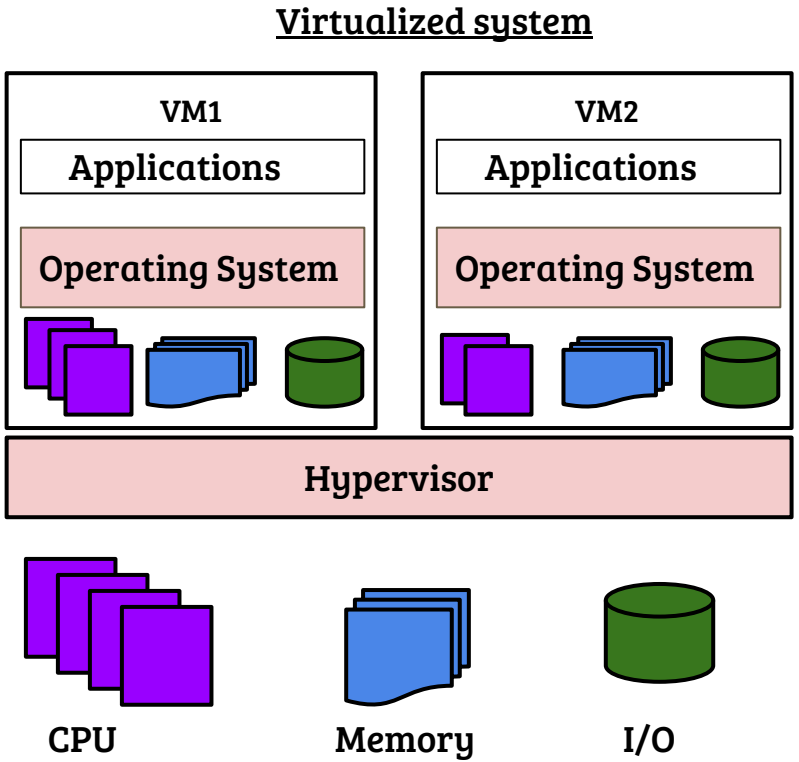
Issues and Challenges in Memory virtualization

CS698E

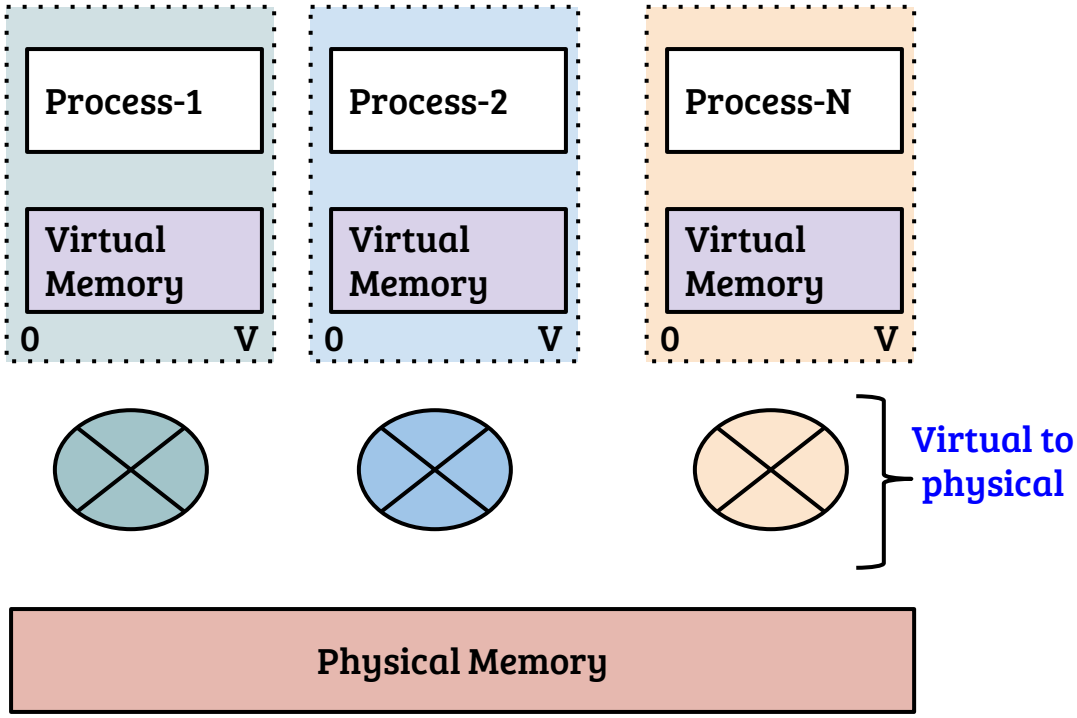
Virtualization: Resource multiplexing with isolation



- ➔ Definition [1] "Not physically existing as such but made by software to appear to do so."
- ➔ Core objectives
  - ◆ Equivalence
  - ◆ Isolation
  - ◆ Resource control
  - ◆ Efficiency
- ➔ Today's lecture: Memory virtualization

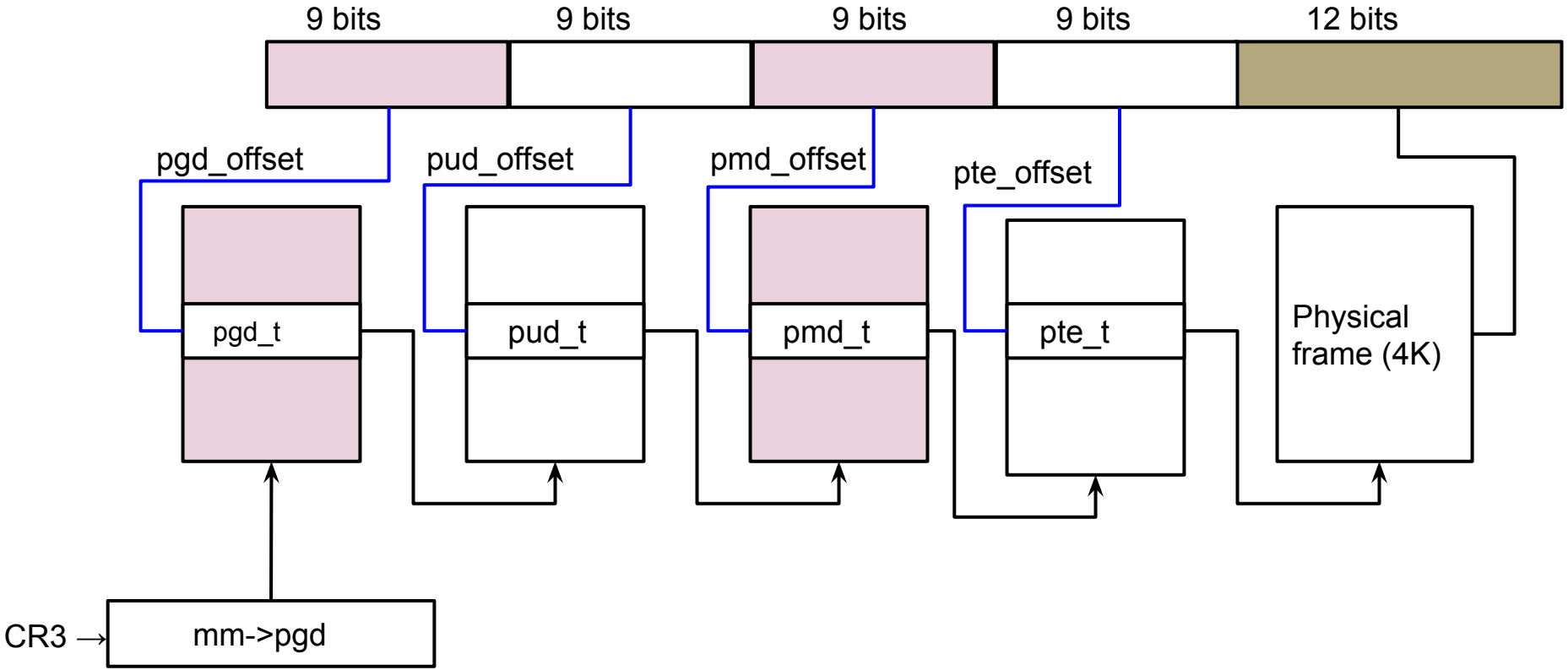
1. Oxford dictionary : https://en.oxforddictionaries.com/definition/virtual

Hang on! Isn't memory already virtualized?

