AN EDUCATIONAL TOOL FOR TESTING HIERARCHICAL MULTILEVEL CACHES

J.A. Gómez Pulido, J.M. Sánchez Pérez, J.A. Moreno Zamora
Department of Computer Sciences
University of Extremadura
Av. Universidad, s/n. 10071 Cáceres. Spain
jangomez@ba.unex.es

ABSTRACT.

In this work we present a simulator for a multilevel cache memory system on a monoprocessor environment. It has incorporated a full graphic interface operating on a PC-DOS environment. At first, the simulator was conceived as a tool for applying it to teaching of cache memories. However, the potentiality of the developed system has proved its utility on program analysis and design strategies of memory systems. The above characteristics enable the simulator to be used for designing systems that run optimally a determinate kind of programs and improve the operating mode of a determinate architecture

KEYWORDS.

Multilevel caches. Performance evaluation. Trace-driven simulation. Education.

1. INTRODUCTION.

The performance of a computer system is a function of the speed of the individual functional unit, such as floating-points units, caches, bus, memory systems, I/O units, and of the workload presented to the system. It is well known that caches are a critical component of any performance computer system. Cache is the simplest cost effective way to achieve high speed memory and its performance is extremely vital for high speed computers [1]. In this paper we focus our attention on the developing of a simulator for multilevel cache memory systems on a monoprocessor environment.

There are well known cache memory simulators working on a monoprocessor environment [1]. Starting from those works, we have developed a new simulator, when has been taken into account new considerations about its multilevel capacity and analysis of characteristics, interface and portability. The design considerations have been oriented to satisfy a set of characteristic with didactic and research goals.

The didactic goal tries to approach to the student the problematic of cache memories, shown graphically all knowledge they may acquire in diverse texts [1],[3]. In order to do that has been necessary to built a friendly interface that allows to develop all analysis jobs. The research goal is necessary in order to use the simulator in jobs such analysis of program locality, behaviour of different architectures, developing of design strategies, etc.

The weight given to each goal will depend on the simulator user, the possibility to use other simulators, or the current and future simulator capabilities.

In the next section of this paper we mention some theoretical aspects of cache memories. Section 3 presents the different characteristics the simulator offers to the user. In section 4 are the results found and, finally, conclusions are presented in the last section.

2. THEORETICAL ASPECTS.

The theoretical considerations are well shown in many computer architecture books [1],[3], and we will not mention them here. But for understanding the simulator scope related with real organizations, we expose some theoretical aspects:

The memory hierarchy has a block organization. From lowest to highest level (main memory) we consider the block as the only information unit, that can be referenced on the transaction between levels. So, the block size in any level does not change. Moreover, the word width (minimum unit referenced by programs) does not change also, is the same in all memory hierarchies. Each line or partition (cache block) has associated the necessary bits for the management of different algorithms related with fetching, reading, writing, mapping operations, etc. These bits constitute the labels, validations, counts, etc.

At last, all operations and algorithms are similar to those found in any computer architecture book. So, the found results have a very close mapping with the real world.

3. ARCHITECTURE.

In this section the different characteristics that simulator offers to the user are exposed. They are grouped according to simulated memory architecture, processing programs, or the simulator program.

Hardware.

In relation to the hardware organization characteristics on the memory hierarchy, the simulator offers the following possibilities:

Number of cache levels in the memory hierarchy.	Up to four cache levels.
Cache type.	Unified (or mixed: 1 cache by level). Separated (instructions and data: 2 caches by level).
Mapping.	Direct. Set-Associative. Fully-associative.
Replacement policies.	Random, LFU (Least Frequently Used). LRU (Least Recently Used). FIFO (First-In, First-Out).
Writing strategies.	Direct. Write-Back.
CPU Word wide.	8, 16, 32 or 64 bits.
Words by block.	1, 2, 4, 8, 16, 32, 64, 128 or 256 words.
Blocks in main memory.	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096 or 8192 blocks.
Maximum block size.	2 KB.
Maximum main memory size.	16 MB.
Maximum number of cache partitions, sets or ways.	512.
Maximum cache size (excluded validity bits, labels, counts, etc).	1 MB.
References.	To memory words.

The different choices selected in the simulator for configuring a given architecture may be stored on a ASCII data file for a future load, so the need of making many selections for configuring the same memory model is avoided.

Programs.

