378 Lecture (11/19/97)

Alec Wolman

A Common Framework

- Caches, TLBs, and Paged VM all use the same underlying principles

Feature	Cache	TLB	Paged VM
Size (blocks)	1K-10K	32-4K	2K-250K
Size (bytes)	8KB-8MB	128-8K	8MB-8GB
Block Size (bytes)	16-256	**X**	4KB-64KB
Miss Penalty (cycles)	10-100	10-100	100us-10ms
Miss Rate	.1-10%	.01-2%	.00001-.0001%

4 Questions

- How do we classify memory hierarchies?
  - Block placement
    - direct mapped, set associative, fully associative
  - Block identification
    - indexing, set search, lookup table
  - Block replacement
    - LRU, Random, MRU
  - Write strategy
    - Write through, Write back

The 3(or 4) C's

- Ways to characterize cache misses
- Compulsory Misses
  - Miss on the first reference
- Capacity Misses
  - Miss because cache size is too small to hold the data between two consecutive references
- Conflict Misses
  - Miss because blocks collide on the set address
- Coherence (multi-processors)

Conflict Misses

- Fully associative placement avoids all conflict misses
- Associativity is expensive in hardware
  - requires comparator logic, and may require a multiplexor
  - Extra logic of a set associative cache affects the *hit* time, and potentially the CPU clock cycle time
- Sometimes called collision misses

Capacity Misses

- Thrashing
  - If working set exceeds the cache size, then much time is spent shuffling data back and forth between upper-level and lower-level of memory hierarchy
  - In a paging system, time is wasted shuffling data between memory and disk. Thus program runs at close to disk speed, which is painfully slow
- Make your cache larger
  - could affect cycle time
- Buy more memory (if you can)

Compulsory Misses

- miss on first reference
- Increase block size to reduce compulsory misses
  - this may increase the miss penalty
    - early restart and requested word first
  - increasing block size too much increases conflict misses
    - pollution point

Advanced Techniques

- stall on use - only works for data caches
- non-blocking caches
  - allows processor to continue executing even with instructions that access the data cache
  - hit under miss (multiple hits during one miss)
  - miss under miss (multiple outstanding misses)
- most effective when processor supports out-of-order execution