Computer Architecture
ELE 475 / COS 475
Slide Deck 3: Cache Review

David Wentzlaff

Department of Electrical Engineering

Princeton University

PRINCETON UNIVERSITY

PRINCETON
School of Engineering and Applied Science

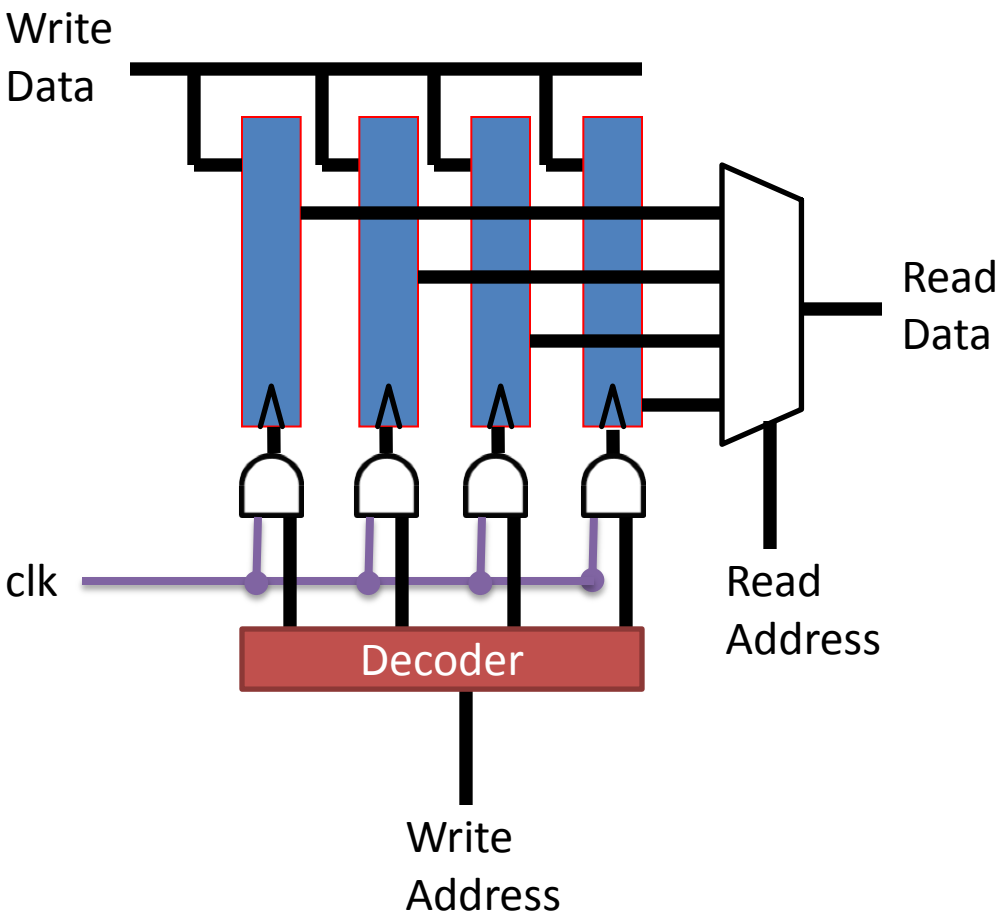
Agenda

- Memory Technology
- Motivation for Caches
- Classifying Caches
- Cache Performance

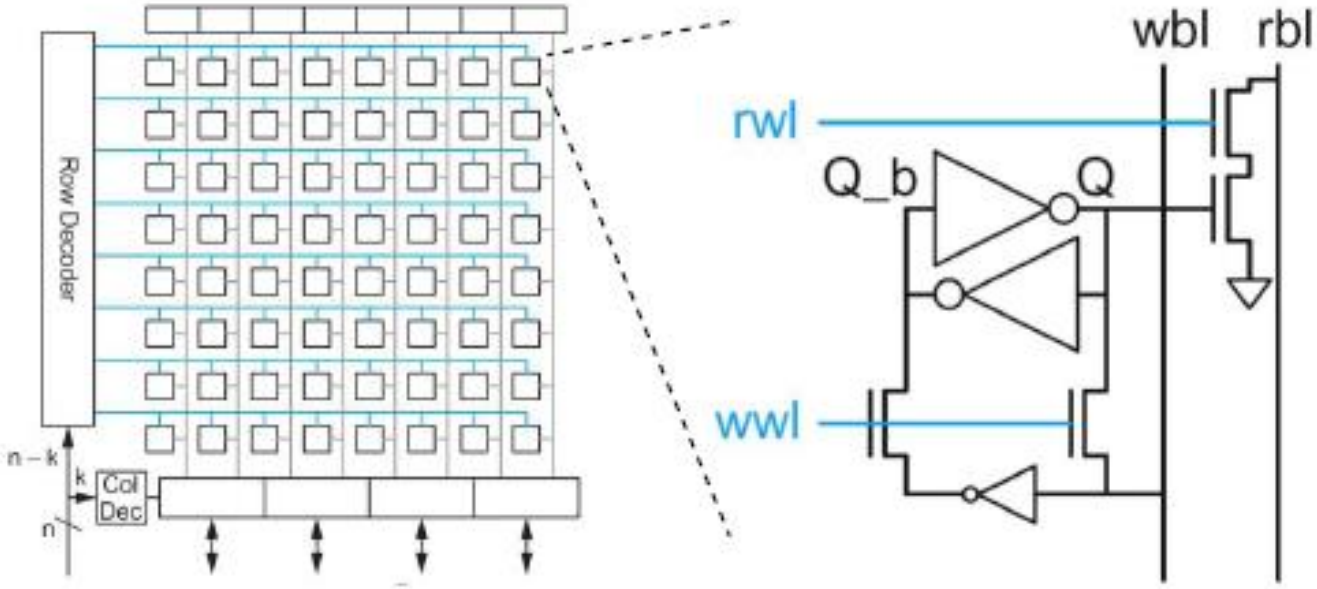
Agenda

- Memory Technology
- Motivation for Caches
- Classifying Caches
- Cache Performance

