CSE 590 Computer Architecture
Project 2 Final Report Analysis Of Cache parameters and trade-offs using Gem5

Name	UBIT NAME
Sai mitra kamasani	saimitra
Indra teja pidathala	indratej
Sai Kiran Paturi	spaturi2
Chitrak Vimalbhai Dave	chitrakv

Contents

1 Problem Statement

In this project, we are required to tune various cache parameters and analyze performance in each case Using Gem5 simulator. We should explore the impact of changing cache parameters on the processor's performance by executing a series of experiments, analyzing the results, and creating visualizations to show trends.

2 Gem5

Gem5 is an open-source computer architecture simulator that provides a modular platform for computer-system architecture research, encompassing both system-level and processor microarchitecture aspects. This versatile simulator allows researchers to study various design decisions, cache hierarchies, and performance trade-offs, making it an invaluable tool in the field of computer architecture.

3 Project Execution

There are two shell scripts available to facilitate the execution of a program, with one of them being gemTest.csh. This script streamlines the process of conducting CPU benchmarks using Gem5, while also generating output files in a folder named "stat_files." This folder comprises all the stats files produced from each Gem5 simulation that was performed. The script can be executed using the command:

csh gemTest.csh <benchmark name> <parameter name> <L1 Data Cache Size> <L1 Instruction Cache Size> <L2 Cache Size> <L1 Data Associativity> <L1 Instruction Cache Associativity> <L2 Cache Associativity> <Block Size>

Another shell script known as getData.csh is available to extract critical statistics from the stats file of a previously executed test. The script is designed to retrieve vital metrics such as overall misses for L1 and L2 cache, miss rate, and the total number of instructions. The usage of this script is specified accordingly. The script can be executed using the command:

`csh getData.csh <benchmark name> <parameter name>`

As part of our analysis, we modified the cache parameters in the initial command and subsequently evaluated the outcomes using the second command. Specifically, we conducted a total of 75 commands. We conducted 25 commands for 401.bzip2 benchmark, 25 commands for 429.mcf benchmark and 25 commands for 458.sjeng benchmark by altering one of five cache parameters at a time, each with five distinct values for the same parameter. We maintained a consistent setting for the remaining parameters throughout each experiment.

4 CPU Benchmarks

CPU benchmarks help us measure the performance of a CPU, which is the major component in a computer system. There are many different types of CPU benchmarks that test the performance of a CPU. Some of the benchmarks are:

1. 401.bzip2
2. 429.mcf
3. 456.hmmer
4. 458.sjeng
5. 470.lbm

In this project, we performed three CPU benchmarks, which are:

1. **401.bzip2:** The 401.bzip benchmark is made to see how well a CPU can decompress and compress data. It calculates how long it takes the CPU to use the bzip2 method to compress and decompress a group of files. This benchmark is designed to assess how well CPUs perform in tasks like file archiving and data storage.
2. **429.mcf:**

   A CPU's capacity to resolve a vehicle routing problem is tested using this benchmark. It mimics a delivery van navigating a collection of sites in order to reduce overall journey time. This benchmark is intended to assess how well CPUs perform in applications like transportation and logistics.
3. **458.sjeng:**

   This benchmark is made to evaluate a CPU's chess-playing capabilities. It calculates how long it takes the CPU to execute a certain sequence of moves in a chess match using the Sjeng chess engine. This benchmark is designed to assess how well CPUs perform in tasks like gaming and artificial intelligence.

5 Cache Parameters

The cache parameters are adjustable characteristics that can be modified to achieve different performance profiles and behaviors. These parameters are responsible for controlling the cache's functionality and effectiveness in storing and retrieving data. There are several cache parameters that can be fine-tuned to optimize its performance. By adjusting these cache parameters, one can determine the cache's behavior to meet specific performance requirements for different applications and workloads. There are different cache parameters like

(a) L1 Data Cache size

(b) L1 Instruction Cache Size

(c) L2 Cache Size

(d) L1 Data Cache associativity

(e) L1 Instruction Cache associativity

(f) L2 Cache associativity

(g) Block size

We were assigned 5 parameters L1 Instruction Cache Size,L1 Instruction Cache Associativity, L2 Cache Size,L2 cache Associativity, and Block Size for the analysis of processor performance.

5.1 L1 Cache

The microprocessor has an internal cache called L1 cache that stores recently accessed data which includes two caches: L1 Instruction Cache and L1 Data Cache. By having separate and dedicated caches for instructions and data, processor efficiency can be improved. Using multiple L1 caches simultaneously improves bandwidth. However, due to their smaller sizes, each cache has a higher miss rate. L1 cache is the fastest cache available but it has a smaller capacity because it is integrated into the microprocessor. The primary objective of the L1 cache is to minimize hit time to support a shorter clock cycle.

We are changing the L1 Instruction Cache size from 2kB to 32 kB that is 2kB, 4kB, 8kB, 16kB, and 32kB because we were assigned it to change and observe the performance of the processor.

5.2 L2 Cache

The L2 cache is a type of CPU cache memory that is located outside of the core of the microprocessor chip despite being on the same packaging. If the L1 cache is unable to locate the data, the L2 cache is accessed instead. The L2 cache is larger than the L1 cache but operates at a slower speed. Incorporating an L2 cache significantly decreases the L1 cache's miss penalty enabling a smaller and faster cache with a higher miss rate. The primary goal of the L2 cache is to lower the miss rate and minimize the time needed to access the main memory.

We are changing the L2 Cache size from 1kB to 16kB that is 1kB,2kB, 4kB, 8kB, and 16kB because we were assigned it to change and observe the performance of the processor.

5.3 Cache Associativity

A cache can be organized in either a direct-mapped or n-way set associative manner. In a direct-mapped cache, each memory block corresponds to a single cache block. Conversely, a fully associative cache can map any cache block to any memory block. When the value of n is equal to the cache's number of entries, the cache is considered fully associative. However, to overcome the limitation of associativity, the cache can be divided into sets where each set comprises n ways resulting in an n-way set associative cache. In this arrangement, each set has n entries and all entries in a set are searched simultaneously.

We are changing the L1 Instruction Cache Associativity from 2 to 32 that 2,4,8,16, and 32. And We are changing the L2 Cache Associativity from 2to 32 that 2,4,8,16, and 32 because we were assigned them to change and observe the performance of the processor.

5.4 Block Size

When the size of a block in a cache becomes a significant proportion of the cache size, the number of blocks that can be stored in the cache decreases, resulting in a higher miss rate. Although larger block sizes should theoretically reduce miss rates due to spatial locality, a fixed-sized cache with larger blocks results in fewer blocks and increased competition among them, ultimately increasing the miss rate. Moreover, larger block sizes may cause a higher miss penalty, which could offset the benefits of having a lower miss rate. When dealing with exceptionally large blocks, the issue of pollution can also arise, resulting in a further increase in the miss rate.

We are changing the Block Size from 32 to 512 that 32,64,128,256, and 512 because we were assigned it to change and observe the performance of the processor.

6 Results

The following output parameters/KPIs has been calculated for each test experiment(bench mark) against each input(parameter).

