Computer Architecture
Chapter 5

Large and Fast:
Exploiting Memory Hierarchy

Joonho Kong
School of EE, KNU

Virtual Memory

Definition

Virtual Memory
Provides an abstraction of (nearly) infinite memory space to programmers

- Much easier programming
- Programmers see the same and identical address space
- It allows efficient and safe sharing of memory between programs
- Because of sharing of memory, the protection of memory space is required
- Virtual address is translated to physical address when memory is accessed

Physical Memory
Actual memory space in DRAMs

What is a problem of direct mapping (have same address) between virtual memory and physical memory?

Virtual Memory

From Virtual Address to Physical Address

- Memory is used as a cache for secondary memory
- It is supported by operating systems (page managements)

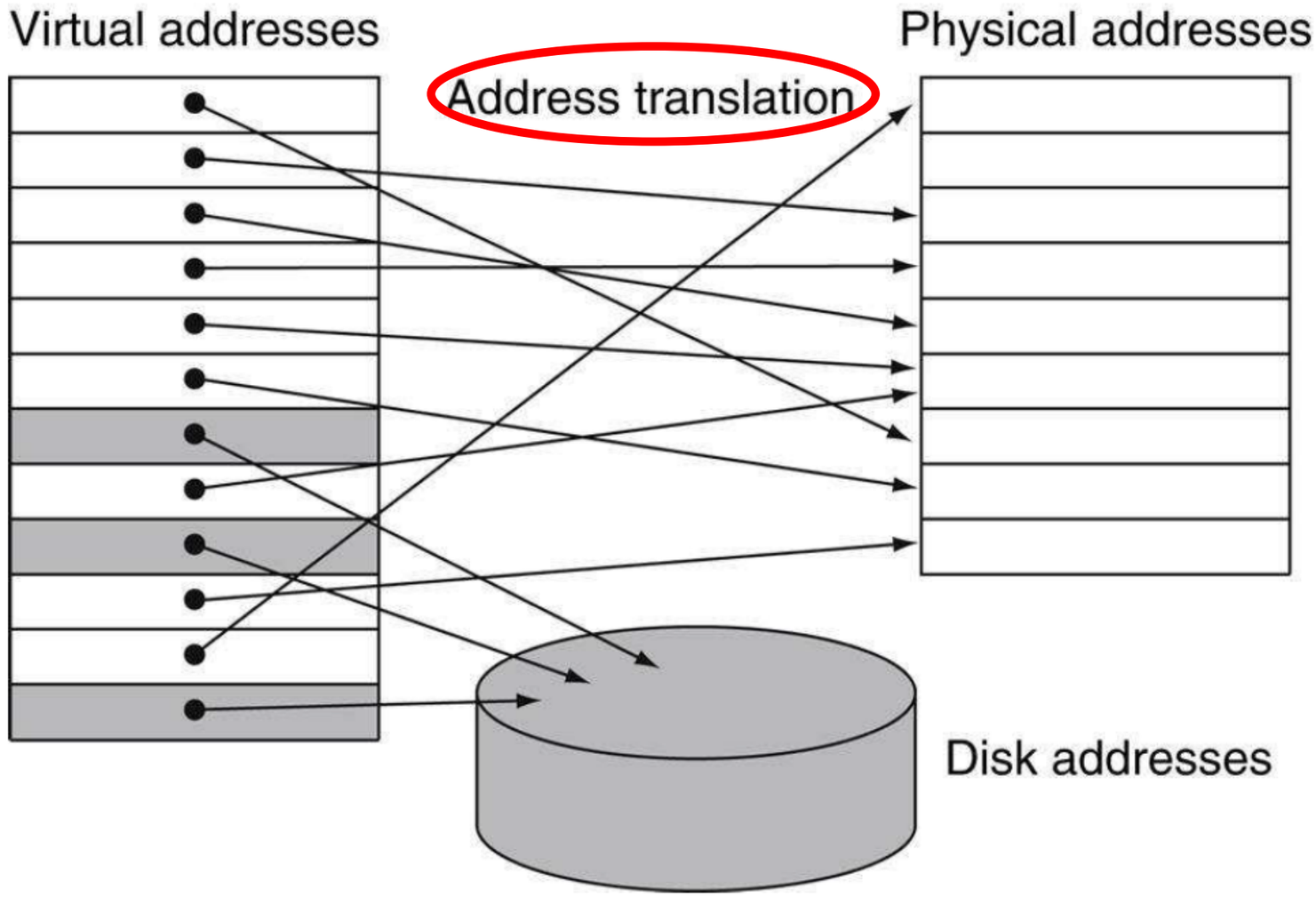


Figure 5.25 is from “Computer Organization and Design”

