Computer Architecture - Homework 3 Report

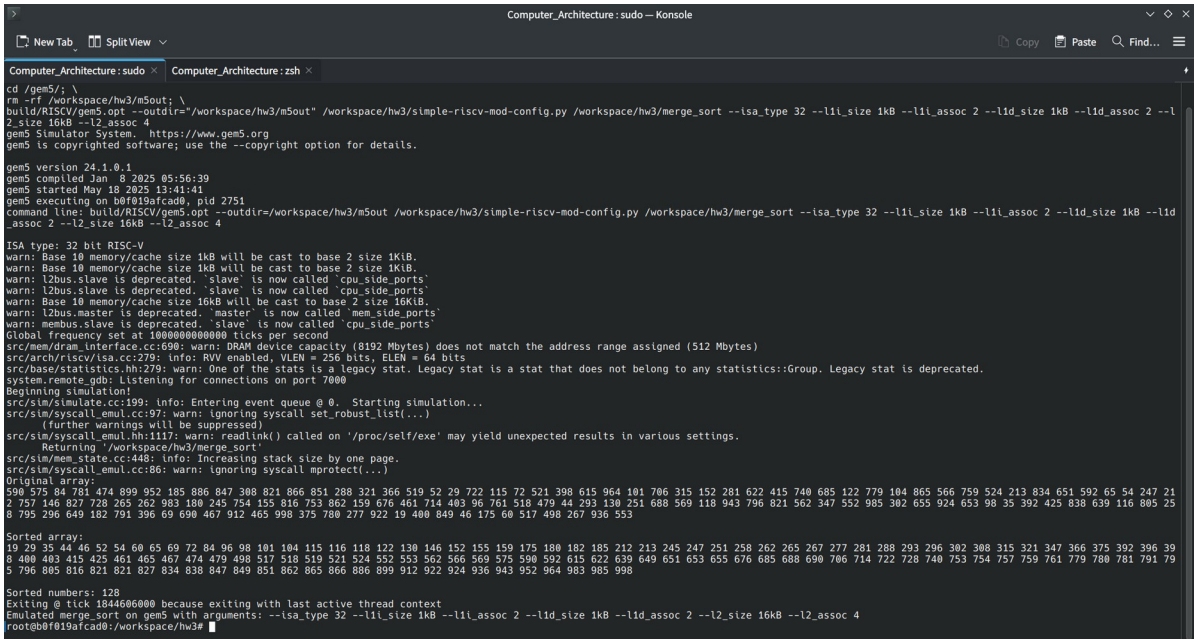
Student ID: B11901110 Name: 陳璿吉

Part1. Add new components to gem5 config.

- **1-1: Implement L2 cache in gem5 config**
- **1-2: Draw and analyze the program results based on the config**

Part1-1. Show screenshots of your program running successfully (refer to P.11) (15%)

The result is shown in the following figure.



Part1-1. Show screenshots of your program and cache summary (refer to P.12) (15%)

<program summary>

```
Program summary
----------------------------------------
simulated time                | 0.001845 s
simulated tick                | 1,844,606,000 ticks
total Inst.                   | 378,971 instructions
total cycle                   | 1,844,606 cycles
CPI                           | 4.863852
IPC                           | 0.205598
Int-Inst. count               | 376,040 instructions
Load-Inst. count              | 80,758 instructions
Store-Inst. count             | 40,786 instructions
Vector-Inst. count            | 0 instructions
```

<L1 cache summary>

```
L1-Instruction-Cache summary
---------------------------------------
$L1-I hit count               | 442,397 counts
$L1-I miss count              | 17,213 counts
$L1-I access count            | 459,610 counts
$L1-I miss rate               | 3.75% miss rate
L1-I assoc                    | 2
L1-I size                     | 1024

L1-Data-Cache summary
---------------------------------------
$L1-D hit count               | 114,824 counts
$L1-D miss count              | 6,682 counts
$L1-D access count            | 121,506 counts
$L1-D miss rate               | 5.50% miss rate
L1-D assoc                    | 2
L1-D size                     | 1024
```