L1-D HIT RATE $= 1 - (L1 - D - MISSRATE)$

L1-I HIT RATE $= 1 - (L1 - I - MISSRATE)$

L2 HIT RATE $= 1 - (L2MISS - RATE)$

CPI $= \frac{(L1-IMISSES+L1-DMISSES)\times 6+L2MISSES\times 50}{INSTRUCTIONS}$

	Test Bench Mark	L1-I SIZE	L1-D HIT-RATE	L1-I HIT-RATE	L2 HIT-RATE	CPI
0	401.bzip	2	0.923148	0.999569	0.141799	3.662722
1	401.bzip	4	0.923148	0.999687	0.141978	3.654771
2	401.bzip	8	0.923148	0.999704	0.141924	3.653862
3	401.bzip	16	0.923148	0.999863	0.142172	3.643098
4	401.bzip	32	0.923148	0.999880	0.142148	3.642132
5	429.mcf	2	0.795895	0.804057	0.215439	16.733786
6	429.mcf	4	0.795895	0.884328	0.278898	10.968484
7	429.mcf	8	0.795895	0.933859	0.323906	7.712629
8	429.mcf	16	0.795895	0.968880	0.390408	5.387728
9	429.mcf	32	0.795895	0.973083	0.441841	4.881616
10	458.sjeng	2	0.585568	0.999894	0.008647	18.857287
11	458.sjeng	4	0.585568	0.999901	0.008652	18.856703
12	458.sjeng	8	0.585568	0.999905	0.008646	18.856490
13	458.sjeng	16	0.585568	0.999916	0.008617	18.856114
14	458.sjeng	32	0.585568	0.999917	0.008617	18.856057

Figure 1: param1-outputs-all-benchmarks

The results for all the experiments are plotted in graphs and are displayed below.

7 Analyses

7.1 param1- L1-I SIZE

the relevant figures for this experiment are 6,6,6,6, 6

7.1.1 L1-D HIT RATE

Expectation : L1-I SIZE does not have a direct impact on L1-D HIT RATE. L1-I cache is responsible for storing instruction data, while L1-D cache is responsible for storing data accessed by the CPU. Therefore, changing the size of the L1-I cache should not directly affect the HIT RATE of the L1-D cache. **ALL BENCH MARKS → NO VARIATION , NO IMPACT**

Observation :

- BenchMark-1(401.bzip2) : TRUE, NO IMPACT, NO VARIATION(0.92)