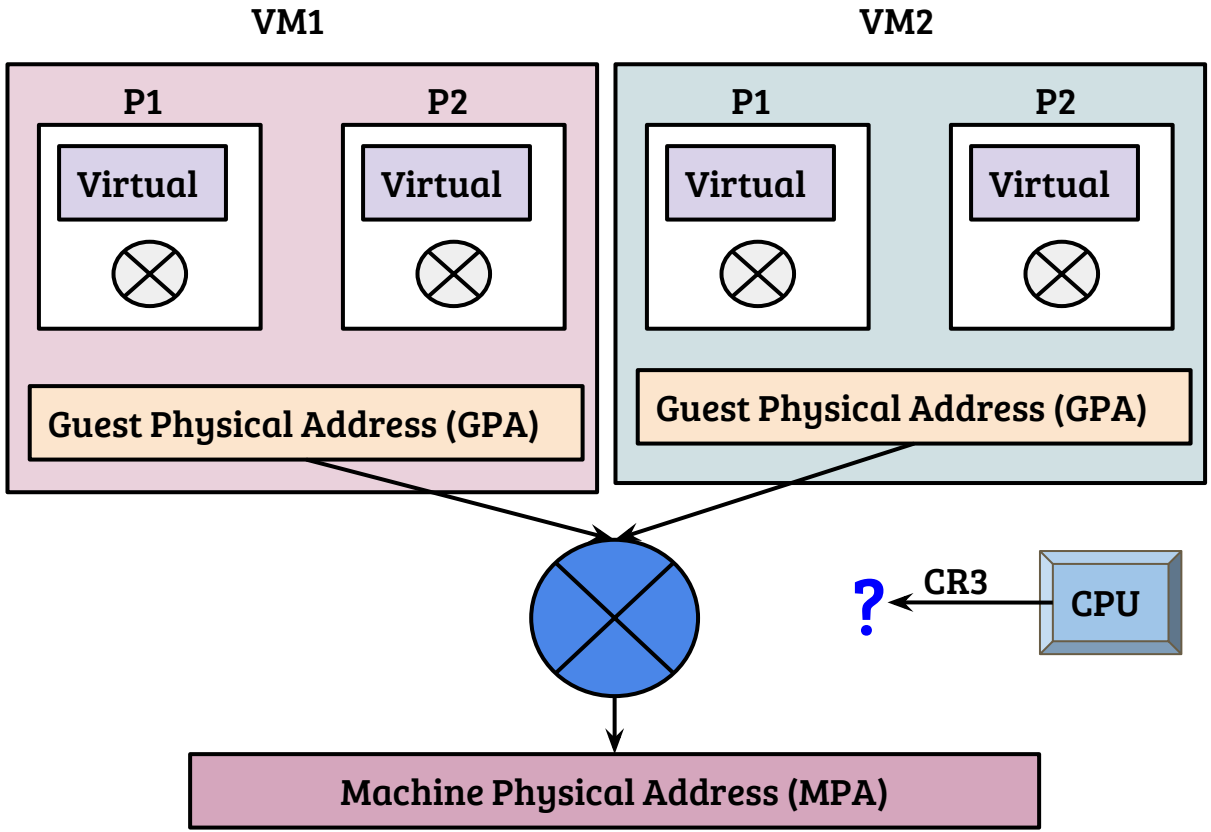


- ➔ Paging is a well known V2P translation scheme
- ➔ Who builds the page table: s/w or CPU?
- ➔ Who walks the page table: s/w or CPU?
- ➔ What happens during context switch?

Example: 4-level page tables (48-bit virtual address)

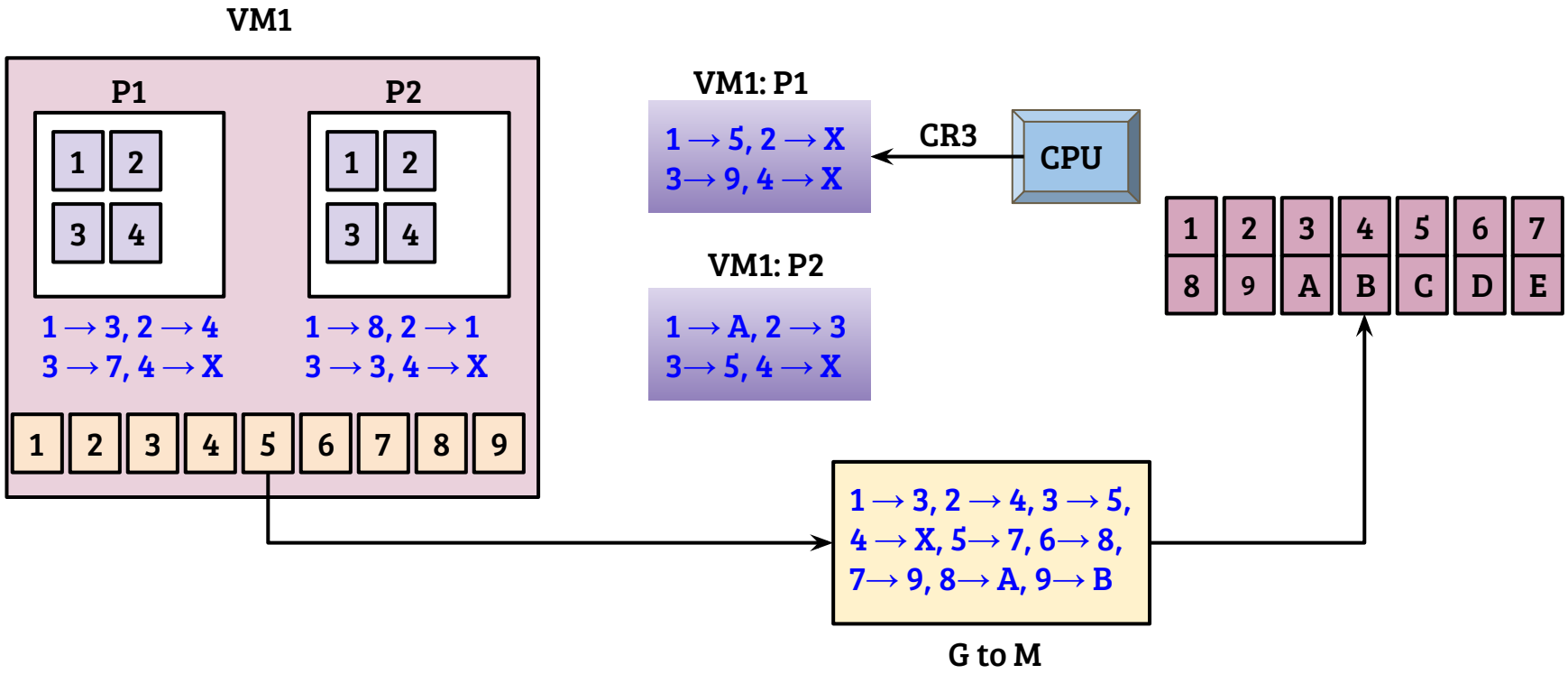


Doubly virtualized memory!

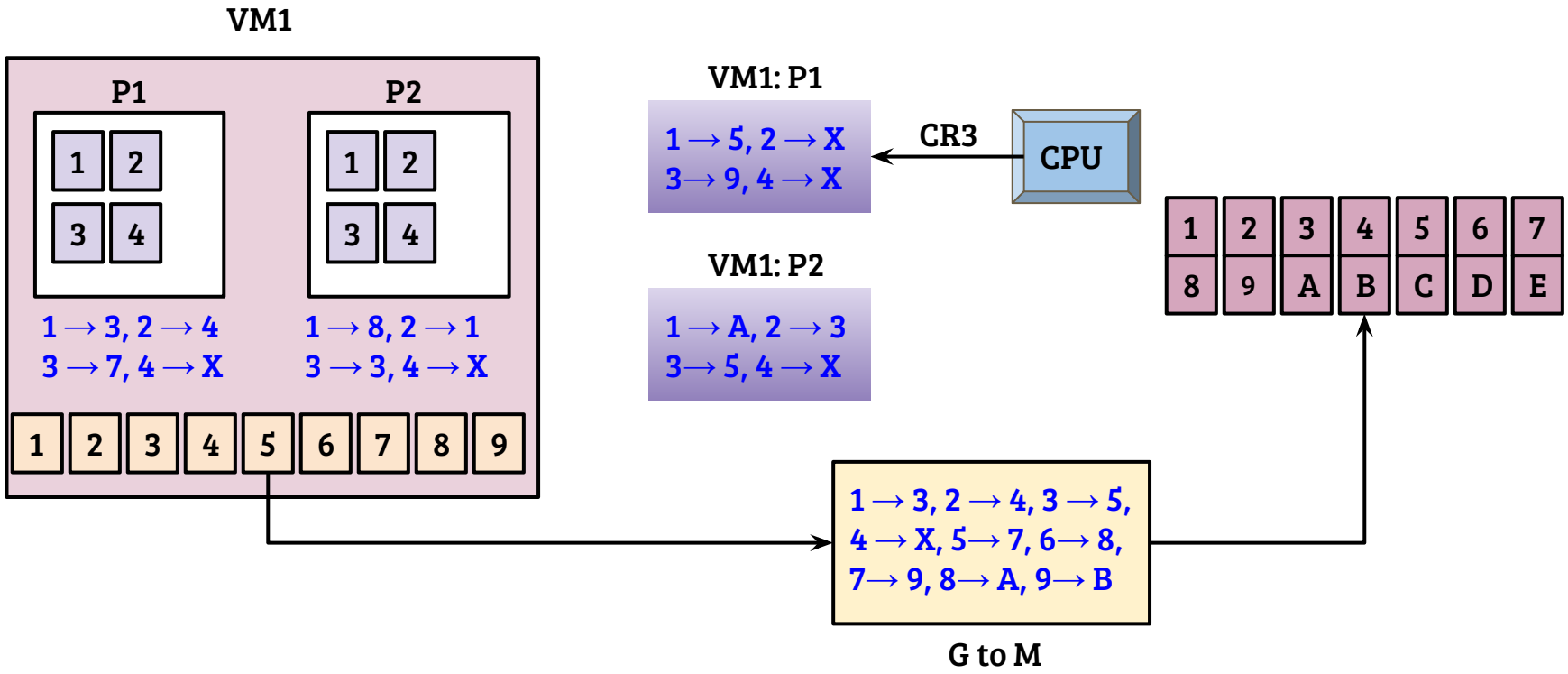


- ➔ Two levels of translation
  - V to P
  - P to M
- ➔ Two types of context switch
  - Intra-VM
  - Inter-VM
- ➔ Two sources of Page fault

