

CSE261: Computer Architecture

18. Memory Hierarchy (4): Virtual Memory #1

Seongil Wi

Notification: Final Exam



- Date: Dec. 19 (Thursday)
- Class Time (1h 15m), Closed book
- T/F problems + Computation problems + Descriptive problems
- Scope: All the material learned *after the midterm*

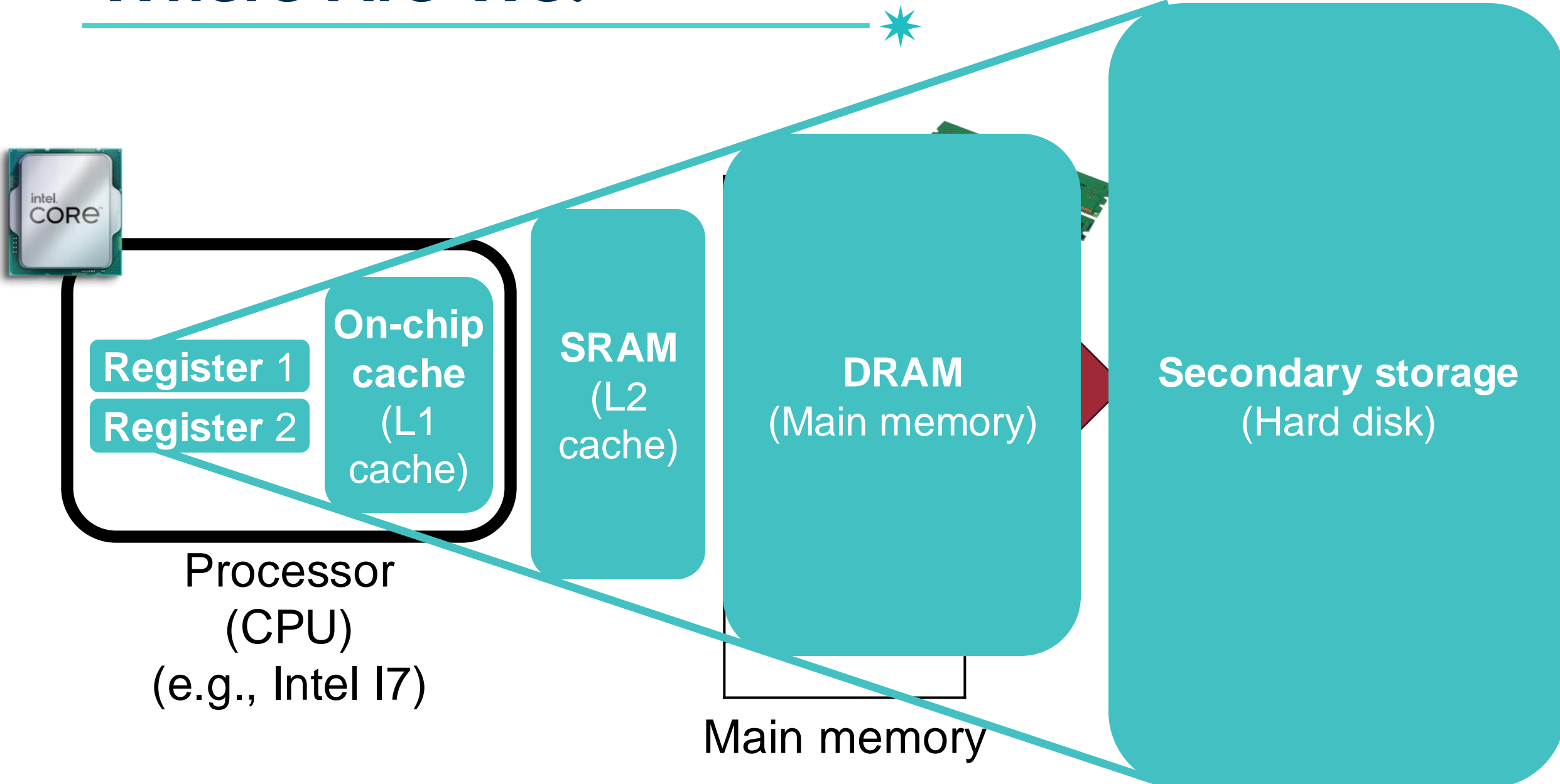
Notification: HW3



- Implementing a cache simulator
 - Due: Dec 15, 11:59 PM
-
- Since the final exam period is approaching, it is recommended to complete this assignment as soon as possible

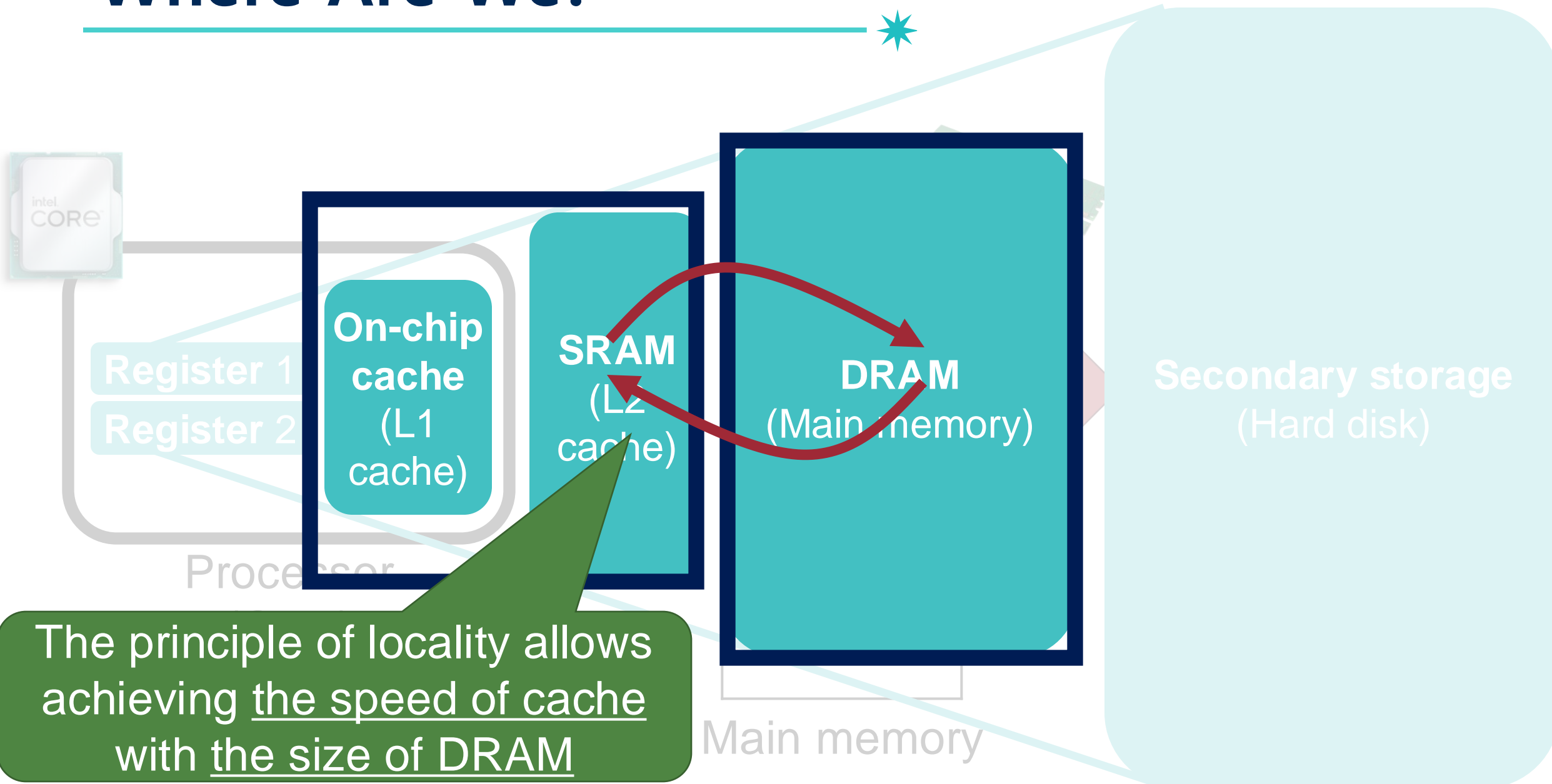
Where Are We?

4

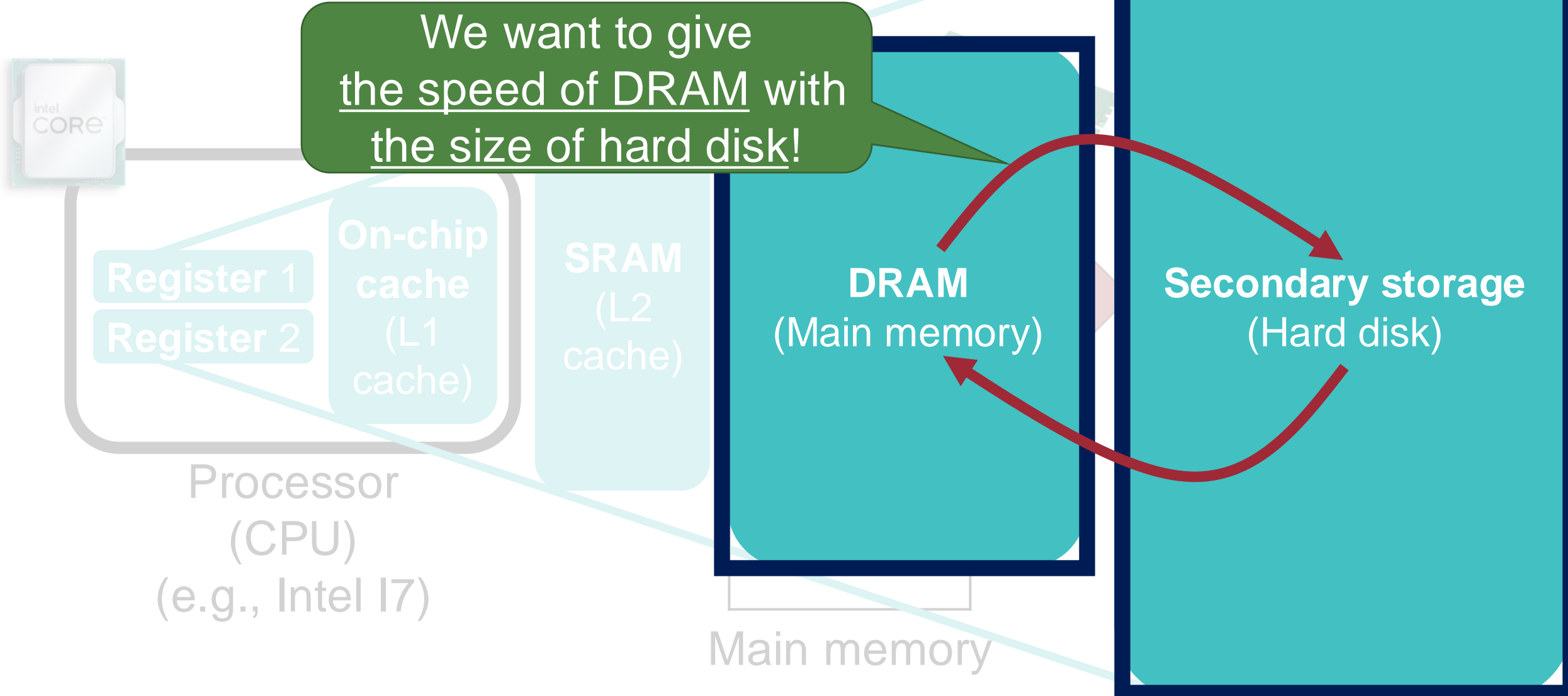


Where Are We?

5



Today's Topic: DRAM/HDD



Introduction to Virtual Memory

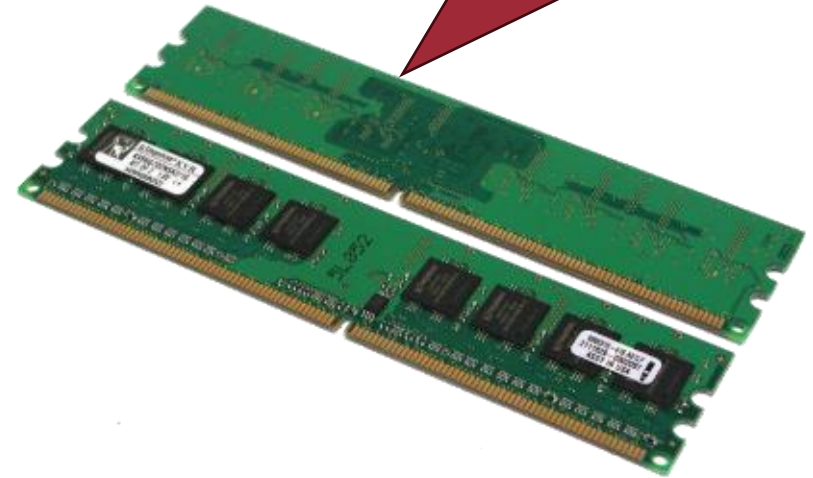
In the Physical World, We Have Limited-sized Memory

8



1w \$1, 4(\$3)

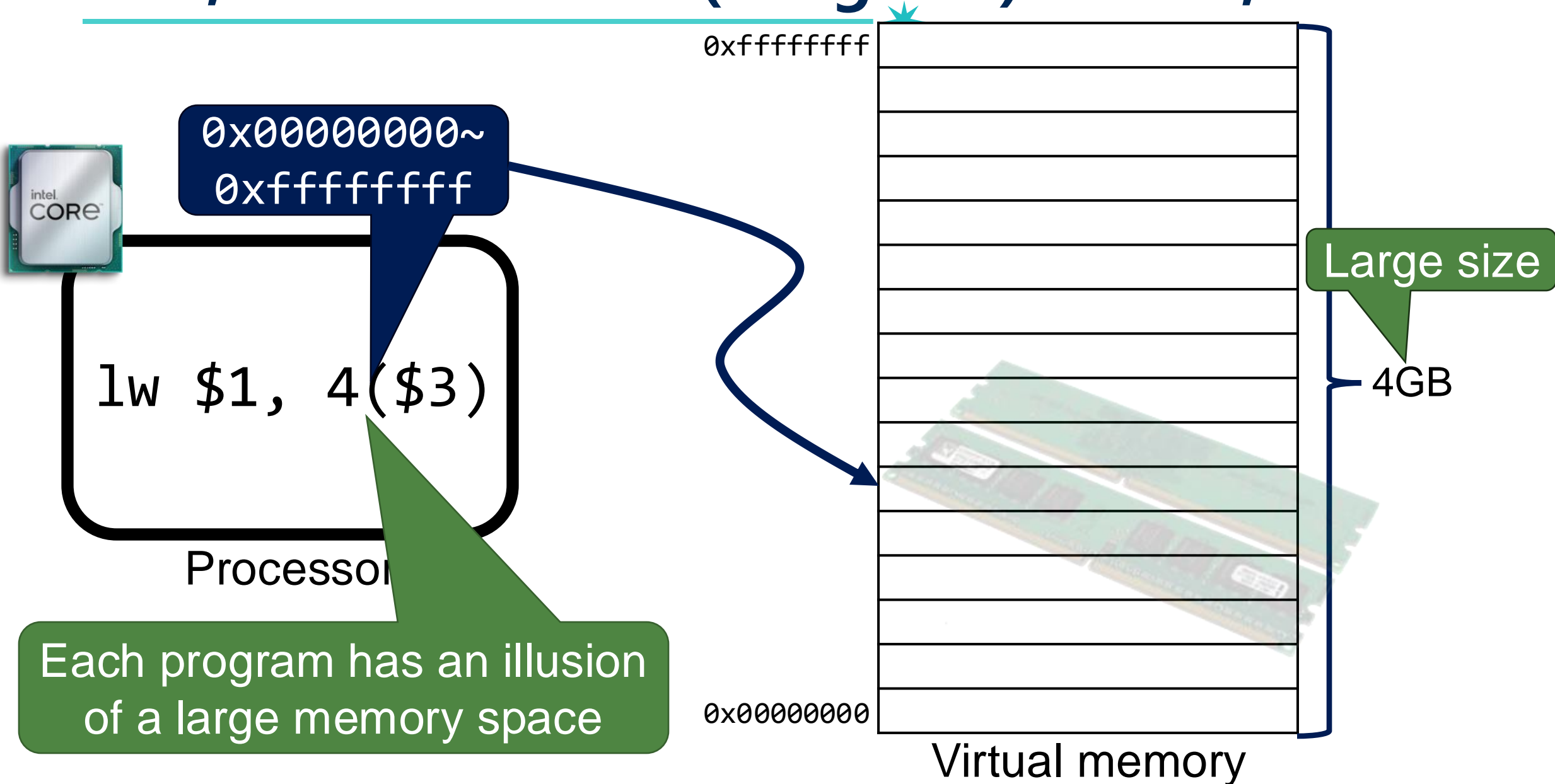
Processor



Very small size
(e.g., 256MB)

Physical memory

But, In the Virtual (Program) World, ...



Virtual Memory



- Give programmers **an illusion of a large memory space** irrespective of actual capacity

Virtual Memory



- Give programmers **an illusion of a large memory space** irrespective of actual capacity

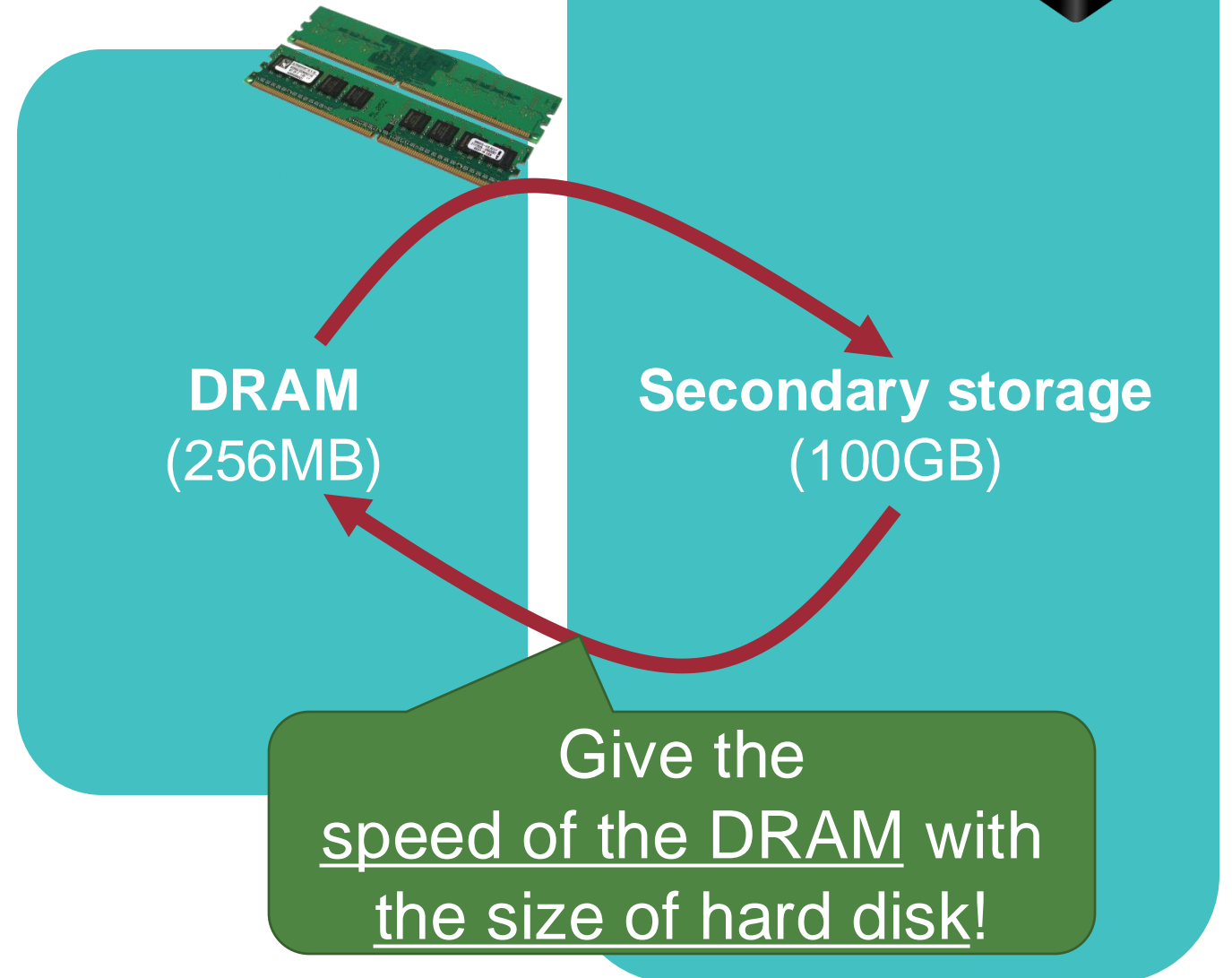
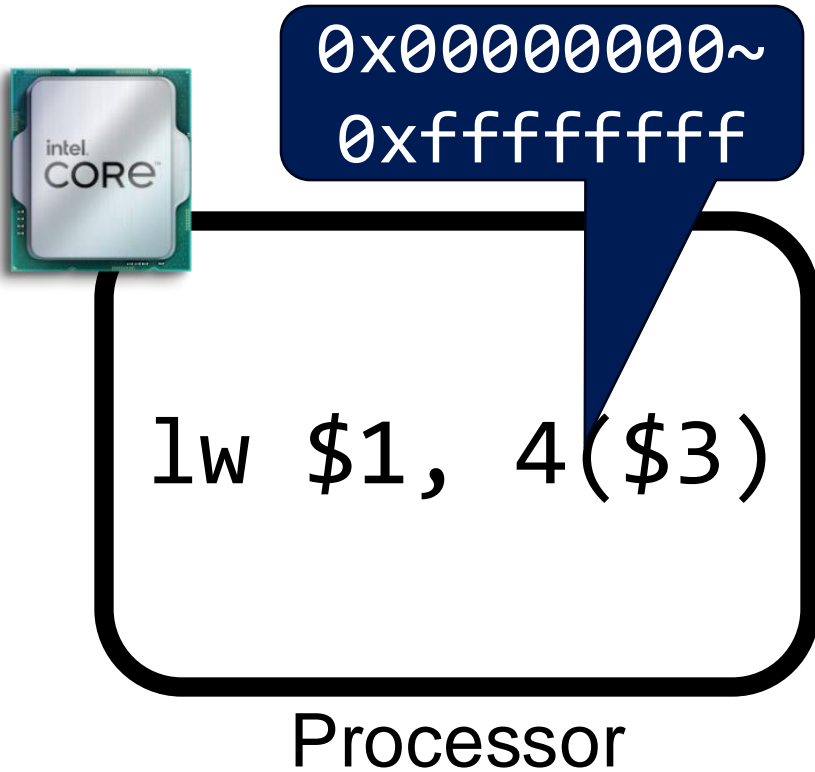


