International Journal of Engineering Education Vol. 27, No. 2, pp. 238–247, 2011
Printed in Great Britain

0949-149X/91 $3.00+0.00

SiMR: A Simulator For Learning Computer Architecture*

FERMÍN SÁNCHEZ[1], DAVID MEGÍAS[2] and JOSEP PRIETO-BLÁZQUEZ[2]
[1] Universitat Politècnica de Catalunya, Arquitectura de Computadors, Edifici D6, Jordi Girona, 1-3, 08034 Barcelona, Spain. E-mail: fermin@ac.upc.edu
[2] Universitat Oberta de Catalunya, Estudis d'Informàtica, Multimèdia i Telecomunicació, Rambla del Poblenou, 156, 08018 Barcelona, Spain. E-mail: dmegias, jprieto@uoc.edu

This paper presents SiMR, a simulator of the Rudimentary Machine designed to be used in a first course of computer architecture of Software Engineering and Computer Engineering programmes. The Rudimentary Machine contains all the basic elements in a RISC computer, and SiMR allows editing, assembling and executing programmes for this processor. SiMR is used at the Universitat Oberta de Catalunya as one of the most important resources in the Virtual Computing Architecture and Organisation Laboratory, since students work at home with the simulator and reports containing their work are automatically generated to be evaluated by tutors. The results obtained from a survey show that most of the students consider SiMR a highly necessary even an indispensable resource to learn the basic concepts about computer architecture.

Keywords: Distance learning environments; virtual laboratories; rudimentary machine; simulation software for processor architecture

1. Introduction

The fundamental concepts of Computer Architecture and Organisation (CAO) are an essential component of Software Engineering and Computer Engineering degrees. Fundamental concepts are concerned with all aspects of the design and organisation of the central processing unit (CPU) and the integration of the CPU into the computer system itself. The computer architecture curriculum must achieve several objectives. It must provide an overview of computer architecture and allow students to learn the operation of a typical computing machine [1].

Different references, such as the computer curricula 2005 [2], a review from *ICECE05* [3] and Sheppard [4], point out that practical laboratory activities are an essential part of any computer curriculum, since they strengthen the concepts presented during lectures. For that reason, new virtual spaces are required in a university such that practical activities can be carried out. Such spaces are called Virtual Laboratories and should include different resources, from human or pedagogical to technological, to improve the learning process [5, 6].

The term Virtual Lab is defined in different ways in the literature. The work presented in [7] takes a simple vision of a Virtual Lab as a local computer hosting. Some authors include simulation capabilities [8, 9] whilst others consider Virtual Labs as an extension of a remote laboratory [10]. Other relevant aspects, such as pedagogical and academic factors in a Virtual Lab, are presented in [11, 12]. This paper presents a microprocessor simulator to learn the concepts about computer architecture as one of the most important technological resources in the Virtual Computing Architecture and Organisation Laboratories (VCAOLabs). VCAOLabs provide students with a full virtual learning environment to carry out practical activities of CAO.

A simulator is a tool which imitates experiments, states or processes. One of the main features of simulators is their interactive capability. Interactive simulations are gaining importance as a means to explore, comprehend and communicate complex ideas [13].

Interactive simulators can be implemented using a combination of computers, high-resolution graphics, simulation programming languages and an Internet connection. The purpose of the simulator in a VCAOLab is to allow the students to understand the different computer components (memory, registers, buses, ALU, etc.) which may be difficult to observe in a real situation.

In order to use a processor in an initial course on computer architecture, old commercial products like the Digital VAX [14], the Zilog Z80 [15] or Motorola series [16] should be disregarded because they include large instruction sets and complex addressing modes. Processors in the Intel 80x86 family [17] should also be disregarded because of their lack of orthogonality and the number of side concepts that must be introduced (even the old 8086 is too cumbersome).

Different proposals related to simulators of VCAOLab have been reviewed in the literature.

* Accepted 21 July 2010.

Existing pedagogic processors like the MIPS DLX [18] are typically well documented and allow the introduction of advanced concepts. However, they are also still far too complex to be easily assimilated by students with very rudimentary knowledge about computer architecture. For this processor, SPIM and XPIM [18] and Simplez/Algoritmez [19] are good examples of simulators. SPIM is a DLX pipelined computer simulator written in the C programming language. It is able to simulate a subset of the DLX instruction set. It uses the same events, signals, logic functions and style of description as in the lectures and textbook. However, they only simulate the CPU at an external level. A full-featured DLX simulator is presented in [20]. It is also based on the DLX simulator provided with the textbook but the interface is friendlier.

Several simulators of a processor can be found in the literature. We detail some of them and their principal features below. The MS is a very simple processor with a memory-memory architecture. The extreme simplicity of MS (it has only four instructions) makes it possible to illustrate some basic concepts of computer architecture, such as the instruction phases. The MS processor is described in [21] and a simulator can be found in [22].

An eight-bit computer is presented in [23]. Instructions can be executed with a single step switch or run with a clock. It is supported with an assembler patterned after the MIPS assembler used with the SPIM simulator. The computer is emulated by using the open source logic emulation package MultiMedia Logic [24].

