

The Memory Hierarchy

Kai Shen

1

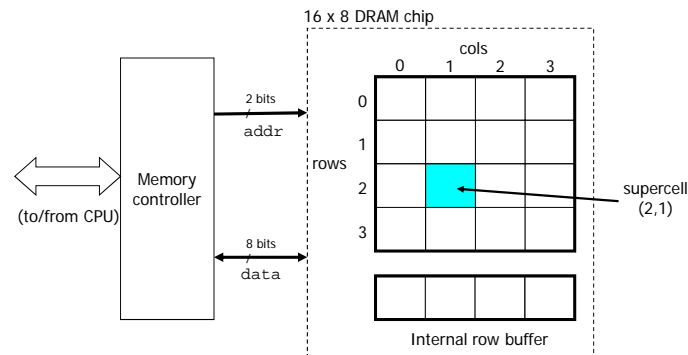
Memory

- Random-access memory (RAM)
 - RAM is traditionally packaged as a chip.
 - Basic storage unit is normally a **cell** (one bit per cell).
 - Multiple RAM chips form a memory.
- Static RAM (SRAM)
 - Each cell stores a bit with a four or six-transistor circuit.
 - Retains value indefinitely, as long as it is kept powered.
 - Relatively insensitive to electrical noise, radiation, etc.
- Dynamic RAM (DRAM)
 - Each cell stores bit with a capacitor. One transistor is used for access.
 - Value must be refreshed every 10-100 ms.
 - More sensitive to disturbances (electrical noise, radiation,...) than SRAM.
 - Slower and cheaper than SRAM.

2

Conventional DRAM Organization

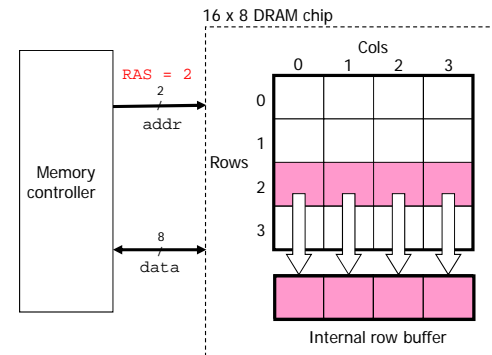
- $d \times w$ DRAM:
 - dw total bits organized as d **supercells** of size w bits



