UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN
FACULTAD DE MINAS

DYNA
http://dyna.medellin.unal.edu.co/

Assembly language and processor design: an integrated project

Iván de Jesús Deras-Tabora & Nelson Alberto Lizardo-Zelaya

Faculty of Engineering, Universidad Tecnológica Centroamericana, San Pedro Sula, Honduras. ivan.deras@unitec.edu.hn, nelson.lizardo@unitec.edu.hn

Received: May 8th, 2019. Received in revised form: December 3rd, 2019. Accepted: December 13th, 2019.

Abstract
The research describes a project in computer organization class with two groups, one in 2017 and the other in 2018, in a trimester course of 68.34 hours which integrated both topics assembly language and hardware design. The project involves the implementation of a reduced version of an embedded MIPS32 system, with a simulated and real hardware implementation using FPGA (Field Programmable Gate Array). Followed by the development of a game in C and assembly language that runs on the embedded system. The results show that all students from the first and second group that coursed computer organization class during a 10-week period expressed high levels of interest and engagement, despite the complexity of the project. With feedback from the first group and with some modifications to the project, all students from the second group successfully completed the project.

Keywords: assembly language; computer organization; FPGA, simulation.

Lenguaje ensamblador y diseño de procesador: un proyecto integrado

Resumen
La investigación describe la aplicación de un proyecto en la clase de organización de computadoras con dos grupos, uno en 2017 y otro en 2018, en un curso trimestral de 68.34 horas que integró lenguaje ensamblador y diseño de hardware. El proyecto implica la implementación de una versión reducida de un sistema MIPS32 embebido, con una implementación simulada y una en hardware utilizando FPGA (Field Programmable Gate Array). Seguido del desarrollo de un juego en C y lenguaje ensamblador que se ejecuta en el sistema embebido. Los resultados muestran que todos los estudiantes del primer y segundo grupo que cursaron la clase de organización de computadoras durante un período de 10 semanas expresaron altos niveles de interés y participación, a pesar de la complejidad del proyecto. Con la retroalimentación del primer grupo y con algunas modificaciones al proyecto, todos los estudiantes del segundo grupo completaron el proyecto con éxito.

Palabras clave: lenguaje de ensamble; organización de computadoras; FPGA; simulación.

1. Introduction

Computer organization is one of the classes whose main objective is to teach students how the world of software and hardware get together. The software part includes the learning of assembly language and the hardware part is mostly about applying digital circuit design to understand the process carried out by the computer in order to execute the software. Usually the software part gets more attention and many simulators [1-3] have been developed with the purpose of teaching assembly language.

The hardware part is more complicated since it involves the application of digital design knowledge, which in many cases is limited for Computer Science students; also, it requires the investment in hardware that allows the implementation of digital circuits. Some approaches have been tried in this regard, like hardware simulation [4,5], using microcontrollers and FPGAs (Field Programmable Gate Array) to implement digital circuits [6,7]. The problem with these approaches is that they work separately, commonly hardware projects and assembly language projects are not integrated.

How to cite: Deras-Tabora, I.J. and Lizardo-Zelaya, N.A, Assembly language and processor design: an integrated project. DYNA, 87(212), pp. 57-62, January - March, 2020.

Taking all this into account we decided to try a new approach by combining the two parts (hardware/software) into a single project. The project is challenging taking into account the limited digital design knowledge of students and timing constraints (the course last only 10 weeks). The project consists of the development of a MIPS32 SOPC (MIPS32 System on Chip) and a simple game implemented using assembly language and C that runs on the SOPC.

This section explains the structure of which the course was developed and its general guideline. Section 2 describes the details of the development of hardware and software requirements and components. In Section 3 laboratory experiments and specific project design tasks are detailed. Section 4 explains the implementation of the software and hardware project. Section 5 emphasizes on the student's experience, taking in account their feedback for improvements and examining their attitude toward the project. To conclude in Section 6, the difficulties and results of the project are discussed and in Section 7 we review areas of improvement for future courses based on the instructor assessment of the project and student's comments.

