National University of Computer & Emerging Sciences

Week-14-Lecture – 01

Memory Virtualization

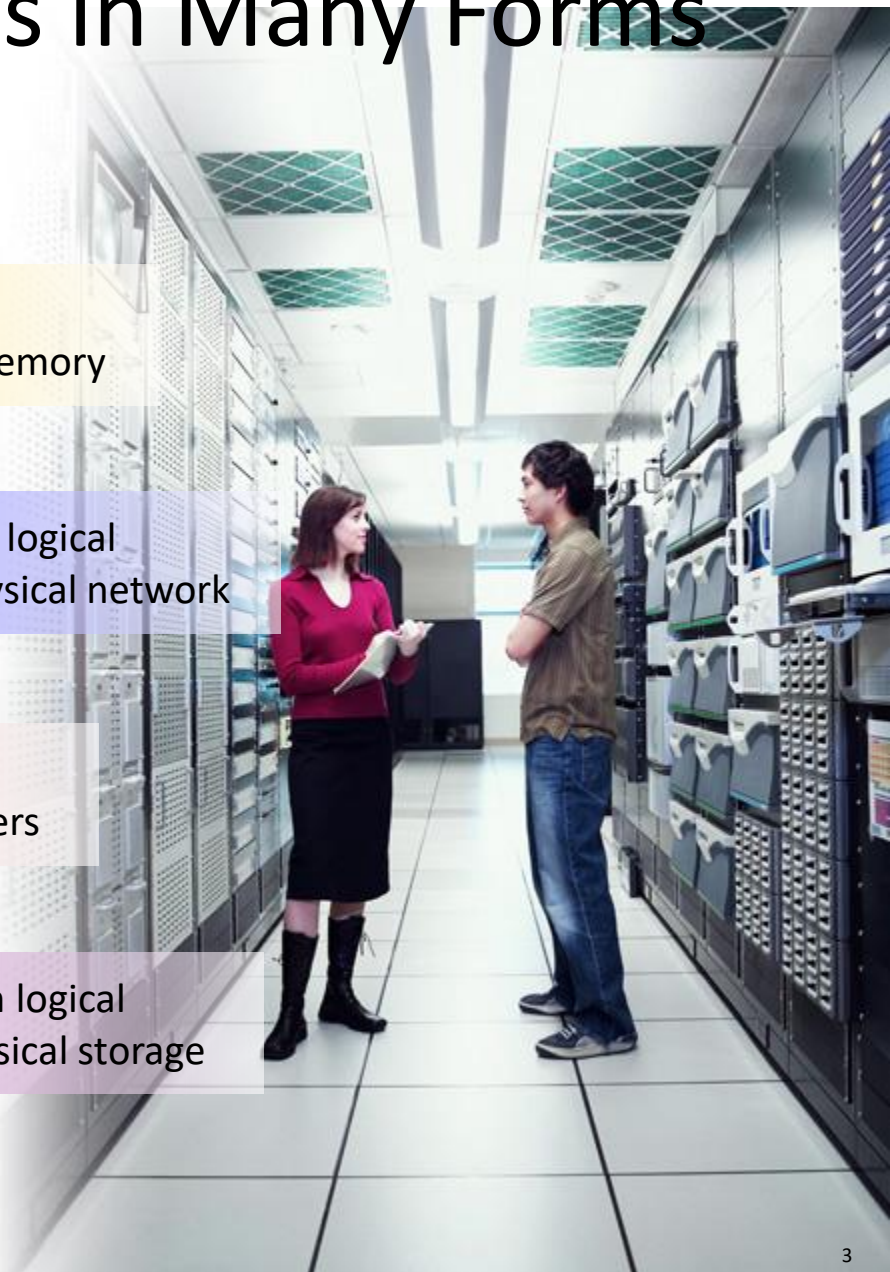
Dr. Hafiz Tayyab Javed

Disclaimer

- These slides are not fully prepared by me!!
- They inherit content from
  - *Cloud Computing course at National Tsing Hua University*

- Please do not distribute these lectures

Virtualization Comes in Many Forms



Virtual Memory

Each application sees its own logical **memory,** independent of physical memory

Virtual Networks

Each application sees its own logical **network,** independent of physical network

Virtual Servers

Each application sees its own logical **server,** independent of physical servers

Virtual Storage

Each application sees its own logical **storage,** independent of physical storage

Shadow page table

Hardware assistance

MEMORY VIRTUALIZATION

Virtual Memory

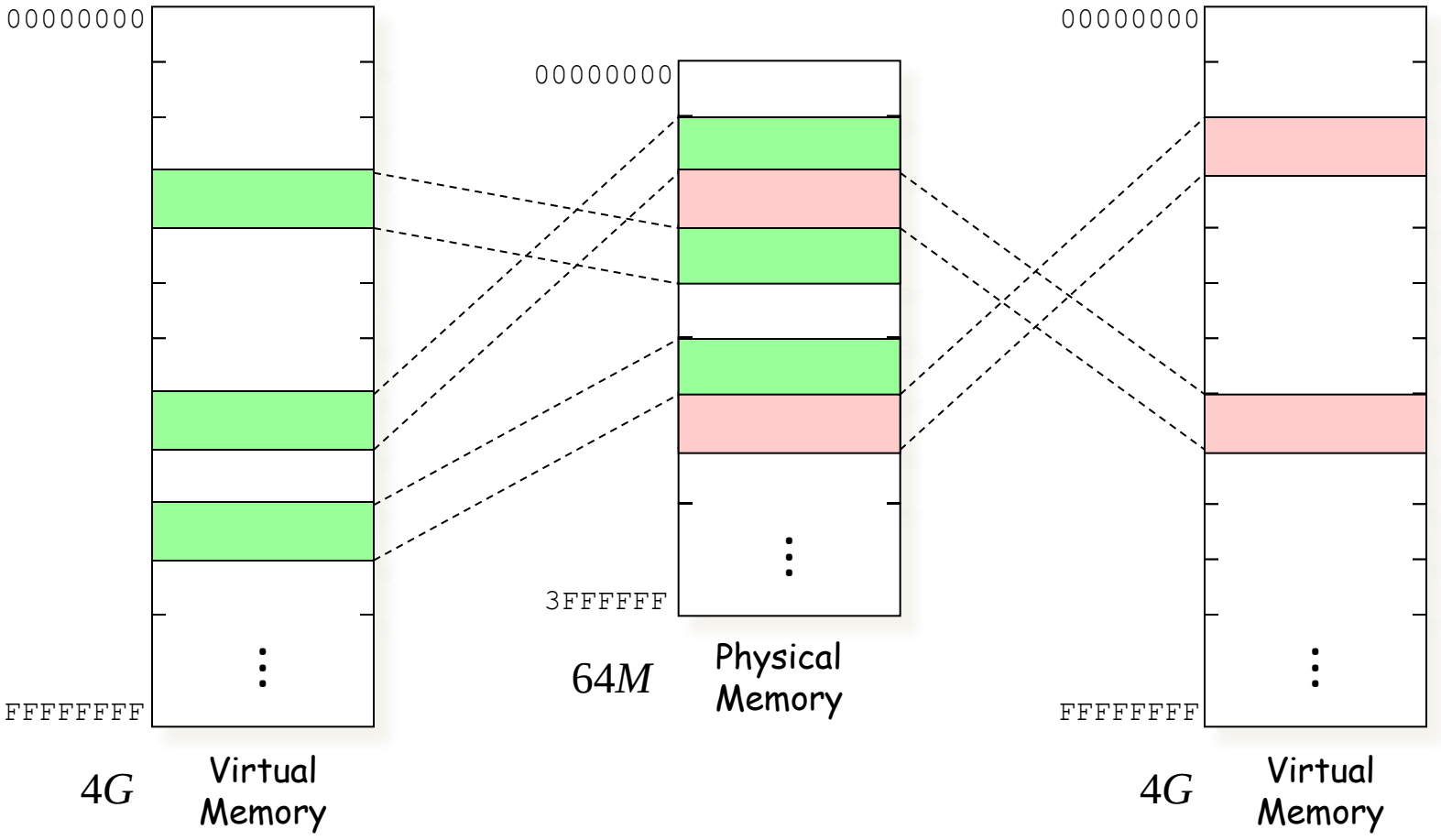
- What to do with programs too big to fit in main-memory?
  - Earlier systems relied on splitting the program in pieces, called overlays
  - It was the responsibility of programmer to split program