Shadow paging: Basic design

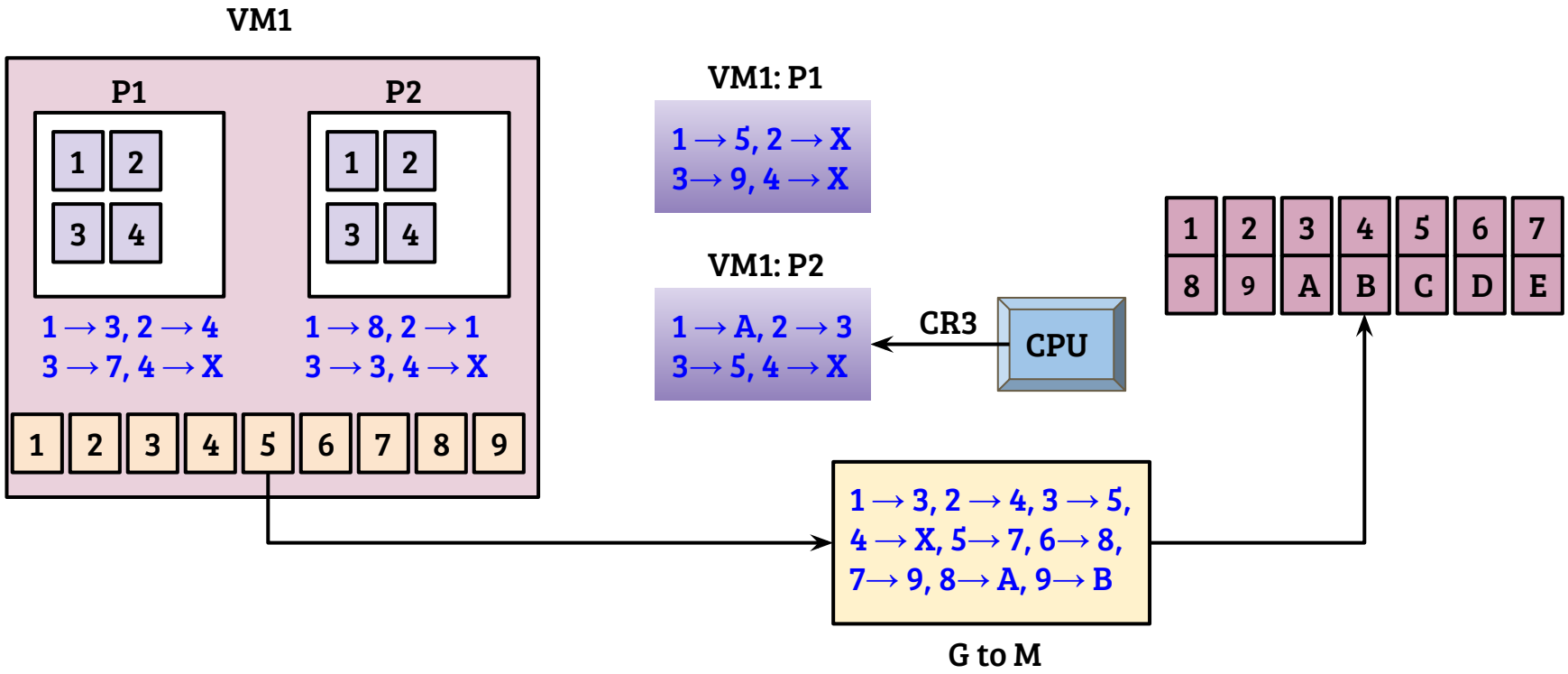


Shadow paging: Basic design

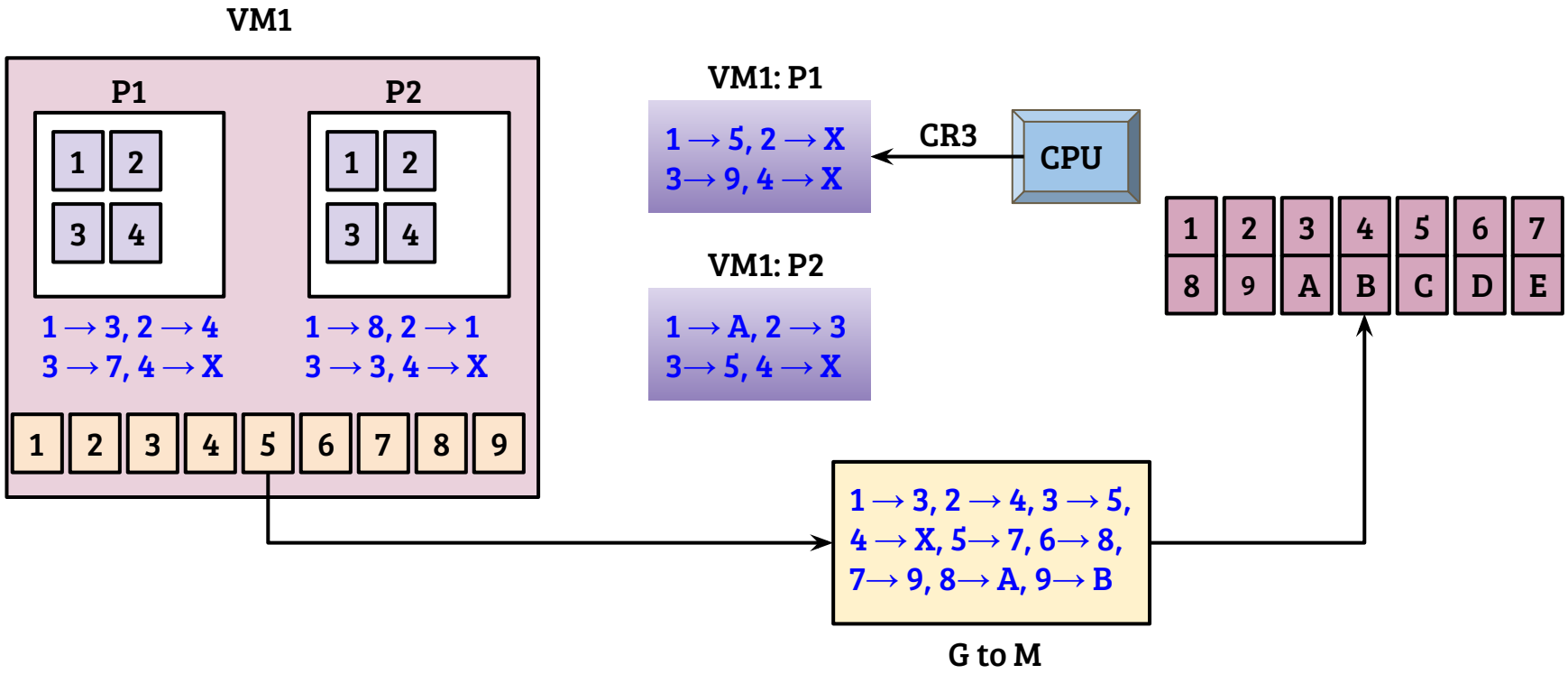


Context switch from P1 to P2

Shadow paging: Basic design

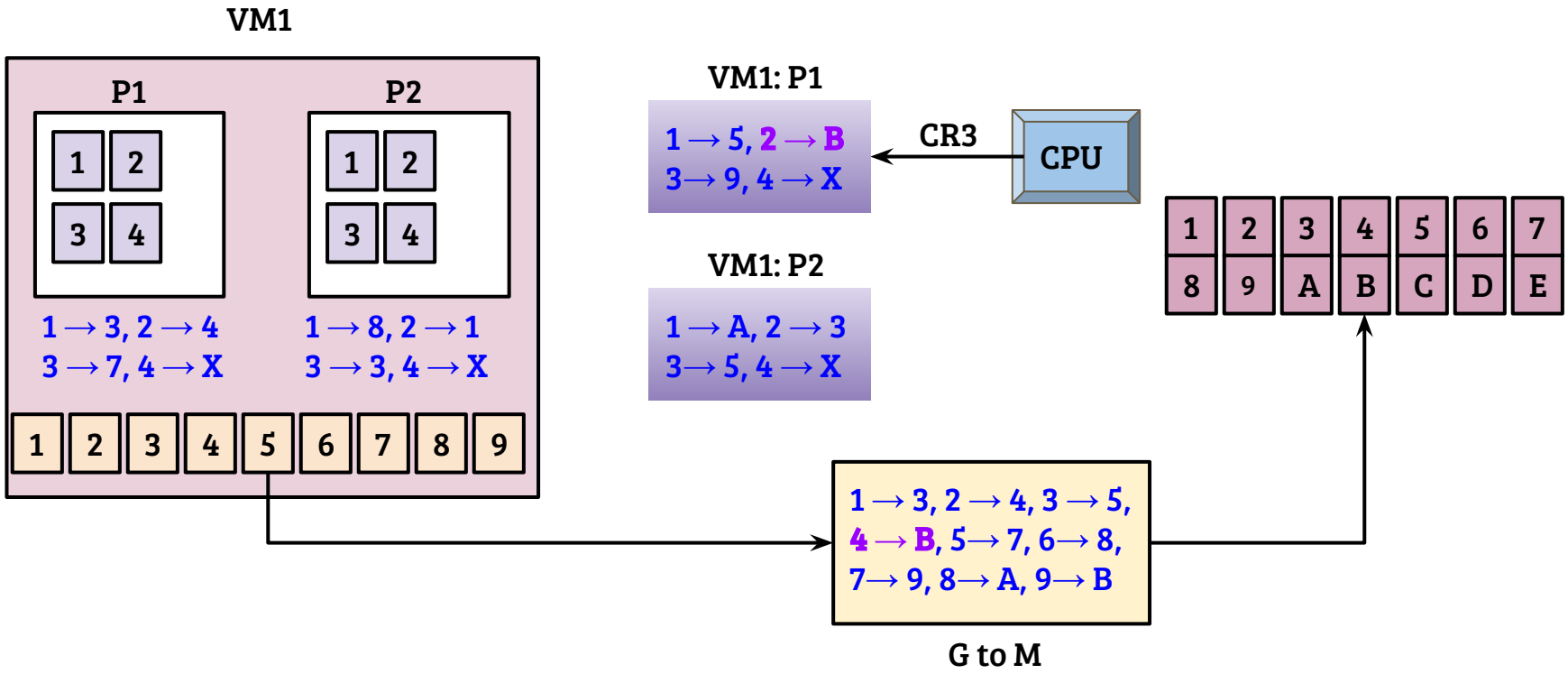


Shadow paging: Page fault handling

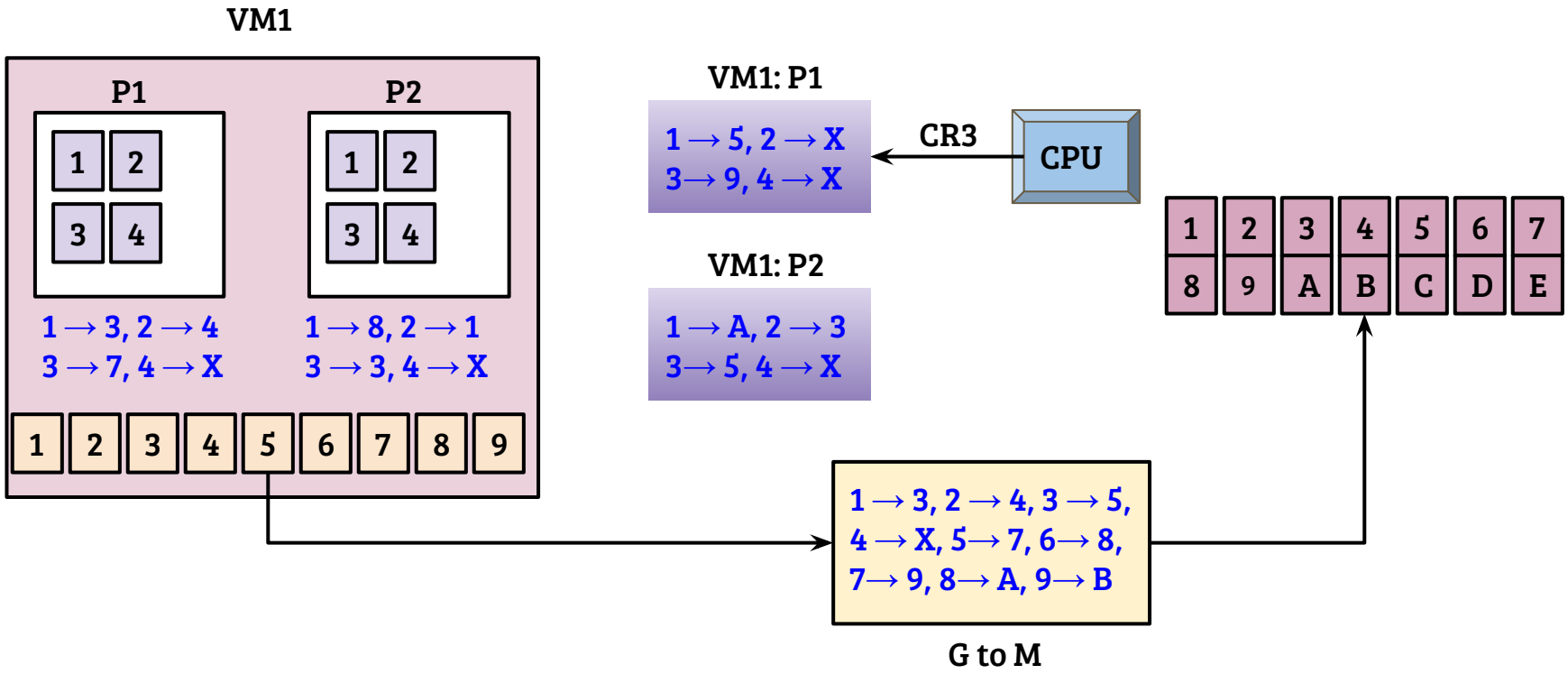


P1 access virtual address 2 → Page fault → Handled @ hypervisor

Shadow paging: Page fault handling

