Miss Penalty Reduction: Multi-Level Cache

- Make smaller, faster cache or larger, slower cache?
  - Solution: multi-level cache
- Problem: Complicates Performance Analysis

Definitions:
- Local miss rate
  - Misses in this cache divided by the total number of memory accesses to this cache (Miss rate$L^1$)
- Global miss rate
  - Misses in this cache divided by the total number of memory accesses generated by the CPU (Miss Rate$L^1 \times Miss Rate_L^2$)

AMAT = Hit Time$L^1 + Miss Rate_L^1 \times Miss Penalty_L^1$

Second-level cache Example

- For a particular application on 2-level cache hierarchy:
  - 1000 memory references
  - 40 misses in L1
  - 20 misses in L2

- Calculate local and global miss rates
  - Miss rate$L^1 = 40/1000 = 4\%$ (global and local)
  - Global miss rate$L^2 = 20/1000 = 2\%$

AMAT = Hit Time$L^1 + Miss Rate_L^1 \times (Hit Time_L^2 + Miss Rate_L^2 + Miss Penalty_L^2)$

Comparing Local and Global Miss Rates

- 32 KByte 1st-level cache; increasing 2nd-level cache
- L2 smaller than L1 is impractical
- Global miss rate similar to single level cache rate provided L2 >> L1
- Local miss rate not a good measure for secondary cache
- Local miss rate$L^2 \neq f(Miss rate_L^1)$

Reducing Misses on L2

- Tradeoffs
  - Goal of cache design: balance fast hits + few misses
  - In L1 want fast hits, in L2 want fewer misses (make big)
  - Associativity: higher is better for L2
    - Small impact on hit time
    - Lower impact of conflict misses
  - Block size: larger size → larger blocks
    - Little impact on conflict misses
    - Larger relative access time to mem
- Overall: multi-level caches use HW to reduce miss penalty

Reducing Miss Penalty: Early Restart and Critical Word First

- Don’t wait for full block to be loaded before restarting CPU
  - Early restart—As soon as the requested word of the block arrives, send it to the CPU and let the CPU continue execution
  - Critical Word First—Request the missed word first from memory and send it to the CPU as soon as it arrives; let the CPU continue execution while filling the rest of the words in the block. Also called wrapped fetch and requested word first
- Generally useful only in large blocks
- Spatial locality problem: tend to want next sequential word, so not clear if benefit by early restart

Reducing Miss Penalty: Read Priority over Write on Miss

- Problem
  - Write back with write buffers offer RAW conflicts with main memory reads on cache misses
  - If simply wait for write buffer to empty might increase read miss penalty by 50% (old MIPS 1000)

- Solution
  - Check write buffer contents before read;
    - if no conflicts, let the memory access continue
  - Write Back?
    - Read miss replacing dirty block
    - Normal: Write dirty block to memory, and then do the read
    - Instead copy the dirty block to a write buffer, then do the read, and then do the write
    - CPU stall less since restarts as soon as do read
Reducing Miss Penalty via Victim Cache

- Problem:
  - Direct mapped cache offers lowest hit time, but suffers from conflict misses
- Solution:
  - Small "victim cache" that contains only blocks discarded due to a miss
  - Check victim cache on miss, if block found then saves time
  - Sample result: 4KB DM cache with 4-entry victim cache removed 20%-90% of conflict misses

Virtual to Physical Addressing

- Problem: CPU uses virtual addresses (dynamic binding and allocation), must map every virtual address to a physical address (this takes time for any memory access)
- Physical requires address translation prior to access
  - Virtual to physical translation, requires flushing and re-aligning
  - Every physical cache memory block in memory maps to one physical location in cache
  - For virtual cache block in memory may map to more than one virtual location in cache
  - Physical address managed by OS, may change through duration of program execution
Virtual vs. Physical caches

- Refers to cache address associated with stored data
- Address translation involves indexing + comparison (tag)
  - Virtual+virtual OR physical+physical OR virtual+physical...
- Common case: hits, so why not use virtual+virtual?
  - Protection: is access allowed? (store local and check every access)
  - Context switch: must flush cache for new P (add PID to tag)
  - Aliasing: virtual A and B map to physical C
    - HW antialiasing: disallow multiple copies of same physical addr in cache
    - SW page coloring: aliases must share same address bits
      - Example: 2^64 sets, 2^12 pages (blocks): 54 for virtual memory
      - Guarantee any virtual/physical address has same last 18 bits (share map to same set)
- I/O: uses physical addresses (address translation)

Practical Alternative to Avoiding Address Translation (Virtually indexed, Physically Tagged)

- Use virtual index + physical tag
  - Page offset is same in virtual and physical address (use as index, find set)
  - Fast location of index allows reads to begin immediately
  - Virtual address translation and cache read overlapped
  - Use translated physical tag to select data

- Limitation: DM cache (large index) can be no bigger than page size
- For bigger cache: increased associativity, page coloring