The Memory Hierarchy

CS2011: Introduction to Computer Systems
Lecture 11 (6.1, 6.2, 6.3)

The Memory Hierarchy

- The memory abstraction
- RAM: main memory building block
- Storage technologies and trends
- Locality of reference
- The memory hierarchy

Reading & Writing Memory

Read

- Transfer data from memory to CPU

  ```
  movq 8(%rsp), %rax
  ```
- "Load" operation

Write

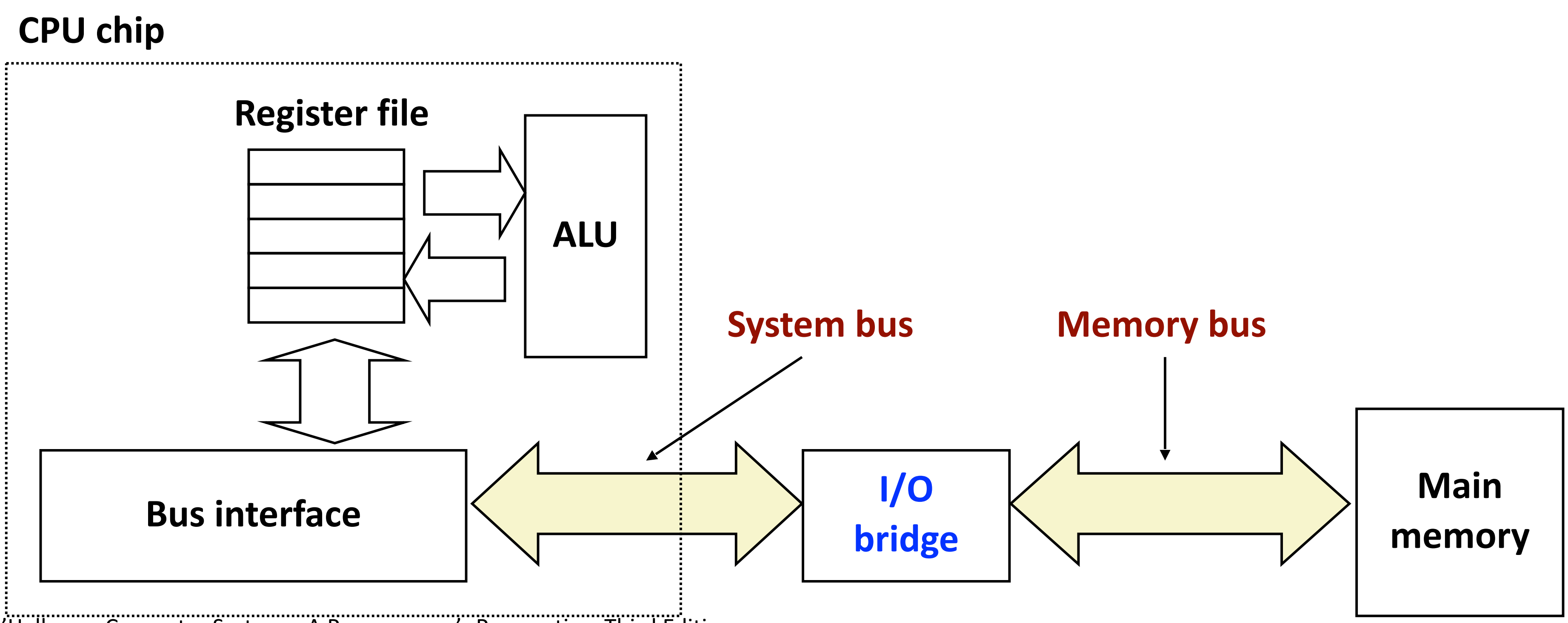
- Transfer data from CPU to memory

  ```
  movq %rax, 8(%rsp)
  ```
- "Store" operation

From 5th lecture

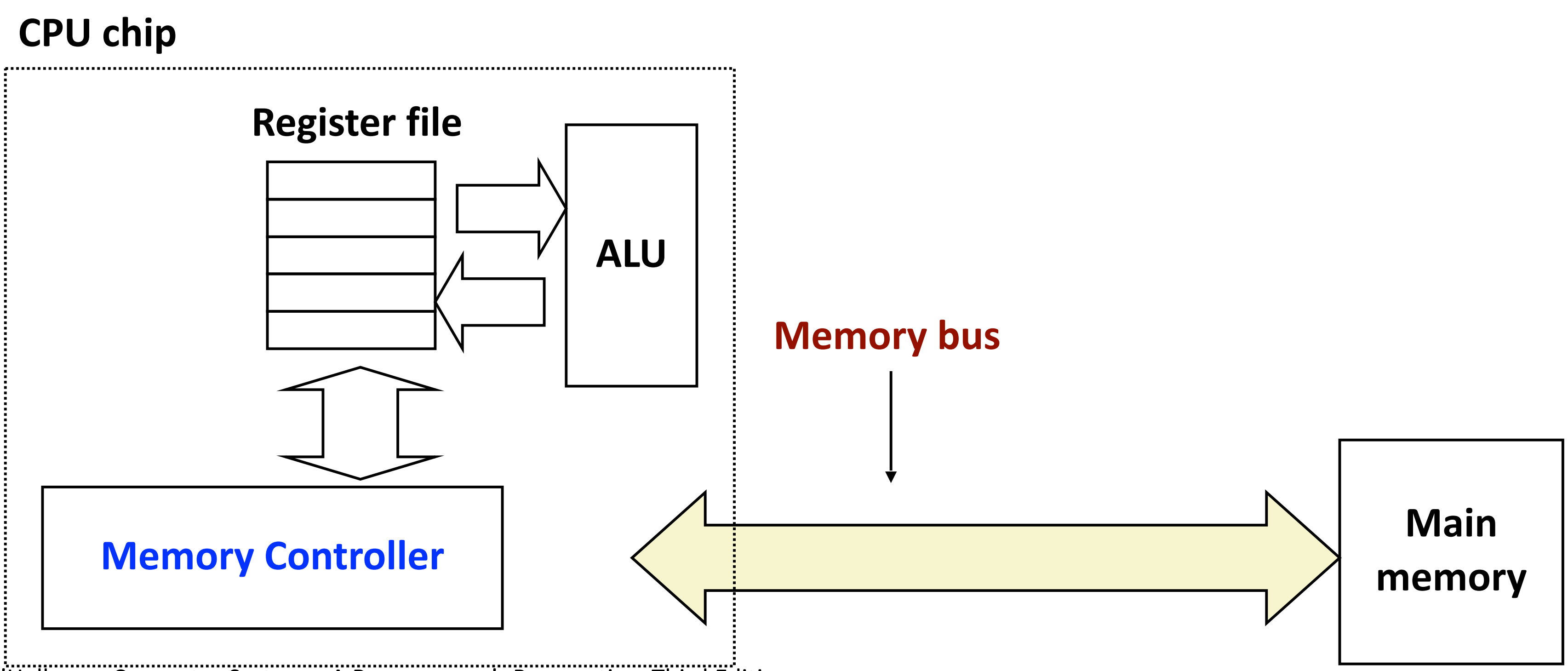
Traditional Bus Structure Connecting CPU and Memory

- A **bus** is a collection of parallel wires that carry **address, data**, and **control signals**. *Address and data can share same set of wires*
- Buses are typically shared by multiple devices.
- **I/O Bridge**: **translates** system bus's **electrical signals to/from** memory bus's electric signals. Also connects to I/O Bus.
  - I/O Bridge also **contains** the **memory controller**



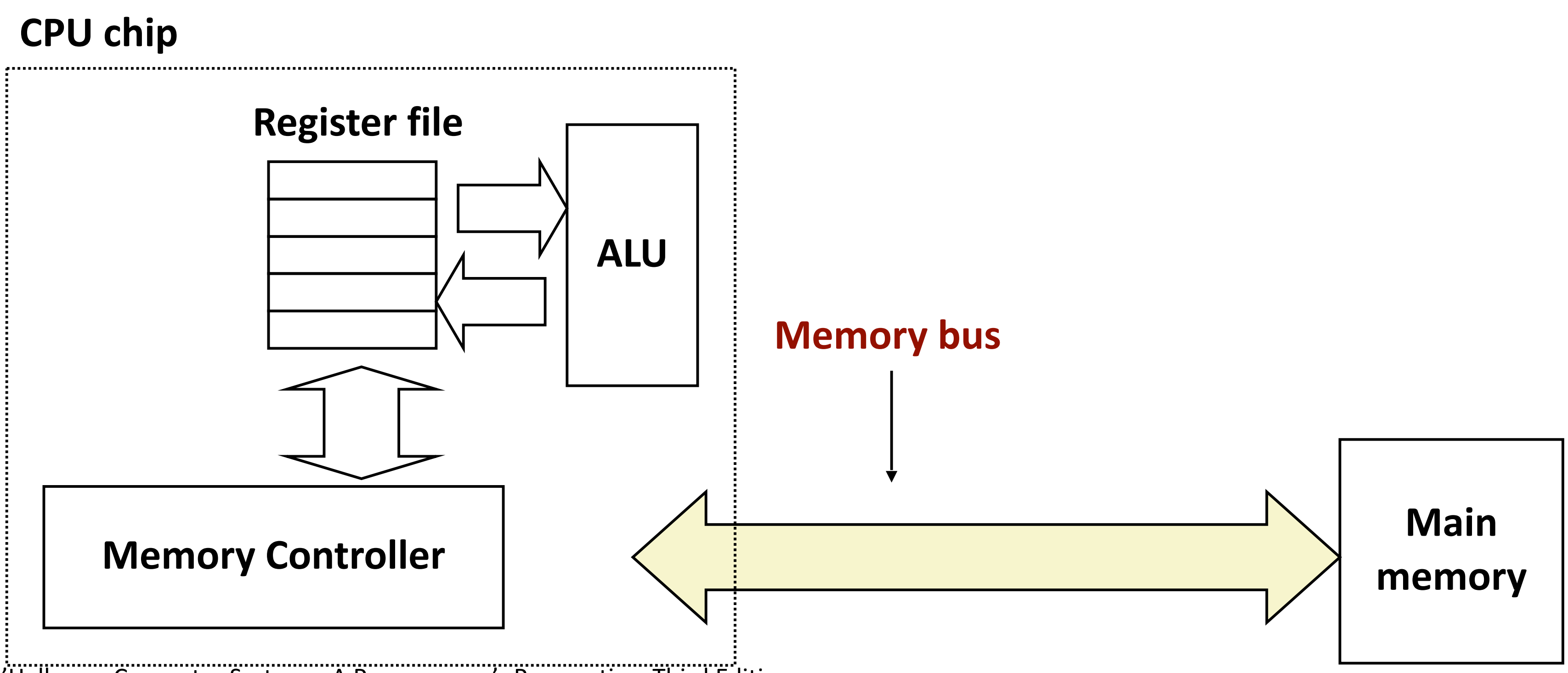
Modern Connection between CPU and Memory

- A **bus** is a collection of parallel wires that carry **address, data**, and **control signals**. *Address and data can share same set of wires*
- Buses are typically shared by multiple devices.
- **Memory controller closer to CPU**



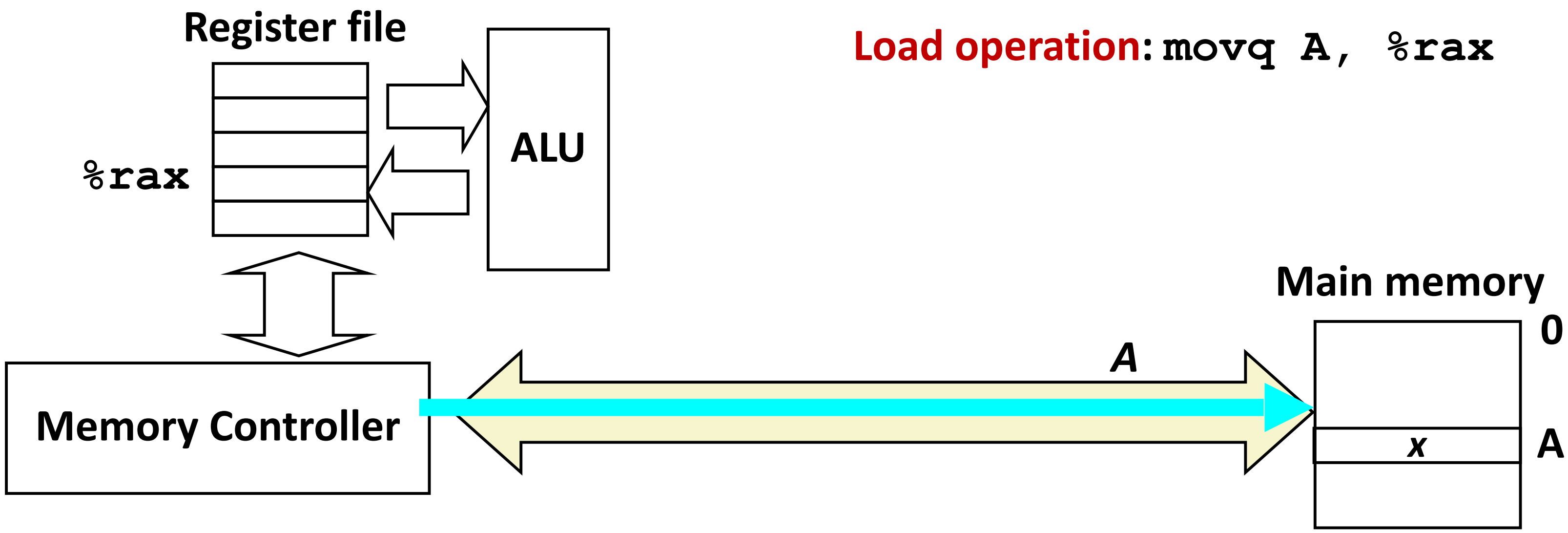
Memory Read/Write Transactions

- **Bus interface** in CPU chip initiates a memory read/write transaction
  - **Control signals** on control wires/bus **identify and synchronize transaction**
- Read or write transaction consists of **three steps**
- Main memory and CPU **sense** address and data signals on the bus when address or data available



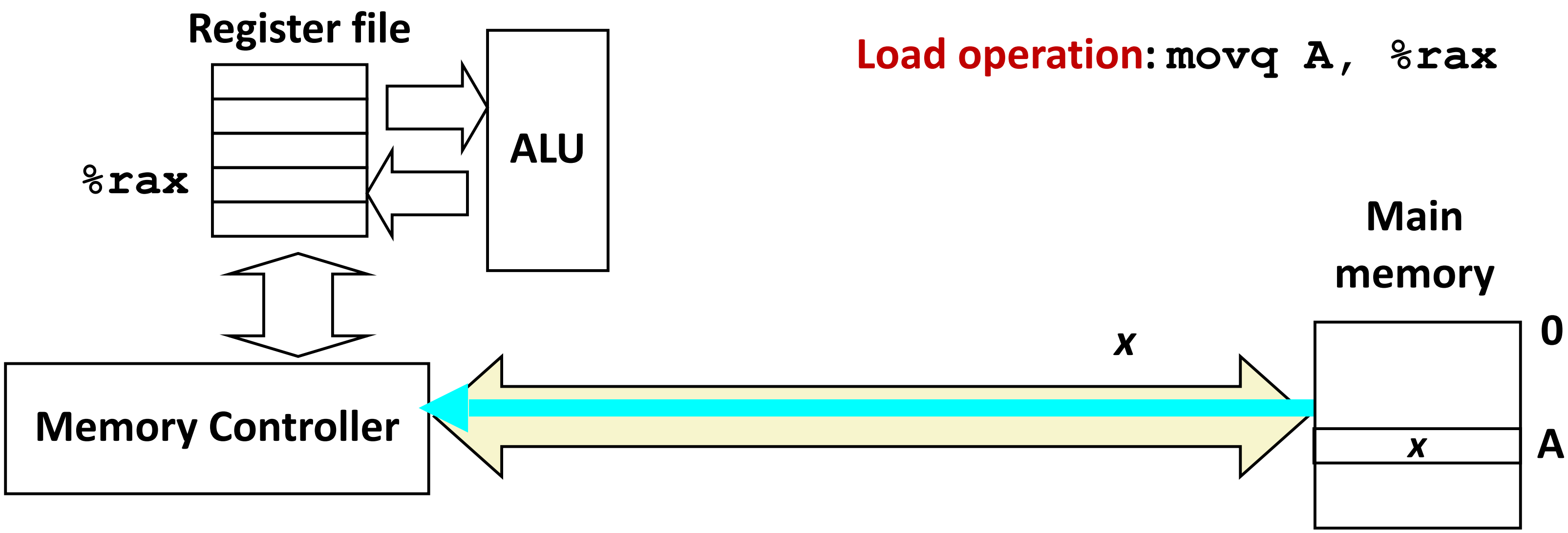
Memory Read Transaction (Step 1)

