

# More on Memory Hierarchy



Professor Hugh C. Lauer

CS-2011, Machine Organization and Assembly Language

(Slides include copyright materials from *Computer Systems: A Programmer's Perspective*, by Bryant and O'Hallaron, and from *The C Programming Language*, by Kernighan and Ritchie)

# Today

Reading Assignment: §6.1 – §6.5

- Storage technologies and trends
- Locality of reference  First
- Caching in the memory hierarchy  Preparation for Cachelab

# 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

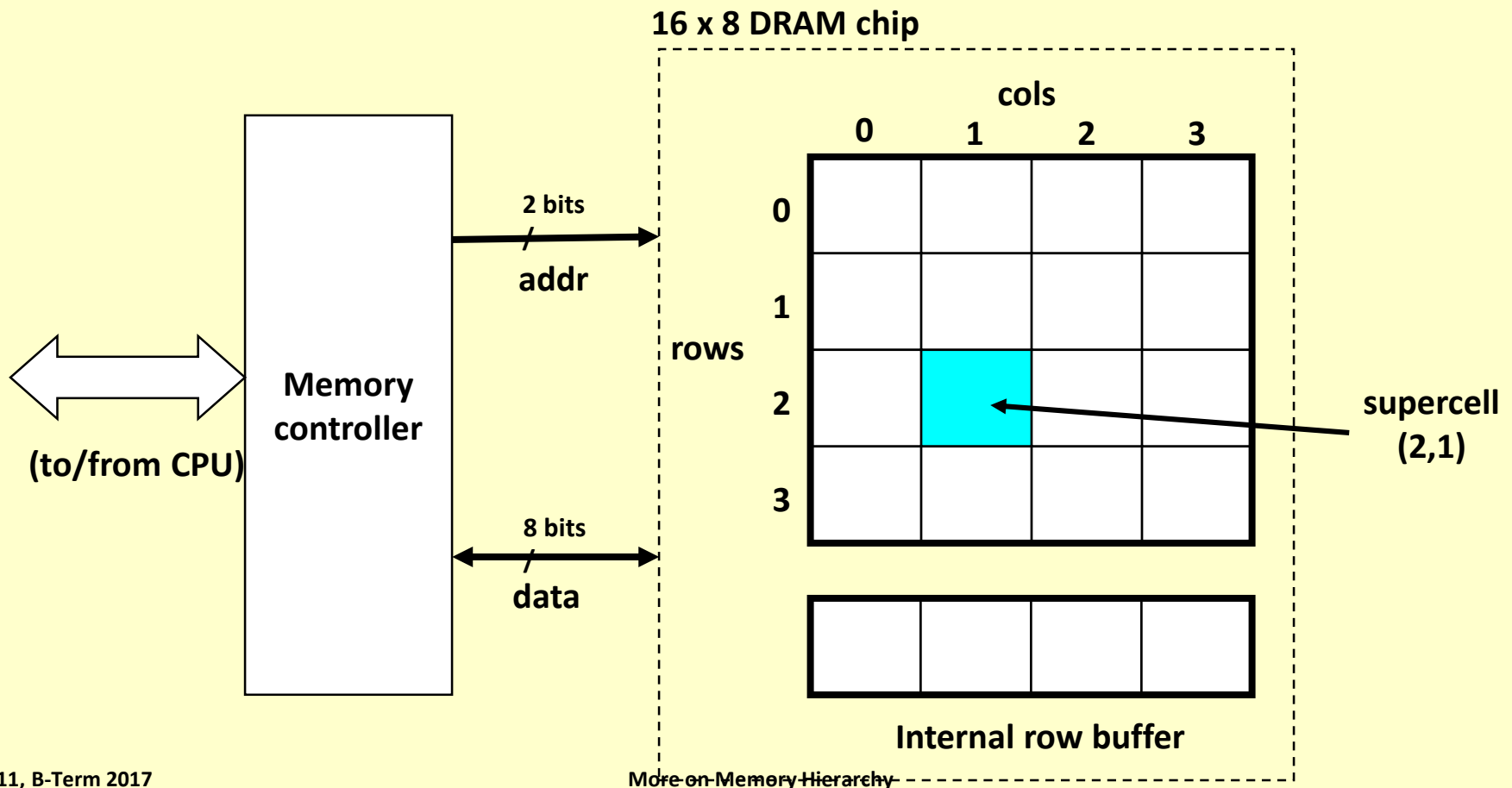
	Trans. per bit	Access time	Needs refresh?	Needs EDC*?	Cost	Applications
SRAM	4 or 6	1X	No	Maybe	100x	Cache memories
DRAM	1	10X	Yes	Yes	1X	Main memories, frame buffers

**EDC = Error Detection and Correction**

# Conventional DRAM Organization

## ■ $d \times w$ DRAM:

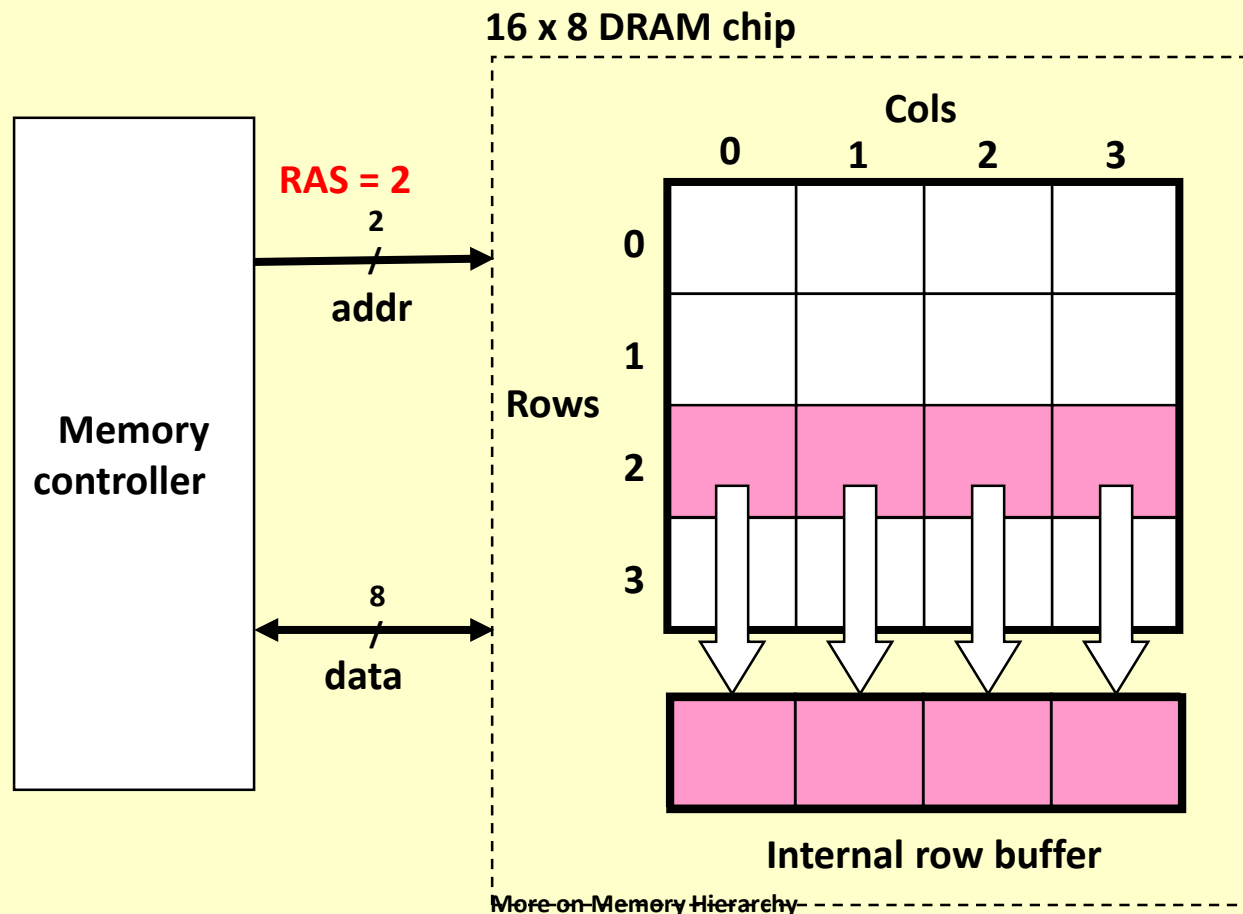
- $d \times w$  total bits organized as  $d$  **supercells** of size  $w$  bits



# Reading DRAM Supercell (2,1)

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

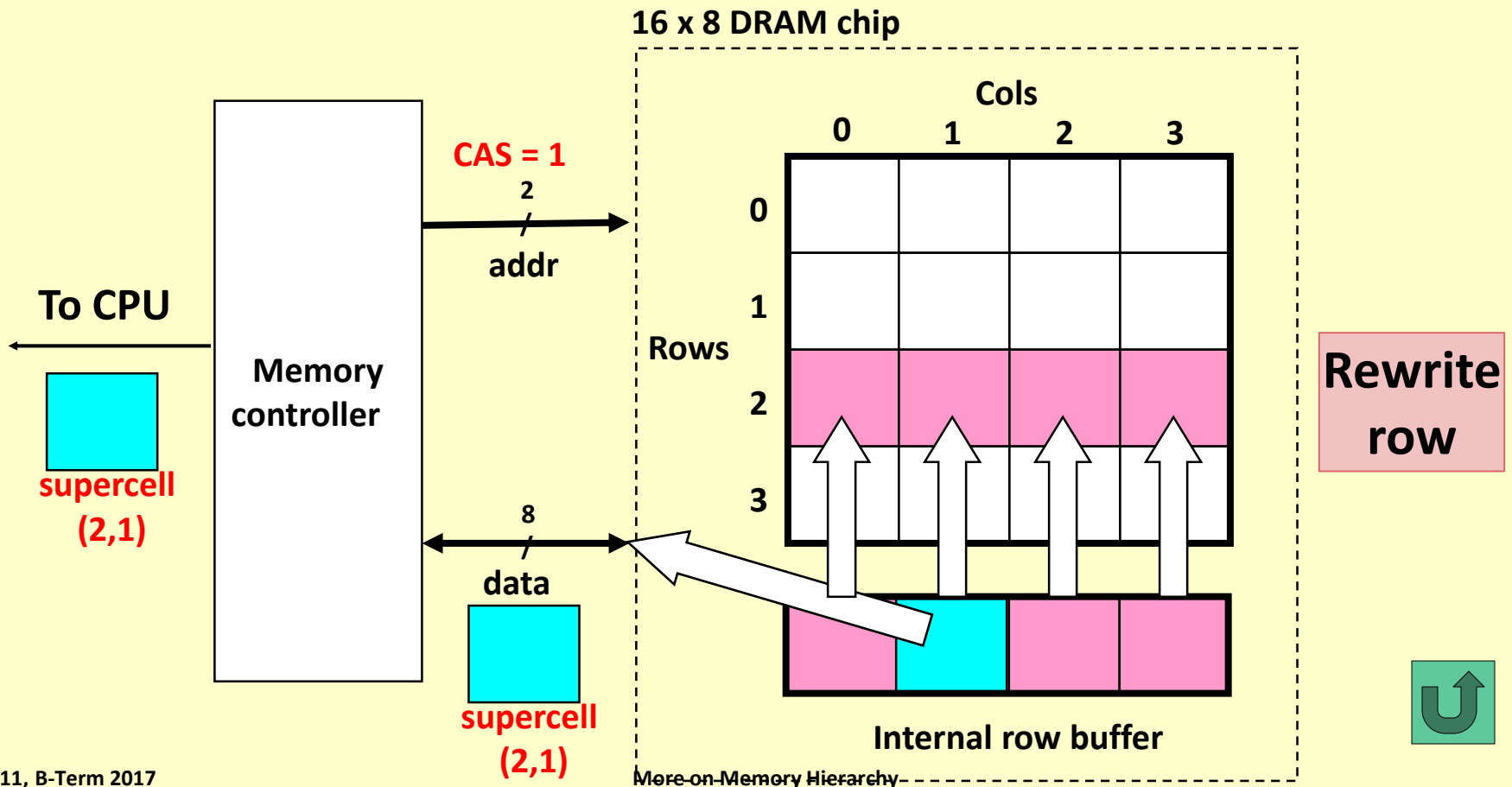
Step 1(b): Row 2 copied from DRAM array to row buffer.



