Computer Architecture

Spring 2016

Lecture 07: Caches II

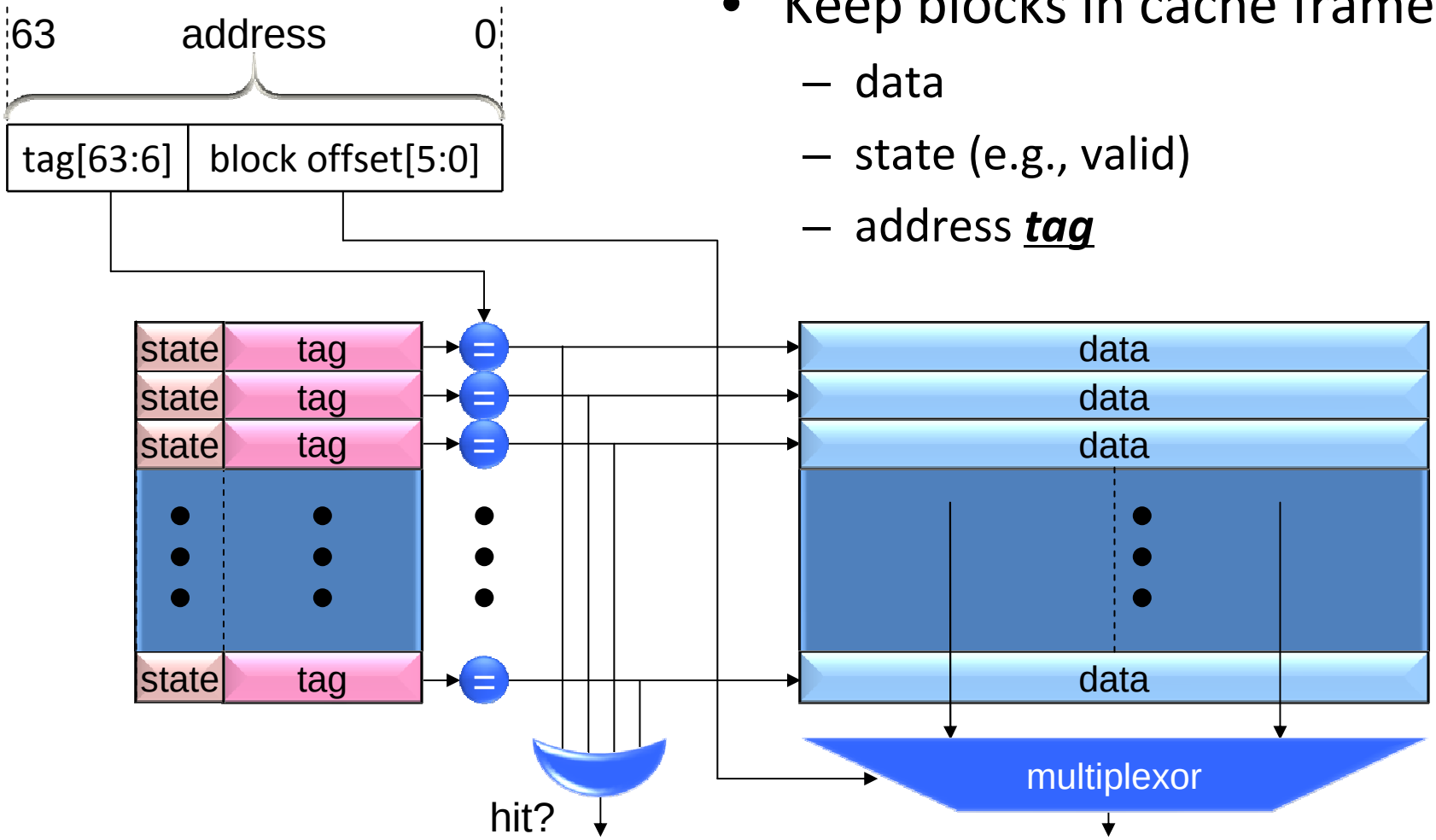
Shuai Wang

Department of Computer Science and Technology

Nanjing University

Fully-Associative Cache

- Keep blocks in cache frames
  - data
  - state (e.g., valid)
  - address ***tag***

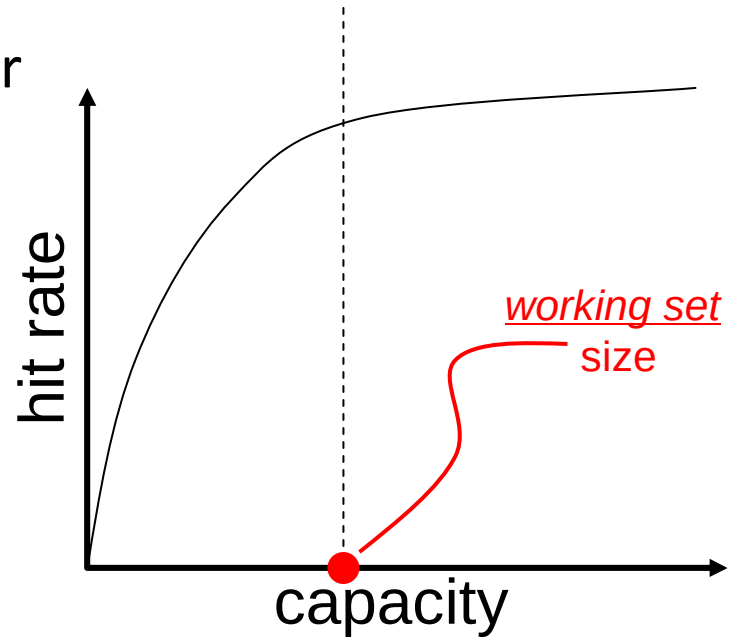


The 3 C's of Cache Misses

- *Compulsory*: Never accessed before
- *Capacity*: Accessed long ago and already replaced
- *Conflict*: Neither compulsory nor capacity (later today)
- *Coherence*: (To appear in multi-core lecture)

