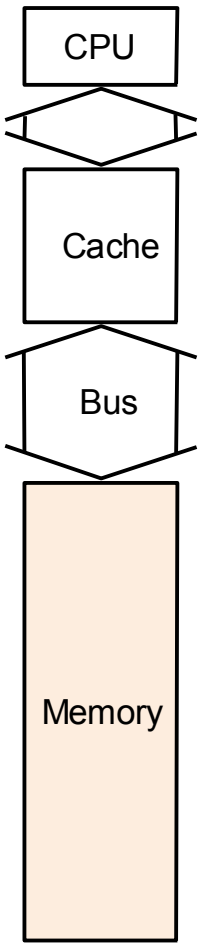
16.480/552 Microprocessor and Embedded Systems II

Virtual Memory

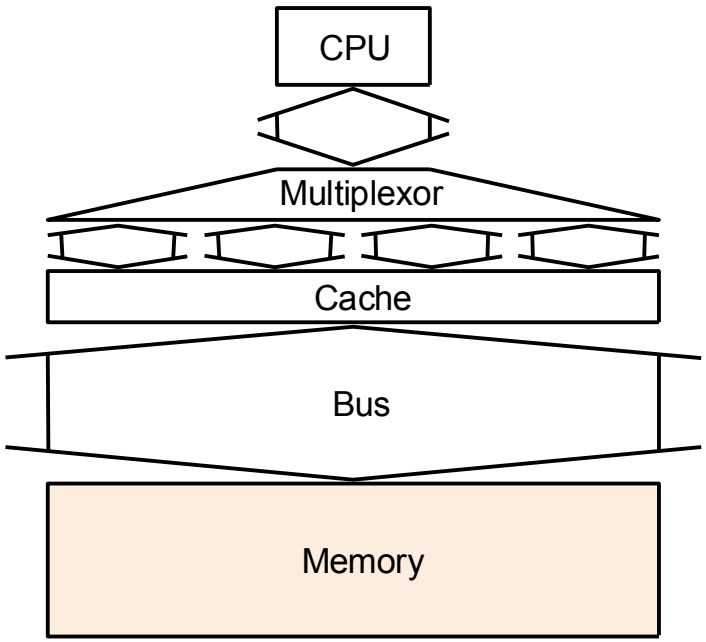
Memory Organization

- What is random access memory (RAM)? What are static RAM (SRAM) and dynamic RAM (DRAM)?
- What is DRAM Cell organization? How are the cells arranged internally? Memory addressing? Refreshing of DRAMs? Difference between DRAM and SRAM?
- Access time of DRAM = Row access time + column access time + refreshing
- What are page-mode and nibble-mode DRAMs?
- Synchronous SRAM or DRAM – Ability to transfer a burst of data given a starting address and a burst length – suitable for transferring a block of data from main memory to cache.