Virtual Memory

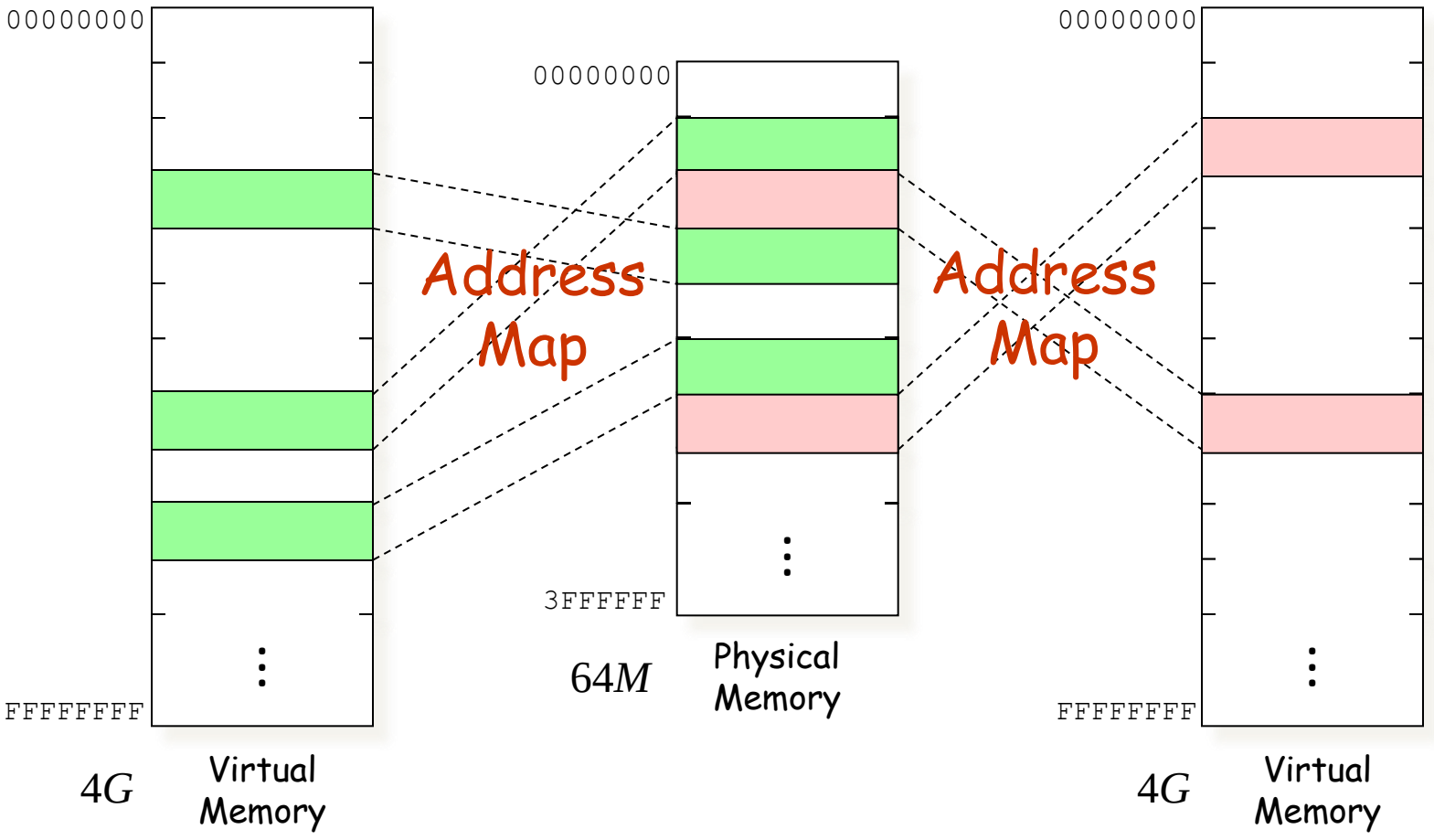
- Allow the process size (program, data …) to exceed the amount of Physical memory available (RAM)

- OS maintains bits and pieces of many programs in RAM at once
  - 4GB process can run on a machine with 1GB RAM by choosing which 1GB to keep in RAM

Principles of Virtual Memory



Principles of Virtual Memory



Paging

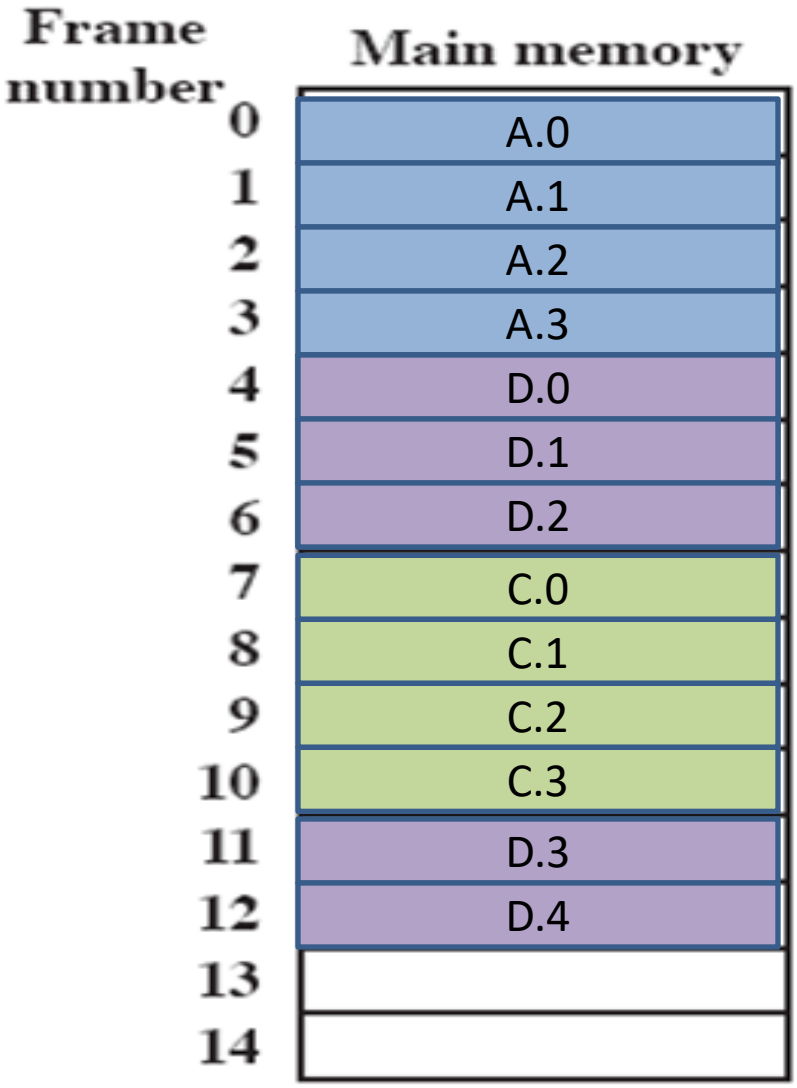
- Most virtual memory systems use Paging

- Idea: logical address space of a process can be made noncontiguous; process is allocated physical memory whenever the latter is available.