SEP (Students' Experimental Processor) [25] was designed to be used in different computer architecture courses. For this reason, it integrates different types of architectures: Memory-Memory, Accumulator, Extended Accumulator, Stack, Register Memory, and Load Store. It was modelled using VHDL and is ready to be implemented on FPGAs. SEP was designed as the first part of an ambitious project consisting of four stages: design a processor, build a simulator, develop a compiler and develop an intergraded system.

In [26], a tool to introduce the students of computer science to the area of computer architecture in only 14 hours of theory and six of practice is presented. This tool allows students to learn the concepts of instruction set, addressing modes, the internal architecture of a CPU and the sequence of steps of the instructions in an easy way.

A debugger is presented in [27], which contains a set of virtual assemblers and a virtual machine. Both tools allow the students to understand the differences and similarities between architectural styles of computer processors. Students write programmes in the virtual assembly code. Programmes are compiled and executed in the virtual machine and the students can follow the execution of the programmes step by step or at full speed. As in [25], several types of architectures are considered: Accumulator, Stack, Memory-Memory, Load-Store and an Index Machine.

A complete and detailed survey can be found in [28]. Other authors offer access to advanced microprocessor hardware through another simulator in [29].

This paper presents a simulator of the rudimentary machine, called SiMR, which allows students to perform practical activities in the field of computer architecture. SiMR is based on the experience gained in design and development of simulators during the past twelve years at the Universitat Oberta the Catalunya.

The rest of the paper is organised as follows. Section 2 summarises the main characteristics of the Universitat Oberta de Catalunya (UOC). Section 3 describes the microprocessor used at the UOC to introduce the fundamentals of computer structure. Section 4 presents an overview of the first few releases of the simulation software tool. Section 5 describes the latest simulator version. Section 6 presents the results of a questionnaire which has been presented to the students of Computer Architecture and Organisation. Finally, the most relevant conclusions and some guidelines for future work are drawn in Section 7.

2. Universitat Oberta de Catalunya

The Universitat Oberta de Catalunya [30] (Open University of Catalonia, UOC) is internationally recognised as the first distance higher education institution to issue educational programmes using exclusively the Internet as the basis of the academic activity. Since the early foundation of the UOC, in 1995, its educational system relays on a Virtual Campus platform which allows the students and the tutors to interact assuming no coincidence in time or space. Nowadays, more than 40,000 students are enrolled in the different graduate, master, postgraduate and Ph.D. programmes at the UOC.

The UOC's student profile is quite different from that of the 'traditional' face-to-face universities. The standard UOC student is an adult person in his/her thirties or forties, with a family, employed and with very little time available for study. Many of these students have other university degrees and enrol in the UOC to acquire knowledge or skills in a new field. Thus, motivation is often one of their most remarkable qualities.

The web-based Virtual Campus platform

provides the means for inter-communication, such as forums for the exchange of information between the members of the community. Some of the features of the Virtual Campus are the following:

- interactive communications between the students and course tutors both asynchronously and, exceptionally, synchronously;
- interactive communications between the students;
- access to the UOC information resources (course materials, libraries, bulletin boards, databases, on-line and offline bibliography, etc.);
- access to the administrative services.

Using the incorporated communication tools and the web (at any Internet-connected computer), the students can exchange messages, ask questions to the course tutors and make enquiries without time constraints.

The UOC faces an important challenge as it develops a new concept of educational model: from teaching to learning. The course tutors are no longer transmitters of knowledge but rather a guide in the learning process, for which the student is the ultimate responsible. This is a student-centric model, since the student is the central element and the other elements are made available to support the learning process.

The following elements are directly involved in the learning process: the learning materials (e.g. installation instructions, software manuals, FAQ, theoretical materials, recommended bibliography, electronic publications, full text databases and examples of solved practical activities and exams), the course tutors, the continuous assessment system and the virtual library. All of these elements are integrated into the Virtual Campus.

The challenge of the UOC's educational model is far more evident in the technological programmes, which usually require practical activities. In face-to-face universities, these activities are often carried out within computer labs using real hardware. Obviously, this possibility is not suitable for a distance university. Therefore, the UOC has created virtual labs for the students of the different computer engineering science and telecommunication programmes.

One of the first few mandatory subjects in these technical programmes is the course on Basic concepts of Computer Architecture and Organisation (CAO) [2, 31], called *Estructura i Tecnologia de Computadors*, in which the students develop the following skills related to the fundamentals of computer hardware:

- the usage and comprehension of the different representations in the binary numeral system (unsigned and signed integers and fixed and floating point);
- the analysis and design of combinational circuits (Boolean algebra, logic gates, truth tables, Karnaugh maps and combinational blocks);
- the analysis and design of sequential circuits (flip-flops, sequential blocks and the Moore's model of a state machine);
- basic computer structure or "the Rudimentary Machine" (basic structure, machine language, assembly language, the processing unit, the control unit and programme execution);
- basic concepts about the Input/Output system and peripherals.

More than ten thousand students have followed this course since it started in 1997. Hence, the number of students per year (the subject is offered twice a year) ranges from 1,500 to 2,000. In order to acquire the technical skills, students must carry out several practical activities related to the internal work of the different computer components (memory, registers, buses, ALU, etc.).

