I did the extra credit

Data Structures

Cache struct. This consisted of the properties of a cache.

- Boolean to determine whether a given cache should perform prefetching
- Integers for: cache size, block size, number of sets, and the associativity
- A char pointer to determine the replacement policy
- A char pointer to determine the association type
- Most importantly, a pointer to serve as an array of Sets

Set struct. This is what the cache has an array of, and a set is an array of lines

- A pointer that serves as an array of Lines

Line struct. This has tag bits, a valid bit, and an age

- Tag bits are the identifier of an address, saying if two addresses are equal or not
- The age is used for FIFO to see which line came in first, and should thus be evicted first
- The valid bit simply says whether the line is empty or not

Simulation struct. An easy way to access all of the sim stats & have them in one place

- Int for hits, misses, reads, and writes

Prefetching

For the direct cache, reads went up, hits went up, and misses went down. This makes sense. We read more addresses, but due to the idea of spatial locality, more hits are achieved, since the prefetched memory is more likely to be used again because it is close in memory. When hits go up, misses go down.

In a fully associative cache, we had the same result, which is to be expected.

The same can be said of an N way associative cache.

This all makes sense. Prefetching leads to a significant amount more hits, and this shows the idea of spatial locality. Memory that is close together tends to get used together, such as elements in an array, so it makes sense to prefetch and bring all of this memory into the cache so we will have faster access to it. Memory reads went up as well, since for every cache miss, we read in 2 memory addresses instead of 1.