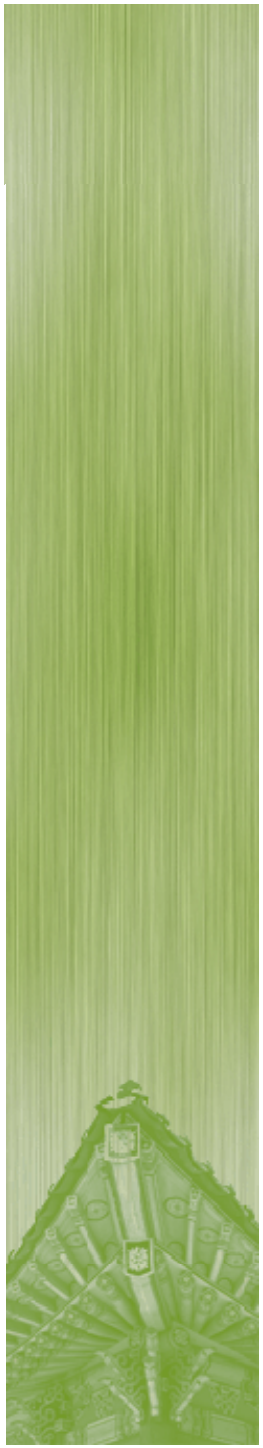


# Memory Hierarchy

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

# The Memory Hierarchy

- **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

# 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

# Finding a Block

| Associativity | Location method | Tag comparisons |
|---|---|---|
| Direct mapped | Index | 1 |
| n-way set associative | Set index, then search entries within the set | n |
| Fully associative | Search all entries | #entries |
| | Full lookup table | 0 |

- **Hardware caches**
  - Reduce comparisons to reduce cost
- **Virtual memory**
  - Full table lookup makes full associativity feasible
  - Benefit in reduced miss rate

# Replacement

- **Choice of entry to replace on a miss**
  - Least recently used (LRU)
    - Complex and costly hardware for high associativity
  - Random
    - Close to LRU, easier to implement
- **Virtual memory**
  - LRU approximation with hardware support

# Write Policy

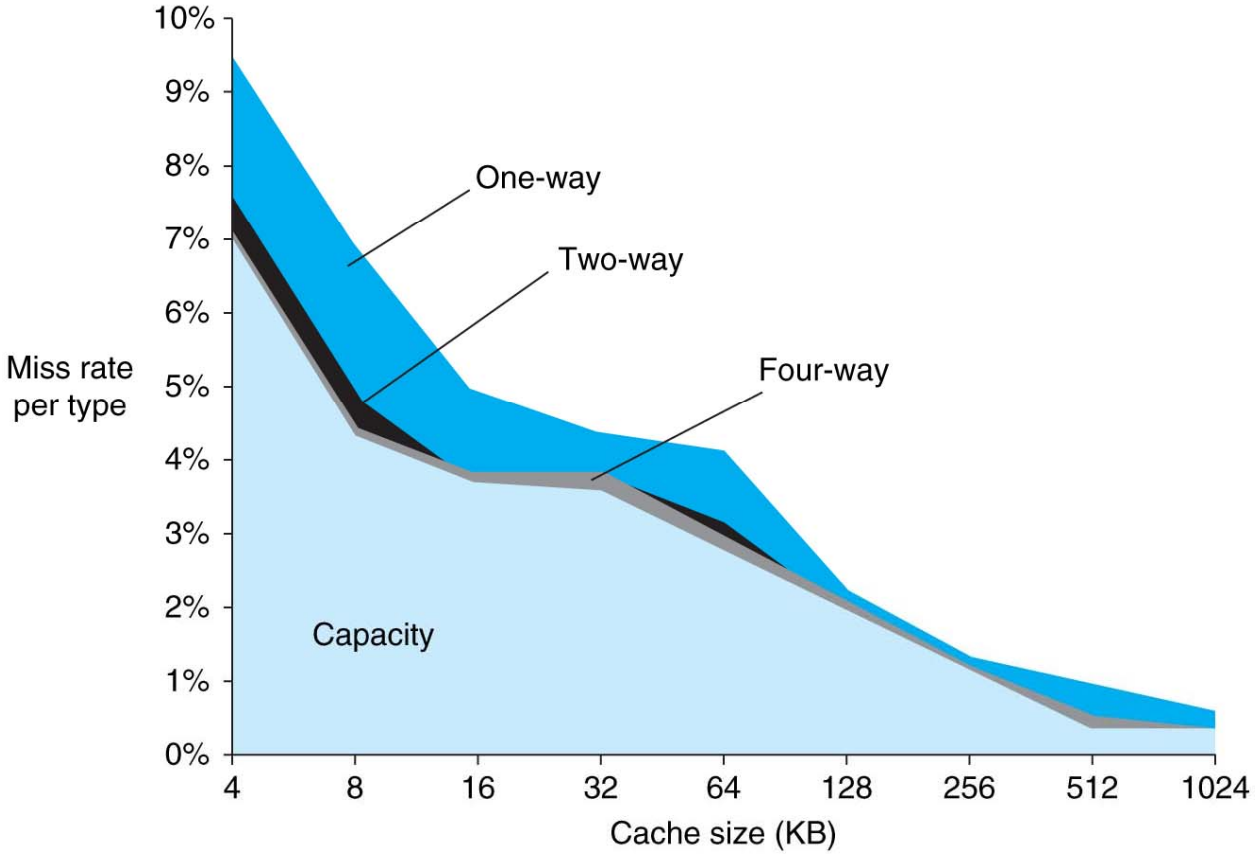
- **Write-through**
  - Update both upper and lower levels
  - Simplifies replacement, but may require write buffer
- **Write-back**
  - Update upper level only
  - Update lower level when block is replaced
  - Need to keep more state
- **Virtual memory**
  - Only write-back is feasible, given disk write latency

