# 虛擬化技術
# *Virtualization Technique*

## System Virtualization

Memory Virtualization

## Agenda

- Basic concept of memory management
- Brief introduction to ARM v7 AMSA
- Memory Virtualization
  - Shadow page table
  - Hardware assistance
  - Comparison

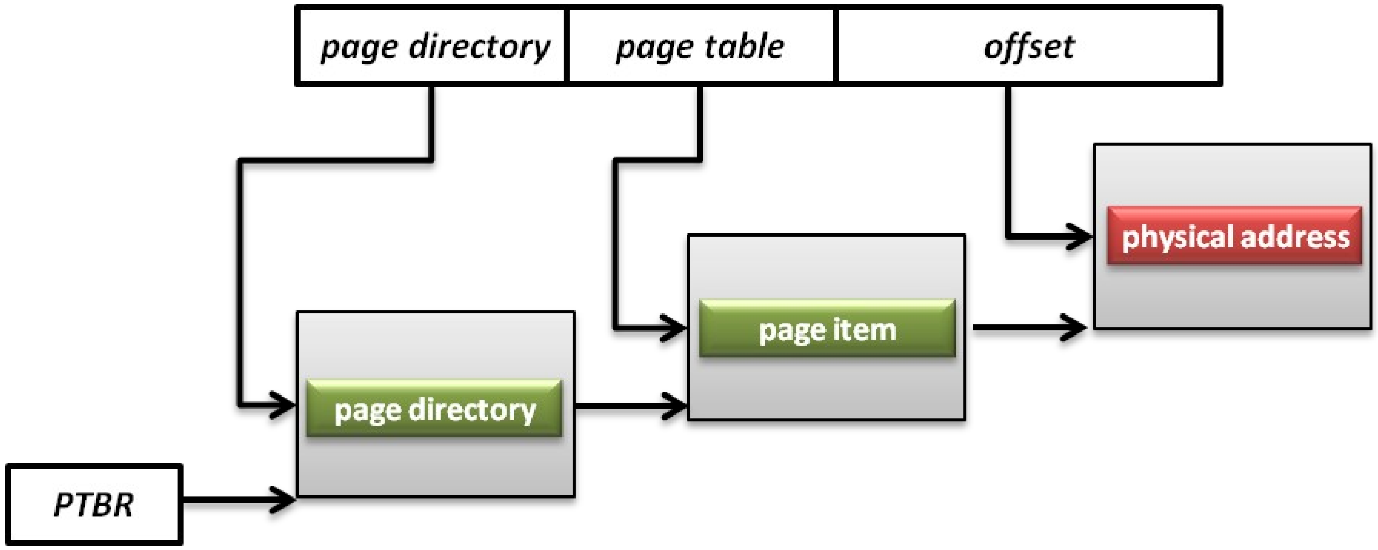
## *Basic concept of memory management*

## *Memory Management Unit*

- A hardware component responsible for handling accesses to memory requested by the CPU
  - Address translation: virtual address to physical address (VA to PA)
  - Memory protection
  - Cache control
  - Bus arbitration
- Page tables are maintained by operating system, and MMU only references them.
- TLB updates are performed automatically by page-table walking hardware

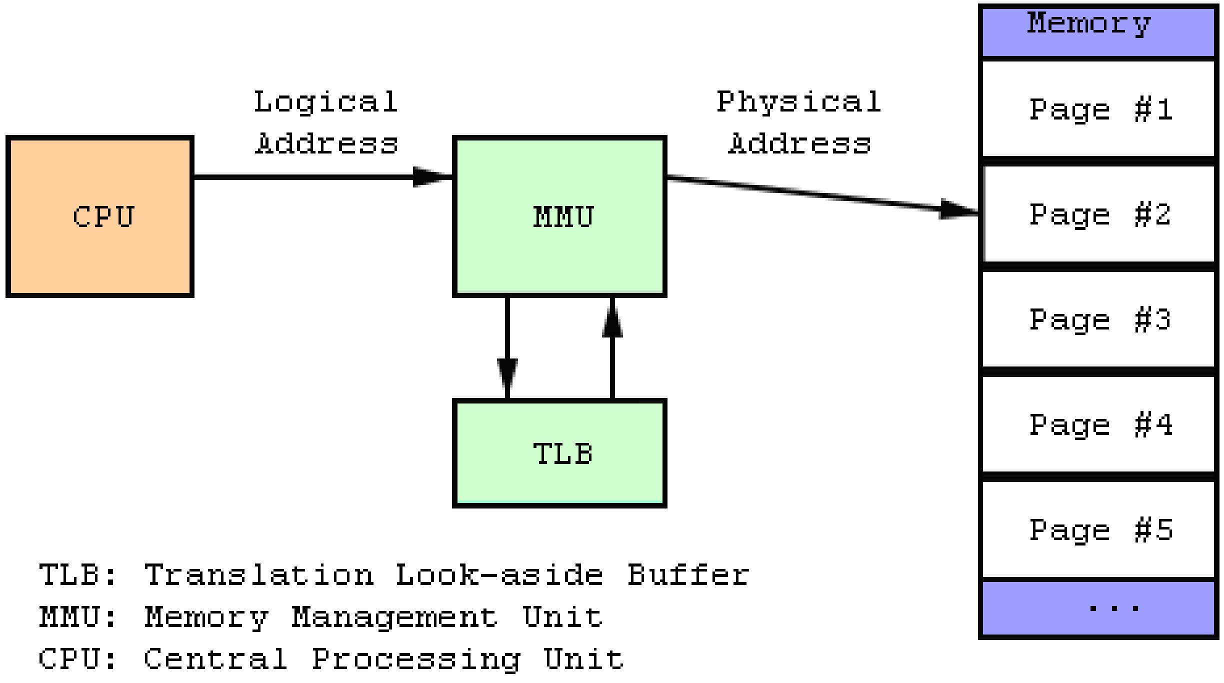
## *Page Tables*

- A page table is the data structure used by a virtual memory system to store the ma pping between virtual addresses and physical addresses
- Translation table base register(TTBR)
  - Also called page table base register
  - A register that stores the address of the base page table for MMU



## TLB

- Translation look-aside buffer
  - A CPU cache that memory management hardware uses to improve virtual address translation speed
  - The TLB is typically implemented as content-addressable memory (CAM)
  - The CAM search key is the virtual address and the search result is a physical address


CPU
Logical Address
MMU
Physical Address
TLB
Memory
Page #1
Page #2
Page #3
Page #4
Page #5
...
TLB: Translation Look-aside Buffer
MMU: Memory Management Unit
CPU: Central Processing Unit

## *Brief introduction to ARMv7 VMSA*

## ARMv7 VMSA

- VMSA: Virtual memory system architecture
- Support four mapping types
  - 16MB Super section
  - 1MB Section
  - 64KB Large page
  - 4KB Small page

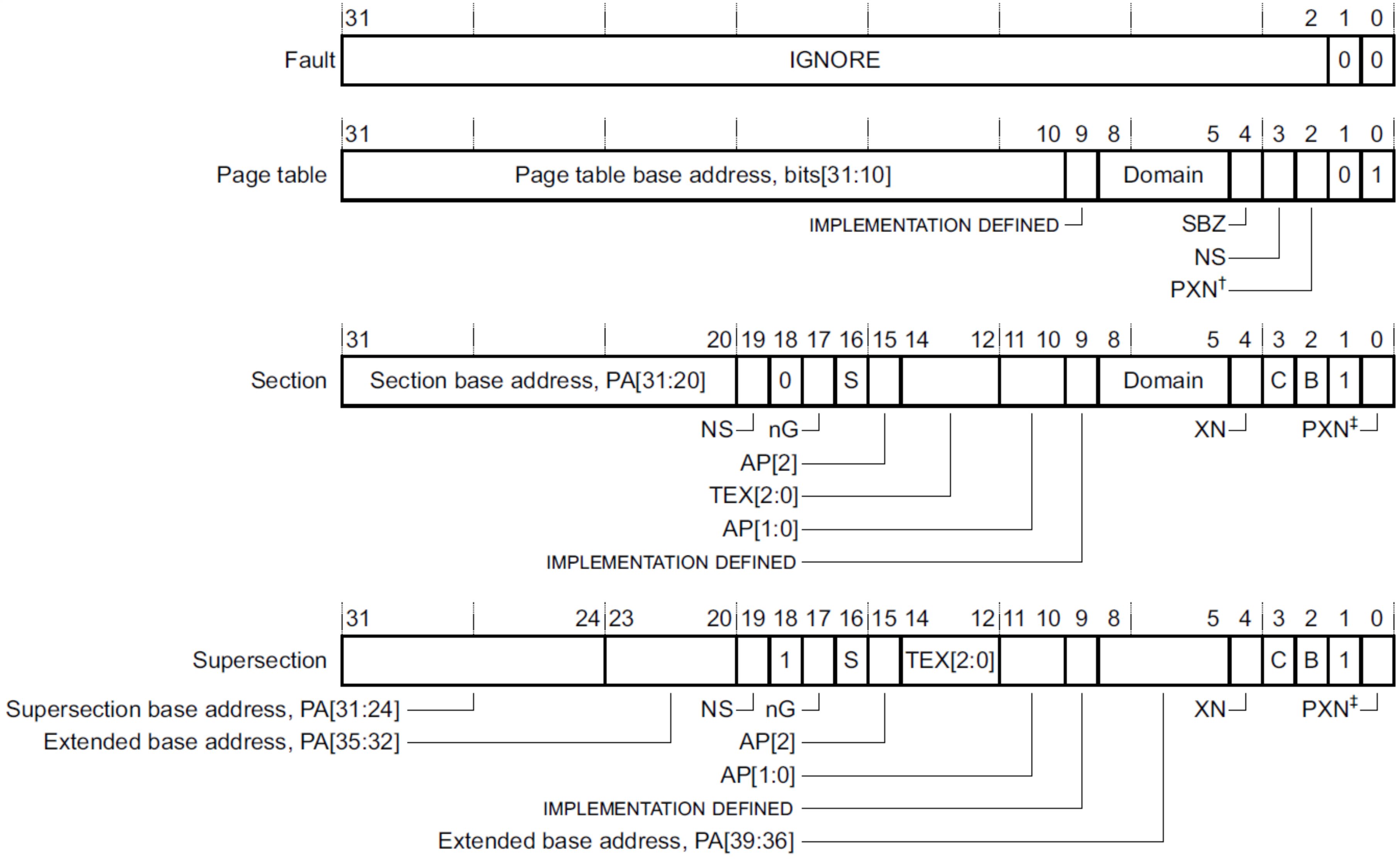
- Basically a 32-bit two-layer translation table
- Other features (virtualization extension and 64-bit large physical address extensio n) would be mentioned in the future class

