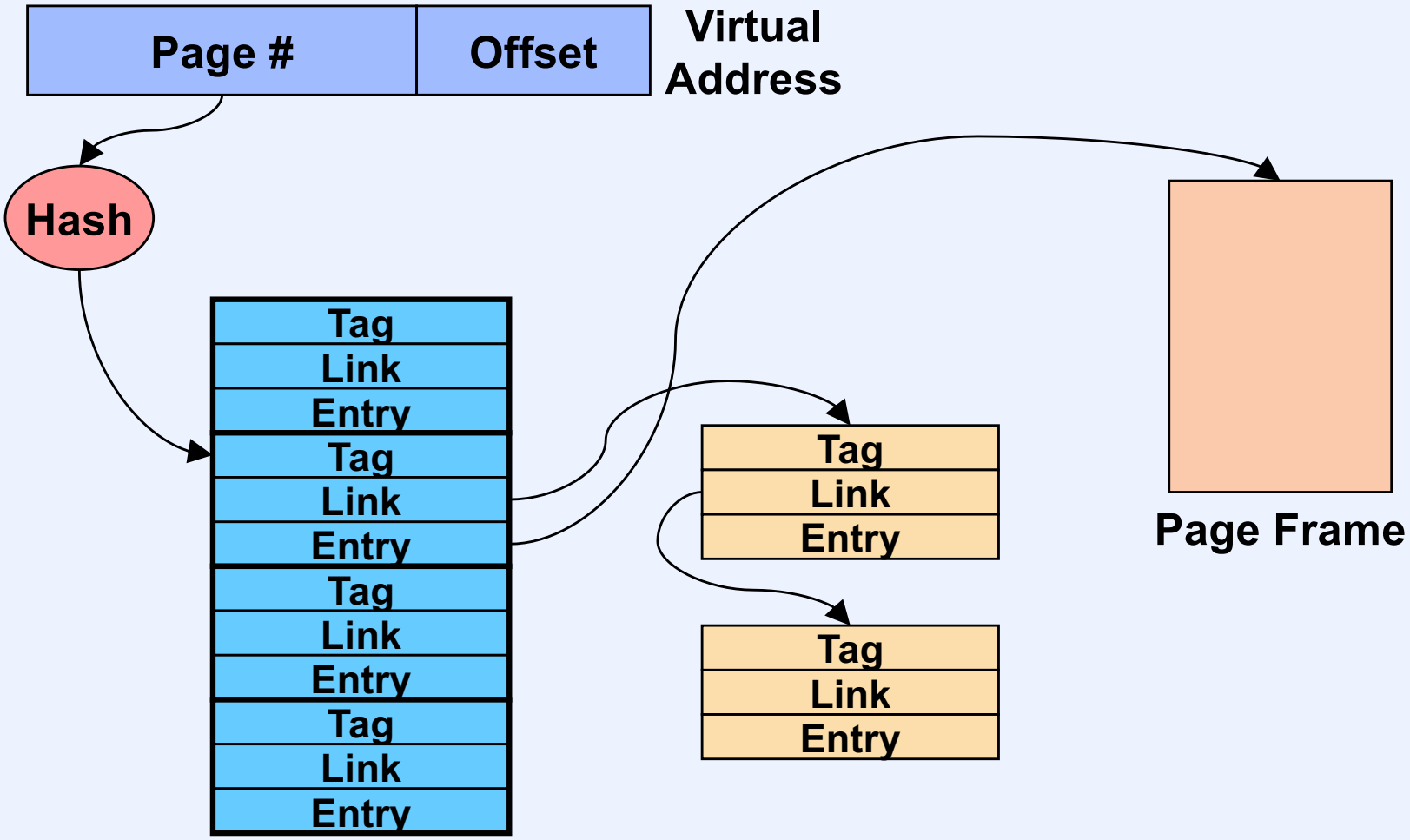
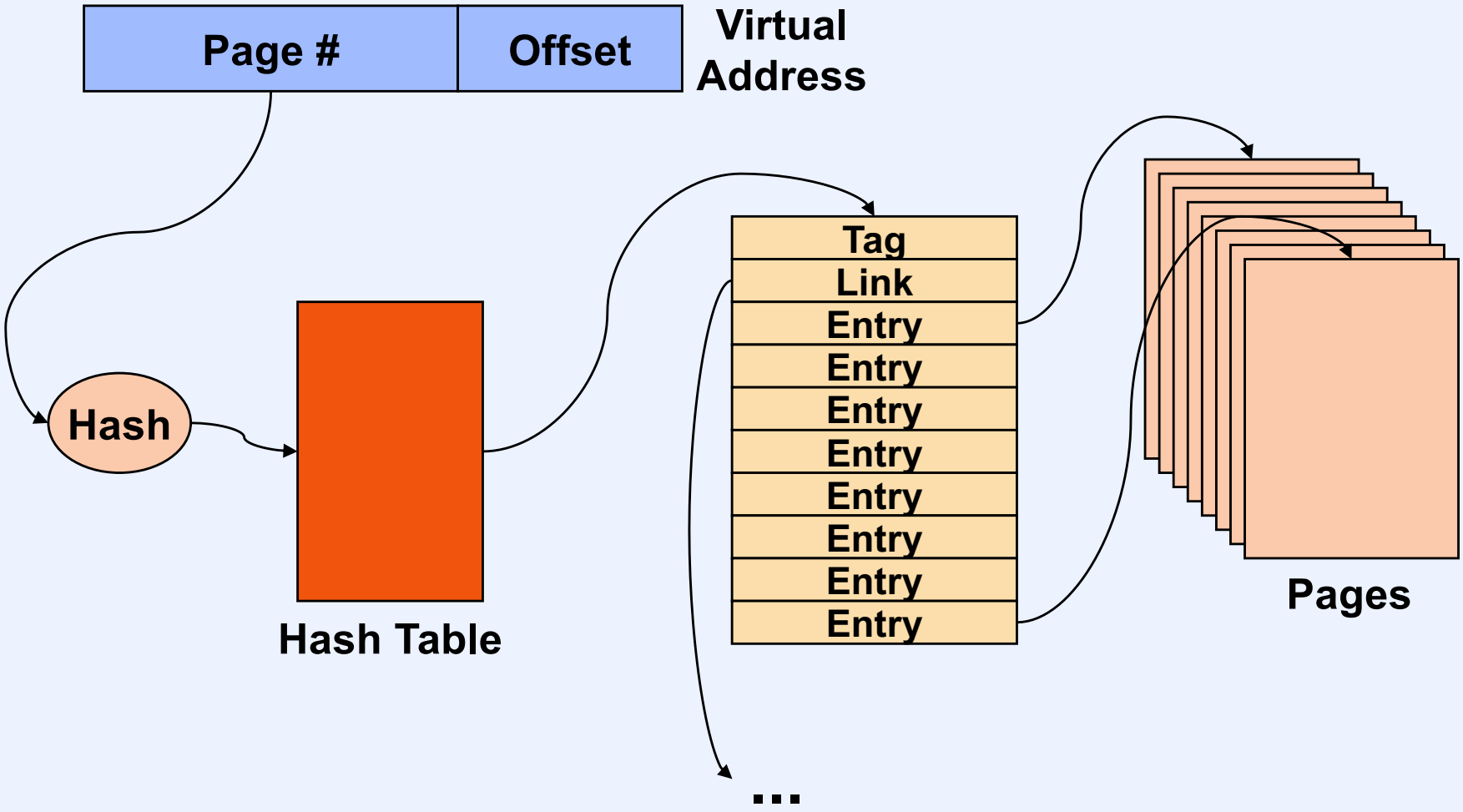
Memory Management Part 3

