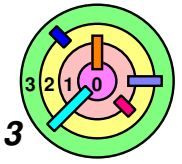
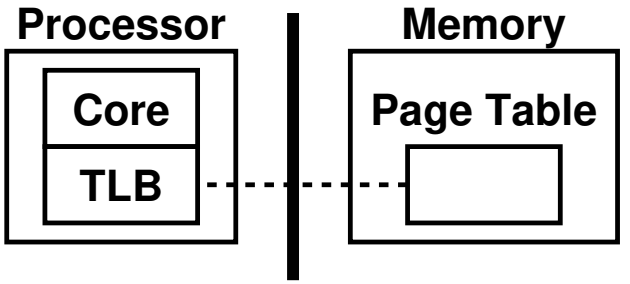
7.2 Hardware Support for Virtual Memory

- Forward-Mapped Page Tables
- Linear Page Tables
- Hashes Page Tables
- *Translation Lookaside Buffers*
- 64-Bit Issues
- Virtualization



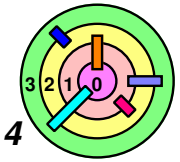
Translation Lookaside Buffers (TLB)

- Table lookup requires one memory access
  - to access memory starting with a virtual address will take at least *two* memory accesses
    - that's one access too many



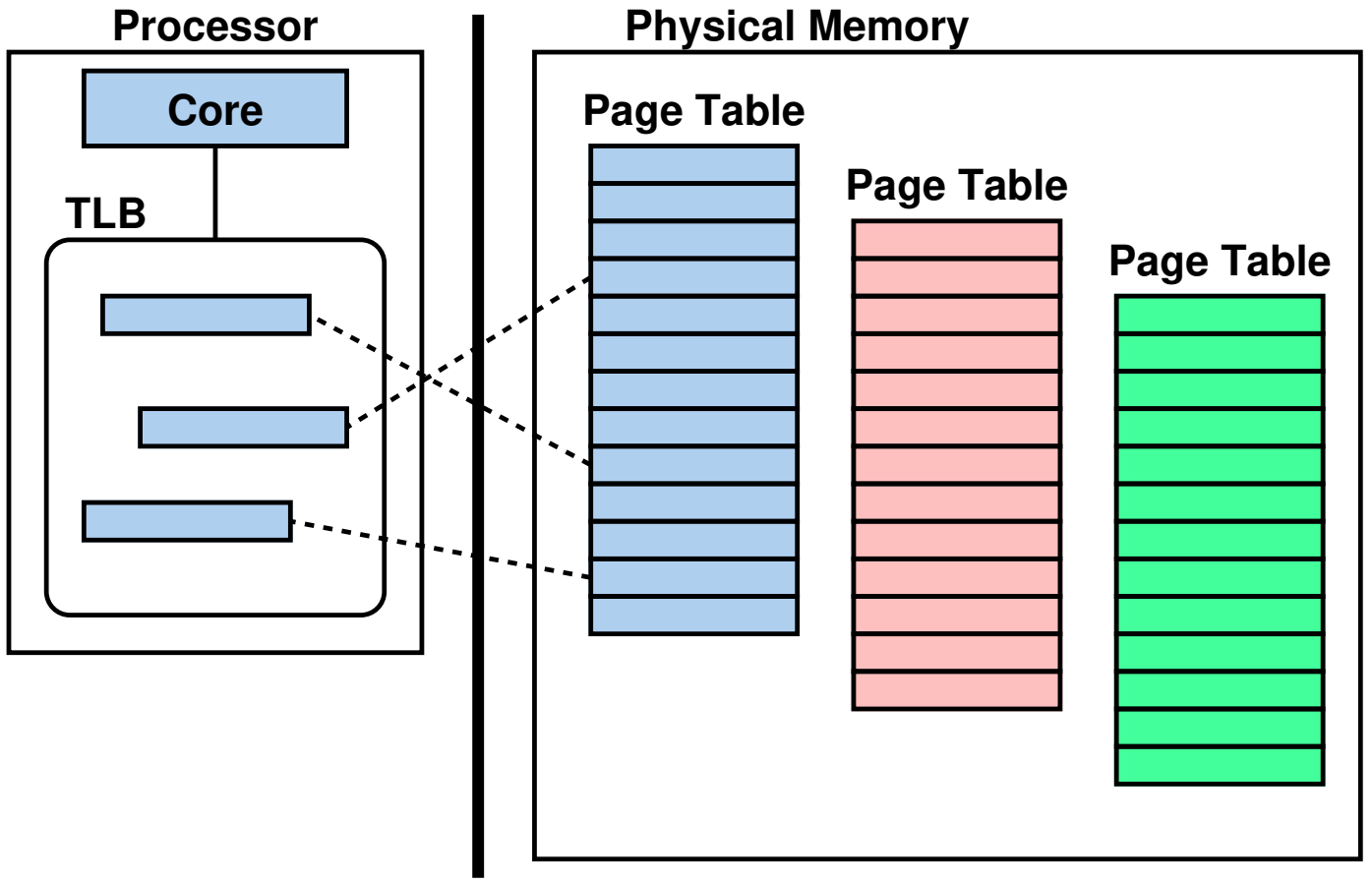
- If the processor has additional memory
  - it can be used to *cache page table entries*
  - *Translation Lookaside Buffer (or TLB)*
    - the TLB caches the *mapping* from virtual page number to physical page number (along with other information in the page table entry)
    - hopefully, resolving a virtual address into a physical address will take one TLB lookup and one addition, inside the processor

- TLB miss vs. page fault
  - penalty for a TLB miss is O(1) memory accesses
  - penalty for a page fault is trap into the kernel



