Presented to Sir Ronald Pascual

Department of Computer Technology – College of Computer Studies

De La Salle University – Manila

Term 3, A.Y. 2023-2024

CSARCH2 Simulation Project Write Up: Cache Simulator (Block-set-associative / LRU)

In Partial Fulfillment of the Course Requirements for

INTRODUCTION TO COMPUTER ORGANIZATION AND ARCHITECTURE 2

Submitted by:

Badiola, Maxine Beatriz Ante

Jaramillo, Brandon Anthony

Ngo, Way We Pelovello

Vitan, Layne Ashley Divinagracia

Group 5

Section S14

July 31, 2024

Introduction

For the Final CSARCH2 Deliverable: the Simulation Project, the members of Group 5 were tasked with creating a web-based application with a Graphical User Interface of a Cache Simulator that uses Block-set-associative mapping and LRU replacement. This application was developed to provide a visual interface for visualizing and understanding cache behavior.

Project Overview

The cache simulator allows various input parameters amd simulate program based on these inputs with the key features includes:

For Input Parameter:

1. Block Size: Size of each block in words.
2. Set Size: Size of each set in blocks.
3. Main Memory Size: Size of main memory specified in blocks or words in the dropdown.
4. Cache Size: Size of cache size specified in blocks or words in the dropdown.
5. Program Flow: Sequence of memory accesses in blocks.

For Output Display:

1. Cache hits - Number of times of the data is found in the cache.
    a. Format: “x/y”
2. Cache misses - Number of times of the data is not found in the cache.
    a. Format: “x/y”
3. Miss penalty - Time Penalty when a cache miss occurs.
    a. Format: “xyz ns“
4. Average memory access time - Average time taken to access memory, considering both hits and misses..
    a. Format: “xyz.ab ns“
5. Total memory access time - Total time taken for all memory accesses during the simulation.
    a. Format: “xyz ns“
6. Cache memory snapshot - Snapshot of the cache memory.

    a. Display final block instances being accessed and their global age.
7. Cache memory table - Tabular representation of the cache memory.
    a. Display the table representing the final blocks accessed.
8. Cache log - A detailed log of the cache operations.
    a. Display which blocks were accessed, set allocations, cache placements, and hit and miss.

How to Access

Access this link on your browser to view the Cache Simulator:

https://ashoolotl.github.io/CSARCH2-SIMULATION-PROJECT/

Usage Instructions

1. Input the following parameters:
    a. Block Size (words) - Enter Block Size in words.
    b. Set Size (blocks) - Enter Set Size in blocks.
    c. Main Memory Size - Enter Main Memory Size and Select between Blocks or Words type from the dropdown box.
    d. Cache Size - Enter Cache Size Select between Blocks or Words type from the dropdown box.
    e. Program Flow - Enter Sequence of the memory access separated by commas.
        i. Example: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
2. Simulate Cache
    a. Click the Simulate Cache button to run the simulation.
    b. The results, including cache hits, cache misses, miss penalty, average memory access time, total memory access time, cache memory snapshot, cache log, cache sequence will be displayed.
3. Export Results (Optional)
    a. Click the Export Results button to download the results as a text file.

Note: The cache access time is a constant of 1ns and the memory access time is a constant of 10ns.

Test Cases

1. Normal Test Case:
   This is the default test case that was already taught in an online recording from canvas with an established correct answer.
   The expected output is (disregarding the age):

Set	Block 0	Block 1
0	0	2
Age	*11*	*10*
1	5	1
Age	*8*	*5*

   Input the following below:

- Block Size (words): 2
- Set Size (blocks): 2
- Main Memory Size: 24 Words or 12 Blocks
- Cache Size: 8 Words or 4 Blocks

- Program Flow: 1, 7, 5, 0, 2, 1, 5, 6, 5, 2, 2, 0 as Blocks

Number of Cache Hits: 4/12

Number of Cache Misses: 8/12

Miss Penalty: 22 ns

Average Memory Access Time: 15.00 ns

Total Memory Access Time: 192 ns

Cache Memory Snapshot:


```
[
 [
  {
 "block": 0,
"lastUsed": 11
  },
  {
 "block": 2,
"lastUsed": 10
  }
 ],
 [
  {
 "block": 5,
"lastUsed": 8
  },
  {
 "block": 1,
"lastUsed": 5
  }
 ]
]
```

