

Introduction

To begin with, in this assignment I filled in the missing functions such that it satisfies the needed requirements of a Direct Map Cache.

The first function is the `dispatcher::dispatch`, it simply parses the `pintool.out` file to search for the appropriate data to form a Direct Map cache. Second the `DirectMap` constructor calculates the offset, the index, and the tag with their appropriate calculations understood from the slides. Then it creates a new `Cachline` object set to the cache map which will be used to access the items. `DirectMap::access` calculates the bits for each of the following: tag, offset, index, cache size, line size; and then calls the `CacheLine` access.

In the `CacheLine::access` I set the mode coming from the method to the character variable added in the header file (to be accessed in the `CacheLine::handle_miss`); and incrementing the `cache_reads`, `cache_writes`, `mem_writes`, and `hits`, in their appropriate if statements, and then at the end calling the `handle_miss` if any has occurred. In the `CacheLine::handle_miss`, I have done if statements that checks on specific cases, making in consideration the option whether it is write back or write through and the Boolean variable I added in the header file to check if the mode is write or read.

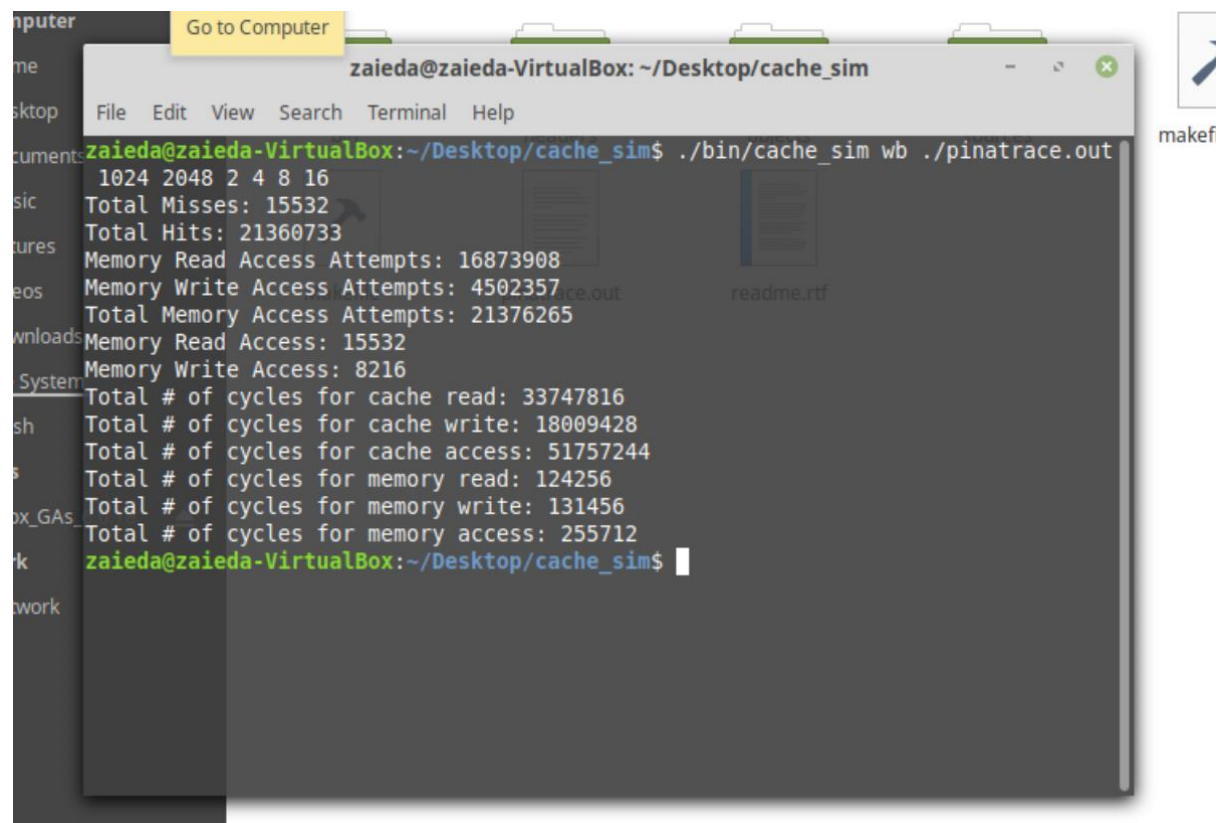
In this report, I will be showcasing different cases that shows the Direct Map implementation over spatial and temporal localities. This will depend on the sample program that I will be modifying to show appropriate results. Spatial locality appears when accessing elements is happening next to each other in memory. Temporal locality appears when accessing one variable reportingly in a program from memory.

Data Collection

1) Original sample program:

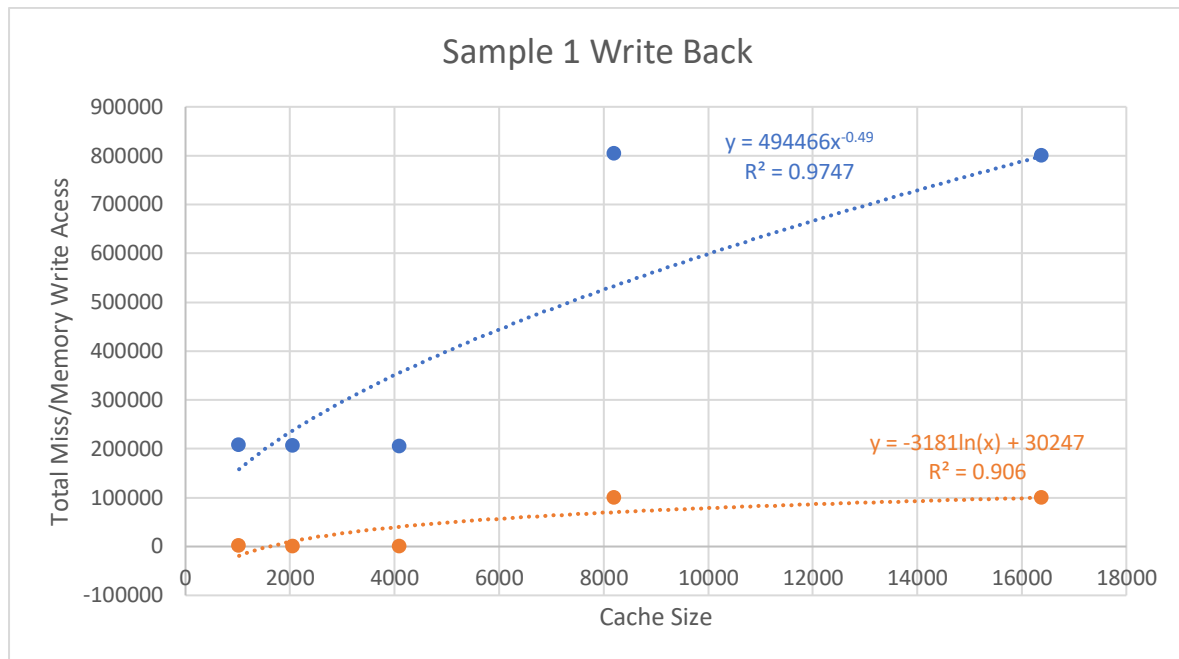
Screenshot of the running program:

In this program the sample shows the access of the elements in the array row by row which satisfies spatial locality. It will allow for less miss compared to the other samples done below.

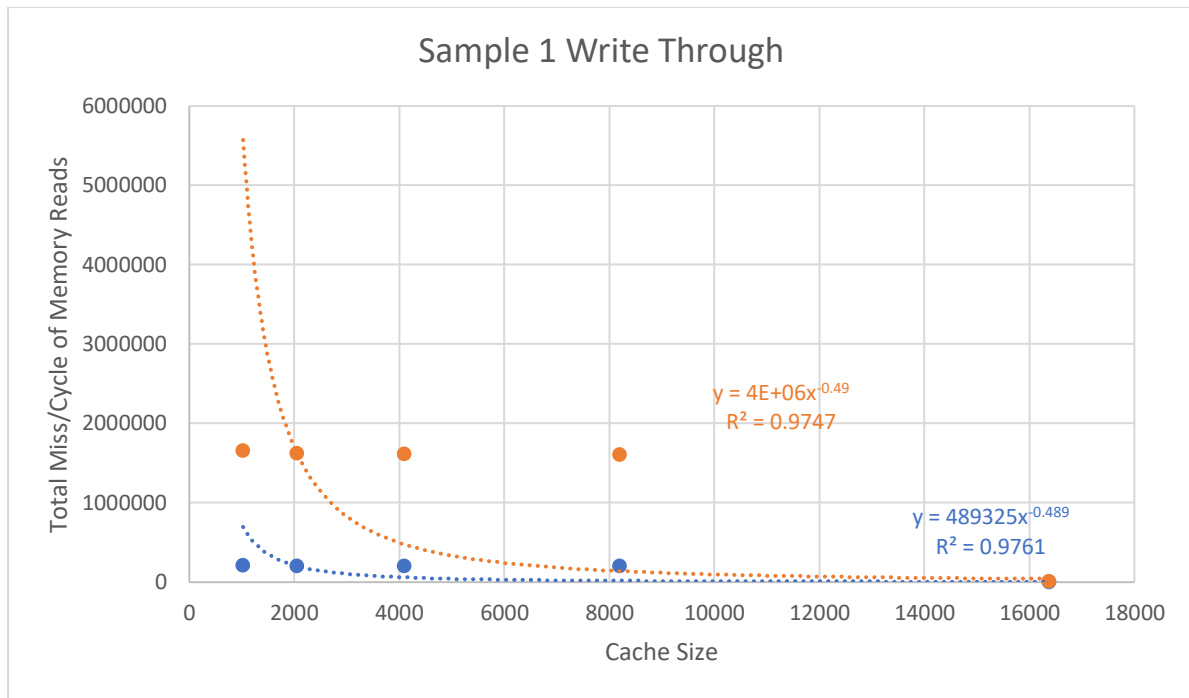


```
zaieda@zaieda-VirtualBox: ~/Desktop/cache_sim
File Edit View Search Terminal Help
zaieda@zaieda-VirtualBox:~/Desktop/cache_sim$ ./bin/cache_sim wb ./pinatrace.out
1024 2048 2 4 8 16
Total Misses: 15532
Total Hits: 21360733
Memory Read Access Attempts: 16873908
Memory Write Access Attempts: 4502357
Total Memory Access Attempts: 21376265
Memory Read Access: 15532
Memory Write Access: 8216
Total # of cycles for cache read: 33747816
Total # of cycles for cache write: 18009428
Total # of cycles for cache access: 51757244
Total # of cycles for memory read: 124256
Total # of cycles for memory write: 131456
Total # of cycles for memory access: 255712
zaieda@zaieda-VirtualBox:~/Desktop/cache_sim$
```

