Chapter 14

USING JAVA AND C# FOR EDUCATIONAL SIMULATORS: THE CASE OF SIMPLE-2

Ramón-Ángel Fernández, Luis Panizo and Lidia Sánchez
Department of Electrical Engineeering and Electronics. University of Leon
Campus de Vegazana s/n, 24071-León (Spain)

Abstract: Due to the impact of the new technologies in our Society, special efforts to use them for teaching-learning computer architectures have been done. This work presents a simulator of a simple architecture (Simple-2) using two different technologies: Java and C#. In the one hand, the Java version is an applet which runs inside a web browser; in the other hand, the C# version runs as an application that needs a virtual machine be installed in the system. The impact of both technologies on first year Computer Science students has been analyzed, as well as the degree of learning achieved when using the simulator for learning an architecture in a semi-autonomous way.

Key words: computer architecture simulation; educational simulator.

1. INTRODUCTION

One of the core fields of Computer Science education at the University consists on learning those concepts related to computer structure and organization, as well as the interactions among computer components in order to run sequences of instructions. The result of this learning is a key matter in order students can successfully progress with their studies.

To complete this learning, a first stage may be considered to let the student acquire a global vision of computers, recognizing their functional units and the operation of every single unit. After that, the student has to assimilate the different ways in which those functional units interact. During this stage, ability to relate concepts and integrate them into a higher abstraction level is required, which involves a considerable difficulty, especially for first year students.

To help the students during this second stage, the concepts explained in the lectures should be reinforced. Solving exercises and doing practices in the laboratory usually are not enough, and other additional tools to stimulate the students to learn should be used. Several software solutions have been developed that allow the students to interact in order to simulate computer behavior and to observe program execution events[1]. There is also software that graphically represents the activity of every component of a computer and the way those components interact[2,3]. Students, in general, and Computer Science students, in particular, find quite attractive the possibility of working and learning through the Internet. That is one reason to explain the efforts in developing network learning tools and checking their efficiency in the learning process.

In the department of Electrical Engineering and Electronics at the University of Leon, a simulator of a simple processor (Simple-2[4]) has been developed so as to be used by the students through a web server. There are two available versions: a Java applet, which runs in a web browser; and a C# application that must be downloaded and runs locally in the student's computer. Anyway, both versions have the same interface and provide the same functionality. After using the software, its influence on the learning has been analyzed on the basis of the results of written tests fulfilled by the students.

2. JUSTIFICATION OF THE WORK AND OBJECTIVES

Computer Organization and Technology is one subject that students learn during their first year of the degree in Computer Science at the University of Leon (Spain). This subject involves two main blocks: the basics of *Analogical and Digital Electronics*, and, based on those fundamentals, *Computer Organization*.

The block related to Computer Organization begins with the description of the main functional units present in Von Neumann architecture based computers: Central Processing Unit, Memory Unit and Input/Output Unit. Next, the operation of the datapath is analyzed for a subset of one RISC processor, the MIPS, as described by Patterson[5]. The lectures are complemented with animated slide presentations that can be downloaded from the departmental website. Those presentations show the paths the data follow as well as how the control unit manages those paths. In the last part of the course, the concept of memory system is presented, including the possibility of using hierarchies to improve the performance. Finally, basic input/output techniques are studied.

In the laboratory, students learn to develop software using the MIPS assembly language and the *xspim* emulator under the Linux operating system. This software helps the student to assimilate the utility of the banks of registers, to understand how the information is stored in the memory system, and to test several input/output techniques. Anyway, *xspim* does not include any utility for datapath visualization, nor shows the operation of the control unit during program execution. Hence, students cannot check in the laboratory some of those concepts explained in the lectures and required for solving exercises. Therefore, we have decided to develop an additional simulator to provide the students a graphical tool to observe the operation of the computer components altogether.

