University of Texas Rio Grande Valley
ScholarWorks @ UTRGV

Theses and Dissertations - UTRGV

8-2023

NRC-SIM: A Node-RED Based Multi-Level, Many-Core Cache Simulator

Ezequiel Treviño
The University of Texas Rio Grande Valley

Follow this and additional works at: https://scholarworks.utrgv.edu/etd

Part of the Electrical and Computer Engineering Commons

Recommended Citation

Treviño, Ezequiel, "NRC-SIM: A Node-RED Based Multi-Level, Many-Core Cache Simulator" (2023). *Theses and Dissertations - UTRGV*. 1410.
https://scholarworks.utrgv.edu/etd/1410

This Thesis is brought to you for free and open access by ScholarWorks @ UTRGV. It has been accepted for inclusion in Theses and Dissertations - UTRGV by an authorized administrator of ScholarWorks @ UTRGV. For more information, please contact justin.white@utrgv.edu, william.flores01@utrgv.edu.

NRC-SIM: A NODE-RED BASED MULTI-LEVEL, MANY-CORE CACHE SIMULATOR

A Thesis

by

EZEQUIEL TREVIÑO

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE IN ENGINEERING

Major Subject: Electrical Engineering

The University of Texas Rio Grande Valley

August 2023

NRC-SIM: A NODE-RED BASED MULTI-LEVEL,

MANY-CORE CACHE SIMULATOR

A Thesis
by
EZEQUIEL TREVIÑO

COMMITTEE MEMBERS

Dr. Mark Yul Chu
Chair of Committee

Dr. Heinrich Foltz
Committee Member

Dr. Bin Fu
Committee Member

August 2023

Copyright 2023 Ezequiel Treviño

All Rights Reserved

ABSTRACT

Treviño, Ezequiel, NRC-SIM: A Node-RED Based Multi-Level, Many-Core Cache Simulator. Master of Science in Engineering (MSE), August, 2023, 331 pp., 2 tables, 121 figures, references, 34 titles.

As computational systems become ever-more integral to daily life, so too does the importance of understanding how these complex systems work. For those unfamiliar with the underlying concepts, this can be a daunting task. In an effort to address such concerns, this paper presents a Node-RED based cache simulator that enables users to observe the effects of their desired cache configuration, with users having the ability to easily modify various parameters, such as the core count, the number of levels within the cache, and coherence protocols, among other parameters. Through the use of Node-RED, NRC-SIM allows for simplicity of use by providing web-based cache simulation to any web-connected device, including but not limited to, computers, laptops, and mobile devices. As such, users need only select their desired parameters, allowing the formatting for execution to be handled in the background. Node-RED's modular, flow-based design enables the execution of NRC-SIM, allowing the user's selected inputs to be manipulated in such a way that they are easily converted into the proper format. As NRC-SIM is a trace-driven cache simulator, various trace files have been collected from benchmark programs found in PARSEC and SPLASH2 through the use of Intel's Pin, an open–source dynamic instrumentation tool framework. NRC-SIM's experimental results show that it is a highly efficient, well-rounded cache simulator that generates results that are either comparable or an improvement over similar established cache simulators, such as SMPCache and SIMNCORE. As such, NRC-SIM can be an effective simulation tool for research purposes, or an educational tool that provides those less familiar with the concepts of cache memory an introduction into the subject matter.

DEDICATION

This thesis is dedicated to my family, without whom, the completion of my Master's degree in Electrical Engineering would not have been possible. Thank you for everything.

ACKNOWLEDGMENTS

I would like to express my gratitude to the members of my thesis committee, whose teaching instilled in me a desire to be the best student I can be. My gratitude is especially so for Dr. Mark Yul Chu, the chair of my thesis committee, for his unwavering support and mentorship. Without him, this thesis, and my interest in its subject matter, would be nonexistent.

TABLE OF CONTENTS

Page

ABSTRACT iii
DEDICATION iv
ACKNOWLEDGMENTS v
TABLE OF CONTENTS vi
LIST OF TABLES viii
LIST OF FIGURES ix
CHAPTER I. INTRODUCTION 1
1.1 Motivation 2
CHAPTER II. RELATED WORKS 4
2.1 Problem Statement 5
CHAPTER III. PROPOSED SIMULATOR: NRC-SIM 7
3.1 Designing Cache Memory 8
3.2 Replacement Policies 11
3.2.1 Least Recently Used 11
3.2.2 Most Recently Used 12
3.2.3 Random 13
3.2.4 Round-Robin 13
3.3 Victim Cache 14
3.4 Coherence Protocols 16
3.4.1 Single-Core Cache Simulation 18
3.4.1.1 Write-Back in Single-Core Processors. 20
3.4.1.2 Write-Through in Single-Core Processors. 24
3.4.2 Multi-Core and Many-Core Cache Simulation 27
3.4.2.1 MSI Coherence Protocol. 27
3.4.2.2 MESI Coherence Protocol. 34
3.4.2.3 FIREFLY Coherence Protocol. 41
3.5 Node-RED Trace Files 47
3.5.1 Generating Trace Files 49

3.5.2 Collecting User Inputs 53
3.5.2.1 Node-RED Dashboard. 53
3.5.3 Execution 55
3.5.4 Outputting Data 55
3.5.4.1 Node-RED Dashboard. 56
3.6 Microsoft Azure 58
CHAPTER IV. SIMULATION METHODOLOGY 60
4.1 Benchmark Programs 61
4.2 Pin 62
CHAPTER V. EXPERIMENTAL RESULTS 65
CHAPTER VI. CONCLUSION 105
6.1 Future Work 105
REFERENCES 106
APPENDIX A 109
APPENDIX B 124
APPENDIX C 131
APPENDIX D 245
BIOGRAPHICAL SKETCH 331

LIST OF TABLES

Page

Table 4.1: SPLASH-2 Benchmarks 61

Table 4.2: PARSEC Benchmarks 62

LIST OF FIGURES

Page

Figure 3.1: NRC-SIM Command Line Parameters 9
Figure 3.2: LRU Replacement Policy Example 12
Figure 3.3: MRU Replacement Policy Example 13
Figure 3.4: Round-Robin Replacement Policy Example 14
Figure 3.5: Victim Cache Organization 15
Figure 3.6: The Coherence Problem Example 17
Figure 3.7: Many-Core Memory Hierarchy 18
Figure 3.8: Single-Core Memory Hierarchy 19
Figure 3.9: Write Policies 20
Figure 3.10: Write-Back Policy for Single-Core Processor Containing Two Levels of Cache and a Victim Cache 21
Figure 3.11: Write-Through Policy for Single-Core Processor Containing Two Levels of Cache and a Victim Cache 25
Figure 3.12: MSI State Transitions: CPU Requests and BUS Requests 29
Figure 3.13: MSI Coherence Protocol for Two Levels of Cache and a Victim Cache 33
Figure 3.14: MESI State Transitions: CPU Requests and BUS Requests 36
Figure 3.15: MESI Coherence Protocol for Two Levels of Cache and a Victim Cache 40
Figure 3.16: FIREFLY State Transitions: CPU Requests and BUS Requests 42
Figure 3.17: FIREFLY Coherence Protocol for Two Levels of Cache and a Victim Cache . . 46
Figure 3.18: Cache Hit/Miss Rates in NRC-SIM 48
Figure 3.19: Trace File Selection in NRC-SIM 49
Figure 3.20: Hamburger Menu in NRC-SIM 50
Figure 3.21: Generate Trace File Landing Page in NRC-SIM 50
Figure 3.22: Benchmark Program Selection for Trace File Generation in NRC-SIM 51
Figure 3.23: Input Parameters for Trace File Generation in NRC-SIM 52
Figure 3.24: Result of Trace File Generation in NRC-SIM 52
Figure 3.25: NRC-SIM User Interface 54
Figure 3.26: NRC-SIM - L1 Only Simulation Output (1/2) 57

Figure 3.27: NRC-SIM - L1 Only Simulation Output (2/2) 57
Figure 4.1: NRC-SIM Overview 60
Figure 4.2: Pin Instrumentation Formatting Example 63
Figure 4.3: Trace File Example 64
Figure 5.1: Effects of Cache Configuration on Streamcluster 65
Figure 5.2: Effects of Cache Configuration on Fluid Animate 66
Figure 5.3: Effects of Cache Configuration on Radix 66
Figure 5.4: Effects of Cache Configuration on Raytrace 67
Figure 5.5: Effects of Cache Configuration on Ocean_CP 67
Figure 5.6: Effects of Associativity on Streamcluster 68
Figure 5.7: Effects of Associativity on Fluid Animate 69
Figure 5.8: Effects of Associativity on Radix 69
Figure 5.9: Effects of Associativity on Raytrace 70
Figure 5.10: Effects of Associativity on Ocean_CP 70
Figure 5.11: Single-Core, L1 Miss Rate 71
Figure 5.12: Single-Core, Victim Cache Hit Rate 72
Figure 5.13: 16-Core, L1 Miss Rate 73
Figure 5.14: 16-Core, Victim Cache Hit Rate 73
Figure 5.15: 16-Core, L1 Miss Rate 74
Figure 5.16: 16-Core, Victim Cache Hit Rate 75
Figure 5.17: 16-Core, L2 Miss Rate 75
Figure 5.18: 16-Core, Global Miss Rate 76
Figure 5.19: Effects of Victim Cache on Streamcluster 77
Figure 5.20: Effects of Victim Cache on Fluid Animate 77
Figure 5.21: Effects of Victim Cache on Radix 78
Figure 5.22: Effects of Victim Cache on Raytrace 78
Figure 5.23: Effects of Victim Cache on Ocean_CP 79
Figure 5.24: Effects of Block Size on Streamcluster 80
Figure 5.25: Effects of Block Size on Fluid Animate 80
Figure 5.26: Effects of Block Size on Radix 81
Figure 5.27: Effects of Block Size on Raytrace 81
Figure 5.28: Effects of Block Size on Ocean_CP 82
Figure 5.29: Single-Core, L1 Only Cache - Effects of Replacement Policy on Streamcluster . 83

Figure 5.30: Single-Core, L1 Only Cache - Effects of Replacement Policy on Fluid Animate 83
Figure 5.31: Single-Core, L1 Only Cache - Effects of Replacement Policy on Radix 84
Figure 5.32: Single-Core, L1 Only Cache - Effects of Replacement Policy on Raytrace . . . 84
Figure 5.33: Single-Core, L1 Only Cache - Effects of Replacement Policy on Ocean_CP . . 85
Figure 5.34: 16-Core, 2 Levels of Cache (L1) - Effects of Replacement Policy on Streamcluster 85
Figure 5.35: 16-Core, 2 Levels of Cache (L1) - Effects of Replacement Policy on Fluid Animate 86
Figure 5.36: 16-Core, 2 Levels of Cache (L1) - Effects of Replacement Policy on Radix . . . 86
Figure 5.37: 16-Core, 2 Levels of Cache (L1) - Effects of Replacement Policy on Raytrace . 87
Figure 5.38: 16-Core, 2 Levels of Cache (L1) - Effects of Replacement Policy on Ocean_CP 87
Figure 5.39: 16-Core, 2 Levels of Cache (L2) - Effects of Replacement Policy on Streamcluster 88
Figure 5.40: 16-Core, 2 Levels of Cache (L2) - Effects of Replacement Policy on Fluid Animate 88
Figure 5.41: 16-Core, 2 Levels of Cache (L2) - Effects of Replacement Policy on Radix . . . 89
Figure 5.42: 16-Core, 2 Levels of Cache (L2) - Effects of Replacement Policy on Raytrace . 89
Figure 5.43: 16-Core, 2 Levels of Cache (L2) - Effects of Replacement Policy on Ocean_CP 90
Figure 5.44: 16-Core, 2 Levels of Cache - Effects of Coherence Protocol on Streamcluster . 91
Figure 5.45: 16-Core, 2 Levels of Cache - Effects of Coherence Protocol on Fluid Animate . 91
Figure 5.46: 16-Core, 2 Levels of Cache - Effects of Coherence Protocol on Radix 92
Figure 5.47: 16-Core, 2 Levels of Cache - Effects of Coherence Protocol on Raytrace 92
Figure 5.48: 16-Core, 2 Levels of Cache - Effects of Coherence Protocol on Ocean_CP . . . 93
Figure 5.49: Effects of Victim Cache on Bus Traffic - Streamcluster (MSI) 94
Figure 5.50: Effects of Victim Cache on Bus Traffic - Fluid Animate (MSI) 94
Figure 5.51: Effects of Victim Cache on Bus Traffic - Radix (MSI) 95
Figure 5.52: Effects of Victim Cache on Bus Traffic - Raytrace (MSI) 95
Figure 5.53: Effects of Victim Cache on Bus Traffic - Ocean_CP (MSI) 96
Figure 5.54: Effects of Victim Cache on Bus Traffic - Streamcluster (MESI) 96
Figure 5.55: Effects of Victim Cache on Bus Traffic - Fluid Animate (MESI) 97
Figure 5.56: Effects of Victim Cache on Bus Traffic - Radix (MESI) 97
Figure 5.57: Effects of Victim Cache on Bus Traffic - Raytrace (MESI) 98
Figure 5.58: Effects of Victim Cache on Bus Traffic - Ocean_CP (MESI) 98
Figure 5.59: Effects of Victim Cache on Bus Traffic - Streamcluster (FIREFLY) 99
Figure 5.60: Effects of Victim Cache on Bus Traffic - Fluid Animate (FIREFLY) 99
Figure 5.61: Effects of Victim Cache on Bus Traffic - Radix (FIREFLY) 100
Figure 5.62: Effects of Victim Cache on Bus Traffic - Raytrace (FIREFLY) 100

Figure 5.63: Effects of Victim Cache on Bus Traffic - Ocean_CP (FIREFLY) 101
Figure 5.64: 16-Core, 2 Levels of Cache - Effects of Core Count on Streamcluster 102
Figure 5.65: 16-Core, 2 Levels of Cache - Effects of Core Count on Fluid Animate 102
Figure 5.66: 16-Core, 2 Levels of Cache - Effects of Core Count on Radix 103
Figure 5.67: 16-Core, 2 Levels of Cache - Effects of Core Count on Raytrace 103
Figure 5.68: 16-Core, 2 Levels of Cache - Effects of Core Count on Ocean_CP 104
Figure A.1: FFT Build Confirmation 112
Figure A.2: Radix Build Confirmation 112
Figure A.3: FFT Run Confirmation (1/2) 113
Figure A.4: FFT Run Confirmation (2/2) 113
Figure A.5: Radix Run Confirmation (1/2) 114
Figure A.6: Radix Run Confirmation (2/2) 114
Figure A.7: Original FFT Executable Location 115
Figure A.8: Original Radix Executable Location 116
Figure A.9: Executables in Pin's ManualExamples Folder 116
Figure A.10:Modifications to pinatrace.cpp 117
Figure A.11:Resulting Output of pinatrace.cpp Compilation 118
Figure A.12:Pin Instrumentation With Associated Parameters and Environment 119
Figure A.13:Resulting Output After Generating Single-Core FFT Trace File 119
Figure A.14:Resulting Output After Generating 8-Core FFT Trace File 120
Figure A.15:Resulting Output After Generating Single-Core Radix Trace File 120
Figure A.16:Resulting Output After Generating 8-Core Radix Trace File 121
Figure A.17:Singe-Core FFT Trace File 122
Figure A.18:8-Core FFT Trace File 122
Figure A.19:Singe-Core Radix Trace File 123
Figure A.20:8-Core Radix Trace File 123
Figure B.1: Generalized Flowchart of NRC-SIM 128
Figure B.2: Detailed Flowchart of NRC-SIM 129
Figure B.3: Victim Cache Size Selection in NRC-SIM 130

CHAPTER I

INTRODUCTION

The evolution of microprocessors is nothing short of extraordinary. From their humble beginnings in the early 1970s [1], to the modern marvels that are behind the worldwide industry valued at over $100 billion [2], microprocessors have become an increasingly important component to daily life for much of the world's population. Within 30 years of its introduction, IBM would introduce the world's first multi-core processor, the POWER4, opening the door for what was possible [3]. With each passing year, microprocessors would continue to improve in performance and expand on the concept of multi-core processing. In time, multi-core processors would come to be defined as having 2-8 cores, with many-core processors being defined as any core count greater than or equal to 10. While increasing the core count generally improves system performance, memory latency and efficiency is still a major bottleneck, as the processor speed is orders of magnitude faster than the fastest memory speed possible - data needs to be accessible to the processing unit as quickly as possible, otherwise, the system will remain in an idle state, resulting in wasted time and computational resources. In an effort to mitigate this issue, cache memory was introduced in 1965, enabling researchers to improve memory latency and efficiency in the years since [4].

Just as any other constituent of microprocessors, cache memory configurations undergo meticulous testing to ensure optimal performance and efficiency. This testing is done via simulations of real world workloads, which help evaluate valuable performance metrics such as the effects of varying associativity, the effectiveness of the desired coherence protocol, detailed bus traffic information, average memory access time, etc. In their survey on cache simulators, Brais, Kalayappan, and Panda, define three modes of simulation, which is how a given simulator works - execution-driven, emulation-driven, and trace-driven [5]. Execution-driven simulators run "the

input program to be simulated natively on the host platform" (e.g., SimpleScalar[6], SMTSIM[7]), while emulation-driven simulators create "a virtual platform on which the input program is run" (e.g., Multi2Sim [8]) [5]. Trace-driven simulation, the final mode of simulation and most relevant to this thesis, requires the use of a "trace file generated from an execution or emulation of a program" to be used as the input [5]. Brais, Kalayappan, and Panda continue by stating trace-driven simulators are often more simple in design and implementation, while being more deterministic to their counterparts, leading to the possibility of more accurate performance evaluations "without having to use statistical methods" [5].

Through the use of Node-RED [9], NRC-SIM allows for complex trace-driven cache simulation through an easy-to-use web-based interface. Node-RED is a powerful browser-based editor that allows users to wire together hardware devices, APIs, and online services. In addition to the baseline functionality of Node-RED, users are able to create and install additional modules to further bolster Node-RED's capabilities. Built on Node.js, Node-RED takes full advantage of its event-driven, non-blocking model, making it ideal to run on low-cost hardware such as the cloud, Raspberry Pi, and Arduino.

1.1 Motivation

As has been established thus far, cache memory is essential in modern computing systems to alleviate memory latency, further boosting overall performance and efficiency. As such, research into its optimization and betterment across a multitude of factors is vitally important. While there are many open-source cache simulators, access to them can be challenging to most. Likewise, of the cache simulators that are available, few of them attempt to simulate many-core systems with multiple levels of cache memory in a manner that is both easy to use and approachable for those beginning their education on the subject. For example, the previously mentioned SimpleScalar is viewed favorably for simulating cache memory configurations on single-core CPUs, however, it is not ideal for simulating multi-core system designs. Meanwhile, simulators that do offer a competent many-core cache simulation, such as SIMNCORE [10], are not readily available to the general public and require knowledge on the Linux environment to proceed with execution via the

command line. NRC-SIM aims to remedy these inconveniences.

One of the major impetuses behind this thesis was providing an easy to use, easy to modify cache simulator that can be used for both research and educational purposes. Furthermore, the cache simulator should be accessible enough as an educational tool for future additions to be made and easily observed. To achieve this, NRC-SIM is a trace-driven cache simulator, in which traces are collected using Intel's Pin [11], an open-source dynamic instrumentation tool framework. By utilizing Node-RED, a flow-based programming tool, NRC-SIM allows users to simulate a multitude of cache configurations, allowing them to observe the effects of their design decisions at both the local and global level, as well as allow them to compare the cost-effectiveness of said designs. As a browser-based platform, Node-RED enables access from a multitude of devices and web browsers, all while maintaining consistent execution and performance. Within the Node-RED editor, the nodes used in NRC-SIM's implementation allow for quick and easy debugging, which is especially useful when handling the amount of data being processed by the cache simulation as well as ensuring proper formatting for execution to occur. Node-RED's ability to easily create functions that can combine the functionality of multiple nodes into one allows for highly performant, simpler flows with a high degree of flexibility. This, in turn, allows users to run complex cache simulations with ease.

CHAPTER II

RELATED WORKS

The SimpleScalar [6] toolset provides an infrastructure for simulation and architectural modeling, allowing users to modify configuration parameters such as the number of cache levels, total cache size, associativity, replacement policy, etc [5]. As it is an execution-driven simulator, SimpleScalar provides "access to all data produced and consumed during program execution", allowing for significant accuracy in the simulation [6]. However, this simulation technique is not only time-consuming, but also introduces increased complexity in the model and difficulties in reproducing experiments that may depend on real-world external factors [6]. In addition to these drawbacks, SimpleScalar only offers simulation for single-core architectural models, removing metrics such as the effects of varying cache coherence. As such, SimpleScalar is no longer representative of modern computing systems.

The Multi2Sim [8] simulation framework does support multi-core simulation, however, it only offers a "fixed and simple" [5] cache coherence configuration. As a timing simulator, Multi2Sim is able to provide a high level of accuracy and "produce detailed statistics about the timing/performance of a target system" [12]. However, modifying cache configurations can be a difficult and strenuous task, and as such, cannot be considered as a suitable educational tool.

Dinero IV [13], unlike SimpleScalar and Multi2Sim, is purely focused on cache simulation. It is a trace-driven simulator that allows for the simulation of multi-level caches for single-core architectures, and only for single-core architectures. It does not support multi-core system configurations, nor many-core system configurations. As such, coherence performance metrics are not possible.

The CASPER [14] simulation framework, unlike the previously mentioned SimpleScalar

and Multi2Sim, is a trace-driven simulator with a sole focus on cache simulation. This focus allows CASPER to deliver a well-rounded performance analysis on varying cache architecture configurations, such as differing cache hierarchies, coherence protocols, and replacement policies. While mostly feature complete, CASPER has a "fixed and simple" write policy, limiting its potential [5]. Likewise, its closed-source nature makes it impossible to modify and improve upon its strong foundation.

The gem5 [15] simulator is a state of the art, highly configurable simulation infrastructure that is capable of simulating various commercial Instruction Set Architectures and, such as ARM, Power, and x86, among other ISAs as well. This simulation framework is the result of a highly collaborative effort between various academic and industrial institutions. The gem5 simulator is open-source and supports many of the features lacking from those previously mentioned, making it ideal for research purposes. For instance, gem5 is capable of modeling multiple levels of cache among many-core systems with support for various coherence protocols [5]. As gem5 is primarily aimed at researchers, however, workloads are immense - e.g., a simulation took over eight hours to complete for a program whose execution should only take one second in reality [16]. This fact, coupled with its research-focused audience makes gem5 an ill-suited educational tool.

Since its inception in 2013, Node-RED has amassed an active development community, resulting in a vast function library. It quickly became an essential development tool for many, culminating in IBM contributing it as an open-source project as part of the JS Foundation, which later merged with the Node.js Foundation to form the OpenJS Foundation [17] in 2019. Due to its ease of use, scalability, and reliability Node-RED has become the standard for many vendors in the industrial and computing sectors including, but not limited to, Raspberry Pi, Siemens, and Emerson. As such, Node-RED was the ideal choice for creating a simplistic, yet robust web-based cache simulator.

2.1 Problem Statement

As has been explained thus far, simulation is a vital component to research and development of microprocessor systems. There are many types of simulation methods, each with their

own distinct advantages and disadvantages. The previously mentioned gem5, for instance, has a significant overhead that is not present in similar trace-driven simulators. In an effort to avoid this pitfall, and expand on the potential educational impact, this thesis focuses on creating a trace-driven cache simulator that enables users to quickly and easily simulate cache architectures of their own design. The trace files will be generated from industry standard benchmark programs, such as those found in the PARSEC [16] and SPLASH-2 [18] Benchmark Suites, and will be used to measure the performance of the desired cache configuration. These benchmarks will then be used to generate the trace files using Intel's Pin for various multi-core and many-core cache memory configurations.

While largely inspired by equivalent cache simulators, NRC-SIM aims to provide an ease of use and modification, allowing students the opportunity to more closely observe the effects of the various parameters that are used in modern cache memory architectures. Its trace-driven nature removes any excessive overhead that may occur with other modes of simulation, such as execution-driven simulators. NRC-SIM's intuitive Node-RED based interface removes any complications or confusion that may arise when dealing with simulators that require considerable knowledge on command line execution. Additionally, Node-RED's modular nature allows for the possibility of further additions to be made in the future. These changes can be small, such as a change of color in the user interface, or large, such as adding additional functionality to the underlying cache simulation. Regardless of the changes made, Node-RED's ability to easily debug and manipulate complex sets of data make it a powerful tool that is perfectly suited for the purposes of this thesis.

CHAPTER III

PROPOSED SIMULATOR: NRC-SIM

Most cache simulators, such as those previously mentioned, require a considerable effort from the user to strictly follow the procedures put in place for execution to occur. Meanwhile, simulators that are not as complex from the user's point of view, are usually not as sophisticated in design, being restricted to either single-core or single-level cache simulation - and in some instances, being restricted to both. NRC-SIM is designed to be a versatile simulation tool, capable of many-core cache simulation with up to two levels of cache, as well as a possible victim cache [19] for the first level. Through the use of Node-RED, NRC-SIM is capable of completing robust cache simulation via a user-friendly, web-based interface, removing any barriers that may arise for the end-user. Users are able to design whatever cache configuration they desire with ease by inputting a multitude of parameters. These parameters include the cache size, block size, coherence protocols, replacement policies, conventional cache schemes such as direct-mapped, n-way set-associative, etc. As the user inputs their desired parameters, Node-RED stores these values and combines them into the proper format for command line execution. For example, consider a user wishing to simulate the following: L1 only cache simulation with a cache size of 16KB, a block size of 64B, utilizing 2-way set-associativity, the write-back policy, and the most recently used replacement policy on 4-cores with the MESI coherence protocol and 5 L1 victim cache lines in conjunction with the corresponding Raytrace benchmark trace file. For a majority of these parameters, the user need only select the desired parameter from a drop-down menu - the cache size, block size, and associativity must be entered manually in the log base 2 format. This was done to not hinder the possibilities of a users desired cache configuration. As previously mentioned, Node-RED then formats these inputs into a string in the format shown in Figure 3.1. In the case of our example, the

command line input would be as follows: "./NRC-SIM -L1 14:6:1:4 -L2 0:0:0:0 -C4 -P1 -W1 -V5 raytrace_4c.out". This string is then passed to an execution node within Node-RED that runs the command in the command line terminal. This process runs underneath everything and is never once seen by the user, meaning they don't have to worry about memorizing or formatting the command line input manually as it is done automatically by Node-RED.

3.1 Designing Cache Memory

NRC-SIM is written in C++ and executed via the command line, however, through the use of Node-RED, the end user never has to interact with the simulator in this manner. Be that as it may, when users select their desired cache parameters via the Node-RED interface, the parameters are then formatted such that the command line execution can be accomplished without fail. To do so, the cache memory hierarchy must be constructed based on the user's desired inputs for the following parameters: cache size, block size, associativity, and replacement policy. Next, the core count is decided, which will affect the following parameter - the coherence protocol. Next, The command line instruction expects the desired write-policy and desired victim cache lines, if any. Lastly, a trace file is used as input. The figure below shows the command line input in detail.

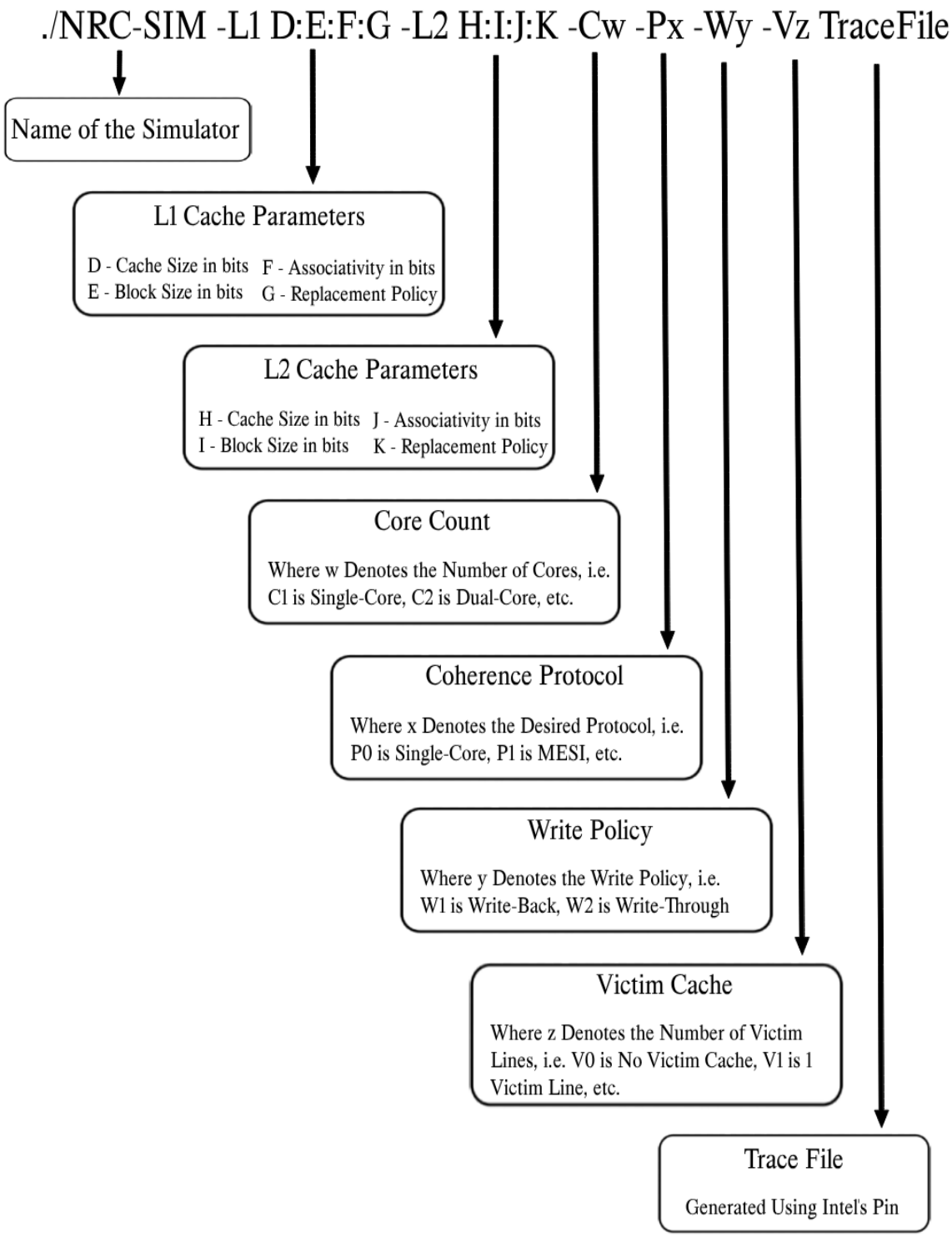

Figure 3.1: NRC-SIM Command Line Parameters

Source: Adapted From [10]

The individual cache parameters are preceded by -L1 and -L2, representing the level 1 cache and level 2 cache, respectively. Three of the four cache parameters, those being the cache size, block size, and associativity, are given values in the log base 2 format. For instance, to achieve a cache size of 1KB, the parameter 'D/H' should be 10, resulting in 2^{10} or 1024 Bytes. Similarly, to achieve a block size of 32B, the 'E/I' parameter should be 5, as 2^5 is equal to 32. The same is also true for the desired associativity - inputting a 0 in the 'F/J' parameter results in 2^0, or 1, which refers to a direct-mapped cache. For a 2-way set associative cache, the 'F/J' parameter should be 1, just as for a 4-way set associative cache, the 'F/J' parameter should be 2, etc. To simulate a cache with only one level, the L2 cache parameters are all set to 0.

Next, the desired core count is decided, which is directly tied to two other parameters - the coherence protocol and the trace file. In the case of the coherence protocol, user's should not be allowed to enter a value that doesn't correspond to their desired core count. For example, a single-core simulation should not be allowed to have the MESI coherence protocol selected. Likewise, a multi-core or many-core simulation will only be allowed to run if a coherence protocol is selected. This is done to ensure optimal results and is set accordingly in the case of a user entering an incompatible value.

After selecting their desired write policy (1 for write-back/write-allocate or 2 for write-through/write no-allocate), users can add an optional L1 victim cache. If the value entered for the victim cache parameter is 0, then no victim cache is included in the simulation. However, any numerical value greater than 0 is representative of the desired number of victim cache lines. In his original paper on the subject [19], Jouppi discovered that small victim caches (e.g., 2 to 5 entries) effectively reduce conflict misses for direct-mapped caches. NRC-SIM extends this by allowing users to add an L1 victim cache up to 16 entries, regardless of the desired L1 associativity.

Lastly, the command line execution requires a trace file to act as the input for the simulator. This trace file must be in the correct format and must properly represent the desired number of cores. This trace file includes all memory references for a given benchmark program. Each memory reference must be properly parsed such that the cache index, block offset, and tag are extracted.

Since the user inputs their desired block size, only the cache index and tag bits are required to be found. This is done using the following equations [20].

$$Index\ Bits = log_2(\frac{Cache\ Size}{Block\ Size * Associativity}) \tag{3.1}$$

$$Tag\ Bits = Total\ Bits - Index\ Bits - Block\ Offset\ Bits \tag{3.2}$$

For example, for a desired cache size of 2KB, block size of 32 Bytes, and a 2-way set-associativity, a 32-bit memory address will be divided into 22 tag bits, 5 index bits, and 5 block offset bits. It is important that the number of tag, index, and block bits are precise as they are how the simulator will decide whether or not there is a cache hit or miss.

It's important to note that the previously mentioned command line execution is done in the background and is never once seen by the end-user, with Node-RED acting as the intermediary between the user and the cache simulation.

3.2 Replacement Policies

Replacement policies determine how data in the cache is replaced. Naturally, the policy decided upon plays a huge role in the overall performance of the simulator. For example, in a direct-mapped cache, data can only ever be in one possible location - in the event of a write miss, this data must be evicted and replaced with the new data. However, for a 4-way set associative cache, this write miss will be able to be placed within any of the four positions within the set. The position decided upon will be based on the selected replacement policy.

3.2.1 Least Recently Used

The least recently used replacement policy keeps an ordered list, tracking the references to each block [21]. This list is constantly updated and must be kept up to date for proper functionality. When a block needs to be replaced, the variable holding the least recently used block within a set is selected. To achieve this, every cache block has an LRU variable. Through a separate function, this LRU variable is either set to 0 (in the case of a cache hit) or incremented (in the case of a

Figure 3.2: LRU Replacement Policy Example
Source: Primary

cache miss). The block with the highest LRU value is then selected as the least recently used block and is replaced. The figure above demonstrates this replacement policy with the following access sequence: A, B, C, D, E, D, F.

3.2.2 Most Recently Used

The most recently used replacement policy, like the LRU policy previously mentioned, keeps an ordered list that tracks the references to each block. Likewise, it too must be constantly updated for proper functionality. Each cache block has an MRU value that is either set to 0 (in the case of a cache miss) or incremented (in the case of a cache hit). The block with the highest MRU value is then selected as the most recently used block and is replaced. The following figure demonstrates this replacement policy with the following access sequence: A, B, C, D, E, C, D, B.

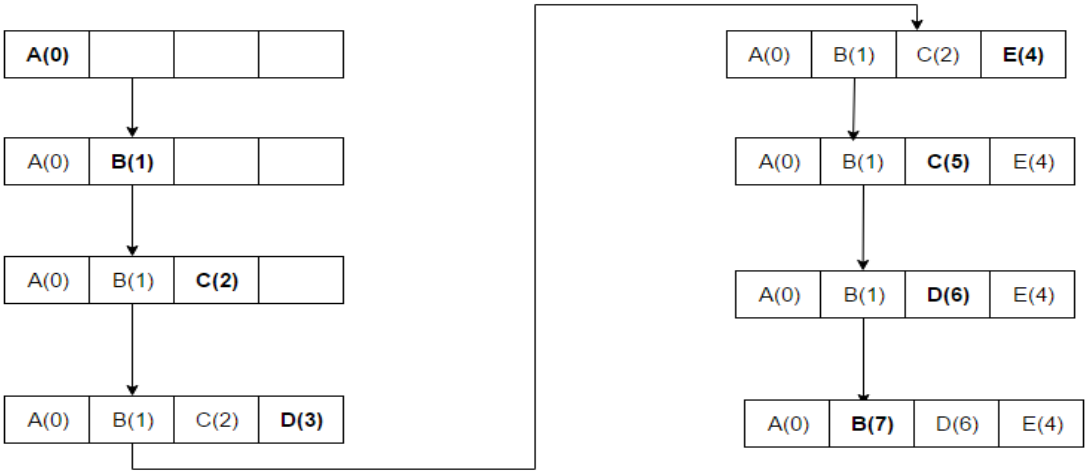

Figure 3.3: MRU Replacement Policy Example
Source: Primary

3.2.3 Random

As its name suggests, random replacement selects a block from the eviction set at random to be replaced [21]. While it may seem unlikely to garner favorable performance, the random replacement policy has been found to be only *slightly* worse than the LRU replacement policy [22], [23]. While implementing a truly random replacement policy would be very difficult, most practical implementations [21], as well as NRC-SIM's, utilize a pseudo-random generation to select the block to be replaced from the candidate set.

3.2.4 Round-Robin

The round-robin replacement policy, or FIFO - First-In-First-Out, increments a counter that is used to select the block to be evicted from the candidate set, selecting the candidate that "has resided in the cache for the longest amount of time" [21]. The queue structure is circular in nature, as the counter will be reset once all previously cached items have been evicted and replaced. The following figure demonstrates this replacement policy with the following access sequence: A, B, C, D, E, A, B, C.

Figure 3.4: Round-Robin Replacement Policy Example
Source: Primary

3.3 Victim Cache

Victim Caching [19] is accomplished by adding a small fully-associative cache that will house any evicted cache lines from the upper level cache. This, in essence, increases the associativity of the upper level cache, eliminating conflict misses. As cache misses incur a miss penalty, reducing the amount of conflict misses results in improved performance. Modern microprocessor systems utilize victim caching throughout various points in the cache hierarchy. For instance, the AMD Opteron [24] contains a 6MB L3 victim cache and the Intel Haswell [25] contains a 128MB L4 victim cache. Despite this, however, NRC-SIM's implementation of victim caching mirrors that of what Jouppi originally proposed, i.e. a small fully associative with at most 16 victim lines. The following figure details the victim cache as it relates to this thesis.

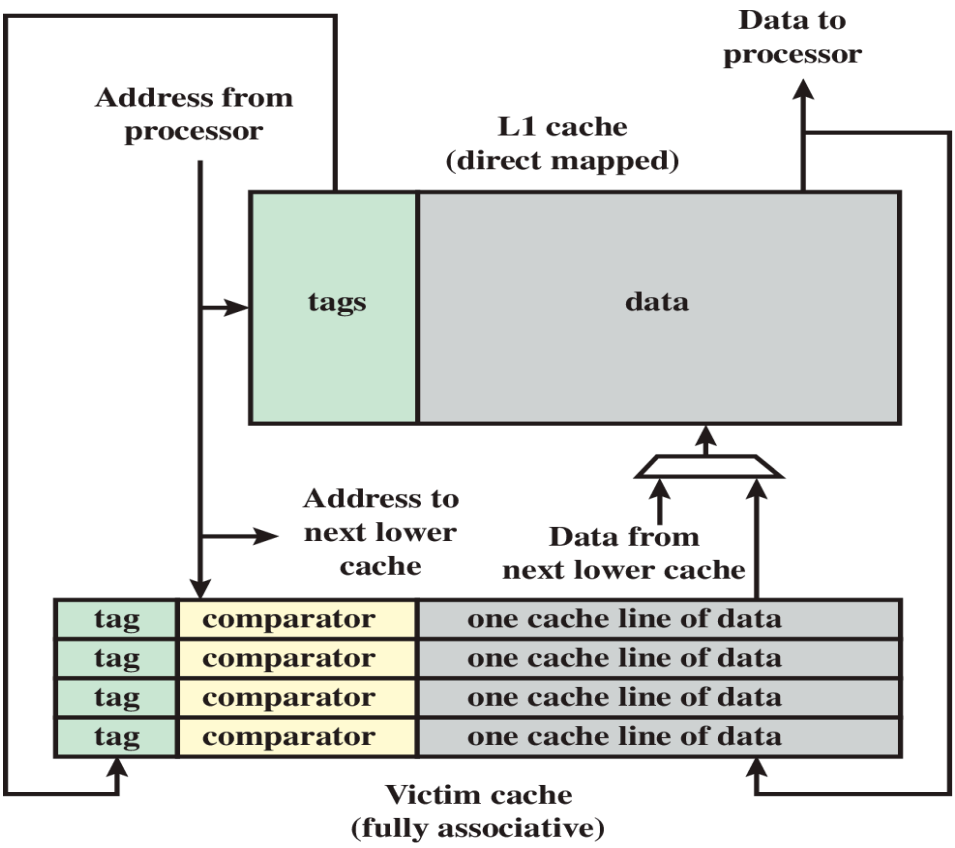

Figure 3.5: Victim Cache Organization

Source: Adapted From [26]

The victim cache follows the exclusive cache model, meaning that data stored within the victim cache is *only* present in the victim cache - data is only loaded into the victim cache via an eviction from the upper level cache. In the event of an L1 miss, the victim cache is searched for the desired data. If there is a hit in the victim cache, the contents of both the upper level cache (L1) and the victim cache are swapped. However, in the event of a victim cache miss, the upper level cache must obtain the desired data from the next lowest level (L2 or memory). The cache block that is evicted from the L1 cache upon updating is stored in the victim cache. If the evicted victim cache block is in the 'M' or 'D' state, the block is stored in the L2 cache or in memory (if no L2 cache is present), with evicted victim cache lines being selected via the Round-Robin replacement policy. As NRC-SIM allows for multi-core and many-core cache simulation, it follows that each core should have its own private victim cache. A visual representation of this can be seen in Figure

3.7.

3.4 Coherence Protocols

When designing a single-core processor, data is either stored in cache or in memory - the same is true for multi-core and many-core systems. However, as accessing memory is orders of magnitude slower than accessing cache, sharing data between each core's private cache grants significant performance benefits. An unfortunate downside to sharing data between multiple private caches is the possibility of having conflicting values, unless, of course, preventative measures are put in place [20]. This issue leads to the necessary implementation of cache coherence protocols - protocols put in place to ensure coherence (at the local level) and consistency (at the global level). To demonstrate this issue, consider the following: Processor P1 reads variable X, which contains the value 1. Processor P2 also reads variable X, which it retrieves from processor P1 via a shared bus. Next, processor P1 writes a value of 0 to variable X. Assuming a write-through cache, processor P1 and memory are both updated to 0, however, processor P2 is left with the incorrect and out-dated value of 1. A visual representation of this example [20] is presented in the following figure.

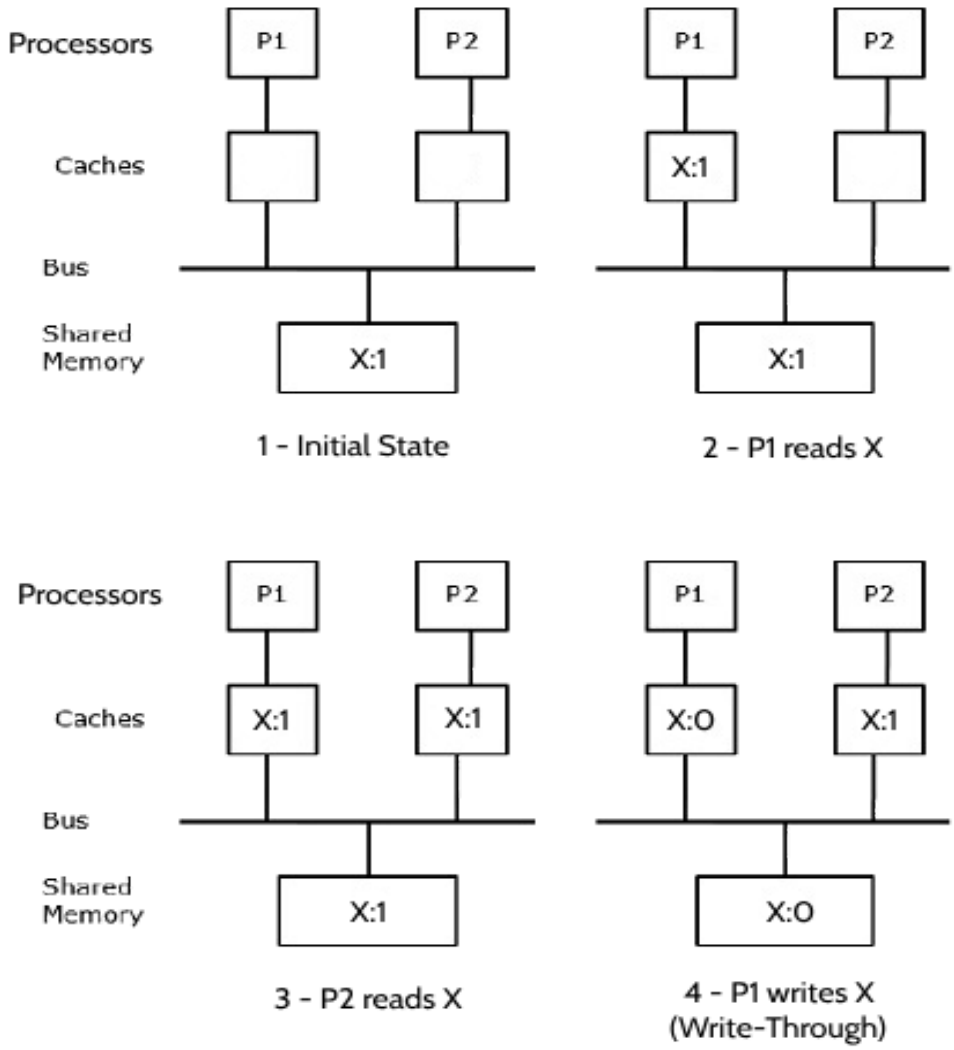

Figure 3.6: The Coherence Problem Example

Source: Adapted From [27]

Naturally, the usage of coherence protocols is only necessary for multi-core and many-core processors. This helps maintain consistency across all cores in the system, eliminating the possibility of an erroneous execution. The previous example demonstrate the coherence problem using two cores, each with a single level of cache. However, as NRC-SIM is more sophisticated, a proper visual is needed to understand the complexity of tackling this issue. The figure below shows, in detail, the many-core memory hierarchy of NRC-SIM.

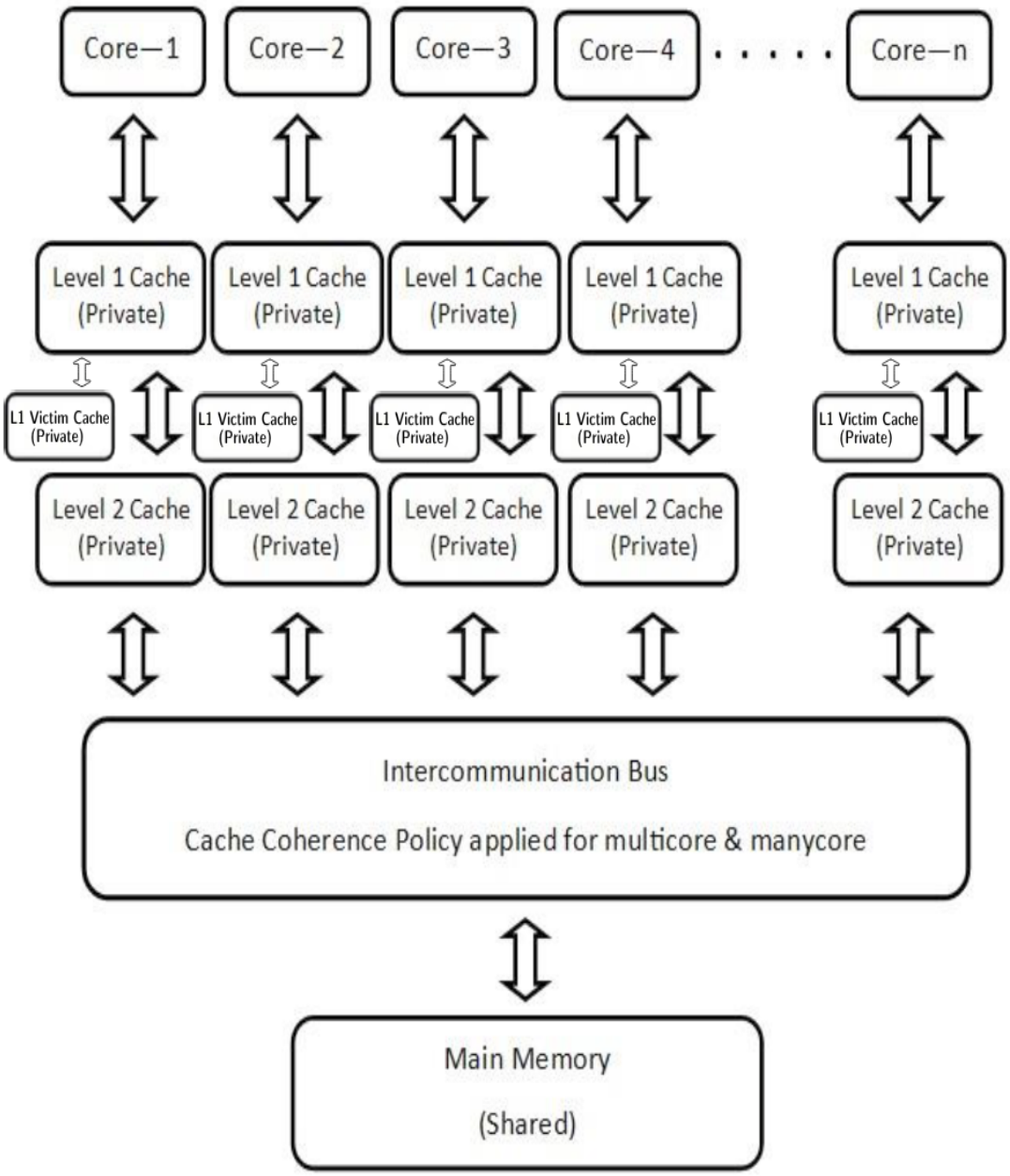

Figure 3.7: Many-Core Memory Hierarchy

Source: Adapted From [10]

3.4.1 Single-Core Cache Simulation

Just as with multi-core and many-core cache simulations, single-core simulations can consist of either a single level of cache, a single level of cache and an L1 victim cache, two levels of cache, or two levels of cache with an L1 victim cache. The following figure shows the possible

single-core configuration used during execution for NRC-SIM.

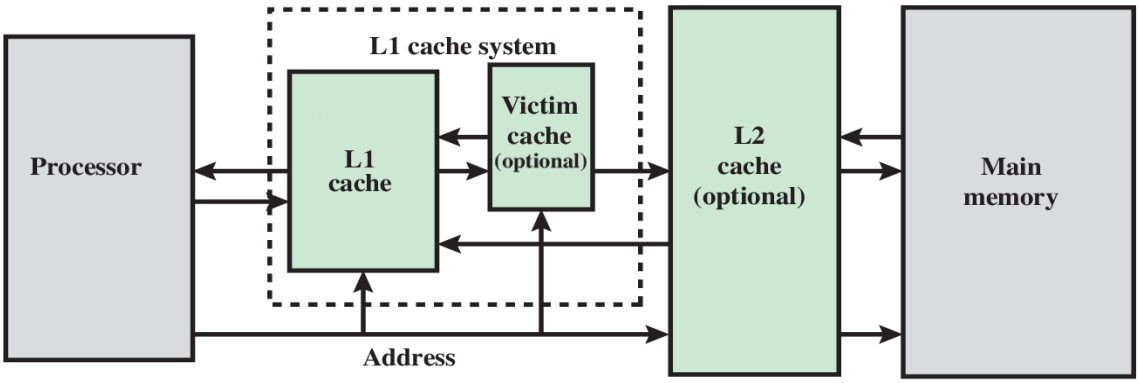

Figure 3.8: Single-Core Memory Hierarchy

Source: Adapted From [26]

When reading the input trace file during the simulation, there are two possible outcomes: a hit or a miss. These outcomes can be further broken down based on instruction type (i.e. read or write), and where in the cache hierarchy they occurred (i.e. L1 miss and L1 victim hit, L1 miss and L1 victim miss, etc.). For instance, in the case of an L1 cache read miss, the requested data is not found in the L1 cache. The next level down is then probed for the requested data. If the data is not found anywhere within the core's cache architecture, it is retrieved from memory. In the case of an L1 cache read hit, the requested data is found within the L1 cache and read by the processor.

Write operations require a write policy to be selected by the user - with their options being either a write-back or a write-through policy. If the user has selected a write-back policy, then in the event of a write hit, the data is only written to the block in the cache, not to memory. This is achieved through the use of a control bit known as a "Dirty Bit". When the cache block and memory contain the same value for a given data, the dirty bit is set to '0'. If there is a write hit, the data in the cache will be updated, setting the dirty bit to '1'. However, when the modified cache block is eventually replaced, it must first update the contents of memory if the dirty bit is set to '1', i.e. the contents of the cache block do not match the contents of memory. Conversely, the write-through policy, updates both the block in the cache *and* the block in the lower-level memory. This

eliminates the need of a dirty bit, however, writing directly to memory every cache write hit results in a significant decline in performance, as the bandwidth increases significantly due to "store traffic exiting the cache" [28]. The figure below illustrates the difference between the write-back policy and the write-through policy.

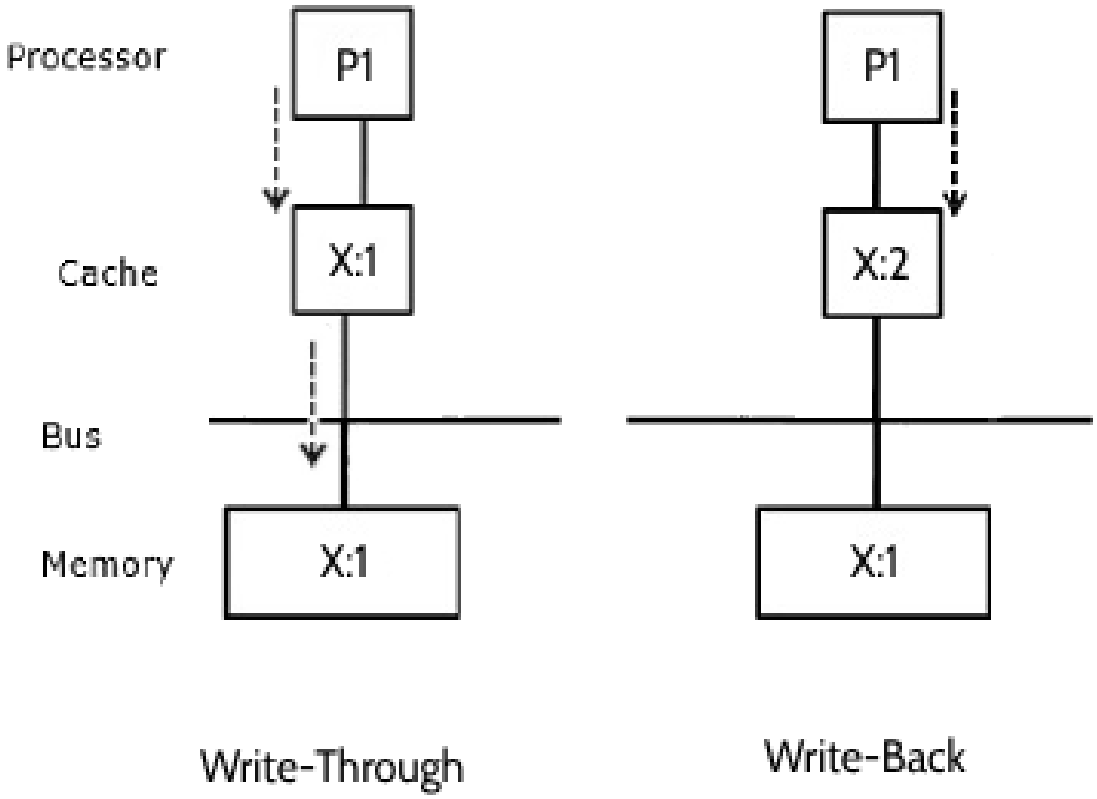

Figure 3.9: Write Policies

Source: Adapted From [27]

3.4.1.1 Write-Back in Single-Core Processors. NRC-SIM features an inclusive cache model with respect to the L1 and L2 caches - that is to say, L1 cache data is always present in the L2 cache. Likewise, in the event of an update to data present in the L1 cache, the corresponding data in L2 will also be updated. In the case of a write-back policy, both levels of cache contain a dirty bit to monitor the status of the cached data. The victim cache, however, does not follow an inclusive cache model - it follows the exclusive cache model. That is, data in the victim cache

can *only* be found in the victim cache, and is not shared elsewhere within the cache architecture. The figure below depicts the write-back policy for a single-core processor containing two levels of cache and a victim cache.

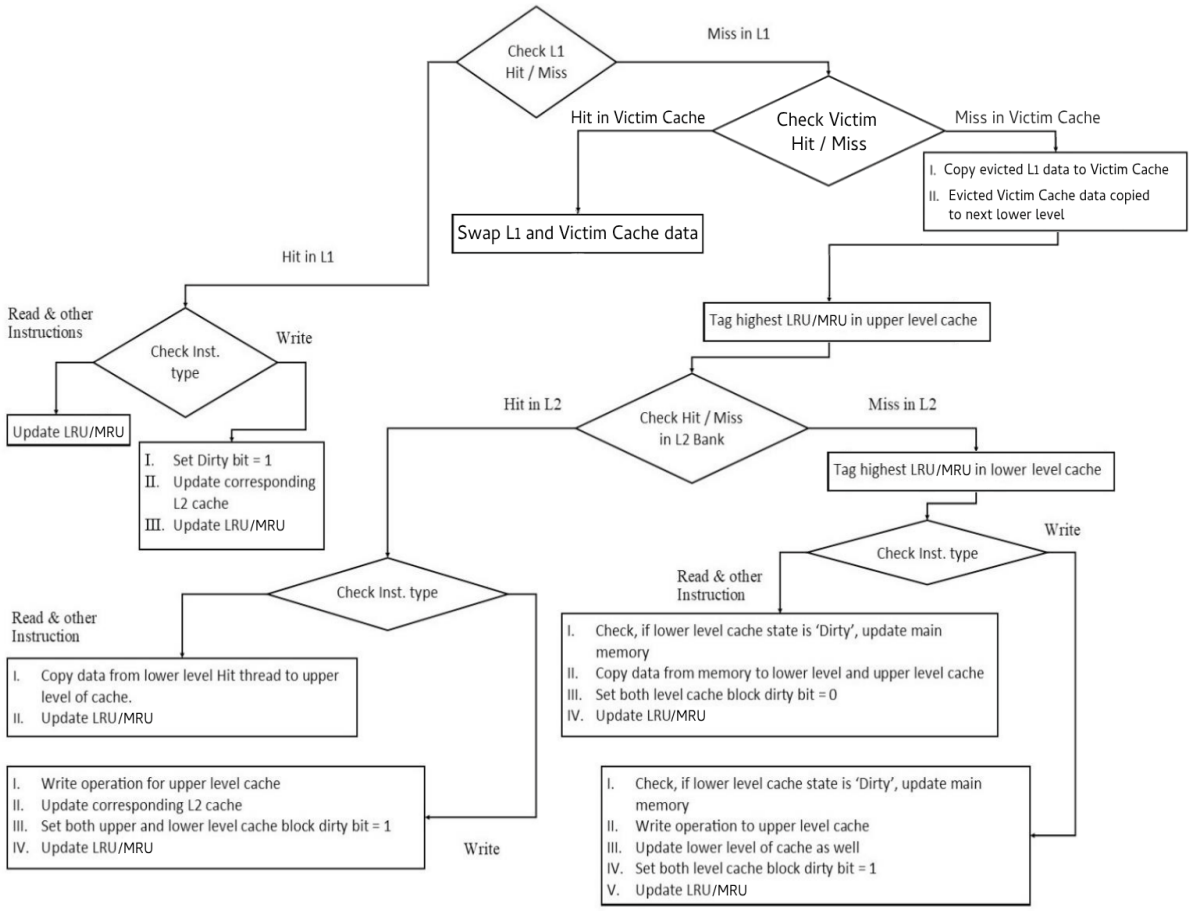

Figure 3.10: Write-Back Policy for Single-Core Processor Containing Two Levels of Cache and a Victim Cache

Source: Adapted From [10]

The diagram above represents the inner-workings of NRC-SIM as it relates to single-core processors containing two levels of cache and a victim cache following a write-back policy. Naturally, this is the most complex a single-core system can be in NRC-SIM. For simpler models, the process remains the same, but lacks the missing component(s) as per the user's desire, i.e. no victim cache, no L2 cache, or no victim cache and no L2 cache. Below is a detailed explanation on the figure above.

1. Hit in L1 Cache

   (a) *Read and Other Instruction*: In the case of an L1 cache read hit, the LRU/MRU is updated, keeping track of which blocks in the cache have been most/least recently hit.

   (b) *Write Instruction*: In the case of an L1 cache write hit, the hit block in both the L1 and the L2 cache is updated - requiring the dirty bit of each to be set to '1'. Additionally, the LRU/MRU values must be updated as well.

2. Miss in L1 Cache, but Hit in Victim Cache

   (a) *Read and Other Instruction*: In the event of a L1 cache miss and a victim cache read hit, the corresponding blocks of data in each are swapped. Once swapped, the LRU/MRU values must be updated.

   (b) *Write Instruction*: In the event of a L1 cache miss and a victim cache write hit, the corresponding blocks of data in each are swapped. Once swapped, the data now stored within the L1 cache is updated, resulting in a dirty bit of '1'. Next, the LRU/MRU values must be updated.

3. Miss in L1 Cache, Miss in Victim Cache, but Hit in L2 Cache

   (a) *Read and Other Instruction*: When a read miss occurs in both the L1 cache and the victim cache, but a read hit occurs in the L2 cache, data will be copied from the L2 cache to the L1 cache to the selected cache bank (based on the selected replacement policies). Once the data is copied into the correct location in the L1 cache, the LRU/MRU values are updated. The evicted L1 data is then stored in the victim cache, with the evicted victim cache data being stored in the L2 cache.

   (b) *Write Instruction*: In the event of a write miss in the L1 cache and in the victim cache, but a write hit in the L2 cache, the new data is written to the selected bank (based on the selected replacement policies) and the corresponding dirty bit is set to '1'. The

LRU/MRU values are then updated. The evicted L1 data is then stored in the victim cache, with the evicted victim cache data being stored in the L2 cache.

4. Miss in L1 Cache, Miss in Victim Cache, and Miss in L2 Cache

   (a) *Read and Other Instructions*: When a read miss occurs at every point in the cache architecture, data is copied to the cache from memory. However, before updating the corresponding banks (based on the selected replacement policy) in both the L1 cache and the L2 cache, the dirty bit must first be checked -

      i. Dirty Cache Block: If the dirty bit is set to '1', the corresponding data must first be written to memory. Once done, the desired data is then copied into the L1 and L2 caches, with the corresponding dirty bit being set to '0'. Evicted L1 data is stored in the victim cache and evicted victim cache data is stored in the L2 cache.

      ii. Clean Cache Block: If the dirty bit is set to '0', the desired data is simply copied to the L1 and L2 caches, with the evicted L1 data being stored in the victim cache and the evicted victim cache data being stored in the L2 cache.

   (b) *Write Instructions*: In the case of a write miss at every point in the cache architecture, data must first be brought into the caches from memory before being updated. However, before doing so, the dirty bit for the corresponding banks (based on the selected replacement policy) in both the L1 cache and the L2 cache, must first be checked -

      i. Dirty Cache Block: If the dirty bit is set to '1', the corresponding data must first be written to memory, changing the dirty bit to '0'. Once done, the desired data is copied from memory to each of the caches, and then updated, once again changing the dirty bit to '1'. Evicted L1 data is stored in the victim cache and evicted victim cache data is stored in the L2 cache.

      ii. Clean Cache Block: If the dirty bit is set to '0', the desired data is copied from memory to each of the caches. Once in the caches, the data is updated accordingly, resulting in the dirty bit being changed to '1'. Evicted L1 data is stored in the

victim cache and evicted victim cache data is stored in the L2 cache.

Once complete, the LRU/MRU values are updated.

For the above, in every case of a hit/miss, a counter corresponding to the location of the hit/miss (i.e. L1 cache, etc.) and the instruction type, is incremented. The variables holding these values are then used to calculate the miss rate of each level for the simulated system, as well as a global total.

3.4.1.2 Write-Through in Single-Core Processors. Unlike the write-back policy, the write-through policy has no concept of a dirty bit. Write operations simultaneously update the cached data and memory, ensuring data is always up to date at every point in the memory hierarchy. The write-through policy is relatively simple to implement, however, as previously mentioned, write-through caches incur significant memory traffic, resulting in wasted computational time [28]. The following figure illustrates the write-through policy for a single-core processor containing two levels of cache and a victim cache.

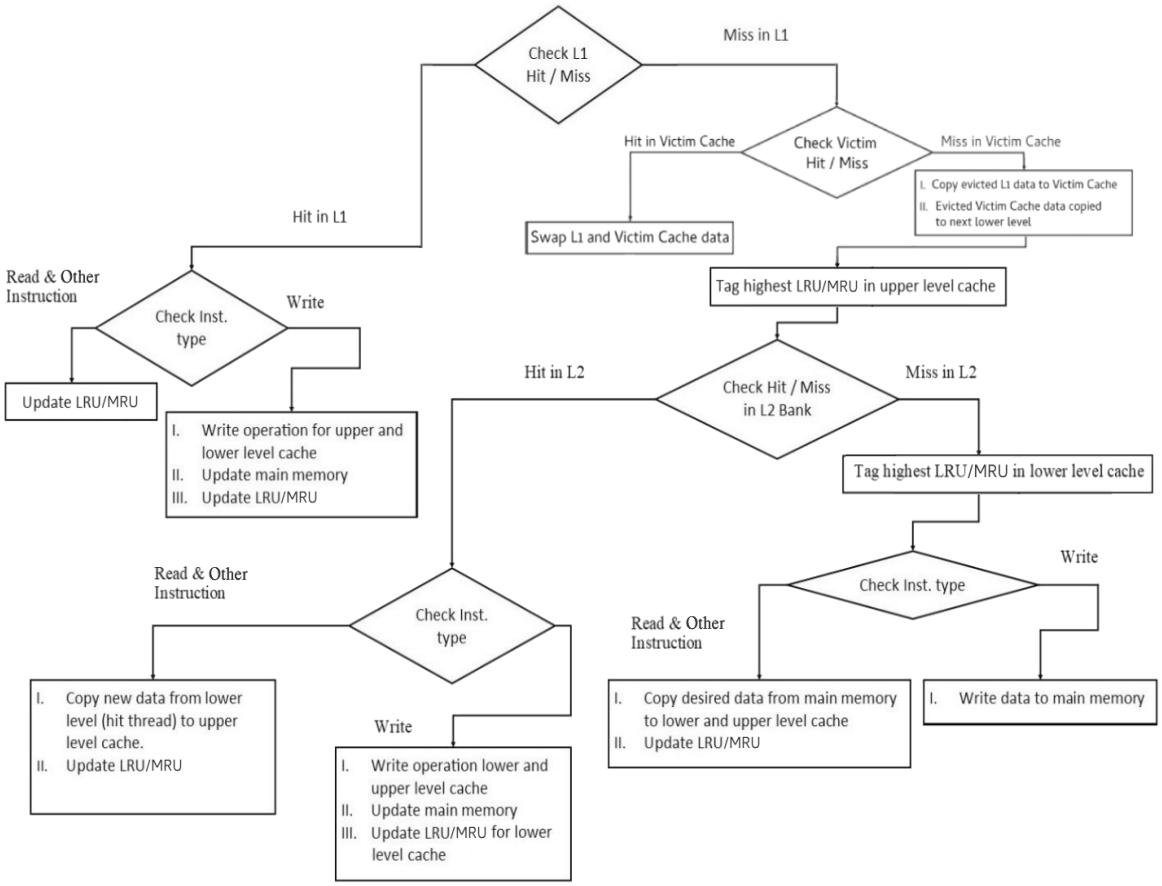

Figure 3.11: Write-Through Policy for Single-Core Processor Containing Two Levels of Cache and a Victim Cache

Source: Adapted From [10]

Below is a detailed description of the preceding figure on single-core processors containing two levels of cache and a victim cache using the write-through write policy.

1. Hit in L1 Cache

   (a) *Read and Other Instruction*: In the case of an L1 read hit, the LRU/MRU values are updated.

   (b) *Write Instruction*: In the case of an L1 write hit, both levels of cache are updated, as is memory and the LRU/MRU values.

2. Miss in L1 Cache, but Hit in Victim Cache

   (a) *Read and Other Instruction*: In the event of a L1 cache miss and a victim cache read hit, the corresponding blocks of data in each are swapped. Once swapped, the LRU/MRU values must be updated.

   (b) *Write Instruction*: In the event of a L1 cache miss and a victim cache write hit, the corresponding blocks of data in each are swapped. Once swapped, the data now stored within the L1 cache is updated, as is the L2 cache and memory. Next, the LRU/MRU values must be updated.

3. Miss in L1 Cache, Miss in Victim Cache, but Hit in L2 Cache

   (a) *Read and Other Instruction*: When a read miss occurs in both the L1 cache and the victim cache, but a read hit occurs in the L2 cache, data will be copied from the L2 cache to the L1 cache to the selected cache bank (based on the selected replacement policies). Once the data is copied into the correct location in the L1 cache, the LRU/MRU values are updated. The evicted L1 data is then stored in the victim cache, with the evicted victim cache data being stored in the L2 cache.

   (b) *Write Instruction*: In the event of a write miss in the L1 cache and in the victim cache, but a write hit in the L2 cache, the new data is written to the selected bank (based on the selected replacement policies) in both levels of cache, as well as memory. The LRU/MRU values are then updated. The evicted L1 data is then stored in the victim cache, with the evicted victim cache data being stored in the L2 cache.

4. Miss in L1 Cache, Miss in Victim Cache, and Miss in L2 Cache

   (a) *Read and Other Instructions*: When a read miss occurs at every point in the cache architecture, data is copied to both levels of the cache from memory. The LRU/MRU values are then updated.

(b) *Write Instructions*: In the case of a write miss at every point in the cache architecture, data is only updated in memory, removing the need to first load it into the cache. The LRU/MRU values are then updated.

For the above, in every case of a hit/miss, a counter corresponding to the location of the hit/miss (i.e. L1 cache, etc.) and the instruction type, is incremented. The variables holding these values are then used to calculate the miss rate for each level in the simulated system, as well as a global total.

3.4.2 Multi-Core and Many-Core Cache Simulation

When dealing with multi-core and many-core systems, the consistency of data is of the utmost importance. All caches within these systems must maintain the most up-to-date data - failing to do so would result in erroneous execution and flawed results. To ensure proper coherence and consistency, protocols are put in place that determine the state of a given block of data. There are two classes of protocols: *Directory-Based* and *Snooping* [20]. Directory-based protocols manage the sharing status of a particular block of physical memory in one centralized location. In snooping coherence protocols, every cache containing a given copy of data stores the sharing status of said block. A given processor will monitor, or *snoop*, a shared bus to check if another processor has a copy. In regards to the latter, NRC-SIM supports three such protocols.

3.4.2.1 MSI Coherence Protocol. The MSI [20] coherence protocol has three states that can be assigned to a given cache line - Modified, Shared, and Invalid. The Modified state is assigned to a cache line that has recently been updated, resulting in its dirty bit being set to '1' - this data is the only copy in the cache and memory. The Shared state is assigned to any cache block that is shared by one or more caches *and* memory. In this instance, the dirty bit is equal to '0'. Lastly, the Invalid state is used to represent a cache block that contains no data. Below is a detailed explanation of each state's possible transitions, followed by a diagram illustrating as such for both processor requests and bus requests.

1. *Modified State*

    (a) CPU

        i. Read Hit: No change to state.

        ii. Write Hit/Miss: No change to state.

    (b) BUS

        i. Read Hit: State is changed to ‘S’.

        ii. Write Hit: State is changed to ‘I’.

2. *Shared State*

    (a) CPU

        i. Read Hit/Miss: No change to state.

        ii. Write Hit: State is changed to ‘M’.

    (b) BUS

        i. Read Hit/Miss: No change to state.

        ii. Write Hit: State is changed to ‘I’.

3. *Invalid State*

    (a) CPU

        i. Read Miss: State is changed to ‘S’.

        ii. Write Miss: State is changed to ‘M’.

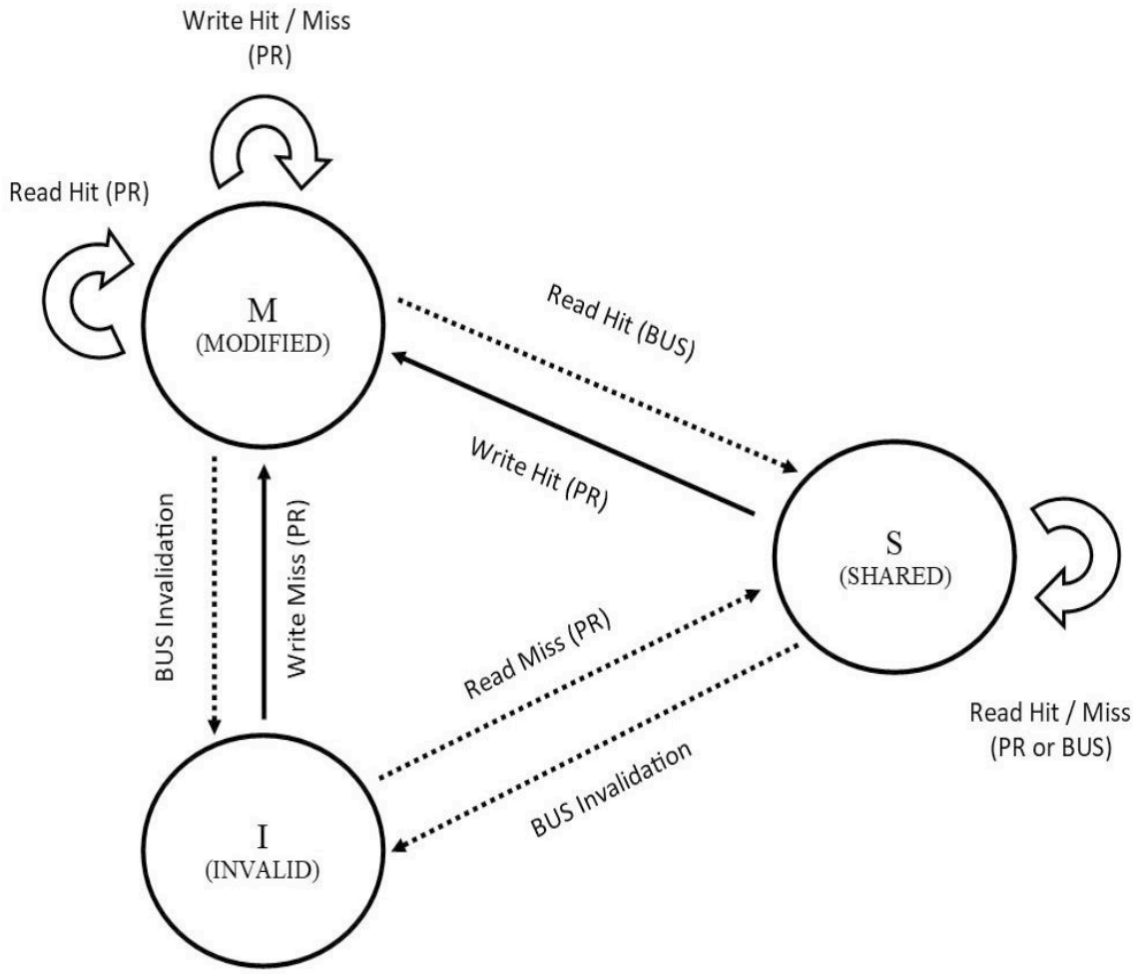

Figure 3.12: MSI State Transitions: CPU Requests and BUS Requests

Source: [10]

As NRC-SIM allows up to two levels of cache and an L1 victim cache, it is important to understand how coherence is handled in such a configuration. To implement the Invalid state, a binary control bit is used - with a valid bit value of '1' representing valid, accessible data, and a valid bit value of '0' representing an invalid cache block, either because there is no data or because the cache block received an invalidation signal from another processor due to an update to said data. What follows is an explanation of how the MSI coherence protocol works relative to the previously mentioned cache configuration, i.e. two levels of cache and a victim cache.

1. Hit in Private L1 Cache

   (a) *Read and Other Instruction*: In the case of an L1 read hit, the state of the cache block remains unchanged.

   (b) *Write Instruction*: In the case of an L1 write hit, the corresponding cache block will receive new data, changing its state to 'M'. In the case that the cache block was previously in the 'S' state, an invalidation signal will be sent to the bus, setting the valid bit of any other cache block with state 'S' to '0'. Once updated in the L1 cache, the corresponding copy residing in the L2 cache must also be updated. The LRU/MRU values are then updated.

2. Miss in L1 Cache, but Hit in Victim Cache

   (a) *Read and Other Instruction*: In the event of a L1 cache miss and a victim cache read hit, the corresponding blocks of data in each are swapped and the state of each remains unchanged. Once swapped, the LRU/MRU values must be updated.

   (b) *Write Instruction*: In the event of a L1 cache miss and a victim cache write hit, the corresponding blocks of data in each are swapped. Once swapped, the data now stored within the L1 cache is updated, changing the block's state to 'M'. An invalidation signal is then sent through the bus for any corresponding cache blocks. The copy stored in the L2 cache is then updated, as are the LRU/MRU values.

3. Miss in L1 Cache, Miss in Victim Cache, but Hit in L2 Cache

   (a) *Read and Other Instruction*: When a read miss occurs in both the L1 cache and the victim cache, but a read hit occurs in the L2 cache, data will be copied from the L2 cache to the L1 cache to the selected cache bank (based on the selected replacement policies). The L1 cache block will receive the same state as its L2 counterpart. Once the data is copied into the correct location in the L1 cache, the LRU/MRU values are

updated. The evicted L1 data is then stored in the victim cache, with the evicted victim cache data being stored in the L2 cache.

(b) *Write Instruction*: In the event of a write miss in the L1 cache and in the victim cache, but a write hit in the L2 cache, the new data is written to the selected bank (based on the selected replacement policies) in both levels of cache, setting the state of each to 'M'. An invalidation signal is then sent through the bus and the LRU/MRU values are updated. The evicted L1 data is stored in the victim cache, and the evicted victim cache is stored in the L2 cache.

4. Miss in L1 Cache, Miss in Victim Cache, Miss in L2 Cache, and Hit in Bus

(a) *Read and Other Instructions*: When a read miss occurs in every level of a core's private cache, but a hit occurs in the bus, i.e. in another core's private cache, data is copied to both levels of the requesting core's cache. If the read hit data from the bus is in the 'M' state, memory must first be updated. Once memory has been updated and the requesting core has received its copies of the data, the state of both is changed to 'S'. The LRU/MRU values are then updated.

(b) *Write Instructions*: In the case of a write miss in every level of a core's private cache, but a hit in the bus, i.e. in another core's private cache, the requesting core's private L1 cache is updated and its state is set to 'M'. The requesting core's private L2 cache is updated accordingly. An invalidation signal is then sent through the bus for any copy that may reside in another core. The LRU/MRU values are then updated.

5. Miss in L1 Cache, Miss in Victim Cache, Miss in L2 Cache, and Miss in Bus

(a) *Read and Other Instructions*: When a read miss occurs at every level within the cache architecture, data is copied to both levels of the requesting core's cache from memory. Before doing so, however, the state of the corresponding L2 cache block must first be checked. If it is in the 'M' state, the corresponding location in memory must be

updated. Once both levels of the cache have been updated from memory, the state of each is changed to 'S'. The LRU/MRU values are then updated.

(b) *Write Instructions*: In the case of a write miss at every level within the cache architecture, the L1 cache is updated with the new data, setting its state to 'M'. The L2 cache is then updated accordingly. Lastly, the LRU/MRU values are then updated.

For the above, in every case of a hit/miss, a counter corresponding to the location of the hit/miss (i.e. L1 cache, etc.) and the instruction type, is incremented. The variables holding these values are then used to calculate the miss rate of each level for every core in the system, as well as the global miss rate for the simulated system. Below is a visual representation of the MSI coherence protocol as described above.

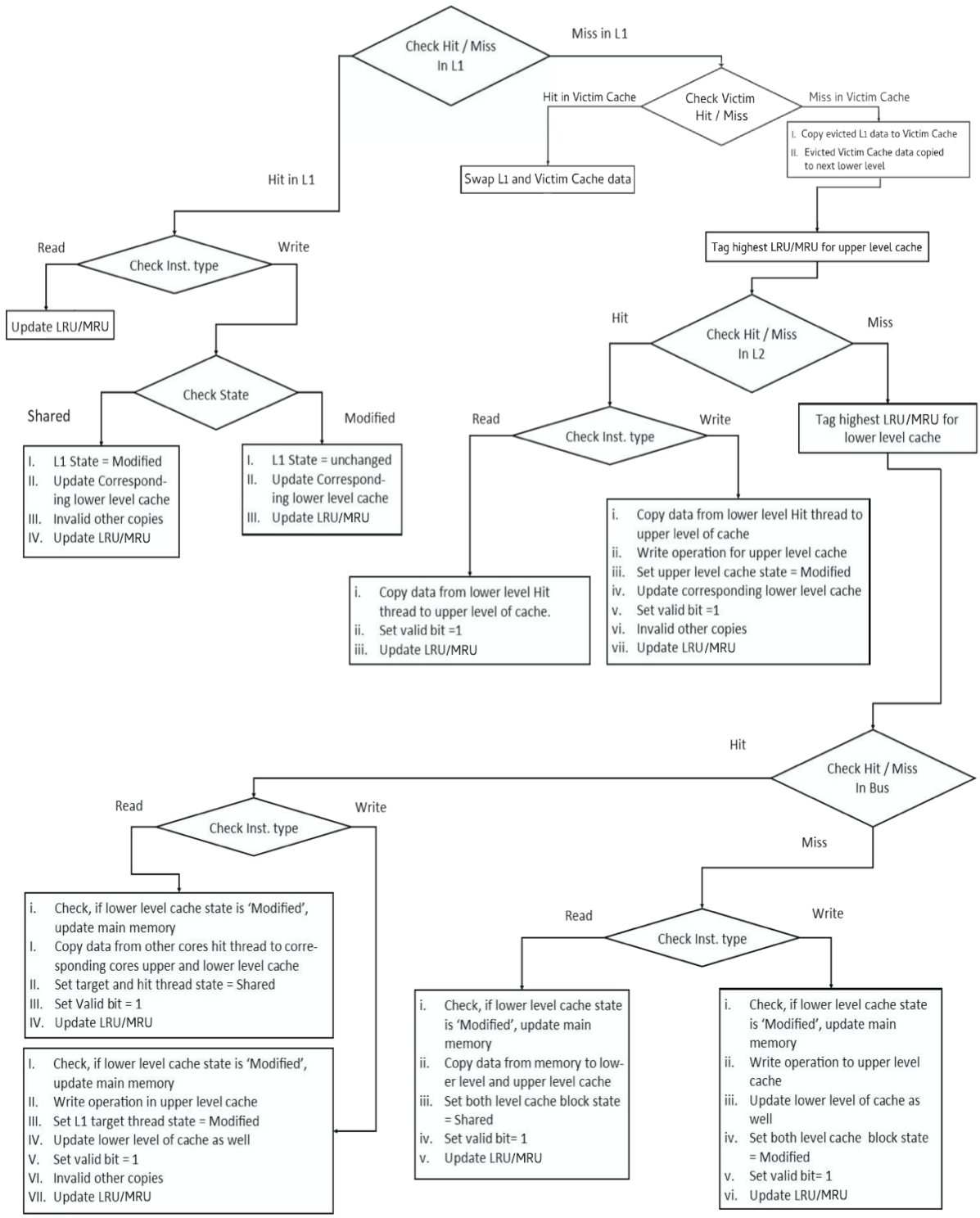

Figure 3.13: MSI Coherence Protocol for Two Levels of Cache and a Victim Cache

Source: Adapted From [10]

3.4.2.2 MESI Coherence Protocol. The MESI [20] coherence protocol builds upon the MSI coherence protocol by adding an additional state - Exclusive. In MESI, the Modified state represents a cache line that has been modified, resulting in the copy in memory being outdated, i.e. a dirty bit of '1'. This data is the only copy within the cache. A cache block in the Exclusive state has data that is the same as memory, i. e. a dirty bit of '0', and is the only cached copy. The Shared state represents a cache block that is the same as memory, i.e. a dirty bit of '0', with copies possibly existing in other caches. Lastly , the Invalid state refers to a cache line that contains no data. Below is a detailed explanation of each state's possible transitions, followed by a diagram illustrating as such for both processor requests and bus requests.

1. *Modified State*
   (a) CPU
      i. Read/Write Hit: No change to state.
   (b) BUS
      i. Read Hit: State is changed to 'S'.
      ii. Read Miss: State is changed to 'E'.
      iii. Write Hit: State is changed to 'I'.
2. *Exclusive State*
   (a) CPU
      i. Read Hit: No change to state.
      ii. Write Hit: State is changed to 'M'.
   (b) BUS
      i. Read Hit: State is changed to 'S'.
      ii. Write Hit: State is changed to 'I'.

3. *Shared State*

   (a) CPU

      i. Read Hit: No change to state.

      ii. Write Hit: State is changed to ‘M’.

   (b) BUS

      i. Write Hit: State is changed to ‘I’.

4. *Invalid State*

   (a) CPU

      i. Read Miss: State is changed to ‘E’.

      ii. Write Miss: State is changed to ‘M’.

   (b) BUS

      i. Read Miss: State is changed to ‘S’.

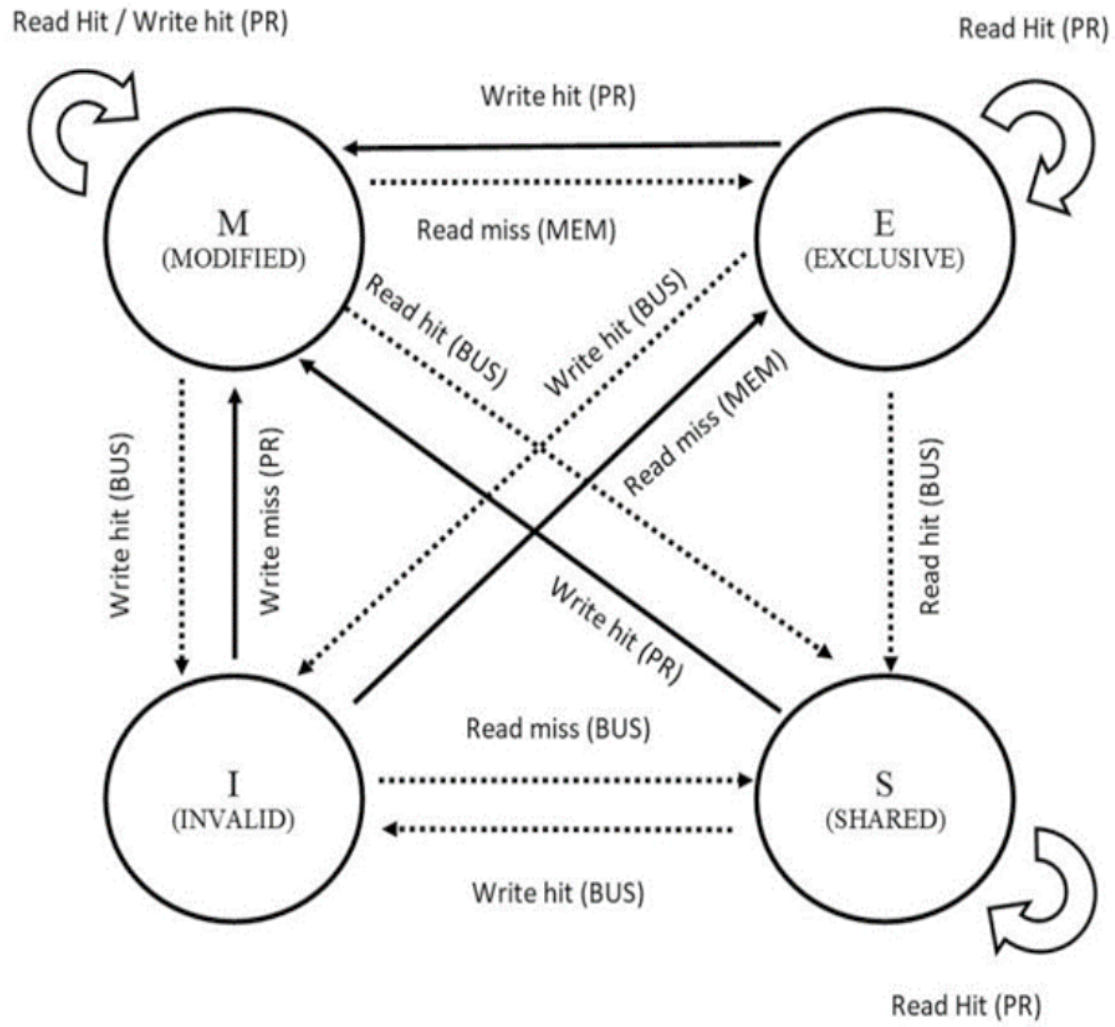

Figure 3.14: MESI State Transitions: CPU Requests and BUS Requests

Source: [10]

As NRC-SIM allows up to two levels of cache and an L1 victim cache, it is important to understand how coherence is handled in such a configuration. To implement the Invalid state, a binary control bit is used - with a valid bit value of '1' representing valid, accessible data, and a valid bit value of '0' representing an invalid cache block, either because there is no data or because the cache block received an invalidation signal from another processor due to an update to said data. What follows is an explanation of how the MESI coherence protocol works relative to the previously mentioned cache configuration, i.e. two levels of cache and a victim cache.

1. Hit in Private L1 Cache

   (a) *Read and Other Instruction*: In the case of an L1 read hit, the state of the cache block remains unchanged.

   (b) *Write Instruction*: In the case of an L1 write hit, the corresponding cache block will receive new data, changing its state to 'M'. In the case that the cache block was previously in the 'S' state, an invalidation signal will be sent to the bus, setting the valid bit of any other cache block with state 'S' to '0'. Once updated in the L1 cache, the corresponding copy residing in the L2 cache must also be updated. The LRU/MRU values are then updated.

2. Miss in L1 Cache, but Hit in Victim Cache

   (a) *Read and Other Instruction*: In the event of a L1 cache miss and a victim cache read hit, the corresponding blocks of data in each are swapped and the state of each remains unchanged. Once swapped, the LRU/MRU values must be updated.

   (b) *Write Instruction*: In the event of a L1 cache miss and a victim cache write hit, the corresponding blocks of data in each are swapped. Once swapped, the data now stored within the L1 cache is updated, changing the block's state to 'M'. An invalidation signal is then sent through the bus for any corresponding cache blocks. The copy stored in the L2 cache is then updated, as are the LRU/MRU values.

3. Miss in L1 Cache, Miss in Victim Cache, but Hit in L2 Cache

   (a) *Read and Other Instruction*: When a read miss occurs in both the L1 cache and the victim cache, but a read hit occurs in the L2 cache, data will be copied from the L2 cache to the L1 cache to the selected cache bank (based on the selected replacement policies). The L1 cache block will receive the same state as its L2 counterpart. Once the data is copied into the correct location in the L1 cache, the LRU/MRU values are

updated. The evicted L1 data is then stored in the victim cache, with the evicted victim cache data being stored in the L2 cache.

(b) *Write Instruction*: In the event of a write miss in the L1 cache and in the victim cache, but a write hit in the L2 cache, the new data is written to the selected bank (based on the selected replacement policies) in both levels of cache, setting the state of each to 'M'. An invalidation signal is then sent through the bus and the LRU/MRU values are updated. The evicted L1 data is stored in the victim cache, and the evicted victim cache is stored in the L2 cache.

4. Miss in L1 Cache, Miss in Victim Cache, Miss in L2 Cache, and Hit in Bus

(a) *Read and Other Instructions*: When a read miss occurs in every level of a core's private cache, but a hit occurs in the bus, i.e. in another core's private cache, data is copied to both levels of the requesting core's cache. If the read hit data from the bus is in the 'M' state, memory must first be updated. Once memory has been updated and the requesting core has received its copies of the data, the state of both is changed to 'S'. The LRU/MRU values are then updated.

(b) *Write Instructions*: In the case of a write miss in every level of a core's private cache, but a hit in the bus, i.e. in another core's private cache, the requesting core's private L1 cache is updated and its state is set to 'M'. The requesting core's private L2 cache is updated accordingly. An invalidation signal is then sent through the bus for any copy that may reside in another core. The LRU/MRU values are then updated.

5. Miss in L1 Cache, Miss in Victim Cache, Miss in L2 Cache, and Miss in Bus

(a) *Read and Other Instructions*: When a read miss occurs at every level within the cache architecture, data is copied to both levels of the requesting core's cache from memory. Before doing so, however, the state of the corresponding L2 cache block must first be checked. If it is in the 'M' state, the corresponding location in memory must be

updated. Once both levels of the cache have been updated from memory, the state of each is changed to 'E'. The LRU/MRU values are then updated.

(b) *Write Instructions*: In the case of a write miss at every level within the cache architecture, the L1 cache is updated with the new data, setting its state to 'M'. The L2 cache is then updated accordingly. Lastly, the LRU/MRU values are then updated.

For the above, in every case of a hit/miss, a counter corresponding to the location of the hit/miss (i.e. L1 cache, etc.) and the instruction type, is incremented. The variables holding these values are then used to calculate the miss rate of each level for every core in the system, as well as the global miss rate for the simulated system. Below is a visual representation of the MESI coherence protocol as described above.

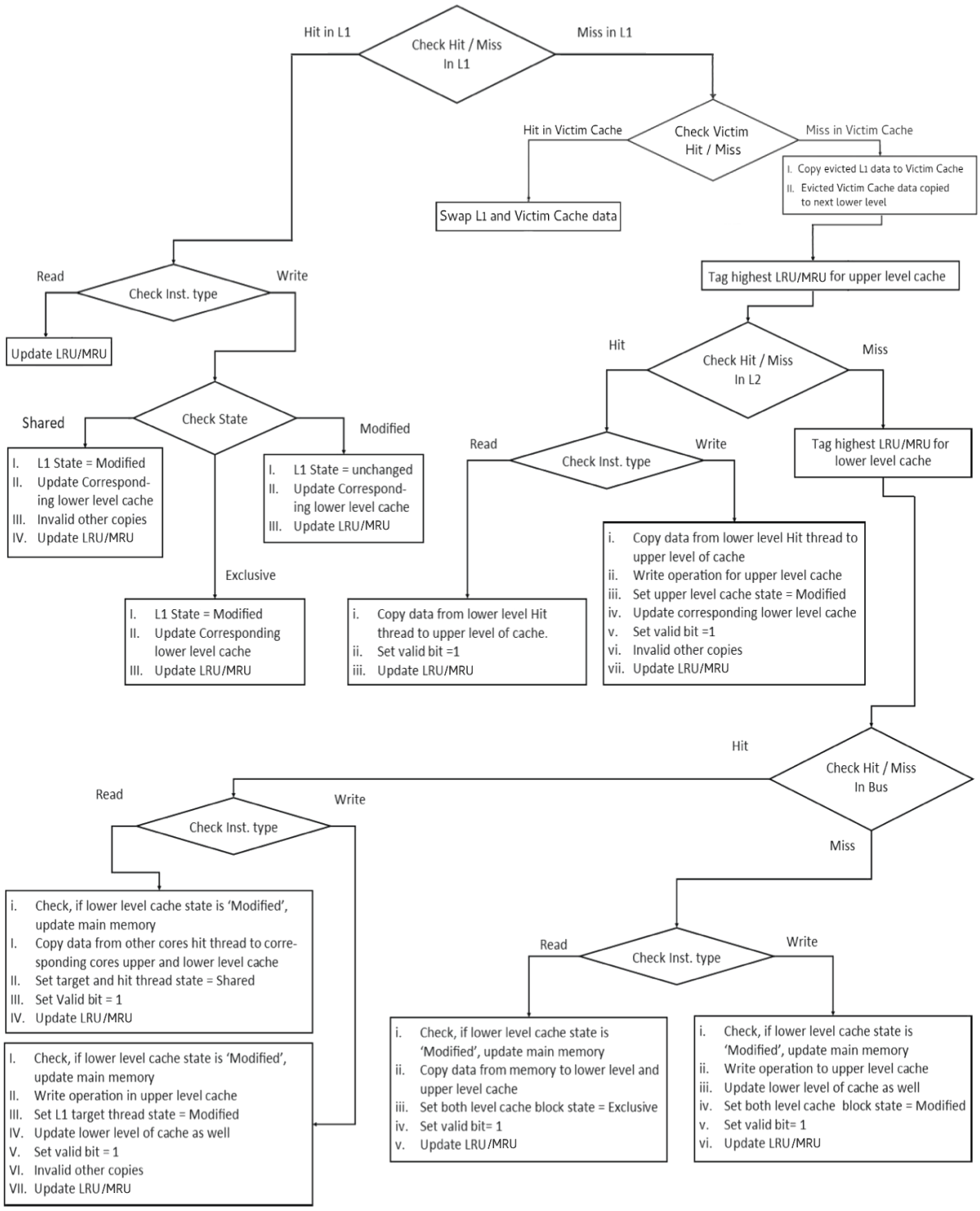

Figure 3.15: MESI Coherence Protocol for Two Levels of Cache and a Victim Cache

Source: Adapted From [10]

3.4.2.3 FIREFLY Coherence Protocol. The FIREFLY [29] coherence protocol consists of three states that can be assigned to a given cache line - Valid-Exclusive, Shared, and Dirty. Cache blocks in the Valid-Exclusive state are both valid and the only copy of data that exists in the cache. Cache blocks in the Shared Stated are both valid and may be found in other cores caches. Cache blocks that are in the Dirty state have been modified and have not updated memory.

