

CMPE110 Lecture 23

Virtual Memory 2

Heiner Litz

<https://canvas.ucsc.edu/courses/19290>

Announcements



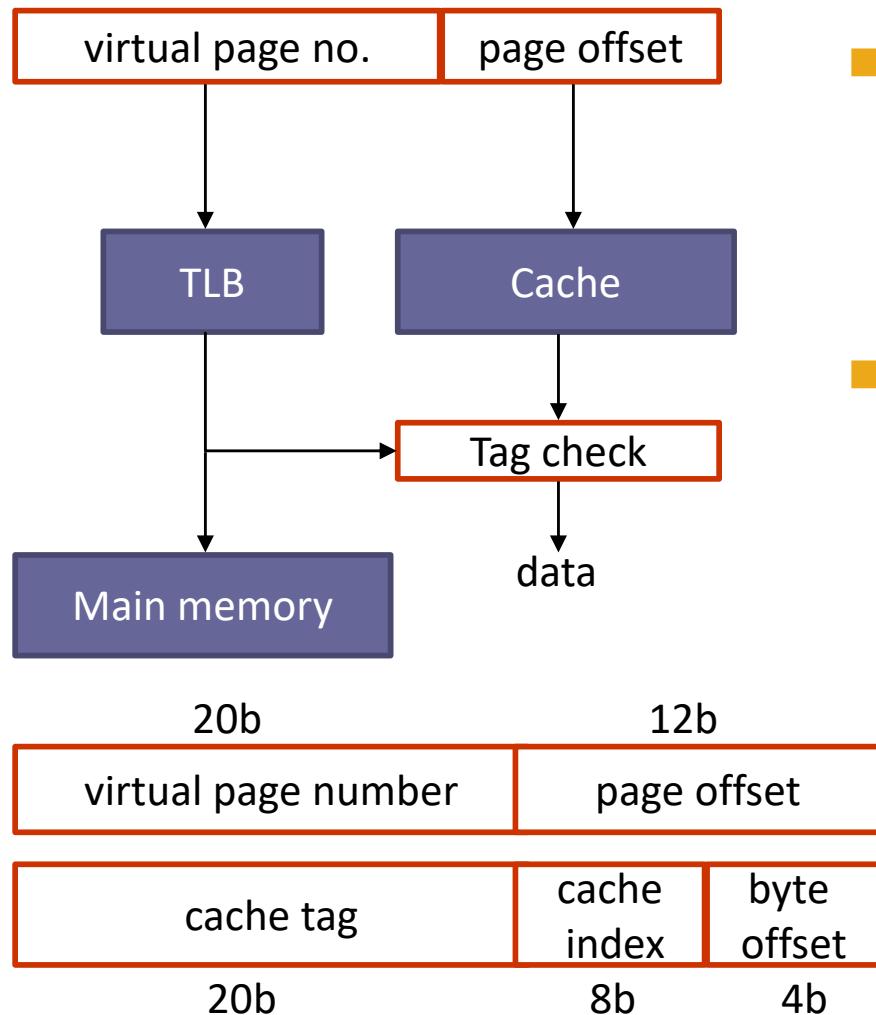
■ Review Lecture on Monday March 11th

- Given by Peter, Minghao, Saba
- Propose Topics on Piazza now
- If we have lots of topics, well poll and select on Friday

Review



Virtually Indexed, Physically Tagged Caches



- Translation & cache access in parallel
 - Start access to cache with page offset
 - Tag check used physical address
- Only works when
 - VPN bits not needed for cache lookup
 - Cache Size \leq Page Size * Associativity
 - I.e. Set Size \leq Page Size
 - Ok, we want L1 to be small anyway

Final Problem: Page Table Size



Page table size is proportional to size of address space

x86 example: virtual addresses are 32 bits, pages are 4 KB

Total number of pages: $2^{32} / 2^{12} = 1 \text{ Million}$

Page Table Entry (PTE) are 32 bits wide

Total page table size is therefore $1M \times 4 \text{ bytes} = 4 \text{ MB}$

But, only a small fraction of those pages are actually used!

Why is this a problem?

The page table must be resident in memory (why?)

What happens for the 64-bit version of x86?

What about running multiple programs?

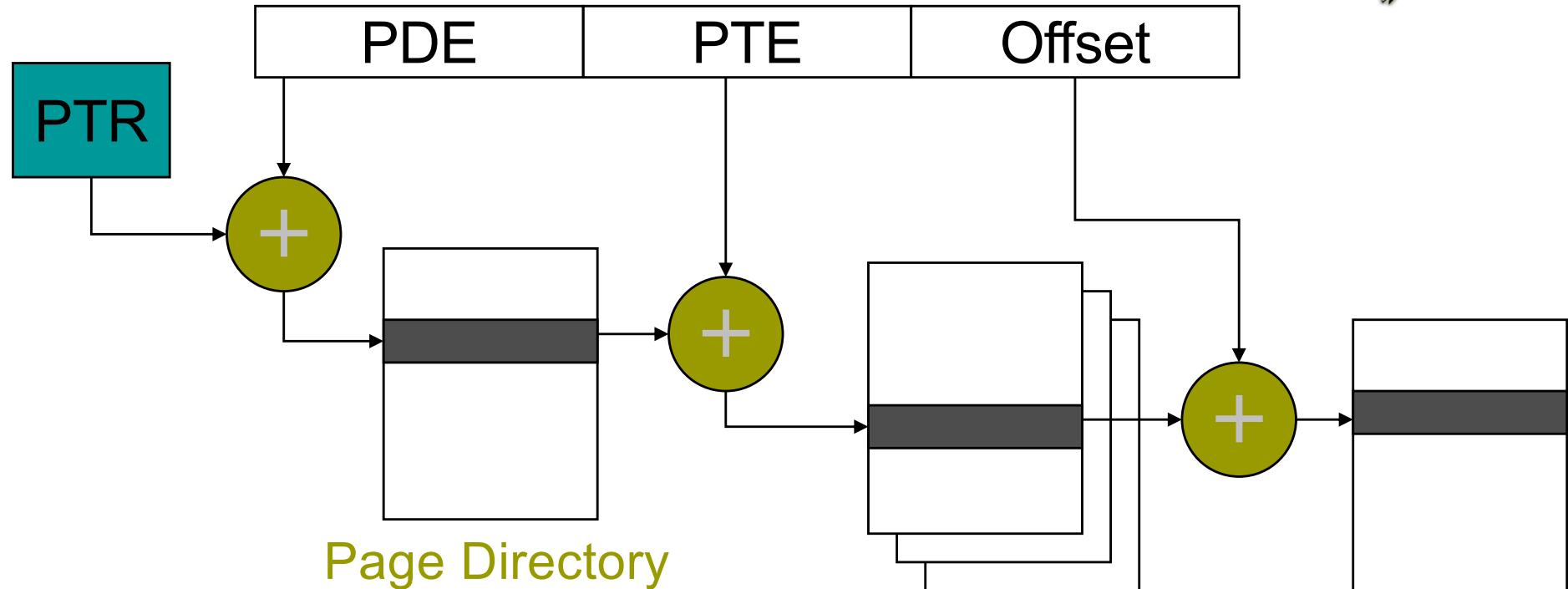
Solution: Multi-Level Page Tables



- Use a hierarchical page table structure
- Example: Two levels
 - First level: directory entries
 - Second level: actual page table entries
- Only top level must be resident in memory
- Remaining levels can be in memory, on disk, or unallocated
 - Unallocated if corresponding ranges of the virtual address space are not used



Two-level Page Tables



- Disadvantage: multiple page faults
 - Accessing a PTE page table can cause a page fault
 - Accessing the actual page can cause a second page fault
 - TLB plays an even more important role

Putting it All Together



Example: Intel P6 Processor



- 32 bit address space with 4KB pages
- 2-level cache hierarchy
 - L1 I/D: 16KB, 4-way, 128 sets, 32B blocks
 - L2: 128KB – 2MB
- TLBs
 - I-TLB: 32 entries, 4-way, 8 sets
 - D-TLB: 64 entries, 4-way, 16 sets
 - 32 entries, 8 sets



P6 2-level Page Table Structure

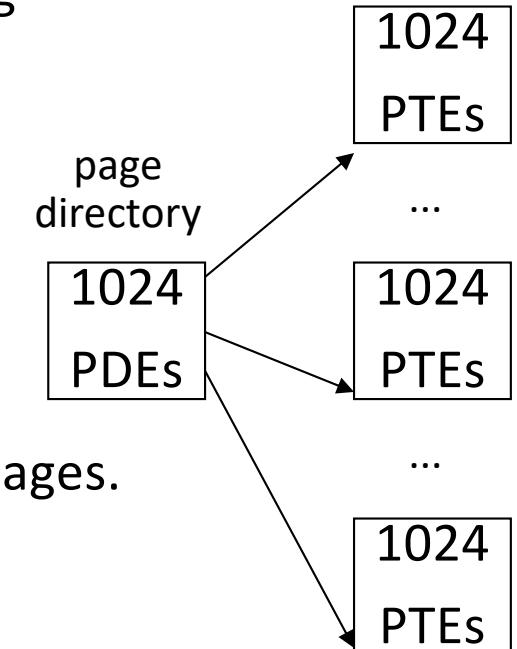
■ Page directory

- One page directory per process.
- 1024 4-byte page directory entries (PDEs) that point to page tables
- Page directory must be in memory if process running
- Always pointed to by PDBR