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



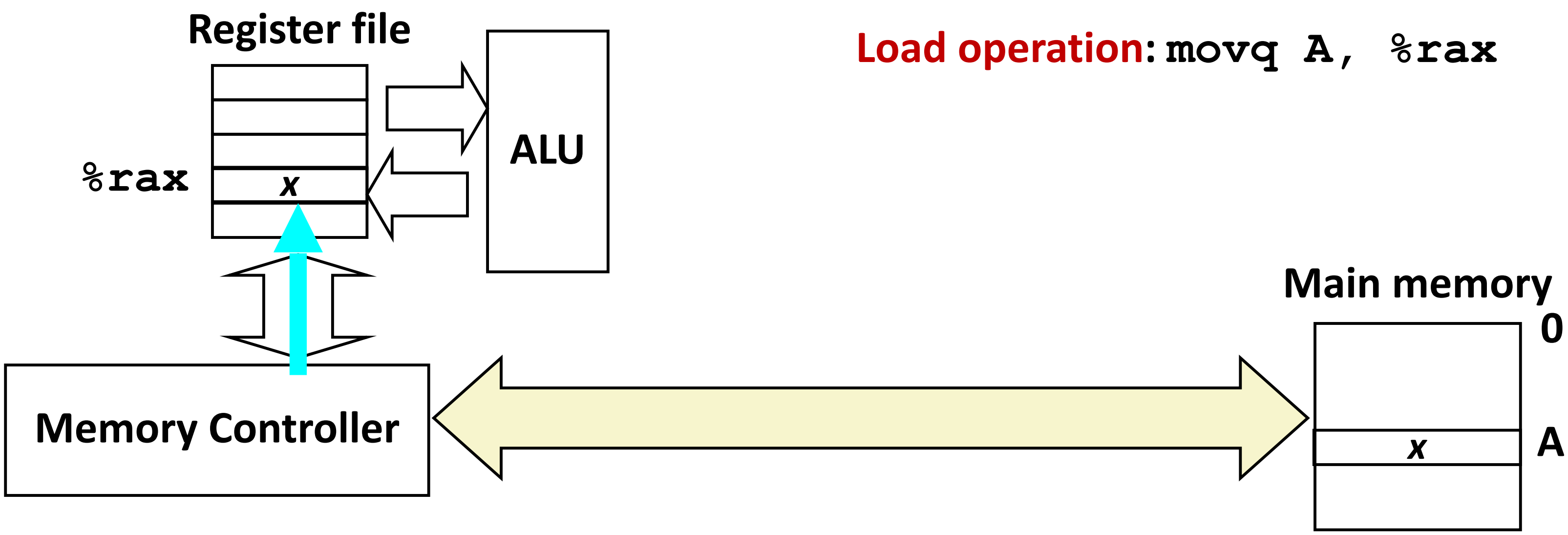
Memory Read Transaction (Step 2)

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



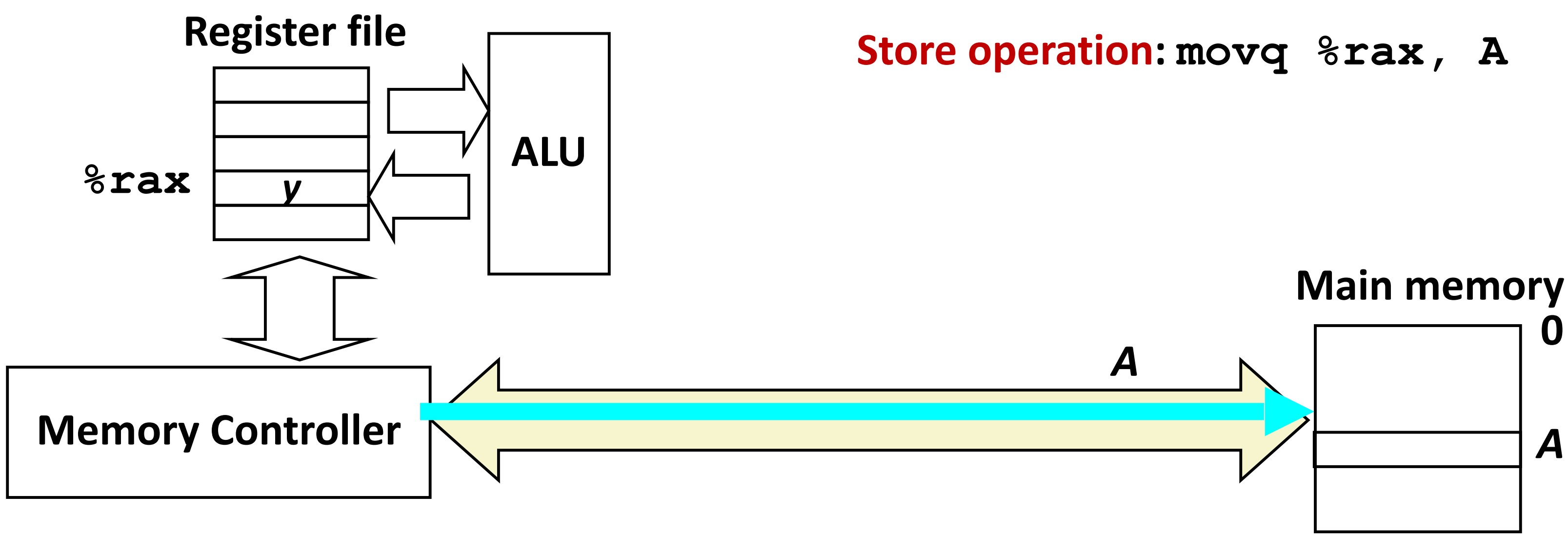
Memory Read Transaction (Step 3)

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



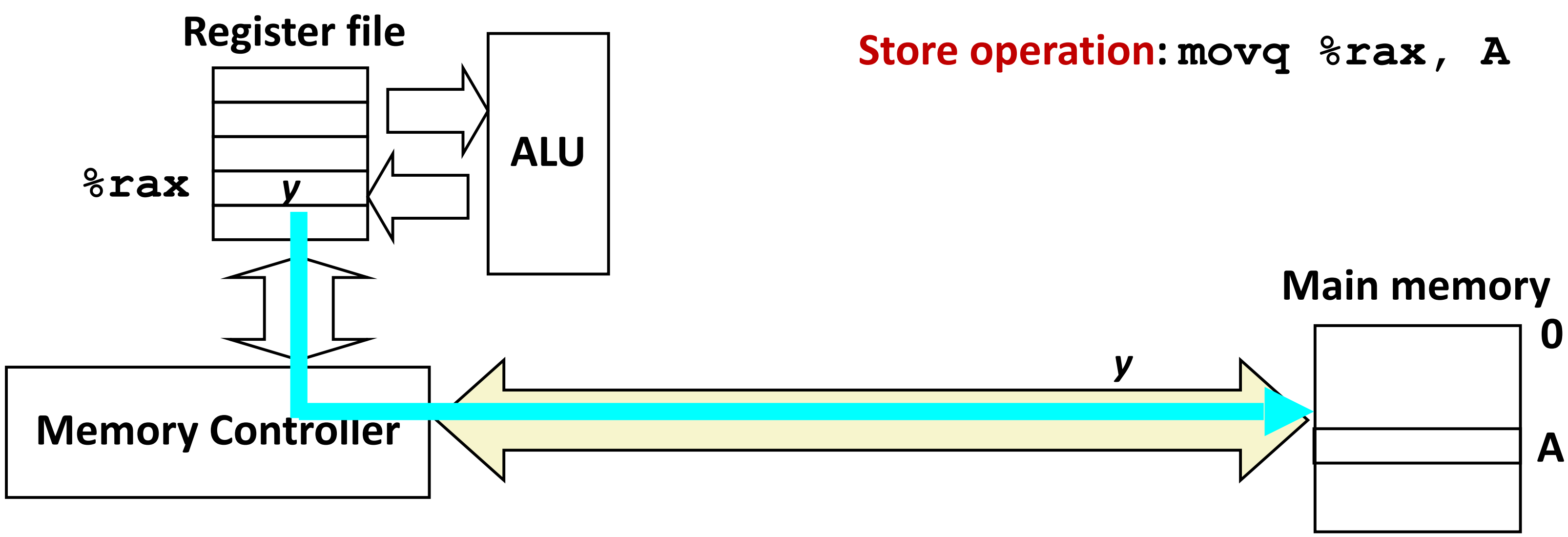
Memory Write Transaction (Step 1)

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



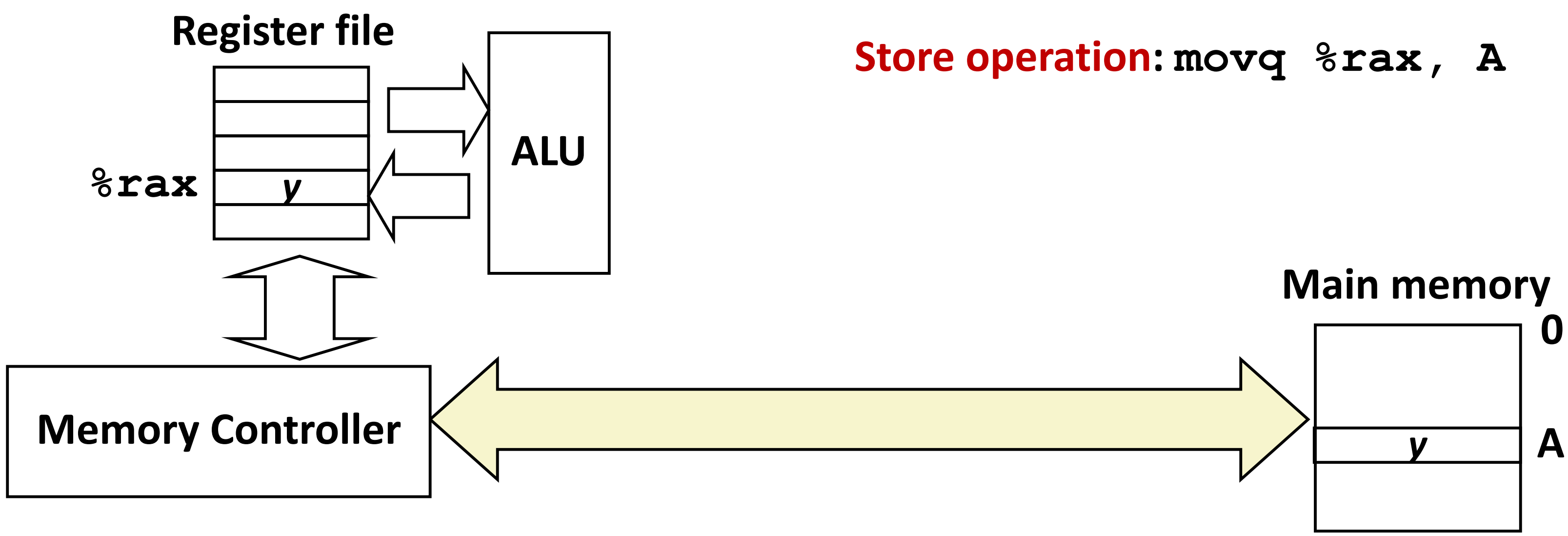
Memory Write Transaction (Step 2)

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



Memory Write Transaction (Step 3)

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



The Memory Hierarchy

- The memory abstraction
- **RAM : main memory building block**
- Storage technologies and trends
- Locality of reference
- The memory hierarchy

Random-Access Memory (RAM)

Key features

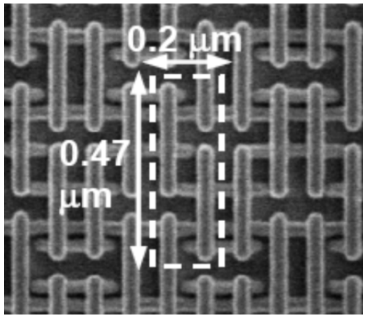
- RAM is **traditionally** packaged as a chip.
  - or embedded as part of processor/CPU chip
- Basic storage unit is normally a cell (one bit per cell).
- Multiple RAM chips form a memory.

RAM comes in two varieties:

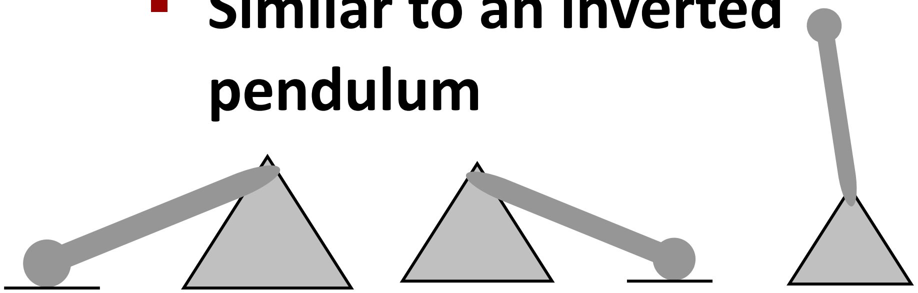
- 1) **S**RAM (**S**tatic RAM)
- 2) **D**RAM (**D**ynamic RAM)

RAM Technologies

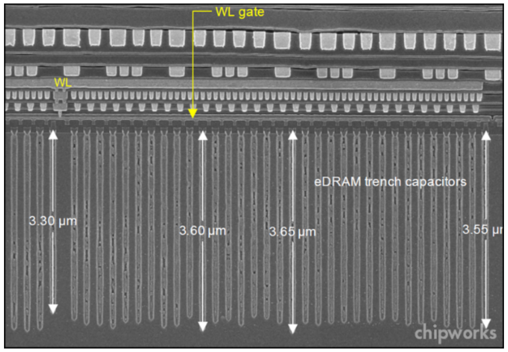
- **SRAM**



- **6 transistors / bit (cell)**
- Holds **state/voltage indefinitely**
  - **Similar to an inverted pendulum**



- **DRAM**



- **1 Transistor + 1 capacitor / bit (cell)**
  - Capacitor oriented vertically
- Must **refresh state/voltage** periodically
  - Due to sensitivity to disturbance or electric noise
  - Reads each bit out and rewrites it back

SRAM vs DRAM Summary

	Trans. per bit	Access time	Needs refresh?	Needs EDC?	Cost	Applications
SRAM	6 or 8	1x	No	Maybe	100x	Cache memories
DRAM	1	10x	Yes	Yes	1x	Main memories, frame buffers

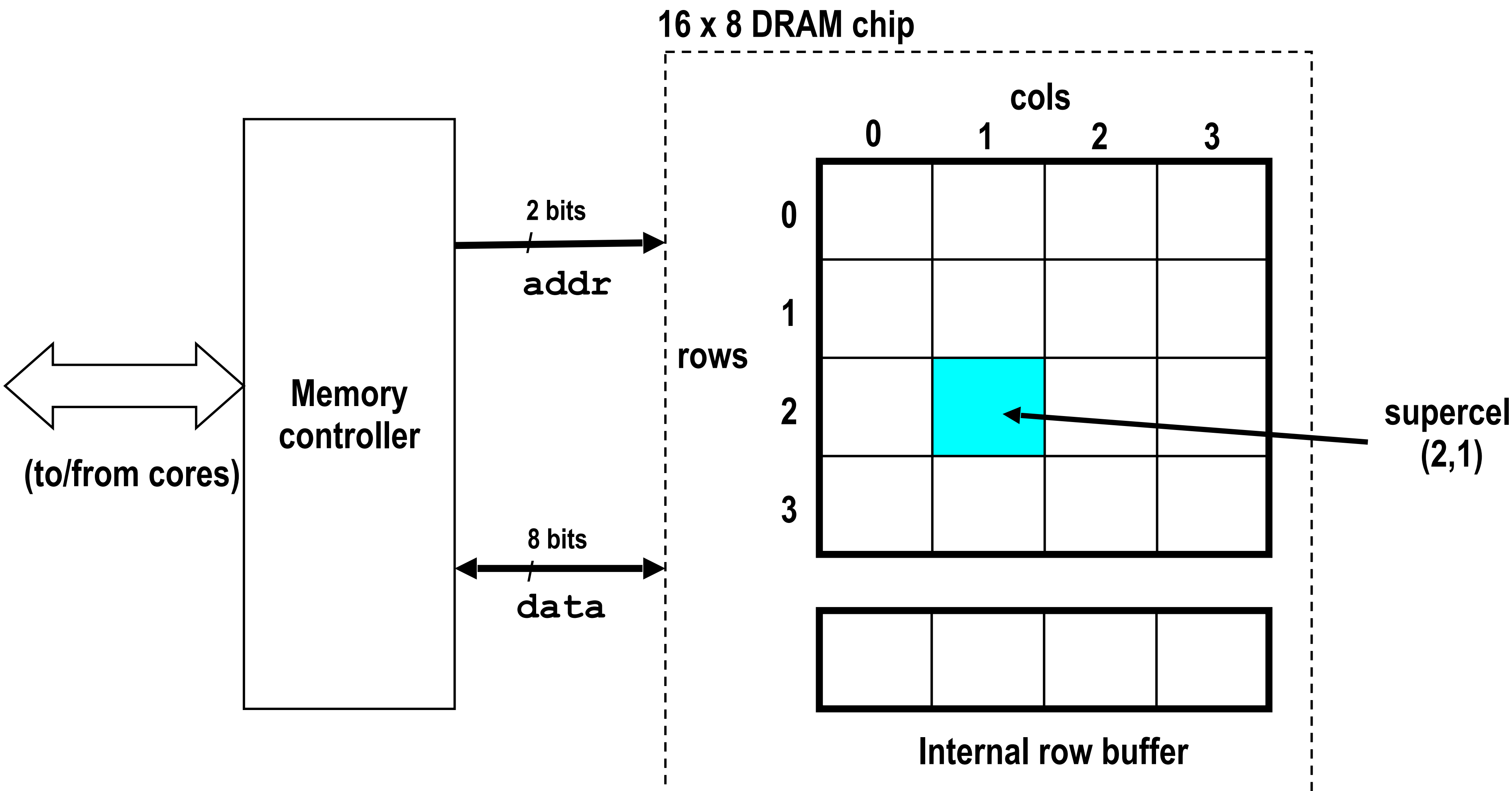
EDC: Error detection and correction (64-bit words are encoded using additional 8 EDC bits)