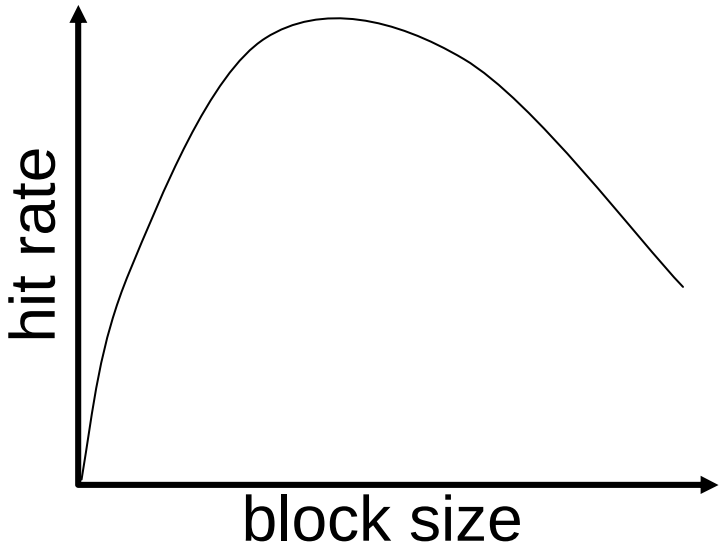
Cache Size

- Cache size is data capacity (don't count tag and state)
  - Bigger can exploit temporal locality better
  - Not always better
- Too large a cache
  - Smaller is faster → bigger is slower
  - Access time may hurt critical path
- Too small a cache
  - Limited temporal locality
  - Useful data constantly replaced