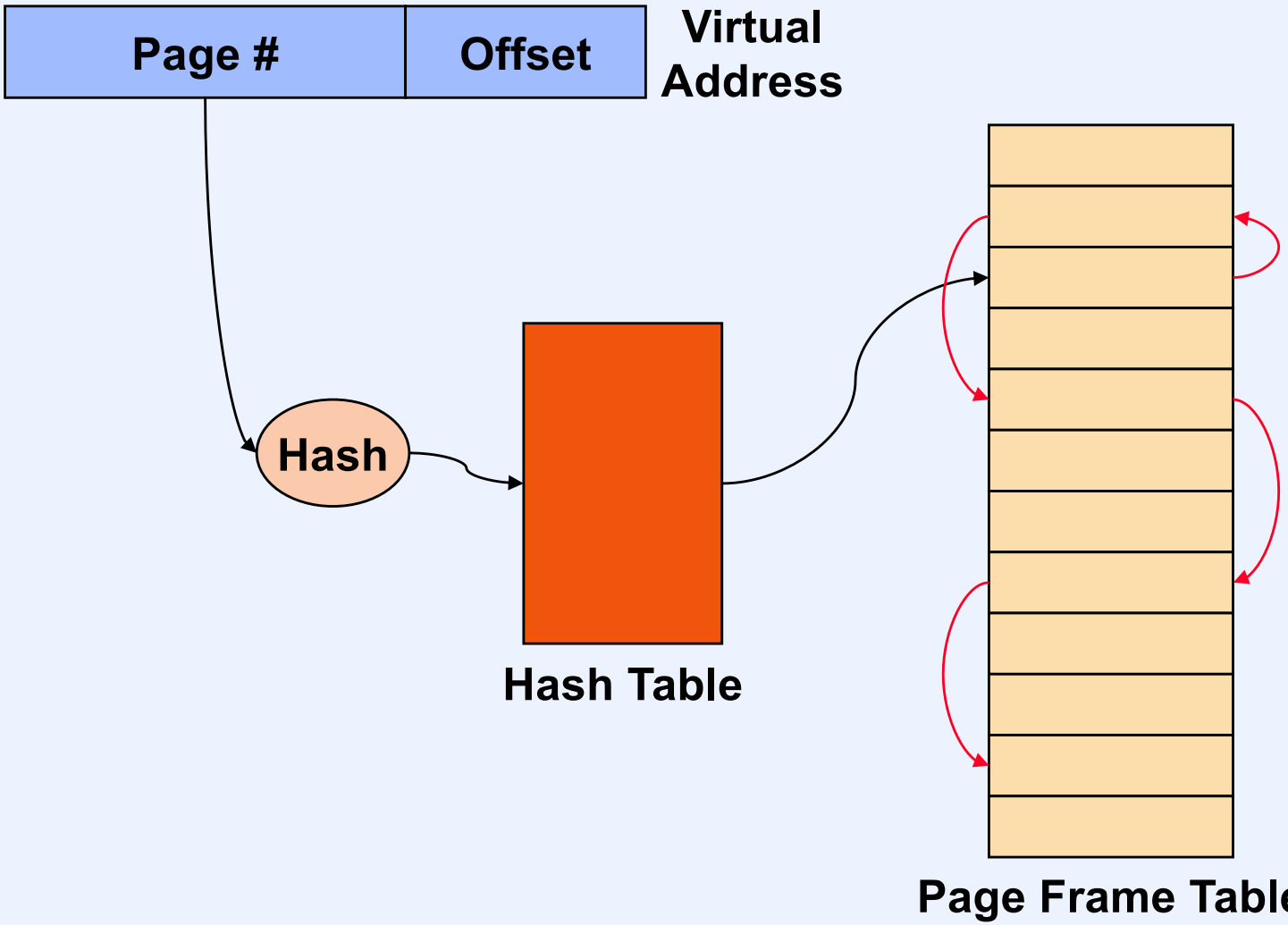
Hashed Page Tables



Clustered Page Tables



Inverted Page Tables



Quiz 1

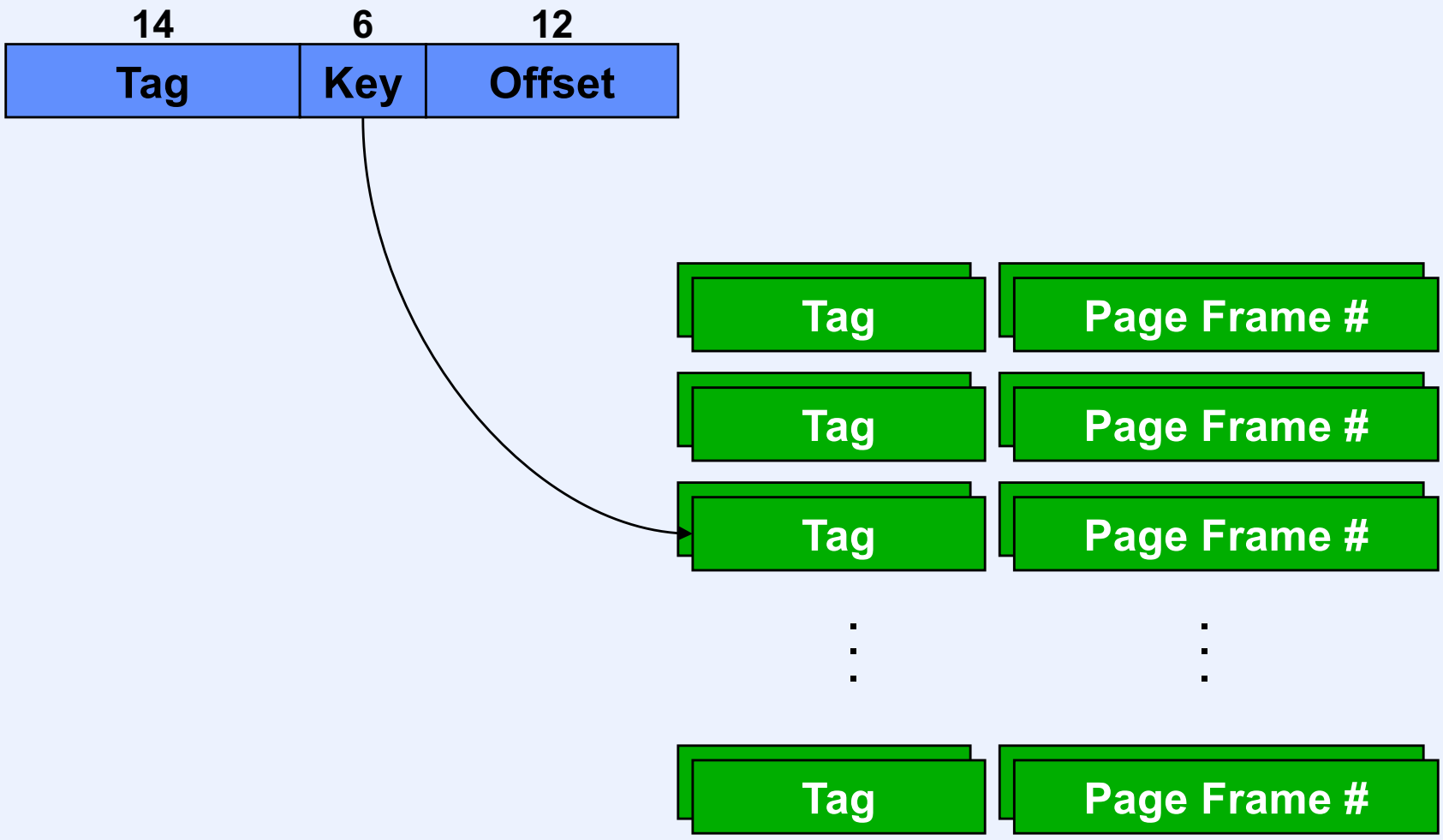
Normal page tables map virtual memory to real memory. More precisely, they map an address space and a location within that address space to real memory. Inverted page tables do the inverse mapping: given an address space ID and a location in real memory, they produce the corresponding virtual location.

a) Inverted page tables work with all Unix systems

b) They don't work with any Unix system

c) They don't work with Unix systems that support *mmap* with shared mappings

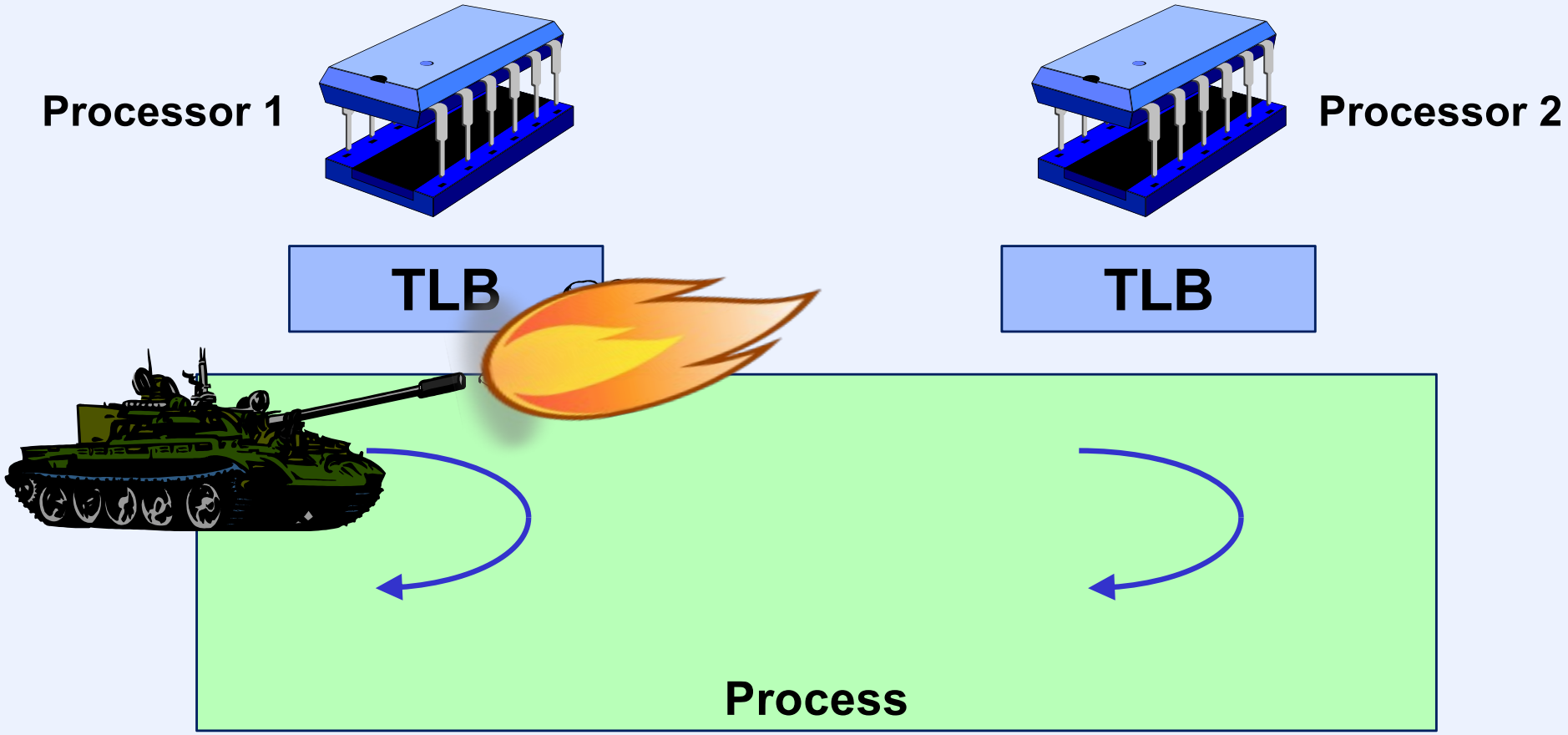
Translation Lookaside Buffers



TLBs and Mappings

- TLBs provide a cache for mappings from virtual addresses to real addresses
- Mappings change when
  - pages are removed (unmapped) from memory
  - when the address space is switched from one process's to another's
- OS must explicitly flush old contents of TLB
  - either individual entries or all of it

TLBs and Multiprocessors

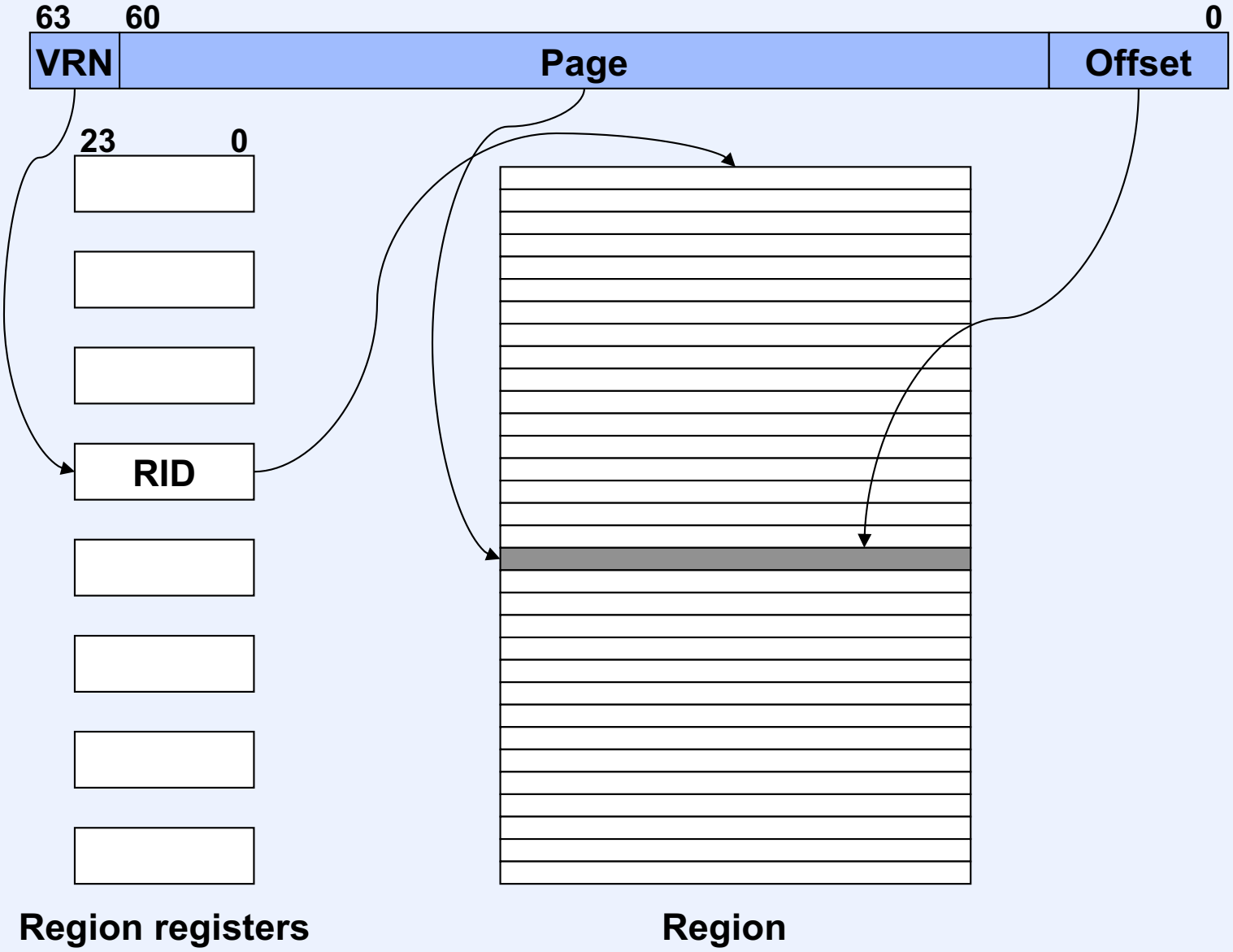


TLB Shootdown Algorithm

```
// shooter code
for all processors i sharing
  address space
   interrupt(i);
for all processors i sharing
  address space
   while (noted[i] == 0)
      ;
modify_page_table();
update_or_flush_tlb();
done[me] = 1;
```

```
// shootee i interrupt handler
receive_interrupt_from_
  processor j
noted[i] = 1
while (done[j] == 0)
   ;
flush_tlb()
```

