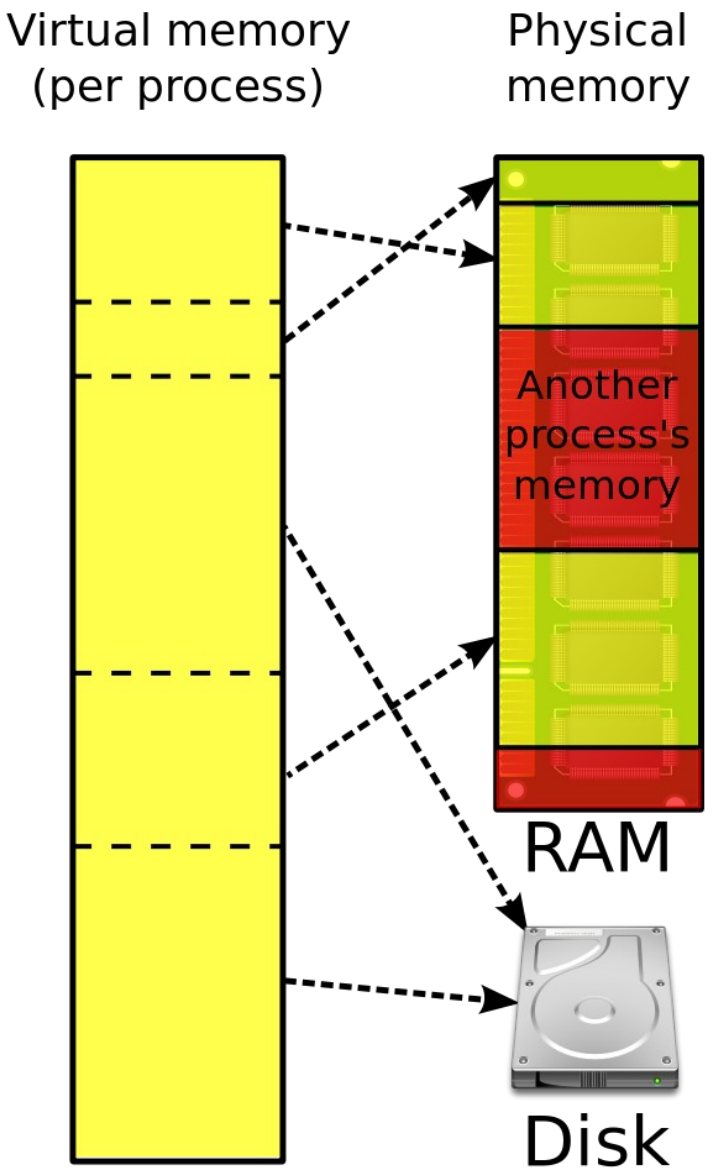
Analysis and modeling of

Computational Performance

Memory accesses – cache memory

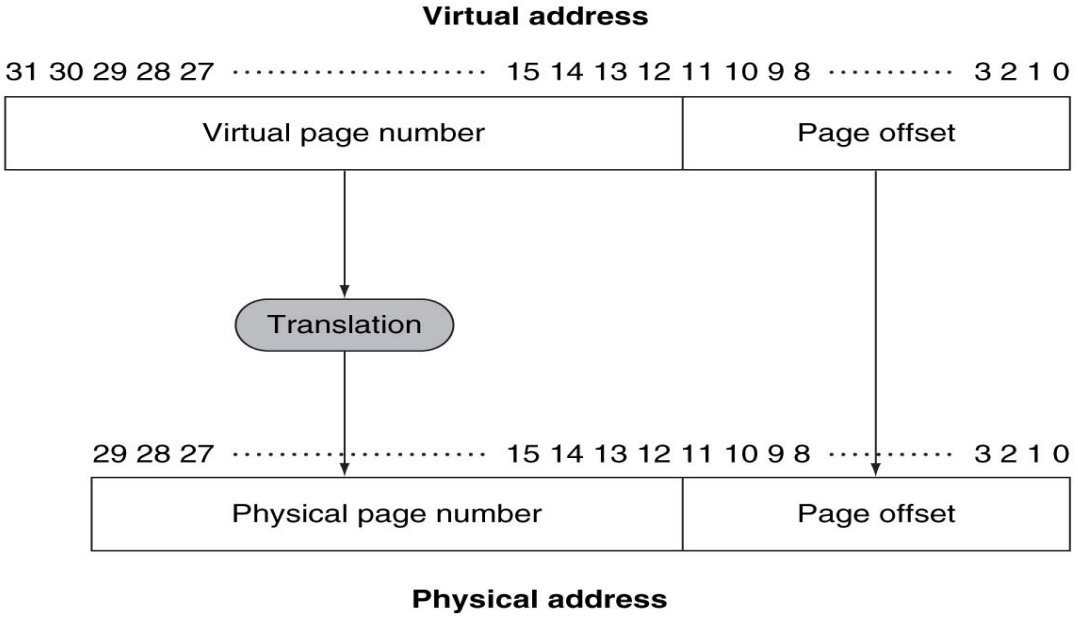
Memory accesses – virtual memory

- Typical memory instruction:
  - movsd %xmm0, 24(%rsp)
- Arguments in:
  - registers
  - main memory (primary storage)
    - address calculated based on the values stored in registers
- Addresses
  - in virtual address space
    - size: 2^n-1, n = 16, 32 or 64
    - n – characteristic to processor and operating system
  - translated to physical addresses
    - paging – the most popular mechanism
    - segments, uniform, ...