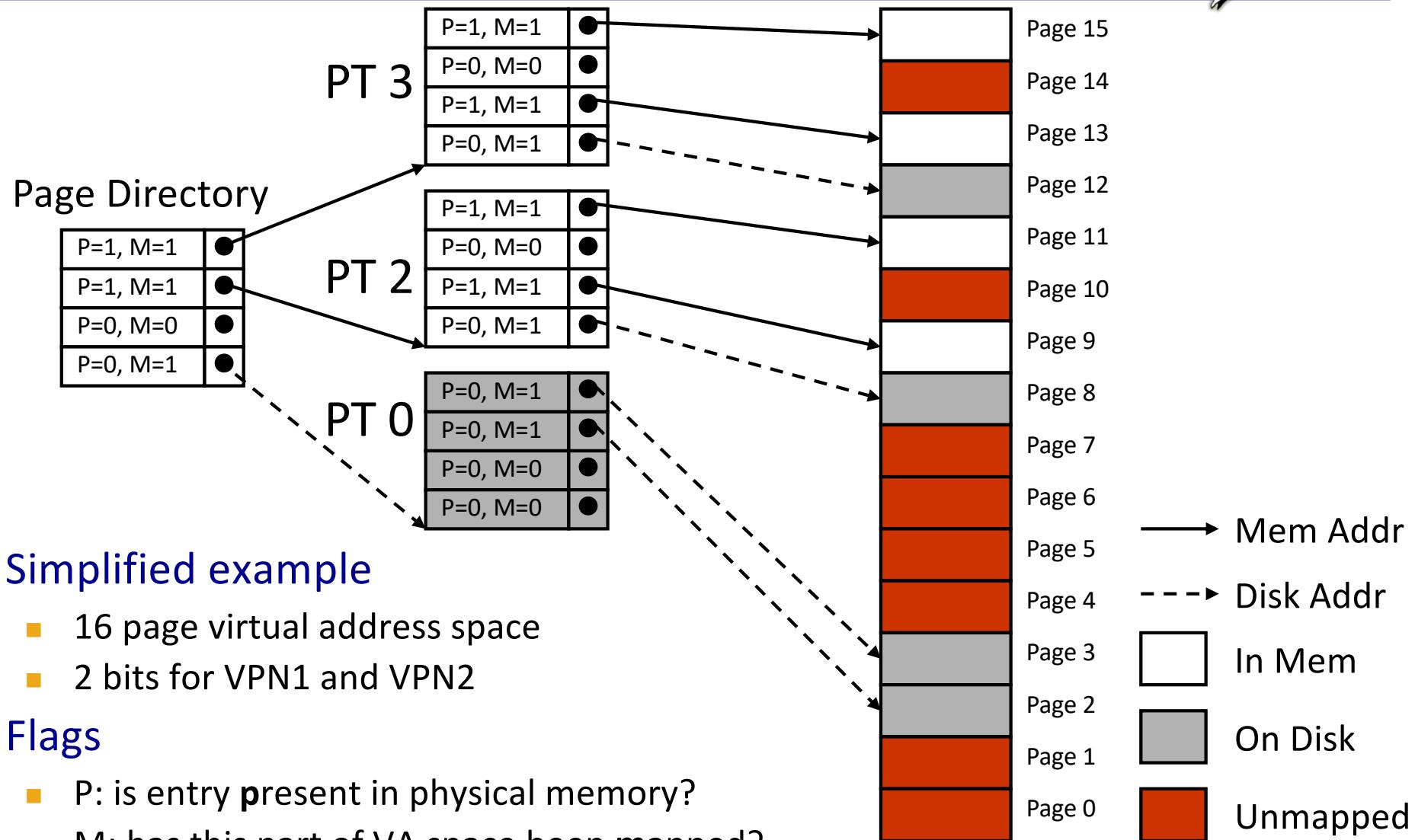
Up to
1024
page
tables

■ Page tables

- 1024 4-byte page table entries (PTEs) that point to pages.
- Page tables can be paged in and out

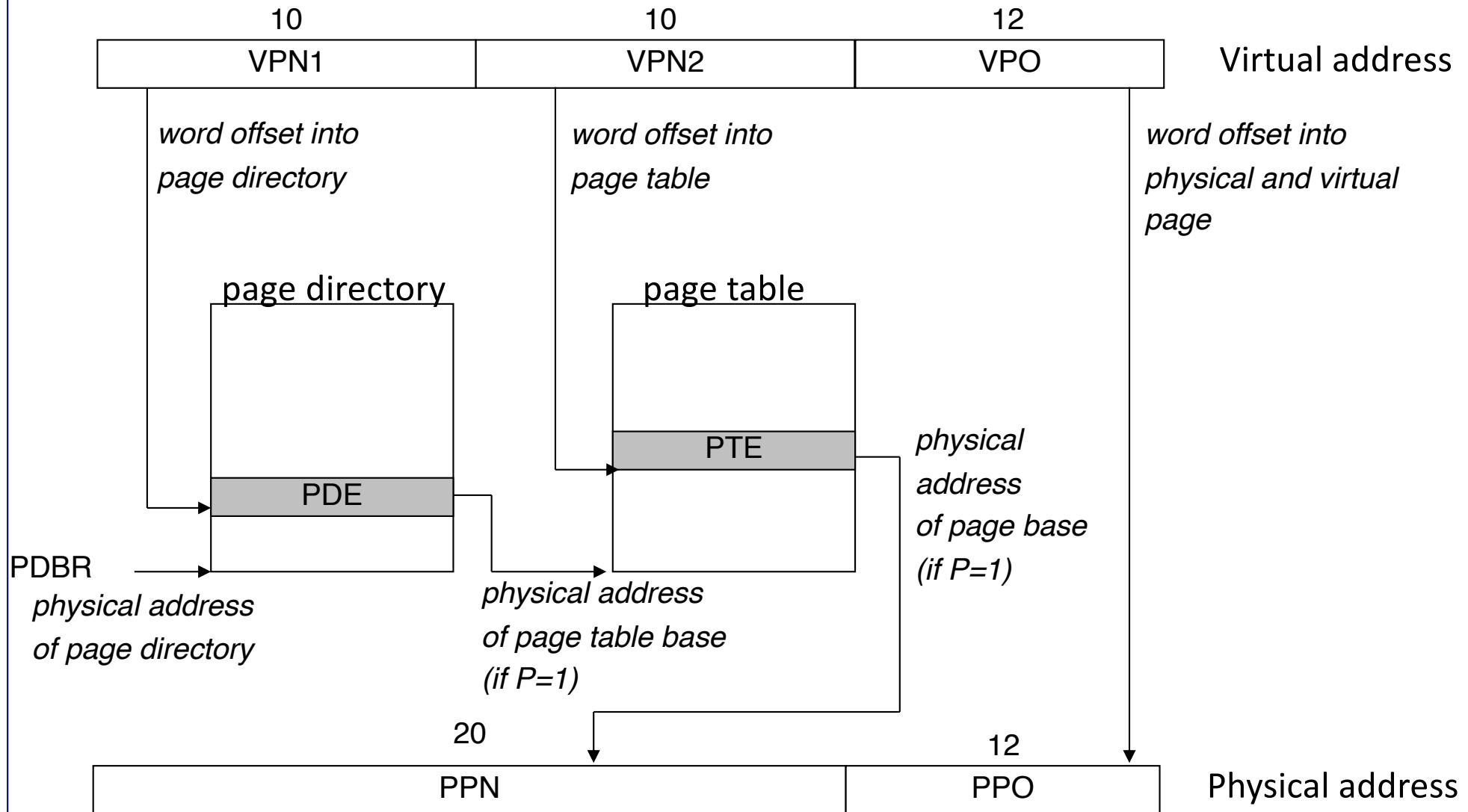


Representation of Virtual Address Space



- Simplified example
 - 16 page virtual address space
 - 2 bits for VPN1 and VPN2
- Flags
 - P: is entry present in physical memory?
 - M: has this part of VA space been mapped?

P6 Translation





P6 Page Directory Entry (PDE)

31	12 11	9	8	7	6	5	4	3	2	1	0
Page table physical base addr	Avail	G	PS		A	CD	WT	U/S	R/W	P=1	

Page table physical base address: 20 most significant bits of physical page table address (forces page tables to be 4KB aligned)

Avail: available for system programmers

G: global page (don't evict from TLB on task switch)

PS: page size 4K (0) or 4M (1)

A: accessed (set by MMU on reads and writes, cleared by software)

CD: cache disabled (1) or enabled (0)

WT: write-through or write-back cache policy for this page table

U/S: user or supervisor mode access

R/W: read-only or read-write access

P: page table is present in memory (1) or not (0)



P6 page table entry (PTE)

31	12	11	9	8	7	6	5	4	3	2	1	0
Page physical base address	Avail	G	0	D	A	CD	WT	U/S	R/W	P=1		

Page base address: 20 most significant bits of physical page address (forces pages to be 4 KB aligned)

Avail: available for system programmers

G: global page (don't evict from TLB on task switch)

D: dirty (set by MMU on writes)

A: accessed (set by MMU on reads and writes)

CD: cache disabled or enabled

WT: write-through or write-back cache policy for this page

U/S: user/supervisor

R/W: read/write

P: page is present in physical memory (1) or not (0)

