Int. J. Engng Ed. Vol. 26, No. 6, pp. 1365–1377, 2010
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Increasing Student's Participation in a Memory Hierarchy Course: Design, Use and Analysis of the MNEME Simulator*

L. MORENO, E. J. GONZÁLEZ, B. POPESCU, J. TORRES, J. TOLEDO
Dpto. de Ingeniería de Sistemas y Automática y ATC, Universidad de La Laguna.
Av. Astrofísico Fco. Sánchez, S/N. CP 38206. La Laguna, Spain. E-mail: ejgonzal@ull.es

Using simulators in engineering disciplines is widespread. Memory Hierarchy is no exception. However, the process of its design and analysis as an educational resource has not been sufficiently explored. On the one hand, student motivation in the course could improve if they were invited to take part in the design of the simulator. On the other hand, this participation implies a deeper knowledge in the topics of the course. In this paper, this process is described about a simulator called MNEME, which includes a complete vision of memory hierarchy topics. This simulator has been validated and improved using feedback from students during three academic years.

Keywords: student's motivation; memory hierarchy; simulation

1. INTRODUCTION

USE OF SIMULATORS is a tested valuable tool for the teaching of engineering disciplines, since students can test how different elements work with a detail that it is impossible to reach by other techniques (the elements to study are difficult to access, expensive, etc.). A clear paradigm of this situation is Computer Architecture, where the elements to study, that is, the different processors, memory units and so on, cannot be accessed in an isolated way. Nevertheless, although simulators are widely used in Computer Architecture courses, simulators by themselves are not enough to significantly improve the learning process. It is necessary to carry out other activities which allow the students to gain knowledge, comprehend theoretical concepts and apply those concepts to the real world [1].

This paper proposes a methodology based on the process of designing and analysing a simulator as an educational resource in a Computer Architecture course. On the one hand, student motivation is supposed to improve if students are invited to take part in the design of the simulator. On the other hand, this participation implies a deeper knowledge in the topics of the course, since these topics have to be included in the simulator features. Many studies and efforts have been carried out regarding the improvement in student motivation in Engineering [2–4] since it has been identified as a key factor in course success [5]. Motivation clearly influences the quality of learning; since it acts as an enabler for learning and academic success [6, 7]: motivation affects effort, effort affects results, positive results lead to an increase in ability [8].

In particular, this paper applies the aforementioned methodology to memory system organization and architecture, designing and implementing a simulator called MNEME (for the Greek muse of the memory) for a Computer Architecture course in a Computer Engineering degree. This subject has been identified as a core topic by the joint IEEE Computer Society and Association for Computing Machinery (ACM) Computer task force, when defining Computer Curricula recommendations [9]. Due to its importance, Memory Hierarchy is taught in several subjects and degrees, with different level of depth and related to different topics. It is important to remark that this approach can be applied to every discipline where simulators are a useful tool, since students do not write any code line, as it is described in the following sections. Nevertheless, the authors consider that this approach is better applied to Engineering students since they have a clearer perception of what the programming process implies; in other words, Engineering students are able to evaluate properly the workload for including a modification in the simulator and the possibility of carrying it out.

The paper focuses on both the proposed approach and the MNEME simulator. On the one hand, the approach is considered to be useful when it is applied with other simulators or other areas. On the other, the MNEME simulator itself is a valuable tool for Memory Hierarchy teaching, being included in other course activities (adaptable to other simulators). This is the reason for describing MNEME in depth in the Presentation section. As can be shown, the need for a simulator is born

* Accepted 8 May 2010

from the application of the methodology itself. Therefore, the proposed methodology can be applied to other fields with their own simulation needs.

2. PRESENTATION

2.1 Context of the Computer Architecture course

A first point to state is the academic background of the user, since this knowledge will help in the design of the course.

A last-year computer science student has previous experience in computer structure and operative system. Regarding to computer structure, topics from digital electronics, including binary code, to computer functional units are explained. The principal functional units explained in the subject are: memory, interrupt system, control unit, arithmetic logic unit, etc. The main functionalities of these units are presented. The subject does not include a deep functionality description of the units, only a simplified description without upgrades or algorithms details. Cache levels, memory description, etc. are not included in the topic.

With respect to operative system, the hardware to high level software connection is presented. The topic presents the software interface to access the hardware, so any computer student (software and hardware specialist) needs it to understand the computer structure. In the topic it's not necessary to know deeply the hardware function, only present main units. In contrast, multitask, virtual memory, segmentation, pagination, are presented. The operative system is presented as a uniform layer to the programmers.

Other necessary aspect consists of expliciting the topics to be taught in the course. Table 1, adapted from [1], gives a detailed description of these topics and the order in which these contents are presented in the Computer Architecture course: fundamentals of computer design, memory hierarchy design, Internal structure of the processor and Interconnection networks for multiprocessors/multicomputers. Concerning memory hierarchy design, this course conforms to the above Computer Curricula recommendations.

2.2 Previous steps and first results

Before a major modification in the learning-teaching process methodology, it is recommended that its implication in the course be analysed. For this reason, the authors have developed an action-research-based process in order to improve the teaching of memory hierarchy process. This process can be divided into four phases (as shown in Table 2) that will be described in the remainder of this section.

Action research process [10–11] is a well-known reflective process that allows for inquiry and discussion as components of the research. That usually implies a collaborative activity among colleagues searching for solutions to everyday, real problems or looking for ways to improve instruction and increase student achievements. In a first step, the authors—teachers and researchers working at a University Department related to Computer Science Engineering subjects—pursue application of the action research process to their subjects. These subjects share many topics, so the action research approach can be applied. In this process, the authors have undertaken several meetings in which participants have examined their own

Table 1. Topics in Computer Architecture course

Topics	Contents	Course Sequence
Fundamentals of computerdesign	Measuring and reporting performance	October (8 hours)
Memory hierarchy design	**Virtual memory and cache memory**	**October–November (30 hours)**
Internal structure of the processor	Instruction set, pipelined processors, vector processors, Instruction Level Parallelism, Thread-Level Parallelism	November–May (100 hours)
Interconnection networks for multiprocessors/ multicomputers	Buses, direct and indirect networks, cache coherence	May–June (12 hours)

Table 2. Phases of action research results

Phase	Educational resources	Description
1	Concept Maps	Global concept map, Refinement of the concept map
2	MNEME	Students take part in the design /debugging of the tool
3	MNEME	Procedures using MNEME
4	Wiki, Moodle, etc.	Other complementary activities

educational practice systematically and carefully, using the techniques of research, looking to improve the teaching and learning process by reinforcing, modifying or changing perceptions based on informal data and non-systematic observations.

