Project 2 REPORT

Cache Performance

CSCE 231/2303: Computer Organization and Assembly Language Programming
Summer 2024

Laila Sayed- 900223389

Farida Said-900221087

Hoda Hussein-900223388

Experiments Execution

Test case 1:

Goal: Identify the functionality of the code, and check if the output results of the direct mapped-cache and fully associative cache are correct.

Verification: Test case 1 idea is providing some sequential addresses. The sequence of the addresses is starting from address 0 till address 80 in the first row, and the addresses are even addresses divisible by 2, then the sequence goes from A to E also in a double count, then goes back to 100,120…. Etc. The purpose of this is testing the output data then comparing it with manual calculations to check whether it is right or not. For these sequential data, the output is calculated manually as follows: First, the cache size is identified (64Kb in our project), second one of the line sizes (16, 32,64,128 bytes) and cache types (Direct map or fully associative) , then in order to calculate number of misses/hits, we divide the line size by the 32-bit address we are working with, so for the 16 bytes line size, if we divide it by the 32-bit address it won't read the full address so the output will be all misses, for the 32 bytes line size, we will get also all misses output, however for the 64 bytes we have 2 words in each bock, one miss (cold start miss), followed by the one hit from the D-RAM and this happens for the sequential data addresses only, then for the 128 bytes line size we have 4 words in each block, one miss followed by three hits. Then we can divide the number of hits by the total number of elements in the cache to calculate the hit ratio. Accordingly, we compare the output calculated manually to the output calculated by our program to check the validity of the results. Furthermore, in the sequential data the hit ratios for both the fully associative and direct map cache is expected to be the same since we don't use the offset, index or tag in calculating it.

Test case 2:

Goal: Check for the conflict misses that can occur in the direct mapped cache and compare it to the fully associative cache output.

Verification: In this test case, there are some random addresses, and some of them are the same to test the conflict. For Direct map cache the address is divided into the bits for offset, bits for index and bits for the tag. The cache line size is used to calculate the offset bits, for example cache line size equals 128 bytes, we take log base 2 for the 128, the output is 7 which means 7 binary bits for the offset, however since our data is represented in Hexadecimal so the 7 binary bits is approximately 2 bits in HEX. Next, the

remaining bits are for the index and the TAG. For the index, we take log base 2 for the number of lines in each cache and the result is the number of bits for the index. Since we have a 32-bit address, we get the Tag bits by subtracting the offset and index bits from the 32-bits. Now, the program starts with cold start misses, then it takes each address after and compare the index bits, if found in the cache it checks for the tag bits, for the first matching tag the program gives a hit, otherwise it gives a miss and override that address by modifying the tag, so if we checked for the old tag again it will give a miss. Finally, it calculates the hit ratio. On the other hand, for the fully associative, there are no bits for the index, so the 32-bits are divided between the offset and the tag, so the program eliminates the offset and checks for any similar tag, the first found matching tag gives a hit, otherwise it is a miss. Consequently, the two cache types give different hit ratios, and because no conflict miss happens in the fully associative cache, it is expected that it gives a higher hit ratio than the direct map cache.

Test case 3:

Goal: Checks for the output whenever there is a capacity misses in both the fully associative and direct mapped caches

Verification: When the case size is less than any of the line sizes, the programs terminate at any value, not the zero, also it doesn't print the rest of addresses, and this is called a capacity miss because the cache is running out of space. In this case the compiler tries to override the existing data in the data, and there are many ways to do this, however the best replacement is the random lines replacement. Accordingly, only a small part of the data is printed, based on how small the capacity of cache size compared to the provided addresses, so the hit ratio is small, and not correct because we don't have all the hits and misses in the output.

Data Collection

Hit Ratios in Direct Mapping Cache

	Line Size (Bytes)			
memGen()	16	32	64	128

1	93.75%	96.875%	98.4375%	99.2187%
2	99.8464%	99.9232%	99.9616%	99.9808%
3	0.0986%	0.0998%	0.0972%	0.0965%
4	99.9744%	99.9872%	99.9936%	99.9968%
5	99.5904%	99.7952%	99.8976%	99.9488%
6	0%	0%	49.9999%	74.9999%

Hit Ratios in Fully Associative Cache

	Line Size(Bytes)			
memGen()	16	32	64	128
1	93.75%	96.875%	98.4375%	99.2187%
2	99.8097%	99.9022%	99.9525%	99.9767%
3	0.1%	0.0986%	0.0996%	0.098%
4	99.9731%	99.9865%	99.9933%	99.9966%
5	98.5682%	99.2797%	99.6639%	99.8248%
6	20.0801%	1.9648%	50.9807%	75.4829%