<L2 cache summary>

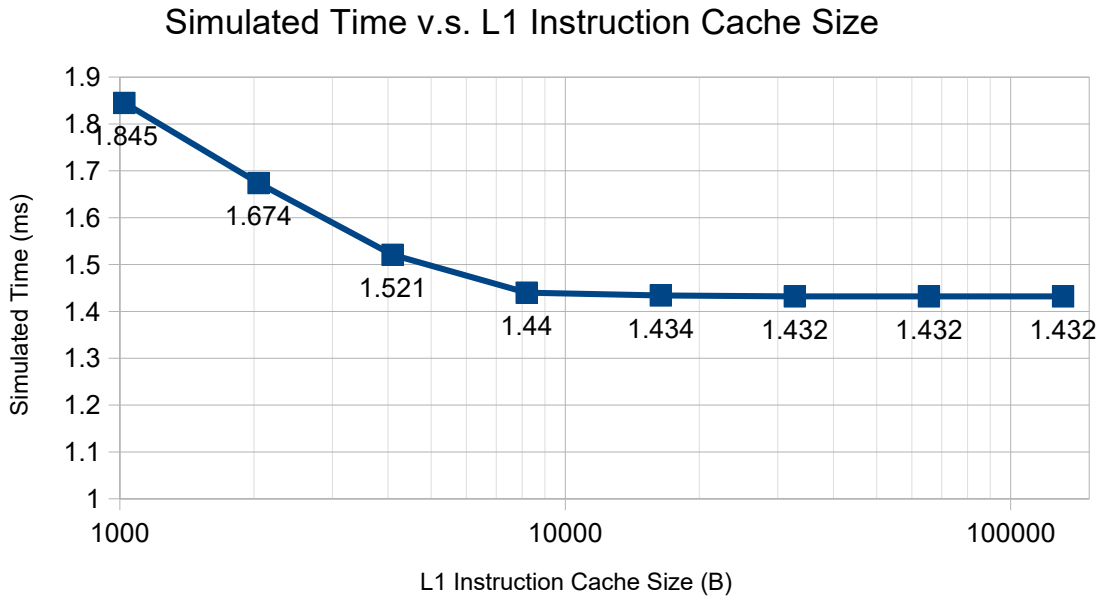
```
L2-Cache summary
----------------------------------------
$L2 hit count              | 22,493 counts
$L2 miss count             | 1,408 counts
$L2 access count           | 23,901 counts
$L2 miss rate              | 5.89% miss rate
L2 assoc                   | 4
L2 size                    | 16384
```

Part1-2. Draw a graph based on the different hardware config (refer to P.13-14). (40%)

Here is the default setting:

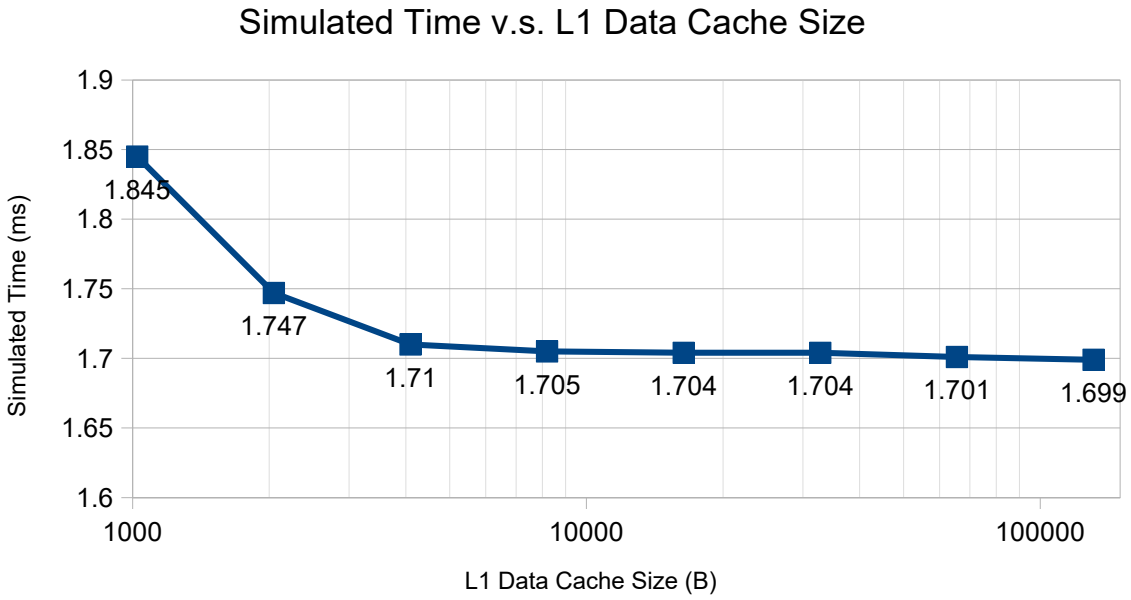
- L1 Instruction Cache Size: 1 kB
- L1 Instruction Cache Associativity: 2
- L1 Data Cache Size: 1 kB
- L1 Data Cache Associativity: 2
- L2 Cache Size: 16 kB
- L2 Cache Associativity: 4

