SICOME 2.0: A teaching simulator for Computer Architecture

María Brox, Andrés Gersnoviez, Miguel A. Montijano, Ezequiel Herruzo, Carlos D. Moreno
Dept. Electronic and Computer Engineering
Escuela Politécnica Superior, Universidad de Córdoba
Córdoba, Spain
[mbrox, andresgm, el1movim, eze, cdiego]@uco.es

***Abstract*—As it is well known, teaching simulators are very useful resources to teach the practices of the subjects and that students understand in a more optimal way the theoretical concepts taught. Specifically, this work presents a teaching simulator, SICOME 2.0, which is used in the practices of Computer Architecture and allows an interactive simulation on a Simple Computer Architecture. The work also describes the practices carried out in the subject with this simulator. The experience with this simulator is very satisfactory and the results obtained show that it helps to improve the comprehension of the subject.**

Keywords—Teaching simulator, Computer Architecture practices, Simple Computer Architecture system

I. INTRODUCTION

The use of simulators in practices is a very useful tool that helps to improve the comprehension of the subjects [1]-[5]. Specifically, the teaching area of Computer Architecture at the University of Cordoba has developed a set of simulators that are used as resources in the teaching of subjects related to this area [6]-[8].

On the other hand, in Computer Architecture, a system based on a Simple Computer Architecture [9]-[10] is a tool widely used to explain fundamental aspects of the subject. Based on this system, a teaching simulator (SICOME 1.0) [8] was developed. However, this simulator presented a set of problems that made it difficult to develop the practices. Among the main disadvantages of this simulator were that in SICOME 1.0, the micro-programmed controller was the only one existing in the computer and besides, the bifurcation control logic (LCB) was already defined previously, so the user only had to analyze the truth table of the control logic to decide which would be the coding used in each case.

For this reason, it has been decided to correct these shortcomings presented in the previous simulator and a second version (SICOME 2.0) has been developed based on the Java architecture of Sun Microsystems that significantly improves the previous teaching environment. This new version can work by implementing a micro-programmed or hardwired unit control, allowing students to analyze the differences between creating an instruction set in one way or another. The truth table for the control logic in micro-programmed control is not previously defined as in the older version so that the user has to define it and introduce it in the computer through a graphical interface. This allows the development of a greater number of instructions and students acquire a better knowledge of the subject by creating their own control logic. This version also includes a greater number of microoperations compared to the previous version, which also facilitates the development of a greater number of instructions by performing them in a more optimal way.

The structure of the article is as follows. A detailed description of SICOME 2.0 is included in section II. Section III shows the practices of Computer Architecture developed with this simulator. A description of the experience of using this simulator in practices is included in Results section. Finally, the article ends by showing the conclusions of the work.

II. DESCRIPTION OF SICOME 2.0

The simulator consists of a graphical interface that includes a menu bar and tools, a Simple Computer Architecture scheme, a representation of the contents of the computer's memory, and a status console. An illustration of the graphical interface of SICOME 2.0 is shown in Fig. 1. The Simple Computer Architecture scheme shows the contents of the registers and memory, controller, arithmetic/logic unit and the available set of microoperations. As it is illustrated in Fig. 1 the arithmetic/logic unit control includes a QR register for the development of multiplication and division algorithms. The memory display consists of a series of fields indicating the addresses, the contents of these addresses and the instruction to which it is equivalent according to the instruction set that is loaded at any given time. Finally, the status console allows the user to know the state of the simulation.

It should be noted that the simulator can work by implementing a micro-programmed or hardwired control unit. The graphical representation of the micro-programmed controller is shown in Fig. 2. This figure also includes the flags that can be used in the control logic which it has to be introduced by the user as it will be described in the subsection III-B. As it is shown, the controller has a sequence counter register (SC) that allows making loops in the instructions. CROM is a memory whose output is the activation code of the computer terminals, associated LCB code and jump direction in the instruction or load value of the SC register (when it is applicable); CMAR is a counter register whose output is the input address of the CROM. Mapping logic converts the code provided by the OPR register into the address where the corresponding instruction begins in the CROM; the LCB,