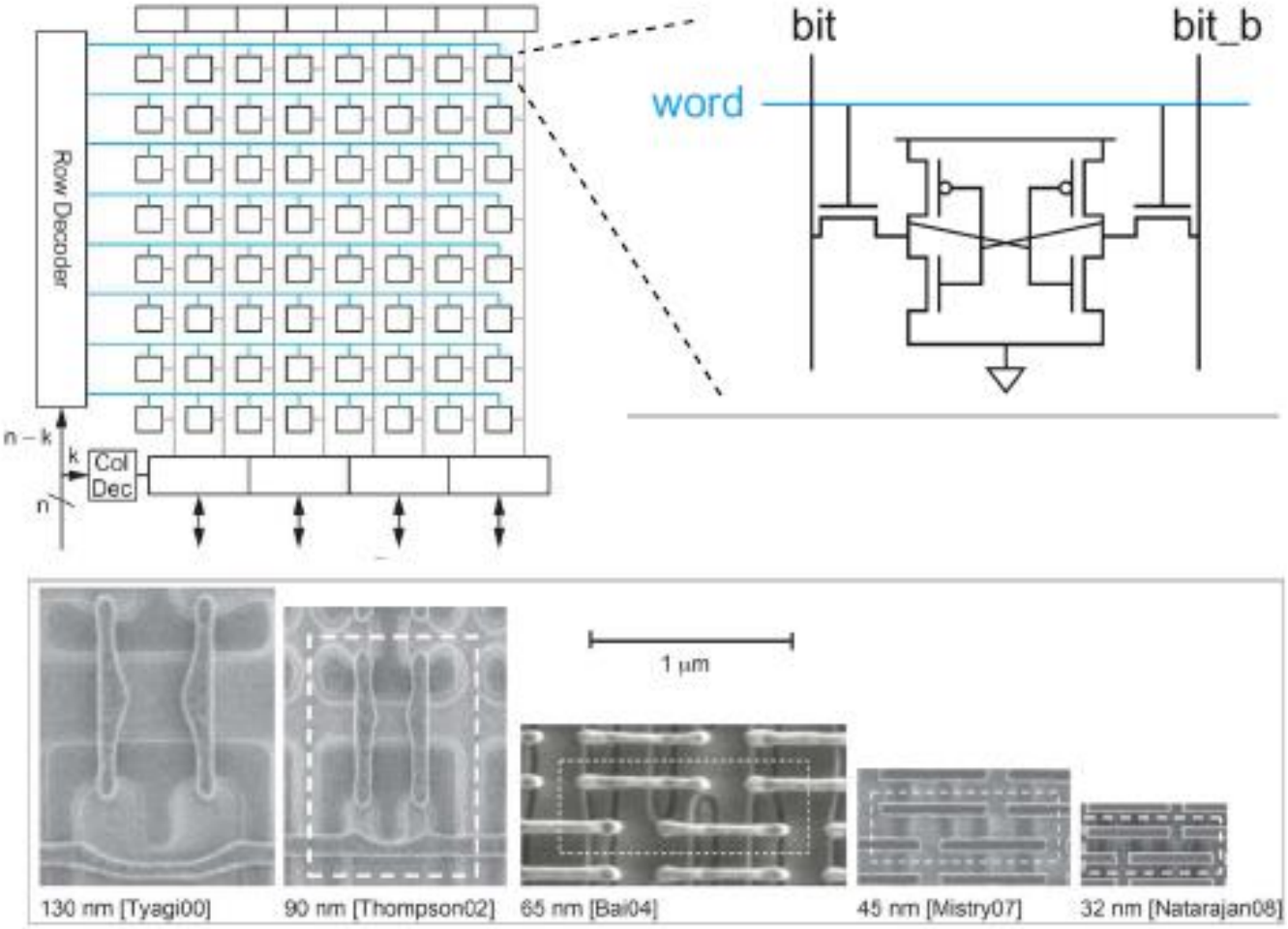
Naive Register File



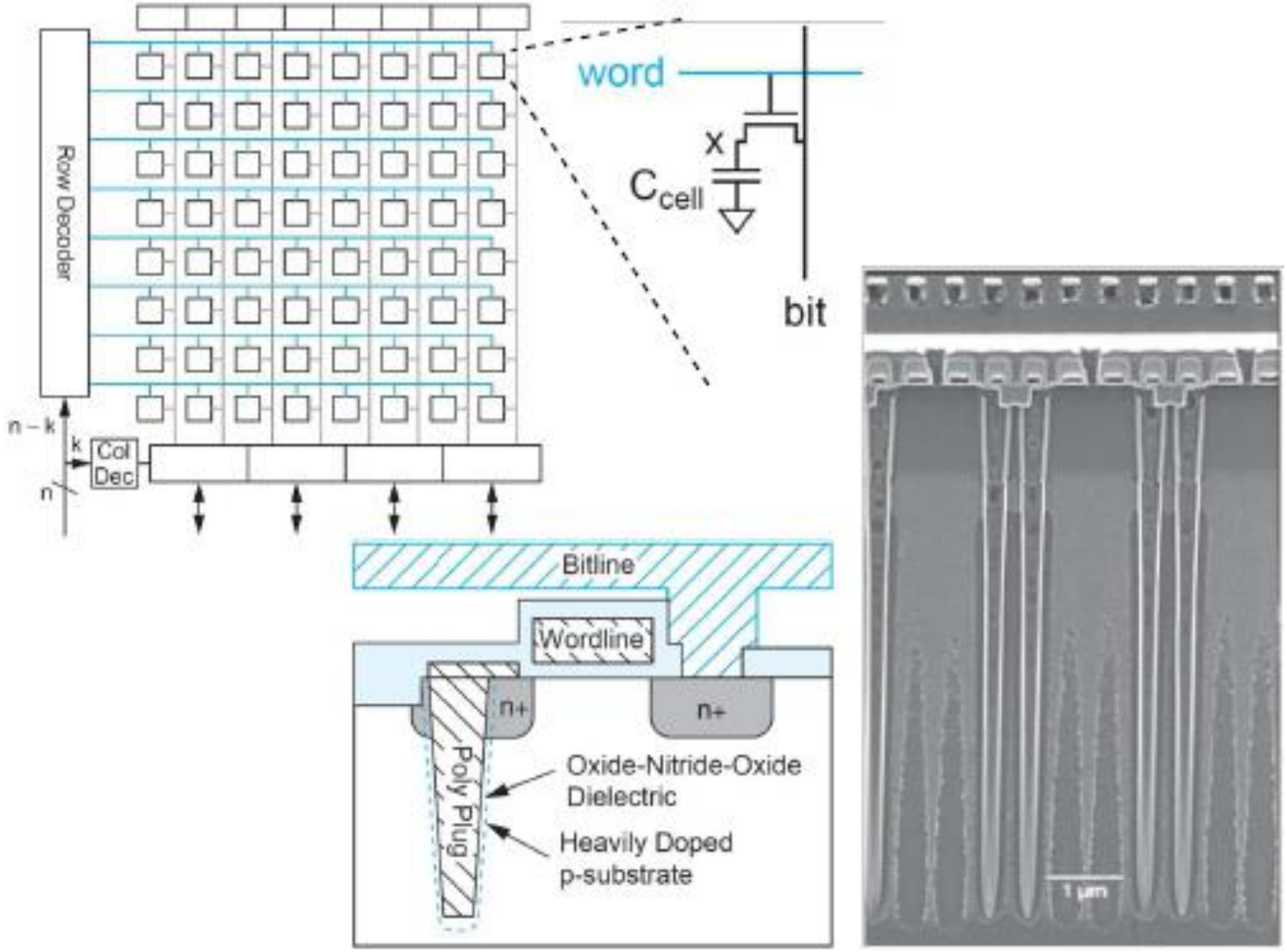
Memory Arrays: Register File



Memory Arrays: SRAM

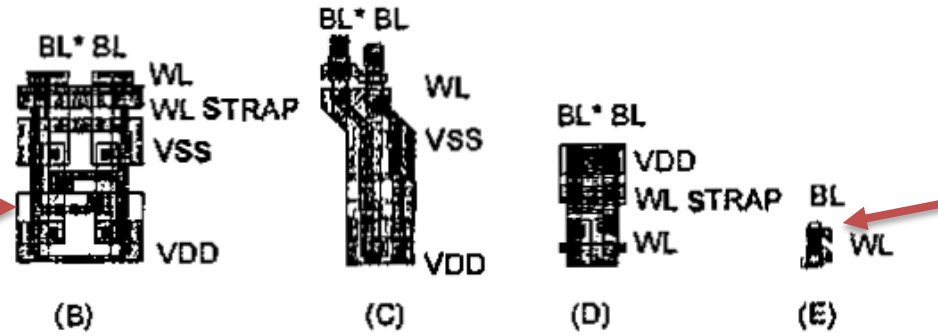


Memory Arrays: DRAM

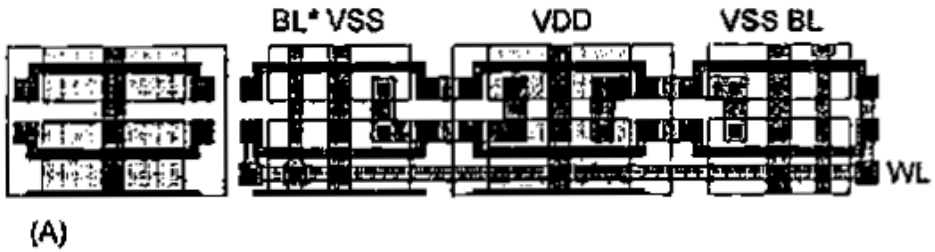


Relative Memory Sizes of SRAM vs. DRAM

On-Chip SRAM on logic chip



DRAM on memory chip



1 Memory cell in 0.5µm processes

a) Gate Array SRAM
b) Embedded SRAM
c) Standard SRAM (6T cell with local interconnect)
d) ASIC DRAM
e) Standard DRAM (stacked cell)

[From Foss, R.C. "Implementing Application-Specific Memory", ISSCC 1996]

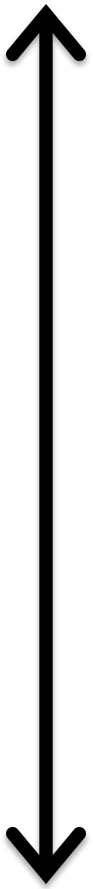
Memory Technology Trade-offs

Latches/Registers

Register File

SRAM

DRAM



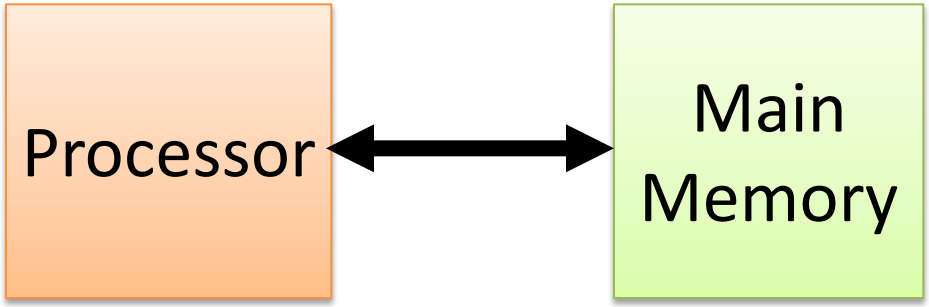
Low Capacity
Low Latency
High Bandwidth
(more and wider ports)

High Capacity
High Latency
Low Bandwidth

Agenda

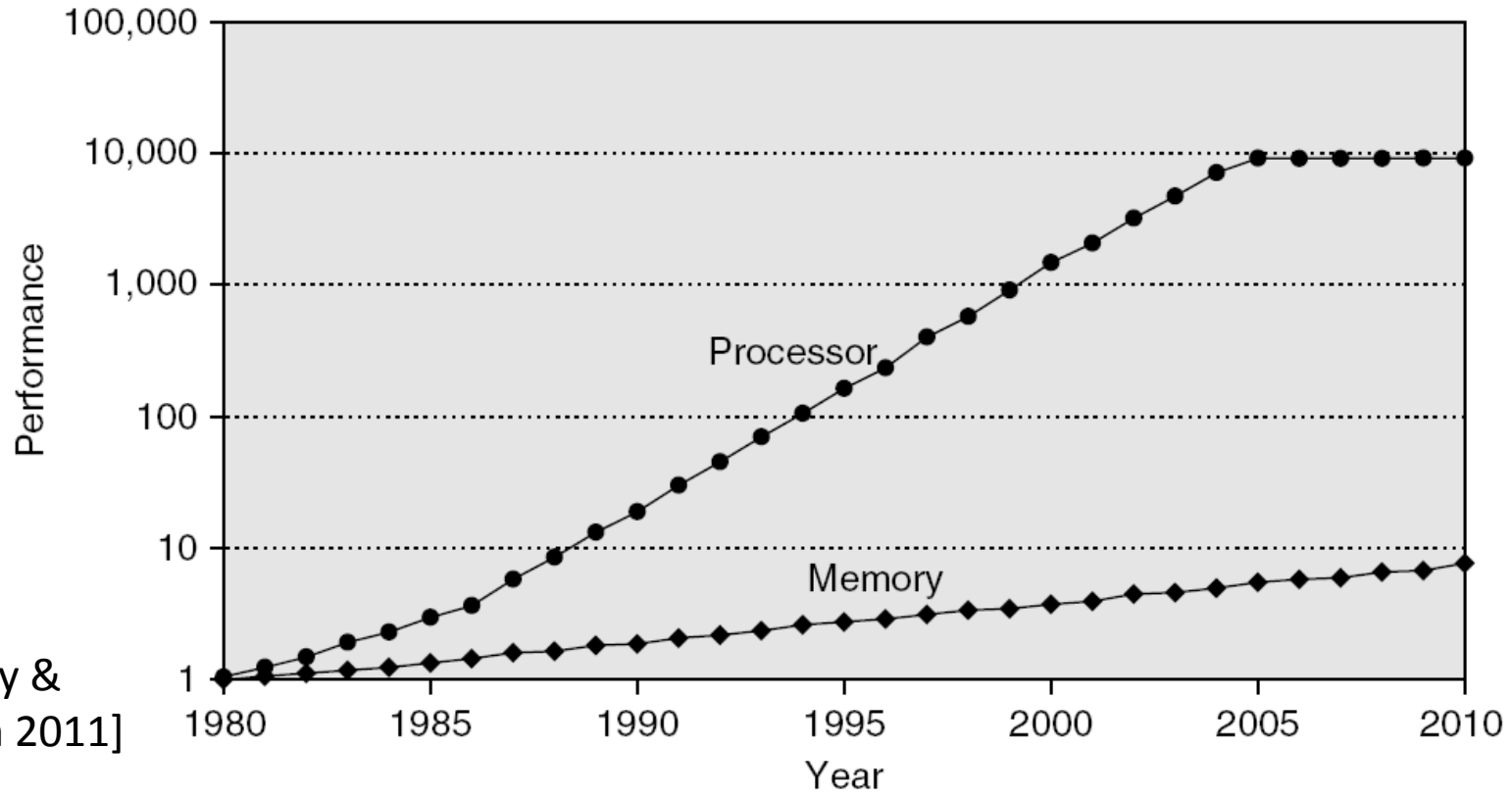
- Memory Technology
- Motivation for Caches
- Classifying Caches
- Cache Performance