*How do we give an illusion
despite having small-sized physical memory?*

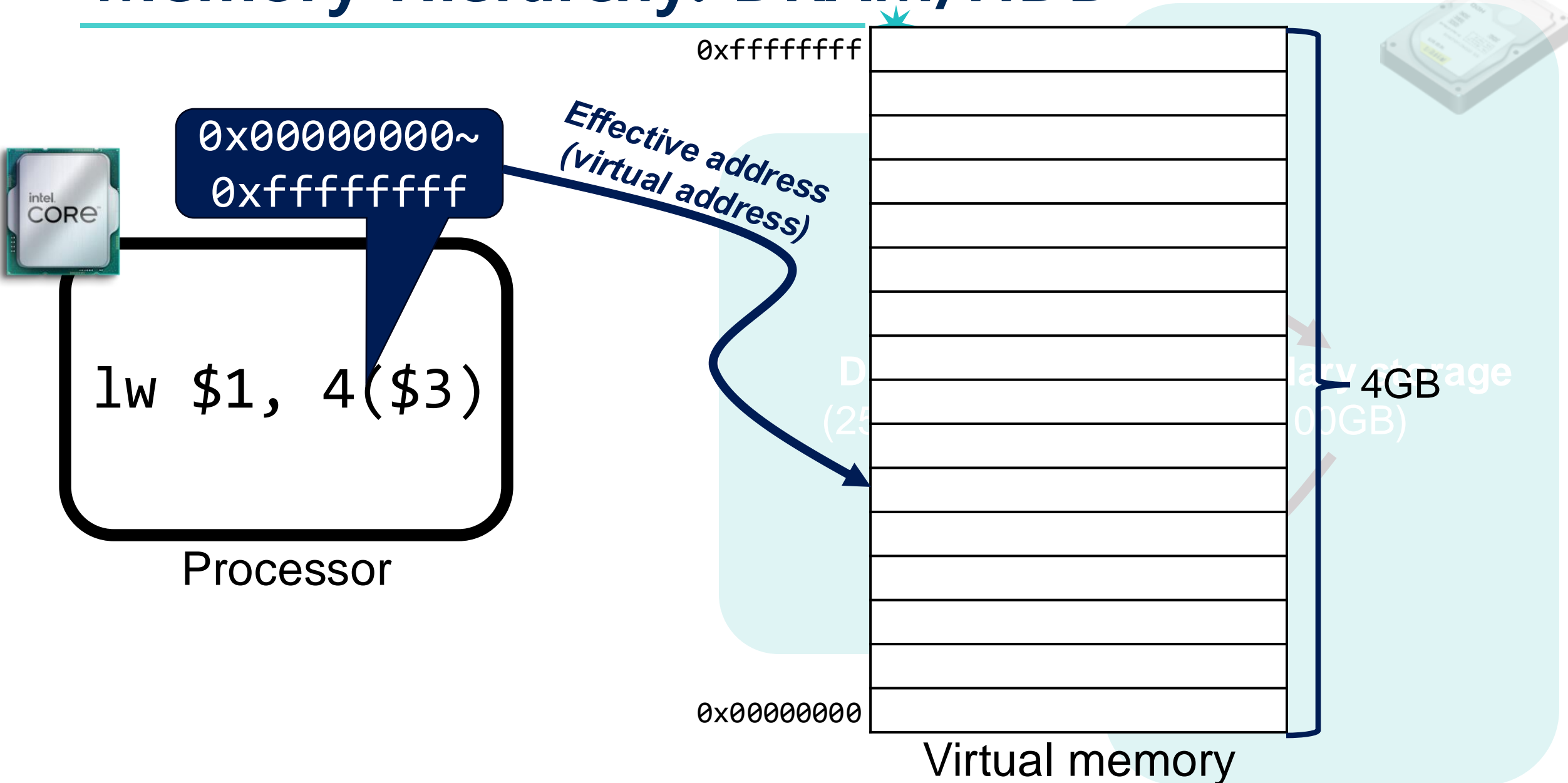
Memory hierarchy between RAM and HDD

Memory Hierarchy: DRAM/HDD

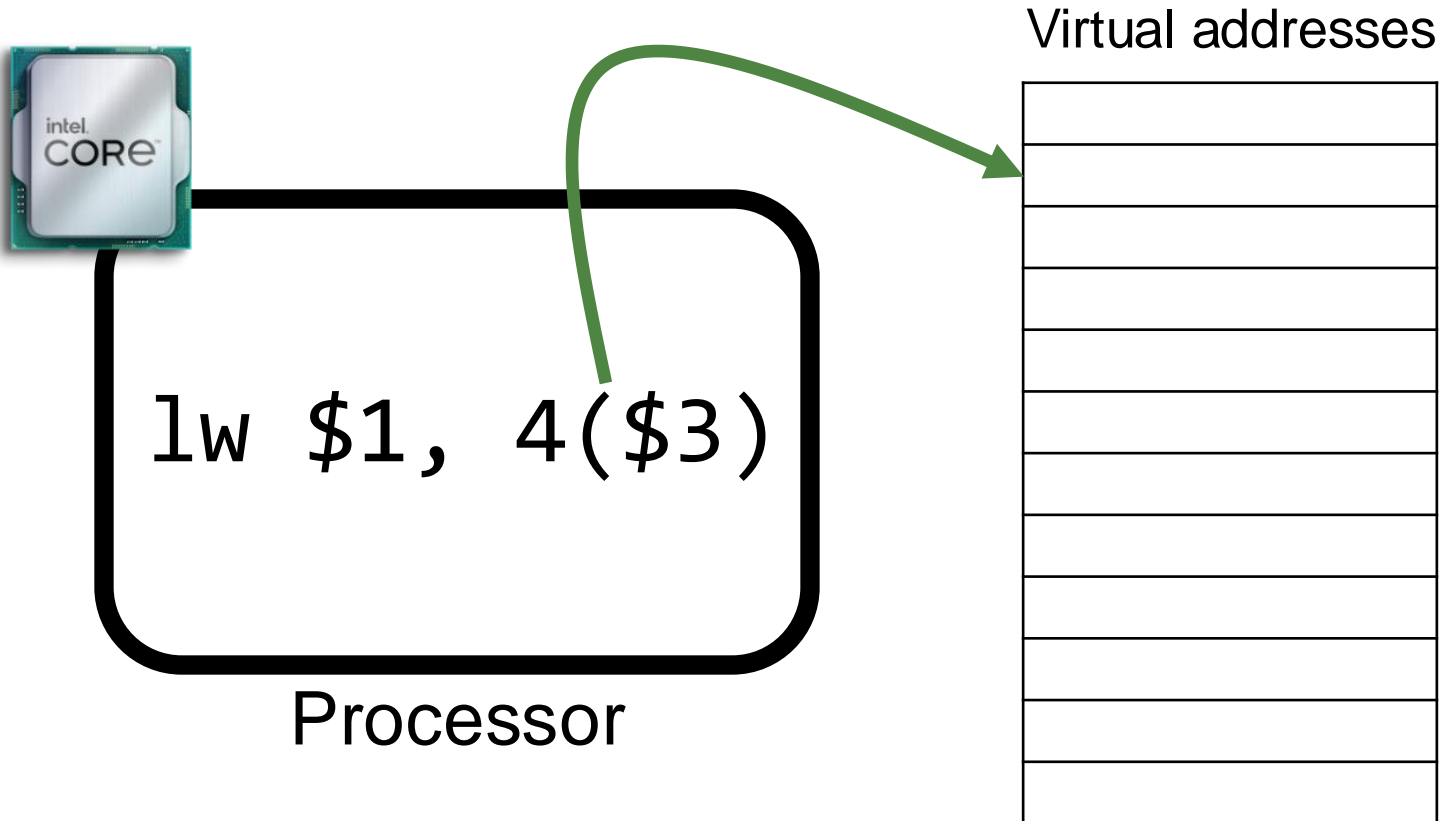
12



Memory Hierarchy: DRAM/HDD



Memory Hierarchy: DRAM/HDD



1w \$1, 4(\$3)

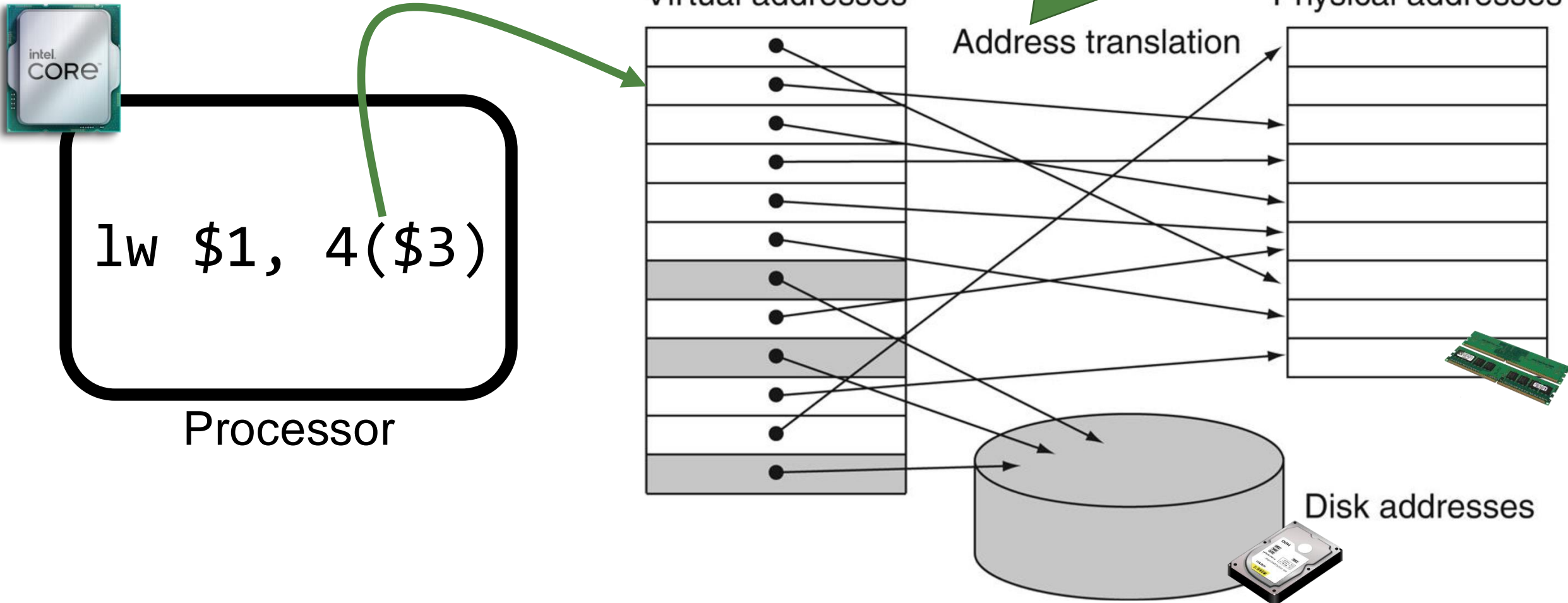
Processor

Virtual addresses

[illegible]

