ELE 455/555
Computer System Engineering

Section 3 – Memory

Class 6 – Optimization

Memory Optimization

Framework

- Common Framework for Memory Hierarchy
  - Caching
    - Used at each level
    - Optimizes access time vs. availability
  - Common Issues
    - Block placement
    - Finding a block
    - Replacement on a miss
    - Write policy

Memory Optimization
Framework

- Common Framework for Memory Hierarchy
  - Block Placement
    - Direct mapped (1-way associative)
      - One choice for placement
    - n-way set associative
      - n choices within a set
    - Fully associative
      - Any location

Scheme name	Number of sets	Blocks per set
Direct mapped	Number of blocks in cache	1
Set associative	$\frac{\text{Number of blocks in the cache}}{\text{Associativity}}$	Associativity (typically 2–16)
Fully associative	1	Number of blocks in the cache

  - Higher associativity reduces miss rate
    - Increases complexity, cost, and access time

Memory Optimization

Framework

- Common Framework for Memory Hierarchy
  - Finding a block

Associativity	Location method	Tag comparisons
Direct mapped	Index	1
n-way set associative	Set index, then search entries within the set	n
Fully associative	Search all entries	#entries
	Full lookup table	0

  - Hardware caches
    - Reduce comparisons to reduce cost
  - Virtual Memory
    - Full table lookup make full associativity feasible
    - → reduced miss rate

Memory Optimization
Framework

- Common Framework for Memory Hierarchy
  - Replacement on a miss
    - Least recently used (LRU)
      - Complex and costly hardware for high associativity
    - Random
      - Close to LRU, easier to implement
  - Virtual memory
    - LRU approximation with hardware support

Memory Optimization
Framework

- Common Framework for Memory Hierarchy
  - Write Policy
    - Write-through
      - Update both upper and lower levels
      - Simplifies replacement, but may require write buffer
    - Write-back
      - Update upper level only
      - Update lower level when block is replaced
      - Need to keep more state
    - Virtual memory
      - Only write-back is feasible, given disk write latency

Memory Optimization

Framework

- Common Framework for Memory Hierarchy
  - Design tradeoffs

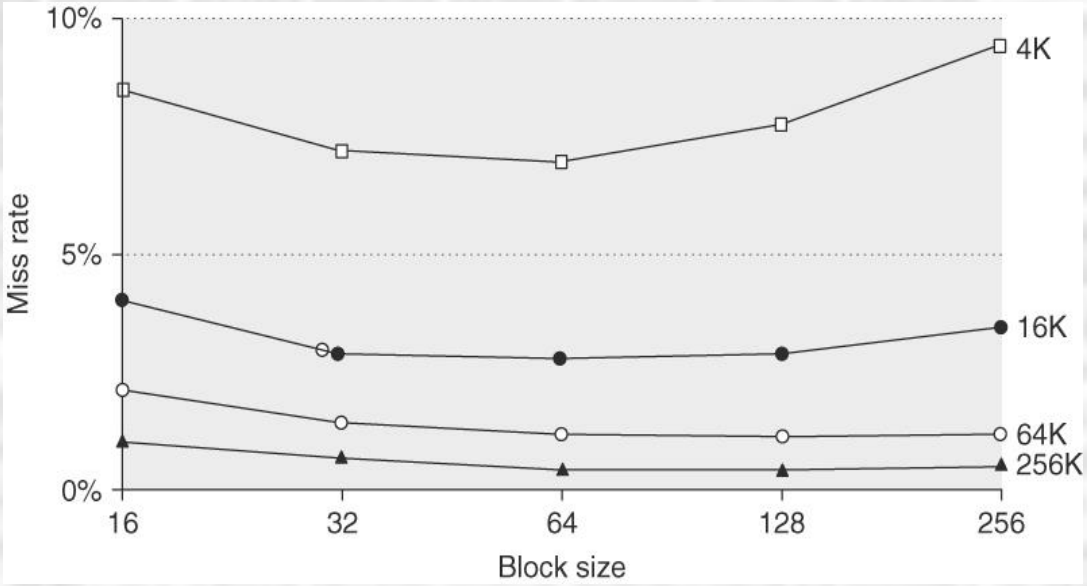
Design change	Effect on miss rate	Negative performance effect
Increase cache size	Decrease capacity misses	May increase access time
Increase associativity	Decrease conflict misses	May increase access time
Increase block size	Decrease compulsory misses	Increases miss penalty. For very large block size, may increase miss rate due to pollution.