One of the topics that the authors have analysed in this action research process is that of the memory hierarchy, since this topic continuously appears in subjects such as Operative Systems, Computer Architecture and Computer Structure. The characteristic steps of action research are followed [10–12]:

1. Identification of problem area. This is usual to summarize a problem in a higher-order concise and meaningful question. In our case, the selected question has been: 'Which resources can we design and implement in order to get better results in memory hierarchy teaching?'
2. Collection and organization of data. Several sources of data can be used: interviews, journals, tests, etc. In the case presented in this work, the authors have based their work on samples of student work, projects and logs of meetings. The authors are implied in several subjects, with different student profiles, thus the collected data offer diversity and richness.
3. Interpretation of data. Due to the nature of the collected data, these data are mainly qualitative in the work presented in this paper. Thus, their interpretation should be carried out carefully. In this case, it has been deduced that students in the mentioned subjects miss any kind of computer-based resource that could help them in their learning process. As mentioned above, the analysed students are from a Computer Science degree course, so software is considered an important tool for the learning process.
4. Action based on data. From the interpretation sketched in step 3, the review of papers appeared in educational journals and collaboration with many experts in education, the authors have designed a plan based on the design and implementation of computer-based tools. This plan, detailed below, will follow the seven principles identified by Chickering and Gamson [13] for good practice in undergraduate education (encouraging contact between students and faculty, reciprocity and cooperation among students, active learning, prompt feedback, emphasizing time on task, communication of high expectations and respecting diverse talents and ways of learning).
5. Reflection. The logical final step in this process is evaluating the results and considering if the process needs to be revisited. Action research is usually an iterative process. Some of the reasons for the need of iteration can be found in the literature: sensitising the researcher to the many variables at play in the hectic, chaotic and multi-faceted classroom environments typically studied [14], enabling real-time and retrospective data analysis to feedback into the study as it progresses, taking the role of systematic variation in traditional experimental designs [15] providing opportunities for researchers to notice and capitalise on interesting and unanticipated events during a study [16], enabling researchers to generate and select design decisions regarding methods, theories, innovations and interventions in response to empirical findings and enabling critical reflection on the part of practitioners during the fast-paced fieldwork phase of a study [17].

As can be deduced from this description, teachers are interested in going a step beyond this process, adopting some aspects from design-based research (DBR) [17,18]: looking for reducing the 'credibility gap' of the students when learning theory, collaborating with experts in education and researchers, taking the initiative in the research process as both researchers and designers. In this phase an approach for the improvement in student motivation is critical. A variety of activity types is strongly recommended for this purpose. Thanasoulas [19] states the need of a motivational repertoire of strategies, including those of increasing the learners' self confidence and creating learner autonomy, and Crookes [20] states that it is important to provide variety and avoiding too-regular patterns of classroom routines at the same time that the learner should 'perceive that important personal needs are being met by the learning situation'. The authors consider that participation of students in the feedback and refinement phases helps in this variety of activity types.

A first result of the action-research process consists of the use of concept maps [21, 22] related to the desired learning field, in this case memory hierarchy. It has been demonstrated that concept maps enhance comprehension and the retention of ideas, helping memory at the same time that they offer the possibility to personalize learning, share knowledge and reinforce learning to acquire skills. Nevertheless, the authors have used this tool after another approach different from the traditional one of asking the students to create their own concept map. In particular, and after several meetings (where students from the implied subjects have shown a great activity) a concept map involving concepts about the memory hierarchy (as seen in every implied subject) was developed. On the one hand, this concept map would allow teachers to determine clearly the boundaries of their subjects, avoiding possible overlapping in some concepts—that is, the same concept should be taught only once to the same student, unless it would be needed to acquire a deeper knowledge about it. Thus, the learning process is improved through coordination among teachers. On the other hand, the students are asked to refine the concept map, including new concepts, and completing them, filling each concept with the theoretical material they have been presented with in a few lessons. In addition to

these lessons, students are encouraged to read about memory hierarchy from [23]. With this activity, students receive a clear direction about their studies, one of the factors that is usually identified as key for the improvement of the student's motivation. Doing this, students may respond in a more positive way [3, 24].

For the mentioned task—carried out in a collaborative way—the students have used an Open Source tool called Compendium. With this tool, the entire concept map can be exported to HTML format in such a way that students can access theoretical content just by clicking on the corresponding concept. Thus, working on the designed concept map helps students to study the subject, organize their notes and prepare educational material that can be used by other students in the future. Moreover, the generated HTML code can be included for its access in an e-learning platform, like Moodle [25].

Using the designed concept map makes it possible to explore strategies in order to improve the teaching/learning process. In particular, the authors have observed three statements:

1. There are many concepts defining themselves many parameters to establish in a practical way: global/local memory, execution/wait queues, eviction algorithms, bus size between the different cache levels, cache memory size, allocation page size, page size in virtual memory, line size in cache memory, write allocate/non write allocate, TLB size, mapping functions, etc.
2. Memory hierarchy teaching process should include every characteristic existing in actual machines: cache levels, page table implementations, eviction algorithms, etc.
3. It would be desirable, even more in a nearly DBR approach, to make experiments on the hardware. Nevertheless, due to technical limitations, experiments are often executed on a relatively old processor, with a simple cache organization, damaging the learning process extensibility. In addition to this, concerning the other alternative, existing simulators, especially trace-driven oriented ones, are often too simple and show how caches work in an isolated way. In contrast, execution-driven simulators are usually too complicated and without a friendly interface, becoming inappropriate for undergraduate students [26]. A survey of simulators can be found in [27, 28].

2.3 Simulators as a result of proposed methodology

A second major iteration of the action-research process consists of the design and implementation of a simulator covering the statements above. In an initial effort, a simulator called SIJEM was developed. It was designed for a basic course of Computer Architecture, thinking in a monoprocessor system, based on address trace and including virtual memory and three cache levels. Looking for simplicity, some important simplifications have been made:

- An only one-level page table.
- Lack of TLB.
- Three cache levels have the same block size.

Nevertheless, using SIJEM was soon shown insufficient for a strict action in the proposed methodology. As a new phase of the proposed methodology, students were involved in the design and implementation of a more powerful simulator, called MNEME (Fig. 1), which reflects those three premises shown at the end of the previous subsection and takes advantage from the flexibility of the software.