# Sources of Misses (1)

- **Compulsory misses (aka cold start misses)**
  - First access to a block
- **Capacity misses**
  - Due to finite cache size
  - A replaced block is later accessed again
- **Conflict misses (aka collision misses)**
  - In a non-fully associativity cache
  - Due to competition for entries in a set
  - Would not occur in a fully associative cache of the same total size

# Sources of Misses (2)

- **Results of SPEC2000 benchmarks**
  - Compulsory misses: 0.006%

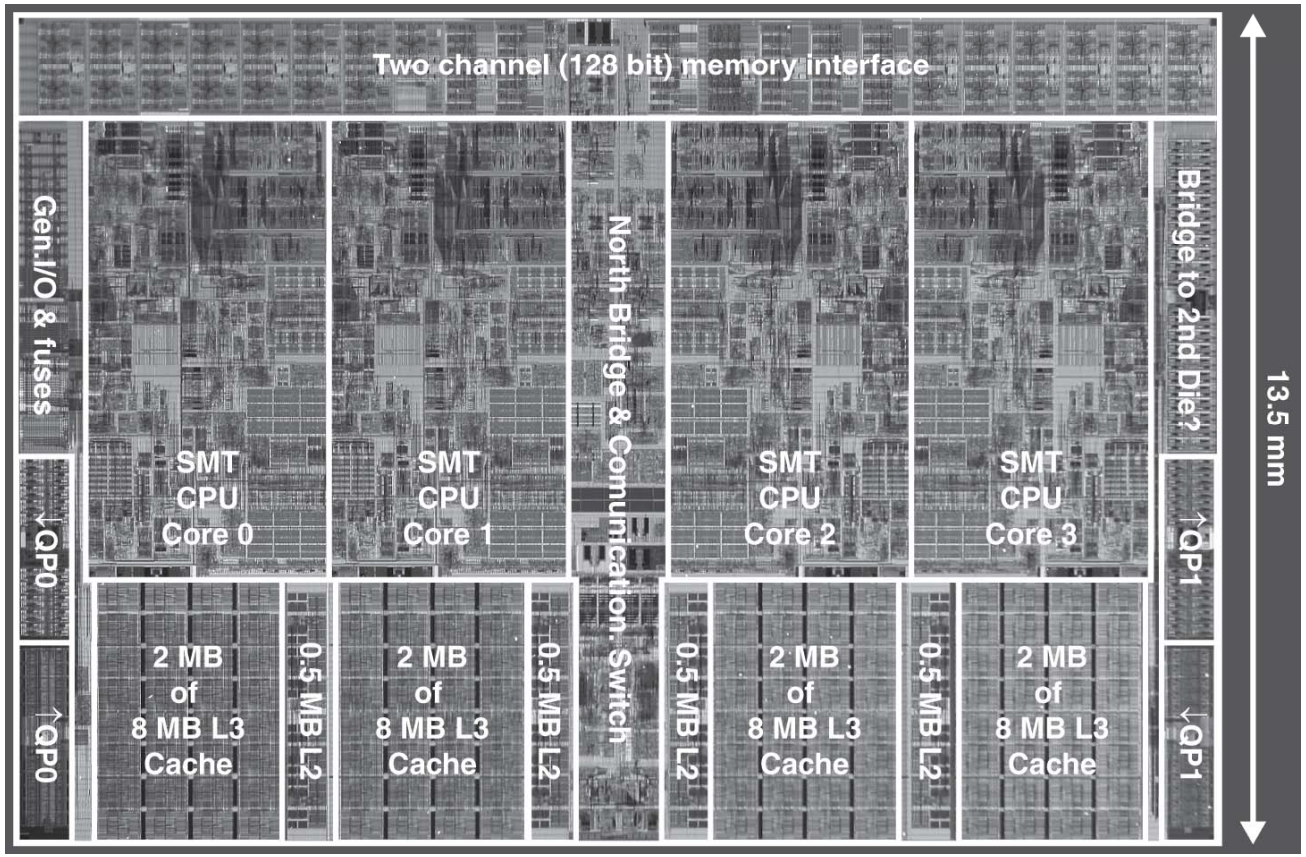


# Cache Design Trade-offs

| Design change | Effect on miss rate | Negative performance effect |
|---|---|---|
| Increase cache size | Decrease capacity misses | May increase access time |
| Increase associativity | Decrease conflict misses | May increase access time |
| Increase block size | Decrease compulsory misses | Increases miss penalty. For very large block size, may increase miss rate due to pollution. |

# Examples (1)

- **Intel Nehalem 4-core processor**
  - Per core: L1 I-/D-cache (32KB each), 512KB L2 cache



# Examples (2)

- 2-Level TLB organization

| | Intel Nehalem | AMD Opteron X4 |
|---|---|---|
| Virtual addr | 48 bits | 48 bits |
| Physical addr | 44 bits | 48 bits |
| Page size | 4KB, 2/4MB | 4KB, 2/4MB |
| L1 TLB (per core) | L1 I-TLB: 128 entries for small pages, 7 per thread (2✕) for large pages<br>L1 D-TLB: 64 entries for small pages, 32 for large pages<br>Both 4-way, LRU replacement | L1 I-TLB: 48 entries<br>L1 D-TLB: 48 entries<br>Both fully associative, LRU replacement |
| L2 TLB (per core) | Single L2 TLB: 512 entries<br>4-way, LRU replacement | L2 I-TLB: 512 entries<br>L2 D-TLB: 512 entries<br>Both 4-way, round-robin LRU |
| TLB misses | Handled in hardware | Handled in hardware |

