Computer Architecture(CS2323)

HOMEWORK-4(CACHE MISS EXPERIMENTS)

NAME: SAKSHAM

ROLL NO: AI22BTECH11024

Introduction:

Efficient cache management is paramount in enhancing computer system performance. This report delves into a series of cache miss experiments conducted using the Ripes simulator. The objective is to scrutinise the impact of diverse cache configurations on hit/miss rates when executing distinct assembly programs.

Q1) L1: .dword 8,8

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
1	0	2	0.7424	49	17	66
2	0	2	0.7424	49	17	66
3	0	2	0.7424	49	17	66
3	0	3	0.8636	57	9	66
3	0	4	0.9242	61	5	66
3	0	5	0.9545	63	3	66
3	0	2	0.7424	49	17	66
3	1	2	0.7424	49	17	66

Explanation:

- 1 line, 0 ways, 2 blocks: This configuration maintains a consistent hit rate of 0.7424 across different experiments. Here, due to the limited number of lines (just 1), the cache doesn't effectively exploit spatial locality. Consequently, 17 out of 66 accesses result in misses as the cache needs to be filled initially.

- 2 lines, 0 ways, 2 blocks: Similar to the previous setting, the hit rate remains steady at 0.7424. With 2 lines, it marginally enhances the cache's ability to capture a bit more spatial locality, yet the overall behaviour is consistent with 17 misses out of 66 accesses.
- 3 lines, 0 ways, varying blocks: As the number of blocks increases from 2 to 5 within 3 lines, there's a noticeable enhancement in hit rates. With 2 blocks, the hit rate stays at 0.7424, but with 3 blocks, it increases to 0.8636, and further to 0.9545 with 5 blocks. The increased number of blocks allows more data to be stored, reducing the number of misses. This highlights the importance of a higher block count in improving cache performance.
- 3 lines, 1 way, 2 blocks: Moving from 0 to 1 way, while maintaining 2 blocks, doesn't impact the hit rate or the access pattern. The hit rate remains constant at 0.7424, as the increase in ways doesn't significantly alter cache behaviour in this setup. Similar to the previous cases, there are 17 misses out of 66 accesses.
  In essence, the observations emphasise that an increase in the number of blocks generally leads to improved hit rates by allowing the cache to hold more data and better exploit spatial locality. Changes in lines and ways, however, do not significantly affect hit rates or access patterns in these specific configurations.

L1: .dword 16, 16

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
1	0	2	0.7481	193	65	258
2	0	2	0.7481	193	65	258
3	0	2	0.7481	193	65	258
3	0	3	0.8721	225	33	258
3	0	4	0.9341	241	17	258
3	0	5	0.9651	249	9	258
3	0	2	0.7481	193	65	258
3	1	2	0.7481	193	65	258

EXPLANATION:

- 1 line, 0 ways, 2 Blocks: In the (.dword 16, 16) setup, similar to the (.dword 8, 8) configuration, this setting maintains a consistent hit rate. However, due to the larger data size (16 instead of 8), the cache exhibits a slightly improved hit rate of 0.7481, indicating a modest enhancement in cache efficiency.

- 2 lines, 0 ways, 2 Blocks: Like the previous setup, increasing to 2 lines within the (.dword 16, 16) setup maintains a comparable hit rate, reinforcing that the alteration in lines doesn't significantly influence cache performance in this context.
- 3 lines, 0 ways, Varying Blocks: Increasing the number of blocks within the (.dword 16, 16) setup displays a comparable trend to the (.dword 8, 8) configuration. However, the hit rates for each block count are slightly higher in the (.dword 16, 16) setup, showcasing that larger data sizes have a positive impact on cache performance.
- 3 lines, 1 way, 2 Blocks: Transitioning from 0 to 1 way, while keeping 2 blocks, maintains consistent hit rates in both configurations. The increase in ways doesn't notably influence cache behavior in either setup, reiterating that the change in ways doesn't significantly affect hit rates or access patterns.

Comparatively, the (.dword 16, 16) observations align closely with the trends observed in the (.dword 8, 8) configuration, showcasing slight enhancements in hit rates across various setups. The larger data size in the (.dword 16, 16) configuration seems to marginally improve cache efficiency in each scenario, maintaining the fundamental trends observed in the smaller data size configuration.

