EECE 321: Computer Organization

Mohammad M. Mansour
Dept. of Electrical and Compute Engineering
American University of Beirut

Lecture 32: Cache Performance

Cache Performance

- Cache performance is proportional to *miss rate* x *miss penalty*.
- Focus on techniques that reduce both factors:
  1. **New block placement policies that reduce miss rate**
  2. **Multi-level caches to reduce miss penalty**

- Memory stall clock cycles = Memory Inst. Per program x Miss Rate x Miss Penalty.

- Example: Impact of cache performance on machine performance
  - Assume I-cache miss rate for GCC is 2%
  - D-cache miss rate is 4%
  - Assume CPI on a perfect cache is 2
  - 36% of instruction in GCC are memory instructions
  - Assume miss penalty is 40 clock cycles
  - Compute actual CPI including cache misses
- Answer: $CPI_{misses} = 2 + (2\% \times 40) + (36\% \times 4\% \times 40) = 2 + 0.8 + 0.56 = 3.36$
  - Percentage cycles on misses is 1.36/3.36 = 41%

Cache Performance

- What happens to CPI if the processor is made faster, by doubling its clock rate?
  - Main memory speed is unlikely to change, so miss penalty becomes 80 new “CPU” clock cycles
  - $CPI_{new} = 2 + (2\% \times 80) + (36\% \times 4\% \times 80) = 4.75$
  - % cycles on misses is $2.75/4.75 \approx 58\%$

- How would cache misses impact a processor’s performance if it is made faster by lowering its CPI from 2 to 1 (without stalls)?
  - $CPI_{misses} = 1 + (2\% \times 40) + (36\% \times 4\% \times 40) = 2.36$
  - % cycles on misses is $1.36/2.36 \approx 58\%$

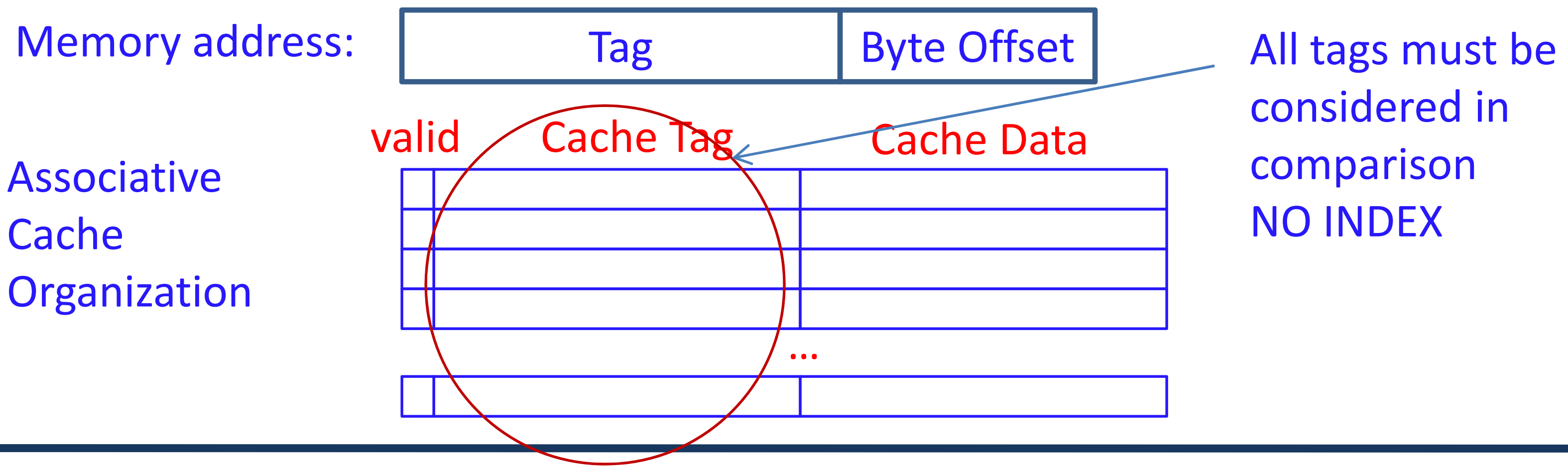
- Conclusion: If a machine improves both clock rate and CPI, the more pronounced the impact of stall cycles on machine performance becomes.