The debate about simple or complex simulators is out of the scope of this paper. Nevertheless, the authors will state that a simulator should not carry out an excessive simplification in order to make it simple. In other words, developers should not oversimplify their design because doing this, they could lead to new misconceptions. In contrast, if the real context is complex itself (that is, with many parameters and many concepts to be monitored for a pedagogical purpose), the simulator needs to be complex. In this last case, implementers must include some kind of mechanism in order to make the simulator useful from a pedagogical point of view. Thus, MNEME will result in a complex simulator, and this is its main strength: its complexity will cover a wide and complete range of architectures and examples.

Although the simulator is a valuable tool in and of itself (for example, when it is used in laboratory procedures), the proposed action-research implementation has taken pedagogical aspects from the design phase. For this phase, researchers have assumed the roles of teacher and observer. Students (from different degrees and subjects) were asked to take part actively in this process, looking for information in technical papers and journals like *IJEE*, testing successive versions of the simulator, documenting its use through tutorials and help files, detecting possible bugs, proposing new features/machines to be included, etc. These iterative activities will allow students to get a deeper knowledge of the subject in a social constructivist way, since they are encouraged to share their impressions and work together. In this way, several principles identified by Chickering and Gamson [8] are reached. It is important to remark that students have not written any code for the simulator, since it is not the goal of the exercise. For these activities, Moodle platform has been used as a collaboration framework by students and teachers. These activities affect the motivation of the student through team work, reward and recognition and social pressure and competition factors, since there is an external performance evaluation with corresponding rewards.

This refinement process has been carried out for two years, obtaining a complete version of

MNEME. Nevertheless, the authors have observed the benefits of this student participation and decided to repeat it as a learning procedure. For this purpose, the students will work in this phase with incomplete/old versions of the simulator and/or including some novelties, such as a new commercial machine configuration.

Due to the complexity of the simulator (high number of parameters, different devices involved in memory hierarchy, use of complex concepts), the students themselves have developed three different tutorials, helping their classmates to take advantage from the features offered by MNEME. For these tutorials, the students celebrated several meetings with the course teachers in order to detail the objectives of these tutorials.

- The first tutorial studies exclusively the virtual memory concept, avoiding the use of the three possible cache levels.
- The second tutorial implements the Intel Centrino processor with two cache levels L1 and L2, inhibiting the page aging technique, and a page table with a 2-level top-down search, a TLB and several replacement algorithms for cache and TLB.
- The last tutorial analyses an implementation in MNEME of multiprocessing and page aging, based on two processes, and how these processes change their location from one context to the other when there are fails in the TLB and in the cache.

In the meetings, the possible extensions of MNEME were also considered, since MNEME offers some limitations in multiprocessor and/or multicore systems. A new version of the simulator is currently being developed. For that, the authors have adopted the AMD HyperTransport, which allows the point to point connection among the nodes of a multiprocessor system. This protocol extends the possibilities of shared memory systems to systems that were previously developed using only message passing. The system includes a coherence system (to be chosen from a set of them), and allows the inclusion of TLB memories, cache levels for each node CPU, sharing as it is desired the memory structure with other node CPUs. The simulator analysed the performance of the multiprocessor system, and allows to compare it with the performance of other memory sharing system.

In subsequent subsections, the complete version of the MNEME is described, together with the third phase of the learning process: designing procedures for using MNEME itself as a learning resource. It is important to note that the methodology is independent from the particular field and from the proposed type of simulator.

2.4 Description of MNEME

MNEME [29–30] is a multiplatform simulation tool developed in Java Swing as an Applet, so it can be run in any Internet browser window. This fact does not mean that the simulator cannot be run locally, since MNEME can be downloaded free of charge and the users can access its features by opening a HTML file, called *sj.html*. A screenshot of the tool is shown in Fig. 1.

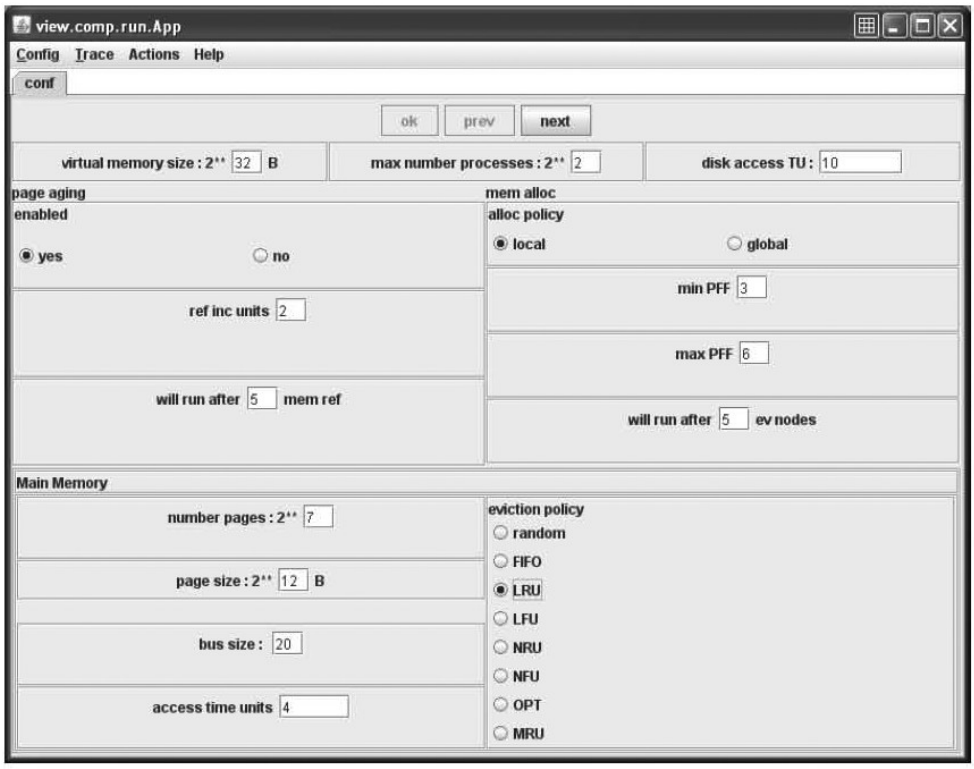

Fig. 1. MNEME main window.

Due to the complexity of the field to model, MNEME results in a complex but complete simulator since its complexity will cover a wide and complete range of architectures and examples. The main features of MNEME can be summarized as follows:

- Multithread
- Hiperpaging
- Direct and reverse mapping
- TLB with levels
- Age paging control
- Data loading from server
- Eviction policy definition
- Sizeable buses
- Control of the simulation speed.