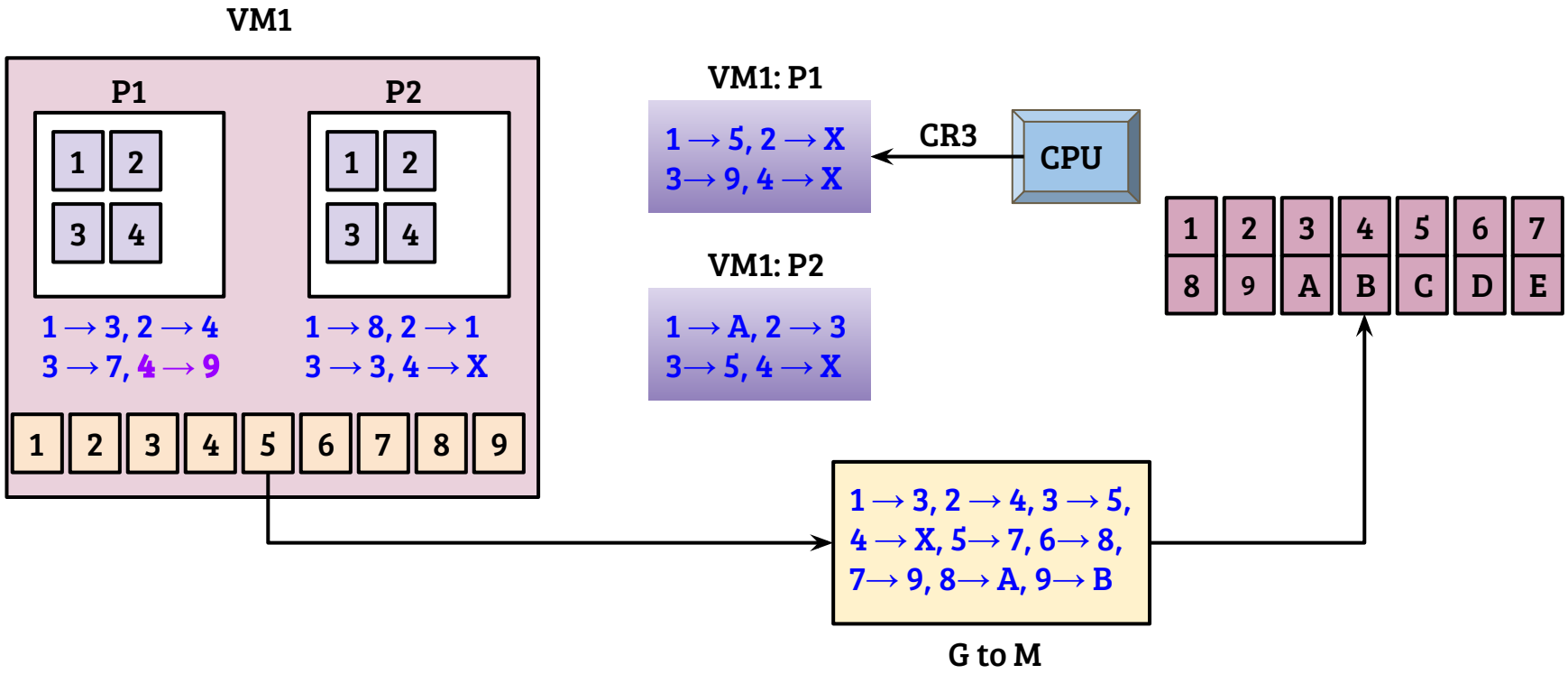


Shadow paging: Page fault handling



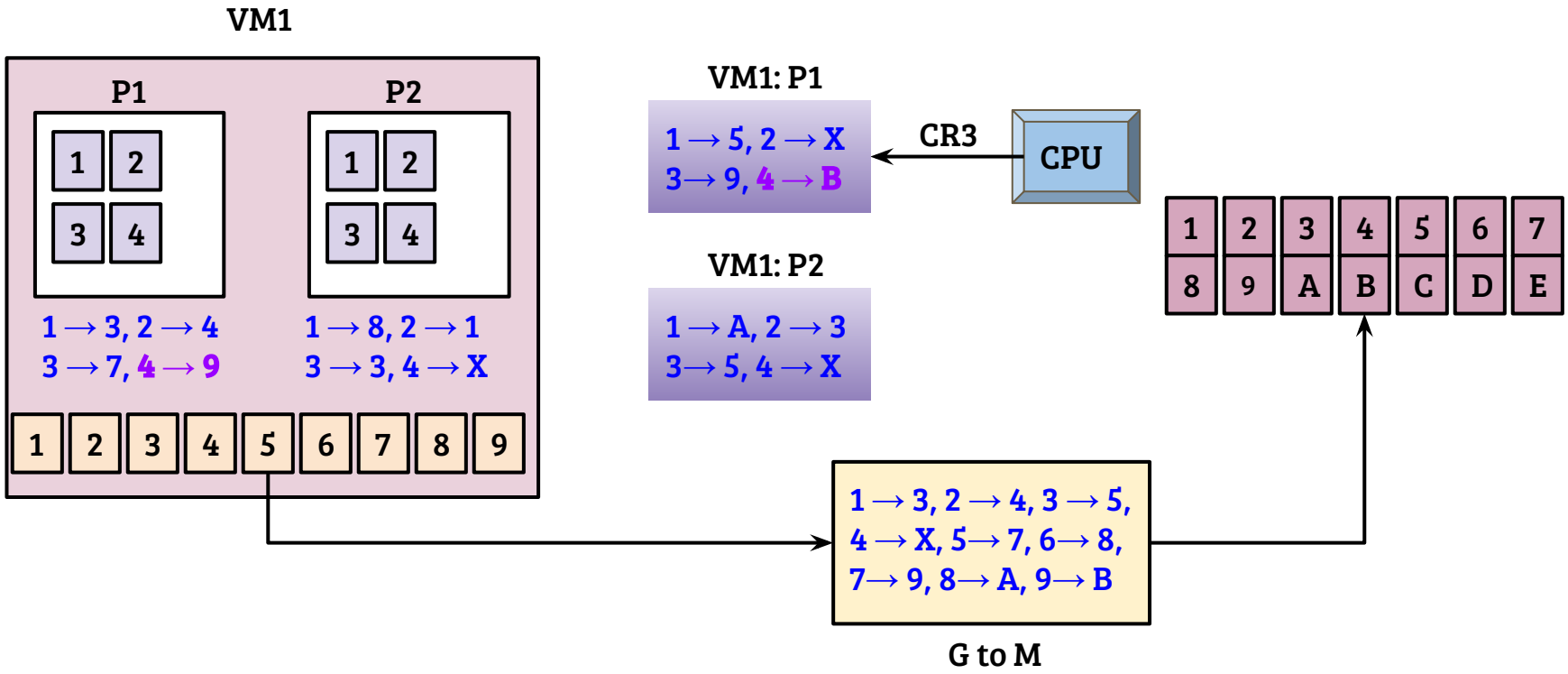
P1 access VA 4 → Page fault →Who handles Page fault?

Shadow paging: Page fault handling



How will the shadow page table be in sync?

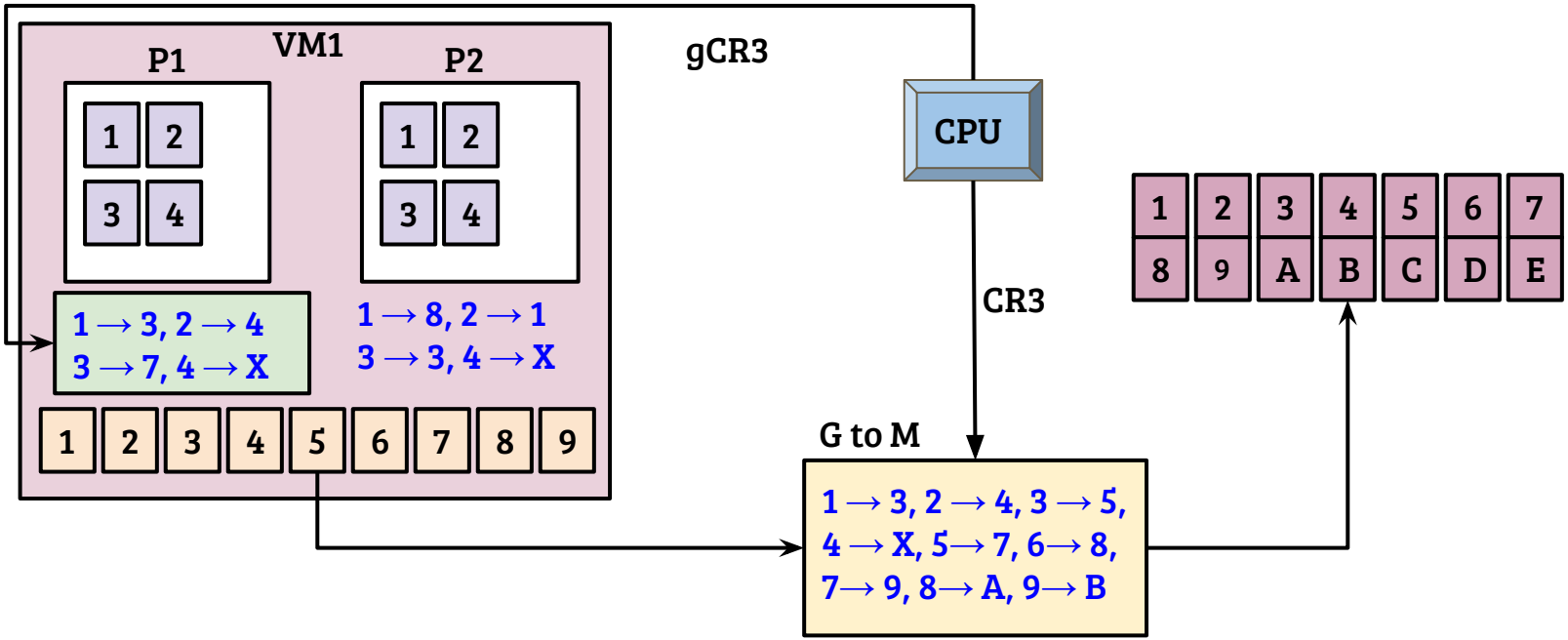
Shadow paging: Page fault handling



Shadow paging: Good, Bad and Ugly!

- ➔ Assume a case when
  - ◆ All virtual addresses are mapped
