CSE 153
Design of Operating Systems

Fall 2018

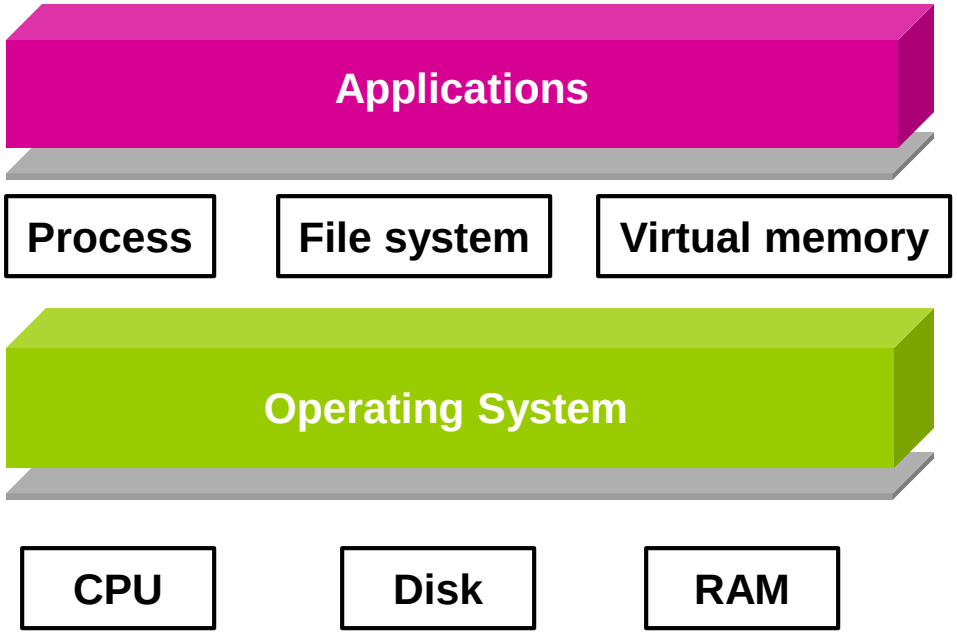
Lecture 08: Memory Management (1)

Some slides from Dave O'Hallaron

Announcements

- Midterm key posted
  - Hope to finish grading over the weekend
- HW2 (Ali) and HW1 (Hadi) should be graded by now
- Lab 2 due Friday
  - hope you are winning the battle against xv6 scheduler
  - Lab 3 will be released, but you will have to read ahead
    - » I may have to push back due date…we'll see
- Optional dynamic memory lab. to be assigned soon

OS Abstractions



Our plan of action

- Memory/storage technologies and trends
  - Memory wall!
- Locality of reference to the rescue
  - Caching in the memory hierarchy
- Abstraction: Address spaces and memory sharing
- Virtual memory
- Today: background and bird's eye view – more details to follow later

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.
- 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 (EMI), radiation, etc.
  - Faster and more expensive than DRAM.
- 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 (EMI, radiation,…) than SRAM.
  - Slower and cheaper than SRAM.

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 – lose info without power

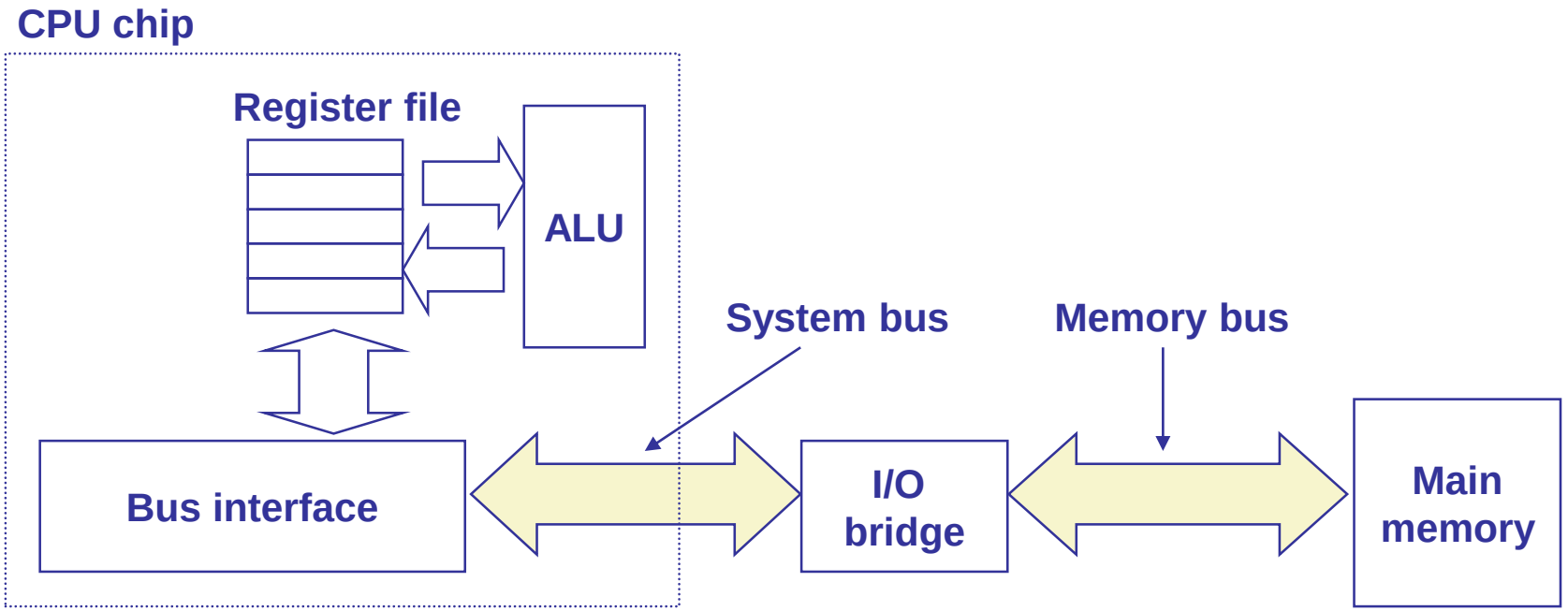
- Nonvolatile memories (NVMs) retain value
  - Read-only memory (ROM): programmed during production
  - Programmable ROM (PROM): can be programmed once
  - Eraseable PROM (EPROM): can be bulk erased (UV, X-Ray)
  - Electrically eraseable PROM (EEPROM): electronic erase
  - Flash memory: EEPROMs with partial (sector) erase capability
    - » Wears out after about 100,000 erasings.
  - Phase Change Memories (PCMs): also wear out
  - Many exciting NVMs at various stages of development

