CSCI 210: Computer Architecture
Lecture 33: Caches

Stephen Checkoway

Oberlin College

Jan. 3, 2022

Slides from Cynthia Taylor

Announcements

- Problem Set 11 due Friday (it’ll be up tonight)

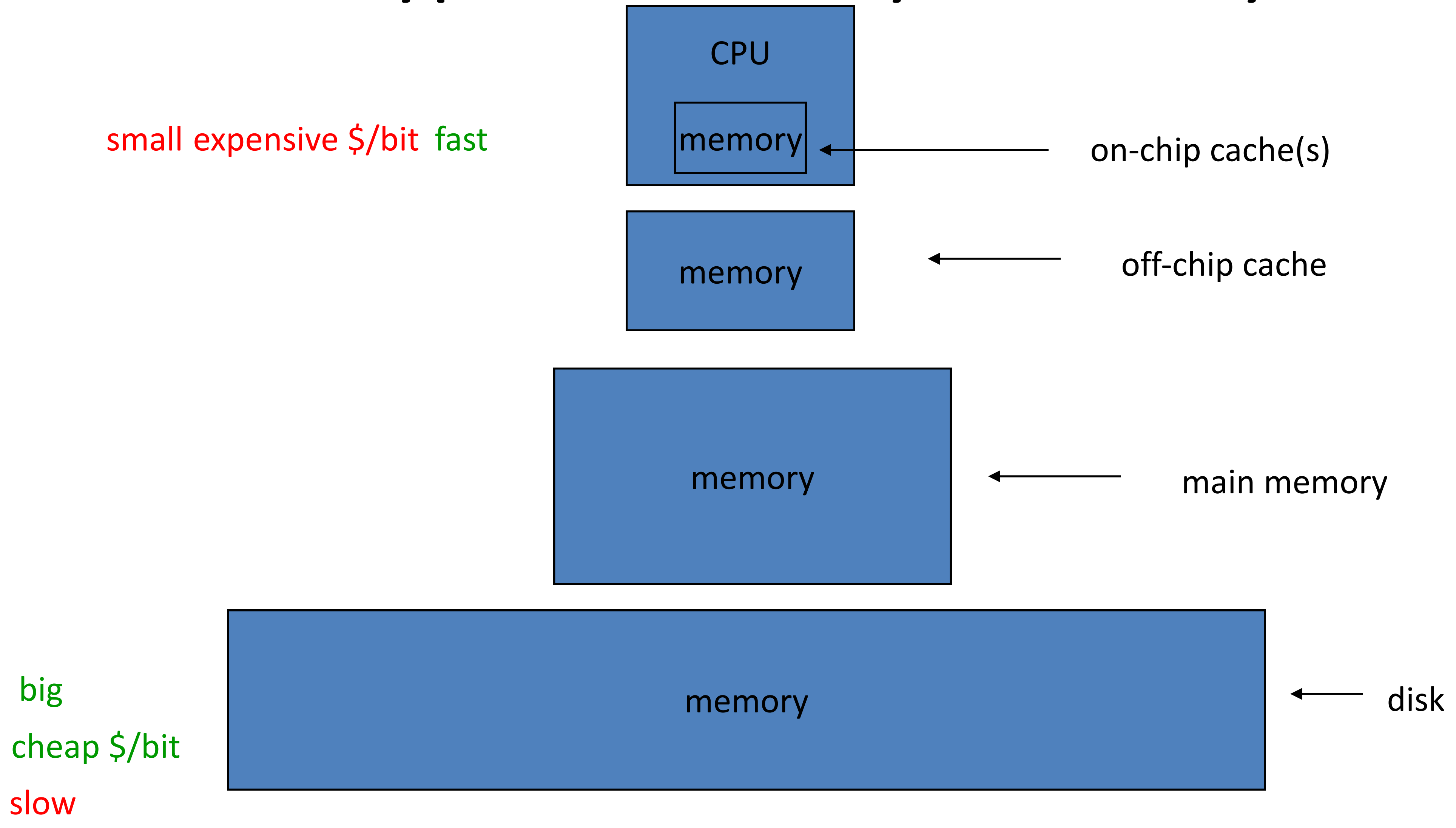
- Cache Lab (final project) due at the end of our scheduled final exam period