Virtual Memory

Translation

- A virtual address will be translated to physical address when memory is accessed

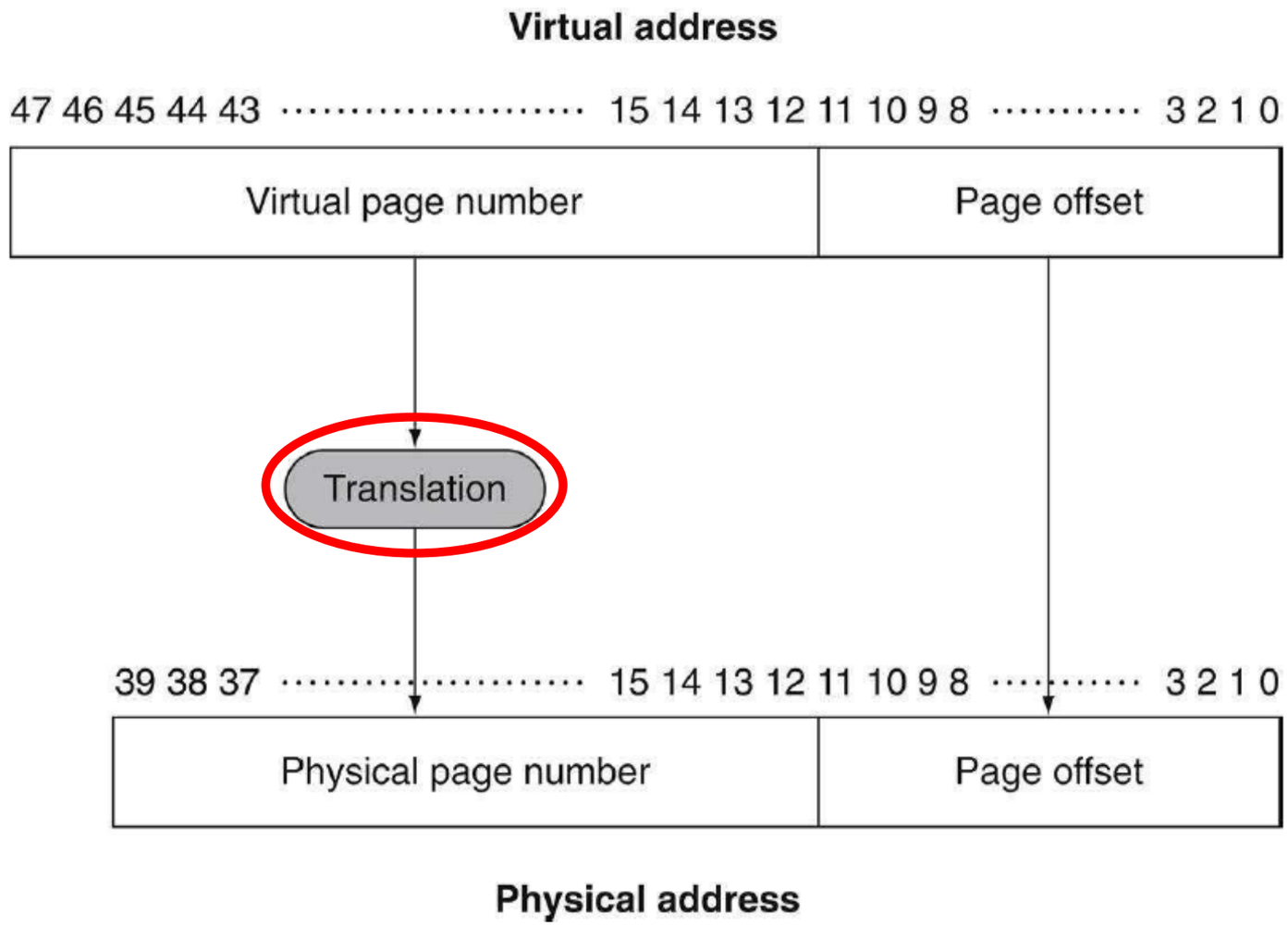


Figure 5.26 is from “Computer Organization and Design”