Trends

- SRAM scales with semiconductor technology
  - *Reaching its limits*
- DRAM scaling limited by need for minimum capacitance
  - Aspect ratio limits how deep can make capacitor
  - *Also reaching its limits*

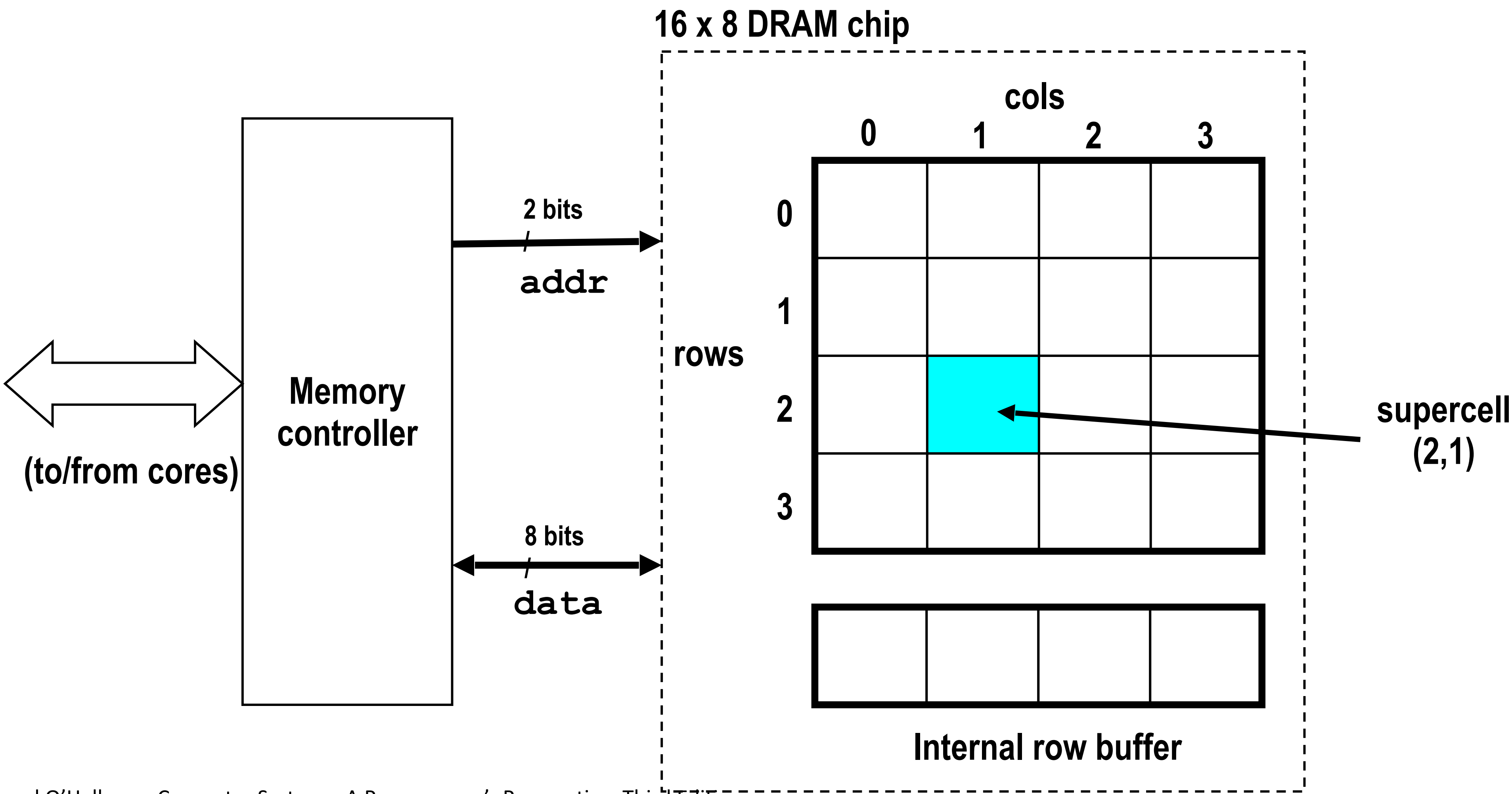
Conventional DRAM Organization

- ***d* x *w* DRAM:**
  - $d \cdot w$ total bits organized as *d* supercells of size *w* bits
  - In the example, **2 address pins** are needed in order to index **4 rows/cols**



Conventional DRAM Organization

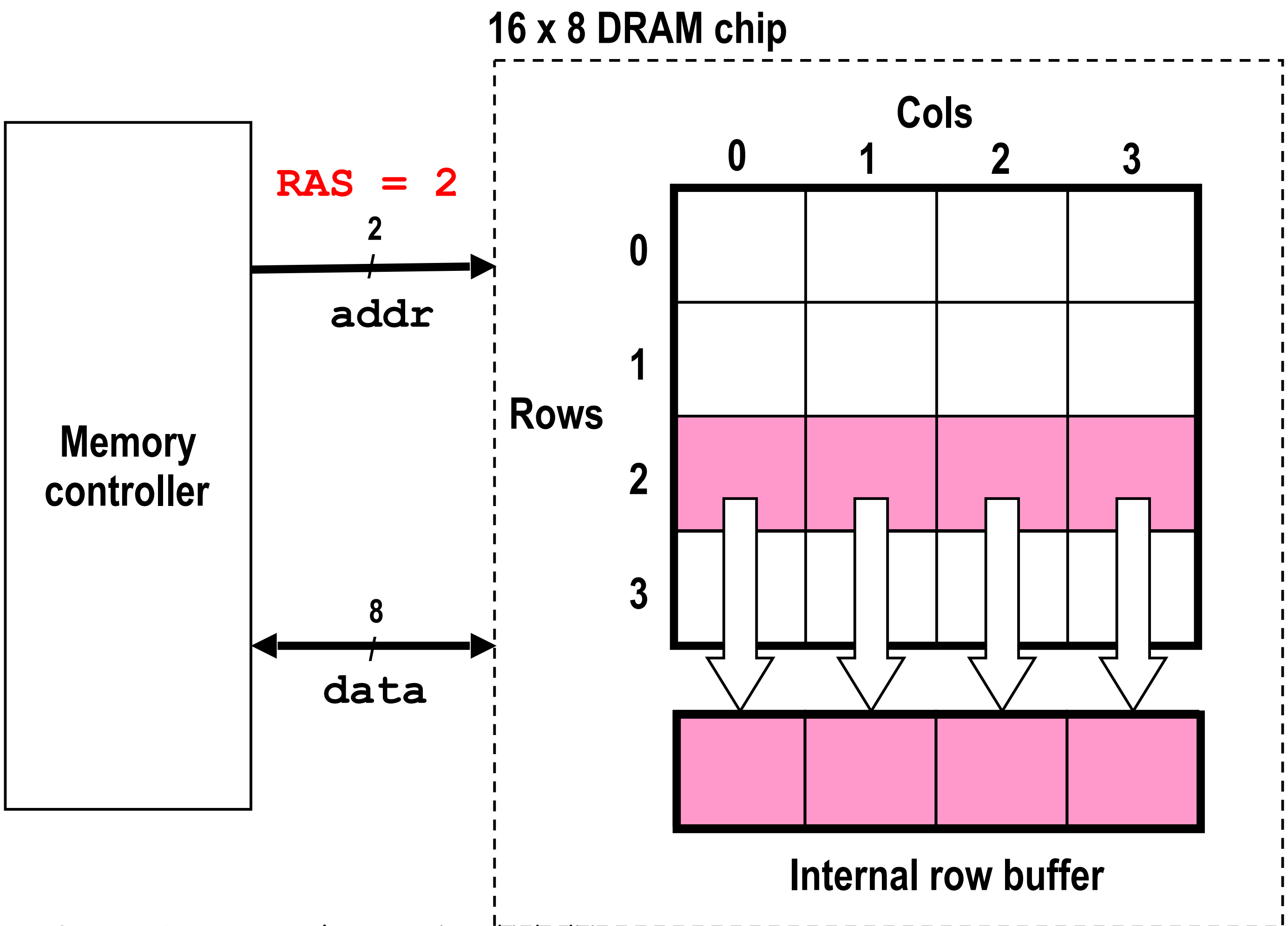
- *d* x *w* DRAM instead of linear array design:
  - Advantage: reduce number of address pins needed on the chip
  - Disadvantage: requires two steps to send address (longer access time)



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**.

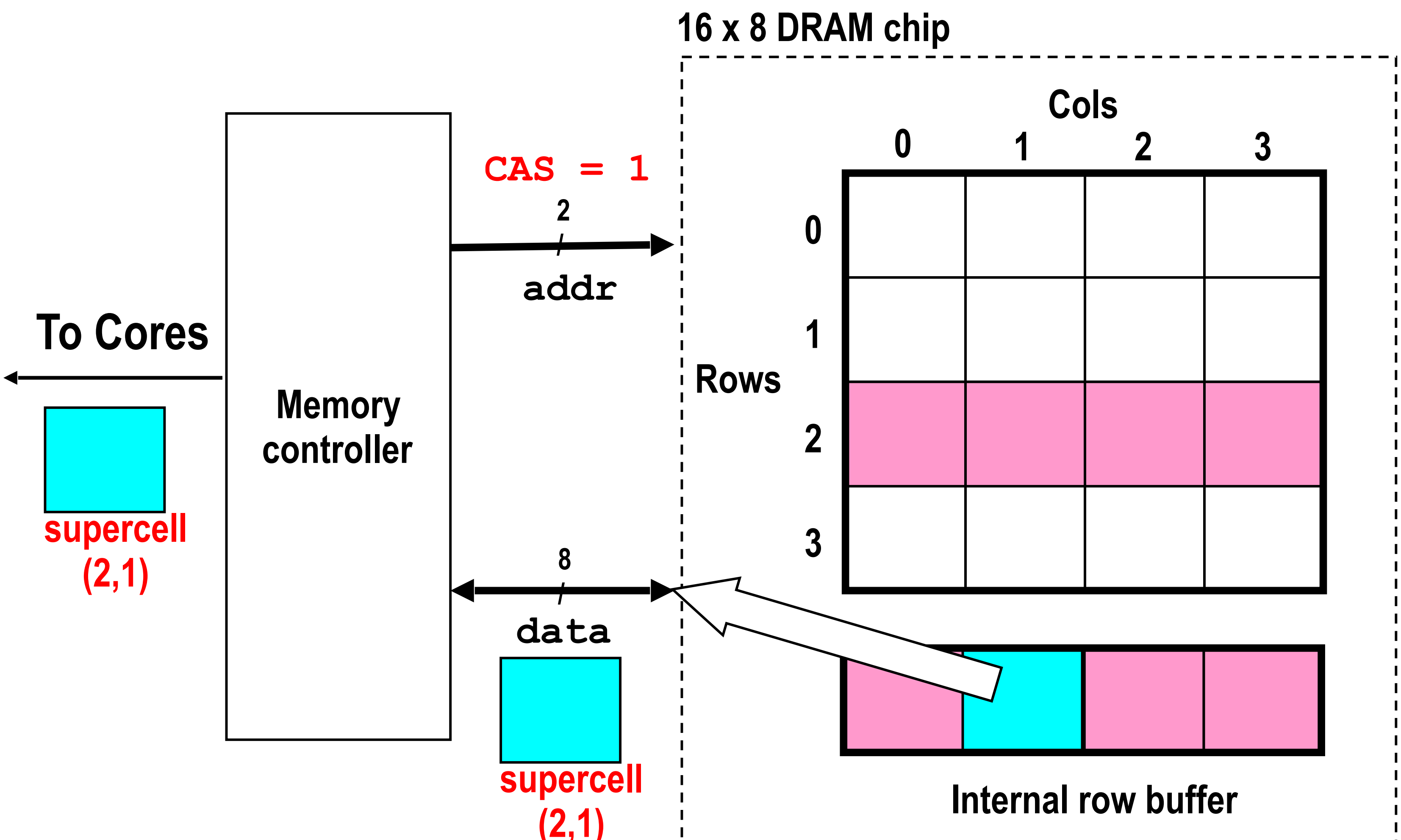


Reading DRAM Supercell (2,1)

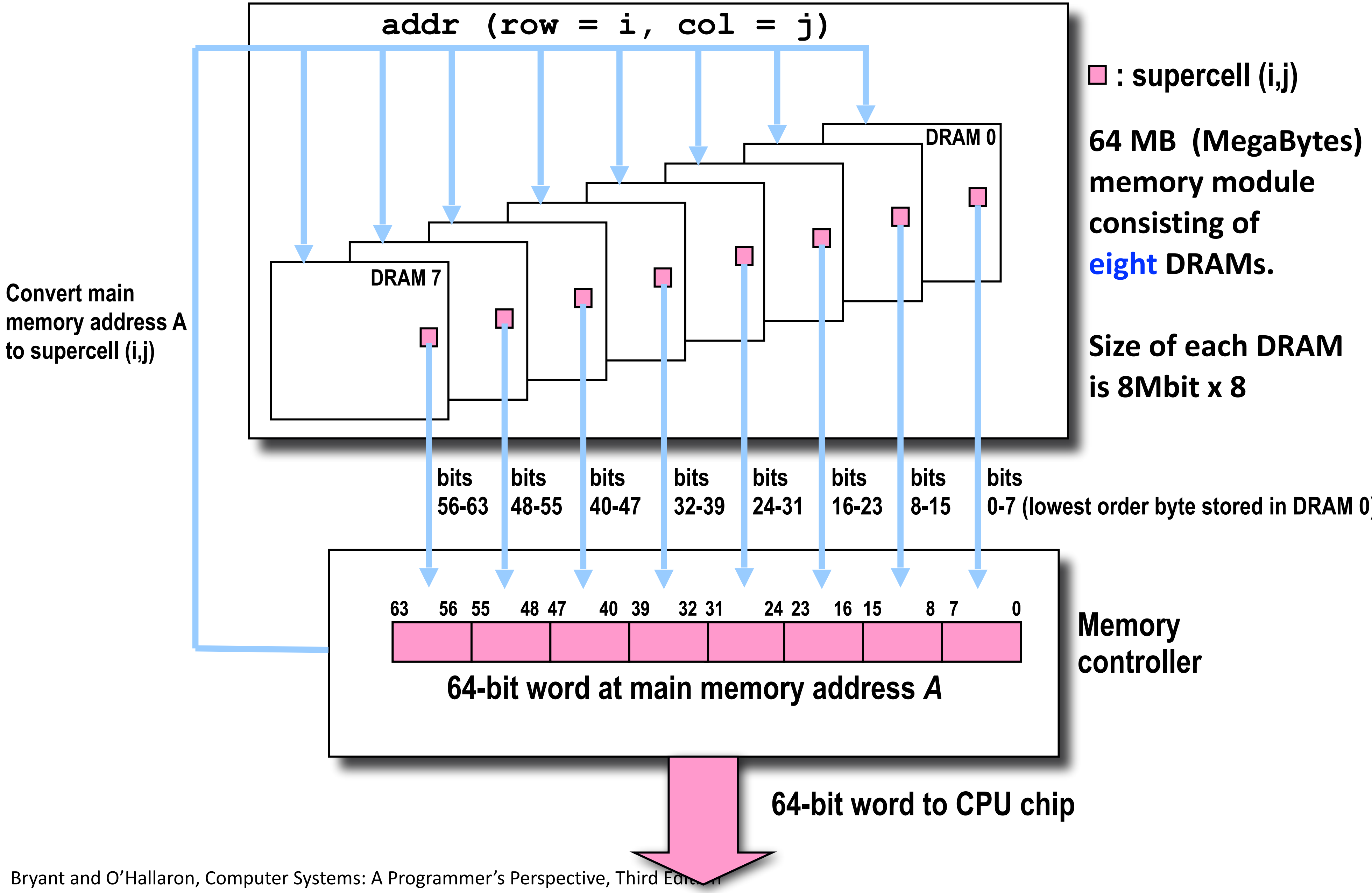
Step 2(a): Column access strobe (CAS) selects column 1.