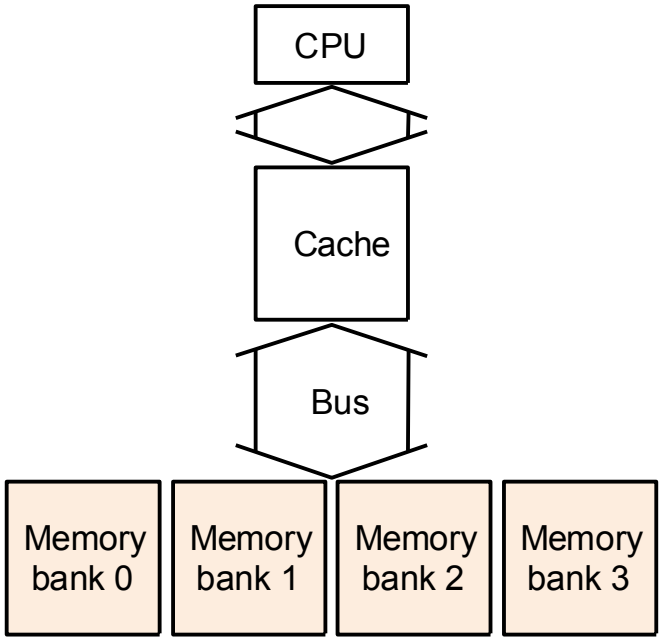
Main Memory Organizations



one-word wide
memory organization



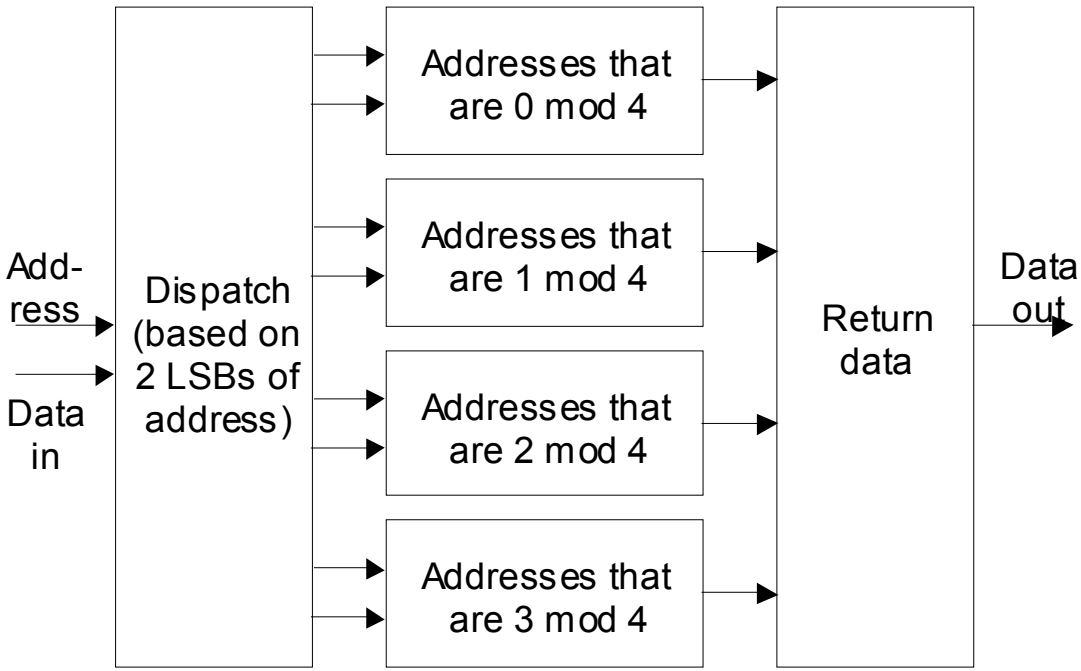
wide memory organization

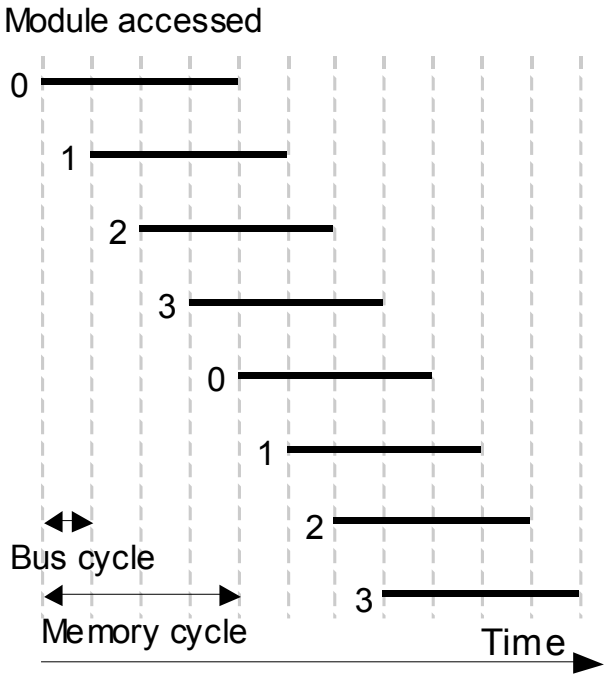


interleaved
memory organization

DRAM access time >> bus transfer time

Memory Interleaving





Interleaved memory is more flexible than wide-access memory in that it can handle multiple independent accesses at once.

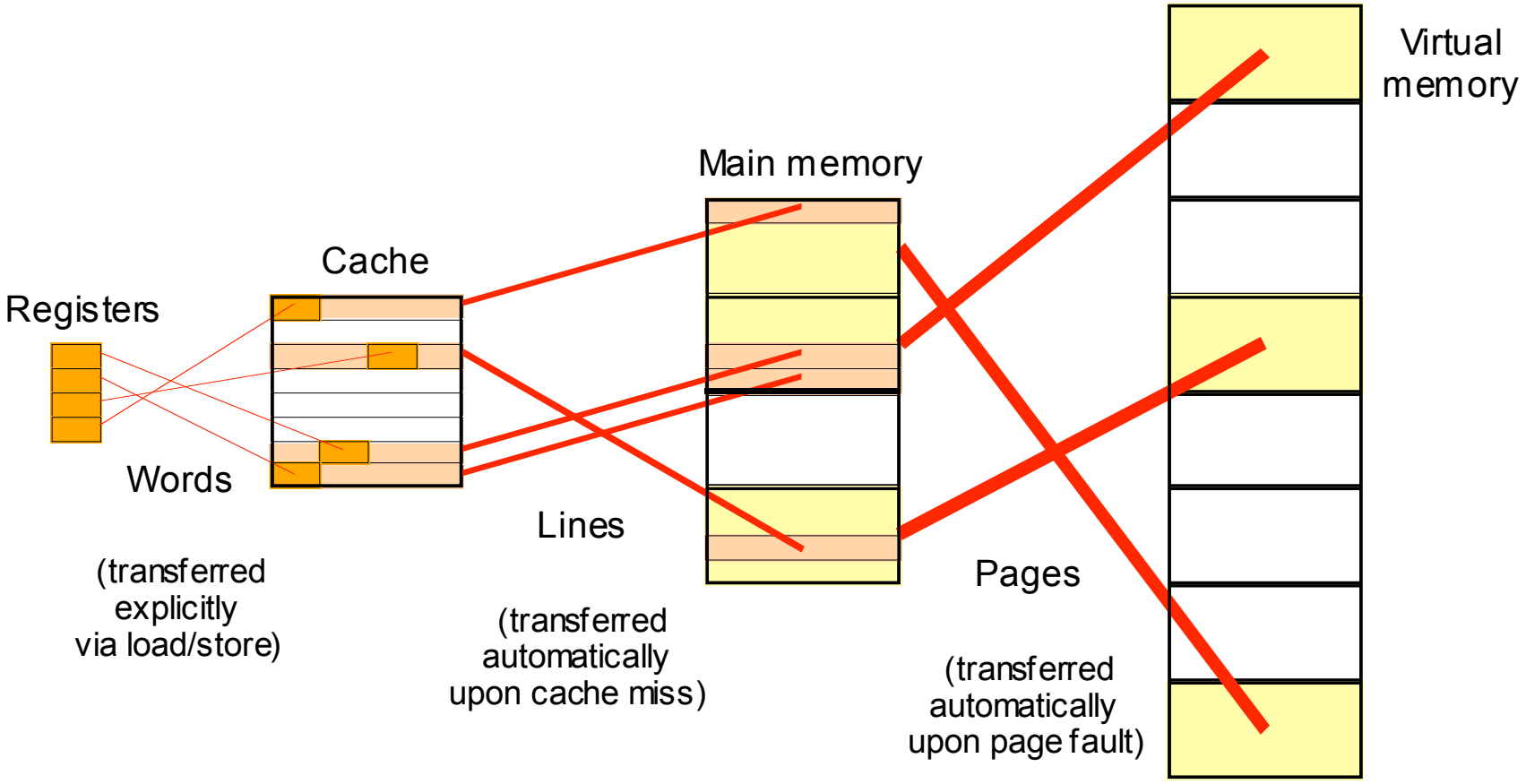
Memory Access Time Example

- Assume that it takes 1 cycle to send the address, 15 cycles for each DRAM access and 1 cycle to send a word of data.
- Assuming a cache block of 4 words and one-word wide DRAM, miss penalty = 1 + 4x15 + 4x1 = 65 cycles
- With main memory and bus width of 2 words, miss penalty = 1 + 2x15 + 2x1 = 33 cycles. For 4-word wide memory, miss penalty is 17 cycles. Expensive due to wide bus and control circuits.
- With interleaved memory of 4 memory banks and same bus width, the miss penalty = 1 + 1x15 + 4x1 = 20 cycles. The memory controller must supply consecutive addresses to different memory banks. Interleaving is universally adapted in high-performance computers.

Virtual Memory

- Idea 1: Many Programs sharing DRAM Memory so that context switches can occur
- Idea 2: Allow program to be written without memory constraints – program can exceed the size of the main memory
- Idea 3: Relocation: Parts of the program can be placed at different locations in the memory instead of a big chunk.
- Virtual Memory:
  - (1) DRAM Memory holds many programs running at same time (processes)
  - (2) use DRAM Memory as a kind of "cache" for disk

Memory Hierarchy: The Big Picture



Data movement in a memory hierarchy.