CPU-Memory Bottleneck



- Performance of high-speed computers is usually limited by memory bandwidth and latency
- Latency is time for a single access
  - Main memory latency is usually >> than processor cycle time
- Bandwidth is the number of accesses per unit time
  - If m instructions are loads/stores, 1 + m memory accesses per instruction, CPI = 1 requires at least 1 + m memory accesses per cycle
- Bandwidth-Delay Product is amount of data that can be in flight at the same time (Little's Law)

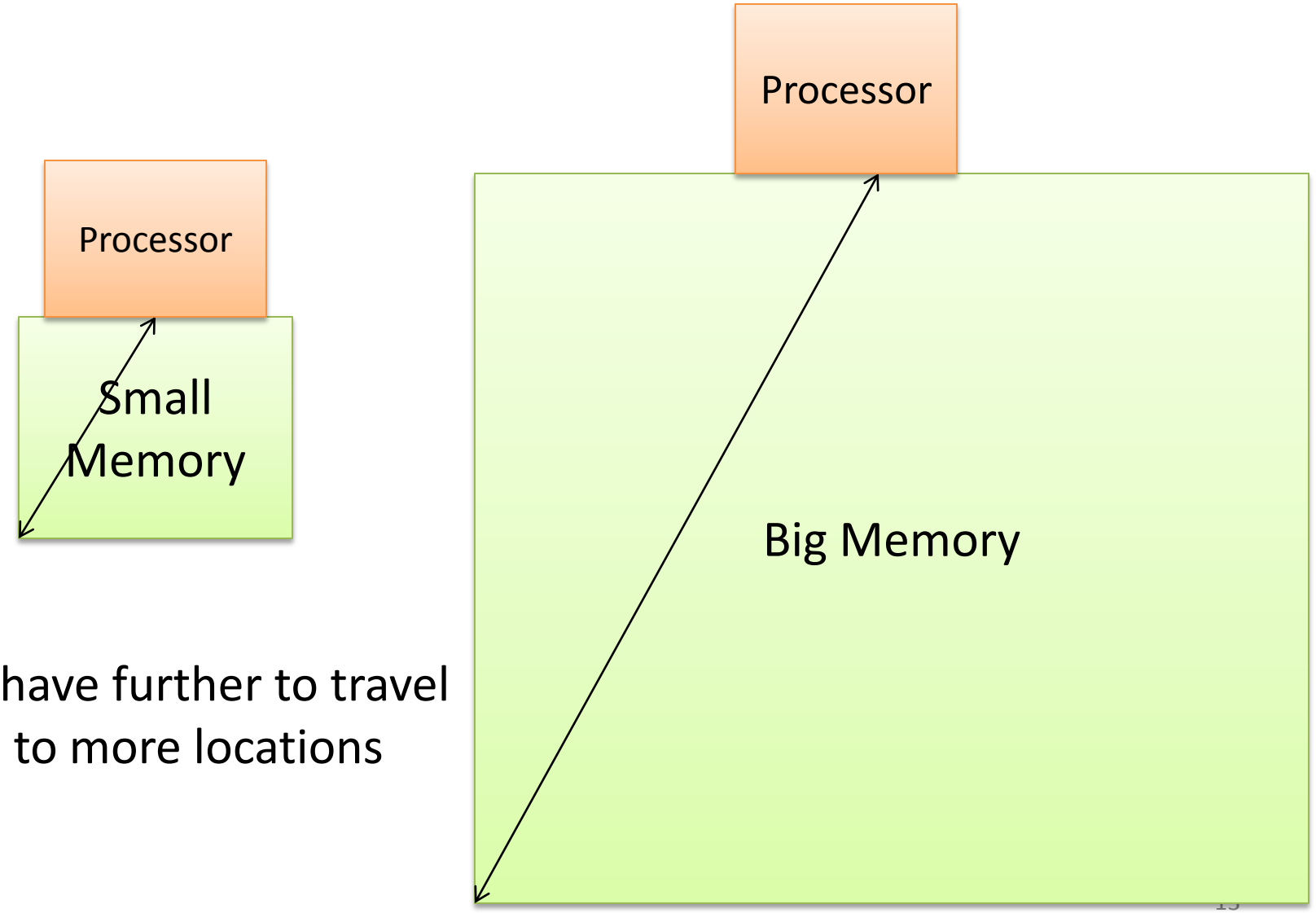
Processor-DRAM Latency Gap



[Hennessy & Patterson 2011]

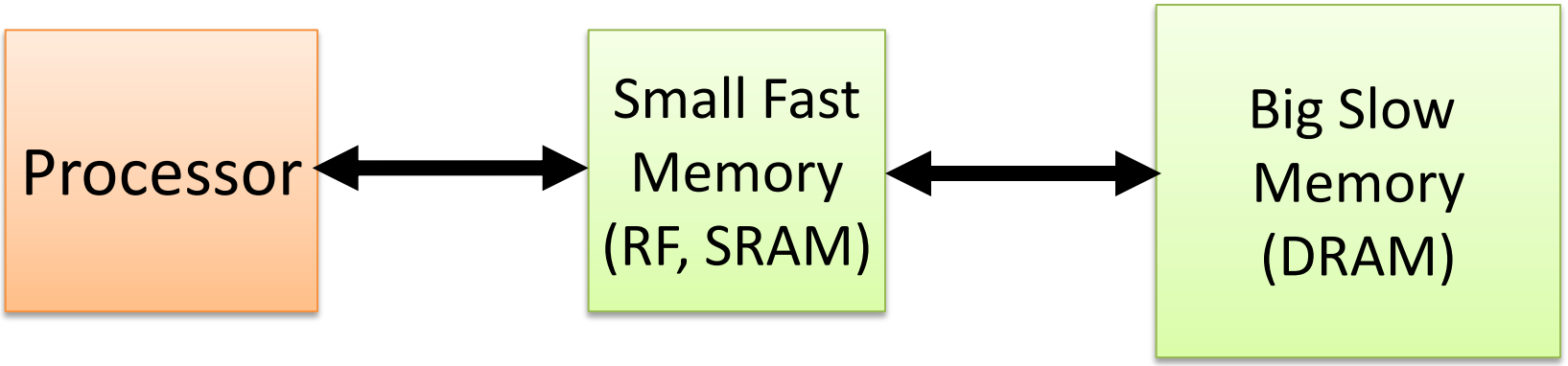
- Four-issue 2 GHz superscalar accessing 100 ns DRAM could execute 800 instructions during the time for one memory access!
- Long latencies mean large bandwidth-delay products which can be difficult to saturate, meaning bandwidth is wasted

Physical Size Affects Latency



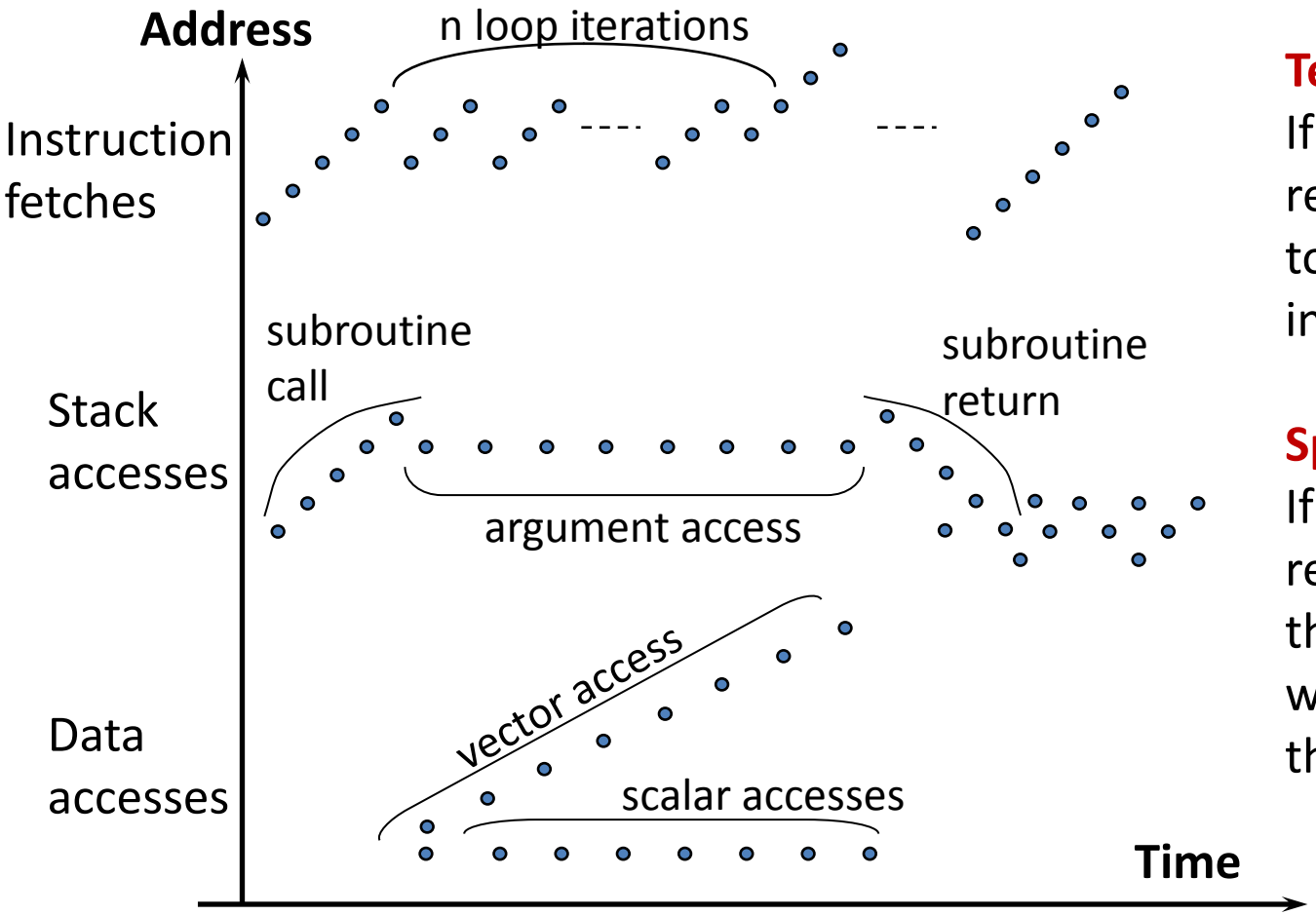
- Signals have further to travel
- Fan out to more locations

