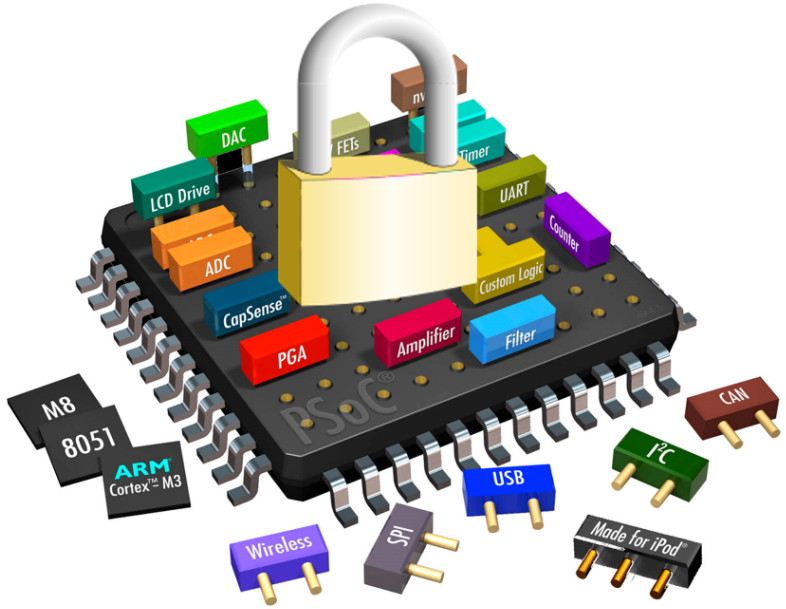
Advanced Computer Architecture

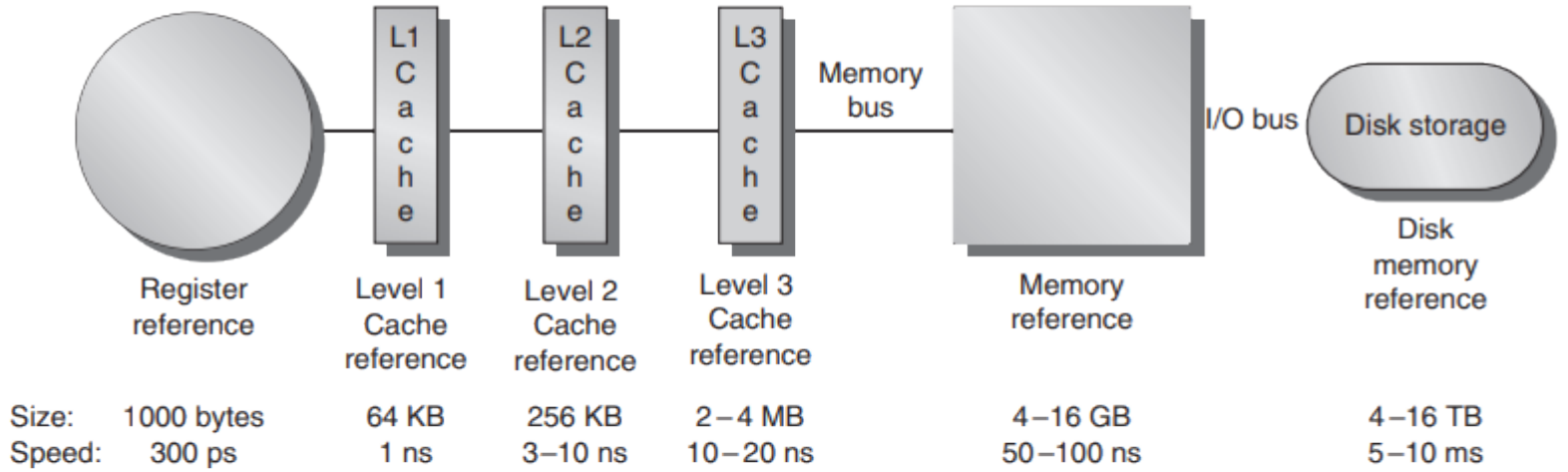
Memory Hierarchy Design



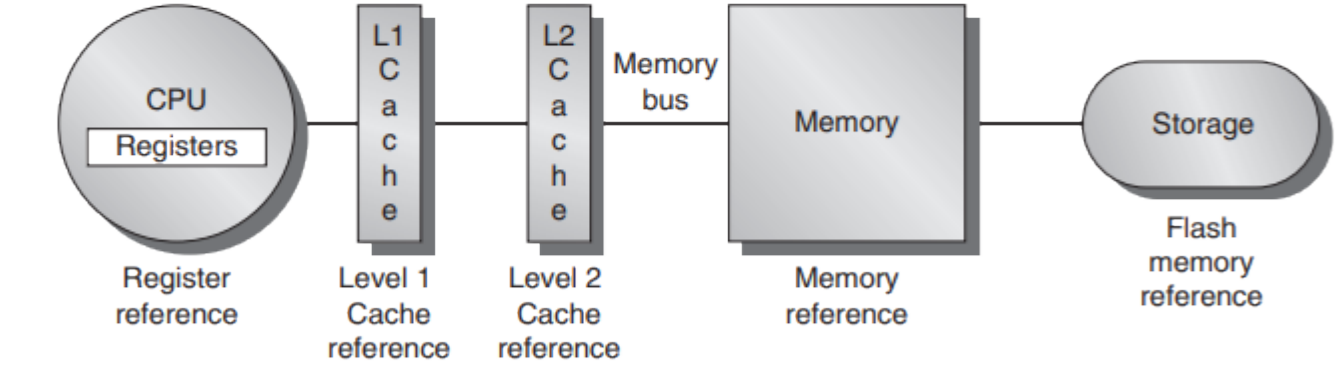
Memory Hierarchy

> "Ideally one would desire an indefinitely large memory capacity such that any particular word would be immediately available. We are forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible."

— A. W. Burks, H. H. Goldstine, and J. von Neumann (1946)

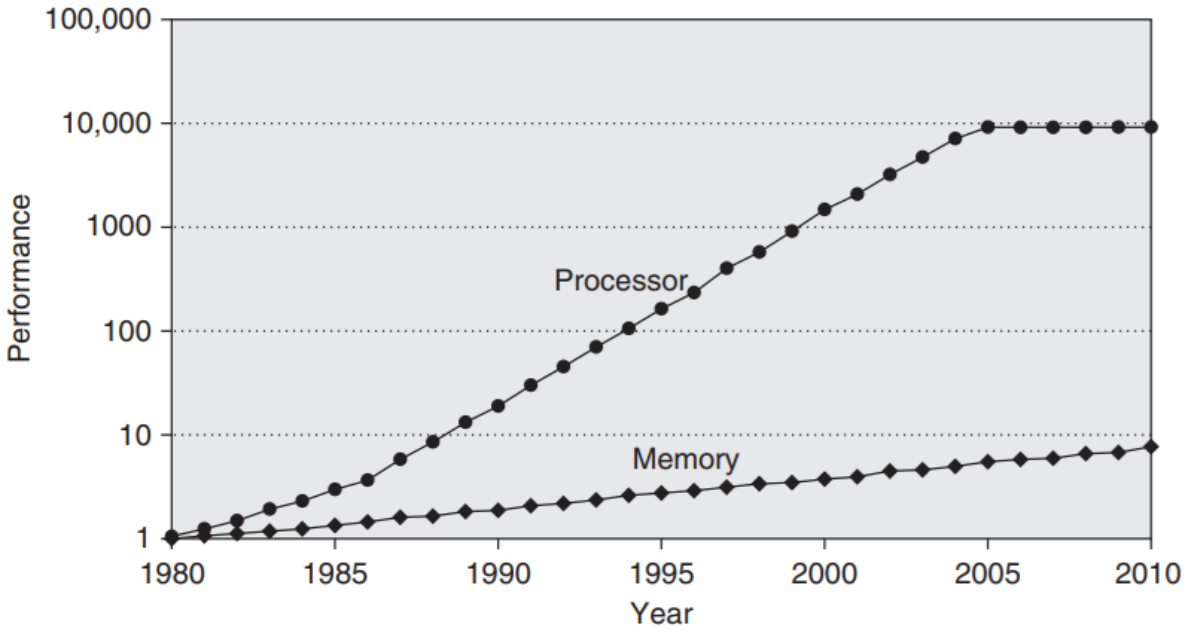


(a) Memory hierarchy for server



(b) Memory hierarchy for a personal mobile device