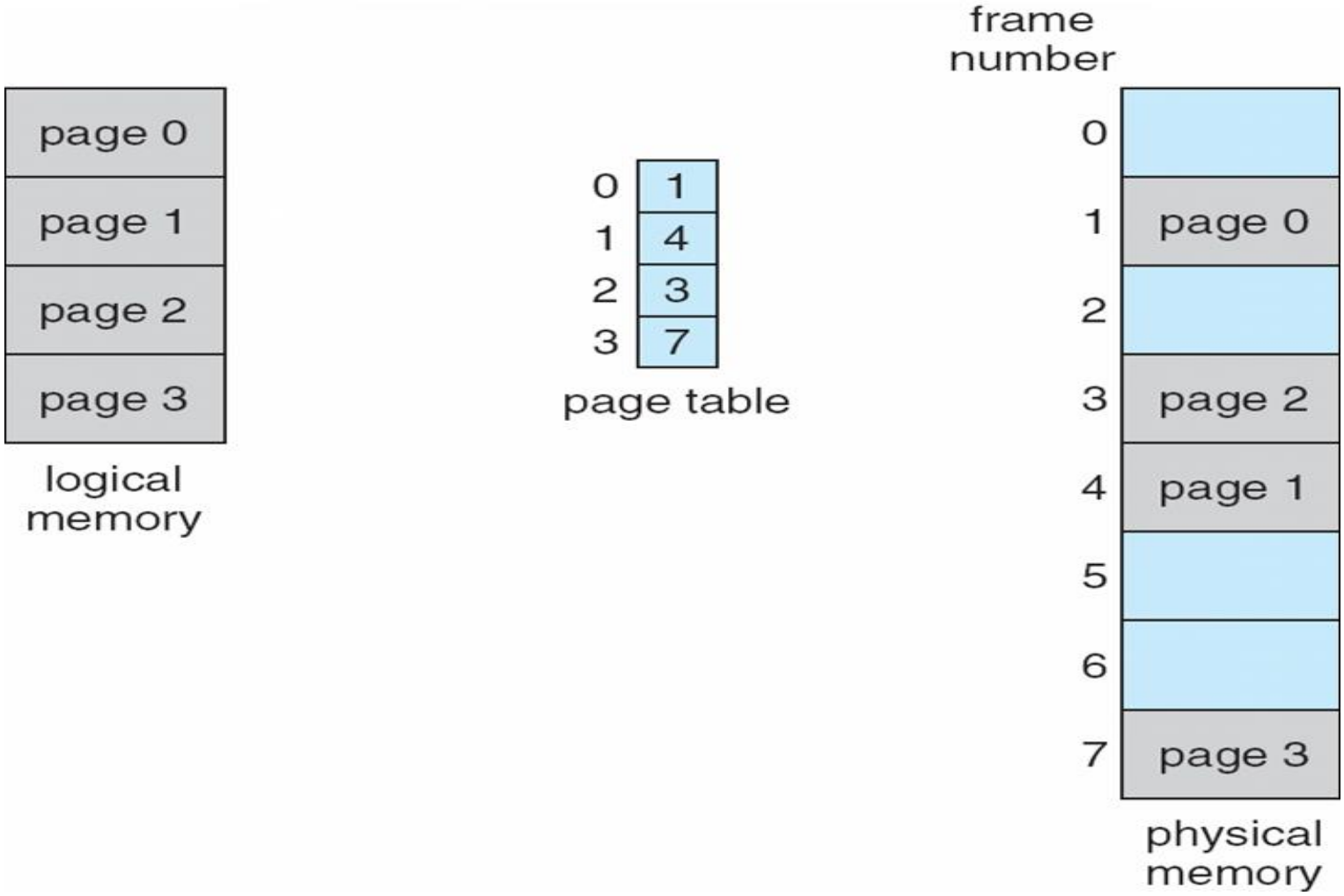
Paging

- Partition memory into small equal fixed-size chunks and divide each process into the same size chunks
  - The chunks of a process are called ***pages***
  - The chunks of memory are called ***frames***

