photo by unsplash.com user @martijnbaudoin

COMP201

Computer Systems & Programming

Lecture #20 – The Memory Hierarchy

KOÇ UNIVERSITY

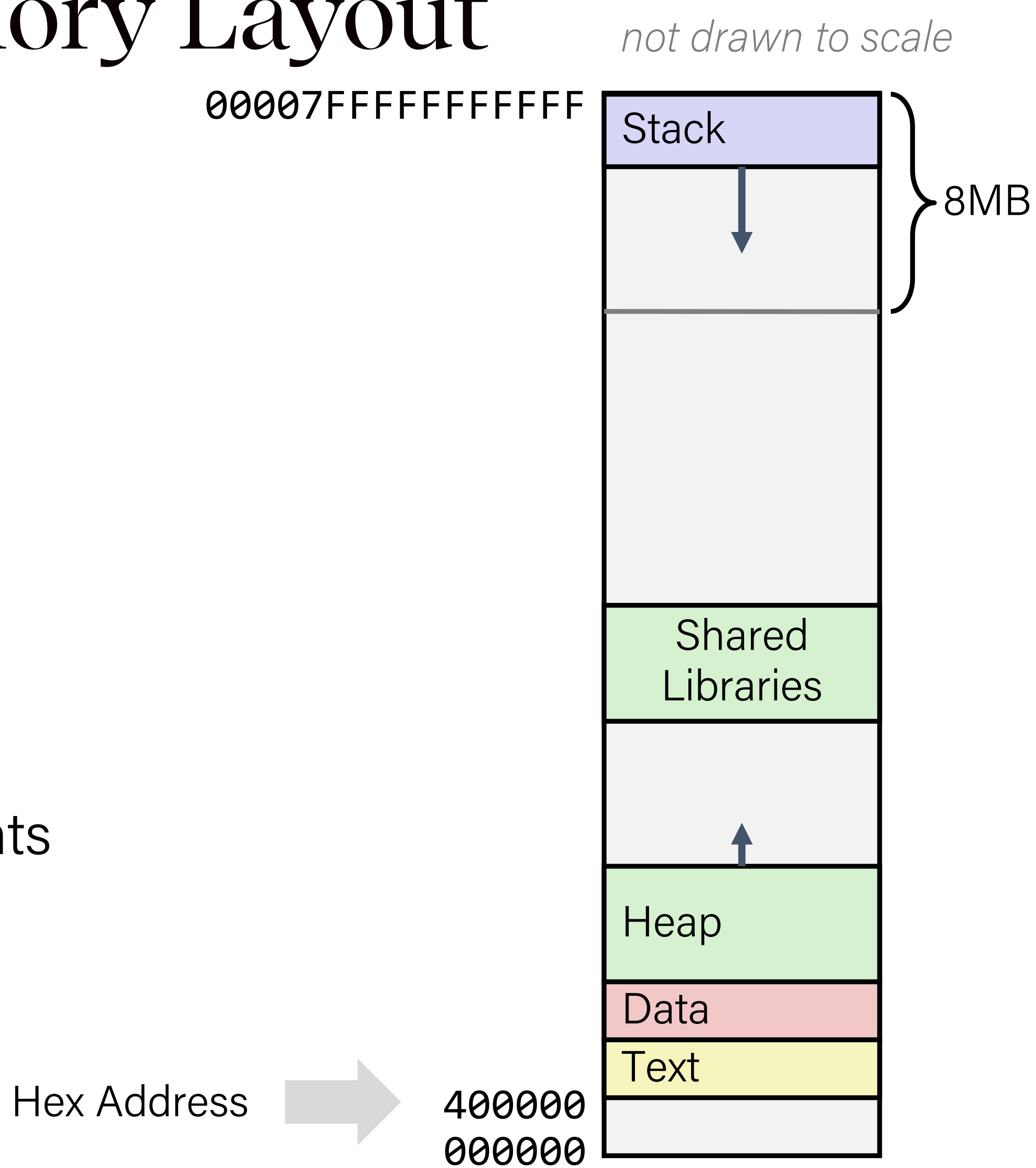
Aykut Erdem // Koç University // Spring 2022

Recap

- Floating Point
- Memory Layout
- Buffer Overflow

Recap: x86-64 Linux Memory Layout

- Stack
  - Runtime stack (8MB limit)
  - E. g., local variables
- Heap
  - Dynamically allocated as needed
  - When call `malloc()`, `calloc()`, `new()`
- Data
  - Statically allocated data
  - E.g., global vars, static vars, string constants
- Text / Shared Libraries
  - Executable machine instructions
  - Read-only



Recap: Memory Referencing Bug Example

```
typedef struct {
  int a[2];
  double d;
} struct_t;

double fun(int i) {
  volatile struct_t s;
  s.d = 3.14;
  s.a[i] = 1073741824; /* Possibly out of bounds */
  return s.d;
}
```

```
fun(0) → 3.14
fun(1) → 3.14
fun(2) → 3.1399998664856
fun(3) → 2.00000061035156
fun(4) → 3.14
fun(6) → Segmentation fault
```

Result is system specific

Recap: Buffer Overflows

- **Buffer overflow bugs can allow remote machines to execute arbitrary code on victim machines**
- Distressingly common in real programs
  - Programmers keep making the same mistakes ☹️
  - Recent measures make these attacks much more difficult
- Examples across the decades
  - Original "Internet worm" (1988)
  - "IM wars" (1999)
  - Twilight hack on Wii (2000s)
  - ... and many, many more
- You will learn some of the tricks in Assignment 5
  - Hopefully to convince you to never leave such holes in your programs!!

COMP201 Topic 7: How does the memory system is organized as a hierarchy of different storage devices with unique capacities?

Plan for Today

- The memory abstraction
- Storage technologies and trends
- Locality of reference
- The memory hierarchy

Disclaimer: Slides for this lecture were borrowed from

—Randal E. Bryant and David R. O'Hallaroni's CMU 15-213 class

—Porter Jones' UW CSE 351 class

Lecture Plan

- The memory abstraction
- Storage technologies and trends
- Locality of reference
- The memory hierarchy

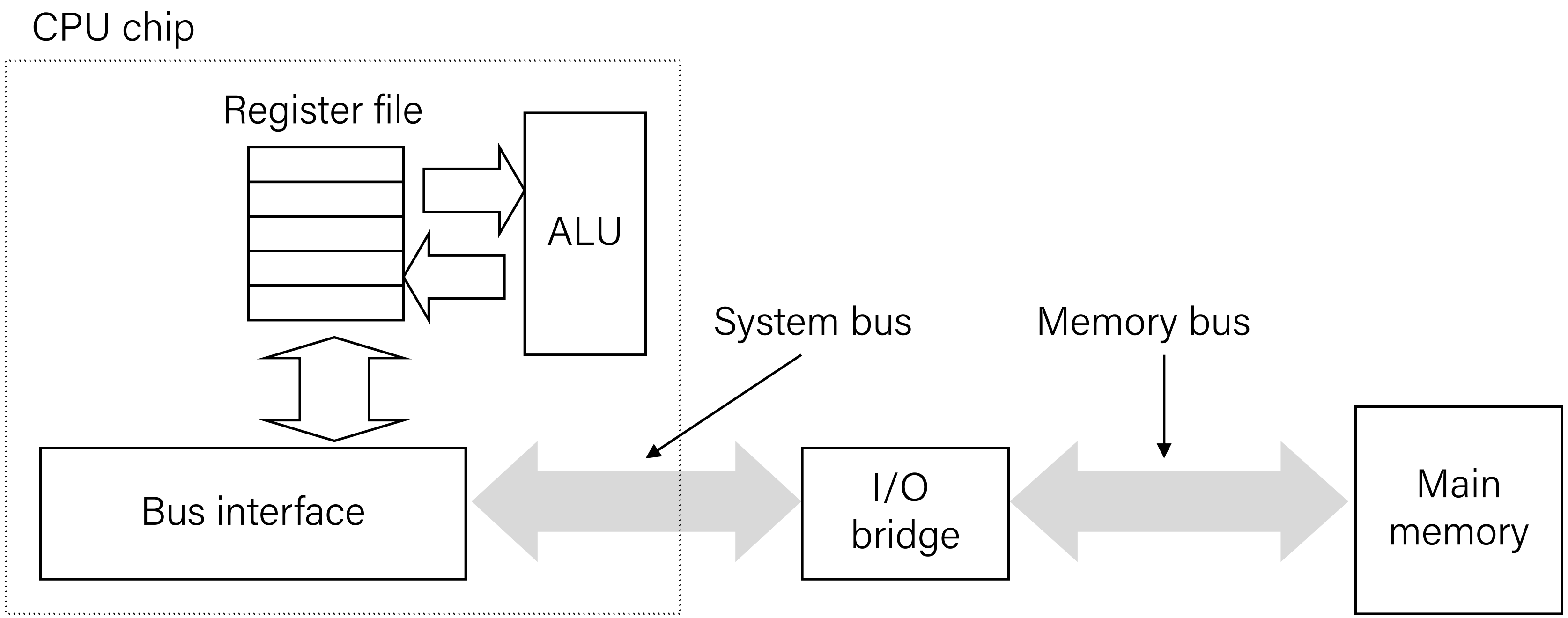
Writing & Reading Memory

- 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

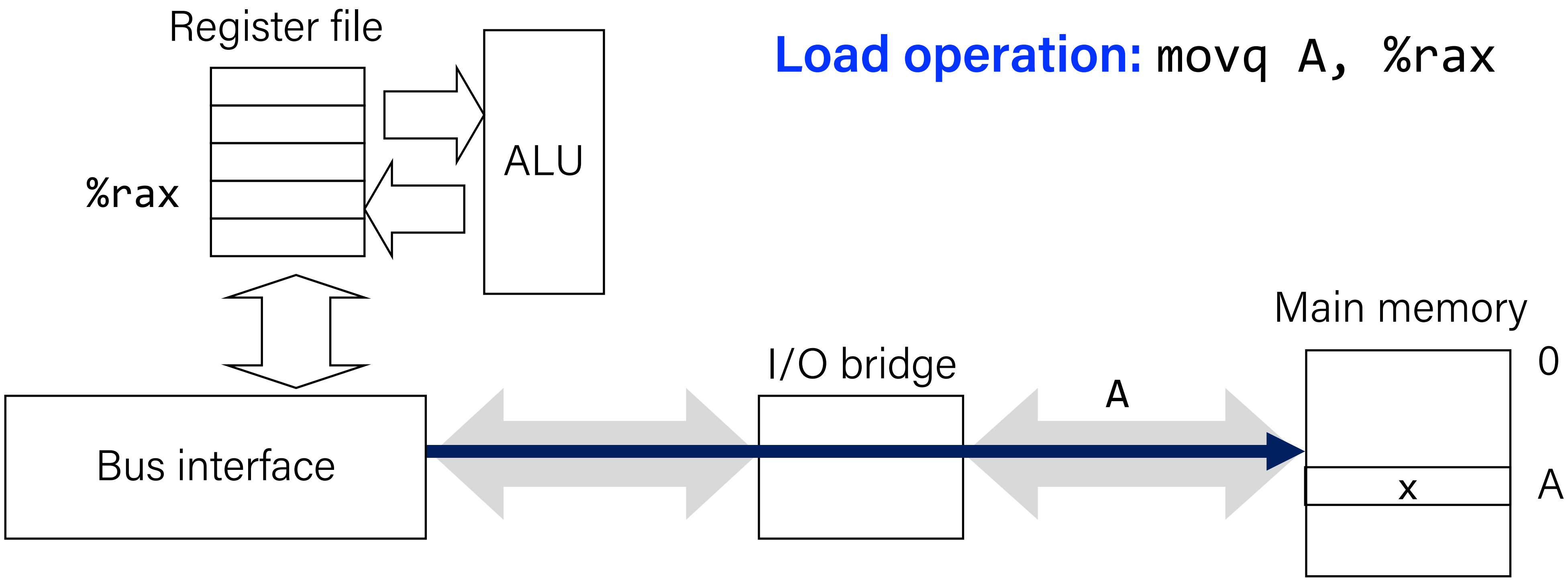
Traditional Bus Structure Connecting CPU and Memory

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



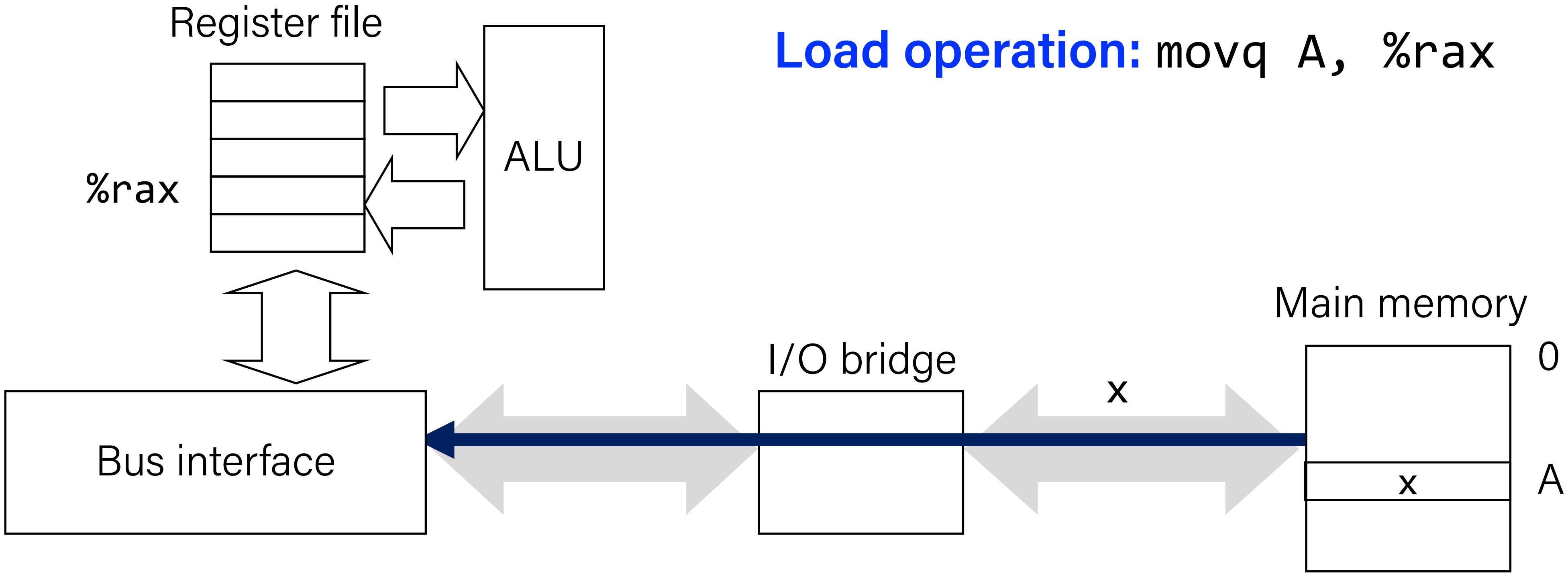
Memory Read Transaction (1)

