**COSC 6351 ADVANCED COMPUTER**

**ARCHITECTURE**

Project 1: Cache Simulator

**UNDER THE GUIDANCE OF**

**Dr. Chen Pan**

**BY**

Sudeesh Naidu Yenugula- (A04253575)

Bhavya Gutta-(A04249172)

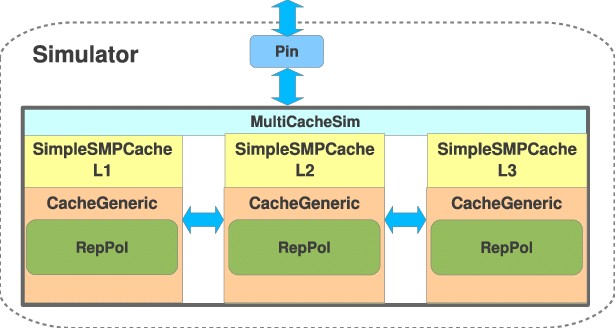
CONTENTS

1. [Exploring the L1 Cache Design 3](#_TOC_250001)
   1. Plot 1 3
   2. Plot 2 4
   3. Discussion based on Plot 1 and Plot 2 5
2. [Exploring the Replacement Policy 5](#_bookmark0)
   1. Plot 3 and Discussion 5
3. [Exploring the L2 Cache Design 7](#_TOC_250000)
   1. Plot 4 7
   2. Plot 5 8
4. [Exploring the Inclusion Property choices 8](#_bookmark1)
   1. Plot 6 8

**INTRODUCTION**

The Computer Architecture and Organization course is a required component of the undergraduate computer science (CS) curriculum and a significant portion of the body of knowledge.

However, the problem arises because high-level programming languages do not provide a clear picture of how the program is executed by the machine. Teaching computer architecture is a difficult task that requires both teachers and students to put in a lot of work. It is almost always scheduled during the first year of study and is a brand-new course for pupils. Visual simulators make teaching easier and boost students' interest in hardware in general. Teachers must select appropriate hands-on activities, assignments, and projects for their students. Liang creates a detailed list of chores and assignments to do. Modern multiprocessors use a layered cache memory technology to reduce the distance between the CPU and main memory and speed up data access. As a result, we must understand not only the architecture but also the multiprocessor's internal organization. We haven't yet found an educational simulator that will help us understand all cache parameters and how they affect program execution. In this project, we show how to use the Cache simulator to visualize cache hit and miss, cache line fulfillment, and the cache associativity problem for sequential and parallel algorithm execution.



## EXP LO RI NG THE L1 CACHE DESIGN

* 1. PLOT 1

30



445%

24. 529%

772%

10. 465%

56%

1.61

69%

1.

53%

1.85

8%

10.

72%

25% 10.7

26.4

35%

13% 27.2

27.9

L1 Miss Rate (%)

20

10

0

0 8 16 32 64

10



757

7.

5004

0.75342

0.7

0.7

4445

605

7185

3.1

874

3.2

501

275 3.2

3.2

0.9

503

7.6

621

7.8

47

8.0

AAT (ns)

5

0

0 8 16 32 64

# L1 Cache Size (KB)

Figure 1.1: L1 cache miss rate, Average Access Time (AAT) and L1 cache size, when L1 associa tivity is 4

From Figure 1.1, We've seen that when the size of the L1 cache grows, the miss rate and average access time of the L1 cache drop. However, we can see that AAT increases somewhat as the L1 cache size in benchmark GCC goes from 32KB to 64KB, as well as when it climbs from 8KB to 32KB in LBM.

0.4



0.3

Area (*mm*2)

0.2

0.1

0

0 8 16 32 64

L1 Cache Size (KB)

Figure 1.2: L1 cache area and L1 cache size when L1 cache associativity is 4

* 1. PLOT 2

30



445%

26. 424%

72%

10.772%

35%

1.64

69%

1.

19%

1% 1.8

2.69

10.7

314%

98% 10.5

26.4

35%

84% 26.5

L1 Miss Rate (%)

20

10

0

0 1 2 4 8

8



757

7. 924

37

3.2

7177

0.74445

0.7

4869

0.7

658

0.9

047

3.

874

3.2

112

3.6

7.6

34

49 7.6

7.7

6

AAT (ns)

4

2

0

0 1 2 4 8

# L1 Associativity (ways)

Figure 1.3: L1 cache miss rate, Average Access Time (AAT) and L1 cache associativity, when L1 cache size is 32KB

From Figure 1.3, We can see that as L1 cache associativity increases from 1 to 4, the L1 cache miss rate and Average Access Time of L1 cache decrease. Furthermore, when L1 cache associativity changes from one way (direct mapped) to two ways, the miss rate and AAT both drop significantly. As L1 cache associativity grows, the pace of decrease in L1 miss rate and AAT slows. When L1 is increased from two to four ways in LBM, the L1 miss rate and AAT increase. When L1 cache associativity is increased from four to eight ways, the L1 miss rate reduces but AAT climbs significantly.

