CSE 490/590, Project 2, Spring 2023
Analysis Of Cache Parameters And Trade-Offs Using Gem5

Team Details

Aakash Baruva – 50457184 - abaruva
Harshavardhan Sivadanam – 50478880 - hsivadan
Priyanka Chakraborty – 50463802 - priyank2
Sai Sree Depa – 50478013 - saisreed

1 Parameters

L1 Data Memory Cache Size:

The cache size parameter determines the amount of data memory that can be cached, enabling more data to be stored closer to the processor. This reduces the time taken to access data from main memory and can improve performance. Consequently, increasing the cache size generally leads to better performance.

L1 Instruction Memory Cache Size:

The instruction memory cache size parameter determines the amount of instruction memory that can be stored in the cache. A larger cache size increases performance by enabling more instructions to be stored in close to the CPU, similar to how the L1 data memory cache operates.

L1 Data Memory Associativity:

This parameter determines the configuration of the data memory cache, specifically, the number of cache lines that can be mapped to a single cache set, which is known as associativity. Increasing the associativity level enables more cache lines to be stored in each cache set, reducing the likelihood of cache conflicts and improving performance.

L2 Associativity:

This parameter determines the configuration of the L2 cache, and, similar to the L1 data memory cache, increasing its associativity level can improve performance. Higher associativity levels allow for the storage of more cache lines in each cache set, reducing the probability of cache conflicts and improving overall performance.

Block Size:

The block size parameter controls the size of the blocks used to store data in the cache. A larger block size can improve performance by allowing more data to be accessed from memory in a single cache request. However, if the cache size is limited, increasing the block size may result in more cache conflicts, reducing overall performance.

2 Procedure (Steps Involved)

Step-1: Understanding the Assigned Parameters

We were assigned three benchmarks:
401.bzip2
429.mcf
456.hmmer
We were given five Cache parameters:
L1 Data Memory Cache Size
L1 Instruction Memory Cache Size
L1 Data Memory Associativity
L2 Associativity
Block Size
For each benchmark, we were instructed to run a Benchmark Test Set consisting of five Parameter Test Sets, each of which contained five test cases.

Step-2: Defining a baseline with parameters fixed

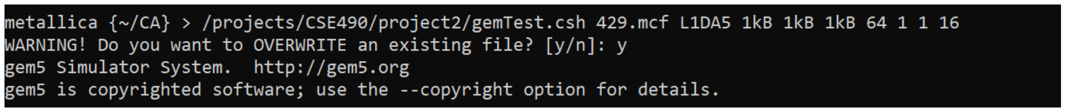
For each Benchmark, we defined baseline parameters as:
L1 Data Memory Cache Size = 1kB
L1 Instruction Memory Cache Size = 1kB
L1 Data Memory Associativity = 1kB
L2 Associativity = 1
Block Size = 16
For each test case, we changed one parameter value and others are fixed.

Step-3: Parameter Test Cases

To evaluate the impact of each cache parameter on the benchmarks' performance, we designed five sets of parameters for each benchmark. Each set of parameters was intended to test a particular cache parameter. Within each parameter set, we included five distinct test cases, each evaluating the performance of a different value for the corresponding cache parameter. We executed the tests using the provided gemTest.csh shell script. Keeping all other parameters fixed, vary one parameter only with five different values which creates five different test cases. Example: In case of L1 Data Memory Cache Size takes value as 1kB, 2kB, 4kB, 8kB and 16kB. In case of L1 Instruction Memory Cache Size takes value as 2kB, 8kB, 16kB, 32kB and 64kB and simiarly for others.

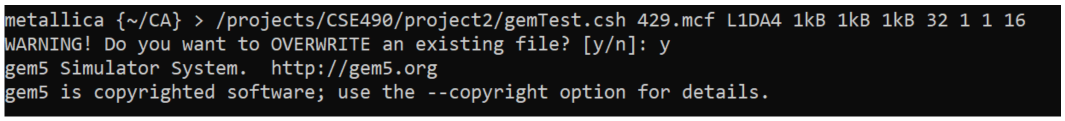
Step-4: Running the Script

To run the script, we initially go to the folder project2 and ran each Parameter Test Set for each benchmark using gemTest.csh script. The following screenshots depicts the sample commands for benchmark 429.mcf used for running various test parameters:



```
metallica {~/CA} > /projects/CSE490/project2/gemTest.csh 429.mcf L1DA5 1kB 1kB 1kB 64 1 1 16
WARNING! Do you want to OVERWRITE an existing file? [y/n]: y
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.
```

Figure 1: Running the Command

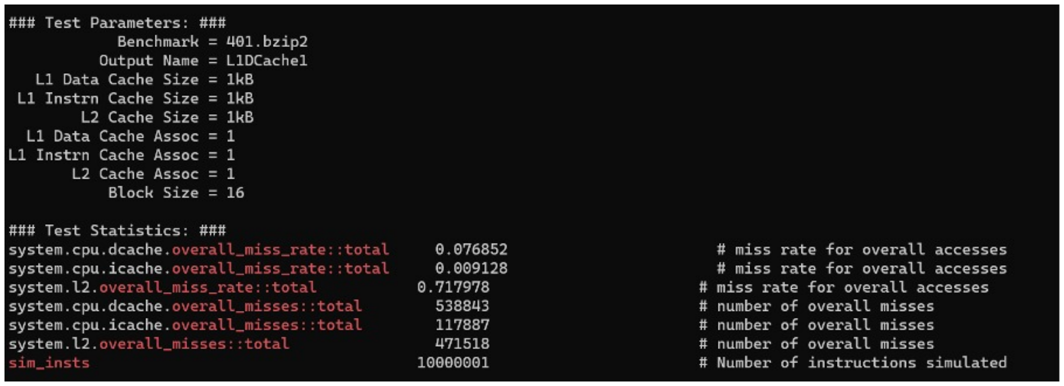


```
metallica {~/CA} > /projects/CSE490/project2/gemTest.csh 429.mcf L1DA4 1kB 1kB 1kB 32 1 1 16
WARNING! Do you want to OVERWRITE an existing file? [y/n]: y
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.
```

Figure 2: Running the Command

Step-5: Execution Results

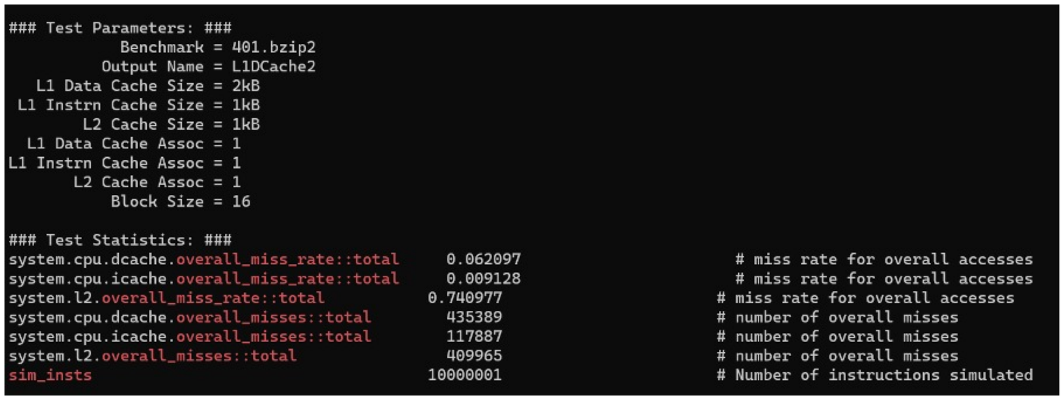
After running the command, we get the results for each benchmark. For instance, the screenshot below shows the test case for benchmark 401.bzip2, where L1 Data Memory Cache Size is varied and other parameters are fixed:



```
### Test Parameters: ###
            Benchmark = 401.bzip2
          Output Name = L1DCache1
   L1 Data Cache Size = 1kB
 L1 Instrn Cache Size = 1kB
        L2 Cache Size = 1kB
  L1 Data Cache Assoc = 1
L1 Instrn Cache Assoc = 1
       L2 Cache Assoc = 1
           Block Size = 16

### Test Statistics: ###
system.cpu.dcache.overall_miss_rate::total     0.076852                       # miss rate for overall accesses
system.cpu.icache.overall_miss_rate::total     0.009128                       # miss rate for overall accesses
system.l2.overall_miss_rate::total           0.717978                       # miss rate for overall accesses
system.cpu.dcache.overall_misses::total        538843                       # number of overall misses
system.cpu.icache.overall_misses::total        117887                       # number of overall misses
system.l2.overall_misses::total                471518                       # number of overall misses
sim_insts                                    10000001                       # Number of instructions simulated
```

