

Computer Organization & Architecture

Chapter 8 – Virtual Memory

Zhang Yang 张杨

cszyang@scut.edu.cn

Autumn 2025

Contents of this lecture

■ 8.8 Virtual Memory

- What is Virtual Memory?
- Motivation for Virtual Memory
- Terminology of Virtual Memory
- Implementation of Virtual Memory
- Paging
- Summary

What is Virtual Memory?

- If you think it's there, and it's there...it's *real*.
- If you think it's not there, and it's there...it's *transparent*.
- If you think it's there, and it's not there...it's *imaginary*.
- If you think it's not there, and it's not there...it's *nonexistent*.
- *Virtual memory* is imaginary memory: it gives you the illusion of a memory arrangement that's not physically there.

Motivations for Virtual Memory

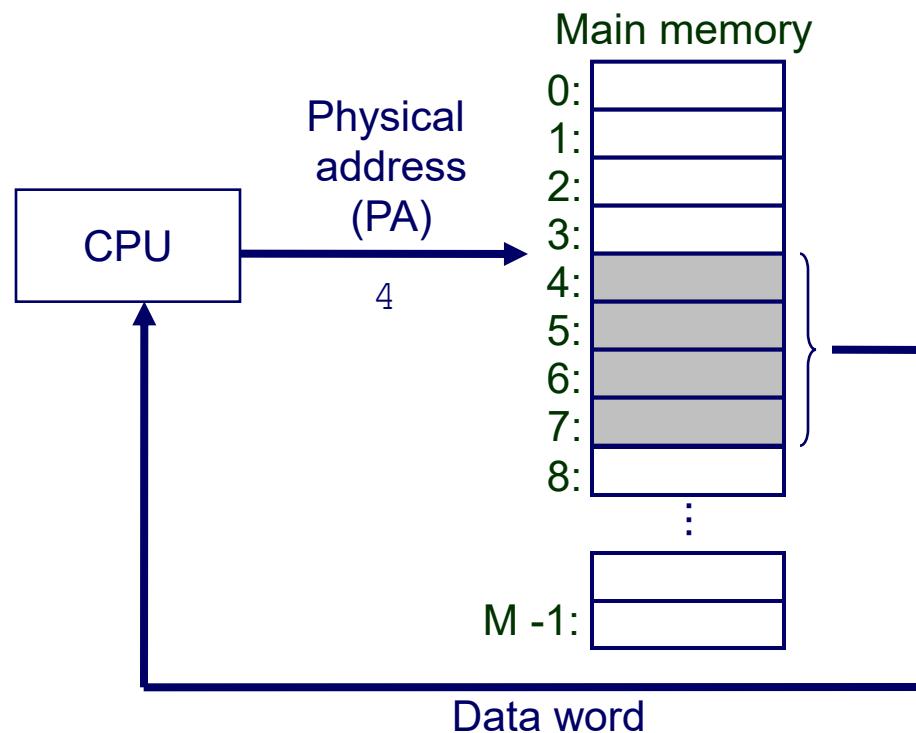
- Use physical DRAM as cache for the disk.
 - Address space of a process can exceed physical memory size.
 - Sum of address spaces of multiple processes can exceed physical memory (more common modern case).
- Simplify memory management.
 - Multiple processes resident in main memory.
 - Each with its own address space.
 - Only “active” code and data is actually in memory.
 - Allocate more memory to process as needed.
- Provide protection.
 - One process can't interfere with another.
 - Because they operate in different address spaces.
 - User process cannot access privileged information.
 - Different sections of address spaces have different permissions.

Terminology of VM

- Physical memory – memory as seen by the processor
- Logical memory – memory as seen by the process
- Physical Address – real location in physical memory; identifies actual storage
- Virtual Address – conventional addressing used by a program which the OS must translate into a physical address

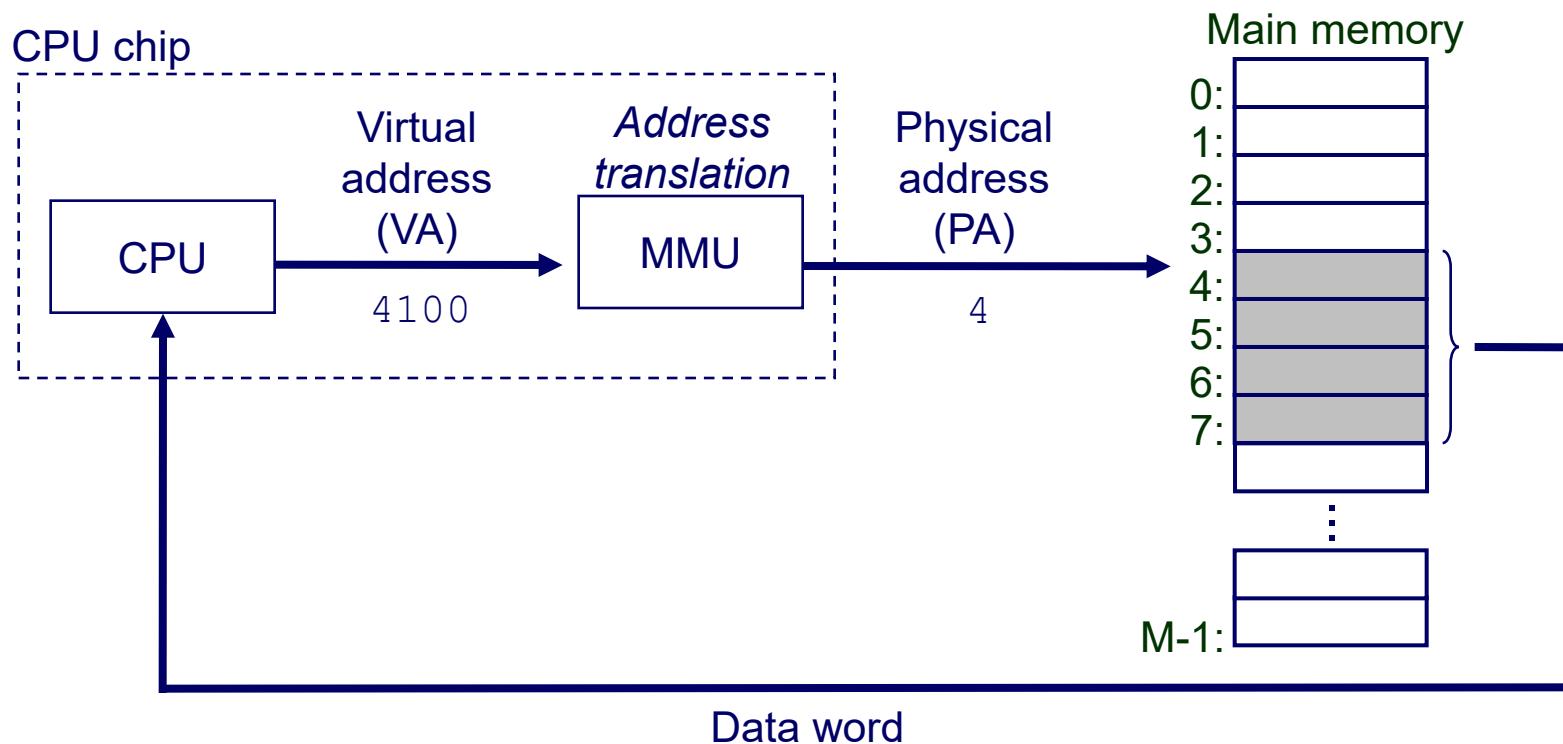
A System Using Physical Addressing

- Used by many digital signal processors and embedded microcontrollers in devices like phones and PDAs.



