Edu.LMC: a new LMC simulation approach using LMC Paradigm

Isabel Pedrosa
ISCAC
Quinta Agrícola, Bencanta
Coimbra
+351.239.802000
ipedrosa@iscac.pt

António José Mendes
DEI - FCTUC
Polo II, Pinhal de Marrocos
Coimbra
+351.239.790000
toze@dei.uc.pt

Mário Zenha Rela
DEI - FCTUC
Polo II, Pinhal de Marrocos
Coimbra
+351.239.790000
mzrela@dei.uc.pt

ABSTRACT
Little Man Computer (LMC) Paradigm, a simple representation of a real computer system with pedagogical purposes, was firstly presented in 1965. It follows von Neumann's Architectural Model and uses a simplified instruction set. Since then, many simulators have been developed based on this Paradigm. Mainly, those simulators have been designed to be used by undergraduate students in Computer Architecture courses. We developed a new LMC simulator – edu.LMC –created especially for students with very basic skills on Computer Architecture, mostly not majors on Computer Science or Computer Engineering. This is a special purpose simulator with a clear pedagogical focus, tested and used during Computer Architecture classes for a Management and Informatics course. edu.LMC includes many features that are difficult to find together in other LMC simulators. Some existing simulators are more complete, but also too complex to be used in this learning context. In this paper, we present edu.LMC's main characteristics, some utilization examples and preliminary evaluation results.

Categories and Subject Descriptors
K.3.[**Computers and Education**]: K3.1 Computer Uses in Education and K.3.2 Computer and Information Science Education

General Terms
Algorithms , Languages.

Keywords
Computer Architecture, Little Man Computer, Simulation.

1. INTRODUCTION
Simulation and Computer Architecture (CA) Education are very often found together and results from this partnership are referred by [8] as making possible for "students to learn the fundamentals of computer organization/architecture by visually observing and interacting with animated data flow within a particular or real machine" and "using animated resources". Such partnership can be found on the many repositories of educational resources on Computer Architecture Simulators, namely [6] and [7].

However, for Management and Informatics (MI) students' learning context, we were looking for CA Simulators that could be used during classes (or as a tool for students' autonomous work, outside classes) but with no need of previous significant skills on CA. One of the possibilities was Little Man Computer, LMC, based on LMC Paradigm. LMC Simulators are simple instruction level simulators and very simple single accumulator based architectures with a small number of instructions, suitable for a first course in Computer Science.

LMC simulators allow students to write and test their own programs using a very simple instruction set. Furthermore, some authors, [5], consider that "LMC Simulators are also important for students to understand Operating Systems (OS) concepts, to understand that Reset is similar to bootstrap process of a real computer accessing a predetermined address in ROM to load the OS kernel". We tested some of those LMC simulators: Son-of-LMC [1] and three similar approaches (Friend of Son of Little Man Computer, FoSoLMC, available at [11], Acquaintance of Friend of Son of Little Man Computer, AoFoSoLMC, presented at [12] and LMC Clone accessible form [13]), Shockwave simulator [1], [4] available at [14], Interactive web-based LMC Simulator - Illinois State University, referred in [1], [3], [4], [5], [9], [10] and available at [3]. After these tests we detected some points that could be improved with a new simulator:

- **Different instruction set**: most LMC simulators use different instruction sets when compared with the original LMC Paradigm.
- **Difficulty in program writing**: many LMC simulators require programs to be written using 3 digits codes instead of mnemonics and don't allow to import programs in text files neither to save programs or reopen previously saved files.
- **Weak input/output support**: there are LMC simulators that don't make very clear how inputs/outputs are treated.
- **No print option**: many LMC simulators don't allow printing.
- **No help or example files**: mostly there aren't elements to help learning LMC "language" built into the simulators.

We have also collected some good practices that we consider important to preserve in our new LMC Simulator, Educative LMC, edu.LMC. For its design we focused on developing a pedagogically adequate LMC Simulator to our main target public (MI undergraduate students) and to include functionalities capable of making edu.LMC an acceptable tool to students' autonomous learning sessions.