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



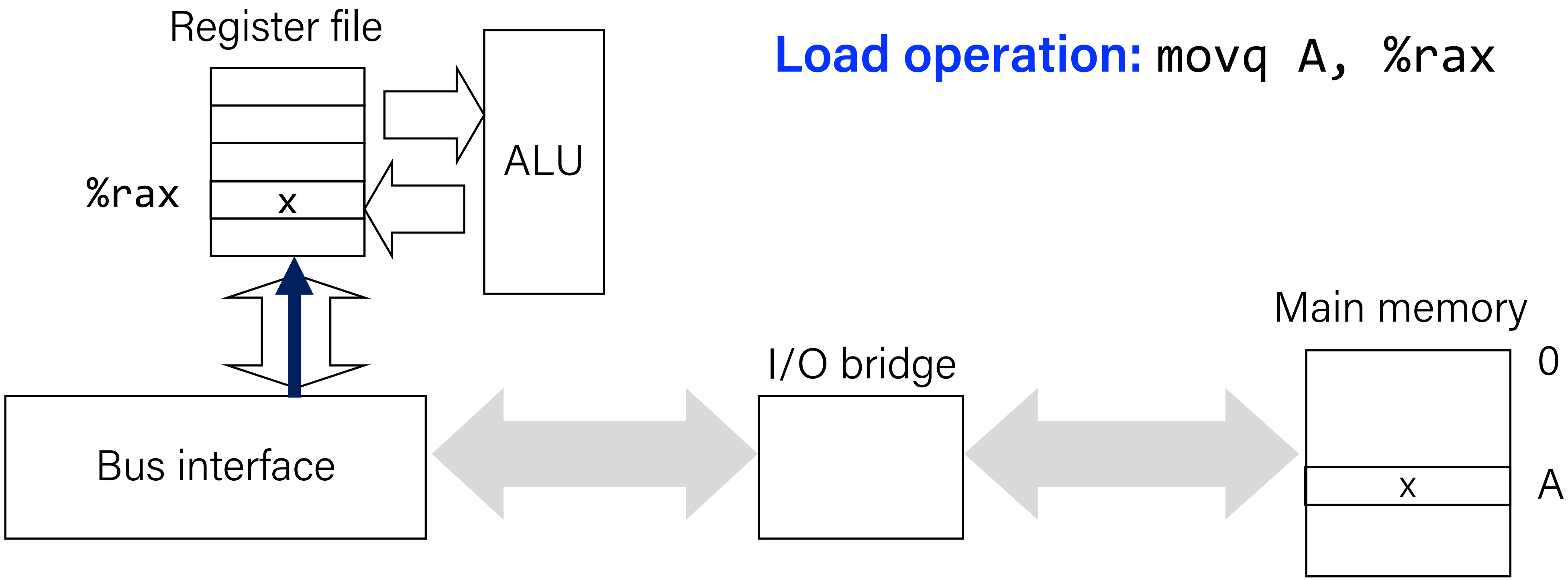
Memory Read Transaction (2)

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



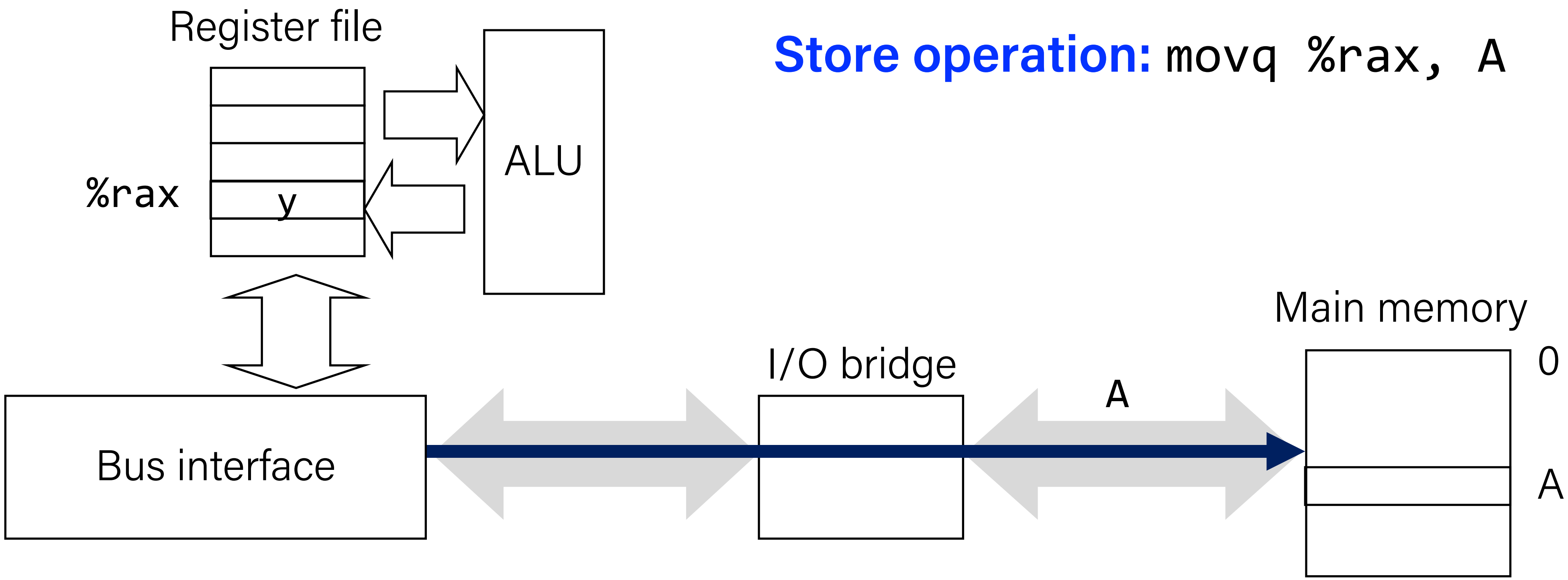
Memory Read Transaction (3)

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



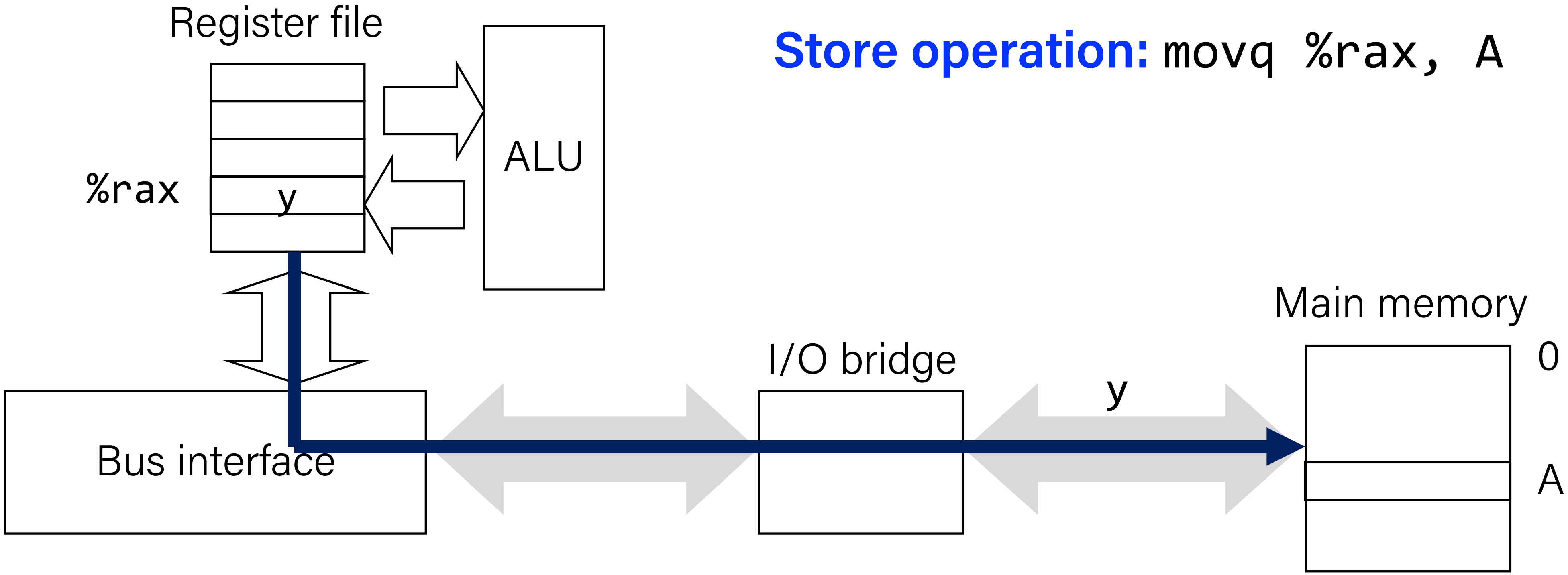
Memory Write Transaction (1)

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



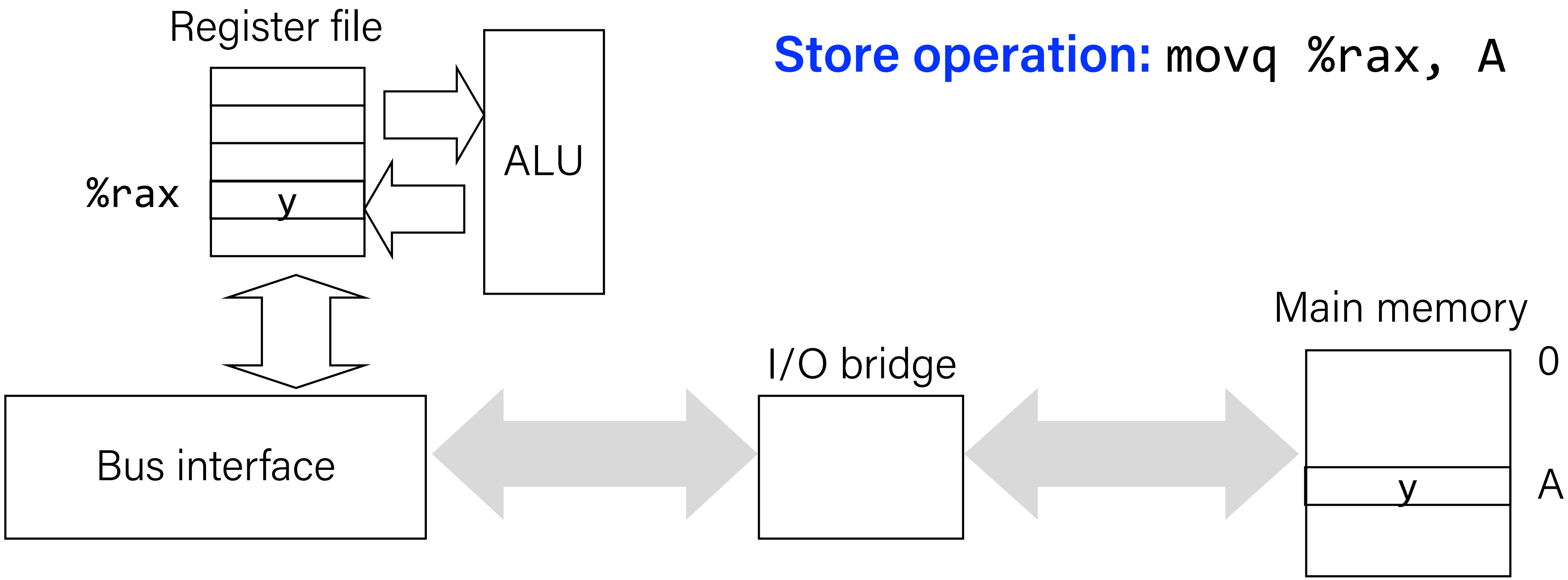
Memory Write Transaction (2)

- 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`.



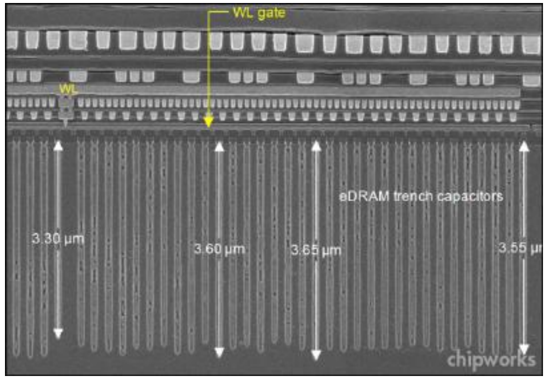
Lecture Plan

- The memory abstraction
- Storage technologies and trends
- Locality of reference
- The memory hierarchy

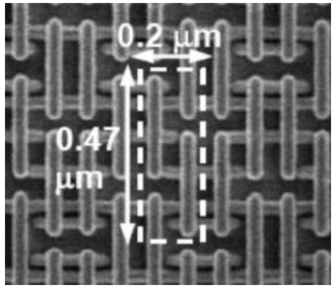
Random-Access Memory (RAM)

- 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.