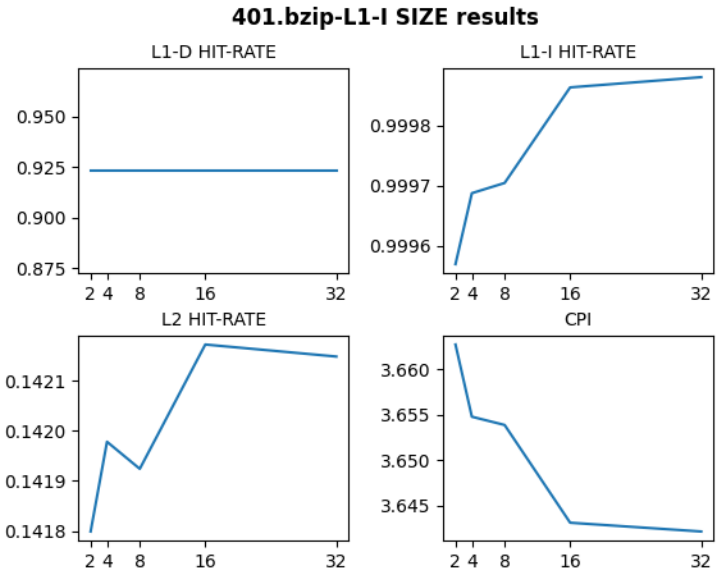

Figure 2: param1-benchmark1

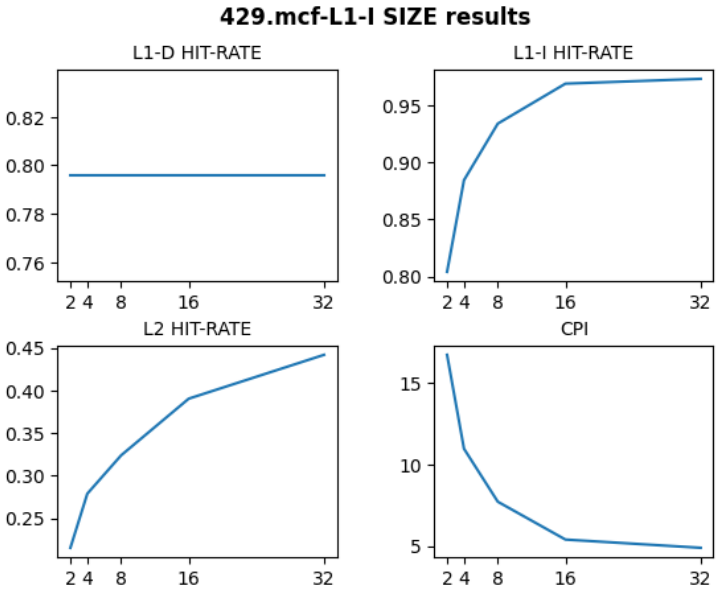

Figure 3: param1-benchmark2

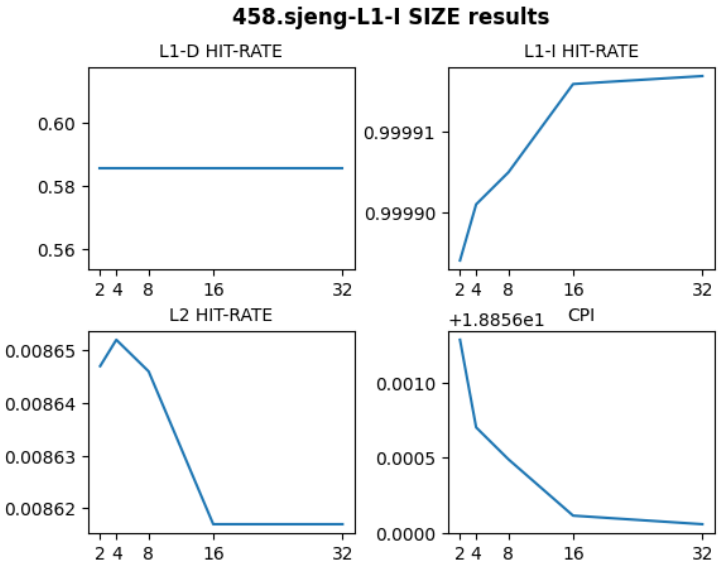

Figure 4: param1-benchmark3

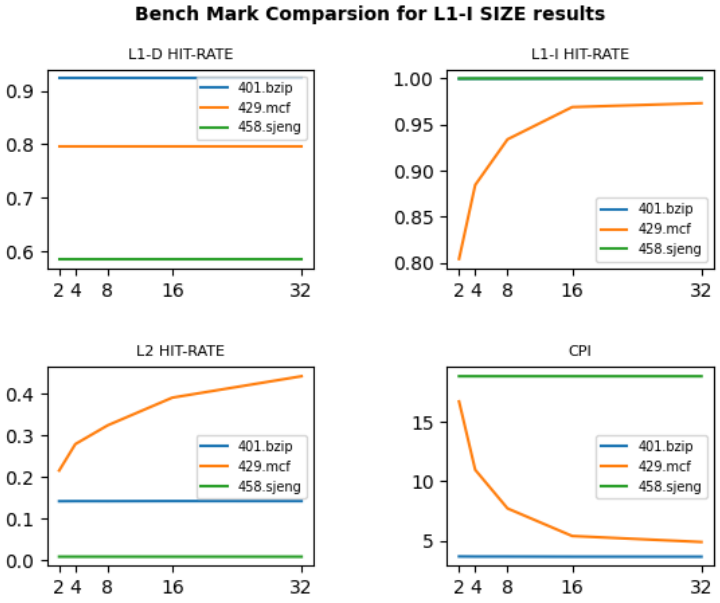

Figure 5: param1-All BenchMarks

	Test Bench Mark	L1-I Associativity	L1-D HIT-RATE	L1-I HIT-RATE	L2 HIT-RATE	CPI
0	401.bzip	2	0.923148	0.999294	0.144925	3.671539
1	401.bzip	4	0.923148	0.999441	0.142210	3.669650
2	401.bzip	8	0.923148	0.999437	0.142295	3.669688
3	401.bzip	16	0.923148	0.999455	0.141865	3.669712
4	401.bzip	32	0.923148	0.999442	0.142135	3.669797
5	429.mcf	2	0.795895	0.805386	0.137228	18.004016
6	429.mcf	4	0.795895	0.803242	0.099019	18.816263
7	429.mcf	8	0.795895	0.804178	0.075722	19.155318
8	429.mcf	16	0.795895	0.804550	0.066443	19.289519
9	429.mcf	32	0.795895	0.804551	0.062422	19.359263
10	458.sjeng	2	0.585568	0.999888	0.008641	18.857887
11	458.sjeng	4	0.585568	0.999893	0.008637	18.857581
12	458.sjeng	8	0.585568	0.999888	0.008663	18.857520
13	458.sjeng	16	0.585568	0.999887	0.008667	18.857505
14	458.sjeng	32	0.585568	0.999887	0.008665	18.857535

Figure 6: param2-outputs-all-benchmarks

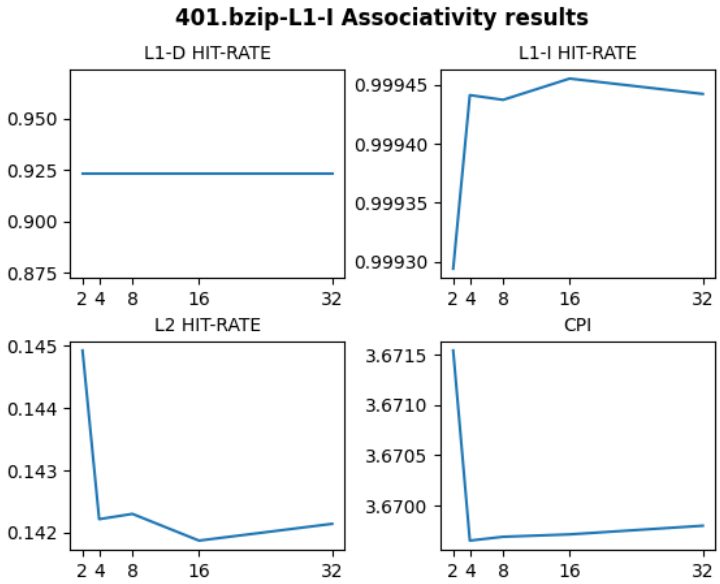

Figure 7: param2-benchmark1

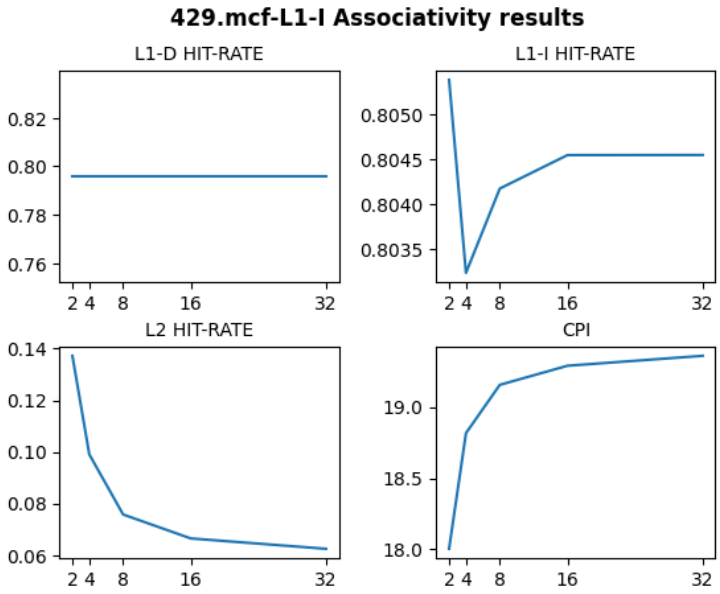

Figure 8: param2-benchmark2

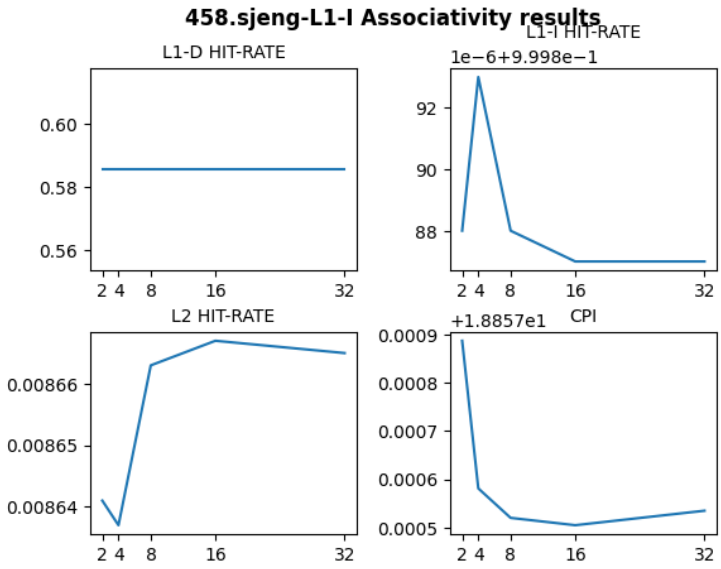

Figure 9: param2-benchmark3

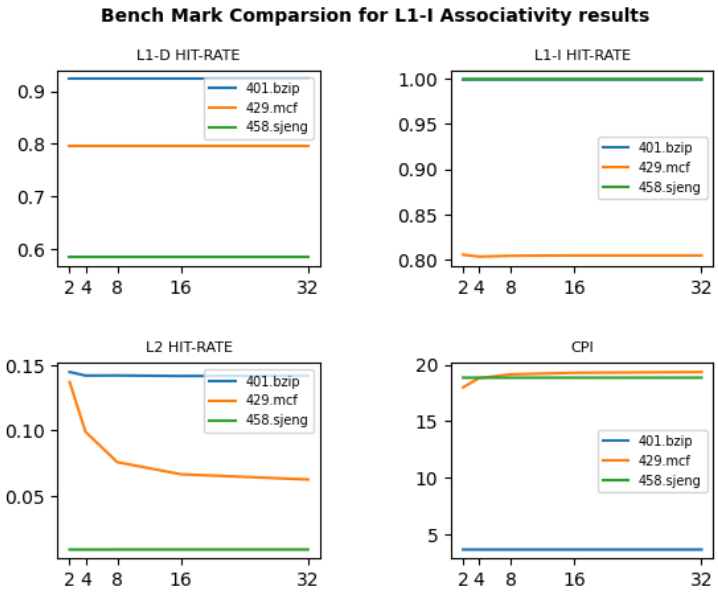

Figure 10: param2-All BenchMarks

	Test Bench Mark	L2 SIZE	L1-D HIT-RATE	L1-I HIT-RATE	L2 HIT-RATE	CPI
0	401.bzip	1	0.923148	0.990872	0.282022	3.751628
1	401.bzip	2	0.923148	0.990872	0.370495	3.461113
2	401.bzip	4	0.923148	0.990872	0.405573	3.345928
3	401.bzip	8	0.923148	0.990872	0.420675	3.296338
4	401.bzip	16	0.923148	0.990872	0.434145	3.252108
5	429.mcf	1	0.795895	0.789698	0.185042	18.189424
6	429.mcf	2	0.795895	0.789698	0.223695	17.478784
7	429.mcf	4	0.795895	0.789698	0.410752	14.039689
8	429.mcf	8	0.795895	0.789698	0.655854	9.533439
9	429.mcf	16	0.795895	0.789698	0.818610	6.541124
10	458.sjeng	1	0.585568	0.999885	0.008657	18.857799
11	458.sjeng	2	0.585568	0.999885	0.012377	18.798014
12	458.sjeng	4	0.585568	0.999885	0.018923	18.692824
13	458.sjeng	8	0.585568	0.999885	0.021802	18.646574
14	458.sjeng	16	0.585568	0.999885	0.022640	18.633109