FIREFLY utilizes a control bit C, or CopiesExist, to identify the transitions between states. When CopiesExist is set to '1', a copy (or copies) of a given cache block exist in the bus. When CopiesExist is set to '0', no such copies exist [30]. Below is a detailed explanation of each state's possible transitions, followed by a diagram illustrating as such for both processor requests and bus requests.

1. *Valid-Exclusive State*
   (a) CPU
      i. Read Miss and No Other Copies Exist: No change to state.
      ii. Write Hit: State is changed to 'D'.
   (b) BUS
      i. Read/Write Request: State is changed to 'S'.
2. *Shared State*
   (a) CPU
      i. Read/Write Miss and No Other Copies Exist: No change to state.
      ii. Write Hit and Other Copies Exist: No change to state.
      iii. Write Hit and No Other Copies Exist: State is changed to 'V'.
   (b) BUS
      i. Read/Write Request: No change to state.

3. *Dirty State*

(a) CPU

i. Write Hit: No change to state.

ii. Write Miss and No Other Copies Exist: No change to state.

(b) BUS

i. Read/Write Request: State is changed to 'S'.

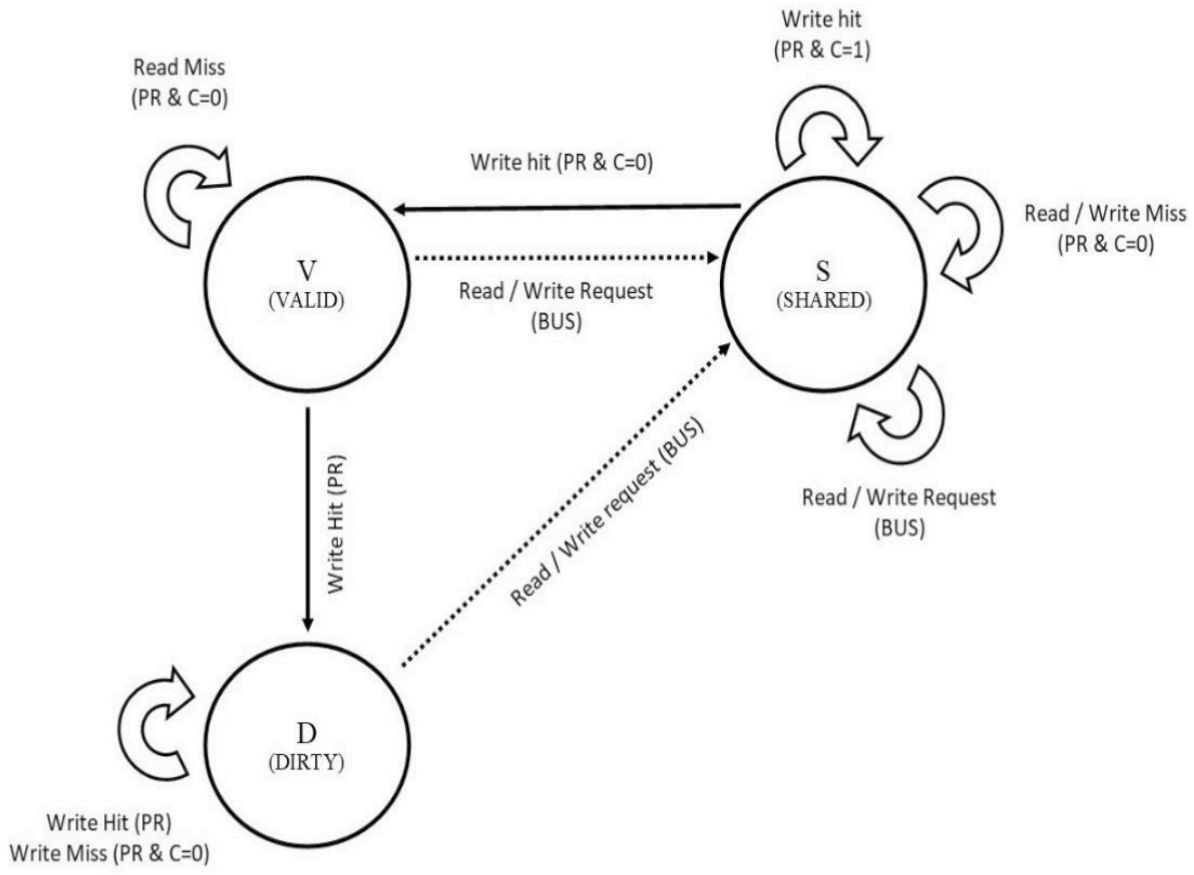

Figure 3.16: FIREFLY State Transitions: CPU Requests and BUS Requests

Source: [10]

As NRC-SIM allows up to two levels of cache and an L1 victim cache, it is important to understand how coherence is handled in such a configuration. Each state transition requires the use of the CopiesExist bit in conjunction with each processor/bus request. What follows is an

explanation of how the FIREFLY coherence protocol works relative to the previously mentioned cache configuration, i.e. two levels of cache and a victim cache.

1. Hit in Private L1 Cache

   (a) *Read and Other Instruction*: In the case of an L1 read hit, the state of the cache block remains unchanged.

   (b) *Write Instruction*: In the case of an L1 write hit, the corresponding cache block will receive new data. If the state of the cache block is either 'D' or 'V', the cache block will be updated and its state will be set to 'D'. If the state of the cache block is 'S', and other copies exist in the cache, the state of the cache block will be left unchanged. If no other copies exist, the state will change to 'D'. Once updated in the L1 cache, the corresponding copy residing in the L2 cache must also be updated. The LRU/MRU values are then updated.

2. Miss in L1 Cache, but Hit in Victim Cache

   (a) *Read and Other Instruction*: In the event of a L1 cache miss and a victim cache read hit, the corresponding blocks of data in each are swapped and the state of each remains unchanged. Once swapped, the LRU/MRU values must be updated.

   (b) *Write Instruction*: In the event of a L1 cache miss and a victim cache write hit, the corresponding blocks of data in each are swapped. Once swapped, the data now stored within the L1 cache is updated. If the state of the cache block is either 'D' or 'V', the cache block will be updated and its state will be set to 'D'. If the state of the cache block is 'S', and other copies exist in the cache, the state of the cache block will be left unchanged. If no other copies exist, the state will change to 'D'. The copy stored in the L2 cache is then updated, as are the LRU/MRU values.

3. Miss in L1 Cache, Miss in Victim Cache, but Hit in L2 Cache

   (a) *Read and Other Instruction*: When a read miss occurs in both the L1 cache and the victim cache, but a read hit occurs in the L2 cache, data will be copied from the L2 cache to the L1 cache to the selected cache bank (based on the selected replacement policies). The L1 cache block will receive the same state as its L2 counterpart. Once the data is copied into the correct location in the L1 cache, the LRU/MRU values are updated. The evicted L1 data is then stored in the victim cache, with the evicted victim cache data being stored in the L2 cache.

   (b) *Write Instruction*: In the event of a write miss in the L1 cache and in the victim cache, but a write hit in the L2 cache, the new data is written to the selected bank (based on the selected replacement policies) in both levels of cache, setting the state of each to 'S' if other copies of the data exist. If no other copies exist, the state of each is set to 'D'. The LRU/MRU values are then updated, and the evicted L1 data is stored in the victim cache, and the evicted victim cache is stored in the L2 cache.

4. Miss in L1 Cache, Miss in Victim Cache, Miss in L2 Cache, and Hit in Bus

   (a) *Read and Other Instructions*: When a read miss occurs in every level of a core's private cache, but a hit occurs in the bus, i.e. in another core's private cache, data is copied to both levels of the requesting core's cache. If the read hit data from the bus is in the 'D' state, memory must first be updated. Once memory has been updated and the requesting core has received its copies of the data, the state of both is changed to 'S'. The LRU/MRU values are then updated.

   (b) *Write Instructions*: In the case of a write miss in every level of a core's private cache, but a hit in the bus, i.e. in another core's private cache, the requesting core's private L1 cache is updated and its state is set to 'D'. The requesting core's private L2 cache is updated accordingly. The state of both the requesting cache and the supplying cache (i.e. bus write hit) is set to 'S'. The LRU/MRU values are then updated.

5. Miss in L1 Cache, Miss in Victim Cache, Miss in L2 Cache, and Miss in Bus

   (a) *Read and Other Instructions*: When a read miss occurs at every level within the cache architecture, data is copied to both levels of the requesting core's cache from memory. Once both levels of the cache have been updated from memory, the state of each is changed to 'V'. The LRU/MRU values are then updated.

   (b) *Write Instructions*: In the case of a write miss at every level within the cache architecture, the L1 cache is updated with the new data, setting its state to 'D'. The L2 cache is then updated accordingly. Lastly, the LRU/MRU values are then updated.

For the above, in every case of a hit/miss, a counter corresponding to the location of the hit/miss (i.e. L1 cache, etc.) and the instruction type, is incremented. The variables holding these values are then used to calculate the miss rate of each level for every core in the system, as well as the global miss rate for the simulated system. Below is a visual representation of the FIREFLY coherence protocol as described above.

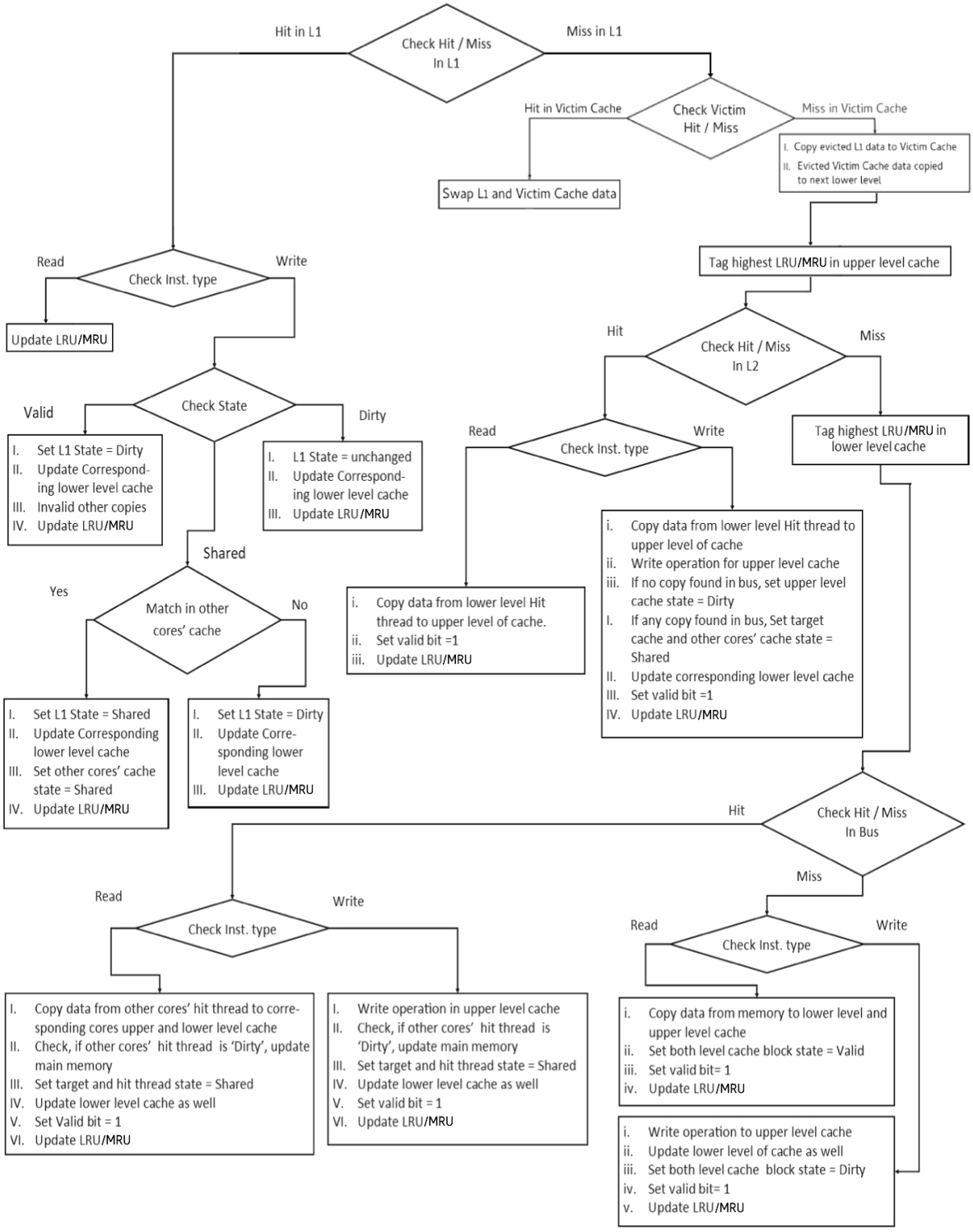

Figure 3.17: FIREFLY Coherence Protocol for Two Levels of Cache and a Victim Cache

Source: Adapted From [10]

3.5 Node-RED Trace Files

NRC-SIM uses trace files of benchmark programs as input, making them an essential part of this thesis. These trace files are generated using Intel's Pin [11] and benchmark programs found in the PARSEC [16] and SPLASH-2 [18] Benchmark Suites. Once a trace file has been generated, it can be used to run the cache simulation.

Upon completion of a simulation, NRC-SIM displays various results for each core's private cache performance, as well as a global total. Examples include the number of reads per core, the number of read hits per core, the number of read misses per core, etc. per cache level, i.e. L1 cache, L2 cache, and victim cache. Additionally, bus traffic is also displayed, enabling users to gain a holistic view of the simulated cache's performance.

When determining miss rates, each core's private caches are considered. The following formulas are used to find each cores L1 and L2 miss rate.

$$Miss\ Rate_{L1} = (\frac{Total\ Core\ Misses_{L1}}{Total\ Core\ Accesses}) \tag{3.3}$$

$$Miss\ Rate_{L2} = (\frac{Total\ Core\ Misses_{L2}}{Total\ Core\ Misses_{L1}}) \tag{3.4}$$

To find the global miss rate, the average of each cache level is taken and multiplied. Victim miss rates are not displayed to the user - instead, NRC-SIM displays each victim cache's hit rate, as well as the average victim cache hit rate. Below is the formula used to generate a core's private victim hit rate as well as figure displaying miss rates as they relate to NRC-SIM's memory hierarchy.

$$Hit\ Rate_{Victim} = (\frac{Total\ Core\ Hits_{Victim}}{Total\ Core\ Accesses_{Victim}}) \tag{3.5}$$

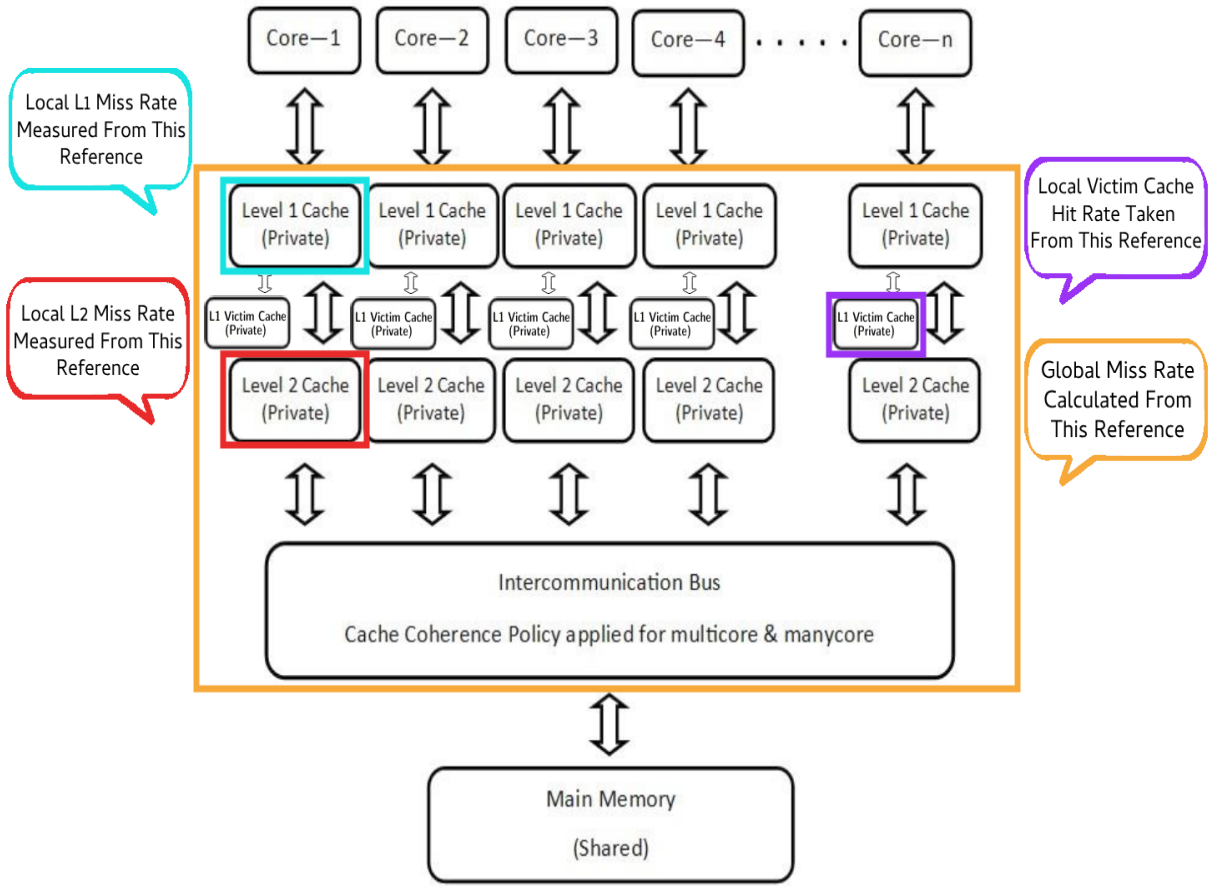

Figure 3.18: Cache Hit/Miss Rates in NRC-SIM

Source: Adapted From [10]

As has been previously mentioned, this thesis aims to design a cache simulator that is both easy to use and easy to modify for both research and educational purposes. A focus on the latter was emphasized to reduce any complications that may arise for one less familiar with the subject matter. With the underlying simulation having already been discussed at length, the way in which users interact with NRC-SIM, is through the use of Node-RED [11]. As an event-driven programming tool, Node-RED allows for seamless integration with the C++ code that is the foundation for this thesis. Once installed, its browser-based flow editor enables users to form a network of "nodes", or black-boxes that describe an applications behavior, that, when coupled with its light-weight run-time, makes it ideal for execution on low-cost hardware and in the cloud.

3.5.1 Generating Trace Files

With NRC-SIM, users can generate their own trace files directly from the user interface without ever having to interact with the command line. As a baseline, NRC-SIM includes pre-generated trace files, which are accessible via the trace file selection menu in NRC-SIM. These pre-generated trace files are for the following benchmarks: Streamcluster, Fluid Animate, Radix, Raytrace, and Ocean_CP. However, there is another option available to users - Generated Trace File. Selecting this will execute the simulation using a trace file generated by the user based on their desired configuration, i.e. core count and benchmark program. Below is an image highlighting the options available to the user when selecting a trace file for simulation.

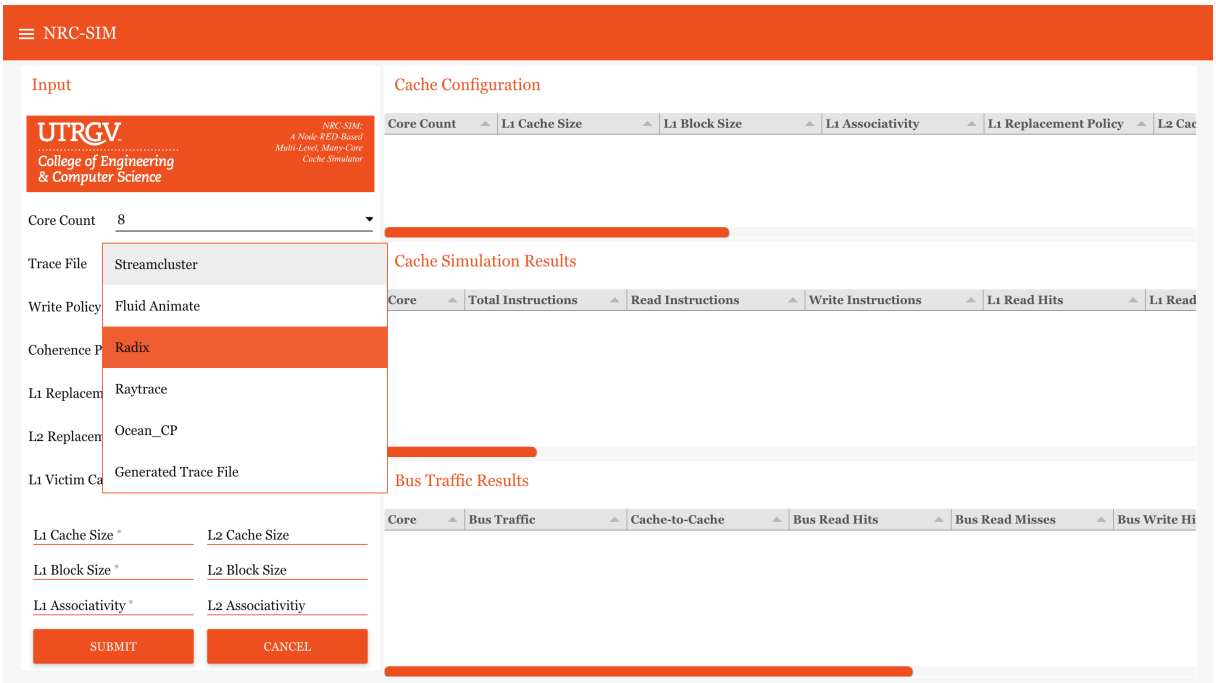

Figure 3.19: Trace File Selection in NRC-SIM

Source: Primary

To generate a trace file, users must click the hamburger menu icon in the top left section of the interface. This will open up a menu, revealing a separate “Help” page describing how to run a simulation and a “Generate Trace File” page that allows users to generate their own trace file. The following images show the hamburger menu options and the “Generate Trace File” landing page.

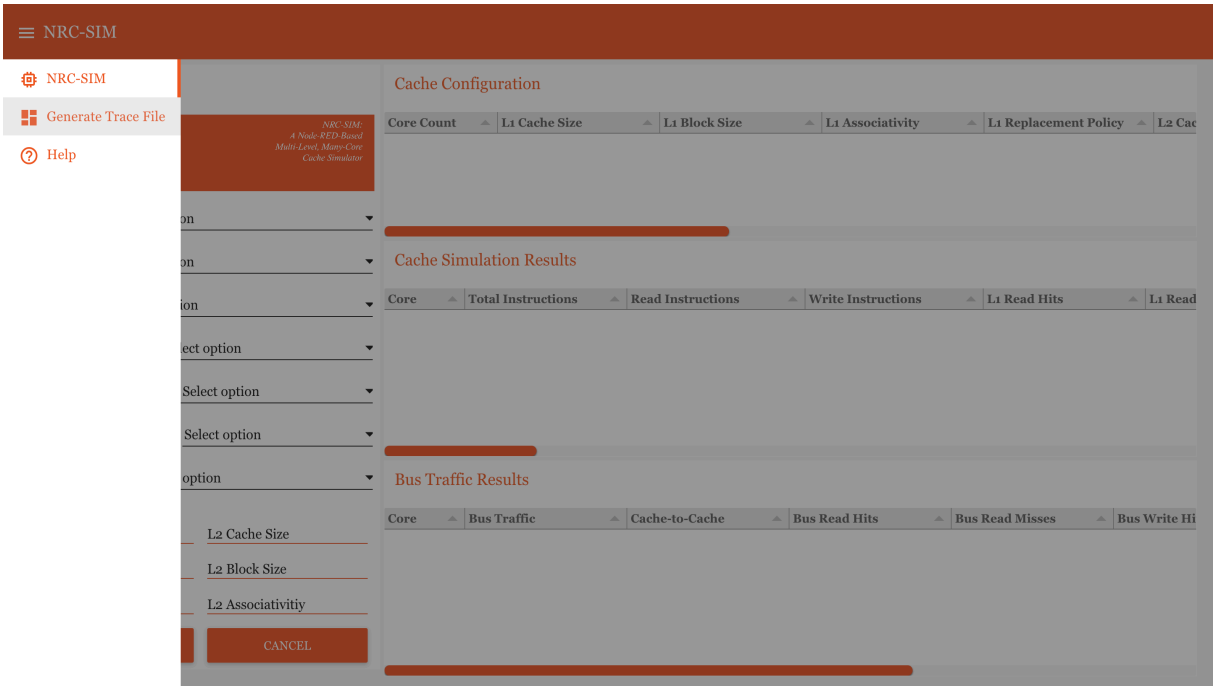

Figure 3.20: Hamburger Menu in NRC-SIM

Source: Primary

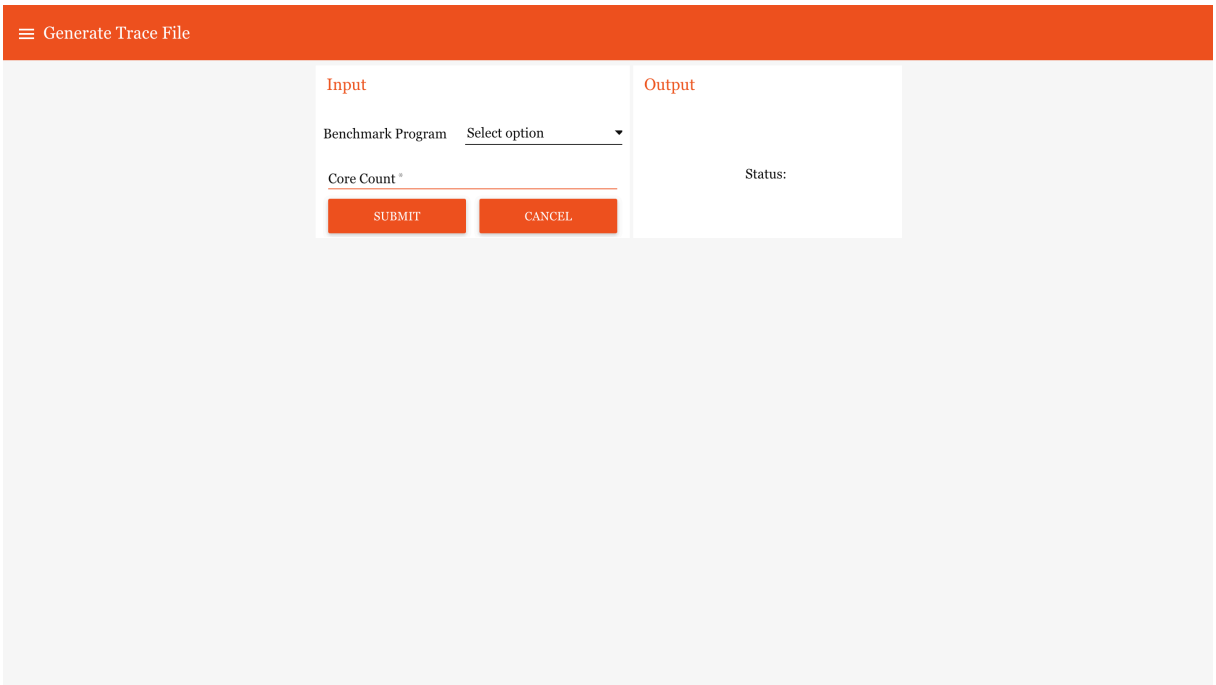

Figure 3.21: Generate Trace File Landing Page in NRC-SIM

Source: Primary

Generating a trace file requires two inputs from the user - the benchmark program they wish the trace file is generated from and the desired core count. Once both inputs have been selected, the user may submit them, with the status of the trace file being shown in the "Output" pane. The process of selecting the input parameters and generating a trace file is shown in the following figures.

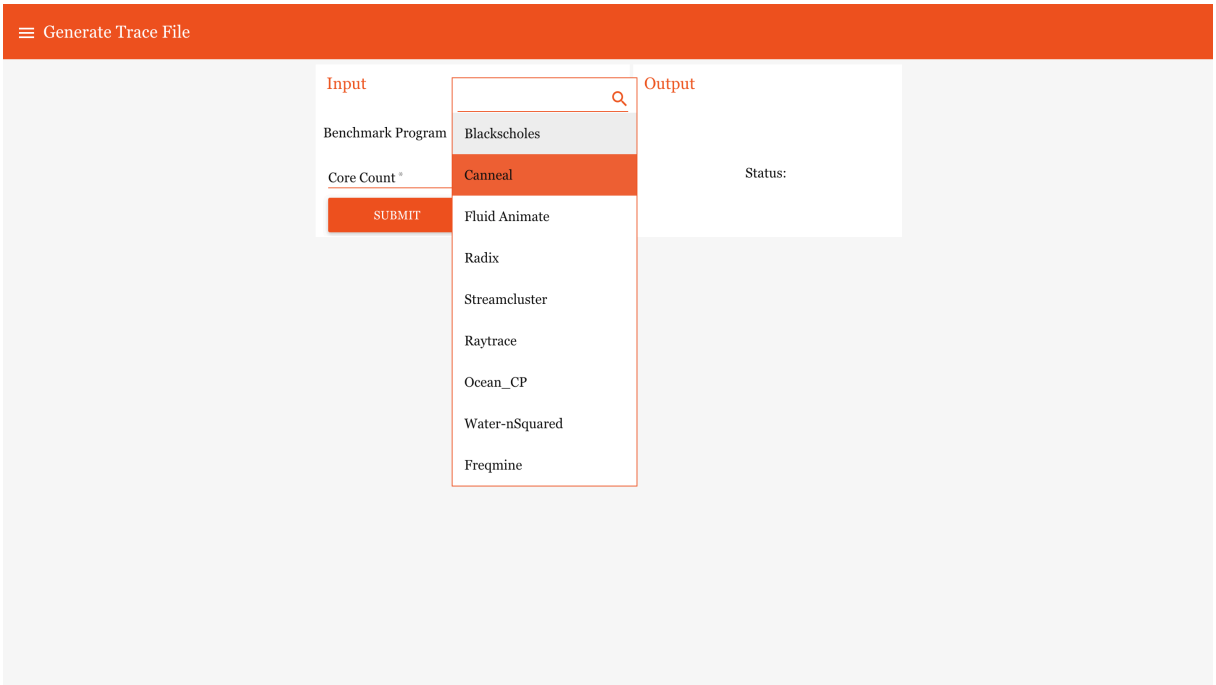

Figure 3.22: Benchmark Program Selection for Trace File Generation in NRC-SIM

Source: Primary

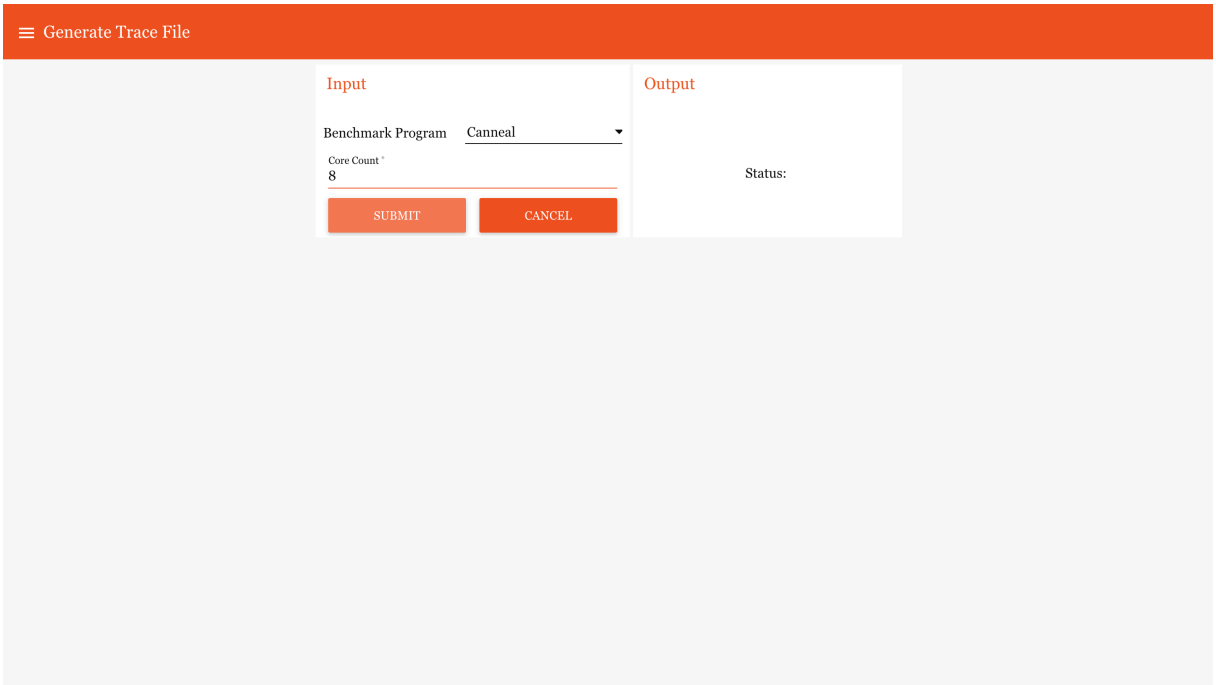

Figure 3.23: Input Parameters for Trace File Generation in NRC-SIM

Source: Primary

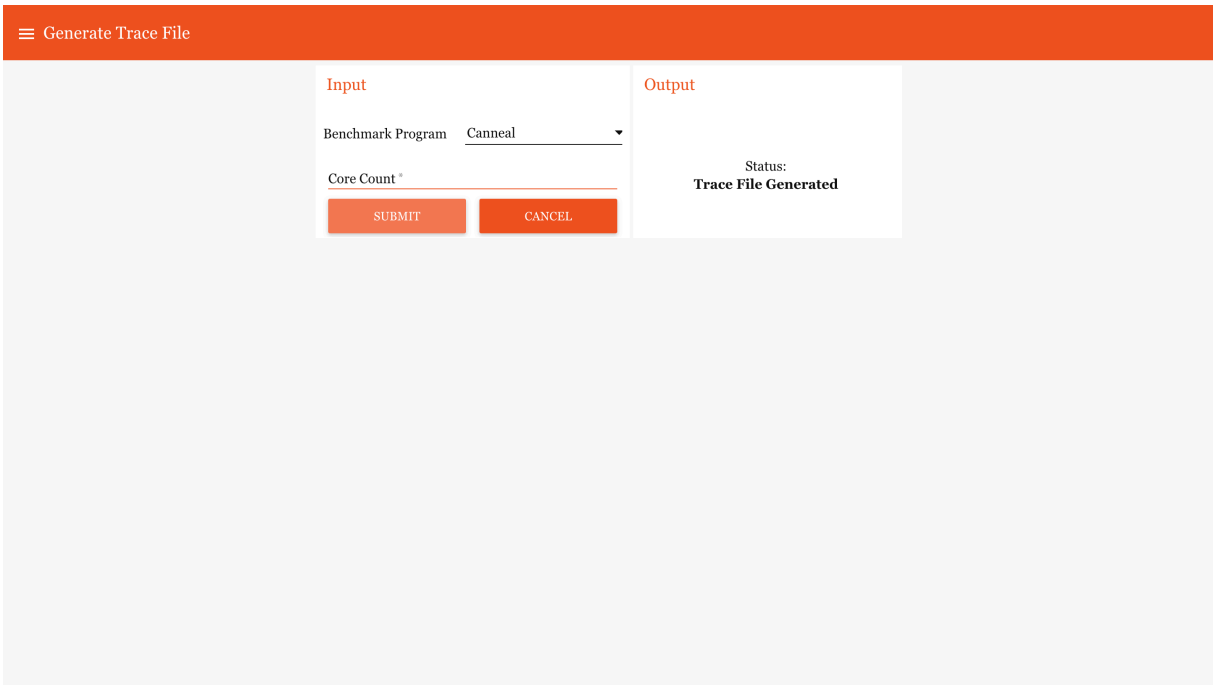

Figure 3.24: Result of Trace File Generation in NRC-SIM

Source: Primary

Once the user has generated their trace file, they can select the "Generated Trace File" option from the main landing page for simulation. As previously mentioned, the process of generating a trace file in NRC-SIM is simple and does not require any interaction with a command line terminal. This is accomplished by strategic use of several nodes (further explained in Section 3.5.2.1) that are included in the baseline Node-RED installation. A 'form' node was utilized to gather the inputs from the user, which were then stored as variables using the 'function' node. Next, the 'read file' node was used to open and modify the C++ code that enables use of the pintool. This manipulation, i.e. setting the core count and ensuring proper formatting, was done in a 'function' node and the result was then written back the C++ file using the 'write file' node. Next, the 'exec' node was used for compilation. Once compilation was complete, another 'function' node was used to format the command line instruction for running the pintool based on the users selected parameters. Another 'exec' node is then used to run the command, followed by several nodes that are used to output a message informing the user that the trace file has been generated.

3.5.2 Collecting User Inputs

When a user selects a value for a given parameter in NRC-SIM, that value is stored as a global variable within Node-RED. As a global variable, these variables can be accessed by all nodes at any point during execution. Once all variables have been entered by the user, they are formatted to match the command line input for the simulation to begin.

3.5.2.1 Node-RED Dashboard. The Node-RED Dashboard module provides a set of nodes that can be used to create a user interface capable of handling live data. As it is an add-on to the baseline Node-RED package, an additional installation is necessary. For the purposes of this thesis, the following Node-RED Dashboard nodes were used:

1. *Dropdown*

    (a) Used to gather the 'Core Count', 'Trace File', 'L1 Replacement Policy', 'L2 Replacement Policy', 'Coherence Protocol', 'Write Policy', and 'Victim Cache' input parameters

2. *Form*

   (a) Used to gather ‘L1 Cache Size’, ‘L1 Block Size’, ‘L1 Associativity’, ‘L2 Cache Size’, ‘L2 Block Size’, and ‘L2 Associativity’ input parameters

   (b) Used to store and submit *all* input parameters

3. *Table*

   (a) Used to output all relevant local and global simulation results, such as reads/writes, misses, bus traffic, etc.

4. *Media*

   (a) Used to display a UTRGV banner on the main screen and an explanation/tutorial on a separate help screen

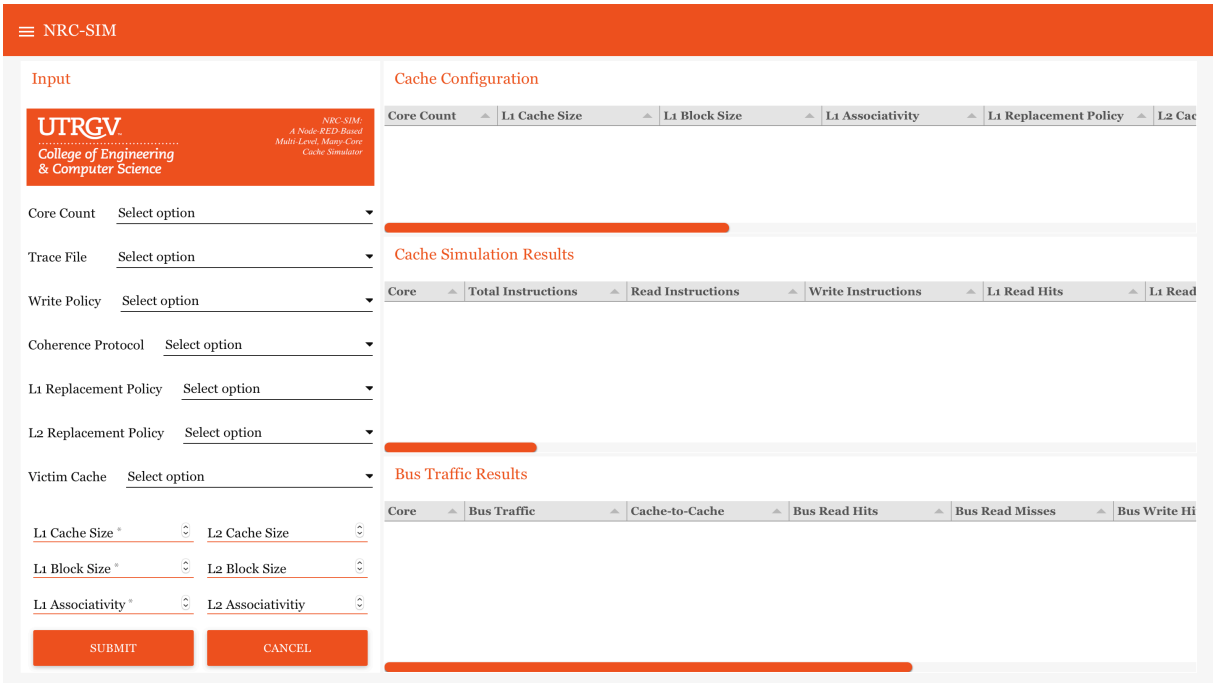

Figure 3.25: NRC-SIM User Interface

Source: Primary

3.5.3 Execution

Once input parameters have been selected by the user, and stored as a global variable, formatting the values takes precedence. This is important as the resulting format should be compliant with the foundational command line executable. For example, if a user wishes to run a single-core simulation, the selected trace file should not consist of multiple processor ID's, nor should the coherence protocol be allowed to be anything other than 'Single-Core'. The same is also true when the user wishes to run a multi-core simulation, i.e. selecting a core count greater than 1 and the 'Single-Core' coherence protocol will result in 'MESI' being selected for the user by default. For proper execution to occur, NRC-SIM requires all input parameters be selected, however, in the event of a user oversight or deliberate exclusion, NRC-SIM initializes every variable such that execution can occur without problem.

The *Function* node is a fundamental node included with every installation of Node-RED, allowing users to run JavaScript code within their project. Node-RED requires its functions return an object, and by default, these objects contain a "payload" property, however, users may add additional properties as needed. Once the user's input parameters have been selected, a function collects all of the variables and formats them as a string to match the required command line execution format, i.e. Figure 3.1. The object's payload is then set to this string and passed to the next node, *Exec*.

The Exec node is included in the baseline Node-RED installation, meaning no additional installs are required. With the Exec node, users can run a command on the command line, which is how Node-RED is used to run the underlying cache simulation. This is essential because of Node-RED's utilization of JavaScript - unlike C++, JavaScript does not allow for direct memory manipulation.

3.5.4 Outputting Data

Upon completion, the Exec node returns the output generated at the command line. A Function node is then used to parse the output into an array, with each element in the array being a

value relevant to simulation performance, i.e core count, cache sizes, miss rates, etc. These values will then be used in conjunction with multiple Table nodes to properly display the simulation results to the user.

3.5.4.1 Node-RED Dashboard. Before the output Table nodes receive any information, the parsed command line output must be properly formatted into a compatible object. To accomplish this, sets of conditional statements are made in a separate function. Each set of conditional statements checks the core count of the simulation as well as whether or not the simulation includes a second level cache. For any given condition, an object is created with properties aligning to the relevant outputs. Each object is treated as a Map and filled with key-value pairs, where the keys are result types and the values are the generated results. For example, when running a single-core, L1 only simulation, object properties, i.e. keys, should include values for properties such as "L1 Cache Size" and "L1 Miss Rate" but not for "L2 Cache Size" and "L2 Miss Rate". When running a multi-core or many-core simulation, an array of objects is required at output, with each object representing a given core's performance metrics. Additionally, an extra object is used to pass overall totals for each individual property, making the number of objects in the array for a multi-core or many-core simulation, n+1, where n represents the number of cores.

To ensure consistency across simulations, regardless of core counts or the number of cache levels, each object contains the same properties, even if they are not used in the simulation. This allows the output table to properly display the information. In NRC-SIM, there are three such tables - one for displaying the user's selected inputs, one for displaying the simulation results of each level of cache, and one for displaying bus traffic results. Each Table node allows users to create columns based on their desired properties. To avoid empty columns, such as in the case of an L1 only simulation, initialized values are used. For example, in an L1 only simulation, the L2 related fields are set to "N/A". An example of an L1 only simulation is shown below. Upon submission, the cache size, block size, and associativity parameters are reset in the input section, however, all other fields remain present, removing the need to resubmit every parameter, further adding to the ease of use for the end-user.

NRC-SIM

Input

UTRGV College of Engineering & Computer Science

NRC-SIM: A Node-RED-Based Multi-Level, Many-Core Cache Simulator

Core Count 6

Trace File Ocean

Write Policy Write-Back / Allocate

Coherence Protocol MESI

L1 Replacement Policy Random Replacement

L2 Replacement Policy Select option

Victim Cache 0 Victim Lines

L1 Cache Size * | L2 Cache Size

L1 Block Size * | L2 Block Size

L1 Associativity * | L2 Associativitiy

SUBMIT | CANCEL

Cache Configuration

Core Count	L1 Cache Size	L1 Block Size	L1 Associativity	L1 Replacement Policy	L2 Cac
6	8,192 Bytes	32 Bytes	Direct-Mapped	Random Replacement	

Cache Simulation Results

Core	Total Instructions	Read Instructions	Write Instructions	L1 Read Hits	L1 Read
Total	142,659,148	118,162,156	24,496,992	48,483,899	
1	23,779,525	19,695,641	4,083,884	8,082,258	
2	23,774,640	19,693,545	4,081,095	8,083,995	
3	23,779,856	19,699,337	4,080,519	8,083,848	
4	23,778,886	19,693,766	4,085,120	8,079,440	
5	23,769,988	19,687,531	4,082,457	8,076,237	
6	23,776,253	19,692,336	4,083,917	8,078,121	

Bus Traffic Results

Core	Bus Traffic	Cache-to-Cache	Bus Read Hits	Bus Read Misses	Bus Write Hi
Total	88,971,936	62,962,064	49,021,622	20,656,635	13,94
1	14,829,037	10,493,423	8,169,208	3,444,175	2,32
2	14,824,589	10,491,669	8,168,981	3,440,569	2,322
3	14,828,737	10,494,635	8,172,570	3,442,919	2,322
4	14,831,549	10,496,457	8,172,626	3,441,700	2,32
5	14,827,752	10,491,162	8,166,942	3,444,352	2,324
6	14,830,272	10,494,718	8,171,295	3,442,920	2,323

Figure 3.26: NRC-SIM - L1 Only Simulation Output (1/2)

Source: Primary

NRC-SIM

Input

UTRGV College of Engineering & Computer Science

NRC-SIM: A Node-RED-Based Multi-Level, Many-Core Cache Simulator

Core Count 6

Trace File Ocean

Write Policy Write-Back / Allocate

Coherence Protocol MESI

L1 Replacement Policy Random Replacement

L2 Replacement Policy Select option

Victim Cache 0 Victim Lines

L1 Cache Size * | L2 Cache Size

L1 Block Size * | L2 Block Size

L1 Associativity * | L2 Associativitiy

SUBMIT | CANCEL

Cache Configuration

Associativity	L2 Replacement Policy	Coherence Protocol	Write Policy	Trace File
N/A	N/A	MESI	Write-Back	Ocean

Cache Simulation Results

L2 Read Misses	L2 Write Hits	L2 Write Misses	L2 Miss Rate	Global Miss Rate
N/A	N/A	N/A	N/A	18.232182%
N/A	N/A	N/A	N/A	
N/A	N/A	N/A	N/A	
N/A	N/A	N/A	N/A	
N/A	N/A	N/A	N/A	
N/A	N/A	N/A	N/A	
N/A	N/A	N/A	N/A	

Bus Traffic Results

Core	Bus Traffic	Cache-to-Cache	Bus Read Hits	Bus Read Misses	Bus Write Hi
Total	88,971,936	62,962,064	49,021,622	20,656,635	13,94
1	14,829,037	10,493,423	8,169,208	3,444,175	2,32
2	14,824,589	10,491,669	8,168,981	3,440,569	2,322
3	14,828,737	10,494,635	8,172,570	3,442,919	2,322
4	14,831,549	10,496,457	8,172,626	3,441,700	2,32
5	14,827,752	10,491,162	8,166,942	3,444,352	2,324
6	14,830,272	10,494,718	8,171,295	3,442,920	2,323

Figure 3.27: NRC-SIM - L1 Only Simulation Output (2/2)

Source: Primary

The first table, Cache Configuration, displays the user's selected parameters. This includes cache size, block size, associativity, and replacement policy for both L1 and L2 caches, coherence protocol, write policy, and selected trace file. The second table, Cache Simulation Results, displays a detailed breakdown of each core's performance. This includes the total number of instructions per core, the number of read instructions per core, the number of write instructions per core, the number of calls for both the victim and L2 caches, the number of read hits, read misses, write hits, and write misses for the L1 cache, victim cache, and the L2 cache, miss rates for the L1 and L2 cache, hit rates for the victim cache, and lastly the global miss rate. The third and final table, Bus Traffic Results, displays all relevant bus traffic information, such as the total amount of traffic, cache-to-cache operations, read hits, read misses, write hits, write misses, and reads from memory.

3.6 Microsoft Azure

NRC-SIM is designed to be a web-accessible and easy to use cache simulator - to accomplish the former, Microsoft's Azure [31] platform was utilized. Once logged into Microsoft Azure console, a virtual machine was created. This virtual machine was selected to be an Ubuntu Server and followed the setup process described in [9]. Next, the virtual machine was started, allowing access to the Serial Console. With the serial console, Node-RED is accessible via the virtual machine on Azure's servers.

As NRC-SIM was developed on a local machine, the transition to a web-accessible server resulted in several issues coming to light, however, they were easily resolved. For instance, one issue was not having permission to run a simulation using NRC-SIM. This can fixed by either recompiling the program in the Azure servers or by granting execution privileges using the following command, where NRC-SIM is the name of the executable:

$$\textbf{chmod u+x NRC-SIM} \tag{3.6}$$

To upload any relevant directories and/or files, such as the Pin directory, the PARSEC benchmark suite directory, and the source code, Azure CLI (Command Line Interface) was installed and

utilized in conjunction with the SCP command. For example to upload the Pin directory to the virtual machine in Azure, the command would be as follows:

scp User/Documents/pin-3.28.tar.gz Username@IPaddress:NRC-SIM (3.7)

The format of the above command is as follows: *scp source destination*, where the source is a compressed directory containing all relevant Pin files and the destination is a directory titled NRC-SIM for the user at the provided IP address. Once entering the user's password for confirmation, the upload begins. With all necessary files being in the Azure servers, execution of NRC-SIM is possible, and through the virtual machine's public IP address, NRC-SIM is accessible from any web-connected device.

CHAPTER IV

SIMULATION METHODOLOGY

As a trace-driven cache simulator, NRC-SIM requires the use of trace files. These trace files contain all of the relevant information needed for execution - processor IDs, instruction types, and memory addresses. Each trace file generates traces of the execution of benchmark programs through the use of Intel's Pin [11]. The figure below illustrates this process as well as provides an overview on this thesis as a whole.

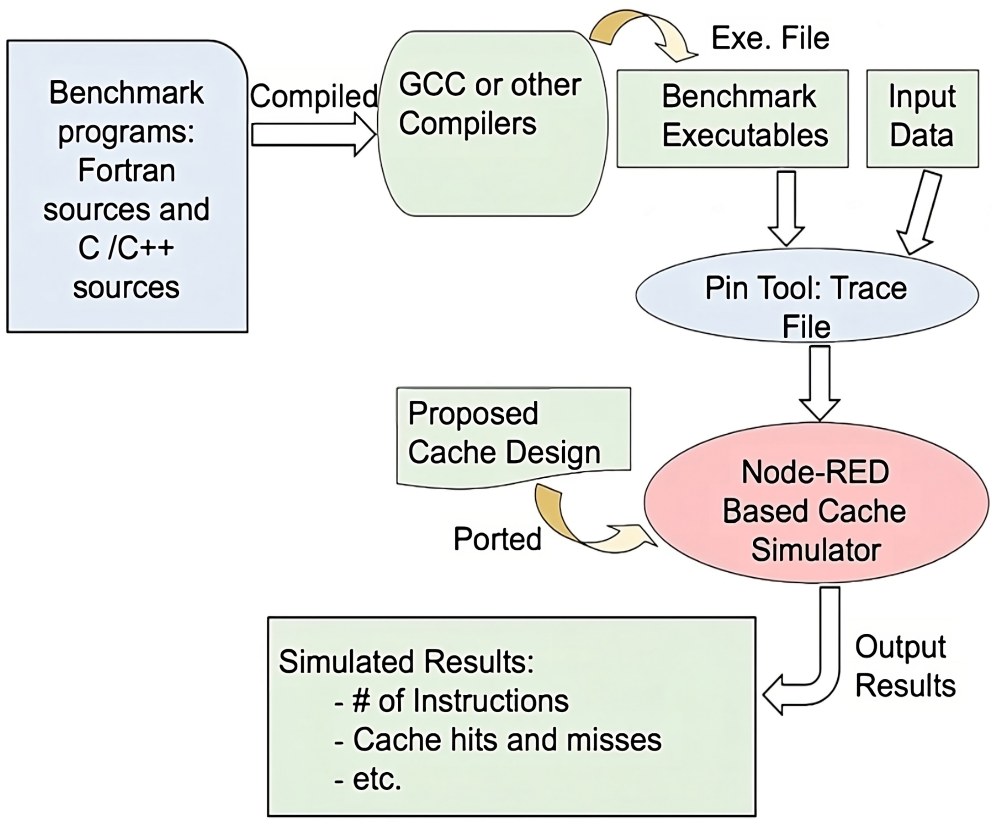

Figure 4.1: NRC-SIM Overview

Source: Primary

4.1 Benchmark Programs

This thesis utilizes benchmark programs found in the PARSEC [16] and SPLASH-2 [18] benchmark suites. Each benchmark suite provides a wide selection of multi-threaded programs that are as representative as possible of real-world scenarios. The following tables provide an in-depth look at the benchmark programs provided in the previously mentioned benchmark suites.

Table 4.1: SPLASH-2 Benchmarks

Source: Adapted From [10]

Benchmarks	Input Parameters and Environment	Number of Memory References	Number of Instructions
barnes	16384 n-bodies	51.27M	212.51M
fft	64K complex doubles	23.17M	59.3M
fmm	256 particles, 2 cluster	3.67M	17.13M
ocean_cp	258x258 grid	168M	407.6M
ocean_ncp	258x258 grid	154.4M	395.26M
radiosity	17108 elements, 364 patches	568M	2267.95M
radix	256K keys, Radix = 1K	24.26M	52.99M
raytrace	teapot.env	149.6M	322.5M
volrend	4x4 image block	28.68M	60.78M
water-nsquared	512 molecules	227.9M	658.8M
water-spatial	512 molecules	211.8M	591.29M

Table 4.2: PARSEC Benchmarks

Source: Adapted From [10]

Benchmarks	Input Parameters and Environment	Number of Memory References	Number of Instructions
blackscholes	simsmall	101.1M	234.6M
bodytrack	simsmall	613.95M	1429.6M
canneal	simsmall	944.2M	1832M
facesim	simsmall	1032M	3451M
ferret	test	11.31M	285.09M
fluidanimate	test	26.84M	60.03M
freqmine	test	27.98M	66.24M
streamcluster	test	1.14M	2.74M
vips	test	89.22M	154.57M
x264	test	0.73M	1.61M

4.2 Pin

Pin is a dynamic binary instrumentation framework developed by Intel that enables detailed program analysis. Run–time information is collected by inserting code dynamically while the executable is running, generating a trace file that will then be fed into the proposed cache simulator. This process does not affect the original program, allowing for detailed and accurate results.

Included in Pin are various Pintools that each serve their own purpose. Users are welcome to write their own Pintools, however, for the purposes of this thesis, modifications were made to the *Pinatrace* tool, which generates a trace of all memory addresses referenced by a given program. This is done to arrange the trace file in the format required by NRC-SIM for proper execution to occur. This format is composed of three columns - the processor ID, the instruction type, and the memory address referenced. The processor ID represents the processor that accesses a given

memory address. For instance, in a single-core simulation, the processor ID of all instructions will be ‘0’, however, a quad-core simulation will have processor IDs ranging between ‘0’ and ‘3’. The instruction type represents whether the referenced memory address was written to (w) or read from (r). Once the modifications were made to Pinatrace, the trace files were generated via Pin instrumentation in the following format:

pin -t obj-intel64/pinatrace.so - - [desired benchmark program] (4.1)

An example of the formatting for Pin instrumentation is shown below as well as an example of the resulting trace file.

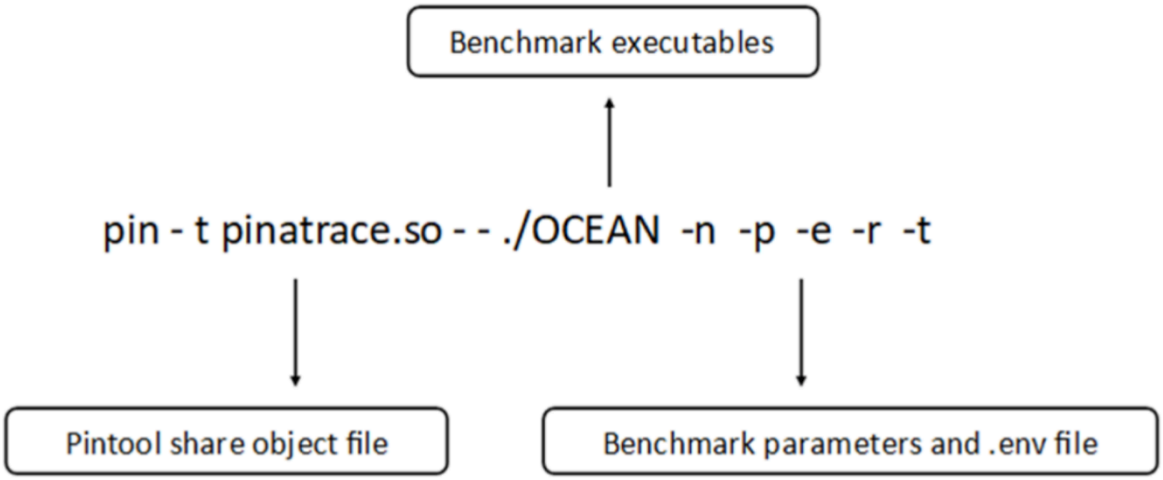

Figure 4.2: Pin Instrumentation Formatting Example

Source: [10]

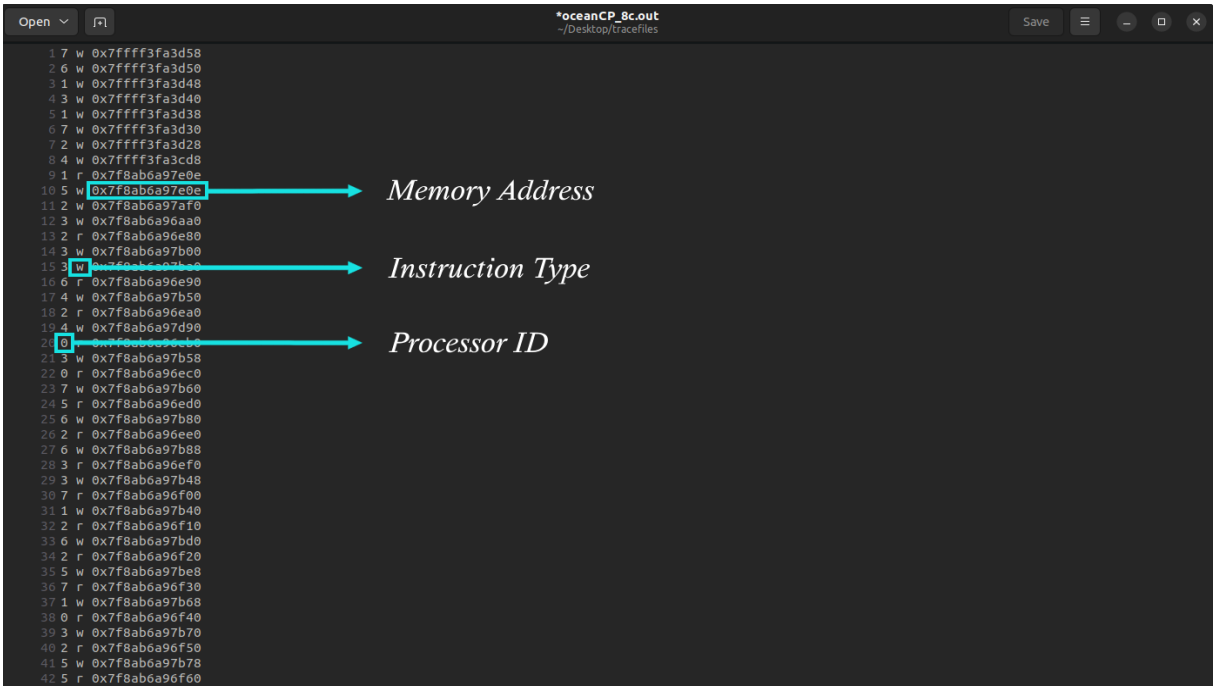

Figure 4.3: Trace File Example

Source: Primary

CHAPTER V

EXPERIMENTAL RESULTS

NRC-SIM is designed to be an easy to use and easy to modify cache simulator. By building upon the work done in [10], NRC-SIM is able to provide such a simulator, capable of multi-level and many-core cache simulation with various parameters that can be modified by the user via a graphical user interface.

To view the effects of a given cache configuration, simulations were run using the following parameters: L1 only cache with cache sizes of 8KB and 32KB, a block size of 64B, LRU replacement policy, MESI coherence protocol, 4 victim lines, and varying associativities on 4 and 8-core systems utilizing the write-back policy. The ensuing results are shown below.

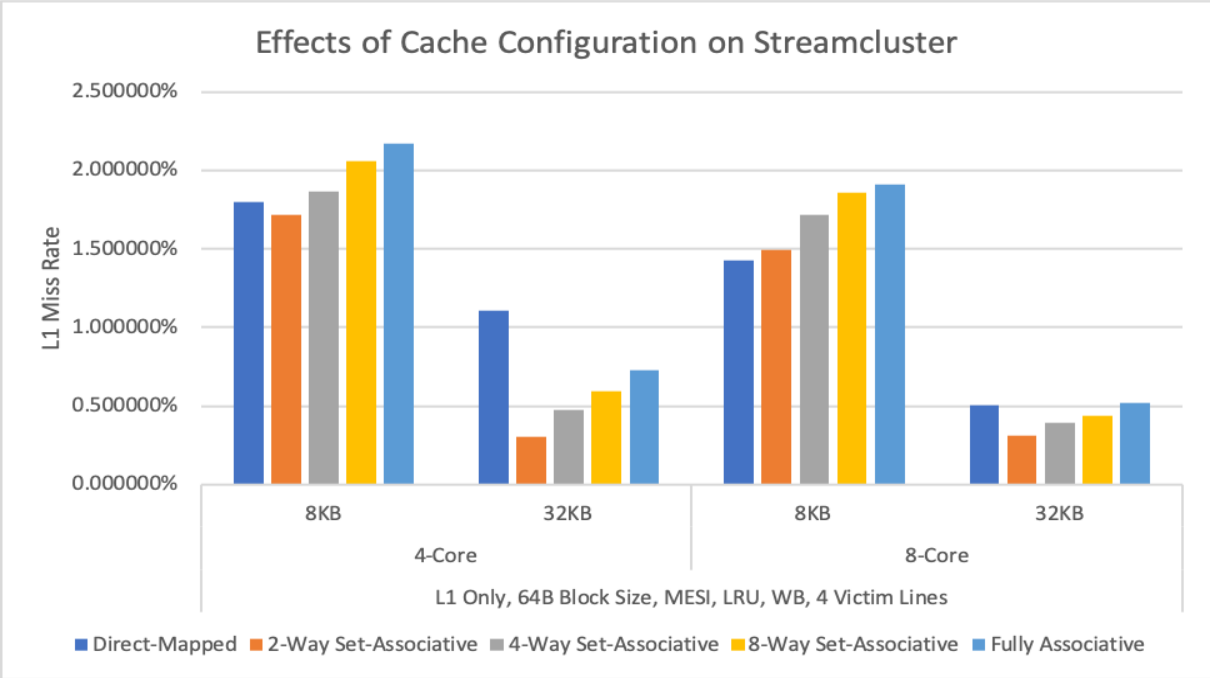

Figure 5.1: Effects of Cache Configuration on Streamcluster

Source: Primary

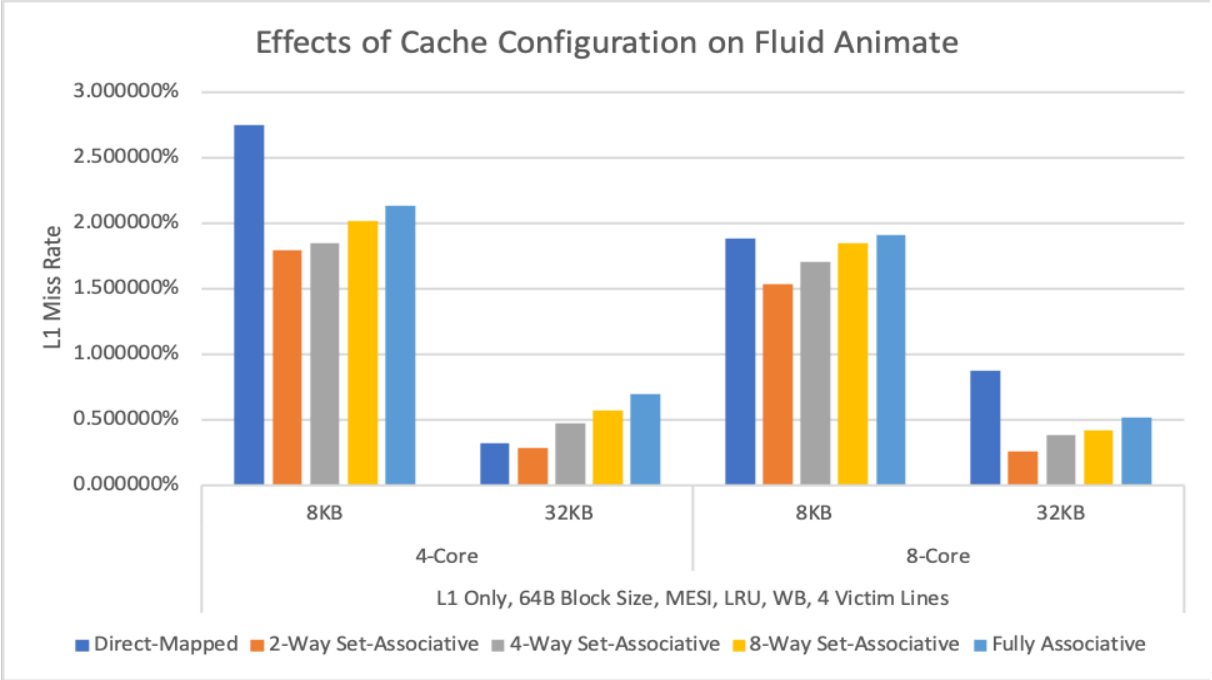

Figure 5.2: Effects of Cache Configuration on Fluid Animate

Source: Primary

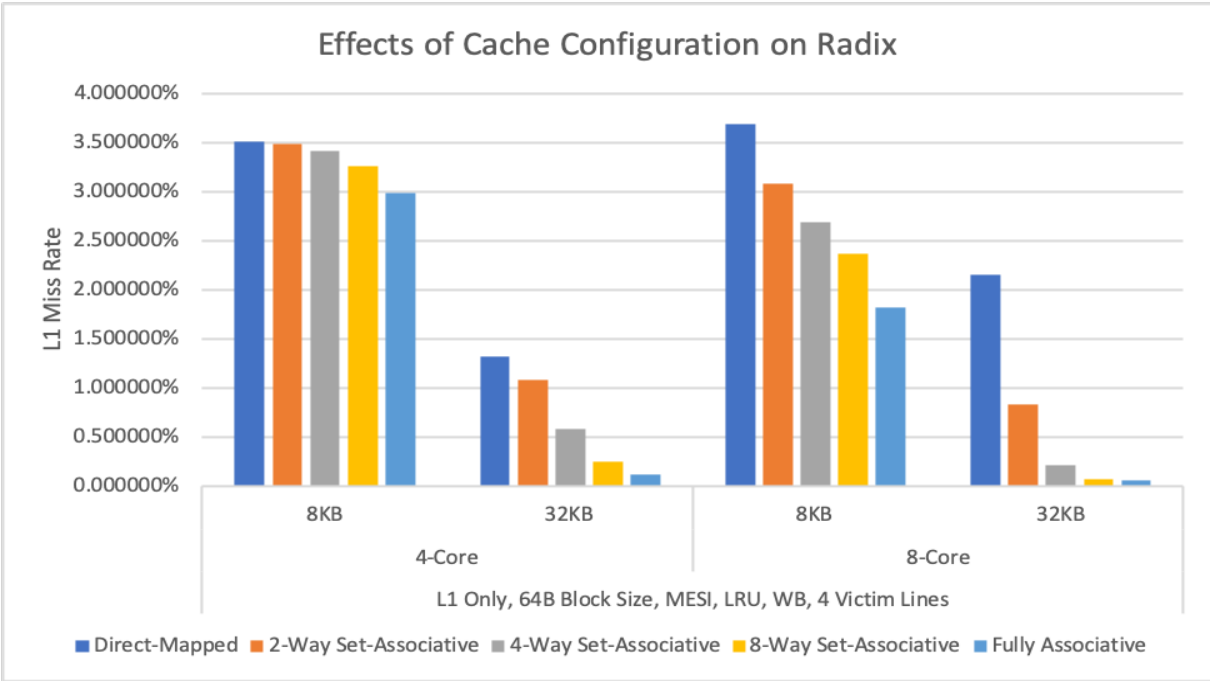

Figure 5.3: Effects of Cache Configuration on Radix

Source: Primary

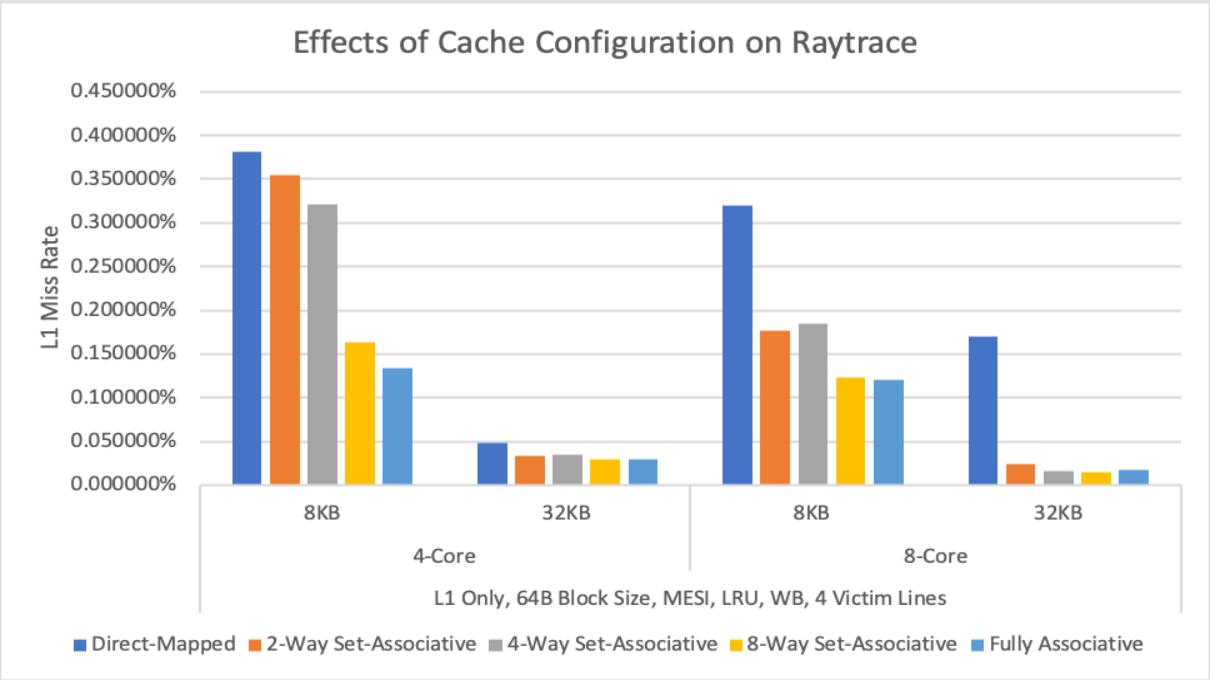

Figure 5.4: Effects of Cache Configuration on Raytrace

Source: Primary

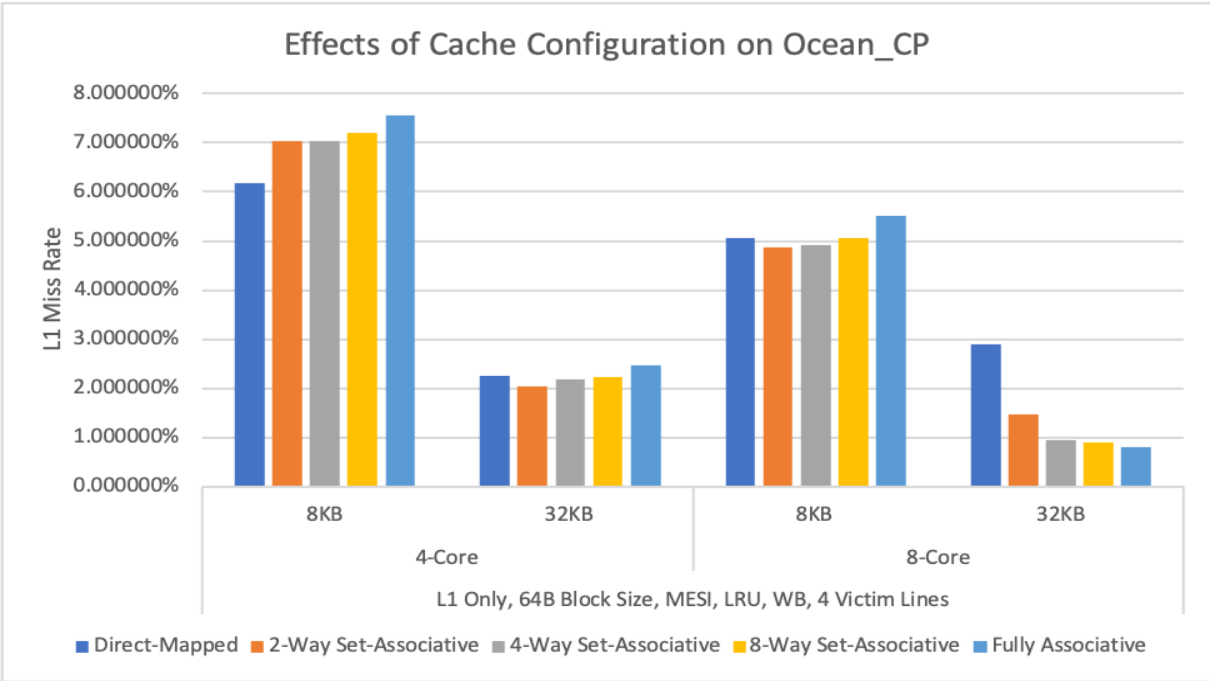

Figure 5.5: Effects of Cache Configuration on Ocean_CP

Source: Primary

To further illustrate the importance of associativity, the following figures were created to illustrate the effects a victim cache has on varying associativities. Moreover, they illustrate the concept of victim caching, which was created to address conflict misses that arise in direct-mapped cache structures. The L1 cache size was selected to be 4KB with a block size of 32 bytes, utilizing the LRU replacement policy.

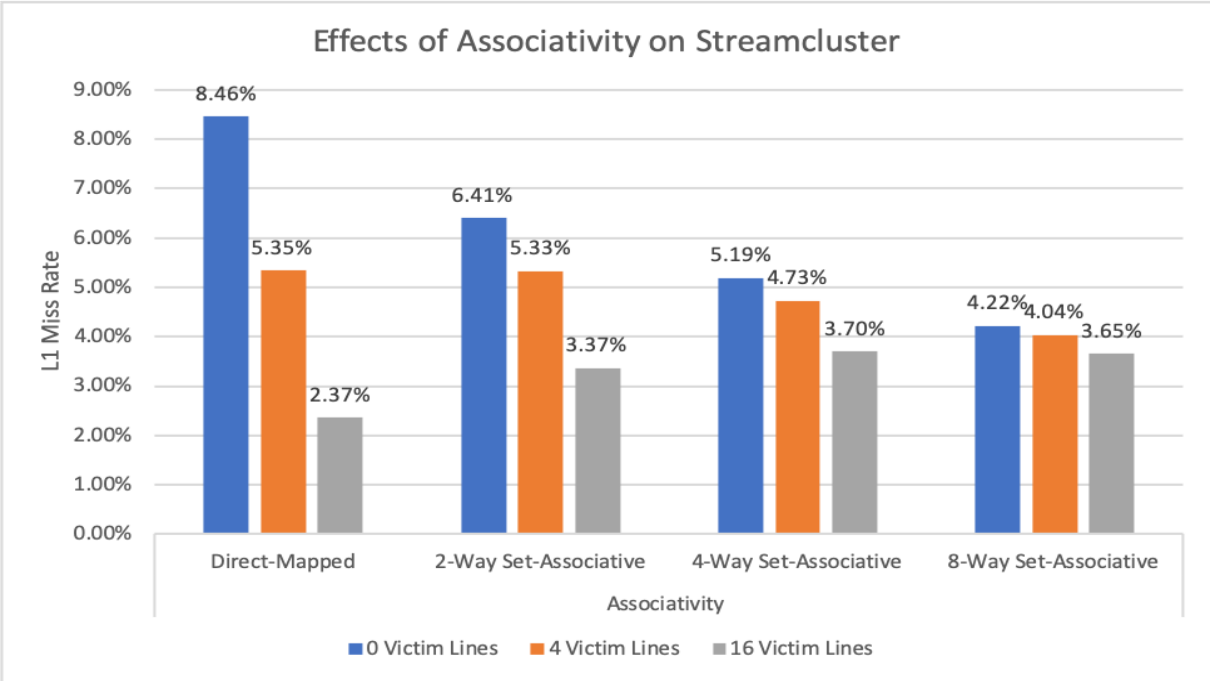

Figure 5.6: Effects of Associativity on Streamcluster

Source: Primary

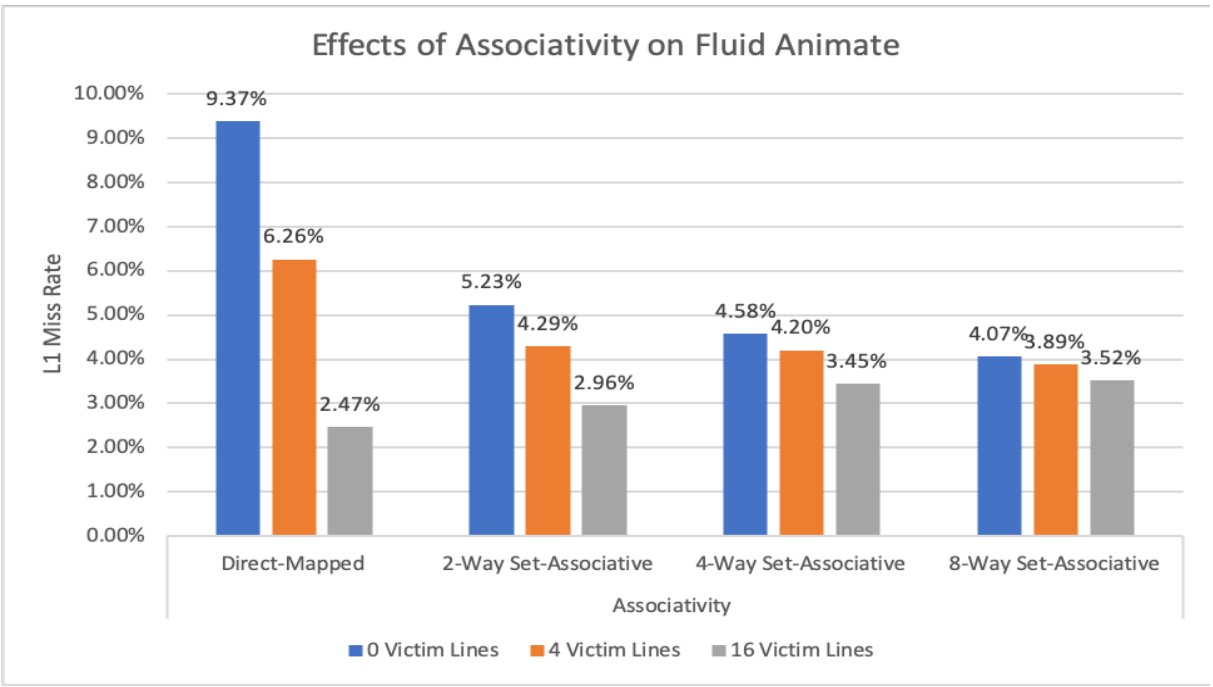

Figure 5.7: Effects of Associativity on Fluid Animate

Source: Primary

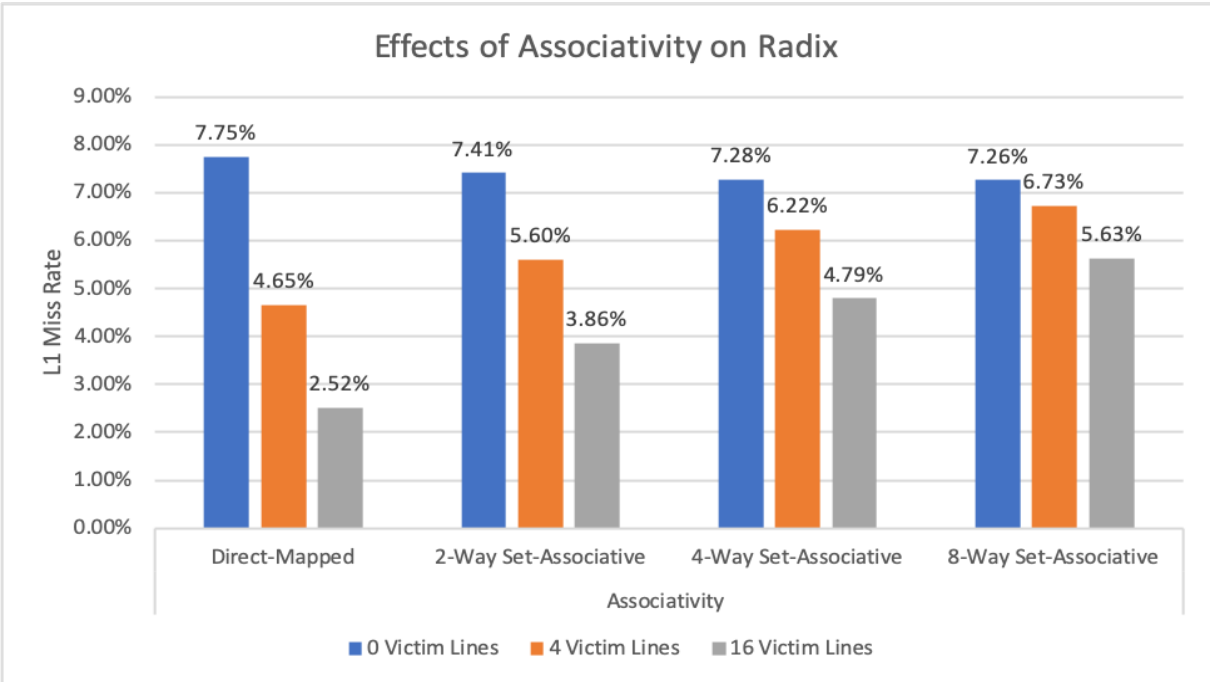

Figure 5.8: Effects of Associativity on Radix

Source: Primary

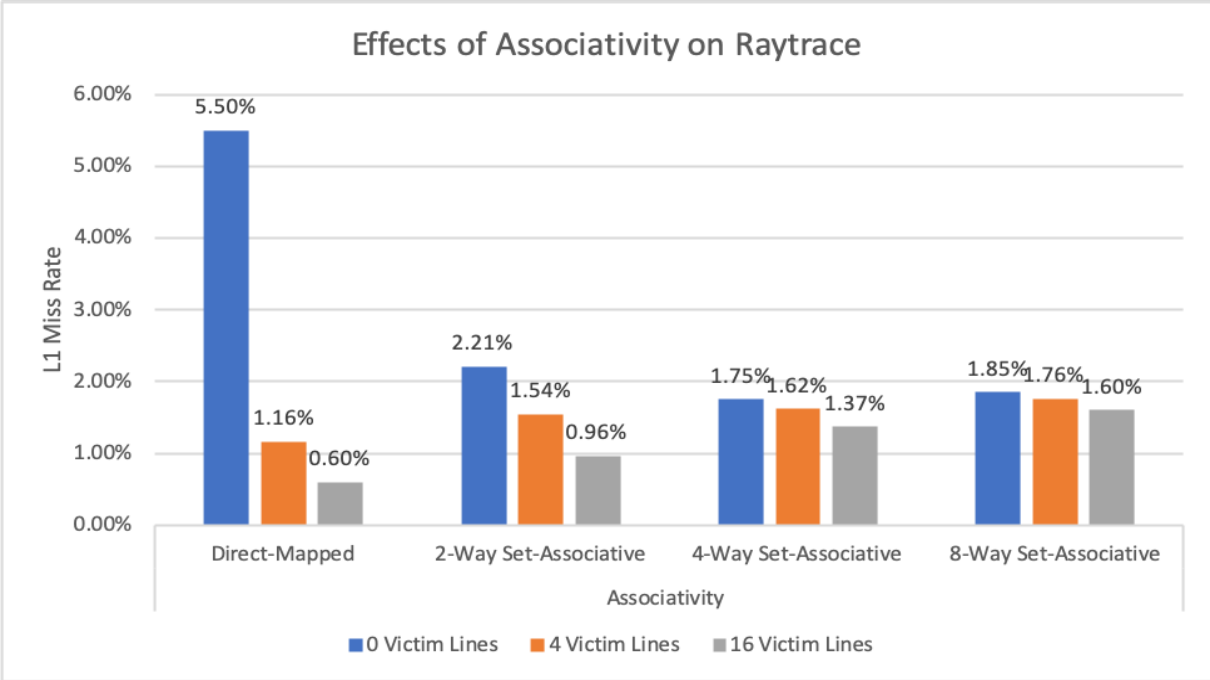

Figure 5.9: Effects of Associativity on Raytrace

Source: Primary

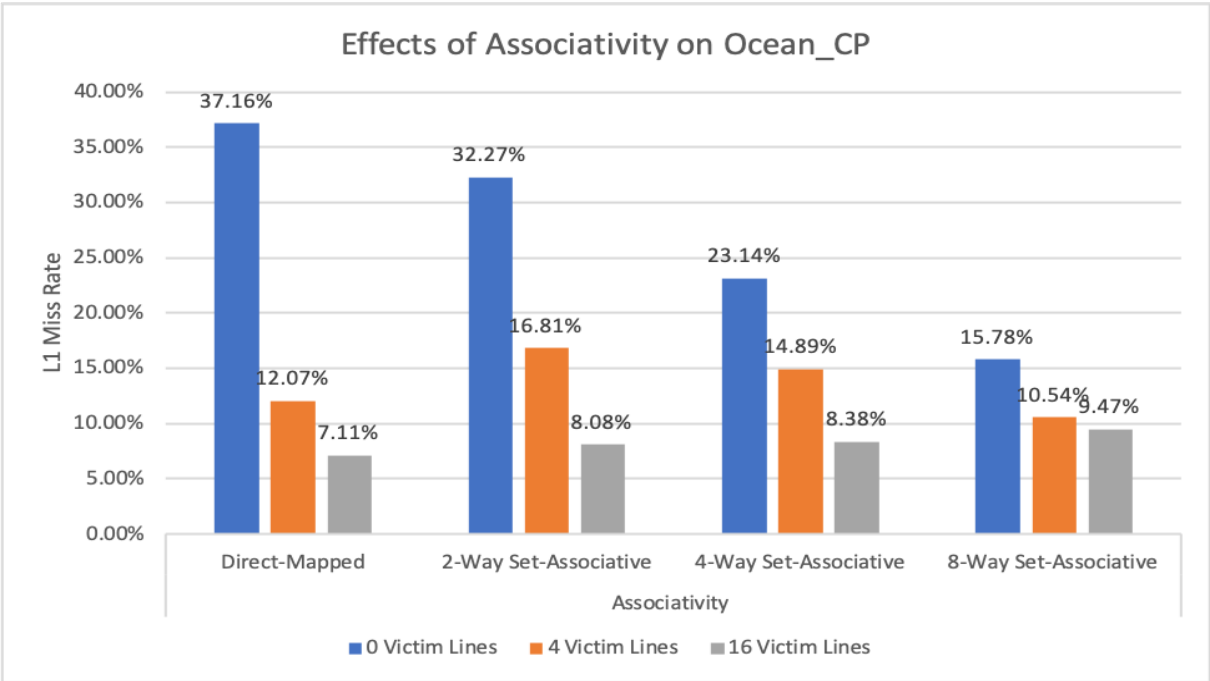

Figure 5.10: Effects of Associativity on Ocean_CP

Source: Primary

As can be seen from the previous results, while beneficial to set-associative caches, direct-mapped caches are *significantly* affected by the inclusion of a victim cache. This makes direct-mapped caches ideal for the inclusion of a victim cache.The following two figures showcase baseline performance for a single-core, L1 only cache simulation with varying cache sizes across five benchmark programs, with a direct-mapped cache structure and a block size of 32 bytes. Additionally, it utilizes the LRU replacement policy and the write-back policy. The simulation was done with the inclusion of a victim cache, which has a round-robin replacement policy, at two sizes - 4 entries and 16 entries.

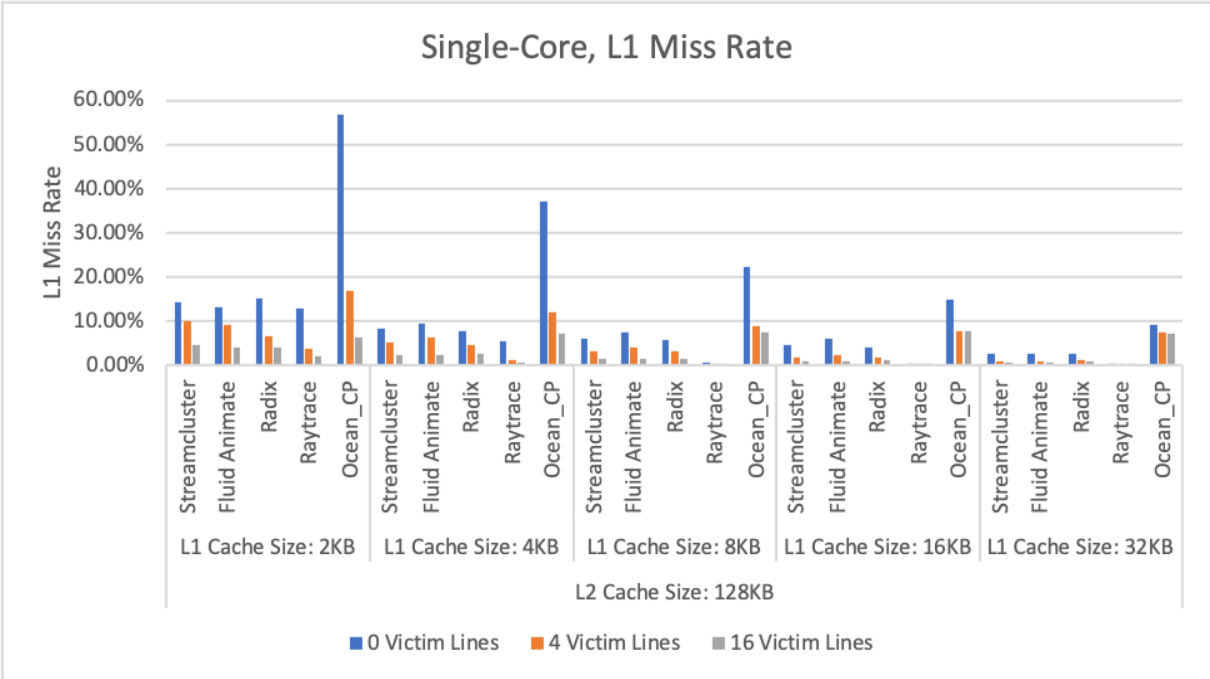

Figure 5.11: Single-Core, L1 Miss Rate

Source: Primary

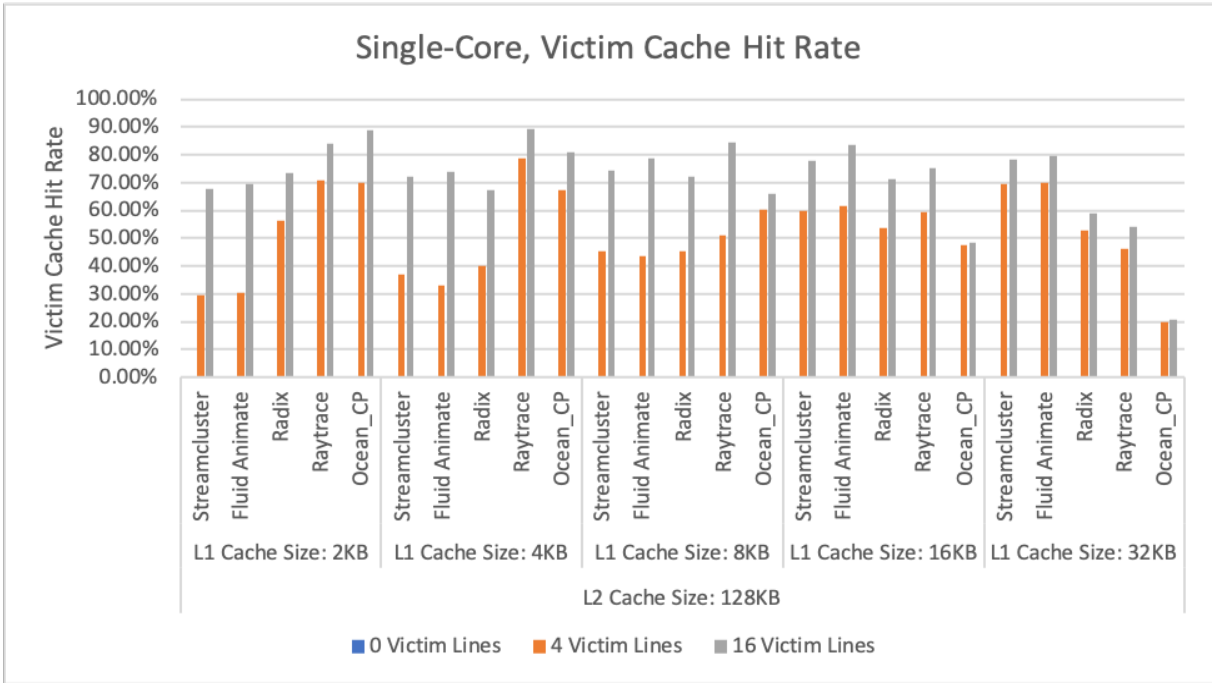

Figure 5.12: Single-Core, Victim Cache Hit Rate

Source: Primary

As shown in the above, adding a small L1 victim cache can greatly reduce the miss rate in an L1 cache. Additionally, as shown in [19], larger victim caches do not offer significantly better results The following two figures represent a 16-core, L1 only simulation. For this simulation, the experimental parameters were the same as the single-core simulation, i.e. a direct-mapped cache structure utilizing the LRU replacement policy and write-back policy, with varying L1 cache sizes over a series of 5 benchmarks, utilizing the MESI coherence protocol. Again, the victim cache was selected to be either 4 or 16 entries. For the results of each miss rate/hit rate, the average was taken across all cores.

