Big Picture

- Part 3: implementation of ISA
  - High-level organization, not circuits design
  - Ch. 4: processor
    - Given ISA, what is a good implementation?
    - Datapath and control, pipelining
  - Ch. 5: memory system design
    1) Memory systems: physical and virtual
    2) Memory hierarchy
    3) Cache memory: structure, operation and performance
    4) Cache and virtual memory

Chapter 5

Large and Fast: Exploiting Memory Hierarchy

Part 2

Some of authors' slides are modified

Virtual Memory

(OS topic; 가볍게)

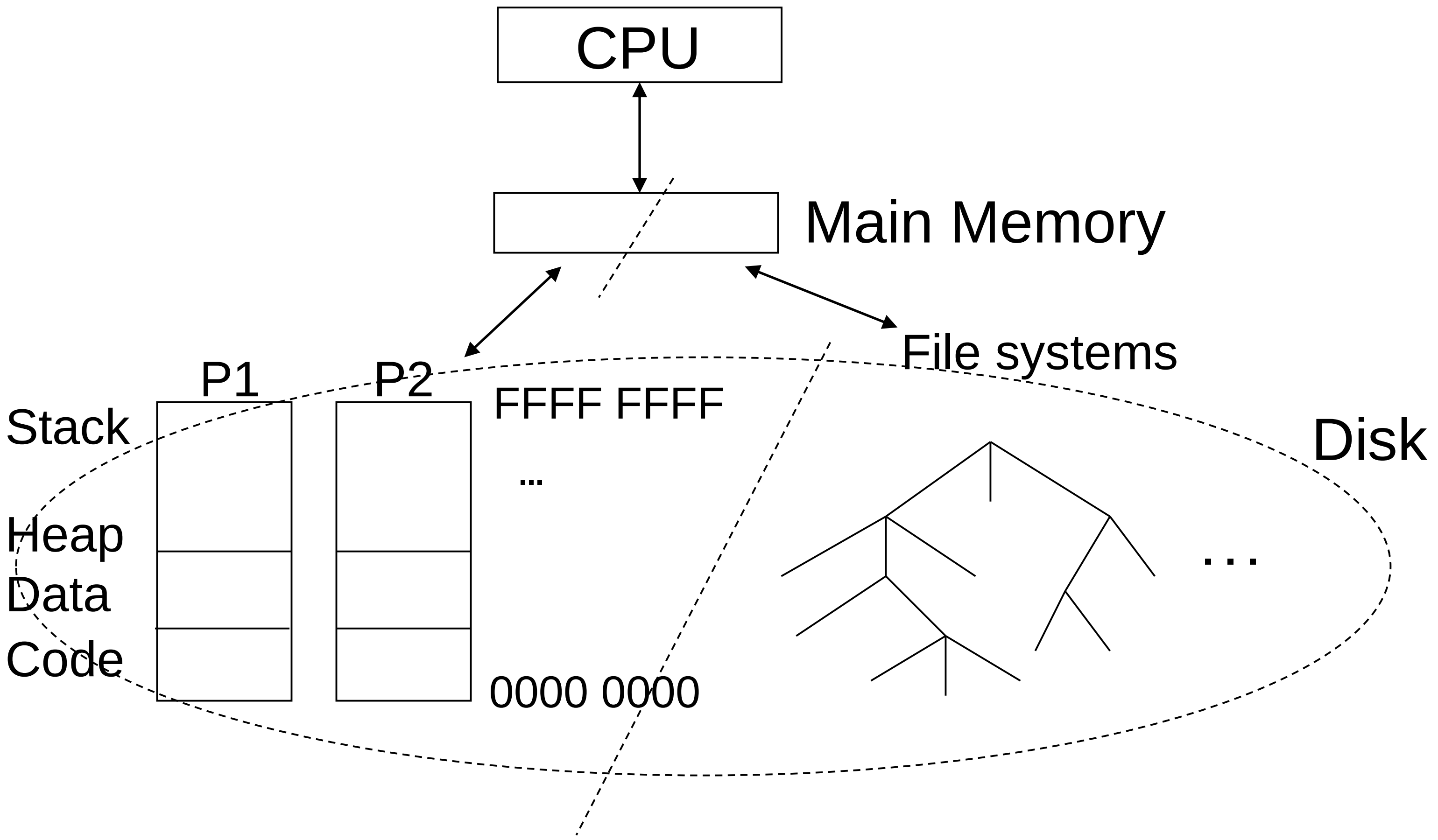
Memory Management (반복)

- ❑ Cache memory management (Architecture topic)
  - Cache part of main memory
  - Implemented by hardware: fast, simple
    - Part of processor (on-chip cache)
    - Hardware accelerator: SW not know about it
- ❑ Virtual memory management (OS topic)
  - Use main memory as cache for disk
  - Implemented by software
    - Disk access is already slow (10 ms)
- ❑ Same principles (caching, locality, management) for both
  - Usage: independent of each other

Memory Management

- Forget about cache memory for now
  - It is hardware accelerator, a part of CPU
  - With cache, CPU feels that main memory is faster
- Virtual memory management
  - Use main memory as a "cache" for disk
  - Focus only on main memory and disk
- Separation of concern