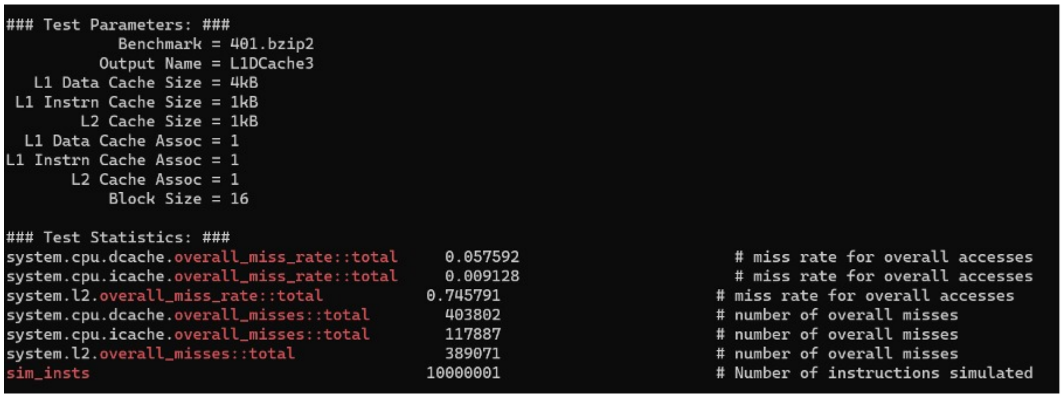
Figure 3: Test 1: Varied L1 Data Memory Cache Size as 1kB



```
### Test Parameters: ###
            Benchmark = 401.bzip2
          Output Name = L1DCache2
   L1 Data Cache Size = 2kB
 L1 Instrn Cache Size = 1kB
        L2 Cache Size = 1kB
  L1 Data Cache Assoc = 1
L1 Instrn Cache Assoc = 1
       L2 Cache Assoc = 1
           Block Size = 16

### Test Statistics: ###
system.cpu.dcache.overall_miss_rate::total     0.062097                       # miss rate for overall accesses
system.cpu.icache.overall_miss_rate::total     0.009128                       # miss rate for overall accesses
system.l2.overall_miss_rate::total           0.740977                       # miss rate for overall accesses
system.cpu.dcache.overall_misses::total        435389                       # number of overall misses
system.cpu.icache.overall_misses::total        117887                       # number of overall misses
system.l2.overall_misses::total                409965                       # number of overall misses
sim_insts                                    10000001                       # Number of instructions simulated
```

Figure 4: Test 2: Varied L1 Data Memory Cache Size as 2kB



```
### Test Parameters: ###
            Benchmark = 401.bzip2
          Output Name = L1DCache3
   L1 Data Cache Size = 4kB
  L1 Instrn Cache Size = 1kB
        L2 Cache Size = 1kB
   L1 Data Cache Assoc = 1
L1 Instrn Cache Assoc = 1
       L2 Cache Assoc = 1
           Block Size = 16

### Test Statistics: ###
system.cpu.dcache.overall_miss_rate::total     0.057592                       # miss rate for overall accesses
system.cpu.icache.overall_miss_rate::total     0.009128                       # miss rate for overall accesses
system.l2.overall_miss_rate::total           0.745791                       # miss rate for overall accesses
system.cpu.dcache.overall_misses::total        403802                       # number of overall misses
system.cpu.icache.overall_misses::total        117887                       # number of overall misses
system.l2.overall_misses::total                389071                       # number of overall misses
sim_insts                                    10000001                       # Number of instructions simulated
```

Figure 5: Test 3: Varied L1 Data Memory Cache Size as 4kB

Step-6: Analysing the Results

For each test case, we recorded the following information: Hit rate of L1 D cache, Hit rate of L1 I cache, Hit rate of L2 cache, and CPI. The Hit rates of each parameter are calculated by using the formula **Hit_rate = 1- Miss_rate**. Where the Miss rates are taken from the output we get after running the code.

• To calculate the Hit Rate of L1 data cache:

We can get the miss rate of the L1 data cache from the stat_files it is of the form **system.cpu.dcache.overall_miss_rate::total** and the we can use this formula to calculate the hit rate.
Hit Rate = 1- Miss Rate of L1 data cache

• Similarly, we calculate the Hit Rate of L1 instruction cache:

We can get the miss rate of the L1 data cache from the stat_files it is of the form **system.cpu.icache.overall_miss_rate::total** and the we can use this formula to calculate the hit rate.
Hit Rate = 1- Miss Rate L1 instruction cache

• Similarly, we calculate the Hit Rate of L2 cache:
We can get the miss rate of the L1 data cache from the stat_files it is of the form **system.cpu.l2.overall_miss_rate::total** and the we can use this formula to calculate the hit rate.

Hit Rate = 1- Miss Rate L2 cache

Once the Hit values are recorded, we calculate the CPI, using the formula:

$$CPI = 1 + \frac{(IL1.miss\_num + DL1.miss\_num) \times 6 + L2.miss\_num \times 50}{Total\_Inst\_num}$$

Figure 6: CPI formula

We can obtain these from the stat_files
IL1.miss_num = system.cpu.icache.overall_misses::total (number of overall misses)
DL1.miss_num = system.cpu.dcache.overall_misses::total (number of overall misses)
L2.miss_num = system.I2overall_misses::total (number of overall misses)
Total_Inst_num = sim_insts::total (number of instructions simulated)

Step-7: Plotting graphs on variation of CPI w.r.t to all five parameters

From the computed CPI values, we have plotted various graphs and analyzed each of them. The detailed explanation of graph has been explained in the below section. The sample graphs can be seen from below figures:

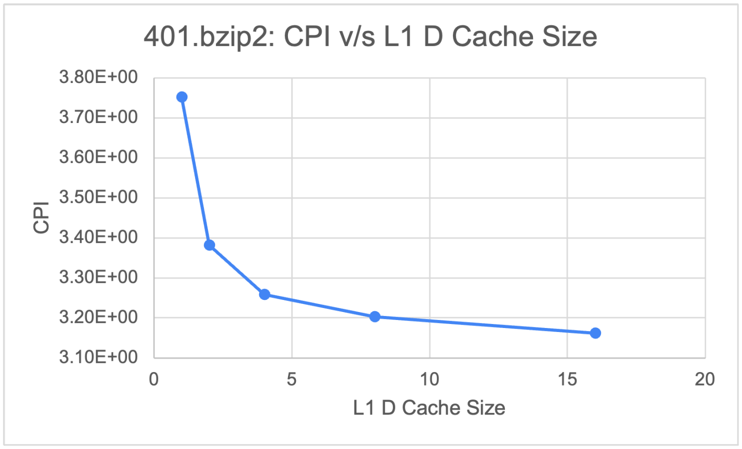


Figure 7: 401.bzip2: CPI v/s L1 D Cache Size

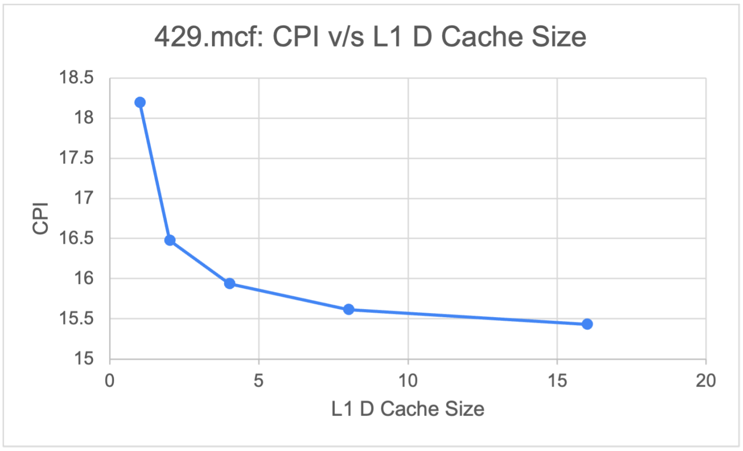


Figure 8: 429.mcf: CPI v/s L1 D Cache Size

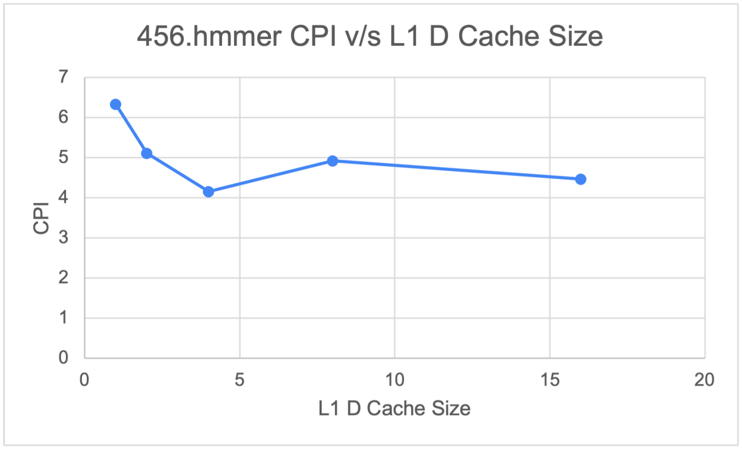


Figure 9: 456.hmmer CPI v/s L1 D Cache Size

3 Trends in the Graphs and Explanation

1. CPI v/s L1 D Cache Size

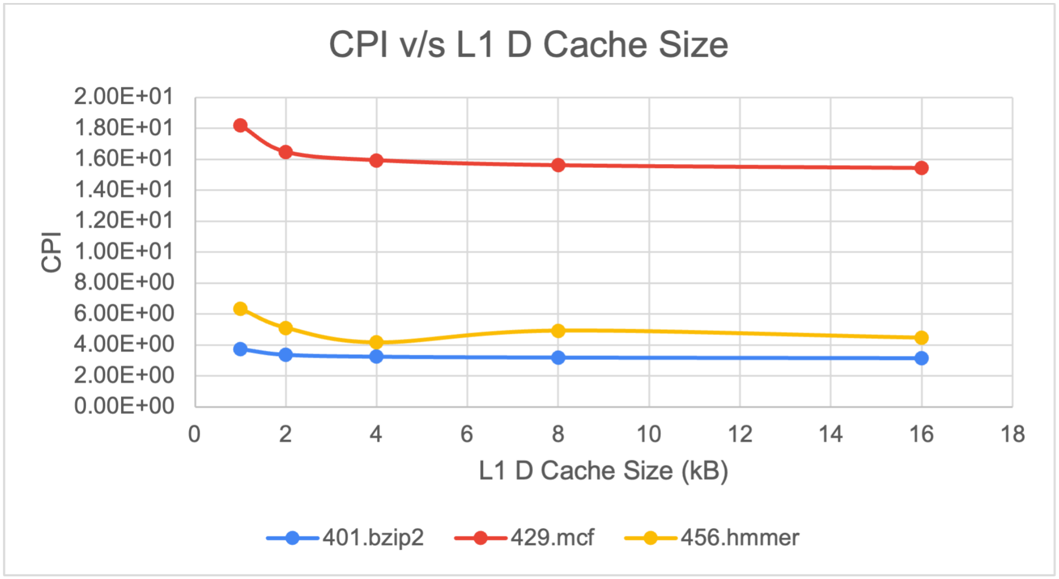


Figure 10: CPI v/s L1 D Cache Size

In this case, L1 Data Memory Cache Size parameter has been varied with values as 1kB, 2kB, 4kB, 8kB and 16kB respectively. The remaining parameters, L1 Instruction Memory Cache Size = 1kB L1 Data Memory Associativity = 1kB L2 Associativity = 1 Block Size = 16 are placed constant for the Benchmark 402.bzip2, 429.mcf and 456.hmmer.

A graph is plotted between the CPI and L1 data Memory Cache Size for all the benchmarks:
1. We can observe from the figure that the CPI value reduces for all three benchmarks as the size of the L1 D cache rises. According to this, increasing cache size should result in less cache misses as well as faster execution times.
2. With respect to all cache sizes, the **429.mcf** benchmark has the **highest CPI** values and has the greatest improvements in performance. This shows that compared to the other two benchmarks, 429.mcf is more sensitive to cache size.
3. Among the three benchmarks, **401.bzip2** has the **lowest CPI** values regardless of the cache

size used, and its performance shows only a small improvement as the cache size increases. This observation indicates that 401.bzip2 is relatively less affected by variations in cache size when compared to the other two benchmarks.
4. The benchmark **456.hmmer** shows moderate improvements in performance with increasing cache size and **intermediate CPI levels**. This result suggests that 456.hmmer is only slightly sensitive to changes in cache size.
5. Overall, the graph shows that increasing the size of the L1 D cache can significantly improve these benchmarks performance