L1: .dword 8, 8

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
1	0	2	0.0303	2	64	66
2	0	2	0.0303	2	64	66
3	0	2	0.04545	3	63	66
3	0	3	0.8636	57	9	66
3	0	4	0.9242	61	5	66
3	0	5	0.9545	63	3	66
3	0	2	0.04545	3	63	66
3	1	2	0.7424	49	17	66

EXPLANATION:

- 1 line, 0 ways, 2 Blocks: This setup demonstrates an extremely low hit rate across the board. With just one line and two blocks, the cache proves insufficient to accommodate the required data, resulting in a mere 2 hits out of 66 accesses. As a consequence, 64 out of 66 accesses incur misses due to the limited cache capacity.

- 2 lines, 0 ways, 2 Blocks: Maintaining the same parameters but doubling the lines, this setup continues to display a very low hit rate i.e. 0.0303. The increase in lines doesn't seem to improve cache performance due to the limited number of blocks and ways.
- 3 lines, 0 ways, Varying Blocks: As the number of blocks increases from 2 to 5 within 3 lines, a noticeable trend emerges. With 2 blocks, the hit rate slightly improves from the initial configuration but remains relatively low. However, once the number of blocks exceeds 2, the hit rate substantially increases, highlighting the significance of a larger block count in enhancing cache efficiency.
- 3 lines, 1 way, 2 Blocks: Transitioning from 0 to 1 way within the 3 lines and 2 blocks setting notably increases the hit rate. While the hit rate is not as high as observed in other configurations, introducing a single way within the cache results in a marked improvement compared to the initial configurations.

L1: .dword 16, 16

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
1	0	2	0.007752	2	256	258
2	0	2	0.007752	2	256	258
3	0	2	0.007752	2	256	258
3	0	3	0.007752	2	256	258
3	0	4	0.01163	3	255	258
3	0	5	0.9574	247	11	258
3	0	2	0.007752	2	256	258
3	1	2	0.007752	2	256	258

EXPLANATION:

- 1 line, 0 ways, 2 Blocks: This setup experiences an exceptionally low hit rate, with just 2 hits out of 258 accesses. Compared to the equivalent L2 (8, 8) setup which had a hit rate of 0.0303, this configuration shows a significant drop in performance.
- 2 lines, 0 ways, 2 Blocks: Similar to the previous setting, this configuration also maintains a very low hit rate of 0.007752, reflecting just 2 hits out of 258 accesses. It indicates consistent and poor performance across different settings.
- 3 lines, 0 ways, 2 Blocks: This configuration sustains a similarly low hit rate, showing only 2 hits out of 258 accesses. The addition of an extra line doesn't positively affect the hit rate, indicating the cache's inefficiency in improving performance with added lines in this setup.

- 3 lines, 0 ways, 3 Blocks: With an increase in blocks, this configuration also displays a low hit rate, reflecting 2 hits out of 258 accesses. Despite the addition of one more block compared to the 2-block setup, the hit rate remains consistently poor.
- 3 lines, 0 ways, 4 Blocks: An increase to 4 blocks results in a slightly higher hit rate of 0.01163, showcasing 3 hits out of 258 accesses. However, the improvement in hit rate is still considerably lower compared to the equivalent configuration in L2 (8, 8), which had a hit rate of 0.9242.
- 3 lines, 0 ways, 5 Blocks: This configuration demonstrates a notably higher hit rate compared to other setups, with 247 hits out of 258 accesses.
- 3 lines, 1 way, 2 Blocks: This configuration sustains an extremely low hit rate, similar to the 1 line, 0 ways, 2 blocks setup, with only 2 hits out of 258 accesses.

Across various configurations, the (.dword 16, 16) setup consistently demonstrates notably lower hit rates compared to the (.dword 8, 8) setup. Each configuration in the (16, 16) setup exhibits a substantial decrease in hit rates compared to its equivalent in the (8, 8) setup.
In Program 2, the observed pattern suggests a significant presence of temporal locality. Despite variations in cache configurations, the hit rates tend to remain consistent or exhibit marginal changes. This stability across configurations indicates that Program 2 leverages temporal locality effectively, meaning that recently accessed memory locations are likely to be accessed again in the near future. As a result, the cache is able to retain and utilise data previously accessed, leading to a consistent or marginally changing hit rate across different cache setups. This observation underlines the strong influence of temporal locality in Program 2's memory access patterns.