Impact of Technology on Virtual Memory

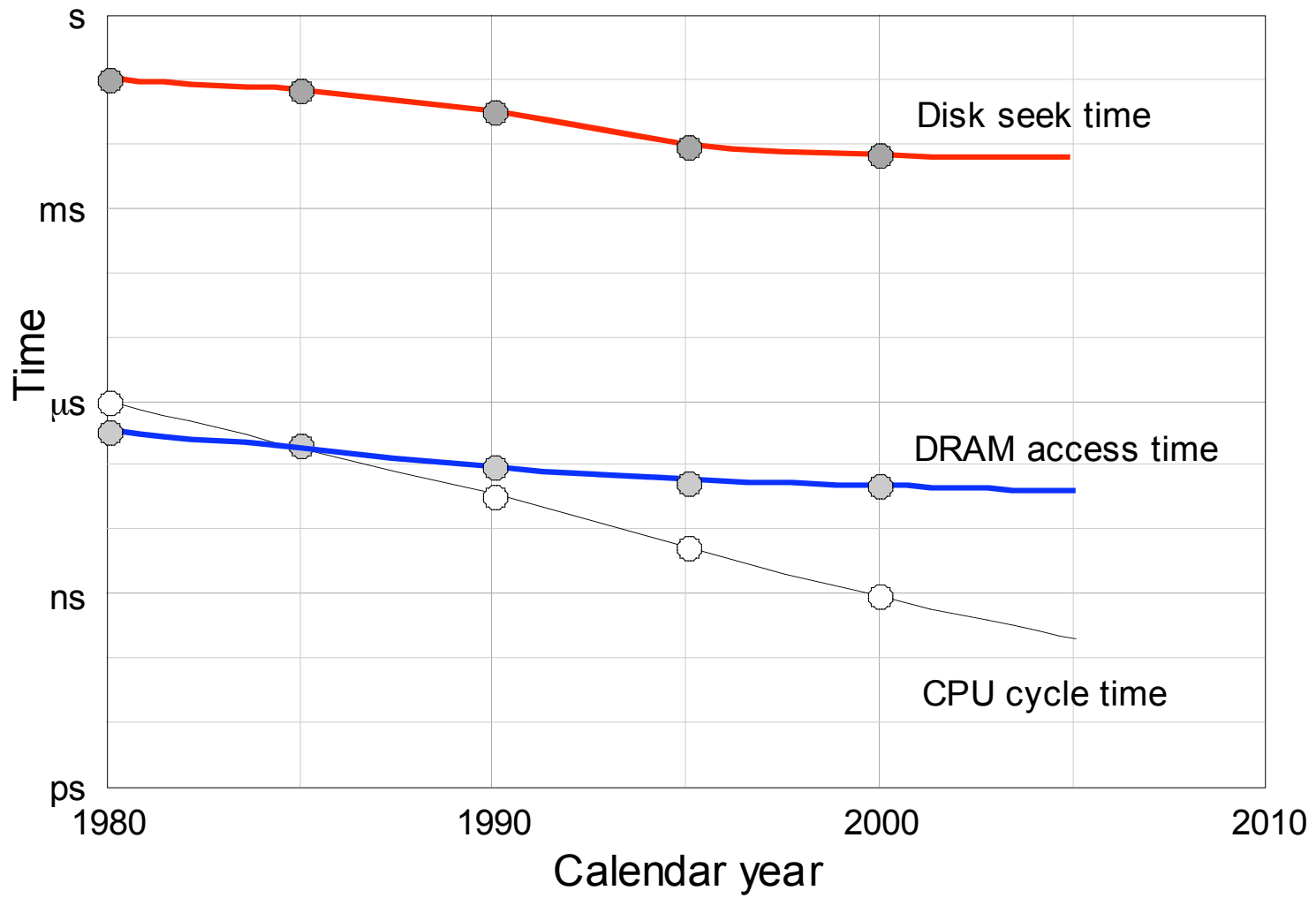


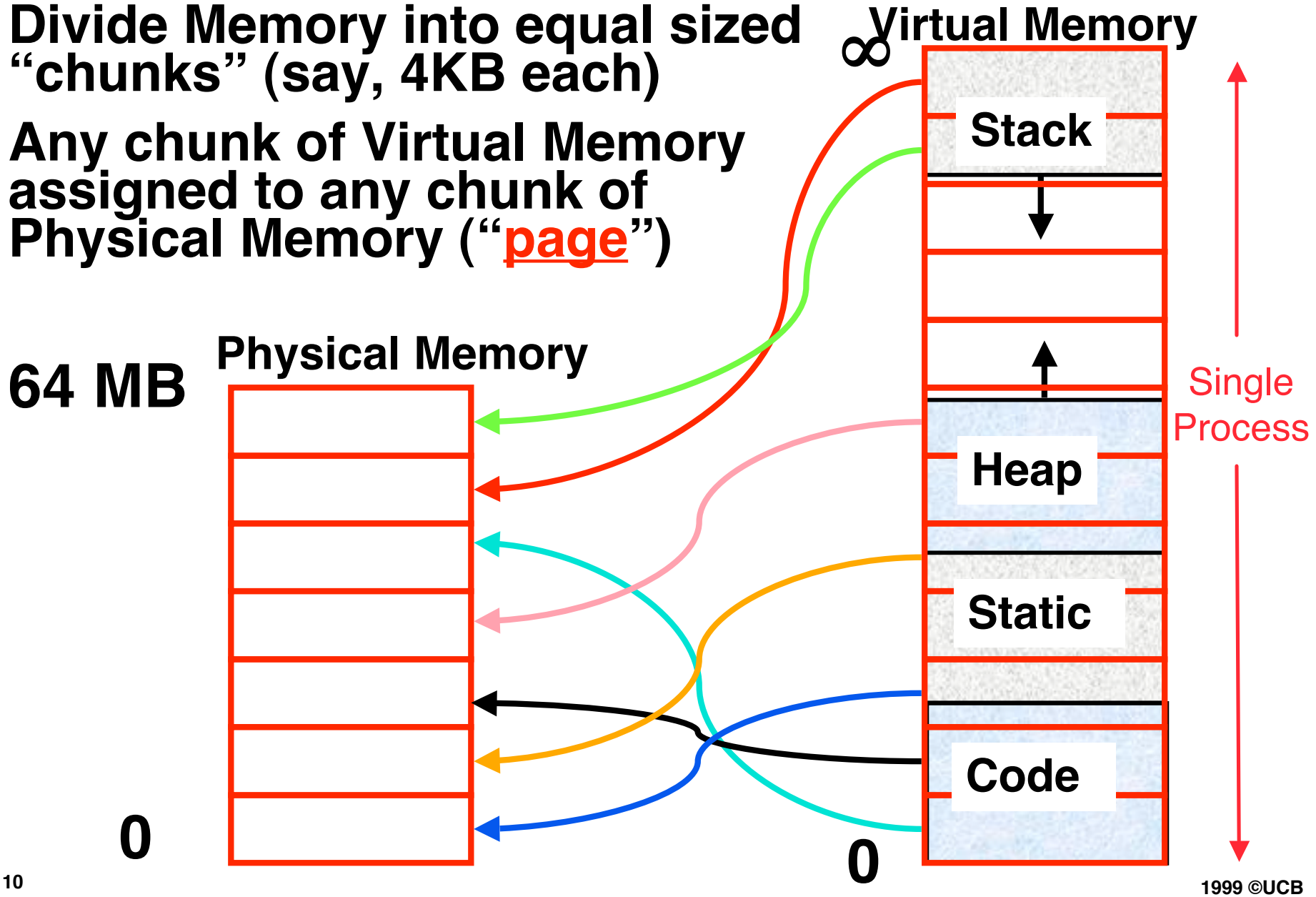
Fig. 20.11 Trends in disk, main memory, and CPU speeds.

