In class, you have studied the organization of the single level cache. Today we will discuss multilevel caches and practice solving cache-related problems. First, let's make sure we understand how a simple cache works.

Problem Here is a series of address references given as word addresses: 2, 3, 11, 16, 21, 13, 64, 48, 19, 11, 3, 22, 4, 27, 6, and 11. Assuming a direct-mapped cache with 16 one-word blocks that is initially empty, label each reference in the list as a hit or a miss and show the final contents of the cache.

Average Memory Access Time and Memory Stall Cycles

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 (at least). The *miss rate* is the percentage of misses. 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 usually 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 as a general guideline. Remember that execution time is still the best performance metric.

Another way of measuring cache performance is by computing the number of cycles the processor must stall for all the memory accesses in a given program. We can compute this with the following formula:

$$\text{Memory Stall Cycles} = \text{Memory Accesses} \times \text{Miss rate} \times \text{Miss penalty}$$

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.

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

$$\begin{aligned} \text{Memory Stall Cycles} &= \text{Memory Accesses} \times \text{Miss rate} \times \text{Miss penalty} \\ &= \\ &= \end{aligned}$$

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! If we have an ideally pipeline (CPI = 1) in this is situation, then our new CPI = $1 + 0.33 * 0.03 * 20 \approx 1.2$. So cache performance can have a big impact on CPU performance.

Multilevel Cache
Cache is helpful only if the needed data is available there. On a cache miss, the processor still needs to access memory to retrieve the data. Each memory access is slow and results in high miss penalty. There are two ways to lower the amount of time lost on cache misses:

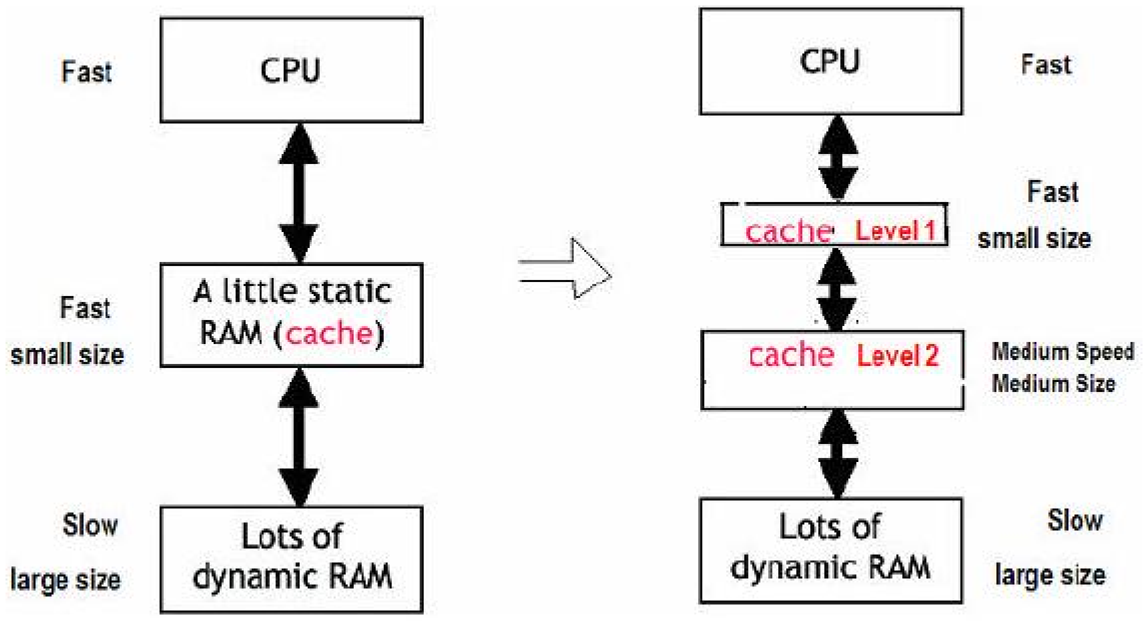
1. Decrease the **miss rate**. This implies building larger caches. But as the size of the cache increases, so does the average time to access it.
2. Decrease the **miss penalty**. We cant just make memory faster, but we can put another, larger layer of cache on top of the slow memory. Even a very large cache is much faster than memory. This concept, called **multilevel cache**, is illustrated below.

On each load and store, the processor needs to access data in memory. First, it checks Level 1 cache (L1). If the data is found, it is read or written there. On L1 cache miss, the processor accesses Level 2 cache (L2) instead of memory. Since L2 is large, it is very likely that the data can be found there. For L2 cache, read/write time is longer than L1 but much shorter than memory. Thus the overall memory access time is lower.

With this cache hierarchy, we achieve good balance between size and speed:

1. By keeping the size of L1 cache small, we retain the high speed memory access to keep the CPU busy.

Figure 1. Concept of a multilevel cache



2. Enlarging the overall cache size by having large L2 cache provides a good backup for L1 cache misses without slowing down the response time on L1 cache hits.

Thus we can avoid the worst case: memory access on a cache miss.

Can we add more levels of cache to achieve even better performance?
The optimum number of levels of cache is determined by the trade-off between cache size and access latency. In practice, 3 levels is pretty common in modern designs with L1=32-64KB, L2=256KB-2MB, and L3=2MB-32MB.

Problems

1. Understand the expressions of AMAT and Memory stall cycles thoroughly.

$$\text{AMAT} = \text{Hit time} + (\text{Miss rate} \times \text{Miss penalty})$$
$$\text{Memory stall cycles} = \text{Memory accesses} \times \text{miss rate} \times \text{miss penalty}$$

The Corleone2004 processor has two levels of data caches, with the characteristics shown below. You can also assume that it takes 50 clock cycles to request and complete a 32-byte transfer between main memory and the L2 cache.

Find the average memory access time (AMAT) for both the L2 and L1 cache.

Table 1. Cache Parameters for Q1

	L1	L2
Data Size	32KB	256KB
Block Size	8bytes	32bytes
Associativity	Direct-mapped	4-way
Hit Time	1 cycle	19 cycles
Miss Rate	5%	2%