2. LMC PARADIGM

LMC Paradigm has been presented in 1965 by Stuart Madnick and John Donnovan. Irv Englander in his textbook [2] has adopted and developed this Paradigm to explain its similarities with a real computer system. Components of LMC Paradigm have a direct correspondence with elements of John von Neumann's Architectural Model as shown in Figure 1 and Table I. Many LMC simulators have been developed since then some of them are currently used in many universities at undergraduate Computer Architecture courses.

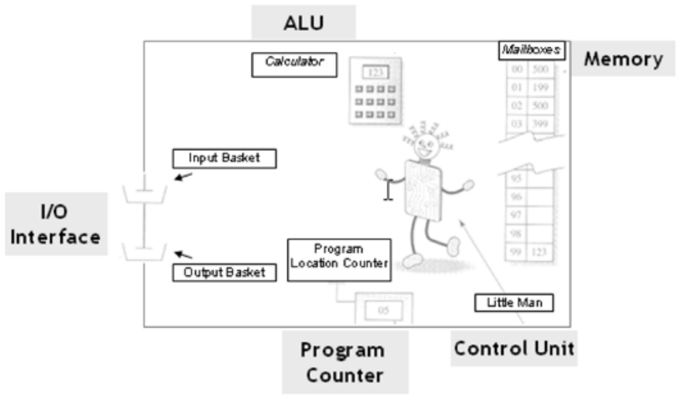


Figure 1: the main components of *Little Man Computer Paradigm* (grey colour) versus John von Neumann's architectural model (white rectangle) [adapted from Irv Englander]

As a conceptual approach, LMC Paradigm represents easy concepts and is a very simple way to understand the functional areas of a real computer system: ALU (Arithmetic and Logical Unit), Control Unit, Memory, Program Counter and Input/Output Areas. LMC main area is a mailroom whose fundamental components are: 100 mailboxes numbered with 2 digits from 00 to 99 (Memory), a calculator (ALU), an instruction location counter (Program Counter), input and output baskets (I/O) and a Little Man (Control Unit) that supervises program execution.

Table 1. Table captions should be placed above the table

LMC Paradigm Component	Description	John von Neumann Architectural Model
Mailboxes	Stores decimal coded numbers: program instructions, user numbers or operations results	Memory
Calculator	Stores (temporarily) input values or output results, does simple arithmetic addictions and subtractions	ALU, Arithmetic and Logical Unit
Input / Output Baskets	Recipients to communicate with the outside of the mailroom	Input/Output
Instruction Location Counter	Identifies the instruction being executed at a specific moment	Program Counter
Little Man	Little Man	Control Unit
Little Man steps		Bus Connections

LMC Paradigm uses an instruction set based on ten different instructions. With these simple instructions, presented on Table II, it is possible to write programs with 3 digits decimal-encoded instructions or mnemonics (if we want to use an easier and simpler approach). It can represent an introduction to Assembly Language, a usual topic in Computer Architecture courses. LDA, STO, ADD, SUB, IN, OUT, COB/HLT are part of the basic LMC instruction set. They can be used to design simple programs with linear sequence between the instructions. LMC extended instruction set, including BR, BRZ and BRP, makes possible to write programs with decisions based on calculator contents.

Table 2. the LMC Paradigm's instruction set

Category	Mnemonics	3 Digits Code	Description (pseudo-coded)	Ins. set
Data Movement	LDA	5XX	Calculator← [mailbox XX]	Basic
	STO	3XX	[mailbox XX] ←Calculator	Basic
Arithmetic operations	ADD	1XX	Calculator← Calculator+ [mailbox XX]	Basic
	SUB	2XX	Calculator← Calculator– [mailbox XX]	Basic
Input/ Output	IN	901	Calculator← input value	Basic
	OUT	902	Output value← Calculator	Basic
Stop	COB/HLT	000	Stops the program	Basic
Branches	BR	6XX	Jumps to mailbox XX	Extended
	BRZ	7XX	If Calculator=0 Then Jumps to mailbox XX	Extended
	BRP	8XX	If Calculator>=0 Then Jumps to mailbox XX	Extended