## L1 Page Table

- First-level table
  - Translation properties for a Section and Super-section
  - Translation properties and pointers to a second-level table for a Large page or a Small page.

- L1 descriptor[1:0]
  - 0b00, invalid or fault entry
    - Translation fault
  - 0b01, page table
    - Address of 2nd level translation table
  - 0b10, section of super-section
  - 0b11, special usage with PXN attribute

# 1st Page Table Descriptor



## L2 Page Table

- Second-level tables
  - The base address and translation properties for a Small page or a Large page.
  - That is, page tables
  - 1KB
- L2 descriptor[1:0]
  - 0b00, invalid or fault entry
    - Translation fault
  - 0b01, large page
  - 0b1x, small page

## 2nd Page Table Descriptor

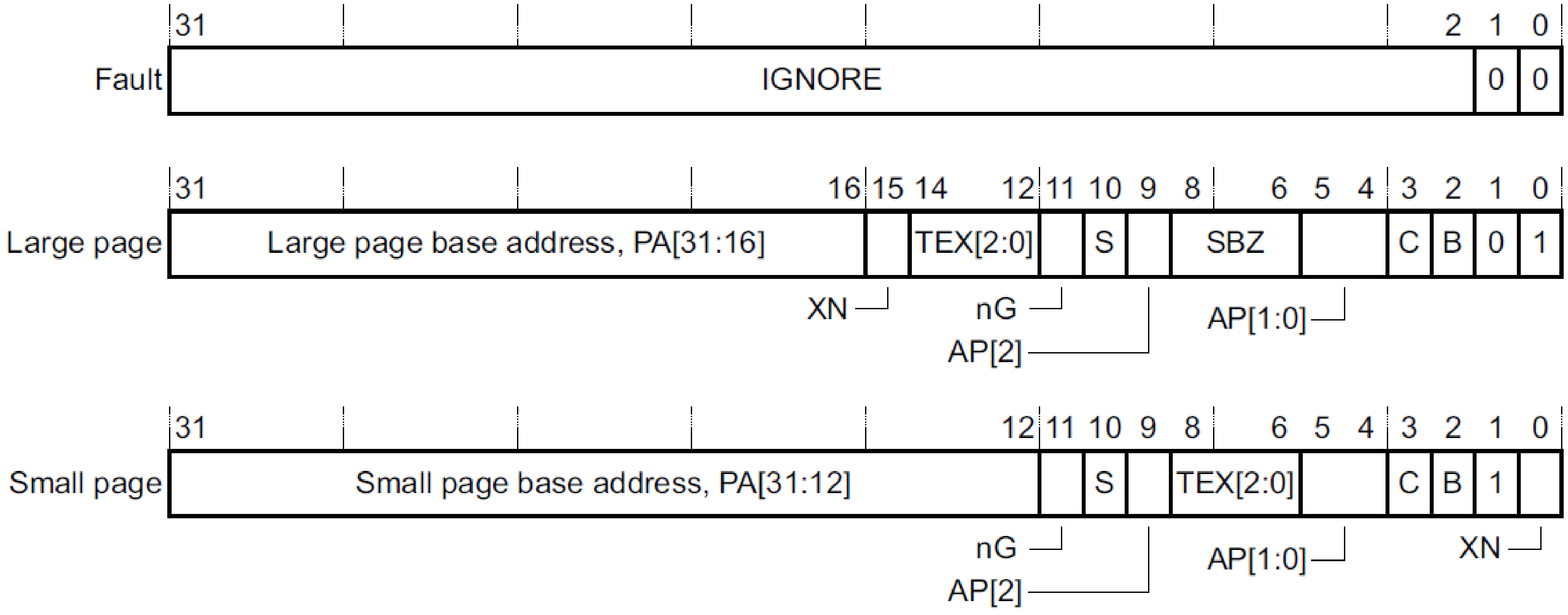


**Figure B3-5 Short-descriptor second-level descriptor formats**

Concepts
Shadow page table
Hardware assistance
Comparison

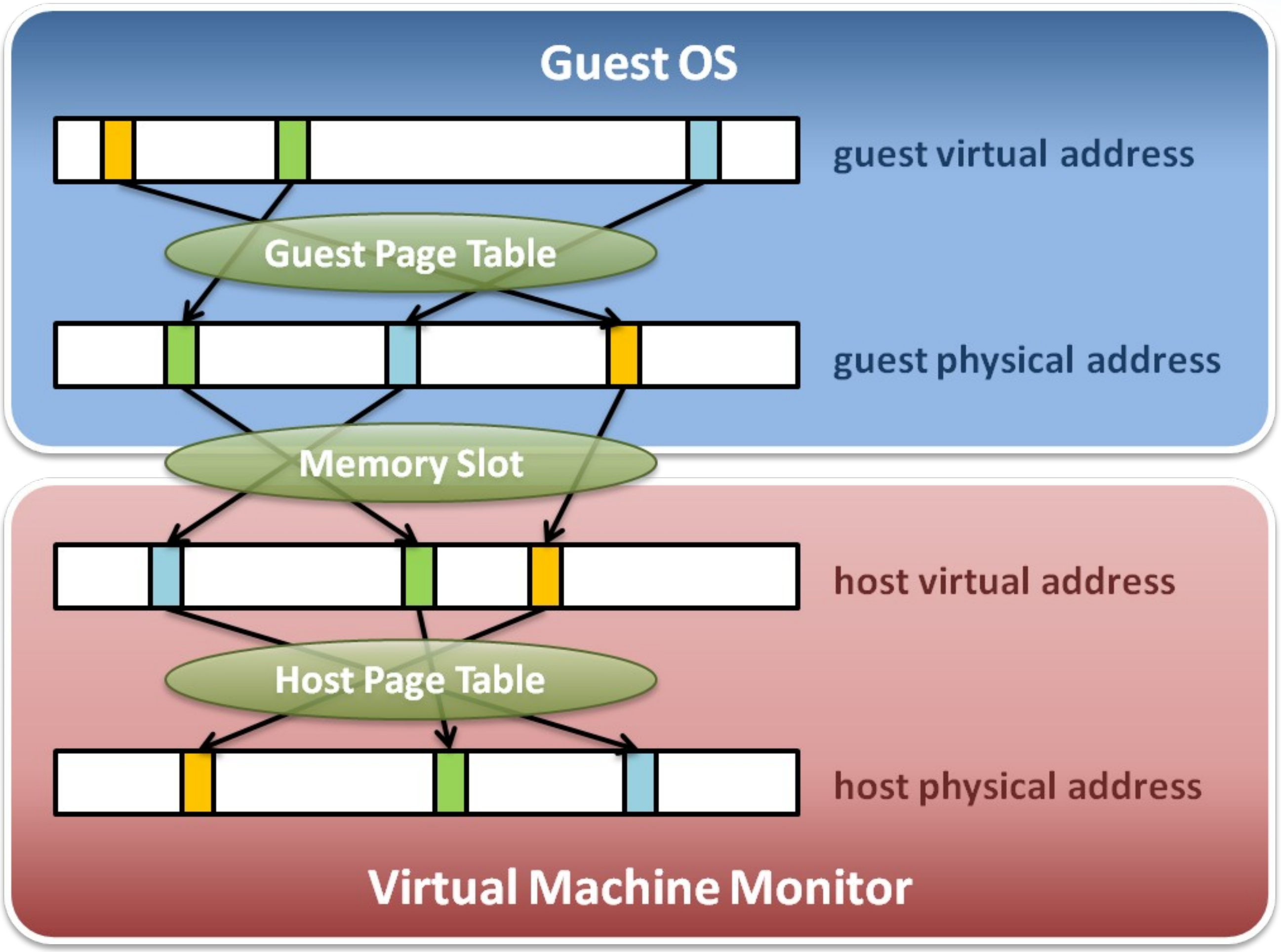
## *Memory Virtualization*

## *Memory Management on a VM*

- Traditionally, OS fully controls all physical memory space and provides a continuo us addressing space to each process
- Guest OS is just one of user space processes of host OS
- If guest OS is allowed to access the physical memory arbitrarily, then what happen s?
- In system virtualization, VMM should make all virtual machines share the physical memory space