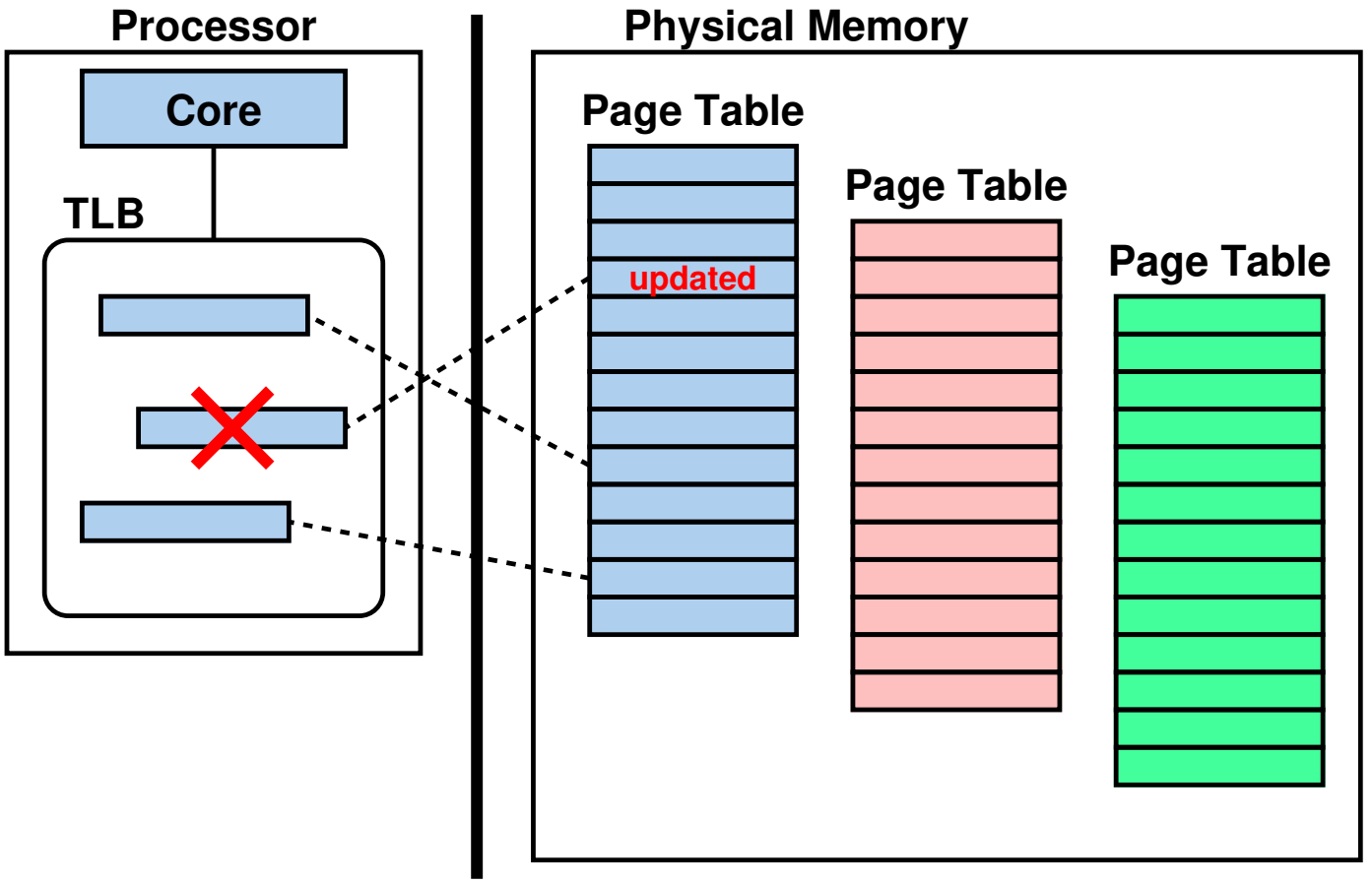
Translation Lookaside Buffers (TLB)

Conceptually:



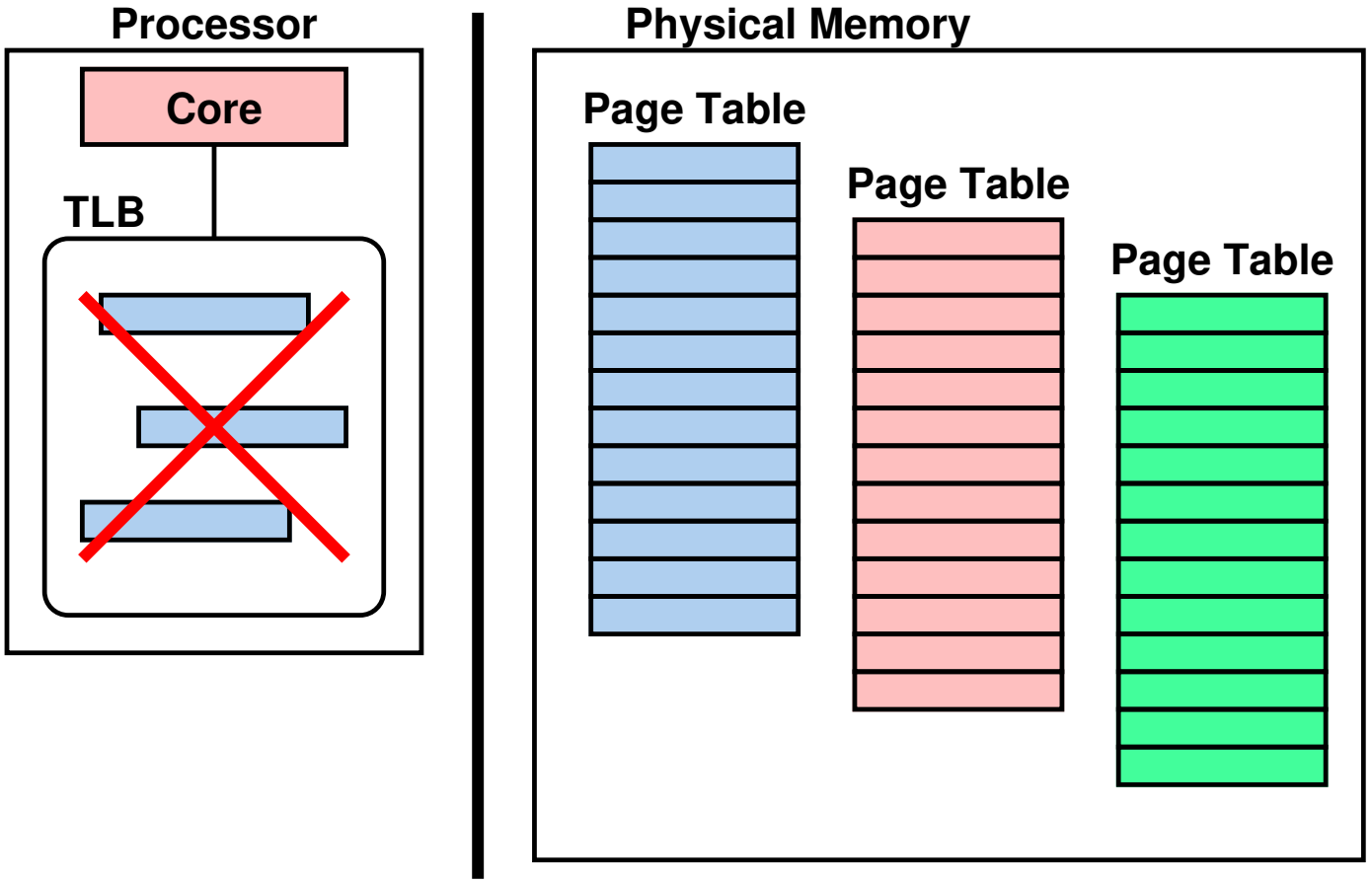
Translation Lookaside Buffers (TLB)

- When a page table entry is modified by the kernel, the OS must *flush* (invalidate) the corresponding TLB entry