3. EDU.LMC SIMULATOR

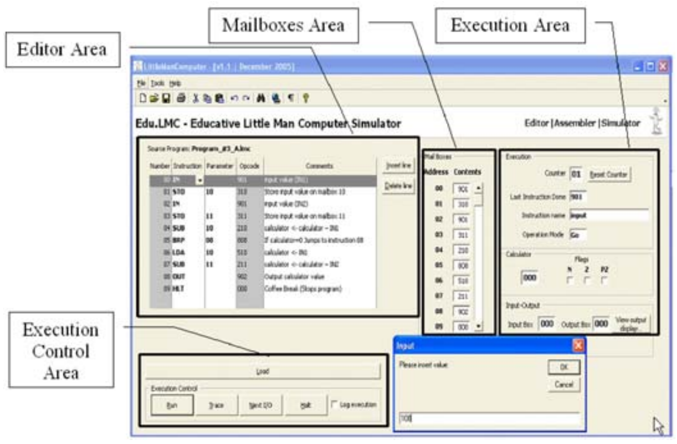
We have developed a new LMC simulator – edu.LMC. It consists on a Win32 application, created specifically for students with very basic skills on Computer Architecture, mostly not majors on Computer Science or Computer Engineering. This is a special purpose simulator, with a clear pedagogical focus, tested and used during Computer Architecture classes for our MI course. edu.LMC uses instruction mnemonics based on LMC Paradigm, allows instruction commentaries and verifies each instruction syntax before the simulation starts. Students can create, run, debug, print and save programs. This simulator includes an examples database and a help area describing each instruction. We aimed to create a simulator that represents a close approach to LMC Paradigm, including all functionalities and features that are difficult to find together in other simulators. Additionally, edu.LMC is very user friendly because our main target public is undergraduate students on MI.

3.1 The application

The application window is presented at Figure 2. edu.LMC simulator is divided into four main areas:

- **Editor Area**: for writing programs using mnemonics. It is possible to write instructions directly or to open programs saved in text files with .lmc extension (if they follow a predefined structure, as shown in Figure 3). Programs are verified syntactically as the student writes each instruction and the operation code (opcode) is generated for each instruction. Instruction mnemonics are picked from a listbox directly linked to an instruction database. User can't proceed if neither the instruction nor parameters are wrong. There are also locked areas - the ones in grey: instruction number and opcode. Operations like add/delete lmc program lines are possible at any program line. Load operation puts the corresponding opcodes in mailboxes. edu.LMC saves programs, presents the current program name, creates new programs even when another is open (in which case it clears the work area), allows printing the program and all commentaries and verifies all syntax errors or inputs that can not be supported.
- **Mailboxes Area**: shows the mailboxes and their 3 decimal coded contents. The first ones are program instructions and, usually, the last ones are values (data) stored by programs.
- **Execution Area**: other functional LMC components like calculator, counter (for instruction location counter), last instruction executed (important if there are jumps), instruction mnemonic, operation mode, flags, input and output boxes, an option to display the output (if the output sequence is important).
- **Execution Control Area**: 4 modes to execute LMC programs, Run (execute from 00 instruction until HALT), Trace (Step by Step), Next I/O (execution stops only for I/O values) and Halt (stops program running).

Edu.LMC includes many functionalities that can be pedagogically interesting in this learning context: create, open, comment, save, verify, execute and print programs. Besides that, it includes examples organized by level of difficulty, and a comprehensive Help Area with executable examples for each instruction. Users can also view the saved "Log Execution" file, receive detailed information about Program Tracing and check or change the Mnemonics Conversion Table.



