CprE 381: Computer Organization and Assembly Level Programming

Cache Design

Henry Duwe
Electrical and Computer Engineering
Iowa State University

Administrative

- HW10 due tonight by 11:59pm
- HW11 posted
- Part 3b due in lab this week
- Part 4 posted – look ahead at what is required
- Exam 2 Curved
  - Average to HW4-HW7

Definitions: Hits and Misses

- A cache hit occurs if the cache contains the data that we're looking for. Hits are good, because the cache can return the data much faster than main memory.
- A cache miss occurs if the cache does not contain the requested data. This is bad, since the CPU must then wait for the slower main memory.

- There are two basic measurements of cache performance.
  - The hit rate is the percentage of memory accesses that are handled by the cache.
  - The miss rate (1 – hit rate) is the percentage of accesses that must be handled by the slower main RAM.
- Typical caches have a hit rate of 95% or higher, so in fact most memory accesses will be handled by the cache and will be dramatically faster.

Memory and Overall Performance

- Average Memory Access Time

  AMAT = Hit time + (Miss rate x Miss penalty)

- How do cache hits and misses affect overall system performance?
  - Assuming a hit time of one CPU clock cycle, program execution will continue normally on a cache hit. (Our earlier computations always assumed one clock cycle for an instruction fetch or data access.)
  - For cache misses, we'll assume the CPU must stall to wait for a load from main memory.

- The total number of stall cycles depends on the number of cache misses *and* the miss penalty.

  Memory stall cycles = Memory accesses x miss rate x miss penalty

- To include stalls due to cache misses in CPU performance equations, we have to add them to the "base" number of execution cycles

CPU time = (CPU execution cycles + Memory stall cycles) x Cycle time

Review: Memory and Overall Performance

- Average Memory Access Time

AMAT = Hit time + (Miss rate x Miss penalty)

- How do cache hits and misses affect overall system performance?
  - Assuming a hit time of one CPU clock cycle, program execution will continue normally on a cache hit. (Our earlier computations always assumed one clock cycle for an instruction fetch or data access.)
  - For cache misses, we'll assume the CPU must stall to wait for a load from main memory.

- The total number of stall cycles depends on the number of cache misses *and* the miss penalty.

Memory stall cycles = Memory accesses x miss rate x miss penalty

- To include stalls due to cache misses in CPU performance equations, we have to add them to the "base" number of execution cycles