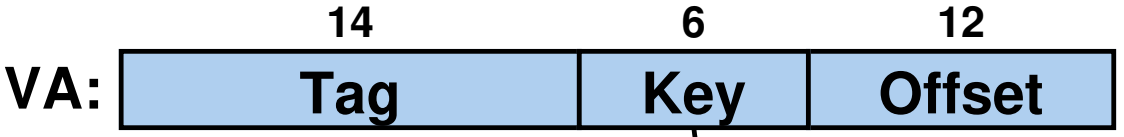


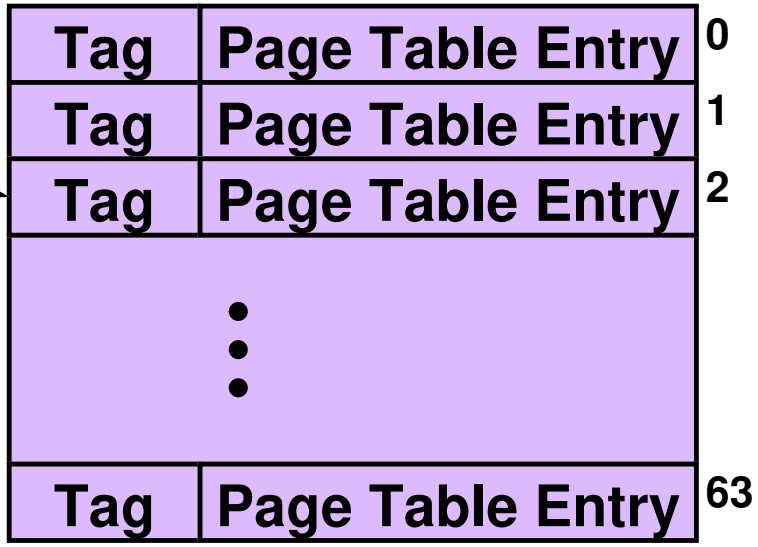
Translation Lookaside Buffers (TLB)

- When a page table entry is modified by the kernel, the OS must *flush* (invalidate) the corresponding TLB entry
- When switching to a different *process*, must *flush* the *entire TLB*
  - in x86, this can be achieved by setting the CR3 register

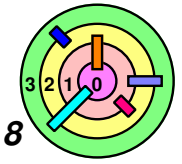


Translation Lookaside Buffers (TLB)

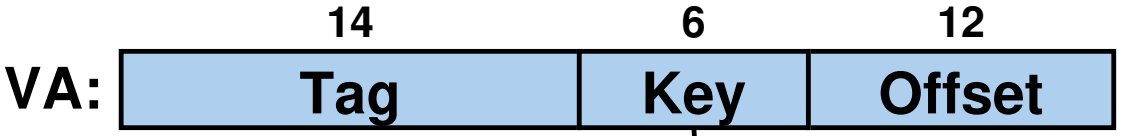


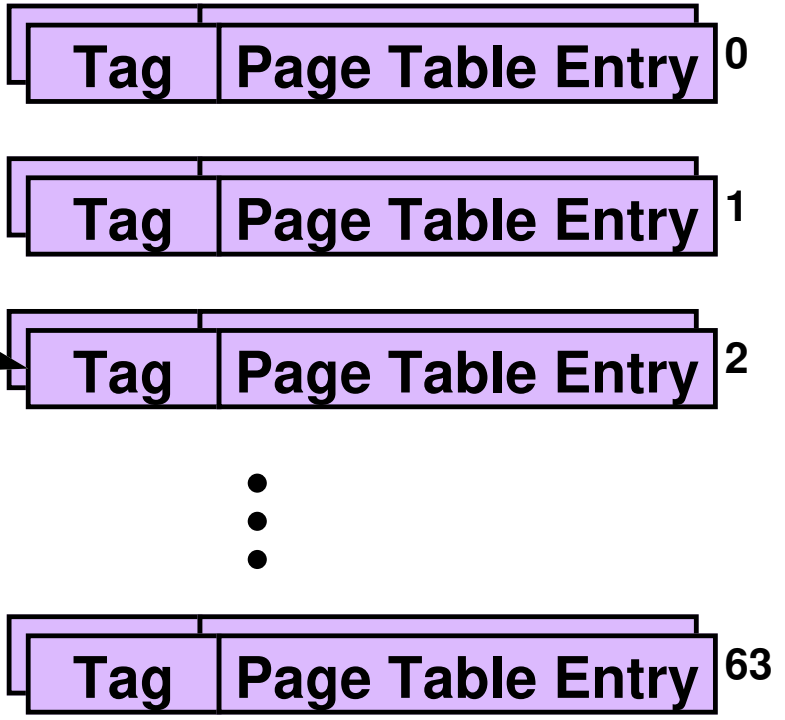


- TLB is often implemented as a *simple hash table in hardware*
- Ex: *direct mapping* cache (hardware cache) with 64 *(= 2^6)* lines
  - number of bits in *key* tells you how many *lines* the TLB has
    - the *key* is used as a *array index* to access the TLB
      - it tells you which line to look at
      - collision resolution chain of length 1 done in hardware
  - *TBL hit:* tag in virtual address == tag in a TLB line
  - *TBL miss:* tag in virtual address != tag in a TLB line
    - need to fetch PTE from memory and replace TBL line with new tag and PTE

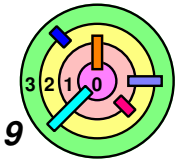


Translation Lookaside Buffers (TLB)