Intel IA-64



Quiz 2

What did the IA-64 designers have in mind as a use for the region registers?

a) Region registers facilitate increasing the sizes of regions

b) They felt that 2^{64} bytes might be too small a limit for address-space size; the region registers provide a means for making it larger

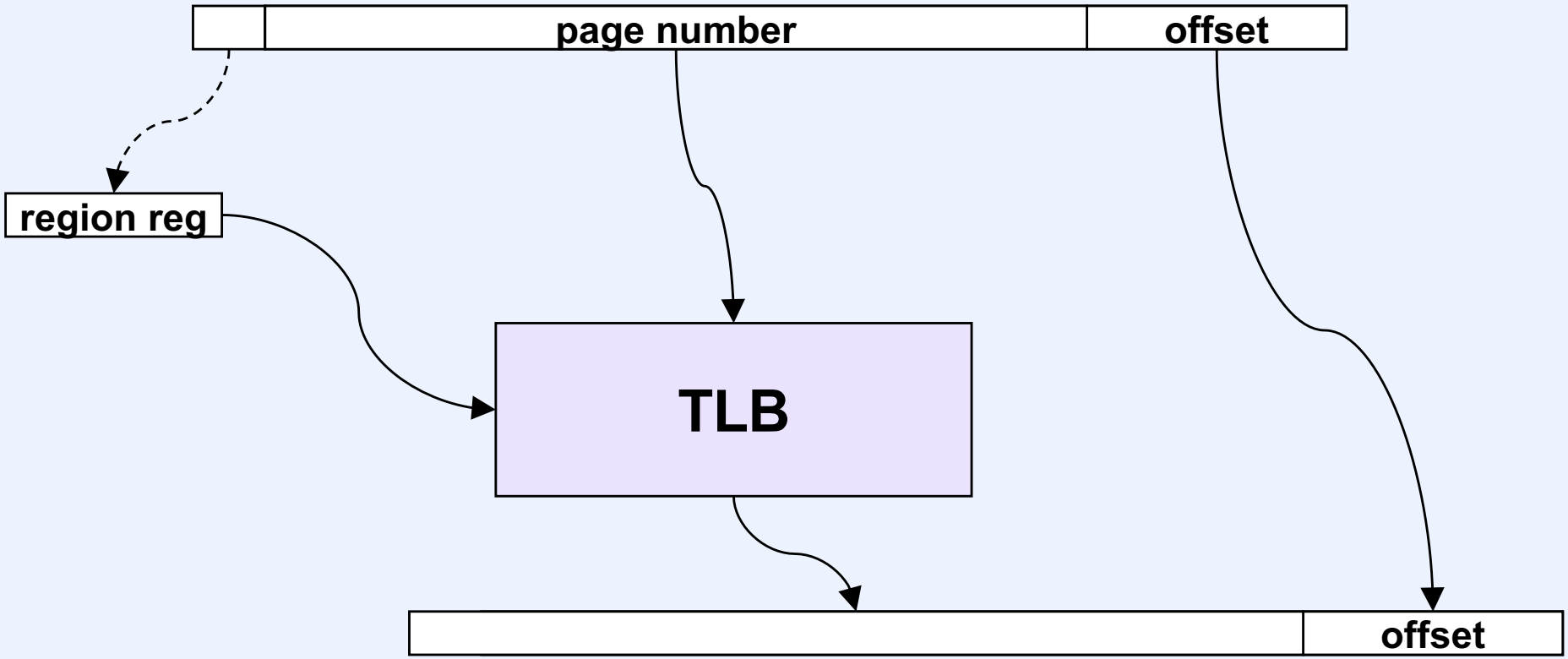
c) Region registers are a means for identifying and mapping in shared objects, such as libraries or datasets

d) Region-register contents are used in the TLB to uniquely identify regions; TLB flushes are minimized on process switches because it's clear which process an address belongs to

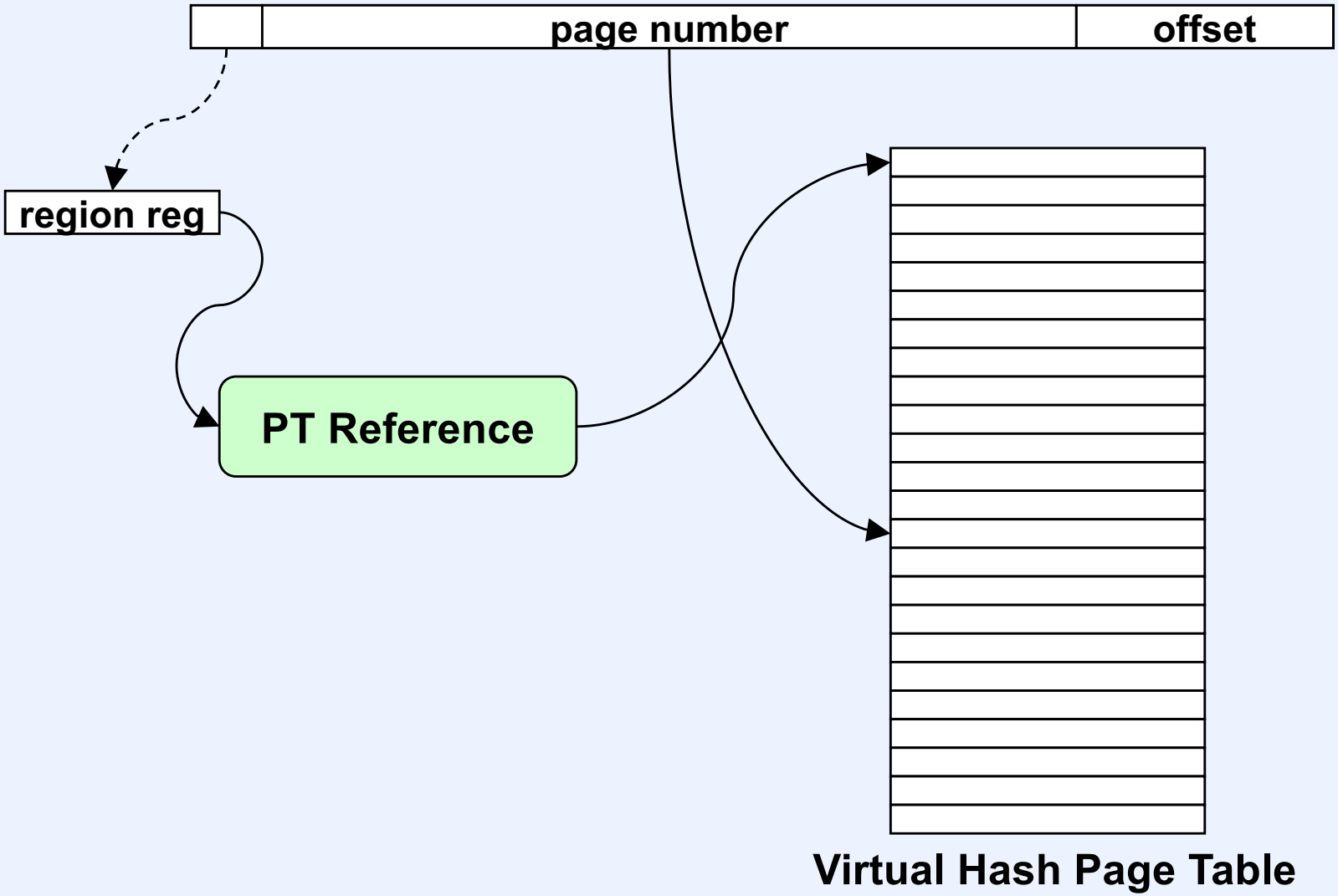
IA-64 Address Translation

- TLB
  - software-managed
- Virtual Hash Page Table (VHPT)
  - per-region linear page table
  - single large hashed page table