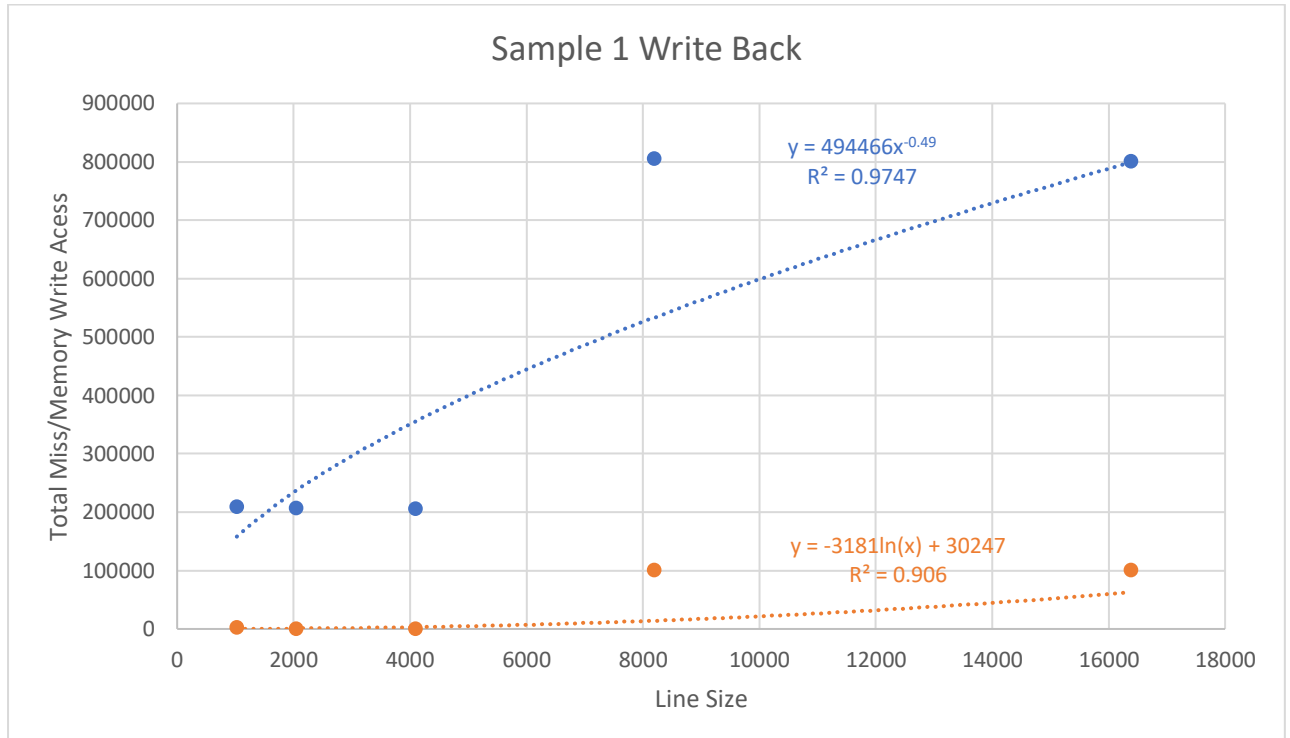
Graphical Interpretations:

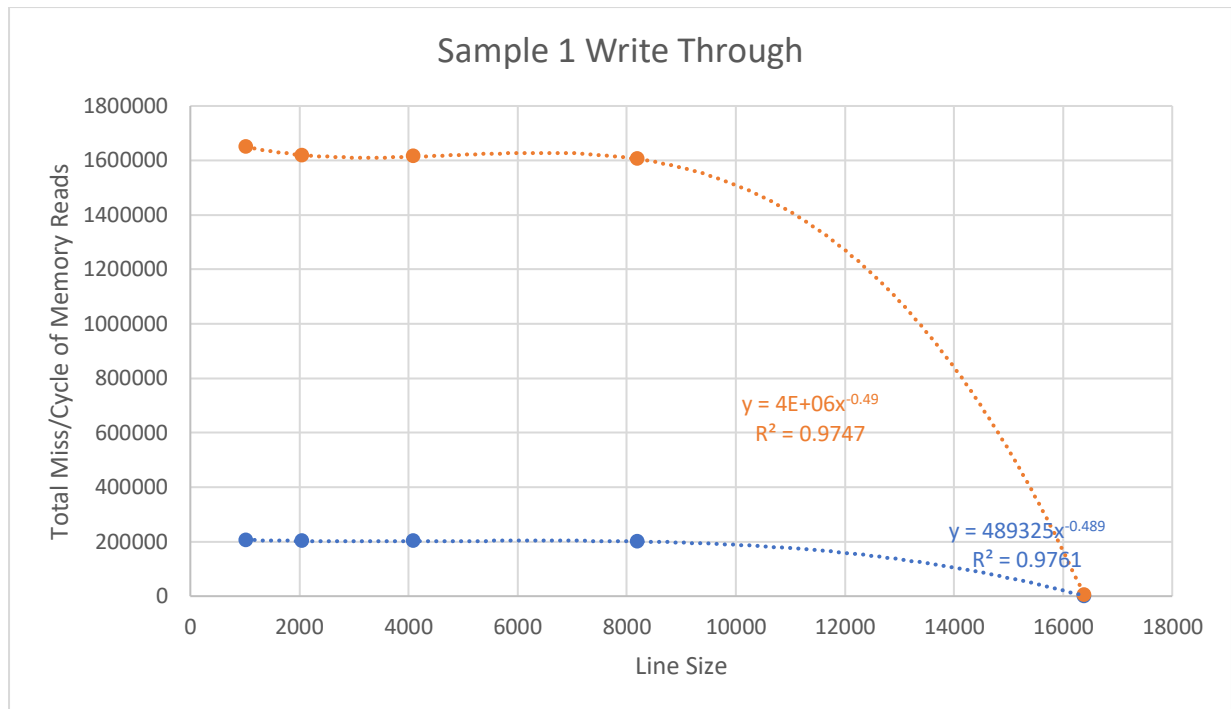


In this graph I have done a comparison between the cache size and the total misses & memory writes which were the main things that were changing in the program's output. As you can see, the graph shows a positive exponential correlation that shows that whenever the cache size increases the numbers increase respectively. This shows the positive correlation in the spatial locality which indicates that the item needed is accessed every time closely in memory to the next element. This also shows that whenever an element, in that case the sum of the array's input, is chosen then there is temporal locality.



In this graph, however, shows that there is a negative correlation and that is because in write through I directly contact the memory without dealing with cache, so typically speaking whenever the cache size is large then the total misses should increase.