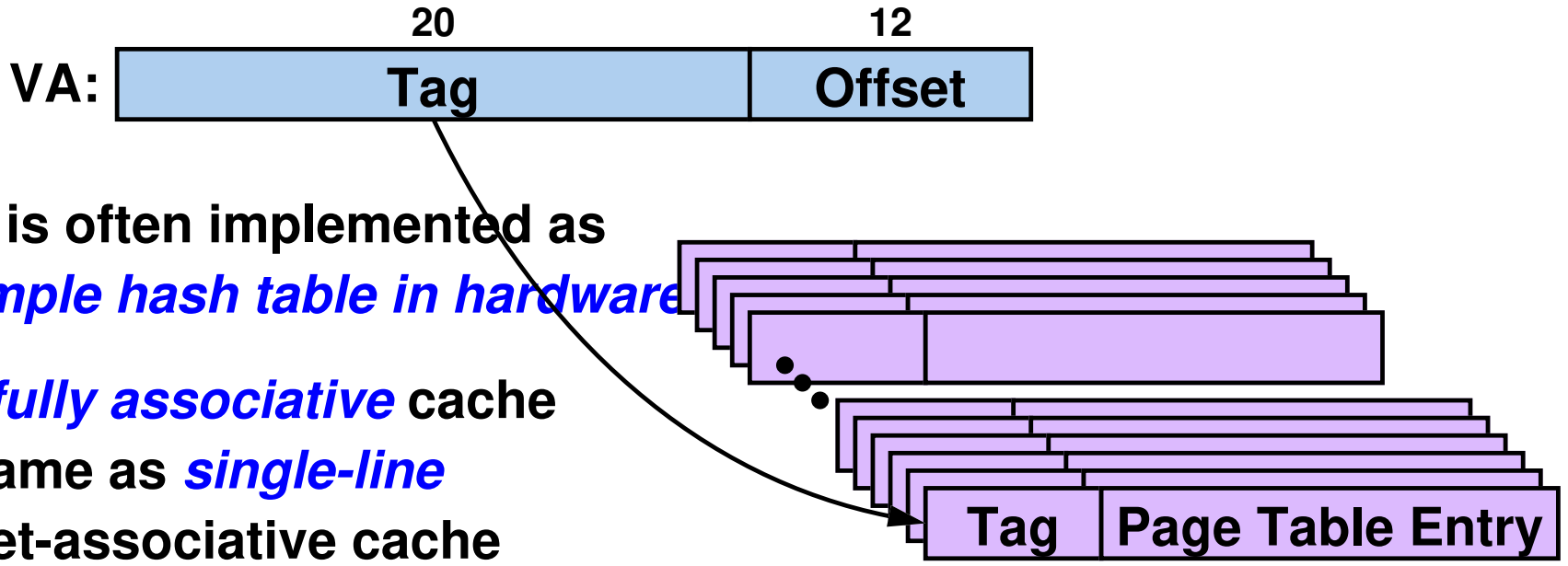




- TLB is often implemented as a *simple hash table in hardware*
- Ex: *two-way set-associative* cache (hardware cache) with 64 *(= 2^6)* lines
  - number of bits in *key* tells you how many *lines* the TLB has
    - the *key* is used as a *array index* to access the TLB
      - it tells you which line to look at
      - collision resolution chain of length 2 done in hardware
  - amount of *set-associativity* is the *"bucket size"* in each line if you view a line as a "bucket" in a hash table
  - the *"tag"* in the virtual address is compared against *all* tags in a line *simultaneously* (i.e., under the same key)

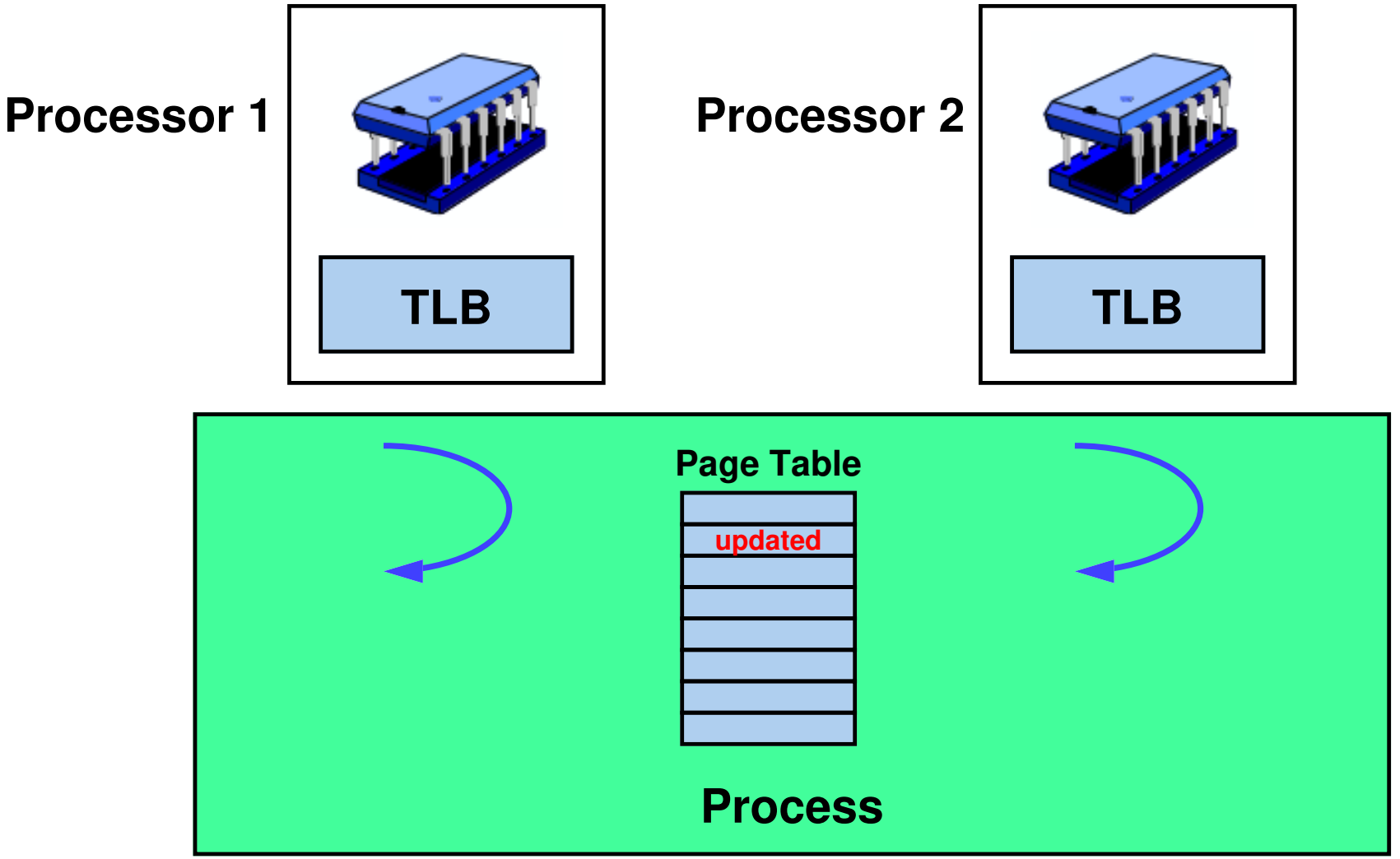


Translation Lookaside Buffers (TLB)



