

# JSCacheSimulator

JSCacheSimulator shows how a processor cache works, how addresses are mapped into cache positions and blocks are replaced, according to various parameters that can be set up for each simulation.

## How can I use it?

The simulator was written in HTML5, using JQuery and Bootstrap libraries. To use it, open the file "index.html" (index.html) with a recent browser.

## How does it work?

### JSCacheSimulator

This page shows how an L1 cache works, given a list of memory accesses. Various Cache parameters, as size, and mapping methods can be specified. Stats for each simulation are shown below. Click here for an explanation.

**Legend:** { T = TAG; L.A = Last Access }



Press "New" to open the new simulation dialog. In this dialog the parameters Cache Size, Block Size, Set Size, Mapping Mode, and Replacement Mode can be set. Additionally, the user is asked to insert a list of memory operations. The list can be also be imported from a txt file or randomly generated by the simulator, which will try to mimic the typical operations in a generic program. Pressing the "Submit" button, the page is updated to show the simulation interface.

## JSCacheSimulator Description

**RAM SIZE:**

16 MB

**CACHE SIZE:**

8 KB

**BLOCK SIZE:**

512 B

**METHOD:**

Direct mapped

**Set:**

8 Blocks

**Algorithm:**

LRU

**Addresses in input (HEX)**

Generate Random

Choose File No file chosen

512C  
104F  
0202  
104F  
9E8F  
9E8B  
080A  
2F2F  
F2F2  
350A  
049F  
150A  
666D

Close Submit

The simulation interface shows the list of operations on the left, and the cache layout on the right.

The "Step" button executes a single memory operation, displaying details about the operation in the central area and updating the operation list and the cache layout. After each step, simulation statistics, displayed in the bottom area of the screen, are updated. The "Run" button executes all the remaining operations at once and updates the statistics (while not updating the cache layout).

A new simulation can be launched with "New". The statistics from previous simulations are kept to allow comparing the performance of various setups.

NewStepRunReset

Memory Access

Memory Access

512C

104F

0202

104F

9E8F

9E8B

080A

2F2F

F2F2

350A

049F

150A

666D

512C

Address

104F

-----24 B Address-----

Tag: 14 BIndex: 1 BOffset: 9 B

000000000001000000001001111

INDEX = Identifies a blockposition in cache 0

OFFSET = Identifies the bytes sequence inside the block

TAG = Label associated with the position of the block

This memory block isn't in cache - MISS

Cache

SetsFrames

0

Frame 0 [T: 20] [L.A: 512C]

Frame 1 [T: 4] [L.A 104F]

Frame 2

Frame 3

Frame 4

Frame 5

Frame 6

Frame 7

1

Frame 0

Frame 1

Frame 2

Frame 3

Frame 4

Frame 5

Frame 6

Frame 7

Memory Size: 2<sup>24</sup>Cache Size: 2<sup>13</sup>Block Size: 2<sup>9</sup>Set Size: 2<sup>3</sup>Mapping Method: SetAlgorithm: LRU

Accesses:2Hits: 0 (0.00%)Misses: 2 (100.00%)

Memory Size: 2<sup>24</sup>Cache Size: 2<sup>13</sup>Block Size: 2<sup>9</sup>Set Size: --Mapping Method: DirectAlgorithm: --

Accesses:26Hits: 11 (42.31%)Misses: 15 (57.69%)

License

Copyright (C) 2014 Emanuele Viglianisi

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.