CS510 Computer Architecture

Lecture 17: Virtual Memory

Soontae Kim

Spring 2017

School of Computing, KAIST

Announcement

- **Homework assignment #3**
  - Posted on class web site
  - Due on May 26 (Firday)

Virtual Memory: Motivation

- **Original Motivation:**
  - **Illusion of having larger physical main memory (using demand paging)**
  - **Allows program and data address relocation by automating the process of code and data movement between main memory and storage.**
- **Additional Current Motivation:**
  - **Fast process start-up.**
  - **Protection from illegal memory access.**
    - ***Needed for multi-tasking operating systems.***
  - **Controlled code and data sharing among processes.**
    - ***Needed for multi-threaded programs***

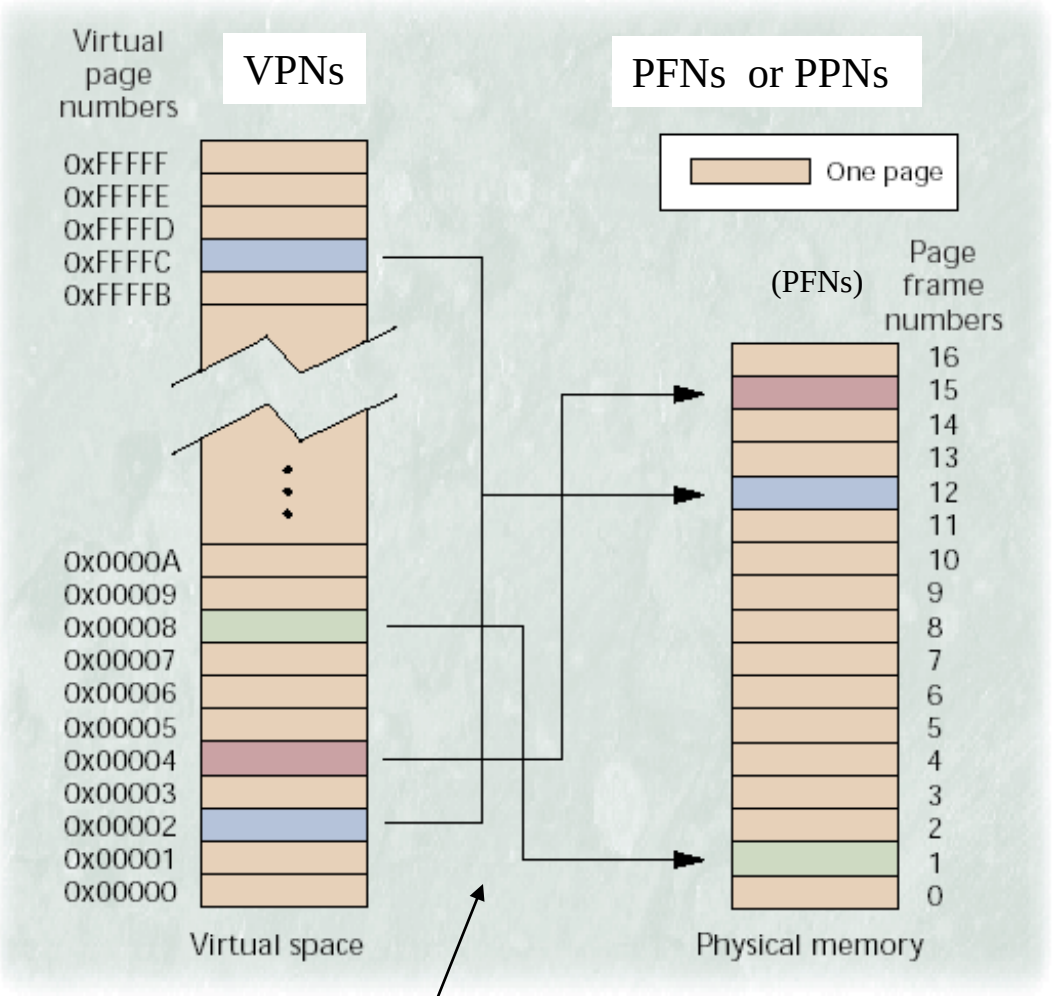
Virtual Memory: Overview

- **Virtual memory controls two levels of the memory hierarchy:**
  - **Main memory (DRAM).**
  - **Mass storage (usually magnetic disks).**
- **Main memory is divided into blocks allocated to different running processes in the system by the OS:**
  - **Fixed size blocks: Pages (size 4k to 64k bytes). (Most common)**
  - **Variable size blocks: Segments (largest size 2^{16} up to 2^{32}).**
  - **Paged segmentation: Large variable/fixed size segments divided into a number of fixed size pages (X86, PowerPC).**
- **At any given time, for any running process, a portion of its data/code is loaded (allocated) in main memory while the rest is available only in mass storage.**
- **A program code/data block needed for process execution and not present in main memory result in a page fault (address fault) and the page has to be loaded into main memory by the OS from disk (demand paging).**
- **A program can run in any location in main memory or disk by using a relocation/mapping mechanism controlled by the operating system which maps (translates) the address from virtual address space (logical program address) to physical address space (main memory, disk).**

Virtual Address Space Vs. Physical Address Space

Virtual memory system stores only the most often used portions of a process address space in main memory and retrieves other portions from a disk as needed (demand paging).

The virtual-memory space is divided into pages identified by virtual page numbers (VPNs), which are mapped to physical page numbers (PPNs) or page frame numbers (PFNs), in physical memory as shown on the right.