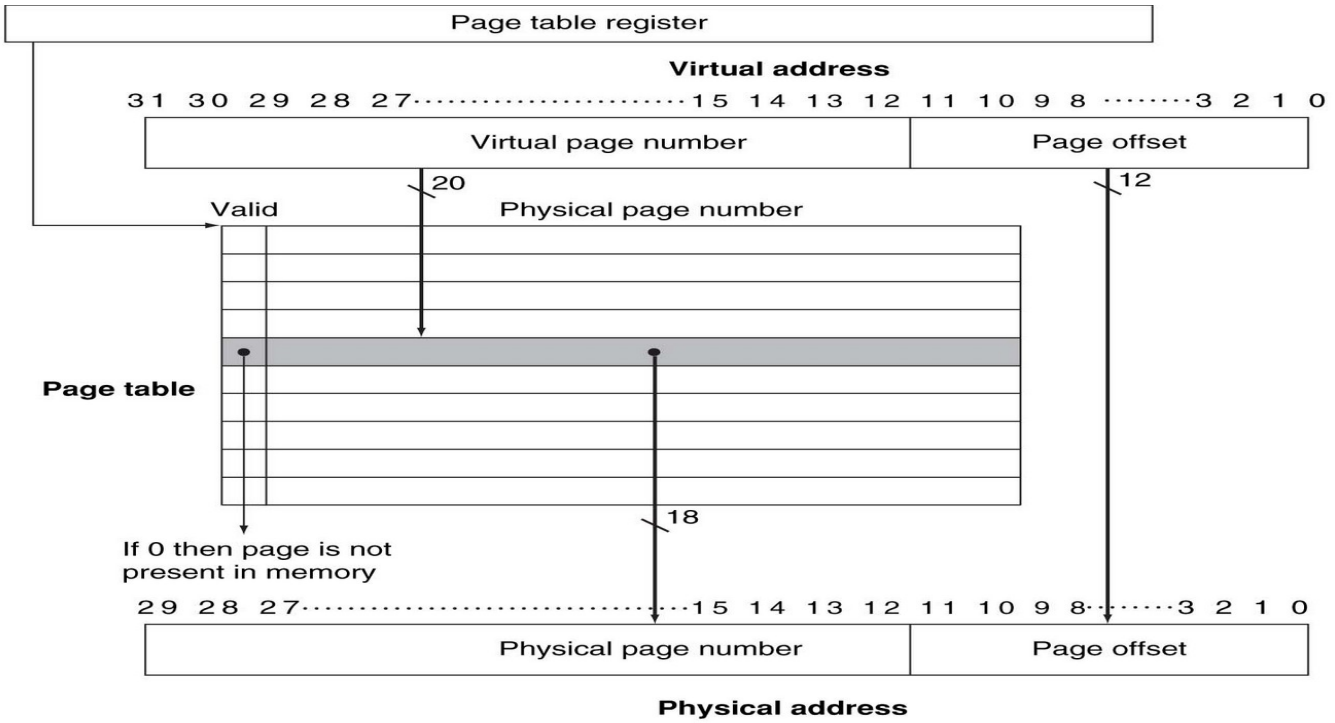
Paged virtual memory

- Paged virtual memory
  - virtual address space of the process divided into (virtual) pages
    - page size variable, usually 4 kb
    - physical memory divided into frames (physical pages) of the same size as virtual memory pages
    - each virtual address divided into page number and address within page (example: 32-bit address)
    - virtual addresses are translated into physical addresses



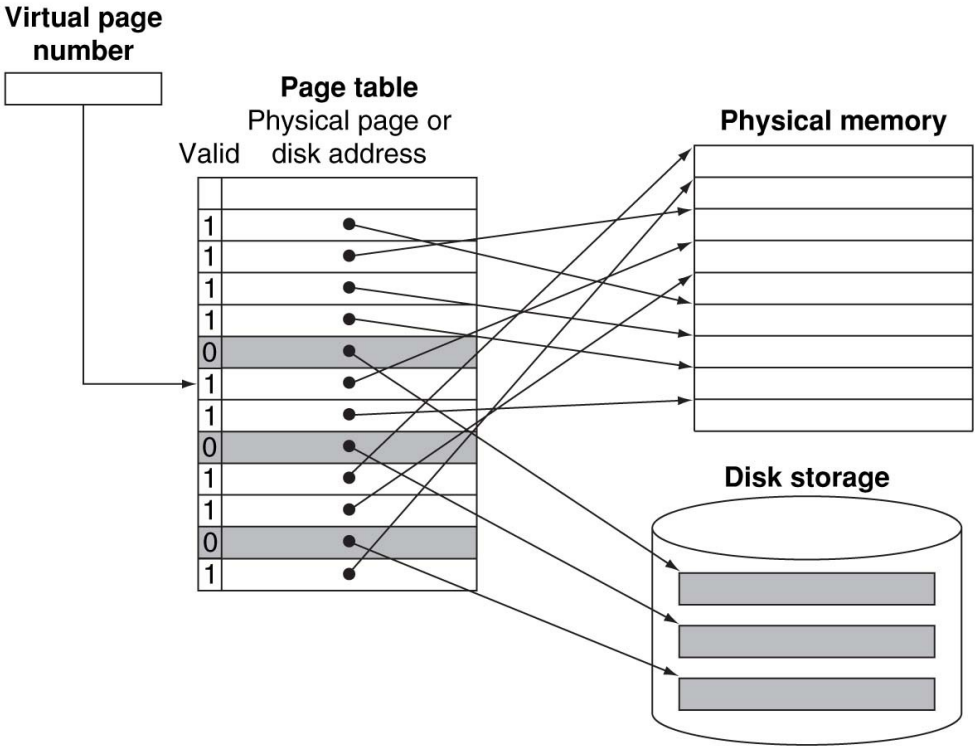
Paged virtual memory

- Paged virtual memory
  - address translation:
    - each page associated with the entry in the page table
    - page table specifies whether the page has assigned physical memory frame with its data
      - if yes, page table contains physical address of virtual page memory



