

# Memory Hierarchy

'22H2

송 인 식

# Outline

- Storage technologies and trends
- Locality of reference
- Caching in the memory hierarchy

# Random-Access Memory (RAM)

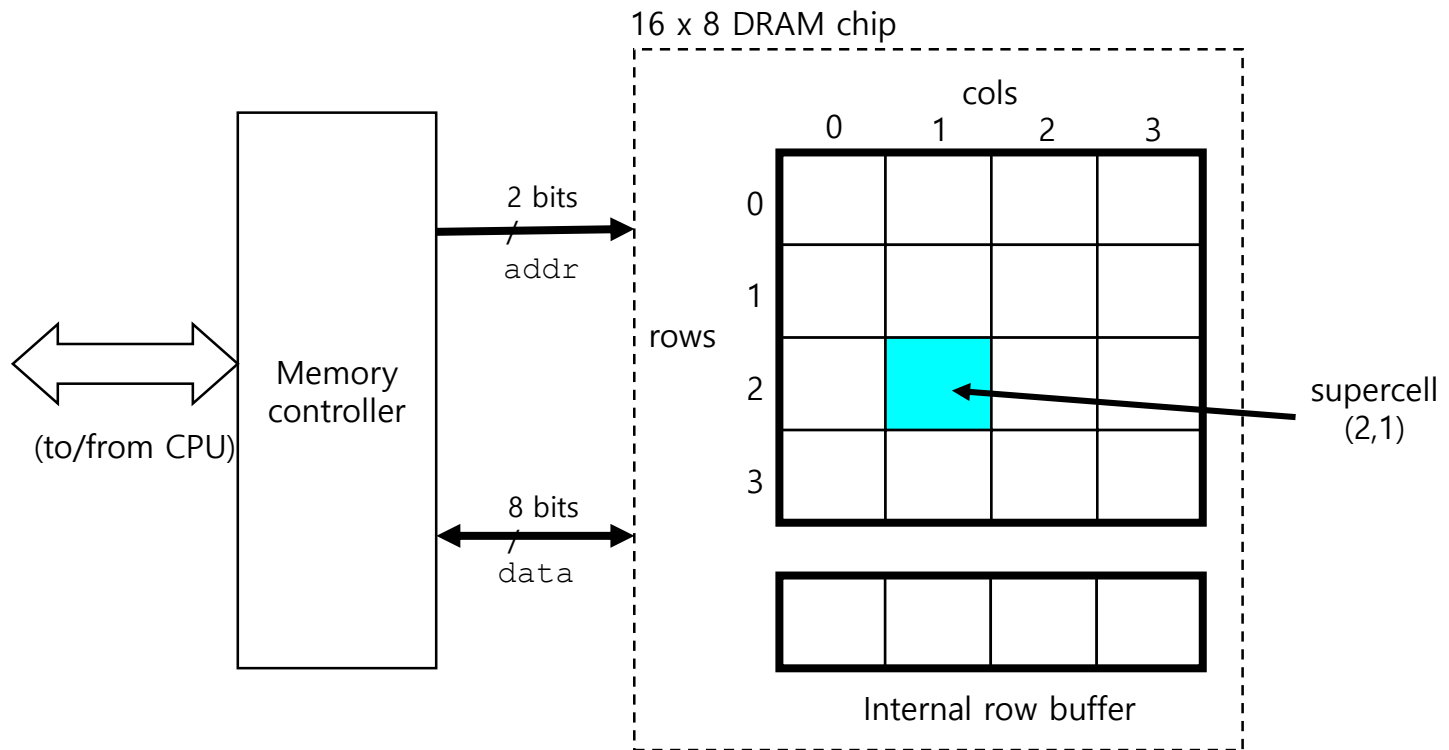
- Key features
  - RAM is traditionally packaged as a chip.
  - Basic storage unit is normally a cell (one bit per cell).
  - Multiple RAM chips form a memory.
- Static RAM (SRAM)
  - Each cell stores a bit with a four or six-transistor circuit.
  - Retains value indefinitely, as long as it is kept powered.
  - Relatively insensitive to electrical noise (EMI), radiation, etc.
  - Faster and more expensive than DRAM.
- Dynamic RAM (DRAM)
  - Each cell stores bit with a capacitor. One transistor is used for access
  - Value must be refreshed every 10-100 ms.
  - More sensitive to disturbances (EMI, radiation,...) than SRAM.
  - Slower and cheaper than SRAM.

# SRAM vs DRAM Summary

	Trans. per bit	Access time	Needs refresh?	Needs EDC?	Cost	Applications
SRAM	4 or 6	1X	No	Maybe	100x	Cache memories
DRAM	1	10X	Yes	Yes	1X	Main memories, frame buffers

# Conventional DRAM Organization

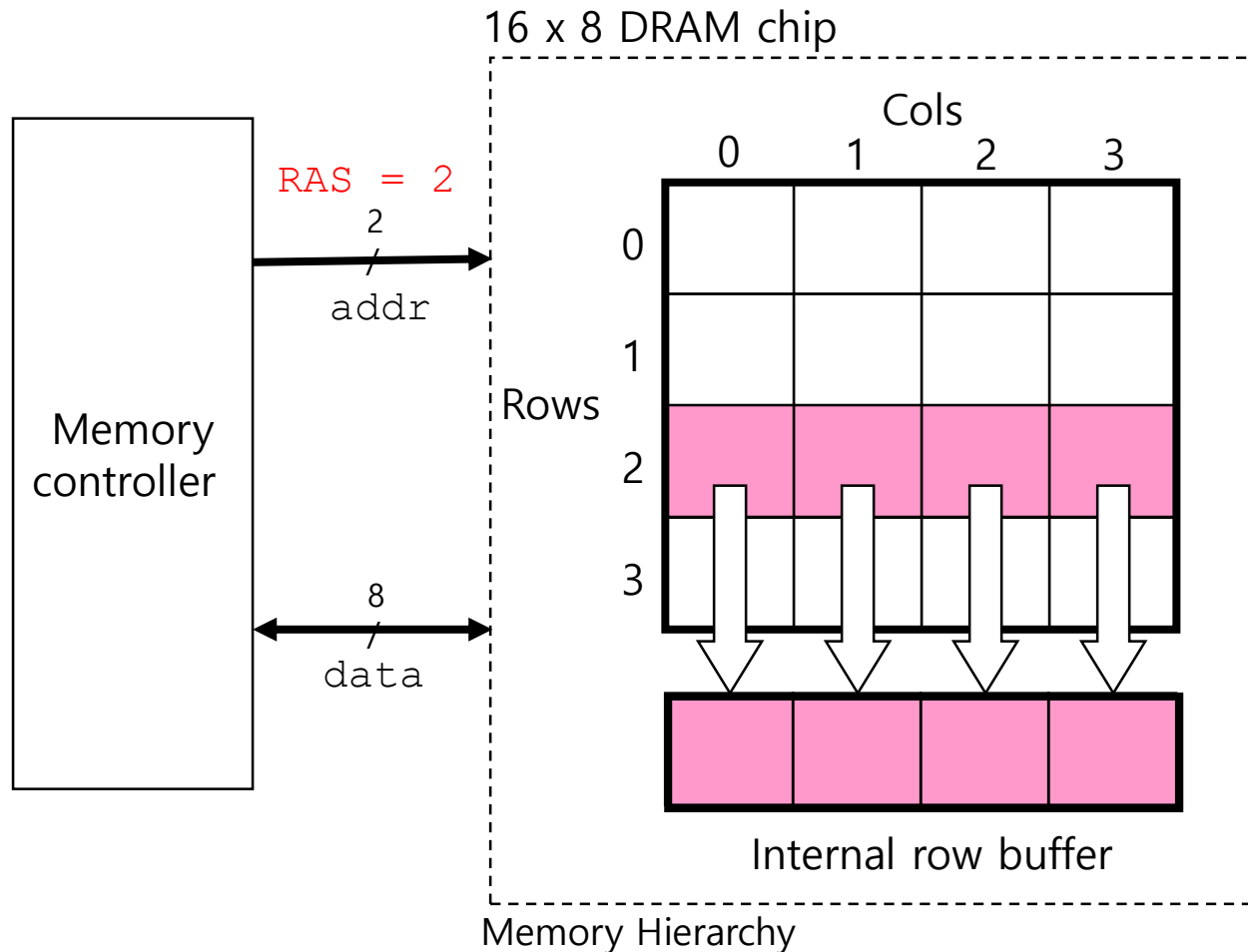
- $d \times w$  DRAM:
  - $dw$  total bits organized as  $d$  **supercells** of size  $w$  bits



# Reading DRAM Supercell (2,1)

Step 1(a): Row access strobe (**RAS**) selects row 2.

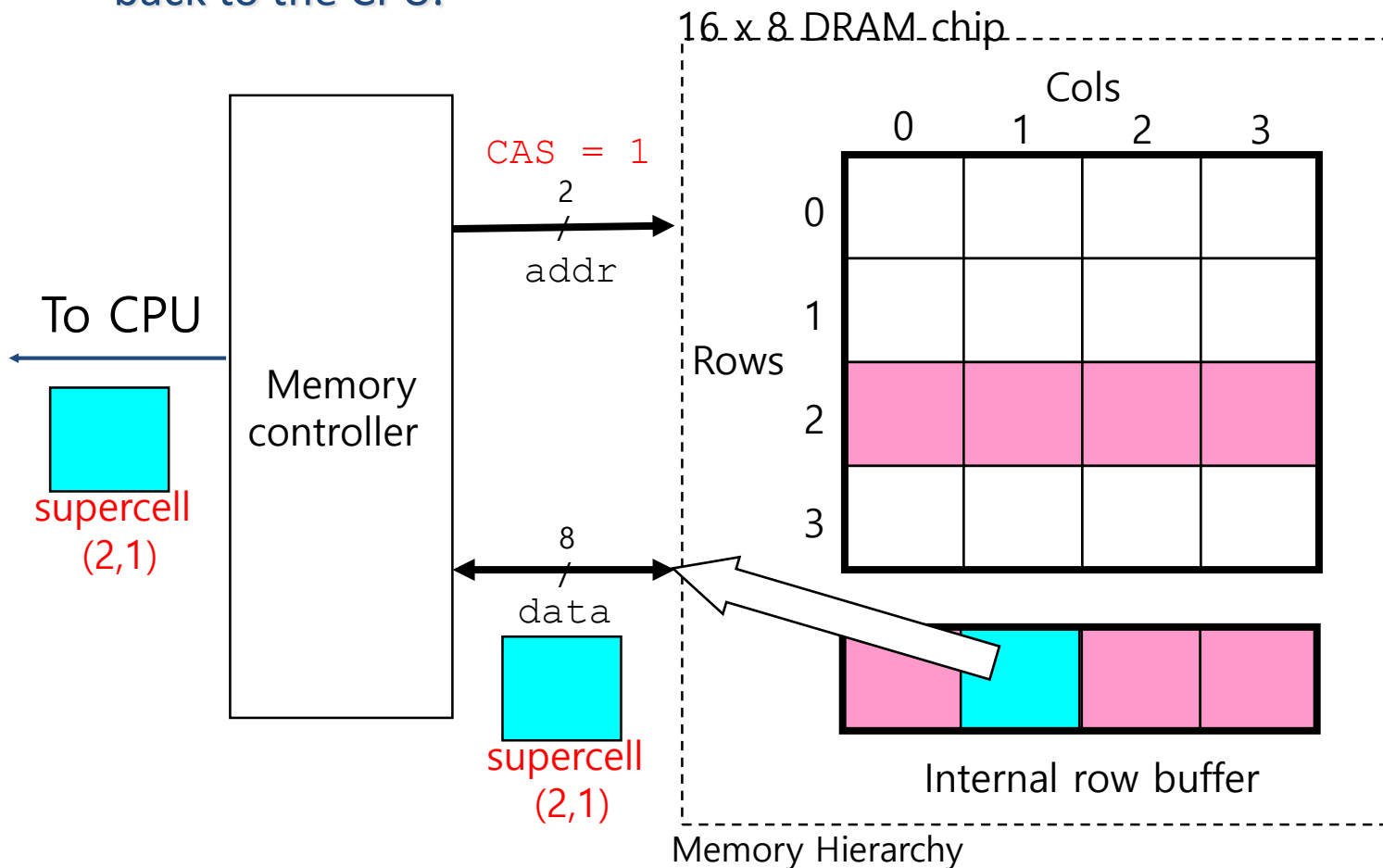
Step 1(b): Row 2 copied from DRAM array to row buffer.