2. Project description

The primary objective of this project is to allow students develop their own hardware and then program it to implement a game that tests the processor in a real situation and with this reduce the gap between technologies used in the industry and academia. Previous approaches to develop similar projects have been done in longer periods, spanning from 1 semester to 2 semesters. Some of this projects have a reduced complexity in programming and in some cases the implementation in hardware is optional, currently we have no evidence of a research were they test it with a game applying the combination of hardware and software in a reduced timeframe. [8-11].

The hardware part of the project composed by a MIPS32 SOPC, which includes a reduced version of a MIPS32 CPU, a VGA (Video Graphics Array) text mode driver, a timer and a keypad. These components were chosen taking into account that students will develop and test a game for this system. The software part involves the development of a game that runs on the SOPC using C and MIPS32 assembly language. We decided to use a game instead of some other software project because games are engaging for students as pointed by [12].

2.1. Hardware description

The interaction between the hardware components is shown in Fig.1.

The keypad, timer and the VGA modules are input/output components, which are available to the software through MMIO (memory mapped input/output). We decided to use MMIO because it is simpler than for example interrupt based IO.

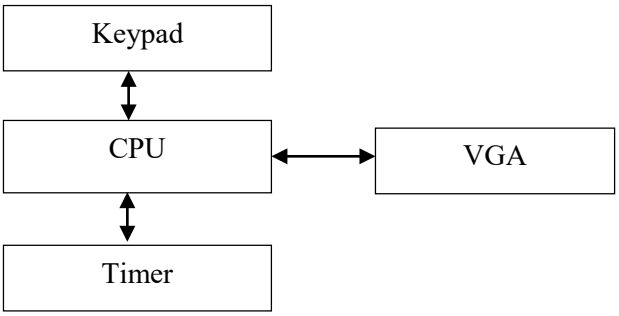

Figure 1. MIPS SOPC hardware component.
Source: The Authors.

2.1.1. The keypad

The keypad provides eight keys; we use the MMIO address 0xffff0004 to read the states of the keypad. The state is represented in a byte were each bit represents the status of an individual key, 0 means the key is released and 1 if the key is pressed.

2.1.2. The timer

The timer counts the number of milliseconds passed since the system was started, this value is stored in a 32-bit special register. We use the MMIO address 0xffff0008 to read the value of such register.

2.1.3. The VGA module

The VGA module is a text mode driver, which provides a resolution of 80 columns by 30 rows and a palette of 16 colors. The content of the VGA display can be accessed (read and write) through the range 0xb800 to 0xcabf, where every word is a 16 bits value which encode the color attribute and the symbol to display. The color attribute is an 8-bit value, where the leftmost 4 bits represents the background color and the other 4 bits represent the foreground color, which is the color used to display the symbol. The symbol is displayed based on a fixed size font that is stored in a read only memory (ROM).

2.1.4. The CPU module

The CPU module implements 39 instructions from the MIPS32 instruction set [13]. The implemented instructions are: nop, sll, srl, sra, sllv, srlv, srav, jr, add, addu, sub, subu, and, or, xor, slt, sltu, jmp, jal, beq, bne, blez, bgtz, addi, addiu, slti, sltiu, andi, ori, xori, lui, lb, lh, lw, lbu, lhu, sb, sh, sw. The CPU includes 16KB of instruction memory to store programs, 8KB of data memory and 8KB for stack.

2.2. Software part description

The development of the game was divided into two parts: the development of controllers for the hardware components

and the actual game logic. The hardware controllers include a VGA controller, keypad controller and a timer controller. The VGA controller was developed in C and the other two controllers were developed in assembly. For the second part every student chooses a game, and then implements it using the code developed in the first part.

The toolchain used in this part was GCC (GNU Compiler Collection) version 7.1 for MIPS32 with some extra tools developed for the project, we will discuss this later on the paper.

3. Project design

Due to the complexity of the project, its implementation required careful planning. As part of this process, we decided to implement the project before applying it to the students in the course. This allows us to get a better understanding of the tools and manuals needed in order to help students implement the project themselves.