Types of Cache Misses: The "Three Cs" Model

- "Three Cs" Model of Misses:
  - Compulsory (or Cold) Misses, Conflict Misses, Capacity Misses
- 1st C: Compulsory Misses (aka 'Cold Start 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
- 2nd C: Conflict Misses
  - Miss that occurs because 2 distinct memory addresses map to the same cache location
  - 2 blocks (which happen to map to the same location) can keep overwriting each other
  - Big problem in direct-mapped caches
- 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
- 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?
    - **Associative Caches**

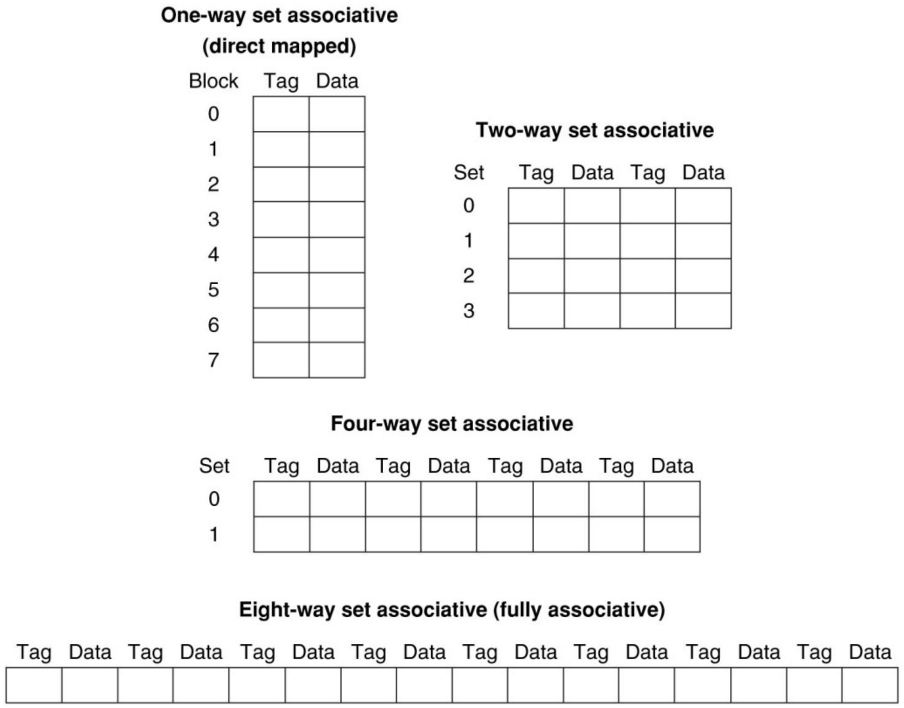
Fully Associative Cache

- Idea: Any block can go anywhere in the cache
- What about the index field in the address? *Answer: It does not exist anymore*
- Memory address fields:
  - Tag: same as before
  - Offset: same as before
  - Index: non-existent
- Benefit of Fully Associative Cache
  - No Conflict Misses (since data can go anywhere)
- Drawbacks of Fully Associative 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


Memory address:
Tag
Byte Offset
All tags must be considered in comparison
NO INDEX
valid
Cache Tag
Cache Data
Associative Cache Organization
...