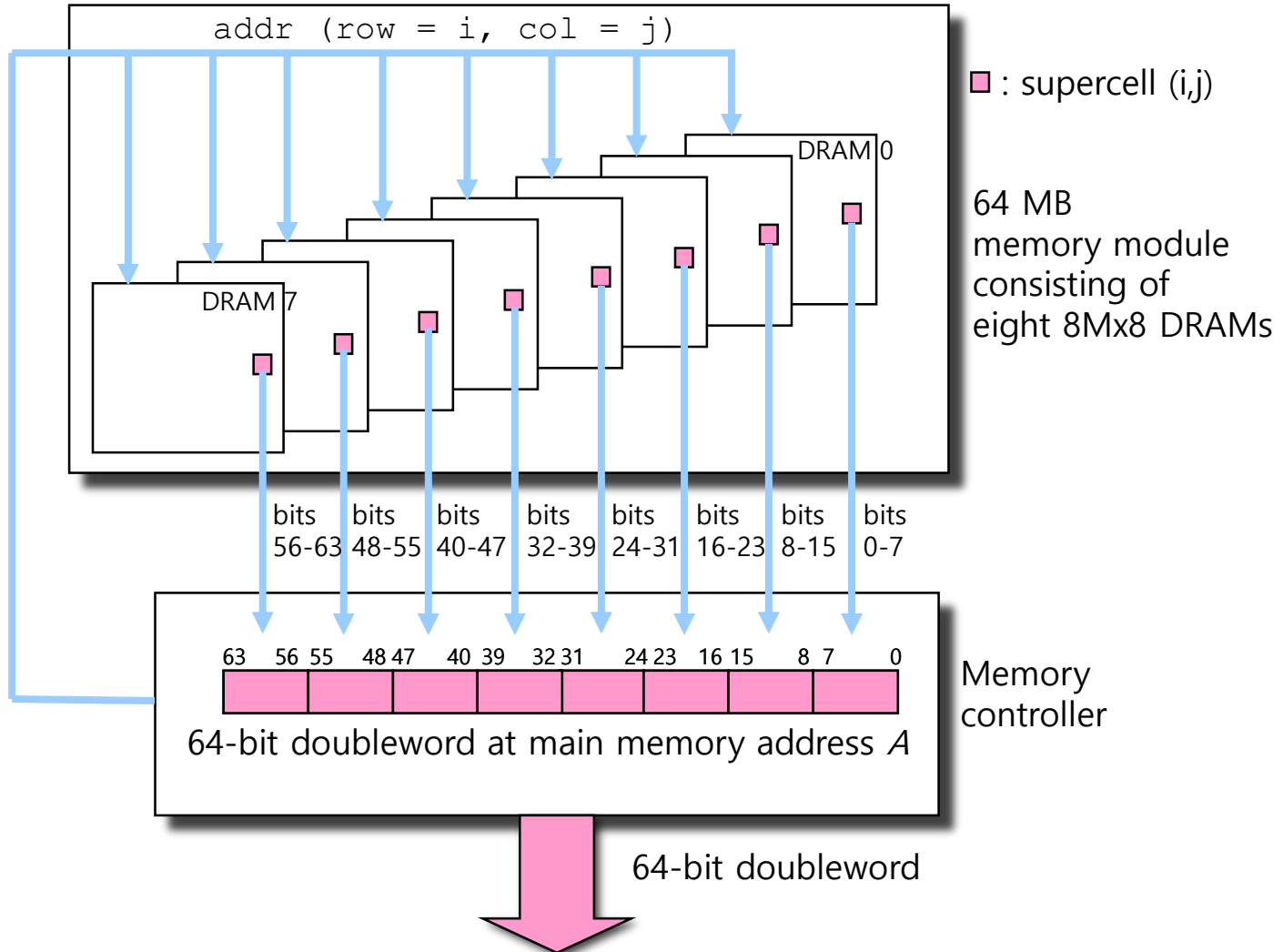
# Reading DRAM Supercell (2,1)

Step 2(a): Column access strobe (**CAS**) selects column 1.

Step 2(b): Supercell (2,1) copied from buffer to data lines, and eventually back to the CPU.



# Memory Modules





# Enhanced DRAMs

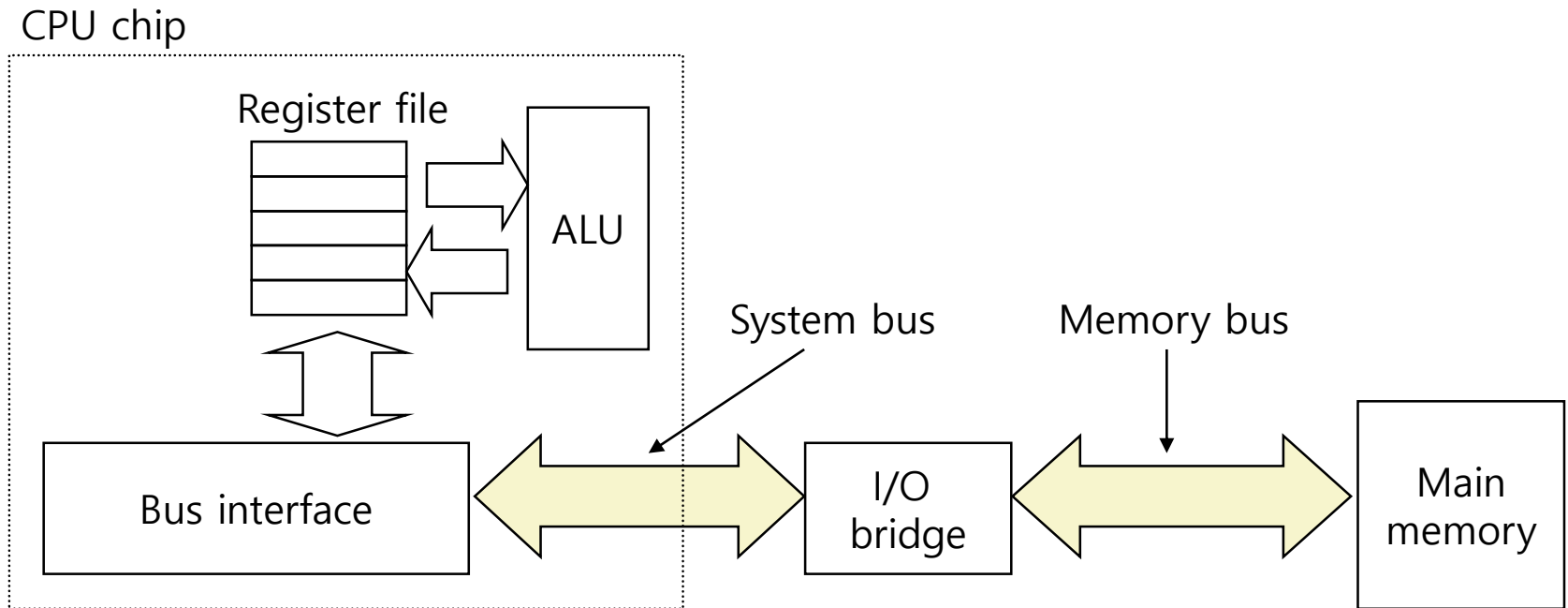
- Basic DRAM cell has not changed since its invention in 1966.
  - Commercialized by Intel in 1970.
- DRAM cores with better interface logic and faster I/O :
  - Synchronous DRAM (**SDRAM**)
    - Uses a conventional clock signal instead of asynchronous control
    - Allows reuse of the row addresses (e.g., RAS, CAS, CAS, CAS)
  - Double data-rate synchronous DRAM (**DDR SDRAM**)
    - Double edge clocking sends two bits per cycle per pin
    - By 2010, standard for most server and desktop systems

# Nonvolatile Memories

- DRAM and SRAM are volatile memories
  - Lose information if powered off.
- Nonvolatile memories retain value even if powered off
  - Read-only memory (ROM): programmed during production
  - Programmable ROM (PROM): can be programmed once
  - Erasable PROM (EPROM): can be bulk erased (UV, X-Ray)
  - Electrically erasable PROM (EEPROM): electronic erase capability
  - Flash memory: EEPROMs with partial erase capability
    - Wears out after 1000 ~ 100,000 erases.
- Uses for Nonvolatile Memories
  - Firmware programs stored in a ROM (BIOS, controllers for disks, network cards, graphics accelerators, security subsystems,...)
  - Solid state disks (replace rotating disks in thumb drives, smart phones, mp3 players, tablets, laptops,...)
  - Disk caches

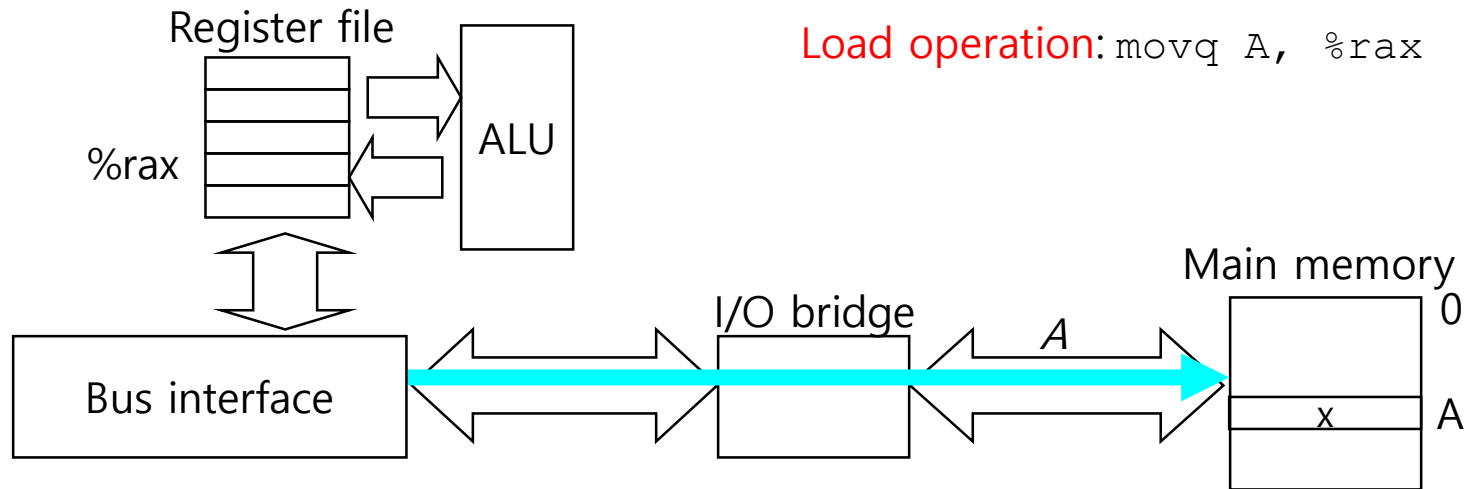
# Traditional Bus Structure Connecting CPU and Memory

- A **bus** is a collection of parallel wires that carry address, data, and control signals.
- Buses are typically shared by multiple devices.