Interaction with the simulator can be defined by the following sequence:

1. Loading a configuration file. The configuration file is a XML file determining the configuration structure of the simulator. In this way, MNEME allows to visualize this configuration in an easy and interactive way. As example, the following excerpt:

```
<config>
. . . .
<mainMemoryConfig>
<numberEntriesNBits>2</number
EntriesNBits>
<blockSizeNBits>12</blockSizeNBits>
<evictionPolicy>FIFO</eviction
Policy>
<busSize>20</busSize>
<accessTimeUnits>4</accessTimeUnits>
<numberSetsNBits>0</numberSetsNBits>
<dataInstrSeparated>false</
dataInstrSeparated>
</mainMemoryConfig>
</config>
```

defines diverse parameters concerning to the configuration of the main memory.

The names of the tags are pretended to be auto-explained, thus students can define their own configuration files. The complete information about the tags and their meaning is added in the simulator help.

2. Confirmation/changes of MNEME configuration. Configuration files are supposed to be an easy way to define new memory hierarchy configurations; users can modify the parameters they have just loaded from the simulator GUI (as shown in Fig. 1) without editing any XML file. Fig. 2 and 3 detail the configuration parameters.
3. Loading a trace file**.** This file contains a set of read/write/fetch instructions. An excerpt of an example is the following:

```
003d49b0 MEMREAD
116f49a0 MEMWRITE
212ba3c0 MEMREAD
```

Each line represents the address to be accessed

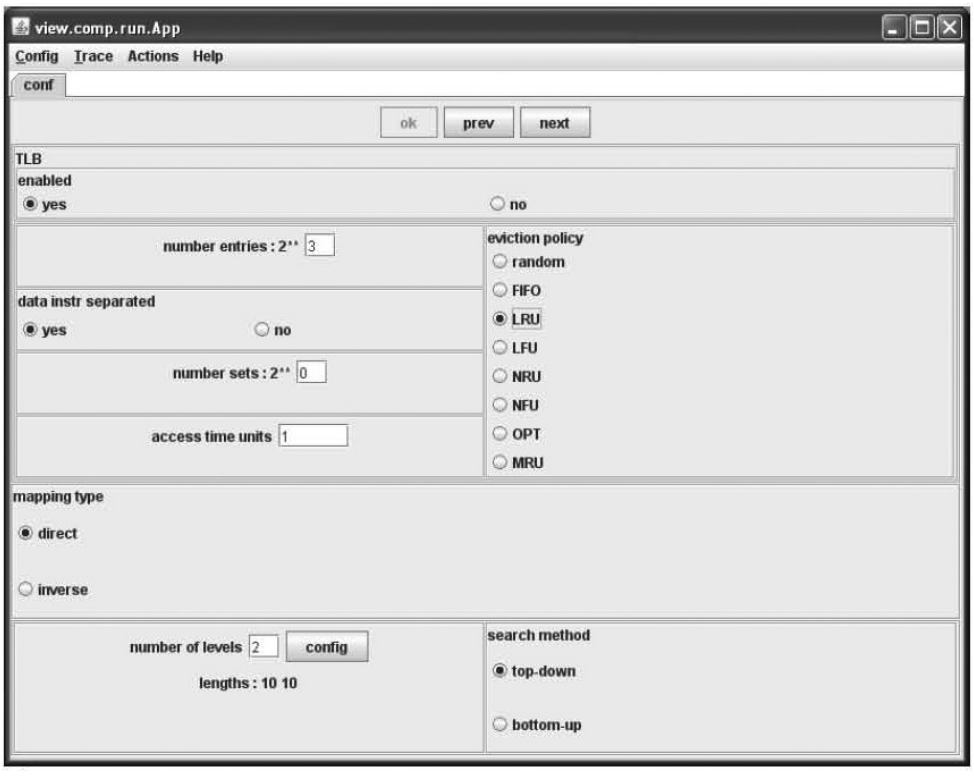

Fig. 2. MNEME configuration features.

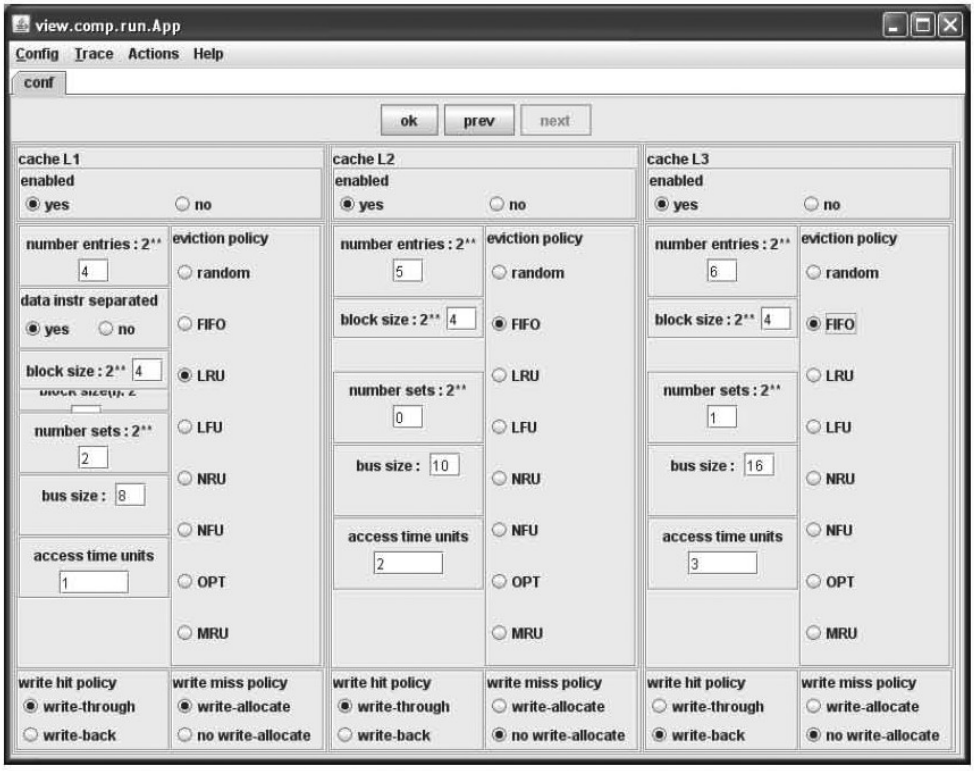

Fig. 3. MNEME configuration details (cache levels).

