
WELCOME
TO
15-213
14-513
18-213
18-613
15-513

The Memory Hierarchy

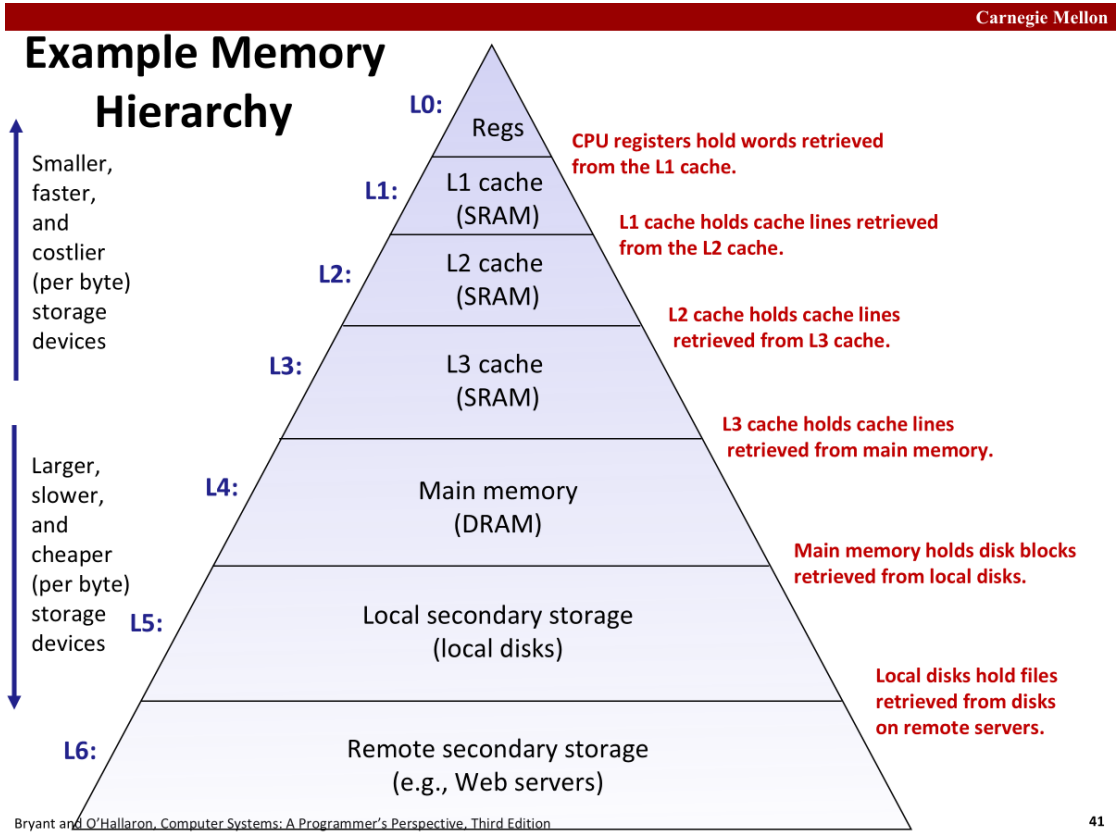
18-213/18-613: Introduction to Computer Systems
9th Lecture, September 23, 2025

Today

- **The memory abstraction** — **CSAPP 6.1.1**
- RAM : main memory building block — CSAPP 6.1.1
- Storage technologies and trends — CSAPP 6.1.2-6.1.4
- The memory hierarchy — CSAPP 6.3
- Working sets — CSAPP 6.2
- Locality of reference — CSAPP 6.2
- Caches — CSAPP 6.4-6.5

Today's Goal

- **Make the system perform almost as if all of the memory is the fastest type of memory, while the average cost per byte is as if all of the memory is the cheapest kind of memory.**



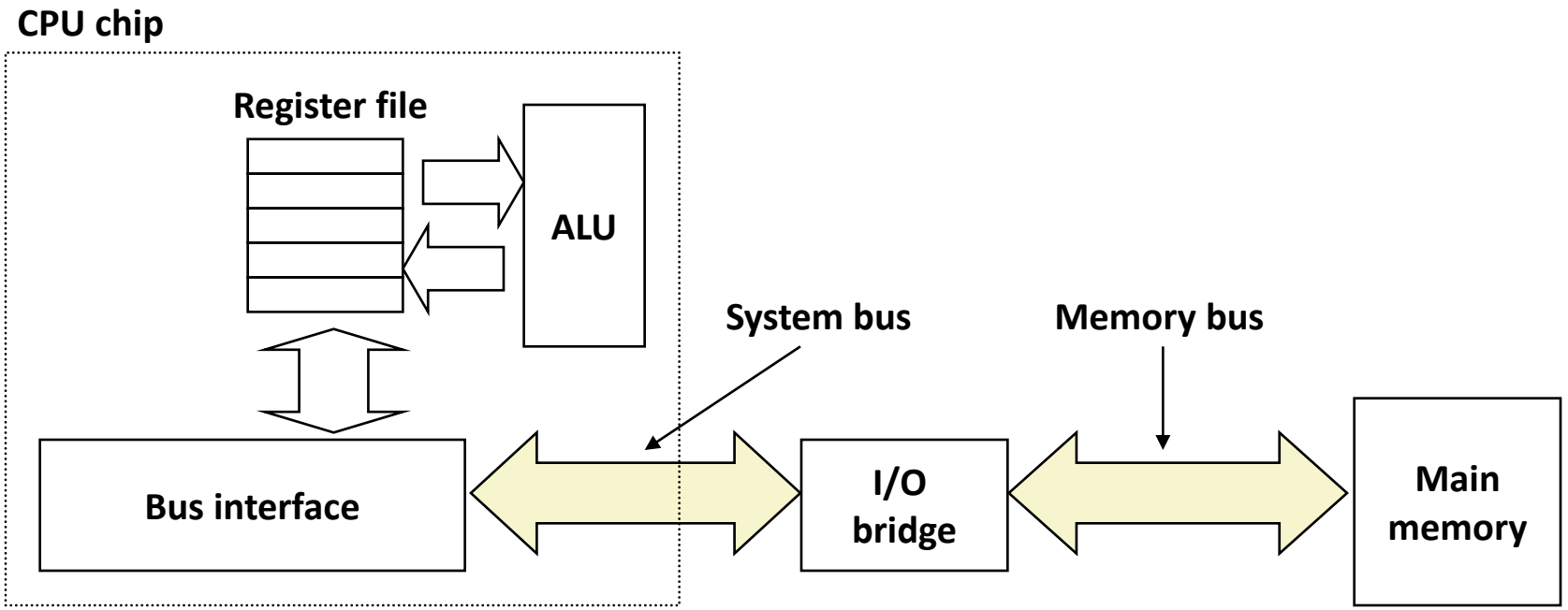
Writing & Reading Memory

- **Write**
  - Transfer data from CPU to memory
    `movq %rax, 8(%rsp)`
  - "Store" operation

- **Read**
  - Transfer data from memory to CPU
    `movq 8(%rsp), %rax`
  - "Load" operation

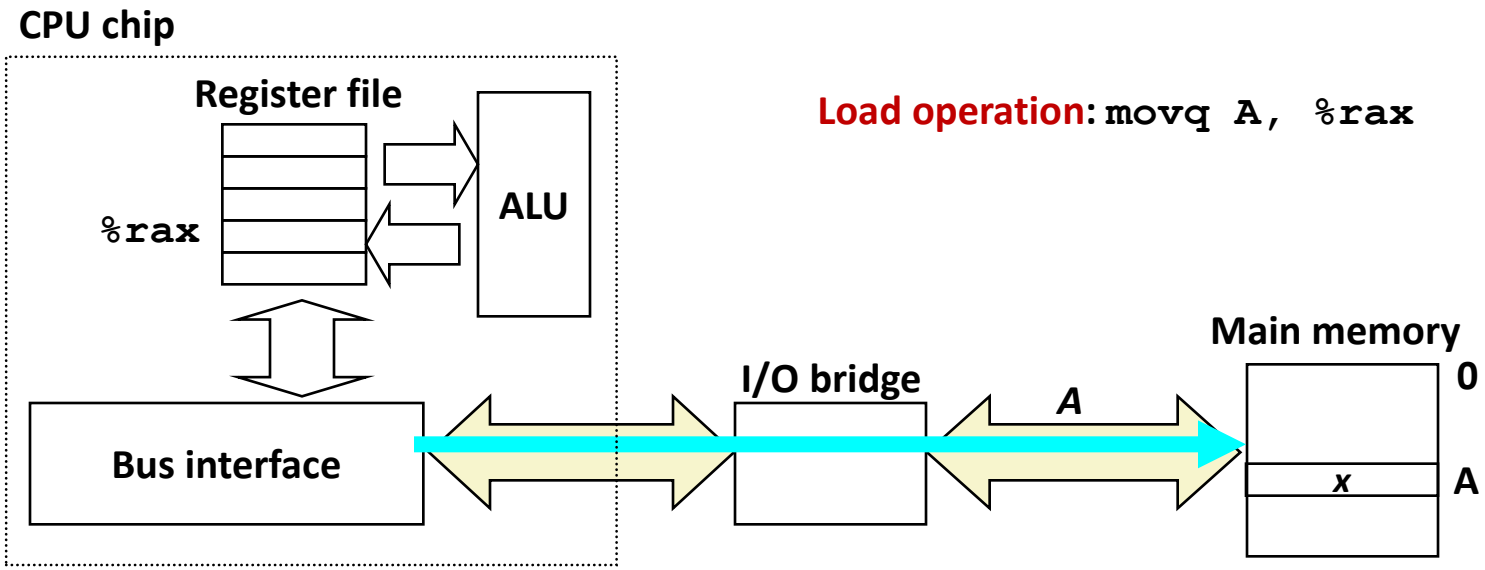
Traditional Bus Structure Connecting CPU and Memory

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



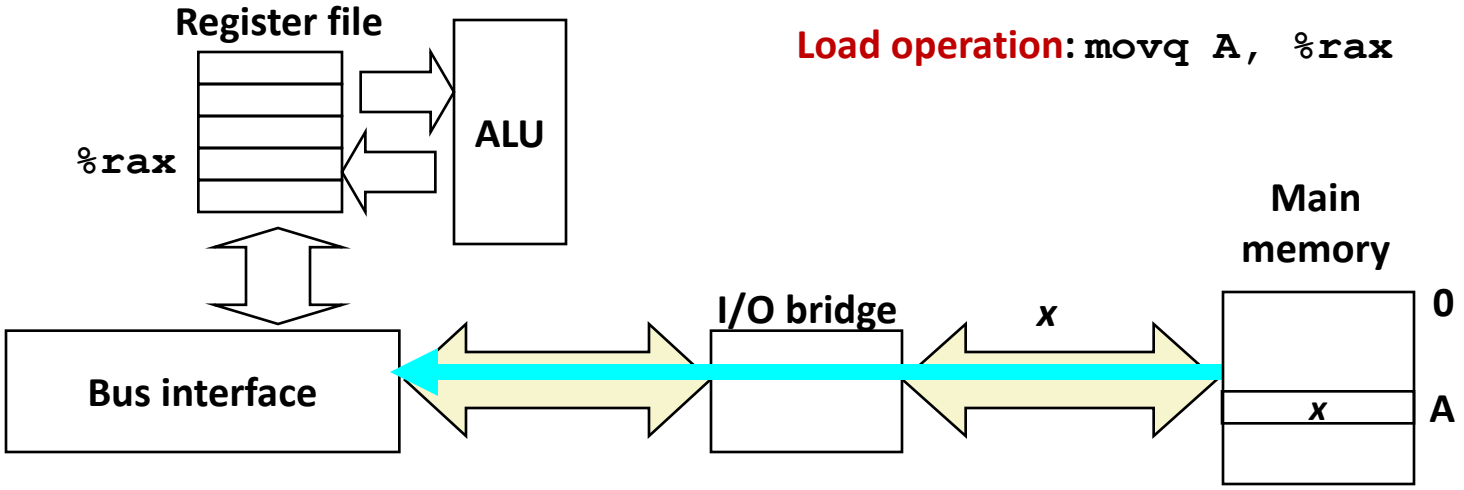
Memory Read Transaction (1)

- CPU places address A on the memory bus.