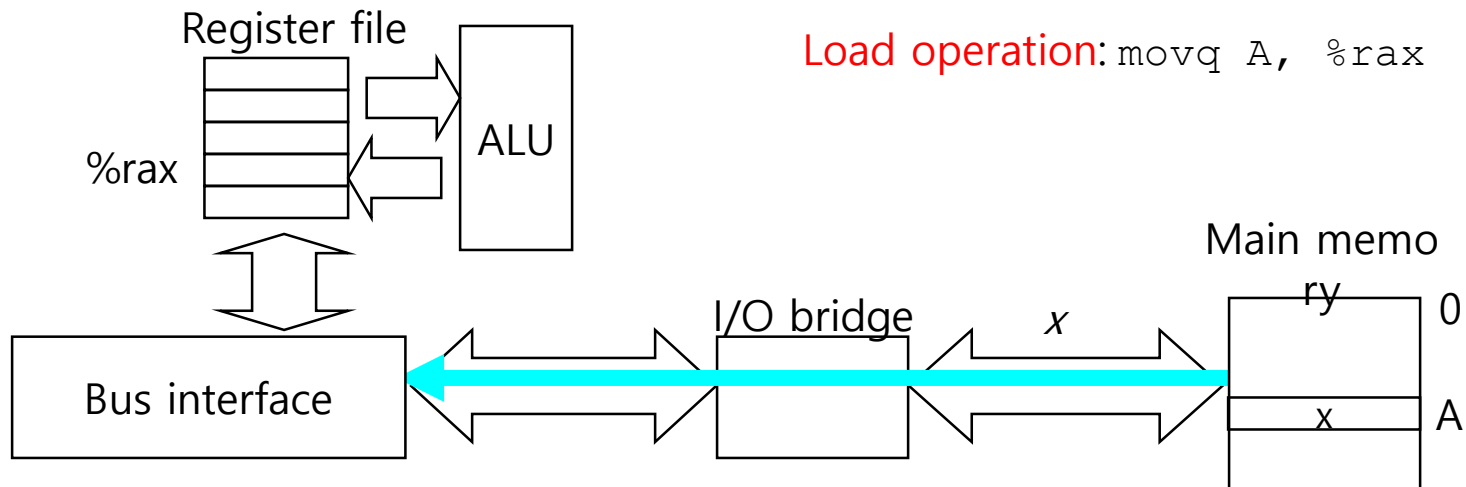
# Memory Read Transaction (1)

- CPU places address  $A$  on the memory bus.



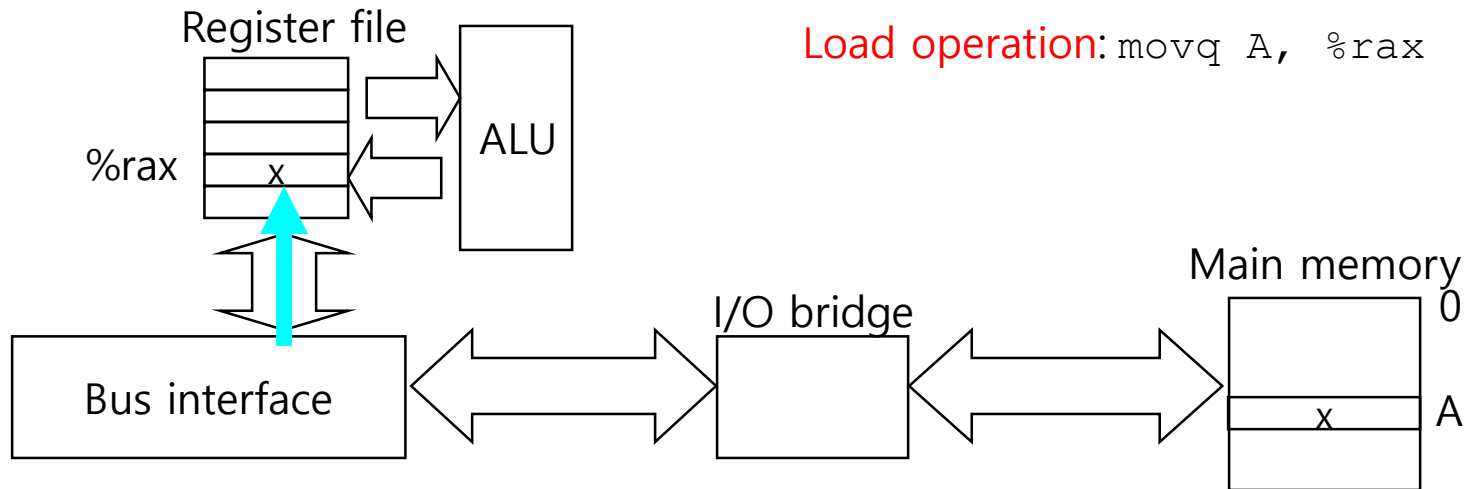
# Memory Read Transaction (2)

- Main memory reads A from the memory bus, retrieves word x, and places it on the bus.



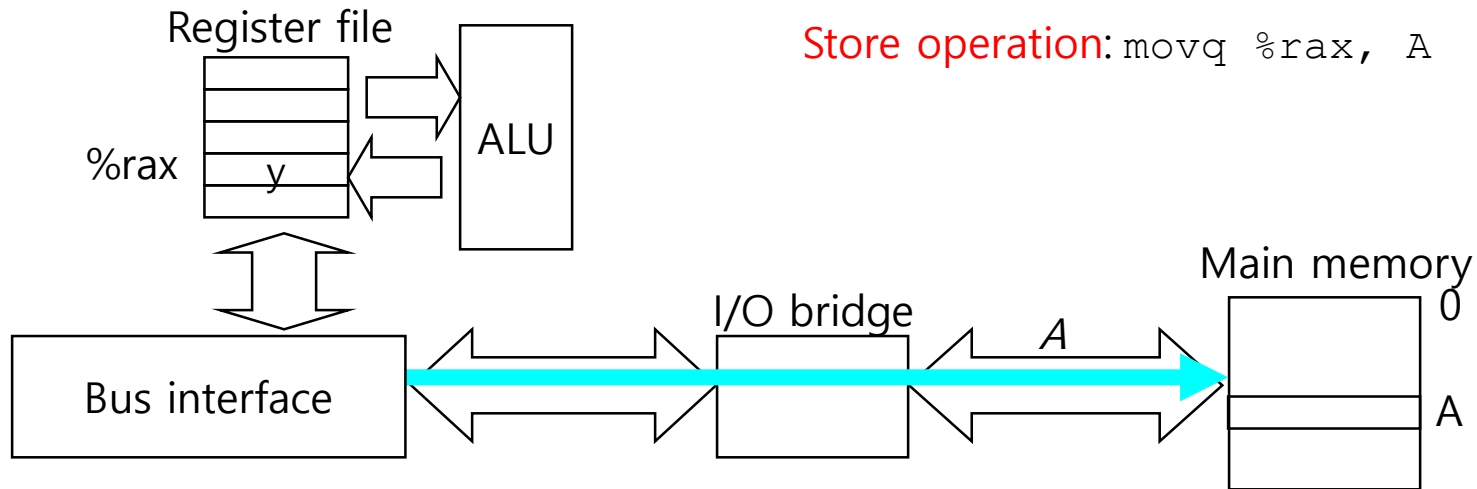
# Memory Read Transaction (3)

- CPU read word  $x$  from the bus and copies it into register `%rax`.



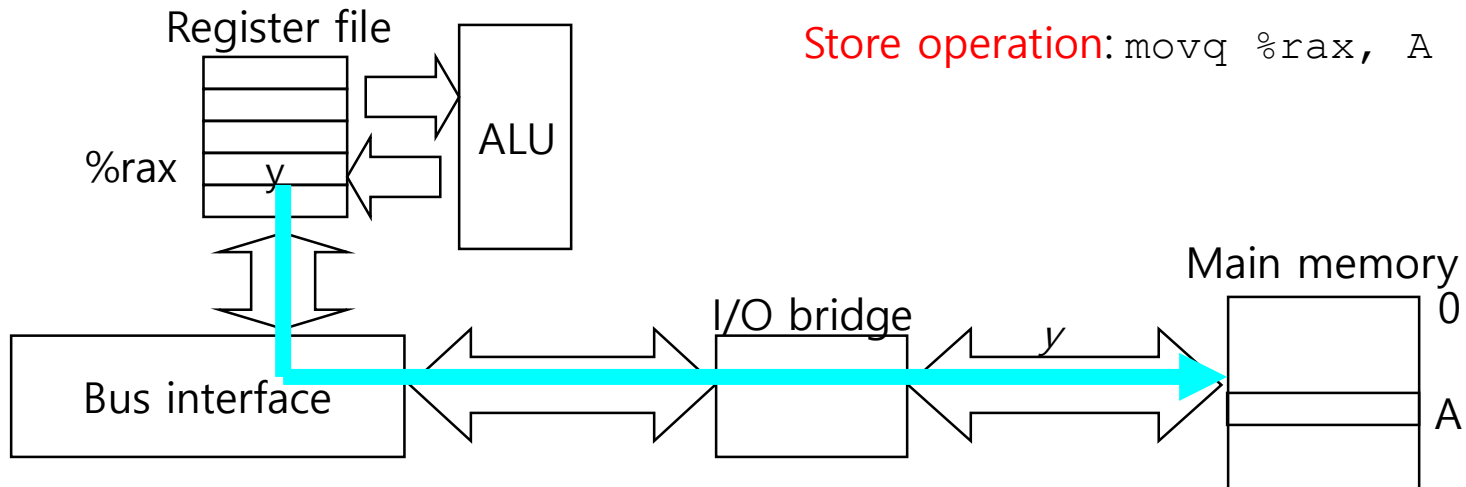
# Memory Write Transaction (1)

- CPU places address  $A$  on bus. Main memory reads it and waits for the corresponding data word to arrive.



# Memory Write Transaction (2)

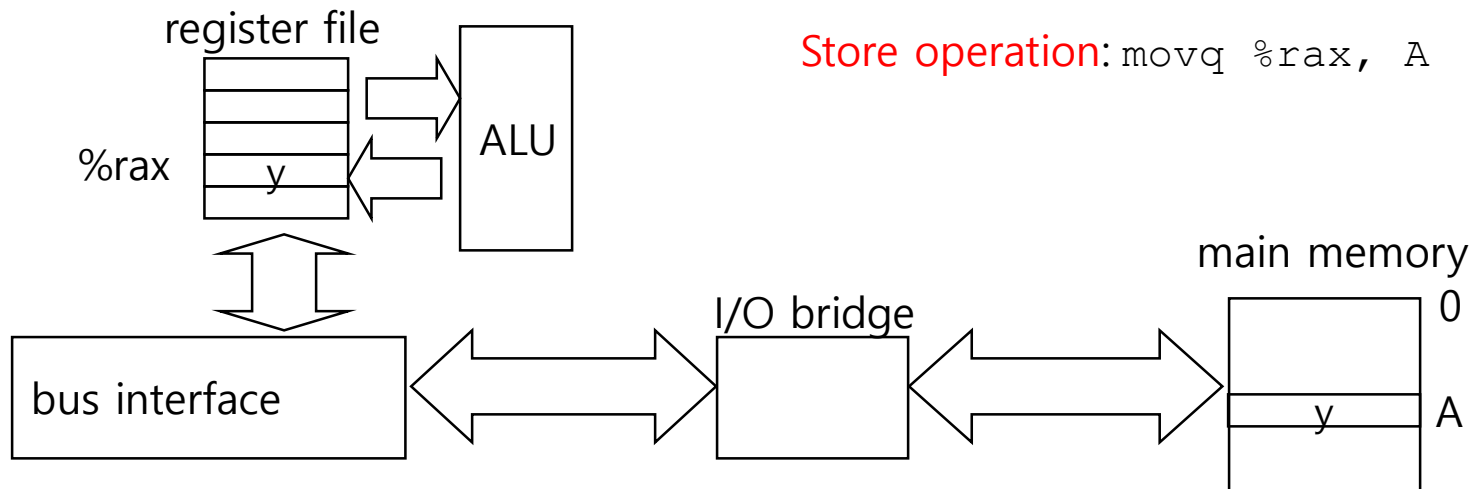
- CPU places data word  $y$  on the bus.