Paged virtual memory

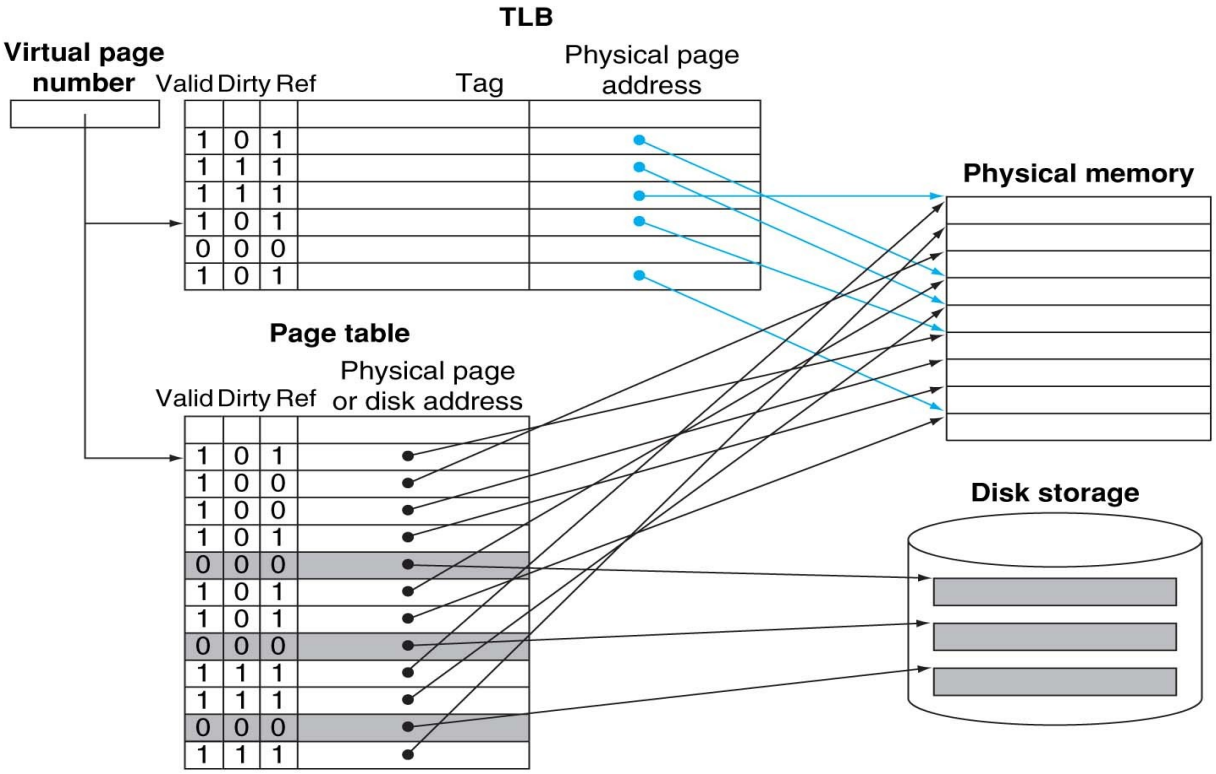
- Paging (page swapping)
  - virtual address space usually larger than physical memory
    - only fraction of all pages from virtual address space in physical memory
    - some pages may be stored in secondary storage (HDD, SDD)
    - operating system manages the storage of pages



  - when accessed page not in physical DRAM memory – page fault
    - » minor page fault – virtual page has not yet assigned physical page – possibly small overhead (no HDD or SDD access)
    - » major page fault – virtual page moved to secondary storage – page swap between secondary and primary storage large overhead

Paged virtual memory

- Page table
  - page table can be large – millions of pages (e.g. 2^(n-12), n=32,64)
    - stored in primary memory and possibly in secondary memory
  - to speed up access to page table it has a separate cache – TLB
    - TLB – translation lookaside buffer
    - within core
    - up to several hundred entries
  - there may be many page tables managed by the operating system (for different processes, segments, etc.)