Memory Optimization

Optimizations

- Larger Block Size
  - \+ Take advantage of spatial locality → lower miss rates
  - \- Higher miss penalty

Miss rate vs Block size for 4 different cache sizes



Memory Optimization

Optimizations

- Larger Cache Size

  \+ Reduces capacity misses → lower miss rates

  \- Possibly longer Hit time
  \- Cost and area

Memory Optimization

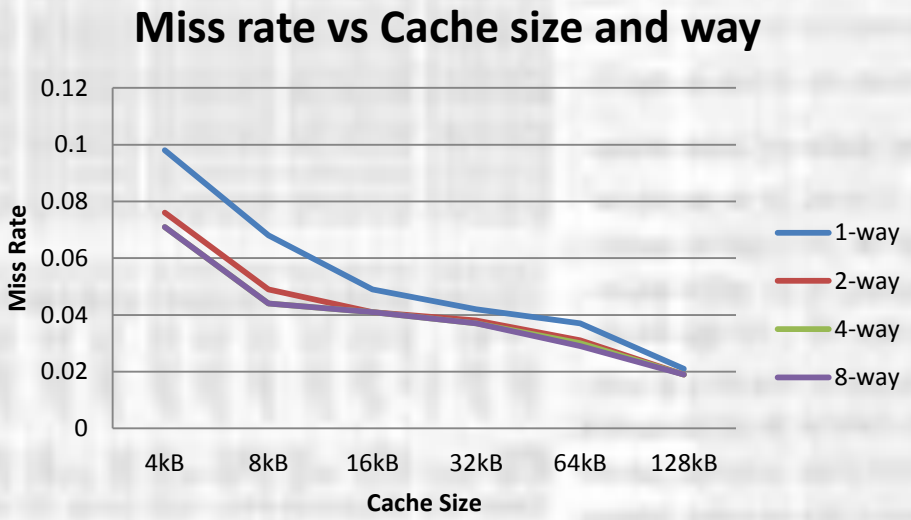
Optimizations

- Higher Associativity
  - \+ Reduces conflict misses → lower miss rates
  - \- Possibly longer Hit time

Rules of thumb:

Diminishing returns for ways > 8

2:1 cache rule – direct mapped cache of size n has the same miss rate as a 2-way cache of size n/2



Memory Optimization

Optimizations

- Multilevel Caches

  \+ Small caches support fast processors
  \- Small caches → higher miss rates

  Memory speeds not keeping up with processor speeds
  → Higher miss penalties

  Multilevel caches allow:
  local caches to be fast to support fast processors
  local to second level cache miss penalties to be smaller
  leverage the global miss rate effect to minimize the impact of large main memory miss penalties

Memory Optimization

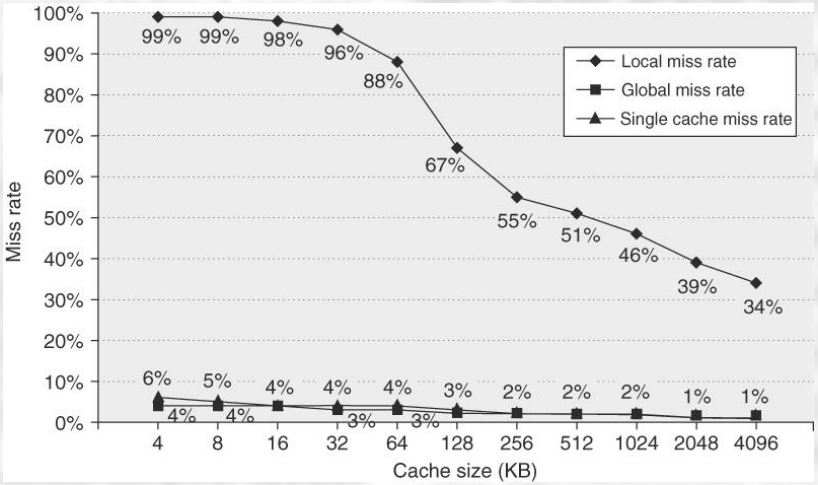
Optimizations

- Multilevel Caches
  - Rules of thumb:

Global miss rate is close to single cache hit rate when L2 is large compared to L1

L2 local miss rate is not a good measure of performance

L2 miss rate with 2 64KB first level caches



Memory Optimization

Optimizations

- Prioritize Read Misses
  - In a write-through cache we introduced the write buffer to free up the cache during writes
    - What if the read is on what is in the write buffer
    - Forced back to waiting on the write
    - Modify the cache to check the write buffer during reads
      - On hit – read the data
  - In a write-back cache we write the dirty block back then read in the new data
    - Add a write buffer here also
      - Write the dirty block to the buffer
      - Do the read
      - Complete the write

Memory Optimization

Optimizations

- Way Prediction
  - Add bits to each way/block to predict the next access
  - Based on prediction, check the predicted way/block first
    → reduced hit times
    - 2-way prediction accuracy can be 90%
    - I caches predict better than D caches

Memory Optimization

Optimizations

- Pipelined Cache Accesses
  - Allows for longer access times but higher throughput
    - Multiple clocks for an access
    - 1 word per clock
    - Can increase the clock rate
  - Increases the miss penalty
    - Especially bad on mispredicted branches

Memory Optimization

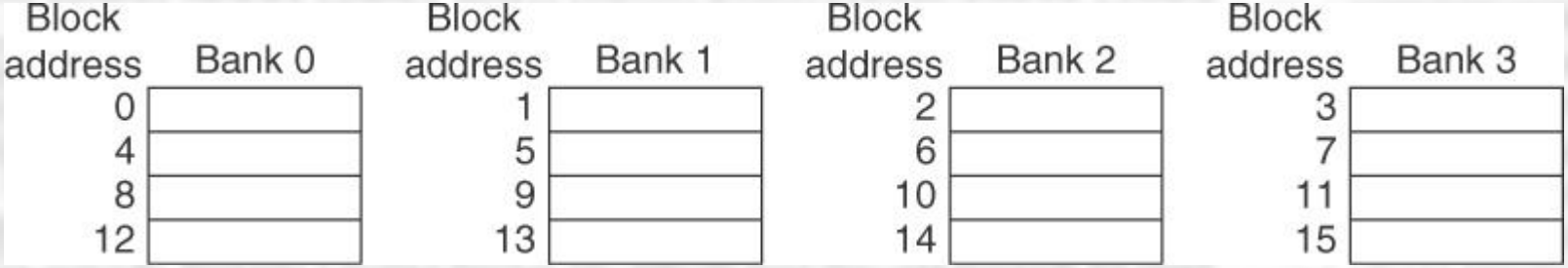
Optimizations

- Non-Blocking Caches
  - Out-of-order execution processors do not need to wait on a miss
  - Called “hit under miss”
  - Works well with L1 misses that hit in L2
    - Enough alternate instructions available to continue with-in response time
  - Cannot hide L2 misses
    - Run out of alternate instructions → stalls

Memory Optimization

Optimizations

- Multibanked Caches
  - Banking allows accesses to be spread across banks
    - Reduces the effective access time
  - Sequential Interleaving – spreads instructions across the banks in modulo form



Memory Optimization

Optimizations

- Critical Word First
  - Valuable for large block sizes
  - Transfers the requested word (out of n in the block) first
    - Reduces wait for reads/loads
- Early Restart
  - Transfer the block in normal order
  - Restart the processor as soon as the requested word is loaded