NVM Uses

- 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,…)
- Caches in high end systems
- Getting better -- many expect Universal memory to come
  - i.e., large replace both DRAM and disk drives

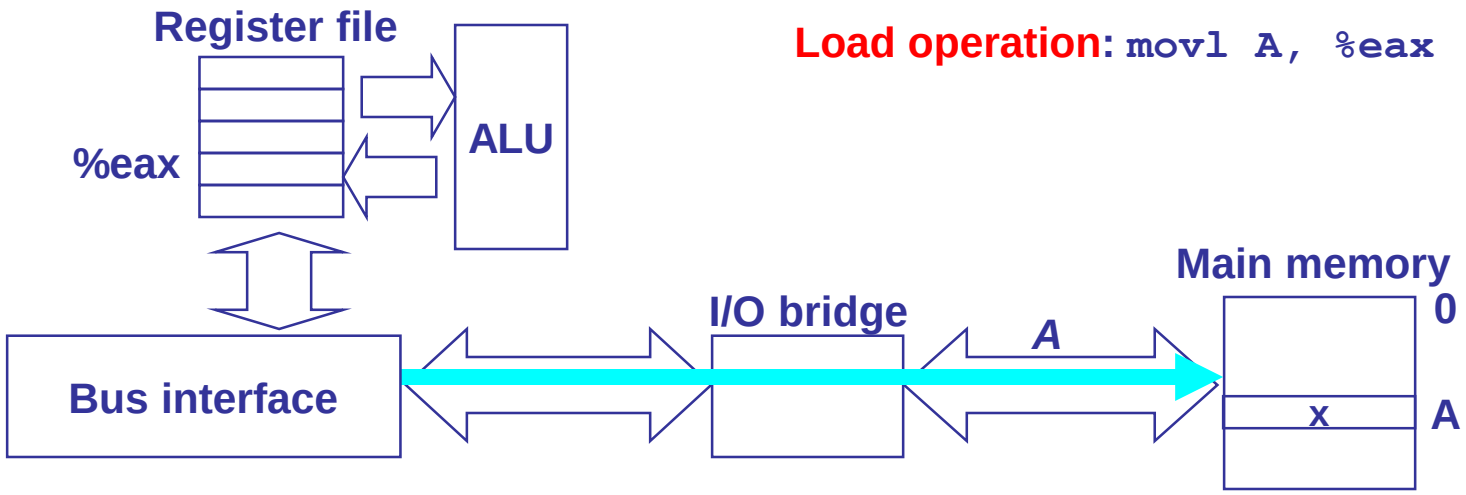
Traditional Bus Structure Connecting CPU and Memory

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



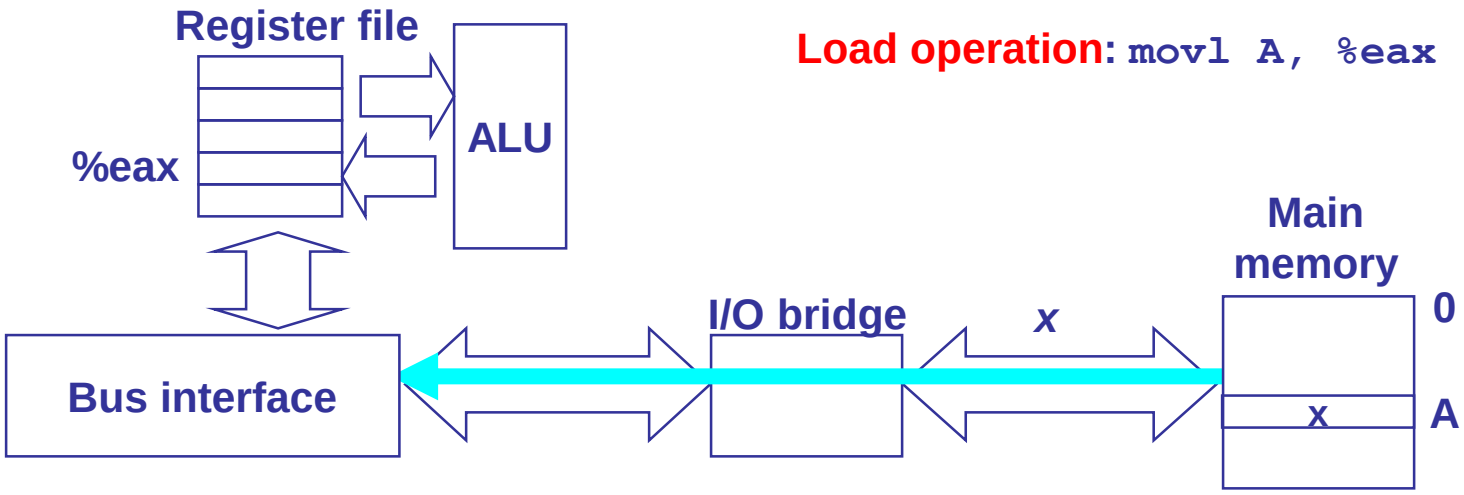
Memory Read Transaction (1)