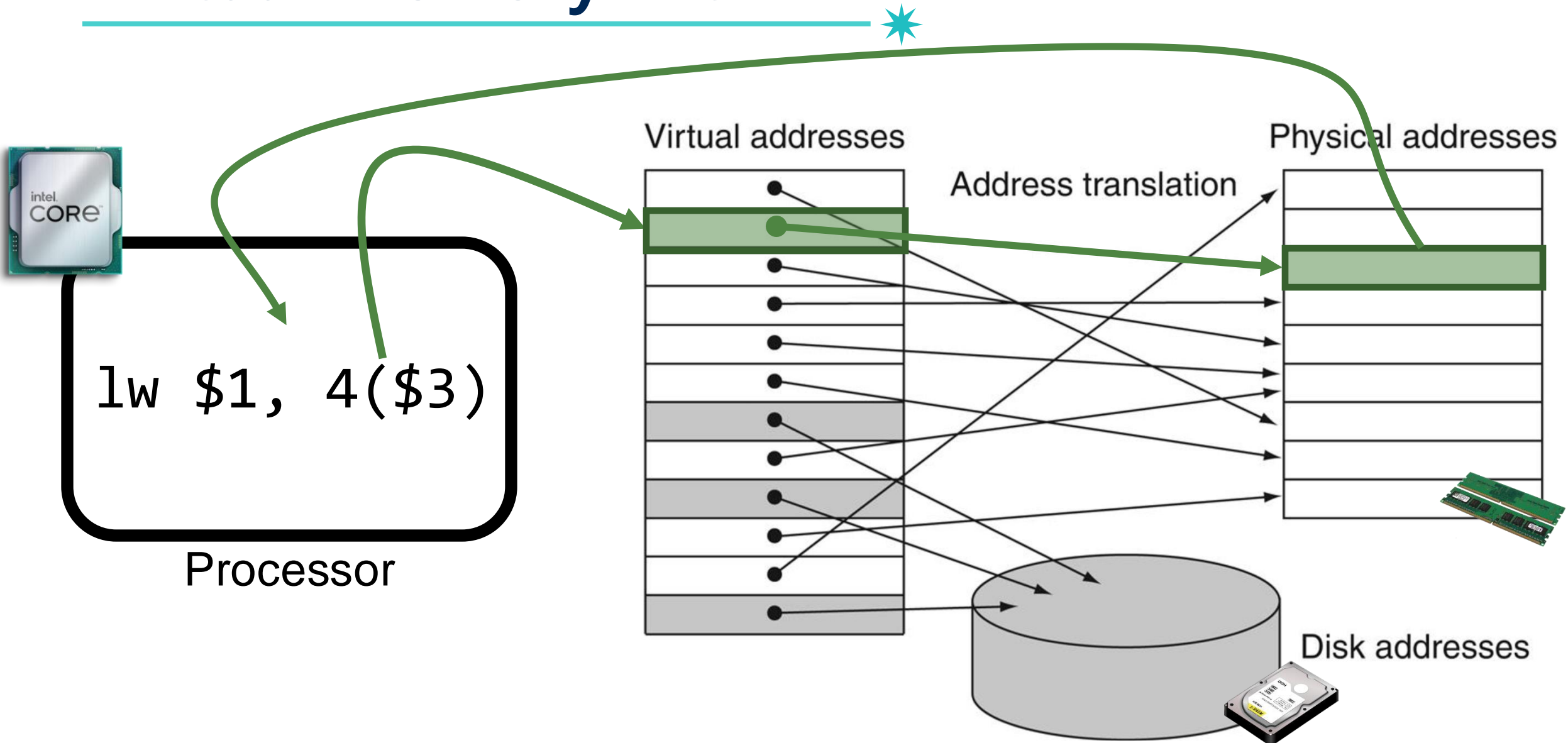
Memory Hi

Address translation table (OS-managed table)
Virtual address → Physical address



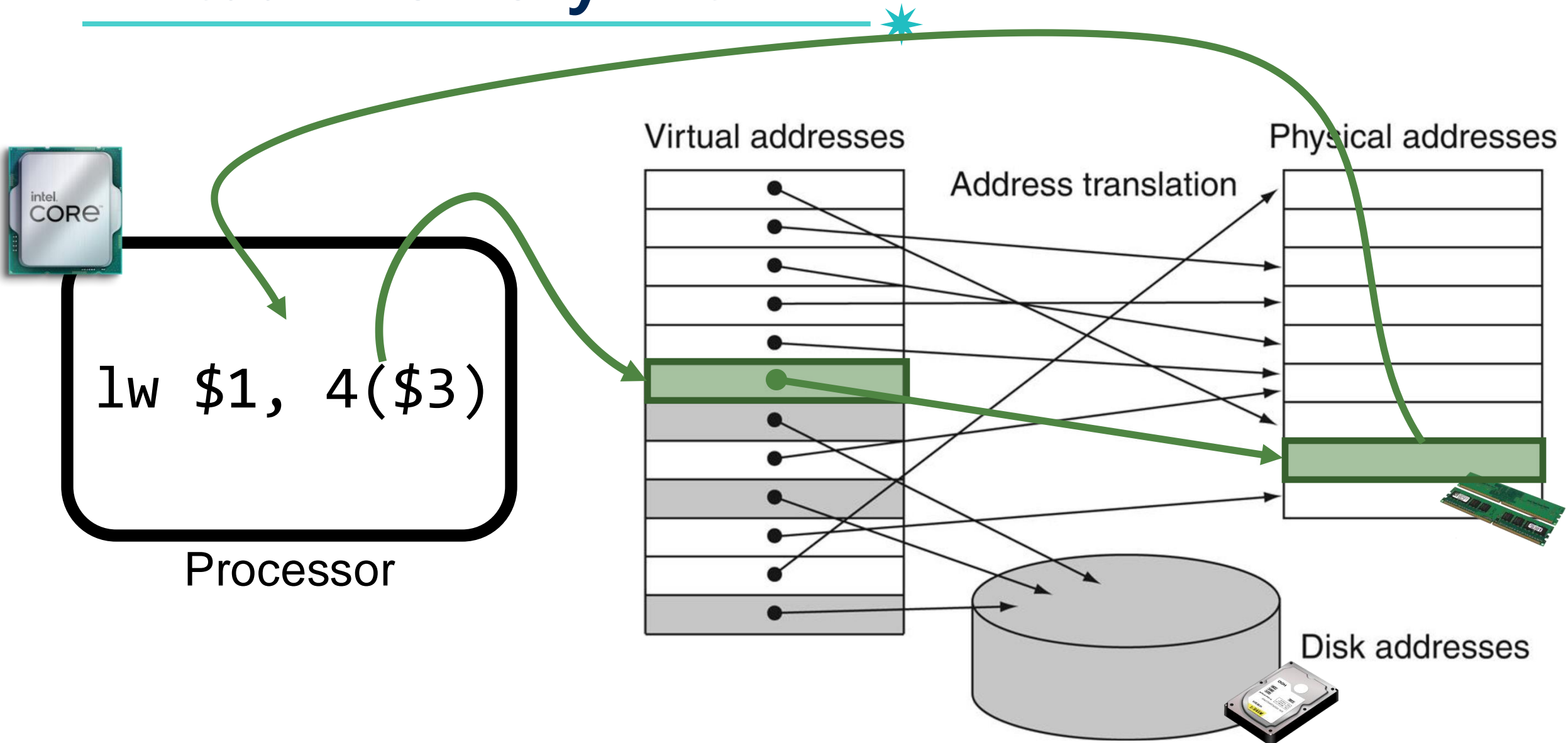
Virtual Memory Hit

16



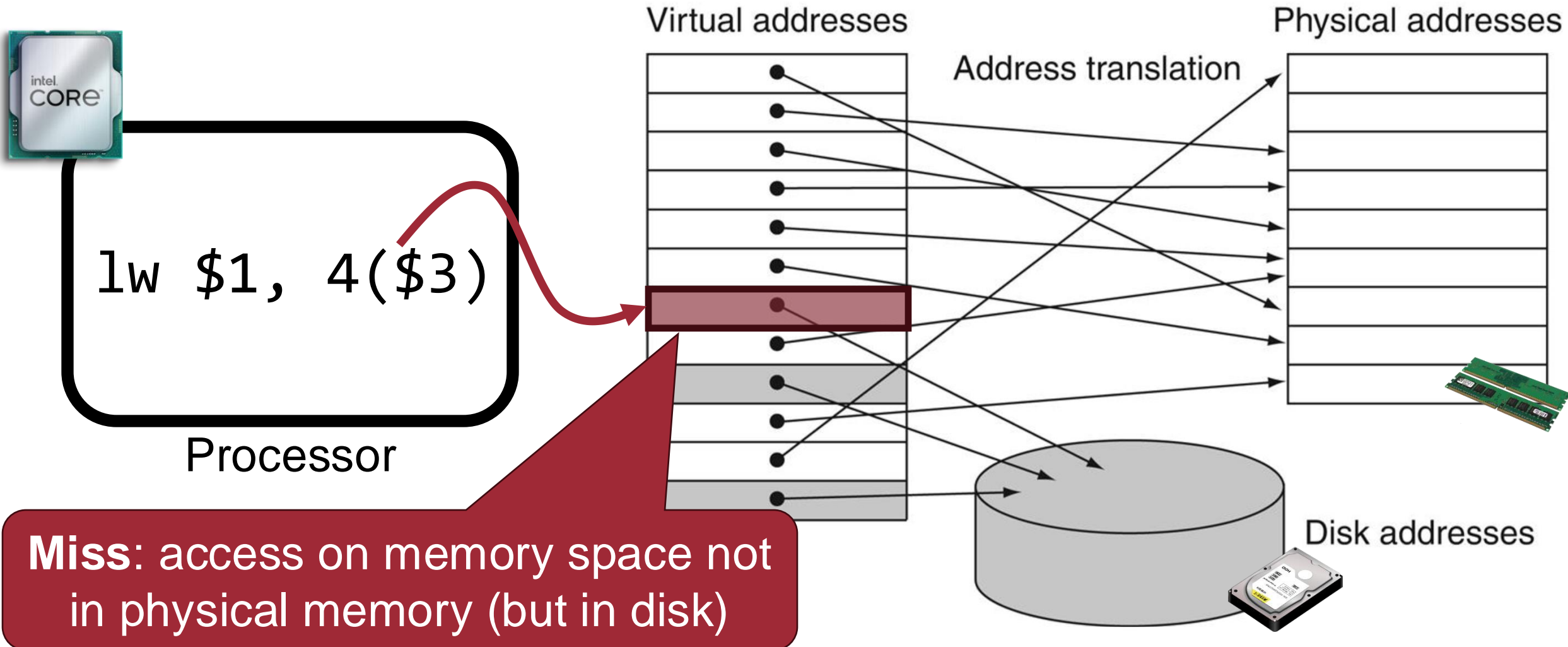
Virtual Memory Hit

17



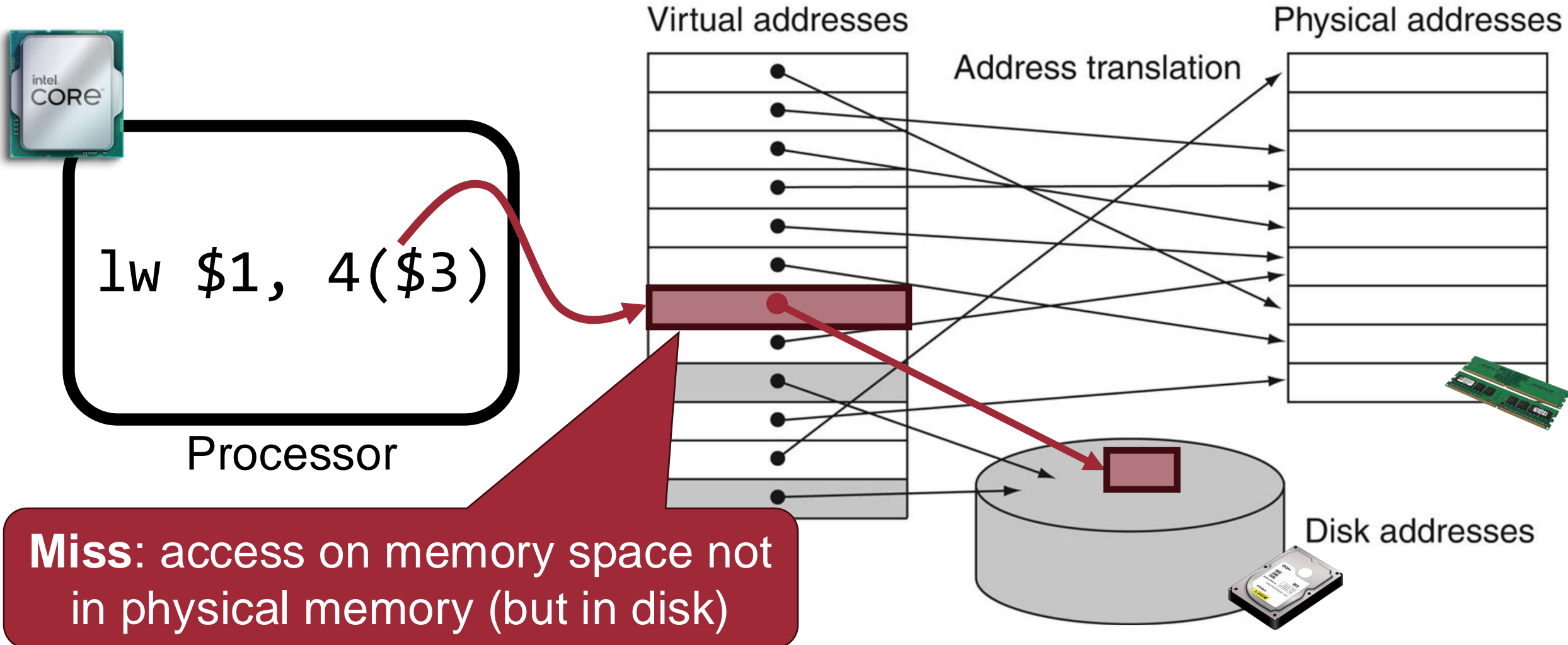
Virtual Memory Miss (a.k.a., Page Fault)

18



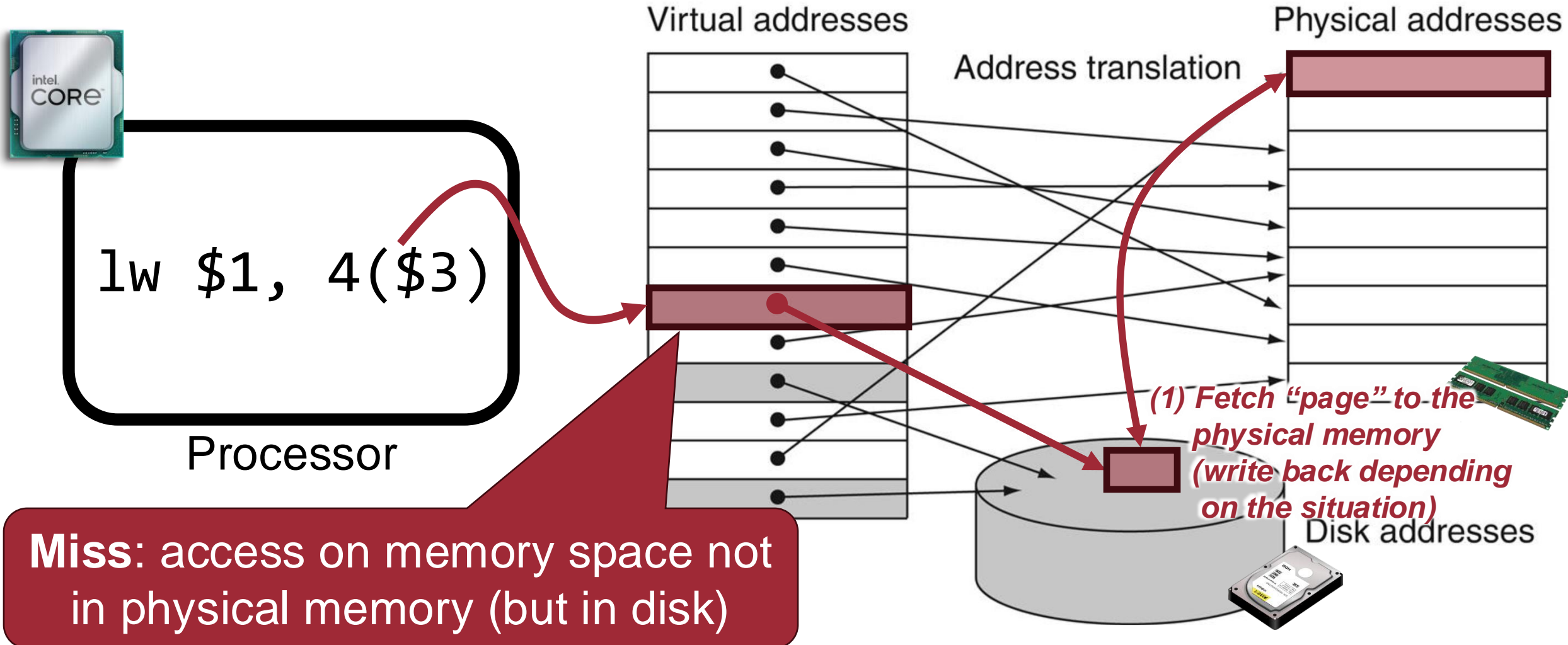
Virtual Memory Miss (a.k.a., Page Fault)

19



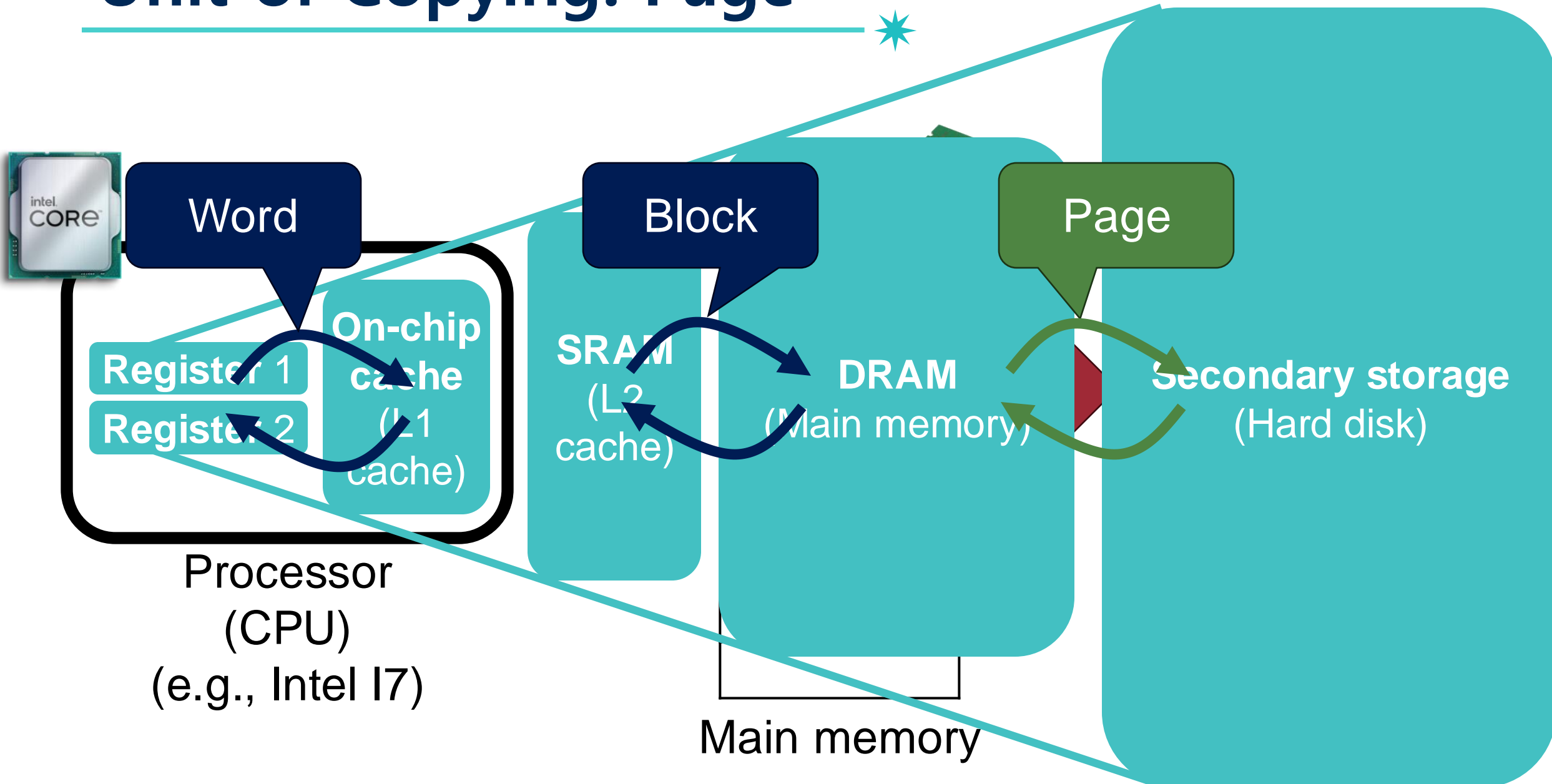
Virtual Memory Miss (a.k.a., Page Fault)

20

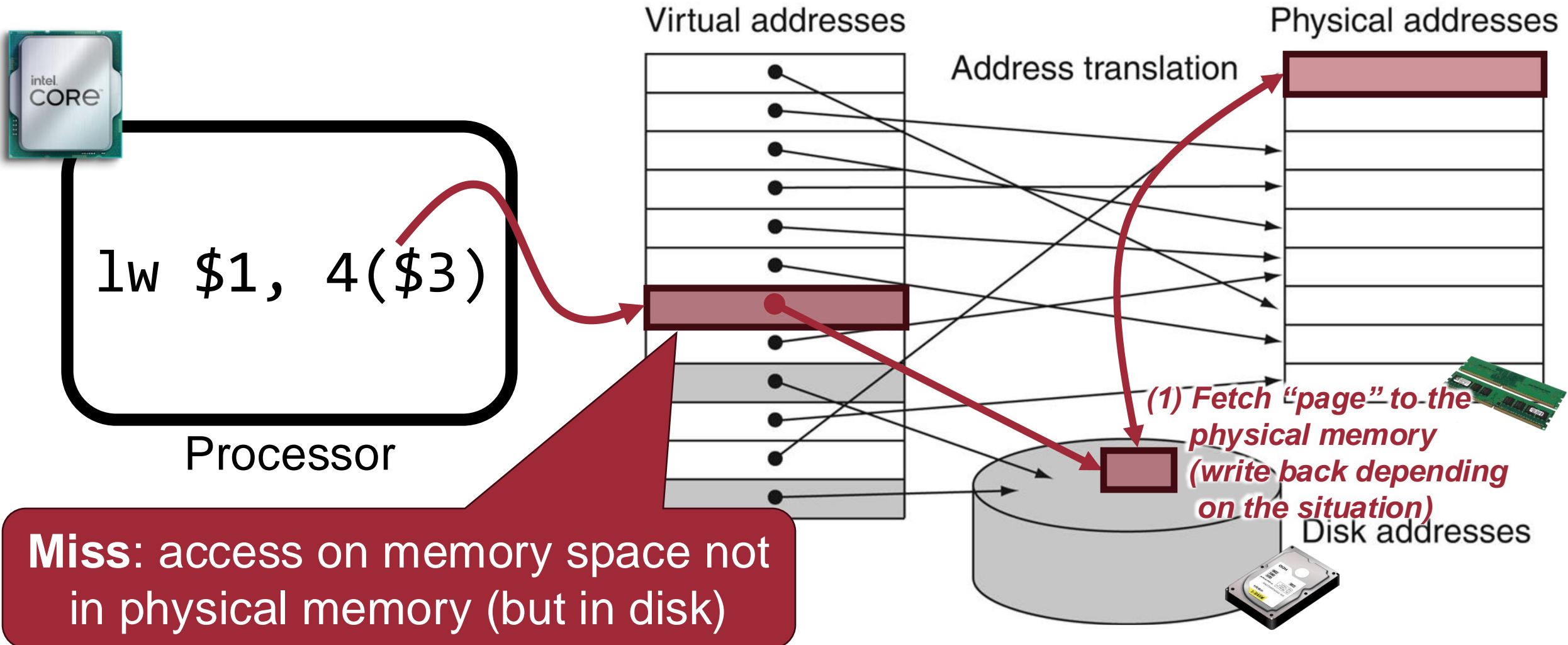


Unit of Copying: Page

21

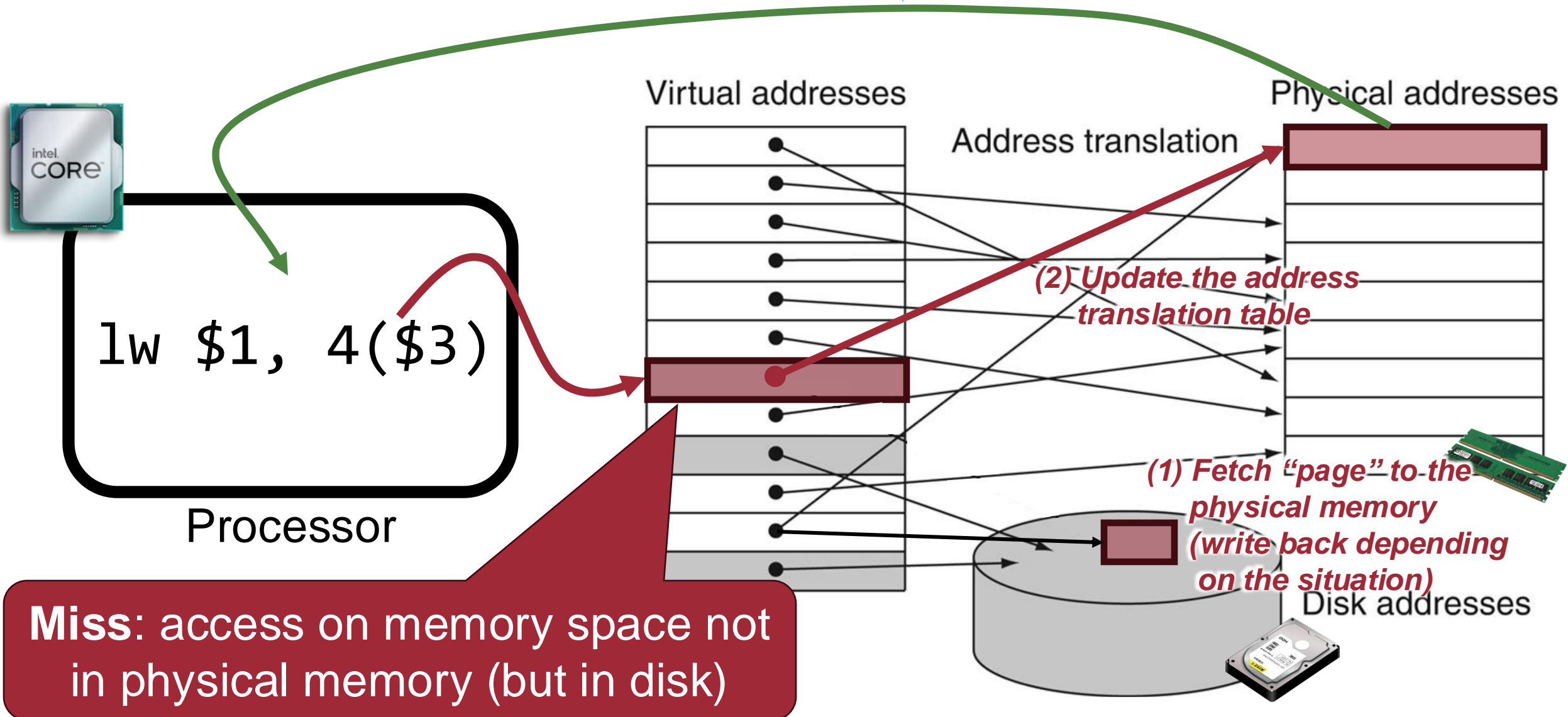


Virtual Memory Miss (a.k.a., Page Fault)



Virtual Memory Miss (a.k.a., Page Fault)