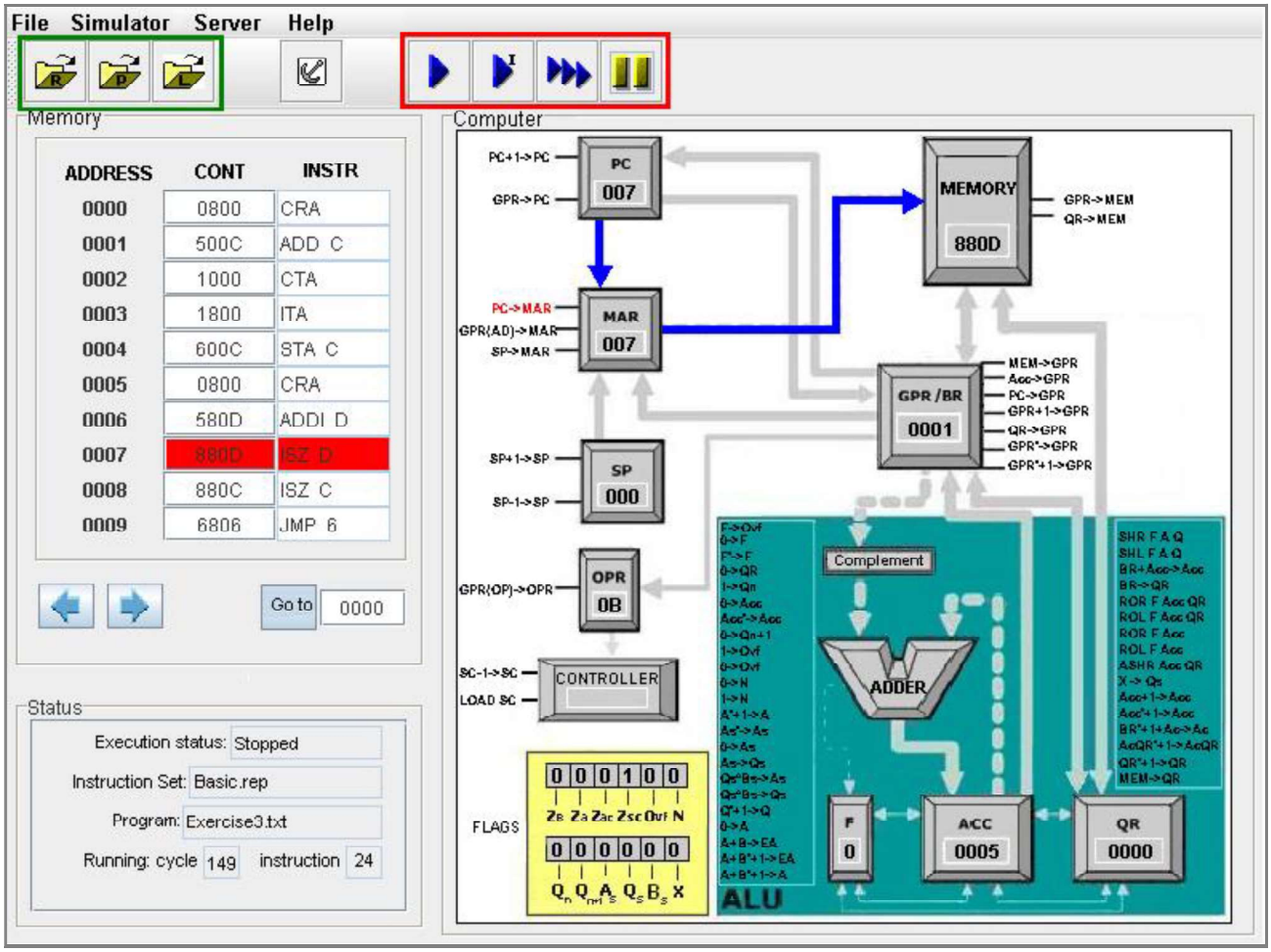

Fig. 1. SICOME graphical interface

according to the code provided by the CROM and flags, determines whether the CMAR should be increased (output I) or load the address provided by the CROM (output B) or that provided by the mapping logic (output R). The LCB also determines if the activation of computer terminals provided by the CROM is enabled (output E).

Fig. 3 shows the graphical representation of the hardwired control unit. It includes the OPR register and the internal counter of the controller. It also has, like the micro-programmed controller, a SC register. In the center of the figure, instruction and time decoders are illustrated, which together with the flags will activate some of the existing microoperations, once these signals go through the block that simulates the control of the logic gates. This last block represents the activation logic of the microoperations in hardwired control and it must be defined by the user, as it will be described in section III-C.