<L1-I cache size>



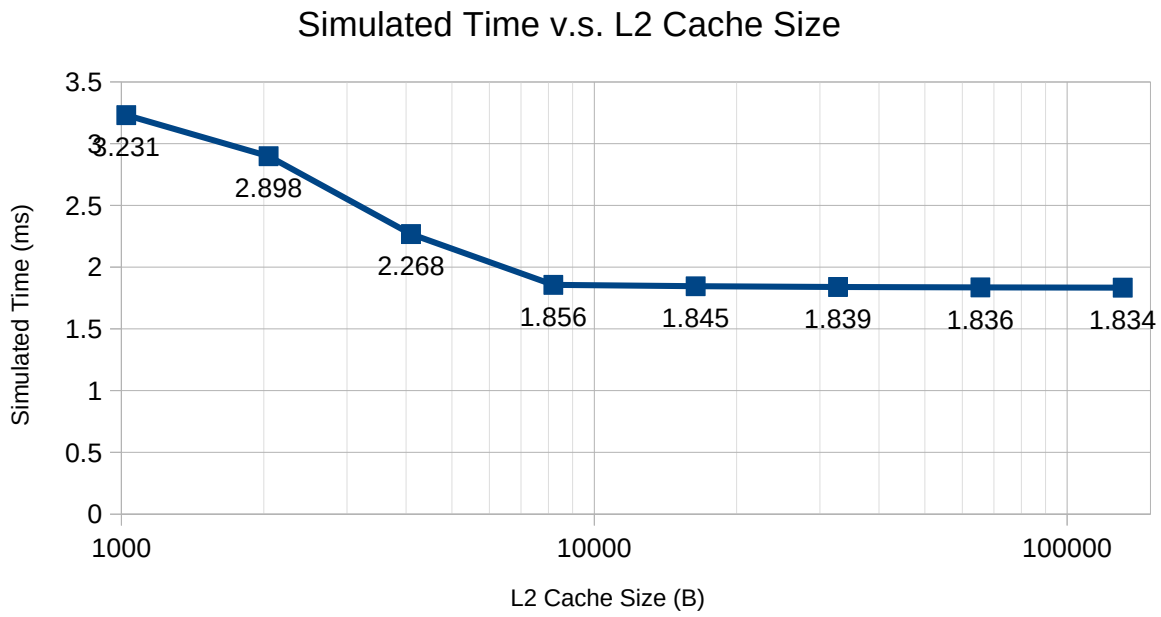
As we increase L1 instruction cache size, we can store more instructions in the cache. The miss rate decreases, but the effect diminished as L1 instruction cache size became larger than 8 kB.

