inst.eecs.berkeley.edu/~cs61c

CS61C : Machine Structures

Lecture 33
Caches III

Lecturer PSOE Dan Garcia

www.cs.berkeley.edu/~ddgarcia

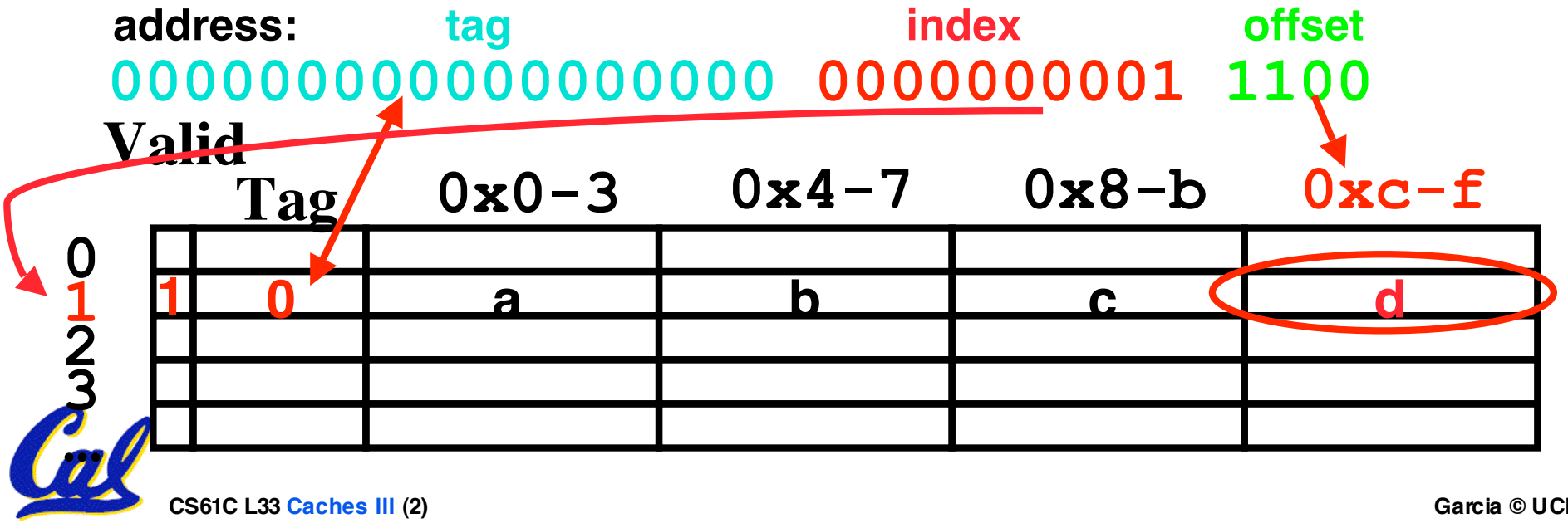
Attend Cal Day tomorrow ⇒

There will be some really cool exhibits you should take the time to see! Concrete canoes, robot races, free massages, and presentations by Dan's UCBUGG and GamesCrafters groups.

www.berkeley.edu/calday/

Review…

- Mechanism for transparent movement of data among levels of a storage hierarchy
  - set of address/value bindings
  - address => index to set of candidates
  - compare desired address with tag
  - service hit or miss
    - load new block and binding on miss



Big Endian vs. Little Endian

Big-endian and little-endian derive from Jonathan Swift's *Gulliver's Travels* in which the Big Endians were a political faction that broke their eggs at the large end ("the primitive way") and rebelled against the Lilliputian King who required his subjects (the Little Endians) to break their eggs at the small end.