2. CPI v/s L1 I Cache Size

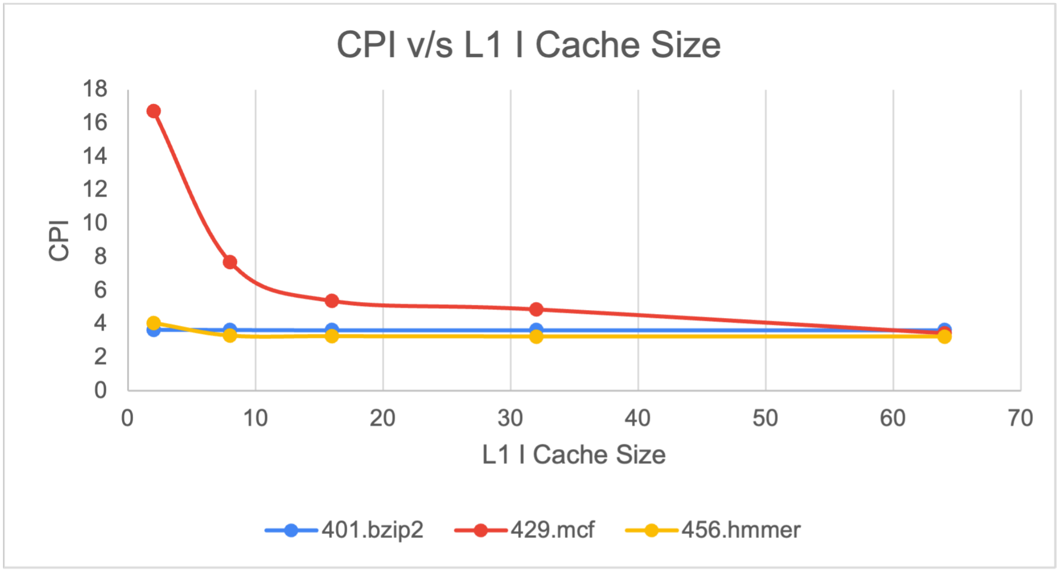


Figure 11: CPI v/s L1 I Cache Size

In this case, L1 Instruction Memory Cache Size parameter has been varied with values as 2kB, 8kB, 16kB, 32kB and 64kB respectively. The remaining parameters, L1 Data Memory Cache Size = 1kB L1 Data Memory Associativity = 1kB L2 Associativity = 1 Block Size = 16 are placed constant for the Benchmark 402.bzip2, 429.mcf and 456.hmmer.

A graph is plotted between the CPI and L1 Instruction Memory Cache Size for all the benchmarks:
1. As the L1 Instruction Cache Size increases, the CPI value decreases for all three benchmarks - 401.bzip2, 429.mcf, and 456.hmmer.
2. It can be understood that by increasing the L1 Instruction Cache Size, the cache can store a greater number of instructions. This effectively decreases the frequency with which the processor must retrieve instructions from slower memory sources like the L2 cache or main memory.
3. From the graph, for the benchmark 401.bzip2, we can see that the CPI decreases as the cache size increases from 2 kB to 64 kB. For the benchmark 429.mcf, the CPI decreases from 16.73 to 3.46 as the cache size increases from 2 kB to 64 kB. Similarly, for the benchmark 456.hmmer, the CPI decreases as the cache size increases from 2 kB to 64 kB.
4. We can see that decreasing CPI values can be significantly impacted by increasing the size of the L1 Instruction Cache, especially for benchmarks **429.mcf** that depend greatly on instruction retrieval. 5. We can also observe that the effect differs among benchmarks and is small for **401.bzip2**. Therefore, when increasing cache sizes, it is essential to carefully analyze the performance characteristics for specific benchmarks.
6.In general, we can come to the conclusion that increasing the L1 Instruction Cache Size can significantly affect a CPU’s performance by reducing the number of instructions that must be fetched from main memory and, thus, reducing the CPI values for the benchmarks.

3. CPI v/s L2 Associativity

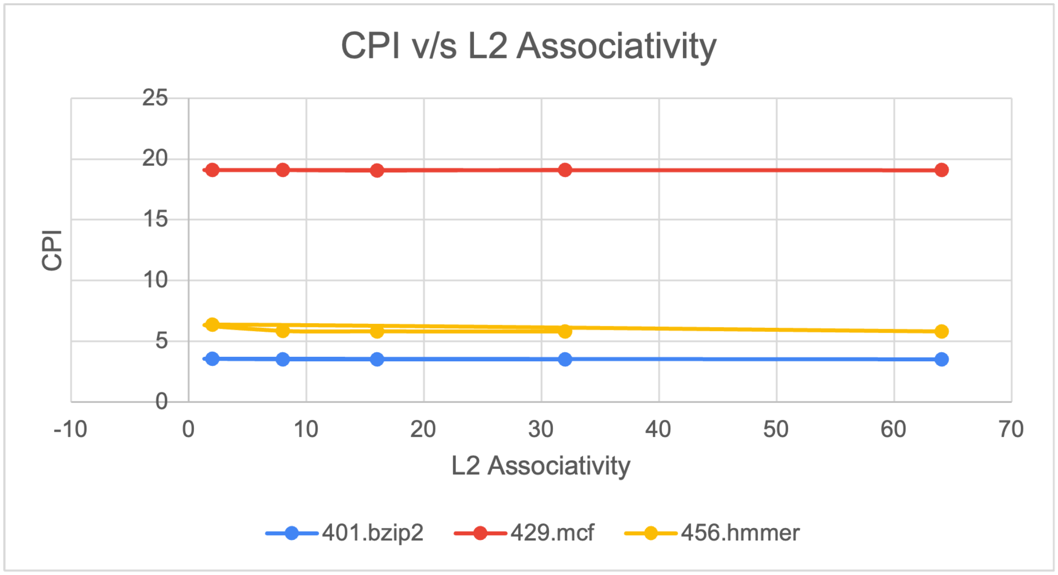


Figure 12: CPI v/s L2 Associativity

In this case, L2 Associativity parameter has been varied with values as 64kB, 2kB, 8kB, 16kB and 32kB respectively. The remaining parameters, L1 Data Memory Cache Size = 1kB L1 Instruction Memory Cache Size = 1kB L1 Data Memory Associativity = 1kB Block Size = 16 are placed constant for the Benchmark 402.bzip2, 429.mcf and 456.hmmer.

A graph is plotted between the CPI and L2 Associativity for all the benchmarks:
1. The data and instructions that cannot fit in the smaller, faster L1 cache are stored in the L2 cache, a larger, slower cache that acts as a secondary cache for the processor.
2. The outcomes demonstrate that the associativity of the L2 cache has little to no effect on the benchmarks performance. In general, all of the curves are flat and near to one another, showing that the associativity of the L2 cache has little effect on the benchmarks performance.
3. The results show that decreasing the associativity of the **L2 cache from 2 to 8** results in a decrease in CPI values for each of the three benchmarks. This indicates that a **larger L2 cache associativity improves performance** by allowing for more flexibility when mapping data and instructions to cache lines, reducing the average CPI and decreasing the frequency of cache misses.
4. Although having shown that increasing L2 cache associativity improves performance, the findings indicate that beyond associativity of 8, the positive effects has no significant effect. For some setups, the CPI values even slightly increase, indicating that increasing associativity beyond a certain point could not result in significant improvements in performance and might instead introduce extra overheads like longer access times.
5. Performance varies slightly for each benchmark, with 429.mcf being the most and 401.bzip2 being the least sensitive to the L2 associativity. Even yet, there is not a lot of a performance difference between associativity levels in the case of 429.mcf.
6. To conclude, these findings imply that the performance advantages of increasing the associativity of the L2 cache are minor for these benchmarks and this cache size.