Other important feature of this simulator is that the datapath of an instruction through each stage is highlighted with different colors allowing to analyze signals and buses that are activated during the simulation as well as analyzing the content of the registers and memory. Finally, it should be noted that the simulator also includes tools that allow to carry out a continuous simulation, execute a complete instruction or to simulate cycle by cycle.

III. COMPUTER ARCHITECTURE PRACTICES USING SICOME 2.0

With SICOME 2.0, three practices of the subject Computer Architecture of second course of Computer Engineering at the University of Cordoba, are developed. The aim of this subject is to provide students with the basic knowledge of the different units that form a computer based on Von Neumann Structure/Architecture. In this subject the units of calculation, control, memory, and input/output of a computer are studied, taking as a reference the structure of a Simple Computer Architecture. Other practices of the subject are developed with OrCAD software and consist of designing a control unit using shift registers for the architecture of a Simple Calculator that is provided already designed to the students. With this software, students also design micro-programmed or wired control units for a Simple Computer Architecture studied in the subject.

The practices developed with SICOME 2.0 and their detailed development, are described below.

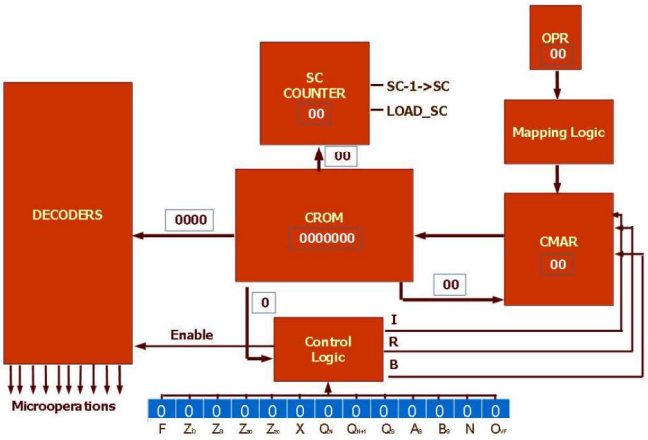

Fig. 2. Graphical representation of the micro-programmed controller

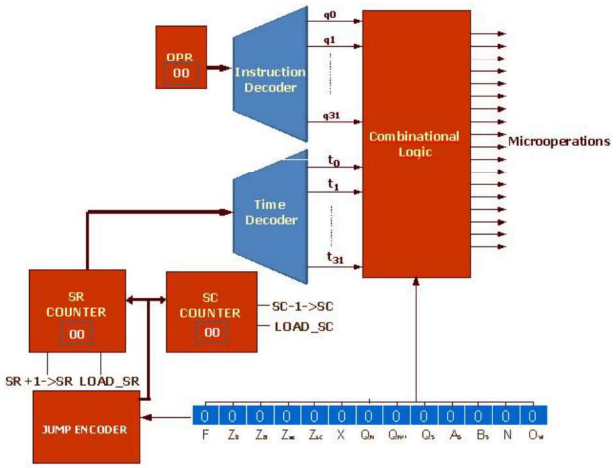

Fig. 3. Graphical representation of the hardwired control unit

A. Introduction to SICOME 2.0

The first practice is dedicated to being an introduction to the simulator where students develop three test programs and simulate them using an instruction set and a control logic provided. The instruction set provided includes basic instructions of addition, storage, jump, clear, rotation, negation, increment, subroutine calling, and execution stop. With this basic instruction set three programs have to be carried out. The first one performs a program that calculates the subtraction of two numbers stored in memory, storing the result in a memory location. The subtraction operation must be implemented as the sum of the minuend and the two's complement of the subtrahend. The second exercise is based on a program that calculates the sum of the absolute values of two numbers stored in memory. The result of the addition must also be stored in a memory location. In this program, students must perform a subroutine that calculates the absolute value of a number by making several calls to this subroutine. Finally the third exercise carries out a program that calculates the sum of a set of numbers stored in a memory table and stores the result in the accumulator register. The table size is known and indicated in a memory location. For this exercise students must perform a loop that does the addition operation the same number of times that elements to add are included in the table. The code of the third programm is shown in Fig. 4 where the code structure of the programs in SICOME is illustrated. A program includes three different sections. Each section is separated from the next section by the character "@". The first section is for the area of declaration of variables where it is necessary to specify its storage address in memory followed by its value, both in hexadecimal code. In the example shown in Fig. 4, memory location C includes the number of elements to be added, while the next memory location indicates the position of the first element to be added; this position will be increased to add the rest of the numbers in the table whose values have been stored in memory location E to address 12. The next section indicates