CSE490 Project 2 Report

Kyle Lemma, Caroline Hart, U Shin, Cristian Pompey

May 5, 2023

Table of Contents

Assignment Details.....3

Block Size.....4

L1 Instruction Cache Size.....7

L2 Cache Size.....12

L1 Instruction Cache Associativity.....15

L1 Data Cache Associativity.....19

Assignment Details

Assigned Benchmarks:

- 401.bzip2
- 456.hmmer
- 458.sjeng

Assigned Parameters:

- L2 Cache Size
- L1 Instruction Memory Cache Size
- L1 Data Memory Associativity
- L1 Instruction Memory Associativity
- Block Size

Process for Completing This Assignment:

1. We had a group meeting to discuss the project. During this meeting we assigned each partner a specific parameter from our assigned parameters. Since there were 5 parameters and 4 partners, Kyle took on two of the assigned parameters. We created a shared Excel spreadsheet to keep track of everything. During this meeting, we decided on which constants we should use for the static parameters.
2. Each group member decided what values to use for their assigned parameter for each of the 5 test cases. The test cases would use the same values for each of the 5 test cases regardless of the benchmark in an effort to be consistent.
3. Each group member ran their test cases, then reported the results in the shared Excel spreadsheet.
4. We had a second group meeting to discuss how to create the charts in the shared Excel spreadsheet. Then, each group member independently created their charts.
5. We created a shared Word document to report our findings. Each member added their data tables, charts, and analyses to the shared document.
6. We had a third group meeting to format the Word document and ensure that each of the charts were consistent with each other.
7. To get ready for submission, each group member shared their stat files in our group's chat. Each of us reviewed the document for errors.

Division of Labor:

Kyle	Block Size & analysis, L2 Cache Size & analysis, Report, Reviewed analyses
Caroline	L1 Instruction Memory Cache Size & analysis, Report, Collected Stat Files
U	L1 Instruction Memory Associativity & analysis
Cris	L1 Data Memory Associativity & analysis

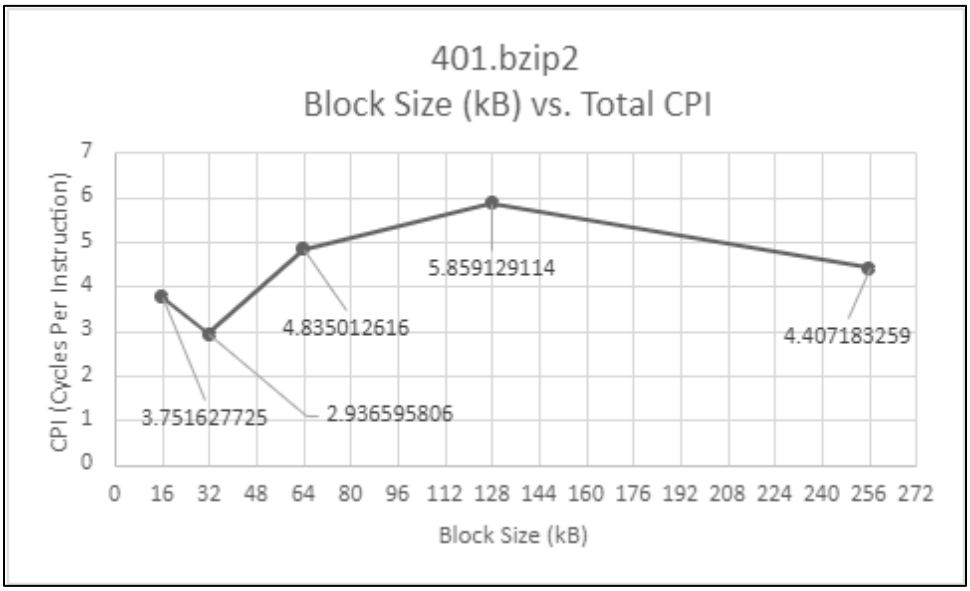
Block Size

Benchmark Results:

401.bzip2

Block Size Results	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
Block Size	16	32	64	128	256
Overall Miss Rate I Cache	0.076852	0.060945	0.272766	0.303992	0.271649
Overall Miss Rate D Cache	0.009128	0.005118	0.003105	0.003002	0.002006
Overall Miss Rate L2	0.717978	0.664984	0.272815	0.327807	0.232974
Overall Misses D Cache	538843	427312	1912480	2131419	1904649
Overall Misses I Cache	117887	66098	40100	38772	25907
Overall Misses L2	471518	328110	532693	711403	449770
Total Instructions	10000001	10000001	10000001	10000001	10000001
Total CPI	3.7516277	2.9365958	4.8350126	5.8591291	4.4071832

From running the benchmark with only changing block size we were able to come up with the data above. This data was then used to calculate the CPI for each block size which is documented in the graph below:



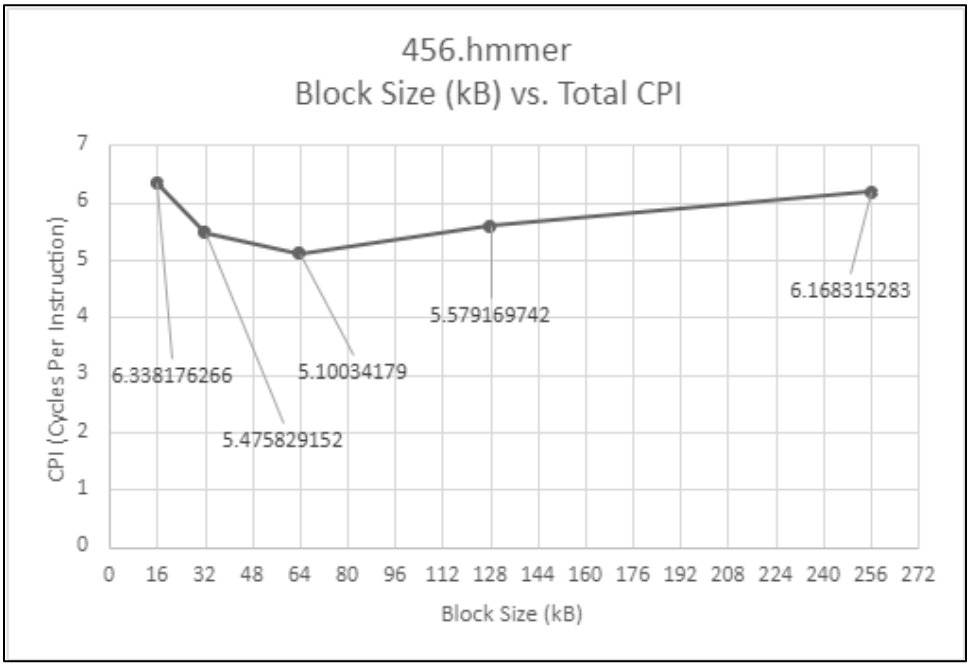
From the 401.bzip2 benchmark we can see that the overall CPI started to decrease but then gradually increased as the block size increased, until finally at a block size of 256 decreased back down again. This is because while the block size grew the total number of L1 D Cache misses increased drastically, along with L2 Cache misses also drastically increased. The only overall misses that decreased were the L1 I Cache, which was always decreasing. This is because that while increasing the block size without increasing the size of the total cache, would mean that there would be a decrease in the total amount of

blocks in each level of the cache. This will then relate to the overall number of misses increasing since there was that decreased number of total blocks.