Growing Processor-Memory Performance Gap

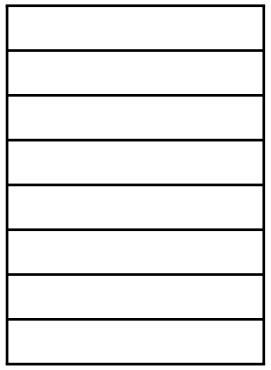


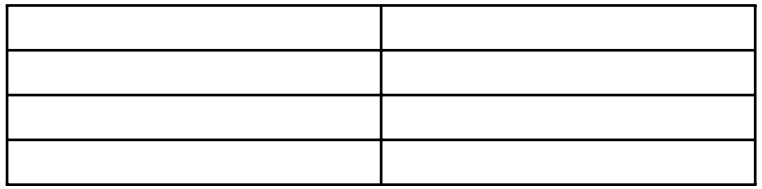
- The importance of memory hierarchy has increased dramatically with advances in processor performance. As shown in this graph, the gap between processor memory requests and DRAM access latency has grown exponentially since 1980.
- Modern high-end processors like the Intel Core i7 with four cores at 3.2 GHz can generate a peak bandwidth demand of 409.6 GB/sec, while DRAM main memory can only provide about 25 GB/sec (just 6% of the peak demand).