Data Analysis

Direct Mapping Cache

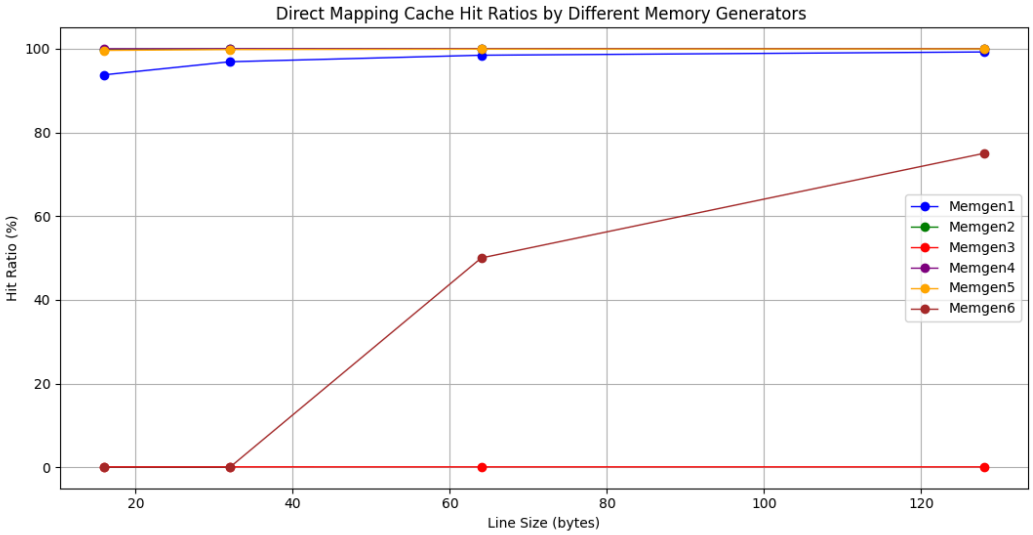

Figure 1:Direct Mapping Cache Hit Ratios by Different Memory Generators line graph

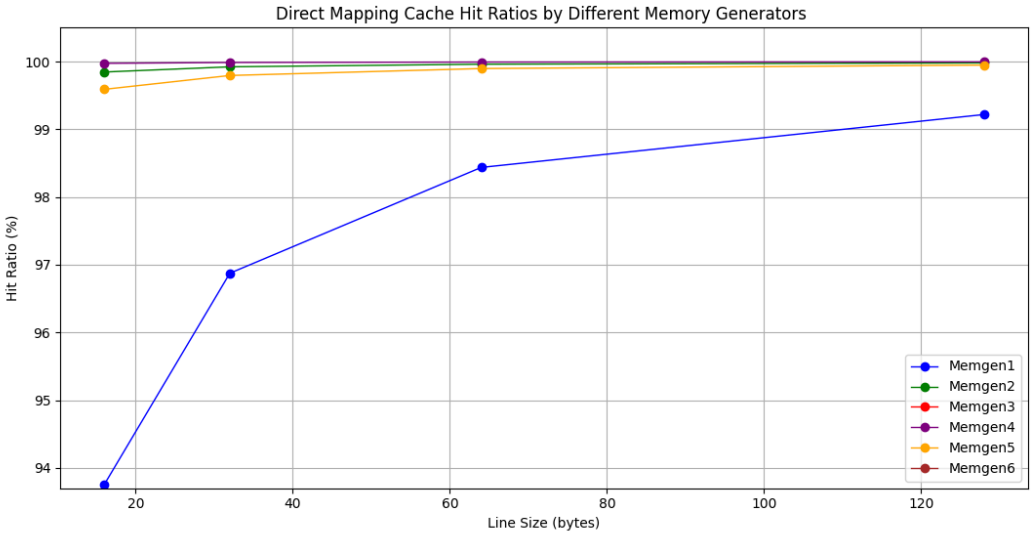

Figure 1:Direct Mapping Cache Hit Ratios by Different Memory Generators line graph(Zoomed in)

Fully associative Cache

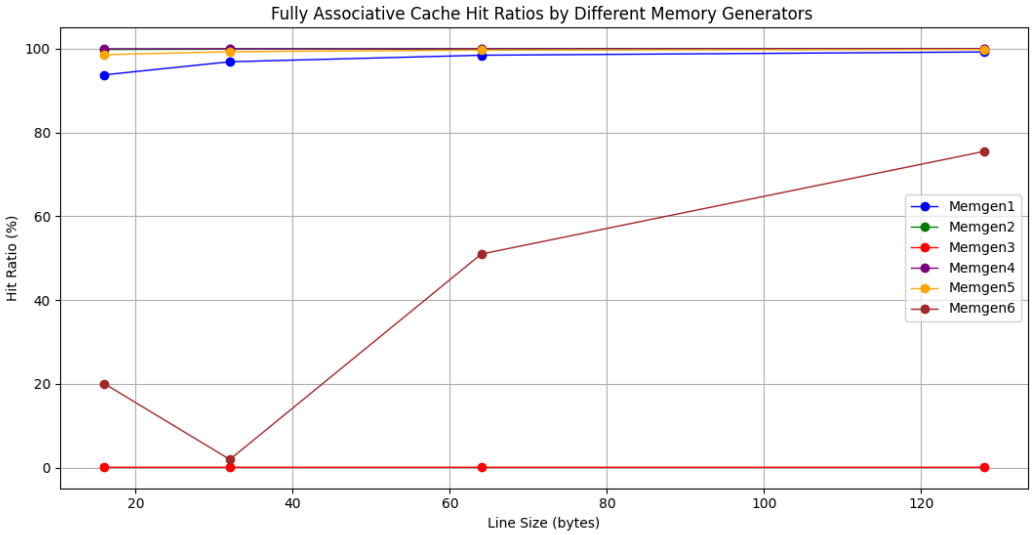

Figure 1:Fully Associative Cache Hit Ratios by Different Memory Generators line graph