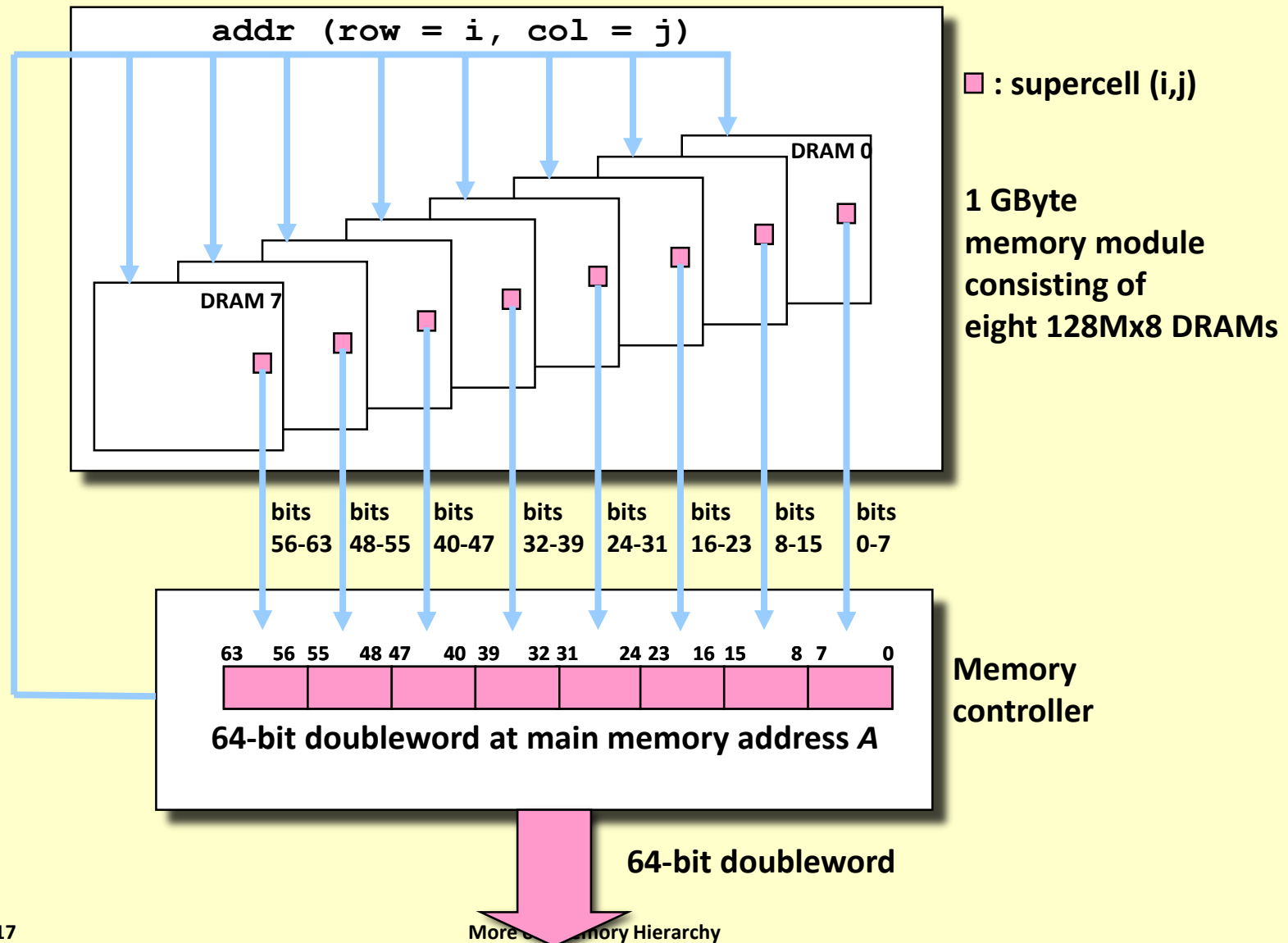
# Reading DRAM Supercell (2,1)

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 invention in 1966.**
  - Commercialized by Intel in 1970
  - (except for tinier cells, more bits per sq millimeter)
- **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
    - Different types distinguished by size of small prefetch buffer:
      - **DDR** (2 bits), **DDR2** (4 bits), **DDR3** (8 bits)
    - By 2010, standard for most server and desktop systems
    - Intel Core i7 supports only DDR3 SDRAM



# Nonvolatile Memories

## ■ DRAM and SRAM are volatile memories

- Lose information if powered off.

## ■ Nonvolatile memories retain value even if powered off

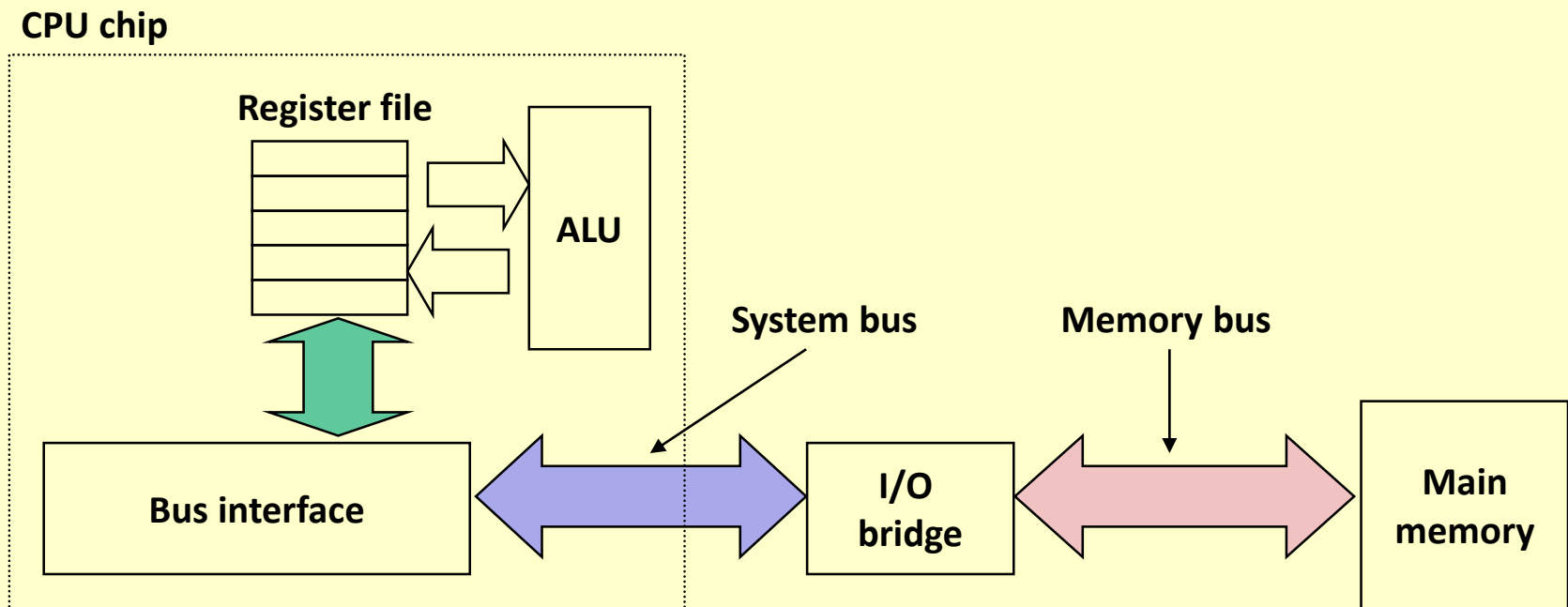
- Read-only memory (**ROM**): programmed during production
- Programmable ROM (**PROM**): can be programmed once
- Erasable PROM (**EPROM**): can be bulk erased (UV, X-Ray)
- Electrically erasable PROM (**EEPROM**): electronic erase capability
- Flash memory: EEPROMs with partial (sector) erase capability
  - Wears out after about 100,000 erasings.