Two main issues were addressed in the project implementation: 1) students limited background in hardware design. 2) time constraints of the project, due to the class timeframe of 10 weeks taught Monday to Thursday with a class duration of 1 hour 20 minutes, with a total of approximately 53.34 hours in total.

We addressed the first issue by having 1 hour and 30 minutes of additional sessions on Friday during the 10 weeks, adding 15 hours to the project and making it a total of 68.34 hours. Even though the students had already taken a digital design class, the extra sessions were used to review digital circuit's implementation in Verilog and to help students resolve issues with their project implementation.

The time constraint was addressed by involving the students with the project since the beginning of the class. We were able to do this by allowing the students to work on the software part of project first. For this part they were given a simulated model of the SOPC, an FPGA configuration file, and compilation tools. They started using C, because they have some background on C++ and Java in addition the students have already coursed 4 programming classes therefore having some knowledge in this area. Then as the class advanced they were asked to implement some parts in assembly.

The simulated model of the SOPC was implemented in Verilog and C++. We used Verilator [14] to compile Verilog code into C++ code, which we extended with other C++ code. The VGA display, the input keys for the keypad and the timer were implemented using SDL2 (Simple DirectMedia Layer) library. The simulated implementation was extremely helpful to debug the logic of the SOPC and for the hardware implementation we compiled the Verilog code using Xilinx ISE Webpack 14.7 then loaded the circuit implementation into a MIMAS V2 FPGA Board, which used a Spartan 6 FPGA.

For the software part, we used a Tetris [15] implementation available in C using the ncurses library. We ported the game to our SOPC by using GCC for MIPS. This process allows us to define the instructions needed in the CPU, which we implemented accordingly.

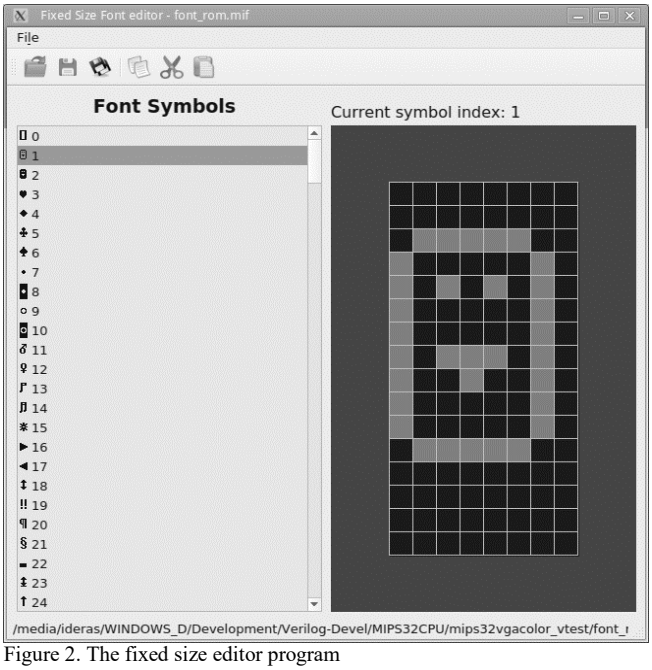

Figure 2. The fixed size editor program
Source: The Authors.

Figure 3. The SOPC simulator running a test program
Source: The Authors.

During the planning process we developed three additional tools (used during the game development part):1) Fixed Size Font Editor (Fig. 2 shows a screenshot), to edit the VGA font. 2) Elf2Mif which extracts the machine code and data from an ELF (executable and link format) file generated by the GCC compiler and output them into two files in hexadecimal format, these files are used in simulation and during compilation of the Verilog code. 3) Elf2Mem does a similar job to Elf2Mif, but the output format is compatible with Xilinx tool data2mem [16]. The output is combined with the FPGA programming file, which allows us to run the program on the FPGA board.

Additionally, we decided to use a graphical tool for circuit design, Digital [17], which was useful for students to

understand the data flow of the hardware part. In the second implementation of the project, we added two features to Digital, one to export a graphical circuit to Verilog and another to use a Verilog circuit. This allowed students to work faster and with fewer errors in the circuit.

