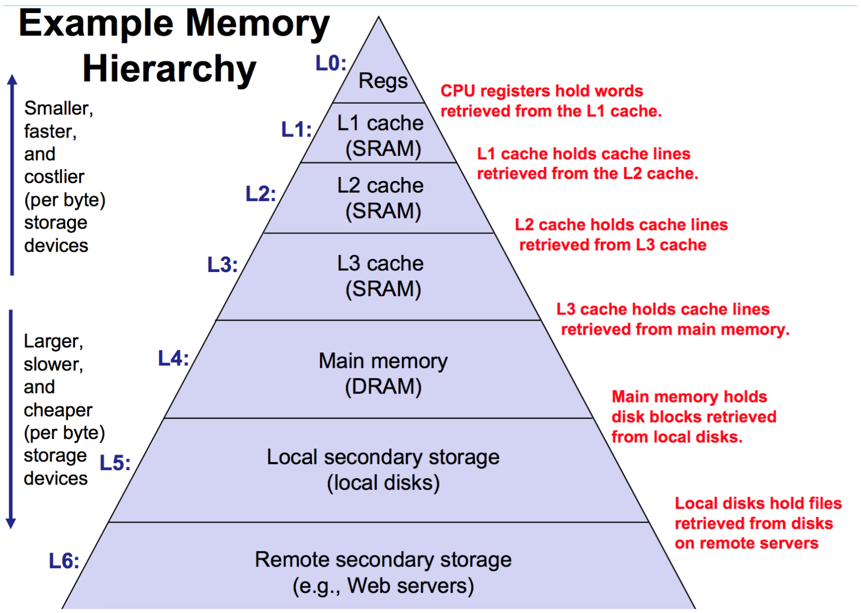
Lecture 10

<2016-05-04 Wed>

Contents

1 Memory Hiearchy



Example Memory Hierarchy

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

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

L0: Regs

L1: L1 cache (SRAM)

L2: L2 cache (SRAM)

L3: L3 cache (SRAM)

L4: Main memory (DRAM)

L5: Local secondary storage (local disks)

L6: Remote secondary storage (e.g., Web servers)

CPU registers hold words retrieved from the L1 cache.

L1 cache holds cache lines retrieved from the L2 cache.

L2 cache holds cache lines retrieved from L3 cache

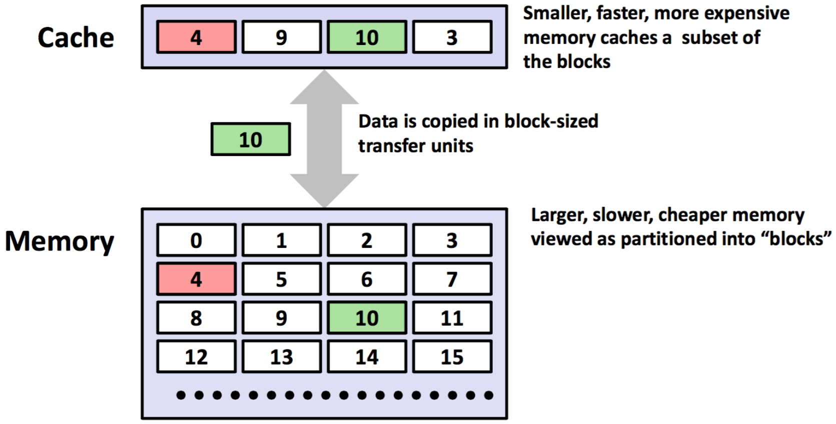
L3 cache holds cache lines retrieved from main memory.

Main memory holds disk blocks retrieved from local disks.

Local disks hold files retrieved from disks on remote servers

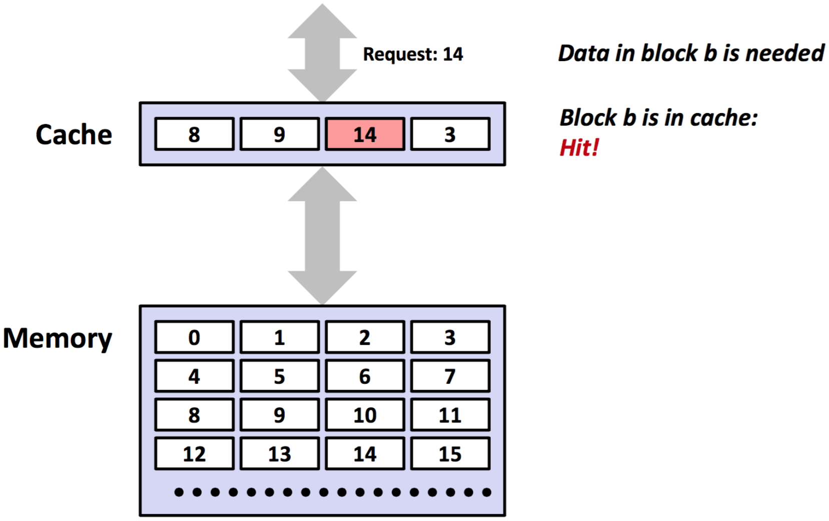
1.1 Cache

- cache: a smaller, faster storage device that acts as a staging area for a subset of 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
- **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 fastest storage near the top.


