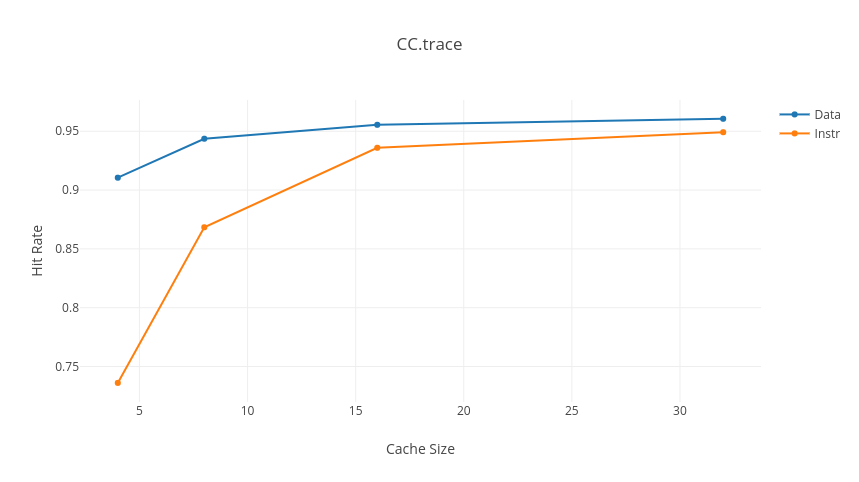
Impact Of Working Set Characterization

In this experiment we show the effect of the cache capacity on the hit rate.

In each case we observe that the hit rate either increases or stays the same as cache size increases. This is because the cache has greater capacity to store values, therefore it has more sets. So the range of indices that can be stored in the cache increases, allowing for more values to be stored, and as a result there are more hits with a higher cache size. But at a point the cache becomes too saturated with sets and has no more need for more sets so the hit rate becomes constant eventually.

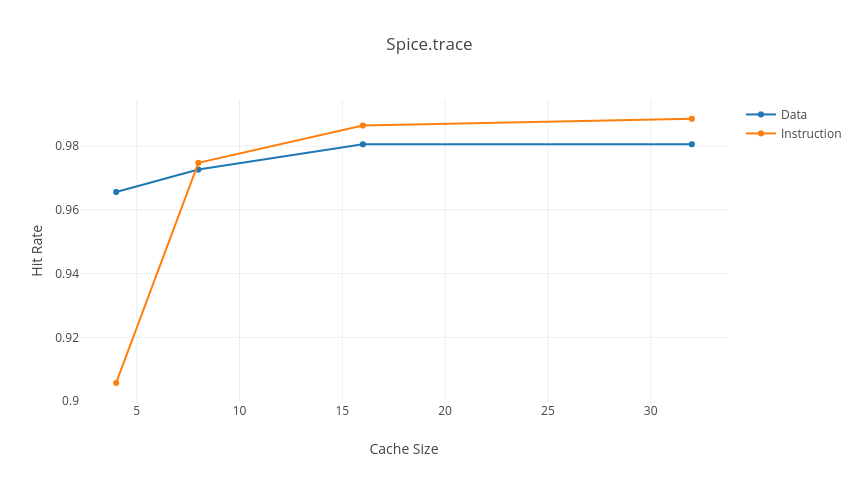
The following graphs show hit rate against cache size for both instruction and data caches in a split cache:

**cc.trace**

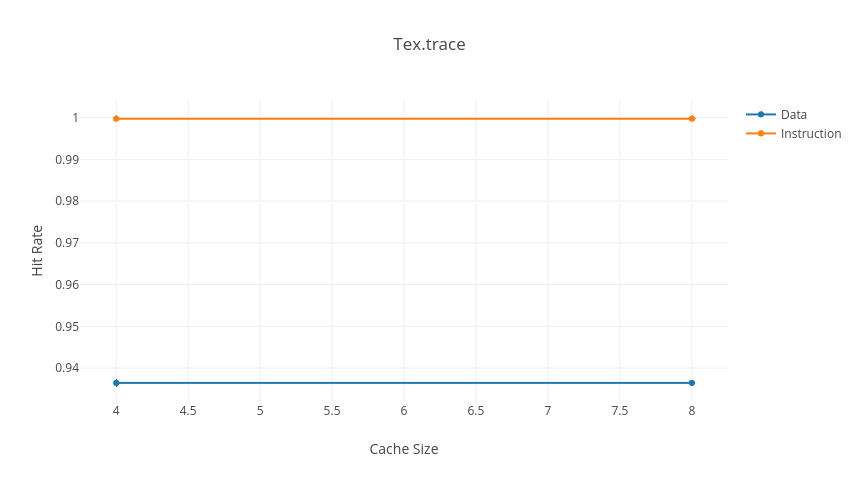
In this graph we observe that, in both cases of instruction and data caches, the hit rate increases with cache size, eventually becoming constant. The increase is greater in the beginning of the graph. Also, increasing the cache size has had a greater effect on the instruction cache by comparison to the data cache.

The total size at which the hit rate becomes constant is 64KB (ie. 32KB for the data cache and 32KB for the instruction cache).

**spice.trace**

In this graph, again both instruction and data caches’ hit rates increase until they become constant, and again the effect has been much greater for the instruction cache than the data cache. But in this case the hit rate of the instruction cache surpassed that of the data cache, despite being initially very low.