CPU time = (CPU execution cycles + Memory stall cycles) x Cycle time

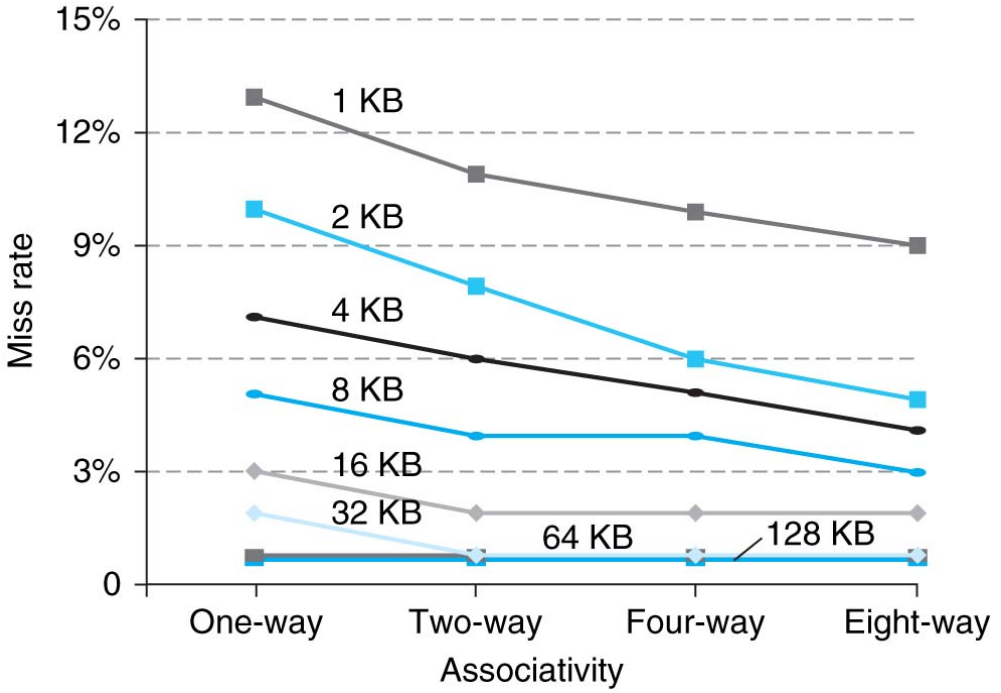
Comparing Cache Organizations

- Like many architectural features, caches are evaluated experimentally
  - As always, performance depends on the actual instruction mix, since different programs will have different memory access patterns
  - Simulating or executing real applications is the most accurate way to measure performance characteristics

- The graphs on the next few slides illustrate the simulated miss rates for several different cache designs
  - Again lower miss rates are generally better, but remember that the miss rate is just one component of average memory access time and execution time

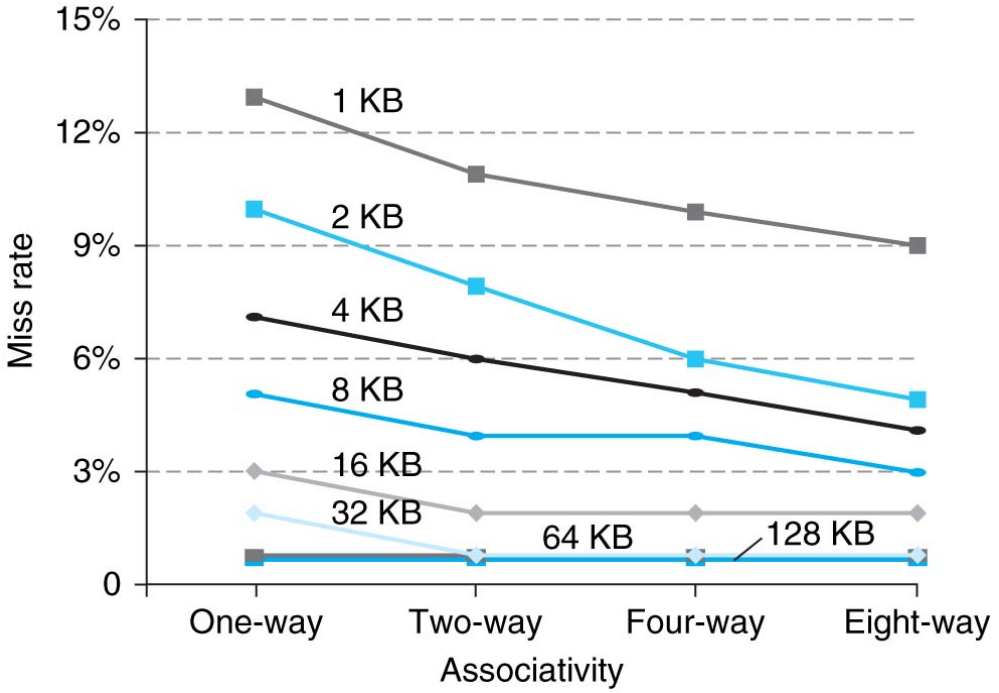
Cache Size and Miss Rates

- The cache size has a significant impact on performance
  - The larger a cache is, the less chance there will be of a conflict
  - Again this means the miss rate decreases, so the AMAT and number of memory stall cycles also decrease
- Figure 5.36 depicts the miss rate as a function of both the cache size and its associativity