Cache
4
9
10
3
Smaller, faster, more expensive memory caches a subset of the blocks
10
Data is copied in block-sized transfer units
Memory
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Larger, slower, cheaper memory viewed as partitioned into "blocks"

1.1.1 Cache Concepts

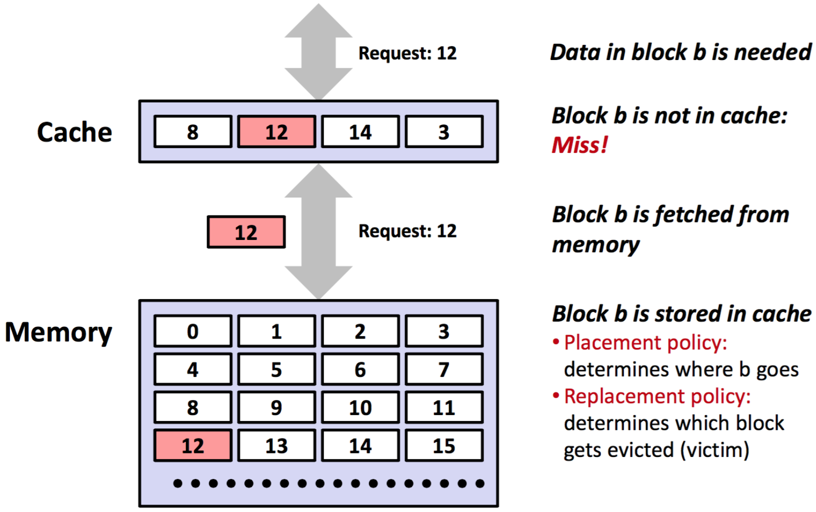
- hit



Data in block b is needed

Block b is in cache:
Hit!

- miss



Data in block b is needed

Block b is not in cache:
Miss!

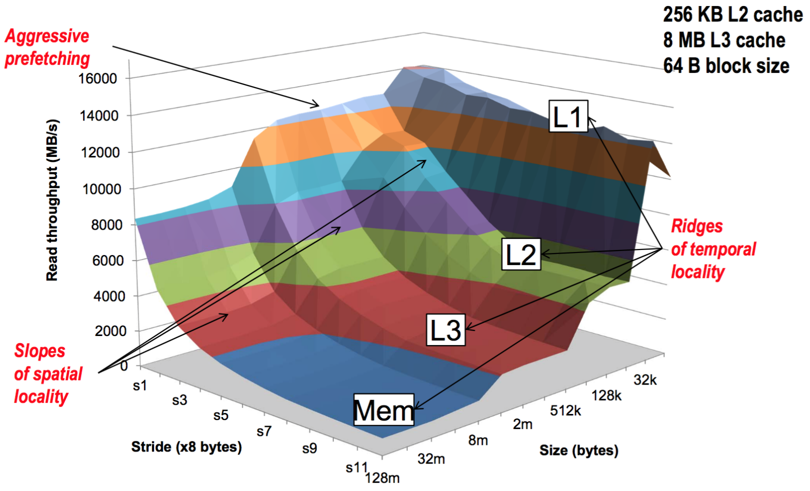
Block b is fetched from memory

Block b is stored in cache
- Placement policy: determines where b goes
- Replacement policy: determines which block gets evicted (victim)

1.1.2 Memory Mountain

- read throughput
  - number of bytes read from memory per second
- memory mountain

  - measured read throughput as a function of spatial and temporal locality
  - compact way to characterize memory system performance
    * Temporal locality refers to the reuse of specific data, and/or resources, within a relatively small time duration.
    * Spatial locality refers to the use of data elements within relatively close storage locations
    * stride
    * size



1.2 Locality

- principle of locality
  - programs tend to use data and instructions with address near or equal to those they have used recently
- temporal locality
  - recent 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

1.2.1 Example

```
int sum = 0;
for (i = 0; i < n; i++)
  sum += a[i];
```

- data references
  - spatial locality: reference array elements in succession (stride-1 reference pattern)
  - temporal locality: reference variable `sum` each iteration
- instruction references
  - spatial locality: reference instructions in sequence
  - temporal locality: cycle through loop repeatedly