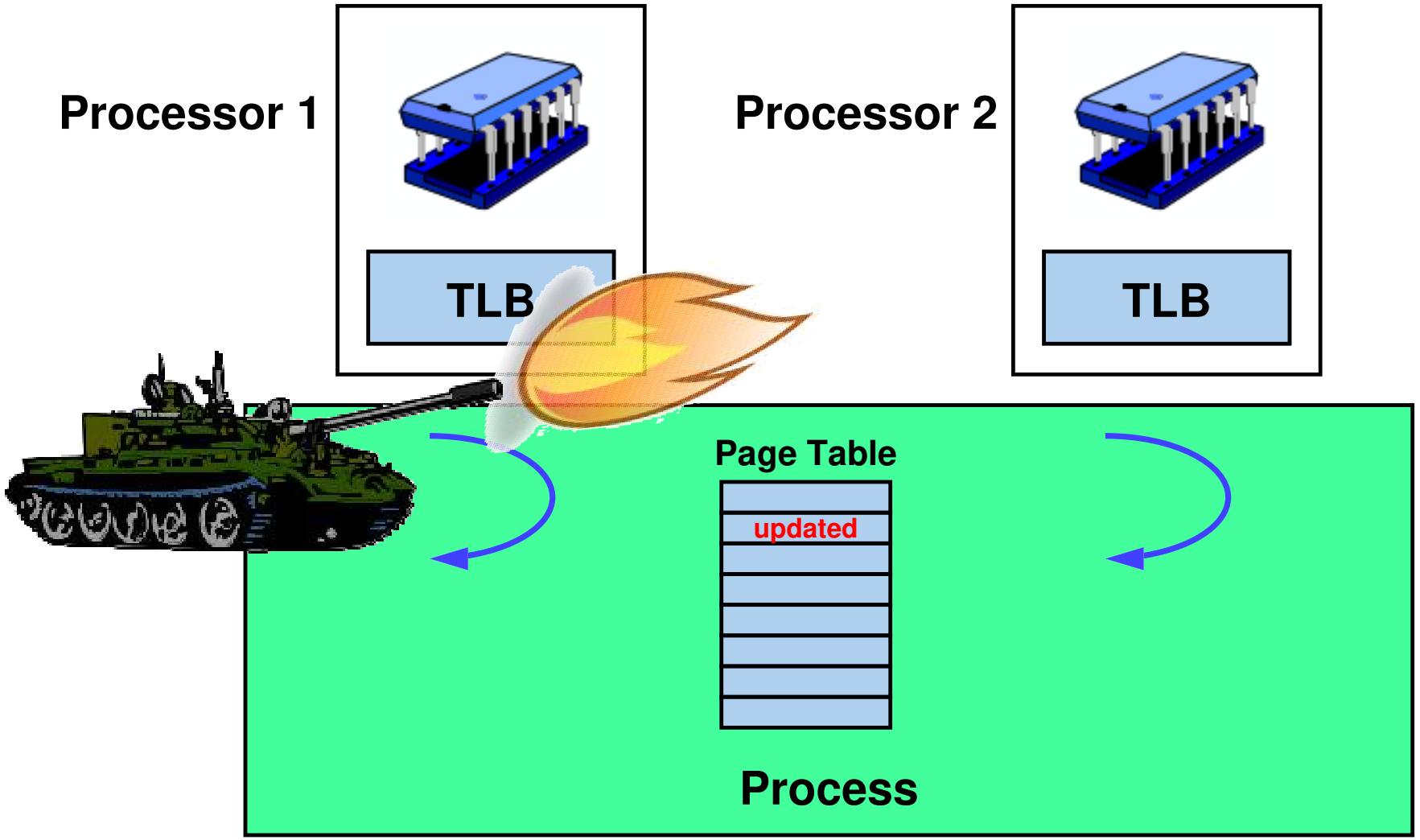
- TLB is often implemented as a *simple hash table in hardware*
- Ex: *fully associative* cache
  - same as *single-line* set-associative cache
  - expensive
    - need to compare with all the tags *in parallel*

TLBs and Multiprocessors



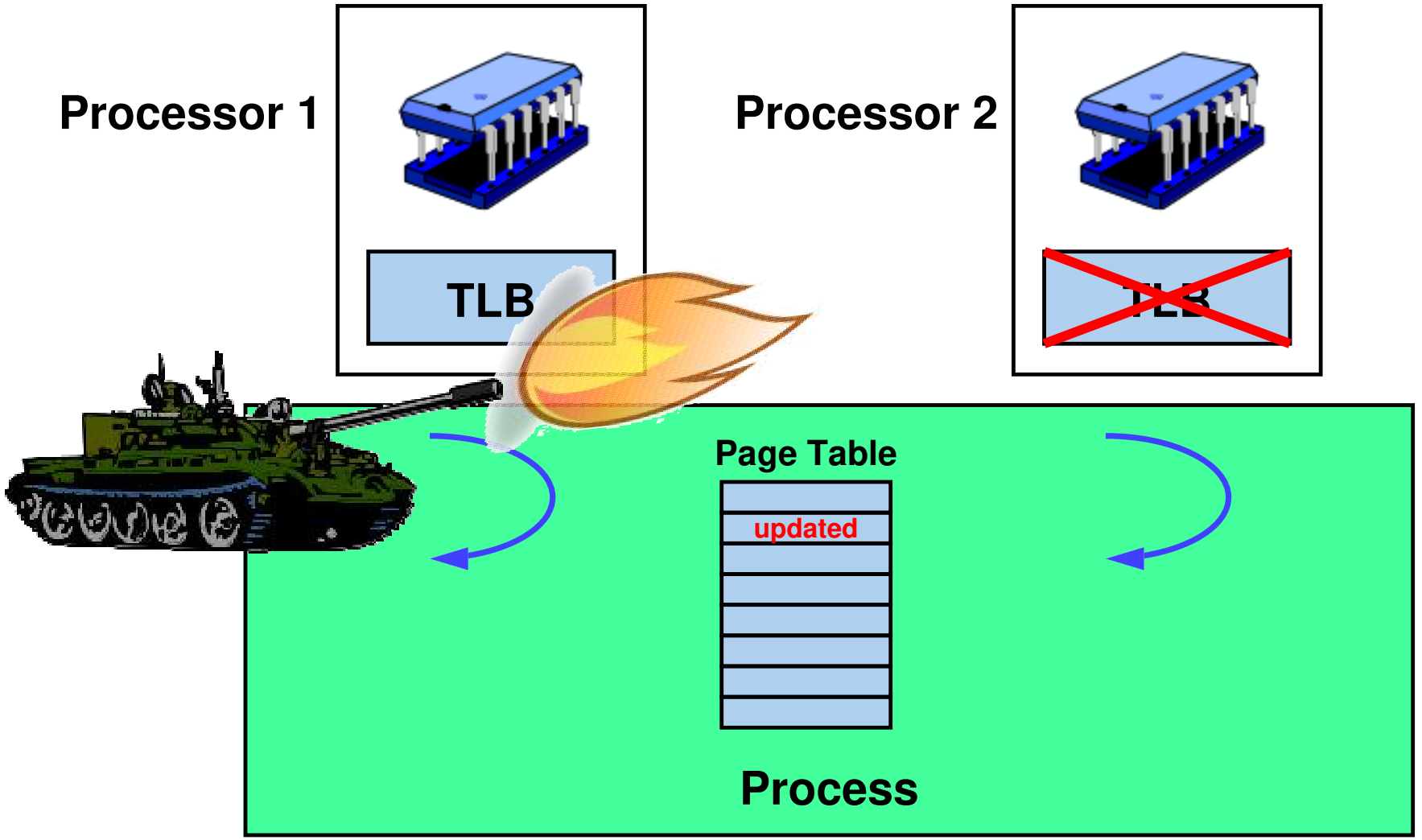
In a multiprocessors environment, one processor can modify a mapping cached in the TLB of another processor

TLBs and Multiprocessors



Before such a mapping is modified, Processor 1 must *shoot-down* (invalidate) the TLB of Processor 2

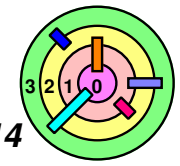
TLBs and Multiprocessors



- Need to get the timing right

TLB Shootdown Algorithm (In Hardware / HAL)