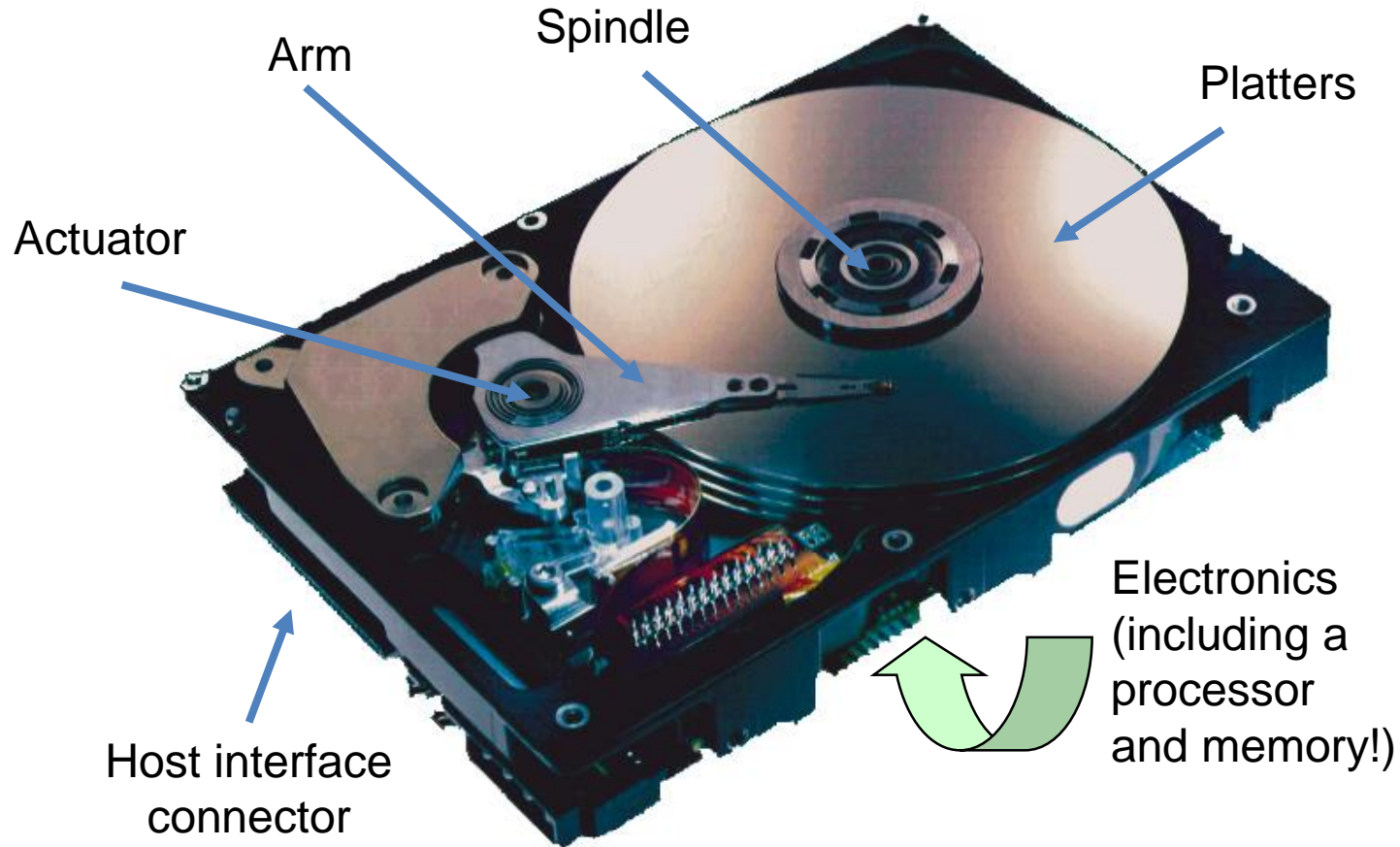


# Memory Write Transaction (3)

- Main memory reads data word  $y$  from the bus and stores it at address  $A$ .



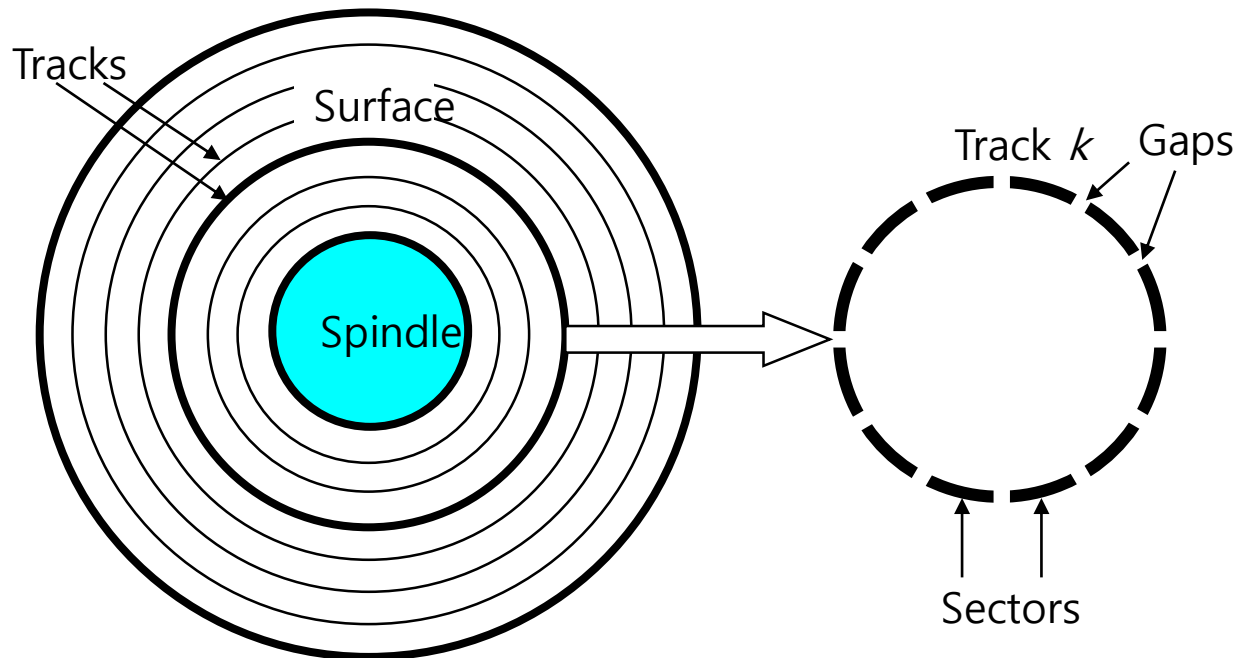
# What's Inside A Disk Drive?



*Image courtesy of Seagate Technology*

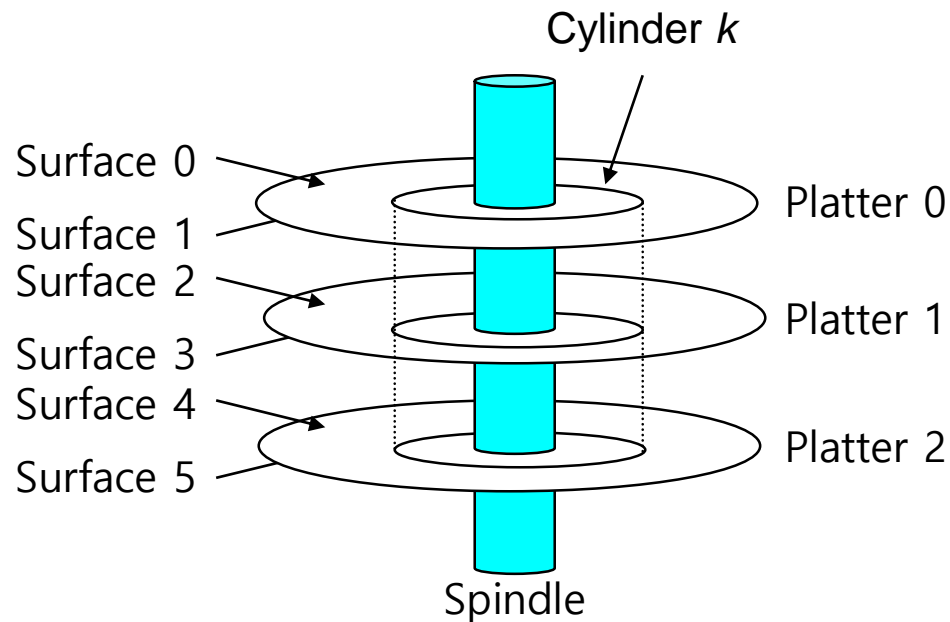
# Disk Geometry

- Disks consist of **platters**, each with two **surfaces**.
- Each surface consists of concentric rings called **tracks**.
- Each track consists of **sectors** separated by **gaps**.



# Disk Geometry (Multiple-Platter View)

- Aligned tracks form a cylinder.



# Disk Capacity

- **Capacity**: maximum number of bits that can be stored.
  - Vendors express capacity in units of gigabytes (GB), where  $1 \text{ GB} = 10^9 \text{ Bytes}$  (Lawsuit pending! Claims deceptive advertising).
- Capacity is determined by these technology factors:
  - **Recording density** (bits/in): number of bits that can be squeezed into a 1 inch segment of a track.
  - **Track density** (tracks/in): number of tracks that can be squeezed into a 1 inch radial segment.
  - **Areal density** (bits/in<sup>2</sup>): product of recording and track density.

# Recording Zones

- Modern disks partition tracks into disjoint subsets called **recording zones**
  - Each track in a zone has the same number of sectors, determined by the circumference of innermost track.
  - Each zone has a different number of sectors/track, , outer zones have more sectors/track than inner zones.
  - So we use **average** number of sectors/track when computing capacity.

