



CS 5541 – Computer Systems

"Based on lecture notes developed by Randal E. Bryant and David R. O'Hallaron in conjunction with their textbook "Computer Systems: A Programmer's Perspective"



Module 3

Caches

Part 1 — The Memory Hierarchy

From: Computer Systems, Chapter 6

Instructor: James Yang
<https://cs.wmich.edu/~zjiang>
zjiang.yang@wmich.edu

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)

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

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 (block-level) 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

What's Inside A Disk Drive?

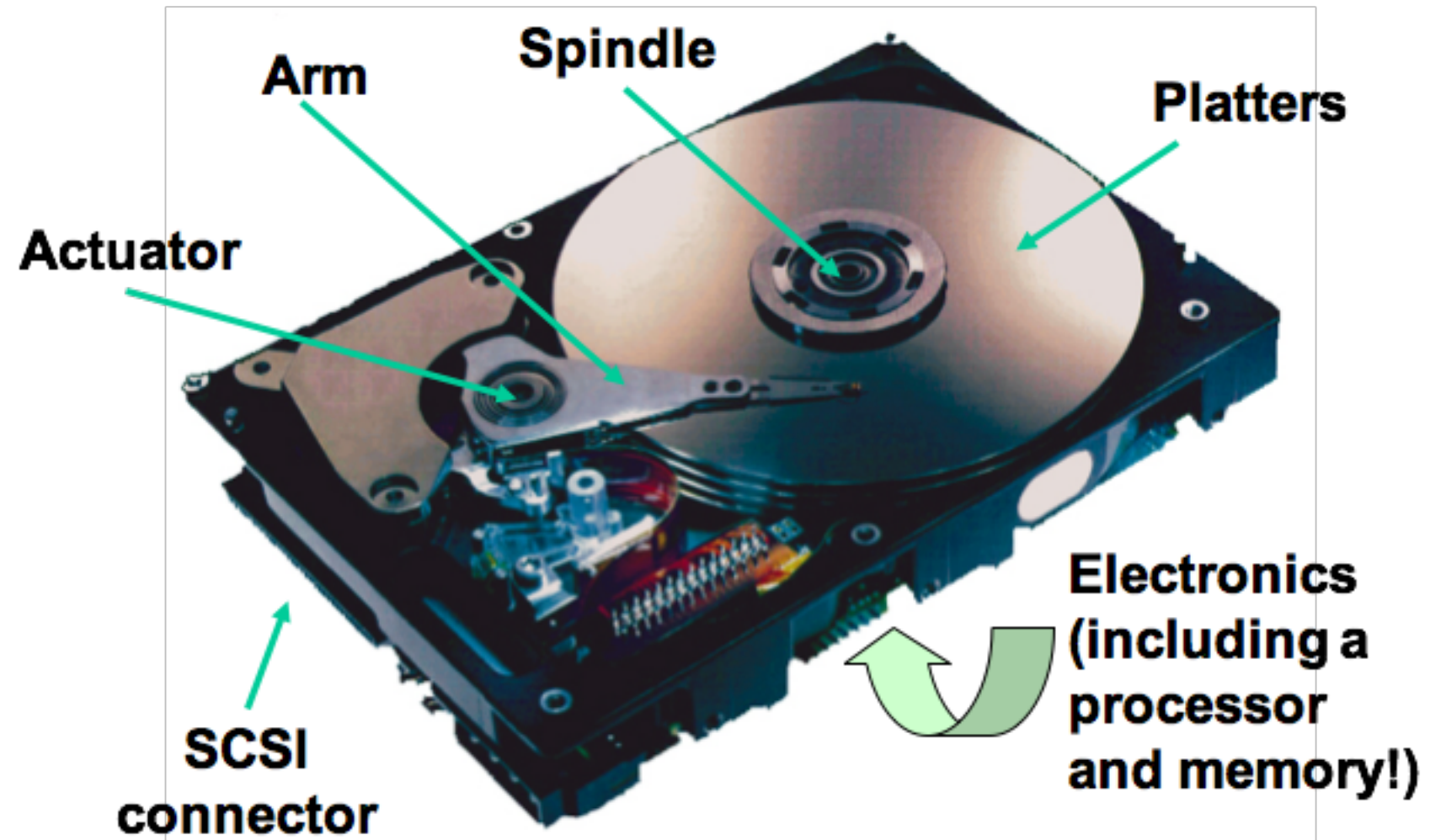
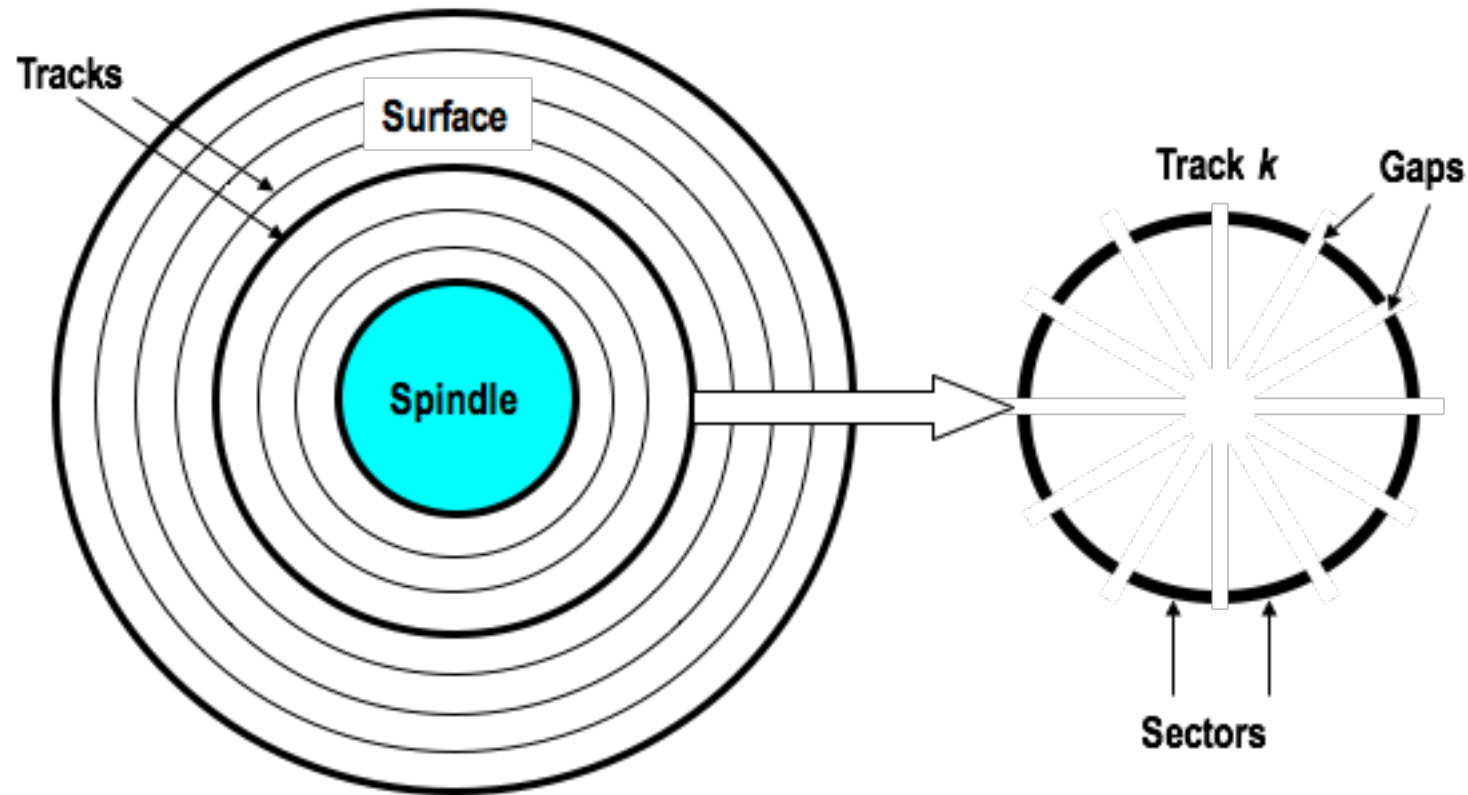