<L1-D cache size>



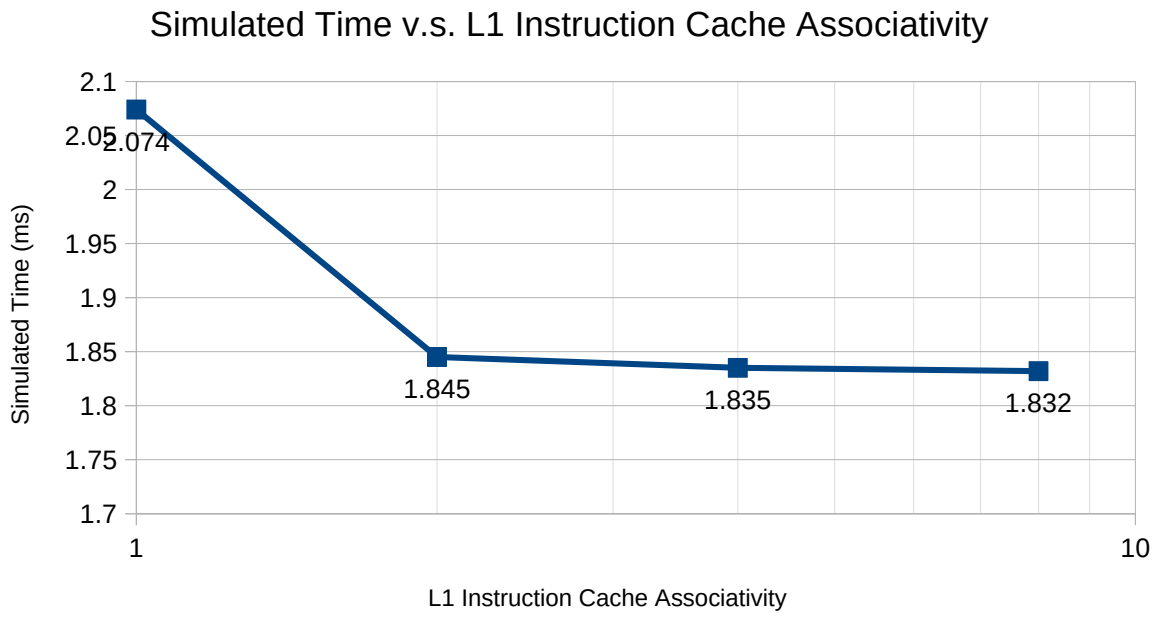
Similar to the case in L1 instruction cache, increasing the size of L1 data cache also results in the decrease of running time. Also, the effect diminished after the size was over 8 kB. However, compared to increasing the size of L1 instruction cache, increasing L1 data cache size has less significant effect.

<L2 cache size>



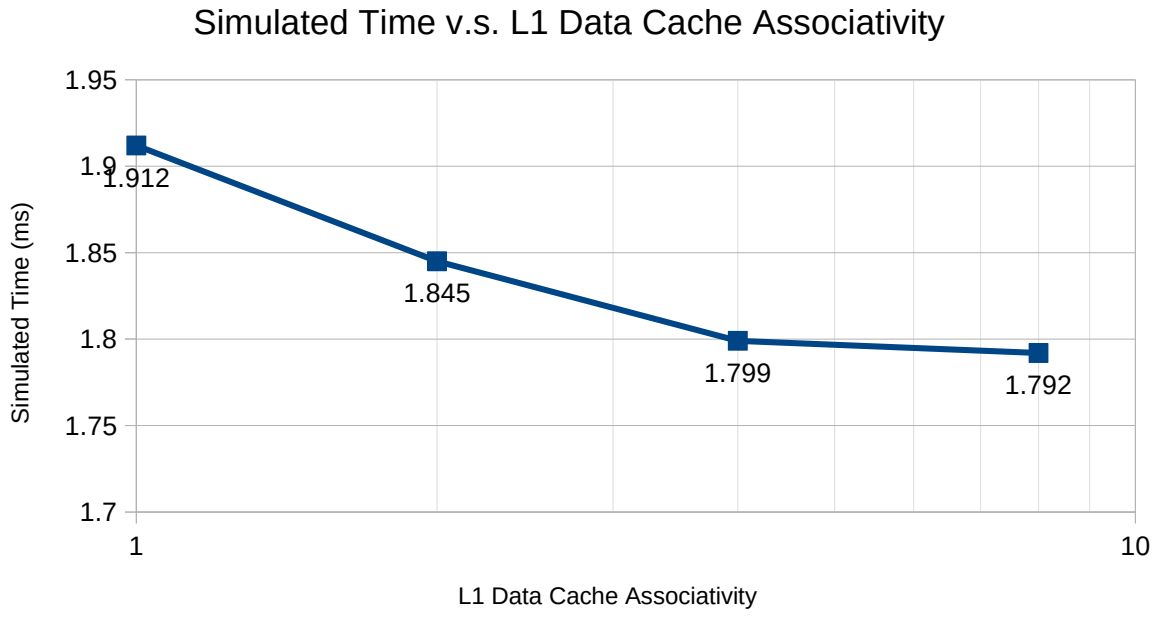
In this program, the default L2 cache size is 16 kB. Once again, the execution time decreases as L2 cache size increases. However, the effect diminished after the cache size exceeded 16 kB.