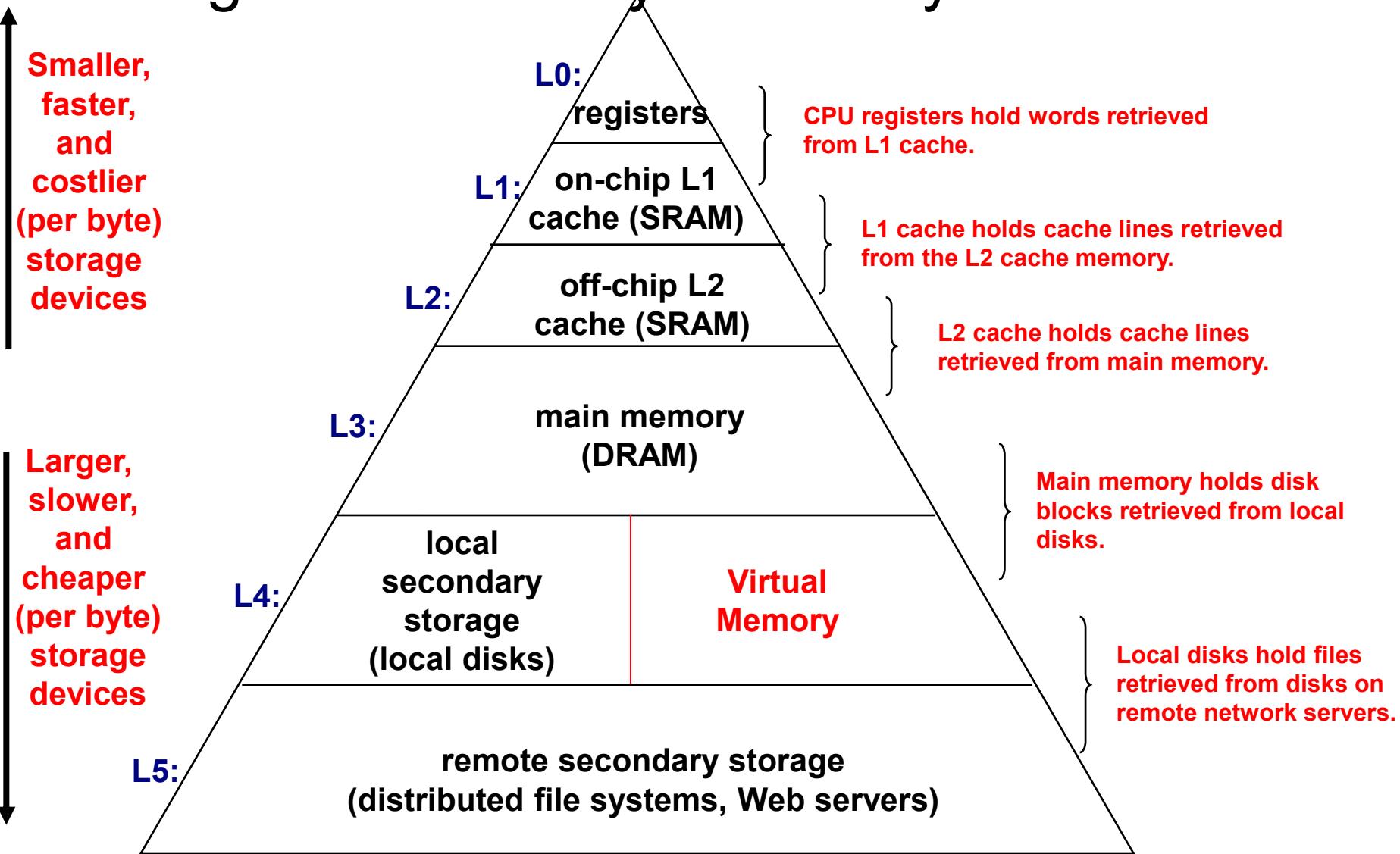
A System Using Virtual Addressing

- Used by all modern desktop and laptop microprocessors.



Recall: Memory Hierarchy

■ A Figure of Memory Hierarchy

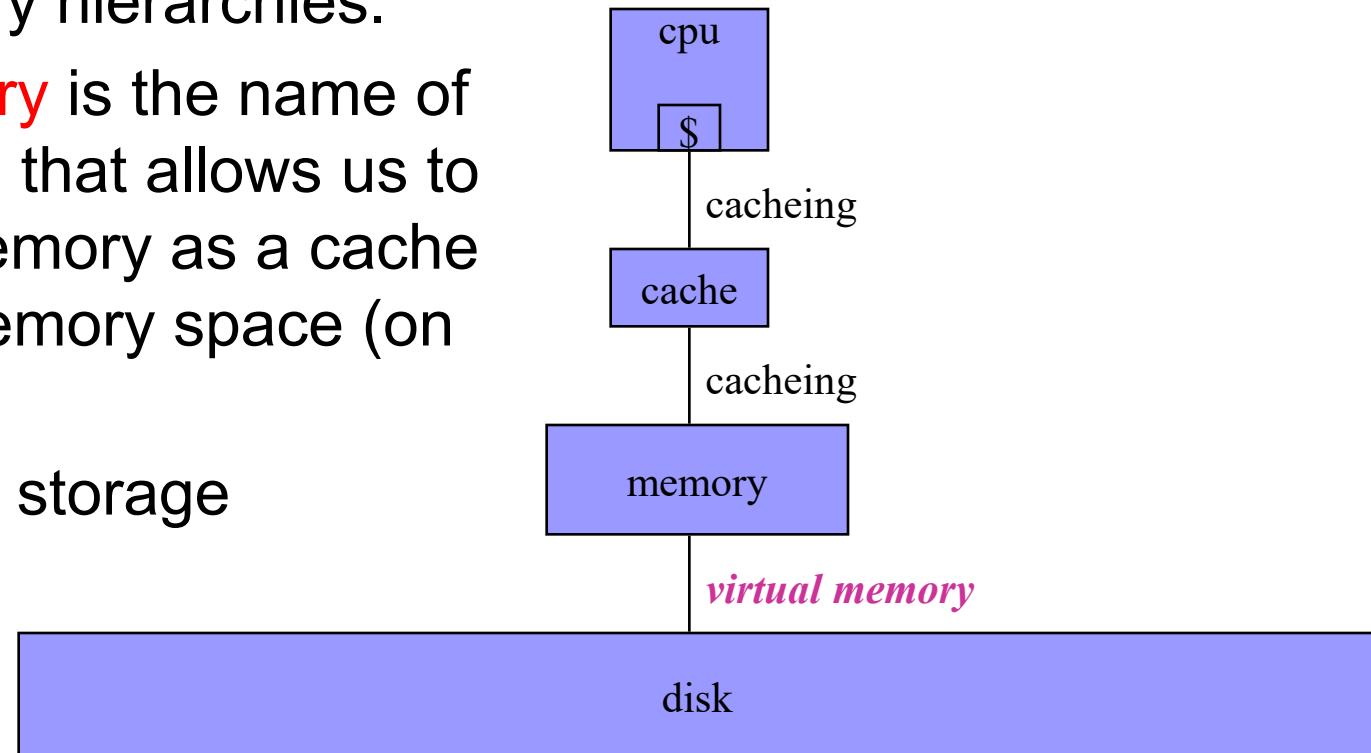


Why Use Disks?