## ■ 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

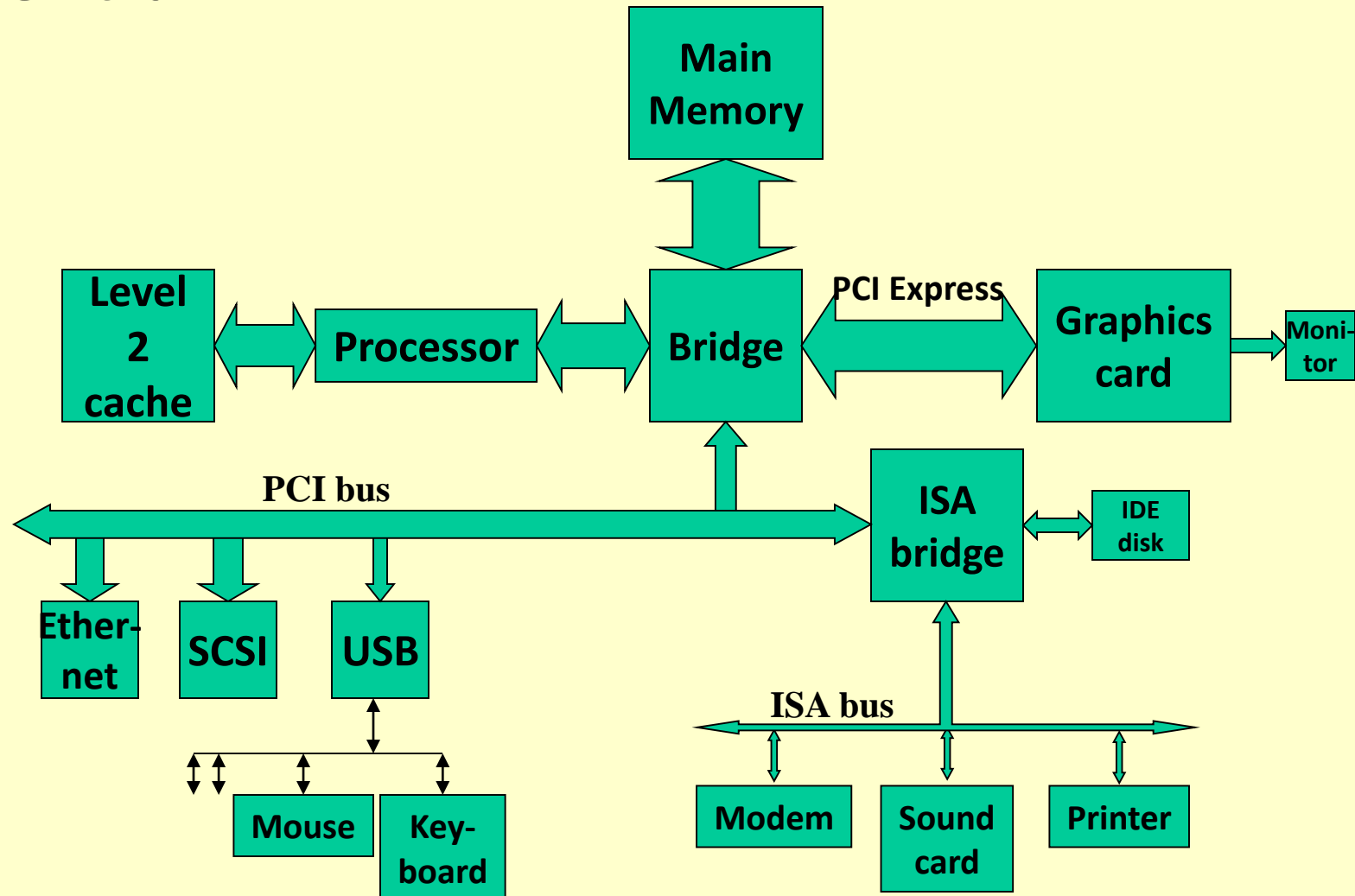
# Traditional Bus Structure Connecting CPU and Memory

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



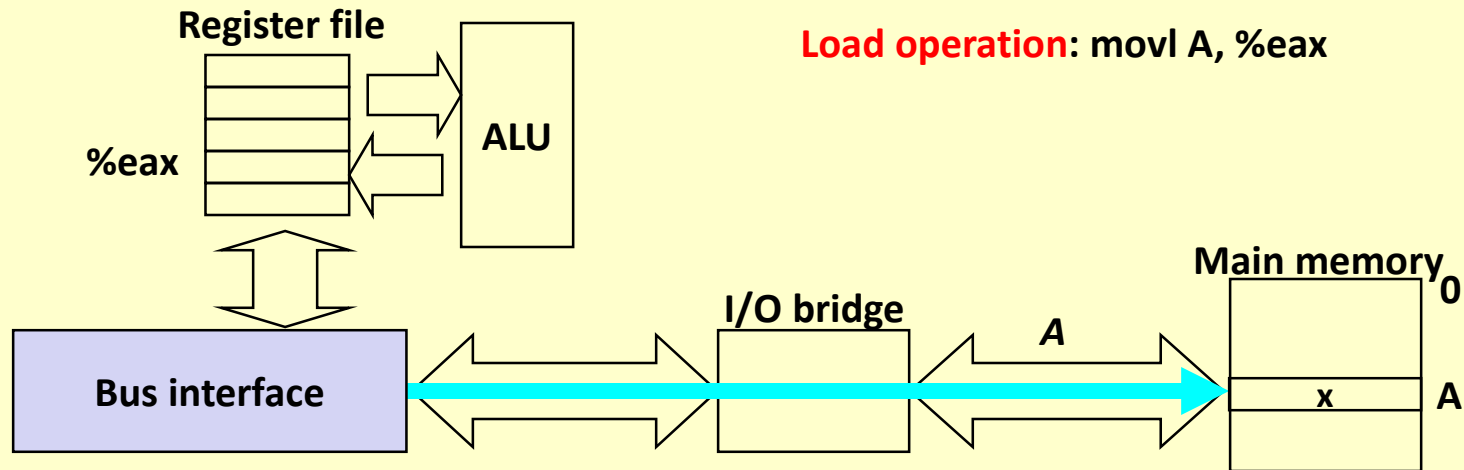
# Hardware Organization — 2005 era

## Pentium



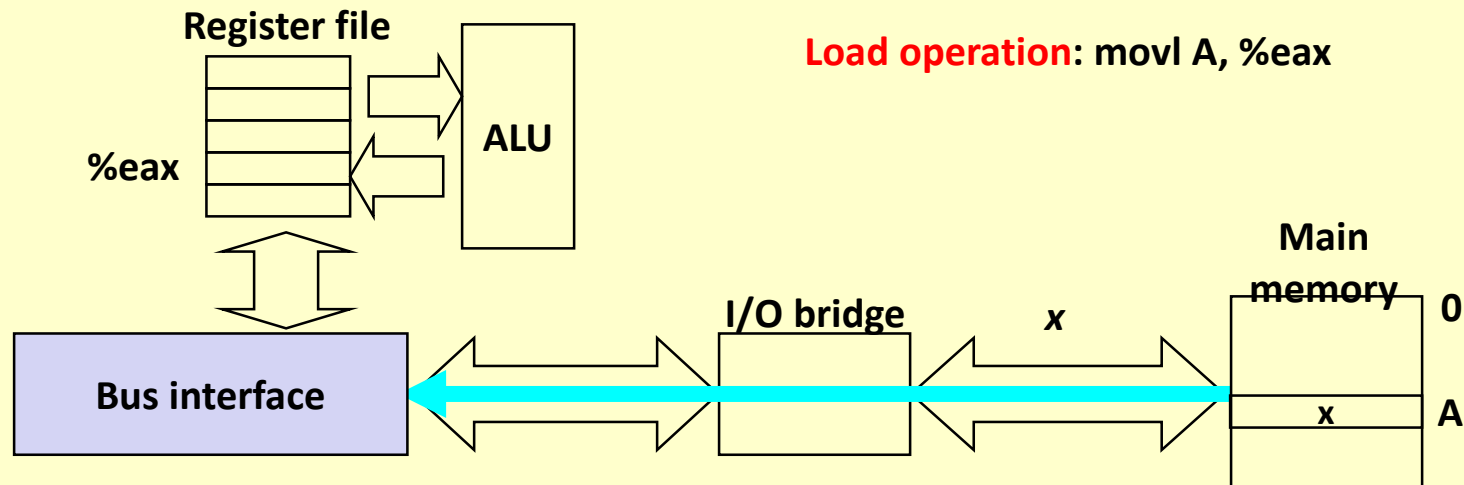
# Memory Read Transaction (1)

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



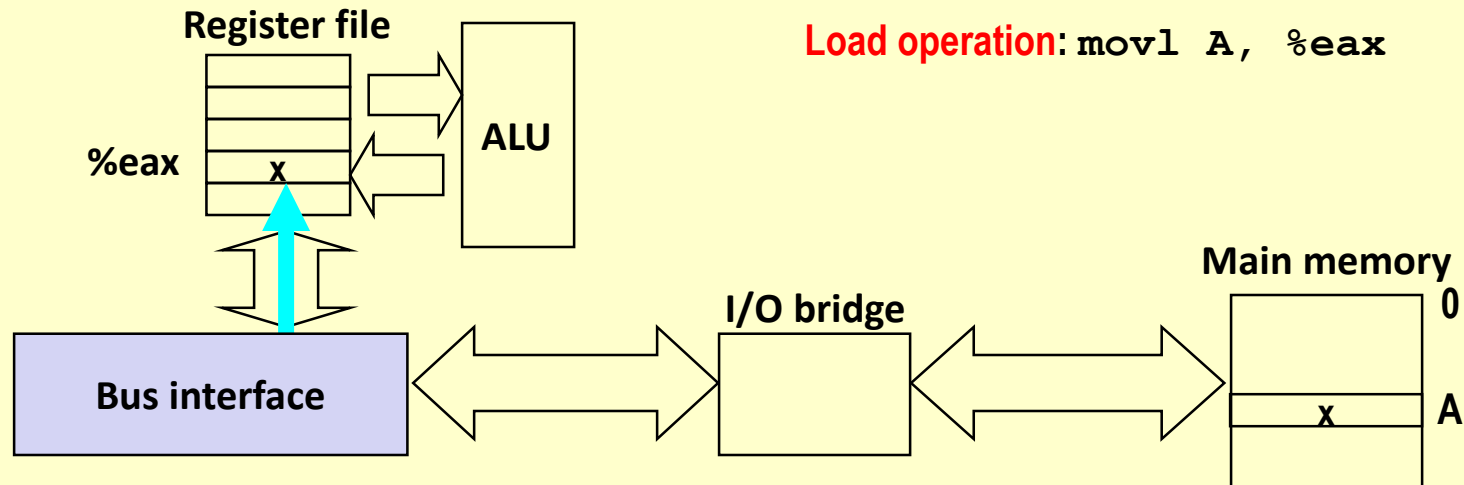
# Memory Read Transaction (2)