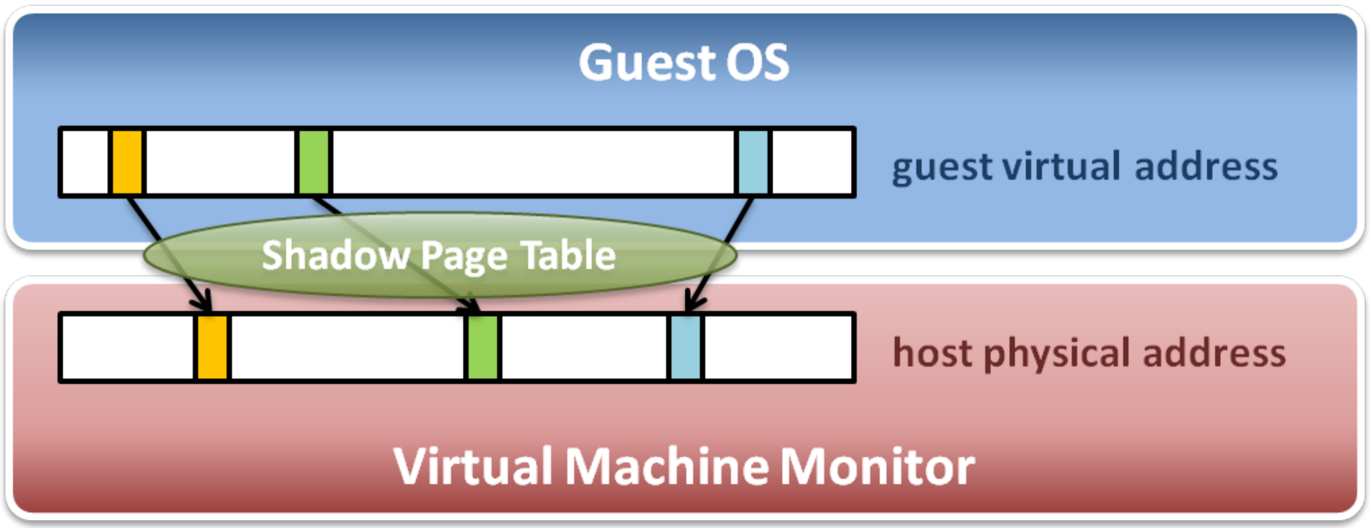
# *Memory Virtualization*

- Memory virtualization architecture



## Memory Virtualization

- The performance drop of memory access is usually unbearable. VMM needs furthe r optimization.
- VMM maintains shadow page tables :
  - Direct virtual-to-physical address mapping
  - Use hardware TLB for address translation



## Goals of Memory Virtualization

- Address Translation
  - Control table-walking hardware that accesses translation tables in main memory.
- Memory Protection
  - Define access permission which uses the Access Control Hardware.
- Access Attribute
  - Define attribute and type of memory region to direct how memory operation to be handled.

- How to implement?
  - Software solution: shadow page table
  - Hardware solution
    - NPT on SVM from AMD
    - EPT on VMX from Intel
    - ARM v7 VMSA (Virtual Memory System Architecture) with virtualization extension

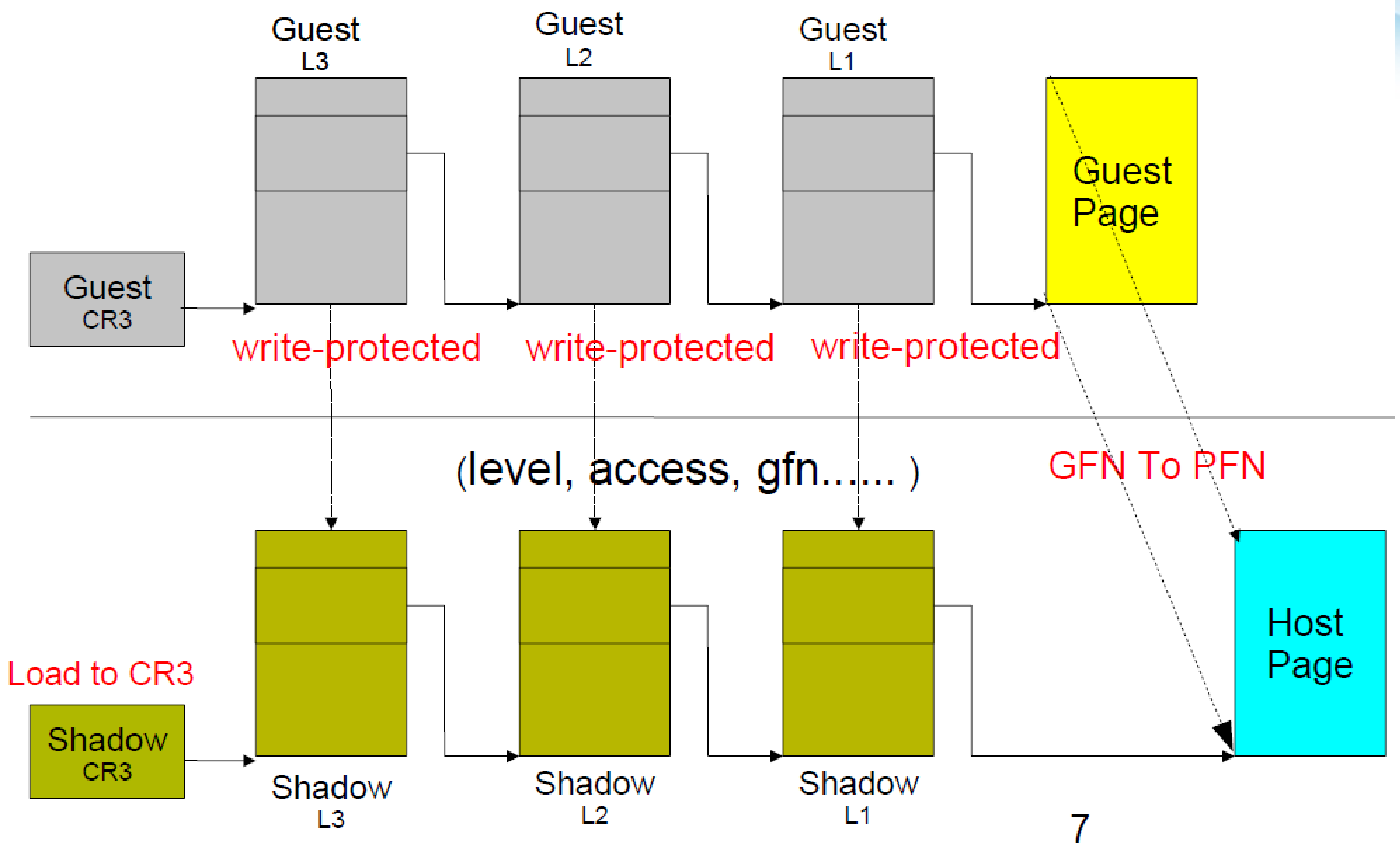
Concepts
Shadow page table
Hardware assistance
Comparison

## *Memory Virtualization*

## Shadow Page Table

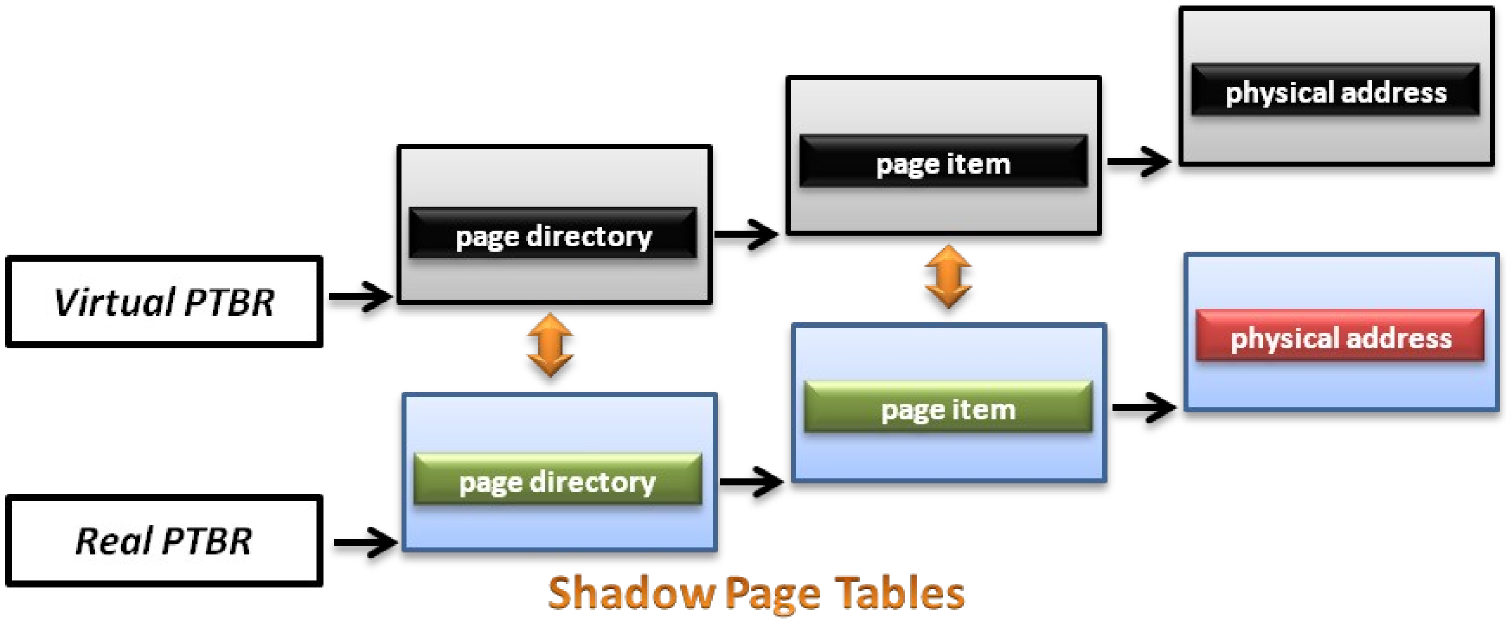
- Map guest virtual address to host physical address
  - Shadow page table
    - Guest OS will maintain its own virtual memory page table in the guest physical memory frames.
    - For each guest physical memory frame, VMM should map it to host physical memory frame.
    - Shadow page table maintains the mapping from guest virtual address to host physical address.
  - Page table protection
    - VMM will apply write protection to all the physical frames of guest page tables, which lead the guest page table write exception and trap to VMM.

