Chapter 09: Caches

Lesson 04:

Replacement policy

Objective

- Understand the replacement Policy
- Comparisons between write back and write through caches

Replacement policy

Replacement after eviction

- When a line must be evicted from a cache to make room for incoming data, either because the cache is full or because of conflicts for a set, the replacement policy determines which line is evicted

Direct Mapped Replacement

- There is no choice about which line to evict, since the incoming line can only be placed in one location in the cache

Replacement after eviction

- Set-associative and fully associative caches contain multiple lines that could be evicted to make room for the incoming line

Replacement after eviction

- The general goal of the replacement policy is to minimize future cache misses by evicting a line that will not be referenced often in the future
- Should reduce future misses sufficiently without but requiring so much hardware that the capacity of the cache get reduced to accommodate it

Design method for a Replacement Policy

Design chosen for a Replacement Policy

- Takes into account the cost of their replacement policy
- Cost should not be high
- Cost high— when a replacement policy reduces future misses slightly but requires so much hardware that the capacity of the cache must be reduced to accommodate it

Design chosen for Replacement Policy

- Should not be the replacement policy with additional cache misses that result from the reduced capacity may overwhelm the savings from the improved replacement policy

Perfect replacement policy

- Examines the future behavior of the program being run and evict the line that results in the fewest cache misses
- Since computers don't know what their programs will do in the future, replacement policies must guess which line should be evicted based on what the program has done in the past

Least-recently used (LRU) replacement policy

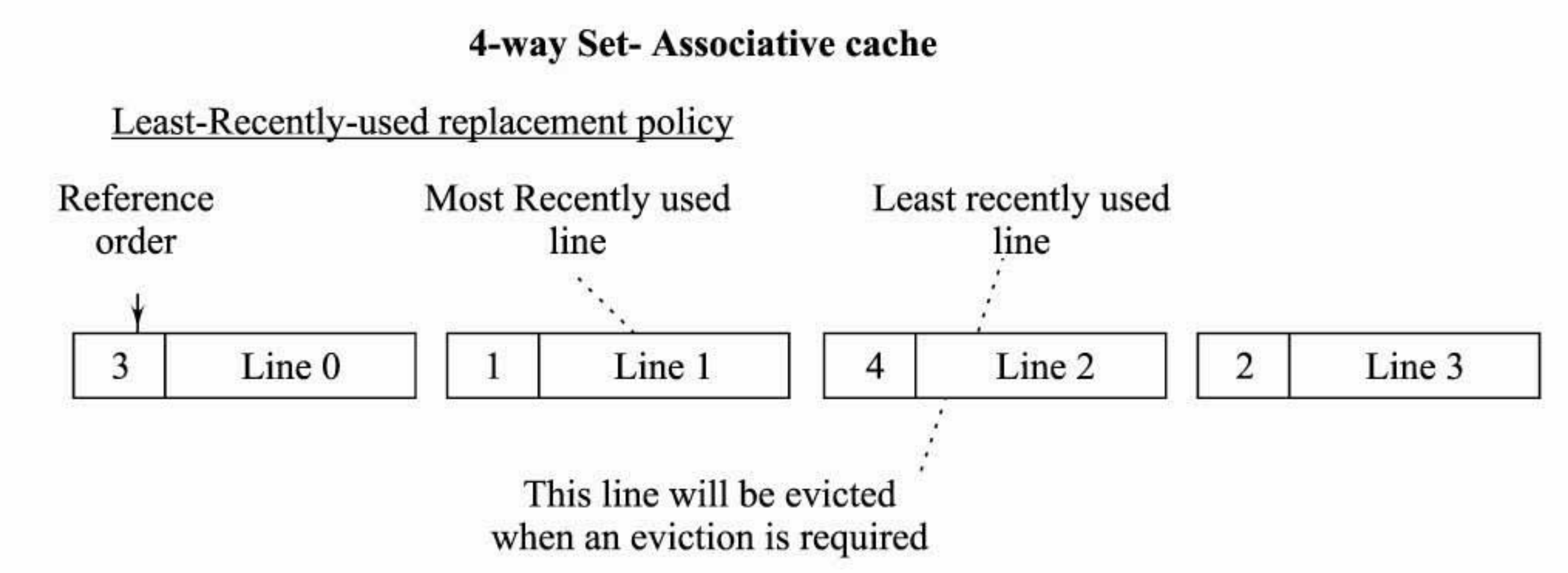
Least-recently used (LRU) replacement policy