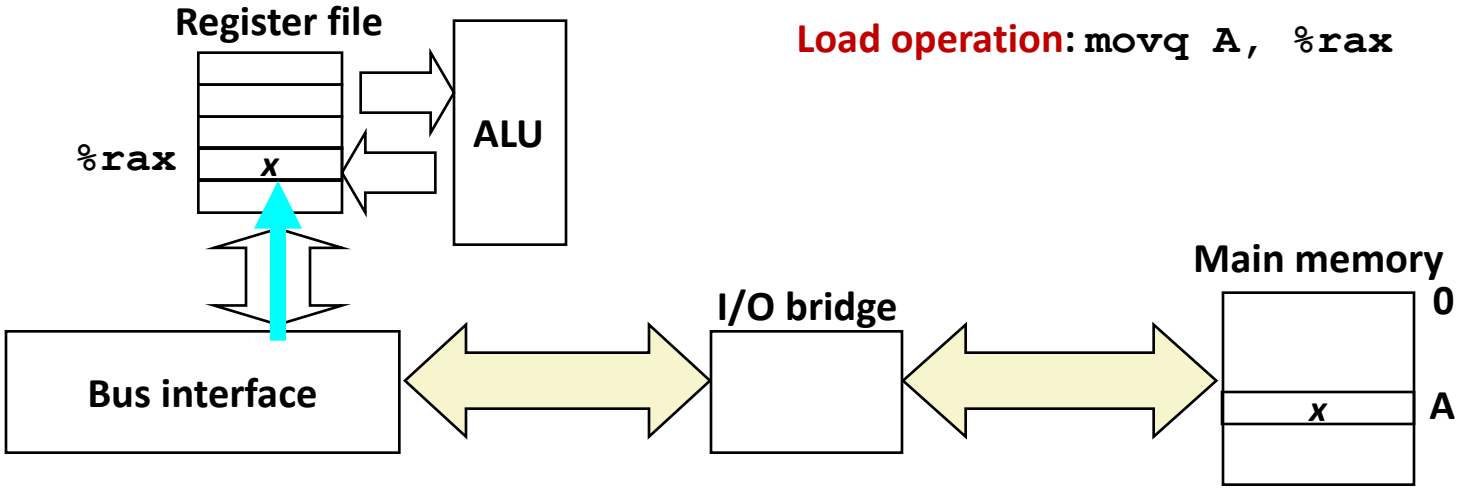
Memory Read Transaction (2)

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



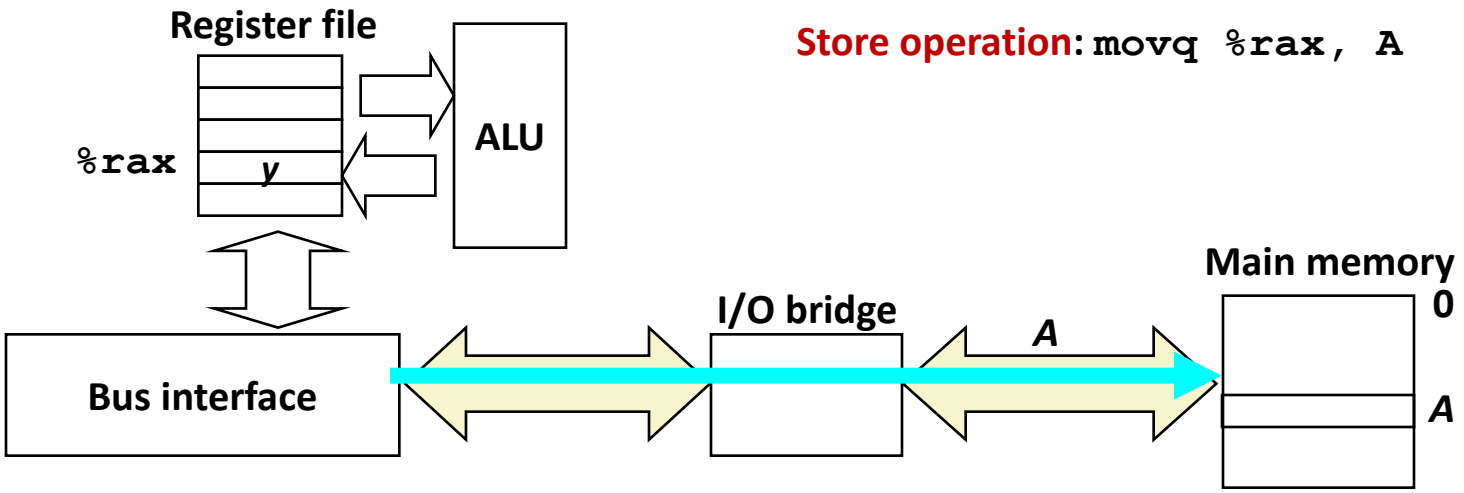
Memory Read Transaction (3)

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



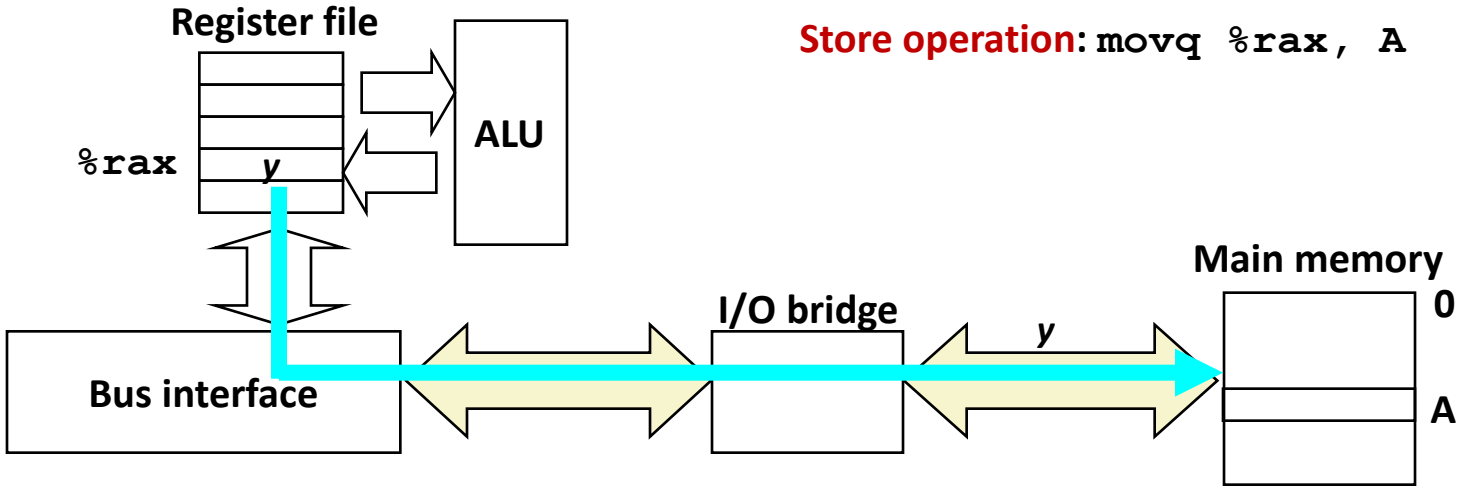
Memory Write Transaction (1)

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



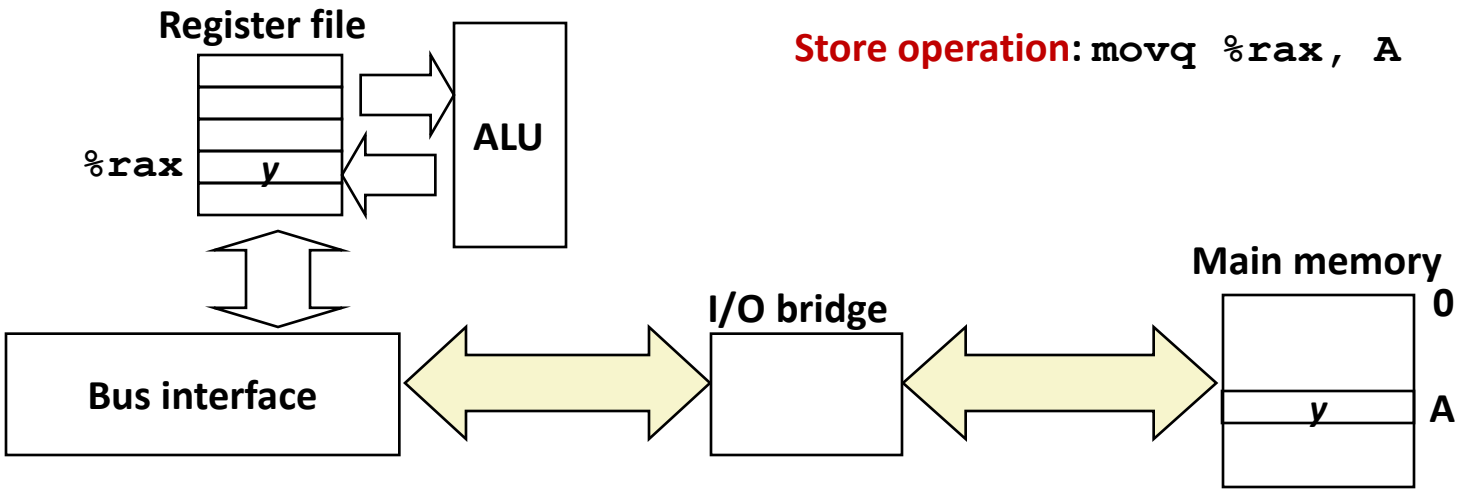
Memory Write Transaction (2)

- 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.



Today

- The memory abstraction — CSAPP 6.1.1
- **RAM : main memory building block — CSAPP 6.1.1**
- Storage technologies and trends — CSAPP 6.1.2-6.1.4
- The memory hierarchy — CSAPP 6.3
- Working sets — CSAPP 6.2
- Locality of reference — CSAPP 6.2
- Caches — CSAPP 6.4-6.5

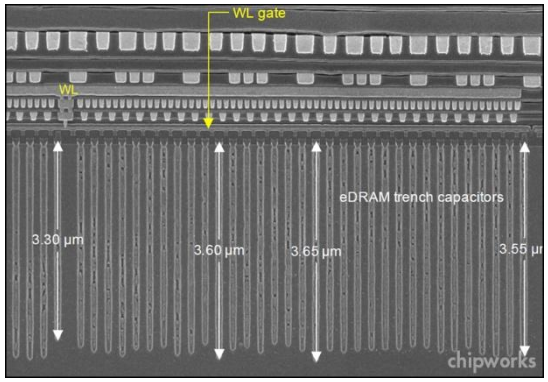
Random-Access Memory (RAM)

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

- **RAM comes in two varieties:**
  - SRAM (Static RAM)
  - DRAM (Dynamic RAM)

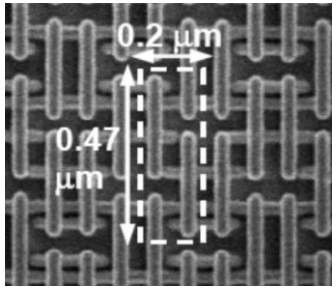
RAM Technologies

- **DRAM**



  - **1 Transistor + 1 capacitor / bit**
    - Capacitor oriented vertically
  - **Must refresh state periodically**

- **SRAM**



  - **6 transistors / bit**
  - **Holds state indefinitely (but will still lose data on power loss)**

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

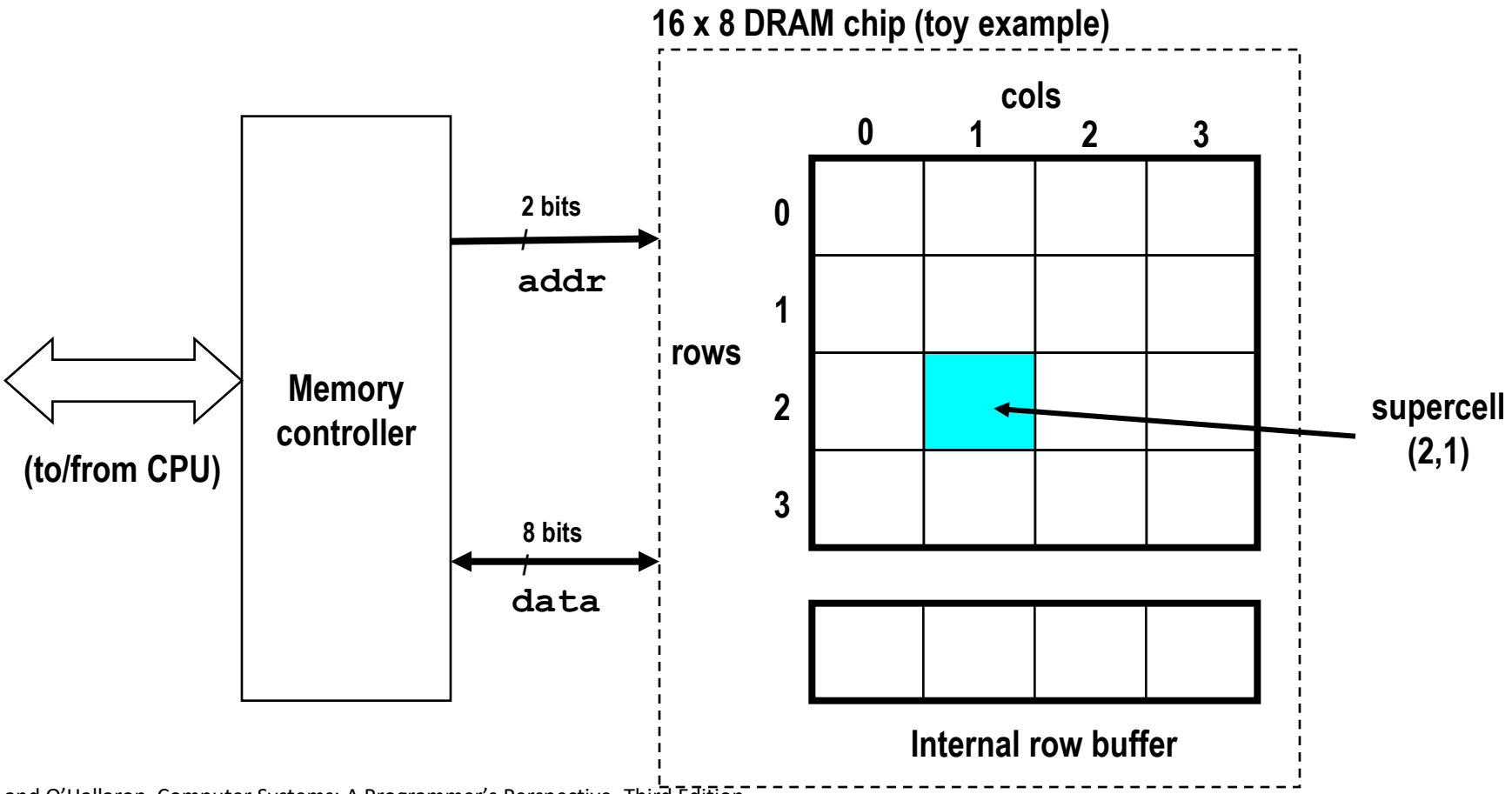
- **Trends**
  - SRAM scales with semiconductor technology
    - Reaching its limits
  - DRAM scaling limited by need for minimum capacitance
    - Aspect ratio limits how deep can make capacitor
    - Also reaching its limits

