CSE331 - Computer Organization

Lecture 13: Calculating and Improving Cache Performance

Cache Performance Measures

- *Hit rate* : fraction found in the cache
  - So high that we usually talk about *Miss rate = 1 - Hit Rate*
- *Hit time* : time to access the cache
- *Miss penalty* : time to replace a block from lower level, including time to replace in CPU
  - *access time* : time to access lower level
  - *transfer time* : time to transfer block
- Average memory-access time (AMAT)

  = Hit time + Miss rate x Miss penalty (ns or clocks)

Measuring and Analyzing Cache Performance

- CPU time can be divided into two parts:

CPU time = (CPU execution clock cycles + Memory-stall clock cycles) x Clock cycle time

Memory-stall clock cycles = Read-stall cycles + Write-stall cycles

$$\text{Read-stall cycles} = \frac{\text{Reads}}{\text{Program}} \times \text{Read miss rate} \times \text{Read miss penalty}$$

$$\text{Write-stall cycles} = \left(\frac{\text{Writes}}{\text{Program}} \times \text{Write miss rate} \times \text{Write miss penalty}\right) + \text{Write buffer stall}$$

Can be ignored using reasonable write buffer

Measuring and Analyzing Cache Performance

- Read and Write stall cycles can be combined by using single miss rate and miss penalty:

$$\text{Memory-stall clock cycles} = \frac{\text{Memory accesses}}{\text{Program}} \times \text{Miss rate} \times \text{Miss penalty}$$

- It can also be written as :

$$\text{Memory-stall clock cycles} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Miss}}{\text{Instruction}} \times \text{Miss penalty}$$

Example: Cache Performance (page 565)

- Assume that:
  - Instruction miss rate %2
  - Data miss rate %4
  - CPI is 2 (without any memory stalls)
  - Miss penalty 40 cycles
  - %36 of instructions are load/store
- Determine how much faster a machine would run with a perfect cache that never missed.

Instruction miss cycles = I x 0.02 x 40 = 0.80 I (**I** is # of instructions)

Data miss cycles = I x 0.36 x 0.04 x 40 = 0.58 I

Total memory stall cycles = 0.80 I + 0.58 I = 1.38 I

$CPI_{stall} = 2 + 1.38 = 3.38$

$$\frac{\text{CPU time with stalls}}{\text{CPU time with perfect cache}} = \frac{I \times CPI_{stall} \times \text{Clock cycle}}{I \times CPI_{perfect} \times \text{Clock cycle}} = \frac{3.38}{2} = 1.69$$

Cache Performance with Increased Clock Rate

- Suppose that clock rate of the machine used in the previous example is doubled but the memory speed, cache misses, and miss rate are same. How much faster the machine be with the faster clock?

Since the clock rate is doubled, new miss penalty will be 2x40=80 clock cycles.

Total memory stall cycles = (0.02 x 80) + 0.36 x (0.04 x 80) = 2.75

$CPI_{\text{fast clock}} = 2 + 2.75 = 4.75$

$$\frac{\text{CPU time with slow clock}}{\text{CPU time with fast clock}} = \frac{I \times CPI_{\text{slow clock}} \times \text{Clock cycle}}{I \times CPI_{\text{fast clock}} \times \frac{\text{Clock cycle}}{2}}$$

$$= \frac{3.38 \times 2}{4.75} = 1.42$$

Calculating AMAT

- If a direct mapped cache has a hit rate of 95%, a hit time of 4 ns, and a miss penalty of 100 ns, what is the AMAT?

  AMAT = Hit time + Miss rate x Miss penalty = 4 + 0.05 x 100 = 9 ns

- If replacing the cache with a 2-way set associative increases the hit rate to 97%, but increases the hit time to 5 ns, what is the new AMAT?

  AMAT = Hit time + Miss rate x Miss penalty = 5 + 0.03 x 100 = 8 ns

Split vs. Unified Cache

- **Unified cache** (mixed cache): Data and instructions are stored together (von Neuman architecture)
- **Split cache**: Data and instructions are stored separately (Harvard architecture)
- Why do instructions caches have a lower miss ratio?