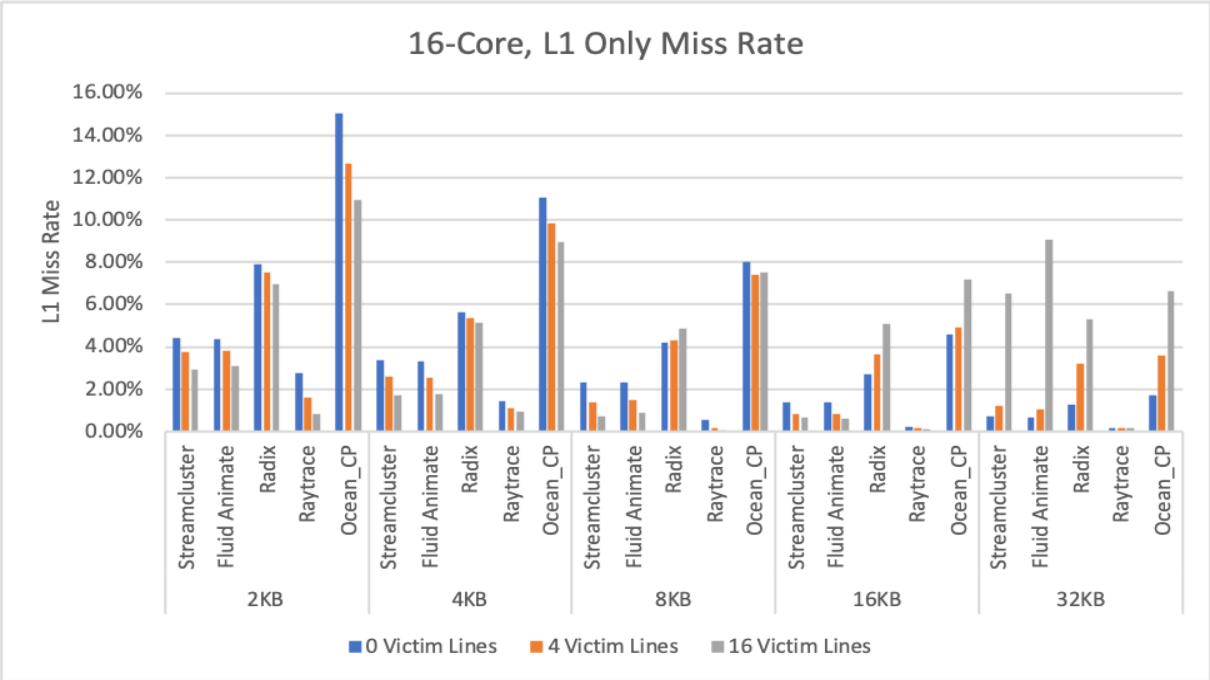

Figure 5.13: 16-Core, L1 Miss Rate

Source: Primary

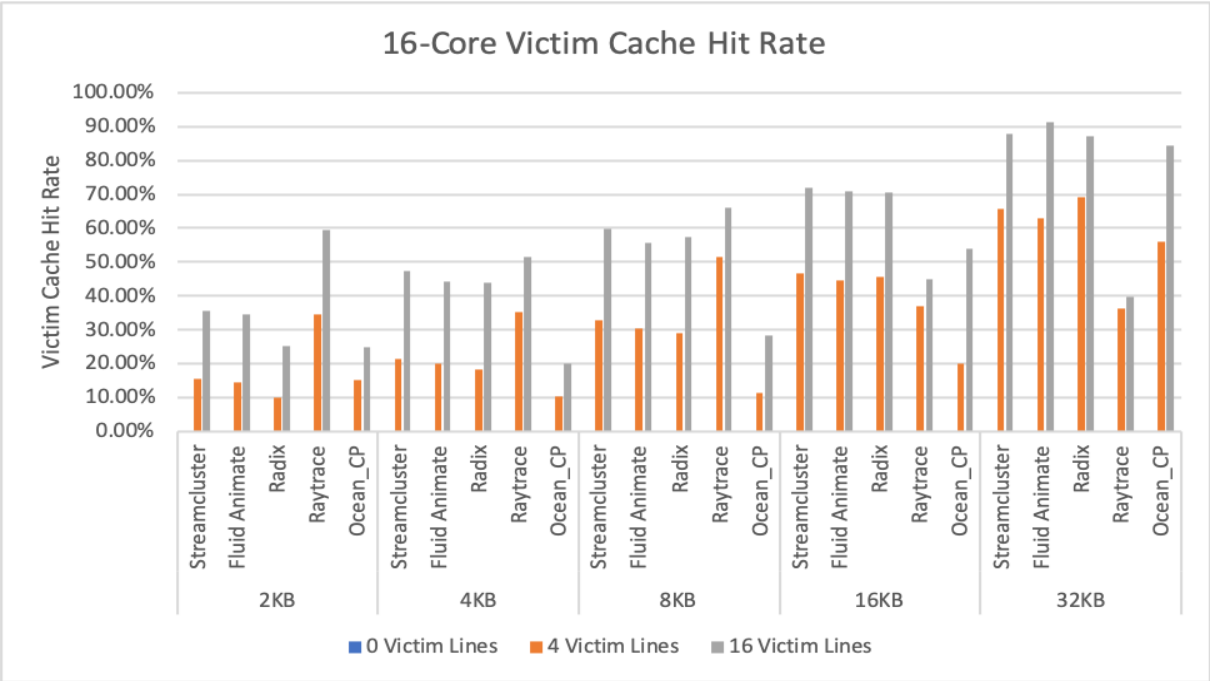

Figure 5.14: 16-Core, Victim Cache Hit Rate

Source: Primary

These results show that in most cases, with L1 cache sizes of 8KB or less, victim caches improved the miss rates for the given simulated parameters. However, larger cache sizes generally did not behave in the same manner. Additionally, the results are comparable with the findings in [19] - "In general smaller direct-mapped caches benefit the most from the addition of a victim cache". Next, are the simulated results for a 16-core, multi-level cache configuration. The L1 cache is, again, treated as before - varying cache sizes, direct-mapped, 32B block size, and LRU replacement policy, and utilizing the MESI coherence protocol. The L2 cache remains static in the experiment - 128KB cache size, 4-way set-associative, 64B block size, and LRU replacement policy. Again, the victim cache was selected to be either 4 or 16 entries and the results of each miss rate/hit rate, was collected as the average taken across all cores.

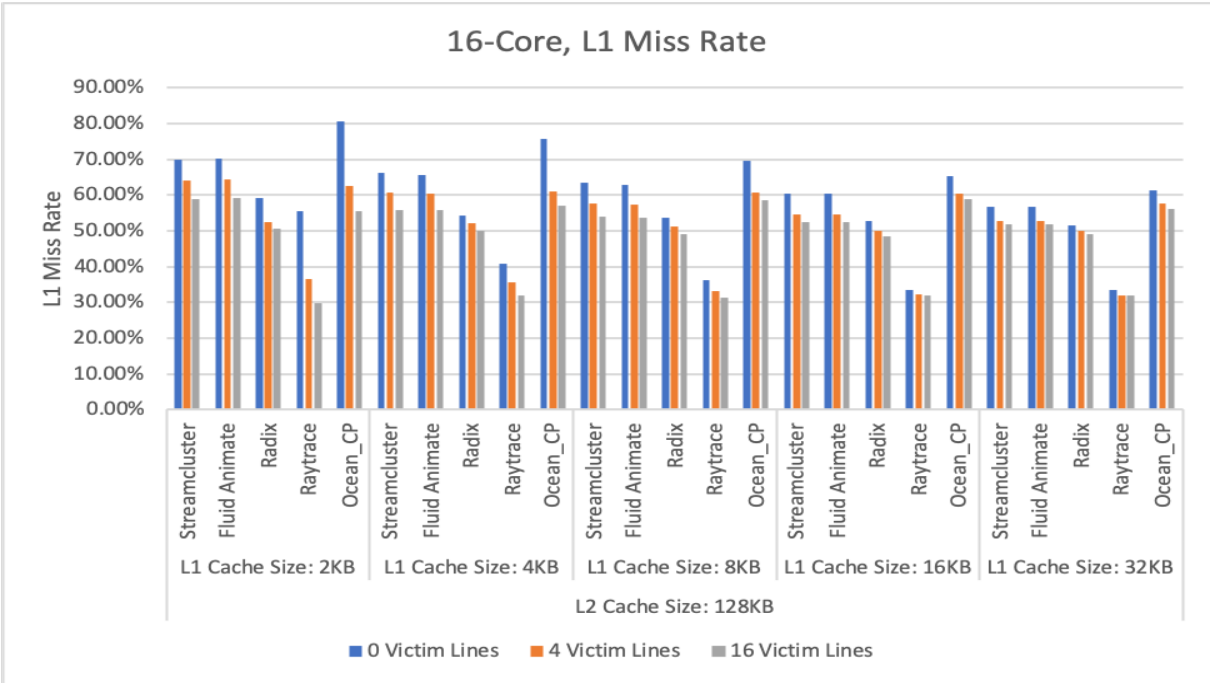

Figure 5.15: 16-Core, L1 Miss Rate

Source: Primary

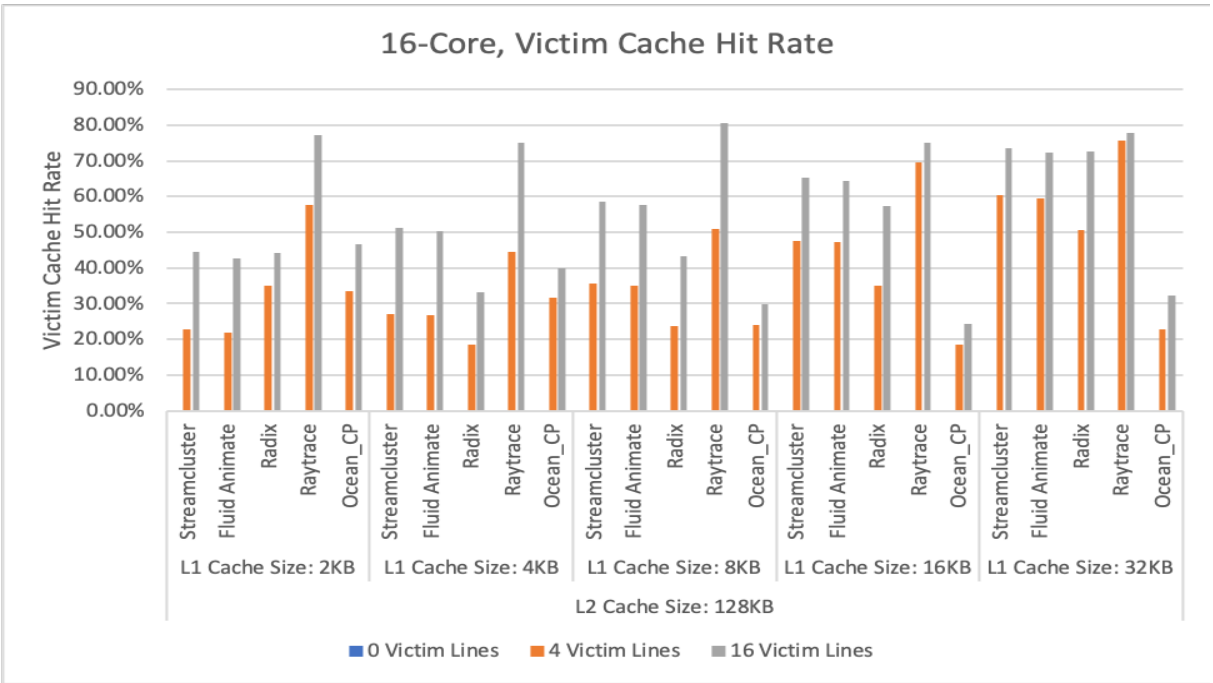

Figure 5.16: 16-Core, Victim Cache Hit Rate

Source: Primary

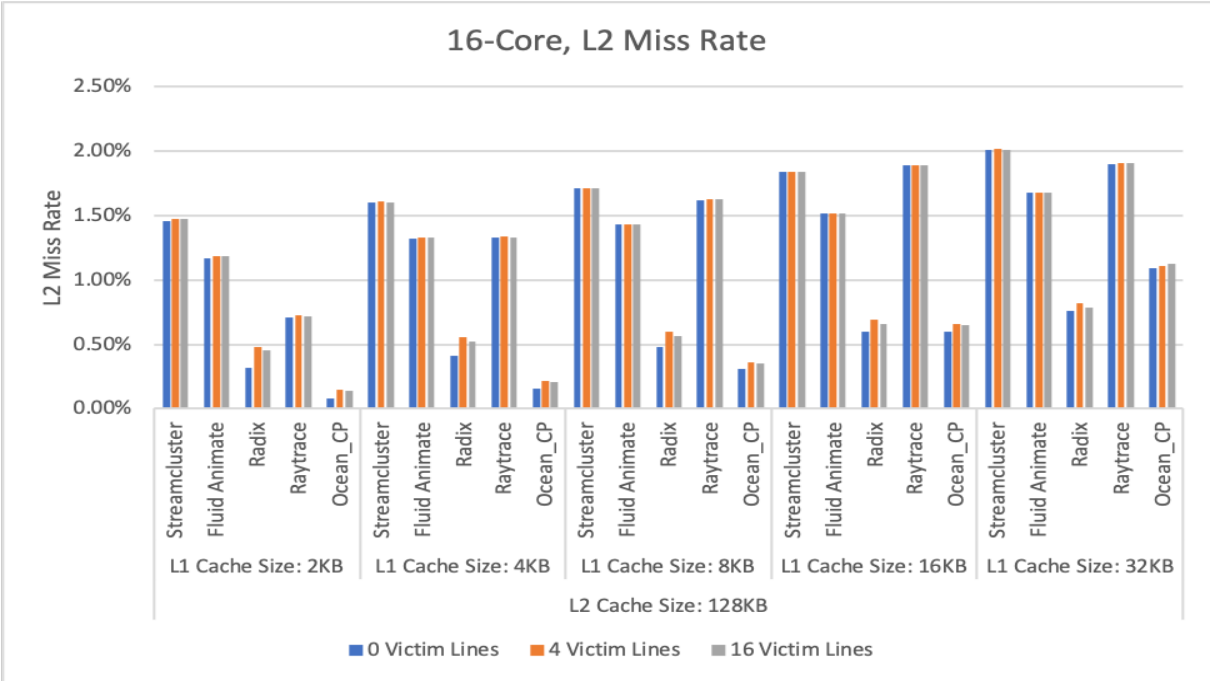

Figure 5.17: 16-Core, L2 Miss Rate

Source: Primary

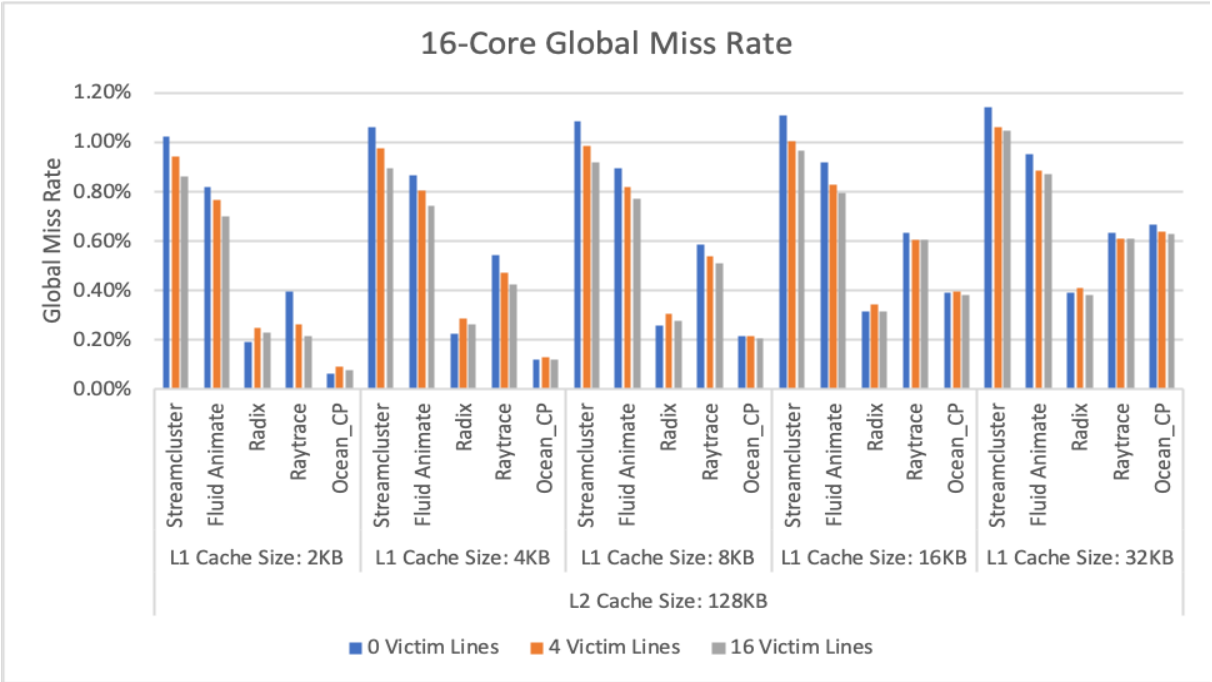

Figure 5.18: 16-Core, Global Miss Rate

Source: Primary

For a 16-core cache simulation, the figures above once again show the effectiveness of a victim cache. While L1 miss rates are improved, L2 miss rates are either equivalent or slightly worse than their 0 victim line counterparts. However, when observing the overall global performance of the simulation, the inclusion of a victim cache almost always leads to an improvement of performance. To better see the effect of an L1 victim cache, single-core L1 only simulations were done, with the ensuing results shown in the following figures. The L1 cache size was kept at 4KB, with a block size of 32 bytes, direct-mapped associativity, and utilized the LRU replacement policy.

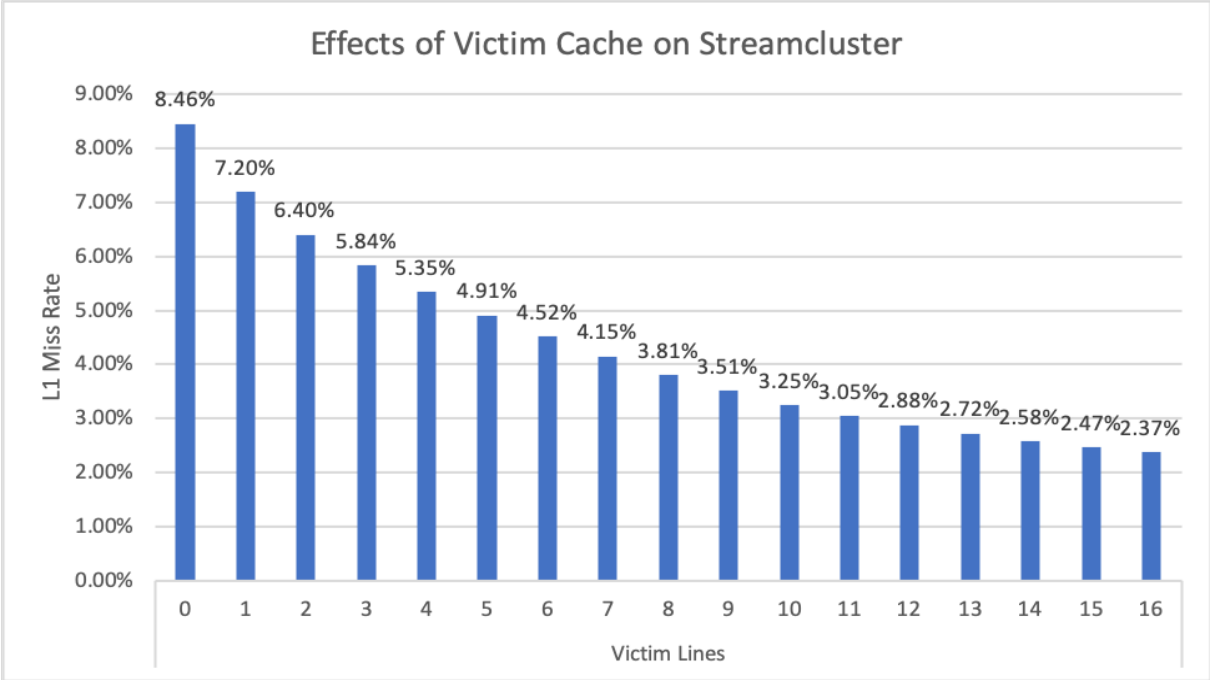

Figure 5.19: Effects of Victim Cache on Streamcluster

Source: Primary

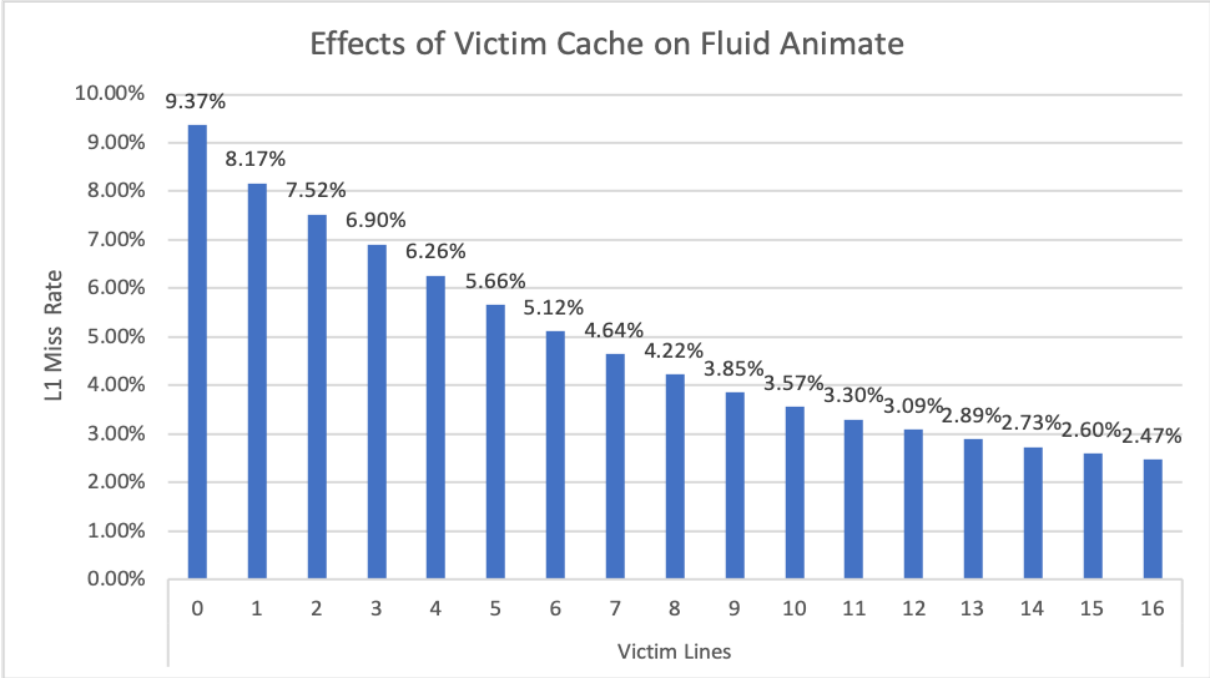

Figure 5.20: Effects of Victim Cache on Fluid Animate

Source: Primary

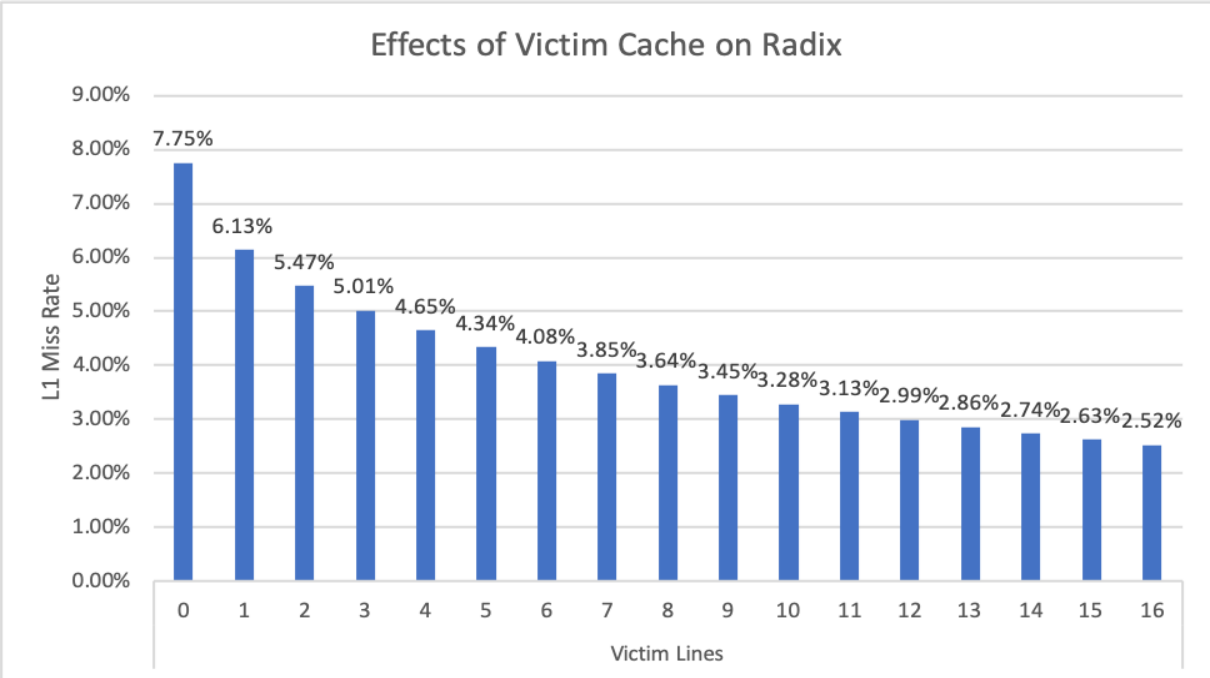

Figure 5.21: Effects of Victim Cache on Radix

Source: Primary

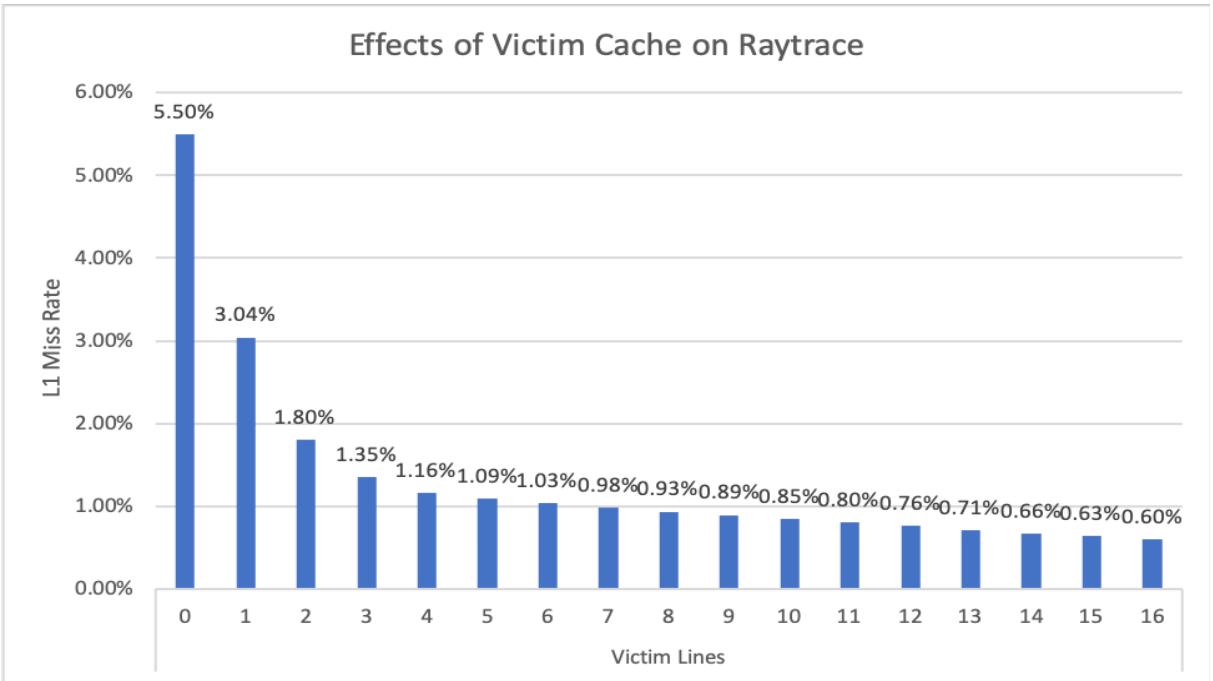

Figure 5.22: Effects of Victim Cache on Raytrace

Source: Primary

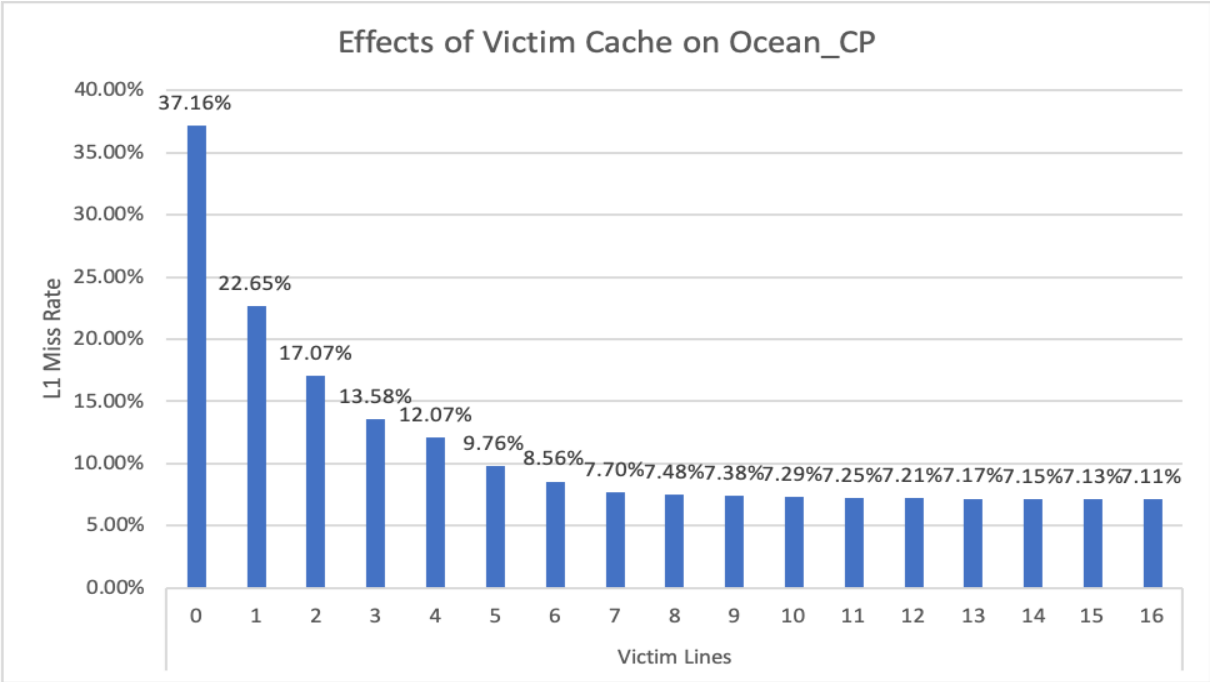

Figure 5.23: Effects of Victim Cache on Ocean_CP

Source: Primary

The previous figures illustrate the effectiveness of victim caches for reducing conflict misses across all benchmarks for all sizes. In the cases of the Raytrace and Ocean_CP benchmarks, the victim cache was most effective at smaller sizes. Larger victim caches did improve performance, but only minutely. For example, in the case of the Ocean_CP benchmark, 0 victim cache lines resulted in an L1 miss rate of 37.16%, whereas 1 victim cache line resulted in an L1 miss rate of 22.65% - a 39% improvement. A victim cache size of 7 entries resulted in an L1 miss rate of 7.70% and a victim cache size of 8 entries resulted in an L1 miss rate of 7.48% - only a 2.86% improvement. Likewise, 15 victim cache entries resulted in an L1 miss rate of 7.13% and 16 victim cache lines resulted in an L1 miss rate of 7.11% - a mere 0.28% improvement. Next, the effects of the block size will be considered. For this experiment, the L1 cache size was kept at 4KB, with a direct-mapped structure and an LRU replacement policy. The following figures showcase the results.

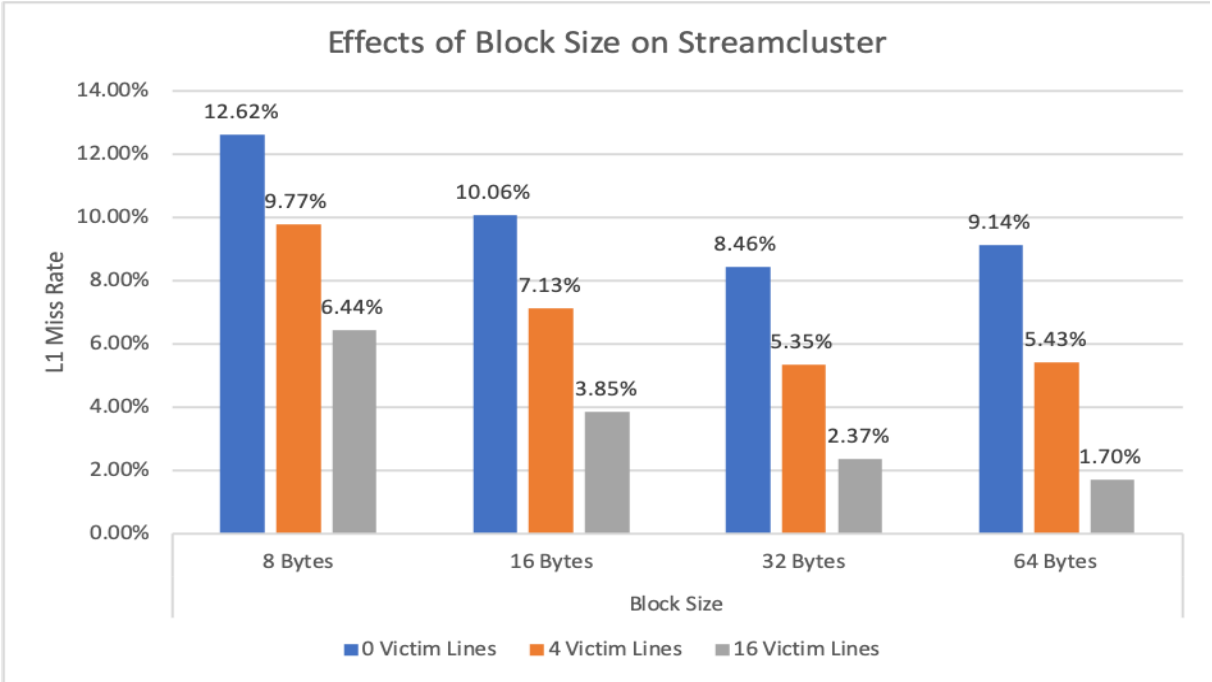

Figure 5.24: Effects of Block Size on Streamcluster

Source: Primary

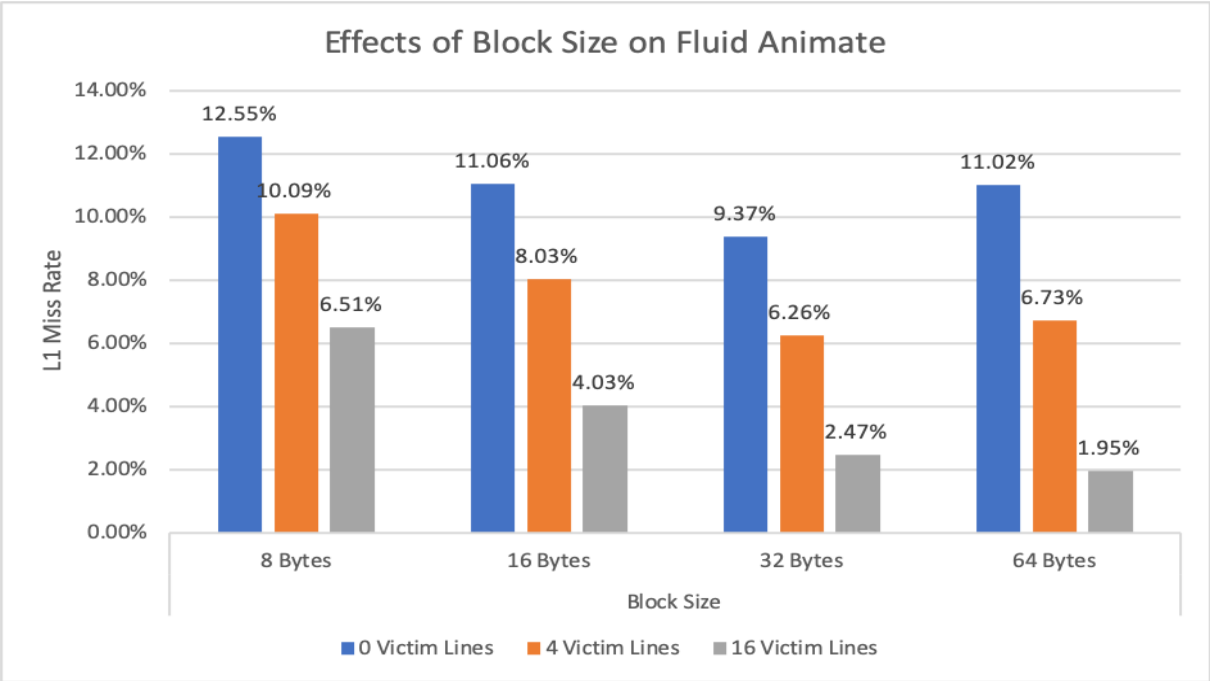

Figure 5.25: Effects of Block Size on Fluid Animate

Source: Primary

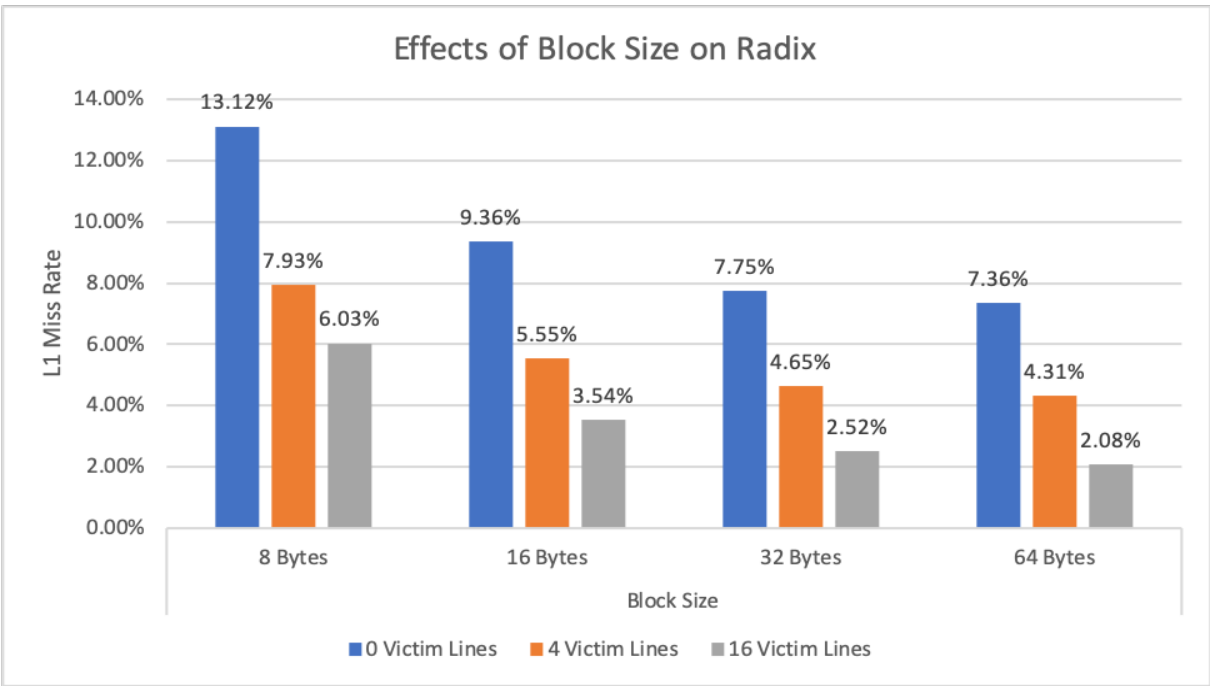

Figure 5.26: Effects of Block Size on Radix

Source: Primary

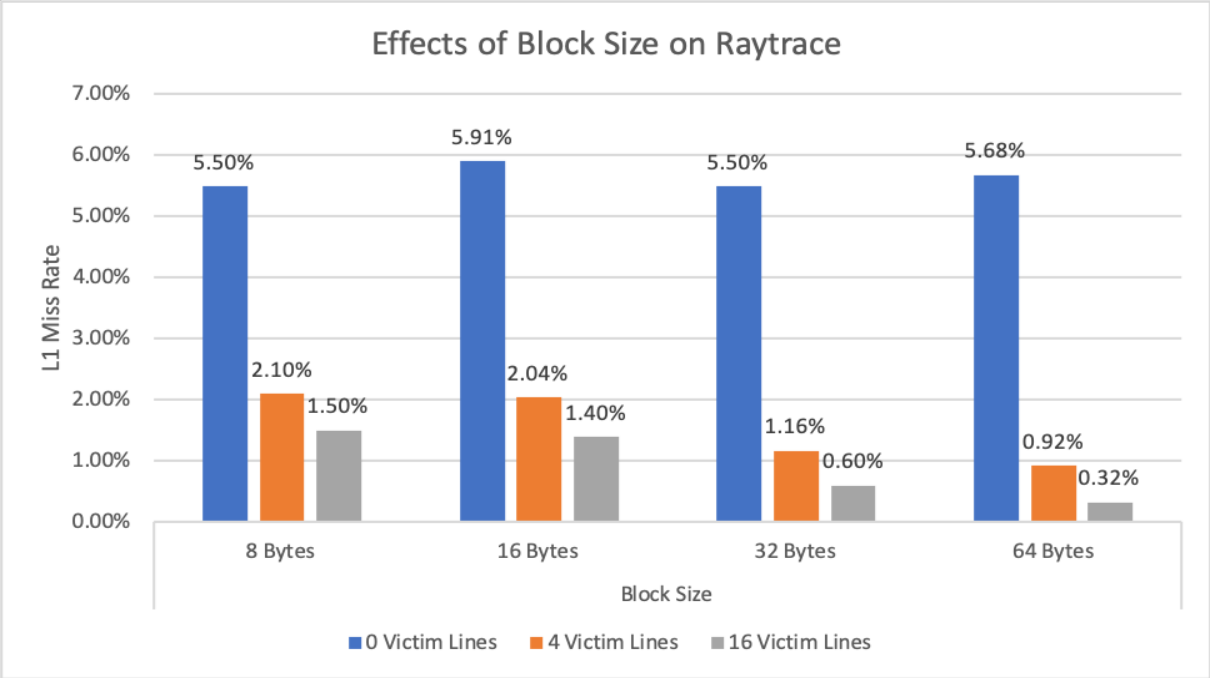

Figure 5.27: Effects of Block Size on Raytrace

Source: Primary

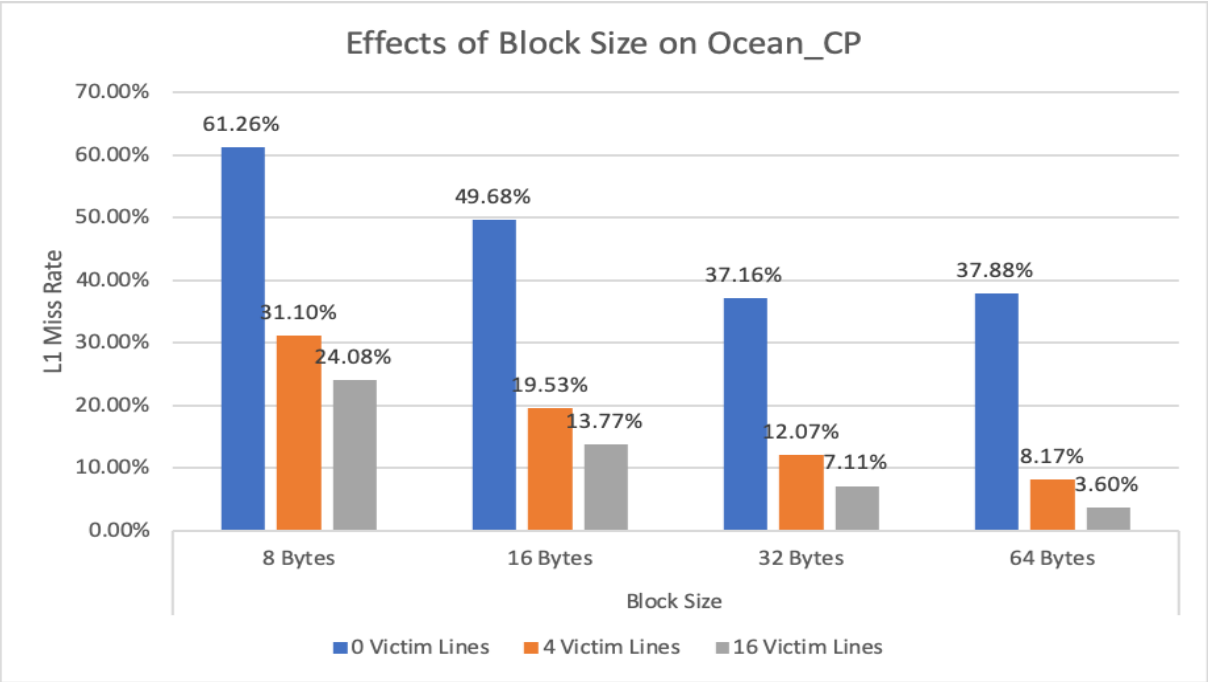

Figure 5.28: Effects of Block Size on Ocean_CP

Source: Primary

As expected, increasing the block size improved performance, however, in some instances, the block size was too large, leading to a scenario in which fewer sets of data were stored in the cache. The advent of fewer sets resulted in a greater number of conflict misses, increasing the miss rate. Next, the various replacement policies will be compared. First, simulations were done to a single-core, L1 only cache, with a cache size of 4KB, a block size of 32 bytes, and a 4-way set-associativity. Then, an experiment was done on a 16-core, L1 and L2 cache simulation utilizing the MESI coherence protocol, with the L1 cache having a size of 8KB, with a block size of 32 bytes, and a 4-way set-associativity. The L2 cache had a size of 128KB, with a block size of 64 bytes, and a 8-way set-associativity. To streamline the testing process, both L1 and L2 caches will be utilizing the same replacement policy in each simulation, i.e. both will be utilizing LRU, MRU, etc. Simulation results were taken as an average across all cores.

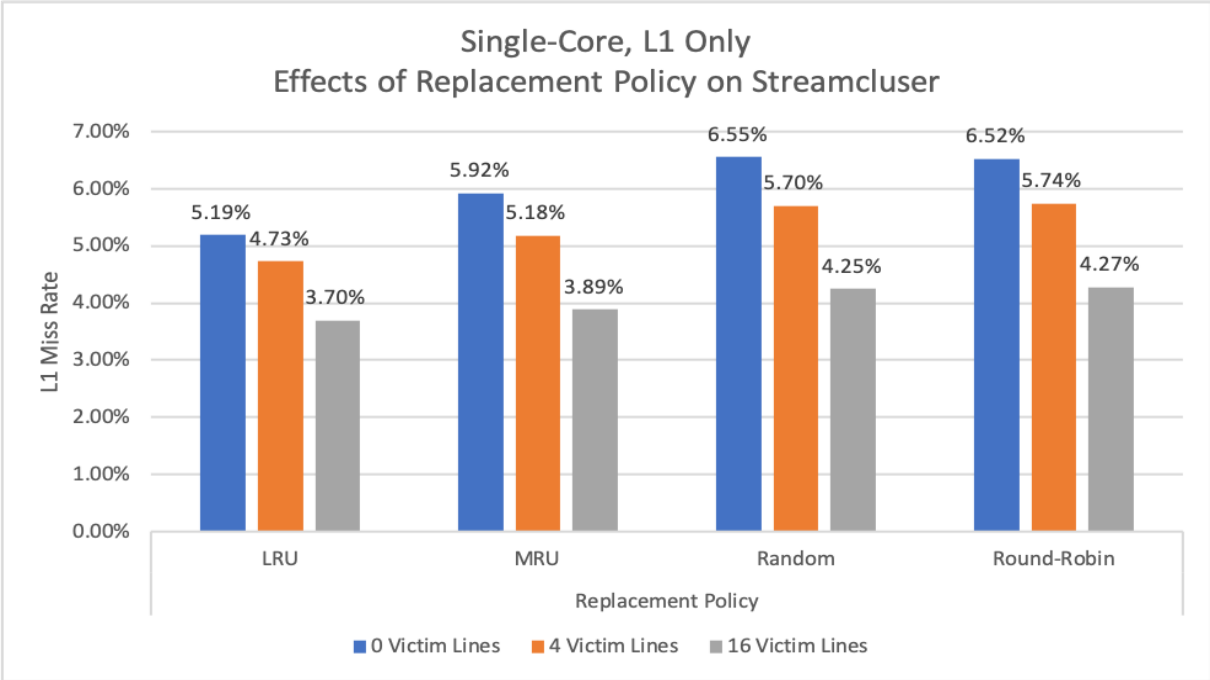

Figure 5.29: Single-Core, L1 Only Cache - Effects of Replacement Policy on Streamcluster

Source: Primary

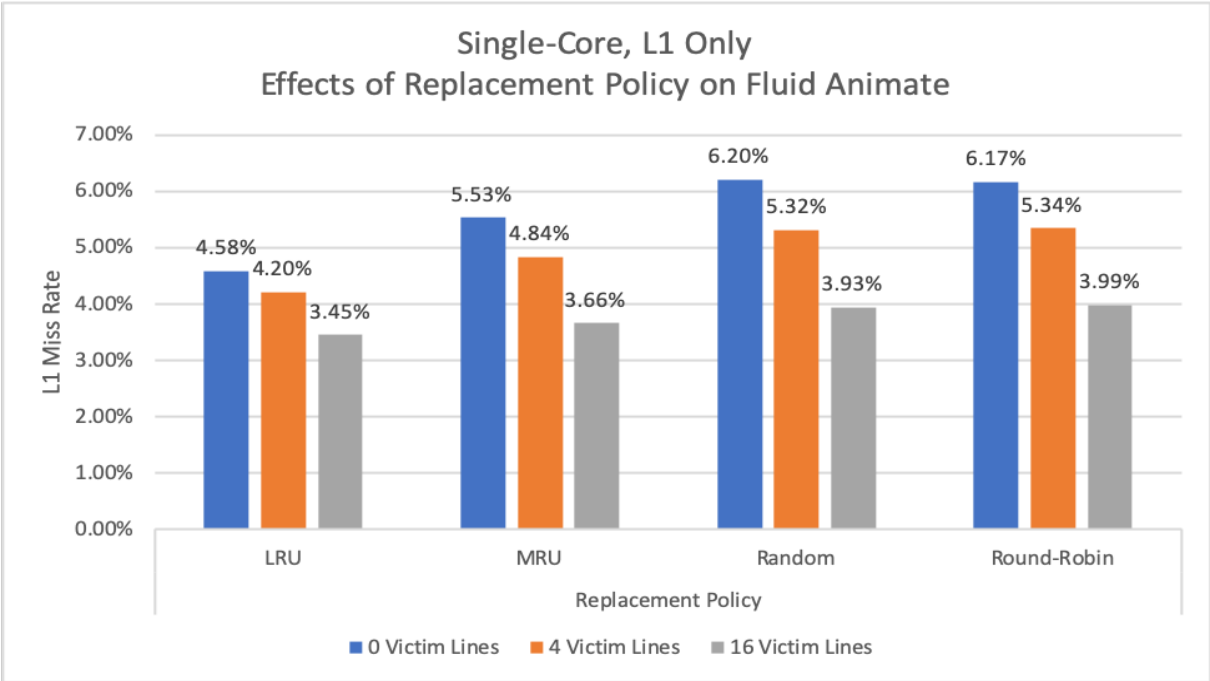

Figure 5.30: Single-Core, L1 Only Cache - Effects of Replacement Policy on Fluid Animate

Source: Primary

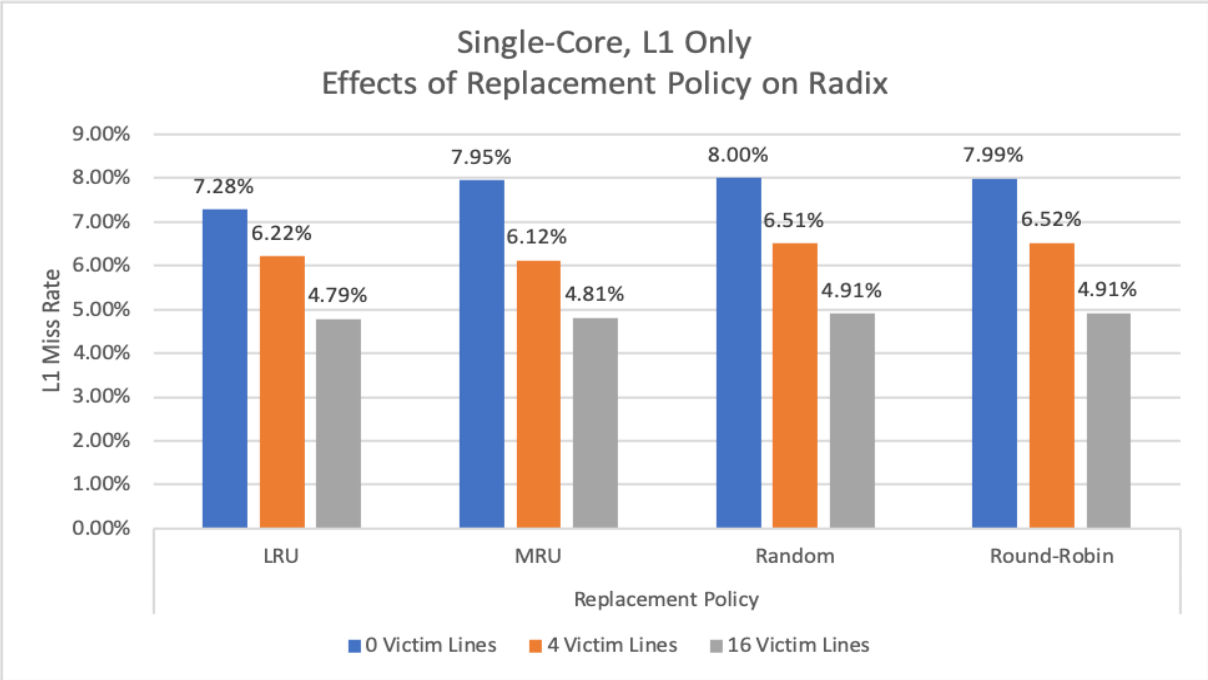

Figure 5.31: Single-Core, L1 Only Cache - Effects of Replacement Policy on Radix

Source: Primary

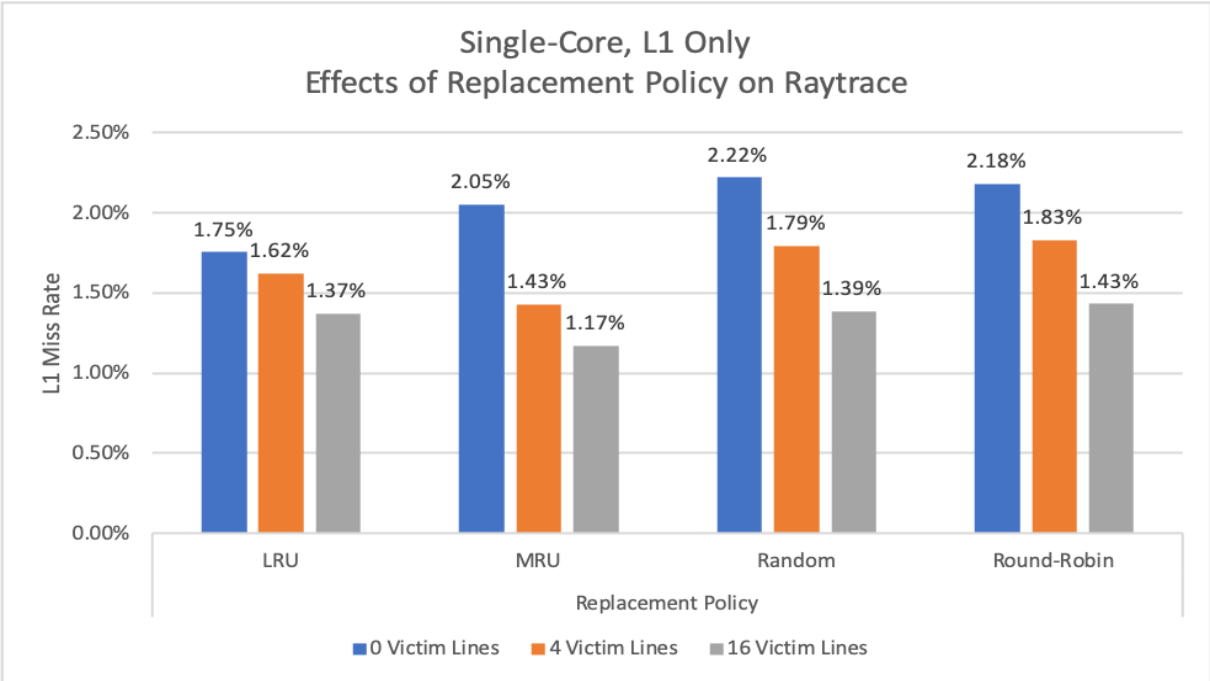

Figure 5.32: Single-Core, L1 Only Cache - Effects of Replacement Policy on Raytrace

Source: Primary

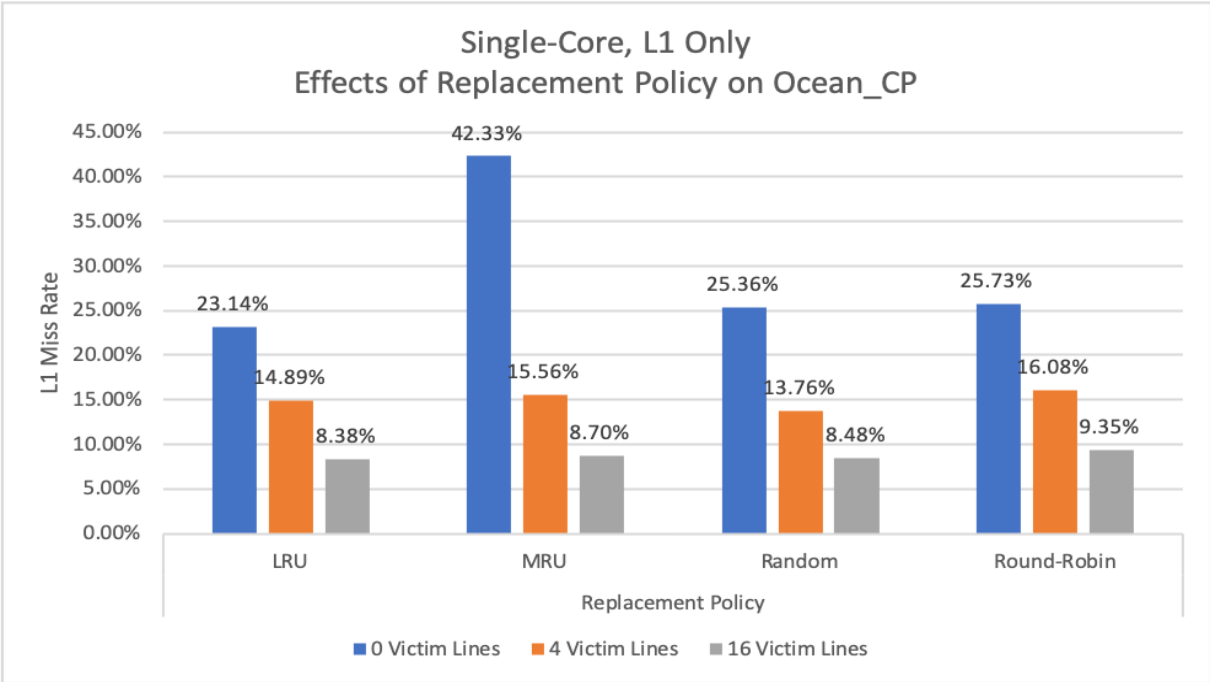

Figure 5.33: Single-Core, L1 Only Cache - Effects of Replacement Policy on Ocean_CP

Source: Primary

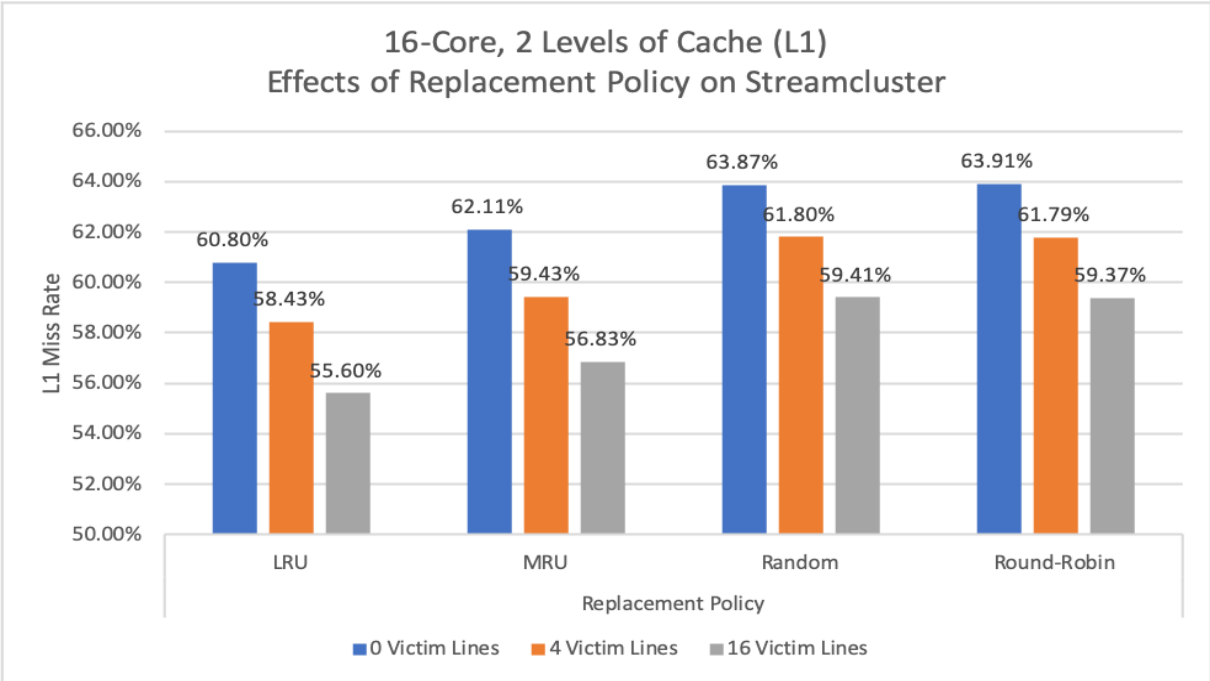

Figure 5.34: 16-Core, 2 Levels of Cache (L1) - Effects of Replacement Policy on Streamcluster

Source: Primary

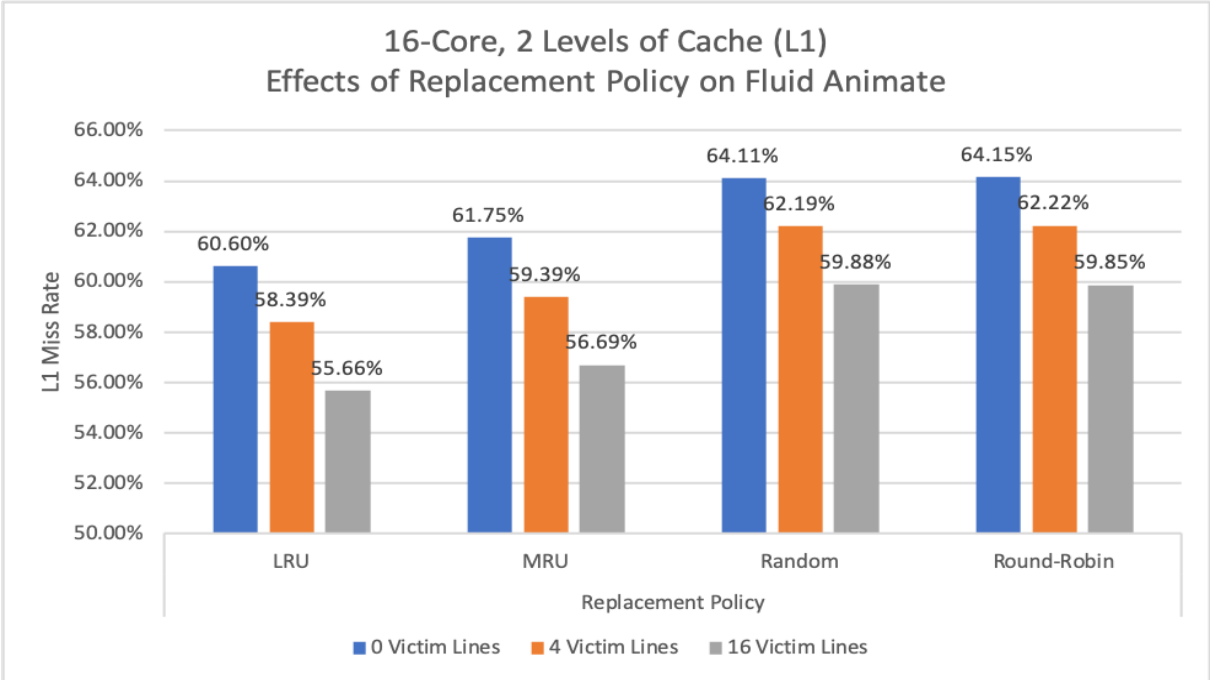

Figure 5.35: 16-Core, 2 Levels of Cache (L1) - Effects of Replacement Policy on Fluid Animate

Source: Primary

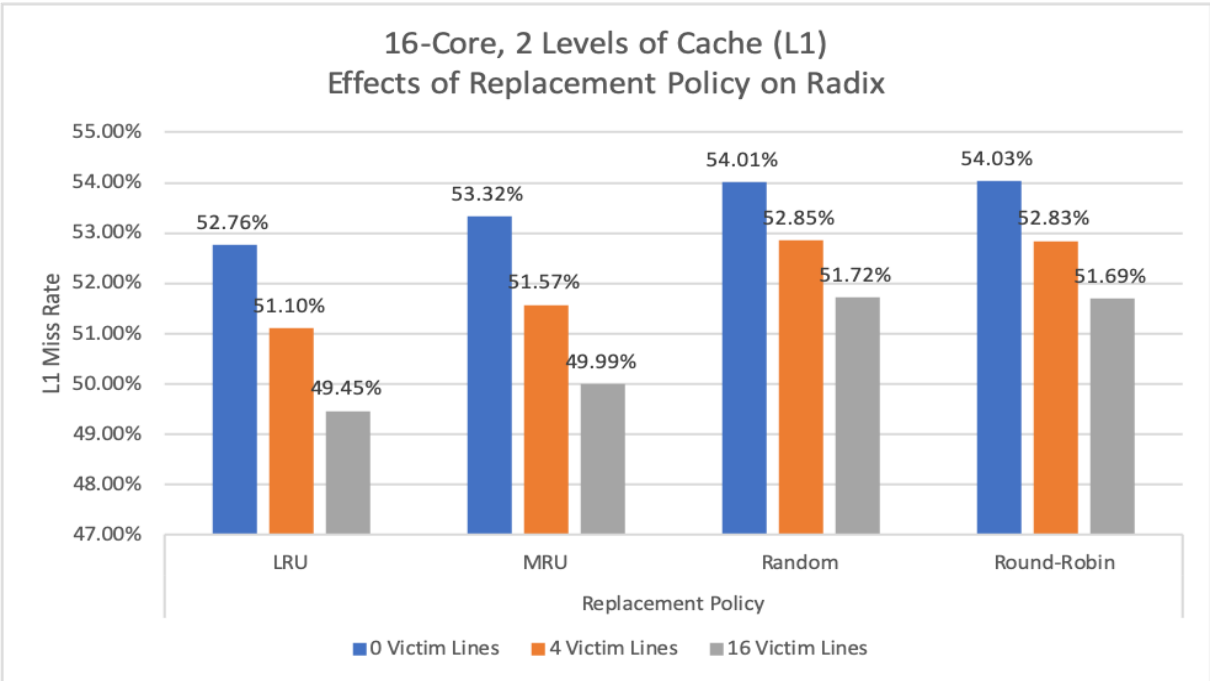

Figure 5.36: 16-Core, 2 Levels of Cache (L1) - Effects of Replacement Policy on Radix

Source: Primary

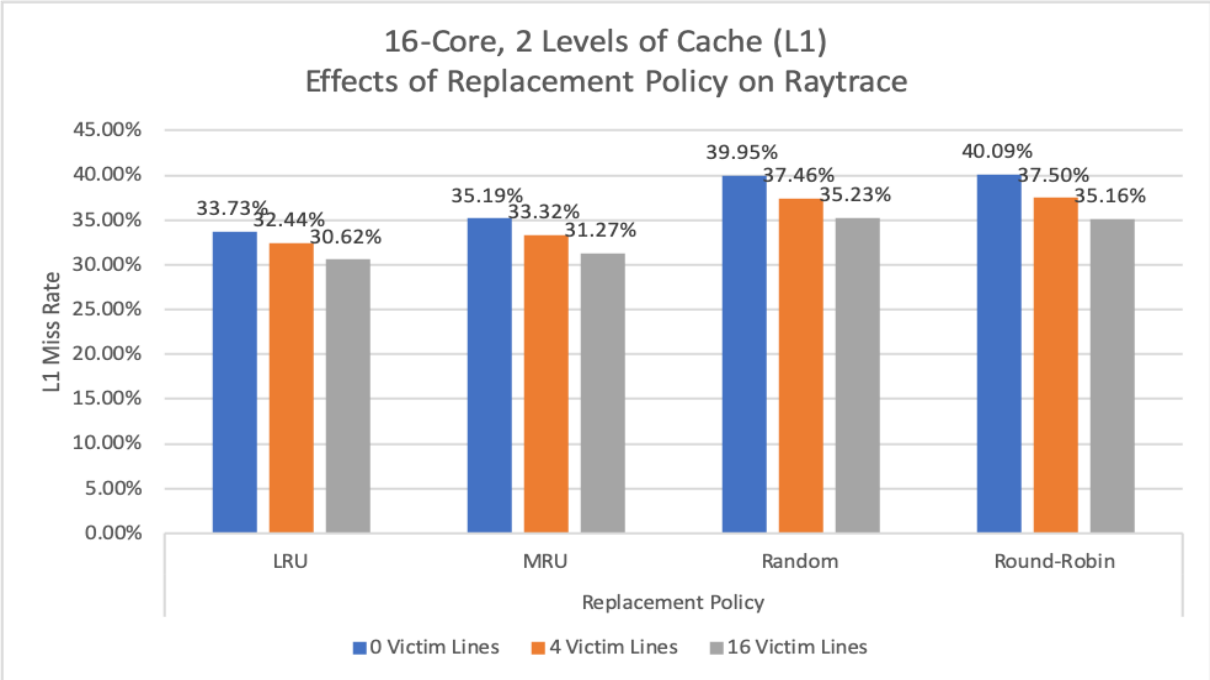

Figure 5.37: 16-Core, 2 Levels of Cache (L1) - Effects of Replacement Policy on Raytrace

Source: Primary

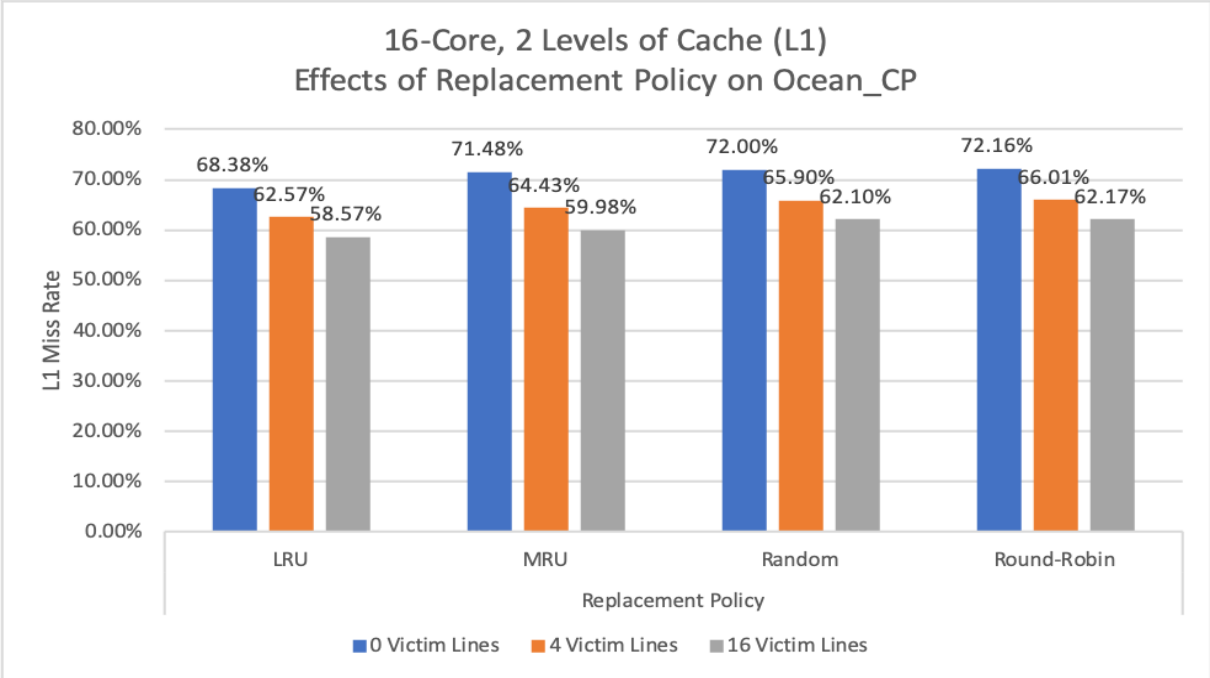

Figure 5.38: 16-Core, 2 Levels of Cache (L1) - Effects of Replacement Policy on Ocean_CP

Source: Primary

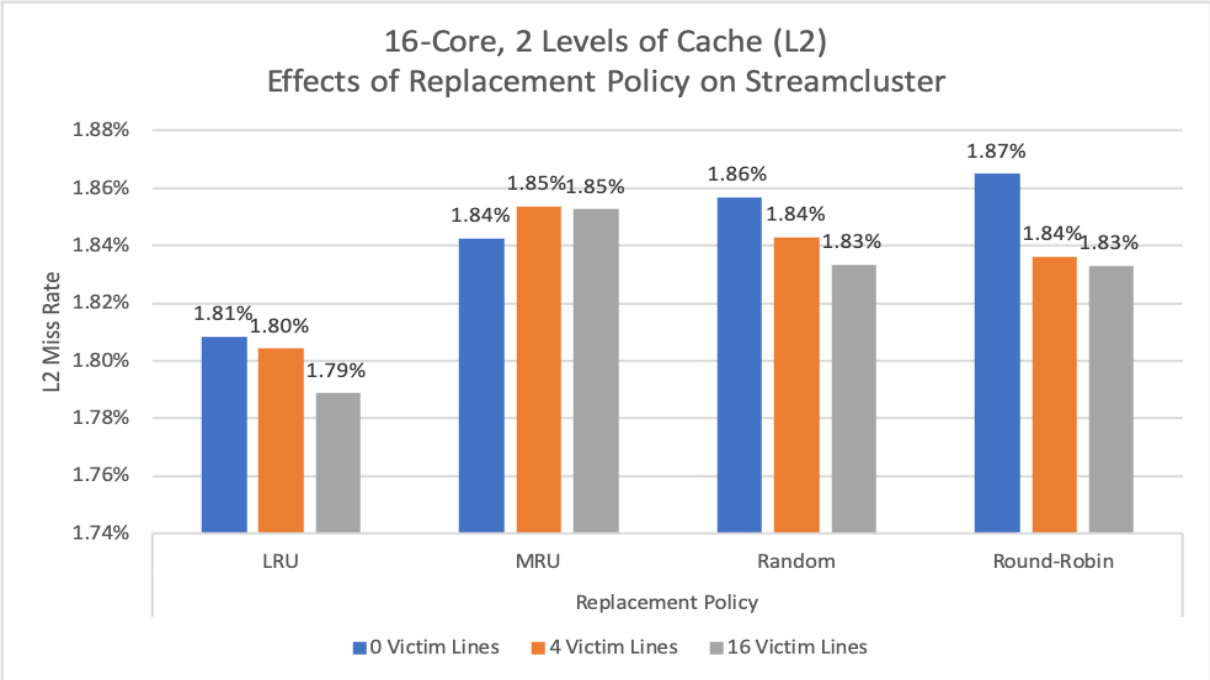

Figure 5.39: 16-Core, 2 Levels of Cache (L2) - Effects of Replacement Policy on Streamcluster

Source: Primary

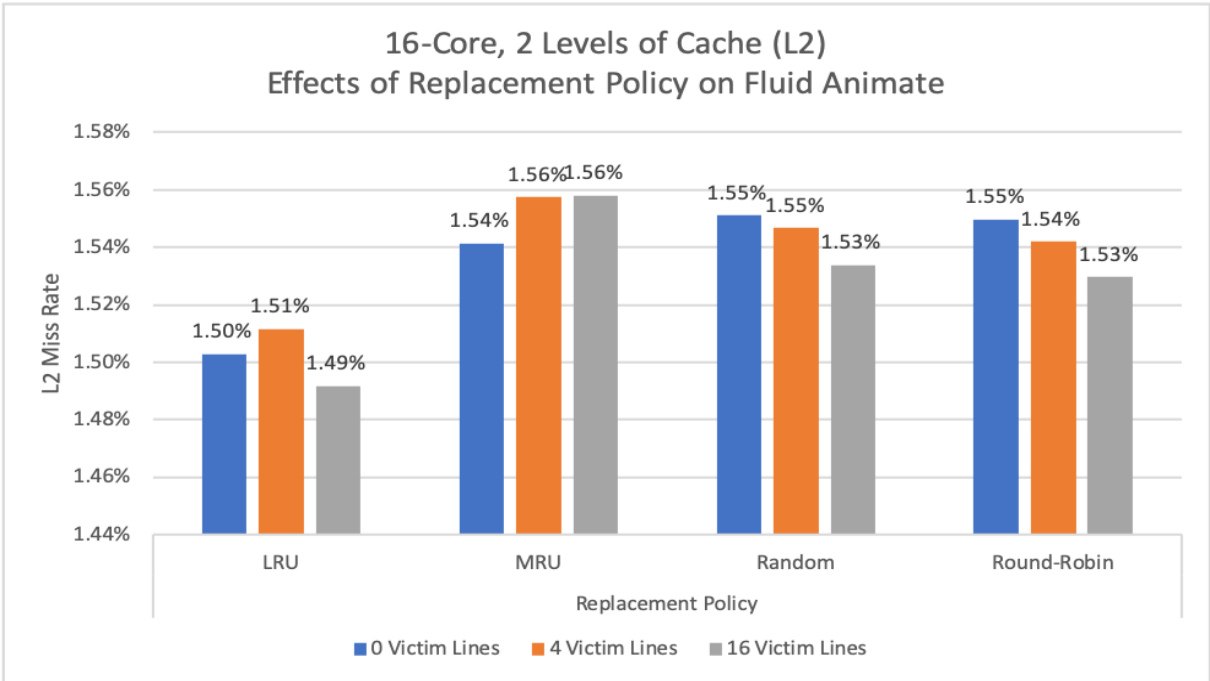

Figure 5.40: 16-Core, 2 Levels of Cache (L2) - Effects of Replacement Policy on Fluid Animate

Source: Primary

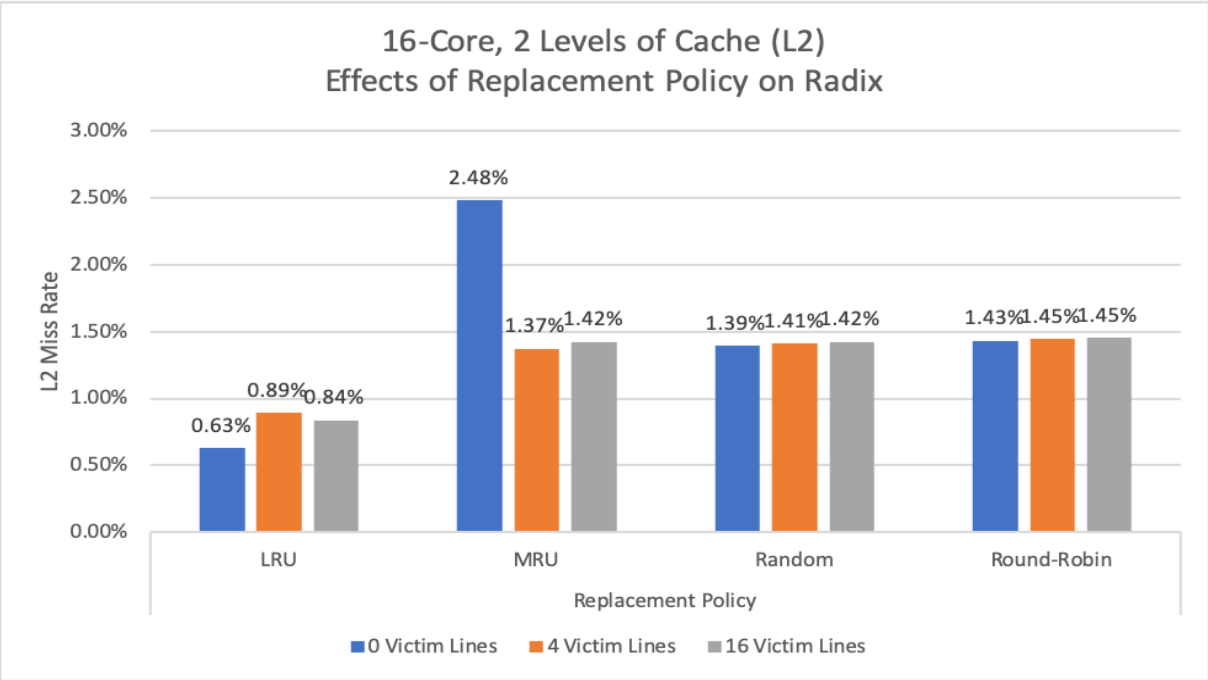

Figure 5.41: 16-Core, 2 Levels of Cache (L2) - Effects of Replacement Policy on Radix

Source: Primary

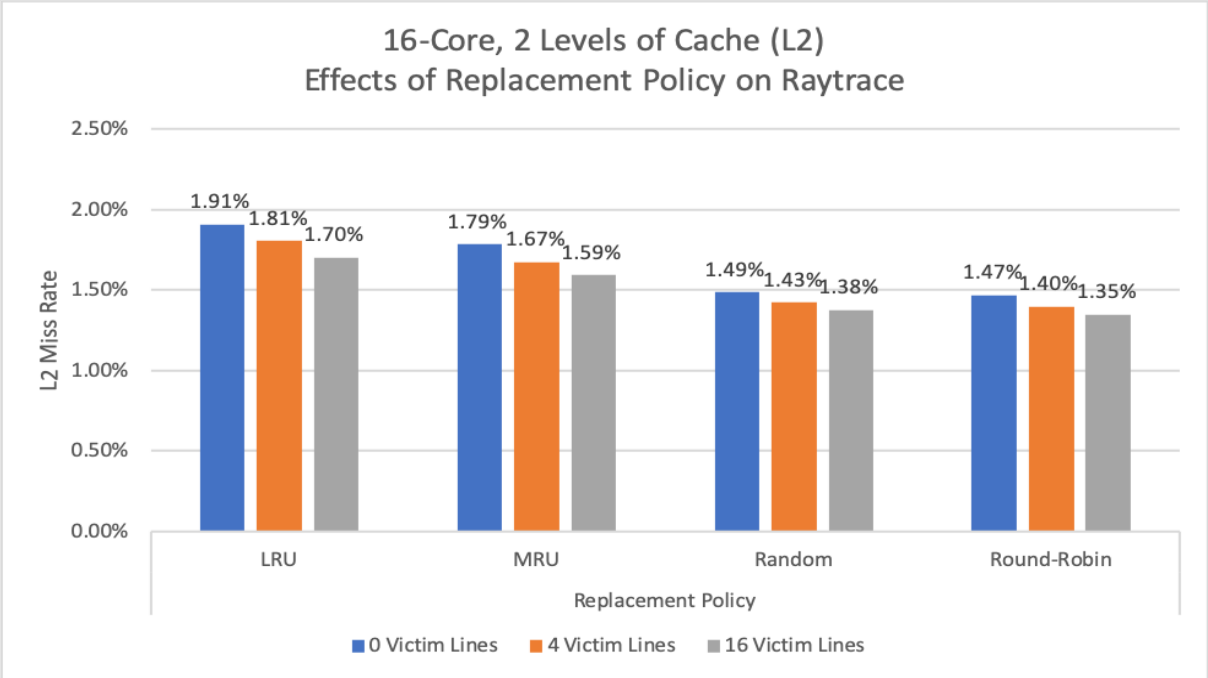

Figure 5.42: 16-Core, 2 Levels of Cache (L2) - Effects of Replacement Policy on Raytrace

Source: Primary

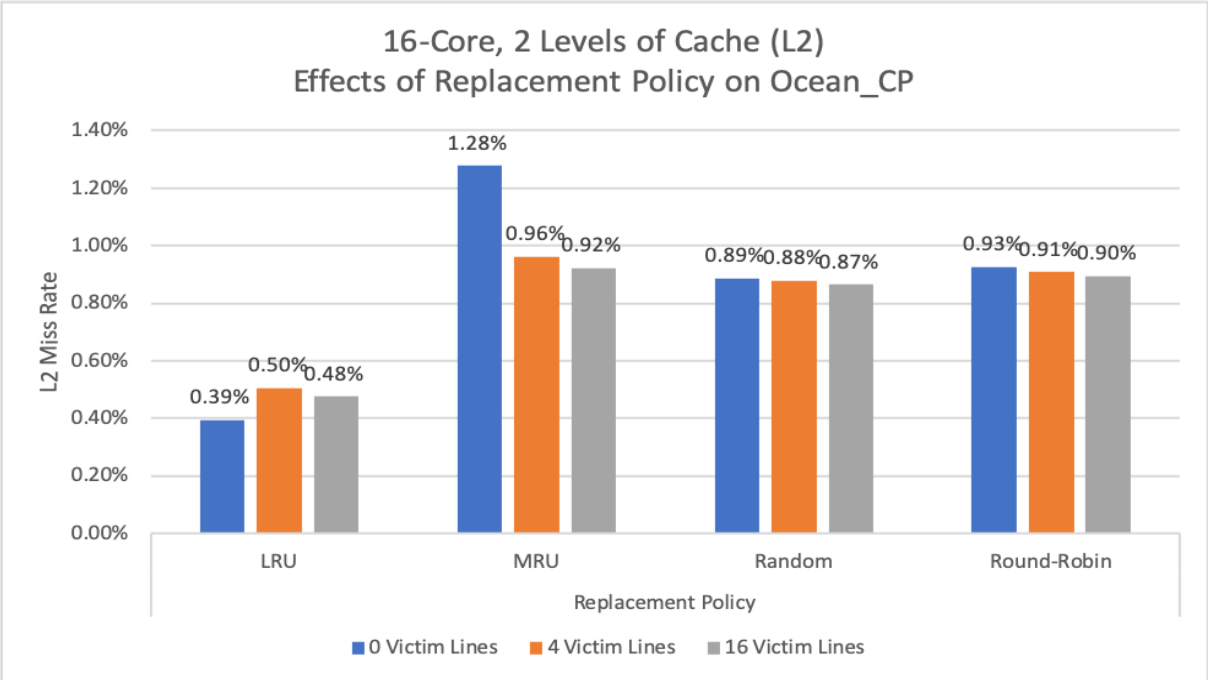

Figure 5.43: 16-Core, 2 Levels of Cache (L2) - Effects of Replacement Policy on Ocean_CP

Source: Primary

The results for the effects of replacement policies indicate that least recently used is the most effective at reducing the miss rate, however, there are instances where it is slightly outperformed, both with and without the inclusion of a victim cache. Next, the effects of the coherence protocol used will be observed. These results were gathered from simulating a 16-core, two level cache structure, where the L1 cache had a size of 4KB, a block size of 32 bytes, direct-mapped associativity, and LRU replacement. The L2 cache had a size of 128KB, a block size of 64 bytes, 4-way set-associativity, and utilized the LRU replacement policy.

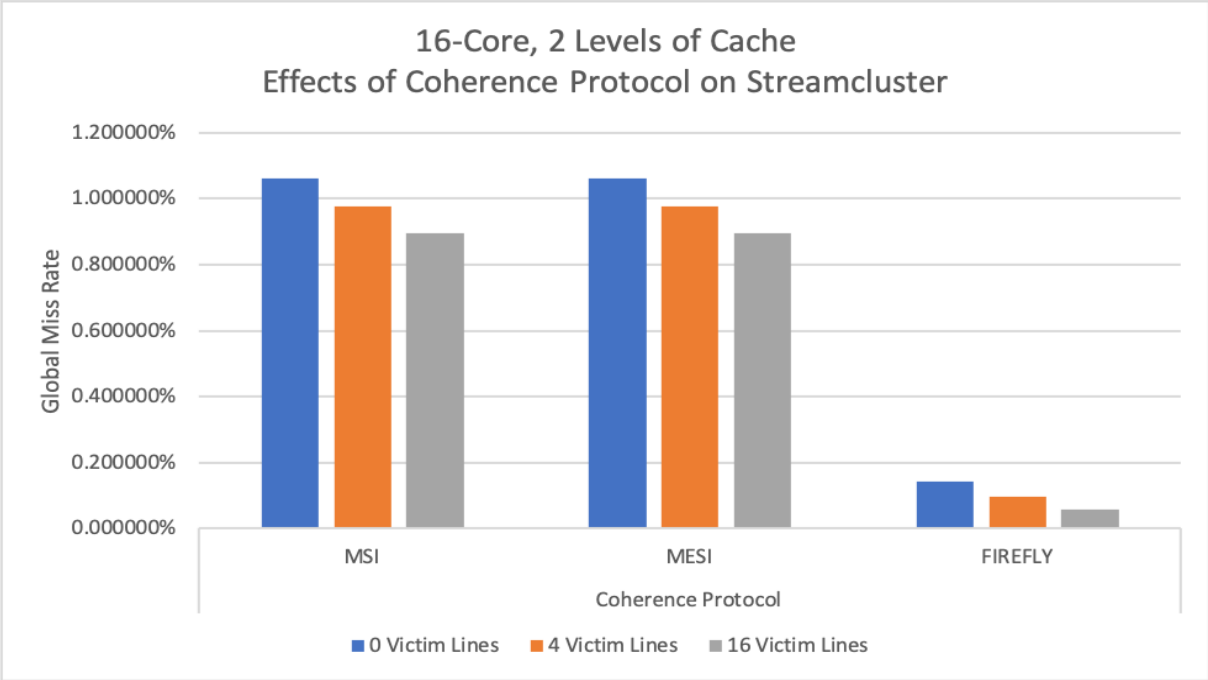

Figure 5.44: 16-Core, 2 Levels of Cache - Effects of Coherence Protocol on Streamcluster

Source: Primary

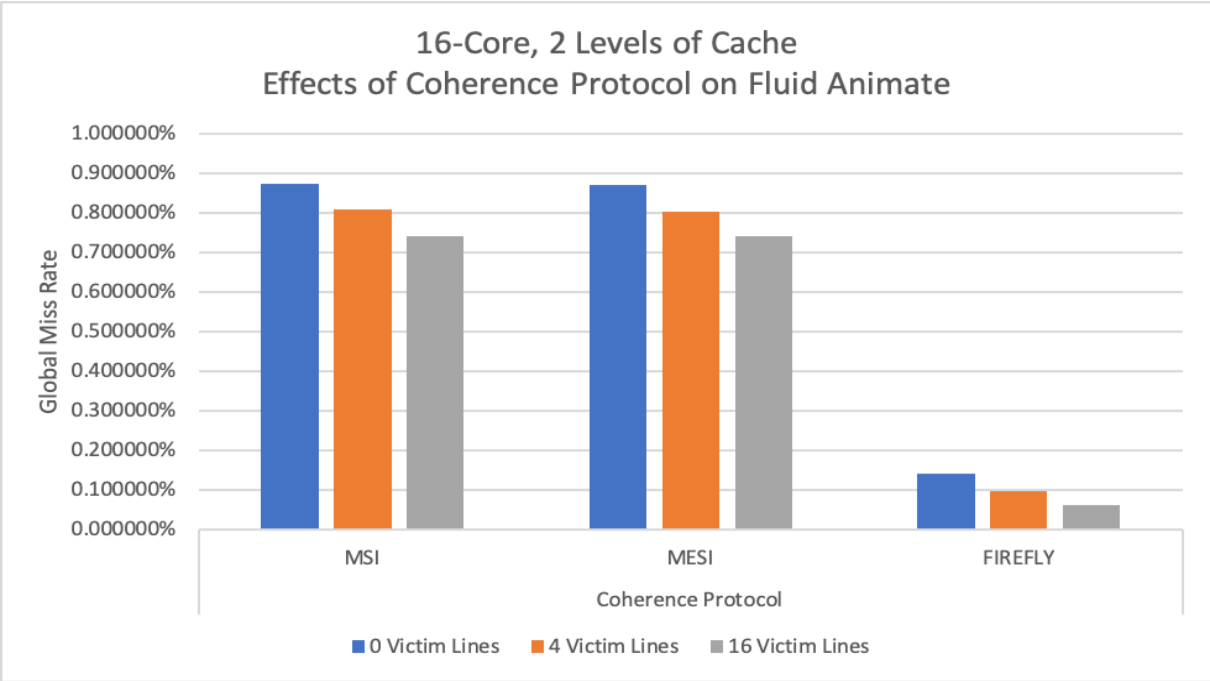

Figure 5.45: 16-Core, 2 Levels of Cache - Effects of Coherence Protocol on Fluid Animate

Source: Primary

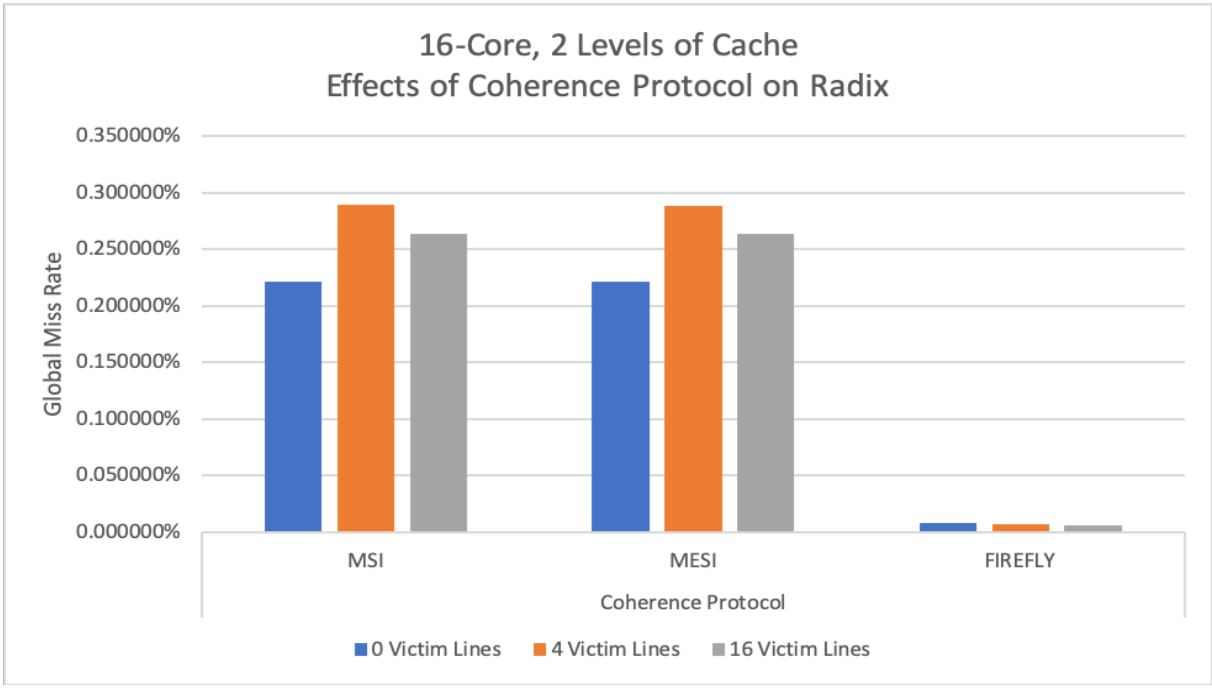

Figure 5.46: 16-Core, 2 Levels of Cache - Effects of Coherence Protocol on Radix

Source: Primary

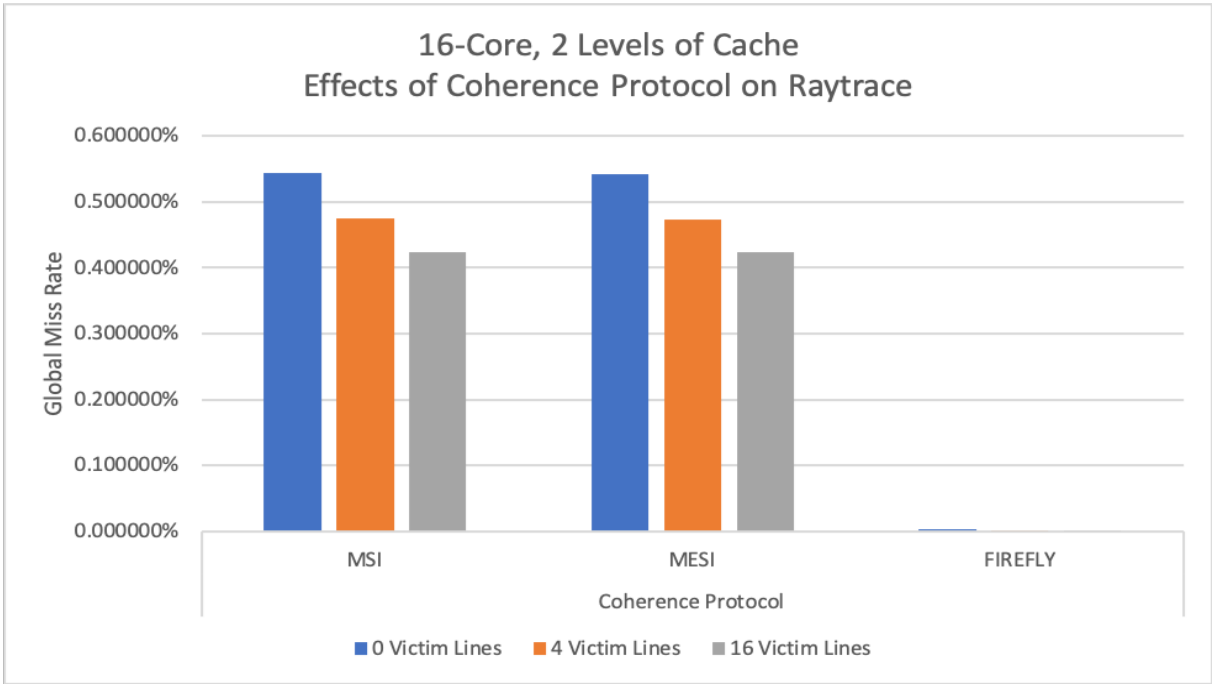

Figure 5.47: 16-Core, 2 Levels of Cache - Effects of Coherence Protocol on Raytrace

Source: Primary

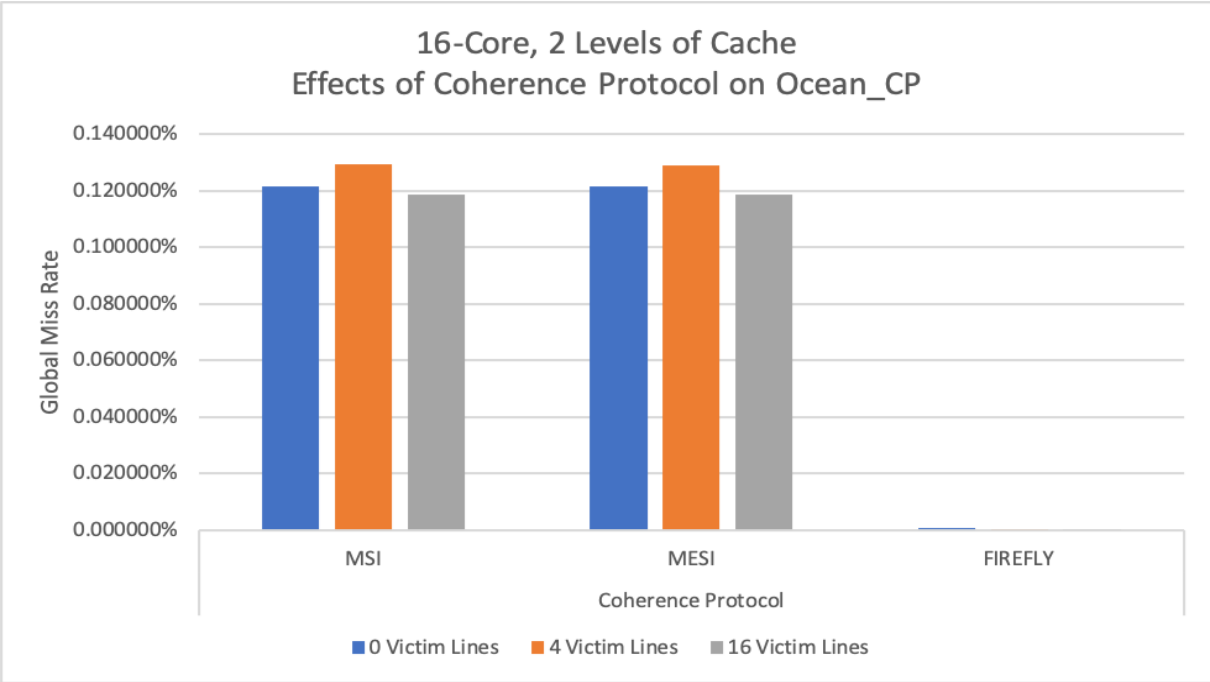

Figure 5.48: 16-Core, 2 Levels of Cache - Effects of Coherence Protocol on Ocean_CP

Source: Primary

Next, the effects of the victim cache on bus traffic will be assessed. For this experiment, the L1 cache size is kept at 2KB with a block size of 16 bytes, direct-mapped associativity, and the least recently used replacement policy. The L2 cache varies between 128KB and 1MB with a block size of 32 bytes, 4-way set-associativity, and the least recently used replacement policy. Additionally, the simulations were done on a 16-core system and for each of the coherence protocols, i.e. MSI, MESI, and FIREFLY.

