Memory Hierarchy

Computer Systems Organization (Spring 2017)
CSCI-UA 201, Section 3

Instructor: Joanna Klukowska

Slides adapted from
Randal E. Bryant and David R. O'Hallaron (CMU)
Mohamed Zahran (NYU)

Storage: Memory and Disk (and other I/O Devices)

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)

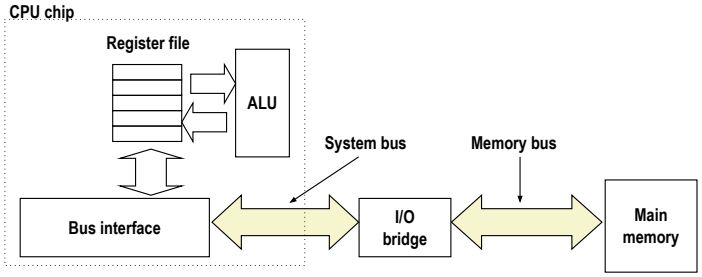
	Trans. per bit	Access time	Needs refresh?	Needs EDC?	Cost	Applications
SRAM	4 or 6	1X	No	Maybe	100x	Cache memories
DRAM	1	10X	Yes	Yes	1X	Main memories, frame buffers

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
  - **Eraseable PROM (EPROM)**: can be bulk erased (UV, X-Ray)
  - **Electrically eraseable 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

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.

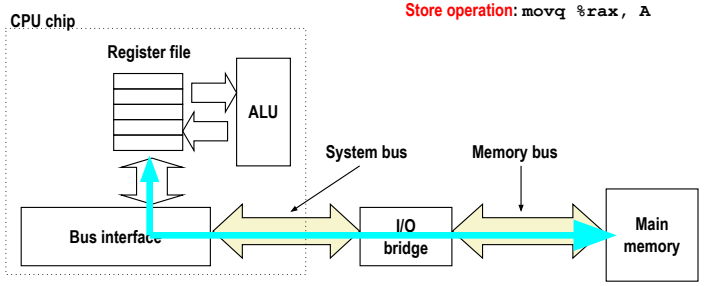


Bus Structure Connecting CPU and Memory

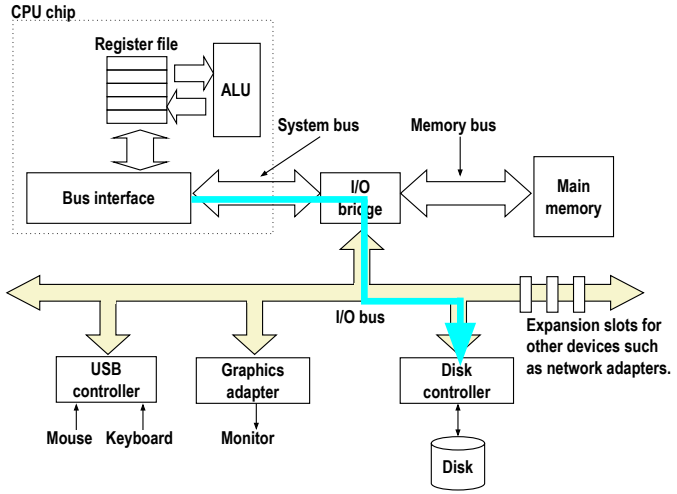
- A bus is a collection of parallel wires that carry address, data, and control signals.
- Buses are typically shared by multiple devices.

Load operation: `movq A, %rax`

Store operation: `movq %rax, A`



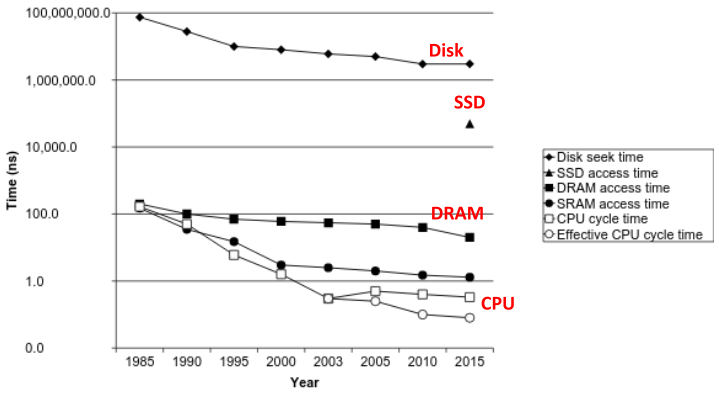
I/O Bus



Slow Storage ⟺ Fast Processors
Locality to the rescue

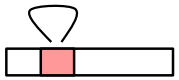
The CPU-Memory Gap

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

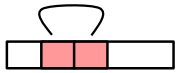


Locality

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



Qualitative Estimates of Locality

- **Being able to look at code and get a qualitative sense of its locality is a key skill for a professional programmer.**

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;
}
```

✔

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;
}
```

✘

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 locality

Temporal locality

Spatial locality

Temporal locality

Locality DNHI

➢ DNHI: 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)?