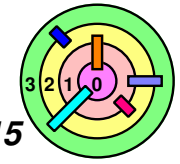
```
// shooter code
for all processors i sharing address space
   interrupt(i);
for all processors i sharing address space
   while (noted[i] == 0)
      ;
modify_page_table();
update_or_flush_tlb();
done[me] = 1;


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



Storage / Cache Hierarchy in Today's Systems

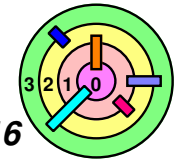
Storage	Access Time	Size	
TLB	1 ns	64 KB	
L2	4 ns	256 KB	inside CPU
L3	10 ns	2 MB	outside CPU
RAM	100 ns	10 GB	
SSD	100 μs	100 GB	need device driver
Remote RAM	100 μs	100 GB	
Disk	1 - 10 ms	1 TB	
Remote Disk	100 ms	1 XB	
...	...	...	

- There can even be something before TLB
  - "Virtually Addressed Cache"
- L2 and L3 are "Physically Addressed Caches"
- TLB, L2, L3 are managed in hardware
  - OS can only invalidate entries in a hardware cache

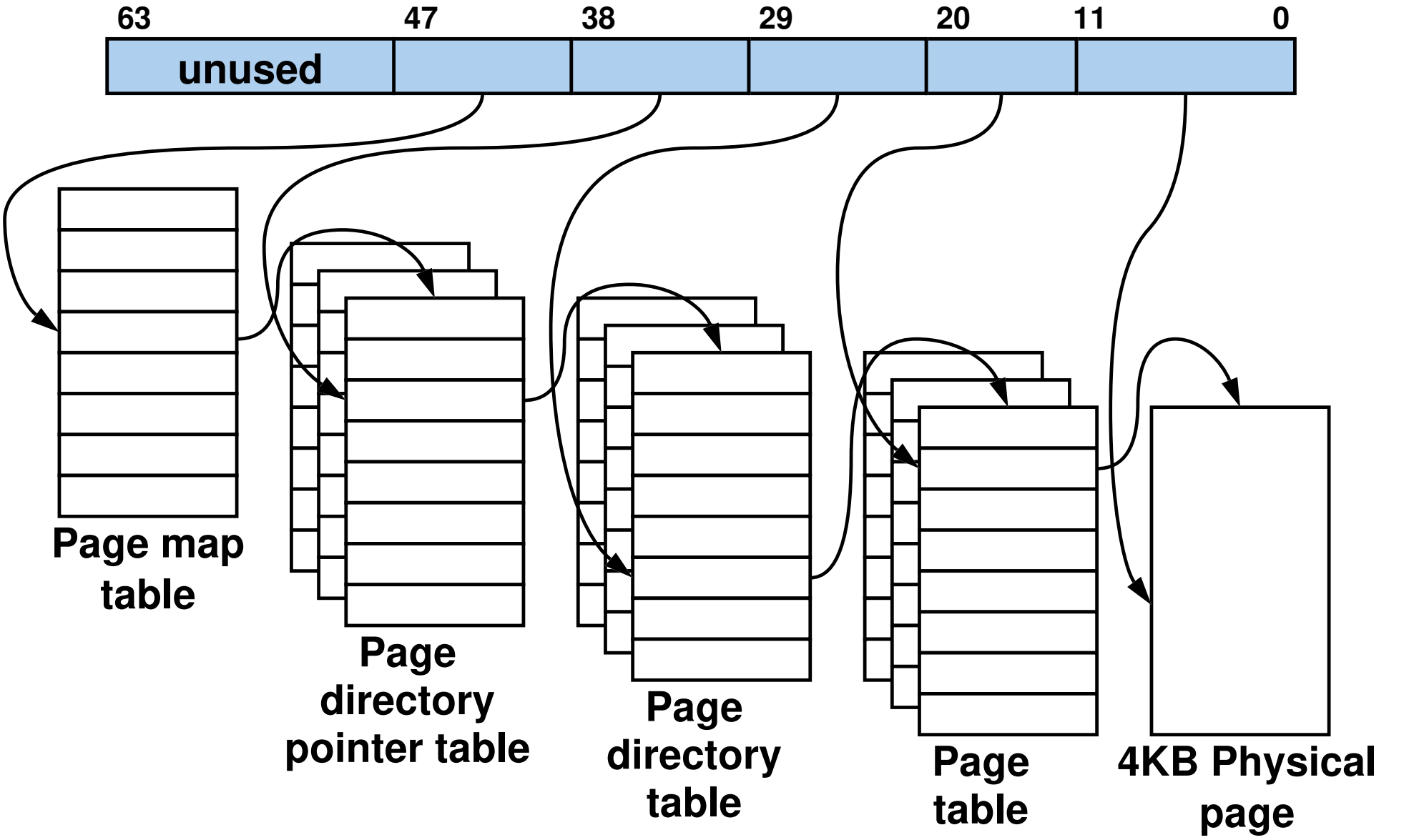


7.2 Hardware Support for Virtual Memory

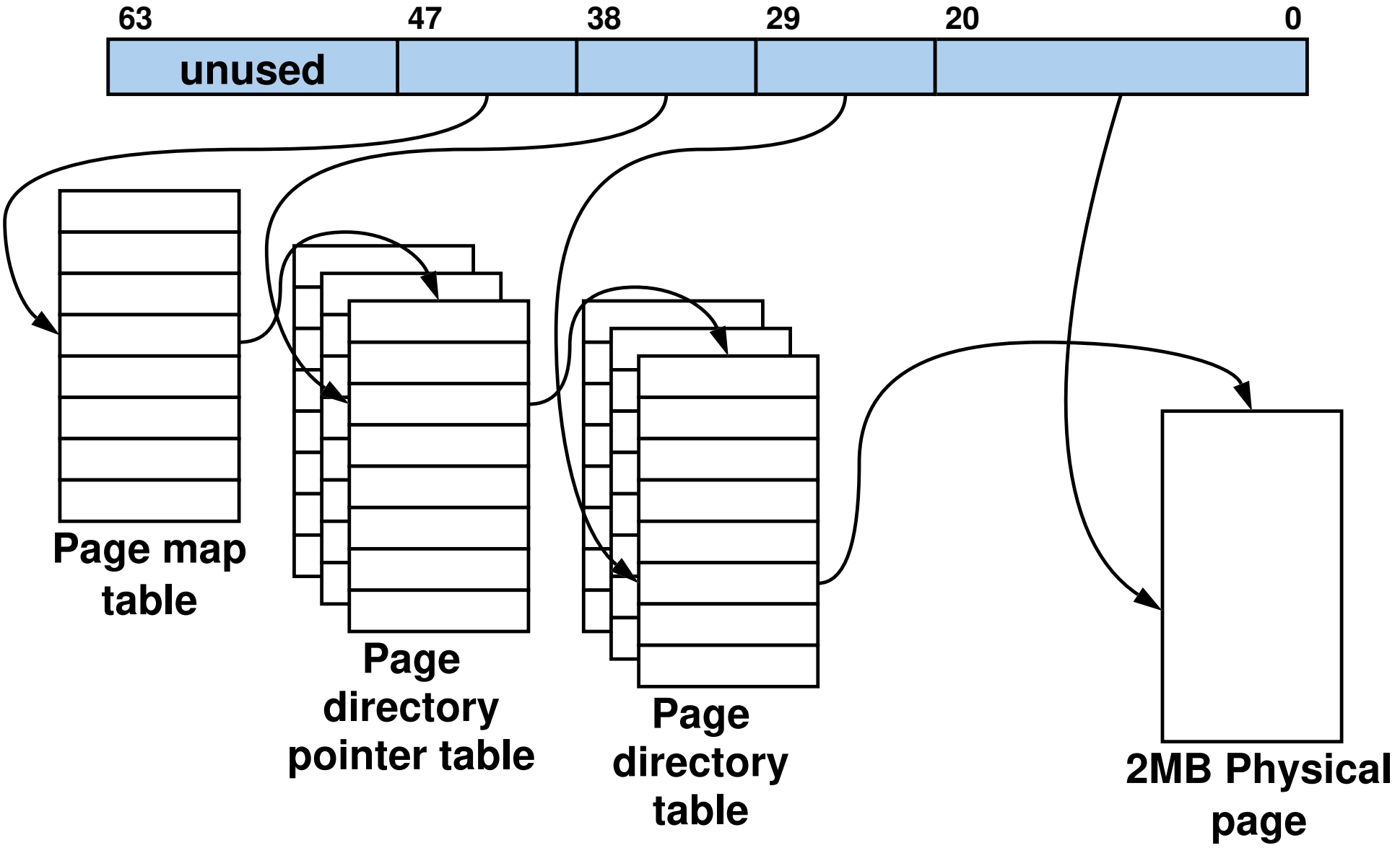
- Forward-Mapped Page Tables
- Linear Page Tables
- Hashes Page Tables
- Translation Lookaside Buffers
- *64-Bit Issues*
- Virtualization



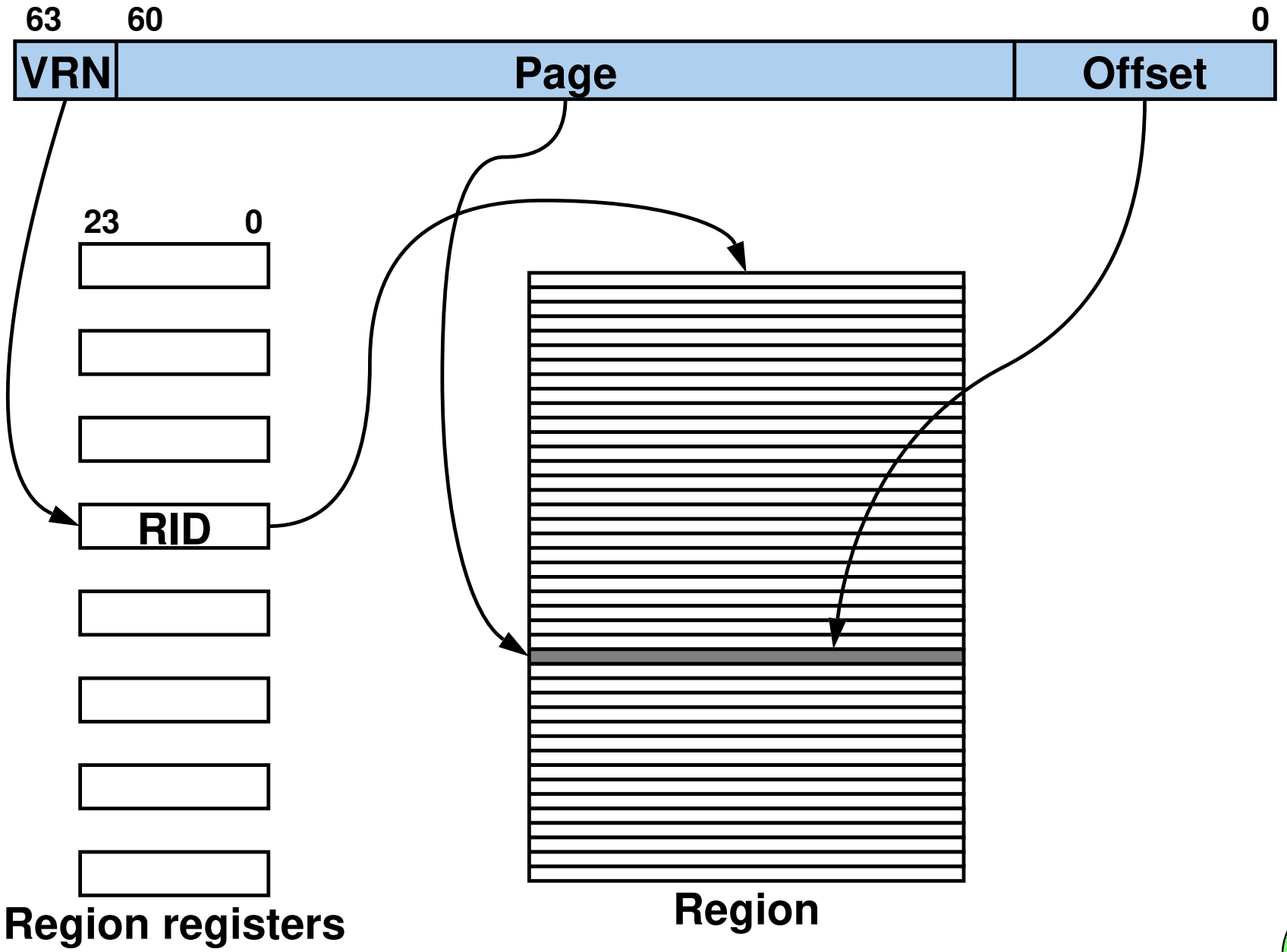
x86-64 (AMD) Virtual Address Format 1



