MIPS X-Ray: A MARS Simulator Plug-in for Teaching Computer Architecture

http://dx.doi.org/10.3991/ijes.v2i2.3527

Marcio R. D. Araújo[1], Flávio L. C. Pádua[1], Fabrício V. Andrade[1] and Fabio L. Corrêa Junior[2]
[1] Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), Belo Horizonte, MG, Brazil.
[2] Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais (IFMG), Belo Horizonte, MG, Brazil.

***Abstract*—In this paper, we address the overall project and resulting development of a new plug-in called MIPS X-Ray, intended for the MARS simulation environment of the MIPS architecture, which is widely used in the processes of teaching and learning about computer architecture in various educational institutions throughout the world. Specifically, through the use of graphical animations, the proposed plug-in enables a better understanding of data flow models and the combined operation of functional units given the execution of certain instructions by the architecture processor. Thus, several computer architecture concepts, which are often complex and abstract, can be more easily presented to and assimilated by the students in this learning area. Through the use of a validation source code with the main functions of the MIPS instruction set, the tests performed with the proposed plug-in demonstrate the significant potential of this tool.**

***Index Terms*—Computer architecture education, graphical visualization of datapaths, MARS simulator, MIPS architecture, MIPS X-Ray plug-in.**

I. INTRODUCTION

In computer architecture and organization classes, the teaching approach that has proven most effective for analyzing the operation of a processor is based on the use of datapath diagrams [1]-[3]. Due to the dynamic nature of the processor, in which a significant amount of data and control signals are exchanged between functional units, a visual aid is necessary to understand correctly the operating logic [4]. However, a full understanding of the operation cannot be achieved by observing only one static diagram, using conventional devices such as pen and paper, for example [5]-[7].

Throughout the execution of an instruction, functional units perform different functions according to the control signals received, changing the unit status at each instant [1]. This dynamic behavior creates difficulty in analyzing the overall operation of the processor when observing the corresponding diagram because it is only relevant at a given moment. Therefore, generating a series of diagrams that address each datapath state is one of the options for displaying all of the operations of the processor. However, this solution is highly inefficient because it results in a significant number of static diagrams [8]. A more acceptable solution is the development of a tool capable of displaying a graphical animation of the operation of the datapath, according to the instruction processed at any given moment.

In light of this scenario, we present a new tool, called MIPS X-Ray, that is capable of addressing the problem of the static representation of a datapath through the creation of graphical representations and animations of the datapath, in response to the instructions entered as inputs. To do so, the MIPS instruction set, belonging to the family of RISC processors, is used, given its widespread use for teaching purposes [1], [9]. The structure of a single cycle without a pipeline using functional units with the black- box structure, i.e., containing only the input, output and control signals for each functional unit, was selected for datapath modeling.

It should be emphasized that it is not our goal in this study to create a full-system simulator but instead to focus on the process of processor graphical representation. To achieve this goal, the present work is based on the use of one of the most complete and functional simulators among those currently available, specifically, the MARS simulator [9]. Thus, the system proposed in this paper was developed in the form of a MARS simulator plug-in, adding to it the capacity of processor operation visualization.

The main advantage of this approach is the use of a mature simulator, which allows the work to be focused on the creation of a specific tool for datapath visualization. The developed system provides the user of the MARS simulator the possibility of viewing the operation of the processor while it executes the code generated by the user. Thus, it creates the possibility to analyze jointly the programming logic and the operational aspects of the architecture hardware.

The MIPS X-Ray tool is potentially applicable in theoretical courses and computer architecture labs, serving as an important aid for users of the MARS development environment who wish to deepen their knowledge of the hardware field.

This paper builds on our previous work [16] with (1) a new and improved paper structure to present the MIPS X-Ray plug-in; (2) an updated discussion of related work; (3) a more comprehensive explanation of the proposed plug-in, including the presentation of new functionalities and enhancements, specifically: (i) a new datapath model, which represents the MIPS architecture more detailed, displaying not only the data flow among functional units, but also the control signals used, (ii) the possibility of using different datapath models; and (4) a new section that briefly describes tests and results obtained.

II. RELATED WORK

In recent years, significant effort has been made to develop simulators, development environments, and hardware descriptors that are able to contribute to the processes of teaching and learning computer architecture [2], [6], [8]-[12].

Brorsson [8] conducted one of the first studies on this topic by proposing a simulation tool, named MipsIt, focused on the modeling and simulation of functional units. Unlike the approach proposed in the present study, the MipsIt tool presents data only in text mode, not offering graphic simulations of the combined operation of these units in a datapath.

Additionally, the authors of the MiniMips simulation project [10] proposed a new system based on the use of functional units with a high level of abstraction, not representing the data exchange in the RT (register-transfer level) model. Similar to MipsIt, the simulations performed by this system do not offer the possibility of performing graphical animations.

