Outline

- ❑ *Concepts, classical CPU virtualization*
  - *Basic interpretation*
- ❑ *Memory virtualization*

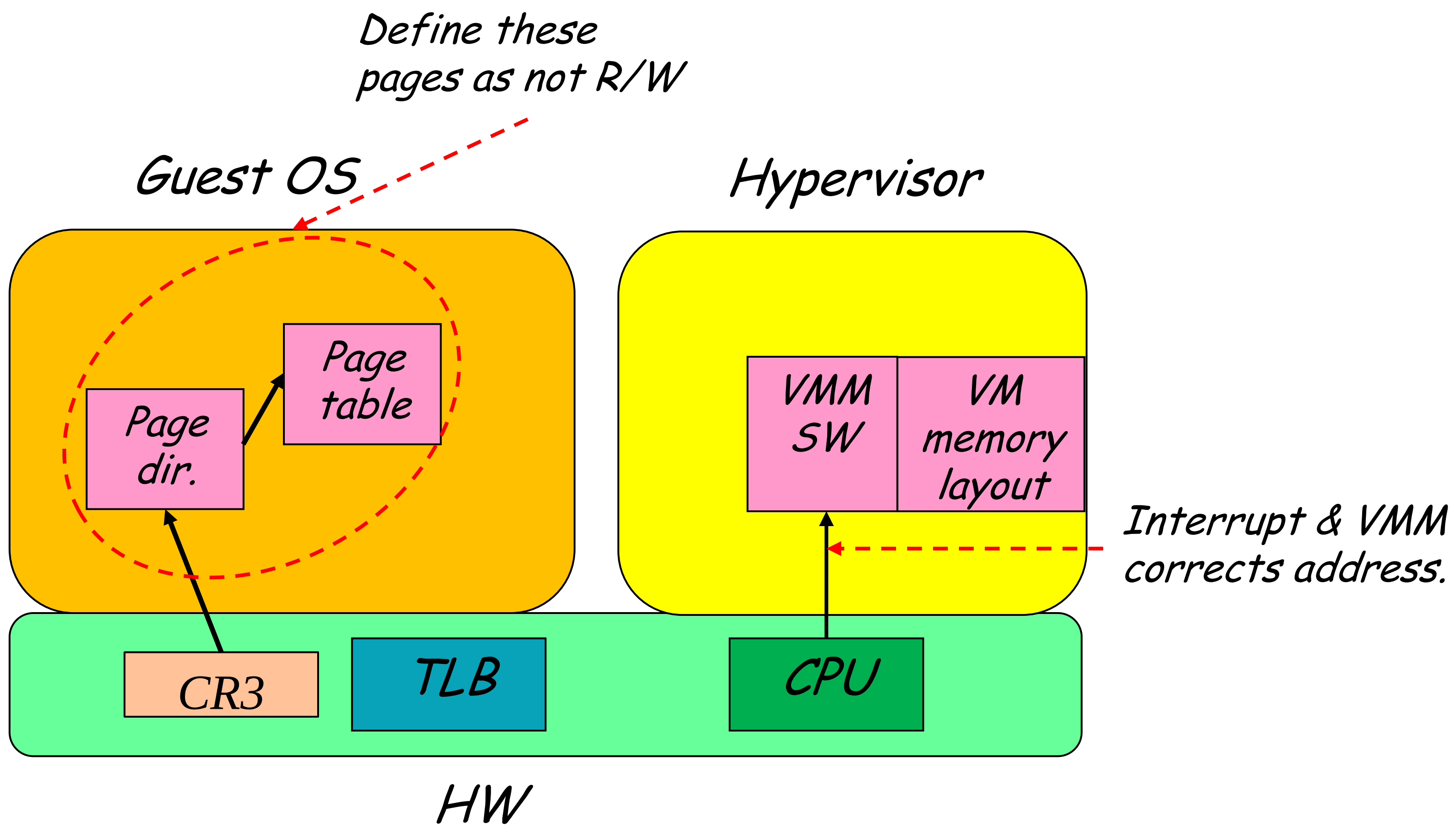
Memory allocation

- Each VM usually receives a contiguous set of physical addresses.
    - 512 Mbyte – 4 Gbyte are typical values.
- As far as VM is concerned, this is the physical memory of the machine.
- The guest OS allocates pages or segments to guest processes.

Memory management

- Assumptions of OS in VM:
  - Physical memory is a contiguous block of addresses from 0 to some n.
  - OS can map any virtual page to any page frame.
- Hypervisor must:
  - Partition memory among VMs.
  - Ensure virtual page mapping only to assigned page frames.
- TLB – page fault in HW-managed TLB (e.g. x86) causes HW to select a page from page table.
- VM OS must not manage real page table.

