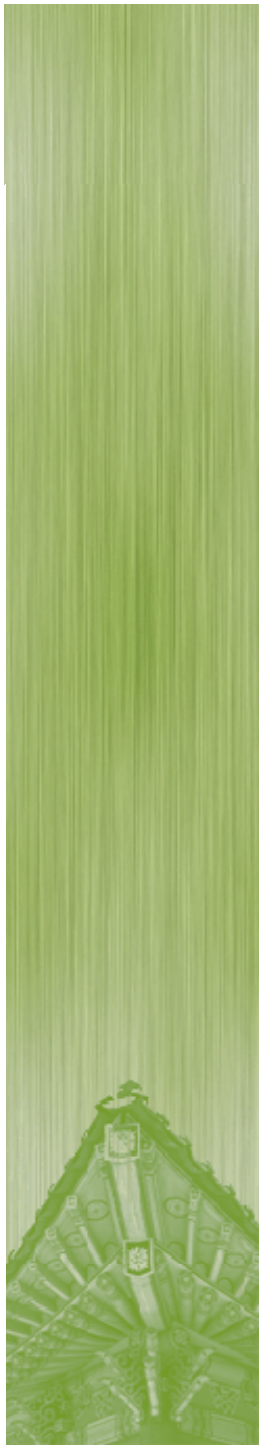


Virtual Memory

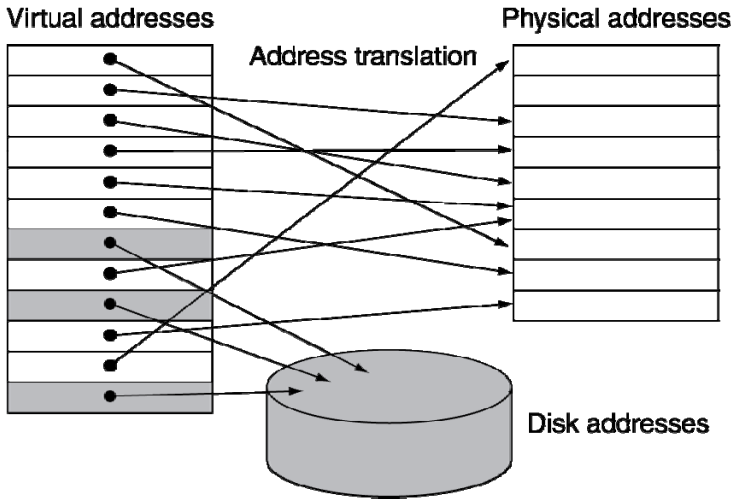
Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