the starting memory address of the program. Finally, the code includes the program instructions. Programs can also include comments that must be preceded by the "#" character and must be included in their own line so that a same line of code cannot contain a program instruction and comment. Instruction set, program and control logic are text files that are loaded into SICOME using the buttons that include "R", "P" and "L" symbols in the menu bar and that have been highlighted with a green rectangle in Fig. 1. Once the program is loaded, it is included in the SICOME memory display as it is shown in Fig. 1. The simulation tools are highlighted in a red rectangle and allow to simulate cycle by cycle, execute a complete instruction, perform a continuous simulation or stop the execution of the program. Using these tools, the program is simulated and the result of the sum of the five numbers included in the table is stored in the

```
#Area of declaration of variables
C 5
D E
E  1
F  1
10 1
11 1
12 1
@
#Starting memory address of the programm
0
@
CRA
ADD C
CTA
ITA
STA C
CRA
ADDI D
ISZ D
ISZ C
JMP 6
STA 13
HALT
@
```

Fig. 4. Structure of a program in SICOME

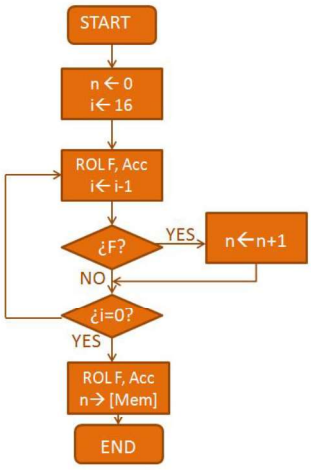

Fig. 5. Flow diagram of ONES instruction

Accumulator register, as it is shown in Fig. 1. Once the program execution is finished, the simulator also allows to clear memory to load a new program, restart the computer to run a new simulation, or reset the computer to introduce a new instruction set, program and control logic.

B. Design of an instruction set in a micro-programmed control unit

In this second practice, students design an instruction set for a micro-programmed control unit. They also have to develop the control logic and carry out test programs, such as those proposed in practice 1, to check that the instruction set has been developed correctly. The proposed instruction set is modified each academic course. An example of an instruction set is the following:

- FECTH
- HALT
- LDA m: It loads the contents of m memory location into the Accumulator register
- ONES m: It counts the number of ones in the Accumulator register and stores it in m memory position

Table I. RTL table of ONES instruction in a micro-programmed unit control

CYCLE	MICROOPERATIONS	NEXT
ADDR(ONES)+0	0→QR; 16→SC	Increase
ADDR(ONES)+1	QR→GPR	Increase
ADDR(ONES)+2	ROL F, Acc; SC-1→SC	Increase
ADDR(ONES)+3	GPR+1→GPR (if F=1)	If Zsc=0 jump to ADDR(ONES)+2 If Zsc=1 increase
ADDR(ONES)+4	ROL F, Acc; GPR→M	Jump to ADDR(FETCH)

The flow diagram of ONES instruction is shown in Fig. 5. In order to count the number of ones, a loop of 16 iterations is performed (because SICOME registers have a size of 16 bits) where rotations to the left of the Accumulator are implemented, so that the most significant bit of this register is shifted to F register. Once the rotation has been carried out, if the value of F is one, n variable, that stores the number of ones in the Accumulator register, is increased. For each one of the instructions, students have to develop an RTL table as shown in Table I for ONES instruction. In this Table the flow diagram of Fig. 5 is described with computer microoperations. To make the loop, the SC register has been used which has a load microoperation of a value (in this case it is loaded with value 16) and a decrement microoperation which is used to decrease the value of the register in a unit each time an iteration is performed. There is a flag bit on the computer, Z_{sc}, which is one if SC register is zero. Each time an iteration of the loop is performed, the value of Z_{sc} is analyzed to know if the loop has finished or there is a new iteration, jumping in this case to the starting direction of the loop. The general purpose register GPR is used to store the number of ones of the accumulator register.