23



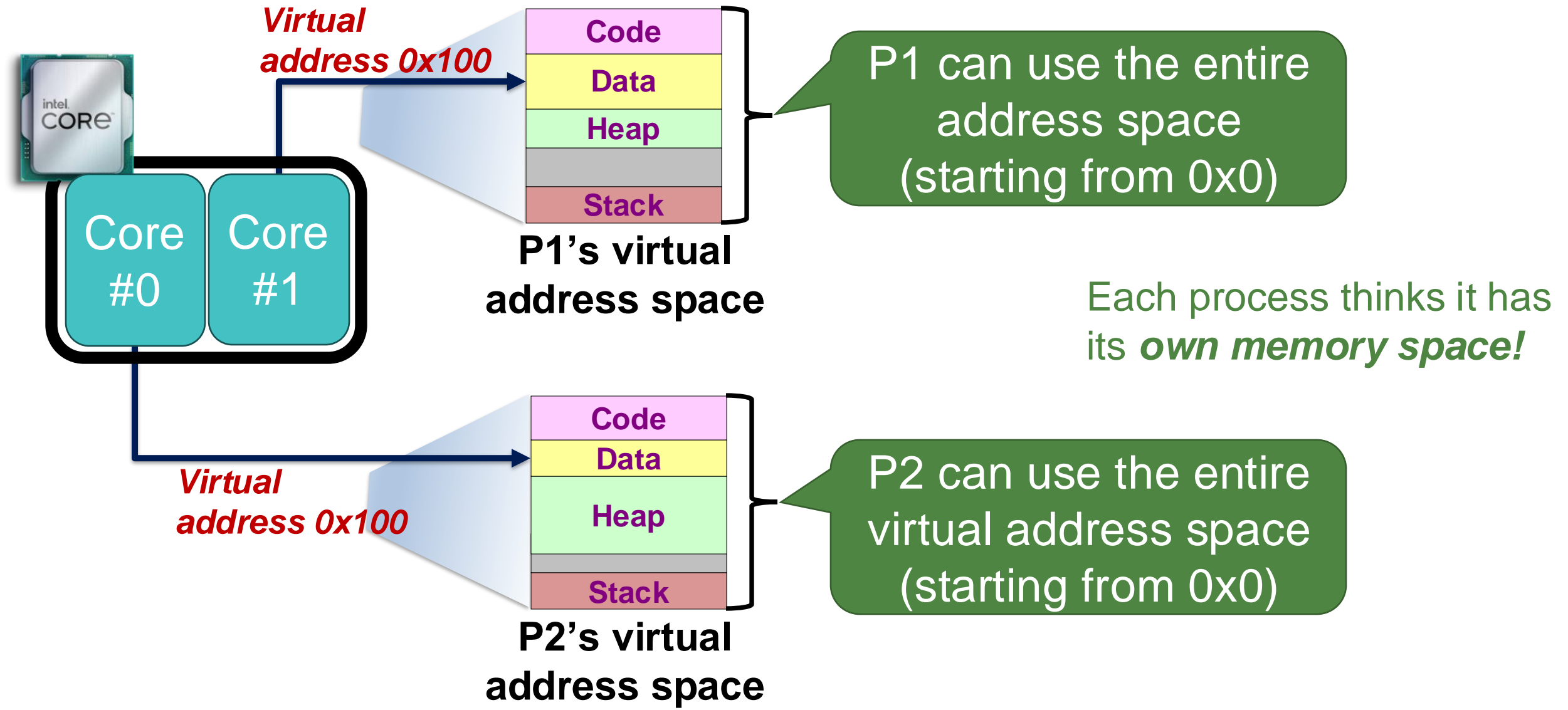
Virtual Memory



- Give programmers **an illusion of a large memory space** irrespective of actual capacity
- Uses main memory as a “cache” for the hard disk
- Other benefits?
 - **Program simplification**: give each program the illusion that it has its *own private memory* (e.g., 0x00000000~0xffffffff)

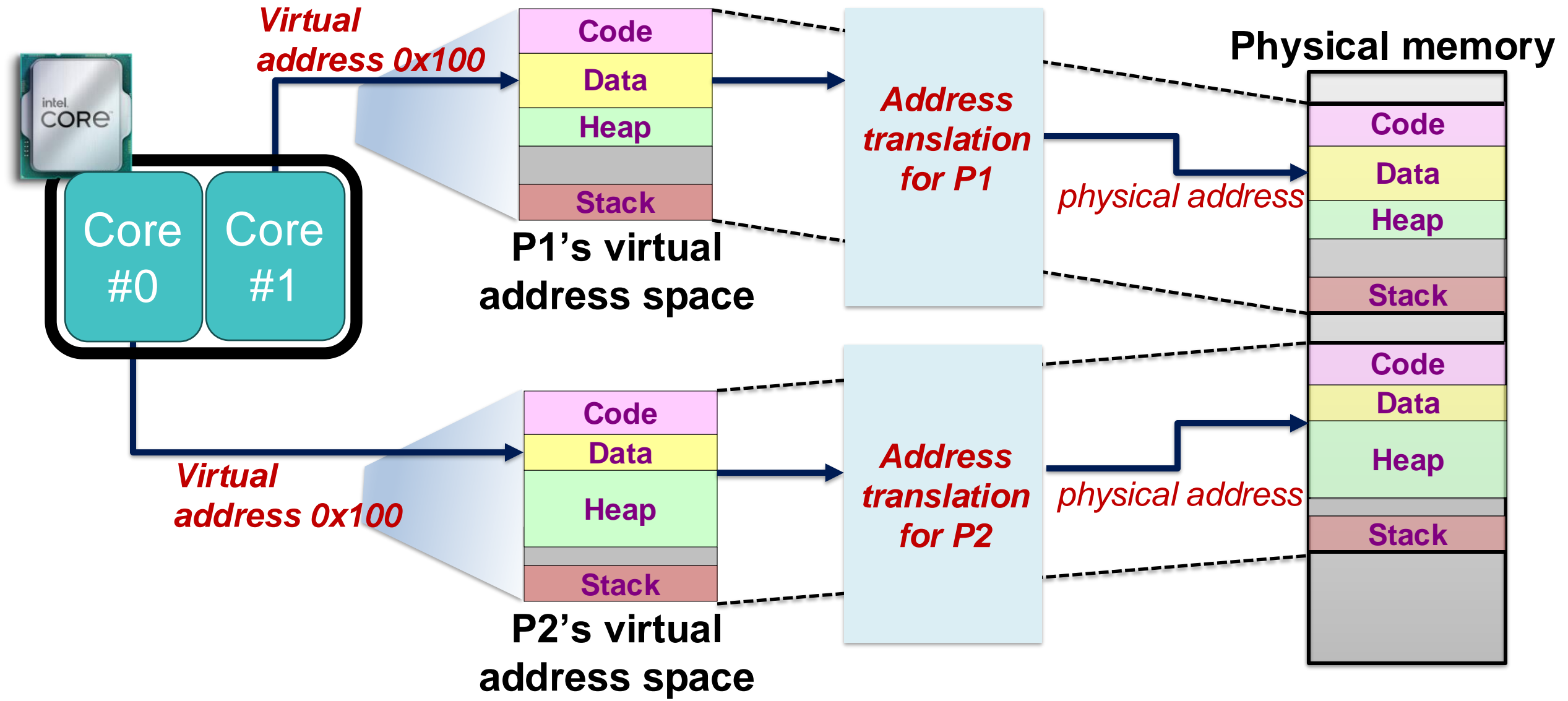
Program Simplification

25

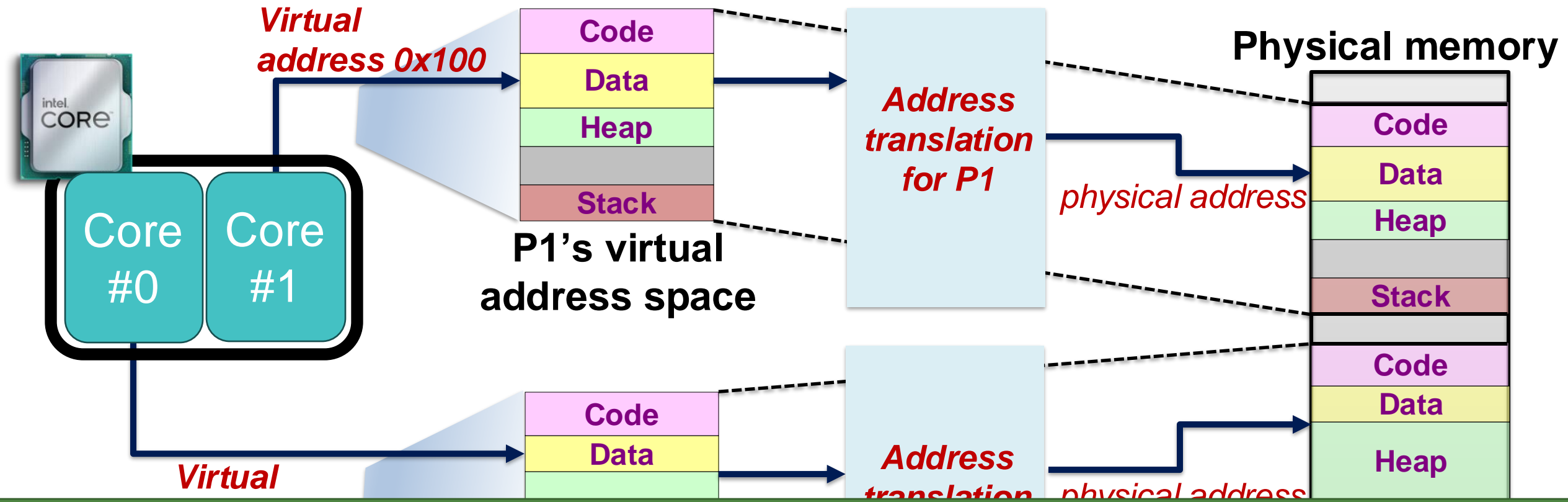


Program Simplification

26



Program Simplification



Programming and storage management ease

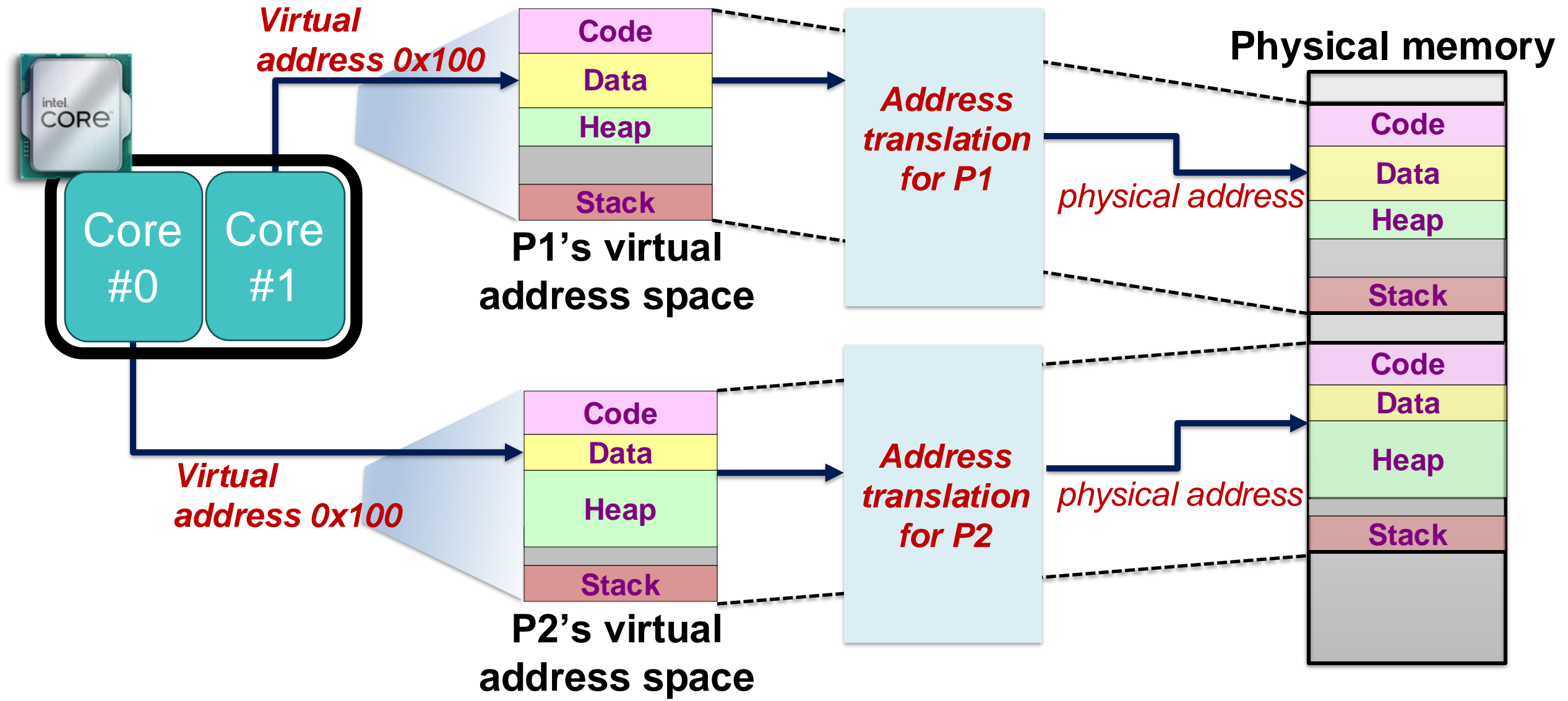
Virtual Memory



- Give programmers **an illusion of a large memory space** irrespective of actual capacity
- Uses main memory as a “cache” for the hard disk
- Other benefits?
 - **Program simplification**: give each program the illusion that it has its *own private memory* (e.g., 0x00000000~0xffffffff)
 - **Security**: memory protection between processes