Virtual Memory has own terminology

- Each process has its own private "virtual address space" (e.g., 2^{32} Bytes); CPU actually generates "virtual addresses"
- Each computer has a "physical address space" (e.g., 128 MegaBytes DRAM); also called "real memory"
- Address translation: mapping virtual addresses to physical addresses
  - Allows multiple programs to use (different chunks of physical) memory at same time
  - Also allows some chunks of virtual memory to be represented on disk, not in main memory (to exploit memory hierarchy)

Mapping Virtual Memory to Physical Memory

- Divide Memory into equal sized "chunks" (say, 4KB each)
- Any chunk of Virtual Memory assigned to any chunk of Physical Memory ("page")



Handling Page Faults

- A page fault is like a cache miss
  - Must find page in lower level of hierarchy
- If valid bit is zero, the Physical Page Number points to a page on disk
- When OS starts new process, it creates space on disk for all the pages of the process, sets all valid bits in page table to zero, and all Physical Page Numbers to point to disk
  - called Demand Paging - pages of the process are loaded from disk only as needed

Comparing the 2 levels of hierarchy

	Cache	*Virtual Memory*
°	Block or Line	Page
°	Miss	Page Fault
°	Block Size: 32-64B	Page Size: 4K-16KB
°	Placement: Direct Mapped, N-way Set Associative	Fully Associative
°	Replacement: LRU or Random	Least Recently Used (LRU) approximation
°	Write Thru or Back	Write Back
°	How Managed: Hardware	Hardware + Software (Operating System)

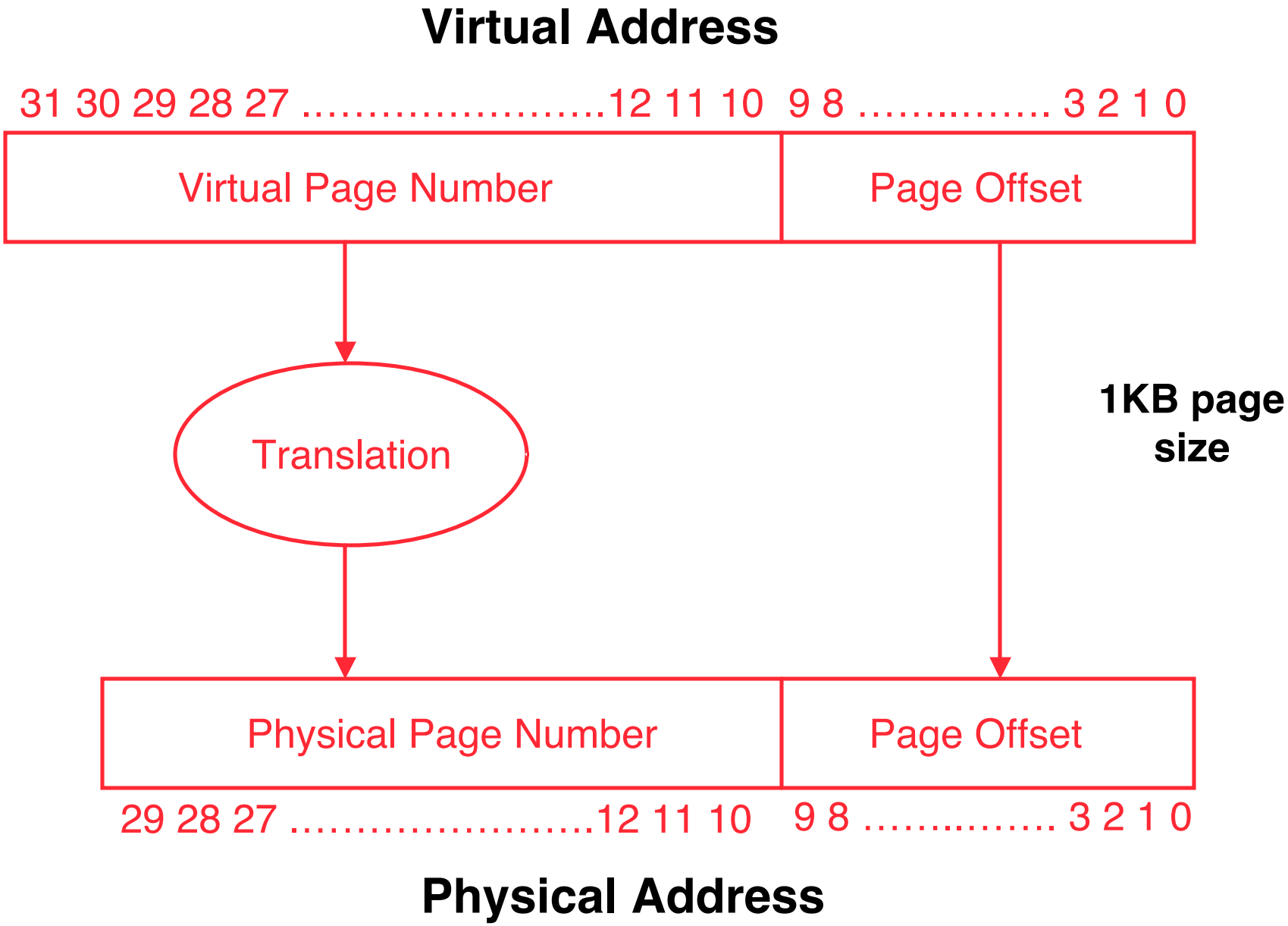
How to Perform Address Translation?