- **The order in which BYTES are stored in memory**
- **Bits always stored as usual. (E.g., 0xC2=0b 1100 0010)**

Consider the number 1025 as we normally write it:

BYTE3 BYTE2 BYTE1 BYTE0

00000000 00000000 00000100 00000001

Big Endian

- ADDR3 ADDR2 ADDR1 ADDR0
  BYTE0 BYTE1 BYTE2 BYTE3
  00000001 00000100 00000000 00000000
- ADDR0 ADDR1 ADDR2 ADDR3
  BYTE3 BYTE2 BYTE1 BYTE0
  00000000 00000000 00000100 00000001

Little Endian

- ADDR3 ADDR2 ADDR1 ADDR0
  BYTE3 BYTE2 BYTE1 BYTE0
  00000000 00000000 00000100 00000001
- ADDR0 ADDR1 ADDR2 ADDR3
  BYTE0 BYTE1 BYTE2 BYTE3
  00000001 00000100 00000000 00000000

www.webopedia.com/TERM/b/big_endian.html
searchnetworking.techtarget.com/sDefinition/0,,sid7_gci211659,00.html
www.noveltheory.com/TechPapers/endian.asp
en.wikipedia.org/wiki/Big_endian

Memorized this table yet?

- Blah blah Cache size 16KB blah blah 2^{23} blocks blah blah how many bits?

- Answer! 2^{XY} means…

X=0 ⇒	no suffix			Y=0 ⇒	1
X=1 ⇒	kibi	~ Kilo	10^3	Y=1 ⇒	2
X=2 ⇒	mebi	~ Mega	10^6	Y=2 ⇒	4
X=3 ⇒	gibi	~ Giga	10^9	Y=3 ⇒	8
X=4 ⇒	tebi	~ Tera	10^{12}	Y=4 ⇒	16
X=5 ⇒	pebi	~ Peta	10^{15}	Y=5 ⇒	32
X=6 ⇒	exbi	~ Exa	10^{18}	Y=6 ⇒	64
X=7 ⇒	zebi	~ Zetta	10^{21}	Y=7 ⇒	128
X=8 ⇒	yobi	~ Yotta	10^{24}	Y=8 ⇒	256
				Y=9 ⇒	512

How Much Information IS that?

www.sims.berkeley.edu/research/projects/how-much-info-2003/

- Print, film, magnetic, and optical storage media produced about 5 exabytes of new information in 2002. 92% of the new information stored on magnetic media, mostly in hard disks.
- Amt of new information stored on paper, film, magnetic, & optical media ~doubled in last 3 yrs
- Information flows through electronic channels -- telephone, radio, TV, and the Internet -- contained ~18 exabytes of new information in 2002, 3.5x more than is recorded in storage media. 98% of this total is the information sent & received in telephone calls - incl. voice & data on fixed lines & wireless.
  - WWW ⇒ 170 Tb of information on its surface; in volume 17x the size of the Lib. of Congress print collections.
  - *Instant messaging* ⇒ 5×10^9 msgs/day (750GB), 274 TB/yr.
  - *Email* ⇒ ~400 PB of new information/year worldwide.

Block Size Tradeoff (1/3)

- Benefits of Larger Block Size
  - Spatial Locality: if we access a given word, we're likely to access other nearby words soon
  - Very applicable with Stored-Program Concept: if we execute a given instruction, it's likely that we'll execute the next few as well
  - Works nicely in sequential array accesses too

Block Size Tradeoff (2/3)

- Drawbacks of Larger Block Size
  - Larger block size means larger miss penalty
    - on a miss, takes longer time to load a new block from next level
  - If block size is too big relative to cache size, then there are too few blocks
    - Result: miss rate goes up
- In general, minimize Average Memory Access Time (AMAT)

  = Hit Time

  + Miss Penalty x Miss Rate

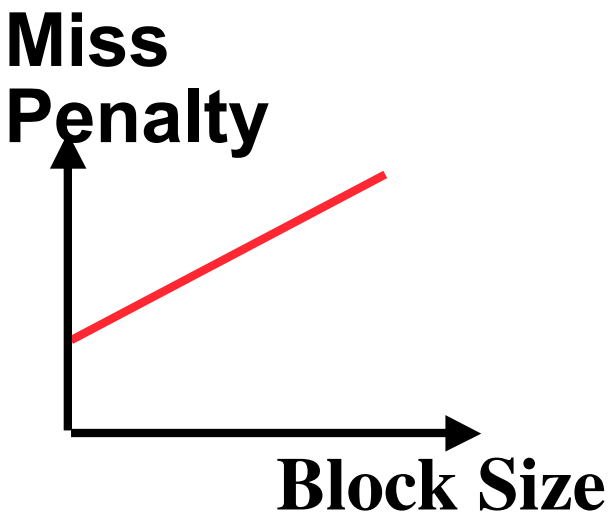
Block Size Tradeoff (3/3)