- Office Hours Tuesday 13:30 – 14:30
  - Zoom

Memory

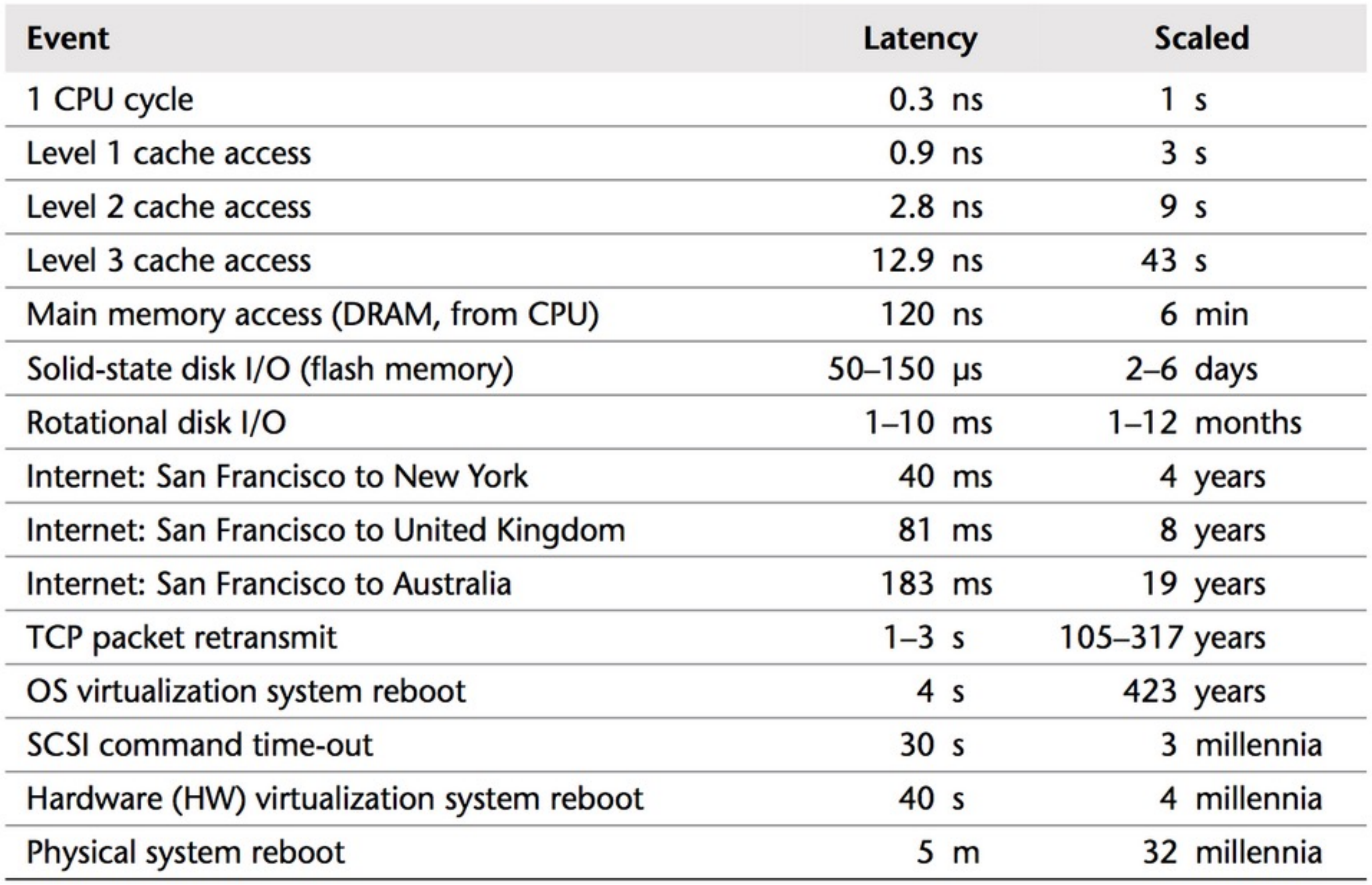
- So far we have only looked at the CPU/datapath
- Now we’re going to look at memory

