THE AMERICAN UNIVERSITY IN CAIRO

100 YEARS

CSCE230301 - Comp Org. and Assembly Lang Prog

Cache Performance

Names: Kareem Elnaghy, Malek Mahmoud, Mazin Bersy

Dr. Mohamed Shalan

I. Introduction

This project aims to develop and analyze cache performance using a simulator developed in C++. The simulator evaluates the performance of different cache types by generating memory addresses through six memory generator functions. Each function generates 1,000,000 memory references per experiment to simulate various memory access patterns. We measure hit ratios for direct mapped and fully associative caches and then use the data collected to plot graphs. This analysis helps us draw conclusions about the impact of line size and cache type on performance.

II. Data Presentation

This section presents the results of our experiments using two tables and two graphs, one for each cache type. We simulated fully associative and direct mapped cache configurations to examine how different block sizes impact hit ratios across six memory generators (memGen1 to memGen6). **Table 1** presents hit ratios for the fully associative cache with block sizes of 16, 32, 64, and 128 bytes, as visualized in **Figure 1**. Similarly, **Table 2** and **Figure 2** display the hit ratios for the direct mapped cache.

Table 1 Fully Associative Cache

Line Size	memGen1	memGen2	memGen3	memGen4	memGen5	memGen6
16	93.75	99.8464	0.1001	99.9744	99.5904	20.1558
32	96.875	99.9232	0.097	99.9872	99.7952	1.9535
64	98.4375	99.9616	0.0968	99.9936	99.8976	50.9871
128	99.2187	99.9808	0.0969	99.9968	99.9488	75.4861

