**Readme for LAB2:**

Compile the code using “**g++ cachesimulator.cpp –o cache**”

Run the code just using **“./cache**”

We made necessary changes for the code to run with just “./cache”

The output file is stored in “**trace.out**”, you can **rename** it to “**trace.txt**” and then open in an editor.

The code flow is as follows:

First we read the cache configurations text file and then initialize the cache according to the specifications.

When an invalid configuration is given such as, the block size not being a multiple of cache size or the block size being greater than the cache size, we assign a default configuration to the cache parameters along with an error message.

When you read an address from the trace file, we split it into corresponding tag, index and offset bits, w.r.t both the caches. We then determine if it’s a hit in L1 by scanning though all the ways of a particular set for a tag match. If it’s a hit, we report a hit in L1 and no access in L2. If it’s a miss in L1, then we scan through L2 for a tag match, if that’s a fail too we update the tag in L2 and then in L1. Eviction from L1 or L2 will result in a NOP according to our implementation.

When you write to a cache, if it’s a miss, the write is forwarded to the next cache level. If it’s a hit, a hit is reported in the cache level