# Shadow Page Table: Overview



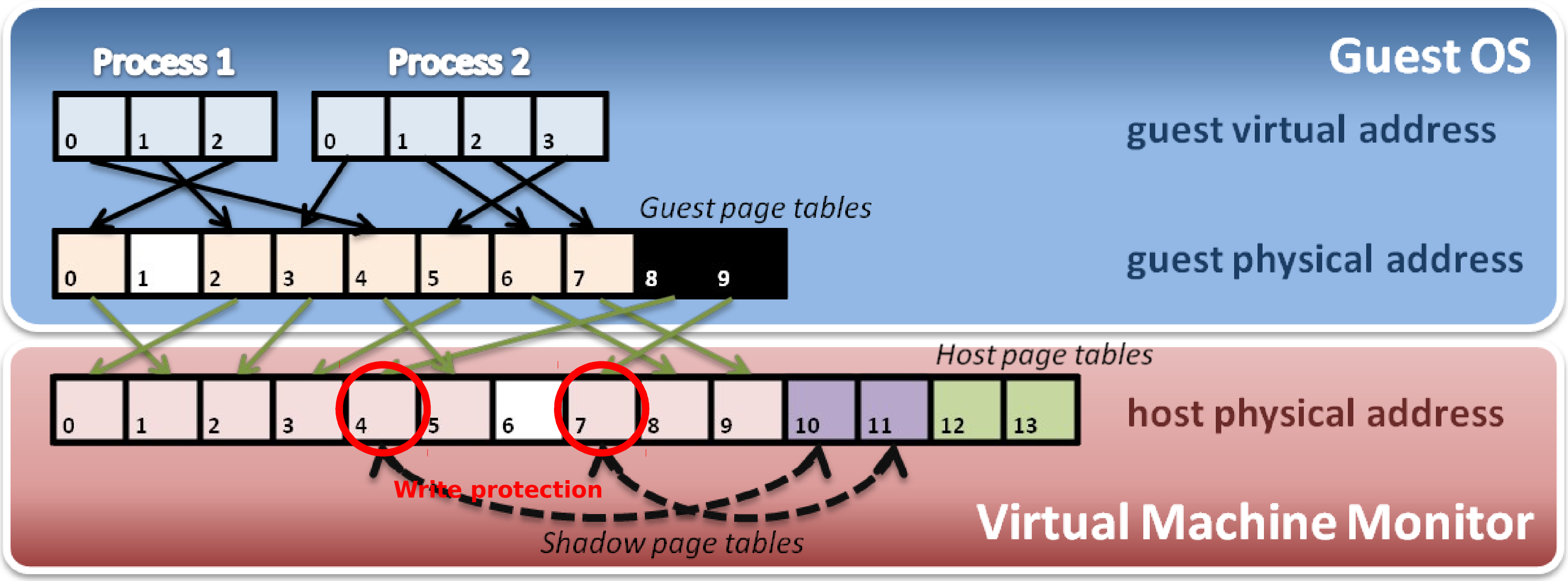
## Shadow Page Table

- How does this technique work ?
  - VMM should make MMU virtualized
    - VMM manages the real PTBR and a virtual PTBR for each VM
    - When a guest OS is activated, the real PTBR points to the corresponding shadow page table of the guest OS
    - When the guest OS attempts to modify the PTBR, it will be intercepted by VMM for further emulation



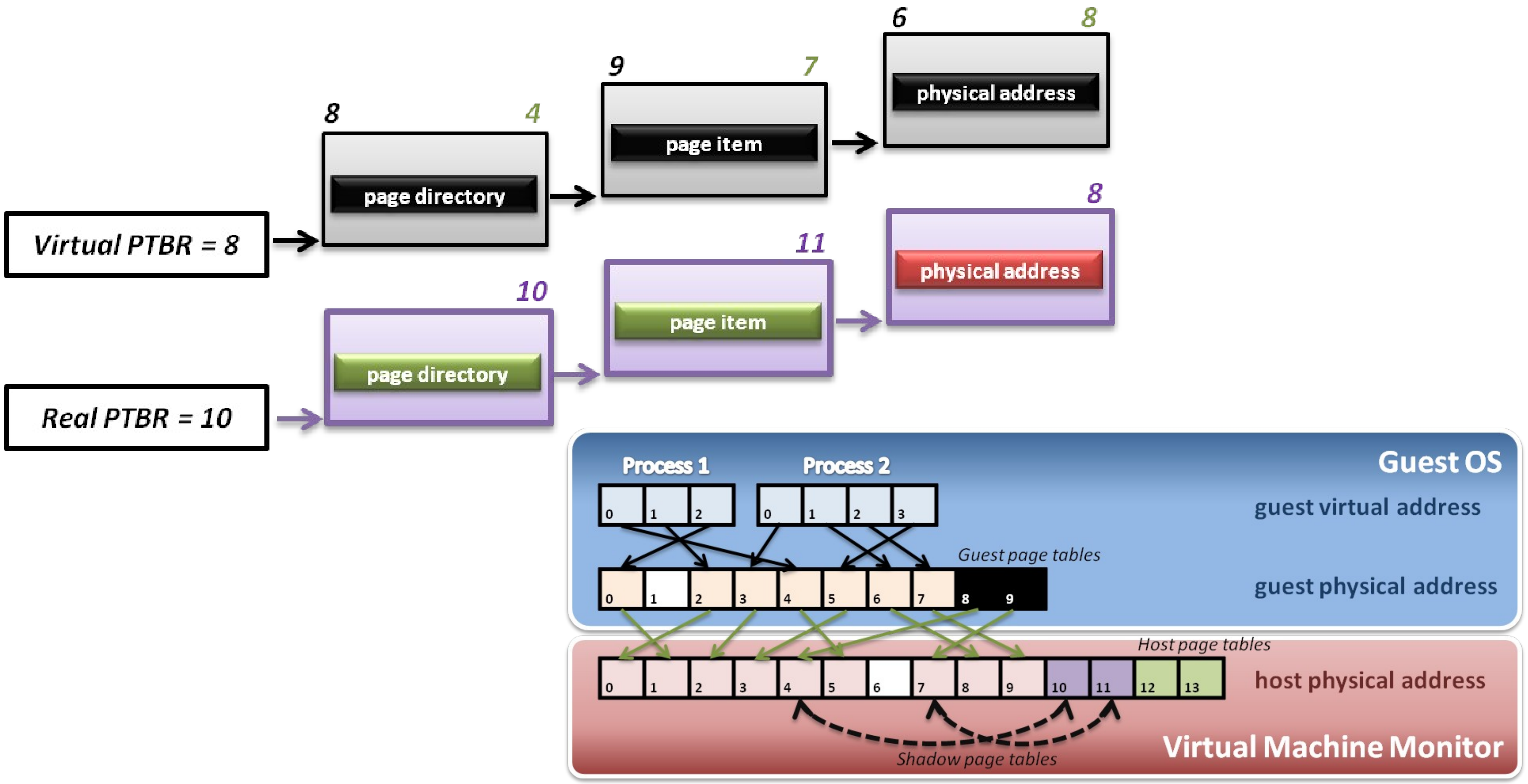
# Shadow Page Table

- Construct shadow page table
  - Guest OS will maintain its own page table for each process.
  - VMM maps each guest physical page to host physical page.
  - Create shadow page tables for each guest page table.
  - VMM should protect host frame which contains guest page table. (that means to write prote ct the guest page tables in host memory)



## Shadow Page Table

- Implement with PTBR :
  - For example, process 2 in guest OS wants to access its memory whose page number is 1.



