# Performance Evaluation of Intel and AMD Memory Hierarchies Using a Simulation-driven Approach With Gem5

**João Victor Amorim Vieira, Matheus Alcântara Souza, Henrique Cota de Freitas**

*Computer Architecture and Parallel Processing Team (CArT)*
Departamento de Ciência da Computação – Pontifícia Universidade Católica de Minas Gerais
Belo Horizonte – MG – Brazil
joao.vieira.1275044@sga.pucminas.br, {matheusalcantara, cota}@pucminas.br

***Abstract.*** *Along with the necessity of more computational power, CPUs are evolving quickly. Companies are developing new generations of CPUs every 1–2 years, and many options compete for market share. Hence, researchers and companies must develop a method to compare and select the best processor that fits their needs. This paper presents a simulation-driven alternative for CPU Memory Hierarchy evaluation and comparison. Furthermore, this work compares two rival processors engineered by Intel and AMD. The investigation culminates in the observation that AMD's processor performs better with fewer cores for several cases, while Intel's equivalent exhibits enhanced results when all available cores are used.*

## 1. Introduction

Over the years, the demand for computational power has seen a significant surge, driven by the increasing reliance of businesses on powerful computer systems to meet their growing needs. This demand has given rise to the field of High-Performance Computing (HPC), which entails using and applying parallel computers to solve scientific, engineering, and technical problems [Chavarría-Miranda et al., 2012]. In this context, the market role of CPUs and GPUs becomes crucial.

In line with Moore's Law, the industry is expected to release a new processor approximately every 18 months. Moreover, numerous companies are dedicated to the development of new processors. Therefore, there are numerous options for a researcher or company to analyze. It is not feasible for a team to acquire every new processor generation and compare them to choose which to buy on a large scale.

To address the challenges imposed by the rapid evolution of processors and the large number of available options, it becomes crucial to establish an approach to compare and evaluate these solutions. In light of this dynamic environment, one can address the following questions: How can individuals keep pace with this rapid evolution? How can one effectively compare the newly launched solutions to determine the most suitable option for their specific needs? Based on this, this paper contributes to the CPU performance area by presenting a simulation-driven method for the CPU's memory hierarchy evaluation and comparing the newly launched CPUs developed by Intel and AMD.

This paper is organized as follows: Section 2 presents the related work, Section 3 presents the simulation details with a brief overview of the gem5 simulator since it was

João Victor is an undergraduate student in Computer Science. The authors thank FAPEMIG, CNPq and PUC Minas for their partial support in this research.

the simulator used in the paper, and follows with the details of the proposed architectures, Section 4 presents the evaluation of the results, and Section 5 concludes the paper.

## 2. Related Work

As simulation is widely used in architectural research, many options can make it difficult to choose one of them. Akram et al. [2019a] present a detailed survey of available computer simulators, comparing and classifying them based on information such as the detail of the simulation, scope of the target, and input needed. In the paper, there is no distinction between ISAs, therefore, it is a generic review of the simulators and each's strengths. Moreover, focused on X86, Akram et al. [2016] compare different simulators in detail. They are gem5 [Binkert et al., 2011], Multi2sim [Ubal et al., 2007], PTLsim [Yourst, 2007] and Sniper [Carlson et al., 2011]. Making evident strong points and limitations regarding flexibility, level of detail, user-friendliness, and simulation models. Finally, the experimental error and speed are compared. The paper concludes that the Sniper simulator is the better choice based on accuracy and simulation speed. However, Sniper does not include detailed processor modeling, multi-ISA support, full-system simulation support, and active development community, which are present in gem5. Consequently, this paper chooses to use gem5. Umeike et al. [2023] profile gem5, aiming at answering questions about how the simulator performance can be increased and the bottlenecks present in the code. The thorough profiling reveals an increase in performance up to three times when using processors with larger L1 cache and page sizes.

Vikas et al. [2014] measured cache misses, memory bandwidth, and system bus throughput in a 2-level cache hierarchy system running with ARM and ALPHA processors, varying the L1 cache size from 16 kB to 64 kB. For this experiment, the authors used the full-system gem5, which consists of the full emulation of a defined system using a proper disk image and a precompiled kernel, and the Splash-2 benchmark.