- CPU places address A on the memory bus.



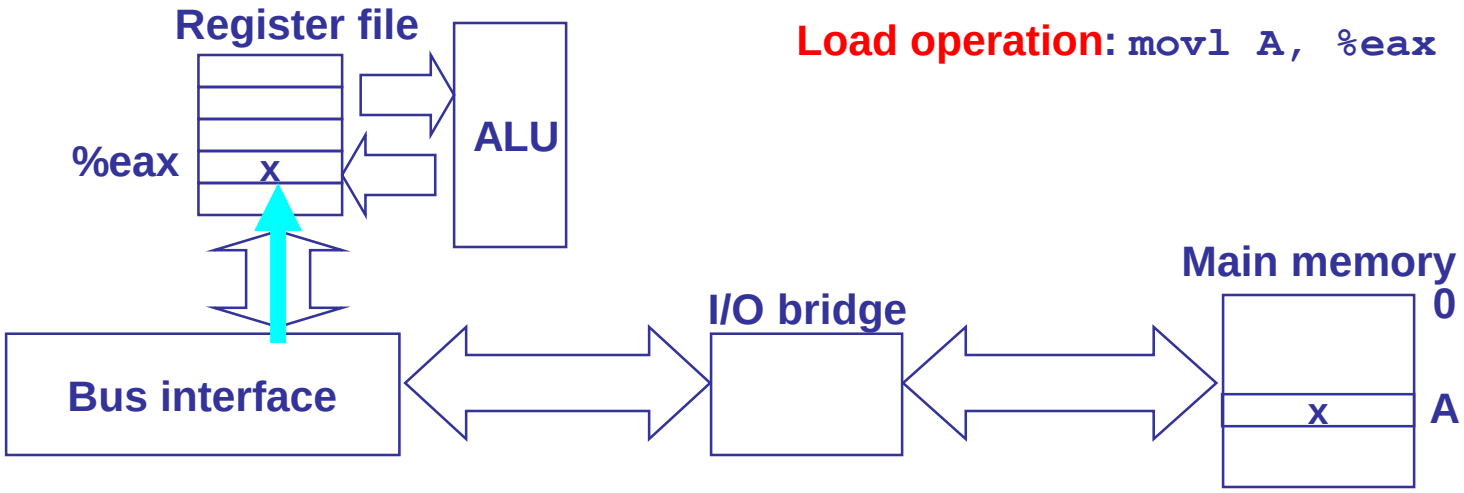
Memory Read Transaction (2)

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



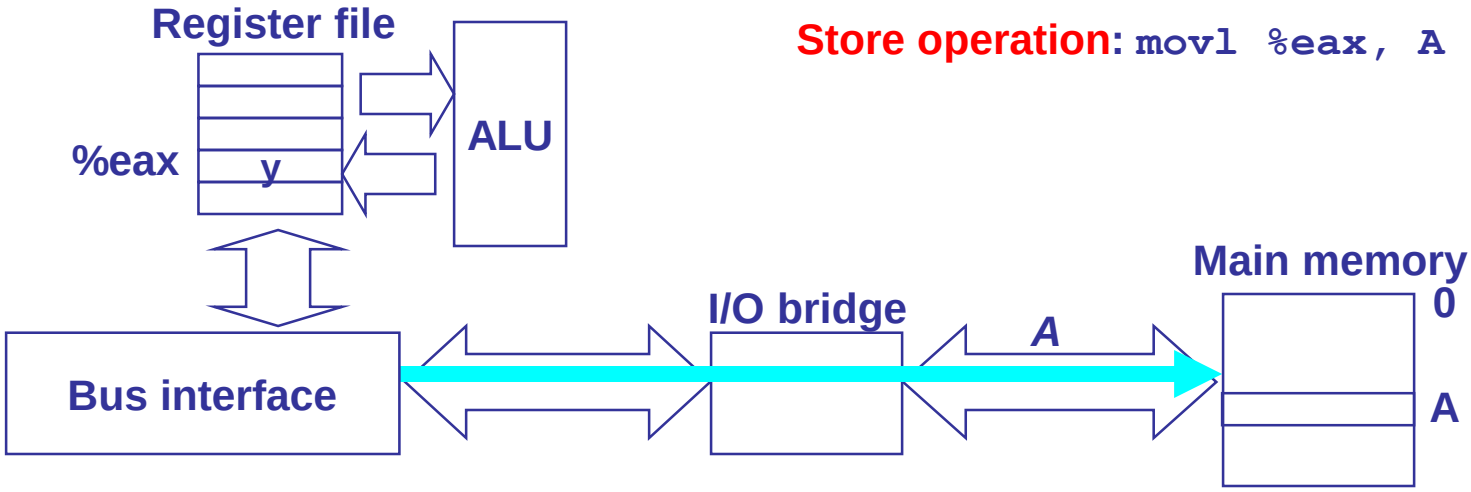
Memory Read Transaction (3)