Memory Optimization

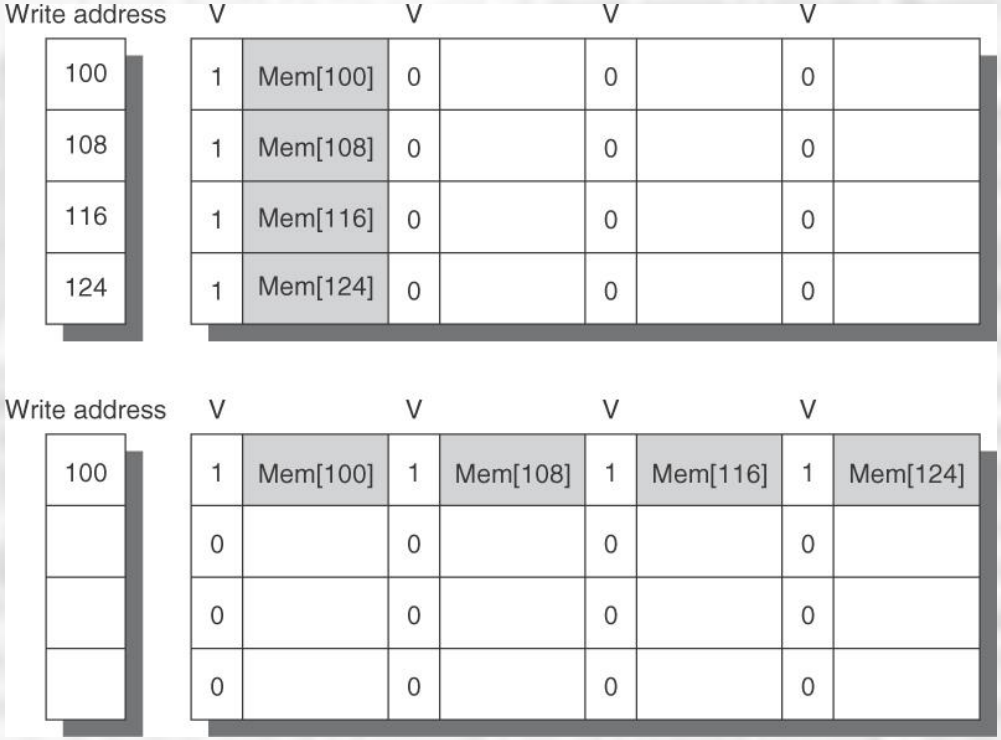
Optimizations

- Merging Write Buffer
  - A second write to a block already in the write buffer
    - Merge the contents
      → reduced stalls

Ex.

4 entry write buffer
each entry 4-64bit words

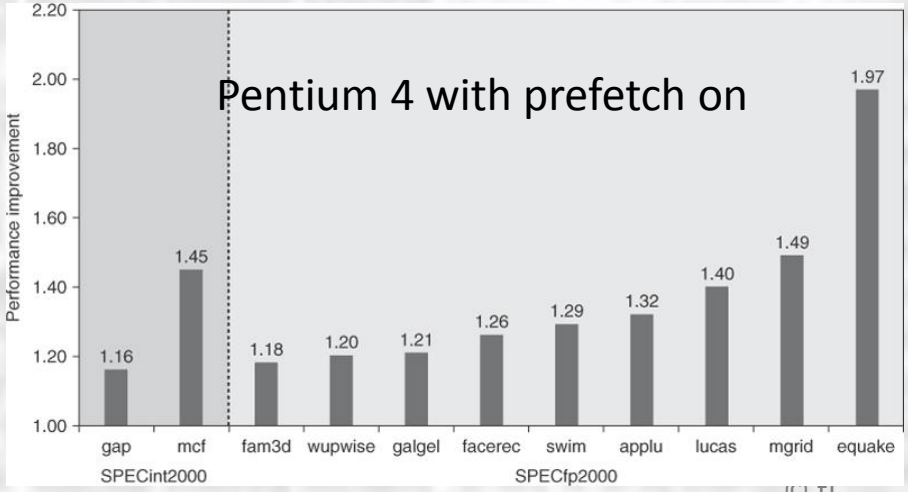
V bit indicates the sequential address is valid