According to Endo et al. [2014], in- and out-of-order ARM cores were implemented and evaluated. In the paper, the gem5 simulated the Cortex-A8 and Cortex-A9 cores. Since the in-order ARM CPU was not implemented, the authors implemented it based on the out-of-order(O3) model in gem5 and compared it against real machines using ten benchmarks. Both models had an absolute error of 7%, showing themselves above the microarchitectural simulator's mean of 15%.

Gem5 is a state-of-the-art computer simulator. Therefore, it is widely used for experiments and research. Souza et al. [2017b] used full-system gem5 to assess different multicore cluster architectures to identify the best configuration for energy efficiency. They varied the number of cores, L2 cache size, and sharing strategies. Along with gem5, MCPAT and CACTI tools are used to estimate cache parameters. The results showed low-performance improvement by simply varying L2's sharing strategy. They also concluded that shared L2 multicore was more energy efficient than private L2 multicore. Moreover, Souza et al. [2018] used full-system gem5 to present a design space exploration over NoC-based manycore processor architectures with distributed and shared caches. The MPPA-256 processor was adopted as a reference point in the paper, considering its similar characteristics to their proposal. The tests varied the number of cores, and the size of the L2 and used 12 types of NoC (Network-on-Chip) topologies. They presented a clustered topology, which obtained performance gains of up to 30.56%, and reduced energy

consumption by up to 38.53%.

Charles et al. [2018] analyzed the impact of directory memory traffic and different cluster modes on NoC traffic and system performance. The authors employed full-system gem5 and the GARNET2.0 interconnection network model to make the simulations more realistic. They showed how to model the LLC/directory to memory traffic and compared different clusters and memory modes supported by CMPs. In addition, they demonstrated the impact of the work on four different cache coherence protocols.

Akram et al. [2019b] validated the gem5 simulator against the x86 architecture, more specifically the Core-I7 Haswell microarchitecture. Through their study, they could find various sources of inaccuracies in the x86 simulation and improve the mean error rate from 136% to 6% by applying modifications to gem5's source code.

Abudaqa et al. [2018] present a comparative study between ARM and x86 using in-order and out-of-order models and compare them using five benchmarks: Basicmath, Dijkstra, Qsort, Patricia, and Bitcount. The authors then compared consumed energy, throughput, CPI, and L2 miss rate while varying the CPU model and the size of L1 and L2 caches. The results show that, in most cases, ARM outperforms the x86 architecture. However, x86 showed itself better when used with the out-of-order processor model.

Diverging from earlier studies, this research focuses on simulating two processors sourced from distinct vendors. The primary goal is to present a method for CPU comparison using simulation-driven experiments. We comprehensively assess and juxtapose the processors' outcome, thereby facilitating a more profound grasp of their computational power, all achieved without the necessity of physical access, allowing the identification of the most suitable option for specific computational needs of researchers and companies.

## 3. Simulation

Simulations benefit flexibility when choosing the specifications of the desired architectures. However, at the same time, simulations present constraints, such as the real time to solution. Furthermore, simulations are not perfect since they are not real hardware. Therefore, there might be inaccuracies when simulating architectures in comparison with the real world [Endo et al., 2014; Akram et al., 2019b].

The simulations performed in this paper attempted to be as close to reality as possible while adapting to constraints imposed by the simulator. In the following subsections, the gem5 simulator and the computer architectures will be presented.

### 3.1. Gem5

Gem5 [Binkert et al., 2011] is a computer simulator framework that provides configuration, multiple ISAs, and different models of CPUs. In this paper, we opted for the O3CPU, which consists of a pipelined, out-of-order CPU. Using the O3CPU, we could collect more accurately the simulation details, although performing a slower simulation. Gem5 includes two system modes: System-Call Emulation(SE) and Full-System (FS). SE mode is more straightforward. The framework emulates only the system calls the running program makes, so it is unnecessary to worry about devices or OS-specific matters. FS is more complex, and in this mode, an entire system is simulated, i.e., user and kernel-level instructions are executed, and all devices are simulated. Therefore, for this mode, it is

necessary to provide a disk image and a pre-compiled kernel. After the machine boot, the desired workload can be run using a terminal, or a script can be previously informed.