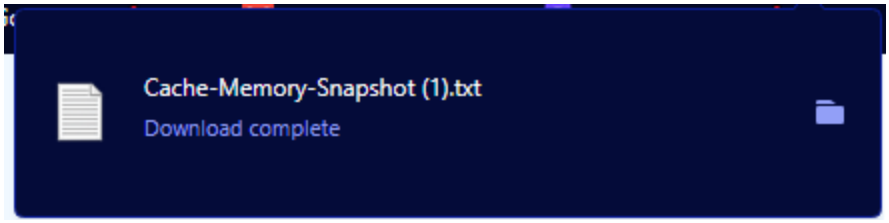

Cache Memory Table:

Set	Block 0	Block 1
Set 0	0	2
Set 1	5	1

Cache Log:

Age	Sequence Number	Set	Block	Hit or Miss
0	1	(1 % 2 = Set 1)	0	Miss
1	7	(7 % 2 = Set 1)	1	Miss
2	5	(5 % 2 = Set 1)	0	Miss
3	0	(0 % 2 = Set 0)	0	Miss
4	2	(2 % 2 = Set 0)	1	Miss
5	1	(1 % 2 = Set 1)	1	Miss
6	5	(5 % 2 = Set 1)	0	Hit
7	6	(6 % 2 = Set 0)	0	Miss
8	5	(5 % 2 = Set 1)	0	Hit
9	2	(2 % 2 = Set 0)	1	Hit
10	2	(2 % 2 = Set 0)	1	Hit
11	0	(0 % 2 = Set 0)	0	Miss

When the Export as TXT Button is clicked:



This is the text file displaying the complete output:

Cache-Memory-Snapshot (1) - Notepad

File Edit Format View Help

```
Number of Cache Hits: 4/12

Number of Cache Misses: 8/12

Miss Penalty: 22 ns

Average Memory Access Time: 15.00 ns

Total Memory Access Time: 192 ns

Cache Memory Snapshot:
[
  [
    {
      "block": 0,
      "lastUsed": 11
    },
    {
      "block": 2,
      "lastUsed": 10
    }
  ],
  [
    {
      "block": 5,
      "lastUsed": 8
    },
    {
      "block": 1,
      "lastUsed": 5
    }
  ]
]
Cache Memory Table:
Set     Block 0 Block 1
Set 0   0       2
Set 1   5       1
Cache Log:
Age     Sequence Number Set     Block   Hit or Miss
0       1       ( 1 % 2 = Set 1)        0       Miss
1       7       ( 7 % 2 = Set 1)        1       Miss
2       5       ( 5 % 2 = Set 1)        0       Miss
3       0       ( 0 % 2 = Set 0)        0       Miss
4       2       ( 2 % 2 = Set 0)        1       Miss
5       1       ( 1 % 2 = Set 1)        1       Miss
6       5       ( 5 % 2 = Set 1)        0       Hit
7       6       ( 6 % 2 = Set 0)        0       Miss
8       5       ( 5 % 2 = Set 1)        0       Hit
9       2       ( 2 % 2 = Set 0)        1       Hit
10      2       ( 2 % 2 = Set 0)        1       Hit
11      0       ( 0 % 2 = Set 0)        0       Miss
```

2. Alternate Values Case:

Similarly with the previous case, this is still a simple 2*2 cache with an expected output of:

Set	Block 0	Block 1
0	4	6
Age	*4*	*5*
1	7	3
Age	*4*	*5*

Input the following below:

- Block Size (words): 4
- Set Size (blocks): 2
- Main Memory Size: 8 blocks or 16 words
- Cache Size: 4 Blocks or 8 words
- Program Flow: 3,2,0,4,9,1,0,4,5,6,7,3 as blocks