A typical memory hierarchy



Latency

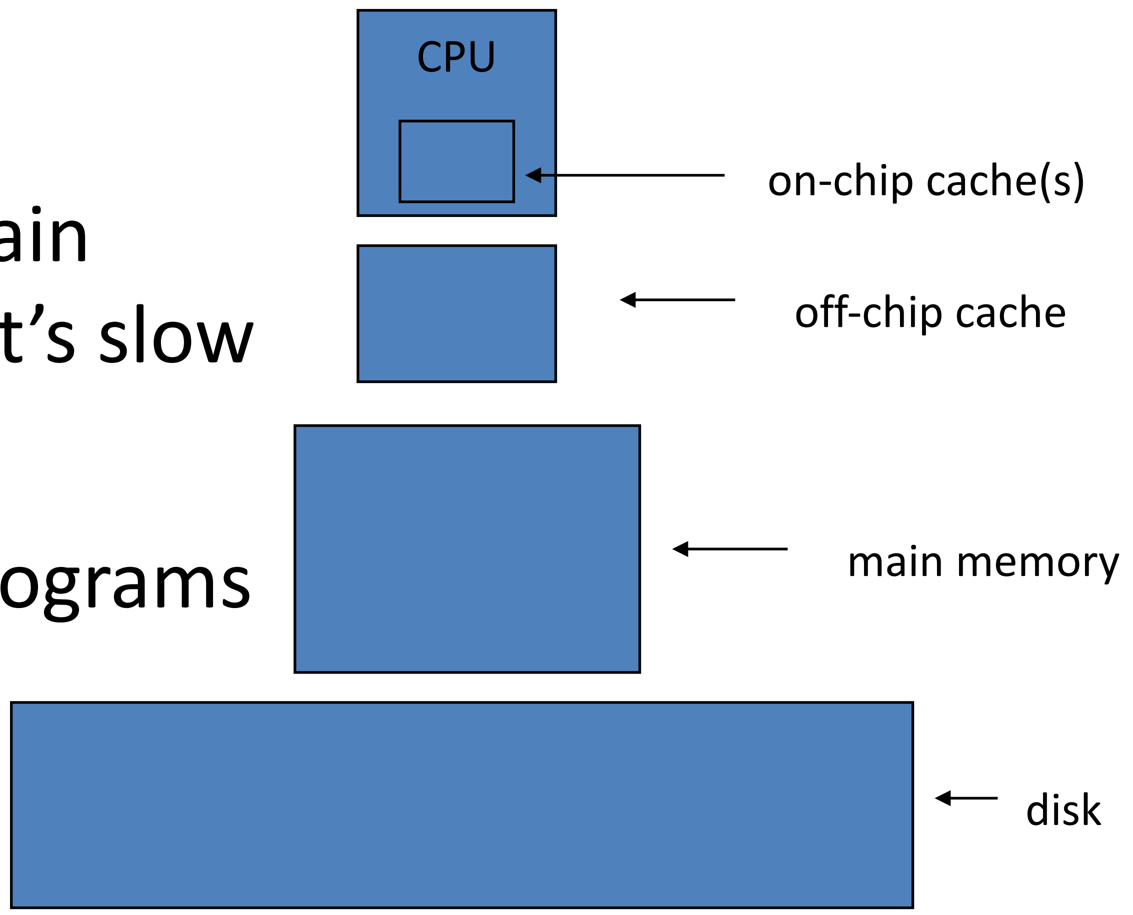
Table 2.2 Example Time Scale of System Latencies



Event	Latency	Scaled
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access (DRAM, from CPU)	120 ns	6 min
Solid-state disk I/O (flash memory)	50–150 μs	2–6 days
Rotational disk I/O	1–10 ms	1–12 months
Internet: San Francisco to New York	40 ms	4 years
Internet: San Francisco to United Kingdom	81 ms	8 years
Internet: San Francisco to Australia	183 ms	19 years
TCP packet retransmit	1–3 s	105–317 years
OS virtualization system reboot	4 s	423 years
SCSI command time-out	30 s	3 millennia
Hardware (HW) virtualization system reboot	40 s	4 millennia
Physical system reboot	5 m	32 millennia

Memory

- Everything is on disk, very few things are in the registers
- Want to avoid going to main memory or disk because it's slow
- Take advantage of how programs actually access memory