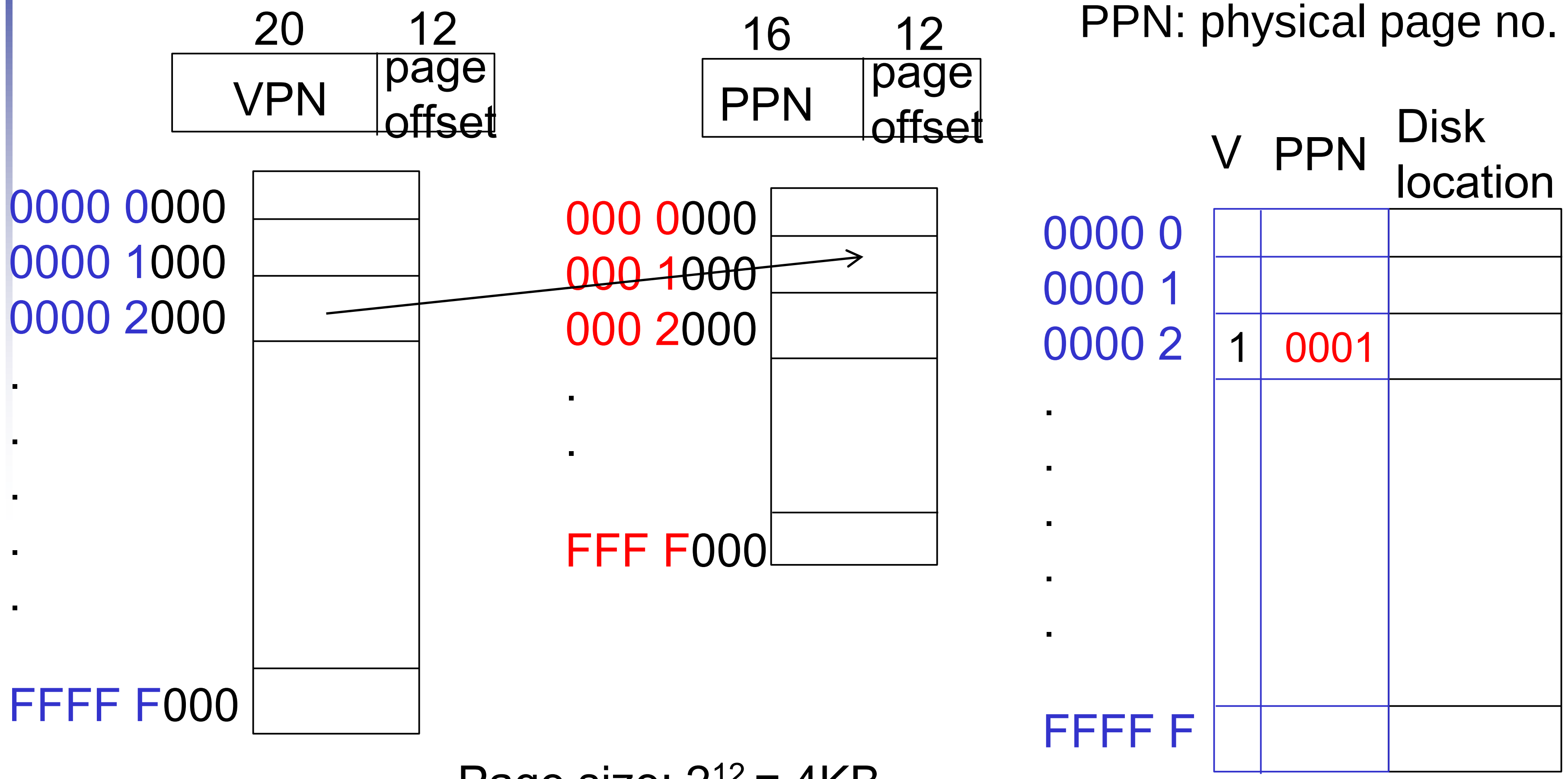
Ignore Cache Memory



Virtual Memory: Placement

VPN: virtual page no.

PPN: physical page no.



Page size: $2^{12} = 4KB$

Virtual space: $2^{32} = 4GB$

Main memory: $2^{28} = 256MB$

Page Table (per-process)

