Summary: caches so far

- introduction to memory hierarchy
  - locality principles
  - levels, blocks, hit, miss, miss penalty
  - direct-mapped cache, handling misses
- cache features and performance
  - review direct-mapped single- and multi-word cache
  - cache performance, read/write stall cycles
  - multi-level cache hierarchy
- associative caches
  - flexible block placement schemes
  - overview of set associative caches
  - block replacement strategies
  - associative cache implementation

Virtual memory and TLB

(3rd Ed: p.511-594, 4th Ed: p.492-517)

- virtual memory: overview
- page table
- page faults
- TLB: accelerating address translation
- TLB, page table and cache
- memory read and write

Virtual memory: motivation

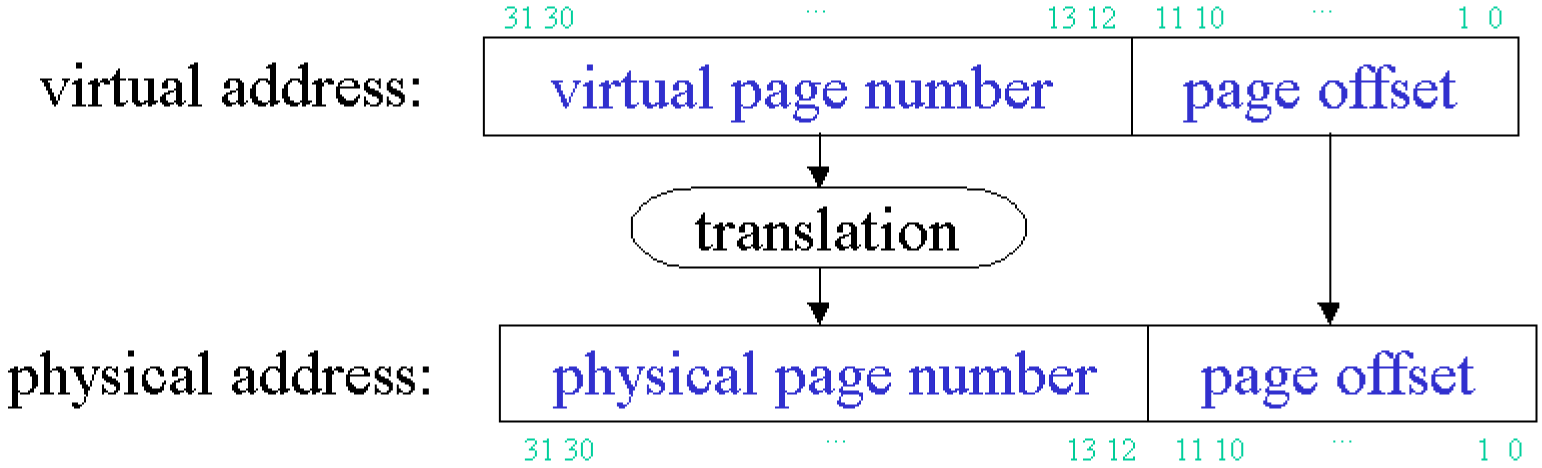
- use main memory as ‘cache’ for secondary memory, e.g. disk (10^2 cheaper, 10^5 slower than DRAM)
- to allow
  - sharing memory among multiple programs
  - running programs too large for physical memory
  - automatic memory management
  - anything else?
- problem:
  - can only manage sharing of physical memory (main memory) at run time
  - mapping and replacement strategies

Virtual memory: introduction

- compile each program using virtual address space
- translate **virtual address** into **physical address** or disk address, isolated from other processes
- terms
  - **page**: virtual memory block, usually fixed size
  - **page fault**: virtual memory miss
  - **relocation**: virtual address can be mapped to any physical address
  - **address translation**: map virtual address to physical or disk address