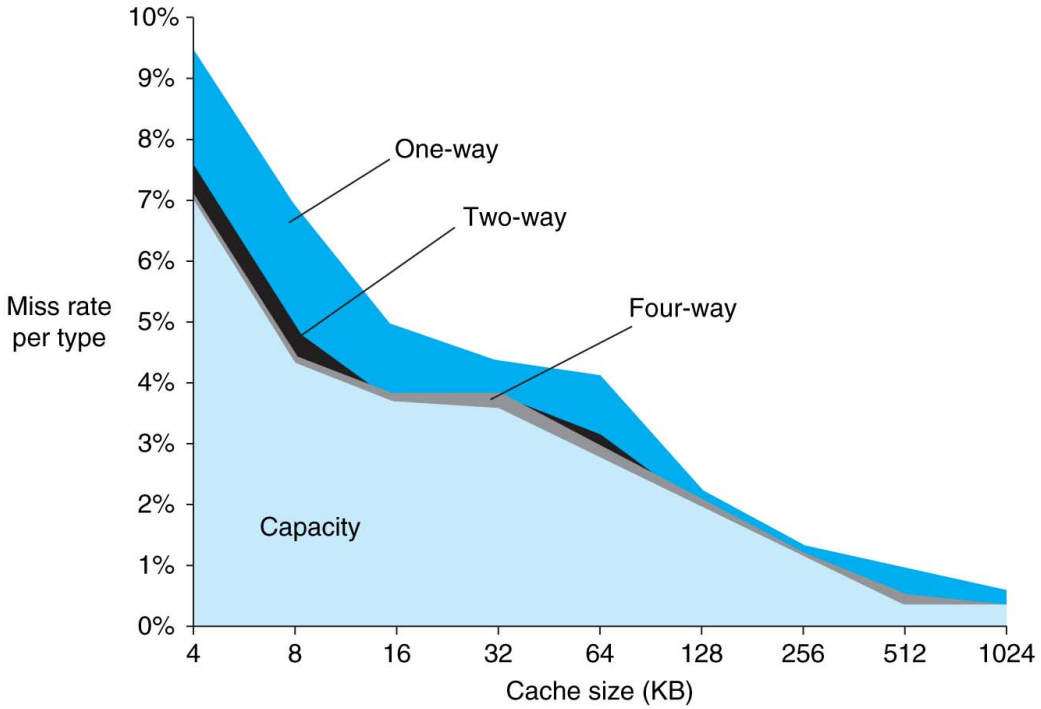
Associativity Tradeoffs and Miss Rates

- As we saw previously, higher associativity means more complex hardware
- But a highly-associative cache will also exhibit a lower miss rate
  - Each set has more blocks, so there's less chance of a conflict between two addresses which both belong in the same set
  - Overall, this will reduce AMAT and memory stall cycles
- Figure 5.36 of the textbook shows the miss rates decreasing as the associativity increases



Categorizing Cache Misses

- **Compulsory** (initial misses when cache is empty)
- **Capacity** (cache is full, depends only on cache size)
- **Conflict** (two addresses corresponding to the same block, depends on cache size and associativity)



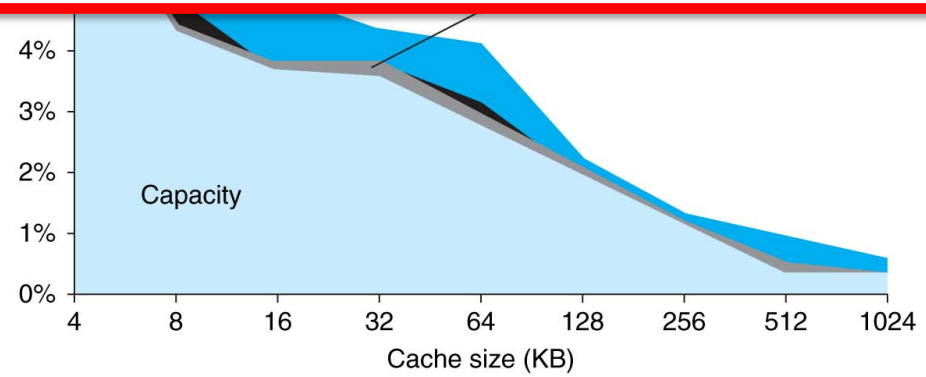
Categorizing Cache Misses

- **Compulsory** (initial misses when cache is empty)
- **Capacity** (cache is full, depends only on cache size)
- **Conflict** (two addresses corresponding to the same

In-class Assessment!

