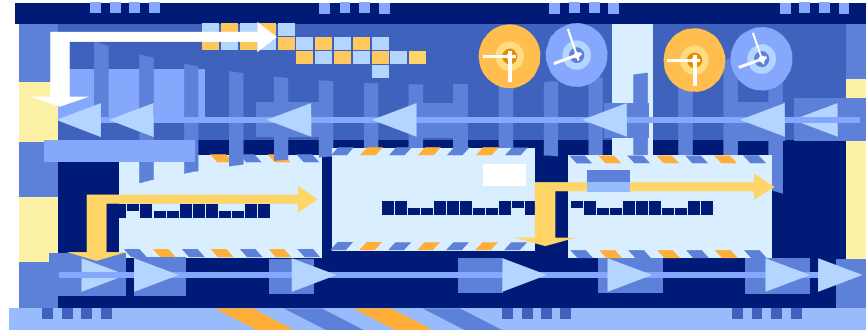


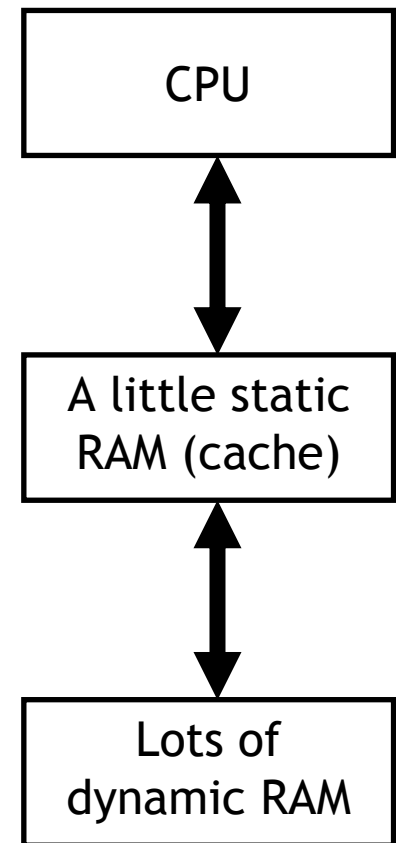
Cache performance



- Caches take advantage of locality to speed up most data accesses.
 - Increasing the **block size** can take advantage of **spatial locality**.
 - Increasing cache **associativity** helps reduce the **miss rate**.
- Today we'll finish up with associativity and do two more things.
 - We'll try to quantify the benefits of different cache designs, and see how caches affect overall performance.
 - We'll also investigate some main memory organizations that can help increase memory system performance.
- Next Monday we'll introduce some of the issues involved with writing to caches, and talk about cache configurations in modern processors.

Hits and misses

- To examine the performance of a memory system, we need to focus on a couple of important factors.
 - How long does it take to send data from the cache to the CPU?
 - How long does it take to copy data from memory into the cache?
 - How often do we have to access main memory?
- There are names for all of these variables.
 - The **hit time** is how long it takes data to be sent from the cache to the processor. This is usually fast, on the order of 1-3 clock cycles.
 - The **miss penalty** is the time to copy data from main memory to the cache. This often requires dozens of clock cycles.
 - The **miss rate** is the percentage of misses.



Average memory access time

- The **average memory access time**, or **AMAT**, can then be computed.

$$\text{AMAT} = \text{Hit time} + (\text{Miss rate} \times \text{Miss penalty})$$

This is just averaging the amount of time for cache hits and the amount of time for cache misses.

- How can we improve the average memory access time of a system?
 - Obviously, a lower AMAT is better.
 - Miss penalties are always much greater than hit times, so the best way to lower AMAT is to reduce the miss penalty *or* the miss rate.
- However, AMAT should only be used a general guideline. Remember that **execution time** is still the best performance metric.

Memory and overall performance

- 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.