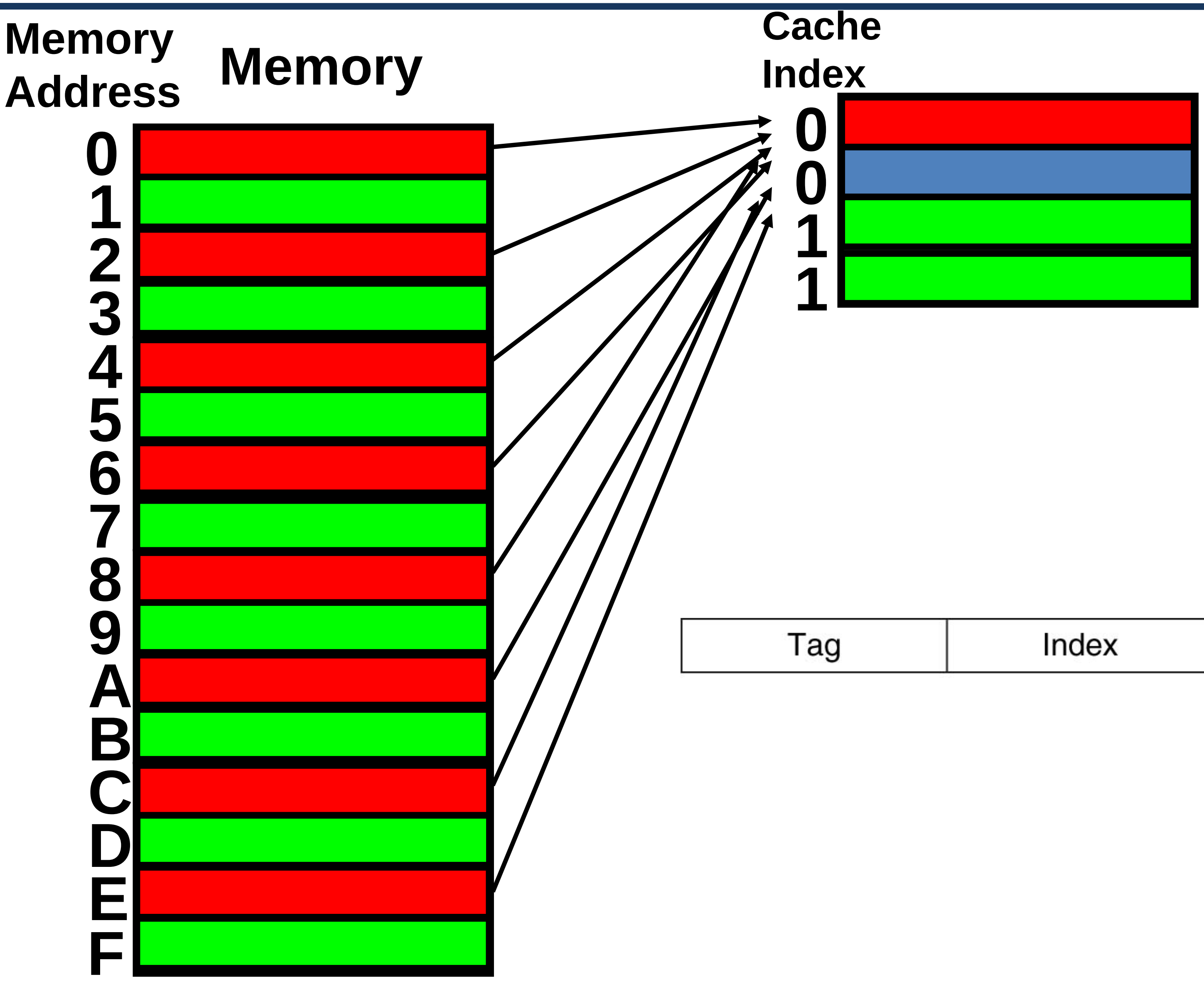
Reducing Miss Rate by a More Flexible Block Placement Policy

- N-way Set-Associative Cache:
  - Divide cache into sets
  - A set contains N-blocks instead of one
  - A block from memory maps to a unique set in cache given by the index field, but can be placed in any block location in that set.
- Mapping: (Block Number) modulo (Number of sets in cache)
- A direct-mapped cache is a 1-way set associative cache.