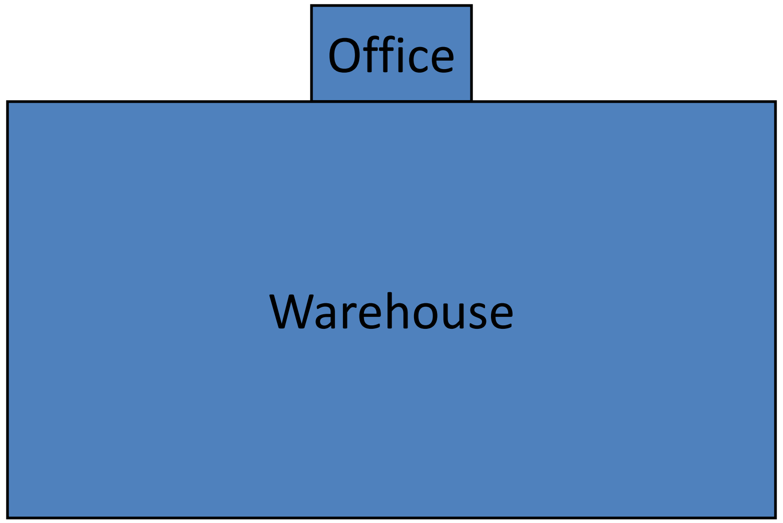


Principle of Locality

- Programs access a small proportion of their address space at any time
- Temporal locality
  - Items accessed recently are likely to be accessed again soon
  - e.g., instructions in a loop, registers spilled to the stack
- Spatial locality
  - Items near those accessed recently are likely to be accessed soon
  - E.g., sequential instruction access, array data

Library

- You have a huge library with EVERY book ever made.
- Getting a book from the library's warehouse takes 15 minutes.
- You can't serve enough people if every checkout takes 15 minutes.
- You have some small shelves in the front office.

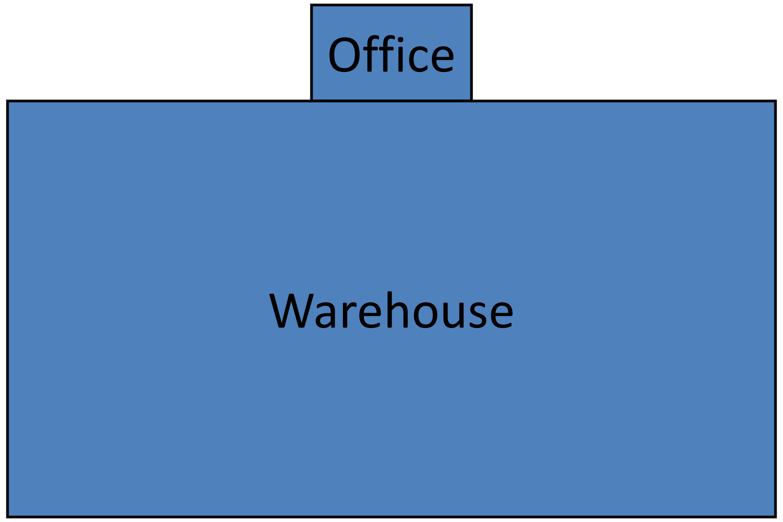


Here are some suggested improvements to the library:

1. Whenever someone checks out a book, just keep it in the front office for a while in case someone else wants to check it out.
2. Watch the trends in books and attempt to guess books that will be checked out soon – put those in the front office.
3. Whenever someone checks out a book in a series, grab the other books in the series and put them in the front.
4. Buy motorcycles to ride in the warehouse to get the books faster

