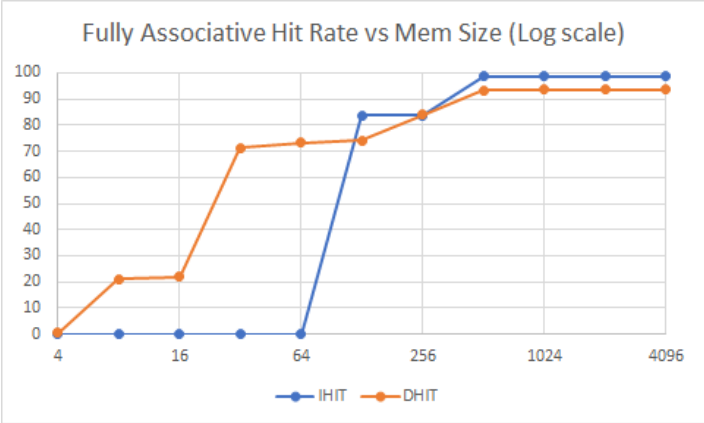
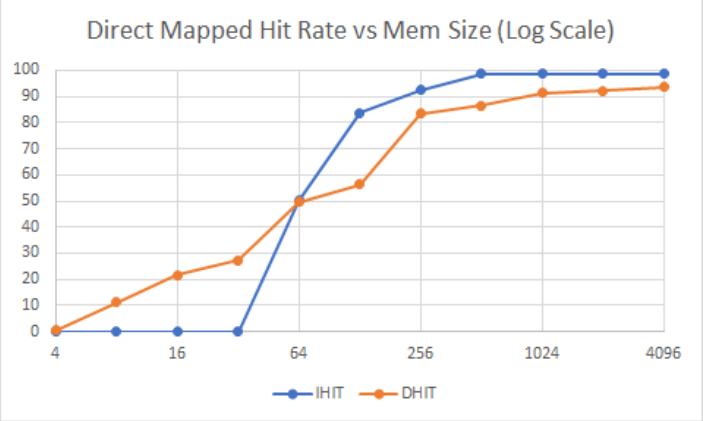
Cache Analysis