In parallel to the construction of RTL table, students develop the truth table of control logic (Table II). This table includes four control bits (B_3, B_2, B_1 and B_0 bits) and flags bits (F, Zb, Za, Zac...) that can take the values of 0, 1 and X indicating the indifference of the selected flag. For each combination of control bits, the values of the different outputs available from CMAR (I, B, R and E) are indicated. For this example, the four control fields common to all instruction sets are included. There is a combination for the case in which the

Table II. Truth table of control logic

B_3	B_2	B_1	B_0	F	Z_b	Z_a	Z_{ac}	Z_{sc}	X	Q_n	Q_{n+1}	A_s	Q_s	B_s	N	I	B	R	E
0	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0
0	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	1	0	0	1
0	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	0	1	0	1
0	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	0	0	1	1
0	1	0	0	0	X	X	X	0	X	X	X	X	X	X	X	0	1	0	0
0	1	0	0	0	X	X	X	1	X	X	X	X	X	X	X	1	0	0	0
0	1	0	0	1	X	X	X	0	X	X	X	X	X	X	X	0	1	0	1
0	1	0	0	1	X	X	X	1	X	X	X	X	X	X	X	1	0	0	1

Table III. CROM Table

Cycle	MAR		OPR AND MEM		SP, PC AND SC			ALU						GPR			CONTROL LOGIC				DIRECTIONS AND LOAD OF SC COUNTER								HEX CODE	
	S_{12}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0	B_3	B_2	B_1	B_0	M-	M_6	M_5	M_4	M_3	M_2	M_1	M_0		
FETCH																														
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	PC→MAR	4000100
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	M→GPR; PC+1→PC	0201100
2	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	GPR(OP)→OPR; GPR(AD)→MAR	B000300
HALT																														
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0000000
LDA																														
4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0→Acc; M→GPR	0009100
5	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	GPR+Acc→Acc	0028200
					LOAD SC			0→QR						ONES							16									
6	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0→QR; 16(Dec)->SC	0A40110
7	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	QR→GPR	0005100
8	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	ROL F, Acc; SC-1→SC	0C30100
9	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	GPR+1→GPR (if F=1)	0004408
A	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	ROL F, Acc; GPR→M	1030200
	1		0		3			0						2			0				0									

CMAR is increased (output I=1); the specific field for when the address provided by the CROM is loaded (B=1); the case associated when the address provided by the mapping logic (R=1) is loaded; and finally, a specific combination for stopping the execution of the program where the four outputs of CMAR have zero value. For the particular case of this instruction set, a combination of control bits has been designed that uses F and Zsc flags simultaneously. If F bit is one, E output is activated to enable the increment of GPR register because a bit of Accumulator with value one has been detected; if F is zero, this increment will not be enabled. On the other hand, Zsc controls that the loop is finished, activating I output to exit of the loop or jumping to the start direction of the loop. This truth table of the control logic can be included in a text file and loaded in SICOME but the user can also specify it using a graphical interface included in the simulator.

Finally, students must store the microprograms corresponding to the instruction set in the CROM memory. The microwords stored by the CROM have a length of 28 bits, of which the first 16 are used as encoding signals for microoperations, the next 4 as control signals, and the last 8 as addressing signals or load value of SC counter, as shown in Table III. This table includes the coding of fetch cycle and execution cycle for HALT, LDA and ONES instructions using microwords. The combinations of bits to code the different microoperations (bits s15 to s0) can be found in SICOME Help. Once the microwords have been completed, a 7-digit hexadecimal coding is obtained and included in the instruction set file.

```
$
CB 4000100
CB 201100
CB B000300
$
HALT false 0
LDA true 0009100 0028200
ONES true 0A40110 0005100 0C30100 0004408 1030200
```

Fig. 6. Structure of an instruction set using micro-programmed control in SICOME

The instruction set file using micro-programmed control is a text file in which the name of the instruction is defined, as well as its corresponding microprogram. In each line of the file, an instruction is declared with the following fields separated by spaces: the name of the instruction; a flag that indicates if the instruction has a parameter or not; and finally the microprogram of the instruction composed of the sequence of hex-coded microwords that was obtained in Table III. For the example included in this description, the instruction set file is illustrated in Fig. 6. To check that the instruction set and control logic developed work well, students perform a small test as shown in Fig. 7 where the Accumulator register with the contents of 0 memory position is loaded. The ONES instruction is then used to count the number of ones of the Accumulator register and store this count in 1 memory position. The instruction set, program and control logic files are loaded in SICOME using "R","P" and "L" buttons and then a simulation can be performed. At the end of the simulation, it is checked if the count stored in 1 position coincides with the number of ones of Accumulator register to detect if there has been any error in the development of the instruction set or control logic.

```
0 1C75
@
10
@
LDA 0
ONES 1
HALT
```