Size	Instruction Cache	Data Cache	Unified Cache
1 KB	3.06%	24.61%	13.34%
2 KB	2.26%	20.57%	9.78%
4 KB	1.78%	15.94%	7.24%
8 KB	1.10%	10.19%	4.57%
16 KB	0.64%	6.47%	2.87%
32 KB	0.39%	4.82%	1.99%
64 KB	0.15%	3.77%	1.35%
128 KB	0.02%	2.88%	0.95%

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.

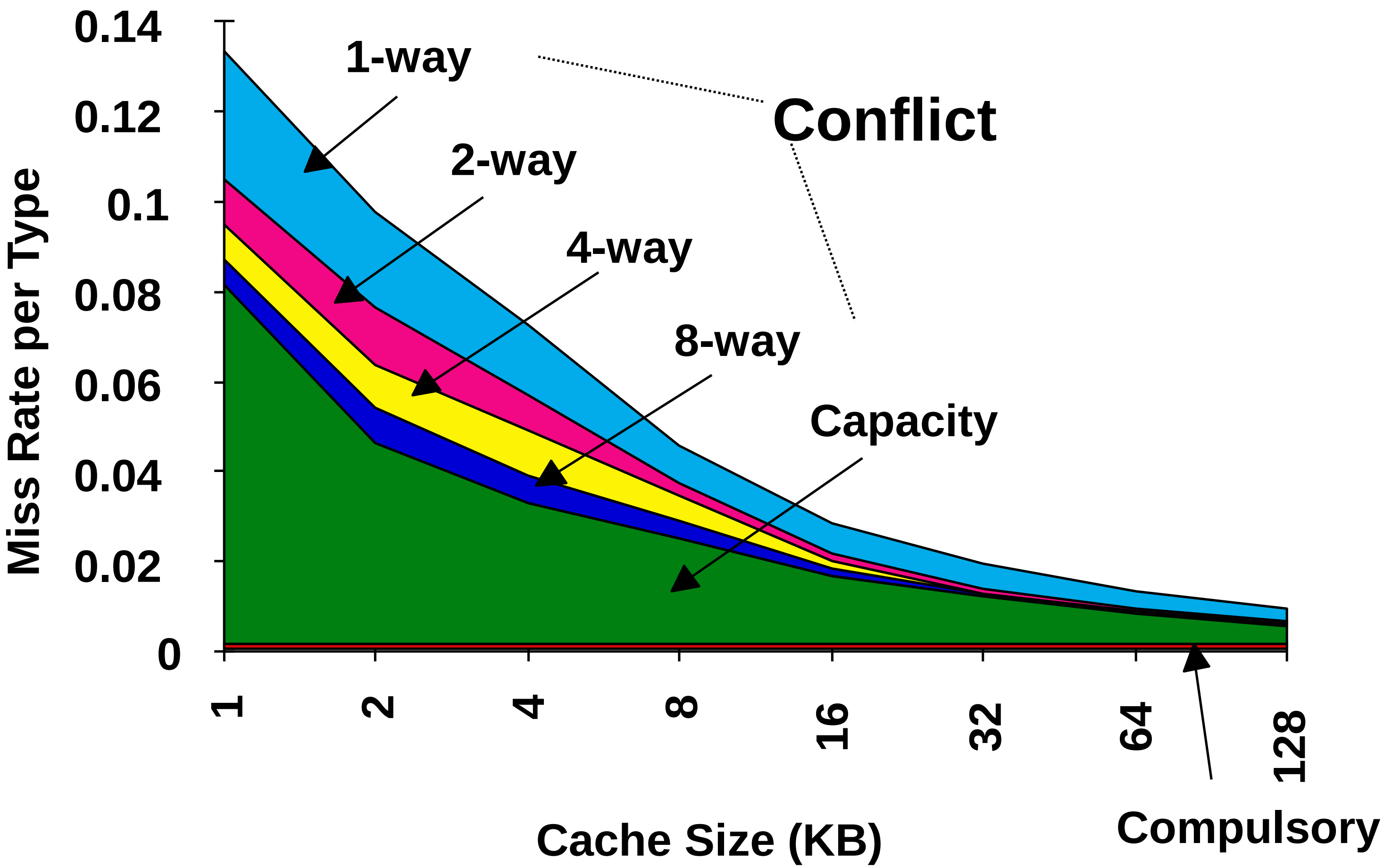
Reducing Misses

- 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 due to size of 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 due to associative and size of cache)*

Increasing Cache Size

- One way to decrease misses is to increase the cache size
  - Reduces capacity and conflict misses
  - No effect on compulsory misses
- However a larger cache may increase the hit time
  - larger cache => larger access time
  - If cache is too large, can’t fit it on the same chip as processor.