Question 2)
Program 1 (.dword 8, 8):
Write through, write allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.7424	49	17	66

Write through, No write Allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.04545	3	63	66

Write Back, Write Allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.7424	49	17	66

Write Back, No write Allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.04545	3	63	66

CONCLUSION:
The utilisation of different write policies—write-through and write-back—yielded distinct impacts on cache performance under the provided cache configuration. When employing the write-through policy, regardless of write allocation or not, the hit rate remained consistent, indicating that this policy didn't significantly influence caching effectiveness within this setup. However, the write-back policy showcased identical performance, maintaining the same hit rate and access patterns. Both write-through without write allocation and write-back without write allocation exhibited similar low hit rates, emphasising that the absence of write allocation significantly impacted cache performance. Overall, in this specific scenario and configuration, the choice between write-through and write-back policies didn't substantially alter cache behaviour.

Program 1(.dword 16, 16):
Write through, write allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.7481	193	65	258

Write through, No write Allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.01163	3	255	258

Write Back, Write Allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.7481	193	65	258

Write Back, No write Allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.01163	3	255	258

CONCLUSION:
Similar to the observations with the previous cache size configuration (8, 8), the performance remains consistent across different write policies—write-through and write-back. Write-through with or without write allocation and write-back with or without write allocation produced similar hit rates and access patterns, showing that the choice of write policy did not significantly affect cache behavior in this specific configuration. The low hit rates in cases without write allocation emphasize the impact of this factor on cache efficiency, while the overall performance between write-through and write-back remained consistent.

Program 2(.dword 8, 8):
Write through, write allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.7424	49	17	66

Write through, No write Allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.04545	3	63	66

Write Back, Write Allocate

Number of	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses

lines						
5	0	2	0.7424	49	17	66

Write Back, No write Allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.04545	3	63	66

CONCLUSION:
Similar to Program 1, Program 2 also showcases consistency in its performance across different write policies and cache configurations. Write policies like write-through and write-back, with or without write allocation, resulted in similar hit rates and access patterns in this specific configuration. Notably, the absence of write allocation leads to significantly lower hit rates due to frequent cache misses, highlighting the importance of this feature in optimising cache performance. However, irrespective of the write policy applied, the cache exhibited consistent behaviour in handling accesses for Program 2.

Program 2(.dword 16, 16):
Write through, write allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.01163	3	255	258

Write through, No write Allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.01163	3	255	258

Write Back, Write Allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses

5	0	2	0.01163	3	255	258

Write Back, No write Allocate

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.01163	3	255	258

FINAL CONCLUSION:
In evaluating the cache behaviours of Program 1 and Program 2 across various configurations and write policies, distinct patterns emerged.
The distinctive behaviour observed between Program 1 and Program 2 can be attributed to their inherent memory access patterns. Program 1 exhibits varying cache sensitivities due to its more erratic memory access behaviours, causing fluctuations in hit rates and access outcomes across different cache configurations and write policies. Conversely, Program 2 showcases a more consistent and predictable memory access pattern, resulting in minimal impact from changes in cache settings or write policies. This difference underscores how Program 1's memory access characteristics are more influenced by the cache's structure and policies, while Program 2's behaviour remains relatively stable and less affected by alterations in cache configurations.

Program 3
Q3)(.dword 8, 8)

(a) 32 entry- 4-word direct mapped

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.01538	2	128	130

(b) 32 entry 4 word 2 way set associative

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
4	1	2	0.7385	96	34	130

c) 32 entry 4 word fully associative

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
0	5	2	0.7385	96	34	130

Program 3

Q3)(.dword 16, 16)

(c) 32 entry- 4-word direct mapped

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
5	0	2	0.06809	35	479	514

(d) 32 entry 4 word 2 way set associative

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
4	1	2	0.7471	384	130	514

c) 32 entry 4 word fully associative

Number of lines	Number of ways	Number of Blocks	HIT RATE	NUMBER OF HITS	Number of misses	Total Accesses
0	5	2	0.7471	384	130	514

CONCLUSION:

The observations from the experiments on Program 3 with different cache configurations reveal intriguing insights into its memory access patterns. When considering an (8, 8) configuration, the direct-mapped cache shows a relatively low hit rate of 0.01538 due to limited associativity. However, transitioning to a 2-way set associative cache demonstrates a significant improvement in the hit rate, marking a notable increase to 0.7385. The fully associative cache, consistent with the 2-way set associative, showcases a similar hit rate and better utilisation of cache resources. In the (16, 16) configuration, the direct-mapped cache's hit rate experiences an increase to 0.06809, highlighting some improvement from the previous direct-mapped (8, 8) setup. The 2-way set associative cache again demonstrates its efficiency, delivering a robust hit rate of 0.7471, similar to the fully associative cache in this setup. These results emphasise the impactful role of associativity in enhancing cache performance, especially evident in the significant boost observed in 2-way set associative and fully associative caches, regardless of the memory block size.