Source Program: Program_#3_A.lmc

Number	Instruction	Parameter	Opcode	Comments
00	IN		901	Input value (IN1)
01	STO	10	310	Store input value on mailbox 10
02	IN		901	input value (IN2)
03	STO	11	311	Store input value on mailbox 11
04	SUB	10	210	calculator <- calculator - IN1
05	BRP	08	808	If calculator>=0 Jumps to instruction 08
06	LDA	10	510	calculator <- IN1
07	SUB	11	211	calculator <- calculator - IN2
08	OUT		902	Output calculator value
09	HLT		000	Coffee Break (Stops program)

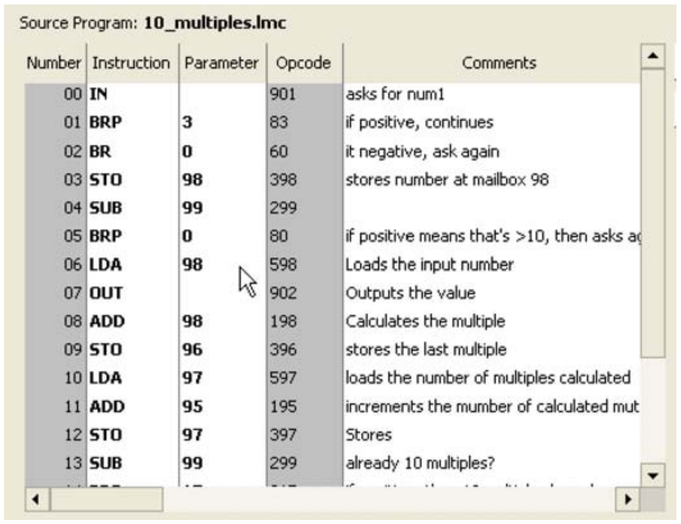
Figure 2. *edu.LMC* aplication window

```
00 IN;asks for num1
01 BRP 03;if positive, continues
02 BR 00; if negative, asks again
03 STO 98; stores number at mailbox 98
```

Figure 3. *The edu.LMC* text file Format

3.2 Main Functionalities

- **Examples Database**: is an important functionality to help students learning lmc programming. There are ten example programs available directly from LMC\Programs after edu.LMC installation. Those programs have distinct difficulty levels and its level is indicated as a comment at the beginning of lmc file. All files have comments on each instruction. Those examples can be useful to help novice users starting to understand how lmc programming and execution work. For instance, example on Figure 4 calculates 10 multiples of an input number. Those multiples are between two bounds: mailbox 99 (lower) and mailbox 98 (upper).

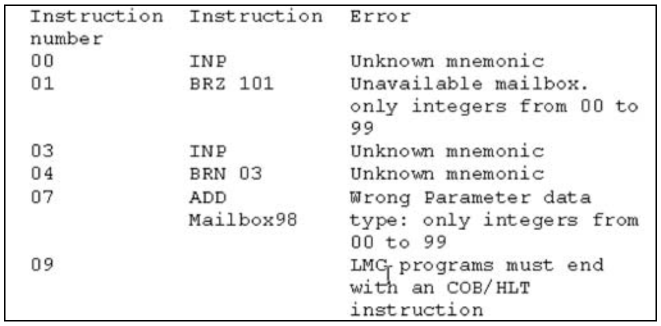


Source Program: **10_multiples.lmc**

Number	Instruction	Parameter	Opcode	Comments
00	IN		901	asks for num1
01	BRP	3	83	if positive, continues
02	BR	0	60	it negative, ask again
03	STO	98	398	stores number at mailbox 98
04	SUB	99	299	
05	BRP	0	80	if positive means that's >10, then asks ag
06	LDA	98	598	Loads the input number
07	OUT		902	Outputs the value
08	ADD	98	198	Calculates the multiple
09	STO	96	396	stores the last multiple
10	LDA	97	597	loads the number of multiples calculated
11	ADD	95	195	increments the mumber of calculated mut
12	STO	97	397	Stores
13	SUB	99	299	already 10 multiples?