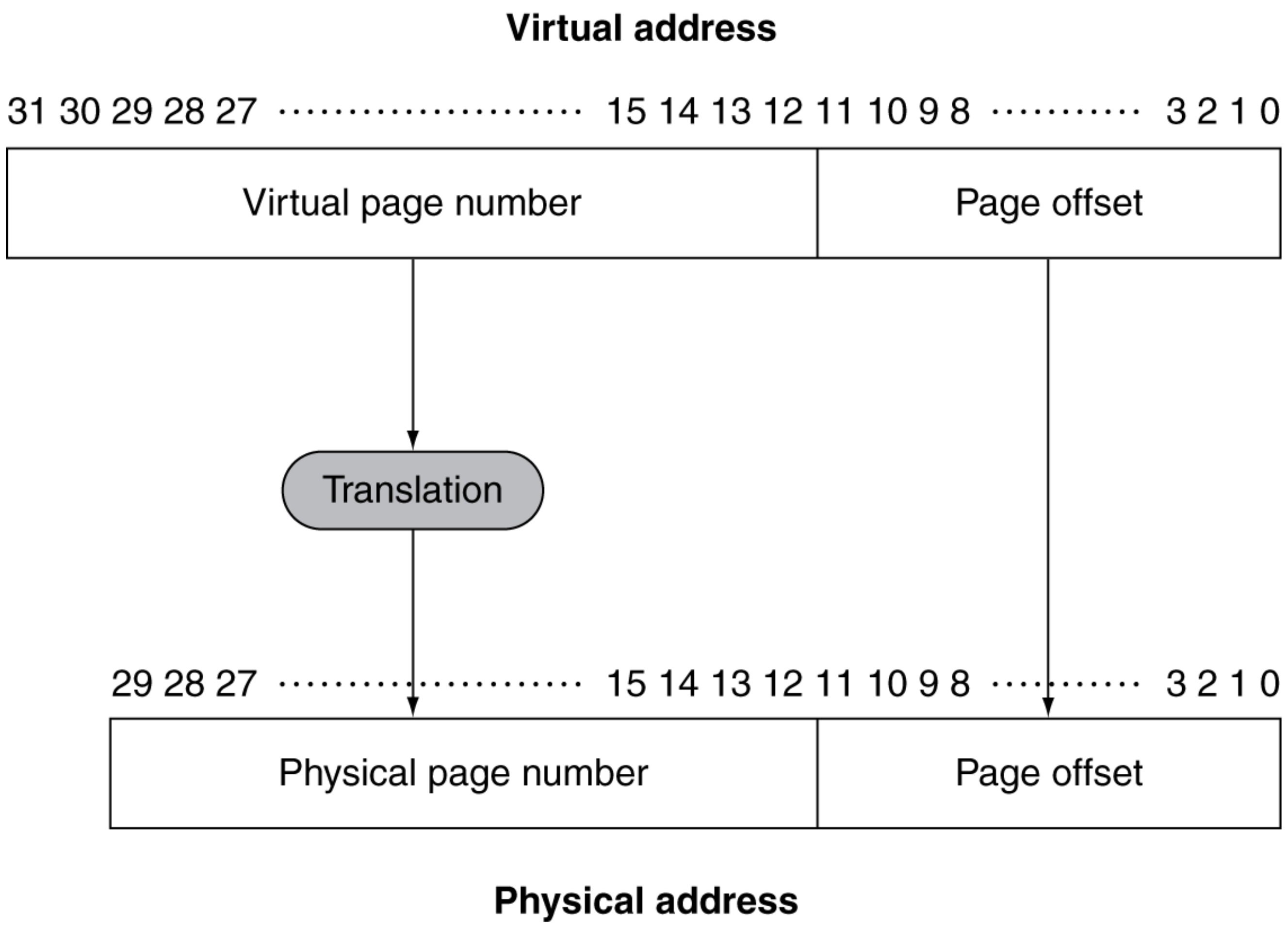
Virtual Memory

- What kind of mapping (placement) do we use?
  - Table data structure in software
- Use main memory as a "cache" for disk
- Multiple user programs share main memory
  - OS manages per-process page table
  - Protected from other programs
- CPU and OS translate virtual addresses to physical addresses

Virtual Memory

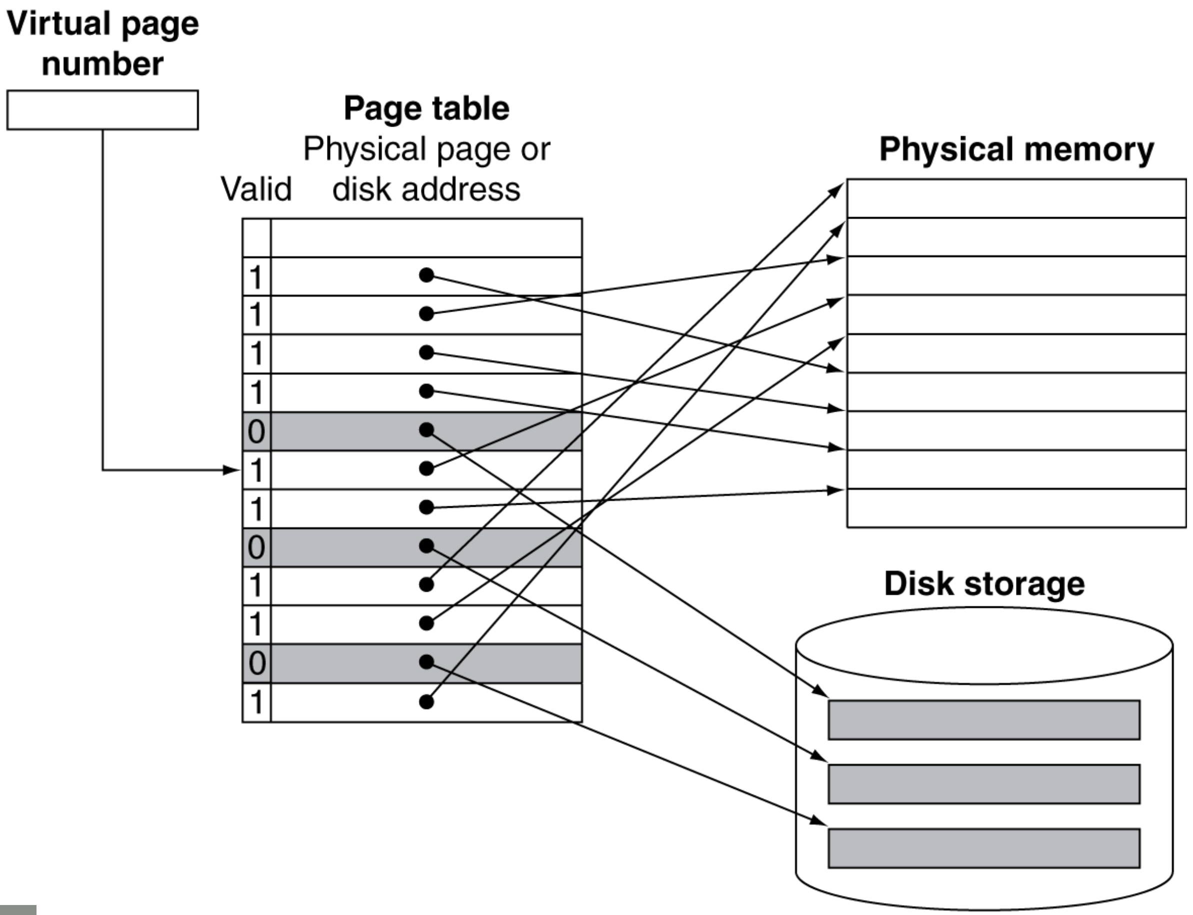
- When creating a process, OS:
  - Create space in disk or flash memory for all pages of process (swap space)
    - Create per-process page table
    - Record the location of each virtual page on disk
  - Also, track which process and which virtual page use each physical page
  - Page replacement (with approximated LRU)
- Process (state of a program)
  - Page table, program counter, registers

Address Translation (V-to-P)



