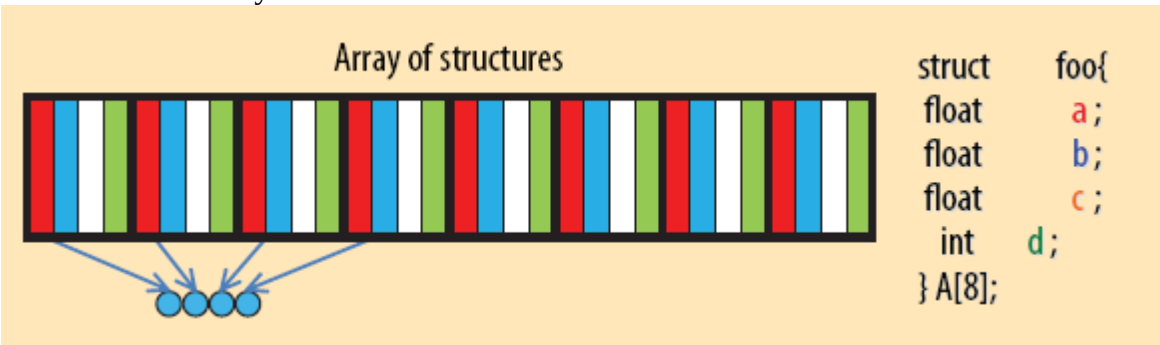
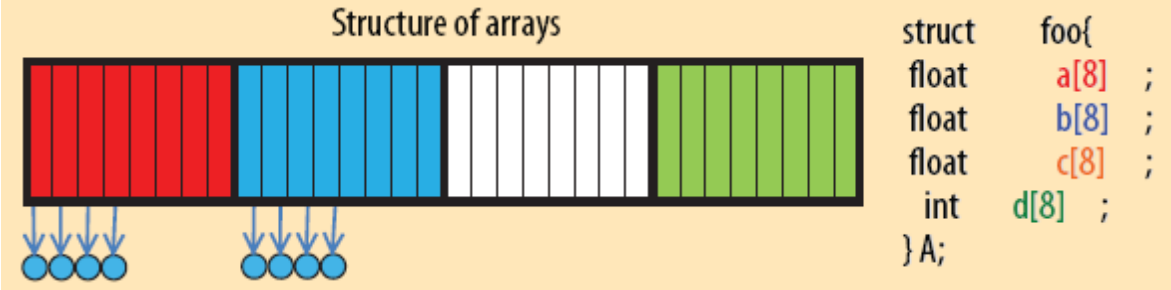
CSCI-GA.3033-009
Multicore Processors: Architecture & Programming
Homework Assignment # 2 Solution

1. a. This is how the array elements are stored in memory and what happens when many threads access different array elements.



b. From the above figure we can see that the different threads are accessing *scattered* parts of the memory and bring them to them to the shared cache. This scattering means more cache blocks are brought in. This means more cache misses.

c. The best way to deal with the cache misses problem is:



2. There are several reasons for this:

- Dividing the work among too many threads means each thread will do very little work. This cannot justify the overhead of starting, terminating, and keeping track of each thread state.
- Too many threads sharing fixed hardware resources is a source of congestion and hence performance degradation.
- Having more software threads that the hardware threads means that the operating system will have to schedule them on a round-robin fashion, giving each thread a time-slice after which it is suspended and another thread is scheduled. These suspension-restoration operations are expensive.

3. a) One way of doing it is to divide the interval among the hardware threads. Each thread will be responsible for n/p numbers and find whether it is a prime or not by dividing each number M in its interval by each number [2, M/2] and see if there is a remainder

b) Here the interval [2, X/2] is divided among the threads. Each thread finds whether X is divisible by any number in its interval. If it is divisible, then the program ends with a conclusion that X is not prime.

4. Use banked memory. If the addresses accessed by the different loads map to different banks, they will be serviced in parallel. Note that the worst case scenario, when all addresses map to the same bank, is almost the same performance as non-banked memory (We say *almost* because an address to a bank requires some overhead).

5. The closer we are from the processor the more the access latency becomes more important (needs to respond to the processor as fast as possible). The closer we are from main memory, the more important the hit rate is (because main memory access is expensive). So for L1: access latency is more important; for L3: hit rate is more important. This is why L1 is smaller in size than L2, which is smaller than L3. This is also why L1 has smaller associativity than L2, which is smaller than L3.