Madhur Singhal 2015CS10235 and Akash Mittal 2015CS10208

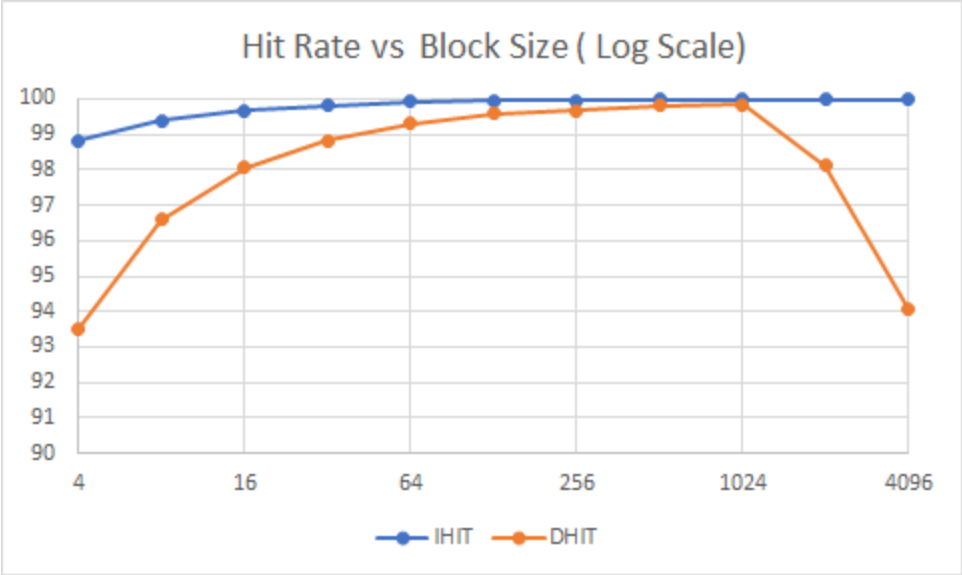
1) Cache Size





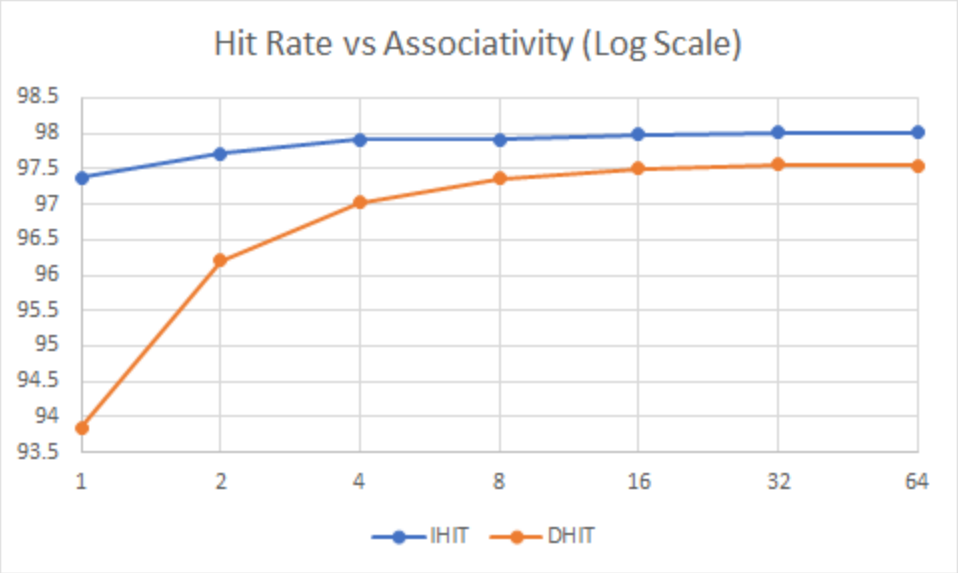
As can be seen from the graphs, the cache hit rate increases with size initially and saturates near a high value eventually, further increasing the cache size does not increase the hit rate substantially. This is because, cache size is only a limiting factor when there are a lot of replacements. When cache size reaches a relatively high value, number of replacements becomes very low and thus increasing cache size further does not change hit rate.

2) Block Size



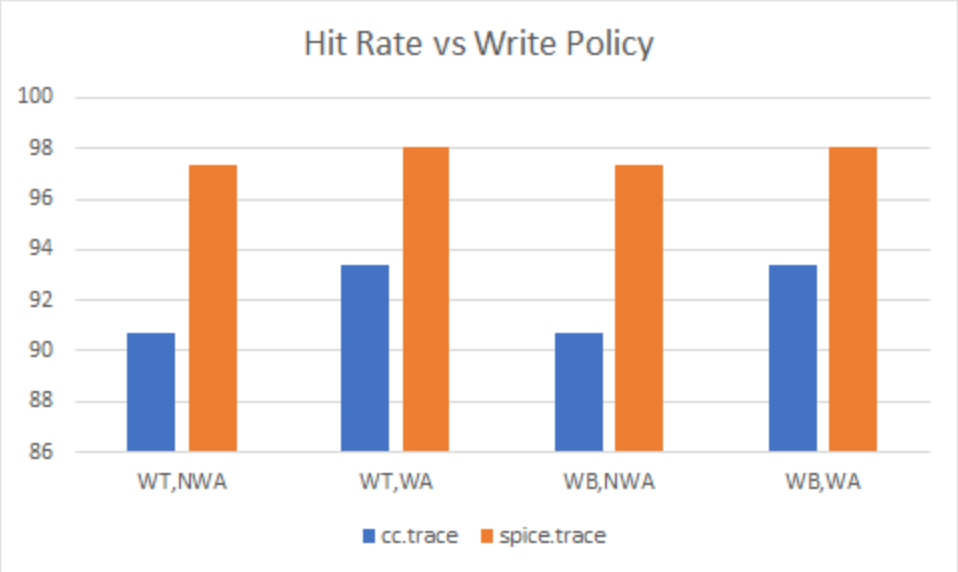
Here too the result obtained fits the model pretty well. We see an increase in data hit rate initially on increasing the block size. This is because of locality in data accesses, thus if we bring back more data with each miss, then there will be fewer misses. When the block size is increased to a very large value, we see the data hit rate drop a lot. This is because now there will be a lot of replacements because the number of blocks becomes too small. Curiously we see that the instruction hit rate is not affected, this is expected since instructions are all in a single sequence so one block suffices to hold them.

3) Associativity



This was done for the cc.trace example. Our hit rates were found to be increasing with associativity. This is expected as a higher associativity implies that each set has more blocks, thus there is less chance of conflict between two addresses. We do pay a higher hardware cost due to a lot of comparators though. Complete associativity gives diminishing returns for large values though as seen here since number of conflicts is reduced significantly by just a few blocks in every set.

4) Write Policy



Here we see that among write allocate and no write allocate policies, the write allocate policy causes a greater percentage of cache hits. No significant difference was found between write back and write through policies. The greater hit rate in write allocate is due to a larger number of data blocks being fetched and replaced. The back/through policy cannot affect the hit rate since it only determines whether a data value is written back to memory every time it is update or not.