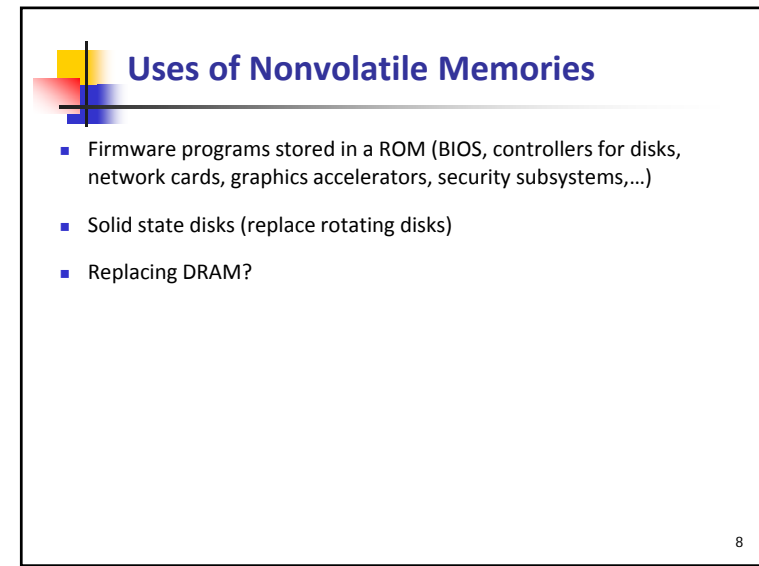
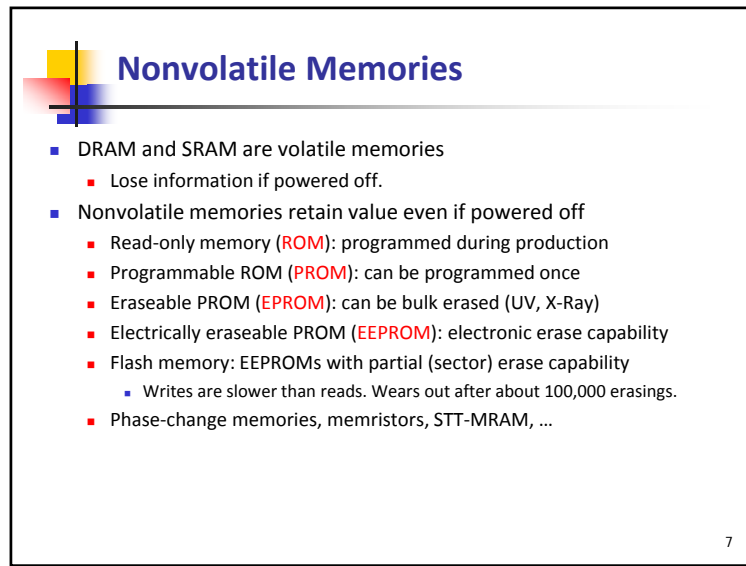
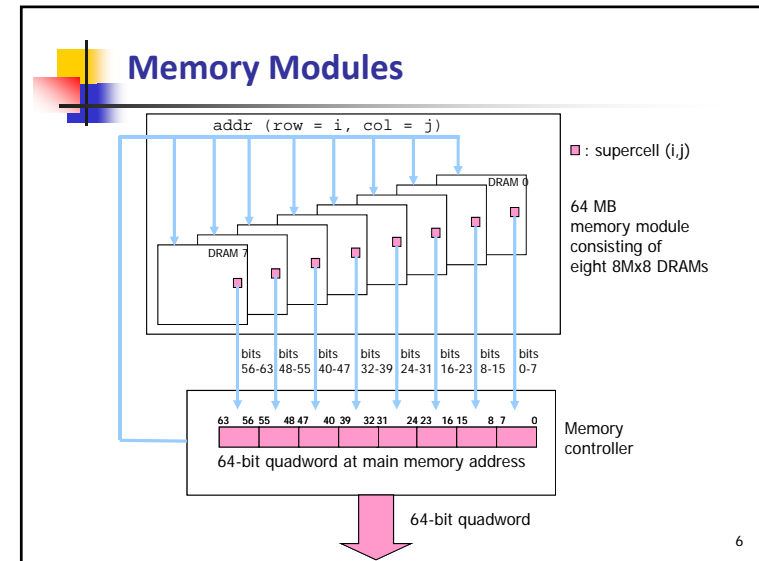
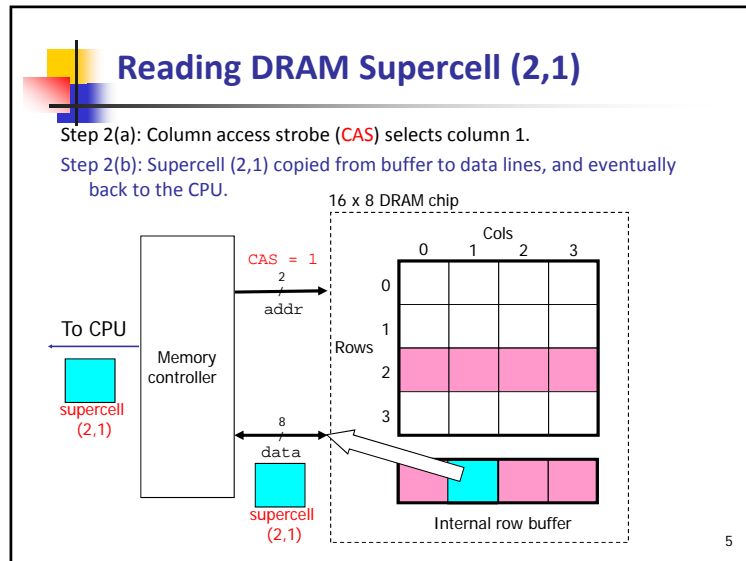
3

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.

