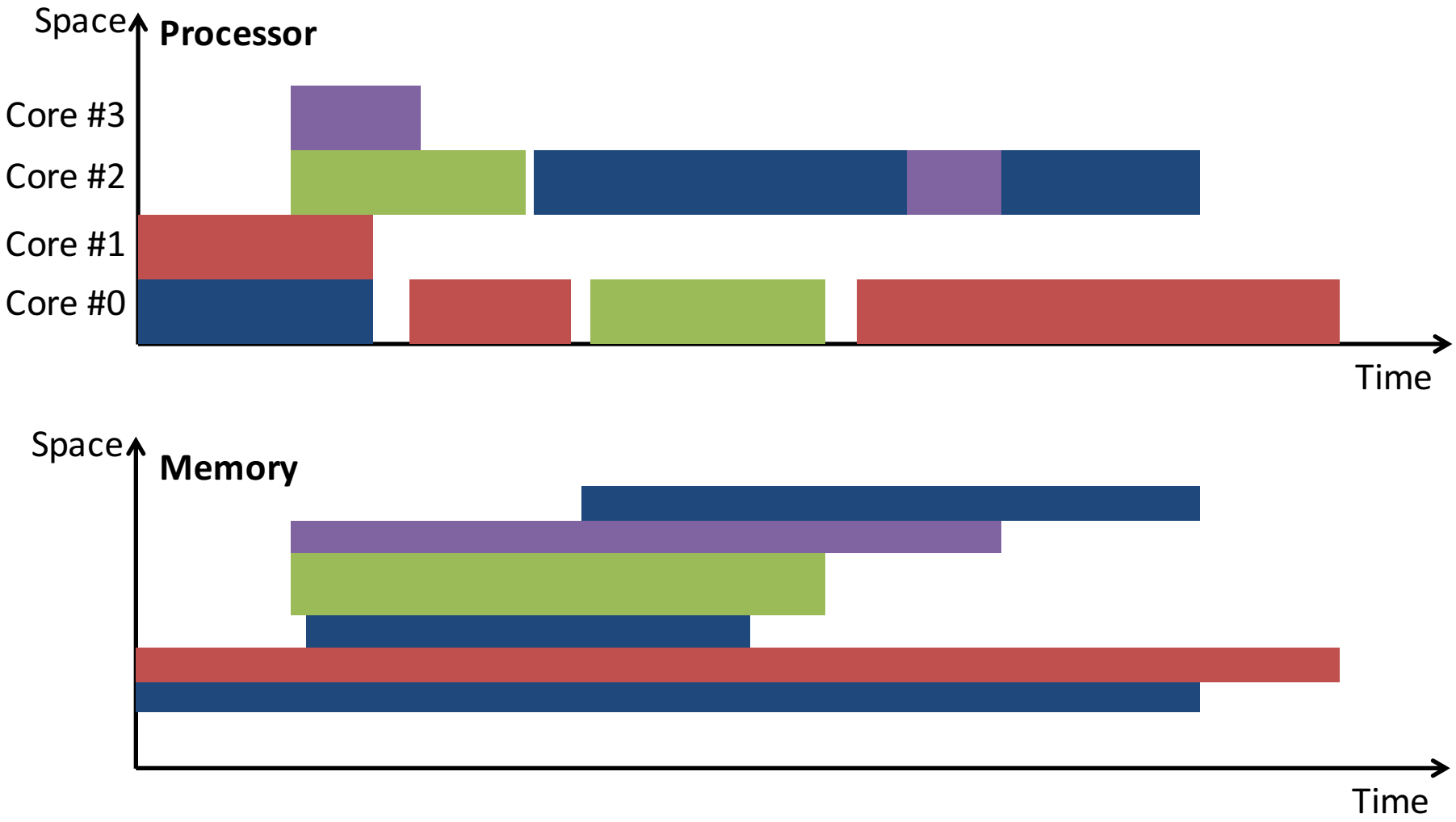
Operating System (OS)

- Resource sharing
  - CPU: Process scheduling
  - Memory: Memory management
- Privilege management
  - Not allow to execute privileged instructions to users' process
    - System call: Dispatch a privileged instruction to kernel
      - e.g., I/O instructions, cli/sti
- Inter-process communication
- etc…

x86/x86-64 Ring Protection



Multitask OS: Resource Sharing

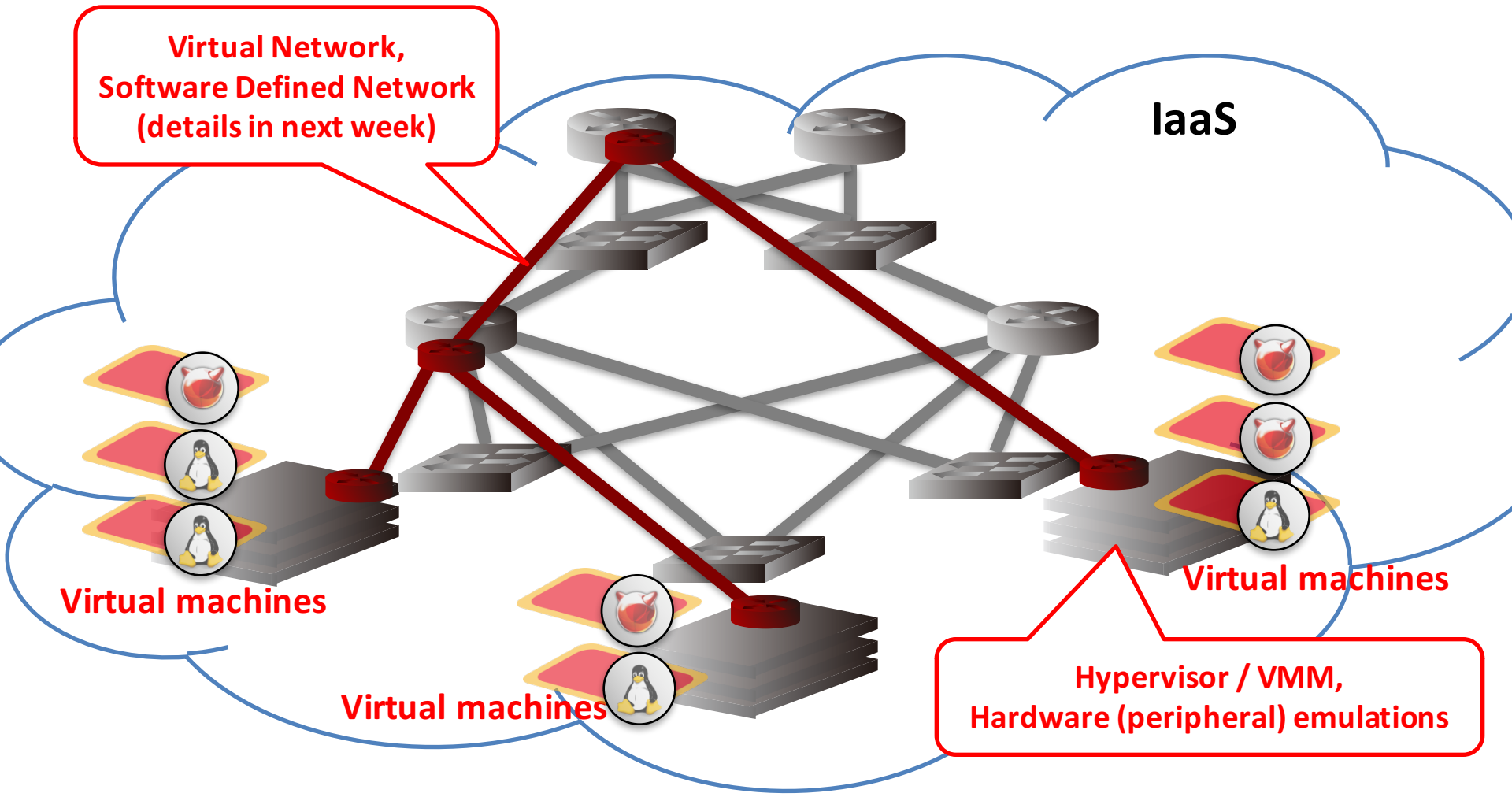


Scheduling Algorithms

- Systems
  - Batch
    - First-Come First-Serve
    - Shortest Job First
    - Shortest Remaining Time Next
  - Interactive
    - Round-Robin Scheduling
    - Priority Scheduling
    - Priority Scheduling w/ Multiple Queue (different quantum)
    - Shortest Process Next
    - Guaranteed Scheduling
    - Lottery Scheduling
    - Fair-Share Scheduling
  - Realtime

