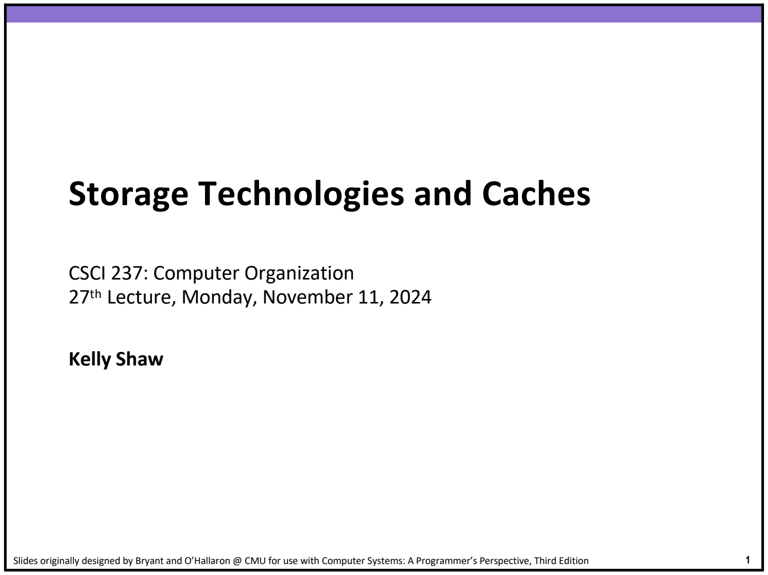
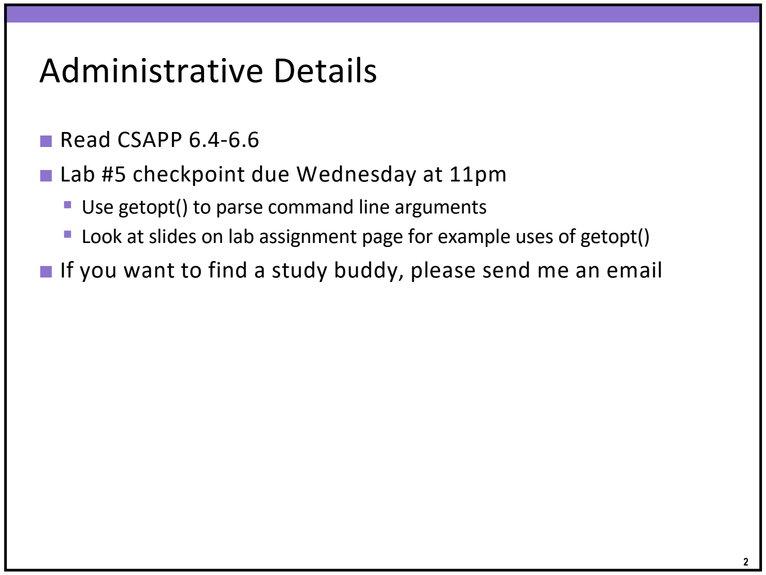


Storage Technologies and Caches

CSCI 237: Computer Organization
27th Lecture, Monday, November 11, 2024

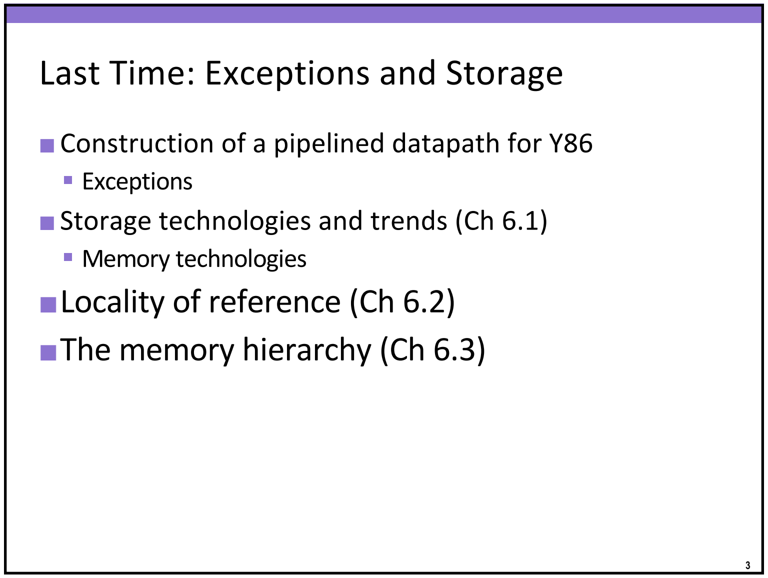
Kelly Shaw

Slides originally designed by Bryant and O'Hallaron @ CMU for use with Computer Systems: A Programmer's Perspective, Third Edition