Processes and Frames



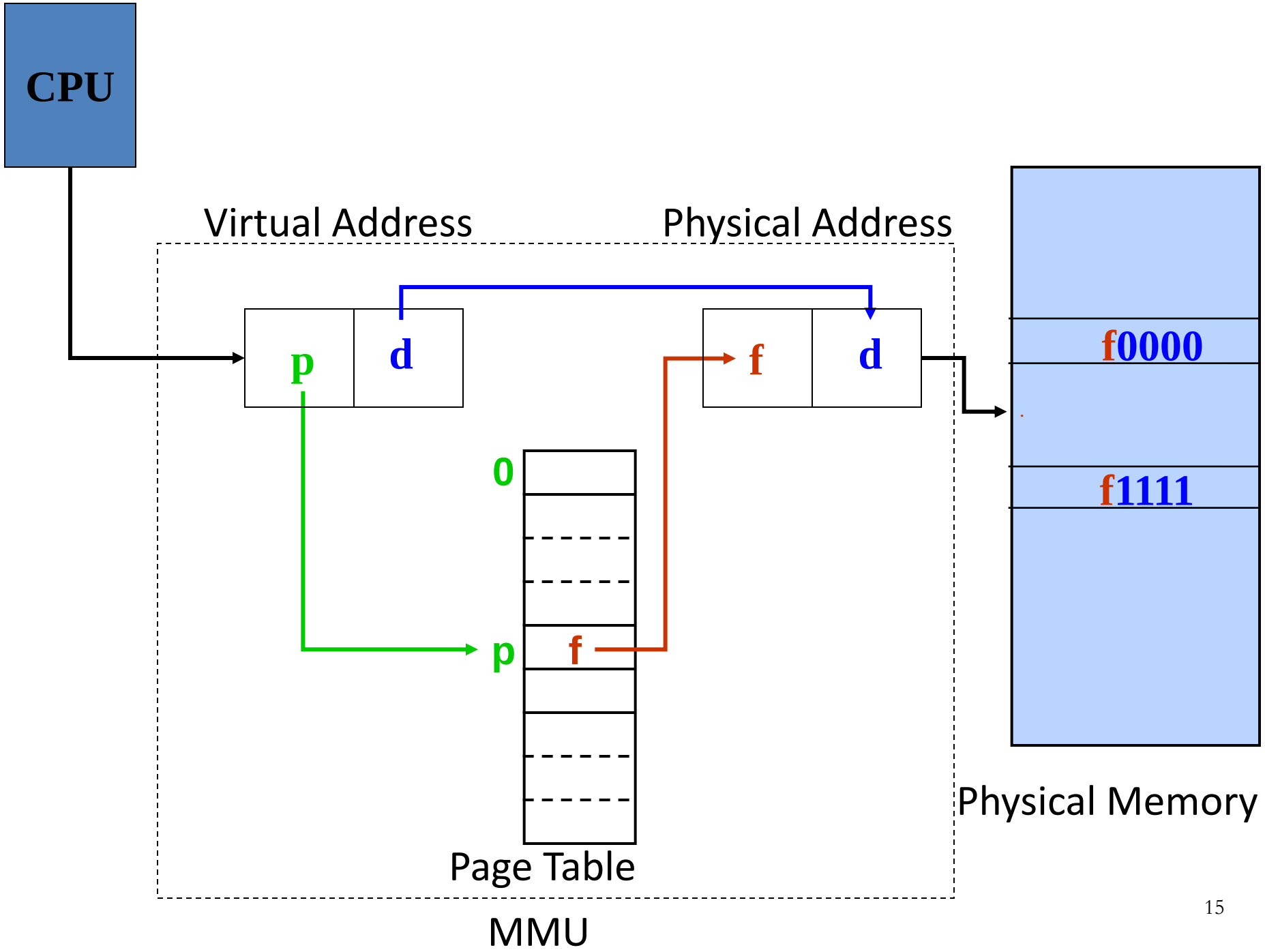
Paging Example



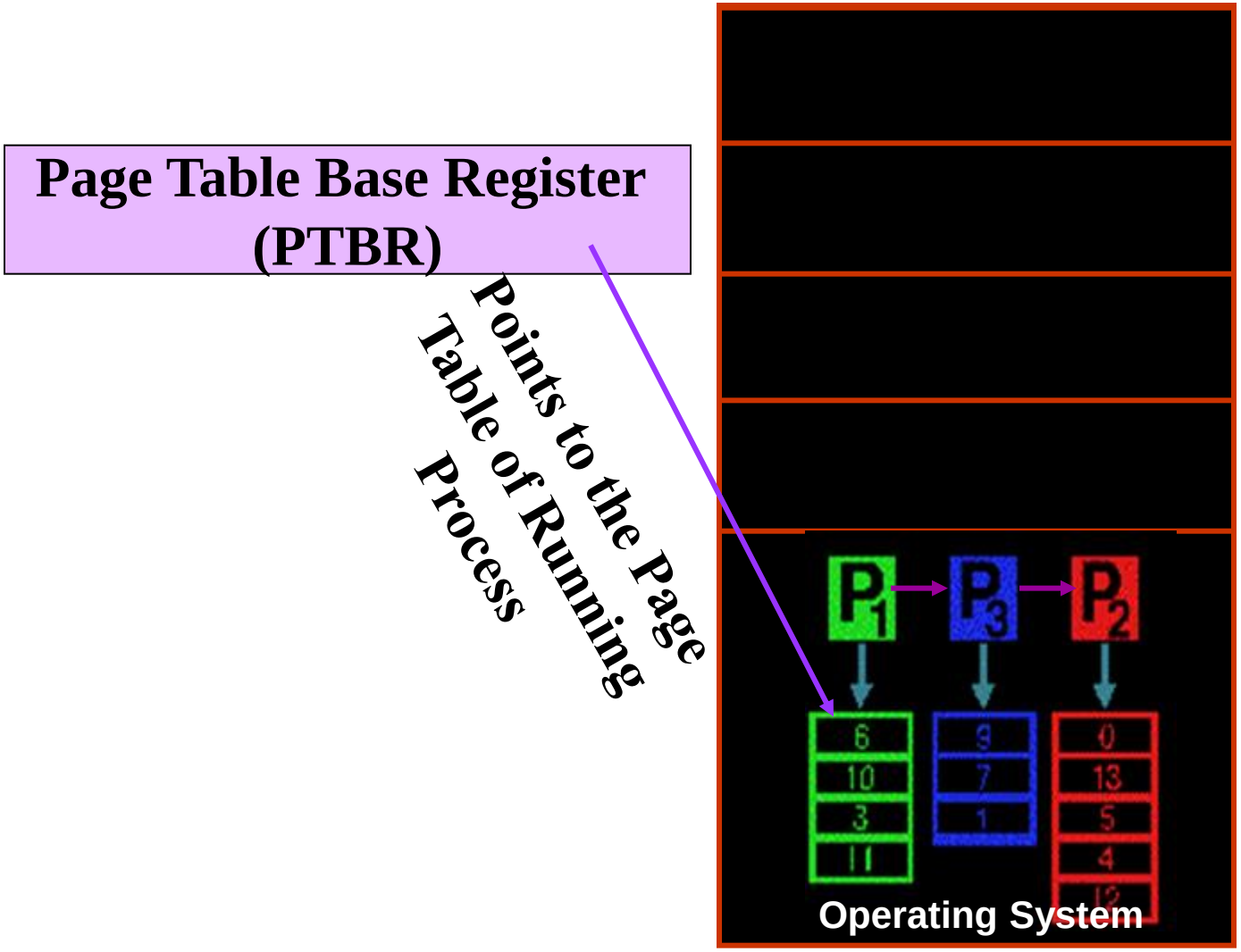
Paging

- If the Program generates the Virtual Address of a page that is unmapped?
- MMU generates an Interrupt called ***Page Fault***
- This invokes the Operating System
  - OS swaps out one of the Page from RAM
  - Swaps in the required page
  - Updates the Page Table

- Address generated by CPU is divided into:
  - **Page number (*p*)** – used as an index into a *page table* which contains base address of each page in physical memory
  - **Page offset (d)** – combined with base address to define the physical memory address that is sent to the memory unit


CPU
Virtual Address
Physical Address
p
d
f
d
f0000
f1111
0
p
f
Page Table
MMU
Physical Memory

Where is the Page Table Stored?



Page Table

- The only hardware required is a register
- Points to the starting address of the Page table
- Whenever, there is a context switch
- The register points to the starting address of the new process’s page table

Page Table

- Two major issues are:
  - The Page table entries can be extremely large
  - The mapping must be fast

Page Table Size

- If we have
  - 32-bit addresses
  - Page size = 4 KB = 2^{12} B
  - Possible Pages??
  - $2^{32}/2^{12}$
  - $= 2^{20}$
  - $= 10^{6}$
  - = 1 Million
- ➔ We need 1 Million Page Table entries in RAM all the Time!!!

Page Table Size

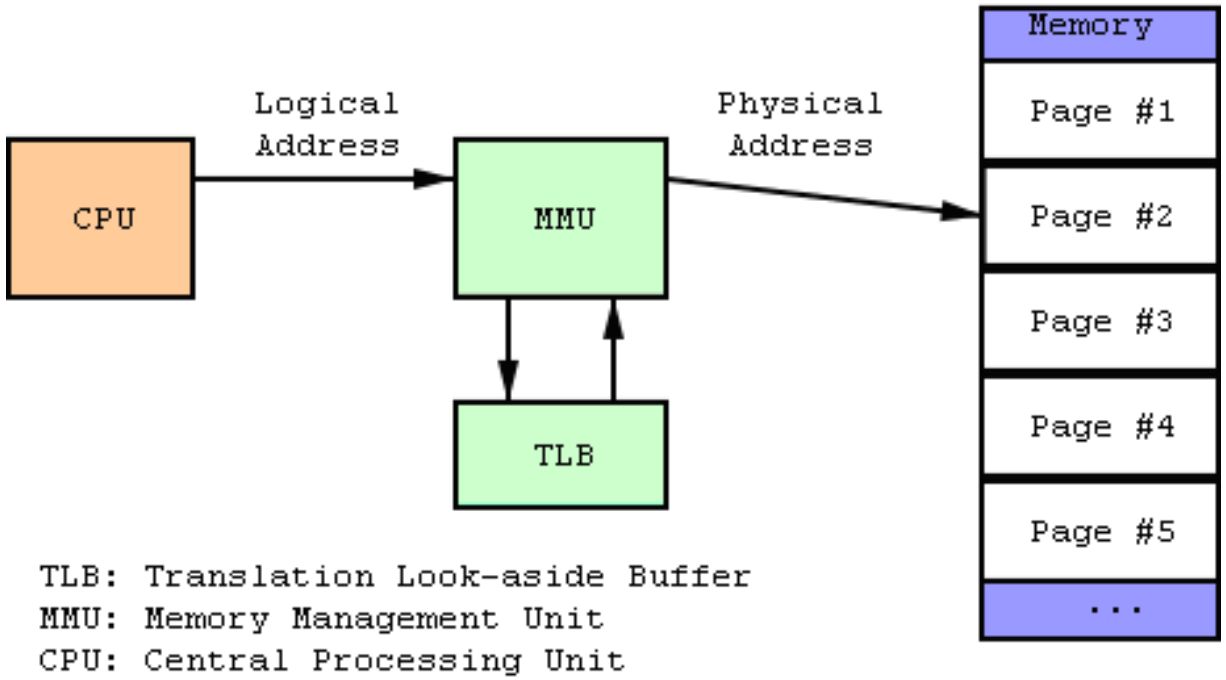
- Increasing the Page size will decrease the number of pages
- …but…
- More Internal Fragmentation

Another solution?

- Keep a number of fast hardware registers
- When a process starts
- OS loads the registers with the Process's entire Page table
- Process's Page table is available in RAM
- During execution no memory references are required for Page tables
- Really?
- No, it is expensive, if page size is large

Translation Lookaside Buffer (TLB)

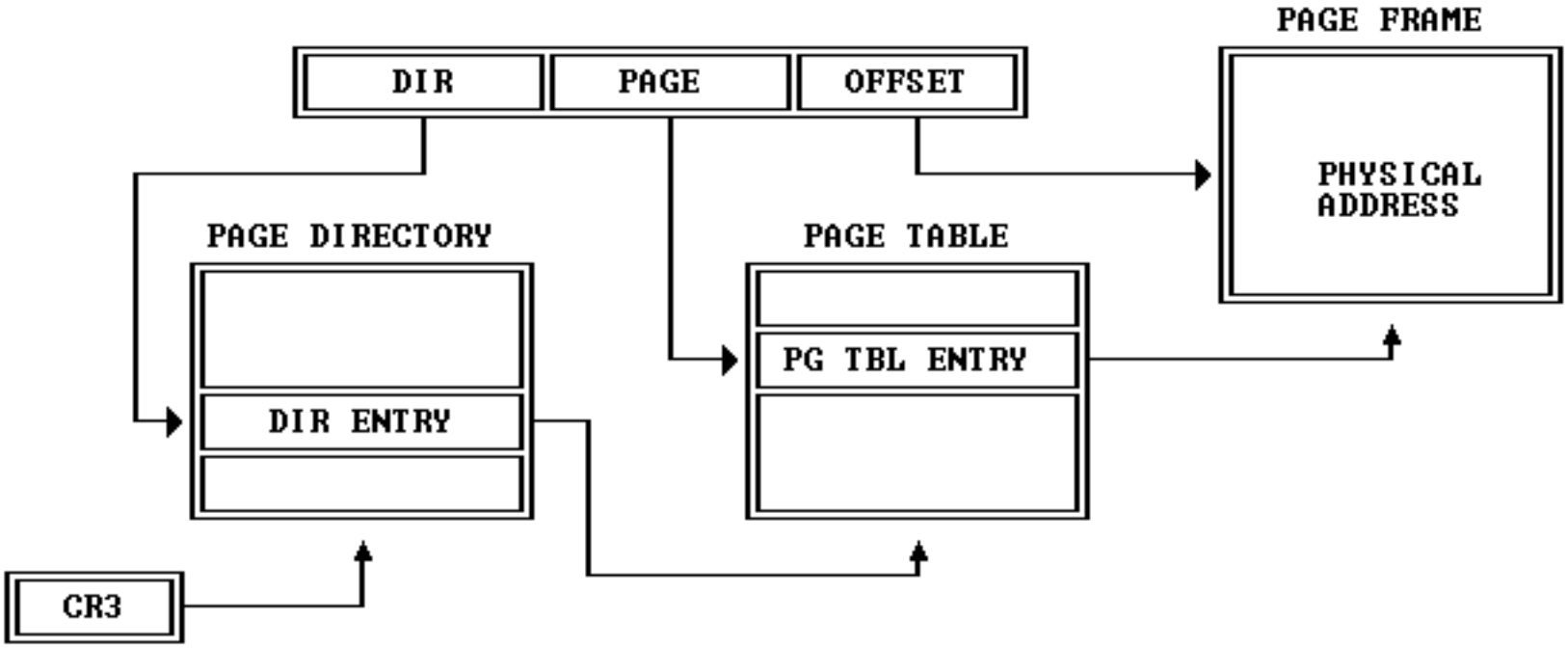
- What is TLB ?
  - A CPU cache that memory management hardware uses to improve virtual address translation speed.