Technologies behind IaaS

Technologies behind IaaS

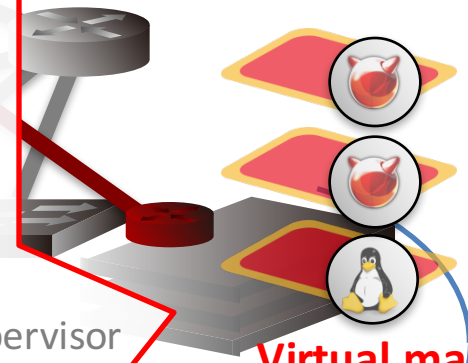


Technologies behind IaaS

Hypervisor / VMM
- Resource sharing
  - CPU
  - Memory
- Privilege management
- Virtual chipset / controllers
  - PIC/APIC
  - PIT
  - RTC CMOS

Virtual hardware (peripheral emulations)
- IDE/SATA HDD / CD drive
- Ethernet NIC
- Virtual display (video RAM)
- BIOS / UEFI
- etc…

may be combined with hypervisor



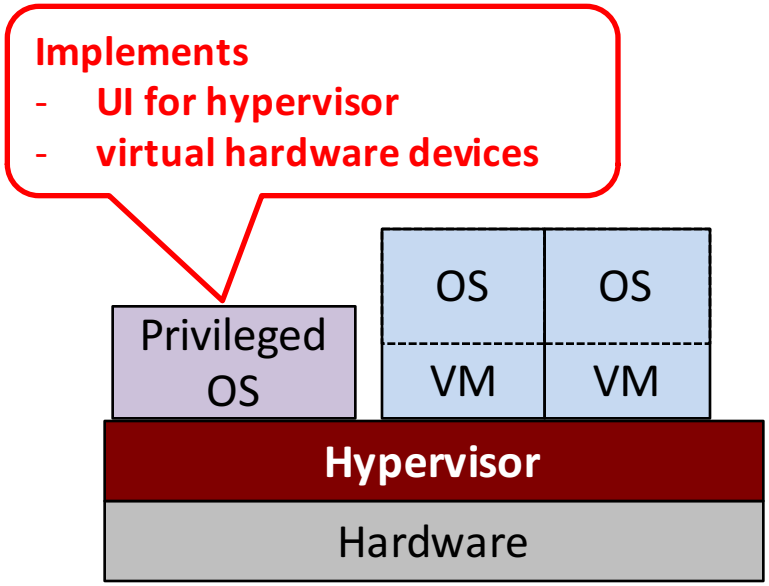
IaaS

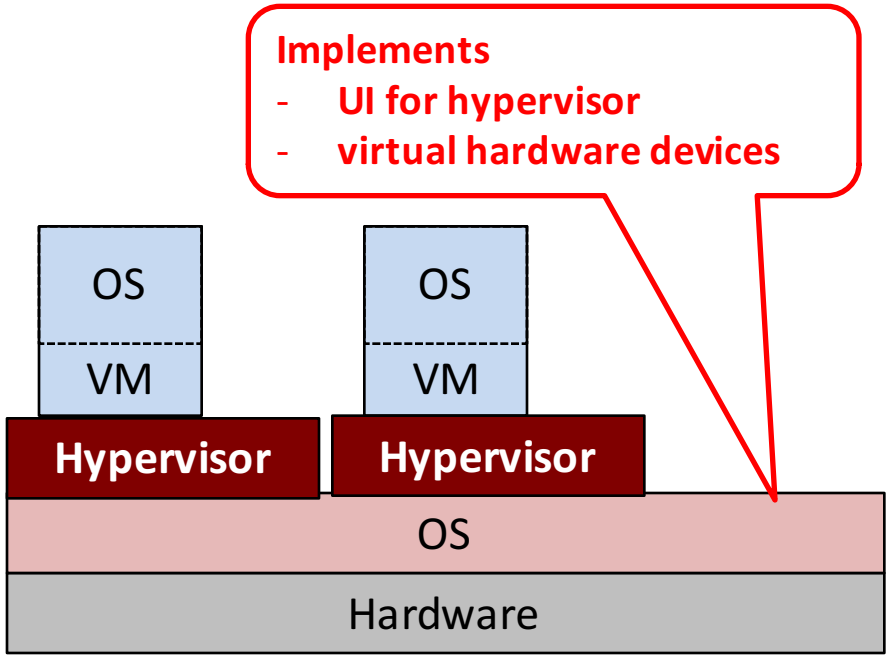
Virtual machines

Terminology

- Hypervisor / Virtual Machine Monitor (VMM)
  - A piece software/firmware that runs virtual machines
- Virtual Machine (VM)
  - A computer emulated by software
    - (usually by the assistance of hardware)
- Guest OS
  - An OS that is executed on a virtual machine

Two Types of Hypervisor





Type 1: Native (bare metal) hypervisor

e.g.,
- Xen
- VMware ESX/ESXi (vSphere Hypervisor)

Type 2: Hosted hypervisor

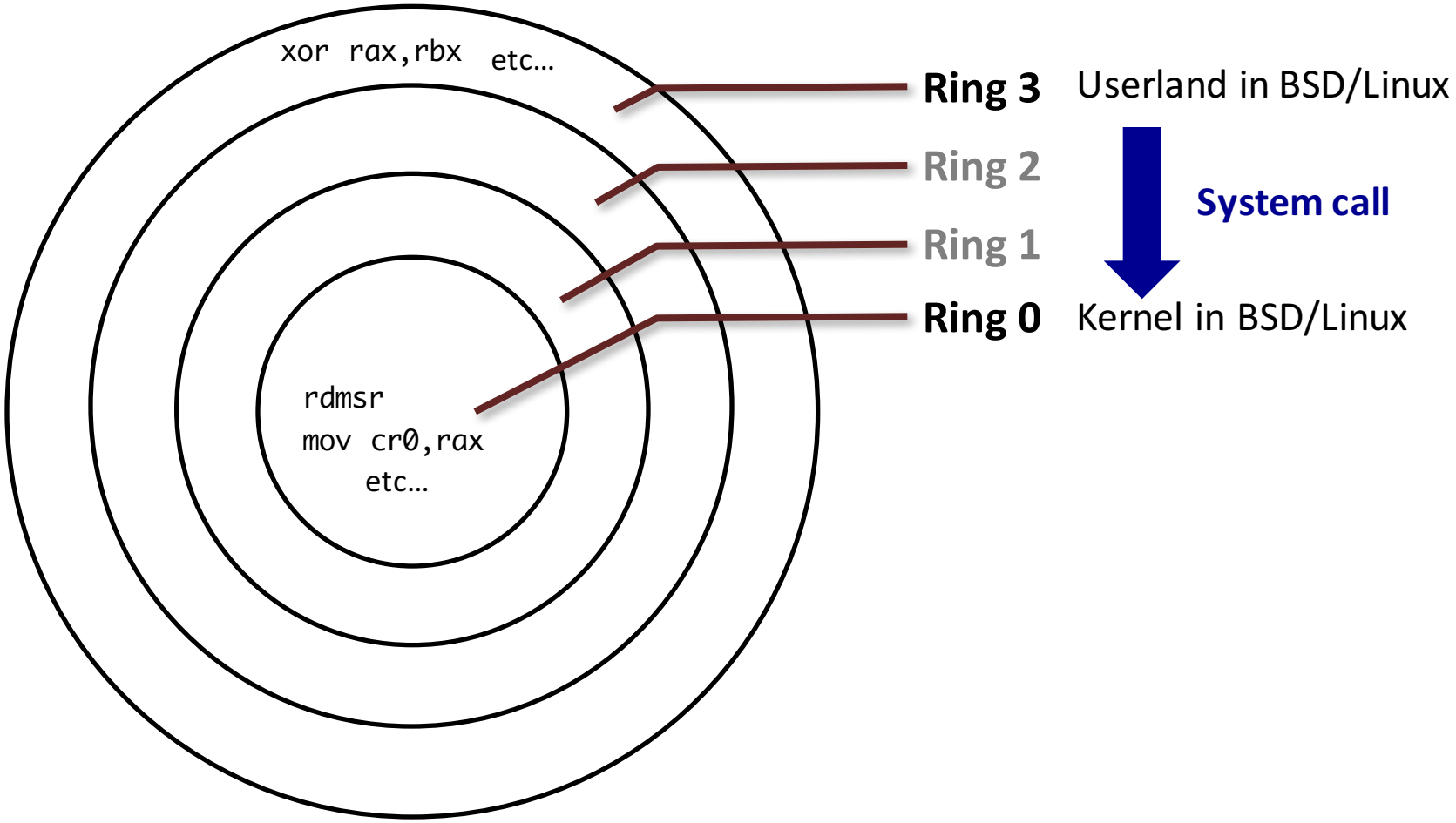
e.g.,
- VirtualBox
- VMware Workstation
- Linux KVM (sometimes classified as type 1)