Face-to-face universities also use simulators to teach the basics of CAO, but the students are usually tutored by lecturers within computer labs during the development of practical activities. In a virtual environment such as the UOC's, the students must be able to solve many different problems on their own (in spite of the support by the course tutor) and, hence, the features of the simulator must be much wider than in face-to-face environments. This paper describes the use of SiMR [32], a software tool which has been developed to satisfy these requirements.

CAO students carry out five Continuous Assessment Activities (CAA) throughout the semester at home. Two of these CAA are related to the Rudimentary Machine (RM) and must be solved using SiMR. Moreover, they must pass an on-site exam. Four of the CAA are elective (they are used to support the learning process and can only increase the final mark). The fifth CEA is a mandatory practical exercise (which can be considered as a virtual part of the final exam) about the RM, and must be carried out using SiMR. The educational objectives covered by the CAA related to RM are the following:

- operation of a von Neumann computer;
- internal structure and operation of the RM's processing unit, specified as a circuit formed by combinational and sequential blocks;
- operation of a control unit specified, a state machine using the Moore's model;
- operation of a small programme (less than 30 lines written in RM's assembly language);

- ability to write simple short programs using the RM's assembly language.

All of these educational objectives can be achieved by using SiMR if appropriate suggestions are made to the students.

3. Rudimentary machine: a basic processor to introduce computer architecture

The Rudimentary Machine [33, 34] is a pedagogic computer. It was designed in 1991 in the Department of Computer Architecture of the Universitat Politècnica de Catalunya [35]. The main objective of the RM was to make learning of the basic concepts on architecture and structure of computers easy to the students of the first year of computer engineering.

The RM is a RISC computer [18] whose processor has a von Neumann architecture [36] of load-storage. The main memory has 256 words of 16 bits. These 16 bits are simultaneously accessed. The instructions have a fixed size of 16 bits and data are integer numbers coded in two's complement in 16 bits. The datapath (processing unit) has a register file with 16 8-bit registers, an 8-bit programme counter (*PC*) and a 16-bit instruction register (*IR*). The process status word (*PSW*) had originally the flags *N* and *Z* [33], but the simulator SiMR incorporates also the overflow bit for integers (*V*). The control unit is very simple and reflects the different phases in the execution of an instruction. It is modelled as a 10-state machine with 11 output signals. An optimised version with only six states (but semantically more complex) allows the execution time of the programmes to be improved by merging some states in the state diagram, reducing the cycles per instruction (CPI) that way.

The RM has three types of instructions:

- Memory access: load and store instructions use the relative mode to access the memory with an 8-bit offset. The assembly format is as follows: Load offset(Ri), Rd and Store Rs, offset(Ri). Rd and Rs are the target and source registers of instructions load and store respectively. Ri is the index register. Rd, Rs and Ri are registers from the register file. The offset is formed by the eight least significant bits of the index register and is used for the calculation of the effective address of the operand (Ri + offset).
- Arithmetic and logical: the RM provides two instructions of addition and two of subtraction (a register-register and another register-immediate of every type, with an immediate operand of five bits), an AND instruction register-register and an instruction for a one-bit right arithmetic shift of a register.
- Branches: the RM has an unconditional branch instruction and six of conditional branch for integers, which take into account the three flags V, Z and N. The branch conditions are the following: greater than, lesser than, greater than or equal to, lesser than or equal to, equal to and not equal to. These instructions use the absolute mode to address the memory.

The RM assembly language allows the definition of macroinstructions (or simply macros). A macroinstruction [33] is a new assembly instruction defined as a sequence of native RM instructions or macroinstructions. Macros are parameterised and allow the user the definition of new assembly instructions, making the writing and understanding of programmes easier.

4. SIMR: a brief history

SiMR, the simulator described in this paper, has been developed according to the experience acquired in the design and use of previous RM simulators [32, 37].

4.1 Previous releases of SiMR

The first release, labelled 1.0, appeared in 1997. It ran under the DOS operating system and the main screen was divided into four fixed-size frames which presented the following items:

- the datapath, which could be enlarged up to full-size screen to show the value of the control signals;
- the Control Unit state diagram;
- the assembly code of the programme which was being executed;
- a special frame which allowed the user to control the simulator activity (working as a command interpreter).

The assembler programme was not integrated with the simulator, and programmes written in assembly code should be translated to machine code before being executed by the simulator. The assembler programme was divided into two sequential steps:

1. pre-assembler: it evaluated the macroinstructions code and translated it to native assembly code;
2. post-assembler: it translated the native assembly code generated in step 1 to machine code.

A year later, in 1998, the second release was available, labelled as 2.0. It was designed to run in a Windows 3.11 environment and had numerous

advantages over the 1.0 release. Besides the easiness of handling offered by Windows 3.11 (mainly management of events through buttons, instead of by using written orders in the command interpreter), the new version included the possibility of carrying out time diagrams for the selected control signals, registers contents and buses. It also allowed the user to switch, in run time, the way macros are shown. The two possibilities were showing the complete internal code or simply their name and parameters.

The main screen was divided into three static areas: one for the datapath (which could be extended to a full-size screen as in the previous release), another for the Control Unit state diagram and a third to show the code of the programme to be executed. As in the previous version, the assembler programme was a separate application (the same as for release 1.0 but some bugs were fixed).