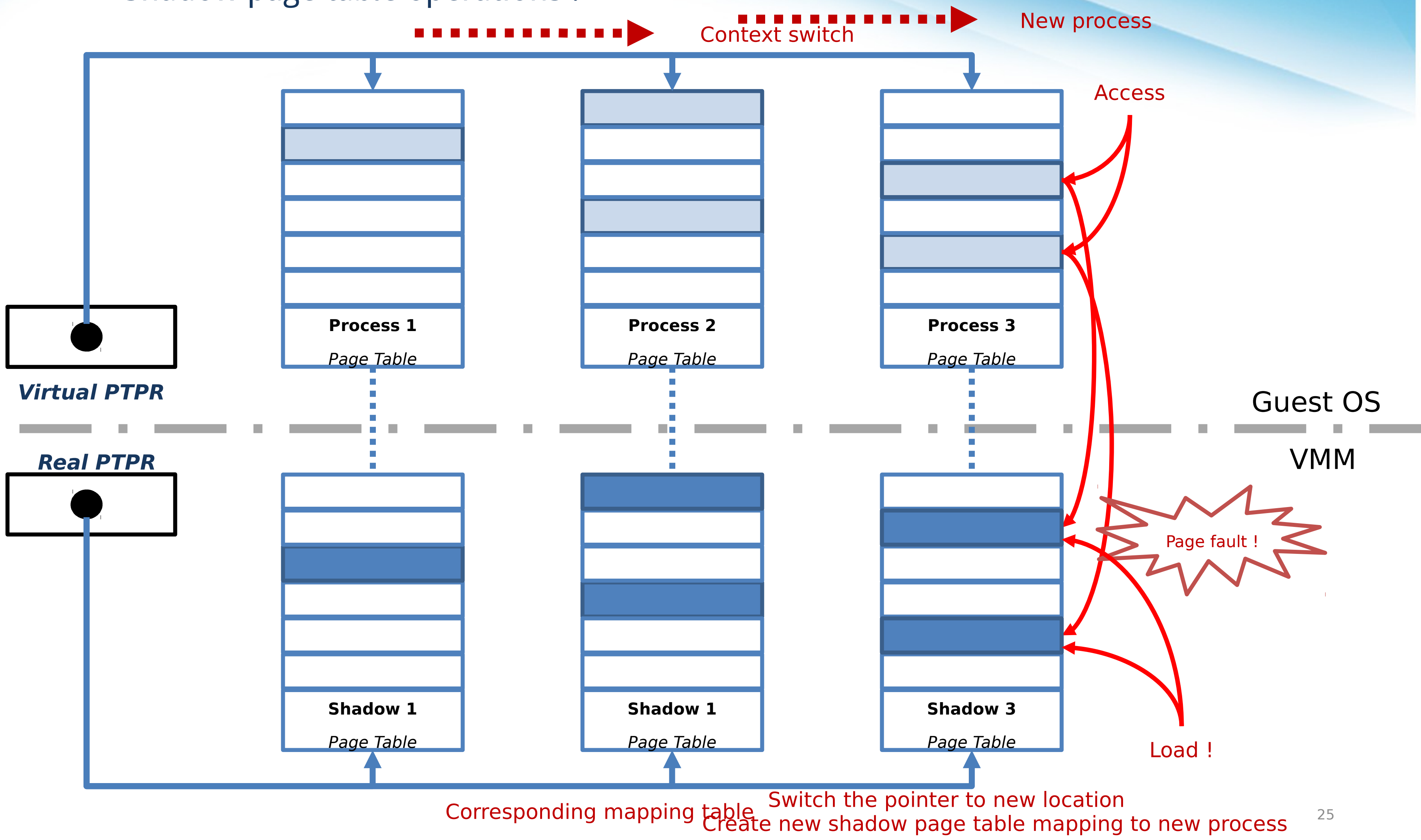
## SPT Maintenance

- If guest OS modify one of its page tables, then the corresponding entry of SPT must be updated.
  - We call it “shadow” because SPT is just like the shadow of page tables of guest OS.

- How to identify this kind of modification?
  - Guest OS could read/write a physical frame with the help of SPT.
  - Mark those physical frames used as guest page tables read-only, so that when a guest OS tri es to modify its guest page table, an exception would be triggered.
  - Then VMM checks the modification and updates the corresponding entry on SPT

## Shadow Page Table

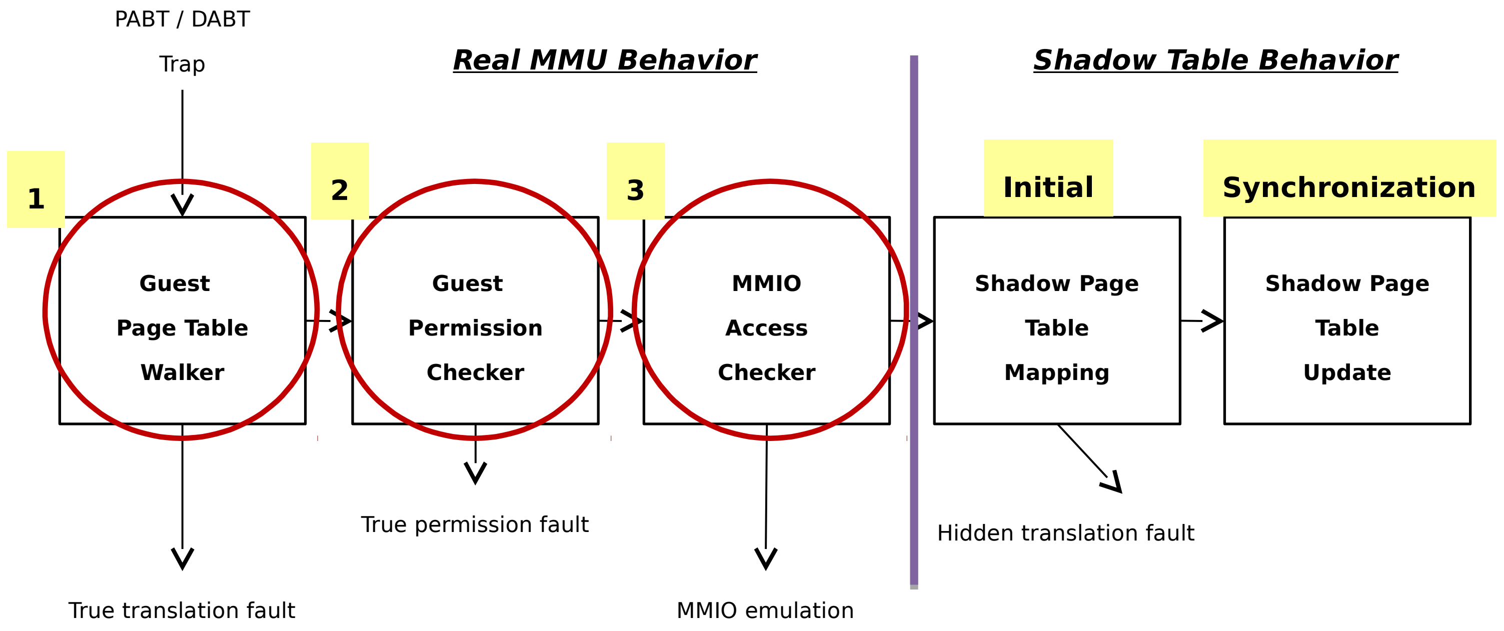
- Shadow page table operations :



## *Big Overhead*

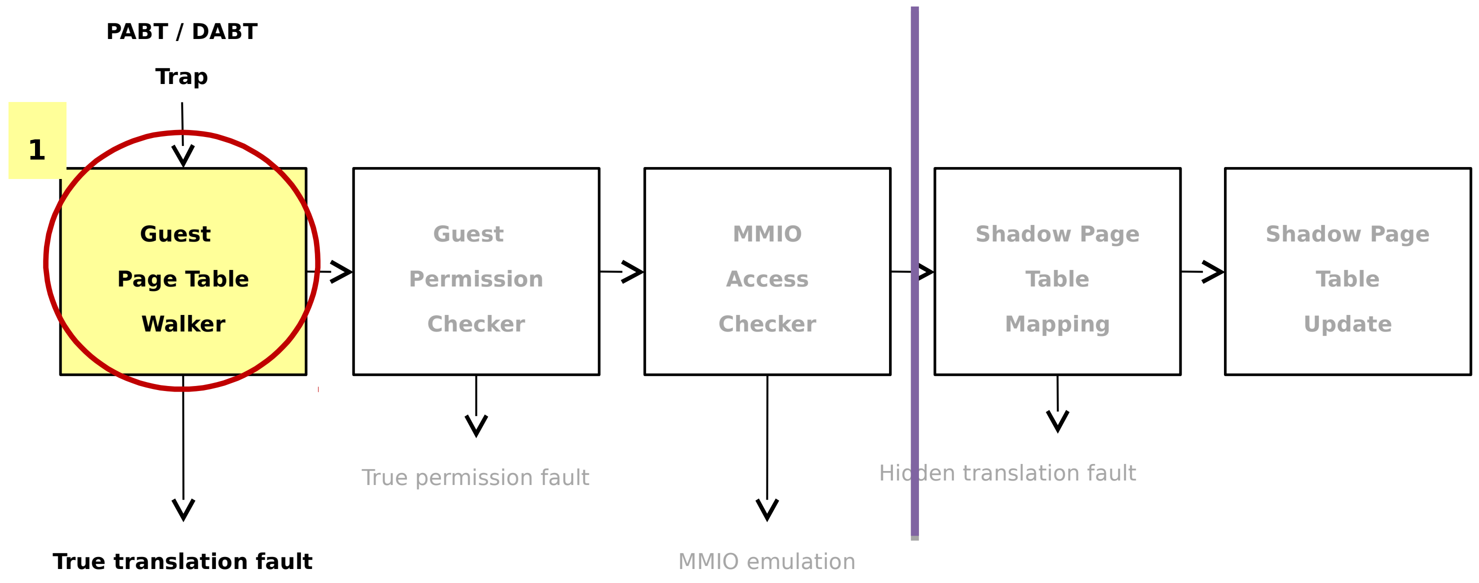
- A page fault caused by guest OS would launch the walking process that costs a lot o f overhead.
  - Several steps to get a new entry on shadow page table
    - Walk page tables on guest OS
    - Check the permission on guest
    - Offset shift: GPA to HVA
    - Walk page tables on VMM
    - Check the permission on VMM
    - New entry established!
    - Invalidate the TLB entry
  - Each new process on guest OS would consume two pages.  One is the page table on guest O S, and the other is the corresponding shadow page table.