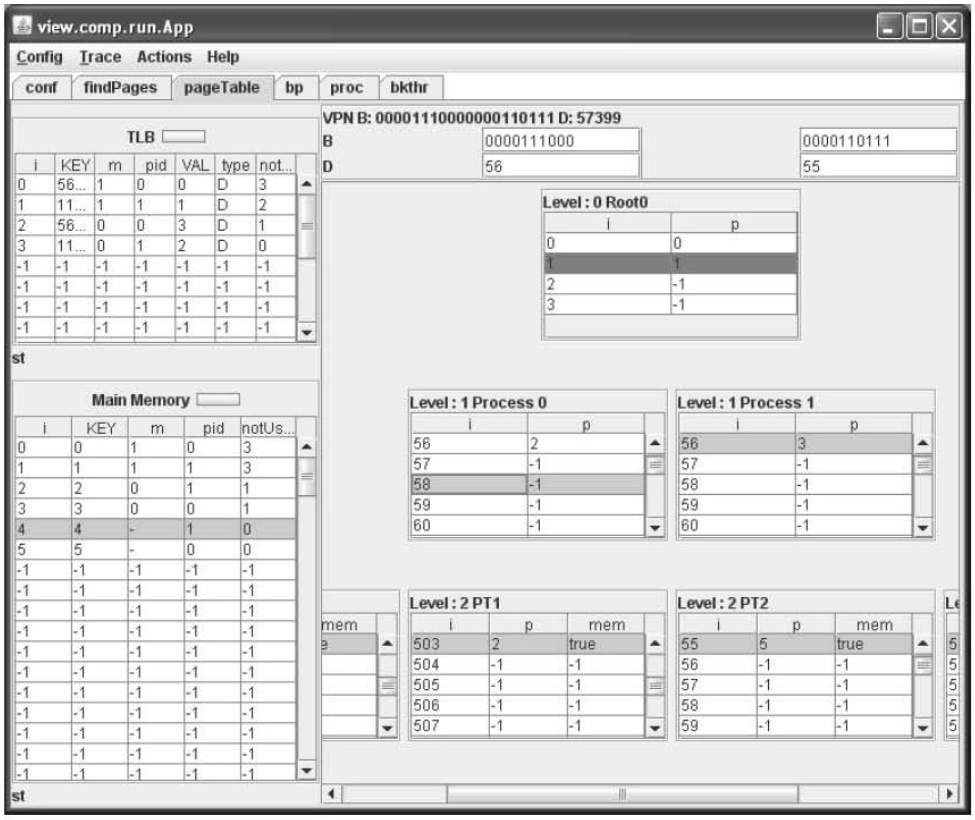

Fig. 4. MNEME pageTable tab.

and the type of access. As can be seen, the address to be accessed is expressed in hexadecimal format. This address, composed of eight digits, is not shown directly in the simulator. The address is divided into three fragments: the first five digits (20 bits), the next two digits (8 bits) and the last digit (4 bits).

4. Simulation. Once the two files have been loaded, simulation starts. MNEME offers a set of tabs where a complete list of parameters can be visualized by the students. Before describing each tab, it is important to remark that MNEME provides a colour code that allows the user to detect in an easy and visual way every effect on the code trace. For each event, the colour of implied code lines will change.

   Before an object is removed (for example, when a page is replaced in the main memory and so does not need to be referenced in the TLB) or replaced, there is a delay (about 2 seconds) that allows the user to follow the changes properly.
5. findPages tab. In the findPages tab, MNEME offers in a visual way, the actual status of the diverse memories of the configured machine. Figure 2 shows three cache levels and the main memory. Data are represented by tables, where each column is an attribute (number of inputs *i*, KEY, *pid* of the referred process, etc.) and each file represent a memory address.
6. PageTable Tab. In this tab, MNEME shows the different memories that take part in the table pages. Fig. 4 shows the main memory TLB and the indirect mapping table. In the case of a direct mapping, the simulator shows the tree of information tables for the mapping and the information concerning the virtual address. An example can be seen in Fig. 4.
7. Bp tab. This tab presents an image of the structure of memories and the actions to be carried out in each step as seen in Fig. 5.
8. Proc tab. In MNEME the number of threads to be loaded will be between 1 and 28. A thread can be in two queues: Execution (E) and Wait (W). These queues can be visualized once a trace file has been loaded into the simulator. The user can visualize parameters such as process identifier, units of time of the process, number of the following instruction to be executed by each process, time left in each queue and the position of each process in the queue. (Fig. 6).

Apart from the described tabs, practical experience is reinforced with contextual help easily accessible from any part of the simulator (it is important to note that some help files have been developed by the students themselves). This help includes a user guide for the simulator and a quick review of the theoretical concepts that the students may need.

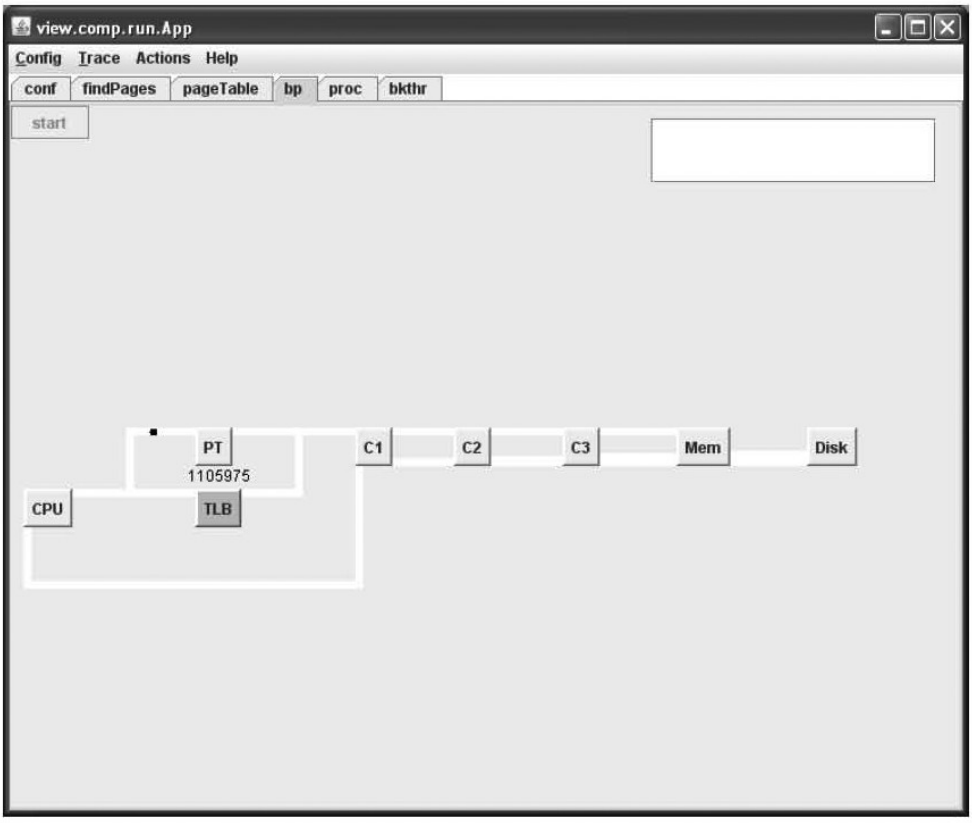

Fig. 5. MNEME bp tab.