Mapping Pages to Storage

- Per-process page tables

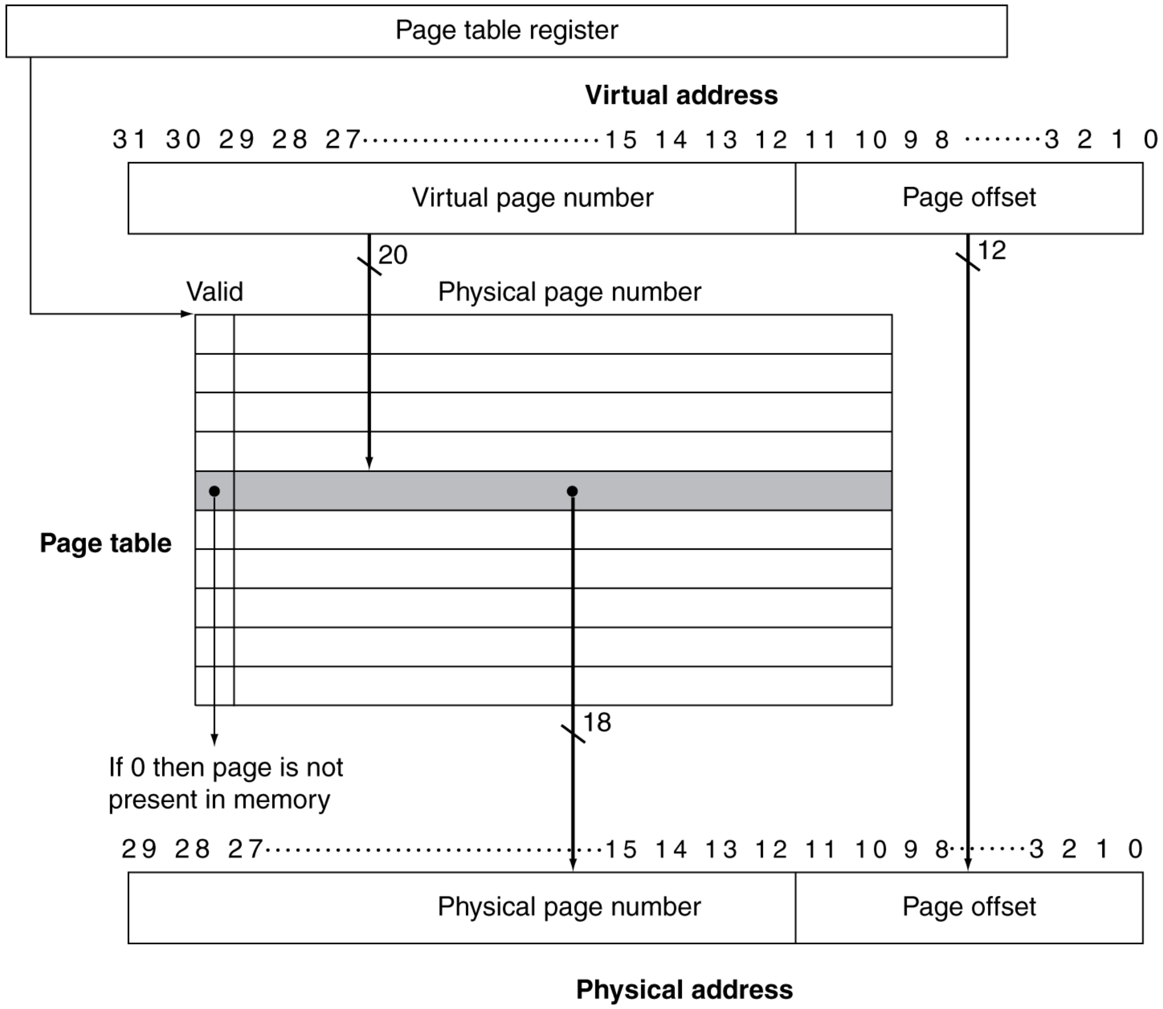


Page hit

Page fault

Translation Using a Page Table

Process 별 page table 시작 주소



Page Tables (부연)

- Stores placement information
  - Array of page table entries, indexed by virtual page number
  - Page table register in CPU points to page table in physical memory
- If page is present in memory
  - PTE (page table entry) stores the physical page no.
  - Plus other status bits (referenced, dirty, …)
- If page is not present (page fault)
  - PTE can refer to location in swap space on disk

Page Fault Penalty

- On page fault, the page must be fetched from disk
  - Takes millions of clock cycles
  - Handled by OS code
    - May switch process (in order of microseconds)
- Try to minimize page fault rate
  - Pages should be large enough
    - From 4KB up to 64KB
  - Fully associative placement
  - Smart replacement algorithms

Four Issues in Virtual Memory

- Q1: placement
- Q2: identification
- Q3: write strategy
  - Use write-back (dirty bit in PTE)
- Q4: replacement policy
  - Some form of LRU (always approximated)

Writes and Replacement

- Disk writes take millions of cycles
  - Block at once, not individual locations
  - Write through is impractical, so use write-back
    - Dirty bit in PTE set when page is written
- To reduce page fault rate, prefer least-recently used (LRU) replacement
  - Reference bit in PTE set to 1 on access to page
    - Periodically cleared to 0 by OS
  - Page with reference bit = 0 has not been used recently

TLB
(Translation-Lookaside Buffer)

(OS Topic)

Fast Translation Using a TLB

- Address translation requires extra memory references
  - One to access the PTE (page table entry)
  - Then the actual memory access
- Huge locality in accessing PTEs (VPN-PPN pairs)
  - Use the idea of cache again: store recently used PTEs (VPN-PPN pairs) in a fast cache
    - Called a Translation Look-aside Buffer (TLB)
    - Typical: 16–512 PTEs, 0.5–1 cycle for hit, 10–100 cycles for miss, 0.01%–1% miss rate

