**Part II: Associative Mapped Cache**

1. Execute the above program by setting block size to 2, 4, 8, 16 and 32 for cache size = 8, 16 and 32. Record the observation in the following table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LRU Replacement Algorithm** | | | | | |
| **Block Size** | **Cache Size** | **# Hits** | **# Misses** | **% Miss Ratio** | **% Hit Ratio** |
| 2 | 8 | 334 | 409 | 55.05% | 44.95% |
| 4 | 324 | 419 | 56.39% | 43.61% |
| 8 | 384 | 359 | 48.32% | 51.68% |
| 2 | 16 | 515 | 228 | 30.69% | 69.31% |
| 4 | 384 | 359 | 48.32% | 51.68% |
| 8 | 417 | 326 | 43.88% | 56.12% |
| 16 | 490 | 253 | 34.05% | 65.95% |
| 2 | 32 | 642 | 101 | 13.59% | 86.41% |
| 4 | 567 | 176 | 23.69% | 76.31% |
| 8 | 501 | 242 | 32.57% | 67.43% |
| 16 | 509 | 234 | 31.49% | 68.51% |
| 32 | 594 | 149 | 20.05% | 79.95% |

The color coding for miss and hit ratio columns is based on the hit ratio column  
Highest hit ratio is darkest shade of blue.  
Lowest hit ratio is darkest shade of red.  
All other values are in between these two bounds with white being the moderate.

1. Plot a single graph of Cache hit ratio Vs Block size with respect to cache size = 8, 16 and 32. Comment on the graph that is obtained.  
     
     
     
   For the given program i.e. Selection Sort; executing the program on CPU-OS Simulator with an LRU replacement algorithm and only using the d-cache (caching only data) gave the results as plotted above. The following observations can be made from the graph.

* ***Irrespective of the block size, increasing cache size gives higher hit ratio.***
* There is a slight variation in hit ratio for a given cache size when we change block sizes, but it seems that block size = cache size or block size = 2 both work better than any other intermittent block size.
* ***In general, increasing block size for a given cache size improves the hit ratio. This could be observed post block size = 4 in cache sizes = [8,16] line plots and post block size = 8 in cache size = 32 line plot respectively.***
* Theoretically, increasing block size means spatial locality is better utilized and we generally see an improvement in the hit ratio for algorithms which use spatial locality the most. The initial decrease in hit ratio for cache sizes of 16 and 32 is an interesting observation and we’re not yet into sorting algorithms, so I can’t comment on the same as yet; but that is something we might be able to answer after sorting algorithms ae covered in DSAD. For now, I can only think that when block size = 2, we are loading two elements from main memory and the first out of those two is always the lowest in that group given the sequential nature of the array [15, 20 | 8, 80 | 30, 35] and selection sort continually tries to find minimal element iteratively from what we could gather from internet; so maybe it has something to do with that.

1. Fill up the following table for three different replacement algorithms and state   
   which replacement algorithm is better and why?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Replacement Algorithm: Random** | | | | |
| **Block Size** | **Cache Size** | **# Misses** | **# Hits** | **Hit Ratio** |
| 2 | 4 | 434 | 309 | 41.59% |
| 2 | 8 | 370 | 373 | 50.20% |
| 2 | 16 | 247 | 496 | 66.76% |
| 2 | 32 | 104 | 639 | 86.00% |
| 2 | 64 | 65 | 678 | 91.25% |
| **Replacement Algorithm: FIFO** | | | | |
| **Block Size** | **Cache Size** | **# Misses** | **# Hits** | **Hit Ratio** |
| 2 | 4 | 453 | 290 | 39.03% |
| 2 | 8 | 410 | 333 | 44.82% |
| 2 | 16 | 296 | 447 | 60.16% |
| 2 | 32 | 105 | 638 | 85.87% |
| 2 | 64 | 82 | 661 | 88.96% |
| **Replacement Algorithm: LRU** | | | | |
| **Block Size** | **Cache Size** | **# Misses** | **# Hits** | **Hit Ratio** |
| 2 | 4 | 453 | 290 | 39.03% |
| 2 | 8 | 409 | 334 | 44.95% |
| 2 | 16 | 228 | 515 | 69.31% |
| 2 | 32 | 101 | 642 | 86.41% |
| 2 | 64 | 79 | 664 | 89.37% |

While random seems to be performing at par with the other two replacement algorithms, LRU in my opinion is the algorithm to go with. I have provided my reasoning for the same after plotting the graph below.

1. Plot the graph of Cache Hit Ratio Vs Cache size with respect to different replacement   
   algorithms. Comment on the graph that is obtained.

It could be seen that for the given program, all three algorithms are performing on a comparatively close scale. But still we can observe that

* The choice of replacement algorithm has a reasonably significant (if not a very huge) impact on the cache hit ratio when the cache size is very small.
  + In that too, Random seems to perform better on average than LRU & FIFO for small cache sizes.
  + For small size cache, random replacement algorithm for programs like selection sort seem to be performing well; however it’s not deterministic in nature so it’s possible that we might get lower hit ratio in some cases.
  + ***The performance improvement of random over LRU strictly speaking is not too high. I would therefore prefer LRU knowing it’s deterministic nature even for smaller cache sizes.***
* As the cache size increases, the choice of replacement algorithm seems to no longer make a big impact on the cache hit ratio. In such cases where cache size is high, we can consciously try to avoid random and go with either of the other two to avoid any stochasticity and go for either LRU or FIFO since the performance is at par if not significantly lower than Random replacement algorithm.
* ***Logically it makes sense to use LRU as it will replace those blocks which have been in the cache longest without having been accessed; irrespective of whatever the cache size be.***