Note: Some implementations may implement virtual hardware devices combined with hypervisor.

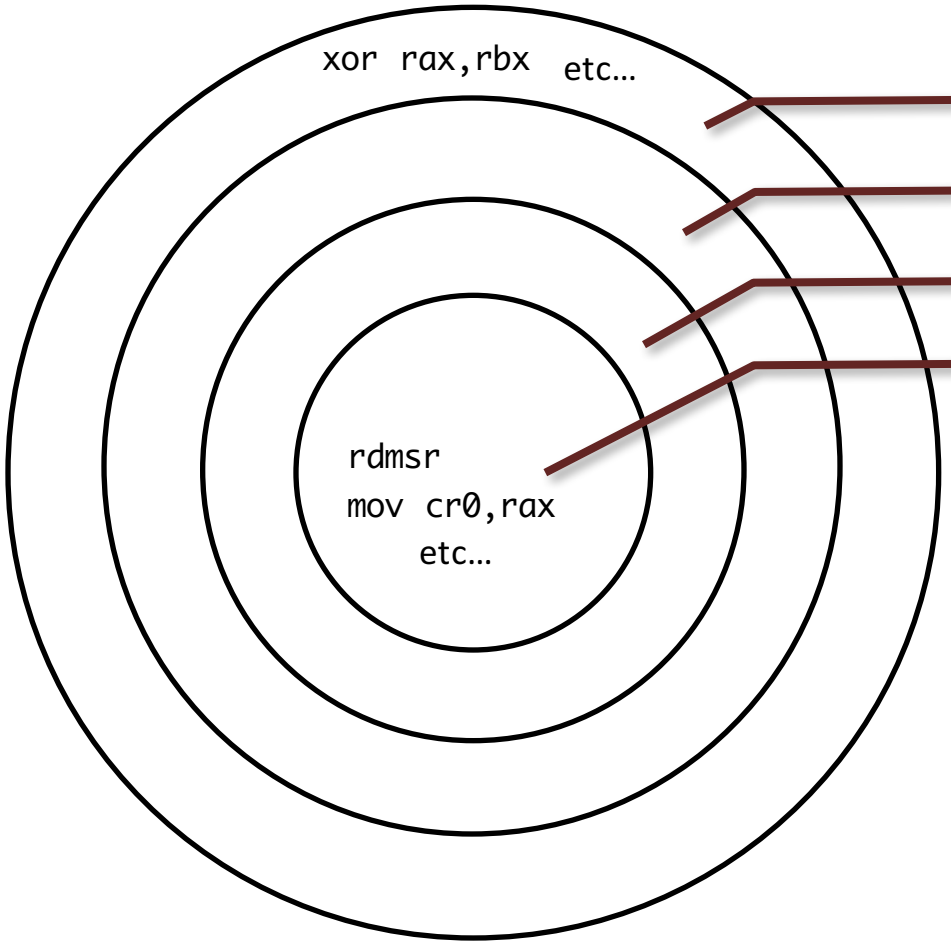
Two Types of Virtualization

- Full virtualization
  - Support the full set of instructions (of corresponding ISA) in VMs
    - Pros: No modifications to guest OS are required.
    - Cons: Hardware-assistance is required.
- Paravirtualization
  - Support a partial set of instructions (of corresponding ISA) in VMs
    - Pros: Hardware-assistance is not required.
    - Cons: Modifications to guest OS are required.
      - Privileged instructions must be replaced with hypervisor call.

Paravirtualization: x86/x86-64 Ring Protection



Paravirtualization: x86/x86-64 Ring Protection



Ring 3 Guest userland (BSD/Linux)

Ring 2

System call

Ring 1 Guest kernel (BSD/Linux)

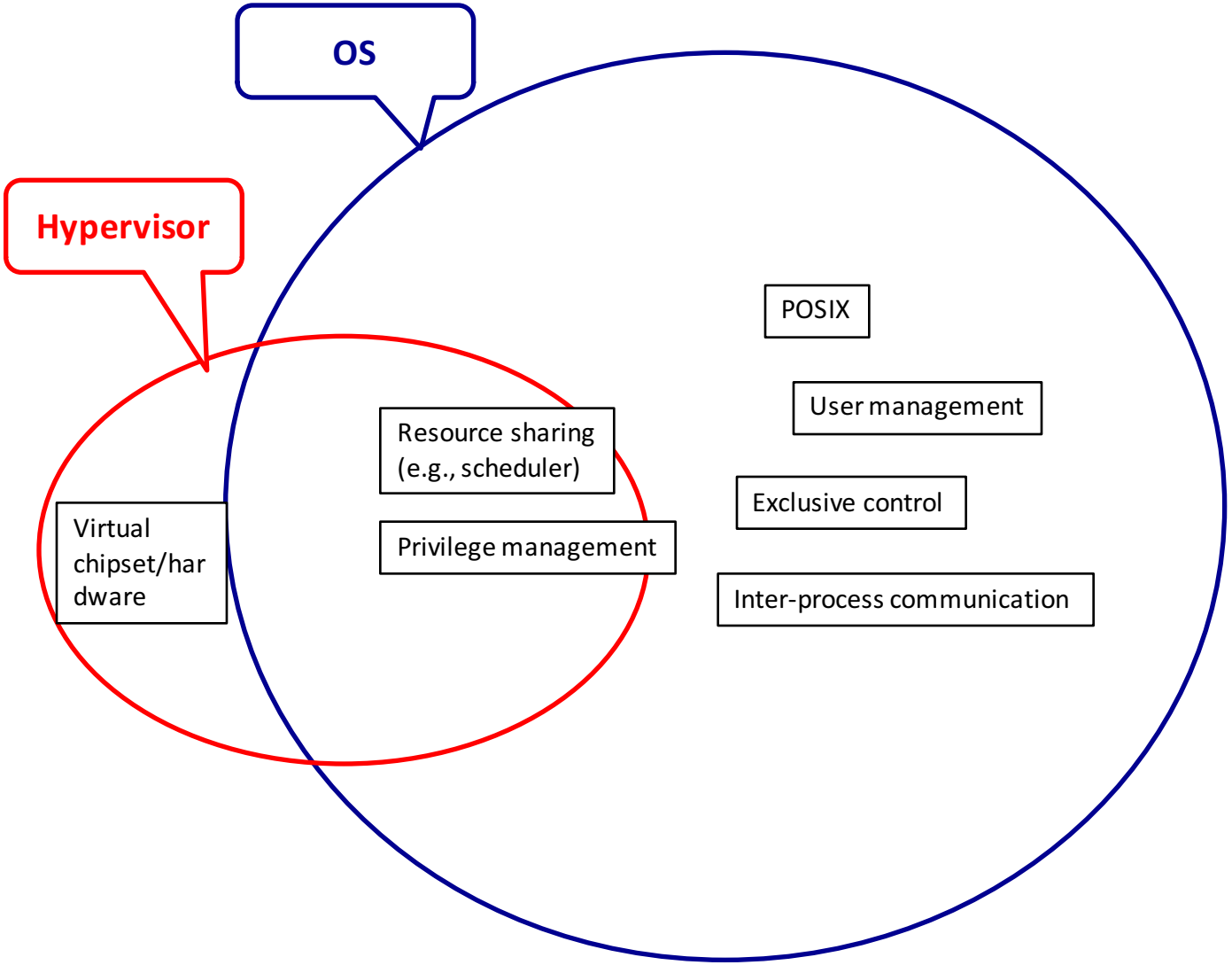
Hypervisor call

Ring 0 Hypervisor (Xen)