4. CPI v/s L1 D Associativity

In this case, L1 D Associativity parameter has been varied with values as 4kB, 8kB, 16kB, 32kB and 64kB respectively. The remaining parameters, L1 Data Memory Cache Size = 1kB L1 Instruction Memory Cache Size = 1kB L2 Associativity = 1kB Block Size = 16 are placed constant for the

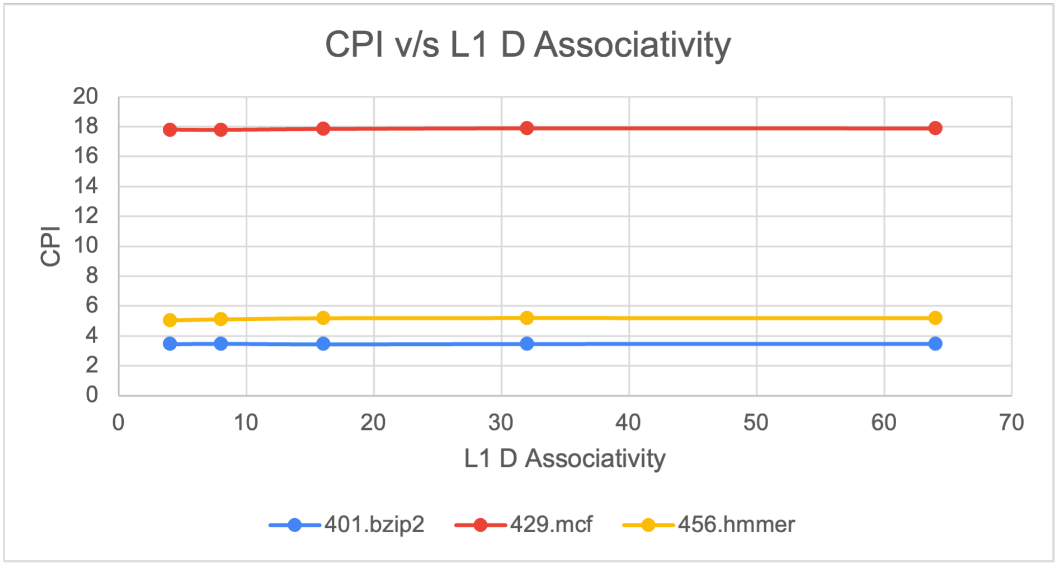


Figure 13: CPI v/s L1 D Associativity

Benchmark 402.bzip2, 429.mcf and 456.hmmer.

A graph is plotted between the CPI and L1 D Associativity for all the benchmarks:
1. In general, increasing the associativity value of the cache improves its flexibility in terms of data storage, reducing the frequency of cache conflicts. More data can be stored in the cache and there are fewer chances of conflicts when mapping memory locations to cache entries when associativity is higher.
2. When we look at the graph, we can see that there are just slight variations in the CPI values for each benchmark across the various associativity values. This shows that the L1 D cache's associativity has little to no effect on the performance of these benchmarks run.
3. For example, in the case of the **429.mcf** benchmark, the CPI values range from 17.7698786 to 17.887883 across the different associativity values, which is a relatively small difference. Similarly, for the **456.hmmer** benchmark, the CPI values range from 5.056379994 to 5.193487781 across the different associativity values.
4. We can conclude that improving the L1 D cache's associativity could not significantly affect the effectiveness of these benchmarks performance. This is most likely due to the small amount of data reuse in the benchmarks used, which means that increasing associativity has no effect on cache hit rates. Furthermore, the L1 data cache tends to be already very associative , so adding more associativity might not be of much use.

5. CPI v/s Block Size

In this case, Block Size parameter has been varied with values as 8, 16, 32, 64, 128 respectively. The remaining parameters, L1 Data Memory Cache Size = 1kB L1 Instruction Memory Cache Size = 1kB L1 D Associativity = 1kB L2 Associativity = 1 are placed constant for the Benchmark 402.bzip2, 429.mcf and 456.hmmer.

A graph is plotted between the CPI and Block Size for all the benchmarks:
1. The amount of data that can be stored in each cache line or block depends on the block size. A block containing the data needed is loaded into the cache when data is requested. If the following data requests are within the same block, they can be immediately executed from the cache without requiring a memory access.
2. In general, larger block sizes can improve performance by taking advantage of temporal and spatial locality.
3. We can see that for **401.bzip2**, the CPI is highest for a block size of 64 bytes and lowest for a block

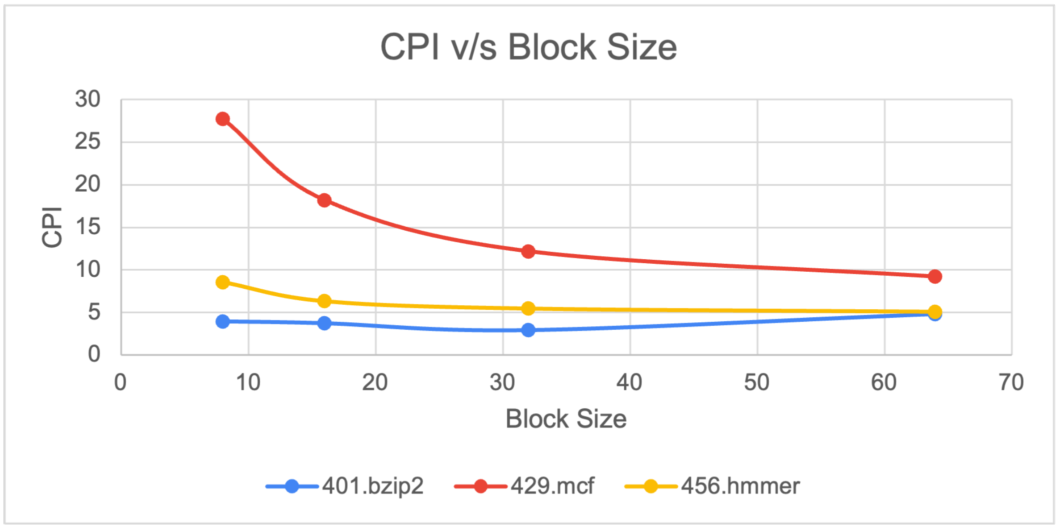


Figure 14: CPI v/s Block Size

size of 32 bytes. This suggests that the best block size for this benchmark is 32 bytes. Requiring to fetch too many blocks from memory leads in a high CPI when the block size is too small (8 bytes). A block size of too many (128 bytes) leads to an increased number of conflict misses and a high CPI.
4. We can observe that the CPI for **429.mcf** declines as block size increases with a block size of 128 bytes giving the lowest CPI. This shows that this benchmark's optimal block size is greater than 401.bzip2's. The number of cache misses is decreased and the cache hit rate is increased with a bigger block size.
5. With respect to **456.hmmer**, we observe that the CPI is highest for blocks of 8 bytes, declines as blocks increase in size, and is lowest for blocks of 64 bytes. This implies that for this benchmark, a larger block size is preferable. A higher block size decreases the amount of cache misses, which are costly in this test because the data access is sequential.
6. Overall, the optimal block size may change from benchmark to benchmark depending on the access patterns of the specific benchmarks and how the block size affects cache performance.

