Cache Optimizations

Muhammad Tahir

Lecture 21

Electrical Engineering Department
University of Engineering and Technology Lahore

Contents

Cache Performance Analysis

- Average memory access time (AMAT) when using cache

$$AMAT = (1 - MissRate) \times HitTime + MissRate \times MissTime$$

- Define *MissTime*

$$MissTime = HitTime + MissPenalty$$

- Updated AMAT

$$AMAT = HitTime + (MissRate) \times (MissPenalty)$$

Cache Performance Analysis Cont'd

- **Hit Time:** Time to find the block in the cache and return to the CPU
- **Miss Rate:** Number of misses divided by the total number of memory accesses made by the CPU
- **Miss Penalty:** Number of additional cycles required upon encountering a miss to fetch a block from the next level of memory hierarchy

Cache Performance Analysis Cont'd

- To reduce AMAT, we require
  - **Hit Time** to be low $\sim$ small and fast cache
  - **Miss Rate** to be low $\sim$ large and/or smart cache
  - **Miss Penalty** to be low $\sim$ main memory access time should be reduced

Reducing the Miss Rate

Larger cache block size

- Advantage
  - Reduces compulsory misses by exploiting spatial locality
- Disadvantage
  - Increases miss penalty

Choosing the right block size is a complex trade-off

Reducing the Miss Rate Cont'd

Larger cache size

- Advantages
  - Reduces capacity misses
  - Reduces conflict misses
- Disadvantages
  - Larger hit time
  - Higher cost, area & power consumption

Reducing the Miss Rate Cont'd

Higher associativity

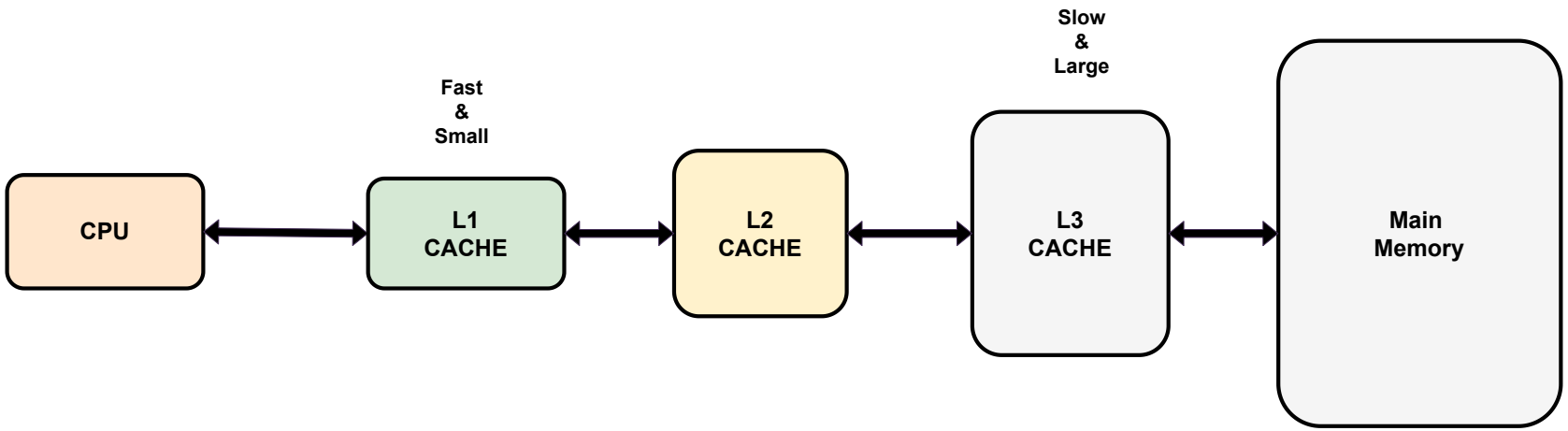
- Advantages
  - Reduces conflict misses
- Disadvantages
  - Increases hit time (due to extra hardware)
  - Complex design

Reducing the Miss Penalty

- Multi-level caches
- Victim Caches
- Critical Word First and Early Restart
- Merging Write Buffer
- Giving Priority to Read Misses over Write misses OR Reducing Read Miss Penalty