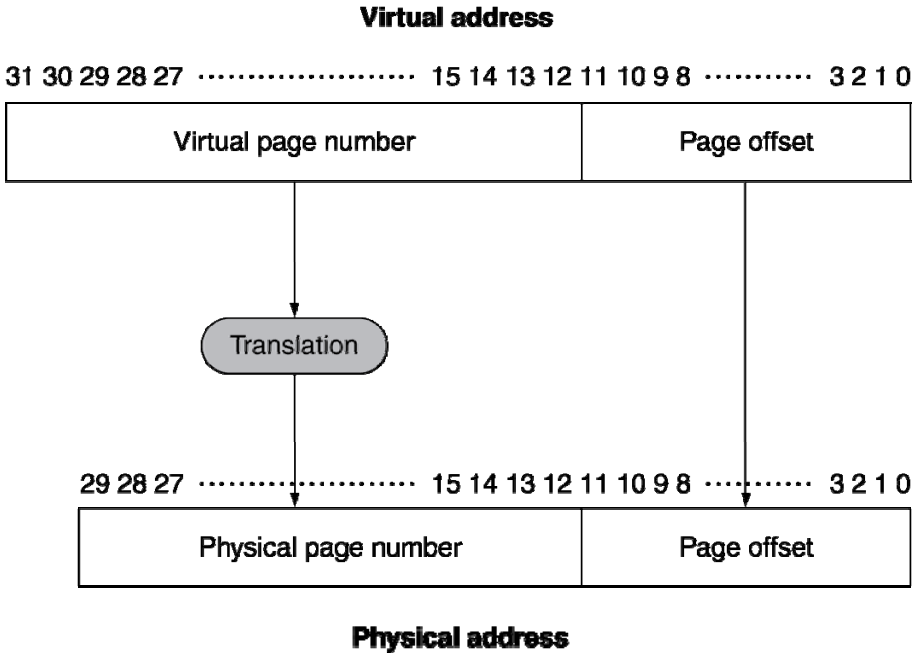
Virtual Memory

- **Use main memory as a "cache" for secondary (disk) storage**
  - Managed jointly by CPU hardware and the OS
- **Programs share main memory**
  - Each gets a private virtual address space holding its frequently used code and data
  - Protected from other programs
- **CPU and OS translate virtual addresses to physical addresses**
  - VM "block" is called a page
  - VM translation "miss" is called a page fault

Address Translation

- Fixed-size pages (e.g., 4KB)





Page Fault Penalty

- **On page fault, the page must be fetched from disk**
  - Takes millions of clock cycles
  - Handled by OS code

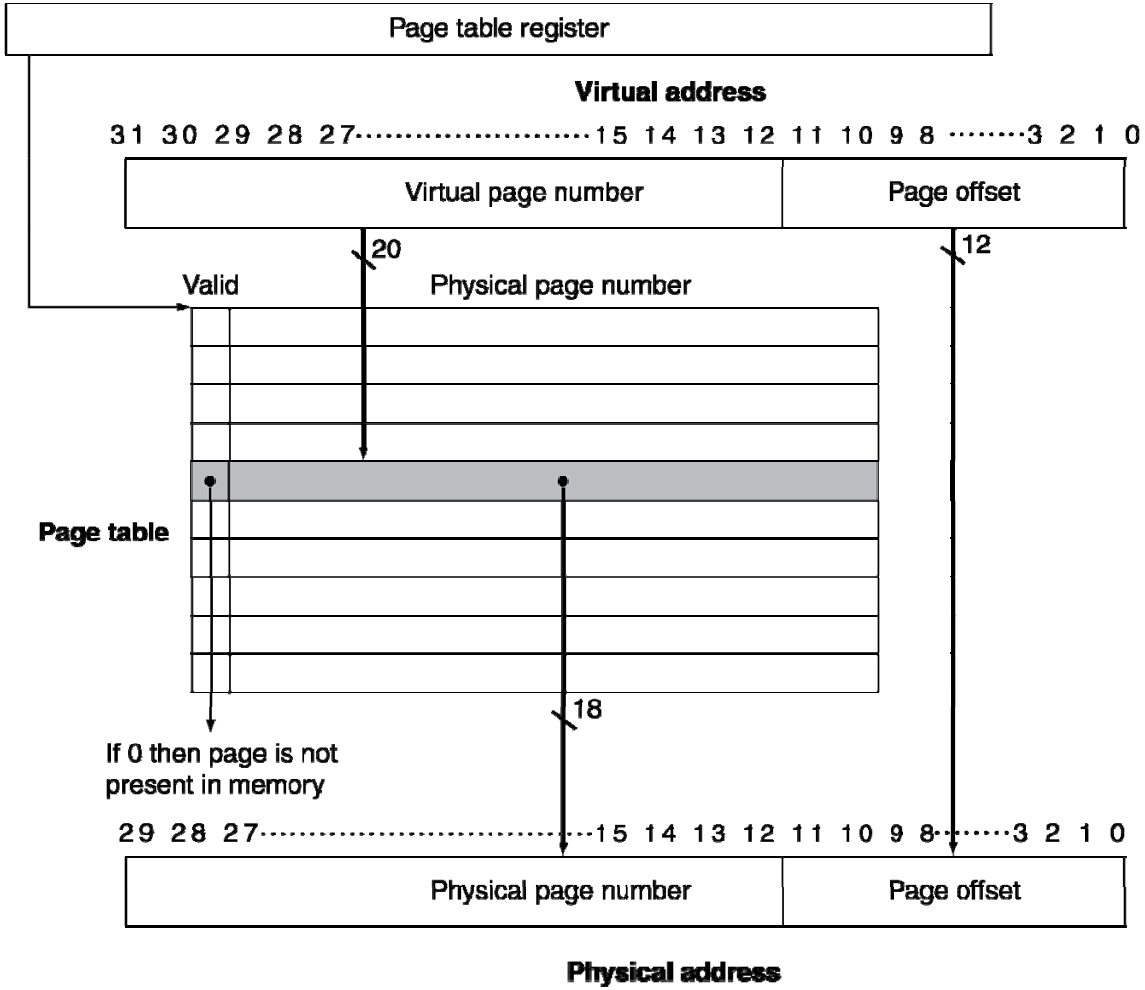
- **Try to minimize page fault rate**
  - Fully associative placement
  - Smart replacement algorithms