<L1-I associativity>

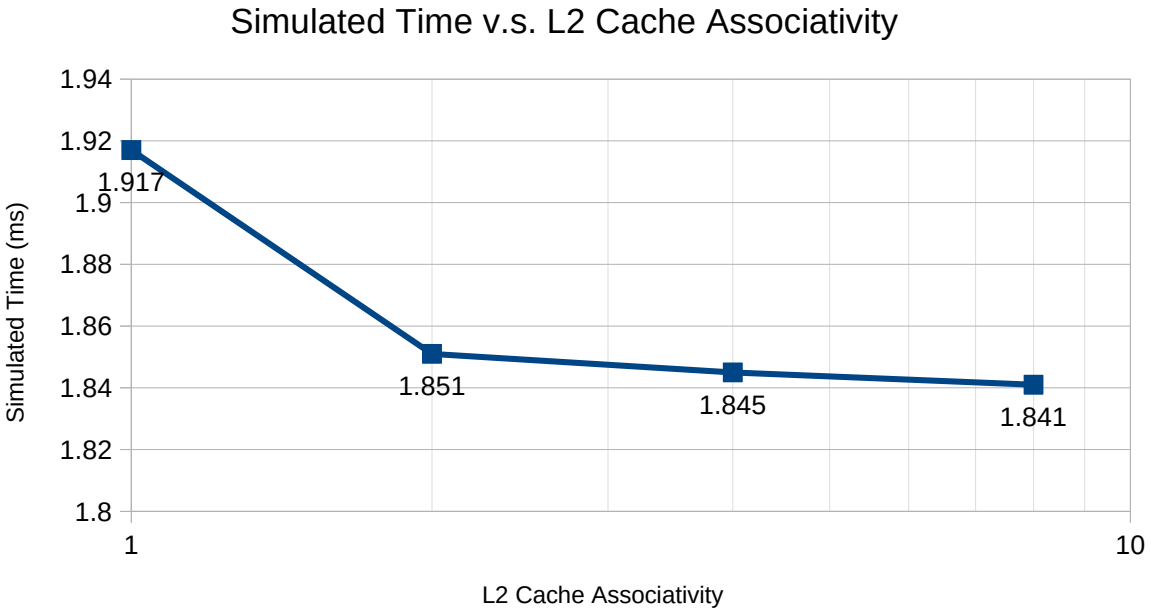


As the graph illustrated, increasing associativity of L1 instruction cache can reduce the execution time of the program. However, the effect diminished as the associativity exceeded 2.

<L1-D associativity>



Similar to the trend observed in L1 instruction cache, the execution time decreases as we increase the associativity of L1 data cache. As oppostite to instruction cache, increasing L1 data cache associativity has more effect compared to increasing L1 instruction cache associativity.
<L2 associativity>



Once again, we observed similar trend as in the previous cases. In particular, the effect is similar to increasing L1 instruction cache associativity.

Part2. Analyze and find the optimal config based on the application

- **Cache Size: 1KB ~ 128KB**
- **Associativity: 1 ~ 8**

Part2-1. Find the optimal config settings for this program. (10%)

Config	Cache size (KB)	Associativity
L1-I cache	32	8
L1-D cache	64	8
L2 cache	1	1

Performance	Score
23	12

※ Save your optimal config in gem5_args.conf

Part2-2. Explain why this config achieves optimal performance ? (20%)

By the nature of the L1 cache, it is the most frequently accessed component of a CPU. Therefore, increasing the size of L1 cache can improve the performance dramatically.

However, which one accounts for more improvement, L1 instruction cache or L1 data cache? This is related to the algorithm --- merge sort. Merge sort is a divide-and-conquer algorithm, which means that there are a lot of recursive calls. A recursive call is realized by uncondition jump in assembly language. Intuitively speaking, the program needs to fetch more instruction to perform recursive call. Therefore, we need to increase the size of the L1 instruction cache. In this case, the number of ticks stablized after the L1 cache size exceeded 32 kB.

When the recursion of merge sort reaches the the bottom (leaves), we need to perform comparison of two nodes, and put them in appropriate addresses. In assembly language, accessing data and make comparison means we need more space for data to reduce miss rate. Once we have enough space for instructions, increasing L1 data cache size can further improve the performance. To make sure the performance is optimized, we set the associativity to 8, the maximum allowed in this assignment.

What about the effect of L2 cache? By the nature of L2 cache, and the data of program execution, L2 cache access (23901 times) is significantly less than L1 cache (459610 and 121506 times). Once we have sufficient space for L1 caches, then L2 cache becomes negligible in this case. Therefore, we set the size of L2 cache to 1 kB, and the associativity to 1.