Memory Hierarchy



- Capacity: Register << SRAM << DRAM
- Latency: Register << SRAM << DRAM
- Bandwidth: on-chip >> off-chip
- On a data access:
  - if data is in fast memory -> low-latency access to SRAM
  - if data is not in fast memory -> long-latency access to DRAM
- Memory hierarchies only work if the small, fast memory actually stores data that is reused by the processor

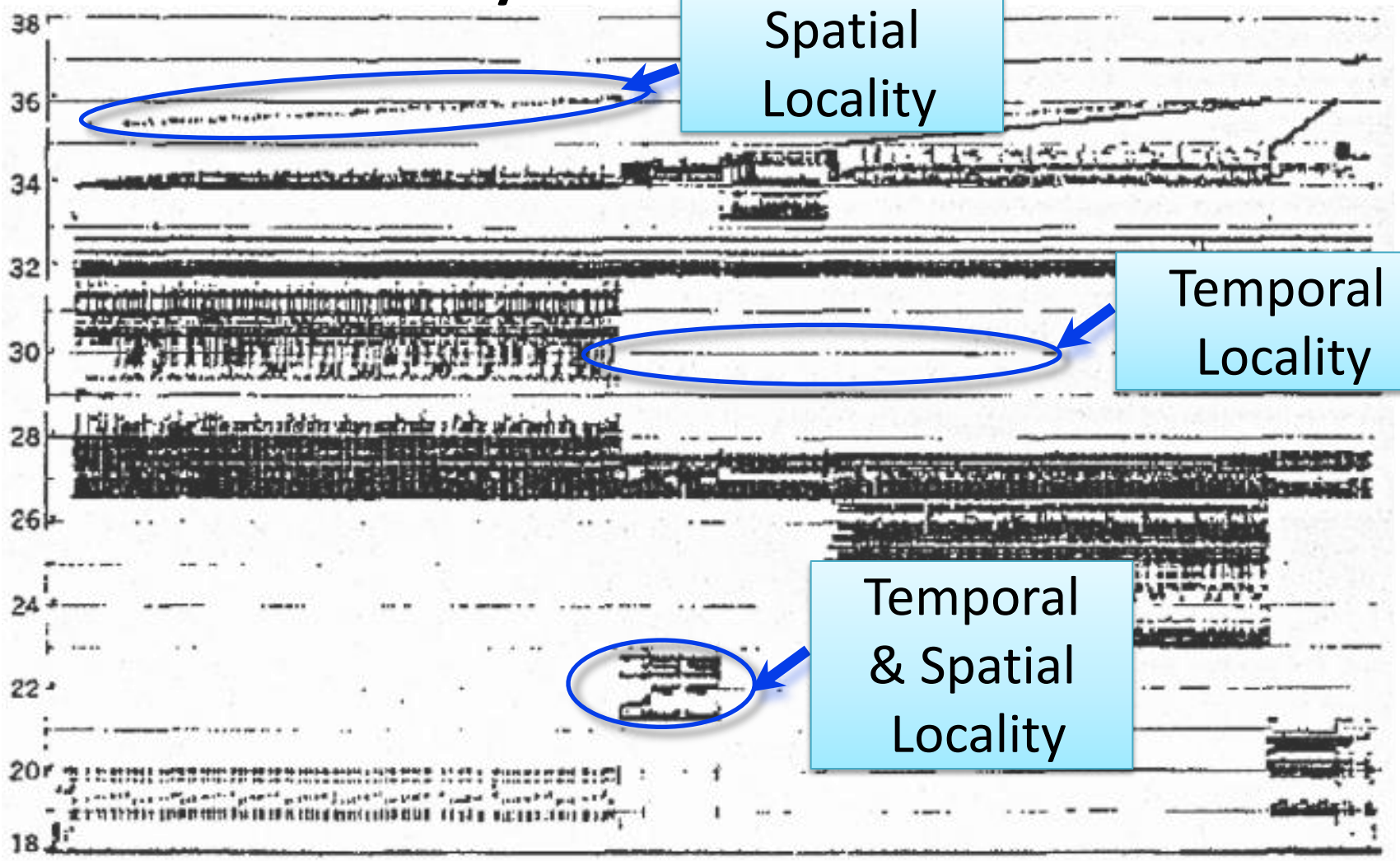
Common And Predictable Memory Reference Patterns



Temporal Locality:
If a location is reference it is likely to be reference again in the near future

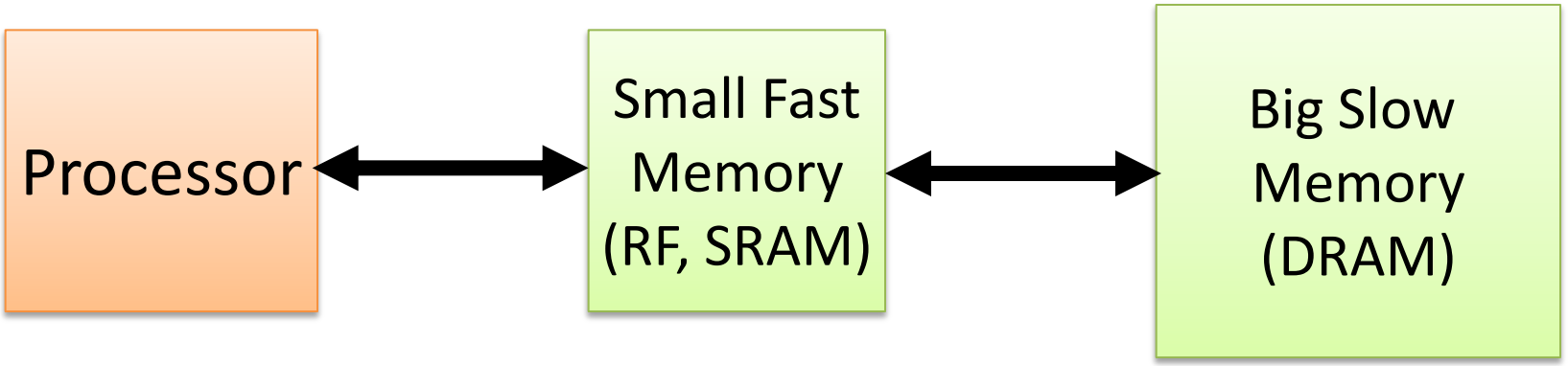
Spatial Locality:
If a location is referenced it is likely that locations near it will be referenced in the near future

Real Memory Reference Patterns



[From Donald J. Hatfield, Jeanette Gerald: Program Restructuring for Virtual Memory. IBM Systems Journal 10(3): 168-192 (1971)]

Caches Exploit Both Types of Locality

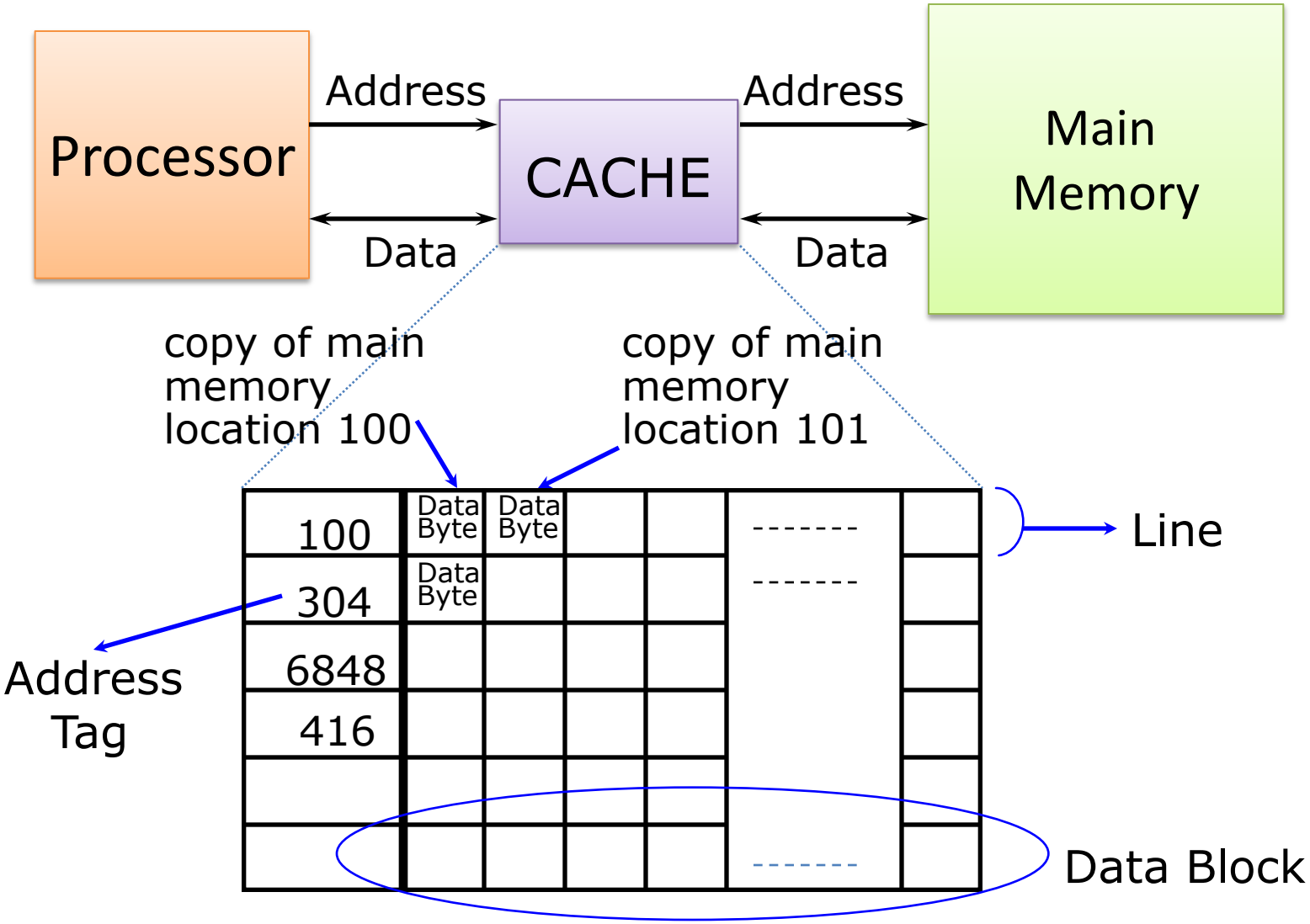


- Exploit temporal locality by remembering the contents of recently accessed locations
- Exploit spatial locality by fetching blocks of data around recently accessed locations