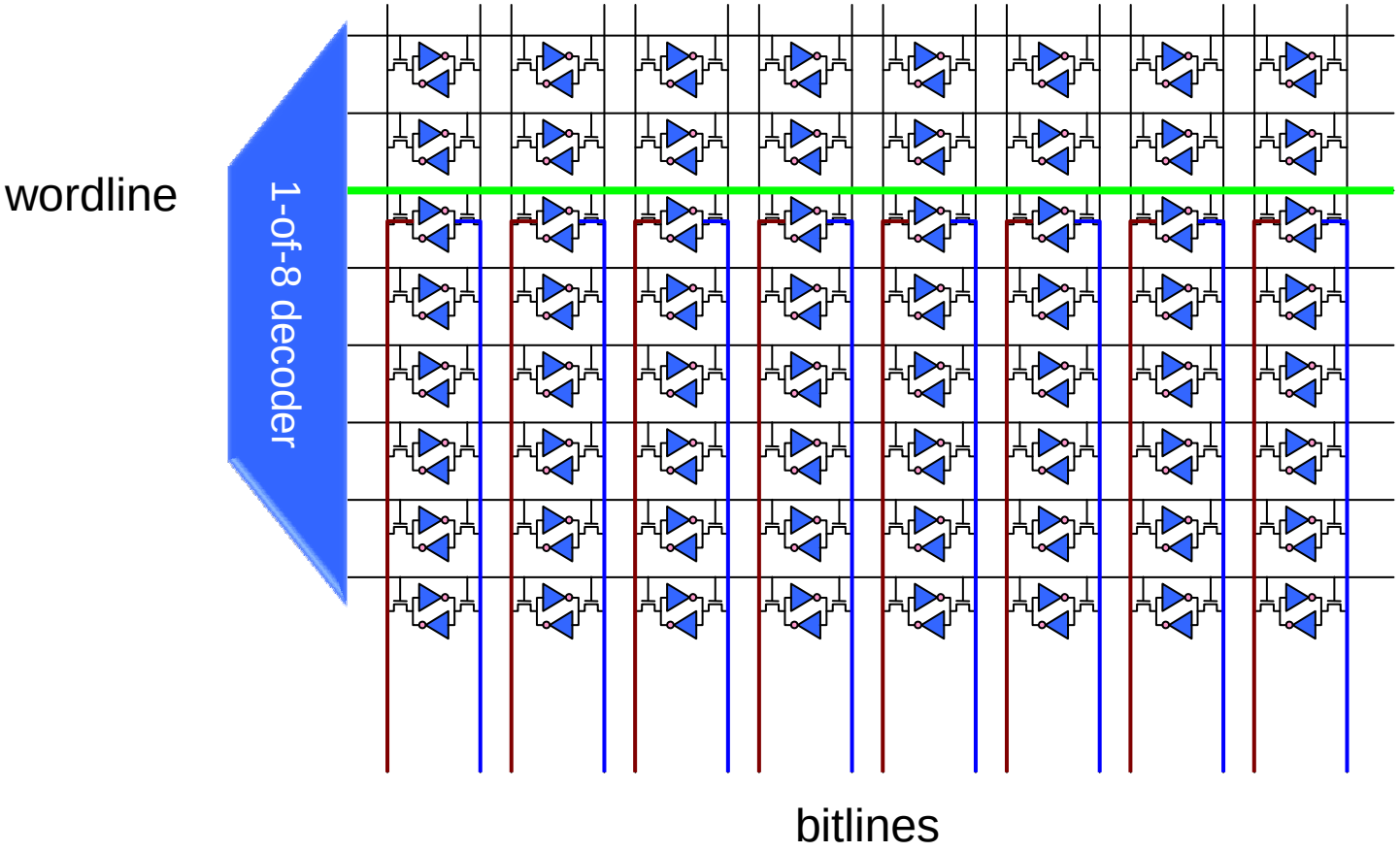


Block Size

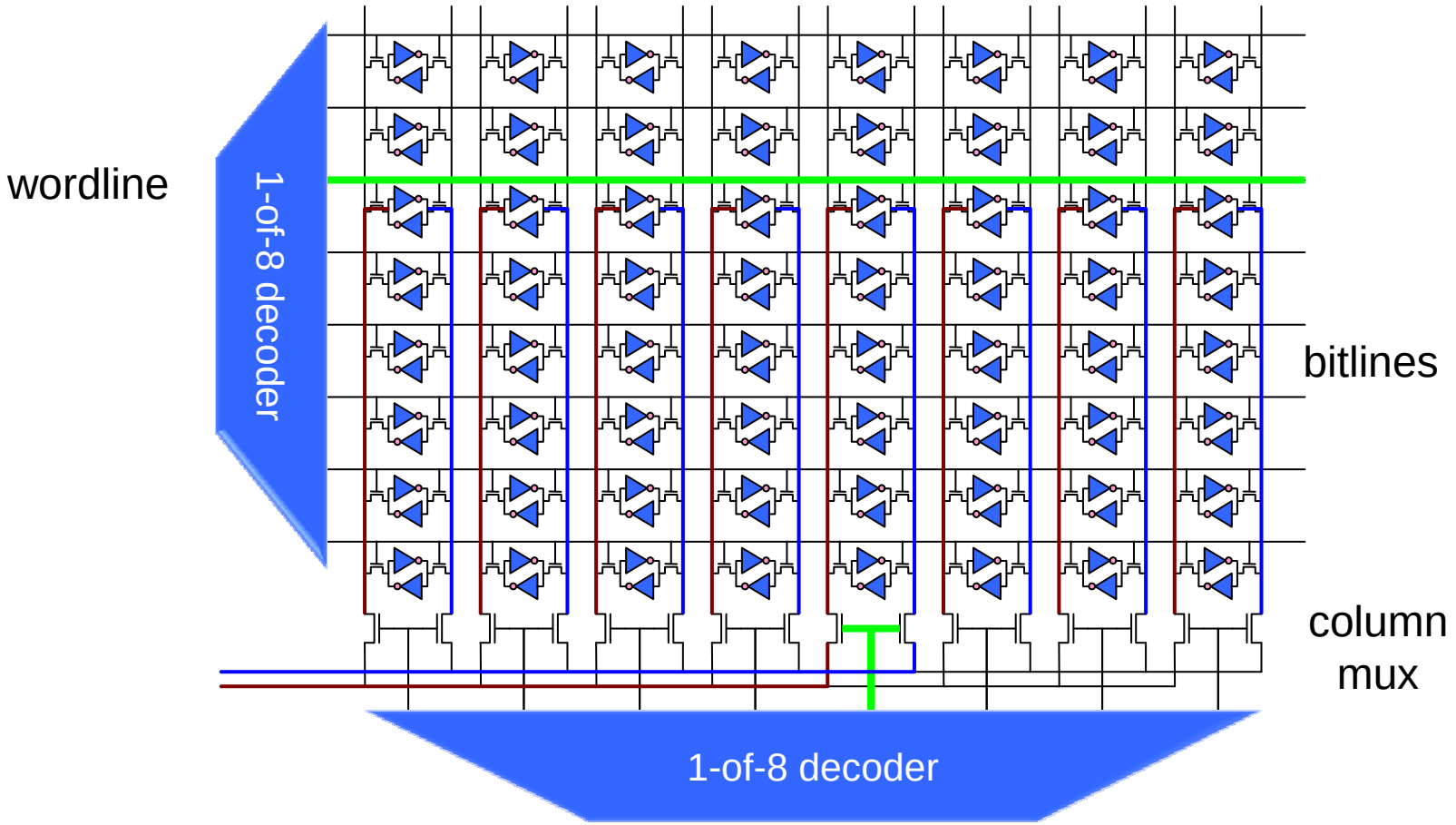
- Block size is the data that is
  - Associated with an address tag
  - Not necessarily the unit of transfer between hierarchies
- Too small a block
  - Don't exploit spatial locality well
  - Excessive tag overhead
- Too large a block
  - Useless data transferred
  - Too few total blocks
    - Useful data frequently replaced