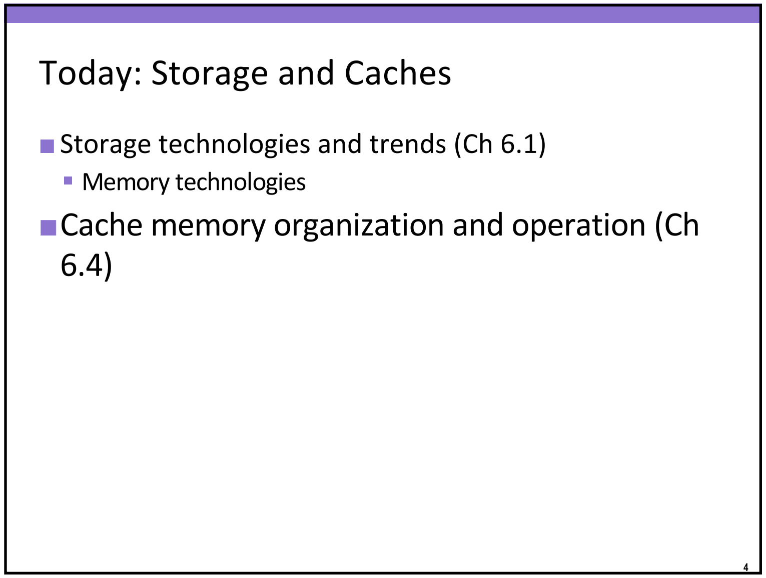
Administrative Details

- Read CSAPP 6.4-6.6
- Lab #5 checkpoint due Wednesday at 11pm
  - Use getopt() to parse command line arguments
  - Look at slides on lab assignment page for example uses of getopt()
- If you want to find a study buddy, please send me an email



Last Time: Exceptions and Storage

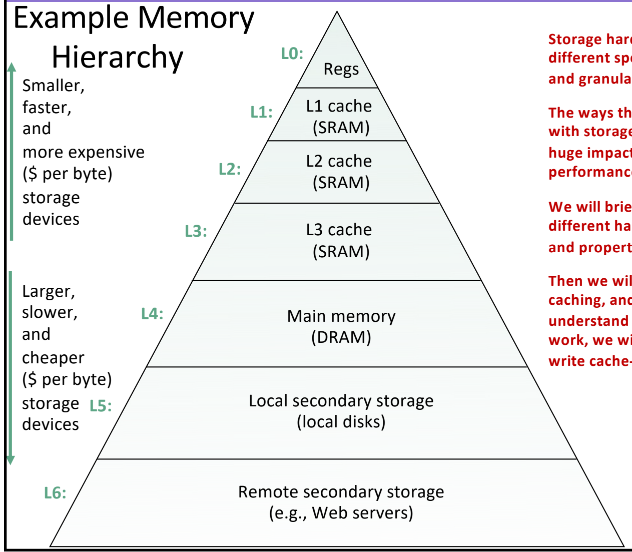
- Construction of a pipelined datapath for Y86
  - Exceptions
- Storage technologies and trends (Ch 6.1)
  - Memory technologies
- Locality of reference (Ch 6.2)
- The memory hierarchy (Ch 6.3)



Today: Storage and Caches

- Storage technologies and trends (Ch 6.1)
  - Memory technologies
- Cache memory organization and operation (Ch 6.4)

Example Memory Hierarchy



Smaller, faster, and more expensive ($ 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)

Storage hardware has different speeds, costs ($), and granularities of access.

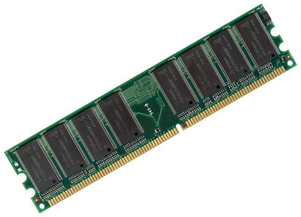
The ways that we interact with storage hardware has a huge impact on program performance.

We will briefly discuss the different hardware types and properties.

Then we will discuss caching, and once we understand how caches work, we will discuss how to write cache-efficient code.

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)
    - Stores bits in bistable cells
  - DRAM (Dynamic RAM)
    - Stores bits as charge on capacitors



SRAM vs DRAM Summary

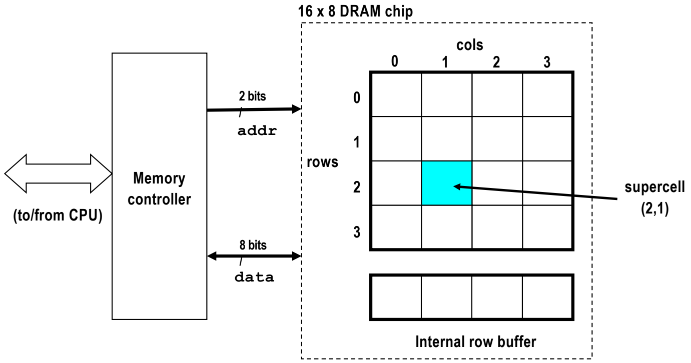
	Trans. per bit	Access time	Needs refresh?	Needs EDC?	Cost	Applications
SRAM	4 or 6	1X	No	No	1000x	Cache memories
DRAM	1	10X	Yes	Yes	1X	Main memories, frame buffers

Persistent?

