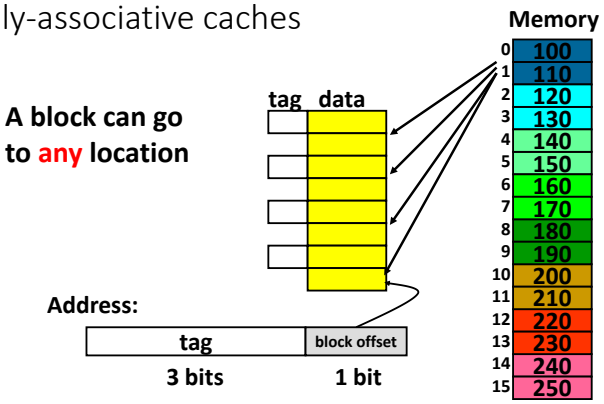
EECS 370

Direct Mapped Caches

Fully-associative caches



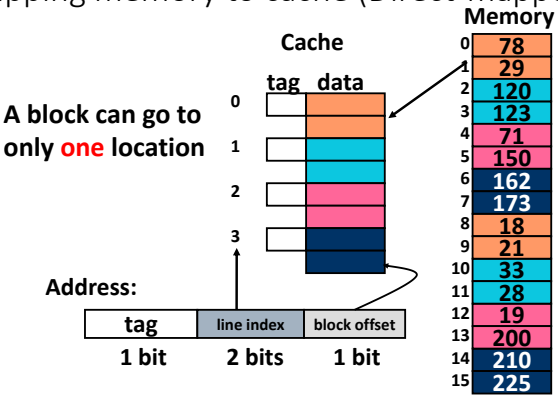
Fully-associative caches

- We designed a fully-associative cache
  - Any memory location can be copied to any cache line.
  - We check every cache tag to determine whether the data is in the cache.
- This approach can be too slow sometimes
  - Parallel tag searches are expensive and can be slow

Direct mapped caches

- We can redesign the cache to eliminate the requirement for parallel tag lookups
  - Direct mapped caches partition memory into as many regions as there are cache lines
  - Each memory region maps to a **single cache line** in which data can be placed
  - Now only **one tag** needs to be checked – the one associated with the region the reference is in

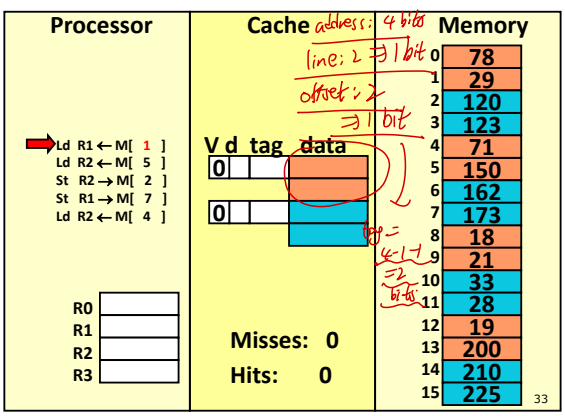
Mapping memory to cache (Direct-mapped)



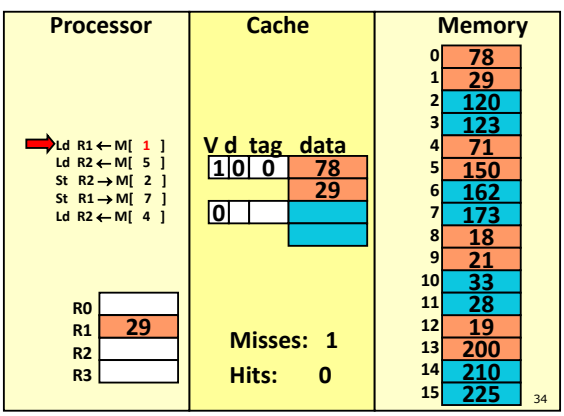
Direct mapped caches

- Two blocks in memory that map to the same index in the cache cannot be present in the cache at the same time
  - One index → one entry
- Can lead to 0% hit rate if more than one block accessed in an interleaved manner map to the same index
  - Assume addresses A and B have the same index bits but different tag bits
  - A, B, A, B, A, B, A, B, …
  - All accesses are conflict misses

Direct-mapped (REF 1)

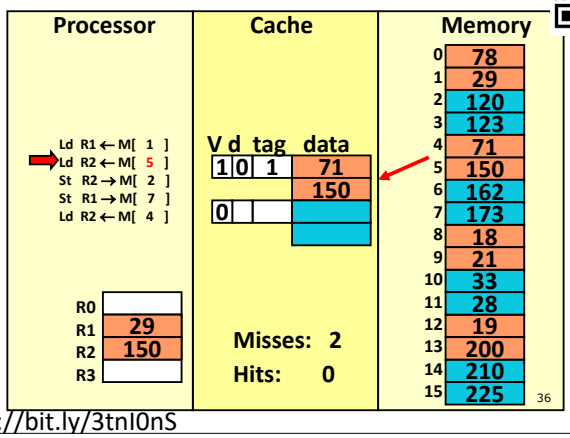


Direct-mapped (REF 1)