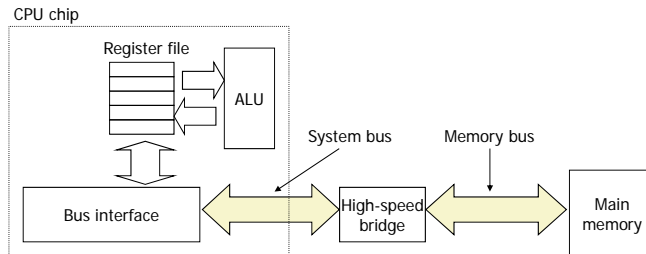


4



Traditional Bus Structure Connecting CPU and Memory

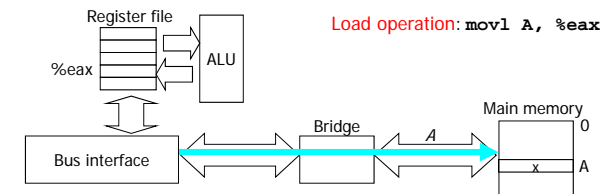
- A **bus** is a collection of parallel wires that carry address, data, and control signals.
- Buses are typically shared by multiple devices.



9

Memory Read Transaction (1)

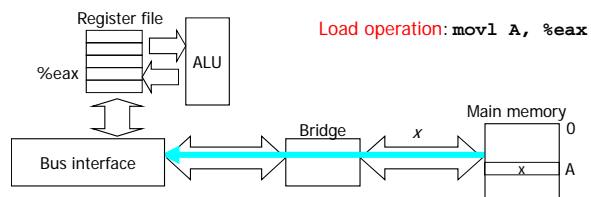
- CPU places address A on the memory bus.



10

Memory Read Transaction (2)

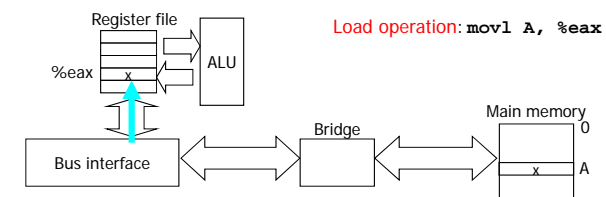
- Main memory reads A from the memory bus, retrieves word x, and places it on the bus.



11

Memory Read Transaction (3)

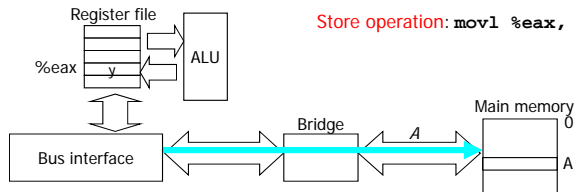
- CPU read word x from the bus and copies it into register %eax.



12

Memory Write Transaction (1)

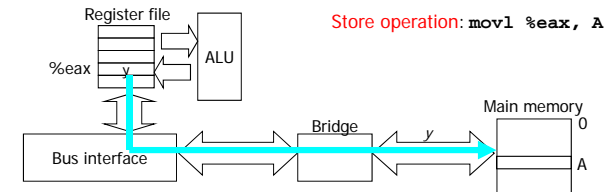
- CPU places address A on bus. Main memory reads it and waits for the corresponding data word to arrive.



13

Memory Write Transaction (2)

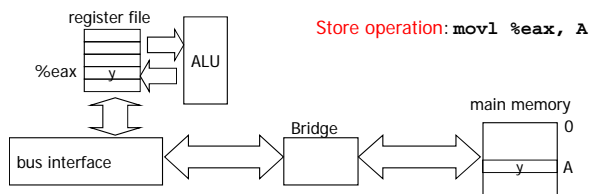
- CPU places data word y on the bus.



14

Memory Write Transaction (3)

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



15

What's Inside A Disk Drive?