4. Project implementation

For the implementation of the project, students had their own laptop computer with a Linux distribution. Additionally, they installed GCC for MIPS compiler, Verilator, the SOPC simulator (Fig. 3 shows a screenshot of the simulator running a test program), the FPGA configuration file for the SOPC, Fixed Size Font editor program, Elf2Mif and El2Mem tools and Digital. The Fixed Size Font Editor program allows the student to edit characters from the VGA ROM to create game sprites. The project was divided into the following parts:

- Software
  - VGA controller
  - Keypad and Timer controller
  - The game
- Hardware
  - Single cycle CPU with asynchronous memory. Supported instructions: lw, sw, add, sub, and, or, slt, beq, bne, j
  - Implement a simple virtual addressing mode for memory. Added support for instructions: addu, addi, addiu, andi, ori, lui
  - Adding support for memory access instructions: lb, lbu, sb, lh, lhu, sh.
  - Adding support for more branch and bit manipulation instructions: bgez, bgtz, blez, bltz, nop, sll, srl, sllv, srlv, sra, srav, xor, xori, sltu, slti, sltiu, jal, jr
  - Building the final circuit.

For the VGA controller students were asked to implement the following functions:

- *clear_screen*: clear the entire VGA display.
- *set_cursor*: Set current position identified by row and column. The current position is the one used to print something on screen.
- *set_color*: Set the current color attribute. This is the background and foreground color.
- *print_char*: Prints a character using the current position and color attribute into the VGA display.
- *puts*: Prints a null-terminated string using the current position and color attribute into the VGA display.

This part was implemented in C and for the keypad and timer controller students were asked to implement the following functions:

- *delay_ms*: Make a delay for specified milliseconds.
- *get_key*: Return the current key pressed. Every key is identified by a number starting at 1. If no key is pressed 0 is returned.

This part was implemented in assembly and for the game part students were allowed to choose a simple game to implement, only one level was required. Among the games chosen by students are: Tetris, Pong, Pacman, Snake, Adventure. Fig. 7 shows screenshots of some of the games implemented by students.

For this part, students used C and assembly. The use of C allowed students to take into account aspects like register and calling convention, which is important to follow in order to integrate assembly code with C code.

For the hardware implementation students starts with a single cycle CPU implementation using asynchronous memory which is simpler to use than synchronous memory, in this part students implement the necessary instructions and test them in Digital. The second part asks students to implement support for virtual addressing mode for the instruction and data memory. In this part they add support for some arithmetic instructions. The third part asks students to add the synchronous memory modules including the VGA and Data memory. At the same time, they add support for the missing memory access instructions from part 1. In the fourth part students implement the missing branch and bit manipulation instructions.

Part 1 through 4 were developed and tested in Digital. We used automated testing supported by Digital, which proved to be very useful to ensure circuits were working correctly before reaching the last phase. The fifth is the last part, here students have to first simulate the whole system using Verilator and implement their project in hardware using Xilinx ISE and the FPGA. They generated the Verilog code from Digital, then changed the code to add modules like: clock manager to generate the input clock for the processor and the VGA module, synchronous memory modules to be able to implement the circuit in hardware and pin mapping. Finally, students run the game using their own SOPC implementation.

5. Student experience

We implemented this project with two generation of students of computer organization class, the first group in 2017 on a class with 6 students and the second group in 2018 with 7 students. Since the beginning of the class we explained clearly to them what was the project about, this makes them realize that the project will require time and dedication.

For the 2017 group all the students finished the game, but nobody finished the MIPS32SOPC at all. One student completed the circuit, but could not implement it on the FPGA, it worked only in simulation. Therefore, to identify the issue we requested from the students what where the limitations on completing the task and we got two main issues: 1) At the moment of the circuit construction they had double workload, graphically in a tool called Digital then manually translate the circuit to Verilog. 2) They mentioned that implementing pipelining was too complex.