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



DRAM



SRAM

SRAM vs DRAM Summary

	Trans. per bit	Access time	Needs refresh?	Need 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

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

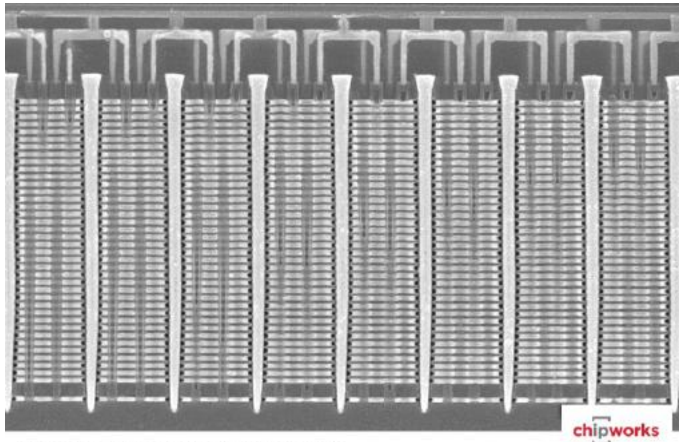
Enhanced DRAMs

- 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

Storage Technologies

- Nonvolatile (Flash) Memory



Close-up image of V-NAND flash array

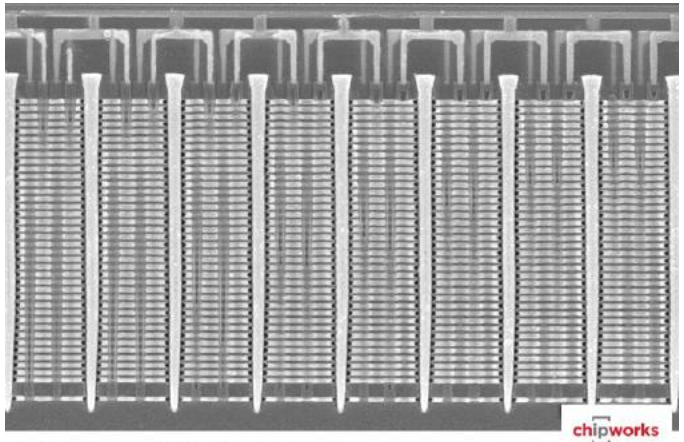
- Store as persistent charge
- Implemented with 3-D structure
  - 100+ levels of cells
  - 3 bits data per cell

- Magnetic Disks

- Store on magnetic medium
- Electromechanical access

Storage Technologies

- 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

- Magnetic Disks

- Store on magnetic medium
- Electromechanical access

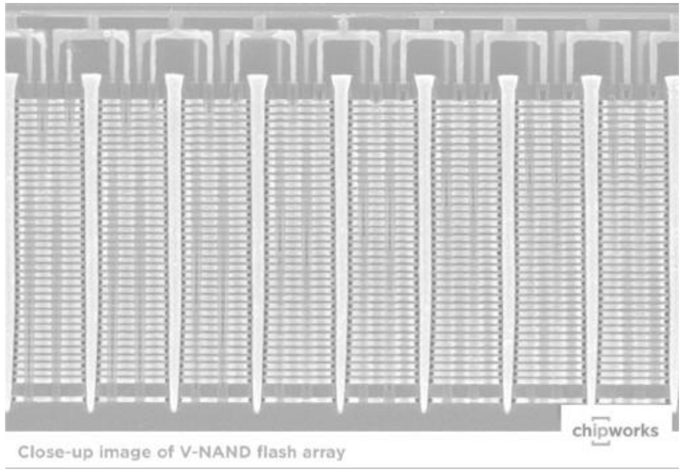
Nonvolatile Memories

- **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 (replace rotating disks in thumb drives, smart phones, mp3 players, tablets, laptops,...)
  - Disk caches

Storage Technologies

- Nonvolatile (Flash) Memory



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

- Magnetic Disks

- Store on magnetic medium
- Electromechanical access

What's Inside A Disk Drive?

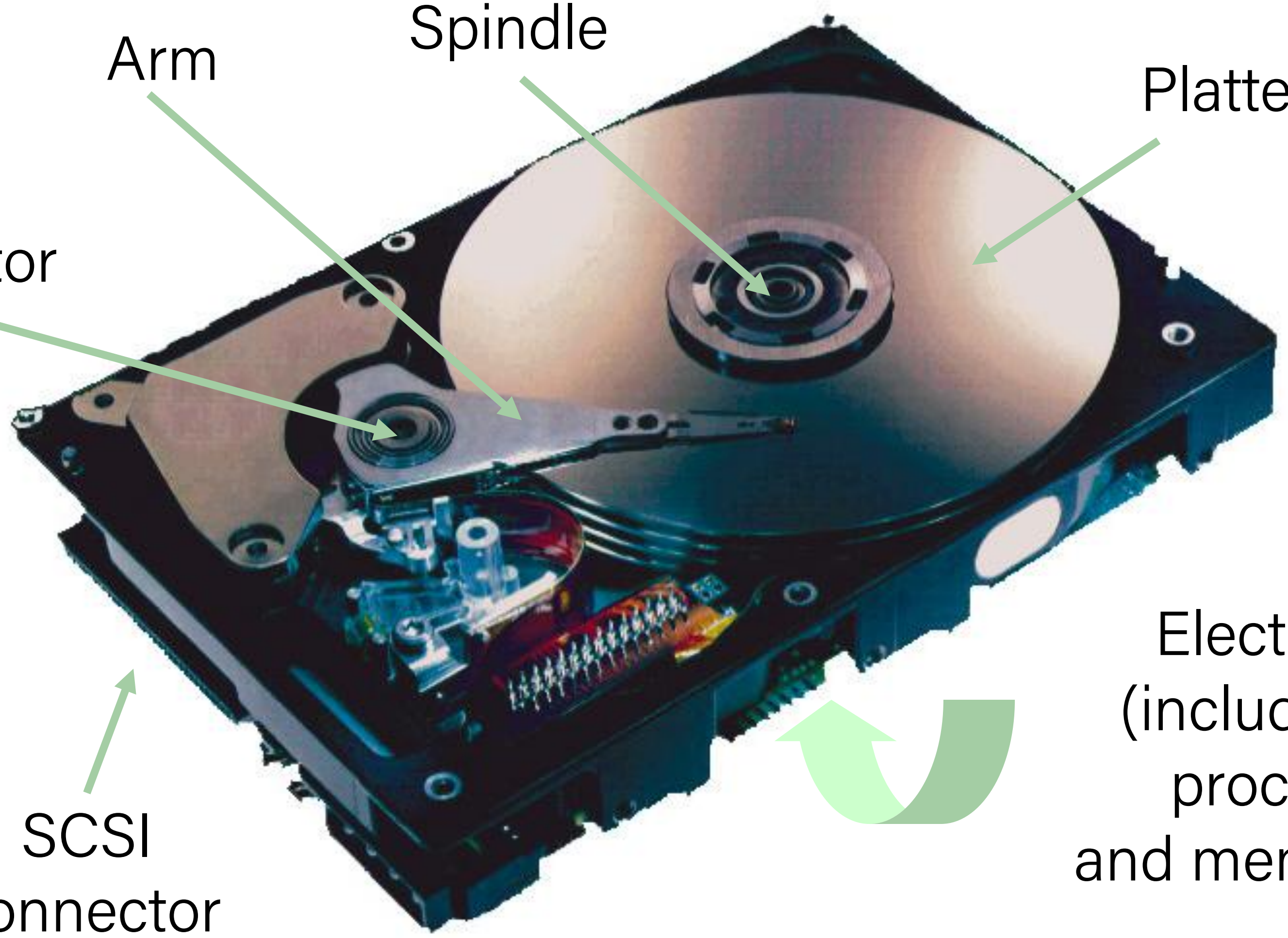
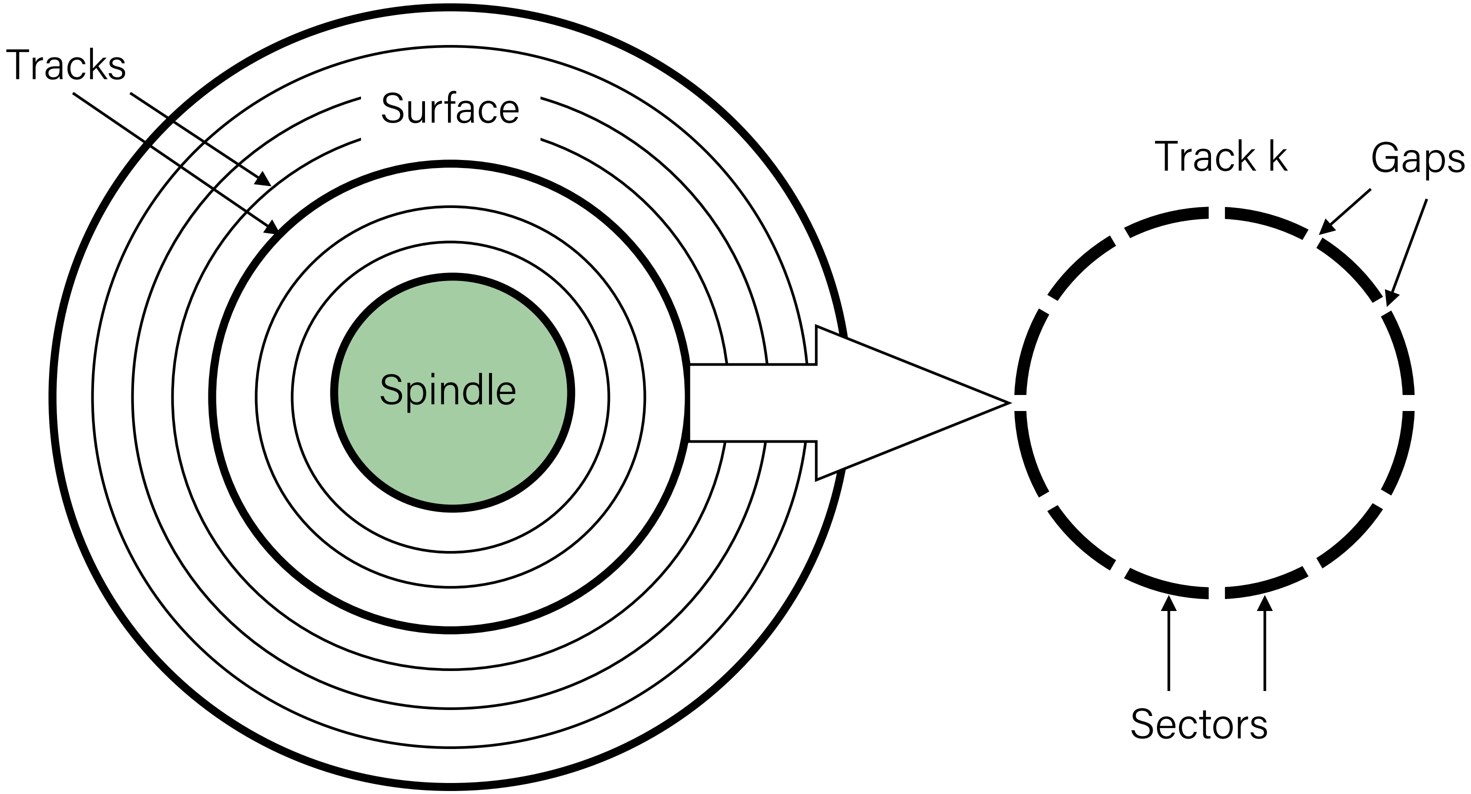


Image courtesy of Seagate Technology

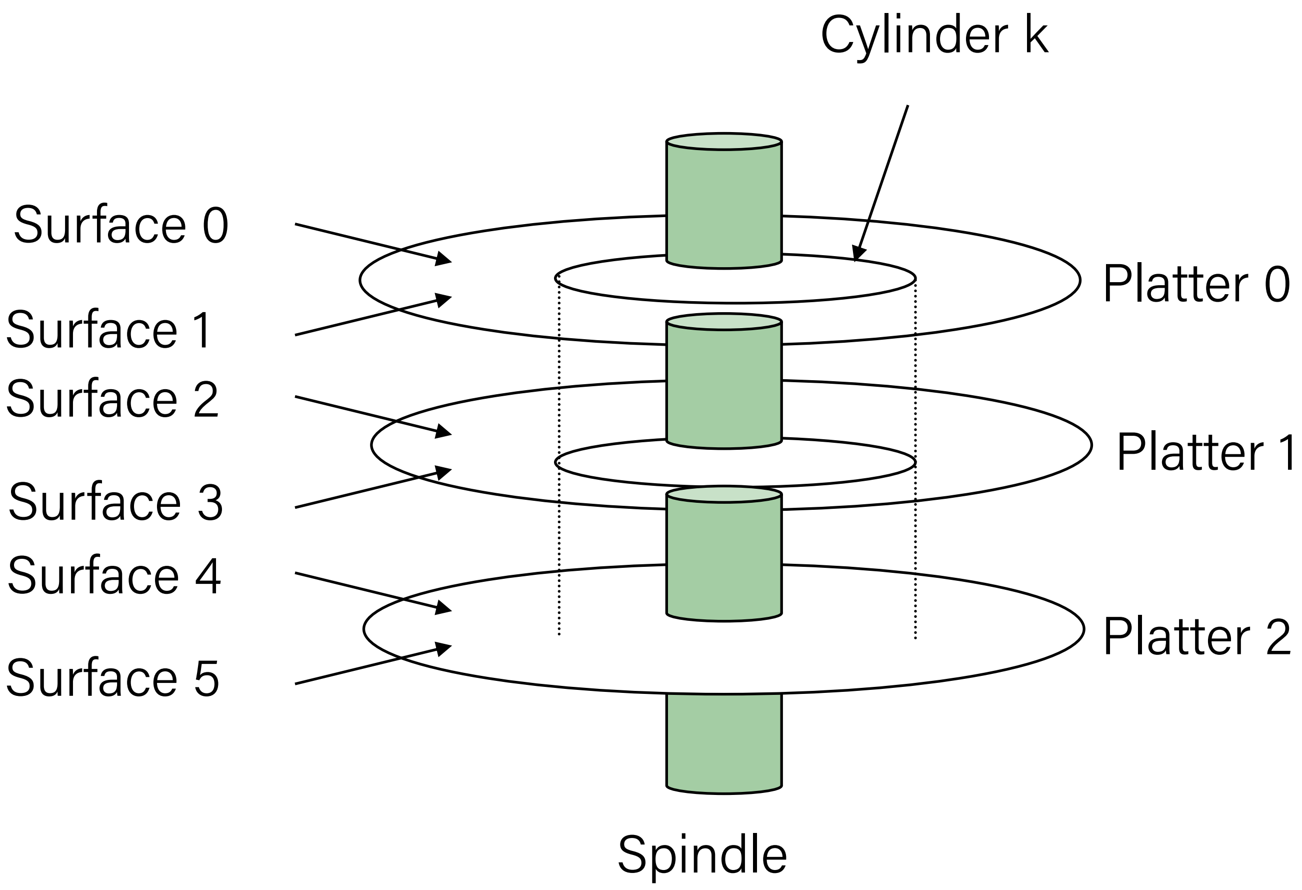
Disk Geometry

- 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 (Multi-Platter View)

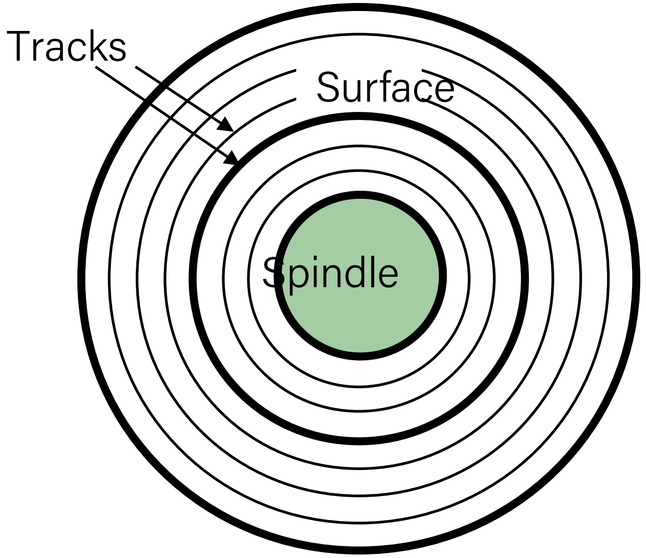
- Aligned tracks form a cylinder.