Memory Optimization

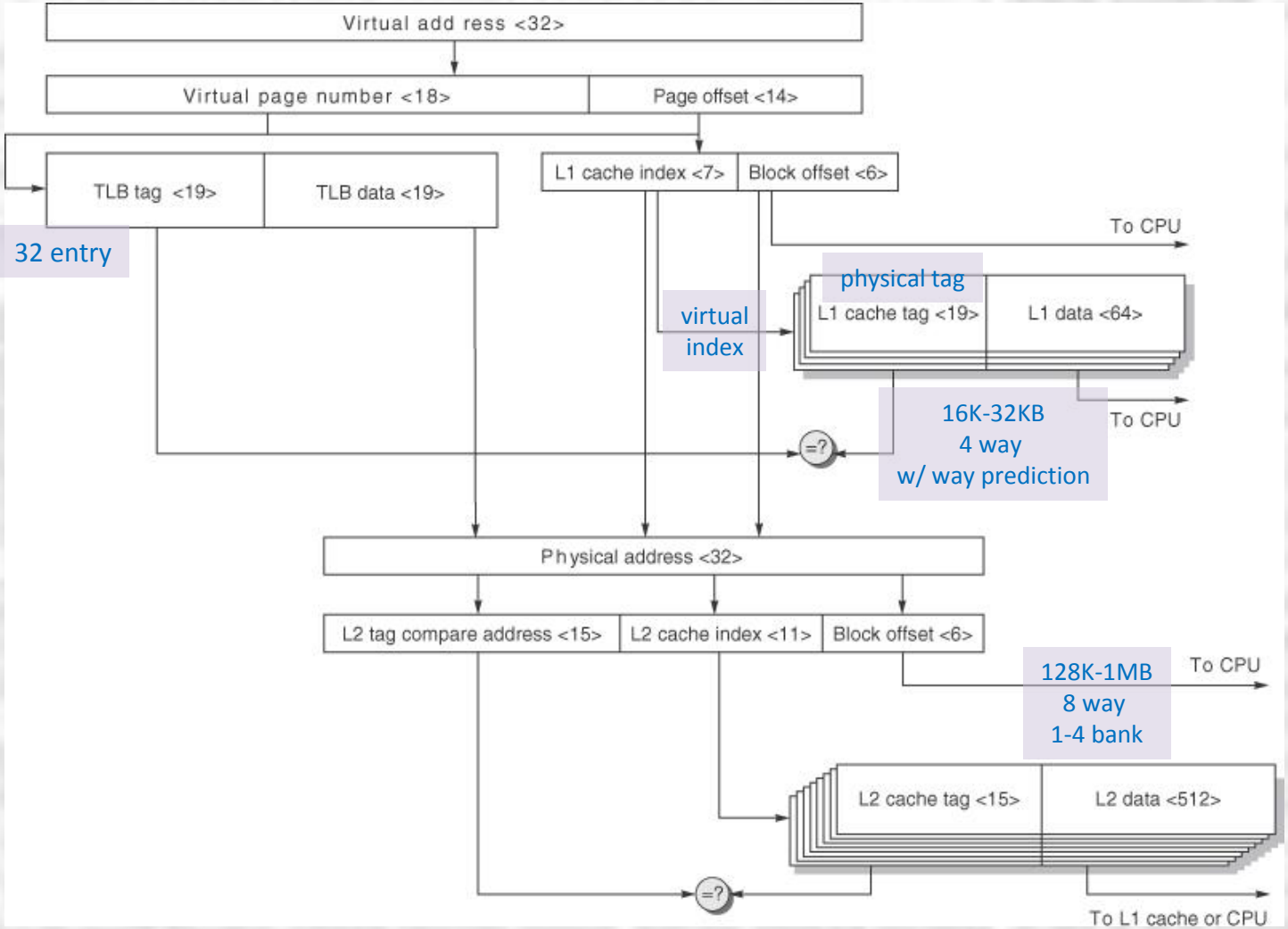
Optimizations

- Hardware Prefetching
  - Instructions
    - On a read from main memory
      - Read the requested block and place in the cache
      - Read the next sequential block and place it in an instruction stream buffer
    - On reads – check the instruction stream buffer first
      - If a hit
        - cancel the main memory read and load from the buffer
        - prefetch the next block
  - Data
    - Same process but less effective
    - Benefits from multiple data stream buffers



Memory Optimization

ARM Cortex A8



Memory Optimization

Intel Core I7

4 core
3 level cache
3 channel memory