- Multiprogramming
 - More than one application running at one time.
 - May need more than available main memory to store all needed programs and data.
 - May want to share data between applications.
- Cheap, large (but, slow)

VM Definition

- Not a physical device but an abstract concept.
- It's just another level in the cache/memory hierarchies.
- **Virtual memory** is the name of the technique that allows us to view main memory as a cache of a larger memory space (on disk).
 - Automatic storage allocation



Typical Organization of VM

- MMU
 - Memory Management Unit
- OS
 - Control MMU
- Address Translation
 - The hardware converts virtual addresses into physical addresses via an OS-managed lookup table (page table)

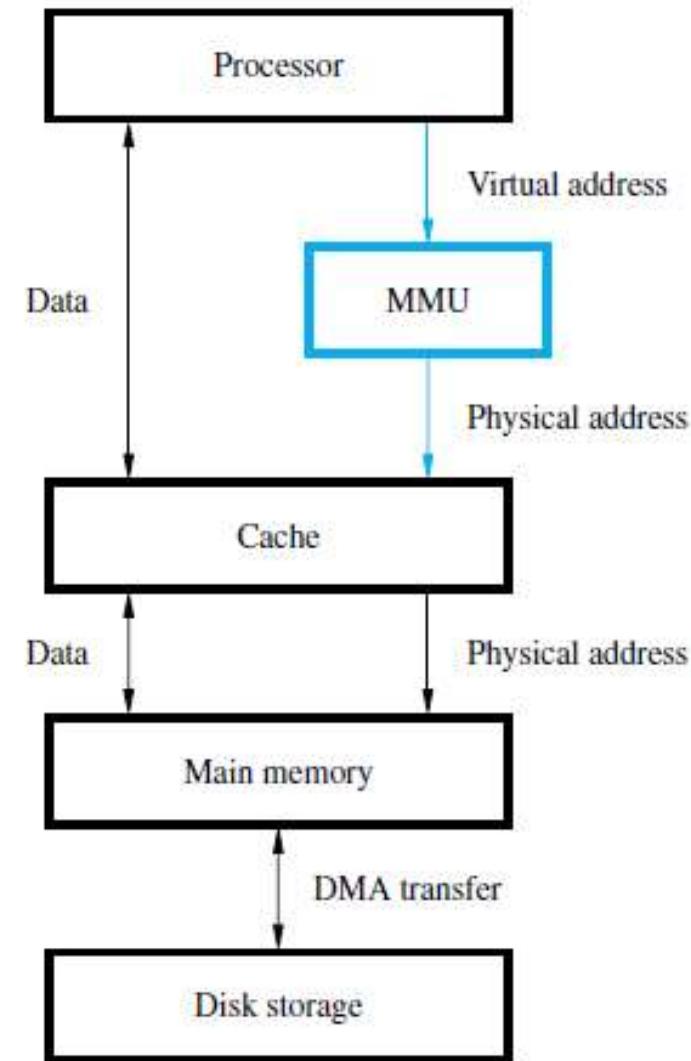


Figure 8.24 Virtual memory organization.

Benefits of VM

- **Large address space:** Virtual address to Physical Address mapping.
- **Simplify relocation:** the same program can run in any location in the physical memory.
- **Reduce time to start a program:** not all code and data need to be in the physical memory before a program can start.
- **Relieve programmers from burden of overlays:** VM automatically manages the two levels of memory hierarchy represented by main memory and secondary storage.

Implementation of VM

■ Three Implementations

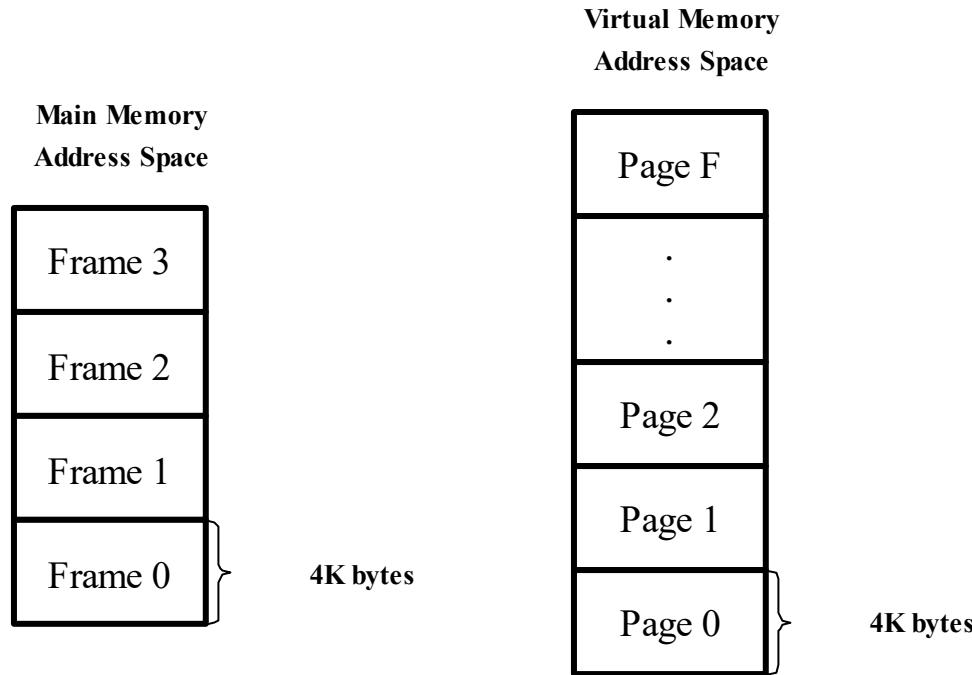
- Paging
- Segmentation (discussed in operating system)
- Segmentation with Paging (discussed in operating system)

Paging

- Address Translation
- Page Replacement
- Write Policy
- Internal Fragmentation
- Page Size

Paging (1)

■ Organization of Virtual Memory and Main Memory



- Page: a block of consecutive words
- Page Frame: an area in the main memory that can hold one page

Paging (2)

■ Organization of Virtual Memory and Main Memory (ctd.)

□ Note

- Pages constitute the basic unit of information that is moved between the main memory and the disk.
- Pages commonly range from 2K to 16K bytes in length.
- Pages should not be too small or too large.

Address Structure

■ Virtual Address (2 fields)

- Virtual page number (high-order bits)
- Offset (low-order bits)
 - Specify the location of a particular byte(or word) within a page.

■ Physical Address (2 fields)

