



Master Informatics Eng.

2016/17

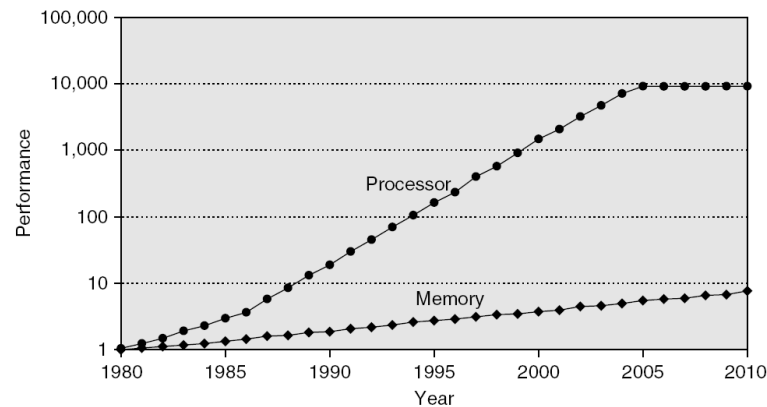
A.J.Proença

Concepts from undergrad Computer Systems (3)
(most slides are borrowed)

Introduction

- Programmers want unlimited amounts of memory with low latency
- Fast memory technology is more expensive per bit than slower memory
- Solution: organize memory system into a hierarchy
 - Entire addressable memory space available in largest, slowest memory
 - Incrementally smaller and faster memories, each containing a subset of the memory below it, proceed in steps up toward the processor
- Temporal and spatial locality insures that nearly all references can be found in smaller memories
 - Gives the illusion of a large, fast memory being presented to the processor

Memory Performance Gap



Memory Hierarchy Design

- Memory hierarchy design becomes more crucial with recent multi-core processors:
 - Aggregate peak bandwidth grows with # cores:
 - Intel Core i7 can generate two references per core per clock
 - Four cores and 3.2 GHz clock
 - 25.6 billion 64-bit data references/second +
 - 12.8 billion 128-bit instruction references
 - = 409.6 GB/s!
 - DRAM bandwidth is only 6% of this (25 GB/s)
 - Requires:
 - Multi-port, pipelined caches
 - Two levels of cache per core
 - Shared third-level cache on chip

The Memory Hierarchy

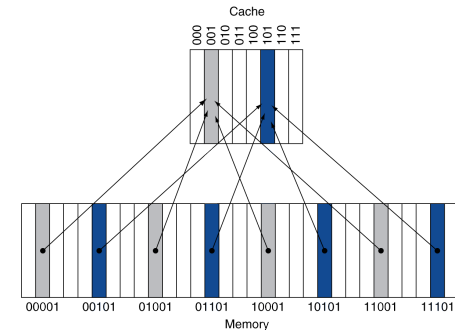
The BIG Picture

- Common principles apply at all levels of the memory hierarchy
 - Based on notions of caching
- At each level in the hierarchy
 - Block placement
 - Finding a block
 - Replacement on a miss
 - Write policy



Direct Mapped Cache

- Location determined by address
- Direct mapped: only one choice
 - (Block address) modulo (#Blocks in cache)



- #Blocks is a power of 2
- Use low-order address bits



Associative Caches

- Fully associative
 - Allow a given block to go in any cache entry
 - Requires all entries to be searched at once
 - Comparator per entry (expensive)
- *n*-way set associative
 - Each set contains *n* entries
 - Block number determines which set
 - (Block number) modulo (#Sets in cache)
 - Search all entries in a given set at once
 - *n* comparators (less expensive)



How Much Associativity

- Increased associativity decreases miss rate
 - But with diminishing returns
- Simulation of a system with 64KB D-cache, 16-word blocks, SPEC2000
 - 1-way: 10.3%
 - 2-way: 8.6%
 - 4-way: 8.3%
 - 8-way: 8.1%



Block Placement

- Determined by associativity
 - Direct mapped (1-way associative)
 - One choice for placement
 - n-way set associative
 - n choices within a set
 - Fully associative
 - Any location
- Higher associativity reduces miss rate
 - Increases complexity, cost, and access time



Replacement Policy

- Direct mapped: no choice
- Set associative
 - Prefer non-valid entry, if there is one
 - Otherwise, choose among entries in the set
- Least-recently used (LRU)
 - Choose the one unused for the longest time
 - Simple for 2-way, manageable for 4-way, too hard beyond that
- Random
 - Gives approximately the same performance as LRU for high associativity



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



Multilevel Caches

- Primary cache attached to CPU
 - Small, but fast
- Level-2 cache services misses from primary cache
 - Larger, slower, but still faster than main memory
- Main memory services L-2 cache misses
- Some high-end systems include L-3 cache

