Michael Wexler
Assignment 4

Design:

The program consists of one C file called sim.c. Here, an array of cache sets are defined. Because this is direct mapped, each spot in array maps to one tuple. Each tuple contains several fields, most notably a field for the tag, whether its valid, or (for write-back) whether it has a dirty bit. First, fundamental cache parameters are calculated, such as size of tag, size of offset, number of sets, etc. Next, the trace file is opened, and line by line is read, and data in each line is parsed. The read/write operation is then called for the address at that line.

Implementation of read/write:

For both read and write operations, the set index, tag, and block size are computed depending on the address referred to. This is accomplished using bit-shift operations. Then, the cache is updated, and output parameters (such as cache_hits) are updated.

The number of memory writes depended on whether we in write-back or write-through mode. For write-through mode, memory is always immediately written to memory after being placed in cache. Therefore, memory writes was incremented any time something was placed in cache. However, for write-back mode, items are not written to memory immediately after being placed in cache. Instead, items newly put in the cache are marked “dirty”, and are eventually placed in the memory if they are ever evicted from the cache to make room for other items. Evictions can occur both in write and read operations. Therefore, in terms of keeping # memory writes accurate, I had to always increment it whenever things would be evicted from cache. Other than that variable, the other variables are the same for both modes, so I pretty much re-used most of my code between my read and write functions.

There were several different types of misses that could occur, and because different variables were incremented in each occurrence, it was important to differentiate them. First of all, misses could occur either in read or write operations. In both situations, the only time a miss occurred was either an item was being placed in cache in a spot marked invalid, or an item was being placed in a valid spot in cache that had an element (with a different tag) already in that spot. The output parameters were updated while taking these different types of misses into account.