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

Conventional DRAM Organization

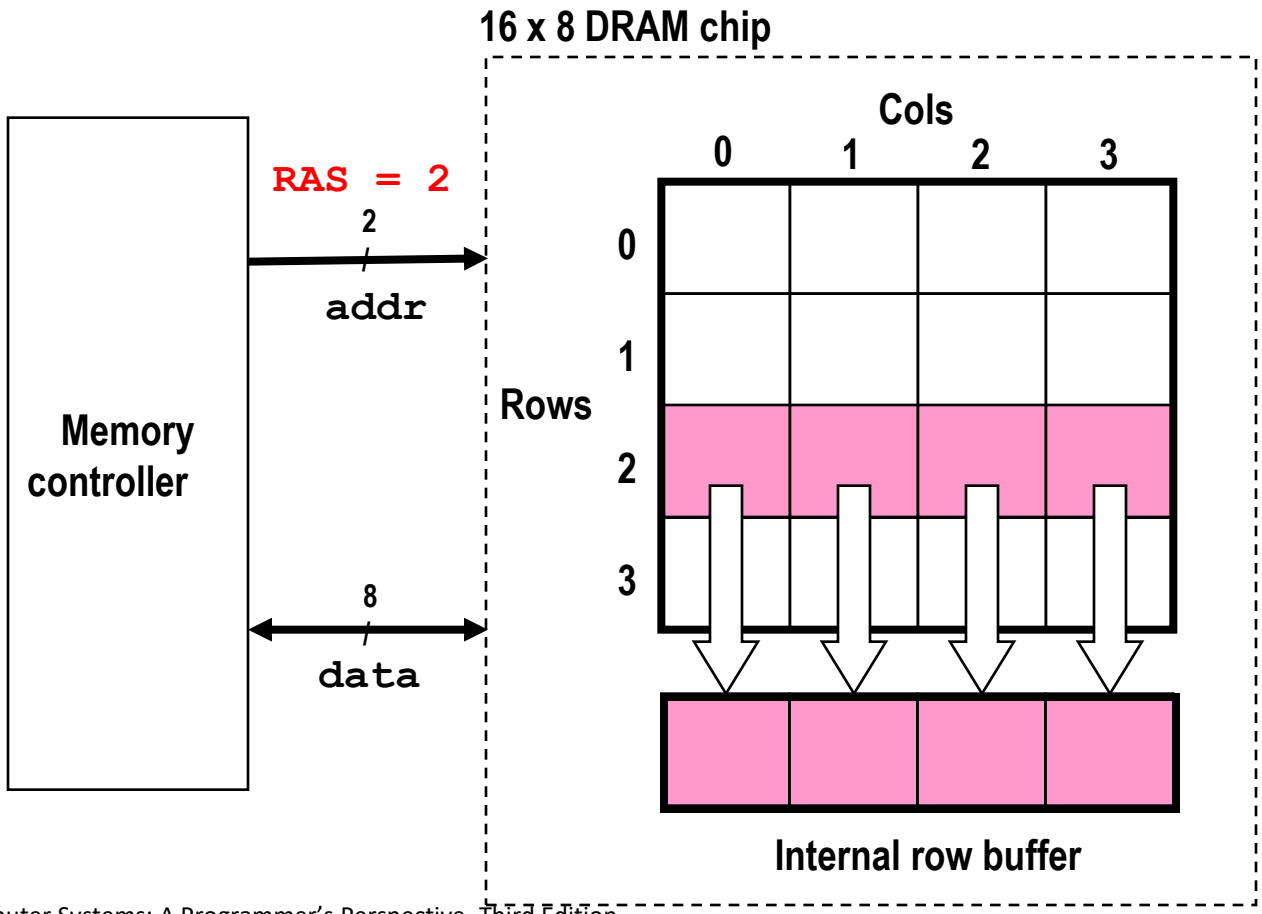
- ***d* x *w* DRAM:**
  - $d \cdot w$ total bits organized as d supercells of size w bits



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.

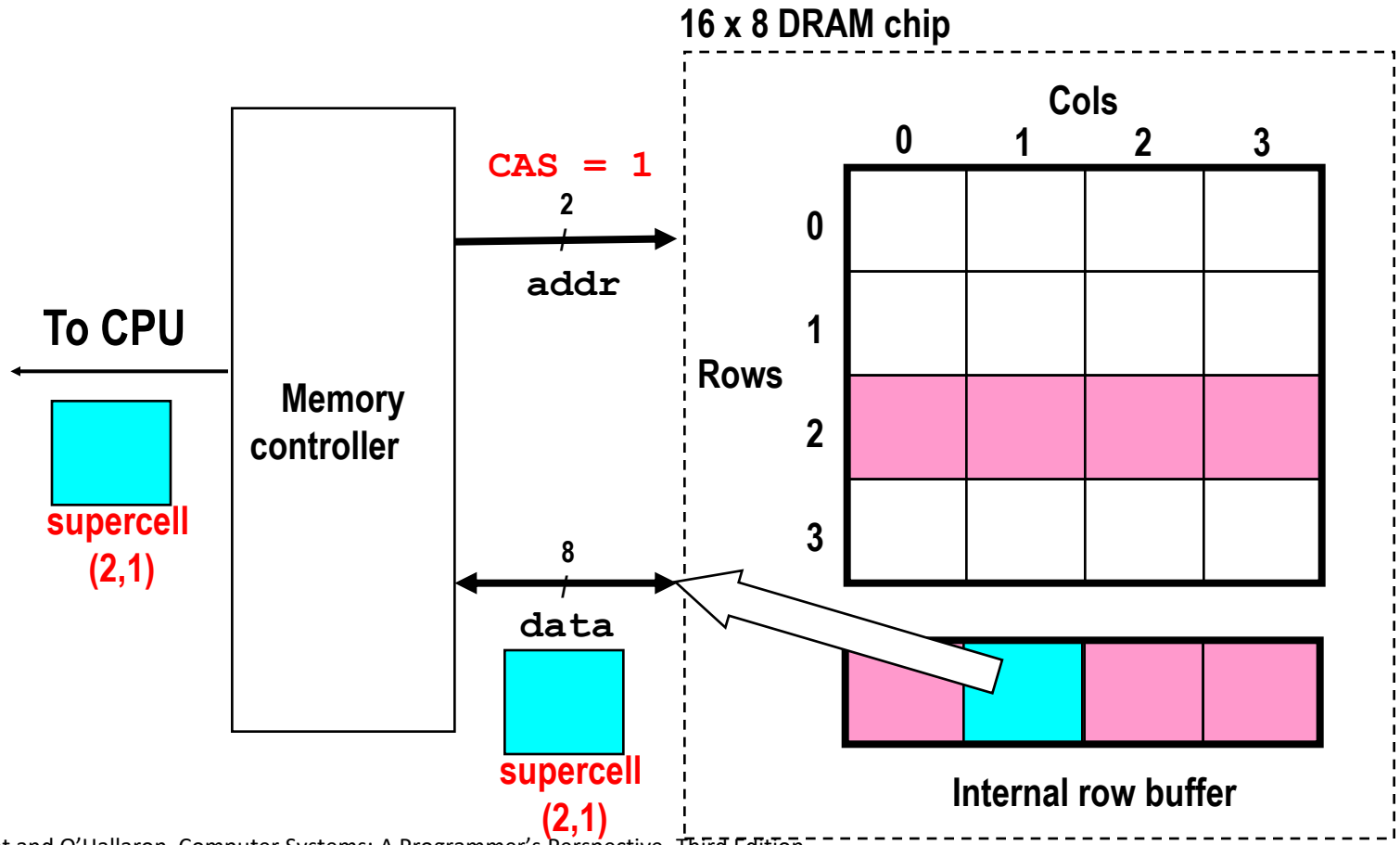


Reading DRAM Supercell (2,1)

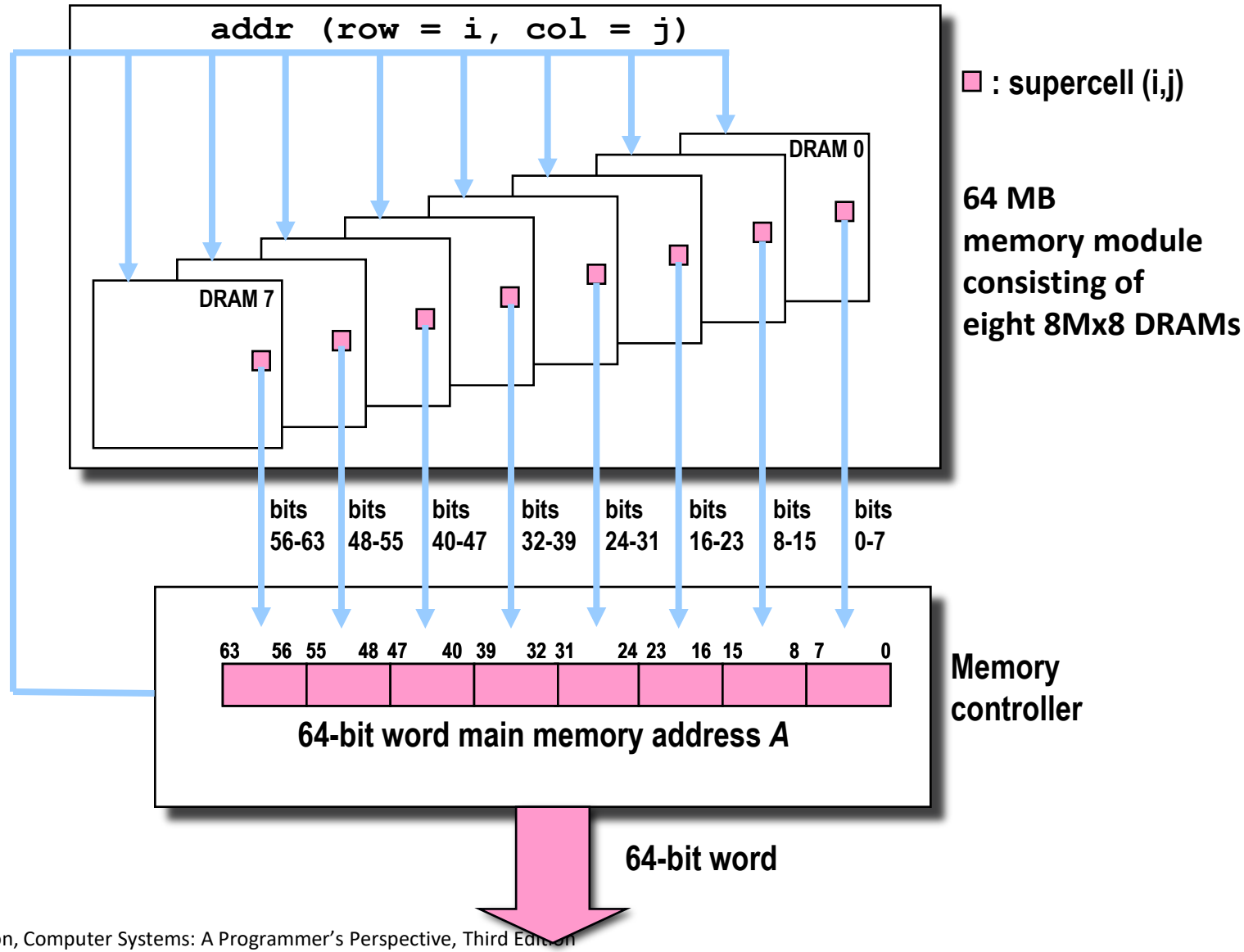
Step 2(a): Column access strobe (CAS) selects column 1.