What else can be done?

- Multi-level page tables
  - First level: page directory, contains pointers to actual page tables.
  - Second level: page tables. Each entry points to a page.

Multi-level page tables

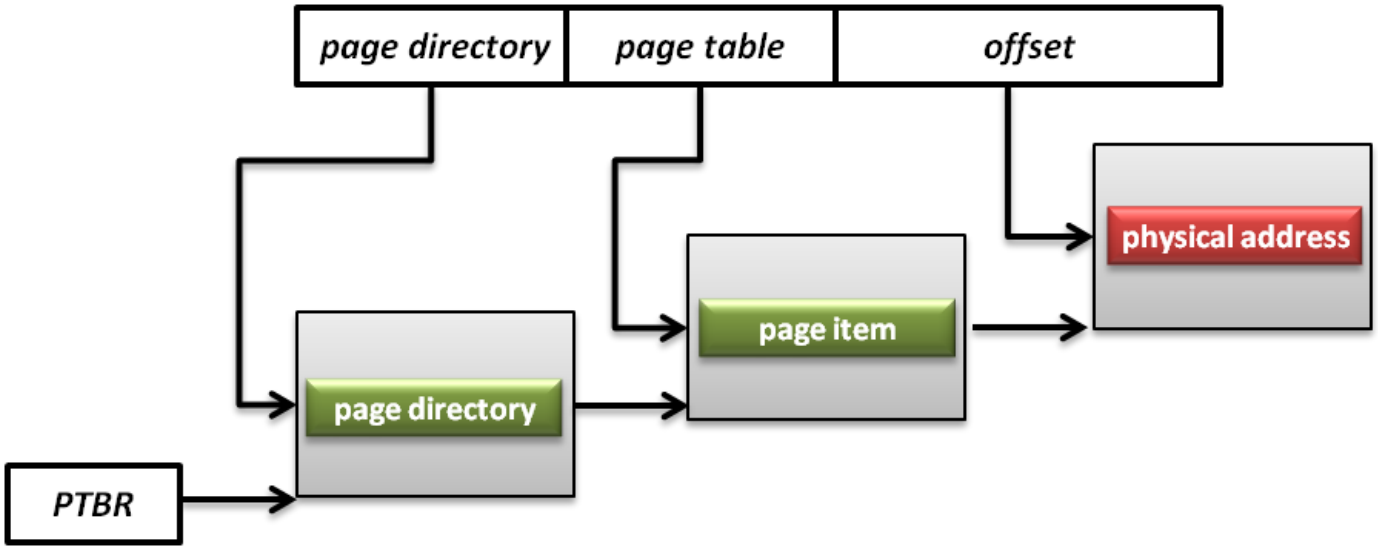


Memory Virtualization

- Memory management in OS
  - Traditionally, OS fully controls all physical memory space and provide a continuous addressing space to each process.
  - In server virtualization, VMM should make all virtual machines share the physical memory space without knowing the fact.
- 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.

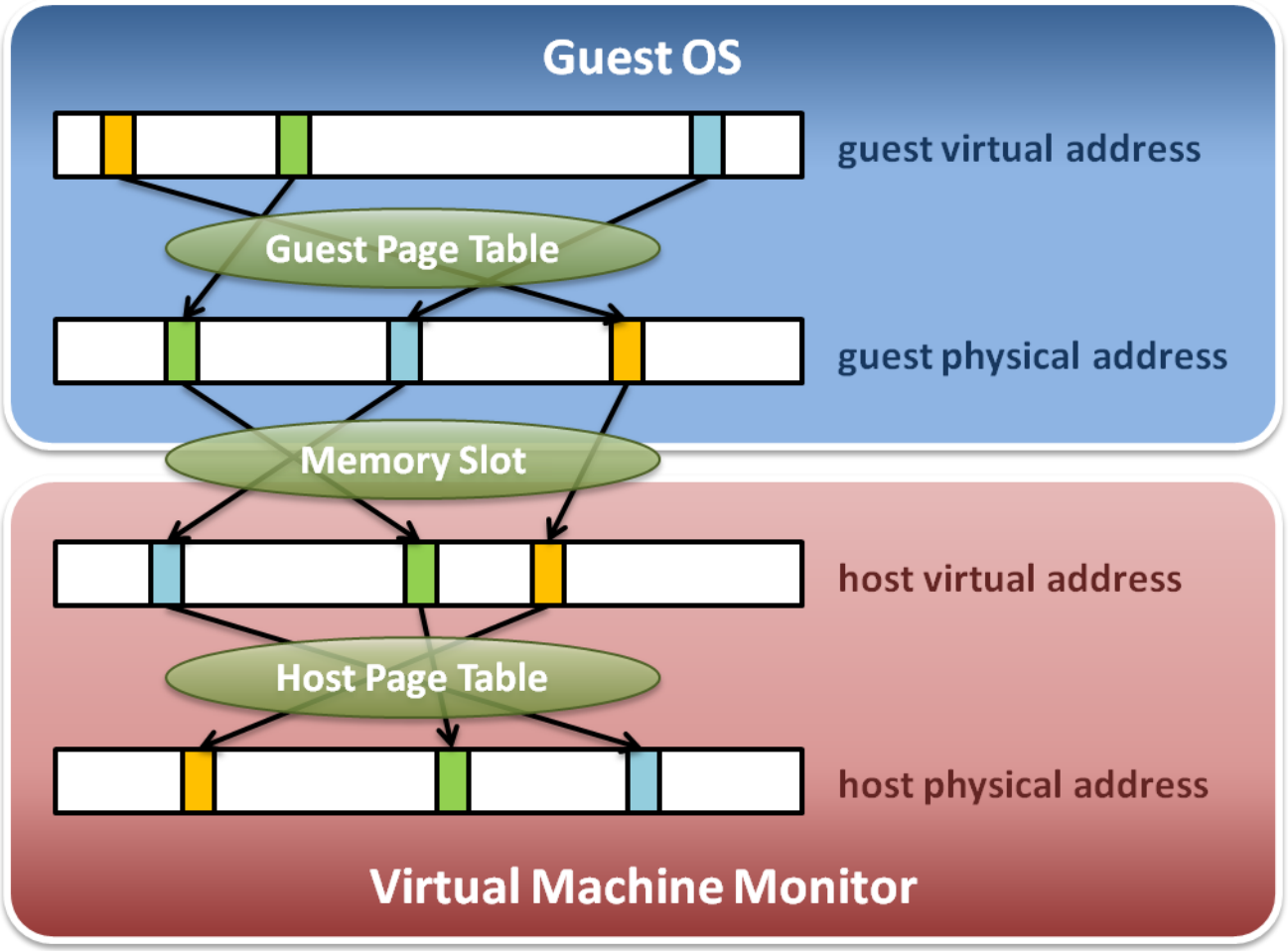
Memory Architecture

- Memory Management Unit (MMU)
  - What is MMU ?
    - A computer hardware component responsible for handling accesses to memory requested by the CPU.
    - Its functions include translation of virtual addresses to physical addresses, memory protection, cache control, bus arbitration and etc.
  - What is PTBR ?
    - Page Table Base Register (PTBR) is a register point to the base of page table for MMU.