For this paper, the full-system (FS) mode had to be used since the workloads were built with OpenMP, and gem5 does not natively support OpenMP system calls. The workloads had to be statically compiled, and to run FS, a pre-compiled Linux kernel version 5.4.49 and a disk image created using the QEMU tool were provided.

### 3.2. Proposed Simulated Architectures

The architectures used in this paper are based on two processors, the Intel Xeon w7-3455 and AMD EPYC 9224. Both are newly launched processors and are mostly used for servers' workloads. Since the processors are not physically available, we chose to simulate both. Some modifications had to be performed to fit our architectures into the simulator limitations. The following paragraph introduces these modifications.

Related to cache size, the gem5 imposes that caches should have a power of 2 size. Thus, Intel's processor's L1 and L3 caches had to be modified from 48kB to 32kB and 67.5MB to 64MB. FS mode presents a limitation, and there can be only one thread per core. Therefore, no SMT present on the original processors could be simulated. Both architectures used the same disk image and kernel, and the main memory size in all tests was 3GB. Table 1 summarizes the original architectures and the necessary modifications to simulate the processors.

| | Intel | Intel Simulated | AMD | AMD Simulated |
|---|---|---|---|---|
| # CPUS/ Threads | 24/48 | 24/24 | 24/48 | 24/24 |
| Clock Speed | 2.5GHz | 2.5GHz | 2.5GHz | 2.5GHz |
| L1 Cache | 24x32 kB (Instructions)<br>24x48 kB (Data) | 24x32 kB (Instructions)<br>24x32 kB (Data) | 24x32 kB (Instructions)<br>24x32 kB (Data) | 24x32 kB (Instructions)<br>24x32 kB (Data) |
| L2 Cache | 24x2MB | 24x2MB | 24x1MB | 24x1MB |
| L3 Cache | 67.5MB | 64MB | 2x32MB | 2x32MB |

**Table 1. Original and Simulated Architectures**

## 4. Result Evaluation

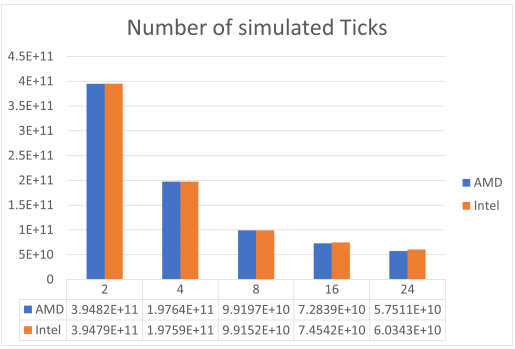
To evaluate the proposed architectures, the CAP Bench [Souza et al., 2017a] was used.

CAP Bench consists of seven parallel applications built with OpenMP. Each program has five input sizes: tiny, small, default, large, and huge. For this paper's purpose, we selected only three programs: Friendly Numbers (FN), K-Means (KM), and Features from accelerated segment test (FAST), with the tiny workload – due to time constraints. We chose these applications since all present different parallel patterns, i.e., MapReduce, Map, and Stencil, respectively, which could give us insights for future workload characterization in this simulation context.

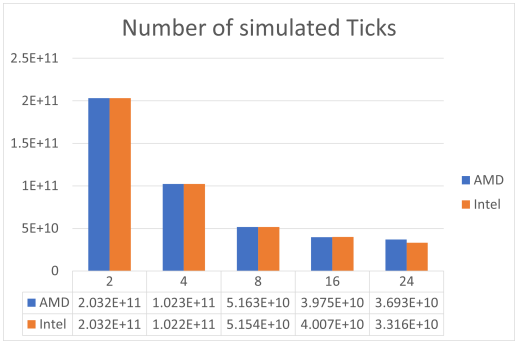
The FN kernel aims at comparing the abundancy of all numbers in a given interval. For the tiny workload, this interval is $8 \times 10^6 + 1$ to $8 \times 10^6 + 2^{12}$. KM implements the well-known K-Means clustering algorithm. The data points are evenly and randomly created. The tiny workload is $2^{12}$ points and 256 centroids. Finally, FAST is a corner detection method. The tiny workload uses a 2048 × 2048 pixels sized image. FAST and KM present irregular task loads, which means that the input may vary the data but not its size, and FN presents a regular task load.

