Allen Ren & Jed Mendoza Alcantara

CS-465-001

Professor Zhong

5/3/2023

Homework 4: Part 2

Group 1: Directed-Mapping

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Blocks | Cache Block Size | Replacement Policy | Hit Rate |
| 8 | 2 | LRU | 18% |
| 8 | 4 | LRU | 59% |
| 8 | 8 | LRU | 80% |
| 8 | 16 | LRU | 90% |

Explanation:

In Group 1, we had variable cache block sizes that utilizes Least Recently Used. An increase in cache block size, led to an increase in the hit rate due to spatial locality. An increase in cache block size, means more memory accesses can be served from the cache in a single cache hit. This is represented as the offset.

Group 2: Directed-Mapping

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Blocks | Cache Block Size | Replacement Policy | Hit Rate: |
| 8 | 8 | LRU | 80% |
| 16 | 8 | LRU | 80% |
| 32 | 8 | LRU | 95% |

Explanation:

In Group 2, we had variable number of memory blocks that utilizes LRU. Although there was no increase in hit rate from 8 to 16, there was a greater hit rate from 16 to 32. Eight, sixteen, and thirty two are 2^3, 4, and 5, respectively. This is represented as the index. The differences in hit rate can best be explained because the total number of memory blocks increased: The requested memory block was more likely be stored in memory and not replaced.

Group 3: Directed-Mapping

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Blocks | Cache Block Size | Replacement Policy | Hit Rate |
| 16 | 8 | LRU | 80% |
| 8 | 16 | LRU | 90% |
| 4 | 32 | LRU | 91% |
| 2 | 64 | LRU | 95% |

Explanation:

In this group, both number of blocks and cache block size were variable with an inverse relationship: We observed an increasing hit rate with increasing block size despite a static size of cache, 16 \* 8 == 2 \* 64. This can be explained due to the initialization of the cache: compulsory misses. When the cache is initialized, smaller block sizes means more instances of caching, and each new instance is a compulsory miss.

Group 4: Fully Associated

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Blocks | Cache Block Size | Replacement Policy | Hit Rate |
| 8 | 4 | LRU | 59% |
| 8 | 4 | Random (3 Times) | (59% + 60% + 59%)/3 = 59.3% |
| 4 | 8 | LRU | 80% |
| 4 | 8 | Random (3 Times) | (80% + 80% + 80%)/3 = 80% |

Explanation:

In Group 4, we used a Random and Least Recently Used. The hit rate were similiar for both replacement policies, but what we needed to observe was when we inverted the number of blocks and cache block size. Similiar to group 3, initializing a cache is an expensive process where larger cache block size lead to a higher rate than smaller cache blocks even with different replacement policies.

Group 5:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Placement | # of Blocks | Block Size | Replacement | Hit Rate |
| Directed - Mapping | 16 | 8 | LRU | 80% |
| Fully Associative | 16 | 8 | LRU | 93% |
| 1 – Way Set | 16 | 8 | LRU | 80% |
| 2 – Way Set | 16 | 8 | LRU | 80% |
| 4 – Way Set | 16 | 8 | LRU | 80% |
| 8 – Way Set | 16 | 8 | LRU | 93% |

Explanation:

In Group 5, we saw the best performance from Fully Associative and 8-Way Set. One of the big problems with direct mapping is conflict misses, which account for the differences between Direct Mapping and Fully Associative/8-Set. This is likely because fully associative and set associative mappings can better utilize the cache by allowing data to be stored in any available cache slot, while direct mapping can suffer from conflict misses that cause data to be evicted prematurely; however, fully associative and [set associative] can better utilizes the cache because data can go anywhere and [within set]. We do see differences between the performances between the sets, but this can be explained because it was able to store more memory blocks than the other sets since blocks = N(N>=2). Fully associative is essentially when N-way is equal to # of Blocks.