Figure 4. *edu.LMC* Editor Area with a program File (10_multiples.lmc) to calculate the 10 multiples of a number

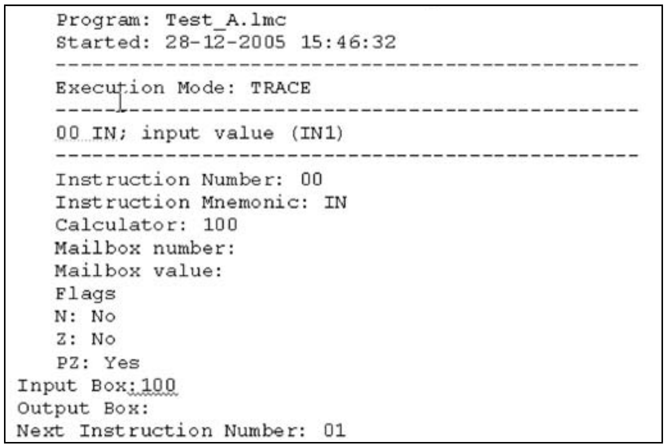
- **Program correction**: Programs written, using edu.LMC, are verified on each instruction at a time. Instruction mnemonics are restricted to instruction set available trough a listbox and parameter correction is immediate – only numbers between 00 and 99 are accepted. But, like users can also create their lmc programs on a text editor, when an lmc program is opened at edu.LMC, lmc program is placed at Editor Area and a list of errors, Error Detection Window, is generated if any errors are detected: wrong instruction numbers, incorrect mnemonics or parameters, misplaced comments. Students can then correct those errors. If they don't, the Load operation won't proceed and another (or the same) Error Window will be present. Figure 5 has an Example of errors detected – it includes a correction to a missing Coffee Break (COB) instruction to end the program - and solutions to correct it.



Instruction number	Instruction	Error
00	INP	Unknown mnemonic
01	BRZ 101	Unavailable mailbox. only integers from 00 to 99
03	INP	Unknown mnemonic
04	BRN 03	Unknown mnemonic
07	ADD Mailbox98	Wrong Parameter data type: only integers from 00 to 99
09		LMC programs must end with an COB/HLT instruction

Figure 5. Error Detection example s and suggested solutions to correct LMC Program

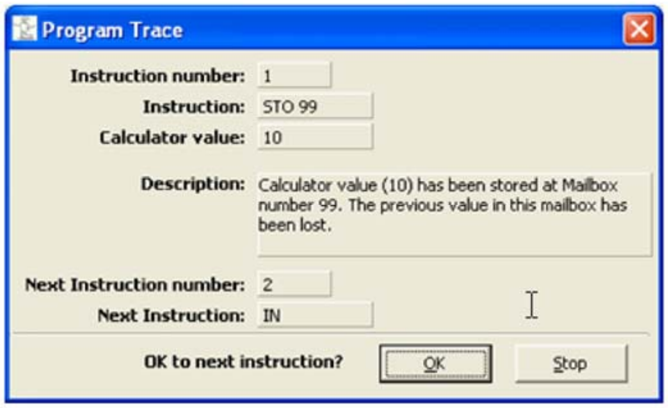
- **Log Execution**: usually, students find difficult to understand the way variables' values change. It is possible to activate the "Log Execution" option (available on Execution Control Area) and generate a text file with information about Program Execution. The default name is LMC_log.txt but it can be changed. At Figure 6 we have a sample, only considering instruction 00 for program Test_A.lmc. The log file can't be directly used from edu.LMC but it can be opened with any Text Editor.



```
   Program: Test_A.lmc
   Started: 28-12-2005 15:46:32
   ---------------------------------------------
   Execution Mode: TRACE
   ---------------------------------------------
   00 IN; input value (IN1)
   ---------------------------------------------
   Instruction Number: 00
   Instruction Mnemonic: IN
   Calculator: 100
   Mailbox number:
   Mailbox value:
   Flags
   N: No
   Z: No
   PZ: Yes
Input Box:100
Output Box:
Next Instruction Number: 01
```