The tests were split between a server with two Intel Xeon E5-2620 v2 processors and 64 GB of RAM and a virtual machine from Google Cloud with 8 cores and 32 GB of RAM. The experiments consisted of a strong scalability test using the tiny input set size from CAP Bench, with 2, 4, 8, 16, and 24 threads.

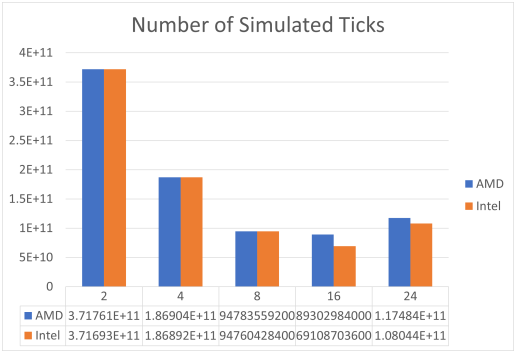
Figure 1 presents the number of ticks simulated based on the number of CPUs used. As we can see, both architectures scale well. Only when using 24 cores and running the KM algorithm, the number of ticks is higher than expected. Both processors show similar numbers, although Intel performs slightly better when using all the available cores, except when running the FAST application. Both FAST and FN applications perform as expected. A higher number of threads implies a smaller number of ticks. However, KM escapes this analysis since it only scales for a maximum of 16 cores.




(a) FAST Application




(b) Friendly Numbers Application



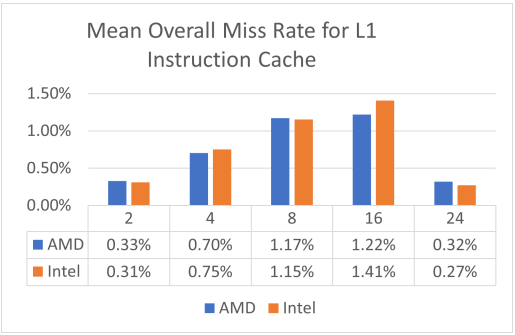

(c) K-Means Application

**Figure 1. Number of simulated Ticks**

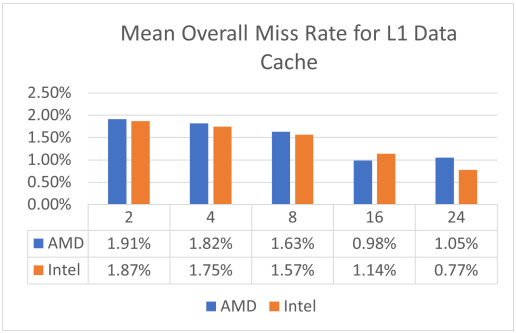
In Figure 2, we see the mean overall miss rate of the L1 cache separated into Instruction and Data Cache for the FN application. Since the L1 caches for both architectures were private, we decided to show a mean of the value, in order to simplify. The instruction cache miss increases until 16 cores but finds its lowest value using 24 cores for both Intel and AMD. Intel again shows itself slightly better using all the cores, but this time AMD performs better along the way, including in 16 cores. The data caches present a linear progression. This demonstrates that the application's scalability performs well on our architecture and tends to increase performance according to the number of threads.

Figures 3 and 4 show the same information using the FAST and KM applications. Both follow the same descending pattern based on the number of cores used. However,

the sharp drop in the miss rate in the L1 instruction caches shows that all instructions fit the cache size when using at least 8 cores. Since these cores have private caches, the more cores one uses, the more cache space is obtained. We see the same descending pattern regarding the data cache, showing good scalability.