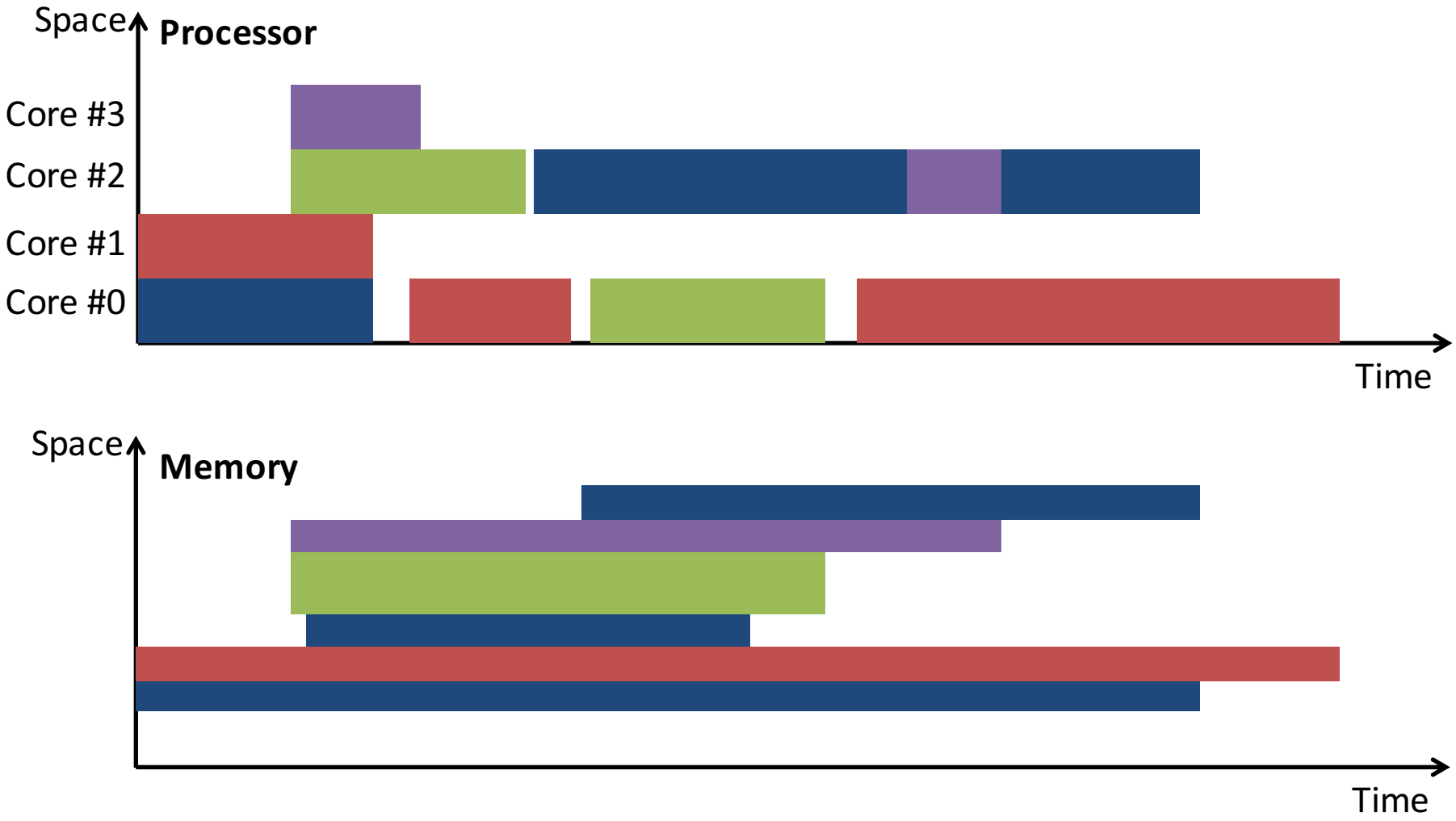
Differences between OS and Hypervisor

- Hypervisor / VMM
  - Resource sharing
    - CPU
    - Memory
  - Privilege management
  - Virtual chipset / controllers
    - PIC/APIC
    - PIT
    - RTC CMOS
- Virtual hardware (peripheral emulations)
  - IDE/SATA HDD / CD drive
  - Ethernet NIC
  - Virtual display (video RAM)
  - BIOS / UEFI
  - etc…

