Assignment 3: Cache Simulation

Aastha A K Verma (2022CS11607)

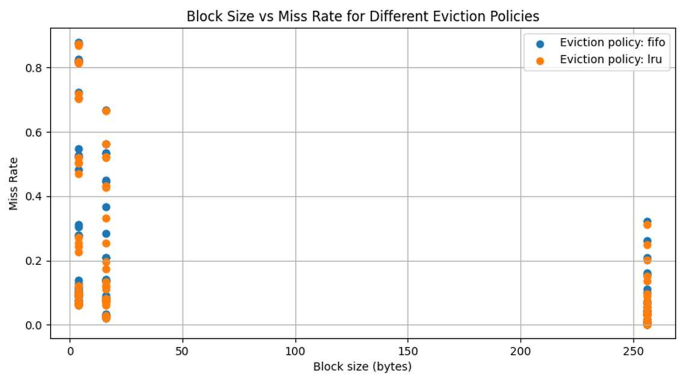
1 Testing Procedure

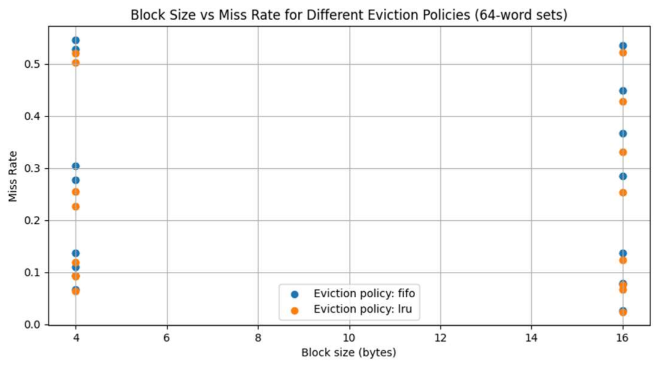
A Python script was written to simulate Cache functioning for different combinations of parameters.
Number of sets ranged from [1, 2, 4, 16, 256].
Associativity ranged from [1, 2, 4, 16, 256].
Block size (bytes) ranged from [4, 16, 256].
Plots of various parameters and results were plotted, including hit rate and total cycles. The following patterns were observed.

2 Observations and Results

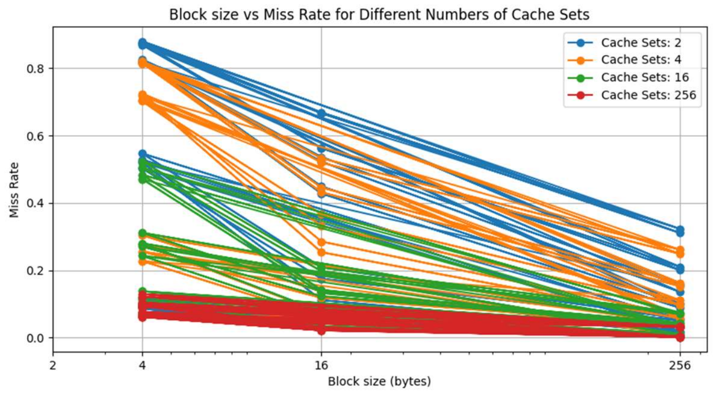
A. Eviction Policies

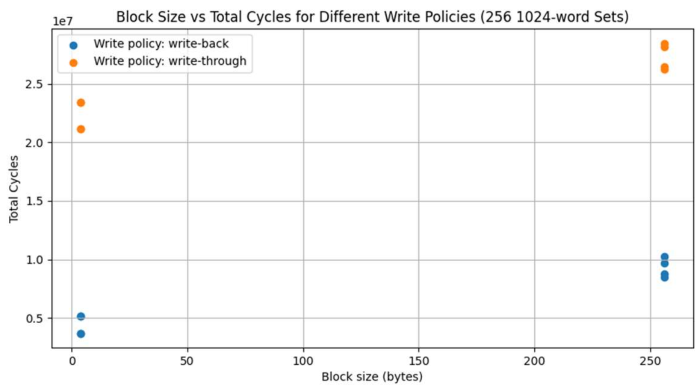
Based on the graphs, LRU seems to give better performance on miss rate.





B. Block Size and Write Policy (write-back/write-through):

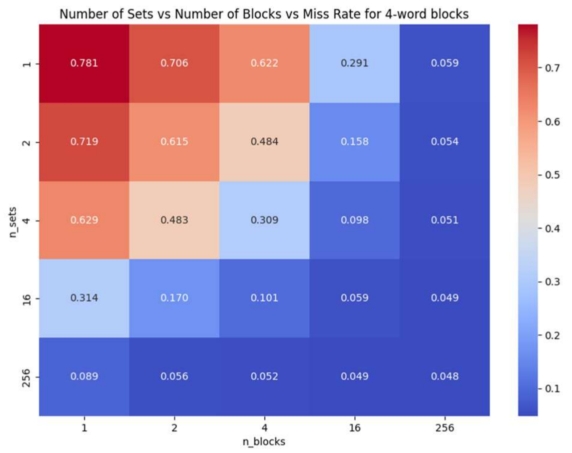


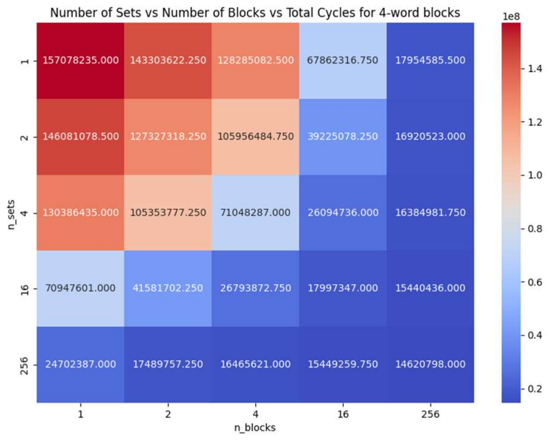


Miss rate seems to decrease with increasing block size in this range, however, as we know, it becomes worse after a certain point. Also, notice that the miss rate seems to become lower with increasing set number. Total cycles increase with block size. Also, write-back takes much less number of cycles.