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



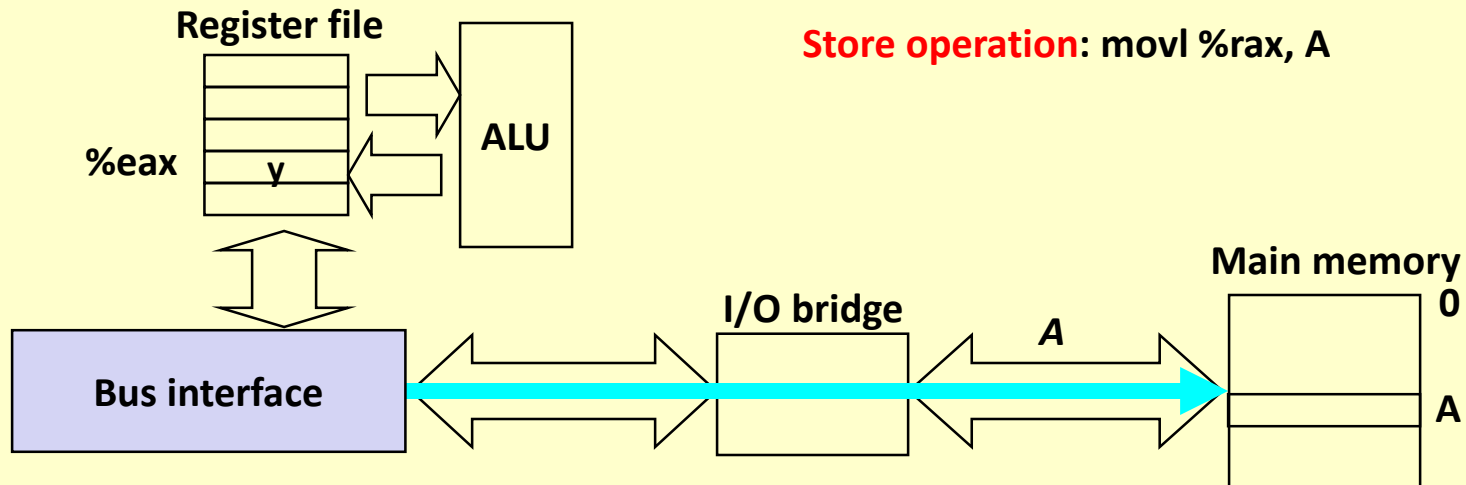
# Memory Read Transaction (3)

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



# Memory Write Transaction (1)

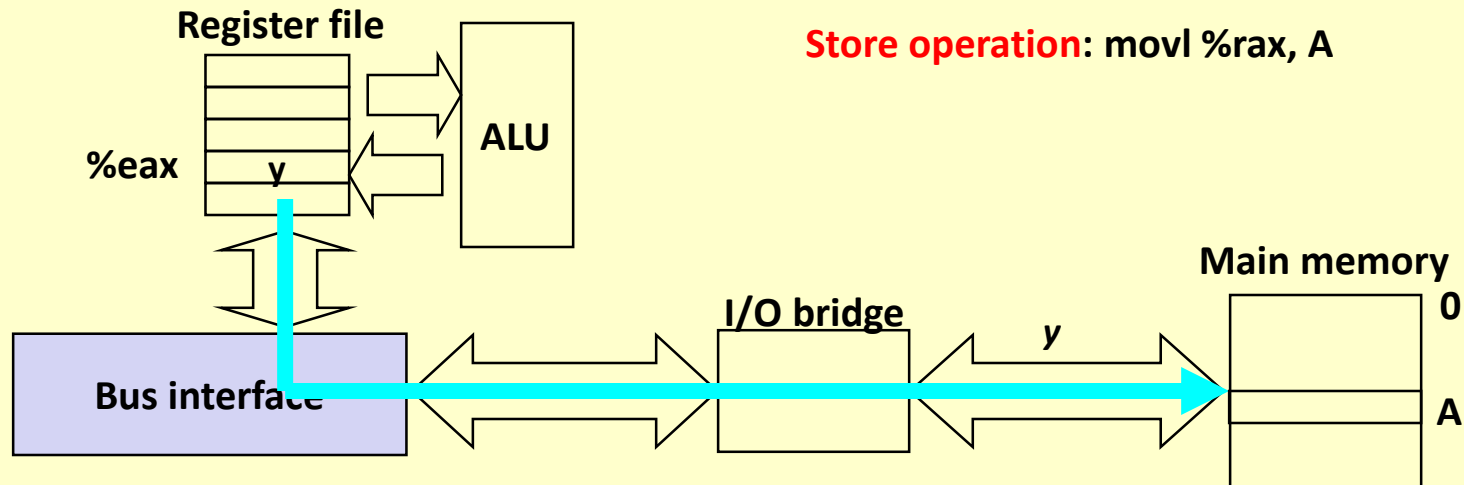
- CPU places address A on bus. Main memory reads it and waits for the corresponding data word to arrive.





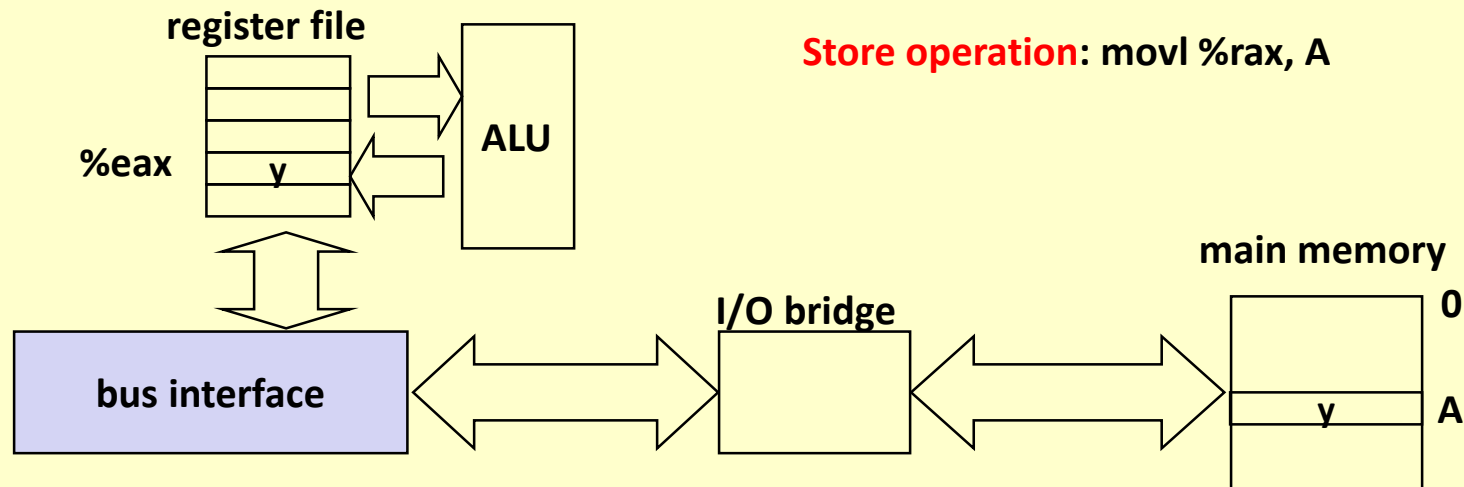
# Memory Write Transaction (2)

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



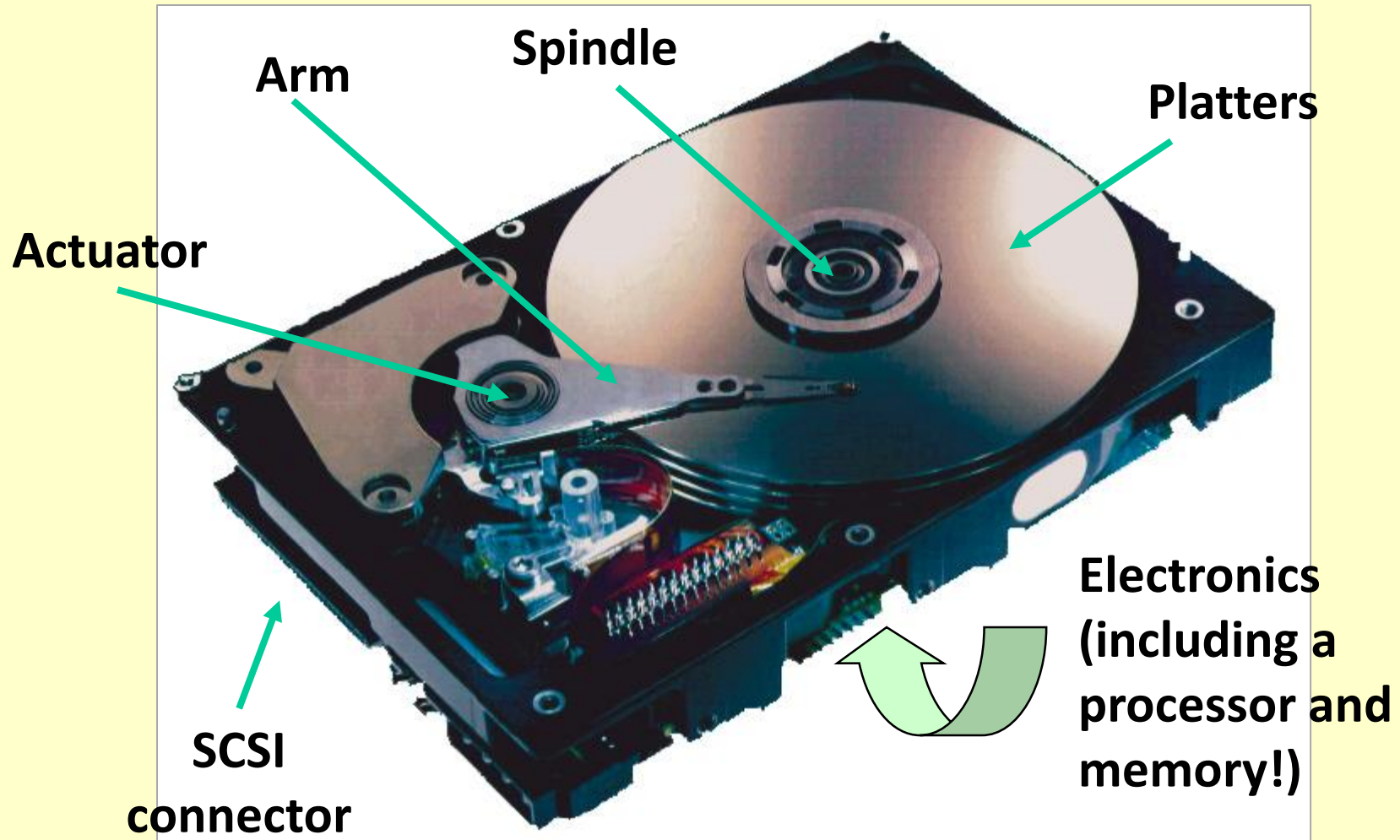
# Memory Write Transaction (3)