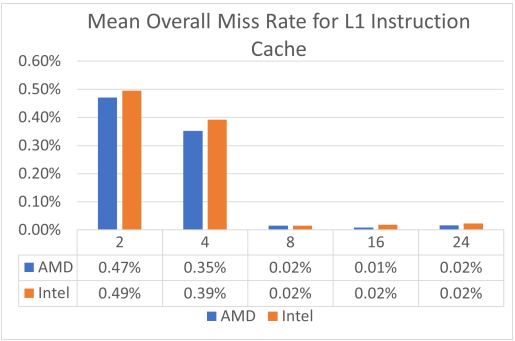



(a) Instruction Cache

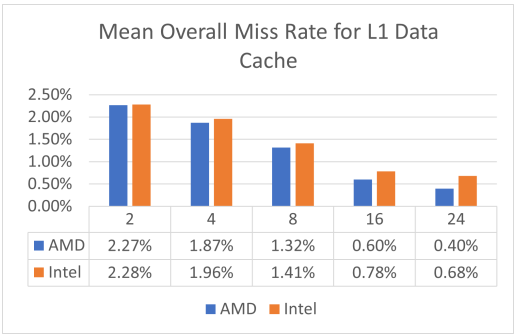



(b) Data Cache

**Figure 2. Mean Miss Rate of L1 Cache with the Friendly Numbers application**

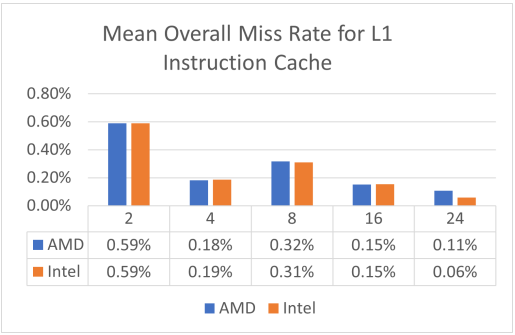



(a) Instruction Cache

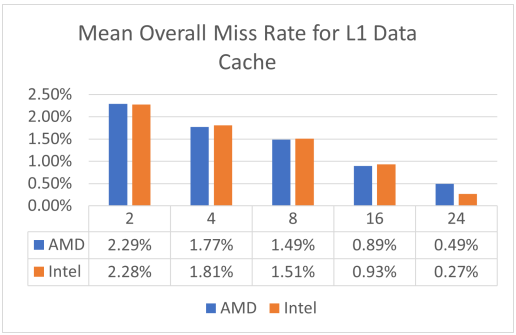



(b) Data Cache

**Figure 3. Miss Rate of L1 Cache with the FAST application**




(a) Instruction Cache

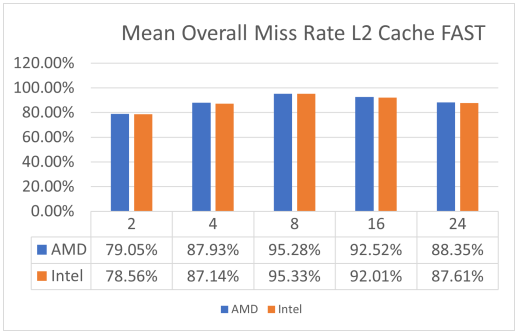



(b) Data Cache

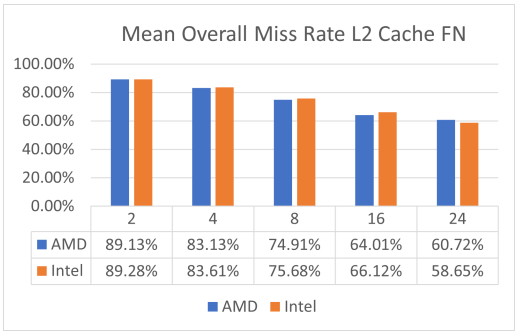
**Figure 4. Mean Miss Rate of L1 Cache with the K-Means application**

Figure 5 depicts the mean miss rate for L2 caches. Since both architectures have private L2 caches, we calculate their mean miss rate, as with the L1 caches. All three applications follow this same pattern and have a high rate of misses. The L2 is not used

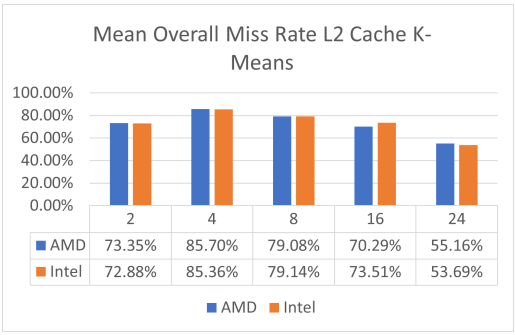
intensively, based on the low absolute number of accesses presented. Therefore, most misses refer to compulsory misses. However, AMD performs better with fewer cores, while Intel performs better with all 24 cores.