Figure 1

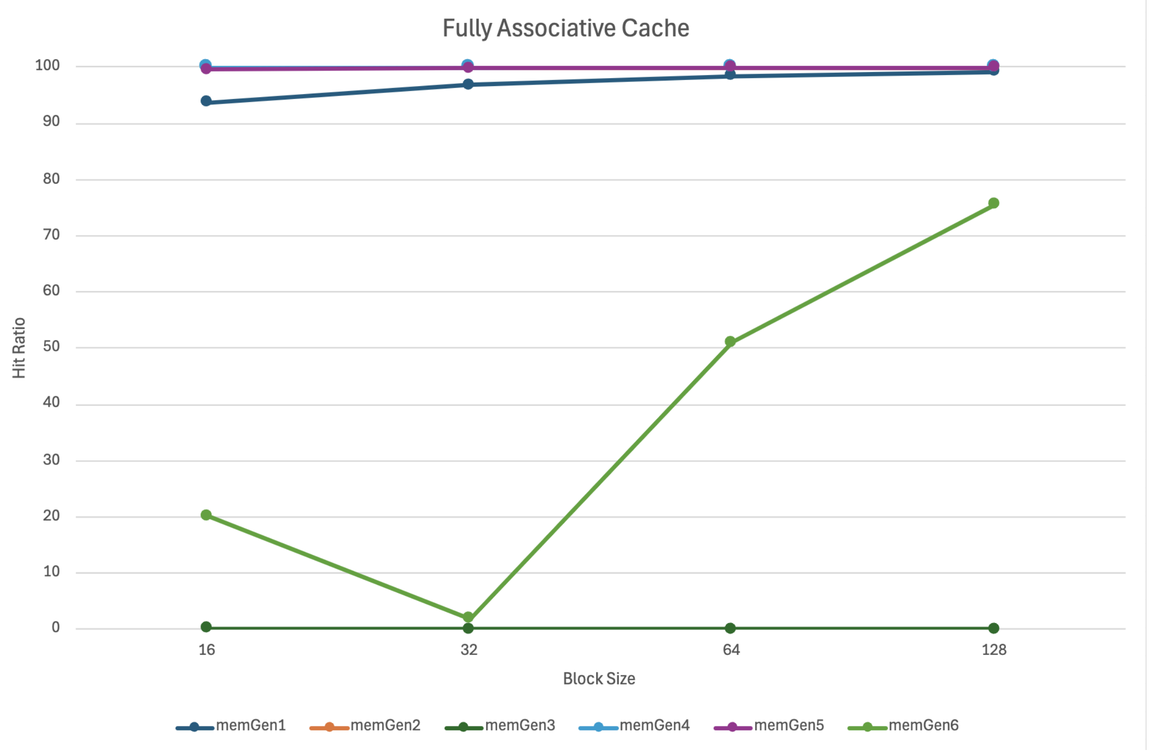
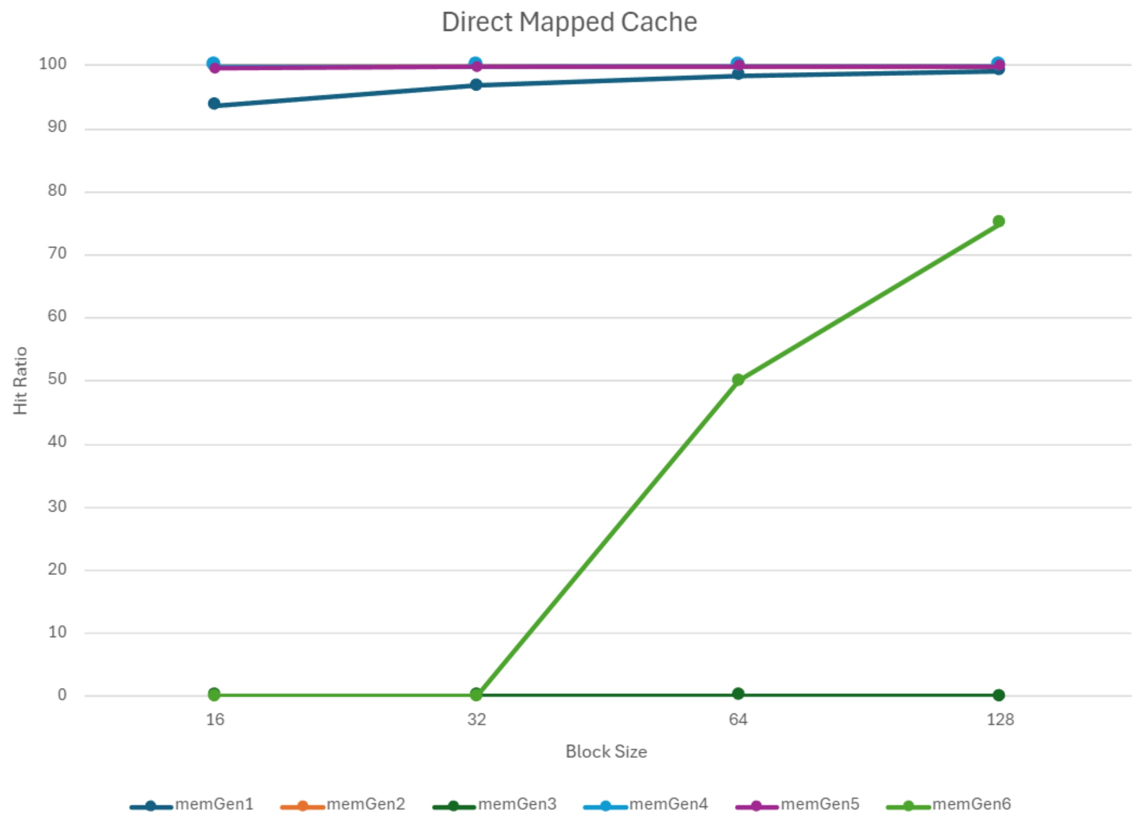


Table 2 Direct Mapped Cache

Line Size	memGen1	memGen2	memGen3	memGen4	memGen5	memGen6
16	93.75	99.8464	0.1021	99.9744	99.5904	0
32	96.875	99.9232	0.1021	99.9872	99.7952	0
64	98.4375	99.9616	0.1023	99.9936	99.8976	49.9999
128	99.2187	99.9808	0.0978	99.9968	99.9488	74.9999

Figure 2



III. Analysis

Memory Generator 1

In this testcase, the addresses are sequentially sent to the cache. It is showcased by the graph that as the line size increases, the hit ratio tends to improve in direct mapping. Initially, when a new line size is accessed for the first time, it results in a compulsory cold start miss. This miss leads to updating the entire line size with the neighboring addresses. Consequently, a series of hits is observed following each miss. To solidify this theory, hits occur after every (line size - 1) iterations. For instance, with a line size of 16, the misses would occur at addresses 0, 16, 32, 48, etc., followed by hits for addresses in between. Once the cache is filled, this pattern continues; however, the nature of misses transitions from compulsory to conflict misses as the cache starts overriding existing lines.