$\text{Memory stall cycles} = \text{Memory accesses} \times \text{miss rate} \times \text{miss penalty}$

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

$\text{CPU time} = (\text{CPU execution cycles} + \text{Memory stall cycles}) \times \text{Cycle time}$

Performance example

- Assume that 33% of the instructions in a program are data accesses. The cache hit ratio is 97% and the hit time is one cycle, but the miss penalty is 20 cycles.
- If the cache was perfect and never missed, the AMAT would be one cycle. But even with just a 3% miss rate, the AMAT here increases 1.6 times!

$$\begin{aligned} \text{AMAT} &= \text{Hit time} + (\text{Miss rate} \times \text{Miss penalty}) \\ &= 1 \text{ cycle} + (3\% \times 20 \text{ cycles}) \\ &= 1.6 \text{ cycles} \end{aligned}$$

- What about the overall performance? If I instructions are executed, then the number of wasted cycles will be $0.2 \times I$.

$$\begin{aligned} \text{Memory stall cycles} &= \text{Memory accesses} \times \text{Miss rate} \times \text{Miss penalty} \\ &= 0.33 I \times 0.03 \times 20 \text{ cycles} \\ &= 0.2 I \text{ cycles} \end{aligned}$$

This code is 1.2 times slower than a program with a “perfect” CPI of 1!

Memory systems are a bottleneck

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

- Processor performance traditionally outpaces memory performance, so the memory system is often the system bottleneck.
- For example, with a base CPI of 1, the CPU time from the last page is:

CPU time = (1 + 0.2 I) × Cycle time

- What if we could *double* the CPU performance so the CPI becomes 0.5, but memory performance remained the same?

CPU time = (0.5 I + 0.2 I) × Cycle time

- The overall CPU time improves by just $1.2/0.7 = 1.7$ times!
- Refer back to Amdahl's Law from Homework 2, and textbook page 101.

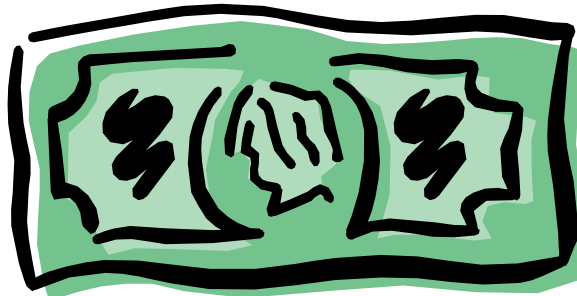
Improving memory (and overall) performance

Memory stall cycles = Memory accesses \times Miss rate \times Miss penalty

- You can decrease the number of stall cycles by reducing any or all of the individual factors.
 - Telling programmers to load and store less doesn't usually work!
 - It's probably easier to reduce the miss rate or the miss penalty.
- There are many methods for reducing the miss rate.
 - Using an associative cache can help reduce conflicts.
 - Making the cache bigger lets us store more stuff in it.
 - Adjusting the block size can take advantage of spatial locality.
- Later today we'll see some ways to reduce the miss penalty as well.