456.hmmer

Block Size Results	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
Block Size	16	32	64	128	256
Overall Miss Rate I Cache	0.109233	0.133374	0.155224	0.203618	0.240379
Overall Miss Rate D Cache	0.081279	0.050478	0.028658	0.019948	0.013474
Overall Miss Rate L2	0.511698	0.527596	0.568678	0.575491	0.610688
Overall Misses D Cache	561417	681322	792831	1039811	1227535
Overall Misses I Cache	1128686	700969	397956	277006	187108
Overall Misses L2	864823	729291	677174	757816	863906
Total Instructions	10000001	10000001	10000001	10000001	10000001
Total CPI	6.338176266	5.475829152	5.10034179	5.579169742	6.168315283

From running the benchmark with only changing block size we were able to come up with the data above. This data was then used to calculate the CPI for each block size which is documented in the graph below:



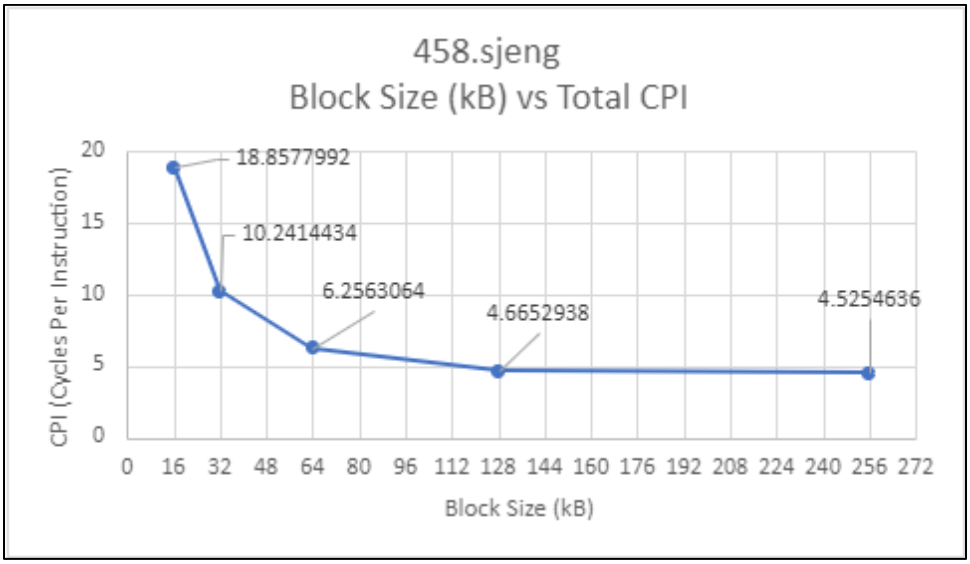
From the 456.hmmer benchmark we can see that the overall CPI started to decrease but then gradually increased as the block size increased, this is very similar to the pattern found in the prior benchmark. This again is because while the block size grew the total number of L1 D Cache misses increased drastically, along with L2 Cache misses also drastically increased. The only overall misses that decreased were the L1 I Cache, which decreased until the block size reached 64 which then started to increase again. This is because that while increasing the block size without increasing the size of the total cache,

would mean that there would be a decrease in the total amount of blocks in each level of the cache. This will then relate to the overall number of misses increasing since there was that decreased number of total blocks.

458.sjeng

Block Size Results	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
Block Size	16	32	64	128	256
Overall Miss Rate I Cache	0.414432	0.218447	0.130736	0.097583	0.098933
Overall Miss Rate D Cache	0.000115	0.00007	0.000047	0.000035	0.000027
Overall Miss Rate L2	0.991343	0.971022	0.916804	0.8486	0.79907
Overall Misses D Cache	3212170	1693133	1013300	756339	766809
Overall Misses I Cache	1562	956	644	484	372
Overall Misses L2	3185912	1644998	929588	642240	613031
Total Instructions	10000000	10000000	10000000	10000000	10000000
Total CPI	18.857792	10.241443	6.2563064	4.6652938	4.5254636

From running the benchmark with only changing block size we were able to come up with the data above. This data was then used to calculate the CPI for each block size which is documented in the graph below:



Finally, for the 458.sjeng benchmark we can see that for the entire time, as block size increases the total CPI always was decreasing exponentially. Compared to the previous two benchmarks, the total number of misses for the L1 I Cache, L1 D Cache, and L2 Cache were always decreasing. This could be because the sizes of the cache were different and large enough to accommodate the larger block sizes. This would result in the behavior that is shown above in the graph, which would result in the improvements of CPI throughout the entire testing. As the block size increased, the total number of misses decreased which resulted in a decrease in CPI.

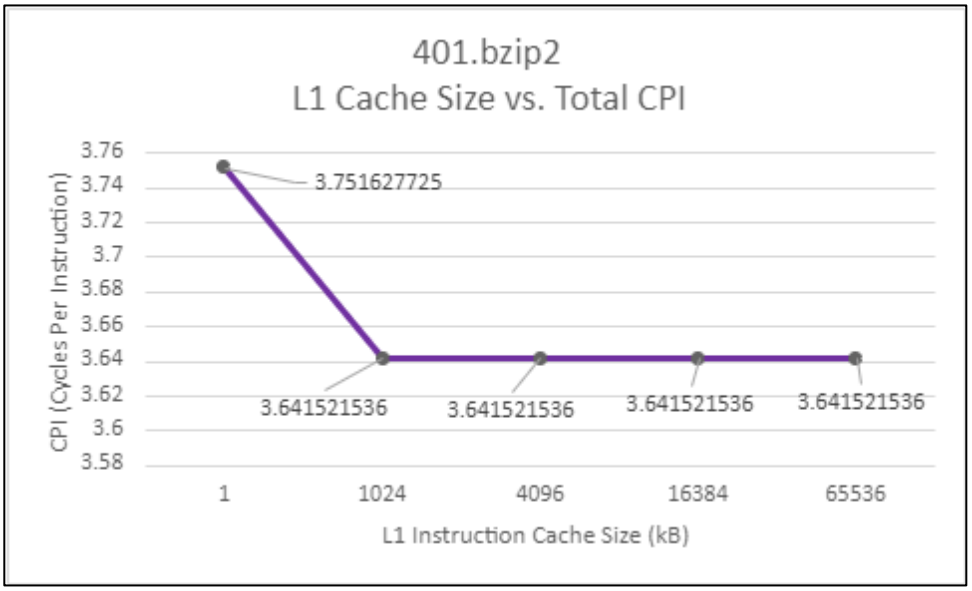
L1 Instruction Cache Size

Benchmark Results:

401.bzip2

L1 Instrn Cache Size Results	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
L1 Instrn Cache Size (kB)	1	1024	4096	16384	65536
Overall Miss Rate D Cache	0.076852	0.076852	0.076852	0.076852	0.076852
Overall Miss Rate I Cache	0.009128	0.000111	0.000111	0.000111	0.000111
Overall Miss Rate L2	0.717978	0.857838	0.857838	0.857838	0.857838
Overall Misses D Cache	538843	538843	538843	538843	538843
Overall Misses I Cache	117887	1435	1435	1435	1435
Overall Misses L2	471518	463471	463471	463471	463471
Total Instructions	10000001	10000001	10000001	10000001	10000001
Total CPI	3.75162772	3.64152154	3.64152154	3.64152154	3.64152154

From running the benchmark with only changing L1 Instruction Cache size we were able to come up with the data above. This data was then used to calculate the CPI for each block size which is documented in the graph below:



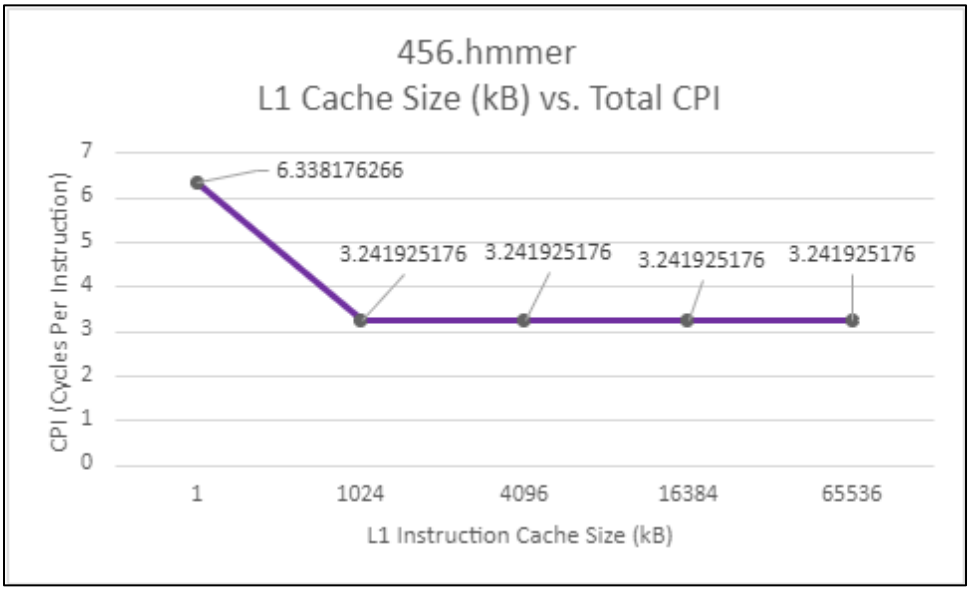
From the 401.bzip2 benchmark we can see that the overall CPI decreased after the L1 Instruction Cache size was changed from 1 kB to 1024 kB. After the L1 Instruction cache size increased to 1024 kB, the CPI stayed constant at roughly 3.64 cycles per instruction no matter how much the L1 Instruction Cache Size increased after that. Additionally, when the L1 instruction cache size was greater than or equal to 1024 kB, both the number of L1 Instruction Cache Misses and the L1 Instruction Cache Miss Rate decreased because with a larger L1 Instruction Cache Size, there is more that can be stored in the L1 Instruction Cache, leading to more cache hits.

During the entire Parameter Test Set, the total number of L1 Data Cache Misses as well as the L1 Data Cache Miss Rate were constant because we only modified the L1 Instruction Cache Size. Additionally, the number of L2 Cache Misses decreased after the L1 Instruction Cache Size was greater than or equal to 1024 kB. This is likely because as the L1 Instruction Cache Size increased, a greater number of instructions were able to be stored in the cache, leading to a decreased need to check the L2 cache because the decreased L1 Instruction Cache miss rate implies a higher hit rate for the L1 Instruction Cache.

456.hmmer

L1 Instrn Cache Size Results	**Test Case 1**	**Test Case 2**	**Test Case 3**	**Test Case 4**	**Test Case 5**
L1 Instrn Cache Size (kB)	1	1024	4096	16384	65536
Overall Miss Rate D Cache	0.109233	0.109233	0.109233	0.109233	0.109233
Overall Miss Rate I Cache	0.081279	0.000313	0.000313	0.000313	0.000313
Overall Miss Rate L2	0.511698	0.672537	0.672537	0.672537	0.672537
Overall Misses D Cache	561417	561417	561417	561417	561417
Overall Misses I Cache	1128686	4342	4342	4342	4342
Overall Misses L2	864823	380494	380494	380494	380494
Total Instructions	10000001	10000001	10000001	10000001	10000001
Total CPI	6.338176266	3.241925176	3.241925176	3.241925176	3.241925176

From running the benchmark with only changing L1 Instruction Cache size we were able to come up with the data above. This data was then used to calculate the CPI for each block size which is documented in the graph below:



From the 456.hmmer benchmark we can see that the overall CPI decreased after the L1 Instruction Cache size was changed from 1 kB to 1024 kB. After the L1 Instruction cache size increased to 1024 kB, the CPI stayed constant at roughly 3.24 cycles per instruction no matter how much the L1 Instruction

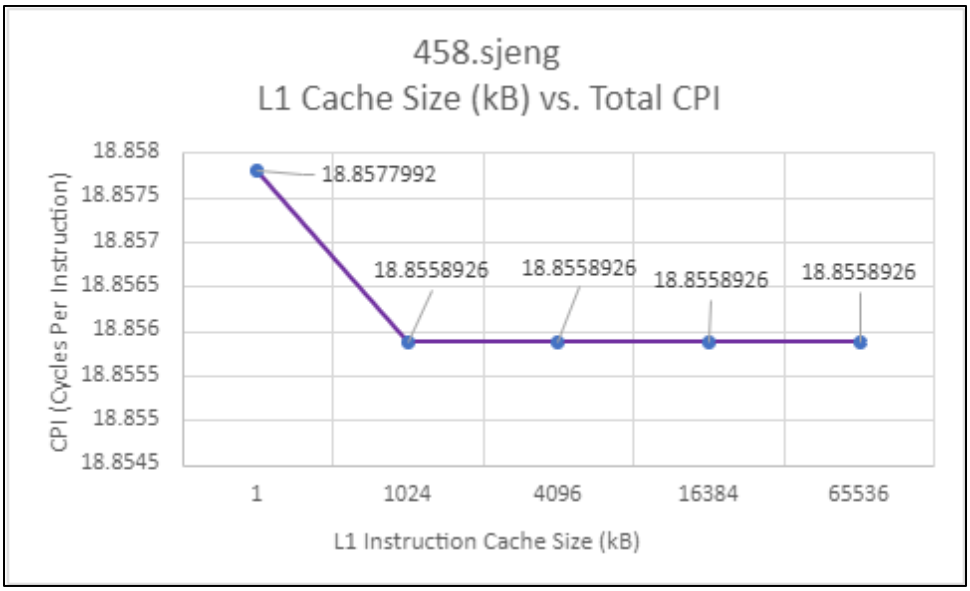
Cache Size increased after that. Additionally, when the L1 instruction cache size was greater than or equal to 1024 kB, both the number of L1 Instruction Cache Misses and the L1 Instruction Cache Miss Rate decreased because with a larger L1 Instruction Cache Size, there is more that can be stored in the L1 Instruction Cache, leading to more cache hits.

