**Mazen Amr Eid**

**900161021**

**Khaled Soliman**

**900162031**

**Project 2 Report**

In all of the following graphs, we notice that as we increase the number of bytes to the cache line, the hit rate increases. This is due to the fact that as the cache line increases and using the spatial locality, the data that the program will likely to need will be found in fewer blocks without the need to go to many other blocks to search for the remaining data. This decreases the number of times where the cache needs to pick data from the DRAM (the miss ratio).

In this graph, the hit ratio is high (87 for 8 bytes and increases to 97 for 64 bytes line size). This is due to the fact that the function memGen1 generates linear addresses corresponding to all the possible locations in the DRAM. This implies that there will be misses once the cache is cold started then the following times will majorly be hits. Which implies a high hit ratio.

This memGen2 function generates addresses from 1 to 32\*1024. Yet the cache size is 64 kilobytes. Therefore, all the addresses will be in the cache. This implies 100% hit ratio. But, there will be misses only when filling the cache. Therefore, this reduces the hit ratio to 99.6 for 8 bytes and increases up to 99.96 for 64 bytes line size.

This memGen3 function generates random addresses between 1 and 64\*1024\*1024. Due to this randomness, almost every time will be a miss because spatial and time locality will both have no effect on this case. Therefore, as we notice the hit ratio doesn’t really change (0.985 to 0.1022).

This memGen4 only generates 1024 linear addresses. All of them will be in the cache once cold started. Therefore, the only misses will be only the first time accessing the location but all of the next times will be hits. This is why the hit ratio is the greatest of all the memGen functions (99.987 to 99.997).

Similar to memGen4, memGen5 generates linear addresses but between 1 and 64\*1024 different addresses. Similarly, the hit ratio is very high (99.2 to 99.8). This is because we are benefiting from the spatial locality.

memGen6 has the lowest hit ratio (= 0) for all line sizes. It’s because we are incrementing by 64 the addresses so every time we come to the same location in the cache, the tag will be different and spatial locality has no effect at all in this particular case.

**Test Case:**

If we generate 1024 linear addresses starting from 1 to 1024 (matches memGen4), there will be 1024 misses (for the cold start), then all the rest will be hits because the data would be loaded. If we do this over 1 million itirations, then:

1 000 000 itirations-1024 misses= 998 976 hits

hit ratio = 998 976 \* 100 / 1 000 000 = 99.9 which corresponds to the result we have for the memGen4.