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

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



Memory Modules



Today

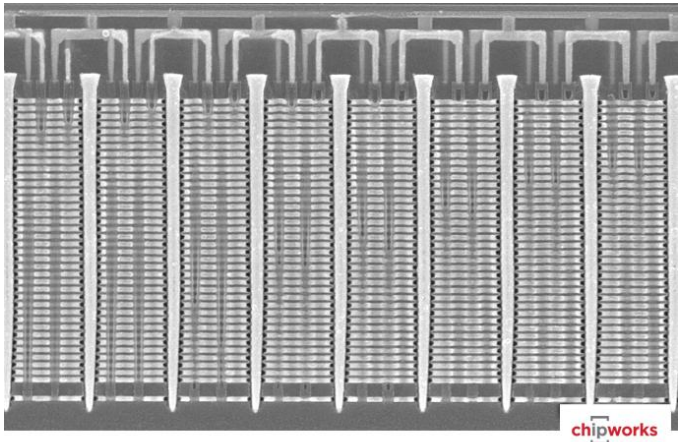
The memory abstraction	CSAPP 6.1.1
RAM : main memory building block	CSAPP 6.1.1
Storage technologies and trends	**CSAPP 6.1.2-6.1.4**
The memory hierarchy	CSAPP 6.3
Working sets	CSAPP 6.2
Locality of reference	CSAPP 6.2
Caches	CSAPP 6.4-6.5

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 bits data per cell

What's Inside A Disk Drive?

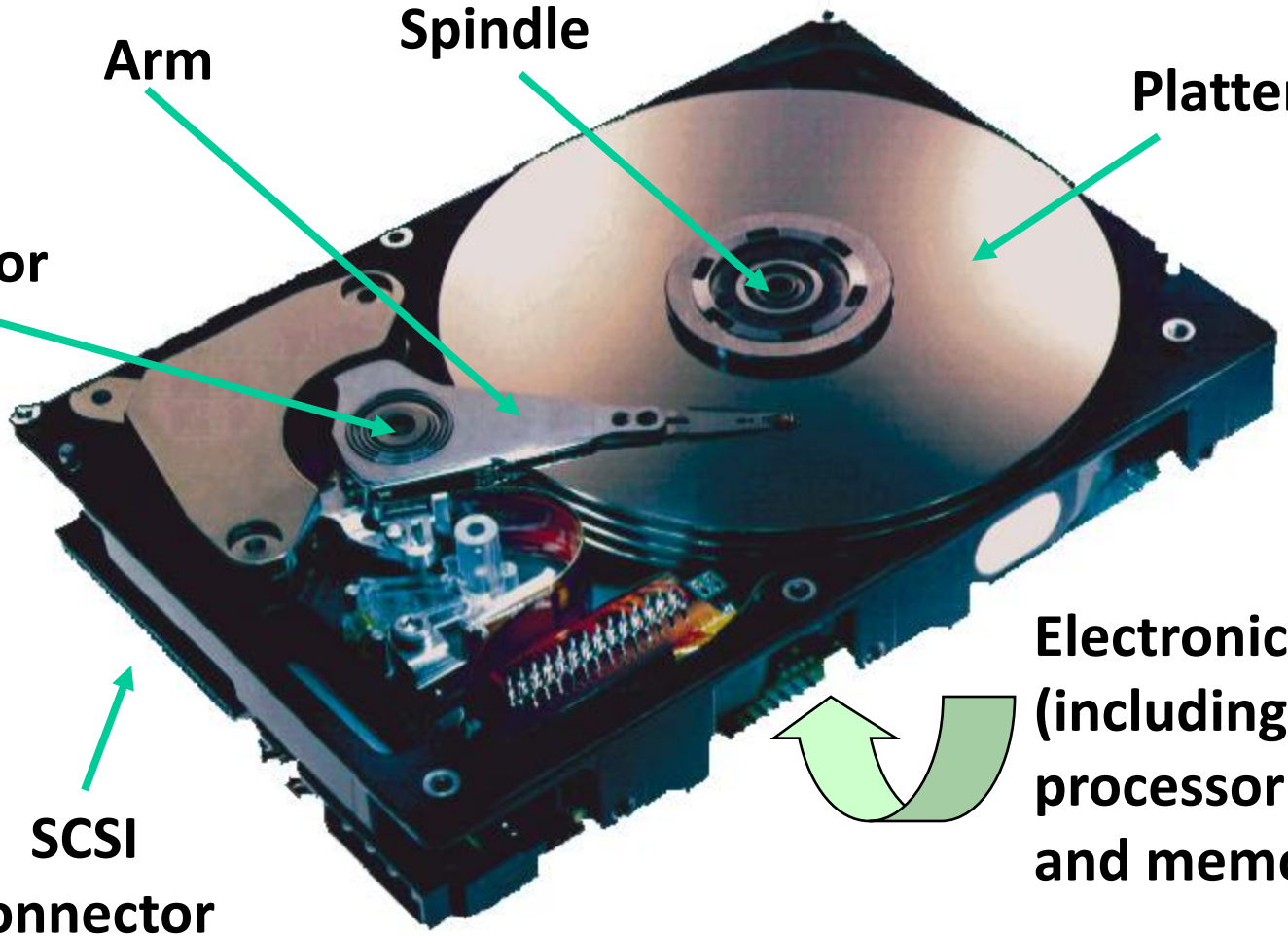
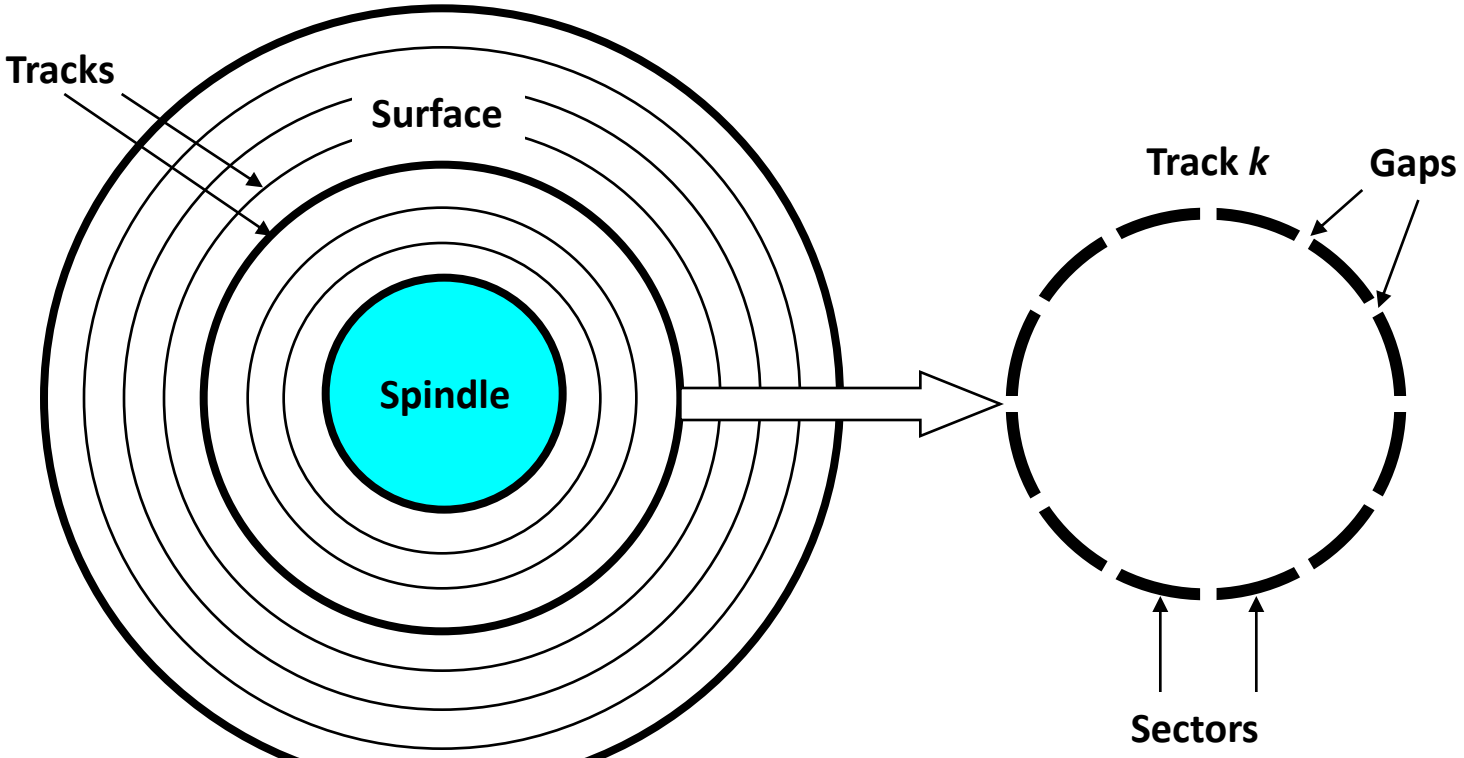


Image courtesy of Seagate Technology

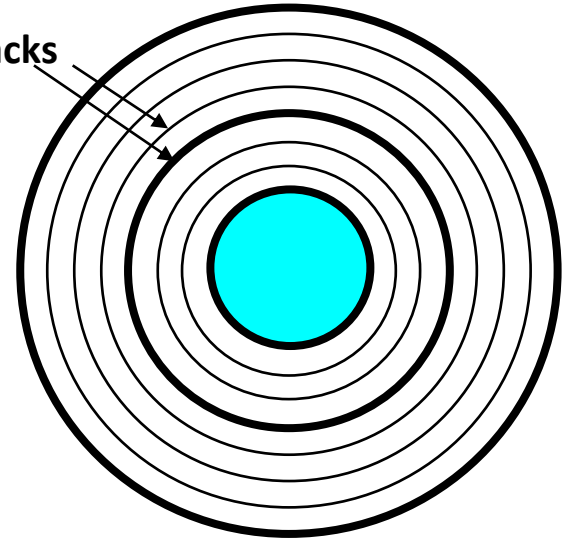
Disk Geometry

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



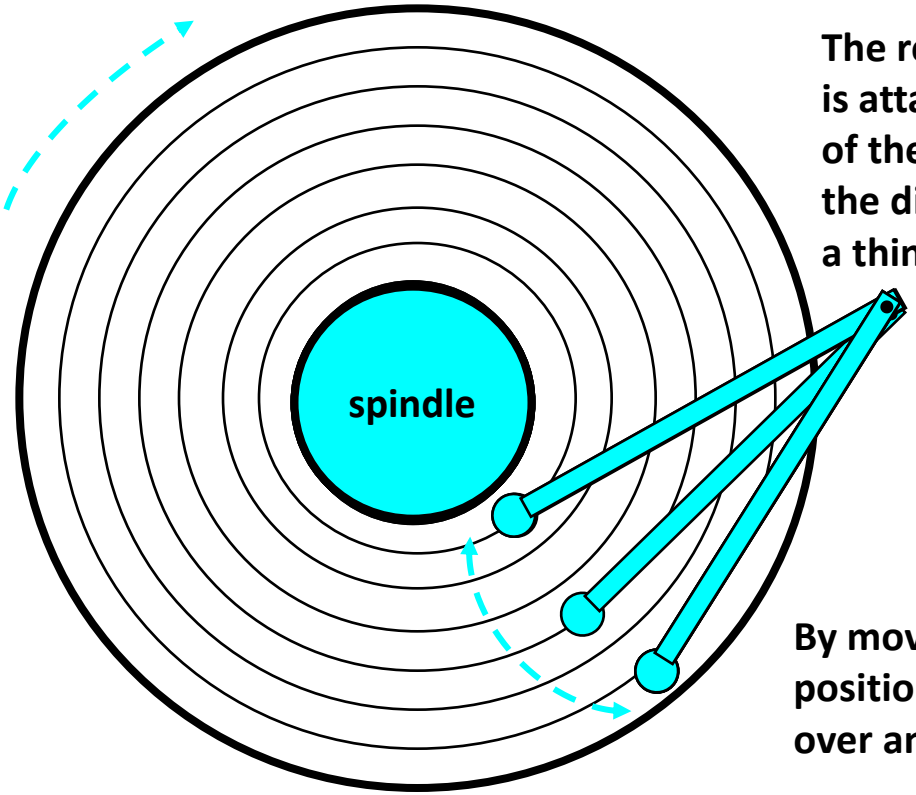
Disk Capacity

- **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.
  - Areal density (bits/in^2): product of recording and track density.