For working with the simulator, it is necessary to use data files with the "calls" and memory addresses demanded by the CPU during the running of a program: the named memory **traces**. These data files (based on **din** and **PDATS** formats [4]) consists of lines, each one has two numbers, separated by only one white space:

label_value

- **label** is a decimal number that identifies the access memory operation type demanded by the CPU, in a given time, according to the instruction program: to capture an instruction (0), to read a memory data (2) or to write a data in memory (3).
- **value** is an hexadecimal number, that indicates the effective address of memory word to be accessed by the CPU. This address will be translated by the simulator for locating the word in the memory system block structure.

As an example, the part of file of the figure shows a memory trace with 6 instruction captures of a determinate program. Three instructions imply data reading, and one ask for writing in memory. By that, those 6 instructions accounting a total of 10 memory access.

```
0 1c07
0 1da4
2 7a50
0 1e03
0 1fb7
2 7a51
0 201b
2 7a70
0 211e
3 7a50
```

We dispose of a wide set of these data files for studying locality and better cache organizations for a certain type of programs. Many of these traces come from tests performed with real programs on different architectures on the Parallel Architecture Research Lab, New Mexico State University.

Simulator.

In this section we show the principal characteristics of the simulator. Next table resumes some of its principal descriptors.

Program	SISMEC v 1.0
Executable file size.	370 KB
Platform.	PC - DOS
Processor.	Intel 486 or Pentium.
Graphic environment.	Colour VGA.
Source code.	C++
Source code size.	200 KB
Mouse capability.	Yes
Menus.	About 20
Buttons.	About 70
Capability to load and save data files.	Yes

For working with the simulator, a friendly graphic interface is provided. With this interface, it is possible to perform memory designs according to our specifications, to select traces for checking programs, to view final and running time results (graphically or numerically), to save data files, to see schemes, etc. All these capabilities make easy a quick familiarization with the simulator analysis jobs.

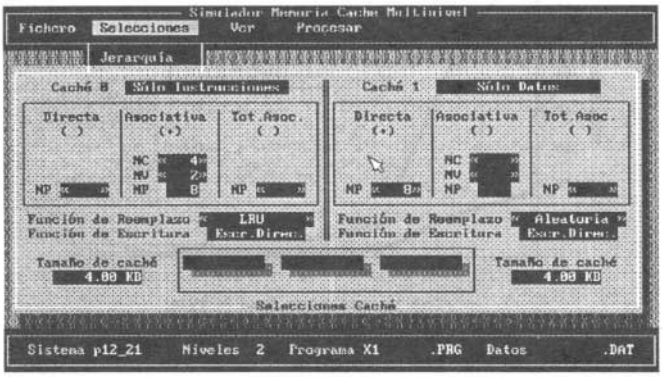
The simulation consists of the programmed reproduction of operations that would be really performed by the circuits of multilevel cache memory system. For that, the adequate computations are performed and, at the same time, the present and accumulated results, are shown. There are possibility of aborting the simulation in any time, for correcting any architectonic detail without waiting the simulation end.

During the simulation, it is also possible to save the essential data in order to may perform, afterwards, the same simulation process without any necessity to do again the same computations. So, it is achieved a "filmed" simulation whose reproduction is quicker because it avoids the realization of many arithmetic and logic operations indicated by the algorithms used for performing the simulation.

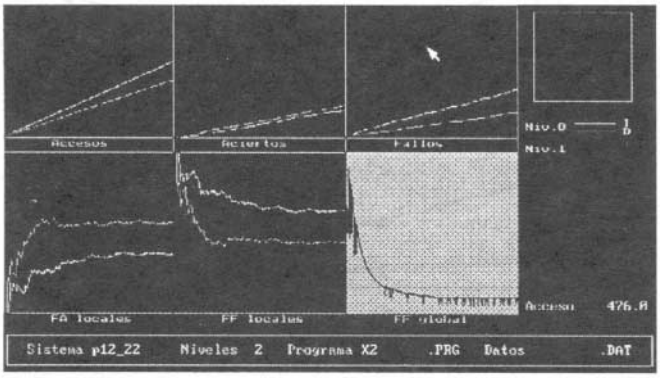
An abstract of the options that offers our simulation program are the following:

- **Memory System Design Parameters**. These parameters have a mapping with the architectural system characteristics, such they were described above (word width, block size, writing strategy, etc.). All parameters are related between them according to the theoretical models. The simulator avoids and tips the user made a choice whose value will be contradictory with the choice for other parameters.
- **Simulation Environments** (result viewing). This is the simulator core, because after performing many choices starts the simulation. The presentation mode of simulation results can be selected. During the simulation process it is possible to stop it, in order to do any previous selection in other menus (we do not have to wait until simulation is finished). Between the different environments can be found: numerical fields and graphics.