Figure 11: param3-outputs-all-benchmarks

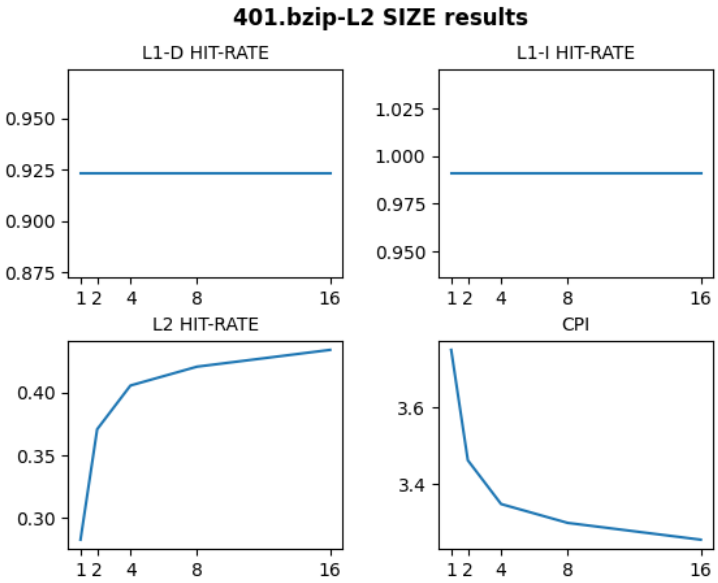

Figure 12: param3-benchmark1

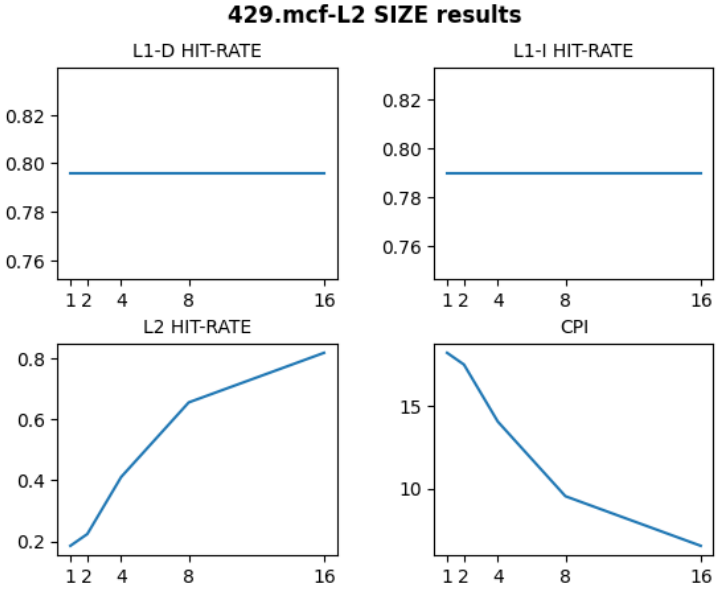

Figure 13: param3-benchmark2

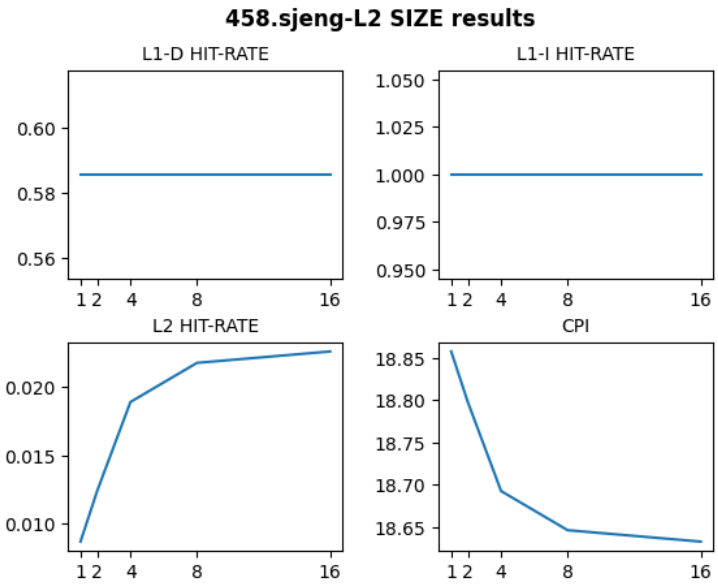

Figure 14: param3-benchmark3

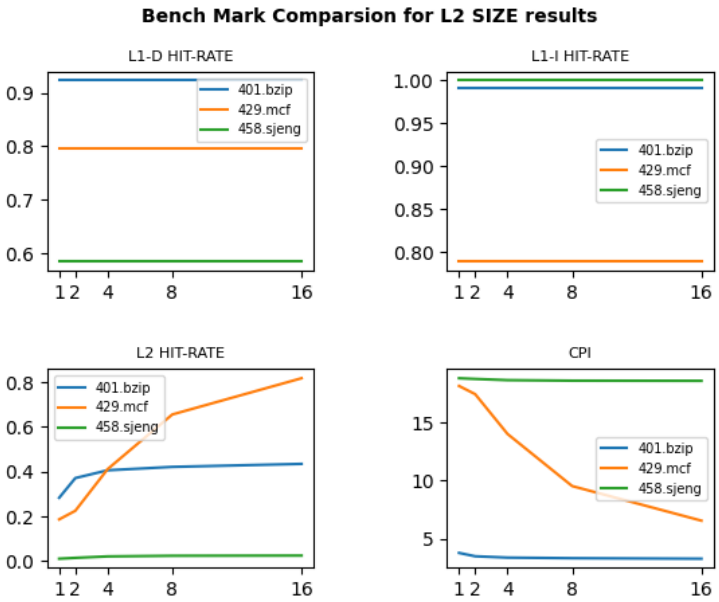

Figure 15: param3-All BenchMarks

	Test Bench Mark	L2 Associativity	L1-D HIT-RATE	L1-I HIT-RATE	L2 HIT-RATE	CPI
0	401.bzip	2	0.923148	0.990872	0.335694	3.575388
1	401.bzip	4	0.923148	0.990872	0.348833	3.532243
2	401.bzip	8	0.923148	0.990872	0.352732	3.519438
3	401.bzip	16	0.923148	0.990872	0.354133	3.514838
4	401.bzip	32	0.923148	0.990872	0.354689	3.513013
5	429.mcf	2	0.795895	0.789698	0.135381	19.102449
6	429.mcf	4	0.795895	0.789698	0.134086	19.126274
7	429.mcf	8	0.795895	0.789698	0.135459	19.101014
8	429.mcf	16	0.795895	0.789698	0.136976	19.073134
9	429.mcf	32	0.795895	0.789698	0.134894	19.111419
10	458.sjeng	2	0.585568	0.999885	0.018507	18.699524
11	458.sjeng	4	0.585568	0.999885	0.019108	18.689854
12	458.sjeng	8	0.585568	0.999885	0.019395	18.685244
13	458.sjeng	16	0.585568	0.999885	0.019385	18.685409
14	458.sjeng	32	0.585568	0.999885	0.019406	18.685069