Paging is assumed here

Virtual address to physical address mapping or translation

Virtual Address Space = Process Logical Address Space

Basic Virtual Memory Management

- **Operating system makes decisions regarding which virtual (logical) pages of a process should be allocated in real physical memory and where (demand paging) assisted with hardware Memory Management Unit (MMU)**
- **On memory access -- If no valid virtual page to physical page translation (i.e page not allocated in main memory)**
  - **Page fault to operating system**
  - **Operating system requests page from disk**
  - **Operating system chooses page for replacement**
    - **writes back to disk if modified**
  - **Operating system allocates a page in physical memory**

Paging is assumed

Typical Parameter Range For Cache & Virtual Memory

Parameter	First-level cache	Virtual memory
Block (page) size	16–128 bytes	4096–65,536 bytes
Hit time	1–3 clock cycles	100–200 clock cycles
Miss penalty	8–200 clock cycles	1,000,000–10,000,000 clock cycles
(access time)	(6–160 clock cycles)	(800,000–8,000,000 clock cycles)
(transfer time)	(2–40 clock cycles)	(200,000–2,000,000 clock cycles)
Miss rate	0.1–10%	0.00001–0.001%
Address mapping	25–45-bit physical address to 14–20-bit cache address	32–64-bit virtual address to 25–45-bit physical address

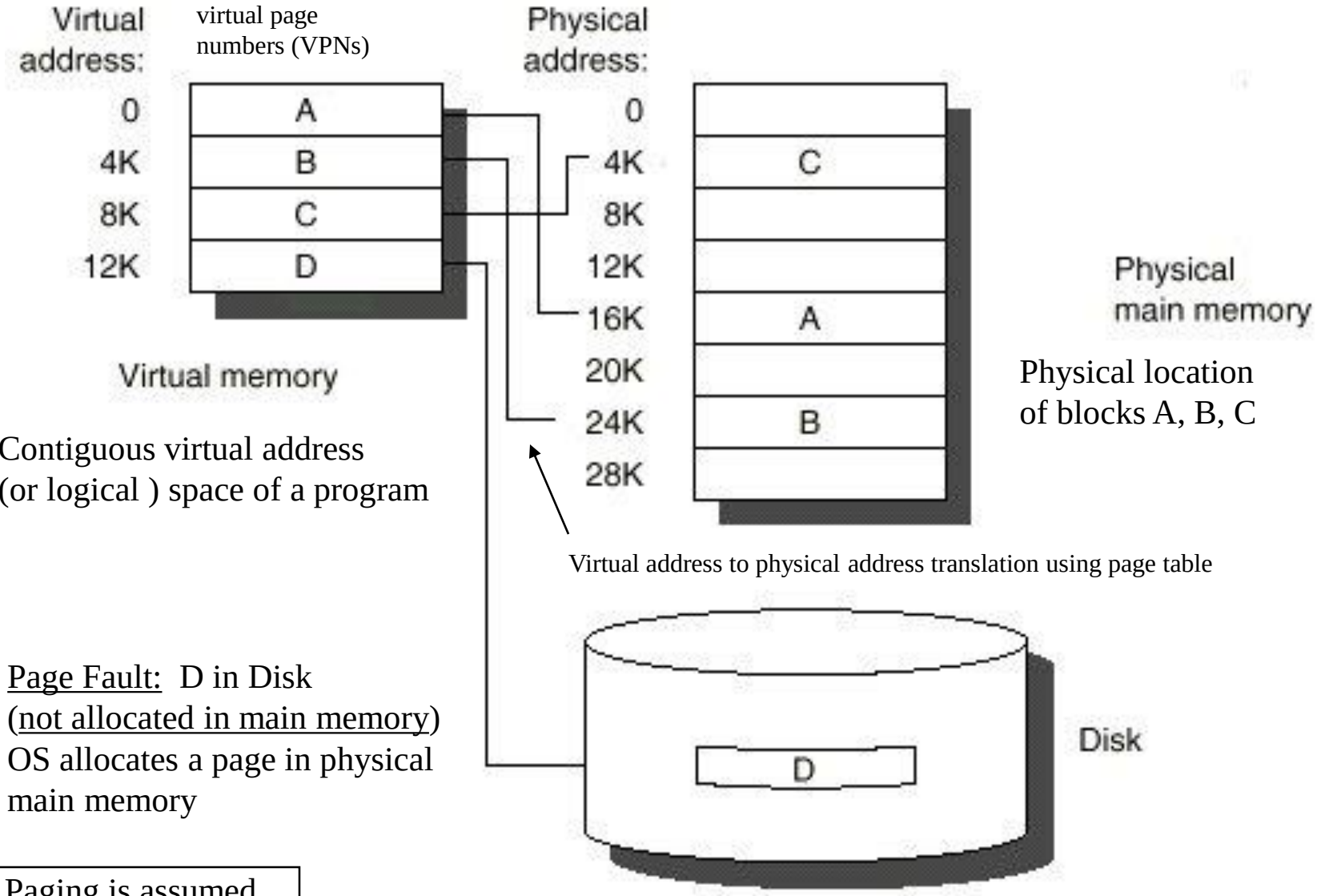
Figure B.20 Typical ranges of parameters for caches and virtual memory. Virtual memory parameters represent increases of 10 to 1,000,000 times over cache parameters. Normally, first-level caches contain at most 1 MB of data, whereas physical memory contains 256 MB to 1 TB.