8×8-bit SRAM Array

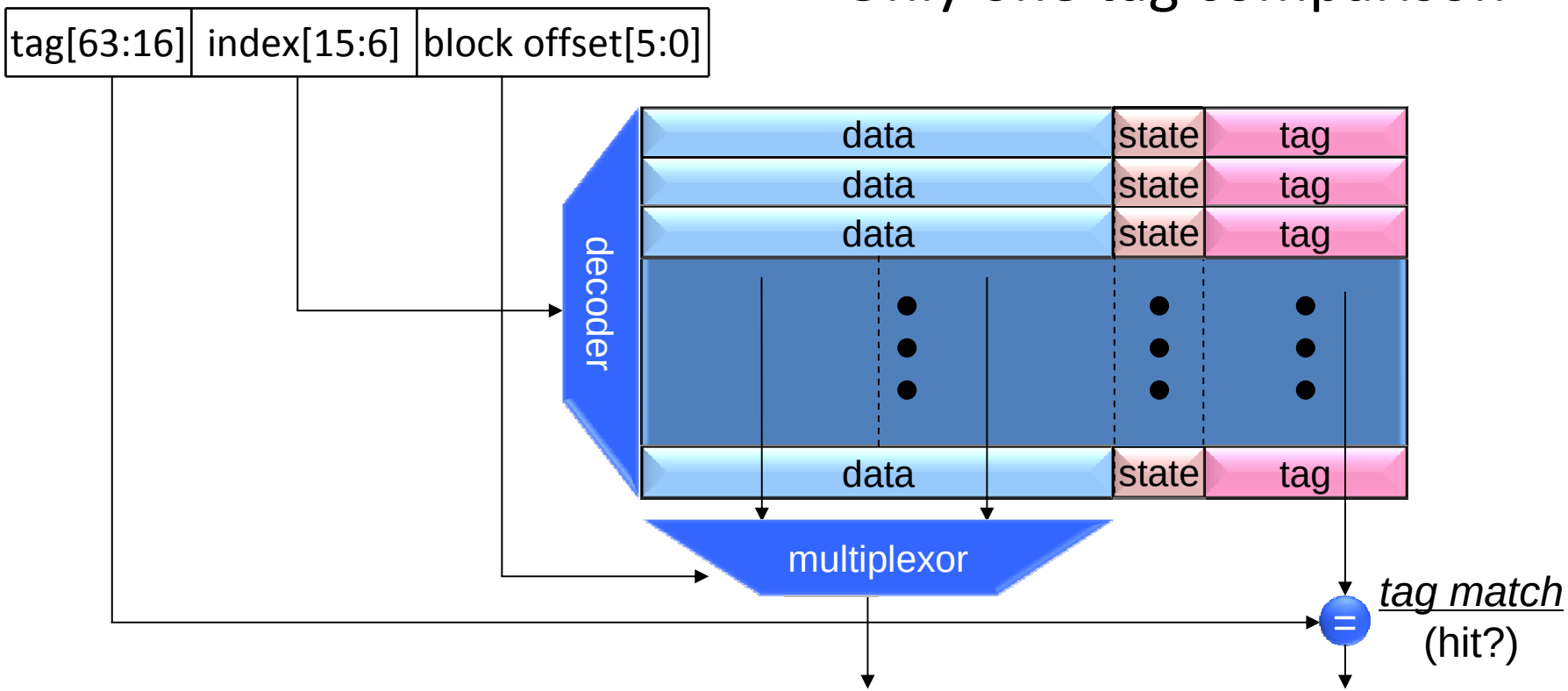


64×1-bit SRAM Array



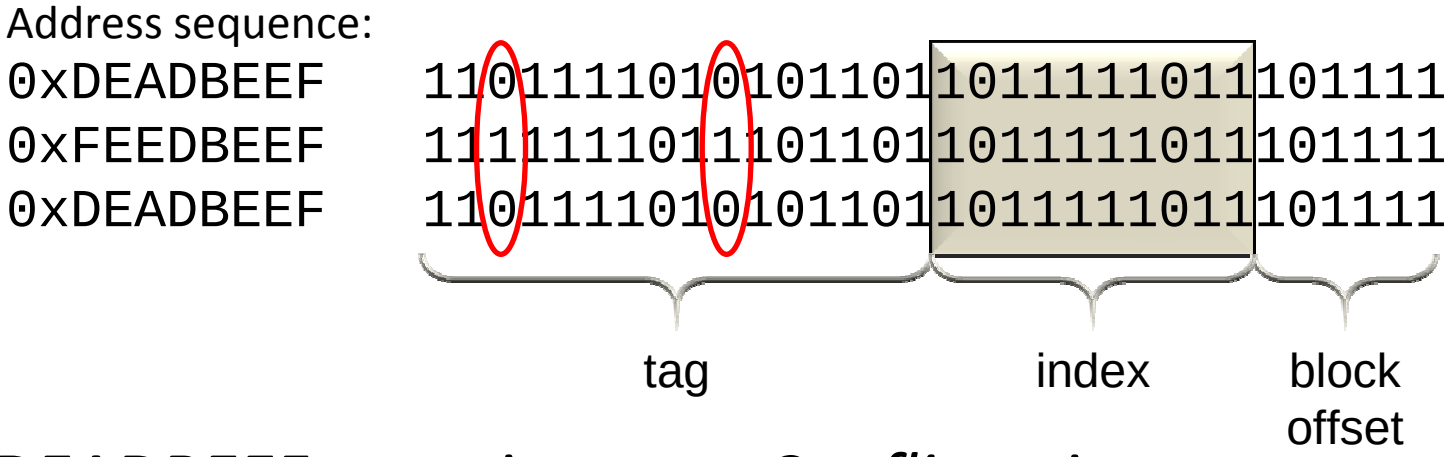
Direct-Mapped Cache

- Use middle bits as index
- Only one tag comparison



Cache Conflicts

- What if two blocks alias on a frame?
  - Same index, but different tags

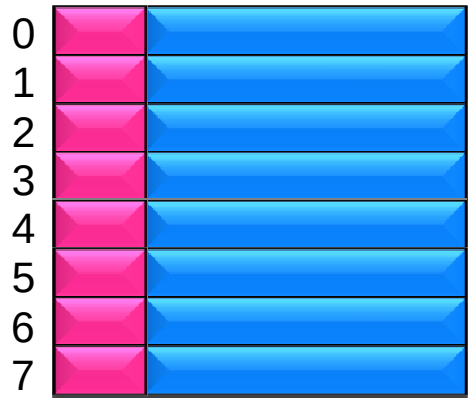


- 0xDEADBEEF experiences a *Conflict* miss
  - Not Compulsory (seen it before)
  - Not Capacity (lots of other indexes available in cache)

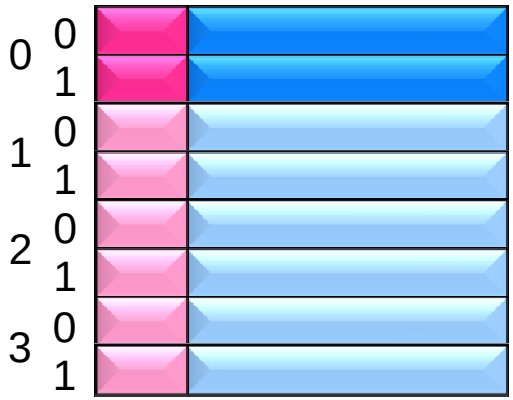
Associativity (1/2)

- Where does block index 12 (b'1100) go?

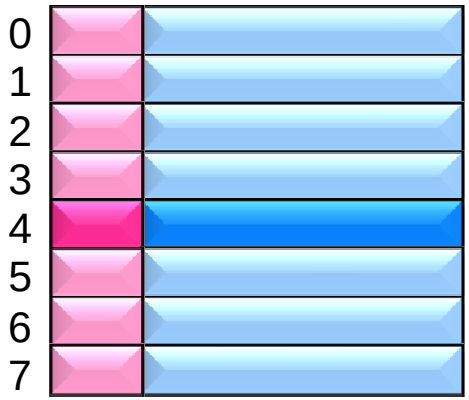
Block



Set/Block



Set



<u>Fully-associative</u>
block goes in any frame

(all frames in 1 set)

<u>Set-associative</u>
block goes in any frame in one set

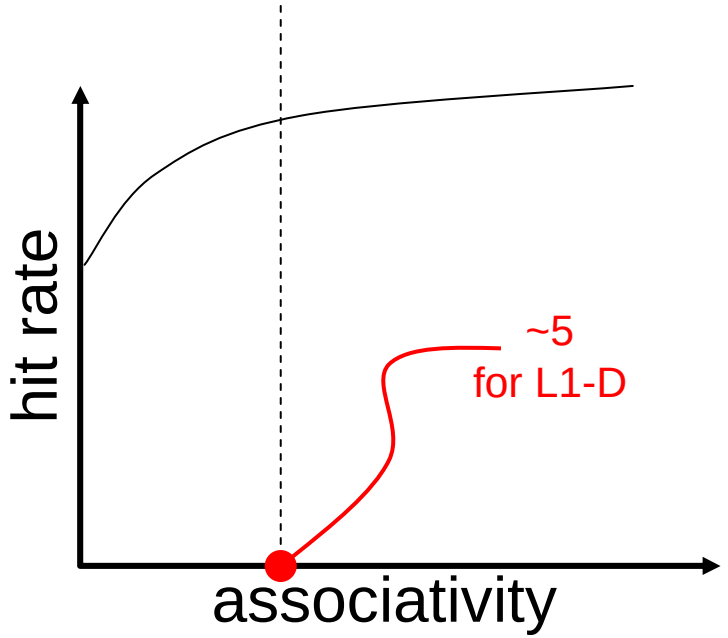
(frames grouped in sets)