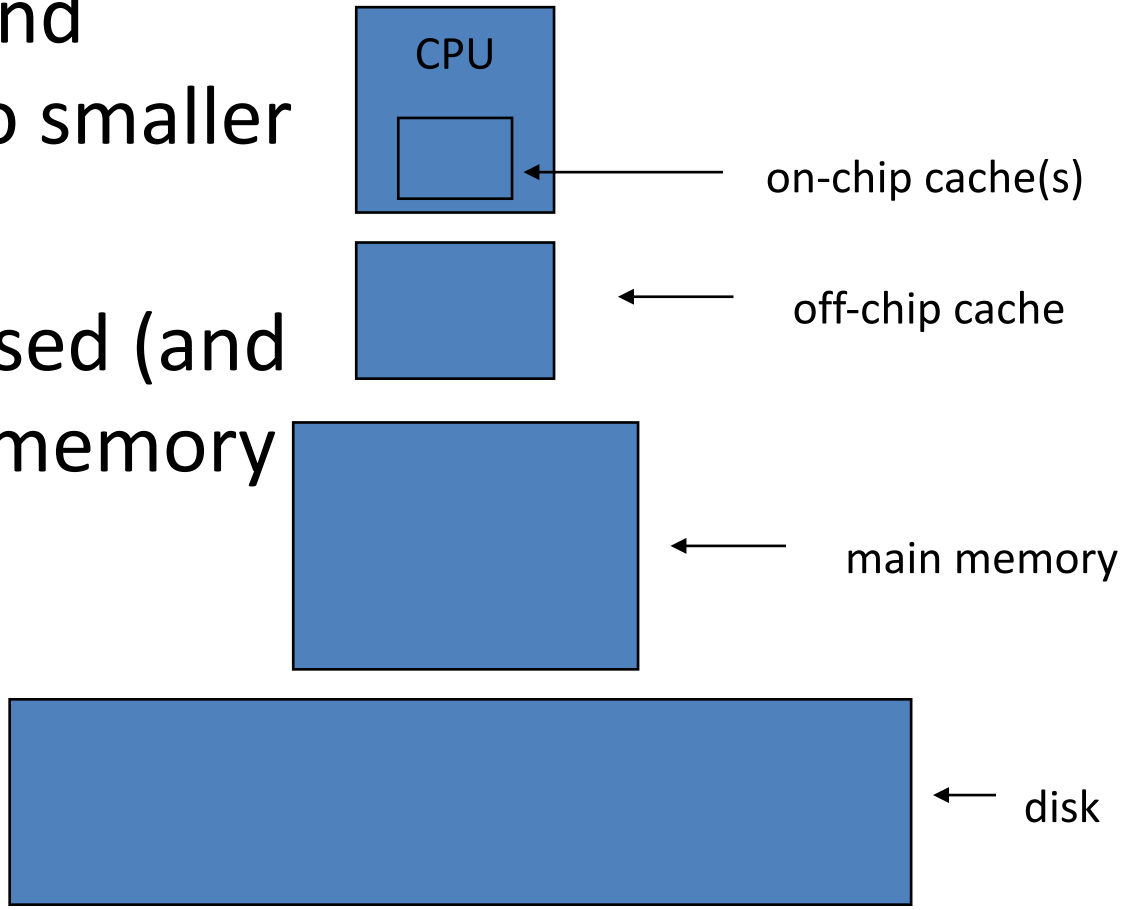
Extending the analogy to locality for caches, which pair of changes most closely matches the analogous cache locality?

Selection	Spatial	Temporal
A	2	1
B	4	2
C	4	3
D	3	1
E	None of the above	



Taking Advantage of Locality

- Store everything on disk
- Copy recently accessed (and nearby) items from disk to smaller main memory
- Copy more recently accessed (and nearby) items from main memory to cache