Virtual Memory Basic Strategies

- **Main memory page placement(allocation): Fully associative placement or allocation (by OS) is used to lower the miss rate.**
- **Page replacement: The least recently used (LRU) page is replaced when a new page is brought into main memory from disk.**
- **Write strategy: Write back is used and only those pages changed in main memory are written to disk (dirty bit scheme is used).**
- **Page Identification and address translation: To locate pages in main memory a page table is utilized to translate virtual page numbers (VPNs) to physical page numbers (PPNs) . The page table is indexed by the virtual page number and contains the physical address of the page.**
  - **In paging: Offset is concatenated to this physical page address.**
  - **In segmentation: Offset is added to the physical segment address.**
- **Utilizing address translation locality, a translation look-aside buffer (TLB) is usually used to cache recent address translations (PTEs) and prevent a second memory access to read the page table.**

PTE = Page Table Entry

Virtual → Physical Address Translation

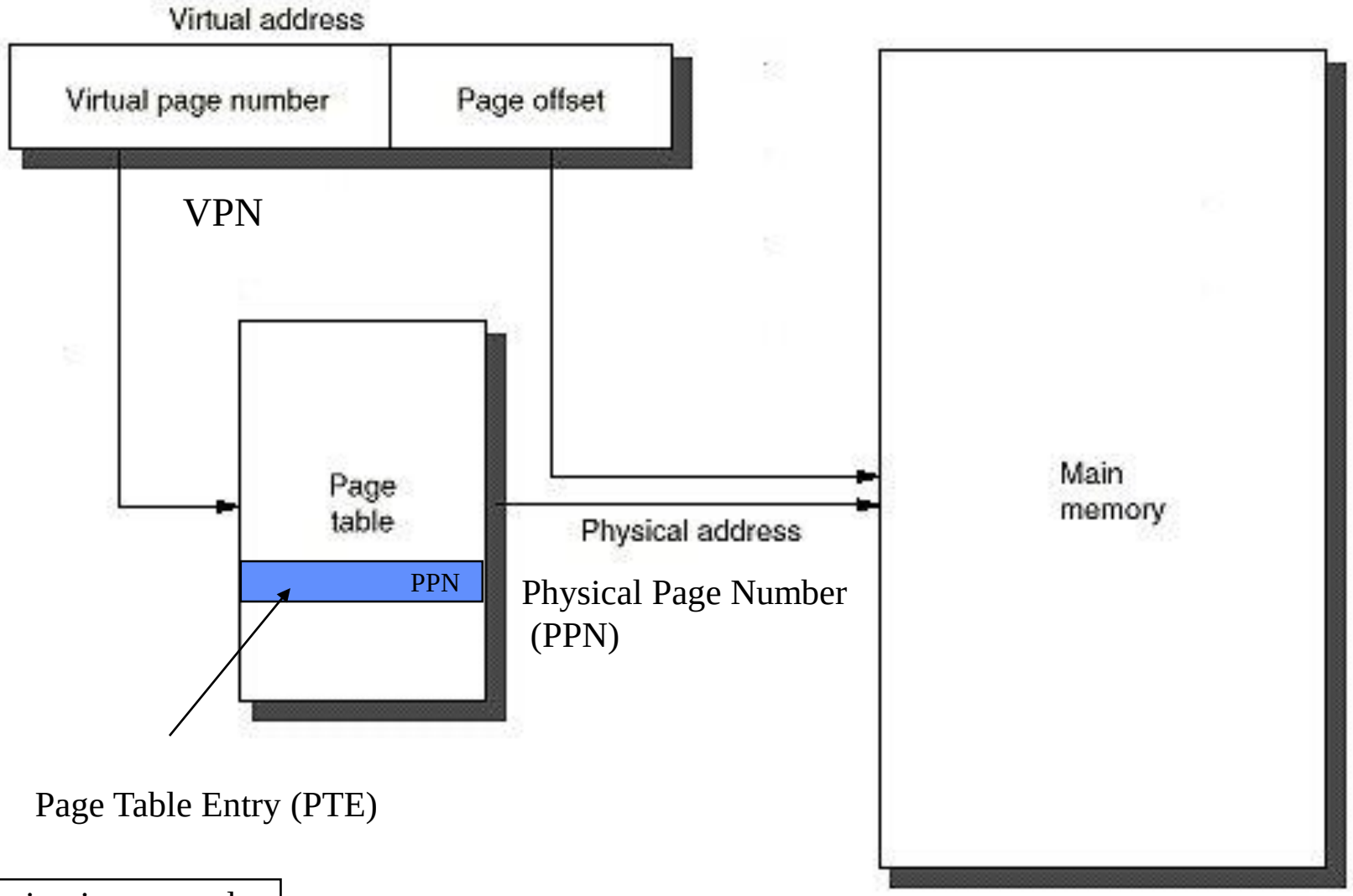


Contiguous virtual address (or logical) space of a program

Page Fault: D in Disk (not allocated in main memory)
OS allocates a page in physical main memory

Paging is assumed

Basic Mapping of Virtual Addresses to Physical Addresses Using A Direct Page Table