<u>Direct-mapped</u>
block goes in exactly one frame

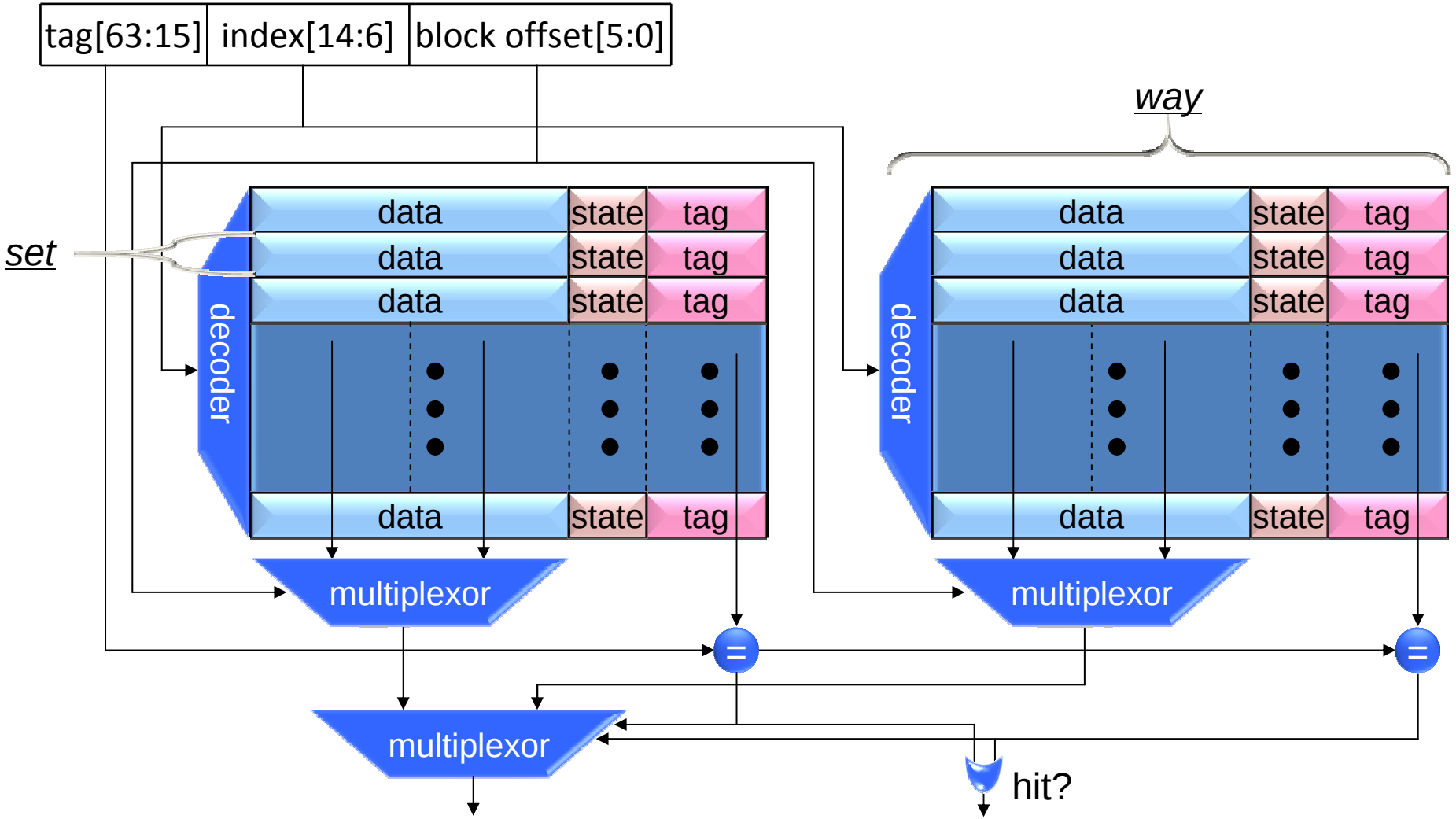
(1 frame per set)

Associativity (2/2)

- Larger associativity
  - lower miss rate (fewer conflicts)
  - higher power consumption
- Smaller associativity
  - lower cost
  - faster hit time



N-Way Set-Associative Cache



Associative Block Replacement

- Which block in a set to replace on a miss?
- Ideal replacement *(Belady's Algorithm)*
  - Replace block accessed farthest in the future
  - Trick question: How do you implement it?
- *Least Recently Used (LRU)*
  - Optimized for temporal locality (expensive for >2-way)
- *Not Most Recently Used (NMRU)*
  - Track MRU, random select among the rest
- *Random*
  - Nearly as good as LRU, sometimes better (when?)

Improve Cache Performance

Average memory access time (AMAT):

$$AMAT = Hit\ Time + Miss\ Rate \times Miss\ Penalty$$

- Reducing cache miss penalty
- Reducing cache miss rate
- Reducing cache miss penalty or miss rate via parallelism
- Reducing hit time

Improve Cache Performance

- Average memory access time (AMAT):

$$AMAT = Hit\ Time + Miss\ Rate \times Miss\ Penalty$$

- Approaches to improve cache performance
  - Reduce the miss penalty
  - Reduce the miss rate
  - Reduce the miss penalty or miss rate via parallelism
  - Reduce the hit time