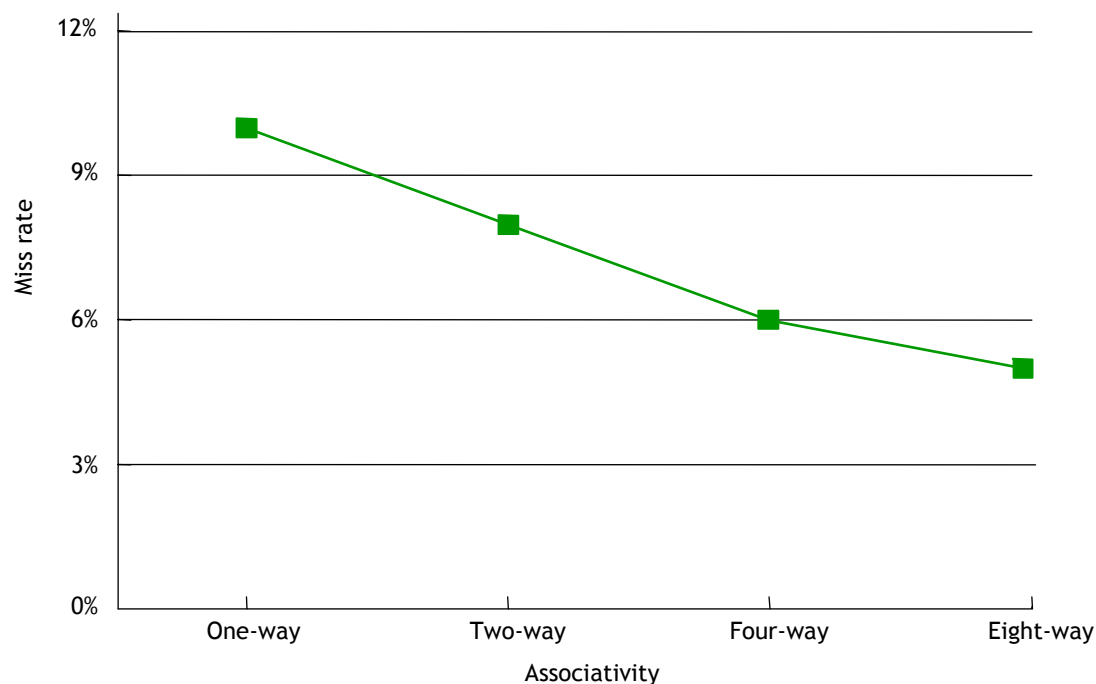
Comparing cache organizations

- 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.
 - You'll probably do some cache simulations if you take CS333.