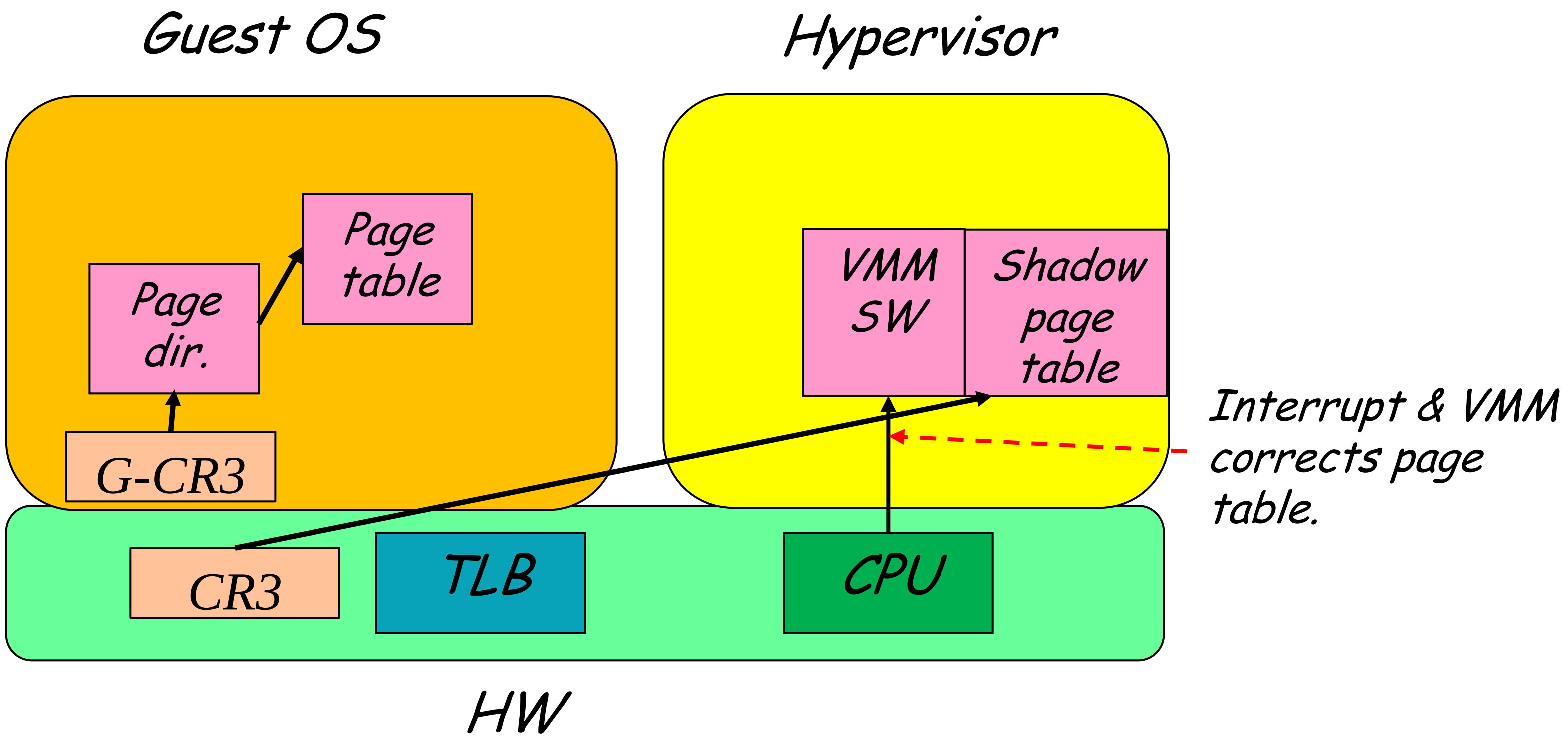
Option 1: brute force



Brute force – description

- Guest page tables are read and write protected in host system.
- If guest OS reads page table (e.g. for page eviction) writes page table (e.g. after page fault), or changes CR3, the system traps.
- The hypervisor then uses a VM memory layout to:
  - Return answers to VM
  - Update the layout
- Hypervisor switches VM memory layout when new VM is scheduled.