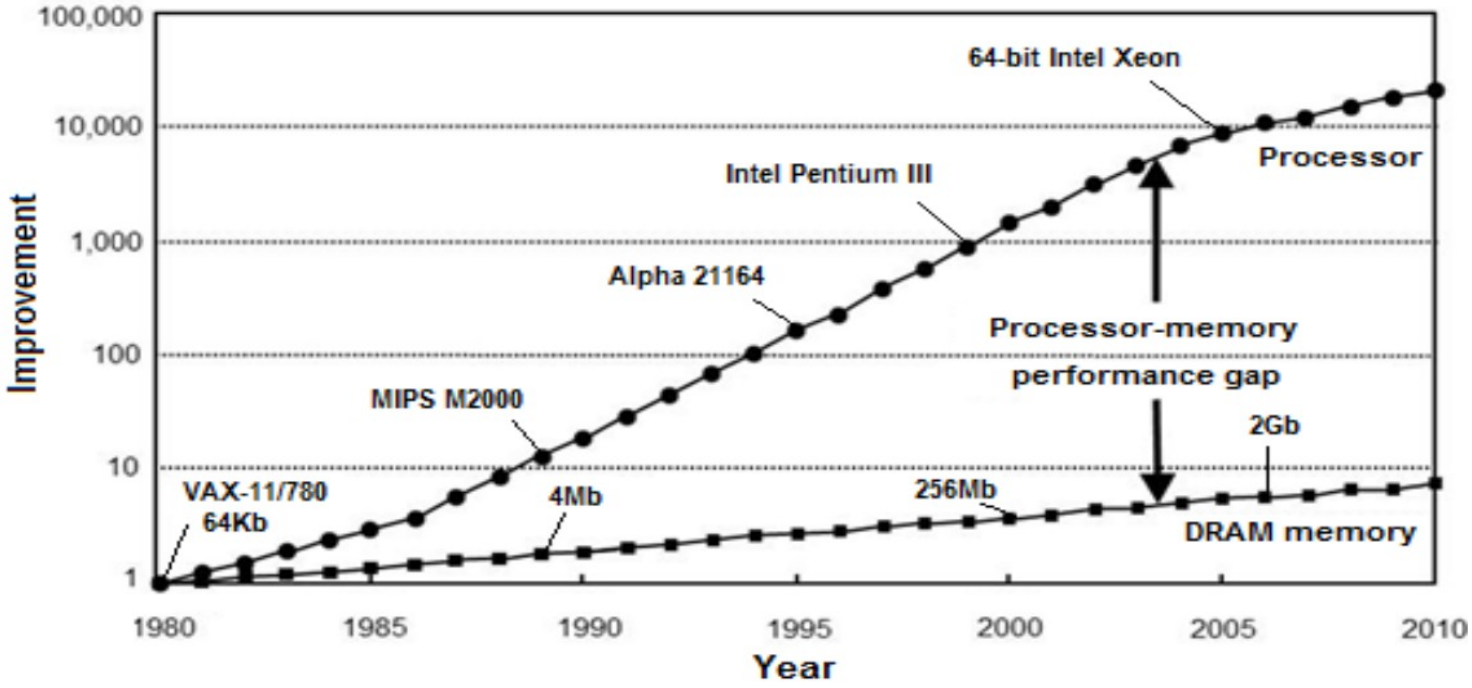


Memory access

- Instruction with virtual address
  - address translation
    - TLB look-up
      - TLB hit -> physical page address with data
      - TLB miss -> page walk – the search in memory
        - » entry found, physical page assigned -> TLB update, memory access retry, TLB hit, physical page address with data
        - » entry found, no physical page assigned -> page fault (both minor and major faults lead to page table and TLB update)
  - physical memory access
    - caches, cache coherence protocol, DRAM
  - some processors perform concurrently (at least partially) address translation and L1 cache access
- Thrashing
  - frequent page swaps due to many major page faults
  - must be avoided (any major page fault should be avoided)

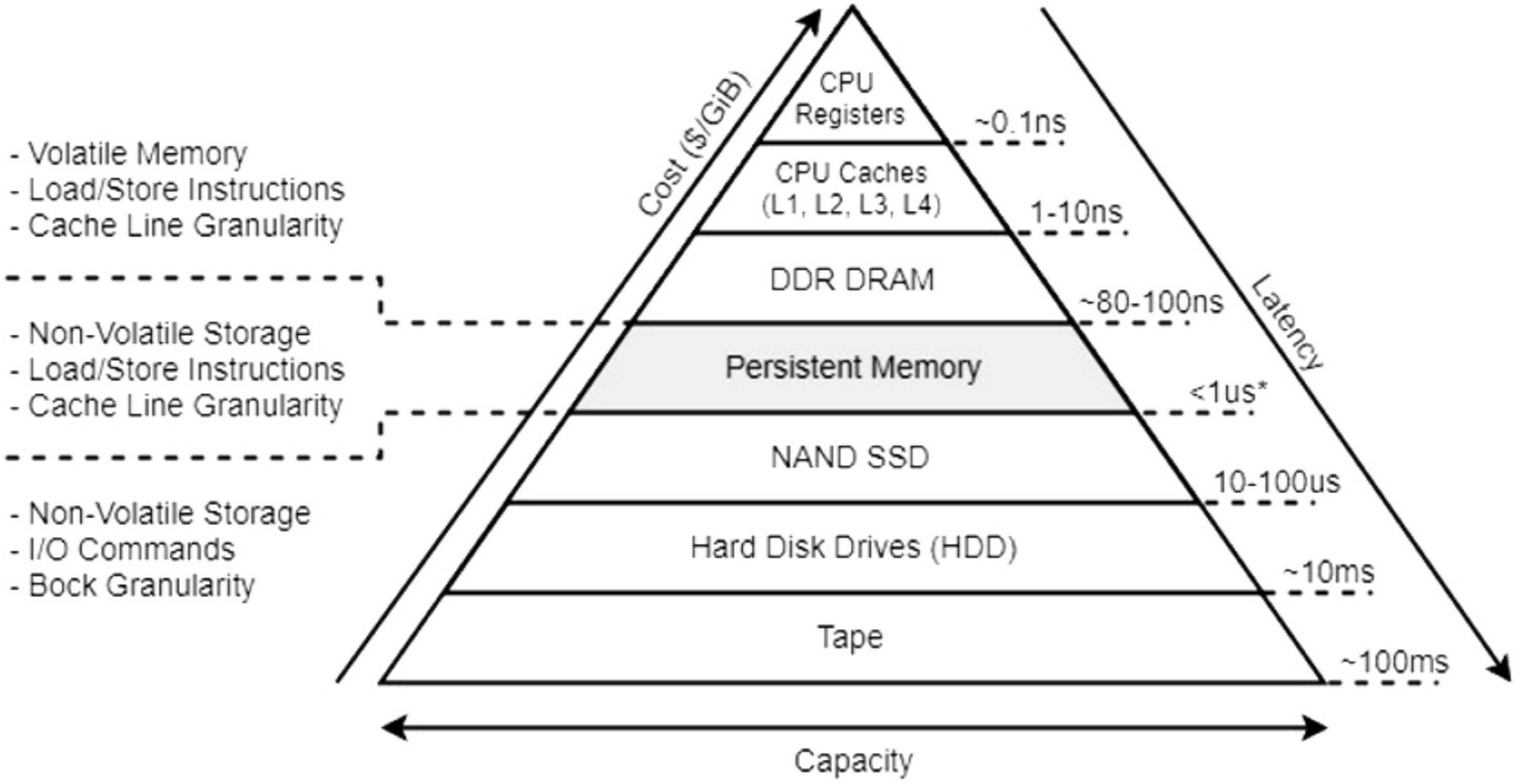
Memory wall

- Typical memory instruction: movsd %xmm0, 24(%rsp)
- Arguments:
  - registers
    - operating at the speed of processor core
  - main memory (primary storage)
    - slow DRAM modules