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



# Questions?

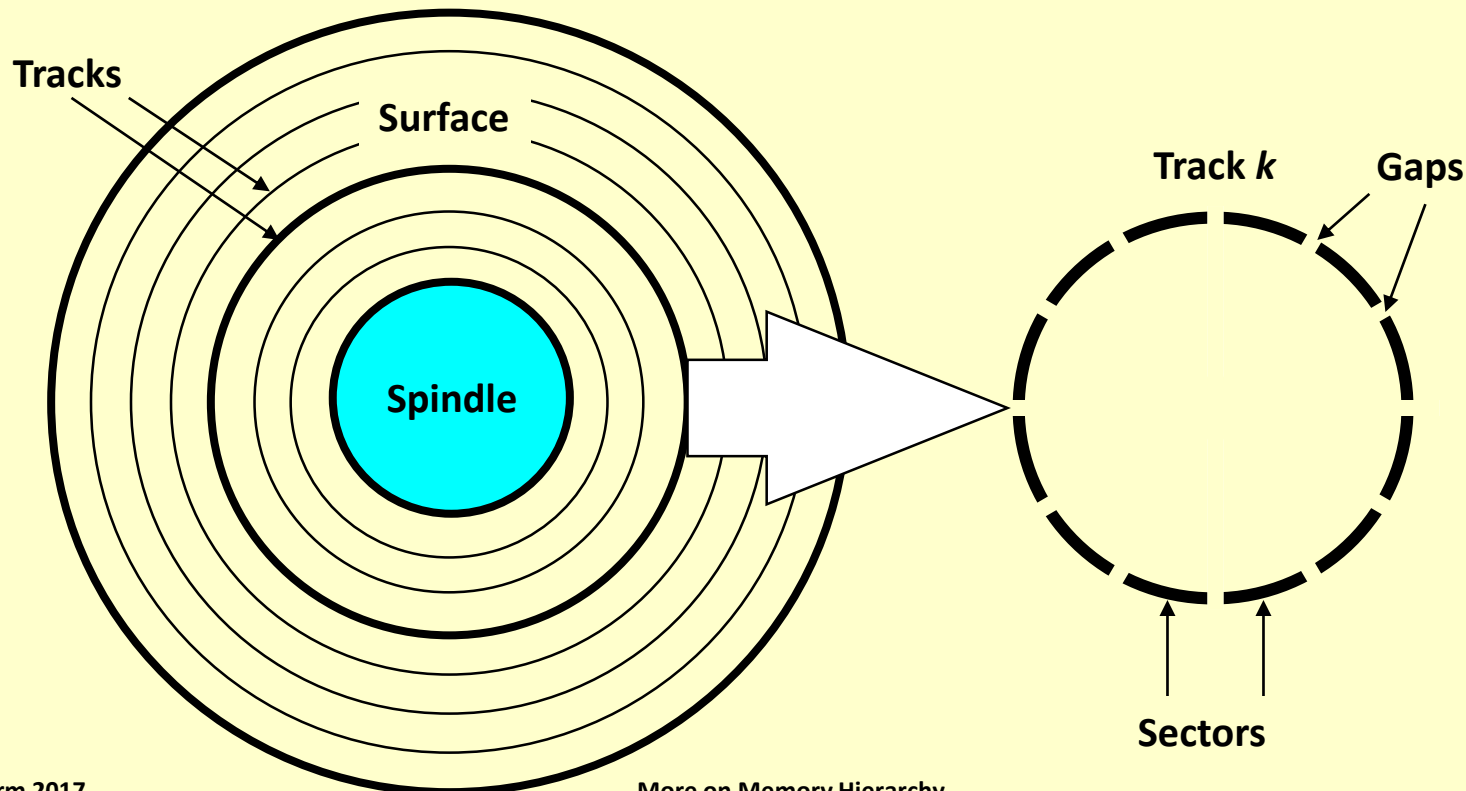
# What's Inside A Disk Drive?



*Image courtesy of Seagate Technology*

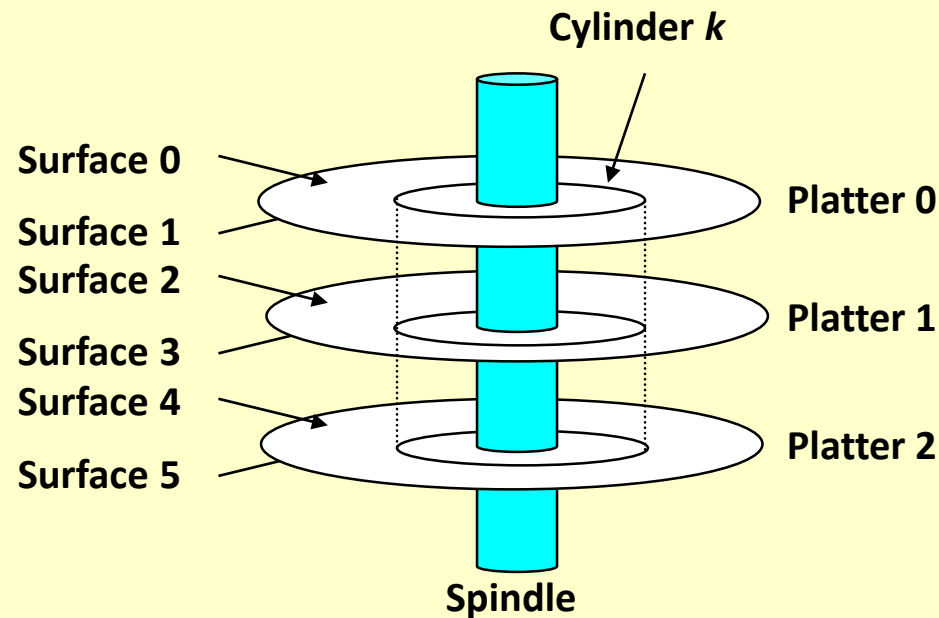
# Disk Geometry

- 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.

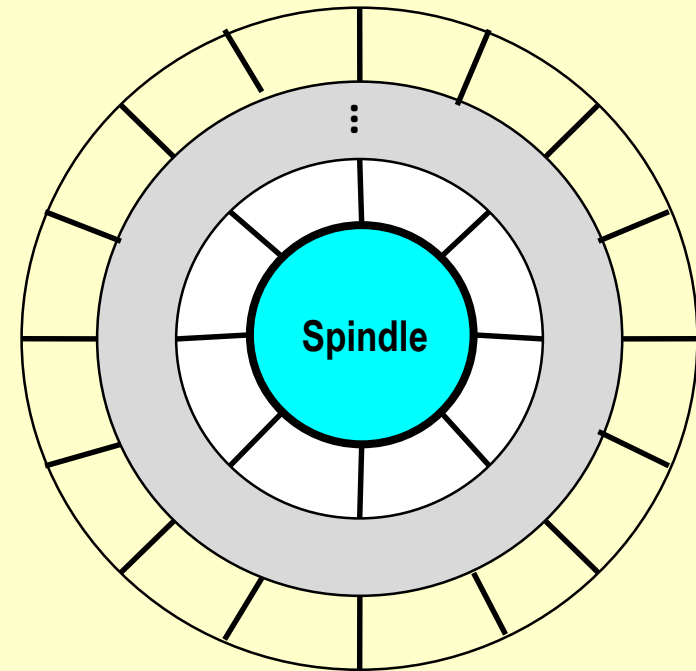


# Disk Capacity

- **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.
- **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

# 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.





# Computing Disk Capacity

$$\text{Capacity} = (\# \text{ bytes/sector}) \times (\text{avg. } \# \text{ sectors/track}) \times \\ (\# \text{ tracks/surface}) \times (\# \text{ surfaces/platter}) \times \\ (\# \text{ platters/disk})$$

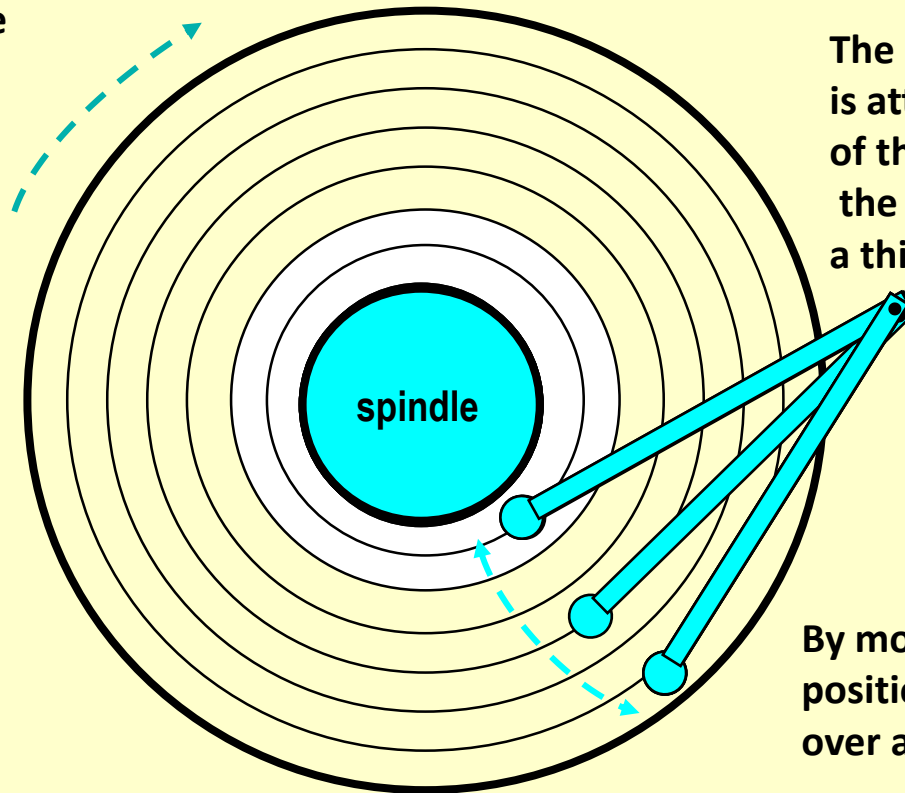
**Example:**

- 512 bytes/sector
- 300 sectors/track (on average)
- 20,000 tracks/surface
- 2 surfaces/platter
- 5 platters/disk

$$\begin{aligned} \text{Capacity} &= 512 \times 300 \times 20000 \times 2 \times 5 \\ &= 30,720,000,000 \\ &= 30.72 \text{ GB} \end{aligned}$$

# Disk Operation (Single-Platter View)

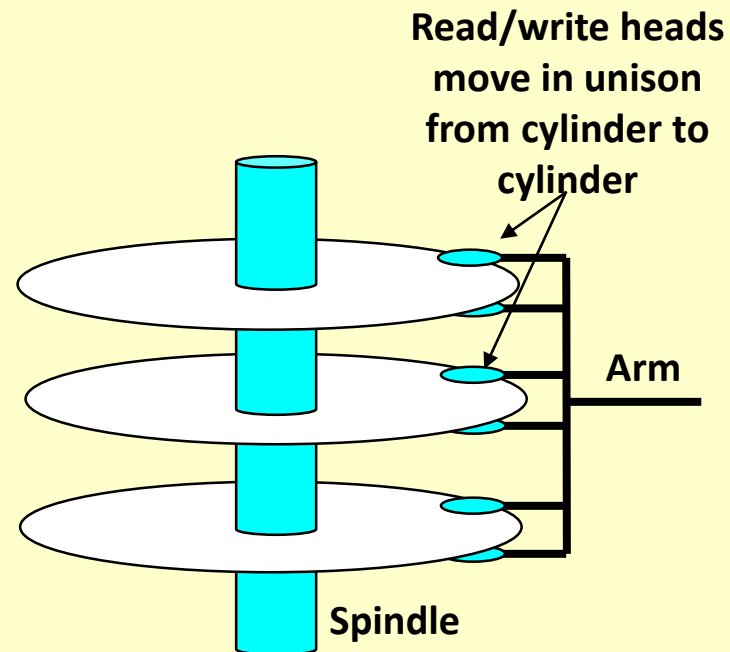
The disk surface spins at a fixed rotational rate



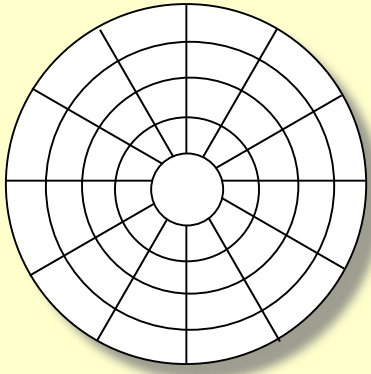
The read/write *head* is attached to the end of the *arm* and flies over the disk surface on a thin cushion of air.