Figure 6. Example for Log Execution File

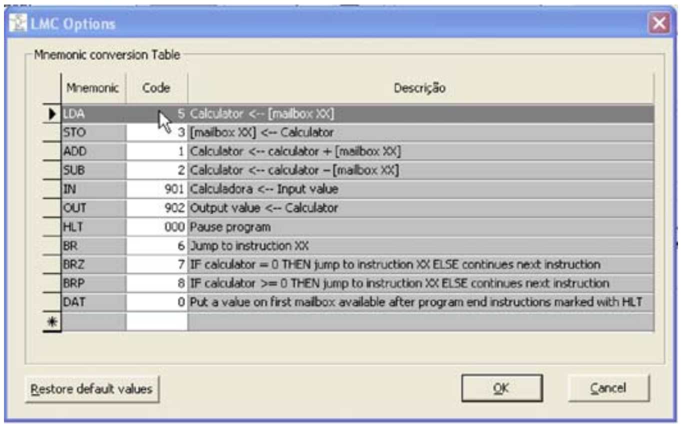
- **Program Tracing Information**: This is a complement for common Tracing Program Execution. If we chose, on Execution Control Area, the Trace option, a dialog box, like the one at Figure 7, is displayed with information about the current and the next instruction to be executed. It asks also if we want to step to the next instruction.



Instruction number: 1
Instruction: STO 99
Calculator value: 10
Description: Calculator value (10) has been stored at Mailbox number 99. The previous value in this mailbox has been lost.
Next Instruction number: 2
Next Instruction: IN

Figure 7. Program Trace Window

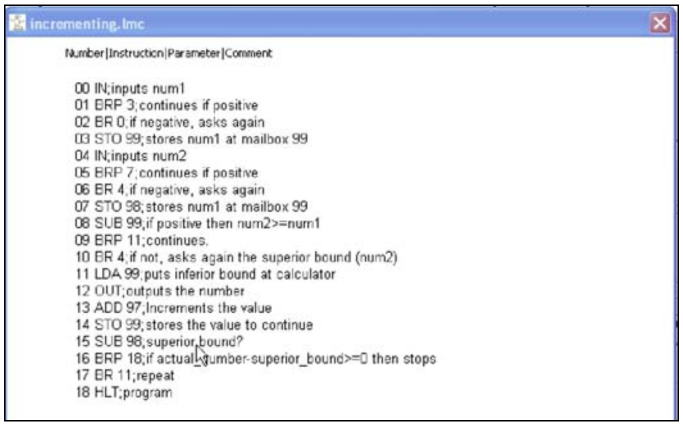
- **Mnemonic Conversion Table**: many LMC Simulators use different instruction codes when comparing Simulator implementation with LMC Paradigm. Table on Figure 8 make it possible to configure those codes or to restore them to their (original) default values. This Mnemonic Conversion Table is applied to correct lmc programs in order to check if the LMC instruction exists.



Mnemonic	Code	Descrição
LDA	5	Calculator <-- [mailbox XX]
STO	3	[mailbox XX] <-- Calculator
ADD	1	Calculator <-- calculator + [mailbox XX]
SUB	2	Calculator <-- calculator – [mailbox XX]
IN	901	Calculadora <-- Input value
OUT	902	Output value <-- Calculator
HLT	000	Pause program
BR	6	Jump to instruction XX
BRZ	7	IF calculator = 0 THEN jump to instruction XX ELSE continues next instruction
BRP	8	IF calculator >= 0 THEN jump to instruction XX ELSE continues next instruction
DAT	0	Put a value on first mailbox available after program end instructions marked with HLT

Figure 8. *edu.LMC* Mnemonic Conversion Table

- **View Output Display**: if students implement a LMC program that has many OUTPUT instructions, it can be useful to have an Output Display to show the output data sequence or to proceed like real outputs on screen. Figure 9 shows the Output sequence associated with one of lmc programs from Examples Database – incrementing.lmc – that asks user for boundary limits (24 and 33 were input) and then shows all the numbers between them.
- **Print Report**: the student can save and/or print his/her programs. The Print Report content is very similar to the Editor Window.