31	1	0
Available for OS (page location in secondary storage)		P=0



Abbreviations

■ MMU

- Memory management unit: controls TLB, handles TLB misses

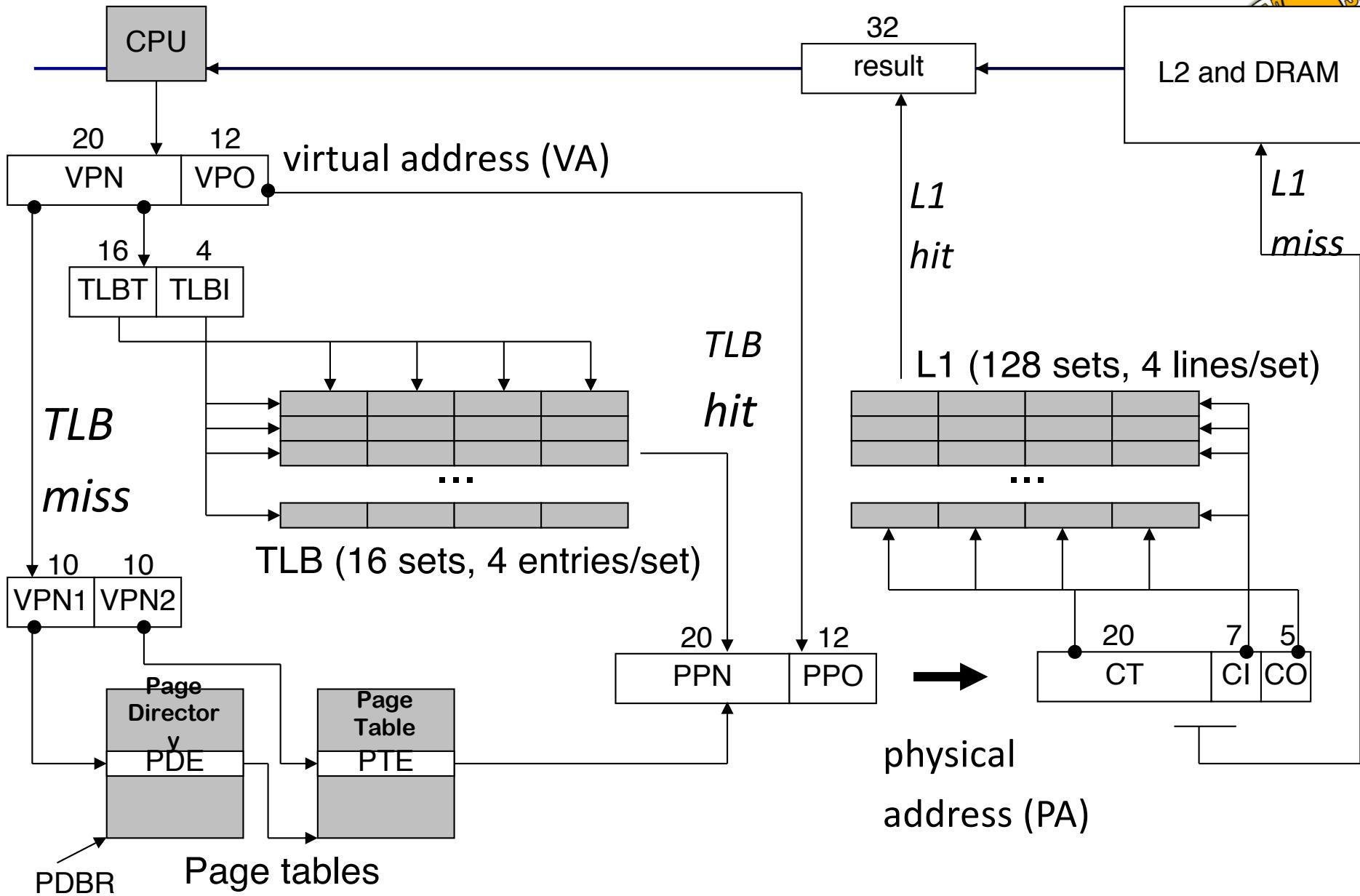
■ Components of the virtual address (VA)

- TLBI: TLB index
- TLBT: TLB tag
- VPO: virtual page offset
- VPN: virtual page number

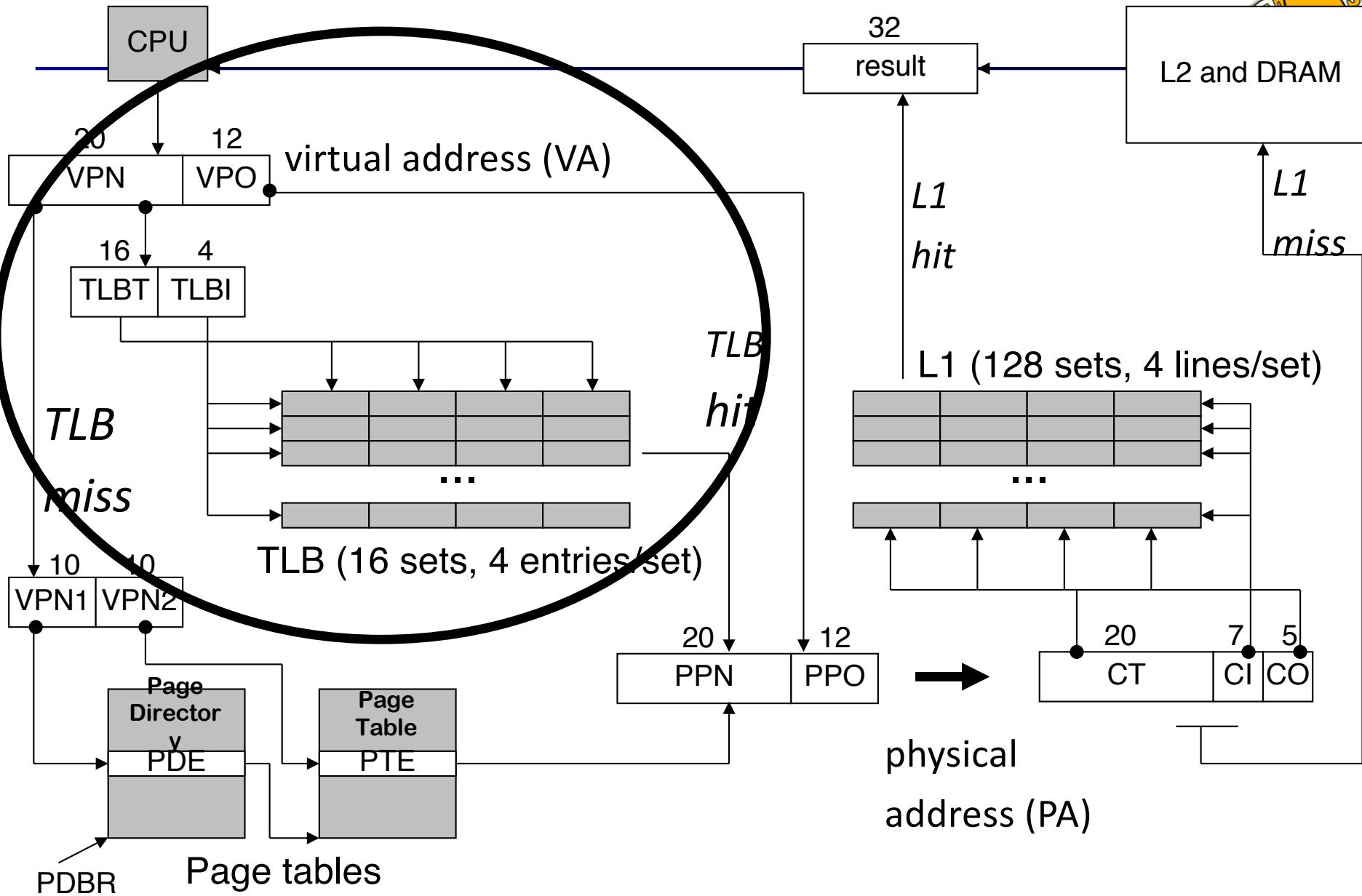
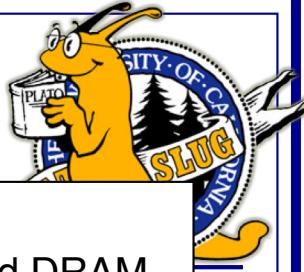
■ Components of the physical address (PA)

- PPO: physical page offset (same as VPO)
- PPN: physical page number
- CO: byte offset within cache line
- CI: cache index
- CT: cache tag