4 All Graphs:

4.1 Graphs for Varied L1 Data Memory Cache Size

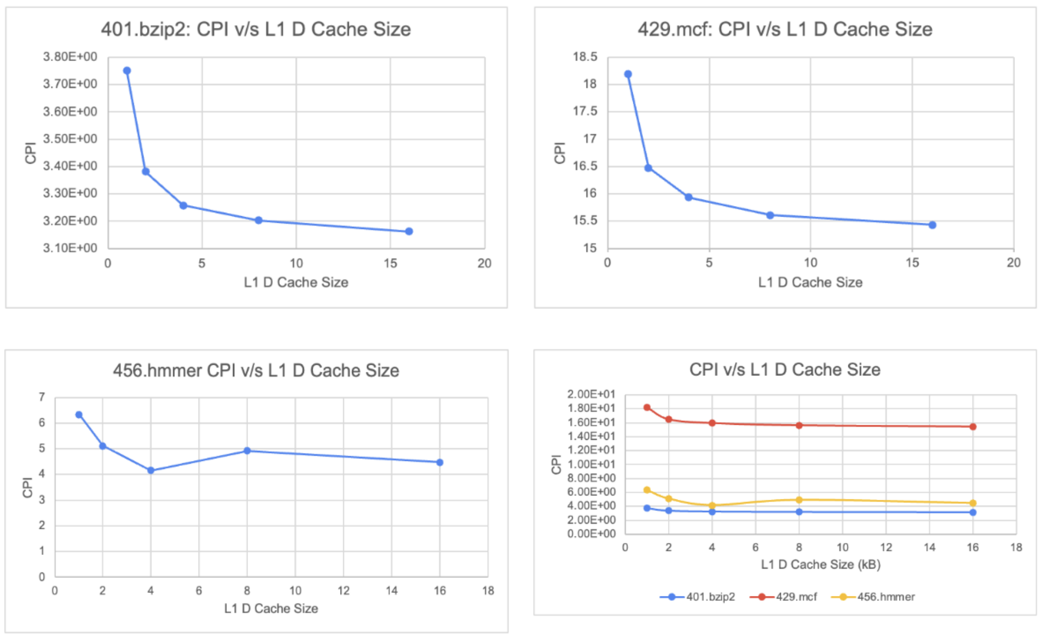


Figure 15: Varied L1 Data Memory Cache Size

4.2 Graphs for Varied L1 Instruction Memory Cache Size

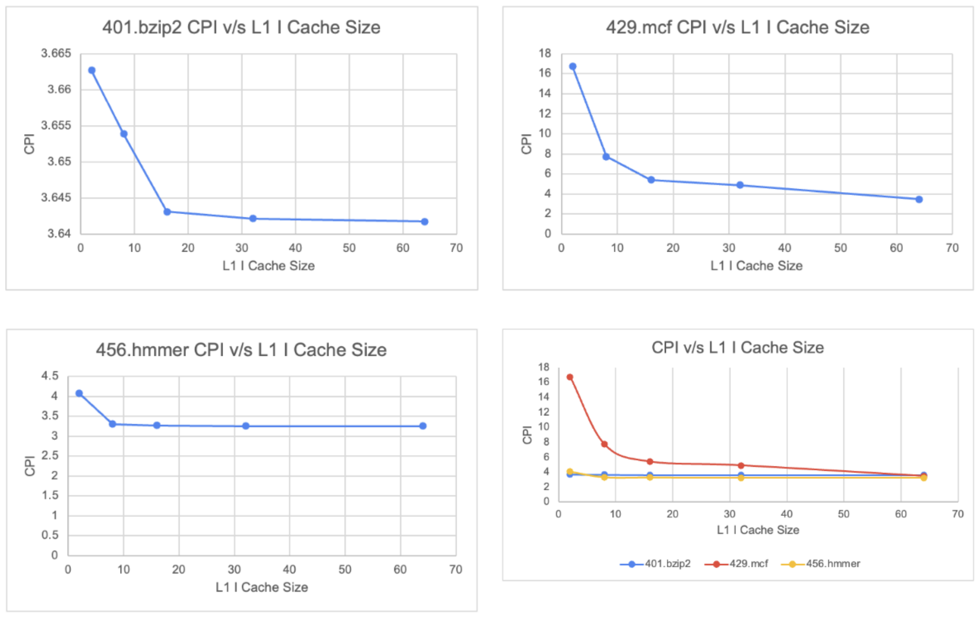


Figure 16: Varied L1 Instruction Memory Cache Size

4.3 Graphs for Varied L1 Data Memory Associativity

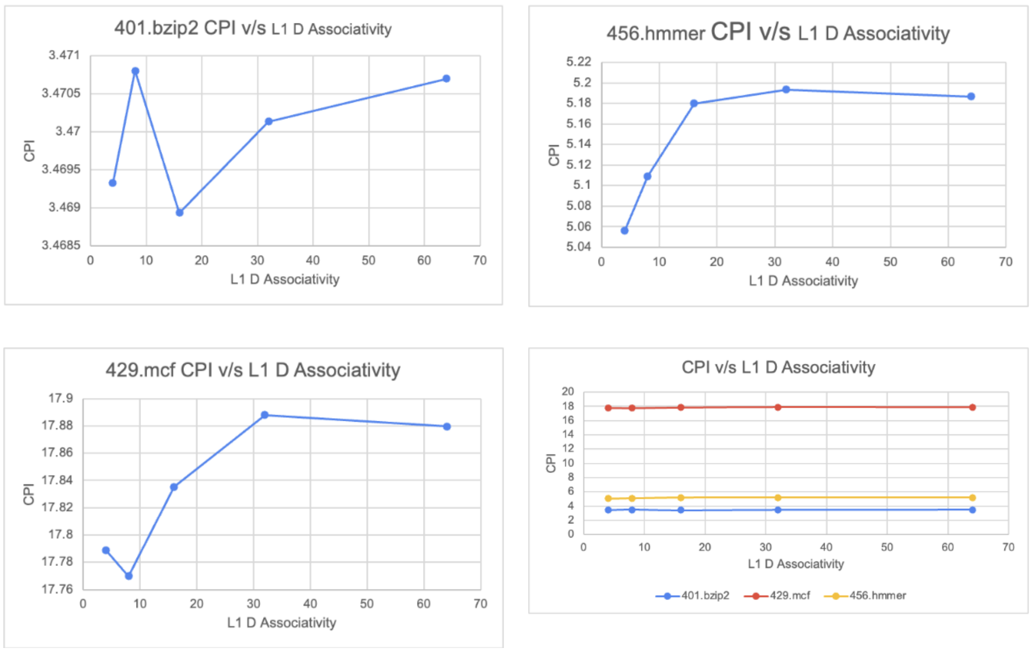


Figure 17: Varied L1 Data Memory Associativity

4.4 Graphs for Varied L2 Associativity

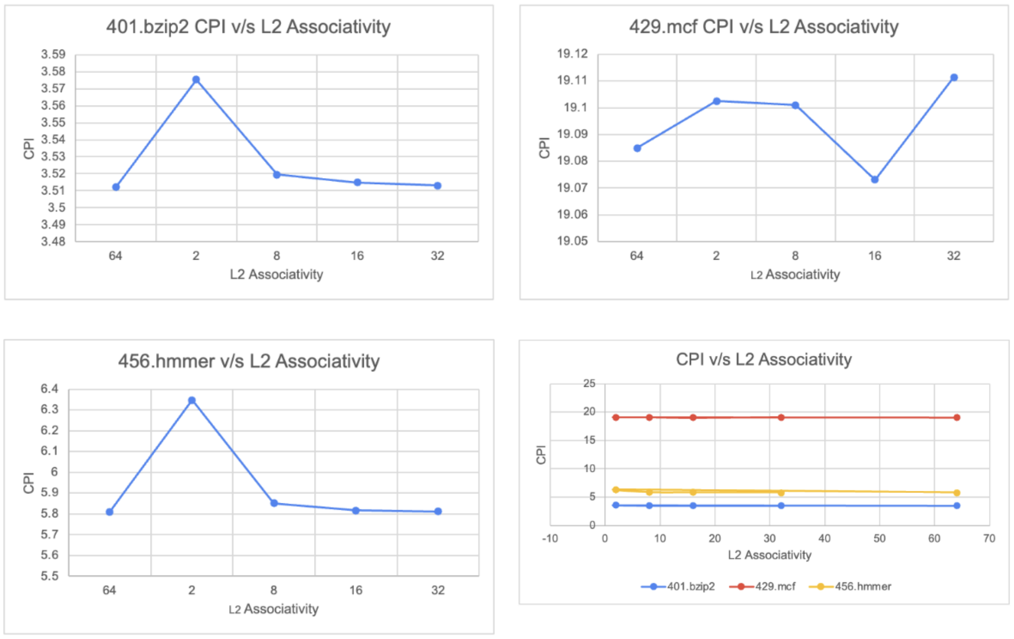


Figure 18: Varied L2 Associativity

4.5 Graphs for Varied Block Size

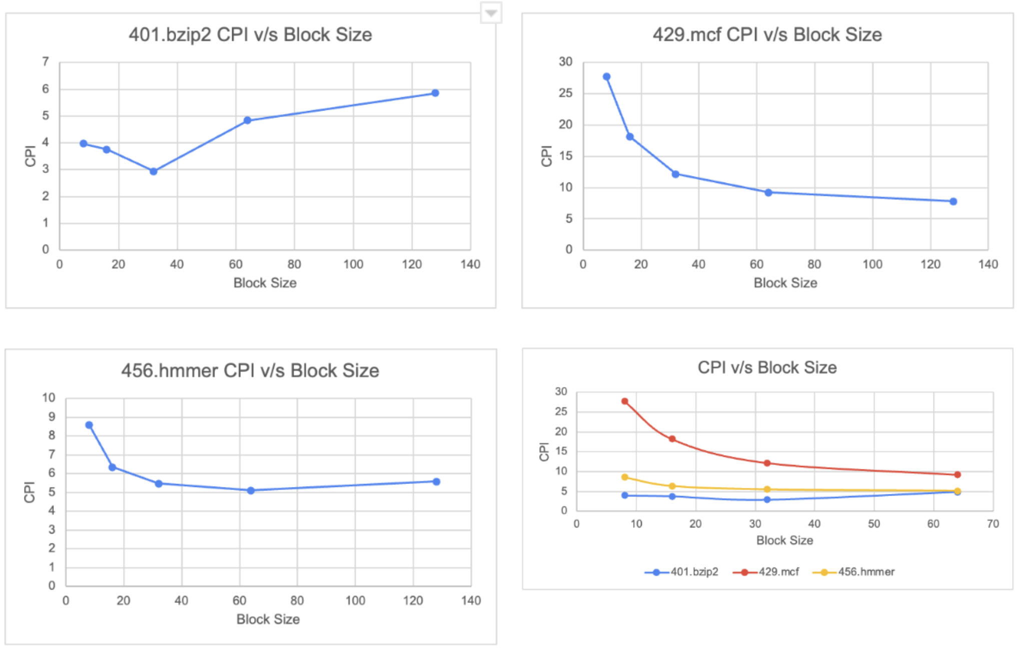


Figure 19: Varied Block Size

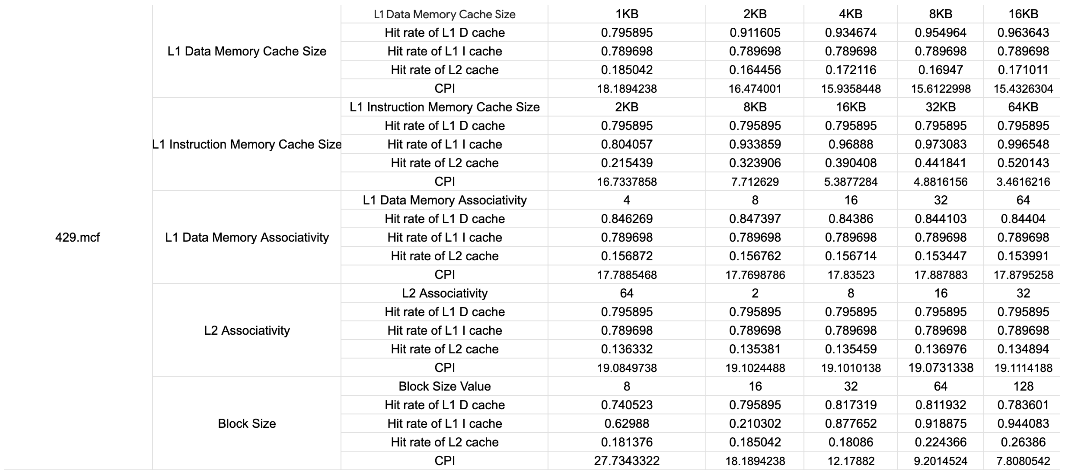
5 Tabulated Results for all test cases

5.1 Test Parameters 401.bzip2:

Assigned Bechmarks	Parameter	Outputs	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5
401.bzip2	L1 Data Memory Cache Size	L1 Data Memory Cache Size Value	1KB	2KB	4KB	8KB	16KB
		Hit rate of L1 D cache	0.923148	0.937903	0.942408	0.946052	0.947245
		Hit rate of L1 I cache	0.990872	0.990872	0.990872	0.990872	0.990872
		Hit rate of L2 cache	0.282022	0.259089	0.254209	0.231831	0.23349
		CPI	3.75E+00	3.381790362	3.258368174	3.20328998	3.162090384
	L1 Instruction Memory Cache Size	L1 Instruction Memory Cache Size	2KB	8KB	16KB	32KB	64KB