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

Step 3: *All data written back to row to provide refresh*



Memory Modules



64 MB (MegaBytes) memory module consisting of eight DRAMs.

Size of each DRAM is 8Mbit x 8

Enhanced DRAMs

- **Operation of DRAM cell has not changed since its invention**
  - 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
  - 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), DDR4 (16 bits)
    - By 2010, standard for most server and desktop systems
    - Intel Core i7 supports DDR3 and DDR4 SDRAM

RAM vs 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
  - Electrically eraseable PROM (EEPROM): electronic erase capability
  - Flash memory: EEPROMs, with partial (block-level) erase capability
    - Wears out after about 100,000 erasings
  - 3D XPoint (Intel Optane) & emerging NVMs
    - New materials

- **Uses for Nonvolatile Memories**
  - Firmware programs stored in a ROM (BIOS, controllers for disks, network cards, graphics accelerators, security subsystems,…)
  - Disk caches
  - **S**olid **s**tate **d**isks (replacing rotating disks)

The Memory Hierarchy

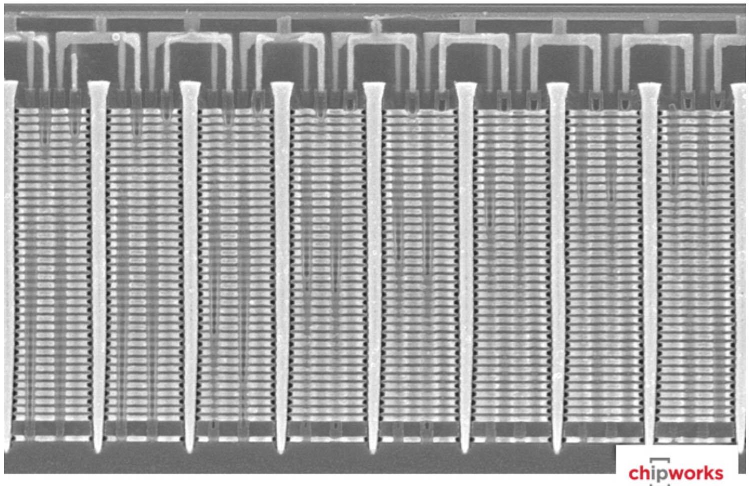
- The memory abstraction
- RAM : main memory building block
- **Storage technologies and trends**
- Locality of reference
- The memory hierarchy

Storage Technologies

- **Magnetic Disks**

- **Store on magnetic medium**
- **Electromechanical access**

- **Nonvolatile (Flash) Memory**



Close-up image of V-NAND flash array

- **Store as persistent charge**
- **Implemented with 3-D structure**
  - 100+ levels of cells
  - 3-4 bits data per cell

What's Inside A Disk Drive?

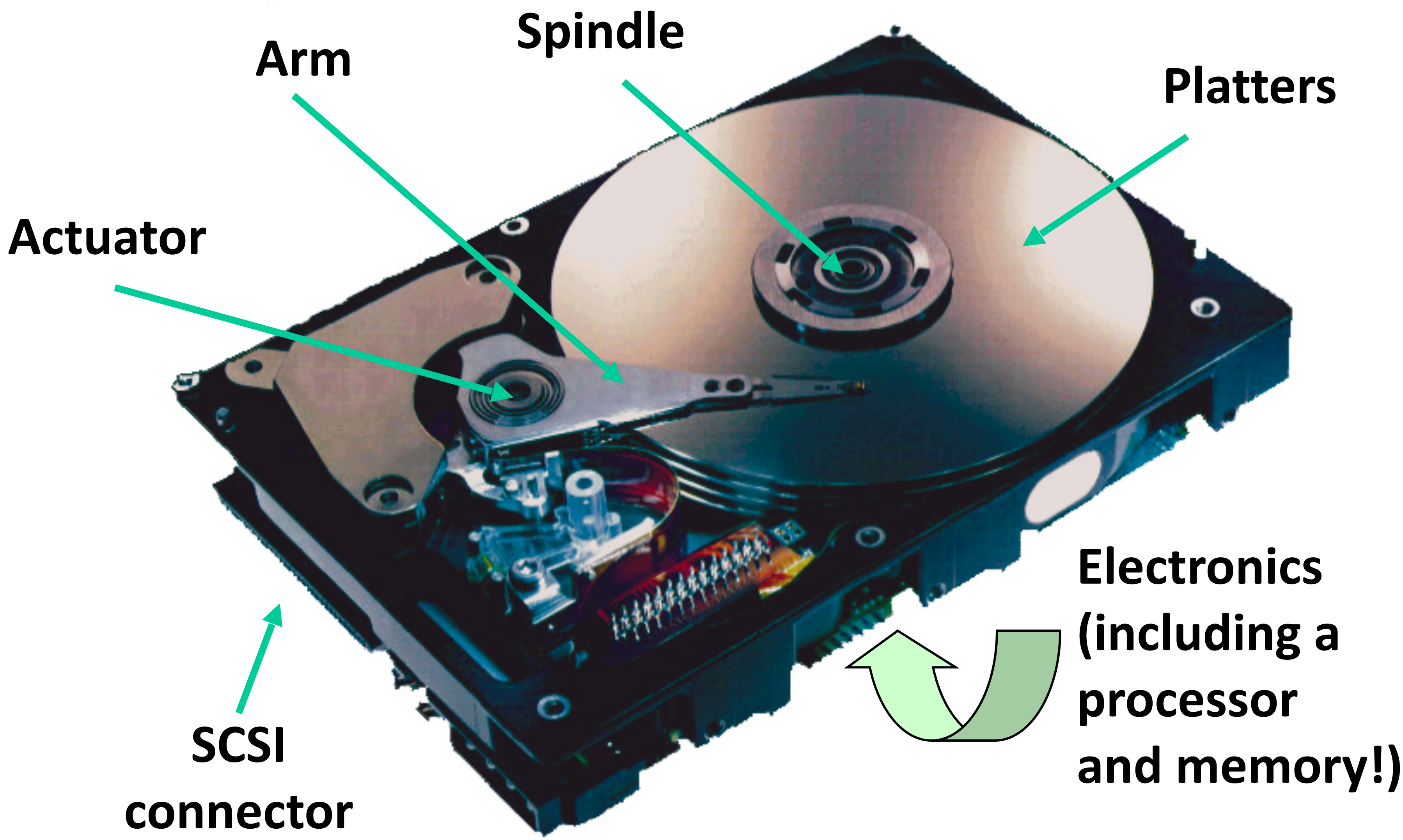
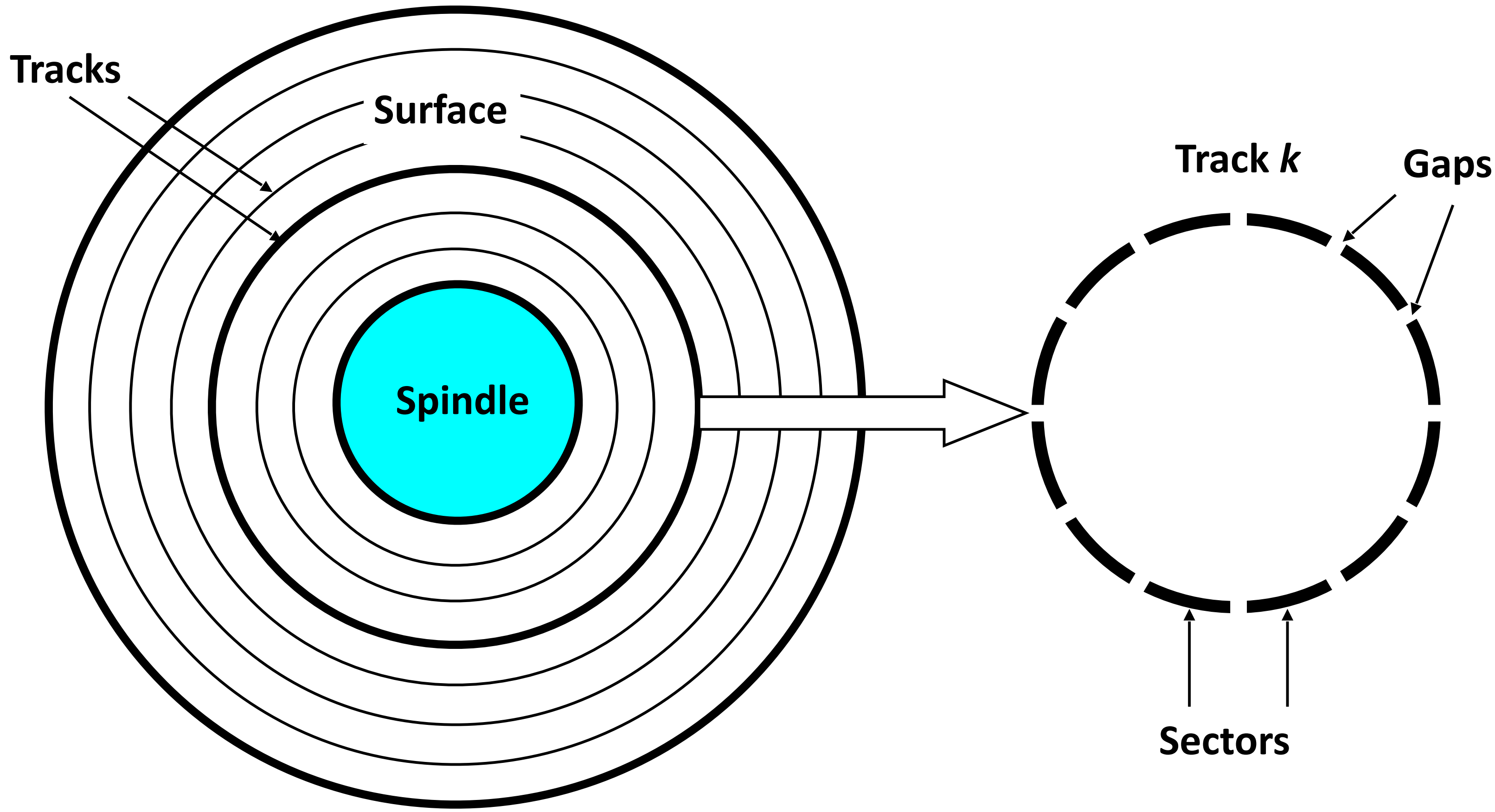


Image courtesy of Seagate Technology

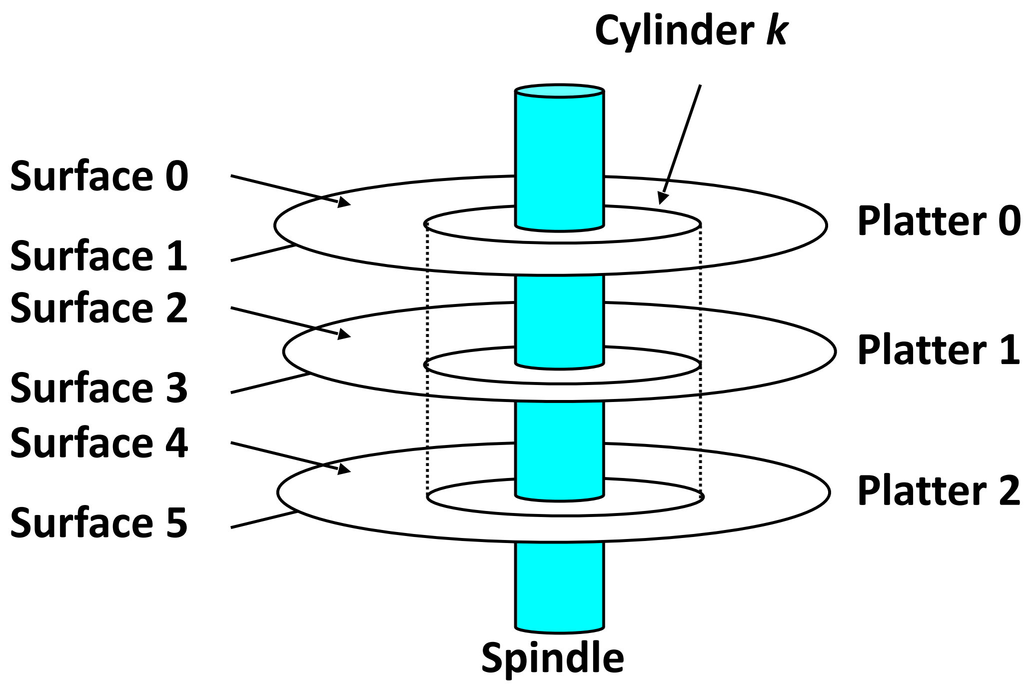
Disk Geometry

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



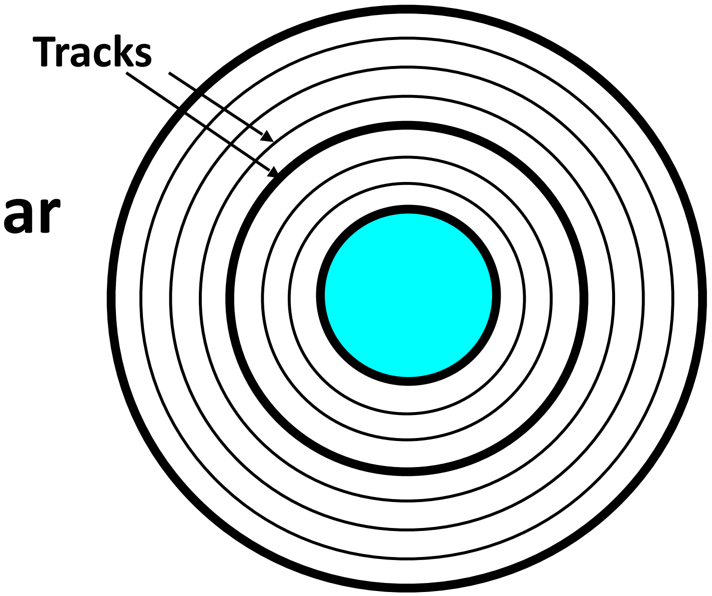
Disk Geometry (Multiple-Platter View)

- Aligned tracks form a cylinder.



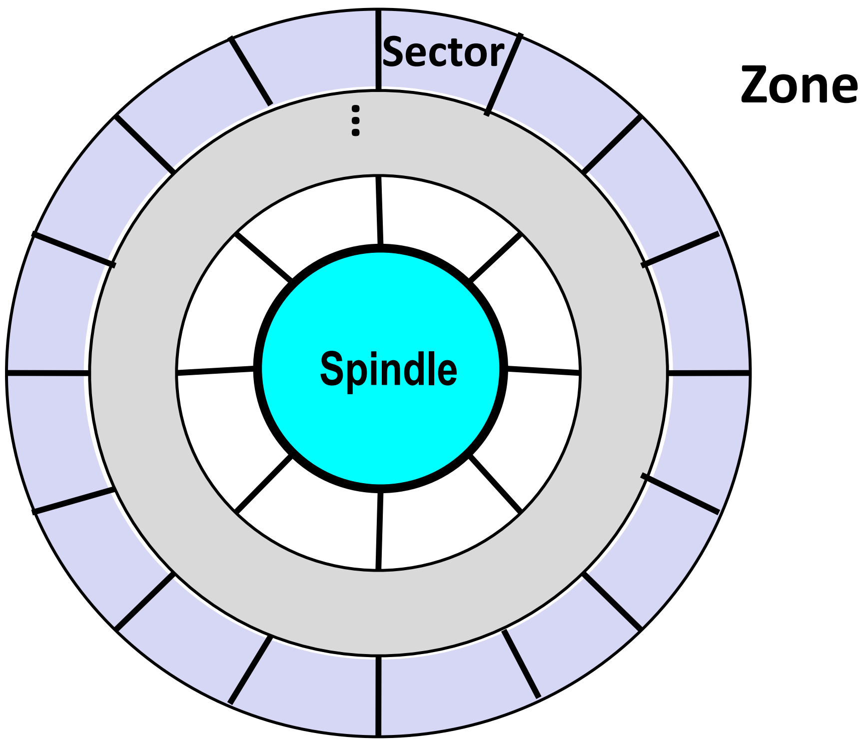
Disk Capacity

- **Capacity: maximum number of bits that can be stored.**
  - Vendors express capacity in units of **gigabytes (GB)** or **terabytes (TB)**, where 1 GB = 10^9 Bytes and 1 TB = 10^{12} Bytes
- **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 (from center).
  - Areal density (bits/in^2): product of recording and track density.
- **Capacity is doubling every couple of year**



Recording zones

- **Disks partition tracks into disjoint subsets called recording zones**
  - Each zone consists of a a **contiguous** collection of cylinders
  - **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

Capacity = (# bytes/sector) x (avg. # sectors/track) x
(# tracks/surface) x (# surfaces/platter) x
(# platters/disk)

