Computer Science 703

Advance Computer Architecture

2006 Semester 1

Lecture Notes

28Mar06

Extended Cache Memory:

Multi-level, virtually addressed, cache coherent

James Goodman

Department of Computer Science

Reference

• Wang, Baer, Levy, "Organization and performance of a two-level virtual-real cache hierarchy”, *ISCA-16*, pp. 140-148, June 1989.

Virtually-Addressed Cache

Problems:

1. Virtually-addressed cache must be capable of handling synonyms, that is, multiple virtual addresses that map to the same physical address.
2. While address translation is not required before a virtual cache lookup, address translation is still needed following a miss.
3. I/O devices use physical addresses, requiring reverse translation.
4. A virtual cache may need to be invalidated on a context switch because virtual addresses are unique to a single process.
5. In a multiprocessor system, the use of a virtually-addressed cache may complicate cache coherence because bus addresses are physical, also requiring a reverse translation.

Write Policies

Write-through vs. write-back

- L1: write-through? (No, use write-back here too)
  - simpler control: no incoherence between levels
- L2: write-back?

Write buffers: overlapping write-through

- But writes tend to be clustered!
- Introduce new coherency problem: must snoop buffers

Translation

L1 cache virtual: translation only required on miss

- But invalidation coming in from bus means reverse translating to find cache line
- Reverse translation: each block in L2 points to block in L1. Note that only a few bits are needed

Context Switching

Keeping track of multiple virtual spaces

- Add processor ID bits to extend virtual address
- But page or PID reassignments may require searching or flushing a virtual address cache
- By marking cache lines invalid after TLB operation allows deferring writebacks (careful about coherence!)
- Neat trick: keep pointers in V-cache to R-cache entry, allowing writeback even after translation becomes invalid

Two-level cache

- L2 cache can't do LRU replacement, since it doesn't have access to appropriate information
- Can data be in L1 cache but not in L2?
- Inclusion property: All data in L1 must reside also in L2.

Cache coherence

Inclusion allows using L2 cache to filter invalidation requests.

- Inclusion property helps replacement (replace any block not in L1)
- Problem with inclusion: L2 is larger, but also contains larger lines
  - What if associativity is not sufficient to hold an arbitrary set of lines in L1?
  - Must purge entry from L1 even if it is active

Final Note on Analysis

Wang, Baer & Levy assumed a blocking cache

- Non-blocking cache is standard today
- Concurrency is higher, but complicates many operations