# Disk Access Time

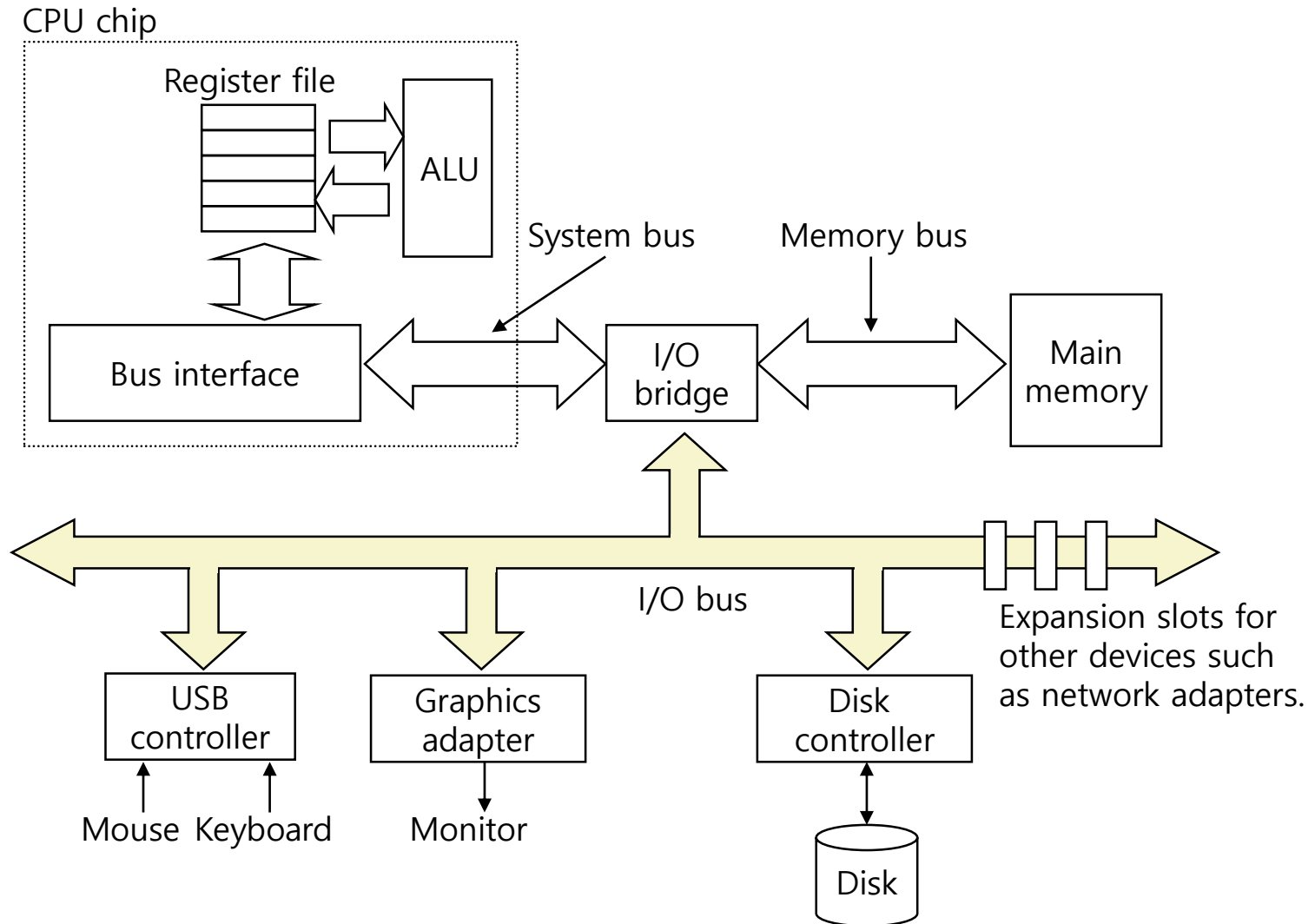
- Average time to access some target sector approximated by :
  - $T_{\text{access}} = T_{\text{avg\_seek}} + T_{\text{avg\_rotation}} + T_{\text{avg\_transfer}}$
- **Seek time** (  $T_{\text{avg\_seek}}$  )
  - Time to position heads over cylinder containing target sector.
  - Typical  $T_{\text{avg\_seek}}$  is 3~9 ms
- **Rotational latency** (  $T_{\text{avg\_rotation}}$  )
  - Time waiting for first bit of target sector to pass under r/w head.
  - $T_{\text{avg\_rotation}} = 1/2 \times 1/\text{RPMs} \times 60 \text{ sec}/1 \text{ min}$
  - Typical  $T_{\text{avg\_rotation}}$  : 7200 RPMs → 4.16 ms
- **Transfer time** (  $T_{\text{avg\_transfer}}$  )
  - Time to read the bits in the target sector.
  - $T_{\text{avg\_transfer}} = 1/\text{RPM} \times 1/(\text{avg \# sectors/track}) \times 60 \text{ secs}/1 \text{ min}.$

# Logical Disk Blocks

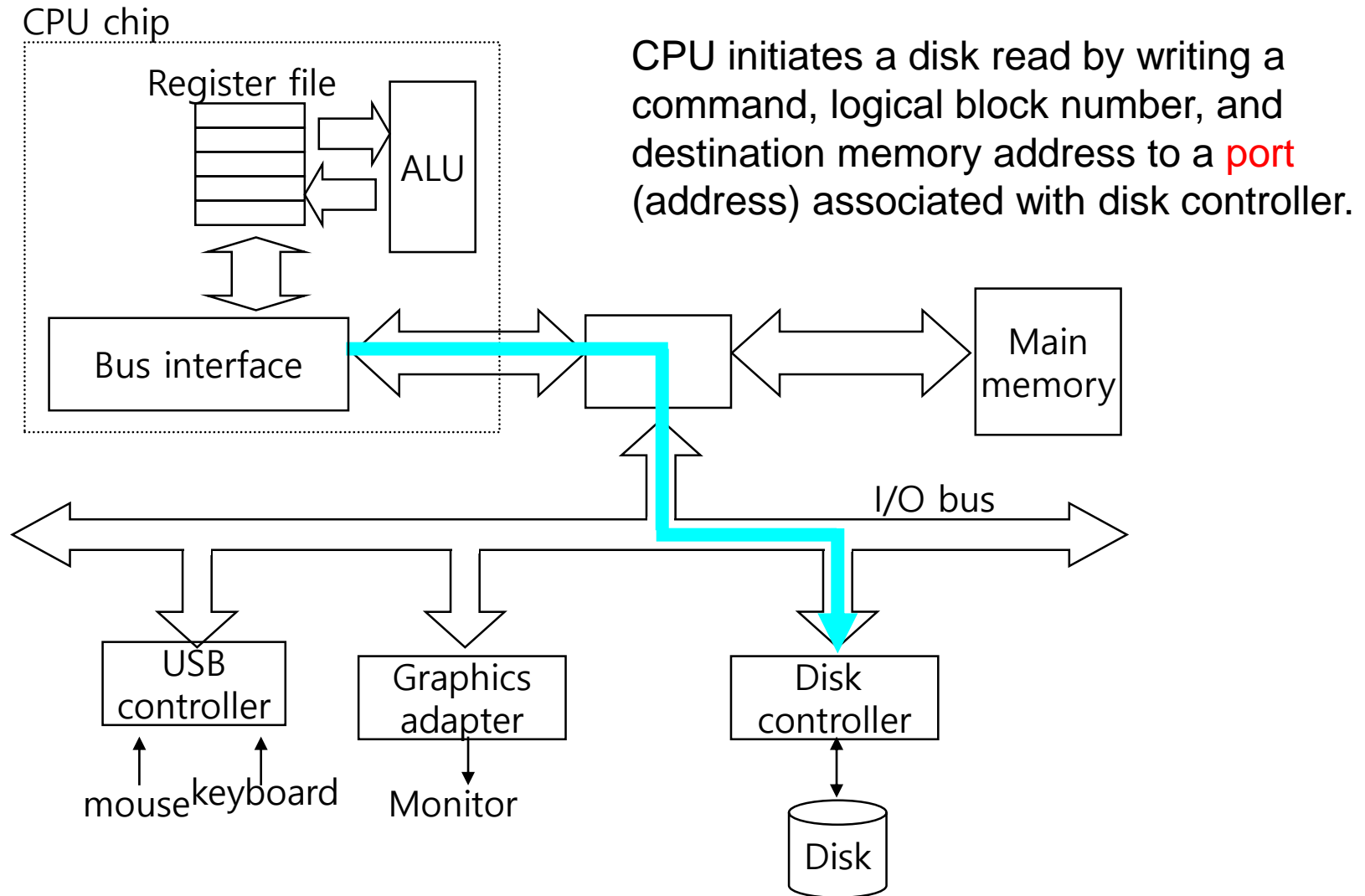
- Modern disks present a simpler abstract view of the complex sector geometry:
  - The set of available sectors is modeled as a sequence of b-sized **logical blocks** (0, 1, 2, ...)
- Mapping between logical blocks and actual (physical) sectors
  - Maintained by hardware/firmware device called disk controller.
  - Converts requests for logical blocks into (surface, track, sector) triples.
- Allows controller to set aside spare cylinders for each zone.