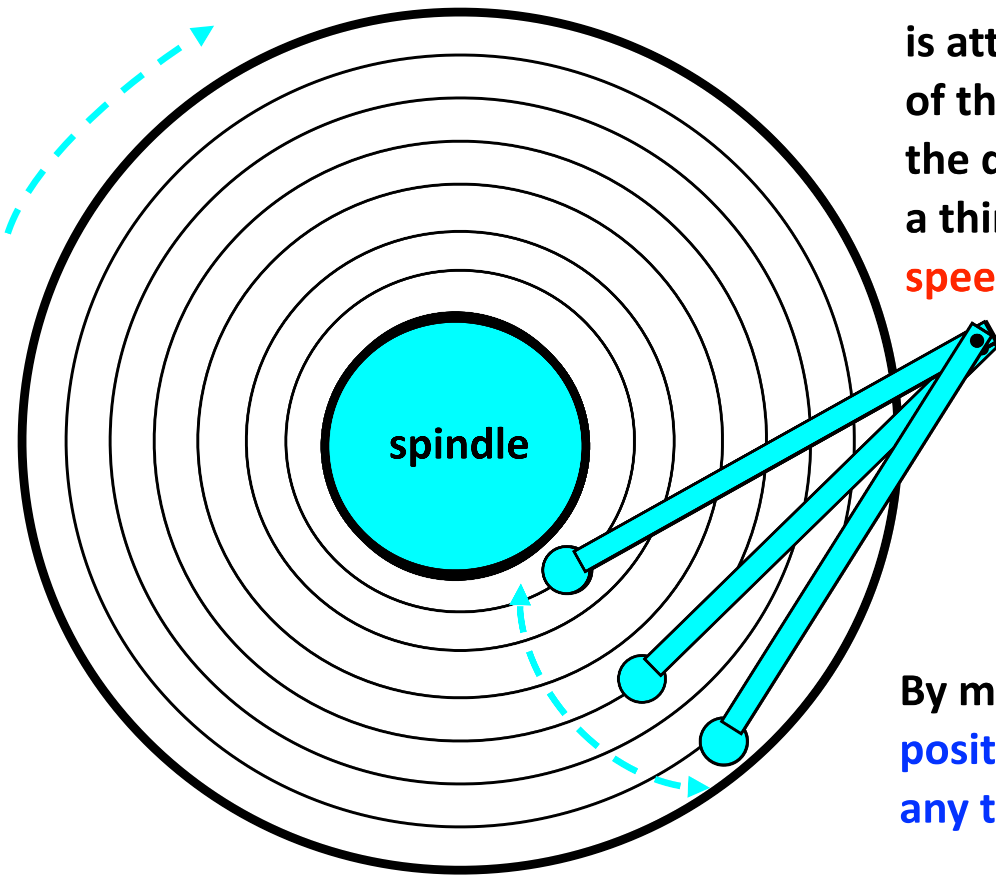
Example:

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

Capacity = 512 x 300 x 20,000 x 2 x 5
= 30,720,000,000
= 30.72 GB

Disk Operation (Single-Platter View)

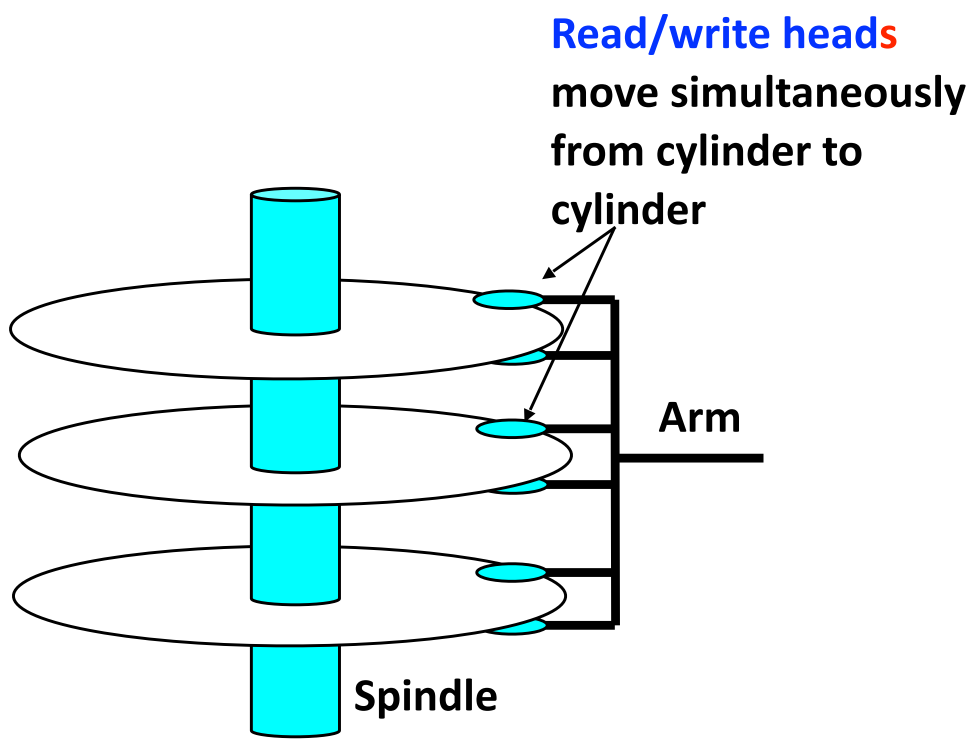
The disk surface spins at a fixed rotational rate



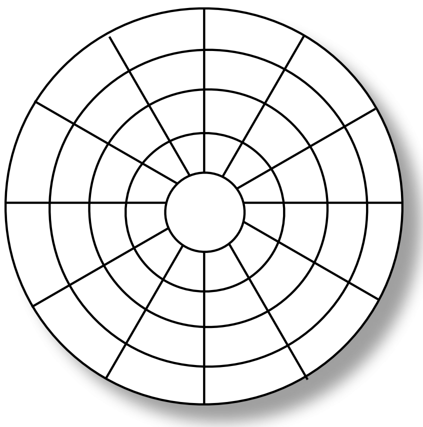
The read/write *head* is attached to the end of the *arm* and flies over the disk surface on a thin cushion of air at a speed of about 80km/h.

By moving radially, the arm can position the read/write head over any track —> seek operation

Disk Operation (Multi-Platter View)



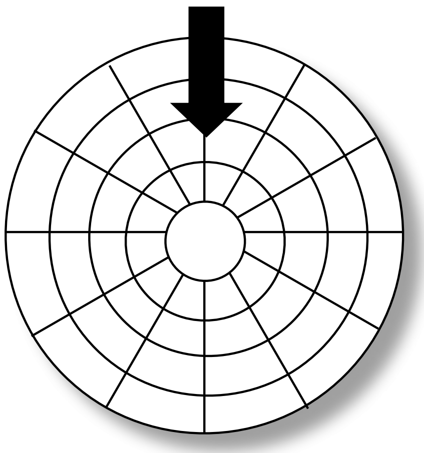
Disk Structure - top view of single platter