Disk Capacity

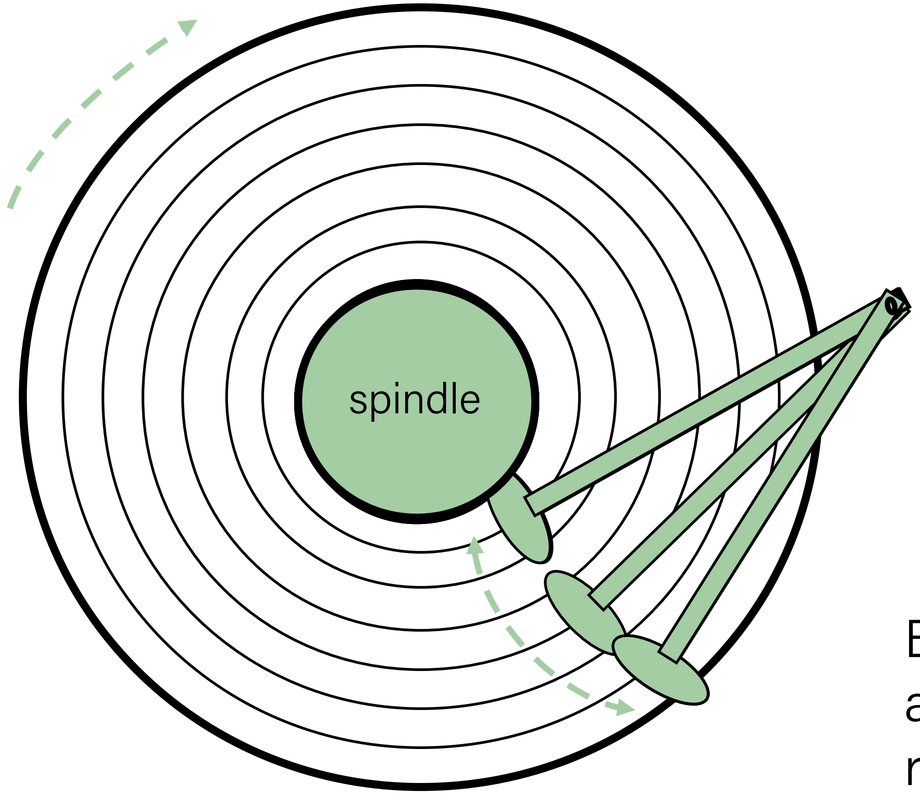
- **Capacity**: maximum number of bits that can be stored.
  - Vendors express capacity in units of gigabytes (GB) or terabytes (TB), where 1 GB = 10^9 bytes and 1 GB = 10^{12} bytes.

- 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/in2): product of recording and track density.



Disk Operation (Single-Platter View)

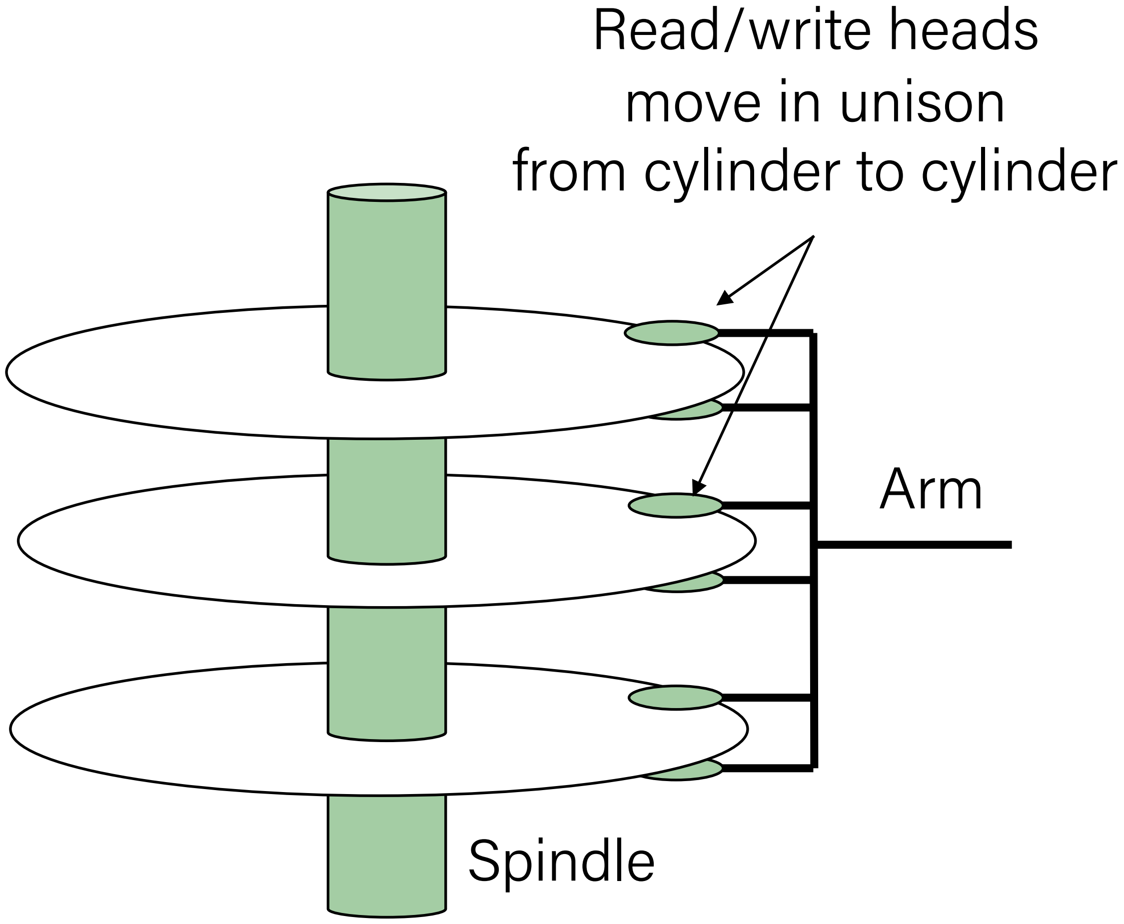
The disk surface spins at a fixed rotational rate



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

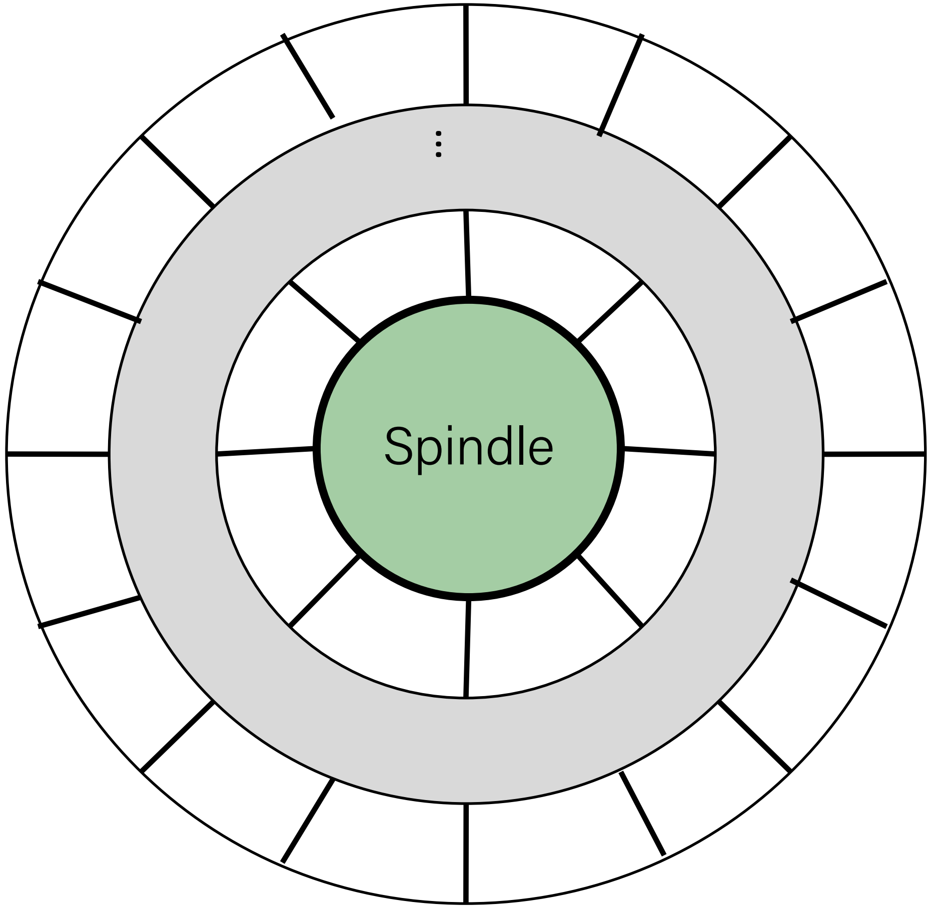
By moving radially, the arm can position the read/write head over any track.

Disk Operation (Multi-Platter View)



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:

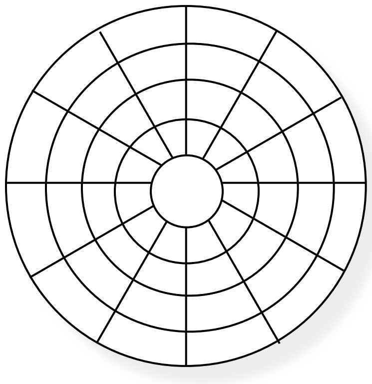
- 512 bytes/sector
- 300 sectors/track (on average)
- 20,000 tracks/surface
- 2 surfaces/platter
- 5 platters/disk

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

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”.

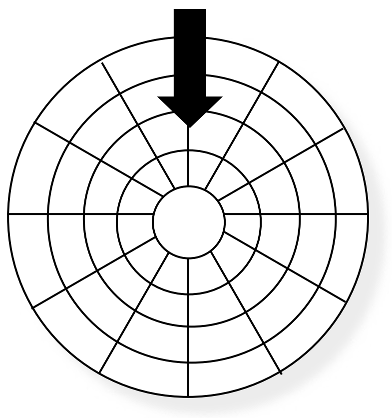
Disk Structure - top view of single platter