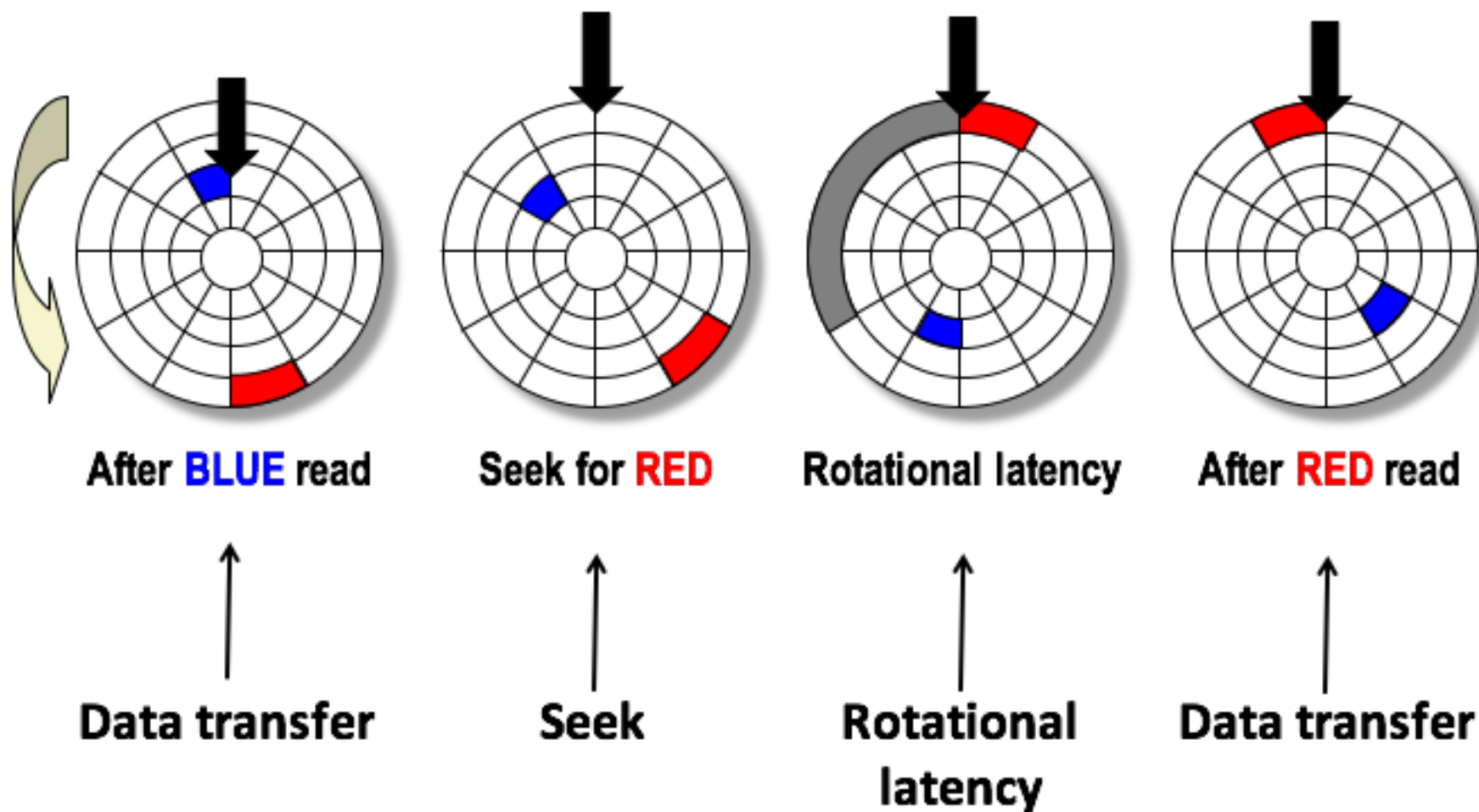
Image courtesy of Seagate Technology

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 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.
 - $T_{\text{avg transfer}} = 1/\text{RPM} \times 1/(\text{avg \# sectors/track}) \times 60 \text{ secs}/1 \text{ min.}$