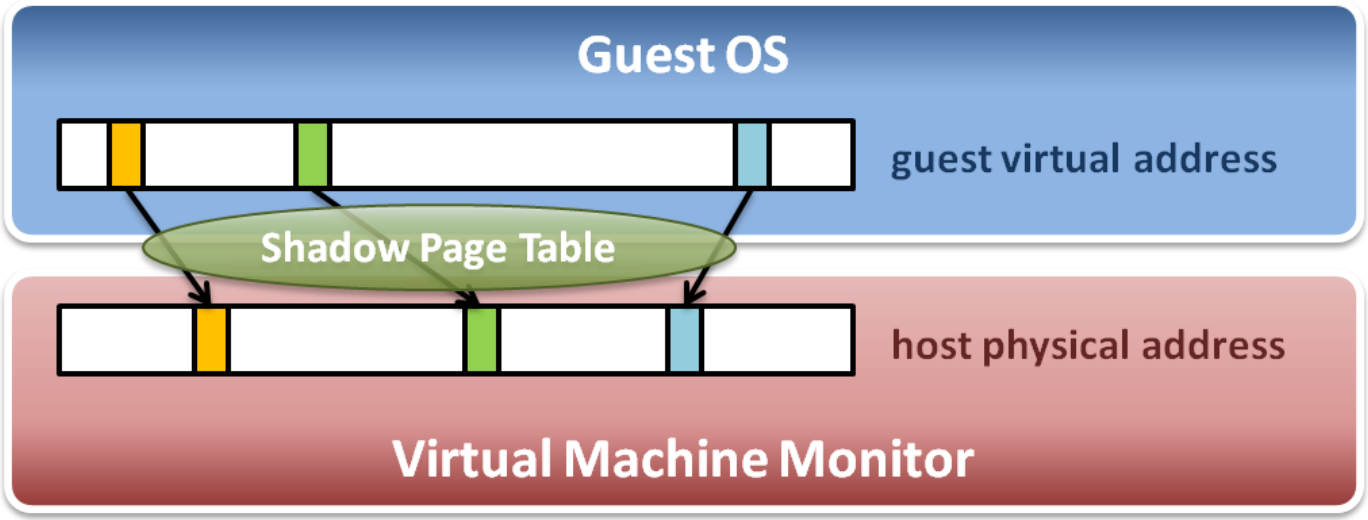
Memory Virtualization

- Memory virtualization assumed architecture



Memory Virtualization

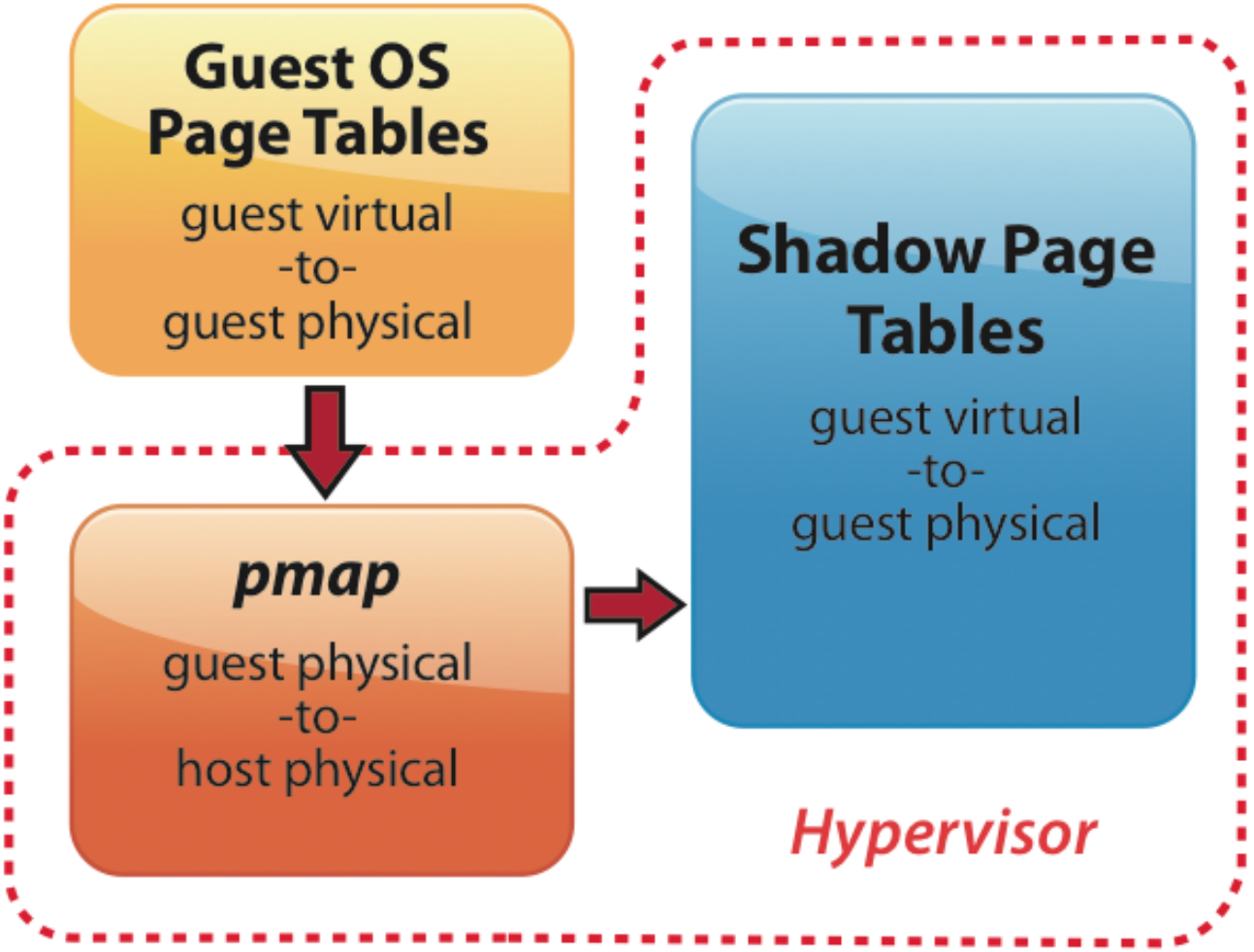
- The performance drop of memory access is usually unbearable. VMM needs further optimization.
- VMM maintains shadow page tables :
  - Direct virtual-to-physical address mapping
  - Use hardware TLB for address translation



Shadow Page Table

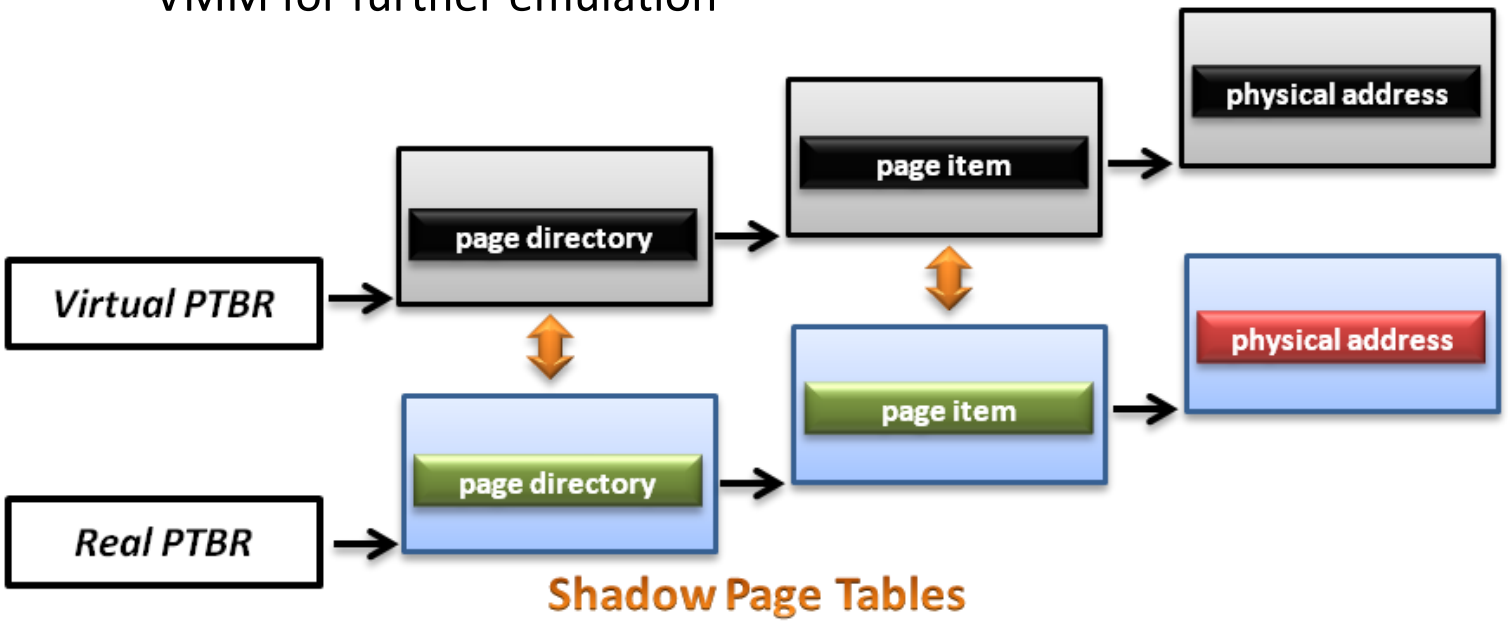
- Map guest virtual address to host physical address
  - Shadow page table
    - Guest OS will maintain its own virtual memory page table in the guest physical memory frames.
    - For each guest physical memory frame, VMM should map it to host physical memory frame.
    - Shadow page table maintains 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