Page Tables (1)

- **Stores placement information**
  - Array of page table entries, indexed by virtual page number
  - page table register in CPU points to page table in physical memory
- **If page is present in memory**
  - PTE stores the physical page number
  - Plus other status bits (referenced, dirty, …)
- **If page is not present**
  - PTE can refer to location in swap space on disk

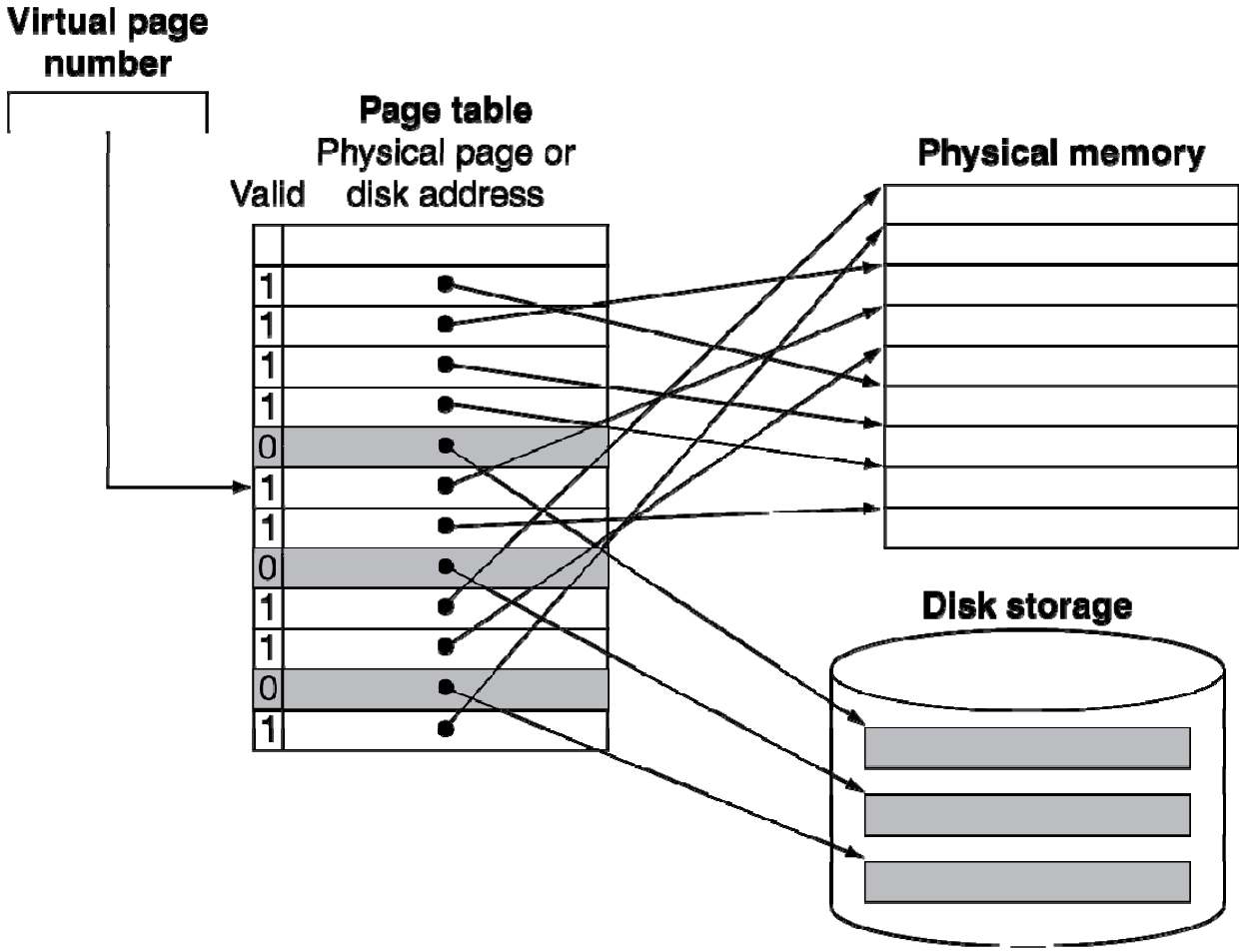
Page Tables (2)