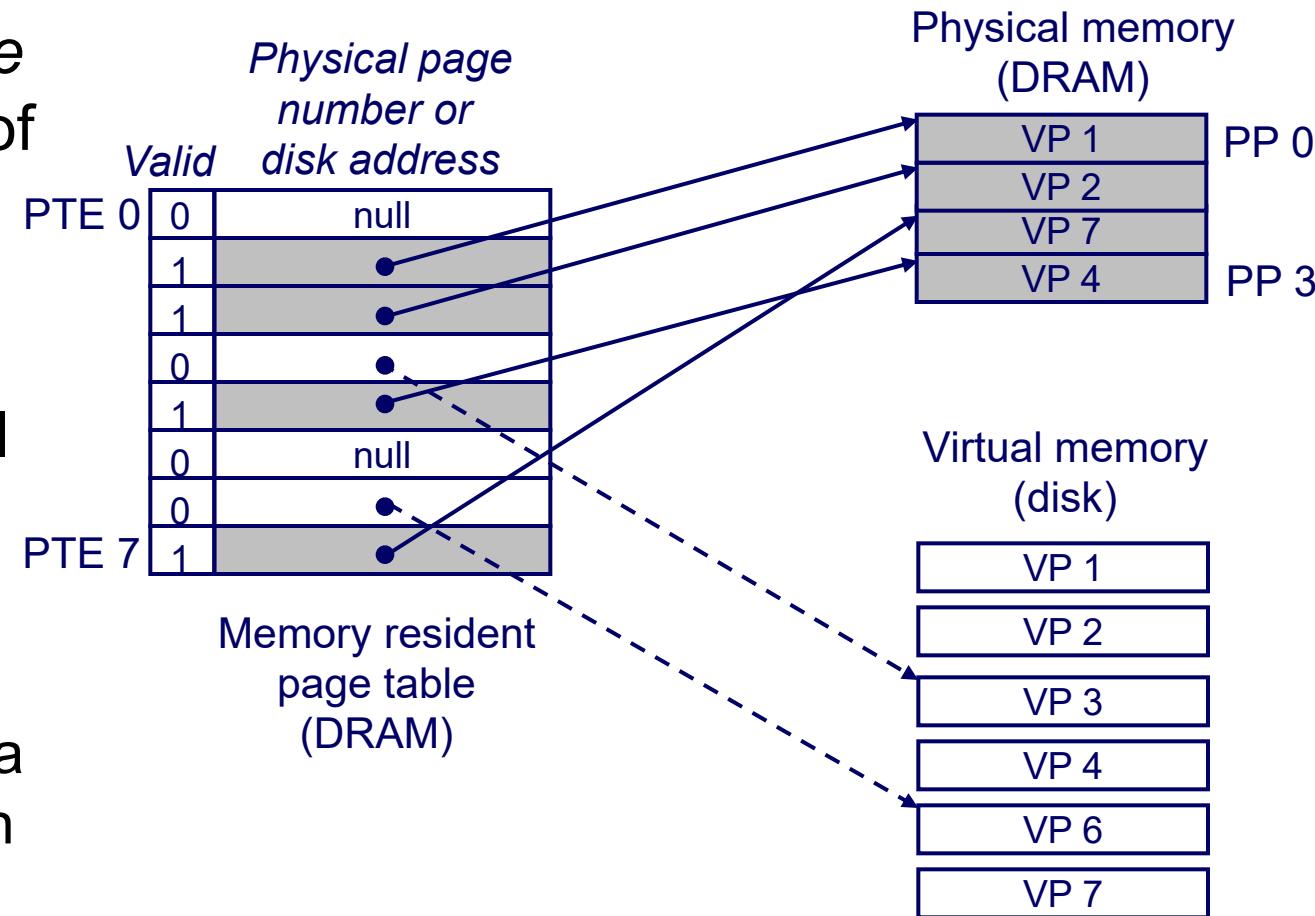
- Physical page number (high-order bits)
- Offset

Page Table

■ Page Table

- A *page table* is an array of page table entries (PTEs) that maps virtual pages to physical pages.

- Kernel data structure in DRAM



Page Table Entry

■ Access Control Bit

- Read, Write, Execute, etc

■ Present Bit (Valid Bit)

- present in main memory (or not)

■ Dirty Bit

- If the page has been modified

■ Use Bit

Access	Present	Dirty	Use	Page Number
--------	---------	-------	-----	-------------

- Recently used?

■ Page Number

- physical page number OR pointer to secondary storage

Address Translation (1)

- Translate Virtual Memory Address to Physical Address

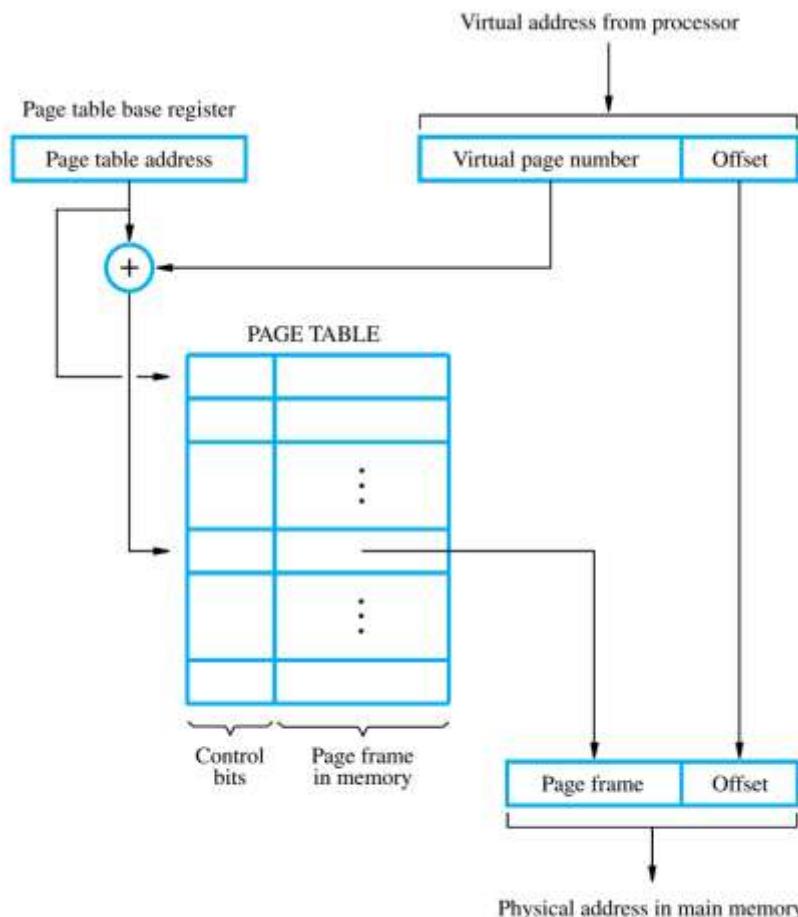
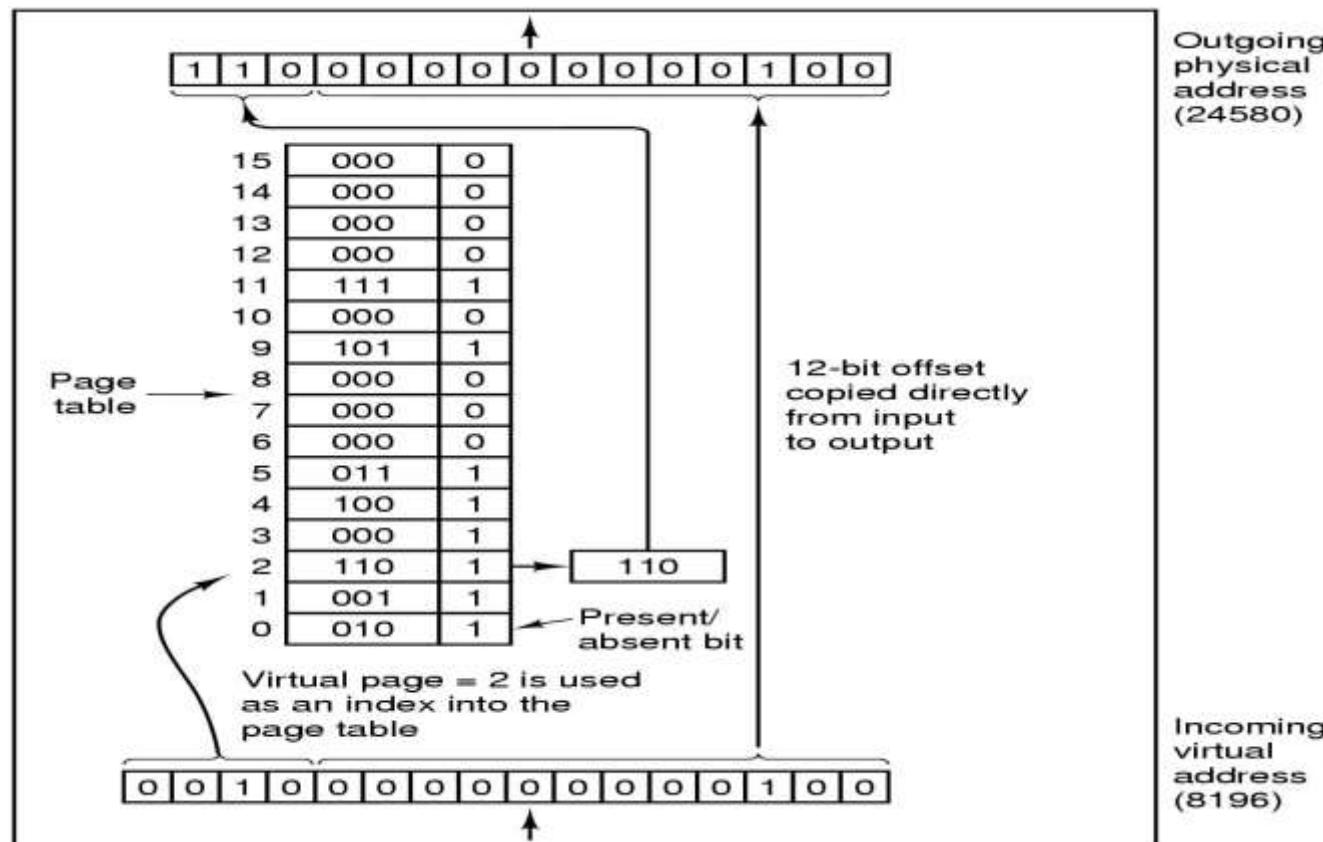