SSD Tradeoffs vs Rotating Disks

■ Advantages

- No moving parts → faster, less power, more rugged

■ Disadvantages

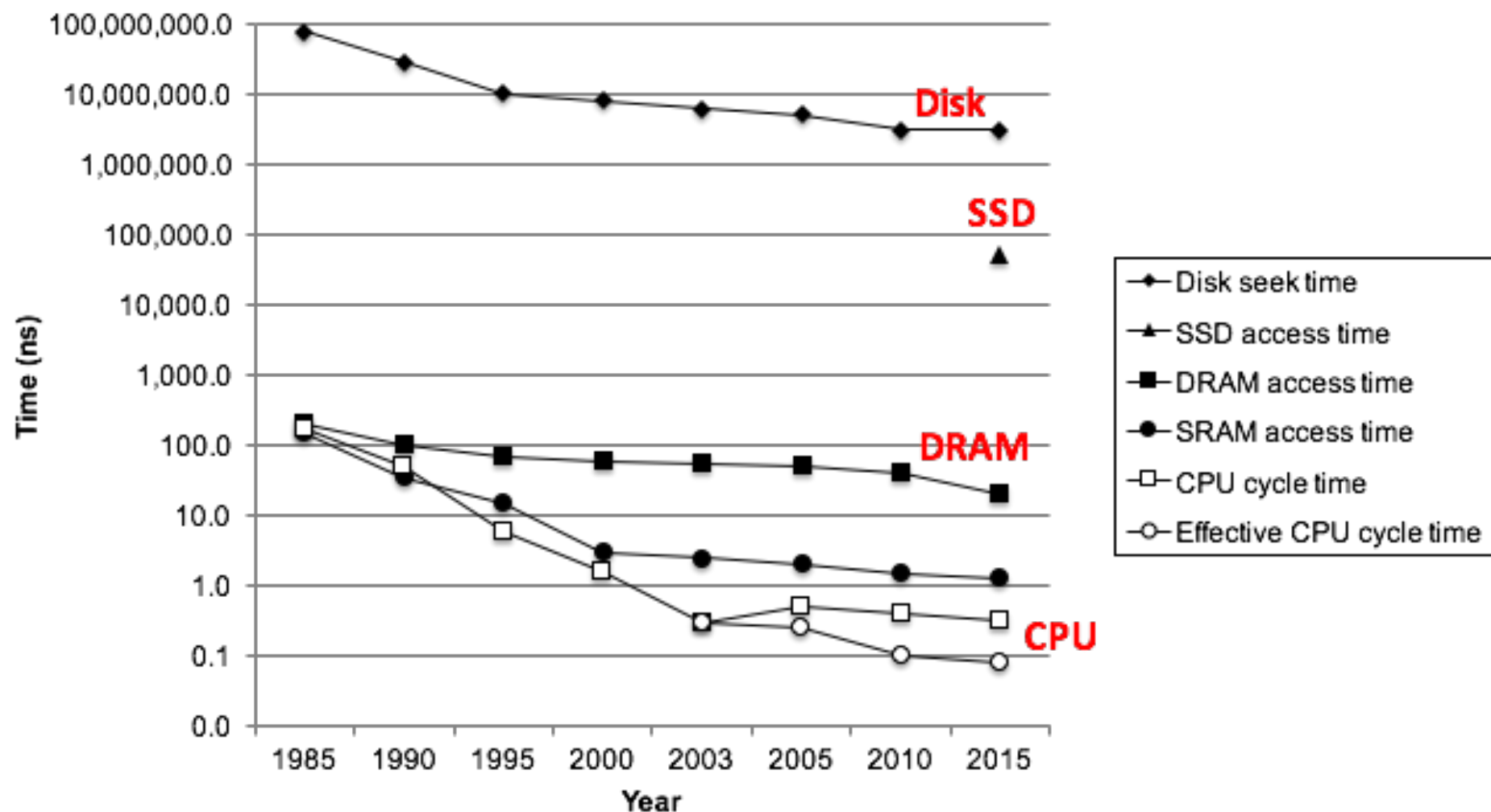
- Have the potential to wear out
 - Mitigated by “wear leveling logic” in flash translation layer
 - E.g. Intel SSD 730 guarantees 128 petabyte (128×10^{15} bytes) of writes before they wear out
- In 2015, about 30 times more expensive per byte