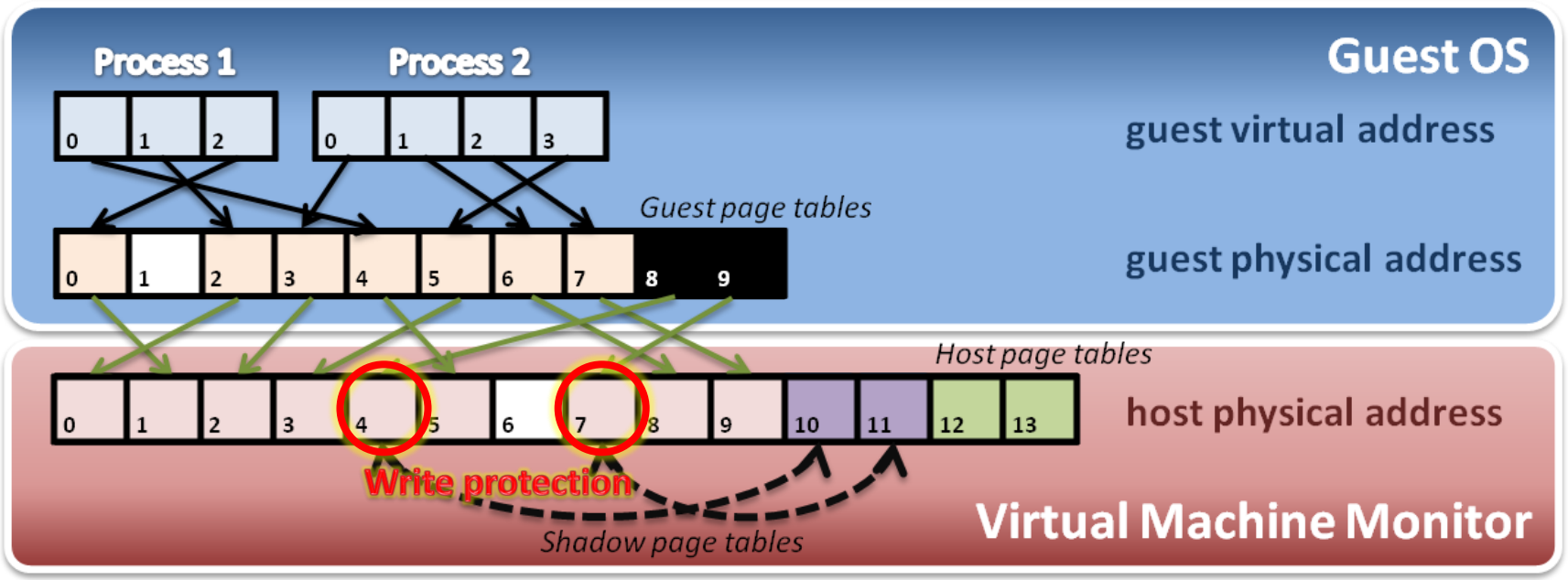
Shadow Page Table

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



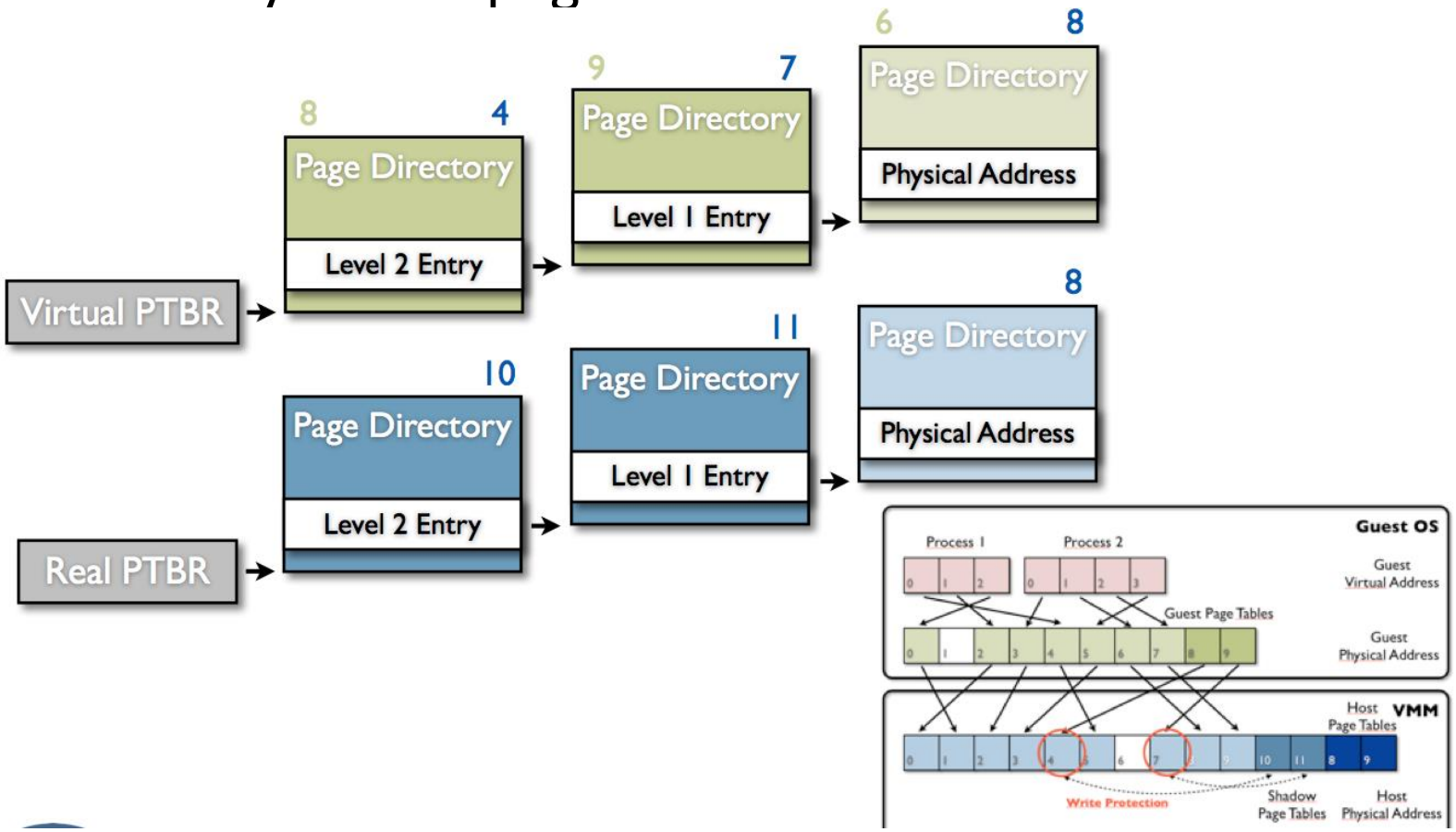
Shadow Page Table

- 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.