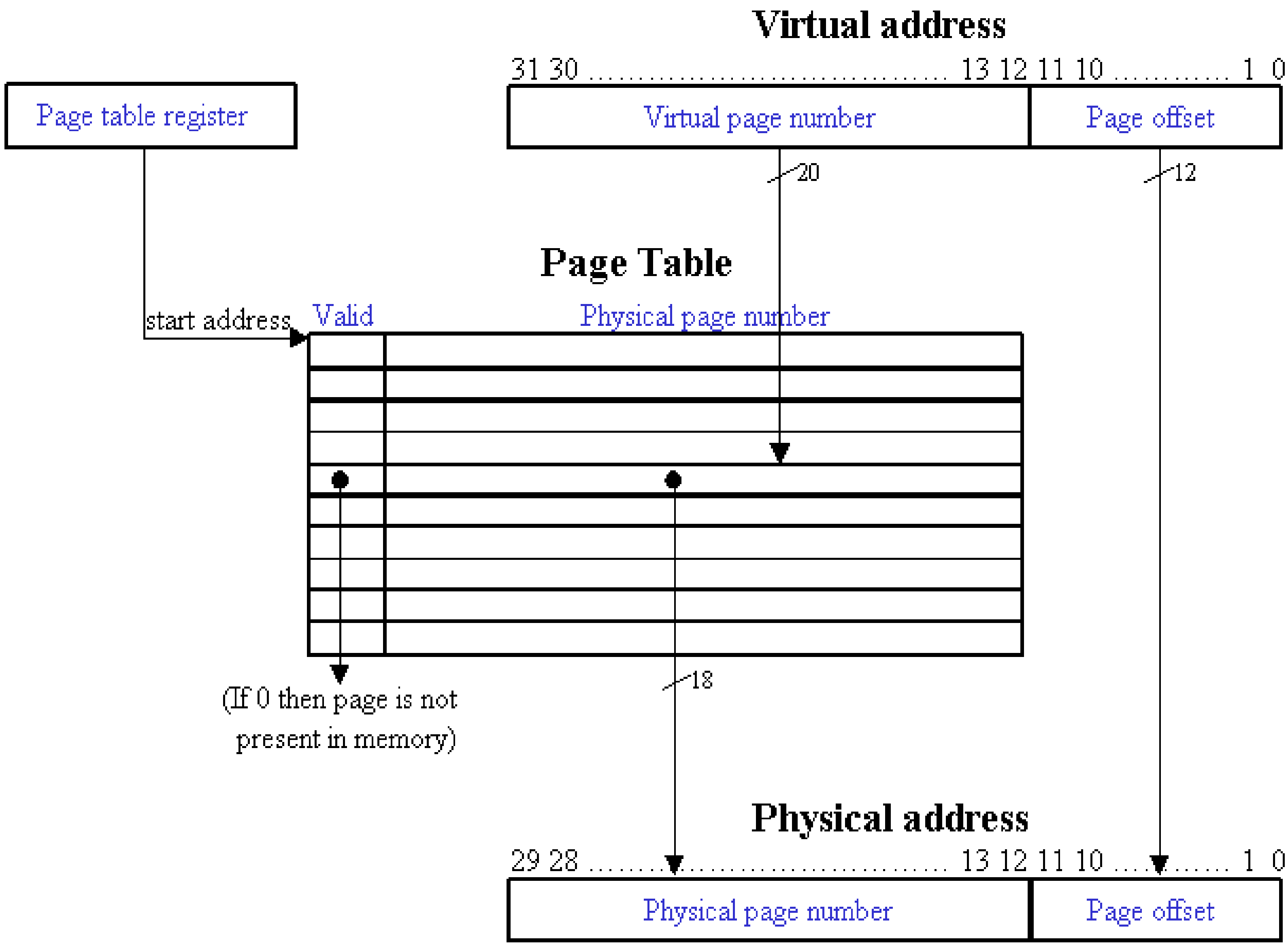
Virtual address

- virtual address = virtual page number + page offset

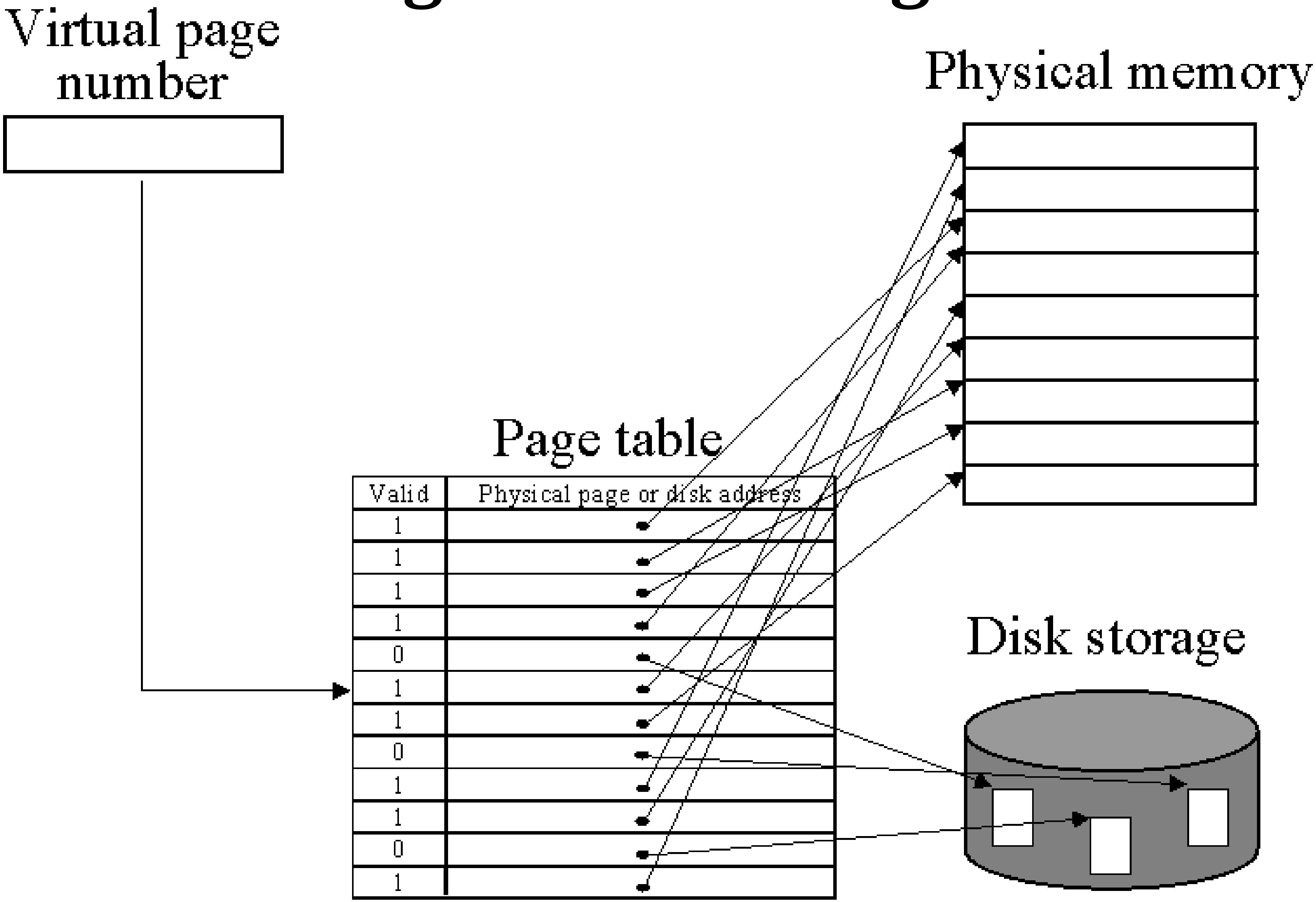


- number of bits in offset field determines page size
- high cost of page miss
  - DRAM 10^2 ns, disk 10^7 ns access time
  - allow data anywhere in DRAM, locate by page table (also example of associative placement)

Page table (for each process)



Page table arrangement



Design considerations

- large page size to amortise long access time
- flexible placement of pages to reduce page faults
- clever miss-handling algorithms to reduce miss rate (page replacement policy)
- use write-back
  - accumulate writes to a page
  - copy back during replacement
  - use dirty bit in page table to indicate if writes occurred while in memory

Page faults

- happen when valid bit in page table is off
- use exceptions to cause operating system to:
  1. locate required page in disk (use page table)
  2. place it somewhere in main memory (page replacement)
- Least Recently Used (LRU) page replacement policy
  - temporal locality: replace the page least recently used
    - e.g. most recent page references: 1, 2, 3, 4, 5, 1
      should replace page 2, then page 3, 4, 5, 1, ...