Disk Operation (Single-Platter View)

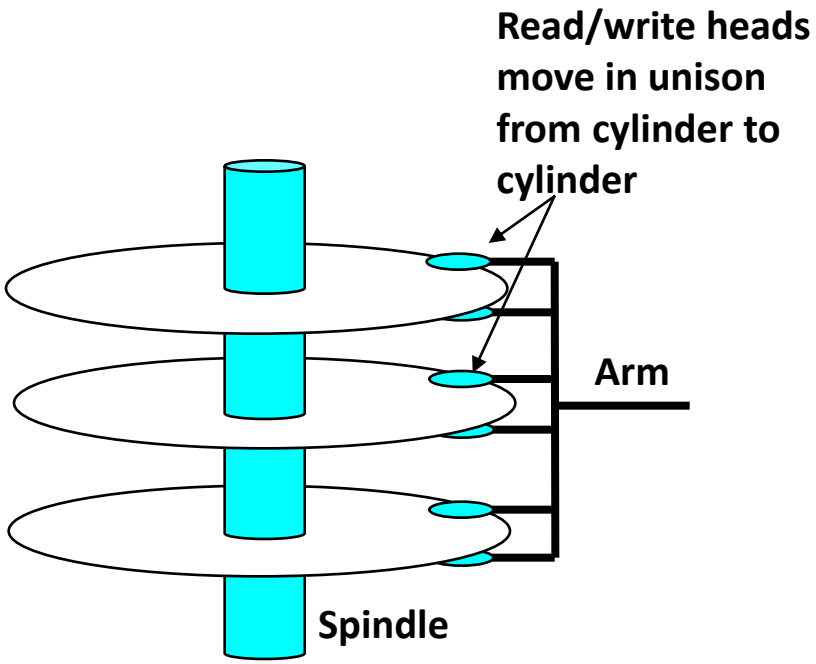
The disk surface spins at a fixed rotational rate



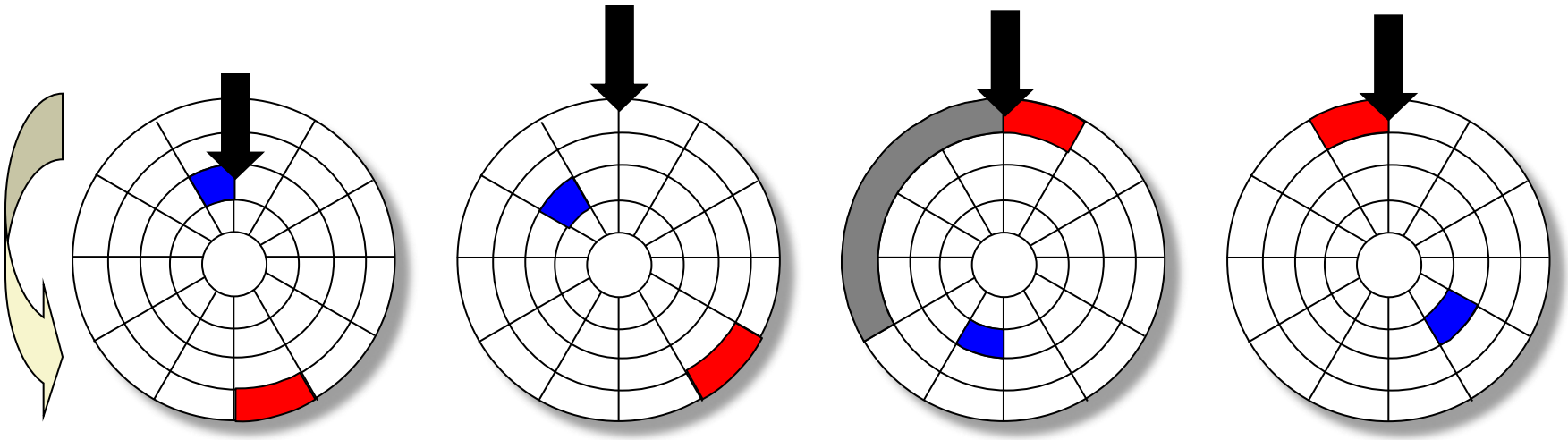
The read/write *head* is attached to the end of the *arm* and flies over the disk surface on a thin cushion of air.

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

Disk Operation (Multi-Platter View)



Disk Access – Service Time Components



Disk Access Time

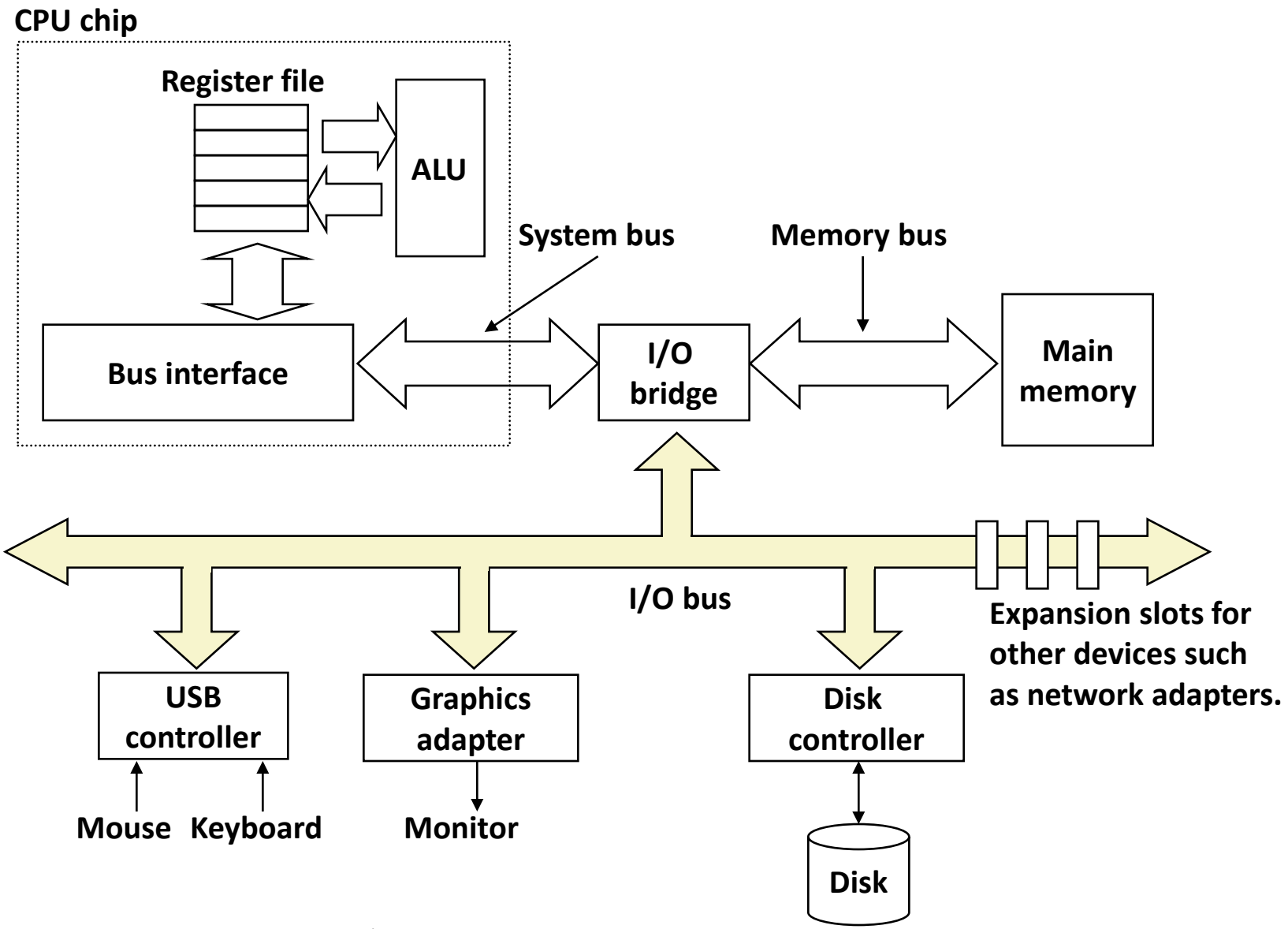
- **Average time to access some target sector approximated by:**
  - $T_{access} = T_{avg\ seek} + T_{avg\ rotation} + T_{avg\ transfer}$
- **Seek time ($T_{avg\ seek}$)**
  - Time to position heads over cylinder containing target sector.
  - Typical $T_{avg\ seek}$ is 3—9 ms
- **Rotational latency ($T_{avg\ rotation}$)**
  - Time waiting for first bit of target sector to pass under r/w head.
  - $T_{avg\ rotation}$ = 1/2 x 1/RPMs x 60 sec/1 min
  - Typical rotational rate = 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

time for one rotation (in minutes) fraction of a rotation to be read

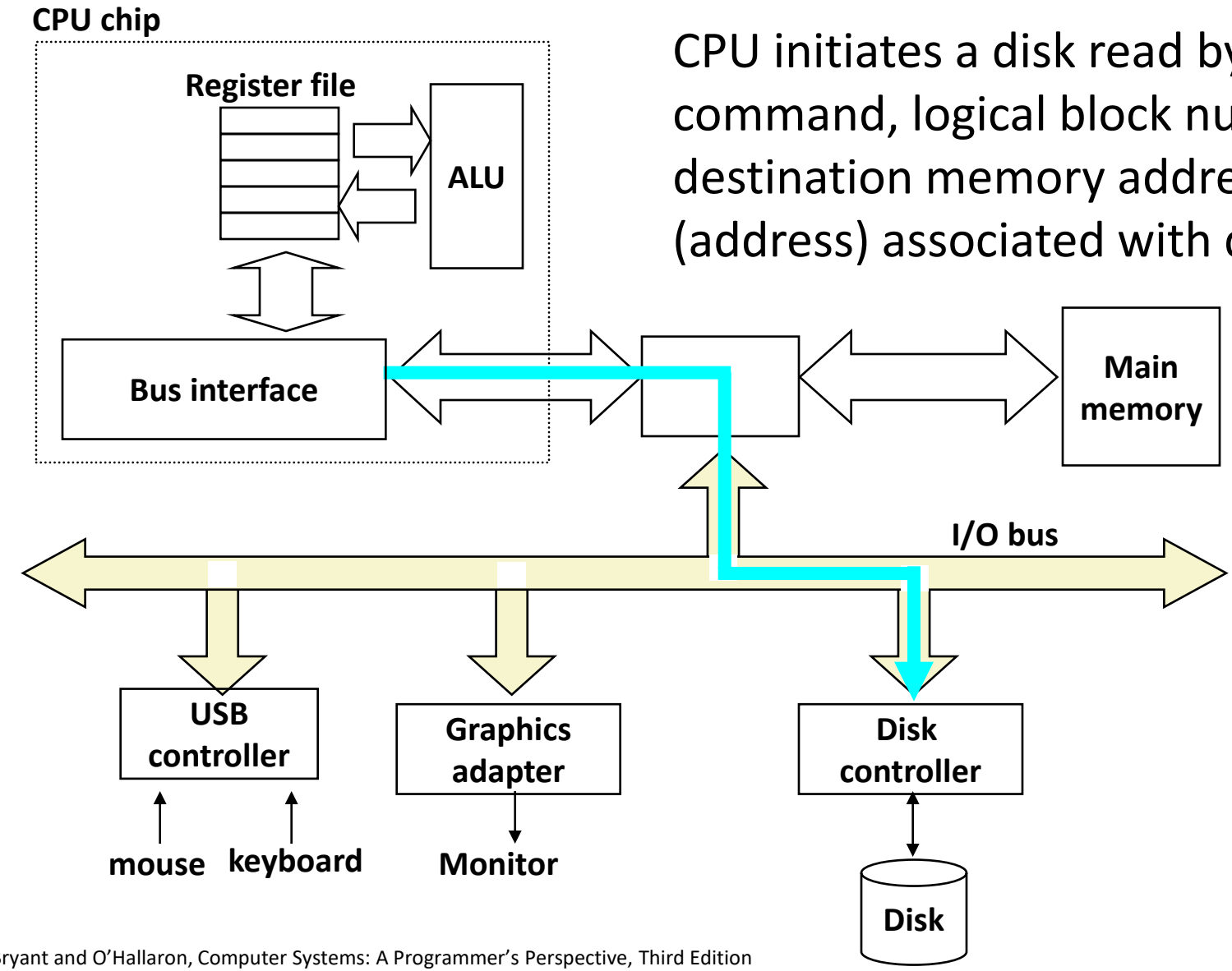
Disk Access Time Example

- **Given:**
  - Rotational rate = 7,200 RPM
  - Average seek time = 9 ms
  - Avg # sectors/track = 400
- **Derived:**
  - $T_{avg\ rotation}$ = 1/2 x (60 secs/7200 RPM) x 1000 ms/sec = 4 ms
  - $T_{avg\ transfer}$ = 60/7200 x 1/400 x 1000 ms/sec = 0.02 ms
  - T_{access} = 9 ms + 4 ms + 0.02 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.***