Figure 8.25 Virtual-memory address translation.

Address Translation (2)

■ Address Translation Example

- Page size: 4KB



Page Hit and Page Fault (1)

■ Page Hit

- A *page hit* is a reference to a VM word that is in physical (main) memory.

■ Page Fault (Similar to “Cache Miss”)

- A *page fault* is caused by a reference to a VM word that is not in physical (main) memory.
- Example: A instruction references a word contained in VP 3, a miss that triggers a page fault exception.
- What if an object is on disk rather than in memory?
 - Page table indicates that the virtual address is not in memory.
 - OS trap handler is invoked, moving data from disk into memory.
 - Current process suspends, others can resume.
 - OS has full control over placement, etc.

Page Hit and Page Fault (2)

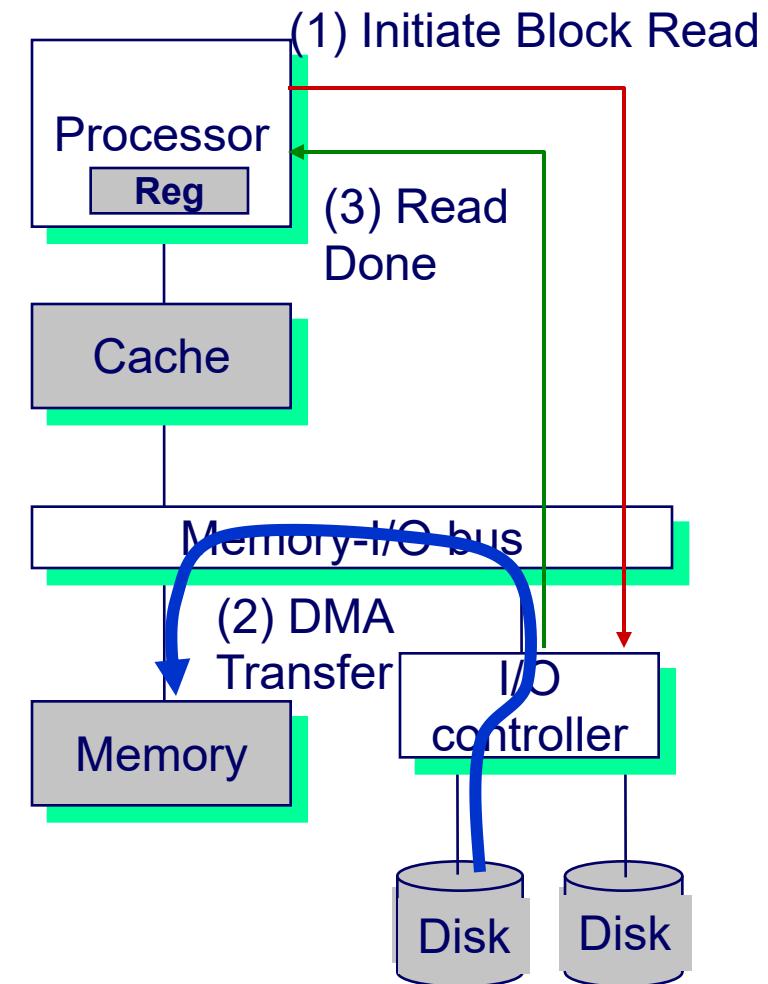
■ Servicing a Page Fault

- Make space in memory by writing physical page to disk.
 - Page Frames
 - Replacement policy?
- Load page
 - Loading pages could waste processor time, use DMA.
 - DMA allows processor to do something else.
- OS updates the process's page table.
 - Desired data is in memory for process to resume.

Page Hit and Page Fault (3)

■ Servicing a Page Fault (ctd.)

- (1) Processor signals disk controller.
 - Read block of length P starting at disk address X and store starting at memory address Y.
- (2) Read occurs
 - Direct Memory Access (DMA)
 - Under control of I/O controller
- (3) Controller signals completion
 - Interrupts processor.
 - OS resumes suspended process.



TLB (1)

■ Disadvantage of Page Table

- Every memory access by a program can take at least twice as long: one memory access to obtain the physical address and a second access to get the data.

■ TLB

- A small cache which is incorporated in MMU consists of a small portion of the page table.
- The contents of the TLB must be coherent with the contents of page tables in the main memory.

TLB (2)