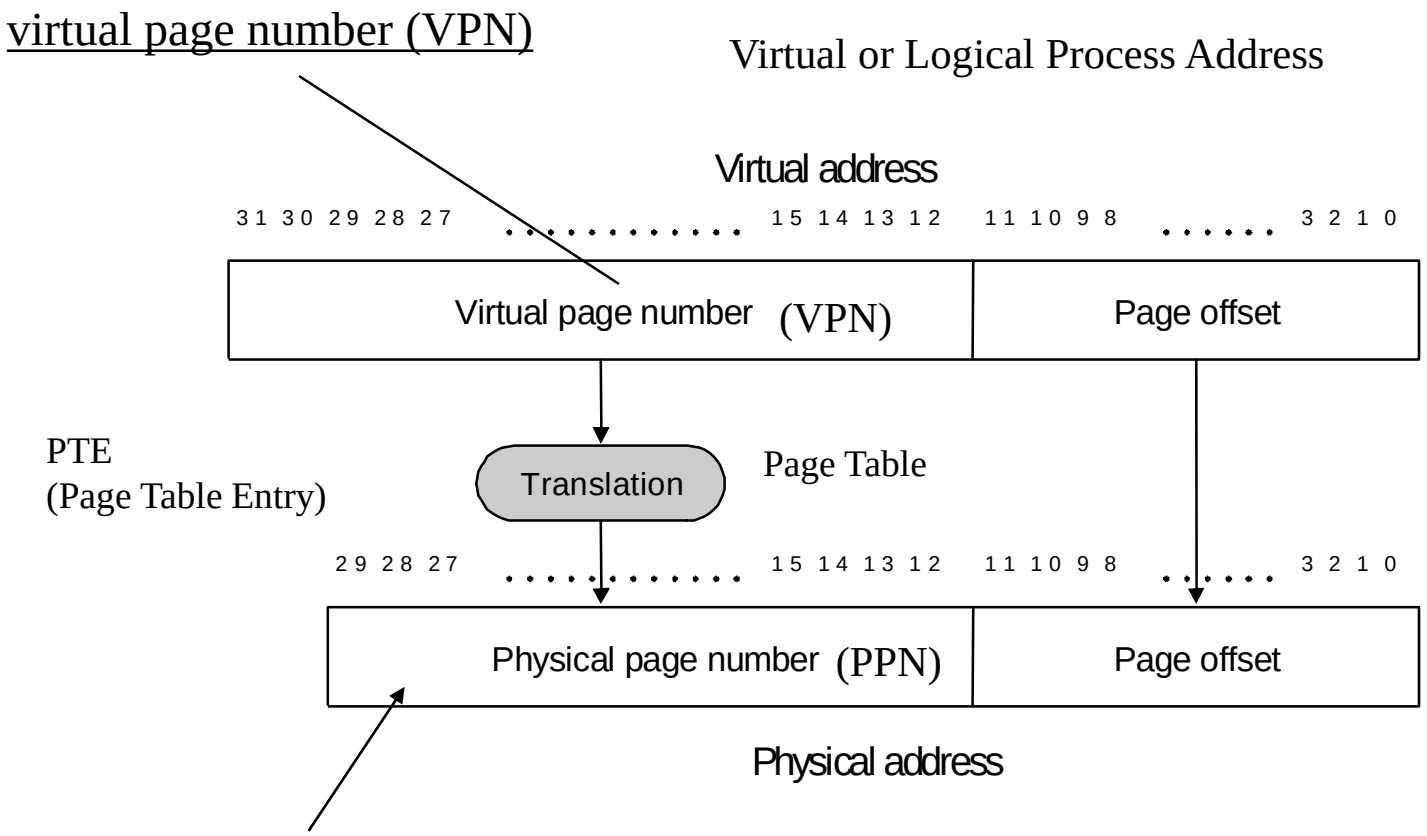


Paging is assumed

Virtual to Physical Address Translation: Page Tables

- **Mapping information from virtual page numbers (VPNs) to physical page numbers is organized into a page table which is a collection of page table entries (PTEs).**
- **At the minimum, a PTE indicates whether its virtual page is in memory, on disk, or unallocated and the PPN (or PFN) if the page is allocated.**
- **Over time, virtual memory evolved to handle additional functions including data sharing, address-space protection and page level protection, so a typical PTE now contains additional information including:**
  - **A *valid* bit, which indicates whether the PTE contains a valid translation;**
  - **The page's location in memory (page frame number, PFN) or location on disk (for example, an offset into a swap file);**
  - **The ID of the page's owner (the *address-space identifier (ASID)*, sometimes called Address Space Number (ASN) or *access key*;**
  - **The virtual page number (VPN);**
  - **A *reference* bit, which indicates whether the page was recently accessed;**
  - **A *modify* bit, which indicates whether the page was recently written; and**
  - ***Page-protection* bits, such as read-write, read only, kernel vs. user, and so on.**

Virtual to Physical Address Translation



physical page numbers (PPN) or page frame numbers (PFN)

Paging is assumed

Virtual Memory Terms

- **Page Table Walking:** The process of searching the page table for the translation PTE. Done by: Software (OS), Hardware (Finite State Machine)
- **Allocated or Mapped Virtual Page:** The OS has mapping information on its location (in memory or on disk) using its PTE in the page table.
- **Unmapped Virtual Page:** A page that has either not yet been allocated or has been deallocated and its mapping information (PTE) has been discarded.
- **Wired down virtual page:** A virtual page for which space is always allocated in physical memory and not allowed to be paged out to disk.
- **Virtual Address Aliasing:** Mapping of two or more virtual pages to the same physical page to allow processes or threads to share memory
- **Superpages:** A superpage contains a number of contiguous physical memory pages but require a single address translation. A number of virtual memory architectures currently support superpages.
- **Memory Management Unit (MMU):** Hardware mechanisms and structures to aid the operating system in virtual memory management including address generation/translation, sharing, protection. Special OS privileged ISA instructions provide software/OS access to this support. (e.g. TLBs, special protected registers)