I/O Bus

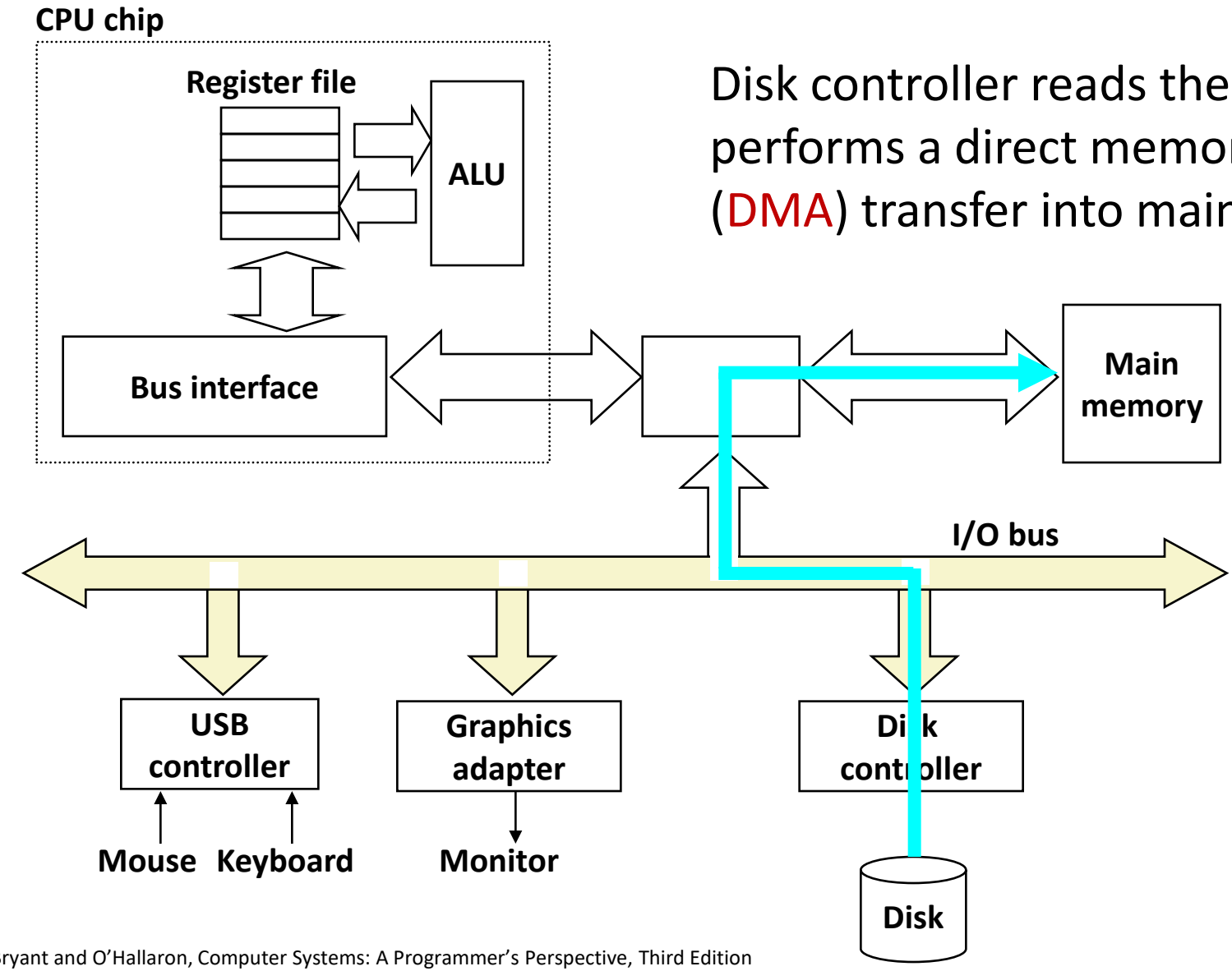

CPU chip
Register file
ALU
System bus
Memory bus
Bus interface
I/O bridge
Main memory
I/O bus
Expansion slots for other devices such as network adapters.
USB controller
Graphics adapter
Disk controller
Mouse
Keyboard
Monitor
Disk

Reading a Disk Sector (1)



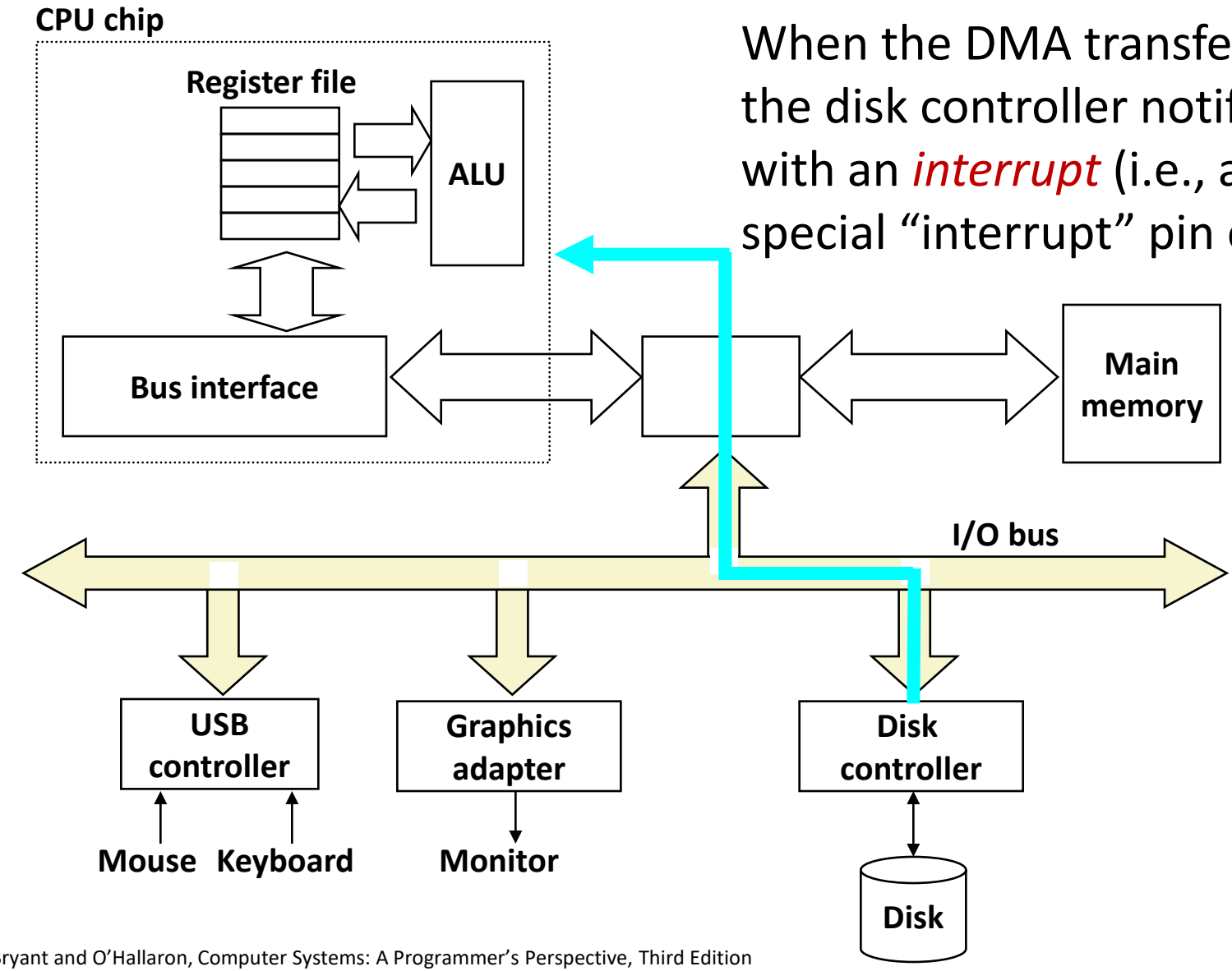
CPU initiates a disk read by writing a command, logical block number, and destination memory address to a port (address) associated with disk controller.

Reading a Disk Sector (2)



Disk controller reads the sector and performs a direct memory access (DMA) transfer into main memory.

Reading a Disk Sector (3)



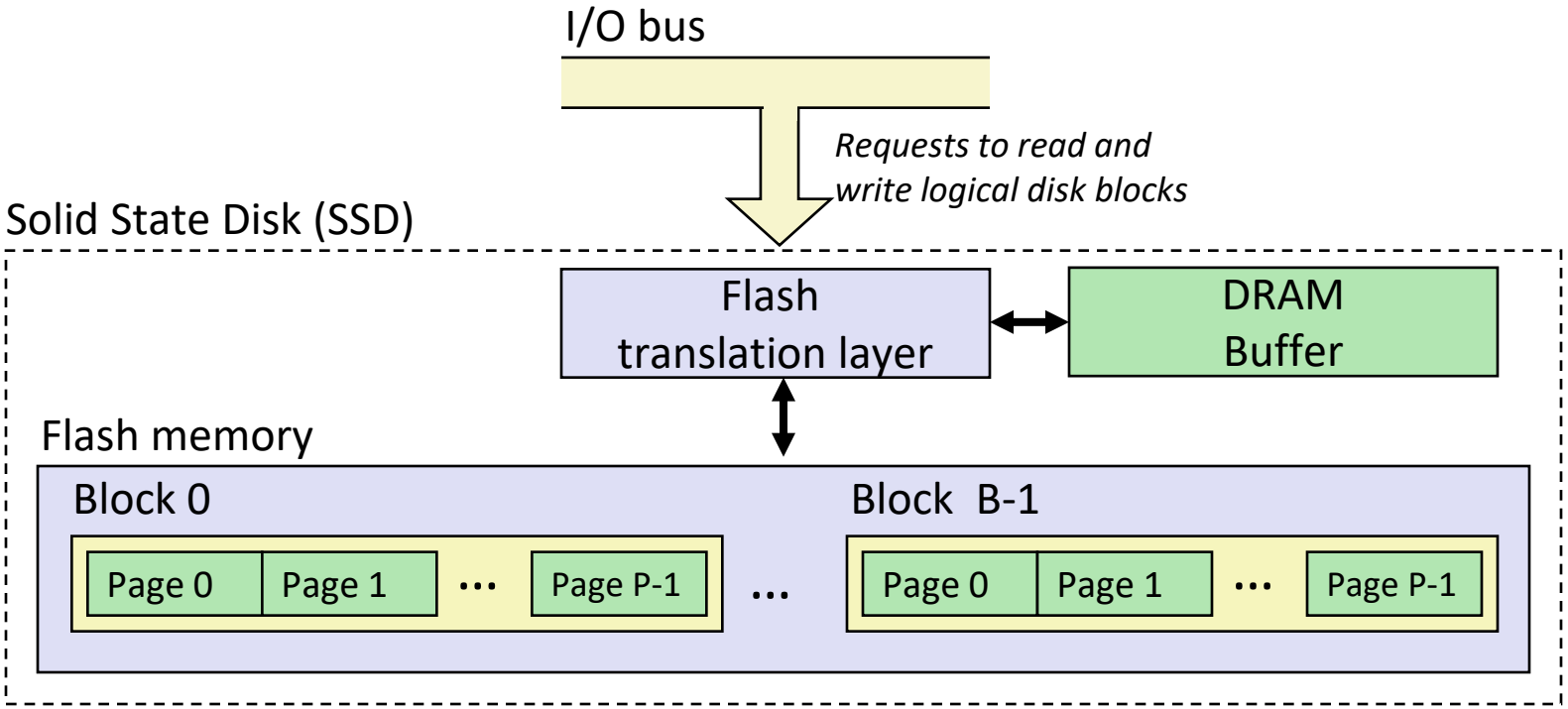
When the DMA transfer completes, the disk controller notifies the CPU with an *interrupt* (i.e., asserts a special "interrupt" pin on the CPU).

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,…)
  - Solid state disks (replacing rotating disks)
  - Disk caches

Solid State Disks (SSDs)



- **Pages: 512B 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

- **Benchmark of Samsung 940 EVO Plus**

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

Sequential read throughput	**2,126 MB/s**	**Sequential write tput**	**1,880 MB/s**
Random read throughput	**140 MB/s**	**Random write tput**	**59 MB/s**

- **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.
  - Flash translation layer allows accumulating series of small writes before doing block write.

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 2 times more expensive per byte
    - 1TB SSD is 8¢/GB. 1TB HDD is 4¢/GB. 12TB HDD is 3¢/GB
- **Applications**
  - Smartphones, laptops
  - Increasingly common in desktops and servers

Today

- The memory abstraction — CSAPP 6.1.1
- RAM : main memory building block — CSAPP 6.1.1
- Storage technologies and trends — CSAPP 6.1.2-6.1.4
- **The memory hierarchy** — **CSAPP 6.3**
- Working sets — CSAPP 6.2
- Locality of reference — CSAPP 6.2
- Caches — CSAPP 6.4-6.5