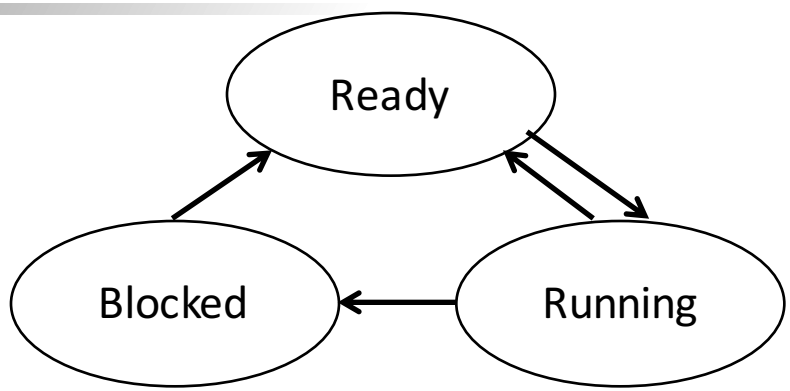
Differences between OS and Hypervisor



Resource Sharing



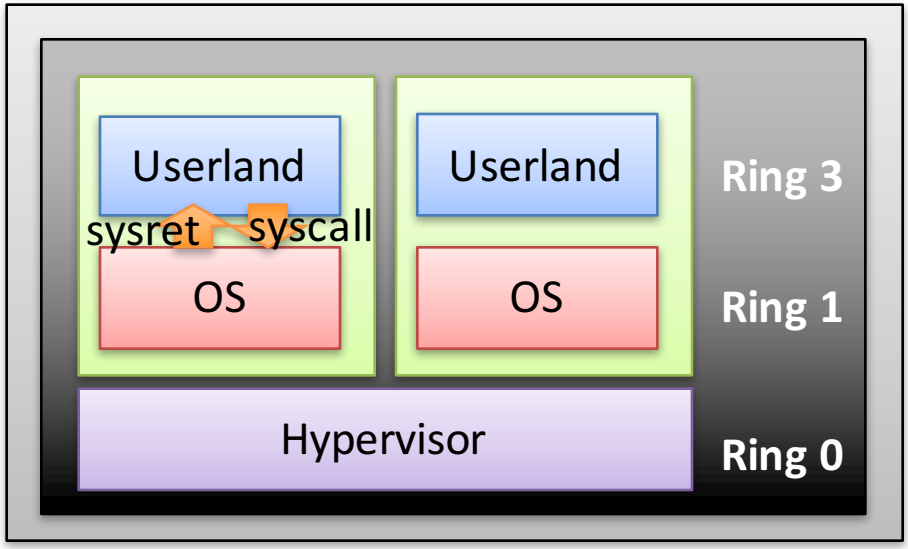
Differences between OS and Hypervisor



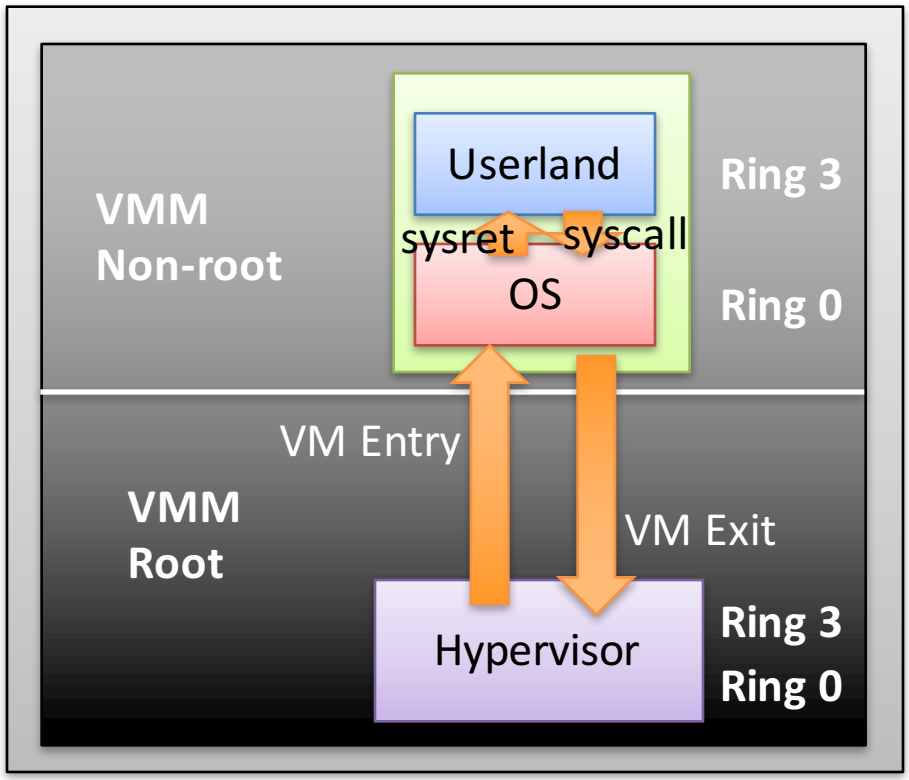
- Scheduler
  - OS: Process scheduler
    - Blocked: Inter-process communication
  - Hypervisor: VM scheduler
    - Blocked: Not inter-VM; between VM and virtual devices
      → Simpler
- Memory management
  - OS: Variety of allocation sizes (several byte – ~Gbytes)
    - Usual page table entry size: 4KiB/2MiB
    - Complex allocation algorithm
      - Linux: Buddy system (for page allocation), Slab allocator (for smaller size allocation)
  - Hypervisor: ~Gbytes
    → Easier

Hardware-assisted Virtualization Technology

- Intel® VT-x, AMD-V

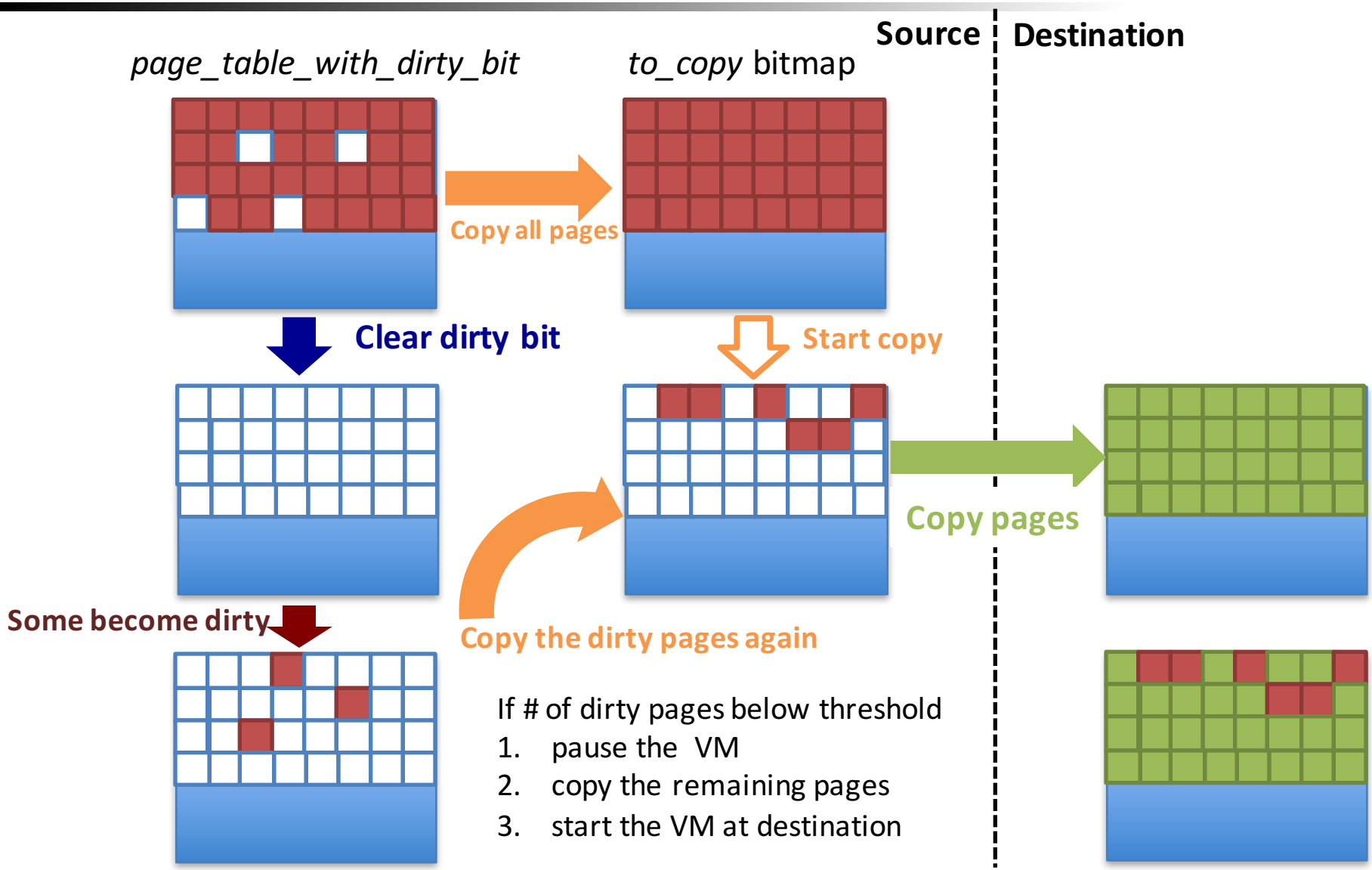


Generic design of hypervisor with IA Ring protection architecture without hardware's virtualization assist



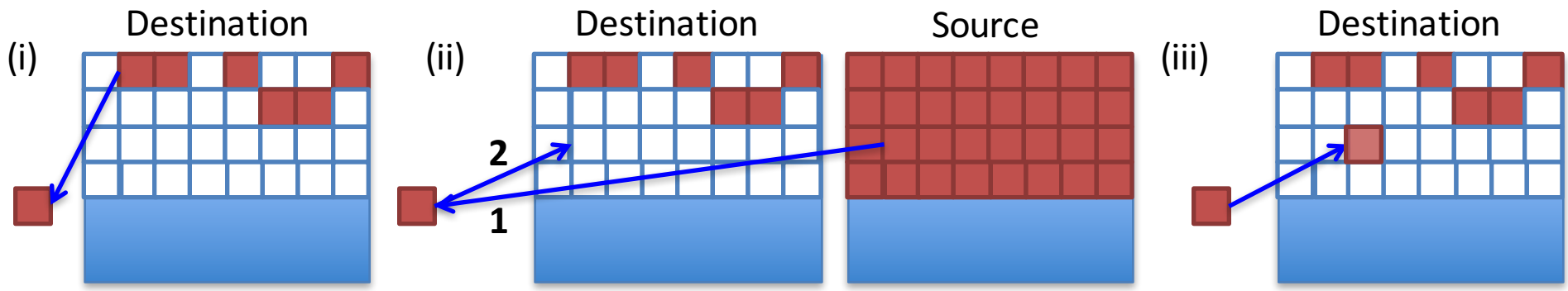
Generic design of hypervisor with hardware's virtualization assist (Intel® VT-x)

Live Migration: Pre Copy



Live Migration: Post Copy

- Read/Write operations
  - Read
    - If the page has already copied† to destination
      - (i) read the page from destination
    - Otherwise
      - (ii) read the page from source and copy it to destination
  - Write
    - (iii) write to destination



(†Copied pages are managed by a bitmap)

“We reject: kings, presidents and voting.