3Cs Absolute Miss Rate

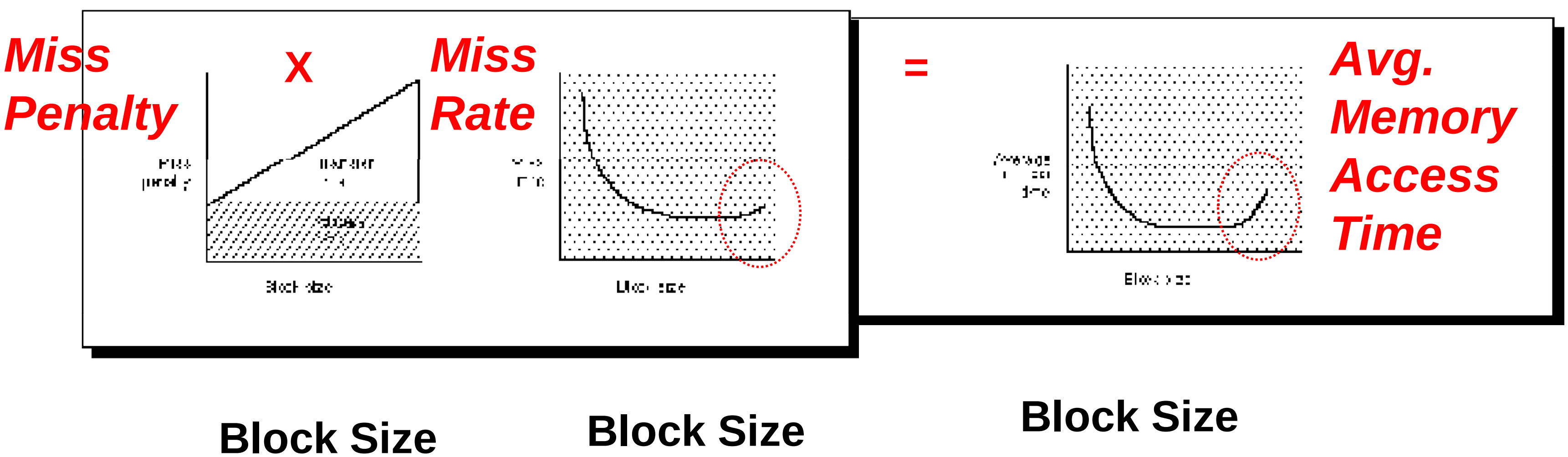


Increasing Block Size

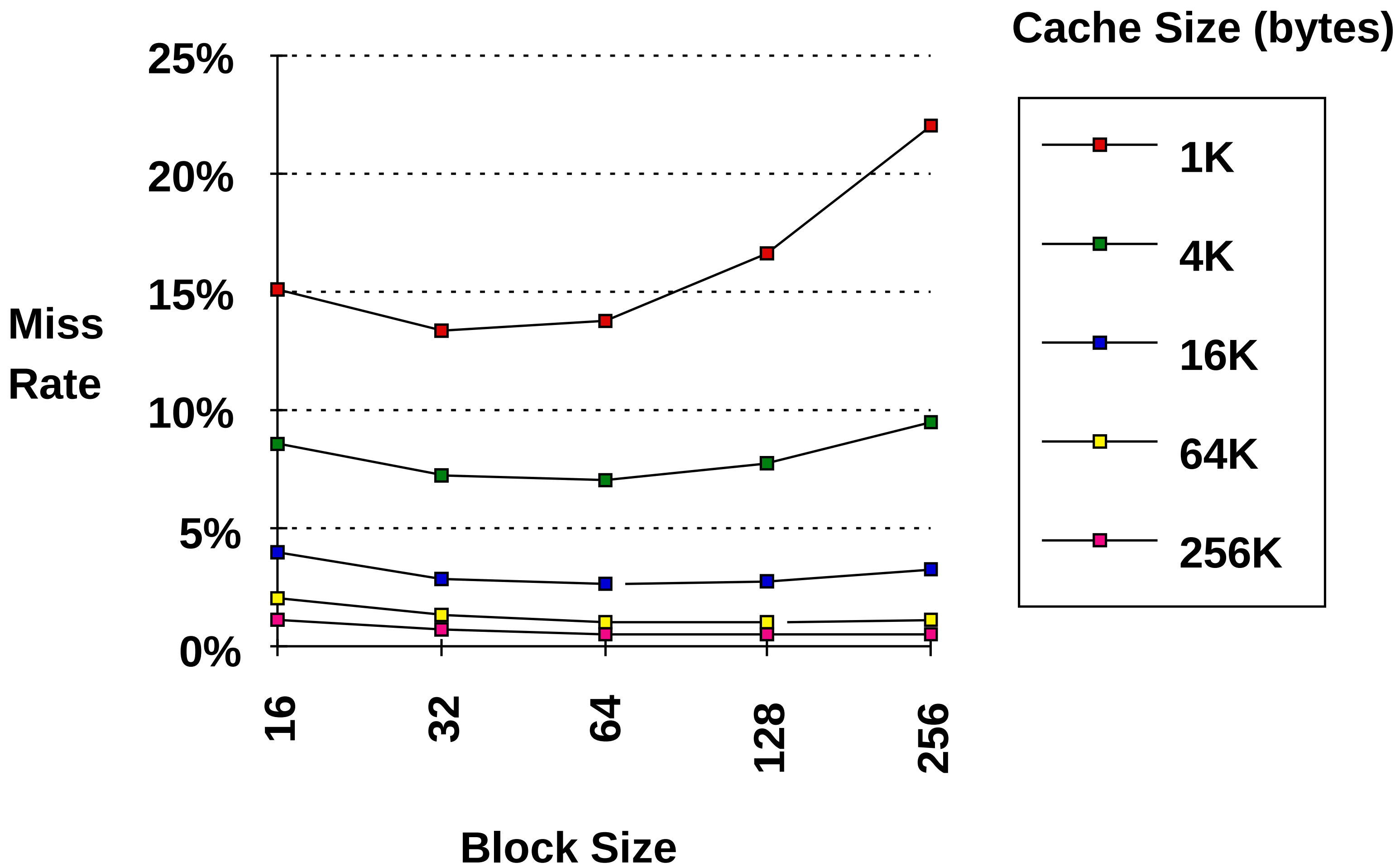
- Another way to reduce the miss rate is to increase the block size
  - Take advantage of spatial locality
  - Decreases compulsory misses
- However, larger blocks have disadvantages
  - May increase the miss penalty (need to get more data)
  - May increase hit time (need to read more data from cache and larger mux)
  - May increase miss rate, since conflict misses
- Increasing the block size can help, but don't overdo it.