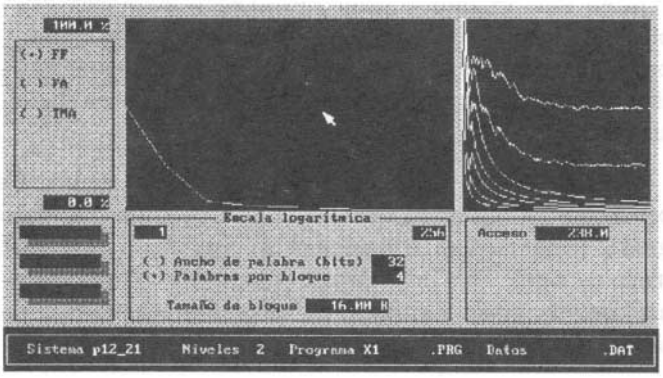
⇒ The numerical fields show instruction counters, accesses, misses, hits, rates, etc. All these values are computed and shown for each cache and for the global system. These fields reports a lot of detailed information, useful to elaborate different stats.



⇒ The curve graphics show the evolution of these parameters respect to the memory accesses. They are very useful for studying the program locality, the influence of cache size, etc. Moreover, the simulation repetitions may, on these graphics, showing information of, for example, the impact of random substitution strategy in caches with associative mapping (because in some curve sections will appear little desviations).



⇒ With comparative graphics it's possible to compare final values of key parameters (miss and hit rates) in relation with the block size. This allow to extract conclusions related with the cache design and according to the theoretical aspects, such as the existence of a pollution point, effect in the spatial locality of the block size, etc.



- **Other environments**. Between other environments, the simulator can show results we mention:
  - ⇒ Schematic visualization of system memory organization for seeing the different cache levels, CPU and main memory, with its more important characteristics.
  - ⇒ Loading of different files (trace and model selections) for performing simulations. This is made in a window in order to made easier a choice quick of simulations. The files are in the same working directory.
  - ⇒ Saving in a file one determinate memory organization. This allows us to build a database with different memory organizations, emulating MIPS, Sparc, PowerPC and other architectures.
  - ⇒ Saving or loading a file with the numerical results of a given simulation. As it was mentioned, this may be used for made simulations without generating operations, obtaining so more speedup for showing results.

When this simulator will be used in depth, it will be possible to obtain more analytic information, than we have shown now.

4. EXPERIMENTAL RESULTS FOR CACHES.

For validating the simulation results, we compare our experimental results with those found by other authors [1], [3], [5]. We show the found results for a set of determinate architectures with a set of concrete traces. So, we consider some traces based on the first thousands of memory accesses of Spice and some **SPEC'92** benchmarks, according real tests made on a MIPS R2000 system.

At first, we consider a simple architecture of word width of 16 bits, write through and blocks of 32 bytes. On **Fig 1** is shown the miss rate vs. the cache size (only a cache level), with an unified cache of direct mapping.

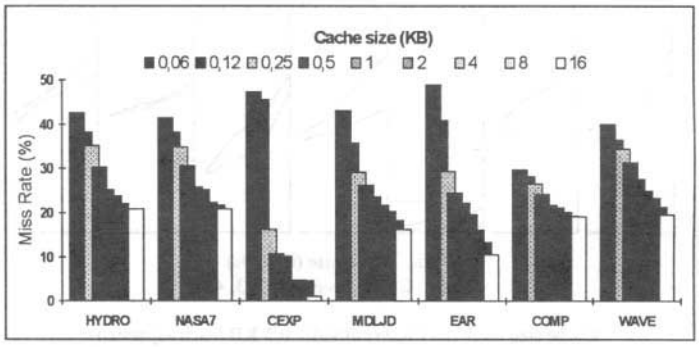


Fig. 1

On **Fig. 2** the same assumptions are considered, except now we have two cache memories (data + instructions) such as the add of its size is the size of the unified memory.

The miss rate decreases with the cache size, as we can see for all benchmarks, and this indicates that, independently of different locality grades, all programs have the same general behaviour. Also, it may be observed that for great sizes of cache, the miss rate is stabilised, this shows a type of miss:

compulsory. The differences of miss rate for a determinate increment of cache size indicates the memory addresses are as near than a little increased of cache size bring about a great increase of performance. Clearly, this point depends of the program. From **Fig. 2** we can conclude the best results are found with instructions and data caches, but increasing the global cache size this advantage, obviously, decreases.