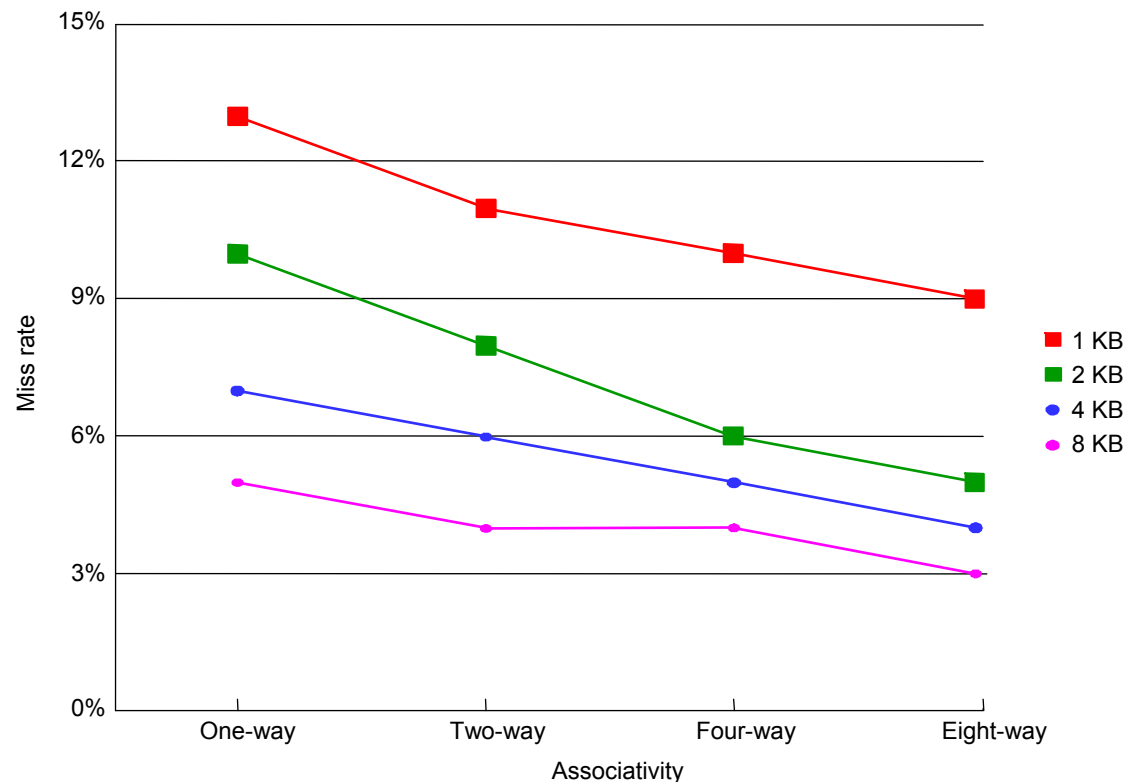
Associativity tradeoffs and miss rates

- As we saw last time, 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 7.29 on p. 604 of the textbook shows the miss rates decreasing as the associativity increases.



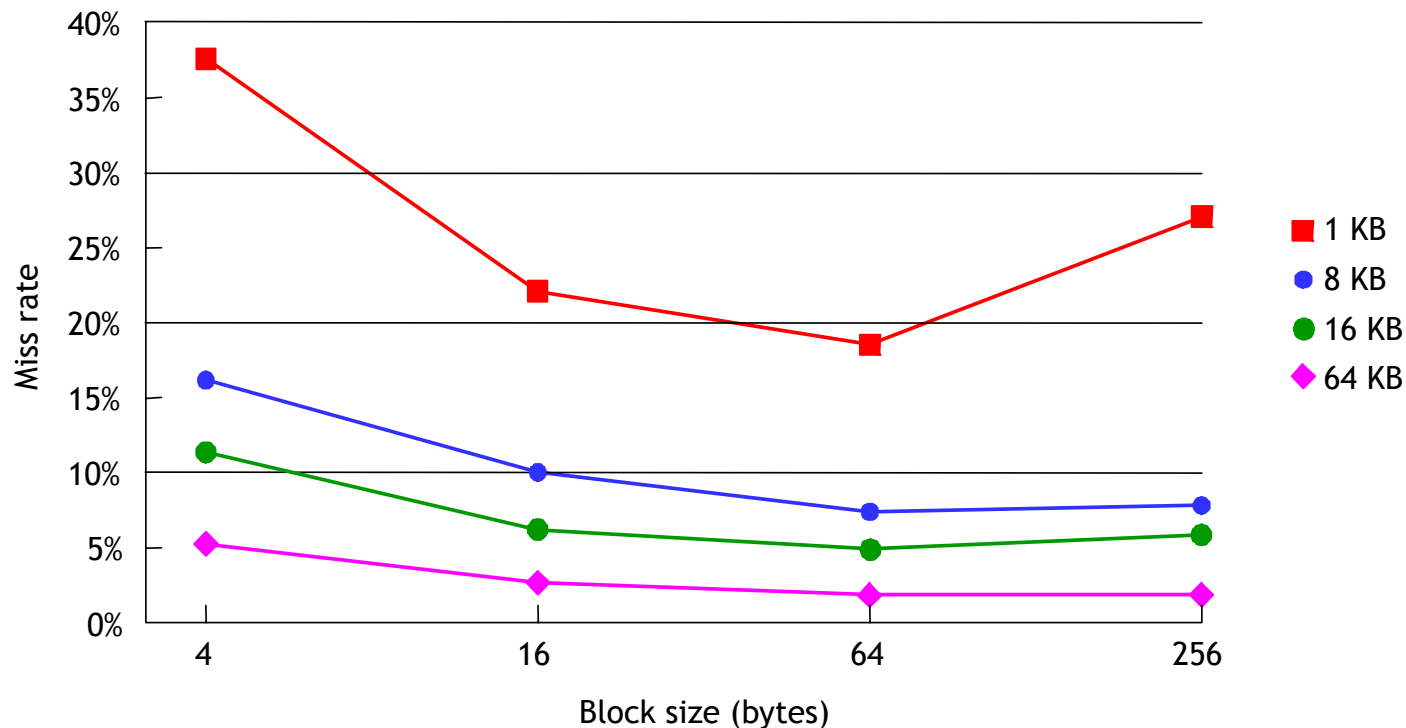
Cache size and miss rates

- The cache size also 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.
- The complete Figure 7.29 depicts the miss rate as a function of both the cache size and its associativity.