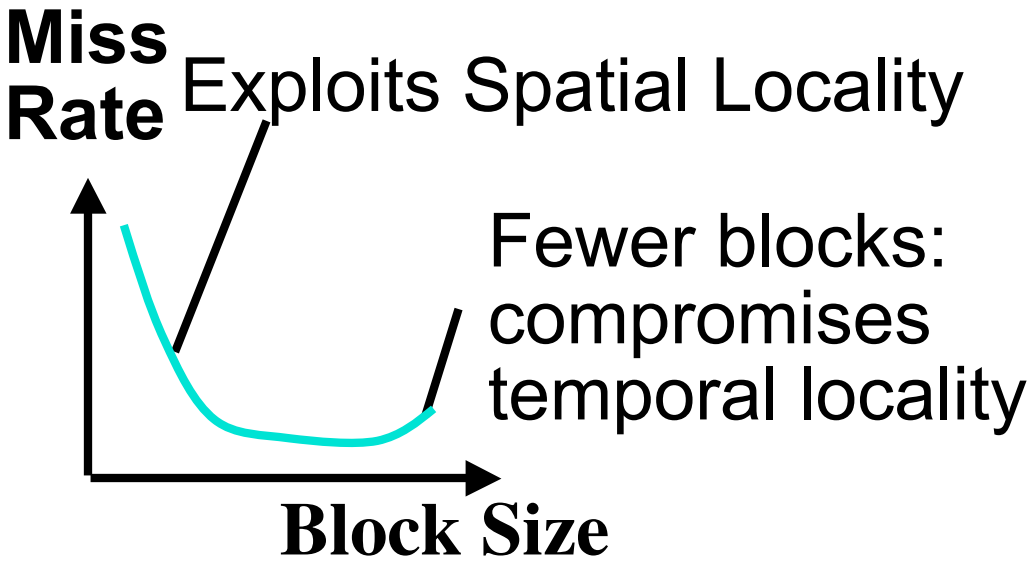
- Hit Time = time to find and retrieve data from current level cache
- Miss Penalty = average time to retrieve data on a current level miss (includes the possibility of misses on successive levels of memory hierarchy)
- Hit Rate = % of requests that are found in current level cache
- Miss Rate = 1 - Hit Rate