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



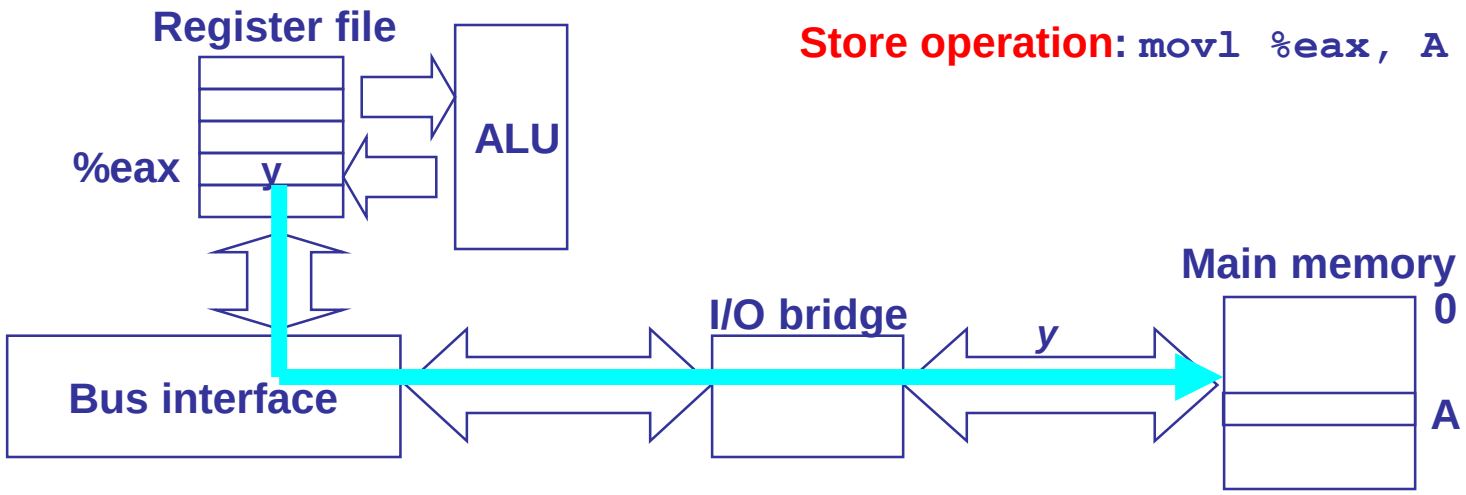
Memory Write Transaction (1)

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



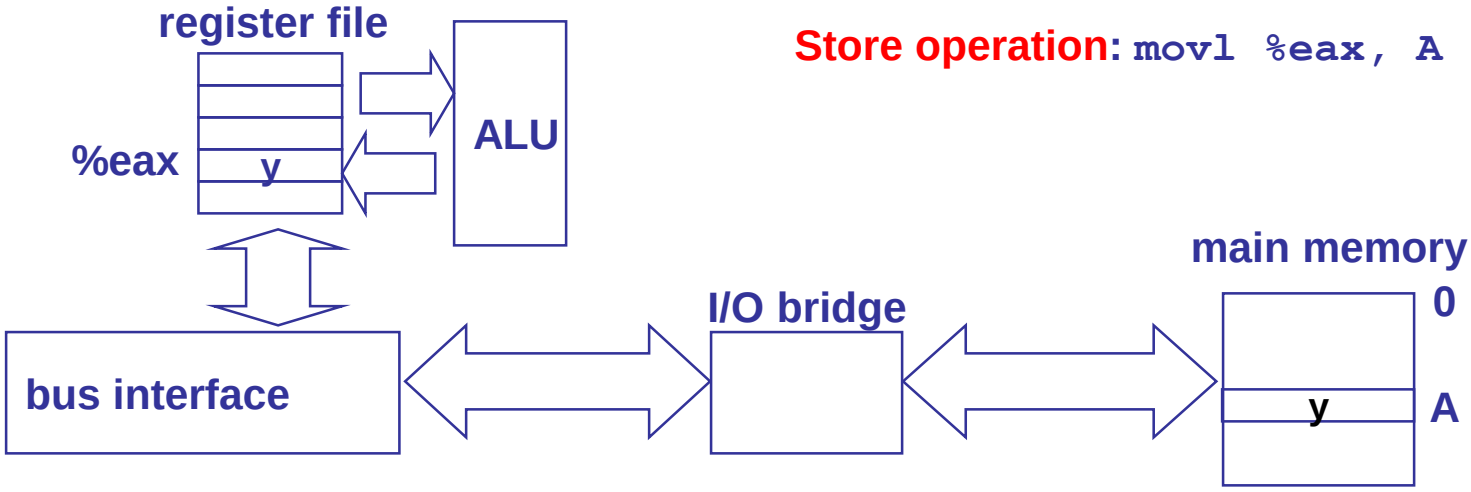
Memory Write Transaction (2)

- CPU places data word y on the bus.