Sensitive to disturbances?

Conventional DRAM Organization

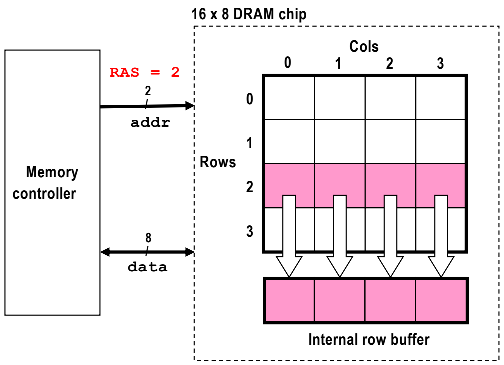
- d x w DRAM:
  - dw total bits organized as d supercells of size w bits



Reading DRAM Supercell (2,1)

Step 1(a): Row access strobe (RAS) selects row 2.

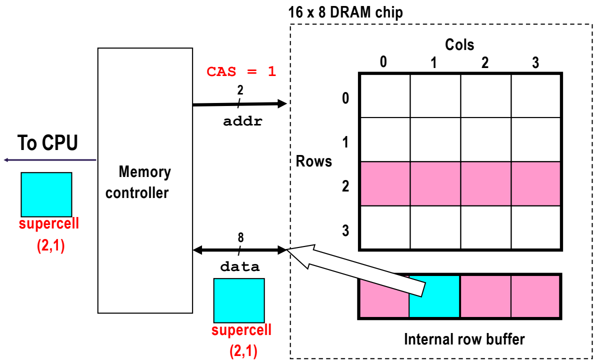
Step 1(b): Row 2 copied from DRAM array to row buffer.



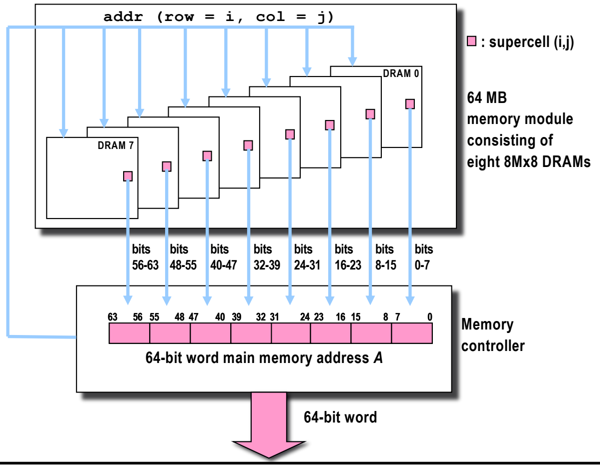
Reading DRAM Supercell (2,1)

Step 2(a): Column access strobe (CAS) selects column 1.

Step 2(b): Supercell (2,1) copied from buffer to data lines, and eventually back to the CPU.