During the entire Parameter Test Set, the total number of L1 Data Cache Misses as well as the L1 Data Cache Miss Rate were constant because we only modified the L1 Instruction Cache Size. Additionally, the number of L2 Cache Misses decreased after the L1 Instruction Cache Size was greater than or equal to 1024 kB. This is likely because as the L1 Instruction Cache Size increased, a greater number of instructions were able to be stored in the cache, leading to a decreased need to check the L2 cache because the decreased L1 Instruction Cache miss rate implies a higher hit rate for the L1 Instruction Cache.

458.sjeng

L1 Instrn Cache Size Results	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
L1 Instrn Cache Size (kB)	1	1024	4096	16384	65536
Overall Miss Rate D Cache	0.414432	0.414432	0.414432	0.414432	0.414432
Overall Miss Rate I Cache	0.000115	0.000081	0.000081	0.000081	0.000081
Overall Miss Rate L2	0.991343	0.991384	0.991384	0.991384	0.991384
Overall Misses D Cache	3212170	3212170	3212170	3212170	3212170
Overall Misses I Cache	1562	1101	1101	1101	1101
Overall Misses L2	3185912	3185586	3185586	3185586	3185586
Total Instructions	10000000	10000000	10000000	10000000	10000000
Total CPI	18.8577992	18.8558926	18.8558926	18.8558926	18.8558926

From running the benchmark with only changing L1 Instruction Cache size we were able to come up with the data above. This data was then used to calculate the CPI for each block size which is documented in the graph below:



From the 456.hmmer benchmark we can see that the overall CPI decreased after the L1 Instruction Cache size was changed from 1 kB to 1024 kB. Increasing the L1 Instruction Cache only marginally affected the Total CPI. After the L1 Instruction cache size increased to 1024 kB, the CPI stayed constant at roughly 18.86 cycles per instruction no matter how much the L1 Instruction Cache Size increased after that. Additionally, when the L1 instruction cache size was greater than or equal to 1024 kB, both the number of L1 Instruction Cache Misses and the L1 Instruction Cache Miss Rate decreased because with a larger L1 Instruction Cache Size, there is more that can be stored in the L1 Instruction Cache, leading to more cache hits.

During the entire Parameter Test Set, the total number of L1 Data Cache Misses as well as the L1 Data Cache Miss Rate were constant because we only modified the L1 Instruction Cache Size. Additionally,

the number of L2 Cache Misses decreased after the L1 Instruction Cache Size was greater than or equal to 1024 kB. This is likely because as the L1 Instruction Cache Size increased, a greater number of instructions were able to be stored in the cache, leading to a decreased need to check the L2 cache because the decreased L1 Instruction Cache miss rate implies a higher hit rate for the L1 Instruction Cache.

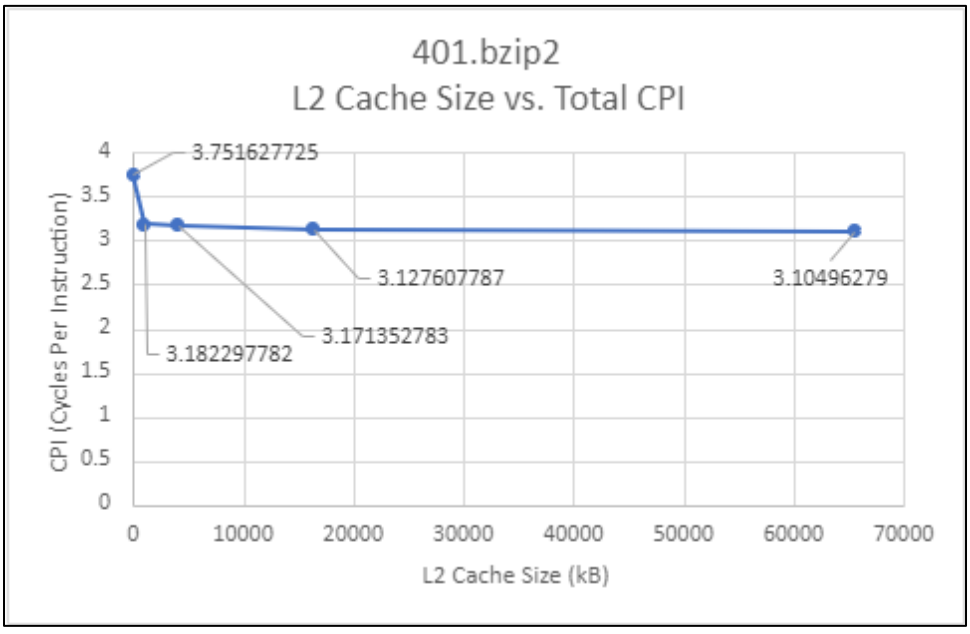
L2 Cache Size

Benchmark Results:

401.bzip2

L2 Cache Size Results	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
L2 Cache Size	1	1024	4096	16384	65536
Overall Miss Rate I Cache	0.076852	0.076852	0.076852	0.076852	0.076852
Overall Miss Rate D Cache	0.009128	0.009128	0.009128	0.009128	0.009128
Overall Miss Rate L2	0.717978	0.544997	0.541262	0.528342	0.521044
Overall Misses D Cache	538843	536643	538843	536643	538843
Overall Misses I Cache	117887	117887	117887	117887	117887
Overall Misses L2	471518	357916	355463	346978	342185
Total Instructions	10000001	10000001	10000001	10000001	10000001
Total CPI	3.7516277	3.1822977	3.1713527	3.1276077	3.1049627

From running the benchmark with only changing L2 Cache size we were able to come up with the data above. This data was then used to calculate the CPI for each block size which is documented in the graph below:

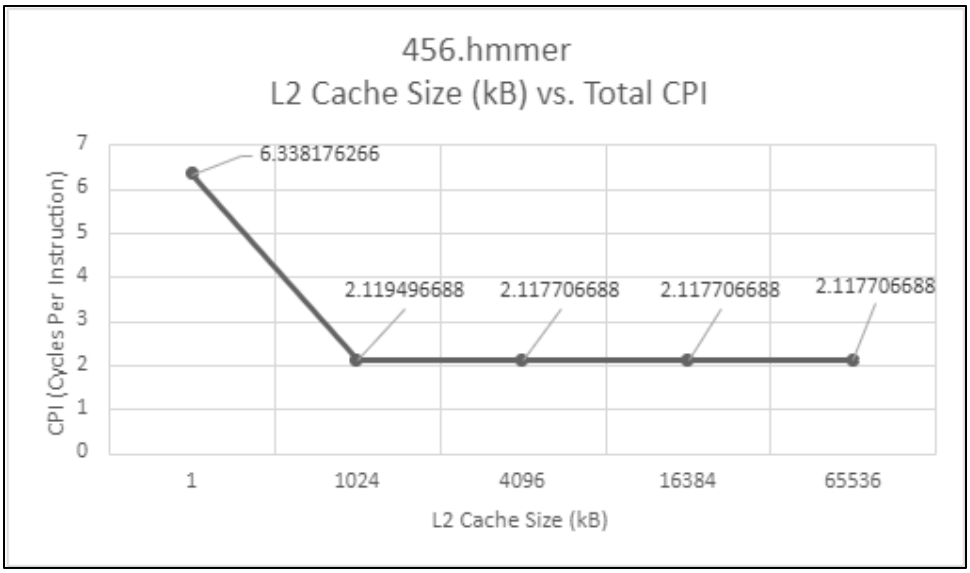


From the 401.bzip2 benchmark we can see that the overall CPI started to decrease but roughly stayed the same after the L2 cache size got larger than 1024kB. This is because as the L2 Cache size grew, once reaching an L2 Cache Size of 1024 kB the number of L2 Cache misses stayed roughly the same. Meanwhile during the entire Parameter Test Set, the total number of L1 D and L1 I Cache misses stayed the same since changing the size of the L2 Cache has no effect on the L1 cache whatsoever. That is why after increasing the L2 Cache so much that there was no other effect of benefit in increasing the size.

456.hmmer

L2 Cache Size Results	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
L2 Cache Size	1	1024	4096	16384	65536
Overall Miss Rate I Cache	0.109233	0.109233	0.109233	0.109233	0.109233
Overall Miss Rate D Cache	0.081279	0.081279	0.081279	0.081279	0.081279
Overall Miss Rate L2	0.511698	0.012477	0.012265	0.012265	0.012265
Overall Misses D Cache	561417	561417	561417	561417	561417
Overall Misses I Cache	1128686	1128686	1128686	1128686	1128686
Overall Misses L2	864823	21087	20729	20729	20729
Total Instructions	10000001	10000001	10000001	10000001	10000001
Total CPI	6.338176266	2.119496688	2.11770668	2.117706688	2.117706688

From running the benchmark with only changing L2 Cache size we were able to come up with the data above. This data was then used to calculate the CPI for each block size which is documented in the graph below:

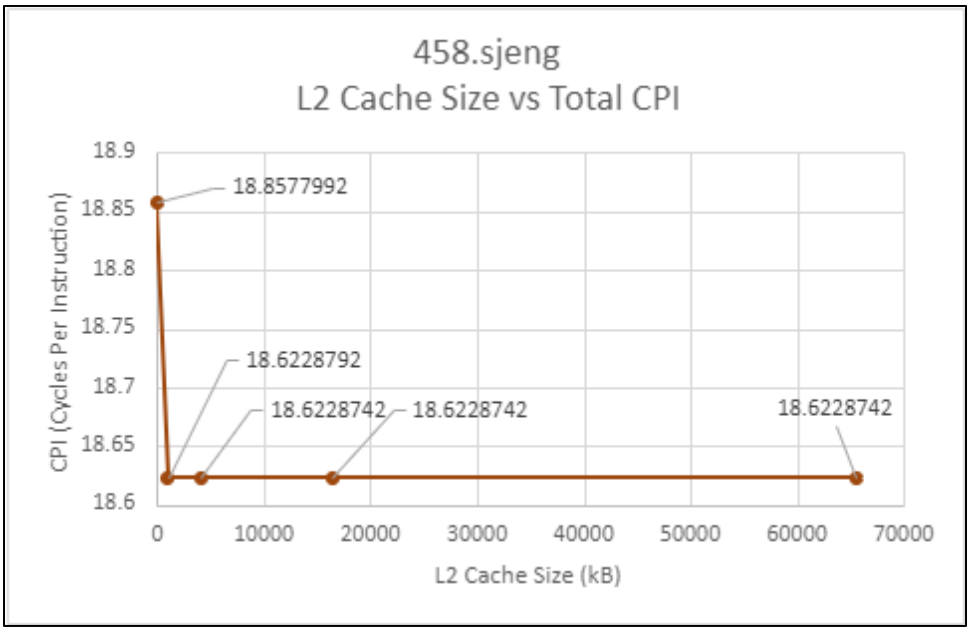


From the 456.hmmer benchmark we can see that the overall CPI again started to decrease but roughly stayed the same after the L2 cache size got larger than 1024 kB. This again is because as the L2 Cache size grew, once reaching an L2 Cache Size of 1024 kB the number of L2 Cache misses stayed roughly the same. Meanwhile during the entire Parameter Test Set, the total number of L1 D and L1 I Cache misses stayed the same since changing the size of the L2 Cache has no effect on the L1 cache whatsoever. That is why after increasing the L2 Cache so much that there was no other effect of benefit in increasing the size.

458.sjeng

L2 Cache Size Results	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
L2 Cache Size	1	1024	4096	16384	65536
Overall Miss Rate I Cache	0.414432	0.414432	0.414432	0.414432	0.414432
Overall Miss Rate D Cache	0.000115	0.000115	0.000115	0.000115	0.000115
Overall Miss Rate L2	0.991343	0.976724	0.976723	0.976723	0.976723
Overall Misses D Cache	3212170	3212170	3212170	3212170	3212170
Overall Misses I Cache	1562	1562	1562	1562	1562
Overall Misses L2	3185912	3138928	3138927	3138927	3138927
Total Instructions	10000000	10000000	10000000	10000000	10000000
Total CPI	18.857799	18.622879	18.622874	18.622874	18.622874

From running the benchmark with only changing L2 Cache size we were able to come up with the data above. This data was then used to calculate the CPI for each block size which is documented in the graph below:



Finally for the 458.sjeng benchmark, we can see that the overall CPI again started to decrease but roughly stayed the same after the L2 cache size got larger than 1024 kB. This again is because as the L2 Cache size grew, once reaching an L2 Cache Size of 1024 kB the number of L2 Cache misses stayed roughly the same. Meanwhile during the entire Parameter Test Set, the total number of L1 D and L1 I Cache misses stayed the same since changing the size of the L2 Cache has no effect on the L1 cache whatsoever. That is why after increasing the L2 Cache so much that there was no other effect of benefit in increasing the size.