4.2 SiMR 3.0

The next version was labelled 3.0 and was designed to be executed on a Windows 95 or 98 environment. It was developed at the end of the nineties and made available in 2000. This tool, named SiMR, is more than a simple simulator: it is a complete environment for developing, assembling and executing programmes in the RM assembly language. Unlike the two previous versions, SiMR 3.0 incorporates the possibility of using both Control Units (represented by six and ten state machines respectively). The main screen has only two variable-size frames; one for the datapath and another for the assembly code (see Fig. 1). The Control Unit appears separately as a small frame of dialogue in the datapath window which can be hidden. The user interface is very similar to most of the Windows applications, with tooltips in the buttons, transparent-hiding of the non-active options and a modern and complete help, so that the student can use the tool more speedily and more intuitively than with previous releases.

The assembler programme has been improved and integrated into SiMR 3.0, as well as a simple text editor which allows editing, debugging, compiling and simulating of the program in the same working environment. The two steps of the assembler programme are now transparent to the user (they are sequentially executed without user intervention) and some bugs detected in previous versions were fixed.

SiMR allows changing of the contents of any memory address or register in the datapath manually, a really useful feature for new users. It enables the student to define an artificial initial state of the RM and to test the result of executing some specific instruction. The memory and register file

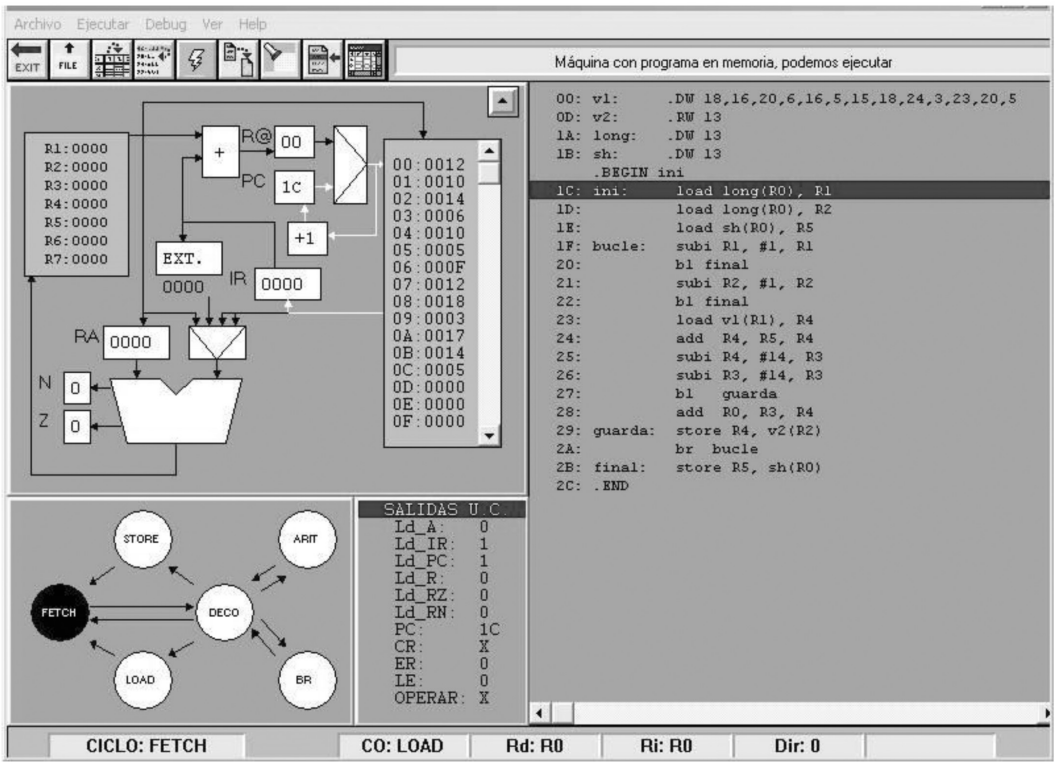

Fig. 1. Main Screen of SiMR 3.0.

contents are presented in several formats (decimal, binary two's complement, hexadecimal and the translation to assembly code for the main memory contents), which makes the interpretation of data easier.

Undoing the actions carried out during the last cycle or during the complete execution of the last instruction is also possible. This option is adapted to repeatedly test aspects of the instruction execution which may have not been correctly understood by the student.

Finally, SiMR 3.0 automatically generates several types of reports to make the work easier for the student as well as presenting results to the tutor.

All RM simulators described in this section can be freely downloaded from the official RM web page [32]. The next section describes the main features of SiMR 3.0 and 3.1 in detail.

5. SiMR 3.1: The next RM Simulator

SiMR is simple and intuitive to work with. When the simulator starts, a dialogue box which requests the name of the student or the group of students (up to three) appears. Filling in the data is not mandatory, but the box must be filled up if the student is solving an assessment activity, since the student names appear in all the reports generated by the simulator. SiMR has a safety system to guarantee that the work is really carried out by the student/s, since a warning is activated when the tutor tries to read a report in which some information has been manipulated.

Next, the two-frame main screen (see Fig. 1) appears. Initially, this screen presents the datapath and the Control Unit in the left frame, and the code window on the right. Editing or uploading a new programme is required before starting the simulation. Once the programme has been assembled:

- the simulator stores it at the position 0 of memory (unless the program specifies another position);
- the Programme Counter is initialised with the address of the first instruction to execute;
- the FETCH state is selected in the Control Unit;
- the simulator leaves control to the user.