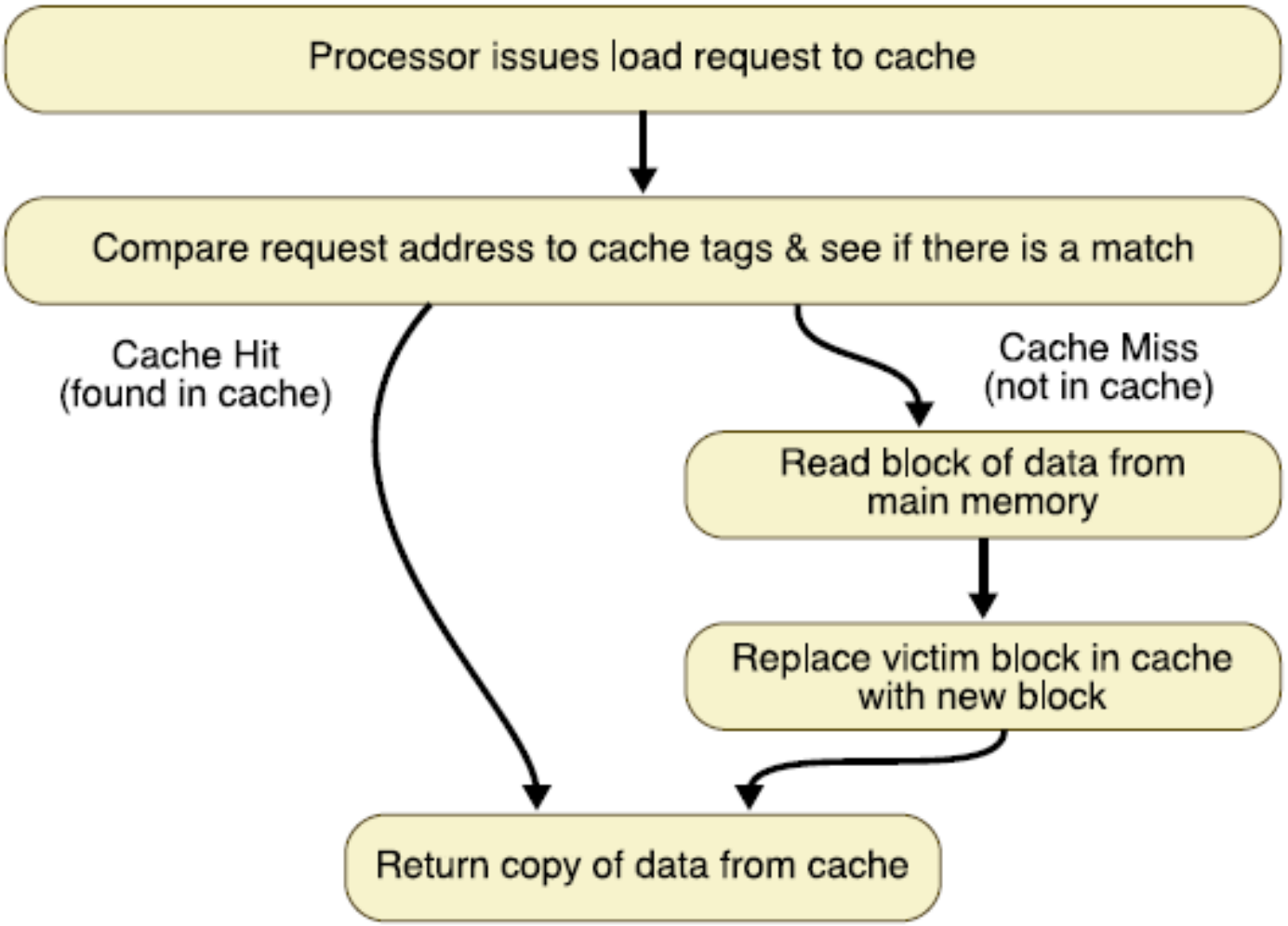
Agenda

- Memory Technology
- Motivation for Caches
- Classifying Caches
- Cache Performance

