

Caches in Systems

COMP 252 - Lecture 4

Antoni Pop

antoni.pop@manchester.ac.uk

- ▶ **Control bits**
 - ▶ Valid & Dirty bits
- ▶ **Locality tradeoffs and compromises**
 - ▶ Impact of cache line size
 - ▶ Spatial vs. temporal locality
 - ▶ Separating Instruction & Data caches
- ▶ **Multiple-level caches**
 - ▶ Why, how
 - ▶ Performance model in a cache hierarchy
- ▶ **First lab**

Today's Lecture – Learning Objectives

- ▶ “3 x C’s” model of cache performance
- ▶ Time penalties for starting with empty cache
- ▶ Systems interconnect issues with caching and solutions!
- ▶ Caching and Virtual Memory

Describing Cache Misses

- ▶ Compulsory Misses
 - ▶ Cold start

- ▶ Capacity Misses
 - ▶ Even with full associativity, cache cannot contain all the blocks of the program

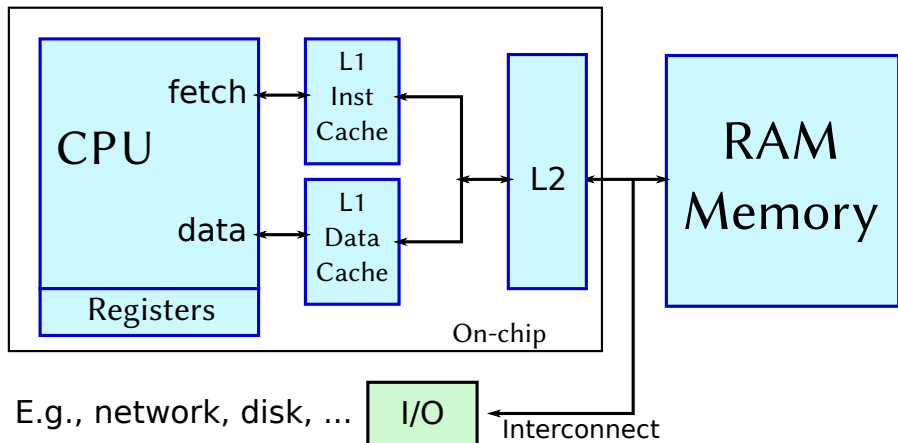
- ▶ Conflict Misses
 - ▶ Multiple blocks compete for the same set.
 - ▶ This would not happen in fully associative cache

Cache Performance

Today's caches, how long does it take

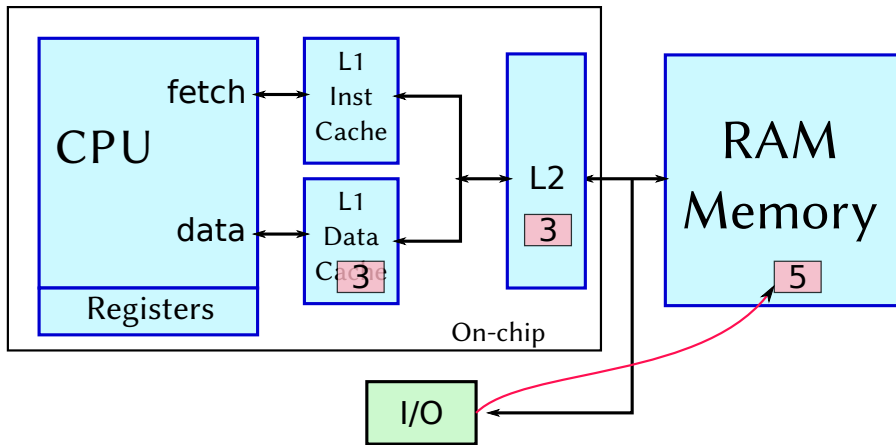
- ▶ To fill L3 cache? (8MB)
- ▶ To fill L2 cache? (256KB)
- ▶ To fill L1 D cache? (32KB)
 - ▶ Number of lines = (cache size) / (line size)
 - ▶ Number of lines = $32\text{K}/64 = 512$
 - ▶ $512 \times$ memory access times at $20\text{nS} = 10 \mu\text{S}$
 - ▶ 20,000 clock cycles at 2GHz

Caches in Systems



- ▶ Typical I/O bandwidth?
- ▶ What could go wrong?