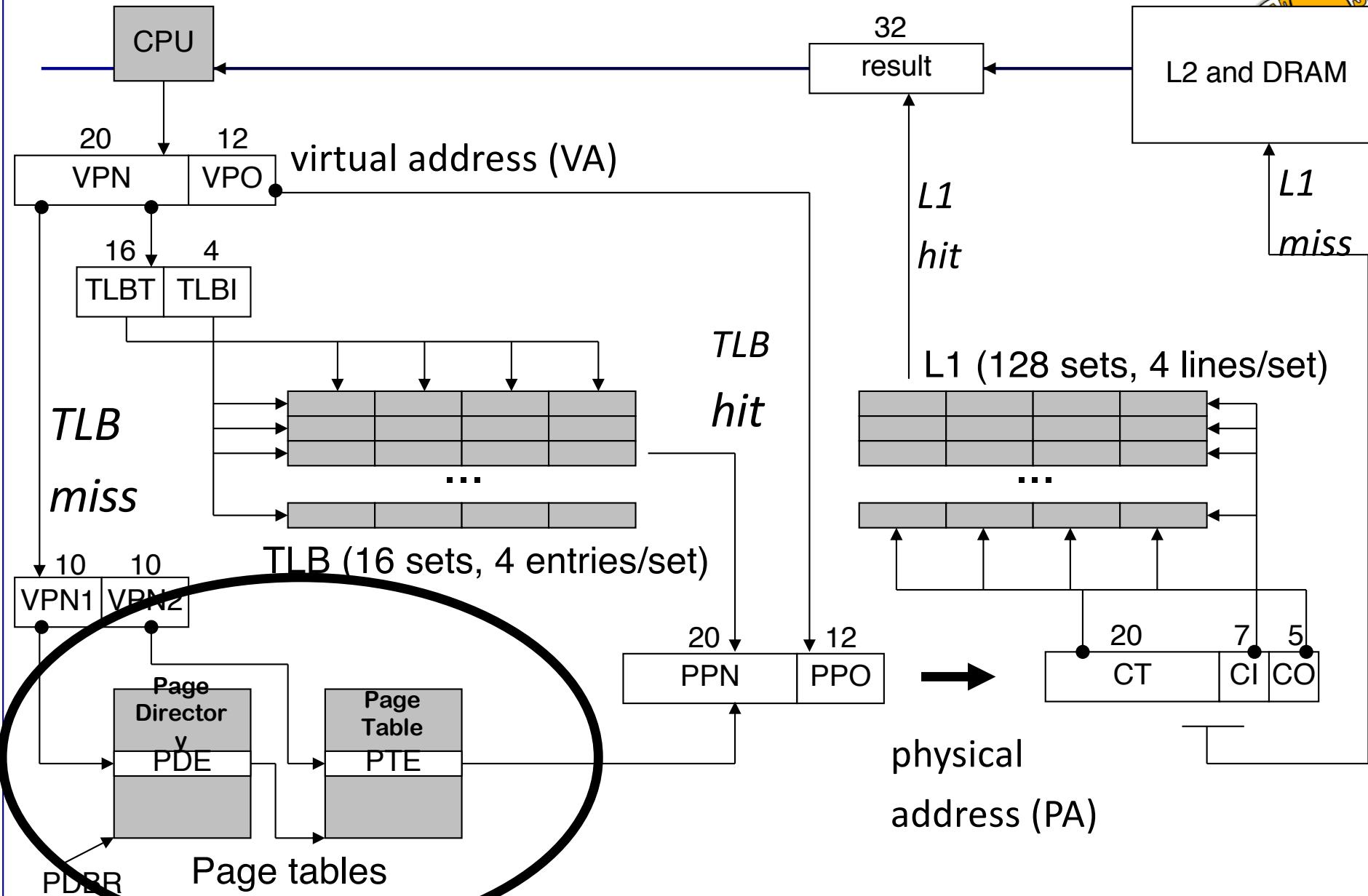
Overview of P6 Address Translation



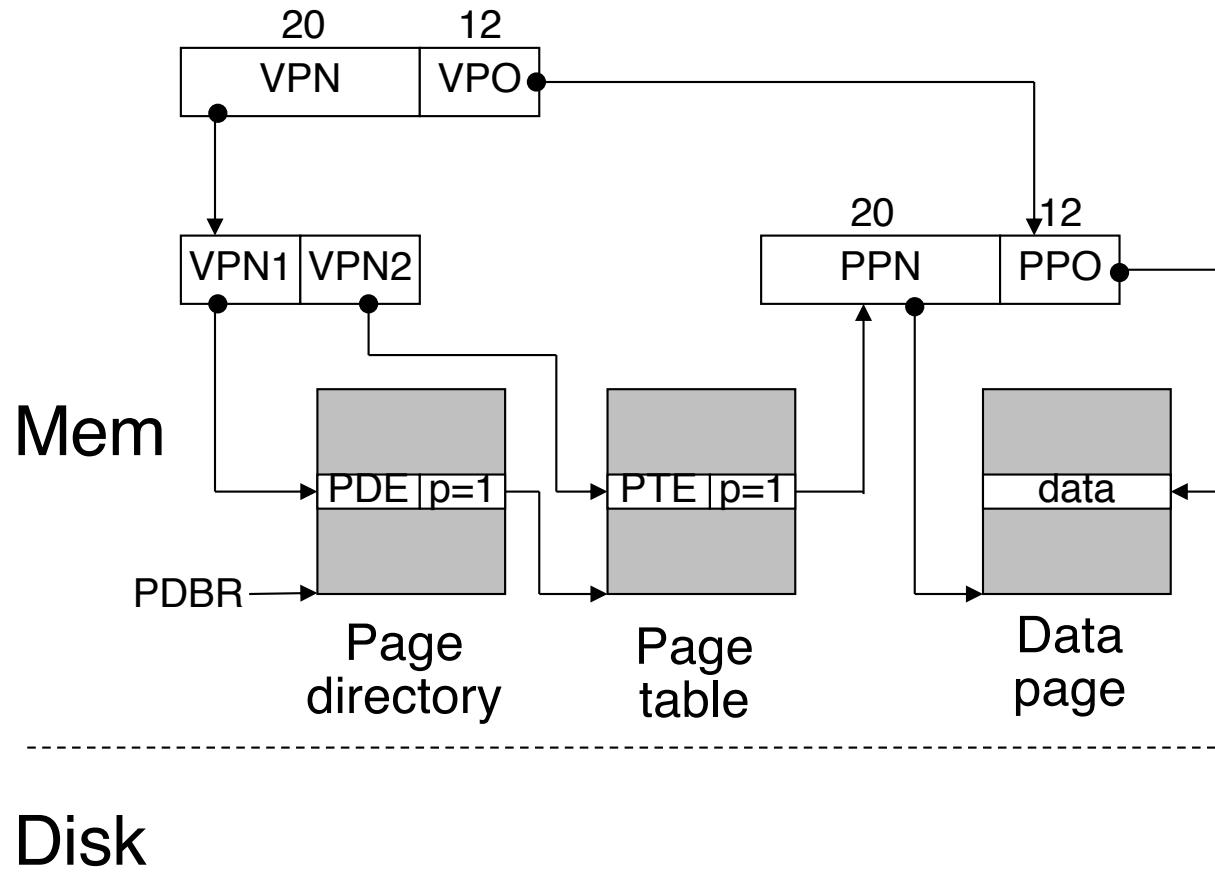
TLB Translation



Page Table Translation

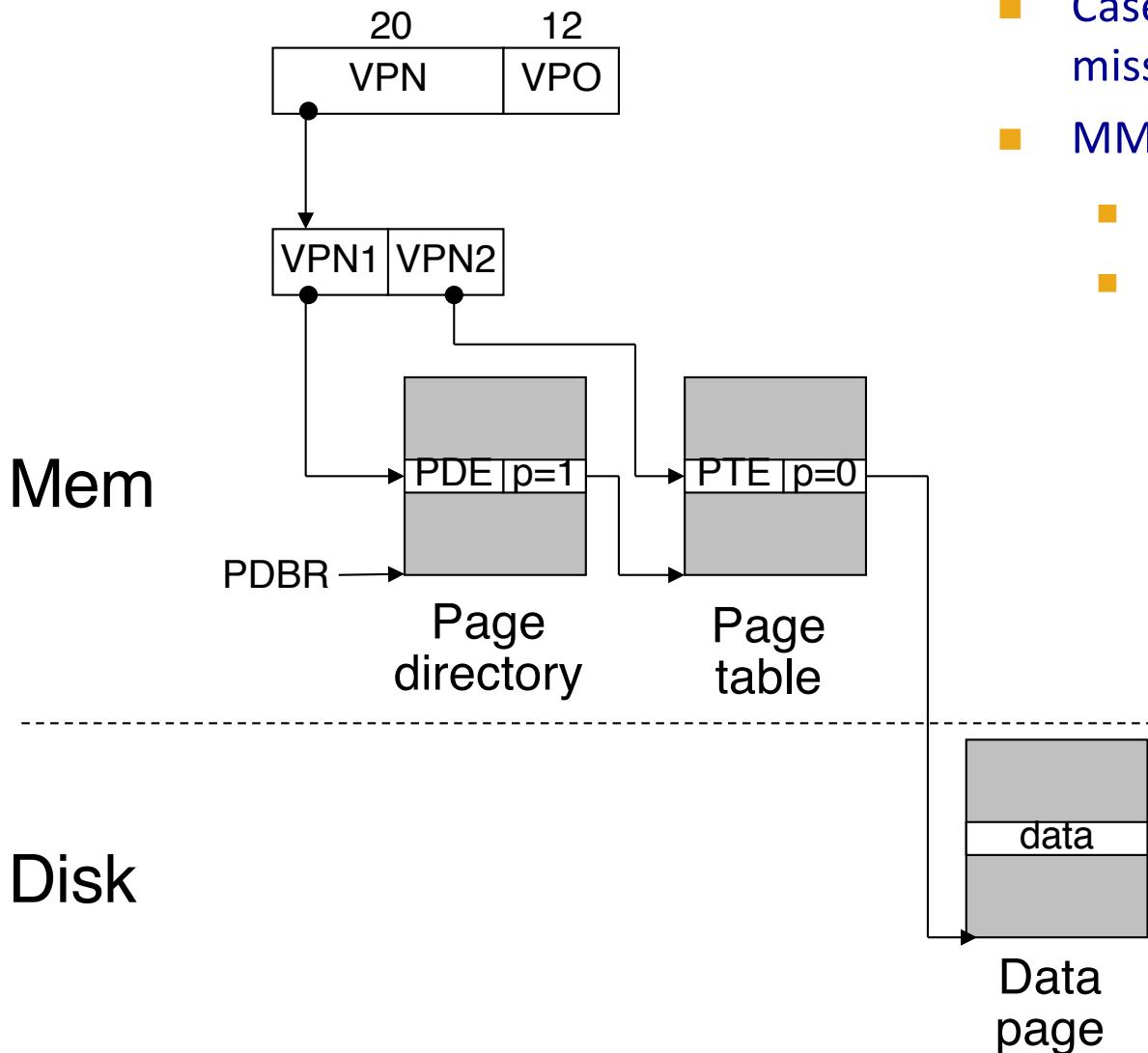


Translating with the P6 page tables (case 1/1)



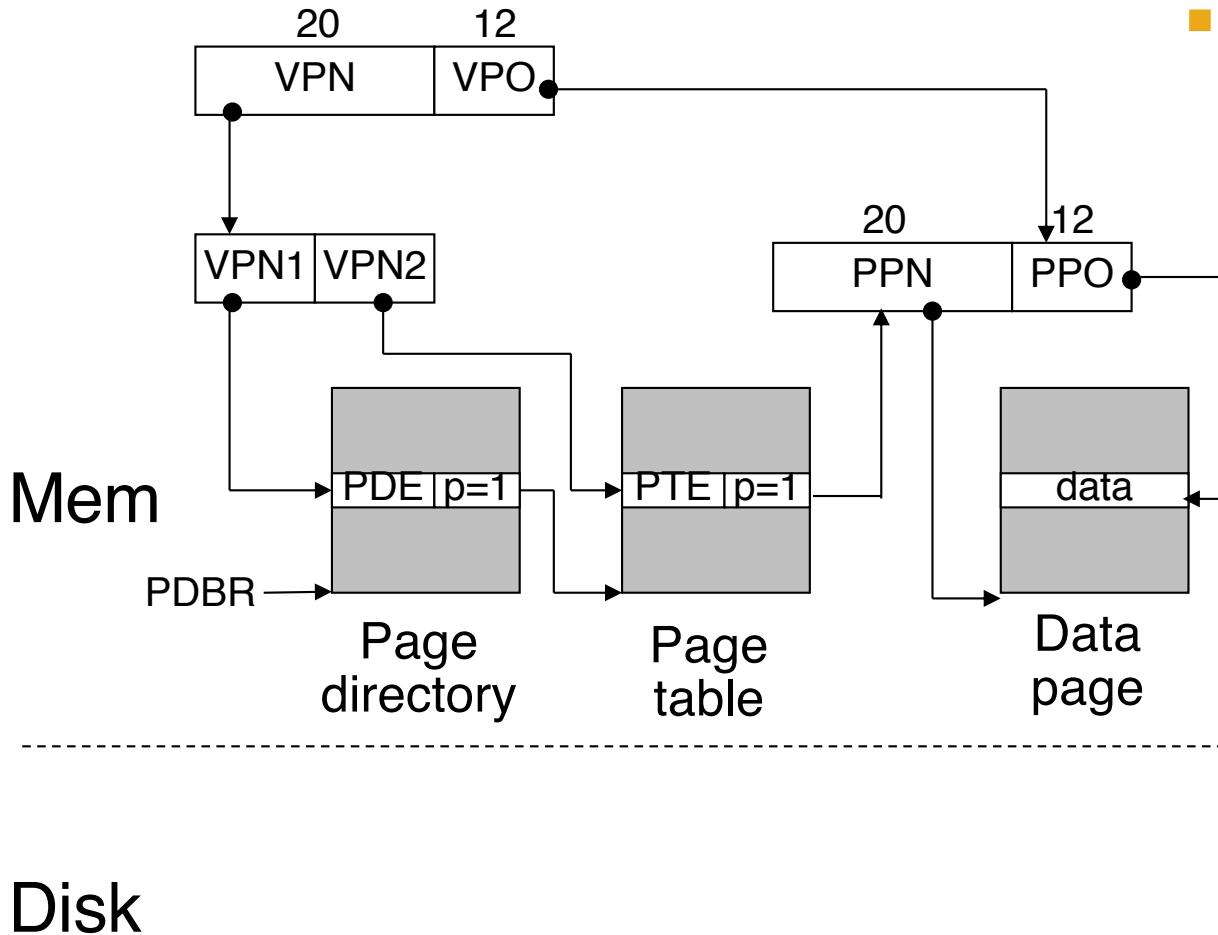
- Case 1/1: page table and page present.
- MMU Action:
 - MMU build physical address and fetch data word.
- OS action
 - none