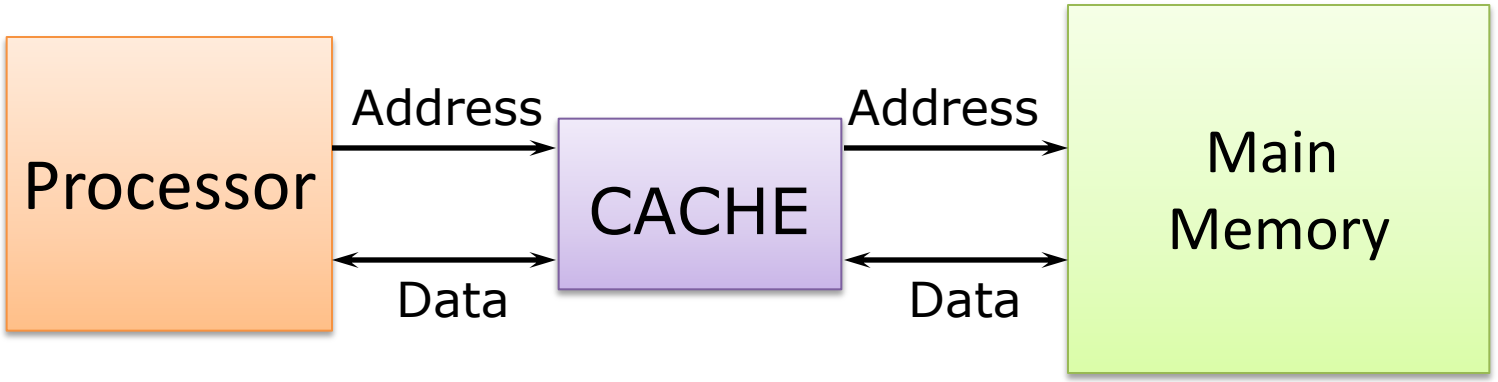
Inside a Cache



Basic Cache Algorithm for a Load

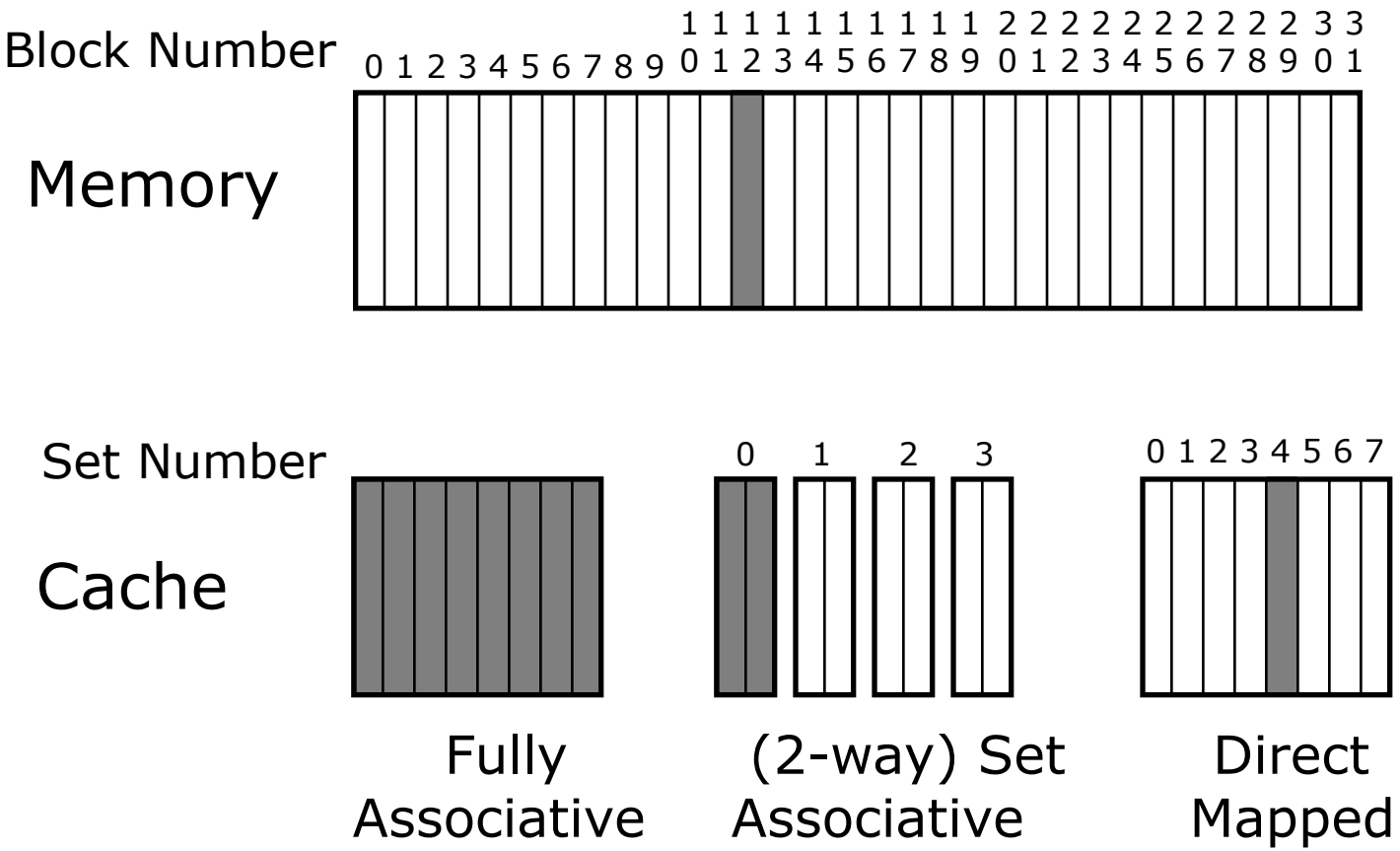


Classifying Caches



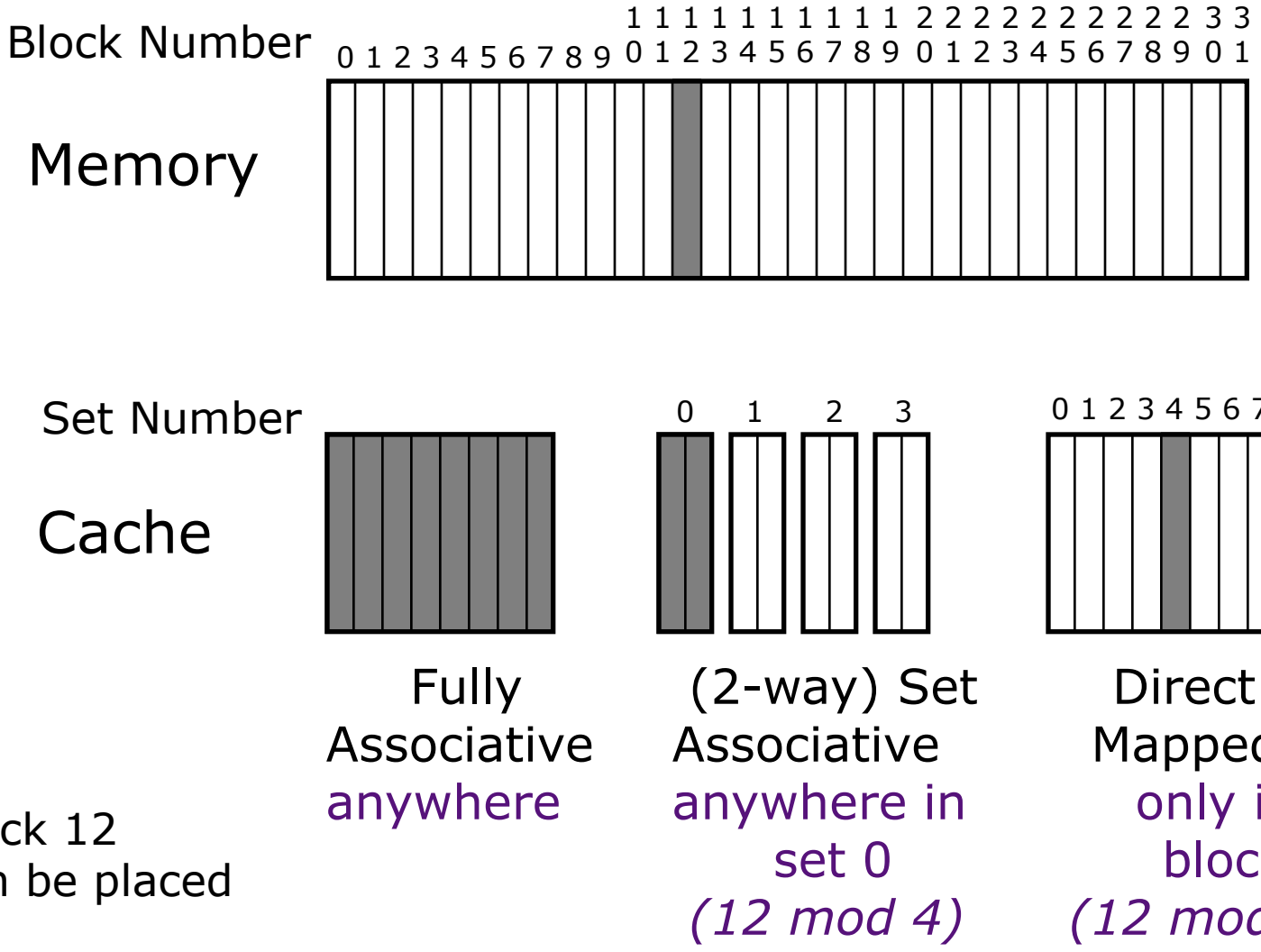
- **Block Placement**: Where can a block be placed in the cache?
- **Block Identification**: How a block is found if it is in the cache?
- **Block Replacement**: Which block should be replaced on a miss?
- **Write Strategy**: What happens on a write?

Block Placement: Where Place Block in Cache?

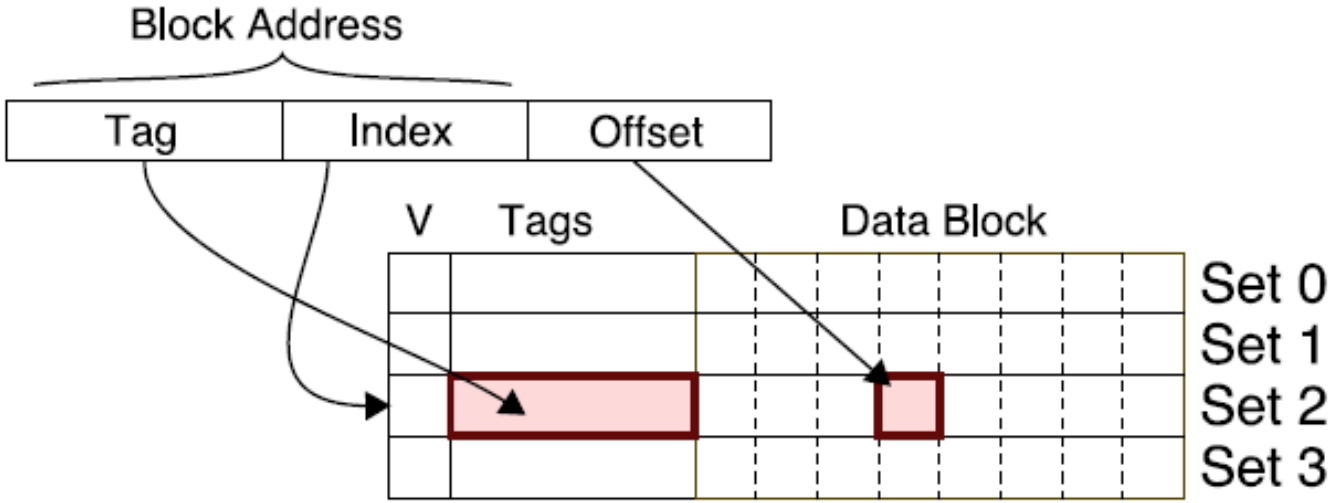


block 12
can be placed

Block Placement: Where Place Block in Cache?

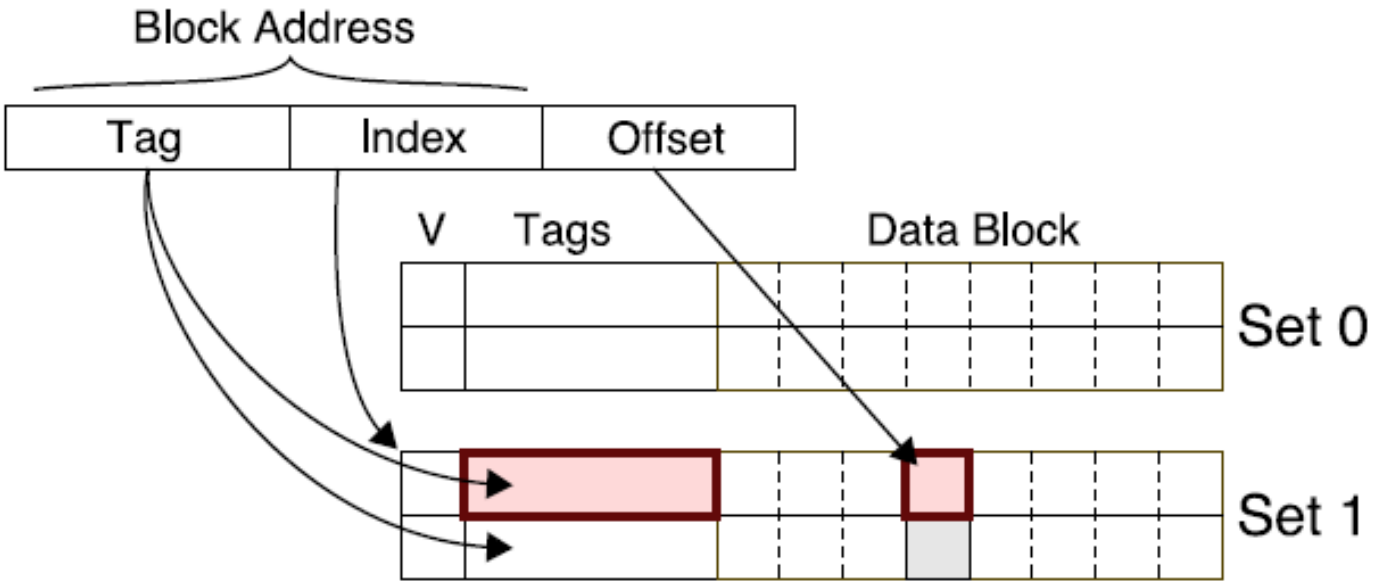


Block Identification: How to find block in cache?



- Cache uses index and offset to find potential match, then checks tag
- Tag check only includes higher order bits
- In this example (Direct-mapped, 8B block, 4 line cache)

Block Identification: How to find block in cache?



- Cache checks all potential blocks with parallel tag check
- In this example (2-way associative, 8B block, 4 line cache)

Block Replacement: Which block to replace?

- No choice in a direct mapped cache
- In an associative cache, which block from set should be evicted when the set becomes full?
- **Random**
- **Least Recently Used (LRU)**
  - LRU cache state must be updated on every access
  - True implementation only feasible for small sets (2-way)
  - Pseudo-LRU binary tree often used for 4-8 way
- **First In, First Out (FIFO) aka Round-Robin**
  - Used in highly associative caches
- **Not Most Recently Used (NMRU)**
  - FIFO with exception for most recently used block(s)

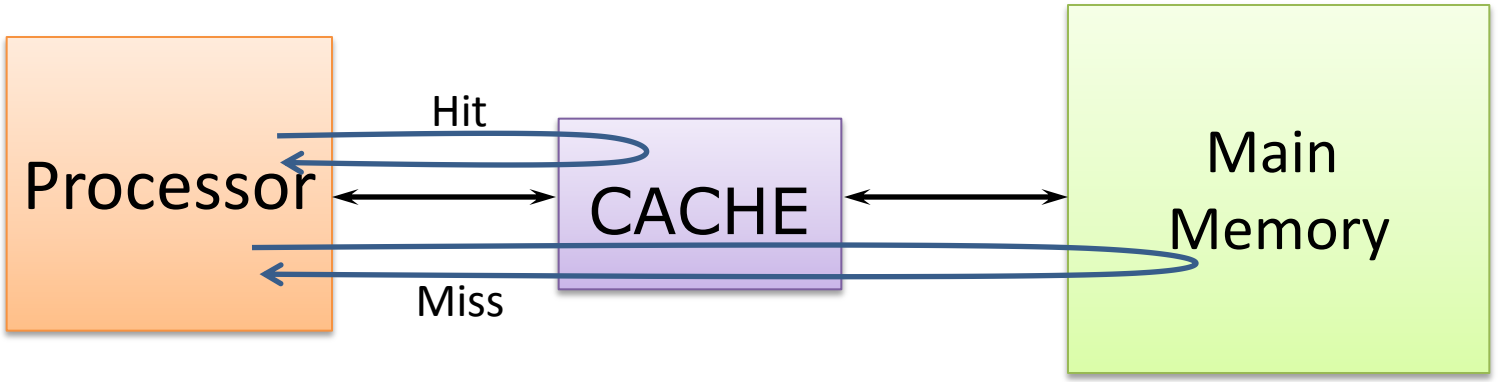
Write Strategy: How are writes handled?

