Lecture 9 slide deck questions

1. What are the three main cache mapping techniques discussed in the slides, and what is one key characteristic of each?

2. In a cache memory system with direct mapping, if w=3 (block size is 8 bytes) and the cache has 16 lines, how many bits would be needed for the tag in a 32-bit address system?

3. What is the "dirty bit" in cache memory and when is it used?

4. Compare write-through and write-back policies in cache memory. What are the main advantages and disadvantages of each?

5. A processor has a two-level cache system. If a memory reference results in a cache miss at both L1 and L2, describe the sequence of events that occurs to handle this miss.

6. What is "thrashing" in the context of cache memory and why does it occur?

7. In modern processor design, why are L1 caches typically split between instructions and data, while higher-level caches are unified?

8. If a cache system uses a 4-way set-associative mapping scheme with a total of 64 cache lines, how many sets are there in the cache?

9. Explain how locality of reference helps improve cache performance, and describe the difference between temporal and spatial locality.

10. In the context of replacement algorithms for cache lines, what are LRU and FIFO, and why might one be preferred over the other?

Answers

1Q: What are the three main cache mapping techniques discussed in the slides, and what is one key characteristic of each?
A: (Slides 42, 68, 73-74)
- Direct Mapping: Simplest to implement but least flexible; each block maps to exactly one cache line
- Associative Mapping: Most flexible but most expensive; blocks can be placed in any cache line
- Set-Associative Mapping: Compromise between the two; blocks are mapped to a specific set but can be placed in any line within that set

2. Q: In a cache memory system with direct mapping, if w=3 (block size is 8 bytes) and the cache has 16 lines, how many bits would be needed for the tag in a 32-bit address system?
A: (Slides 63)
For a 32-bit address with:
- w = 3 bits (for block offset)
- 16 lines = 2^4 lines, so 4 bits for line number
- Tag bits = 32 - (3 + 4) = 25 bits needed for tag

3. Q: What is the "dirty bit" in cache memory and when is it used?
A: (Slide 88)
The dirty bit is used in write-back policy. When a block is first loaded into a cache line, the dirty bit is cleared. When the cache line is modified, the dirty bit is set. The dirty bit indicates whether the block needs to be written back to main memory when it's replaced - if set, the block must be written back; if clear, it can be discarded.

4. Q: Compare write-through and write-back policies in cache memory. What are the main advantages and disadvantages of each?
A: (Slide 88)
Write-through:
- Simpler but more expensive in terms of bus traffic
- All writes immediately update both cache and main memory
Write-back:
- More efficient but more complex
- Uses dirty bit to track modified blocks
- Only writes to main memory when necessary
- Requires special handling for I/O

5. Q: A processor has a two-level cache system. If a memory reference results in a cache miss at both L1 and L2, describe the sequence of events that occurs to handle this miss.
A: (Slides 29, 90-91)
1. CPU checks L1 cache (miss)
2. L1 miss triggers L2 cache check (miss)
3. System must fetch data from main memory
4. Data is loaded into both L2 and L1 caches
5. Data is finally delivered to the CPU

6. Q: What is "thrashing" in the context of cache memory and why does it occur?
A: (Slide 67)

Thrashing occurs in direct mapping when programs frequently reference two or more different blocks that map to the same cache line. This causes the blocks to be repeatedly swapped in and out of the cache, significantly degrading performance.

7. Q: In modern processor design, why are L1 caches typically split between instructions and data, while higher-level caches are unified?
A: (Slide 93)
Split L1 caches allow:
- Parallel access to both instructions and data
- Elimination of cache contention between instruction fetch and data access
Higher levels are unified because:
- It automatically balances the load between instruction and data access
- Simplifies cache management
- Current state of the art favors this hybrid approach

8. Q: If a cache system uses a 4-way set-associative mapping scheme with a total of 64 cache lines, how many sets are there in the cache?
A: (Slide 75)
Using the formula $m = v \times k$, where:
m = total lines (64)
k = ways (4)
v = number of sets
Therefore: $64 = v \times 4$
$v = 16$ sets

9. Q: Explain how locality of reference helps improve cache performance.
A: (Slides 24)
Locality of reference improves cache performance because:
- Programs tend to access memory addresses that cluster together
- Over a short window of time, a processor tends to work within only a few clusters of addresses
- This clustering behavior allows the cache to effectively predict and store the most likely-to-be-used data

10. Q: In the context of replacement algorithms for cache lines, what are LRU and FIFO, and why might one be preferred over the other?
A: (Slide 86)
LRU (Least Recently Used) and FIFO (First-In-First-Out) are both replacement algorithms:
- LRU replaces the line that hasn't been accessed for the longest time
- FIFO replaces the line that was loaded first, regardless of usage

LRU might be preferred because it better reflects actual program behavior and locality of reference, though it's more complex to implement in hardware.