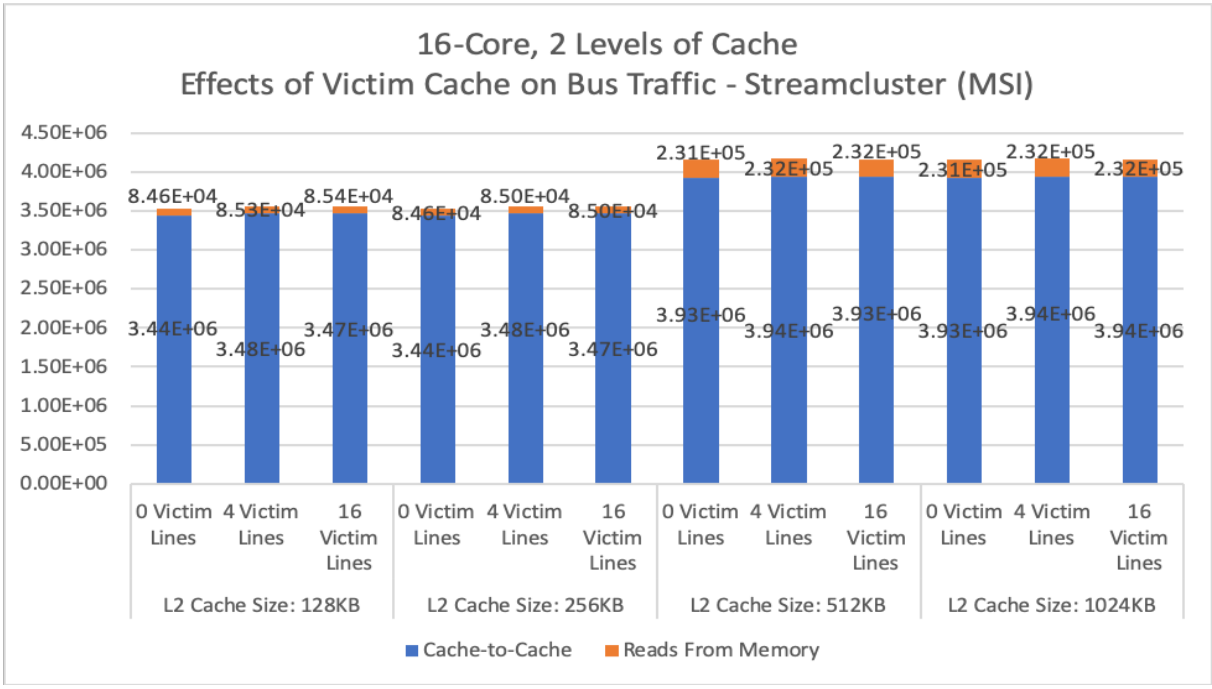

Figure 5.49: Effects of Victim Cache on Bus Traffic - Streamcluster (MSI)

Source: Primary

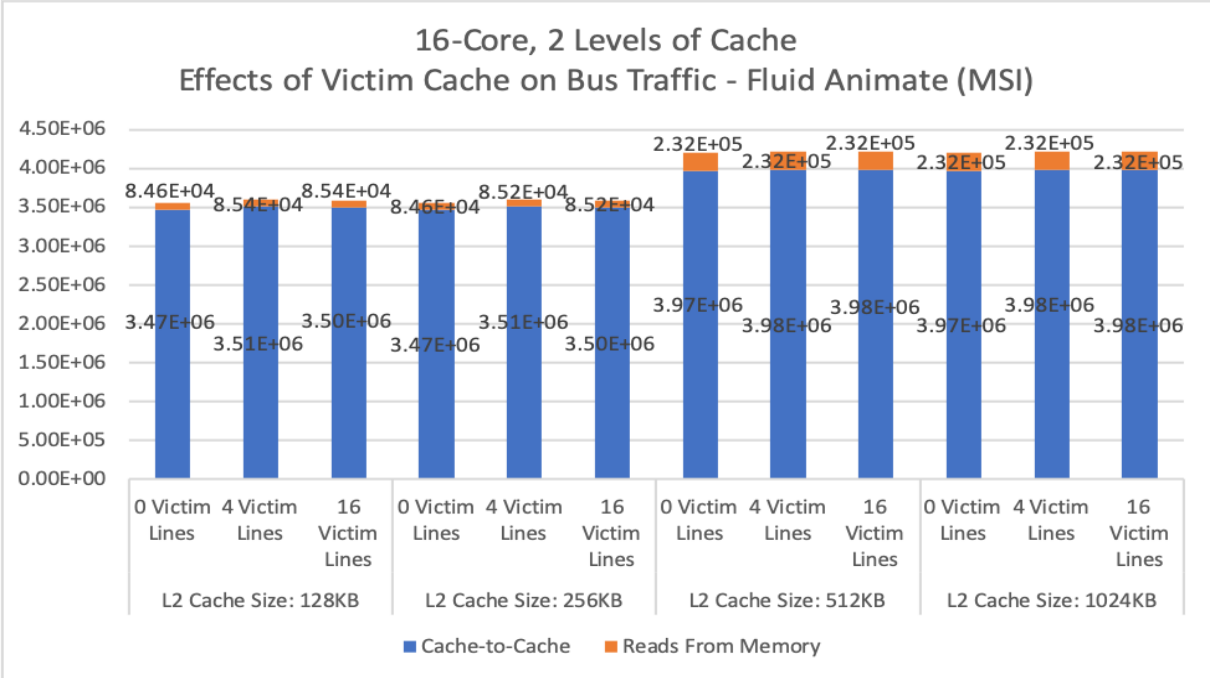

Figure 5.50: Effects of Victim Cache on Bus Traffic - Fluid Animate (MSI)

Source: Primary

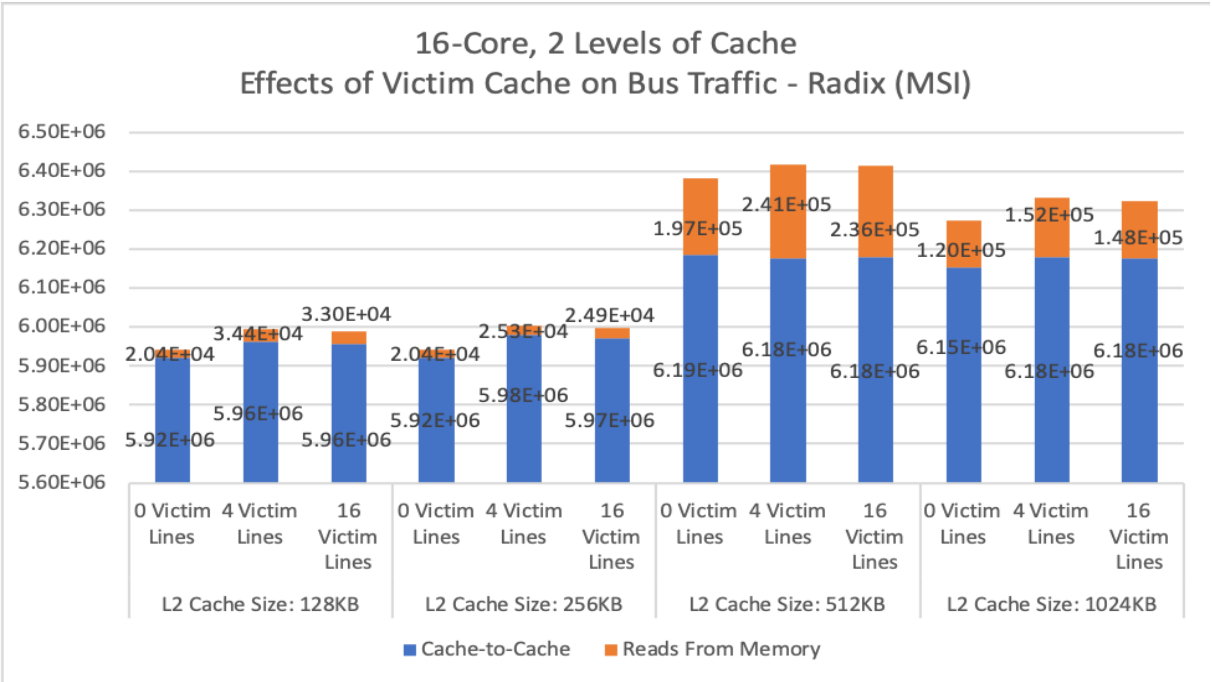

Figure 5.51: Effects of Victim Cache on Bus Traffic - Radix (MSI)

Source: Primary

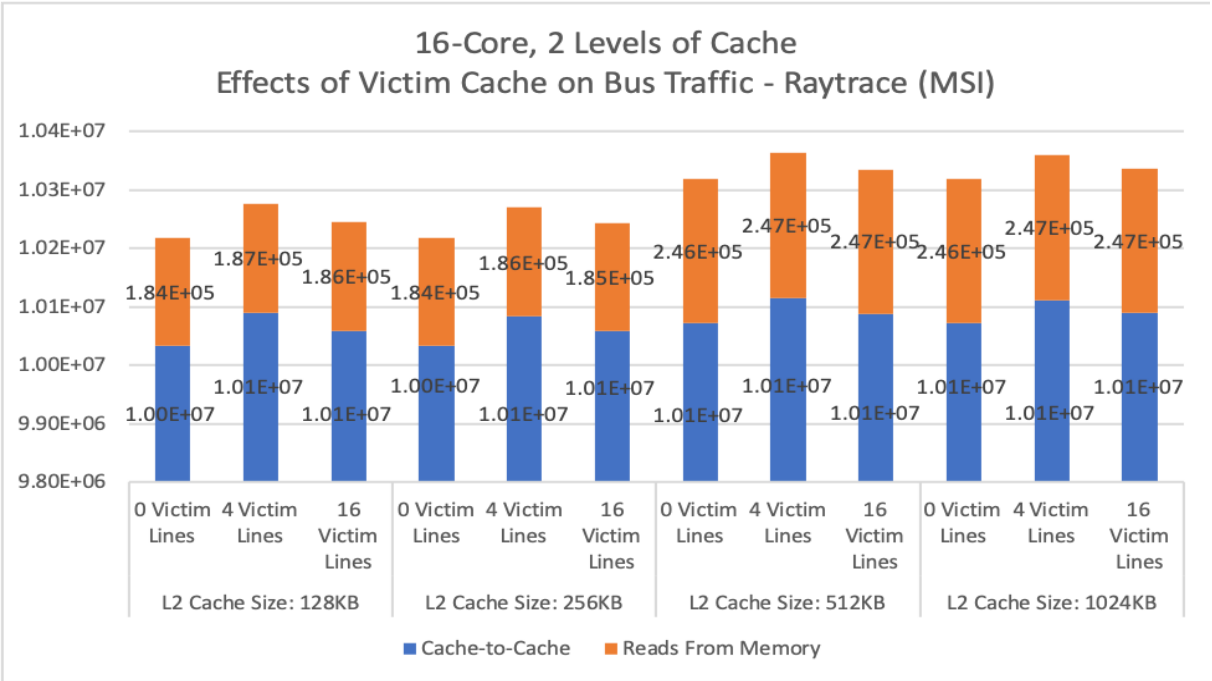

Figure 5.52: Effects of Victim Cache on Bus Traffic - Raytrace (MSI)

Source: Primary

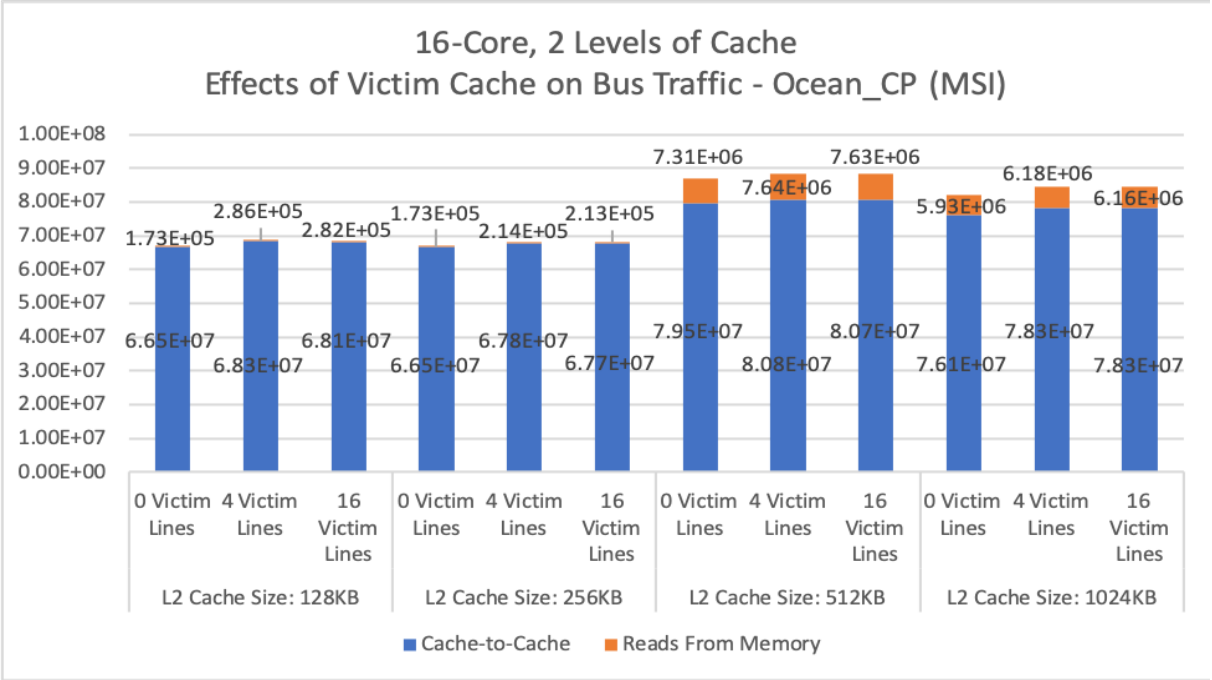

Figure 5.53: Effects of Victim Cache on Bus Traffic - Ocean_CP (MSI)

Source: Primary

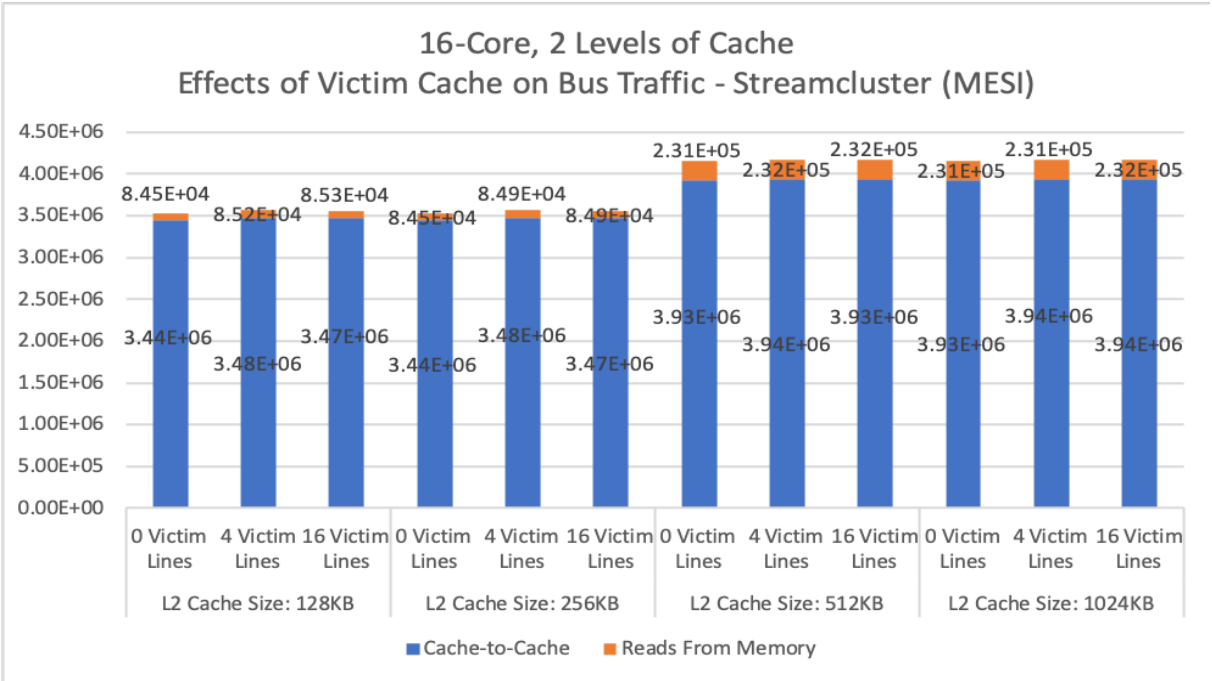

Figure 5.54: Effects of Victim Cache on Bus Traffic - Streamcluster (MESI)

Source: Primary

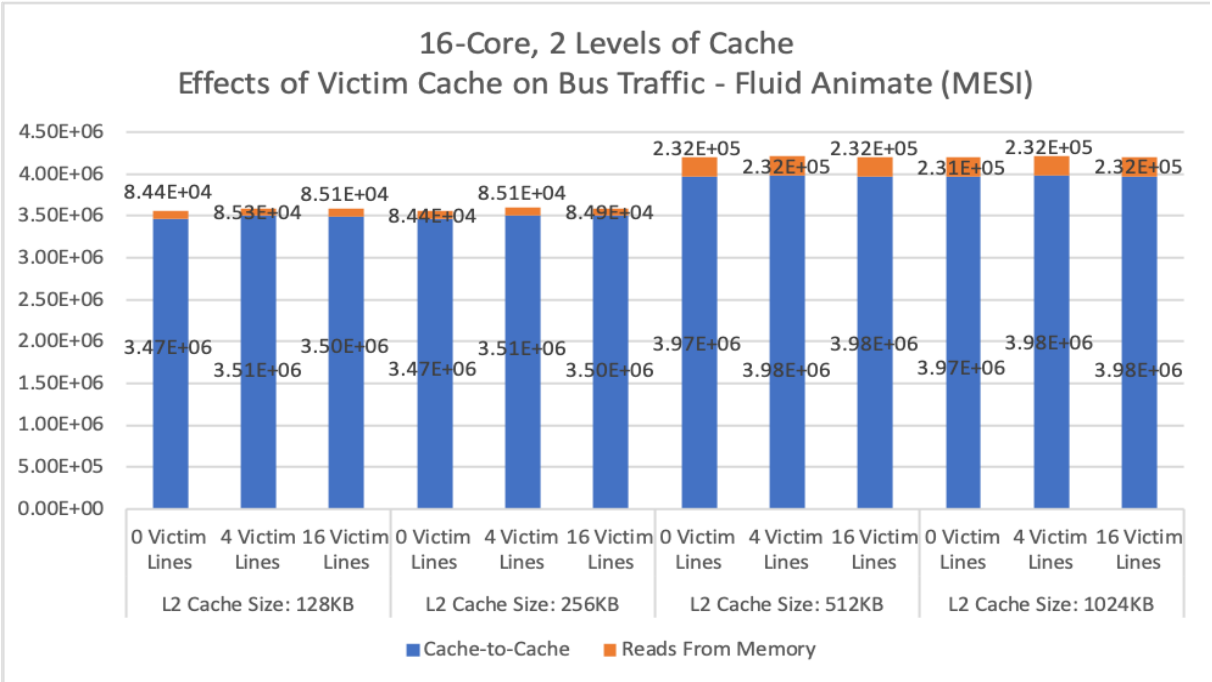

Figure 5.55: Effects of Victim Cache on Bus Traffic - Fluid Animate (MESI)

Source: Primary

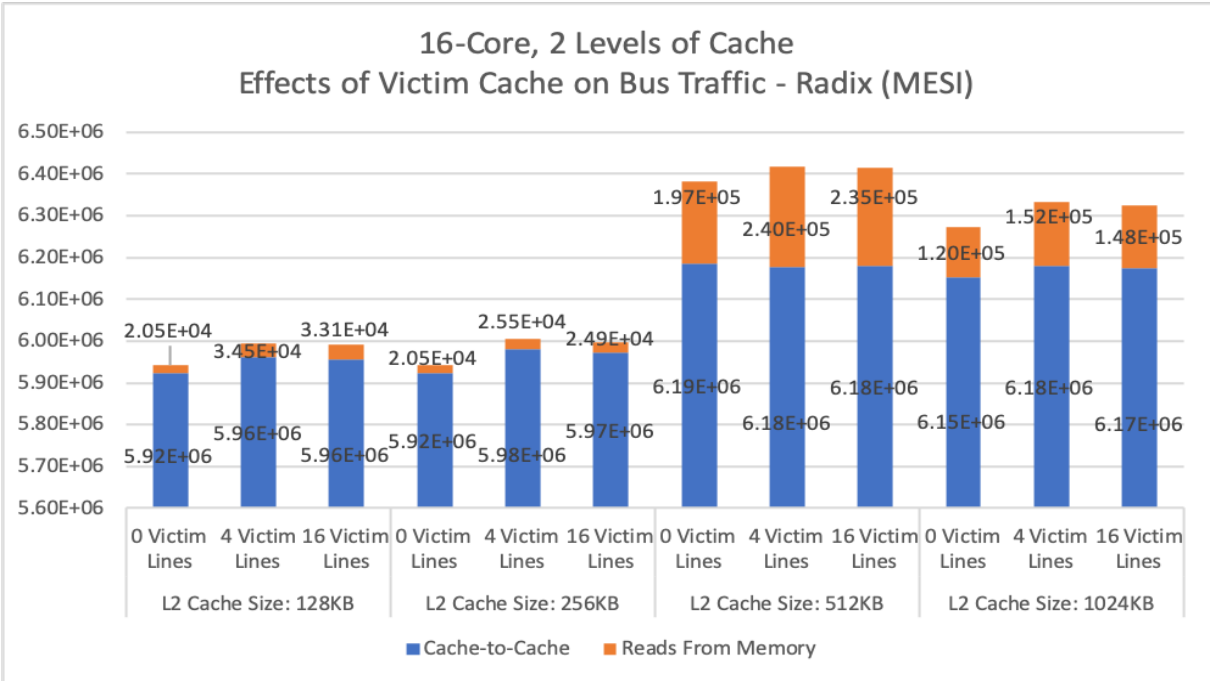

Figure 5.56: Effects of Victim Cache on Bus Traffic - Radix (MESI)

Source: Primary

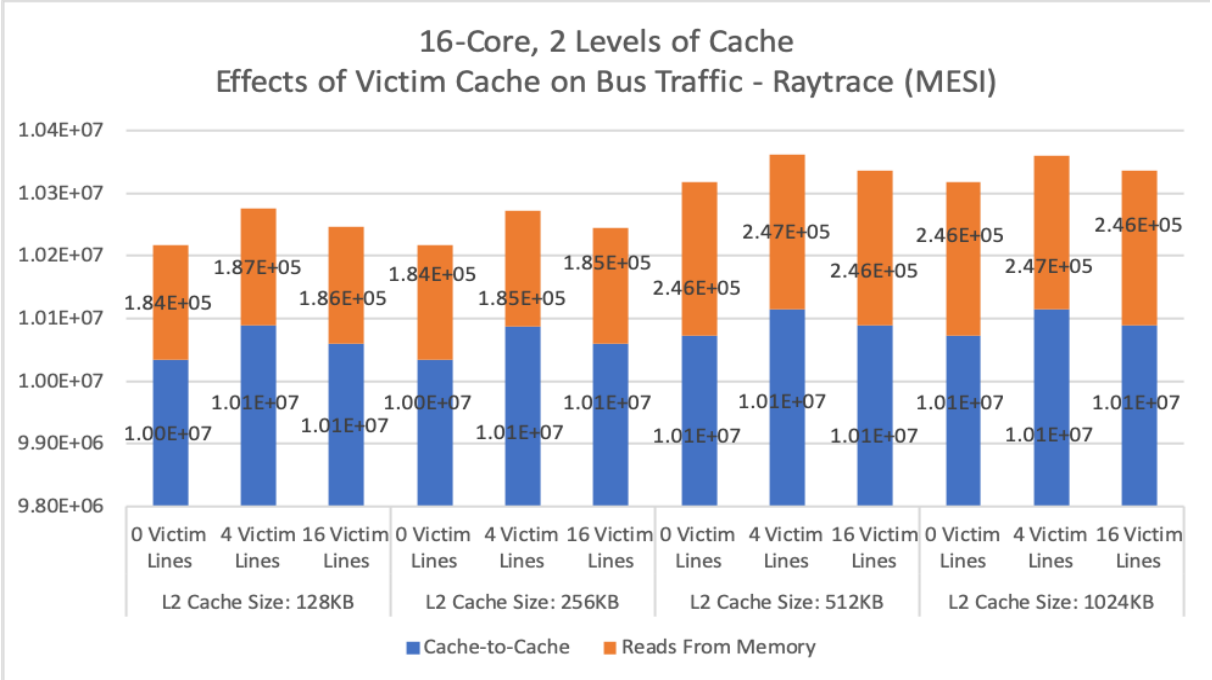

Figure 5.57: Effects of Victim Cache on Bus Traffic - Raytrace (MESI)

Source: Primary

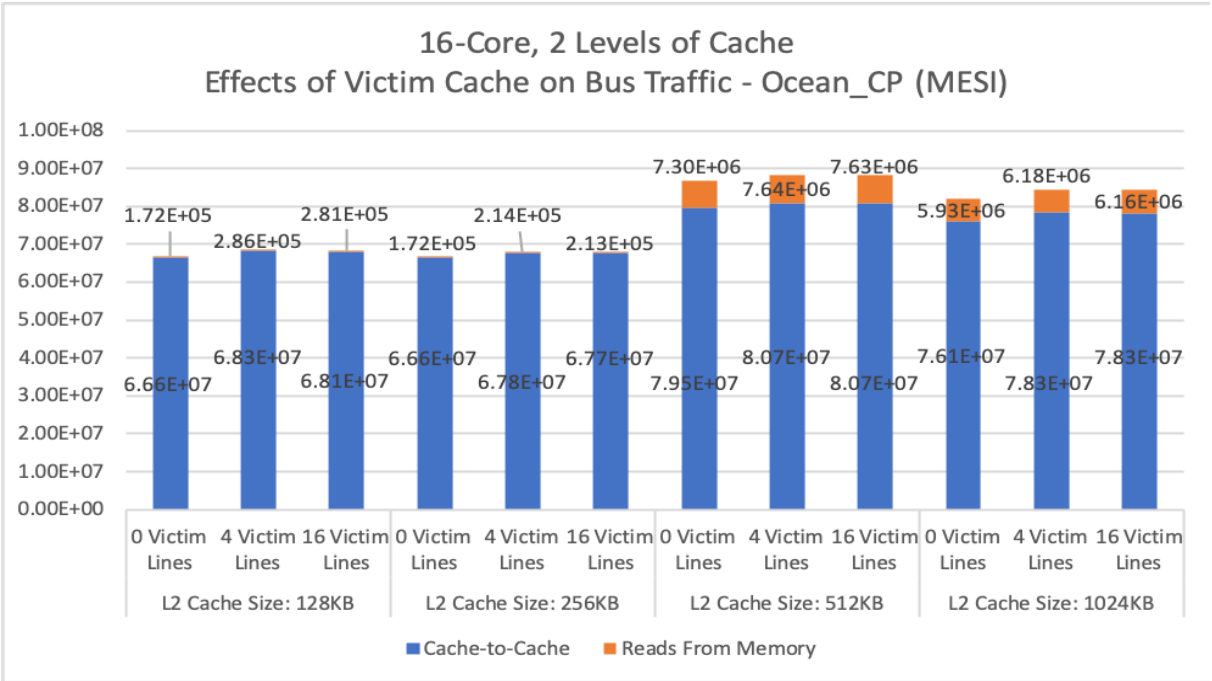

Figure 5.58: Effects of Victim Cache on Bus Traffic - Ocean_CP (MESI)

Source: Primary

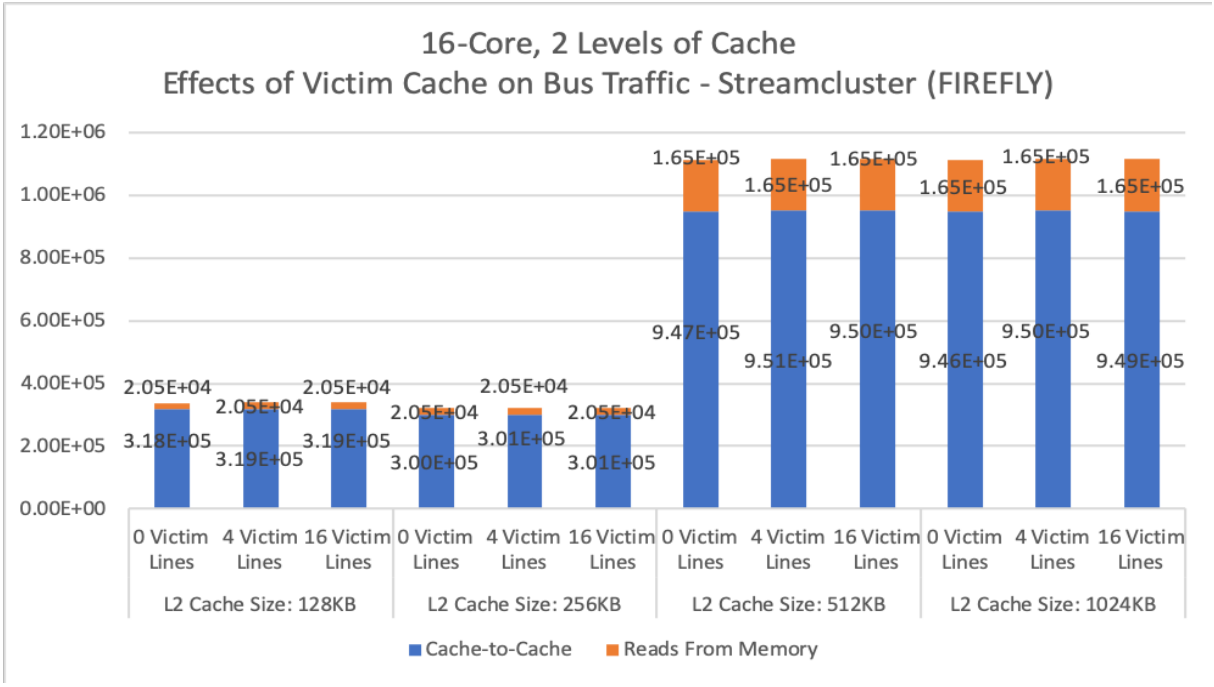

Figure 5.59: Effects of Victim Cache on Bus Traffic - Streamcluster (FIREFLY)

Source: Primary

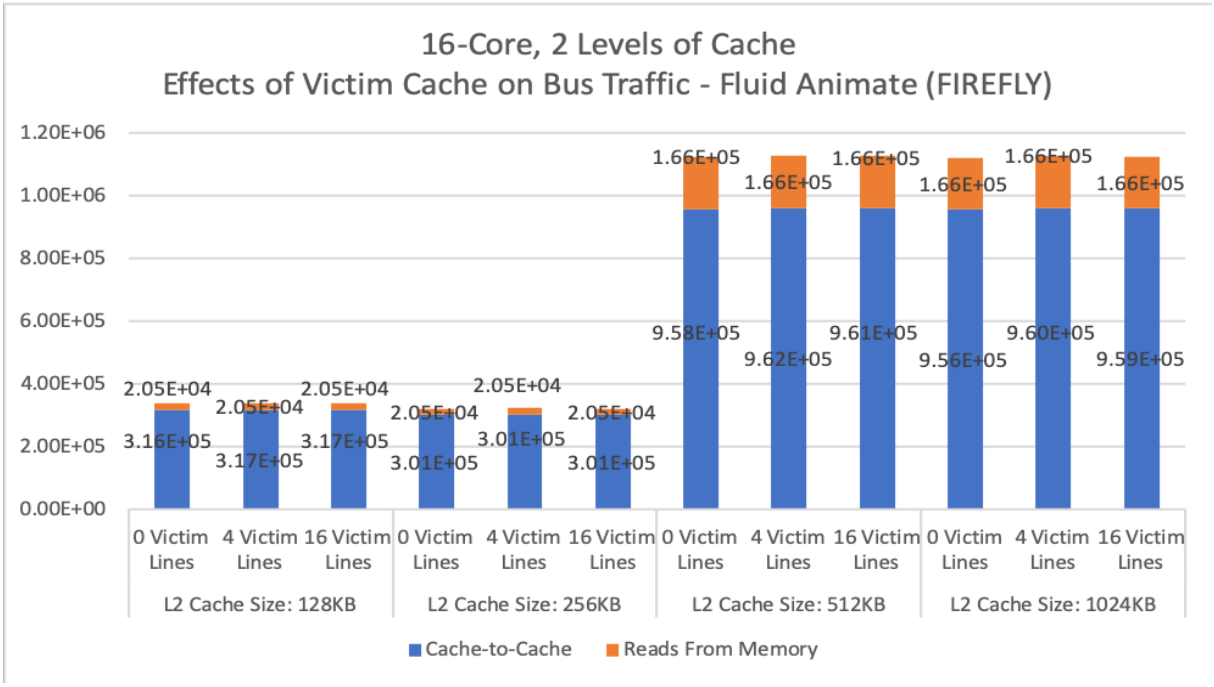

Figure 5.60: Effects of Victim Cache on Bus Traffic - Fluid Animate (FIREFLY)

Source: Primary

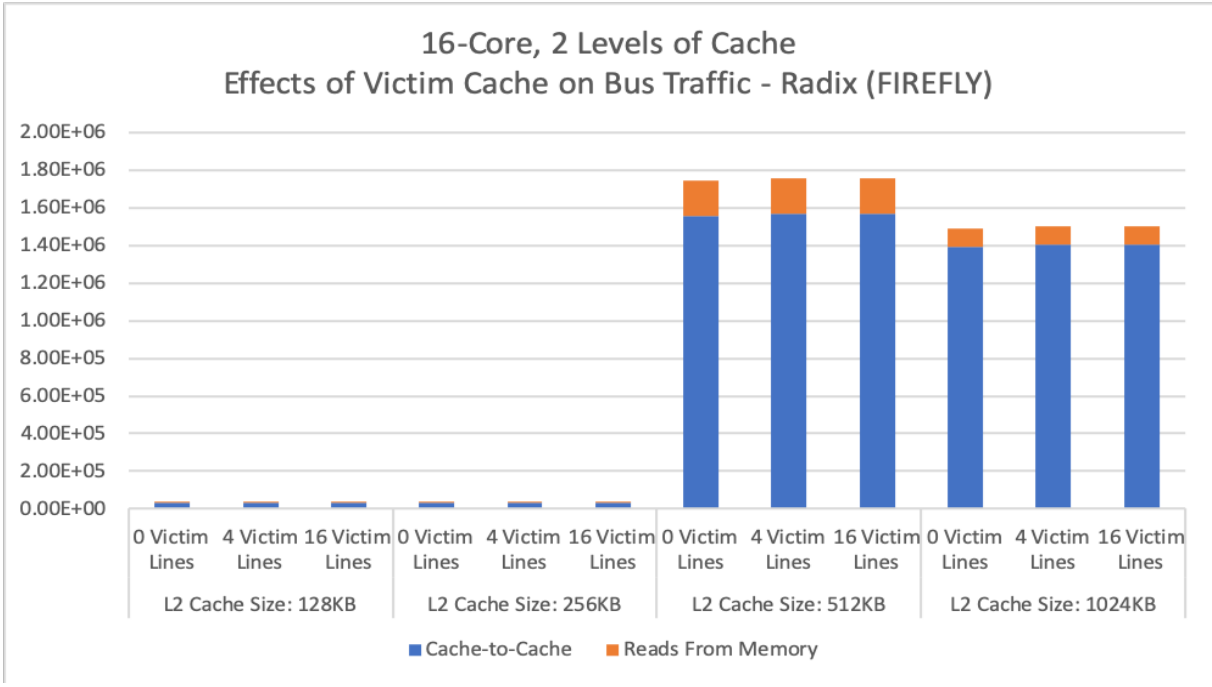

Figure 5.61: Effects of Victim Cache on Bus Traffic - Radix (FIREFLY)

Source: Primary

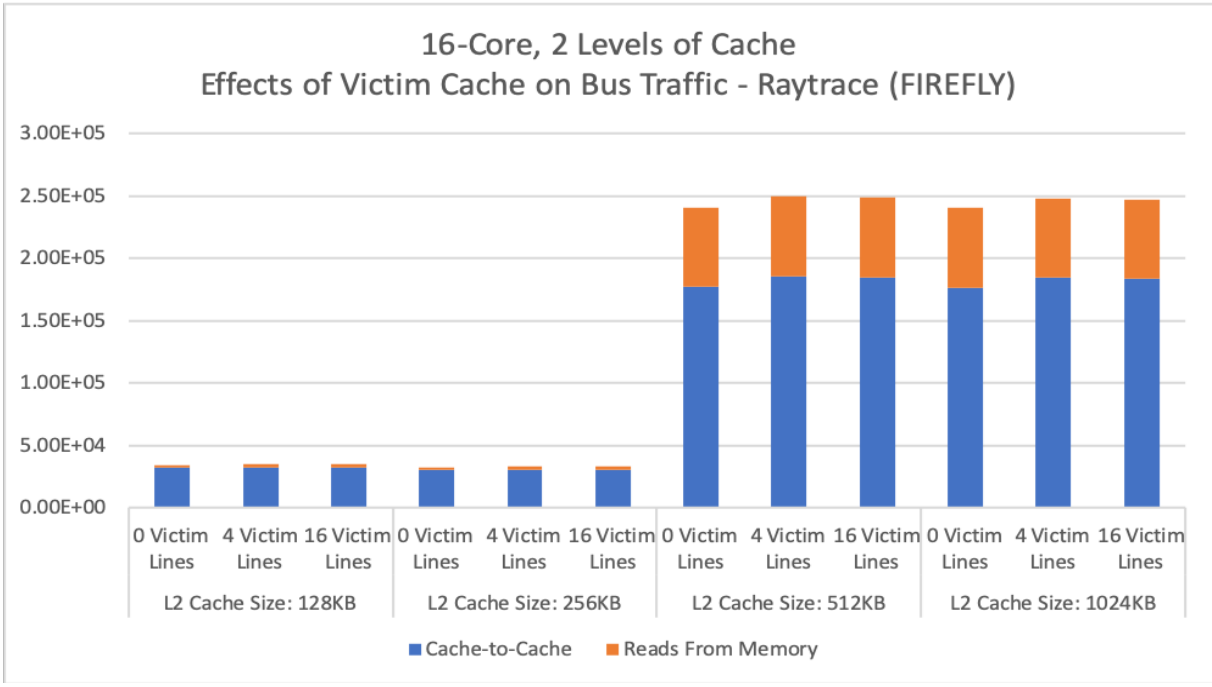

Figure 5.62: Effects of Victim Cache on Bus Traffic - Raytrace (FIREFLY)

Source: Primary

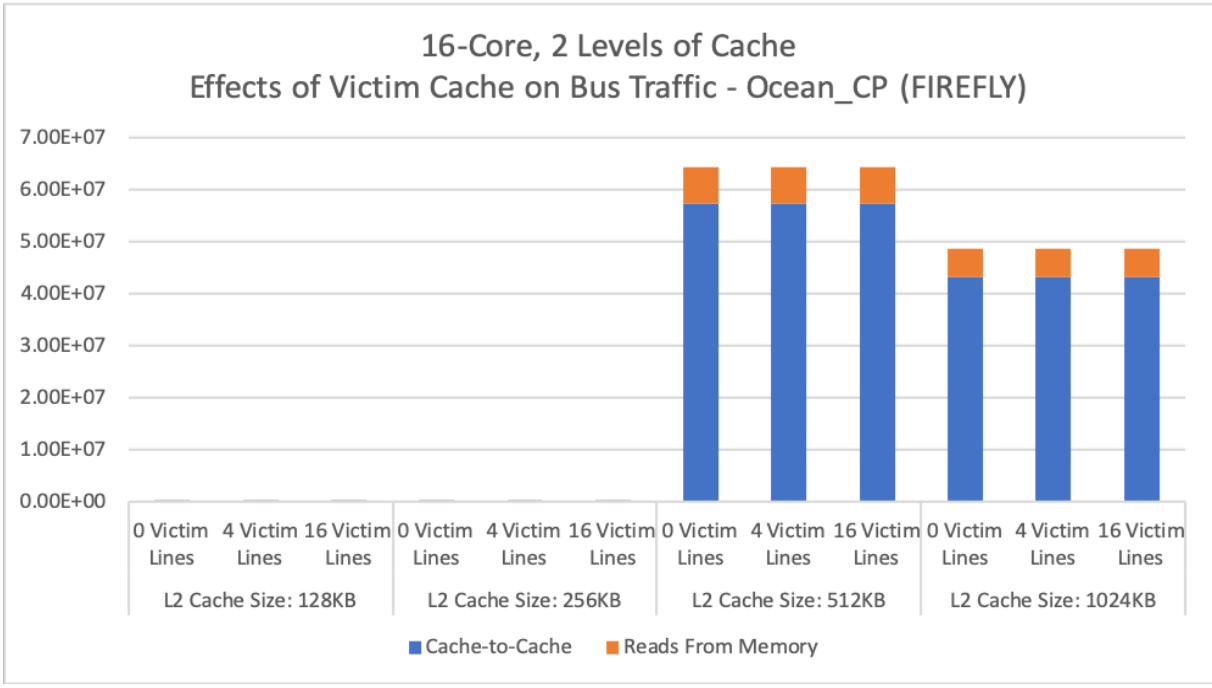

Figure 5.63: Effects of Victim Cache on Bus Traffic - Ocean_CP (FIREFLY)

Source: Primary

Lastly, the core count will be evaluated. The figures below represent two level cache simulations, where the L1 cache size is 4KB with a block size of 32 bytes. Additionally, it utilizes a direct-mapped associativity and the least recently used replacement policy. The L2 cache has a size of 128KB, with a block size of 64 bytes, a 4-way set-associativity, and utilizes the lease recently used replacement policy. The results shown reflect the global miss rate of the given simulation using the MESI coherence protocol.

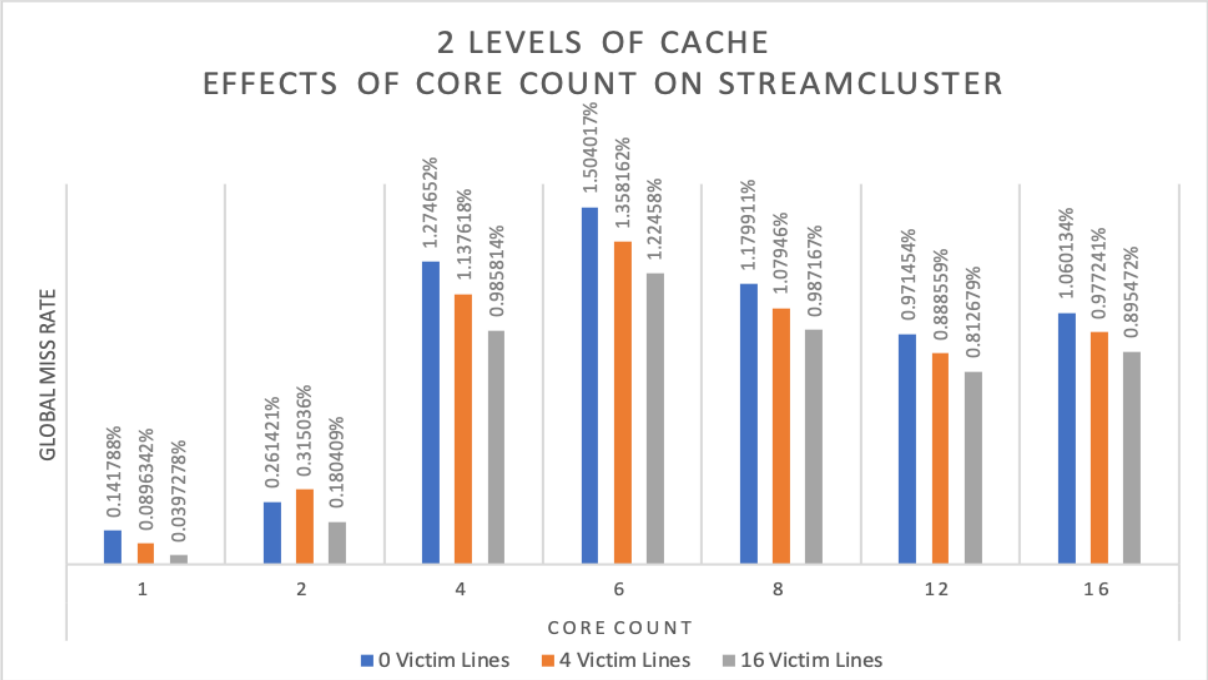

Figure 5.64: 16-Core, 2 Levels of Cache - Effects of Core Count on Streamcluster

Source: Primary

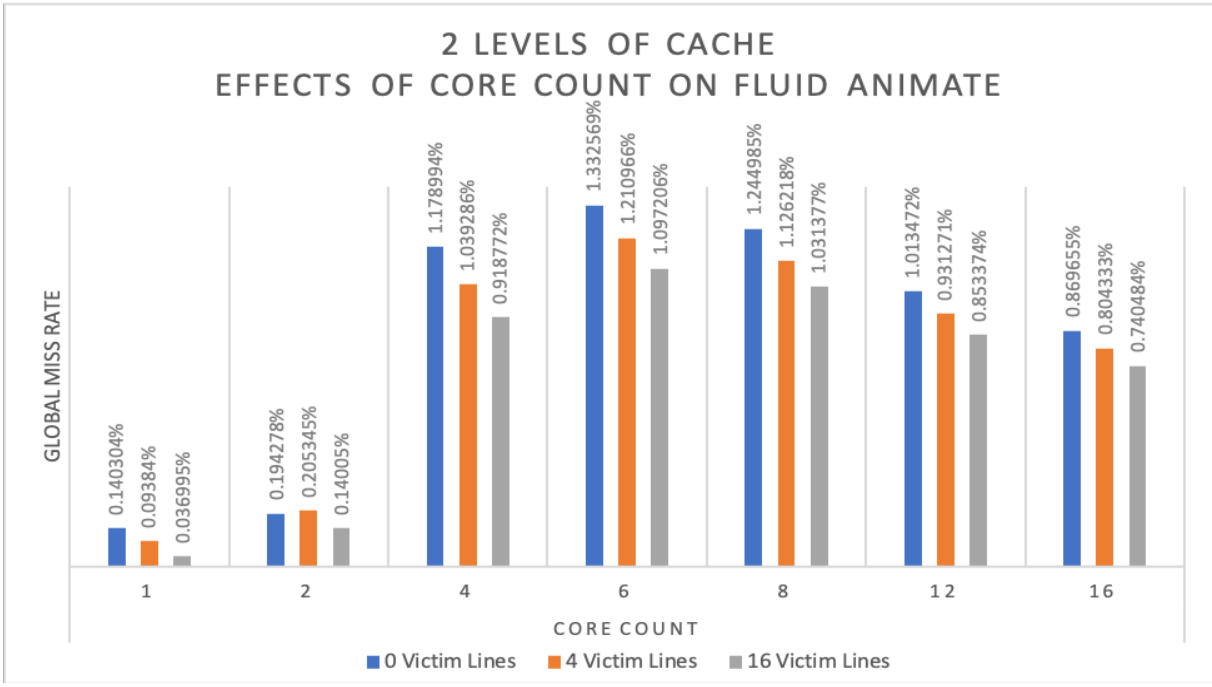

Figure 5.65: 16-Core, 2 Levels of Cache - Effects of Core Count on Fluid Animate

Source: Primary

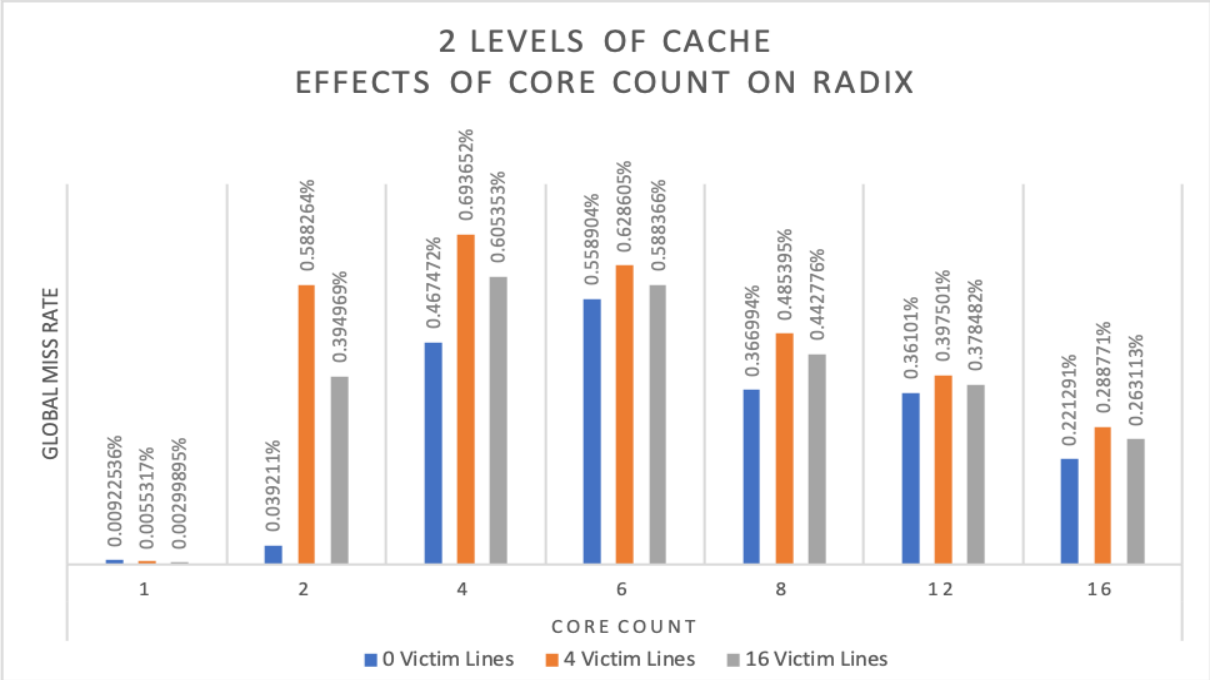

Figure 5.66: 16-Core, 2 Levels of Cache - Effects of Core Count on Radix

Source: Primary

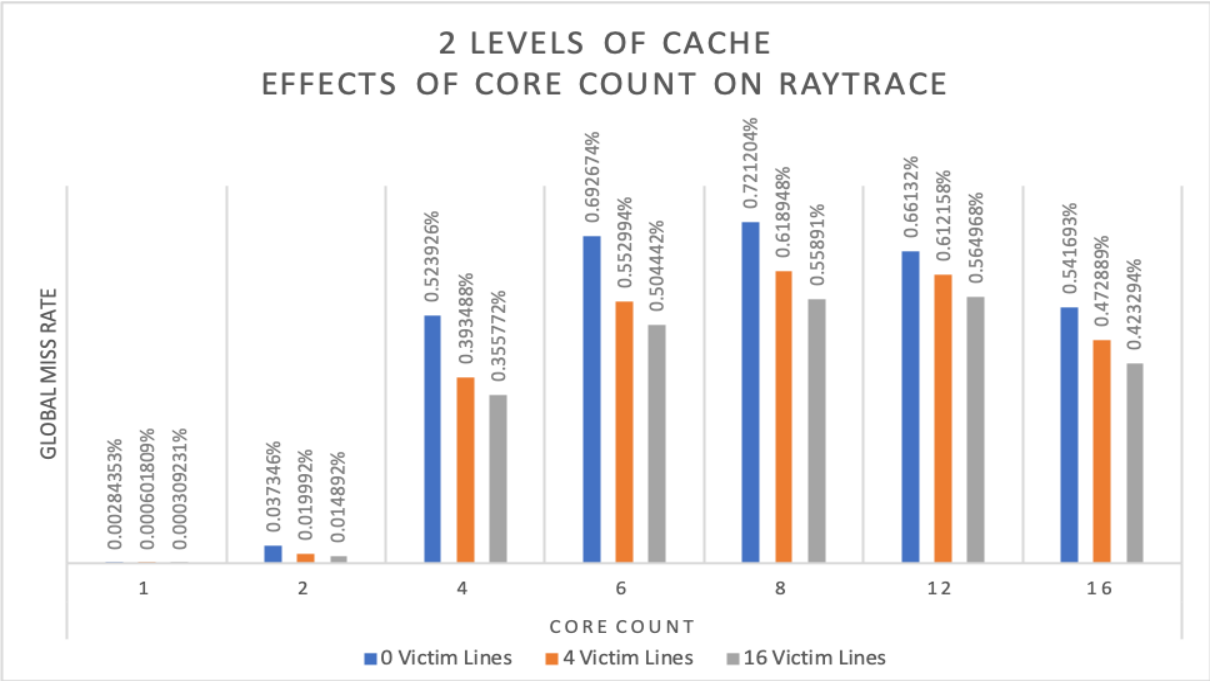

Figure 5.67: 16-Core, 2 Levels of Cache - Effects of Core Count on Raytrace

Source: Primary

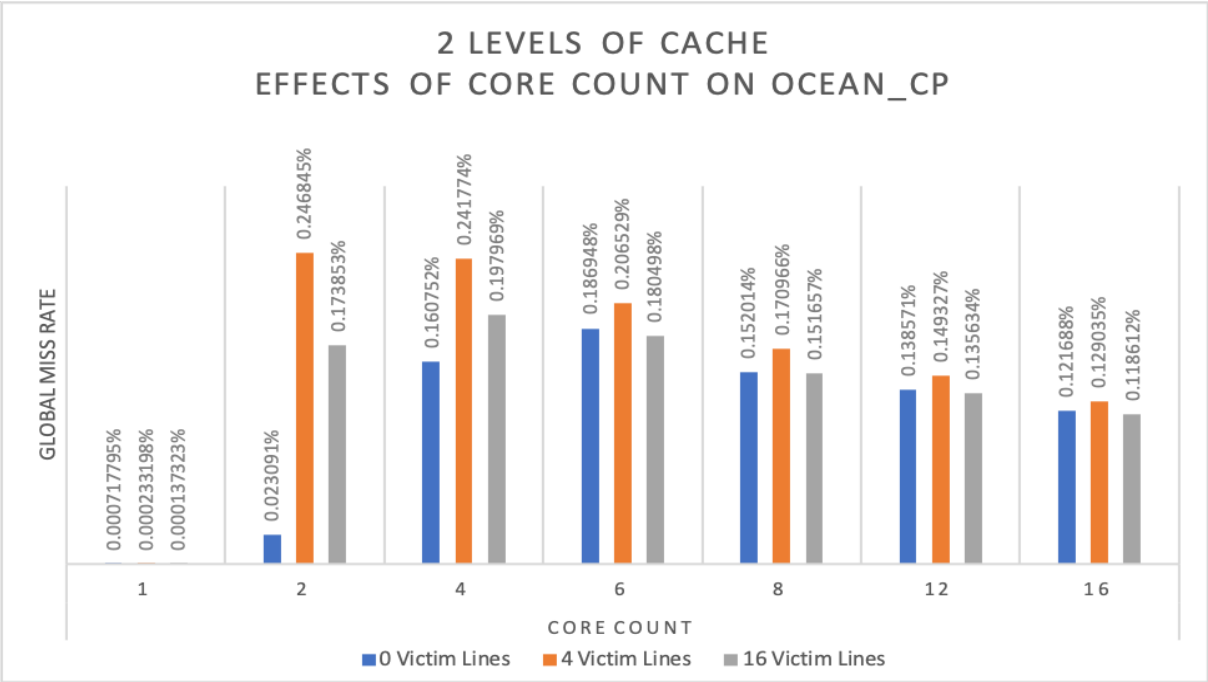

Figure 5.68: 16-Core, 2 Levels of Cache - Effects of Core Count on Ocean_CP

Source: Primary

CHAPTER VI

CONCLUSION

NRC-SIM is designed to be a powerful, yet simplistic, cache simulation tool - powerful in its capabilities as a multi-level, many-core cache simulator and simplistic enough to be a practical and convenient educational tool. By building upon the work done in [10], NRC-SIM is able to provide users with the ability to design and simulate cache memory configurations of their own design via an intuitive user interface. To test its validity, several benchmark programs were tested, ranging from animation to high-performance computing. The results gathered from these tests indicate that the functionality of NRC-SIM, both the Node-RED interface and the additions made to the simulation parameters, is true and accurate.

6.1 Future Work

NRC-SIM added several features to a well grounded foundation. Likewise, more can be done to expand on the work done in this thesis. The inclusion of an L3 or even L4 cache could be possible, allowing for improvements to the simulation method and a greater flexibility in its capabilities. The same can be said for adding additional replacement policies and coherence protocols. Furthermore, work can also be done in regards to measuring power dissipation, energy consumption, and the thermal effects at the micro-architectural level based on the work done in CACTI [32], Wattch [33], and Power Analyzer for Pocket Computing (PAPC) [34] to better understand the performance trade-offs inherent in different cache sizes and organizations. Additionally, from the Node-RED perspective, it would be interesting to see if real-time graphing could be done of the simulated results, offering a visual representation of each core's performance.

REFERENCES

[1] Intel, "The story of the intel® 4004." https://www.intel.com/content/www/us/en/history/museum-story-of-intel-4004.html.

[2] G. V. Research, "Microprocessor market size, share & trend analysis report by architecture type (arm mpu, x64, x86, mips), by appilcation, by region, and segment forecasts, 2023 -2030." https://www.grandviewresearch.com/industry-analysis/microprocessor-market, May 2020.

[3] S. B. *et al*., "The power4 system," in *The POWER4 Processor Introduction and Tuning Guide*, ch. 2, pp. 5–23, IBM Redbooks, 2001.

[4] M. V. Wilkes, "Slave memories and dynamic storage allocation," *IEEE Transactions on Electronic Computers*, vol. EC-14, no. 2, pp. 270–271, 1965.

[5] H. Brais, R. Kalayappan, and P. R. Panda, "A survey of cache simulators," *ACM Comput. Surv.*, vol. 53, feb 2020.

[6] T. Austin, E. Larson, and D. Ernst, "Simplescalar: an infrastructure for computer system modeling," *Computer*, vol. 35, no. 2, pp. 59–67, 2002.

[7] D. M. Tullsen, "Fellowship - simulation and modeling of a simultaneous multithreading processor," in *Int. CMG Conference*, 1996.

[8] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli, "Multi2sim: A simulation framework for cpu-gpu computing," in *2012 21st International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 335–344, 2012.

[9] openJS Foundation and Contributors, "Node-red." https://nodered.org.

[10] P. Roy, "Simncore: Multilevel cache memory design simulator for manycore system," 2022.

[11] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," in *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '05, (New York, NY, USA), p. 190–200, Association for Computing Machinery, 2005.

[12] A. Akram and L. Sawalha, "A survey of computer architecture simulation techniques and tools," *IEEE Access*, vol. 7, pp. 78120–78145, 2019.

[13] J. Edler and M. D. Hill, "Dinero iv trace-driven uniprocessor cache simulator." https://pages.cs.wisc.edu/~markhill/DineroIV/, 1998.

[14] R. Iyer, "On modeling and analyzing cache hierarchies using casper," in *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003.*, pp. 182–187, 2003.

[15] N. Binkert, B. Beckmann, G. Black, S. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib Bin Altaf, N. Vaish, M. Hill, and D. Wood, "The gem5 simulator," *SIGARCH Computer Architecture News*, vol. 39, pp. 1–7, 08 2011.

[16] C. Bienia, *Benchmarking Modern Multiprocessors*. PhD thesis, USA, 2011. AAI3445564.

[17] T. L. Foundation, "Openjs foundation." https://openjsf.org/, 2019.

[18] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta, "The splash-2 programs: characterization and methodological considerations," in *Proceedings 22nd Annual International Symposium on Computer Architecture*, pp. 24–36, 1995.

[19] N. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," 1990.

[20] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Amsterdam: Morgan Kaufmann, 5 ed., 2012.

[21] J. P. Shen and M. H. Lipasti, "Memory and i/o systems," in *Modern Processor Design: Fundamentals of SuperScalar Processors*, ch. 3, pp. 110–136, Waveland Press, Inc., 2013.

[22] R. F. Olanrewaju, A. Baba, B. U. I. Khan, M. Yaacob, A. W. Azman, and M. S. Mir, "A study on performance evaluation of conventional cache replacement algorithms: A review," in *2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pp. 550–556, 2016.

[23] H. Al-Zoubi, A. Milenkovic, and M. Milenkovic, "Performance evaluation of cache replacement policies for the spec cpu2000 benchmark suite," in *Proceedings of the 42nd Annual Southeast Regional Conference*, ACM-SE 42, (New York, NY, USA), p. 267–272, Association for Computing Machinery, 2004.

[24] P. Conway, N. Kalyanasundharam, G. Donley, K. Lepak, and B. Hughes, "Cache hierarchy and memory subsystem of the amd opteron processor," *IEEE Micro*, vol. 30, no. 2, pp. 16–29, 2010.

[25] P. Hammarlund, A. J. Martinez, A. A. Bajwa, D. L. Hill, E. Hallnor, H. Jiang, M. Dixon, M. Derr, M. Hunsaker, R. Kumar, R. B. Osborne, R. Rajwar, R. Singhal, R. D'Sa, R. Chappell, S. Kaushik, S. Chennupaty, S. Jourdan, S. Gunther, T. Piazza, and T. Burton, "Haswell: The fourth-generation intel core processor," *IEEE Micro*, vol. 34, no. 2, pp. 6–20, 2014.

[26] W. Stallings, "Appendix d victim cache strategies," 2012.

[27] TutorialsPoint, "Cache coherence and synchronization." https://www.tutorialspoint.com/parallel_computer_architecture/parallel_computer_architecture_cache_coherence_synchronization.html.

[28] N. P. Jouppi, "Cache write policies and performance," in *Proceedings of the 20th Annual International Symposium on Computer Architecture*, ISCA '93, (New York, NY, USA), p. 191–201, Association for Computing Machinery, 1993.

[29] B. Hashemi, "Simulation and evaluation snoopy cache coherence protocols with update strategy in shared memory multiprocessor systems," in *2011 IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications Workshops*, pp. 256–259, 2011.

[30] N. B. Mallya, G. Patil, and B. Raveendran, "Simulation based performance study of cache coherence protocols," in *2015 IEEE International Symposium on Nanoelectronic and Information Systems*, pp. 125–130, 2015.

[31] Microsoft, "Azure." https://azure.microsoft.com/en-us, Feburary 1, 2010.

[32] P. Shivakumar and N. Jouppi, "Cacti 3.0: An integrated cache timing, power, and area model," 01 2001.

[33] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *Proceedings of 27th International Symposium on Computer Architecture (IEEE Cat. No.RS00201)*, pp. 83–94, 2000.

[34] T. N. Mudge, N. S. Kim, J. Ringenberg, and T. Kgil, "Power analyzer for pocket computing (papc)," 2004.

APPENDIX A

APPENDIX A

GENERATING TRACE FILES

To generate trace files for use with NRC-SIM, use of benchmark programs, such as those found in the PARSEC benchmark suite, and Intel's Pin is required. Pin is a dynamic binary instrumentation framework that enables program analysis. It collects run-time information by inserting code dynamically while the executable is running and is used to generate a trace file that will then be fed into NRC-SIM as input. Traces of all memory addresses referenced by a program are represented, as well as each respective core and instruction type. The PARSEC benchmark suite is composed of multi-threaded programs and offers a wide selection of programs that try to be as representative as possible of real-world scenarios, focusing on programs from all domains, such as desktop and server applications. The current version of the PARSEC benchmark suite offers 13 programs, ranging from computer vision to animation physics and image processing. Additionally, programs from the SPLASH-2 and SPLASH-2X benchmark suites are included as well. Below is a detailed overview of how to generate trace files for NRC-SIM.

1. Download Pin and PARSEC

   (a) Pin download link: `https://www.intel.com/content/www/us/en/developer/articles/tool/pin-a-binary-instrumentation-tool-downloads.html`

   (b) PARSEC download link: `https://parsec.cs.princeton.edu/download.htm`

2. Build Benchmark Programs

   For this example, the FFT and Radix benchmark programs will be demonstrated.

   (a) Unpack the downloaded PARSEC 3.0 package

i. Go to the directory containing the download and enter the following command:

tar -xzf parsec-3.0.tar.gz

(b) Go to the PARSEC-3.0 folder and setup the environment variable

i. Enter the following commands:

cd parsec-3.0

source env.sh

(c) Build and run the benchmarks

i. To build the benchmarks, enter the following commands:

parsecmgmt –a build -p splash2.fft

parsecmgmt –a build -p splash2.radix

ii. To run the benchmarks, enter the following commands:

parsecmgmt –a run -p splash2.fft

parsecmgmt –a run -p splash2.radix

The following figures show the resulting output upon building and running the FFT and Radix bencmarks.

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/parsec-3.0
ubuntu@ubuntu-VirtualBox:~/Desktop/parsec-3.0$ parsecmgmt -a build -p splash2.fft
[PARSEC] Packages to build:  splash2.fft

[PARSEC] [========== Building package splash2.fft [1] ==========]
[PARSEC] [---------- Analyzing package splash2.fft ----------]
[PARSEC] Package splash2.fft already exists, proceeding.
[PARSEC]
[PARSEC] BIBLIOGRAPHY
[PARSEC]
[PARSEC] [1] Woo et al. The SPLASH-2 Programs: Characterization and Methodological
 Considerations. ISCA, 1995.
[PARSEC]
[PARSEC] Done.
ubuntu@ubuntu-VirtualBox:~/Desktop/parsec-3.0$
```

Figure A.1: FFT Build Confirmation

Source: Primary

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/parsec-3.0
ubuntu@ubuntu-VirtualBox:~/Desktop/parsec-3.0$ parsecmgmt -a build -p splash2.radix
[PARSEC] Packages to build:  splash2.radix

[PARSEC] [========== Building package splash2.radix [1] ==========]
[PARSEC] [---------- Analyzing package splash2.radix ----------]
[PARSEC] Package splash2.radix already exists, proceeding.
[PARSEC]
[PARSEC] BIBLIOGRAPHY
[PARSEC]
[PARSEC] [1] Woo et al. The SPLASH-2 Programs: Characterization and Methodological
Considerations. ISCA, 1995.
[PARSEC]
[PARSEC] Done.
ubuntu@ubuntu-VirtualBox:~/Desktop/parsec-3.0$
```

Figure A.2: Radix Build Confirmation

Source: Primary

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/parsec-3.0
ubuntu@ubuntu-VirtualBox:~/Desktop/parsec-3.0$ parsecmgmt -a run -p splash2.fft
[PARSEC] Benchmarks to run:  splash2.fft

[PARSEC] [========== Running benchmark splash2.fft [1] ==========]
[PARSEC] NOTE: SPLASH-2 only supports "test" input sets.
[PARSEC]       Please use SPLASH-2x for multiple input sets.
[PARSEC] NOTE: SPLASH-2 only supports "test" input sets.
[PARSEC]       Please use SPLASH-2x for multiple input sets.
[PARSEC] Deleting old run directory.
[PARSEC] Setting up run directory.
[PARSEC] No archive for input 'test' available, skipping input setup.
[PARSEC] Running 'time /home/ubuntu/Desktop/parsec-3.0/ext/splash2/kernels/fft/inst
/amd64-linux.gcc/bin/run.sh 1 test':
[PARSEC] [---------- Beginning of output ----------]
Generating input file ...
Running /home/ubuntu/Desktop/parsec-3.0/ext/splash2/kernels/fft/inst/amd64-linux.gc
c/bin/fft -m16 -p1:

FFT with Blocking Transpose
   65536 Complex Doubles
   1 Processors
   65536 Cache lines
   16 Byte line size
```

Figure A.3: FFT Run Confirmation (1/2)

Source: Primary

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/parsec-3.0

                  TIMING INFORMATION
Start time                        : 1681916572813896
Initialization finish time        : 1681916572818272
Overall finish time               : 1681916572834192
Total time with initialization    :            20296
Total time without initialization :            15920
Overall transpose time            :             1504
Overall transpose fraction        :          0.09447


real    0m0.058s
user    0m0.014s
sys     0m0.000s
[PARSEC] [----------    End of output    ----------]
[PARSEC]
[PARSEC] BIBLIOGRAPHY
[PARSEC]
[PARSEC] [1] Woo et al. The SPLASH-2 Programs: Characterization and Methodological
Considerations. ISCA, 1995.
[PARSEC]
[PARSEC] Done.
ubuntu@ubuntu-VirtualBox:~/Desktop/parsec-3.0$
```

Figure A.4: FFT Run Confirmation (2/2)

Source: Primary

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/parsec-3.0
ubuntu@ubuntu-VirtualBox:~/Desktop/parsec-3.0$ parsecmgmt -a run -p splash2.radix
[PARSEC] Benchmarks to run:  splash2.radix

[PARSEC] [========== Running benchmark splash2.radix [1] ==========]
[PARSEC] NOTE: SPLASH-2 only supports "test" input sets.
[PARSEC]       Please use SPLASH-2x for multiple input sets.
[PARSEC] NOTE: SPLASH-2 only supports "test" input sets.
[PARSEC]       Please use SPLASH-2x for multiple input sets.
[PARSEC] Deleting old run directory.
[PARSEC] Setting up run directory.
[PARSEC] No archive for input 'test' available, skipping input setup.
[PARSEC] Running 'time /home/ubuntu/Desktop/parsec-3.0/ext/splash2/kernels/radix/in
st/amd64-linux.gcc/bin/run.sh 1 test':
[PARSEC] [---------- Beginning of output ----------]
Generating input file ...
Running /home/ubuntu/Desktop/parsec-3.0/ext/splash2/kernels/radix/inst/amd64-linux.
gcc/bin/radix -p1 -r1024 -n262144 -m524288:

Integer Radix Sort
     262144 Keys
     1 Processors
     Radix = 1024
     Max key = 524288
```

Figure A.5: Radix Run Confirmation (1/2)

Source: Primary

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/parsec-3.0
 Proc          Time             Time            Time
    0           2762             602            2157

                 TIMING INFORMATION
Start time                        : 1681916669064931
Initialization finish time        : 1681916669143501
Overall finish time               : 1681916669146263
Total time with initialization    :            81332
Total time without initialization :             2762


real    0m0.098s
user    0m0.030s
sys     0m0.007s
[PARSEC] [----------    End of output    ----------]
[PARSEC]
[PARSEC] BIBLIOGRAPHY
[PARSEC]
[PARSEC] [1] Woo et al. The SPLASH-2 Programs: Characterization and Methodological
Considerations. ISCA, 1995.
[PARSEC]
[PARSEC] Done.
ubuntu@ubuntu-VirtualBox:~/Desktop/parsec-3.0$
```

Figure A.6: Radix Run Confirmation (2/2)

Source: Primary

3. Place Benchmark Programs into the ManualExamples Folder in Pin

   (a) For example, the benchmark programs will be located in the following:

      i. FFT

      /parsec-3.0/ext/splash2/kernels/fft/obj/amd64-linux.gcc/fft

      i. Radix

      /parsec-3.0/ext/splash2/kernels/radix/obj/amd64-linux.gcc/radix

   These programs will then be placed in the following:

   /pin-3.26/source/tools/ManualExamples

   The original locations of the FFT and Radix executables are pictured below, along with their new location in pin.

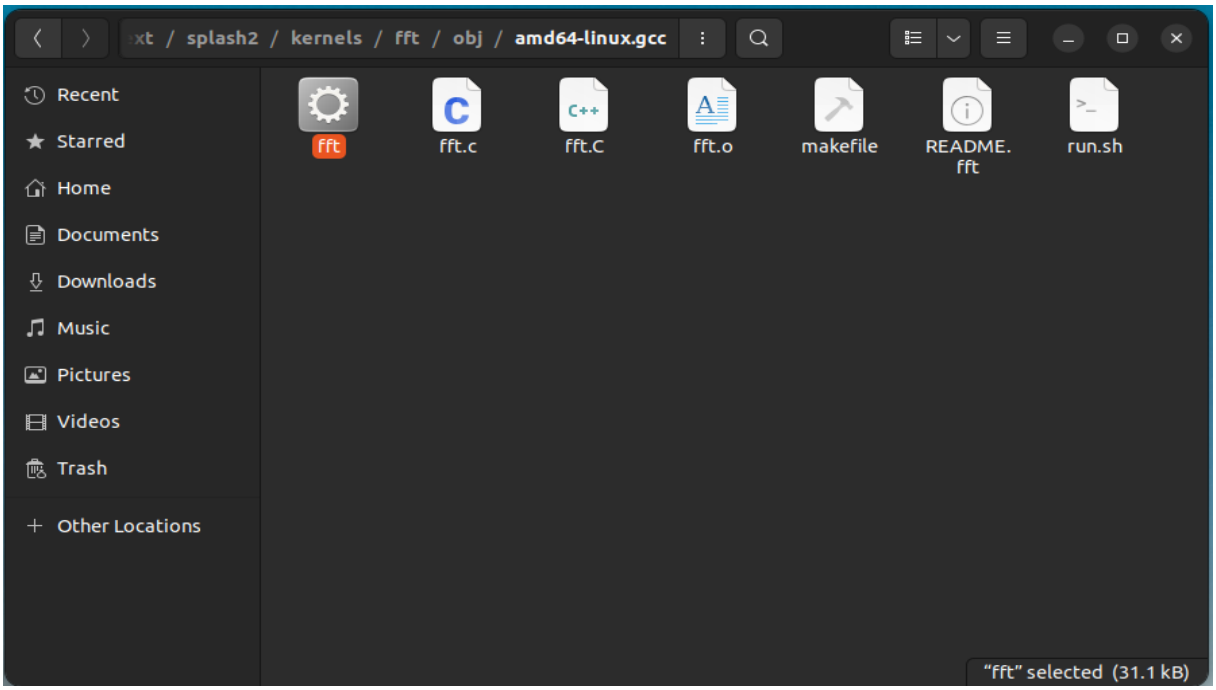

Figure A.7: Original FFT Executable Location

Source: Primary

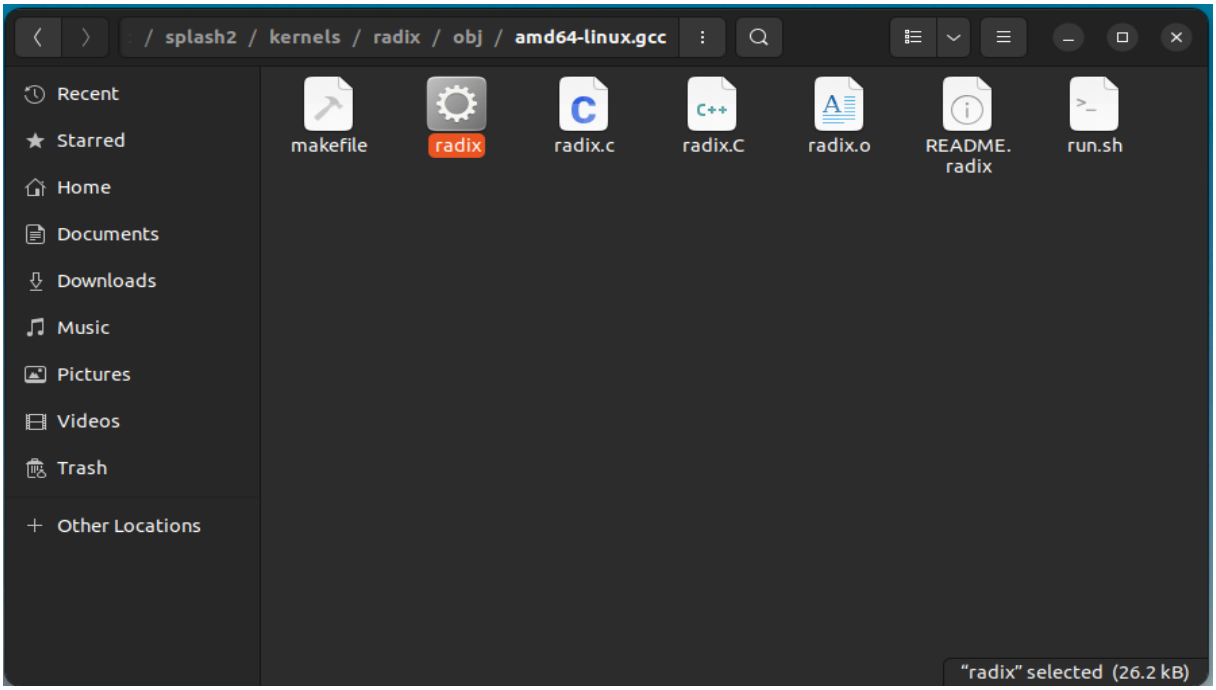

Figure A.8: Original Radix Executable Location

Source: Primary

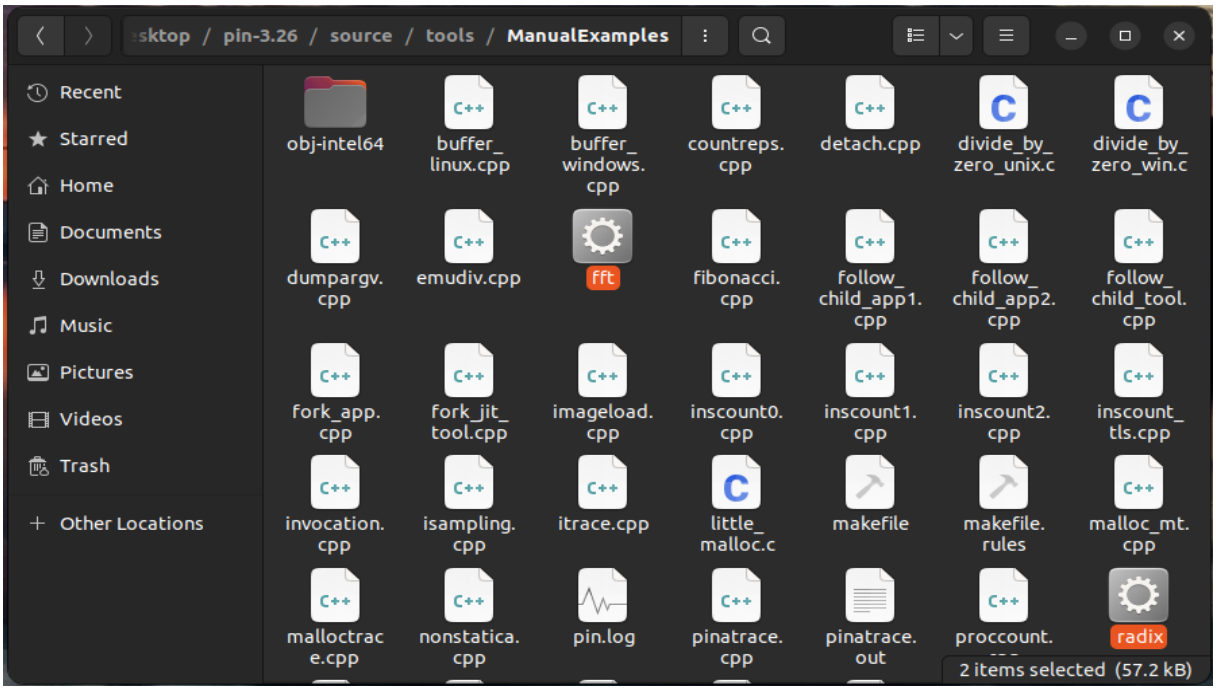

Figure A.9: Executables in Pin's ManualExamples Folder

Source: Primary

4. Modify pinatrace.cpp

   (a) Modifications must be made to pinatrace.cpp to ensure that the trace files are in the proper format for the simulator. The trace file format should be as follows:

<Core ID> <Instruction Type> <Memory Address>

Open

*pinatrace.cpp
~/Desktop/pin-3.26/source/tools/ManualExamples

```
/*
 * Copyright (C) 2004-2021 Intel Corporation.
 * SPDX-License-Identifier: MIT
 */

/*
 *  This file contains an ISA-portable PIN tool for tracing memory accesses.
 */

#include <stdio.h>
#include "pin.H"

FILE* trace;

int n; //core-count

// Print a memory read record
VOID RecordMemRead(int num, VOID* addr)
{
        num = (rand()%((n-1)-0+1));
        fprintf(trace, "%d r %p\n", num, addr);
}

// Print a memory write record
VOID RecordMemWrite(int num, VOID* addr)
{
        num = (rand()%((n-1)-0+1));
        fprintf(trace, "%d w %p\n", num, addr);
}
```

Figure A.10: Modifications to pinatrace.cpp

Source: Primary

In the figure above, the core-count, *n*, should be set to the desired core-count for the trace files, i.e. 1 for single-core, 2 for dual-core, etc.

5. Compile pinatrace.cpp

   (a) To compile pinatrace.cpp, use the following command:

make pinatrace.test

This will have to be done each time pinatrace.cpp is modified, i.e. each time the core count is changed. The figure below shows the resulting output.

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/pin-3.26/source/tools/Manu...
ubuntu@ubuntu-VirtualBox:~/Desktop/pin-3.26/source/tools/ManualExamples$ make pinatrac
e.test
g++ -Wall -Werror -Wno-unknown-pragmas -DPIN_CRT=1 -fno-stack-protector -fno-exception
s -funwind-tables -fasynchronous-unwind-tables -fno-rtti -DTARGET_IA32E -DHOST_IA32E -
fPIC -DTARGET_LINUX -fabi-version=2 -faligned-new -I../../../source/include/pin -I../.
./../source/include/pin/gen -isystem /home/ubuntu/Desktop/pin-3.26/extras/cxx/include
-isystem /home/ubuntu/Desktop/pin-3.26/extras/crt/include -isystem /home/ubuntu/Deskto
p/pin-3.26/extras/crt/include/arch-x86_64 -isystem /home/ubuntu/Desktop/pin-3.26/extra
s/crt/include/kernel/uapi -isystem /home/ubuntu/Desktop/pin-3.26/extras/crt/include/ke
rnel/uapi/asm-x86 -I../../../extras/components/include -I../../../extras/xed-intel64/i
nclude/xed -I../../../source/tools/Utils -I../../../source/tools/InstLib -O3 -fomit-fr
ame-pointer -fno-strict-aliasing  -Wno-dangling-pointer -c -o obj-intel64/pinatrace.o
pinatrace.cpp
g++ -shared -Wl,--hash-style=sysv ../../../intel64/runtime/pincrt/crtbeginS.o -Wl,-Bsy
mbolic -Wl,--version-script=../../../source/include/pin/pintool.ver -fabi-version=2
-o obj-intel64/pinatrace.so obj-intel64/pinatrace.o -L../../../intel64/runtime/pincrt
-L../../../intel64/lib -L../../../intel64/lib-ext -L../../../extras/xed-intel64/lib -l
pin -lxed ../../../intel64/runtime/pincrt/crtendS.o -lpindwarf -ldl-dynamic -nostdlib
-lc++ -lc++abi -lm-dynamic -lc-dynamic -lunwind-dynamic
../../../pin   -t obj-intel64/pinatrace.so -- ../../../source/tools/Utils/obj-intel64/
cp-pin.exe makefile obj-intel64/pinatrace.makefile.copy \
  > obj-intel64/pinatrace.out 2>&1
cmp makefile obj-intel64/pinatrace.makefile.copy
rm obj-intel64/pinatrace.makefile.copy
rm obj-intel64/pinatrace.out
ubuntu@ubuntu-VirtualBox:~/Desktop/pin-3.26/source/tools/ManualExamples$
```

Figure A.11: Resulting Output of pinatrace.cpp Compilation

Source: Primary

6. Generate the Trace Files via Pin Instrumentation

   (a) To generate the trace files, go to the ManualExamples folder within Pin and enter a command in the following format:

   pin –t obj-intel64/pinatrace.so – [desired benchmark program]

   The following figure illustrates the command format for generating trace files using benchmark parameters and .env files.

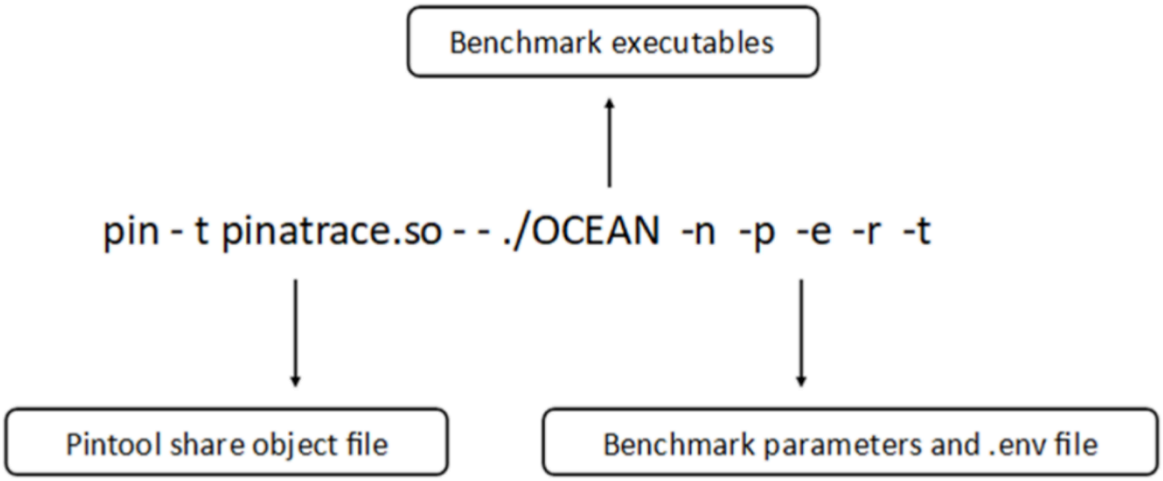

Figure A.12: Pin Instrumentation With Associated Parameters and Environment

Source: Primary

The following figures show the resulting output of generating single-core and 8-core trace files for both the FFT and Radix benchmarks.

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/pin-3.26/source/tools/Manu...
ubuntu@ubuntu-VirtualBox:~/Desktop/pin-3.26/source/tools/ManualExamples$ ../../../pin
-t obj-intel64/pinatrace.so -- ./fft -p1

FFT with Blocking Transpose
   1024 Complex Doubles
   1 Processors
   65536 Cache lines
   16 Byte line size
   4096 Bytes per page


                PROCESS STATISTICS
            Computation      Transpose     Transpose
 Proc          Time            Time        Fraction
    0          922670          153397       0.16625

                TIMING INFORMATION
Start time                          : 1681923451923031
Initialization finish time          : 1681923453742463
Overall finish time                 : 1681923454665133
Total time with initialization      :          2742102
Total time without initialization   :           922670
Overall transpose time              :           153397
Overall transpose fraction          :          0.16625

ubuntu@ubuntu-VirtualBox:~/Desktop/pin-3.26/source/tools/ManualExamples$
```

Figure A.13: Resulting Output After Generating Single-Core FFT Trace File

Source: Primary

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/pin-3.26/source/tools/Manu...
ubuntu@ubuntu-VirtualBox:~/Desktop/pin-3.26/source/tools/ManualExamples$ ../../../pin
-t obj-intel64/pinatrace.so -- ./fft -p8

FFT with Blocking Transpose
   1024 Complex Doubles
   8 Processors
   65536 Cache lines
   16 Byte line size
   4096 Bytes per page


                PROCESS STATISTICS
            Computation      Transpose     Transpose
 Proc          Time            Time        Fraction
    0          606900          17454       0.02876

                TIMING INFORMATION
Start time                          : 1681922888133592
Initialization finish time          : 1681922889226965
Overall finish time                 : 1681922889833865
Total time with initialization      :          1700273
Total time without initialization   :           606900
Overall transpose time              :            17454
Overall transpose fraction          :          0.02876

ubuntu@ubuntu-VirtualBox:~/Desktop/pin-3.26/source/tools/ManualExamples$
```

Figure A.14: Resulting Output After Generating 8-Core FFT Trace File

Source: Primary

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/pin-3.26/source/tools/Manu...
ubuntu@ubuntu-VirtualBox:~/Desktop/pin-3.26/source/tools/ManualExamples$ ../../../pin
-t obj-intel64/pinatrace.so -- ./radix -p1

Integer Radix Sort
     262144 Keys
     1 Processors
     Radix = 1024
     Max key = 524288


                PROCESS STATISTICS
              Total            Rank            Sort
 Proc         Time             Time            Time
    0       25373720         9666603        15693976

                TIMING INFORMATION
Start time                          : 1681923583305772
Initialization finish time          : 1681923626576336
Overall finish time                 : 1681923651950056
Total time with initialization      :         68644284
Total time without initialization   :         25373720

ubuntu@ubuntu-VirtualBox:~/Desktop/pin-3.26/source/tools/ManualExamples$
```

Figure A.15: Resulting Output After Generating Single-Core Radix Trace File

Source: Primary

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/pin-3.26/source/tools/Manu...
ubuntu@ubuntu-VirtualBox:~/Desktop/pin-3.26/source/tools/ManualExamples$ ../../../pin
-t obj-intel64/pinatrace.so -- ./radix -p8

Integer Radix Sort
     262144 Keys
     8 Processors
     Radix = 1024
     Max key = 524288


                PROCESS STATISTICS
               Total            Rank            Sort
 Proc          Time             Time            Time
    0       25846897        11183509        13758775

                TIMING INFORMATION
Start time                        : 1681923189829510
Initialization finish time        : 1681923228534438
Overall finish time               : 1681923254381335
Total time with initialization    :         64551825
Total time without initialization :         25846897

ubuntu@ubuntu-VirtualBox:~/Desktop/pin-3.26/source/tools/ManualExamples$
```

Figure A.16: Resulting Output After Generating 8-Core Radix Trace File

Source: Primary

7. Confirm Proper Formatting of Trace File and Save

   (a) After generating one of the trace files, the result will be stored in the ManualExamples folder under the name “pinatrace.out”. Open this file and confirm the trace file is for the desired core-count and in the desired format (i.e. the format specified in step 4).

   (b) After confirming that the trace file is in the proper format, change the name of the file, i.e. “pinatrace.out”, to represent which trace file it is and set aside for use with NRC-SIM. The following figures show the resulting trace files for the FFT and Radix benchmarks.