Block Size vs. Cache Measures

- Increasing Block Size generally increases Miss Penalty and decreases Miss Rate
- As the block size increases the AMAT starts to decrease, but eventually increases



Increasing Block Size



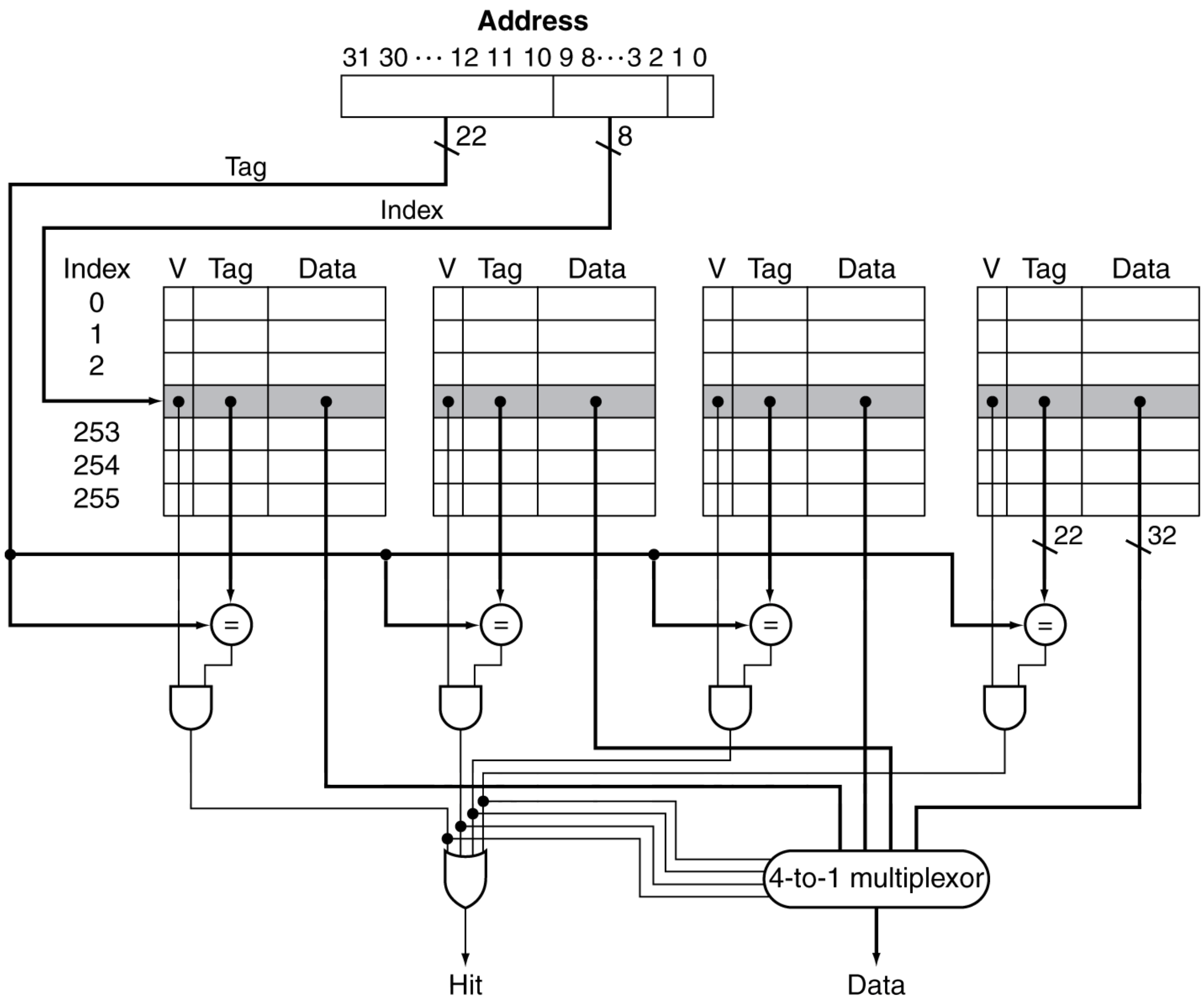
How Much Associativity

- 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%

Increasing Associativity

- Increasing associativity helps reduce conflict misses
- 2:1 Cache Rule:
  - The miss rate of a direct mapped cache of size N is about equal to the miss rate of a 2-way set associative cache of size N/2
  - For example, the miss rate of a 32 Kbyte direct mapped cache is about equal to the miss rate of a 16 Kbyte 2-way set associative cache
- Disadvantages of higher associativity
  - Need to do large number of comparisons
  - Need n-to-1 multiplexor for n-way set associative
  - Could increase hit time