Fast Translation Using a TLB

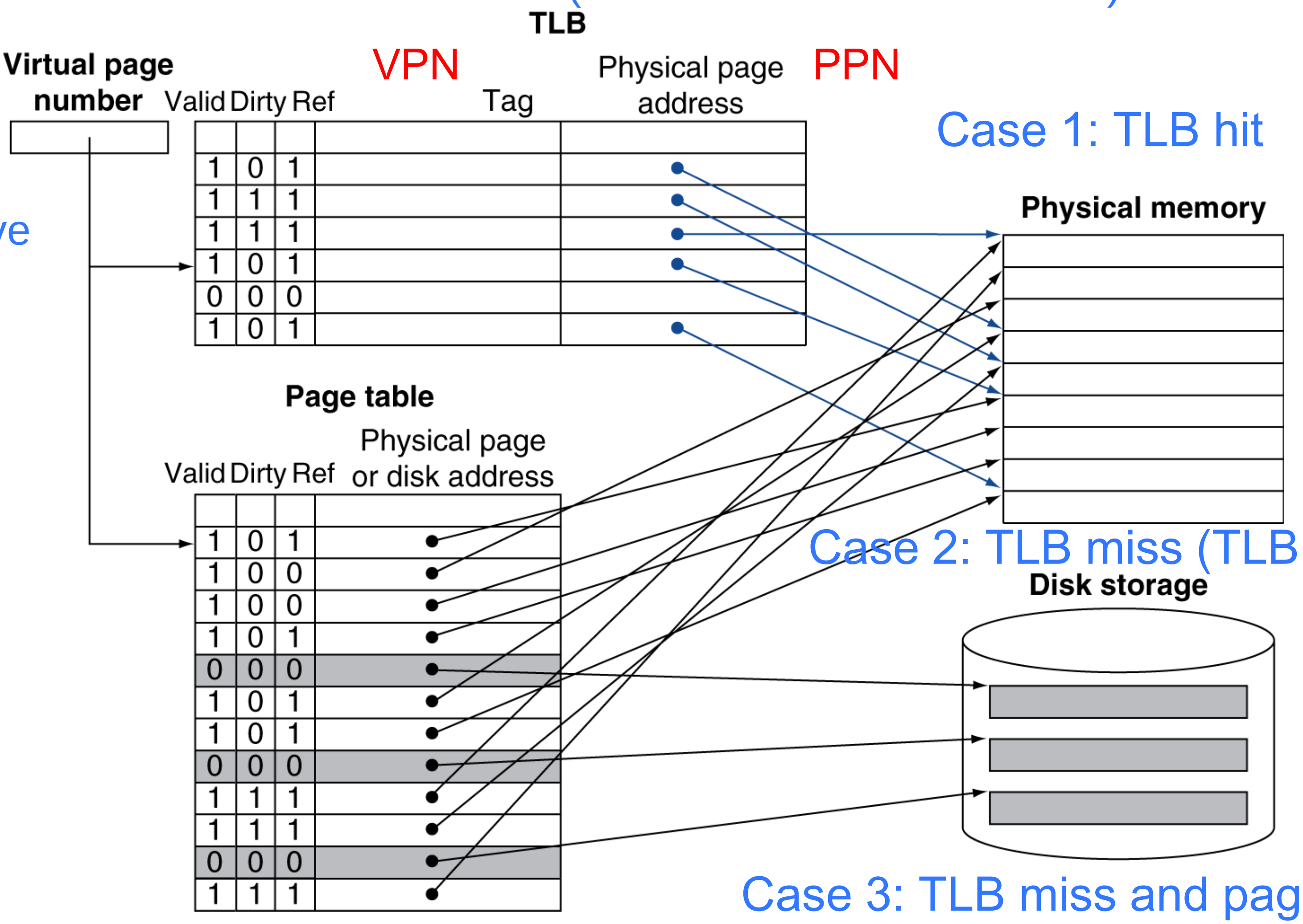
(Translation-lookaside buffer)

Fully associative mapping

Case 1: TLB hit

Case 2: TLB miss (TLB update)

Case 3: TLB miss and page fault (update page table and TLB)



1

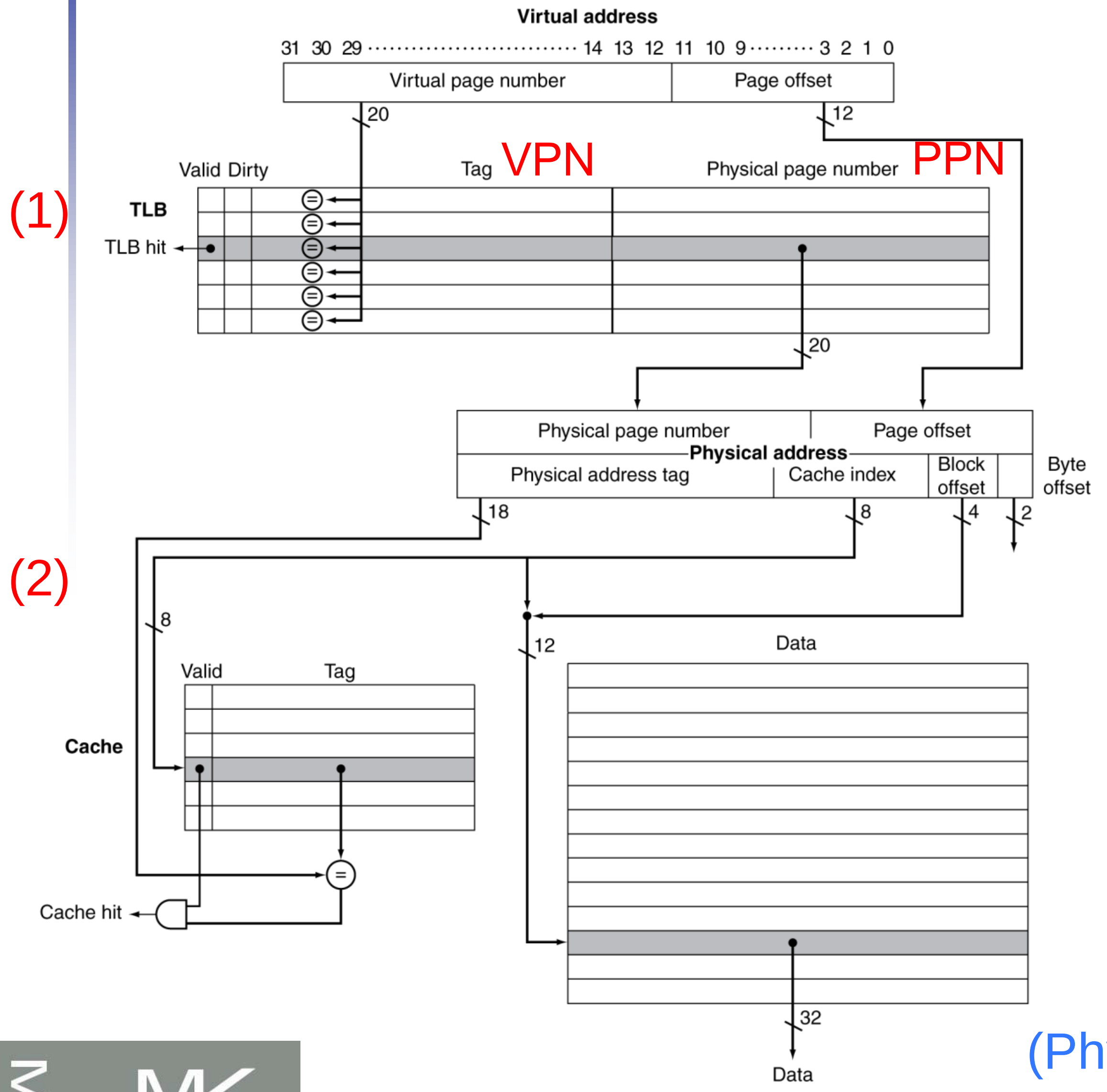
TLB Misses (부연)