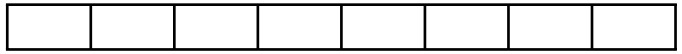
Cache Organization

Set Associativity

- A key design decision is where blocks can be placed in a cache:
  - **Direct-mapped:** One block per set (fixed location)
  - **N-way set associative:** N blocks per set
  - **Fully associative:** One set (block can go anywhere)







Write Strategies

- **Write-through:** Updates both cache and memory
- **Write-back:** Updates only cache; writes to memory when block is replaced
- **Write-allocate**: Allocates a cache block for write requests

- A write buffer can reduce write latency.

Cache Misses

Compulsory Misses

The very first access to a block cannot be in the cache, so the block must be brought into the cache. These misses occur even with infinite cache size.

Capacity Misses

If the cache cannot contain all blocks needed during program execution, capacity misses occur because blocks are discarded and later retrieved.

Conflict Misses

If the block placement strategy is not fully associative, conflict misses occur when multiple blocks map to the same set and accesses to different blocks are intermingled.

Multithreading and multiple cores add a fourth C: **Coherency misses** due to cache flushes to keep multiple caches coherent in a multiprocessor system.

Measuring Cache Performance

- $Miss\ rate = Number\ of\ misses\ /\ Number\ of\ accesses$
- $Average\ memory\ access\ time = Hit\ time + Miss\ rate \times Miss\ penalty$
- $Average\ memory\ access\ time\ (multilevel\ caches) = Hit\ time_{L1} + Miss\ rate_{L1} \times (Hit\ time_{L2} + Miss\ rate_{L2} \times Miss\ Penalty_{L2})$

Six Basic Cache Optimizations

1. **Larger Block Size**
   Reduces compulsory misses by exploiting spatial locality, but increases miss penalty and may increase conflict misses.

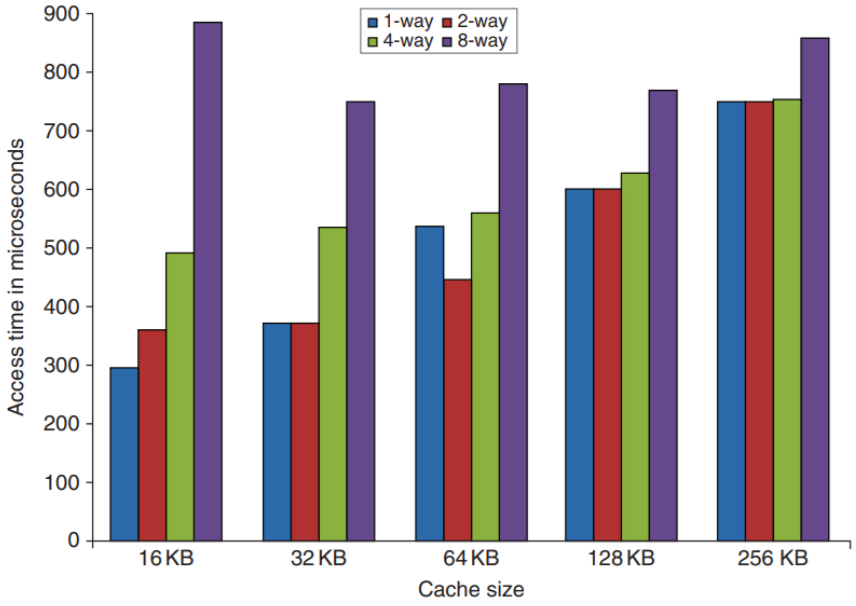
2. **Bigger Caches**
   Reduces capacity misses but increases hit time, cost, and power consumption.

3. **Higher Associativity**
   Reduces conflict misses but may increase hit time and power consumption.

4. **Multilevel Caches**
   Allows small L1 cache with fast hit time while larger L2/L3 caches capture memory accesses that would go to main memory.

5. **Read Priority**
   Giving priority to read misses over writes reduces miss penalty by using write buffers effectively.

6. **Virtual Indexing**
   Avoiding address translation during cache indexing reduces hit time by using page offset to index the cache.



Access times generally increase as cache size and associativity are Increased due to the increased interconnect delays, cost of tag checks and multiplexing