We believe in: rough consensus and running code.”

by David D. Clark

RUNNING(?) CODE OF OPERATING SYSTEM AND HYPERVISOR

Page Table (x86-64 long mode)

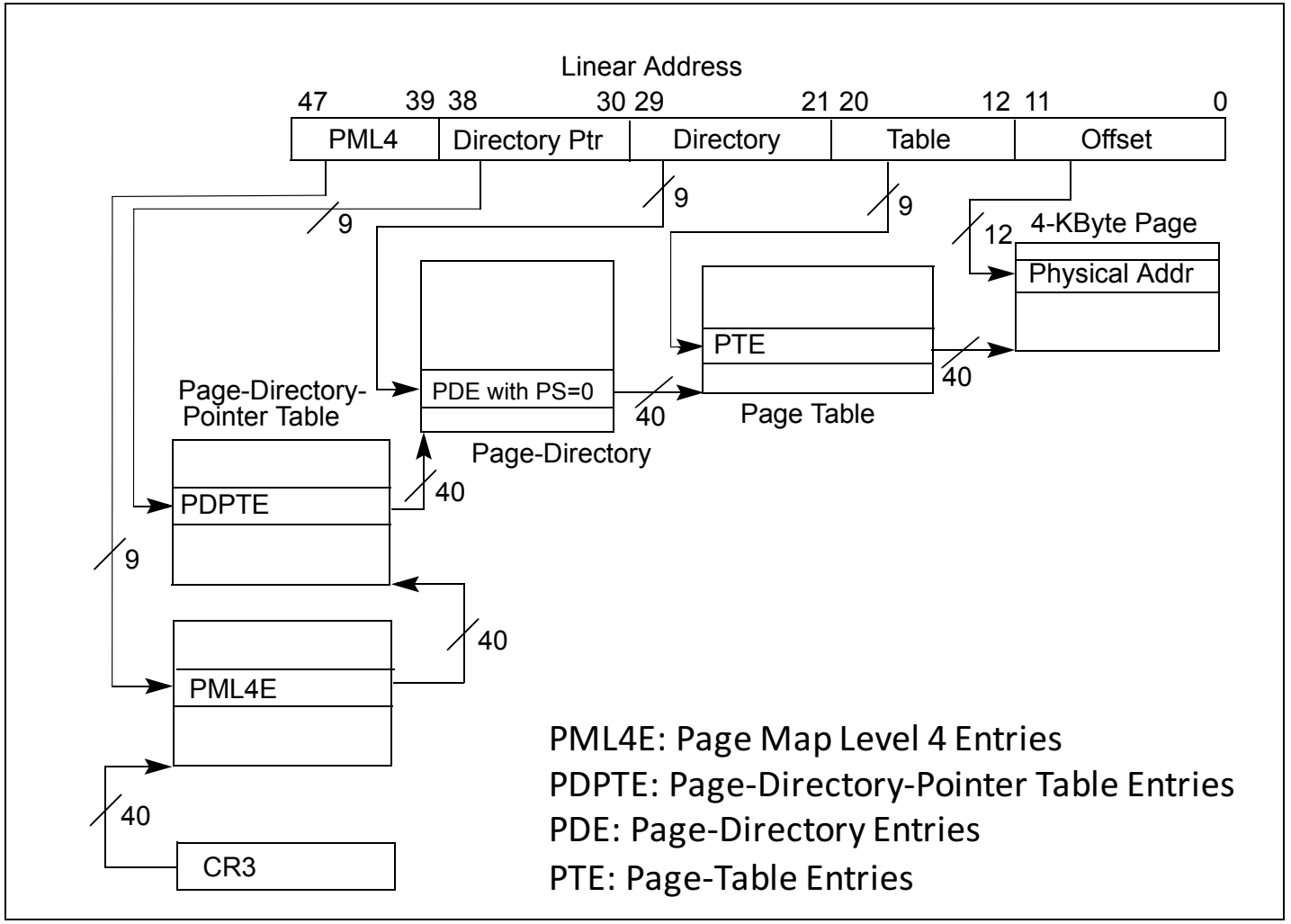


Figure from Intel® 64 and IA-32 Architectures Software Developer's Manual Vol. 3A 4.5

Page Table (x86-64 long mode)

63	62	61	60	59	58	57	56	55	54	53	52	51		M^1	M-1			32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved[2]															Address of PML4 table																								Ignored							PCD	PWT	Ign.			CR3
XD[3]	Ignored													Rsvd.	Address of page-directory-pointer table																								Ign.			Rsvd	Ign	A	PCD	PWT	U/S	R/W	1	PML4E: present
Ignored																																																		0	PML4E: not present
XD	Ignored													Rsvd.	Address of 1GB page frame													Reserved									PAT	Ign.			G	1	D	A	PCD	PWT	U/S	R/W	1	PDPTE: 1GB page
XD	Ignored													Rsvd.	Address of page directory																								Ign.			0	Ign	A	PCD	PWT	U/S	R/W	1	PDPTE: page directory
Ignored																																																		0	PDTPE: not present
XD	Ignored													Rsvd.	Address of 2MB page frame																								Reserved			PAT	Ign.			G	1	D	A	PCD	PWT	U/S	R/W	1	PDE: 2MB page
XD	Ignored													Rsvd.	Address of page table																								Ign.			0	Ign	A	PCD	PWT	U/S	R/W	1	PDE: page table
Ignored																																																		0	PDE: not present
XD	Ignored													Rsvd.	Address of 4KB page frame																								Ign.			G	PAT	D	A	PCD	PWT	U/S	R/W	1	PTE: 4KB page
Ignored																																																		0	PTE: not present

Figure from Intel® 64 and IA-32 Architectures Software Developer's Manual Vol. 3A 4.5