Again the total size is 64KB, equally divided between the two parts of the split cache.

**tex.trace**

For tex.trace the hit rate was already constant by the second iteration; the cache size appeared not to make any difference on the hit rate at all.

We can say that the smallest total size the cache was constant at was 8KB – 4KB each for instruction and data.

Impact Of Block Size

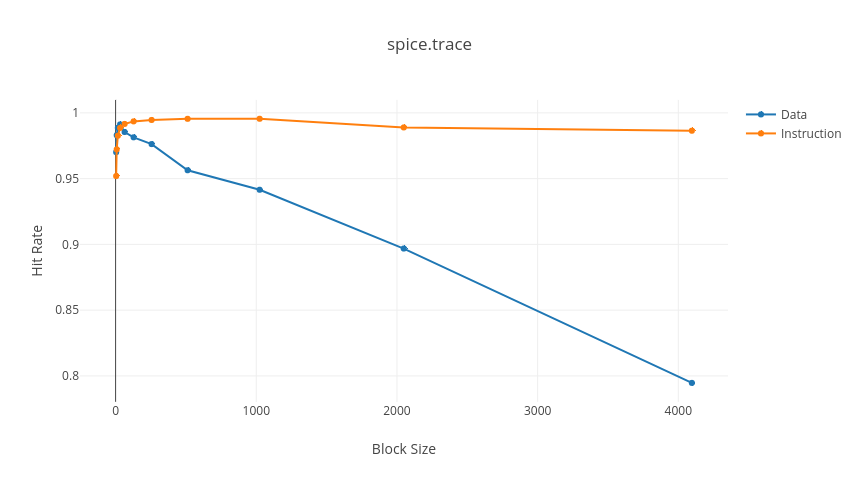
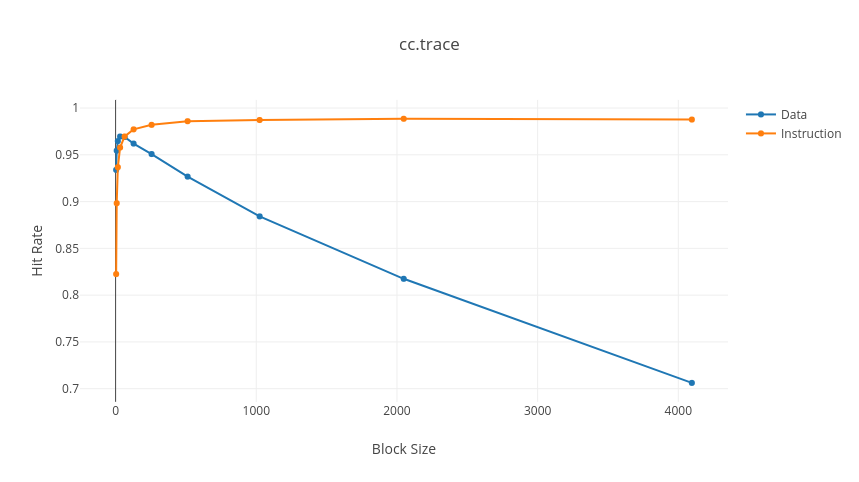
In this experiment we observe the effect that the block size has on the hit rate of the cache.

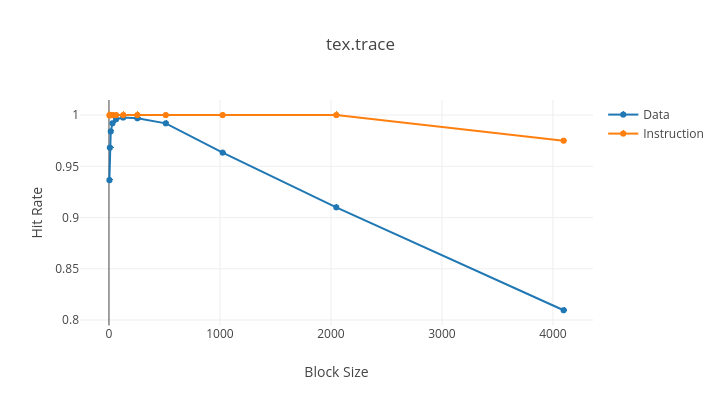
We observe that hit rate increases with cache block size until a point, after which it drops again. The reason for this has to do with spatial locality: when the blocksize is greater, then it means there will be less tags in the cache overall (given a constant cache size, and with each tag having more data stored against it). This increases competition between data with different tags, as fewer tags can actually be stored. As a result the hit rate goes down since there are more misses from accessing data from addresses with different tags.

Optimal block size can depend upon spatial locality, ie. if accesses are being made from relatively close-by data then increasing block size will increase hit rate. This means that compulsory misses are greater for smaller blocksize – however, for greater blocksize, there will be more conflict misses.

The optimal block size is found at the balance between these two points, when the gradient of the graph becomes 0.

The optimal block size may differ between data references and instruction references; this is because of the spatial locality of the accesses, ie. how close or far away accesses are from each other in terms of memory space and time of access. Instructions appear to be less affected by block size, implying that they are mostly close together in the memory and called in a way that they share tags with existing values in the cache.





Impact Of Associativity

In this experiment we observe the effect that the associativity has on the hit rate of the cache.

Hit rate increases with associativity because with greater associativity, more tags are checked for each set/index. So more tags sharing the same index can be stored in the cache.