In the case of fully associative it is not much different as we substitute the index and tag are combined into tag. In addition, when the lines are filled up the capacity misses start to take place as there are no more empty lines so the cache overwrites filled blocks at random to replace them. In a fully associative cache, there is no fixed index for each block, and any block can be placed in any cache line. Therefore, the entire address serves as the tag, instead of dividing the address into separate index and tag fields. Initially, when addresses are accessed, compulsory misses (cold misses) occur as the blocks are brought into the cache for the first time. Once the cache lines are filled, capacity misses start to take place. Since there are no more empty lines available, the cache must overwrite existing blocks to handle new ones. This fully associative cache implements the replacement policies of random replacement in order to determine which block to overwrite.

Therefore, while fully associative caches minimize conflict misses, they still experience compulsory and capacity misses. On the other hand, direct-mapped caches experience conflict and compulsory misses. They both deal with the same number of compulsory misses. Moreover, they share the same total number of misses when considering conflict and capacity misses combined. Hence, sharing the same hit ratio.

Memory Generator 2

This generator deals with random addresses submitted into the cache, leading to a high number of compulsory cold misses initially. As the cache lines begin to fill, the hit ratio starts to increase as the same addresses or neighboring ones are accessed repeatedly. Additionally, the observed trend of higher hit ratios with larger line sizes is observed. This improvement occurs because larger line sizes result in fewer cache blocks, as the number of blocks is determined by dividing the cache size by the line size. Fewer blocks reduce the likelihood of cache evictions and increase the probability that subsequent random addresses will fall within the same cache line, thus enhancing the hit ratio.

Due to the address range for the memory generator being below the cache size the FA does not face any capacity misses. Meaning there will exist empty lines. By experimenting, DM doesn’t encounter any conflict misses in this memory generator. These two factors are crucial in the high hit ratio. Therefore, with a high number of iterations and only a few initial cold starts, the hit ratio remains high.

Whether using a direct-mapped or fully associative cache, the result is similar in terms of compulsory cold misses, as both cache types experience the same number of initial cold misses due to the nature of accessing new data. Hence, producing the same numbers in FA and DM.

Memory Generator 3

The very low hit rates for both FA and DM caches in this benchmark can be attributed to the large discrepancy between the DRAM size (64MB) and the cache size (64KB). The memGen3() function generates random memory addresses within the 64MB DRAM and due to the small size of the cache

(1/1024 the size of the DRAM), the chances of that address being in cache is very low. In the FA cache, the flexibility of placing any block in any line does not significantly improve performance when the working set is much larger than the cache size, leading to frequent capacity misses. With the DM cache, because each address maps to just one cache line, there are frequent conflict misses that lower the hit rate. Generally speaking, the address range in use is much larger than the cache size, and few addresses are actually in cache, so the vast majority of memory references bring about a cache miss, regardless of the type of cache.

Variations in hit rates for Fully Associative (FA) and Direct Mapped (DM) caches across different line sizes show distinct patterns that are impacted by cache designs and workload features. The hit rate was highest at 1.001% for the FA cache design when line size was 16 bytes, which slightly descends to 0.968% when line size is increased to 64 bytes and slightly improves to 0.969% when line size becomes 128 bytes. This trend indicates that the smaller the line size, the more effective FA caches become at balancing the count of cache lines and the likelihood of accessing a block in cache, whereas larger line sizes bring in more capacity misses even with more spatial locality. The DM cache design has consistent hit rates in the line size range of 16-64 bytes, with results around 1.021%-1.023%, and then significantly falls to 0.978% when line size becomes 128 bytes, indicating that the number of conflict misses increase notably as the number of cache lines decrease. Results of the experiment show that a DM cache design is most effective with a 64-byte line size, minimizing conflict misses while taking advantage of spatial locality, whereas the FA cache benefits most from a 16-byte line size, providing more lines to reduce capacity misses and effectively managing the workload's spatial locality.

Memory Generators 4 and 5

memGen4 and 5 generate addresses sequentially which are submitted into the cache. In addition, memgen4 generates addresses for a very small range (0 to 4KB) in a cyclic fashion. As the address range is significantly less than the size of cache (64KB), nearly every request will be in the cache and high hit rate is seen for all line sizes (99.9744% for 16 bytes and 99.9968% for 128 bytes) and in both Fully Associative (FA) and Direct Mapped (DM) caches. There is a slight increase in hit rate with larger line sizes which shows that getting more data per miss is beneficial due to the high spatial locality within this small address range. In a DM cache, there are no conflicts due to the small address range which means hit rates are almost perfect, which matches what we see with the FA cache. Due to the address range being so small there are some lines in the cache that are not used which means there will be fewer cold start misses compared to if we were using a large address space.