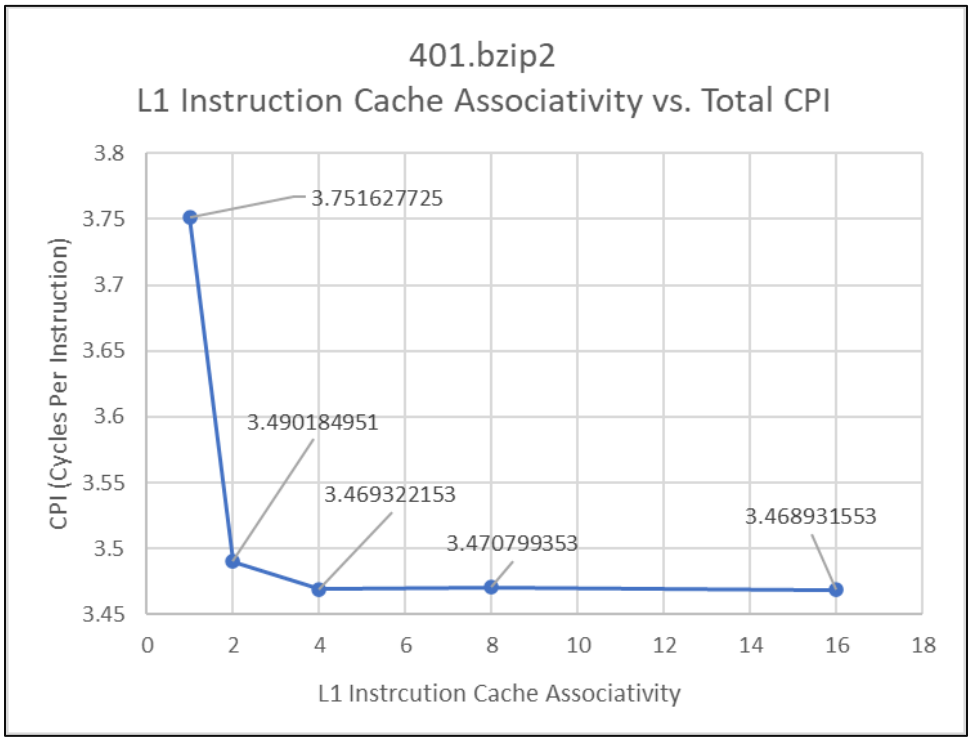
L1 Instruction Cache Associativity

Benchmark Results:

401.bzip2

L1I Assoc Results	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
L1 Instrn Cache Assoc (kB)	1	2	4	8	16
Overall Miss Rate I Cache	0.076852	0.064898	0.063092	0.06288	0.062782
Overall Miss Rate D Cache	0.009128	0.009128	0.009128	0.009128	0.009128
Overall Miss Rate L2	0.717978	0.749301	0.761501	0.764377	0.764798
Overall Misses D Cache	538843	455030	442367	440879	440191
Overall Misses I Cache	117887	117887	117887	117887	117887
Overall Misses L2	471518	429287	426634	427108	426817
Total Instructions	10000001	10000001	10000001	10000001	10000001
Total CPI	3.7516277	3.4901849	3.4693221	3.4707993	3.4689315

From running the benchmark 401.bzip2 with only changing L1 Cache Associativity size, we were able to come up with data that was used to calculate the CPI for each associativity size. The results are documented in the graph below:



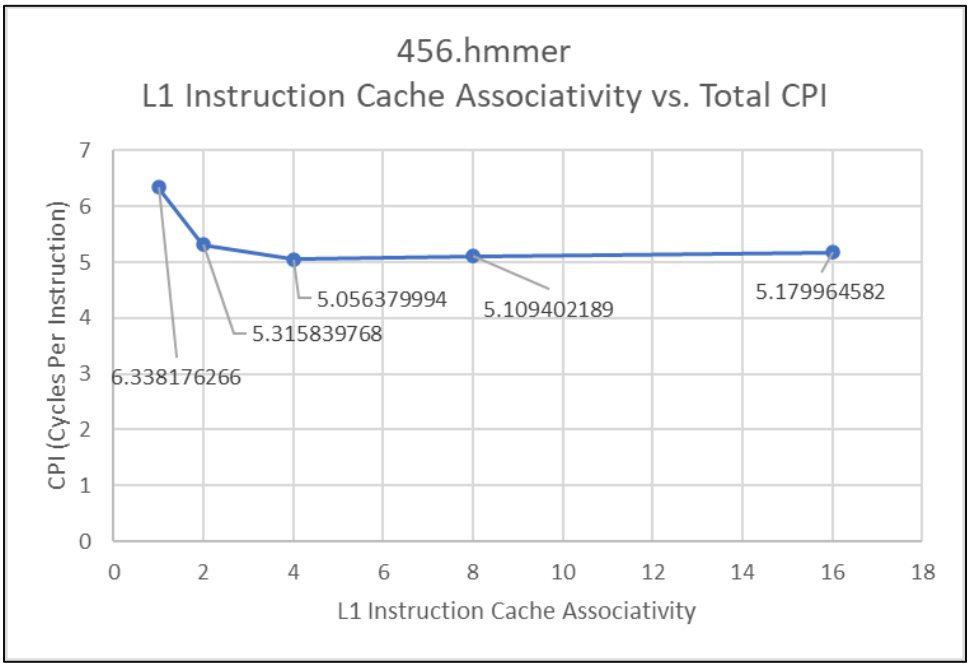
The 401.bzip2 benchmark data shows that the L1 instruction cache associativity size has an impact on the overall miss rate of the cache. As the associativity size increases from 1 to 16, the overall miss rate of the L1 instruction cache decreases gradually for all test cases. However, the L1 data cache miss rate

stays constant for all associativity sizes. Interestingly, the overall miss rate of the L2 cache remains largely unchanged even as the associativity size of the L1 instruction cache increases. This indicates that increasing the L1 instruction cache associativity size does not have a significant impact on the L2 cache's performance. Therefore, optimizing the L1 instruction cache associativity size beyond a certain point may not provide any additional benefits in terms of improving the overall CPI.

456.hmmer

L1I Assoc Results	**Test Case 1**	**Test Case 2**	**Test Case 3**	**Test Case 4**	**Test Case 5**
L1 Instrn Cache Assoc (kB)	1	2	4	8	16
Overall Miss Rate I Cache	0.109233	0.058152	0.043427	0.042986	0.044933
Overall Miss Rate D Cache	0.081279	0.081279	0.081279	0.081279	0.081279
Overall Miss Rate L2	0.511698	0.484643	0.480108	0.488971	0.49487
Overall Misses D Cache	561417	298881	223198	220935	230939
Overall Misses I Cache	1128686	1128686	1128686	1128686	1128686
Overall Misses L2	864823	691860	649050	659926	672838
Total Instructions	10000001	10000001	10000001	10000001	10000001
Total CPI	6.3381762	5.3158397	5.0563799	5.1094021	5.1799645

From running the benchmark 456.hmmer with only changing L1 Cache Associativity size, we were able to come up with data that was used to calculate the CPI for each associativity size. The results are documented in the graph below:



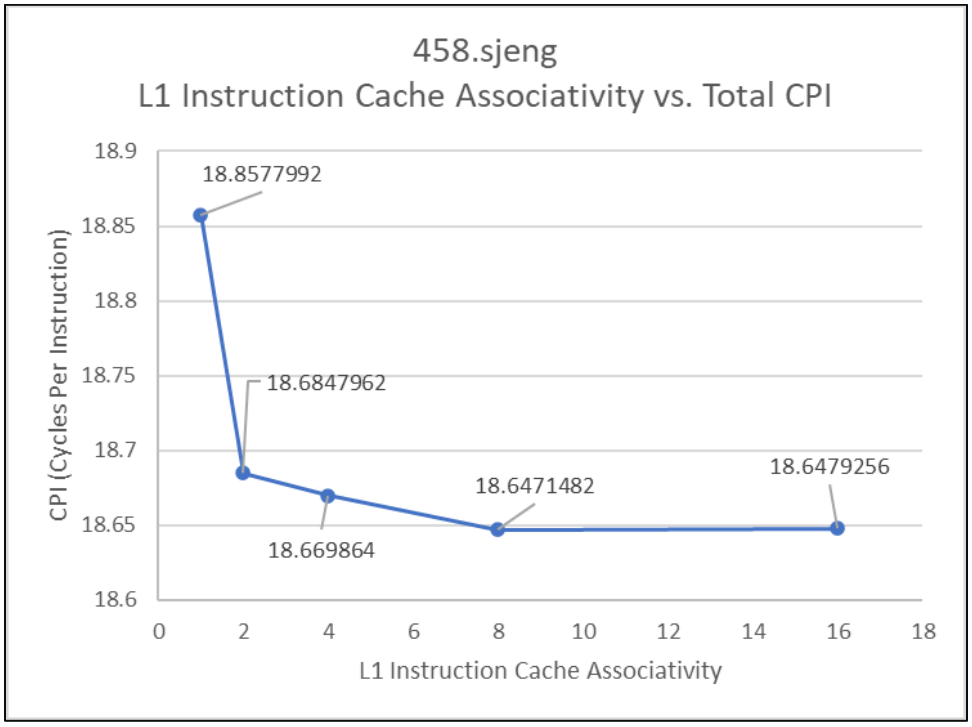
The 456.hmmer benchmark data shows that the L1 instruction cache associativity size on the overall miss rate of the cache. For all test cases, increasing the associativity size from 1 to 16 results in a gradual

decrease in the overall miss rate of the L1 instruction cache. However, the L1 data cache miss rate remains constant regardless of the associativity size. It is interesting to note that despite the improvements in the L1 instruction cache miss rate, the overall miss rate of the L2 cache remains largely unchanged. This suggests that increasing the L1 instruction cache associativity size beyond a certain point may not significantly affect the L2 cache's performance and optimizing it further may not lead to any noticeable improvements in the overall CPI.

458.sjeng:

L1I Assoc Results	**Test Case 1**	**Test Case 2**	**Test Case 3**	**Test Case 4**	**Test Case 5**
L1 Instrn Cache Assoc(kB)	1	2	4	8	16
Overall Miss Rate D Cache	0.000115	0.000115	0.000115	0.000115	0.000115
Overall Miss Rate I Cache	0.414432	0.409074	0.408218	0.406571	0.406587
Overall Miss Rate L2	0.991343	0.994986	0.99638	0.99946	0.999463
Overall Misses D Cache	3212170	3170640	3164003	3151235	3151364
Overall Misses I Cache	1562	1562	1562	1562	1562
Overall Misses L2	3185912	3156295	3154105	3151094	3151234
Total Instructions	10000000	10000000	10000000	10000000	10000000
Total CPI	18.857799	18.684796	18.66986	18.647148	18.647925

From running the benchmark 458.sjeng with only changing L1 Cache Associativity size, we were able to come up with data that was used to calculate the CPI for each associativity size. The results are documented in the graph below:



The 458.sjeng benchmark data shows that the L1 instruction cache associativity size has a significant impact on the overall miss rate of the cache. As the associativity size increases from 1 to 16, the overall miss rate of the L1 instruction cache decreases gradually for all test cases. However, the L1 data cache miss rate stays constant for all associativity sizes. Notably, the overall miss rate of the L2 cache remains mostly unchanged even as the associativity size of the L1 instruction cache increases. This indicates that increasing the L1 instruction cache associativity size may not provide significant benefits in terms of improving the overall CPI beyond a certain point.

L1 Data Cache Associativity

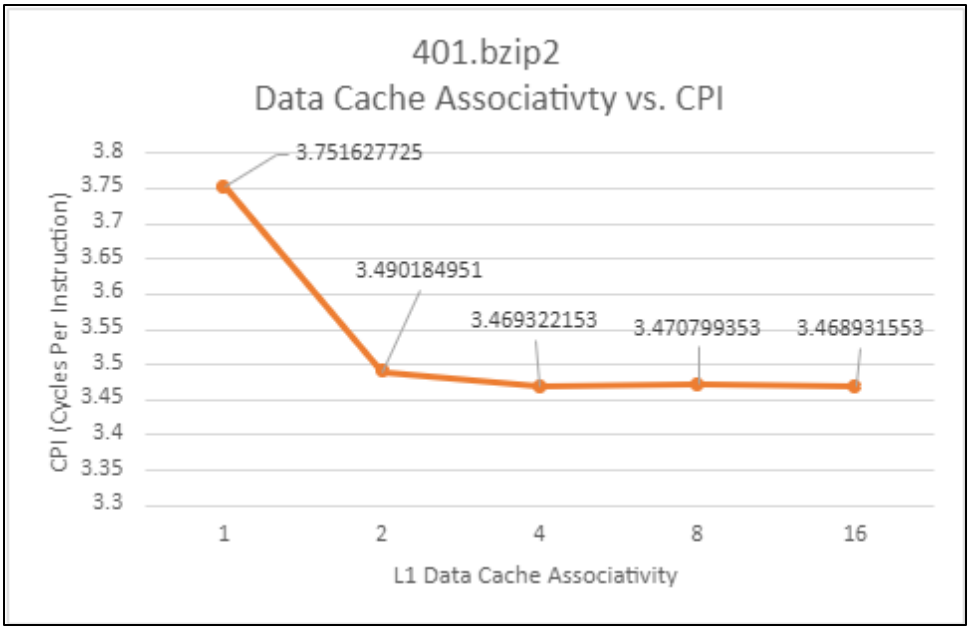
In this section, we analyze varying the L1 Data Cache associativity while keeping other baseline parameters constant amongst the three benchmarks: 401.bzip2, 456.hmmer, and 458.sjeng. For each of these benchmarks, we varied the L1 data cache associativity from [1,16] in powers of two for five test cases. Each of these benchmark results are illustrated by a table displaying the individual misses amongst the test cases, and a graph showing the total CPI as a function of the L1 data cache associativity.

Benchmark Results:

401.bzip2

L1 data Assoc Results	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
L1 Data Cache Assoc	1	2	4	8	16
Overall Miss Rate I Cache	0.076852	0.064898	0.063092	0.06288	0.062782
Overall Miss Rate D Cache	0.009128	0.009128	0.009128	0.009128	0.009128
Overall Miss Rate L2	0.717978	0.749301	0.761501	0.764377	0.764798
Overall Misses D Cache	538843	455030	442367	440879	440191
Overall Misses I Cache	117887	117887	117887	117887	117887
Overall Misses L2	471518	429287	426634	427108	426817
Total Instructions	10000001	10000001	10000001	10000001	10000001
Total CPI	3.75162772	3.49018495	3.46932215	3.47079935	3.46893155

In this benchmark, we see that the overall miss rate for the instruction cache decreases gradually which contributes to an overall increase in the L2 cache.