Virtual Memory

Paging

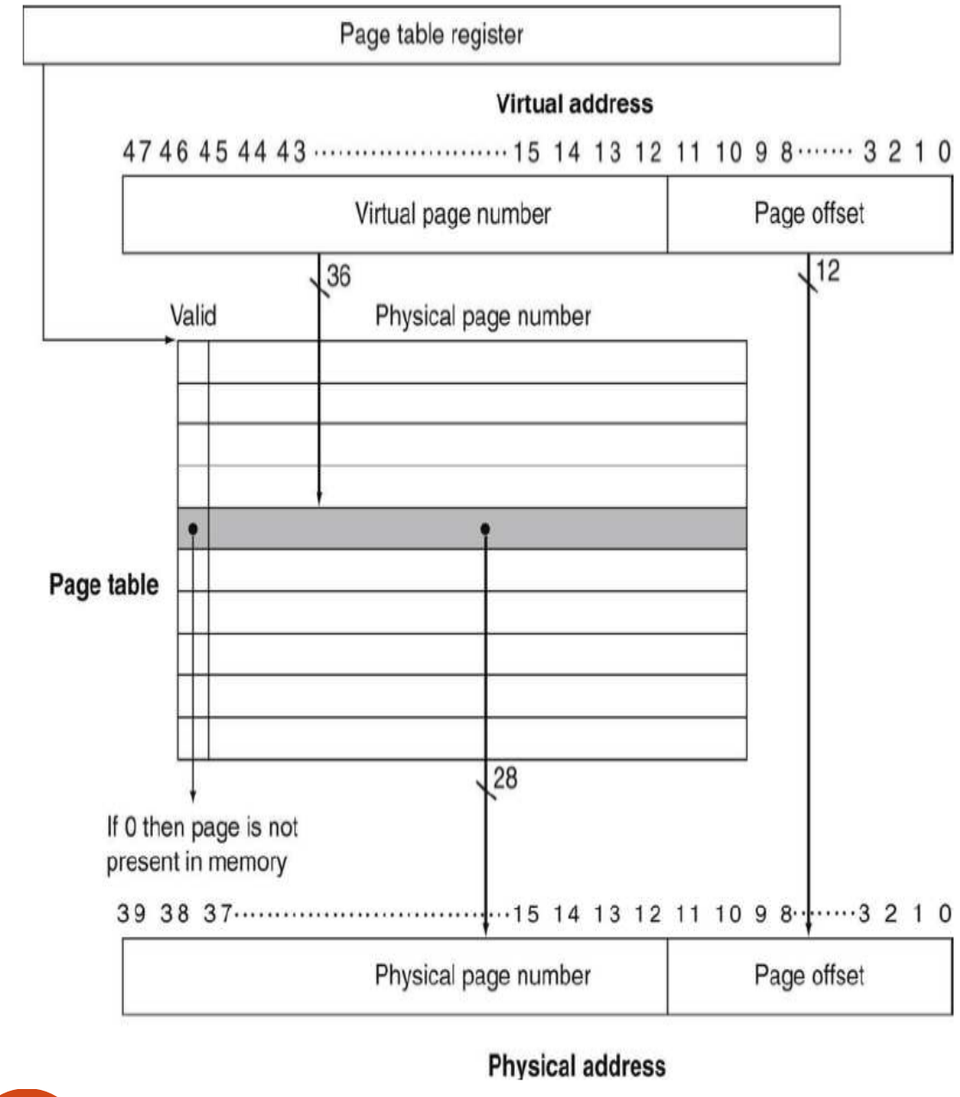
Page: A virtual memory block
Frame: A physical memory block

Page table:
- The table containing mapping information of the virtual-physical address translations
- The table is stored in main memory
- Each entry in the table contains the mapping between the page and frame

Page fault:
- It happens when an accessed page is not in main memory (we should fetch this page from the secondary storage)
- It is handled by operating system

Virtual Memory

Page Table Structure



1. Indexed by virtual page number
2. Check the valid bit in page table
3. If valid bit is on,
   correspond entry is used to physical page number

Figure 5.27 is from “Computer Organization and Design”

Virtual Memory

Page Fault

When the valid bit is 0,

1. The hardware generates an exception
2. The OS finds the page in the next level of hierarchy (secondary or permanent storage)
3. The OS puts corresponding page in the main memory