  - ◆ No updates to page table mappings
  - ◆ Shadow paging performance = ?
  - ◆ TLB effectiveness?
- ➔ How many shadow pages to be maintained?
  - ◆ N VMs with M active processes each
- ➔ Assume a case when
  - ◆ A lot of page faults (memory alloc and dealloc)
  - ◆ Context switch

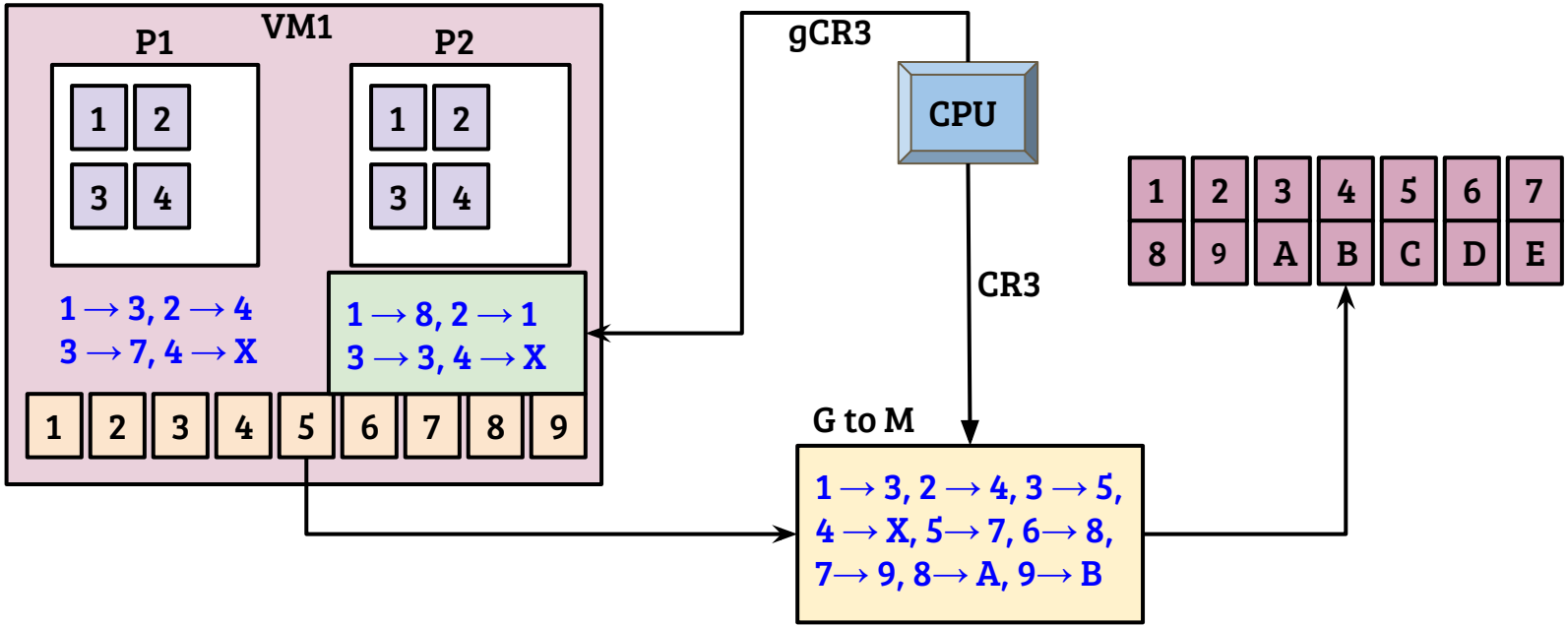
H/W assisted paging: Basic design



Intel Extended Page Tables (EPT)[1] and AMD Nested Page Tables (NPT)[2]