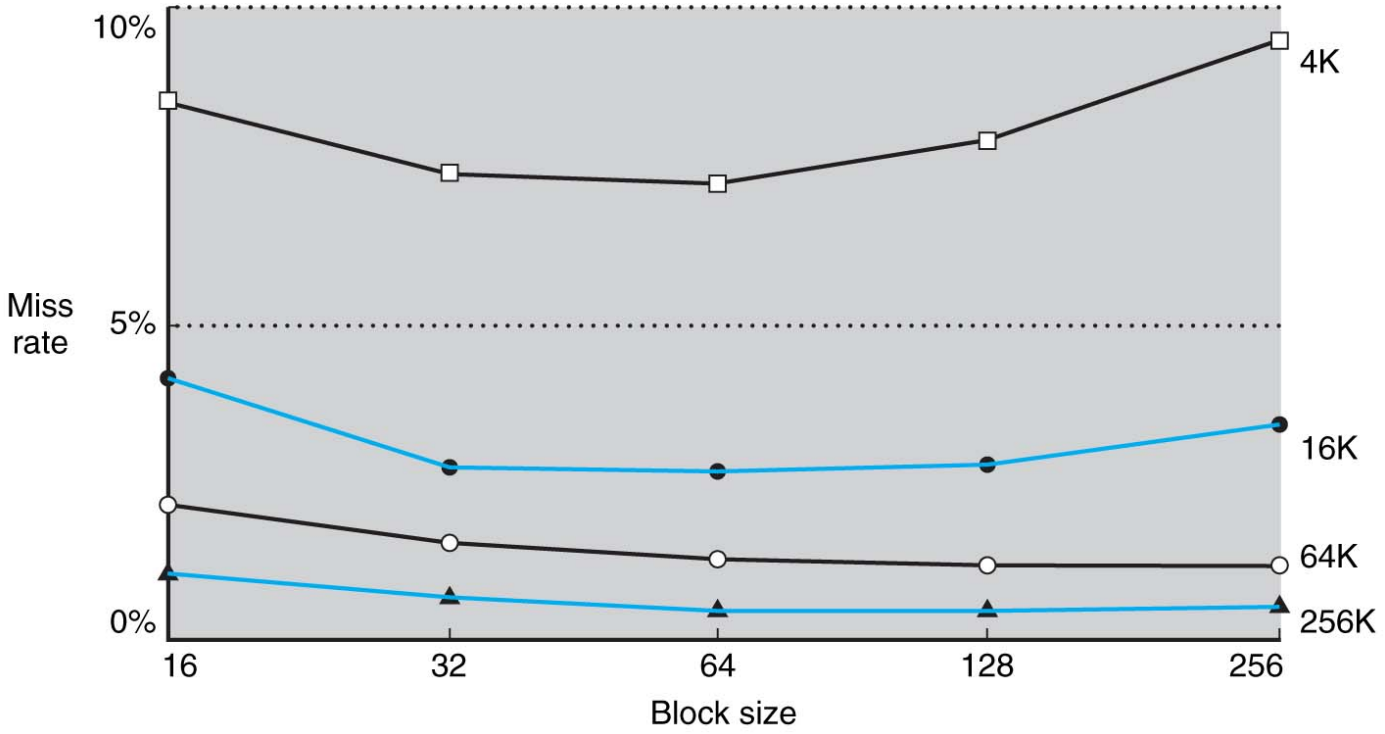
Access Code: Broke

Note: sharing access code to those outside of classroom or using access code while outside of classroom is considered cheating



Block Size and Miss Rates

- Finally, Figure 5.11 shows miss rates relative to the block size and overall cache size
  - Smaller blocks do not take maximum advantage of spatial locality