- VM divides memory into equal sized pages
- Address translation relocates entire pages
  - offsets within the pages do not change
  - if make page size a power of two, the virtual address separates into two fields:

virtual address

  - like cache index, offset fields

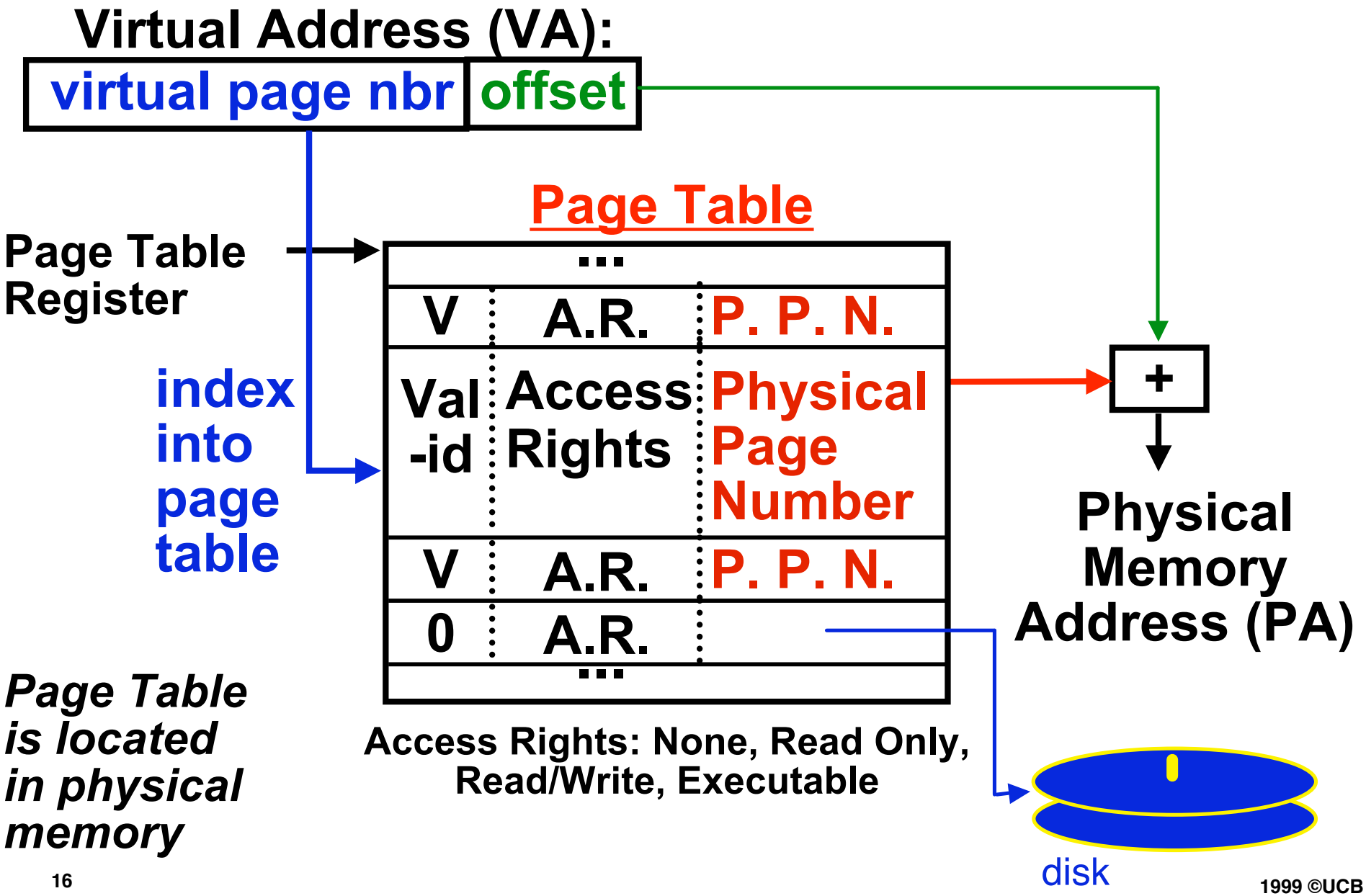
Mapping Virtual to Physical Address



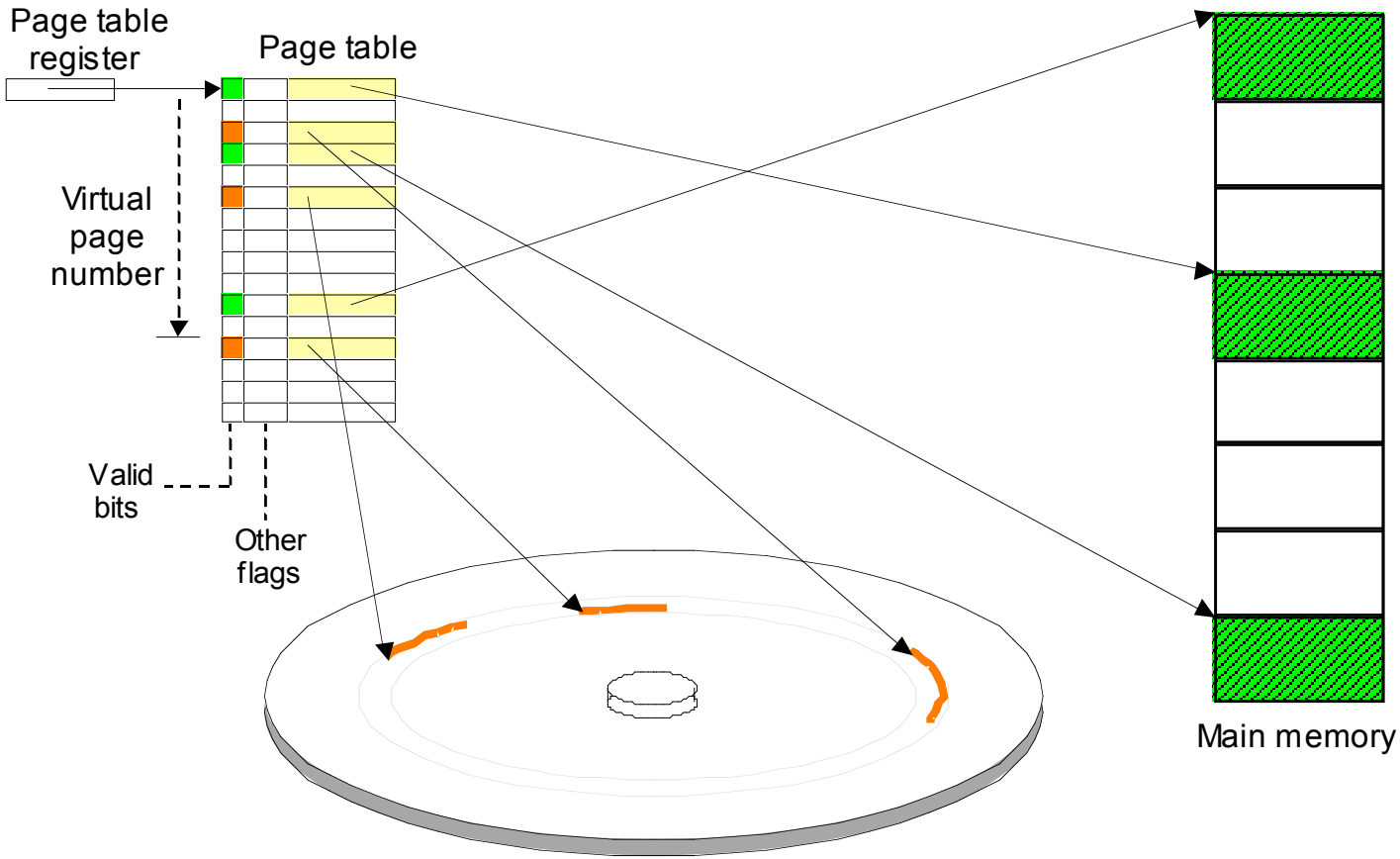
Address Translation

- Want fully associative page placement
- How to locate the physical page?
- Search impractical (too many pages)
- A page table is a data structure which contains the mapping of virtual pages to physical pages
  - There are several different ways, all up to the operating system, to keep this data around
- Each process running in the system has its own page table