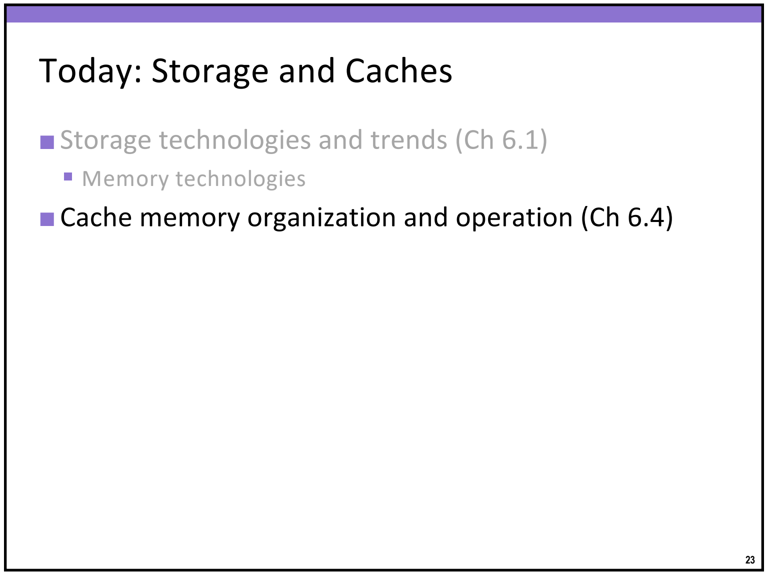


Memory Modules



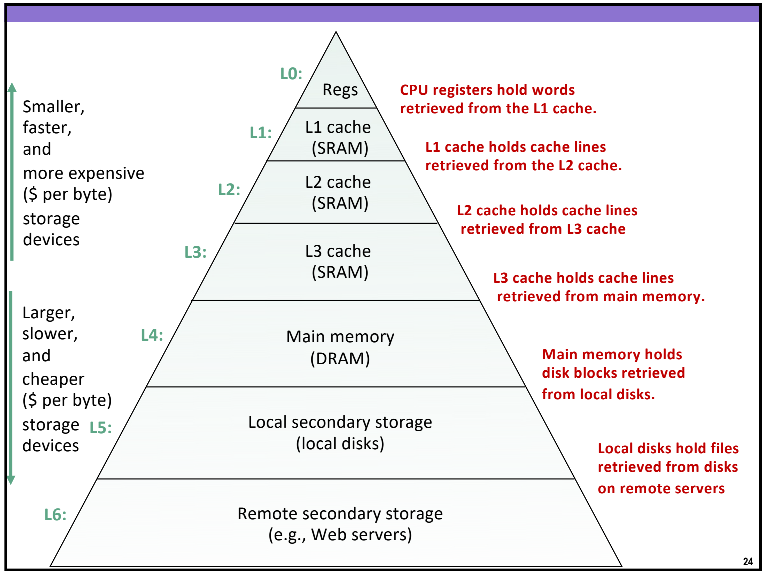
Enhanced DRAMs

- Basic DRAM cell has not changed since its invention in 1966.
  - Commercialized by Intel in 1970.
- DRAM cores with better interface logic and faster I/O:
  - Synchronous DRAM (SDRAM)
    - Uses a conventional clock signal instead of asynchronous control
    - Allows reuse of the row addresses which leads to faster output rate
  - Double data-rate synchronous DRAM (DDR SDRAM)
    - Double edge clocking sends two bits per cycle per pin
    - Different types distinguished by size of small prefetch buffer:
      - DDR (2 bits), DDR2 (4 bits), DDR3 (8 bits)
    - By 2010, standard for most server and desktop systems
    - Intel Core i7 supports DDR3 and DDR4 (8 bits, but faster) SDRAM



Today: Storage and Caches

- Storage technologies and trends (Ch 6.1)
  - Memory technologies
- Cache memory organization and operation (Ch 6.4)



Smaller, faster, and more expensive ($ per byte) storage devices

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

L0: Regs — CPU registers hold words retrieved from the L1 cache.

L1: L1 cache (SRAM) — L1 cache holds cache lines retrieved from the L2 cache.

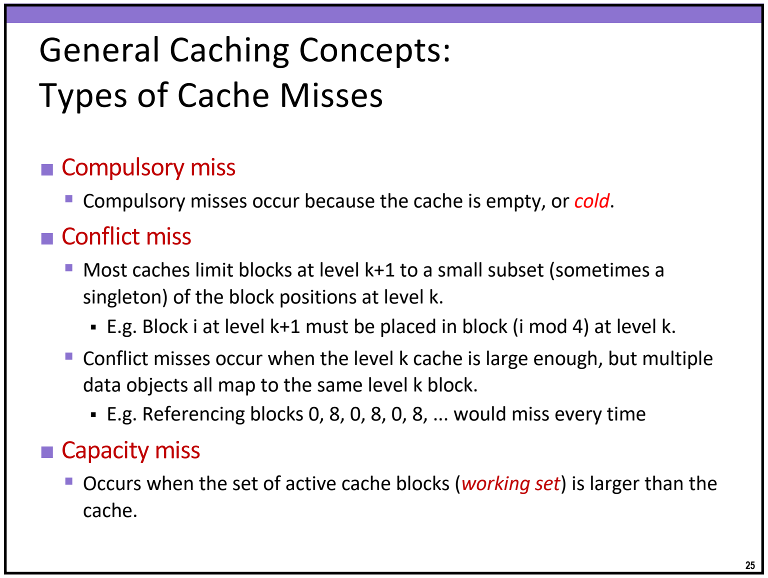
L2: L2 cache (SRAM) — L2 cache holds cache lines retrieved from L3 cache

L3: L3 cache (SRAM) — L3 cache holds cache lines retrieved from main memory.

L4: Main memory (DRAM) — Main memory holds disk blocks retrieved from local disks.