Block size and miss rates

- Finally, Figure 7.12 on p. 559 shows miss rates relative to the block size and overall cache size.
 - Smaller blocks do not take maximum advantage of spatial locality.
 - But if blocks are *too* large, there will be fewer blocks available, and more potential misses due to conflicts.

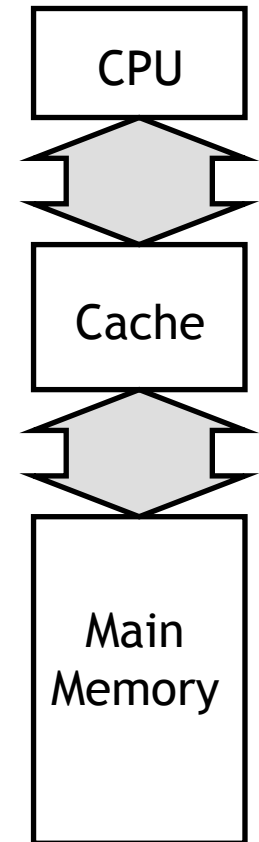


Basic main memory design

- 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 byte wide.
- If the cache has one-byte blocks, then filling a block from RAM (i.e., the miss penalty) would take 17 cycles.

$$1 + 15 + 1 = 17 \text{ clock cycles}$$

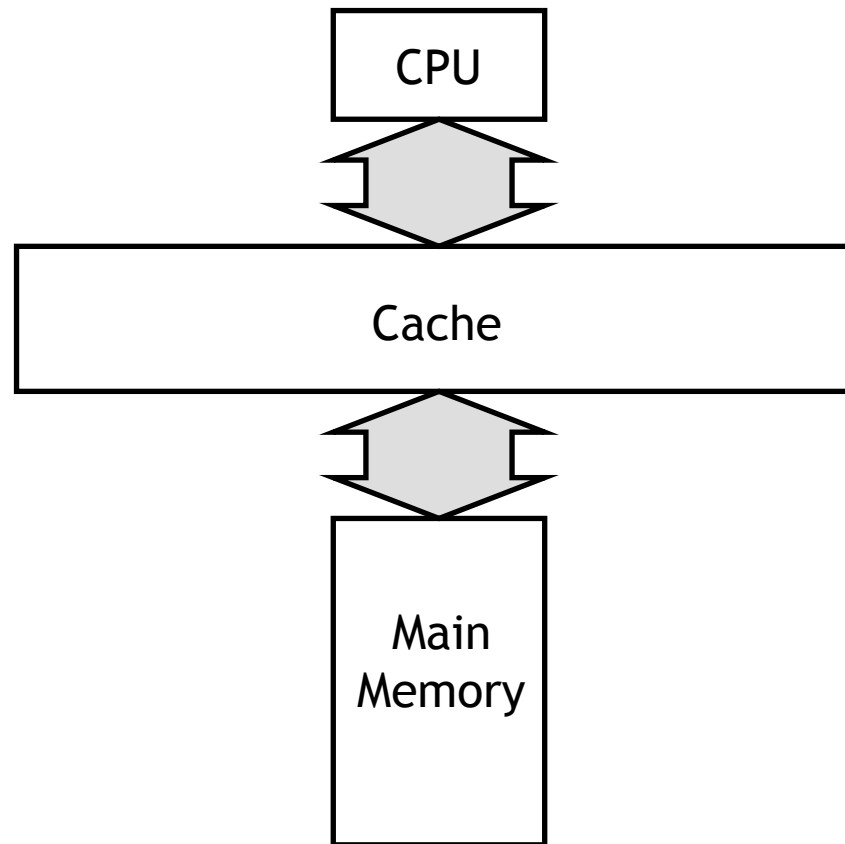
- The cache controller has to send the desired address to the RAM, wait and receive the data.