Memory hierarchy

- Solution to "memory wall" -> memory hierarchy

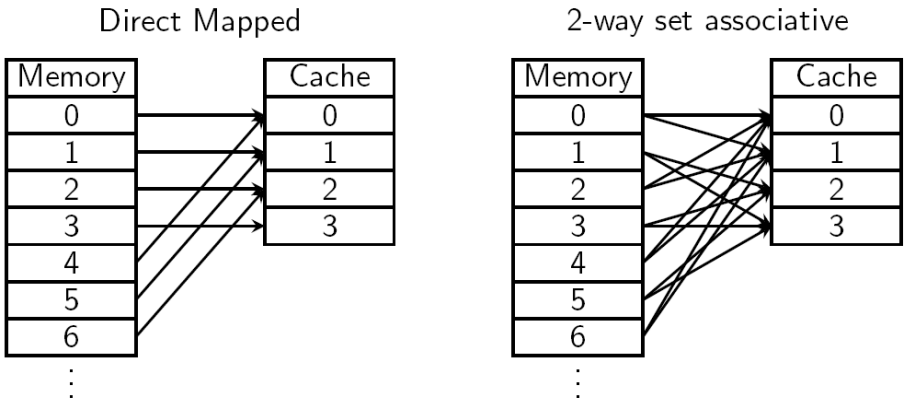


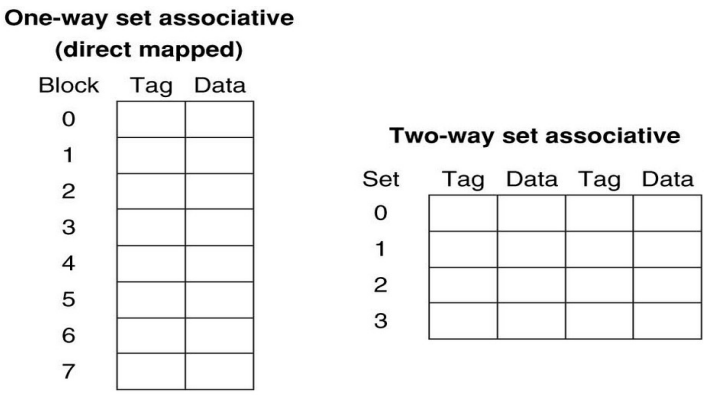
Cache hierarchy

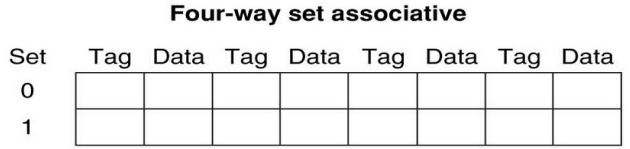
- Cache levels: L1, L2, L3, (L4?)
- Memory access:
  - memory request
    - L1 hit or ...
    - L1 miss
      - L2 hit (L1 cache line replacement) or ...
      - L2 miss
        - » etc. ... and finally:
          - » DRAM access (cache lines replacement)
- Cache effectiveness
  - locality of accesses
    - temporal – the same element accessed several times in a short period of time
    - spatial – several close in memory elements accessed in a short period of time

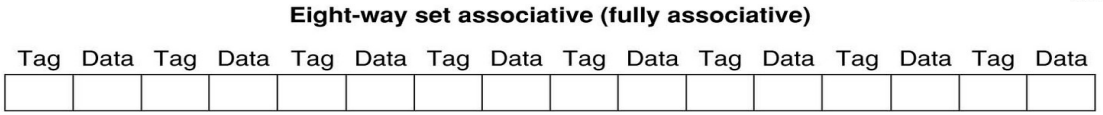
Cache organization

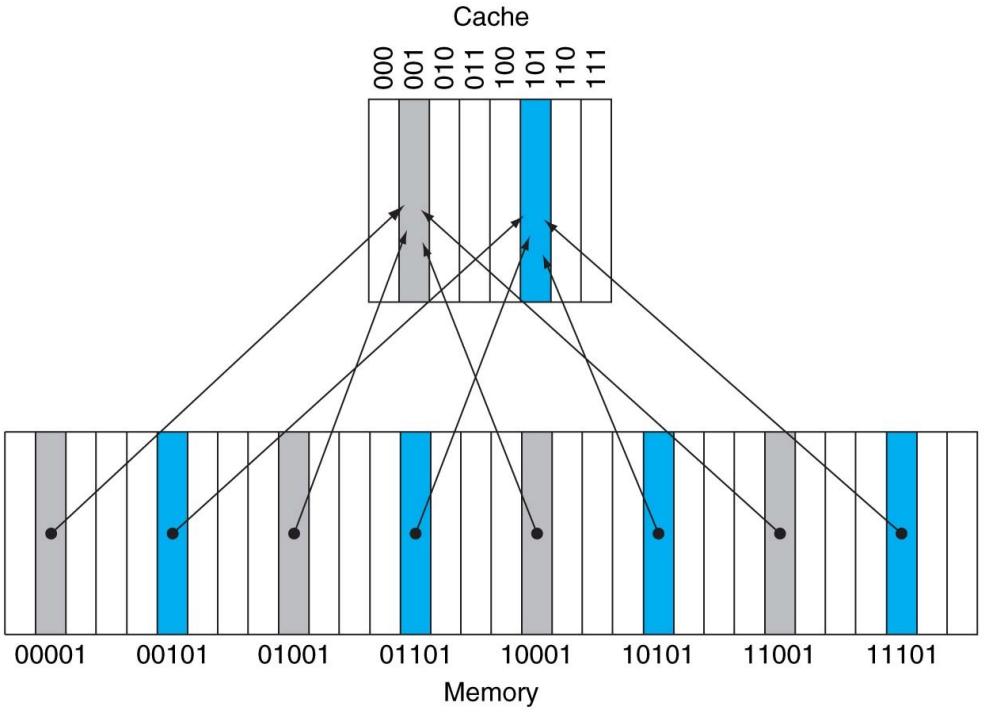
- Cache organization:
  - cache size
  - cache line size
  - cache associativity