- If page is in memory
  - Load the PTE from memory and retry
  - Could be handled in hardware
  - Or in software
    - Raise a special exception, with optimized handler
- If page is not in memory (page fault)
  - OS handles fetching the page and updating the page table
  - Then restart the faulting instruction

Cache and Virtual Memory:

Interactions

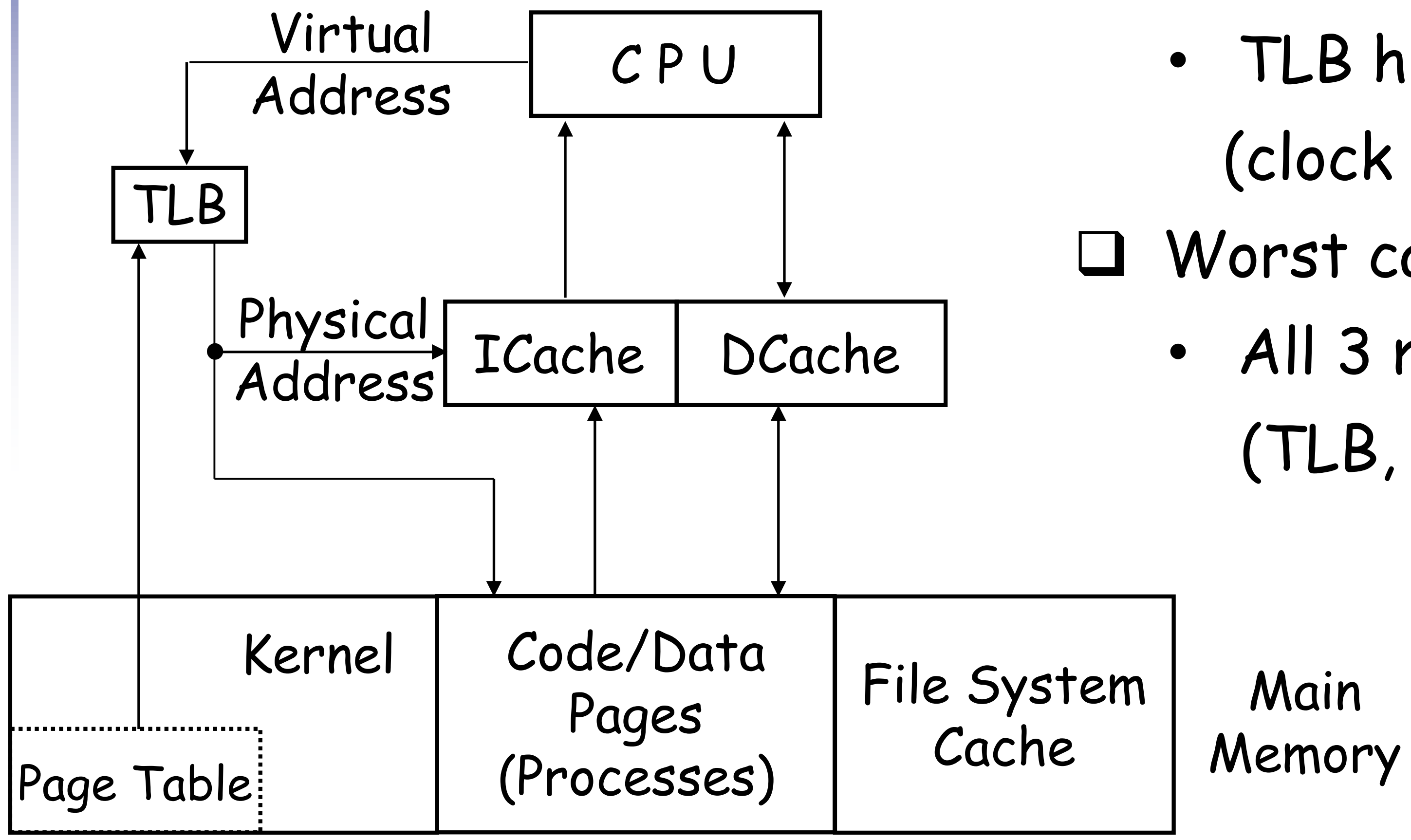
TLB and Cache Interaction



- If cache tag uses physical address
  - Need to translate before cache lookup
- Alternative: use virtual address tag
  - Complications due to aliasing
  - Different virtual addresses for shared physical address

(Physical cache vs. virtual cache)