Miss penalties for larger cache blocks

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

$$4 \times (1 + 15 + 1) = 68 \text{ clock cycles}$$

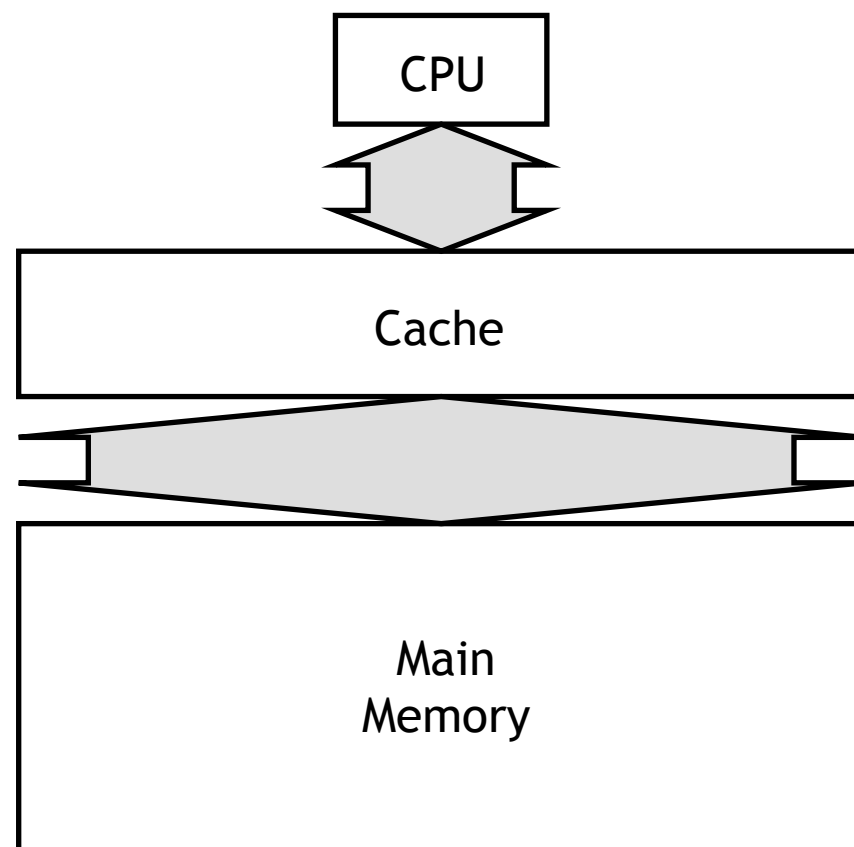


A wider memory

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

$$1 + 15 + 1 = 17 \text{ cycles}$$

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

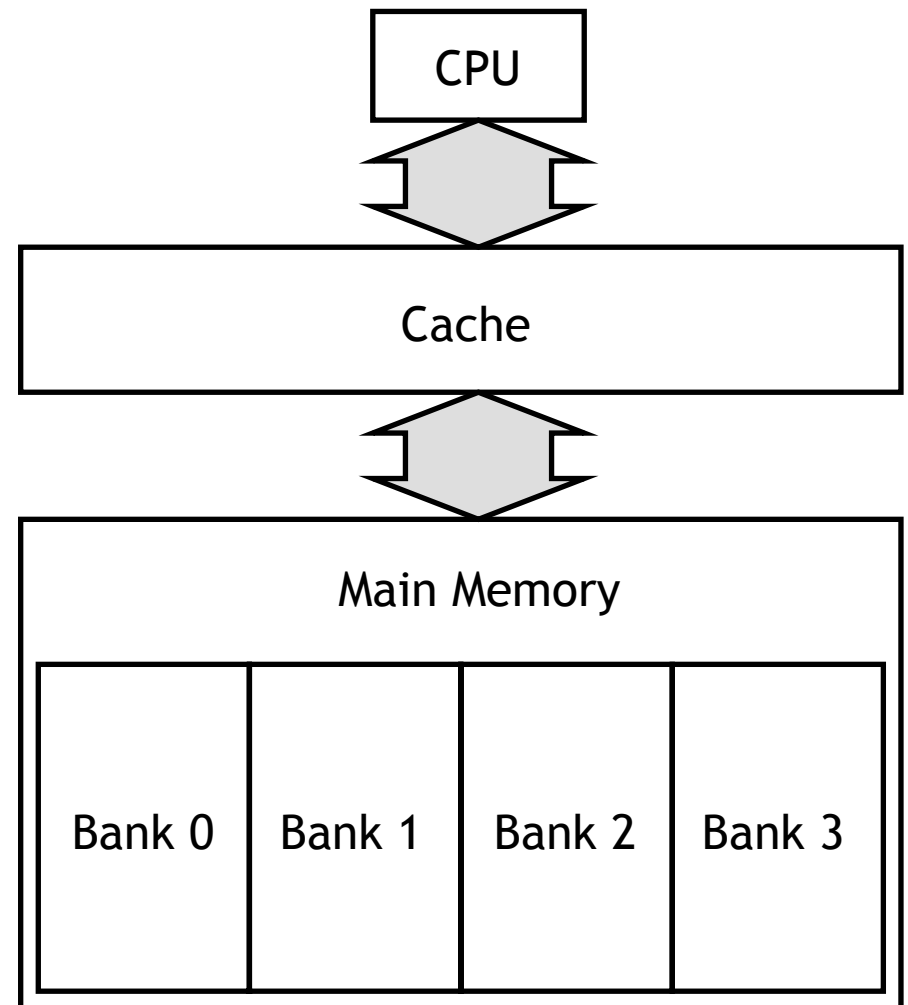


An interleaved memory

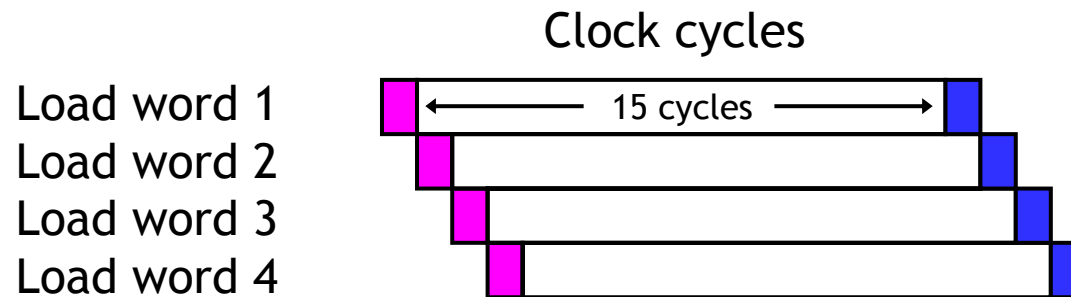
- Another approach is to **interleave** the memory, or split it into “banks” that can be accessed individually.
- The main benefit is overlapping the latencies of accessing each word.
- For example, if our main memory has four banks, each one byte wide, then we could load four bytes into a cache block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

$$1 + 15 + (4 \times 1) = 20 \text{ cycles}$$

- 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 accesses



- 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.

Summary

- Memory system performance depends upon the cache **hit time**, **miss rate** and **miss penalty**, as well as the actual program being executed.
 - We can use these numbers to find the **average memory access time**.
 - We can also revise our CPU time formula to include **stall cycles**.

$AMAT = \text{Hit time} + (\text{Miss rate} \times \text{Miss penalty})$

$\text{Memory stall cycles} = \text{Memory accesses} \times \text{miss rate} \times \text{miss penalty}$

$\text{CPU time} = (\text{CPU execution cycles} + \text{Memory stall cycles}) \times \text{Cycle time}$

- The organization of a memory system affects its performance.
 - The cache size, block size, and associativity affect the miss rate.
 - We can organize the main memory to help reduce miss penalties. For example, **interleaved memory** supports pipelined data accesses.