In the case 2018 group, we used the feedback of the previous group for improving the project. The modification of Digital to generate Verilog to prevented double workload and the new multicycle processor version excluded pipelining. The addition of a detailed scheduled task delivery

with automated tests for the hardware part. This helped students to ensure that every part of the circuit was working correctly before moving to the next part. The results show that successfully all 7 students could implement the circuit in simulation and in the FPGA also all of them were able to run their game.

For the 2018 group we handed a survey with 8 questions to the seven students: 1) How difficult was the project? 2) What was the most difficult part? 3) Contribution level of the project, 4) How engaging and entertaining the project was? 5) Do you recommend this project for future classes? 6) How do you consider the instructions? 7) About the tools? 8) External factors that influence the development of the project. Questions 1, 3, 4 and 6 used a Likert scale, question 5 required a yes/no answer. The other questions (2, 5, 7) were open questions. Figs. 4, 5 and 6 shows a chart for the answer to questions 1, 3 and 4. From that, we can conclude that even though students consider the project difficult, it contributed to the learning and was engaging and entertaining.

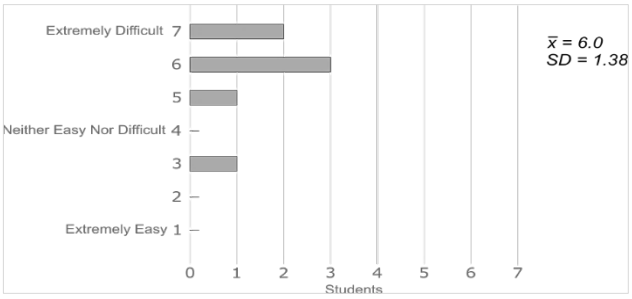

Figure 4. Level of difficulty of the project
Source: The Authors.

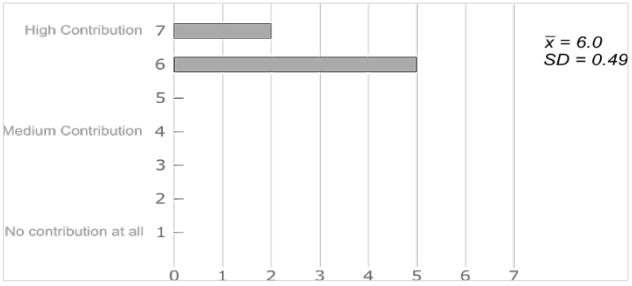

Figure 5. Project contribution to learning process
Source: The Authors.

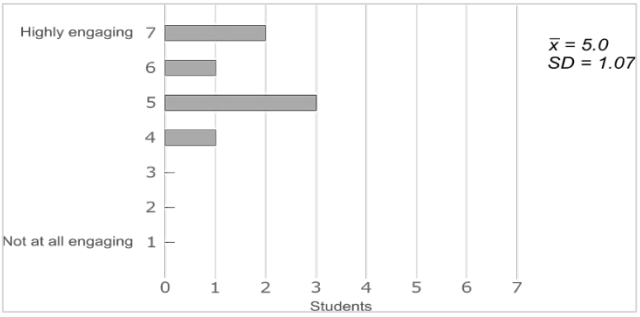

Figure 6. Level of engagement and entertaining
Source: The Authors.