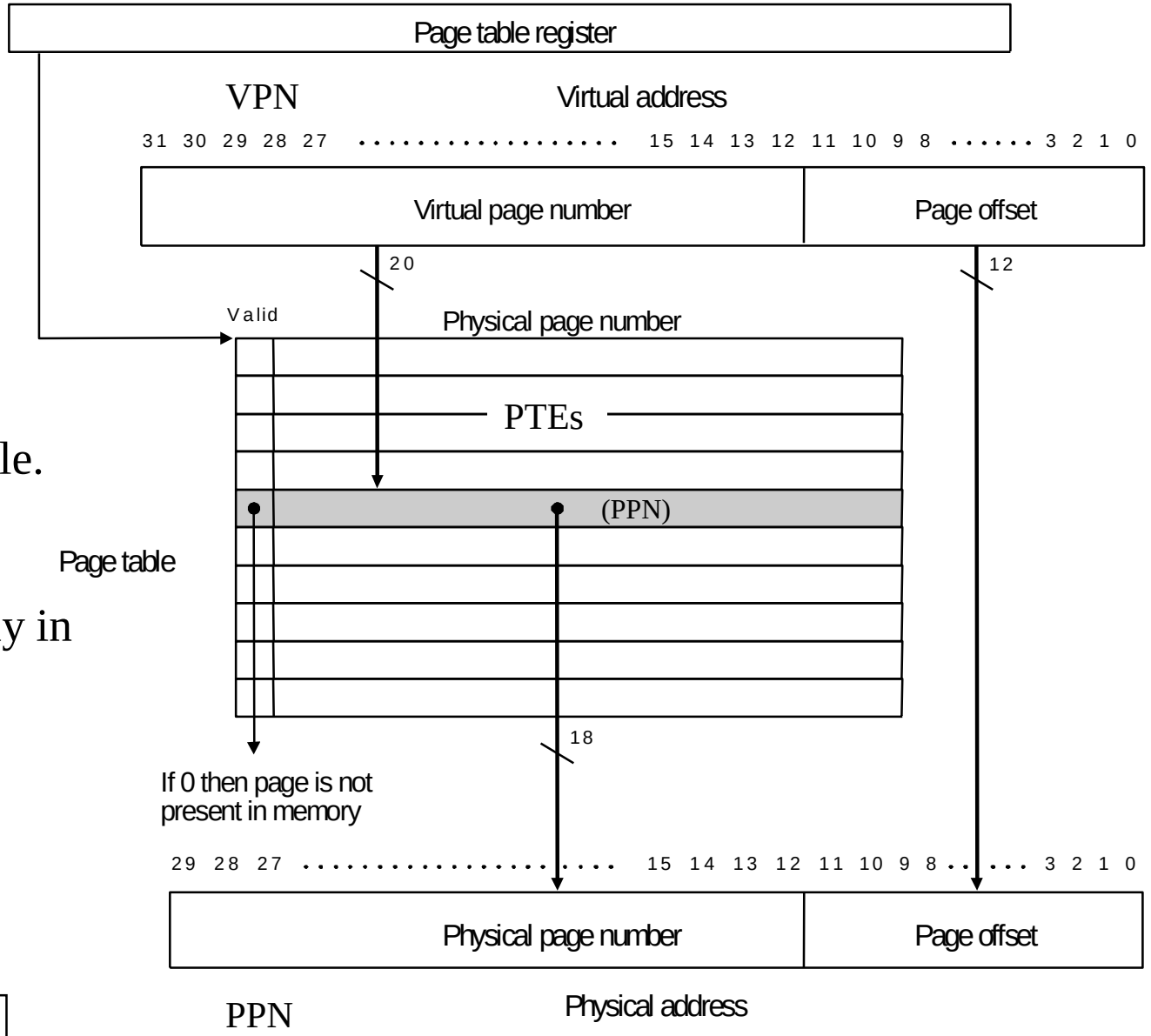
Page Table Organizations

Direct (Basic) Page Table:

- **When address spaces were much smaller, a single-level table—called a *direct table* mapped the entire virtual address space and was small enough to be contained in SRAM and maintained entirely in hardware (hardware page table walking).**
- **As address spaces grew larger, the table size grew to the point that system designers were forced to move it into main memory.**

- **Limitations:**
  - **Translation requires a main memory access:**
    - Solution: Speedup translation by caching recently used PTEs in **a Translation Lookaside Buffer (TLB).**
  - **Large size of direct table:**
    - Example: A 32 bit virtual address with 2^{12} = 4k byte pages and 4 byte PTE entries requires a direct page table with 2^{20} = 1M PTEs and occupies 4M bytes in memory.
    - **Solution: Alternative page table organizations:**
      - **Hierarchical page tables**
      - **Inverted or hashed page tables**

Direct Page Table Organization



Two memory accesses needed:

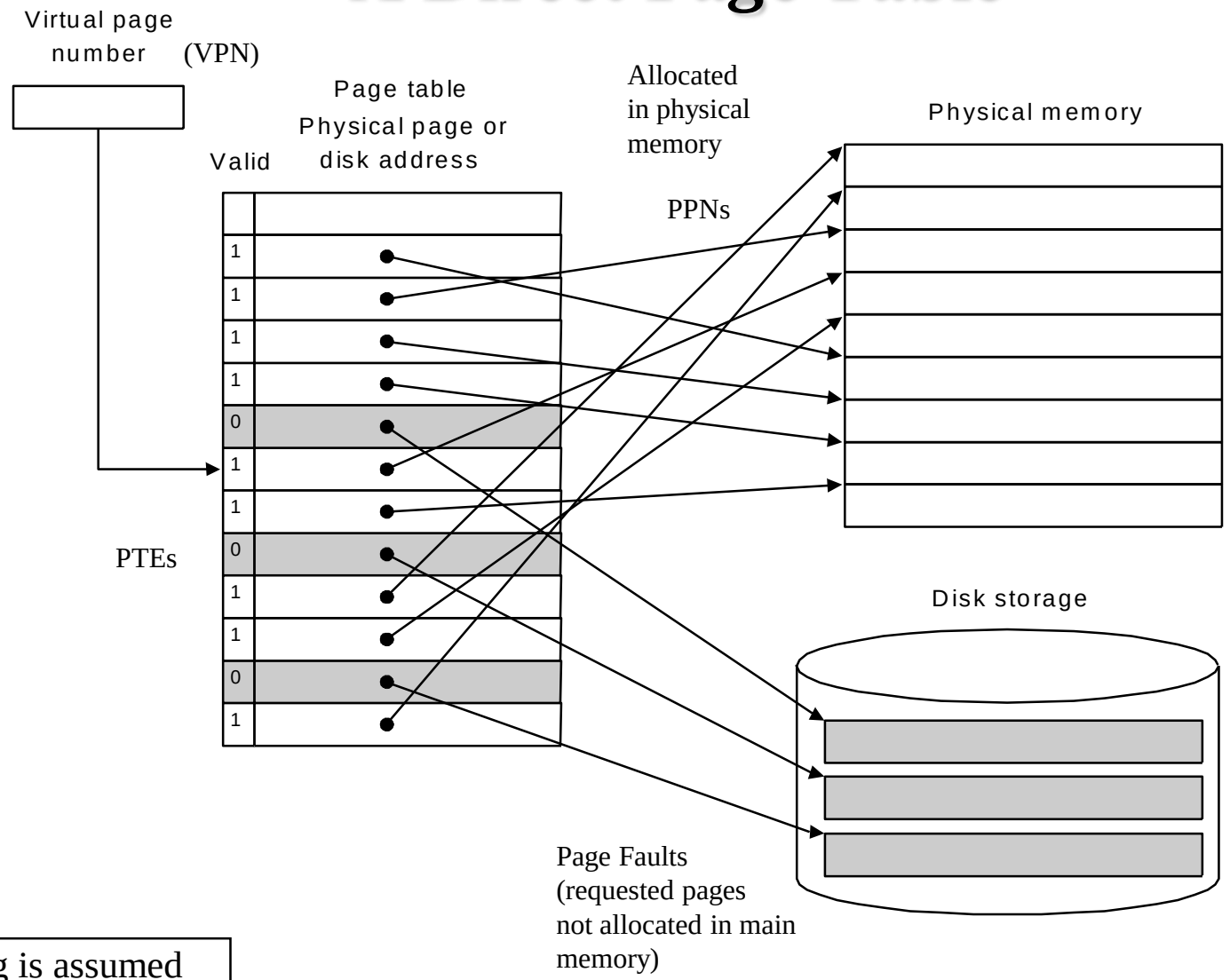
- First to page table.
- Second to item.

- Page table usually in main memory.

How to speedup virtual to physical address translation?

Paging is assumed

Virtual Address Translation Using A Direct Page Table



Paging is assumed

Speeding Up Address Translation:
Translation Lookaside Buffer (TLB)

- **Translation Lookaside Buffer (TLB) : Utilizing address reference temporal locality, a small on-chip cache used for address translations (PTEs).**
  - TLB entries usually 32-128
  - High degree of associativity usually used
  - Separate instruction TLB (I-TLB) and data TLB (D-TLB) are usually used.
  - A unified larger second level TLB is often used to improve TLB performance and to reduce the associativity of level 1 TLBs.

- **If a virtual address is found in TLB (a TLB hit), the page table in main memory is not accessed.**
- **TLB-Refill: If a virtual address is not found in TLB, a TLB miss occurs and the system must search (walk) the page table for the appropriate entry and place it into the TLB, which is accomplished by the TLB-refill mechanism.**
- **Types of TLB-refill mechanisms:**
  - **Hardware-managed TLB:** A hardware finite state machine is used to refill the TLB on a TLB miss by walking the page table. (PowerPC, IA-32)