Address Translation: Page Table

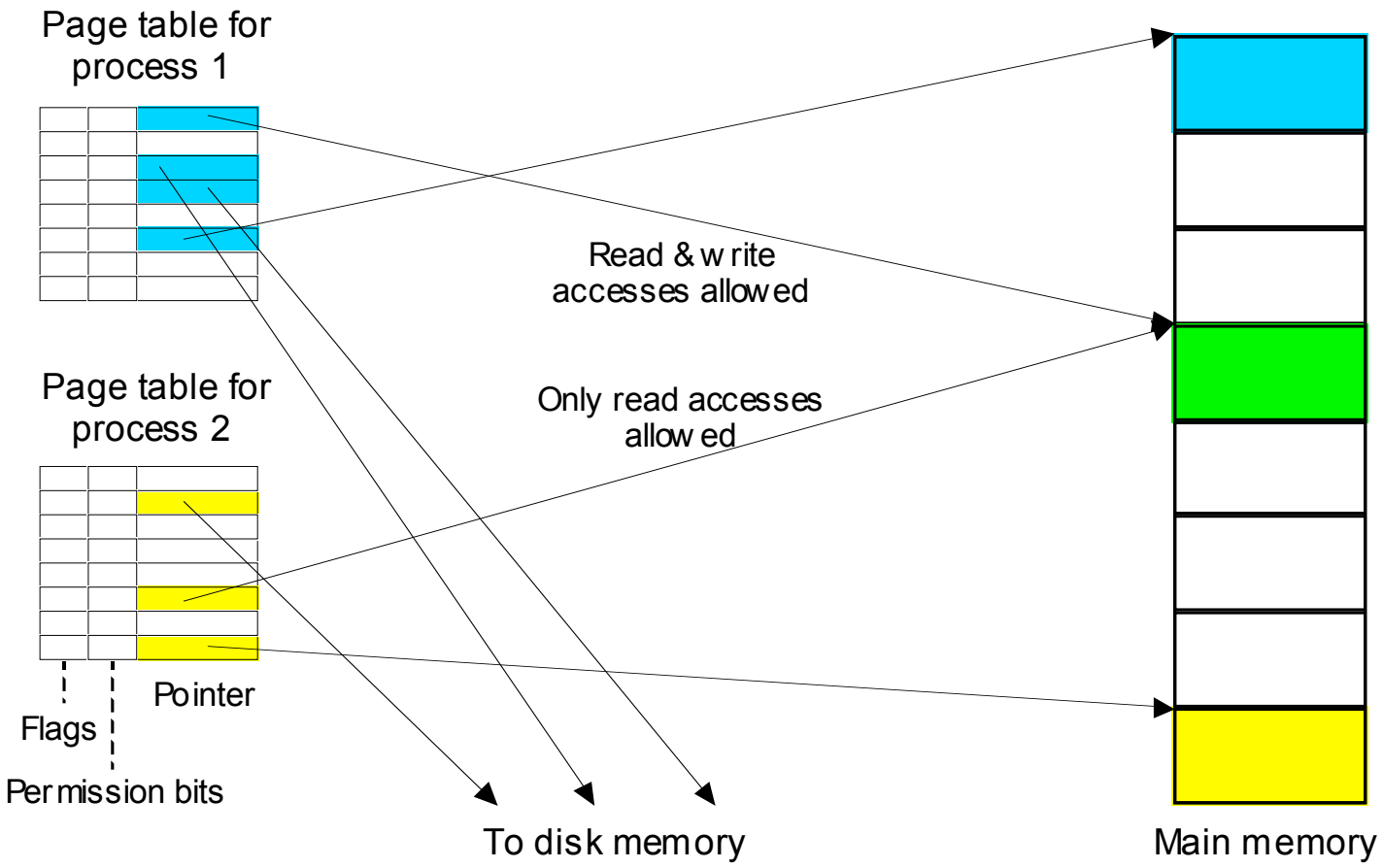


Page Tables and Address Translation



The role of page table in the virtual-to-physical address translation process.

Protection and Sharing in Virtual Memory



Virtual memory as a facilitator of sharing and memory protection.

Optimizing for Space

- **Page Table too big!**
  - 4GB Virtual Address Space ÷ 4 KB page
    ⇒ 2^{20} (~ 1 million) Page Table Entries
    ⇒ 4 MB just for Page Table of single process!

- **Variety of solutions to tradeoff Page Table size for slower performance when miss occurs in TLB**

  Use a limit register to restrict page table size and let it grow with more pages,Multilevel page table, Paging page tables, etc.