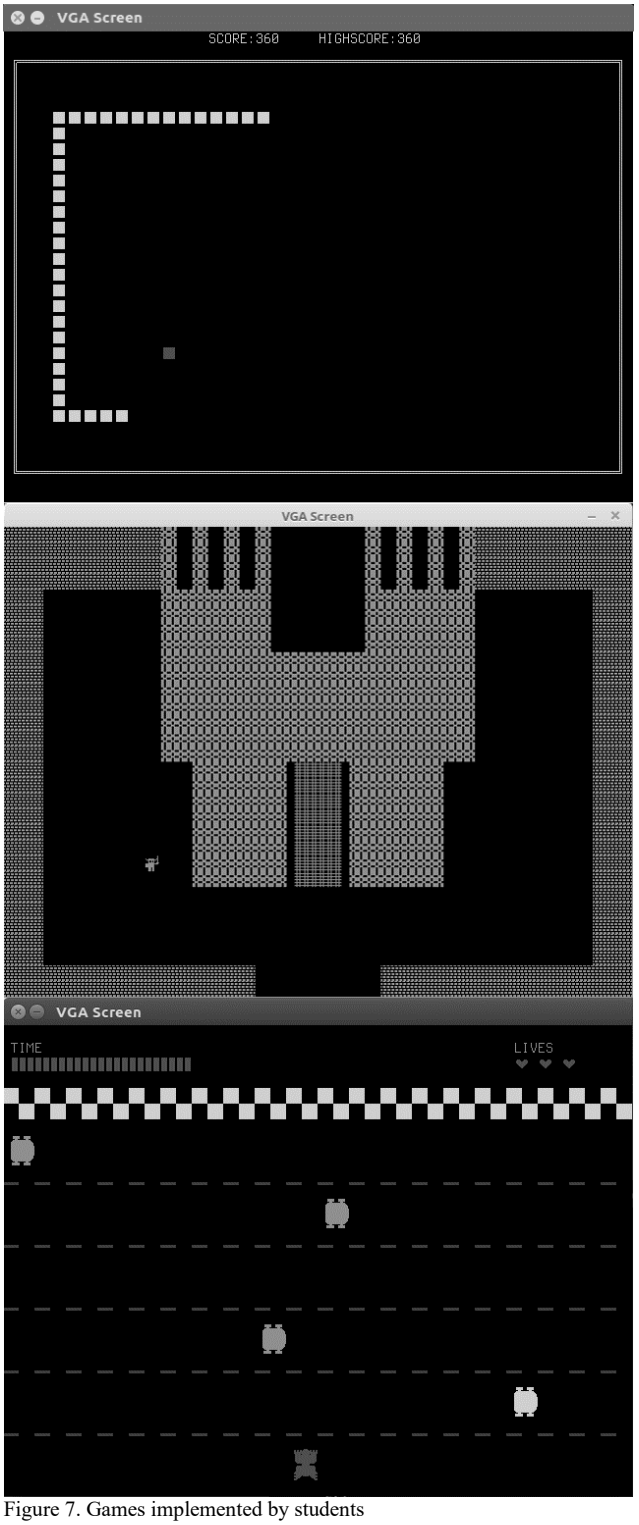

Figure 7. Games implemented by students
Source: The Authors.

6. Conclusion

Learning assembly language and applying Digital Circuit Design to Computer Hardware Organization are two crucial topics of a Computer Organization class. As far as we know

this is the first attempt to combine the two topics with this level of integration.

The paper shows that is possible to get students involved not only with hardware implementation but also with assembly language programming, which helps them get a better understanding of how the hardware and software integrates in building complex computational systems.

With the feedback from the first group of computer organization class in 2017 and with improved pedagogical techniques, the 2018 group manage to complete the software and hardware successfully in the short time period of 10 weeks.

The results show that is possible to get students involved not only with hardware implementation but also with assembly language programming, which aids in getting a better understanding of how the hardware and software integrates in building complex computational systems.

7. Future work

We are pleased with how the project was implemented, however in future courses we want to test the use of C++ instead of C in the software phase. This brings certain challenges since C++ is difficult to integrate with assembler compared to C, due to features like: inheritance, polymorphism and other object oriented programming characteristics. On the other hand, we have been working with the author of Digital to integrate the changes to generate Verilog code from a graphical representation of a circuit and also be able to use components with descriptions in Verilog as part of a circuit in Digital. At the time of elaboration of this article, now Digital includes all the contributions of this project. Therefore, is possible to generate Verilog code from Digital and also the use of a Verilog circuit on Digital.