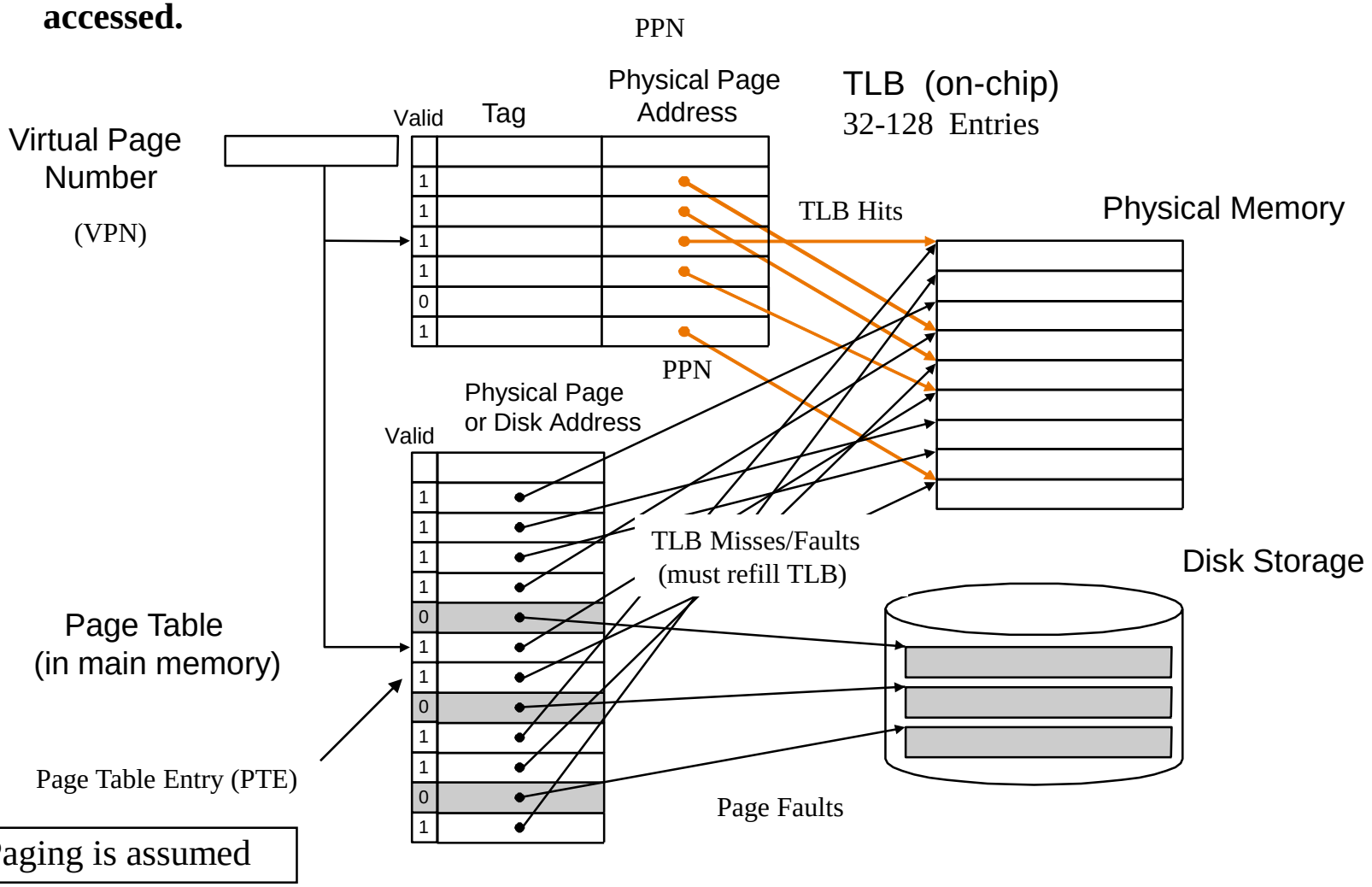
    Fast but not flexible
  - **Software-managed TLB:** TLB refill handled by the operating system. (MIPS, Alpha, UltraSPARC, HP PA-RISC, …)

    Flexible but slow

Speeding Up Address Translation:

Translation Lookaside Buffer (TLB)

- **TLB: A small on-chip cache that contains recent address translations (PTEs).**
- **If a virtual address is found in TLB (a TLB hit), the page table in main memory is not accessed.**