■ Applications

- MP3 players, smart phones, laptops
- Beginning to appear in desktops and servers

The CPU-Memory Gap

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

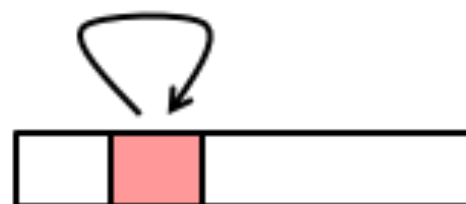


Locality

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

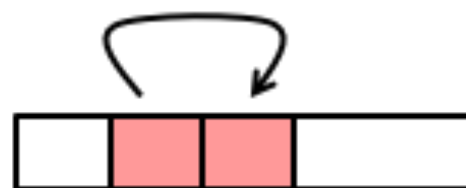
- **Temporal locality:**

- 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).
- Reference variable `sum` each iteration.

Spatial locality

Temporal locality

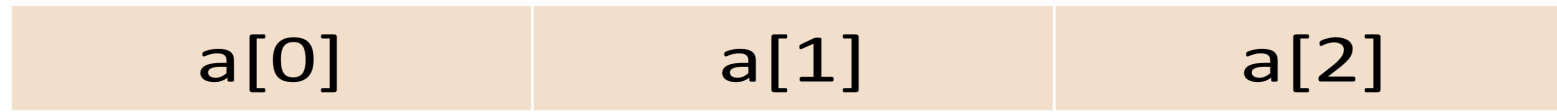
■ Instruction references

- Reference instructions in sequence.
- Cycle through loop repeatedly.

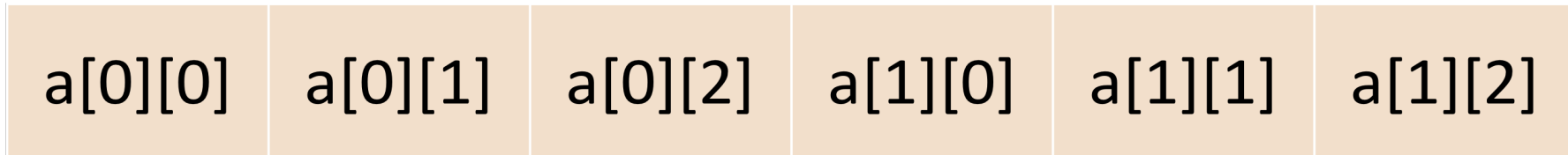
Spatial locality

Temporal locality

int a[3];



int a[3][3];



Stride-1 Reference Pattern: Refers to a function that visits each element of a vector in memory sequentially.

Stride-k Reference Pattern: Refers to a function that visits every k-th element of a vector in memory.