Multi-level Caches

- Make cache fast (L1) to keep pace with higher clock rate of the processor
- Make cache large (L2 and L3) to reduce the main memory accesses

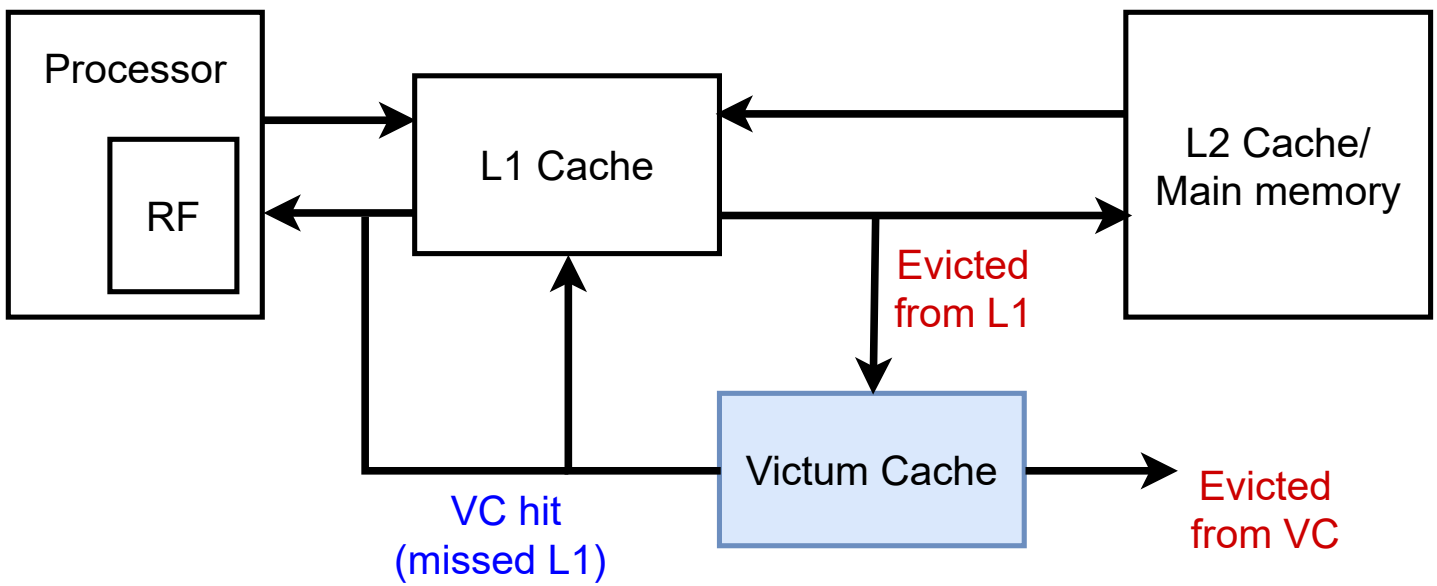


Multi-level Caches Cont'd

- First-level caches (applicable to both instruction and data)
  - Latency is the most critical parameter
  - Smaller with lower associativity
  - Tag and data are accessed simultaneously
- Second-level caches
  - Designed for better tradeoff between hit rate and access latency
  - Larger size with higher associativity
  - Tag and data are accessed sequentially