One-way set associative (direct mapped)
Block Tag Data
0 1 2 3 4 5 6 7
Two-way set associative
Set Tag Data Tag Data
0 1 2 3
Four-way set associative
Set Tag Data Tag Data Tag Data Tag Data
0 1
Eight-way set associative (fully associative)
Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data Tag Data

Index field is now a "set index"

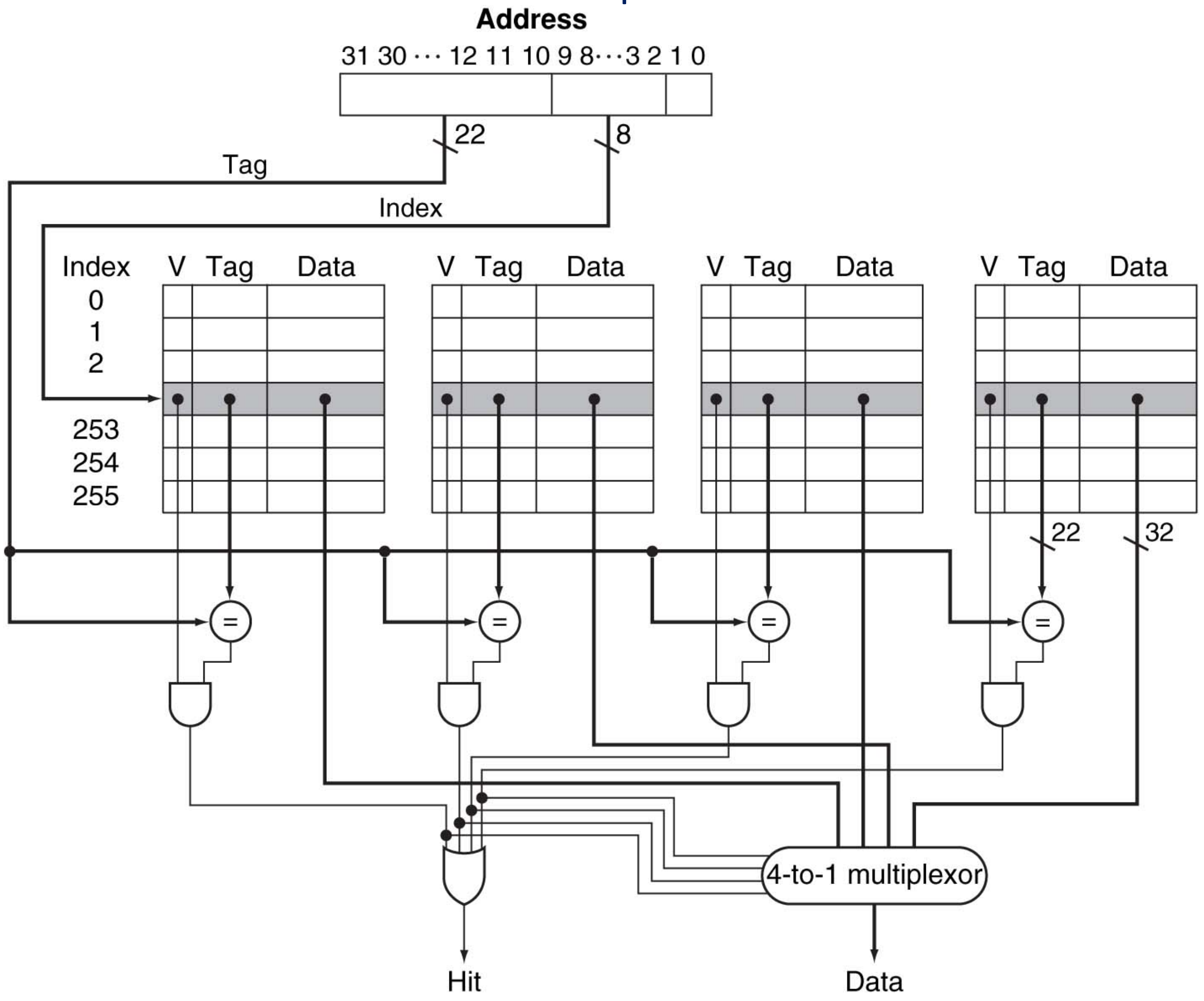
Example: A 2-Way Set Associative Cache



Tag	Index	Block offset

Locating a Block in a 4-Way Set-Associative Cache

- Tags of all blocks in a set are searched in parallel.