0.24



Area (*mm*2)

0.23

0.22

0.21

0.2

0 1 2 4 8

L1 Associativity (ways)

Figure 1.4: L1 cache area and L1 cache associativity when L1 cache size is 32KB

* 1. DISCUSSION BASED ON PLOT 1 AND PLOT 2

In sections 1.1 and 1.2, we have some preliminary observations on the relationship between L1 miss rate, AAT, and L1 cache size, as well as associativity. As seen in Figures 1.1 and 1.3, a reduced L1 miss rate does not imply a lower AAT. We have, for no-L2-cache design,

AAT *=* L1 hit time *+* L1 miss rate *×* Miss Penalty

Increasing the size of the L1 cache or associativity results in a decreased L1 miss rate. However, as a side effect, it may increase L1 hit time. As a result, the rivalry between L1 hit time change and L1 miss rate change determines whether AAT decreases.

To improve AAT, better values for parameters of the L1 cache structure are now required.

Table 1.1: L1 Cache Area

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Area | | Cache Size | | | |
|  | | | |
| Cache Associatity | 1 | 0.053293238 | 0.096748995 | 0.210543576 | 0.330469394 |
| 2 | 0.083756164 | 0.130107044 | 0.205554649 | 0.350242085 |
| 4 | 0.068434156 | 0.105941693 | 0.236647681 | 0.302289370 |
| 8 | 0.102584886 | 0.130444675 | 0.242170635 | 0.360317611 |

Based on table 1.1, I'd want to choose 16KB 4-ways or 32KB 2-ways cache, considering what's described in sections 1.1 and 1.2.

## EXP LO RI NG THE REPL A C EM ENT POLI CY

* 1. PLOT 3 AND DISCUSSION

Benchmark:

29



L1 Miss Rate (%)

28

27

26

25

24

0 8 16 32 64

8.2

8

AAT (ns)

7.8

7.6

7.4

7.2

0 8 16 32 64

L1 Cache Size (KB)

Figure 2.1: L1 cache miss rate, AAT and L1 cache size as well as replacement policy when L1 cache associativity is 4

Benchmark:

11.5



11

L1 Miss Rate (%)

10.5

10

0 8 16 32 64

3.35



3.3

AAT (ns)

3.25

3.2

3.15

0 8 16 32 64

L1 Cache Size (KB)

Benchmark:

3.5

3

L1 Miss Rate (%)

2.5

2

1.5

0 8 16 32 64

1.2

1.1

AAT (ns)

1

0.9

0.8

0.7

0 8 16 32 64

L1 Cache Size (KB)

Figure 2.2: L1 cache miss rate, AAT and L1 cache size as well as replacement policy when L1 cache associativity is 4

We can see from the results of three benchmarks that the L1 miss rate and AAT decrease as the L1 cache capacity grows, regardless of the replacement policy used, precisely as we saw in section 1.1, with the exception.

Furthermore, we can see that the blue line is constantly below the other two, indicating that the LRU replacement policy has the best miss rate and AAT performance of all. The deadline is always higher than the other two, indicating that the one with the FIFO replacement policy performs the worst. However, as the L1 cache capacity grows, the difference in performance, i.e., miss rate and AAT, between these three replacement policies becomes smaller and smaller.

Figures 2.1 and 2.2 indicate that the blue and yellow lines are almost parallel in Figures 2.1 and 2.2. This suggests that pseudoLRU policy is almost as good as LRU policy in terms of performance.

Updating LRU and FIFO rankings is an O (1) operation in my simulator program, while fetching the LRU or FIFO position (the way that will be replaced) is an O(n) operation. Updating pseudoLRU tree is an O (log n) operation, while fetching pseudoLRU leaf (the way that will be replaced) is an O (log n) operation. Cache, on the other hand, has an associativity of no more than 8, implying that there are almost no variances in time complexity. Other methods for achieving LRU policy include using state bits or a matrix. The state bits technique saves space but makes operations more complex; the matrix method makes operations easier but takes up too much space. A pseudoLRU tree takes up less space than a matrix approach and is easier to operate than a state bits method, while performing similarly to an LRU policy. As a result, pseudoLRU is a practical policy to employ.

## EXP LO RI NG THE L2 CACHE DESIGN

* 1. PLOT 4

60



018%

45.6

39.1004%

35.9

941%

35.

41%

35.9

462%

45.

574%

45.5

172%

986%

546%

50.9

.9704%

4

53.3

573%

56.1

.623%

L2 Miss Rate (%)

50

40

30

128 256 512 1024

6



574

4.

0

7942

0.4

0.4

8133

.48019 0.4

394

8364

1.

803

1.3

676

1.3

1.4565

165

4.4

427

4.6

8914

4

AAT (ns)

2

0

128 256 512 1024

# L2 Cache Size (KB)

Figure 3.1: L2 cache miss rate, Average Access Time (AAT) and L2 cache size, when L1 cache size is 16KB, L1 associativity is 4, L2 associativity is 8

* 1. PLOT 5

70



31%

907%

60.3782%

59.

004%

39.1

76%

39.3

704%

45.