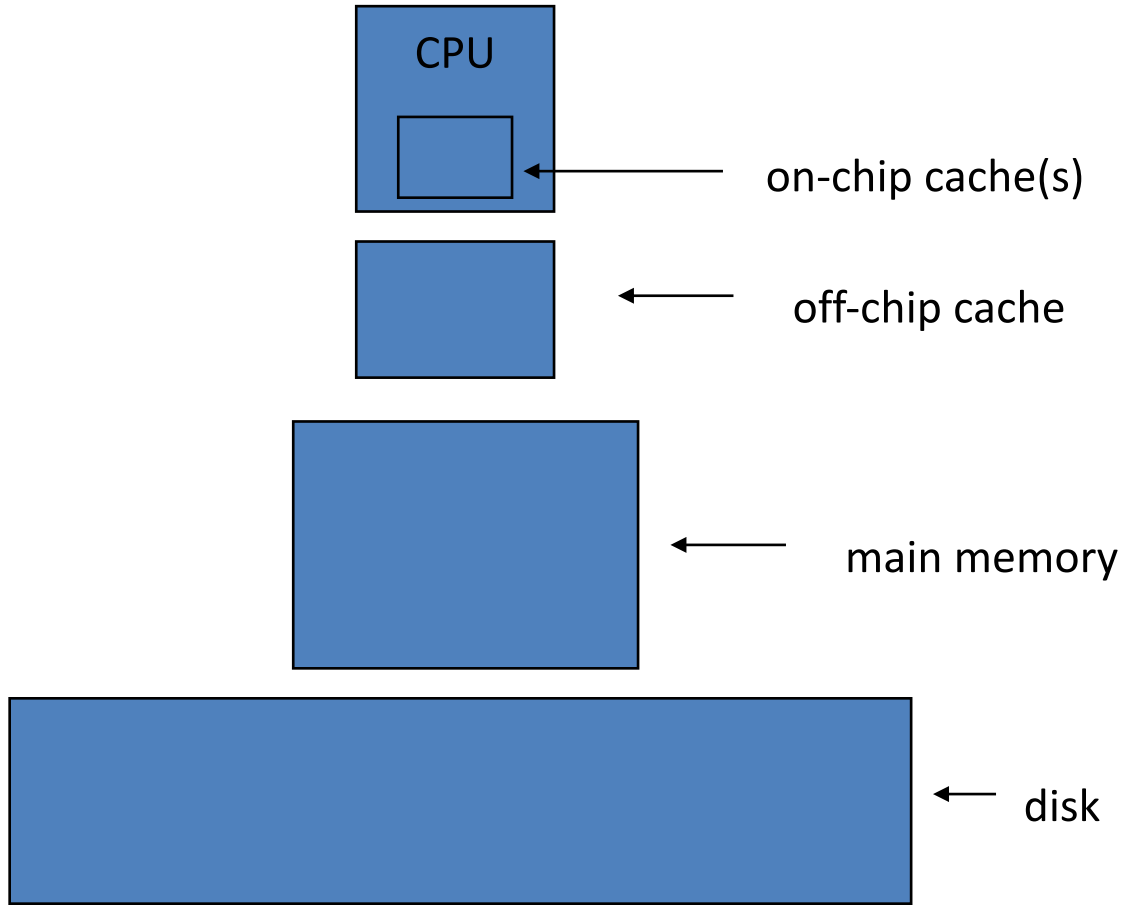


We know SRAM is very fast, expensive ($/GB), and small. We also know disks are slow, inexpensive ($/GB), and large. Which statement best describes the role of cache when it works.

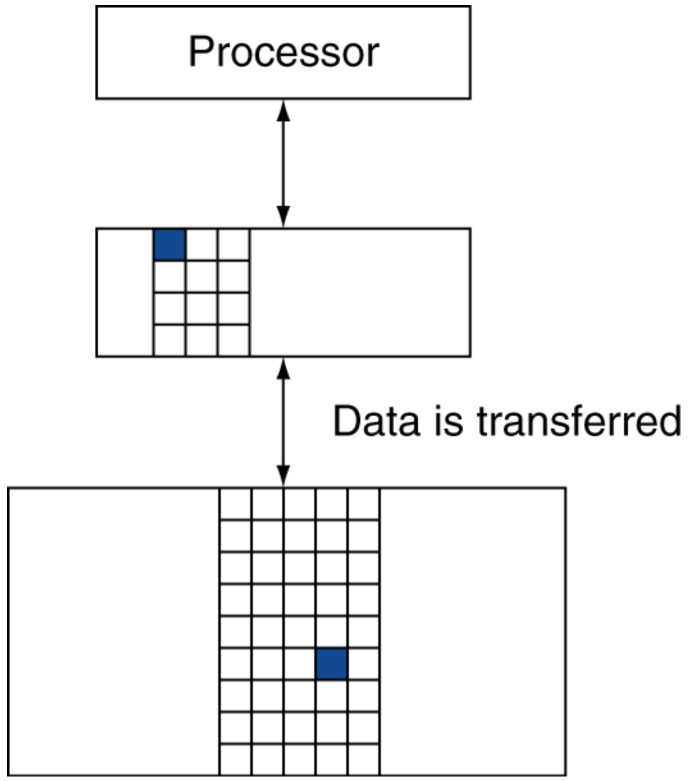
Selection	Role of caching
A	Locality allows us to keep frequently touched data in SRAM.
B	Locality allows us the illusion of memory as fast as SRAM but as large as a disk.
C	SRAM is too expensive to have large – so it must be small and caching helps use it well.
D	Disks are too slow – we have to have something faster for our processor to access.
E	None of these accurately describes the roll of cache.

Memory Access

- Use main memory addresses
- When looking for data, check
  - 1. cache
  - 2. main memory
  - 3. disk