# Page Walking Process on ARM



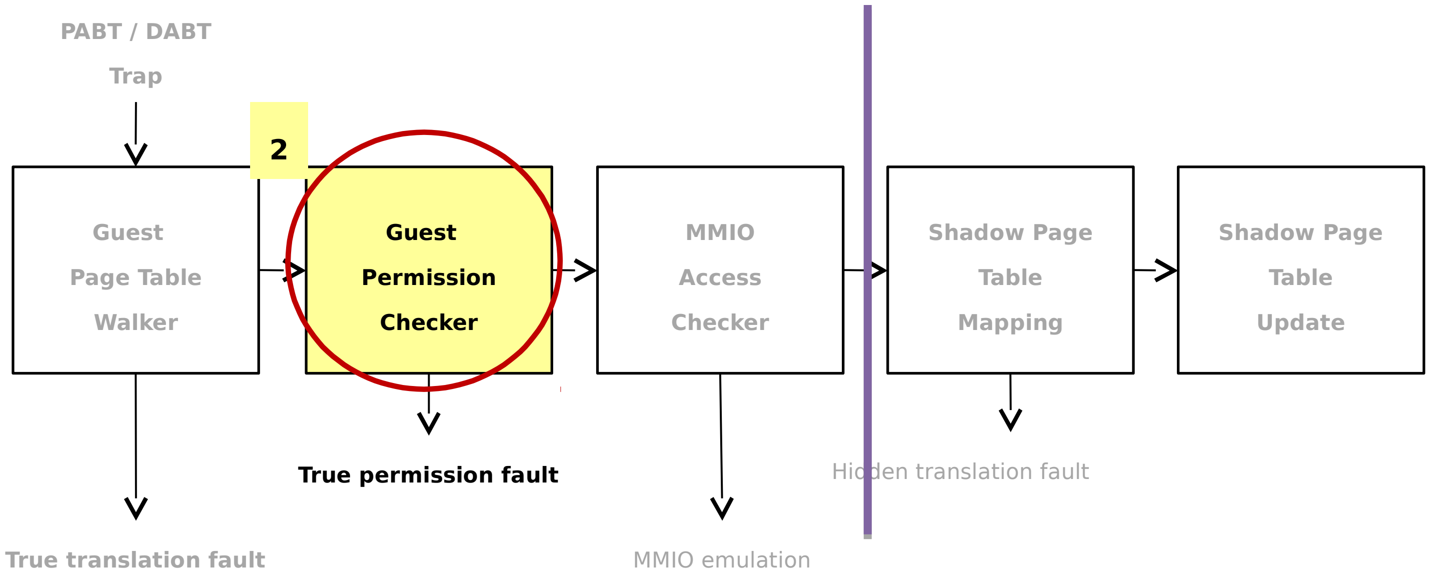
## *Step 1*

- While a page fault is occurred, Guest Page Table Walker will walk through guest page table to check if the fault is from guest



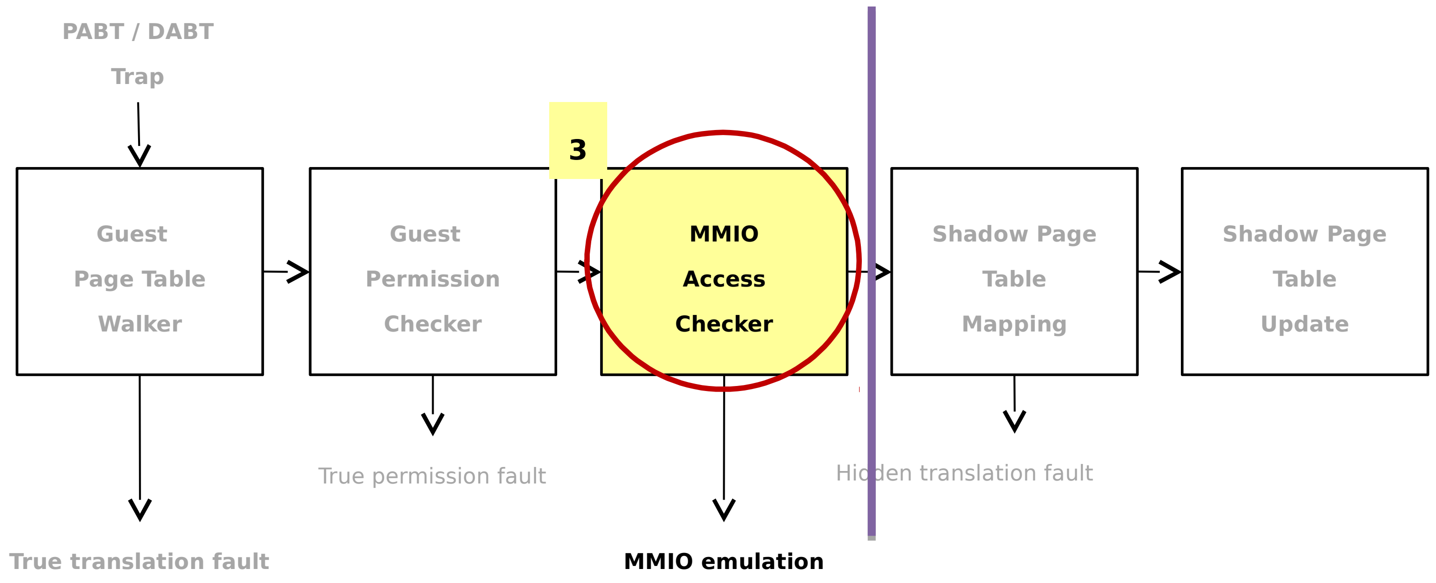
# *Step 2*

- Step 2 will check if guest access permission is not allowed



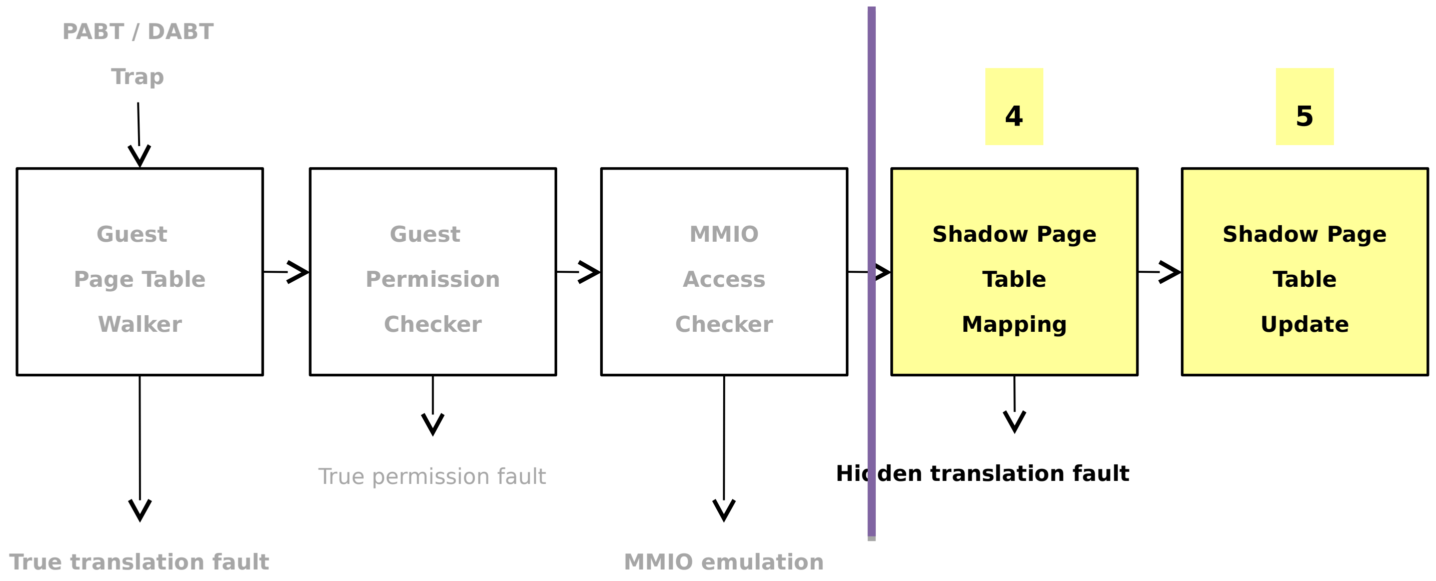
## *Step 3*

- Step 3 will check if the guest physical memory address used is located in the ra nge of MMIO address