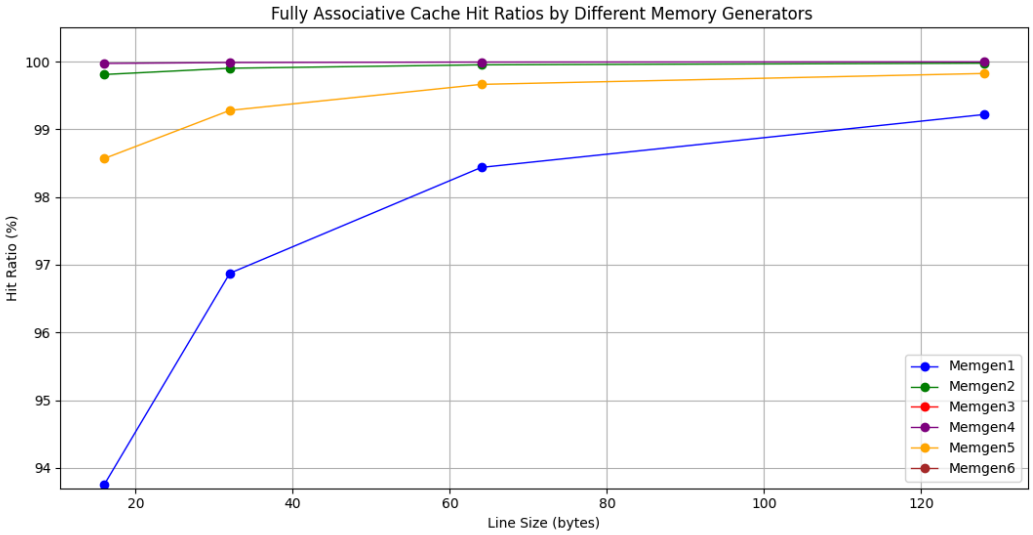

Figure 1:Fully Associative Cache Hit Ratios by Different Memory Generators line graph(Zoomed in)

Graphs Interpretation:

- The general trend is that increasing the line size increases the Hit ratio because it makes better use of the spatial locality principle.
- The increase in hit ratio is expected to be more significant when the addresses are accessed sequentially as in the case of memgen1,memgen4, and memgen5.
- For a fixed line size, misses depend on the relation between the number of lines in the cache and the number of lines needed to represent a given range of addresses. That's why although memgen1,memgen4, and memgen5 are all accessed sequentially, we can see that the hit ratio is smaller in memgen1 because the number of lines needed to represent all addresses given by memgen1 is greater than the number of lines in the cache, which increases the probability of misses. On the other hand, the number of lines needed to represent memgen4 is less than the number of lines in the cache, and the number of lines needed to represent memgen5 is equal to the number of lines in the cache. Thus, their only types of misses are compulsory misses, and that explains why they have very high and very close hit ratios.

- Memgen 2 and 3 are both nonsequential, however, it's evident that the hit rate in memgen2 is way higher compared to mimgen3, this is because for a fixed line size the number of lines needed in memgen2 is less than the number of lines in the cache, so the only misses are the compulsory misses. On the other hand, the number of lines needed in memgen3 is way greater than the number of lines in the cache, so we will have a lot of conflict and capacity misses.

- For memgen6 in DM, when the line size is 16 or 32, we have a hit ratio of 0 that is because the addresses are being accessed with an increment of 32 every time, so it will never be a hit and this access pattern doesn't make good use of spatial locality. However line size 64, it's expected to have a miss followed by a hit, so we should have a 50% hit rate, and the same goes for line size 128, the hit rate is slightly less than 75% because we have 1 miss followed by 3 hits.

- For memgen6 in FA when the line size is 16, the cache still doesn't exploit spatial locality. However, we have a hit ratio of 20.0801% because we don't have conflict misses as in the direct mapping. Still, we have compulsory and capacity misses because the total number of lines needed to represent the addresses generated by memGen6 is more than the number of lines in the cache by 1. When the line size is 32, we still don't exploit spatial locality and we also increase the capacity miss as the cache would be filled and start replacement faster. That's why we have a very low hit ratio of 1.9648%. Line sizes 64 and 128 Are the same as direct cache mapping.

CONCLUSION

- In the memGens where data are accessed sequentially (1,4, and 5), we have the exact same hit ratios for both DM cache and FA.
- In the memGens where data are accessed randomly(2,3), there is a slight variation between the hit ratios for both DM cache and FA.
- Memgen6 is an example of how conflict misses could significantly reduce the hit ratio, particularly when the spatial locality principle isn't being used in how the data are accessed.