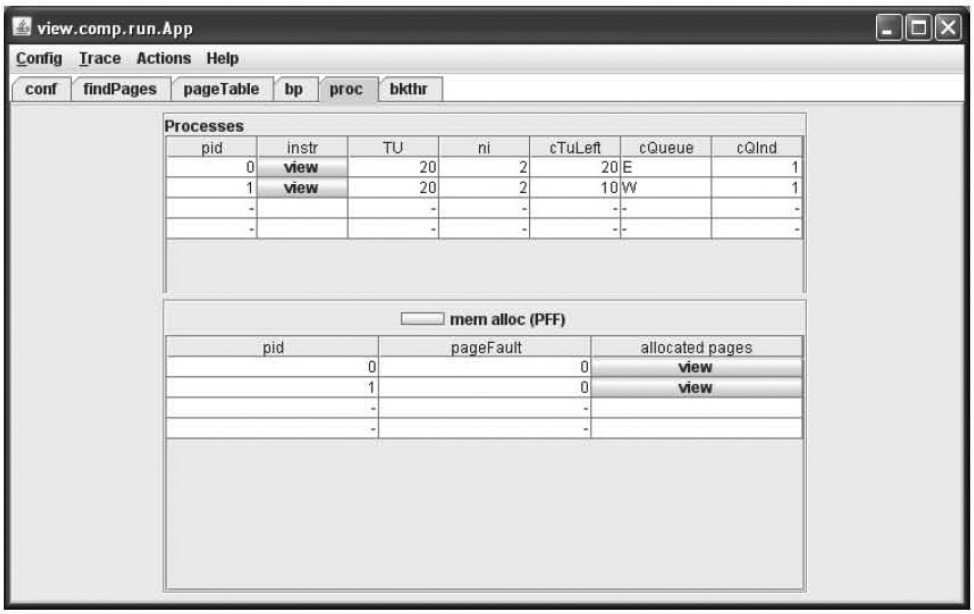

Fig. 6. MNEME proc tab.

2.5 Procedures for using MNEME as a learning resource

The MNEME simulator has been designed to be applied in several subjects. It is clear that different subjects imply different levels of depth and different focus. As an example, the authors will describe a typical sequence of introductory procedures based on the MNEME simulator. They are focused on different paging algorithms to be used in an Operative System. As well as these procedures, students have simultaneously developed activities regarding design, refinement and analysis of the simulator.

2.5.1 Introduction to the MNEME simulator

The first goal is that the student learns how to use the simulator. For this purpose, the students are presented with a very simple case of pagination: a hierarchical pagination system with two levels, and only one process accessing the memory. The students are very familiar with this case, since they have studied it in theoretical classes. Students are expected to pay more attention to MNEME features rather than the particular application. Moreover, this case is easy to follow and easy to predict. The code of the application consists of a sequence of accesses to memory, including both cache misses and hits.

By now, students have learnt about the MNEME menus and options and to interpret its tables and graphics. Moreover, the students are able to follow a trace of memory accesses from the process code to the simulator GUI.

2.5.2 Paging in a multiprogramming system

In the second practical procedure, MNEME is applied to a multiprogramming paging system. Students define the number of processes accessing memory concurrently, the quantum time of each process and the structure of the page table. MNEME manages a process scheduler which controls the process execution and will carry out the CPU if the process makes a page fault or if the quantum time finishes.

2.5.3 Memory management system.

In the third procedure, students learn about the MNEME memory and process management characteristics, so different table page structures will be tested. A basic program will be studied applying different techniques, in order to test the efficiency of the algorithms. The code example has a high number of page faults, with a small size memory, so the Operative System will send these pages to disk frequently. Local and global strategy will be tested in order to show the simulator characteristics.

2.5.4 Advanced paging management.

The last exercise is based on advanced paging methods such as reverse mapped page table, page aging and the lineal virtual memory page table. The simulator makes these advanced techniques easy for the student. In this exercise, the TLB will be tested too, testing the efficiency of the system in relation to the TLB size. All the MNEME characteristics will be shown, so students will see the data go out to the memory, inside the CPU Cache. The system is shown as a global system, from the hardware to the operative system.

This exercise is focused on the Operative system paging management, so MNEME will be configured using only these characteristics. Other features such as multiple cache levels will be used in other subjects, e.g. Computer Architecture. A simulator like MNEME which is able to simulate a

wide range of systems, is very useful in a Computer Science degree. Students will know the simulator and will test the computer structure with different points of view: hardware, operative system and computer architecture.

2.6 Other activities in the course

From their experience in other analogue subjects (as detailed in [2]) and as a fourth phase of the action-research process, the authors extrapolate some reinforcement experiences to the memory hierarchy subject.

1. Presentation on state of the art of commercial machines.
   Each group of 2–3 students is assigned a commercial machine in order to study it from a memory hierarchy context and design a 30–40 minutes Powerpoint-like presentation to be shown in class. After the presentation, they carry out a discussion with other groups that have been assigned similar architectures in order to compare their results. Thus, they make contact with powerful ideas as objects to extrapolation and appropriation.
2. Wiki of memory hierarchy concepts about state of the art machines.
   Each group produces a collaborative document about concepts and characteristics of the analysed machines. This helps them to consolidate their learning by means of the transference and synthesis of the studied concepts. Moreover, students of successive years will make use of these documents by correcting their weaknesses and profiting from their valuable contents.
3. Wiki analysis.
   The teachers point out similarities and differences among the analysed machines. Additionally they mediate in several discussions about these topics. In this way analysis, thoughts and learning about possible mistakes or misconceptions are achieved. At the same time, this procedure involves more social pressure from classmates, increasing student motivation.
4. Final evaluation.
   This mixed evaluation method includes observation, automatic registration, interview, individual test, teacher notebook, etc. The following methods are used in this task:
   - Quantitative Methods (individual tests, automatic registers in Moodle);
   - Qualitative Methods (observations, interviews);
   - Social Networks (observation of face to face relations, interactions in Moodle).