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*?

```
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;

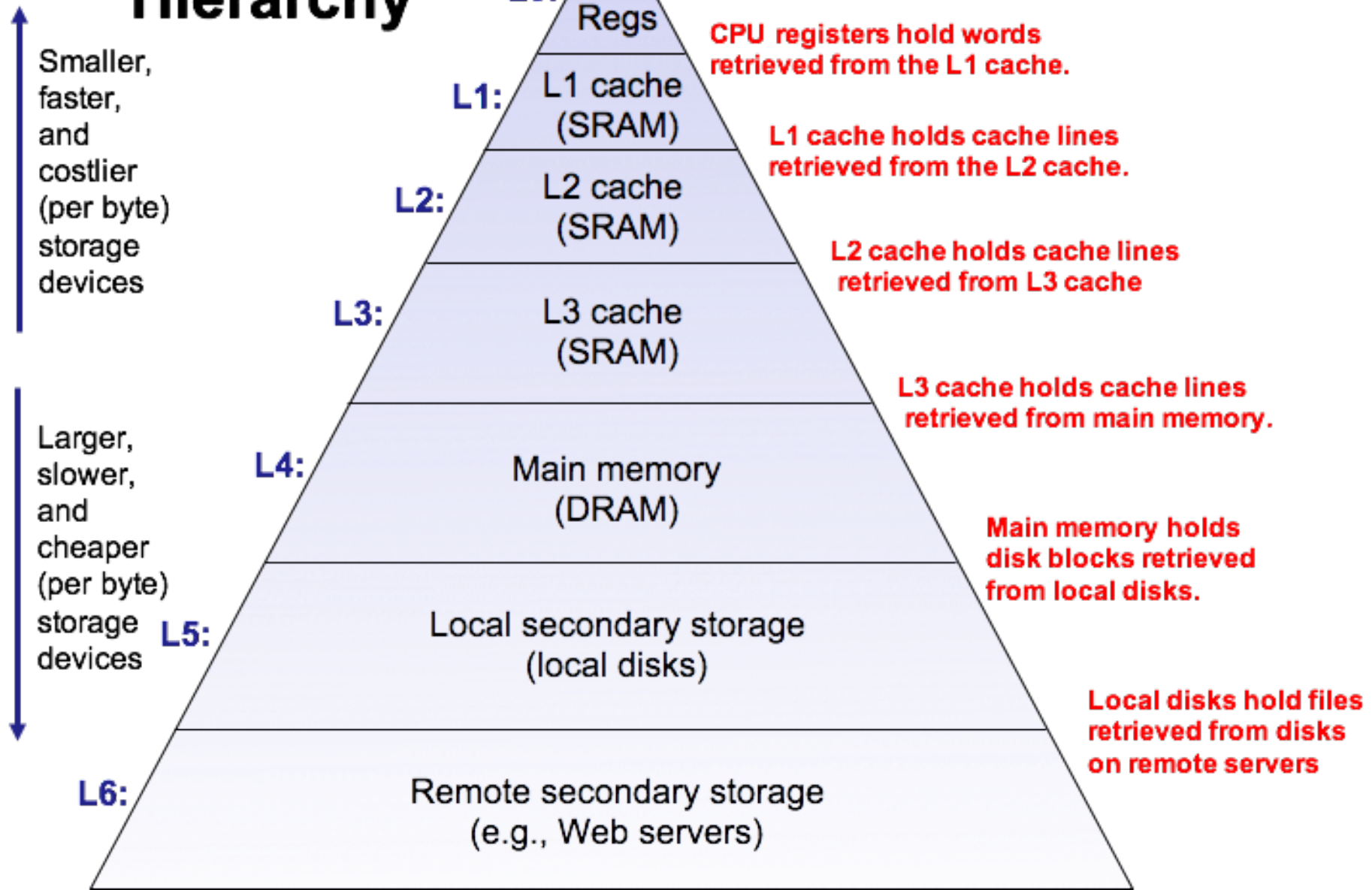
    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];

    return sum;
}
```


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



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.

Module 3 (Part 1)

Summary

- **The speed gap**

- The speed gap between CPU, memory, and mass storage continues to widen

- **Locality**

- Well-written programs exhibit a property called locality
- Memory hierarchies based on caching close the gap by exploiting locality