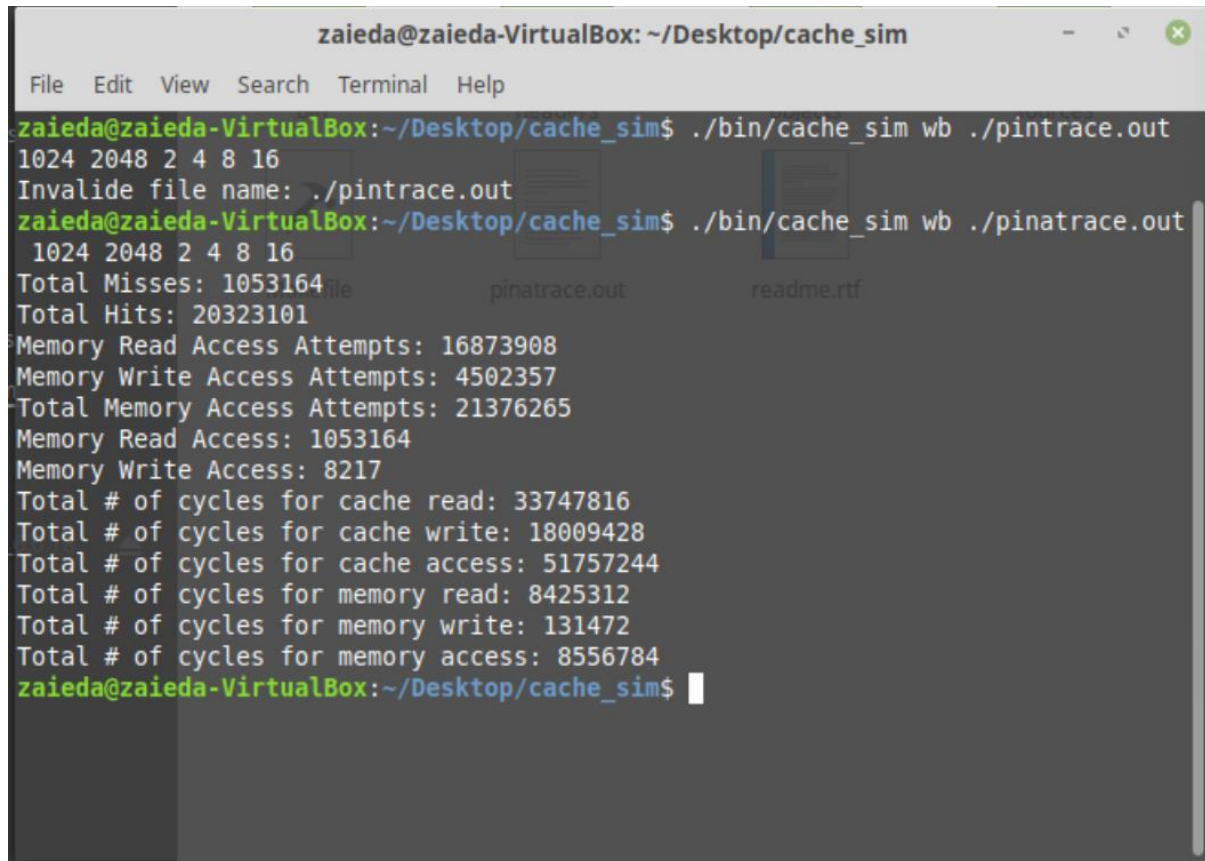


In these two graphs we see that the first one has a negative correlation between the two axes, this is because whenever the line increase in the write back the total misses practically will increase because I am dealing with the cache directly instead of the memory directly. Hence the Memory write should also decrease. Off course when the memory write decrease the total misses should decrease. The line size has a factor in decreasing the missed because it allows for bigger space to store items in the cache.

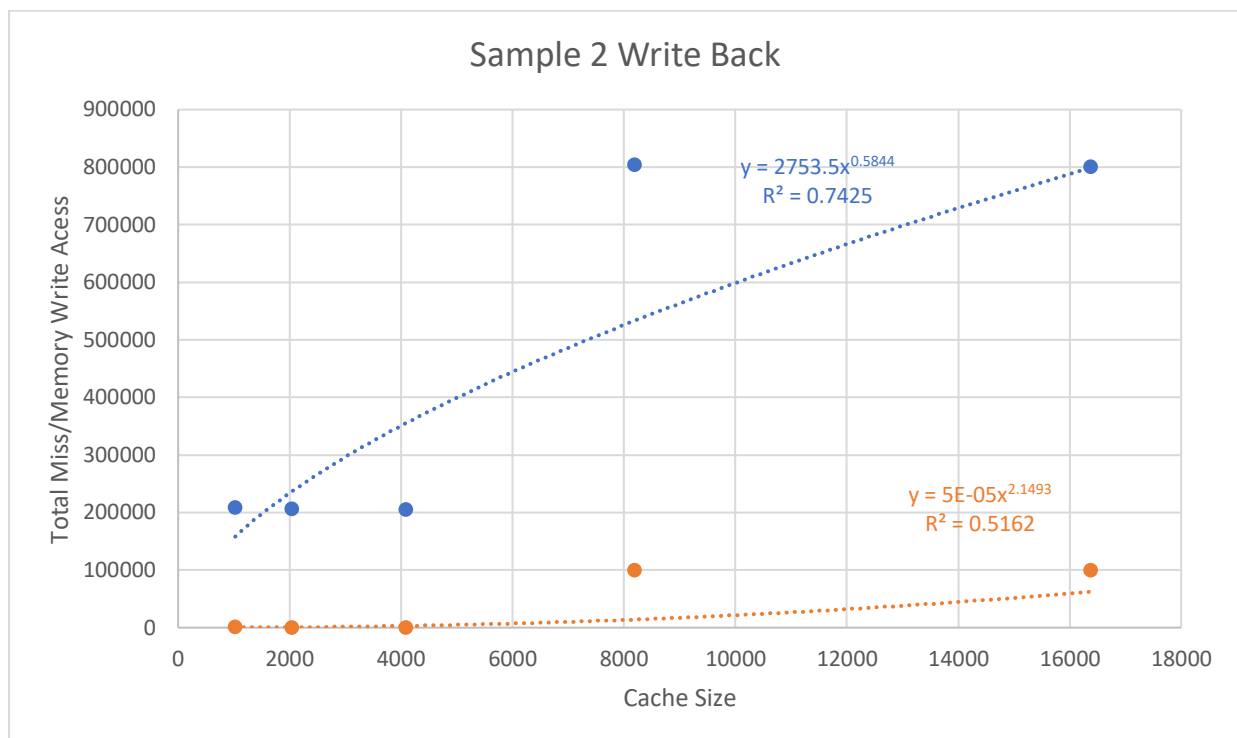
2) Second sample:

Screenshot of the running program:

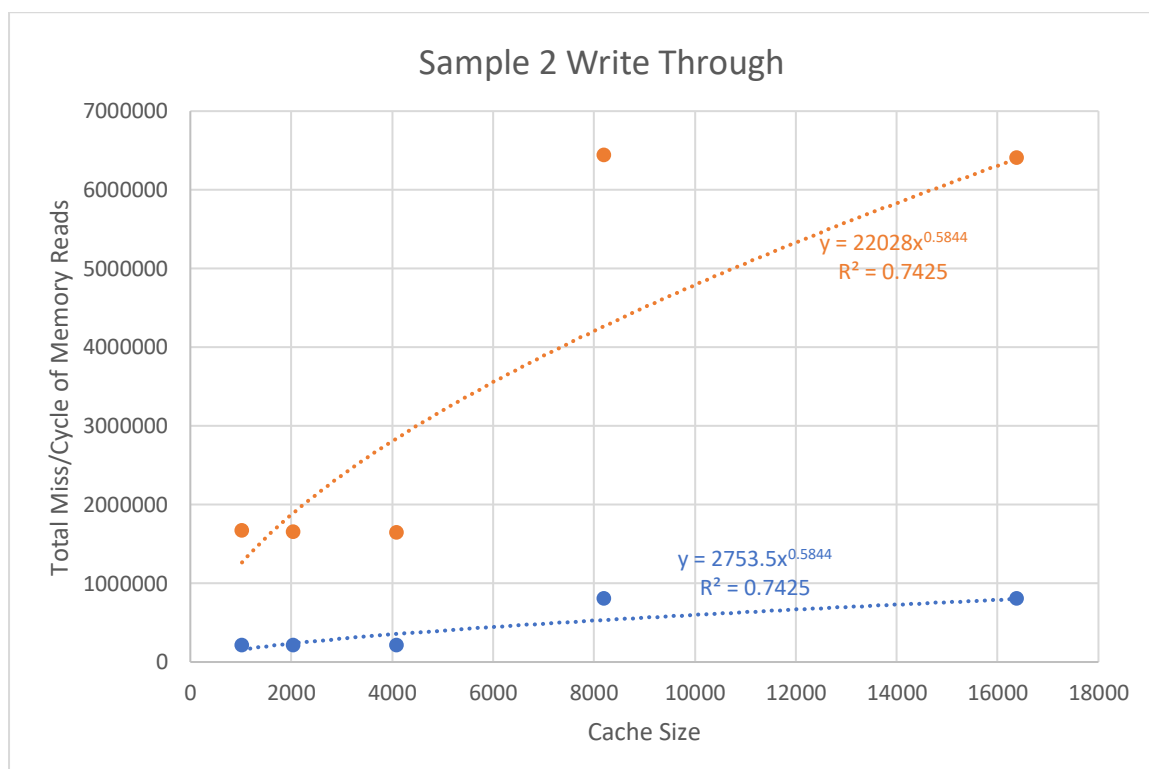
In this program the for loop that calculates the sum was changed. The i variable in the for loop is changed with the j variable such that it access column by column and not row by row.



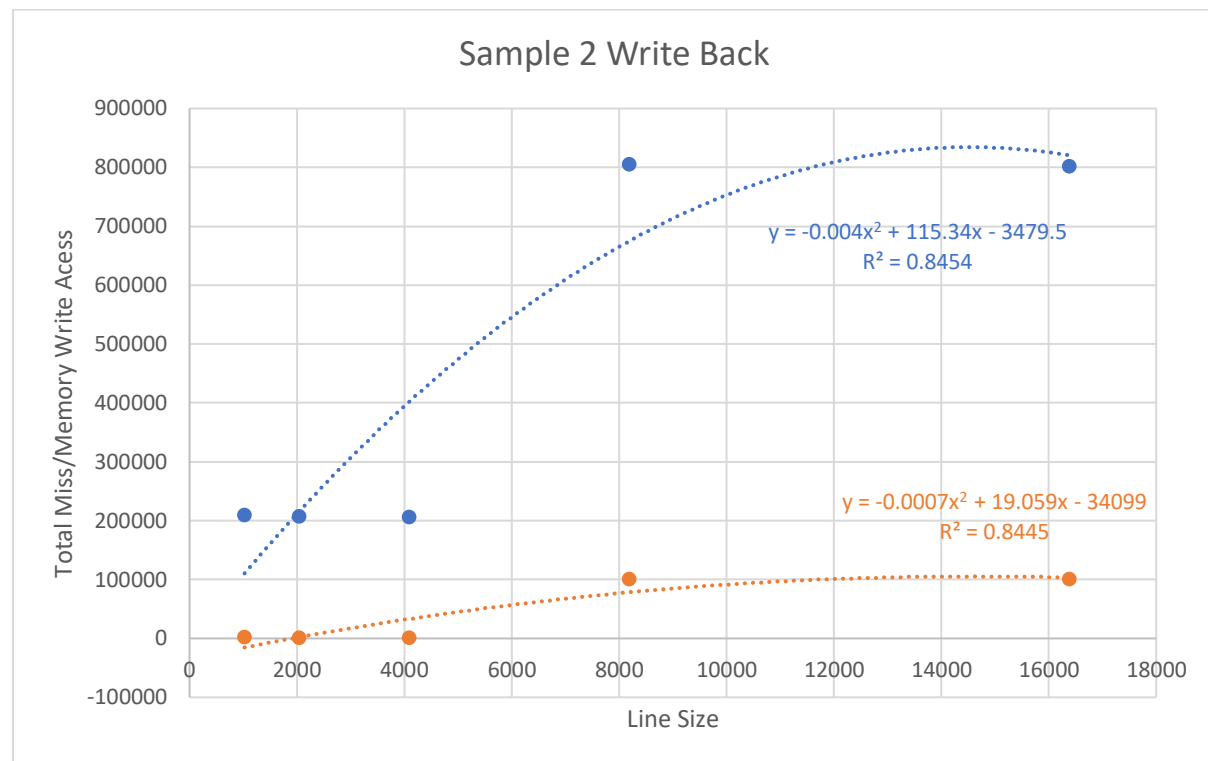
```
zaieda@zaieda-VirtualBox: ~/Desktop/cache_sim
File Edit View Search Terminal Help
zaieda@zaieda-VirtualBox:~/Desktop/cache_sim$ ./bin/cache_sim wb ./pintrace.out
1024 2048 2 4 8 16
Invalide file name: ./pintrace.out
zaieda@zaieda-VirtualBox:~/Desktop/cache_sim$ ./bin/cache_sim wb ./pinatrace.out
1024 2048 2 4 8 16
Total Misses: 1053164
Total Hits: 20323101
Memory Read Access Attempts: 16873908
Memory Write Access Attempts: 4502357
Total Memory Access Attempts: 21376265
Memory Read Access: 1053164
Memory Write Access: 8217
Total # of cycles for cache read: 33747816
Total # of cycles for cache write: 18009428
Total # of cycles for cache access: 51757244
Total # of cycles for memory read: 8425312
Total # of cycles for memory write: 131472
Total # of cycles for memory access: 8556784
zaieda@zaieda-VirtualBox:~/Desktop/cache_sim$
```