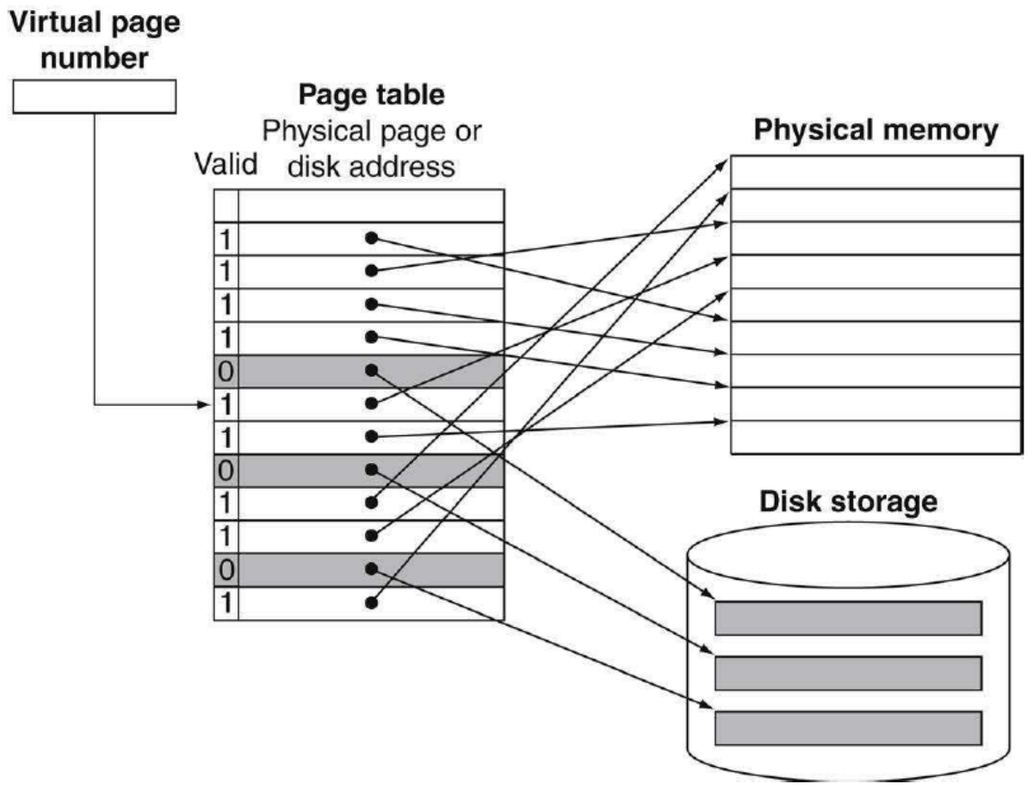


Figure 5.28 is from a book named “Computer Organization and Design”

Virtual Memory

Least Recently Used (LRU)

If all the pages in main memory are in use,
- The OS chooses a page to replace
- The replaced pages are written to secondary memory (swap memory)

Virtual Memory

Virtual Memory for Large Virtual Address

Let's assume that the page table has 64 billion entries and each size of entry is 8 bytes
It would require 0.5 TiB for the page table
A technique is needed to reduce the amount of storage for the page table

Virtual Memory

Virtual Memory for Large Virtual Address

Multiple-levels of page tables
- RISC-V uses this solution to reduce the size of address translation
- This method can allocate the entire page table in DRAM
- But multi-level mapping is the complex process for address translation
- In the worst case, if the required entry is in last level table, it requires more time to find the corresponding page entry