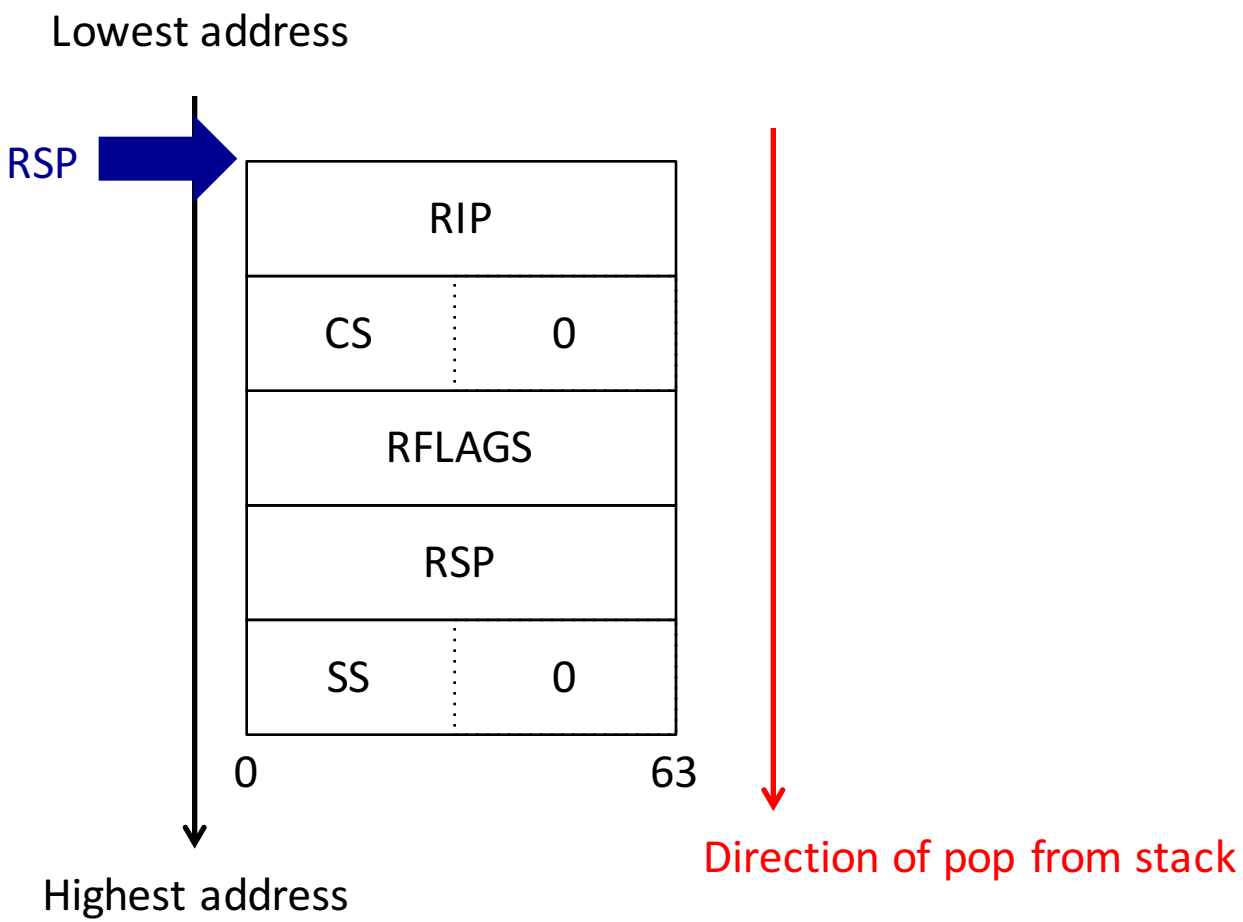
Page Table (x86-64 long mode)

```
/* Create 64bit page table */
pg_setup:
        movl    $KERNEL_PGT,%ebx        /* Low 12 bit must be zero */
        movl    %ebx,%edi
        xorl    %eax,%eax
        movl    $(512*8*6/4),%ecx
        rep     stosl                   /* Initialize %ecx*4 bytes from %edi */
                                        /*  with %eax */
/* Level 4 page map */
        leal    0x1007(%ebx),%eax
        movl    %eax,(%ebx)
/* Page directory pointers (PDPE) */
        leal    0x1000(%ebx),%edi
        leal    0x2007(%ebx),%eax
        movl    $4,%ecx
pg_setup.1:
        movl    %eax,(%edi)
        addl    $8,%edi
        addl    $0x1000,%eax
        loop    pg_setup.1
/* Page directories (PDE) */
        leal    0x2000(%ebx),%edi
        movl    $0x183,%eax
        movl    $(512*4),%ecx
pg_setup.2:
        movl    %eax,(%edi)
        addl    $8,%edi
        addl    $0x00200000,%eax
        loop    pg_setup.2

/* Setup page table register */
        movl    %ebx,%cr3
```

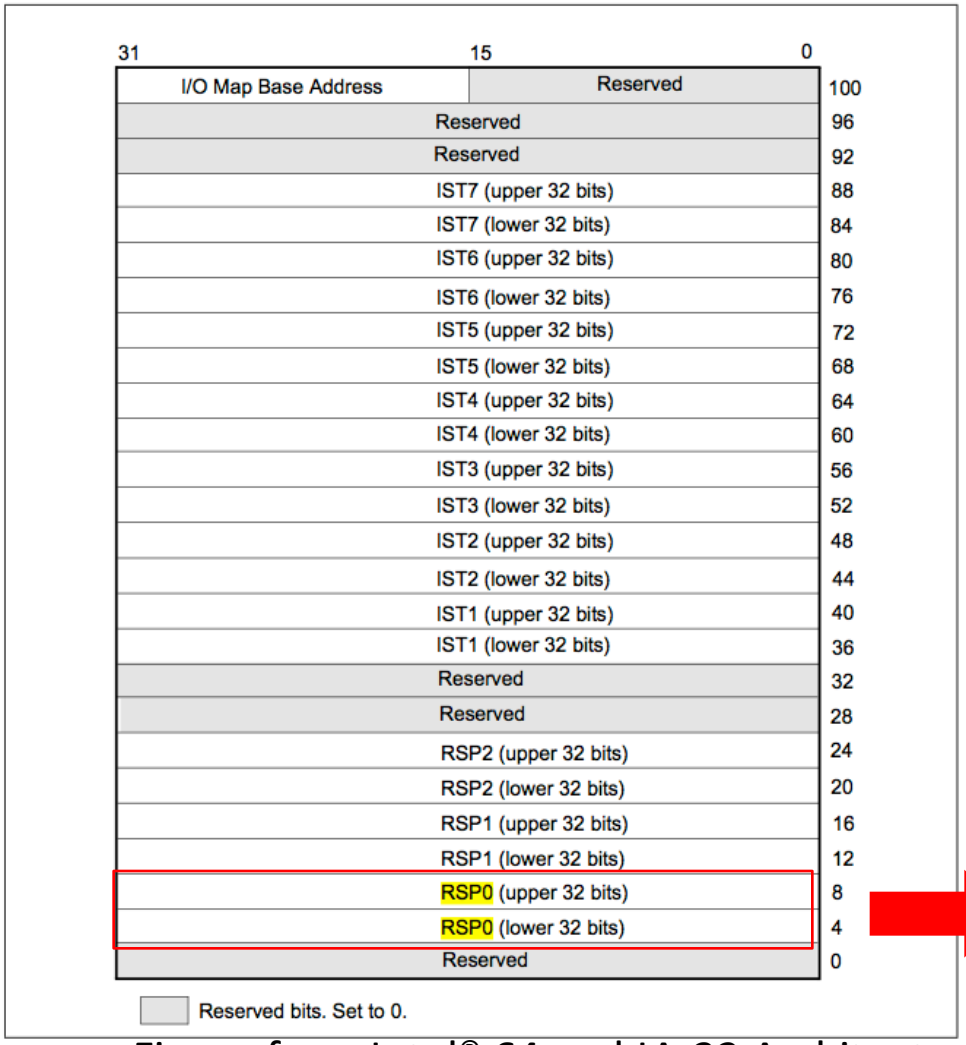
Multitask OS: Context Switch

IRETQ



Intel® 64 and IA-32 Architectures Software Developer's Manual Vol. 3A 5.8.5.1

Multitask OS: TSS (Task State Segment)



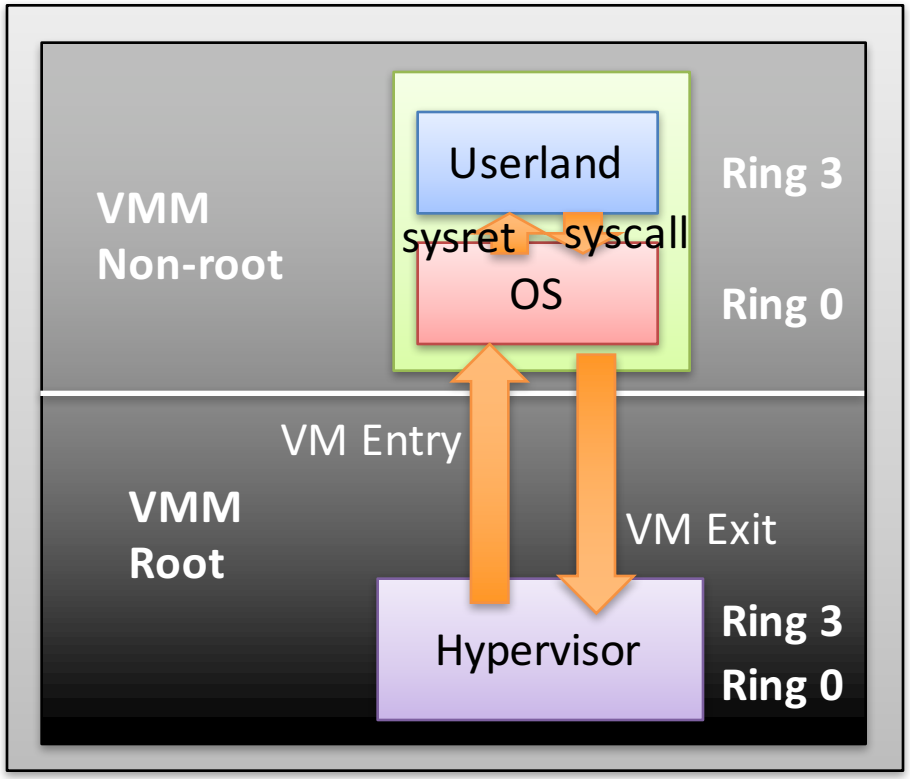
Point to kernel stack (Ring 0)