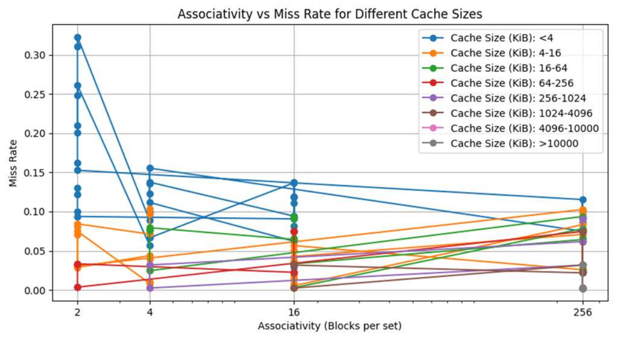
C. Associativity and Write-Allocate Policy

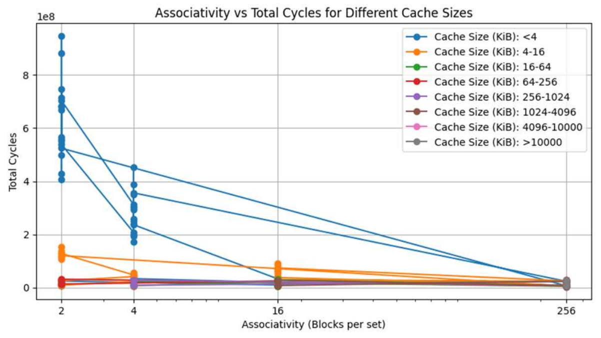
In the following heatmaps, the top edge represents fully-associative caches, and the left edge represents direct-mapped caches. The off diagonal represents similar order of cache size. In both cases, values suggest that set-associative performed better than full-associative and direct-mapped, as we move away from the left and top edges.

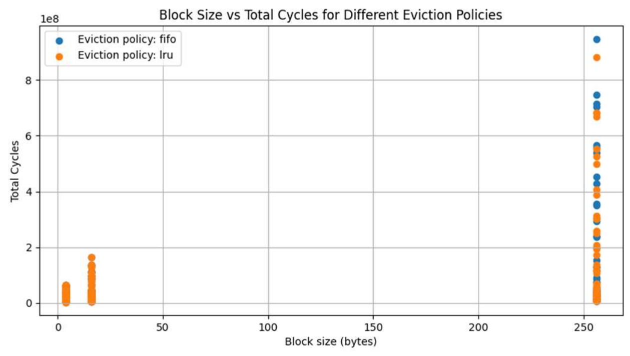


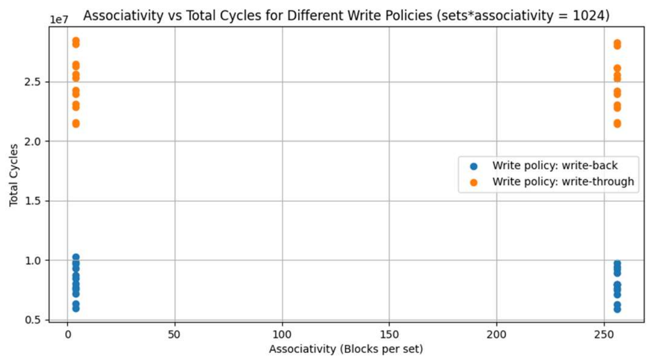


Performance seems to increase with increasing associativity.



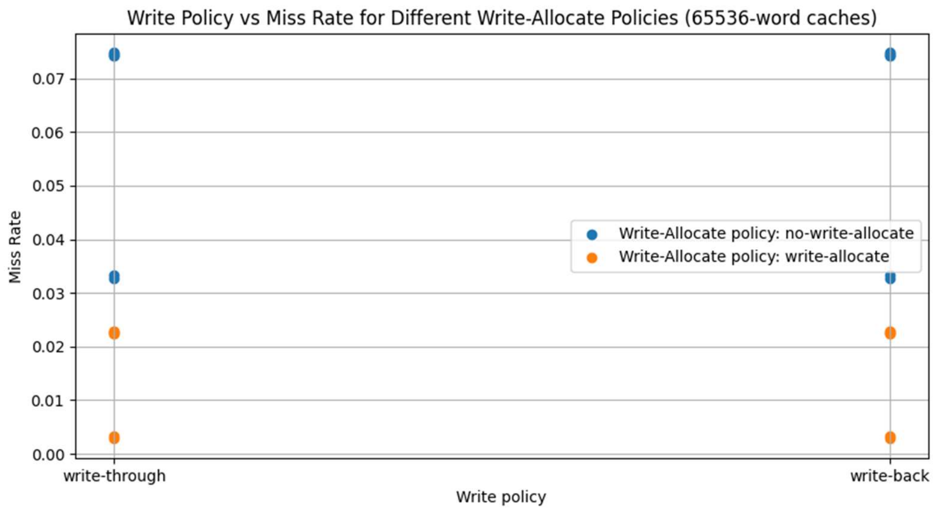






D. Write-Allocation Policy

Miss rate is higher for no-write-allocate cache.



3 Result

A write-back, write-allocate cache with a medium degree of associativity (~16-64) can be chosen for a good trade-off in miss rate and miss penalty. In the write-through, choose no-write-allocate. It is always better to have set-associative caches to move away from the extreme as shown in the heatmaps.