Translating with the P6 page tables (case 1/0)



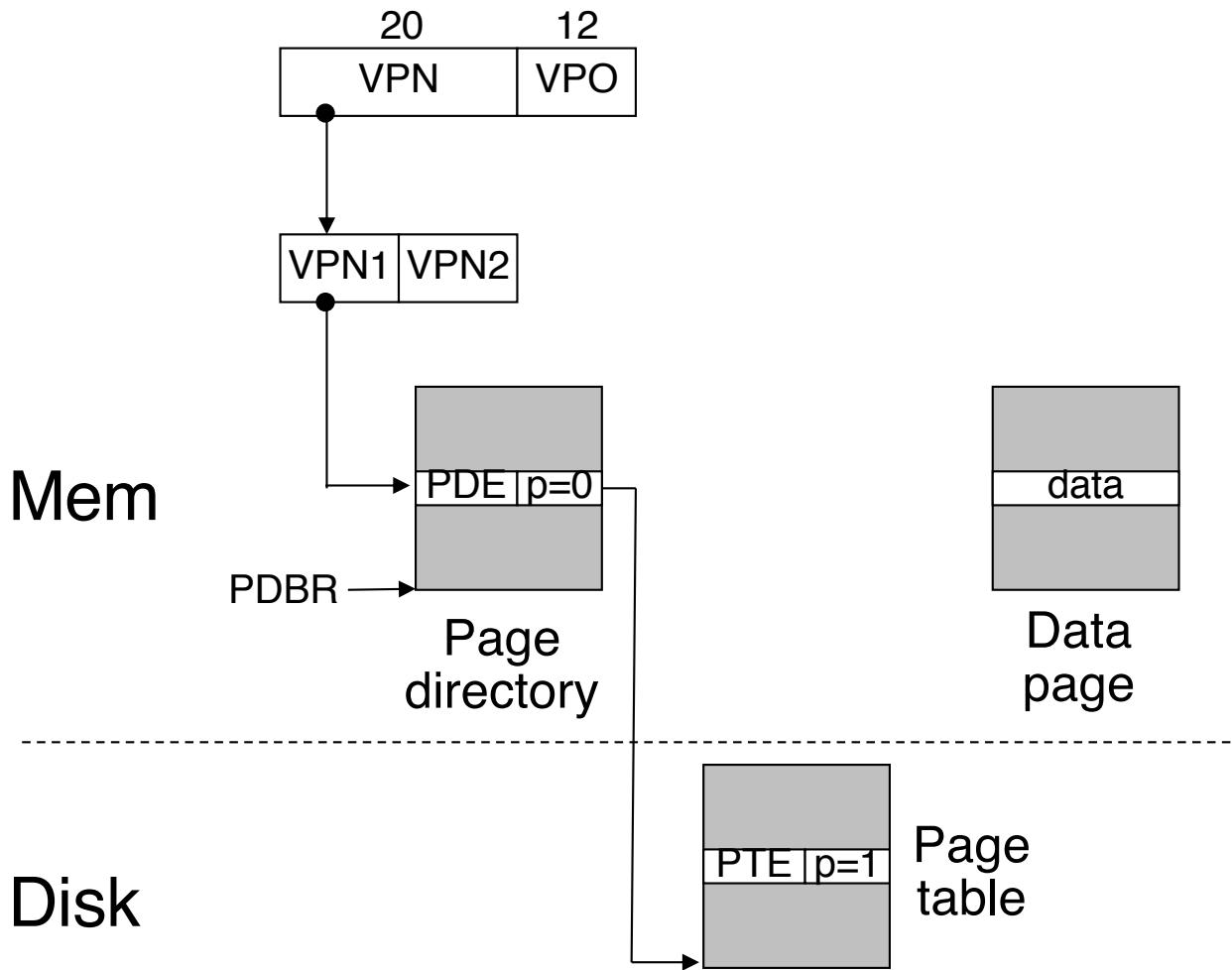
- Case 1/0: page table present but page missing.
- MMU Action:
 - page fault exception
 - handler receives the following args:
 - VA that caused fault
 - fault caused by non-present page or page-level protection violation
 - read/write
 - user/supervisor

Translating with the P6 page tables (case 1/0, cont)



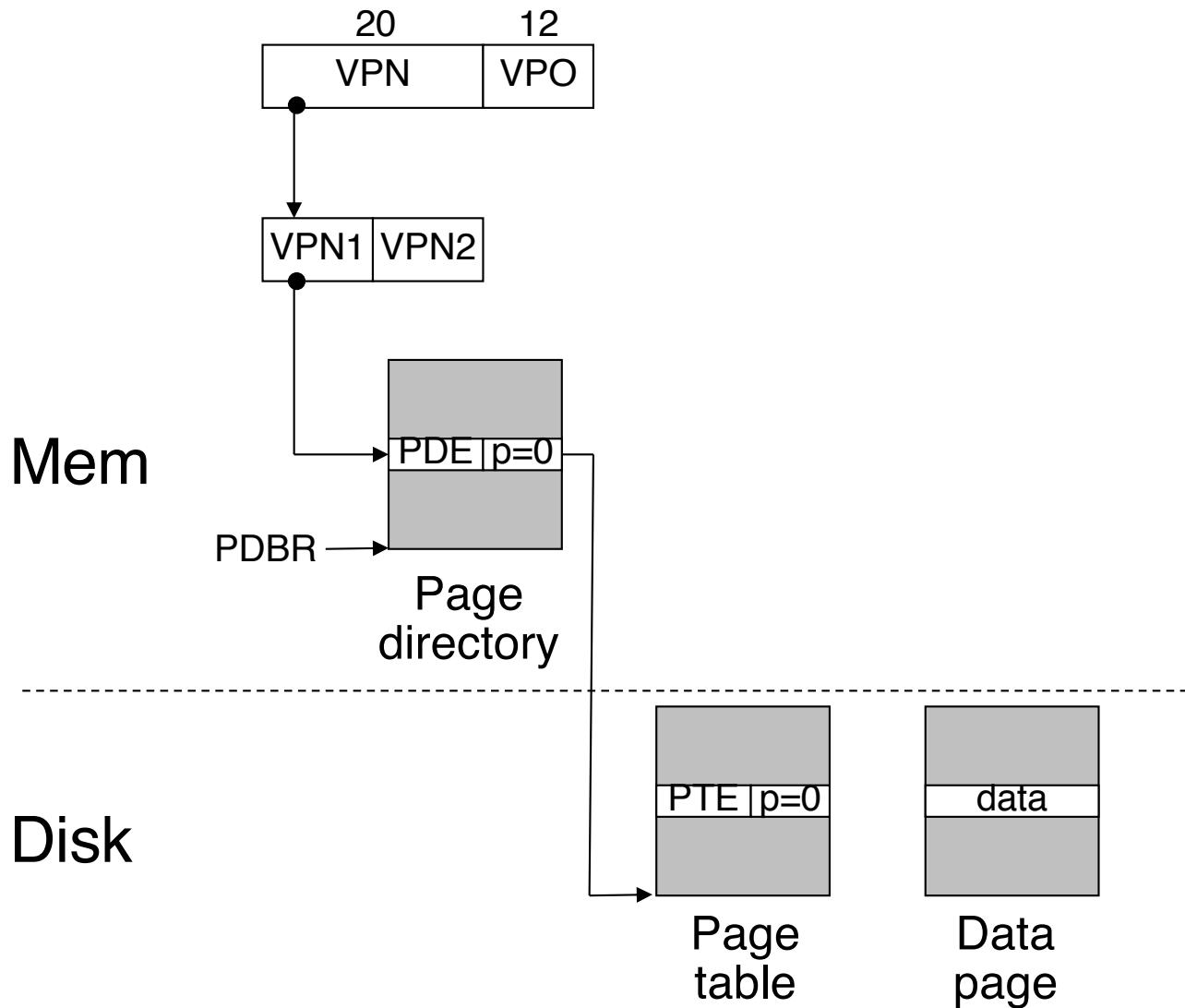
- OS Action:
 - Check for a legal virtual address.
 - Read PTE through PDE.
 - Find free physical page (swapping out current page if necessary)
 - Read virtual page from disk and copy to physical page
 - Restart faulting instruction by returning from exception handler.