Memory Protection

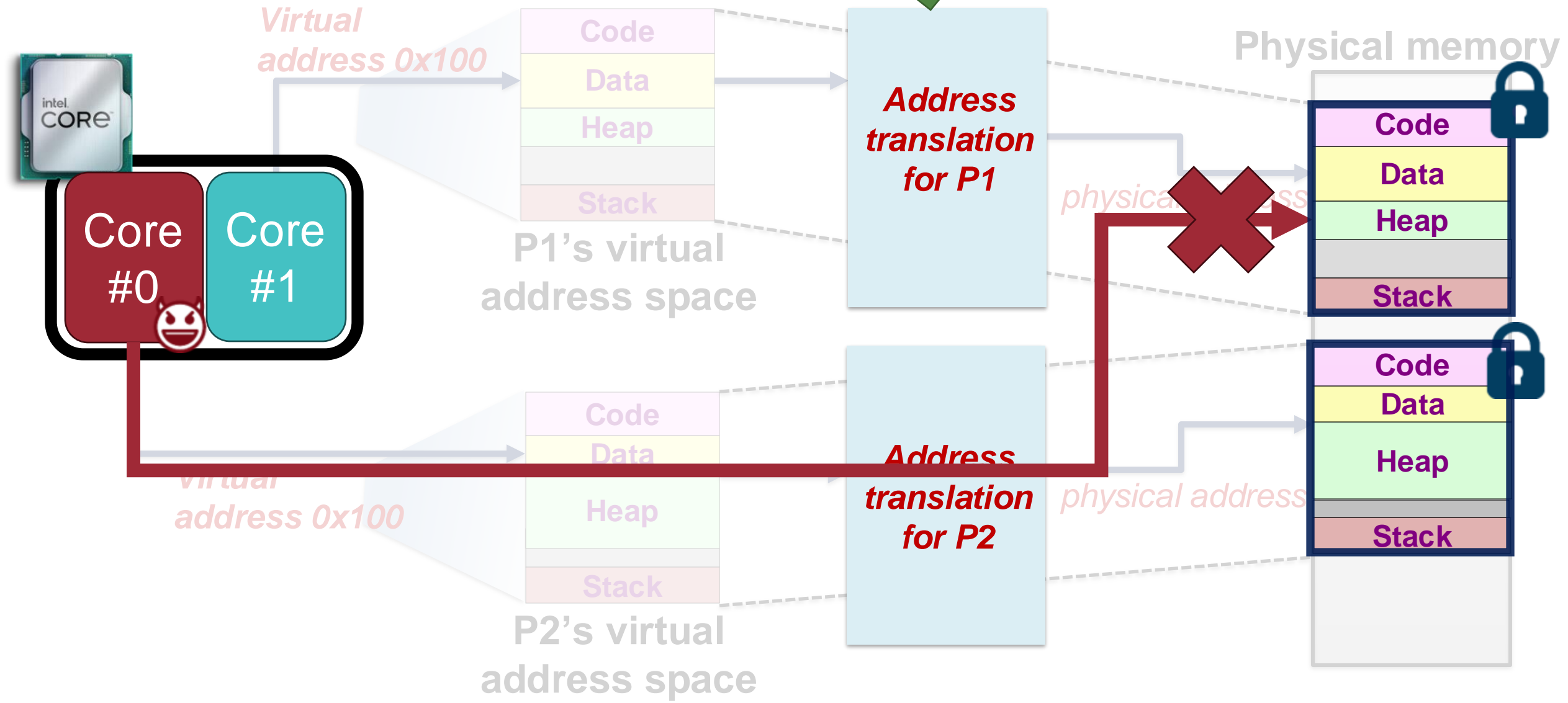
29



Memory Protection

30

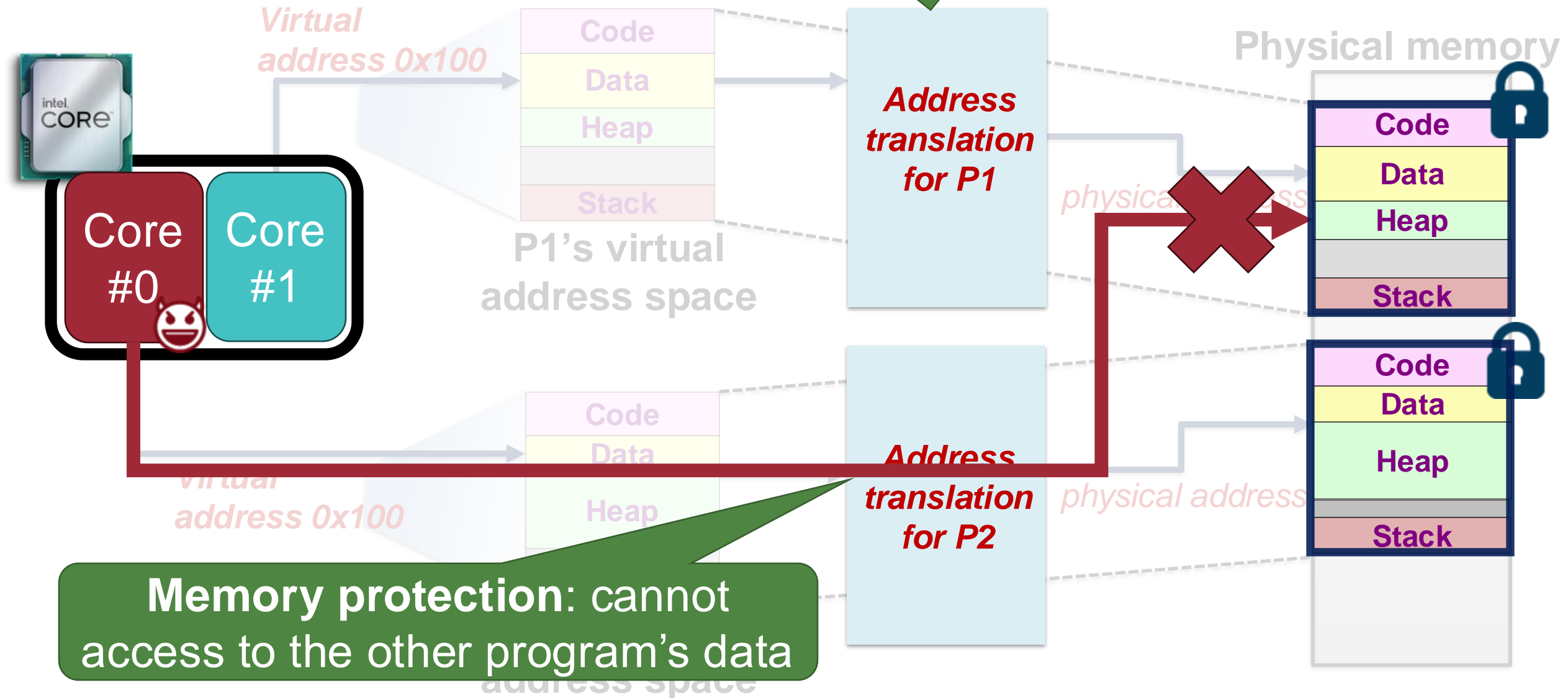
OS-managed table



Memory Protection

31

OS-managed table



Virtual Memory Summary

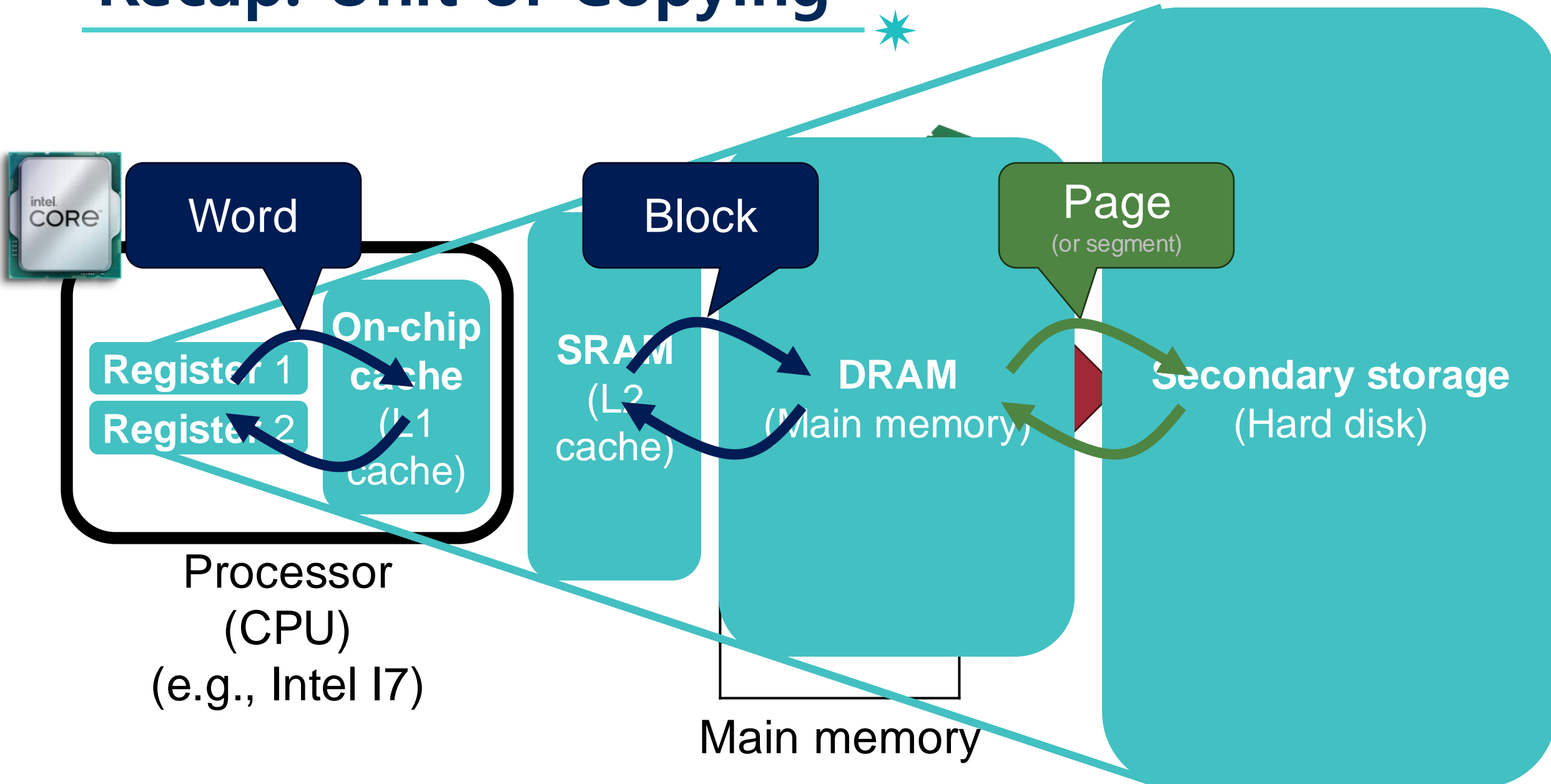


- Idea: Give programmers **an illusion of a large memory space** irrespective of actual capacity
 - Programmers assume they have “infinite” amount of physical memory
 - Programmers do not need to worry about how to manage it (simplifies memory management for multi-processing system)
- CPU and OS cooperatively manage physical memory space to provide the illusion on behalf of users
 - Illusion is maintained for each independent process
 - Let OS to share memory, protect programs from each other

Unit of Copying: Page

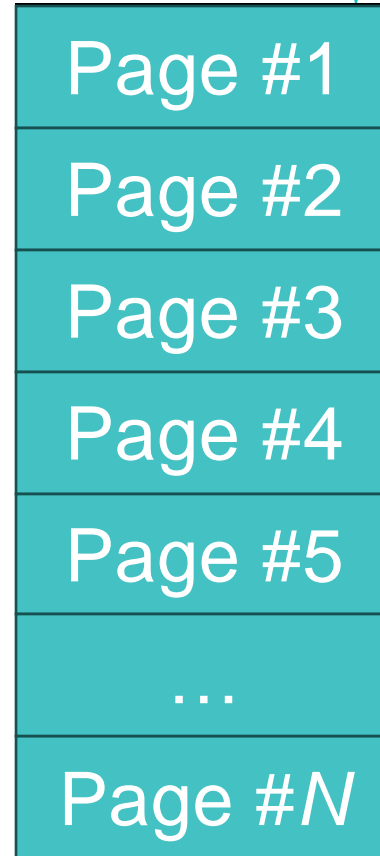
Recap: Unit of Copying

34



Virtual Memory Block Type #1: Page

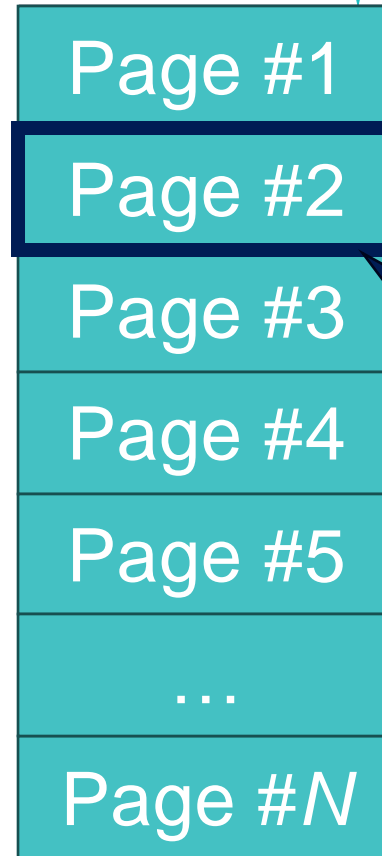
- A block of virtual memory
- Unit of copying
 - between DRAM and Disk
- Fixed length
 - e.g., 4KB
 - Generally, 4KB ~ 4MB



Process A
(Virtual memory)

Virtual Memory Block Type #1: Page

- A block of virtual memory
- Unit of copying
 - between DRAM and Disk
- Fixed length
 - e.g., 4KB
 - Generally, 4KB ~ 4MB

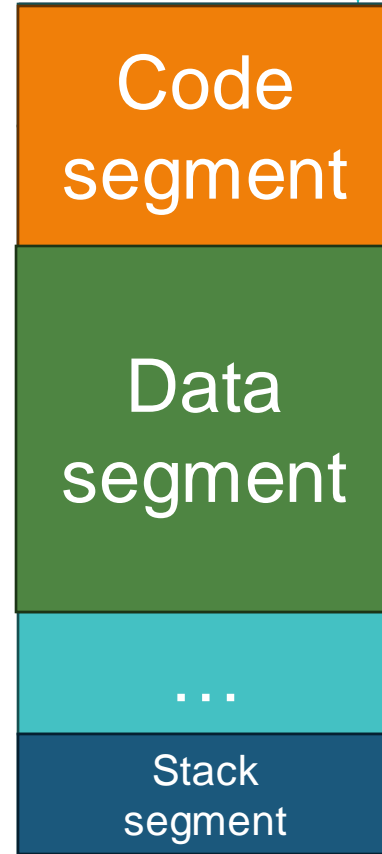


Fixed-length
virtual memory block

Process A
(Virtual memory)

Virtual Memory Block Type #2: Segment

Segment length is variable!



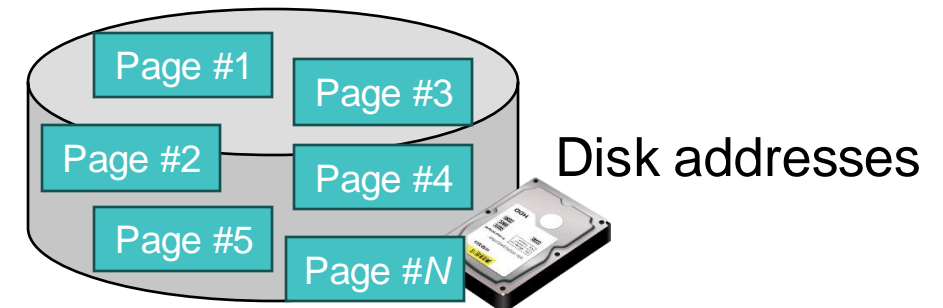
Process A
(Virtual memory)

Page

- A block of virtual memory
- Unit of copying
 - between DRAM and Disk
- Fixed length
 - e.g., 4KB
 - Generally, 4KB ~ 4MB



Process A
(Virtual memory)



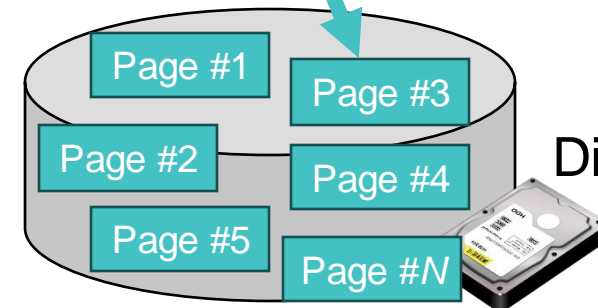
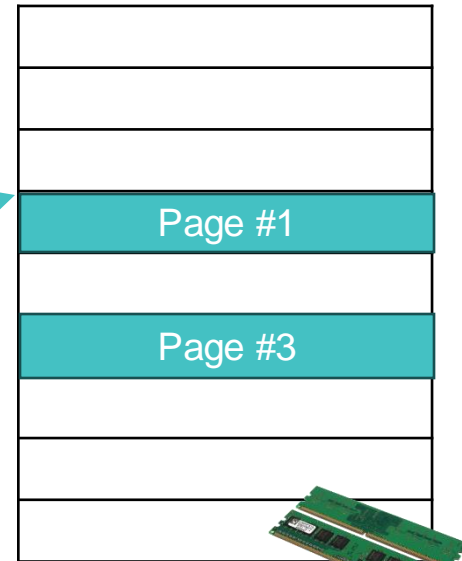
Memory Paging (a.k.a., Swapping)

Memory management scheme by which a computer **stores and retrieves data** from secondary storage for use in main memory

Page: unit of copying!

*Paging
(Swapping)*

Physical addresses

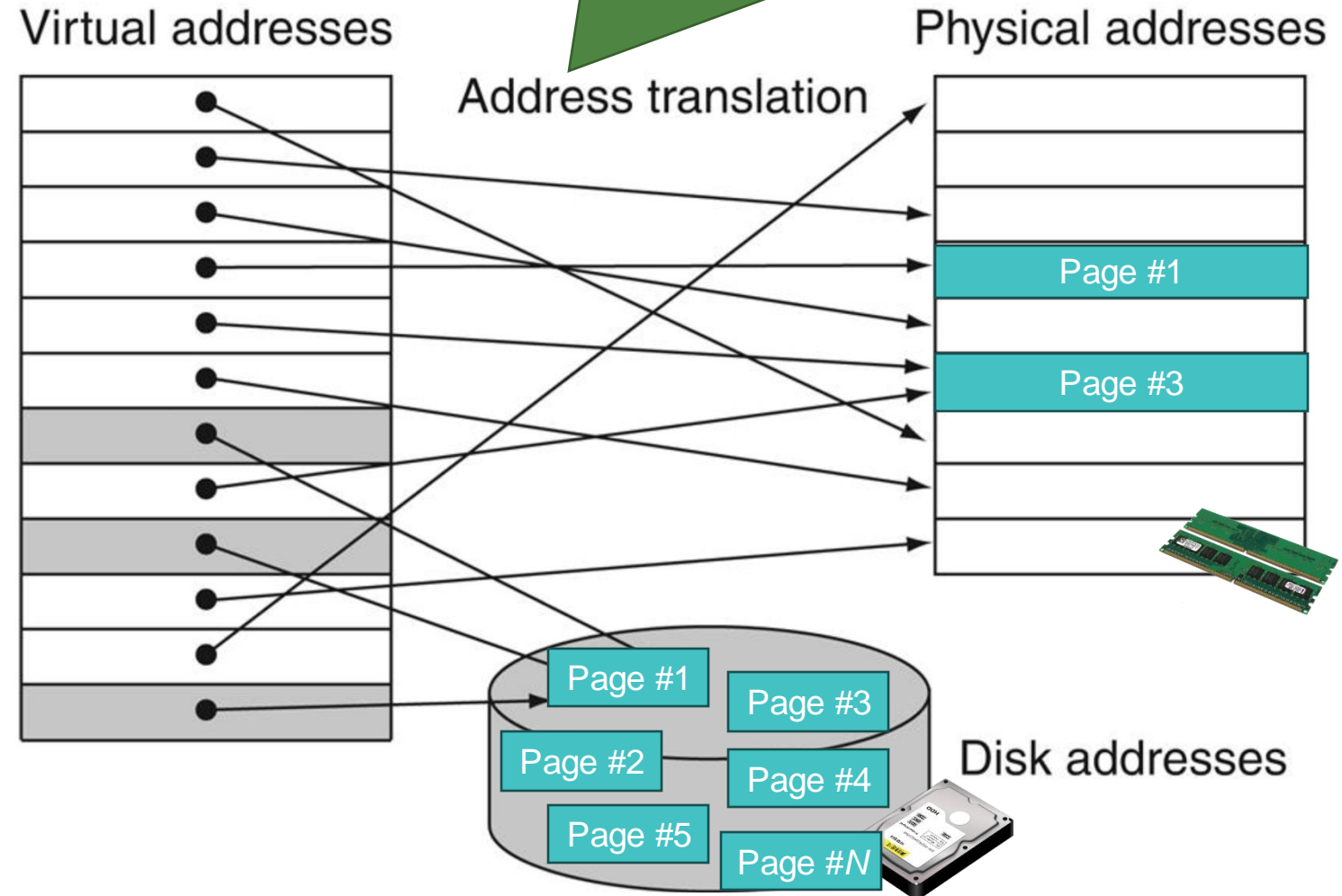


Disk addresses

Address Translation

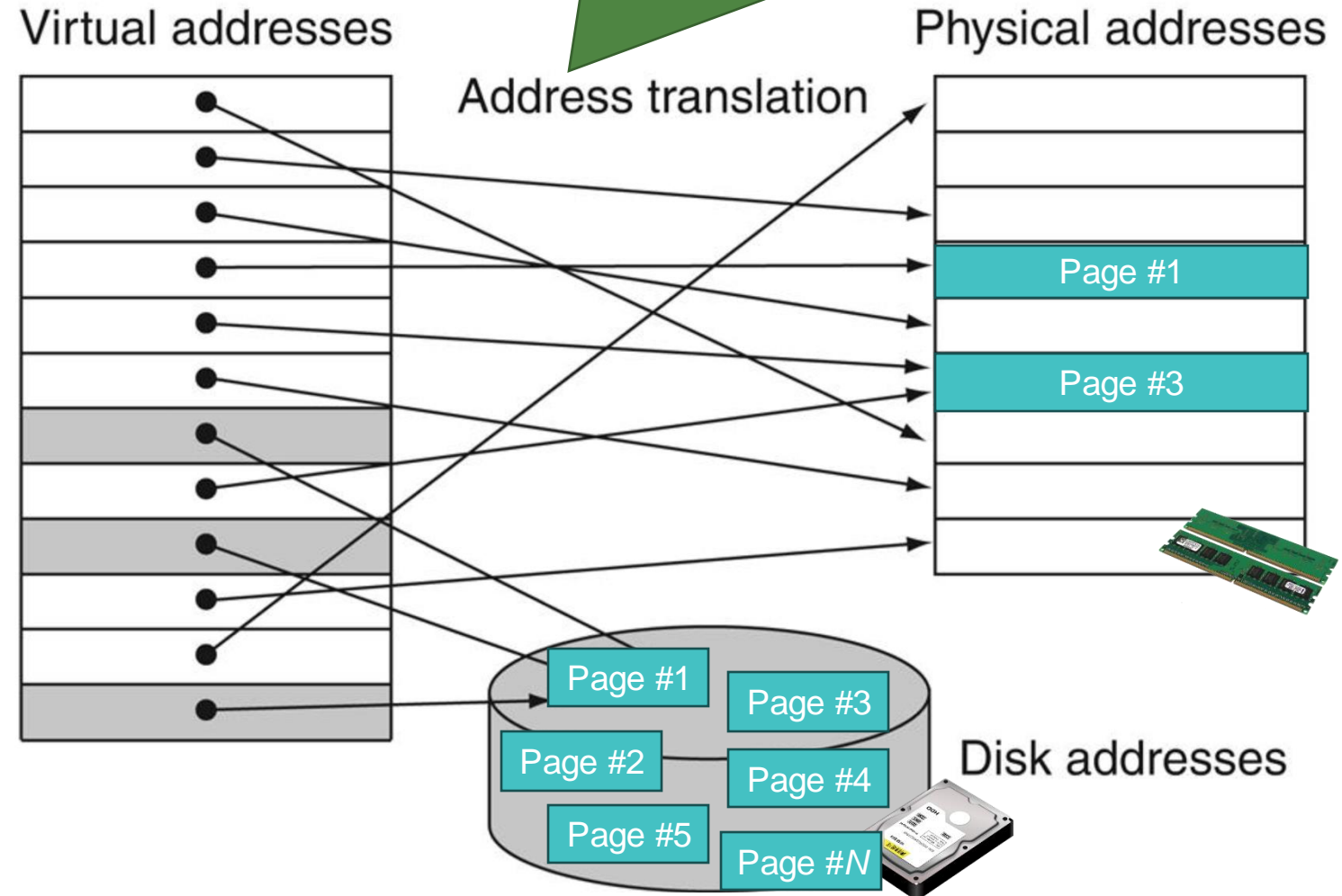
Memory Hi

Address translation table (OS-managed table)
Virtual address → Physical address



Memory Hi

“Page Table”



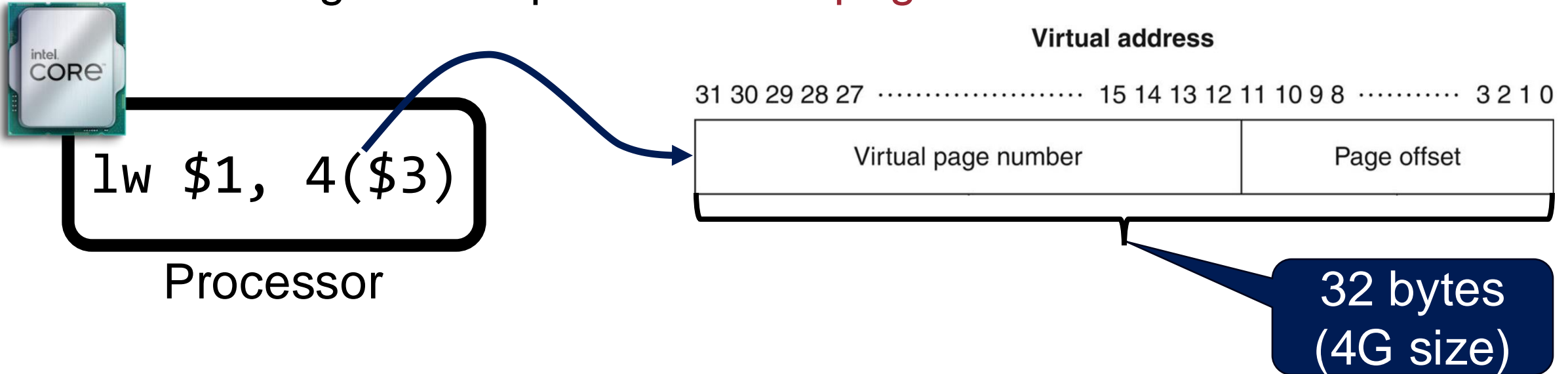
Page Table: Address Translation



CPU converts virtual addresses into physical addresses via an OS-managed lookup table called **page table**

Page Table: Address Translation

CPU converts virtual addresses into physical addresses via an OS-managed lookup table called **page table**



Recap: the range of memory addresses a 32-bit machine (e.g., MIPS) can access?