```
0 w 0x7ffd856e2d38
0 w 0x7ffd856e2d30
0 w 0x7ffd856e2d28
0 w 0x7ffd856e2d20
0 w 0x7ffd856e2d18
0 w 0x7ffd856e2d10
0 w 0x7ffd856e2d08
0 w 0x7ffd856e2cb8
0 r 0x7f764dac6e0e
0 w 0x7f764dac6e0e
0 w 0x7f764dac6af0
0 w 0x7f764dac5aa0
0 r 0x7f764dac5e80
0 w 0x7f764dac6b00
0 w 0x7f764dac6ba0
0 r 0x7f764dac5e90
0 w 0x7f764dac6b50
0 r 0x7f764dac5ea0
0 w 0x7f764dac6d90
0 r 0x7f764dac5eb0
0 w 0x7f764dac6b58
0 r 0x7f764dac5ec0
0 w 0x7f764dac6b60
0 r 0x7f764dac5ed0
0 w 0x7f764dac6b80
0 r 0x7f764dac5ee0
0 w 0x7f764dac6b88
0 r 0x7f764dac5ef0
0 w 0x7f764dac6b48
0 r 0x7f764dac5f00
0 w 0x7f764dac6b40
0 r 0x7f764dac5f10
0 w 0x7f764dac6bd0
0 r 0x7f764dac5f20
0 w 0x7f764dac6be8
0 r 0x7f764dac5f30
0 w 0x7f764dac6b68
0 r 0x7f764dac5f40
0 w 0x7f764dac6b70
0 r 0x7f764dac5f50
```

Figure A.17: Singe-Core FFT Trace File

Source: Primary

```
7 w 0x7ffef259f3a8
6 w 0x7ffef259f3a0
1 w 0x7ffef259f398
3 w 0x7ffef259f390
1 w 0x7ffef259f388
7 w 0x7ffef259f380
2 w 0x7ffef259f378
4 w 0x7ffef259f328
1 r 0x7f627612de0e
5 w 0x7f627612de0e
2 w 0x7f627612daf0
3 w 0x7f627612caa0
2 r 0x7f627612ce80
3 w 0x7f627612db00
3 w 0x7f627612dba0
6 r 0x7f627612ce90
4 w 0x7f627612db50
2 r 0x7f627612cea0
4 w 0x7f627612dd90
0 r 0x7f627612ceb0
3 w 0x7f627612db58
0 r 0x7f627612cec0
7 w 0x7f627612db60
5 r 0x7f627612ced0
6 w 0x7f627612db80
2 r 0x7f627612cee0
6 w 0x7f627612db88
3 r 0x7f627612cef0
3 w 0x7f627612db48
7 r 0x7f627612cf00
1 w 0x7f627612db40
2 r 0x7f627612cf10
6 w 0x7f627612dbd0
2 r 0x7f627612cf20
5 w 0x7f627612dbe8
7 r 0x7f627612cf30
1 w 0x7f627612db68
0 r 0x7f627612cf40
3 w 0x7f627612db70
2 r 0x7f627612cf50
```

Figure A.18: 8-Core FFT Trace File

Source: Primary

```
radix_1c.out
~/Desktop/pintool_manual/trace_files
0 w 0x7fff0da9bee8
0 w 0x7fff0da9bee0
0 w 0x7fff0da9bed8
0 w 0x7fff0da9bed0
0 w 0x7fff0da9bec8
0 w 0x7fff0da9bec0
0 w 0x7fff0da9beb8
0 w 0x7fff0da9be68
0 r 0x7f12df3dce0e
0 w 0x7f12df3dce0e
0 w 0x7f12df3dcaf0
0 w 0x7f12df3dbaa0
0 r 0x7f12df3dbe80
0 w 0x7f12df3dcb00
0 w 0x7f12df3dcba0
0 r 0x7f12df3dbe90
0 w 0x7f12df3dcb50
0 r 0x7f12df3dbea0
0 w 0x7f12df3dcd90
0 r 0x7f12df3dbeb0
0 w 0x7f12df3dcb58
0 r 0x7f12df3dbec0
0 w 0x7f12df3dcb60
0 r 0x7f12df3dbed0
0 w 0x7f12df3dcb80
0 r 0x7f12df3dbee0
0 w 0x7f12df3dcb88
0 r 0x7f12df3dbef0
0 w 0x7f12df3dcb48
0 r 0x7f12df3dbf00
0 w 0x7f12df3dcb40
0 r 0x7f12df3dbf10
0 w 0x7f12df3dcbd0
0 r 0x7f12df3dbf20
0 w 0x7f12df3dcbe8
0 r 0x7f12df3dbf30
0 w 0x7f12df3dcb68
0 r 0x7f12df3dbf40
0 w 0x7f12df3dcb70
0 r 0x7f12df3dbf50
```

Figure A.19: Singe-Core Radix Trace File

Source: Primary

```
radix_8c.out
~/Desktop/pintool_manual
7 w 0x7ffd6a8951a8
6 w 0x7ffd6a8951a0
1 w 0x7ffd6a895198
3 w 0x7ffd6a895190
1 w 0x7ffd6a895188
7 w 0x7ffd6a895180
2 w 0x7ffd6a895178
4 w 0x7ffd6a895128
1 r 0x7fe7a23efe0e
5 w 0x7fe7a23efe0e
2 w 0x7fe7a23efaf0
3 w 0x7fe7a23eeaa0
2 r 0x7fe7a23eee80
3 w 0x7fe7a23efb00
3 w 0x7fe7a23efba0
6 r 0x7fe7a23eee90
4 w 0x7fe7a23efb50
2 r 0x7fe7a23eeea0
4 w 0x7fe7a23efd90
0 r 0x7fe7a23eeeb0
3 w 0x7fe7a23efb58
0 r 0x7fe7a23eeec0
7 w 0x7fe7a23efb60
5 r 0x7fe7a23eeed0
6 w 0x7fe7a23efb80
2 r 0x7fe7a23eeee0
6 w 0x7fe7a23efb88
3 r 0x7fe7a23eeef0
3 w 0x7fe7a23efb48
7 r 0x7fe7a23eef00
1 w 0x7fe7a23efb40
2 r 0x7fe7a23eef10
6 w 0x7fe7a23efbd0
2 r 0x7fe7a23eef20
5 w 0x7fe7a23efbe8
7 r 0x7fe7a23eef30
1 w 0x7fe7a23efb68
0 r 0x7fe7a23eef40
3 w 0x7fe7a23efb70
2 r 0x7fe7a23eef50
```

Figure A.20: 8-Core Radix Trace File

Source: Primary

APPENDIX B

APPENDIX B

VICTIM CACHE MANUAL

To implement the victim cache design used in NRC-SIM, the following functions were either created or modified in the Functions.h header file:

1. Modified Functions

   (a) **void commandline(int argc, char **argv)**

   Modifications made to this function enable victim cache parameter to be set by the user via the command line.

   (b) **void parameter()**

   Modifications made to this function were necessary to determine the number of bits for the cache's index and tag fields based on the total number of bits in each memory address.

   (c) **void cache_design()**

   Modifications made to this function allow for the dynamic allocation of the memory needed to create the victim cache(s) based on the users desired parameters.

   (d) **void level1_sc()**

   Modifications made to this function were made to search the victim cache in the event of an L1 cache miss.

   (e) **void only_l1_mesi()**

   Modifications made to this function were made to search the victim cache in the event of an L1 cache miss. Additionally, when appropriate, the data in the victim cache is

invalidated.

(f) **void level1_mesi()**

Modifications made to this function were made to search the victim cache in the event of an L1 cache miss. Additionally, when appropriate, the data in the victim cache is invalidated.

(g) **void level2_mesi()**

Modifications made to this function were made to invalidate the data in the victim cache when appropriate.

(h) **void only_l1_msi()**

Modifications made to this function were made to search the victim cache in the event of an L1 cache miss. Additionally, when appropriate, the data in the victim cache is invalidated.

(i) **void level1_msi()**

Modifications made to this function were made to search the victim cache in the event of an L1 cache miss. Additionally, when appropriate, the data in the victim cache is invalidated.

(j) **void level2_msi()**

Modifications made to this function were made to invalidate the data in the victim cache when appropriate.

(k) **void only_l1_firefly()**

Modifications made to this function were made to search the victim cache in the event of an L1 cache miss.

(l) **void level1_firefly()**

Modifications made to this function were made to search the victim cache in the event of an L1 cache miss.

(m) **void log_sc()**

Modifications made to this function were necessary to output all relevant victim cache results, i.e. total victim lines, total accesses, hits, misses, etc.

(n) **void log_mesi_only_l1()**

Modifications made to this function were necessary to output all relevant victim cache results, i.e. total victim lines, total accesses, hits, misses, etc.

(o) **void log_mesi()**

Modifications made to this function were necessary to output all relevant victim cache results, i.e. total victim lines, total accesses, hits, misses, etc.

(p) **void log_firefly_only_l1()**

Modifications made to this function were necessary to output all relevant victim cache results, i.e. total victim lines, total accesses, hits, misses, etc.

(q) **void log_firefly()**

Modifications made to this function were necessary to output all relevant victim cache results, i.e. total victim lines, total accesses, hits, misses, etc.

2. Created Functions

(a) **void vCache()**

Function created to search the victim cache for the requested data. In the event of a victim hit, the data in the victim cache and the L1 cache is swapped and a record of the instruction type hit is incremented. In the event of a victim miss, the L1 data is loaded into the victim cache, with any evicted victim cache data being sent to the next lowest level of memory. Additionally, a record of the instruction type miss is incremented.

In addition to the above, changes were made to the Build.h and Variable.h header files, such as the creation of variables that are used in the previously mentioned functions, instantiating and initializing a structure of type *level1* to create the victim cache, and adding additional member

variables to the previously mentioned structure that are necessary for the implementation of a victim cache in NRC-SIM. A generalized flowchart of NRC-SIM is shown below, showcasing the role of the victim cache.

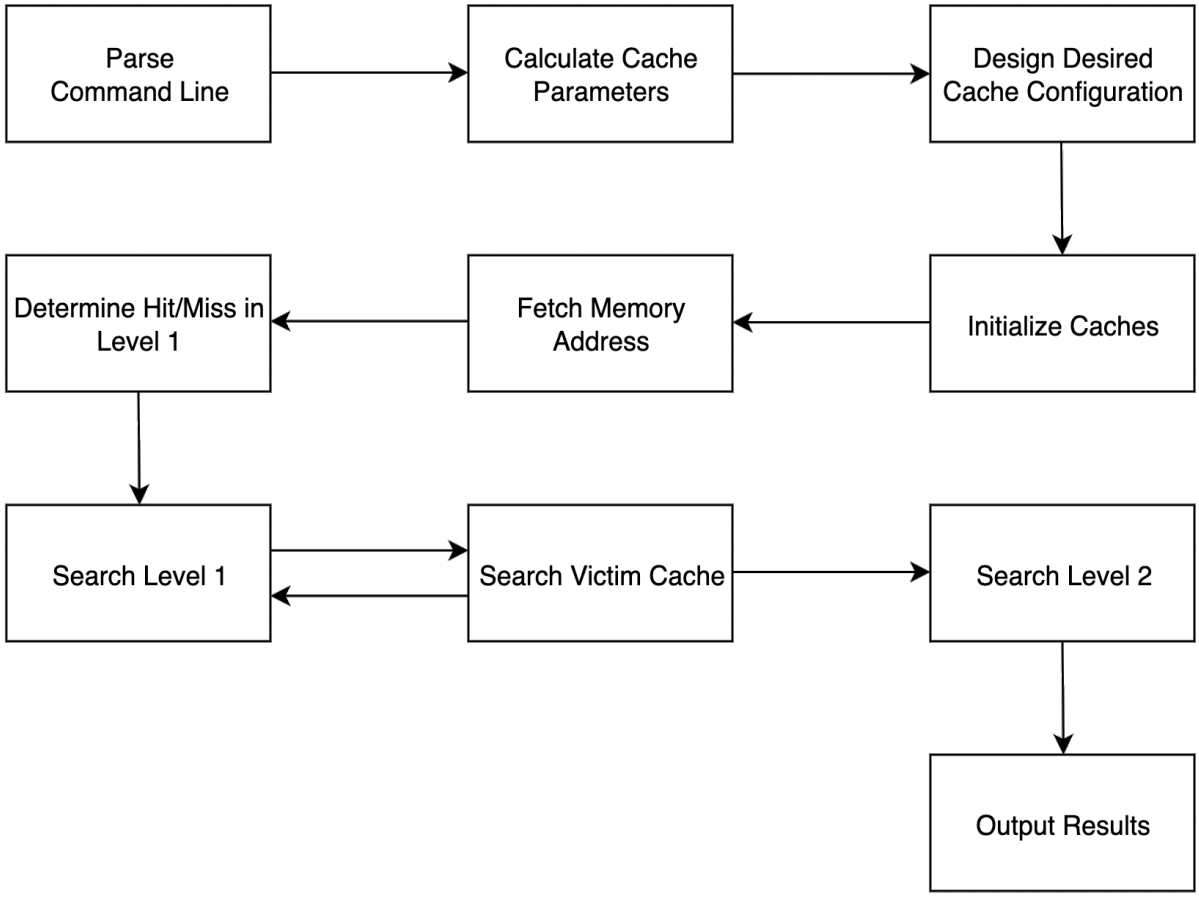

Figure B.1: Generalized Flowchart of NRC-SIM

Source: Primary

The following flowchart shows a more detailed view of NRC-SIM, illustrating NRC-SIM's behavior for single-core and multi-core/many-core simulations, with each state representing a function within the C++ source code.

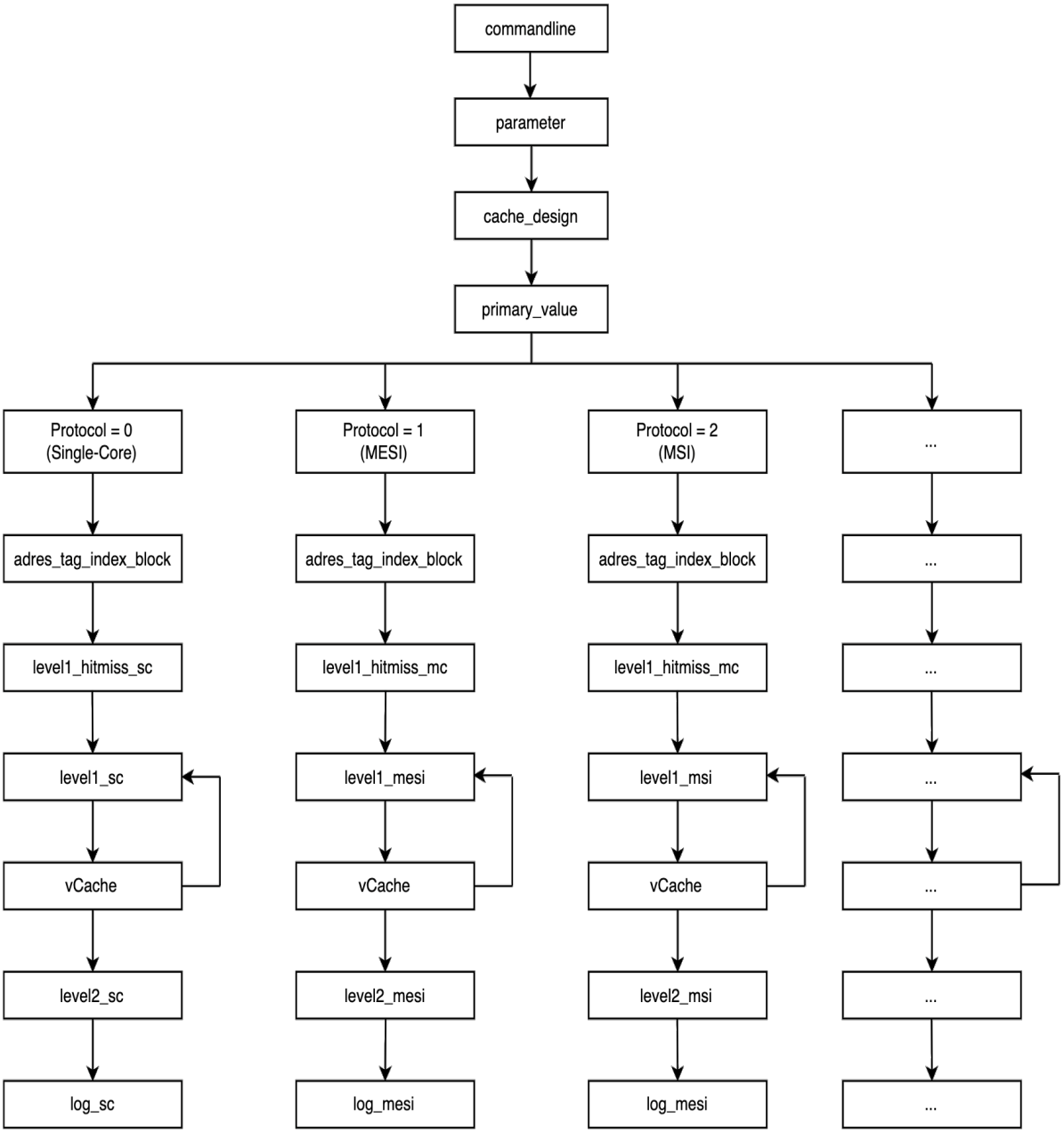

Figure B.2: Detailed Flowchart of NRC-SIM

Source: Primary

When using NRC-SIM, the user only needs to select their desired number of victim lines and submit them along with their other desired parameters, i.e. core count, coherence protocol, cache size(s), etc. Users are able to select up to 16 victim lines, and in the case of a desired core count greater than one, each core will contain an L1 victim cache based on the user's desired victim

cache size, For example, a 4-core simulation with a desired victim cache size of 6 lines will contain 24 total victim lines - 6 victim lines per core.

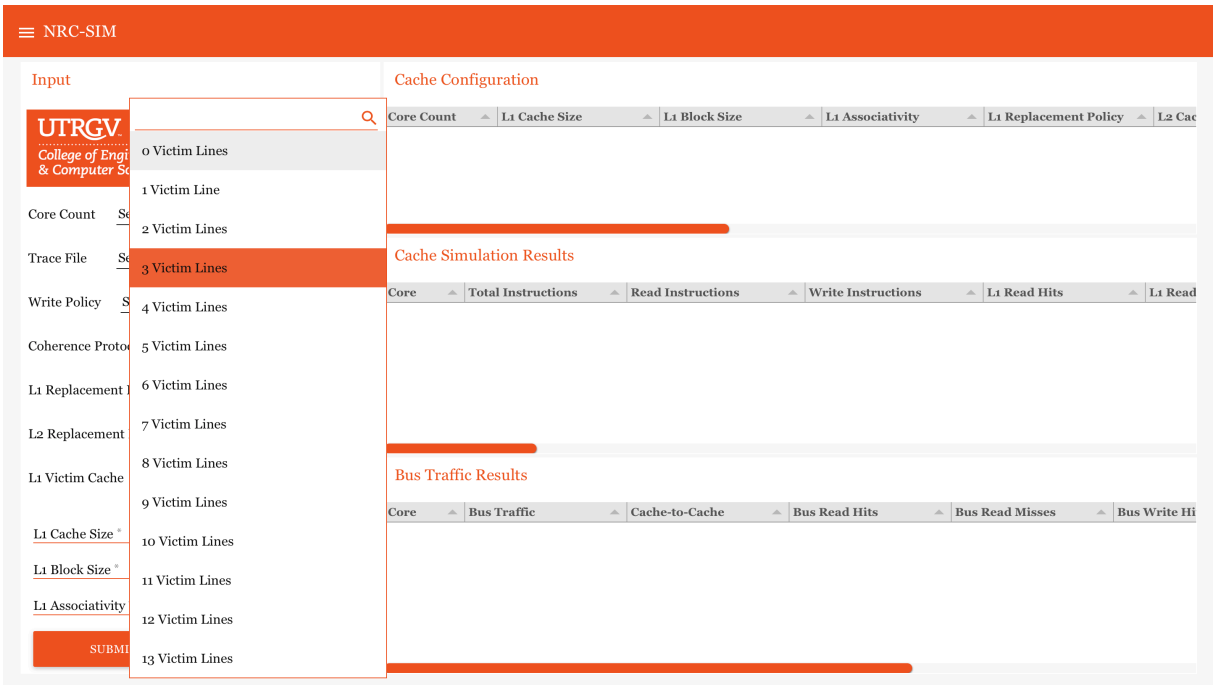

Figure B.3: Victim Cache Size Selection in NRC-SIM

Source: Primary

APPENDIX C

APPENDIX C

C++ SOURCE CODE

Header.h

```
#include <stdio.h>
#include <stdlib.h>
#include <iomanip>
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <bitset>
#include <bits/stdc++.h>

#include "variable.h"
#include "build.h"
#include "myfunction.h"

using namespace std;
```

Variable.h

```
int coreno1, coreno2, coreno, protocol, wp, vL1, vL2, victimLines;
int queueIL1 = 0, queueIL2 = 0;
int countMRU = 0;
int vCacheCount = 0;

int victimIndex = 0;
int vHit = 0;
int globalVictimHit = 0;

//in bits

int ind1, blk1, tbit1, ca1, asc1, rp1;
int ind2, blk2, tbit2, ca2, asc2, rp2;

int para1, para2; // parameter

int core;

std::string line, cmdl1, cmdl2;

std::string parametersL1;
std::string parametersL2;

std::string adrs; // address

unsigned int dec_adrs; // address in decimal
```

```
unsigned int  block1, index1, tag1;
unsigned int block2, index2, tag2;

char type; //instruction type

int a, n, out1;
int b, m, out2;

//ht= hit thread, tt=target thread
int ht1, tt1;
int ht2, tt2;
int th2;
int matchl2;

// con_adrs = convert address
unsigned int conv_adrs21, conv_adrs12;
int p, q, r;
int u, v, w;

unsigned int  newblock1, newindex1, newtag1;
unsigned int newblock2, newindex2, newtag2;
char newstat1, newstat2;
```

Build.h

```
// t=tag, v=validity, db= dirty bit, stat = status of the data

struct level1
{
   int t1, v1, lru1, mru1, db1;
   int data1, vData, vValid;
   char stat1, vState;
   int victimTag;
};

struct level1** L1[4096];
struct level1** victimCache = NULL;

struct level2
{
   int t2, v2, lru2, mru2, db2;
   char stat2;
};
```

```
struct level2** L2[4096];


//format output values with commas, i.e. 1,000,000
class comma_numpunct : public std::numpunct<char>
{
  protected:
    virtual char do_thousands_sep() const
    {
        return ',';
    }

    virtual std::string do_grouping() const
    {
        return "\03";
    }
};


// temporary cache

struct temp
{
    int temp_t, temp_v, temp_lru, temp_db;
    int temp_data;
    char temp_stat;
};

struct temp tempcash;


struct lookup1
{
    int mcore1, mthread1;
    char mstat1;
} ;

struct lookup1 matchcore1[4096];

struct lookup2
{
    int mcore2, mthread2;
    char mstat2;
} ;

struct lookup2 matchcore2[4096];
```

```
// for log
double probe1=0;

double bustraffic=0; //bus traffic for l-1 cache threads
double totalctoc=0;
double totalins=0;
double totalmissl1=0;
double missratel1=0;
double missratel2=0;
double hitratel2=0; //  hit rate for l-2 (specially single core)

double totalrml1=0;
double totalwml1=0;
double totaliml1=0;
double totalread1=0;

double totalcalll2=0; //total l-2 cache call
double totalmissl2=0; // total l-2 cache miss

double totalrhpl1=0;
double totalwhpl1=0;
double totalrhbl1=0;
double totalwhbl1=0;

double totalrml2=0;
double totalwml2=0;
double totaliml2=0;


struct archive0
{
   int rmem=0, wmem=0;
   int totalread = 0, totalwrite=0, totalins=0;
   int trade=0; // between l-1 and l-2
   int busmiss=0;

}  cash0log;

struct archive1
{
   double rhit1, whit1, rmiss1, wmiss1, ihit1, imiss1;

} cash1log;
```

```
struct archive2
{
   double rhit2, whit2, rmiss2, wmiss2, ihit2, imiss2;

} cash2log;

struct archive3
{
   double rhl1, whl1, rml1, wml1, ihl1, iml1;
   double rhl2, whl2, rml2, wml2, ihl2, iml2;

   double  mrate; // miss rate in private cache;
   double emrate; //effective miss rate in l-1

   int rhp1=0; // read hit private
   int whp1=0; // write hit private
   int ihp1=0; // ins hit private

   int rhp2=0; // read hit private
   int whp2=0; // write hit private
   int ihp2=0; // ins hit private

   double rhb=0; // read hit bus
   double whb=0; // wite hti bus
   double rmb=0; // read miss bus
   double wmb=0; // read miss bus
   double ihb=0; // ins hit bus
   double imb=0; // ins hit bus

   int mih1=0; // modified read hit
   int eih1=0; // exclusive read hit
   int sih1=0; // shared read hit

   int mrh1=0; // modified read hit
   int erh1=0; // exclusive read hit
   int srh1=0; // shared read hit

   int mwh1=0; // modified write hit
   int ewh1=0; // exclusive write hit
   int swh1=0; // shared write hit

   int mih2=0; // modified read hit
   int eih2=0; // exclusive read hit
   int sih2=0; // shared read hit

   int mrh2=0; // modified read hit
```

```
    int erh2=0; // exclusive read hit
    int srh2=0; // shared read hit

    int mwh2=0; // modified write hit
    int ewh2=0; // exclusive write hit
    int swh2=0; // shared write hit

    int vrh1=0; // valid read hit
    int drh1=0; // dirty read hit

    int vwh1=0; // valid write hit
    int dwh1=0; // dirty write hit

    int vih1=0; // valid read hit
    int dih1=0; // dirty read hit

    int vrh2=0; // valid read hit
    int drh2=0; // dirty read hit

    int vwh2=0; // valid write hit
    int dwh2=0; // dirty write hit

    int vih2=0; // valid read hit
    int dih2=0; // dirty read hit

    double bu=0; // bus update
    //int bt=0; // bus traffic
    int pw=0; // processor write to l-1

    double ctoc=0; // cache to cache in level-1
    int exchange=0; // exchange between level-1 and level-2


    //victim r/w/z
    long double vrh = 0, vrm = 0, vwh = 0, vwm = 0, vih = 0, vim = 0;
};

struct archive3 corelog[4096];
```

NRC-SIM.cpp

```
#include "header.h"
#include <cmath>
#include <locale>
```

```
int main(int argc, char ** argv)
{
   std::locale myLocale(std::locale(), new comma_numpunct());
   std::cout.imbue(myLocale);

   srand((unsigned) time(NULL));

   commandline(argc, argv);
   parameter();
   cache_design();
   primary_value();

   cout /*<< "Core Count:"*/<< coreno<< endl;
   if (ca2!=0)
   {
      cout /*<<"L1 Cache Size:"*/<<(1<<ca1)<< " Bytes" << std::endl;//'\t';
      cout << /*"L1 Block size:"<<*/(1<<blk1)<< " Bytes" << std::endl;
      //cout << /*"L1 Associativity:";//<<*/(1<<asc1) << std::endl;
      //int n1 = (1<<asc1);
      if(asc1 == 0)
         std::cout << "Direct-Mapped" << std::endl;
      else if(ind1 == 0)
         std::cout << "Fully Associative" << std::endl;
      else
         std::cout << pow(2, asc1) << "-Way Set Associative" << std::endl;


      if (rp1==1)
      {cout << /*"L1 Replacement Policy:"<<*/"Least Recently Used"<<std::endl/*'\t'<<'\t'*/;}
      else if(rp1 == 2)
         std::cout << /*"L1 Replacement Policy:" << */"Random Replacement" << std::endl;
      else if(rp1 == 3)
         std::cout << /*"L1 Replacement Policy:" << */"Round-Robin Replacement" << std::endl;
      else if(rp1 == 4)
         std::cout << /*"L1 Replacement Policy:" << */"Most Recently Used" << std::endl;


      cout <</*"L2 Cache Size:"<<*/(1<<ca2)<< " Bytes" <<endl;
      //cout << "L1 Block size:"<<(1<<blk1)<< std::endl;//'\t'<<'\t';
      cout << /*"L2 Block size:"<<*/(1<<blk2)<< " Bytes" <<endl;
      //cout << "L1 Associativity:"<<(1<<asc1) << std::endl;//<<'\t'<<'\t';
      //cout << /*"L2 Associativity:";//<<*/(1<<asc2)<<endl;
      //int n2 = (1<<asc2);

      if(asc2 == 0)
         std::cout << "Direct-Mapped" << std::endl;
```

```
    else if(ind2 == 0)
        std::cout << "Fully Associative" << std::endl;
    else
        std::cout << pow(2, asc2) << "-Way Set Associative" << std::endl;

    //if (rp1==1)
    //{cout << "L1 Replacement Policy:"<<"LRU"<<std::endl/*'\t'<<'\t'*/;}
    if (rp2==1)
    {cout << /*"L2 Replacement Policy:"<<*/"Least Recently Used"<<endl;}
    else if(rp2 == 2)
        std::cout << /*"L2 Replacement Policy:" << */"Random Replacement" << std::endl;
    else if(rp2 == 3)
        std::cout << /*"L2 Replacement Policy:" << */"Round-Robin Replacement" << std::endl;
    else if(rp2 == 4)
        std::cout << /*"L2 Replacement Policy:" << */"Most Recently Used" << std::endl;
}

if (ca2==0)     //single core, only L1
{
    cout <</*"L1 Cache Size:"<<*/(1<<ca1) << " Bytes" << std::endl;
    cout <</* "L1 Block Size:"<<*/(1<<blk1)<< " Bytes" << std::endl;
    //cout << "L1 Associativity:";//<<(1<<asc1) << std::endl;
    if(asc1 == 0)
        std::cout << "Direct-Mapped" << std::endl;
    else if(ind1 == 0)
        std::cout << "Fully Associative" << std::endl;
    else
        std::cout << pow(2, asc1) << "-Way Set Associative" << std::endl;

    if (rp1==1)
    {cout << /*"L1 Replacement Policy:"<<*/"Least Recently Used"<<std::endl;}
    else if(rp1 == 2)
        std::cout << /*"L1 Replacement Policy:" << */"Random Replacement" << std::endl;
    else if(rp1 == 3)
        std::cout << /*"L1 Replacement Policy:" << */"Round-Robin Replacement" << std::endl;
    else if(rp1 == 4)
        std::cout << /*"L1 Replacement Policy:" << */"Most Recently Used" << std::endl;

}

//cout << "\ncore no:"<< coreno<< endl;

if(protocol==0)
{cout << /*"Coherence Protocol:*/"Single-Core"<< endl;}

if(protocol==1)
```

```
{cout << /*"Coherence Protocol:*/"MESI"<< endl;}

if(protocol==2)
{cout << /*"Coherence Protocol:*/"MSI"<< endl;}

if(protocol==3)
{cout << /*"Coherence Protocol:*/"FIREFLY"<< endl;}

if (wp==1)
{cout<</*"Write Policy:*/"Write-Back"<<endl;}

if (wp==2)
{cout<</*"Write Policy:*/"Write-Through"<<endl;}

ifstream filename (argv[argc-1]);

if(protocol==0)//Singal core
{
   //cout<<"\n\t-------- simulation started -------\n";

   while (filename >> core >> type >> adrs)
   {
      out2 = 2;
      adres_tag_index_block();
      level1_hitmiss_sc();
      //only_l1_sc();
      level1_sc();
      level2_sc();
   }
      log_sc();
}

else if (protocol==1)// multi-core protocol: mesi
{
   //cout<<"\n\t-------- simulation started -------\n";

   while (filename >> core >> type >> adrs)
   {
      if (ca2==0)
      {
         adres_tag_index_block();
         level1_hitmiss_mc();
         only_l1_mesi();
      }

      if(ca2!=0)
```

```
        {
            out2 = 2;
            adres_tag_index_block();
            level1_hitmiss_mc();
            level1_mesi();
            level2_mesi();
        }
    }
    if(ca2==0)
        log_mesi_only_l1();
    if(ca2!=0)
        log_mesi();
}

 else if (protocol==2)// multi-core protocol: msi
{
    //cout<<"\n\t-------- simulation started -------\n";

    while (filename >> core >> type >> adrs)
    {
        if (ca2==0)
        {
            adres_tag_index_block();
            level1_hitmiss_mc();
            only_l1_msi();
        }

        if(ca2!=0)
        {
            out2 = 2;
            adres_tag_index_block();
            level1_hitmiss_mc();
            level1_msi();
            level2_msi();
        }
    }

    if(ca2==0)
        log_mesi_only_l1();
    if(ca2!=0)
        log_mesi();
}

 else if (protocol==3)// multi-core protocol: firefly
{
    //cout<<"\n\t-------- simulation started -------\n";
```

```
        while (filename >> core >> type >> adrs)
        {
            if (ca2==0)
            {
                adres_tag_index_block();
                level1_hitmiss_mc();
                only_l1_firefly();
            }

            if(ca2!=0)
            {
                out2 = 2;
                adres_tag_index_block();
                level1_hitmiss_mc();
                level1_firefly();
                level2_firefly();
            }
        }

        if(ca2==0)
            log_firefly_only_l1();
        if(ca2!=0)
            log_firefly();
    }
}
```

Functions.h

```
#include <locale>
#include <cstdlib>
using namespace std;

void vCache();
int randomReplacementL1();
int randomReplacementL2();
int queueReplacementL1();
int queueReplacementL2();

int highest_mru1();
int highest_mru2();
void check_mru1(int coreid1, int threadid1);
void check_mru2(int threadid2);
void update_mru1(int coreid1, int threadid1);
void update_mru2(int threadid2);
```

```
void commandline(int argc, char **argv);
void parameter();
void cache_design();
void primary_value();

void adres_tag_index_block();

void level1_hitmiss_sc();
void only_l1_sc();
void level1_sc();
void level2_sc();

void level1_hitmiss_mc();

void only_l1_mesi();
void only_l1_msi();
void only_l1_firefly();

void level1_mesi();
void level2_mesi();

void level1_msi();
void level2_msi();

void level1_firefly();
void level2_firefly();

void copy_l1_to_l2(int targetthread1, int targetthread2);
void update_l2_from_l1(int targetthread1);
int find_l2(int coreid2);

int highest_lru1();
int highest_lru2();
void update_lru1(int coreid1, int threadid1);
void check_lru1(int coreid1, int threadid1);
void check_lru2(int threadid2);
void update_lru2(int threadid2);
void invalidate_l2(int coreid2);
int searchcores1(int coreid1);
int searchcores2(int coreid2);

void log_sc();
void log_mesi_only_l1();
void log_mesi();
```

```
void log_firefly_only_l1();
void log_firefly();



void vCache()
{
  for(int i = 0; i < victimLines; i++)
  {
    if(victimCache[core][i].victimTag == tag1)  //victimHit
    {
      int tempTag = victimCache[core][i].victimTag;
      int tempData = victimCache[core][i].vData;
      int tempState = victimCache[core][i].vState;

      victimCache[core][i].victimTag = L1[core][tt1][index1].t1;
      victimCache[core][i].vData = L1[core][tt1][index1].data1;
      victimCache[core][i].vState = L1[core][tt1][index1].stat1;

      L1[core][tt1][index1].t1 = tempTag;
      L1[core][tt1][index1].data1 = tempData;
      L1[core][tt1][index1].stat1 = tempState;

      if(type == 'r')
      {
        corelog[core].rml1--;
        corelog[core].rhl1++;
        corelog[core].vrh++;
      }
      else if(type == 'w')
      {
        corelog[core].wml1--;
        corelog[core].whl1++;
        corelog[core].vwh++;
      }
      else if(type == 'z')
      {
        corelog[core].iml1--;
        corelog[core].ihl1++;
        corelog[core].vih++;
      }

      globalVictimHit = 1;
      return;
    }
  }
```

```
  //victim miss
  globalVictimHit = 0;
  if(globalVictimHit == 0)
  {
    if(type == 'r')
      corelog[core].vrm++;
    else if(type == 'w')
      corelog[core].vwm++;
    else if(type == 'z')
      corelog[core].vim++;
  }

  //send evicted victim data to L2
  if(victimCache[core][victimIndex].vState == 'm' || victimCache[core][victimIndex].vState ==
'd')
  {
    L2[core][tt2][index2].t2 = victimCache[core][victimIndex].victimTag;
    L2[core][tt2][index2].v2 = victimCache[core][victimIndex].vValid;
    L2[core][tt2][index2].stat2 = victimCache[core][victimIndex].vState;
  }

  //copy L1 eviction to victim cache
  victimCache[core][victimIndex].victimTag = L1[core][tt1][index1].t1;
  victimCache[core][victimIndex].vData = L1[core][tt1][index1].data1;
  victimCache[core][victimIndex].vState = L1[core][tt1][index1].stat1;
  victimCache[core][victimIndex].vValid = L1[core][tt1][index1].v1;


  if(victimIndex < victimLines-1)
    victimIndex++;
  else
    victimIndex = 0;
}



int randomReplacementL1()
{
  int n = (1 << asc1);
  int randomNumber = rand() % n;

  return randomNumber;
}

int randomReplacementL2()
```

```
{
  int n = (1 << asc2);
  int randomNumber = rand() % n;

  return randomNumber;
}

int queueReplacementL1()
{
  int n = (1 << asc1);
  int temp = 0;

  for(int i = 0; i < n; i++)
  {
    if(i == queueIL1)
    {
      temp = i;
      break;
    }
  }

  if(queueIL1 < n)
    queueIL1++;
  else
    queueIL1 = 0;

  if(countMRU < n)
    countMRU++;

  return temp;
}

int queueReplacementL2()
{
  int n = (1 << asc2);
  int temp = 0;

  for(int i = 0; i < n; i++)
  {
    if(i == queueIL2)
    {
      temp = i;
      break;
    }
  }
```

```
  if(queueIL2 < n)
    queueIL2++;
  else
    queueIL2 = 0;

  return temp;
}

void update_lru1(int coreid1, int threadid1)
{
  int up1, n;

  up1 = L1[coreid1][threadid1][index1].lru1;
  n = 1 << asc1;

  for (int i = 0; i < n; i++)
  {
    if (L1[coreid1][i][index1].lru1 < up1)
    {
      L1[coreid1][i][index1].lru1++;
    }
  }

  L1[coreid1][threadid1][index1].lru1 = 0;
}

void check_lru1(int coreid1, int threadid1)
{
  int check1, n;

  check1 = L1[coreid1][threadid1][index1].lru1;
  n= 1 << asc1;

  for (int i = 0; i < n; i++)
  {
    if (L1[coreid1][i][index1].lru1 > check1)
    {
      L1[coreid1][i][index1].lru1--;
    }
  }

  L1[coreid1][threadid1][index1].lru1 = (n-1);
}

void check_lru2(int coreid2,int threadid2)
{
```

```
  int check2, m;

  check2 = L2[coreid2][threadid2][index2].lru2;
  m = 1 << asc2;

  for (int i = 0; i < m; i++)
  {
    if (L2[coreid2][i][index2].lru2 > check2)
    {
      L2[coreid2][i][index2].lru2--;
    }
  }

  L2[coreid2][threadid2][index2].lru2 = (m-1);
}

void update_lru2(int coreid2, int threadid2)
{
  int up2, m;

  up2 = L2[coreid2][threadid2][index2].lru2;
  m = 1 << asc2;

  for (int i = 0; i < m; i++)
  {
    if (L2[coreid2][i][index2].lru2 < up2)
    {
      L2[coreid2][i][index2].lru2++;
    }
  }

  L2[coreid2][threadid2][index2].lru2 = 0;
}

int highest_lru2()
{
  int cc, m, hlthread2;

  cc = L2[core][0][index2].lru2;
  m = 1 << asc2;
  for (int i = 0; i < m; i++)
  {
    if (cc <= L2[core][i][index2].lru2)
    {
      hlthread2 = i;
      cc = L2[core][i][index2].lru2;
```

```
    }
  }
  return hlthread2;
}

int highest_lru1()
{
  int aa, n, hlthread1;

  aa = L1[core][0][index1].lru1;

  n = 1 << asc1;

  for (int i = 0; i < n; i++)
  {
    if (aa <= L1[core][i][index1].lru1)
    {
      hlthread1 = i;
      aa = L1[core][i][index1].lru1;
    }
  }
  return hlthread1;
}

void update_mru1(int coreid1, int threadid1)
{
  int up1, n;

  up1 = L1[coreid1][threadid1][index1].mru1;
  n = 1 << asc1;

  for (int i = 0; i < n; i++)
  {
    if (L1[coreid1][i][index1].mru1 < up1)
    {
      L1[coreid1][i][index1].mru1 = 0;
    }
  }

  L1[coreid1][threadid1][index1].mru1++;
}

void check_mru1(int coreid1, int threadid1)
{
  int check1, n;
```

```
  check1 = L1[coreid1][threadid1][index1].mru1;
  n= 1 << asc1;

  for (int i = 0; i < n; i++)
  {
    if (L1[coreid1][i][index1].mru1 > check1)
    {
      L1[coreid1][i][index1].mru1++;
    }
  }

  L1[coreid1][threadid1][index1].mru1 = 0;
}

void check_mru2(int coreid2,int threadid2)
{
  int check2, m;

  check2 = L2[coreid2][threadid2][index2].mru2;
  m = 1 << asc2;

  for (int i = 0; i < m; i++)
  {
    if (L2[coreid2][i][index2].mru2 > check2)
    {
      L2[coreid2][i][index2].mru2++;
    }
  }

  L2[coreid2][threadid2][index2].mru2 = 0;
}

void update_mru2(int coreid2, int threadid2)
{
  int up2, m;

  up2 = L2[coreid2][threadid2][index2].mru2;
  m = 1 << asc2;

  for (int i = 0; i < m; i++)
  {
    if (L2[coreid2][i][index2].mru2 < up2)
    {
      L2[coreid2][i][index2].mru2 = 0;
    }
  }
```

```
  L2[coreid2][threadid2][index2].mru2++;
}

int highest_mru2()
{
  int cc, m, hlthread2;

  cc = L2[core][0][index2].mru2;
  m = 1 << asc2;
  for (int i = 0; i < m; i++)
  {
    if (cc >= L2[core][i][index2].mru2)
    {
      hlthread2 = i;
      cc = L2[core][i][index2].mru2;
    }
  }
  return hlthread2;
}

int highest_mru1()
{
  int aa, n, hlthread1;

  aa = L1[core][0][index1].mru1;

  n = 1 << asc1;

  for (int i = 0; i < n; i++)
  {
    if (aa >= L1[core][i][index1].mru1)
    {
      hlthread1 = i;
      aa = L1[core][i][index1].mru1;
    }
  }
  return hlthread1;
}

int searchcores1(int coreid1)
{
  int n, match1=0;

  n=1<<asc1;
```

```
  for (int j=0; j<coreno;j++)
  {
    if (j!=coreid1)
    {
      for (int i=0; i<n;i++)
      {
        if (L1[j][i][index1].v1==1)
        {
          if (L1[j][i][index1].t1==tag1)
          {
            matchcore1[match1].mcore1=j;
            matchcore1[match1].mthread1=i;
            matchcore1[match1].mstat1=L1[j][i][index1].stat1;
            match1++;
          }
        }
      }
    }
  }

  return match1;
}

int searchcores2(int coreid2)
{
  int m, match2=0;

  m=1<<asc2;

  for (int y=0; y<coreno;y++)
  {
    if (y!=coreid2)
    {
      for (int x=0; x<m;x++)
      {
        if (L2[y][x][index2].v2==1)
        {
          if (L2[y][x][index2].t2==tag2)
          {
            matchcore2[match2].mcore2=y;
            matchcore2[match2].mthread2=x;
            matchcore2[match2].mstat2=L2[y][x][index2].stat2;
            match2++;
          }
        }
      }
```

```
    }
  }

  return match2;
}

void update_l2_from_l1(int coreid1, int targetthread1)
{
  w = L1[coreid1][targetthread1][index1].t1;
  newtag2 = w>>(tbit1-tbit2);

  int th2 = 0;
  int m = 1 << asc2;
  for (int j = 0; j < m; j++)
  {
    if (L2[coreid1][j][index2].v2 == 1)
    {
      if (L2[coreid1][j][index2].t2 == newtag2)
      {
        th2 = j;
        break;
      }
    }
  }

  if (L2[coreid1][th2][newindex2].stat2='m')
  {
    //log
    //update memory
    cash0log.wmem++;

    L2[coreid1][th2][newindex2].stat2='e';
  }

  // after updating l2
  L2[coreid1][th2][newindex2].stat2='m';
}

int find_l2(int coreid2)
{
  int th2 = 0;
  int m = 1 << asc2;

  for (int j = 0; j < m; j++)
  {
    if (L2[coreid2][j][index2].v2 == 1)
```

```
    {
      if (L2[coreid2][j][index2].t2 == tag2)
      {
        th2 = j;
        break;
      }
    }
  }
  return th2;
}

void invalidate_l2(int coreid2)
{
  int matchl2= 0; //highest_lru2();
  if(parametersL2.back() == 52)
    matchl2 = highest_mru2();
  if(parametersL2.back() == 49)
    matchl2 = highest_lru2();

  int yy = 0; int mm = 1 << asc2;
  for (int i = 0; i < mm; i++)
  {
    if (L2[coreid2][i][index2].v2 == 1)
    {
      if (L2[coreid2][i][index2].t2 == tag2)
      {
        matchl2 = i;
        break;
      }
    }
  }

  //invalid copy in l2
  L2[coreid2][matchl2][newindex2].stat2='i';
  L2[coreid2][matchl2][newindex2].v2=0;
}


void commandline(int argc, char **argv)
{
  //  string line;
  for (int i = 1; i < argc; i++)
  {
    line = argv[i];
    if (line[0] == 45) // 45 is ASCII value of '-'
    {
```

```
if (line[1] == 108) // 108 is ASCII value of 'l'
{
  while (i == 1)
  {
    if (line[2] == 49) // 49 is ASCII value of '1'
    {
      i++;

      // Get_Cache_parameter(argv[i]);
      cmdl1 = argv[i];
      parametersL1 = cmdl1;
    }

    else
    {
      std::cout << "error l-1 command" << std::endl;
      break;
    }
  }

  while (i == 3)
  {
    if (line[2] == 50) //// 50 is ASCII value of '2'
    {
      i++;

      // Get_Cache_parameter for level 2 (argv[i]);
      cmdl2 = argv[i];
      parametersL2 = cmdl2;
    }

    else
    {
      std::cout << "error l-2 command" << std::endl;
      break;
    }
  }
}

else if (line[1] == 67) // 67 is ASCII value of 'C'
{
  stringstream ss;
  ss << line[2];
  ss >> coreno1;
  coreno=coreno1;
```

```
      stringstream rr;
      rr<<line[3];
      rr>>coreno2;

      if(coreno2!=0)
      {coreno=coreno*10+coreno2;}
    }

    else if (line[1] == 80) // 80 is ASCII value of 'P'
    {
      stringstream pp;
      pp << line[2];
      pp >> protocol;
    }

    // for write policy
    else if(line[1]==87) // 87 is ascii value of "W"
    {
      stringstream qq;
      qq << line[2];
      qq >> wp;
    }

    //victim cache
    else if(line[1]==86)  //86 is ASCII of 'V'
    {
      stringstream vv;
      vv << line[2];
      vv >> vL1;
      victimLines=vL1;

      stringstream bb;
      bb<<line[3];
      bb>>vL2;

      if(vL2!=0)
        victimLines=victimLines*10+vL2;
    }
  }
 }
}

void parameter()
{
 // parameter for level-1
 int j = 0; int para1[5] = {0, 0, 0, 0, 0};
```

```
for (int i = 0; i <= 4; i++)
{
  if (cmdl1[0] != 58)
  {
    char m = cmdl1[j];
    para1[i] = (int)m - 48;
    j = j + 1;

    if (j < cmdl1.length())
    {
      m = cmdl1[j];
      while (m != 58)
      {
        para1[i] = para1[i] * 10;
        para1[i] = para1[i] + ((int)m - 48);
        j = j + 1;
        m = cmdl1[j];
      }
      j = j + 1;
    }
    else
    {
      break;
    }
  }
  else
  {
    cout << "command line error" << endl;
  }
}

ca1 = para1[0];
blk1 = para1[1];
asc1 = para1[2];

rp1 = para1[3];
ind1 = ca1 - blk1 - asc1;
tbit1 = 48 - blk1 - ind1;

// parameter for level-2
int k = 0; int para2[5] = {0, 0, 0, 0, 0};

for (int i = 0; i <= 4; i++)
{
  if (cmdl2[0] != 58)
```

```
    {
      char m = cmdl2[k];
      para2[i] = (int)m - 48;
      k = k + 1;

      if (k < cmdl2.length())
      {
        m = cmdl2[k];
        while (m != 58)
        {
          para2[i] = para2[i] * 10;
          para2[i] = para2[i] + ((int)m - 48);
          k = k + 1;
          m = cmdl2[k];
        }
        k = k + 1;
      }
      else
      {
        break;
      }
    }
    else
    {
      cout << "command line error" << endl;
    }
  }

  ca2 = para2[0];
  blk2 = para2[1];
  asc2 = para2[2];

  rp2 = para2[3];
  ind2 = ca2 - blk2 - asc2;
  tbit2 = 48 - blk2 - ind2;
}

void cache_design()
{
  // level-1 cache
  for (int i = 0; i < coreno; i++)
  {
    L1[i] = new struct level1 *[1 << asc1];

    for (int n = 0; n < (1 << asc1); n++)
    {
```

```
      L1[i][n] = new struct level1[1 << ind1];
    }
  }

  // level-2 cache
  for (int x = 0; x < (1 << coreno); x++)
  {
    L2[x] = new struct level2 *[1 << asc2];

    for (int m = 0; m < (1 << asc2); m++)
    {
      L2[x][m] = new struct level2[1 << ind2];
    }
  }

  //victim cache
  victimCache = (level1 **)malloc(coreno * sizeof(struct level1*));
  for (int i = 0; i < coreno; i++)
    victimCache[i] = (level1 *)malloc(victimLines * sizeof(struct level1));
}

void primary_value()
{
  // for level-1
  for (int i = 0; i < coreno; i++)
  {
    for (int j = 0; j < (1 << asc1); j++)
    {
      for (int k = 0; k < (1 << ind1); k++)
      {
        L1[i][j][k].t1 = -1;
        L1[i][j][k].v1 = 0;
        L1[i][j][k].lru1 = j;
        L1[i][j][k].mru1 = j;
        L1[i][j][k].db1 = 1;
        L1[i][j][k].stat1 = 'i';
        L1[i][j][k].data1 = -1;
      }
    }
  }

  // for level-2
  for (int x = 0; x < (1 << coreno); x++)
  {
    for (int y = 0; y < (1 << asc2); y++)
    {
```

```
      for (int z = 0; z < (1 << ind2); z++)
      {
        L2[x][y][z].t2 = -1;
        L2[x][y][z].v2 = 0;
        L2[x][y][z].lru2 = y;
        L2[x][y][z].mru2 = y;
        L2[x][y][z].db2 = 1;
        L2[x][y][z].stat2 = 'i';
      }
    }
  }

  // assign value
  for (int i = 0; i < coreno; i++)
  {
    corelog[i].rhl1 = 0;
    corelog[i].whl1 = 0;
    corelog[i].rml1 = 0;
    corelog[i].wml1 = 0;
    corelog[i].ihl1 = 0;
    corelog[i].iml1 = 0;

    corelog[i].rhl2 = 0;
    corelog[i].whl2 = 0;
    corelog[i].rml2 = 0;
    corelog[i].wml2 = 0;
    corelog[i].ihl2 = 0;
    corelog[i].iml2 = 0;
  }
}

void adres_tag_index_block()
{
  // get address in decimal
  dec_adrs = std::stoul(adrs, nullptr, 16);

  // get tag, index, block address in decimal for level-1
  block1 = dec_adrs & ((1 << blk1) - 1);
  index1 = (dec_adrs >> blk1) & ((1 << ind1) - 1);
  tag1 = (dec_adrs >> (ind1 + blk1)) & ((1 << tbit1) - 1);

  // get tag, index, block address in decimal for level-2
  block2 = dec_adrs & ((1 << blk2) - 1);
  index2 = (dec_adrs >> blk2) & ((1 << ind2) - 1);
  tag2 = (dec_adrs >> (ind2 + blk2)) & ((1 << tbit2) - 1);
}
```

```
void level1_hitmiss_sc()
{
  int a = 2;
  n = 1 << asc1;

  for (int i = 0; i < n; i++)
  {
    if (L1[core][i][index1].v1 == 1)
    {
      if (L1[core][i][index1].t1 == tag1) // in case of hit in level-1
      {
        ht1 = i;
        a = 1;
        out1 = 1;

        // log
        if (type == 'z')
        {corelog[core].ihl1++;}

        if (type == 'r')
        {corelog[core].rhl1++;}

        if (type == 'w')
        {corelog[core].whl1++;}

        break;
      }
    }
  }

  if (a == 2) // in case of miss in level-1
  {
    out1 = 0;

    // log
    if (type == 'z')
    {corelog[core].iml1++;}

    if (type == 'r')
    {corelog[core].rml1++;}

    if (type == 'w')
    {corelog[core].wml1++;}
  }
}
```

```
void level1_sc()
{
  if (out1 == 1) // when hit in level-1
  {
    //find copy in l2
    th2=find_l2(core);


    if (type=='z'&& wp==1)
    {
      update_lru1(core, ht1);
      update_mru1(core, ht1);
    }

    if (type == 'z'&& wp==2) // write-through
    {
      update_lru1(core, ht1);
      update_mru1(core, ht1);
    }

    if (type == 'r' && wp==1)
    {
      update_lru1(core, ht1);
      update_mru1(core, ht1);
    }

    if (type == 'r' && wp==2) // write-through
    {
      update_lru1(core, ht1);
      update_mru1(core, ht1);
    }

    if (type == 'w' && wp==1)
    {
      L1[core][ht1][index1].data1 = block1; // copy data to cache from processor
      L1[core][ht1][index1].db1 = 1;        // dirtybit =1

      //update l2
      L2[core][th2][index2].db2 = 1;

      update_lru1(core, ht1);
      update_lru2(core, th2);
```

```
    update_mru1(core, ht1);
    update_mru2(core, th2);
  }

  if (type == 'w' && wp==2) //write-through
  {
    L1[core][ht1][index1].data1 = block1; // copy data to cache from processor

    L2[core][tt2][index2].t2=tag2;
    L2[core][tt2][index2].v2=1;

    //log
    cash0log.wmem++;

    update_lru1(core, ht1);
    update_lru2(core, tt2);

    update_mru1(core, ht1);
    update_mru2(core, tt2);
  }
}

if (out1 == 0) // when miss in level-1 check in level-2
{
  //check victim cache
  if(L1[core][tt1][index1].v1 == 1 && victimLines > 0)
  {
    vCache();
  }

  //replacement policy
  if(parametersL1.back() == 49)     //49 is ASCII of 1; check for LRU
    tt1 = highest_lru1();
  else if(parametersL1.back() == 50)   //50 is ASCII of 2; Random
    tt1 = randomReplacementL1();
  else if(parametersL1.back() == 51)    //51 is ASCII of 3; queue
    tt1 = queueReplacementL1();
  else if(parametersL1.back() == 52)     //52 is ASCII of 4; MRU
      tt1 = highest_mru1();


  //replacement policy
  if(parametersL2.back() == 49)         //LRU
    tt2 = highest_lru2();
  else if(parametersL2.back() == 50)    //Random
    tt2 = randomReplacementL2();
```

```
    else if(parametersL2.back() == 51)    //queue
      tt2 = queueReplacementL2();
    else if(parametersL1.back() == 52)    //MRU
      tt2 = highest_mru2();


    b = 0, m = 1 << asc2;
    for (int i = 0; i < m; i++)
    {
      if (L2[core][i][index2].v2 == 1)
      {
        if (L2[core][i][index2].t2 == tag2)
        {
          ht2 = i;
          b = 1;
          out2 = 1; // in case of hit in level-2

          break;
        }
      }

      if (b == 0) // in case of miss in level-2
      {
        out2 = 0;
      }
    }
  }
}

void level2_sc()
{
  if (out2==1) // when hit in level-2
  {
    //log
    if (type == 'z')
    {corelog[core].ihl2++;}

    if (type == 'r')
    {corelog[core].rhl2++;}

    if (type == 'w')
    {corelog[core].whl2++;}

    if (type == 'z' && wp==1)
    {
      //copy from level-2 hit thread to l-1 target thread
```

```
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].data1 = block1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].db1=0;

  update_lru1(core, tt1);
  update_lru2(core, ht2);

  update_mru1(core, tt1);
  update_mru2(core, ht2);
}

if (type == 'z' && wp==2) // write-through
{
  //copy from level-2 hit thread to l-1 target thread
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].data1 = block1;
  L1[core][tt1][index1].v1=1;

  update_lru1(core, tt1);
  update_lru2(core, ht2);

  update_mru1(core, tt1);
  update_mru2(core, ht2);
}

if (type == 'r' && wp==1)
{
  //copy from level-2 hit thread to l-1 target thread
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].data1 = block1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].db1=0;

  update_lru1(core, tt1);
  update_lru2(core, ht2);

  update_mru1(core, tt1);
  update_mru2(core, ht2);
}

if (type == 'r' && wp==2) // write-through
{
  //copy from level-2 hit thread to l-1 target thread
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].data1 = block1;
```

```
  L1[core][tt1][index1].v1=1;

  update_lru1(core, tt1);
  update_lru2(core, ht2);

  update_mru1(core, tt1);
  update_mru2(core, ht2);
}

if (type == 'w' && wp==1) //write-back
{
  //copy from l-2 hit thread to l-1 target thread
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;

  // copy data to cache from processor
  L1[core][tt1][index1].data1 = block1;
  L1[core][tt1][index1].db1=1;

  //update l2
  L2[core][ht2][index2].t2=tag2;
  L2[core][ht2][index2].db2=1;

  update_lru1(core, tt1);
  update_lru2(core, ht2);

  update_mru1(core, tt1);
  update_mru2(core, ht2);
}

if (type == 'w' && wp==2) // write-through
{
  //update l1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].data1 = block1;

  //update l2
  L2[core][ht2][index2].t2=tag2;

  //log
  cash0log.wmem++;

  update_lru1(core, tt1);
  update_lru2(core, ht2);
```

```
    update_mru1(core, tt1);
    update_mru2(core, ht2);
  }
}

if (out2==0) // miss in level-2
{
  // log
  if (type == 'z')
  {corelog[core].iml2++;}

  if (type == 'r')
  {corelog[core].rml2++;}

  if (type == 'w')
  {corelog[core].wml2++;}

  if (type == 'z' && wp==1) //write-back
  {
    if (L1[core][tt1][index1].db1==1)
    {
      // log
      // update main memory
      cash0log.wmem++;
    }


    //copy data from mmemory to tt1
    L1[core][tt1][index1].t1=tag1;
    L1[core][tt1][index1].data1=block1;
    L1[core][tt1][index1].v1 = 1;
    L1[core][tt1][index1].db1=0;

    // upate l2 tt2
    L2[core][tt2][index2].t2=tag2;
    L2[core][tt2][index2].v2=1;
    L2[core][tt2][index2].db2=0;

    // log
    cash0log.rmem++;

    update_lru1(core, tt1);
    update_lru2(core, tt2);

    update_mru1(core, tt1);
    update_mru2(core, tt2);
```

```
}

if (type == 'z' && wp==2) // write-through / no allocate
{
  // copy from tt2 to tt1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].data1=block1;
  L1[core][tt1][index1].v1 = 1;

  // update l2
  L2[core][tt2][index2].t2=tag2;
  L2[core][tt2][index2].v2=1;

  // log
  cash0log.rmem++;

  update_lru1(core, tt1);
  update_lru2(core, tt2);

  update_mru1(core, tt1);
  update_mru2(core, tt2);
}

if (type == 'r' && wp==1) //write-back
{
  if (L1[core][tt1][index1].db1==1)
  {
    // log
    // update main memory
    cash0log.wmem++;
  }


  //copy data from mmemory to tt1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].data1=block1;
  L1[core][tt1][index1].v1 = 1;
  L1[core][tt1][index1].db1=0;

  // upate l2 tt2
  L2[core][tt2][index2].t2=tag2;
  L2[core][tt2][index2].v2=1;
  L2[core][tt2][index2].db2=0;

  // log
  cash0log.rmem++;
```

```
  update_lru1(core, tt1);
  update_lru2(core, tt2);

  update_mru1(core, tt1);
  update_mru2(core, tt2);
}

if (type == 'r' && wp==2) // write-through / no allocate
{
   // copy from tt2 to tt1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].data1=block1;
  L1[core][tt1][index1].v1 = 1;

  // update l2
  L2[core][tt2][index2].t2=tag2;
  L2[core][tt2][index2].v2=1;

  // log
  cash0log.rmem++;

  update_lru1(core, tt1);
  update_lru2(core, tt2);

  update_mru1(core, tt1);
  update_mru2(core, tt2);
}

if (type == 'w' && wp==1)
{
  if (L1[core][tt1][index1].db1==1)
  {
      // log
      // update main memory
      cash0log.wmem++;
  }

  // after write operation to tt1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1 = 1;
  L1[core][tt1][index1].data1=block1;
  L1[core][tt1][index1].db1=1;

  // update l2
  L2[core][tt2][index2].t2=tag2;
```

```
      L2[core][tt2][index2].v2=1;
      L2[core][tt2][index2].db2=1;

      update_lru1(core, tt1);
      update_lru2(core, tt2);

      update_mru1(core, tt1);
      update_mru2(core, tt2);
    }

    if (type == 'w' && wp==2)
    {
      //log
      cash0log.wmem++;
    }
  }
}

void level1_hitmiss_mc()
{
  int a=2, n=1<<asc1;

  for (int i=0;i<n;i++)
  {
    if (L1[core][i][index1].v1==1)
    {
      if (L1[core][i][index1].t1==tag1)
      {
        a=1;
        out1=1;
        ht1=i;

        // log
        if (type=='z')
        {corelog[core].ihl1++;}

        if (type=='r')
        {corelog[core].rhl1++;}

        if (type=='w')
        {corelog[core].whl1++;}

        break;
      }
    }
  }
```

```
  if (a==2)
  {
    out1=0;

    // log
    if (type=='z')
    {corelog[core].iml1++;}

    if (type=='r')
    {corelog[core].rml1++;}

    if (type=='w')
    {corelog[core].wml1++;}
  }
}

void only_l1_mesi()
{
  if (out1==1) //in case of hit in level-1 private core
  {
    if (type =='z')
    {
      // log
      corelog[core].ihp1++;

      if (L1[core][ht1][index1].stat1=='m')
      {corelog[core].mih1++;}

      if (L1[core][ht1][index1].stat1=='e')
      {corelog[core].eih1++;}

      if (L1[core][ht1][index1].stat1=='s')
      {corelog[core].sih1++;}
    }

    if (type =='r')
    {
      // log
      corelog[core].rhp1++;

      if (L1[core][ht1][index1].stat1=='m')
      {corelog[core].mrh1++;}

      if (L1[core][ht1][index1].stat1=='e')
      {corelog[core].erh1++;}
```

```
  if (L1[core][ht1][index1].stat1=='s')
  {corelog[core].srh1++;}
}

if (type == 'w')
{
  // log
  corelog[core].whp1++;

  if (L1[core][ht1][index1].stat1=='m')
  {corelog[core].mwh1++;}

  if (L1[core][ht1][index1].stat1=='e')
  {corelog[core].ewh1++;}

  if (L1[core][ht1][index1].stat1=='s')
  {corelog[core].swh1++;}

  // invalidate other copies
  if (L1[core][ht1][index1].stat1=='s')
  {
    //send invalidation signal in l1
    int totalmatch1 = 0;
    totalmatch1=searchcores1(core);

    for(int i=0; i<totalmatch1; i++)
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='i';
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].v1=0;
      check_lru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
      check_mru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
    }

    //invalidate victim cache
    for(int i = 0; i < coreno; i++)
    {
      for(int j = 0; j < victimLines; j++)
      {
        if(victimCache[i][j].vValid == 1)
        {
          if(victimCache[i][j].victimTag == tag1)
          {
            victimCache[i][j].vState = 'i';
            victimCache[i][j].vValid = 0;
          }
```

```
        }
      }
    }

    //log
    if(totalmatch1!=0)
    {
      corelog[core].bu++; //update bus
    }
  }

  //write to private l1 cache
  L1[core][ht1][index1].stat1='m';
 }

 update_lru1(core, ht1);
 update_mru1(core, ht1);
}

if (out1==0) //miss in l1
{
 //check victim cache
 if(L1[core][tt1][index1].v1 == 1 && victimLines > 0)
 {
  vCache();
 }


 //replacement policy
 if(parametersL1.back() == 49)
  tt1 = highest_lru1();
 else if(parametersL1.back() == 50)
  tt1 = randomReplacementL1();
 else if(parametersL1.back() == 51)
  tt1 = queueReplacementL1();
 else if(parametersL1.back() == 52)
  tt1 = highest_mru1();

 int totalmatch1 = 0;
 totalmatch1=searchcores1(core);

 if(totalmatch1!=0) //hit in bus
 {
  // log
  corelog[core].ctoc++;
```

```
if (type=='z')
{
 //log
 corelog[core].ihb++;

 //copy l2 to l1
 L1[core][tt1][index1].t1=tag1;
 L1[core][tt1][index1].v1=1;
 L1[core][tt1][index1].stat1='s';

 //change state to shared of copies in bus
 int totalmatch1=0;
 totalmatch1=searchcores1(core);
 for (int i=0; i<totalmatch1; i++)
 {
  if (matchcore1[i].mstat1!='s')
  {
   L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
  }
 }

 update_lru1(core, tt1);
 update_mru1(core, tt1);
}

if (type=='r')
{
 //log
 corelog[core].rhb++;

 // copy l2 to l1
 L1[core][tt1][index1].t1=tag1;
 L1[core][tt1][index1].v1=1;
 L1[core][tt1][index1].stat1='s';

 // change state to shared of copies in bus
 int totalmatch1=0;
 totalmatch1=searchcores1(core);
 for (int i=0; i<totalmatch1; i++)
 {
  if (matchcore1[i].mstat1!='s')
  {
   L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
  }
 }
```

```
    update_lru1(core, tt1);
    update_mru1(core, tt1);
  }

  if (type=='w')
  {
    //log
    corelog[core].whb++;

    // copy from bus to l-1
    L1[core][tt1][index1].t1=tag1;
    L1[core][tt1][index1].v1=1;

    // write to l1
    L1[core][tt1][index1].stat1='m';

    //invalidate copies in bus
    int totalmatch1=0;
    totalmatch1=searchcores1(core);
    for(int i=0; i<totalmatch1;i++)
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='i';
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].v1=0;
      check_lru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
    }

    update_lru1(core, tt1);
    update_mru1(core, tt1);
  }
}

if(totalmatch1==0) //miss in bus
{
  if (type == 'z')
  {
    // log
    corelog[core].imb++;

    //copy from memory to tt1
    L1[core][tt1][index1].t1=tag1;
    L1[core][tt1][index1].v1=1;
    L1[core][tt1][index1].stat1='s';

    update_lru1(core, tt1);
    update_mru1(core, tt1);
  }
```

```
      if (type == 'r')
      {
        // log
        corelog[core].rmb++;

        //copy from memory to tt1
        L1[core][tt1][index1].t1=tag1;
        L1[core][tt1][index1].v1=1;
        L1[core][tt1][index1].stat1='s';

        update_lru1(core, tt1);
        update_mru1(core, tt1);
      }

      if (type == 'w')
      {
        // log
        corelog[core].wmb++;

        //copy from memory to tt1
        L1[core][tt1][index1].t1=tag1;
        L1[core][tt1][index1].v1=1;

        // write operation in tt1
        L1[core][tt1][index1].stat1='m';

        update_lru1(core, tt1);
        update_mru1(core, tt1);
      }
    }
  }
}

void level1_mesi()
{
  if (out1==1) //in case of hit in level-1 private core
  {
    //find copy in l2
    th2=find_l2(core);

    if (type =='z')
    {
      // log
      corelog[core].ihp1++;
```

```
  if (L1[core][ht1][index1].stat1=='m')
  {corelog[core].mih1++;}

  if (L1[core][ht1][index1].stat1=='e')
  {corelog[core].eih1++;}

  if (L1[core][ht1][index1].stat1=='s')
  {corelog[core].sih1++;}
}

if (type =='r')
{
  // log
  corelog[core].rhp1++;

  if (L1[core][ht1][index1].stat1=='m')
  {corelog[core].mrh1++;}

  if (L1[core][ht1][index1].stat1=='e')
  {corelog[core].erh1++;}

  if (L1[core][ht1][index1].stat1=='s')
  {corelog[core].srh1++;}
}

if (type == 'w')
{
  // log
  corelog[core].whp1++;

  if (L1[core][ht1][index1].stat1=='m')
  {corelog[core].mwh1++;}

  if (L1[core][ht1][index1].stat1=='e')
  {corelog[core].ewh1++;}

  if (L1[core][ht1][index1].stat1=='s')
  {corelog[core].swh1++;}

  // invalidate other copies
  if (L1[core][ht1][index1].stat1=='s')
  {
    //send invalidation signal in l1
    int totalmatch1 = 0;
    totalmatch1=searchcores1(core);
```

```
  for(int i=0; i<totalmatch1; i++)
  {
   L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='i';
   L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].v1=0;
   check_lru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
   check_mru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
  }

  //send invalidation signal in l2
  int totalmatch2 = 0;
  totalmatch2=searchcores2(core);
  for(int i=0; i<totalmatch2; i++)
  {
   L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='i';
   L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].v2=0;
   check_lru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
   check_mru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
  }

  //invalidate victim cache
  for(int i = 0; i < coreno; i++)
  {
   for(int j = 0; j < victimLines; j++)
   {
    if(victimCache[i][j].vValid == 1)
    {
     if(victimCache[i][j].victimTag == tag1)
     {
      victimCache[i][j].vState = 'i';
      victimCache[i][j].vValid = 0;
     }
    }
   }
  }

  //log
  if(totalmatch2!=0)
  {
   corelog[core].bu++; //update bus
  }
 }

 //write to private l1 cache
 L1[core][ht1][index1].stat1='m';

 //update private l2 cache
```

```
    L2[core][th2][index2].stat2='m';
   }

   update_lru1(core, ht1);
   update_lru2(core, th2);

   update_mru1(core, ht1);
   update_mru2(core, th2);
  }

 if (out1==0) // when miss in level-1 private core
 {
   //check victim cache
   if(L1[core][tt1][index1].v1 == 1 && victimLines > 0)
   {
     //swap
     vCache();
   }

   //replacement policy
   if(parametersL1.back() == 49)
     tt1 = highest_lru1();
   else if(parametersL1.back() == 50)
     tt1 = randomReplacementL1();
   else if(parametersL1.back() == 51)
     tt1 = queueReplacementL1();
   else if(parametersL1.back() == 52)
     tt1 = highest_mru1();


   // check hit or miss in level-2
   b = 2, m = 1 << asc2;
   for (int i = 0; i < m; i++)
   {
     if (L2[core][i][index2].v2 == 1)
     {
       if (L2[core][i][index2].t2 == tag2)
       {
         ht2 = i;
         b = 1;
         out2 = 1; // in case of hit in level-2

         if (type=='z')
         {corelog[core].ihl2++;} //log

         if (type=='r')
```

```
        {corelog[core].rhl2++;} //log

        if (type=='w')
        {corelog[core].whl2++;} //log

        break;
      }
    }
  }

  if (b == 2) // in case of miss in level-2
  {
    out2 = 0;

    if (type=='z')
    {corelog[core].iml2++;} //log

    if (type=='r')
    {corelog[core].rml2++;} //log

    if (type=='w')
    {corelog[core].wml2++;} //log
  }
 }
}

void level2_mesi()
{
 if (out2 == 1) // hit in level-2
 {
  if (type=='z')
  {
    corelog[core].ihp2++; //log

    if (L2[core][ht2][index2].stat2=='m')
    {corelog[core].mih2++;}

    if (L2[core][ht2][index2].stat2=='e')
    {corelog[core].eih2++;}

    if (L2[core][ht2][index2].stat2=='s')
    {corelog[core].sih2++;}

    // copy data from l2 to l1
    L1[core][tt1][index1].t1=tag1;
    L1[core][tt1][index1].v1=1;
```

```
  L1[core][tt1][index1].stat1 = L2[core][ht2][index2].stat2;
  //L1[core][tt1][index1].data1=block1;

  update_lru1(core, tt1);
  update_lru2(core, ht2);

  update_mru1(core, tt1);
  update_mru2(core, ht2);
}

if (type=='r')
{
  // log
  corelog[core].rhp2++;

  if (L2[core][ht2][index2].stat2=='m')
   {corelog[core].mrh2++;}

  if (L2[core][ht2][index2].stat2=='e')
   {corelog[core].erh2++;}

  if (L2[core][ht2][index2].stat2=='s')
   {corelog[core].srh2++;}

  // copy data from l2 to l1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1 = L2[core][ht2][index2].stat2;
  //L1[core][tt1][index1].data1=block1;

  update_lru1(core, tt1);
  update_lru2(core, ht2);

  update_mru1(core, tt1);
  update_mru2(core, ht2);
}

if (type=='w')
{

  // log
  corelog[core].whp2++;

  if (L2[core][ht2][index2].stat2=='m')
   {corelog[core].mwh2++;}
```

```
if (L2[core][ht2][index2].stat2=='e')
{corelog[core].ewh2++;}

if (L2[core][ht2][index2].stat2=='s')
{corelog[core].swh2++;}

if( L2[core][ht2][index2].stat2!='s')
{
 //invalidate other copies
 //invalid copies in bus
 int totalmatch1 = 0;
 totalmatch1=searchcores1(core);
 for(int i=0; i<totalmatch1;i++)
 {
  L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='i';
  L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].v1=0;
  check_lru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
  check_mru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
 }

 //invalid copies in bus
 int totalmatch2 = 0;
 totalmatch2=searchcores2(core);
 for(int i=0; i<totalmatch2;i++)
 {
  L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='i';
  L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].v2=0;
  check_lru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
  check_mru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
 }

 //invalidate victim cache
 for(int i = 0; i < coreno; i++)
 {
  for(int j = 0; j < victimLines; j++)
  {
   if(victimCache[i][j].vValid == 1)
   {
    if(victimCache[i][j].victimTag == tag2)
    {
     victimCache[i][j].vState = 'i';
     victimCache[i][j].vValid = 0;
    }
   }
  }
 }
```

```
      //log
      if(totalmatch1!=0)
      {
        corelog[core].bu++; //update bus
      }
    }

    // copy data from l2 to l1
    L1[core][tt1][index1].t1=tag1;
    L1[core][tt1][index1].v1=1;

    //write operation in tt1
    L1[core][tt1][index1].stat1='m';

    // update l2
    L2[core][ht2][index2].stat2='m';

    update_lru1(core, tt1);
    update_lru2(core, ht2);

    update_mru1(core, tt1);
    update_mru2(core, ht2);
  }
}

if (out2 == 0) // miss in level-2
{
  //replacement policy
  if(parametersL2.back() == 49)
    tt2 = highest_lru2();
  else if(parametersL2.back() == 50)
    tt2 = randomReplacementL2();
  else if(parametersL2.back() == 51)
    tt2 = queueReplacementL2();
  else if(parametersL2.back() == 52)
    tt2 = highest_mru2();


  // find hit or miss in bus
  int totalmatch2 = 0;
  totalmatch2=searchcores2(core);

  if(totalmatch2!=0) //hit in bus
  {
    // log
```

```
corelog[core].ctoc++;

if (type=='z')
{
  //log
  corelog[core].ihb++;

  if (L2[core][tt2][index2].stat2=='m')
  {
    // log
    cash0log.wmem++; // write to memory
  }

  //copy from bus to target l2
  L2[core][tt2][index2].t2=tag2;
  L2[core][tt2][index2].v2=1;
  L2[core][tt2][index2].stat2='s';

  //copy l2 to l1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='s';

  //change state to shared of copies in bus
  int totalmatch1=0;
  totalmatch1=searchcores1(core);
  for (int i=0; i<totalmatch1; i++)
  {
    if (matchcore1[i].mstat1!='s')
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
    }
  }

  // change state to shared of copies in bus
  int totalmatch2=0;
  totalmatch2=searchcores2(core);
  for (int i=0; i<totalmatch2; i++)
  {
    if (matchcore2[i].mstat2!='s')
    {
      L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='s';
    }
  }

  update_lru1(core, tt1);
```

```
  update_lru2(core, tt2);

  update_mru1(core, tt1);
  update_mru2(core, tt2);
}

if (type=='r')
{
  //log
  corelog[core].rhb++;

  if (L2[core][tt2][index2].stat2=='m')
  {
    // log
    cash0log.wmem++; // write to memory

    L2[core][tt2][index2].stat2='e';
  }

  //copy from bus hit thread to target l2
  L2[core][tt2][index2].t2=tag2;
  L2[core][tt2][index2].v2=1;
  L2[core][tt2][index2].stat2='s';

  // copy l2 to l1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='s';

  // change state to shared of copies in bus
  int totalmatch1=0;
  totalmatch1=searchcores1(core);
  for (int i=0; i<totalmatch1; i++)
  {
    if (matchcore1[i].mstat1!='s')
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
    }
  }

  // state change in other cores l2
  int totalmatch2=0;
  totalmatch2=searchcores2(core);
  for (int i=0; i<totalmatch2; i++)
  {
    if (matchcore2[i].mstat2!='s')
```

```
    {
      L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='s';
    }
  }

  update_lru1(core, tt1);
  update_lru2(core, tt2);

  update_mru1(core, tt1);
  update_mru2(core, tt2);
}

if (type=='w')
{
  //log
  corelog[core].whb++;

  if (L2[core][tt2][index2].stat2=='m')
  {
    // log
    cash0log.wmem++; // write to memory

    L2[core][tt2][index2].stat2='e';
  }

  //write operation to target l1 and l2 cache
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='m';

  L2[core][tt2][index2].t2=tag2;
  L2[core][tt2][index2].v2=1;
  L2[core][tt2][index2].stat2='m';

  //invalid copies in bus
  int totalmatch1=0;
  totalmatch1=searchcores1(core);
  for(int i=0; i<totalmatch1;i++)
  {
    L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='i';
    L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].v1=0;
    check_lru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
    check_mru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
  }

  int totalmatch2=0;
```

```
    totalmatch2=searchcores2(core);
    for(int i=0; i<totalmatch2;i++)
    {
      L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='i';
      L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].v2=0;
      check_lru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
      check_mru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
    }

    update_lru1(core, tt1);
    update_lru2(core, tt2);

    update_mru1(core, tt1);
    update_mru2(core, tt2);
  }
}

if(totalmatch2==0) //miss in bus
{
  // log
  cash0log.busmiss++;

  if (type == 'z')
  {
    // log
    corelog[core].imb++;

    if (L2[core][tt2][index2].stat2=='m')
    {
      // log
      cash0log.wmem++; // write to memory
    }

    // copy from memory to l1 and l2 cache
    L2[core][tt2][index2].t2=tag2;
    L2[core][tt2][index2].v2=1;
    L2[core][tt2][index2].stat2='e';

    L1[core][tt1][index1].t1=tag1;
    L1[core][tt1][index1].v1=1;
    L1[core][tt1][index1].stat1='e';

    update_lru1(core, tt1);
    update_lru2(core, tt2);

    update_mru1(core, tt1);
```

```
  update_mru2(core, tt2);
}

if (type == 'r')
{
  // log
  corelog[core].rmb++;

  if (L2[core][tt2][index2].stat2=='m')
  {
    // log
    cash0log.wmem++; // write to memory
  }

  // copy from memory to l1 and l2 cache
  L2[core][tt2][index2].t2=tag2;
  L2[core][tt2][index2].v2=1;
  L2[core][tt2][index2].stat2='e';

  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='e';

  update_lru1(core, tt1);
  update_lru2(core, tt2);

  update_mru1(core, tt1);
  update_mru2(core, tt2);
}

if (type == 'w')
{
  // log
  corelog[core].wmb++;

  if (L2[core][tt2][index2].stat2=='m')
  {
    // log
    cash0log.wmem++; // write to memory
  }

  // copy from memory to l1 and l2 cache
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='m';
```

```
        L2[core][tt2][index2].t2=tag2;
        L2[core][tt2][index2].v2=1;
        L2[core][tt2][index2].stat2='m';

        update_lru1(core, tt1);
        update_lru2(core, tt2);

        update_mru1(core, tt1);
        update_mru2(core, tt2);
      }
    }
  }
}

// msi protocol
void only_l1_msi()
{
  if (out1==1) //in case of hit in level-1 private core
  {
    if (type =='z')
    {
      // log
      corelog[core].ihp1++;

      if (L1[core][ht1][index1].stat1=='m')
      {corelog[core].mih1++;}

      if (L1[core][ht1][index1].stat1=='s')
      {corelog[core].sih1++;}
    }

    if (type =='r')
    {
      // log
      corelog[core].rhp1++;

      if (L1[core][ht1][index1].stat1=='m')
      {corelog[core].mrh1++;}

      if (L1[core][ht1][index1].stat1=='s')
      {corelog[core].srh1++;}
    }

    if (type == 'w')
    {
      // log
```

```
corelog[core].whp1++;

if (L1[core][ht1][index1].stat1=='m')
{corelog[core].mwh1++;}

if (L1[core][ht1][index1].stat1=='s')
{corelog[core].swh1++;}

// invalidate other copies
if (L1[core][ht1][index1].stat1=='s')
{
  //send invalidation signal in l1
  int totalmatch1 = 0;
  totalmatch1=searchcores1(core);

  for(int i=0; i<totalmatch1; i++)
  {
    L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='i';
    L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].v1=0;
    check_lru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
    check_mru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
  }

  //invalidate victim cache
  for(int i = 0; i < coreno; i++)
  {
    for(int j = 0; j < victimLines; j++)
    {
      if(victimCache[i][j].vValid == 1)
      {
        if(victimCache[i][j].victimTag == tag1)
        {
          victimCache[i][j].vState = 'i';
          victimCache[i][j].vValid = 0;
        }
      }
    }
  }

  //log
  if(totalmatch1!=0)
  {
    corelog[core].bu++; //update bus
  }
}
```

```
    //write to private l1 cache
    L1[core][ht1][index1].stat1='m';
  }

  update_lru1(core, ht1);
  update_mru1(core, ht1);

}

if (out1==0) //miss in l1
{
  //check victim cache
  if(L1[core][tt1][index1].v1 == 1 && victimLines > 0)
    {
      //swap
      vCache();
    }

  //replacement policy
  if(parametersL1.back() == 49)
    tt1 = highest_lru1();
  else if(parametersL1.back() == 50)
    tt1 = randomReplacementL1();
  else if(parametersL1.back() == 51)
    tt1 = queueReplacementL1();
  else if(parametersL1.back() == 52)
    tt1 = highest_mru1();

  // find hit or miss in bus
  int totalmatch1 = 0;
  totalmatch1=searchcores1(core);

  if(totalmatch1!=0) //hit in bus
  {
    // log
    corelog[core].ctoc++;

    if (type=='z')
    {
      //log
      corelog[core].ihb++;

      //copy l2 to l1
      L1[core][tt1][index1].t1=tag1;
      L1[core][tt1][index1].v1=1;
      L1[core][tt1][index1].stat1='s';
```

```
  //change state to shared of copies in bus
  int totalmatch1=0;
  totalmatch1=searchcores1(core);
  for (int i=0; i<totalmatch1; i++)
  {
    if (matchcore1[i].mstat1!='s')
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
    }
  }

  update_lru1(core, tt1);
  update_mru1(core, tt1);
}

if (type=='r')
{
  //log
  corelog[core].rhb++;

  // copy l2 to l1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='s';

  // change state to shared of copies in bus
  int totalmatch1=0;
  totalmatch1=searchcores1(core);
  for (int i=0; i<totalmatch1; i++)
  {
    if (matchcore1[i].mstat1!='s')
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
    }
  }

  update_lru1(core, tt1);
  update_mru1(core, tt1);
}

if (type=='w')
{
  //log
  corelog[core].whb++;
```

```
    // copy from bus to l-1
    L1[core][tt1][index1].t1=tag1;
    L1[core][tt1][index1].v1=1;

    // write to l1
    L1[core][tt1][index1].stat1='m';

    //invalidate copies in bus
    int totalmatch1=0;
    totalmatch1=searchcores1(core);
    for(int i=0; i<totalmatch1;i++)
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='i';
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].v1=0;
      check_lru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
      check_mru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
    }

    update_lru1(core, tt1);
    update_mru1(core, tt1);
  }
}

if(totalmatch1==0) //miss in bus
{
  // log
  cash0log.busmiss++;

  if (type == 'z')
  {
    // log
    corelog[core].imb++;

    //copy from memory to tt1
    L1[core][tt1][index1].t1=tag1;
    L1[core][tt1][index1].v1=1;
    L1[core][tt1][index1].stat1='s';

    update_lru1(core, tt1);
    update_mru1(core, tt1);
  }

  if (type == 'r')
  {
    // log
    corelog[core].rmb++;
```

```
      //copy from memory to tt1
      L1[core][tt1][index1].t1=tag1;
      L1[core][tt1][index1].v1=1;
      L1[core][tt1][index1].stat1='s';

      update_lru1(core, tt1);
      update_mru1(core, tt1);
     }

     if (type == 'w')
     {
      // log
      corelog[core].wmb++;

      //copy from memory to tt1
      L1[core][tt1][index1].t1=tag1;
      L1[core][tt1][index1].v1=1;

      // write operation in tt1
      L1[core][tt1][index1].stat1='m';

      update_lru1(core, tt1);
      update_mru1(core, tt1);
     }
    }
  }
}

void level1_msi()
{
  if (out1==1) //in case of hit in level-1 private core
  {
   //find copy in l2
   th2=find_l2(core);

   if (type =='z')
   {
    // log
    corelog[core].ihp1++;

    if (L1[core][ht1][index1].stat1=='m')
    {corelog[core].mih1++;}

    if (L1[core][ht1][index1].stat1=='s')
    {corelog[core].sih1++;}
```

```
}

if (type =='r')
{
 // log
 corelog[core].rhp1++;

 if (L1[core][ht1][index1].stat1=='m')
 {corelog[core].mrh1++;}

 if (L1[core][ht1][index1].stat1=='s')
 {corelog[core].srh1++;}
}

if (type == 'w')
{
 // log
 corelog[core].whp1++;

 if (L1[core][ht1][index1].stat1=='m')
 {corelog[core].mwh1++;}

 if (L1[core][ht1][index1].stat1=='s')
 {corelog[core].swh1++;}

 // invalidate other copies
 if (L1[core][ht1][index1].stat1=='s')
 {
  //send invalidation signal in l1
  int totalmatch1 = 0;
  totalmatch1=searchcores1(core);

  for(int i=0; i<totalmatch1; i++)
  {
   L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='i';
   L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].v1=0;
   check_lru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
   check_mru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
  }

  //send invalidation signal in l2
  int totalmatch2 = 0;
  totalmatch2=searchcores2(core);
  for(int i=0; i<totalmatch2; i++)
  {
   L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='i';
```

```
        L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].v2=0;
        check_lru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
        check_mru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
      }

      //invalidate victim cache
      for(int i = 0; i < coreno; i++)
      {
        for(int j = 0; j < victimLines; j++)
        {
          if(victimCache[i][j].vValid == 1)
          {
            if(victimCache[i][j].victimTag == tag1)
            {
              victimCache[i][j].vState = 'i';
              victimCache[i][j].vValid = 0;
            }
          }
        }
      }

      //log
      if(totalmatch1!=0)
      {
        corelog[core].bu++; //update bus
      }
    }

    //write to private l1 cache
    L1[core][ht1][index1].stat1='m';

    //update private l2 cache
    L2[core][th2][index2].stat2='m';
  }

  update_lru1(core, ht1);
  update_lru2(core, th2);

  update_mru1(core, ht1);
  update_mru2(core, th2);
}

if (out1==0) // when miss in level-1 private core
{
  //check victim cache
  if(L1[core][tt1][index1].v1 == 1 && victimLines > 0)
```

```
  {
    //swap
    vCache();
  }

//replacement policy
if(parametersL1.back() == 49)
  tt1 = highest_lru1();
else if(parametersL1.back() == 50)
  tt1 = randomReplacementL1();
else if(parametersL1.back() == 51)
  tt1 = queueReplacementL1();
else if(parametersL1.back() == 52)
  tt1 = highest_mru1();


// check hit or miss in level-2
b = 2, m = 1 << asc2;
for (int i = 0; i < m; i++)
{
  if (L2[core][i][index2].v2 == 1)
  {
    if (L2[core][i][index2].t2 == tag2)
    {
      ht2 = i;
      b = 1;
      out2 = 1; // in case of hit in level-2

      // log
      if (type=='z')
      {corelog[core].ihl2++;}

      if (type=='r')
      {corelog[core].rhl2++;}

      if (type=='w')
      {corelog[core].whl2++;}

      break;
    }
  }
}

if (b == 2) // in case of miss in level-2
{
  out2 = 0;
```

```
      //log
      if (type=='z')
      {corelog[core].iml2++;}

      if (type=='r')
      {corelog[core].rml2++;}

      if (type=='w')
      {corelog[core].wml2++;}
    }
  }
}

void level2_msi()
{
  if (out2 == 1) // hit in level-2
  {
    // log
    corelog[core].exchange++;

    if (type=='z')
    {
      // log
      corelog[core].ihp2++;

      if (L2[core][ht2][index2].stat2=='m')
      {corelog[core].mih2++;}

      if (L2[core][ht2][index2].stat2=='s')
      {corelog[core].sih2++;}

      // copy data from l2 to l1
      L1[core][tt1][index1].t1=tag1;
      L1[core][tt1][index1].v1=1;
      L1[core][tt1][index1].stat1 = L2[core][ht2][index2].stat2;

      update_lru1(core, tt1);
      update_lru2(core, ht2);

      update_mru1(core, tt1);
      update_mru2(core, ht2);
    }

    if (type=='r')
    {
```

```
  // log
  corelog[core].rhp2++;

  if (L2[core][ht2][index2].stat2=='m')
  {corelog[core].mrh2++;}

  if (L2[core][ht2][index2].stat2=='s')
  {corelog[core].srh2++;}

  // copy data from l2 to l1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1 = L2[core][ht2][index2].stat2;

  update_lru1(core, tt1);
  update_lru2(core, ht2);

  update_mru1(core, tt1);
  update_mru2(core, ht2);
}

if (type=='w')
{

  // log
  corelog[core].whp2++;

  if (L2[core][ht2][index2].stat2=='m')
  {corelog[core].mwh2++;}

  if (L2[core][ht2][index2].stat2=='s')
  {corelog[core].swh2++;}

  if( L2[core][ht2][index2].stat2!='s')
  {
    //invalidate other copies
    //invalid copies in bus
    int totalmatch1 = 0;
    totalmatch1=searchcores1(core);
    for(int i=0; i<totalmatch1;i++)
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='i';
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].v1=0;
      check_lru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
      check_mru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
    }
```

```
  //invalid copies in bus
  int totalmatch2 = 0;
  totalmatch2=searchcores2(core);
  for(int i=0; i<totalmatch2;i++)
  {
    L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='i';
    L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].v2=0;
    check_lru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
    check_mru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
  }

  //invalidate victim cache
  for(int i = 0; i < coreno; i++)
  {
    for(int j = 0; j < victimLines; j++)
    {
      if(victimCache[i][j].vValid == 1)
      {
        if(victimCache[i][j].victimTag == tag2)
        {
          victimCache[i][j].vState = 'i';
          victimCache[i][j].vValid = 0;
        }
      }
    }
  }

  //log
  if(totalmatch2!=0)
  {
    corelog[core].bu++; //update bus
  }
}

// copy data from l2 to l1
L1[core][tt1][index1].t1=tag1;
L1[core][tt1][index1].v1=1;

//write operation in tt1
L1[core][tt1][index1].stat1='m';

// update l2
L2[core][ht2][index2].stat2='m';

update_lru1(core, tt1);
```

```
    update_lru2(core, ht2);

    update_mru1(core, tt1);
    update_mru2(core, ht2);
  }
}

if (out2 == 0) // miss in level-2
{
  //replacement policy
  if(parametersL2.back() == 49)
    tt2 = highest_lru2();
  else if(parametersL2.back() == 50)
    tt2 = randomReplacementL2();
  else if(parametersL2.back() == 51)
    tt2 = queueReplacementL2();
  else if(parametersL2.back() == 52)
    tt2 = highest_mru2();

  // find hit or miss in bus
  int totalmatch2 = 0;
  totalmatch2=searchcores2(core);

  if(totalmatch2!=0) //hit in bus
  {
    // log
    corelog[core].ctoc++;

    if (type=='z')
    {
      //log
      corelog[core].ihb++;

      if (L2[core][tt2][index2].stat2=='m')
      {
        // log
        cash0log.wmem++; // write to memory
      }

      //copy from bus to target l2
      L2[core][tt2][index2].t2=tag2;
      L2[core][tt2][index2].v2=1;
      L2[core][tt2][index2].stat2='s';

      //copy bus to l1
      L1[core][tt1][index1].t1=tag1;
```

```
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='s';

  //change state to shared of copies in bus
  int totalmatch1=0;
  totalmatch1=searchcores1(core);
  for (int i=0; i<totalmatch1; i++)
  {
    if (matchcore1[i].mstat1!='s')
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
    }
  }

  // change state to shared of copies in bus
  int totalmatch2=0;
  totalmatch2=searchcores2(core);
  for (int i=0; i<totalmatch2; i++)
  {
    if (matchcore2[i].mstat2!='s')
    {
      L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='s';
    }
  }

  update_lru1(core, tt1);
  update_lru2(core, tt2);

  update_mru1(core, tt1);
  update_mru2(core, tt2);
}

if (type=='r')
{
  //log
  corelog[core].rhb++;

  if (L2[core][tt2][index2].stat2=='m')
  {
    // log
    cash0log.wmem++; // write to memory
  }

  //copy from bus hit thread to target l2
  L2[core][tt2][index2].t2=tag2;
  L2[core][tt2][index2].v2=1;
```

```
  L2[core][tt2][index2].stat2='s';

  // copy l2 to l1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='s';

  // change state to shared of copies in bus
  int totalmatch1=0;
  totalmatch1=searchcores1(core);
  for (int i=0; i<totalmatch1; i++)
  {
    if (matchcore1[i].mstat1!='s')
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
    }
  }

  // state change in other cores l2
  int totalmatch2=0;
  totalmatch2=searchcores2(core);
  for (int i=0; i<totalmatch2; i++)
  {
    if (matchcore2[i].mstat2!='s')
    {
      L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='s';
    }
  }

  update_lru1(core, tt1);
  update_lru2(core, tt2);

  update_mru1(core, tt1);
  update_mru2(core, tt2);
}

if (type=='w')
{
  //log
  corelog[core].whb++;

  if (L2[core][tt2][index2].stat2=='m')
  {
    // log
    cash0log.wmem++; // write to memory
  }
```

```
    //copy from other hit thread to target l2 and target l1
    L2[core][tt2][index2].t2=tag2;
    L2[core][tt2][index2].v2=1;

    L1[core][tt1][index1].t1=tag1;
    L1[core][tt1][index1].v1=1;

    // write to l1
    L1[core][tt1][index1].stat1='m';

    //update l2
    L2[core][tt2][index2].stat2='m';

    //invalid copies in bus
    int totalmatch1=0;
    totalmatch1=searchcores1(core);
    for(int i=0; i<totalmatch1;i++)
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='i';
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].v1=0;
      check_lru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
      check_mru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
    }

    int totalmatch2=0;
    totalmatch2=searchcores2(core);
    for(int i=0; i<totalmatch2;i++)
    {
      L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='i';
      L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].v2=0;
      check_lru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
      check_mru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
    }

    update_lru1(core, tt1);
    update_lru2(core, tt2);

    update_mru1(core, tt1);
    update_mru2(core, tt2);
  }
}

if(totalmatch2==0) //miss in bus
{
  if (type == 'z')
```

```
{
  // log
  corelog[core].imb++;

  if (L2[core][tt2][index2].stat2=='m')
  {
    // log
    cash0log.wmem++; // write to memory
  }

  // copy from memory to l2
  L2[core][tt2][index2].t2=tag2;
  L2[core][tt2][index2].v2=1;
  L2[core][tt2][index2].stat2='s';

  //copy from tt2 to tt1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='s';

  update_lru1(core, tt1);
  update_lru2(core, tt2);

  update_mru1(core, tt1);
  update_mru2(core, tt2);
}

if (type == 'r')
{
  // log
  corelog[core].rmb++;

  if (L2[core][tt2][index2].stat2=='m')
  {
    // log
    cash0log.wmem++; // write to memory
  }

  // copy from memory to TT2
  L2[core][tt2][index2].t2=tag2;
  L2[core][tt2][index2].v2=1;
  L2[core][tt2][index2].stat2='s';

  //copy from tt2 to tt1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
```

```
      L1[core][tt1][index1].stat1='s';

      update_lru1(core, tt1);
      update_lru2(core, tt2);

      update_mru1(core, tt1);
      update_mru2(core, tt2);
    }

    if (type == 'w')
    {
      // log
      corelog[core].wmb++;

      if (L2[core][tt2][index2].stat2=='m')
      {
        // log
        cash0log.wmem++; // write to memory
      }

      // copy from memory to l2
      L2[core][tt2][index2].t2=tag2;
      L2[core][tt2][index2].v2=1;

      //copy from l2 to l1
      L1[core][tt1][index1].t1=tag1;
      L1[core][tt1][index1].v1=1;

      // write operation in tt1
      L1[core][tt1][index1].stat1='m';

      //update l2
      L2[core][tt2][index2].stat2='m';

      update_lru1(core, tt1);
      update_lru2(core, tt2);

      update_mru1(core, tt1);
      update_mru2(core, tt2);
    }
  }
 }
}

void only_l1_firefly()
{
```

```
if (out1==1) //in case of hit in level-1 private core
{
 if (type =='r')
 {
  // log
  corelog[core].rhp1++;

  if (L1[core][ht1][index1].stat1=='v')
  {corelog[core].vrh1++;}

  if (L1[core][ht1][index1].stat1=='d')
  {corelog[core].drh1++;}

  if (L1[core][ht1][index1].stat1=='s')
  {corelog[core].srh1++;}
 }

 if (type == 'w')
 {
  // log
  corelog[core].whp1++;

  if (L1[core][ht1][index1].stat1=='v')
  {corelog[core].vwh1++;}

  if (L1[core][ht1][index1].stat1=='d')
  {corelog[core].dwh1++;}

  if (L1[core][ht1][index1].stat1=='s')
  {corelog[core].swh1++;}

  // log
  corelog[core].whp1++;

  if (L1[core][ht1][index1].stat1=='v')
  {
   L1[core][ht1][index1].stat1='d';
  }

  else if (L1[core][ht1][index1].stat1=='s')
  {
   int totalmatch1=0;
   totalmatch1=searchcores1(core);

   if (totalmatch1=!0)
   {
```

```
        L1[core][ht1][index1].stat1='s';

        //log
        corelog[core].bu++;
      }

      if (totalmatch1==0)
      {
        L1[core][ht1][index1].stat1='d';
      }
    }
    // change
    L1[core][ht1][index1].data1=block1;
  }
  update_lru1(core,ht1);
  update_mru1(core,ht1);
}

if (out1==0) // when miss in level-1 private core
{
  //check victim cache
  if(L1[core][tt1][index1].v1 == 1 && victimLines > 0)
    {
      //swap
      vCache();
    }

  //replacement policy
  if(parametersL1.back() == 49)
    tt1 = highest_lru1();
  else if(parametersL1.back() == 50)
    tt1 = randomReplacementL1();
  else if(parametersL1.back() == 51)
    tt1 = queueReplacementL1();
  else if(parametersL1.back() == 52)
    tt1 = highest_mru1();

  int totalmatch1=0;

  totalmatch1 = searchcores1(core);

  if (totalmatch1!=0) // hit in level-1 other cores
  {
    // log
    corelog[core].ctoc++;
```

```
if (type =='r')
{
  // log
  corelog[core].rhb++;

  // copy from other hit thread to target thread in l-1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='s';
  L1[core][tt1][index1].data1=block1;

  // change state for other cores to 'shared'
  for (int i=0; i<totalmatch1; i++)
  {
    if (matchcore1[i].mstat1!='s')
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
    }
  }

  //log
  corelog[core].bu++;

  update_lru1(core, tt1);
  update_mru1(core, tt1);
}

if (type =='w')
{
  // log
  corelog[core].whb++;

  // copy from other hit thread to target thread in l-1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;

  // after write operation
  L1[core][tt1][index1].stat1='s';
  L1[core][tt1][index1].data1=block1;

  // invalid other copies
  for(int i=0; i<totalmatch1;i++)
  {
    L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
    L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].v1=1;
    check_lru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
```

```
        check_mru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
      }

      //log
      corelog[core].bu++;

      update_lru1(core, tt1);
      update_mru1(core, tt1);
    }
  }

  if (totalmatch1==0) // miss in level-1 private and other cores
  {
    if (type=='r')
    {
      //log
      corelog[core].rmb++;

      //copy from memory to level-1
      L1[core][tt1][index1].t1=tag1;
      L1[core][tt1][index1].v1=1;
      L1[core][tt1][index1].stat1='v';
      L1[core][tt1][index1].data1=block1;

      update_lru1(core, tt1);
      update_mru1(core, tt1);
    }

    if (type=='w')
    {
      //log
      corelog[core].wmb++;

      //copy from processor to level-1
      L1[core][tt1][index1].t1=tag1;
      L1[core][tt1][index1].v1=1;
      L1[core][tt1][index1].data1=block1;
      L1[core][tt1][index1].stat1='d';

      update_lru1(core, tt1);
      update_mru1(core, tt1);
    }
  }
 }
}
```

```
void level1_firefly()
{
  if (out1==1) //in case of hit in level-1 private core
  {
    if (type =='z')
    {
      // log
      corelog[core].ihp1++;

      if (L1[core][ht1][index1].stat1=='v')
      {corelog[core].vrh1++;}

      if (L1[core][ht1][index1].stat1=='d')
      {corelog[core].drh1++;}

      if (L1[core][ht1][index1].stat1=='s')
      {corelog[core].srh1++;}
    }

    if (type =='r')
    {
      // log
      corelog[core].rhp1++;

      if (L1[core][ht1][index1].stat1=='v')
      {corelog[core].vrh1++;}

      if (L1[core][ht1][index1].stat1=='d')
      {corelog[core].drh1++;}

      if (L1[core][ht1][index1].stat1=='s')
      {corelog[core].srh1++;}
    }

    if (type == 'w')
    {
      // log
      corelog[core].whp1++;

      if (L1[core][ht1][index1].stat1=='v')
      {corelog[core].vwh1++;}

      if (L1[core][ht1][index1].stat1=='d')
      {corelog[core].dwh1++;}

      if (L1[core][ht1][index1].stat1=='s')
```

```
    {corelog[core].swh1++;}

    // log
    corelog[core].whp1++;

    if (L1[core][ht1][index1].stat1=='v')
    {
      L1[core][ht1][index1].stat1='d';
      L2[core][th2][index2].stat2='d';
    }

    else if (L1[core][ht1][index1].stat1=='s')
    {
      int totalmatch1=0;
      totalmatch1=searchcores1(core);

      int totalmatch2 = 0;
      totalmatch2=searchcores2(core);

      if (totalmatch2=!0)
      {
        L1[core][ht1][index1].stat1='s';
        L2[core][th2][index2].stat2='s';