1st. Reduce Miss Penalty: Multilevel Caches

- **AMAT with a two-level cache:**

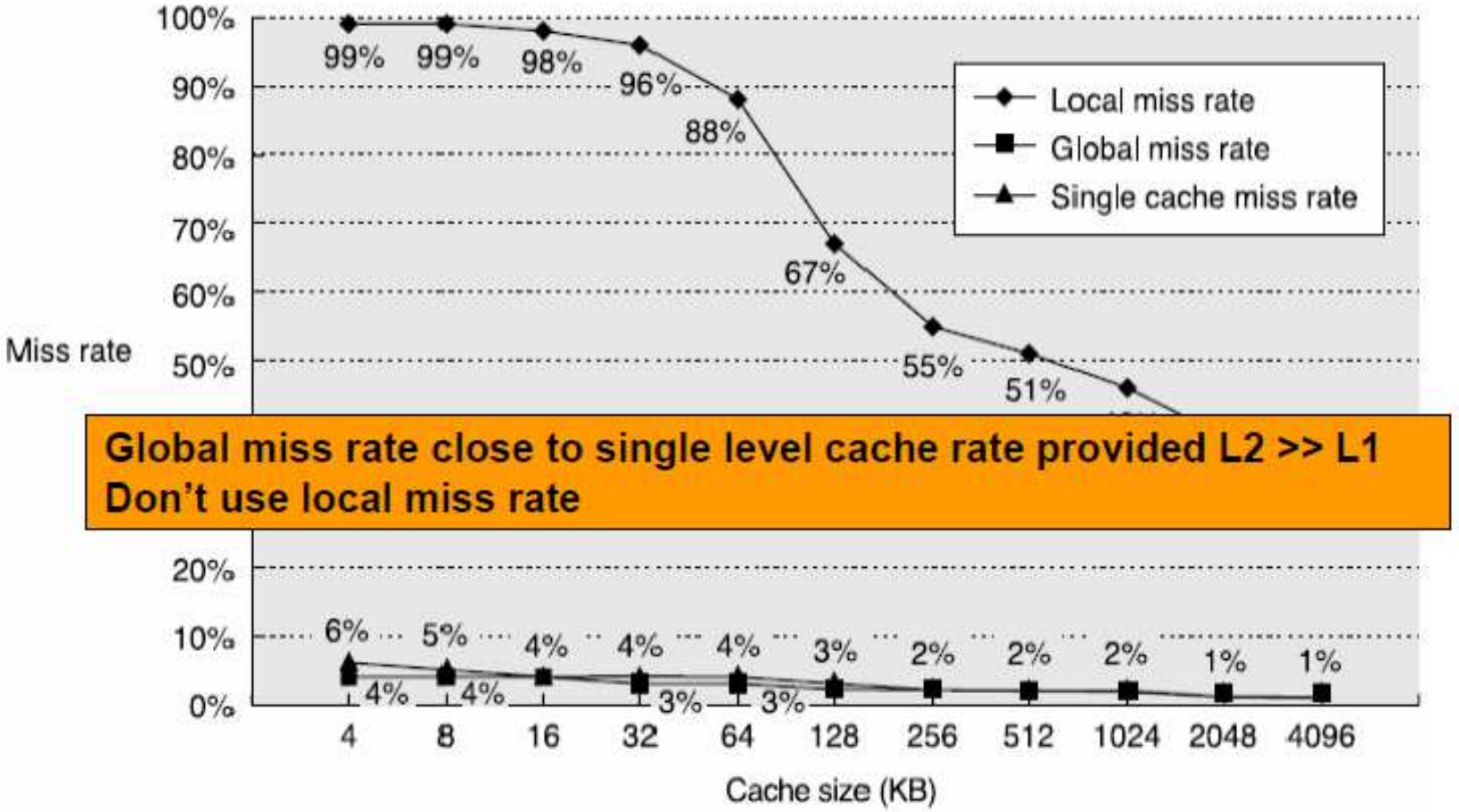
  $AMAT = Hit\ Time_{L1} + Miss\ Rate_{L1} \times Miss\ Penalty_{L1}$

  $Miss\ Penalty_{L1} = Hit\ Time_{L2} + Miss\ Rate_{L2} \times Miss\ Penalty_{L2}$

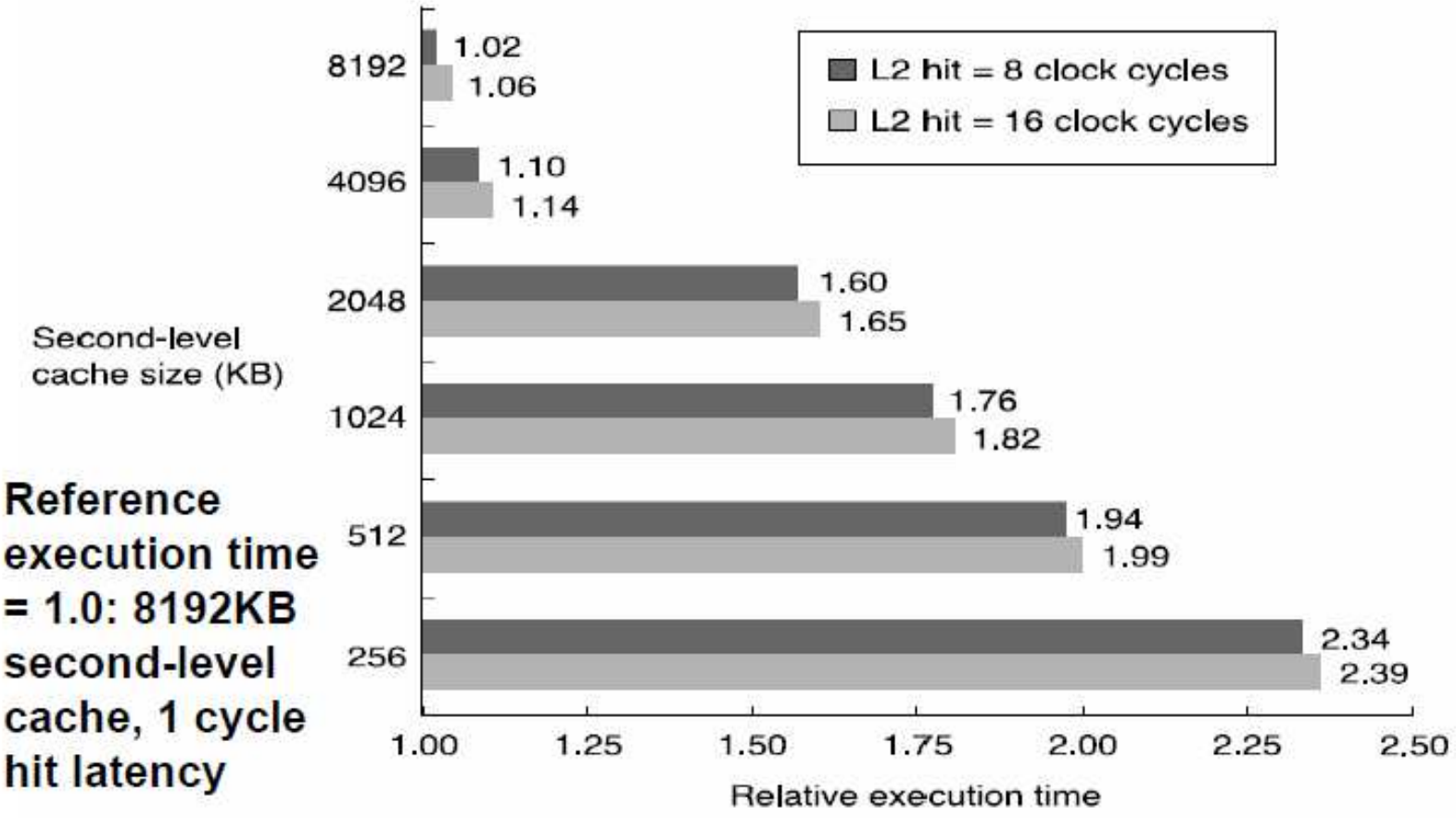
  $AMAT = Hit\ Time_{L1} + Miss\ Rate_{L1} \times (Hit\ Time_{L2} + Miss\ Rate_{L2} + Miss\ Penalty_{L2})$

