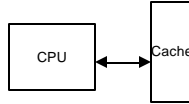


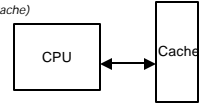
## Improving Cache Performance

- Average memory-access time = Hit time + Miss rate x Miss penalty
- Improve performance by:
  1. Reduce the miss rate,
  2. Reduce the miss penalty, or
  3. Reduce the time to hit in the cache.
  4. Increase ILP

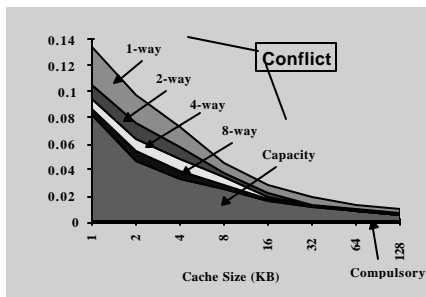


## A Closer Look at Misses

- Classifying Misses: 3 Cs
  - **Compulsory**—The first access to a block is not in the cache, so the block must be brought into the cache. These are also called *cold start misses* or *first reference misses* (*Misses in Infinite Cache*)
  - **Capacity**—If the cache cannot contain all the blocks needed during execution of a program, capacity misses will occur due to blocks being discarded and later retrieved. (*Misses in Size X Cache*)
  - **Conflict**—If the block-placement strategy is set associative or direct mapped, conflict misses (in addition to compulsory and capacity misses) will occur because a block can be discarded and later retrieved if too many blocks map to its set. These are also called *collision misses* or *interference misses* (*Misses in N-way Associative, Size X Cache*)



## 3Cs Absolute Miss Rate

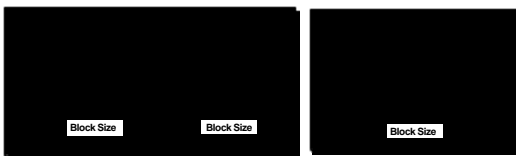


## How Can Reduce the Miss Rate?

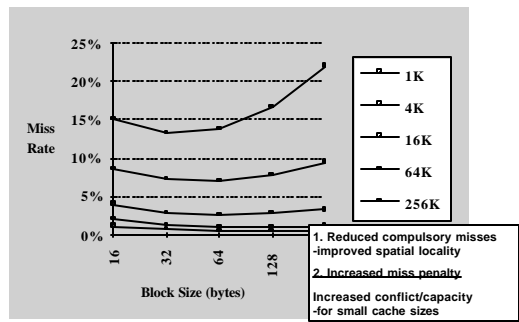
- Block Size?
- Cache Size?
- Associativity?
- Prediction?
- Compiler?

## Block Size vs. Cache Measures

- Increasing Block Size generally increases Miss Penalty and decreases Miss Rate



## Reduce Misses via Larger Block Size



## Example: Optimal block size

- Memory system: 40 cycles overhead, hit time = 1 cycle
- 16 bytes / 2 clocks → 16 bytes in 42 cycles, 32 bytes in 44 cycles, ...
- For chart below, which block size has the minimum average memory access time for each cache size?

Block size (bytes)	Miss rates by cache size				
	1K	4K	16K	64K	256K
16	15.05%	8.57%	3.94%	2.04%	1.09%
32	13.34%	7.24%	2.87%	1.35%	.70%
64	13.76%	7.00%	2.64%	1.06%	.51%
128	16.64%	7.78%	2.77%	1.02%	.49%
256	22.01%	9.51%	3.29%	1.15%	.49%

## Example: Optimal block size

- Miss penalties: 40 cycles + 2 \* (block size / 16)
- AMAT = hit time + miss rate x miss penalty
  - 16KB cache, 16 byte block:  $1 + .0394 * 42 = 2.6548$
  - 16KB cache, 32 byte block:  $1 + .0287 * 44 = 2.2628$
  - 16KB cache, 64 byte block:  $1 + .0264 * 48 = 2.2672$
  - 16KB cache, 128 byte block:  $1 + .0277 * 56 = 2.5512$
  - 16KB cache, 256 byte block:  $1 + .0329 * 72 = 3.3688$