Representable range = $0 \sim 2^{32}-1$

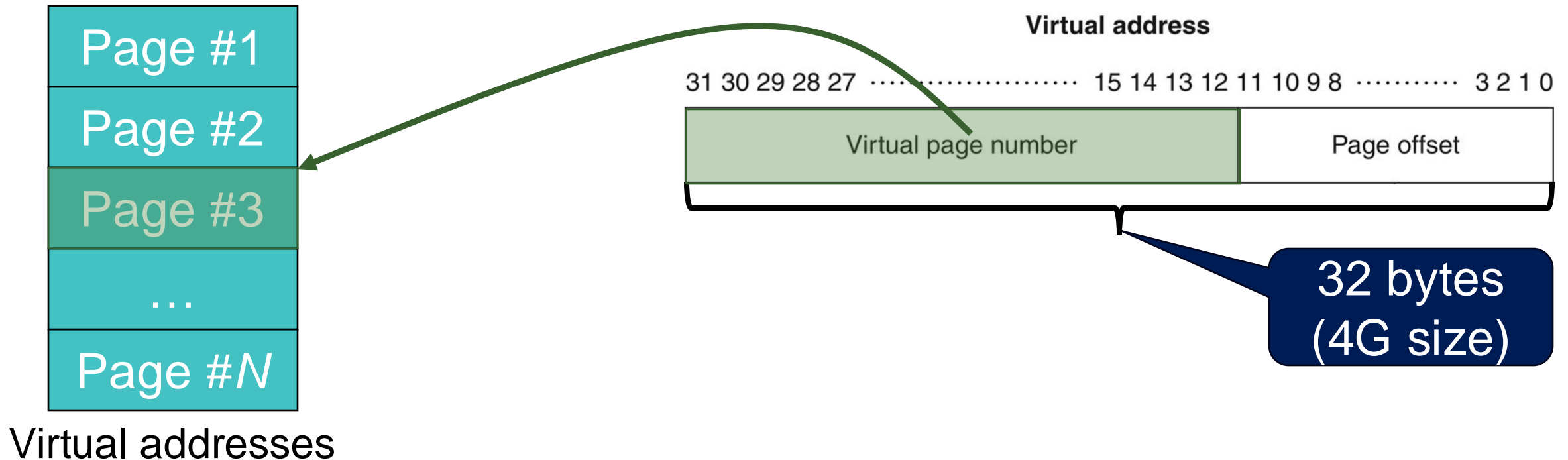
0xffffffff
⋮
0x00000001
0x00000000

Q. What is the maximum memory capacity a 32-bit machine can handle?

2^{32} bytes = 4GB

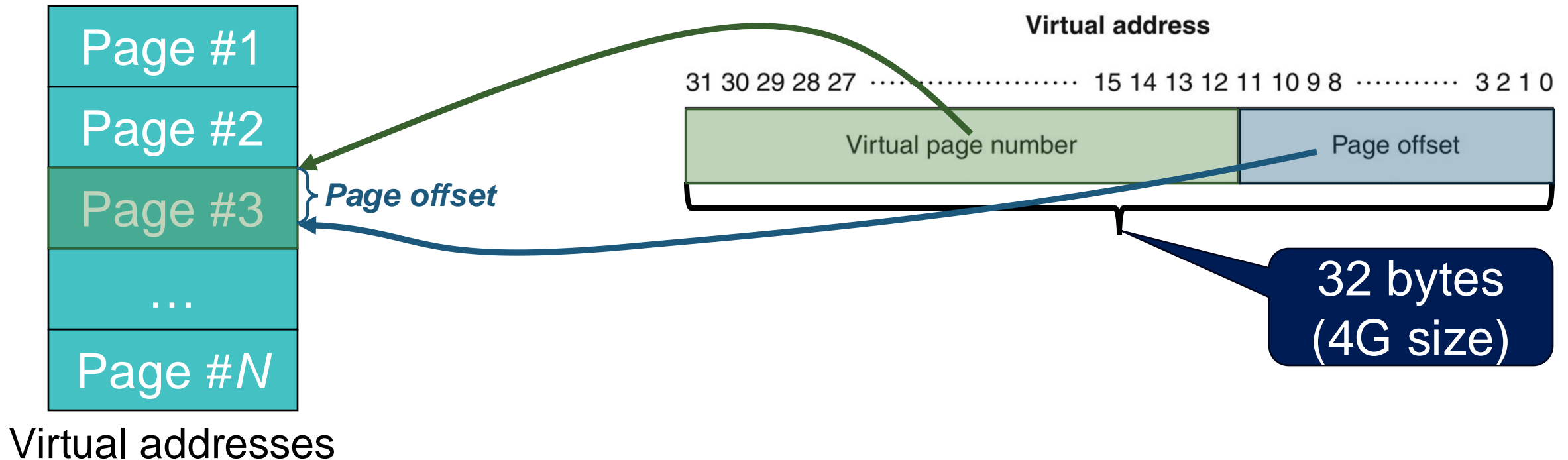
Programmer's Point of View

CPU converts virtual addresses into physical addresses via an OS-managed lookup table called **page table**



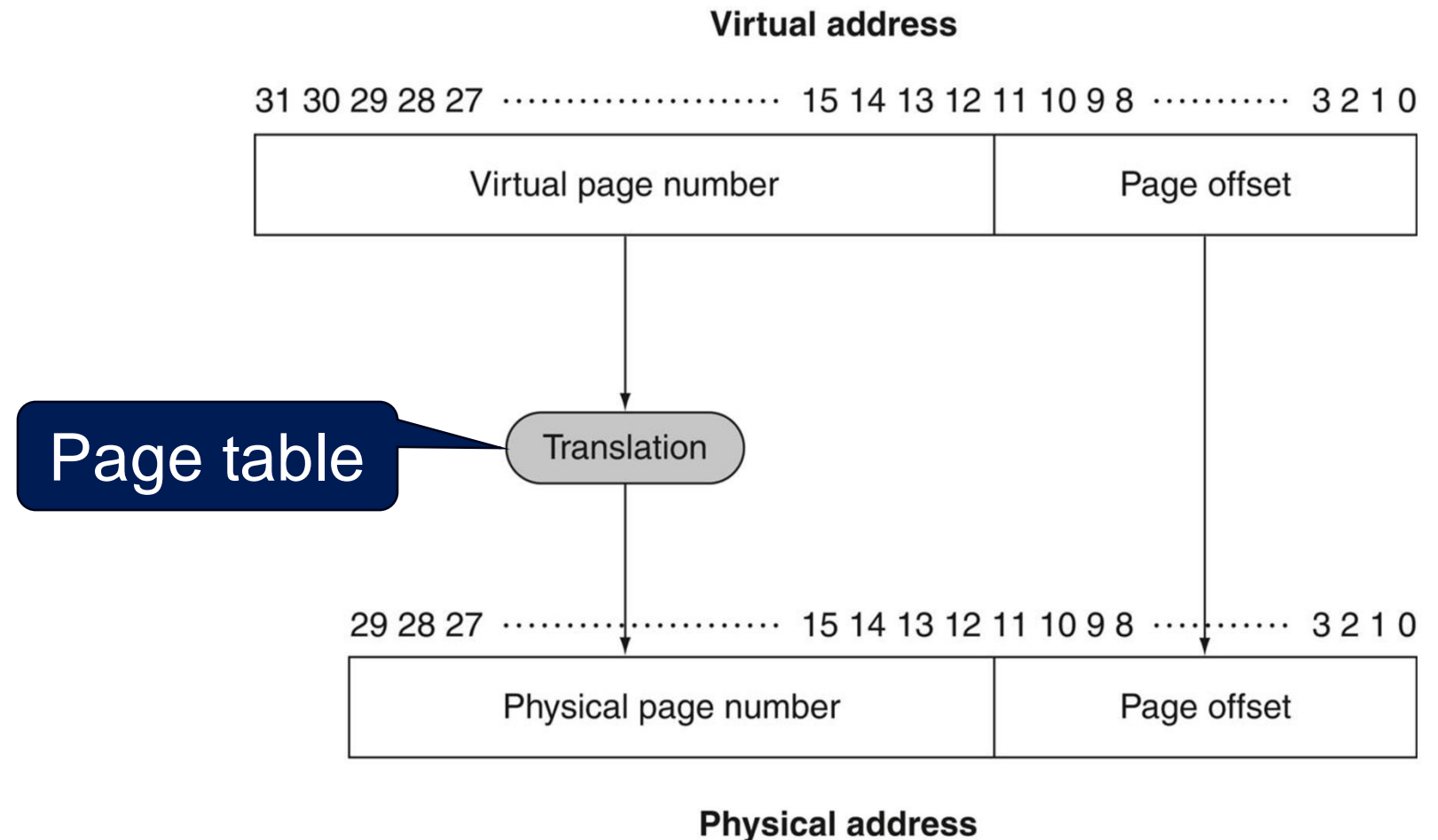
Programmer's Point of View

CPU converts virtual addresses into physical addresses via an OS-managed lookup table called **page table**



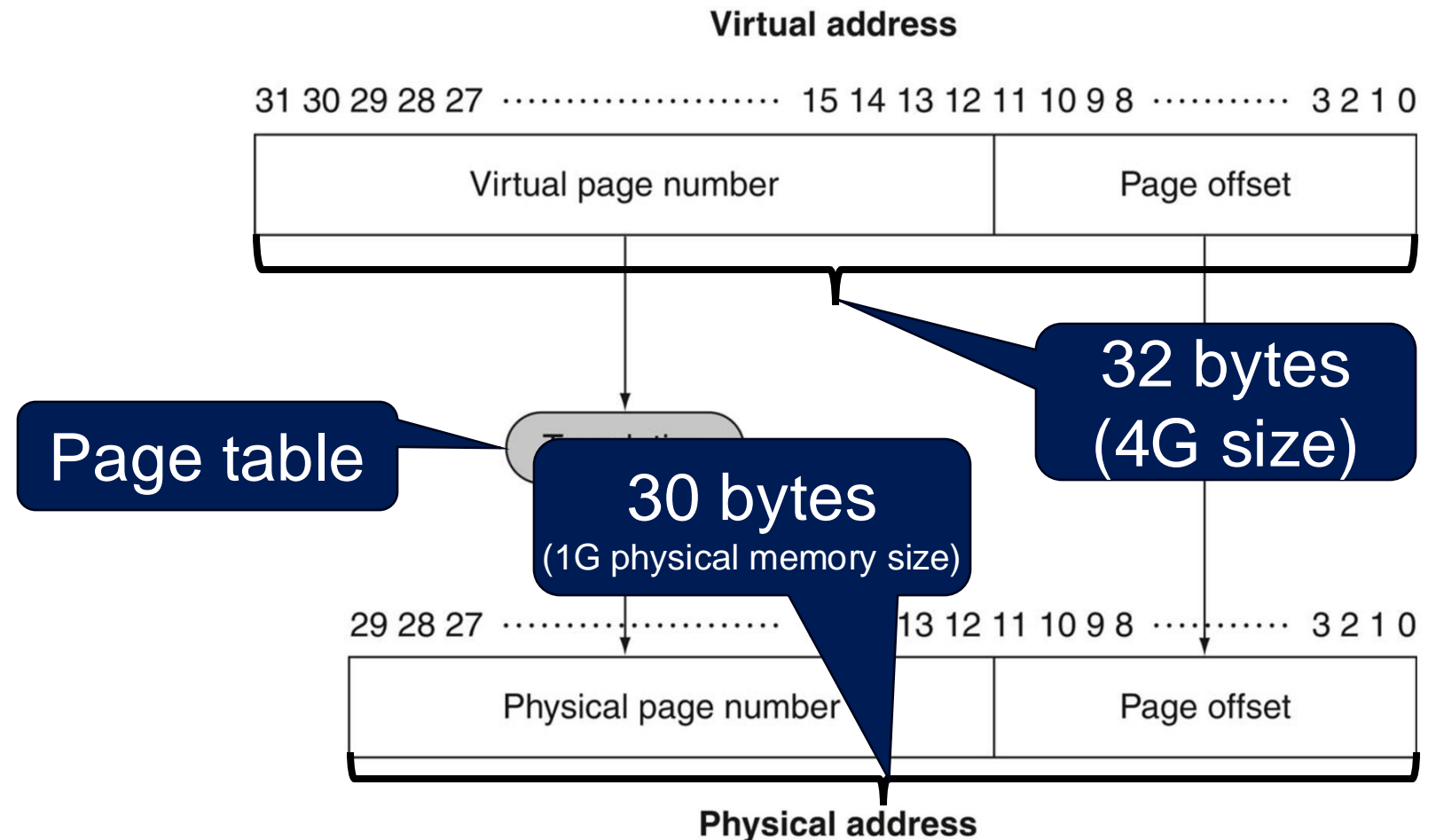
Page Table: Address Translation

CPU converts virtual addresses into physical addresses via an OS-managed lookup table called **page table**



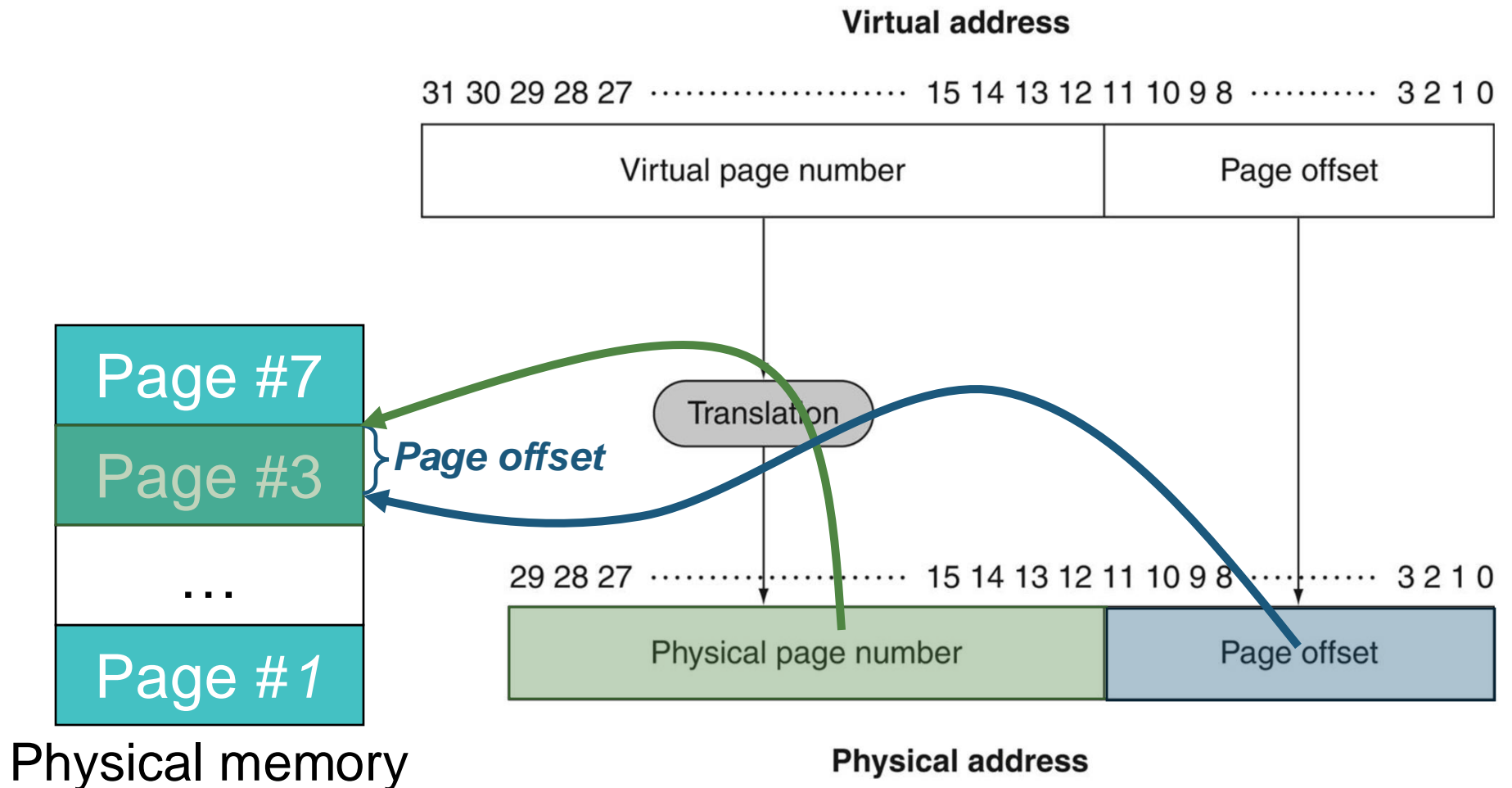
Page Table: Address Translation

CPU converts virtual addresses into physical addresses via an OS-managed lookup table called **page table**



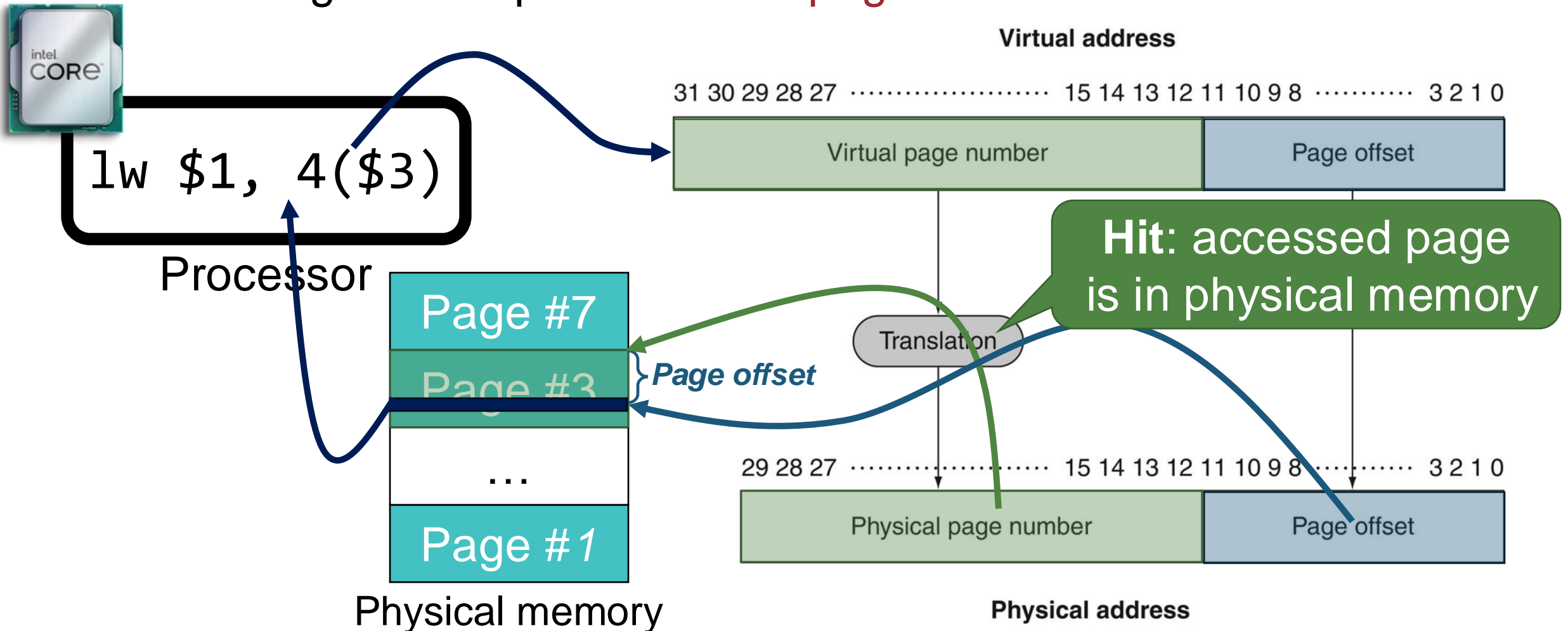
Page Table: Address Translation

CPU converts virtual addresses into physical addresses via an OS-managed lookup table called **page table**



Address Translation Flow (Hit Case)

CPU converts virtual addresses into physical addresses via an OS-managed lookup table called **page table**

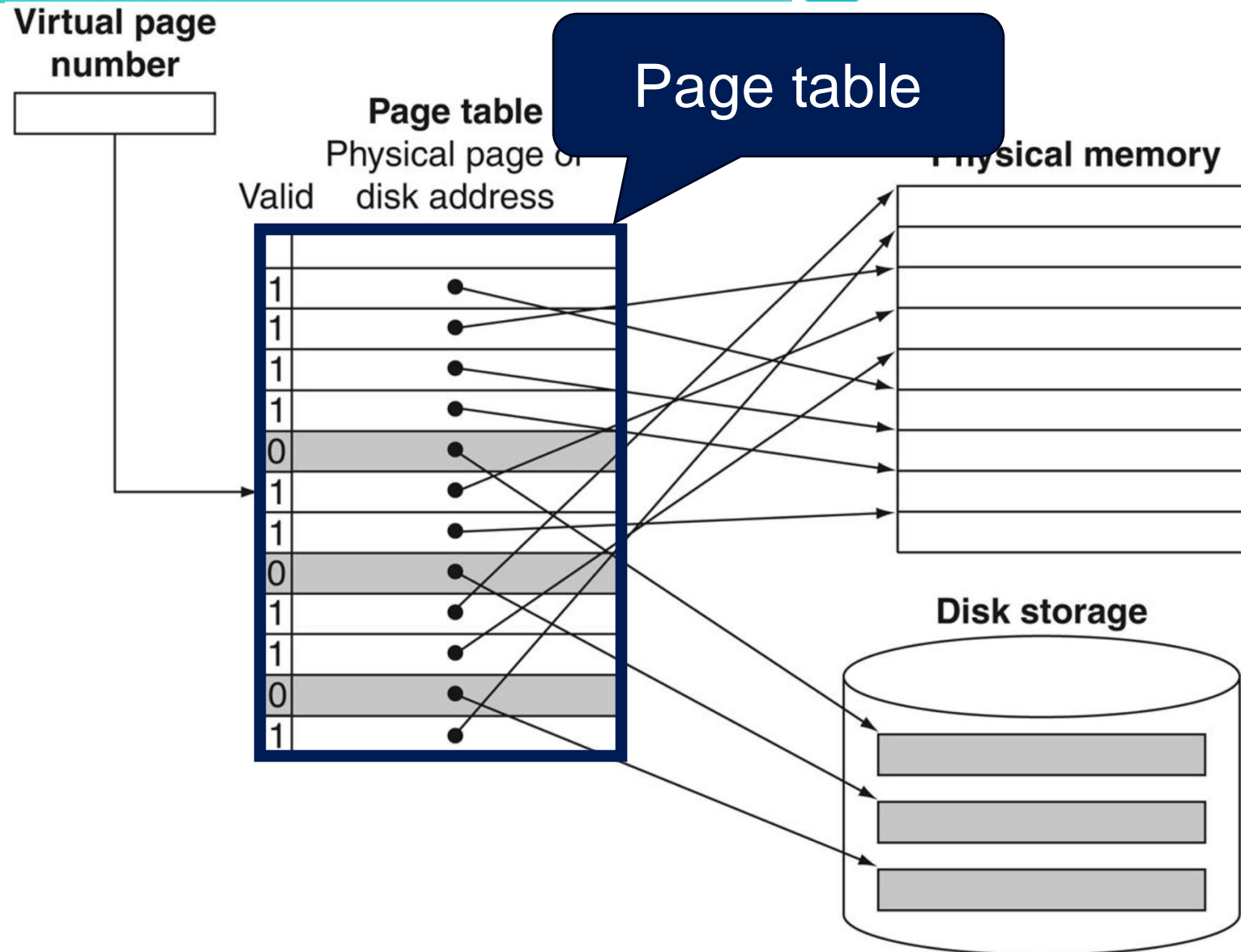


Page Table: Address Translation



- CPU converts virtual addresses into physical addresses via an OS-managed lookup table called **page table**
- **Mapping from a virtual to physical address**
 - **Virtual address** = virtual page number + page offset
 - **Physical address** = physical page number + page offset
- Each program has its own page table!