Number of Cache Hits: 2/12

Number of Cache Misses: 10/12

Miss Penalty: 42 ns

Average Memory Access Time: 35.17 ns

Total Memory Access Time: 458 ns

Cache Memory Snapshot:


```
[
  [
    {
  "block": 4,
"lastUsed": 7
    },
    {
  "block": 6,
"lastUsed": 9
    }
  ],
  [
    {
  "block": 7,
"lastUsed": 10
    },
    {
  "block": 3,
"lastUsed": 11
    }
  ]
]
```

Cache Memory Table:

Set	Block 0	Block 1
Set 0	4	6
Set 1	7	3

Cache Log:

Age	Sequence Number	Set	Block	Hit or Miss
0	3	(3 % 2 = Set 1)	0	Miss
1	2	(2 % 2 = Set 0)	0	Miss
2	0	(0 % 2 = Set 0)	1	Miss
3	4	(4 % 2 = Set 0)	0	Miss
4	9	(9 % 2 = Set 1)	1	Miss
5	1	(1 % 2 = Set 1)	0	Miss
6	0	(0 % 2 = Set 0)	1	Hit
7	4	(4 % 2 = Set 0)	0	Hit
8	5	(5 % 2 = Set 1)	1	Miss
9	6	(6 % 2 = Set 0)	1	Miss
10	7	(7 % 2 = Set 1)	0	Miss
11	3	(3 % 2 = Set 1)	1	Miss

3. All MISS case:

The following test involves a scenario that will not involve any accessing of the same blocks with an expected output of:

Set	Block 0	Block 1	Block 2	Block 3
0	2	4	6	8
Age	*1*	*3*	*5*	*7*
1	1	3	5	7
Age	*0*	*2*	*4*	*6*

Input the following below:

- Block Size (words): 2
- Set Size (blocks): 4
- Main Memory Size: 4 blocks or 8 words
- Cache Size: 8 Blocks or 16 words
- Program Flow: 1,2,3,4,5,6,7,8 as blocks

Number of Cache Hits: 0/8

Number of Cache Misses: 8/8

Miss Penalty: 22 ns

Average Memory Access Time: 22.00 ns

Total Memory Access Time: 184 ns

Cache Memory Snapshot:


```
[
  [
    {
  "block": 2,
 "lastUsed": 1
    },
    {
  "block": 4,
 "lastUsed": 3
    },
    {
  "block": 6,
 "lastUsed": 5
    },
    {
  "block": 8,
 "lastUsed": 7
    }
  ],
  [
    {
  "block": 1,
 "lastUsed": 0
    },
    {
  "block": 3,
 "lastUsed": 2
    },
    {
  "block": 5,
 "lastUsed": 4
    },
    {
  "block": 7,
 "lastUsed": 6
    }
  ]
]
```


Cache Memory Table:

Set	Block 0	Block 1	Block 2	Block 3
Set 0	2	4	6	8
Set 1	1	3	5	7

Cache Log:

Age	Sequence Number	Set	Block	Hit or Miss
0	1	(1 % 2 = Set 1)	0	Miss
1	2	(2 % 2 = Set 0)	0	Miss
2	3	(3 % 2 = Set 1)	1	Miss
3	4	(4 % 2 = Set 0)	1	Miss
4	5	(5 % 2 = Set 1)	2	Miss
5	6	(6 % 2 = Set 0)	2	Miss
6	7	(7 % 2 = Set 1)	3	Miss
7	8	(8 % 2 = Set 0)	3	Miss

4. Other Sets not used Case:

This case will involve 4 sets within the cache but only having set 1 and set 3 being used with an expected output of:

Set	Block 0	Block 1
0		
Age		
1	1	5
Age	2	3
2		
Age		
3	7	3
Age	3	2

Input the following below:

- Block Size (words): 2
- Set Size (blocks): 2
- Main Memory Size: 4 blocks or 8 words
- Cache Size: 8 Blocks or 16 words
- Program Flow: 1,2,3,4,5,6,7,8 as blocks