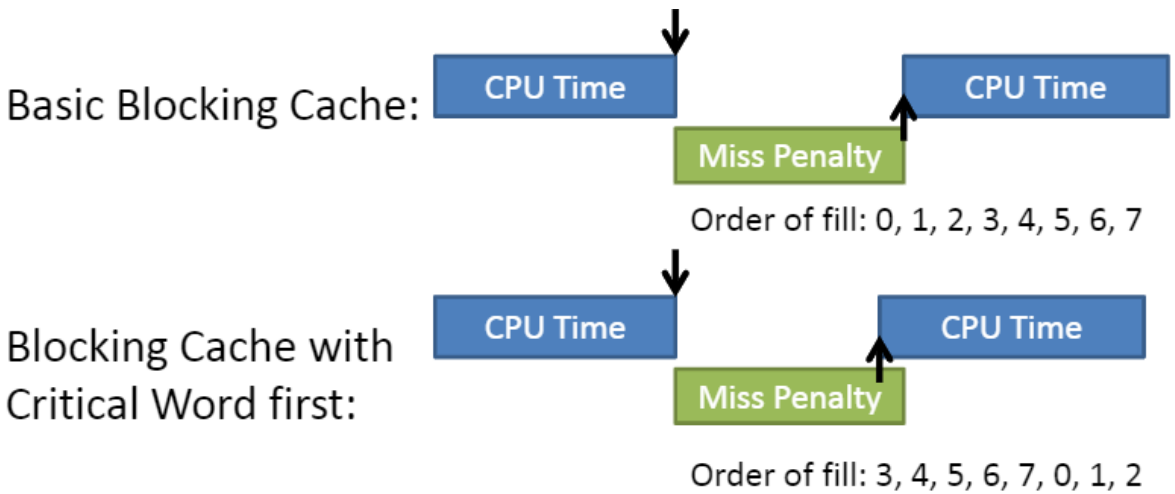
Victim Caches

- Victim cache (VC) is a small associative back up cache added to direct mapped caches
- Holds most recently evicted cache lines
- Leads to fast hit time of direct mapped with reduced conflict misses



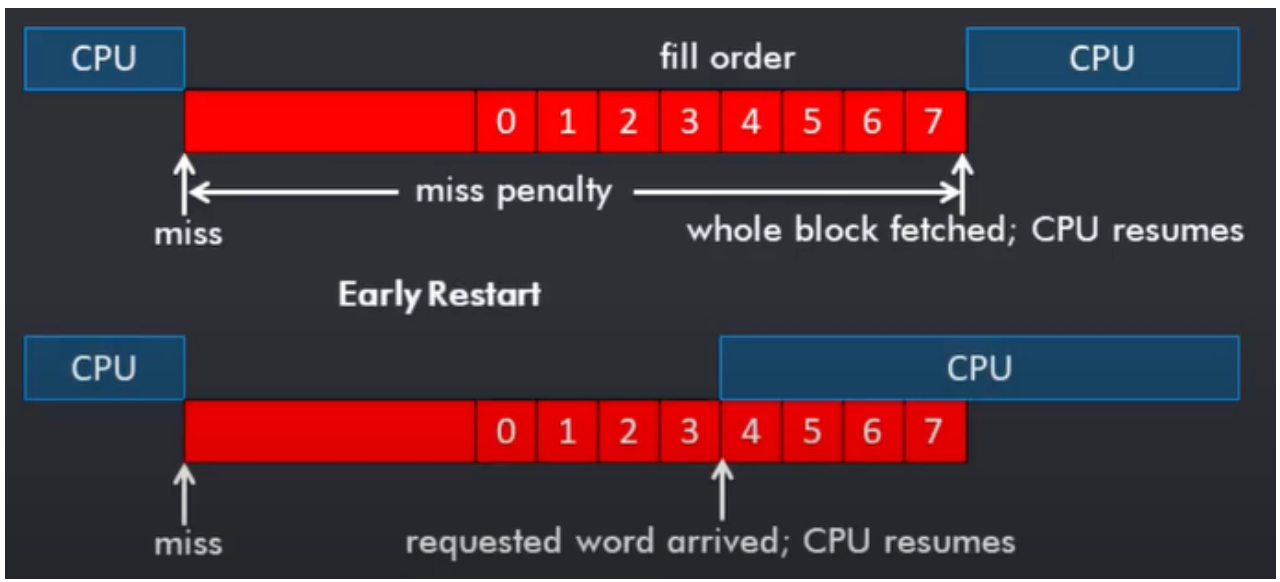
Critical Word First

- Request the required data word first from memory
- Let the processor continue execution while rest of the cache line is being filled



Early Restart

- Data from (main) memory arrives in order
- Processor resumes execution as soon as the requested word arrives during block transfer