(Take O/S Class to learn more)

How Translate Fast?

- Problem: Virtual Memory requires two memory accesses!
  - one to translate Virtual Address into Physical Address (page table lookup)
  - one to transfer the actual data (cache hit)
  - But Page Table is in physical memory! => 2 main memory accesses!
- Observation: since there is locality in pages of data, must be locality in virtual addresses of those pages!
- Why not create a cache of virtual to physical address translations to make translation fast? (smaller is faster)
- For historical reasons, such a "page table cache" is called a Translation Lookaside Buffer, or TLB

Typical TLB Format

Virtual Page Nbr	Physical Page Nbr	Valid	Ref	Dirty	Access Rights

"tag" "data"

- TLB just a cache of the page table mappings

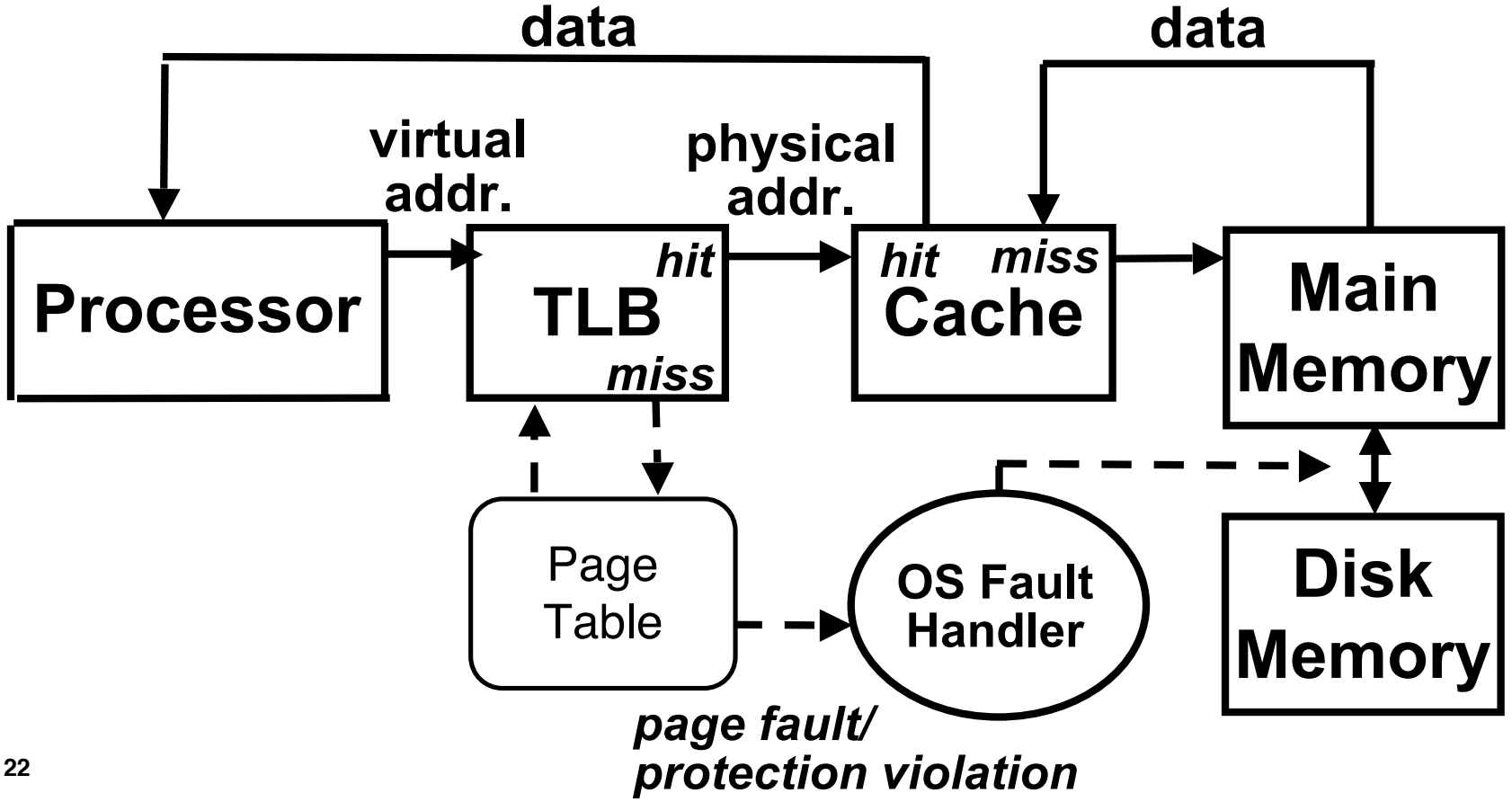
- Dirty: since use write back, need to know whether or not to write page to disk when replaced
- Ref: Used to calculate LRU on replacement

- TLB access time comparable to cache (much less than main memory access time)