## *Steps 4 & 5*

- Step 4 and step 5 are used to build up shadow page tables and maintain their c onsistency between guest and shadow ones



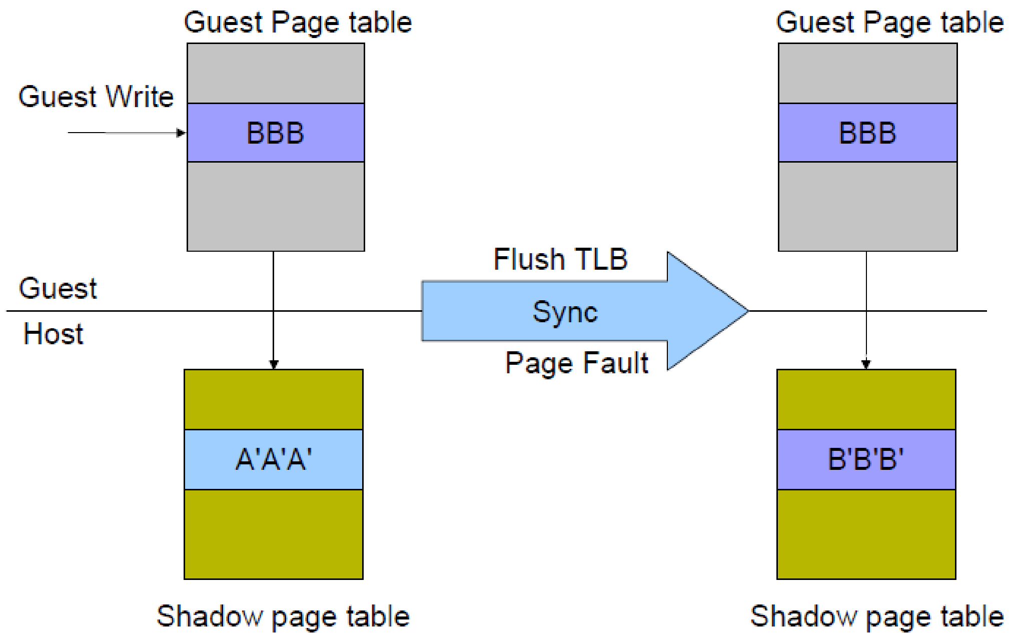
## Optimization For SPT

- Para-virtualization
  - Reduce VM exits
  - Guest OS would send a hyper call to VMM when guest OS sets the page table entries.
  - This method will eliminate from using write-protection for synchronization

- No trap for non-present PTEs
  - Reduce VM exits
  - If the page-fault is caused by PTE not present, it is not intercepted by the host
  - Only works on VMX

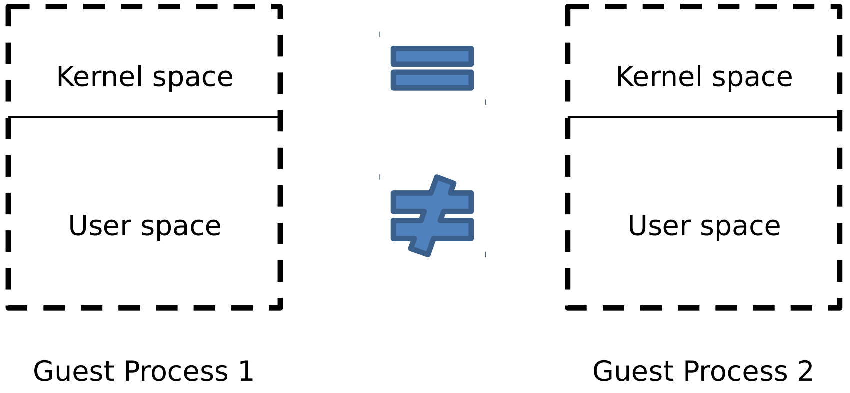
# Optimization For SPT

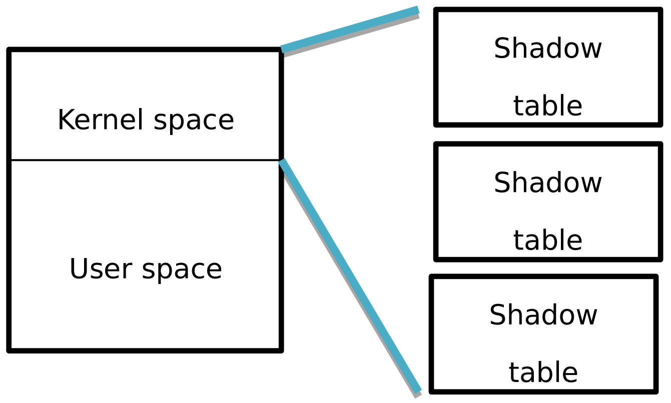
- Un-synchronize shadow page table pages
  - Reduce VM exits
  - Allow the guest page table to be writable if and only if the page is the last level page-structu re (level 1)
  - Based on TLB rules
    - We need to flush TLB to ensure the translation use the modified page structures, then we can Intercept the TLB flush operations and sync shadow pages
    - Sometimes, TLB need not be flushed, then it can be synced through page fault



# Optimization for SPT

- KSM: kernel shared memory





*The shadow tables of kernel space are shared by all guest processes*

## *Other Issues*

- Page fault and page protection issue
  - When a physical page fault occurs, VMM needs to decide whether this exception should be i njected to guest OS or not
    - If the page entry in a guest page table is still valid, VMM prepares for the corresponding page and does not inject any exception to guest OS.
    - If the page entry in a guest page table is invalid, then VMM directly injects the virtual page fault to guest OS.

Concepts

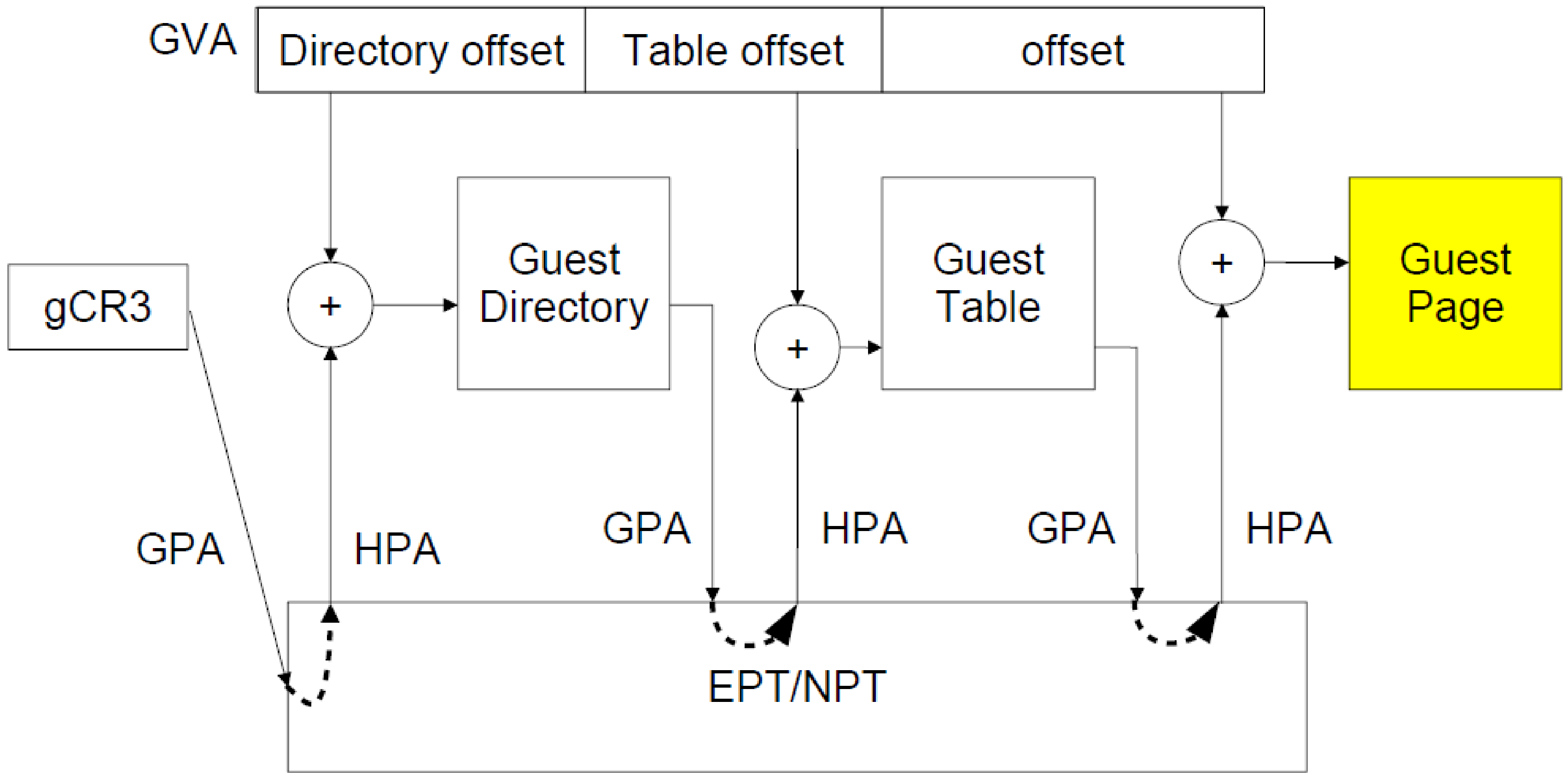
Shadow page table

Hardware assistance

Comparison

## *Memory Virtualization*

## Hardware Assistance: Overview



## *Hardware Solution*

- Difficulties of shadow page table technique :
  - Shadow page table implementation is extremely complex.
  - Page fault mechanism and synchronization issues are critical.
  - Host memory space overhead is considerable.