- Cache Hit
  - **Write Through –** write both cache and memory, generally higher traffic but simpler to design
  - **Write Back –** write cache only, memory is written when evicted, dirty bit per block avoids unnecessary write backs, more complicated
- Cache Miss
  - **No Write Allocate –** only write to main memory
  - **Write Allocate –** fetch block into cache, then write
- Common Combinations
- Write Through & No Write Allocate
- Write Back & Write Allocate

Agenda

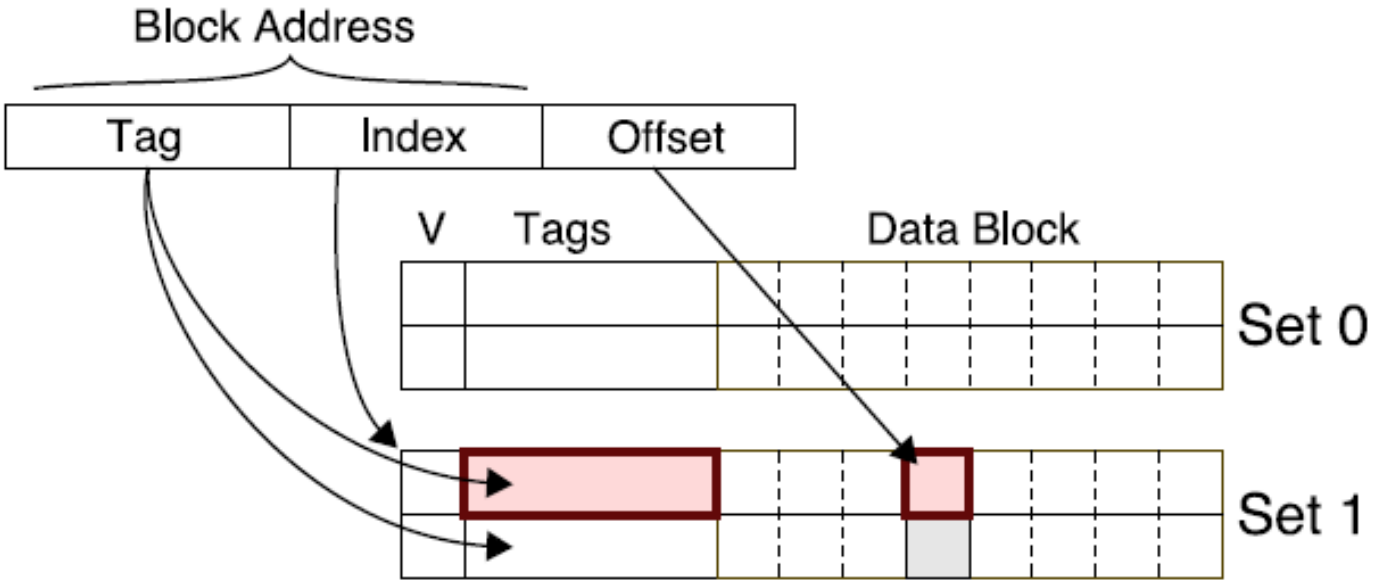
- Memory Technology
- Motivation for Caches
- Classifying Caches
- Cache Performance

Average Memory Access Time



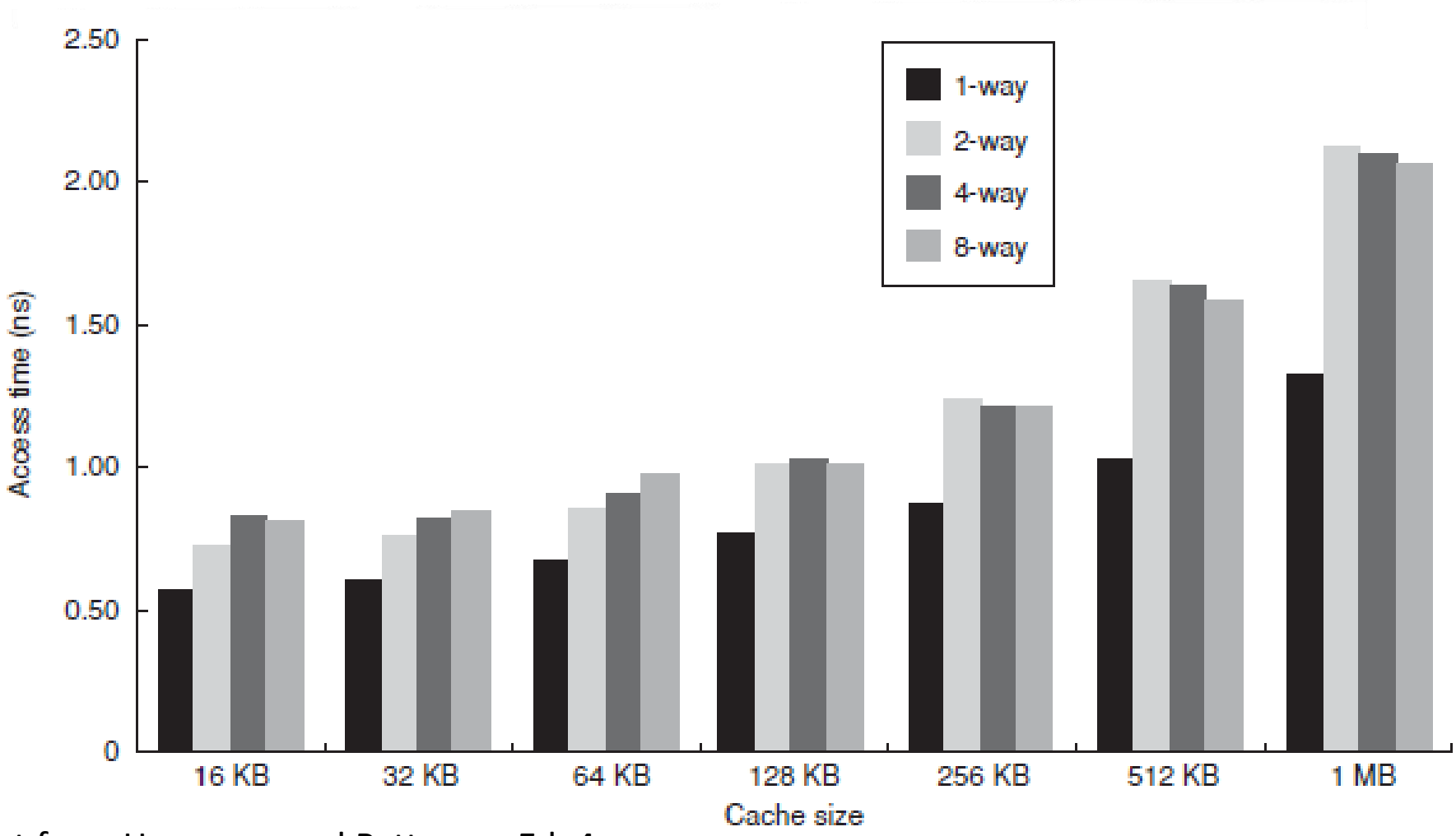
- Average Memory Access Time = Hit Time + (Miss Rate * Miss Penalty)

Categorizing Misses: The Three C's



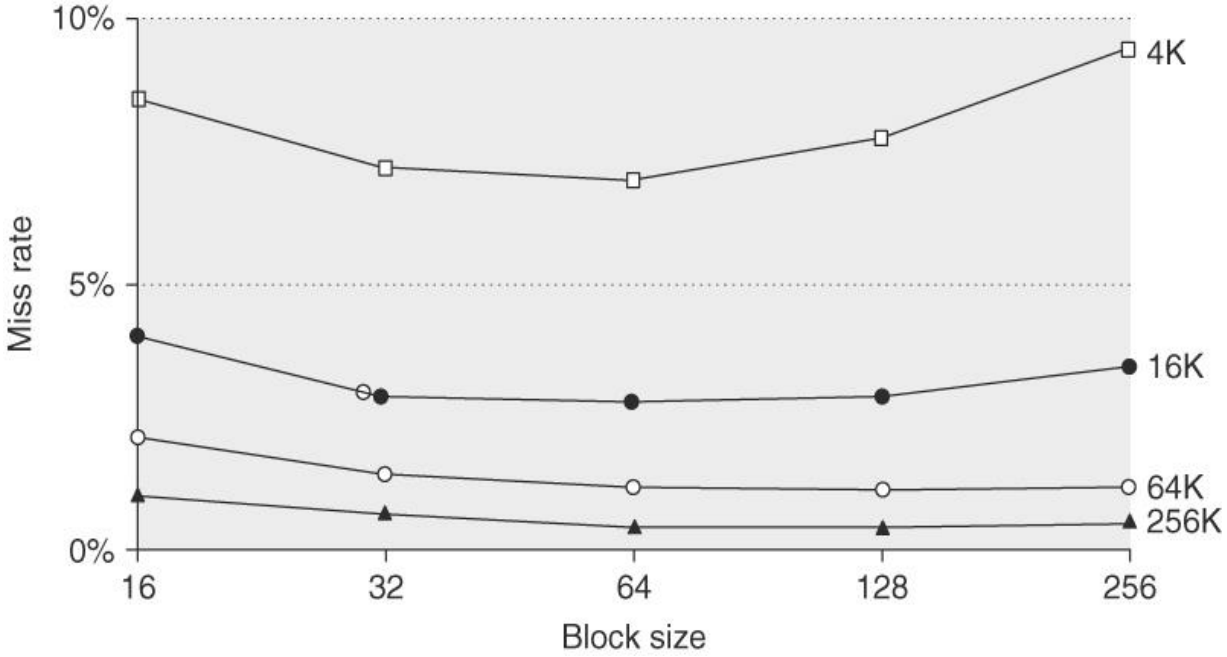
- **Compulsory** – first-reference to a block, occur even with infinite cache
- **Capacity** – cache is too small to hold all data needed by program, occur even under perfect replacement policy (loop over 5 cache lines)
- **Conflict** – misses that occur because of collisions due to less than full associativity (loop over 3 cache lines)