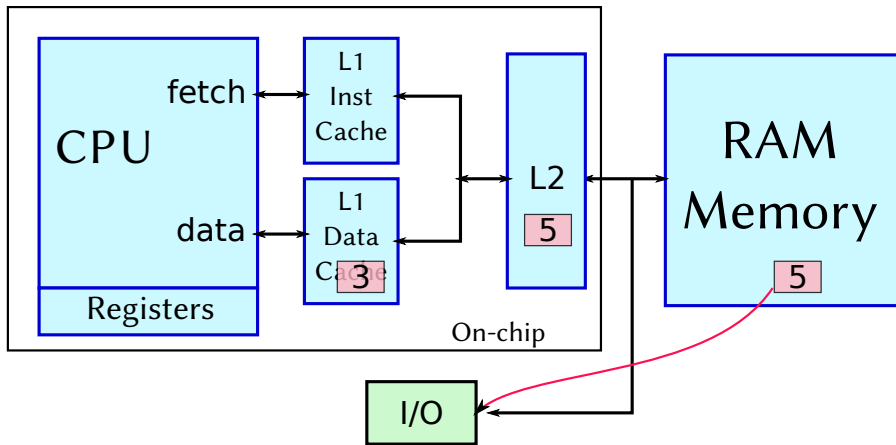
Cache Consistency Problem (1)



► Problem

- I/O writes to memory
- Cache data is no longer up-to-date

Cache Consistency Problem (2)



► Problem

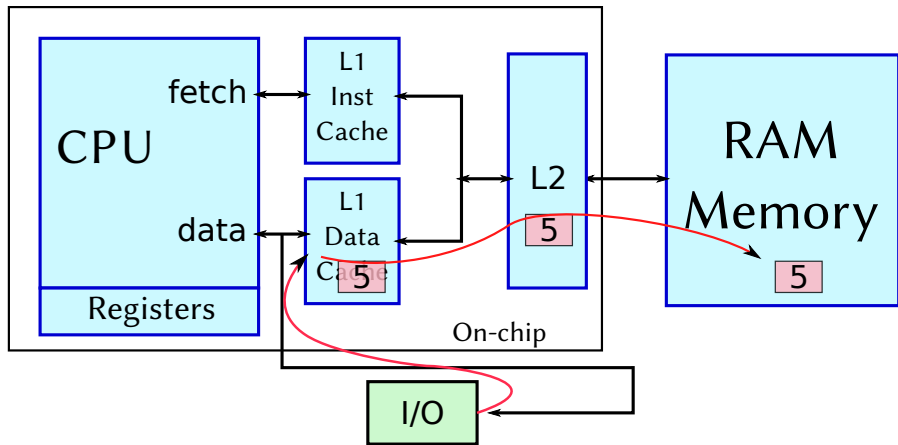
- I/O reads from memory
- But the cache has a new, updated value

Cache Consistency Software Solutions

- ▶ O/S knows where I/O takes place in memory
 - ▶ Mark I/O areas as non-cacheable (how?)

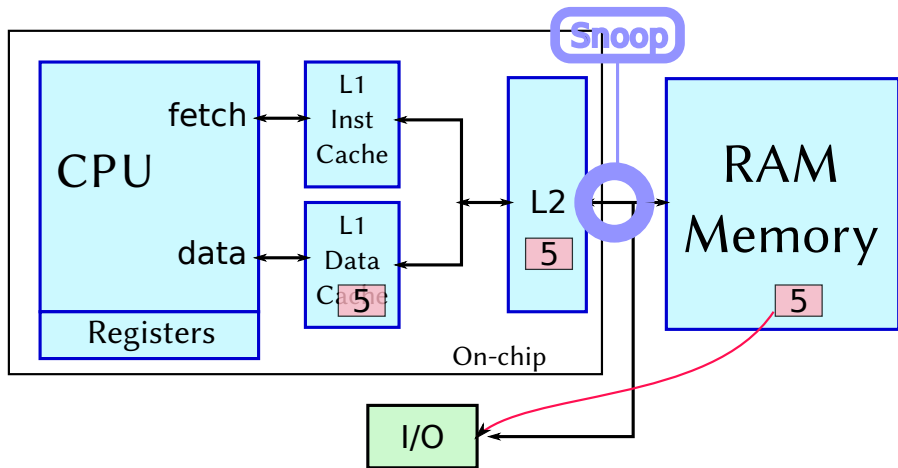
- ▶ O/S knows when I/O starts and finishes
 - ▶ Clear caches before & after I/O?

Hardware Solution 1



- ▶ Disadvantage
 - ▶ Slows down the cache
 - ▶ "Pollutes" the cache (replaces potentially useful data)

Hardware Solution 2 (Snooping)



- ▶ Snoop logic in the cache
 - ▶ Observes every memory cycle
 - ▶ Scalability issues