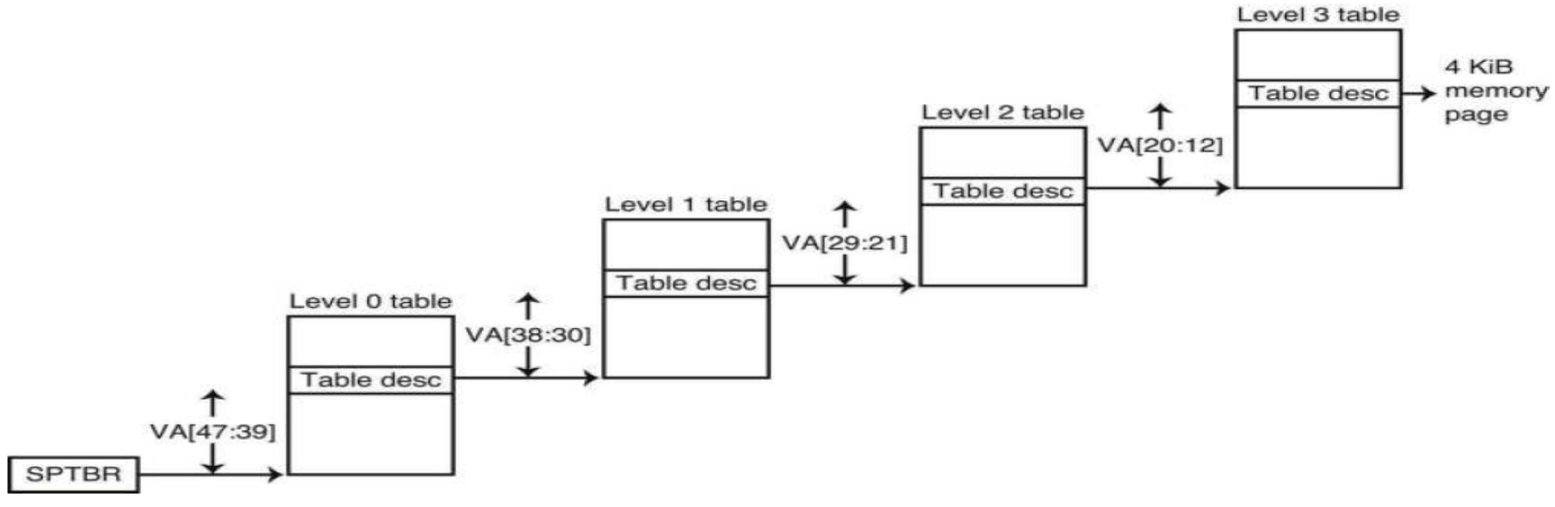


Figure 5.29 is from a book named “Computer Organization and Design”

Virtual Memory

Problem

- ld/st instruction or program counter uses virtual memory address
- Cache uses physical memory address

What would be a potential problem?

Figure 5.30 is from a book named “Computer Organization and Design”

Virtual Memory

Translation-Lookaside Buffer (TLB)

- A cache memory of the page table
- When a memory is accessed, TLB access is performed first
- If the TLB hit occurs, the access to main memory (page table walk) can be avoided