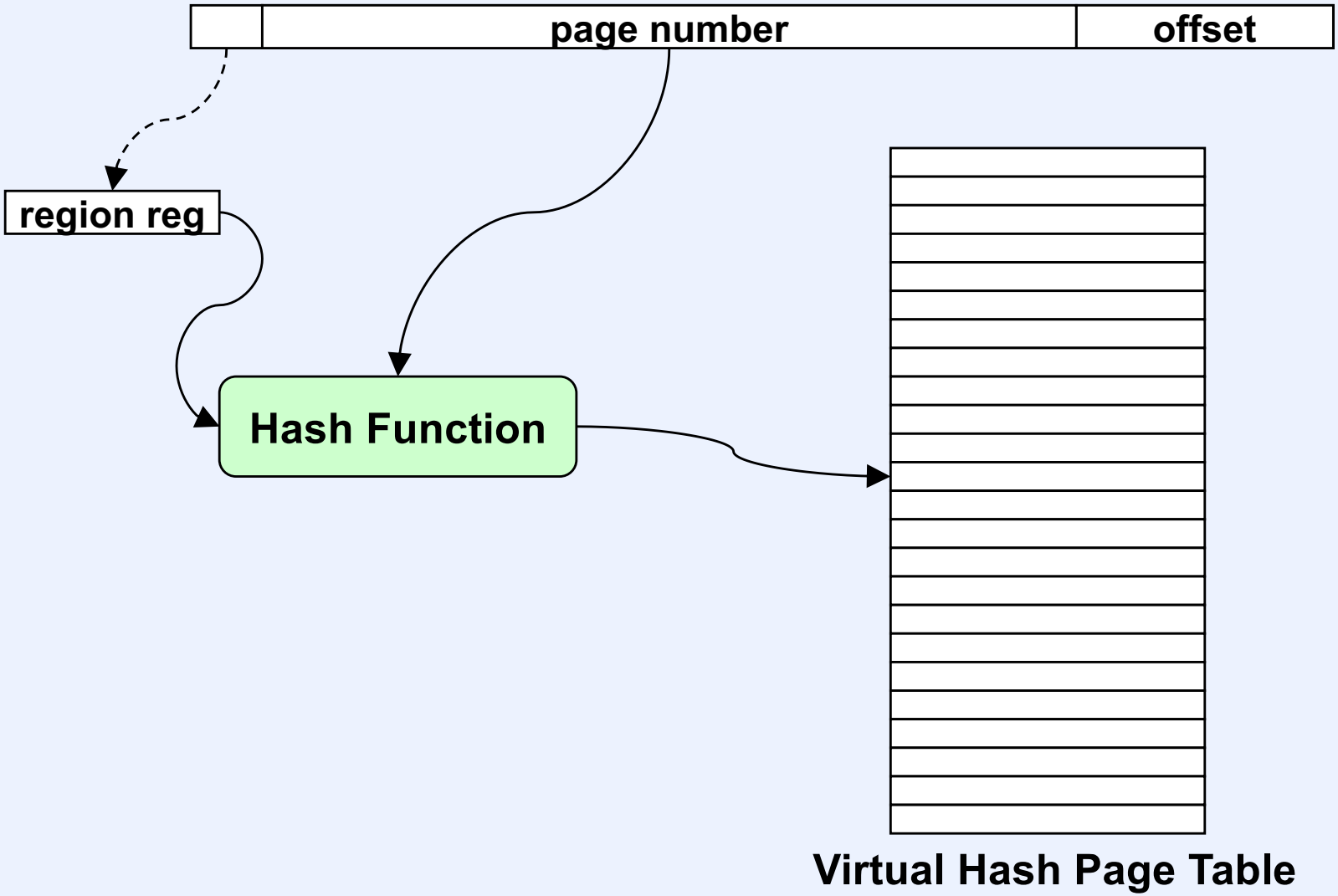
Translation: TLB



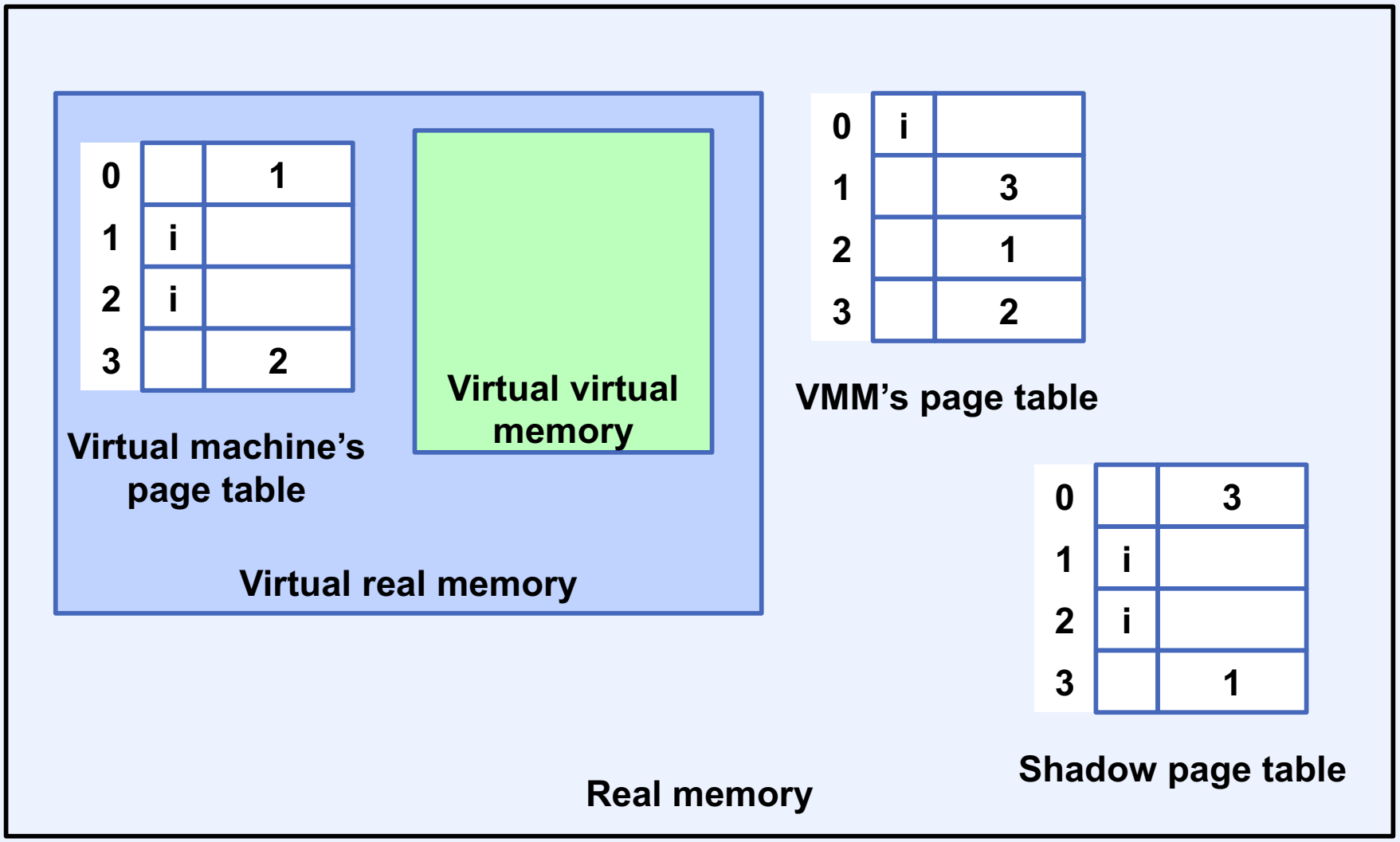
Translation: TLB Miss (LPT)



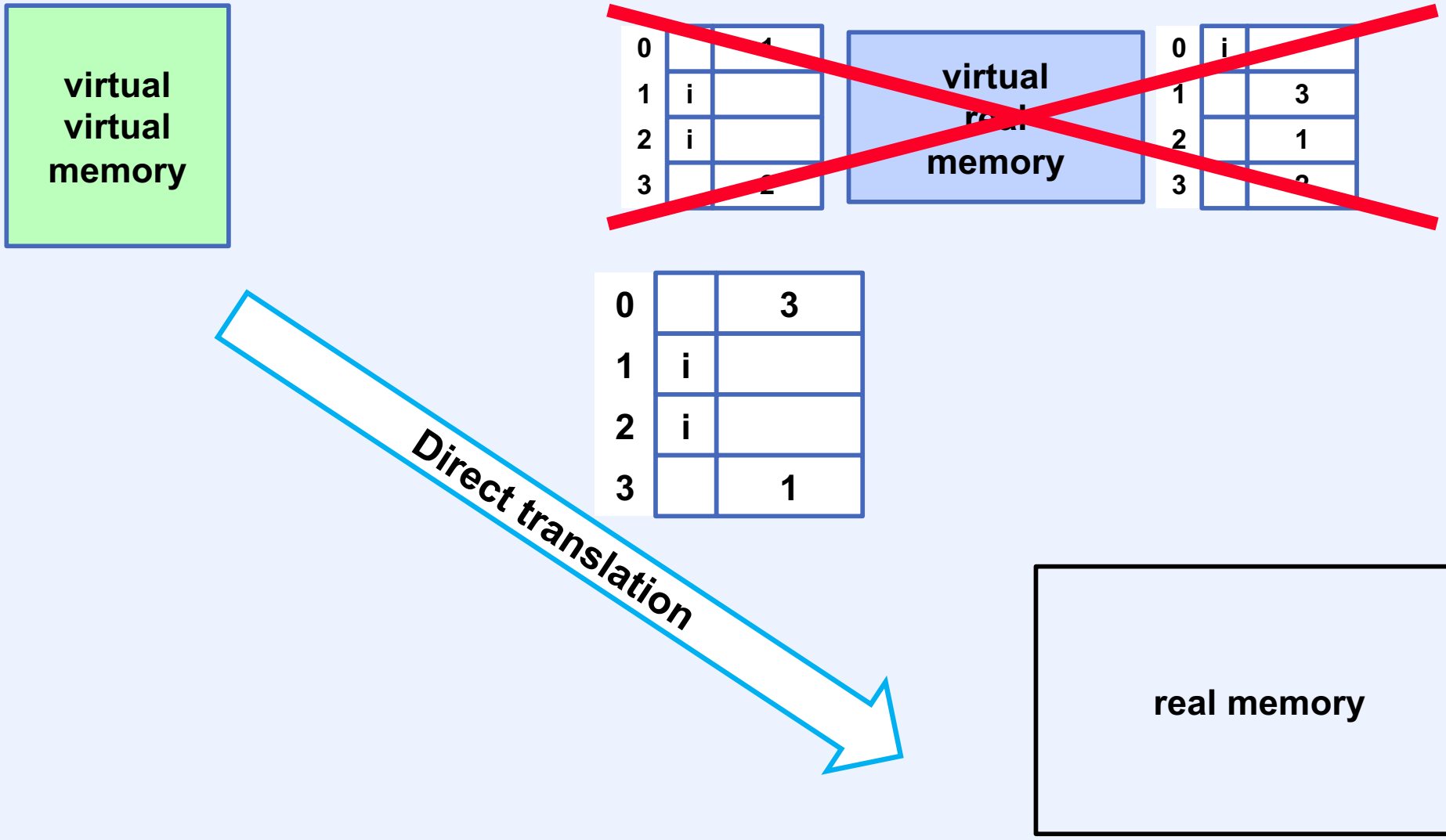
Translation: TLB Miss (HPT)