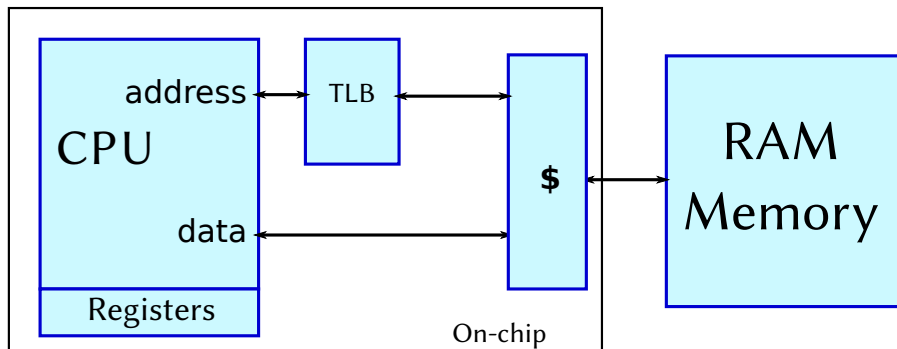
Caches and Virtual Addresses

- ▶ CPU addresses – virtual
- ▶ Memory addresses – physical

- ▶ Recap...
 - ▶ Use Translation-Lookaside Buffer (TLB) to translate V-to-P

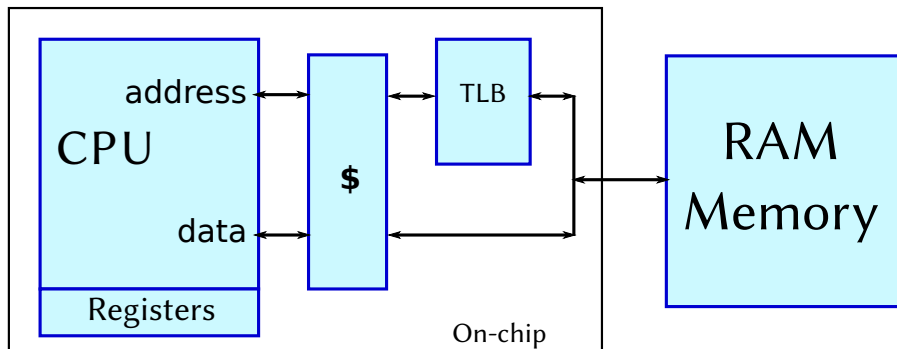
- ▶ What addresses in cache?

Option 1 : Cache by Physical Addresses



- ▶ **Slow**
 - ▶ Address translation is **in series** with cache

Option 2 : Cache by Virtual Addresses



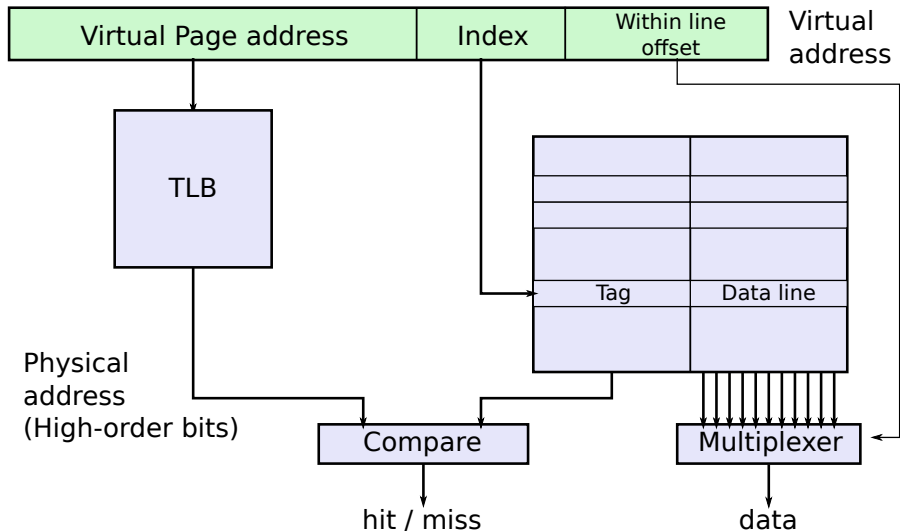
► **More functional difficulties**

- Snooping
- Aliasing

Option 3 : Translate in Parallel with Cache Lookup

- ▶ Translation only affects high-order bits of address
- ▶ Address within page remains unchanged
- ▶ Low-order bits of Physical Address = low-order bits of Virtual Address
- ▶ Select “index” field of cache address from within low-order bits
- ▶ Only “Tag” bits changed by translation

Option 3 in operation



Summary

- ▶ “3 x C’s” model of cache performance
- ▶ Systems interconnect issues with caching and solutions!
 - ▶ Non-cacheable areas
 - ▶ Cache flushing
 - ▶ Snooping
- ▶ Caching and Virtual Memory
 - ▶ Physical to virtual conversion (TLB)
 - ▶ Cache architectures to support P-to-V conversion

Caches Module Summary

▶ Why are caches essential?

- ▶ Speed imbalance CPU vs. RAM
- ▶ **Performance**

▶ How do caches work?

- ▶ Locality
- ▶ Associativity
- ▶ Replacement policy
- ▶ Line size
- ▶ Cache lookup: how is data found (address splitting in tag, index, word ID, alignment)
- ▶ Cache hierarchy
- ▶ **Cache misses (3 Cs)**

▶ What is the impact of caches?

- ▶ **Performance model** (including in a cache hierarchy)
- ▶ Interaction with other system components

▶ Does this matter to you?