With the objective of evaluating the impact of the use of a simulator in the learning process, we have decided to use an architecture different to the one described in the lectures. Then, the influence of the lecturer on the first approach to the architecture is bypassed, although students may talk to the professor and make questions (in the office, by e-mail, instant messages or Internet forums). The chosen architecture is the pedagogical computer Simple-2[4], that is used for teaching Computer Organization and Structure II at the *Escuela Universitaria de Informática* at the *Universidad Nacional de Educación a Distancia* (UNED, Spain).

Then, the students have to learn the Simple-2 architecture by themselves, just from a paper with an explanation and with the help of the simulators they can access through the Internet. In order to increase the motivation of the students, an extra bonus over their final qualifications is offered by means of a test about that architecture.

3. SIMPLE-2 SIMULATOR

As mentioned above, two identical versions of the Simple-2 architecture simulator have been developed[1] using platform independent technologies. The first version is a Java applet, which runs in any Internet browser. The other one is an application developed using C#. The use of this second version needs the user install a C# virtual machine as *mono* (http://www.mono-project.com).

[1] The two versions of the simulator have been developed as Final Projects by Jesús A. Fernández and Montserrat Sotomayor, students of Computer Science at the University of Leon.

3.1 User Interface

The graphical user interface has three different panels: one to edit the source code, one to visualize the machine status, and another one to visualize the datapath activity.

The panel for the source code (figure 14-1) lets the user write assembly code and also has a tool for detecting and displaying errors. Once the source code has no errors, the machine code is displayed for every instruction. This panel has three main objectives: students learn to write assembly code correctly, they check whether they are able to translate source code to machine code, and they also test their skills in using the hexadecimal representation for binary information.

The panel that displays the machine status (figure 14-2) shows the values of every register and also the values of the main memory cells during a program execution. This module helps the students analyze the evolution of the machine status as long as a program is executed.

The third panel (figure 14-3) graphically displays the activity in the datapath while the instructions are executed. This module provides a view of the functional units, the mechanisms they use to interact and the paths followed by the data to carry out the execution of each instruction. One Simple-2 instruction consists in a sequence of micro-instructions that need one clock cycle to finish. As one instruction is composed of a set of micro-operations, Simple-2 architecture divides each cycle into 4 sub-cycles. At every moment, the simulator shows the micro-instruction that is running and the values of its fields. In addition, those functional units that are involved in the current micro-operation are highlighted in the datapath.

To make easier the analysis of both, the machine status and the flow of data through the datapath, both versions of the simulator allow pausing and resuming the execution.

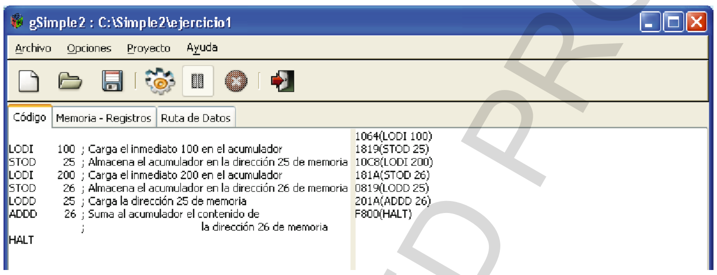
4. EXPERIMENTS

The experiments have been designed in order to analyze the influence of the simulator over a population of 190 students of Computer Organization and Technology at the University of Leon.

The questions used to evaluate the learning have the following objectives:

1. Finding out the version of the simulator used by the student: the Java applet or the C# application.

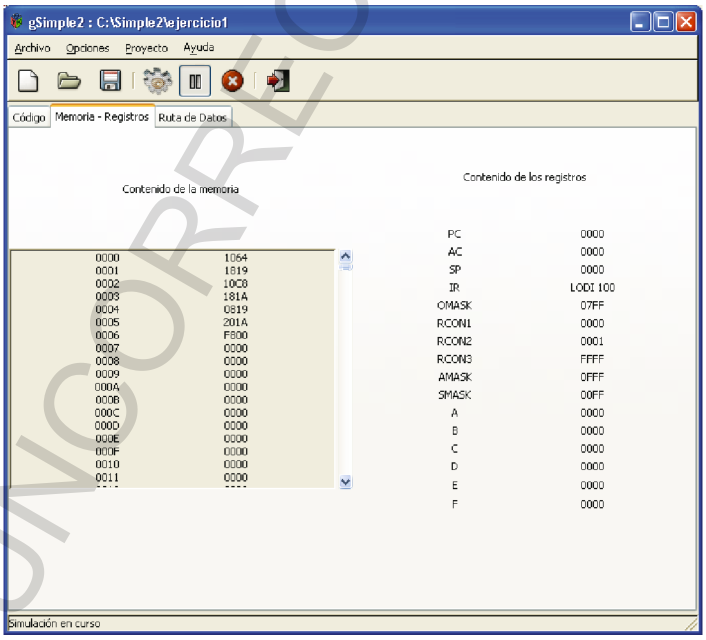
2. Evaluating the student skills in writing assembly code for the Simple-2 architecture.
3. Evaluating the student skills in identifying the computer functional units.
4. Evaluating the ability of the student for relating the functional units working altogether to the execution of instructions.



```
LODI    100  ; Carga el inmediato 100 en el acumulador
STOD     25  ; Almacena el acumulador en la dirección 25 de memoria
LODI    200  ; Carga el inmediato 200 en el acumulador
STOD     26  ; Almacena el acumulador en la dirección 26 de memoria
LODD     25  ; Carga la dirección 25 de memoria
ADDD     26  ; Suma al acumulador el contenido de
             ;                    la dirección 26 de memoria
HALT
```

```
1064(LODI 100)
1819(STOD 25)
10C8(LODI 200)
181A(STOD 26)
0819(LODD 25)
201A(ADDD 26)
F800(HALT)
```

Figure 14-1. Graphical User Interface: Panel for Simple-2 assembly code edition and machine code visualization.



Contenido de la memoria

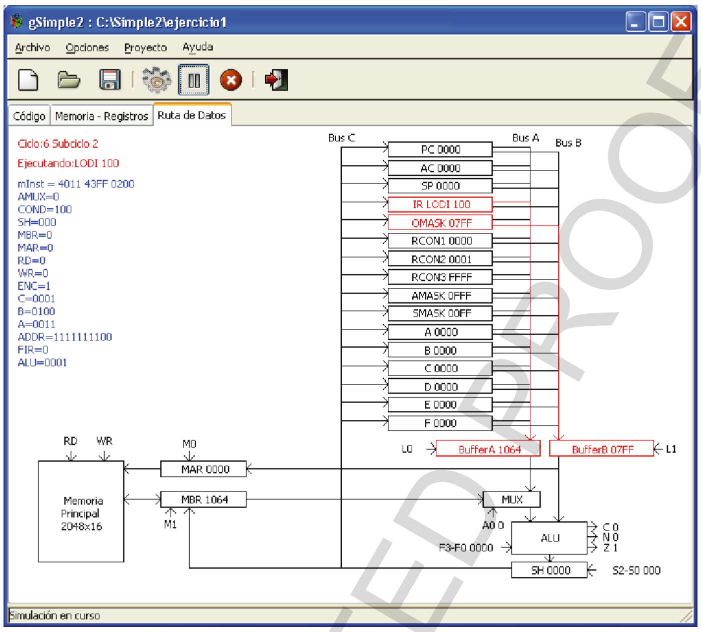
0000	1064
0001	1819
0002	10C8
0003	181A
0004	0819
0005	201A
0006	F800
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	0000
0010	0000
0011	0000

Contenido de los registros

PC	0000
AC	0000
SP	0000
IR	LODI 100
OMASK	07FF
RCON1	0000
RCON2	0001
RCON3	FFFF
AMASK	0FFF
SMASK	00FF
A	0000
B	0000
C	0000
D	0000
E	0000
F	0000

Simulación en curso

Figure 14-2. Graphical User Interface: Values of memory and registers during execution.