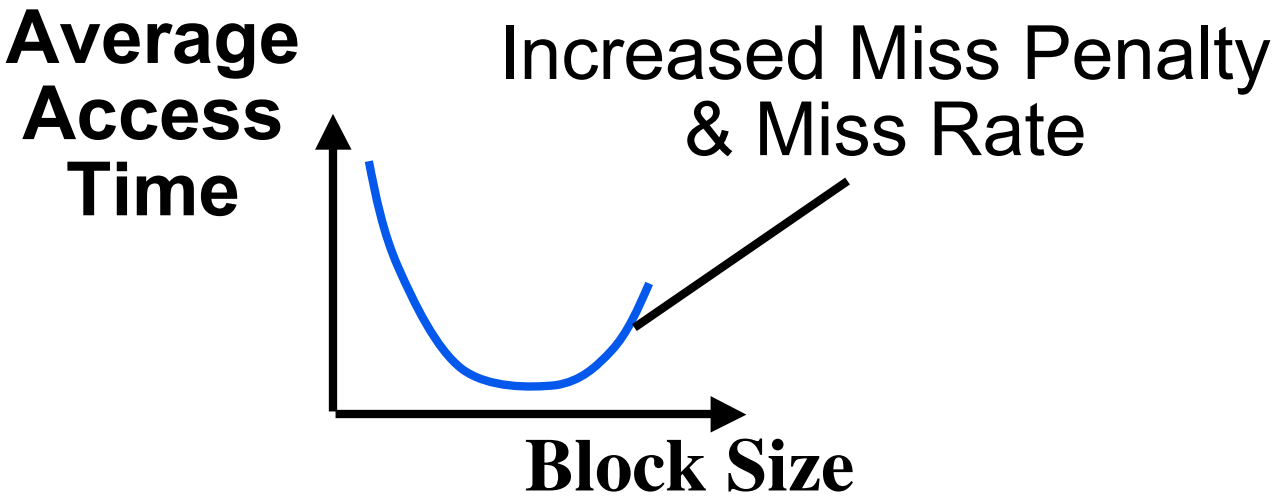
Extreme Example: One Big Block

- Cache Size = 4 bytes Block Size = 4 bytes
  - Only ONE entry in the cache!
- If item accessed, likely accessed again soon
  - But unlikely will be accessed again immediately!
- The next access will likely to be a miss again
  - Continually loading data into the cache but discard data (force out) before use it again
  - Nightmare for cache designer: Ping Pong Effect

Block Size Tradeoff Conclusions







Administrivia

- Any administrivia?

Types of Cache Misses (1/2)

- "Three Cs" Model of Misses
- 1st C: Compulsory Misses
  - occur when a program is first started