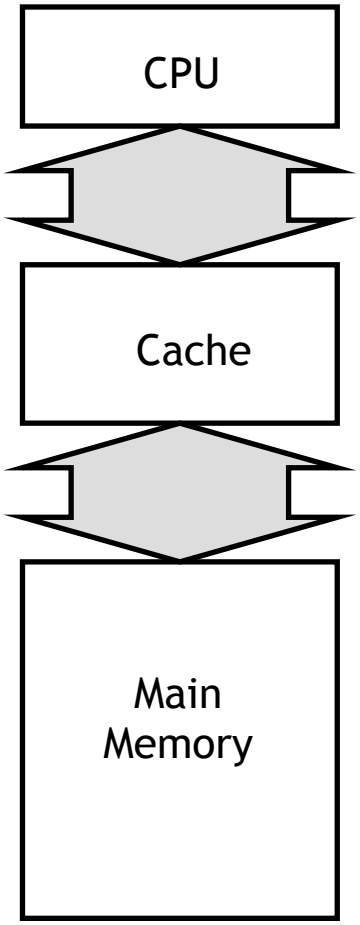


Basic Main Memory Design

- There are some ways the main memory can be organized to reduce miss penalties and help with caching
- For some concrete examples, let's assume the following three steps are taken when a cache needs to load data from the main memory
  1. It takes 1 cycle to send an address to the RAM
  2. There is a 15-cycle latency for each RAM access
  3. It takes 1 cycle to return data from the RAM
- In the setup shown here, the buses from the CPU to the cache and from the cache to RAM are all one word wide
- If the cache has one-word blocks, then filling a block from RAM (*i.e.*, the miss penalty) would take 17 cycles.

1 + 15 + 1 = 17 clock cycles

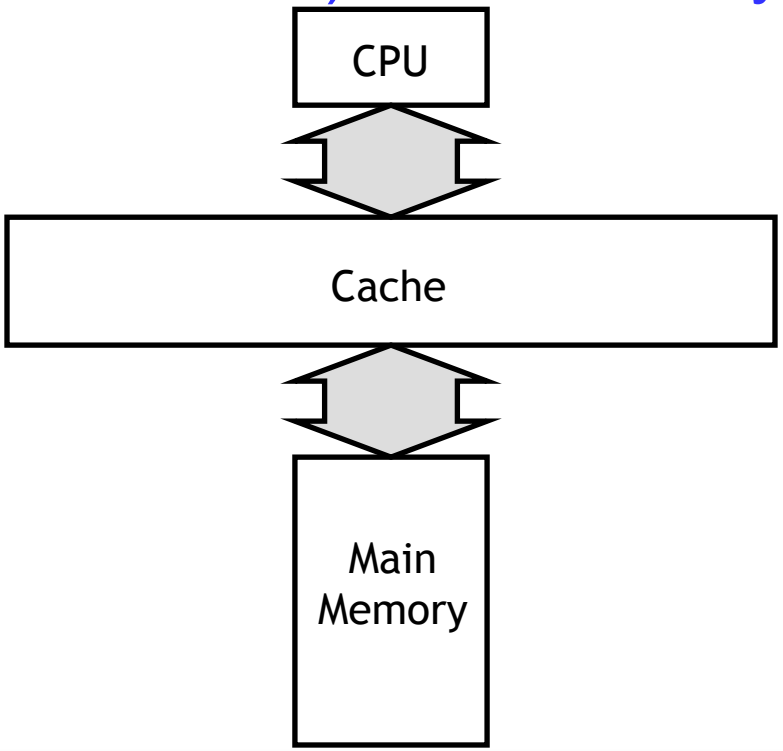
- The cache controller has to send the desired address to the RAM, wait and receive the data



Miss Penalties for Larger Cache Blocks

- If the cache has four-word blocks, then loading a single block would need four individual main memory accesses, and a miss penalty of 68 cycles!