858%

523%

46.0

61%

43.2

75%

50.2

623%

46.96

62%

62.1

66.1

L2 Miss Rate (%)

60

50

40

30

0 1 2 4 8

6

4



788

1.4608

1.4565

0

27

1

0.48487

2

0.4804

4

0.48019

8

1.5

89

1.

914

4.8

433

4.9

08

5.

32

5.3

AAT (ns)

2

0

L2 Cache Size (KB)

Figure 3.2: L2 cache miss rate, Average Access Time(AAT) and L2 cache associativity, when L1 cache size is 16KB, L1 associativity is 4, L2 cache size is 128KB

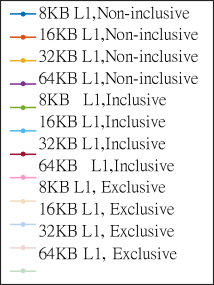
## EXP LO RI NG THE INC LUSIO N PROPERTY CH O IC ES

* 1. PLOT 6

105 Benchmark: MCF

×

11



Number of Cache Misses resulting in Memory Read

10

9

8

7

6

5

4

3

128 256 512 1024

L2 Cache Size (KB)

105 Benchmark:

×

4.6



128

256

512

1024

Number of Cache Misses resulting in Memory Read

4.5

4.4

4.3

4.1

4

3.9

3.8

128 256 512 1024

L2 Cache Size (KB)

7.5

7

6.5

6

AAT (ns)

5.5

5

4.5

4

Benchmark:

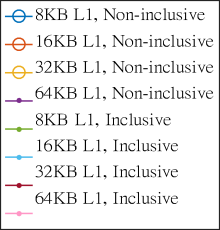
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | |  | |
|  |  |  |  | |  | |
|  |  |  |  | 8KB L1, Non-inclusive 16KB L1, Non-inclusive 32KB L1, Non-inclusive 64KB L1, Non-inclusive 8KB L1, Inclusive 16KB L1, Inclusive 32KB L1, Inclusive 64KB L1, Inclusive 8KB L1, Exclusive 16KB L1, Exclusive 32KB L1, Exclusive 64KB L1, Exclusive | |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | |  | |
|  |  |  |  | |  | |

128 256 512 1024

L2 Cache Size (KB)

Benchmark:

5



4.9

4.8

4.7

AAT (ns)

4.6

4.5

4.4

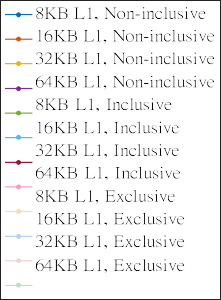
128 256 512 1024

L2 Cache Size (KB)

105 Benchmark:

×

4



Number of Cache Misses resulting in Memory Read

3.5

3

2.5

2

1.5

1

128 256 512 1024

L2 Cache Size (KB)

105 Benchmark:

×

1.46



128

256

Number of Cache Misses resulting in Memory Read

1.44

1.42

1.4

1.38

1.36

1.34

1.32

1.3

128 256 512 1024

L2 Cache Size (KB)

2.8

2.6

2.4

2.2

AAT (ns)

2

1.8

1.6

1.4

1.2

Benchmark:

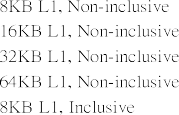
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | | |  | |
|  |  |  |  |  | 8KB L1, Non-inclusive 16KB L1, Non-inclusive 32KB L1, Non-inclusive 64KB L1, Non-inclusive 8KB L1, Inclusive 16KB L1, Inclusive 32KB L1, Inclusive | |  |
|  |  |  |  |  |  |
|  |
|  |
|  |  |  |  |  |
|  |
| 64KB L1, Inclusive  8KB L1, Exclusive | | |
|  |  |  |  |  |
|  | 16KB L1, Exclusive  32KB L1, Exclusive 64KB L1, Exclusive | |
|  |
|  |  |  |  |  |
|  |
|  |
|  | | |  | |
|  |  |  |  | | |  | |
|  |  |  |  | | |  | |

128 256 512 1024

L2 Cache Size (KB)

Benchmark:

1.6



128

128

128

1.55

1.5

AAT (ns)

1.45

1.4

1.35

1.3

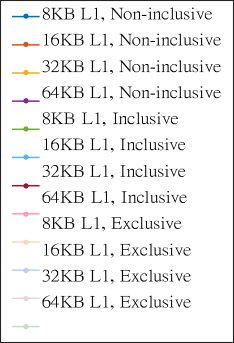
128 256 512 1024

L2 Cache Size (KB)

104 Benchmark:

×

8.5



Number of Cache Misses resulting in Memory Read

8

7.5

128

128

7

128

256

6.5

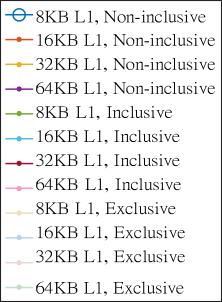
6

128 256 512 1024

L2 Cache Size (KB)

Benchmark:

0.85



128 256

128 256

128 256

0.8

0.75

0.7

AAT (ns)

0.65

0.6

0.55

0.5

0.45

128 256 512 1024

L2 Cache Size (KB)