Surface organized into tracks

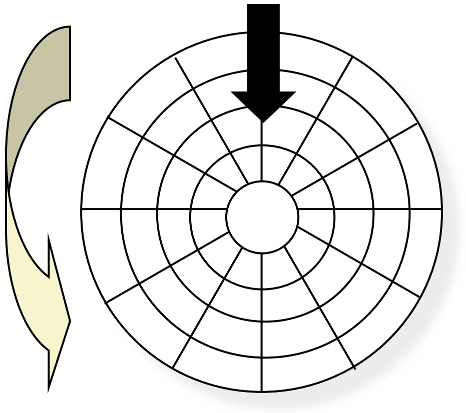
Tracks divided into sectors

Disk Access



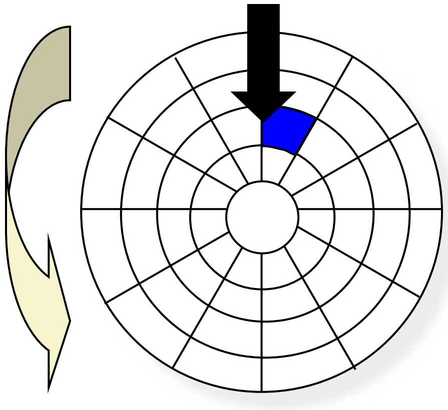
Head in position above a track

Disk Access



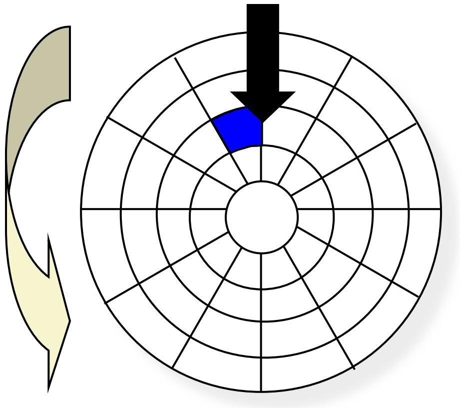
Rotation is counter-clockwise

Disk Access – Read



About to read blue sector

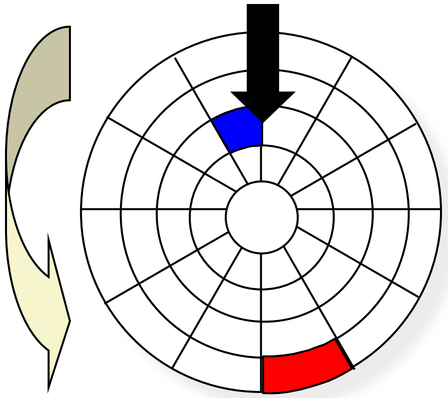
Disk Access – Read



After BLUE read

After reading blue sector

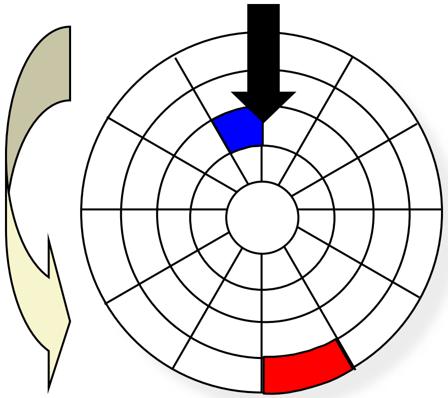
Disk Access – Read



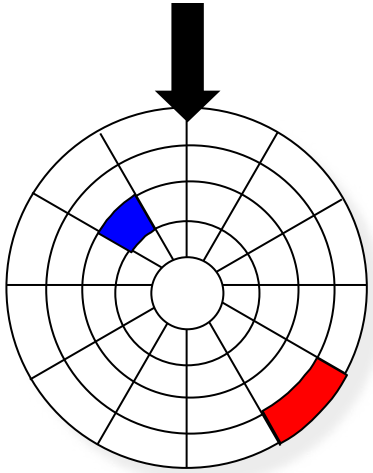
After BLUE read

Red request scheduled next

Disk Access – Seek



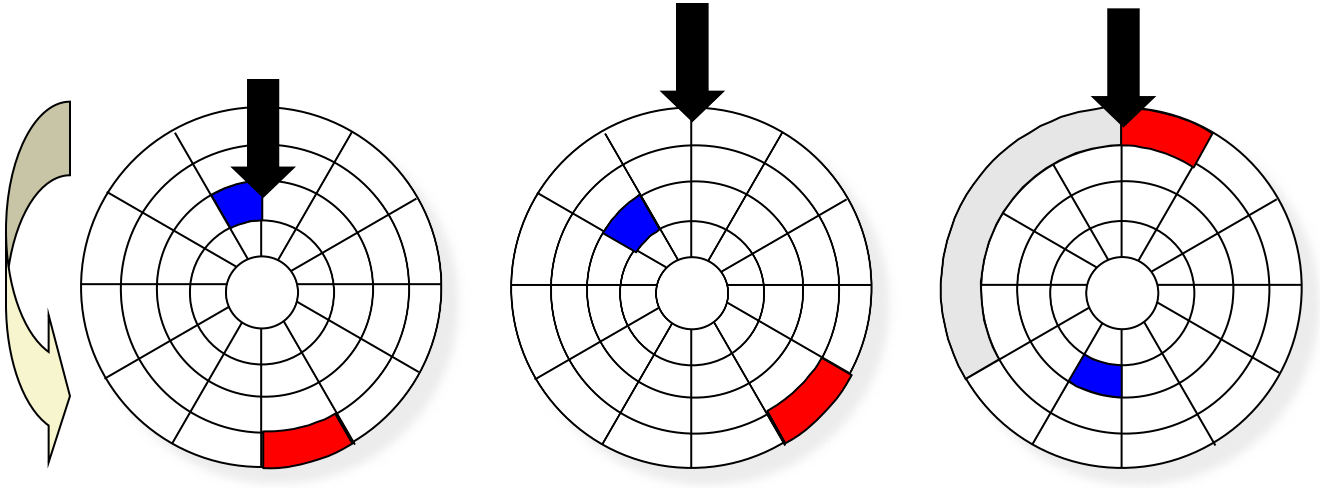
After BLUE read



Seek for RED

Seek to red’s track

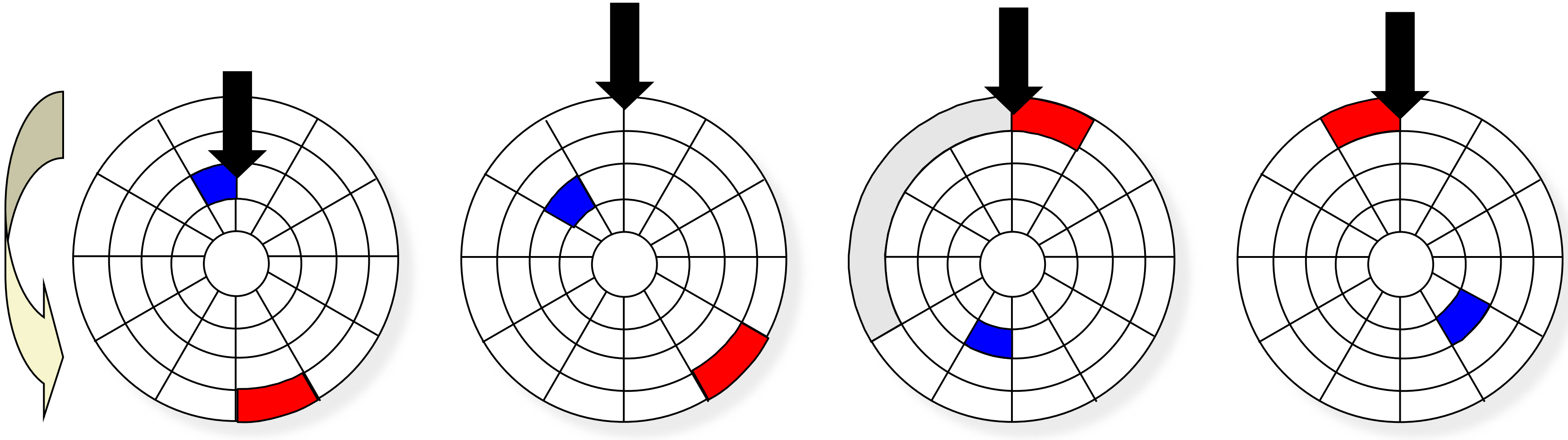
Disk Access – Rotational Latency



After BLUE read Seek for RED Rotational latency

Wait for red sector to rotate around

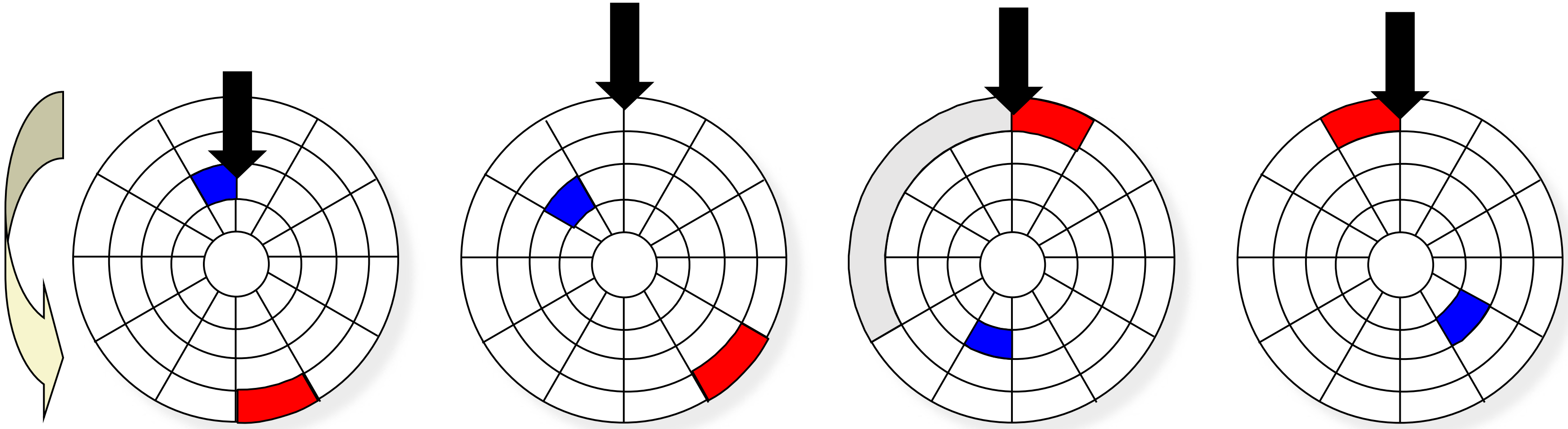
Disk Access – Read



After BLUE read Seek for RED Rotational latency After RED read

Complete read of red

Disk Access – Service Time Components



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

Data transfer | Seek | Rotational latency | Data transfer

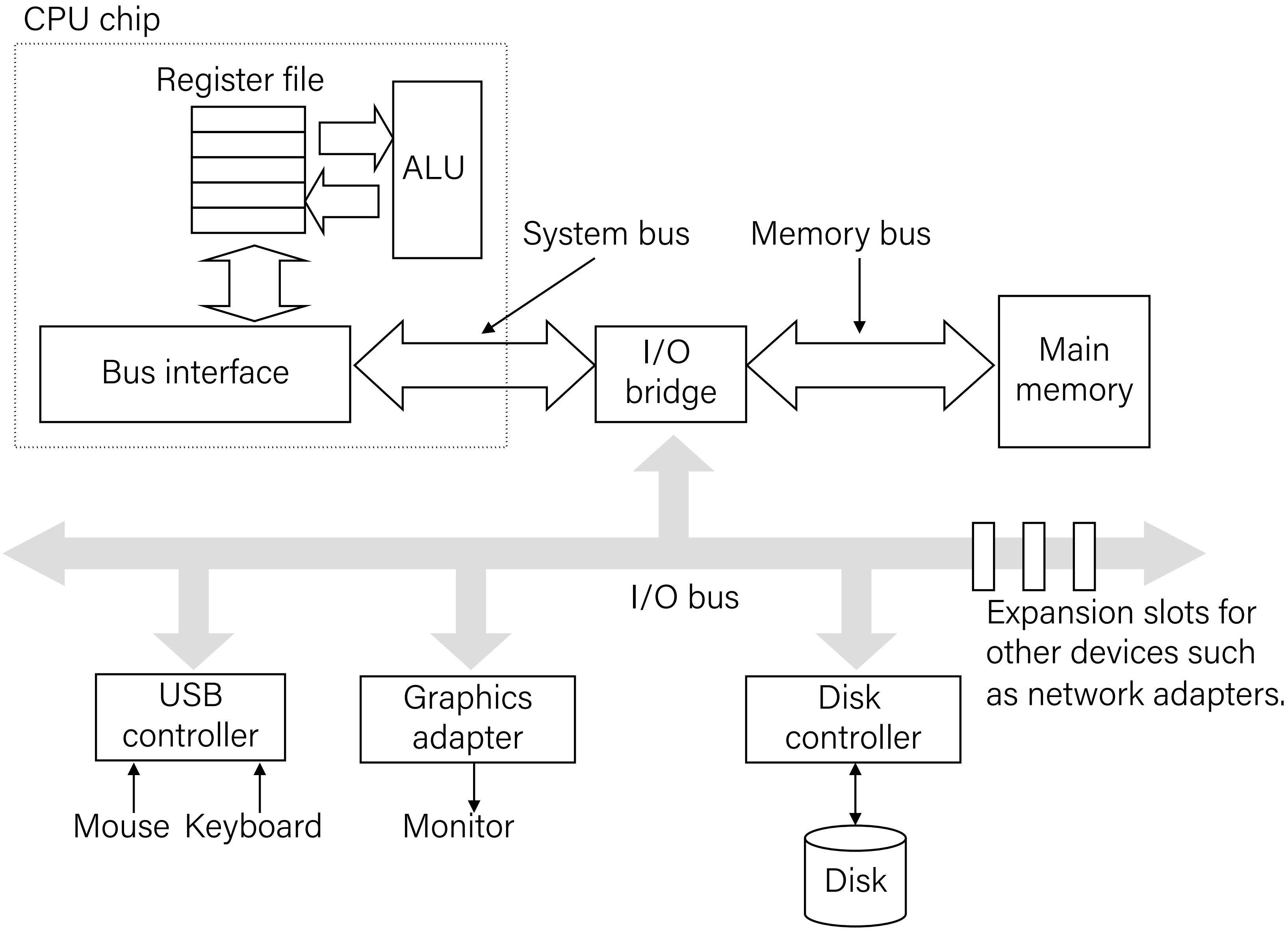
Disk Access Time

- Average time to access some target sector approximated by:
  - Taccess = Tavg seek + Tavg rotation + Tavg transfer
- Seek time (Tavg seek)
  - Time to position heads over cylinder containing target sector.
  - Typical Tavg seek is 3—9 ms
- Rotational latency (Tavg rotation)
  - Time waiting for first bit of target sector to pass under r/w head.
  - Tavg rotation = 1/2 × 1/RPMs x 60 sec/1 min
  - Typical Tavg rotation = 7200 RPMs
- Transfer time (Tavg transfer)
  - Time to read the bits in the target sector.
  - Tavg transfer = 1/RPM × 1/(avg # sectors/track) × 60 secs/1 min.

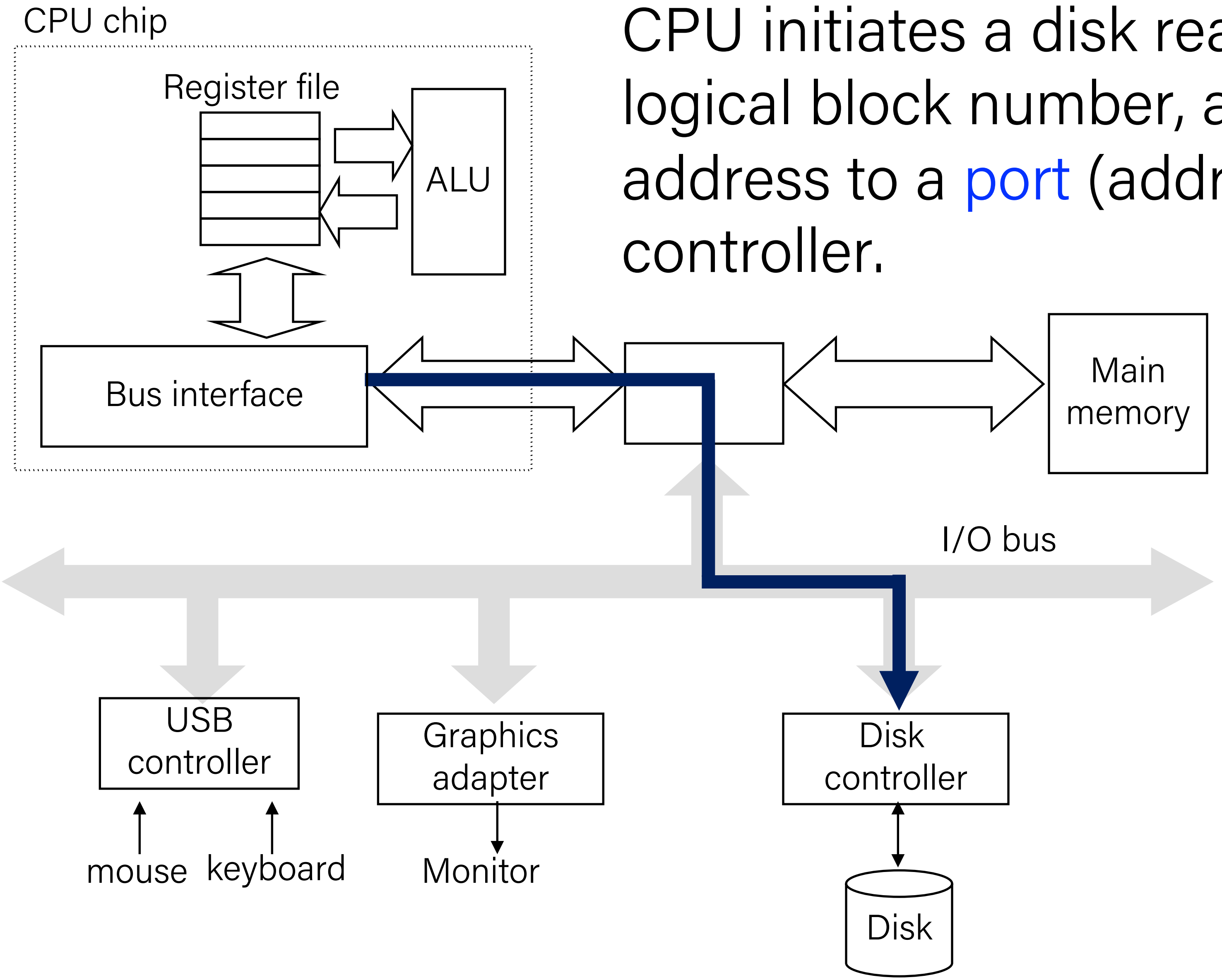
Disk Access Time Example

- Given:
  - Rotational rate = 7,200 RPM
  - Average seek time = 9 ms.
  - Avg # sectors/track = 400.
- Derived:
  - Tavg rotation = 1/2 × (60 secs/7200 RPM) × 1000 ms/sec = 4 ms.
  - Tavg transfer = 60/7200 RPM × 1/400 secs/track × 1000 ms/sec = 0.02 ms
  - Taccess = 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