Figure 16: param4-outputs-all-benchmarks

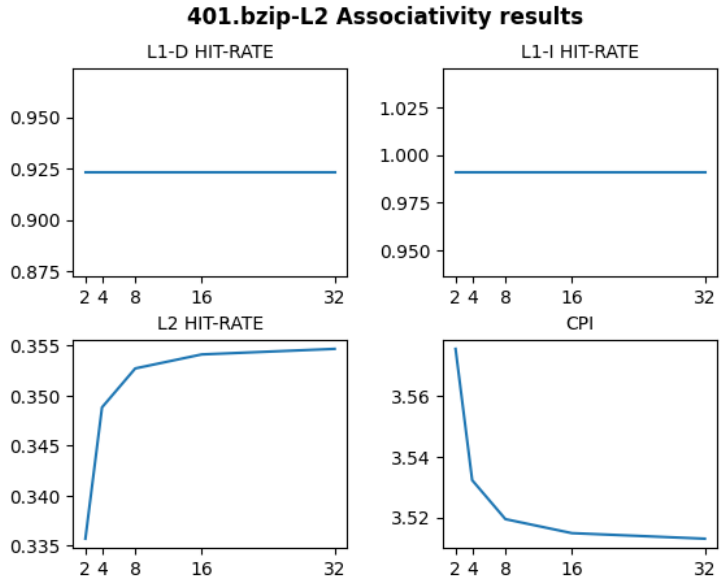

Figure 17: param4-benchmark1

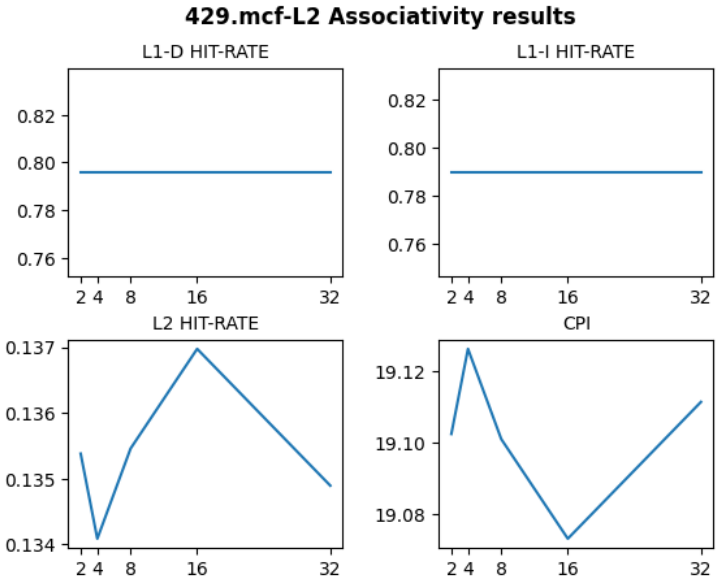

Figure 18: param4-benchmark2

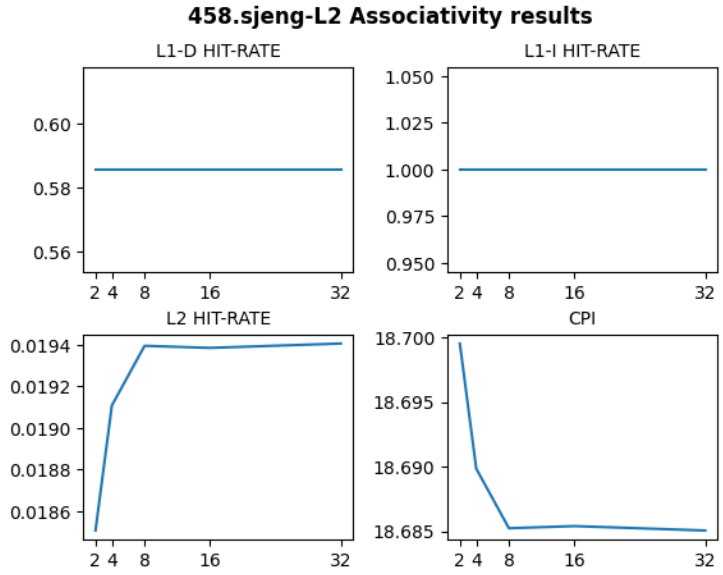

Figure 19: param5-benchmark1

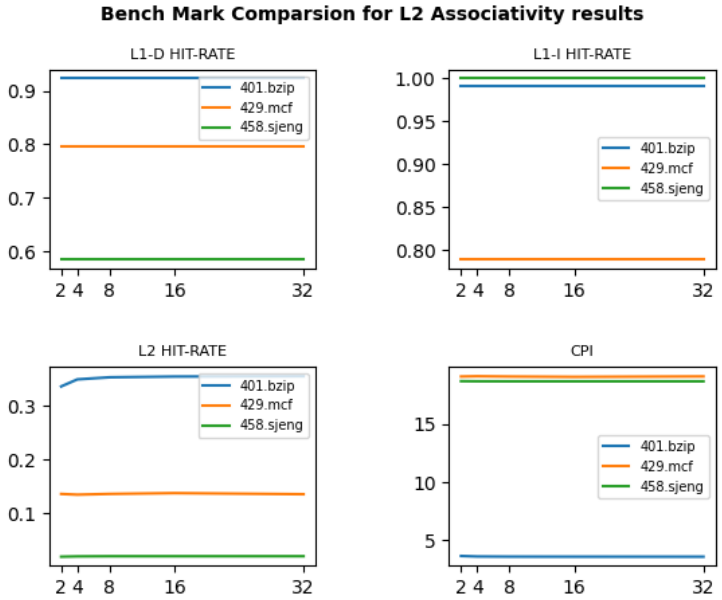

Figure 20: param4-All BenchMarks

Test Bench Mark	BLOCK SIZE	L1-D HIT-RATE	L1-I HIT-RATE	L2 HIT-RATE	CPI
401.bzip	32	0.939055	0.994882	0.335016	2.936596
401.bzip	64	0.727234	0.996895	0.727185	4.835013
401.bzip	128	0.696008	0.996998	0.672193	5.859129
401.bzip	256	0.728351	0.997997	0.767026	4.407183
401.bzip	512	0.629499	0.998012	0.653692	7.116593
429.mcf	32	0.817319	0.877652	0.180860	12.178820
429.mcf	64	0.811932	0.918875	0.224366	9.201452
429.mcf	128	0.783601	0.944083	0.263860	7.808054
429.mcf	256	0.744790	0.955089	0.274702	7.699568
429.mcf	512	0.965658	0.965516	0.219762	8.988029
458.sjeng	32	0.781553	0.999930	0.028978	10.241443
458.sjeng	64	0.869264	0.999953	0.083196	6.256306
458.sjeng	128	0.902417	0.999965	0.151400	4.665294
458.sjeng	256	0.901067	0.999973	0.200930	4.525464
458.sjeng	512	0.877272	0.999974	0.211304	5.323540