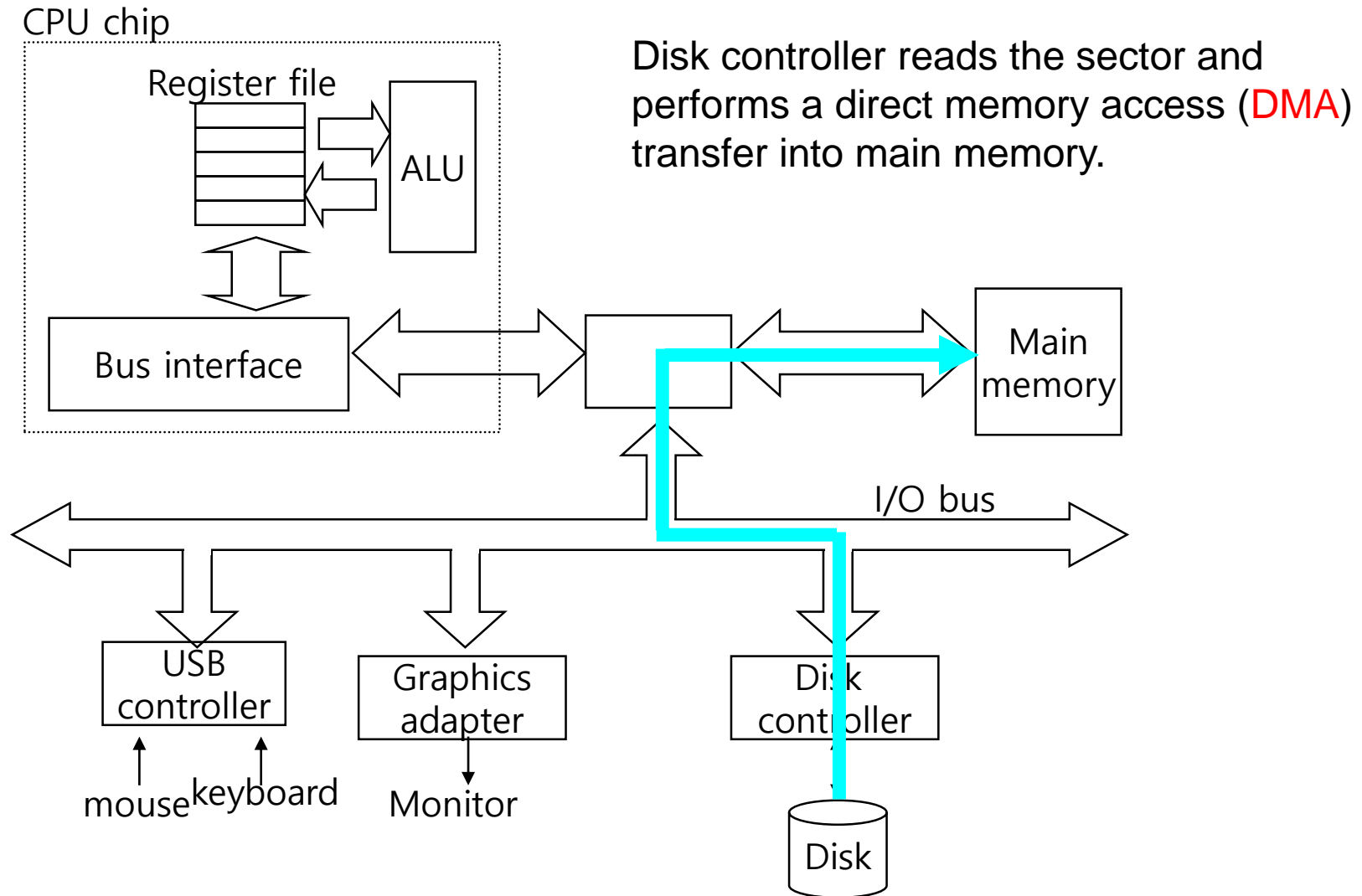
# I/O Bus



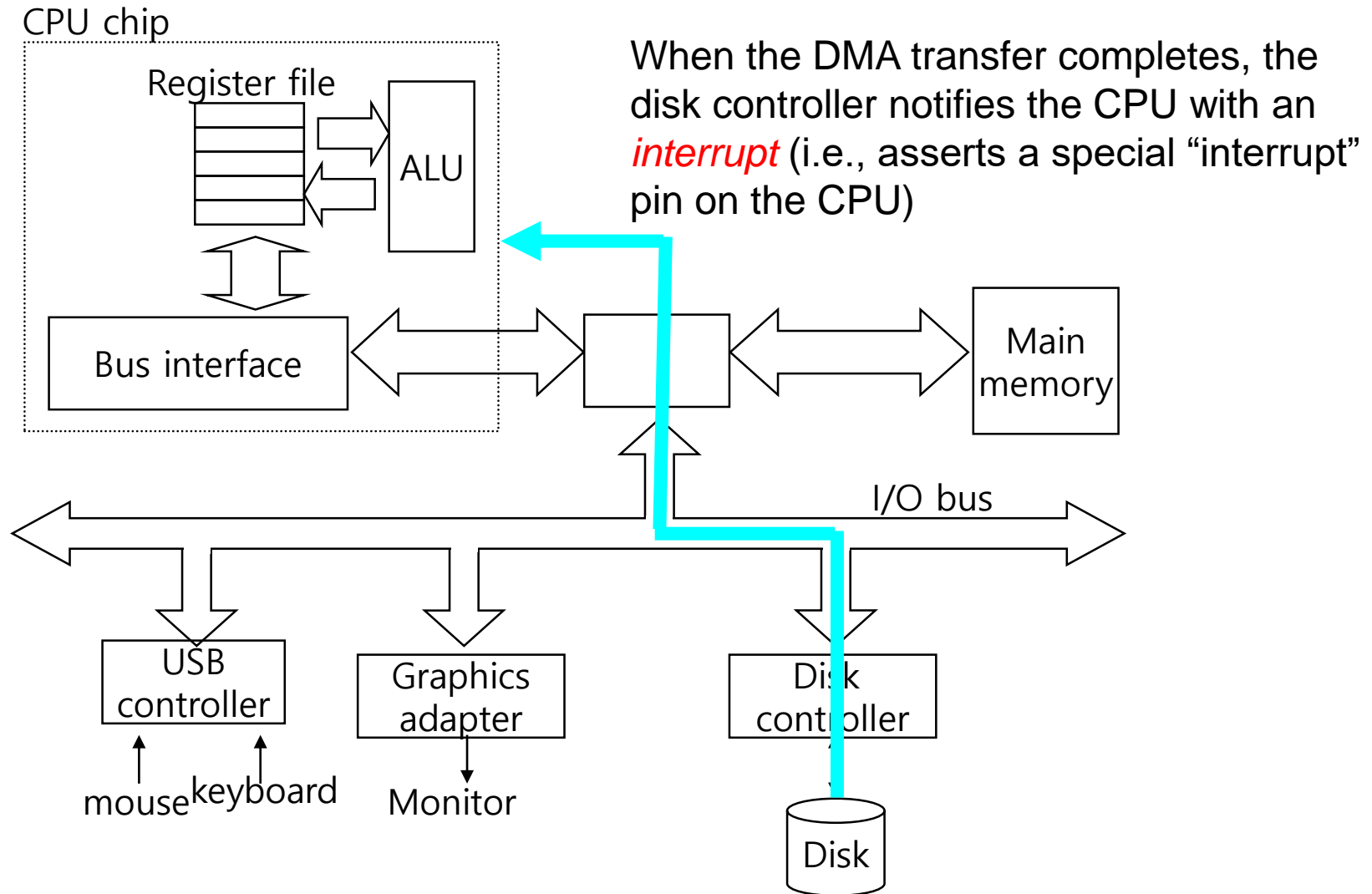
# Reading a Disk Sector (1)



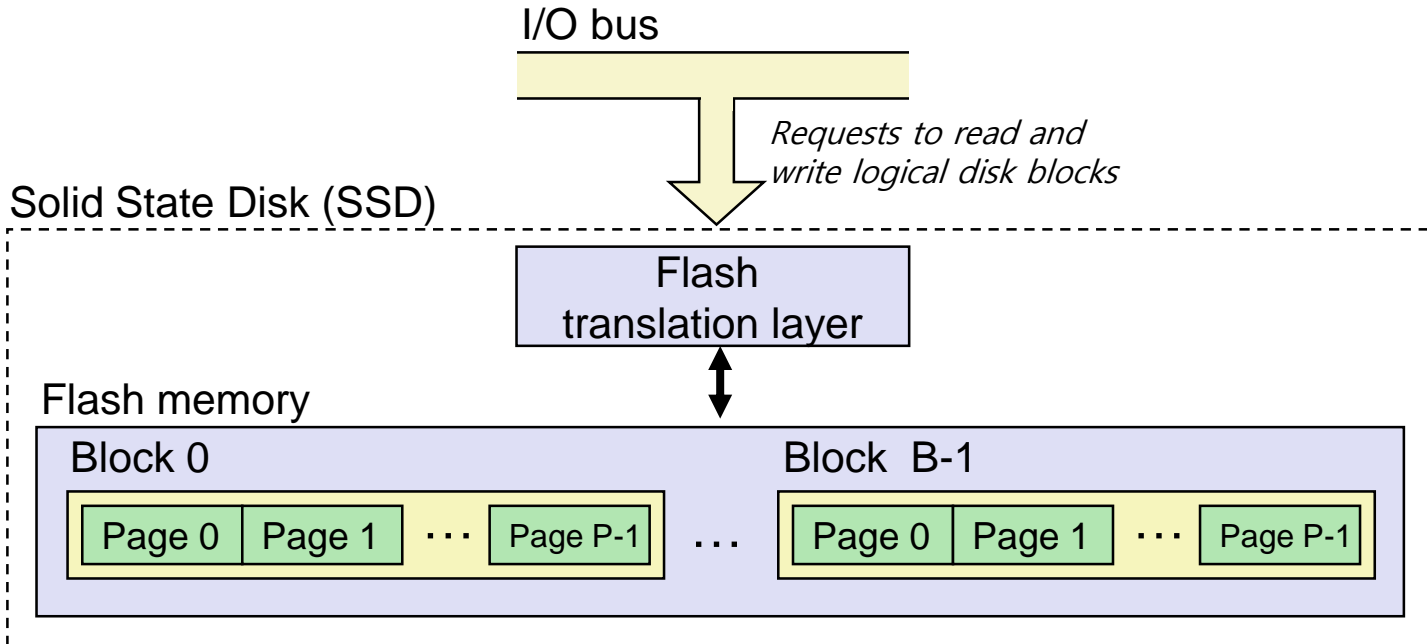
# Reading a Disk Sector (2)



# Reading a Disk Sector (3)



# Solid State Disks (SSDs)



- Pages: 512KB to 4KB, Blocks: 32 to 128 pages
- Data read/written in units of pages.
- Page can be written only after its block has been erased
- A block wears out after about 100,000 repeated writes.

# SSD Performance Characteristics

Sequential read tput	550 MB/s	Sequential write tput	470 MB/s
Random read tput	365 MB/s	Random write tput	303 MB/s
Avg seq read time	50 us	Avg seq write time	60 us

- Sequential access faster than random access
  - Common theme in the memory hierarchy
- Random writes are somewhat slower
  - Erasing a block takes a long time (~1 ms)
  - Modifying a block page requires all other pages to be copied to new block
  - In earlier SSDs, the read/write gap was much larger.

Source: Intel SSD 730 product specification.

# SSD Tradeoffs vs Rotating Disks

- Advantages
  - No moving parts → faster, less power, more rugged
- Disadvantages
  - Have the potential to wear out
    - Mitigated by “wear leveling logic” in flash translation layer
    - E.g. Intel SSD 730 guarantees 128 petabyte ( $128 \times 10^{15}$  bytes) of writes before they wear out
  - In 2015, about 30 times more expensive per byte
- Applications
  - MP3 players, smart phones, laptops
  - Beginning to appear in desktops and servers

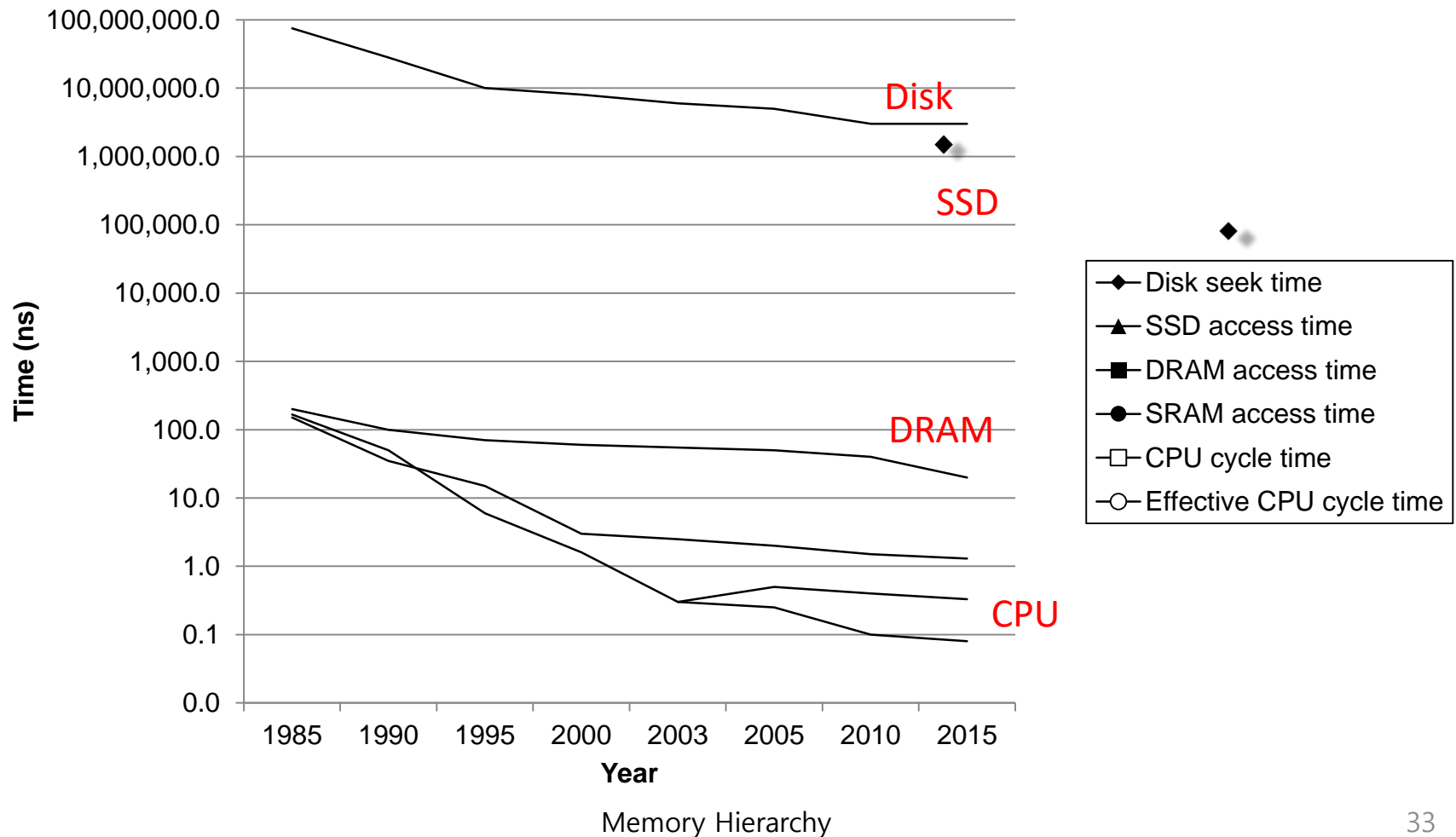
# Outline

- Storage technologies and trends
- Locality of reference
- Caching in the memory hierarchy



# The CPU-Memory Gap

The gap widens between DRAM, disk, and CPU speeds.