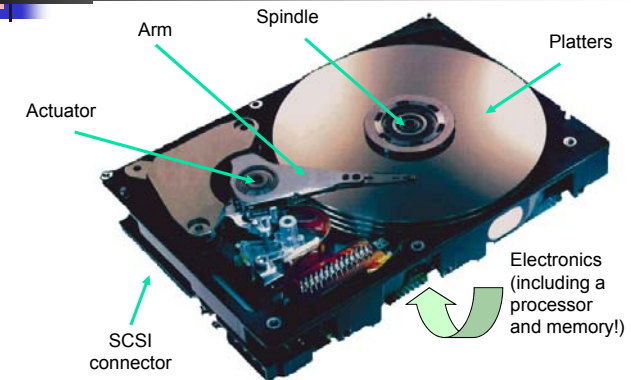
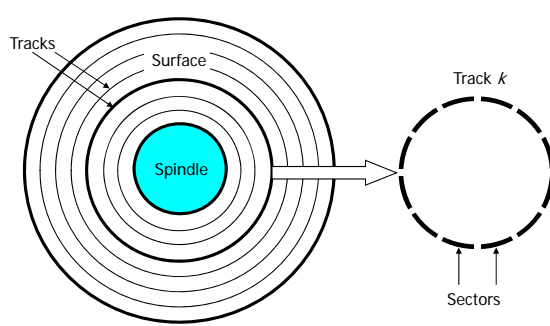


Image courtesy of Seagate Technology

16

Disk Geometry

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



The diagram illustrates the geometry of a disk surface. It shows a central **Spindle** with concentric circles representing **Tracks** on a **Surface**. An arrow points from one track to a detailed view of **Track k**, which is divided into **Sectors**.

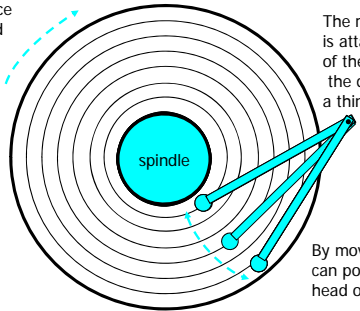
17

Disk Operation

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.

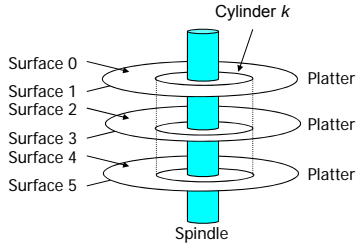


The diagram shows a disk with a central **spindle** and concentric tracks. A **read/write head** is shown at the end of an **arm**, positioned over a track. A dashed line indicates the radial movement of the arm.

18

Disk Geometry (Multiple-Platter View)

- Aligned tracks form a cylinder.

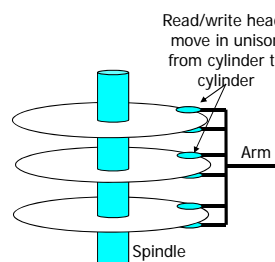


The diagram shows a vertical stack of three platters: **Platter 0**, **Platter 1**, and **Platter 2**, all mounted on a common **Spindle**. The surfaces are labeled **Surface 0** through **Surface 5**. A vertical cylinder, labeled **Cylinder k**, passes through the center of each platter, representing the alignment of tracks across different surfaces.