As can be observed from this methodological approach, the role of MNEME as learning mediator is essential in order to improve an educational experience that covers all memory hierarchy processor concepts.

2.7. Feedback from students

Feedback from students is a crucial piece of information in order to verify the usefulness of the proposed exercises [31]. Some weaknesses and potential improvements of the simulator and the methodology can also be identified in this way. As stated above, a first approach to this feedback was carried out in the design of the simulator, since a group of students took part significantly in this design, proposing some improvements that were included in the final version of MNEME. These improvements were especially focused on the usability of the simulator, due to the inherent complexity of the system to model.

Apart from this first feedback phase, 50 students from Fifth Course in Computer Science degree (with experience in software evaluation, simulators use and programming procedures) were asked to carry out a collaborative-type validation and to fill a questionnaire about MNEME and how the simulator helped them in the subject. This questionnaire basically focused on three main aspects:

1. The suitability of the simulator for the educational requirements of the students, including motivation.
2. The features and functionality of the software.
3. The technical aspects of the simulator, such as malfunctions or bugs.

As a result of the feedback process, some statements can be reported (as seen in Table 3):

- 100% of the students have tested MNEME for 3–6 hours before answering the questionnaire.
- 90% of the students did not know if there was any similar software for memory hierarchy simulation.
- 90 % of the students agree that MNEME helps to understand the subject topics better.
- 70 % of the students believe that MNEME keeps their interest.
- Nevertheless, students also confirm that it is necessary to make some improvements in the MNEME environment, since every interviewed student pointed out that the architecture in MNEME is not easy to follow.
- 90% of the students point out that the help files are clear and concise; students themselves took part in the redaction of MNEME help files. A similar percentage agrees that MNEME is highly flexible in order to design new procedures.
- Concerning technical aspects of the simulator, such as malfunctions or bugs, all of the students agree that MNEME works fine and it can be perfectly run in different operative systems.

3. DISCUSSION

Table 4 contains the evolution in the marks of the students of the subject when the different resources proposed in this paper are applied. As can be seen from these data, there is a clear tendency to get better marks with the application

Table 3. Data from questionnaires about MNEME

	< 2 hours	**2–4 hours**	**4–6 hours**	**> 6 hours**
How much time have I used MNEME before answering this questionnaire?	0%	56 %	44 %	0%
	Agree strongly	**Agree**	**Disagree**	**Disagree strongly**
I do not know about any similar software for memory hierarchy simulation	0%	10%	52%	38%
MNEME helps me to understand the topics of the course	20%	70%	10%	0%
I don't learn much using MNEME	0%	10%	90%	0%
MNEME keeps my interest on the course.	20%	50%	30%	0%
The designed architectures in MNEME is easy to understand	0%	0%	100%	0%
MNEME help files are clear and concise.	60%	30%	10%	0%
MNEME is highly flexible in order to design new procedures	48%	44%	8%	0%
MNEME works fine and it can be perfectly run in different operative systems	64%	36%	0%	0%

Table 4. Evolution of marks in the course

Mark	**Without simulators**	**With simulators (SIJEM)**	**Simulators (MNEME) + methodology**
E,F	27.59%	4.35%	0.00%
C,D	41.38%	65.22%	41.03%
A,B	31.03%	30.43%	58.97%

of the proposed methodology. Nevertheless, this tendency has to be confirmed with results from students in the next courses. With this caveat, from the authors' point of view, the described use of MNEME and the proposed motivational activities have helped students to better understand memory hierarchy theoretical concepts. The simulator is a valuable tool for teaching memory hierarchy that offers advantages that no other available simulators can provide. As memory hierarchy implies a huge set of concepts and parameters, MNEME results in a complex but rich environment, where non-trivial configurations can be proved. Thus, complexity in the simulator will improve the learning process.

However, the most remarkable results were related to student-teacher interaction. The students showed a higher level of motivation, which was reflected in a significant increase in class participation and general interest in the subject.

4. CONCLUSIONS

Some learning strategies based on action-research for teaching memory hierarchy have been presented in this work. The aim of the application of these strategies is to improve the learning process through a reflexive analysis of this process. From this analysis, carried out in a set of meetings among professors, the authors propose a methodology that can be divided into four phases:

1. use of global concept maps among the involved subjects,;
2. a memory hierarchy simulator (MNEME) in whose design students have taken part significantly;
3. code-based procedures using that simulator;
4. some complementary activities such as presentation on state of the art of commercial machines, wiki of memory hierarchy concepts, etc.

The described methodology has been tested by students of different degrees in the University of La Laguna, since memory hierarchy continuously appears in subjects such as Operative Systems, Computer Architecture and Computer Structure.

In addition to this, since the students have taken part in MNEME design, they have shown a greater motivation, difficult to find in these types of courses. Due to this motivation success, this methodology will also be applied to teaching other aspects of the course, such as multiprocessing, and to other courses.

The numerous tests which have been performed have confirmed the utility of MNEME as an educational tool to support teaching of memory

hierarchy, although more research is needed in order to confirm statistically the results.

In the next years, students will take advantage from this work, in a cycle of 3–4 years, since the advances in the field probably require a more sophisticated simulator and the number and quality of new contributions by students are supposed to decrease.

Currently, new features are being developed for the simulator, including those related to usability. These improvements include the design of an interactive tutorial about MNEME configuration and the implementation of a tool that will allow the professor to hide some features of the simulator depending on the knowledge of the student. In other words, MNEME will become a highly reconfigurable simulator, loading different configurations from a metadata file. Moreover, software modules about multicore processors and NUMA configuration are being added to the simulator.