Figure 21: param5-outputs-all-benchmarks

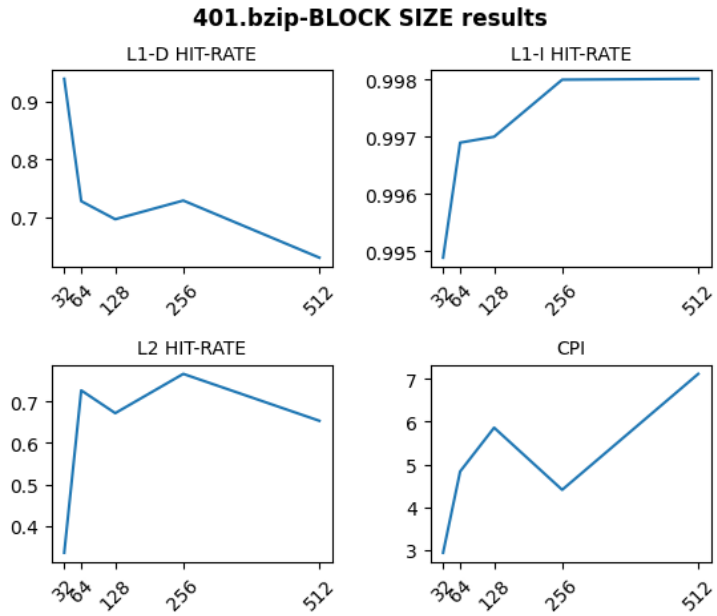

Figure 22: param5-benchmark1

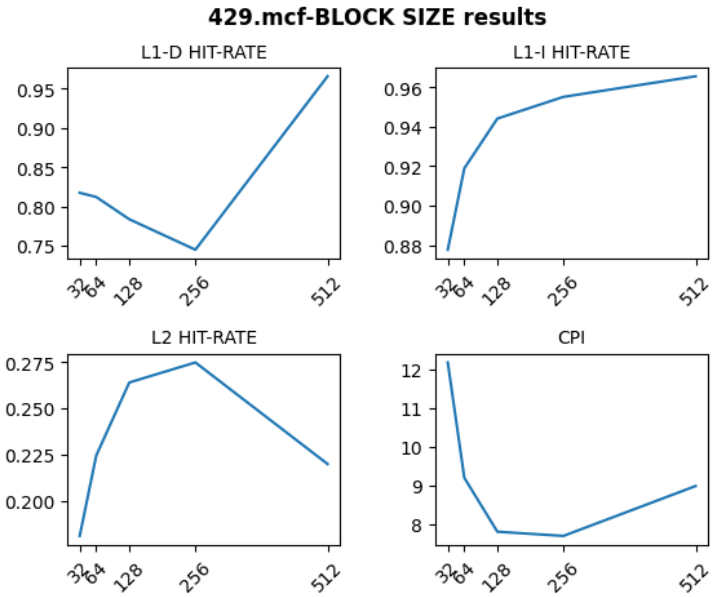

Figure 23: param5-benchmark2

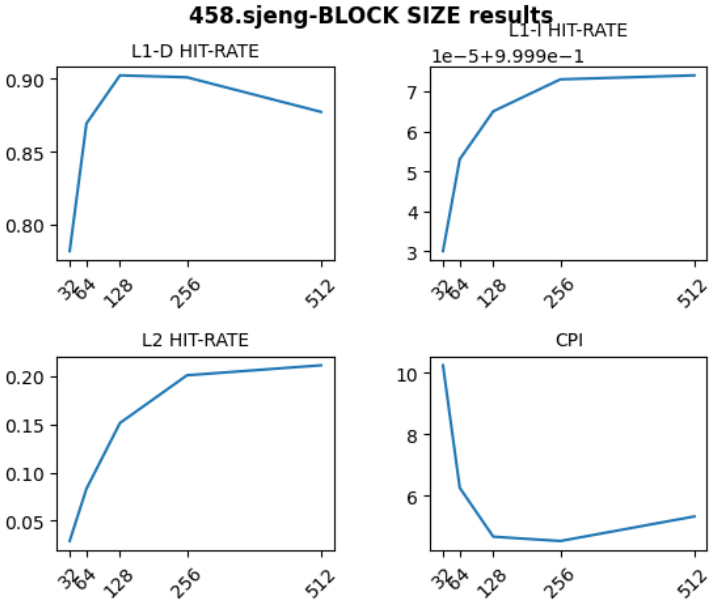

Figure 24: param5-benchmark3

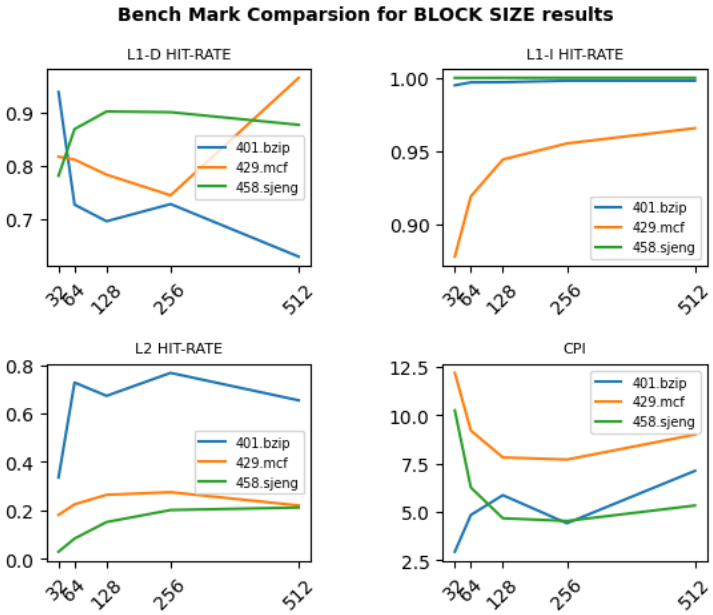

Figure 25: param5-All BenchMarks

- BenchMark-2(429.mcf) : TRUE, NO IMPACT, NO VARIATION(0.79)
- BenchMark-3(458.sjeng) : TRUE, NO IMPACT, NO VARIATION(0.58)

7.1.2 L1-I HIT RATE

Expectation : Increasing L1-I SIZE can potentially increase L1-I HIT RATE, as there is more space available to store instructions, and the probability of finding an instruction in the cache increases. When the L1-I cache has sufficient capacity to hold a larger portion of the working set of instructions, it can reduce the number of L1-I MISSES and increase the L1-I HIT RATE.

Observation :

- BenchMark-1(401.bzip2) : TRUE, increasing by a very small amount(0.0001).
- BenchMark-2(429.mcf) : TRUE, significant increase observed(0.80 to 0.99).
- BenchMark-3(458.sjeng) : TRUE, increasing by a very small amount(0.000017).

7.1.3 L2 HIT RATE

Expectation : L1-I SIZE can indirectly impact L2 HIT RATE through its impact on L1-I HIT RATE and L1-D MISS rate.

Increasing L1-I SIZE can potentially increase L1-I HIT RATE, which reduces the number of accesses to L2 cache for instruction data. When the L1-I cache can hold a larger portion of the working set of instructions, it can reduce the number of L1-I MISSES and increase the L1-I HIT RATE, which leads to fewer L2 cache accesses for instruction data. As a result, the L2 cache can be less congested, and L2 HIT RATE may increase.

Observation :

- BenchMark-1(401.bzip2) : TRUE, increasing by a very small amount BUT NOT MUCH(0.00001).
- BenchMark-2(429.mcf) : TRUE, significant increase observed(0.21 to 0.44).
- BenchMark-3(458.sjeng) : NO VARIATION(0.9999)

7.1.4 CPI

Expectation : Increasing L1-I SIZE can potentially decrease CPI by reducing the L1-I MISS rate, as a larger cache can hold a larger portion of the working set of instructions, which reduces the number of instructions that need to be fetched from the main memory. When there are fewer L1-I MISSes, the processor can fetch and execute instructions faster, which can decrease CPI. However, this reduction in CPI may be offset by the increase in cache access latency, which may occur as the cache becomes larger.

