2.1. When we were discussing floating point addition, we made the simplifying assumption that each of the functional units took the same amount of time. Suppose that fetch and store each take 2 nanoseconds and the remaining operations each take 1 nanosecond.
a. How long does a floating point addition take with these assumptions?
b. How long will an unpipelined addition of 1000 pairs of floats take with these assumptions?
c. How long will a pipelined addition of 1000 pairs of floats take with these assumptions?
d. The time required for fetch and store may vary considerably if the operands/ results are stored in different levels of the memory hierarchy. Suppose that a fetch from a level 1 cache takes two nanoseconds, while a fetch from a level 2 cache takes five nanoseconds, and a fetch from main memory takes fifty nanoseconds. What happens to the pipeline when there is a level 1 cache miss on a fetch of one of the operands? What happens when there is a level 2 miss?

2.2. Explain how a queue, implemented in hardware in the CPU, could be used to improve the performance of a write-through cache.

2.3. Recall the example involving cache reads of a two-dimensional array. How does a larger matrix and a larger cache affect the performance of the two pairs of nested loops? What happens if MAX = 8 and the cache can store four lines? How many misses occur in the reads of A in the first pair of nested loops? How many misses occur in the second pair?