- Translation using a page table



Page Tables (3)

- Mapping pages to storage



Replacement and Writes

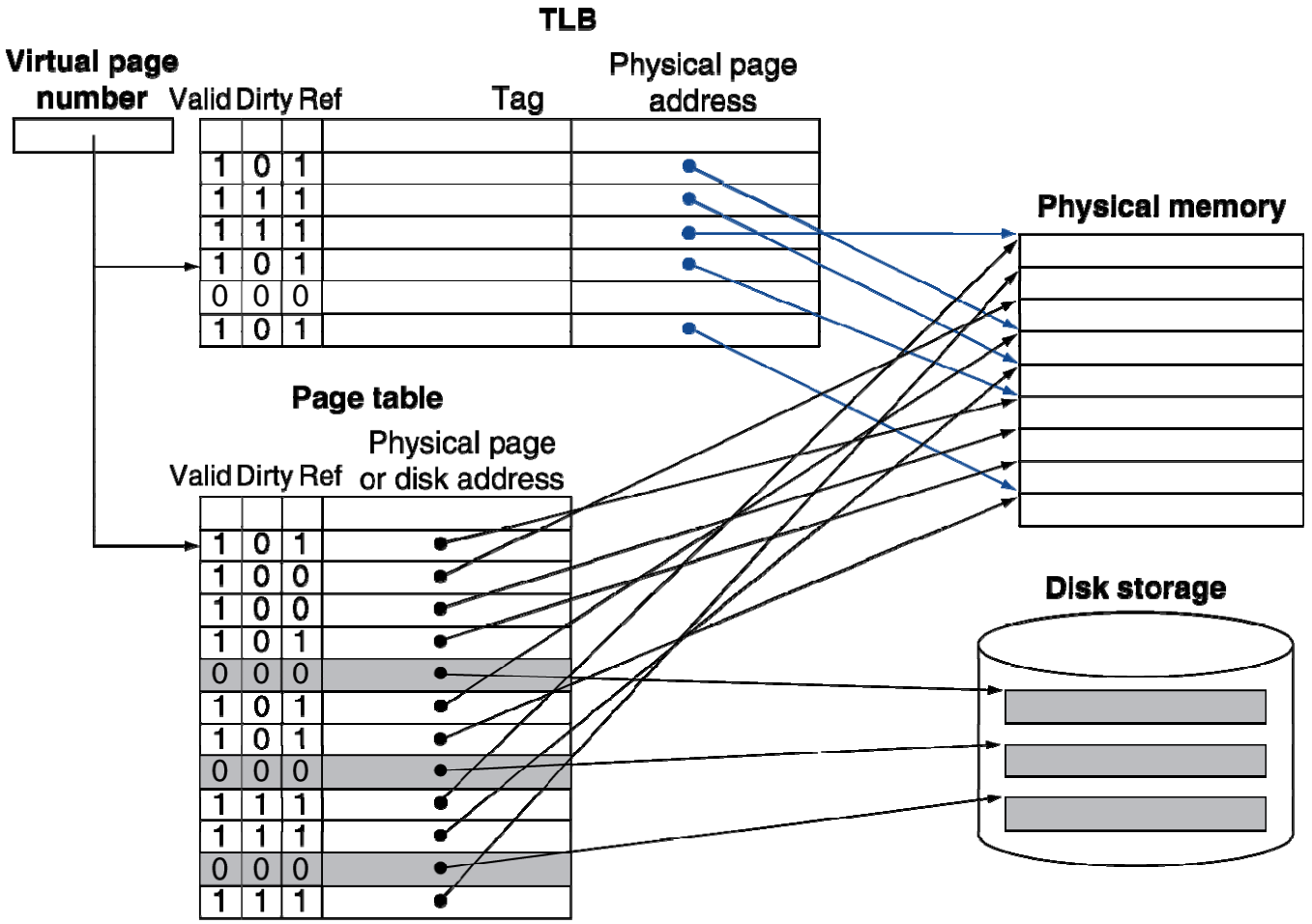
- **To reduce page fault rate, prefer least-recently used (LRU) replacement**
  - Reference bit (aka use bit) in PTE set to 1 on access to page
  - Periodically cleared to 0 by OS
  - A page with reference bit = 0 has not been used recently
- **Disk writes take millions of cycles**
  - Block at once, not individual locations
  - Use write-back: write through is impractical
  - Dirty bit in PTE set when page is written