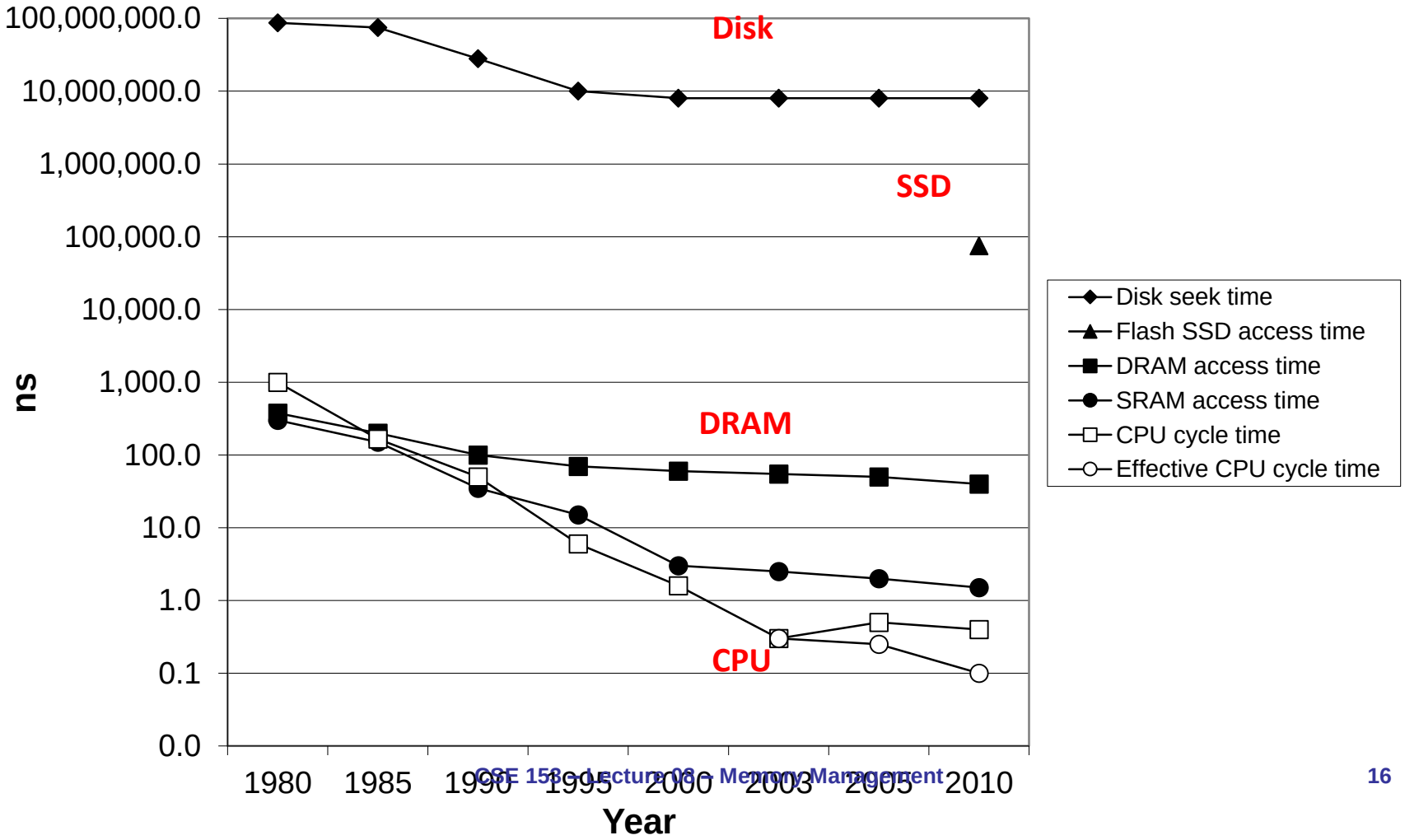
Memory Write Transaction (3)

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



The CPU-Memory Gap

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



Locality to the Rescue!

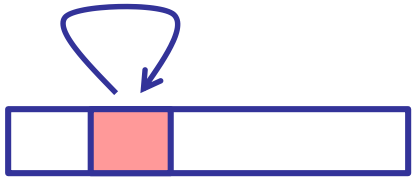
The key to bridging this CPU-Memory gap is a fundamental property of computer programs known as locality

Today

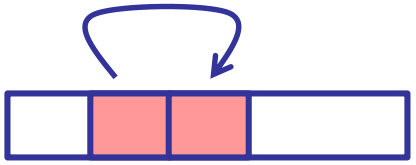
- Storage technologies and trends
- Locality of reference
- Caching in the memory hierarchy
- Virtual memory and memory sharing

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 — **Spatial locality**
  - Reference instructions in sequence. — **Temporal locality**
  - Cycle through loop repeatedly.