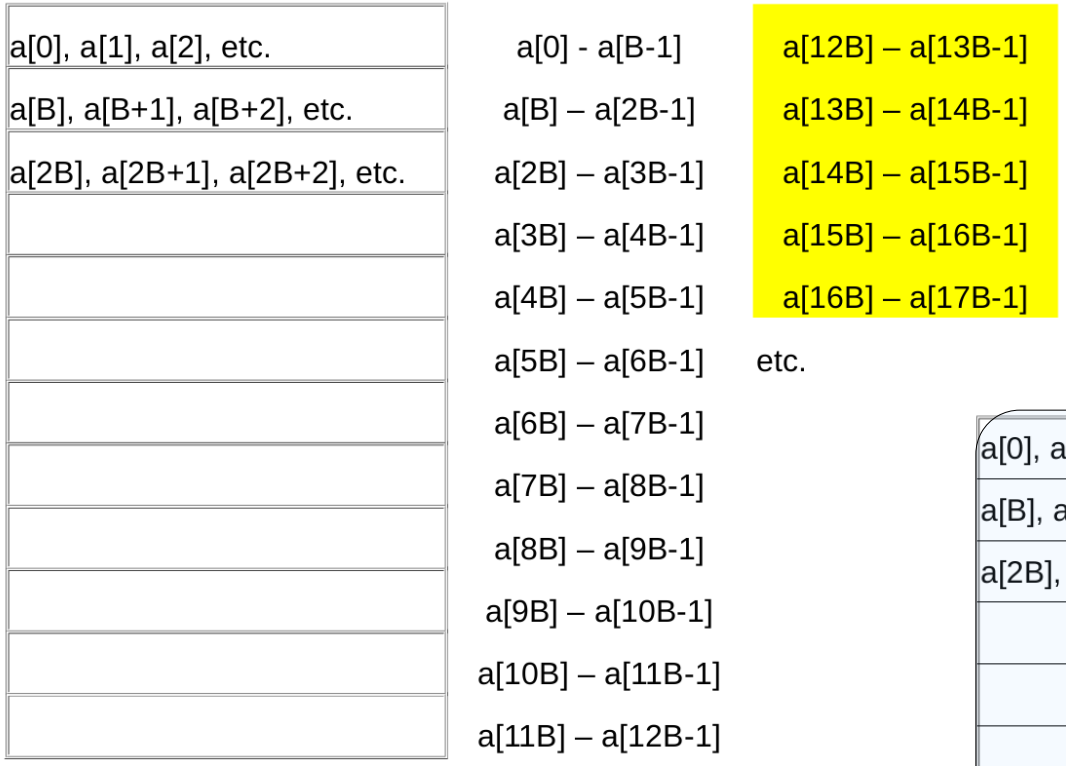








Mapping of memory blocks to cache lines

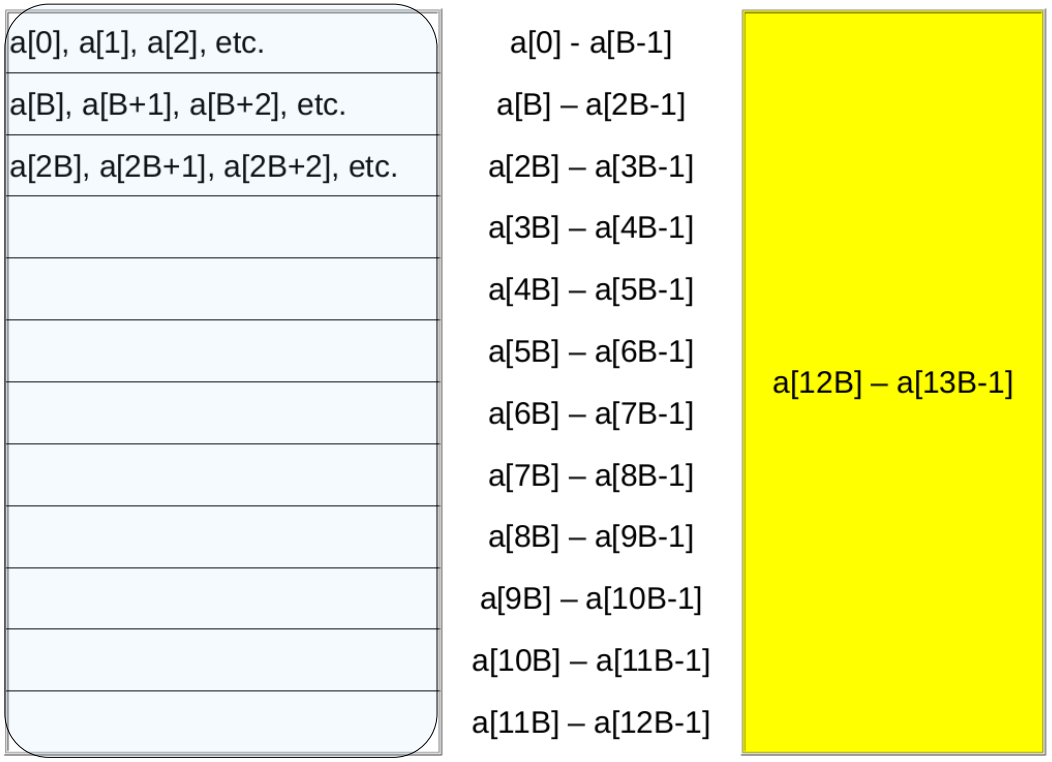


<- Direct mapped cache

- B elements of array **a** in one line
- each block in memory has exactly one corresponding cache line