		Hit rate of L1 D cache	0.923148	0.923148	0.923148	0.923148	0.923148
		Hit rate of L1 I cache	0.999569	0.999704	0.999863	0.99988	0.999885
		Hit rate of L2 cache	0.141799	0.141924	0.142172	0.142148	0.142163
		CPI	3.662721934	3.653861735	3.643098336	3.642131736	3.641744136
	L1 Data Memory Associativity	L1 Data Memory Associativity	4	8	16	32	64
		Hit rate of L1 D cache	0.936908	0.93712	0.937218	0.937211	0.937203
		Hit rate of L1 I cache	0.990872	0.990872	0.990872	0.990872	0.990872
		Hit rate of L2 cache	0.238499	0.235623	0.235202	0.234857	0.234733
		CPI	3.469322153	3.470799353	3.468931553	3.470133953	3.470698953
	L2 Associativity	L2 Associativity	64	2	8	16	32
		Hit rate of L1 D cache	0.923148	0.923148	0.923148	0.923148	0.923148
		Hit rate of L1 I cache	0.990872	0.990872	0.990872	0.990872	0.990872
		Hit rate of L2 cache	0.354983	0.335694	0.352732	0.354133	0.354689
		CPI	3.512047749	3.575387742	3.519437748	3.514837749	3.513012749
	Block Size	Block Size Value	8	16	32	64	128
		Hit rate of L1 D cache	0.881119	0.923148	0.939055	0.727234	0.696008
		Hit rate of L1 I cache	0.982885	0.990872	0.994882	0.996895	0.996998
		Hit rate of L2 cache	0.359983	0.282022	0.335016	0.727185	0.672193
		CPI	3.972527703	3.751627725	2.936595806	4.835012616	5.859129114