  - cache does not contain any of that program's data yet, so misses are bound to occur
  - can't be avoided easily, so won't focus on these in this course

Types of Cache Misses (2/2)

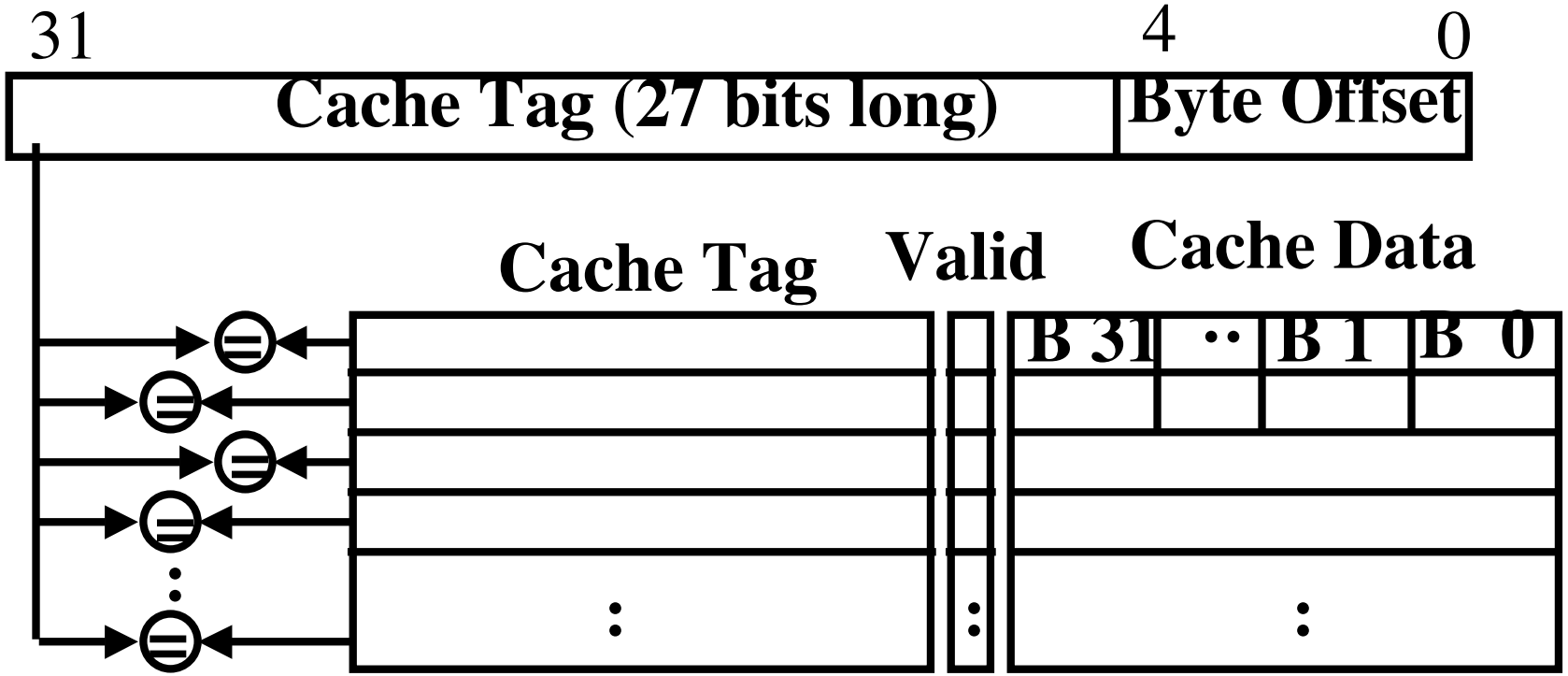
- 2nd C: Conflict Misses
  - miss that occurs because two distinct memory addresses map to the same cache location
  - two blocks (which happen to map to the same location) can keep overwriting each other
  - big problem in direct-mapped caches
  - how do we lessen the effect of these?
- Dealing with Conflict Misses
  - Solution 1: Make the cache size bigger
    - Fails at some point
  - Solution 2: Multiple distinct blocks can fit in the same cache Index?