Set Associative Cache Organization



AMAT vs. Associativity

Cache Size (KB)	Associativity 1-way	2-way	4-way	8-way
1	7.65	6.60	6.22	5.44
2	5.90	4.90	4.62	4.09
4	4.60	3.95	3.57	3.19
8	3.30	3.00	2.87	2.59
16	2.45	2.20	2.12	2.04
32	2.00	1.80	1.77	1.79
64	1.70	1.60	1.57	1.59
128	1.50	1.45	1.42	1.44

Red means A.M.A.T. not improved by more associativity

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

Using a 2nd Level Cache

- A second level (L2) cache reduces the miss penalty by providing a large cache between the first level (L1) cache and main memory
- L2 Equations

$$\textbf{AMAT} = \textbf{Hit Time}_{L1} + \textbf{Miss Rate}_{L1} \times \textbf{Miss Penalty}_{L1}$$

$$\textbf{Miss Penalty}_{L1} = \textbf{Hit Time}_{L2} + \textbf{Miss Rate}_{L2} \times \textbf{Miss Penalty}_{L2}$$

$$\textbf{AMAT} = \textbf{Hit Time}_{L1} + \textbf{Miss Rate}_{L1} \times (\textbf{Hit Time}_{L2} + \textbf{Miss Rate}_{L2} \times \textbf{Miss Penalty}_{L2})$$

Adding an L2 Cache

- If a direct mapped cache has a hit rate of 95%, a hit time of 4 ns, and a miss penalty of 100 ns, what is the AMAT?

AMAT = Hit time + Miss rate x Miss penalty = 4 + 0.05 x 100 = 9 ns

- If an L2 cache is added with a hit time of 20 ns and a hit rate of 50%, what is the new AMAT?

AMAT = Hit Time_{L1} + Miss Rate_{L1} x (Hit Time_{L2} + Miss Rate_{L2} x Miss Penalty_{L2})

= 4 + 0.05 x (20 + 0.5x100) = 7.5 ns

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

Multilevel Cache Example

- Given
  - CPU base CPI = 1, clock rate = 4GHz
  - Miss rate/instruction = 2%
  - Main memory access time = 100ns
- With just primary cache
  - Miss penalty = 100ns/0.25ns = 400 cycles
  - Effective CPI = 1 + 0.02 × 400 = 9

Example (cont.)

- Now add L-2 cache
  - Access time = 5ns
  - Global miss rate to main memory = 0.5%
- Primary miss with L-2 hit
  - Penalty = 5ns/0.25ns = 20 cycles
- Primary miss with L-2 miss
  - Extra penalty = 500 cycles
- CPI = 1 + 0.02 × 20 + 0.005 × 400 = 3.4
- Performance ratio = 9/3.4 = 2.6

Multilevel Cache Considerations

- Primary cache
  - Focus on minimal hit time
- L-2 cache
  - Focus on low miss rate to avoid main memory access
  - Hit time has less overall impact
- Results
  - L-1 cache usually smaller than a single cache
  - L-1 block size smaller than L-2 block size

Cache Performance Summary

- AMAT = Hit time + Miss rate x Miss penalty
- Split vs. Unified Cache
- 3 C's of misses
  - compulsory
  - capacity
  - conflict
- Methods for improving performance
  - increase (change) cache size
  - increase (change) block size
  - increase (change) associativity
  - add a 2nd level cache