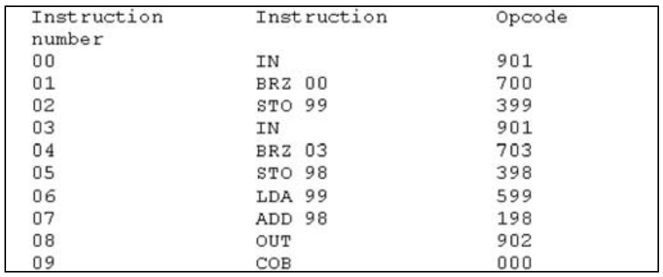
```
Number|Instruction|Parameter|Comment

00 IN;inputs num1
01 BRP 3;continues if positive
02 BR 0;if negative, asks again
03 STO 99;stores num1 at mailbox 99
04 IN;inputs num2
05 BRP 7;continues if positive
06 BR 4,if negative, asks again
07 STO 98;stores num1 at mailbox 99
08 SUB 99;if positive then num2>=num1
09 BRP 11;continues.
10 BR 4;if not, asks again the superior bound (num2)
11 LDA 99;puts inferior bound at calculator
12 OUT;outputs the number
13 ADD 97;Increments the value
14 STO 99;stores the value to continue
15 SUB 98;superior bound?
16 BRP 18;if actual_number-superior_bound>=0 then stops
17 BR 11;repeat
18 HLT;program
```

Figure 9. Print Report example

4. EDU.LMC SIMULATOR TESTS WITH MI STUDENTS

Edu.LMC was tested with AC MI students during two formal testing sessions. All students previously attended classes about LMC Paradigm and, at least, have done LMC Programming "on paper". The first group was composed by students that didn't work previously with a LMC Simulator and the second group knew and had already manipulated a LMC simulator: Shockwave LMC. We gave an "edu.LMC Session Guide" to each student and they were supposed to explore the simulator basic options, and, after that, implement a new LMC program (the code was provided) using the application.

On the first half of the session, they opened an existing .lmc file and run the program, exploring the various options about Execution. They had to register a set of values about calculator, mailboxes, instruction location counter, instruction name and flags (to check if values are positive, negative or zero). There were no significant differences about the two student's group behavior. On the second half, we asked students to write the program in Figure 10, to describe each instruction result, to save, execute and print the program. This program is a very simple example of using LMC instructions: it asks the user for a non zero number, stores it in mailbox 99, asks for a second non zero number and stores it in mailbox 98, adds the two numbers, presents the sum result and stops execution. There is an instruction that isn't really necessary: 06 LDA99. Instead, it's possible to optimize the program, because the second number is still in the calculator, so, it's possible to sum it with the number in mailbox 99 (the first input number), 06 ADD 99. With this solution there is no reason to store the second number because it's possible to sum it immediately. All these optimization details were observed during this session. They are also important in other programming language contexts.



Instruction number	Instruction	Opcode
00	IN	901
01	BRZ 00	700
02	STO 99	399
03	IN	901
04	BRZ 03	703
05	STO 98	398
06	LDA 99	599
07	ADD 98	198
08	OUT	902
09	COB	000

Figure 10. Print Report example

In what concerns edu.LMC evaluation, the second group of student's considered that it was easier to write programs with this simulator than with Shockwave LMC and appreciated the new functionalities, namely: open and save lmc files, print files, access to the help file and the examples database. When we asked them the functionality they appreciated the most and the answers were unanimous: functionalities about files. Shockwave LMC imposes programming in 3 digits code (no mnemonics allowed), the programs are directly written using the mailboxes and it isn't possible to save or print programs.
The first group didn't focus on the same functionalities: since they had implement LMC programs always "on paper" they refer, as the most interesting feature, the Execution Area that allows LMC program's testing. Both groups also mentioned also that edu.LMC really helped them to understand the basic concepts of Computer Architecture and assembly language.