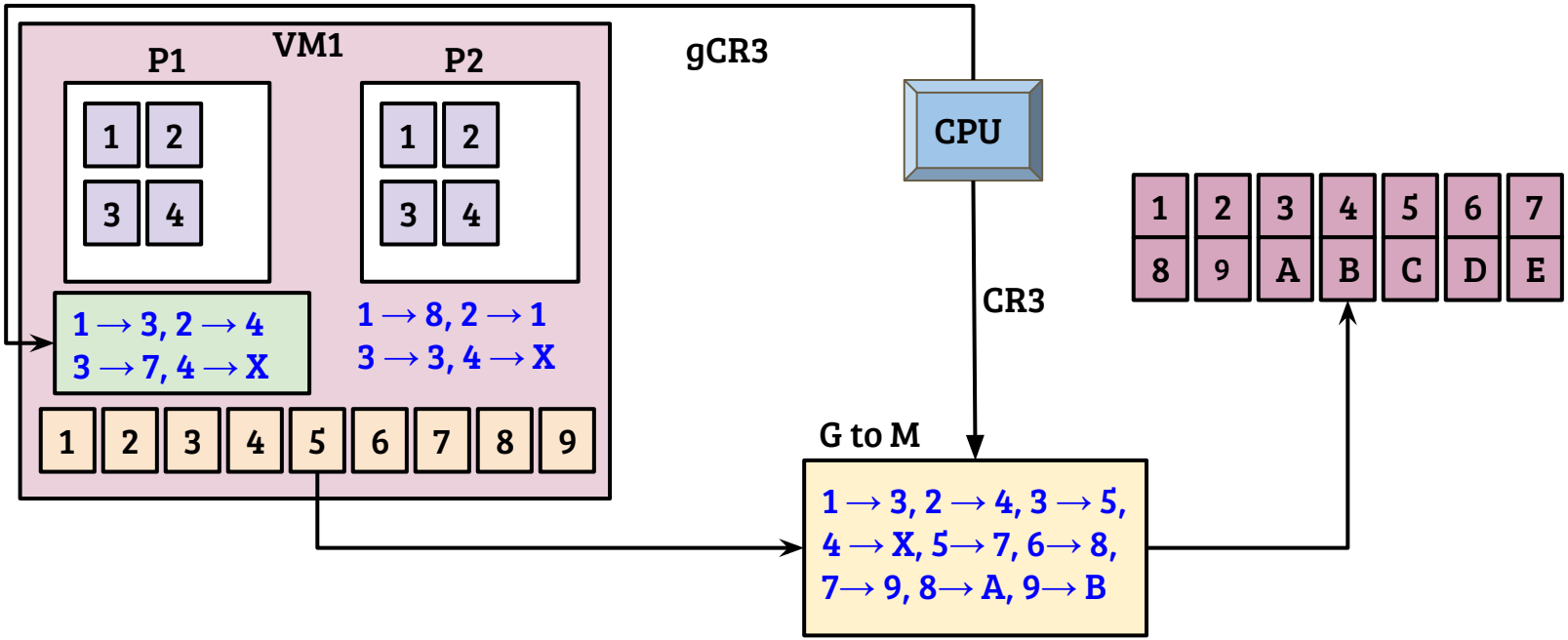
1. https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-3c-part-3-manual.pdf
2. http://developer.amd.com/wordpress/media/2012/10/NPT-WP-1%201-final-TM.pdf

H/W assisted paging: page table management



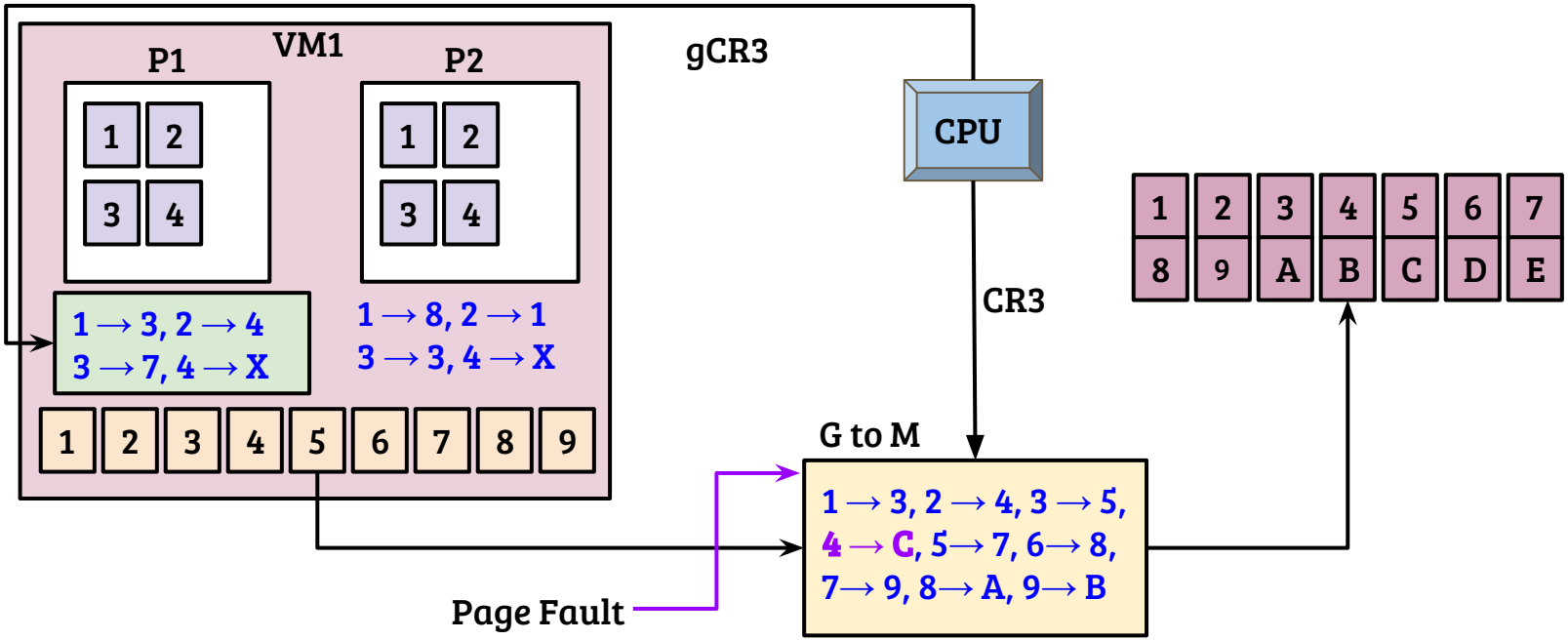
- ➔ Intra-VM context switch does not require hypervisor involvement
- ➔ What about page fault handling?

H/W assisted paging: Page fault handling



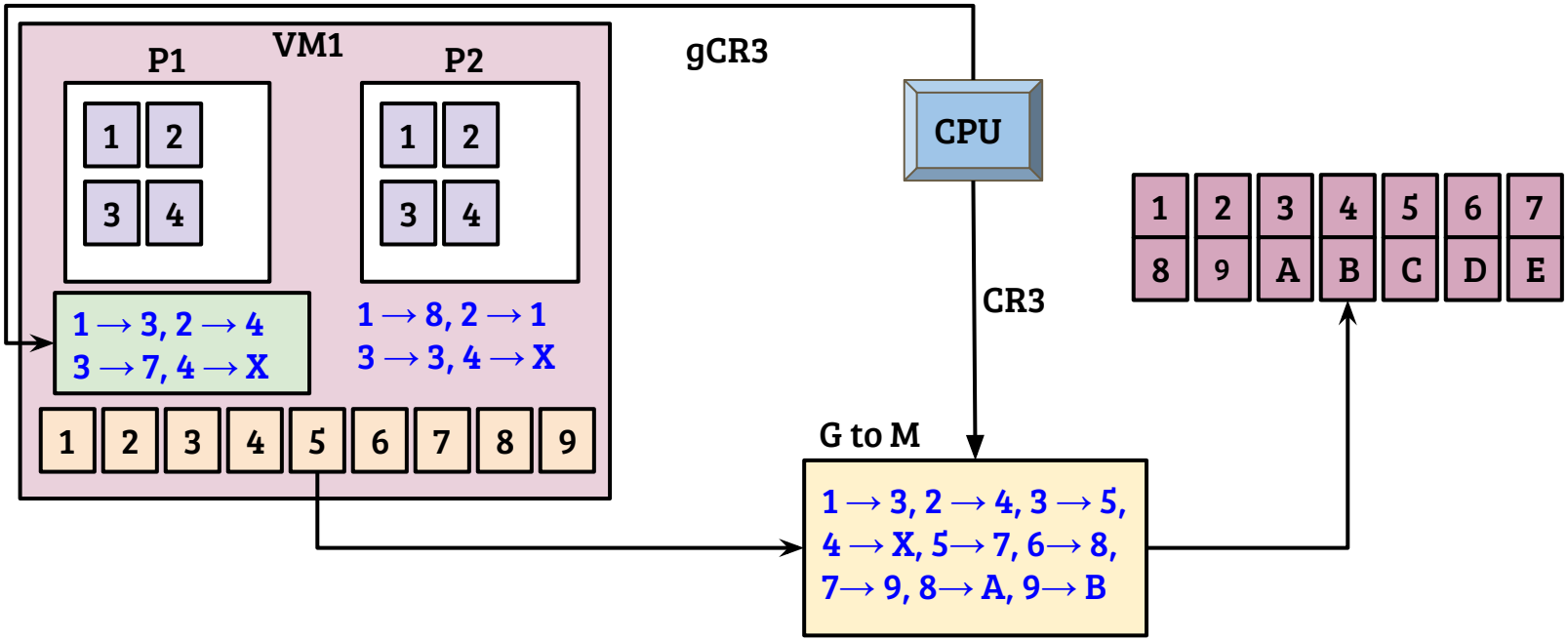
P1 access virtual address 2, how the page fault handled?