- **Definitions:**
  - *Local miss rate*— misses in this cache divided by the total number of memory accesses to this cache ($Miss\ rate_{L2}$)
  - *Global miss rate* —misses in this cache divided by the total number of memory accesses generated by the CPU ($Miss\ Rate_{L1} \times Miss\ Rate_{L2}$)
  - Global Miss Rate is what matters

Local and Global Miss Rates



Performance vs. Second Level Cache Size



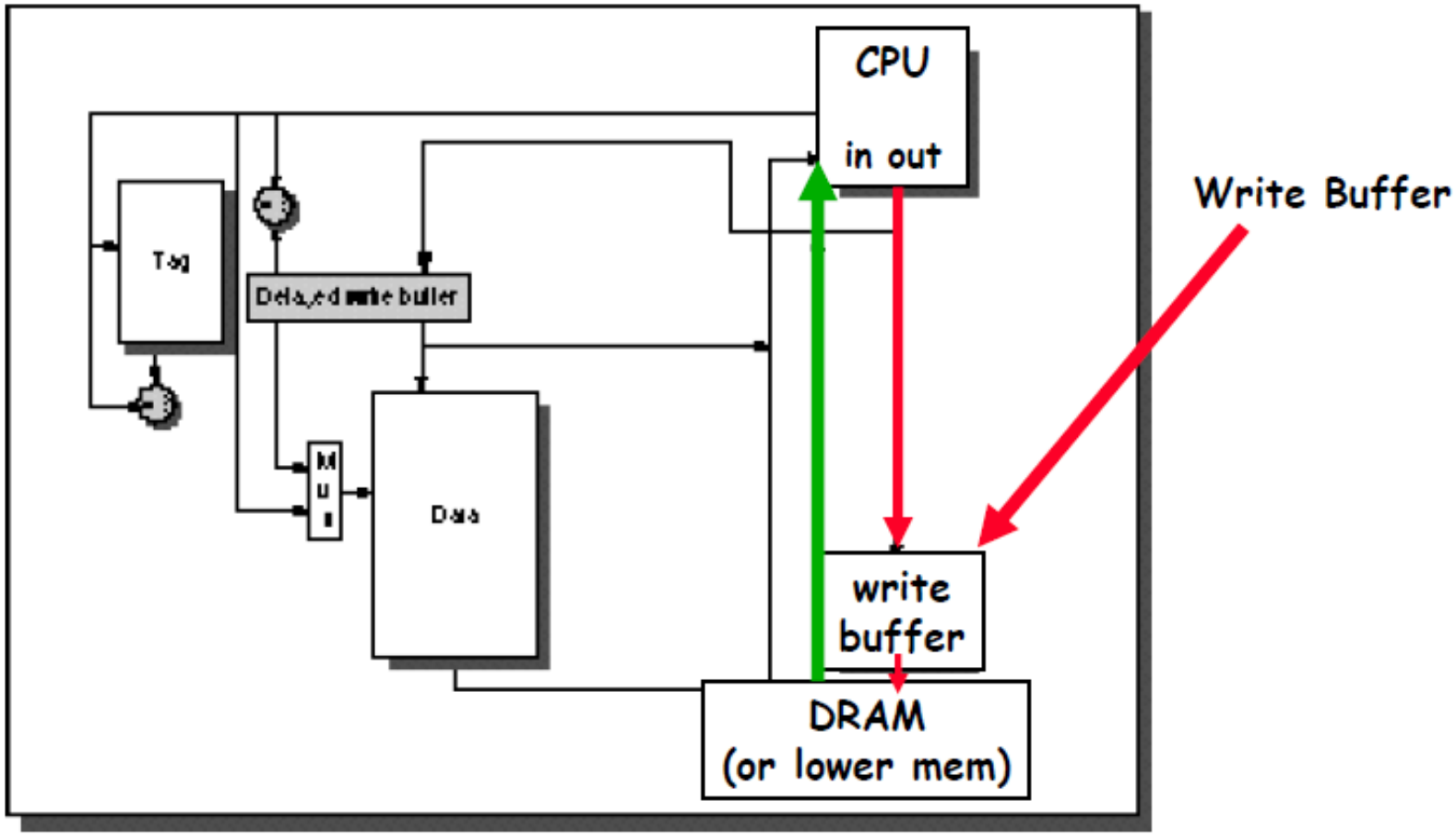
2nd. Reduce Miss Penalty: Critical Word First and Early Restart

- **Don’t wait for full block to be loaded before restarting CPU**
  - Critical Word First—Request the missed word first from memory and send it to the CPU as soon as it arrives; let the CPU continue execution while filling the rest of the words in the block. Also called wrapped fetch and requested word first
  - Early Restart—As soon as the requested word of the block arrives, send it to the CPU and let the CPU continue execution

- **Generally useful only in large blocks**

- **Spatial locality a problem; tend to want next sequential word, so not clear if benefit by early restart**