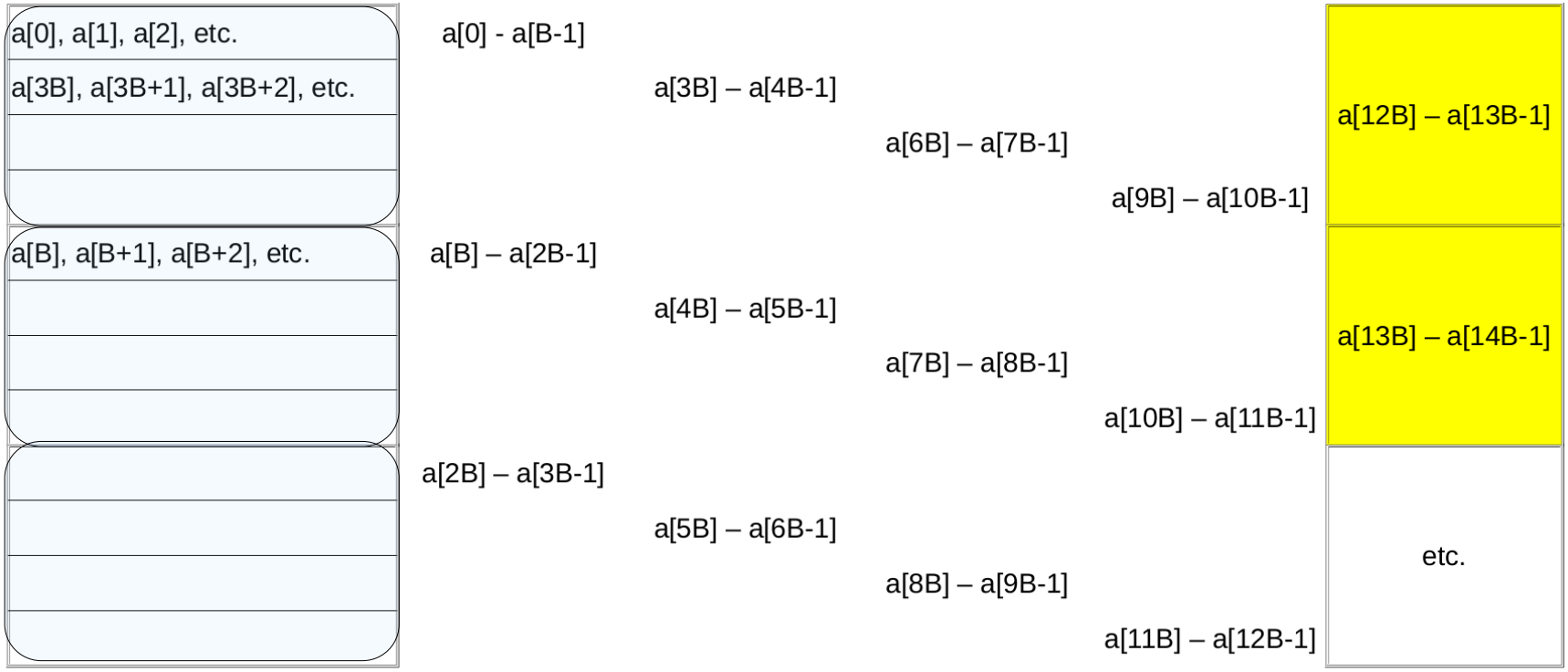
Fully associative cache ->

- B elements of array **a** in one line
- each block in memory can be stored in any cache line



4-way set associative cache

- Cache line – B variables
- First B elements of array **a** in one of 4 lines in a set
- The next B elements in one of 4 lines of the next set
- etc. , at certain moment one of cache lines must be replaced