Number of Cache Hits: 3/8

Number of Cache Misses: 5/8

Miss Penalty: 22 ns

Average Memory Access Time: 14.13 ns

Total Memory Access Time: 121 ns

Cache Memory Snapshot:


```
[
 [
  {
 "block": -1,
"lastUsed": -1
  },
  {
 "block": -1,
"lastUsed": -1
  }
 ],
 [
  {
 "block": 1,
"lastUsed": 5
  },
  {
 "block": 5,
"lastUsed": 7
  }
 ],
 [
  {
 "block": -1,
"lastUsed": -1
  },
  {
 "block": -1,
"lastUsed": -1
  }
 ],
 [
  {
 "block": 7,
"lastUsed": 6
  },
  {
 "block": 3,
"lastUsed": 4
  }
 ]
]
```


Cache Memory Table:

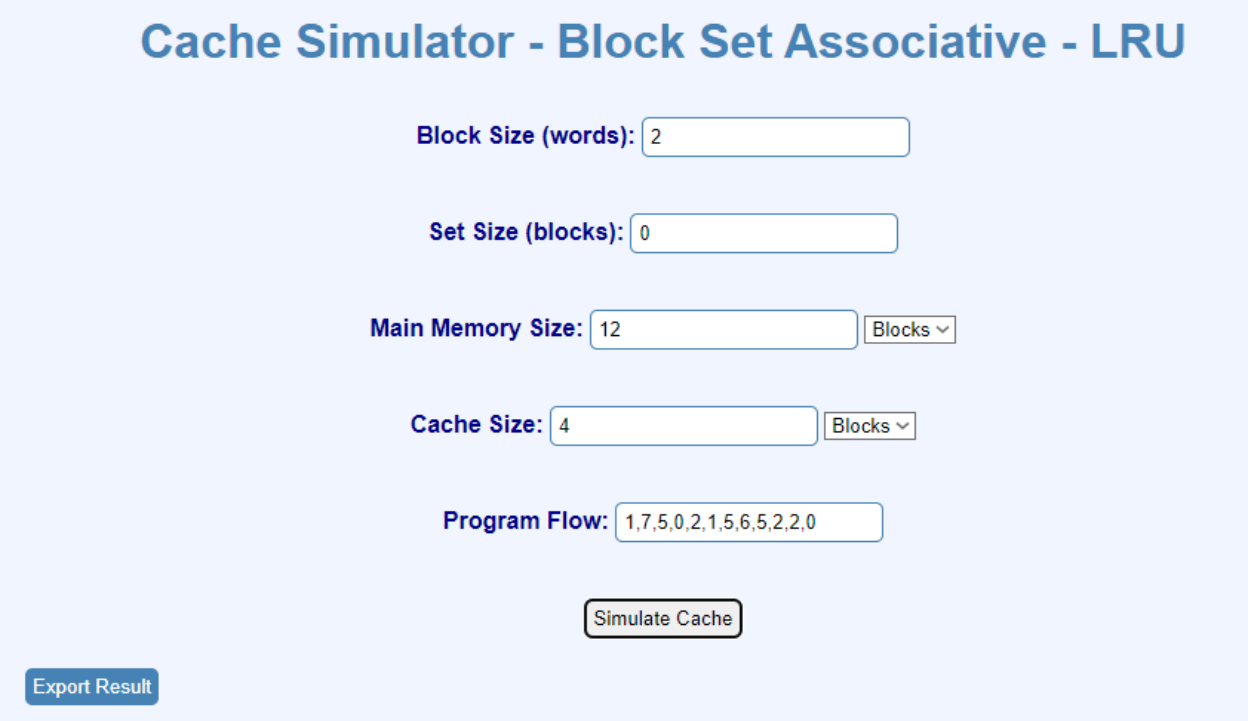
Set	Block 0	Block 1
Set 0	-	-
Set 1	1	5
Set 2	-	-
Set 3	7	3

Cache Log:

Age	Sequence Number	Set	Block	Hit or Miss
0	7	(7 % 4 = Set 3)	0	Miss
1	9	(9 % 4 = Set 1)	0	Miss
2	3	(3 % 4 = Set 3)	1	Miss
3	5	(5 % 4 = Set 1)	1	Miss
4	3	(3 % 4 = Set 3)	1	Hit
5	1	(1 % 4 = Set 1)	0	Miss
6	7	(7 % 4 = Set 3)	0	Hit
7	5	(5 % 4 = Set 1)	1	Hit

5. Invalid Input Case:

   This case will demonstrate what will happen if an invalid set size value of 0 is input:



   The image above shows that in the event of an invalid set size value, nothing will be printed. If the export result button were to be used, the text file would be empty.