All the operations the user carries out from this point remain reflected in the working reports which SiMR generates automatically.

As can be observed in Fig. 1, the Menubar and Toolbar are on the top of the main screen. The buttons in the Toolbar allow fast access to the most frequent options of the Menubar. A Statebar presents information, at the bottom of the screen, about the current state of the Control Unit and the instruction in progress (operation code and location of the operands).

The following subsections describe the main features of Menubar and Toolbar briefly.

5.1 Menubar

The Menubar has the following options:

- **File**: It has the options new, open, save, save as, close and exit. Files are programmes written in assembly code (.asm) or in machine code (already compiled, .cod);
- **Edit:** It allows the user to undo the last action and to cut, to copy and to paste text in the code window. It is active when a programme in assembly language is being edited;
- **View**: it allows the user to view/hide the Toolbar and the Statebar;
- **Preferences**: it allows the clock frequency to be changed; the state of the RM (contents of the main memory positions and datapath registers) to be saved, restored, viewed and changed; also change the RM configuration (delay of the different components) and the interface colours; define a file of macros by default, update the identification of the group of students and select between the two state machines of the Control Unit;
- **Run**: it allows selection whether the simulation of the programme's execution is done cycle by cycle (step), instruction to instruction (instruction) or until a breakpoint is found or the program is finished (run). It can also stop the simulation, insert breakpoints and reset the simulator;
- **Chronogram**: it allows definition of the signals and buses which will be shown in the chronogram (time diagram), saves the current chronogram, shows a previously saved chronogram or the current one and prints a chronogram;
- **Reports**: Different kinds of reports are given by SiMR in a transparent way to the user. They can be created, printed or just shown on the screen. The user only has to decide (by clicking the option) the moment at which the report starts to store the actions performed by SiMR. By default, all reports start when the simulator begins running. Reports contain information about the process of assembly (assembler and machine code, Table of Symbols and assembly errors), activity (a trace of the steps followed by the student is saved, so that the tutor can follow step by step the work done by the student), execution (save only the steps related to the execution of a given programme) and results

(state of the RM before and after the simulation);
- **Compiler**: It allows assembling, assembling and executing (only if no assembling errors are found) and showing the Table of Symbols;
- **Window**: It is used to organise the open windows (several instances of the simulator can be simultaneously open). It allows situating them in *cascade, mosaic* or *vertical mosaic*. The bottom of this menu allows a fast access to the current open windows;
- **Help**: It accesses to a very powerful and complete on-line help tool and supplies technical information about both the simulator and the Rudimentary Machine.

5.2 Toolbar

The Toolbar allows to:

- create, open and save a programme;
- cut, copy and paste text in a programme;
- undo the execution of the last instruction or the last cycle of execution;
- view/hide the control signals in the datapath;
- access the dialogue box which allows you to view and edit the contents of the main memory, the register file and the different datapath registers;
- view a chronogram of the control signals, registers contents and buses predefined by the user;
- reset the simulator (but maintaining the current programme stored in memory);
- advance the simulation by a cycle, an instruction or until it finds the first breakpoint or the end of the programme;
- stop the simulation of the programme in execution;
- insert or remove a breakpoint;
- assemble the programme or execute (assembling it previously, if necessary) the program stored in memory;
- give information about the simulator;
- interactive help.

5.3 Other features of SiMR

Besides the features mentioned in the previous sections, SiMR allows the programme to expand/compress the macros through a special button strategically situated on the right of the code frame.

The code window allows the user to insert or to remove a breakpoint or a stop easily by positioning the cursor at any line of the programme and by pressing the right mouse button. A stop is a temporary breakpoint, which disappears once the programme stops for the first time in that instruction.

The dialogue box of the Control Unit allows selection between two possible state machines of the Control Unit: one optimised with six states, ideal for advanced users, and another semantically simpler, ideal for new users, with 10 states, in which the execution phases of an instruction are clearly shown.

Finally, the content of the main memory, the register file and the datapath registers can be changed by simply clicking on the element. In the same way, the delay of any datapath component can also be changed at any time.

6. Results evaluation

In order to obtain the opinion of the students concerning the SiMR and to know about their experience using the virtual learning environment, a web questionnaire was prepared in the Autumn 2009 semester. Similar works are found in [5] and [40]. The objectives of our study were as follows:

1. know the level of student satisfaction;
2. know whether student learning was improved by the use of SiMR;
3. gather information to improve the SiMR tool: weak and strong points;
4. know new features to be incorporated into SiMR.

The web questionnaire was available for ten days to 676 students (of whom 203 participated) through a link in the Virtual Classroom. The survey was anonymous and voluntary, and respondents were restricted to two subjects which use the VCAOLab: 1-Computer Architecture and Organisation (part of the Computer Science programme) and 2-Computer Fundamentals (part of the Telecommunication programme).

The web questionnaire consisted of ten questions (Q1–Q10) divided into four parts: the student profile part, the SiMR satisfaction part, the effect on the student's learning process part, and the features and advantages using SiMR part. The recommendations in [38, 39] were taken into account to design this survey.