4 x (1 + 15 + 1) = 68 clock cycles

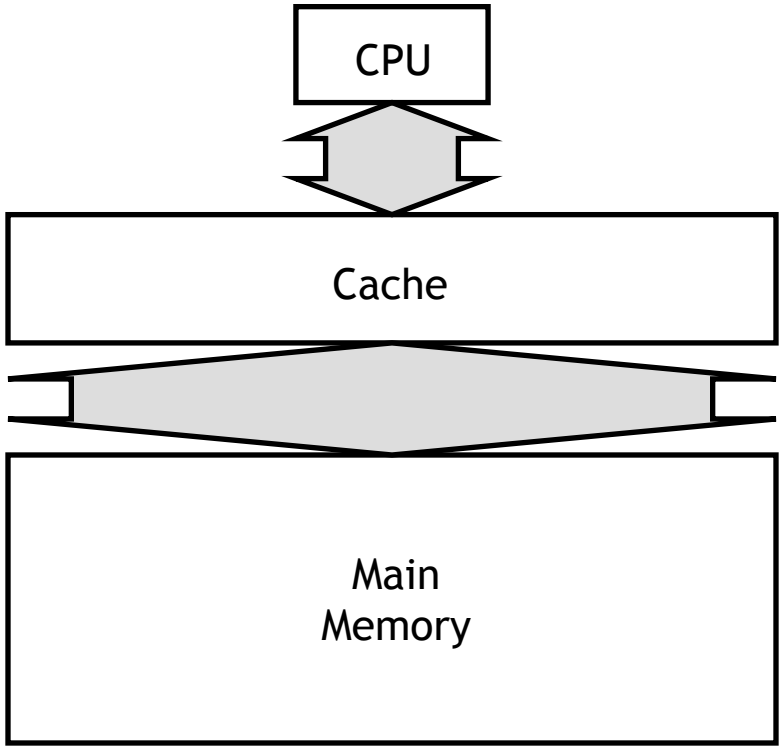


A Wider Memory

- A simple way to decrease the miss penalty is to widen the memory and its interface to the cache, so we can read multiple words from RAM in one shot
- If we could read four words from the memory at once, a four-word cache load would need just 17 cycles

  1 + 15 + 1 = 17 cycles

- The disadvantage is the cost of the wider buses – each additional bit of memory width requires another connection to the cache