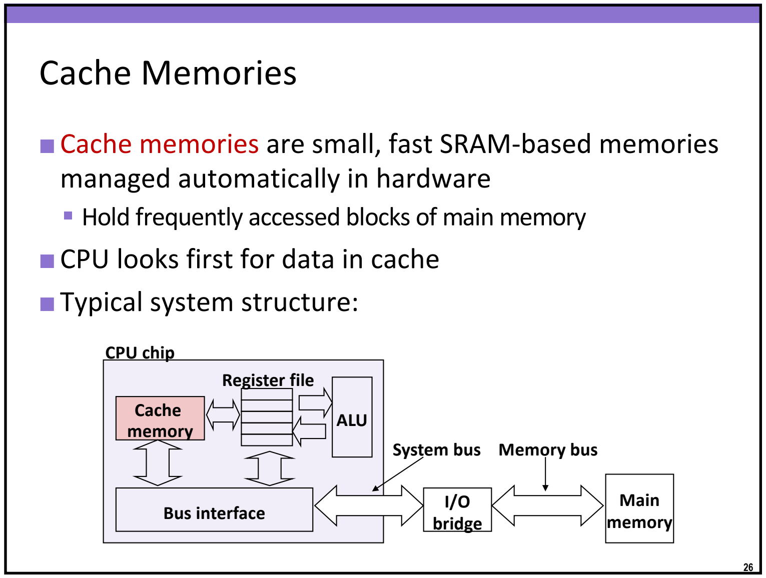
L5: Local secondary storage (local disks) — Local disks hold files retrieved from disks on remote servers

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



General Caching Concepts: Types of Cache Misses

- Compulsory miss
  - Compulsory misses occur because the cache is empty, or *cold*.
- 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.



Cache Memories

- Cache memories are small, fast SRAM-based memories managed automatically in hardware
  - Hold frequently accessed blocks of main memory
- CPU looks first for data in cache
- Typical system structure:

CPU chip

Register file

Cache memory

ALU

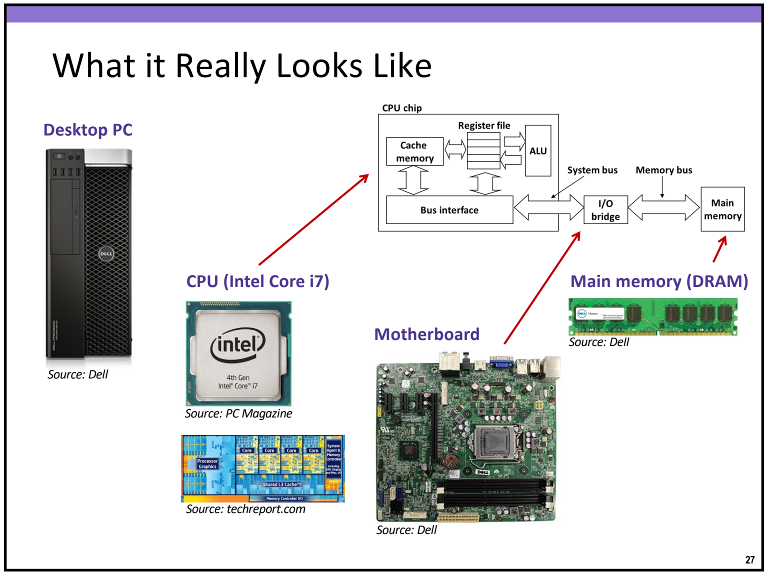
Bus interface

System bus

Memory bus

I/O bridge

Main memory



What it Really Looks Like

Desktop PC

Source: Dell

CPU (Intel Core i7)

Source: PC Magazine

Source: techreport.com

Motherboard

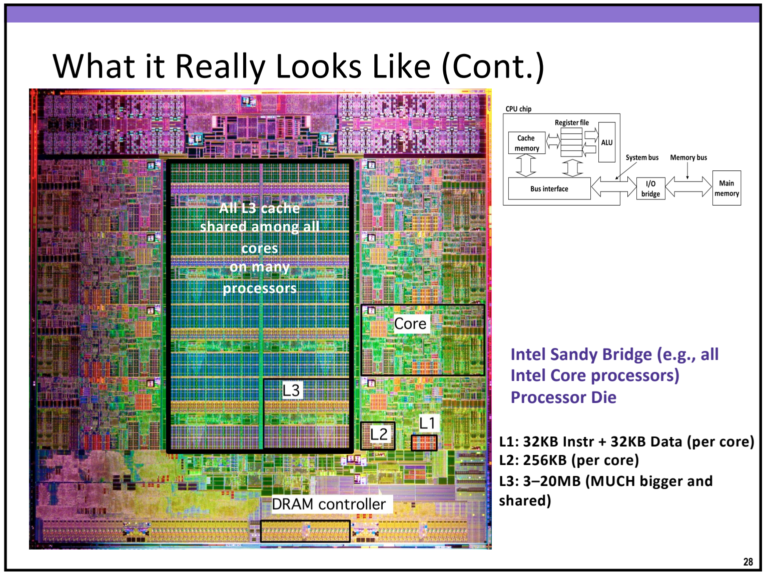
Source: Dell

Main memory (DRAM)