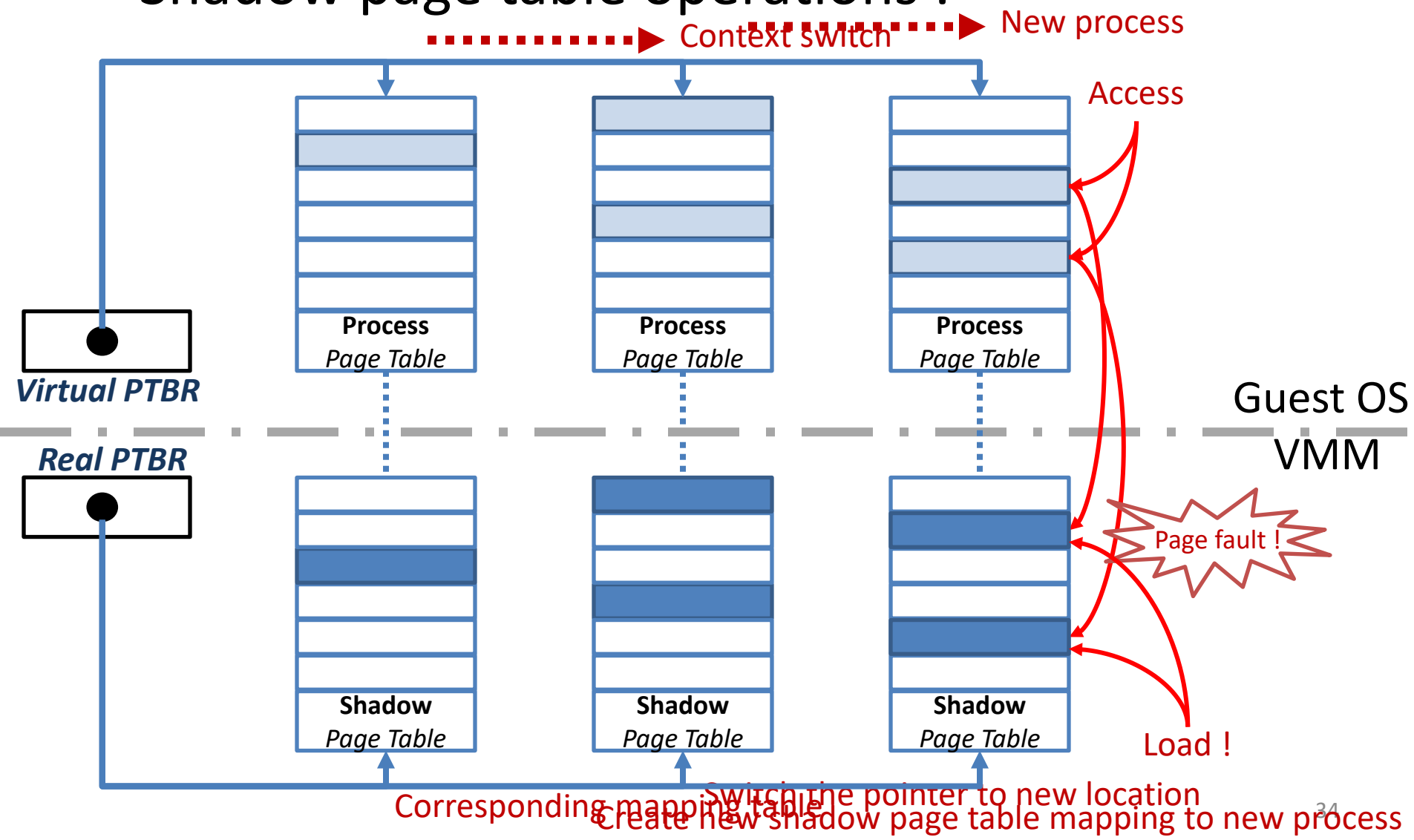
Shadow Page Table

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



Shadow Page Table

- Shadow page table operations :



Other Issues

- Page fault and page protection issue
  - When a physical page fault occurs, VMM need to decide whether this exception should be injected to guest OS or not.
    - If the page entry in the page table of guest OS is still valid, VMM should prepare the corresponding page and not inject any exception to guest OS.
    - If the page entry in the page table of guest Os is invalid either, then VMM should directly inject the virtual page fault to guest OS.
  - When guest OS want to modify its page tables, VMM need to intercept this operation.
    - When guest OS reload PTBR, CPU will trap to VMM due to the Ring Compression nature.
    - VMM will walk the page table of guest OS and modify the related shadow page table to make MMU get host physical address.

Shadow page table

Hardware assistance

MEMORY VIRTUALIZATION

Hardware Solution

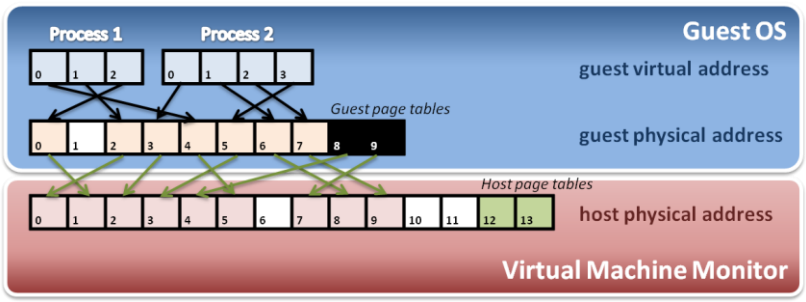
- Difficulties of shadow page table technique :
  - Shadow page table implementation is extremely complex.
  - Page fault mechanism and synchronization issues are critical.
  - Host memory space overhead is considerable.
- But why we need this technique to virtualize MMU ?
  - MMU do not first implemented for virtualization.
  - MMU knows nothing about two level page address translation.
- Now, let us consider hardware solution.

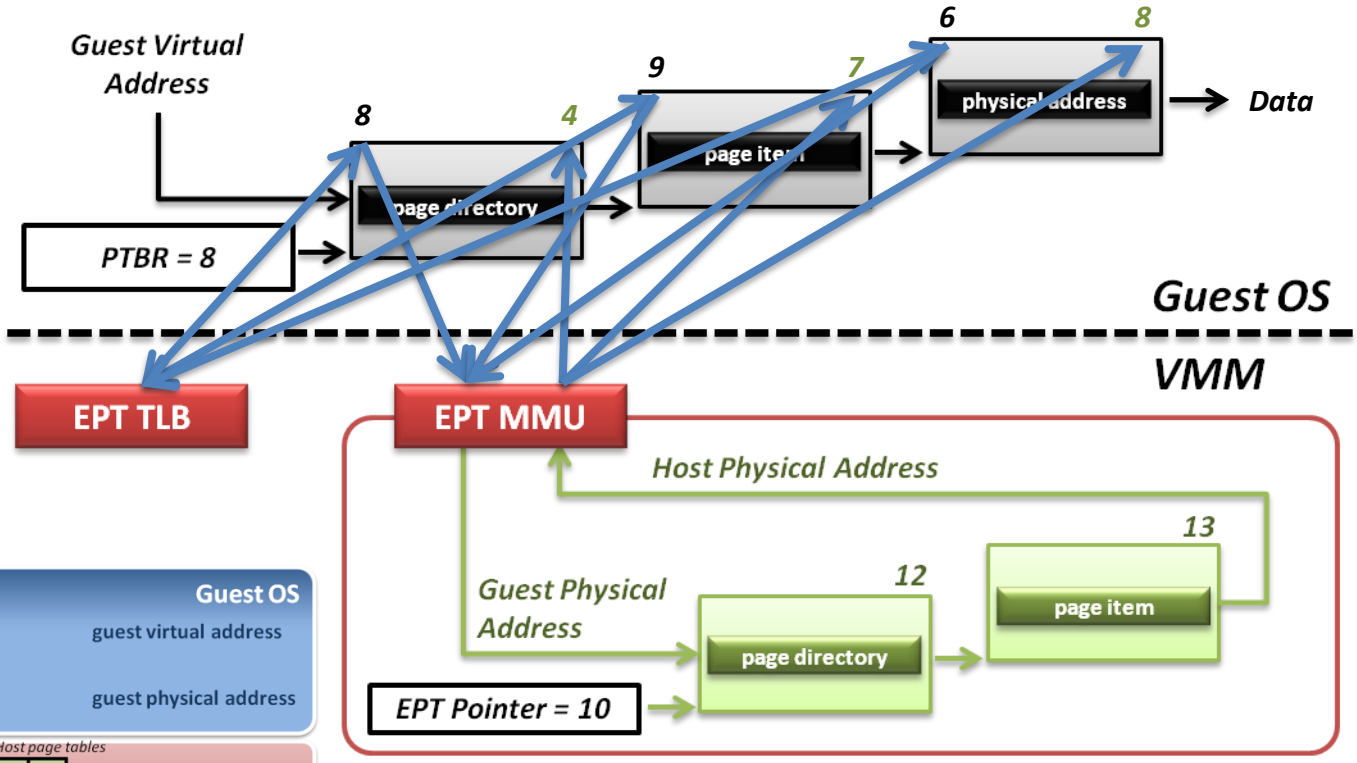
Extended Page Table

- 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 physical address.
  - For each memory access operation, EPT MMU will directly get guest physical address from guest page table, and then get host physical address by the VMM mapping table automatically.

Extended Page Table

- Memory operation :





Memory Virtualization Summary

- Software implementation
  - Memory architecture
    - MMU (memory management unit)
    - TLB (translation lookaside 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

Conclusion

- In Cloud computing fast translation of shadow page table can be done using hardware support EPT-MMU and EPT-TLB