By moving radially, the arm can position the read/write head over any track.

# Disk Operation (Multi-Platter View)



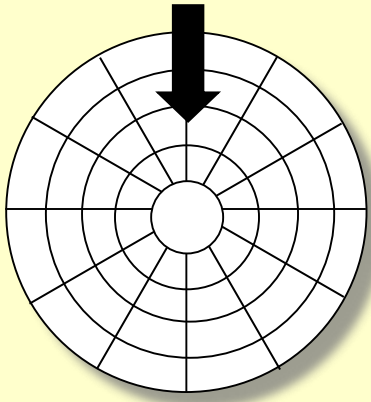
# Disk Structure - top view of single platter



**Surface organized into tracks**

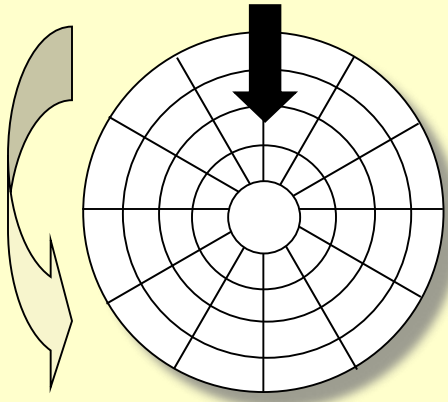
**Tracks divided into sectors**

# Disk Access



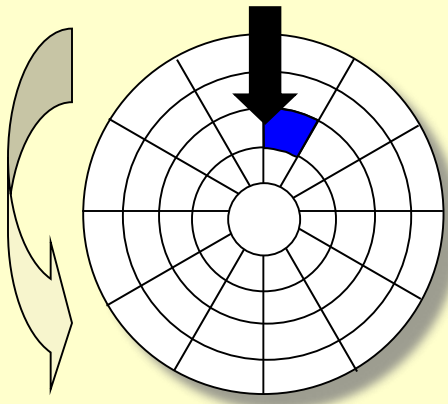
**Head in position above a track**

# Disk Access



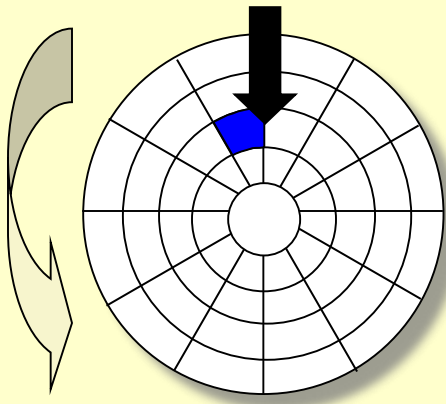
**Rotation is counter-clockwise**

# Disk Access – Read



**About to read blue sector**

# Disk Access – Read

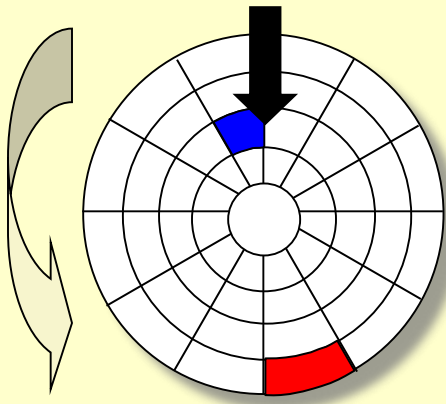


After **BLUE** read

After reading blue sector



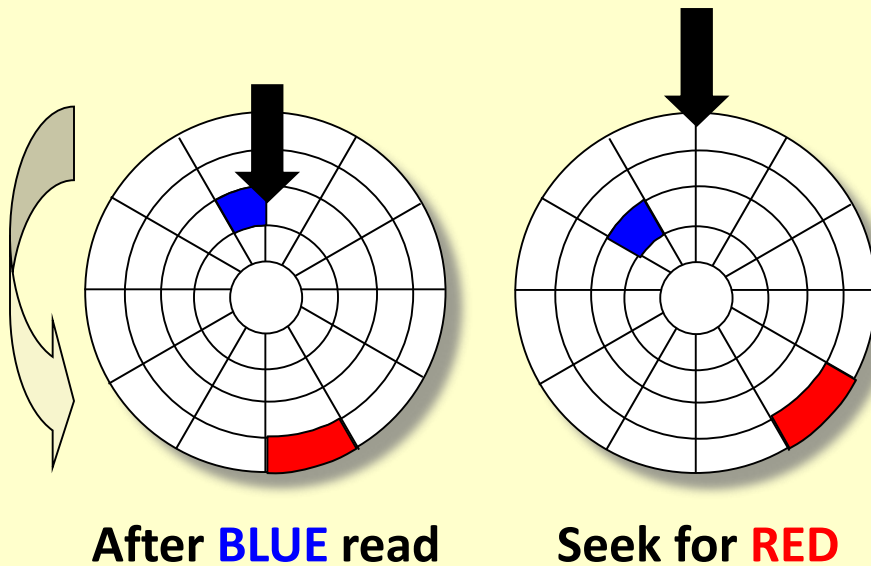
# Disk Access – Read



After **BLUE** read

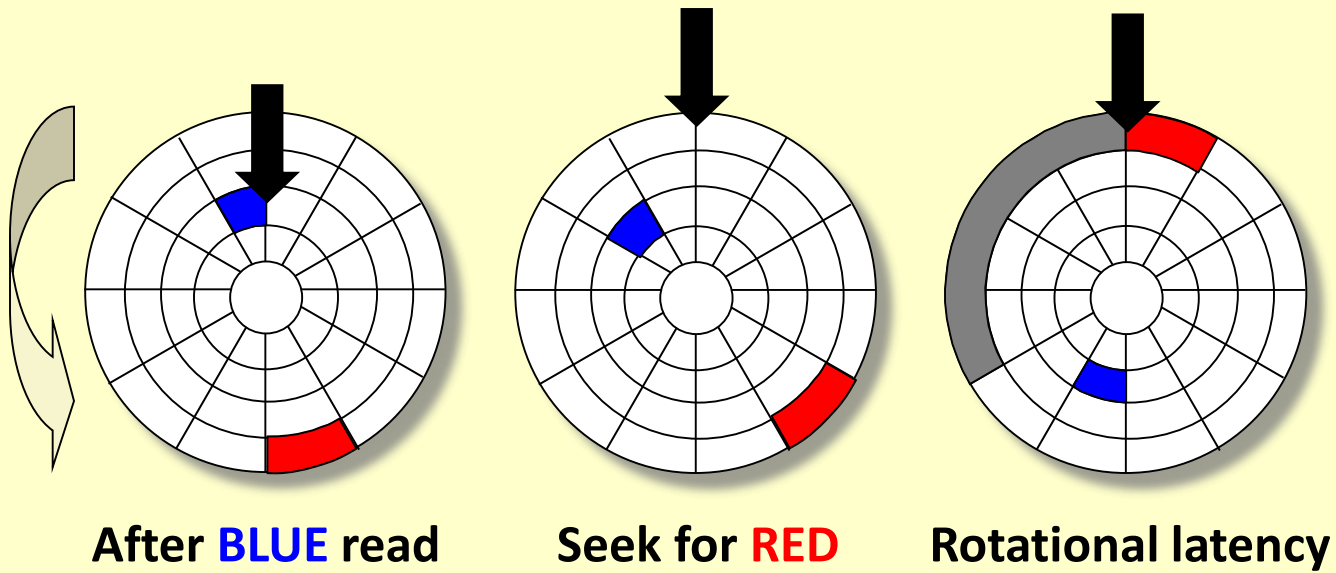
**Red request scheduled next**

# Disk Access – Seek



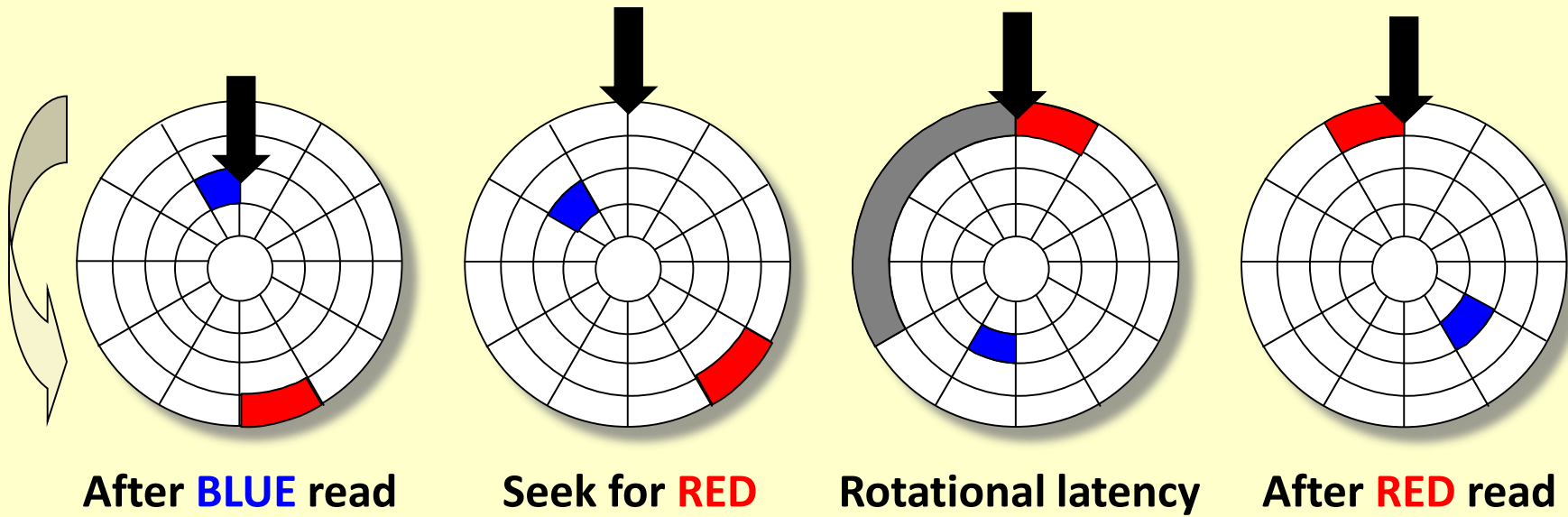
Seek to red's track

# Disk Access – Rotational Latency



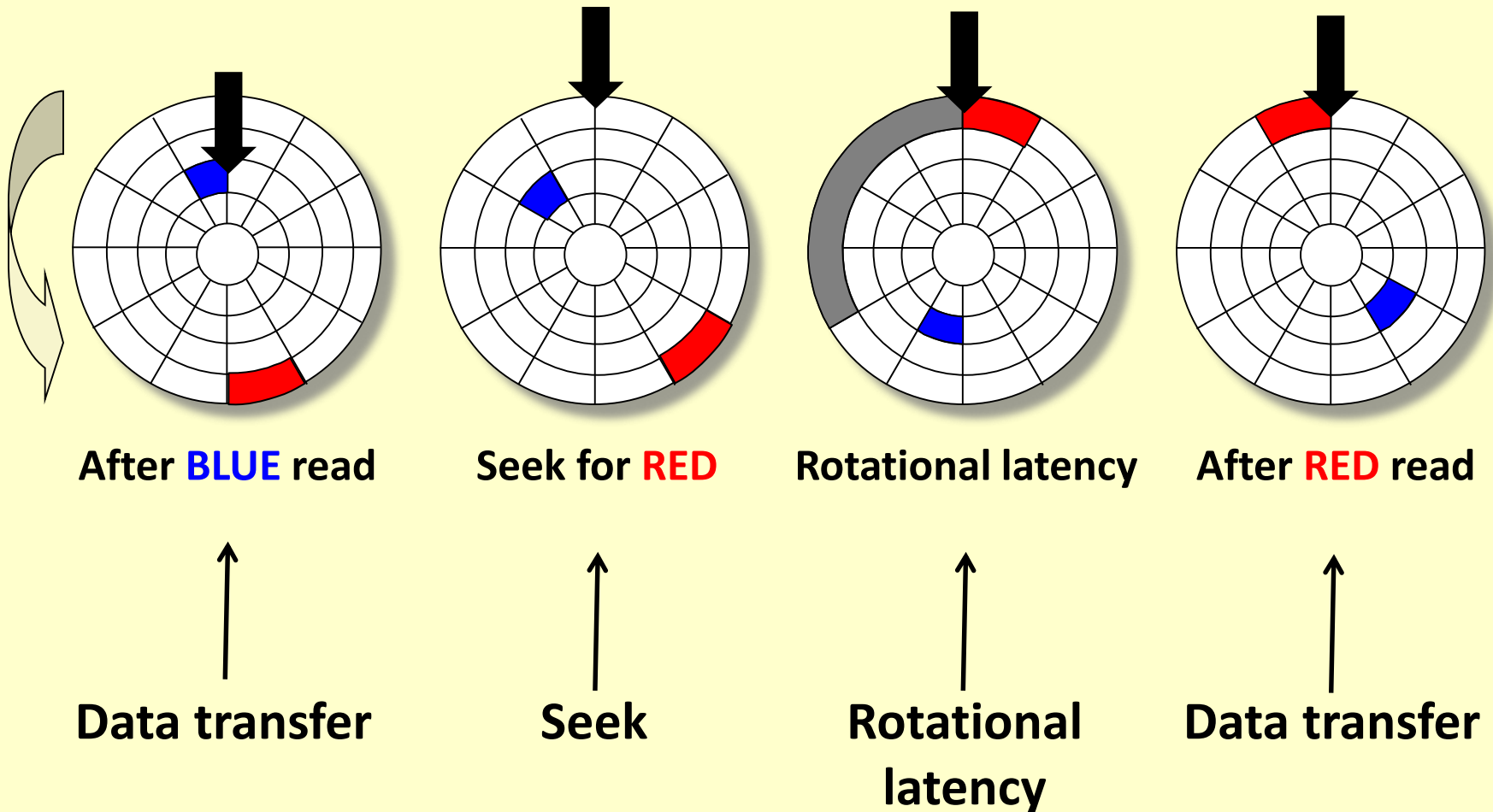