Surface organized into tracks

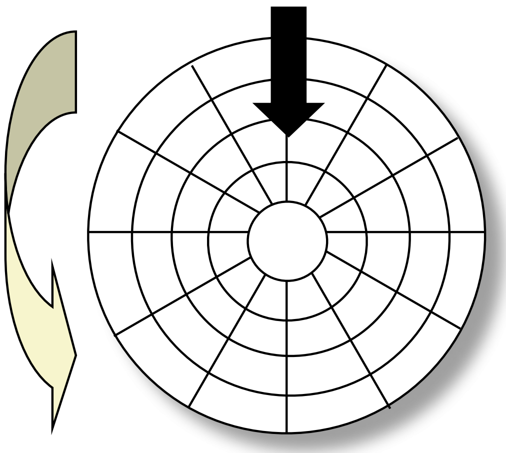
Tracks divided into sectors

Disk Access



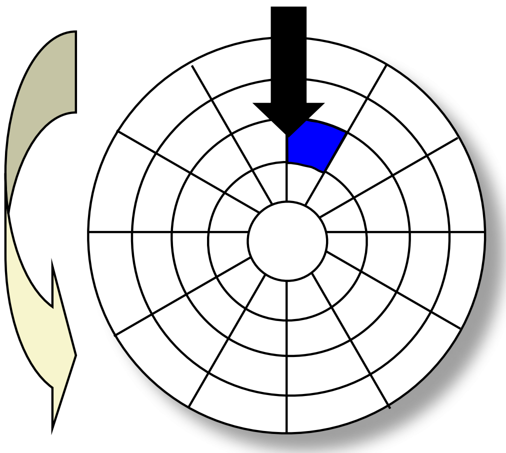
Head in position above a track

Disk Access



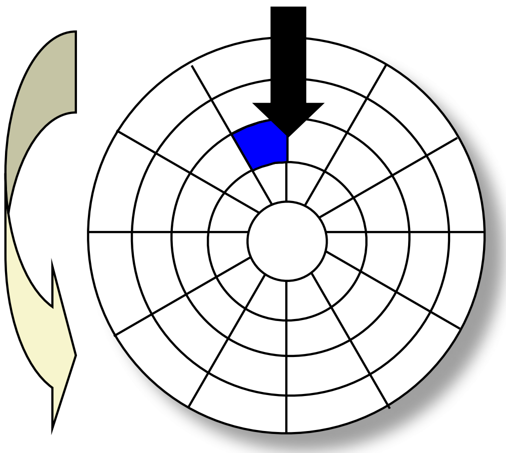
Rotation is counter-clockwise

Disk Access – Read



About to read blue sector

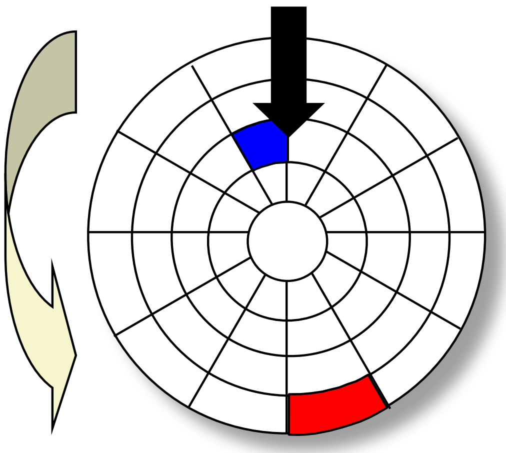
Disk Access – Read



After **BLUE** read

After reading blue sector

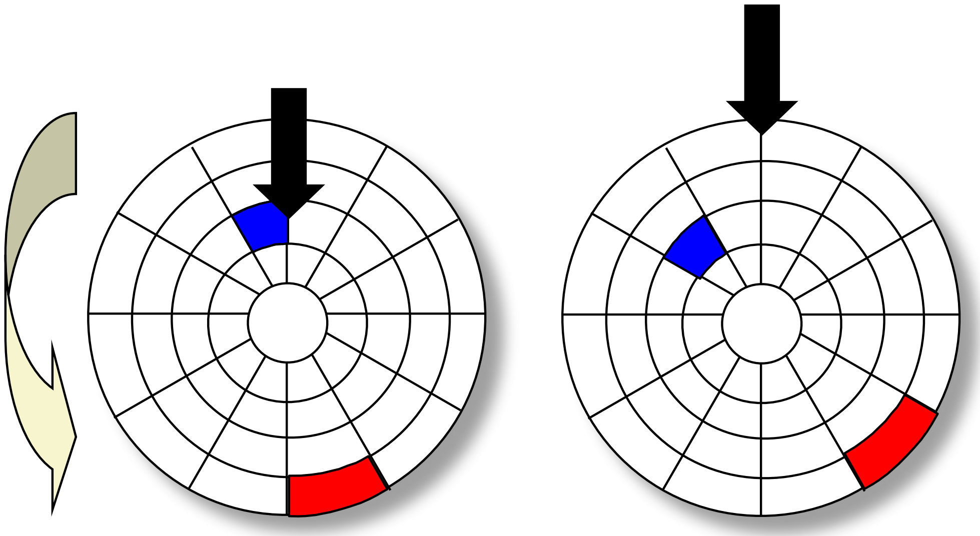
Disk Access – Read



After **BLUE** read

Red request scheduled next

Disk Access – Seek

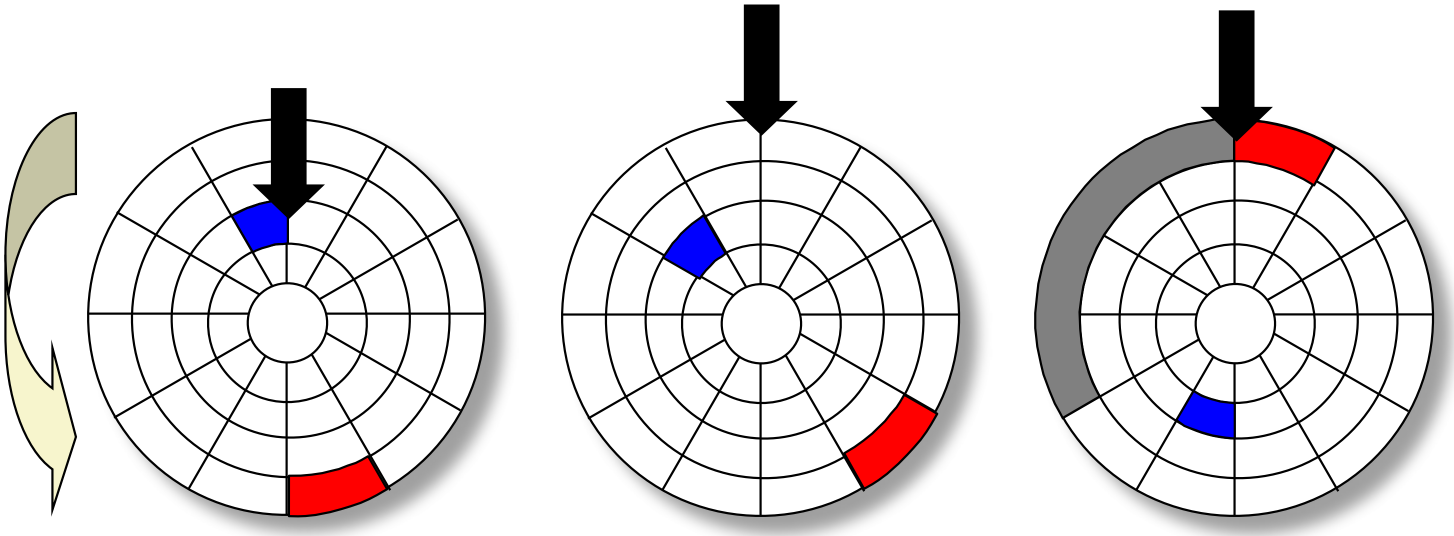


After **BLUE** read

Seek for **RED**

Seek to red's track

Disk Access – Rotational Latency



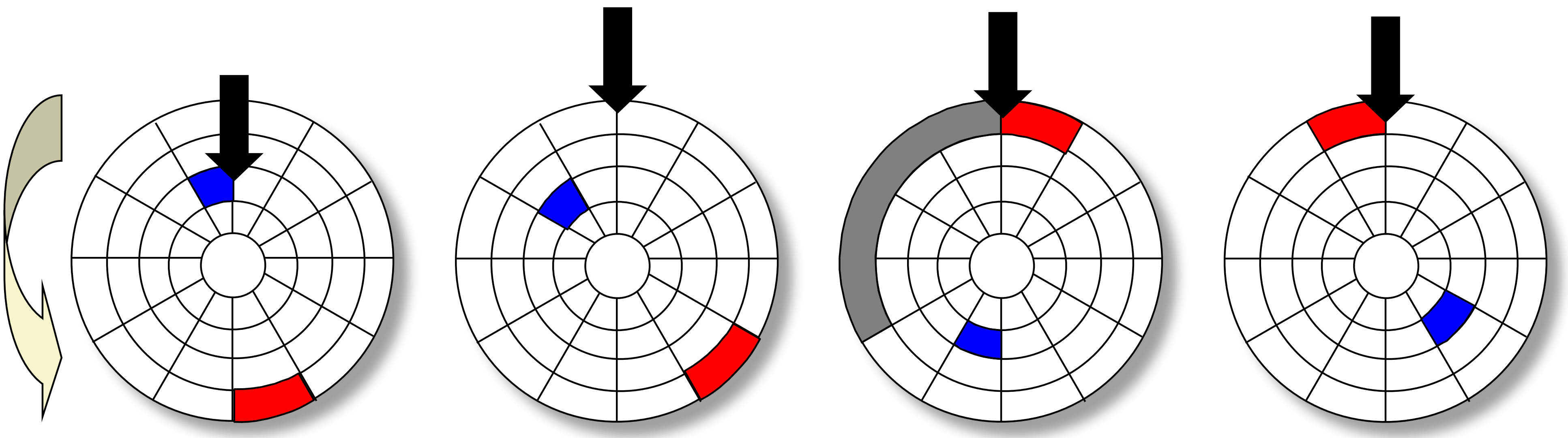
After **BLUE** read

Seek for **RED**

Rotational latency

Wait for red sector to rotate around

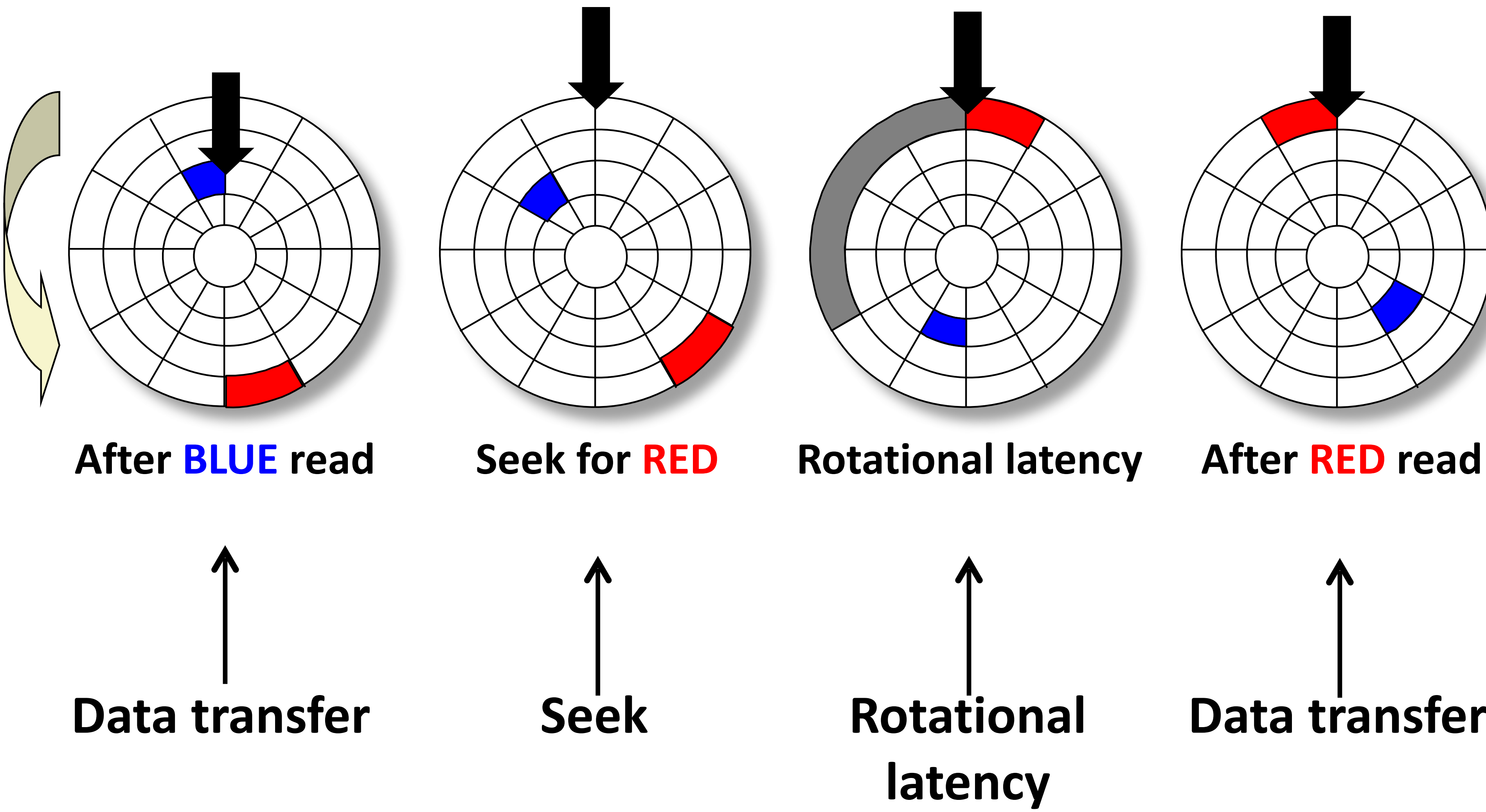
Disk Access – Read



After **BLUE** read | Seek for **RED** | Rotational latency | After **RED** read

Complete read of red

Disk Access – Service Time Components



Disk Access Time

- **Average time to access some target sector approximated by:**
  - $T_{access} = T_{avg\ seek} + T_{avg\ rotation} + T_{avg\ transfer}$
- **Seek time ($T_{avg\ seek}$)**
  - Time to position heads over cylinder containing target sector.
  - **Typical** $T_{avg\ seek}$ is **3—9 ms**
- **Rotational latency ($T_{avg\ rotation}$)**
  - Time waiting for first bit of target sector to pass under r/w head.
  - $T_{avg\ rotation} = 1/2 \times \mathbf{T_{max\ rotation}} = 1/2 \times 1/\text{RPMs} \times 60\ \text{sec}/1\ \text{min}$
  - **Typical** rotational rate = **7,200 RPMs (Rotation or Revolutions Per Minute)**
- **Transfer time ($T_{avg\ transfer}$)**
  - Time to read the bits in the target sector.
  - $T_{avg\ transfer} = 1/\text{RPM} \times \mathbf{1/(avg\ \#\ sectors/track)} \times 60\ \text{secs}/1\ \text{min}$

time for one rotation (in minutes)

fraction of a rotation to be read

Disk Access Time Example

- **Given:**
  - Rotational rate = 7,200 RPM
  - Average seek time = 9 ms.
  - Avg # sectors/track = 400.
- **Derived:**
  - $T_{avg\ rotation}$ =
  - $T_{avg\ transfer}$ =
  - T_{access} =

- **Average time to access some target sector approximated by:**
  - $T_{access} = T_{avg\ seek} + T_{avg\ rotation} + T_{avg\ transfer}$
- **Seek time ($T_{avg\ seek}$)**
  - Time to position heads over cylinder containing target sector.
  - Typical $T_{avg\ seek}$ is 3—9 ms
- **Rotational latency ($T_{avg\ rotation}$)**
  - Time waiting for first bit of target sector to pass under r/w head.
  - $T_{avg\ rotation}$ = 1/2 x 1/RPMs x 60 sec/1 min
  - Typical $T_{avg\ rotation}$ = 7,200 RPMs
- **Transfer time ($T_{avg\ transfer}$)**
  - Time to read the bits in the target sector.
  - $T_{avg\ transfer}$ = 1/RPM x 1/(avg # sectors/track) x 60 secs/1 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 x (60 secs/7200 RPM) x 1000 ms/sec = 4 ms
- $T_{\text{avg transfer}}$ = 60/7200 x 1/400 x 1000 ms/sec = 0.02 ms
- T_{access} = 9 ms + 4 ms + 0.02 ms = 13.02 ms

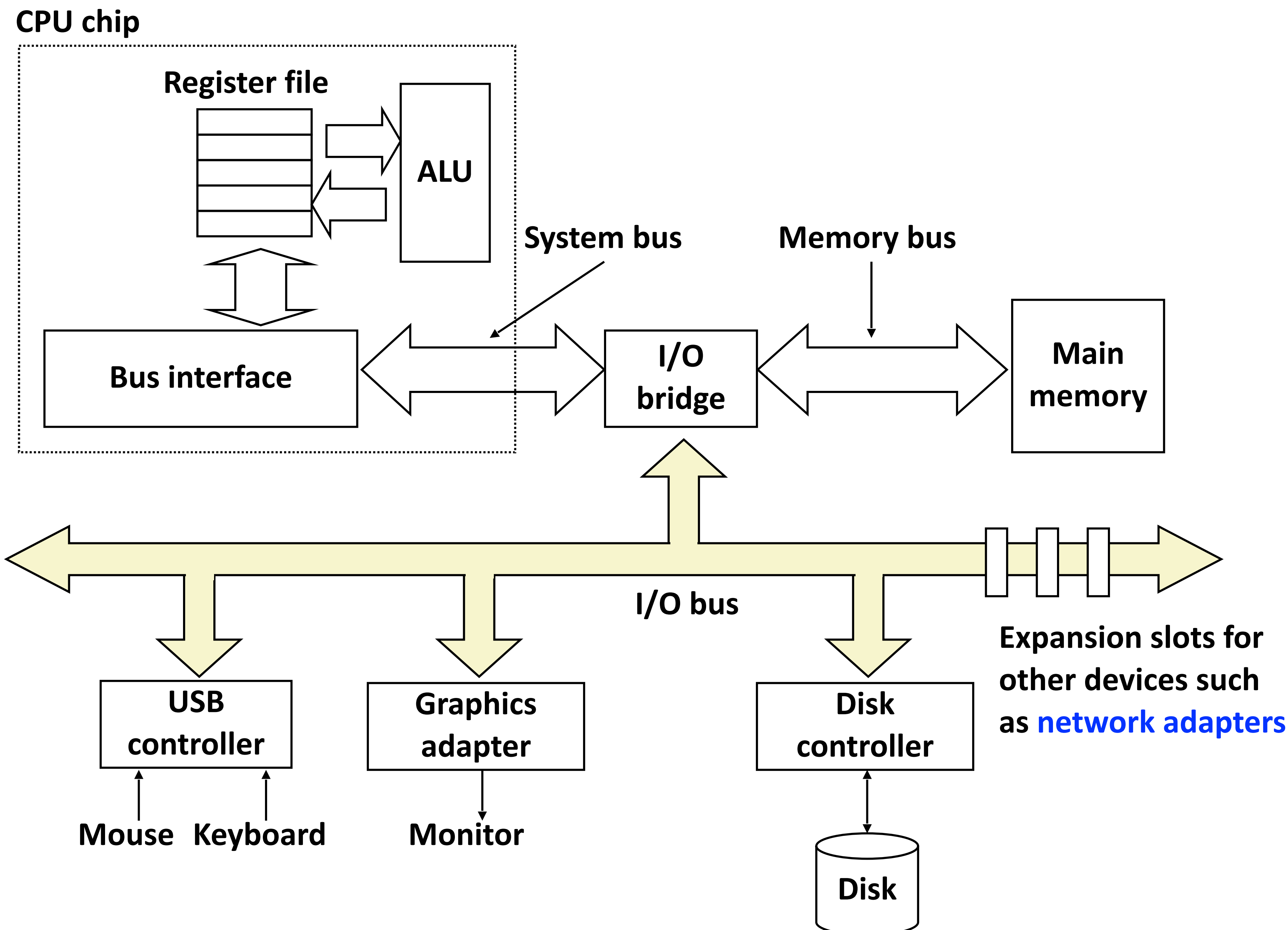
Important points:

- 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 to the **Operation System (OS)**:
  - The set of available sectors is modeled as a sequence of B-sized logical blocks (0, 1, 2, ..., B-1)
- **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".

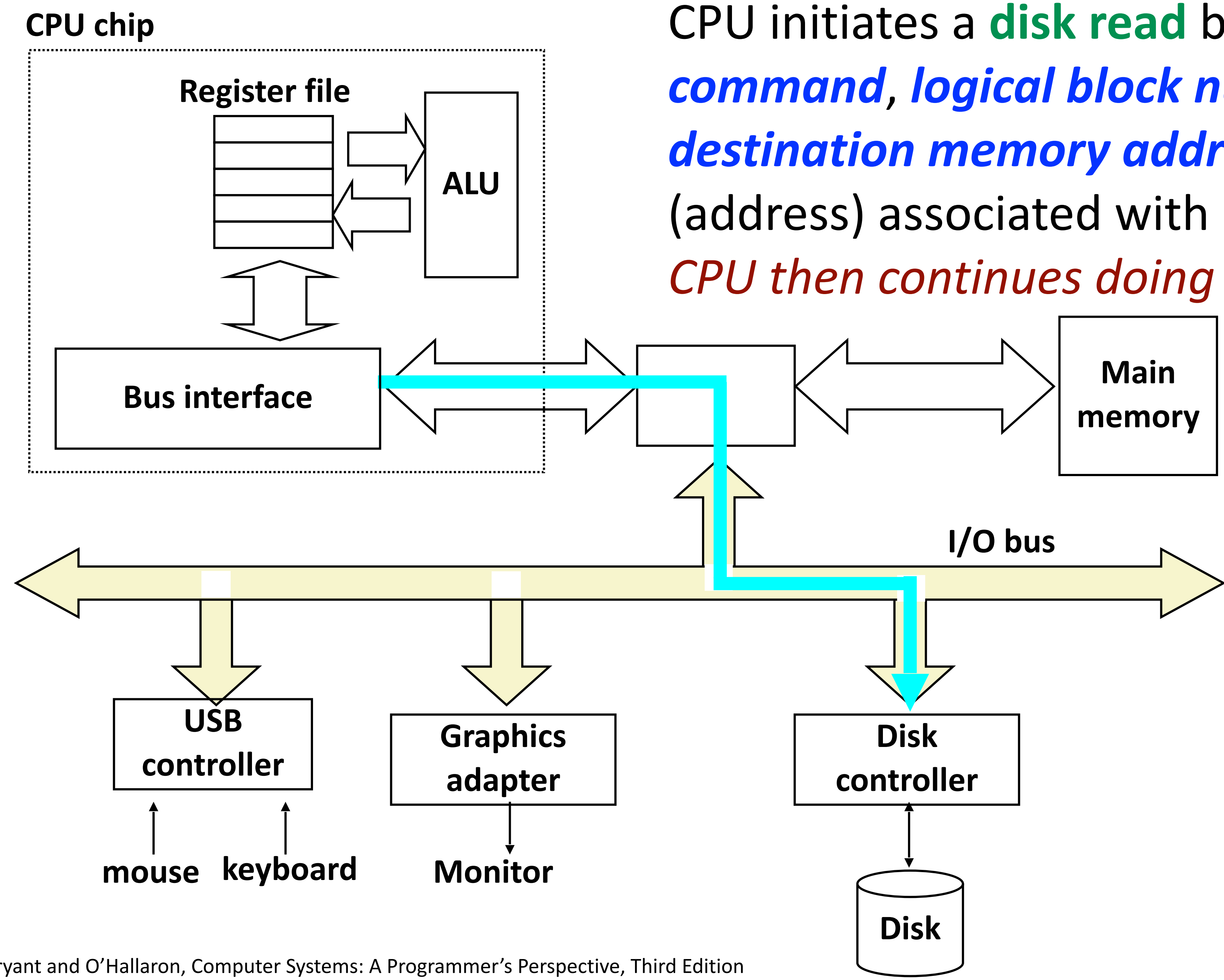
I/O Bus


CPU chip
Register file
ALU
System bus
Memory bus
Bus interface
I/O
bridge
Main
memory
I/O bus
Expansion slots for
other devices such
as network adapters.
USB
controller
Graphics
adapter
Disk
controller
Mouse
Keyboard
Monitor
Disk

Reading a Disk Sector (1)

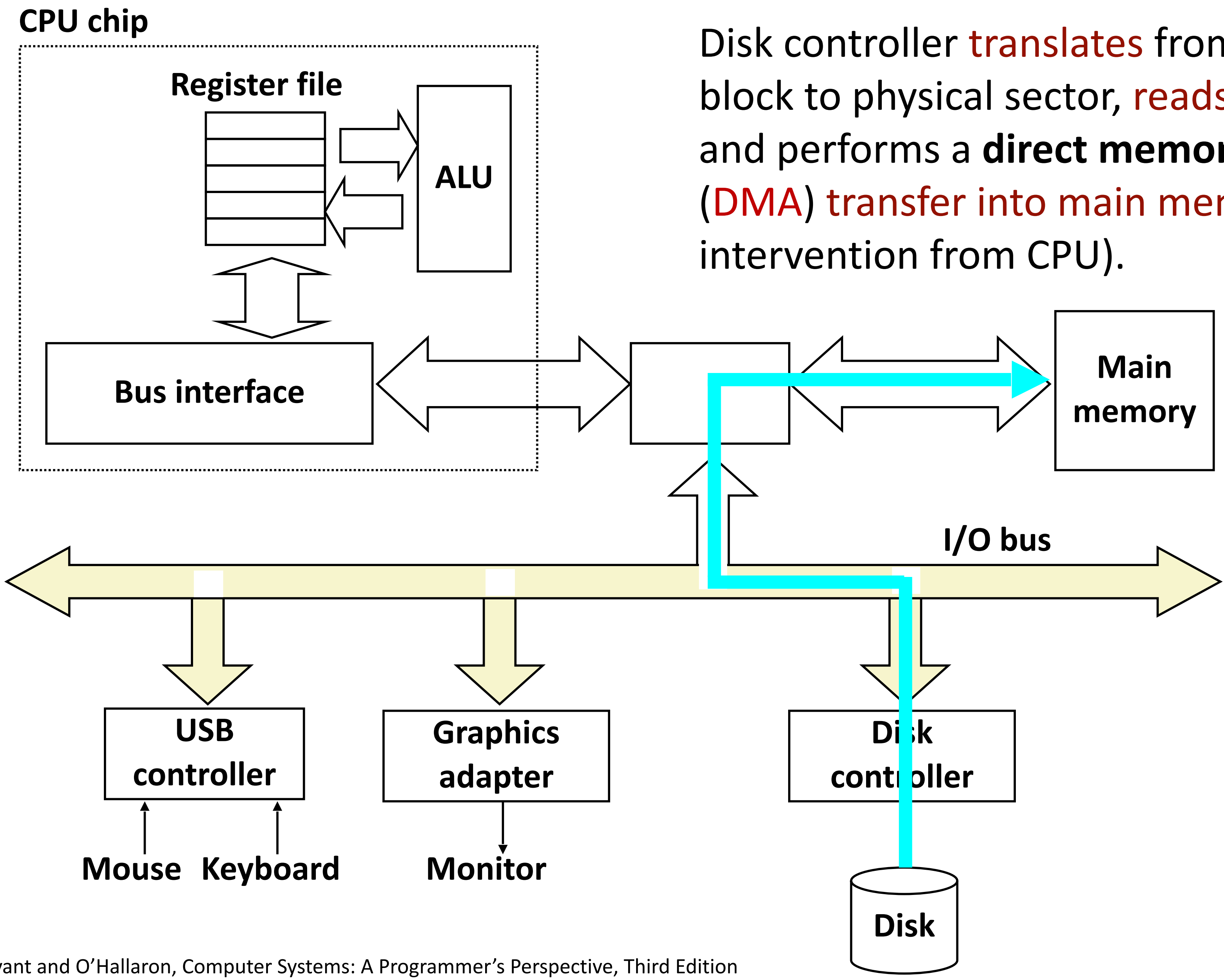
CPU initiates a **disk read** by writing a ***command***, ***logical block number***, and ***destination memory address*** to a port (address) associated with disk controller.