Figure from Intel® 64 and IA-32 Architectures Software Developer's Manual Vol. 3A 7.7

Multitask OS: Context Switch (Source Code)

```
/* Restart a task */
_task_restart:
        /* If the next task is not assigned, immediately restart */
        cmpq    $0,(_next_task)
        jz      1f
        /* Save stack pointer */
        movq    (_cur_task),%rax
        movq    %rsp,TASK_RP(%rax)
        /* Task switch (set the stack frame of the new task) */
        movq    (_next_task),%rax
        movq    %rax,(_cur_task)
        movq    TASK_RP(%rax),%rsp      /* Copy sp0 */
        movq    $0,(_next_task)
        /* ToDo: Load LDT/cr3 */
        /* Setup sp0 in TSS */
        movq    (_cur_task),%rax
        movq    TASK_SP0(%rax),%rdx
        movq    (_tss),%rbp
        movq    %rdx,TSS_SP0(%rbp)
1:
        /* Pop registers */
        intr_irq_done
        iretq
```

Hypervisor



VMXON: Enable VMX
VMXOFF: Disable VMX

Virtual Machine Control Structure (VMCS)
- Guest-state area: Registers etc.
- Host-state area: Host state/exit point on every VM exit.
- VM-execution control fields
- VM-exit control fields
- VM-entry control fields
- VM-exit information fields

* Current VMCS per logical processor (of physical machine)
* Active VMCS per virtual processor

1. **VMREAD/VMWRITE/VMCLEAR** to configure current VMCS
2. Setup and prepare peripherals (virtual devices)
3. **VMLAUNCH**
4. Handle the instructions corresponding to VM exits and schedule/switch VMs w/ **VMRESUME**
   - VM exits caused by
     1. instructions of VM
     2. signal by hypervisor's VMX-preemption timer
     3. **VMCALL** from VM

Details in Intel® 64 and IA-32 Architectures Software Developer's Manual Vol. 3C Chapter 23

How to Implement Live Migration

- Memory migration
  - Pre copy or Post copy (or hybrid)
- VM state copy
  - VM exit (at source hypervisor)
  - Just copy VMCS
  - **VMRESUME** to start VM at destination hypervisor
- Storage
  - Network storage
  - Distributed storage
  - Storage migration like memory migration
- Network
  - Copy configuration (i.e., MAC address) of virtual NICs

Live Migration Support

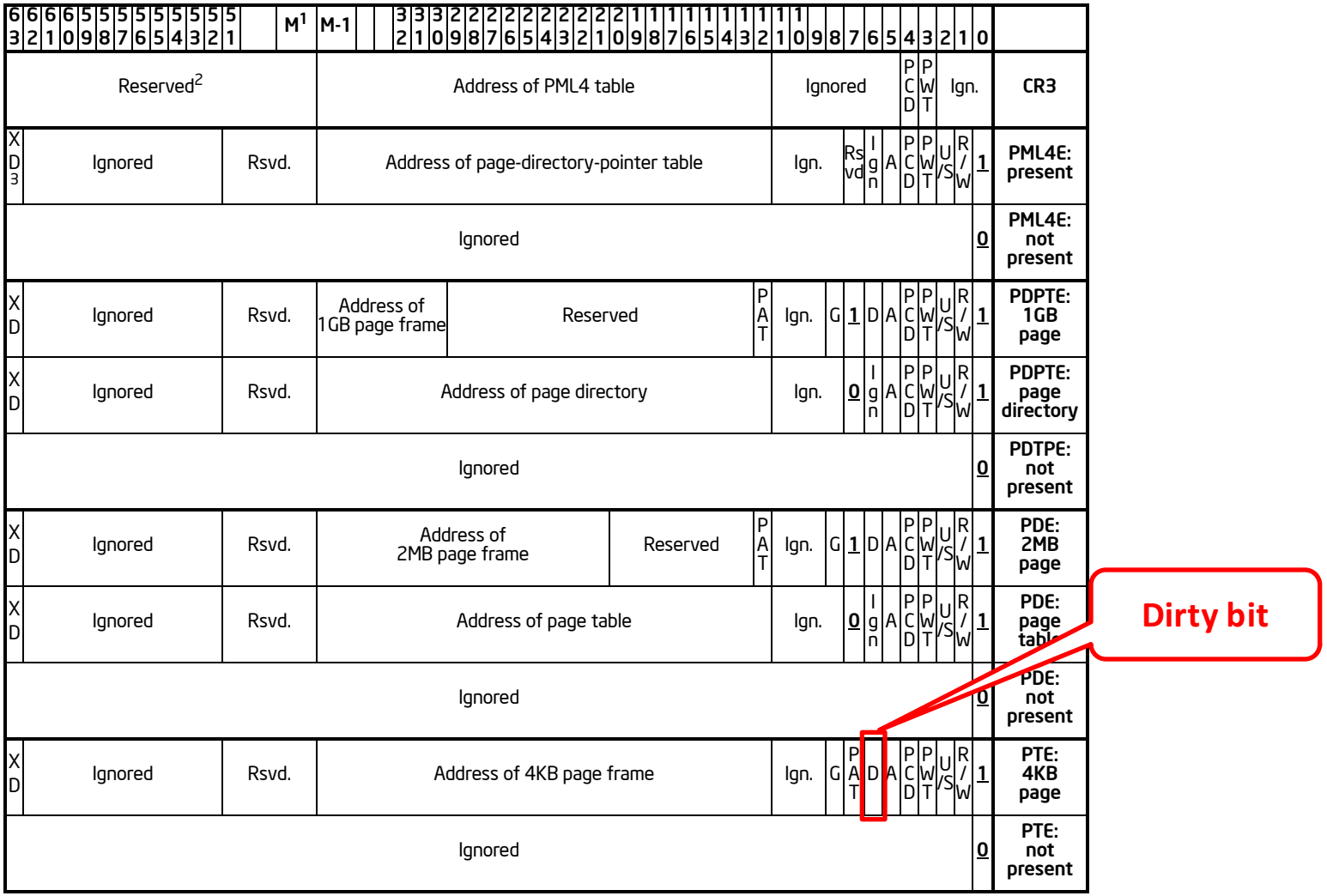


Figure from Intel® 64 and IA-32 Architectures Software Developer's Manual Vol. 3A 4.5

Technologies behind PaaS

Technologies behind PaaS

- LXC (Linux Containers)
  - Lightweight virtual machine
    - Run many instances
- Platforms and Frameworks
  - Platforms
    - Hadoop (for distributed computing)
    - Docker (for VM image management/deployment)
  - Frameworks
    - Ruby on Rails (for Web)
  - Load balancing
    - (These network portion will be in next week.)

Technologies behind SaaS

Technologies behind SaaS

- Rich UI over browser
  - HTML5
    - Support of new features
      - Canvas
      - Local storage / Session storage
      - Local file access w/o page transition (e.g., FileReader)
      - Drag & Drop
      - Multimedia support w/o third-party plugins
      - Offline application w/ cache
    - WebSockets / WebRTC
  - CSS3
    - New style modules e.g., gradient
  - JavaScript
    - Development of faster JavaScript engines
      - V8
      - asm.js

Group PBL

- IaaS
  - (A1) Cloud controller
    - Hypervisor: KVM
    - Cloud controller: (using libvirt API)
  - (A2) Cloud storage
    - High-speed storage: 10GbE / RAID card + SSD
- PaaS
  - (B1) Hadoop cluster
    - Hadoop (MapReduce)
  - (B2) PaaS cloud
    - Linux container
- SaaS
  - (C1) SaaS application
    - Server: Scale-out capable application
    - Client: Browser

For Group PBL

- Send your info & preference
  - Student # (学生証番号)
  - Name (氏名)
  - E-mail address (メールアドレス)
  - 4 ranked preferences from the list in next slide
    - 希望のPBL課題番号を4つ（希望順に）

  to <panda@edu.jar.jp> .

  Subject MUST be “Cloud PBL course 2016”