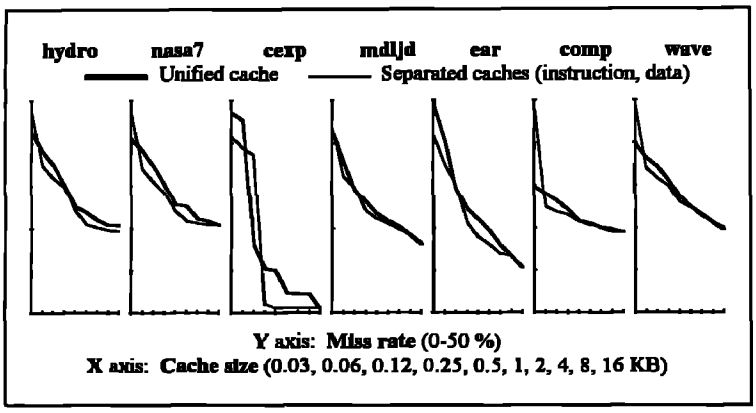


Fig. 2

For studying the effect of adding more cache levels, we have used other architecture (word width of 32 bits, write-through, blocks of 64 words). **Fig. 3** shows the miss rate, for direct mapping caches, depending on the considered level number. For each level, a cache size fixed is assumed, and there may be an unified or two separated caches. As it could be hoped, increasing the level number a better miss rate is achieved. So, the modern processors are designed, as minimum, with two cache levels. At this point, also can be observed it is better to use separated caches instead of unified.

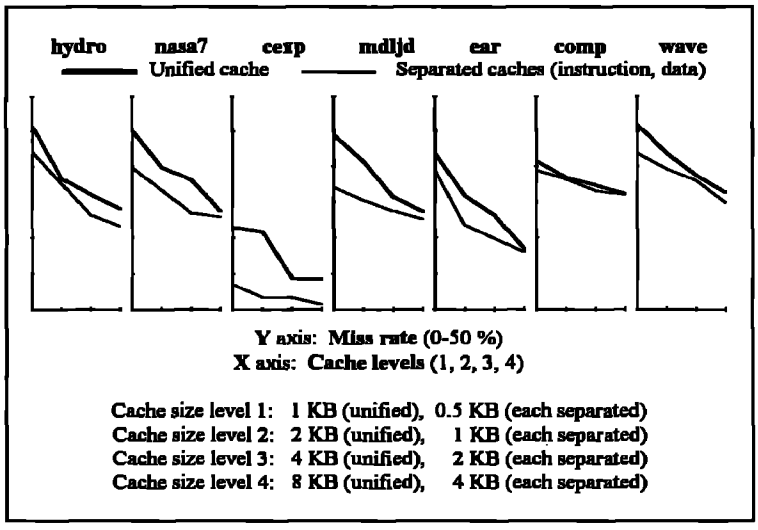


Fig. 3

Now we study the influence of the cache size as design parameter on a second level cache. For Spice program, we consider a first level cache of 1 Kbyte and direct mapping, and a second level cache, also of direct mapping. **Fig. 4** tell us that increasing the size of the second level cache, decreases the miss rate, but from a given time this decreasing is much less, because of the appearance of compulsory misses.

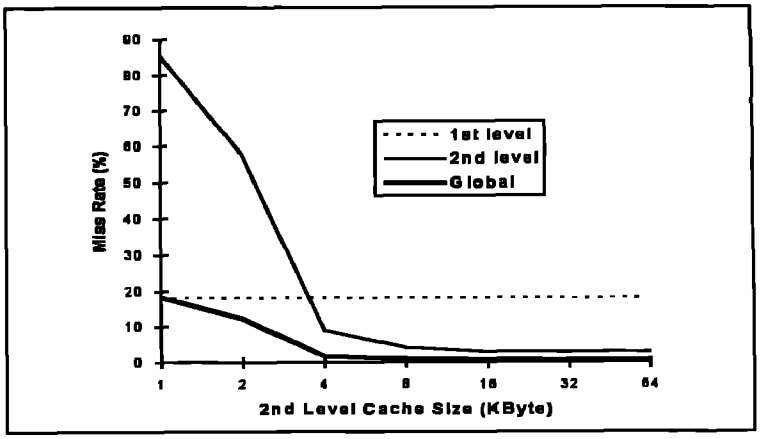


Fig. 4

Other aspect we have considered is the influence of the block size. Also for Spice program, using an unified and one level of direct mapping cache, we have gathered the results shown in **Fig. 5**. From these results can be observed the existence of a pollution point, which influence is smaller when increases the cache size.