In the first part, questions 1 to 4 (Q1–Q4) were concerned with the profile of the respondents. The results are shown in Table 1. The respondents were highly heterogeneous. Combining the responses of both subjects, the results show that 69% of the students were enrolled at the UOC for only one or two semesters, while the remaining 31% were enrolled for three or more semesters (Q1). The results of Q2–Q4 show that the majority of the students had no previous experience in simulators (62.56%), computer architecture background (64.53%) and assembly language (60.59%).

Table 1. Q1–Q4: frequency results of students' profile

Id	Question description:	Answer	N	Frequency
Q1	How many semesters have you studied at UOC in a Virtual Learning Environment?	*1 semester*	*100*	*49.26%*
		2 semesters	*40*	*19.70%*
		3 semesters	*28*	*13.79%*
		4 or more semesters	35	*17.24%*
		Total	*203*	*100%*
Q2	Had you studied with simulators in a Virtual Learning Environment before this semester?	*Yes*	*76*	*37.44%*
		No	*127*	*62.56%*
		Total	*203*	*100%*
Q3	Did you have previous knowledge in computer architecture?	*Yes*	*72*	*35.47%*
		No	*131*	*64.53%*
		Total	*203*	*100%*
Q4	Did you have previous knowledge in the assembly language?	*Yes*	*80*	*39.41%*
		No	*123*	*60.59%*
		Total	*203*	*100%*

Table 2. Q5–Q9: satisfaction questions

Id	Question description
Q5	What is your general satisfaction grade of the SiMR?
Q6	What has been the facility level for installing the simulator?
Q7	What has been the facility level for using the simulator?
Q8	Does SiMR help you in the learning process?
Q9	Can you evaluate SiMR from indispensable to not necessary?

To obtain information about the level of satisfaction using SiMR, the second part of the web questionnaire had 3 questions (Q5–Q7). Next, the third part of the questionnaire had two questions (Q8–Q9) to achieve information about the effect of the SiMR tool on the students' learning process. The descriptions of these questions (Q5–Q9) are shown in Table 2. Students were asked to evaluate from 1 to 4 their satisfaction grade and how their learning had improved, as follows: 1 (high), 2 (medium-high), 3 (medium-low) and 4 (low). They were instructed that by 'improved learning' they should take into account the ability of solving exercises related to the course before and after getting acquainted with the SiMR.

The number of respondents, the frequency of each value, the percentage of answers with 1 or 2 value, the mean value, the standard deviation and the variance for these five questions are summarised in Table 3.

The first analysis of the survey results shows high satisfaction of students with SiMR. Between 73.13% and 89.45% of students are highly or medium-highly satisfied with the SiMR, relating to three different aspects: installation, use and general satisfaction.

Secondly, the results show that the majority of students considered SiMR as a powerful tool to improve the learning process, since 88.61% of students reported that the SiMR had helped them to improve the learning process and 88.67% evaluated it as an indispensable (value = 1) or highly important tool (value = 2) for learning the basic concepts about Computer Architecture. The mean values of these two questions (Q8 and Q9) were of 1.62 and 1.88, respectively.

Finally, in the fourth part of the web questionnaire, a multiple-choice question (Q10) was given in order to find what the most important features and advantages of using SiMR are, in the opinion of students. According to Table 4, the 'easy use' (54.77% of the answers) and the 'high capacity to learn concepts' (50.75% of the answers) are the most important features and advantages from the students' point of view.

Table 3. Q5–Q9: descriptive and frequency analysis of SiMR satisfaction

	N	1 High	2	3	4 Low	1–2	Mean	Std. Dev.	Variance
Q5	201	46	101	39	15	73.13%	2.11	0.842	0.708
Q6	202	112	64	19	7	87.13%	1.61	0.796	0.634
Q7	199	85	93	17	4	89.45%	1.70	0.709	0.502
Q8	202	102	77	20	3	88.61%	1.62	0.722	0.522
Q9	203	55	125	16	7	88.67%	1.88	0.687	0.473

Table 4. Q10: what are the most important advantages of using SiMR?

N = 199	Answers	Frequency
Easy to use	109	54.77%
Capacity to learn concepts	101	50.75%
Anywhere	83	41.71%
Anytime	62	31.16%
Others	14	7.04%

7. Conclusions

In this paper we present SiMR, a job environment which integrates an assembler programme and a simulation tool of the Rudimentary Machine. The Rudimentary Machine is a simple RISC computer designed to learn the basic concepts about computer architecture. SiMR can be used as an effective remote tool and has been designed to help Computer Science Engineering students to understand how a processor executes instructions.

SiMR allows editing, assembling and executing cycle by cycle, instruction by instruction, or until the end of a programme. Breakpoints can be included anywhere in the assembly code and a timing diagram including the desired signals can be automatically generated. Two different Control Units are available. One of them emphasises the execution phases of an instruction. The other focuses on reduction of the CPI for each instruction. SiMR also generates, in a transparent way, different reports to show the work developed by the student. These reports incorporate a security system to avoid forgeries or illegal copies.

In order to analyse the usefulness of the proposed remote tool, a survey was sent to the students in Autumn 2009. The results show that most of the students consider that SiMR helped them to learn and, moreover, it as an appropriate tool to be used in a Computer Architecture and Organisation subject. In addition, they conclude that SiMR is a highly necessary or even an indispensable resource to learn the basic concepts about computer architecture in a virtual learning environment.