**Wait for red sector to rotate around**

# Disk Access – Read



**Complete read of red**

# Disk Access – Service Time Components



# 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  $T_{\text{avg rotation}} = 7200 \text{ RPMs}$

## ■ Transfer time ( $T_{\text{avg transfer}}$ )

- Time to read the bits in the target sector.
- $T_{\text{avg transfer}} = 1/\text{RPM} \times 1/(\text{avg \# sectors/track}) \times 60 \text{ secs}/1 \text{ min.}$

# Disk Access Time Example

## ■ Given:

- Rotational rate = 7,200 RPM
- Average seek time = 9 ms.
- Avg # sectors/track = 400.

## ■ Derived:

- $T_{\text{avg rotation}} = 1/2 \times (60 \text{ secs}/7200 \text{ RPM}) \times 1000 \text{ ms/sec} = 4 \text{ ms}.$
- $T_{\text{avg transfer}} = 60/7200 \text{ RPM} \times 1/400 \text{ secs/track} \times 1000 \text{ ms/sec} = 0.02 \text{ ms}$
- $T_{\text{access}} = 9 \text{ ms} + 4 \text{ ms} + 0.02 \text{ ms}$

## ■ Important points:

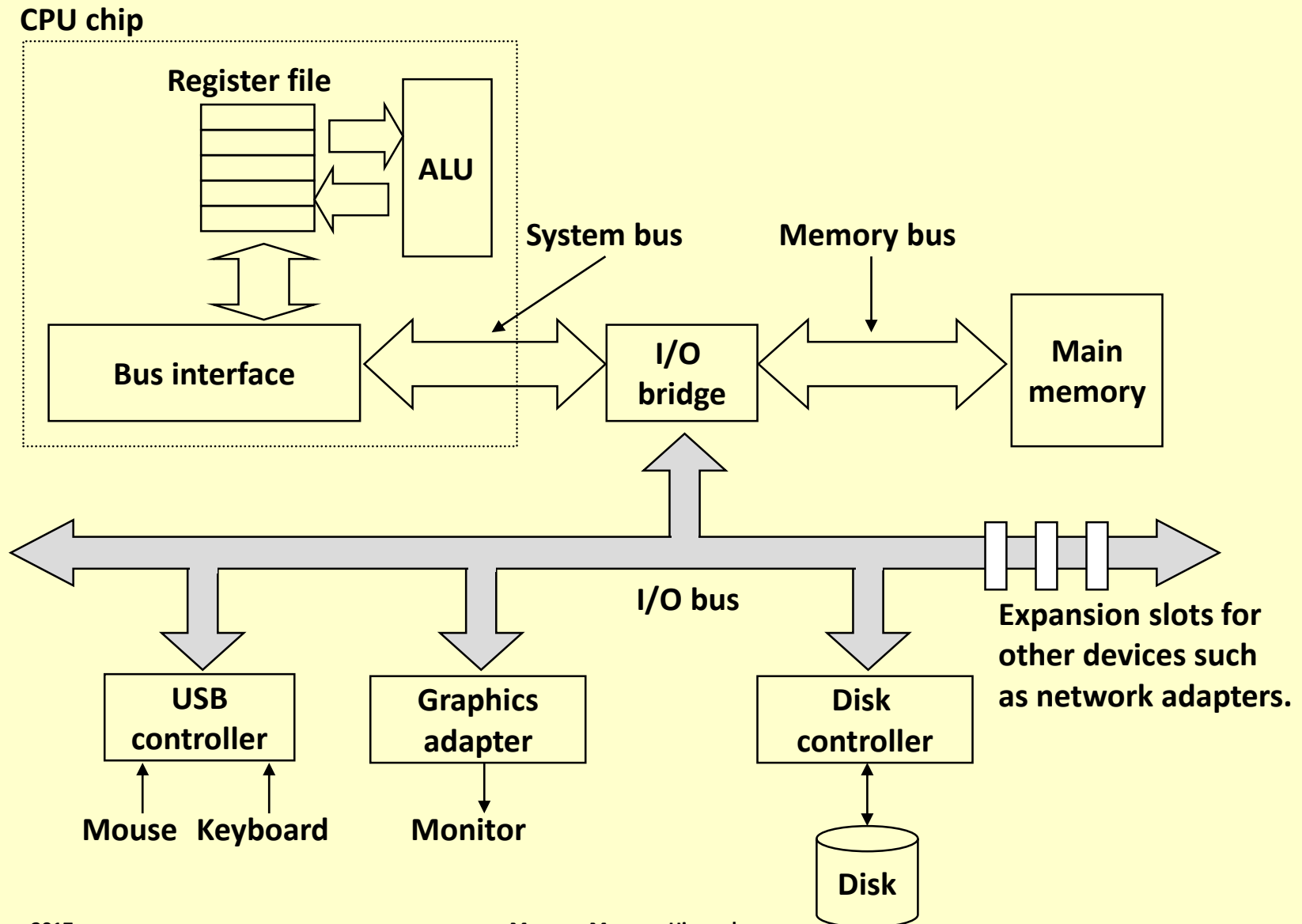
- Access time dominated by seek time and rotational latency.
- First bit in a sector is the most expensive, the rest are free.
- SRAM access time is about 4 ns/doubleword, DRAM about 60 ns
  - Disk is about 40,000 times slower than SRAM,
  - 2,500 times slower than DRAM.