Page Table: Details



Page Table: Details

Virtual page
number

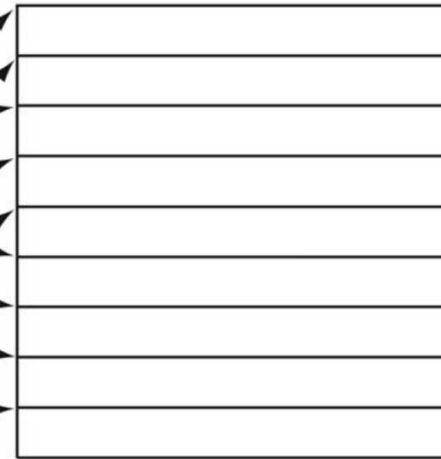
#5

Page table

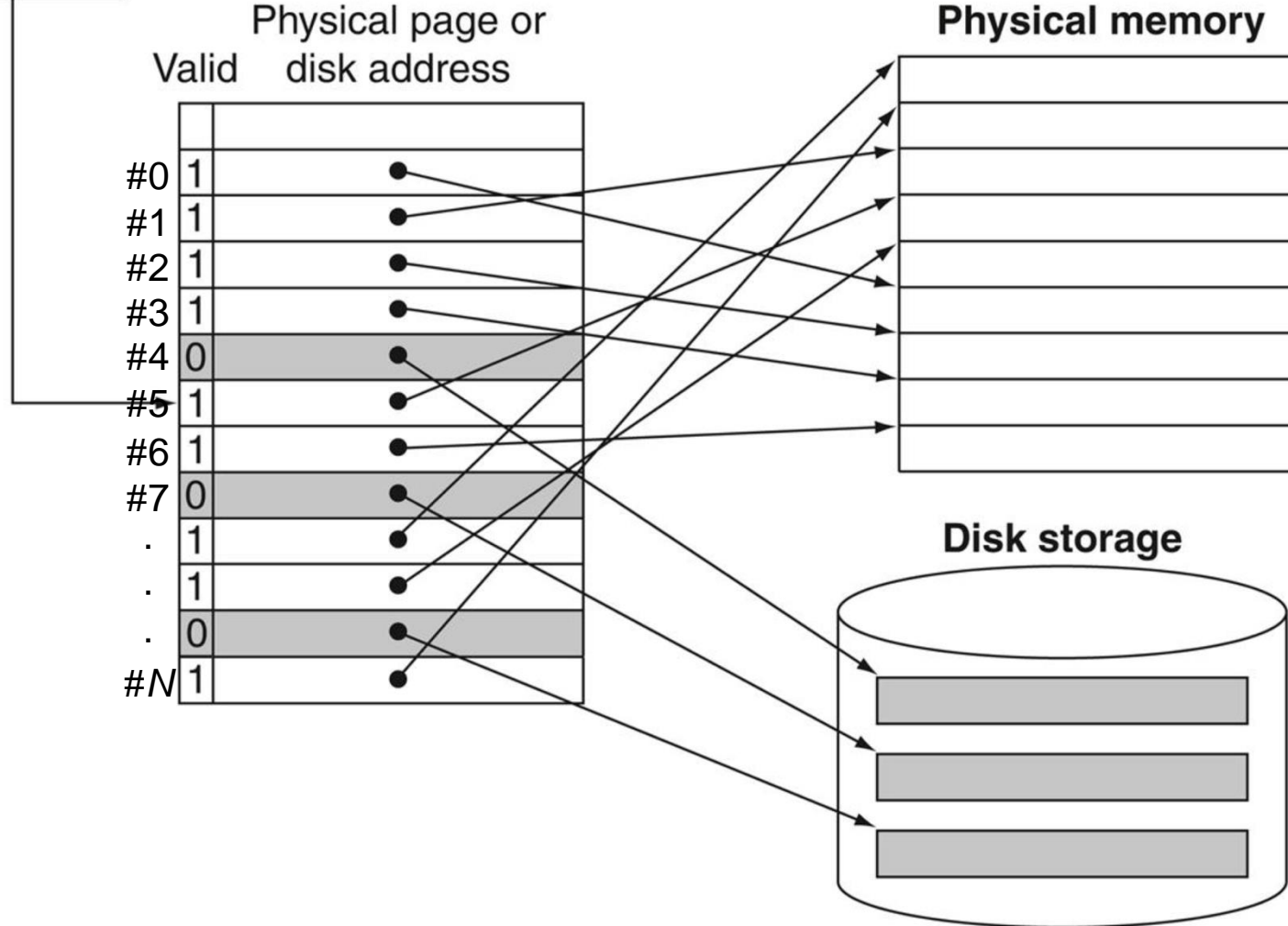
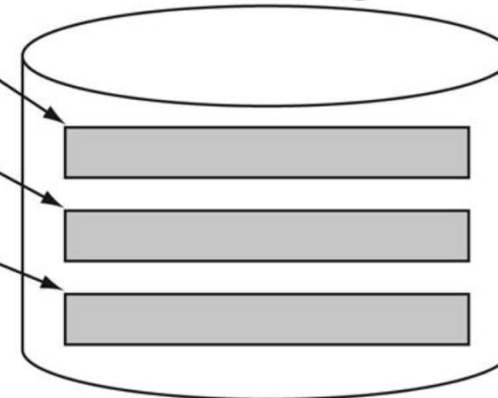
Physical page or
Valid disk address

	Valid	Physical page or disk address
#0	1	•
#1	1	•
#2	1	•
#3	1	•
#4	0	•
#5	1	•
#6	1	•
#7	0	•
.	1	•
.	1	•
.	0	•
#N	1	•

Physical memory



Disk storage



Page Table: Details

Virtual page number

#5

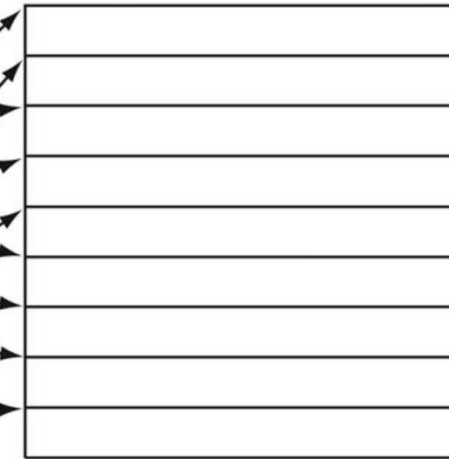
Page table

Physical page or
Valid disk address

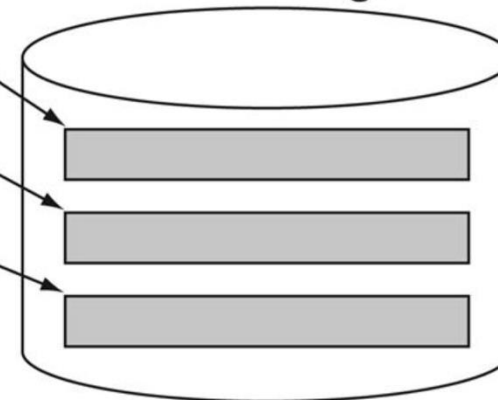
	Valid	Physical page or disk address
#0	1	•
#1	1	•
#2	1	•
#3	1	•
#4	0	•
#5	1	•
#6	1	•
#7	0	•
.	1	•
.	1	•
.	0	•
#N	1	•

Entry: Physical page number
(or disk address)

Physical memory

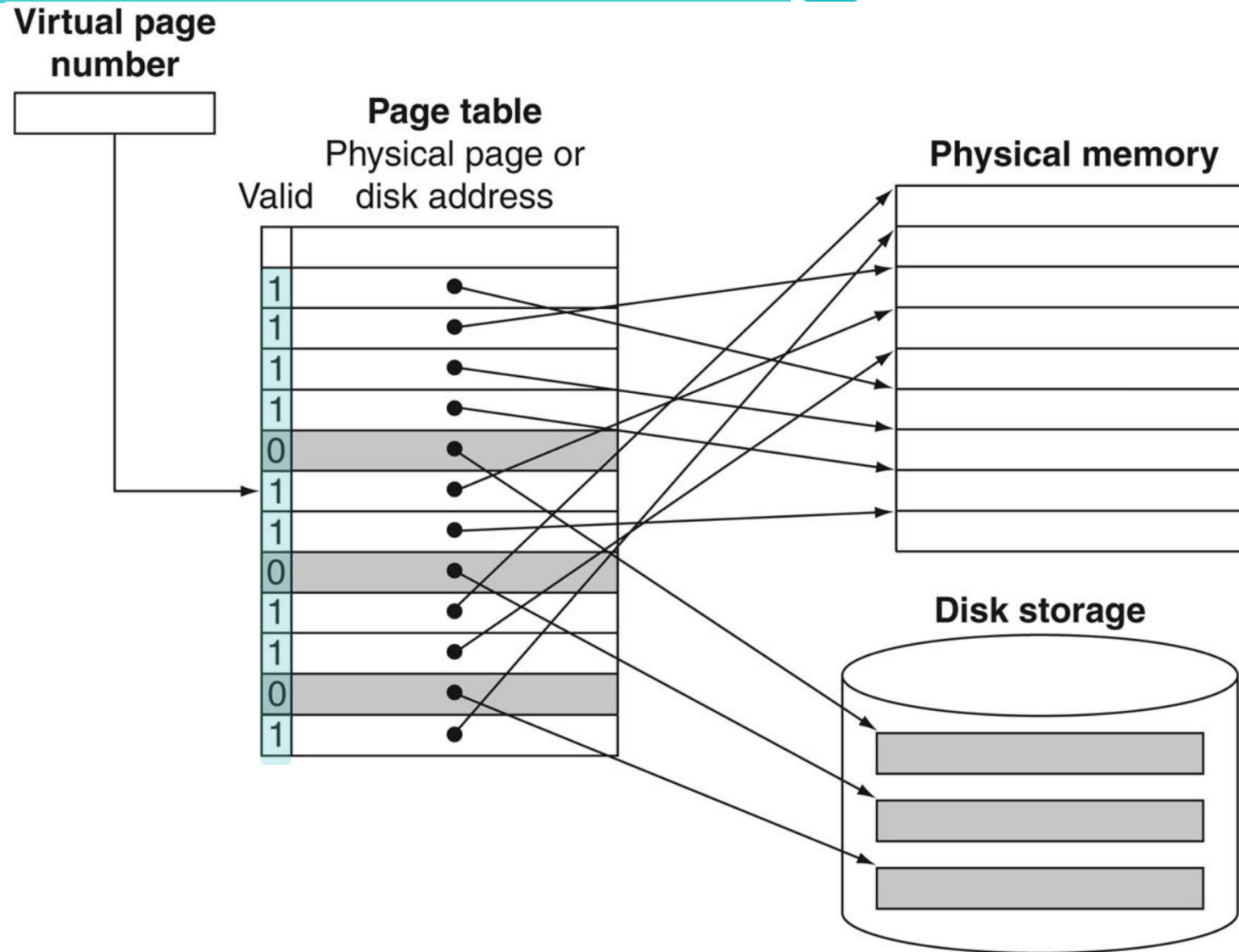


Disk storage



Index: virtual page number (as offset)
(+ page table base address)

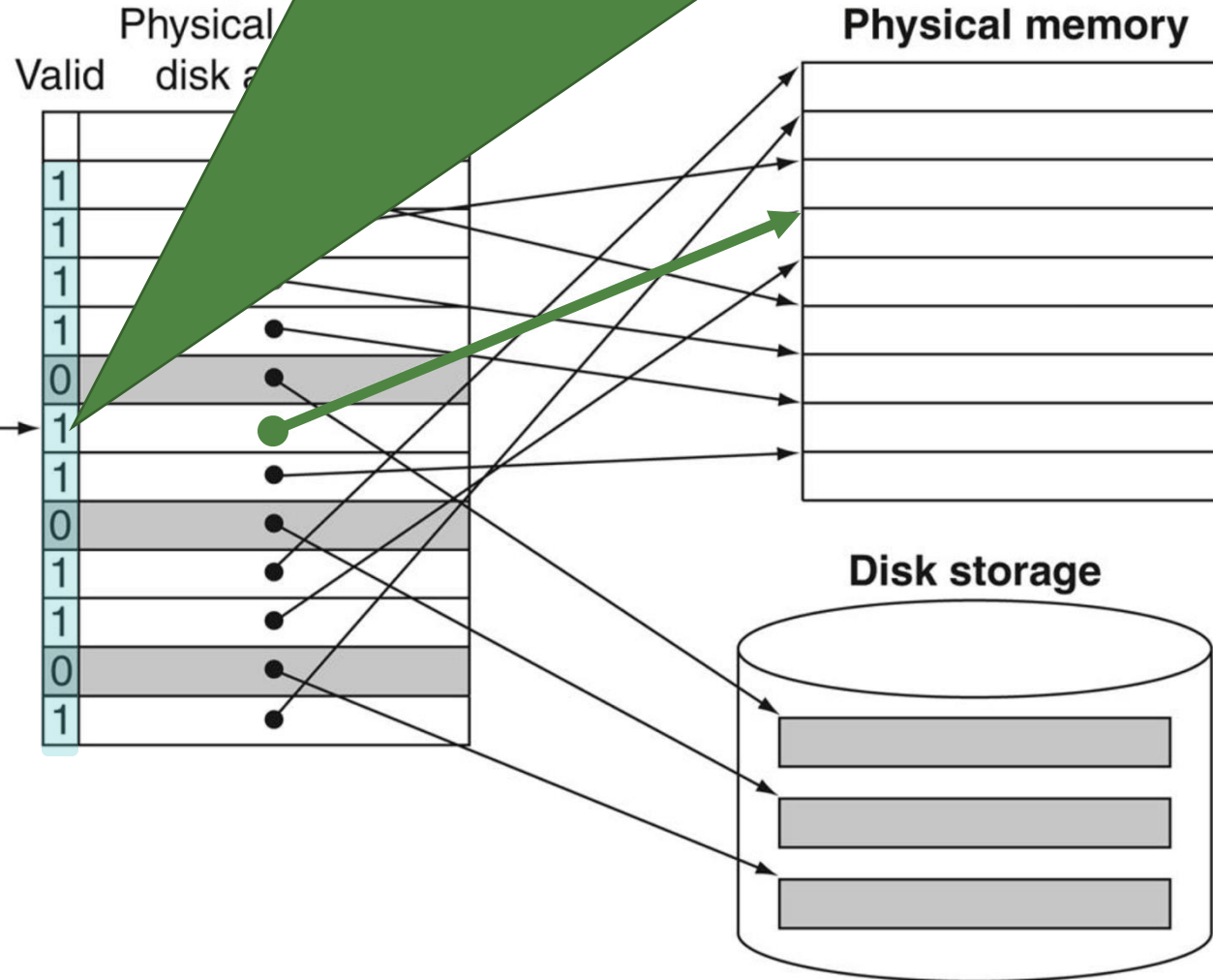
Valid Bit in Page Table



Valid Bit in Page Table

Virtual page number

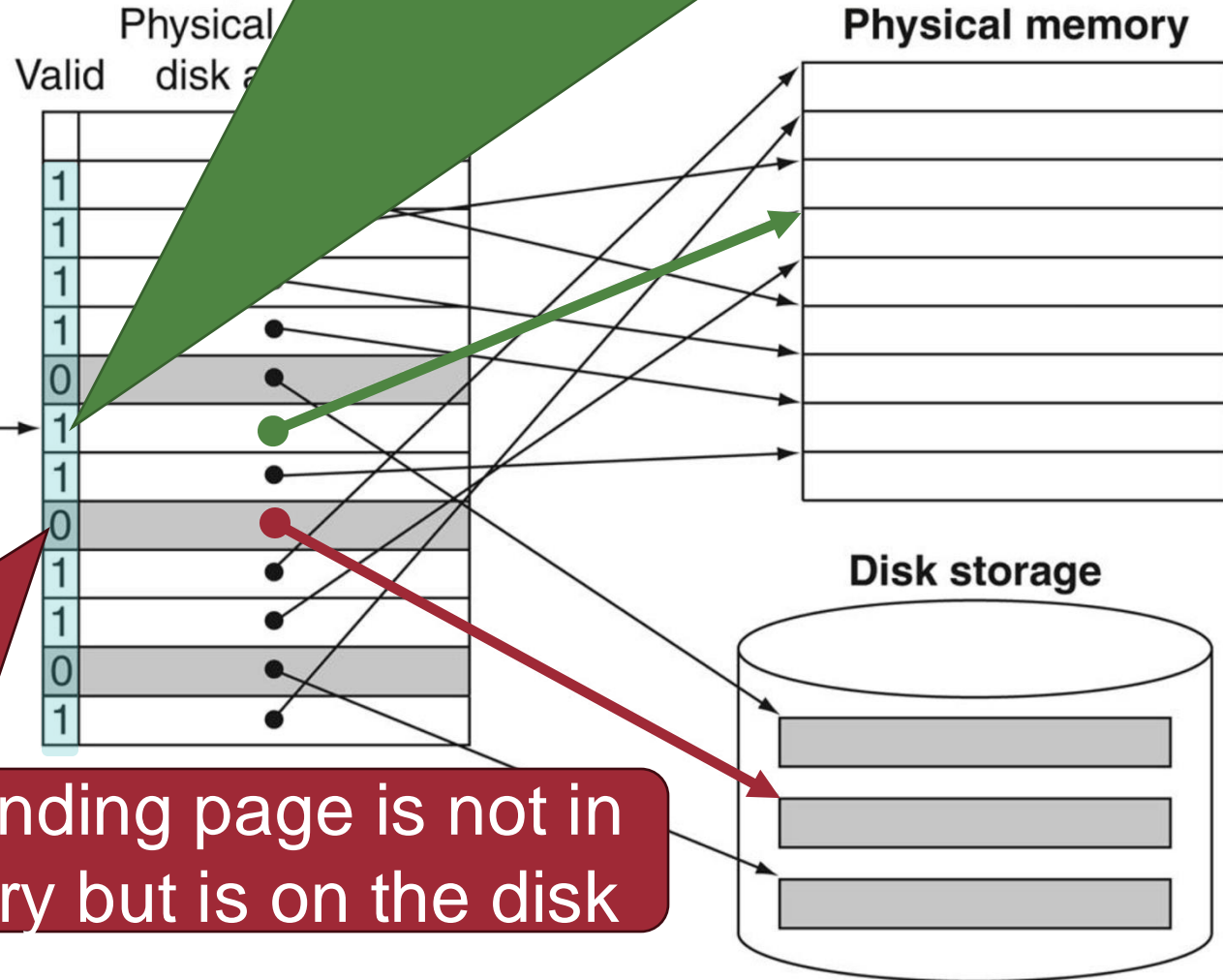
1: the corresponding page is in physical memory (i.e., mapped physical address exists in the entry)



Valid Bit in Page Table

Virtual page number

1: the corresponding page is in physical memory (i.e., mapped physical address exists in the entry)