- But why we need this technique to virtualize MMU ?
  - MMU do not first implemented for virtualization.
  - MMU is knowing nothing about two level page address translation.

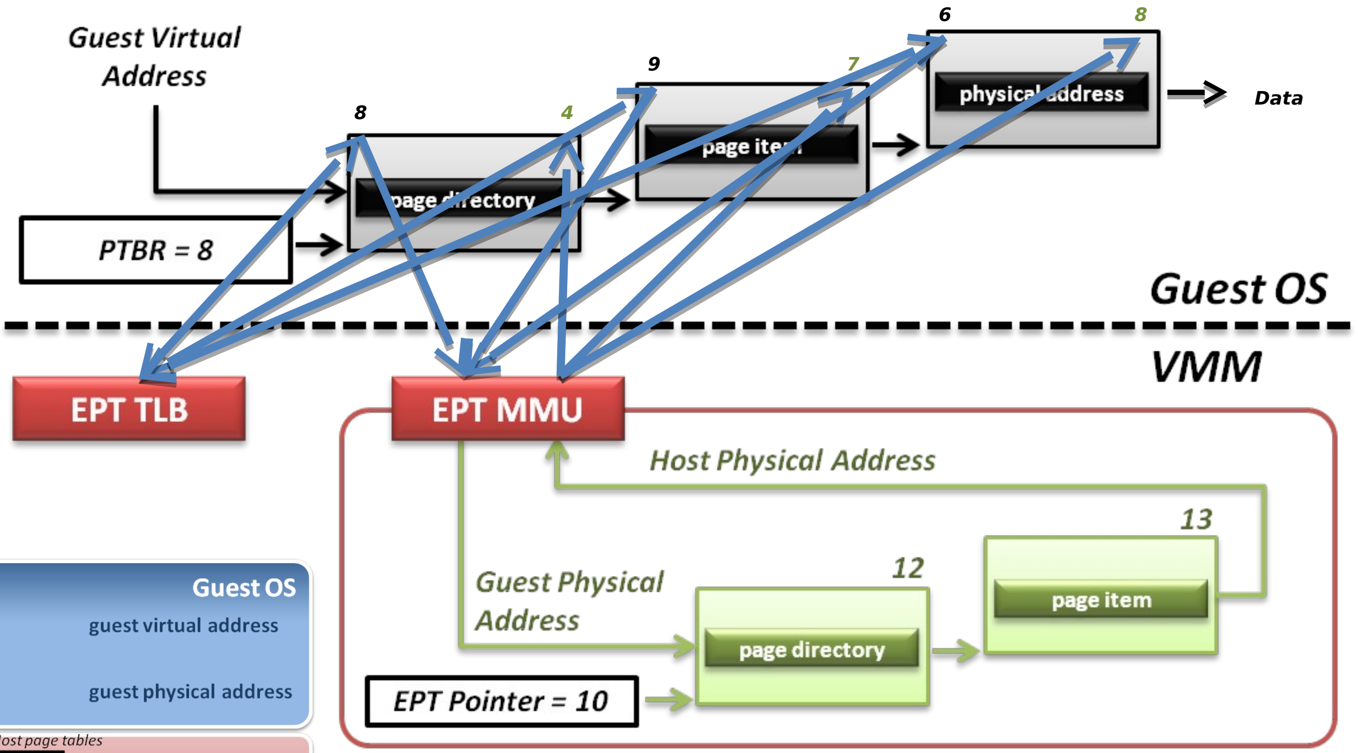
- Now, let us consider hardware solution.

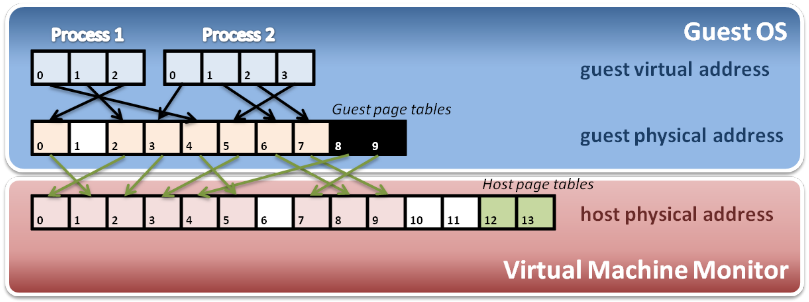
## Extended Page Table

- Concept of Extended Page Table (EPT) :
  - Instead of walking along with only one page table hierarchy, EPT technique implement one more page table hierarchy.
    - One page table is maintained by guest OS, which is used to generate guest physical address.
    - The other page table is maintained by VMM, which is used to map guest physical address to host physic al address.
  - For each memory access operation, EPT MMU directly gets the guest physical address from guest page table, and then gets the host physical address by the VMM mapping table autom atically.

# Extended Page Table

- Memory operation :





Concepts

Shadow page table

Hardware assistance

Comparison

## *Memory Virtualization*

## *Question*

- Computer architecture with virtualization extension is a trend.
- Hardware-assisted techniques replace many software methods of virtualization.
- However, is hardware-assisted implementation a definite winner?

## Comparisons

### Hardware-assisted

- Walk any requested address
  - Appropriate to programs that have a large am ount of page table miss when executing
  - Less chance to exit VM (less context switch)
- Two-layer EPT
  - Means each access needs to walk two tables
- Easier to develop
  - Many particular registers
  - Hardware helps guest OS to notify the VMM

### Software solution

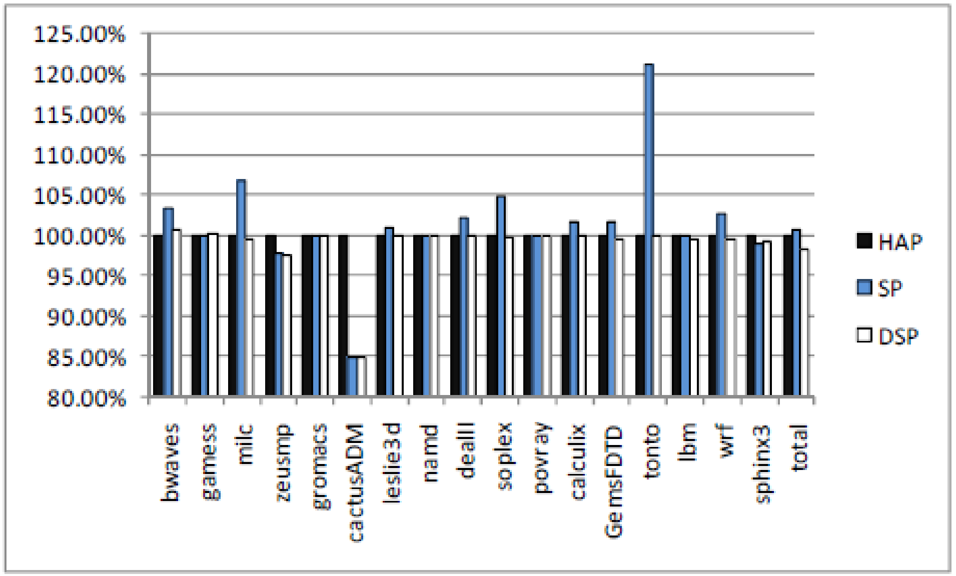
- Only walk when SPT entry miss
  - Appropriate to programs that would access on ly some addresses frequently
  - Every access might be intercepted by VMM (m any traps)
- One reference
  - Fast and convenient when page hit
- Hard to develop
  - Two-layer structure
  - Complicated reverse map
  - Permission emulation

## *Combination?*

- Is hardware-assisted implementation a definite winner?
  - No
  - How about combining these two methods?
- Selective memory virtualization
  - The VMM can dynamically choose the memory management mechanism depending on the executing status.
  - Challenge
    - Hard to figure out the standard that we judge a program performs too many page table misses
    - An accurate algorithm to sample behaviors of a program is necessary
  - Group from Peking University completed a selective solution on VEE 2011
    - Gain just a little bit performance improvement
    - For now, it's not worth to do so.

# Selective Memory Virtualization

- Experiment data
  - The bar shows the normalized executio n time
  - The lower, the better.
  - HAP: hardware-assisted page table
  - SP: shadow page table
  - DSP: dynamic selective page table
  - Take hardware extension as 100%
- We can see that...
  - Generally HAP is faster than SPT, but n ot always.
  - The performance of DSP is best , but o nly tiny disparities. (about 2%)



## Memory Virtualization Summary

- Software implementation
  - Memory architecture
    - MMU (memory management unit)
    - TLB (translation look-aside buffer)
  - Shadow page table
    - MMU virtualization by virtual PTBR
    - Shadow page table construction
    - Page fault and page table protection
- Hardware assistance
  - Extended page table
    - Hardware walk guest and host page table simultaneously

## Reference

- Selective hardware/software memory virtualization
  - http://www.cs.mtu.edu/~zlwang/papers/vee11.pdf
- ARM® Architecture Reference Manual: ARMv7-A and ARMv7-R edition