6. Large volume of Program Flow Case:

   For this case, the program flow will consist of 108 blocks of input:

   Program Flow:

   1,7,5,0,2,1,5,6,5,2,2,0,1,7,5,0,2,1,5,6,5,2,2,0,1,7,5,0,2,1,5,6,5,2,2,0,1,7,5,0,2,1,5,6,5,2,2,0,1,7,5,0,2,1,5,6,5,2,2,0,1,7,5,0,2,1,5,6,5,2,2,0,1,7,5,0,2,1,5,6,5,2,2,0,1,7,5,0,2,1,5,6,5,2,2,0,1,7,5,0,2,1,5,6,5,2,2,0 as blocks.

Number of Cache Hits: 60/108

Number of Cache Misses: 48/108

Miss Penalty: 22 ns

Average Memory Access Time: 10.33 ns

Total Memory Access Time: 1224 ns

Cache Memory Snapshot:


```
[
 [
  {
"block": 0,
"lastUsed": 107
  },
  {
"block": 2,
"lastUsed": 106
  }
 ],
 [
  {
"block": 5,
"lastUsed": 104
  },
  {
"block": 1,
"lastUsed": 101
  }
 ]
]
```


Cache Memory Table:

Set	Block 0	Block 1
Set 0	0	2
Set 1	5	1

Cache Log:

Age	Sequence Number	Set	Block	Hit or Miss
0	1	(1 % 2 = Set 1)	0	Miss
1	7	(7 % 2 = Set 1)	1	Miss
2	5	(5 % 2 = Set 1)	0	Miss
3	0	(0 % 2 = Set 0)	0	Miss
4	2	(2 % 2 = Set 0)	1	Miss
5	1	(1 % 2 = Set 1)	1	Miss
6	5	(5 % 2 = Set 1)	0	Hit
7	6	(6 % 2 = Set 0)	0	Miss
8	5	(5 % 2 = Set 1)	0	Hit
9	2	(2 % 2 = Set 0)	1	Hit
10	2	(2 % 2 = Set 0)	1	Hit
11	0	(0 % 2 = Set 0)	0	Miss
12	1	(1 % 2 = Set 1)	1	Hit
13	7	(7 % 2 = Set 1)	0	Miss
14	5	(5 % 2 = Set 1)	1	Miss
15	0	(0 % 2 = Set 0)	0	Hit
16	2	(2 % 2 = Set 0)	1	Hit
17	1	(1 % 2 = Set 1)	0	Miss
18	5	(5 % 2 = Set 1)	1	Hit
94	2	(2 % 2 = Set 0)	1	Hit
95	0	(0 % 2 = Set 0)	0	Miss
96	1	(1 % 2 = Set 1)	0	Hit
97	7	(7 % 2 = Set 1)	1	Miss
98	5	(5 % 2 = Set 1)	0	Miss
99	0	(0 % 2 = Set 0)	0	Hit
100	2	(2 % 2 = Set 0)	1	Hit
101	1	(1 % 2 = Set 1)	1	Miss
102	5	(5 % 2 = Set 1)	0	Hit
103	6	(6 % 2 = Set 0)	0	Miss
104	5	(5 % 2 = Set 1)	0	Hit
105	2	(2 % 2 = Set 0)	1	Hit
106	2	(2 % 2 = Set 0)	1	Hit
107	0	(0 % 2 = Set 0)	0	Miss

Files

HTML (index.html)

The frontend for the cache simulator interface, including user inputs and display area.

JavaScript (script.js)

The backend for the cache simulator logic.

CSS (style.css)

The visual style of the cache simulator interface, including layout, and colors.