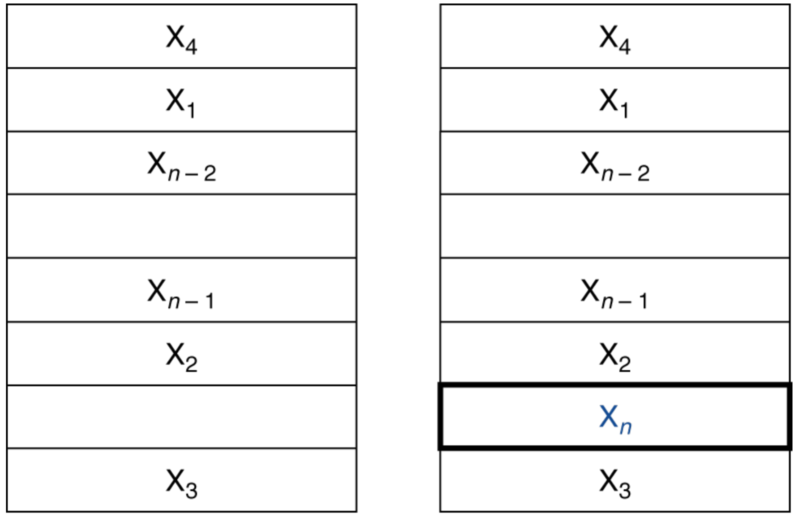
Memory Hierarchy Terms



- Block: unit of copying
  - May be multiple words
  - On x86-64, a block is 64 bytes
- Hit: access satisfied by upper level
  - Hit ratio: hits/accesses
- Miss: block copied from lower level
  - Time taken: miss penalty
  - Miss ratio: misses/accesses
    = 1 – hit ratio

Cache Memory

- Basic problem: Smaller than main memory, but we look for things based on larger main memory addresses
- Given accesses $X_1, \ldots, X_{n-1}, X_n$

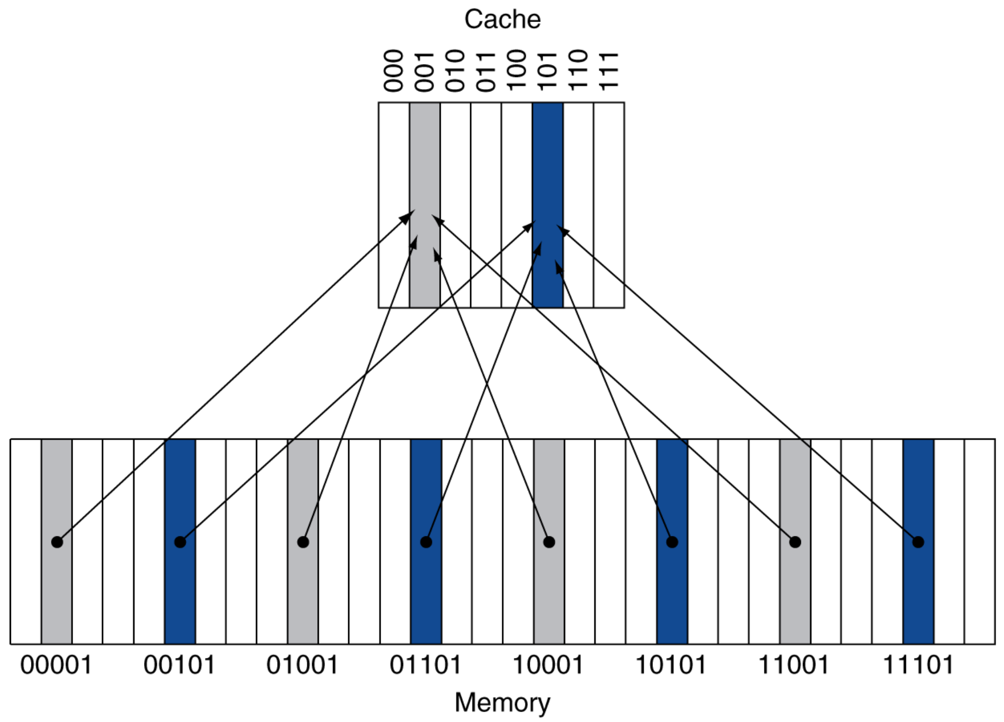


a. Before the reference to X_n b. After the reference to X_n

- How do we know if the data is present?
- Where do we look?

Direct Mapped Cache

- Location determined by address
- Direct mapped: only one possible location
  - (Block address) modulo (#Blocks in cache)



- #Blocks is a power of 2
- Use low-order address bits

Problem: Collisions

- Multiple addresses map to the same position on the cache via mod

- How do we know which particular block is stored in a cache location?
  - Store block address as well as the data
  - Actually, only need the high-order bits
  - Called the **tag**

Cache layout (so far)

- Tag stores high-order bits of address
- Data stores all of the data for the block (e.g., 32 bytes)

Tag	Data
0000420	FE FF 3C 7F …
0012345	32 A0 5C 21 …
000F3CB	00 00 00 00 …

How do we know if it’s in the cache?