References

[1] Vollmar, K. and Sanderson, P., MARS: an education-oriented MIPS assembly language simulator. In: SIGCSE '06 Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, New York, USA, 2006. DOI: 10.1145/1121341.1121415

[2] Zilles, C., SPIMbot: an engaging, problem-based approach to teaching assembly language programming. In: SIGCSE '05 Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, New York, USA, 2005. DOI: 10.1145/1047124.1047391

[3] Black, M.D. and Komala, P., A full system x86 simulator for teaching computer organization. In: SIGCSE '11 Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, New York, USA, 2011. DOI: 10.1145/1953163.1953272

[4] Norris, C. and Wilkes, J., YESS: a Y86 pipelined processor simulator. In: ACM-SE 45 Proceedings of the 45th annual southeast regional conference, New York, USA, 2007. DOI: 10.1145/1233341.1233369

[5] Black M. and Waggoner, N., Emumaker86: a hardware simulator for teaching CPU design. In: SIGCSE '13 Proceeding of the 44th ACM Technical Symposium on Computer Science Education, New York, USA, 2013. DOI: 10.1145/2445196.2445294

[6] Black, M. Export to arduino: a tool to teach processor design on real hardware, Journal of Computing Sciences in Colleges, 31(6), pp. 21-26, 2016.

[7] Jipping, M.J., Henry, S., Ludewig, K. and Tableman, L., How to integrate FPGAs into a computer organization course,. In: SIGCSE '06 Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, New York, USA, 2016. DOI: 10.1145/1121341.1121414

[8] Black, M., A processor design project for a first course in computer organization.in: Annual Conference & Exposition, Pittsburgh, Pennsylvania, USA. [online]. 2008, 12 P. Available at: https://peer.asee.org/4444

[9] Jung, Y.-K., An innovative rapid processor platform design for early engineering education. In: Proceedings of the 2005 American Society for Engineering Education Annual Conference & Exposition, Fayetteville, AR, USA, 2005

[10] Matos, J.S., Alves, J.C., Mendonça, H.S. and Araújo, A.J., From Boolean algebra to processor architecture and assembly programming in one semester. In: Design of Circuits and Integrated Systems, Madrid, 2014, pp. 1-5. DOI: 10.1109/dcis.2014.7035605

[11] Patt, Y., Introduction to computer systems from bits and gates to C and Beyond, McGraw-Hill Education, 1999, 656 P.

[12] C.A. Bodnar, Anastasio, D., Enszer, J.A. and Burkey, D.D., Engineers at Play: games as teaching tools for undergraduate engineering students, Journal of Engineering Education, 105(1), pp. 1-200, 2016. DOI: 10.1002/jee.20106

[13] MIPS, MIPS32 Architecture for programmers. Vol. II: The MIPS32 Instruction Set, MIPS Technologies, Inc., California, USA2001.

[14] Snyder, W., Galbi, D. and Wasson, P., Verilator, Veripool.org, [Online]. 2017. [Accessed August 31th of 2017]. Available at: https://www.veripool.org/wiki/verilator.

[15] Brennan, S., Tetris, [Online]. [Accessed August 31th of 2017]. Available at: https://github.com/brenns10/tetris.

[16] Xilinx, Data2MEM User Guide, Xilinx, 2007.

[17] Neemann, H., Digital, [Online]. [Accessed August 31th of 2017]. Available at: https://github.com/hneemann/Digital.

I. de J. Deras-Tabora, received the BSc. Eng in Computer Systems in 2002 from the Universidad Tecnologica Centroamericana, Tegucigalpa, Honduras, and MSc. in Computer Science in 2017 from the Stony Brook University, Nueva York, USA. Since 2003 has been working as a teacher in the Universidad Tecnologica Centroamericana, in 2018 he started working as a research teacher. His research interests are Computer Science Education and Computers Applied to Education.
ORCID: 0000-0001-8094-843X

N.L. Lizardo-Zelaya, received the BSc. Eng in Logistics Engineering in 2009, the MSc. in Finance in 2011, and MBA in 2012, all of them from the Universidad Tecnologica Centroamericana, Honduras. He finished a PhD in Economics and Finance in 2018 from Hanyang University, Republic of Korea with focus in risk management and financial engineering. Since 2018 has been working in the Universidad Tecnologica Centroamericana, San Pedro Sula, Honduras, he is working as a researcher and professor in the Faculty of Engineering. His research interest are risk management and logistic systems.
ORCID: 0000-0002-3963-5690