# Storage Trends

## SRAM

Metric	1985	1990	1995	2000	2005	2010	2015	<i>2015:1985</i>
\$/MB	2,900	320	256	100	75	60	<i>25</i>	<i>116</i>
access (ns)	150	35	15	3	2	1.5	<i>1.3</i>	<i>115</i>

## DRAM


Metric	1985	1990	1995	2000	2005	2010	2015	<i>2015:1985</i>
\$/MB	880	100	30	1	0.1	0.06	0.02	<i>44,000</i>
access (ns)	200	100	70	60	50	40	20	<i>10</i>
typical size (MB)	0.256	4	16	64	2,000	8,000	16,000	<i>62,500</i>

## Disk

Metric	1985	1990	1995	2000	2005	2010	2015	<i>2015:1985</i>
\$/GB	100,000	8,000	300	10	5	0.3	0.03	<i>3,333,333</i>
access (ms)	75	28	10	8	<i>5</i>	<i>3</i>	<i>3</i>	<i>25</i>
typical size (GB)	0.01	0.16	1	20	160	1,500	3,000	<i>300,000</i>

# CPU Clock Rates

Inflection point in computer history  
when designers hit the “Power Wall”



	1985	1990	1995	2003	2005	2010	2015	<i>2015:1985</i>
CPU	80286	80386	Pentium	P-4	Core 2	Core i7(n)	Core i7(h)	
Clock rate (MHz)	6	20	150	3,300	2,000	2,500	3,000	500
Cycle time (ns)	166	50	6	0.30	0.50	0.4	0.33	500
Cores	1	1	1	1	2	4	4	4
Effective cycle time (ns)	166	50	6	0.30	0.25	0.10	0.08	2,075

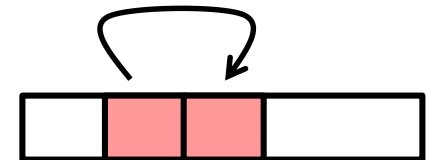
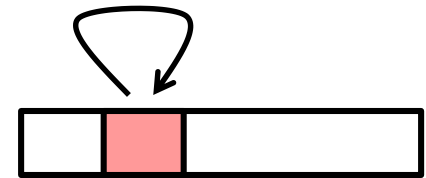
(n) Nehalem processor  
(h) Haswell processor

# Locality to the Rescue!

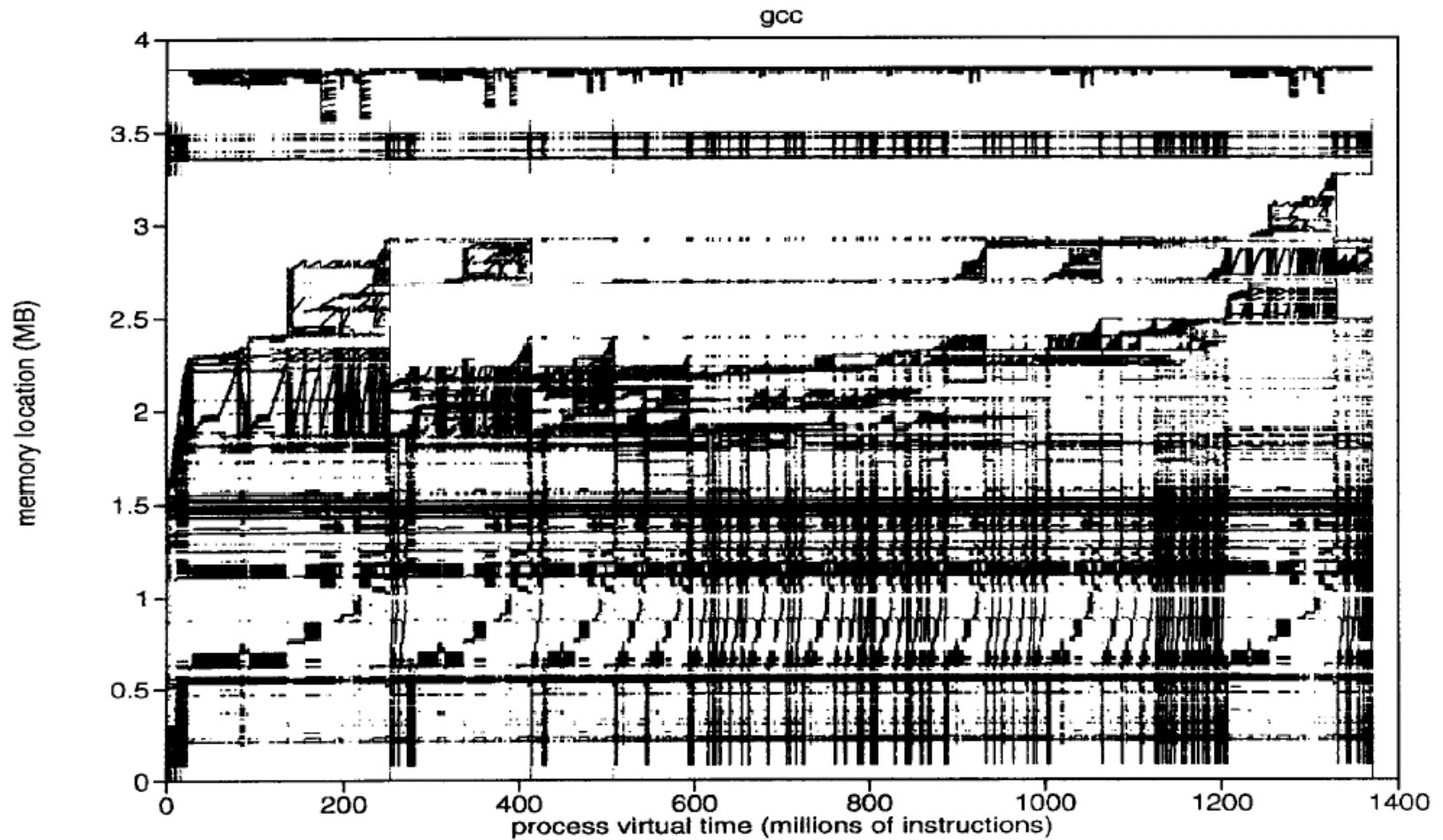
The key to bridging this CPU-Memory gap is a fundamental property of computer programs known as **locality**

# Locality

- **Principle of Locality:** Programs tend to use data and instructions with addresses near or equal to those they have used recently
- **Temporal locality:**
  - Recently referenced items are likely to be referenced again in the near future
- **Spatial locality:**
  - Items with nearby addresses tend to be referenced close together in time



# Locality



Source: Glass & Cao (1997 ACM SIGMETRICS)

# Locality Example

```
sum = 0;  
for (i = 0; i < n; i++)  
    sum += a[i];  
return sum;
```

- Data references
  - Reference array elements in succession (stride-1 reference pattern). Spatial locality
  - Reference variable `sum` each iteration. Temporal locality
- Instruction references
  - Reference instructions in sequence. Spatial locality
  - Cycle through loop repeatedly. Temporal locality

# Outline

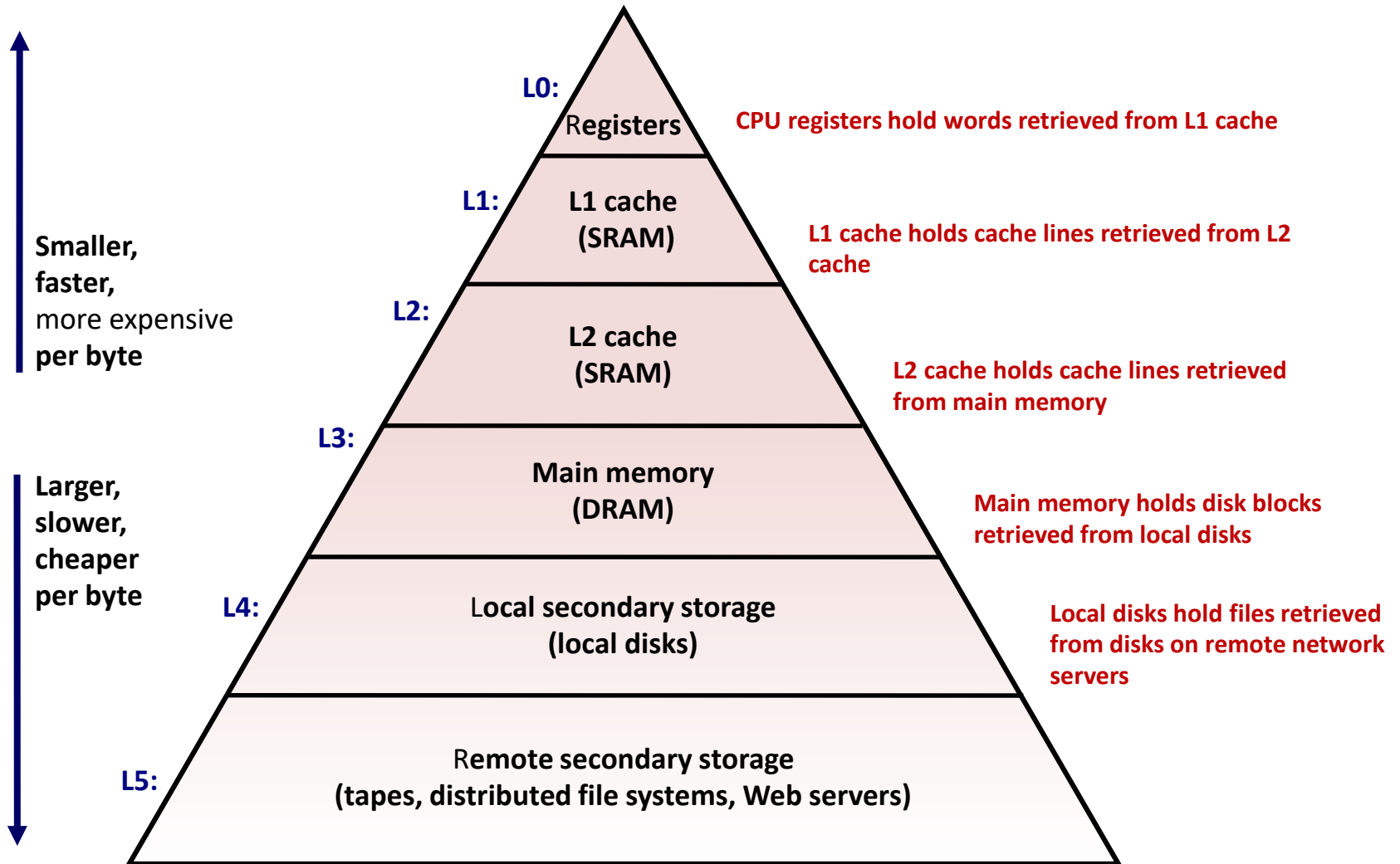
- Storage technologies and trends
- Locality of reference
- Caching in the memory hierarchy



# Memory Hierarchies

- Some fundamental and enduring properties of hardware and software:
  - Fast storage technologies cost more per byte, have less capacity, and require more power (heat!).
  - The gap between CPU and main memory speed is widening.
  - Well-written programs tend to exhibit good locality.
- These fundamental properties complement each other beautifully.
- They suggest an approach for organizing memory and storage systems known as a **memory hierarchy**.