Source: Dell

CPU chip
Register file
Cache memory
ALU
System bus
Memory bus
Bus interface
I/O bridge
Main memory

27



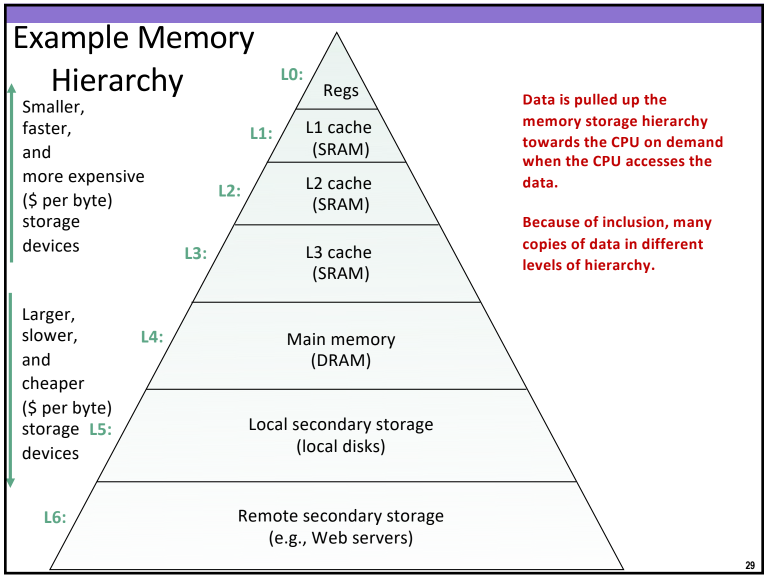
What it Really Looks Like (Cont.)

All L3 cache shared among all cores on many processors
Core
L3
L2
L1
DRAM controller
CPU chip
Register file
Cache memory
ALU
System bus
Memory bus
Bus interface
I/O bridge
Main memory

Intel Sandy Bridge (e.g., all Intel Core processors) Processor Die

L1: 32KB Instr + 32KB Data (per core)
L2: 256KB (per core)
L3: 3–20MB (MUCH bigger and shared)

28



Example Memory Hierarchy

Smaller, faster, and more expensive ($ per byte) storage devices

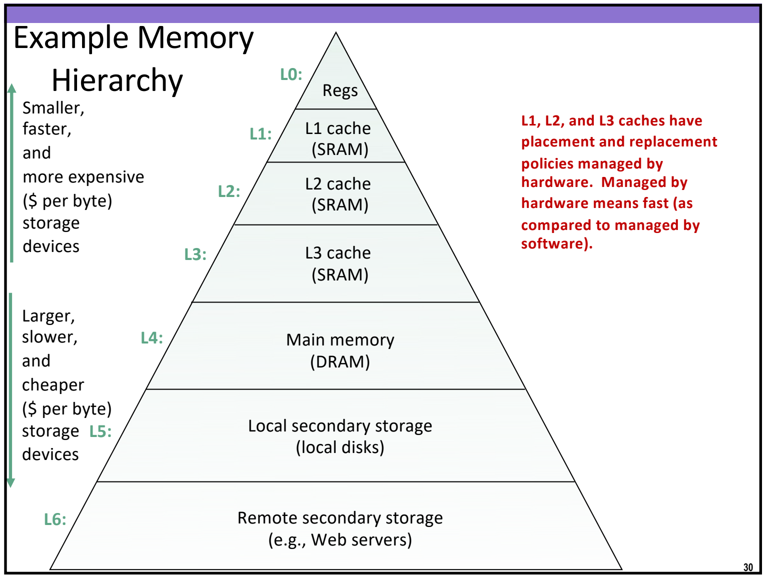
Larger, slower, and cheaper ($ per byte) storage devices

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)

Data is pulled up the memory storage hierarchy towards the CPU on demand when the CPU accesses the data.

Because of inclusion, many copies of data in different levels of hierarchy.