■ Address Translation with TLB

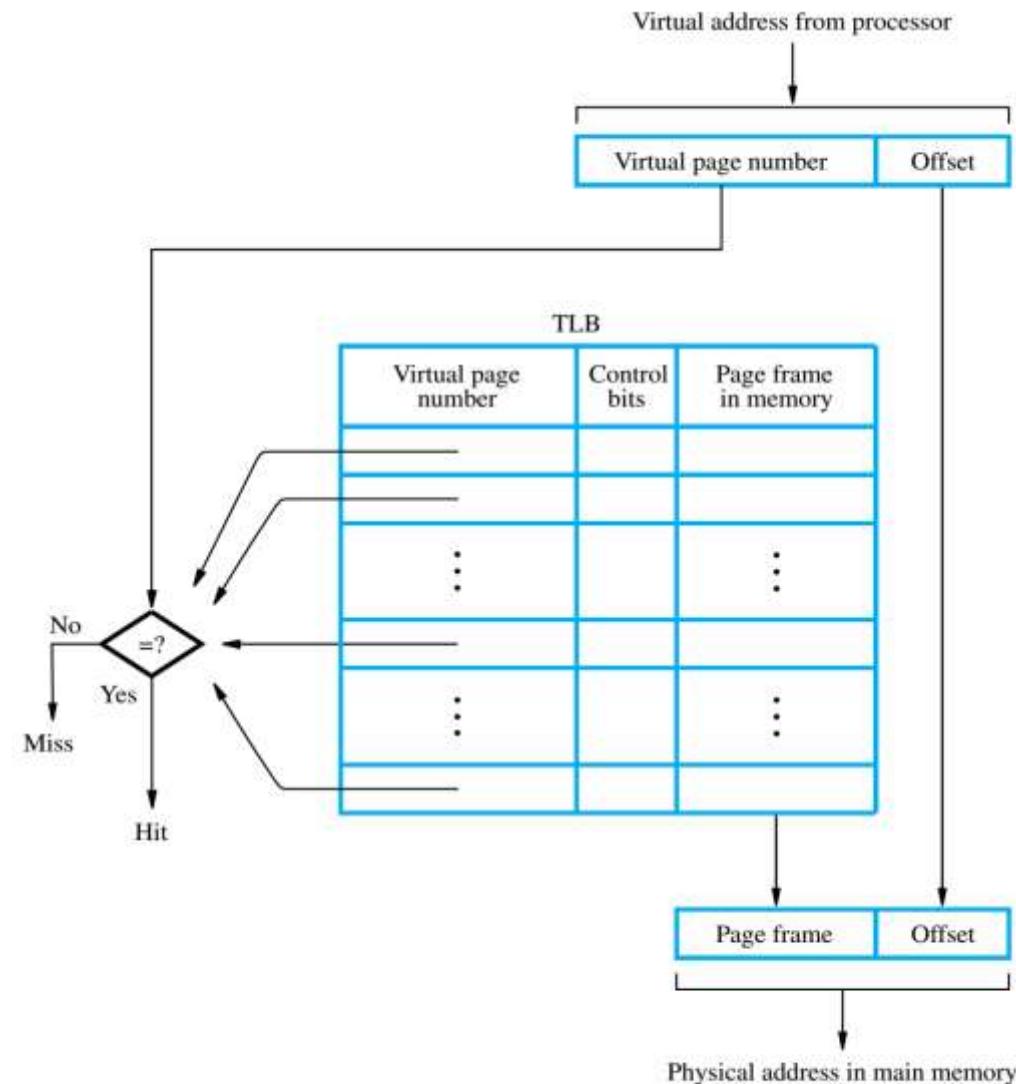
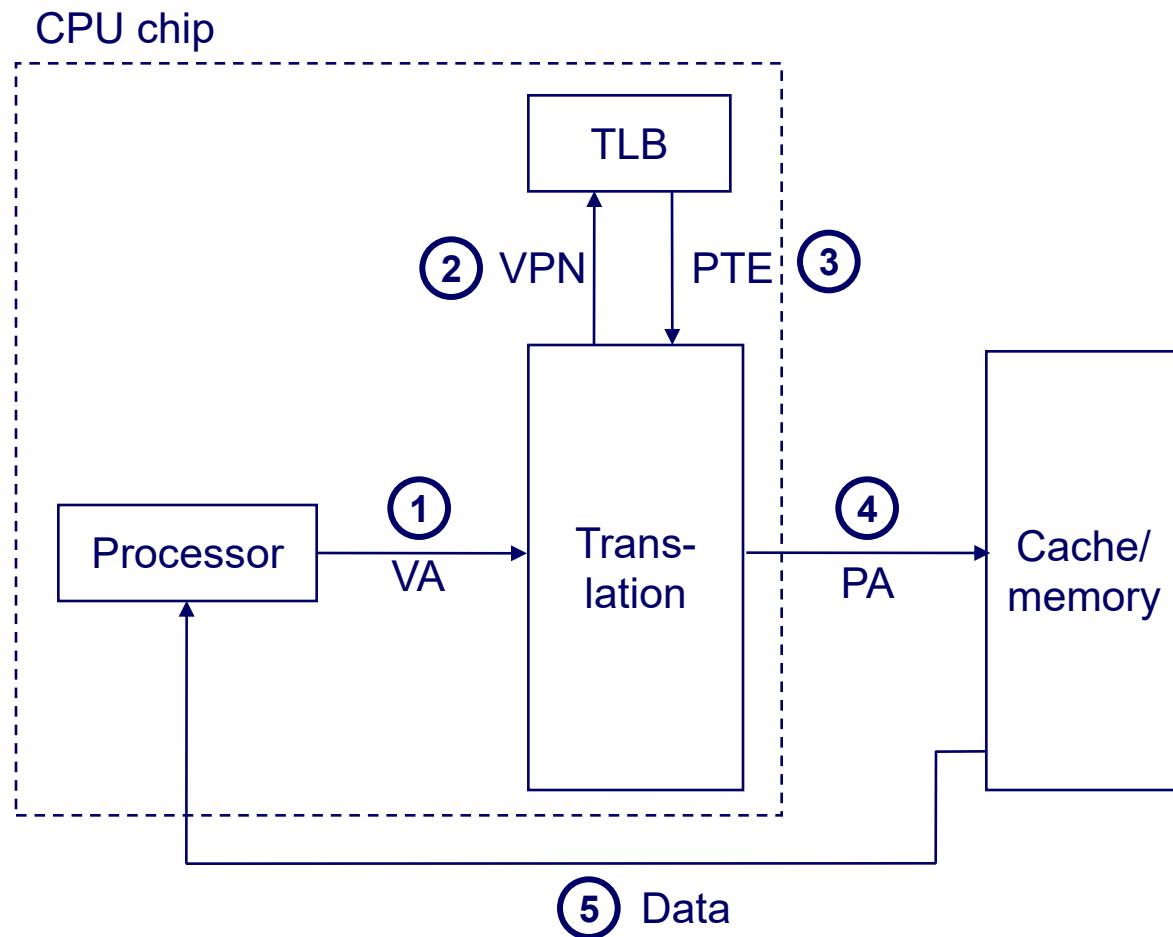


Figure 8.26 Use of an associative-mapped TLB.

TLB (3)

■ TLB Hit

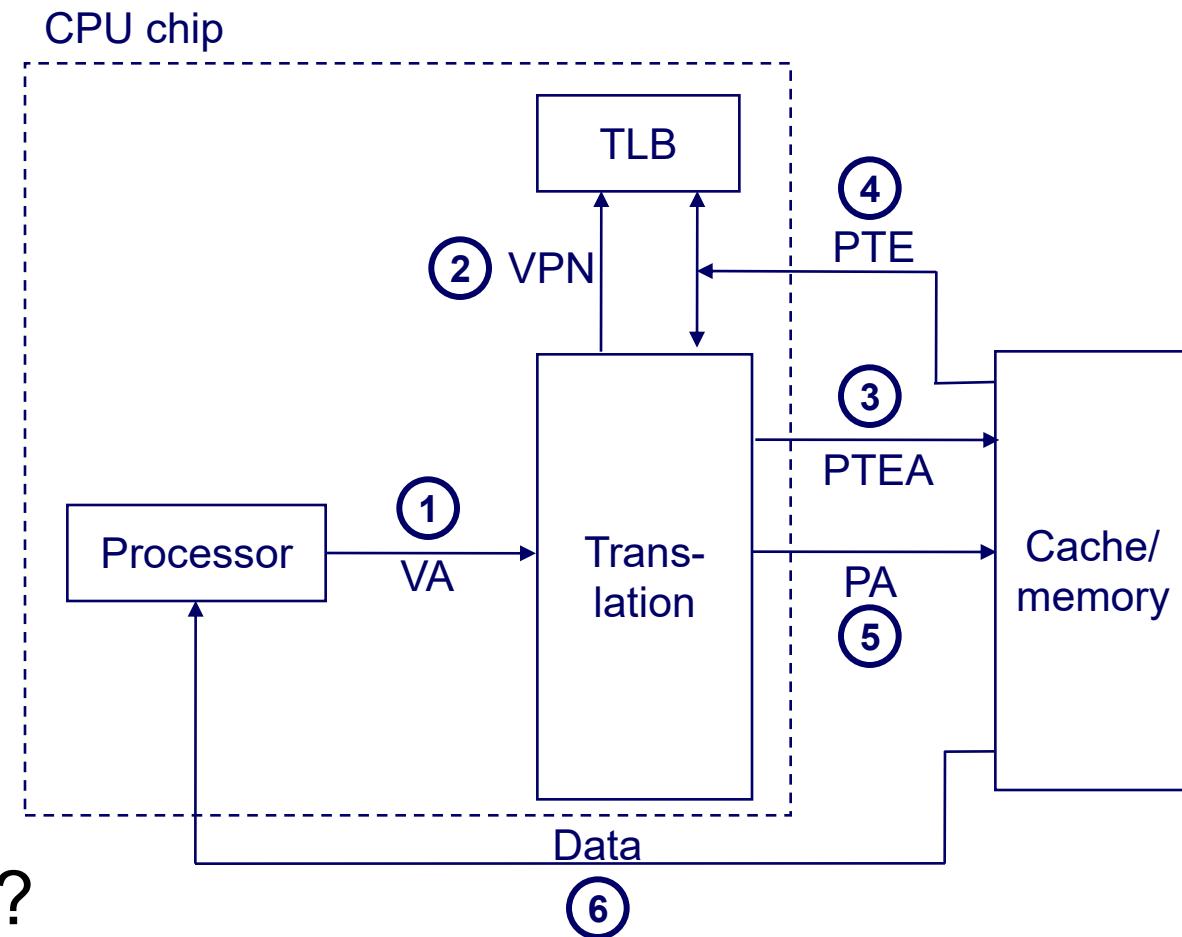
- A TLB hit eliminates a memory access.