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;
}
```

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

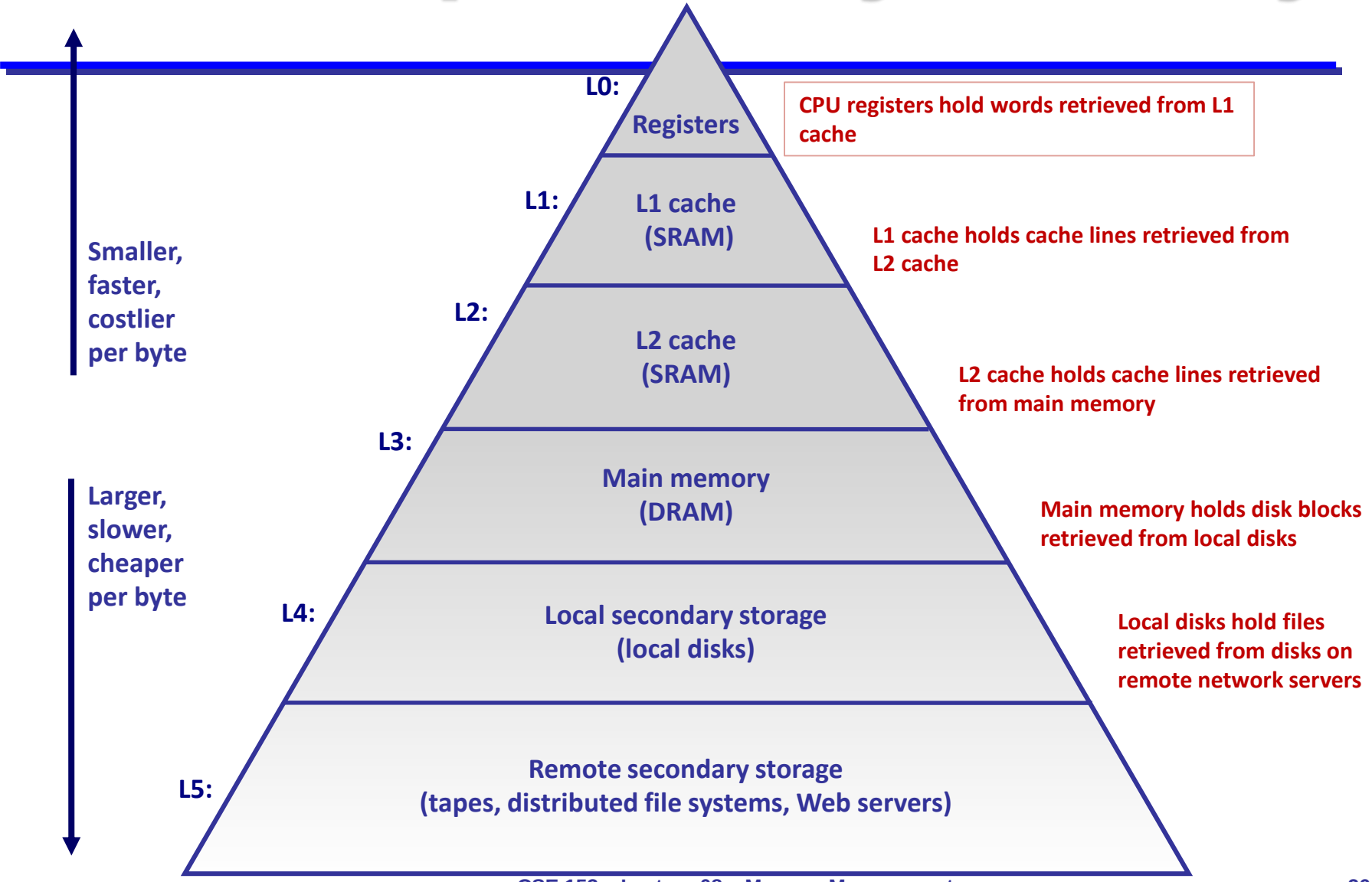
Memory Hierarchies

- 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.

Today

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

An Example Memory Hierarchy



Memory hierarchy

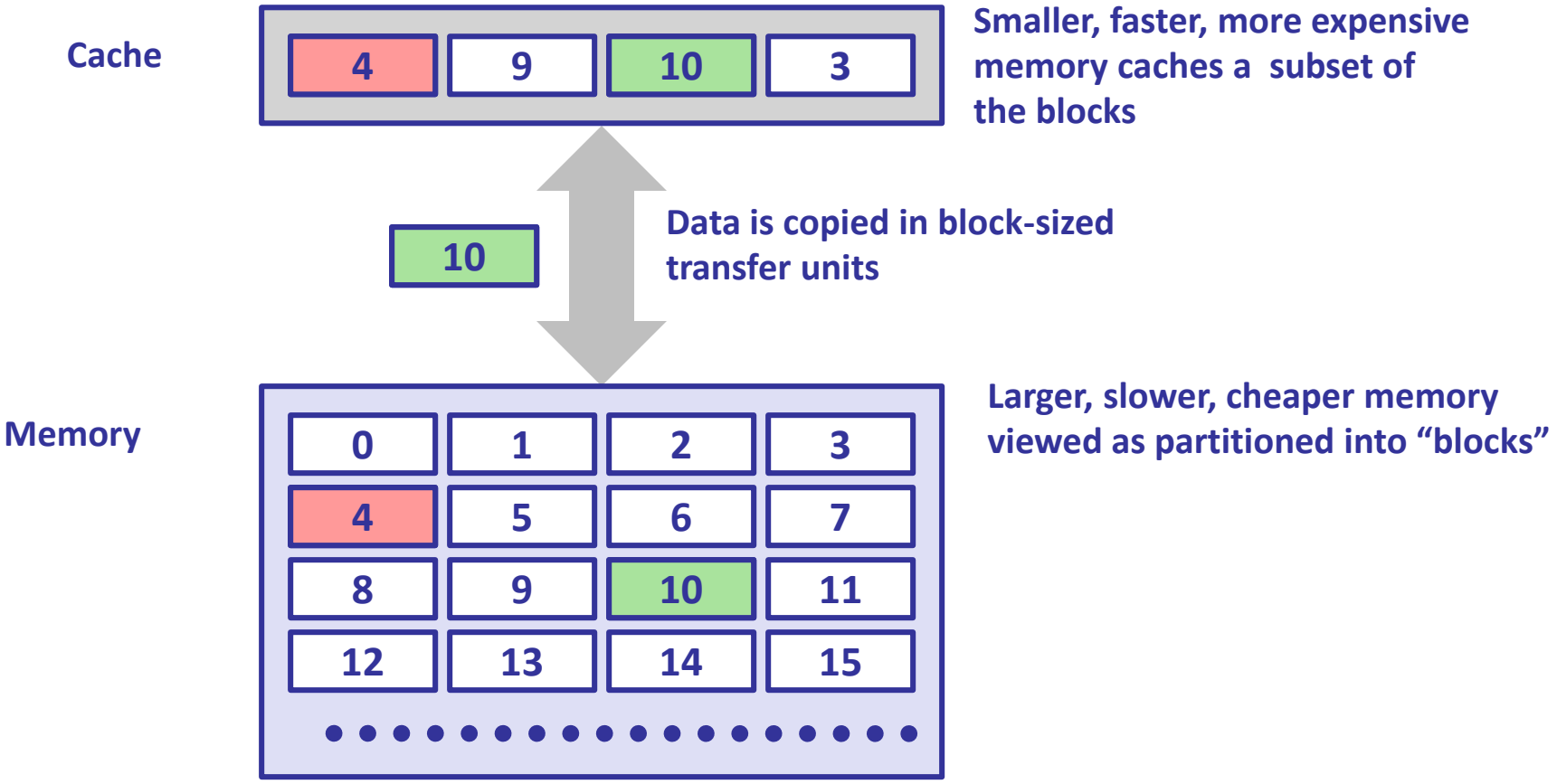
- *Cache:* A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.

- Fundamental idea of a memory hierarchy:
  - For each layer, faster, smaller device caches larger, slower device

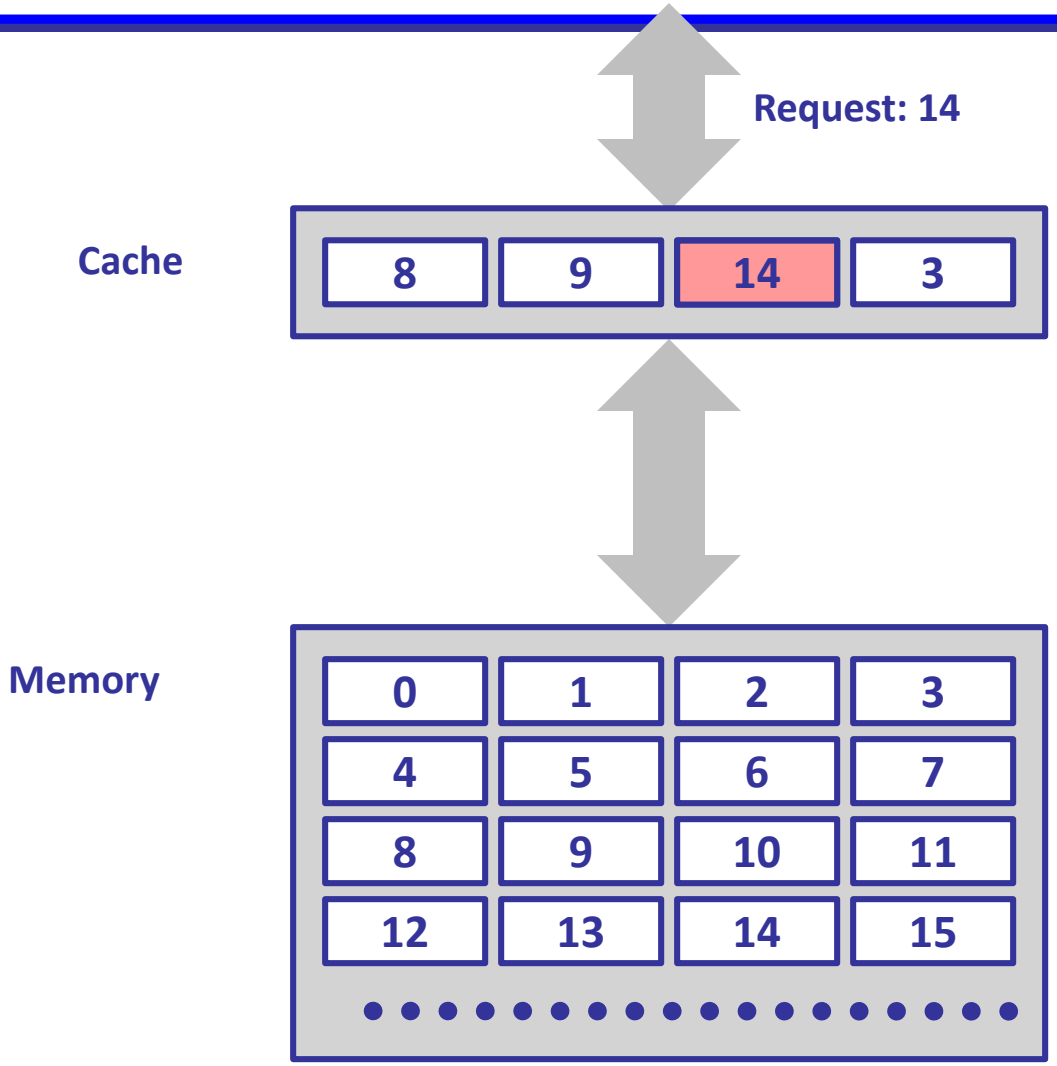
  .
- Why do memory hierarchies work?
  - Because of locality!
    - » Hit fast memory much more frequently even though its smaller
  - Thus, the storage at level k+1 can be slower (but larger and cheaper!)

- *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



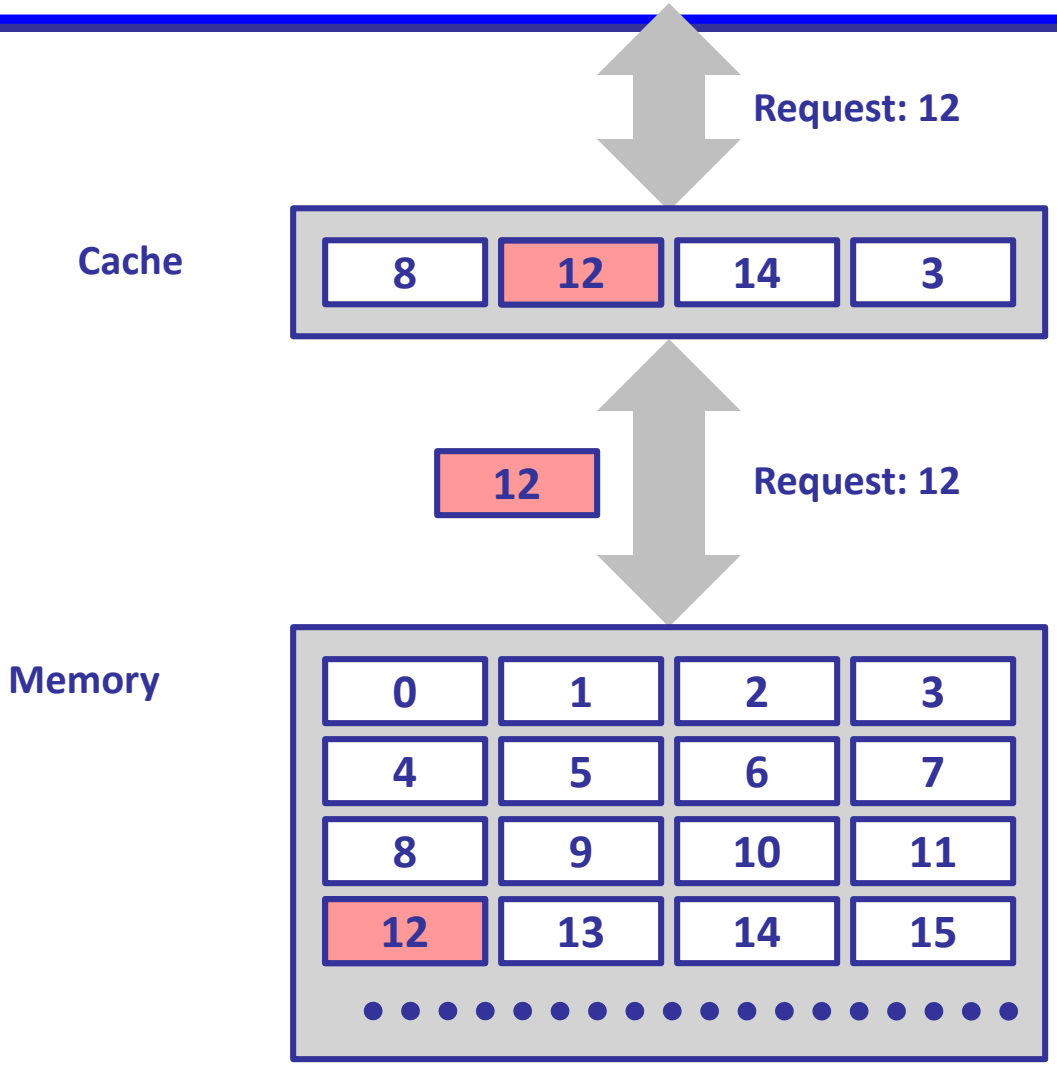
General Cache Concepts: Hit



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)

General Caching Concepts: 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.

Examples of Caching in the 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
L1 cache	64-bytes block	On-Chip L1	1	Hardware
L2 cache	64-bytes block	On/Off-Chip L2	10	Hardware
Virtual Memory	4-KB page	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	AFS/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 so far

- 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.