(a) FAST Application




(b) Friendly Numbers Application




(c) K-Means Application

**Figure 5. Mean Miss Rate of L2 Cache**

## 5. Conclusion

The objective of this paper was to present a novel method for CPU comparison through simulation-driven experiments, employing the gem5 simulator to contrast two processors' memory hierarchies from the most prominent market companies, Intel and AMD. Specifically, the processors were the Intel Xeon w7-3455 and AMD EPYC 9224. As expounded upon in Section 4, AMD's hierarchy exhibits better performance when running with a reduced number of cores, whereas Intel's counterpart performs better when using all available cores. The utilization of the simulation-driven experiment offers numerous advantages, by employing it, we were able to circumvent the need for specific physical hardware, mitigating logistical challenges, empowering not only researchers but also companies to invest in hardware more safely. At the same time, simulation presents downsides such as overhead when compared to real hardware, and presents only approximated results.

In future work, we aim to go beyond CPU comparison. We would like to deeply inspect individual processors, examining their microarchitecture and operation, compare the results obtained in this paper with a real machine, measure how close to reality our results are, and analyze other metrics that would be able to explain better the results obtained and compare them to results obtained by similar papers. In addition, we intend to characterize the CAP Bench workloads for these simulation purposes.

## References


Anas Ahmad Abudaqa et al. Simulation of arm and x86 microprocessors using in-order and out-of-order cpu models with gem5 simulator. In *2018 5th Int. Conf. on Electrical and Electronic Engineering (ICEEE)*, pages 317–322, 2018.

Ayaz Akram et al. ×86 computer architecture simulators: A comparative study. In *2016 IEEE 34th Int. Conf. on Computer Design (ICCD)*, pages 638–645, 2016.

Ayaz Akram et al. A survey of computer architecture simulation techniques and tools. *IEEE Access*, 7:78120–78145, 2019a.

Ayaz Akram et al. Validation of the gem5 simulator for x86 architectures. In *2019 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, pages 53–58, 2019b.

Nathan Binkert et al. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, aug 2011. ISSN 0163-5964.

Trevor E. Carlson et al. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *SC '11: Proceedings of 2011 Int. Conf. for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2011.

Subodha Charles et al. Exploration of memory and cluster modes in directory-based many-core cmps. In *2018 Twelfth IEEE/ACM Int. Symposium on Networks-on-Chip (NOCS)*, pages 1–8, 2018.

Daniel Chavarría-Miranda et al. High-performance computing (hpc): Application use in the power grid. In *2012 IEEE Power and Energy Society General Meeting*, pages 1–7, 2012.

Fernando A. Endo et al. Micro-architectural simulation of in-order and out-of-order arm microprocessors with gem5. In *2014 Int. Conf. on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV)*, pages 266–273, 2014.

Matheus A. Souza et al. Cap bench: a benchmark suite for performance and energy evaluation of low-power many-core processors. *Concurrency and Computation: Practice and Experience*, 29(4):e3892, 2017a.

Matheus A. Souza et al. Energy consumption improvement of shared-cache multicore clusters based on explicit simultaneous multithreading. In *2017 Int. Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW)*, pages 1–6, 2017b.

Matheus A. Souza et al. Design space exploration of energy efficient noc-and cache-based many-core architecture. In *2018 30th Int. Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 402–409, 2018.

Rafael Ubal et al. Multi2sim: A simulation framework to evaluate multicore-multithreaded processors. In *19th Int. Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'07)*, pages 62–68, 2007.

Johnson Umeike et al. Profiling gem5 simulator. In *2023 IEEE Int. Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 103–113, 2023.

B. Vikas et al. On the cache behavior of splash-2 benchmarks on arm and alpha processors in gem5 full system simulator. In *2014 3rd Int. Conf. on Eco-friendly Computing and Communication Systems*, pages 5–8, 2014.

Matt T. Yourst. Ptlsim: A cycle accurate full system x86-64 microarchitectural simulator. In *2007 IEEE Int. Symposium on Performance Analysis of Systems Software*, pages 23–34, 2007.