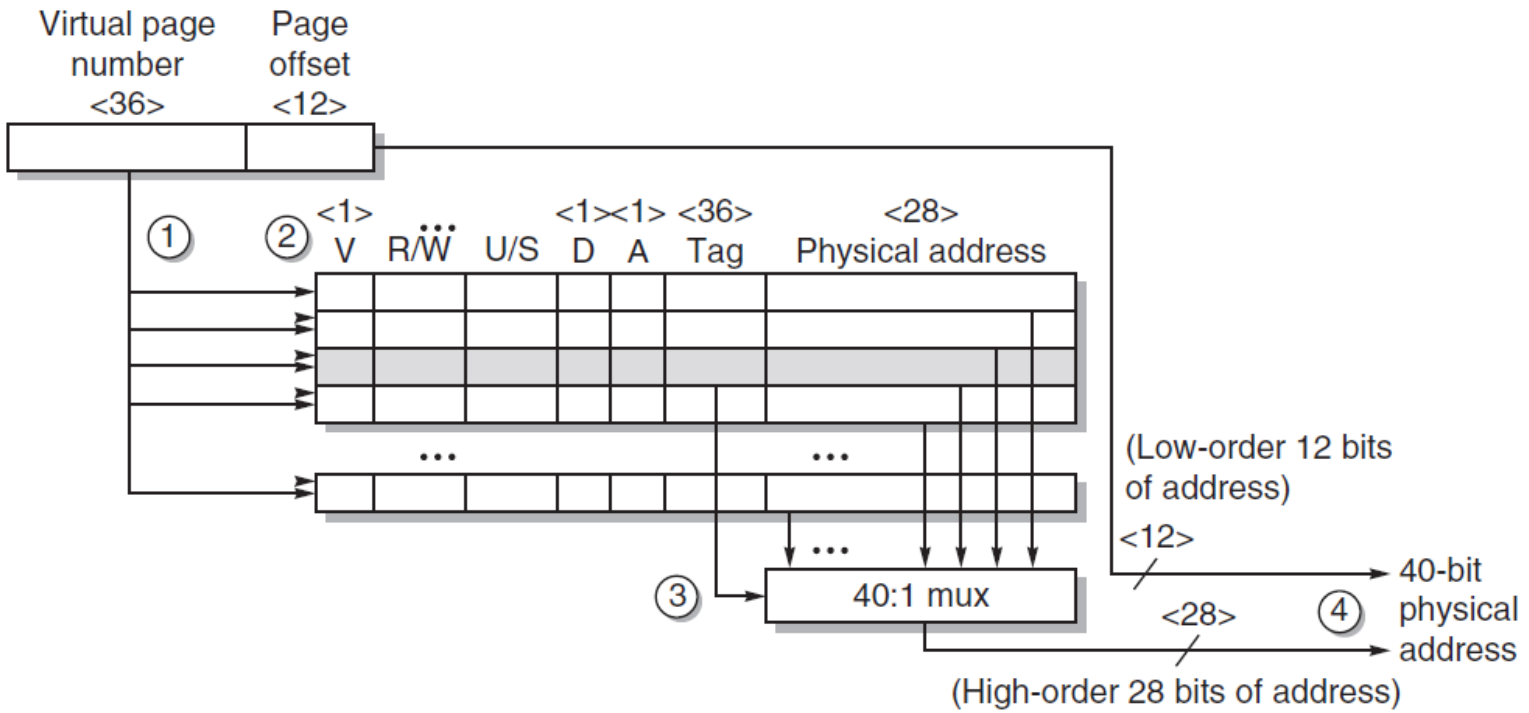


Figure B.24 Operation of the Opteron data TLB during address translation. The four steps of a TLB hit are shown as circled numbers. This TLB has 40 entries. Section B.5 describes the various protection and access fields of an Opteron page table entry.

Opteron:
L1 Data Cache: Virtually Indexed, Physically Tagged

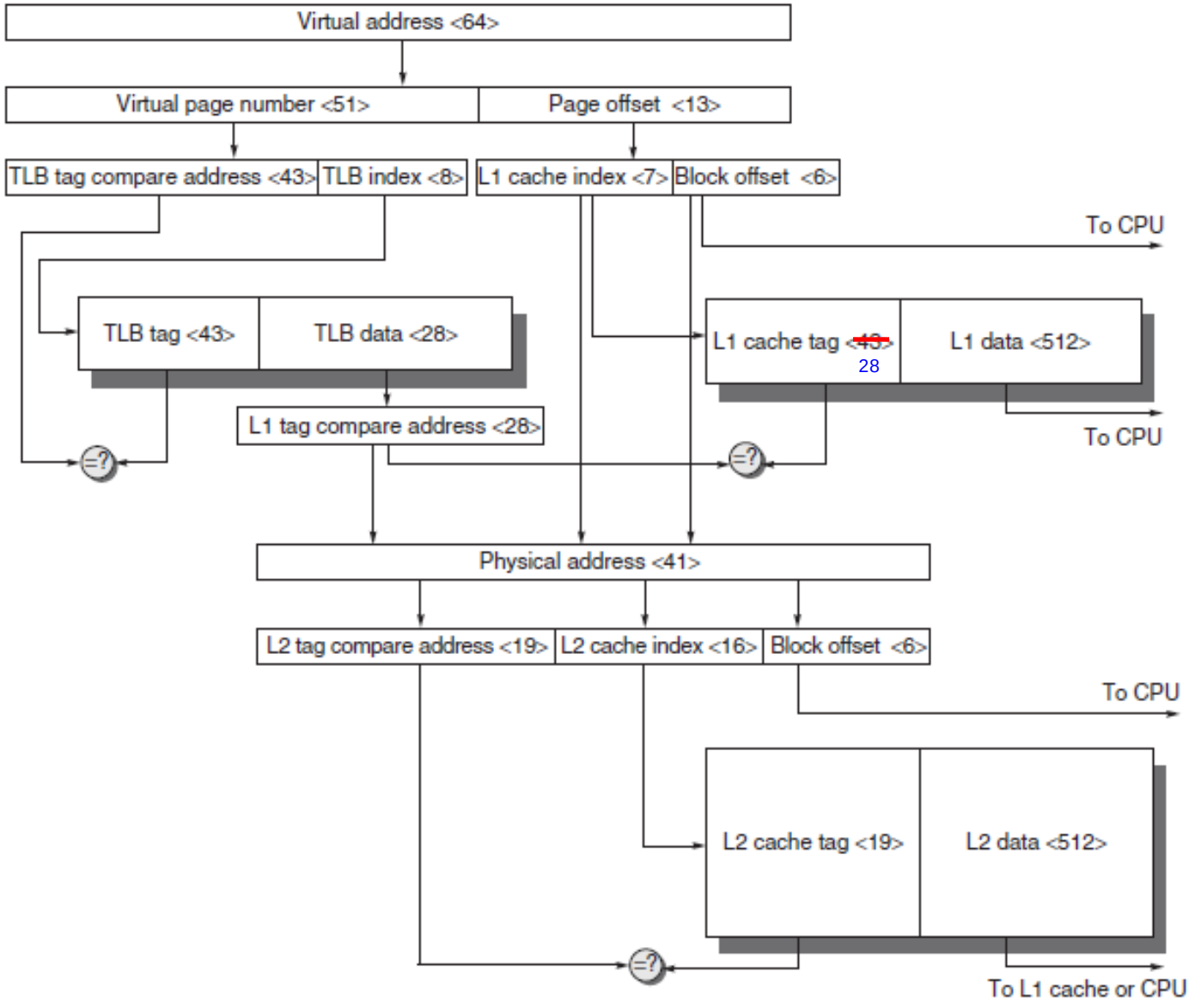


Figure B.25 The overall picture of a hypothetical memory hierarchy going from virtual address to L2 cache access. The page size is 8 KB. The TLB is direct mapped with 256 entries. The L1 cache is a direct-mapped 8 KB, and the L2 cache is a direct-mapped 4 MB. Both use 64-byte blocks. The virtual address is 64 bits and the physical address is 41 bits. The primary difference between this simple figure and a real cache is replication of pieces of this figure.