Ciclo:6 Subciclo 2

Ejecutando:LODI 100

```
mInst = 4011 43FF 0200
AMUX=0
COND=100
SH=000
MBR=0
MAR=0
RD=0
WR=0
ENC=1
C=0001
B=0100
A=0011
ADDR=1111111100
FIR=0
ALU=0001
```

Simulación en curso

Figure 14-3. Graphical User Interface: Datapath status during program execution. At every moment, the involved microinstruction is displayed. In each sub-cycle, the functional units that execute the current micro-operation are highlighted.

5. RESULTS AND CONCLUSIONS

The results of the experiments are shown in table 14-1. They show that most of the students (85.8%) preferred using the Java applet, much easier to access and that does not require any additional effort to be run. Anyway, although only a few students chose the C# application (14.2%), all of them passed the tests. As both versions are identical, the main reason for this variation in the success rates may only reside in the initial auto-motivation of the students. At the beginning of the experience, those who chose the C# application knew that they should do an extra work in order to install a virtual machine that was not included in any operating system. Therefore, as they were interested in learning how to install *mono* and how to run a C# application, we can assume that they were also more motivated to learn the details of the architecture than most of the rest of their partners.

Table 14-1. Results of the experiments with students of first year in Computer Science. The qualifications belong to [0,100]

	SOFTWARE VERSION			
	Java Applet		C# Application	
Results of the evaluation	Students	%	Students	%
Programming skills				
≥75	15	7.9	14	7.4
[50,75)	92	48.4	13	6.8
<50	56	29.5	0	0.0
Sum	163	85.8	27	14.2
Functional units identification				
Correct	134	70.5	27	14.2
Incorrect	29	15.3	0	0.0
Sum	163	85.8	27	14.2
Ability for relating functional units work to instruction execution				
≥75	21	11.1	25	13.2
[50,75)	102	53.7	2	1.0
<50	40	21.0	0	0.0
Sum	163	85.8	27	14.2

In former years, the students have also studied the Simple-2 architecture by themselves, but without the help of any additional software, being the success rates of below 50%. From table 14-1, we can see that more than 70% of the students passed the tests (70.5% passed the programming test, 84.7% passed the functional units test, and 79.0% passed the instruction execution one). Hence, we can see that the success rates are quite bigger with the use of the simulators, and we can conclude that they improve the learning of the students with low costs. Furthermore, the analysis of the software version the students choose is useful for finding out the initially most auto-motivated ones and also for paying special attention to the others. Naturally, for the first semesters, simple computer architectures should be used in order the students to be able to identify their components and analyze their functionalities. In addition, the use of simulators increases the interest of those students with lower motivation, although those with higher initial motivation get the best results.

6. FUTURE WORK

Future work pretends to increase the possibilities of the simulator in order to allow the user to define new instructions. The graphical user interface is to be improved and also a language selection option is going to be included.

The experimental future work will try to develop an evaluation system that provide some measure of the rate students' efforts/learning when using software simulators and comparing them to those rates obtained without the simulators.

REFERENCES

Moure, J.C., Rexachs, D.I., and Luque, E., 2002, The Kscalar simulator, *ACM Journal of Educational Resources in Computing (JERIC)*, **2**(1)**:** 73-116.

Campos, A.M., García, D.F., Entrialgo, J., and Díaz, J.L., 2002, Simulador educacional de un computador elemental basado en la arquitectura Von Neumann, *Actas de las XXIII Jornadas de Paralelismo*, Lleida, 95-98.

Brorsson, M., 2002, MipsIt - a simulation and development environment using animation for computer architecture education, *Proceedings of the Workshop on Computer Architecture Education*. Anchorage, Alaska, 65-72.

Dormido, S., Canto, M.A., Mira, J., and Delgado, A.E., 2000, *Estructura y Tecnología de Computadores*, Sanz y Torres, Madrid.

Patterson, D.A., and Hennesy, J.L., 1997, *Computer Organization and Design: The Hardware/Software Interface*, 2nd ed., Morgan-Kaufmann, San Mateo, Ca.