Translating with the P6 page tables (case 0/1)

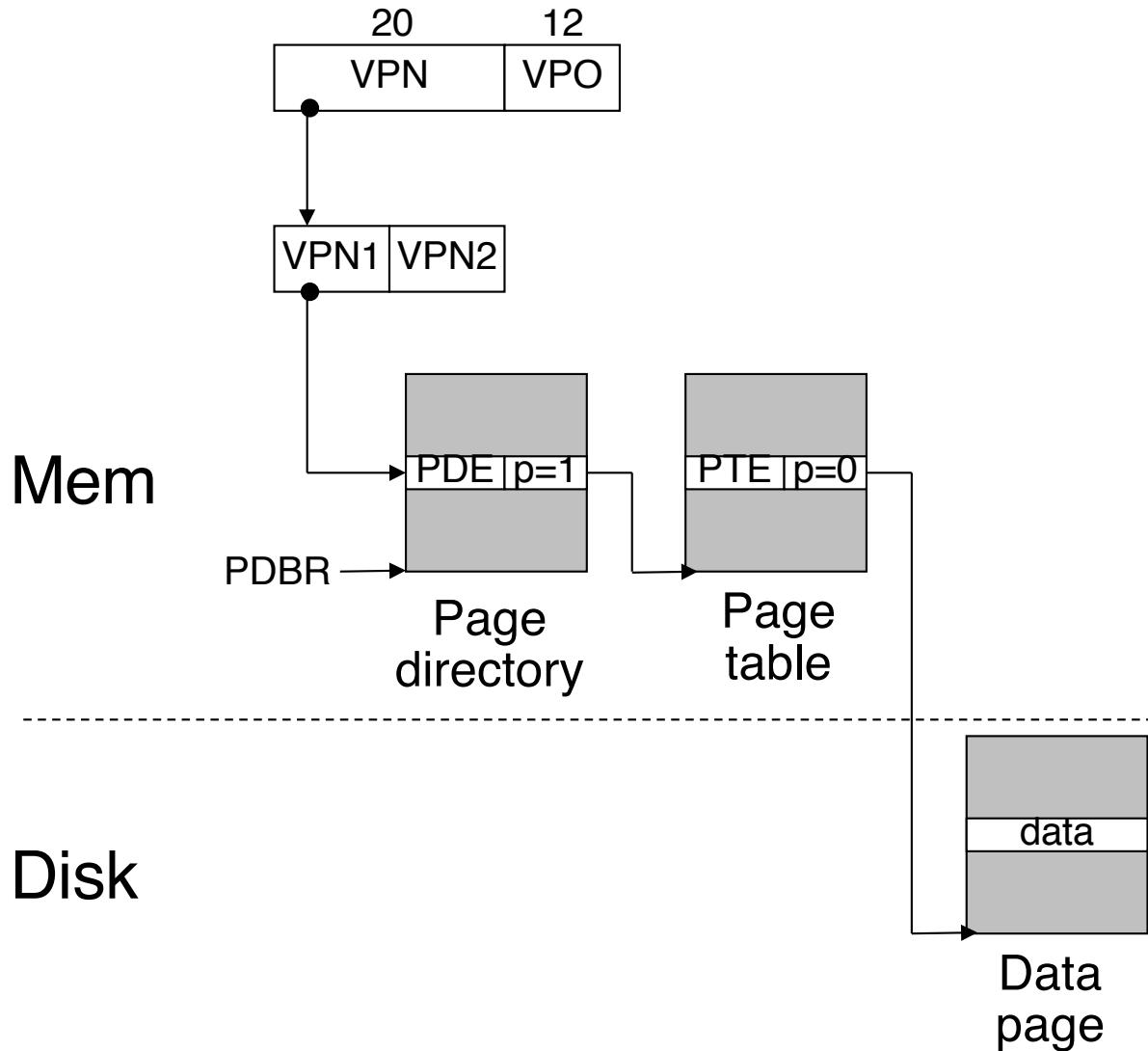


- Case 0/1: page table missing but page present.
- Introduces consistency issue.
 - potentially every page out requires update of disk page table.
- Linux disallows this
 - if a page table is swapped out, then swap out its data pages too.

Translating with the P6 page tables (case 0/0)

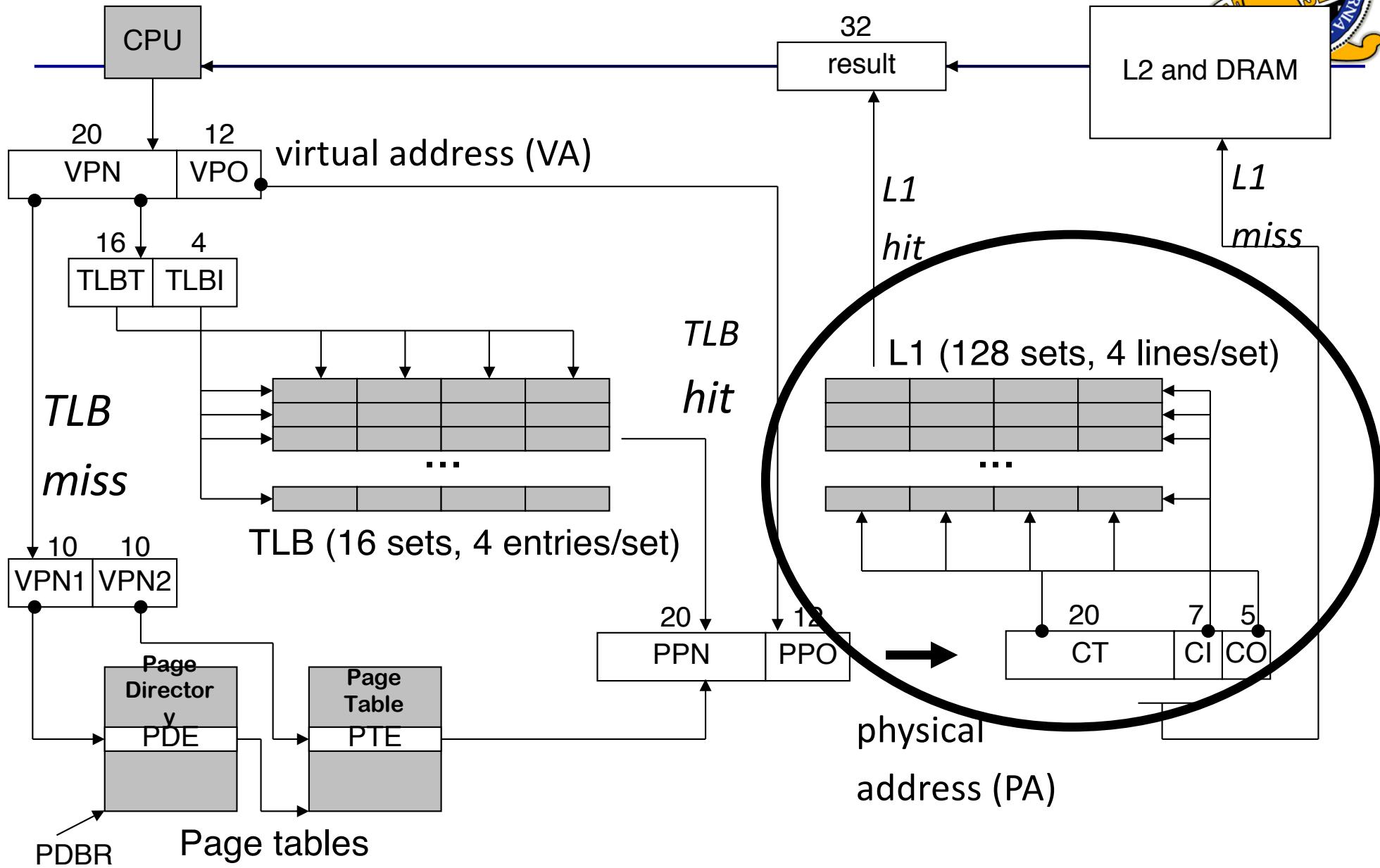
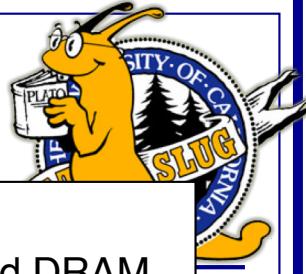


Translating with the P6 page tables (case 0/0, cont)

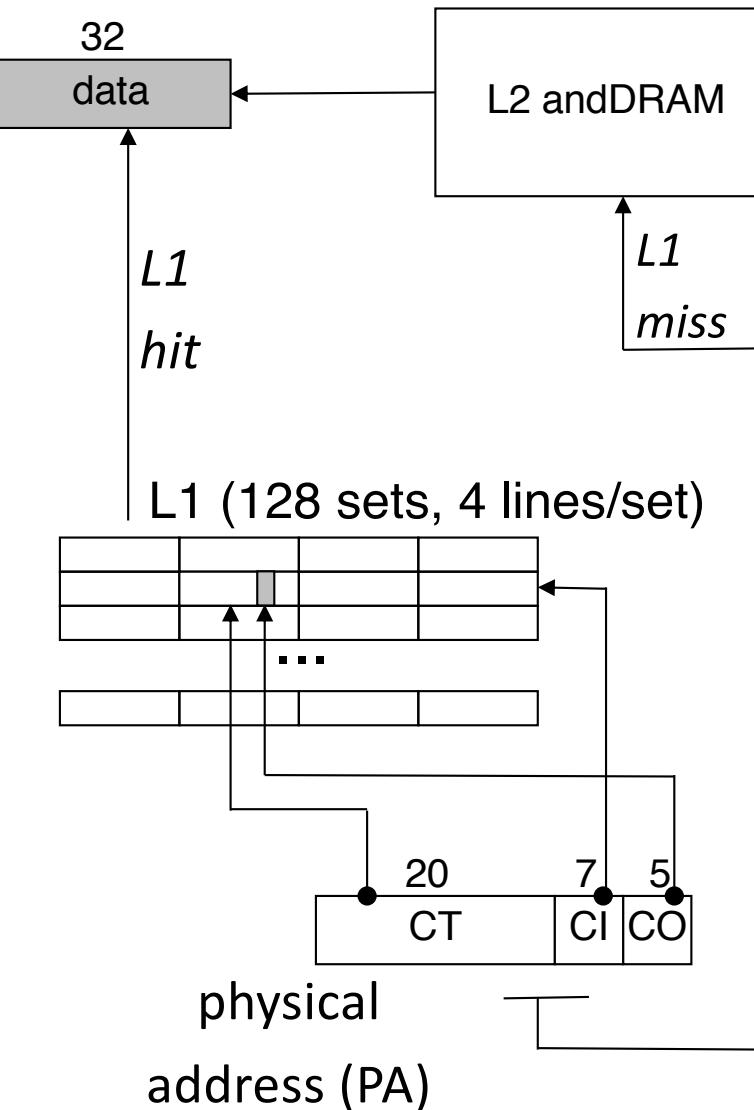


- OS action:
 - swap in page table.
 - restart faulting instruction by returning from handler.
- Like case 0/1 from here on.