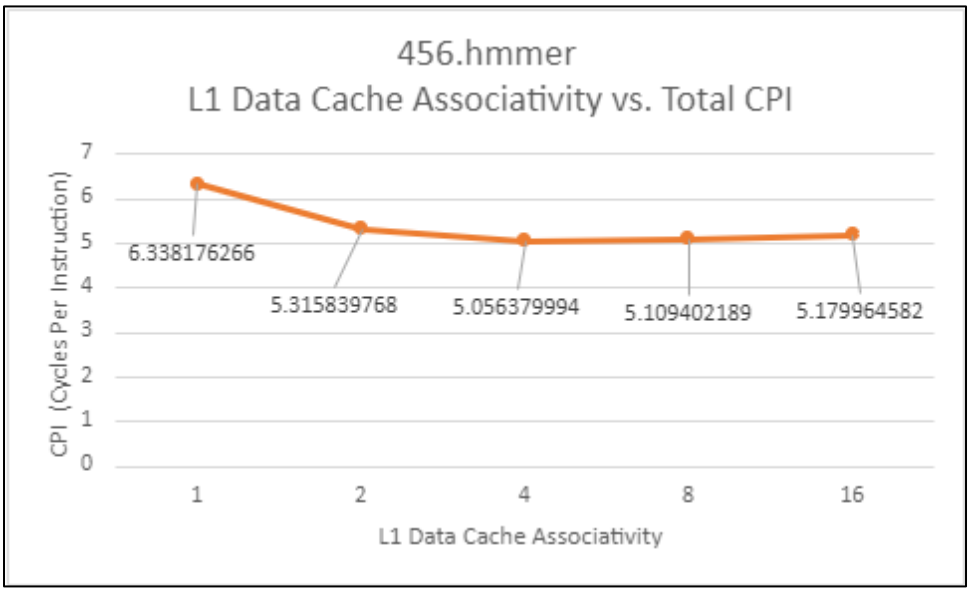


Furthermore, in the graph we see that this contributes to an overall decrease in CPI with each test case in increasing the data cache associativity for this benchmark. Although this decrease decelerates when the data cache associativity reaches 2 to 4 memory locations in each set. Extrapolating suggests that further increasing the associativity may not decrease the CPI as much and may not be worth the extra hardware cost. This may be due to the fact the though set associativity increases the diversity memory address data that can be stored in cache, this has less of an effect on the performance of the program as the associativity increases beyond 4.

456.hmmer

L1 data cache Assoc Results	**Test Case 1**	**Test Case 2**	**Test Case 3**	**Test Case 4**	**Test Case 5**
L1 Data Cache Assoc	1	2	4	8	16
Overall Miss Rate D Cache	0.109233	0.058152	0.043427	0.042986	0.044933
Overall Miss Rate I Cache	0.081279	0.081279	0.081279	0.081279	0.081279
Overall Miss Rate L2	0.511698	0.484643	0.480108	0.488971	0.49487
Overall Misses D Cache	561417	298881	223198	220935	230939
Overall Misses I Cache	1128686	1128686	1128686	1128686	1128686
Overall Misses L2	864823	691860	649050	659926	672838
Total Instructions	10000001	10000001	10000001	10000001	10000001
CPI	6.338176266	5.315839768	5.05637999	5.109402189	5.179964582

From these results from this benchmark, we see that increasing the data cache associativity temporarily decreases the overall misses of the data cache which slightly rebounds after test case 4.

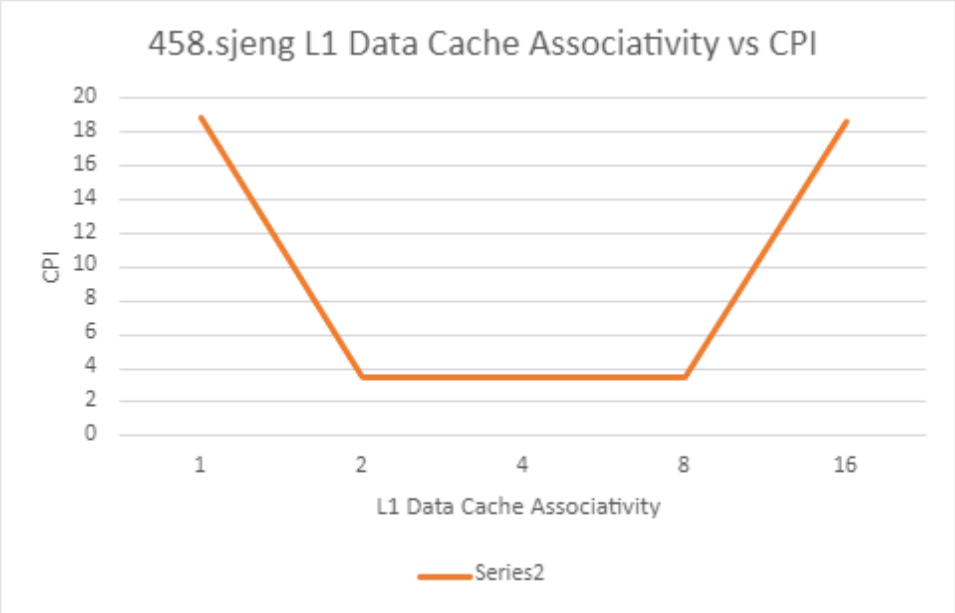


This graph also illustrates how this overall data cache change results in an initial decrease in overall CPI followed by a slight rebound when the set associativity exceeds 4 memory locations per set. Extrapolating may suggest that this slight increase may either continue to increase slightly or accelerate after more iterations of set associativity. This is due to the fact that although more locations per cache block will increase the number of hits in the cache, continuously increasing it further also makes the search time and thus clock cycles needed for the hit to increase. This may explain the slight rebound in CPI after test case 4. This shows the cost of further increasing the set associativity past the threshold of 4 memory locations per block will outweigh the improvement on CPI.

458.sjeng

L1 Data Cache Assoc Results	**Test Case 1**	**Test Case 2**	**Test Case 3**	**Test Case 4**	**Test Case 5**
L1 Data Cache Assoc	1	2	4	8	16
Overall Miss Rate D Cache	0.414432	0.064898	0.063092	0.06288	0.406587
Overall Miss Rate I Cache	0.000115	0.009128	0.009128	0.009128	0.000115
Overall Miss Rate L2	0.991343	0.749301	0.761501	0.764377	0.999463
Overall Misses D Cache	3212170	455030	442367	440879	3151364
Overall Misses I Cache	1562	117887	117887	117887	1562
Overall Misses L2	3185912	429287	426634	427108	3151234
Total Instructions	10000000	10000001	10000001	10000001	10000000
CPI	18.8577992	3.49018495	3.46932215	3.47079935	18.6479256

We see from the results of this benchmark that the overall data cache misses sharply decreases when the L1 data cache reaches a set associativity of 2, remains mostly flat for the next two test cases, and then sharply increases again at a similar rate at which it initially decreased. These overall L1 data cache misses are calculated into the overall misses.



This graph shows how this change in overall misses is reflected in the total CPI. CPI sharply decreases to a minimum of approximately 4 clock cycles per instructions until it sharply rebounds when the set associativity exceeds 8 memory locations per set. This suggests that for this benchmark the performance of the program can be significantly improved when modifying the data cache set associativity to be 4. However, further increases after this may result in the program mapping the same memory locations to most of the cache, resulting in the cache filling up faster with less diverse memory addresses, thus resulting in a larger miss rate over the executions of the program.