H/W assisted paging: Page fault handling



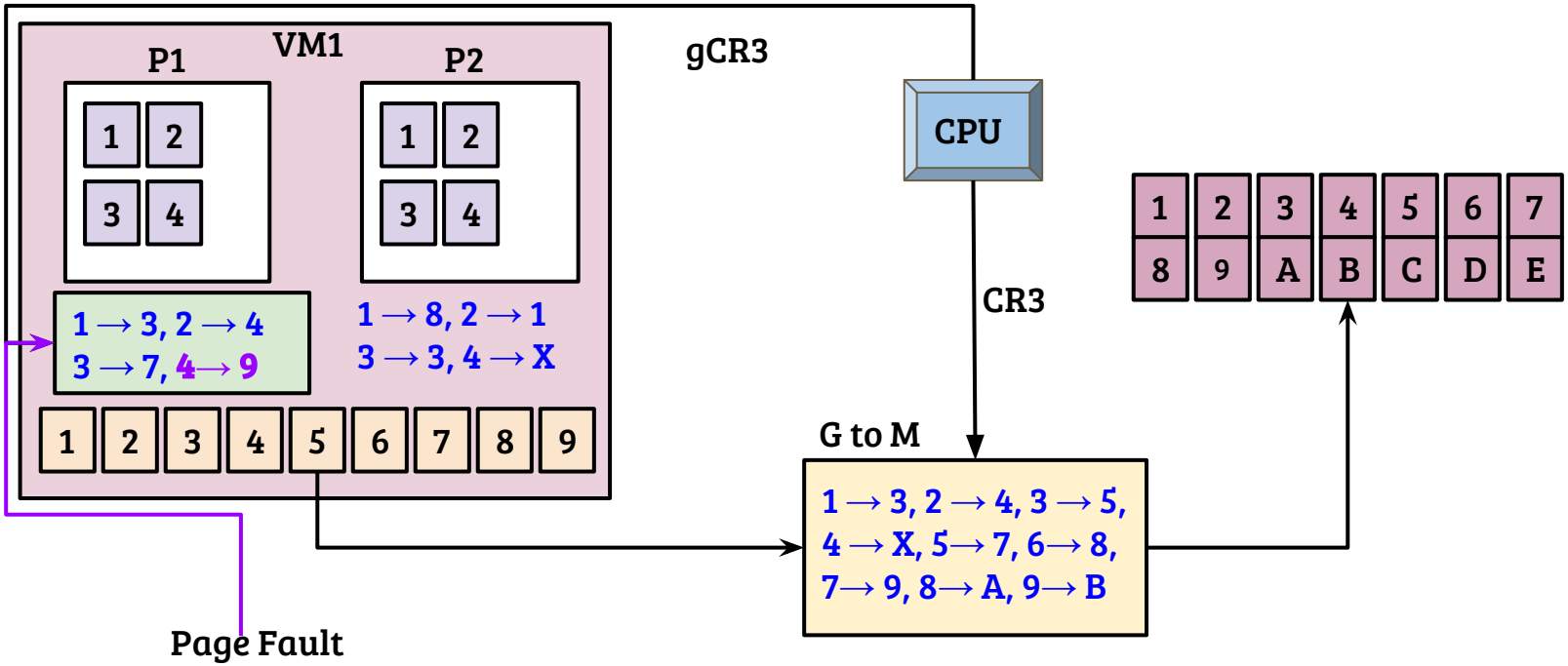
Page fault handled at the hypervisor in a guest OS transparent manner

H/W assisted paging: Page fault handling



P1 access virtual address 4, how the page fault handled?

H/W assisted paging: Page fault handling

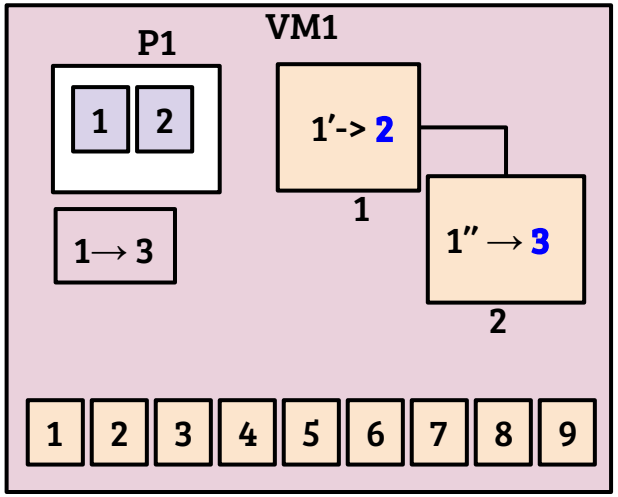


Page fault handled by guest OS → no VMExit and hypervisor involvement

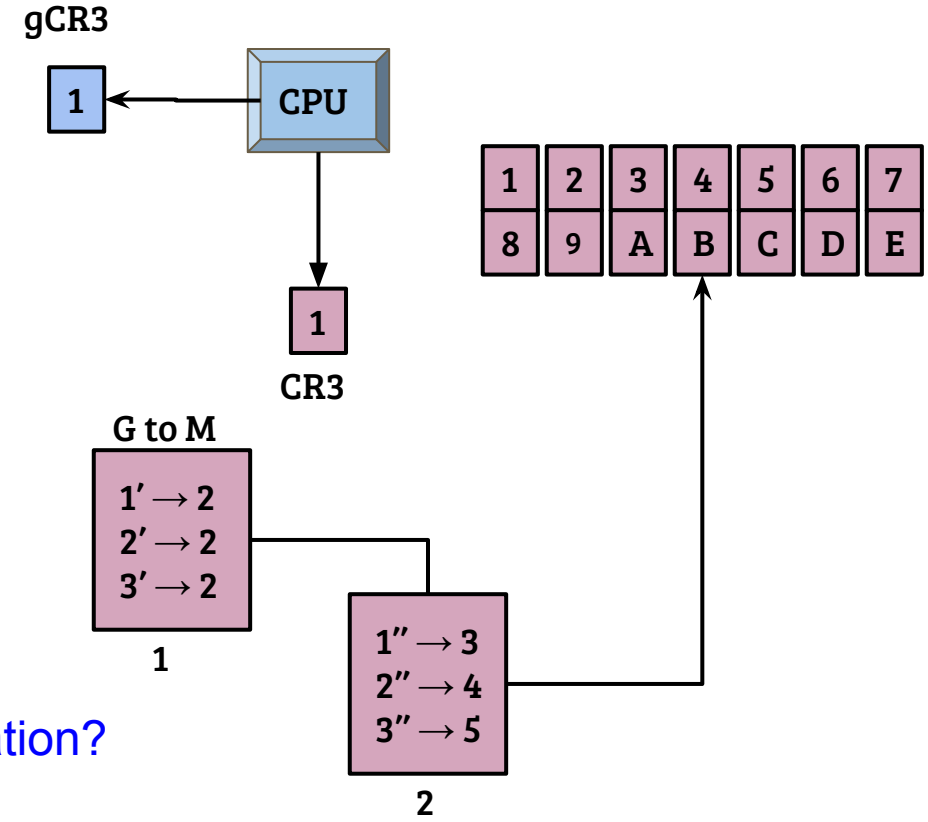
Nested paging: Good, Bad and Ugly!

- ➔ Good
  - ◆ No VMExit, cumbersome page table sync
  - ◆ P to M mapping fixed for all processes
- ➔ Bad
  - ◆ What about TLB efficiency?
  - ◆ One TLB entry in native system == ___ TLB entries in NPT/EPT
- ➔ Ugly
  - ◆ Costly memory translation in case of TLB miss
  - ◆ How costly?

Nested page walk

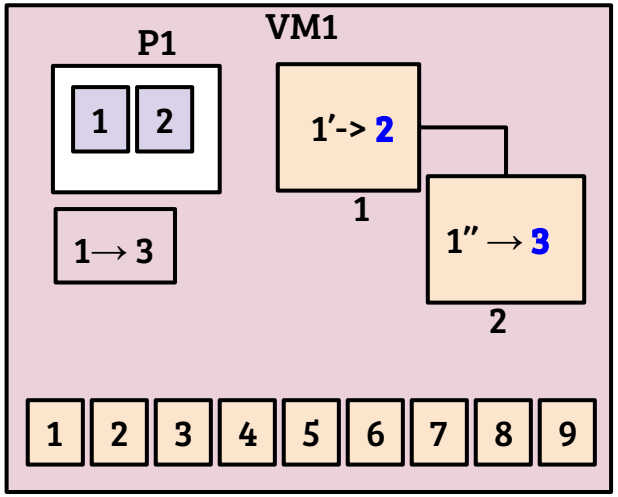


2-level page table, e.g., 32-bit (10 + 10 + 12)

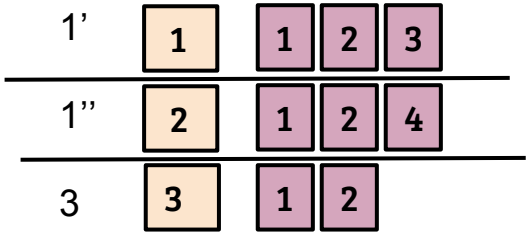


#of memory accesses for translation?