Block Replacement Policy

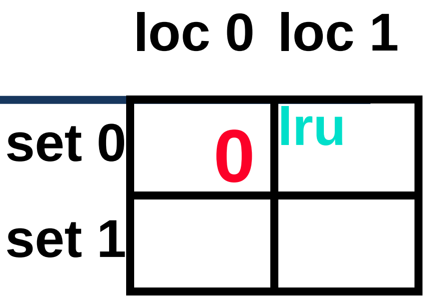
- If a cache block can map to any of the N-blocks in an N-way set associative cache, which one to choose to replace?
- In a direct-mapped cache, there isn't much of a choice; only one possibility.
- The most commonly used scheme is the least recently used (LRU) block replacement policy.
- In an LRU policy, the block that has not been used for the longest time is replaced.
  - Tracking the use of the blocks is implemented by adding extra bits to the blocks to record history of block access.
  - As associativity increases, LRU policy becomes harder to implement.
- Typically, for large associativity random block replacement is used.
- Example: We have a 2-way set associative cache with a four word total capacity and one word blocks. We perform the following word accesses (ignore bytes for this problem):

  0, 2, 0, 1, 4, 0, 2, 3, 5, 4
- How many hits and how many misses will there be for the LRU block replacement policy?

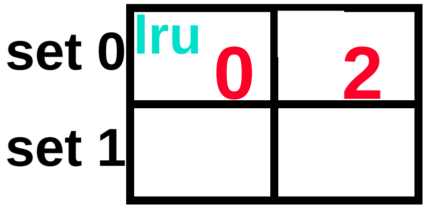
Block Replacement Example: LRU

- Addresses: 0, 2, 0, 1, 4, 0, ...

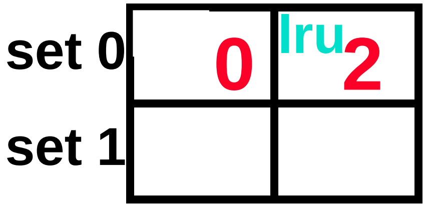
0: miss, bring into set 0 (loc 0)



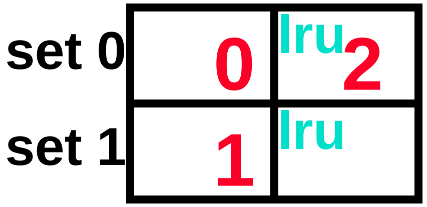
2: miss, bring into set 0 (loc 1)



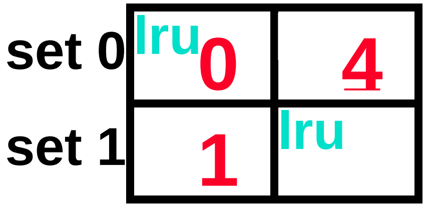
0: hit



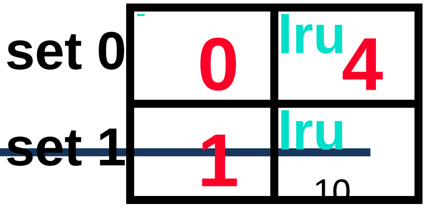
1: miss, bring into set 1 (loc 0)



4: miss, bring into set 0 (loc 1, replace 2)