Acknowledgment—The authors would like to thank all the students which have contributed with their work to design the different RM simulators. This paper would not have been possible without their valuable work. This work was partially supported by the Spanish MEC and the FEDER funds under grant TSI2007-65406-C03-03 'E-AEGIS', TIN2007-60625, TIN2006-15107-C02-01 'PERSONAL' and CONSOLIDER CSD2007-00004 'ARES', funded by the Spanish Ministry of Science and Education.

References

1. I. J. T. F. on Computing Curricula, *Curriculum guidelines for undergraduate degree programs in computer engineering*, IEEE Computer Society Press and ACM Press, 2004.
2. I. J. T. F. on Computing Curricula, *Computing curricula 2005*, IEEE Computer Society Press and ACM Press, 2005.
3. E. Tovar, M. Castro, Building common spaces in engineering education: A review from icece05, *IEEE Transactions on Education,* **50**, 2007, pp. 79–84.
4. S. Sheppard, A. Colby, K. Macatangay and W. Sullivan, What is Engineering Practice?, *The International Journal of Engineering Education,* **22**(3), 2006, pp. 429–438.
5. J. Prieto-Blázquez, J. Arnedo-Moreno and J. Herrera-Joancomartí, An Integrated Structure for a Virtual Networking Laboratory*, IEEE Transactions on Industrial Electronics,* **55**(6), 2008, pp. 2334–2342.
6. P. Bauer, J. Dudak, D. Maga and V. Hajek, Distance Practical Education for Power Electronics, *The International Journal of Engineering Education,* **23**(6), 2007, pp. 1210–1218.
7. K. Chiu, *What are the benefits of a virtual laboratory for student learning?*, HERDSA Annual International conference, Melbourne, 1999, pp. 12–15.
8. U. Harms, *Virtual and remote labs in physics education*, Second European Conference on Physics Teaching in Engineering Education, Budapest, 2000.
9. L. J. Leitner and J. W. Cane, A virtual laboratory environment for online it education, in SIGITE '05: *Proceedings of the 6th conference on Information technology education.* New York, NY, USA: ACM Press, 2005, pp. 283–289.
10. J. E. Corter, J. V. Nickerson, S. K. Esche and C. Chassapis, Remote versus hands-on labs: a comparative study, *FIE'04: 34th Annual Conference on Frontiers in Education*, **2,** 2004, pp. F1G: 17–21.
11. C. Levert, and S. Pierre, Designing distributed virtual laboratories: Methodological and telecommunications aspects, *International Journal on E-Learning,* **2**(3), 2003, pp. 18–28.
12. A. M. Rak and M. Godziemba-Maliszewski, A proposal of virtual laboratory structure, *Proceedings of The 23rd IEEE in Instrumentation and Measurement Technology Conference,* 2006*,* pp. 847–850.
13. A. Repenning and J. Ioannidou, *Collaborative use and design of interactive simulations*, in CSCL Proceedings, Stanford, CA, 1999.
14. E. F. Sowell, Programming in Assembly Language VAX-11, Addison-Wesley, 1987.
15. H. B. Diab and I. Demashkieh, A computer-aided teaching package for microprocessor systems education, *IEEE Transactions on Education*, **34**(2), 1991.
16. W. D. Henderson, Animated models for teaching aspects of computer systems organization, *IEEE Transactions on Education*, **37**(3), 1994.
17. Barry B. Brey, *Programming the 80286, 80386, 80486 and Pentium-based Personal Computer, MacMillan Pub Co., 1996.*
18. J. L. Hennessy and D. A. Patterson. *Computer Architecture, Third Edition, A quantitative approach*. Morgan Kaufmann, 2002
19. G. Fernández, *Conceptos básicos de Arquitectura y Sistemas Operativos, Sistemas y Servicios de Telecomunicación,* 1994.
20. P. López, R. Calpe, DLXVSim: A DLXV vector computer simulator http://dlxv.disca.upv.es/tools/dlxv.html
21. E. Aiguadé, J.J. Navarro, and M. Valero-García, *La Máquina Sencilla: Introducción a la Estructura Básica de un Computador*, Edicions UPC, Col.lecció Aula, 26, 1992.
22. J. M. Angulo, I. Angulo, B. García, J. Sáez. *Microcontroladores avanzados dsPIC, controladores digitales de señales.* Ed. Paraninfo, 1996. (In Spanish).
23. T. D. Stanley and M. Wang. An Emulated Computer with Assembler for Teaching Undergraduate Computer Architecture. *Proceedings of the 2005 workshop on Computer architecture education: held in conjunction with the 32nd International Symposium on Computer Architecture.* June 2006. pp. 38–45.
24. Softronics Inc. http://www.softronix.com (Accessed 23 April 2010).
25. L. S. K. Udagama and J. C. Geeganage, Students' experimental processor: a processor integrated with different