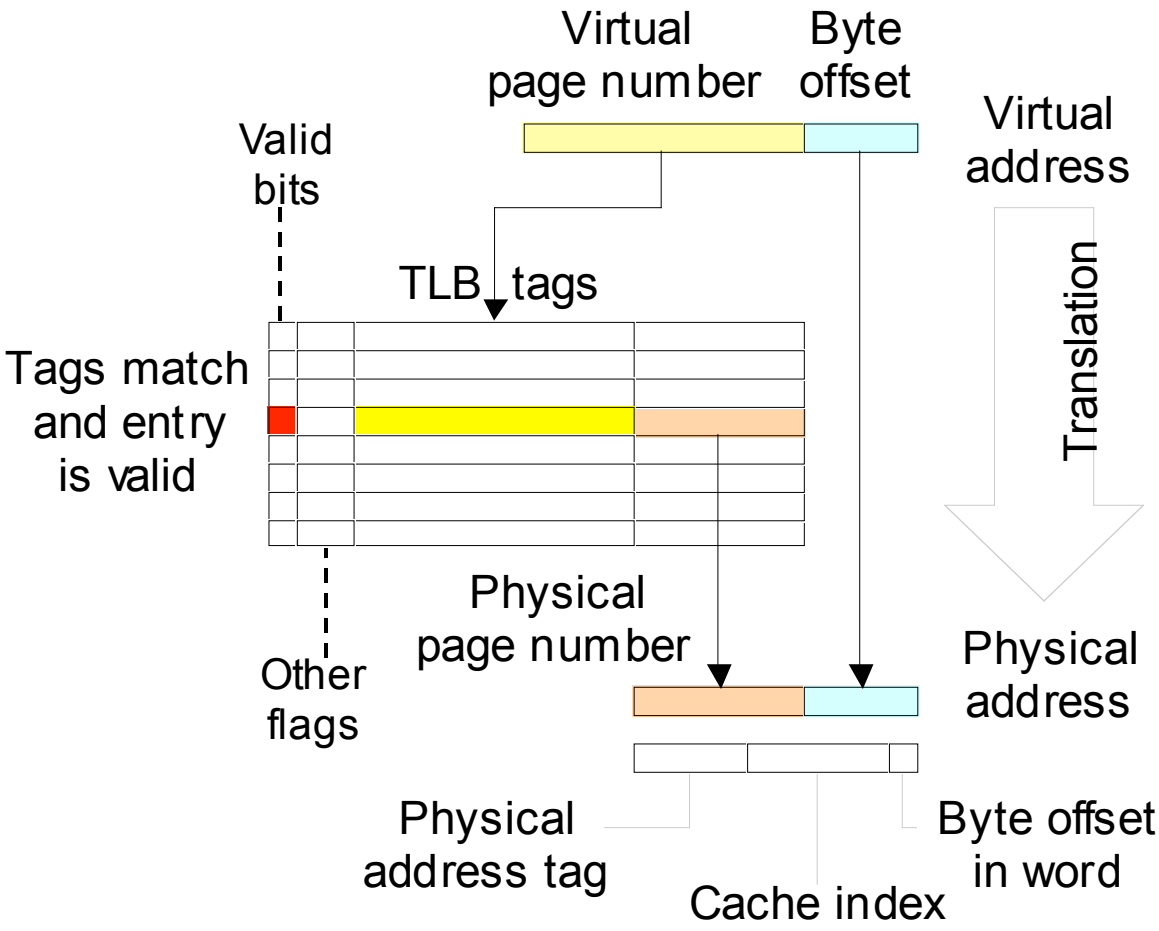
Translation Look-Aside Buffers

- TLB is usually small, typically 32-4,096 entries

- Like any other cache, the TLB can be fully associative, set associative, or direct mapped

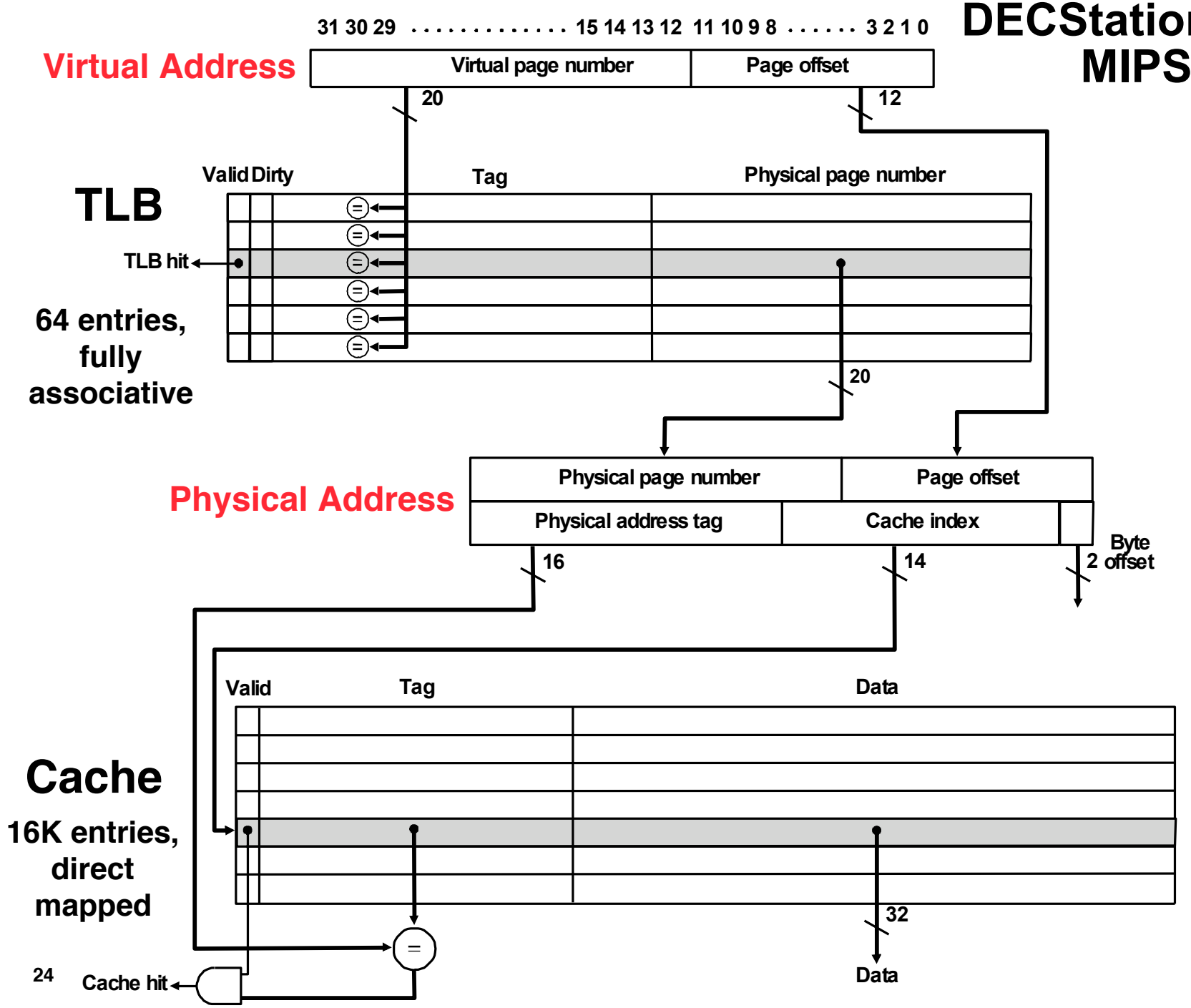


Translation Lookaside Buffer



Virtual-to-physical address translation by a TLB and how the resulting physical address is used to access the cache memory.

DECStation 3100/ MIPS R2000



Real Stuff: Pentium Pro Memory Hierarchy

- Address Size: 32 bits (VA, PA)
- VM Page Size: 4 KB, 4 MB
- TLB organization: separate i,d TLBs
  (i-TLB: 32 entries,
  d-TLB: 64 entries)
  4-way set associative
  LRU approximated
  hardware handles miss
- L1 Cache: 8 KB, separate i,d
  4-way set associative
  LRU approximated
  32 byte block
  write back
- L2 Cache: 256 or 512 KB