        //log
        corelog[core].bu++;
      }

      if (totalmatch2==0)
      {
        L1[core][ht1][index1].stat1='d';
        L2[core][th2][index2].stat2='d';
      }
    }
  }

  update_lru1(core,ht1);
  update_lru1(core,th2);

  update_mru1(core,ht1);
  update_mru1(core,th2);
}

if (out1==0) // when miss in level-1 private core
{
  //check victim cache
```

```
if(L1[core][tt1][index1].v1 == 1 && victimLines > 0)
{
  //swap
  vCache();
}

//replacement policy
if(parametersL1.back() == 49)
  tt1 = highest_lru1();
else if(parametersL1.back() == 50)
  tt1 = randomReplacementL1();
else if(parametersL1.back() == 51)
  tt1 = queueReplacementL1();
else if(parametersL1.back() == 52)
  tt1 = highest_mru1();


// check hit or miss in level-2
b = 2, m = 1 << asc2;
for (int i = 0; i < m; i++)
{
  if (L2[core][i][index2].v2 == 1)
  {
    if (L2[core][i][index2].t2 == tag2)
    {
      ht2 = i;
      b = 1;
      out2 = 1; // in case of hit in level-2

      // log
      if (type=='z')
      {corelog[core].ihl2++;}

      if (type=='r')
      {corelog[core].rhl2++;}

      if (type=='w')
      {corelog[core].whl2++;}

      break;
    }
  }
}

if (b == 2) // in case of miss in level-2
{
```

```
      out2 = 0;

      //log
      if (type=='z')
      {corelog[core].iml2++;}

      if (type=='r')
      {corelog[core].rml2++;}

      if (type=='w')
      {corelog[core].wml2++;}
    }
  }
}

void level2_firefly()
{
  if (out2 == 1) // hit in level-2
  {
    if (type=='z')
    {
      // log
      corelog[core].ihp2++;

      if (L2[core][ht2][index2].stat2=='d')
      {corelog[core].dih2++;}

      if (L2[core][ht2][index2].stat2=='v')
      {corelog[core].vih2++;}

      if (L2[core][ht2][index2].stat2=='s')
      {corelog[core].sih2++;}

      // copy data from l2 to l1
      L1[core][tt1][index1].t1=tag1;
      L1[core][tt1][index1].v1=1;
      L1[core][tt1][index1].stat1 = L2[core][ht2][index2].stat2;

      update_lru1(core, tt1);
      update_lru2(core, ht2);

      update_mru1(core, tt1);
      update_mru2(core, ht2);
    }

    if (type=='r')
```

```
{
  // log
  corelog[core].rhp2++;

  if (L2[core][ht2][index2].stat2=='d')
   {corelog[core].drh2++;}

  if (L2[core][ht2][index2].stat2=='v')
   {corelog[core].vrh2++;}

  if (L2[core][ht2][index2].stat2=='s')
   {corelog[core].srh2++;}

  // copy data from l2 to l1
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1 = L2[core][ht2][index2].stat2;

  update_lru1(core, tt1);
  update_lru2(core, ht2);

  update_mru1(core, tt1);
  update_mru2(core, ht2);
}

if (type=='w')
{
  // log
  corelog[core].whp2++;

  if (L2[core][ht2][index2].stat2=='m')
   {corelog[core].dwh2++;}

  if (L2[core][ht2][index2].stat2=='v')
   {corelog[core].vwh2++;}

  if (L2[core][ht2][index2].stat2=='s')
   {corelog[core].swh2++;}

  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;

  int totalmatch1=0; totalmatch1=searchcores1(core);
  int totalmatch2=0; totalmatch2=searchcores2(core);

  if (totalmatch2==0)
```

```
    {
      L1[core][tt1][index1].stat1='d';
      L2[core][tt2][index2].stat2='d';
    }

    if (totalmatch2!=0)
    {
      for(int i=0; i<totalmatch1;i++)
      {
        L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
        update_lru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
        update_mru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
      }

      for(int i=0; i<totalmatch2;i++)
      {
        L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='s';
        update_lru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
        update_mru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
      }


      L1[core][tt1][index1].stat1='s';
      L2[core][tt2][index2].stat2='s';
    }

    //log
    if(totalmatch2!=0)
    {
      corelog[core].bu++; //update bus
    }

    update_lru1(core, tt1);
    update_lru2(core, ht2);

    update_mru1(core, tt1);
    update_mru2(core, ht2);
  }
}


if (out2 == 0) // miss in level-2
{
  //replacement policy
  if(parametersL2.back() == 49)
    tt2 = highest_lru2();
```

```
else if(parametersL2.back() == 50)
  tt2 = randomReplacementL2();
else if(parametersL2.back() == 51)
  tt2 = queueReplacementL2();
else if(parametersL2.back() == 52)
  tt2 = highest_mru2();

// find hit or miss in bus
int totalmatch1 = 0; totalmatch1=searchcores1(core);
int totalmatch2 = 0; totalmatch2=searchcores2(core);

if(totalmatch2!=0) //hit in bus
{
  // log
  corelog[core].ctoc++;

  //copy from bus to target l2 and l1 cache
  L2[core][tt2][index2].t2=tag2;
  L2[core][tt2][index2].v2=1;
  L2[core][tt2][index2].stat2='s';

  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='s';

  if (type=='z')
  {
    //log
    corelog[core].ihb++;

    if (L2[core][tt2][index2].stat2=='d')
    {
      // log
      cash0log.wmem++; // write to memory
    }

    // change state to shared of copies in bus
    for (int i=0; i<totalmatch1; i++)
    {
      if (matchcore1[i].mstat1!='s')
      {
        L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
      }
    }

    for (int i=0; i<totalmatch2; i++)
```

```
  {
    if (matchcore2[i].mstat2!='s')
    {
      L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='s';
    }
  }
}

if (type=='r')
{
  //log
  corelog[core].rhb++;

  if (L2[core][tt2][index2].stat2=='d')
  {
    // log
    cash0log.wmem++; // write to memory
  }

  // change state to shared of copies in bus
  for (int i=0; i<totalmatch1; i++)
  {
    if (matchcore1[i].mstat1!='s')
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
    }
  }

  for (int i=0; i<totalmatch2; i++)
  {
    if (matchcore2[i].mstat2!='s')
    {
      L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='s';
    }
  }
}

if (type=='w')
{
  //log
  corelog[core].whb++;
  cash0log.wmem++; // write to memory

  if (L2[core][tt2][index2].stat2=='d')
  {
    // log
```

```
      cash0log.wmem++; // write to memory
    }

    //send shared signal in bus
    for(int i=0; i<totalmatch1;i++)
    {
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].stat1='s';
      L1[matchcore1[i].mcore1][matchcore1[i].mthread1][index1].v1=1;
      check_lru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
      check_mru1(matchcore1[i].mcore1,matchcore1[i].mthread1);
    }

    for(int i=0; i<totalmatch2;i++)
    {
      L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].stat2='s';
      L2[matchcore2[i].mcore2][matchcore2[i].mthread2][index2].v2=1;
      check_lru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
      check_mru2(matchcore2[i].mcore2,matchcore2[i].mthread2);
    }
  }
  update_lru1(core, tt1);
  update_lru2(core, tt2);

  update_mru1(core, tt1);
  update_mru2(core, tt2);
}

if(totalmatch2==0) //miss in bus
{
  // log
  cash0log.busmiss++;

  if (type == 'z')
  {
    // log
    corelog[core].imb++;

    if (L2[core][tt2][index2].stat2=='d')
    {
      // log
      cash0log.wmem++; // write to memory
    }

    // copy from memory to l1 and l2 cache
    L2[core][tt2][index2].t2=tag2;
    L2[core][tt2][index2].v2=1;
```

```
  L2[core][tt2][index2].stat2='v';

  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='v';
}

if (type == 'r')
{
  // log
  corelog[core].rmb++;

  if (L2[core][tt2][index2].stat2=='d')
  {
    // log
    cash0log.wmem++; // write to memory
  }

  // copy from memory to l1 and l2 cache
  L2[core][tt2][index2].t2=tag2;
  L2[core][tt2][index2].v2=1;
  L2[core][tt2][index2].stat2='v';

  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='v';
}

if (type == 'w')
{
  // log
  corelog[core].wmb++;

  if (L2[core][tt2][index2].stat2=='d')
  {
    // log
    cash0log.wmem++; // write to memory
  }

  // copy from memory to l1 and l2 cache
  L1[core][tt1][index1].t1=tag1;
  L1[core][tt1][index1].v1=1;
  L1[core][tt1][index1].stat1='d';

  L2[core][tt2][index2].t2=tag2;
  L2[core][tt2][index2].v2=1;
```

```
      L2[core][tt2][index2].stat2='d';
     }
     update_lru1(core, tt1);
     update_lru2(core, tt2);

     update_mru1(core, tt1);
     update_mru2(core, tt2);
    }
  }
}

void log_sc()
{
  std::locale myLocale(std::locale(), new comma_numpunct());
  std::cout.imbue(myLocale);

  for (int i=0; i<coreno; i++)
  {

totalins=totalins+corelog[i].rhl1+corelog[i].rml1+corelog[i].whl1+corelog[i].wml1+corelog[i].ih
l1+corelog[i].iml1;

    totalrml1= totalrml1+corelog[i].rml1;
    totalwml1= totalwml1+corelog[i].wml1;
    totaliml1= totaliml1+corelog[i].iml1;
    totalmissl1=totalmissl1+corelog[i].rml1+corelog[i].wml1+corelog[i].iml1;

    totalrml2= totalrml2+corelog[i].rml2;
    totalwml2= totalwml2+corelog[i].wml2;
    totaliml2= totaliml2+corelog[i].iml2;
    totalmissl2=totalmissl2+corelog[i].rml2+corelog[i].wml2+corelog[i].iml2;

  }
  //cout <<"starting....."<<endl;
  int totalReads = totalrml1+corelog[0].rhl1;
  int totalWrites = totalwml1+corelog[0].whl1;
  double missRateL1 = 0, missRateL2 = 0;

  if(victimLines == 0)
    missRateL1 = (100*((totalrml1+totalwml1+totaliml1)))/totalins;
  else
    missRateL1 = (100*((totalrml1+totalwml1+totaliml1)))/totalins;

  missRateL2 = ((corelog[0].rml2+corelog[0].wml2)/
(corelog[0].rml1+corelog[0].wml1+corelog[0].vrh+corelog[0].vwh)) * 100;
  double globalMissRate = (missRateL1 * missRateL2)/100;
```

```
  int victimCacheAccesses = (corelog[0].vrh+corelog[0].vwh +
corelog[0].vrm+corelog[0].vwm);

  //check if L2 parameters are all 0, i.e. just L1
  if(parametersL2.back() != 48) //48 is ASCII of 0, L1 and L2
  {
    cout << /*"Total Instructions:"<<*/static_cast<int>(totalins)<<endl;
    std::cout << /*"Total Reads:" << */static_cast<int>(totalReads) << std::endl;
    std::cout << /*"Total Writes:" << */static_cast<int>(totalWrites) << std::endl;
    std::cout << /*"L1 read hit:" << */static_cast<int>(corelog[0].rhl1) << std::endl;
    cout<< /*"L1 Read Miss:"<< */static_cast<int>(totalrml1)<<endl;
    std::cout << /*"L1 Write Hit:" << */static_cast<int>(corelog[0].whl1) << std::endl;
    cout<< /*"L1 Write Miss:"<< */static_cast<int>(totalwml1)<<endl;
    cout<< /*"L1 Miss Rate:"<< */missRateL1 <<"%"<<endl;


    std::cout << /*"Total victimLines:" << */victimLines << std::endl;
    if(victimLines == 0)
    {
      int victimMisses = (corelog[0].vrm+corelog[0].vwm);
      std::cout << /*"Total victimCache Accesses:" <<
*/static_cast<int>(corelog[0].vrm+corelog[0].vwm) << std::endl;
      std::cout << /*"Total victimCache Hits:" <<
*/static_cast<int>((corelog[0].vrh+corelog[0].vwh)) << std::endl;
      std::cout << /*"Total victimCache Misses:" << */static_cast<int>(victimMisses)<< std::endl;
      std::cout << /*"Total victimCache Read Hits:" << */static_cast<int>(corelog[0].vrh) <<
std::endl;
      std::cout << /*"Total victimCache Read Miss:" << */static_cast<int>(corelog[0].vrm) <<
std::endl;
      std::cout << /*"Total victimCache Write Hits:" << */static_cast<int>(corelog[0].vwh) <<
std::endl;
      std::cout << /*"Total victimCache Write Miss:" << */static_cast<int>(corelog[0].vwm) <<
std::endl;
      std::cout << /*"victimCache Hit Rate:" << */(corelog[0].vrm+corelog[0].vwm) << "%" <<
std::endl;
    }
    else
    {
      std::cout << /*"Total victimCache Accesses:" <<
*/static_cast<int>(corelog[0].vrh+corelog[0].vrm+corelog[0].vwh+corelog[0].vwm) <<
std::endl;
      std::cout << /*"Total victimCache Hits:" <<
*/static_cast<int>((corelog[0].vrh+corelog[0].vwh)) << std::endl;
      std::cout << /*"Total victimCache Misses:" <<
*/static_cast<int>(corelog[0].vrm+corelog[0].vwm)<< std::endl;
```

```
    std::cout << /*"Total victimCache Read Hits:" << */static_cast<int>(corelog[0].vrh) <<
std::endl;
    std::cout << /*"Total victimCache Read Miss:" << */static_cast<int>(corelog[0].vrm) <<
std::endl;
    std::cout << /*"Total victimCache Write Hits:" << */static_cast<int>(corelog[0].vwh) <<
std::endl;
    std::cout << /*"Total victimCache Write Miss:" << */static_cast<int>(corelog[0].vwm) <<
std::endl;
    std::cout << /*"victimCache Hit Rate:" <<
*/(corelog[0].vrh+corelog[0].vwh)/(corelog[0].vrh+corelog[0].vrm+corelog[0].vwh+corelog[0].
vwm)*100 << "%" << std::endl;
   }


   std::cout << /*"Total L2 Calls:" << */static_cast<int>(totalrml1+totalwml1+totaliml1) <<
std::endl;
   std::cout << /*"L2 Read Calls:" << */static_cast<int>(corelog[0].rhl2+corelog[0].rml2) <<
std::endl;
   std::cout << /*"L2 Write Calls:" << */static_cast<int>(corelog[0].wml2+corelog[0].whl2) <<
std::endl;
   std::cout << /*"L2 Read Hit:" << */static_cast<int>(corelog[0].rhl2) << std::endl;
   std::cout << /*"L2 Read Miss:" << */static_cast<int>(corelog[0].rml2) << std::endl;
   std::cout << /*"L2 Write Hit:" << */static_cast<int>(corelog[0].whl2) << std::endl;
   std::cout << /*"L2 Write Miss:" << */static_cast<int>(corelog[0].wml2) << std::endl;
   std::cout << /*"L2 Miss Rate:" << */missRateL2 << "%" << std::endl;
   std::cout << /*"Global Miss Rate:" << */globalMissRate << "%" << std::endl;
  }
  else            //L1 only
  {
   cout << /*"Total Instructions:"<< */std::fixed << static_cast<int>(totalins)<<endl;
   std::cout << /*"Total Reads:" << */static_cast<int>(totalReads) << std::endl;
   std::cout << /*"Total Writes:" << */static_cast<int>(totalWrites) << std::endl;
   std::cout << /*"L1 Read Hit:" << */static_cast<int>(corelog[0].rhl1) << std::endl;
   cout<</*"L1 Read Miss:"<< */static_cast<int>(totalrml1)<<endl;
   std::cout << /*"L1 Write Hit:" << */static_cast<int>(corelog[0].whl1) << std::endl;
   cout<</*"L1 Write Miss:"<< */static_cast<int>(totalwml1)<<endl;
   cout<</*"L1 Miss Rate:"<< */missRateL1 << "%" << endl;

   std::cout << /*"Total victimLines:" << */victimLines << std::endl;
   if(victimLines == 0)
   {
    int victimMisses = (corelog[0].vrm+corelog[0].vwm);
    std::cout << /*"Total victimCache Accesses:" <<
*/static_cast<int>(corelog[0].vrm+corelog[0].vwm) << std::endl;
    std::cout << /*"Total victimCache Hits:" <<
*/static_cast<int>((corelog[0].vrh+corelog[0].vwh)) << std::endl;
```

```
      std::cout << /*"Total victimCache Misses:" << */static_cast<int>(victimMisses)<< std::endl;
      std::cout << /*"Total victimCache Read Hits:" << */static_cast<int>(corelog[0].vrh) <<
std::endl;
      std::cout << /*"Total victimCache Read Miss:" << */static_cast<int>(corelog[0].vrm) <<
std::endl;
      std::cout << /*"Total victimCache Write Hits:" << */static_cast<int>(corelog[0].vwh) <<
std::endl;
      std::cout << /*"Total victimCache Write Miss:" << */static_cast<int>(corelog[0].vwm) <<
std::endl;
      std::cout << /*"victimCache Hit Rate:" <<
*/(corelog[0].vrh+corelog[0].vwh)/(corelog[0].vrh+corelog[0].vrm+corelog[0].vwh+corelog[0].
vwm)*100 << "%" << std::endl;
    }
    else
    {
      int victimMisses = (corelog[0].rml1+corelog[0].wml1) - (corelog[0].vrh+corelog[0].vwh);
      std::cout << /*"Total victimCache Accesses:" <<
*/static_cast<int>(corelog[0].vrh+corelog[0].vrm+corelog[0].vwh+corelog[0].vwm) <<
std::endl;
      std::cout << /*"Total victimCache Hits:" <<
*/static_cast<int>((corelog[0].vrh+corelog[0].vwh)) << std::endl;
      std::cout << /*"Total victimCache Misses:" <<
*/static_cast<int>(corelog[0].vrm+corelog[0].vwm)<< std::endl;
      std::cout << /*"Total victimCache Read Hits:" << */static_cast<int>(corelog[0].vrh) <<
std::endl;
      std::cout << /*"Total victimCache Read Miss:" << */static_cast<int>(corelog[0].vrm) <<
std::endl;
      std::cout << /*"Total victimCache Write Hits:" << */static_cast<int>(corelog[0].vwh) <<
std::endl;
      std::cout << /*"Total victimCache Write Miss:" << */static_cast<int>(corelog[0].vwm) <<
std::endl;
      std::cout << /*"victimCache Hit Rate:" <<
*/static_cast<float>((corelog[0].vrh+corelog[0].vwh)/(corelog[0].vrh+corelog[0].vrm+corelog[0
].vwh+corelog[0].vwm)*100) << "%" << std::endl;
    }
  }
}

void log_mesi_only_l1()
{
  std::locale myLocale(std::locale(), new comma_numpunct());
  std::cout.imbue(myLocale);

  for (int i=0; i<coreno; i++)
  {
```

```
totalins=totalins+corelog[i].rhl1+corelog[i].rml1+corelog[i].whl1+corelog[i].wml1+corelog[i].ih
l1+corelog[i].iml1;

    totalrml1= totalrml1+corelog[i].rml1;
    totalwml1= totalwml1+corelog[i].wml1;
    totaliml1= totaliml1+corelog[i].iml1;
    totalmissl1=totalmissl1+corelog[i].rml1+corelog[i].wml1+corelog[i].iml1;

    totalctoc=totalctoc+corelog[i].ctoc;

bustraffic=bustraffic+corelog[i].imb+corelog[i].ihb+corelog[i].rmb+corelog[i].rhb+corelog[i].w
mb+corelog[i].whb+corelog[i].bu;
  }

  cout<</*"Total Instructions:"<<*/ std::fixed << static_cast<int>(totalins) <<endl;
  long totalReads = 0, totalWrites = 0;
  int totalReadHitsL1 = 0, totalReadMissL1 = 0;
  int totalWriteHitsL1 = 0, totalWriteMissL1 = 0;
  int totalVictimHits = 0, totalVictimMisses = 0;

  int totalBusReadHits = 0, totalBusReadMiss = 0;
  int totalBusWriteHits = 0, totalBusWriteMiss = 0;

  float missRateL1 = 0;
  float totalMissRateL1 = 0;
  float averageMissRateL1 = 0;
  float totalVictimHitRate = 0;

  int totalVictimLines = 0, victimAccessesTotal = 0;
  int totalVictimCacheHits = 0, totalVicitmCacheMisses = 0;
  int totalVictimReadHits = 0, totalVictimReadMisses = 0;
  int totalVictimWriteHits = 0, totalVictimWriteMisses = 0;


  for(int i = 0; i < coreno; i++)
  {
    totalReads += corelog[i].rhl1;
    totalReads += corelog[i].rml1;
    totalWrites += corelog[i].whl1;
    totalWrites += corelog[i].wml1;

    totalReadHitsL1 += corelog[i].rhl1;
    totalReadMissL1 += corelog[i].rml1;
    totalWriteHitsL1 += corelog[i].whl1;
    totalWriteMissL1 += corelog[i].wml1;
```

```
    totalBusReadHits += corelog[i].rhb;
    totalBusReadMiss += corelog[i].rmb;
    totalBusWriteHits += corelog[i].whb;
    totalBusWriteMiss += corelog[i].wmb;

    totalVictimLines += victimLines;
    victimAccessesTotal += (corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm);
    totalVictimCacheHits += (corelog[i].vrh+corelog[i].vwh);
    totalVicitmCacheMisses += (corelog[i].vrm+corelog[i].vwm);
    totalVictimReadHits += (corelog[i].vrh);
    totalVictimReadMisses += (corelog[i].vrm);
    totalVictimWriteHits += (corelog[i].vwh);
    totalVictimWriteMisses += (corelog[i].vwm);
  }
  std::cout << /*"Total Reads:" << */static_cast<int>(totalReads) << std::endl;
  std::cout << /*"Total Writes:" << */static_cast<int>(totalWrites) << std::endl;

  for ( int i=0; i<coreno;i++)    //local results (core specific)
  {
    missRateL1 = 100*((corelog[i].rml1+corelog[i].wml1)-corelog[i].ctoc) /
(corelog[i].rhl1+corelog[i].whl1+corelog[i].rml1+corelog[i].wml1);
    totalMissRateL1 += missRateL1;

    cout << /*"Core:"<<*/i+1 <<endl;
    std::cout << /*"Core Instructions:" <<
*/static_cast<int>(corelog[i].rhl1+corelog[i].rml1+corelog[i].whl1+corelog[i].wml1) <<
std::endl;
    cout << /*"Core Reads:" << */static_cast<int>(corelog[i].rhl1+corelog[i].rml1)<<endl;
    cout << /*"Core Writes:" << */static_cast<int>(corelog[i].whl1+corelog[i].wml1)<<endl;

    std::cout << /*"L1 Read Hit:" << */static_cast<int>(corelog[i].rhl1) << std::endl;
    cout << /*"L1 Read Miss:"<<*/static_cast<int>(corelog[i].rml1)<< endl;
    std::cout << /*"L1 Write Hit:" << */static_cast<int>(corelog[i].whl1) << std::endl;
    cout << /*"L1 Write Miss:"<<*/static_cast<int>(corelog[i].wml1)<< endl;

    std::cout << /*"Core Bus Traffic:" <<
*/static_cast<int>(corelog[i].rhb+corelog[i].rmb+corelog[i].whb+corelog[i].wmb) << std::endl;
    std::cout << /*"Read Hit Bus:" << */static_cast<int>(corelog[i].rhb) << std::endl;
    cout << /*"Read Miss Bus:"<<*/static_cast<int>(corelog[i].rmb)<<endl;
    std::cout << /*"Write Hit Bus:" << */static_cast<int>(corelog[i].whb) << std::endl;
    cout << /*"Write Miss Bus:"<<*/static_cast<int>(corelog[i].wmb)<<endl;
    cout << /*"Core Cache-to-Cache:"<< */static_cast<int>(corelog[i].ctoc)<<endl;
    cout << /*"Core L1 Miss Rate:" << */missRateL1 <<"%"<<endl;
```

```
   std::cout << /*"Total victimLines:" << */victimLines << std::endl;
   if(victimLines == 0)
   {
     int victimMisses = (corelog[i].vrm+corelog[i].vwm);
     std::cout << /*"Core victimCache Accesses:" <<
*/static_cast<int>(corelog[i].vrm+corelog[i].vwm) << std::endl;
     std::cout << /*"Core victimCache Hits:" <<
*/static_cast<int>((corelog[i].vrh+corelog[i].vwh)) << std::endl;
     std::cout << /*"Core victimCache Misses:" << */static_cast<int>(victimMisses)<< std::endl;
     std::cout << /*"Core victimCache Read Hits:" << */static_cast<int>(corelog[i].vrh) <<
std::endl;
     std::cout << /*"Core victimCache Read Miss:" << */static_cast<int>(corelog[i].vrm) <<
std::endl;
     std::cout << /*"Core victimCache Write Hits:" << */static_cast<int>(corelog[i].vwh) <<
std::endl;
     std::cout << /*"Core victimCache Write Miss:" << */static_cast<int>(corelog[i].vwm) <<
std::endl;
     std::cout << /*"Core victimCache Hit Rate:" <<
*/(corelog[i].vrh+corelog[i].vwh)/(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vw
m)*100 << "%" << std::endl;
   }
   else
   {
     std::cout << /*"Core victimCache Accesses:" <<
*/static_cast<int>(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm) << std::endl;
     std::cout << /*"Core victimCache Hits:" <<
*/static_cast<int>((corelog[i].vrh+corelog[i].vwh)) << std::endl;
     std::cout << /*"Core victimCache Misses:" <<
*/static_cast<int>(corelog[i].vrm+corelog[i].vwm)<< std::endl;
     std::cout << /*"Core victimCache Read Hits:" << */static_cast<int>(corelog[i].vrh) <<
std::endl;
     std::cout << /*"Core victimCache Read Miss:" << */static_cast<int>(corelog[i].vrm) <<
std::endl;
     std::cout << /*"Core victimCache Write Hits:" << */static_cast<int>(corelog[i].vwh) <<
std::endl;
     std::cout << /*"CorevictimCache Write Miss:" << */static_cast<int>(corelog[i].vwm) <<
std::endl;
     std::cout << /*"Core victimCache Hit Rate:" <<
*/(corelog[i].vrh+corelog[i].vwh)/(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vw
m)*100 << "%" << std::endl;
     totalVictimHitRate +=
(corelog[i].vrh+corelog[i].vwh)/(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm)
*100;
   }
   std:cout << /*"Core Reads From Memory:" <<
*/static_cast<int>((corelog[i].rml1+corelog[i].wml1) - corelog[i].ctoc) << std::endl;
```

```
  }

  //global results
  cout <</*"Total Cache-to-Cache:"<<*/static_cast<int>(totalctoc)<<endl;
  cout <</*"Total Bus
Traffic:"<<*/static_cast<int>(totalBusReadHits+totalBusReadMiss+totalBusWriteHits+totalBus
WriteMiss/*bustraffic*/)<<endl;
  cout <</*"Reads From Memory:"<<*/static_cast<int>((totalmissl1-totalctoc))<<endl;
  cout<</*"Global Miss Rate:"<< */(totalMissRateL1/coreno) << "%" <<
std::endl;//(((totalmissl1-totalctoc-allVictimHits)*100)/totalins)<<"%"<<endl;

  std::cout << /*"Total Read Hits L1:" << */totalReadHitsL1 << std::endl;
  std::cout << /*"Total Read Miss L1:" << */totalReadMissL1 << std::endl;
  std::cout << /*"Total Write Hits L1:" << */totalWriteHitsL1 << std::endl;
  std::cout << /*"Total Write Miss L1:" << */totalWriteMissL1 << std::endl;

  std::cout << /*"Total Bus Read Hits:" << */totalBusReadHits << std::endl;
  std::cout << /*"Total Bus Read Miss:" << */totalBusReadMiss << std::endl;
  std::cout << /*"Total Bus Write Hits:" << */totalBusWriteHits << std::endl;
  std::cout << /*"Total Bus Write Miss:" << */totalBusWriteMiss << std::endl;

  std::cout << /*"Total VictimLines: " << */totalVictimLines << std::endl;
  std::cout << /*"victimAccessesTotal: " << */victimAccessesTotal << std::endl;
  std::cout << /*"totalVictimCacheHits: " << */totalVictimCacheHits << std::endl;
  std::cout << /*"totalVicitmCacheMisses: " << */totalVicitmCacheMisses << std::endl;
  std::cout << /*"totalVictimReadHits: " << */totalVictimReadHits << std::endl;
  std::cout << /*"totalVictimReadMisses: " << */totalVictimReadMisses << std::endl;
  std::cout << /*"totalVictimWriteHits: " << */totalVictimWriteHits << std::endl;
  std::cout << /*"totalVictimWriteMisses: " << */totalVictimWriteMisses << std::endl;

  std::cout << "Average: " << totalMissRateL1/coreno << "%" << std::endl;
  std::cout << "Average: " << totalVictimHitRate/coreno << "%" << std::endl;
}

void log_mesi()
{
  std::locale myLocale(std::locale(), new comma_numpunct());
  std::cout.imbue(myLocale);

  for (int i=0; i<coreno; i++)
  {

totalins=totalins+corelog[i].rhl1+corelog[i].rml1+corelog[i].whl1+corelog[i].wml1+corelog[i].ih
l1+corelog[i].iml1;

    totalrml1= totalrml1+corelog[i].rml1;
```

```
    totalwml1= totalwml1+corelog[i].wml1;
    totaliml1= totaliml1+corelog[i].iml1;
    totalmissl1=totalmissl1+corelog[i].rml1+corelog[i].wml1+corelog[i].iml1;

    totalrml2= totalrml2+corelog[i].rml2;
    totalwml2= totalwml2+corelog[i].wml2;
    totaliml2= totaliml2+corelog[i].iml2;
    totalmissl2=totalmissl2+corelog[i].rml2+corelog[i].wml2+corelog[i].iml2;

    totalctoc=totalctoc+corelog[i].ctoc;

bustraffic=bustraffic+corelog[i].imb+corelog[i].ihb+corelog[i].rmb+corelog[i].rhb+corelog[i].w
mb+corelog[i].whb+corelog[i].bu;
  }

  cout<</*"Total Instructions:"<< */std::fixed << static_cast<int>(totalins)<<endl;    //instruction
results
  int totalReads = 0, totalWrites = 0;
  int totalReadsL2 = 0, totalWritesL2 = 0;
  double coreMissL2 = 0, coreMissL1 = 0;
  double totalMissL1 = 0, totalMissL2 = 0;
  double totalHitsL1 = 0, totalHitsL2 = 0;

  int totalReadHitsL1 = 0, totalReadMissL1 = 0;
  int totalWriteHitsL1 = 0, totalWriteMissL1 = 0;
  int totalBusReadHits = 0, totalBusReadMiss = 0;
  int totalBusWriteHits = 0, totalBusWriteMiss = 0;

  int totalReadHitsL2 = 0, totalReadMissL2 = 0;
  int totalWriteHitsL2 = 0, totalWriteMissL2 = 0;

  float missRateL1 = 0, missRateL2 = 0;
  float totalMissRateL1 = 0, totalMissRateL2 = 0;
  float averageMissRateL1 = 0, averageMissRateL2 = 0;
  float totalVictimHitRate = 0;

  int totalVictimLines = 0, victimAccessesTotal = 0;
  int totalVictimCacheHits = 0, totalVicitmCacheMisses = 0;
  int totalVictimReadHits = 0, totalVictimReadMisses = 0;
  int totalVictimWriteHits = 0, totalVictimWriteMisses = 0;

  for(int i = 0; i < coreno; i++)
  {
    totalReads += corelog[i].rhl1;
    totalReads += corelog[i].rml1;
    totalWrites += corelog[i].whl1;
```

```
  totalWrites += corelog[i].wml1;

  totalReadsL2 += corelog[i].rhl2;
  totalReadsL2 += corelog[i].rml2;
  totalWritesL2 += corelog[i].whl2;
  totalWritesL2 += corelog[i].wml2;

  totalReadHitsL1 += corelog[i].rhl1;
  totalReadMissL1 += corelog[i].rml1;
  totalWriteHitsL1 += corelog[i].whl1;
  totalWriteMissL1 += corelog[i].wml1;

  totalBusReadHits += corelog[i].rhb;
  totalBusReadMiss += corelog[i].rmb;
  totalBusWriteHits += corelog[i].whb;
  totalBusWriteMiss += corelog[i].wmb;

  totalReadHitsL2 += corelog[i].rhl2;
  totalReadMissL2 += corelog[i].rml2;
  totalWriteHitsL2 += corelog[i].whl2;
  totalWriteMissL2 += corelog[i].wml2;

  totalVictimLines += victimLines;
  victimAccessesTotal += (corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm);
  totalVictimCacheHits += (corelog[i].vrh+corelog[i].vwh);
  totalVicitmCacheMisses += (corelog[i].vrm+corelog[i].vwm);
  totalVictimReadHits += (corelog[i].vrh);
  totalVictimReadMisses += (corelog[i].vrm);
  totalVictimWriteHits += (corelog[i].vwh);
  totalVictimWriteMisses += (corelog[i].vwm);
}
std::cout << /*"Total Reads:" << */static_cast<int>(totalReads) << std::endl;
std::cout << /*"Total Writes:" << */static_cast<int>(totalWrites) << std::endl;

for ( int i=0; i<coreno;i++)        //local results (core specific)
{
  coreMissL2 = 0;
  coreMissL1 = 0;

  coreMissL2 += corelog[i].rml2;
  coreMissL2 += corelog[i].wml2;
  coreMissL1 += corelog[i].rml1;
  coreMissL1 += corelog[i].wml1;

  totalMissL1 += corelog[i].rml1;
  totalMissL1 += corelog[i].wml1;
```

```
    totalMissL2 += corelog[i].rml2;
    totalMissL2 += corelog[i].wml2;

    totalHitsL1 += corelog[i].rhl1;
    totalHitsL1 += corelog[i].whl1;
    totalHitsL1 += corelog[i].ctoc;
    totalHitsL2 += corelog[i].rhl2;
    totalHitsL2 += corelog[i].whl2;


    missRateL1 = 100*((corelog[i].rml1+corelog[i].wml1)) /
(corelog[i].rhl1+corelog[i].whl1+corelog[i].rml1+corelog[i].wml1);
    totalMissRateL1 += missRateL1;

    std::cout <</*"Core:" << */i+1 << std::endl;
    std::cout << /*"Core Instructions:" <<
*/static_cast<int>(corelog[i].rhl1+corelog[i].rml1+corelog[i].whl1+corelog[i].wml1) <<
std::endl;
    cout << /*"Core Reads:" << */static_cast<int>(corelog[i].rhl1+corelog[i].rml1)<<endl;
    cout << /*"Core Writes:" << */static_cast<int>(corelog[i].whl1+corelog[i].wml1)<<endl;

    std::cout << /*"L1 Read Hit:" << */static_cast<int>(corelog[i].rhl1) << std::endl;
    cout << /*"L1 Read Miss:"<<*/static_cast<int>(corelog[i].rml1)<< endl;
    std::cout << /*"L1 Write Hit:" << */static_cast<int>(corelog[i].whl1) << std::endl;
    cout << /*"L1 Write Miss:"<<*/static_cast<int>(corelog[i].wml1)<< endl;
    cout << /*"Core L1 Miss Rate:"<< */missRateL1 << "%" << endl;

    std::cout << /*"Total victimLines:" << */victimLines << std::endl;
    if(victimLines == 0)
    {
      int victimMisses = (corelog[i].vrm+corelog[i].vwm);
      std::cout << /*"Core victimCache Accesses:" <<
*/static_cast<int>(corelog[i].vrm+corelog[i].vwm) << std::endl;
      std::cout << /*"Core victimCache Hits:" <<
*/static_cast<int>((corelog[i].vrh+corelog[i].vwh)) << std::endl;
      std::cout << /*"Core victimCache Misses:" << */static_cast<int>(victimMisses)<< std::endl;
      std::cout << /*"Core victimCache Read Hits:" << */static_cast<int>(corelog[i].vrh) <<
std::endl;
      std::cout << /*"Core victimCache Read Miss:" << */static_cast<int>(corelog[i].vrm) <<
std::endl;
      std::cout << /*"Core victimCache Write Hits:" << */static_cast<int>(corelog[i].vwh) <<
std::endl;
      std::cout << /*"Core victimCache Write Miss:" << */static_cast<int>(corelog[i].vwm) <<
std::endl;
```

```
    std::cout << /*"Core victimCache Hit Rate:" <<
*/(corelog[i].vrh+corelog[i].vwh)/(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vw
m)*100 << "%" << std::endl;
   }
   else
   {
    std::cout << /*"Core victimCache Accesses:" <<
*/static_cast<int>(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm) << std::endl;
    std::cout << /*"Core victimCache Hits:" <<
*/static_cast<int>((corelog[i].vrh+corelog[i].vwh)) << std::endl;
    std::cout << /*"Core victimCache Misses:" <<
*/static_cast<int>(corelog[i].vrm+corelog[i].vwm)<< std::endl;
    std::cout << /*"Core victimCache Read Hits:" << */static_cast<int>(corelog[i].vrh) <<
std::endl;
    std::cout << /*"Core victimCache Read Miss:" << */static_cast<int>(corelog[i].vrm) <<
std::endl;
    std::cout << /*"Core victimCache Write Hits:" << */static_cast<int>(corelog[i].vwh) <<
std::endl;
    std::cout << /*"CorevictimCache Write Miss:" << */static_cast<int>(corelog[i].vwm) <<
std::endl;
    std::cout << /*"Core victimCache Hit Rate:" <<
*/(corelog[i].vrh+corelog[i].vwh)/(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vw
m)*100 << "%" << std::endl;
    totalVictimHitRate +=
(corelog[i].vrh+corelog[i].vwh)/(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm)
*100;
   }

   missRateL2 = ((coreMissL2-corelog[i].ctoc)/ (coreMissL1+corelog[i].vrh+corelog[i].vwh)) *
100;
   totalMissRateL2 += missRateL2;


   cout << /*"Core L2 Calls:" << */static_cast<int>(corelog[i].rml1 + corelog[i].wml1)<<endl;
   std::cout << /*"L2 Read Calls:" << */static_cast<int>(corelog[i].rhl2 + corelog[i].rml2) <<
std::endl;
   std::cout << /*"L2 Write Calls:" << */static_cast<int>(corelog[i].whl2 + corelog[i].wml2) <<
std::endl;
   std::cout << /*"L2 Read Hit:" << */static_cast<int>(corelog[i].rhl2) << std::endl;
   cout << /*"L2 Read Miss:"<<*/static_cast<int>(corelog[i].rml2) << endl;
   std::cout << /*"L2 Write Hit:" << */static_cast<int>(corelog[i].whl2) << std::endl;
   cout << /*"L2 Write Miss:"<<*/static_cast<int>(corelog[i].wml2)<< endl;
   std::cout << /*"Core L2 Miss Rate:" << */missRateL2 << "%" << std::endl;

   std::cout << /*"Core Bus Traffic:" <<
*/static_cast<int>(corelog[i].rhb+corelog[i].rmb+corelog[i].whb+corelog[i].wmb) << std::endl;
```

```
    std::cout << /*"Read Hit Bus:" << */static_cast<int>(corelog[i].rhb) << std::endl;
    cout << /*"Read Miss Bus:"<<*/static_cast<int>(corelog[i].rmb)<<endl;
    std::cout << /*"Write Hit Bus:" << */static_cast<int>(corelog[i].whb) << std::endl;
    cout << /*"Write Miss Bus:"<<*/static_cast<int>(corelog[i].wmb)<<endl;
    cout << /*"Core Cache-to-Cache:"<< */static_cast<int>(corelog[i].ctoc)<<endl;
    std:cout << /*"Core Reads From Memory:" <<
*/static_cast<int>((corelog[i].rml2+corelog[i].wml2) - corelog[i].ctoc) << std::endl;
  }

  averageMissRateL1 = totalMissRateL1/coreno;
  averageMissRateL2 = totalMissRateL2/coreno;


  cout <</*"Total Cache-to-Cache:"<<*/static_cast<int>(totalctoc)<<endl;
  cout <</*"Total Bus Traffic:"<<*/static_cast<int>(totalBusReadHits + totalBusReadMiss +
totalBusWriteHits + totalBusWriteMiss/*bustraffic*/)<<endl;
  cout <</*"Reads From Memory:"<<*/static_cast<int>((totalmissl2-totalctoc))<<endl;
  cout<</*"Global Miss Rate:"<< */(averageMissRateL1*averageMissRateL2)/100 << "%" <<
std::endl;//(((totalmissl2-totalctoc-allVictimHits)*100)/totalins)<<"%"<<endl;

  std::cout << /*"Total Read Hits L1:" << */totalReadHitsL1 << std::endl;
  std::cout << /*"Total Read Miss L1:" << */totalReadMissL1 << std::endl;
  std::cout << /*"Total Write Hits L1:" << */totalWriteHitsL1 << std::endl;
  std::cout << /*"Total Write Miss L1:" << */totalWriteMissL1 << std::endl;

  std::cout << /*"Total Bus Read Hits:" << */totalBusReadHits << std::endl;
  std::cout << /*"Total Bus Read Miss:" << */totalBusReadMiss << std::endl;
  std::cout << /*"Total Bus Write Hits:" << */totalBusWriteHits << std::endl;
  std::cout << /*"Total Bus Write Miss:" << */totalBusWriteMiss << std::endl;

  std::cout << /*"Total L2 Calls:" << */totalReadHitsL2 + totalReadMissL2 + totalWriteHitsL2 +
totalWriteMissL2 << std::endl;
  std::cout << /*"Total L2 Read Calls:" << */totalReadHitsL2 + totalReadMissL2 << std::endl;
  std::cout << /*"Total L2 Write Calls:" << */totalWriteHitsL2 + totalWriteMissL2 << std::endl;
  std::cout << /*"L2 Read Hits:" << */totalReadHitsL2 << std::endl;
  std::cout << /*"L2 Read Miss:" << */totalReadMissL2 << std::endl;
  std::cout << /*"L2 Write Hit:" << */totalWriteHitsL2 << std::endl;
  std::cout << /*"L2 Write Miss:" << */totalWriteMissL2 << std::endl;

  std::cout << /*"Total VictimLines: " << */totalVictimLines << std::endl;
  std::cout << /*"victimAccessesTotal: " << */victimAccessesTotal << std::endl;
  std::cout << /*"totalVictimCacheHits: " << */totalVictimCacheHits << std::endl;
  std::cout << /*"totalVicitmCacheMisses: " << */totalVicitmCacheMisses << std::endl;
  std::cout << /*"totalVictimReadHits: " << */totalVictimReadHits << std::endl;
  std::cout << /*"totalVictimReadMisses: " << */totalVictimReadMisses << std::endl;
  std::cout << /*"totalVictimWriteHits: " << */totalVictimWriteHits << std::endl;
```

```
  std::cout << /*"totalVictimWriteMisses: " << */totalVictimWriteMisses << std::endl;

  std::cout << "Average: " << averageMissRateL1 << "%" << std::endl;
  std::cout << "Average: " << totalVictimHitRate/coreno << "%" << std::endl;
  std::cout << "Average: " << averageMissRateL2 << "%" << std::endl;
}

void log_firefly_only_l1()
{
  std::locale myLocale(std::locale(), new comma_numpunct());
  std::cout.imbue(myLocale);

  for (int i=0; i<coreno; i++)
  {

totalins=totalins+corelog[i].rhl1+corelog[i].rml1+corelog[i].whl1+corelog[i].wml1+corelog[i].ih
l1+corelog[i].iml1;

    totalrml1= totalrml1+corelog[i].rml1;
    totalwml1= totalwml1+corelog[i].wml1;
    totaliml1= totaliml1+corelog[i].iml1;
    totalmissl1=totalmissl1+corelog[i].rml1+corelog[i].wml1+corelog[i].iml1;

    totalctoc=totalctoc+corelog[i].ctoc;

bustraffic=bustraffic+corelog[i].imb+corelog[i].ihb+corelog[i].rmb+corelog[i].rhb+corelog[i].w
mb+corelog[i].whb+corelog[i].bu;
  }

  cout<</*"Total Instructions:"<<*/ std::fixed << static_cast<int>(totalins) <<endl;   //instruction
results
  long totalReads = 0, totalWrites = 0;
  int totalReadHitsL1 = 0, totalReadMissL1 = 0;
  int totalWriteHitsL1 = 0, totalWriteMissL1 = 0;
  int totalBusReadHits = 0, totalBusReadMiss = 0;
  int totalBusWriteHits = 0, totalBusWriteMiss = 0;
  float missRateL1 = 0;
  float totalMissRateL1 = 0;
  float averageMissRateL1 = 0;
  float totalVictimHitRate = 0;

  int totalVictimLines = 0, victimAccessesTotal = 0;
  int totalVictimCacheHits = 0, totalVicitmCacheMisses = 0;
  int totalVictimReadHits = 0, totalVictimReadMisses = 0;
  int totalVictimWriteHits = 0, totalVictimWriteMisses = 0;
```

```
  for(int i = 0; i < coreno; i++)
  {
    totalReads += corelog[i].rhl1;
    totalReads += corelog[i].rml1;
    totalWrites += corelog[i].whl1;
    totalWrites += corelog[i].wml1;

    totalReadHitsL1 += corelog[i].rhl1;
    totalReadMissL1 += corelog[i].rml1;
    totalWriteHitsL1 += corelog[i].whl1;
    totalWriteMissL1 += corelog[i].wml1;

    totalBusReadHits += corelog[i].rhb;
    totalBusReadMiss += corelog[i].rmb;
    totalBusWriteHits += corelog[i].whb;
    totalBusWriteMiss += corelog[i].wmb;

    totalVictimLines += victimLines;
    victimAccessesTotal += (corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm);
    totalVictimCacheHits += (corelog[i].vrh+corelog[i].vwh);
    totalVicitmCacheMisses += (corelog[i].vrm+corelog[i].vwm);
    totalVictimReadHits += (corelog[i].vrh);
    totalVictimReadMisses += (corelog[i].vrm);
    totalVictimWriteHits += (corelog[i].vwh);
    totalVictimWriteMisses += (corelog[i].vwm);
  }
  std::cout << /*"Total Reads:" <<*/ static_cast<int>(totalReads) << std::endl;
  std::cout << /*"Total Writes:" <<*/ static_cast<int>(totalWrites) << std::endl;

  for ( int i=0; i<coreno;i++)    //local results (core specific)
  {
    missRateL1 = 100*((corelog[i].rml1+corelog[i].wml1)-corelog[i].ctoc) /
(corelog[i].rhl1+corelog[i].whl1+corelog[i].rml1+corelog[i].wml1);
    totalMissRateL1 += missRateL1;

    cout << /*"Core:"<< */ i+1 <<endl;
    std::cout << /*"Core Instructions:" <<
*/static_cast<int>(corelog[i].rhl1+corelog[i].rml1+corelog[i].whl1+corelog[i].wml1) <<
std::endl;
    cout << /*"Core Reads:" << */static_cast<int>(corelog[i].rhl1+corelog[i].rml1)<<endl;
    cout << /*"Core Writes:" << */static_cast<int>(corelog[i].whl1+corelog[i].wml1)<<endl;

    std::cout << /*"L1 Read Hit:" <<*/ static_cast<int>(corelog[i].rhl1) << std::endl;
    cout << /*"L1 Read Miss:"<<*/static_cast<int>(corelog[i].rml1)<< endl;
    std::cout << /*"L1 Write Hit:" <<*/ static_cast<int>(corelog[i].whl1) << std::endl;
```

```
    cout << /*"L1 Write Miss:"<<*/static_cast<int>(corelog[i].wml1)<< endl;

    std::cout << /*"Core Bus Traffic:" <<*/
static_cast<int>(corelog[i].rhb+corelog[i].rmb+corelog[i].whb+corelog[i].wmb) << std::endl;
    std::cout << /*"Read Hit Bus:" <<*/ static_cast<int>(corelog[i].rhb) << std::endl;
    cout << /*"Read Miss Bus:"<<*/static_cast<int>(corelog[i].rmb)<<endl;
    std::cout << /*"Write Hit Bus:" <<*/ static_cast<int>(corelog[i].whb) << std::endl;
    cout << /*"Write Miss Bus:"<<*/static_cast<int>(corelog[i].wmb)<<endl;
    cout << /*"Core Cache-to-Cache:"<<*/ static_cast<int>(corelog[i].ctoc)<<endl;
    cout << /*"Core L1 Miss Rate:" <<*/ missRateL1 <<"%"<<endl;



    std::cout << /*"Total victimLines:" << */victimLines << std::endl;
    if(victimLines == 0)
    {
      int victimMisses = (corelog[i].vrm+corelog[i].vwm);
      std::cout << /*"Core victimCache Accesses:" <<*/
static_cast<int>(corelog[i].vrm+corelog[i].vwm) << std::endl;
      std::cout << /*"Core victimCache Hits:" <<*/
static_cast<int>((corelog[i].vrh+corelog[i].vwh)) << std::endl;
      std::cout << /*"Core victimCache Misses:" <<*/ static_cast<int>(victimMisses)<< std::endl;
      std::cout << /*"Core victimCache Read Hits:" <<*/ static_cast<int>(corelog[i].vrh) <<
std::endl;
      std::cout << /*"Core victimCache Read Miss:" <<*/ static_cast<int>(corelog[i].vrm) <<
std::endl;
      std::cout << /*"Core victimCache Write Hits:" <<*/ static_cast<int>(corelog[i].vwh) <<
std::endl;
      std::cout << /*"Core victimCache Write Miss:" <<*/ static_cast<int>(corelog[i].vwm) <<
std::endl;
      std::cout << /*"Core victimCache Hit Rate:" <<*/
(corelog[i].vrh+corelog[i].vwh)/(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm)
*100 << "%" << std::endl;
    }
    else
    {
      std::cout << /*"Core victimCache Accesses:" <<*/
static_cast<int>(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm) << std::endl;
      std::cout << /*"Core victimCache Hits:" <<*/
static_cast<int>((corelog[i].vrh+corelog[i].vwh)) << std::endl;
      std::cout << /*"Core victimCache Misses:" <<*/
static_cast<int>(corelog[i].vrm+corelog[i].vwm)<< std::endl;
      std::cout << /*"Core victimCache Read Hits:" <<*/ static_cast<int>(corelog[i].vrh) <<
std::endl;
      std::cout << /*"Core victimCache Read Miss:" <<*/ static_cast<int>(corelog[i].vrm) <<
std::endl;
```

```
    std::cout << /*"Core victimCache Write Hits:" <<*/ static_cast<int>(corelog[i].vwh) <<
std::endl;
    std::cout << /*"CorevictimCache Write Miss:" <<*/ static_cast<int>(corelog[i].vwm) <<
std::endl;
    std::cout << /*"Core victimCache Hit Rate:" <<*/
(corelog[i].vrh+corelog[i].vwh)/(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm)
*100 << "%" << std::endl;
    totalVictimHitRate +=
(corelog[i].vrh+corelog[i].vwh)/(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm)
*100;
   }

   std:cout << "Core Reads From Memory:" <<
static_cast<int>((corelog[i].rml1+corelog[i].wml1) - corelog[i].ctoc) << std::endl;
  }

  //global results
  cout <</*"Total Cache-to-Cache:"<<*/static_cast<int>(totalctoc)<<endl;
  cout <</*"Total Bus
Traffic:"<<*/static_cast<int>(totalBusReadHits+totalBusReadMiss+totalBusWriteHits+totalBus
WriteMiss/*bustraffic*/)<<endl;
  cout <</*"Reads From Memory:"<<*/static_cast<int>((totalmissl1-totalctoc))<<endl;
  cout<</*"Global Miss Rate:"<< */(totalMissRateL1/coreno) << "%" <<
std::endl;//(((totalmissl1-totalctoc-allVictimHits)*100)/totalins)<<"%"<<endl;

  std::cout << /*"Total Read Hits L1:" << */totalReadHitsL1 << std::endl;
  std::cout << /*"Total Read Miss L1:" << */totalReadMissL1 << std::endl;
  std::cout << /*"Total Write Hits L1:" << */totalWriteHitsL1 << std::endl;
  std::cout << /*"Total Write Miss L1:" << */totalWriteMissL1 << std::endl;

  std::cout << /*"Total Bus Read Hits:" << */totalBusReadHits << std::endl;
  std::cout << /*"Total Bus Read Miss:" << */totalBusReadMiss << std::endl;
  std::cout << /*"Total Bus Write Hits:" << */totalBusWriteHits << std::endl;
  std::cout << /*"Total Bus Write Miss:" << */totalBusWriteMiss << std::endl;

  std::cout << /*"Total VictimLines: " << */totalVictimLines << std::endl;
  std::cout << /*"victimAccessesTotal: " << */victimAccessesTotal << std::endl;
  std::cout << /*"totalVictimCacheHits: " << */totalVictimCacheHits << std::endl;
  std::cout << /*"totalVicitmCacheMisses: " << */totalVicitmCacheMisses << std::endl;
  std::cout << /*"totalVictimReadHits: " << */totalVictimReadHits << std::endl;
  std::cout << /*"totalVictimReadMisses: " << */totalVictimReadMisses << std::endl;
  std::cout << /*"totalVictimWriteHits: " << */totalVictimWriteHits << std::endl;
  std::cout << /*"totalVictimWriteMisses: " << */totalVictimWriteMisses << std::endl;

  std::cout << "Average: " << totalMissRateL1/coreno << "%" << std::endl;
  std::cout << "Average: " << totalVictimHitRate/coreno << "%" << std::endl;
```

```
}

void log_firefly()
{
  std::locale myLocale(std::locale(), new comma_numpunct());
  std::cout.imbue(myLocale);

  for (int i=0; i<coreno; i++)
  {

totalins=totalins+corelog[i].rhl1+corelog[i].rml1+corelog[i].whl1+corelog[i].wml1+corelog[i].ih
l1+corelog[i].iml1;

    totalrml1= totalrml1+corelog[i].rml1;
    totalwml1= totalwml1+corelog[i].wml1;
    totaliml1= totaliml1+corelog[i].iml1;
    totalmissl1=totalmissl1+corelog[i].rml1+corelog[i].wml1+corelog[i].iml1;

    totalrml2= totalrml2+corelog[i].rml2;
    totalwml2= totalwml2+corelog[i].wml2;
    totaliml2= totaliml2+corelog[i].iml2;
    totalmissl2=totalmissl2+corelog[i].rml2+corelog[i].wml2+corelog[i].iml2;

    totalctoc=totalctoc+corelog[i].ctoc;

bustraffic=bustraffic+corelog[i].imb+corelog[i].ihb+corelog[i].rmb+corelog[i].rhb+corelog[i].w
mb+corelog[i].whb+corelog[i].bu;
  }

  cout<</*"Total Instructions:"<<*/ std::fixed << static_cast<int>(totalins)<<endl;    //instruction
results
  int totalReads = 0, totalWrites = 0;
  int totalReadsL2 = 0, totalWritesL2 = 0;
  double coreMissL2 = 0, coreMissL1 = 0;
  double totalMissL1 = 0, totalMissL2 = 0;
  double totalHitsL1 = 0, totalHitsL2 = 0;

  int totalReadHitsL1 = 0, totalReadMissL1 = 0;
  int totalWriteHitsL1 = 0, totalWriteMissL1 = 0;
  int totalBusReadHits = 0, totalBusReadMiss = 0;
  int totalBusWriteHits = 0, totalBusWriteMiss = 0;

  int totalReadHitsL2 = 0, totalReadMissL2 = 0;
  int totalWriteHitsL2 = 0, totalWriteMissL2 = 0;

  float missRateL1 = 0, missRateL2 = 0;
```

```
float totalMissRateL1 = 0, totalMissRateL2 = 0;
float averageMissRateL1 = 0, averageMissRateL2 = 0;
float totalVictimHitRate = 0;

int totalVictimLines = 0, victimAccessesTotal = 0;
int totalVictimCacheHits = 0, totalVicitmCacheMisses = 0;
int totalVictimReadHits = 0, totalVictimReadMisses = 0;
int totalVictimWriteHits = 0, totalVictimWriteMisses = 0;

for(int i = 0; i < coreno; i++)
{
  totalReads += corelog[i].rhl1;
  totalReads += corelog[i].rml1;
  totalWrites += corelog[i].whl1;
  totalWrites += corelog[i].wml1;

  totalReadsL2 += corelog[i].rhl2;
  totalReadsL2 += corelog[i].rml2;
  totalWritesL2 += corelog[i].whl2;
  totalWritesL2 += corelog[i].wml2;

  totalReadHitsL1 += corelog[i].rhl1;
  totalReadMissL1 += corelog[i].rml1;
  totalWriteHitsL1 += corelog[i].whl1;
  totalWriteMissL1 += corelog[i].wml1;

  totalBusReadHits += corelog[i].rhb;
  totalBusReadMiss += corelog[i].rmb;
  totalBusWriteHits += corelog[i].whb;
  totalBusWriteMiss += corelog[i].wmb;

  totalReadHitsL2 += corelog[i].rhl2;
  totalReadMissL2 += corelog[i].rml2;
  totalWriteHitsL2 += corelog[i].whl2;
  totalWriteMissL2 += corelog[i].wml2;

  totalVictimLines += victimLines;
  victimAccessesTotal += (corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm);
  totalVictimCacheHits += (corelog[i].vrh+corelog[i].vwh);
  totalVicitmCacheMisses += (corelog[i].vrm+corelog[i].vwm);
  totalVictimReadHits += (corelog[i].vrh);
  totalVictimReadMisses += (corelog[i].vrm);
  totalVictimWriteHits += (corelog[i].vwh);
  totalVictimWriteMisses += (corelog[i].vwm);
}
std::cout << /*"Total Reads:" <<*/ static_cast<int>(totalReads) << std::endl;
```

```
  std::cout << /*"Total Writes:" <<*/ static_cast<int>(totalWrites) << std::endl;

  for ( int i=0; i<coreno;i++)        //local results (core specific)
  {
    coreMissL2 = 0;
    coreMissL1 = 0;

    coreMissL2 += corelog[i].rml2;
    coreMissL2 += corelog[i].wml2;
    coreMissL1 += corelog[i].rml1;
    coreMissL1 += corelog[i].wml1;

    totalMissL1 += corelog[i].rml1;
    totalMissL1 += corelog[i].wml1;
    totalMissL2 += corelog[i].rml2;
    totalMissL2 += corelog[i].wml2;

    totalHitsL1 += corelog[i].rhl1;
    totalHitsL1 += corelog[i].whl1;
    totalHitsL1 += corelog[i].ctoc;
    totalHitsL2 += corelog[i].rhl2;
    totalHitsL2 += corelog[i].whl2;

    missRateL1 = 100*((corelog[i].rml1+corelog[i].wml1)) /
(corelog[i].rhl1+corelog[i].whl1+corelog[i].rml1+corelog[i].wml1);
    totalMissRateL1 += missRateL1;

    std::cout <</*"Core:" <<*/ i+1 << std::endl;
    std::cout << /*"Core Instructions:" <<*/
static_cast<int>(corelog[i].rhl1+corelog[i].rml1+corelog[i].whl1+corelog[i].wml1) << std::endl;
    cout << /*"Core Reads:" <<*/ static_cast<int>(corelog[i].rhl1+corelog[i].rml1)<<endl;
    cout << /*"Core Writes:" <<*/ static_cast<int>(corelog[i].whl1+corelog[i].wml1)<<endl;

    std::cout << /*"L1 Read Hit:" <<*/ static_cast<int>(corelog[i].rhl1) << std::endl;
    cout << /*"L1 Read Miss:"<<*/static_cast<int>(corelog[i].rml1)<< endl;
    std::cout << /*"L1 Write Hit:" <<*/ static_cast<int>(corelog[i].whl1) << std::endl;
    cout << /*"L1 Write Miss:"<<*/static_cast<int>(corelog[i].wml1)<< endl;
    cout << /*"Core L1 Miss Rate:"<<*/ missRateL1 << "%" << endl;


    std::cout << /*"Total victimLines:" <<*/ victimLines << std::endl;
    if(victimLines == 0)
    {
      int victimMisses = (corelog[i].vrm+corelog[i].vwm);
      std::cout << /*"Core victimCache Accesses:" <<*/
static_cast<int>(corelog[i].vrm+corelog[i].vwm) << std::endl;
```

```
    std::cout << /*"Core victimCache Hits:" <<*/
static_cast<int>((corelog[i].vrh+corelog[i].vwh)) << std::endl;
    std::cout << /*"Core victimCache Misses:" <<*/ static_cast<int>(victimMisses)<< std::endl;
    std::cout << /*"Core victimCache Read Hits:" <<*/ static_cast<int>(corelog[i].vrh) <<
std::endl;
    std::cout << /*"Core victimCache Read Miss:" <<*/ static_cast<int>(corelog[i].vrm) <<
std::endl;
    std::cout << /*"Core victimCache Write Hits:" <<*/ static_cast<int>(corelog[i].vwh) <<
std::endl;
    std::cout << /*"Core victimCache Write Miss:" <<*/ static_cast<int>(corelog[i].vwm) <<
std::endl;
    std::cout << /*"Core victimCache Hit Rate:" <<*/
(corelog[i].vrh+corelog[i].vwh)/(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm)
*100 << "%" << std::endl;
   }
   else
   {
    std::cout << /*"Core victimCache Accesses:" <<*/
static_cast<int>(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm) << std::endl;
    std::cout << /*"Core victimCache Hits:" <<*/
static_cast<int>((corelog[i].vrh+corelog[i].vwh)) << std::endl;
    std::cout << /*"Core victimCache Misses:" <<*/
static_cast<int>(corelog[i].vrm+corelog[i].vwm)<< std::endl;
    std::cout << /*"Core victimCache Read Hits:" <<*/ static_cast<int>(corelog[i].vrh) <<
std::endl;
    std::cout << /*"Core victimCache Read Miss:" <<*/ static_cast<int>(corelog[i].vrm) <<
std::endl;
    std::cout << /*"Core victimCache Write Hits:" <<*/ static_cast<int>(corelog[i].vwh) <<
std::endl;
    std::cout << /*"CorevictimCache Write Miss:" <<*/ static_cast<int>(corelog[i].vwm) <<
std::endl;
    std::cout << /*"Core victimCache Hit Rate:" <<*/
(corelog[i].vrh+corelog[i].vwh)/(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm)
*100 << "%" << std::endl;
    totalVictimHitRate +=
(corelog[i].vrh+corelog[i].vwh)/(corelog[i].vrh+corelog[i].vrm+corelog[i].vwh+corelog[i].vwm)
*100;
   }

   missRateL2 = ((coreMissL2-corelog[i].ctoc)/ (coreMissL1+corelog[i].vrh+corelog[i].vwh)) *
100;
   totalMissRateL2 += missRateL2;

   cout << /*"Core L2 Calls:" <<*/ static_cast<int>(corelog[i].rml1 + corelog[i].wml1)<<endl;
   std::cout << /*"L2 Read Calls:" <<*/ static_cast<int>(corelog[i].rhl2 + corelog[i].rml2) <<
std::endl;
```

```
    std::cout << /*"L2 Write Calls:" <<*/ static_cast<int>(corelog[i].whl2 + corelog[i].wml2) <<
std::endl;
    std::cout << /*"L2 Read Hit:" <<*/ static_cast<int>(corelog[i].rhl2) << std::endl;
    cout << /*"L2 Read Miss:"<<*/static_cast<int>(corelog[i].rml2) << endl;
    std::cout << /*"L2 Write Hit:" <<*/ static_cast<int>(corelog[i].whl2) << std::endl;
    cout << /*"L2 Write Miss:"<<*/static_cast<int>(corelog[i].wml2)<< endl;
    std::cout << /*"Core L2 Miss Rate:" <<*/ missRateL2 << "%" << std::endl;

    std::cout << /*"Core Bus Traffic:" <<*/
static_cast<int>(corelog[i].rhb+corelog[i].rmb+corelog[i].whb+corelog[i].wmb) << std::endl;
    std::cout << /*"Read Hit Bus:" <<*/ static_cast<int>(corelog[i].rhb) << std::endl;
    cout << /*"Read Miss Bus:"<<*/static_cast<int>(corelog[i].rmb)<<endl;
    std::cout << /*"Write Hit Bus:" <<*/ static_cast<int>(corelog[i].whb) << std::endl;
    cout << /*"Write Miss Bus:"<<*/static_cast<int>(corelog[i].wmb)<<endl;
    cout << /*"Core Cache-to-Cache:"<<*/ static_cast<int>(corelog[i].ctoc)<<endl;

    std:cout << /*"Core Reads From Memory:" <<*/
static_cast<int>((corelog[i].rml2+corelog[i].wml2) - corelog[i].ctoc) << std::endl;
  }

  averageMissRateL1 = totalMissRateL1/coreno;
  averageMissRateL2 = totalMissRateL2/coreno;


  cout <</*"Total Cache-to-Cache:"<<*/static_cast<int>(totalctoc)<<endl;
  cout <</*"Total Bus Traffic:"<<*/static_cast<int>(totalBusReadHits + totalBusReadMiss +
totalBusWriteHits + totalBusWriteMiss/*bustraffic*/)<<endl;
  cout <</*"Reads From Memory:"<<*/static_cast<int>((totalmissl2-totalctoc))<<endl;
  cout<</*"Global Miss Rate:"<<*/ (averageMissRateL1*averageMissRateL2)/100 << "%" <<
std::endl;//(((totalmissl2-totalctoc-allVictimHits)*100)/totalins)<<"%"<<endl;

  std::cout << /*"Total Read Hits L1:" <<*/ totalReadHitsL1 << std::endl;
  std::cout << /*"Total Read Miss L1:" <<*/ totalReadMissL1 << std::endl;
  std::cout << /*"Total Write Hits L1:" <<*/ totalWriteHitsL1 << std::endl;
  std::cout << /*"Total Write Miss L1:" <<*/ totalWriteMissL1 << std::endl;

  std::cout << /*"Total Bus Read Hits:" <<*/ totalBusReadHits << std::endl;
  std::cout << /*"Total Bus Read Miss:" <<*/ totalBusReadMiss << std::endl;
  std::cout << /*"Total Bus Write Hits:" <<*/ totalBusWriteHits << std::endl;
  std::cout << /*"Total Bus Write Miss:" <<*/ totalBusWriteMiss << std::endl;

  std::cout << /*"Total L2 Calls:" <<*/ totalReadHitsL2 + totalReadMissL2 + totalWriteHitsL2 +
totalWriteMissL2 << std::endl;
  std::cout << /*"Total L2 Read Calls:" <<*/ totalReadHitsL2 + totalReadMissL2 << std::endl;
  std::cout << /*"Total L2 Write Calls:" <<*/ totalWriteHitsL2 + totalWriteMissL2 << std::endl;
  std::cout << /*"L2 Read Hits:" <<*/ totalReadHitsL2 << std::endl;
```

```
  std::cout << /*"L2 Read Miss:" <<*/ totalReadMissL2 << std::endl;
  std::cout << /*"L2 Write Hit:" <<*/ totalWriteHitsL2 << std::endl;
  std::cout << /*"L2 Write Miss:" <<*/ totalWriteMissL2 << std::endl;

  std::cout << /*"Total VictimLines: " <<*/ totalVictimLines << std::endl;
  std::cout << /*"victimAccessesTotal: " <<*/ victimAccessesTotal << std::endl;
  std::cout << /*"totalVictimCacheHits: " <<*/ totalVictimCacheHits << std::endl;
  std::cout << /*"totalVicitmCacheMisses: " <<*/ totalVicitmCacheMisses << std::endl;
  std::cout << /*"totalVictimReadHits: " <<*/ totalVictimReadHits << std::endl;
  std::cout << /*"totalVictimReadMisses: " <<*/ totalVictimReadMisses << std::endl;
  std::cout << /*"totalVictimWriteHits: " <<*/ totalVictimWriteHits << std::endl;
  std::cout << /*"totalVictimWriteMisses: " <<*/ totalVictimWriteMisses << std::endl;

  std::cout << /*"Average: " <<*/ averageMissRateL1 << "%" << std::endl;
  std::cout << /*"Average: " <<*/ totalVictimHitRate/coreno << "%" << std::endl;
  std::cout << /*"Average: " <<*/ averageMissRateL2 << "%" <<std::endl;
}
```

APPENDIX D

APPENDIX D

NODE-RED SOURCE CODE

NRC-SIM.json


```
[
  {
    "id": "0b67313e235aaf0e",
    "type": "tab",
    "label": "NRC-SIM",
    "disabled": false,
    "info": "",
    "env": []
  },
  {
    "id": "9e5b8a904a32ec7c",
    "type": "ui_form",
    "z": "0b67313e235aaf0e",
    "name": "Parameters",
    "label": "",
    "group": "866972eb095a975d",
    "order": 10,
    "width": "0",
    "height": "0",
    "options": [
      {
        "label": "L1 Cache Size ",
        "value": "cacheSizeL1",
        "type": "number",
        "required": true,
        "rows": null
      },
      {
        "label": "L2 Cache Size",
        "value": "cacheSizeL2",
        "type": "number",
        "required": false,
        "rows": null
      },
      {
        "label": "L1 Block Size ",
        "value": "blockSizeL1",
        "type": "number",
        "required": true,
        "rows": null
      },
      {
        "label": "L2 Block Size",
        "value": "blockSizeL2",
        "type": "number",
```

```
            "required": false,
            "rows": null
        },
        {
            "label": "L1 Associativity",
            "value": "associativityL1",
            "type": "number",
            "required": true,
            "rows": null
        },
        {
            "label": "L2 Associativitiy",
            "value": "associativityL2",
            "type": "number",
            "required": false,
            "rows": null
        }
    ],
    "formValue": {
        "cacheSizeL1": "",
        "cacheSizeL2": "",
        "blockSizeL1": "",
        "blockSizeL2": "",
        "associativityL1": "",
        "associativityL2": ""
    },
    "payload": "",
    "submit": "Submit",
    "cancel": "Cancel",
    "topic": "topic",
    "topicType": "msg",
    "splitLayout": true,
    "className": "",
    "x": 190,
    "y": 540,
    "wires": [
        [
            "8b005e0b6c7db6ee"
        ]
    ]
},
{
    "id": "f9c353b48c1a62c5",
    "type": "exec",
    "z": "0b67313e235aaf0e",
    "command": "./NRC-SIM -l1",
```

"addpay": "payload",
"append": "",
"useSpawn": "false",
"timer": "",
"winHide": false,
"oldrc": false,
"name": "",
"x": 780,
"y": 540,
"wires": [
[
"1d1c79b9f0b6146b"
],
[],
[]
]
},
{
"id": "8b005e0b6c7db6ee",
"type": "function",
"z": "0b67313e235aaf0e",
"name": "setCommandLineParameters",
"func": "let cacheSizeL1 = Math.abs(msg.payload.cacheSizeL1);\nlet blockSizeL1 = Math.abs(msg.payload.blockSizeL1);\nlet associativityL1 = Math.abs(msg.payload.associativityL1);\nlet replacementPolicyL1 = global.get(\"replacementPolicyL1\"); //= msg.payload.replacementPolicyL1;\nlet cacheSizeL2 = Math.abs(msg.payload.cacheSizeL2);\nlet blockSizeL2 = Math.abs(msg.payload.blockSizeL2);\nlet associativityL2 = Math.abs(msg.payload.associativityL2);\nlet replacementPolicyL2 = global.get(\"replacementPolicyL2\"); //= msg.payload.replacementPolicyL2;\nlet coreCount = global.get(\"coreCount\"); //msg.payload.coreCount;\nlet coherenceProtocol = global.get(\"coherenceProtocol\"); //= msg.payload.coherenceProtocol;\nlet traceFile;\nlet writePolicy = global.get(\"writePolicy\");\nlet victimLines = global.get(\"victimLines\");\n\n\nvar cacheParametersMsg = \n{\n \"coreCount\" : coreCount,\n \"coherence\" : coherenceProtocol\n};\n\n\n//global.set(\"coreCount\", coreCount);\n\n//global.payload = global.get(\"coreCount\");\n\n//msg.payload=global.payload;//get(\"coreCount\");\n//return msg;\n\n//set msg.payload = selected tracefile from prior node\nmsg.payload = msg.payload.traceFile;\n//set tracefile = msg.payload\ntraceFile = global.get(\"traceFile\");//= msg.payload;\n\n\n\nif(coreCount == 1)\n{\n if(traceFile == \"Streamcluster\")\n traceFile = \"streamcluster_1c.out\";\n else if(traceFile == \"fluidAnimate\")\n traceFile = \"fluidAnimate_1c.out\"\n else if(traceFile == \"Radix\")\n traceFile = \"radix_1c.out\";\n else if(traceFile == \"Raytrace\")\n traceFile = \"raytrace_1c.out\";\n else if(traceFile == \"Ocean_CP\")\n traceFile = \"oceanCP_1c.out\";\n else if(traceFile == \"Generated Trace File\")\n traceFile = \"pinatrace.out\";\n else \n traceFile = \"fluidAnimate_1c.out\";\n}\nelse if(coreCount == 2)\n{\n if(traceFile == \"Streamcluster\")\n

traceFile = \"streamcluster_2c.out\";\n else if(traceFile == \"fluidAnimate\")\n traceFile = \"fluidAnimate_2c.out\"\n else if(traceFile == \"Radix\")\n traceFile = \"radix_2c.out\";\n else if(traceFile == \"Raytrace\")\n traceFile = \"raytrace_2c.out\";\n else if(traceFile == \"Ocean_CP\")\n traceFile = \"oceanCP_2c.out\";\n else if (traceFile == \"Generated Trace File\")\n traceFile = \"pinatrace.out\";\n else \n traceFile = \"fluidAnimate_2c.out\";\n}\nelse if(coreCount == 4)\n{\n if(traceFile == \"Streamcluster\")\n traceFile = \"streamcluster_4c.out\";\n else if(traceFile == \"fluidAnimate\")\n traceFile = \"fluidAnimate_4c.out\"\n else if(traceFile == \"Radix\")\n traceFile = \"radix_4c.out\";\n else if(traceFile == \"Raytrace\")\n traceFile = \"raytrace_4c.out\";\n else if(traceFile == \"Ocean_CP\")\n traceFile = \"oceanCP_4c.out\";\n else if (traceFile == \"Generated Trace File\")\n traceFile = \"pinatrace.out\";\n else\n traceFile = \"fluidAnimate_4c.out\";\n}\nelse if(coreCount == 6)\n{\n if(traceFile == \"Streamcluster\")\n traceFile = \"streamcluster_6c.out\";\n else if(traceFile == \"fluidAnimate\")\n traceFile = \"fluidAnimate_6c.out\"\n else if(traceFile == \"Radix\")\n traceFile = \"radix_6c.out\";\n else if(traceFile == \"Raytrace\")\n traceFile = \"raytrace_6c.out\";\n else if(traceFile == \"Ocean_CP\")\n traceFile = \"oceanCP_6c.out\";\n else if (traceFile == \"Generated Trace File\")\n traceFile = \"pinatrace.out\";\n else\n traceFile = \"fluidAnimate_6c.out\";\n}\nelse if(coreCount == 8)\n{\n if(traceFile == \"Streamcluster\")\n traceFile = \"streamcluster_8c.out\";\n else if(traceFile == \"fluidAnimate\")\n traceFile = \"fluidAnimate_8c.out\"\n else if(traceFile == \"Radix\")\n traceFile = \"radix_8c.out\";\n else if(traceFile == \"Raytrace\")\n traceFile = \"raytrace_8c.out\";\n else if(traceFile == \"Ocean_CP\")\n traceFile = \"oceanCP_8c.out\";\n else if (traceFile == \"Generated Trace File\")\n traceFile = \"pinatrace.out\";\n else\n traceFile = \"fluidAnimate_8c.out\";\n}\nelse if(coreCount == 12)\n{\n if(traceFile == \"Streamcluster\")\n traceFile = \"streamcluster_12c.out\";\n else if(traceFile == \"fluidAnimate\")\n traceFile = \"fluidAnimate_12c.out\"\n else if(traceFile == \"Radix\")\n traceFile = \"radix_12c.out\";\n else if(traceFile == \"Raytrace\")\n traceFile = \"raytrace_12c.out\";\n else if(traceFile == \"Ocean_CP\")\n traceFile = \"oceanCP_12c.out\";\n else if (traceFile == \"Generated Trace File\")\n traceFile = \"pinatrace.out\";\n else\n traceFile = \"fluidAnimate_12c.out\";\n}\nelse if(coreCount == 16)\n{\n if(traceFile == \"Streamcluster\")\n traceFile = \"streamcluster_16c.out\";\n else if(traceFile == \"fluidAnimate\")\n traceFile = \"fluidAnimate_16c.out\"\n else if(traceFile == \"Radix\")\n traceFile = \"radix_16c.out\";\n else if(traceFile == \"Raytrace\")\n traceFile = \"raytrace_16c.out\";\n else if(traceFile == \"Ocean_CP\")\n traceFile = \"oceanCP_16c.out\";\n else if (traceFile == \"Generated Trace File\")\n traceFile = \"pinatrace.out\";\n else\n traceFile = \"fluidAnimate_16c.out\";\n}\n\n//L1 Replacement Policy\nif(replacementPolicyL1 == \"LRU\")\n replacementPolicyL1 = 1;\nelse if(replacementPolicyL1 == \"randomReplacement\")\n replacementPolicyL1 = 2;\nelse if(replacementPolicyL1 == \"roundRobin\")\n replacementPolicyL1 = 3;\nelse if(replacementPolicyL1 == \"MRU\")\n replacementPolicyL1 = 4;\nelse\n replacementPolicyL1 = 1;\n \n//L2 Replacement Policy\nif(replacementPolicyL2 == \"LRU\")\n replacementPolicyL2 = 1;\nelse if(replacementPolicyL2 == \"randomReplacement\")\n replacementPolicyL2 = 2;\nelse if(replacementPolicyL2 == \"roundRobin\")\n replacementPolicyL2 = 3;\nelse if(replacementPolicyL2 == \"MRU\")\n replacementPolicyL2 = 4;\nelse \n replacementPolicyL2 = 1;\n \n//Write Policy (write back/ write-through) \nif(writePolicy == \"writeBack\")\n writePolicy = 1;\nelse if(writePolicy ==

\"writeThrough\") //\"Write-Through/No-Allocate\")\n   writePolicy = 2; \n   \n
\n\nif(coreCount == 1 && coherenceProtocol != \"coherence_0\")   //check if valid\n
coherenceProtocol = \"coherence_0\";\nif(coreCount != 1 && coherenceProtocol ==
\"coherence_0\")\n   coherenceProtocol = \"coherence_1\";\n//global.set(\"coherenceProtocol\",
coherenceProtocol);\n\n   \n\n//Set Coherence Protocol\nif(coherenceProtocol ==
\"coherence_0\")\n   coherenceProtocol = 0;\nelse if(coherenceProtocol == \"coherence_1\")\n
coherenceProtocol = 1;\nelse if(coherenceProtocol == \"coherence_2\")\n   coherenceProtocol =
2;\nelse if(coherenceProtocol == \"coherence_3\")\n   coherenceProtocol = 3;\n\n\n//check for
null L2 Parameters\nif(cacheSizeL2 == null || blockSizeL2 == null || associativityL2 ==
null)\n{\n   cacheSizeL2 = 0;\n   blockSizeL2 = 0;\n   associativityL2 = 0;\n
replacementPolicyL2 = 0;\n}\n\n//global.set(\"cacheFlagL1\", 0);\n//global.set(\"blockFlagL1\",
0);\n//global.set(\"cacheFlagL2\", 0);\n//global.set(\"blockFlagL2\", 0);\n//check for combination
of zero and non zero L2 Parameters\nif(cacheSizeL2 == 0 && blockSizeL2 != 0)\n{\n
cacheSizeL2 = 0;\n   blockSizeL2 = 0; \n   associativityL2 = 0;\n   replacementPolicyL2 = 0;\n
//global.set(\"cacheFlagL2\", 1);\n}\nif(cacheSizeL2 != 0 && blockSizeL2 == 0)\n{\n
cacheSizeL2 = 0;\n   blockSizeL2 = 0;\n   associativityL2 = 0;\n   replacementPolicyL2 = 0;\n
//global.set(\"blockFlagL2\", 1)\n}\n\n//check for combination of zero and non zero L1
parameters\nif(cacheSizeL1 == 0 && blockSizeL1 != 0)\n{\n   cacheSizeL1 = 13;\n
//global.set(\"cacheFlagL1\", 1);\n}\nif(cacheSizeL1 != 0 && blockSizeL1 == 0)\n{\n
blockSizeL1 = 5;\n   //global.set(\"blockFlagL1\", 1);\n}\n\n\nif(cacheSizeL2 == 0 &&
blockSizeL2 == 0 && associativityL2 == 0)\n   replacementPolicyL2 = 0;\n   \n   \n   \n
\nif(coreCount == null)\n   coreCount = 1;\nif(coherenceProtocol == null)\n{\n   if(coreCount
== null || coreCount == 1)\n   coherenceProtocol = 0;\n   else\n   coherenceProtocol =
1;\n}\nif(coreCount == 1 && coherenceProtocol != 0)   //check if valid\n   coherenceProtocol =
0;\nif(coreCount != 1 && coherenceProtocol == 0)\n   coherenceProtocol = 1;\n\nif(victimLines
== null)\n   victimLines = 0;\nif(writePolicy == null)\n   writePolicy = 1;\nif(traceFile ==
null)\n   traceFile = \"fluidAnimate_1c.out\";\nif(replacementPolicyL1 == null)\n
replacementPolicyL1 = 2;\n\n\n\n\n\nglobal.set(\"cacheSizeL2\", cacheSizeL2);\nmsg.payload =
`${cacheSizeL1}:${blockSizeL1}:${associativityL1}:${replacementPolicyL1} -l2
${cacheSizeL2}:${blockSizeL2}:${associativityL2}:${replacementPolicyL2} -C${coreCount} -
P${coherenceProtocol} -W${writePolicy} -V${victimLines} ${traceFile}`;\n//msg.payload =
global.get(\"coreCount\");\nreturn msg;\n\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 480,
        "y": 540,
        "wires": [
            [
                "f9c353b48c1a62c5"
            ]
        ]
    },
    {

"id": "269da549a28ddc40",
"type": "ui_dropdown",
"z": "0b67313e235aaf0e",
"name": "",
"label": "Trace File",
"tooltip": "",
"place": "Select option",
"group": "866972eb095a975d",
"order": 4,
"width": 0,
"height": 0,
"passthru": false,
"multiple": false,
"options": [
  {
    "label": "Streamcluster",
    "value": "Streamcluster",
    "type": "str"
  },
  {
    "label": "Fluid Animate",
    "value": "fluidAnimate",
    "type": "str"
  },
  {
    "label": "Radix",
    "value": "Radix",
    "type": "str"
  },
  {
    "label": "Raytrace",
    "value": "Raytrace",
    "type": "str"
  },
  {
    "label": "Ocean_CP",
    "value": "Ocean_CP",
    "type": "str"
  },
  {
    "label": "Generated Trace File",
    "value": "Generated Trace File",
    "type": "str"
  }
],
"payload": "",

"topic": "topic",
"topicType": "msg",
"className": "",
"x": 160,
"y": 280,
"wires": [
[
"19b55ad9e73cc697"
]
]
},
{
"id": "19b55ad9e73cc697",
"type": "function",
"z": "0b67313e235aaf0e",
"name": "setTraceFile",
"func": "var traceFile = msg.payload;\n\n\nglobal.set(\"traceFile\", traceFile);\nglobal.payload = global.get(\"traceFile\");\n\n\nif(traceFile == null)\n global.set(\"tF\", \"fluidAnimate\");\nelse\n global.set(\"tF\", traceFile);\n\n\nmsg.payload = global.payload;\n\nreturn msg;",
"outputs": 1,
"noerr": 0,
"initialize": "",
"finalize": "",
"libs": [],
"x": 370,
"y": 280,
"wires": [
[]
]
},
{
"id": "37961ea6d3450e5a",
"type": "ui_dropdown",
"z": "0b67313e235aaf0e",
"name": "",
"label": "L1 Replacement Policy ",
"tooltip": "",
"place": "Select option",
"group": "866972eb095a975d",
"order": 7,
"width": 0,
"height": 0,
"passthru": false,
"multiple": false,
"options": [

```
            {
                "label": "Least Recently Used",
                "value": "LRU",
                "type": "str"
            },
            {
                "label": "Most Recently Used",
                "value": "MRU",
                "type": "str"
            },
            {
                "label": "Random Replacement",
                "value": "randomReplacement",
                "type": "str"
            },
            {
                "label": "Round-Robin Replacement",
                "value": "roundRobin",
                "type": "str"
            }
        ],
        "payload": "",
        "topic": "topic",
        "topicType": "msg",
        "className": "",
        "x": 210,
        "y": 320,
        "wires": [
            [
                "e7769fd676510fe6"
            ]
        ]
    },
    {
        "id": "e7769fd676510fe6",
        "type": "function",
        "z": "0b67313e235aaf0e",
        "name": "setReplacementPolicyL1",
        "func": "var replacementPolicyL1 = msg.payload;\n\nglobal.set(\"replacementPolicyL1\",
replacementPolicyL1);\nglobal.payload = global.get(\"replacementPolicyL1\");\n\nmsg.payload
= global.payload;\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
```

```
    "x": 450,
    "y": 320,
    "wires": [
        []
    ]
},
{
    "id": "dfce6a7625f1305e",
    "type": "ui_dropdown",
    "z": "0b67313e235aaf0e",
    "name": "",
    "label": "Write Policy",
    "tooltip": "",
    "place": "Select option",
    "group": "866972eb095a975d",
    "order": 5,
    "width": 0,
    "height": 0,
    "passthru": true,
    "multiple": false,
    "options": [
        {
            "label": "Write-Back / Allocate",
            "value": "writeBack",
            "type": "str"
        },
        {
            "label": "Write-Through / No-Allocate",
            "value": "writeThrough",
            "type": "str"
        }
    ],
    "payload": "",
    "topic": "topic",
    "topicType": "msg",
    "className": "",
    "x": 170,
    "y": 440,
    "wires": [
        [
            "fd9d2e9de6a9eea6"
        ]
    ]
},
{
    "id": "fd9d2e9de6a9eea6",
```

"type": "function",
"z": "0b67313e235aaf0e",
"name": "setWritePolicy",
"func": "var writePolicy = msg.payload;\n\nglobal.set(\"writePolicy\", writePolicy);\nmsg.payload = global.get(\"writePolicy\");\n\n//msg.payload = global.payload;//global.get(\"writePolicy\");\nreturn msg;",
"outputs": 1,
"noerr": 0,
"initialize": "",
"finalize": "",
"libs": [],
"x": 380,
"y": 440,
"wires": [
[]
]
},
{
"id": "8d39ddf8a2c95f1f",
"type": "ui_dropdown",
"z": "0b67313e235aaf0e",
"name": "",
"label": "Core Count",
"tooltip": "",
"place": "Select option",
"group": "866972eb095a975d",
"order": 3,
"width": 0,
"height": 0,
"passthru": false,
"multiple": false,
"options": [
{
"label": "1",
"value": "coreCount_1c",
"type": "str"
},
{
"label": "2",
"value": "coreCount_2c",
"type": "str"
},
{
"label": "4",
"value": "coreCount_4c",
"type": "str"

```
        },
        {
          "label": "6",
          "value": "coreCount_6c",
          "type": "str"
        },
        {
          "label": "8",
          "value": "coreCount_8c",
          "type": "str"
        },
        {
          "label": "12",
          "value": "coreCount_12c",
          "type": "str"
        },
        {
          "label": "16",
          "value": "coreCount_16c",
          "type": "str"
        }
      ],
      "payload": "",
      "topic": "topic",
      "topicType": "msg",
      "className": "",
      "x": 170,
      "y": 240,
      "wires": [
        [
          "75dd2787e7aba161"
        ]
      ]
    },
    {
      "id": "75dd2787e7aba161",
      "type": "function",
      "z": "0b67313e235aaf0e",
      "name": "setCoreCount",
      "func": "var numCores;// = msg.payload;\n\nif(msg.payload == \"coreCount_1c\")\n
global.set(\"coreCount\", 1);\nelse if(msg.payload == \"coreCount_2c\")\n
global.set(\"coreCount\", 2);\nelse if(msg.payload == \"coreCount_4c\")\n
global.set(\"coreCount\", 4);\nelse if(msg.payload == \"coreCount_6c\")\n
global.set(\"coreCount\", 6);\nelse if(msg.payload == \"coreCount_8c\")\n
global.set(\"coreCount\", 8);\n    else if(msg.payload == \"coreCount_12c\")\n
global.set(\"coreCount\", 12);\nelse if(msg.payload == \"coreCount_16c\")\n
```

global.set(\"coreCount\", 16);\nelse if(msg.payload == \"coreCount_24c\")\n global.set(\"coreCount\", 24);\n\n//global.set(\"traceFile\", traceFile);\nglobal.payload = global.get(\"coreCount\");\n\nmsg.payload = global.payload;\n\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 380,
    "y": 240,
    "wires": [
      []
    ]
  },
  {
    "id": "09fbece92d66b4ce",
    "type": "ui_dropdown",
    "z": "0b67313e235aaf0e",
    "name": "",
    "label": "L2 Replacement Policy ",
    "tooltip": "",
    "place": "Select option",
    "group": "866972eb095a975d",
    "order": 8,
    "width": 0,
    "height": 0,
    "passthru": false,
    "multiple": false,
    "options": [
      {
        "label": "Least Recently Used",
        "value": "LRU",
        "type": "str"
      },
      {
        "label": "Most Recently Used",
        "value": "MRU",
        "type": "str"
      },
      {
        "label": "Random Replacement",
        "value": "randomReplacement",
        "type": "str"
      },
      {
        "label": "Round-Robin Replacement",