TLB (1)

- **Address translation would appear to require extra memory references**
  - One to access the PTE
  - Then the actual memory access
- **But access to page tables has good locality**
  - So use a fast cache of PTEs within the CPU
  - Called a Translation Look-aside Buffer (TLB)
  - Typical: 16-512 PTEs, 0.5-1 cycle for hit, 10-100 cycles for miss, 0.01%-1% miss rate
  - Misses could be handled by hardware or software

TLB (2)

- Fast translation using a TLB



TLB (3)

- **TLB misses: if page is in memory**
  - Load the PTE from memory and retry
  - Could be handled in hardware
    - Can get complex for more complicated page table structures
  - Or in software
    - Raise a special exception, with optimized handler
- **TLB misses: if page is not in memory (page fault)**
  - OS handles fetching the page and updating the page table
  - Then restart the faulting instruction

TLB (4)

- **TLB miss handler**
  - TLB miss indicates
    - Page present, but PTE not in TLB
    - Page not present
  - Must recognize TLB miss before destination register overwritten
    - Raise exception
  - Handler copies PTE from memory to TLB
    - Then restarts instruction
    - If page not present, page fault will occur

Page Fault Handler

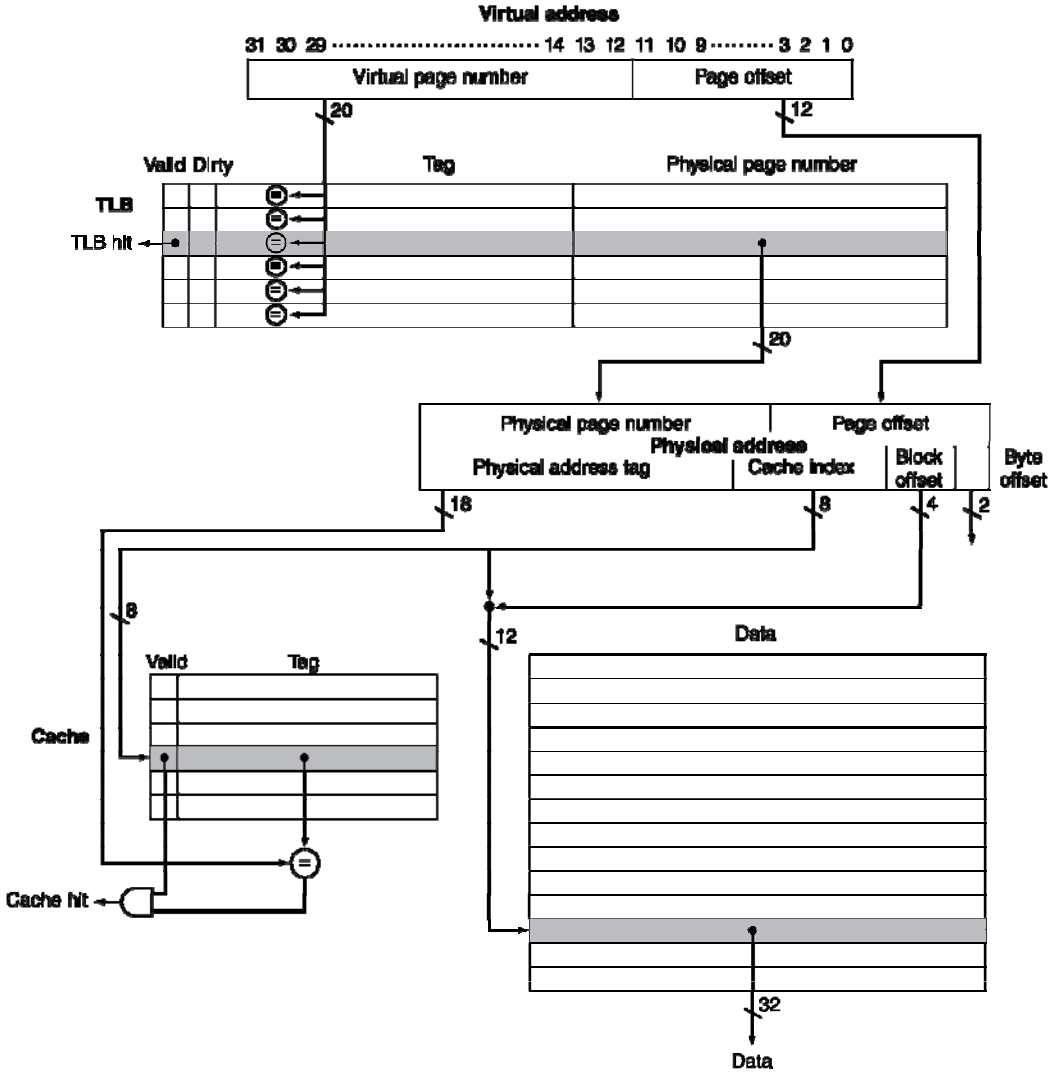
- **Handling page faults**
  - Use faulting virtual address to find PTE
  - Locate page on disk
  - Choose page to replace
    - If dirty, write to disk first
  - Read page into memory and update page table
  - Make process runnable again
    - Restart from faulting instruction