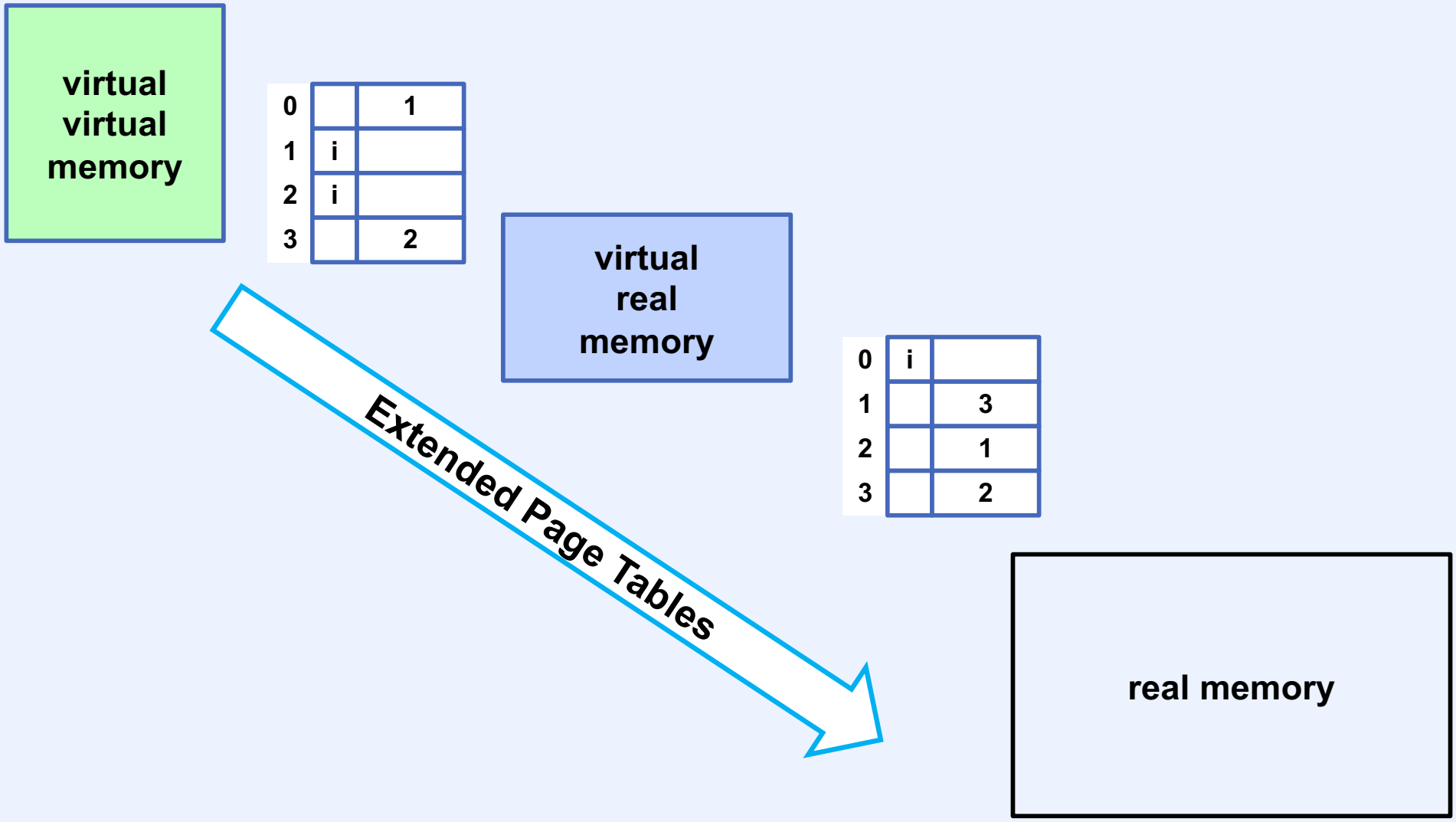
Virtual Machines Meet Virtual Memory



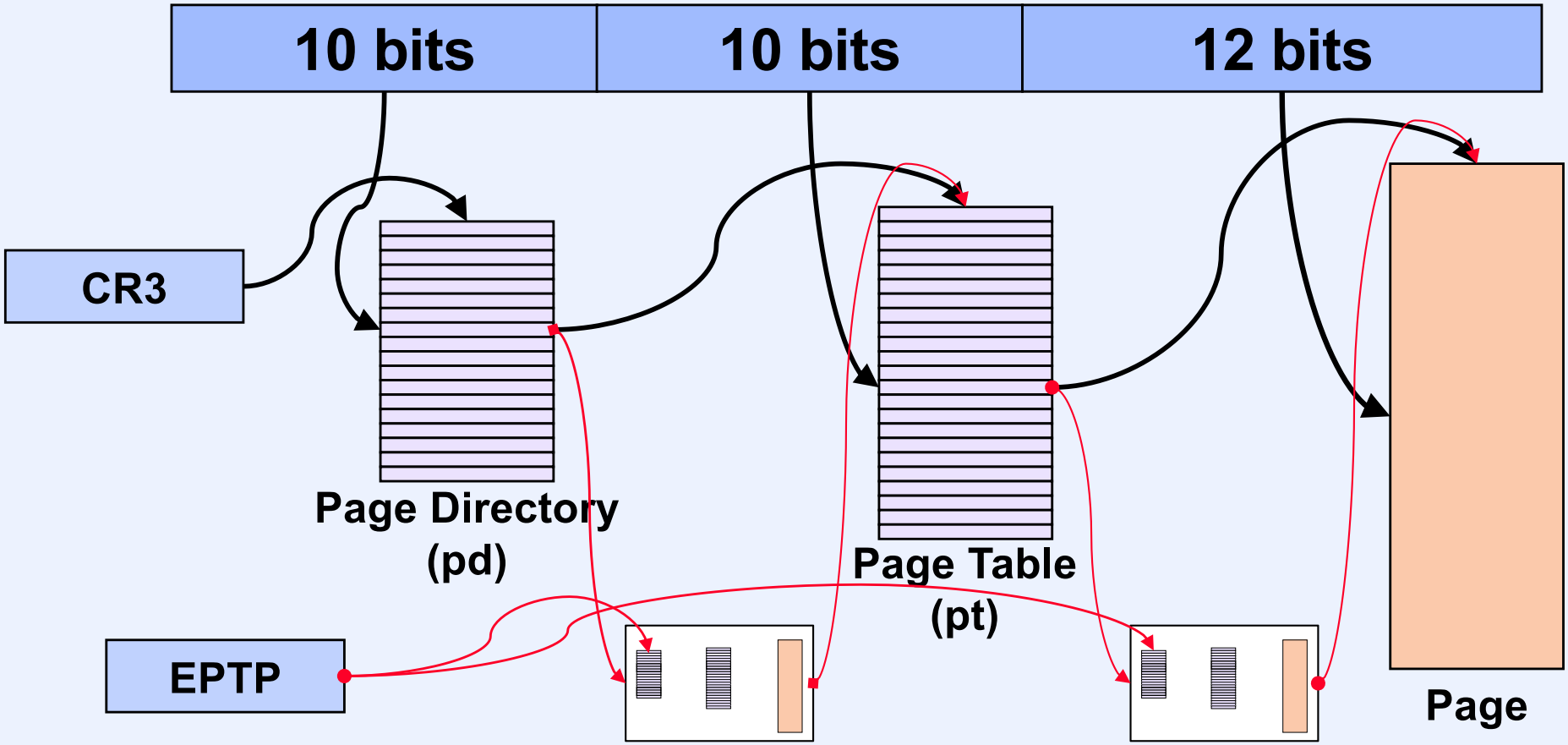
Paravirtualization to the Rescue



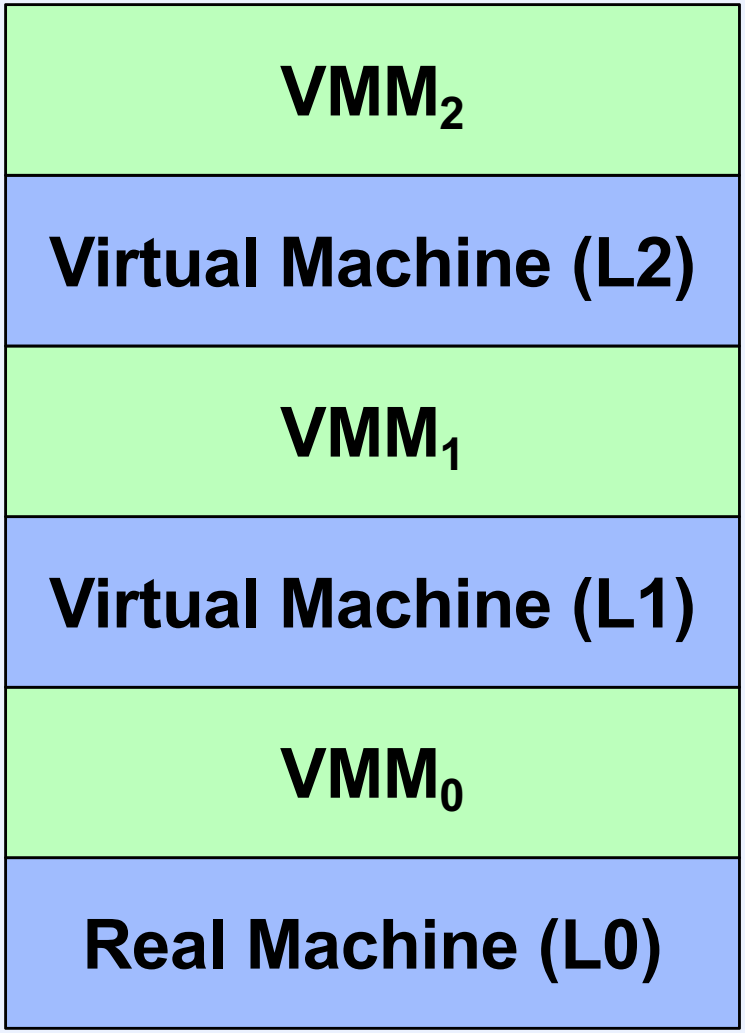
Hardware to the Rescue



x86 Paging with EPT



Nested Virtualization



Quiz 3

We'd like to virtualize EPT. Assume that setting EPTP causes a VMexit if done on a VMM that's not running in real ring -1. What does the VMM running at level 0 (in ring -1) do when it receives such a VMexit from a VMM running at level 1?

a) it sets EPTP to point to the composition of the page tables mapping VMM_0's address space to real memory and the page tables pointed to by the value being attempted to be put in EPT

b) nothing: the EPT mechanism is virtualized by the hardware

c) something else

VMX

- New processor mode: root
  - ring -1: root mode
  - rings 0-3: non-root mode
- Certain actions cause processor in non-root mode to switch to root mode
  - VMexit
- When in root mode, processor can switch back to non-root mode
  - VMenter

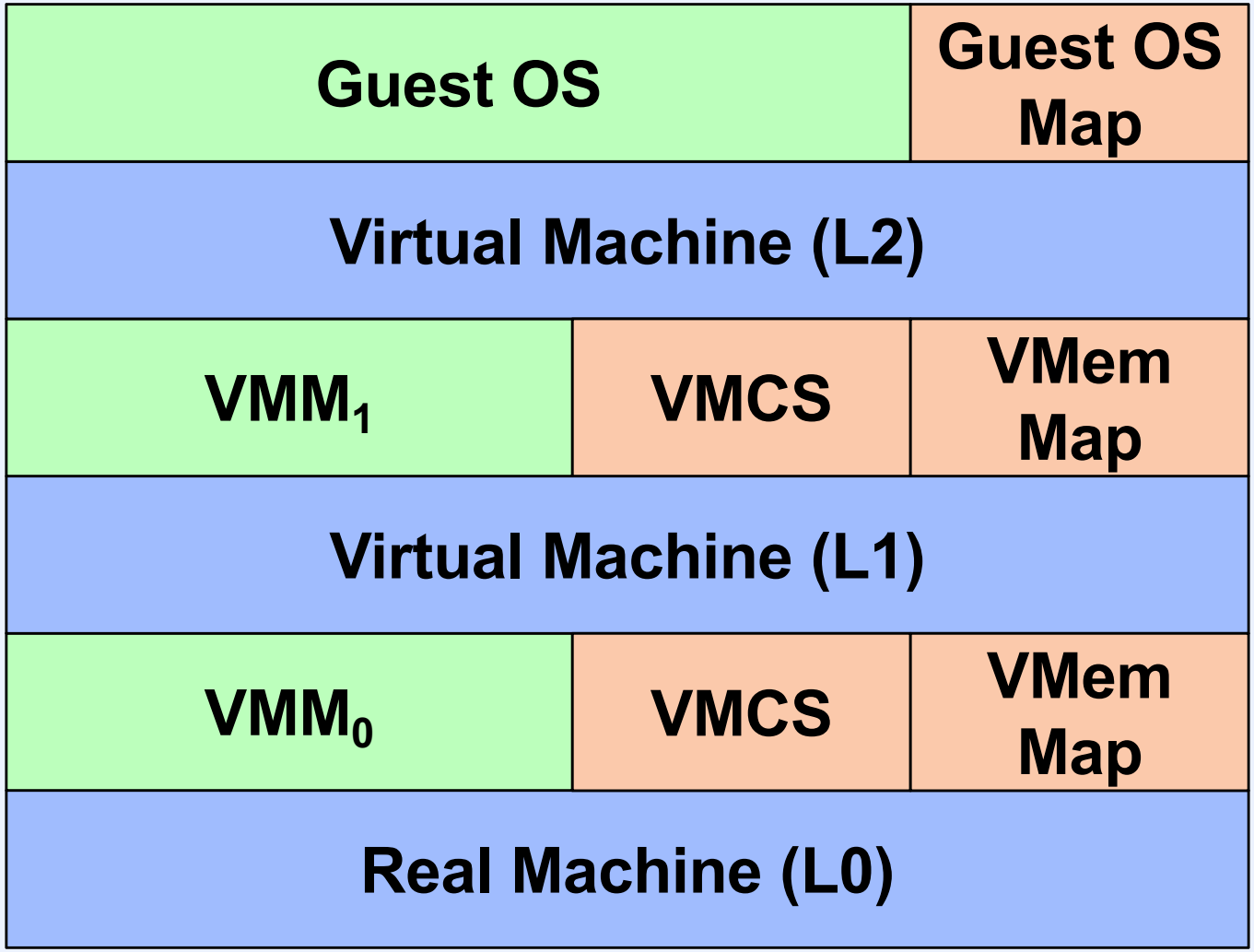
VMCS

- Virtual machine control structures
  - guest state
    - virtualized CPU registers (non-root mode)
  - host state
    - registers to be restored when switching to root mode (VMexit)
  - control data
    - which events in non-root mode cause VMexits