19

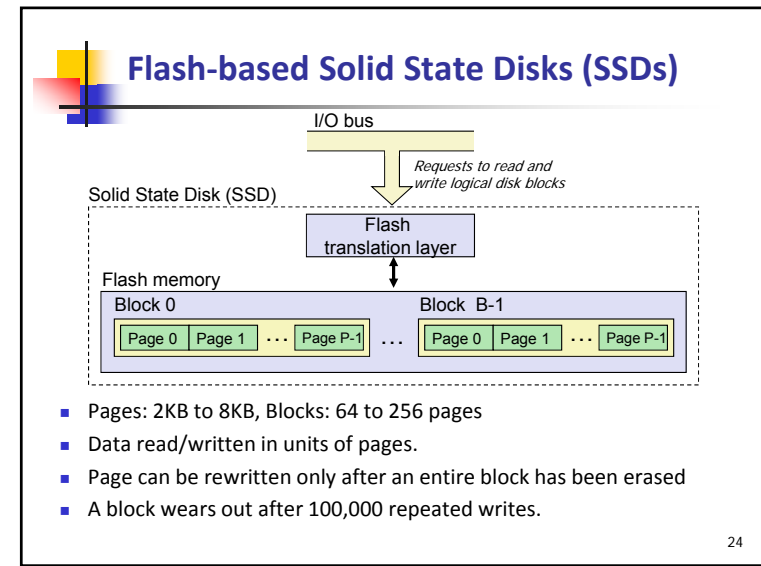
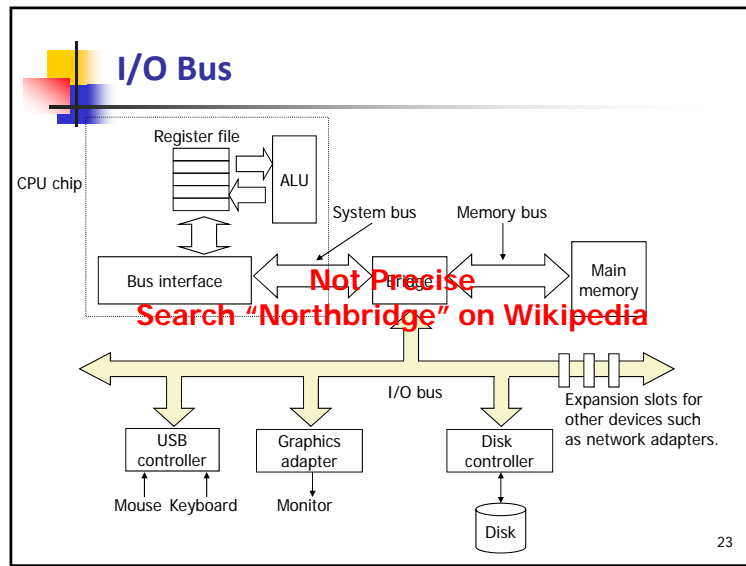
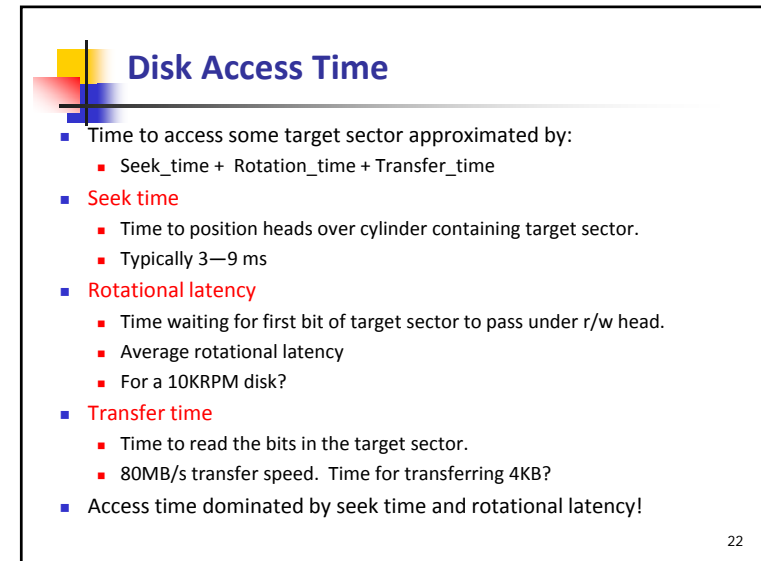
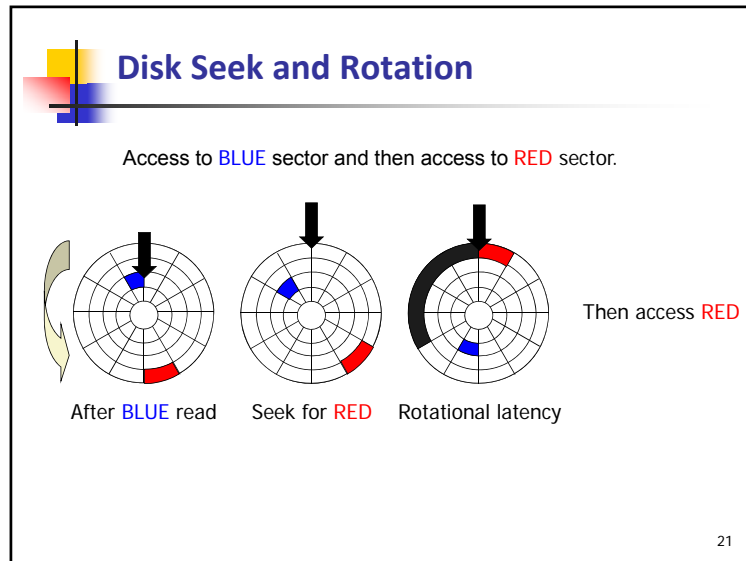
Disk Operation (Multi-Platter View)

Read/write heads move in unison from cylinder to cylinder.