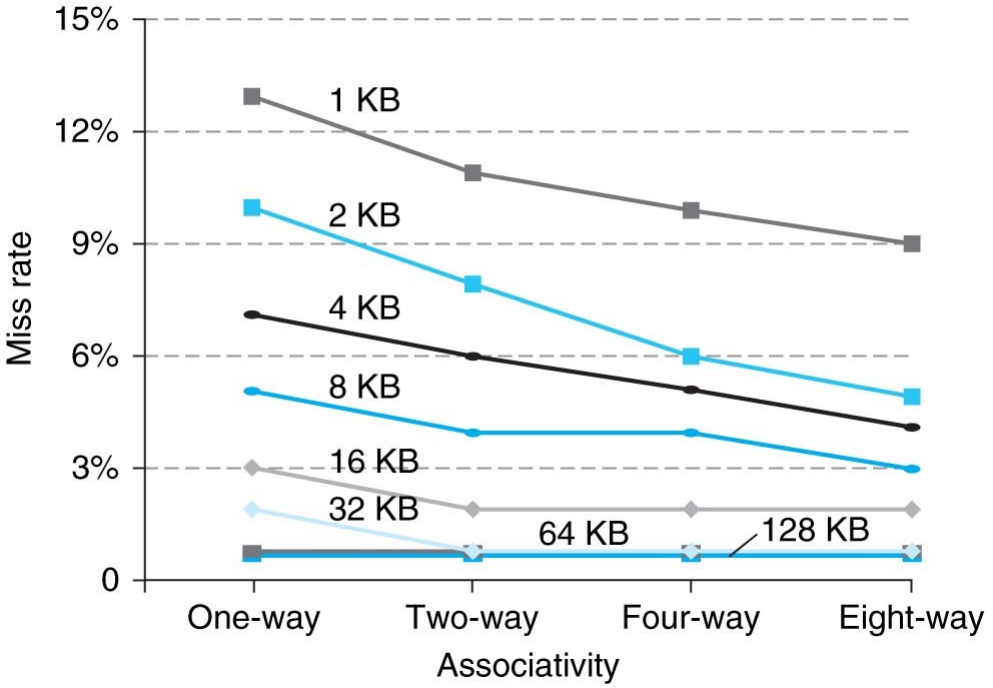


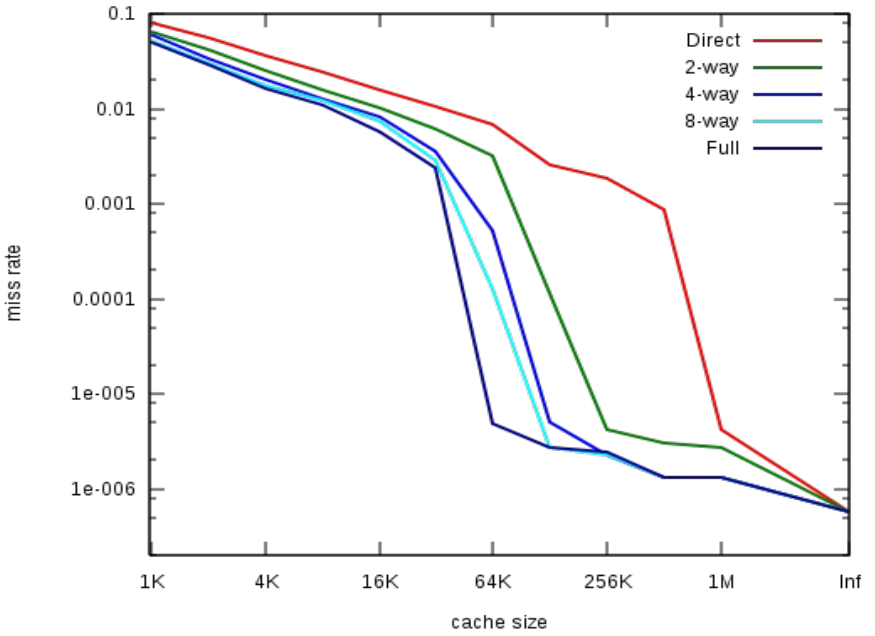
Cache misses

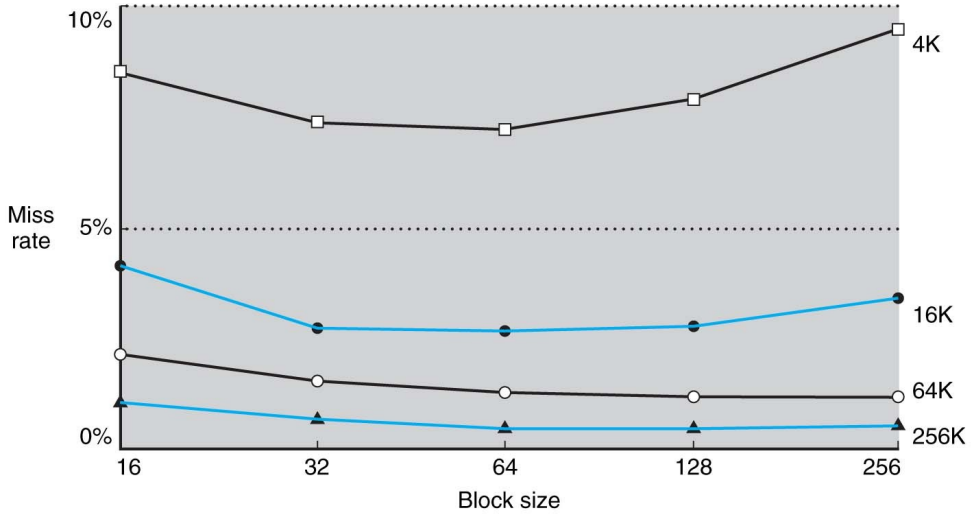
- Compulsory misses (aka cold start misses)
  - First access to a block
- Capacity misses
  - Due to finite cache size
  - A replaced block is later accessed again
- Conflict misses (aka collision misses) - In a non-fully associative cache
  - Due to competition for entries in a set. Would not occur in a fully associative cache of the same total size
- Coherency misses
  - Due to cache flushes to keep multiple caches coherent in a mutliprocessor

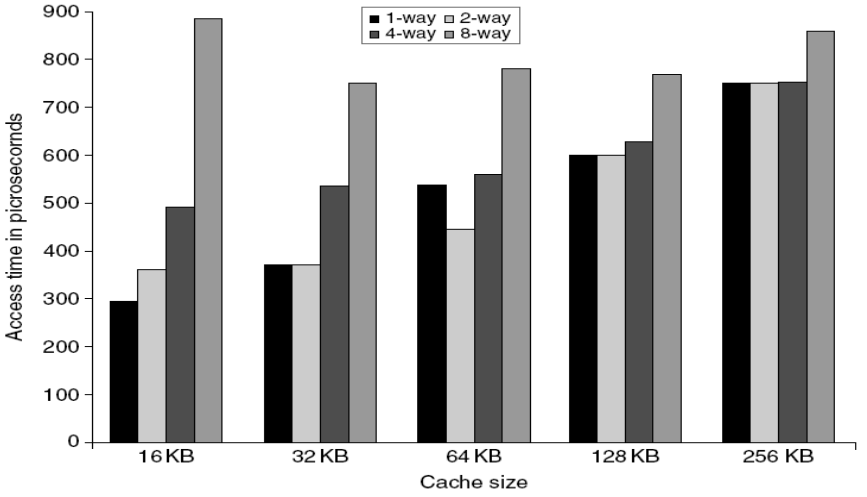
Design change	Effect on miss rate	Possible negative performance effect
Increase cache size	Decreases capacity misses	May increase access time
Increase associativity	Decreases miss rate due to conflict misses	May increase access time
Increase block size	Decreases miss rate for a wide range of block sizes due to spatial locality	Increases miss penalty. Very large block could increase miss rate

Cache performance









Cache organization

- Cache organization details:
  - inclusive vs exclusive
    - e.g. victim cache
  - harvard architecture
  - cache coherence
  - pipelining
  - non-blocking
  - multi banking
  - line replacement strategies:
    - random
    - FIFO
    - LRU (*least recently used*)
    - LFU (*least frequently used*)
    - more complex algorithms