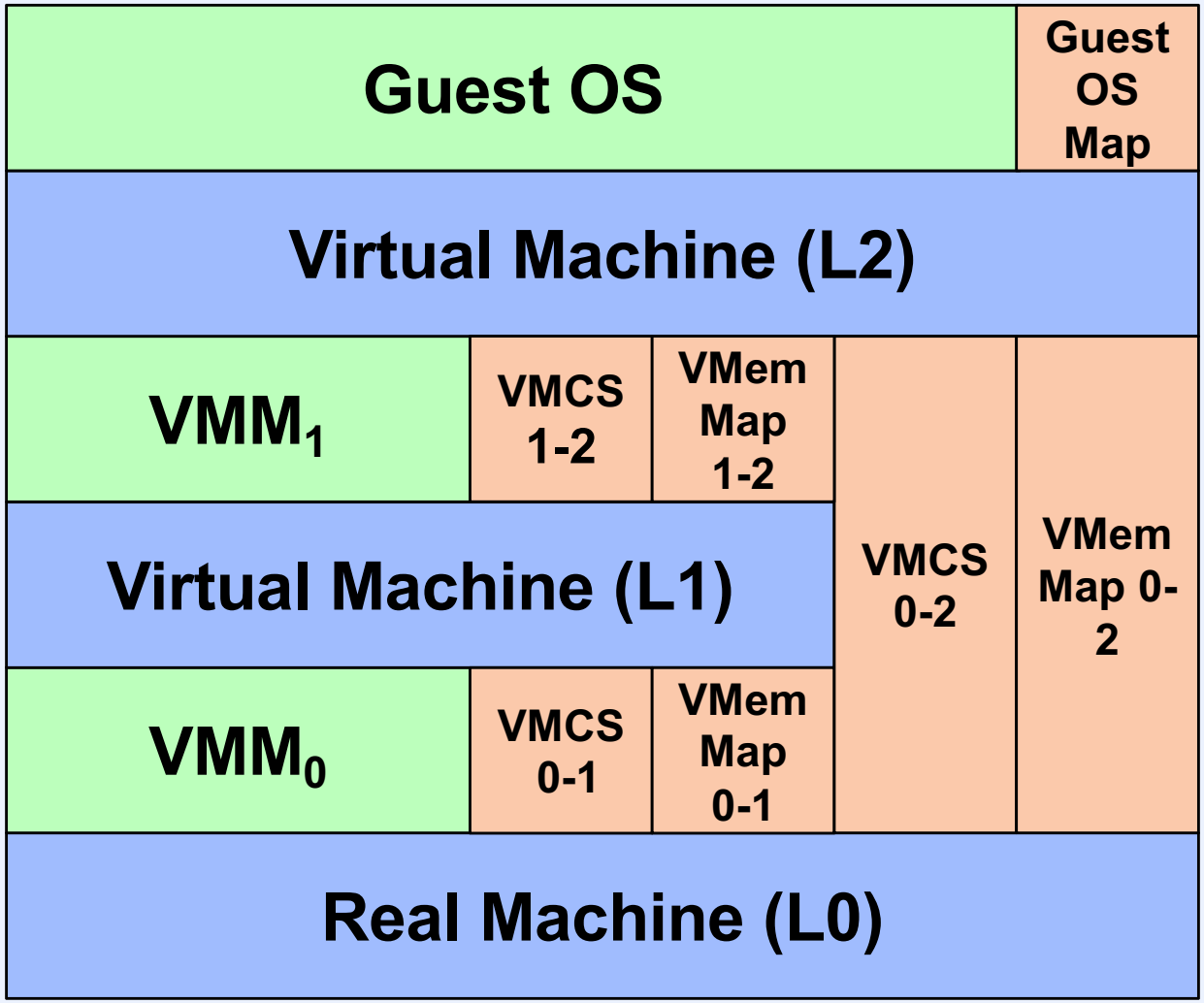
Nested Virtualization on VMX

- A VMM is designed to use VMX extensions (including EPT)
- It supports VMs that appear to be real x86's (but without VMX extensions)
- Can the VMM run in a VM of the level-0 VMM?

Nested Virtualization with VMX



Composed Virtualization



Traditional OS Paging Issues

- Fetch policy
- Placement policy
- Replacement policy

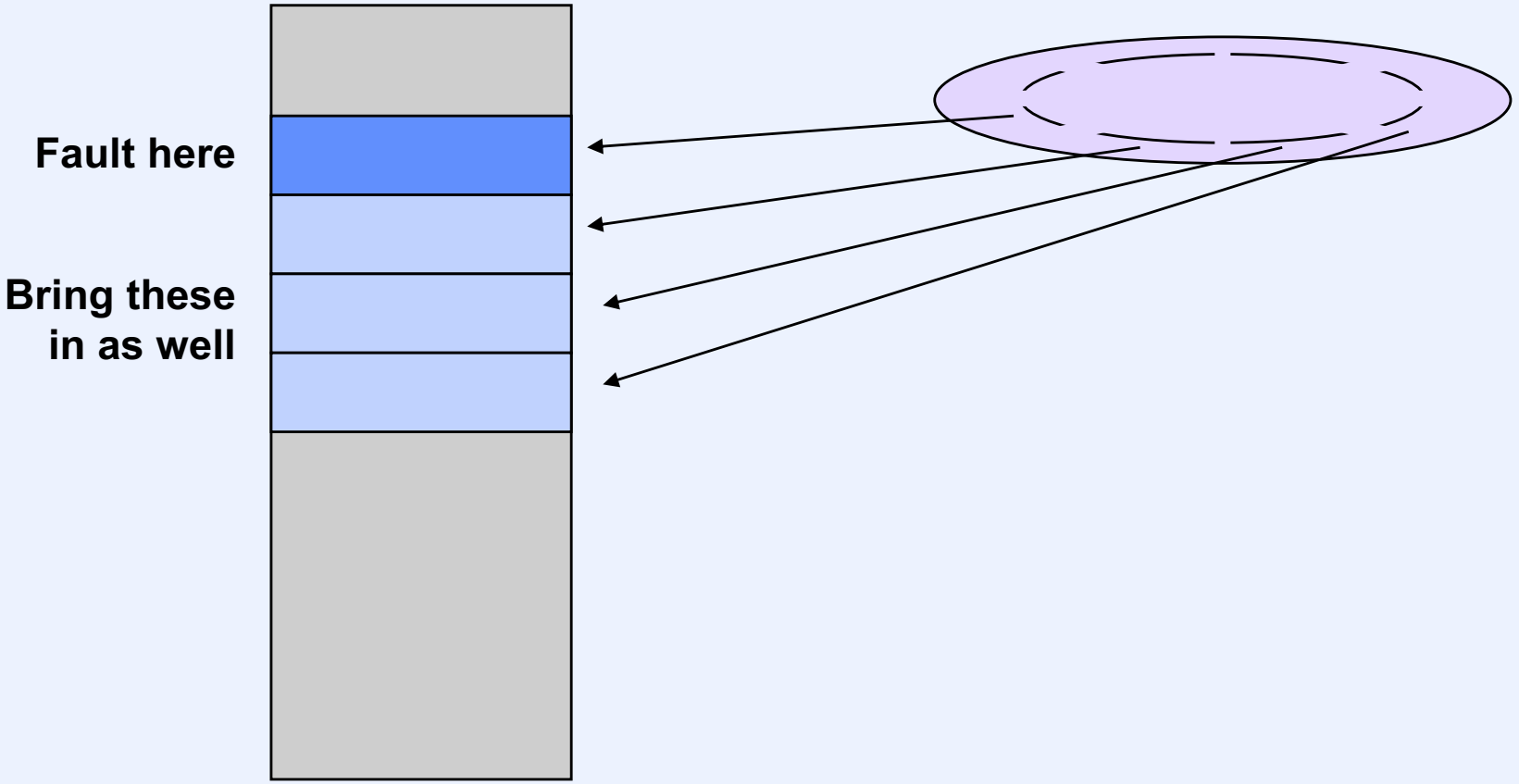
A Simple Paging Scheme

- Fetch policy
  - start process off with no pages in primary storage
  - bring in pages on demand (and only on demand — this is known as demand paging)
- Placement policy
  - it usually doesn't matter — put the incoming page in the first available page frame
- Replacement policy
  - replace the page that has been in primary storage the longest (FIFO policy)

Performance

1) Trap occurs (page fault)
2) Find free page frame
3) Write page out if no free page frame
4) Fetch page
5) Return from trap

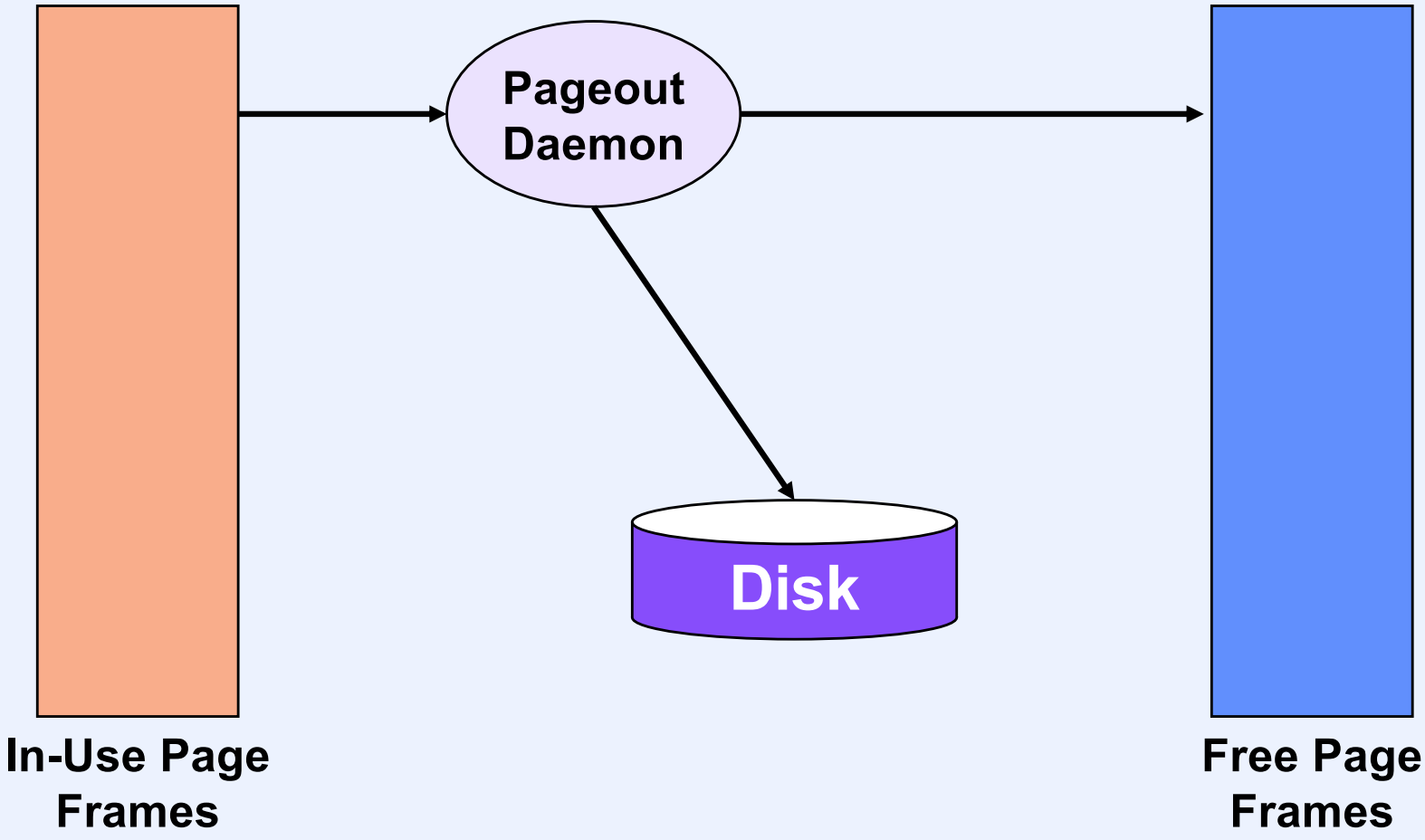
Improving the Fetch Policy



Improving the Replacement Policy

- When is replacement done?
  - doing it “on demand” causes excessive delays
  - should be performed as a separate, concurrent activity
- Which pages are replaced?
  - FIFO policy is not good
  - want to replace those pages least likely to be referenced soon