Direct-mapped (REF 2)

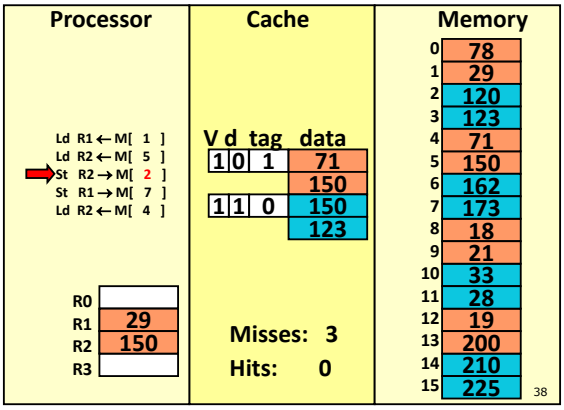
Poll: Complete the last few instructions yourself



36

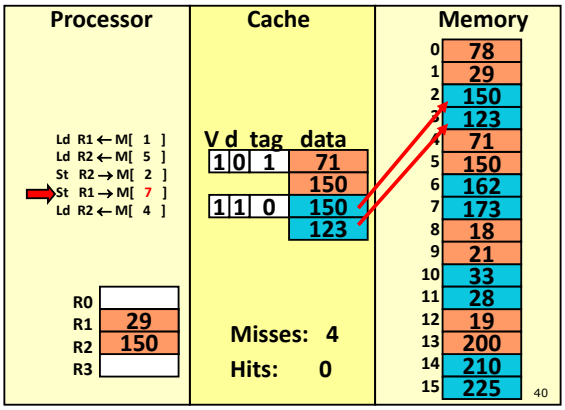
https://bit.ly/3tnI0nS

Direct-mapped (REF 3)



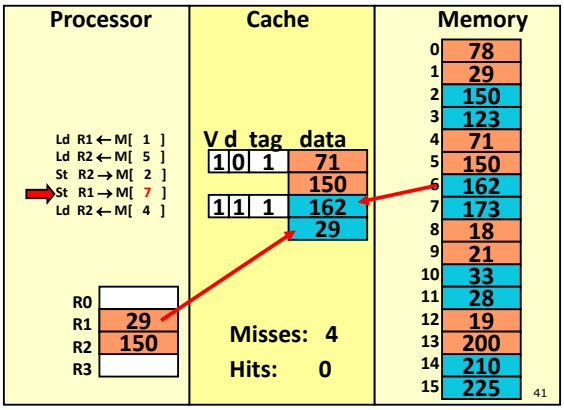
38

Direct-mapped (REF 4)



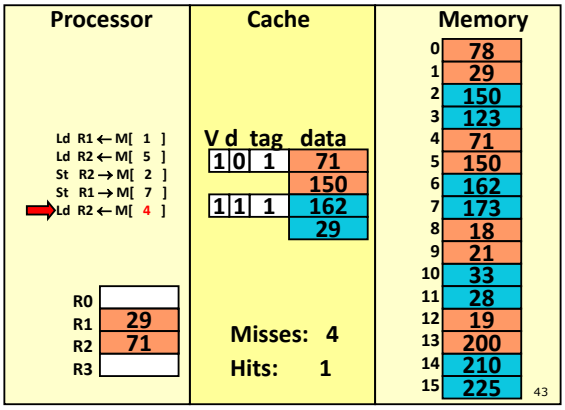
40

Direct-mapped (REF 4)



41

Direct-mapped (REF 5)



43

Next time

- Discuss intermediate between fully-associative and direct mapped caches
  - "Set Associative" caches

Poll: What storage is needed per block?

Class Problem—Storage overhead

- Consider the following cache:
  32-bit memory addresses, byte addressable, 64KB cache
  64B cache block size, write-allocate, write-back, *fully associative*

  This cache will need 512 kilobits for the data area (64 kilobytes times 8 bits per byte). Note that in this context, 1 kilobyte = 1024 bytes (NOT 1000 bytes!) Besides the actual cached data, this cache will need other storage. Consider tags, valid bits, dirty bits, bits to keep track of LRU, and anything else that you think is necessary.
- How many additional bits (not counting the data) will be needed to implement this cache ?

  Tag bits = 32 – log(64) = 26 bits
  #lines = 64KB/64B = 1024
  LRU = log(1024) = 10 bits
  1 valid bit, 1 dirty bit

47

Class Problem—Analyze performance

- Suppose that accessing a cache takes 10ns while accessing main memory in case of cache-miss takes 100ns. What is the average memory access time if the cache hit rate is 97%?

  AMAT = 10 + (1 - 0.97)*100 = 13 ns

- To improve performance, the cache size is increased. It is determined that this will increase the hit rate by 1%, but it will also increase the time for accessing the cache by 2ns. Will this improve the overall average memory access time?

  AMAT = 12 + (1 - 0.98)*100 = 14 ns

49