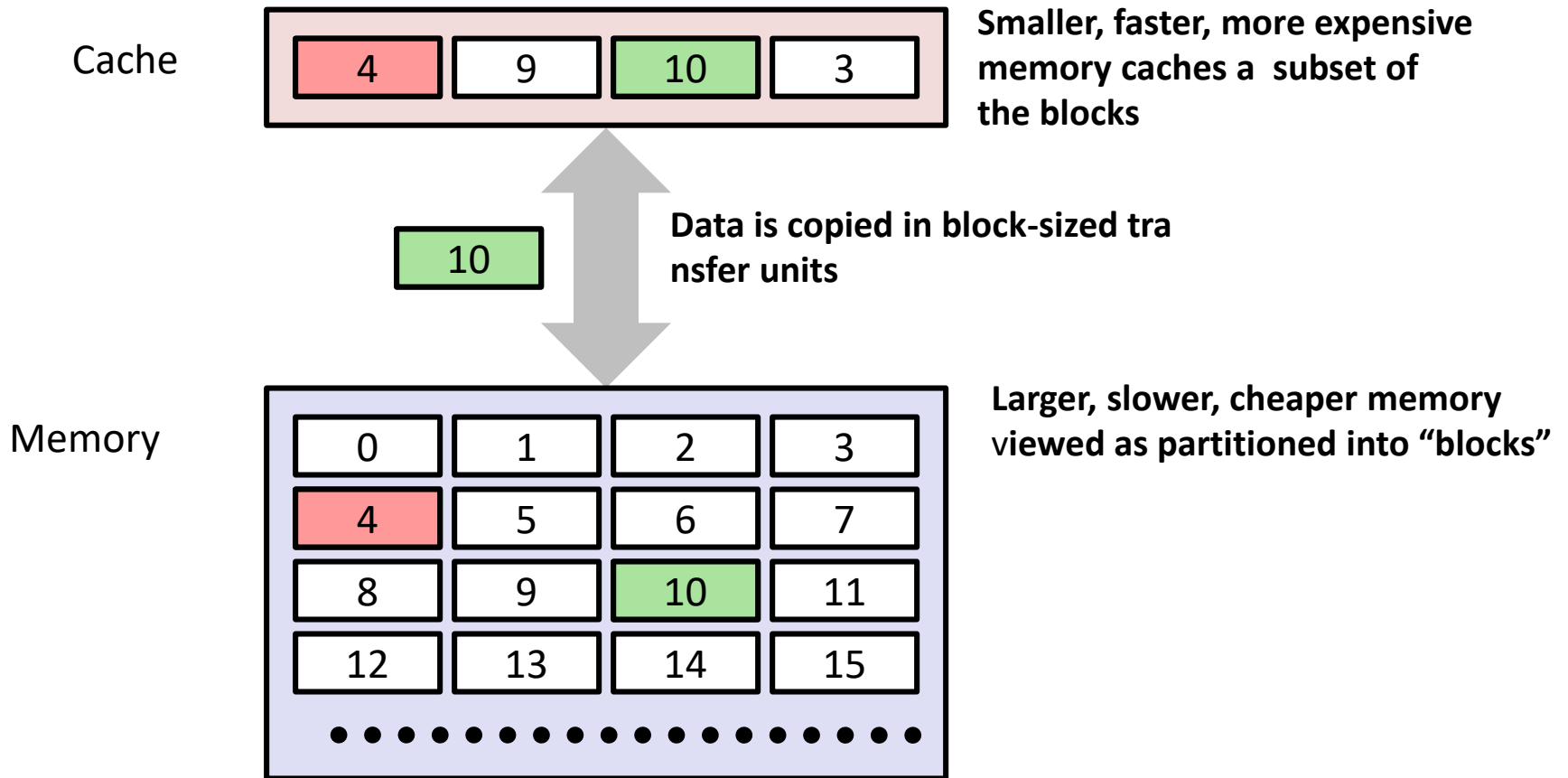
# An Example Memory Hierarchy



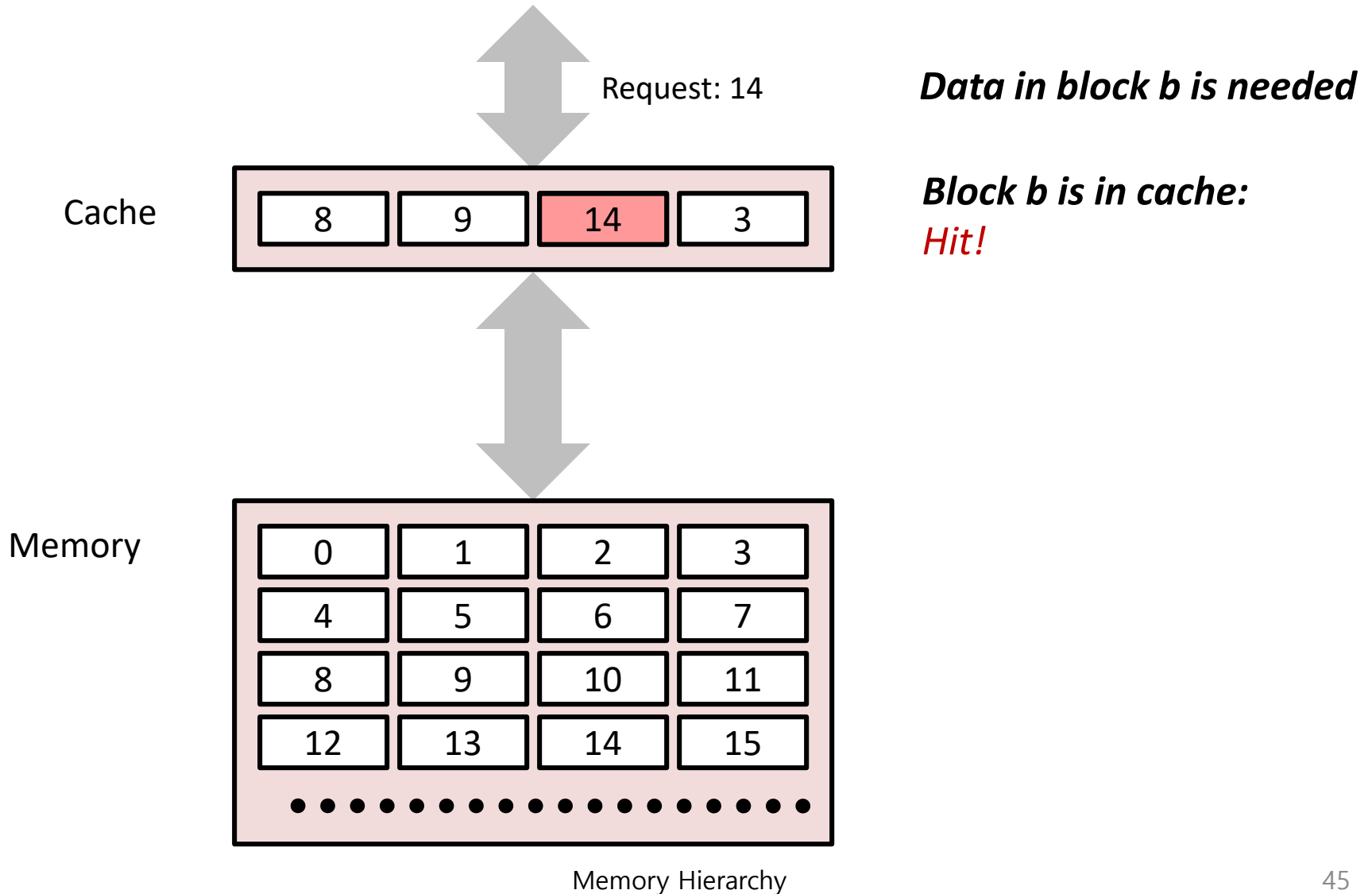
# Caches

- *Cache*: A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.
- An optimization resulting from a perfect match between memory technology and two types of program locality
  - Temporal locality (locality in time)
    - If an item is referenced, it will tend to be referenced again soon.
  - Spatial locality (locality in space)
    - If an item is referenced, items whose addresses are close by will tend to be referenced soon..
- *Big Idea*: To provide a “virtual” memory technology (an illusion) that has an access time of the highest-level memory with the size and cost of the lowest-level memory

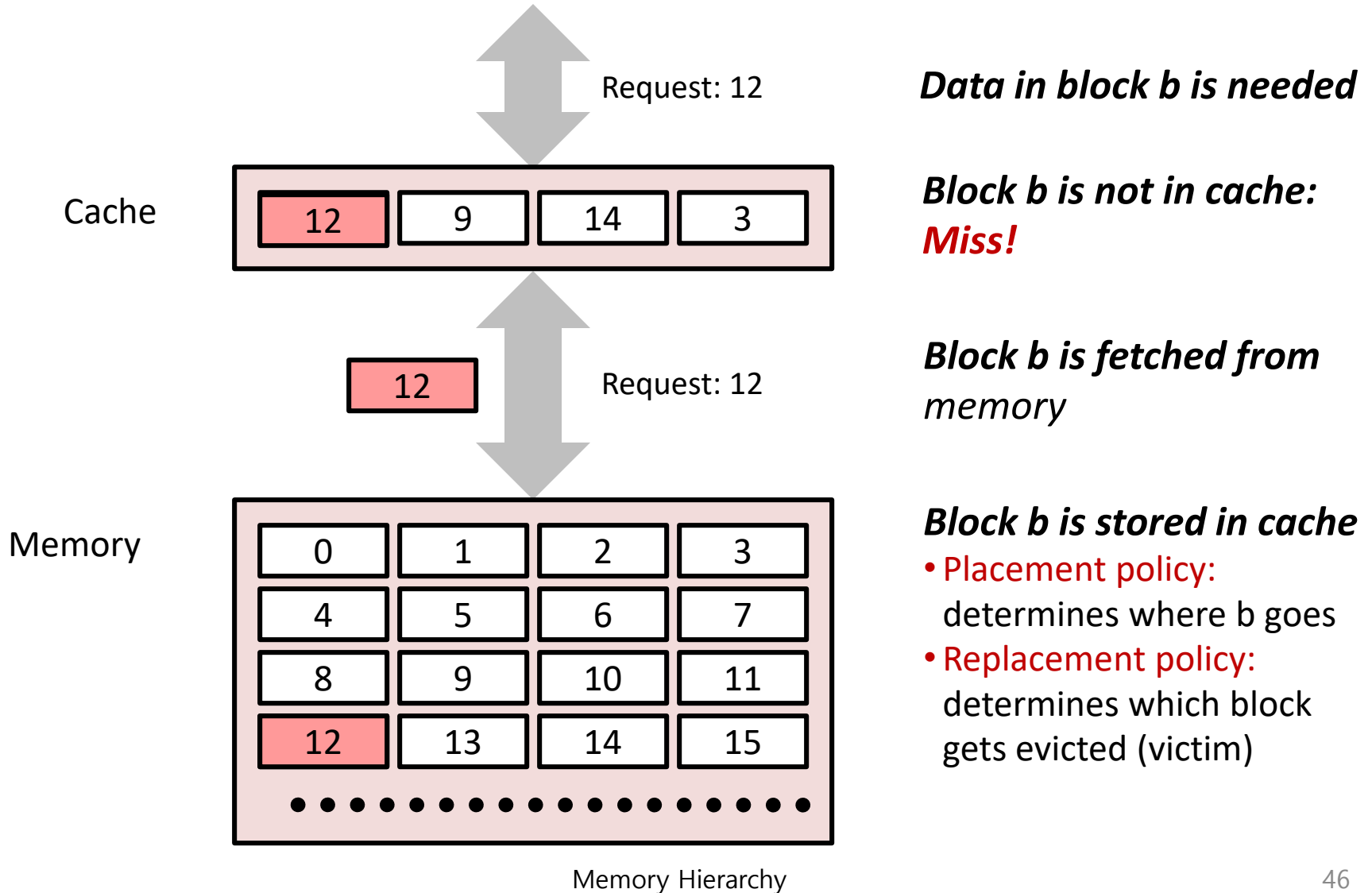
# General Cache Concepts



# General Cache Concepts: Hit



# General Cache Concepts: Miss



# General Caching Concepts:

## Types of Cache Misses

- **Cold (compulsory) miss**
  - Cold misses occur when a block is accessed for the first time.
- **Conflict miss**
  - Most caches limit blocks at level  $k+1$  to a small subset of blocks (sometimes a single block) at level  $k$ .
    - e.g., Block  $i$  at level  $k+1$  must be placed in block  $(i \bmod 4)$  at level  $k$ .
  - Conflict misses occur even when the level  $k$  cache is large enough if multiple data objects all map to the same level  $k$  block.
    - e.g., Referencing blocks 0, 8, 0, 8, 0, 8, ... would miss every time.
- **Capacity miss**
  - Occurs when the set of active cache blocks (**working set**) is larger than the cache.

# Examples of Caching in the Hierarchy

Cache Type	What is Cached?	Where is it Cached?	Latency (cycles)	Managed By
Registers	4-8 bytes words	CPU core	0	Compiler
TLB	Address translations	On-Chip TLB	0	Hardware
L1 cache	64-bytes block	On-Chip L1	1	Hardware
L2 cache	64-bytes block	On/Off-Chip L2	10	Hardware
Virtual Memory	4-KB page	Main memory	100	Hardware + OS
Buffer cache	Parts of files	Main memory	100	OS
Disk cache	Disk sectors	Disk controller	100,000	Disk firmware
Network buffer cache	Parts of files	Local disk	10,000,000	AFS/NFS client
Browser cache	Web pages	Local disk	10,000,000	Web browser
Web cache	Web pages	Remote server disks	1,000,000,000	Web proxy server



# Summary

- The speed gap between CPU, memory and mass storage continues to widen.
- Well-written programs exhibit a property called locality.
- Memory hierarchies based on caching close the gap by exploiting locality.

# Questions?