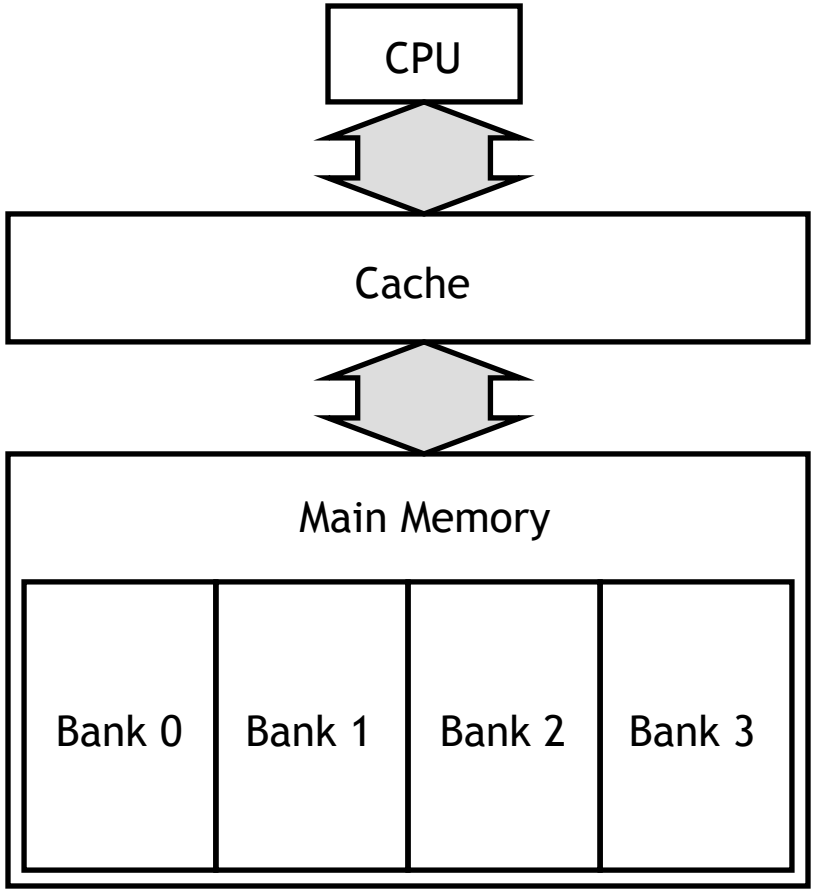


An Interleaved Memory

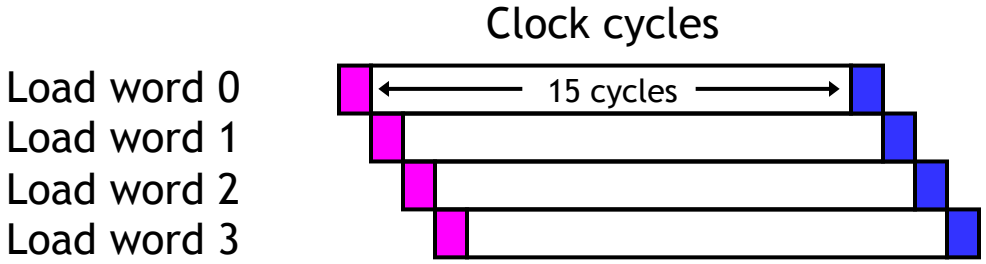
- Another approach is to interleave the memory, or split it into "banks" that can be accessed individually
- The main benefit is overlapping the latencies of accessing each word
- For example, if our main memory has four banks, each one byte wide, then we could load four bytes into a cache block in just 20 cycles

  1 + 15 + (4 x 1) = 20 cycles

- Our buses are still one byte wide here, so four cycles are needed to transfer data to the caches
- This is cheaper than implementing a four-byte bus, but not too much slower



Interleaved Memory Access



- Here is a diagram to show how the memory accesses can be interleaved
  - The magenta cycles represent sending an address to a memory bank
  - Each memory bank has a 15-cycle latency, and it takes another cycle (shown in blue) to return data from the memory
- This is the same basic idea as pipelining!
  - As soon as we request data from one memory bank, we can go ahead and request data from another bank as well
  - Each individual load takes 17 clock cycles, but four overlapped loads require just 20 cycles

Which is Better?

- Increasing block size can improve hit rate (due to spatial locality), but transfer time increases. Which cache configuration would be better?

	Cache #1	Cache #2
Block size	32-bytes	64-bytes
Miss rate	5%	4%

- Assume both caches have single cycle hit times. Memory accesses take 15 cycles, and the memory bus is 8-bytes wide:
  - i.e., an 16-byte memory access takes 18 cycles:

1 (send address) + 15 (memory access) + 2 (two 8-byte transfers)

recall: AMAT = Hit time + (Miss rate x Miss penalty)

Preview: Virtual Machines

- Host computer emulates guest operating system and machine resources
  - Improved isolation of multiple guests
  - Avoids security and reliability problems
  - Aids sharing of resources
- Virtualization has some performance impact
  - Feasible with modern high-performance comptuers
- Examples
  - IBM VM/370 (1970s technology!)
  - VMWare
  - Microsoft Virtual PC

Acknowledgments

- These slides contain material developed and copyright by:
  - Joe Zambreno (Iowa State)
  - Akhilesh Tyagi (Iowa State)
  - David Patterson (UC Berkeley)
  - Mary Jane Irwin (Penn State)
  - Christos Kozyrakis (Stanford)
  - Onur Mutlu (Carnegie Mellon)
  - Krste Asanović (UC Berkeley)
  - Morgan Kaufmann