0: the corresponding page is not in physical memory but is on the disk

Virtual Memory Hit

Virtual page number

#5

Page table

Physical page or
disk address

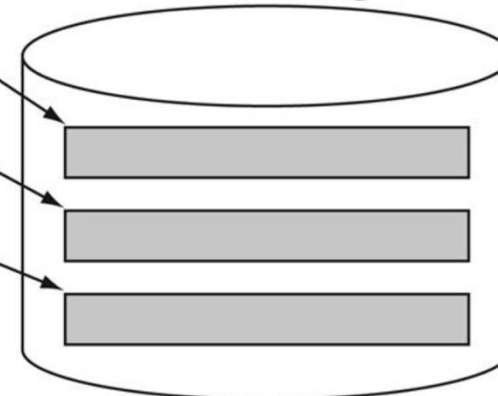
	Valid	Physical page or disk address
#0	1	•
#1	1	•
#2	1	•
#3	1	•
#4	0	•
#5	1	•
#6	1	•
#7	0	•
.	1	•
.	1	•
.	0	•
#N	1	•

Physical memory



Do not need to
access to the disk

Disk storage



Virtual Memory Miss (Page Fault)

Virtual page
number

#7

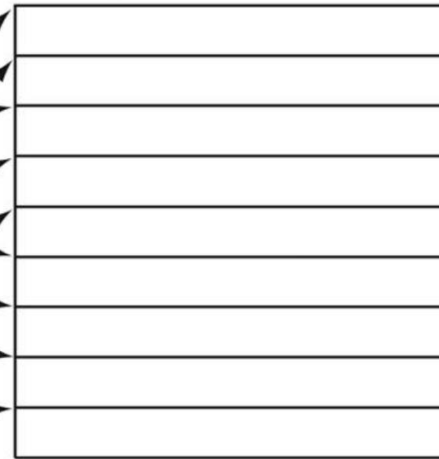
Page table

Physical page or
disk address

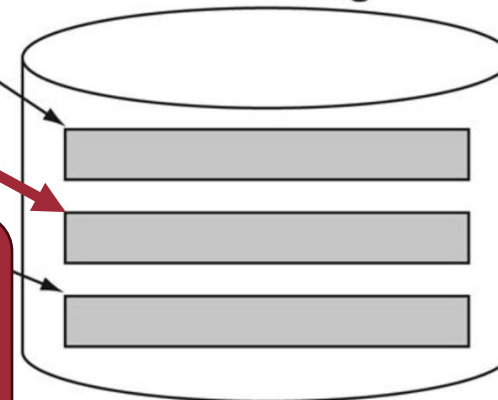
Valid

	Valid	Physical page or disk address
#0	1	•
#1	1	•
#2	1	•
#3	1	•
#4	0	•
#5	1	•
#6	1	•
#7	0	•
•	1	•
•	1	•
•	0	•
#N	1	•

Physical memory



Disk storage



Miss (valid bit 0): access on memory space not in physical memory (but in disk)

Virtual Memory Miss (Page Fault)

Virtual page
number

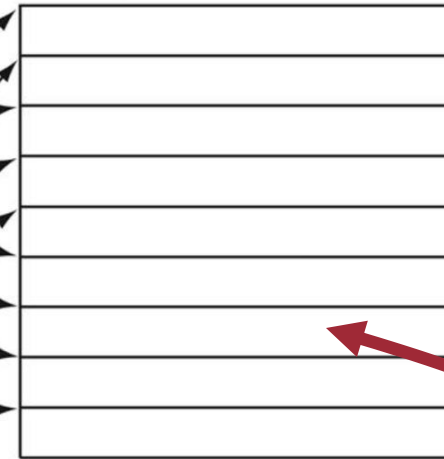
#7

Page table

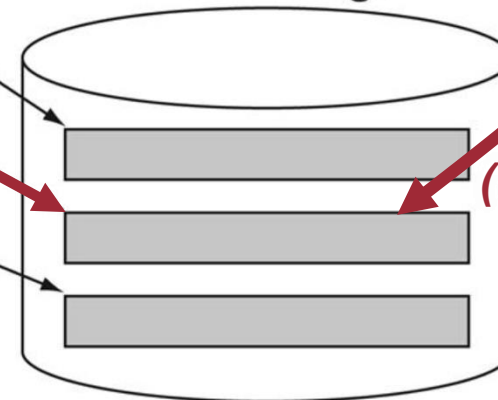
Physical page or
Valid disk address

	Valid	Physical page or disk address
#0	1	•
#1	1	•
#2	1	•
#3	1	•
#4	0	•
#5	1	•
#6	1	•
#7	0	•
•	1	•
•	1	•
•	0	•
#N	1	•

Physical memory



Disk storage



**Paging
(swapping)**

(1) Fetch "page" to the
physical memory
(write back depending
on the situation)

Virtual Memory Miss (Page Fault)

Virtual page number

#7

Page table

Physical page or
Valid disk address

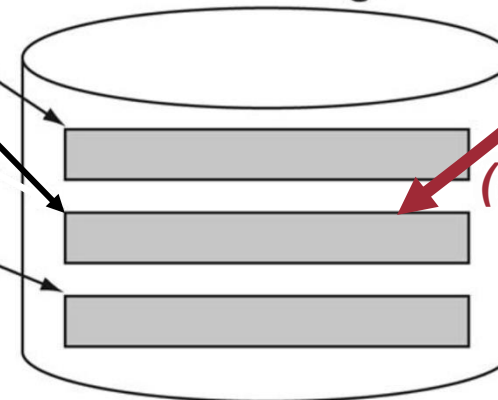
	Valid	Physical page or disk address
#0	1	•
#1	1	•
#2	1	•
#3	1	•
#4	0	•
#5	1	•
#6	1	•
#7	1	•
•	1	•
•	1	•
•	0	•
#N	1	•

(2) Update the address translation table

Physical memory



Disk storage

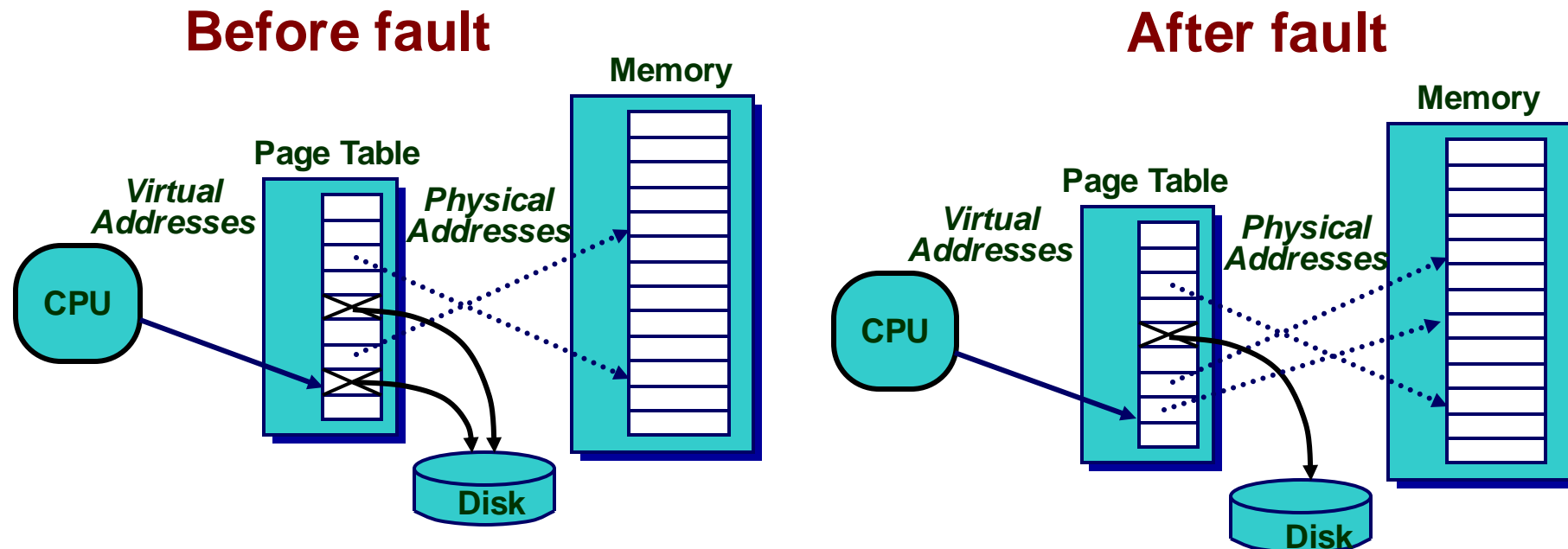


(1) Fetch "page" to the physical memory (write back depending on the situation)

Paging (swapping)

Virtual Memory Miss (Page Fault)

- If a page is not in physical memory but disk (if valid == 0)
 - Page table entry (especially, valid bit) indicates that the page not in memory
 - **Page fault exception** is occurred! **OS trap handler** invoked to move data from disk into memory
 - OS has full control over placement!



Page Fault Penalty



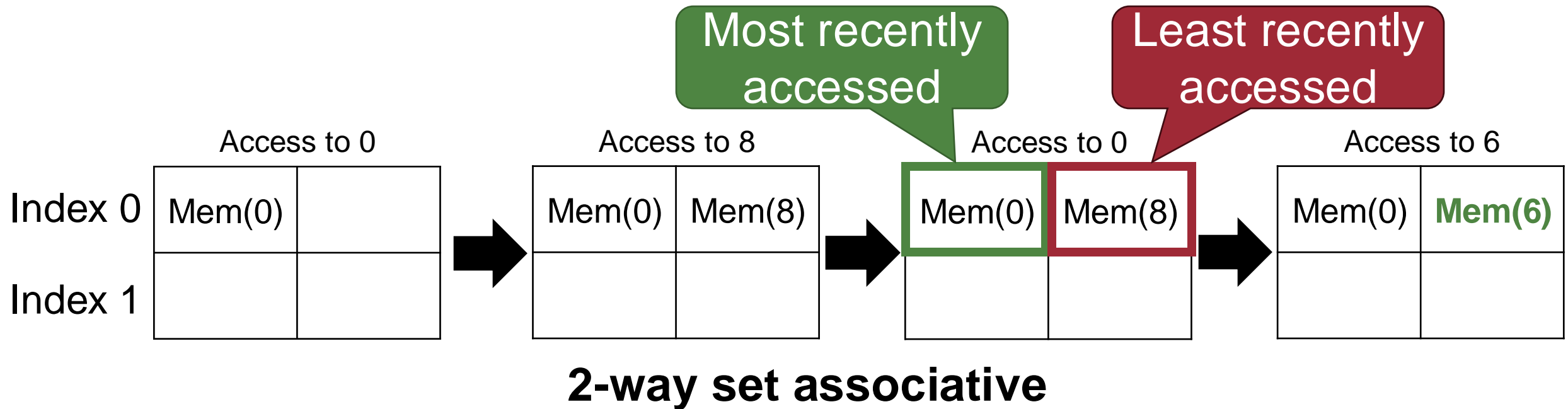
- On page fault, the page must be fetched from disk
 - Takes millions of clock cycles
- Try to minimize page fault rate and fault penalty
 - Prefer **write back** (write through is impractical)
 - **Dirty** bit in each page table entry
 - Prefer **LRU** replacement
 - Reference bit in each page table set to 1 on access to page
 - Periodically cleared to 0 by OS
 - A page with **reference** bit = 0 has NOT been used recently

Reference	Dirty	Valid	Physical page number
-----------	-------	-------	----------------------

Page table entry

Recap: LRU

- **Least Recently Used (LRU):** replace the one NOT used (accessed) for the longest time
 - Temporal locality of access is considered
 - Need a reference history information



Place of the Page Table?

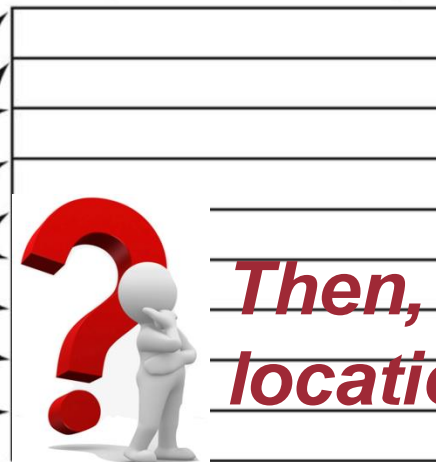
Virtual page
number

Page table

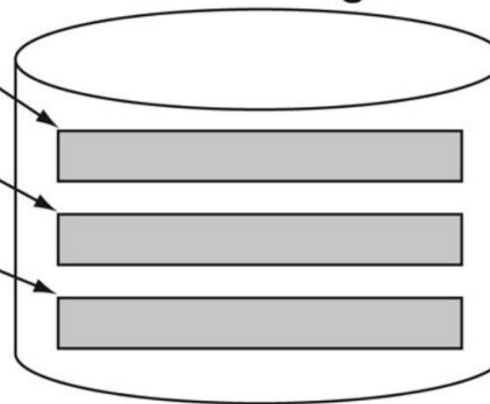
Valid Physical page or
disk address

1	•
1	•
1	•
1	•
0	•
1	•
1	•
0	•
1	•
1	•
0	•
1	•

Physical memory



Disk storage



*Where does the
page table exist?*

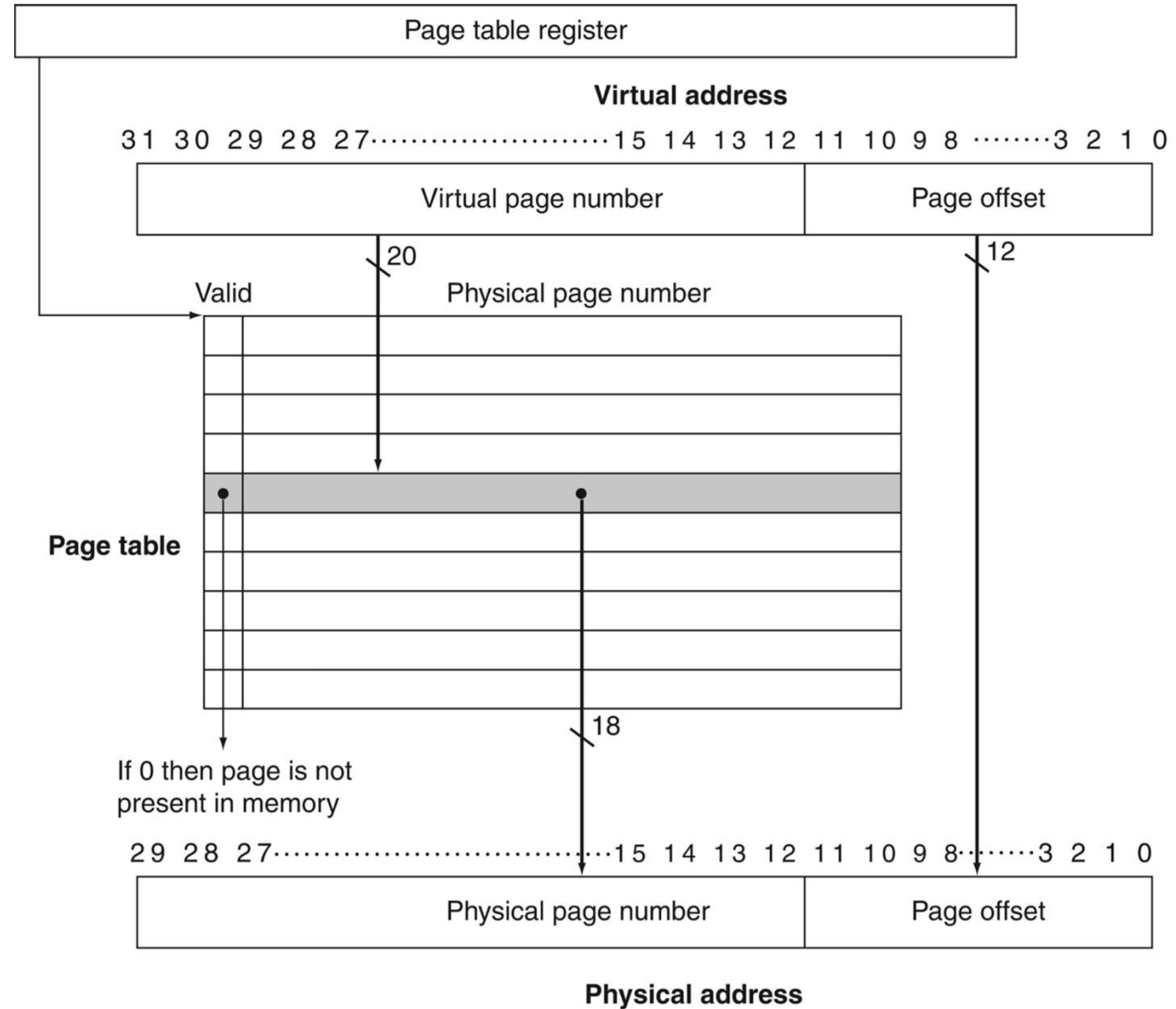
Physical memory!

*Then, how can we know the
location of the page table itself?*

Use a page table (base) register

Page Table Register

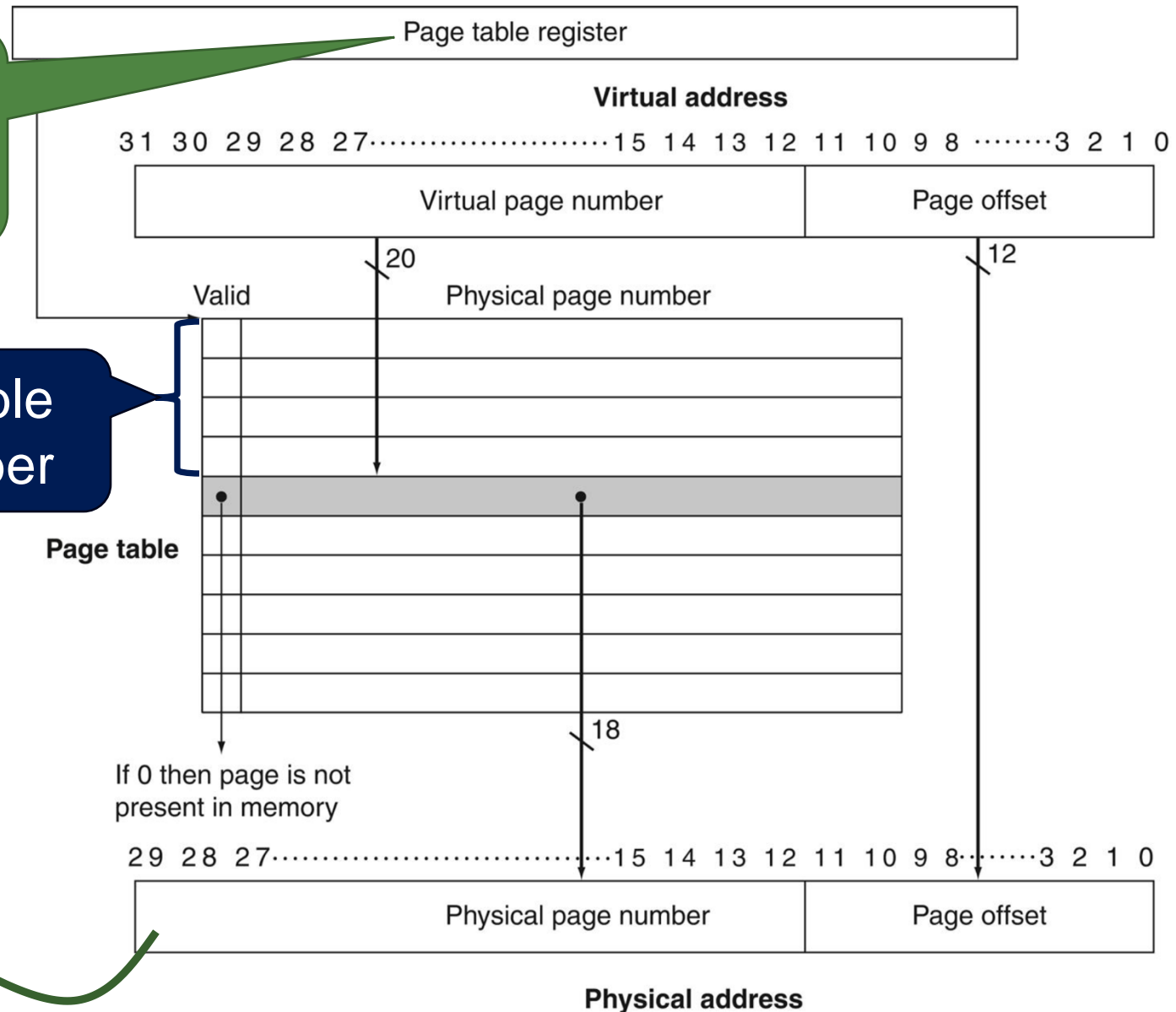
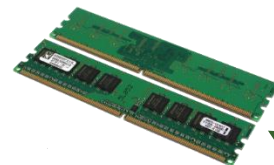
68



Page Table Register

CPU register pointing to
page table in physical memory
(Absolute address)

Index: value of the page table
register + virtual page number



Question?