```
            "value": "roundRobin",
            "type": "str"
        }
    ],
    "payload": "",
    "topic": "topic",
    "topicType": "msg",
    "className": "",
    "x": 210,
    "y": 360,
    "wires": [
        [
            "3ad0e3a51e16c1b3"
        ]
    ]
},
{
    "id": "3ad0e3a51e16c1b3",
    "type": "function",
    "z": "0b67313e235aaf0e",
    "name": "setReplacementPolicyL2",
    "func": "var replacementPolicyL2 = msg.payload;\n\nglobal.set(\"replacementPolicyL2\",
replacementPolicyL2);\nglobal.payload = global.get(\"replacementPolicyL2\");\n\nmsg.payload
= global.payload;\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 450,
    "y": 360,
    "wires": [
        []
    ]
},
{
    "id": "f45fe5ae3ad714c8",
    "type": "ui_dropdown",
    "z": "0b67313e235aaf0e",
    "name": "",
    "label": "Coherence Protocol ",
    "tooltip": "",
    "place": "Select option",
    "group": "866972eb095a975d",
    "order": 6,
    "width": 0,
```

"height": 0,
"passthru": true,
"multiple": false,
"options": [
{
"label": "Single-Core",
"value": "coherence_0",
"type": "str"
},
{
"label": "MSI",
"value": "coherence_2",
"type": "str"
},
{
"label": "MESI",
"value": "coherence_1",
"type": "str"
},
{
"label": "FIREFLY",
"value": "coherence_3",
"type": "str"
}
],
"payload": "",
"topic": "topic",
"topicType": "msg",
"className": "",
"x": 200,
"y": 400,
"wires": [
[
"ff902c08dcb45862"
]
]
},
{
"id": "ff902c08dcb45862",
"type": "function",
"z": "0b67313e235aaf0e",
"name": "setCoherenceProtocol",
"func": "var coherenceProtocol = msg.payload;\n\nglobal.set(\"coherenceProtocol\", coherenceProtocol);\nglobal.payload = global.get(\"coherenceProtocol\");\n\nmsg.payload = global.payload;\nreturn msg;",
"outputs": 1,

"noerr": 0,
"initialize": "",
"finalize": "",
"libs": [],
"x": 420,
"y": 400,
"wires": [
[]
]
},
{
"id": "1d1c79b9f0b6146b",
"type": "function",
"z": "0b67313e235aaf0e",
"name": "parseOutput",
"func": "let coreCount = global.get(\"coreCount\");\nlet cacheSizeL2 = global.get(\"cacheSizeL2\");\n\nlet output = msg.payload;\nlet myArray = output.split(/\\n/);\n\nlet traceFile = global.get(\"tF\");\nif(traceFile == \"streamcluster\")\n traceFile = \"Streamcluster\";\nelse if(traceFile == \"fluidAnimate\")\n traceFile = \"Fluid Animate\";\n \n \nif(coreCount == 1) //SC\n{\n if(cacheSizeL2 == 0) //SC, L1 only\n {\n let newMsg = {payload : \"\"};\n \n newMsg.payload = [\n {\n \"Core\" : myArray[0],\n \"Core Count\" : myArray[0],\n \"L1 Cache Size\" : myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4], \n \"Coherence Protocol\" : myArray[5],\n \"Write Policy\" : myArray[6],\n \"Total Instructions\" : myArray[7],\n \"Total Read Instructions\" : myArray[8],\n \"Total Write Instructions\" : myArray[9],\n \"L1 Read Hits\" : myArray[10],\n \"L1 Read Misses\" : myArray[11],\n \"L1 Write Hits\" : myArray[12],\n \"L1 Write Misses\" : myArray[13],\n \"L1 Miss Rate\" : myArray[14],\n \"Global Miss Rate\" : myArray[14], \n \n \"Victim Lines\" : myArray[15],\n \"Victim Cache Calls\" : myArray[16],\n \"Victim Cache Hits\" : myArray[17],\n \"Victim Cache Misses\" : myArray[18], \n \"Victim Cache Read Hits\" : myArray[19],\n \"Victim Cache Read Misses\" : myArray[20],\n \"Victim Cache Write Hits\" : myArray[21],\n \"Victim Cache Write Misses\" : myArray[22],\n \"Victim Cache Hit Rate\" : myArray[23],\n \n \"L2 Cache Size\" : \"N/A\",\n \"L2 Block Size\" : \"N/A\",\n \"L2 Associativity\" : \"N/A\", \n \"L2 Replacement Policy\" : \"N/A\", \n \"Total L2 Calls\" : \"N/A\",\n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\",\n \n //might need to include \"N/A\" for bus table \n \"Cache-to-Cache\" : \"N/A\",\n \"Total Bus Traffic\" : \"N/A\", \n \"Reads From Memory\" : \"N/A\",\n \"Bus Read Hits\" : \"N/A\", \n \"Bus Read Misses\" : \"N/A\", \n \"Bus Write Hits\" : \"N/A\", \n \"Bus Write Misses\" : \"N/A\", \n \"Trace File\" : traceFile\n }];\n return newMsg;\n }\n else //SC, L1 and L2\n {\n let newMsg = {payload : \"\"}\n \n newMsg.payload = [\n {\n \"Core\" : myArray[0],\n \"Core Count\" : myArray[0],\n \"L1 Cache

Size\" : myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4], \n \"L2 Cache Size\" : myArray[5],\n \"L2 Block Size\" : myArray[6],\n \"L2 Associativity\" : myArray[7],\n \"L2 Replacement Policy\" : myArray[8], \n \"Coherence Protocol\" : myArray[9],\n \"Write Policy\" : myArray[10],\n \"Total Instructions\" : myArray[11],\n \"Total Read Instructions\" : myArray[12],\n \"Total Write Instructions\" : myArray[13],\n \"L1 Read Hits\" : myArray[14],\n \"L1 Read Misses\" : myArray[15],\n \"L1 Write Hits\" : myArray[16],\n \"L1 Write Misses\" : myArray[17],\n \"L1 Miss Rate\" : myArray[18],\n \n \"Victim Lines\" : myArray[19],\n \"Victim Cache Calls\" : myArray[20],\n \"Victim Cache Hits\" : myArray[21],\n \"Victim Cache Misses\" : myArray[22], \n \"Victim Cache Read Hits\" : myArray[23],\n \"Victim Cache Read Misses\" : myArray[24],\n \"Victim Cache Write Hits\" : myArray[25],\n \"Victim Cache Write Misses\" : myArray[26],\n \"Victim Cache Hit Rate\" : myArray[27],\n \n \"Total L2 Calls\" : myArray[28],\n \"L2 Read Calls\" : myArray[29],\n \"L2 Write Calls\" : myArray[30],\n \"L2 Read Hits\" : myArray[31],\n \"L2 Read Misses\" : myArray[32],\n \"L2 Write Hits\" : myArray[33],\n \"L2 Write Misses\" : myArray[34],\n \"L2 Miss Rate\" : myArray[35],\n \"Global Miss Rate\" : myArray[36], \n \n \"Cache-to-Cache\" : \"N/A\",\n \"Total Bus Traffic\" : \"N/A\", \n \"Reads From Memory\" : \"N/A\",\n \"Bus Read Hits\" : \"N/A\", \n \"Bus Read Misses\" : \"N/A\", \n \"Bus Write Hits\" : \"N/A\", \n \"Bus Write Misses\" : \"N/A\", \n \"Trace File\" : traceFile\n }];\n return newMsg;\n }\n}\nelse if(coreCount == 2) //2C\n{\n if(cacheSizeL2 == 0) //2C, L1 only\n {\n let newMsg = {payload : \"\"}\n \n newMsg.payload = [\n {\n \"Core\" : \"Total\",\n \"Core Count\" : myArray[0],\n \"L1 Cache Size\" : myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4], \n \"Coherence Protocol\" : myArray[5],\n \"Write Policy\" : myArray[6],\n \"Total Instructions\" : myArray[7],\n \"Total Read Instructions\" : myArray[8],\n \"Total Write Instructions\" : myArray[9],\n \n \"Cache-to-Cache\" : myArray[60],\n \"Total Bus Traffic\" : myArray[61], \n \"Reads From Memory\" : myArray[62],\n \"Global Miss Rate\" : myArray[63],\n \"L1 Read Hits\" : myArray[64],\n \"L1 Read Misses\" : myArray[65],\n \"L1 Write Hits\" : myArray[66],\n \"L1 Write Misses\" : myArray[67],\n \n \"Bus Read Hits\" : myArray[68], \n \"Bus Read Misses\" : myArray[69], \n \"Bus Write Hits\" : myArray[70], \n \"Bus Write Misses\" : myArray[71],\n \n \"Victim Lines\" : myArray[72],\n \"Victim Cache Calls\" : myArray[73],\n \"Victim Cache Hits\" : myArray[74],\n \"Victim Cache Misses\" : myArray[75], \n \"Victim Cache Read Hits\" : myArray[76],\n \"Victim Cache Read Misses\" : myArray[77],\n \"Victim Cache Write Hits\" : myArray[78],\n \"Victim Cache Write Misses\" : myArray[79],\n \n \"L2 Cache Size\" : \"N/A\",\n \"L2 Block Size\" : \"N/A\",\n \"L2 Associativity\" : \"N/A\", \n \"L2 Replacement Policy\" : \"N/A\", \n \"Total L2 Calls\" : \"N/A\",\n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\", \n \n \"L1 Miss Rate\" : myArray[80], \n \"Victim Cache Hit Rate\" : myArray[81], \n \"L2 Miss Rate\" : \"N/A\", \n \"Trace File\" : traceFile\n }, \n {\n \"Core\" : myArray[10],

//core 1\n \"Total Instructions\" : myArray[11], \n \"Total Read Instructions\" : myArray[12],\n \"Total Write Instructions\" : myArray[13], \n \"L1 Read Hits\" : myArray[14], \n \"L1 Read Misses\" : myArray[15], \n \"L1 Write Hits\" : myArray[16], \n \"L1 Write Misses\" : myArray[17],\n \"Total Bus Traffic\" : myArray[18], \n \"Bus Read Hits\" : myArray[19],\n \"Bus Read Misses\" : myArray[20], \n \"Bus Write Hits\" : myArray[21], \n \"Bus Write Misses\" : myArray[22], \n \"Cache-to-Cache\" : myArray[23], \n \"L1 Miss Rate\" : myArray[24], \n \n \"Victim Lines\" : myArray[25], \n \"Victim Cache Calls\" : myArray[26], \n \"Victim Cache Hits\" : myArray[27], \n \"Victim Cache Misses\" : myArray[28], \n \"Victim Cache Read Hits\" : myArray[29], \n \"Victim Cache Read Misses\" : myArray[30], \n \"Victim Cache Write Hits\" : myArray[31], \n \"Victim Cache Write Misses\" : myArray[32],\n \"Victim Cache Hit Rate\" : myArray[33], \n \"Reads From Memory\" : myArray[34],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[35], //core 2\n \"Total Instructions\" : myArray[36], \n \"Total Read Instructions\" : myArray[37],\n \"Total Write Instructions\" : myArray[38], \n \"L1 Read Hits\" : myArray[39], \n \"L1 Read Misses\" : myArray[40], \n \"L1 Write Hits\" : myArray[41], \n \"L1 Write Misses\" : myArray[42],\n \"Total Bus Traffic\" : myArray[43], \n \"Bus Read Hits\" : myArray[44],\n \"Bus Read Misses\" : myArray[45], \n \"Bus Write Hits\" : myArray[46], \n \"Bus Write Misses\" : myArray[47],\n \"Cache-to-Cache\" : myArray[48], \n \"L1 Miss Rate\" : myArray[49], \n \n \"Victim Lines\" : myArray[50], \n \"Victim Cache Calls\" : myArray[51], \n \"Victim Cache Hits\" : myArray[52], \n \"Victim Cache Misses\" : myArray[53], \n \"Victim Cache Read Hits\" : myArray[54], \n \"Victim Cache Read Misses\" : myArray[55], \n \"Victim Cache Write Hits\" : myArray[56], \n \"Victim Cache Write Misses\" : myArray[57],\n \"Victim Cache Hit Rate\" : myArray[58], \n \"Reads From Memory\" : myArray[59],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }];\n return newMsg;\n }\n else //2C, L1 and L2\n {\n let newMsg = {payload : \"\"}\n \n newMsg.payload = [\n {\n \"Core\" : \"Total\",\n \"Core Count\" : myArray[0],\n \"L1 Cache Size\" : myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4],\n \"L2 Cache Size\" : myArray[5],\n \"L2 Block Size\" : myArray[6],\n \"L2 Associativity\" : myArray[7], \n \"L2 Replacement Policy\" : myArray[8], \n \"Coherence Protocol\" : myArray[9],\n \"Write Policy\" : myArray[10],\n \"Total Instructions\" : myArray[11],\n \"Total Read Instructions\" : myArray[12],\n \"Total Write Instructions\" : myArray[13],\n \n \"Cache-to-Cache\" : myArray[80],\n \"Total Bus Traffic\" : myArray[81], \n \"Reads From Memory\" : myArray[82],\n \"Global Miss Rate\" : myArray[83],\n \"L1 Read Hits\" : myArray[84],\n \"L1 Read Misses\" : myArray[85],\n \"L1 Write Hits\" : myArray[86],\n \"L1 Write Misses\" : myArray[87],\n \n \"Bus Read Hits\" : myArray[88], \n \"Bus

Read Misses\" : myArray[89], \n \"Bus Write Hits\" : myArray[90], \n \"Bus Write Misses\" : myArray[91],\n \n \"Total L2 Calls\" : myArray[92],\n \"L2 Read Calls\" : myArray[93],\n \"L2 Write Calls\" : myArray[94],\n \"L2 Read Hits\" : myArray[95],\n \"L2 Read Misses\" : myArray[96],\n \"L2 Write Hits\" : myArray[97],\n \"L2 Write Misses\" : myArray[98],\n \n \"Victim Lines\" : myArray[99],\n \"Victim Cache Calls\" : myArray[100],\n \"Victim Cache Hits\" : myArray[101],\n \"Victim Cache Misses\" : myArray[102], \n \"Victim Cache Read Hits\" : myArray[103],\n \"Victim Cache Read Misses\" : myArray[104],\n \"Victim Cache Write Hits\" : myArray[105],\n \"Victim Cache Write Misses\" : myArray[106], \n \n \"L1 Miss Rate\" : myArray[107], \n \"Victim Cache Hit Rate\" : myArray[108], \n \"L2 Miss Rate\" : myArray[109], \n \"Trace File\" : traceFile\n }, \n {\n \"Core\" : myArray[14], //core 1\n \"Total Instructions\" : myArray[15], \n \"Total Read Instructions\" : myArray[16],\n \"Total Write Instructions\" : myArray[17], \n \"L1 Read Hits\" : myArray[18], \n \"L1 Read Misses\" : myArray[19], \n \"L1 Write Hits\" : myArray[20], \n \"L1 Write Misses\" : myArray[21],\n \"L1 Miss Rate\" : myArray[22],\n \n \"Victim Lines\" : myArray[23], \n \"Victim Cache Calls\" : myArray[24], \n \"Victim Cache Hits\" : myArray[25], \n \"Victim Cache Misses\" : myArray[26], \n \"Victim Cache Read Hits\" : myArray[27], \n \"Victim Cache Read Misses\" : myArray[28], \n \"Victim Cache Write Hits\" : myArray[29], \n \"Victim Cache Write Misses\" : myArray[30],\n \"Victim Cache Hit Rate\" : myArray[31], \n \n \"Total L2 Calls\" :myArray[32], \n \"L2 Read Calls\" : myArray[33],\n \"L2 Write Calls\" : myArray[34],\n \"L2 Read Hits\" : myArray[35],\n \"L2 Read Misses\" : myArray[36],\n \"L2 Write Hits\" : myArray[37],\n \"L2 Write Misses\" : myArray[38],\n \"L2 Miss Rate\" : myArray[39], \n \n \"Total Bus Traffic\" : myArray[40], \n \"Bus Read Hits\" : myArray[41],\n \"Bus Read Misses\" : myArray[42], \n \"Bus Write Hits\" : myArray[43], \n \"Bus Write Misses\" : myArray[44], \n \"Cache-to-Cache\" : myArray[45], \n \"Reads From Memory\" : myArray[46]\n }, \n {\n \"Core\" : myArray[47], //core 2\n \"Total Instructions\" : myArray[48], \n \"Total Read Instructions\" : myArray[49],\n \"Total Write Instructions\" : myArray[50], \n \"L1 Read Hits\" : myArray[51], \n \"L1 Read Misses\" : myArray[52], \n \"L1 Write Hits\" : myArray[53], \n \"L1 Write Misses\" : myArray[54],\n \"L1 Miss Rate\" : myArray[55],\n \n \"Victim Lines\" : myArray[56], \n \"Victim Cache Calls\" : myArray[57], \n \"Victim Cache Hits\" : myArray[58], \n \"Victim Cache Misses\" : myArray[59], \n \"Victim Cache Read Hits\" : myArray[60], \n \"Victim Cache Read Misses\" : myArray[61], \n \"Victim Cache Write Hits\" : myArray[62], \n \"Victim Cache Write Misses\" : myArray[63],\n \"Victim Cache Hit Rate\" : myArray[64], \n \n \"Total L2 Calls\" :myArray[65], \n \"L2 Read Calls\" : myArray[66],\n \"L2 Write Calls\" : myArray[67],\n \"L2 Read Hits\" : myArray[68],\n \"L2 Read Misses\" : myArray[69],\n \"L2 Write Hits\" : myArray[70],\n \"L2 Write Misses\" : myArray[71],\n \"L2 Miss Rate\" : myArray[72], \n \n \"Total Bus Traffic\" : myArray[73], \n \"Bus Read Hits\" : myArray[74],\n \"Bus Read Misses\" : myArray[75], \n \"Bus Write Hits\" : myArray[76], \n \"Bus Write Misses\" : myArray[77], \n \"Cache-to-Cache\" : myArray[78], \n \"Reads From Memory\" : myArray[79]\n }];\n return newMsg;\n }\n}\nelse if(coreCount == 4)

//4C\n{\n if(cacheSizeL2 == 0) //4C, L1 only\n {\n let newMsg = {payload : \"\"}\n \n newMsg.payload = [\n {\n \"Core\" : \"Total\",\n \"Core Count\" : myArray[0],\n \"L1 Cache Size\" : myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4], \n \"Coherence Protocol\" : myArray[5],\n \"Write Policy\" : myArray[6],\n \"Total Instructions\" : myArray[7],\n \"Total Read Instructions\" : myArray[8],\n \"Total Write Instructions\" : myArray[9],\n \n \"Cache-to-Cache\" : myArray[110],\n \"Total Bus Traffic\" : myArray[111], \n \"Reads From Memory\" : myArray[112],\n \"Global Miss Rate\" : myArray[113],\n \"L1 Read Hits\" : myArray[114],\n \"L1 Read Misses\" : myArray[115],\n \"L1 Write Hits\" : myArray[116],\n \"L1 Write Misses\" : myArray[117],\n \n \"Bus Read Hits\" : myArray[118], \n \"Bus Read Misses\" : myArray[119], \n \"Bus Write Hits\" : myArray[120], \n \"Bus Write Misses\" : myArray[121],\n \n \"Victim Lines\" : myArray[122],\n \"Victim Cache Calls\" : myArray[123],\n \"Victim Cache Hits\" : myArray[124],\n \"Victim Cache Misses\" : myArray[125], \n \"Victim Cache Read Hits\" : myArray[126],\n \"Victim Cache Read Misses\" : myArray[127],\n \"Victim Cache Write Hits\" : myArray[128],\n \"Victim Cache Write Misses\" : myArray[129],\n \n \"L2 Cache Size\" : \"N/A\",\n \"L2 Block Size\" : \"N/A\",\n \"L2 Associativity\" : \"N/A\", \n \"L2 Replacement Policy\" : \"N/A\", \n \"Total L2 Calls\" : \"N/A\",\n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\", \n \n \"L1 Miss Rate\" : myArray[130], \n \"Victim Cache Hit Rate\" : myArray[131], \n \"L2 Miss Rate\" : \"N/A\", \n \"Trace File\" : traceFile\n }, \n {\n \"Core\" : myArray[10], //core 1\n \"Total Instructions\" : myArray[11], \n \"Total Read Instructions\" : myArray[12],\n \"Total Write Instructions\" : myArray[13], \n \"L1 Read Hits\" : myArray[14], \n \"L1 Read Misses\" : myArray[15], \n \"L1 Write Hits\" : myArray[16], \n \"L1 Write Misses\" : myArray[17],\n \"Total Bus Traffic\" : myArray[18], \n \"Bus Read Hits\" : myArray[19],\n \"Bus Read Misses\" : myArray[20], \n \"Bus Write Hits\" : myArray[21], \n \"Bus Write Misses\" : myArray[22], \n \"Cache-to-Cache\" : myArray[23], \n \"L1 Miss Rate\" : myArray[24], \n \n \"Victim Lines\" : myArray[25], \n \"Victim Cache Calls\" : myArray[26], \n \"Victim Cache Hits\" : myArray[27], \n \"Victim Cache Misses\" : myArray[28], \n \"Victim Cache Read Hits\" : myArray[29], \n \"Victim Cache Read Misses\" : myArray[30], \n \"Victim Cache Write Hits\" : myArray[31], \n \"Victim Cache Write Misses\" : myArray[32],\n \"Victim Cache Hit Rate\" : myArray[33], \n \"Reads From Memory\" : myArray[34],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[35], //core 2\n \"Total Instructions\" : myArray[36], \n \"Total Read Instructions\" : myArray[37],\n \"Total Write Instructions\" : myArray[38], \n \"L1 Read Hits\" : myArray[39], \n \"L1 Read Misses\" : myArray[40], \n \"L1 Write Hits\" : myArray[41], \n \"L1 Write Misses\" : myArray[42],\n \"Total Bus Traffic\" : myArray[43], \n \"Bus Read Hits\" : myArray[44],\n \"Bus Read Misses\" : myArray[45], \n \"Bus Write Hits\" :

myArray[46], \n \"Bus Write Misses\" : myArray[47],\n \"Cache-to-Cache\" : myArray[48], \n \"L1 Miss Rate\" : myArray[49], \n \n \"Victim Lines\" : myArray[50], \n \"Victim Cache Calls\" : myArray[51], \n \"Victim Cache Hits\" : myArray[52], \n \"Victim Cache Misses\" : myArray[53], \n \"Victim Cache Read Hits\" : myArray[54], \n \"Victim Cache Read Misses\" : myArray[55], \n \"Victim Cache Write Hits\" : myArray[56], \n \"Victim Cache Write Misses\" : myArray[57],\n \"Victim Cache Hit Rate\" : myArray[58], \n \"Reads From Memory\" : myArray[59],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[60], //core 3\n \"Total Instructions\" : myArray[61], \n \"Total Read Instructions\" : myArray[62],\n \"Total Write Instructions\" : myArray[63], \n \"L1 Read Hits\" : myArray[64], \n \"L1 Read Misses\" : myArray[65], \n \"L1 Write Hits\" : myArray[66], \n \"L1 Write Misses\" : myArray[67],\n \"Total Bus Traffic\" : myArray[68], \n \"Bus Read Hits\" : myArray[69],\n \"Bus Read Misses\" : myArray[70], \n \"Bus Write Hits\" : myArray[71], \n \"Bus Write Misses\" : myArray[72], \n \"Cache-to-Cache\" : myArray[73], \n \"L1 Miss Rate\" : myArray[74], \n \n \"Victim Lines\" : myArray[75], \n \"Victim Cache Calls\" : myArray[76], \n \"Victim Cache Hits\" : myArray[77], \n \"Victim Cache Misses\" : myArray[78], \n \"Victim Cache Read Hits\" : myArray[79], \n \"Victim Cache Read Misses\" : myArray[80], \n \"Victim Cache Write Hits\" : myArray[81], \n \"Victim Cache Write Misses\" : myArray[82],\n \"Victim Cache Hit Rate\" : myArray[83], \n \"Reads From Memory\" : myArray[84],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[85], //core 4\n \"Total Instructions\" : myArray[86], \n \"Total Read Instructions\" : myArray[87],\n \"Total Write Instructions\" : myArray[88], \n \"L1 Read Hits\" : myArray[89], \n \"L1 Read Misses\" : myArray[90], \n \"L1 Write Hits\" : myArray[91], \n \"L1 Write Misses\" : myArray[92],\n \"Total Bus Traffic\" : myArray[93], \n \"Bus Read Hits\" : myArray[94],\n \"Bus Read Misses\" : myArray[95], \n \"Bus Write Hits\" : myArray[96], \n \"Bus Write Misses\" : myArray[97],\n \"Cache-to-Cache\" : myArray[98], \n \"L1 Miss Rate\" : myArray[99], \n \n \"Victim Lines\" : myArray[100], \n \"Victim Cache Calls\" : myArray[101], \n \"Victim Cache Hits\" : myArray[102], \n \"Victim Cache Misses\" : myArray[103], \n \"Victim Cache Read Hits\" : myArray[104], \n \"Victim Cache Read Misses\" : myArray[105], \n \"Victim Cache Write Hits\" : myArray[106], \n \"Victim Cache Write Misses\" : myArray[107],\n \"Victim Cache Hit Rate\" : myArray[108], \n \"Reads From Memory\" : myArray[109],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }];\n return newMsg;\n }\n else //4C, L1 and L2\n {\n let newMsg = {payload : \"\"}\n \n newMsg.payload = [\n {\n \"Core\" : \"Total\",\n \"Core Count\" : myArray[0],\n \"L1 Cache Size\" :

myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4],\n \"L2 Cache Size\" : myArray[5],\n \"L2 Block Size\" : myArray[6],\n \"L2 Associativity\" : myArray[7], \n \"L2 Replacement Policy\" : myArray[8], \n \"Coherence Protocol\" : myArray[9],\n \"Write Policy\" : myArray[10],\n \"Total Instructions\" : myArray[11],\n \"Total Read Instructions\" : myArray[12],\n \"Total Write Instructions\" : myArray[13],\n \n \"Cache-to-Cache\" : myArray[146],\n \"Total Bus Traffic\" : myArray[147], \n \"Reads From Memory\" : myArray[148],\n \"Global Miss Rate\" : myArray[149],\n \"L1 Read Hits\" : myArray[150],\n \"L1 Read Misses\" : myArray[151],\n \"L1 Write Hits\" : myArray[152],\n \"L1 Write Misses\" : myArray[153],\n \n \"Bus Read Hits\" : myArray[154], \n \"Bus Read Misses\" : myArray[155], \n \"Bus Write Hits\" : myArray[156], \n \"Bus Write Misses\" : myArray[157],\n \n \"Total L2 Calls\" : myArray[158],\n \"L2 Read Calls\" : myArray[159],\n \"L2 Write Calls\" : myArray[160],\n \"L2 Read Hits\" : myArray[161],\n \"L2 Read Misses\" : myArray[162],\n \"L2 Write Hits\" : myArray[163],\n \"L2 Write Misses\" : myArray[164],\n \n \"Victim Lines\" : myArray[165],\n \"Victim Cache Calls\" : myArray[166],\n \"Victim Cache Hits\" : myArray[167],\n \"Victim Cache Misses\" : myArray[168], \n \"Victim Cache Read Hits\" : myArray[169],\n \"Victim Cache Read Misses\" : myArray[170],\n \"Victim Cache Write Hits\" : myArray[171],\n \"Victim Cache Write Misses\" : myArray[172], \n \n \"L1 Miss Rate\" : myArray[173], \n \"Victim Cache Hit Rate\" : myArray[174], \n \"L2 Miss Rate\" : myArray[175], \n \"Trace File\" : traceFile\n }, \n {\n \"Core\" : myArray[14], //core 1\n \"Total Instructions\" : myArray[15], \n \"Total Read Instructions\" : myArray[16],\n \"Total Write Instructions\" : myArray[17], \n \"L1 Read Hits\" : myArray[18], \n \"L1 Read Misses\" : myArray[19], \n \"L1 Write Hits\" : myArray[20], \n \"L1 Write Misses\" : myArray[21],\n \"L1 Miss Rate\" : myArray[22],\n \n \"Victim Lines\" : myArray[23], \n \"Victim Cache Calls\" : myArray[24], \n \"Victim Cache Hits\" : myArray[25], \n \"Victim Cache Misses\" : myArray[26], \n \"Victim Cache Read Hits\" : myArray[27], \n \"Victim Cache Read Misses\" : myArray[28], \n \"Victim Cache Write Hits\" : myArray[29], \n \"Victim Cache Write Misses\" : myArray[30],\n \"Victim Cache Hit Rate\" : myArray[31], \n \n \"Total L2 Calls\" :myArray[32], \n \"L2 Read Calls\" : myArray[33],\n \"L2 Write Calls\" : myArray[34],\n \"L2 Read Hits\" : myArray[35],\n \"L2 Read Misses\" : myArray[36],\n \"L2 Write Hits\" : myArray[37],\n \"L2 Write Misses\" : myArray[38],\n \"L2 Miss Rate\" : myArray[39], \n \n \"Total Bus Traffic\" : myArray[40], \n \"Bus Read Hits\" : myArray[41],\n \"Bus Read Misses\" : myArray[42], \n \"Bus Write Hits\" : myArray[43], \n \"Bus Write Misses\" : myArray[44], \n \"Cache-to-Cache\" : myArray[45], \n \"Reads From Memory\" : myArray[46]\n }, \n {\n \"Core\" : myArray[47], //core 2\n \"Total Instructions\" : myArray[48], \n \"Total Read Instructions\" : myArray[49],\n \"Total Write Instructions\" : myArray[50], \n \"L1 Read Hits\" : myArray[51], \n \"L1 Read Misses\" : myArray[52], \n \"L1 Write Hits\" : myArray[53], \n \"L1 Write Misses\" : myArray[54],\n \"L1 Miss Rate\" : myArray[55],\n \n \"Victim Lines\" : myArray[56], \n \"Victim Cache Calls\" : myArray[57], \n \"Victim Cache Hits\" : myArray[58], \n \"Victim

Cache Misses\" : myArray[59], \n \"Victim Cache Read Hits\" : myArray[60], \n \"Victim Cache Read Misses\" : myArray[61], \n \"Victim Cache Write Hits\" : myArray[62], \n \"Victim Cache Write Misses\" : myArray[63],\n \"Victim Cache Hit Rate\" : myArray[64], \n \n \"Total L2 Calls\" :myArray[65], \n \"L2 Read Calls\" : myArray[66],\n \"L2 Write Calls\" : myArray[67],\n \"L2 Read Hits\" : myArray[68],\n \"L2 Read Misses\" : myArray[69],\n \"L2 Write Hits\" : myArray[70],\n \"L2 Write Misses\" : myArray[71],\n \"L2 Miss Rate\" : myArray[72], \n \n \"Total Bus Traffic\" : myArray[73], \n \"Bus Read Hits\" : myArray[74],\n \"Bus Read Misses\" : myArray[75], \n \"Bus Write Hits\" : myArray[76], \n \"Bus Write Misses\" : myArray[77], \n \"Cache-to-Cache\" : myArray[78], \n \"Reads From Memory\" : myArray[79]\n }, \n {\n \"Core\" : myArray[80], //core 3\n \"Total Instructions\" : myArray[81], \n \"Total Read Instructions\" : myArray[82],\n \"Total Write Instructions\" : myArray[83], \n \"L1 Read Hits\" : myArray[84], \n \"L1 Read Misses\" : myArray[85], \n \"L1 Write Hits\" : myArray[86], \n \"L1 Write Misses\" : myArray[87],\n \"L1 Miss Rate\" : myArray[88],\n \n \"Victim Lines\" : myArray[89], \n \"Victim Cache Calls\" : myArray[90], \n \"Victim Cache Hits\" : myArray[91], \n \"Victim Cache Misses\" : myArray[92], \n \"Victim Cache Read Hits\" : myArray[93], \n \"Victim Cache Read Misses\" : myArray[94], \n \"Victim Cache Write Hits\" : myArray[95], \n \"Victim Cache Write Misses\" : myArray[96],\n \"Victim Cache Hit Rate\" : myArray[97], \n \n \"Total L2 Calls\" :myArray[98], \n \"L2 Read Calls\" : myArray[99],\n \"L2 Write Calls\" : myArray[100],\n \"L2 Read Hits\" : myArray[101],\n \"L2 Read Misses\" : myArray[102],\n \"L2 Write Hits\" : myArray[103],\n \"L2 Write Misses\" : myArray[104],\n \"L2 Miss Rate\" : myArray[105], \n \n \"Total Bus Traffic\" : myArray[106], \n \"Bus Read Hits\" : myArray[107],\n \"Bus Read Misses\" : myArray[108], \n \"Bus Write Hits\" : myArray[109], \n \"Bus Write Misses\" : myArray[110], \n \"Cache-to-Cache\" : myArray[111], \n \"Reads From Memory\" : myArray[112]\n }, \n {\n \"Core\" : myArray[113], //core 4\n \"Total Instructions\" : myArray[114], \n \"Total Read Instructions\" : myArray[115],\n \"Total Write Instructions\" : myArray[116], \n \"L1 Read Hits\" : myArray[117], \n \"L1 Read Misses\" : myArray[118], \n \"L1 Write Hits\" : myArray[119], \n \"L1 Write Misses\" : myArray[120],\n \"L1 Miss Rate\" : myArray[121],\n \n \"Victim Lines\" : myArray[122], \n \"Victim Cache Calls\" : myArray[123], \n \"Victim Cache Hits\" : myArray[124], \n \"Victim Cache Misses\" : myArray[125], \n \"Victim Cache Read Hits\" : myArray[126], \n \"Victim Cache Read Misses\" : myArray[127], \n \"Victim Cache Write Hits\" : myArray[128], \n \"Victim Cache Write Misses\" : myArray[129],\n \"Victim Cache Hit Rate\" : myArray[130], \n \n \"Total L2 Calls\" :myArray[131], \n \"L2 Read Calls\" : myArray[132],\n \"L2 Write Calls\" : myArray[133],\n \"L2 Read Hits\" : myArray[134],\n \"L2 Read Misses\" : myArray[135],\n \"L2 Write Hits\" : myArray[136],\n \"L2 Write Misses\" : myArray[137],\n \"L2 Miss Rate\" : myArray[138], \n \n \"Total Bus Traffic\" : myArray[139], \n \"Bus Read Hits\" : myArray[140],\n \"Bus Read Misses\" : myArray[141], \n \"Bus Write Hits\" : myArray[142], \n \"Bus Write Misses\" : myArray[143], \n \"Cache-to-Cache\" : myArray[144], \n \"Reads From Memory\" : myArray[145]\n }];\n return newMsg;\n }\n}\nelse if(coreCount == 6)

//6C\n{\n if(cacheSizeL2 == 0) //6C, L1 only\n {\n let newMsg = {payload : \"\"}\n \n newMsg.payload = [\n {\n \"Core\" : \"Total\",\n \"Core Count\" : myArray[0],\n \"L1 Cache Size\" : myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4], \n \"Coherence Protocol\" : myArray[5],\n \"Write Policy\" : myArray[6],\n \"Total Instructions\" : myArray[7],\n \"Total Read Instructions\" : myArray[8],\n \"Total Write Instructions\" : myArray[9],\n \n \"Cache-to-Cache\" : myArray[160],\n \"Total Bus Traffic\" : myArray[161], \n \"Reads From Memory\" : myArray[162],\n \"Global Miss Rate\" : myArray[163],\n \"L1 Read Hits\" : myArray[164],\n \"L1 Read Misses\" : myArray[165],\n \"L1 Write Hits\" : myArray[166],\n \"L1 Write Misses\" : myArray[167],\n \n \"Bus Read Hits\" : myArray[168], \n \"Bus Read Misses\" : myArray[169], \n \"Bus Write Hits\" : myArray[170], \n \"Bus Write Misses\" : myArray[171],\n \n \"Victim Lines\" : myArray[172],\n \"Victim Cache Calls\" : myArray[173],\n \"Victim Cache Hits\" : myArray[174],\n \"Victim Cache Misses\" : myArray[175], \n \"Victim Cache Read Hits\" : myArray[176],\n \"Victim Cache Read Misses\" : myArray[177],\n \"Victim Cache Write Hits\" : myArray[178],\n \"Victim Cache Write Misses\" : myArray[179],\n \n \"L2 Cache Size\" : \"N/A\",\n \"L2 Block Size\" : \"N/A\",\n \"L2 Associativity\" : \"N/A\", \n \"L2 Replacement Policy\" : \"N/A\", \n \"Total L2 Calls\" : \"N/A\",\n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\", \n \n \"L1 Miss Rate\" : myArray[180], \n \"Victim Cache Hit Rate\" : myArray[181], \n \"L2 Miss Rate\" : \"N/A\", \n \"Trace File\" : traceFile\n }, \n {\n \"Core\" : myArray[10], //core 1\n \"Total Instructions\" : myArray[11], \n \"Total Read Instructions\" : myArray[12],\n \"Total Write Instructions\" : myArray[13], \n \"L1 Read Hits\" : myArray[14], \n \"L1 Read Misses\" : myArray[15], \n \"L1 Write Hits\" : myArray[16], \n \"L1 Write Misses\" : myArray[17],\n \"Total Bus Traffic\" : myArray[18], \n \"Bus Read Hits\" : myArray[19],\n \"Bus Read Misses\" : myArray[20], \n \"Bus Write Hits\" : myArray[21], \n \"Bus Write Misses\" : myArray[22], \n \"Cache-to-Cache\" : myArray[23], \n \"L1 Miss Rate\" : myArray[24], \n \n \"Victim Lines\" : myArray[25], \n \"Victim Cache Calls\" : myArray[26], \n \"Victim Cache Hits\" : myArray[27], \n \"Victim Cache Misses\" : myArray[28], \n \"Victim Cache Read Hits\" : myArray[29], \n \"Victim Cache Read Misses\" : myArray[30], \n \"Victim Cache Write Hits\" : myArray[31], \n \"Victim Cache Write Misses\" : myArray[32],\n \"Victim Cache Hit Rate\" : myArray[33], \n \"Reads From Memory\" : myArray[34],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[35], //core 2\n \"Total Instructions\" : myArray[36], \n \"Total Read Instructions\" : myArray[37],\n \"Total Write Instructions\" : myArray[38], \n \"L1 Read Hits\" : myArray[39], \n \"L1 Read Misses\" : myArray[40], \n \"L1 Write Hits\" : myArray[41], \n \"L1 Write Misses\" : myArray[42],\n \"Total Bus Traffic\" : myArray[43], \n \"Bus Read Hits\" : myArray[44],\n \"Bus Read Misses\" : myArray[45], \n \"Bus Write Hits\" :

myArray[46], \n \"Bus Write Misses\" : myArray[47],\n \"Cache-to-Cache\" : myArray[48], \n \"L1 Miss Rate\" : myArray[49], \n \n \"Victim Lines\" : myArray[50], \n \"Victim Cache Calls\" : myArray[51], \n \"Victim Cache Hits\" : myArray[52], \n \"Victim Cache Misses\" : myArray[53], \n \"Victim Cache Read Hits\" : myArray[54], \n \"Victim Cache Read Misses\" : myArray[55], \n \"Victim Cache Write Hits\" : myArray[56], \n \"Victim Cache Write Misses\" : myArray[57],\n \"Victim Cache Hit Rate\" : myArray[58], \n \"Reads From Memory\" : myArray[59],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[60], //core 3\n \"Total Instructions\" : myArray[61], \n \"Total Read Instructions\" : myArray[62],\n \"Total Write Instructions\" : myArray[63], \n \"L1 Read Hits\" : myArray[64], \n \"L1 Read Misses\" : myArray[65], \n \"L1 Write Hits\" : myArray[66], \n \"L1 Write Misses\" : myArray[67],\n \"Total Bus Traffic\" : myArray[68], \n \"Bus Read Hits\" : myArray[69],\n \"Bus Read Misses\" : myArray[70], \n \"Bus Write Hits\" : myArray[71], \n \"Bus Write Misses\" : myArray[72], \n \"Cache-to-Cache\" : myArray[73], \n \"L1 Miss Rate\" : myArray[74], \n \n \"Victim Lines\" : myArray[75], \n \"Victim Cache Calls\" : myArray[76], \n \"Victim Cache Hits\" : myArray[77], \n \"Victim Cache Misses\" : myArray[78], \n \"Victim Cache Read Hits\" : myArray[79], \n \"Victim Cache Read Misses\" : myArray[80], \n \"Victim Cache Write Hits\" : myArray[81], \n \"Victim Cache Write Misses\" : myArray[82],\n \"Victim Cache Hit Rate\" : myArray[83], \n \"Reads From Memory\" : myArray[84],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[85], //core 4\n \"Total Instructions\" : myArray[86], \n \"Total Read Instructions\" : myArray[87],\n \"Total Write Instructions\" : myArray[88], \n \"L1 Read Hits\" : myArray[89], \n \"L1 Read Misses\" : myArray[90], \n \"L1 Write Hits\" : myArray[91], \n \"L1 Write Misses\" : myArray[92],\n \"Total Bus Traffic\" : myArray[93], \n \"Bus Read Hits\" : myArray[94],\n \"Bus Read Misses\" : myArray[95], \n \"Bus Write Hits\" : myArray[96], \n \"Bus Write Misses\" : myArray[97],\n \"Cache-to-Cache\" : myArray[98], \n \"L1 Miss Rate\" : myArray[99], \n \n \"Victim Lines\" : myArray[100], \n \"Victim Cache Calls\" : myArray[101], \n \"Victim Cache Hits\" : myArray[102], \n \"Victim Cache Misses\" : myArray[103], \n \"Victim Cache Read Hits\" : myArray[104], \n \"Victim Cache Read Misses\" : myArray[105], \n \"Victim Cache Write Hits\" : myArray[106], \n \"Victim Cache Write Misses\" : myArray[107],\n \"Victim Cache Hit Rate\" : myArray[108], \n \"Reads From Memory\" : myArray[109],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[110], //core 5\n \"Total Instructions\" : myArray[111], \n \"Total Read Instructions\" : myArray[112],\n \"Total Write Instructions\" : myArray[113], \n \"L1 Read Hits\" : myArray[114], \n

\"L1 Read Misses\" : myArray[115], \n \"L1 Write Hits\" : myArray[116], \n \"L1 Write Misses\" : myArray[117],\n \"Total Bus Traffic\" : myArray[118], \n \"Bus Read Hits\" : myArray[119],\n \"Bus Read Misses\" : myArray[120], \n \"Bus Write Hits\" : myArray[121], \n \"Bus Write Misses\" : myArray[122], \n \"Cache-to-Cache\" : myArray[123], \n \"L1 Miss Rate\" : myArray[124], \n \n \"Victim Lines\" : myArray[125], \n \"Victim Cache Calls\" : myArray[126], \n \"Victim Cache Hits\" : myArray[127], \n \"Victim Cache Misses\" : myArray[128], \n \"Victim Cache Read Hits\" : myArray[129], \n \"Victim Cache Read Misses\" : myArray[130], \n \"Victim Cache Write Hits\" : myArray[131], \n \"Victim Cache Write Misses\" : myArray[132],\n \"Victim Cache Hit Rate\" : myArray[133], \n \"Reads From Memory\" : myArray[134],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[135], //core 6\n \"Total Instructions\" : myArray[136], \n \"Total Read Instructions\" : myArray[137],\n \"Total Write Instructions\" : myArray[138], \n \"L1 Read Hits\" : myArray[139], \n \"L1 Read Misses\" : myArray[140], \n \"L1 Write Hits\" : myArray[141], \n \"L1 Write Misses\" : myArray[142],\n \"Total Bus Traffic\" : myArray[143], \n \"Bus Read Hits\" : myArray[144],\n \"Bus Read Misses\" : myArray[145], \n \"Bus Write Hits\" : myArray[146], \n \"Bus Write Misses\" : myArray[147],\n \"Cache-to-Cache\" : myArray[148], \n \"L1 Miss Rate\" : myArray[149], \n \n \"Victim Lines\" : myArray[150], \n \"Victim Cache Calls\" : myArray[151], \n \"Victim Cache Hits\" : myArray[152], \n \"Victim Cache Misses\" : myArray[153], \n \"Victim Cache Read Hits\" : myArray[154], \n \"Victim Cache Read Misses\" : myArray[155], \n \"Victim Cache Write Hits\" : myArray[156], \n \"Victim Cache Write Misses\" : myArray[157],\n \"Victim Cache Hit Rate\" : myArray[158], \n \"Reads From Memory\" : myArray[159],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }];\n return newMsg;\n }\n else //6C, L1 and L2\n {\n let newMsg = {payload : \"\"}\n \n newMsg.payload = [\n {\n \"Core\" : \"Total\",\n \"Core Count\" : myArray[0],\n \"L1 Cache Size\" : myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4],\n \"L2 Cache Size\" : myArray[5],\n \"L2 Block Size\" : myArray[6],\n \"L2 Associativity\" : myArray[7], \n \"L2 Replacement Policy\" : myArray[8], \n \"Coherence Protocol\" : myArray[9],\n \"Write Policy\" : myArray[10],\n \"Total Instructions\" : myArray[11],\n \"Total Read Instructions\" : myArray[12],\n \"Total Write Instructions\" : myArray[13],\n \n \"Cache-to-Cache\" : myArray[212],\n \"Total Bus Traffic\" : myArray[213], \n \"Reads From Memory\" : myArray[214],\n \"Global Miss Rate\" : myArray[215],\n \"L1 Read Hits\" : myArray[216],\n \"L1 Read Misses\" : myArray[217],\n \"L1 Write Hits\" : myArray[218],\n \"L1 Write Misses\" : myArray[219],\n \n \"Bus Read Hits\" : myArray[220], \n \"Bus Read Misses\" : myArray[221], \n \"Bus Write Hits\" : myArray[222], \n \"Bus Write Misses\" : myArray[223],\n \n \"Total L2 Calls\" : myArray[224],\n

\"L2 Read Calls\" : myArray[225],\n \"L2 Write Calls\" : myArray[226],\n \"L2 Read Hits\" : myArray[227],\n \"L2 Read Misses\" : myArray[228],\n \"L2 Write Hits\" : myArray[229],\n \"L2 Write Misses\" : myArray[230],\n \n \"Victim Lines\" : myArray[231],\n \"Victim Cache Calls\" : myArray[232],\n \"Victim Cache Hits\" : myArray[233],\n \"Victim Cache Misses\" : myArray[234], \n \"Victim Cache Read Hits\" : myArray[235],\n \"Victim Cache Read Misses\" : myArray[236],\n \"Victim Cache Write Hits\" : myArray[237],\n \"Victim Cache Write Misses\" : myArray[238], \n \n \"L1 Miss Rate\" : myArray[239], \n \"Victim Cache Hit Rate\" : myArray[240], \n \"L2 Miss Rate\" : myArray[241], \n \"Trace File\" : traceFile\n }, \n {\n \"Core\" : myArray[14], //core 1\n \"Total Instructions\" : myArray[15], \n \"Total Read Instructions\" : myArray[16],\n \"Total Write Instructions\" : myArray[17], \n \"L1 Read Hits\" : myArray[18], \n \"L1 Read Misses\" : myArray[19], \n \"L1 Write Hits\" : myArray[20], \n \"L1 Write Misses\" : myArray[21],\n \"L1 Miss Rate\" : myArray[22],\n \n \"Victim Lines\" : myArray[23], \n \"Victim Cache Calls\" : myArray[24], \n \"Victim Cache Hits\" : myArray[25], \n \"Victim Cache Misses\" : myArray[26], \n \"Victim Cache Read Hits\" : myArray[27], \n \"Victim Cache Read Misses\" : myArray[28], \n \"Victim Cache Write Hits\" : myArray[29], \n \"Victim Cache Write Misses\" : myArray[30],\n \"Victim Cache Hit Rate\" : myArray[31], \n \n \"Total L2 Calls\" :myArray[32], \n \"L2 Read Calls\" : myArray[33],\n \"L2 Write Calls\" : myArray[34],\n \"L2 Read Hits\" : myArray[35],\n \"L2 Read Misses\" : myArray[36],\n \"L2 Write Hits\" : myArray[37],\n \"L2 Write Misses\" : myArray[38],\n \"L2 Miss Rate\" : myArray[39], \n \n \"Total Bus Traffic\" : myArray[40], \n \"Bus Read Hits\" : myArray[41],\n \"Bus Read Misses\" : myArray[42], \n \"Bus Write Hits\" : myArray[43], \n \"Bus Write Misses\" : myArray[44], \n \"Cache-to-Cache\" : myArray[45], \n \"Reads From Memory\" : myArray[46]\n }, \n {\n \"Core\" : myArray[47], //core 2\n \"Total Instructions\" : myArray[48], \n \"Total Read Instructions\" : myArray[49],\n \"Total Write Instructions\" : myArray[50], \n \"L1 Read Hits\" : myArray[51], \n \"L1 Read Misses\" : myArray[52], \n \"L1 Write Hits\" : myArray[53], \n \"L1 Write Misses\" : myArray[54],\n \"L1 Miss Rate\" : myArray[55],\n \n \"Victim Lines\" : myArray[56], \n \"Victim Cache Calls\" : myArray[57], \n \"Victim Cache Hits\" : myArray[58], \n \"Victim Cache Misses\" : myArray[59], \n \"Victim Cache Read Hits\" : myArray[60], \n \"Victim Cache Read Misses\" : myArray[61], \n \"Victim Cache Write Hits\" : myArray[62], \n \"Victim Cache Write Misses\" : myArray[63],\n \"Victim Cache Hit Rate\" : myArray[64], \n \n \"Total L2 Calls\" :myArray[65], \n \"L2 Read Calls\" : myArray[66],\n \"L2 Write Calls\" : myArray[67],\n \"L2 Read Hits\" : myArray[68],\n \"L2 Read Misses\" : myArray[69],\n \"L2 Write Hits\" : myArray[70],\n \"L2 Write Misses\" : myArray[71],\n \"L2 Miss Rate\" : myArray[72], \n \n \"Total Bus Traffic\" : myArray[73], \n \"Bus Read Hits\" : myArray[74],\n \"Bus Read Misses\" : myArray[75], \n \"Bus Write Hits\" : myArray[76], \n \"Bus Write Misses\" : myArray[77], \n \"Cache-to-Cache\" : myArray[78], \n \"Reads From Memory\" : myArray[79]\n }, \n {\n \"Core\" : myArray[80], //core 3\n \"Total Instructions\" : myArray[81], \n \"Total Read Instructions\" : myArray[82],\n \"Total Write Instructions\" : myArray[83],

\n \"L1 Read Hits\" : myArray[84], \n \"L1 Read Misses\" : myArray[85], \n \"L1 Write Hits\" : myArray[86], \n \"L1 Write Misses\" : myArray[87],\n \"L1 Miss Rate\" : myArray[88],\n \n \"Victim Lines\" : myArray[89], \n \"Victim Cache Calls\" : myArray[90], \n \"Victim Cache Hits\" : myArray[91], \n \"Victim Cache Misses\" : myArray[92], \n \"Victim Cache Read Hits\" : myArray[93], \n \"Victim Cache Read Misses\" : myArray[94], \n \"Victim Cache Write Hits\" : myArray[95], \n \"Victim Cache Write Misses\" : myArray[96],\n \"Victim Cache Hit Rate\" : myArray[97], \n \n \"Total L2 Calls\" :myArray[98], \n \"L2 Read Calls\" : myArray[99],\n \"L2 Write Calls\" : myArray[100],\n \"L2 Read Hits\" : myArray[101],\n \"L2 Read Misses\" : myArray[102],\n \"L2 Write Hits\" : myArray[103],\n \"L2 Write Misses\" : myArray[104],\n \"L2 Miss Rate\" : myArray[105], \n \n \"Total Bus Traffic\" : myArray[106], \n \"Bus Read Hits\" : myArray[107],\n \"Bus Read Misses\" : myArray[108], \n \"Bus Write Hits\" : myArray[109], \n \"Bus Write Misses\" : myArray[110], \n \"Cache-to-Cache\" : myArray[111], \n \"Reads From Memory\" : myArray[112]\n }, \n {\n \"Core\" : myArray[113], //core 4\n \"Total Instructions\" : myArray[114], \n \"Total Read Instructions\" : myArray[115],\n \"Total Write Instructions\" : myArray[116], \n \"L1 Read Hits\" : myArray[117], \n \"L1 Read Misses\" : myArray[118], \n \"L1 Write Hits\" : myArray[119], \n \"L1 Write Misses\" : myArray[120],\n \"L1 Miss Rate\" : myArray[121],\n \n \"Victim Lines\" : myArray[122], \n \"Victim Cache Calls\" : myArray[123], \n \"Victim Cache Hits\" : myArray[124], \n \"Victim Cache Misses\" : myArray[125], \n \"Victim Cache Read Hits\" : myArray[126], \n \"Victim Cache Read Misses\" : myArray[127], \n \"Victim Cache Write Hits\" : myArray[128], \n \"Victim Cache Write Misses\" : myArray[129],\n \"Victim Cache Hit Rate\" : myArray[130], \n \n \"Total L2 Calls\" :myArray[131], \n \"L2 Read Calls\" : myArray[132],\n \"L2 Write Calls\" : myArray[133],\n \"L2 Read Hits\" : myArray[134],\n \"L2 Read Misses\" : myArray[135],\n \"L2 Write Hits\" : myArray[136],\n \"L2 Write Misses\" : myArray[137],\n \"L2 Miss Rate\" : myArray[138], \n \n \"Total Bus Traffic\" : myArray[139], \n \"Bus Read Hits\" : myArray[140],\n \"Bus Read Misses\" : myArray[141], \n \"Bus Write Hits\" : myArray[142], \n \"Bus Write Misses\" : myArray[143], \n \"Cache-to-Cache\" : myArray[144], \n \"Reads From Memory\" : myArray[145]\n }, \n {\n \"Core\" : myArray[146], //core 5\n \"Total Instructions\" : myArray[147], \n \"Total Read Instructions\" : myArray[148],\n \"Total Write Instructions\" : myArray[149], \n \"L1 Read Hits\" : myArray[150], \n \"L1 Read Misses\" : myArray[151], \n \"L1 Write Hits\" : myArray[152], \n \"L1 Write Misses\" : myArray[153],\n \"L1 Miss Rate\" : myArray[154],\n \n \"Victim Lines\" : myArray[155], \n \"Victim Cache Calls\" : myArray[156], \n \"Victim Cache Hits\" : myArray[157], \n \"Victim Cache Misses\" : myArray[158], \n \"Victim Cache Read Hits\" : myArray[159], \n \"Victim Cache Read Misses\" : myArray[160], \n \"Victim Cache Write Hits\" : myArray[161], \n \"Victim Cache Write Misses\" : myArray[162],\n \"Victim Cache Hit Rate\" : myArray[163], \n \n \"Total L2 Calls\" :myArray[164], \n \"L2 Read Calls\" : myArray[165],\n \"L2 Write Calls\" : myArray[166],\n \"L2 Read Hits\" : myArray[167],\n \"L2 Read Misses\" : myArray[168],\n \"L2 Write Hits\" : myArray[169],\n \"L2 Write Misses\" : myArray[170],\n \"L2 Miss Rate\" :

myArray[171], \n \n \"Total Bus Traffic\" : myArray[172], \n \"Bus Read Hits\" : myArray[173],\n \"Bus Read Misses\" : myArray[174], \n \"Bus Write Hits\" : myArray[175], \n \"Bus Write Misses\" : myArray[176], \n \"Cache-to-Cache\" : myArray[177], \n \"Reads From Memory\" : myArray[178]\n }, \n {\n \"Core\" : myArray[179], //core 6\n \"Total Instructions\" : myArray[180], \n \"Total Read Instructions\" : myArray[181],\n \"Total Write Instructions\" : myArray[182], \n \"L1 Read Hits\" : myArray[183], \n \"L1 Read Misses\" : myArray[184], \n \"L1 Write Hits\" : myArray[185], \n \"L1 Write Misses\" : myArray[186],\n \"L1 Miss Rate\" : myArray[187],\n \n \"Victim Lines\" : myArray[188], \n \"Victim Cache Calls\" : myArray[189], \n \"Victim Cache Hits\" : myArray[190], \n \"Victim Cache Misses\" : myArray[191], \n \"Victim Cache Read Hits\" : myArray[192], \n \"Victim Cache Read Misses\" : myArray[193], \n \"Victim Cache Write Hits\" : myArray[194], \n \"Victim Cache Write Misses\" : myArray[195],\n \"Victim Cache Hit Rate\" : myArray[196], \n \n \"Total L2 Calls\" :myArray[197], \n \"L2 Read Calls\" : myArray[198],\n \"L2 Write Calls\" : myArray[199],\n \"L2 Read Hits\" : myArray[200],\n \"L2 Read Misses\" : myArray[201],\n \"L2 Write Hits\" : myArray[202],\n \"L2 Write Misses\" : myArray[203],\n \"L2 Miss Rate\" : myArray[204], \n \n \"Total Bus Traffic\" : myArray[205], \n \"Bus Read Hits\" : myArray[206],\n \"Bus Read Misses\" : myArray[207], \n \"Bus Write Hits\" : myArray[208], \n \"Bus Write Misses\" : myArray[209], \n \"Cache-to-Cache\" : myArray[210], \n \"Reads From Memory\" : myArray[211]\n }];\n return newMsg;\n }\n}\nelse if(coreCount == 8) //8C\n{\n if(cacheSizeL2 == 0) //8C, L1 only\n {\n let newMsg = {payload : \"\"}\n \n newMsg.payload = [\n {\n \"Core\" : \"Total\",\n \"Core Count\" : myArray[0],\n \"L1 Cache Size\" : myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4], \n \"Coherence Protocol\" : myArray[5],\n \"Write Policy\" : myArray[6],\n \"Total Instructions\" : myArray[7],\n \"Total Read Instructions\" : myArray[8],\n \"Total Write Instructions\" : myArray[9],\n \n \"Cache-to-Cache\" : myArray[210],\n \"Total Bus Traffic\" : myArray[211], \n \"Reads From Memory\" : myArray[212],\n \"Global Miss Rate\" : myArray[213],\n \"L1 Read Hits\" : myArray[214],\n \"L1 Read Misses\" : myArray[215],\n \"L1 Write Hits\" : myArray[216],\n \"L1 Write Misses\" : myArray[217],\n \n \"Bus Read Hits\" : myArray[218], \n \"Bus Read Misses\" : myArray[219], \n \"Bus Write Hits\" : myArray[220], \n \"Bus Write Misses\" : myArray[221],\n \n \"Victim Lines\" : myArray[222],\n \"Victim Cache Calls\" : myArray[223],\n \"Victim Cache Hits\" : myArray[224],\n \"Victim Cache Misses\" : myArray[225], \n \"Victim Cache Read Hits\" : myArray[226],\n \"Victim Cache Read Misses\" : myArray[227],\n \"Victim Cache Write Hits\" : myArray[228],\n \"Victim Cache Write Misses\" : myArray[229],\n \n \"L2 Cache Size\" : \"N/A\",\n \"L2 Block Size\" : \"N/A\",\n \"L2 Associativity\" : \"N/A\", \n \"L2 Replacement Policy\" : \"N/A\", \n \"Total L2 Calls\" : \"N/A\",\n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\", \n \n \"L1 Miss Rate\" : myArray[230], \n \"Victim Cache Hit Rate\" : myArray[231], \n \"L2 Miss Rate\" : \"N/A\", \n \"Trace File\" : traceFile\n }, \n {\n \"Core\"

: myArray[10], //core 1\n \"Total Instructions\" : myArray[11], \n \"Total Read Instructions\" : myArray[12],\n \"Total Write Instructions\" : myArray[13], \n \"L1 Read Hits\" : myArray[14], \n \"L1 Read Misses\" : myArray[15], \n \"L1 Write Hits\" : myArray[16], \n \"L1 Write Misses\" : myArray[17],\n \"Total Bus Traffic\" : myArray[18], \n \"Bus Read Hits\" : myArray[19],\n \"Bus Read Misses\" : myArray[20], \n \"Bus Write Hits\" : myArray[21], \n \"Bus Write Misses\" : myArray[22], \n \"Cache-to-Cache\" : myArray[23], \n \"L1 Miss Rate\" : myArray[24], \n \n \"Victim Lines\" : myArray[25], \n \"Victim Cache Calls\" : myArray[26], \n \"Victim Cache Hits\" : myArray[27], \n \"Victim Cache Misses\" : myArray[28], \n \"Victim Cache Read Hits\" : myArray[29], \n \"Victim Cache Read Misses\" : myArray[30], \n \"Victim Cache Write Hits\" : myArray[31], \n \"Victim Cache Write Misses\" : myArray[32],\n \"Victim Cache Hit Rate\" : myArray[33], \n \"Reads From Memory\" : myArray[34],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[35], //core 2\n \"Total Instructions\" : myArray[36], \n \"Total Read Instructions\" : myArray[37],\n \"Total Write Instructions\" : myArray[38], \n \"L1 Read Hits\" : myArray[39], \n \"L1 Read Misses\" : myArray[40], \n \"L1 Write Hits\" : myArray[41], \n \"L1 Write Misses\" : myArray[42],\n \"Total Bus Traffic\" : myArray[43], \n \"Bus Read Hits\" : myArray[44],\n \"Bus Read Misses\" : myArray[45], \n \"Bus Write Hits\" : myArray[46], \n \"Bus Write Misses\" : myArray[47],\n \"Cache-to-Cache\" : myArray[48], \n \"L1 Miss Rate\" : myArray[49], \n \n \"Victim Lines\" : myArray[50], \n \"Victim Cache Calls\" : myArray[51], \n \"Victim Cache Hits\" : myArray[52], \n \"Victim Cache Misses\" : myArray[53], \n \"Victim Cache Read Hits\" : myArray[54], \n \"Victim Cache Read Misses\" : myArray[55], \n \"Victim Cache Write Hits\" : myArray[56], \n \"Victim Cache Write Misses\" : myArray[57],\n \"Victim Cache Hit Rate\" : myArray[58], \n \"Reads From Memory\" : myArray[59],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[60], //core 3\n \"Total Instructions\" : myArray[61], \n \"Total Read Instructions\" : myArray[62],\n \"Total Write Instructions\" : myArray[63], \n \"L1 Read Hits\" : myArray[64], \n \"L1 Read Misses\" : myArray[65], \n \"L1 Write Hits\" : myArray[66], \n \"L1 Write Misses\" : myArray[67],\n \"Total Bus Traffic\" : myArray[68], \n \"Bus Read Hits\" : myArray[69],\n \"Bus Read Misses\" : myArray[70], \n \"Bus Write Hits\" : myArray[71], \n \"Bus Write Misses\" : myArray[72], \n \"Cache-to-Cache\" : myArray[73], \n \"L1 Miss Rate\" : myArray[74], \n \n \"Victim Lines\" : myArray[75], \n \"Victim Cache Calls\" : myArray[76], \n \"Victim Cache Hits\" : myArray[77], \n \"Victim Cache Misses\" : myArray[78], \n \"Victim Cache Read Hits\" : myArray[79], \n \"Victim Cache Read Misses\" : myArray[80], \n \"Victim Cache Write Hits\" : myArray[81], \n \"Victim Cache Write Misses\" : myArray[82],\n \"Victim Cache Hit Rate\" : myArray[83], \n \"Reads From Memory\" : myArray[84],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read

Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[85], //core 4\n \"Total Instructions\" : myArray[86], \n \"Total Read Instructions\" : myArray[87],\n \"Total Write Instructions\" : myArray[88], \n \"L1 Read Hits\" : myArray[89], \n \"L1 Read Misses\" : myArray[90], \n \"L1 Write Hits\" : myArray[91], \n \"L1 Write Misses\" : myArray[92],\n \"Total Bus Traffic\" : myArray[93], \n \"Bus Read Hits\" : myArray[94],\n \"Bus Read Misses\" : myArray[95], \n \"Bus Write Hits\" : myArray[96], \n \"Bus Write Misses\" : myArray[97],\n \"Cache-to-Cache\" : myArray[98], \n \"L1 Miss Rate\" : myArray[99], \n \n \"Victim Lines\" : myArray[100], \n \"Victim Cache Calls\" : myArray[101], \n \"Victim Cache Hits\" : myArray[102], \n \"Victim Cache Misses\" : myArray[103], \n \"Victim Cache Read Hits\" : myArray[104], \n \"Victim Cache Read Misses\" : myArray[105], \n \"Victim Cache Write Hits\" : myArray[106], \n \"Victim Cache Write Misses\" : myArray[107],\n \"Victim Cache Hit Rate\" : myArray[108], \n \"Reads From Memory\" : myArray[109],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[110], //core 5\n \"Total Instructions\" : myArray[111], \n \"Total Read Instructions\" : myArray[112],\n \"Total Write Instructions\" : myArray[113], \n \"L1 Read Hits\" : myArray[114], \n \"L1 Read Misses\" : myArray[115], \n \"L1 Write Hits\" : myArray[116], \n \"L1 Write Misses\" : myArray[117],\n \"Total Bus Traffic\" : myArray[118], \n \"Bus Read Hits\" : myArray[119],\n \"Bus Read Misses\" : myArray[120], \n \"Bus Write Hits\" : myArray[121], \n \"Bus Write Misses\" : myArray[122], \n \"Cache-to-Cache\" : myArray[123], \n \"L1 Miss Rate\" : myArray[124], \n \n \"Victim Lines\" : myArray[125], \n \"Victim Cache Calls\" : myArray[126], \n \"Victim Cache Hits\" : myArray[127], \n \"Victim Cache Misses\" : myArray[128], \n \"Victim Cache Read Hits\" : myArray[129], \n \"Victim Cache Read Misses\" : myArray[130], \n \"Victim Cache Write Hits\" : myArray[131], \n \"Victim Cache Write Misses\" : myArray[132],\n \"Victim Cache Hit Rate\" : myArray[133], \n \"Reads From Memory\" : myArray[134],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[135], //core 6\n \"Total Instructions\" : myArray[136], \n \"Total Read Instructions\" : myArray[137],\n \"Total Write Instructions\" : myArray[138], \n \"L1 Read Hits\" : myArray[139], \n \"L1 Read Misses\" : myArray[140], \n \"L1 Write Hits\" : myArray[141], \n \"L1 Write Misses\" : myArray[142],\n \"Total Bus Traffic\" : myArray[143], \n \"Bus Read Hits\" : myArray[144],\n \"Bus Read Misses\" : myArray[145], \n \"Bus Write Hits\" : myArray[146], \n \"Bus Write Misses\" : myArray[147],\n \"Cache-to-Cache\" : myArray[148], \n \"L1 Miss Rate\" : myArray[149], \n \n \"Victim Lines\" : myArray[150], \n \"Victim Cache Calls\" : myArray[151], \n \"Victim Cache Hits\" : myArray[152], \n \"Victim Cache Misses\" : myArray[153], \n \"Victim Cache

Read Hits\" : myArray[154], \n \"Victim Cache Read Misses\" : myArray[155], \n \"Victim Cache Write Hits\" : myArray[156], \n \"Victim Cache Write Misses\" : myArray[157],\n \"Victim Cache Hit Rate\" : myArray[158], \n \"Reads From Memory\" : myArray[159],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[160], //core 7\n \"Total Instructions\" : myArray[161], \n \"Total Read Instructions\" : myArray[162],\n \"Total Write Instructions\" : myArray[163], \n \"L1 Read Hits\" : myArray[164], \n \"L1 Read Misses\" : myArray[165], \n \"L1 Write Hits\" : myArray[166], \n \"L1 Write Misses\" : myArray[167],\n \"Total Bus Traffic\" : myArray[168], \n \"Bus Read Hits\" : myArray[169],\n \"Bus Read Misses\" : myArray[170], \n \"Bus Write Hits\" : myArray[171], \n \"Bus Write Misses\" : myArray[172], \n \"Cache-to-Cache\" : myArray[173], \n \"L1 Miss Rate\" : myArray[174], \n \n \"Victim Lines\" : myArray[175], \n \"Victim Cache Calls\" : myArray[176], \n \"Victim Cache Hits\" : myArray[177], \n \"Victim Cache Misses\" : myArray[178], \n \"Victim Cache Read Hits\" : myArray[179], \n \"Victim Cache Read Misses\" : myArray[180], \n \"Victim Cache Write Hits\" : myArray[181], \n \"Victim Cache Write Misses\" : myArray[182],\n \"Victim Cache Hit Rate\" : myArray[183], \n \"Reads From Memory\" : myArray[184],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[185], //core 8\n \"Total Instructions\" : myArray[186], \n \"Total Read Instructions\" : myArray[187],\n \"Total Write Instructions\" : myArray[188], \n \"L1 Read Hits\" : myArray[189], \n \"L1 Read Misses\" : myArray[190], \n \"L1 Write Hits\" : myArray[191], \n \"L1 Write Misses\" : myArray[192],\n \"Total Bus Traffic\" : myArray[193], \n \"Bus Read Hits\" : myArray[194],\n \"Bus Read Misses\" : myArray[195], \n \"Bus Write Hits\" : myArray[196], \n \"Bus Write Misses\" : myArray[197],\n \"Cache-to-Cache\" : myArray[198], \n \"L1 Miss Rate\" : myArray[199], \n \n \"Victim Lines\" : myArray[200], \n \"Victim Cache Calls\" : myArray[201], \n \"Victim Cache Hits\" : myArray[202], \n \"Victim Cache Misses\" : myArray[203], \n \"Victim Cache Read Hits\" : myArray[204], \n \"Victim Cache Read Misses\" : myArray[205], \n \"Victim Cache Write Hits\" : myArray[206], \n \"Victim Cache Write Misses\" : myArray[207],\n \"Victim Cache Hit Rate\" : myArray[208], \n \"Reads From Memory\" : myArray[209],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }];\n return newMsg;\n }\n else //8C, L1 and L2\n {\n let newMsg = {payload : \"\"}\n \n newMsg.payload = [\n {\n \"Core\" : \"Total\",\n \"Core Count\" : myArray[0],\n \"L1 Cache Size\" : myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4],\n \"L2 Cache Size\" : myArray[5],\n \"L2 Block Size\" : myArray[6],\n \"L2 Associativity\" : myArray[7], \n \"L2 Replacement Policy\" :

myArray[8], \n \"Coherence Protocol\" : myArray[9],\n \"Write Policy\" : myArray[10],\n \"Total Instructions\" : myArray[11],\n \"Total Read Instructions\" : myArray[12],\n \"Total Write Instructions\" : myArray[13],\n \n \"Cache-to-Cache\" : myArray[278],\n \"Total Bus Traffic\" : myArray[279], \n \"Reads From Memory\" : myArray[280],\n \"Global Miss Rate\" : myArray[281],\n \"L1 Read Hits\" : myArray[282],\n \"L1 Read Misses\" : myArray[283],\n \"L1 Write Hits\" : myArray[284],\n \"L1 Write Misses\" : myArray[285],\n \n \"Bus Read Hits\" : myArray[286], \n \"Bus Read Misses\" : myArray[287], \n \"Bus Write Hits\" : myArray[288], \n \"Bus Write Misses\" : myArray[289],\n \n \"Total L2 Calls\" : myArray[290],\n \"L2 Read Calls\" : myArray[291],\n \"L2 Write Calls\" : myArray[292],\n \"L2 Read Hits\" : myArray[293],\n \"L2 Read Misses\" : myArray[294],\n \"L2 Write Hits\" : myArray[295],\n \"L2 Write Misses\" : myArray[296],\n \n \"Victim Lines\" : myArray[297],\n \"Victim Cache Calls\" : myArray[298],\n \"Victim Cache Hits\" : myArray[299],\n \"Victim Cache Misses\" : myArray[300], \n \"Victim Cache Read Hits\" : myArray[301],\n \"Victim Cache Read Misses\" : myArray[302],\n \"Victim Cache Write Hits\" : myArray[303],\n \"Victim Cache Write Misses\" : myArray[304], \n \n \"L1 Miss Rate\" : myArray[305], \n \"Victim Cache Hit Rate\" : myArray[306], \n \"L2 Miss Rate\" : myArray[307], \n \"Trace File\" : traceFile\n }, \n {\n \"Core\" : myArray[14], //core 1\n \"Total Instructions\" : myArray[15], \n \"Total Read Instructions\" : myArray[16],\n \"Total Write Instructions\" : myArray[17], \n \"L1 Read Hits\" : myArray[18], \n \"L1 Read Misses\" : myArray[19], \n \"L1 Write Hits\" : myArray[20], \n \"L1 Write Misses\" : myArray[21],\n \"L1 Miss Rate\" : myArray[22],\n \n \"Victim Lines\" : myArray[23], \n \"Victim Cache Calls\" : myArray[24], \n \"Victim Cache Hits\" : myArray[25], \n \"Victim Cache Misses\" : myArray[26], \n \"Victim Cache Read Hits\" : myArray[27], \n \"Victim Cache Read Misses\" : myArray[28], \n \"Victim Cache Write Hits\" : myArray[29], \n \"Victim Cache Write Misses\" : myArray[30],\n \"Victim Cache Hit Rate\" : myArray[31], \n \n \"Total L2 Calls\" :myArray[32], \n \"L2 Read Calls\" : myArray[33],\n \"L2 Write Calls\" : myArray[34],\n \"L2 Read Hits\" : myArray[35],\n \"L2 Read Misses\" : myArray[36],\n \"L2 Write Hits\" : myArray[37],\n \"L2 Write Misses\" : myArray[38],\n \"L2 Miss Rate\" : myArray[39], \n \n \"Total Bus Traffic\" : myArray[40], \n \"Bus Read Hits\" : myArray[41],\n \"Bus Read Misses\" : myArray[42], \n \"Bus Write Hits\" : myArray[43], \n \"Bus Write Misses\" : myArray[44], \n \"Cache-to-Cache\" : myArray[45], \n \"Reads From Memory\" : myArray[46]\n }, \n {\n \"Core\" : myArray[47], //core 2\n \"Total Instructions\" : myArray[48], \n \"Total Read Instructions\" : myArray[49],\n \"Total Write Instructions\" : myArray[50], \n \"L1 Read Hits\" : myArray[51], \n \"L1 Read Misses\" : myArray[52], \n \"L1 Write Hits\" : myArray[53], \n \"L1 Write Misses\" : myArray[54],\n \"L1 Miss Rate\" : myArray[55],\n \n \"Victim Lines\" : myArray[56], \n \"Victim Cache Calls\" : myArray[57], \n \"Victim Cache Hits\" : myArray[58], \n \"Victim Cache Misses\" : myArray[59], \n \"Victim Cache Read Hits\" : myArray[60], \n \"Victim Cache Read Misses\" : myArray[61], \n \"Victim Cache Write Hits\" : myArray[62], \n \"Victim Cache Write Misses\" : myArray[63],\n \"Victim Cache Hit Rate\" : myArray[64], \n \n \"Total L2 Calls\" :myArray[65], \n \"L2

Read Calls\" : myArray[66],\n \"L2 Write Calls\" : myArray[67],\n \"L2 Read Hits\" : myArray[68],\n \"L2 Read Misses\" : myArray[69],\n \"L2 Write Hits\" : myArray[70],\n \"L2 Write Misses\" : myArray[71],\n \"L2 Miss Rate\" : myArray[72], \n \n \"Total Bus Traffic\" : myArray[73], \n \"Bus Read Hits\" : myArray[74],\n \"Bus Read Misses\" : myArray[75], \n \"Bus Write Hits\" : myArray[76], \n \"Bus Write Misses\" : myArray[77], \n \"Cache-to-Cache\" : myArray[78], \n \"Reads From Memory\" : myArray[79]\n }, \n {\n \"Core\" : myArray[80], //core 3\n \"Total Instructions\" : myArray[81], \n \"Total Read Instructions\" : myArray[82],\n \"Total Write Instructions\" : myArray[83], \n \"L1 Read Hits\" : myArray[84], \n \"L1 Read Misses\" : myArray[85], \n \"L1 Write Hits\" : myArray[86], \n \"L1 Write Misses\" : myArray[87],\n \"L1 Miss Rate\" : myArray[88],\n \n \"Victim Lines\" : myArray[89], \n \"Victim Cache Calls\" : myArray[90], \n \"Victim Cache Hits\" : myArray[91], \n \"Victim Cache Misses\" : myArray[92], \n \"Victim Cache Read Hits\" : myArray[93], \n \"Victim Cache Read Misses\" : myArray[94], \n \"Victim Cache Write Hits\" : myArray[95], \n \"Victim Cache Write Misses\" : myArray[96],\n \"Victim Cache Hit Rate\" : myArray[97], \n \n \"Total L2 Calls\" :myArray[98], \n \"L2 Read Calls\" : myArray[99],\n \"L2 Write Calls\" : myArray[100],\n \"L2 Read Hits\" : myArray[101],\n \"L2 Read Misses\" : myArray[102],\n \"L2 Write Hits\" : myArray[103],\n \"L2 Write Misses\" : myArray[104],\n \"L2 Miss Rate\" : myArray[105], \n \n \"Total Bus Traffic\" : myArray[106], \n \"Bus Read Hits\" : myArray[107],\n \"Bus Read Misses\" : myArray[108], \n \"Bus Write Hits\" : myArray[109], \n \"Bus Write Misses\" : myArray[110], \n \"Cache-to-Cache\" : myArray[111], \n \"Reads From Memory\" : myArray[112]\n }, \n {\n \"Core\" : myArray[113], //core 4\n \"Total Instructions\" : myArray[114], \n \"Total Read Instructions\" : myArray[115],\n \"Total Write Instructions\" : myArray[116], \n \"L1 Read Hits\" : myArray[117], \n \"L1 Read Misses\" : myArray[118], \n \"L1 Write Hits\" : myArray[119], \n \"L1 Write Misses\" : myArray[120],\n \"L1 Miss Rate\" : myArray[121],\n \n \"Victim Lines\" : myArray[122], \n \"Victim Cache Calls\" : myArray[123], \n \"Victim Cache Hits\" : myArray[124], \n \"Victim Cache Misses\" : myArray[125], \n \"Victim Cache Read Hits\" : myArray[126], \n \"Victim Cache Read Misses\" : myArray[127], \n \"Victim Cache Write Hits\" : myArray[128], \n \"Victim Cache Write Misses\" : myArray[129],\n \"Victim Cache Hit Rate\" : myArray[130], \n \n \"Total L2 Calls\" :myArray[131], \n \"L2 Read Calls\" : myArray[132],\n \"L2 Write Calls\" : myArray[133],\n \"L2 Read Hits\" : myArray[134],\n \"L2 Read Misses\" : myArray[135],\n \"L2 Write Hits\" : myArray[136],\n \"L2 Write Misses\" : myArray[137],\n \"L2 Miss Rate\" : myArray[138], \n \n \"Total Bus Traffic\" : myArray[139], \n \"Bus Read Hits\" : myArray[140],\n \"Bus Read Misses\" : myArray[141], \n \"Bus Write Hits\" : myArray[142], \n \"Bus Write Misses\" : myArray[143], \n \"Cache-to-Cache\" : myArray[144], \n \"Reads From Memory\" : myArray[145]\n }, \n {\n \"Core\" : myArray[146], //core 5\n \"Total Instructions\" : myArray[147], \n \"Total Read Instructions\" : myArray[148],\n \"Total Write Instructions\" : myArray[149], \n \"L1 Read Hits\" : myArray[150], \n \"L1 Read Misses\" : myArray[151], \n \"L1 Write Hits\" : myArray[152], \n \"L1 Write Misses\" : myArray[153],\n \"L1 Miss Rate\" :

myArray[154],\n \n \"Victim Lines\" : myArray[155], \n \"Victim Cache Calls\" : myArray[156], \n \"Victim Cache Hits\" : myArray[157], \n \"Victim Cache Misses\" : myArray[158], \n \"Victim Cache Read Hits\" : myArray[159], \n \"Victim Cache Read Misses\" : myArray[160], \n \"Victim Cache Write Hits\" : myArray[161], \n \"Victim Cache Write Misses\" : myArray[162],\n \"Victim Cache Hit Rate\" : myArray[163], \n \n \"Total L2 Calls\" :myArray[164], \n \"L2 Read Calls\" : myArray[165],\n \"L2 Write Calls\" : myArray[166],\n \"L2 Read Hits\" : myArray[167],\n \"L2 Read Misses\" : myArray[168],\n \"L2 Write Hits\" : myArray[169],\n \"L2 Write Misses\" : myArray[170],\n \"L2 Miss Rate\" : myArray[171], \n \n \"Total Bus Traffic\" : myArray[172], \n \"Bus Read Hits\" : myArray[173],\n \"Bus Read Misses\" : myArray[174], \n \"Bus Write Hits\" : myArray[175], \n \"Bus Write Misses\" : myArray[176], \n \"Cache-to-Cache\" : myArray[177], \n \"Reads From Memory\" : myArray[178]\n }, \n {\n \"Core\" : myArray[179], //core 6\n \"Total Instructions\" : myArray[180], \n \"Total Read Instructions\" : myArray[181],\n \"Total Write Instructions\" : myArray[182], \n \"L1 Read Hits\" : myArray[183], \n \"L1 Read Misses\" : myArray[184], \n \"L1 Write Hits\" : myArray[185], \n \"L1 Write Misses\" : myArray[186],\n \"L1 Miss Rate\" : myArray[187],\n \n \"Victim Lines\" : myArray[188], \n \"Victim Cache Calls\" : myArray[189], \n \"Victim Cache Hits\" : myArray[190], \n \"Victim Cache Misses\" : myArray[191], \n \"Victim Cache Read Hits\" : myArray[192], \n \"Victim Cache Read Misses\" : myArray[193], \n \"Victim Cache Write Hits\" : myArray[194], \n \"Victim Cache Write Misses\" : myArray[195],\n \"Victim Cache Hit Rate\" : myArray[196], \n \n \"Total L2 Calls\" :myArray[197], \n \"L2 Read Calls\" : myArray[198],\n \"L2 Write Calls\" : myArray[199],\n \"L2 Read Hits\" : myArray[200],\n \"L2 Read Misses\" : myArray[201],\n \"L2 Write Hits\" : myArray[202],\n \"L2 Write Misses\" : myArray[203],\n \"L2 Miss Rate\" : myArray[204], \n \n \"Total Bus Traffic\" : myArray[205], \n \"Bus Read Hits\" : myArray[206],\n \"Bus Read Misses\" : myArray[207], \n \"Bus Write Hits\" : myArray[208], \n \"Bus Write Misses\" : myArray[209], \n \"Cache-to-Cache\" : myArray[210], \n \"Reads From Memory\" : myArray[211]\n }, \n {\n \"Core\" : myArray[212], //core 7\n \"Total Instructions\" : myArray[213], \n \"Total Read Instructions\" : myArray[214],\n \"Total Write Instructions\" : myArray[215], \n \"L1 Read Hits\" : myArray[216], \n \"L1 Read Misses\" : myArray[217], \n \"L1 Write Hits\" : myArray[218], \n \"L1 Write Misses\" : myArray[219],\n \"L1 Miss Rate\" : myArray[220],\n \n \"Victim Lines\" : myArray[221], \n \"Victim Cache Calls\" : myArray[222], \n \"Victim Cache Hits\" : myArray[223], \n \"Victim Cache Misses\" : myArray[224], \n \"Victim Cache Read Hits\" : myArray[225], \n \"Victim Cache Read Misses\" : myArray[226], \n \"Victim Cache Write Hits\" : myArray[227], \n \"Victim Cache Write Misses\" : myArray[228],\n \"Victim Cache Hit Rate\" : myArray[229], \n \n \"Total L2 Calls\" :myArray[230], \n \"L2 Read Calls\" : myArray[231],\n \"L2 Write Calls\" : myArray[232],\n \"L2 Read Hits\" : myArray[233],\n \"L2 Read Misses\" : myArray[234],\n \"L2 Write Hits\" : myArray[235],\n \"L2 Write Misses\" : myArray[236],\n \"L2 Miss Rate\" : myArray[237], \n \n \"Total Bus Traffic\" : myArray[238], \n \"Bus Read Hits\" : myArray[239],\n \"Bus Read Misses\" : myArray[240], \n \"Bus Write

Hits\" : myArray[241], \n \"Bus Write Misses\" : myArray[242], \n \"Cache-to-Cache\" : myArray[243], \n \"Reads From Memory\" : myArray[244]\n }, \n {\n \"Core\" : myArray[245], //core 8\n \"Total Instructions\" : myArray[246], \n \"Total Read Instructions\" : myArray[247],\n \"Total Write Instructions\" : myArray[248], \n \"L1 Read Hits\" : myArray[249], \n \"L1 Read Misses\" : myArray[250], \n \"L1 Write Hits\" : myArray[251], \n \"L1 Write Misses\" : myArray[252],\n \"L1 Miss Rate\" : myArray[253],\n \n \"Victim Lines\" : myArray[254], \n \"Victim Cache Calls\" : myArray[255], \n \"Victim Cache Hits\" : myArray[256], \n \"Victim Cache Misses\" : myArray[257], \n \"Victim Cache Read Hits\" : myArray[258], \n \"Victim Cache Read Misses\" : myArray[259], \n \"Victim Cache Write Hits\" : myArray[260], \n \"Victim Cache Write Misses\" : myArray[261],\n \"Victim Cache Hit Rate\" : myArray[262], \n \n \"Total L2 Calls\" :myArray[263], \n \"L2 Read Calls\" : myArray[264],\n \"L2 Write Calls\" : myArray[265],\n \"L2 Read Hits\" : myArray[266],\n \"L2 Read Misses\" : myArray[267],\n \"L2 Write Hits\" : myArray[268],\n \"L2 Write Misses\" : myArray[269],\n \"L2 Miss Rate\" : myArray[270], \n \n \"Total Bus Traffic\" : myArray[271], \n \"Bus Read Hits\" : myArray[272],\n \"Bus Read Misses\" : myArray[273], \n \"Bus Write Hits\" : myArray[274], \n \"Bus Write Misses\" : myArray[275], \n \"Cache-to-Cache\" : myArray[276], \n \"Reads From Memory\" : myArray[277]\n }];\n return newMsg;\n }\n}\nelse if(coreCount == 12) //12C\n{\n if(cacheSizeL2 == 0) //12C, L1 only\n {\n let newMsg = {payload : \"\"}\n \n newMsg.payload = [\n {\n \"Core\" : \"Total\",\n \"Core Count\" : myArray[0],\n \"L1 Cache Size\" : myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4], \n \"Coherence Protocol\" : myArray[5],\n \"Write Policy\" : myArray[6],\n \"Total Instructions\" : myArray[7],\n \"Total Read Instructions\" : myArray[8],\n \"Total Write Instructions\" : myArray[9],\n \n \"Cache-to-Cache\" : myArray[310],\n \"Total Bus Traffic\" : myArray[311], \n \"Reads From Memory\" : myArray[312],\n \"Global Miss Rate\" : myArray[313],\n \"L1 Read Hits\" : myArray[314],\n \"L1 Read Misses\" : myArray[315],\n \"L1 Write Hits\" : myArray[316],\n \"L1 Write Misses\" : myArray[317],\n \n \"Bus Read Hits\" : myArray[318], \n \"Bus Read Misses\" : myArray[319], \n \"Bus Write Hits\" : myArray[320], \n \"Bus Write Misses\" : myArray[321],\n \n \"Victim Lines\" : myArray[322],\n \"Victim Cache Calls\" : myArray[323],\n \"Victim Cache Hits\" : myArray[324],\n \"Victim Cache Misses\" : myArray[325], \n \"Victim Cache Read Hits\" : myArray[326],\n \"Victim Cache Read Misses\" : myArray[327],\n \"Victim Cache Write Hits\" : myArray[328],\n \"Victim Cache Write Misses\" : myArray[329],\n \n \"L2 Cache Size\" : \"N/A\",\n \"L2 Block Size\" : \"N/A\",\n \"L2 Associativity\" : \"N/A\", \n \"L2 Replacement Policy\" : \"N/A\", \n \"Total L2 Calls\" : \"N/A\",\n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \n \"L1 Miss Rate\" : myArray[330], \n \"Victim Cache Hit Rate\" : myArray[331], \n \"L2 Miss Rate\" : \"N/A\", \n \"Trace File\" : traceFile\n }, \n {\n \"Core\" : myArray[10], //core 1\n \"Total Instructions\" : myArray[11], \n \"Total Read Instructions\" : myArray[12],\n \"Total Write Instructions\" : myArray[13],

\n \"L1 Read Hits\" : myArray[14], \n \"L1 Read Misses\" : myArray[15], \n \"L1 Write Hits\" : myArray[16], \n \"L1 Write Misses\" : myArray[17],\n \"Total Bus Traffic\" : myArray[18], \n \"Bus Read Hits\" : myArray[19],\n \"Bus Read Misses\" : myArray[20], \n \"Bus Write Hits\" : myArray[21], \n \"Bus Write Misses\" : myArray[22], \n \"Cache-to-Cache\" : myArray[23], \n \"L1 Miss Rate\" : myArray[24], \n \n \"Victim Lines\" : myArray[25], \n \"Victim Cache Calls\" : myArray[26], \n \"Victim Cache Hits\" : myArray[27], \n \"Victim Cache Misses\" : myArray[28], \n \"Victim Cache Read Hits\" : myArray[29], \n \"Victim Cache Read Misses\" : myArray[30], \n \"Victim Cache Write Hits\" : myArray[31], \n \"Victim Cache Write Misses\" : myArray[32],\n \"Victim Cache Hit Rate\" : myArray[33], \n \"Reads From Memory\" : myArray[34],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[35], //core 2\n \"Total Instructions\" : myArray[36], \n \"Total Read Instructions\" : myArray[37],\n \"Total Write Instructions\" : myArray[38], \n \"L1 Read Hits\" : myArray[39], \n \"L1 Read Misses\" : myArray[40], \n \"L1 Write Hits\" : myArray[41], \n \"L1 Write Misses\" : myArray[42],\n \"Total Bus Traffic\" : myArray[43], \n \"Bus Read Hits\" : myArray[44],\n \"Bus Read Misses\" : myArray[45], \n \"Bus Write Hits\" : myArray[46], \n \"Bus Write Misses\" : myArray[47],\n \"Cache-to-Cache\" : myArray[48], \n \"L1 Miss Rate\" : myArray[49], \n \n \"Victim Lines\" : myArray[50], \n \"Victim Cache Calls\" : myArray[51], \n \"Victim Cache Hits\" : myArray[52], \n \"Victim Cache Misses\" : myArray[53], \n \"Victim Cache Read Hits\" : myArray[54], \n \"Victim Cache Read Misses\" : myArray[55], \n \"Victim Cache Write Hits\" : myArray[56], \n \"Victim Cache Write Misses\" : myArray[57],\n \"Victim Cache Hit Rate\" : myArray[58], \n \"Reads From Memory\" : myArray[59],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[60], //core 3\n \"Total Instructions\" : myArray[61], \n \"Total Read Instructions\" : myArray[62],\n \"Total Write Instructions\" : myArray[63], \n \"L1 Read Hits\" : myArray[64], \n \"L1 Read Misses\" : myArray[65], \n \"L1 Write Hits\" : myArray[66], \n \"L1 Write Misses\" : myArray[67],\n \"Total Bus Traffic\" : myArray[68], \n \"Bus Read Hits\" : myArray[69],\n \"Bus Read Misses\" : myArray[70], \n \"Bus Write Hits\" : myArray[71], \n \"Bus Write Misses\" : myArray[72], \n \"Cache-to-Cache\" : myArray[73], \n \"L1 Miss Rate\" : myArray[74], \n \n \"Victim Lines\" : myArray[75], \n \"Victim Cache Calls\" : myArray[76], \n \"Victim Cache Hits\" : myArray[77], \n \"Victim Cache Misses\" : myArray[78], \n \"Victim Cache Read Hits\" : myArray[79], \n \"Victim Cache Read Misses\" : myArray[80], \n \"Victim Cache Write Hits\" : myArray[81], \n \"Victim Cache Write Misses\" : myArray[82],\n \"Victim Cache Hit Rate\" : myArray[83], \n \"Reads From Memory\" : myArray[84],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\"

: \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[85], //core 4\n \"Total Instructions\" : myArray[86], \n \"Total Read Instructions\" : myArray[87],\n \"Total Write Instructions\" : myArray[88], \n \"L1 Read Hits\" : myArray[89], \n \"L1 Read Misses\" : myArray[90], \n \"L1 Write Hits\" : myArray[91], \n \"L1 Write Misses\" : myArray[92],\n \"Total Bus Traffic\" : myArray[93], \n \"Bus Read Hits\" : myArray[94],\n \"Bus Read Misses\" : myArray[95], \n \"Bus Write Hits\" : myArray[96], \n \"Bus Write Misses\" : myArray[97],\n \"Cache-to-Cache\" : myArray[98], \n \"L1 Miss Rate\" : myArray[99], \n \n \"Victim Lines\" : myArray[100], \n \"Victim Cache Calls\" : myArray[101], \n \"Victim Cache Hits\" : myArray[102], \n \"Victim Cache Misses\" : myArray[103], \n \"Victim Cache Read Hits\" : myArray[104], \n \"Victim Cache Read Misses\" : myArray[105], \n \"Victim Cache Write Hits\" : myArray[106], \n \"Victim Cache Write Misses\" : myArray[107],\n \"Victim Cache Hit Rate\" : myArray[108], \n \"Reads From Memory\" : myArray[109],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[110], //core 5\n \"Total Instructions\" : myArray[111], \n \"Total Read Instructions\" : myArray[112],\n \"Total Write Instructions\" : myArray[113], \n \"L1 Read Hits\" : myArray[114], \n \"L1 Read Misses\" : myArray[115], \n \"L1 Write Hits\" : myArray[116], \n \"L1 Write Misses\" : myArray[117],\n \"Total Bus Traffic\" : myArray[118], \n \"Bus Read Hits\" : myArray[119],\n \"Bus Read Misses\" : myArray[120], \n \"Bus Write Hits\" : myArray[121], \n \"Bus Write Misses\" : myArray[122], \n \"Cache-to-Cache\" : myArray[123], \n \"L1 Miss Rate\" : myArray[124], \n \n \"Victim Lines\" : myArray[125], \n \"Victim Cache Calls\" : myArray[126], \n \"Victim Cache Hits\" : myArray[127], \n \"Victim Cache Misses\" : myArray[128], \n \"Victim Cache Read Hits\" : myArray[129], \n \"Victim Cache Read Misses\" : myArray[130], \n \"Victim Cache Write Hits\" : myArray[131], \n \"Victim Cache Write Misses\" : myArray[132],\n \"Victim Cache Hit Rate\" : myArray[133], \n \"Reads From Memory\" : myArray[134],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[135], //core 6\n \"Total Instructions\" : myArray[136], \n \"Total Read Instructions\" : myArray[137],\n \"Total Write Instructions\" : myArray[138], \n \"L1 Read Hits\" : myArray[139], \n \"L1 Read Misses\" : myArray[140], \n \"L1 Write Hits\" : myArray[141], \n \"L1 Write Misses\" : myArray[142],\n \"Total Bus Traffic\" : myArray[143], \n \"Bus Read Hits\" : myArray[144],\n \"Bus Read Misses\" : myArray[145], \n \"Bus Write Hits\" : myArray[146], \n \"Bus Write Misses\" : myArray[147],\n \"Cache-to-Cache\" : myArray[148], \n \"L1 Miss Rate\" : myArray[149], \n \n \"Victim Lines\" : myArray[150], \n \"Victim Cache Calls\" : myArray[151], \n \"Victim Cache Hits\" : myArray[152], \n \"Victim Cache Misses\" : myArray[153], \n \"Victim Cache Read Hits\" : myArray[154], \n \"Victim Cache Read Misses\" : myArray[155], \n \"Victim Cache Write Hits\" : myArray[156], \n \"Victim Cache Write Misses\" :

myArray[157],\n \"Victim Cache Hit Rate\" : myArray[158], \n \"Reads From Memory\" : myArray[159],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[160], //core 7\n \"Total Instructions\" : myArray[161], \n \"Total Read Instructions\" : myArray[162],\n \"Total Write Instructions\" : myArray[163], \n \"L1 Read Hits\" : myArray[164], \n \"L1 Read Misses\" : myArray[165], \n \"L1 Write Hits\" : myArray[166], \n \"L1 Write Misses\" : myArray[167],\n \"Total Bus Traffic\" : myArray[168], \n \"Bus Read Hits\" : myArray[169],\n \"Bus Read Misses\" : myArray[170], \n \"Bus Write Hits\" : myArray[171], \n \"Bus Write Misses\" : myArray[172], \n \"Cache-to-Cache\" : myArray[173], \n \"L1 Miss Rate\" : myArray[174], \n \n \"Victim Lines\" : myArray[175], \n \"Victim Cache Calls\" : myArray[176], \n \"Victim Cache Hits\" : myArray[177], \n \"Victim Cache Misses\" : myArray[178], \n \"Victim Cache Read Hits\" : myArray[179], \n \"Victim Cache Read Misses\" : myArray[180], \n \"Victim Cache Write Hits\" : myArray[181], \n \"Victim Cache Write Misses\" : myArray[182],\n \"Victim Cache Hit Rate\" : myArray[183], \n \"Reads From Memory\" : myArray[184],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[185], //core 8\n \"Total Instructions\" : myArray[186], \n \"Total Read Instructions\" : myArray[187],\n \"Total Write Instructions\" : myArray[188], \n \"L1 Read Hits\" : myArray[189], \n \"L1 Read Misses\" : myArray[190], \n \"L1 Write Hits\" : myArray[191], \n \"L1 Write Misses\" : myArray[192],\n \"Total Bus Traffic\" : myArray[193], \n \"Bus Read Hits\" : myArray[194],\n \"Bus Read Misses\" : myArray[195], \n \"Bus Write Hits\" : myArray[196], \n \"Bus Write Misses\" : myArray[197],\n \"Cache-to-Cache\" : myArray[198], \n \"L1 Miss Rate\" : myArray[199], \n \n \"Victim Lines\" : myArray[200], \n \"Victim Cache Calls\" : myArray[201], \n \"Victim Cache Hits\" : myArray[202], \n \"Victim Cache Misses\" : myArray[203], \n \"Victim Cache Read Hits\" : myArray[204], \n \"Victim Cache Read Misses\" : myArray[205], \n \"Victim Cache Write Hits\" : myArray[206], \n \"Victim Cache Write Misses\" : myArray[207],\n \"Victim Cache Hit Rate\" : myArray[208], \n \"Reads From Memory\" : myArray[209],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[210], //core 9\n \"Total Instructions\" : myArray[211], \n \"Total Read Instructions\" : myArray[212],\n \"Total Write Instructions\" : myArray[213], \n \"L1 Read Hits\" : myArray[214], \n \"L1 Read Misses\" : myArray[215], \n \"L1 Write Hits\" : myArray[216], \n \"L1 Write Misses\" : myArray[217],\n \"Total Bus Traffic\" : myArray[218], \n \"Bus Read Hits\" : myArray[219],\n \"Bus Read Misses\" : myArray[220], \n \"Bus Write Hits\" : myArray[221], \n \"Bus Write Misses\" : myArray[222], \n \"Cache-to-Cache\" : myArray[223], \n \"L1 Miss Rate\" : myArray[224], \n \n \"Victim Lines\" :

myArray[225], \n \"Victim Cache Calls\" : myArray[226], \n \"Victim Cache Hits\" : myArray[227], \n \"Victim Cache Misses\" : myArray[228], \n \"Victim Cache Read Hits\" : myArray[229], \n \"Victim Cache Read Misses\" : myArray[230], \n \"Victim Cache Write Hits\" : myArray[231], \n \"Victim Cache Write Misses\" : myArray[232],\n \"Victim Cache Hit Rate\" : myArray[233], \n \"Reads From Memory\" : myArray[234],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[235], //core 10\n \"Total Instructions\" : myArray[236], \n \"Total Read Instructions\" : myArray[237],\n \"Total Write Instructions\" : myArray[238], \n \"L1 Read Hits\" : myArray[239], \n \"L1 Read Misses\" : myArray[240], \n \"L1 Write Hits\" : myArray[241], \n \"L1 Write Misses\" : myArray[242],\n \"Total Bus Traffic\" : myArray[243], \n \"Bus Read Hits\" : myArray[244],\n \"Bus Read Misses\" : myArray[245], \n \"Bus Write Hits\" : myArray[246], \n \"Bus Write Misses\" : myArray[247],\n \"Cache-to-Cache\" : myArray[248], \n \"L1 Miss Rate\" : myArray[249], \n \n \"Victim Lines\" : myArray[250], \n \"Victim Cache Calls\" : myArray[251], \n \"Victim Cache Hits\" : myArray[252], \n \"Victim Cache Misses\" : myArray[253], \n \"Victim Cache Read Hits\" : myArray[254], \n \"Victim Cache Read Misses\" : myArray[255], \n \"Victim Cache Write Hits\" : myArray[256], \n \"Victim Cache Write Misses\" : myArray[257],\n \"Victim Cache Hit Rate\" : myArray[258], \n \"Reads From Memory\" : myArray[259],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[260], //core 11\n \"Total Instructions\" : myArray[261], \n \"Total Read Instructions\" : myArray[262],\n \"Total Write Instructions\" : myArray[263], \n \"L1 Read Hits\" : myArray[264], \n \"L1 Read Misses\" : myArray[265], \n \"L1 Write Hits\" : myArray[266], \n \"L1 Write Misses\" : myArray[267],\n \"Total Bus Traffic\" : myArray[268], \n \"Bus Read Hits\" : myArray[269],\n \"Bus Read Misses\" : myArray[270], \n \"Bus Write Hits\" : myArray[271], \n \"Bus Write Misses\" : myArray[272], \n \"Cache-to-Cache\" : myArray[273], \n \"L1 Miss Rate\" : myArray[274], \n \n \"Victim Lines\" : myArray[275], \n \"Victim Cache Calls\" : myArray[276], \n \"Victim Cache Hits\" : myArray[277], \n \"Victim Cache Misses\" : myArray[278], \n \"Victim Cache Read Hits\" : myArray[279], \n \"Victim Cache Read Misses\" : myArray[280], \n \"Victim Cache Write Hits\" : myArray[281], \n \"Victim Cache Write Misses\" : myArray[282],\n \"Victim Cache Hit Rate\" : myArray[283], \n \"Reads From Memory\" : myArray[284],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[285], //core 12\n \"Total Instructions\" : myArray[286], \n \"Total Read Instructions\" : myArray[287],\n \"Total Write Instructions\" : myArray[288], \n \"L1 Read Hits\" : myArray[289], \n \"L1 Read Misses\" : myArray[290], \n \"L1 Write Hits\" : myArray[291], \n \"L1

Write Misses\" : myArray[292],\n \"Total Bus Traffic\" : myArray[293], \n \"Bus Read Hits\" : myArray[294],\n \"Bus Read Misses\" : myArray[295], \n \"Bus Write Hits\" : myArray[296], \n \"Bus Write Misses\" : myArray[297],\n \"Cache-to-Cache\" : myArray[298], \n \"L1 Miss Rate\" : myArray[299], \n \n \"Victim Lines\" : myArray[300], \n \"Victim Cache Calls\" : myArray[301], \n \"Victim Cache Hits\" : myArray[302], \n \"Victim Cache Misses\" : myArray[303], \n \"Victim Cache Read Hits\" : myArray[304], \n \"Victim Cache Read Misses\" : myArray[305], \n \"Victim Cache Write Hits\" : myArray[306], \n \"Victim Cache Write Misses\" : myArray[307],\n \"Victim Cache Hit Rate\" : myArray[308], \n \"Reads From Memory\" : myArray[309],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }];\n return newMsg;\n }\n else //12C, L1 and L2\n {\n let newMsg = {payload : \"\"}\n \n newMsg.payload = [\n {\n \"Core\" : \"Total\",\n \"Core Count\" : myArray[0],\n \"L1 Cache Size\" : myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4],\n \"L2 Cache Size\" : myArray[5],\n \"L2 Block Size\" : myArray[6],\n \"L2 Associativity\" : myArray[7], \n \"L2 Replacement Policy\" : myArray[8], \n \"Coherence Protocol\" : myArray[9],\n \"Write Policy\" : myArray[10],\n \"Total Instructions\" : myArray[11],\n \"Total Read Instructions\" : myArray[12],\n \"Total Write Instructions\" : myArray[13],\n \n \"Cache-to-Cache\" : myArray[410],\n \"Total Bus Traffic\" : myArray[411], \n \"Reads From Memory\" : myArray[412],\n \"Global Miss Rate\" : myArray[413],\n \"L1 Read Hits\" : myArray[414],\n \"L1 Read Misses\" : myArray[415],\n \"L1 Write Hits\" : myArray[416],\n \"L1 Write Misses\" : myArray[417],\n \n \"Bus Read Hits\" : myArray[418], \n \"Bus Read Misses\" : myArray[419], \n \"Bus Write Hits\" : myArray[420], \n \"Bus Write Misses\" : myArray[421],\n \n \"Total L2 Calls\" : myArray[422],\n \"L2 Read Calls\" : myArray[423],\n \"L2 Write Calls\" : myArray[424],\n \"L2 Read Hits\" : myArray[425],\n \"L2 Read Misses\" : myArray[426],\n \"L2 Write Hits\" : myArray[427],\n \"L2 Write Misses\" : myArray[428],\n \n \"Victim Lines\" : myArray[429],\n \"Victim Cache Calls\" : myArray[430],\n \"Victim Cache Hits\" : myArray[431],\n \"Victim Cache Misses\" : myArray[432], \n \"Victim Cache Read Hits\" : myArray[433],\n \"Victim Cache Read Misses\" : myArray[434],\n \"Victim Cache Write Hits\" : myArray[435],\n \"Victim Cache Write Misses\" : myArray[436], \n \n \"L1 Miss Rate\" : myArray[437], \n \"Victim Cache Hit Rate\" : myArray[438], \n \"L2 Miss Rate\" : myArray[439], \n \"Trace File\" : traceFile\n }, \n {\n \"Core\" : myArray[14], //core 1\n \"Total Instructions\" : myArray[15], \n \"Total Read Instructions\" : myArray[16],\n \"Total Write Instructions\" : myArray[17], \n \"L1 Read Hits\" : myArray[18], \n \"L1 Read Misses\" : myArray[19], \n \"L1 Write Hits\" : myArray[20], \n \"L1 Write Misses\" : myArray[21],\n \"L1 Miss Rate\" : myArray[22],\n \n \"Victim Lines\" : myArray[23], \n \"Victim Cache Calls\" : myArray[24], \n \"Victim Cache Hits\" : myArray[25], \n \"Victim Cache Misses\" : myArray[26], \n \"Victim Cache Read Hits\" : myArray[27], \n \"Victim Cache Read Misses\" : myArray[28], \n \"Victim Cache Write Hits\" :

myArray[29], \n \"Victim Cache Write Misses\" : myArray[30],\n \"Victim Cache Hit Rate\" : myArray[31], \n \n \"Total L2 Calls\" :myArray[32], \n \"L2 Read Calls\" : myArray[33],\n \"L2 Write Calls\" : myArray[34],\n \"L2 Read Hits\" : myArray[35],\n \"L2 Read Misses\" : myArray[36],\n \"L2 Write Hits\" : myArray[37],\n \"L2 Write Misses\" : myArray[38],\n \"L2 Miss Rate\" : myArray[39], \n \n \"Total Bus Traffic\" : myArray[40], \n \"Bus Read Hits\" : myArray[41],\n \"Bus Read Misses\" : myArray[42], \n \"Bus Write Hits\" : myArray[43], \n \"Bus Write Misses\" : myArray[44], \n \"Cache-to-Cache\" : myArray[45], \n \"Reads From Memory\" : myArray[46]\n }, \n {\n \"Core\" : myArray[47], //core 2\n \"Total Instructions\" : myArray[48], \n \"Total Read Instructions\" : myArray[49],\n \"Total Write Instructions\" : myArray[50], \n \"L1 Read Hits\" : myArray[51], \n \"L1 Read Misses\" : myArray[52], \n \"L1 Write Hits\" : myArray[53], \n \"L1 Write Misses\" : myArray[54],\n \"L1 Miss Rate\" : myArray[55],\n \n \"Victim Lines\" : myArray[56], \n \"Victim Cache Calls\" : myArray[57], \n \"Victim Cache Hits\" : myArray[58], \n \"Victim Cache Misses\" : myArray[59], \n \"Victim Cache Read Hits\" : myArray[60], \n \"Victim Cache Read Misses\" : myArray[61], \n \"Victim Cache Write Hits\" : myArray[62], \n \"Victim Cache Write Misses\" : myArray[63],\n \"Victim Cache Hit Rate\" : myArray[64], \n \n \"Total L2 Calls\" :myArray[65], \n \"L2 Read Calls\" : myArray[66],\n \"L2 Write Calls\" : myArray[67],\n \"L2 Read Hits\" : myArray[68],\n \"L2 Read Misses\" : myArray[69],\n \"L2 Write Hits\" : myArray[70],\n \"L2 Write Misses\" : myArray[71],\n \"L2 Miss Rate\" : myArray[72], \n \n \"Total Bus Traffic\" : myArray[73], \n \"Bus Read Hits\" : myArray[74],\n \"Bus Read Misses\" : myArray[75], \n \"Bus Write Hits\" : myArray[76], \n \"Bus Write Misses\" : myArray[77], \n \"Cache-to-Cache\" : myArray[78], \n \"Reads From Memory\" : myArray[79]\n }, \n {\n \"Core\" : myArray[80], //core 3\n \"Total Instructions\" : myArray[81], \n \"Total Read Instructions\" : myArray[82],\n \"Total Write Instructions\" : myArray[83], \n \"L1 Read Hits\" : myArray[84], \n \"L1 Read Misses\" : myArray[85], \n \"L1 Write Hits\" : myArray[86], \n \"L1 Write Misses\" : myArray[87],\n \"L1 Miss Rate\" : myArray[88],\n \n \"Victim Lines\" : myArray[89], \n \"Victim Cache Calls\" : myArray[90], \n \"Victim Cache Hits\" : myArray[91], \n \"Victim Cache Misses\" : myArray[92], \n \"Victim Cache Read Hits\" : myArray[93], \n \"Victim Cache Read Misses\" : myArray[94], \n \"Victim Cache Write Hits\" : myArray[95], \n \"Victim Cache Write Misses\" : myArray[96],\n \"Victim Cache Hit Rate\" : myArray[97], \n \n \"Total L2 Calls\" :myArray[98], \n \"L2 Read Calls\" : myArray[99],\n \"L2 Write Calls\" : myArray[100],\n \"L2 Read Hits\" : myArray[101],\n \"L2 Read Misses\" : myArray[102],\n \"L2 Write Hits\" : myArray[103],\n \"L2 Write Misses\" : myArray[104],\n \"L2 Miss Rate\" : myArray[105], \n \n \"Total Bus Traffic\" : myArray[106], \n \"Bus Read Hits\" : myArray[107],\n \"Bus Read Misses\" : myArray[108], \n \"Bus Write Hits\" : myArray[109], \n \"Bus Write Misses\" : myArray[110], \n \"Cache-to-Cache\" : myArray[111], \n \"Reads From Memory\" : myArray[112]\n }, \n {\n \"Core\" : myArray[113], //core 4\n \"Total Instructions\" : myArray[114], \n \"Total Read Instructions\" : myArray[115],\n \"Total Write Instructions\" : myArray[116], \n \"L1 Read Hits\" : myArray[117], \n \"L1 Read Misses\" :

myArray[118], \n \"L1 Write Hits\" : myArray[119], \n \"L1 Write Misses\" : myArray[120],\n \"L1 Miss Rate\" : myArray[121],\n \n \"Victim Lines\" : myArray[122], \n \"Victim Cache Calls\" : myArray[123], \n \"Victim Cache Hits\" : myArray[124], \n \"Victim Cache Misses\" : myArray[125], \n \"Victim Cache Read Hits\" : myArray[126], \n \"Victim Cache Read Misses\" : myArray[127], \n \"Victim Cache Write Hits\" : myArray[128], \n \"Victim Cache Write Misses\" : myArray[129],\n \"Victim Cache Hit Rate\" : myArray[130], \n \n \"Total L2 Calls\" :myArray[131], \n \"L2 Read Calls\" : myArray[132],\n \"L2 Write Calls\" : myArray[133],\n \"L2 Read Hits\" : myArray[134],\n \"L2 Read Misses\" : myArray[135],\n \"L2 Write Hits\" : myArray[136],\n \"L2 Write Misses\" : myArray[137],\n \"L2 Miss Rate\" : myArray[138], \n \n \"Total Bus Traffic\" : myArray[139], \n \"Bus Read Hits\" : myArray[140],\n \"Bus Read Misses\" : myArray[141], \n \"Bus Write Hits\" : myArray[142], \n \"Bus Write Misses\" : myArray[143], \n \"Cache-to-Cache\" : myArray[144], \n \"Reads From Memory\" : myArray[145]\n }, \n {\n \"Core\" : myArray[146], //core 5\n \"Total Instructions\" : myArray[147], \n \"Total Read Instructions\" : myArray[148],\n \"Total Write Instructions\" : myArray[149], \n \"L1 Read Hits\" : myArray[150], \n \"L1 Read Misses\" : myArray[151], \n \"L1 Write Hits\" : myArray[152], \n \"L1 Write Misses\" : myArray[153],\n \"L1 Miss Rate\" : myArray[154],\n \n \"Victim Lines\" : myArray[155], \n \"Victim Cache Calls\" : myArray[156], \n \"Victim Cache Hits\" : myArray[157], \n \"Victim Cache Misses\" : myArray[158], \n \"Victim Cache Read Hits\" : myArray[159], \n \"Victim Cache Read Misses\" : myArray[160], \n \"Victim Cache Write Hits\" : myArray[161], \n \"Victim Cache Write Misses\" : myArray[162],\n \"Victim Cache Hit Rate\" : myArray[163], \n \n \"Total L2 Calls\" :myArray[164], \n \"L2 Read Calls\" : myArray[165],\n \"L2 Write Calls\" : myArray[166],\n \"L2 Read Hits\" : myArray[167],\n \"L2 Read Misses\" : myArray[168],\n \"L2 Write Hits\" : myArray[169],\n \"L2 Write Misses\" : myArray[170],\n \"L2 Miss Rate\" : myArray[171], \n \n \"Total Bus Traffic\" : myArray[172], \n \"Bus Read Hits\" : myArray[173],\n \"Bus Read Misses\" : myArray[174], \n \"Bus Write Hits\" : myArray[175], \n \"Bus Write Misses\" : myArray[176], \n \"Cache-to-Cache\" : myArray[177], \n \"Reads From Memory\" : myArray[178]\n }, \n {\n \"Core\" : myArray[179], //core 6\n \"Total Instructions\" : myArray[180], \n \"Total Read Instructions\" : myArray[181],\n \"Total Write Instructions\" : myArray[182], \n \"L1 Read Hits\" : myArray[183], \n \"L1 Read Misses\" : myArray[184], \n \"L1 Write Hits\" : myArray[185], \n \"L1 Write Misses\" : myArray[186],\n \"L1 Miss Rate\" : myArray[187],\n \n \"Victim Lines\" : myArray[188], \n \"Victim Cache Calls\" : myArray[189], \n \"Victim Cache Hits\" : myArray[190], \n \"Victim Cache Misses\" : myArray[191], \n \"Victim Cache Read Hits\" : myArray[192], \n \"Victim Cache Read Misses\" : myArray[193], \n \"Victim Cache Write Hits\" : myArray[194], \n \"Victim Cache Write Misses\" : myArray[195],\n \"Victim Cache Hit Rate\" : myArray[196], \n \n \"Total L2 Calls\" :myArray[197], \n \"L2 Read Calls\" : myArray[198],\n \"L2 Write Calls\" : myArray[199],\n \"L2 Read Hits\" : myArray[200],\n \"L2 Read Misses\" : myArray[201],\n \"L2 Write Hits\" : myArray[202],\n \"L2 Write Misses\" : myArray[203],\n \"L2 Miss Rate\" : myArray[204], \n \n \"Total Bus

Traffic\" : myArray[205], \n \"Bus Read Hits\" : myArray[206],\n \"Bus Read Misses\" : myArray[207], \n \"Bus Write Hits\" : myArray[208], \n \"Bus Write Misses\" : myArray[209], \n \"Cache-to-Cache\" : myArray[210], \n \"Reads From Memory\" : myArray[211]\n }, \n {\n \"Core\" : myArray[212], //core 7\n \"Total Instructions\" : myArray[213], \n \"Total Read Instructions\" : myArray[214],\n \"Total Write Instructions\" : myArray[215], \n \"L1 Read Hits\" : myArray[216], \n \"L1 Read Misses\" : myArray[217], \n \"L1 Write Hits\" : myArray[218], \n \"L1 Write Misses\" : myArray[219],\n \"L1 Miss Rate\" : myArray[220],\n \n \"Victim Lines\" : myArray[221], \n \"Victim Cache Calls\" : myArray[222], \n \"Victim Cache Hits\" : myArray[223], \n \"Victim Cache Misses\" : myArray[224], \n \"Victim Cache Read Hits\" : myArray[225], \n \"Victim Cache Read Misses\" : myArray[226], \n \"Victim Cache Write Hits\" : myArray[227], \n \"Victim Cache Write Misses\" : myArray[228],\n \"Victim Cache Hit Rate\" : myArray[229], \n \n \"Total L2 Calls\" :myArray[230], \n \"L2 Read Calls\" : myArray[231],\n \"L2 Write Calls\" : myArray[232],\n \"L2 Read Hits\" : myArray[233],\n \"L2 Read Misses\" : myArray[234],\n \"L2 Write Hits\" : myArray[235],\n \"L2 Write Misses\" : myArray[236],\n \"L2 Miss Rate\" : myArray[237], \n \n \"Total Bus Traffic\" : myArray[238], \n \"Bus Read Hits\" : myArray[239],\n \"Bus Read Misses\" : myArray[240], \n \"Bus Write Hits\" : myArray[241], \n \"Bus Write Misses\" : myArray[242], \n \"Cache-to-Cache\" : myArray[243], \n \"Reads From Memory\" : myArray[244]\n }, \n {\n \"Core\" : myArray[245], //core 8\n \"Total Instructions\" : myArray[246], \n \"Total Read Instructions\" : myArray[247],\n \"Total Write Instructions\" : myArray[248], \n \"L1 Read Hits\" : myArray[249], \n \"L1 Read Misses\" : myArray[250], \n \"L1 Write Hits\" : myArray[251], \n \"L1 Write Misses\" : myArray[252],\n \"L1 Miss Rate\" : myArray[253],\n \n \"Victim Lines\" : myArray[254], \n \"Victim Cache Calls\" : myArray[255], \n \"Victim Cache Hits\" : myArray[256], \n \"Victim Cache Misses\" : myArray[257], \n \"Victim Cache Read Hits\" : myArray[258], \n \"Victim Cache Read Misses\" : myArray[259], \n \"Victim Cache Write Hits\" : myArray[260], \n \"Victim Cache Write Misses\" : myArray[261],\n \"Victim Cache Hit Rate\" : myArray[262], \n \n \"Total L2 Calls\" :myArray[263], \n \"L2 Read Calls\" : myArray[264],\n \"L2 Write Calls\" : myArray[265],\n \"L2 Read Hits\" : myArray[266],\n \"L2 Read Misses\" : myArray[267],\n \"L2 Write Hits\" : myArray[268],\n \"L2 Write Misses\" : myArray[269],\n \"L2 Miss Rate\" : myArray[270], \n \n \"Total Bus Traffic\" : myArray[271], \n \"Bus Read Hits\" : myArray[272],\n \"Bus Read Misses\" : myArray[273], \n \"Bus Write Hits\" : myArray[274], \n \"Bus Write Misses\" : myArray[275], \n \"Cache-to-Cache\" : myArray[276], \n \"Reads From Memory\" : myArray[277]\n }, \n {\n \"Core\" : myArray[278], //core 9\n \"Total Instructions\" : myArray[279], \n \"Total Read Instructions\" : myArray[280],\n \"Total Write Instructions\" : myArray[281], \n \"L1 Read Hits\" : myArray[282], \n \"L1 Read Misses\" : myArray[283], \n \"L1 Write Hits\" : myArray[284], \n \"L1 Write Misses\" : myArray[285],\n \"L1 Miss Rate\" : myArray[286],\n \n \"Victim Lines\" : myArray[287], \n \"Victim Cache Calls\" : myArray[288], \n \"Victim Cache Hits\" : myArray[289], \n \"Victim Cache Misses\" : myArray[290], \n \"Victim Cache Read Hits\" : myArray[291], \n

\"Victim Cache Read Misses\" : myArray[292], \n \"Victim Cache Write Hits\" : myArray[293], \n \"Victim Cache Write Misses\" : myArray[294],\n \"Victim Cache Hit Rate\" : myArray[295], \n \n \"Total L2 Calls\" :myArray[296], \n \"L2 Read Calls\" : myArray[297],\n \"L2 Write Calls\" : myArray[298],\n \"L2 Read Hits\" : myArray[299],\n \"L2 Read Misses\" : myArray[300],\n \"L2 Write Hits\" : myArray[301],\n \"L2 Write Misses\" : myArray[302],\n \"L2 Miss Rate\" : myArray[303], \n \n \"Total Bus Traffic\" : myArray[304], \n \"Bus Read Hits\" : myArray[305],\n \"Bus Read Misses\" : myArray[306], \n \"Bus Write Hits\" : myArray[307], \n \"Bus Write Misses\" : myArray[308], \n \"Cache-to-Cache\" : myArray[309], \n \"Reads From Memory\" : myArray[310]\n }, \n {\n \"Core\" : myArray[311], //core 10\n \"Total Instructions\" : myArray[312], \n \"Total Read Instructions\" : myArray[313],\n \"Total Write Instructions\" : myArray[314], \n \"L1 Read Hits\" : myArray[315], \n \"L1 Read Misses\" : myArray[316], \n \"L1 Write Hits\" : myArray[317], \n \"L1 Write Misses\" : myArray[318],\n \"L1 Miss Rate\" : myArray[319],\n \n \"Victim Lines\" : myArray[320], \n \"Victim Cache Calls\" : myArray[321], \n \"Victim Cache Hits\" : myArray[322], \n \"Victim Cache Misses\" : myArray[323], \n \"Victim Cache Read Hits\" : myArray[324], \n \"Victim Cache Read Misses\" : myArray[325], \n \"Victim Cache Write Hits\" : myArray[326], \n \"Victim Cache Write Misses\" : myArray[327],\n \"Victim Cache Hit Rate\" : myArray[328], \n \n \"Total L2 Calls\" :myArray[329], \n \"L2 Read Calls\" : myArray[330],\n \"L2 Write Calls\" : myArray[331],\n \"L2 Read Hits\" : myArray[332],\n \"L2 Read Misses\" : myArray[333],\n \"L2 Write Hits\" : myArray[334],\n \"L2 Write Misses\" : myArray[335],\n \"L2 Miss Rate\" : myArray[336], \n \n \"Total Bus Traffic\" : myArray[337], \n \"Bus Read Hits\" : myArray[338],\n \"Bus Read Misses\" : myArray[339], \n \"Bus Write Hits\" : myArray[340], \n \"Bus Write Misses\" : myArray[341], \n \"Cache-to-Cache\" : myArray[342], \n \"Reads From Memory\" : myArray[343]\n },\n {\n \"Core\" : myArray[344], //core 11\n \"Total Instructions\" : myArray[345], \n \"Total Read Instructions\" : myArray[346],\n \"Total Write Instructions\" : myArray[347], \n \"L1 Read Hits\" : myArray[348], \n \"L1 Read Misses\" : myArray[349], \n \"L1 Write Hits\" : myArray[350], \n \"L1 Write Misses\" : myArray[351],\n \"L1 Miss Rate\" : myArray[352],\n \n \"Victim Lines\" : myArray[353], \n \"Victim Cache Calls\" : myArray[354], \n \"Victim Cache Hits\" : myArray[355], \n \"Victim Cache Misses\" : myArray[356], \n \"Victim Cache Read Hits\" : myArray[357], \n \"Victim Cache Read Misses\" : myArray[358], \n \"Victim Cache Write Hits\" : myArray[359], \n \"Victim Cache Write Misses\" : myArray[360],\n \"Victim Cache Hit Rate\" : myArray[361], \n \n \"Total L2 Calls\" :myArray[362], \n \"L2 Read Calls\" : myArray[363],\n \"L2 Write Calls\" : myArray[364],\n \"L2 Read Hits\" : myArray[365],\n \"L2 Read Misses\" : myArray[366],\n \"L2 Write Hits\" : myArray[367],\n \"L2 Write Misses\" : myArray[368],\n \"L2 Miss Rate\" : myArray[369], \n \n \"Total Bus Traffic\" : myArray[370], \n \"Bus Read Hits\" : myArray[371],\n \"Bus Read Misses\" : myArray[372], \n \"Bus Write Hits\" : myArray[373], \n \"Bus Write Misses\" : myArray[374], \n \"Cache-to-Cache\" : myArray[375], \n \"Reads From Memory\" : myArray[376]\n }, \n {\n \"Core\" : myArray[377], //core 12\n \"Total Instructions\" : myArray[378],

\n \"Total Read Instructions\" : myArray[379],\n \"Total Write Instructions\" : myArray[380], \n \"L1 Read Hits\" : myArray[381], \n \"L1 Read Misses\" : myArray[382], \n \"L1 Write Hits\" : myArray[383], \n \"L1 Write Misses\" : myArray[384],\n \"L1 Miss Rate\" : myArray[385],\n \n \"Victim Lines\" : myArray[386], \n \"Victim Cache Calls\" : myArray[387], \n \"Victim Cache Hits\" : myArray[388], \n \"Victim Cache Misses\" : myArray[389], \n \"Victim Cache Read Hits\" : myArray[390], \n \"Victim Cache Read Misses\" : myArray[391], \n \"Victim Cache Write Hits\" : myArray[392], \n \"Victim Cache Write Misses\" : myArray[393],\n \"Victim Cache Hit Rate\" : myArray[394], \n \n \"Total L2 Calls\" :myArray[395], \n \"L2 Read Calls\" : myArray[396],\n \"L2 Write Calls\" : myArray[397],\n \"L2 Read Hits\" : myArray[398],\n \"L2 Read Misses\" : myArray[399],\n \"L2 Write Hits\" : myArray[400],\n \"L2 Write Misses\" : myArray[401],\n \"L2 Miss Rate\" : myArray[402], \n \n \"Total Bus Traffic\" : myArray[403], \n \"Bus Read Hits\" : myArray[404],\n \"Bus Read Misses\" : myArray[405], \n \"Bus Write Hits\" : myArray[406], \n \"Bus Write Misses\" : myArray[407], \n \"Cache-to-Cache\" : myArray[408], \n \"Reads From Memory\" : myArray[409]\n }];\n return newMsg;\n }\n}\nelse if(coreCount == 16) //16C\n{\n if(cacheSizeL2 == 0) //16C, L1 only\n {\n let newMsg = {payload : \"\"}\n \n newMsg.payload = [\n {\n \"Core\" : \"Total\",\n \"Core Count\" : myArray[0],\n \"L1 Cache Size\" : myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4], \n \"Coherence Protocol\" : myArray[5],\n \"Write Policy\" : myArray[6],\n \"Total Instructions\" : myArray[7],\n \"Total Read Instructions\" : myArray[8],\n \"Total Write Instructions\" : myArray[9],\n \n \"Cache-to-Cache\" : myArray[410],\n \"Total Bus Traffic\" : myArray[411], \n \"Reads From Memory\" : myArray[412],\n \"Global Miss Rate\" : myArray[413],\n \"L1 Read Hits\" : myArray[414],\n \"L1 Read Misses\" : myArray[415],\n \"L1 Write Hits\" : myArray[416],\n \"L1 Write Misses\" : myArray[417],\n \n \"Bus Read Hits\" : myArray[418], \n \"Bus Read Misses\" : myArray[419], \n \"Bus Write Hits\" : myArray[420], \n \"Bus Write Misses\" : myArray[421],\n \n \"Victim Lines\" : myArray[422],\n \"Victim Cache Calls\" : myArray[423],\n \"Victim Cache Hits\" : myArray[424],\n \"Victim Cache Misses\" : myArray[425], \n \"Victim Cache Read Hits\" : myArray[426],\n \"Victim Cache Read Misses\" : myArray[427],\n \"Victim Cache Write Hits\" : myArray[428],\n \"Victim Cache Write Misses\" : myArray[429],\n \n \"L2 Cache Size\" : \"N/A\",\n \"L2 Block Size\" : \"N/A\",\n \"L2 Associativity\" : \"N/A\", \n \"L2 Replacement Policy\" : \"N/A\", \n \"Total L2 Calls\" : \"N/A\",\n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\", \n \n \"L1 Miss Rate\" : myArray[430], \n \"Victim Cache Hit Rate\" : myArray[431], \n \"L2 Miss Rate\" : \"N/A\", \n \"Trace File\" : traceFile\n }, \n {\n \"Core\" : myArray[10], //core 1\n \"Total Instructions\" : myArray[11], \n \"Total Read Instructions\" : myArray[12],\n \"Total Write Instructions\" : myArray[13], \n \"L1 Read Hits\" : myArray[14], \n \"L1 Read Misses\" : myArray[15], \n \"L1 Write Hits\" : myArray[16], \n \"L1 Write Misses\" : myArray[17],\n \"Total Bus Traffic\" : myArray[18], \n \"Bus Read Hits\" : myArray[19],\n \"Bus Read

Misses\" : myArray[20], \n \"Bus Write Hits\" : myArray[21], \n \"Bus Write Misses\" : myArray[22], \n \"Cache-to-Cache\" : myArray[23], \n \"L1 Miss Rate\" : myArray[24], \n \n \"Victim Lines\" : myArray[25], \n \"Victim Cache Calls\" : myArray[26], \n \"Victim Cache Hits\" : myArray[27], \n \"Victim Cache Misses\" : myArray[28], \n \"Victim Cache Read Hits\" : myArray[29], \n \"Victim Cache Read Misses\" : myArray[30], \n \"Victim Cache Write Hits\" : myArray[31], \n \"Victim Cache Write Misses\" : myArray[32],\n \"Victim Cache Hit Rate\" : myArray[33], \n \"Reads From Memory\" : myArray[34],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[35], //core 2\n \"Total Instructions\" : myArray[36], \n \"Total Read Instructions\" : myArray[37],\n \"Total Write Instructions\" : myArray[38], \n \"L1 Read Hits\" : myArray[39], \n \"L1 Read Misses\" : myArray[40], \n \"L1 Write Hits\" : myArray[41], \n \"L1 Write Misses\" : myArray[42],\n \"Total Bus Traffic\" : myArray[43], \n \"Bus Read Hits\" : myArray[44],\n \"Bus Read Misses\" : myArray[45], \n \"Bus Write Hits\" : myArray[46], \n \"Bus Write Misses\" : myArray[47],\n \"Cache-to-Cache\" : myArray[48], \n \"L1 Miss Rate\" : myArray[49], \n \n \"Victim Lines\" : myArray[50], \n \"Victim Cache Calls\" : myArray[51], \n \"Victim Cache Hits\" : myArray[52], \n \"Victim Cache Misses\" : myArray[53], \n \"Victim Cache Read Hits\" : myArray[54], \n \"Victim Cache Read Misses\" : myArray[55], \n \"Victim Cache Write Hits\" : myArray[56], \n \"Victim Cache Write Misses\" : myArray[57],\n \"Victim Cache Hit Rate\" : myArray[58], \n \"Reads From Memory\" : myArray[59],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[60], //core 3\n \"Total Instructions\" : myArray[61], \n \"Total Read Instructions\" : myArray[62],\n \"Total Write Instructions\" : myArray[63], \n \"L1 Read Hits\" : myArray[64], \n \"L1 Read Misses\" : myArray[65], \n \"L1 Write Hits\" : myArray[66], \n \"L1 Write Misses\" : myArray[67],\n \"Total Bus Traffic\" : myArray[68], \n \"Bus Read Hits\" : myArray[69],\n \"Bus Read Misses\" : myArray[70], \n \"Bus Write Hits\" : myArray[71], \n \"Bus Write Misses\" : myArray[72], \n \"Cache-to-Cache\" : myArray[73], \n \"L1 Miss Rate\" : myArray[74], \n \n \"Victim Lines\" : myArray[75], \n \"Victim Cache Calls\" : myArray[76], \n \"Victim Cache Hits\" : myArray[77], \n \"Victim Cache Misses\" : myArray[78], \n \"Victim Cache Read Hits\" : myArray[79], \n \"Victim Cache Read Misses\" : myArray[80], \n \"Victim Cache Write Hits\" : myArray[81], \n \"Victim Cache Write Misses\" : myArray[82],\n \"Victim Cache Hit Rate\" : myArray[83], \n \"Reads From Memory\" : myArray[84],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[85], //core 4\n \"Total Instructions\" : myArray[86], \n \"Total Read Instructions\" : myArray[87],\n \"Total Write Instructions\" : myArray[88], \n \"L1 Read Hits\" :

myArray[89], \n \"L1 Read Misses\" : myArray[90], \n \"L1 Write Hits\" : myArray[91], \n \"L1 Write Misses\" : myArray[92],\n \"Total Bus Traffic\" : myArray[93], \n \"Bus Read Hits\" : myArray[94],\n \"Bus Read Misses\" : myArray[95], \n \"Bus Write Hits\" : myArray[96], \n \"Bus Write Misses\" : myArray[97],\n \"Cache-to-Cache\" : myArray[98], \n \"L1 Miss Rate\" : myArray[99], \n \n \"Victim Lines\" : myArray[100], \n \"Victim Cache Calls\" : myArray[101], \n \"Victim Cache Hits\" : myArray[102], \n \"Victim Cache Misses\" : myArray[103], \n \"Victim Cache Read Hits\" : myArray[104], \n \"Victim Cache Read Misses\" : myArray[105], \n \"Victim Cache Write Hits\" : myArray[106], \n \"Victim Cache Write Misses\" : myArray[107],\n \"Victim Cache Hit Rate\" : myArray[108], \n \"Reads From Memory\" : myArray[109],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[110], //core 5\n \"Total Instructions\" : myArray[111], \n \"Total Read Instructions\" : myArray[112],\n \"Total Write Instructions\" : myArray[113], \n \"L1 Read Hits\" : myArray[114], \n \"L1 Read Misses\" : myArray[115], \n \"L1 Write Hits\" : myArray[116], \n \"L1 Write Misses\" : myArray[117],\n \"Total Bus Traffic\" : myArray[118], \n \"Bus Read Hits\" : myArray[119],\n \"Bus Read Misses\" : myArray[120], \n \"Bus Write Hits\" : myArray[121], \n \"Bus Write Misses\" : myArray[122], \n \"Cache-to-Cache\" : myArray[123], \n \"L1 Miss Rate\" : myArray[124], \n \n \"Victim Lines\" : myArray[125], \n \"Victim Cache Calls\" : myArray[126], \n \"Victim Cache Hits\" : myArray[127], \n \"Victim Cache Misses\" : myArray[128], \n \"Victim Cache Read Hits\" : myArray[129], \n \"Victim Cache Read Misses\" : myArray[130], \n \"Victim Cache Write Hits\" : myArray[131], \n \"Victim Cache Write Misses\" : myArray[132],\n \"Victim Cache Hit Rate\" : myArray[133], \n \"Reads From Memory\" : myArray[134],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[135], //core 6\n \"Total Instructions\" : myArray[136], \n \"Total Read Instructions\" : myArray[137],\n \"Total Write Instructions\" : myArray[138], \n \"L1 Read Hits\" : myArray[139], \n \"L1 Read Misses\" : myArray[140], \n \"L1 Write Hits\" : myArray[141], \n \"L1 Write Misses\" : myArray[142],\n \"Total Bus Traffic\" : myArray[143], \n \"Bus Read Hits\" : myArray[144],\n \"Bus Read Misses\" : myArray[145], \n \"Bus Write Hits\" : myArray[146], \n \"Bus Write Misses\" : myArray[147],\n \"Cache-to-Cache\" : myArray[148], \n \"L1 Miss Rate\" : myArray[149], \n \n \"Victim Lines\" : myArray[150], \n \"Victim Cache Calls\" : myArray[151], \n \"Victim Cache Hits\" : myArray[152], \n \"Victim Cache Misses\" : myArray[153], \n \"Victim Cache Read Hits\" : myArray[154], \n \"Victim Cache Read Misses\" : myArray[155], \n \"Victim Cache Write Hits\" : myArray[156], \n \"Victim Cache Write Misses\" : myArray[157],\n \"Victim Cache Hit Rate\" : myArray[158], \n \"Reads From Memory\" : myArray[159],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n

\"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[160], //core 7\n \"Total Instructions\" : myArray[161], \n \"Total Read Instructions\" : myArray[162],\n \"Total Write Instructions\" : myArray[163], \n \"L1 Read Hits\" : myArray[164], \n \"L1 Read Misses\" : myArray[165], \n \"L1 Write Hits\" : myArray[166], \n \"L1 Write Misses\" : myArray[167],\n \"Total Bus Traffic\" : myArray[168], \n \"Bus Read Hits\" : myArray[169],\n \"Bus Read Misses\" : myArray[170], \n \"Bus Write Hits\" : myArray[171], \n \"Bus Write Misses\" : myArray[172], \n \"Cache-to-Cache\" : myArray[173], \n \"L1 Miss Rate\" : myArray[174], \n \n \"Victim Lines\" : myArray[175], \n \"Victim Cache Calls\" : myArray[176], \n \"Victim Cache Hits\" : myArray[177], \n \"Victim Cache Misses\" : myArray[178], \n \"Victim Cache Read Hits\" : myArray[179], \n \"Victim Cache Read Misses\" : myArray[180], \n \"Victim Cache Write Hits\" : myArray[181], \n \"Victim Cache Write Misses\" : myArray[182],\n \"Victim Cache Hit Rate\" : myArray[183], \n \"Reads From Memory\" : myArray[184],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[185], //core 8\n \"Total Instructions\" : myArray[186], \n \"Total Read Instructions\" : myArray[187],\n \"Total Write Instructions\" : myArray[188], \n \"L1 Read Hits\" : myArray[189], \n \"L1 Read Misses\" : myArray[190], \n \"L1 Write Hits\" : myArray[191], \n \"L1 Write Misses\" : myArray[192],\n \"Total Bus Traffic\" : myArray[193], \n \"Bus Read Hits\" : myArray[194],\n \"Bus Read Misses\" : myArray[195], \n \"Bus Write Hits\" : myArray[196], \n \"Bus Write Misses\" : myArray[197],\n \"Cache-to-Cache\" : myArray[198], \n \"L1 Miss Rate\" : myArray[199], \n \n \"Victim Lines\" : myArray[200], \n \"Victim Cache Calls\" : myArray[201], \n \"Victim Cache Hits\" : myArray[202], \n \"Victim Cache Misses\" : myArray[203], \n \"Victim Cache Read Hits\" : myArray[204], \n \"Victim Cache Read Misses\" : myArray[205], \n \"Victim Cache Write Hits\" : myArray[206], \n \"Victim Cache Write Misses\" : myArray[207],\n \"Victim Cache Hit Rate\" : myArray[208], \n \"Reads From Memory\" : myArray[209],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[210], //core 9\n \"Total Instructions\" : myArray[211], \n \"Total Read Instructions\" : myArray[212],\n \"Total Write Instructions\" : myArray[213], \n \"L1 Read Hits\" : myArray[214], \n \"L1 Read Misses\" : myArray[215], \n \"L1 Write Hits\" : myArray[216], \n \"L1 Write Misses\" : myArray[217],\n \"Total Bus Traffic\" : myArray[218], \n \"Bus Read Hits\" : myArray[219],\n \"Bus Read Misses\" : myArray[220], \n \"Bus Write Hits\" : myArray[221], \n \"Bus Write Misses\" : myArray[222], \n \"Cache-to-Cache\" : myArray[223], \n \"L1 Miss Rate\" : myArray[224], \n \n \"Victim Lines\" : myArray[225], \n \"Victim Cache Calls\" : myArray[226], \n \"Victim Cache Hits\" : myArray[227], \n \"Victim Cache Misses\" : myArray[228], \n \"Victim Cache Read Hits\" : myArray[229], \n \"Victim Cache Read Misses\" : myArray[230], \n

\"Victim Cache Write Hits\" : myArray[231], \n \"Victim Cache Write Misses\" : myArray[232],\n \"Victim Cache Hit Rate\" : myArray[233], \n \"Reads From Memory\" : myArray[234],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[235], //core 10\n \"Total Instructions\" : myArray[236], \n \"Total Read Instructions\" : myArray[237],\n \"Total Write Instructions\" : myArray[238], \n \"L1 Read Hits\" : myArray[239], \n \"L1 Read Misses\" : myArray[240], \n \"L1 Write Hits\" : myArray[241], \n \"L1 Write Misses\" : myArray[242],\n \"Total Bus Traffic\" : myArray[243], \n \"Bus Read Hits\" : myArray[244],\n \"Bus Read Misses\" : myArray[245], \n \"Bus Write Hits\" : myArray[246], \n \"Bus Write Misses\" : myArray[247],\n \"Cache-to-Cache\" : myArray[248], \n \"L1 Miss Rate\" : myArray[249], \n \n \"Victim Lines\" : myArray[250], \n \"Victim Cache Calls\" : myArray[251], \n \"Victim Cache Hits\" : myArray[252], \n \"Victim Cache Misses\" : myArray[253], \n \"Victim Cache Read Hits\" : myArray[254], \n \"Victim Cache Read Misses\" : myArray[255], \n \"Victim Cache Write Hits\" : myArray[256], \n \"Victim Cache Write Misses\" : myArray[257],\n \"Victim Cache Hit Rate\" : myArray[258], \n \"Reads From Memory\" : myArray[259],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[260], //core 11\n \"Total Instructions\" : myArray[261], \n \"Total Read Instructions\" : myArray[262],\n \"Total Write Instructions\" : myArray[263], \n \"L1 Read Hits\" : myArray[264], \n \"L1 Read Misses\" : myArray[265], \n \"L1 Write Hits\" : myArray[266], \n \"L1 Write Misses\" : myArray[267],\n \"Total Bus Traffic\" : myArray[268], \n \"Bus Read Hits\" : myArray[269],\n \"Bus Read Misses\" : myArray[270], \n \"Bus Write Hits\" : myArray[271], \n \"Bus Write Misses\" : myArray[272], \n \"Cache-to-Cache\" : myArray[273], \n \"L1 Miss Rate\" : myArray[274], \n \n \"Victim Lines\" : myArray[275], \n \"Victim Cache Calls\" : myArray[276], \n \"Victim Cache Hits\" : myArray[277], \n \"Victim Cache Misses\" : myArray[278], \n \"Victim Cache Read Hits\" : myArray[279], \n \"Victim Cache Read Misses\" : myArray[280], \n \"Victim Cache Write Hits\" : myArray[281], \n \"Victim Cache Write Misses\" : myArray[282],\n \"Victim Cache Hit Rate\" : myArray[283], \n \"Reads From Memory\" : myArray[284],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[285], //core 12\n \"Total Instructions\" : myArray[286], \n \"Total Read Instructions\" : myArray[287],\n \"Total Write Instructions\" : myArray[288], \n \"L1 Read Hits\" : myArray[289], \n \"L1 Read Misses\" : myArray[290], \n \"L1 Write Hits\" : myArray[291], \n \"L1 Write Misses\" : myArray[292],\n \"Total Bus Traffic\" : myArray[293], \n \"Bus Read Hits\" : myArray[294],\n \"Bus Read Misses\" : myArray[295], \n \"Bus Write Hits\" : myArray[296], \n \"Bus Write Misses\" : myArray[297],\n \"Cache-

to-Cache\" : myArray[298], \n \"L1 Miss Rate\" : myArray[299], \n \n
\"Victim Lines\" : myArray[300], \n \"Victim Cache Calls\" : myArray[301], \n
\"Victim Cache Hits\" : myArray[302], \n \"Victim Cache Misses\" : myArray[303], \n
\"Victim Cache Read Hits\" : myArray[304], \n \"Victim Cache Read Misses\" :
myArray[305], \n \"Victim Cache Write Hits\" : myArray[306], \n \"Victim Cache
Write Misses\" : myArray[307],\n \"Victim Cache Hit Rate\" : myArray[308], \n
\"Reads From Memory\" : myArray[309],\n \n \"Total L2 Calls\" :\"N/A\", \n
\"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" :
\"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n
\"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n
\"Core\" : myArray[310], //core 13\n \"Total Instructions\" : myArray[311],
\n \"Total Read Instructions\" : myArray[312],\n \"Total Write Instructions\" :
myArray[313], \n \"L1 Read Hits\" : myArray[314], \n \"L1 Read Misses\" :
myArray[315], \n \"L1 Write Hits\" : myArray[316], \n \"L1 Write Misses\" :
myArray[317],\n \"Total Bus Traffic\" : myArray[318], \n \"Bus Read Hits\" :
myArray[319],\n \"Bus Read Misses\" : myArray[320], \n \"Bus Write Hits\" :
myArray[321], \n \"Bus Write Misses\" : myArray[322], \n \"Cache-to-Cache\" :
myArray[323], \n \"L1 Miss Rate\" : myArray[324], \n \n \"Victim Lines\" :
myArray[325], \n \"Victim Cache Calls\" : myArray[326], \n \"Victim Cache Hits\"
: myArray[327], \n \"Victim Cache Misses\" : myArray[328], \n \"Victim Cache
Read Hits\" : myArray[329], \n \"Victim Cache Read Misses\" : myArray[330], \n
\"Victim Cache Write Hits\" : myArray[331], \n \"Victim Cache Write Misses\" :
myArray[332],\n \"Victim Cache Hit Rate\" : myArray[333], \n \"Reads From
Memory\" : myArray[334],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read
Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n
\"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\"
: \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" :
myArray[335], //core 14\n \"Total Instructions\" : myArray[336], \n
\"Total Read Instructions\" : myArray[337],\n \"Total Write Instructions\" :
myArray[338], \n \"L1 Read Hits\" : myArray[339], \n \"L1 Read Misses\" :
myArray[340], \n \"L1 Write Hits\" : myArray[341], \n \"L1 Write Misses\" :
myArray[342],\n \"Total Bus Traffic\" : myArray[343], \n \"Bus Read Hits\" :
myArray[344],\n \"Bus Read Misses\" : myArray[345], \n \"Bus Write Hits\" :
myArray[346], \n \"Bus Write Misses\" : myArray[347],\n \"Cache-to-Cache\" :
myArray[348], \n \"L1 Miss Rate\" : myArray[349], \n \n \"Victim Lines\" :
myArray[350], \n \"Victim Cache Calls\" : myArray[351], \n \"Victim Cache Hits\"
: myArray[352], \n \"Victim Cache Misses\" : myArray[353], \n \"Victim Cache
Read Hits\" : myArray[354], \n \"Victim Cache Read Misses\" : myArray[355], \n
\"Victim Cache Write Hits\" : myArray[356], \n \"Victim Cache Write Misses\" :
myArray[357],\n \"Victim Cache Hit Rate\" : myArray[358], \n \"Reads From
Memory\" : myArray[359],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read
Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n
\"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\"
: \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n },\n {\n \"Core\" : myArray[360],
//core 15\n \"Total Instructions\" : myArray[361], \n \"Total Read Instructions\" :
myArray[362],\n \"Total Write Instructions\" : myArray[363], \n \"L1 Read Hits\" :

myArray[364], \n \"L1 Read Misses\" : myArray[365], \n \"L1 Write Hits\" : myArray[366], \n \"L1 Write Misses\" : myArray[367],\n \"Total Bus Traffic\" : myArray[368], \n \"Bus Read Hits\" : myArray[369],\n \"Bus Read Misses\" : myArray[370], \n \"Bus Write Hits\" : myArray[371], \n \"Bus Write Misses\" : myArray[372], \n \"Cache-to-Cache\" : myArray[373], \n \"L1 Miss Rate\" : myArray[374], \n \n \"Victim Lines\" : myArray[375], \n \"Victim Cache Calls\" : myArray[376], \n \"Victim Cache Hits\" : myArray[377], \n \"Victim Cache Misses\" : myArray[378], \n \"Victim Cache Read Hits\" : myArray[379], \n \"Victim Cache Read Misses\" : myArray[380], \n \"Victim Cache Write Hits\" : myArray[381], \n \"Victim Cache Write Misses\" : myArray[382],\n \"Victim Cache Hit Rate\" : myArray[383], \n \"Reads From Memory\" : myArray[384],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }, \n {\n \"Core\" : myArray[385], //core 16\n \"Total Instructions\" : myArray[386], \n \"Total Read Instructions\" : myArray[387],\n \"Total Write Instructions\" : myArray[388], \n \"L1 Read Hits\" : myArray[389], \n \"L1 Read Misses\" : myArray[390], \n \"L1 Write Hits\" : myArray[391], \n \"L1 Write Misses\" : myArray[392],\n \"Total Bus Traffic\" : myArray[393], \n \"Bus Read Hits\" : myArray[394],\n \"Bus Read Misses\" : myArray[395], \n \"Bus Write Hits\" : myArray[396], \n \"Bus Write Misses\" : myArray[397],\n \"Cache-to-Cache\" : myArray[398], \n \"L1 Miss Rate\" : myArray[399], \n \n \"Victim Lines\" : myArray[400], \n \"Victim Cache Calls\" : myArray[401], \n \"Victim Cache Hits\" : myArray[402], \n \"Victim Cache Misses\" : myArray[403], \n \"Victim Cache Read Hits\" : myArray[404], \n \"Victim Cache Read Misses\" : myArray[405], \n \"Victim Cache Write Hits\" : myArray[406], \n \"Victim Cache Write Misses\" : myArray[407],\n \"Victim Cache Hit Rate\" : myArray[408], \n \"Reads From Memory\" : myArray[409],\n \n \"Total L2 Calls\" :\"N/A\", \n \"L2 Read Calls\" : \"N/A\",\n \"L2 Write Calls\" : \"N/A\",\n \"L2 Read Hits\" : \"N/A\",\n \"L2 Read Misses\" : \"N/A\",\n \"L2 Write Hits\" : \"N/A\",\n \"L2 Write Misses\" : \"N/A\",\n \"L2 Miss Rate\" : \"N/A\"\n }];\n return newMsg;\n }\n else //16C, L1 and L2\n {\n let newMsg = {payload : \"\"}\n \n newMsg.payload = [\n {\n \"Core\" : \"Total\",\n \"Core Count\" : myArray[0],\n \"L1 Cache Size\" : myArray[1],\n \"L1 Block Size\" : myArray[2],\n \"L1 Associativity\" : myArray[3],\n \"L1 Replacement Policy\" : myArray[4],\n \"L2 Cache Size\" : myArray[5],\n \"L2 Block Size\" : myArray[6],\n \"L2 Associativity\" : myArray[7], \n \"L2 Replacement Policy\" : myArray[8], \n \"Coherence Protocol\" : myArray[9],\n \"Write Policy\" : myArray[10],\n \"Total Instructions\" : myArray[11],\n \"Total Read Instructions\" : myArray[12],\n \"Total Write Instructions\" : myArray[13],\n \n \"Cache-to-Cache\" : myArray[542],\n \"Total Bus Traffic\" : myArray[543], \n \"Reads From Memory\" : myArray[544],\n \"Global Miss Rate\" : myArray[545],\n \"L1 Read Hits\" : myArray[546],\n \"L1 Read Misses\" : myArray[547],\n \"L1 Write Hits\" : myArray[548],\n \"L1 Write Misses\" : myArray[549],\n \n \"Bus Read Hits\" : myArray[550], \n \"Bus Read Misses\" : myArray[551], \n \"Bus Write Hits\" : myArray[552], \n \"Bus Write Misses\" : myArray[553],\n \n

\"Total L2 Calls\" : myArray[554],\n \"L2 Read Calls\" : myArray[555],\n \"L2 Write Calls\" : myArray[556],\n \"L2 Read Hits\" : myArray[557],\n \"L2 Read Misses\" : myArray[558],\n \"L2 Write Hits\" : myArray[559],\n \"L2 Write Misses\" : myArray[560],\n \n \"Victim Lines\" : myArray[561],\n \"Victim Cache Calls\" : myArray[562],\n \"Victim Cache Hits\" : myArray[563],\n \"Victim Cache Misses\" : myArray[564], \n \"Victim Cache Read Hits\" : myArray[565],\n \"Victim Cache Read Misses\" : myArray[566],\n \"Victim Cache Write Hits\" : myArray[567],\n \"Victim Cache Write Misses\" : myArray[568], \n \n \"L1 Miss Rate\" : myArray[569], \n \"Victim Cache Hit Rate\" : myArray[570], \n \"L2 Miss Rate\" : myArray[571], \n \"Trace File\" : traceFile\n }, \n {\n \"Core\" : myArray[14], //core 1\n \"Total Instructions\" : myArray[15], \n \"Total Read Instructions\" : myArray[16],\n \"Total Write Instructions\" : myArray[17], \n \"L1 Read Hits\" : myArray[18], \n \"L1 Read Misses\" : myArray[19], \n \"L1 Write Hits\" : myArray[20], \n \"L1 Write Misses\" : myArray[21],\n \"L1 Miss Rate\" : myArray[22],\n \n \"Victim Lines\" : myArray[23], \n \"Victim Cache Calls\" : myArray[24], \n \"Victim Cache Hits\" : myArray[25], \n \"Victim Cache Misses\" : myArray[26], \n \"Victim Cache Read Hits\" : myArray[27], \n \"Victim Cache Read Misses\" : myArray[28], \n \"Victim Cache Write Hits\" : myArray[29], \n \"Victim Cache Write Misses\" : myArray[30],\n \"Victim Cache Hit Rate\" : myArray[31], \n \n \"Total L2 Calls\" :myArray[32], \n \"L2 Read Calls\" : myArray[33],\n \"L2 Write Calls\" : myArray[34],\n \"L2 Read Hits\" : myArray[35],\n \"L2 Read Misses\" : myArray[36],\n \"L2 Write Hits\" : myArray[37],\n \"L2 Write Misses\" : myArray[38],\n \"L2 Miss Rate\" : myArray[39], \n \n \"Total Bus Traffic\" : myArray[40], \n \"Bus Read Hits\" : myArray[41],\n \"Bus Read Misses\" : myArray[42], \n \"Bus Write Hits\" : myArray[43], \n \"Bus Write Misses\" : myArray[44], \n \"Cache-to-Cache\" : myArray[45], \n \"Reads From Memory\" : myArray[46]\n }, \n {\n \"Core\" : myArray[47], //core 2\n \"Total Instructions\" : myArray[48], \n \"Total Read Instructions\" : myArray[49],\n \"Total Write Instructions\" : myArray[50], \n \"L1 Read Hits\" : myArray[51], \n \"L1 Read Misses\" : myArray[52], \n \"L1 Write Hits\" : myArray[53], \n \"L1 Write Misses\" : myArray[54],\n \"L1 Miss Rate\" : myArray[55],\n \n \"Victim Lines\" : myArray[56], \n \"Victim Cache Calls\" : myArray[57], \n \"Victim Cache Hits\" : myArray[58], \n \"Victim Cache Misses\" : myArray[59], \n \"Victim Cache Read Hits\" : myArray[60], \n \"Victim Cache Read Misses\" : myArray[61], \n \"Victim Cache Write Hits\" : myArray[62], \n \"Victim Cache Write Misses\" : myArray[63],\n \"Victim Cache Hit Rate\" : myArray[64], \n \n \"Total L2 Calls\" :myArray[65], \n \"L2 Read Calls\" : myArray[66],\n \"L2 Write Calls\" : myArray[67],\n \"L2 Read Hits\" : myArray[68],\n \"L2 Read Misses\" : myArray[69],\n \"L2 Write Hits\" : myArray[70],\n \"L2 Write Misses\" : myArray[71],\n \"L2 Miss Rate\" : myArray[72], \n \n \"Total Bus Traffic\" : myArray[73], \n \"Bus Read Hits\" : myArray[74],\n \"Bus Read Misses\" : myArray[75], \n \"Bus Write Hits\" : myArray[76], \n \"Bus Write Misses\" : myArray[77], \n \"Cache-to-Cache\" : myArray[78], \n \"Reads From Memory\" : myArray[79]\n }, \n {\n \"Core\" : myArray[80], //core 3\n \"Total Instructions\" : myArray[81], \n \"Total Read Instructions\" : myArray[82],\n \"Total Write Instructions\" : myArray[83],

\n \"L1 Read Hits\" : myArray[84], \n \"L1 Read Misses\" : myArray[85], \n \"L1 Write Hits\" : myArray[86], \n \"L1 Write Misses\" : myArray[87],\n \"L1 Miss Rate\" : myArray[88],\n \n \"Victim Lines\" : myArray[89], \n \"Victim Cache Calls\" : myArray[90], \n \"Victim Cache Hits\" : myArray[91], \n \"Victim Cache Misses\" : myArray[92], \n \"Victim Cache Read Hits\" : myArray[93], \n \"Victim Cache Read Misses\" : myArray[94], \n \"Victim Cache Write Hits\" : myArray[95], \n \"Victim Cache Write Misses\" : myArray[96],\n \"Victim Cache Hit Rate\" : myArray[97], \n \n \"Total L2 Calls\" :myArray[98], \n \"L2 Read Calls\" : myArray[99],\n \"L2 Write Calls\" : myArray[100],\n \"L2 Read Hits\" : myArray[101],\n \"L2 Read Misses\" : myArray[102],\n \"L2 Write Hits\" : myArray[103],\n \"L2 Write Misses\" : myArray[104],\n \"L2 Miss Rate\" : myArray[105], \n \n \"Total Bus Traffic\" : myArray[106], \n \"Bus Read Hits\" : myArray[107],\n \"Bus Read Misses\" : myArray[108], \n \"Bus Write Hits\" : myArray[109], \n \"Bus Write Misses\" : myArray[110], \n \"Cache-to-Cache\" : myArray[111], \n \"Reads From Memory\" : myArray[112]\n }, \n {\n \"Core\" : myArray[113], //core 4\n \"Total Instructions\" : myArray[114], \n \"Total Read Instructions\" : myArray[115],\n \"Total Write Instructions\" : myArray[116], \n \"L1 Read Hits\" : myArray[117], \n \"L1 Read Misses\" : myArray[118], \n \"L1 Write Hits\" : myArray[119], \n \"L1 Write Misses\" : myArray[120],\n \"L1 Miss Rate\" : myArray[121],\n \n \"Victim Lines\" : myArray[122], \n \"Victim Cache Calls\" : myArray[123], \n \"Victim Cache Hits\" : myArray[124], \n \"Victim Cache Misses\" : myArray[125], \n \"Victim Cache Read Hits\" : myArray[126], \n \"Victim Cache Read Misses\" : myArray[127], \n \"Victim Cache Write Hits\" : myArray[128], \n \"Victim Cache Write Misses\" : myArray[129],\n \"Victim Cache Hit Rate\" : myArray[130], \n \n \"Total L2 Calls\" :myArray[131], \n \"L2 Read Calls\" : myArray[132],\n \"L2 Write Calls\" : myArray[133],\n \"L2 Read Hits\" : myArray[134],\n \"L2 Read Misses\" : myArray[135],\n \"L2 Write Hits\" : myArray[136],\n \"L2 Write Misses\" : myArray[137],\n \"L2 Miss Rate\" : myArray[138], \n \n \"Total Bus Traffic\" : myArray[139], \n \"Bus Read Hits\" : myArray[140],\n \"Bus Read Misses\" : myArray[141], \n \"Bus Write Hits\" : myArray[142], \n \"Bus Write Misses\" : myArray[143], \n \"Cache-to-Cache\" : myArray[144], \n \"Reads From Memory\" : myArray[145]\n }, \n {\n \"Core\" : myArray[146], //core 5\n \"Total Instructions\" : myArray[147], \n \"Total Read Instructions\" : myArray[148],\n \"Total Write Instructions\" : myArray[149], \n \"L1 Read Hits\" : myArray[150], \n \"L1 Read Misses\" : myArray[151], \n \"L1 Write Hits\" : myArray[152], \n \"L1 Write Misses\" : myArray[153],\n \"L1 Miss Rate\" : myArray[154],\n \n \"Victim Lines\" : myArray[155], \n \"Victim Cache Calls\" : myArray[156], \n \"Victim Cache Hits\" : myArray[157], \n \"Victim Cache Misses\" : myArray[158], \n \"Victim Cache Read Hits\" : myArray[159], \n \"Victim Cache Read Misses\" : myArray[160], \n \"Victim Cache Write Hits\" : myArray[161], \n \"Victim Cache Write Misses\" : myArray[162],\n \"Victim Cache Hit Rate\" : myArray[163], \n \n \"Total L2 Calls\" :myArray[164], \n \"L2 Read Calls\" : myArray[165],\n \"L2 Write Calls\" : myArray[166],\n \"L2 Read Hits\" : myArray[167],\n \"L2 Read Misses\" : myArray[168],\n \"L2 Write Hits\" : myArray[169],\n \"L2 Write Misses\" : myArray[170],\n \"L2 Miss Rate\" :

myArray[171], \n \n \"Total Bus Traffic\" : myArray[172], \n \"Bus Read Hits\" : myArray[173],\n \"Bus Read Misses\" : myArray[174], \n \"Bus Write Hits\" : myArray[175], \n \"Bus Write Misses\" : myArray[176], \n \"Cache-to-Cache\" : myArray[177], \n \"Reads From Memory\" : myArray[178]\n }, \n {\n \"Core\" : myArray[179], //core 6\n \"Total Instructions\" : myArray[180], \n \"Total Read Instructions\" : myArray[181],\n \"Total Write Instructions\" : myArray[182], \n \"L1 Read Hits\" : myArray[183], \n \"L1 Read Misses\" : myArray[184], \n \"L1 Write Hits\" : myArray[185], \n \"L1 Write Misses\" : myArray[186],\n \"L1 Miss Rate\" : myArray[187],\n \n \"Victim Lines\" : myArray[188], \n \"Victim Cache Calls\" : myArray[189], \n \"Victim Cache Hits\" : myArray[190], \n \"Victim Cache Misses\" : myArray[191], \n \"Victim Cache Read Hits\" : myArray[192], \n \"Victim Cache Read Misses\" : myArray[193], \n \"Victim Cache Write Hits\" : myArray[194], \n \"Victim Cache Write Misses\" : myArray[195],\n \"Victim Cache Hit Rate\" : myArray[196], \n \n \"Total L2 Calls\" :myArray[197], \n \"L2 Read Calls\" : myArray[198],\n \"L2 Write Calls\" : myArray[199],\n \"L2 Read Hits\" : myArray[200],\n \"L2 Read Misses\" : myArray[201],\n \"L2 Write Hits\" : myArray[202],\n \"L2 Write Misses\" : myArray[203],\n \"L2 Miss Rate\" : myArray[204], \n \n \"Total Bus Traffic\" : myArray[205], \n \"Bus Read Hits\" : myArray[206],\n \"Bus Read Misses\" : myArray[207], \n \"Bus Write Hits\" : myArray[208], \n \"Bus Write Misses\" : myArray[209], \n \"Cache-to-Cache\" : myArray[210], \n \"Reads From Memory\" : myArray[211]\n }, \n {\n \"Core\" : myArray[212], //core 7\n \"Total Instructions\" : myArray[213], \n \"Total Read Instructions\" : myArray[214],\n \"Total Write Instructions\" : myArray[215], \n \"L1 Read Hits\" : myArray[216], \n \"L1 Read Misses\" : myArray[217], \n \"L1 Write Hits\" : myArray[218], \n \"L1 Write Misses\" : myArray[219],\n \"L1 Miss Rate\" : myArray[220],\n \n \"Victim Lines\" : myArray[221], \n \"Victim Cache Calls\" : myArray[222], \n \"Victim Cache Hits\" : myArray[223], \n \"Victim Cache Misses\" : myArray[224], \n \"Victim Cache Read Hits\" : myArray[225], \n \"Victim Cache Read Misses\" : myArray[226], \n \"Victim Cache Write Hits\" : myArray[227], \n \"Victim Cache Write Misses\" : myArray[228],\n \"Victim Cache Hit Rate\" : myArray[229], \n \n \"Total L2 Calls\" :myArray[230], \n \"L2 Read Calls\" : myArray[231],\n \"L2 Write Calls\" : myArray[232],\n \"L2 Read Hits\" : myArray[233],\n \"L2 Read Misses\" : myArray[234],\n \"L2 Write Hits\" : myArray[235],\n \"L2 Write Misses\" : myArray[236],\n \"L2 Miss Rate\" : myArray[237], \n \n \"Total Bus Traffic\" : myArray[238], \n \"Bus Read Hits\" : myArray[239],\n \"Bus Read Misses\" : myArray[240], \n \"Bus Write Hits\" : myArray[241], \n \"Bus Write Misses\" : myArray[242], \n \"Cache-to-Cache\" : myArray[243], \n \"Reads From Memory\" : myArray[244]\n }, \n {\n \"Core\" : myArray[245], //core 8\n \"Total Instructions\" : myArray[246], \n \"Total Read Instructions\" : myArray[247],\n \"Total Write Instructions\" : myArray[248], \n \"L1 Read Hits\" : myArray[249], \n \"L1 Read Misses\" : myArray[250], \n \"L1 Write Hits\" : myArray[251], \n \"L1 Write Misses\" : myArray[252],\n \"L1 Miss Rate\" : myArray[253],\n \n \"Victim Lines\" : myArray[254], \n \"Victim Cache Calls\" : myArray[255], \n \"Victim Cache Hits\" : myArray[256], \n \"Victim Cache Misses\" : myArray[257], \n \"Victim Cache

Read Hits\" : myArray[258], \n \"Victim Cache Read Misses\" : myArray[259], \n \"Victim Cache Write Hits\" : myArray[260], \n \"Victim Cache Write Misses\" : myArray[261],\n \"Victim Cache Hit Rate\" : myArray[262], \n \n \"Total L2 Calls\" :myArray[263], \n \"L2 Read Calls\" : myArray[264],\n \"L2 Write Calls\" : myArray[265],\n \"L2 Read Hits\" : myArray[266],\n \"L2 Read Misses\" : myArray[267],\n \"L2 Write Hits\" : myArray[268],\n \"L2 Write Misses\" : myArray[269],\n \"L2 Miss Rate\" : myArray[270], \n \n \"Total Bus Traffic\" : myArray[271], \n \"Bus Read Hits\" : myArray[272],\n \"Bus Read Misses\" : myArray[273], \n \"Bus Write Hits\" : myArray[274], \n \"Bus Write Misses\" : myArray[275], \n \"Cache-to-Cache\" : myArray[276], \n \"Reads From Memory\" : myArray[277]\n },\n {\n \"Core\" : myArray[278], //core 9\n \"Total Instructions\" : myArray[279], \n \"Total Read Instructions\" : myArray[280],\n \"Total Write Instructions\" : myArray[281], \n \"L1 Read Hits\" : myArray[282], \n \"L1 Read Misses\" : myArray[283], \n \"L1 Write Hits\" : myArray[284], \n \"L1 Write Misses\" : myArray[285],\n \"L1 Miss Rate\" : myArray[286],\n \n \"Victim Lines\" : myArray[287], \n \"Victim Cache Calls\" : myArray[288], \n \"Victim Cache Hits\" : myArray[289], \n \"Victim Cache Misses\" : myArray[290], \n \"Victim Cache Read Hits\" : myArray[291], \n \"Victim Cache Read Misses\" : myArray[292], \n \"Victim Cache Write Hits\" : myArray[293], \n \"Victim Cache Write Misses\" : myArray[294],\n \"Victim Cache Hit Rate\" : myArray[295], \n \n \"Total L2 Calls\" :myArray[296], \n \"L2 Read Calls\" : myArray[297],\n \"L2 Write Calls\" : myArray[298],\n \"L2 Read Hits\" : myArray[299],\n \"L2 Read Misses\" : myArray[300],\n \"L2 Write Hits\" : myArray[301],\n \"L2 Write Misses\" : myArray[302],\n \"L2 Miss Rate\" : myArray[303], \n \n \"Total Bus Traffic\" : myArray[304], \n \"Bus Read Hits\" : myArray[305],\n \"Bus Read Misses\" : myArray[306], \n \"Bus Write Hits\" : myArray[307], \n \"Bus Write Misses\" : myArray[308], \n \"Cache-to-Cache\" : myArray[309], \n \"Reads From Memory\" : myArray[310]\n }, \n {\n \"Core\" : myArray[311], //core 10\n \"Total Instructions\" : myArray[312], \n \"Total Read Instructions\" : myArray[313],\n \"Total Write Instructions\" : myArray[314], \n \"L1 Read Hits\" : myArray[315], \n \"L1 Read Misses\" : myArray[316], \n \"L1 Write Hits\" : myArray[317], \n \"L1 Write Misses\" : myArray[318],\n \"L1 Miss Rate\" : myArray[319],\n \n \"Victim Lines\" : myArray[320], \n \"Victim Cache Calls\" : myArray[321], \n \"Victim Cache Hits\" : myArray[322], \n \"Victim Cache Misses\" : myArray[323], \n \"Victim Cache Read Hits\" : myArray[324], \n \"Victim Cache Read Misses\" : myArray[325], \n \"Victim Cache Write Hits\" : myArray[326], \n \"Victim Cache Write Misses\" : myArray[327],\n \"Victim Cache Hit Rate\" : myArray[328], \n \n \"Total L2 Calls\" :myArray[329], \n \"L2 Read Calls\" : myArray[330],\n \"L2 Write Calls\" : myArray[331],\n \"L2 Read Hits\" : myArray[332],\n \"L2 Read Misses\" : myArray[333],\n \"L2 Write Hits\" : myArray[334],\n \"L2 Write Misses\" : myArray[335],\n \"L2 Miss Rate\" : myArray[336], \n \n \"Total Bus Traffic\" : myArray[337], \n \"Bus Read Hits\" : myArray[338],\n \"Bus Read Misses\" : myArray[339], \n \"Bus Write Hits\" : myArray[340], \n \"Bus Write Misses\" : myArray[341], \n \"Cache-to-Cache\" : myArray[342], \n \"Reads From Memory\" : myArray[343]\n }, \n {\n \"Core\" : myArray[344],

//core 11\n \"Total Instructions\" : myArray[345], \n \"Total Read Instructions\" : myArray[346],\n \"Total Write Instructions\" : myArray[347], \n \"L1 Read Hits\" : myArray[348], \n \"L1 Read Misses\" : myArray[349], \n \"L1 Write Hits\" : myArray[350], \n \"L1 Write Misses\" : myArray[351],\n \"L1 Miss Rate\" : myArray[352],\n \n \"Victim Lines\" : myArray[353], \n \"Victim Cache Calls\" : myArray[354], \n \"Victim Cache Hits\" : myArray[355], \n \"Victim Cache Misses\" : myArray[356], \n \"Victim Cache Read Hits\" : myArray[357], \n \"Victim Cache Read Misses\" : myArray[358], \n \"Victim Cache Write Hits\" : myArray[359], \n \"Victim Cache Write Misses\" : myArray[360],\n \"Victim Cache Hit Rate\" : myArray[361], \n \n \"Total L2 Calls\" :myArray[362], \n \"L2 Read Calls\" : myArray[363],\n \"L2 Write Calls\" : myArray[364],\n \"L2 Read Hits\" : myArray[365],\n \"L2 Read Misses\" : myArray[366],\n \"L2 Write Hits\" : myArray[367],\n \"L2 Write Misses\" : myArray[368],\n \"L2 Miss Rate\" : myArray[369], \n \n \"Total Bus Traffic\" : myArray[370], \n \"Bus Read Hits\" : myArray[371],\n \"Bus Read Misses\" : myArray[372], \n \"Bus Write Hits\" : myArray[373], \n \"Bus Write Misses\" : myArray[374], \n \"Cache-to-Cache\" : myArray[375], \n \"Reads From Memory\" : myArray[376]\n }, \n {\n \"Core\" : myArray[377], //core 12\n \"Total Instructions\" : myArray[378], \n \"Total Read Instructions\" : myArray[379],\n \"Total Write Instructions\" : myArray[380], \n \"L1 Read Hits\" : myArray[381], \n \"L1 Read Misses\" : myArray[382], \n \"L1 Write Hits\" : myArray[383], \n \"L1 Write Misses\" : myArray[384],\n \"L1 Miss Rate\" : myArray[385],\n \n \"Victim Lines\" : myArray[386], \n \"Victim Cache Calls\" : myArray[387], \n \"Victim Cache Hits\" : myArray[388], \n \"Victim Cache Misses\" : myArray[389], \n \"Victim Cache Read Hits\" : myArray[390], \n \"Victim Cache Read Misses\" : myArray[391], \n \"Victim Cache Write Hits\" : myArray[392], \n \"Victim Cache Write Misses\" : myArray[393],\n \"Victim Cache Hit Rate\" : myArray[394], \n \n \"Total L2 Calls\" :myArray[395], \n \"L2 Read Calls\" : myArray[396],\n \"L2 Write Calls\" : myArray[397],\n \"L2 Read Hits\" : myArray[398],\n \"L2 Read Misses\" : myArray[399],\n \"L2 Write Hits\" : myArray[400],\n \"L2 Write Misses\" : myArray[401],\n \"L2 Miss Rate\" : myArray[402], \n \n \"Total Bus Traffic\" : myArray[403], \n \"Bus Read Hits\" : myArray[404],\n \"Bus Read Misses\" : myArray[405], \n \"Bus Write Hits\" : myArray[406], \n \"Bus Write Misses\" : myArray[407], \n \"Cache-to-Cache\" : myArray[408], \n \"Reads From Memory\" : myArray[409]\n }, \n {\n \"Core\" : myArray[410], //core 13\n \"Total Instructions\" : myArray[411], \n \"Total Read Instructions\" : myArray[412],\n \"Total Write Instructions\" : myArray[413], \n \"L1 Read Hits\" : myArray[414], \n \"L1 Read Misses\" : myArray[415], \n \"L1 Write Hits\" : myArray[416], \n \"L1 Write Misses\" : myArray[417],\n \"L1 Miss Rate\" : myArray[418],\n \n \"Victim Lines\" : myArray[419], \n \"Victim Cache Calls\" : myArray[420], \n \"Victim Cache Hits\" : myArray[421], \n \"Victim Cache Misses\" : myArray[422], \n \"Victim Cache Read Hits\" : myArray[423], \n \"Victim Cache Read Misses\" : myArray[424], \n \"Victim Cache Write Hits\" : myArray[425], \n \"Victim Cache Write Misses\" : myArray[426],\n \"Victim Cache Hit Rate\" : myArray[427], \n \n \"Total L2 Calls\" :myArray[428], \n \"L2 Read Calls\" : myArray[429],\n \"L2 Write Calls\" : myArray[430],\n \"L2

Read Hits\" : myArray[431],\n \"L2 Read Misses\" : myArray[432],\n \"L2 Write Hits\" : myArray[433],\n \"L2 Write Misses\" : myArray[434],\n \"L2 Miss Rate\" : myArray[435], \n \n \"Total Bus Traffic\" : myArray[436], \n \"Bus Read Hits\" : myArray[437],\n \"Bus Read Misses\" : myArray[438], \n \"Bus Write Hits\" : myArray[439], \n \"Bus Write Misses\" : myArray[440], \n \"Cache-to-Cache\" : myArray[441], \n \"Reads From Memory\" : myArray[442]\n }, \n {\n \"Core\" : myArray[443], //core 14\n \"Total Instructions\" : myArray[444], \n \"Total Read Instructions\" : myArray[445],\n \"Total Write Instructions\" : myArray[446], \n \"L1 Read Hits\" : myArray[447], \n \"L1 Read Misses\" : myArray[448], \n \"L1 Write Hits\" : myArray[449], \n \"L1 Write Misses\" : myArray[450],\n \"L1 Miss Rate\" : myArray[451],\n \n \"Victim Lines\" : myArray[452], \n \"Victim Cache Calls\" : myArray[453], \n \"Victim Cache Hits\" : myArray[454], \n \"Victim Cache Misses\" : myArray[455], \n \"Victim Cache Read Hits\" : myArray[456], \n \"Victim Cache Read Misses\" : myArray[457], \n \"Victim Cache Write Hits\" : myArray[458], \n \"Victim Cache Write Misses\" : myArray[459],\n \"Victim Cache Hit Rate\" : myArray[460], \n \n \"Total L2 Calls\" :myArray[461], \n \"L2 Read Calls\" : myArray[462],\n \"L2 Write Calls\" : myArray[463],\n \"L2 Read Hits\" : myArray[464],\n \"L2 Read Misses\" : myArray[465],\n \"L2 Write Hits\" : myArray[466],\n \"L2 Write Misses\" : myArray[467],\n \"L2 Miss Rate\" : myArray[468], \n \n \"Total Bus Traffic\" : myArray[469], \n \"Bus Read Hits\" : myArray[470],\n \"Bus Read Misses\" : myArray[471], \n \"Bus Write Hits\" : myArray[472], \n \"Bus Write Misses\" : myArray[473], \n \"Cache-to-Cache\" : myArray[474], \n \"Reads From Memory\" : myArray[475]\n }, \n {\n \"Core\" : myArray[476], //core 15\n \"Total Instructions\" : myArray[477], \n \"Total Read Instructions\" : myArray[478],\n \"Total Write Instructions\" : myArray[479], \n \"L1 Read Hits\" : myArray[480], \n \"L1 Read Misses\" : myArray[481], \n \"L1 Write Hits\" : myArray[482], \n \"L1 Write Misses\" : myArray[483],\n \"L1 Miss Rate\" : myArray[484],\n \n \"Victim Lines\" : myArray[485], \n \"Victim Cache Calls\" : myArray[486], \n \"Victim Cache Hits\" : myArray[487], \n \"Victim Cache Misses\" : myArray[488], \n \"Victim Cache Read Hits\" : myArray[489], \n \"Victim Cache Read Misses\" : myArray[490], \n \"Victim Cache Write Hits\" : myArray[491], \n \"Victim Cache Write Misses\" : myArray[492],\n \"Victim Cache Hit Rate\" : myArray[493], \n \n \"Total L2 Calls\" :myArray[494], \n \"L2 Read Calls\" : myArray[495],\n \"L2 Write Calls\" : myArray[496],\n \"L2 Read Hits\" : myArray[497],\n \"L2 Read Misses\" : myArray[498],\n \"L2 Write Hits\" : myArray[499],\n \"L2 Write Misses\" : myArray[500],\n \"L2 Miss Rate\" : myArray[501], \n \n \"Total Bus Traffic\" : myArray[502], \n \"Bus Read Hits\" : myArray[503],\n \"Bus Read Misses\" : myArray[504], \n \"Bus Write Hits\" : myArray[505], \n \"Bus Write Misses\" : myArray[506], \n \"Cache-to-Cache\" : myArray[507], \n \"Reads From Memory\" : myArray[508]\n }, \n {\n \"Core\" : myArray[509], //core 16\n \"Total Instructions\" : myArray[510], \n \"Total Read Instructions\" : myArray[511],\n \"Total Write Instructions\" : myArray[512], \n \"L1 Read Hits\" : myArray[513], \n \"L1 Read Misses\" : myArray[514], \n \"L1 Write Hits\" : myArray[515], \n \"L1 Write Misses\" : myArray[516],\n \"L1 Miss Rate\" : myArray[517],\n \n \"Victim Lines\" :

myArray[518], \n \"Victim Cache Calls\" : myArray[519], \n \"Victim Cache Hits\" : myArray[520], \n \"Victim Cache Misses\" : myArray[521], \n \"Victim Cache Read Hits\" : myArray[522], \n \"Victim Cache Read Misses\" : myArray[523], \n \"Victim Cache Write Hits\" : myArray[524], \n \"Victim Cache Write Misses\" : myArray[525],\n \"Victim Cache Hit Rate\" : myArray[526], \n \n \"Total L2 Calls\" :myArray[527], \n \"L2 Read Calls\" : myArray[528],\n \"L2 Write Calls\" : myArray[529],\n \"L2 Read Hits\" : myArray[530],\n \"L2 Read Misses\" : myArray[531],\n \"L2 Write Hits\" : myArray[532],\n \"L2 Write Misses\" : myArray[533],\n \"L2 Miss Rate\" : myArray[534], \n \n \"Total Bus Traffic\" : myArray[535], \n \"Bus Read Hits\" : myArray[536],\n \"Bus Read Misses\" : myArray[537], \n \"Bus Write Hits\" : myArray[538], \n \"Bus Write Misses\" : myArray[539], \n \"Cache-to-Cache\" : myArray[540], \n \"Reads From Memory\" : myArray[541]\n }];\n return newMsg;\n }\n}\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 1010,
    "y": 540,
    "wires": [
      [
        "1cdb6f7cda794e32",
        "16f158f9fe629a47",
        "4c879448d2252454"
      ]
    ]
  },
  {
    "id": "15ee83bf52d787cf",
    "type": "ui_dropdown",
    "z": "0b67313e235aaf0e",
    "name": "",
    "label": "L1 Victim Cache",
    "tooltip": "",
    "place": "Select option",
    "group": "866972eb095a975d",
    "order": 9,
    "width": 0,
    "height": 0,
    "passthru": true,
    "multiple": false,
    "options": [
      {
        "label": "0 Victim Lines",
        "value": "0",

"type": "str"
},
{
"label": "1 Victim Line",
"value": "1",
"type": "str"
},
{
"label": "2 Victim Lines",
"value": "2",
"type": "str"
},
{
"label": "3 Victim Lines",
"value": "3",
"type": "str"
},
{
"label": "4 Victim Lines",
"value": "4",
"type": "str"
},
{
"label": "5 Victim Lines",
"value": "5",
"type": "str"
},
{
"label": "6 Victim Lines",
"value": "6",
"type": "str"
},
{
"label": "7 Victim Lines",
"value": "7",
"type": "str"
},
{
"label": "8 Victim Lines",
"value": "8",
"type": "str"
},
{
"label": "9 Victim Lines",
"value": "9",
"type": "str"