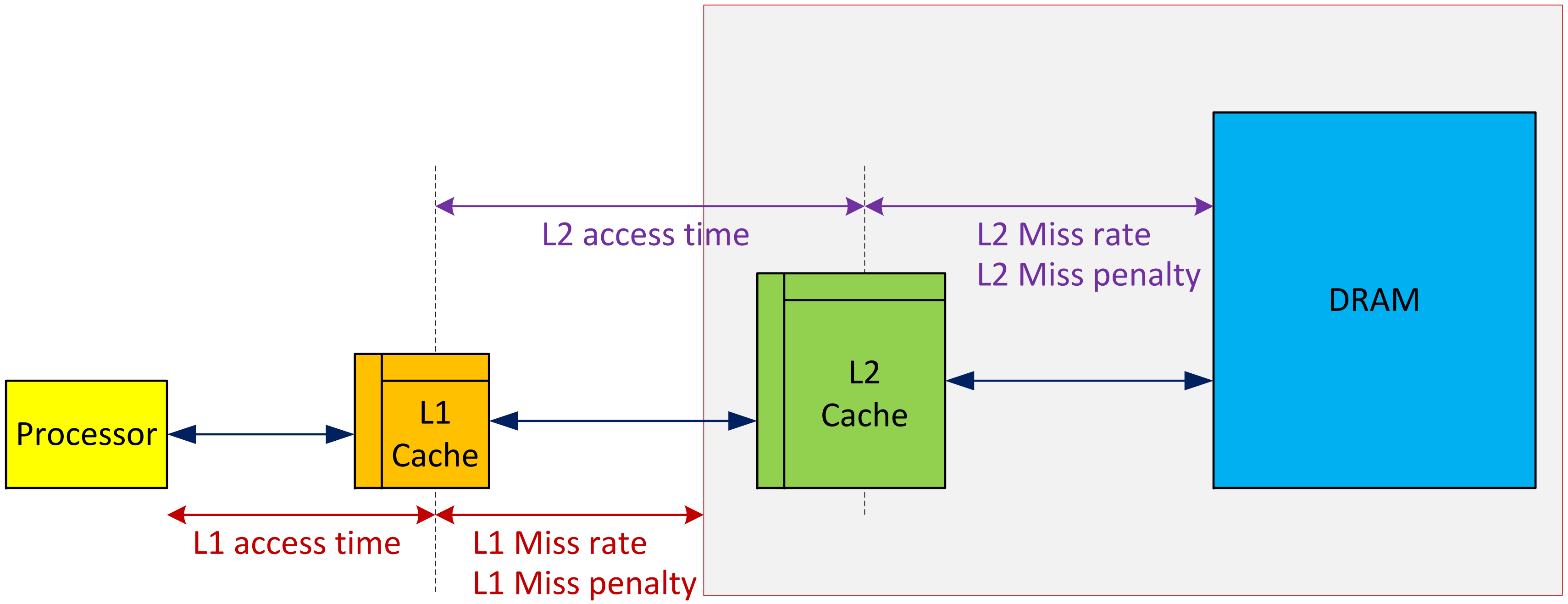


0: hit



Multilevel Caches

- One effective method of reducing miss penalty is to use a multilevel cache.
  - For a two-level cache, if the second-level cache contains the desired data, the miss penalty will be the access time of the second-level cache.
- This access time is typically much less than the access time of main memory.



Avg. Mem Access Time = L1 Access Time + L1 Miss Rate * L1 Miss Penalty

L1 Miss Penalty = L2 Access Time + L2 Miss Rate * L2 Miss Penalty

Miss Rates

- L1 miss rate: Fraction of memory accesses from processor that miss in L1
  - These may hit in L2, or may miss in L2 as well
- L2 miss rate: (looking from the L1 side) Fraction of L1 misses that also miss in L2.
  - Ratio of all misses in L2 cache over the number of accesses to L2.
- Global miss rate: (looking from the processor side) Fraction of all memory accesses from processor that miss in all cache levels.

- Example: Assume L1 miss rate of 5%, L2 miss rate of 20%
  - Out of 1000 memory accesses issued by processor to L1, 50 instructions miss in L1
  - These 50 instructions will be the number of memory accesses that L2 sees
  - Out of these 50, 50x20%=10 instructions miss in L2 and go to Memory
  - Therefore, out of the total 1000 memory accesses, only 10 go to memory (i.e. miss in L1 and L2), resulting in a global miss rate of 10/1000 = 1%.
  - Or directly, global miss rate = L1 miss rate x L2 miss rate