Fig. 7. Test program

Table IV. RTL table of ONES instruction in a hardwired unit control

CONDITIONS	MICROOPERATIONS	NEXT
FETCH		
t_0	PC→MAR	SR+1→SR
t_1	PC+1→PC; M→GPR	SR+1→SR
t_2	GPR(OP)→OPR; GPR(AD)→MAR	SR+1→SR
LDA		
$q_1.t_3$	0→Acc; M→GPR	SR+1→SR
$q_1.t_4$	GPR+Acc→Acc	0→SR
ONES		
$q_2.t_3$	0→QR; 16→SC	SR+1→SR
$q_2.t_4$	QR→GPR	SR+1→SR
$q_2.t_5$	ROL F, Acc; SC-1→SC	SR+1→SR
$q_2.t_6.F$	GPR+1→GPR	
$q_2.t_6.Z_{SC}$		SR+1→SR
$q_2.t_6.Z'_{SC}$		5→SR
$q_2.t_7$	ROL F, Acc; GPR→M	0→SR

C. *Design of an instruction set in a hardwired control unit*

In the third practice the students develop the same instruction set of the previous practice but using hardwired control unit. Similarly to the previous practice, the RTL table of each instruction must be designed. The RTL table of ONES instruction using hardwired control is shown in Table IV.

Once this table has been finished, students can obtain the control expressions that activate each one of the microoperations and that for this particular case are shown in Table V. These control expressions can be included in the simulator using a text file whose format is shown in Fig. 8. As can be seen in the figure, the text starts with the special $ symbol and places each microoperation in a different line. The name of each microoperation is followed by all expressions that are part of the logic function of that microoperation. The user can use this text file and load it into the simulator using "L" button or SICOME also has a graphical interface that allows the user to define the logic of microoperations activation.

Table V. Control Expressions

MICROOPERATION	CONTROL EXPRESSIONS
GENERIC OPERATIONS	
PC→MAR	t_0
PC+1→PC	t_1
M→GPR	$t_1+q_1.t_3$
GPR(OP)→OPR	t_2
GPR(AD)→MAR	t_2
0→Acc	$q_1.t_3$
GPR+Acc→Acc	$q_1.t_4$
0→QR	$q_2.t_3$
QR→GPR	$q_2.t_4$
ROL F, Acc	$q_2.t_5+q_2.t_7$
GPR+1→GPR	$q_2.t_6.F$
GPR→M	$q_2.t_7$
SR OPERATIONS	
SR+1→SR	$t_0+t_1+ t_2+ q_1.t_3+ q_2.t_3+ q_2.t_4+ q_2.t_5+ q_2.t_6.Z_{SC}$
LOAD SR	$q_1.t_4(0)+ q_2.t_6.Z'_{SC}(5)+q_2.t_7(0)$
SC OPERATIONS	
SC-1→SC	$q_2.t_5$
LOAD SC	$q_2.t_3(16)$

```
$
PC->MAR: t0
PC+1->PC: t1
M->GPR: t1 + t3·q1
GPR(OP)->OPR: t2
GPR(AD)->MAR: t2
0->ACC: t3·q1
GPR+ACC->ACC: t4·q1
0->QR: t3·q2
QR->GPR: t4·q2
ROL FA: t5·q2 + t7·q2
GPR+1->GPR: t6·q2·F
GPR->M: t7·q2
$
SR+1->SR: t0 + t1 + t2 + t3·q1 + t3·q2 + t4·q2 + t5·q2 +
t6·q2·Zsc
LOAD SR: t4·q1-0 + t6·q2·Zsc'-5 + t7·q2-0
SC-1->SC: t5·q2
LOAD SC: t3·q2-16
$
```

Fig. 8. Text file of control expressions in SICOME

The instruction set file in this case is shown in Fig. 9.

```
HALT false 0 q0
LDA true 2 q1
ONES true 5 q2
```

Fig. 9. Structure of an instruction set using hardwired control in SICOME

Each line of the file declares an instruction that consists of the following fields separated by spaces: name of the instruction; flag that indicates whether the instruction has a parameter or not; the number of cycles of the instruction; and the identifier of the instruction where the letter q is used followed by an index that must match with the order in which the instruction has been defined in the directory. By loading the set instruction, the activation logic and using the same test program shown in Fig. 7 a simulation can be performed to verify that the instruction set is correct.