# 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.**
  - Accounts for the difference in “formatted capacity” and “maximum capacity”

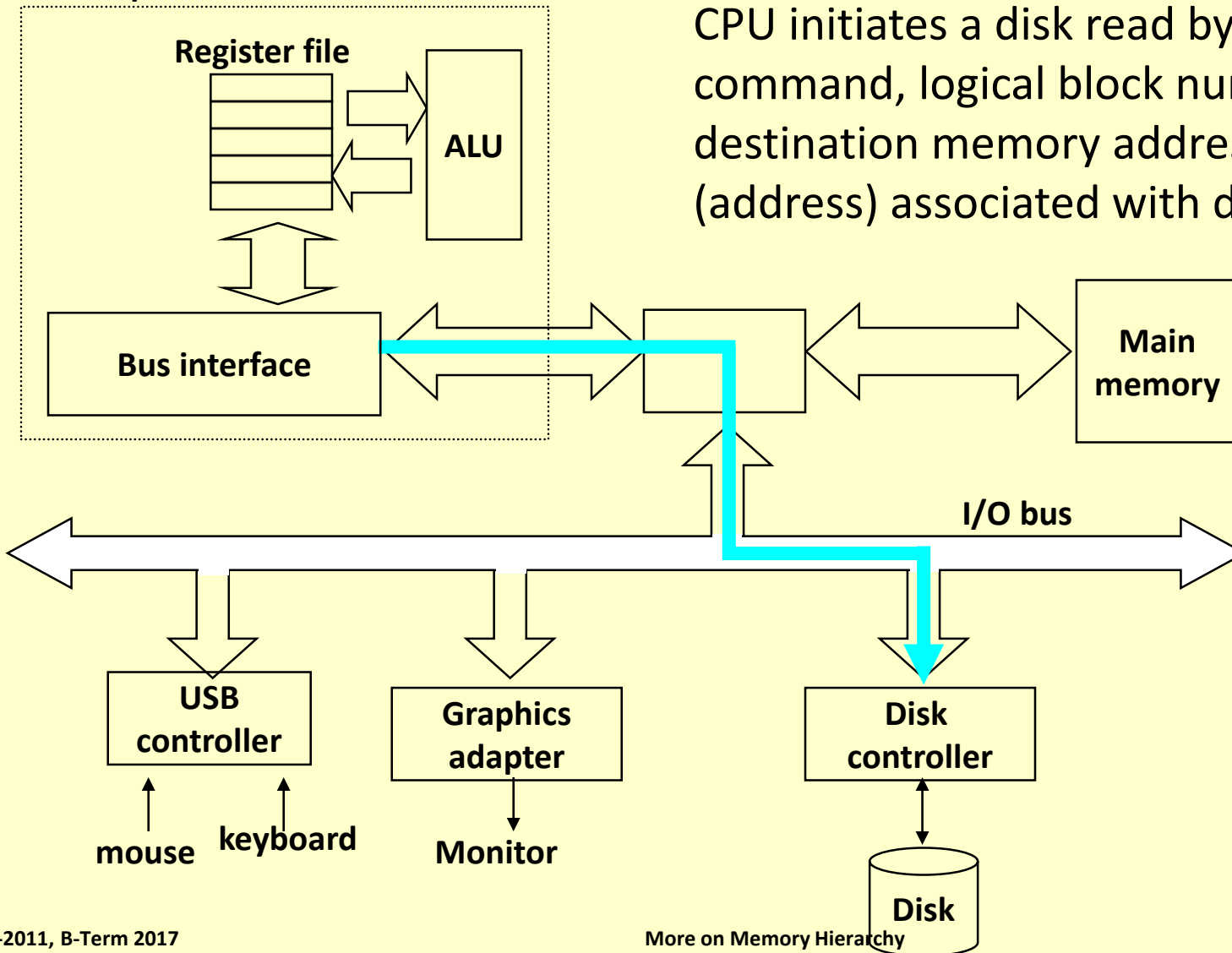


# I/O Bus



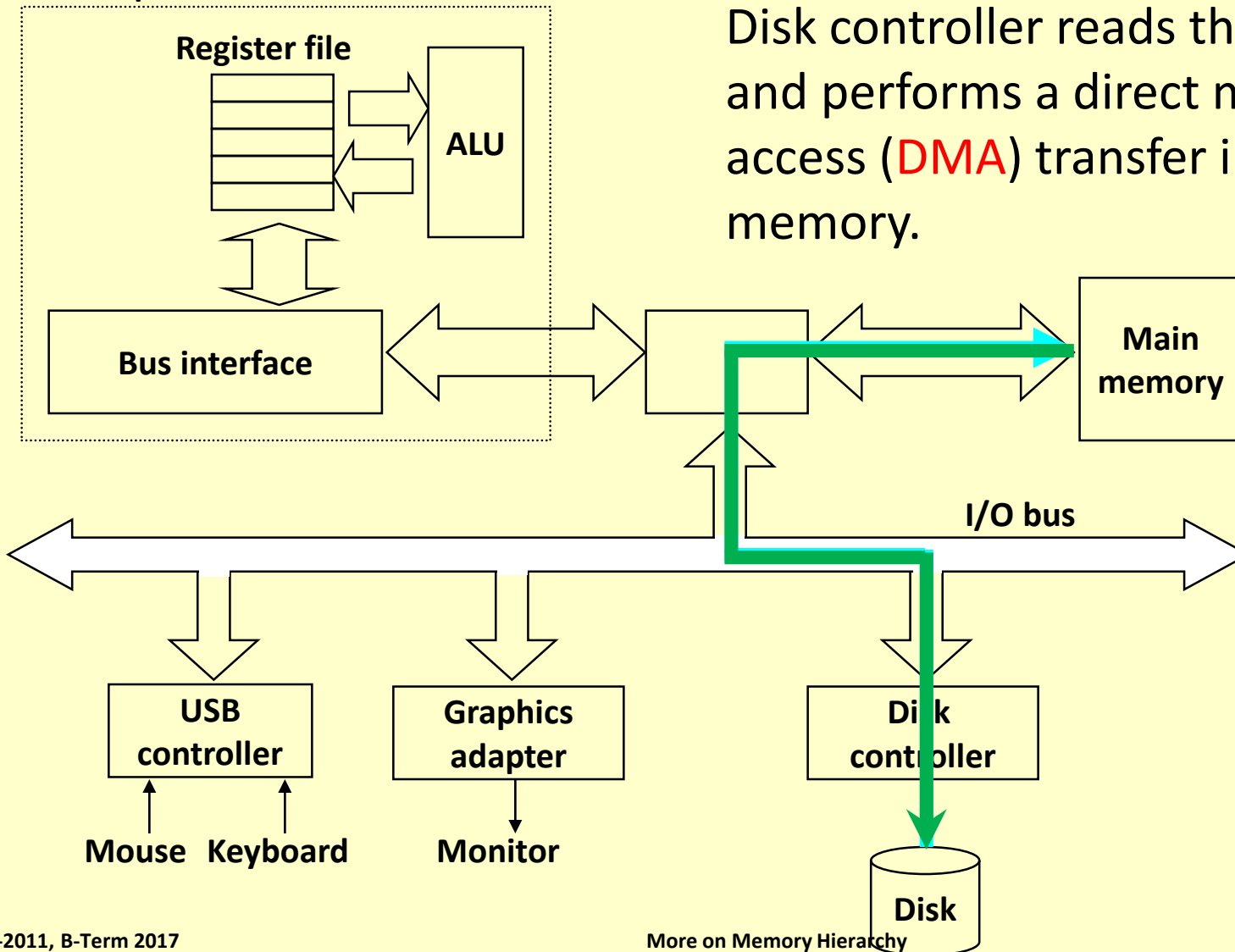
# Reading a Disk Sector (1)

CPU chip

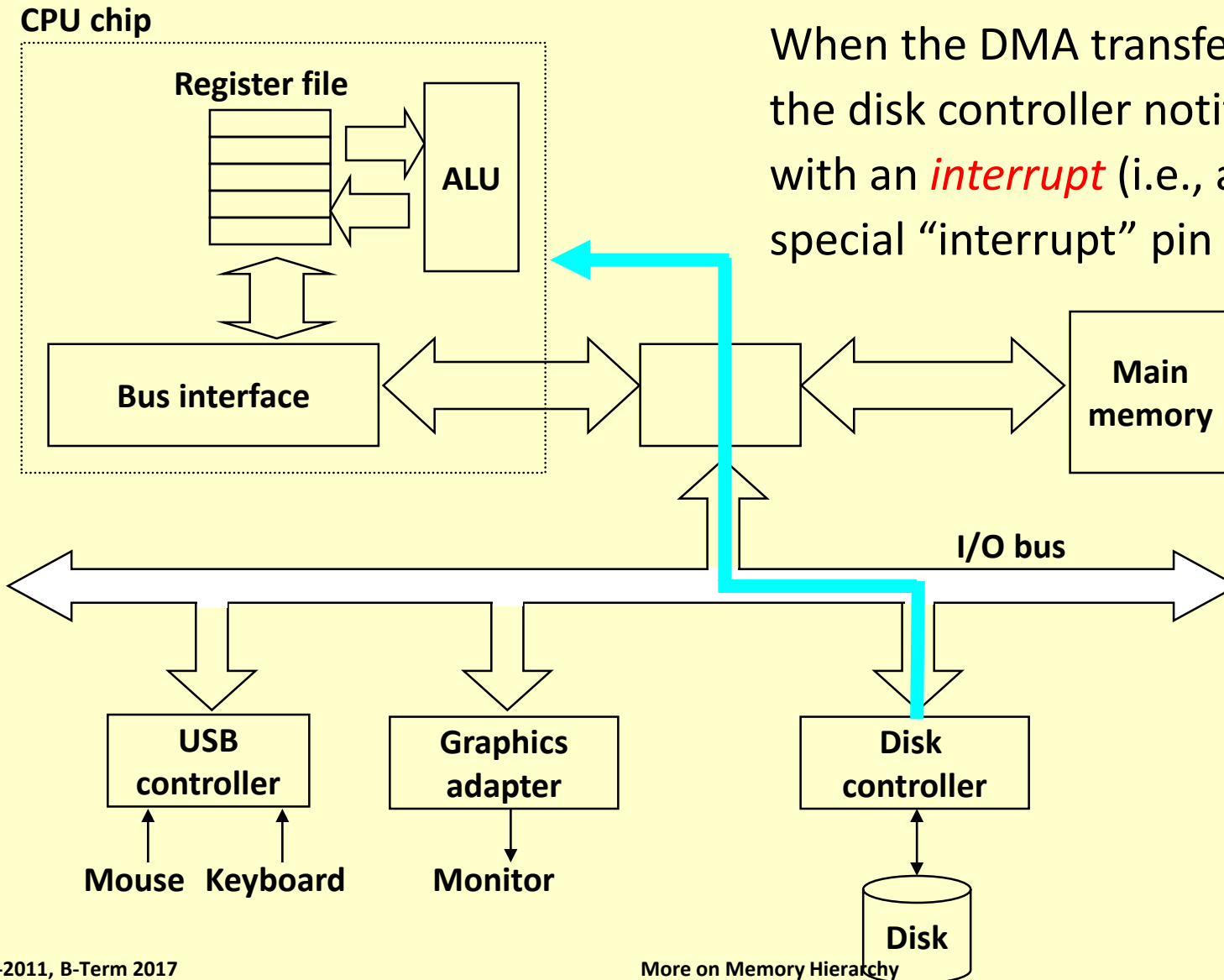


# Reading a Disk Sector (2)

CPU chip



# Reading a Disk Sector (3)



# Questions?