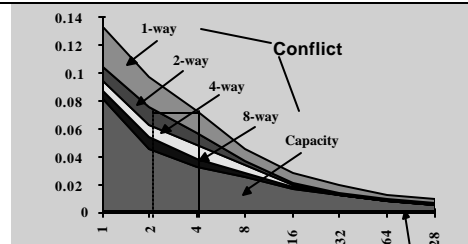
Block size (bytes)	Miss penalty	AMAT by cache size				
		1K	4K	16K	64K	256K
16	42	7.321	4.599	2.655	1.857	1.458
32	44	<u>6.870</u>	<u>4.186</u>	<u>2.263</u>	1.594	1.308
64	48	7.605	4.360	2.267	<u>1.509</u>	<u>1.245</u>
128	56	10.318	5.357	2.551	1.571	1.274
256	72	16.847	7.847	3.369	1.828	1.353

## Reduce Misses via Larger Caches

- Take advantage of technology (memory gap)
- Decrease capacity misses
- Increase hit time and cost

## General Rules of thumb

8-way set associative is practically as effective as fully associative



2:1 Cache Rule: Miss rate of a direct mapped cache of size n is about the same as a two-way set associative cache of size n/2 (i.e. miss rates similar for half sized cache with small increase (to two-way) in associativity)

## Reduce Misses via Higher Associativity

- Remember tradeoff:
  - Increase block size → reduced miss rate, increase miss penalty
- New tradeoff:
  - Higher associativity → reduced miss rate, increased hit time
- Beware: Execution time is only final measure!
  - Will Clock Cycle time increase?
  - Hill [1988] suggested hit time external cache +10%, internal + 2% for 2-way vs. 1-way

## Example: AMAT vs. miss rate

- Relative clock cycle hit time for different associativities
  - Cycle hit time for 2-way = 1.10 x 1-way cycle hit time
  - Cycle hit time for 4-way = 1.12 x 1-way cycle hit time
  - Cycle hit time for 8-way = 1.14 x 1-way cycle hit time
- Direct mapped (1-way) cycle hit time = 1 cycle
- Direct mapped (1-way) miss penalty = 50 cycles
- Using data below, find cache sizes that make these statements true:
  - AMAT(8-way) < AMAT(4-way)
  - AMAT(4-way) < AMAT(2-way)
  - AMAT(2-way) < AMAT(1-way)

Degree Associative	Miss rates by cache size							
	1K	2K	4K	8K	16K	32K	64K	128K
1-way	.133	.098	.072	.046	.029	.020	.014	.010
2-way	.105	.076	.057	.038	.022	.014	.010	.007
4-way	.095	.064	.049	.035	.020	.013	.009	.006
8-way	.087	.054	.039	.029	.018	.013	.009	.006

## Example: AMAT vs. miss rate

- $AMAT_{n\text{-way}} = \text{hit time}_{n\text{-way}} + \text{miss rate}_{n\text{-way}} \times \text{miss penalty}_{n\text{-way}}$ 
  - 16KB cache, 1-way:  $1 + .029 \cdot 50 = 2.45$
  - 16KB cache, 2-way:  $1.10 + .022 \cdot 50 = 2.2$
  - 16KB cache, 4-way:  $1.12 + .020 \cdot 50 = 2.12$
  - 16KB cache, 8-way:  $1.14 + .018 \cdot 50 = 2.04$
- Results:
  - $AMAT(8\text{-way}) < AMAT(4\text{-way})$  for cache sizes  $< 32K$
  - $AMAT(4\text{-way}) < AMAT(2\text{-way})$  for all cache sizes
  - $AMAT(2\text{-way}) < AMAT(1\text{-way})$  for all cache sizes

Degree Associative	AMAT by cache size							
	1K	2K	4K	8K	16K	32K	64K	128K
1-way	7.65	5.90	4.60	3.30	2.45	2.00	1.70	1.50
2-way	6.60	4.90	3.95	3.00	2.20	1.80	1.60	1.45
4-way	6.22	4.62	3.57	2.87	2.12	1.77	1.57	1.42
8-way	5.44	4.09	3.19	2.59	2.04	<u>1.72</u>	<u>1.59</u>	<u>1.44</u>

## Reducing Misses via Way prediction and Pseudo-Associativity

- Problem
  - Direct mapped cache offers lowest hit time
  - 2-way set associative reduces conflict misses
- Solution
  - Way-prediction: extra bits predict which block to try on next access
  - Pseudo-associativity
    - Divide cache based on simple association (ex: inverse of MSB)
    - On miss check other half of cache (pseudo-hit)
- Complications
  - Way prediction use on Alpha 21264, MIPS R4300
  - Pipelining implementation difficult for varying hit times
  - Better suited for caches not tightly coupled to processor