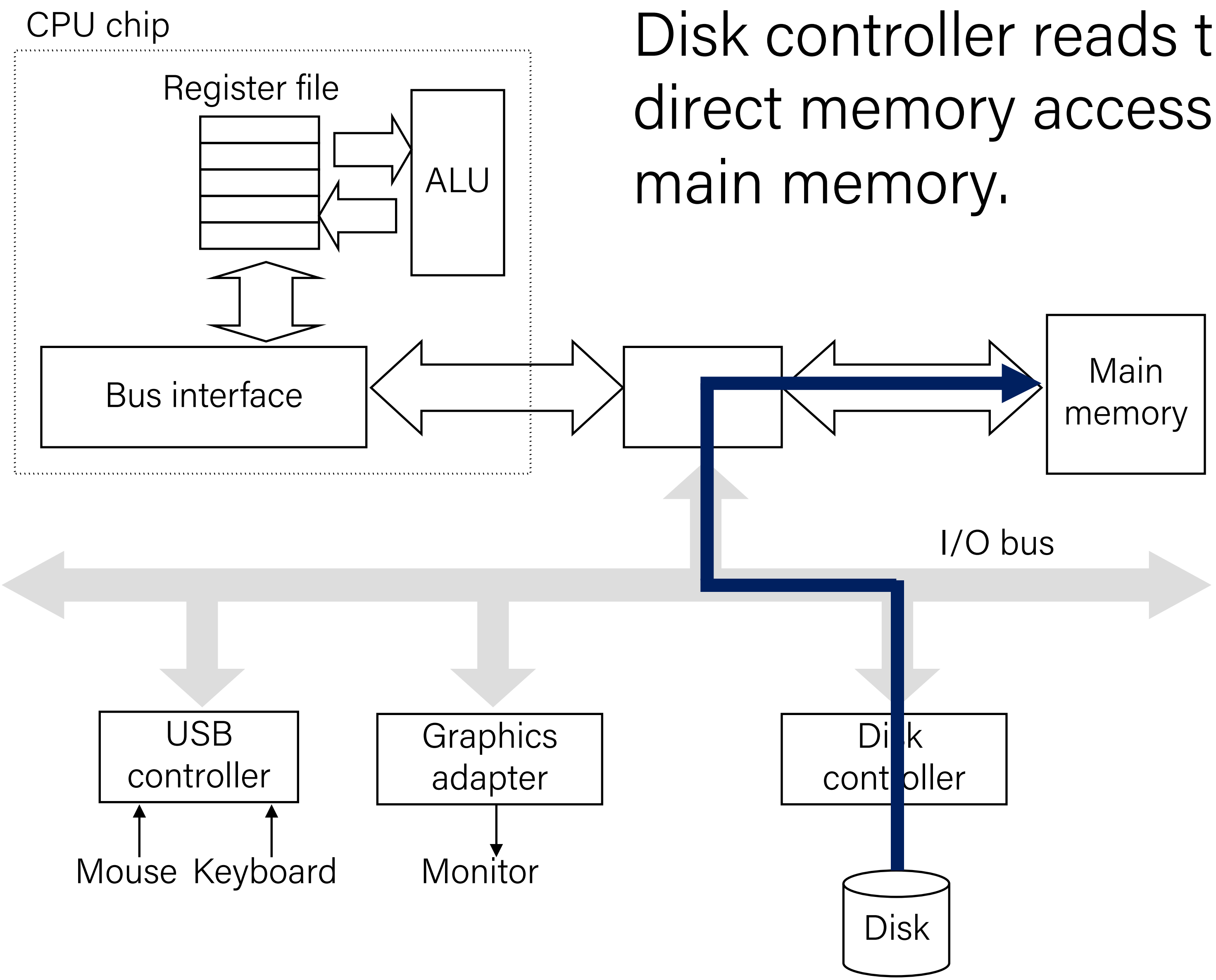
Reading a Disk Sector (1)



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

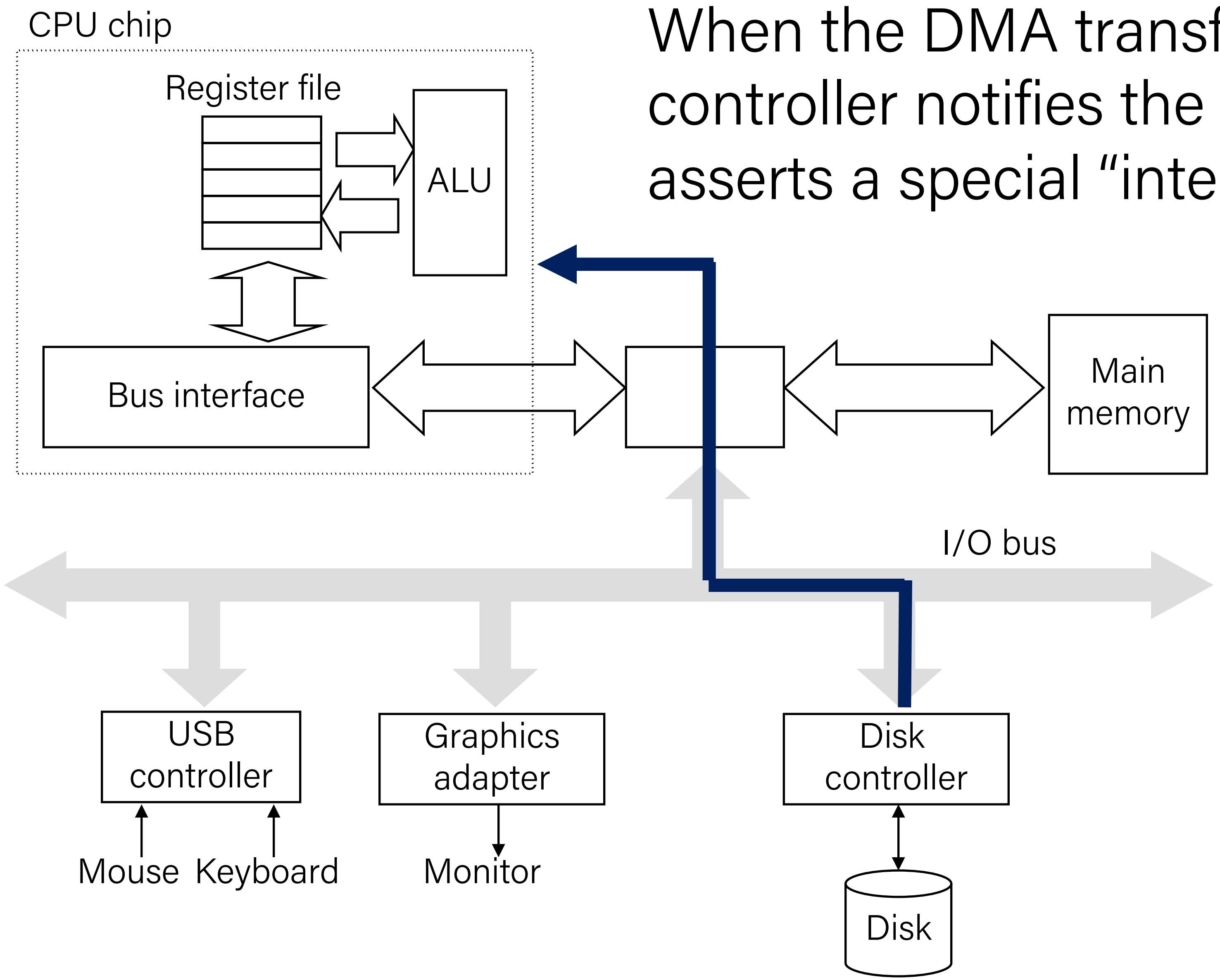
Reading a Disk Sector (2)

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

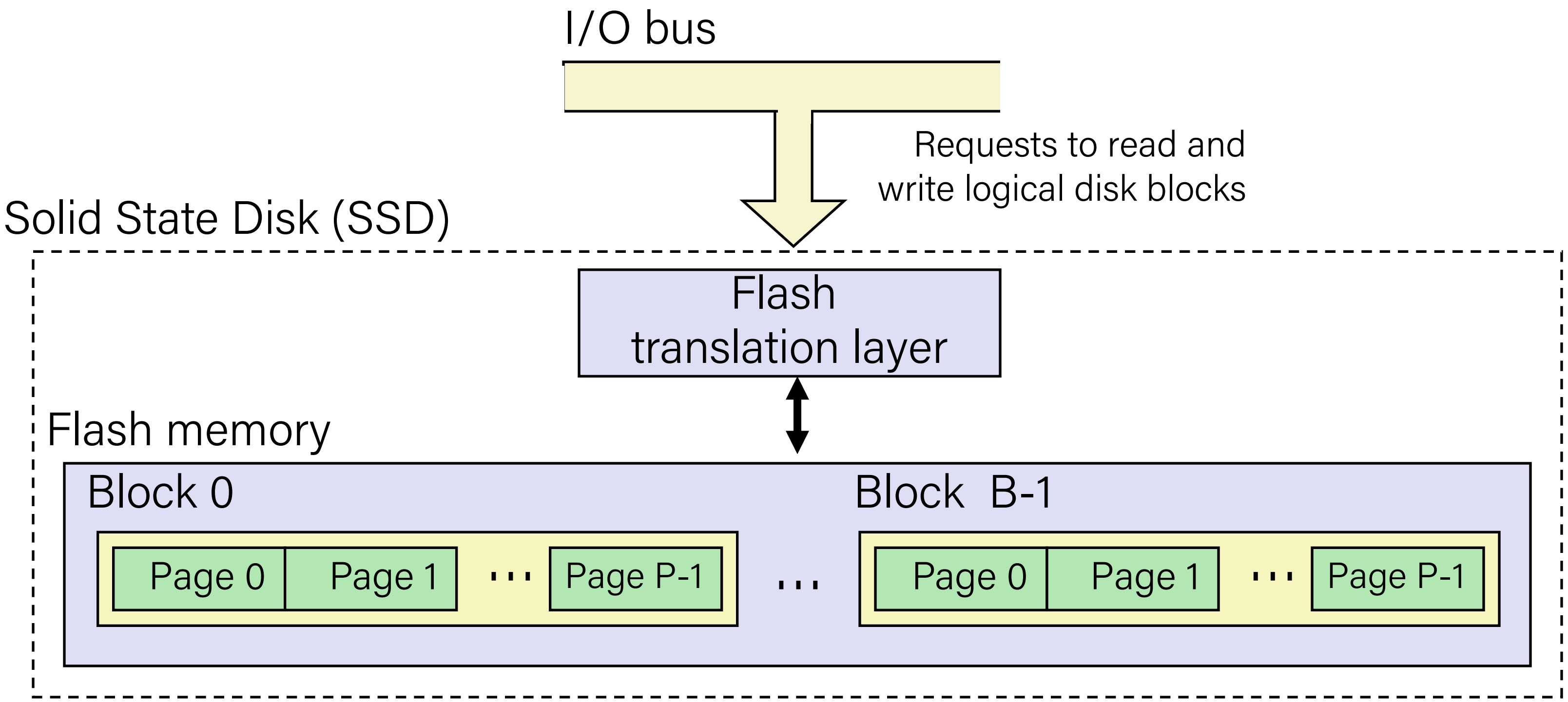


Reading a Disk Sector (3)

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



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

- Benchmark of Samsung 940 EVO Plus

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

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

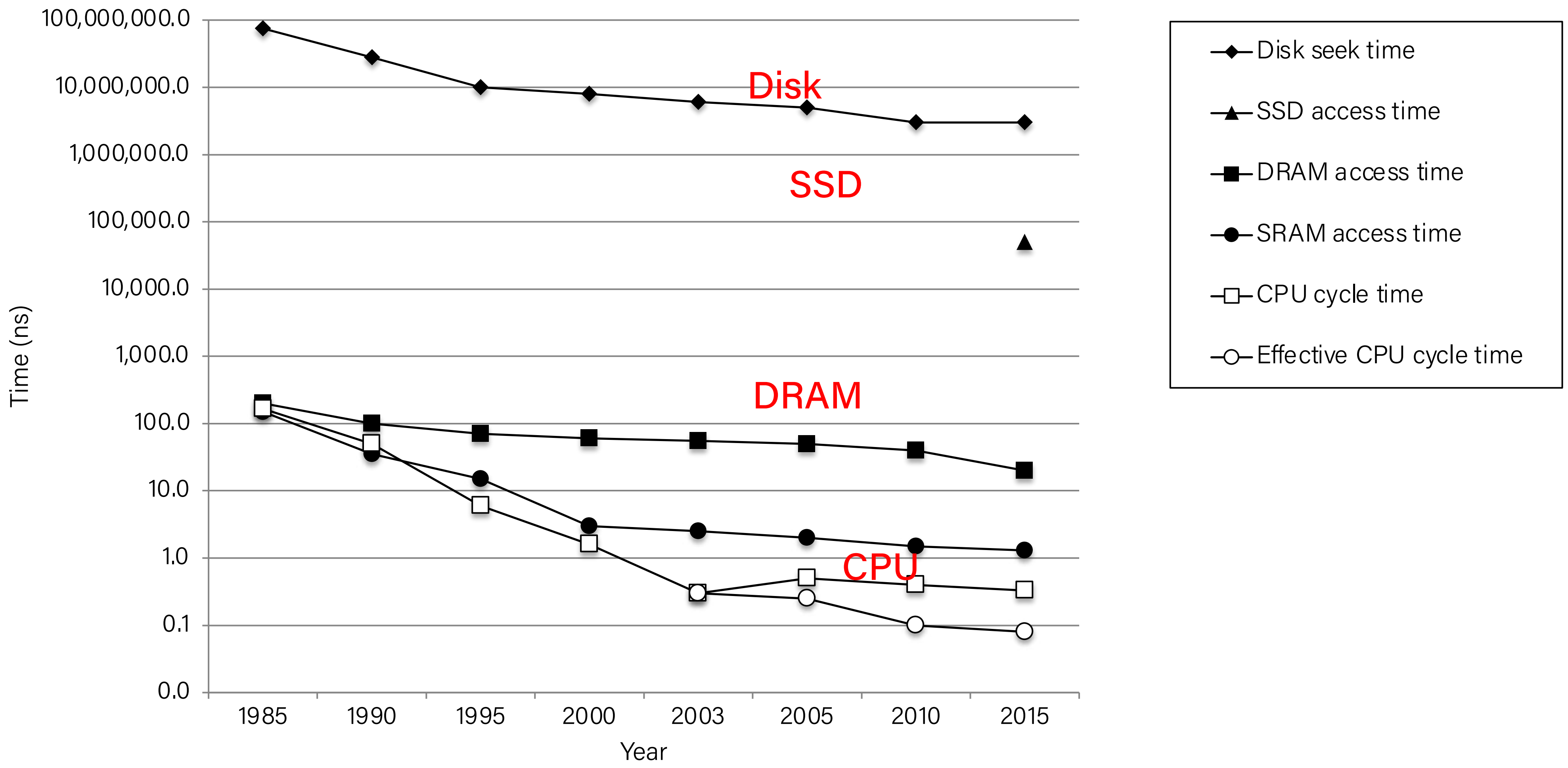
- Sequential access faster than random access
  - Common theme in the memory hierarchy
- Random writes are somewhat slower
  - Erasing a block takes a long time (~1 ms)
  - Modifying a block page requires all other pages to be copied to new block
  - Flash translation layer allows accumulating series 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 2019, about 4 times more expensive per byte
    - And, relative cost will keep dropping
- **Applications**
  - MP3 players, smart phones, laptops
  - Beginning to appear in desktops and servers

The CPU-Memory Gap

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



Locality to the Rescue!