- The cache ranks each of the lines in a set according to how recently they have been accessed and evicts the least-recently used line from a set when an eviction is necessary
- Observation is that lines that have not been referenced in the recent past are unlikely to be referenced in the near future

LRU replacement complexity

- Whenever a line in a set is referenced, the information on how recently all of the lines in the set have been referenced must be updated, leading to relatively complicated hardware

LRU replacement policy in four-way set associative Cache


4-way Set- Associative cache

Least-Recently-used replacement policy

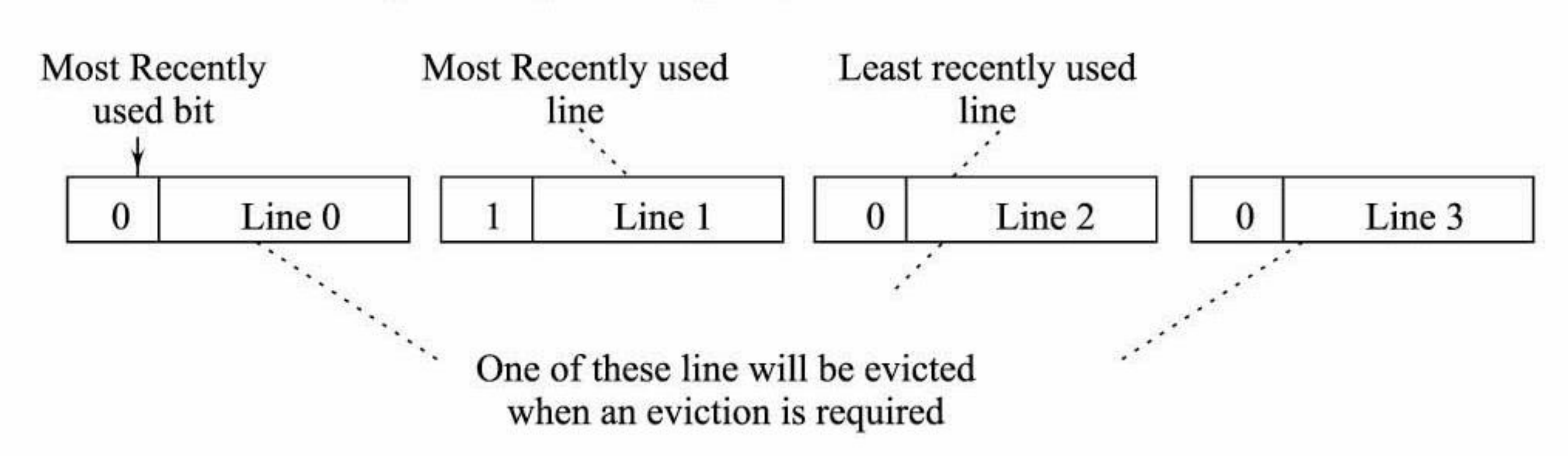
Random replacement policy

Random replacement policy

- A randomly selected line from the appropriate set is evicted to make room for incoming data
- Studies have shown that LRU replacement generally gives slightly higher hit rates than random replacement, but that the differences are very small for caches of reasonable size

Not-most recently used (NMRU) replacement policy

Not-most recently used replacement policy in four-way set associative Cache



Not-most-recently used replacement policy

- The cache keeps track of the line in each set that has been referenced most recently
- Evicts one of the other lines (often selected at random) whenever an eviction is necessary

NMRU in Two-way set-associative caches

- NMRU is equivalent to LRU replacement
- For more-associative caches, this policy ensures that the most-recently used line, which is statistically the most likely line to be referenced in the near future, is kept in the cache at a lower hardware cost than LRU replacement

Summary

We learnt

- Replacement Policy- LRU, NMRU and random
- Takes into account the cost of their replacement policy
- Should reduce future misses sufficiently without but requiring so much hardware that the capacity of the cache get reduced to accommodate it

End of Lesson 04 on

Replacement policy