CPU then continues doing other stuff

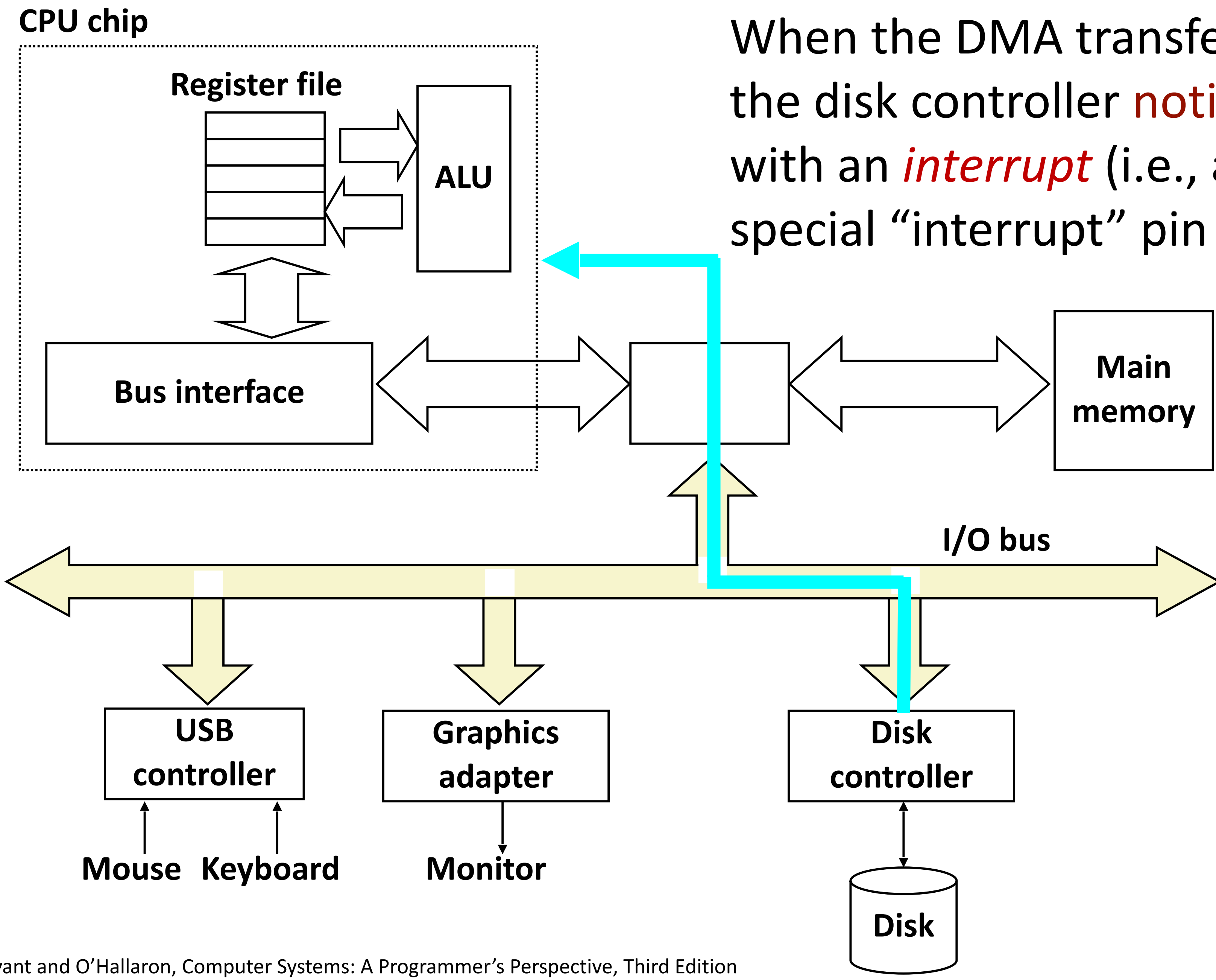


Reading a Disk Sector (2)

Disk controller translates from logical block to physical sector, reads the sector and performs a **direct memory access** (DMA) transfer into main memory (no intervention from CPU).

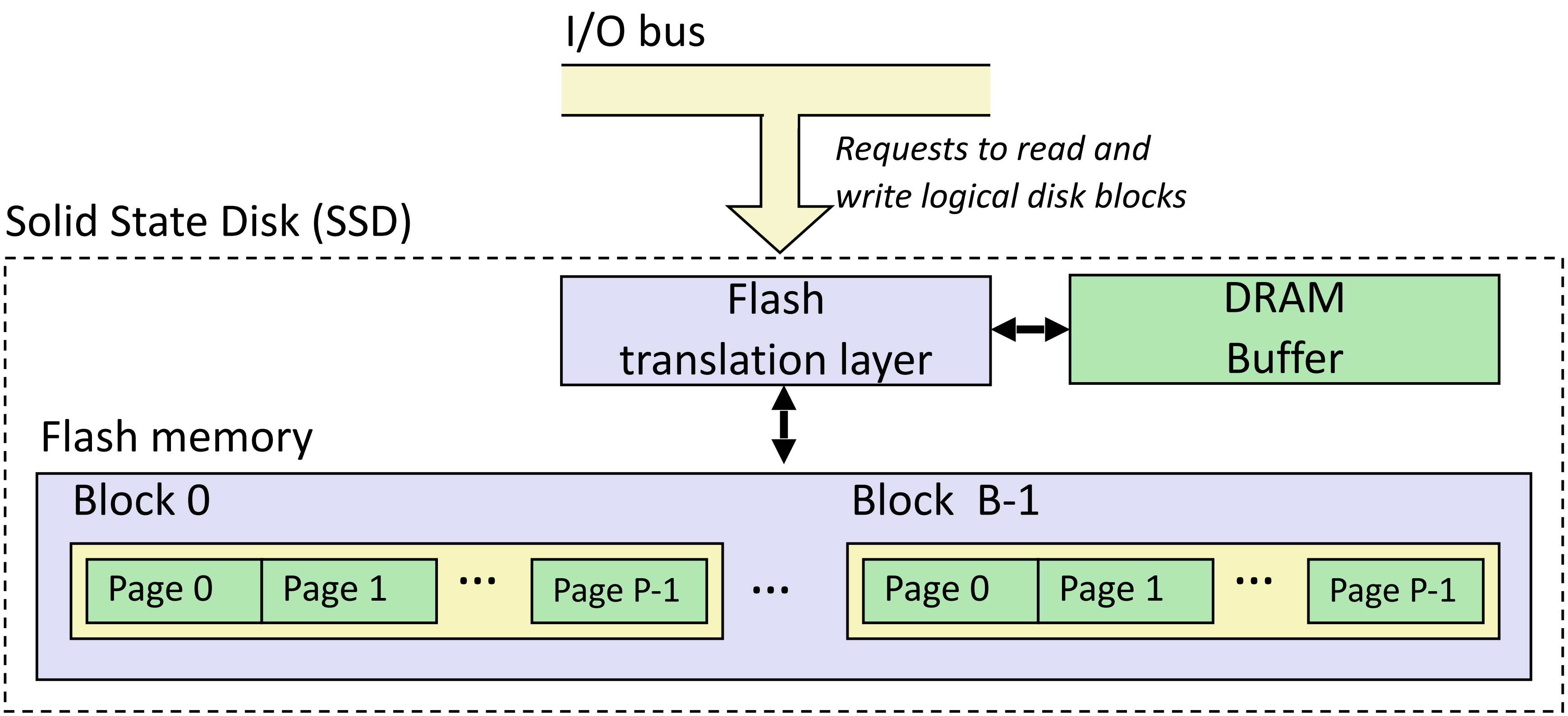


Reading a Disk Sector (3)



When the DMA transfer completes, the disk controller notifies the CPU with an *interrupt* (i.e., asserts a special "interrupt" pin on the CPU).

Solid State Disks (SSDs)



- **Flash translation layer**: plays **same** role as **disk controller** in rotating disks
- Blocks: 32 to 128 pages. Each page: 512KB to 4KB,
- Data **read/written in units of pages**.
- **Page can be written only after its block has been erased**.
- A block **wears out** after about **10,000 repeated writes**.

SSD Performance Characteristics

- **Benchmark of Samsung 970 EVO Plus**

https://ssd.userbenchmark.com/SpeedTest/711305/Samsung-SSD-970-EVO-Plus-250GB

Sequential read throughput	2,221 MB/s	Sequential write tput	1,912 MB/s
Random ST throughput	61.7 MB/s	Random write tput	165 MB/s
Random DQ throughput	947 MB/s	Random DQ write	1028 MB/s

- **Sequential access faster than random access**
  - Common theme in the memory hierarchy
  - **DQ** = **d**eep **q**ueue, issuing many concurrent reads (latency hurts!)
- **Random writes are tricky**
  - **Erasing a block takes a long time (~1 ms)**, but the SSD has a pool of **pre-erased blocks**
  - **Modifying a** block **page requires all other pages to be copied to new block**.
  - But the SSD has a **write cache** that it accumulates writes into…

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. Samsung 940 EVO Plus **guarantees 600 writes/byte of writes before they wear out**
  - Controller migrates data to minimize wear level
- In 2022, about 4-5 times more expensive per byte than rotating disks
  - And, relative cost will keep dropping

Where are rotating disks still used?

- Bulk storage – video, huge datasets / databases, etc.
- Cheap storage – desktops.

Storage Trends

SRAM

Metric	**1985**	**1990**	**1995**	**2000**	**2005**	**2010**	**2015**	***2015:1985***
$/**MB**	2,900	320	256	100	75	60	***320***	*116*
access (**ns**)	150	35	15	3	2	1.5	*200*	***115***

DRAM

Metric	**1985**	**1990**	**1995**	**2000**	**2005**	**2010**	**2015**	***2015:1985***
$/**MB**	880	100	30	1	0.1	0.06	**0.02**	*44,000*
access (**ns**)	200	100	70	60	50	40	20	***10***
typical size (MB)	0.256	4	16	64	2,000	8,000	16.000	*62,500*

Disk

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

CPU Clock Rates

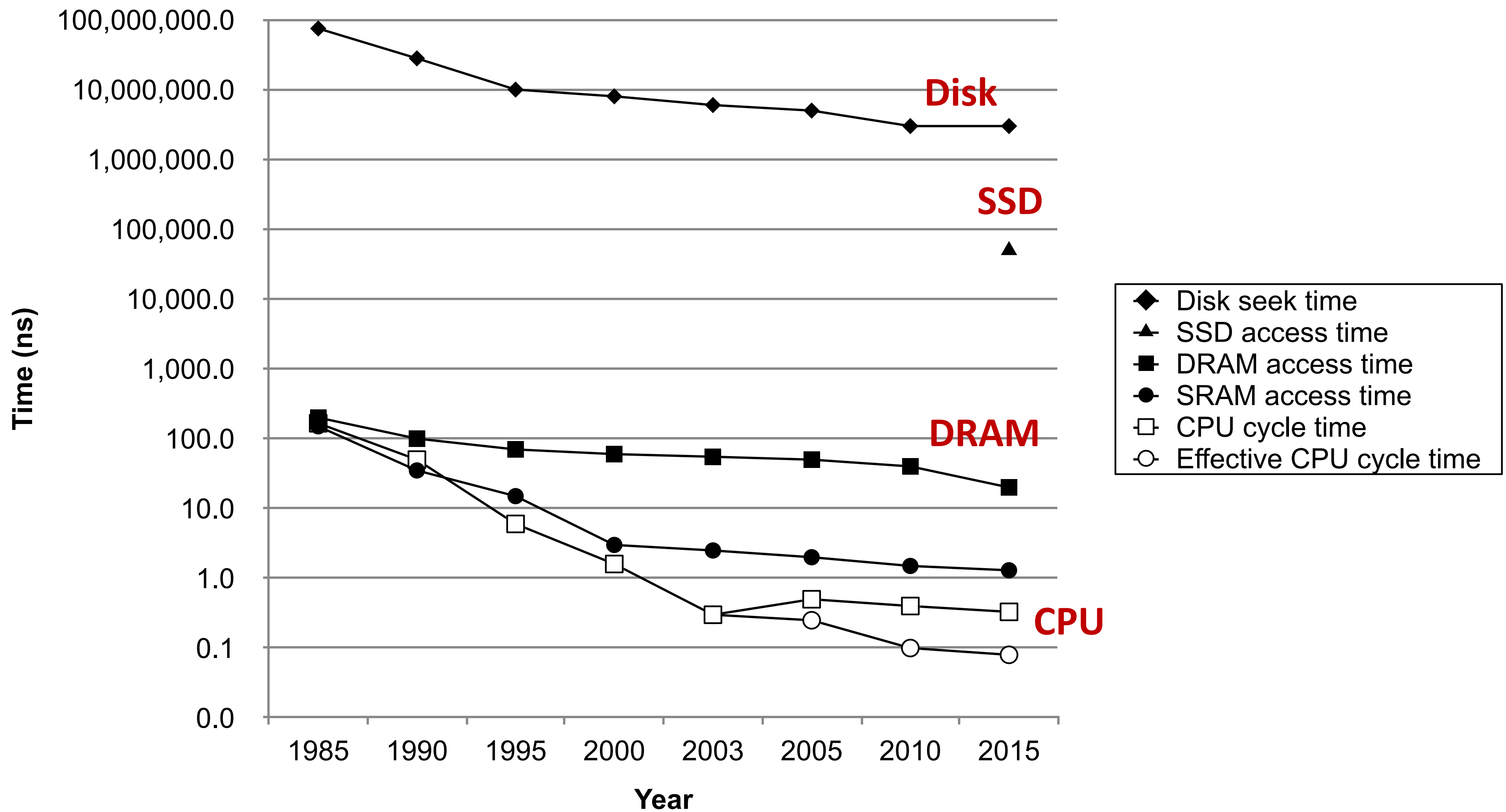
Inflection point in computer history when designers hit the "Power Wall"

	1985	1990	1995	2003	2005	2010	2015	*2015:1985*
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

The CPU-Memory Gap

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



The Memory Hierarchy

- The memory Abstraction
- DRAM : main memory building block
- Storage technologies and trends
- Locality of reference
- The memory hierarchy

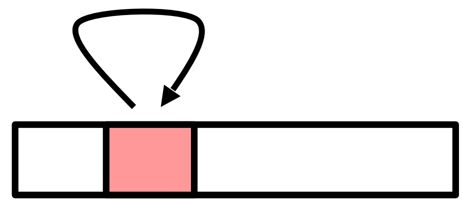
Locality to the Rescue!

The key to bridging this CPU-Memory gap is an important property of computer programs known as locality.

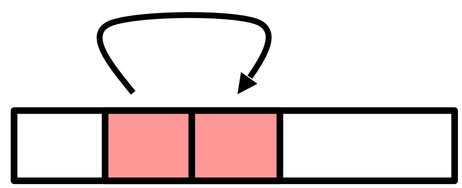
Locality

- **Principle of Locality:** **Many 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 Example

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

Spatial or Temporal Locality?

- **Data references**
  - Reference array elements in succession (stride-1 reference pattern). — **spatial**
  - Reference variable `sum` each iteration. — **temporal**
- **Instruction references**
  - Reference instructions in sequence. — **spatial**
  - Cycle through loop repeatedly. — **temporal**

Qualitative Estimates of Locality

- **Compilers** attempts to **improve locality** but can do just as much
- **Claim:** Being able to look at code and get a qualitative sense of its locality is a key skill for a **professional programmer.**
- **Question:** Does this function have **good locality with respect to array a**?

Hint: array layout is row-major order

Answer: yes

```
int sum_array_rows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

a [0] [0]	• • •	a [0] [N-1]	a [1] [0]	• • •	a [1] [N-1]	• • •	a [M-1] [0]	• • •	a [M-1] [N-1]

Locality Example

- **Question:** Does this function have **good locality with respect to array a**?