TLB (4)

■ TLB Miss

- A TLB miss incurs an additional memory access (the PTE).
- Fortunately, TLB misses are rare. Why?



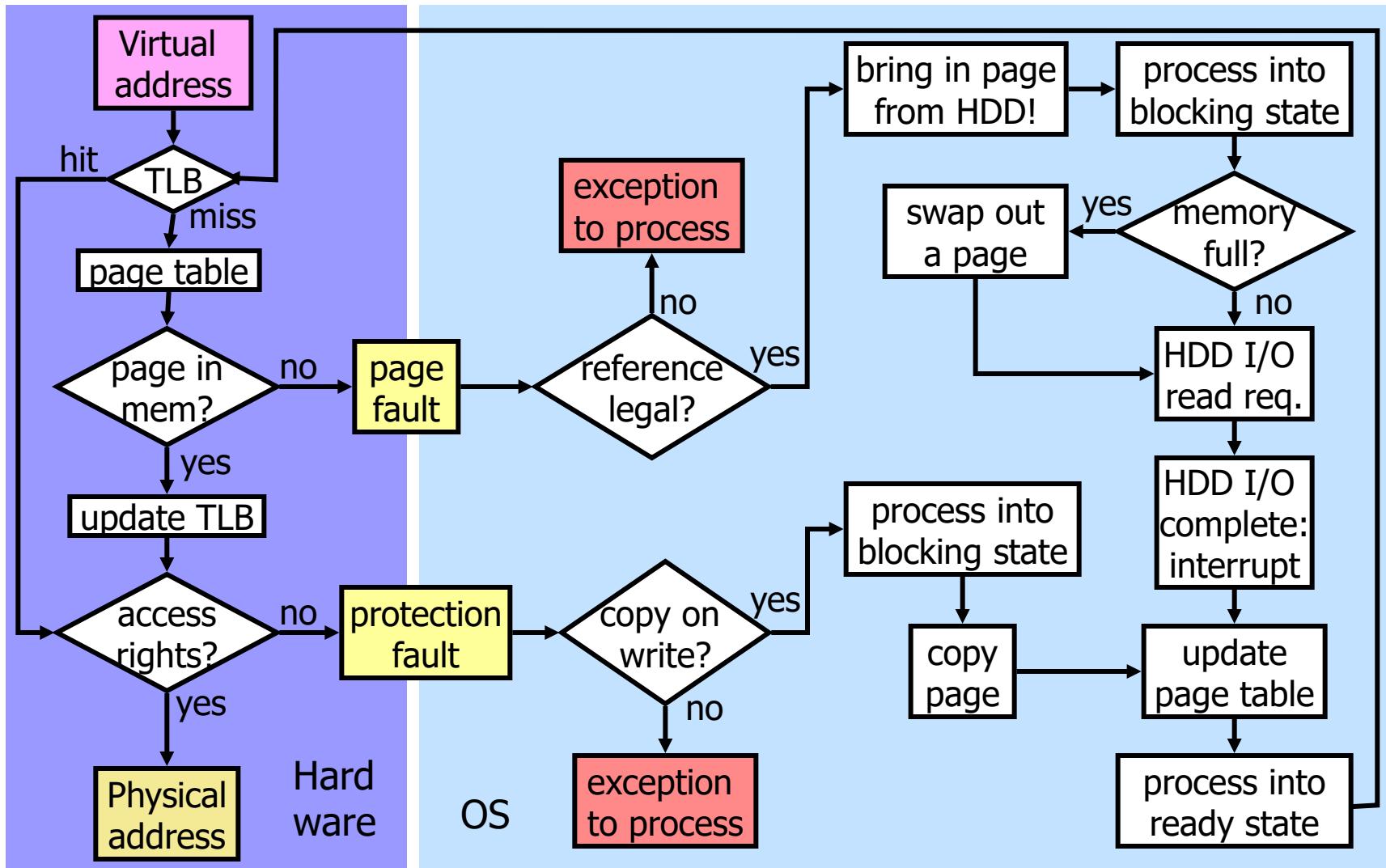
TLB (5)

■ Design Issues

- Accessed Frequently
 - Speed is important.
- Minimize TLB Miss
 - Can be fully-associative, set-associative, or direct mapped.
 - Often fully associative, can be set associative.
 - Sized to maximize hits, but make timing.
 - Usually not more than 128, 256 entries (associativity).

Address Translation Summary

Figure



Page Replacement

- LRU (needs use bits)
- FIFO
- ...
- Software (OS) implementation

Write Policy

- Write back
- Write through is not suitable for virtual memory. Why?

Internal Fragmentation

- Disadvantage of Paging: Internal Fragmentation
 - Last page is unlikely to be full.
 - Example: If the program and data need 26000 bytes on a machine with 4096 bytes per page, the first 6 page will be full, totaling $6 \times 4096 = 24576$ bytes, and the last page will contain $26000 - 24576 = 1424$ bytes. So 2672 bytes will be wasted.
 - Conclusion
 - If the page size is n bytes, the average amount of space wasted in the last page of a program by internal fragmentation will be $n/2$ bytes.

Page Size (1)

- Choosing a page size is a question of balancing forces that favor a larger page size versus those favoring a smaller size.
- Advantages of Choosing Large Page Size
 - The size of the page table is inversely proportional to the page size: Memory can therefore be saved by making the pages bigger
 - Transferring larger pages to or from secondary storage, possibly over a network, is more efficient than transferring smaller pages.