Full Picture: Cache and Virtual Memory

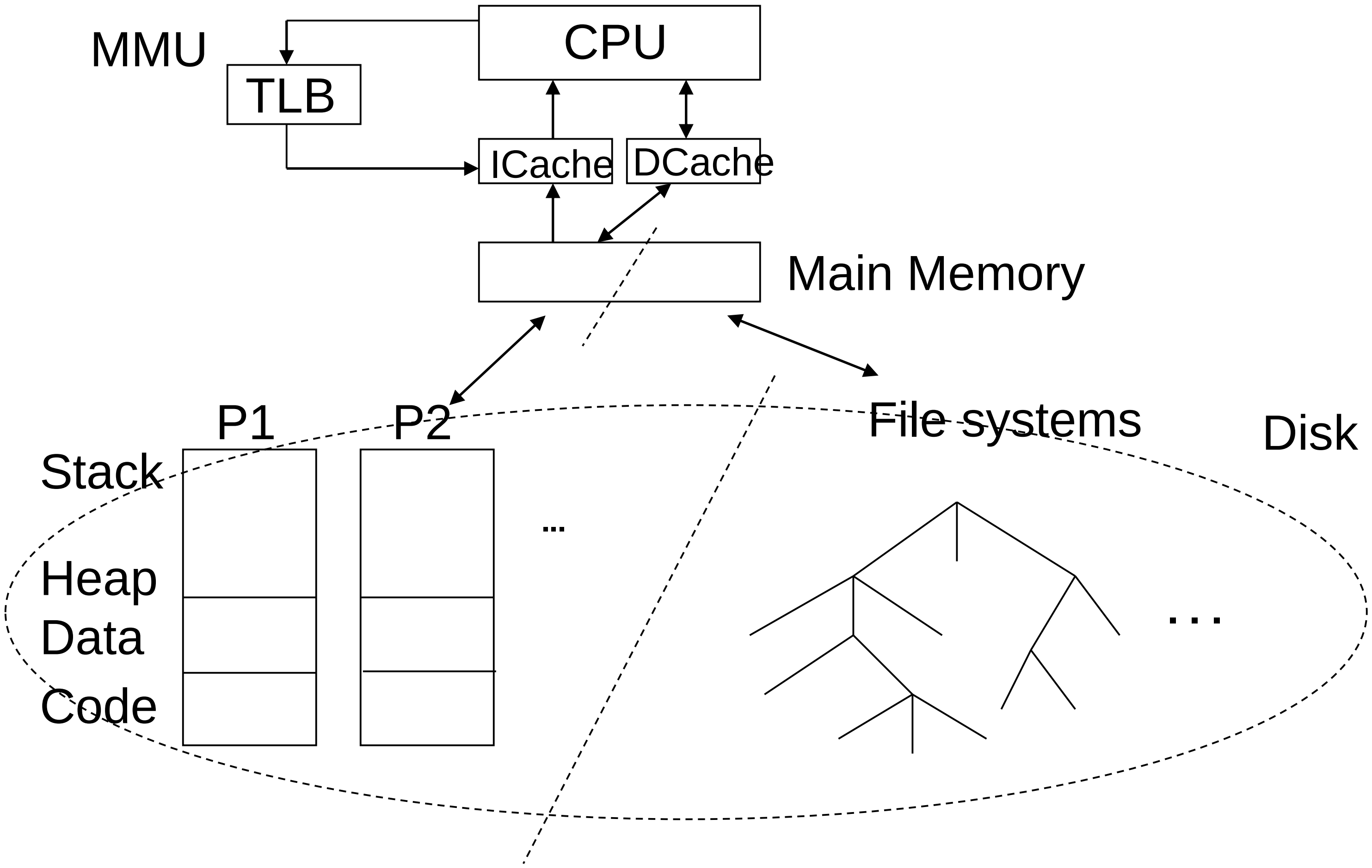


- Best case
  - TLB hit, cache hit (clock cycle time)
- Worst case
  - All 3 misses (TLB, PT, cache)

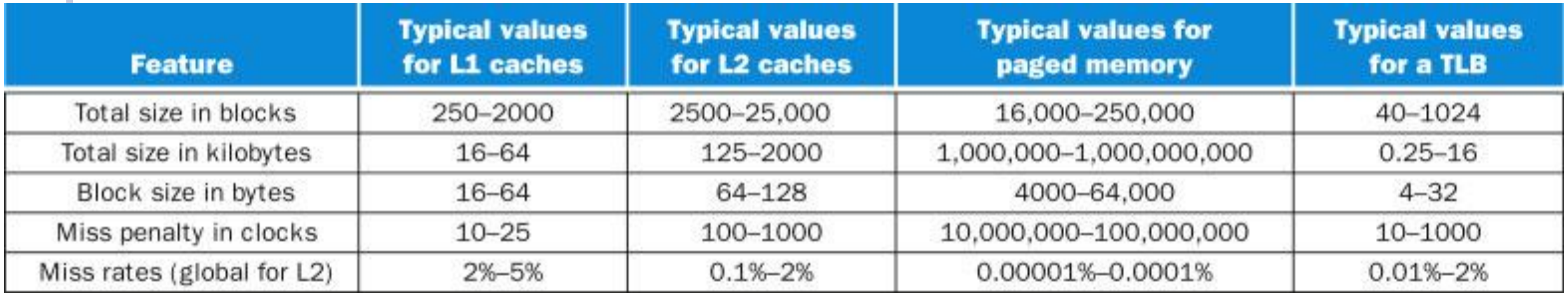
TLB, Cache, Main Memory (부연)

- TLB hit and cache hit: best case (clock cycle time)
- On TLB miss
  - Page table hit
    - Address translation using page table in main
    - Update TLB with new VPN-PPN pair
  - Page table miss (page fault)
    - OS handles fetching the page, updates the page table and TLB (process switch may occur)
    - Then restarts the faulting instruction
- On cache miss, check with main memory

General-Purpose Computers



Typical Values As of 2008 (참고)



Feature	Typical values for L1 caches	Typical values for L2 caches	Typical values for paged memory	Typical values for a TLB
Total size in blocks	250–2000	2500–25,000	16,000–250,000	40–1024
Total size in kilobytes	16–64	125–2000	1,000,000–1,000,000,000	0.25–16
Block size in bytes	16–64	64–128	4000–64,000	4–32
Miss penalty in clocks	10–25	100–1000	10,000,000–100,000,000	10–1000
Miss rates (global for L2)	2%–5%	0.1%–2%	0.00001%–0.0001%	0.01%–2%

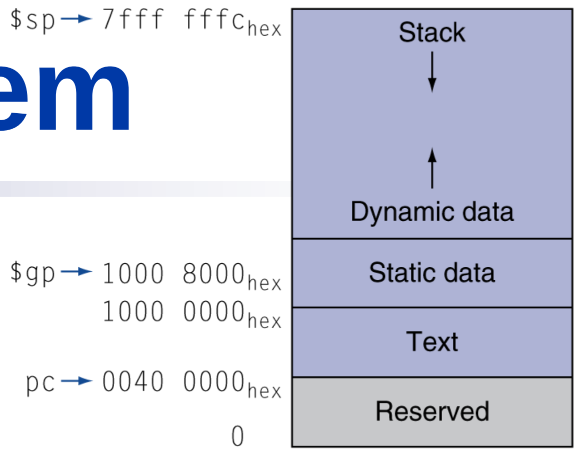
Cache Performance:

Three-C Model (optional)

Sources of 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

How to Compute Them



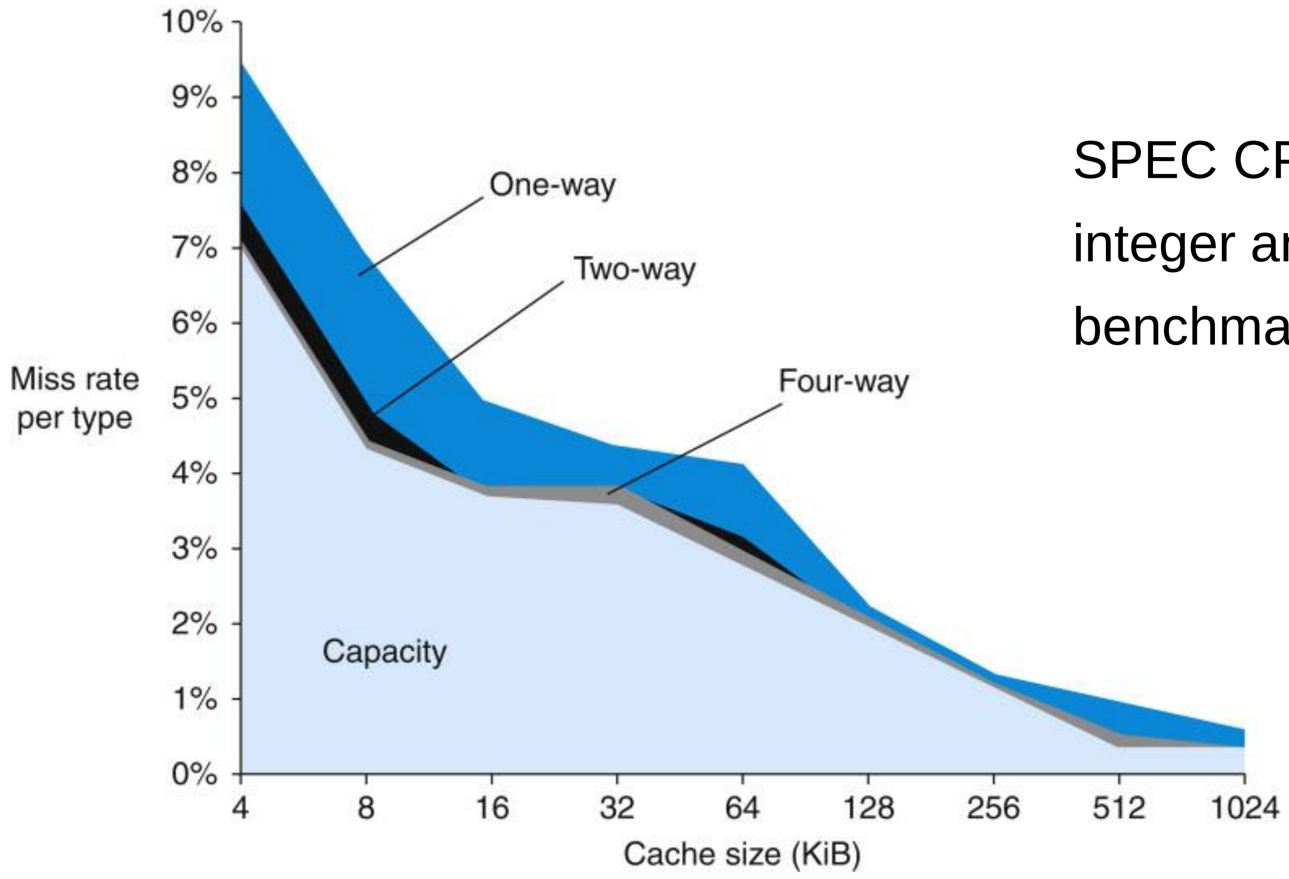
- Compulsory misses
  - Accessed regions (address trace) ÷ block size
- Cache simulation with given cache size and fully associative mapping
  - Will get compulsory and capacity misses
- Cache simulation with given cache size and given mapping
  - Will get compulsory and capacity and conflict misses
- Cache configuration: cache size, mapping, block size

Cache Design Trade-offs

Design change	Effect on miss rate	Negative performance effect
Increase cache size	Decrease capacity misses	May increase access time
Increase associativity	Decrease conflict misses	May increase access time
Increase block size	Decrease compulsory misses	Increases miss penalty. For very large block size, may increase miss rate due to pollution.

Three-C Model

- Compulsory misses: almost negligible (0.006%)



SPEC CPU2000 integer and FP benchmarks

Concluding Remarks

- ❑ Fast memories are small, large memories are slow
  - We really want fast, large memories ☹
  - Caching gives this illusion ☺
- ❑ Principle of locality
  - Programs use a small part of their memory space frequently
- ❑ Memory hierarchy
  - L1 cache $\leftrightarrow$ L2 cache $\leftrightarrow$ … $\leftrightarrow$ DRAM memory $\leftrightarrow$ disk
- ❑ Memory system design is critical for high performance

Homework #14 (see Class Homepage)

1) Write a report summarizing the materials discussed in Topics 5-1 and 5-2 (이번 주 수업 내용)

** 문장으로 써도 좋고 파워포인트 형태의 개조식 정리도 좋음

2) Solve Chapter 5 exercises 5.2.3, 5.3, 5.6, 5.7.3

- Due: see Blackboard
  - Submit electronically to Blackboard

Big Picture

- Issue 1: Fundamental concepts and principles
  - What is computer, CSE, computer architecture?
- Issue 2: ISA (HW-SW interface) design
  - Ch. 1: computer performance
  - Ch. 2: language of computer; ISA
  - Ch. 3: data representation and ALU
- Issue 3: implementation of ISA (internal design)
  - Ch. 4: processor (data path, control, pipelining)
  - Ch. 5: memory system (cache memory)
- Ch. 6: short introduction to parallel processors