Reduce Hit Time: Small & Simple Caches



Plot from Hennessy and Patterson Ed. 4

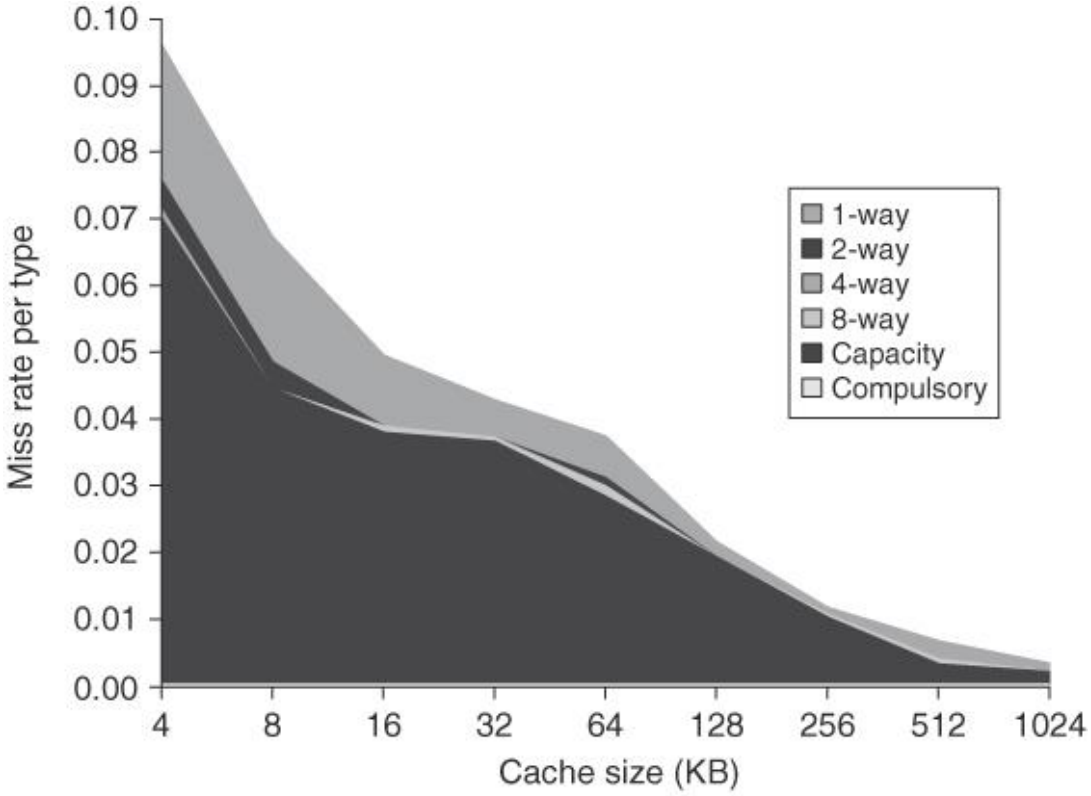
Reduce Miss Rate: Large Block Size



- Less tag overhead
- Exploit fast burst transfers from DRAM
- Exploit fast burst transfers over wide on-chip busses

- Can waste bandwidth if data is not used
- Fewer blocks -> more conflicts

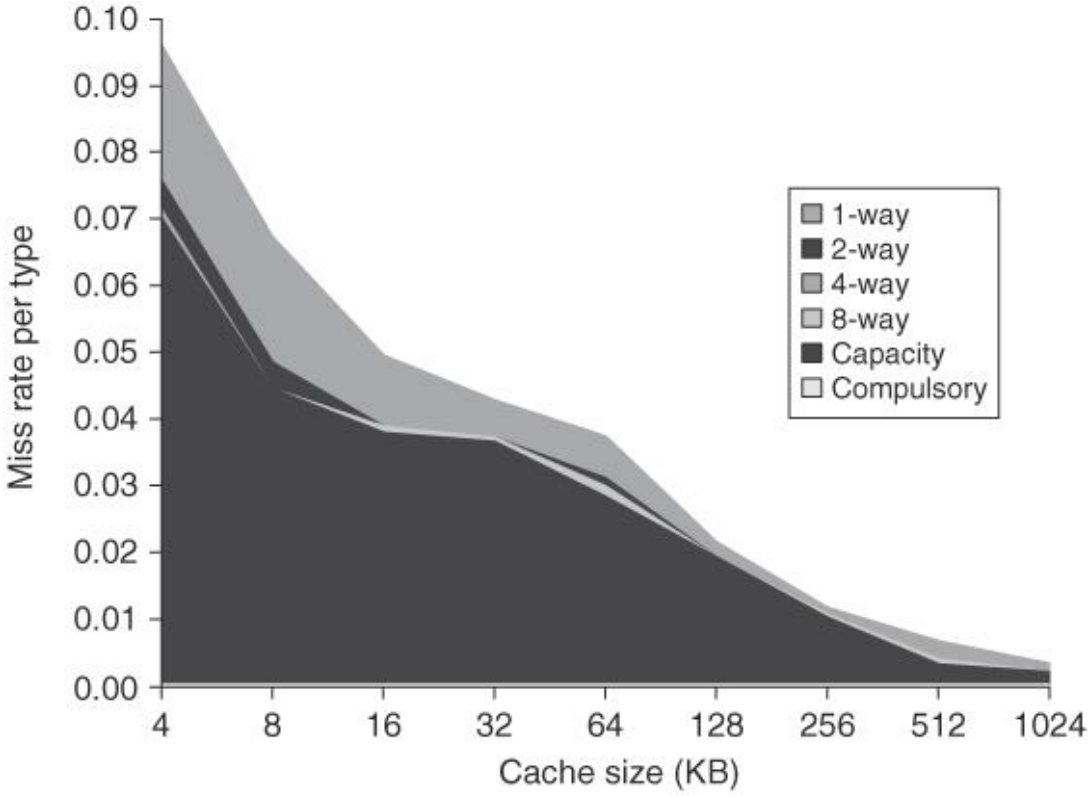
Reduce Miss Rate: Large Cache Size



Empirical Rule of Thumb:

If cache size is doubled, miss rate usually drops by about $\sqrt{2}$

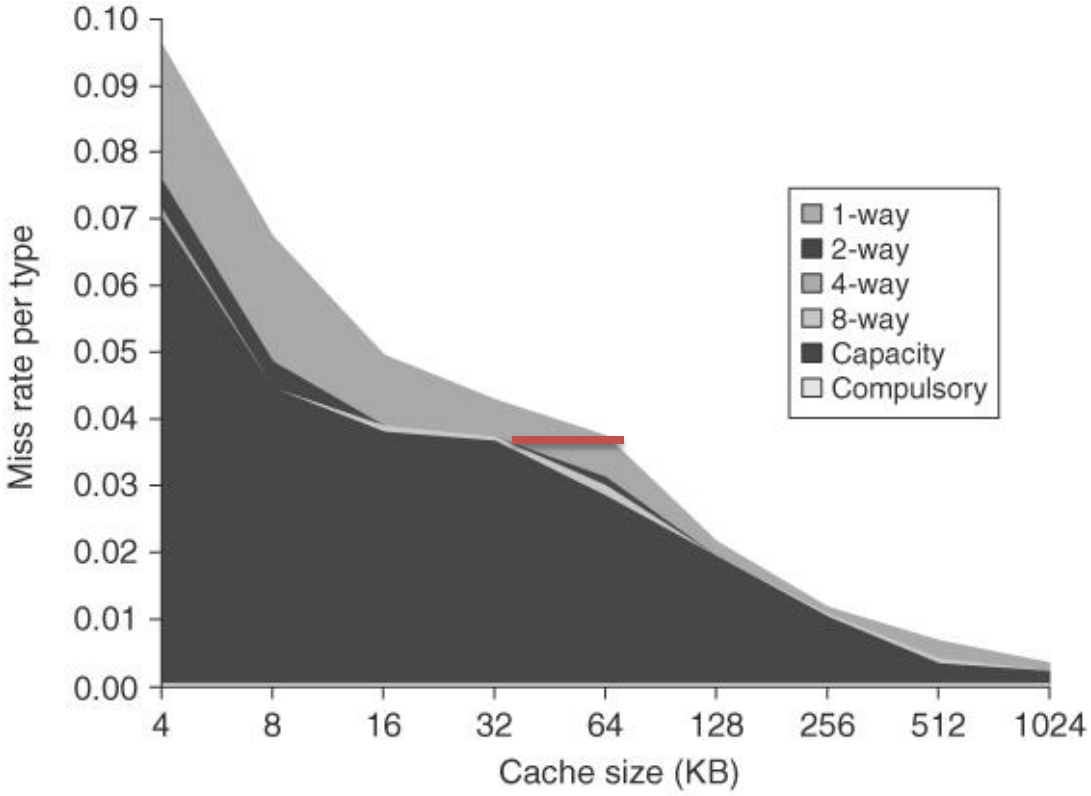
Reduce Miss Rate: High Associativity



Empirical Rule of Thumb:

Direct-mapped cache of size N has about the same miss rate as a two-way set- associative cache of size N/2

Reduce Miss Rate: High Associativity



Empirical Rule of Thumb:

Direct-mapped cache of size N has about the same miss rate as a two-way set- associative cache of size N/2

Agenda

- Memory Technology
- Motivation for Caches
- Classifying Caches
- Cache Performance

Acknowledgements

- These slides contain material developed and copyright by:
  - Arvind (MIT)
  - Krste Asanovic (MIT/UCB)
  - Joel Emer (Intel/MIT)
  - James Hoe (CMU)
  - John Kubiatowicz (UCB)
  - David Patterson (UCB)
  - Christopher Batten (Cornell)

- MIT material derived from course 6.823
- UCB material derived from course CS252 & CS152
- Cornell material derived from course ECE 4750