x86-64 (AMD) Virtual Address Format 2

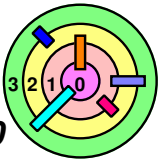


Intel IA-64

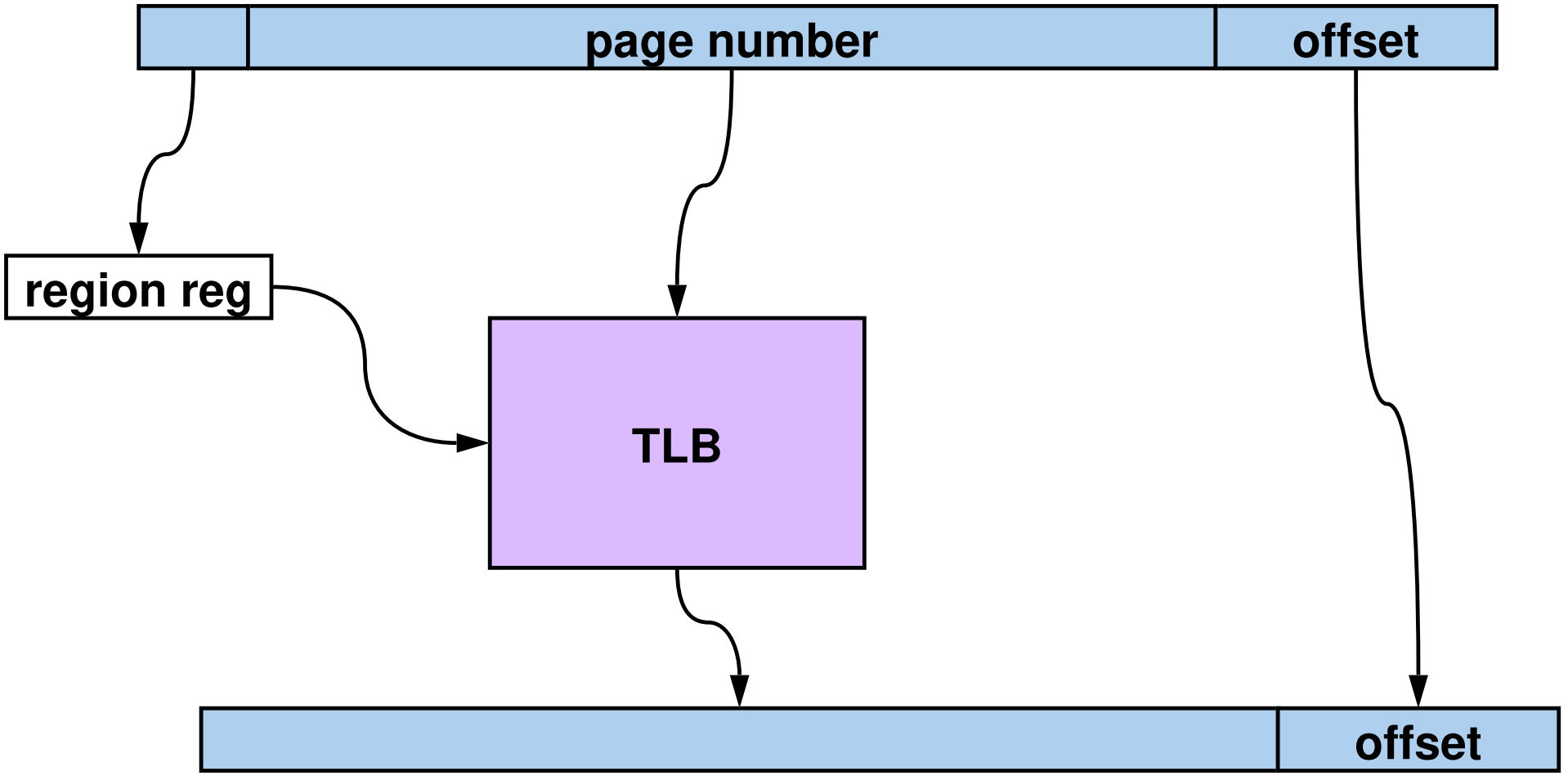


IA-64 Address Translation

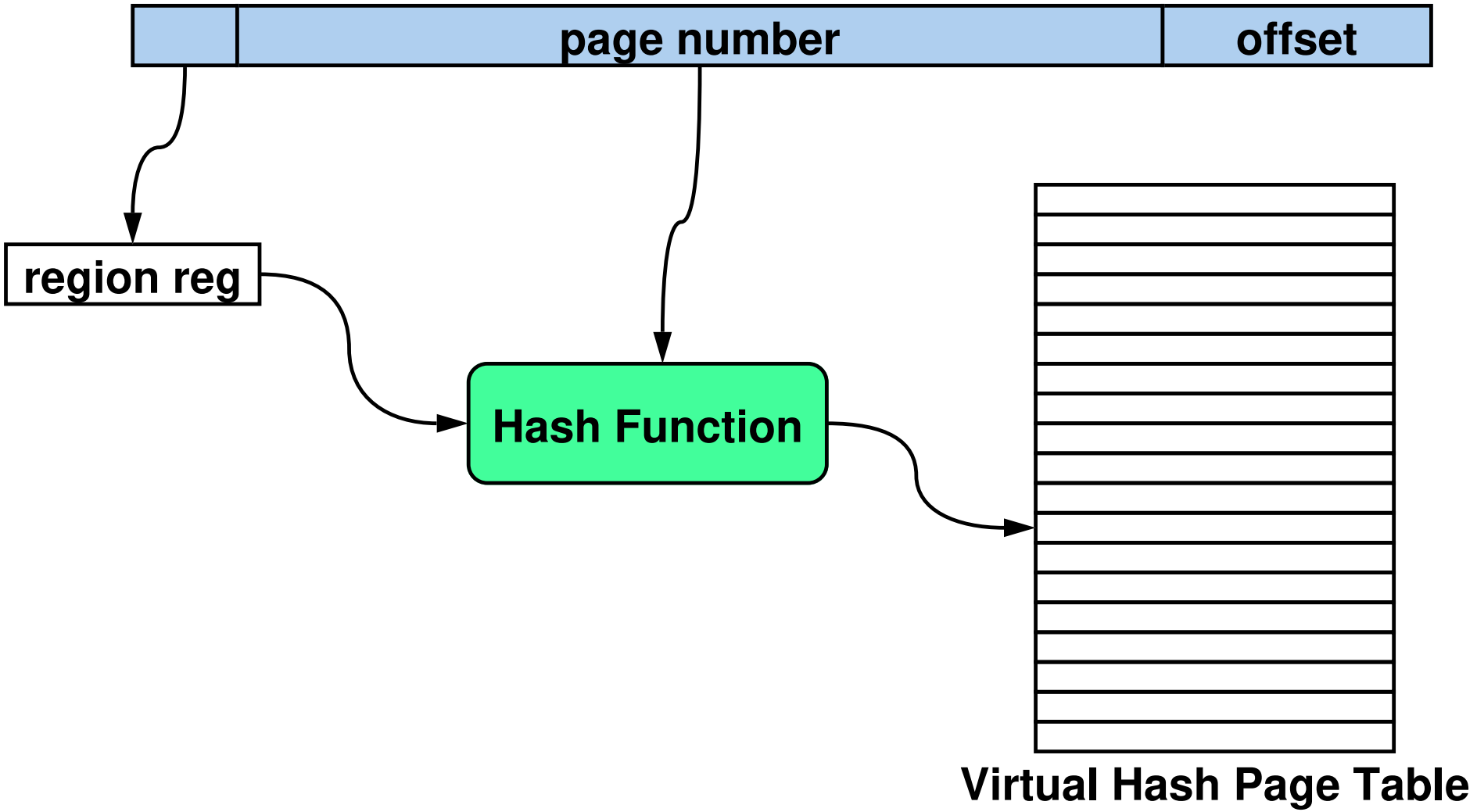
- TLB
  - software-managed
- Virtual Hash Page Table (VHPT)
  - per-region linear page table
  - single large hashed page table



Translation: TLB



Translation: TLB Miss



7.2 Hardware Support for Virtual Memory

- Forward-Mapped Page Tables
- Linear Page Tables
- Hashes Page Tables
- Translation Lookaside Buffers
- 64-Bit Issues
- *Virtualization*
  - will come back here after we have covered *Virtual Machines*