Fully Associative Cache (1/3)

- Memory address fields:
  - Tag: same as before
  - Offset: same as before
  - Index: non-existant
- What does this mean?
  - no “rows”: any block can go anywhere in the cache
  - must compare with all tags in entire cache to see if data is there

Fully Associative Cache (2/3)

- **Fully Associative Cache (e.g., 32 B block)**
  - **compare tags in parallel**



Fully Associative Cache (3/3)

- Benefit of Fully Assoc Cache
  - No Conflict Misses (since data can go anywhere)
- Drawbacks of Fully Assoc Cache
  - Need hardware comparator for every single entry: if we have a 64KB of data in cache with 4B entries, we need 16K comparators: infeasible

Third Type of Cache Miss

- **Capacity Misses**
  - miss that occurs because the cache has a limited size
  - miss that would not occur if we increase the size of the cache
  - sketchy definition, so just get the general idea
- This is the primary type of miss for Fully Associative caches.

N-Way Set Associative Cache (1/4)

- Memory address fields:
  - Tag: same as before
  - Offset: same as before
  - Index: points us to the correct "row" (called a set in this case)
- So what's the difference?
  - each set contains multiple blocks
  - once we've found correct set, must compare with all tags in that set to find our data

N-Way Set Associative Cache (2/4)

- Summary:
  - cache is direct-mapped w/respect to sets
  - each set is fully associative
  - basically N direct-mapped caches working in parallel: each has its own valid bit and data

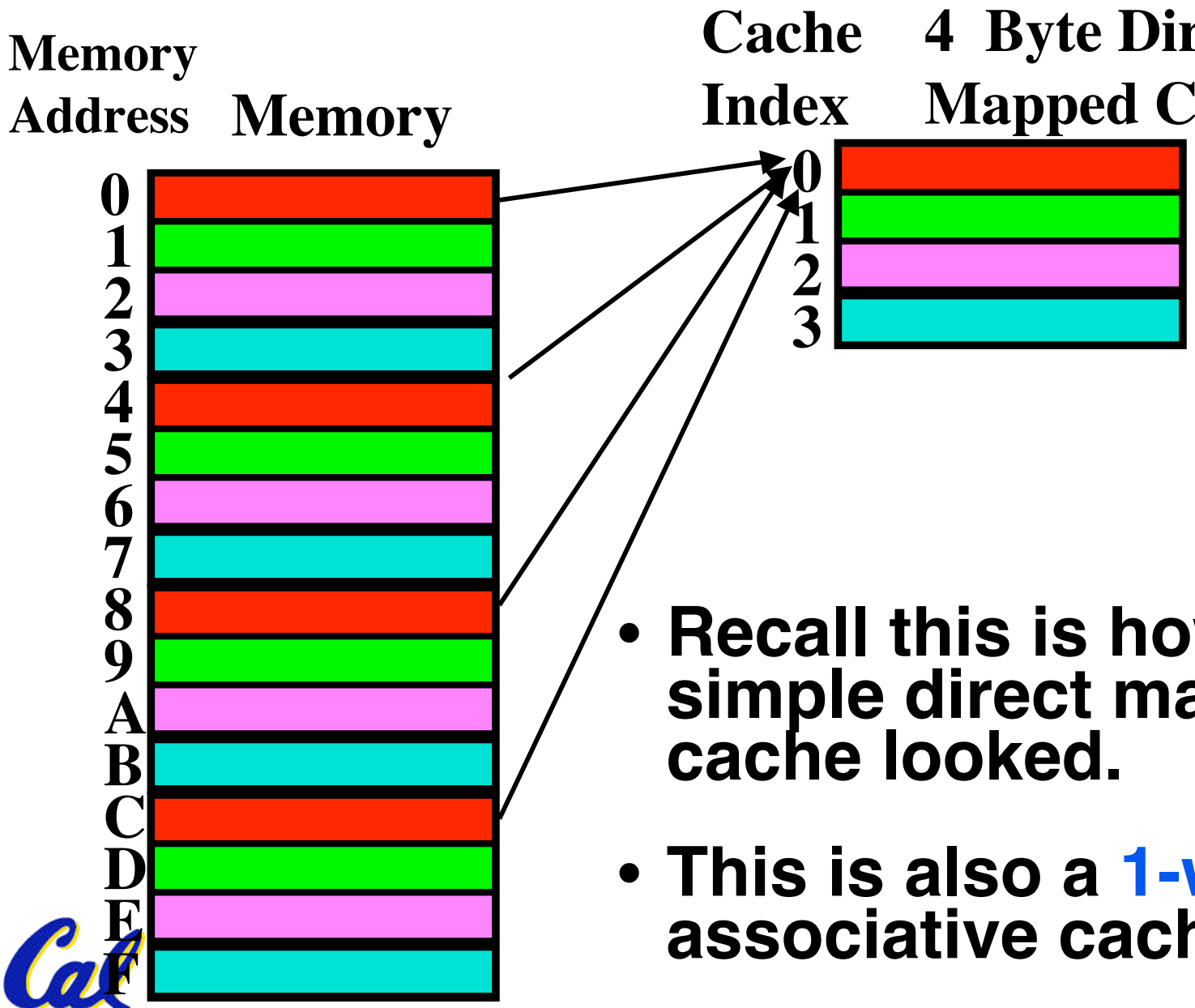
N-Way Set Associative Cache (3/4)