REFERENCES

1. L. Moreno, C. S. González, I. Castilla, E.J. González and J. F. Sigut. Use of Constructivism and Collaborative Teaching in an ILP Processors Course. *IEEE Transactions on Education*, **2**(50), 2007, pp. 101–111.
2. K. M. Y. Law and K. B. Chuah, What motivates engineering students: a study in Taiwan, *International Journal of Engineering Education*, **25**(5), 2009, pp. 1068–1074.
3. K. M. Y. Law, V. C. S. Lee and Y. T. Yu. Learning motivation in e-learning facilitated computer programming courses. *Computers & Education*. **55**(1), 2010, pp. 218–228.
4. B. Reynolds, M. Mehalik, M. Lovell, and C.D. Schunn. Increasing student awareness of and interest in engineering as a career option through design-based learning. *International Journal of Engineering Education*, **25**(1), 2009, pp. 788–798.
5. M. Yaman, C. Nerdel and H. Bayrhuber. The effects of instructional support and learner interests when learning using computer simulations. *Computers & Education*, **51**(4), 2008, pp. 1784–1794.
6. E. A. Linnenbrink, P.R. Pintrich, Motivation as an enabler for academic success. *School Psychology. Review*, **31**(3), 2002, pp. 313–328.
7. D. J. Lynch, Motivational factors, learning strategies and resources management as predictors of course grades. *College Student Journal*. **40**(2), 2006, pp. 423–428
8. M. Rost, Generating Student Motivation, 2006, http://www.pearsonlongman.com/ae/worldview/motivation.pdf (Accessed: 26 April 2010).
9. ACM/IEEE-CS Joint Curriculum Task Force Rep. *Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering*, 2004.
10. C. Argyris, R. Putnam, D. Smith. *Action science: concepts, methods and skills for research and intervention*. San Francisco, Ca.: Jossey-Bass, 1985.
11. F. A. Heller. Group feedback analysis as a method of action research. In *AW Clark, Experimenting with organisational life*. Plenum, New York, 1976.
12. K. Lewin. Action research and minority problems. *Journal of Social Issues.* **2**(34-3), 1946.
13. A. Chickering and Z. Gamson. Applying the Seven Principles for Good Practice in Undergraduate Education. *New Directions for Teaching and Learning,* vol. 47, Jossey-Bass Inc., San Francisco, 1991.
14. A. L. Brown. Design experiments: theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences,* **2**(2), 1992, pp. 141–178.
15. P. Cobb, J. Confrey, A. diSessa, R. Lehrer and L. Schauble, Design experiments in educational research. *Educational Researcher* (32), 2003, pp. 1–9.
16. A. A. diSessa, Metarepresentation: Native competence and targets for instruction. *Cognition and Instruction* **22**(3), 2005, pp. 293–331.
17. W. A. Sandoval and P. L. Bell. Design-based research methods for studying learning in context: introduction. *Educational Psychologist*, **39**(4), 2004, pp. 199–201
18. D. Joseph. The practice of design-based research: uncovering the interplay between design, research, and the real-world context. *Educational Psychologist*, **39**(4), 2004, pp. 235–242.
19. D. Thenasoulas. Motivation and motivating in the foreign language classroom. *The Internet TESL Journal*, **VIII**(11), 2002.
20. G. Crookes. *A practicum in TESOL: Professional development through teaching practice*. Cambridge University Press, Cambridge, 2003.
21. J. D. Novak. Clarify with concept maps: a tool for students and teachers alike. *The Science Teacher*. **58**(7), 1991, pp. 45–49.
22. J. D. Novak. How do we learn our lesson? : taking students through the process. *The Science Teacher*. **60**(3), 1993, pp. 50–55.
23. J. L. Hennesy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. 3rd ed.,: Morgan Kaufmann, San Mateo, CA, 2003.
24. D. Stipek. Motivation and instruction. In D. C. Berliner & R.C. Calfee (Eds.), *Handbook of educational psychology*.: Simon & Schuster/Macmillan. New York, 1996, pp. 85–113.
25. http://moodle.org/ (Accessed: 26 April 2010).
26. J. Sahuquillo, N. Tomás, S. Petit and A. Pont. Spim-Cache: A pedagogical tool for teaching cache memories through code-based exercises. *IEEE Transactions on Education*, **50**(3), 2007, pp. 244–250
27. W. Yurcik, G. S. Wolffe and M. A. Holiday, A survey of simulators used in computer organization/architecture courses. *Proceedings of Summer Computer Simulation Conference.* Orlando, FL, Jul. 2001, pp. 524–529.
28. R. Quislant, E. Herruzo, O. Plata, J. I. Benavides and E.L. Zapata. Teaching the cache memory system using a reconfigurable approach. *IEEE Transactions on Education*, **51**(3), 2008, pp. 336–341.

29. MNEME can be downloaded from http://www.isaatc.ull.es/portal/proyectos/mneme/descargas (Accessed: 26 April 2010).
30. L. Moreno, E. .J. González, B. Popescu, J. Torres, J. Toledo, C.S. González. Simuladores de Jerarquía de Memoria en el Contexto de un Proceso de Investigación-Acción. *Proceedings of XIII Jornadas de Enseñanza Universitaria de la Informática*. Teruel, Spain, July 2007.
31. L. Harvey. *Student feedback: a report to the Higher Education Funding Council for England.* Technical report. http://www.uce.ac.uk/crq/publications/studentfeedback.pdf (Accessed 26 April 2010).

Lorenzo Moreno received his M.S. and Ph.D. degrees from the Universidad Complutense de Madrid, Spain, in 1973 and 1977 respectively. From 1977 to 1979 he was an Associate Professor in the Department of Computer and Control Engineering, Universidad del País Vasco, Spain. From 1979 to 1988 he was an Associate Professor in the Department of Computer Science, University Autónoma de Barcelona, Spain. From 1989 he has been Professor in the University of La Laguna, Tenerife, Spain. His areas of interest include computer architecture and computer education.

Evelio Gonzalez received the M.S. degree in Applied Physics in 1998 and his Ph.D. degree in Computer Science in 2004 from the University of La Laguna, Tenerife, Spain. From 1998 to 2001, he was a Research Student in the Department of Applied Physics, Electronics and Systems at the same university. Currently, he works as Assistant Professor in the University de La Laguna. His areas of interest include Simulation, Digital Control, Computer Architecture, Artificial Intelligence and Intelligent Agents.

B. Popescu received his degree in Computer Science in 2004 from the University of Bucarest, Romania. From 2004 to 2005 she has been working as a programmer at GZK Software, Bucarest, Romania. From 2006 she has been on a scholarship in the University of La Laguna, Spain.

Jonay T. Toledo Carrillo received his degree in Computer Science Engineering in 2001 and his degree in Electronics Engineering in 2002 from the University of La Laguna, Tenerife, Spain. From 2002 to 2008 he has been a Researcher at the Department of Systems Engineering and Automation of the University of La Laguna, where he is currently an Associate Professor. He received his Ph. D. in 2008 from the University of La Laguna, Tenerife, Spain. His two main research fields are robotics and control systems.

J. Torres received his degree in Electronics Engineering in 2001 from the University of La Laguna, Tenerife Spain. From 2001 to 2008 he has been a Researcher at Department of Systems Engineering and Control and Computer Architecture in the University of La Laguna, where he is currently an Associate Professor. He worked in autonomous robot heading and position localization systems and in artificial hearing to develop sensorial substitution devices for deaf. But currently his main research field is the computer vision applied to pose and gesture recognition.