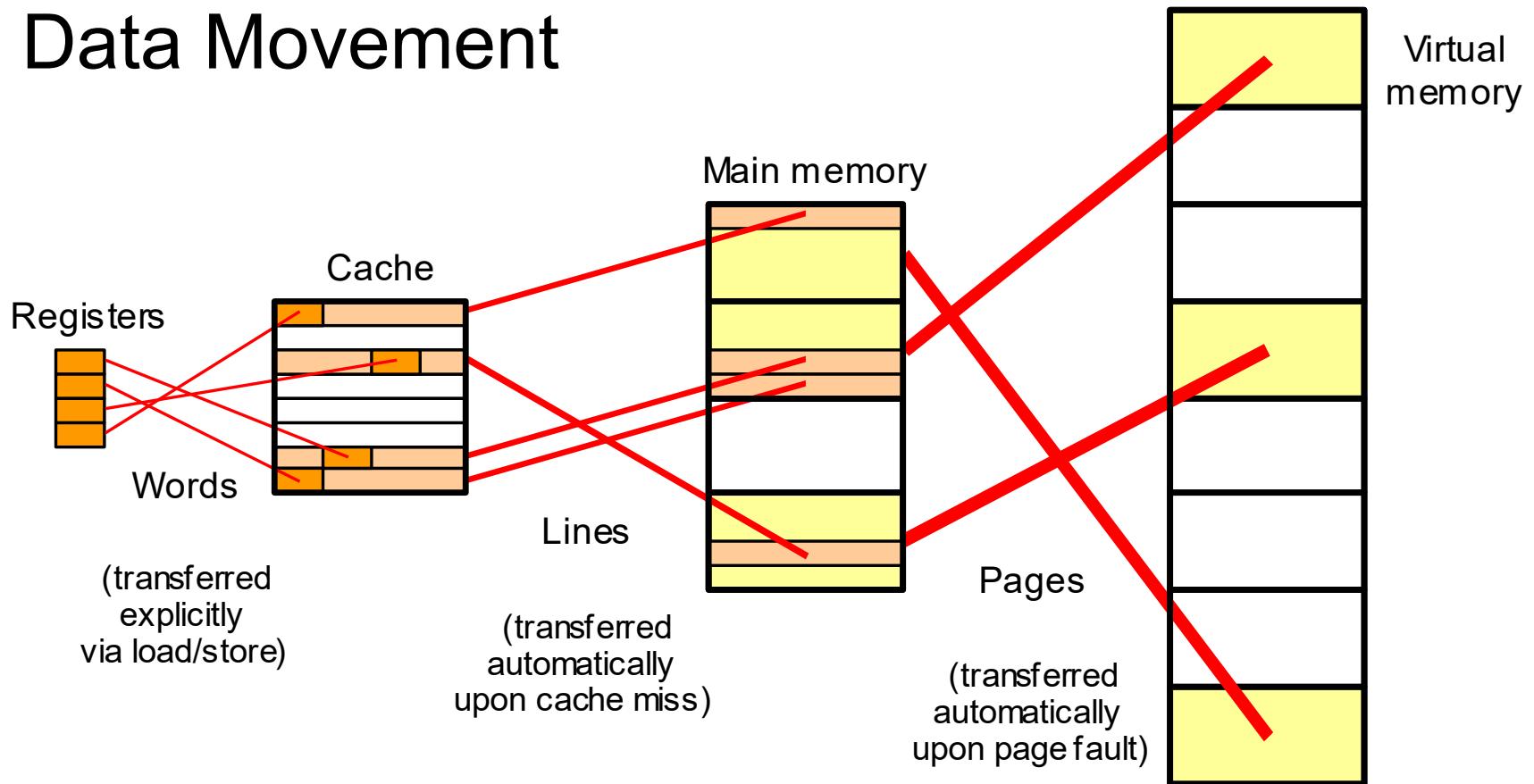
Page Size (2)

■ Advantage of Choosing Smaller Page Size

- The main reason for choosing smaller page is conserving storage. A small page size will result in less wasted storage when a contiguous region of virtual memory is not equal in size to a multiple of the page size.

Paging Summary (1)

■ Data Movement



Data movement in a memory hierarchy.

Paging Summary (2)

■ Comparing the 2 levels of hierarchy

Cache

Block or Line

Miss

Block Size: 32-64B

Placement:

Direct Mapped,
N-way Set Associative

Replacement:

LRU or Random

Write Thru or Back

How Managed:

Hardware

Virtual Memory

Page

Page Fault

Page Size: 4K-16KB

Fully Associative

Least Recently Used
(LRU) approximation

Write Back

Hardware + Software
(Operating System)

Summary (1)

■ 知识点: Virtual Memory

- 了解Motivations of VM
- 掌握What is VM
- 了解Implementation of VM: Paging,
Segmentation, Segmentation with paging

■ 知识点: Paging

- 掌握Address Translation
- 理解Page hit and Page fault
- 理解TLB的用途

Summary (2)

■ 知识点: Paging (ctd.)

- 理解Page table and page table entry
- 了解Page replacement
- 理解Write policy
- 理解Internal fragmentation
- 掌握Page size
 - Smaller vs. bigger

Exercise (1)

- 简答: What are advantages and disadvantages of choosing smaller page size in a paging system?
- Solution
 - Advantages: A small page size will result in less wasted storage when a contiguous region of virtual memory is not equal in size to a multiple of the page size.
 - Disadvantages:
 - (1) The size of the page table is inversely proportional to the page size. Memory can therefore be wasted by making the pages smaller.
 - (2) Transferring smaller pages to or from secondary storage, possibly over a network, is less efficient than transferring larger pages.

Exercise (2)

- 简答: Under what circumstances do page faults occur?
- Solution
 - A page fault occurs when an access to a page that has not been brought into main memory takes place.

Exercise (3)

■ 计算: Assume the following memory system:

- Virtual addresses are 14 bits
- Physical addresses are 12 bits
- The page size is 64 bytes
- The first 16 entries in the page table are as follows: (VPN means virtual page number, PPN means physical page number)

VPN	PPN	Valid
00	28	1
01	-	0
02	33	1
03	02	1
04	-	0
05	16	1
06	-	0
07	-	0

VPN	PPN	Valid
08	13	1
09	17	1
0A	09	1
0B	-	0
0C	1E	1
0D	2D	1
0E	11	1
0F	0D	1

Exercise (4)

- (1) What is the format of a virtual address? In other words, how many bits are used for the virtual page number and how many bits are used for the virtual page offset?
- (2) What is the format of a physical address? In other words, how many bits are used for the physical page number and how many bits are used for the physical page offset within a page?
- (3) Show how this memory system translates the virtual address 0x030D into the corresponding physical address. **Write the translation process!**

Exercise (5)

■ Solution:

- (1)Virtual page offset: 6 bits, virtual page number: 8 bits
- (2)Physical page offset: 6 bits, physical page number: 6 bits
- (3) $0x030D = (00\ 0011\ 0000\ 1101)_2$, so the virtual page number is $(00001100)_2 = 0xC$. Use this VPN to lookup the page table, get the PPN=1E.

So the physical address = $(011110001101)_2 = 0x078D$

Homework (1)

■ 补充题

5. Consider the memory system with the following specifications:

Byte-addressable.

Virtual address space: 4G bytes.

Main memory size: 16M bytes.

Cache size: 512K bytes.

Page size: 64K bytes.

Block size: 32 bytes

Mapping Strategies: Main Memory to Cache: 8-way set associative; Hard Disk to Main Memory: fully associative.

Virtual address is first translated to physical address. Then, it accesses the cache memory using the physical address.

Homework (2)

■ 补充题（续）

- (1) How many sets are there in the cache memory?
- (2) How long is the tag field of the cache?
- (3) Given a virtual address 0547AF33 (hexadecimal), its corresponding virtual page is stored in physical page 3B (hexadecimal).
 - ① What is its physical address under such mapping?
 - ② Which set can this address be possibly found in the cache?
 - ③ Which byte does this address point to out of the 32 bytes in a block?