- Given memory address:
  - Find correct set using Index value.
  - Compare Tag with all Tag values in the determined set.
  - If a match occurs, hit!, otherwise a miss.
  - Finally, use the offset field as usual to find the desired data within the block.

N-Way Set Associative Cache (4/4)

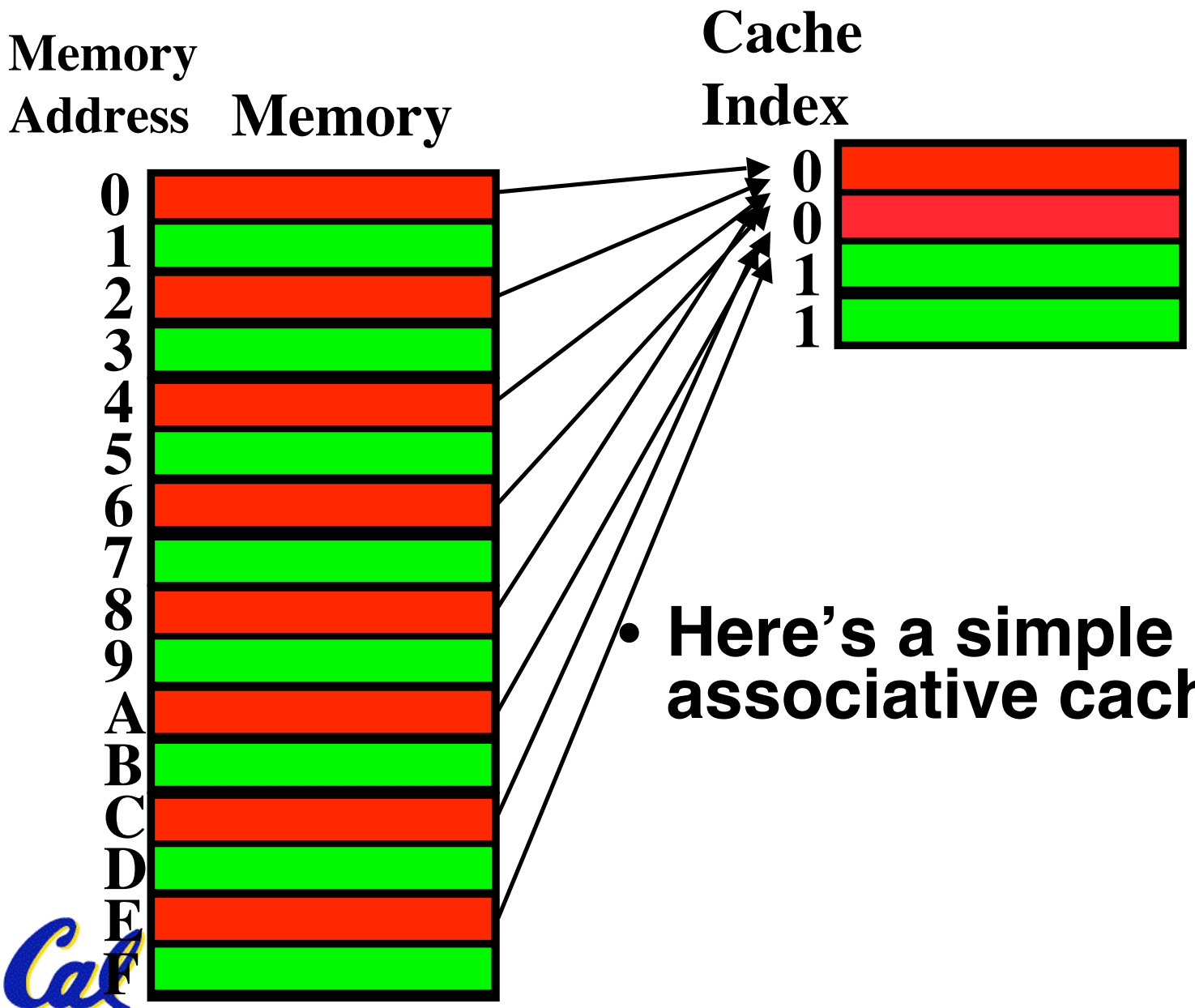
- What's so great about this?
  - even a 2-way set assoc cache avoids a lot of conflict misses
  - hardware cost isn't that bad: only need N comparators
- In fact, for a cache with M blocks,
  - it's Direct-Mapped if it's 1-way set assoc
  - it's Fully Assoc if it's M-way set assoc
  - so these two are just special cases of the more general set associative design

Associative Cache Example



- Recall this is how a simple direct mapped cache looked.
- This is also a 1-way set-associative cache!

Associative Cache Example



- Here's a simple 2 way set associative cache.

Peer Instructions

1. In the last 10 years, the gap between the access time of DRAMs & the cycle time of processors has decreased. (I.e., is closing)
2. A 2-way set-associative cache can be outperformed by a direct-mapped cache.
3. Larger block size ⇒ lower miss rate

	ABC
1:	FFF
2:	FFT
3:	FTF
4:	FTT
5:	TFF
6:	TFT
7:	TTF
8:	TTT

Peer Instructions Answer

1. That was was one of the motivation for caches in the first place -- that the memory gap is big and widening.

2. Sure, consider the caches from the previous slides with the following workload: 0, 2, 0, 4, 2
   2-way: 0m, 2m, 0h, 4m, 2m; DM: 0m, 2m, 0h, 4m, 2h

3. Larger block size ⇒ lower miss rate, true until a certain point, and then the ping-pong effect takes over

1. In the last 10 years, the gap between the access time of DRAMs & the cycle time of processors has decreased. (I.e., is closing)

2. A 2-way set-associative cache can be outperformed by a direct-mapped cache.

3. Larger block size ⇒ lower miss rate

	ABC
1:	FFF
2:	FFT
3:	FTF
4:	FTT
5:	TFF
6:	TFT
7:	TTF
8:	TTT

Cache Things to Remember

- Caches are NOT mandatory:
  - Processor performs arithmetic
  - Memory stores data
  - Caches simply make data transfers go *faster*
- Each level of Memory Hiererarchy subset of next higher level
- Caches speed up due to temporal locality: store data used recently
- Block size > 1 wd spatial locality speedup: Store words next to the ones used recently
- Cache design choices:
  - size of cache: speed v. capacity
  - N-way set assoc: choice of N (direct-mapped, fully-associative just special cases for N)