```
    },
    {
        "label": "10 Victim Lines",
        "value": "10",
        "type": "str"
    },
    {
        "label": "11 Victim Lines",
        "value": "11",
        "type": "str"
    },
    {
        "label": "12 Victim Lines",
        "value": "12",
        "type": "str"
    },
    {
        "label": "13 Victim Lines",
        "value": "13",
        "type": "str"
    },
    {
        "label": "14 Victim Lines",
        "value": "14",
        "type": "str"
    },
    {
        "label": "15 Victim Lines",
        "value": "15",
        "type": "str"
    },
    {
        "label": "16 Victim Lines",
        "value": "16",
        "type": "str"
    }
],
"payload": "",
"topic": "topic",
"topicType": "msg",
"className": "",
"x": 180,
"y": 480,
"wires": [
    [
        "4933563ad970c2c5"
```

```
        ]
      ]
    },
    {
      "id": "4933563ad970c2c5",
      "type": "function",
      "z": "0b67313e235aaf0e",
      "name": "setVictimCache",
      "func": "var victimLines = msg.payload;\n\nglobal.set(\"victimLines\",
victimLines);\nmsg.payload = global.get(\"victimLines\");\n\n//msg.payload =
global.payload;//global.get(\"writePolicy\");\nreturn msg;",
      "outputs": 1,
      "noerr": 0,
      "initialize": "",
      "finalize": "",
      "libs": [],
      "x": 360,
      "y": 480,
      "wires": [
        []
      ]
    },
    {
      "id": "1cdb6f7cda794e32",
      "type": "ui_table",
      "z": "0b67313e235aaf0e",
      "group": "3623c6559b1d0b9b",
      "name": "Core Results Table",
      "order": 3,
      "width": "18",
      "height": "4",
      "columns": [
        {
          "field": "Core",
          "title": "Core",
          "width": "10px",
          "align": "center",
          "formatter": "plaintext",
          "formatterParams": {
            "target": "_blank"
          }
        },
        {
          "field": "Total Instructions",
          "title": "Total Instructions",
          "width": "20px",
```

```
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Total Read Instructions",
    "title": "Read Instructions",
    "width": "22px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Total Write Instructions",
    "title": "Write Instructions",
    "width": "22px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "L1 Read Hits",
    "title": "L1 Read Hits",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "L1 Read Misses",
    "title": "L1 Read Misses",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
```

```
{
   "field": "L1 Write Hits",
   "title": "L1 Write Hits",
   "width": "20px",
   "align": "center",
   "formatter": "plaintext",
   "formatterParams": {
      "target": "_blank"
   }
},
{
   "field": "L1 Write Misses",
   "title": "L1 Write Misses",
   "width": "20px",
   "align": "center",
   "formatter": "plaintext",
   "formatterParams": {
      "target": "_blank"
   }
},
{
   "field": "L1 Miss Rate",
   "title": "L1 Miss Rate",
   "width": "20px",
   "align": "center",
   "formatter": "plaintext",
   "formatterParams": {
      "target": "_blank"
   }
},
{
   "field": "Victim Lines",
   "title": "Victim Lines",
   "width": "20px",
   "align": "center",
   "formatter": "plaintext",
   "formatterParams": {
      "target": "_blank"
   }
},
{
   "field": "Victim Cache Calls",
   "title": "Victim Calls",
   "width": "20px",
   "align": "center",
   "formatter": "plaintext",
```

```
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Victim Cache Hits",
    "title": "Victim Hits",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Victim Cache Misses",
    "title": "Victim Misses",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Victim Cache Read Hits",
    "title": "Victim Read Hits",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Victim Cache Read Misses",
    "title": "Victim Read Misses",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Victim Cache Write Hits",
```

```
    "title": "Victim Write Hits",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Victim Cache Write Misses",
    "title": "Victim Write Misses",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Victim Cache Hit Rate",
    "title": "Victim Cache Hit Rate",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Total L2 Calls",
    "title": "Total L2 Calls",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "L2 Read Calls",
    "title": "L2 Read Calls",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
```

```
    }
},
{
    "field": "L2 Write Calls",
    "title": "L2 Write Calls",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "L2 Read Hits",
    "title": "L2 Read Hits",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "L2 Read Misses",
    "title": "L2 Read Misses",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "L2 Write Hits",
    "title": "L2 Write Hits",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "L2 Write Misses",
    "title": "L2 Write Misses",
    "width": "20px",
```

```
            "align": "center",
            "formatter": "plaintext",
            "formatterParams": {
                "target": "_blank"
            }
        },
        {
            "field": "L2 Miss Rate",
            "title": "L2 Miss Rate",
            "width": "20px",
            "align": "center",
            "formatter": "plaintext",
            "formatterParams": {
                "target": "_blank"
            }
        },
        {
            "field": "Global Miss Rate",
            "title": "Global Miss Rate",
            "width": "20px",
            "align": "center",
            "formatter": "plaintext",
            "formatterParams": {
                "target": "_blank"
            }
        }
    ],
    "outputs": 0,
    "cts": false,
    "x": 1250,
    "y": 500,
    "wires": []
},
{
    "id": "16f158f9fe629a47",
    "type": "ui_table",
    "z": "0b67313e235aaf0e",
    "group": "4a8ce417ea9189ee",
    "name": "Cache Table",
    "order": 1,
    "width": "18",
    "height": "3",
    "columns": [
        {
            "field": "Core Count",
            "title": "Core Count",
```

```
    "width": "14px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "L1 Cache Size",
    "title": "L1 Cache Size",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "L1 Block Size",
    "title": "L1 Block Size",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "L1 Associativity",
    "title": "L1 Associativity",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "L1 Replacement Policy",
    "title": "L1 Replacement Policy",
    "width": "21px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
```

```
},
{
  "field": "L2 Cache Size",
  "title": "L2 Cache Size",
  "width": "20px",
  "align": "center",
  "formatter": "plaintext",
  "formatterParams": {
    "target": "_blank"
  }
},
{
  "field": "L2 Block Size",
  "title": "L2 Block Size",
  "width": "20px",
  "align": "center",
  "formatter": "plaintext",
  "formatterParams": {
    "target": "_blank"
  }
},
{
  "field": "L2 Associativity",
  "title": "L2 Associativity ",
  "width": "20px",
  "align": "center",
  "formatter": "plaintext",
  "formatterParams": {
    "target": "_blank"
  }
},
{
  "field": "L2 Replacement Policy",
  "title": "L2 Replacement Policy",
  "width": "21px",
  "align": "center",
  "formatter": "plaintext",
  "formatterParams": {
    "target": "_blank"
  }
},
{
  "field": "Coherence Protocol",
  "title": "Coherence Protocol",
  "width": "20px",
  "align": "center",
```

```
            "formatter": "plaintext",
            "formatterParams": {
                "target": "_blank"
            }
        },
        {
            "field": "Write Policy",
            "title": "Write Policy",
            "width": "20px",
            "align": "center",
            "formatter": "plaintext",
            "formatterParams": {
                "target": "_blank"
            }
        },
        {
            "field": "Trace File",
            "title": "Trace File",
            "width": "20px",
            "align": "center",
            "formatter": "plaintext",
            "formatterParams": {
                "target": "_blank"
            }
        }
    ],
    "outputs": 0,
    "cts": false,
    "x": 1230,
    "y": 540,
    "wires": []
},
{
    "id": "4c879448d2252454",
    "type": "ui_table",
    "z": "0b67313e235aaf0e",
    "group": "faf997caa826b9e3",
    "name": "Bus Table",
    "order": 7,
    "width": "18",
    "height": "4",
    "columns": [
        {
            "field": "Core",
            "title": "Core",
            "width": "10px",
```

```
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Total Bus Traffic",
    "title": "Bus Traffic",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Cache-to-Cache",
    "title": "Cache-to-Cache",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Bus Read Hits",
    "title": "Bus Read Hits",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
{
    "field": "Bus Read Misses",
    "title": "Bus Read Misses",
    "width": "20px",
    "align": "center",
    "formatter": "plaintext",
    "formatterParams": {
        "target": "_blank"
    }
},
```

```
        {
            "field": "Bus Write Hits",
            "title": "Bus Write Hits",
            "width": "20px",
            "align": "center",
            "formatter": "plaintext",
            "formatterParams": {
                "target": "_blank"
            }
        },
        {
            "field": "Bus Write Misses",
            "title": "Bus Write Misses",
            "width": "20px",
            "align": "center",
            "formatter": "plaintext",
            "formatterParams": {
                "target": "_blank"
            }
        },
        {
            "field": "Reads From Memory",
            "title": "Reads From Memory",
            "width": "24px",
            "align": "center",
            "formatter": "plaintext",
            "formatterParams": {
                "target": "_blank"
            }
        }
    ],
    "outputs": 0,
    "cts": false,
    "x": 1220,
    "y": 580,
    "wires": []
},
{
    "id": "1aee994689733115",
    "type": "ui_media",
    "z": "0b67313e235aaf0e",
    "group": "866972eb095a975d",
    "name": "UTRGV-LOGO",
    "width": "8",
    "height": "2",
    "order": 1,
```

"category": "Logo",
"file": "utrgv_NRC-SIM_banner.png",
"layout": "center",
"showcontrols": true,
"loop": true,
"onstart": false,
"scope": "local",
"tooltip": "",
"x": 280,
"y": 140,
"wires": [
[]
]
},
{
"id": "f00924766d7dc2a6",
"type": "ui_media",
"z": "0b67313e235aaf0e",
"group": "317c654e36df64a9",
"name": "Help",
"width": "26",
"height": "13",
"order": 1,
"category": "Help",
"file": "NRC-SIM_help.png",
"layout": "adjust",
"showcontrols": true,
"loop": true,
"onstart": false,
"scope": "local",
"tooltip": "",
"x": 250,
"y": 80,
"wires": [
[]
]
},
{
"id": "da64291b51057b98",
"type": "ui_form",
"z": "0b67313e235aaf0e",
"name": "",
"label": "",
"group": "7a9ed32ad002511e",
"order": 0,
"width": 0,

```
    "height": 0,
    "options": [
        {
            "label": "Core Count",
            "value": "traceFileCoreCount",
            "type": "number",
            "required": true,
            "rows": null
        }
    ],
    "formValue": {
        "traceFileCoreCount": ""
    },
    "payload": "",
    "submit": "Submit",
    "cancel": "Cancel",
    "topic": "topic",
    "topicType": "msg",
    "splitLayout": "",
    "className": "",
    "x": 70,
    "y": 920,
    "wires": [
        [
            "7923ba3c171cf1c3"
        ]
    ]
},
{
    "id": "fb79c0c82ac5decc",
    "type": "ui_dropdown",
    "z": "0b67313e235aaf0e",
    "name": "",
    "label": "Benchmark Program",
    "tooltip": "",
    "place": "Select option",
    "group": "7a9ed32ad002511e",
    "order": 1,
    "width": 0,
    "height": 0,
    "passthru": true,
    "multiple": false,
    "options": [
        {
            "label": "Blackscholes",
            "value": "blackscholes",
```

"type": "str"
},
{
"label": "Canneal",
"value": "canneal",
"type": "str"
},
{
"label": "Fluid Animate",
"value": "fluidanimate",
"type": "str"
},
{
"label": "Radix",
"value": "radix",
"type": "str"
},
{
"label": "Streamcluster",
"value": "streamcluster",
"type": "str"
},
{
"label": "Raytrace",
"value": "rtview",
"type": "str"
},
{
"label": "Ocean_CP",
"value": "ocean_cp",
"type": "str"
},
{
"label": "Water-nSquared",
"value": "water_nsquared",
"type": "str"
},
{
"label": "Freqmine",
"value": "freqmine",
"type": "str"
}
],
"payload": "",
"topic": "topic",
"topicType": "msg",

"className": "",
"x": 120,
"y": 860,
"wires": [
[
"53b6fefb28dd10b4"
]
]
},
{
"id": "6288fe9a9229eb81",
"type": "file in",
"z": "0b67313e235aaf0e",
"name": "pinatrace.txt",
"filename": "pinatrace.txt",
"filenameType": "str",
"format": "utf8",
"chunk": false,
"sendError": false,
"encoding": "none",
"allProps": false,
"x": 490,
"y": 920,
"wires": [
[
"348597880f93fe21"
]
]
},
{
"id": "348597880f93fe21",
"type": "function",
"z": "0b67313e235aaf0e",
"name": "setCoreCountPinatraceCPP",
"func": "let coreCount = global.get(\"traceFileCoreCount\");\n\nlet s = msg.payload;\nlet myArray = s.split(/(?:,| |\\t)+/);\n//let myArrayCopy = myArray;\nlet newCoreCount = \"int n = \" + coreCount.traceFileCoreCount + \";\";\n\n/////myArray[5] = \"int n = \" + coreCount.traceFileCoreCount + \";\";\n//myArray[5] = myArrayCopy;\n//myArray.toString();\nmyArray = msg.payload.replace(\";;\", newCoreCount);\n\nmsg.payload = myArray;\nreturn msg;\n\n\n//for (let i = 0; i < 4; i++) {\n // myArray[i] = \"\";\n//}\n\ns = myArray;\ns = s.join('');\n\ns = s.replace(\"2\", coreCount);\n\ns = s.replace((new RegExp(\"VOID\", 'g')), \"VOID \");\ns = s.replace((new RegExp(\"int\", 'g')), \"int \");\ns = s.replace(\"num\", \"num,\");\ns = s.replace(\"num2\", \"num2,\");\ns = s.replace(\"INSins\", \"INS ins,\");\ns = s.replace(\"UINT32memOperands\", \"UINT32 memOperands\");\ns = s.replace(\"UINT32memOp\", \"UINT32 memOp\")\ns = s.replace(\"INT32code\", \"INT32 code, \")\ns = s.replace(\"INT32Usage\", \"INT32 Usage\");\ns

= s.replace(\"argcchar*argv[])\", \"argc, char* argv[])\");\ns = s.replace(\"argcargv\", \"argc, argv\");\ns = s.replace(\"Instruction0\", \"Instruction, 0\");\ns = s.replace(\"Fini0\", \"Fini, 0\");\ns = s.replace((new RegExp(\"fprint f\", 'g')), \"fprintf\");\ns = s.replace(\"FILE*trace\", \"FILE* trace\");\ns = s.replace(\"numVOID\", \"num, VOID\");\ns = s.replace((new RegExp('trace\"', 'g')), 'trace, \"');\ns = s.replace(\"numaddr\", \", num, addr\");\ns = s.replace(\"num2addr\", \", num2, addr\");\ns = s.replace((new RegExp(\"insmemOp\", 'g')), \"ins, memOp\");\ns = s.replace(\"insIPOINT_BEFORE(AFUNPTR)\", \"ins, IPOINT_BEFORE, (AFUNPTR)\");\ns = s.replace(\"insIPOINT_BEFORE(AFUNPTR)\", \"ins, IPOINT_BEFORE, (AFUNPTR)\");\ns = s.replace(\"RecordMemWriteIARG_INST_PTRIARG_MEMORYOP_EAmemOpIARG_END\",\n   \"RecordMemWrite, IARG_INST_PTR, IARG_MEMORYOP_EA, memOp, IARG_END\");\ns = s.replace(\"RecordMemReadIARG_INST_PTRIARG_MEMORYOP_EAmemOpIARG_END\",\n   \"RecordMemRead, IARG_INST_PTR, IARG_MEMORYOP_EA, memOp, IARG_END\");\ns = s.replace('trace\"#eof', 'trace, \"#eof');\ns = s.replace(\"returnUsage\", \"return Usage\");\ns = s.replace(\"return0\", \"return 0\");\ns = s.replace('out\"\"w', 'out\", \"w');\ns = s.replace(\"%dr%p\", \"%d r %p\");\ns = s.replace(\"%dw%p\", \"%d w %p\");\n\n\n\nmsg.payload = s;\nmsg.payload[0].replace(/\"/g, \");\n\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 720,
        "y": 920,
        "wires": [
            [
                "db58996c6c29438e"
            ]
        ]
    },
    {
        "id": "db58996c6c29438e",
        "type": "file",
        "z": "0b67313e235aaf0e",
        "name": "pinatrace.cpp",
        "filename": "pinatrace.cpp",
        "filenameType": "str",
        "appendNewline": true,
        "createDir": true,
        "overwriteFile": "true",
        "encoding": "none",
        "x": 960,
        "y": 920,
        "wires": [
            [

```
            "d13488e00db366d3"
          ]
        ]
    },
    {
        "id": "d13488e00db366d3",
        "type": "exec",
        "z": "0b67313e235aaf0e",
        "command": "make pinatrace.test",
        "addpay": "",
        "append": "",
        "useSpawn": "false",
        "timer": "",
        "winHide": false,
        "oldrc": false,
        "name": "",
        "x": 1180,
        "y": 920,
        "wires": [
          [
            "db97dfe9f0558dc2"
          ],
          [],
          []
        ]
    },
    {
        "id": "7923ba3c171cf1c3",
        "type": "function",
        "z": "0b67313e235aaf0e",
        "name": "setTraceFileCoreCount",
        "func": "var traceFileCoreCount = msg.payload;\n\nglobal.set(\"traceFileCoreCount\",
traceFileCoreCount);\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 280,
        "y": 920,
        "wires": [
          [
            "6288fe9a9229eb81"
          ]
        ]
    },
```

{
"id": "af840e84e946b0e2",
"type": "function",
"z": "0b67313e235aaf0e",
"name": "outputMessage",
"func": "msg.payload = \"Trace File Generated\";\nreturn msg;",
"outputs": 1,
"noerr": 0,
"initialize": "// Code added here will be run once\n// whenever the node is started.\n//let execComplete = { payload: \"\" };\n//execComplete.payload = \"Trace File Generated\";\n\n//return execComplete;",
"finalize": "",
"libs": [],
"x": 1760,
"y": 900,
"wires": [
[
"b40f08124ed84cfc",
"6e4f736c272fc377"
]
]
},
{
"id": "b40f08124ed84cfc",
"type": "ui_text",
"z": "0b67313e235aaf0e",
"group": "e3e2e42938dd7cfb",
"order": 1,
"width": "6",
"height": "3",
"name": "",
"label": "Status:",
"format": "{{msg.payload}}",
"layout": "col-center",
"className": "",
"style": false,
"font": "",
"fontSize": 16,
"color": "#000000",
"x": 2100,
"y": 900,
"wires": []
},
{
"id": "57069c62eca42a5c",
"type": "exec",

"z": "0b67313e235aaf0e",
"command": "",
"addpay": "payload",
"append": "",
"useSpawn": "false",
"timer": "",
"winHide": false,
"oldrc": false,
"name": "",
"x": 1590,
"y": 900,
"wires": [
[
"af840e84e946b0e2"
],
[],
[]
]
},
{
"id": "53b6fefb28dd10b4",
"type": "function",
"z": "0b67313e235aaf0e",
"name": "setTraceFileBenchmark",
"func": "var traceFileBenchmark = msg.payload;\n\nglobal.set(\"traceFileBenchmark\", traceFileBenchmark);\nreturn msg;",
"outputs": 1,
"noerr": 0,
"initialize": "",
"finalize": "",
"libs": [],
"x": 370,
"y": 860,
"wires": [
[]
]
},
{
"id": "db97dfe9f0558dc2",
"type": "function",
"z": "0b67313e235aaf0e",
"name": "pinCommand",
"func": "let traceFileBenchmark = global.get(\"traceFileBenchmark\");\n\nlet pinCommand = `../../../pin -t obj-intel64/pinatrace.so -- ./${traceFileBenchmark}`\nmsg.payload = pinCommand;\n\nreturn msg;",
"outputs": 1,

```
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 1420,
    "y": 900,
    "wires": [
        [
            "57069c62eca42a5c"
        ]
    ]
},
{
    "id": "6e4f736c272fc377",
    "type": "delay",
    "z": "0b67313e235aaf0e",
    "name": "",
    "pauseType": "delay",
    "timeout": "3",
    "timeoutUnits": "seconds",
    "rate": "1",
    "nbRateUnits": "1",
    "rateUnits": "second",
    "randomFirst": "1",
    "randomLast": "5",
    "randomUnits": "seconds",
    "drop": false,
    "allowrate": false,
    "outputs": 1,
    "x": 1940,
    "y": 960,
    "wires": [
        [
            "6bf8bd5b0ed7fb38"
        ]
    ]
},
{
    "id": "6bf8bd5b0ed7fb38",
    "type": "function",
    "z": "0b67313e235aaf0e",
    "name": "resetOutputMessage",
    "func": "msg.payload = ";\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
```

```
    "finalize": "",
    "libs": [],
    "x": 2140,
    "y": 960,
    "wires": [
        [
            "b40f08124ed84cfc"
        ]
    ]
},
{
    "id": "866972eb095a975d",
    "type": "ui_group",
    "name": "Input",
    "tab": "e8e17ac1438e8612",
    "order": 3,
    "disp": true,
    "width": "8",
    "collapse": false,
    "className": ""
},
{
    "id": "3623c6559b1d0b9b",
    "type": "ui_group",
    "name": "Cache Simulation Results",
    "tab": "e8e17ac1438e8612",
    "order": 5,
    "disp": true,
    "width": "18",
    "collapse": false,
    "className": ""
},
{
    "id": "4a8ce417ea9189ee",
    "type": "ui_group",
    "name": "Cache Configuration",
    "tab": "e8e17ac1438e8612",
    "order": 4,
    "disp": true,
    "width": "18",
    "collapse": false,
    "className": ""
},
{
    "id": "faf997caa826b9e3",
    "type": "ui_group",
```

"name": "Bus Traffic Results",
"tab": "e8e17ac1438e8612",
"order": 6,
"disp": true,
"width": "18",
"collapse": false,
"className": ""
},
{
"id": "317c654e36df64a9",
"type": "ui_group",
"name": "NRC-SIM Help",
"tab": "3b2a8c09d1bbb59e",
"order": 7,
"disp": true,
"width": "26",
"collapse": false,
"className": ""
},
{
"id": "7a9ed32ad002511e",
"type": "ui_group",
"name": "Input",
"tab": "e27ddf2a5c3ffd7a",
"order": 1,
"disp": true,
"width": "7",
"collapse": false,
"className": ""
},
{
"id": "e3e2e42938dd7cfb",
"type": "ui_group",
"name": "Output",
"tab": "e27ddf2a5c3ffd7a",
"order": 2,
"disp": true,
"width": "6",
"collapse": false,
"className": ""
},
{
"id": "e8e17ac1438e8612",
"type": "ui_tab",
"name": "NRC-SIM",
"icon": "memory",

```
        "order": 2,
        "disabled": false,
        "hidden": false
    },
    {
        "id": "3b2a8c09d1bbb59e",
        "type": "ui_tab",
        "name": "Help",
        "icon": "mi-help_outline",
        "order": 5,
        "disabled": false,
        "hidden": false
    },
    {
        "id": "e27ddf2a5c3ffd7a",
        "type": "ui_tab",
        "name": "Generate Trace File",
        "icon": "dashboard",
        "order": 3,
        "disabled": false,
        "hidden": false
    }
]
```

BIOGRAPHICAL SKETCH

Ezequiel Treviño earned his Master of Science in Engineering in Electrical Engineering from The University of Texas Rio Grande Valley in August 2023 and his Bachelor of Science in Computer Engineering from The University of Texas Rio Grande Valley in December 2019. He has worked as a Graduate Teaching Assistant, providing aid to undergraduate students across a variety of courses, and as a Graduate Research Assistant, helping design and develop micro-credential courses in high-performance computing and MPI/OpenMP for The University of Texas Rio Grande. He can be reached via the following email address: ezequiel.trevino01@gmail.com.