- What if there is no data in a location?
  - Valid bit: 1 = present, 0 = not present
  - Initially 0

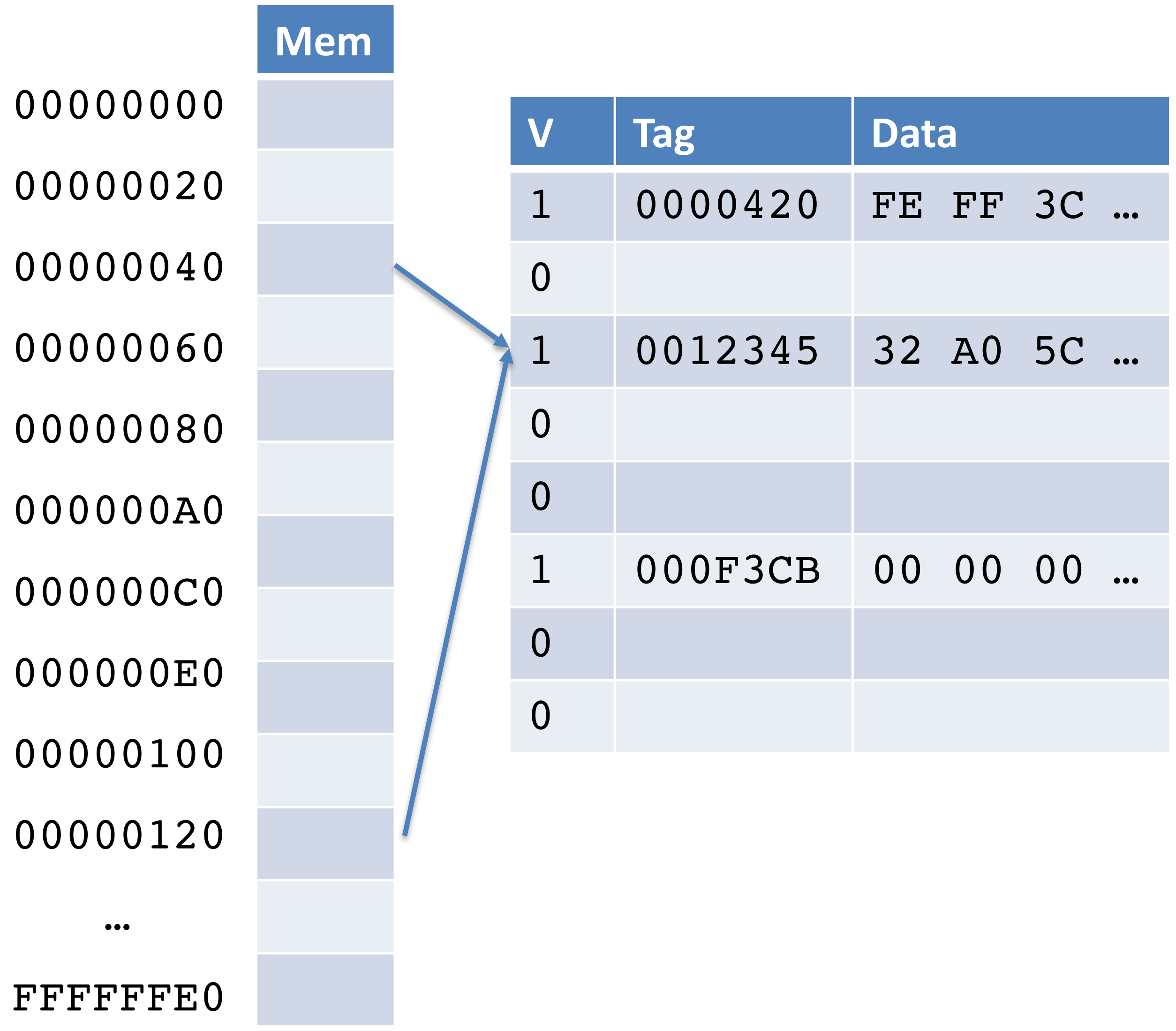
Cache layout (so far)

- Valid stores 1 if data is present in cache
- Tag stores high-order bits of address
- Data stores all of the data for the block (e.g., 32 bytes)

Valid	Tag	Data
1	0000420	FE FF 3C 7F …
0		
1	0012345	32 A0 5C 21 …
0		
0		
1	000F3CB	00 00 00 00 …
0		
0		

High-level cache strategy

- Divide all of memory into consecutive blocks
- Copy data (memory ↔ cache) one block at a time
- Cache lookup:
  - Get the index of the block in the cache from the address
  - Check the valid bit; compare the tag to the address



V	Tag	Data
1	0000420	FE FF 3C …
0		
1	0012345	32 A0 5C …
0		
0		
1	000F3CB	00 00 00 …
0		
0		

Reading

- Next lecture: More Caches!
  - Section 6.3
- Problem Set 11 due Friday