Observation :

- BenchMark-1(401.bzip2) : TRUE, small decrease from (3.67 to 3.64) NO IMPACT
- BenchMark-2(429.mcf) : TRUE, significant decrease observed(16.7 to 4.8).
- BenchMark-3(458.sjeng) : NO IMPACT (18.55)

Overall, from figure 6, we can see that this parameter - L1-I SIZE is making an impact only on benchmark-2(429.mcf). other benchmarks not showing much variation in any output KPI.

7.2 param2- L1-I Associativity

the relevant figures for this experiment are 6,6,6,6, 6

7.2.1 L1-D HIT RATE

Expectation : L1-I Associativity does not have a direct impact on L1-D HIT RATE. L1-I cache and L1-D cache are two separate caches designed for different purposes, and their associativity does not directly affect each other. **ALL BENCH MARKS → NO VARIATION , NO IMPACT**

Observation :

- BenchMark-1(401.bzip2) : TRUE, NO IMPACT, NO VARIATION(0.92)
- BenchMark-2(429.mcf) : TRUE, NO IMPACT, NO VARIATION(0.79)
- BenchMark-3(458.sjeng) : TRUE, NO IMPACT, NO VARIATION(0.58)

7.2.2 L1-I HIT RATE

Expectation : Increasing L1-I Associativity can potentially increase L1-I HIT RATE by allowing the cache to hold more instructions, which reduces the probability of a cache miss. When the L1-I cache has a higher associativity, it can map the incoming instructions to a larger number of cache sets, reducing the likelihood of a collision, and increasing the probability that a given instruction is already in the cache. As a result, more instructions can be fetched and executed from the L1-I cache, which improves L1-I HIT RATE

Observation :

- BenchMark-1(401.bzip2) : small increase(0.9993 to 0.994)
- BenchMark-2(429.mcf) : NO IMPACT , NOT MUCH VARIATION(0.80)
- BenchMark-3(458.sjeng) : . NO IMPACT , NOT MUCH VARIATION(0.99)

7.2.3 L2 HIT RATE

Expectation : Increasing L1-I Associativity can potentially increase L2 HIT RATE by reducing the L1-I MISS rate. When the L1-I cache has higher associativity, it can reduce the probability of collisions and increase the probability that a given instruction is already present in the cache. This improves the L1-I HIT RATE, and hence, fewer instructions need to be fetched from L2 cache or main memory, reducing the L2 cache's demand for memory bandwidth. As a result, the L2 cache may experience fewer cache misses and achieve a higher HIT RATE.

Observation :

- BenchMark-1(401.bzip2) : NO IMPACT , NOT MUCH VARIATION(0.14)
- BenchMark-2(429.mcf) : small decrease(0.14 to 0.06)
- BenchMark-3(458.sjeng) : . NO IMPACT , NOT MUCH VARIATION(0.008)

7.2.4 CPI

Expectation : Increasing L1-I Associativity can potentially decrease CPI by reducing the L1-I MISS rate. When the L1-I cache has a higher associativity, it can reduce the probability of collisions and increase the probability that a given instruction is already present in the cache. This improves the L1-I HIT RATE, and hence, fewer instructions need to be

fetched from L2 cache or main memory, reducing the L1-I MISS rate and the associated cache miss penalty. This, in turn, reduces the overall memory access latency and improves the CPI.

Observation :

- BenchMark-1(401.bzip2) : small decrease(3.67 to 3.66)
- BenchMark-2(429.mcf) : small increase(18.0 to 18.19.3)
- BenchMark-3(458.sjeng) : . NO IMPACT , NO VARIATION(18.8)

Overall, from figure 6, we can see that this parameter - 'L1-I Associativity' is not making an impact only on any benchmark.

7.3 param3- L2 SIZE

the relevant figures for this experiment are 6,6,6,6, 6

7.3.1 L1-D HIT RATE

Expectation : When the L2 cache is larger, it can potentially hold more data and reduce the number of misses from main memory. However L2 SIZE does not have a direct impact on L1-D HIT RATE . **ALL BENCH MARKS → NO VARIATION , NO IMPACT**

Observation :

- BenchMark-1(401.bzip2) : TRUE, NO IMPACT, NO VARIATION (0.92)
- BenchMark-2(429.mcf) : TRUE, NO IMPACT, NO VARIATION (0.79)
- BenchMark-3(458.sjeng) : TRUE, NO IMPACT, NO VARIATION (0.58)

7.3.2 L1-I HIT RATE

Expectation : When the L2 cache is larger, it can potentially hold more data and reduce the number of misses from main memory. However L2 SIZE does not have a direct impact on L1-I HIT RATE . **ALL BENCH MARKS → NO VARIATION , NO IMPACT**

Observation :

- BenchMark-1(401.bzip2) : TRUE, NO IMPACT, NO VARIATION (0.99)
- BenchMark-2(429.mcf) : TRUE, NO IMPACT, NO VARIATION (0.78)
- BenchMark-3(458.sjeng) : TRUE, NO IMPACT, NO VARIATION (0.99)

7.3.3 L2 HIT RATE

Expectation : Increasing L2 SIZE can potentially increase L2 HIT RATE by reducing the probability of conflict and capacity misses. When the L2 cache is larger, it can hold more data and reduce the number of misses from main memory. This can increase the probability of finding the requested data in the L2 cache and improve the L2 HIT RATE.

Observation :

- BenchMark-1(401.bzip2) : TRUE , Significant increase from (0.2 to 0.43)
- BenchMark-2(429.mcf) : TRUE , Significant increase from (0.1 to 0.8)
- BenchMark-3(458.sjeng) : TRUE , small increase from (0.008 to 0.022)

7.3.4 CPI

Expectation : Increasing L2 SIZE can potentially decrease CPI by reducing the overall memory access latency of the system. This is because a larger L2 cache can hold more data and reduce the number of misses from main memory, thereby reducing the memory access latency and improving the overall performance of the system. Moreover, a larger L2 cache can also reduce the contention for memory bandwidth and improve the effective bandwidth of the memory system, reducing the overall memory access latency.

Observation :

- BenchMark-1(401.bzip2) : TRUE , small decrease from (3.75 to 3.25)
- BenchMark-2(429.mcf) : TRUE , Significant decrease from (18.2 to 6.5)
- BenchMark-3(458.sjeng) : TRUE , small decrease from (18.8 to 18.6)

Overall, from figure 6, we can see that this parameter - 'L2 SIZE' with larger size is making a significant impact on CPI only on benchmark-2(429.mcf).

7.4 param4- L2 Associativity

the relevant figures for this experiment are 6,17,6,**??**, 6

7.4.1 L1-D HIT RATE

Expectation : Increasing L2 Associativity can potentially increase L2 cache capacity. When the L2 cache is larger, it can potentially hold more data and reduce the number of misses from main memory. However L2 SIZE does not have a direct impact on L1-D HIT RATE . **ALL BENCH MARKS → NO VARIATION , NO IMPACT**

Observation :

- BenchMark-1(401.bzip2) : TRUE, NO IMPACT, NO VARIATION (0.92)
- BenchMark-2(429.mcf) : TRUE, NO IMPACT, NO VARIATION (0.79)
- BenchMark-3(458.sjeng) : TRUE, NO IMPACT, NO VARIATION (0.58)

7.4.2 L1-I HIT RATE

Expectation : Increasing L2 Associativity can potentially increase L2 cache capacity. When the L2 cache is larger, it can potentially hold more data and reduce the number of misses from main memory. However L2 SIZE does not have a direct impact on L1-I HIT RATE . **ALL BENCH MARKS → NO VARIATION , NO IMPACT**

Observation :

- BenchMark-1(401.bzip2) : TRUE, NO IMPACT, NO VARIATION (0.99)
- BenchMark-2(429.mcf) : TRUE, NO IMPACT, NO VARIATION (0.78)
- BenchMark-3(458.sjeng) : TRUE, NO IMPACT, NO VARIATION (0.99)

7.4.3 L2 HIT RATE

Expectation : Increasing L2 Associativity can potentially increase L2 HIT RATE by reducing the probability of conflict and capacity misses. When the L2 cache is larger, it can hold more data and reduce the number of misses from main memory. This can increase the probability of finding the requested data in the L2 cache and improve the L2 HIT RATE.