Option 2: shadow page tables



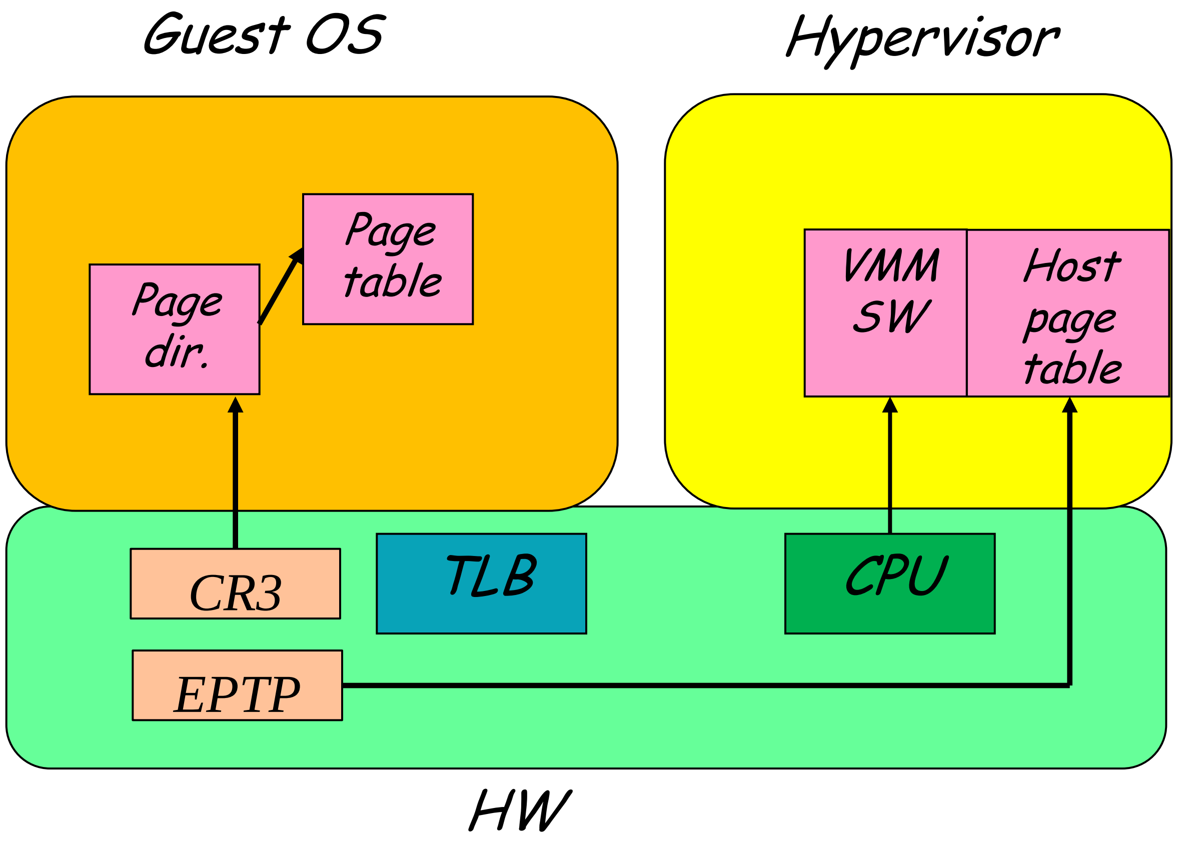
Shadow page tables – description

- Hypervisor maintains “shadow page tables”.
- Guest page tables map: Guest VA→ Guest PA
- Shadow tables: Guest VA→ Host PA.
- Hypervisor does not trap guest updates to its page table.
  - Result – inconsistent guest page table and shadow page table.
- When guest process accesses virtual address
  - The physical address is not in the guest page table, but in the shadow page table.
  - HW translates correctly, because it is aware only of shadow tables.

Shadow page tables – description (continued)

- If address in TLB – TLB hit and no problem.
- When guest process causes a page fault
  - Hypervisor begins execution.
  - Hypervisor updates guest page table with new page.
  - Hypervisor updates shadow page table.
- Performance is as good as native execution as long as there are no page faults.
- Shadow page tables should be cached so that once a VM is re-scheduled the page table does not have to be rebuilt from scratch.

Option 3: nested page tables



Nested page tables - description

- The name implies having page tables within page tables.
- The essence of the idea is a hardware assist.
  - Hardware has an extra pointer and the ability to walk an extra set of page tables.
  - Idea is called Extended Page Tables (EPT) by Intel
- Guest page tables hold Guest VA →Guest PA mapping, access by standard CR3
- Extended page tables hold Host VA → Host PA mapping, access by EPTP (EPT pointer).
- Host VA=Guest PA

Nested page tables – description (cont'd)

- ❑ TLB as usual holds Guest VA →Host PA
- ❑ On memory access
  - o If found in TLB – no problem.
  - o If not in TLB, but no page fault, hardware walks both tables and updates TLB.
  - o If page fault, then hardware hypervisor gets host physical page and provides host virtual page (guest physical) to VM.

Sources

- “Modern operating systems”, 4‘th edition, A. Tanenbaum and H. Bos
- “Virtual machines”, J. E. Smith and R. Nair
- A presentation by Niv Gilboa from CSE@BGU
- “Formal requirements for virtualizable third generation architectures”, G. J. Popek and R. P. Goldberg, CACM, 1974
- “A comparison of software and hardware techniques for x86 virtualization”, K. Adams and O. Ageson, ASPLOS 2006