types of architectures for educational purposes, *Proceedings of the 2006 workshop on Computer architecture education:* held in conjunction with the *33rd International Symposium on Computer Architecture*. Boston, Massachusetts. 2006.
26. J. R. Arias and D. F. Garcia, Introducing computer architecture education in the first course of computer science career, *Computer Architecture Newsletter*, IEEE Computer Society, February 1999, pp. 37–39.
27. O. Agren, Virtual machines as an aid in teaching computer concepts, *Computer Architecture Newsletter,* IEEE Computer Society, September 2000, pp. 72–76.
28. G. S. Wolfe, W.Yurcik, H. Osborne, and M.A.Holliday, *Teaching computer organization/architecture with limited resources using simulators*. Proceedings of the SIGCSE'02, Covington, Kentucky, USA.
29. B. Fechner, J. Keller and W. Schiffmann, *Evaluation of a Virtual Computer Architecture Lab*, FernUniversität Hagen, 2006.
30. Universitat Oberta de Catalunya. http://www.uoc.edu/portal/english/ (Accessed 23 April 2010).
31. *The Generic ICT skills profiles, future skills for tomorrow's worlds*, Office for Official Publications of the European Communities, 2001.
32. Máquina Rudimentaria (MR). http://docencia.ac.upc.edu/eines/MR/ (Accessed 23 April 2010 (In Spanish).
33. R. Hermida, A. M. del Corral, E. Pastor, and F. Sánchez. *Fundamentos de Computadores*. Ed. Síntesis, 1998.
34. E. Pastor, F. Sánchez, and A.M. del Corral, *A Rudimentary Machine. Experiences in the Design of a Pedagogic Computer*. Workshop on Computer Architecture Education (held in conjunction with ISCA-25), Barcelona (Spain), June 1998D. Also in *Computer Architecture Newsletter*, IEEE Computer Society, 1999, pp. 51–53 http://tab.computer.org/tcca/NEWS/feb99/index.html
35. Universitat Politècnica de Catalunya. http://www.upc.edu/eng (Accessed 23 April 2010).
36. A. W. Burks, H. H. Goldstine and J. von Neumann. *Preliminary discussion of the logical design of an electronic computing instrument*. A.H.Taub ed., collected works of John von Neumann, The Macmillan Company, **5,** 1963 , pp. 34–79.
37. F. Sánchez and L. Ibarria. *SiMR: Entorno de simulación de la Máquina Rudimentaria.* VII Jornadas de Enseñanza Universitaria sobre Informática JENUI'01, July 2001.
38. D. A. Dillman, R. D. Tortora and D. Bowker, *Principles for constructing web surveys*. Digital Equipment Corporation, Washington, Tech. Rep. 1998, p. 50, .
39. D. Solomon, Conducting web-based surveys. *Practical Assessment, Research and Evaluation*, **7**(19), 2001.
40. J. Prieto-Blázquez, J. Herrera-Joancomartí and A. Guerrero-Roldán, A Virtual Laboratory Structure for Developing Programming Lab, IEEE Transactions on Industrial Electronics, *International Journal of Emerging Technologies in Learning* (iJET), **4**, 2009, pp. 47–52.

Fermín Sánchez obtained a degree in Industrial Electronics for the E.A. SEAT in 1981, Engineer in Computer Science since 1987 and Doctor in Computer Science since 1996, the two last titles were obtained in the Universitat Politècnica de Catalunya (UPC). At present, his research is focused on the development of new multithread architectures for VLIW processors and in the development and introduction of new educational strategies to adapt the Spanish university studies to the EHEA. Since 1987 he has worked in the Computer Architecture Department at UPC, where he has been assistant professor since 1997. He is also tutor at the Universitat Oberta de Catalunya (UOC) since 1997. He has several publications related to his subjects of research and is reviewer of numerous conferences and journals. Also, he is author and coauthor of several books, some of which have been awarded with international prizes. He has been a member of the organisation committee of different conferences and other national and international events, is coordinator in the Barcelona Supercomputing Centre—Centro Nacional de Supercomputación of the European mobility programme HPC-Europa since March 2004, director of the Museum of Computer Architecture since February 2006, member of the board of management of Cercle Fiber since November 2002 and vice dean of innovation at the Facultat d'Informàtica de Barcelona since May 2007.

David Megías achieved the Ph.D. degree in Computer Science in 2000, the M.Sc. degree in Computer Science (Advanced Automatic Control) in 1996 and the BSc degree in Computer Engineering in 1994, all of them at the Universitat Autònoma de Barcelona (UAB) in Spain. He has made research stays at the Department of Engineering Science of the University of Oxford and at the Departamento de Ingeniería de Sistemas y Automática of the Universidad de Valladolid, in both cases as a visiting scholar. He was an assistant lecturer at the UAB from September 1994 to October 2001. Nowadays, he is an associate professor at the Universitat Oberta de Catalunya (UOC) in Barcelona (Spain), with a permanent position since October 2001. He is the Associate Director of the UOC's Doctoral Programme in Information and Knowledge Society and the coordinator of the Network and Information Technologies field of this programme. His current interests include information security and, more precisely, copyright protection, watermarking and data hiding schemes. He has participated in several national and international joint research projects both as a contributor and as a manager (main researcher)

Josep Prieto-Blázquez achieved the Ph.D. in Computer Science, in January 2009 from the Universitat Oberta de Catalunya and received the MS degree in Computer Science from the Universitat Politècnica de Catalunya. Since 1998 he has worked as a lecturer in the department of Computer Science and Multimedia at the Universitat Oberta de Catalunya, where he has been director of the Computer Engineering (CE) programme since February 2001. His line of research centres on exploratory and application technology in the field of ICT. He has also participated in the wireless, free software and virtual learning environments projects.