5. CONCLUSIONS AND FUTURE WORK

We presented edu.LMC, a new simulation approach using LMC Paradigm. This simulator has a clear pedagogical focus for a specific target public – undergraduate MI students – and collects numerous features which are difficult to find together in one unique simulator. We tested our simulator on a learning context, with two distinct groups, and concluded that this simulator could help students to learn Computer Architecture concepts and Assembly language. Students considered it an efficient way to understand the LMC Paradigm and the related concepts. edu.LMC can be improved by adding functionalities concerning address modes, adding more examples to the database, implementing the array concept, creating a more efficient way to represent the fetch/execute cycle and, since edu.LMC is a pedagogical simulating tool, one possibility about future work is putting simulation on a context were students can find solutions in a collaborative learning basis and making edu.LMC available through an Internet Platform to support that functionality. edu.LMC is available for download at www.iscac.pt/~ipedrosa/LMC/edu_lmc.htm and we welcome all feedback about it.

6. ACKNOWLEDGMENTS

We would like to express our gratitude to ISCAC students that tested edu.LMC, giving precious advices, in order to turn it a better pedagogical simulation tool.

7. REFERENCES

[1] Yurcik, W., Osborne, H., "A Crowd of Little Man Computers: Visual Computer Simulator Teaching Tools", *Proceedings of the 2001 Winter Simulation Conference*, 2001.

[2] Englander, I., "The Architecture of Computer Hardware and Systems Software: an information technology approach", 2.nd Edition, John Willey and Sons Inc., 2000.

[3] Little Man Computer. In Illinois State University: School of Information Technology, USA, http://www.itk.ilstu.edu /faculty/javila/lmc/ [online], created on 1998, last modified :01-05-2000, e-mail: ljbrumb@ilstu.edu, accessed on 10-07-2005.

[4] Osborne, H, Yurcik, B., "The Educational Range of Visual Simulations of the Little Man Computer Architecture Paradigm", *Proceedings on the 32nd ASEE/IEEE Frontiers in Education*, 2002.

[5] Yurcik, W., Vila, J., Brumbaugh, L., "An Interactive Web-Based Simulation of a General Computer Architecture", *IEEE International Conference on Engineering and Computer Education (ICEDE 2000)*, San Paulo, Brazil, August, 2000.

[6] CAALE - The Computer Architecture and Assembly Language Education Homepage, http://www.sosresearch.org/caale/ [online], last modified: 12-09-2005, accessed on 12-01-2006.

[7] WWW Computer Architecture Page – Simulators, http://www.cs.wisc.edu/~arch/www/tools.html [online], last modified: 04-01-2006, accessed on 10-01-2006.

[8] Cassel, L., Holliday, M., Kumar, D., Impagliazzo, J., Bolding, K., Pearson, M., Davies, J., Wolffe, G., Yurcik, W., "Distributed Expertise for Teaching Computer Organization & Architecture", *ACM SIGCSE Bulletin*, Vol. 33, n.º 2, June 2001, pp. 111-126.

[9] Yurcik, W., Brumbaugh, L."A Web-Based Little Man Computer Simulator", *Proceedings of the 32nd ACM SIGCSE Technical Symposium on Computer Science Education*, ACM Press: 204-208, 2001.

[10] Yehezkel, C., Yurcik, W., Pearson, M., Armstrong, D., "Three Simulator Tools for Teaching Computer Architecture: EasyCPU, Little Man Computer, and RTLSim", *ACM Journal of Educational Resources in Computing*, Vol. 1, No. 4, December 2001, Pages 60-80.

[11] Friend of Son of Little Man Computer (FoSoLMC) http://fosolmc.lazyblue.com/fosolmc/#release [online], accessed on 11-11-2005.

[12] Acquaintance of Friend of Son of Little Man Computer (AoFoSoLMC), http://fosolmc.lazyblue.com/fosolmc/aofosolmc/ [online], accessed on 15-11-2005.

[13] dat2343/01f/notes, http://teaching.idallen.com/dat2343/01f/notes/checkbox_files .cgi, [online], created on 1998, last modified :17-04-2005, accessed on 09-07-2005.

[14] Little Man Computer, http://www.herts.ac.uk/ltdu/projects/ mm5/ [online], accessed on 03-03-2006