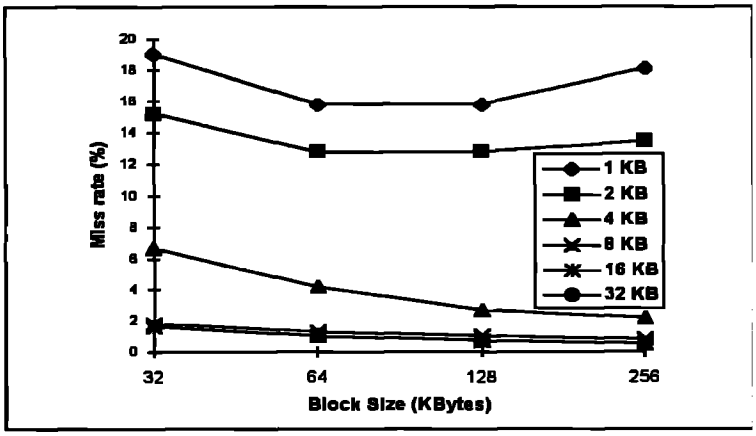


Fig. 5

These results are fully consistent with the theory we can find in computer architecture texts: Increasing the block size, to more word number in the block, less miss rate, because of spatial locality since increases the probability, in a near future, for accessing data nearest to the accessed word. However, from a given time (pollution point), to more word number in the block, more stride between some words, so referencing one, the probability for accessing the more strides decreases. It is clear, than more cache capacity, less negative effects when the block size increases.

Finally, in **Fig 6** are gathered some results using the ear benchmark. It represents the miss rate versus different mapping grades (with random replacement strategy for the associative mapping case). This indicates the miss rate improves when the associativity grade increases, although the benefits are every time less significants.

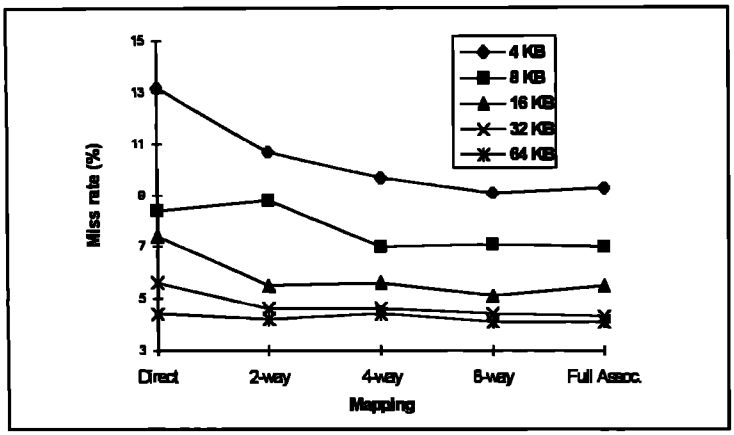


Fig. 6

5. CONCLUSIONS.

In this work we have shown the principal characteristics of a multilevel cache memory system simulator. It has been satisfactorily used in our academic environment [6]. Students have used it as tool for experimenting the different theoretical aspects about cache memories learned in the regular courses of computer structures. Professors have used it because it allows to analyse different situations with an easy and modest tool.

At the present, we are working for incorporating (adding) new capabilities, such as average access time, miss penalties, etc. Also, we are modifying the simulator in order to be applicable to multiprocessor systems, specially for solving questions related to cache coherence problem.

At last, in order to contribute to a better knowledge of the multilevel cache problematic, the simulator is available, with educational purposes, for universities and research centers, free of fee. People can contact with the authors.

6. REFERENCES.

- [1]. Hennesy, J.L. and Patterson, D.A. *Computer Architecture - A Quantitative Approach.* Morgan Kauffmann. 1996.
- [2]. Hill, M.D. and Smith, A.J. *"Evaluating Associativity in CPU caches".* IEEE Transactions on Computers, C-38, 12, December 1989, p. 1612-1630.
- [3]. Patterson, D.A. and Hennesy, J.L. *Computer Organization and Design. The Hardware/Software Interface.* Morgan Kauffmann. 1993.
- [4]. Johnson, E.E. and Ha, J. *"Lossless Address Trace Compression For Reducing File Size And Access Time".* Proceedings, 1994 IEEE International Phoenix Conference on Computers and Communications, p. 213-219.
- [5]. Obaidat, M.S., Khalid, H. and Sadiq, K. *"A Methodology for Evaluating the Performance of CISC Computer Systems under Simple and Two Cache Environments".* Microprocessor and Microprogramming Journal, July 1994, p. 411-421.
- [6]. Gómez Pulido, J.A., Sánchez Pérez, J.M and Moreno Zamora, J.A. *"Designing a Cache System for Teaching Purposes".* Proceedings, 1996 IEEE Mixed Design of Integrated Circuits and Systems, Lodz, Poland, 30 May-1 June 1996, p. 359-364.