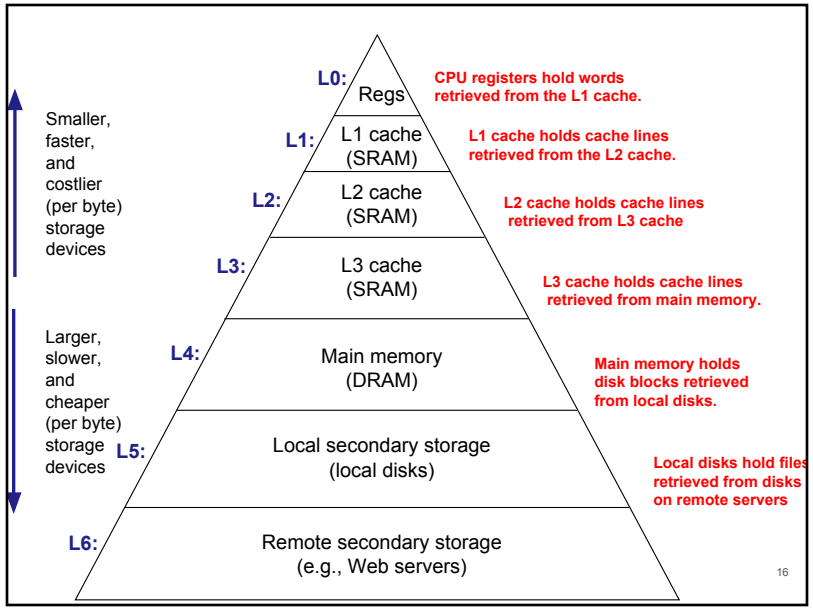
```
int sum_array_3d(int a[M][N][N])
{
    int i, j, k, sum = 0;

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


Memory Hierarchy
(otherwise locality does not help)

Memory Hierarchies

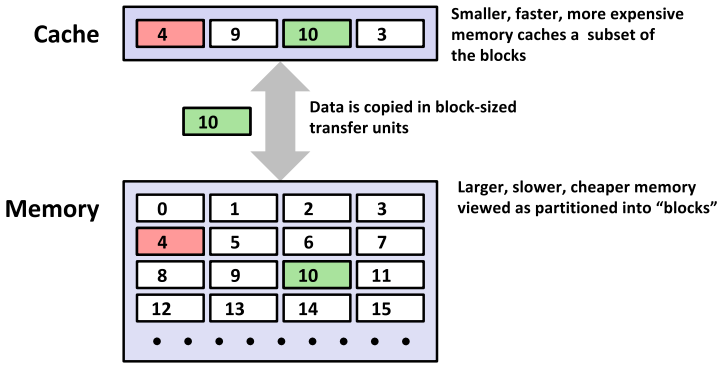
- Some fundamental and enduring properties of hardware and software:
  - Fast storage technologies cost more per byte, have less capacity, and require more power (heat!).
  - The gap between CPU and main memory speed is widening.
  - Well-written programs tend to exhibit good locality.
- These fundamental properties complement each other beautifully.
- They suggest an approach for organizing memory and storage systems known as a memory hierarchy.



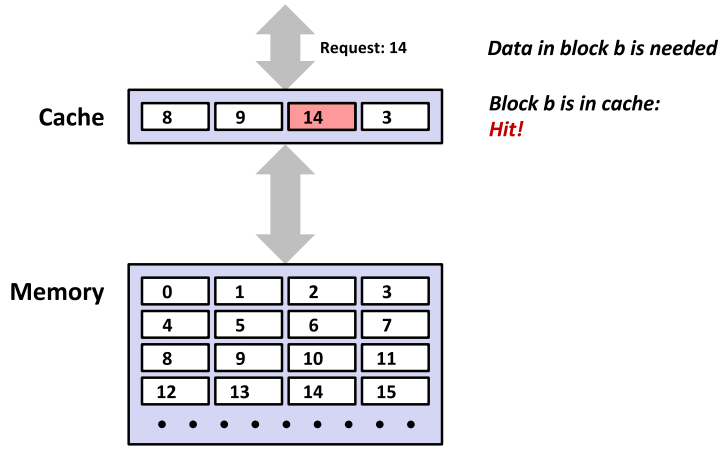
Caches

- **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.

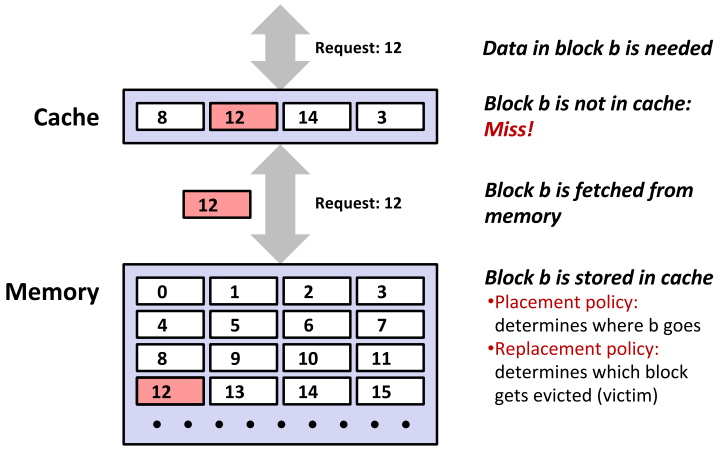
General Cache Concepts



General Cache Concepts: Hit



General Cache Concepts: Miss



General Caching: Types of Cache Misses

- **Cold (compulsory) miss**
  - Cold misses occur because the cache is empty.
- **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.
- **Capacity miss**
  - Occurs when the set of active cache blocks (working set) is larger than the cache.