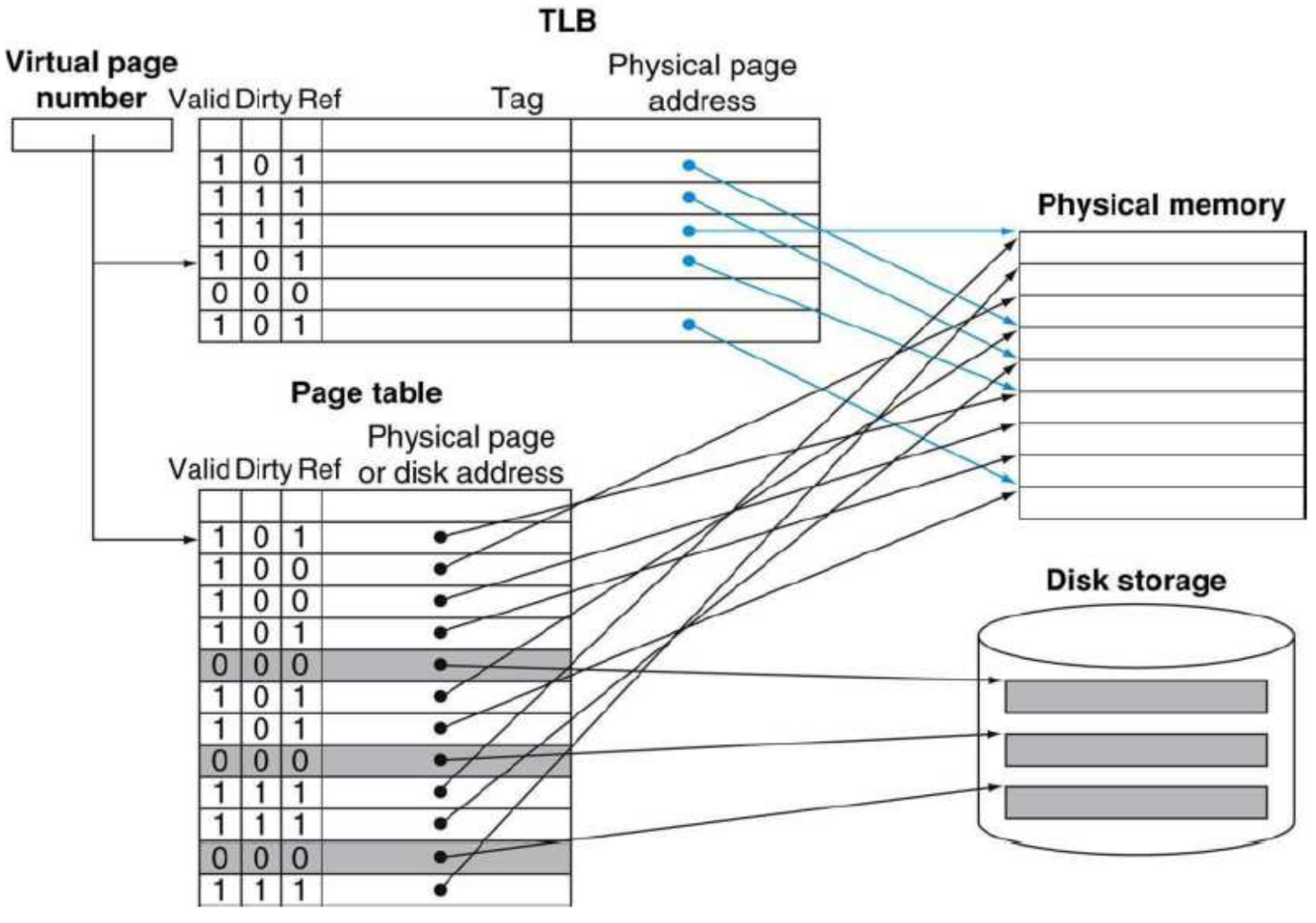


Figure 5.30 is from a book named “Computer Organization and Design”

Virtual Memory

About TLB Miss

When a TLB miss is occurred, it must be determined whether the page fault or TLB miss

Handling TLB misses, the page exists in memory
1) The hardware generates an exception and invokes the operating system
2) accesses the page table (page table walk)
3) Updates the TLB from the page table and resume the execution

Virtual Memory

About TLB Miss

Handling true page fault
1) Look up the page table entry and find the location of the page in secondary memory
2) Choose a physical page to replace
3) Read access to bring the page from secondary memory to main memory

- The last step will take millions of clock cycles (an order of milliseconds)
- The OS will select another process to execute (context switching) until the disk access completes
- Read from disk is complete, the OS restore the state of the original process

Virtual Memory

Virtual Memory Protection

Because we may share a single main memory by multi-processors or multi-cores, memory protection is important

- One process cannot access another process's data
- A user process is unable to change the page table mapping
- The processes can share information in a limited way (e.g., inter-process communication provided by OS system calls)
- Operating system is responsible for protection

Multi-core(-processor) cache

Sharing of the data in different cores or processors

Time	Event	Cache contents for CPU A	Cache contents for CPU B	Memory contents for location X
0				1
1	CPU A reads X	1		1
2	CPU B reads X	1	1	1
3	CPU A stores 0 into X	0	1	0

Data in X has different values in CPU A and B??
Which one is correct?

Parallelism and Memory Hierarchy: Cache Coherence

Cache Coherence

Cache coherence problem:
data in the same address actually exists multiple places
How can we manage the data coherently in the multiple places?

- Due to the shared data, the cache coherence need to be preserved
- Each processor may have different values for the same location

Time step	Event	Cache contents for CPU A	Cache contents for CPU B	Memory contents for location X
0				0
1	CPU A reads X	0		0
2	CPU B reads X	0	0	0
3	CPU A stores 1 into X	1	0	1

Figure 5.40 is from a book named “Computer Organization and Design”

Parallelism and Memory Hierarchy: Cache Coherence

Basic Scheme for Enforcing Coherence

The caches provide both migration and replication of shared data:

1. Migration
- Move data to a local cache and use data in there
- Reduce the latency to access a shared data

2. Replication
- The caches make a copy of the data in the local cache
- Reduce both the latency of access and contention for simultaneous read

Parallelism and Memory Hierarchy: Cache Coherence

Snooping Protocols

Write invalidate protocol:
It invalidates copies in other caches on a write

Processor activity	Bus activity	Contents of CPU A's cache	Contents of CPU B's cache	Contents of memory location X
				0
CPU A reads X	Cache miss for X	0		0
CPU B reads X	Cache miss for X	0	0	0
CPU A writes a 1 to X	Invalidation for X	1		0
CPU B reads X	Cache miss for X	1	1	1

update invalidation

Figure 5.41 is from a book named "Computer Organization and Design"