3rd. Reduce Miss Penalty: Giving Priority to Read Misses over Writes



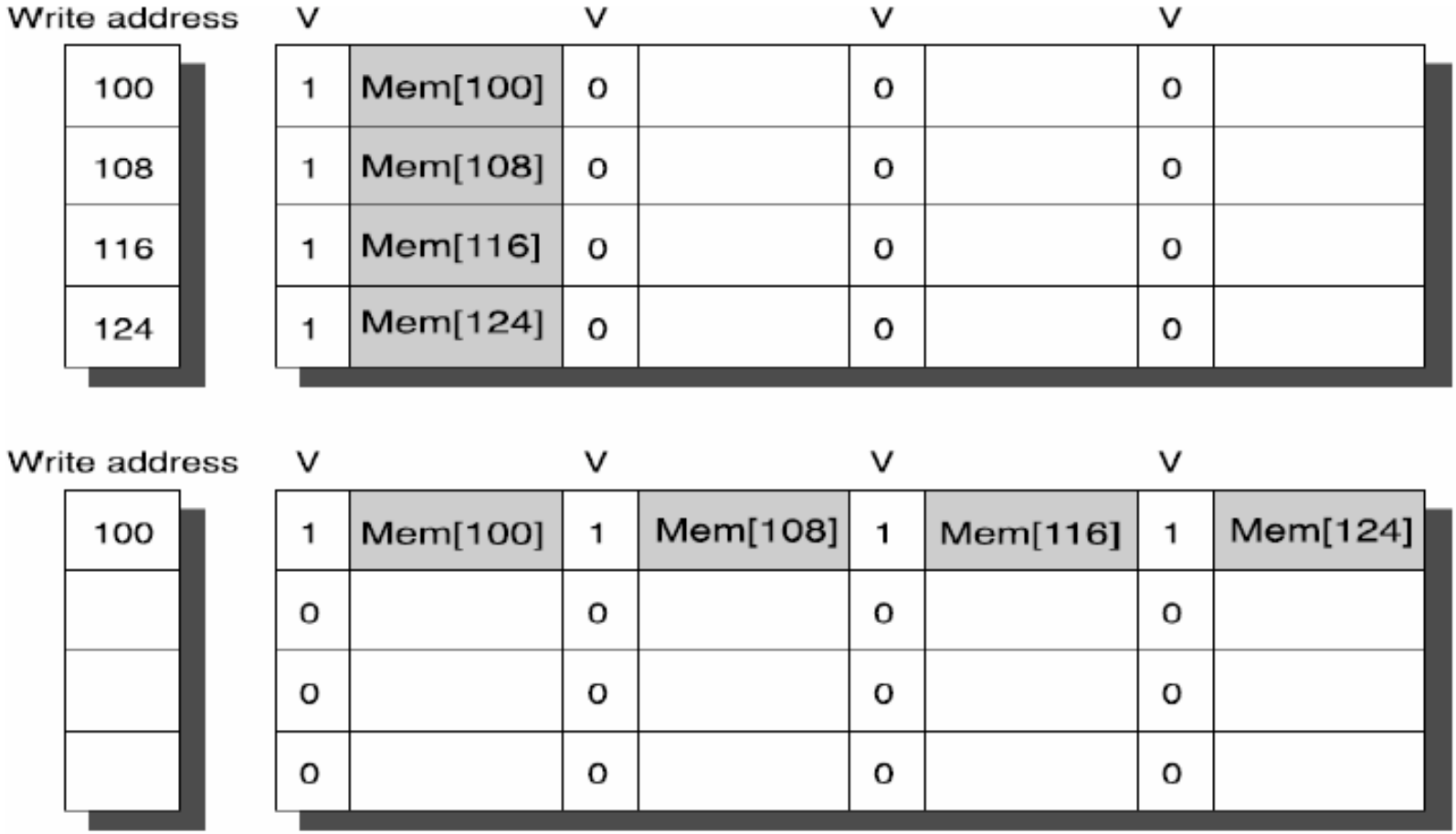
3rd. Reduce Miss Penalty: Giving Priority to Read Misses over Writes

- **Write-through with write buffers offer RAW conflicts with main memory reads on cache misses**
  - If simply wait for write buffer to empty, might increase read miss penalty (old MIPS 1000 by 50%)
  - Check write buffer contents before read;

    if no conflicts, let the memory access continue

- **Write-back also want buffer to hold misplaced blocks**
  - Read miss replacing dirty block
  - Normal: Write dirty block to memory, and then do the read
  - Instead copy the dirty block to a write buffer, then do the read, and then do the write
  - CPU stall less since restarts as soon as do read

4th. Reduce Miss Penalty: Merging Write Buffer

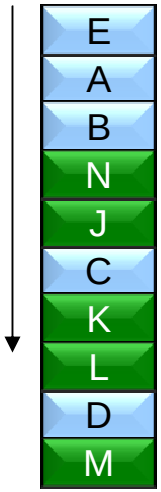


5th. Reduce Miss Penalty: Victim Caches

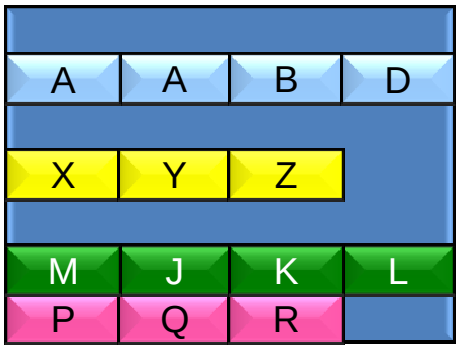
- To hold recently discarded data in case it is needed again
- Add a small fully associative victim cache between a cache and its refill path
- Victim cache contains only blocks discarded from a cache miss
- Victim cache is checked during a cache miss before going to the lower-level memory
- If the data is found in victim cache, victim block and cache block are swapped
- Example: AMD Athlon has a victim cache with 8 entries

Victim Cache

Access Sequence:

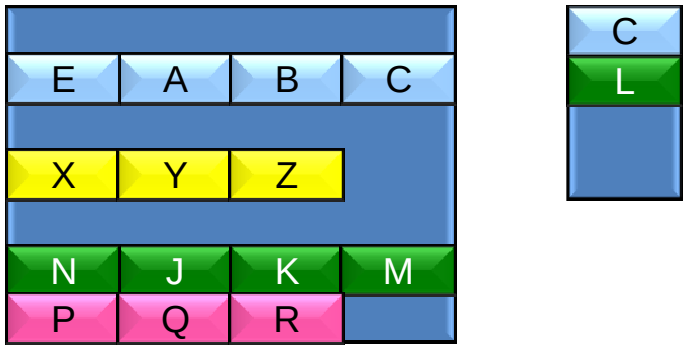


4-way Set-Associative L1 Cache



Every access is a miss! ABCDE and JKLMN do not "fit" in a 4-way set associative cache

4-way Set-Associative L1 Cache + Fully-Associative Victim Cache



Victim cache provides a "fifth way" so long as only four sets overflow into it at the same time

Can even provide 6th or 7th … ways

Reducing Miss Penalty Summary

- Four techniques
  - Read priority over write on miss
  - Early Restart and Critical Word First on miss
  - Non-blocking Caches (Hit under Miss, Miss under Miss)
  - Second Level Cache

- Can be applied recursively to Multilevel Caches
  - Danger is that time to DRAM will grow with multiple levels in between
  - First attempts at L2 caches can make things worse, since increased worst case is worse