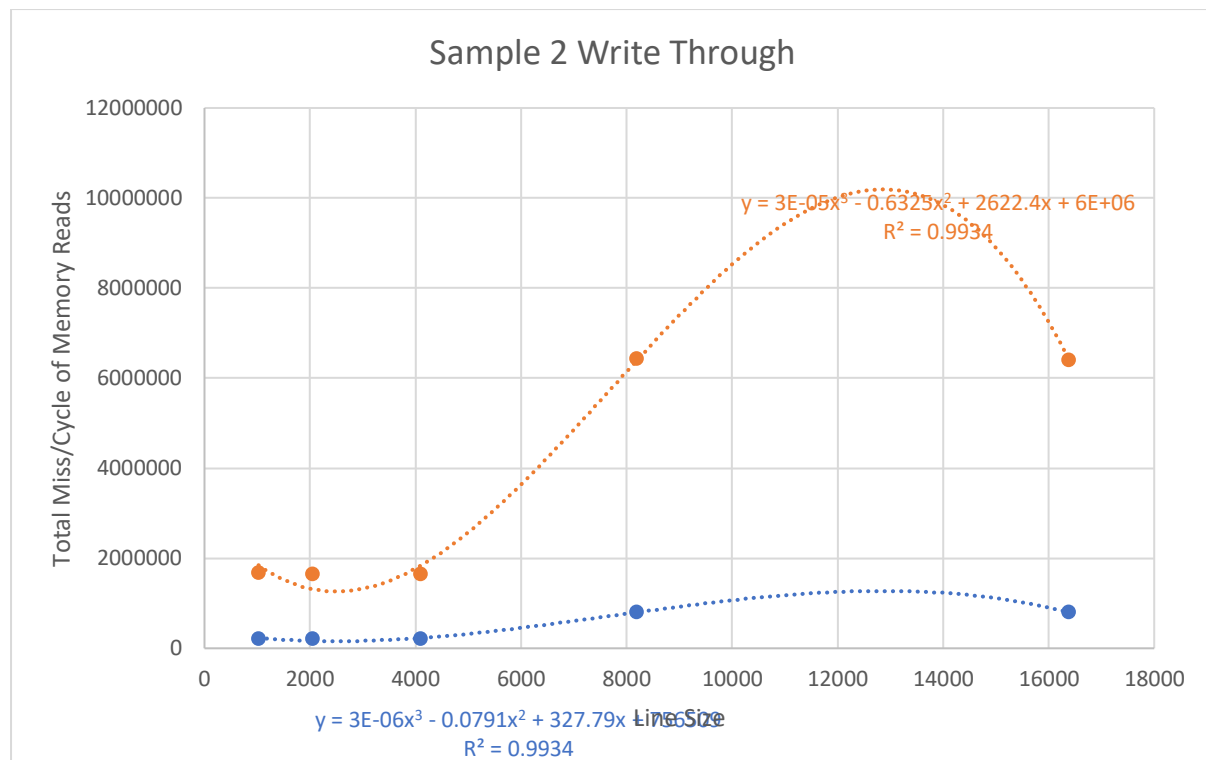
Graphical Interpretation:

In this graph I have showed the total misses increase whenever the cache size increase because it neglects the spatial locality which allows for more misses. This is because the access of memory locations is column by column instead of row by row which contradicts with what c++ does.



In this graph, the write through has the same correlation as the write back because the given total misses is large to convey the difference between the write back and the write through option. **The exponential relationship shown above tells us that whenever the cache size increases the misses will increase in the write through as well as the number of cycles of memory reads.**





In this graph, the line size shows that the increase in the misses happens because the program simply prints the numbers in a for loop. The peak that happens is done because the misses increases then decreases.

3) Sample three:

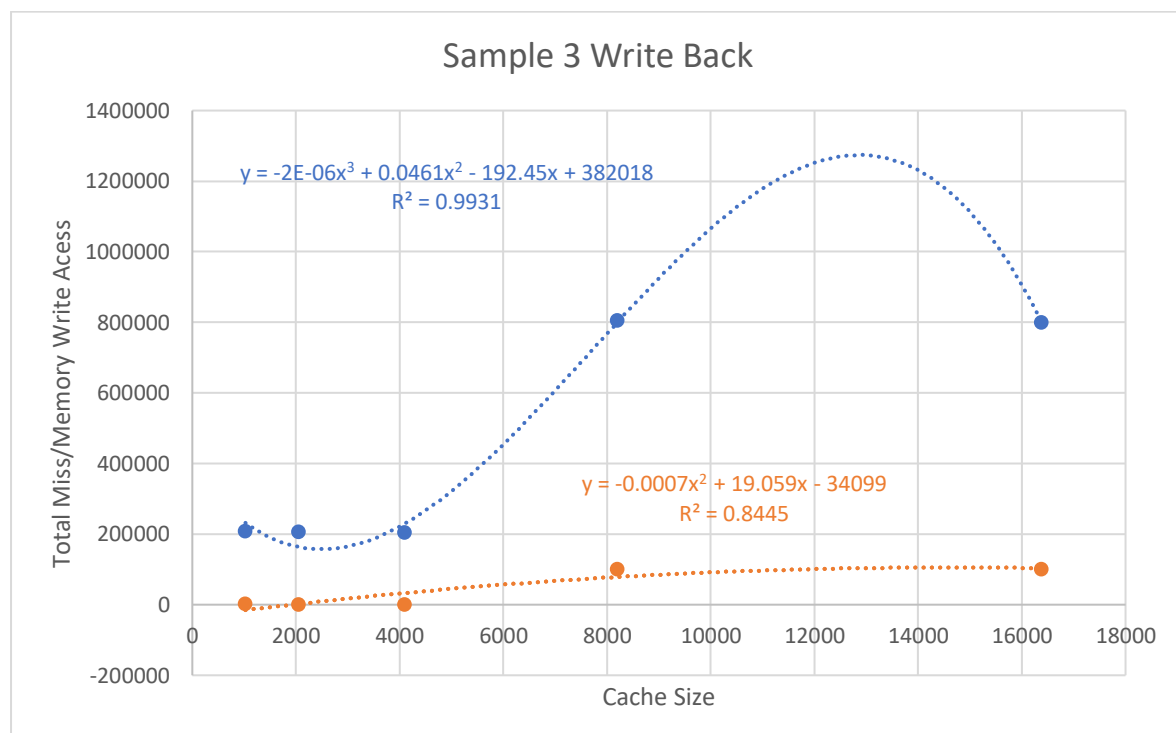
Screenshot of Running Program:

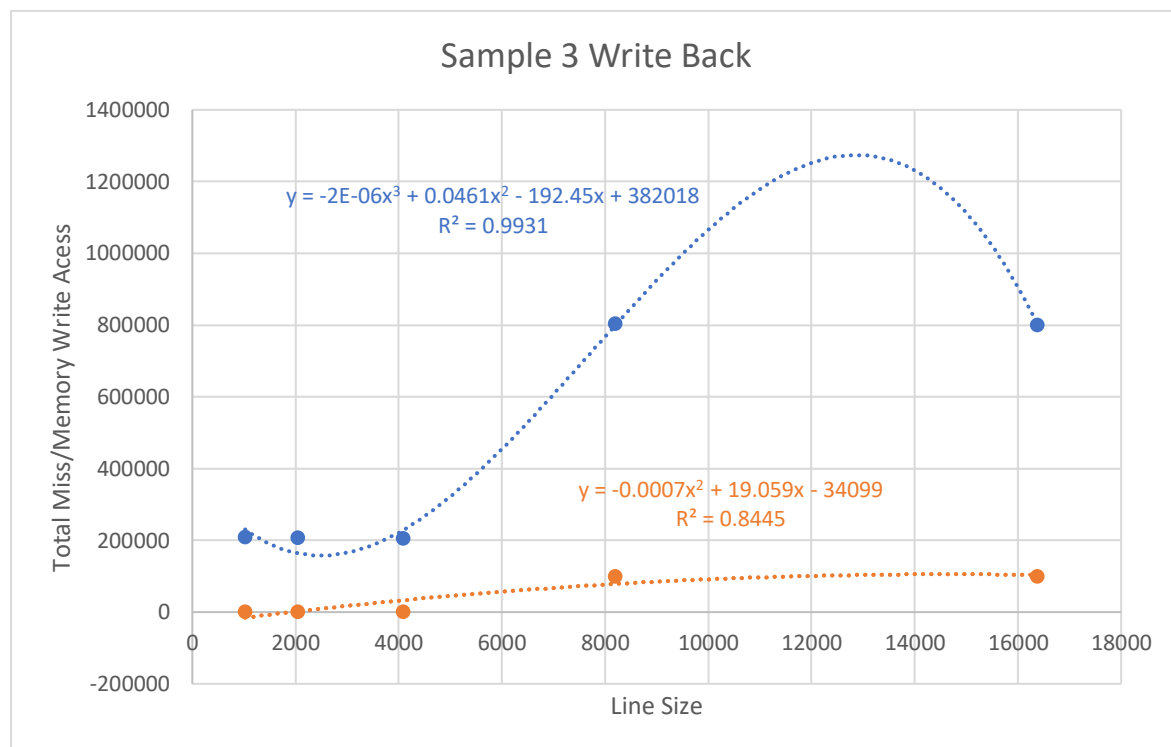
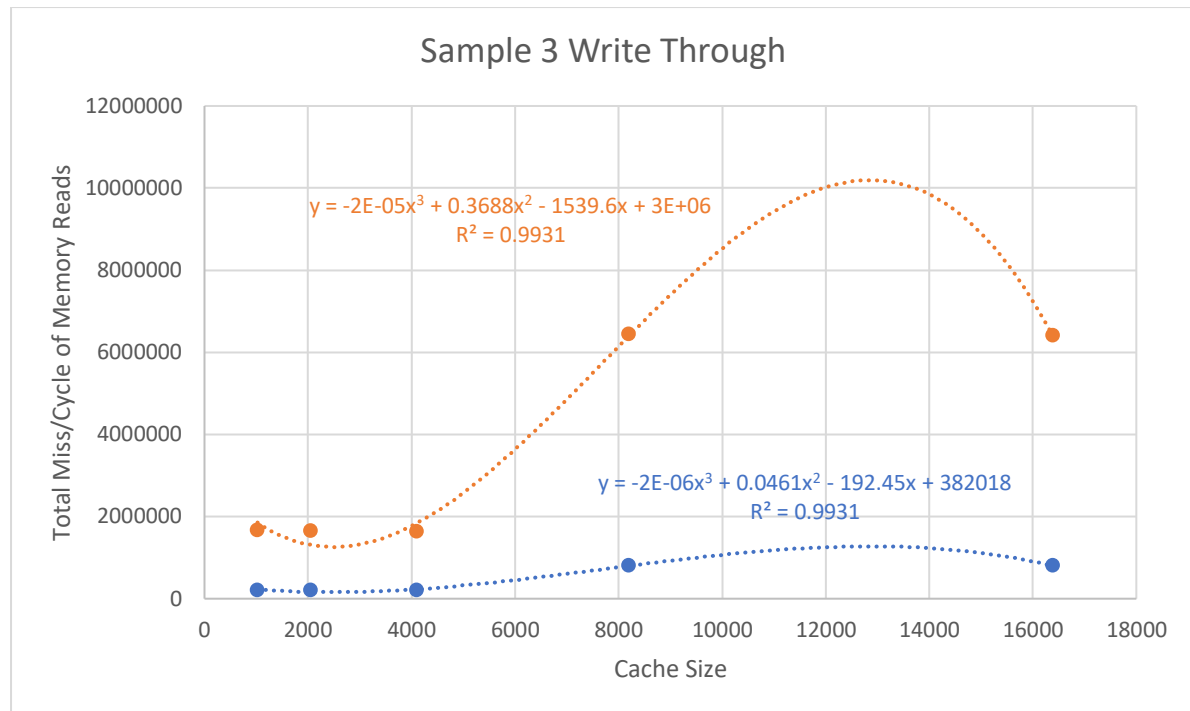
In this program I only print the integers from 1 – 100,000 to showcase the spatial locality.

```

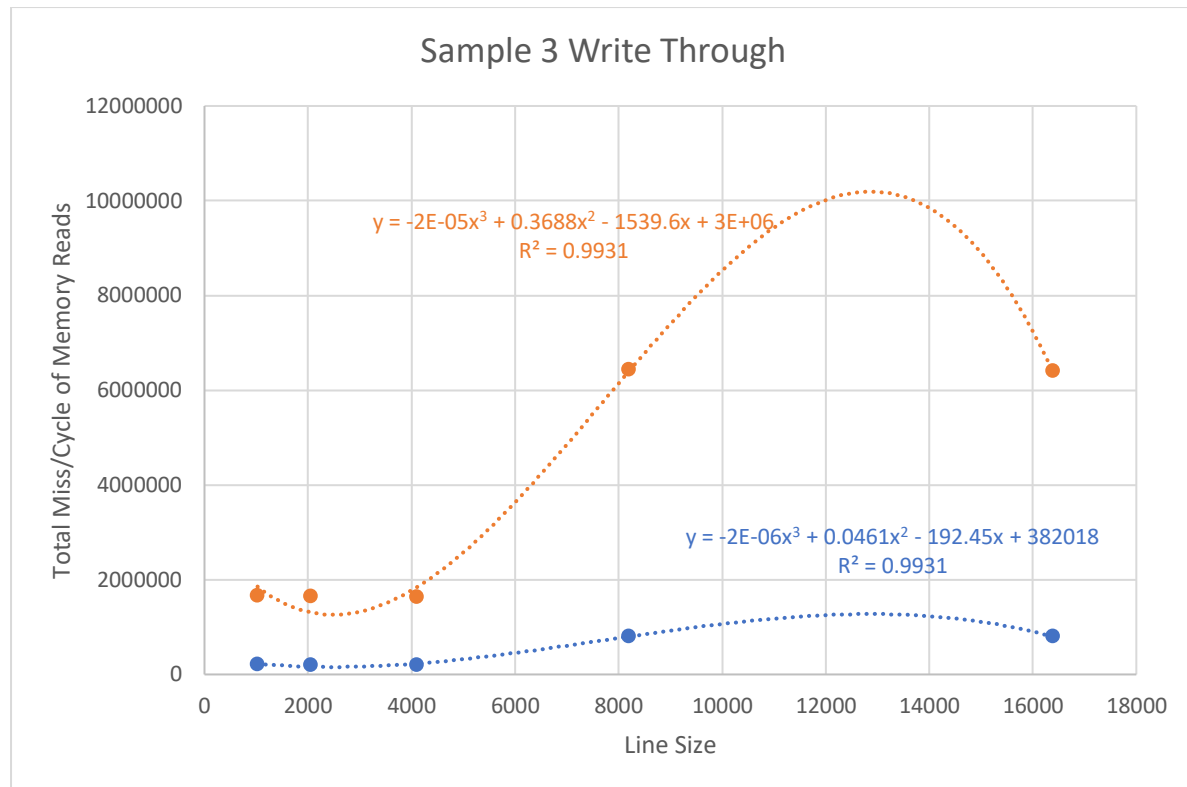
zaieda@zaieda-VirtualBox: ~/Desktop/cache_sim
File Edit View Search Terminal Help
zaieda@zaieda-VirtualBox:~/Desktop/cache_sim$ ./bin/cache_sim wb ./pinatrace.out
1024 2048 2 4 8 16
Total Misses: 206362
Total Hits: 19460929
Memory Read Access Attempts: 12697850
Memory Write Access Attempts: 6969441
Total Memory Access Attempts: 19667291
Memory Read Access: 206362
Memory Write Access: 170
Total # of cycles for cache read: 25395700
Total # of cycles for cache write: 27877764
Total # of cycles for cache access: 53273464
Total # of cycles for memory read: 1650896
Total # of cycles for memory write: 2720
Total # of cycles for memory access: 1653616
zaieda@zaieda-VirtualBox:~/Desktop/cache_sim$

```

Graphical Interpretation:



In these graphs, we have the same positive then negative correlation for the reason that there exists a peak that allows no items to be stored causing a less misses because simply the line size or the cache size has increased.



All of these graphs show the positive correlation then a negative correlation, this is because there exists a peak such the cache allows no more items to be stored when wither the cache size or the line size has increased.

Conclusion:

In conclusion, after performing this assignment, we can see the existence of spatial locality in sample 1 and sample 3 only, because there is always an item to be accessed in the row manner from memory to perform a certain logic and thus it has a high level of spatial locality implementation. Along with, the first, the second, and the third samples show temporal locality as well because there is a continuous access of the same variable every single time to perform their logic, either by summing the variable 's' or printing the content stored in the item 'i'. The difference between write through and write back, is that write through writes through the memory directly without writing in the cache, and hence the numbers presented in graphs shall be less in write through than in write back. The difference between the line size and the cache size increments that I perform is that the cache size allows for greater possibility of temporal locality. A bigger line size allows for bigger possibility of spatial locality depending on the cache size of course, everything with their dependencies, and that is why I kept these variables constant. **Note that the r^2 value shows how close the tradeline is to the points. The tradeline equation is just there for graphical interpretation.**