The diagram shows the same stack of platters as in slide 19. It illustrates the **Arm** holding multiple **Read/write heads** that move in unison across the different surfaces of the platters, accessing data from a specific **cylinder** (a set of aligned tracks).

20



SSD Performance Characteristics

- Performance characteristics
 - Sequential reads/writes comparable (a bit faster) than disks
 - Random reads are much faster than disks (0.1ms vs. 10ms)
 - Random writes are somewhat slower than reads (still much faster than disks)
- Why are random writes slow?
 - Erasing a block is slow (around 1 ms) and it forces a copy of all useful pages in the block

25

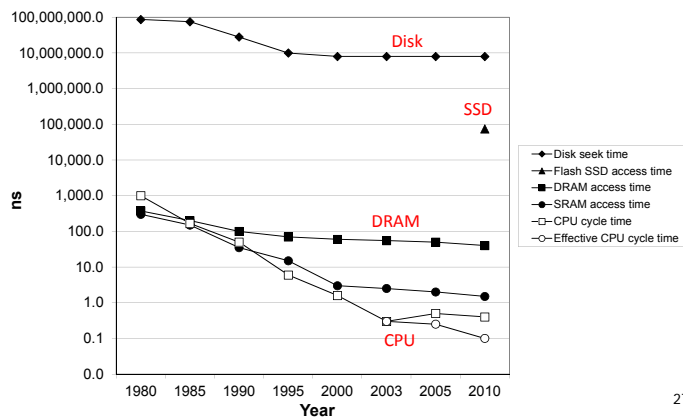
SSD Tradeoffs vs Rotating Disks

- 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
 - 10 (or more) times more expensive per byte
- Applications
 - MP3 players, smart phones, laptops
 - Start to appear in desktops and servers

26

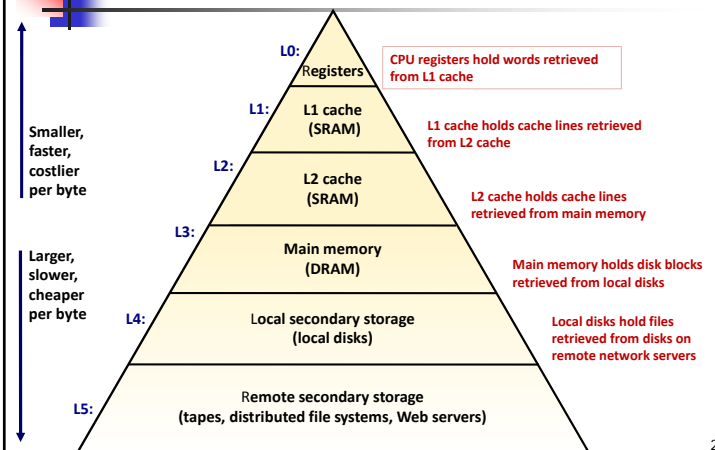
The CPU-Memory Gap

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



27

Memory Hierarchy



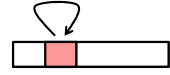
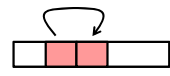
28

Memory Hierarchy

- 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 memory/storage speed is widening.
 - Must pick between space or speed?
- Locality to the rescue
 - Well-written programs tend to exhibit good locality.

29

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

30

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**

31

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;
}
```

32



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;
}
```

33



Disclaimer

These slides were adapted from the CMU course slides provided along with the textbook of "Computer Systems: A programmer's Perspective" by Bryant and O'Hallaron. The slides are intended for the sole purpose of teaching the computer organization course at the University of Rochester.

34