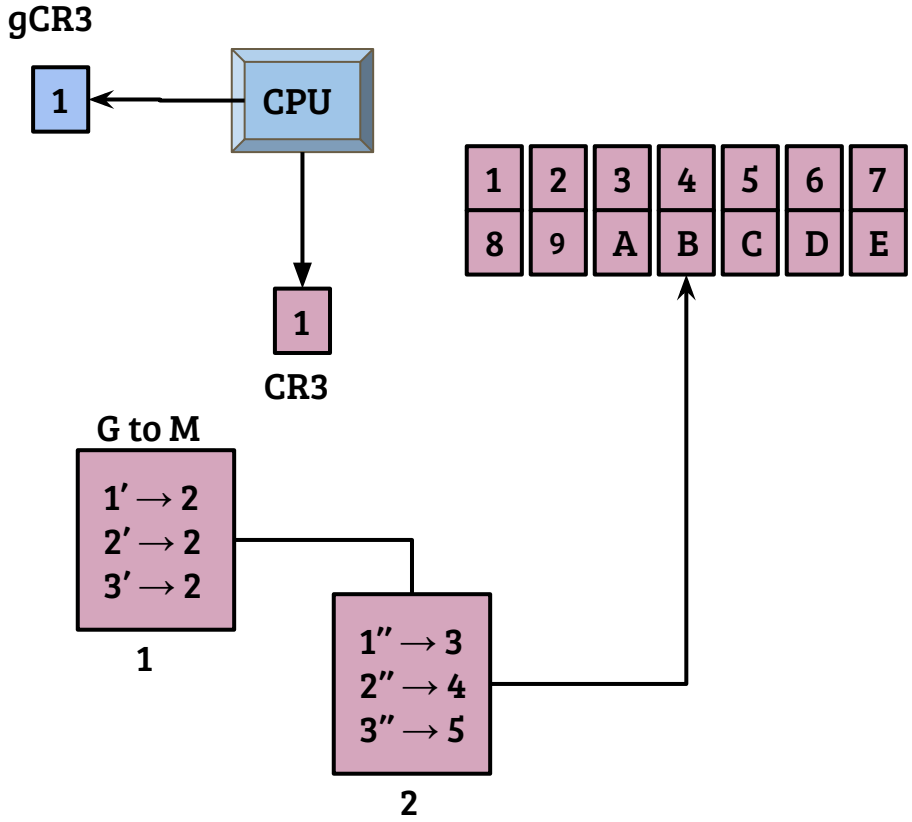
Nested page walk



Memory accesses for translation

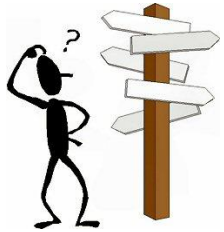


2-level page table, e.g., 32-bit (10 + 10 + 12)



Lost in translation? Here is a summary.

Memory Virtualization



Shadow Paging

- ➔ Near native performance, after page table is in place and not updated
- ➔ Messy interactions across the two layers for correctness and isolation → overheads

EPT/NPT

- ➔ Everything is great when there are no TLB miss
- ➔ Nested walk becomes a bottleneck
  - ◆ Page structure/walk caches
  - ◆ L1/L2/LLC caches

Motivation for a hybrid scheme [1]

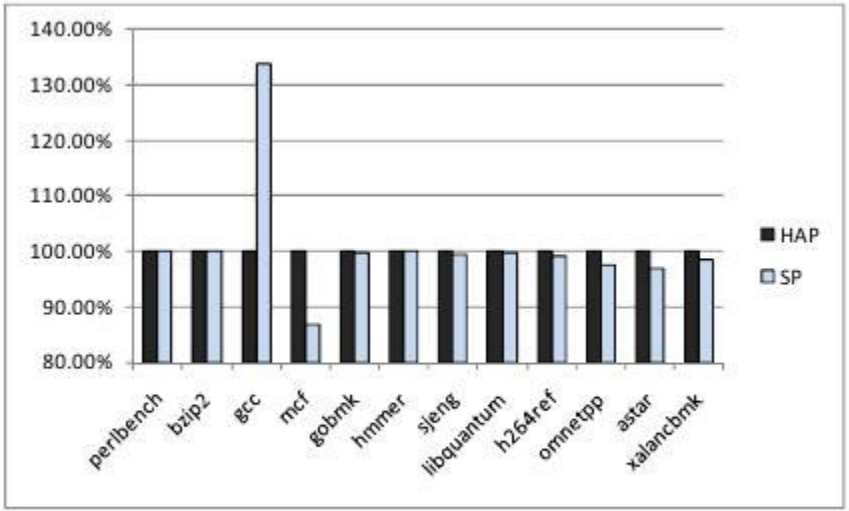


Figure 3. Normalized execution time (SPEC Int).

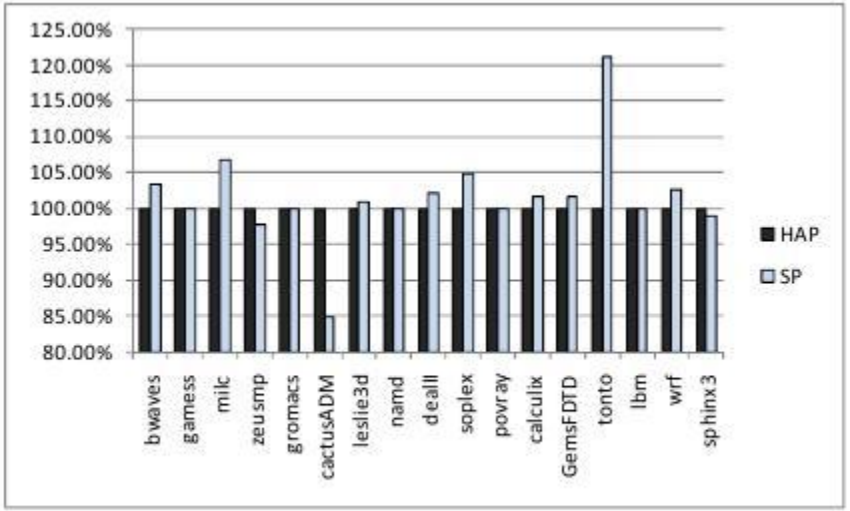


Figure 4. Normalized execution time (SPEC FP)

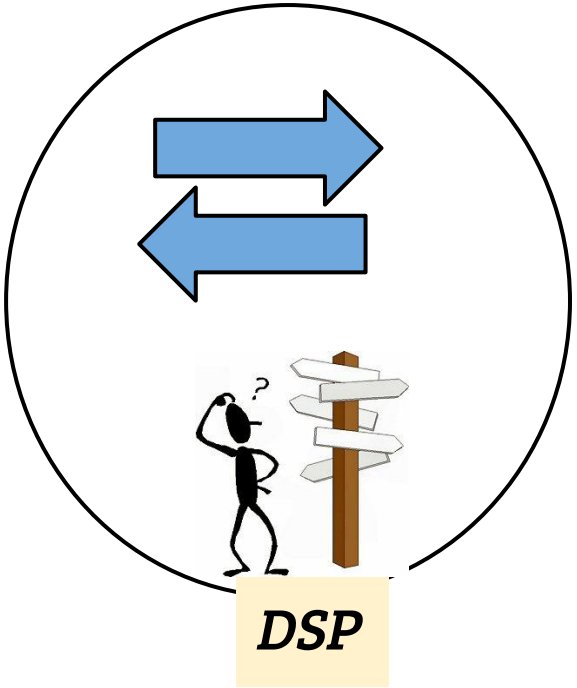
➔ Neither of the techniques is a clear winner.

➔ What is the solution? How to combine?

1. Selective Hardware/Software Memory Virtualization, Wang et.al. VEE 2011, Figures used are from the paper

Dynamic Switching Paging (DSP)[1]

Shadow Paging



EPT/NPT

DSP

➔ Dynamic switching

- How?
- When?

1. Selective Hardware/Software Memory Virtualization, Wang et.al. VEE 2011,

DSP: Switching challenges

- ➔ Where is the switch?
- ➔ Maintain relevant states when paging mode inactive
  - ◆ In SP mode → Maintain EPT structures
  - ◆ In EPT mode → Maintain SP structures

- ➔ Page table pages: RO ← → RW
  - ◆ Unanswered in paper
- ➔ TLB flush or no TLB flush?
  - ◆ Unanswered in paper
  - ◆ Carry forward TLB → Any issues?

DSP: Switching strategy

- ➔ Players in the game: TLB misses and VMExits
  - ◆ Both monitored continuously
  - ◆ In EPT-mode
    - How to monitor VMExits?
    - VMExits ~ _____
- ➔ Cost (VMExit) vs. Cost (TLB misses)
- ➔ DSP uses thresholds, why?