Write Buffer

- Write-through caches rely on write buffers
- Write-back caches use a simple buffer when performing block replacement
- Once data and full address are written to the buffer, write is finished from the processor's view point
- While the processor continues, the write buffer writes data to the next level memory in the hierarchy

Merging Write Buffer

- Multi-word writes are usually more efficient
- Need a valid bit per word and requires address checking
- Reduces stalls due to write buffer being full and improves buffer efficiency

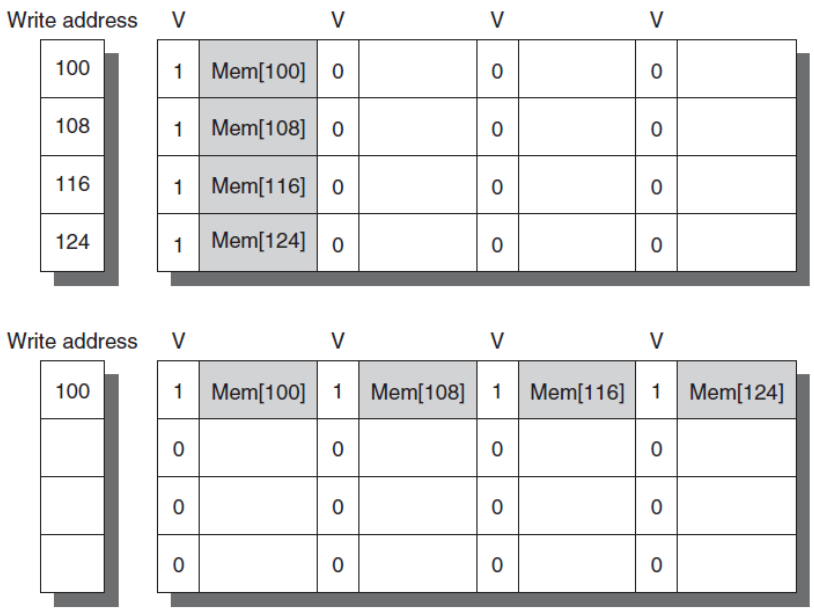


Figure 1: Write buffer merging (Source: Fig. 2.12 [Patterson and Hennessy, 2019]).

Giving Priority to Read Miss over Writes

- Read misses must be handled as soon as possible, as the processor is stalled waiting for data
- Writes can happen in the background
- Must maintain memory order:
  - Load should return value written by most recent store to the same address
  - On a read miss, the write buffer must be checked first

Reducing Hit Time

Small & simple first level cache

- Faster clock & limited power encourage smaller L1 caches
- Lower level of associativity reduces both hit time and power
- Critical timing path in a cache hit is the three-step process
  - Accessing tag memory using index field of the address
  - Comparing the read tag to the address (tag field)
  - Setting the output multiplexer to choose the correct data item
- Simple Case: Use direct mapped cache as they can overlap the tag check with the transmission of the data, effectively reducing hit time.

Reducing Hit Time Cont'd

Pipelining Access and Multi-banked Cache

- Advantages
  - Pipelining L1 allows a higher clock frequency, but at the cost of increased latency
  - More suited for instruction cache due to better performance of branch prediction
  - Multi-banked cache increases the memory throughput (suitable for super-scalar processors)

Reducing Hit Time Cont'd

Reducing write hit time

- Writes take two cycles
- One cycle for tag check and second cycle for data write if hit
- Design data cache that can perform write in one cycle, restore old value if tag does not match
- Pipelined writes: Hold write data in store buffer ahead of cache, write cache data during next store's tag check

Suggested Reading

- Read relevant sections of Chapter 5 of [Patterson and Hennessy, 2021].
- Read Section 2.3 of [Patterson and Hennessy, 2019].

Acknowledgment

- Preparation of this material was partly supported by Lampro Mellon Pakistan.

References

Patterson, D. and Hennessy, J. (2021).

Computer Organization and Design RISC-V Edition: The Hardware Software Interface, 2nd Edition.

Morgan Kaufmann.

Patterson, D. and Hennessy, J. (6th Edition, 2019).

Computer Architecture: A Quantitative Approach.

Morgan Kaufmann.