```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum;
}
```

Answer: no, unless…

M is very small

If **M is large**, **row** will already be **evicted from cache** when accessed second time (cache has limited size)

a [0] [0]	• • •	a [0] [N-1]	a [1] [0]	• • •	a [1] [N-1]	• • •	a [M-1] [0]	• • •	a [M-1] [N-1]

Locality Example

- **Question: Can you permute the loops so that the function scans the 3-d array a with a stride-1 reference pattern (and thus has good spatial locality)? Assume M and N are large**

```
int sum_array_3d(int a[M][N][N])
{
    int i, j, k, sum = 0;

    for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
            for (k = 0; k < M; k++)
                sum += a[k][i][j];
    return sum;
}
```

Answer: make k the outer loop and j the inner loop

The Memory Hierarchy

- The memory abstraction
- DRAM : main memory building block
- Storage technologies and trends
- Locality of reference
- **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**.

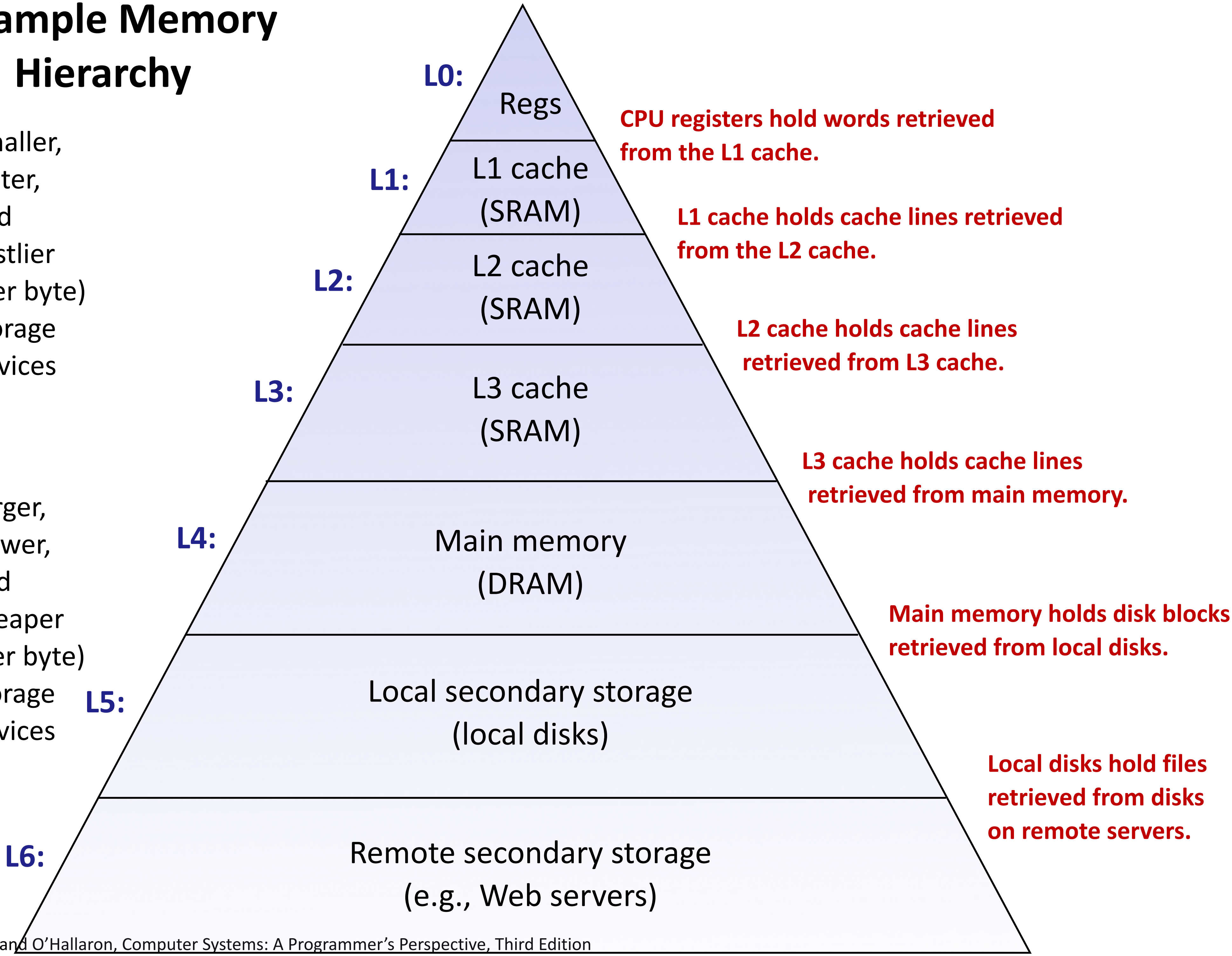
- These properties complement each other well for many types of programs.

- They suggest an **approach** for organizing memory and storage systems known as a **memory hierarchy**.

Example Memory Hierarchy

Smaller,
faster,
and
costlier
(per byte)
storage
devices

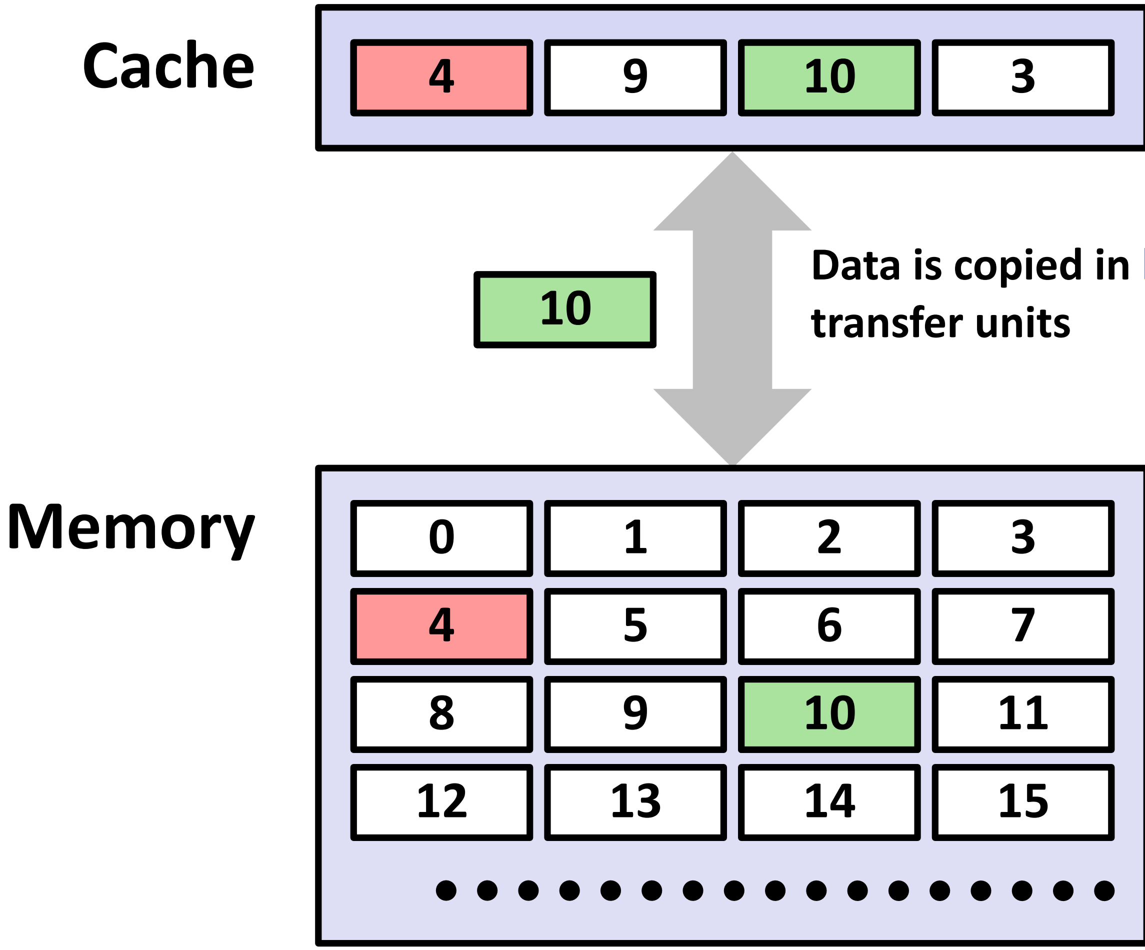
Larger,
slower,
and
cheaper
(per byte)
storage
devices



Caches

- ***Cache:*** **A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.**
- **Fundamental idea of a memory hierarchy:**
  - For each k, the **faster, smaller device at level k** serves as a cache for the **larger, slower device at level k+1**.
- **Why do memory hierarchies work?**
  - Because of **locality**: programs tend to access the data at level k more often than they access the data at level k+1.
  - Thus, the storage at level k+1 can be slower, and thus larger and cheaper per bit.
- ***Big Idea (Ideal):*** The memory hierarchy creates a large pool of storage that costs as much as the **cheap** storage near the **bottom**, but that serves data to programs at the rate of the **fast** storage near the **top**.

General Cache Concepts

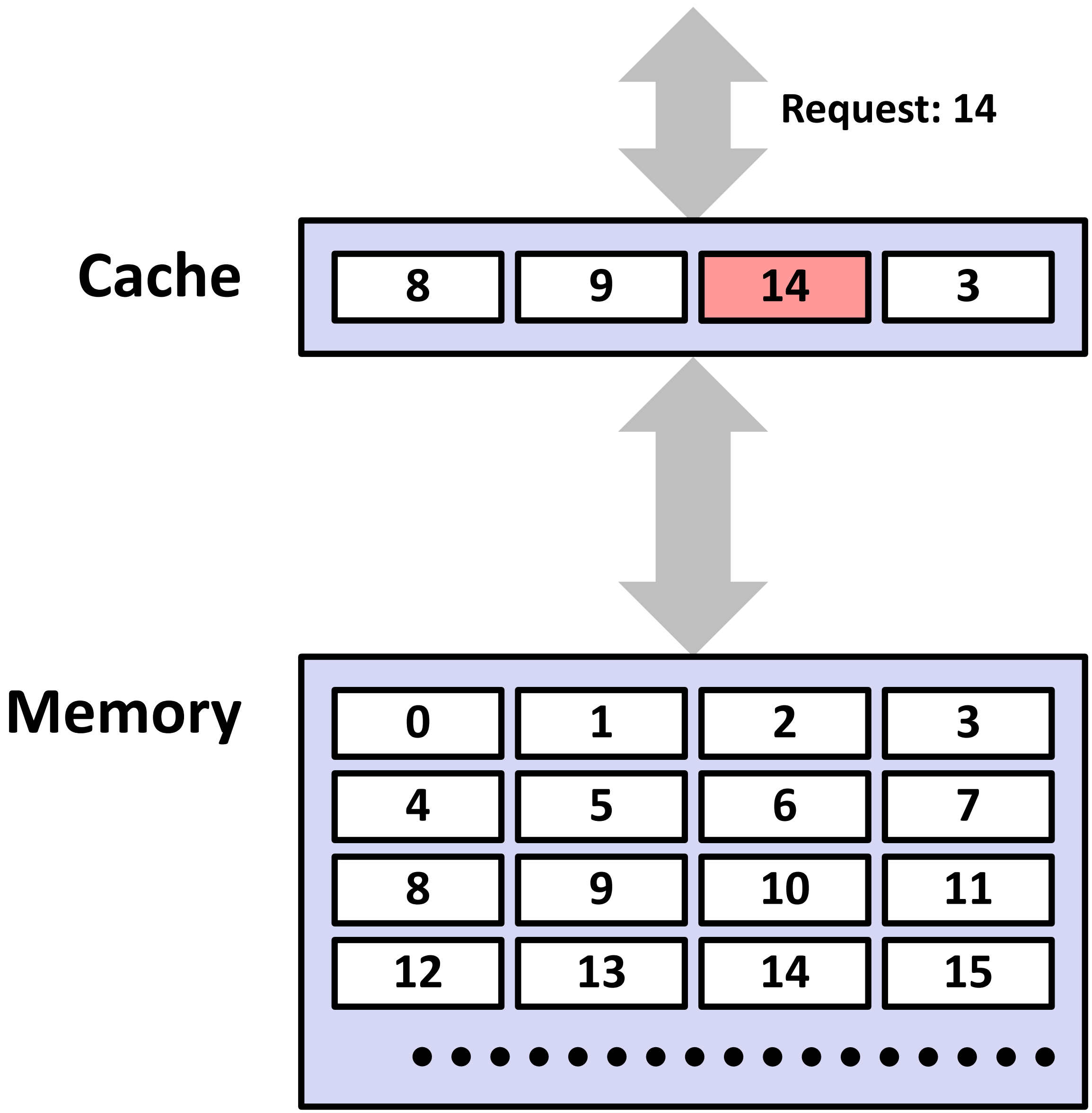


Smaller, faster, more expensive memory caches a subset of the blocks

Data is copied in block-sized transfer units

Larger, slower, cheaper memory viewed as partitioned into “blocks”

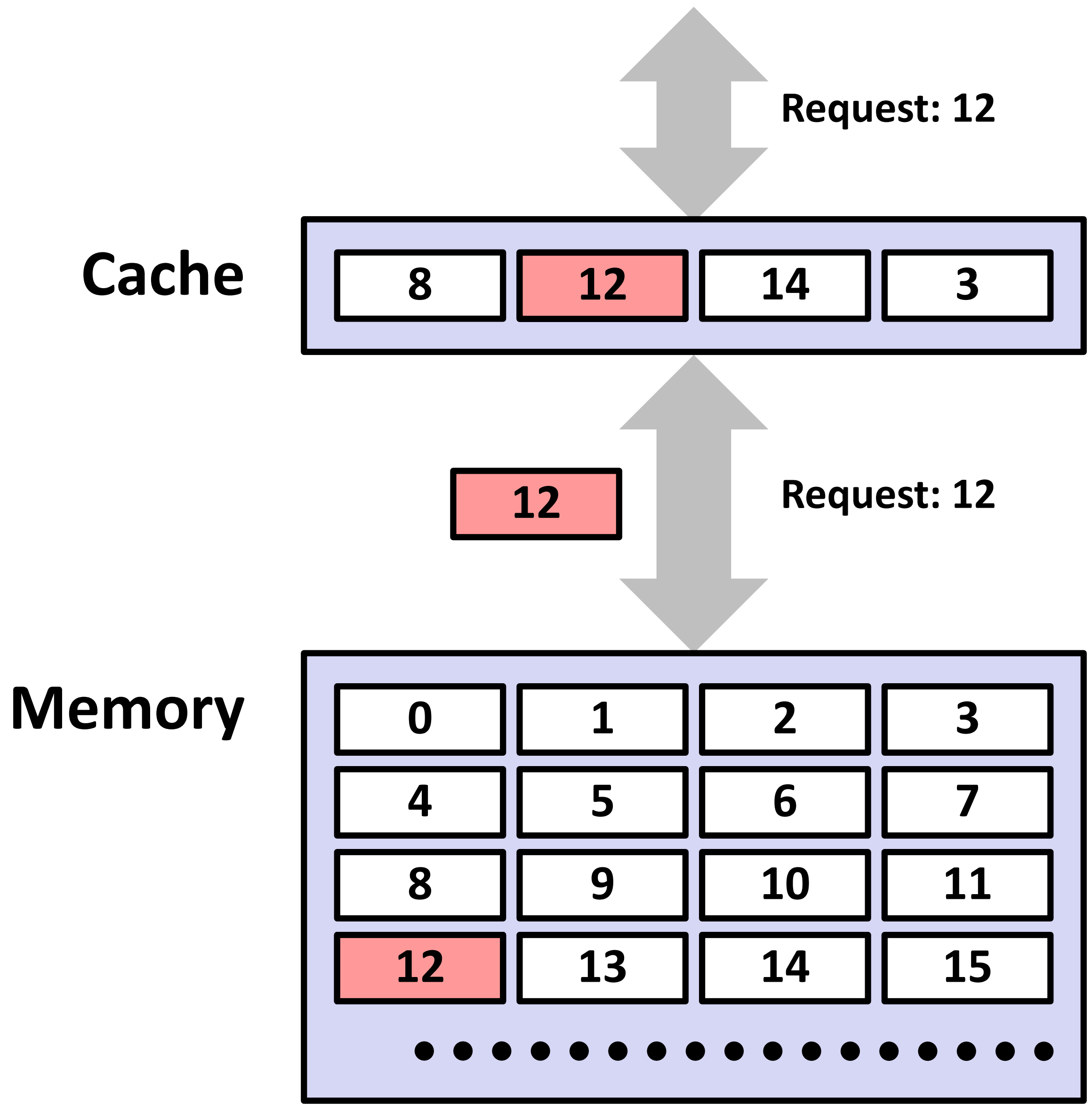
General Cache Concepts: Hit



Data in block b is needed

Block b is in cache:
Hit!

General Cache Concepts: Miss



Data in block b is needed

Block b is not in cache: Miss!

Block b is fetched from memory

Block b is stored in cache

- Placement policy: determines where b goes
- Replacement policy: determines which block gets evicted (victim)

General Caching Concepts:
3 Types of Cache Misses

- **Cold (compulsory) miss**
  - Cold misses occur because the cache starts empty and this is the **first reference to the block**.
- **Capacity miss**
  - Occurs when the **set of active cache blocks** (working set) is **larger than the cache**.
- **Conflict miss**
  - Most caches limit blocks at level k+1 to a small subset (sometimes a singleton) of the block positions at level k.
    - E.g. Block i at level k+1 must be placed in block (i mod 4) at level k.
  - Conflict misses occur when the level k cache is large enough, but **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.

Examples of Caching in the Mem. Hierarchy

Cache Type	What is Cached?	Where is it Cached?	Latency (cycles)	Managed By
Registers	4-8 byte words	CPU core	0	Compiler
TLB	Address translations	On-Chip TLB	0	Hardware MMU
L1 cache	64-byte blocks	On-Chip L1	4	Hardware
L2 cache	64-byte blocks	On-Chip L2	10	Hardware
Virtual Memory	4-KB pages	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	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.**

- **Flash memory progress outpacing all other memory and storage technologies (DRAM, SRAM, magnetic disk)**
  - Able to stack cells in three dimensions