Observation :

- BenchMark-1(401.bzip2) : TRUE , small increase from (0.33 to 0.35)
- BenchMark-2(429.mcf) : NOT MUCH VARIATION(0.13)
- BenchMark-3(458.sjeng) : TRUE , small increase from (0.018 to 0.02)

7.4.4 CPI

Expectation : Increasing L2 Associativity can potentially decrease CPI by reducing the overall memory access latency of the system. This is because a larger L2 cache can hold more data and reduce the number of misses from main memory, thereby reducing the memory access latency and improving the overall performance of the system. Moreover, a larger L2 cache can also reduce the contention for memory bandwidth and improve the effective bandwidth of the memory system, reducing the overall memory access latency.

Observation :

- BenchMark-1(401.bzip2) : TRUE , small decrease from (3.57 to 3.51)
- BenchMark-2(429.mcf) : TRUE , small decrease from (19.10 to 19.07)
- BenchMark-3(458.sjeng) : TRUE , small decrease from (18.7 to 18.6)

Overall, from figure 6, we can see that this parameter - 'L2 Associativity' with larger size is not making a significant impact on CPI on any benchmark.

7.5 param5- BLOCK SIZE

the relevant figures for this experiment are 6,22,6,6, 6

7.5.1 L1-D HIT RATE

Expectation : Increasing the BLOCK SIZE parameter can potentially increase the spatial locality of memory accesses, which can improve L1-D HIT RATE by reducing the number of cache misses caused by the lack of spatial locality. When a memory block is accessed, it is likely that nearby memory blocks will be accessed in the near future due to spatial locality. By increasing the BLOCK SIZE, more data can be fetched at once, increasing the likelihood that the nearby data will be brought into the cache as well. This can reduce the number of cache misses and improve L1-D HIT RATE.

However, increasing BLOCK SIZE can also lead to higher cache miss penalties, which can degrade L1-D HIT RATE. When a larger BLOCK SIZE is used, the cache will fetch more data on a cache miss. If the requested data is not in the fetched block, the cache will need to access main memory again to fetch the missing data, resulting in a higher cache miss penalty. Additionally, larger BLOCK SIZE can also lead to increased cache thrashing due to higher memory fragmentation, which can reduce L1-D HIT RATE

Observation :

- BenchMark-1(401.bzip2) : significant decrease (0.94 to 0.62)
- BenchMark-2(429.mcf) : small increase (0.81 to 0.97)
- BenchMark-3(458.sjeng) : small increase (0.78 to 0.87)

7.5.2 L1-I HIT RATE

Expectation : Increasing BLOCK SIZE can improve spatial locality and reduce the number of cache misses caused by a lack of spatial locality. This can potentially improve L1-I HIT RATE by increasing the likelihood of bringing in nearby instructions into the cache along with the requested instruction. This can reduce the number of L1-I cache misses and improve the overall L1-I HIT RATE.

However, increasing BLOCK SIZE can also increase cache miss penalties and reduce L1-I HIT RATE. When a larger BLOCK SIZE is used, the cache fetches more data on a cache miss. If the requested instruction is not in the fetched block, the cache will need to access main memory again to fetch the missing instruction, resulting in a higher cache miss penalty. Additionally, larger BLOCK SIZE can also lead to increased cache thrashing due to higher memory fragmentation, which can reduce L1-I HIT RATE.

Observation :

- BenchMark-1(401.bzip2) : NO VARIATION (0.99)
- BenchMark-2(429.mcf) : small increase (0.87 to 0.96)
- BenchMark-3(458.sjeng) : small increase (0.99)

7.5.3 L2 HIT RATE

Expectation : Increasing BLOCK SIZE can potentially increase the spatial locality of memory accesses, which can improve L2 HIT RATE by reducing the number of cache misses caused by the lack of spatial locality. When a memory block is accessed, it is likely that nearby memory blocks will be accessed in the near future due to spatial locality. By increasing the BLOCK SIZE, more data can be fetched at once, increasing the likelihood that the nearby data will be brought into the cache as well. This can reduce the number of cache misses and improve L2 HIT RATE.

Observation :

- BenchMark-1(401.bzip2) : TRUE , small increase from (0.33 to 0.65)
- BenchMark-2(429.mcf) : TRUE , small increase from (0.1 to 0.2)
- BenchMark-3(458.sjeng) : TRUE , significant increase from (0.02 to 0.2)

7.5.4 CPI

Expectation : Increasing the BLOCK SIZE can potentially reduce the cache miss rate by improving spatial locality, as discussed in the previous answers. This can reduce the number of cycles required to access memory, thereby reducing CPI.

On the other hand, increasing BLOCK SIZE can also increase the cache miss penalty, as discussed earlier. This can increase the number of cycles required to access memory, thereby increasing CPI.

Observation :

- BenchMark-1(401.bzip2) : significant increase from (2.9 to 7.1)
- BenchMark-2(429.mcf) : small decrease from (12.1 to 8.9)
- BenchMark-3(458.sjeng) : significant decrease from (10.2 to 5.3)

Overall, from figure 6, we can see that this parameter - 'BLOCK SIZE' with larger size is improving CPI for benchmark-2,3 while reducing for benchmark-1.

8 Work Distribution

In this project, we have been assigned the following CPU benchmarks:

(a) 401.bzip2

(b) 429.mcf

(c) 458.sjeng

To ensure an equal distribution of work, we will divide the tasks among four team members:

8.1 Sai Mitra Kamasani: 401.bzip2 Benchmark

Sai Mitra Kamasani is responsible for setting up, executing, and analyzing the 401.bzip2 benchmark. He will configure cache parameters, sizes, and associativity, as well as run the gemTest.csh and getData.csh scripts for this benchmark. He will also extract relevant metrics and analyze the results.

8.2 Chitrak Vimalbhai Dave: 429.mcf Benchmark

Chitrak Vimalbhai Dave is responsible for setting up, executing, and analyzing the 429.mcf benchmark. He will configure cache parameters, sizes, and associativity, as well as run the gemTest.csh and getData.csh scripts for this benchmark. He will also extract relevant metrics and analyze the results.

8.3 Sai Kiran Paturi: 458.sjeng Benchmark

Sai Kiran Paturi is responsible for setting up, executing, and analyzing the 458.sjeng benchmark. He will configure cache parameters, sizes, and associativity, as well as run the gemTest.csh and getData.csh scripts for this benchmark. He will also extract relevant metrics and analyze the results.

8.4 Indra Teja Pidathala: Data Analysis and Trend Identification

Indra Teja Pidathala will be responsible for analyzing the data obtained from all three benchmarks. He will be responsible for identifying any patterns or trends, comparing the results of the different benchmarks, and preparing graphs and visualizations to aid in understanding the findings.

Project Report: All four members evenly contributed to the project report.

Throughout the project, all team members have been contacted through regular meetings and executed the project.

9 References

- Project Manual
- Article
- Intel Article
- Overleaf(LateX)
- Wikipedia