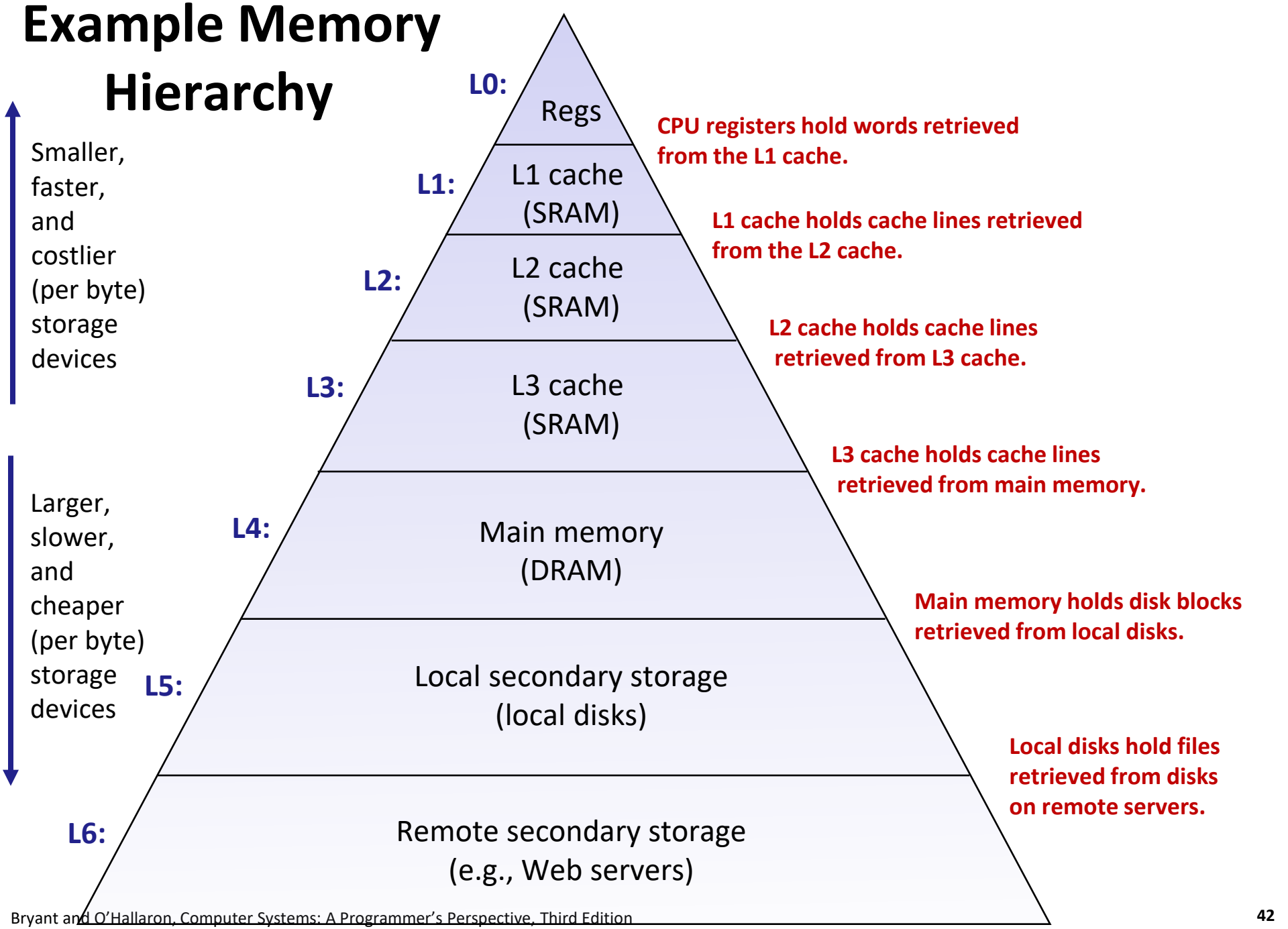
Memory Hierarchies

- **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.**

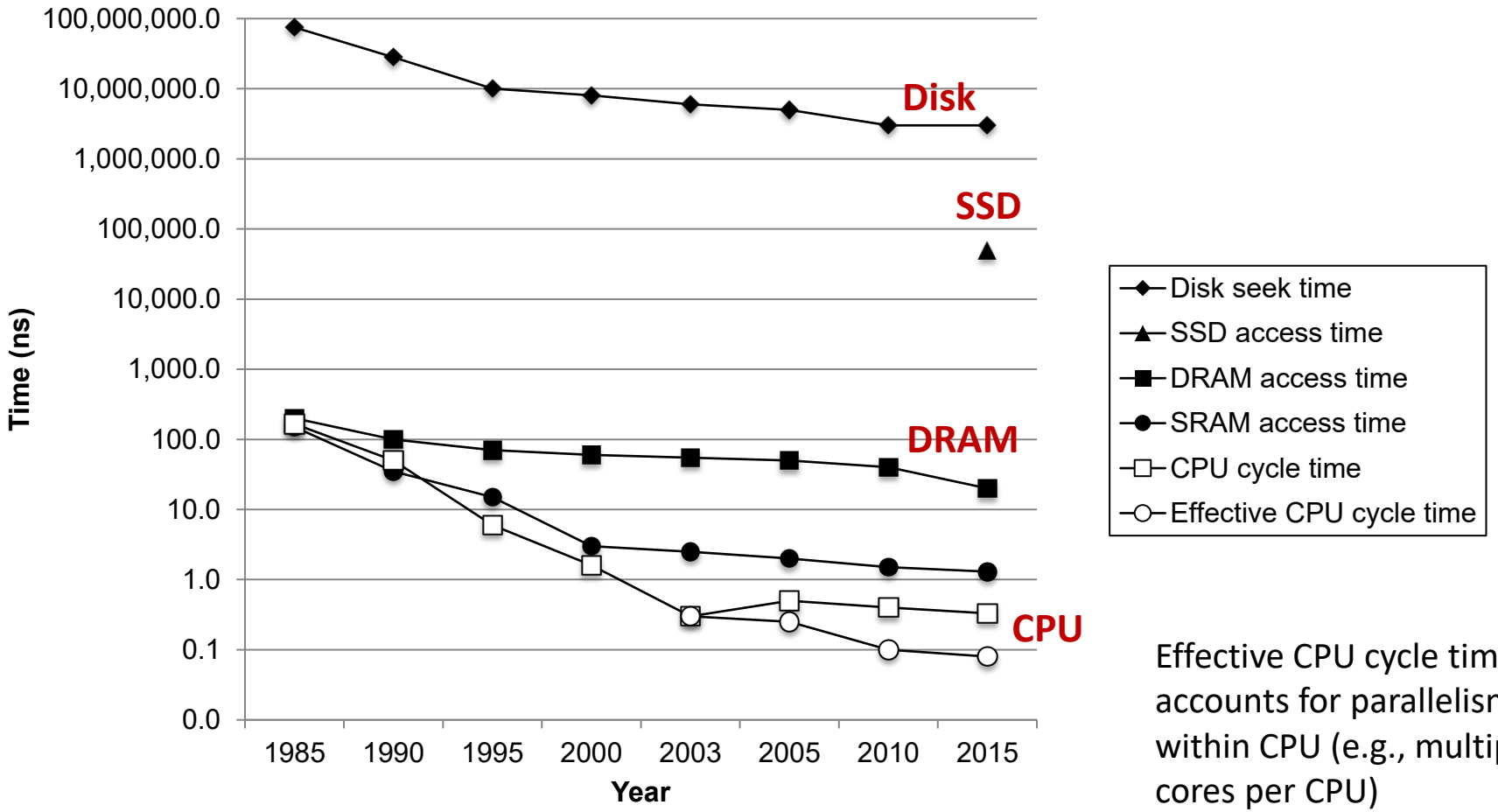
- **They suggest an approach for organizing memory and storage systems known as a memory hierarchy.**

Example Memory Hierarchy



The CPU-Memory Gap

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



Effective CPU cycle time: accounts for parallelism within CPU (e.g., multiple cores per CPU)

Today

- The memory abstraction — CSAPP 6.1.1
- RAM : main memory building block — CSAPP 6.1.1
- Storage technologies and trends — CSAPP 6.1.2-6.1.4
- The memory hierarchy — CSAPP 6.3
- **Working sets** — **CSAPP 6.2**
- Locality of reference — CSAPP 6.2
- Caches — CSAPP 6.4-6.5

Working Sets

- Think about working on the bomb lab. Maybe you have the write-up open in one window, the debugger open in another window, and an assembly reference in yet another window.
- Think about eating a meal. You need room on the table for your fork, your knife, your plate, your napkin, and your glass.
- The set of resources actively needed for a task are called the **working set** for the task.

Working Sets

- If we can keep the working set for the active task in the fastest memory, the task will perform at the speed of the fastest memory.
- This is true even if the working sets for inactive tasks that we are in slower memory.
- If we don’t have the enough fast memory to hold the whole working set of the active task, we end up repeatedly paying the price to swap what we’ll need soon for we’ll need right now.
- The cost of the initial, likely incremental, movement of the working set from slower memory to faster memory may be expensive, but it is often amortized to something negligible over the lifetime of the task.

Determining the Working Set

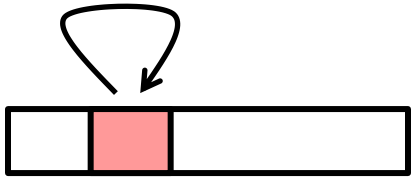
- **If the system can determine the working set, and it has enough fast memory, it can arrange to bring it into the fast memory or keep it there once it gets there.**
  - Sort of like setting the table
  - Or leaving everything set up on the workbench at night to be ready for morning.
- **But, the processors doesn't understand the task. It can't see (much of) the future. It just does what it is told. It can't know the working set.**
  - And the task, and therefore the associated working set, can change at any time.
- **But it can use heuristics to estimate or approximate it.**
  - Let's talk about **Locality: Spatial locality** and **Temporal locality.**

Today

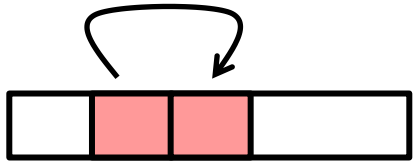
- The memory abstraction — CSAPP 6.1.1
- RAM : main memory building block — CSAPP 6.1.1
- Storage technologies and trends — CSAPP 6.1.2-6.1.4
- The memory hierarchy — CSAPP 6.3
- Working sets — CSAPP 6.2
- **Locality of reference** — **CSAPP 6.2**
- Caches — CSAPP 6.4-6.5

Locality:
A Heuristic for Approximating the Working Set

- **Principle of Locality:** **Programs tend to use data and instructions with addresses near or equal to those they have used recently**

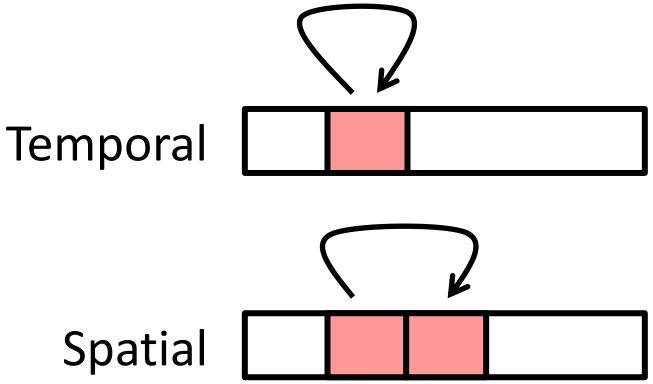


- **Temporal locality:**
  - Recently referenced items are likely to be referenced again in the near future



- **Spatial locality:**
  - Items with nearby addresses tend to be referenced close together in time

Locality 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**).
  - Reference variable **`sum`** each iteration.
- **Instruction references**
  - Reference instructions in sequence.
  - Cycle through loop repeatedly.

Spatial or Temporal Locality?

spatial

temporal

spatial

temporal

Qualitative Estimates of Locality

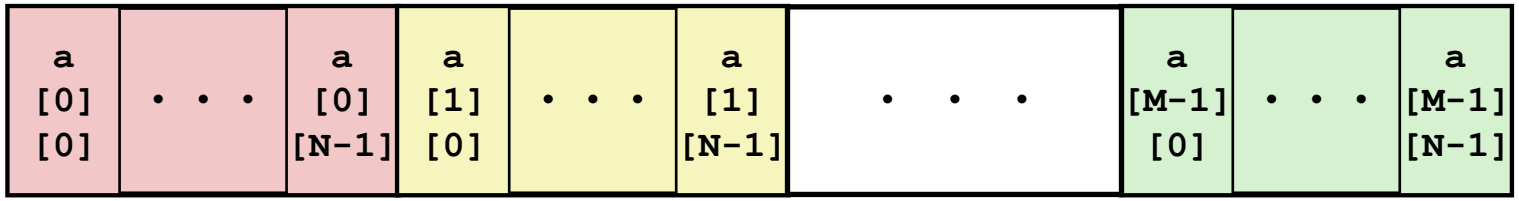
- **Claim:** Being able to look at code and get a qualitative sense of its locality is a key skill for a professional programmer.
- **Question:** Does this function have good locality with respect to array a?

Hint: array layout is row-major order

Answer: yes
Stride-1 reference pattern

```
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;
}
```



Locality Example

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

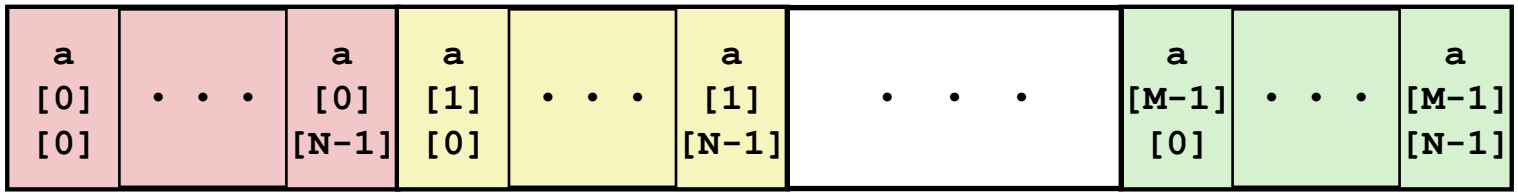
```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;

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

Answer: no

Stride N reference pattern

Note: If M is very small then good locality. Why?



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)?