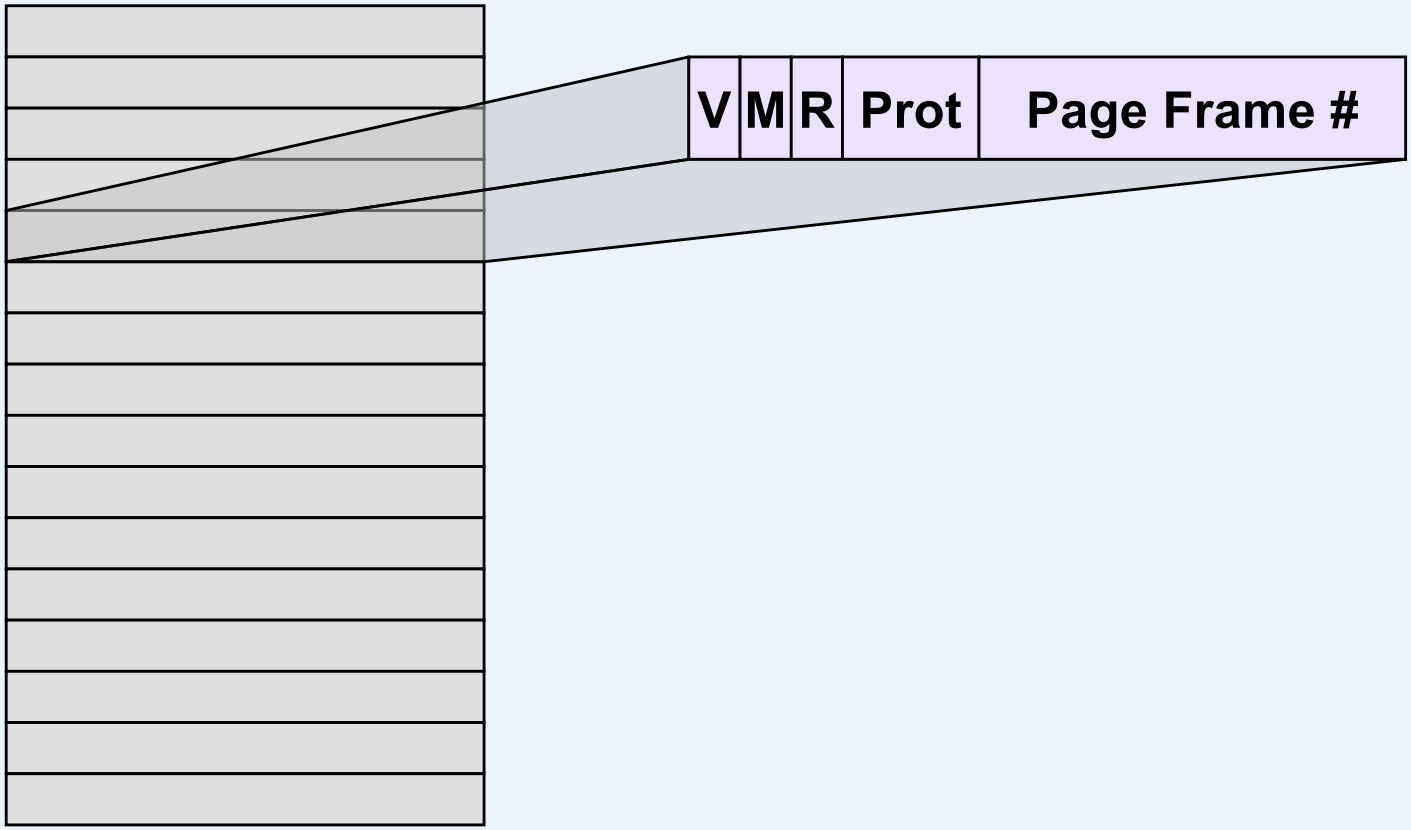
The “Pageout Daemon”



Choosing the Page to Remove

- Idealized policies:
  - FIFO (First-In-First-Out)
  - LRU (Least-Recently-Used)
  - LFU (Least-Frequently-Used)
- Optimal
  - replace page so as to minimize number of page faults
    - replace page whose next reference is furthest in the future

Implementing LRU



Quiz 4

Your computer is running one process. Pretty much all available real memory is being actively used and processor utilization is around 90%. You now add another process that's similar to the first in terms of both memory and processor utilization (though it's running a different program). Assume the LRU page replacement policy is used.

a) Processor utilization will rise to nearly 100%

b) Processor utilization will stay at around 90%

c) Processor utilization will drop precipitously

Global vs. Local Allocation

- Global allocation
  - all processes compete for page frames from a single pool
- Local allocation
  - each process has its own private pool of page frames

Thrashing

- Consider a system that has exactly two page frames:
  - process A has a page in frame 1
  - process B has a page in frame 2
- Process A causes a page fault
- The page in frame 2 is removed
- Process B faults; the page in frame 1 is removed
- Process A resumes execution and faults again; the page in frame 2 is removed
- ...

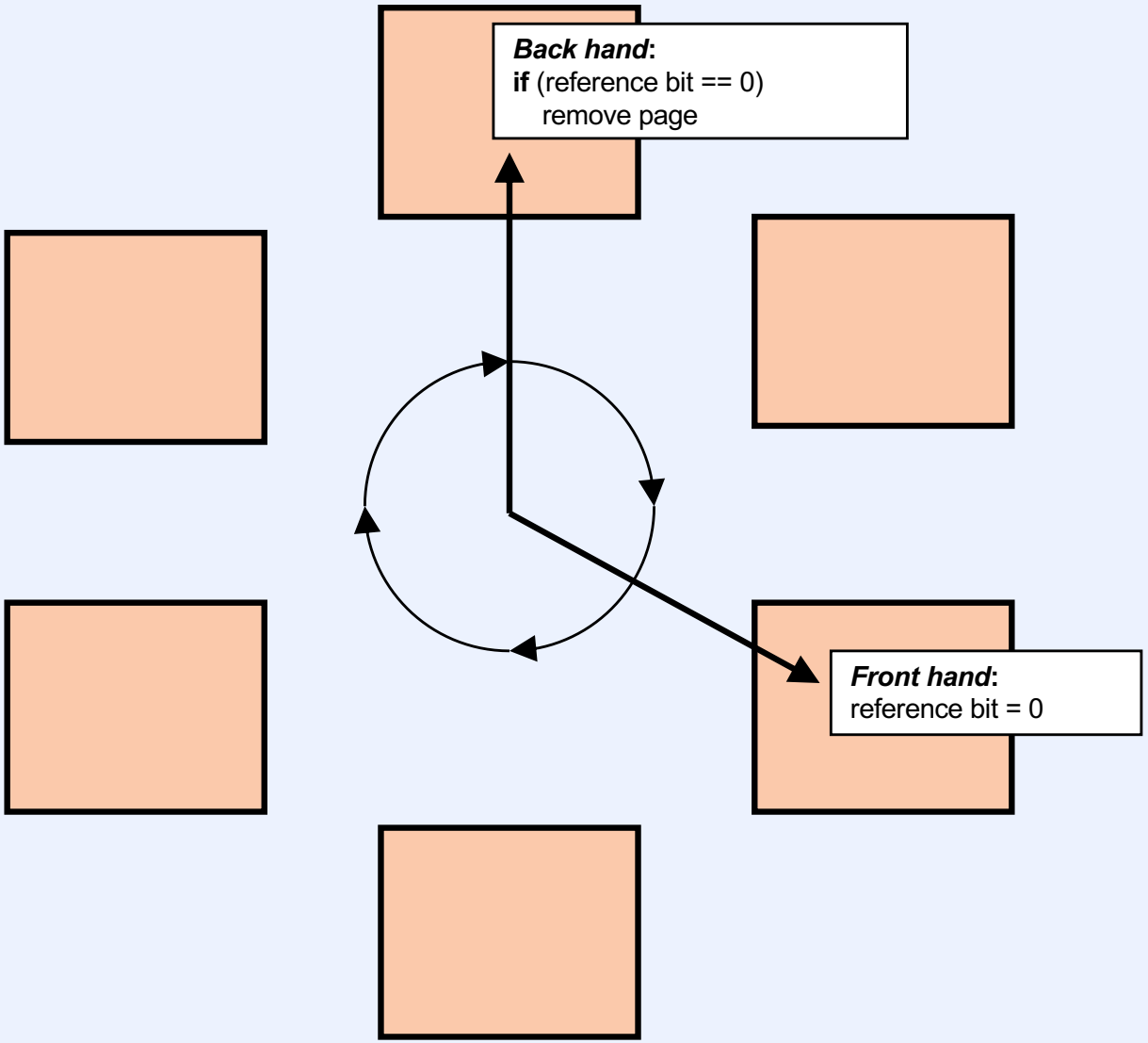
The Working-Set Principle

- The set of pages being used by a program (the working set) is relatively small and changes slowly with time
  - WS(P,T) is the set of pages used by process P over time period T
- Over time period T, P should be given |WS(P,T)| page frames
  - if space isn't available, then P should not run and should be swapped out

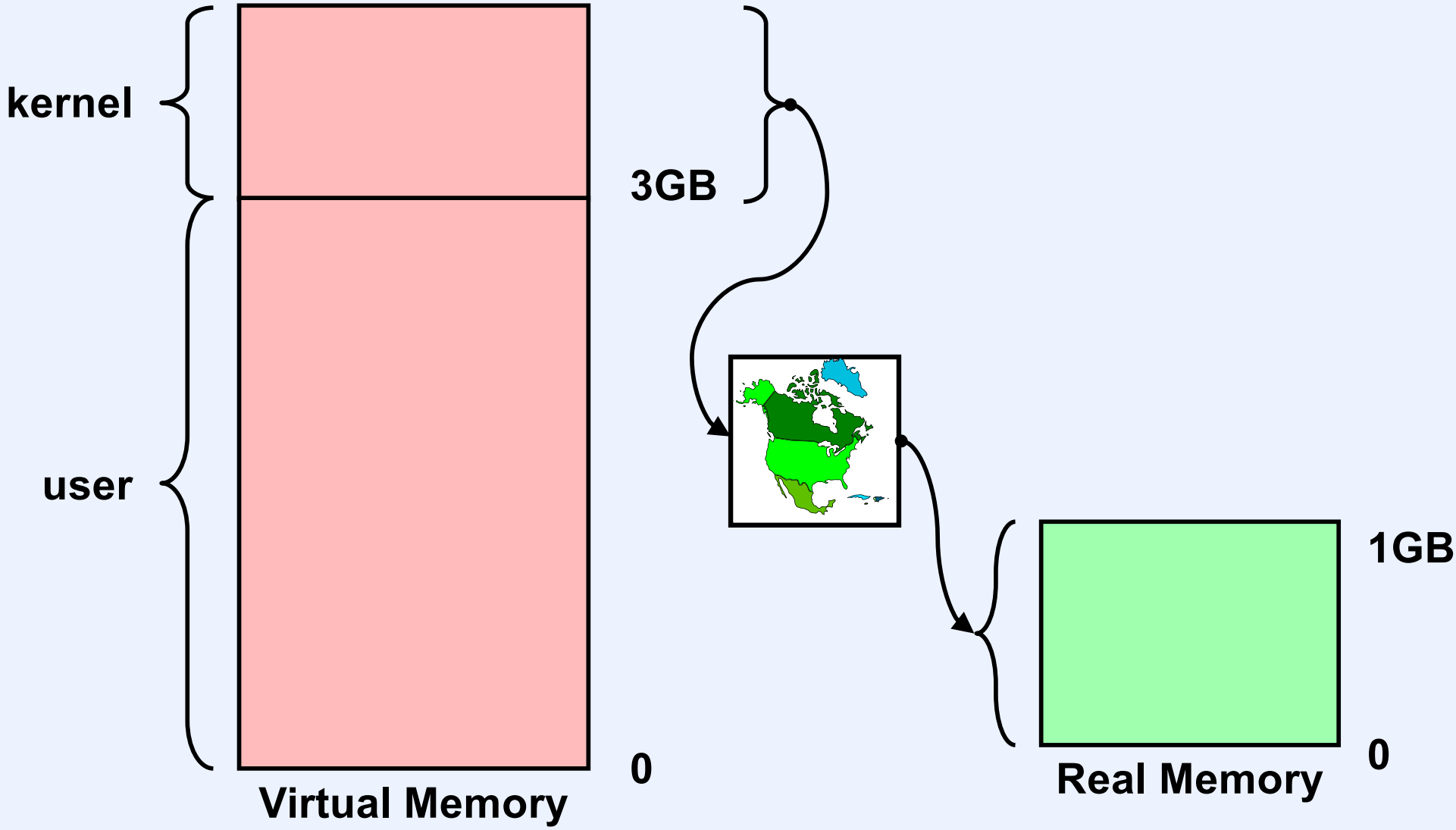
Two Issues

- If a process is active, which of its pages should be in real memory?
- If there is too much of a demand for memory, which processes should run (and which should not run)?