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

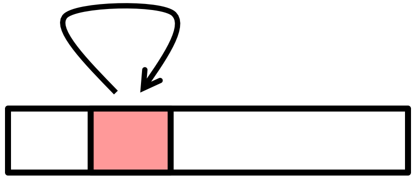
Lecture Plan

- The memory abstraction
- Storage technologies and trends
- Locality of reference
- The memory hierarchy

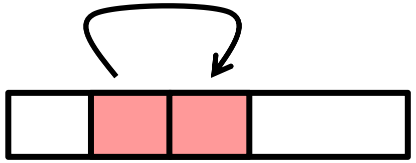
Locality

- **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 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

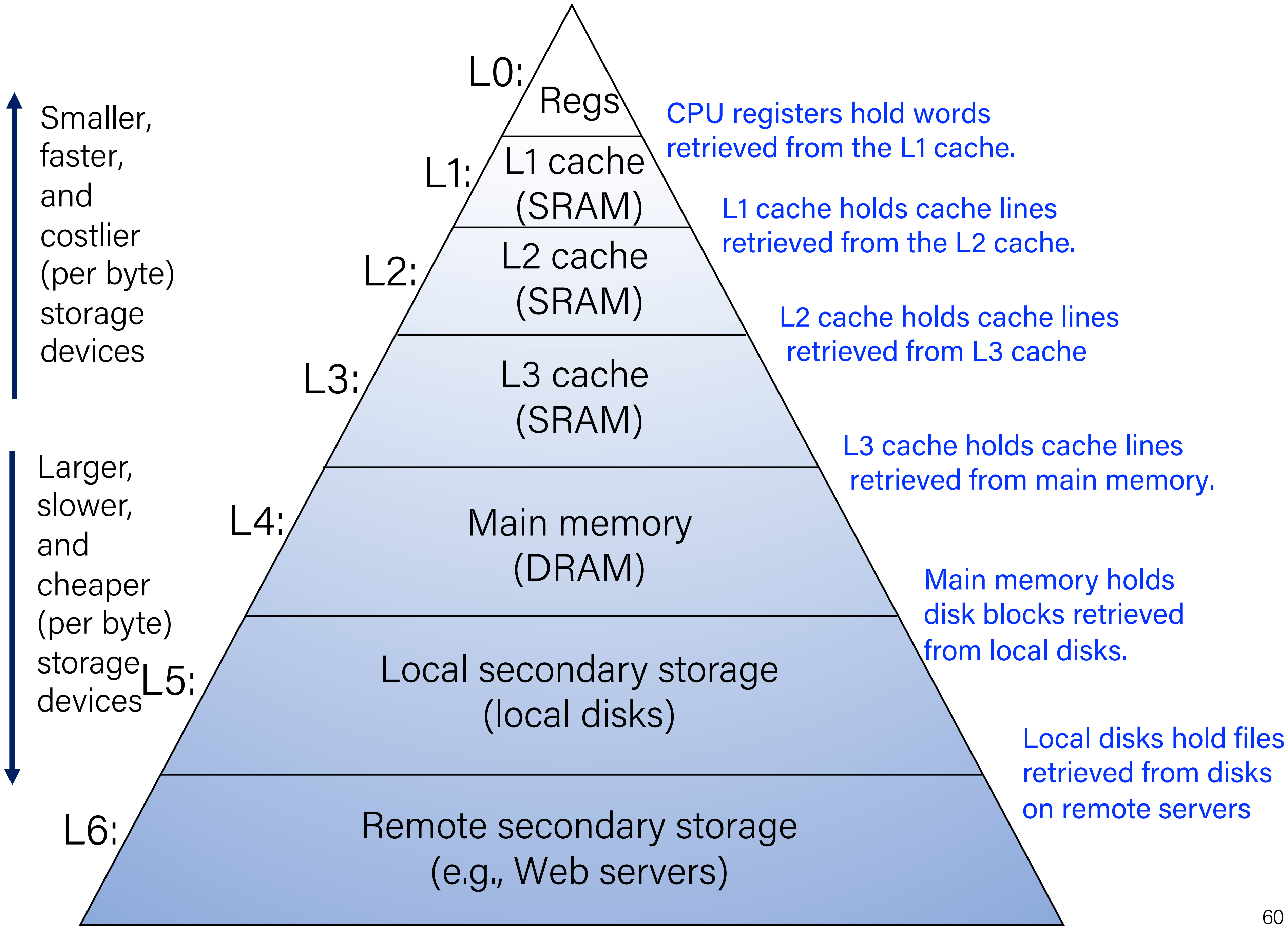
Lecture Plan

- The memory abstraction
- 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.
    - True for: registers ↔ cache, cache ↔ DRAM, DRAM ↔ disk, etc.
  - 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**.
  - For each level k, the faster, smaller device at level k serves as a cache for the larger, slower device at level k+1

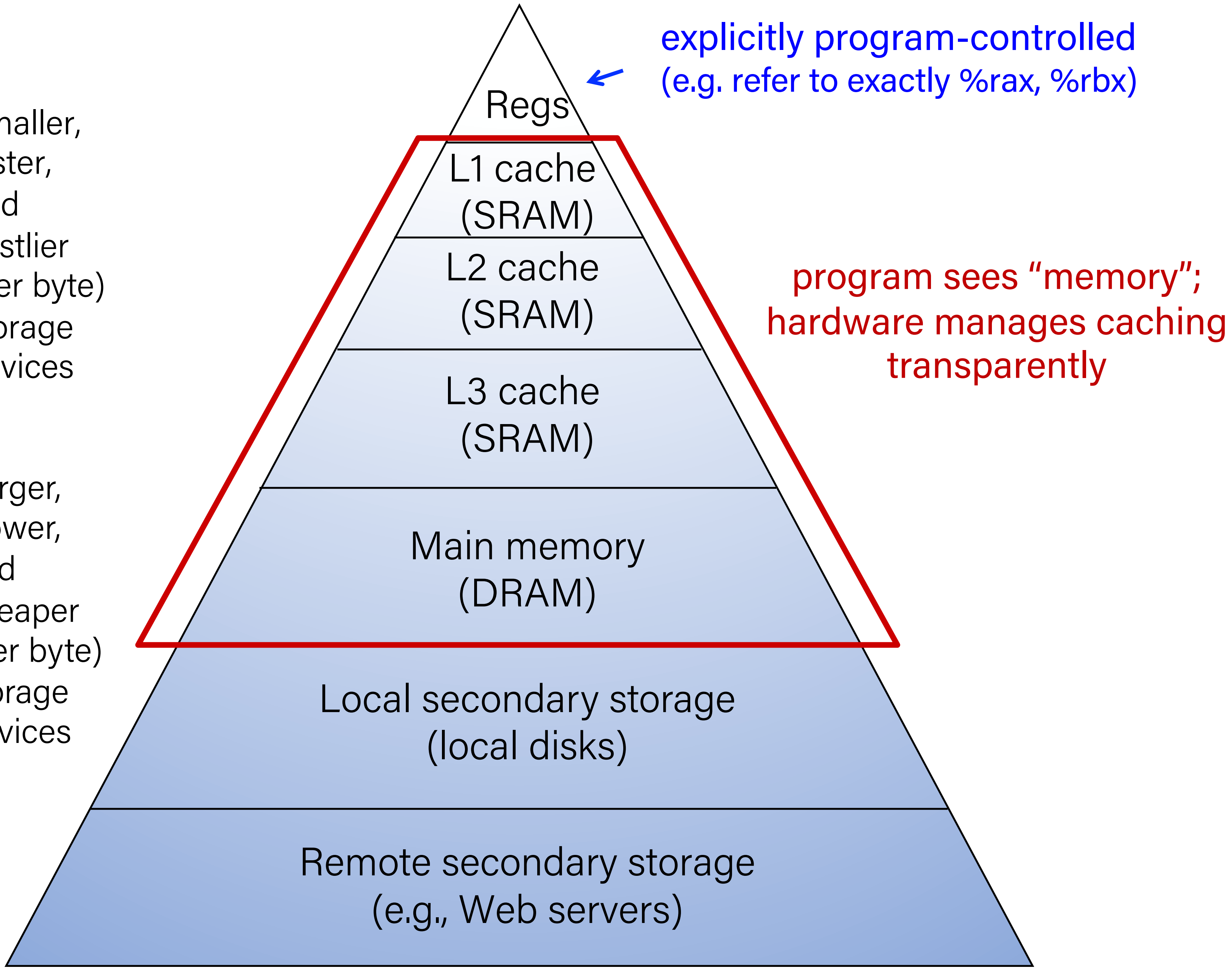
Example Memory Hierarchy



Example Memory Hierarchy

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

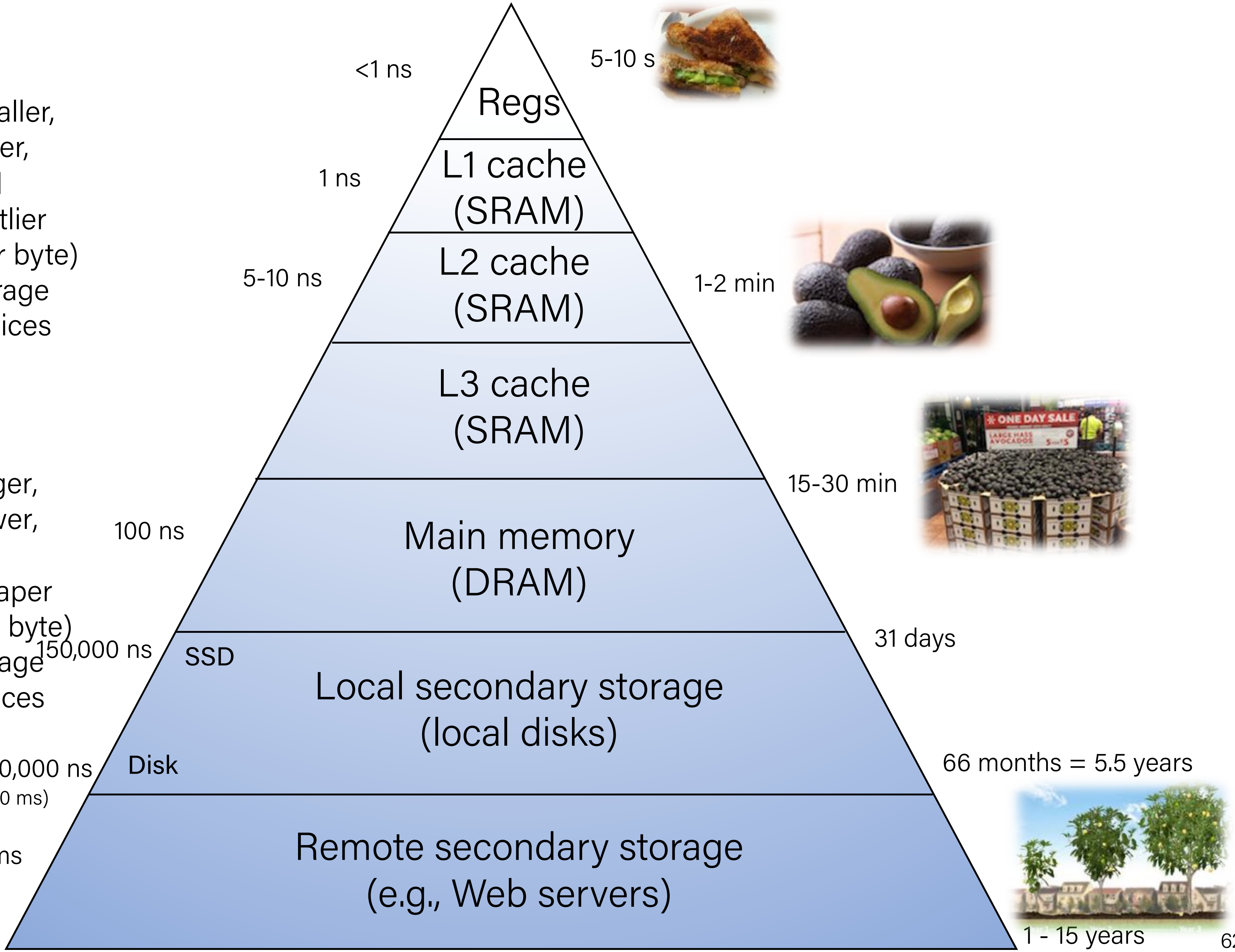
Larger,
slower,
and
cheaper
(per byte)
storage
devices