29



Example Memory Hierarchy

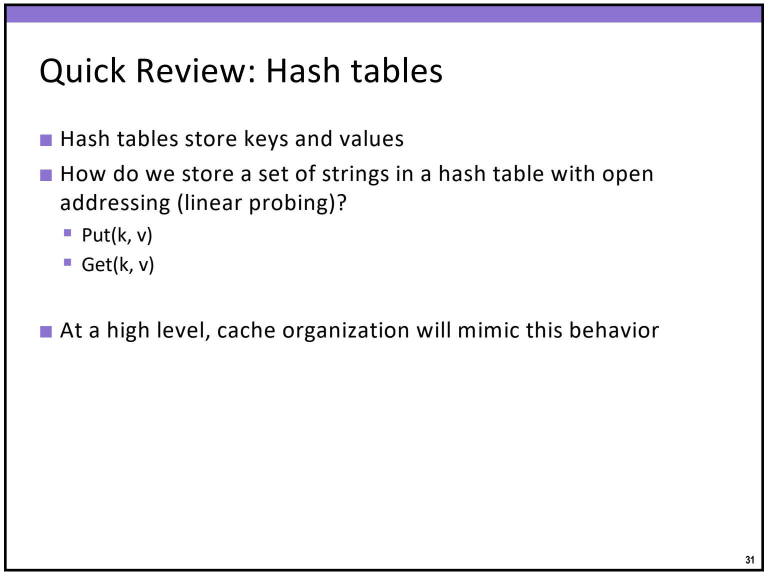
Smaller, faster, and more expensive ($ per byte) storage devices

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)

L1, L2, and L3 caches have placement and replacement policies managed by hardware. Managed by hardware means fast (as compared to managed by software).

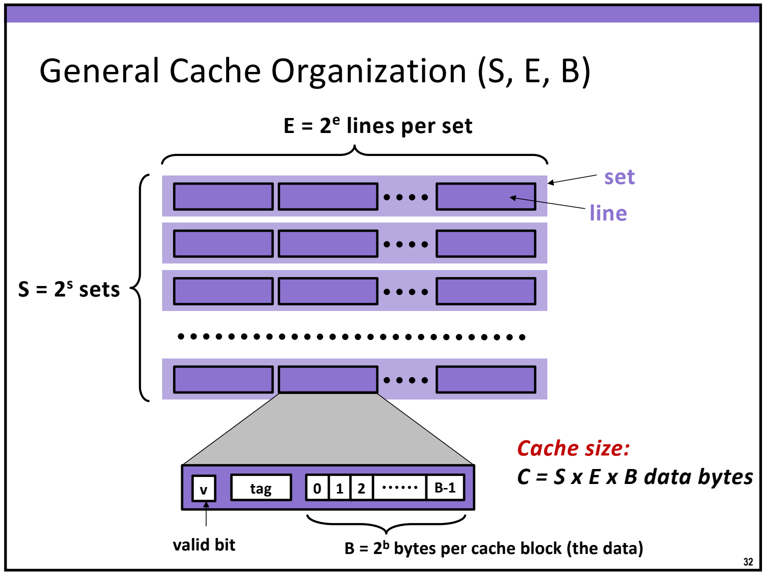
30



Quick Review: Hash tables

- Hash tables store keys and values
- How do we store a set of strings in a hash table with open addressing (linear probing)?
  - Put(k, v)
  - Get(k, v)
- At a high level, cache organization will mimic this behavior

31

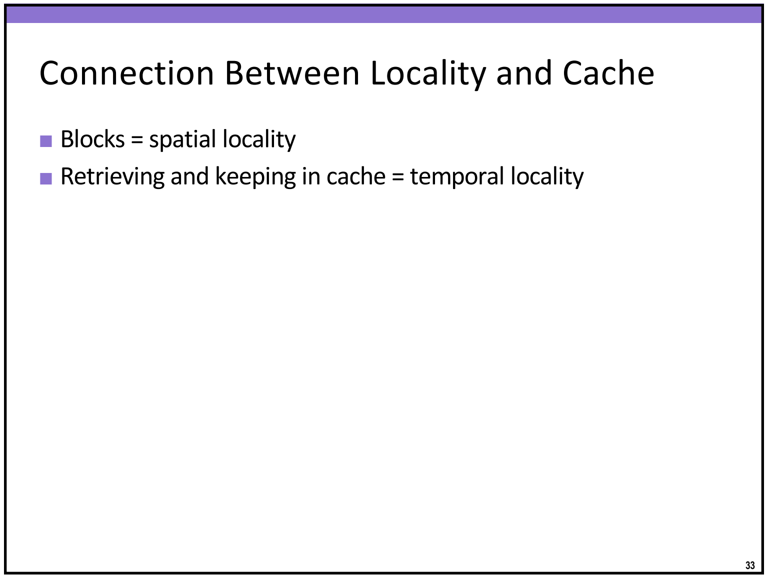


General Cache Organization (S, E, B)

Cache size:

C = S x E x B data bytes

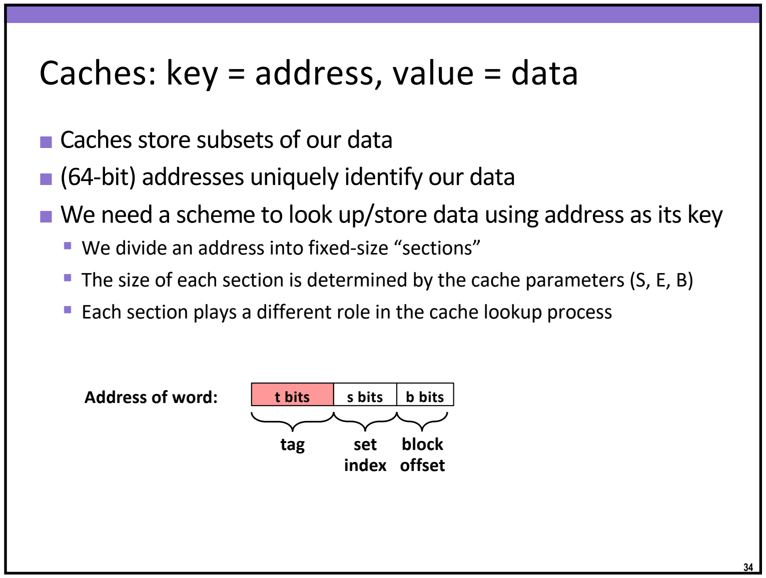
32



Connection Between Locality and Cache

- Blocks = spatial locality
- Retrieving and keeping in cache = temporal locality

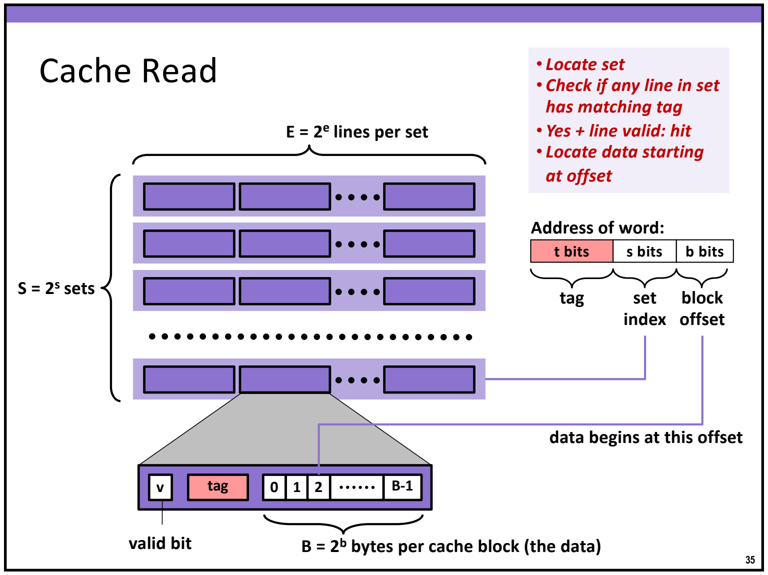
33



Caches: key = address, value = data

- Caches store subsets of our data
- (64-bit) addresses uniquely identify our data
- We need a scheme to look up/store data using address as its key
  - We divide an address into fixed-size "sections"
  - The size of each section is determined by the cache parameters (S, E, B)
  - Each section plays a different role in the cache lookup process

34

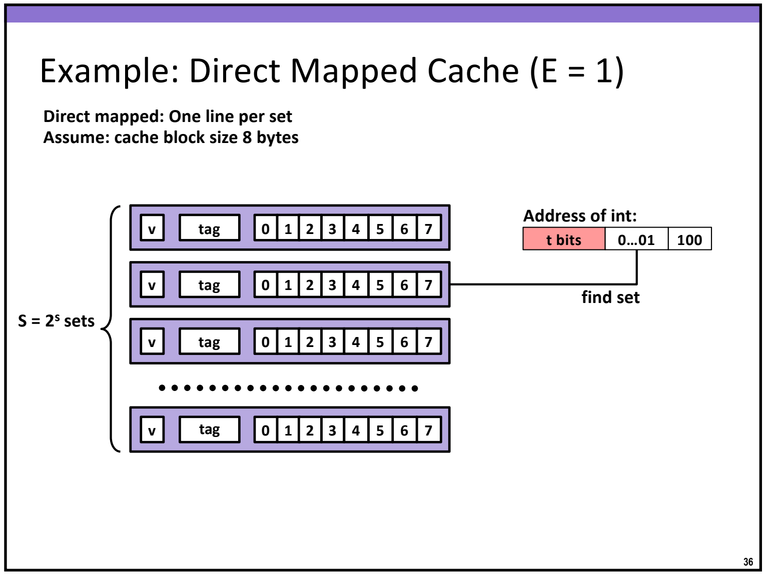


Cache Read

- *Locate set*
- *Check if any line in set has matching tag*
- *Yes + line valid: hit*
- *Locate data starting at offset*

35

35



Example: Direct Mapped Cache (E = 1)

Direct mapped: One line per set
Assume: cache block size 8 bytes

36

36