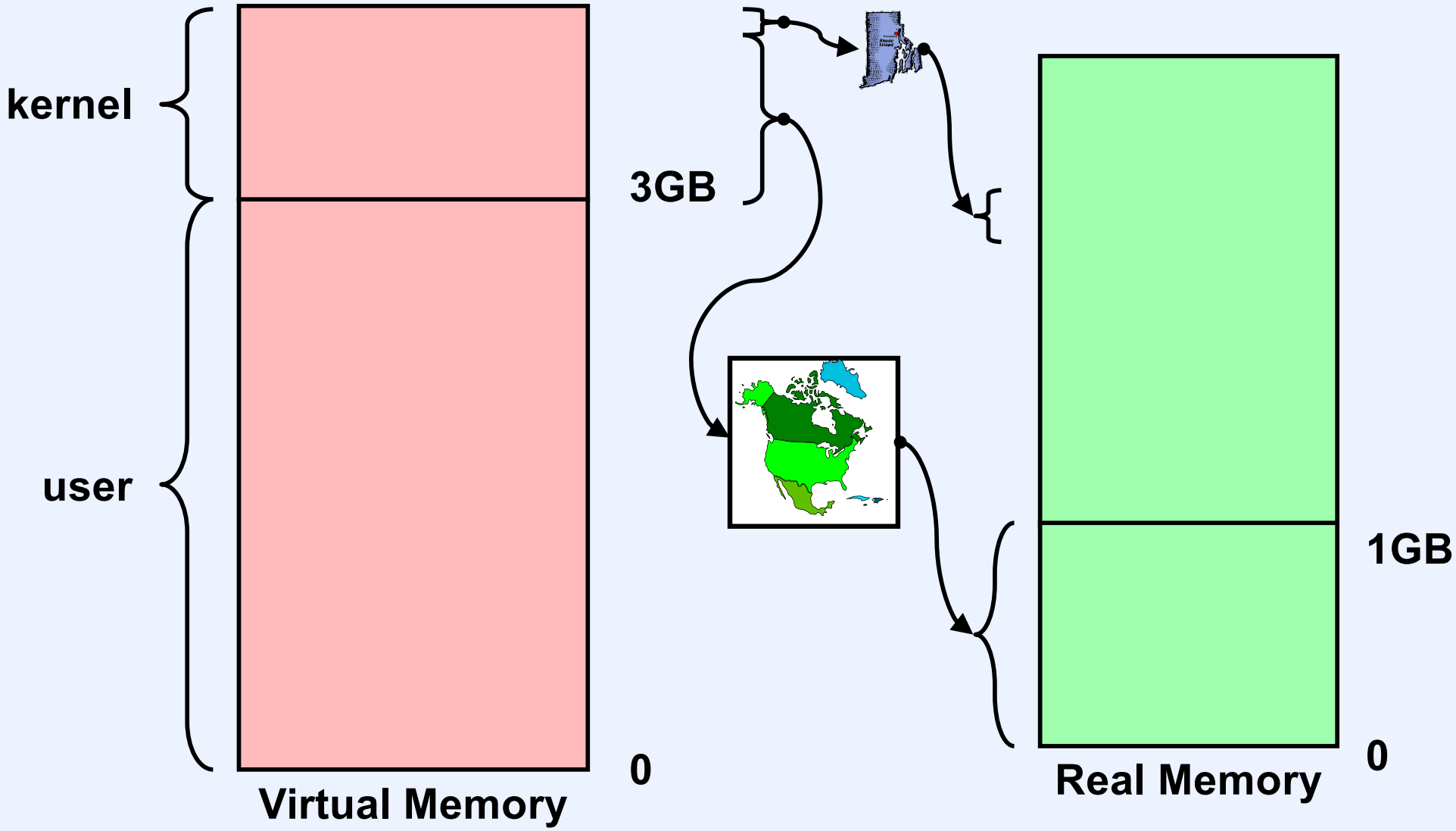
Clock Algorithm



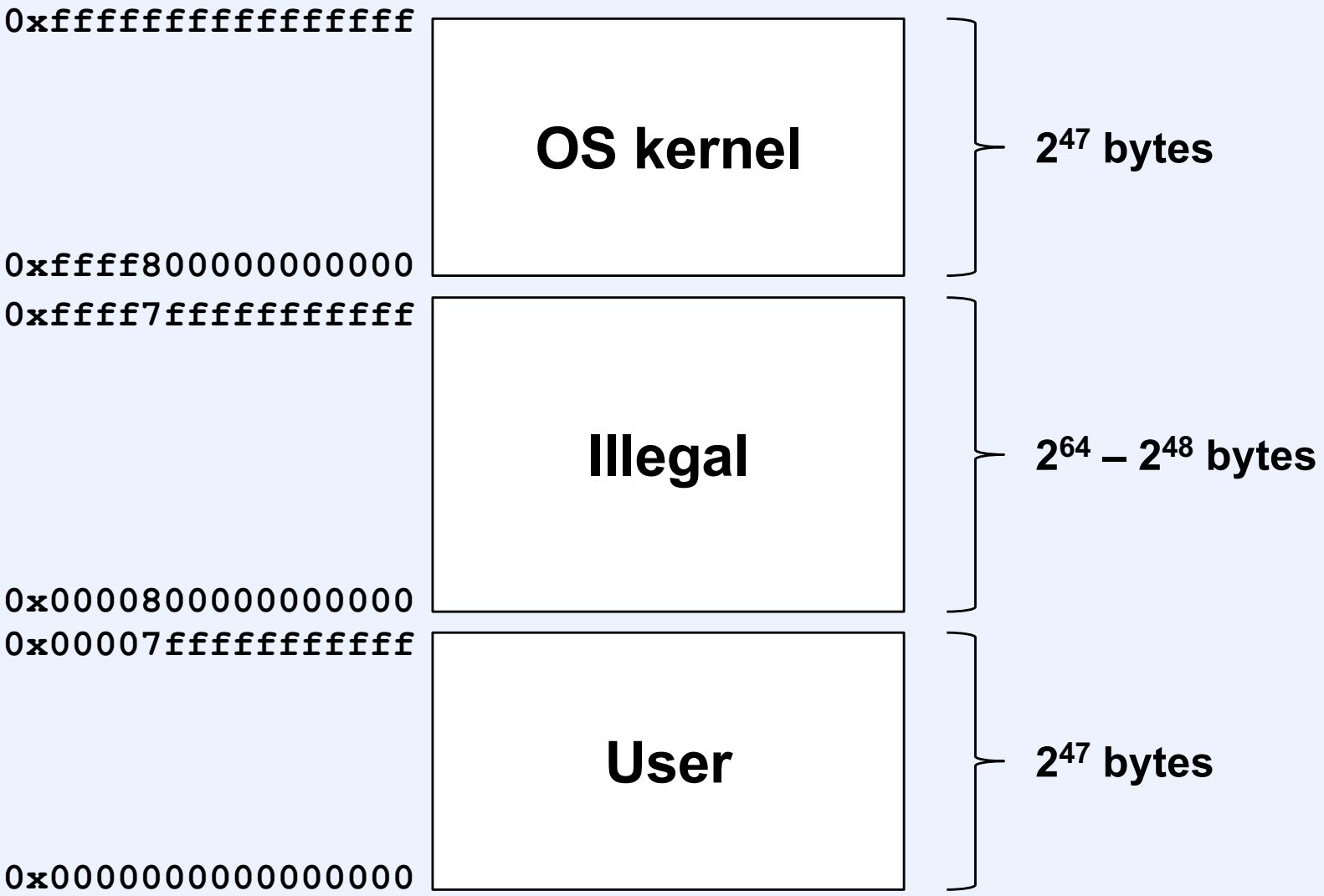
Linux and Real Memory



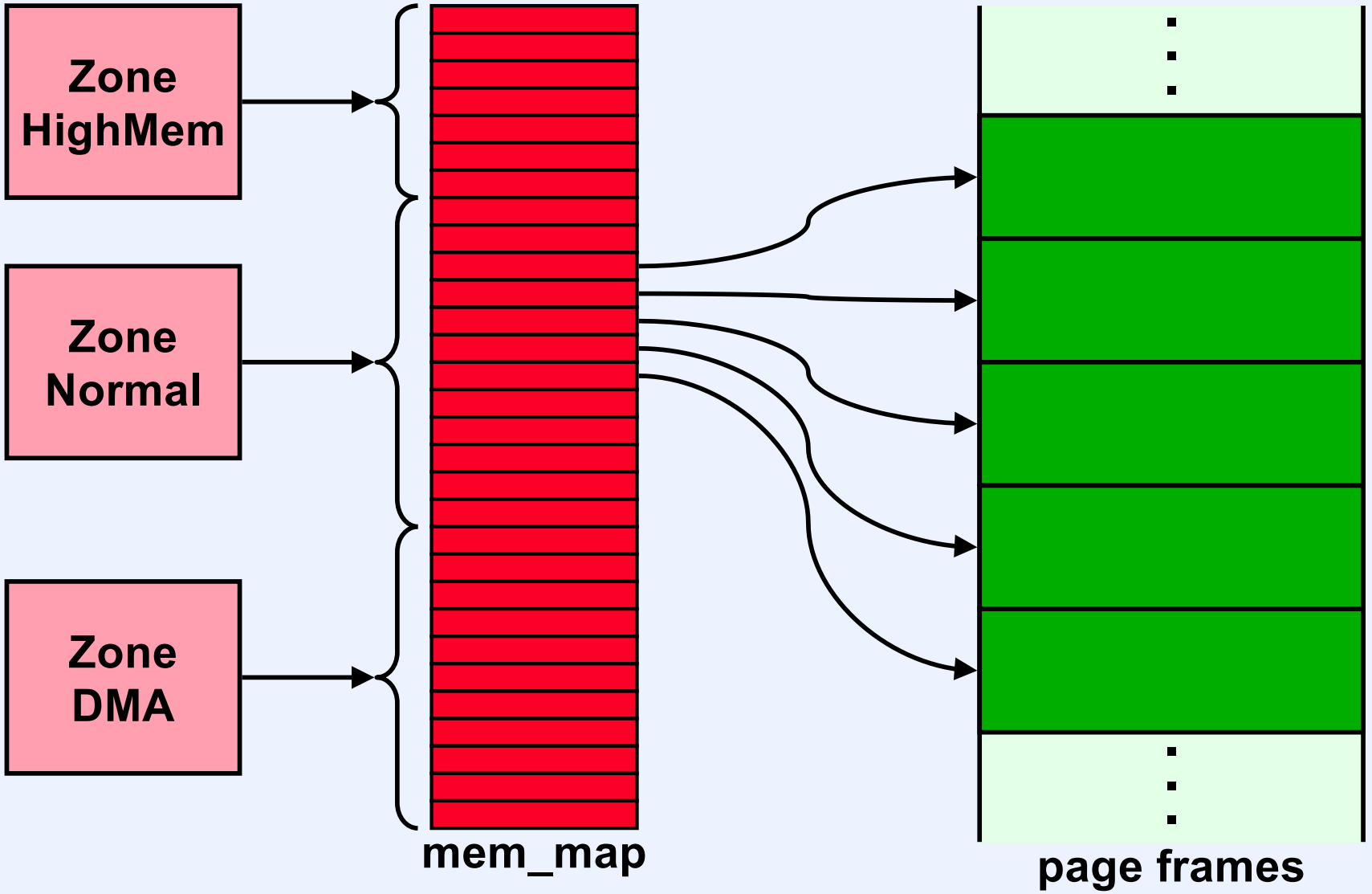
Lots of Real Memory



Address Space



Mem_map and Zones



Quiz 5

We have a disk whose controller uses 64-bit memory addresses. We'd like to set it up for a read that will transfer 32K bytes. Thus the block will be read into a buffer that occupies eight 4KB pages.

a) The buffer must occupy eight page frames of contiguous real memory

b) Since our system uses virtual memory, the buffer occupies contiguous pages of virtual memory, which may be mapped into non-contiguous page frames of real memory

c) As in b, except that the contiguous pages of virtual memory need not have valid mappings (and thus page faults are generated and handled)