# Examples (3)

- 3-Level cache organization

| | Intel Nehalem | AMD Opteron X4 |
|---|---|---|
| L1 caches (per core) | L1 I-cache: 32KB, 64-byte blocks, 4-way, approx LRU replacement, hit time n/a<br>L1 D-cache: 32KB, 64-byte blocks, 8-way, approx LRU replacement, write-back/allocate, hit time n/a | L1 I-cache: 32KB, 64-byte blocks, 2-way, LRU replacement, hit time 3 cycles<br>L1 D-cache: 32KB, 64-byte blocks, 2-way, LRU replacement, write-back/allocate, hit time 9 cycles |
| L2 unified cache (per core) | 256KB, 64-byte blocks, 8-way, approx LRU replacement, write-back/allocate, hit time n/a | 512KB, 64-byte blocks, 16-way, approx LRU replacement, write-back/allocate, hit time n/a |
| L3 unified cache (shared) | 8MB, 64-byte blocks, 16-way, replacement n/a, write-back/allocate, hit time n/a | 2MB, 64-byte blocks, 32-way, replace block shared by fewest cores, write-back/allocate, hit time 32 cycles |

# Examples (4)

- **Miss penalty reduction**
  - Return requested word first
    - Then back-fill rest of block
  - Non-blocking miss processing
    - Hit under miss: allow hits to proceed
    - Miss under miss: allow multiple outstanding misses
  - Hardware prefetch: instructions and data
  - Opteron X4: bank interleaved L1 D-cache
    - Two concurrent accesses per cycle
- **Inclusion vs. exclusion policy**
  - Intel & most other processors: inclusion policy
  - AMD processors: exclusion policy

# Concluding Remarks

- **Fast memories are small, large memories are slow**
  - We really want fast, large memories ☹
  - Caching gives this illusion ☺
- **Principle of locality**
  - Programs use a small part of their memory space frequently
- **Memory hierarchy**
  - L1 cache ↔ L2 cache ↔ … ↔ DRAM memory ↔ disk
- **Memory system design is critical for multiprocessors**