```
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;
}
```

```
$ time ./loopijk

real    0m2.765s
user    0m2.328s
sys     0m0.422s


$ time ./loopkij

real    0m1.651s
user    0m1.234s
sys     0m0.422s
```

Answer: make j the inner loop

What Makes Locality A Good Heuristic?

- Remember that we care about spatial and temporal locality because they allow us to estimate the working set.
- But, what makes them a particularly good way of doing that?
- They are simple enough to implement efficiently in hardware without slowing things down too much
  - Temporal locality can be managed by keeping the most recently used objects and letting go of the least recently used objects.
  - Spatial locality is natural to model by maintaining blocks of nearby objects vs individual objects
  - They can be learned quickly enough to be of value in the future
  - They don't require prior knowledge or understanding of the task.

Is Locality Always a Good Heursitic?

- **Spatial and temporal locality are heuristics. They are not necessarily applicable to all workloads.**

- **As one example, consider *streaming* data to preprocess it enroute to an AI algorithm or other application.**
  - Spatial locality still applies: If we access one piece of data, we're likely to access the next piece of data next.
  - But, temporal locality no longer applies: Once we've seen it and processed it, we won't look back
  - The working set is a dynamically moving window.

- **In this situation, we might want to a heuristic that suggests, "After N sequential accesses, beginning prefetching ahead of the current accesses to overlap I/O and processing."**
  - Future attraction: We see this in the OS paging system, for example.

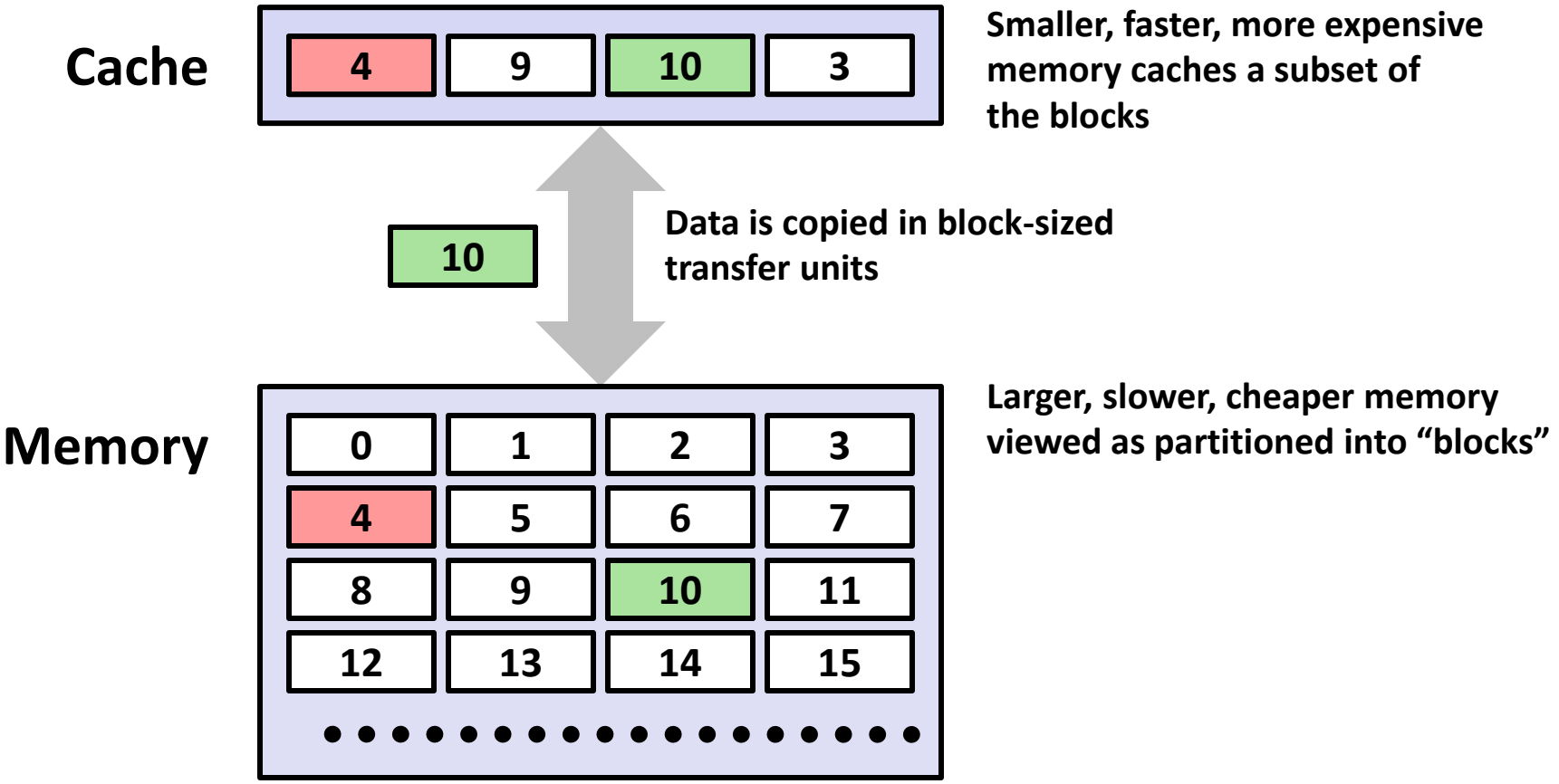
Today

- The memory abstraction — CSAPP 6.1.1
- RAM : main memory building block — CSAPP 6.1.1
- Storage technologies and trends — CSAPP 6.1.2-6.1.4
- The memory hierarchy — CSAPP 6.3
- Working sets — CSAPP 6.2
- Locality of reference — CSAPP 6.2
- **Caches** — **CSAPP 6.4-6.5**

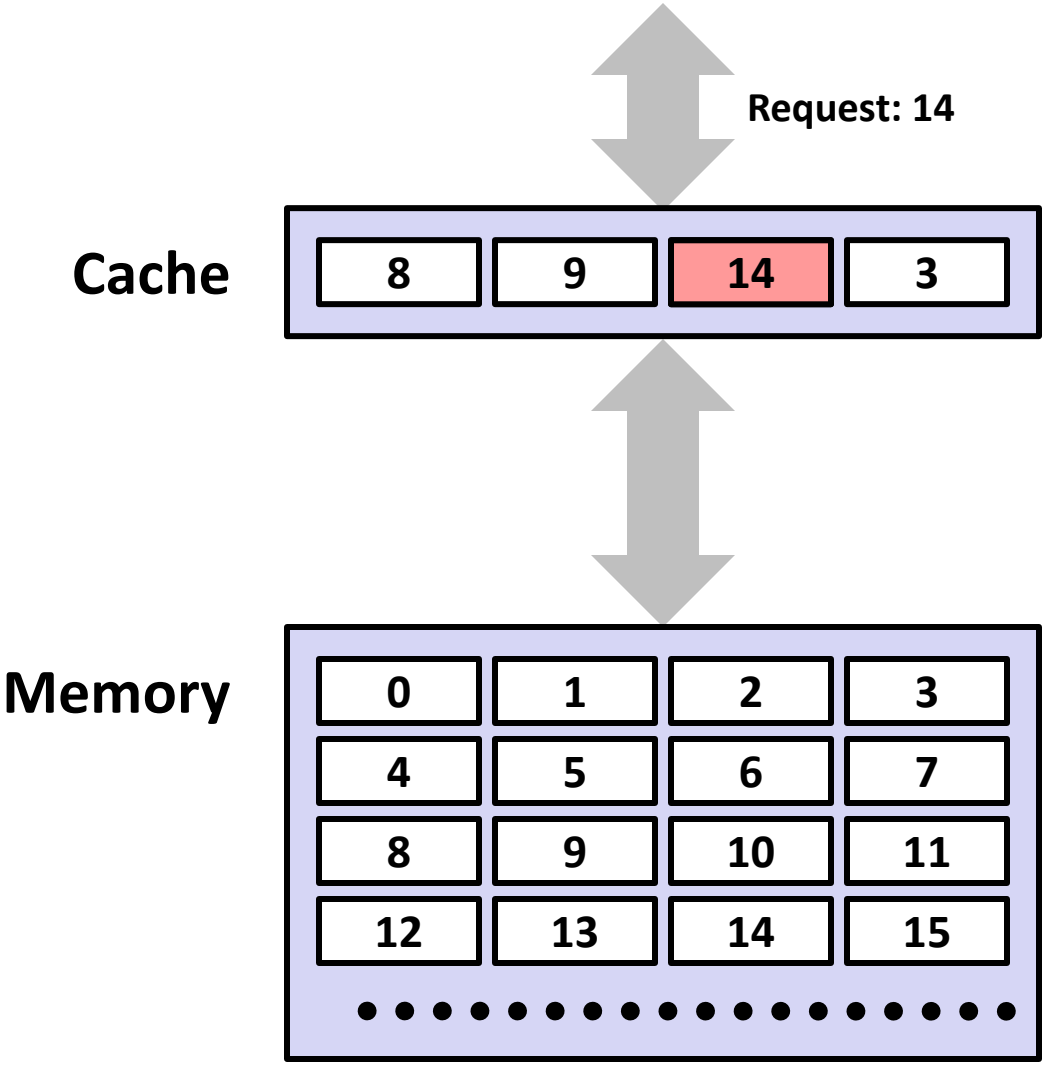
Caches

- ***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



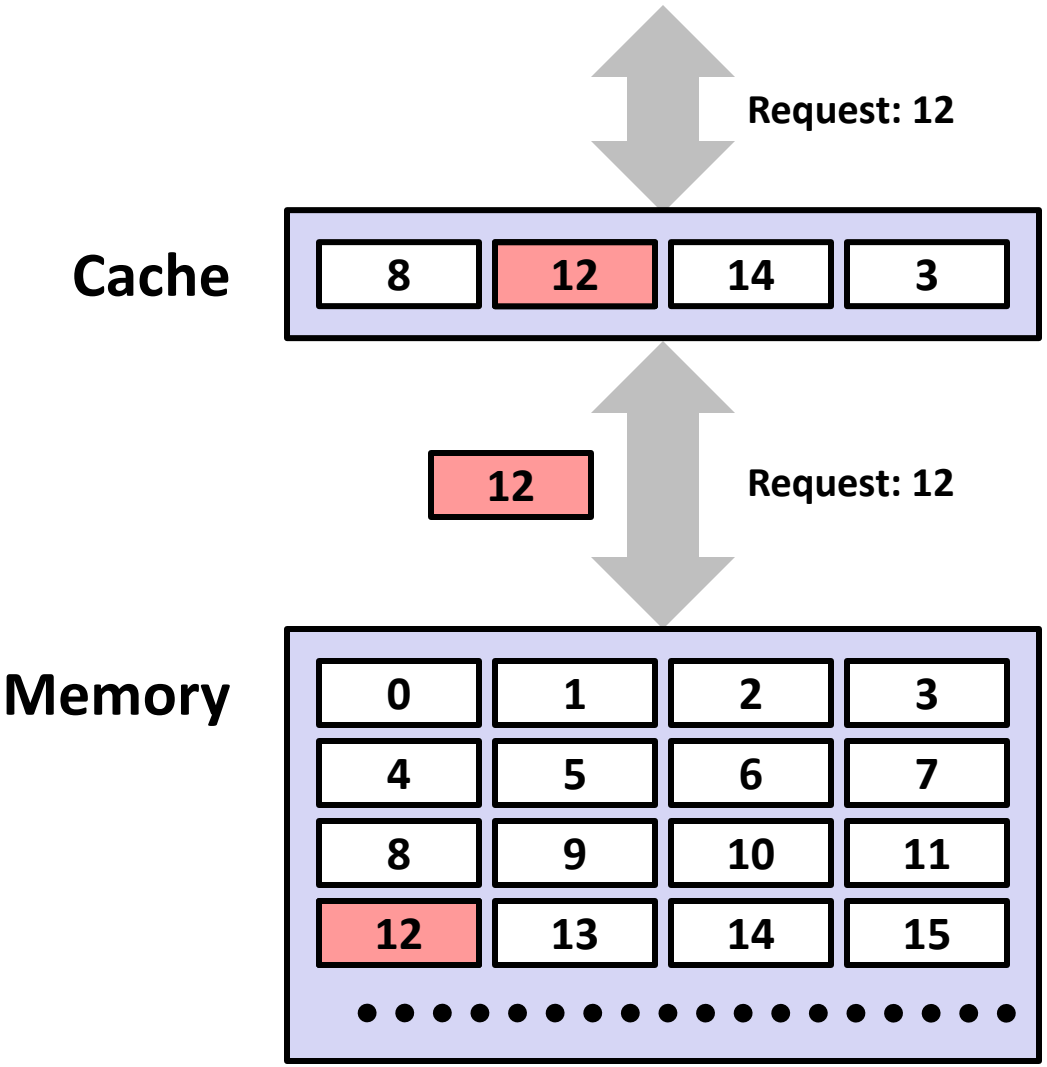
General Cache Concepts: Hit



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)

Impact of spatial locality on number of misses?

General Caching Concepts: 3 Types of Cache Misses

- **Compulsory miss**
  - Compulsory 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 and the average cost of a storage approximate our cheapest memory**
- **Careful movement of data can allow us to have the average access approximate our fastest memory**
  - An understanding of the behavior of our work, including its working set, locality, etc, can enable us to facilitate this movement.
- **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

Supplemental slides

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