Figure 20: Varied Block Size

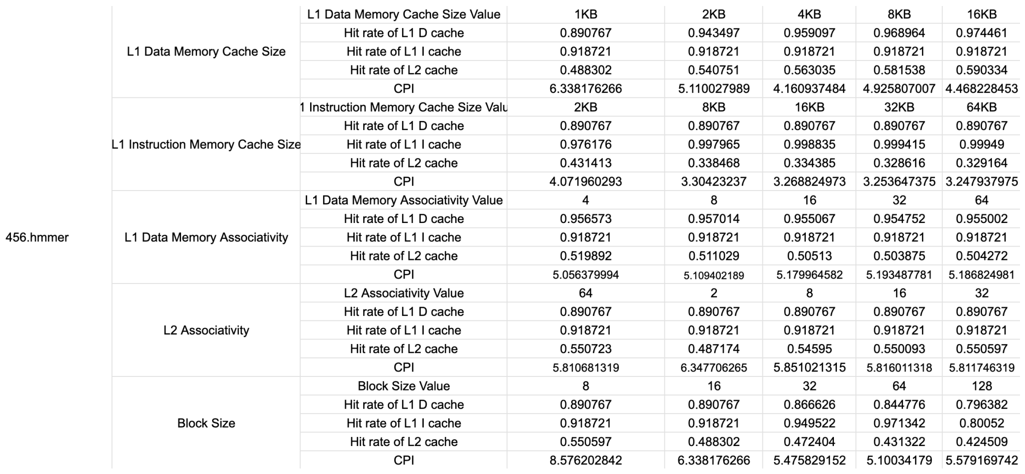
5.2 Test Parameters 429.mcf:



429.mcf	L1 Data Memory Cache Size	L1 Data Memory Cache Size	1KB	2KB	4KB	8KB	16KB
		Hit rate of L1 D cache	0.795895	0.911605	0.934674	0.954964	0.963643
		Hit rate of L1 I cache	0.789698	0.789698	0.789698	0.789698	0.789698
		Hit rate of L2 cache	0.185042	0.164456	0.172116	0.16947	0.171011
		CPI	18.1894238	16.474001	15.9358448	15.6122998	15.4326304
	L1 Instruction Memory Cache Size	L1 Instruction Memory Cache Size	2KB	8KB	16KB	32KB	64KB
		Hit rate of L1 D cache	0.795895	0.795895	0.795895	0.795895	0.795895
		Hit rate of L1 I cache	0.804057	0.933859	0.96888	0.973083	0.996548
		Hit rate of L2 cache	0.215439	0.323906	0.390408	0.441841	0.520143
		CPI	16.7337858	7.712629	5.3877284	4.8816156	3.4616216
	L1 Data Memory Associativity	L1 Data Memory Associativity	4	8	16	32	64
		Hit rate of L1 D cache	0.846269	0.847397	0.84386	0.844103	0.84404
		Hit rate of L1 I cache	0.789698	0.789698	0.789698	0.789698	0.789698
		Hit rate of L2 cache	0.156872	0.156762	0.156714	0.153447	0.153991
		CPI	17.7885468	17.7698786	17.83523	17.887883	17.8795258
	L2 Associativity	L2 Associativity	64	2	8	16	32
		Hit rate of L1 D cache	0.795895	0.795895	0.795895	0.795895	0.795895
		Hit rate of L1 I cache	0.789698	0.789698	0.789698	0.789698	0.789698
		Hit rate of L2 cache	0.136332	0.135381	0.135459	0.136976	0.134894
		CPI	19.0849738	19.1024488	19.1010138	19.0731338	19.1114188
	Block Size	Block Size Value	8	16	32	64	128
		Hit rate of L1 D cache	0.740523	0.795895	0.817319	0.811932	0.783601
		Hit rate of L1 I cache	0.62988	0.210302	0.877652	0.918875	0.944083
		Hit rate of L2 cache	0.181376	0.185042	0.18086	0.224366	0.26386
		CPI	27.7343322	18.1894238	12.17882	9.2014524	7.8080542

Figure 21: Varied Block Size

5.3 Test Parameters 456.hmmer:



456.hmmer	L1 Data Memory Cache Size	L1 Data Memory Cache Size Value	1KB	2KB	4KB	8KB	16KB
		Hit rate of L1 D cache	0.890767	0.943497	0.959097	0.968964	0.974461
		Hit rate of L1 I cache	0.918721	0.918721	0.918721	0.918721	0.918721
		Hit rate of L2 cache	0.488302	0.540751	0.563035	0.581538	0.590334
		CPI	6.338176266	5.110027989	4.160937484	4.925807007	4.468228453
	L1 Instruction Memory Cache Size	1 Instruction Memory Cache Size Valu	2KB	8KB	16KB	32KB	64KB
		Hit rate of L1 D cache	0.890767	0.890767	0.890767	0.890767	0.890767
		Hit rate of L1 I cache	0.976176	0.997965	0.998835	0.999415	0.99949
		Hit rate of L2 cache	0.431413	0.338468	0.334385	0.328616	0.329164
		CPI	4.071960293	3.30423237	3.268824973	3.253647375	3.247937975
	L1 Data Memory Associativity	L1 Data Memory Associativity Value	4	8	16	32	64
		Hit rate of L1 D cache	0.956573	0.957014	0.955067	0.954752	0.955002
		Hit rate of L1 I cache	0.918721	0.918721	0.918721	0.918721	0.918721
		Hit rate of L2 cache	0.519892	0.511029	0.50513	0.503875	0.504272
		CPI	5.056379994	5.109402189	5.179964582	5.193487781	5.186824981
	L2 Associativity	L2 Associativity Value	64	2	8	16	32
		Hit rate of L1 D cache	0.890767	0.890767	0.890767	0.890767	0.890767
		Hit rate of L1 I cache	0.918721	0.918721	0.918721	0.918721	0.918721
		Hit rate of L2 cache	0.550723	0.487174	0.54595	0.550093	0.550597
		CPI	5.810681319	6.347706265	5.851021315	5.816011318	5.811746319
	Block Size	Block Size Value	8	16	32	64	128
		Hit rate of L1 D cache	0.890767	0.890767	0.866626	0.844776	0.796382
		Hit rate of L1 I cache	0.918721	0.918721	0.949522	0.971342	0.80052
		Hit rate of L2 cache	0.550597	0.488302	0.472404	0.431322	0.424509
		CPI	8.576202842	6.338176266	5.475829152	5.10034179	5.579169742

Figure 22: Varied Block Size

6 Conclusion

• The CPI is proportional to the miss rate, which means that if the miss rate goes up or down, the CPI will also go up or down in the same amount.
• As the cache size grows, the number of requests that are successful goes up and the number of requests that are missed goes down.
• The hit rate goes up in a straight line with how well the cache works.
• Changing the amount or associativity of the L2 cache has no effect on how often the L1D and L1I caches get hits or misses.

7 Work Distribution

Although each of us made an equal contribution to the project and report, the following sections were handled independently:
Aakash Baruva: Did simulation for the benchmark 456.hmmer with all parameters
Harshavardhan Sivadanam: Did simulation for the benchmark 429.mcf with all parameters
Priyanka Chakraborty: Did simulation for the benchmark 401.bzip2 with all parameters
Sai Sree Depa: Did the analysis, simulation and generated graphs comparing cpi with all the benchmarks and parameters

References

[1] Took this pdf as reference CSE 490/590, Project 2, Spring 2023, "Analysis Of Cache Parameters And Trade-Offs Using Gem5".

[2] Cache Memory in Computer Organization, Geeks for geeks