Memory Protection

- **Different tasks can share parts of their virtual address spaces**
  - But need to protect against errant access
  - Requires OS assistance
- **Hardware support for OS protection**
  - Privileged supervisor mode (aka kernel mode)
  - Privileged instructions
  - Page tables and other state information only accessible in supervisor mode
  - System call exception (e.g., syscall in MIPS)

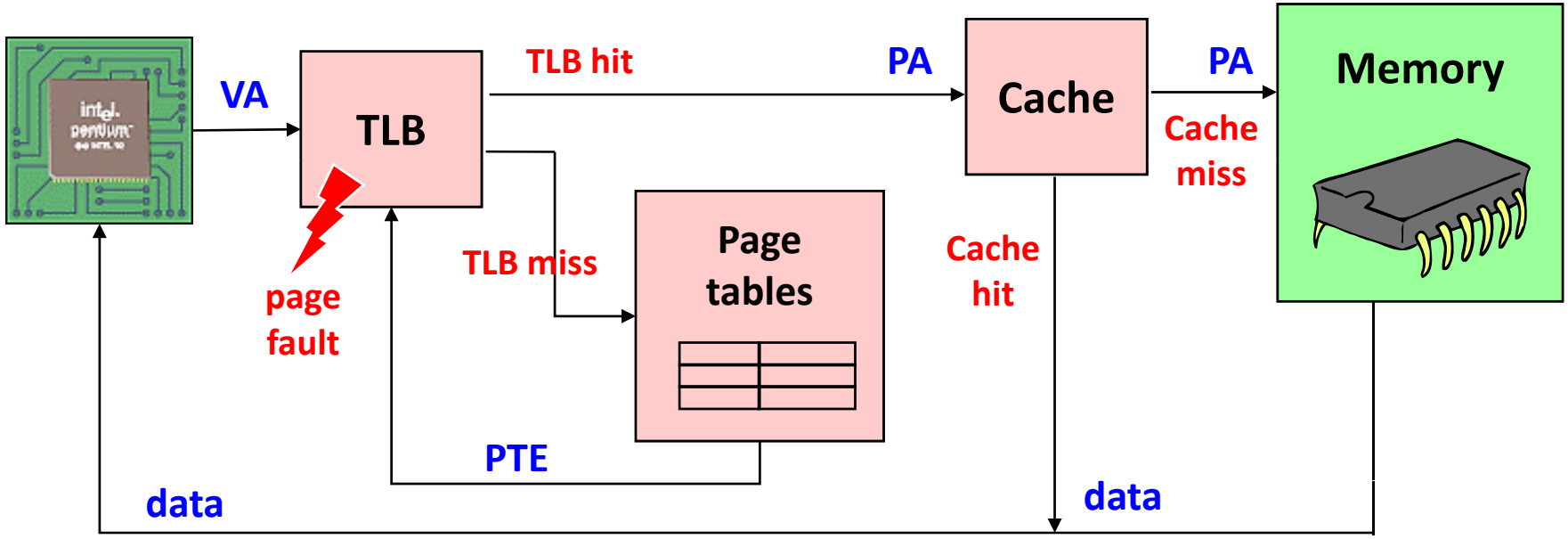
TLB & Cache Interaction

- **If cache tag uses physical address**
  - Need to translate before cache lookup
- **Alternative: use virtual address tag**
  - Complications due to aliasing
  - Different virtual addresses for shared physical address



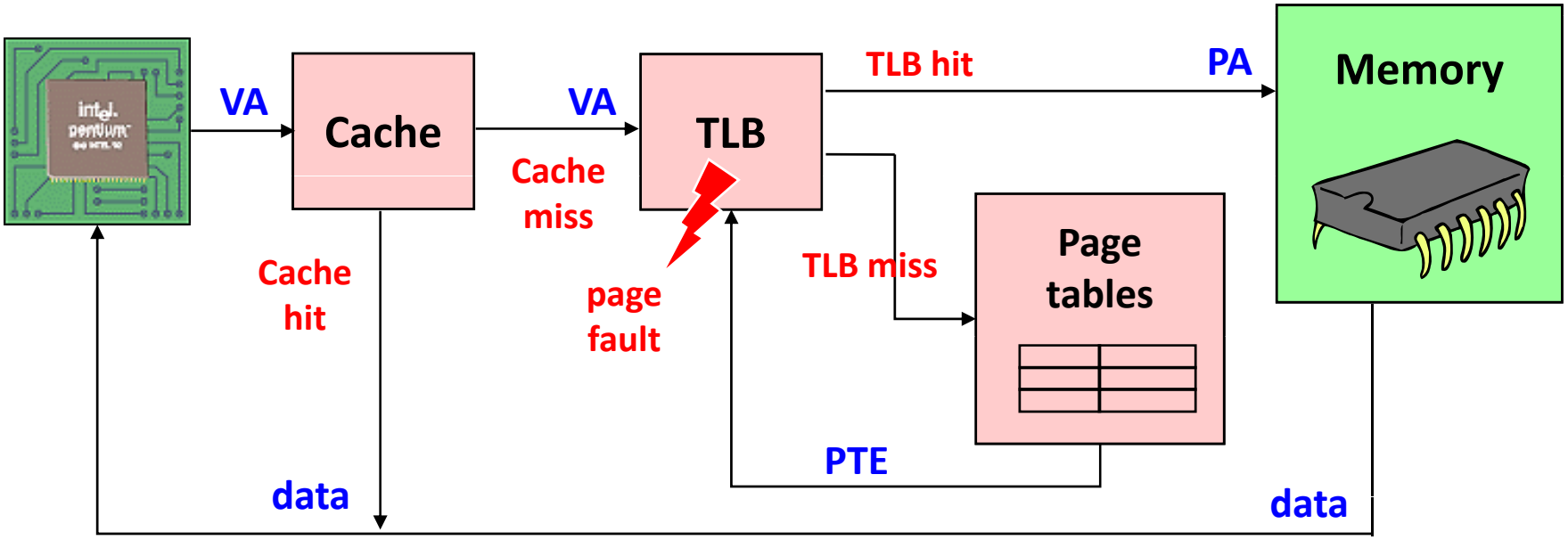
Integrating VM and Cache (1)

- **Physically addressed caches**
  - Allows multiple processes to have blocks in cache at the same time.
  - Allows multiple processes to share pages.
  - Address translation is on the critical path.



Integrating VM and Cache (2)

- **Virtually addressed, virtually tagged caches**
  - Homonym problem:
    - Each process has a different translation of the same virtual address.
  - Address synonyms or aliases problem.
    - Two different virtual addresses point to the same physical address.



Integrating VM and Cache (3)

- **Virtually addressed, physically tagged caches**
  - Use virtual address to parallel access to the TLB and cache.
  - TLB produces the PFN – which must match the physical tag of the accessed cache line for it to be a "hit".