L1 Cache Access

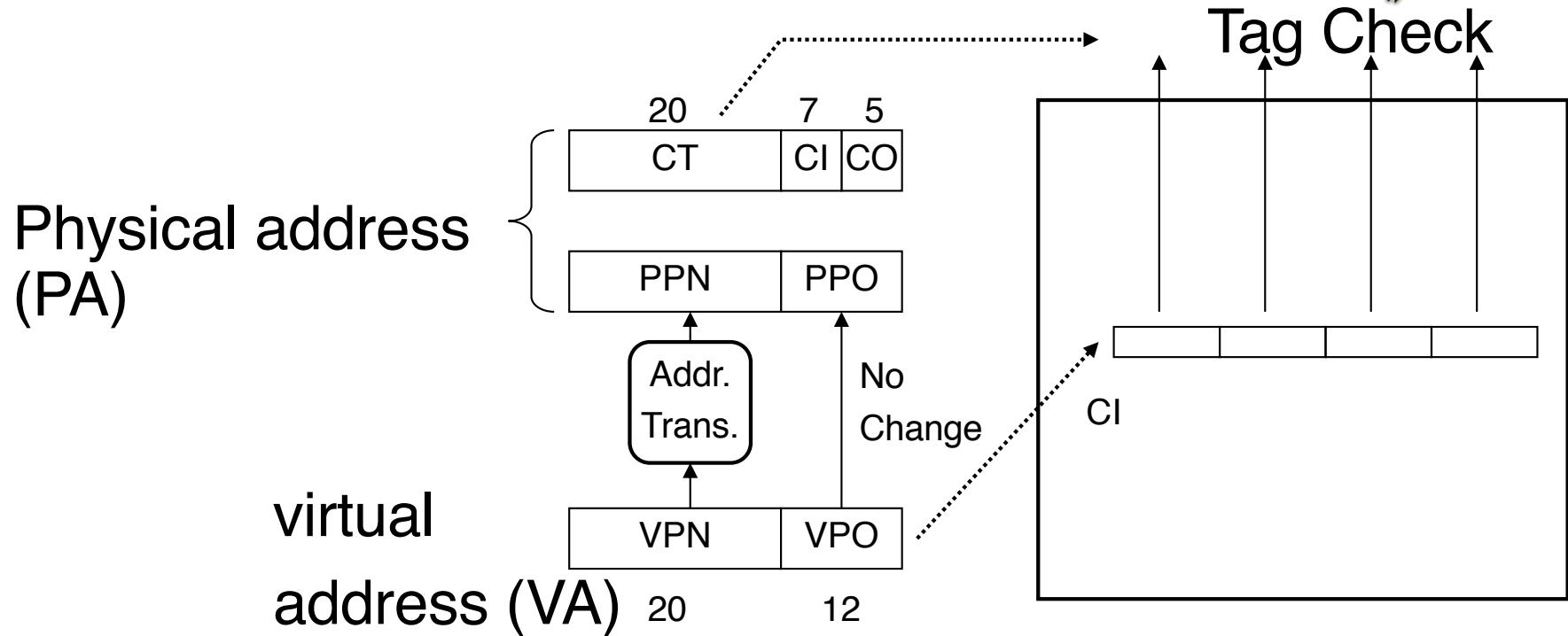


L1 Cache Access



- Partition PA into CO, CI, and CT
- Use CT to if line at set CI is hit or miss
 - If miss, check L2
 - If hit, extract word at byte offset CO and return to processor.

Speeding Up L1 Access



Observation

- Bits that determine CI identical in virtual and physical address
- Can index into cache while address translation being performed
- Then check with CT from physical address
- "Virtually indexed, physically tagged"



Virtual Memory Summary

Use hard disk (or Flash) as a large storage for data of all programs

Main memory (DRAM) is a cache for the disk

Managed jointly by hardware and the operating system (OS)

Each running program has its own virtual address space

Address space as shown in previous figure ($0 - 2^{32}-1$ bytes)

Protected from other programs

Frequently-used portions of virtual address space copied to DRAM

DRAM = physical address space

Hardware + OS translate virtual addresses (VA) used by program to physical addresses (PA) used by the hardware

Translation enables relocation (DRAM<->disk) & protection

Hardware Modes (Protecting OS from User Programs)



2 modes are needed, but some architectures have more

User mode: Used to run user processes

Accessible instructions: user portion of the ISA

The instructions we have seen so far

Accessible state: user registers and memory

But virtual memory translation is always on

Cannot access EPC, Cause, ... registers

Exceptions and interrupts are always on

Kernel mode: used to run (most of) the kernel

Accessible instructions: the whole ISA

User + privileged instructions

Accessible state: everything

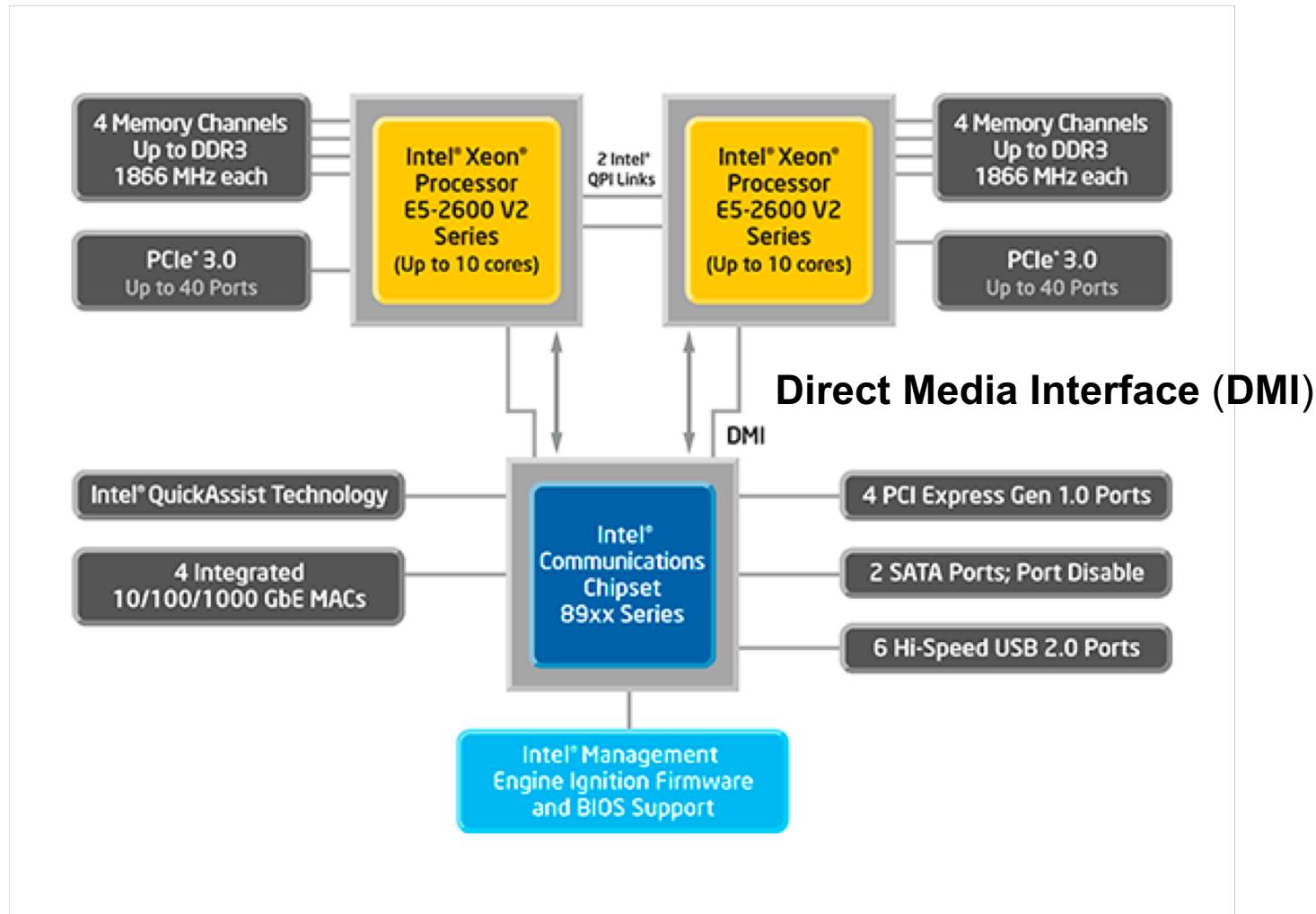
Virtual memory, exceptions, and interrupts may be turned off

Input/Output (I/O)





Typical Server I/O



I/O Technologies



- I/O devices
- Hardware interfaces and mechanisms
- Software interfaces and mechanisms



I/O Devices Diversity

- I/O devices can be characterized by

- Behavior: input, output, storage
- Partner: human or machine
- Data rate: bytes/sec, transfers/sec
- Required response time

Device	Behavior	Partner	Data Rate (KB/sec)
Keyboard	Input	Human	0.01
Mouse	Input	Human	0.02
Line Printer	Output	Human	1.00
Laser Printer	Output	Human	100.00
Graphics	Output	Human	100,000.00
Network-LAN	Communication	Machine	10,000.00
Floppy disk	Storage	Machine	50.00
Optical Disk	Storage	Machine	10,000.00
Magnetic Disk	Storage	Machine	30,000.00

Question



- What does the wide diversity imply?
 - For HW or for SW?