Likewise, memGen5 is similar to memGen4 but it deals with a larger address space.
It generates addresses sequentially in a range equal to the size of cache (0 to 64KB) in a cyclic fashion. In this case the address space matches exactly to the size of cache, however, since the address generation is cyclic, the addresses are reused often, leading to very high hit rates across all line sizes (99.5904% for 16 bytes and 99.9488% for 128 bytes) and in both FA and DM caches. As line size increases, we see an improvement in hit rate, meaning that larger lines are able to capture more spatial locality, reducing the number of misses. Due to submitting addresses sequentially, the DM and FA results end up identical.

The difference in hit rates between the memory generators can be attributed to their address ranges. memGen4 has a 4KB address range, which is within the 64KB cache. Therefore, memGen4 has such high hit rates (nearly 100%) because the entire address range fits within the 64KB cache, which ensures no cache misses. memGen4 also has a smaller address range, which means fewer cold start misses because there are fewer "empty" cache lines. On the other hand, memGen5 addresses a range that is equal to the cache size, the 64KB number. Hence, extra cold misses are suffered in memGen5 due to the larger address range being dealt with. This analysis underscores the significance of considering both the address range and cache configuration to optimize cache performance for different applications.

Memory Generator 6

The memGen6 function generates addresses by incrementing an address by 32 bytes on each call, cycling within a range of 256K addresses. For the fully associative cache with a block size of 16 bytes, the hit ratio is approximately 20.16%, primarily because the address increments result in every other cache line being accessed, causing frequent cache misses. When the block size increases to 32 bytes, the hit ratio decreases to around 1.95% as each new address maps to a different cache line, resulting in a low chance of reaccessing visited blocks. In contrast, a block size of 64 bytes sees the hit ratio jump to approximately 50.99%. This improvement occurs because each cache line can accommodate two consecutive 32-byte increments, leading to every other access being a hit. The greatest improvement is observed with a block size of 128 bytes, where the hit ratio reaches approximately 75.49%. Here, the larger block size allows four consecutive 32-byte increments to fit within the same cache line, maximizing reuse and the number of hits.

For the same memory generator, using the direct mapped cache, we observe different results. With a block size of 16 bytes, the hit ratio is 0% due to each 32-byte address increment accessing a different cache line, leading to frequent cache misses. Similarly, with a block size of 32 bytes, the hit ratio remains at 0% as each address increment maps to a completely new cache line, resulting in continuous misses. In contrast, when the block size increases to 64 bytes, the hit ratio significantly improves to approximately 50%, as every alternate access hits the same cache line due to the address increments fitting twice within the 64-byte block. The biggest improvement, similar to the fully associative cache, is observed with a block size of 128 bytes, where the hit ratio reaches around 75%. This occurs because four consecutive 32-byte increments fall within the same cache line, resulting in three out of four accesses being hits.

When comparing the two cache types, several differences arise. For smaller block sizes, 16 and 32 bytes, the direct mapped cache results in a hit ratio of 0%, whereas the fully associative cache achieves hit ratios of 20.16% and 1.95%, respectively. This improvement in the fully associative cache is due to its greater flexibility in data placement, which reduces the number of conflict misses. For larger block sizes, 64 and 128 bytes, the fully associative cache also has slightly higher hit ratios, 50.99% and 75.49%, compared to the direct mapped cache's 50% and 75%, for the same reason of enhanced data placement flexibility.

IV. Conclusion

The performance comparison of different memory generators (memGen1 to memGen6) for Direct Mapped (DM) and Fully Associative (FA) caches reveals distinct patterns based on memory access patterns and cache configurations. Both DM and FA caches show improved hit ratios with larger line sizes, especially in sequential access patterns seen in memGen1, memGen4, and memGen5, where FA caches generally minimize conflict misses better than DM caches. However, for random access patterns like in memGen2, both cache types exhibit high hit ratios with minimal performance differences. In cases with a large address range compared to the cache size, such as memGen3, both caches suffer low hit rates due to frequent capacity and conflict misses, with slight variations based on line sizes. MemGen6 demonstrates the benefit of larger line sizes in both cache types, showing improved hit ratios as line sizes increase, although FA caches consistently perform slightly better due to their flexibility in data placement. Overall, while FA caches generally handle conflict misses more efficiently, the performance gap between DM and FA caches narrows under specific memory access patterns and cache configurations.