IV. RESULTS

The use of SICOME 2.0 in Computer Architecture practices has allowed to verify that students improve the theoretical knowledge learned because the simulator allows them to analyze signals and buses that are activated during the simulation as well as analyzing the content of the registers and memory.

Another aspect that is very important to understand the structure of a Simple Computer Architecture is that the simulator allows them to carry out a continuous simulation, execute a complete instruction or to simulate cycle by cycle, so that they can easily detect errors in the design of the programs and instruction set developed.

On the other hand, the structure of the practices also allows them to analyze differences by creating a same instruction set on a micro-programmed and hardwired control unit.

The evaluation of the practices is continuous so that the attendance and attitude of the students in the development of them is evaluated in the final qualification. The students must explain to the teachers each of the proposed practices showing that the practices are correct and have understood the objectives of each one of them and the results obtained. For the first practice only one session is used, while for the second and third, two sessions are necessary. There are also more tutoring sessions so that students can finish the practices, ask any questions and defend them in case they have finished them. The simulator software is also provided to students for installation on their personal computers and they can work with it outside the laboratory, analyzing the functioning of some complex instructions that are studied in theoretical classes or problems. Following this evaluation method, 97% of the students approved the practical part of the subject during the last academic year.

V. CONCLUSIONS

A new version of a simulator for the teaching of Computer Architecture has been presented. The great functionality offered by the simulator ensures that SICOME 2.0 is a high quality teaching tool. The students and teachers of this subject are very satisfied with the great advance in the learning of students using the simulator in the practices, and consider that SICOME 2.0 allows them to reach a high level of comprehension in the subject.

REFERENCES

[1] Contreras, G.A., García, R., Ramírez, M.S., "Uso de simuladores como recurso digital para la transferencia del conocimiento", Revista de Innovación Educativa, vol.2, no.1, pp.86-100, 2010

[2] Nikolic, B., Radivojevi, Z., Djordjevi, J., Milutinovic, V., "A Survey and Evaluation of Simulators Suitable for Teaching Courses in Computer Architecture and Organization", IEEE Transactions on Education, vol.52, no.4, 2009

[3] Yehezke, C., Yurzik, W., Pearson, M., Armstrong, D., "Three simulator tools for teaching computer architecture: Easy CPU, Little Man Computer, and RTLSim", ACM Journal Educational Resources in Computing, vol.1, no.4, pp. 60-80, 2001

[4] Aguilar, I., Heredia, J.R., "Simuladores y laboratorios virtuales para Ingeniería en Computación", 2º Congreso Virtual sobre Tecnología, Educación y Sociedad (CTES), Mexico, 2013

[5] Grossi, M.D., Jiménez Rey, E., Servetto, A., Perichinsky, G., "Un simulador de una máquina computadora como herramienta para la enseñanza de la arquitectura de computadoras", I Jornadas de Educación en Informática y TICs en Argentina, 2005

[6] Benavides, J.I., Herruzo, E., Paloamres, J.M., "Herramientas docentes basadas en Internet para la docencia de Arquitectura de Computadores", Congreso Uruguayo de Informática y CLEI (infoUYclei 2002), Montevideo, 2002

[7] Herruzo, E., Benavides, J.I., Saez, E., Montijano. M.A., Paloamres, J.M., "Desarrollo de simuladores de Arquitectura de Computadores y su aplicación en la enseñanza", Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica (TAEE 2002), Las Palmas de Gran Canaria, 2002

[8] Conde, S., Muñoz, A., Herruzo, E., Benavides, J.I., "Aplicación interactiva de simulación del comportamiento de una Computadora sencilla con fines docentes", IX Congreso Universitario de Innovación Educativa en las Enseñanzas Técnicas, Vigo, 2001

[9] Taub, H., 'Digital Circuits and Microprocessors', McGraw-Hill, 1983

[10] Morris Mano, M., 'Computer Architecture', Ed. Prentice-Hall S.A., 1983