Example Memory Hierarchy

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

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

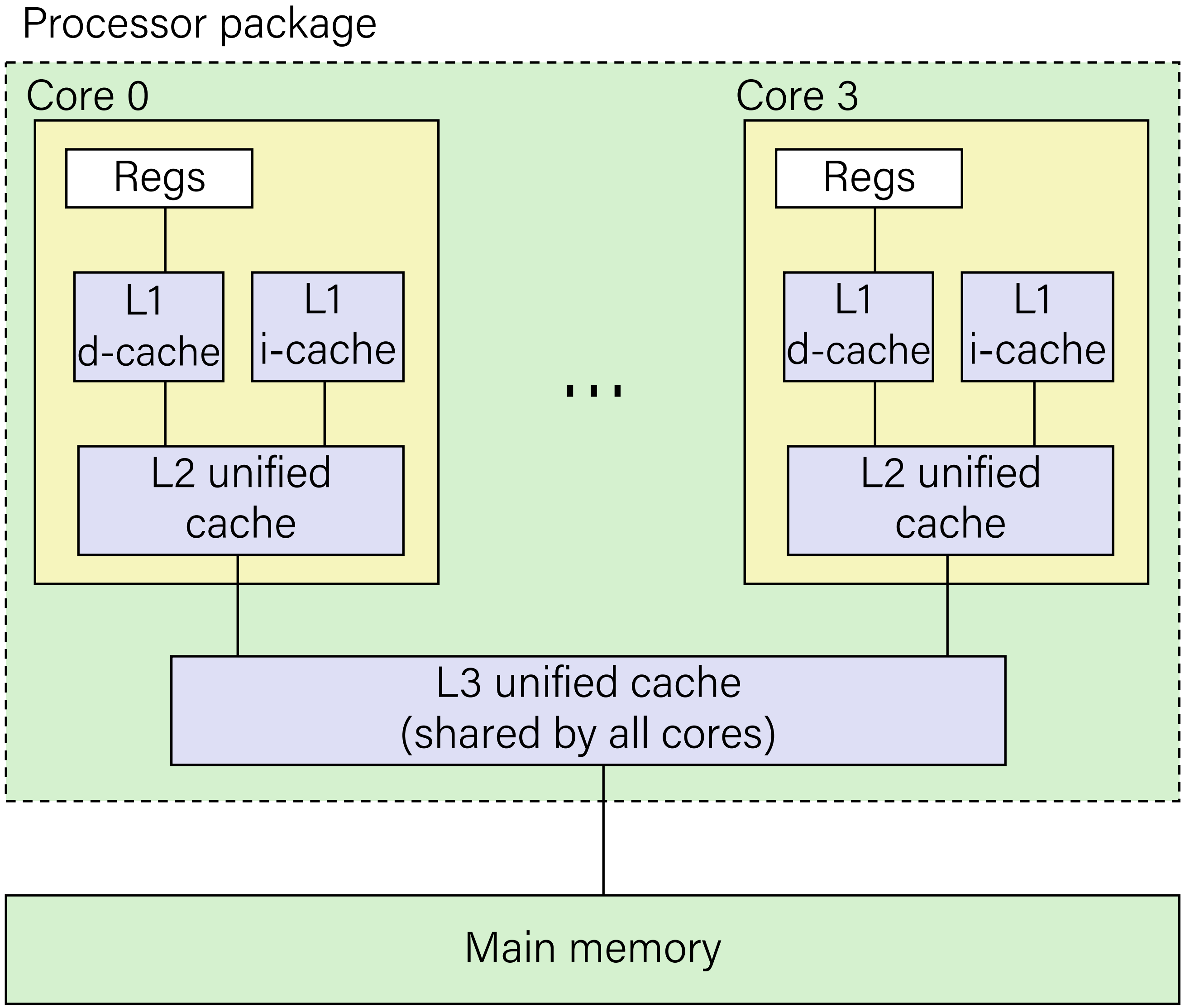


Caches

- **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: 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.

Intel Core i7 Cache Hierarchy



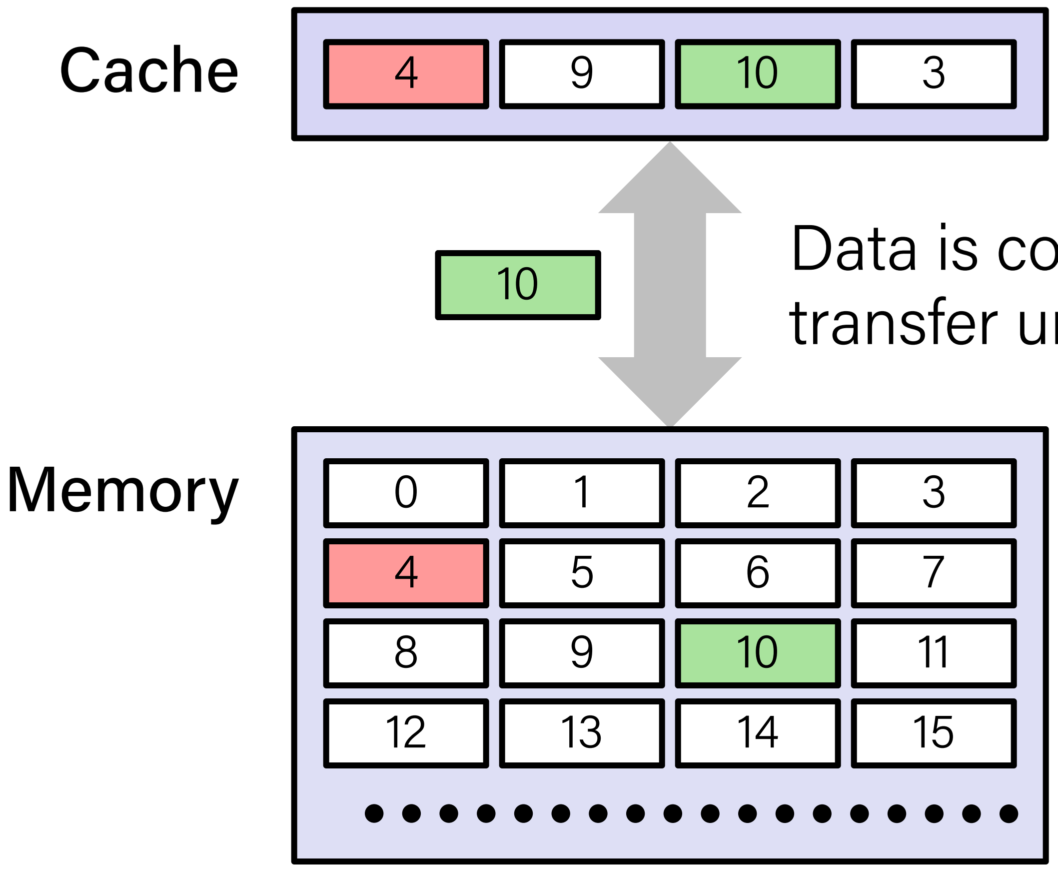
L1 i-cache and d-cache:
32 KB, 8-way,
Access: 4 cycles

L2 unified cache:
256 KB, 8-way,
Access: 10 cycles

L3 unified cache:
8 MB, 16-way,
Access: 40-75 cycles

Block size: 64 bytes for all caches.

General Cache Concepts

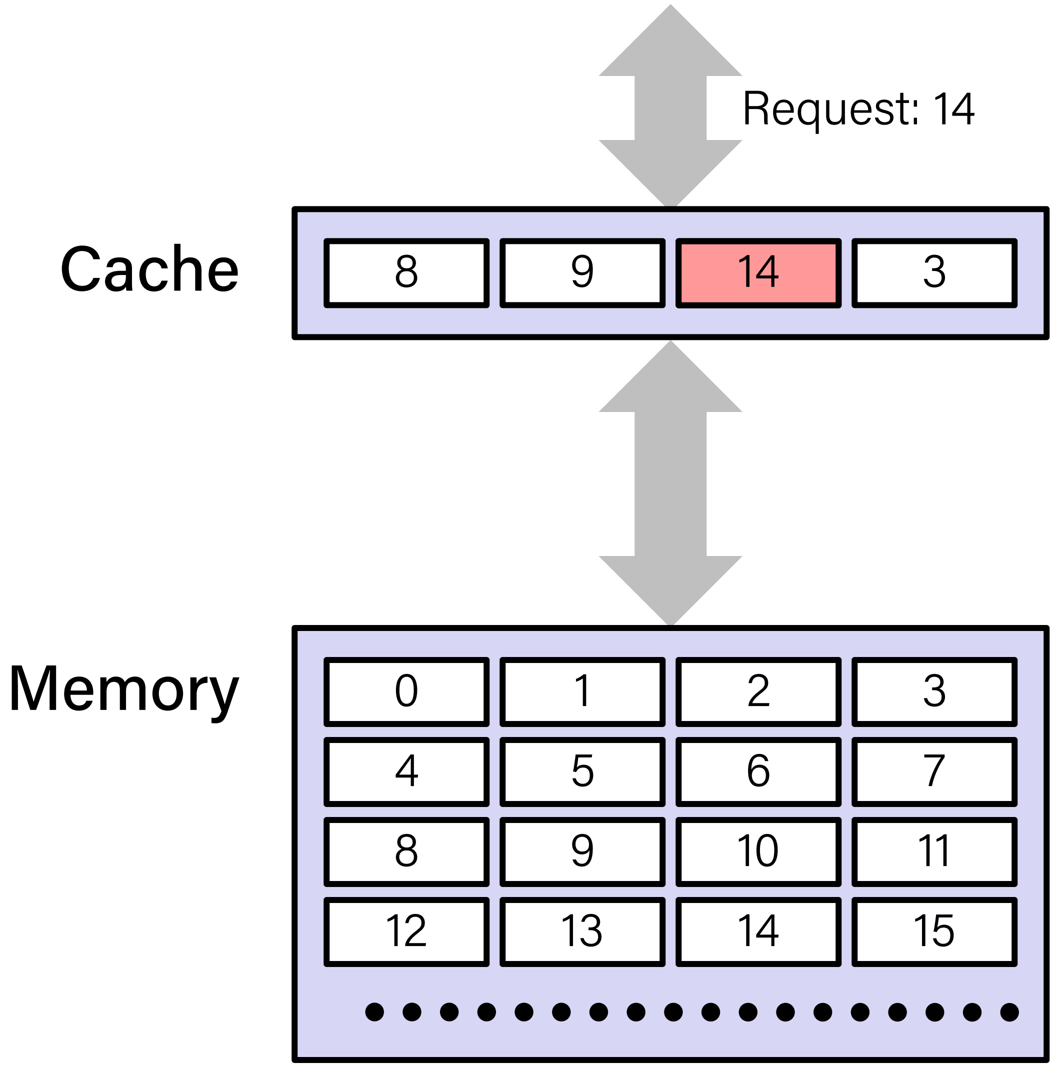


Smaller, faster, more expensive memory caches a subset of the blocks

Data is copied in block-sized transfer units

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

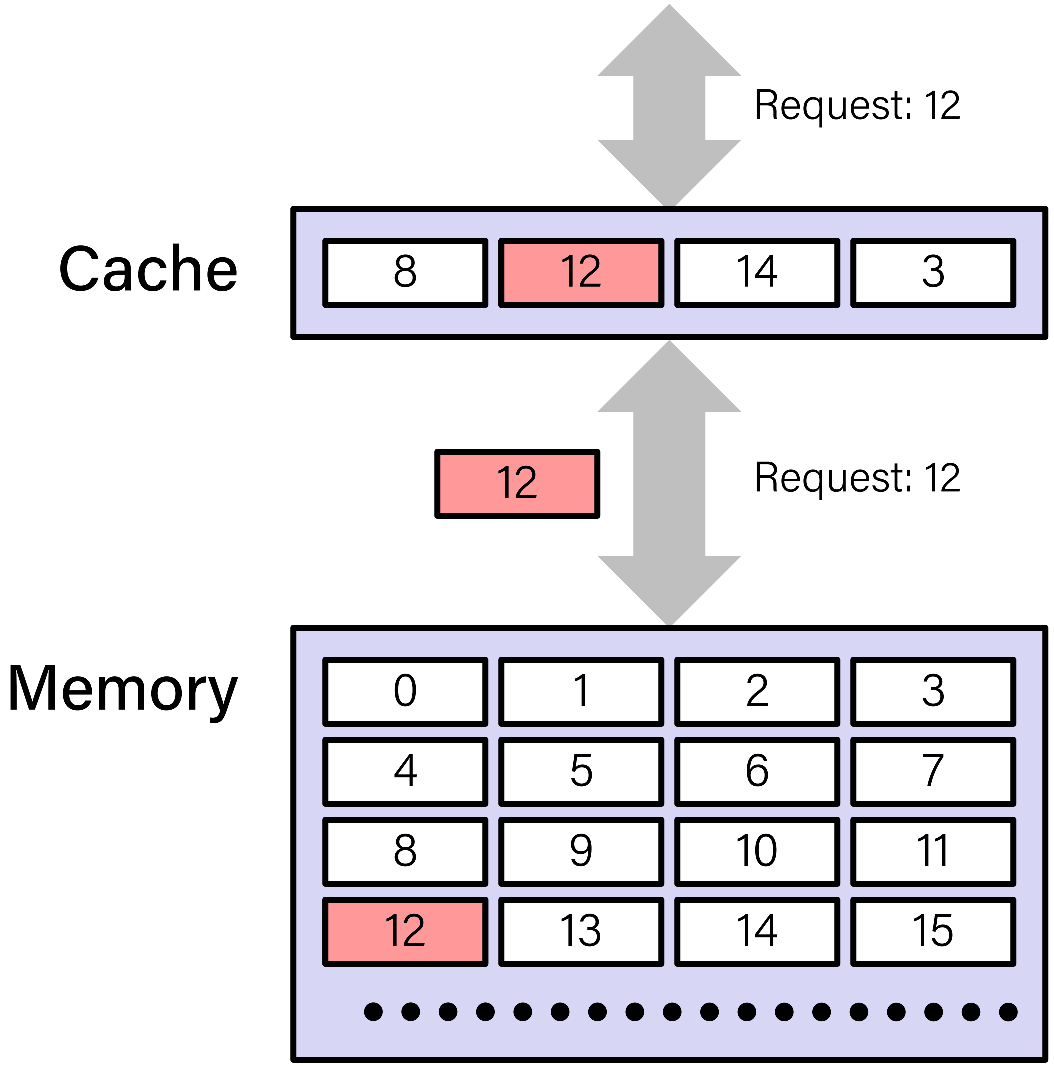
General Cache Concepts: Hit



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)

Examples of Caching in the Mem. 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 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

Recap

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

Next: *Cache memories*