I/O Metrics



■ Throughput

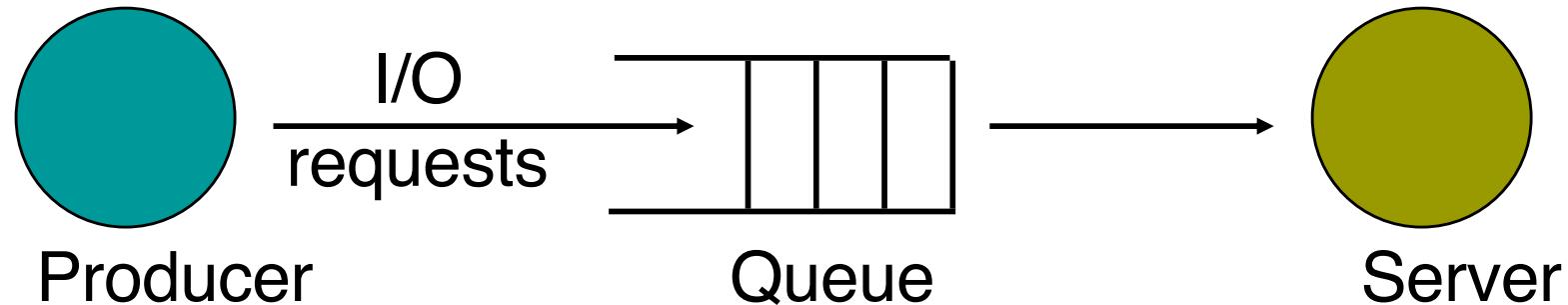
- Aggregate measure of amount of data moved per unit time
 - Bytes/sec or transfers/sec, average over a window
- Sometimes referred to as bandwidth
 - Examples: Memory bandwidth, disk bandwidth

■ Response time or latency

- Response time to do a single I/O operation
 - Measured in seconds or cycles
 - Example: Write a block of bytes to disk
 - Example: Send a data packet over the network
- What happens if you don't respond in time
 - Retry request, loose data forever, or blow up engine ☺

■ Other metrics?

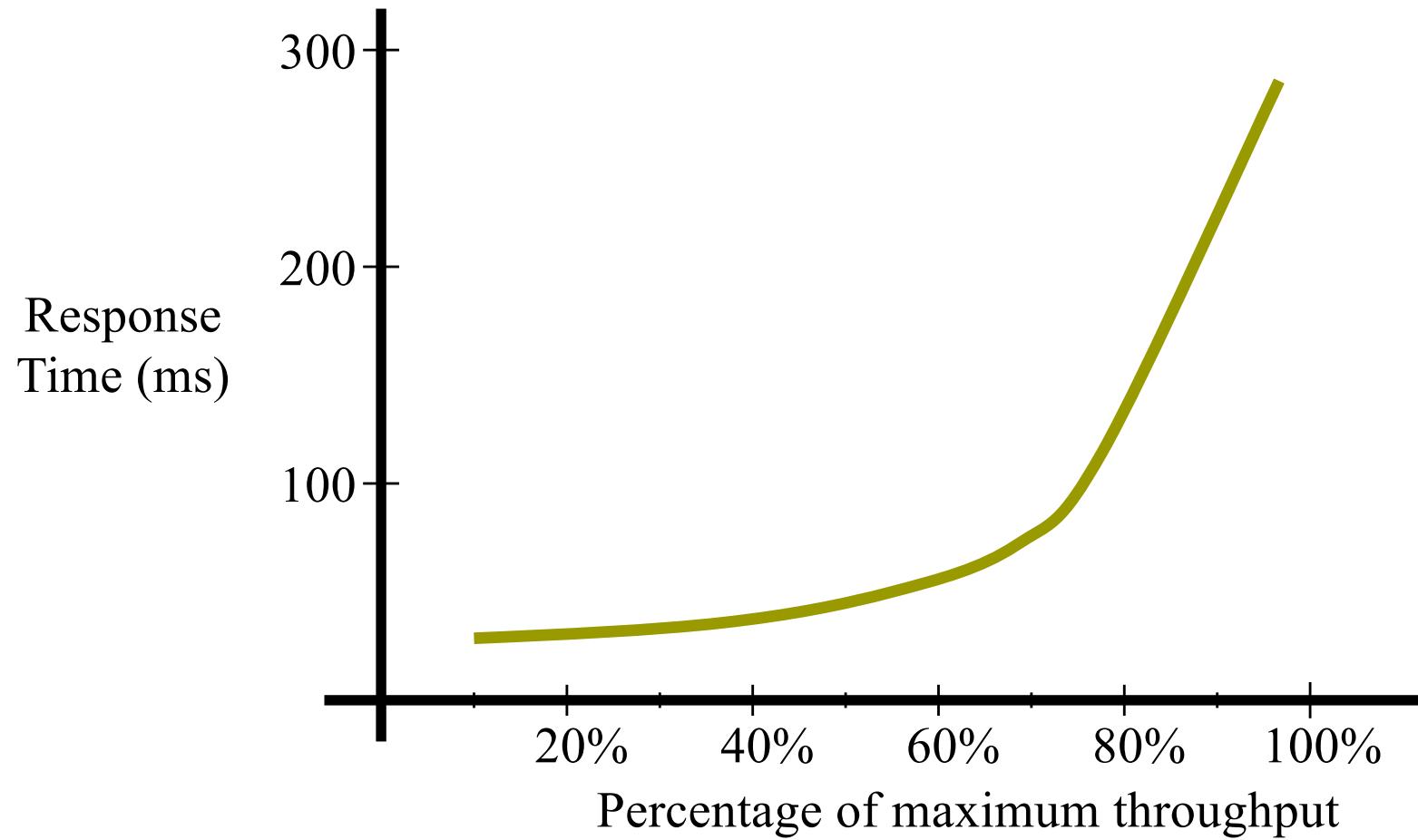
Throughput – Latency Tradeoffs



- Tradeoff between throughput and response time
 - Highest throughput when server is always busy and queue is never empty
 - Lowest latency when queue is empty and server is idle when task arrives
- Buffering/queuing can be helpful



Throughput – Latency Tradeoff



Magnetic Hard Disks



■ Characteristics

- Long term, nonvolatile storage
- Large, inexpensive, but slow (mechanical device)

■ Usage

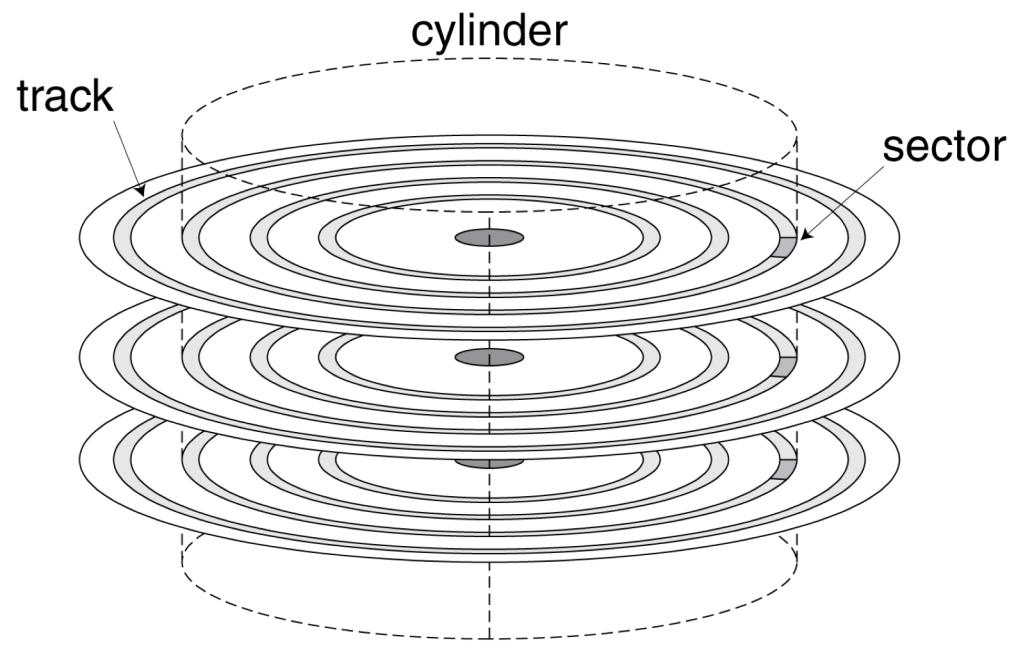
- Virtual memory (swap area)
- File system



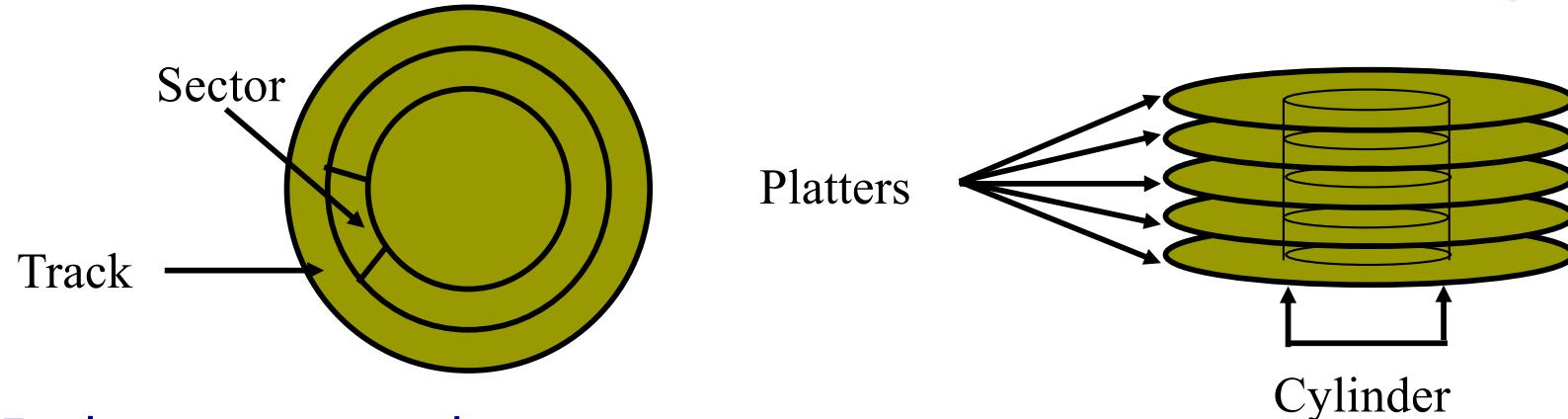
Magnetic Hard Disks



- Nonvolatile, rotating magnetic storage



Hard Disk Organization



- Each sector records
 - Sector ID and data (512 bytes, 4KB proposed)
 - Error correcting code (ECC), synchronization fields and gaps
- Access to a sector involves
 - Queuing delay if other accesses are pending
 - Seek: move the heads
 - Rotational latency
 - Data transfer
 - Controller overheads