In [11], the authors introduced the WebMips system, the goals of which resemble the goals of the system proposed in this paper. However, unlike the MIPS X-Ray, the WebMips system performs datapath simulations with a pipeline, abstracting the control signals of the functional units and representing only the data flow and the state of each functional unit.

Garton [12] introduced the ProcessorSim, a tool that performs a simulation of the operation of a single-cycle datapath using the MIPS R2000 processor as the base. Similar to the MIPS X-Ray system, the ProcessorSim shows the data flow between functional units, although only one functional unit sends messages at a time, which varies from the actual operation of a processor that sends simultaneous messages.

Vollmar and Sanderson [9] developed one of the most complete and efficient simulators for the MIPS architecture, called MARS, providing a mature development environment, registers and memory simulation tools, as well as educational tools that demonstrate concepts of computer architecture that interact with the source code created by the user.

In [6], the authors presented a simulator and assembler capable of simulating a single-cycle MIPS processor, focused on the display of values contained in each functional unit at the time of the simulation; however, the simulation was devoid of a light mesh of the input, output and control signals.

Finally, Kabir et al. [2] presented the VisiMips tool, with particular focus on the simulation of pipelines. Among its other features, this tool implements a multi-cycle MIPS processor with a pipeline, representing a hazard detection unit in its datapath, as well as a forwarding unit and other functional units. It is noteworthy that this tool has a parser and assembler unit for compiling the implemented source code.

The remainder of this article is organized as follows. Section III of this paper initially presents information regarding the MIPS architecture, such as an instruction set and the features of the logical units, and then a brief description of the MARS simulator is given, describing its basic features, in particular those used in this study to integrate the MIPS X-Ray system in the form of a plug-in. The datapath structure used in the simulations is then presented with a description of the functional units, as well as the signals generated and received by the functional units. Section IV presents and analyzes a set of test results developed using the MIPS X-Ray system, and Section V presents the conclusions and proposals of future work

III. MIPS X-RAY

The MIPS (Microprocessor without Interlocked Pipeline Stages) architecture is derived from the RISC (Reduced Instruction Set Computing) architecture and possesses a wide range of applications, such as routers, printers and onboard devices [1], [3], [13].

Among the basic features of the MIPS architecture, the existence of instructions of fixed length can be highlighted, each of which possesses the same number of bits, allowing for a more rapid interpretation of the operation to be performed [2].

A. The MIPS Instruction Set

The instruction set of the MIPS architecture, and of the MIPS32 architecture in particular [14], is formed by 32-bit instructions, as illustrated in Figure 1. In this case, a fixed-length 6-bit opcode is used, while the subsequent bits may vary according to the type of instruction to be executed.

Generally, these instructions can be classified into three types [15]:

1) Register type (R-Type) is a type of instruction in which all of the data used are stored in registers. Each register is addressed using 5 bits: the first two 5-bit groups are used to store the register addresses, which will provide the operation data, and the subsequent 5-bit groups are used to provide the register address, which will store the results of the operation. The R-type instructions are the most used from the architecture, comprising arithmetic and logical operations [13];

2) Immediate type (I-Type) is a type of arithmetic instruction in which the data are entered directly into the instruction bits. The first group of 5 bits is used to store the address of the registers, while the remaining 16 bits store the data that will be used in processing. Sum operations with an immediate value, value comparison and storage of data from memory into registers are examples of operations based on this type of instruction;

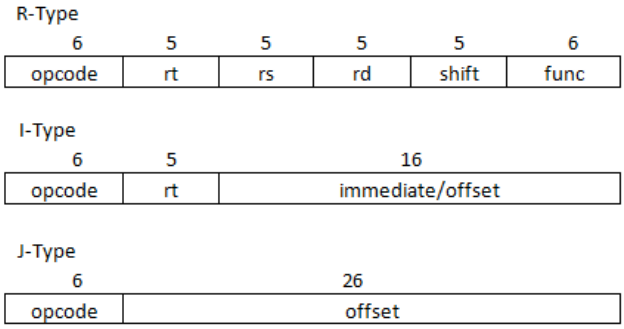

Figure 1. The MIPS instruction format.

3) Jump Type (J-Type) is a type of instruction for flow deviation, in which the sequence of instructions is changed for a jump to a given point. The deviations are unconditional, i.e., they occur without the need for a test. Jump and Jump Register operations are given as examples of such instructions.

B. The Simulation Environment of MARS

The MARS (MIPS Assembler and Runtime Simulator) application is an integrated development environment for the assembly language in the MIPS architecture. MARS was specifically developed to support teaching activities on architecture and computer organization [9].

Among other resources, MARS displays the simulation features of the MIPS32 architecture [14], containing step-by-step execution control, 32 registers, a memory simulator, and an integrated code editor and assembler.

In addition to its native features, the MARS application, which was developed with an open-source license, frequently adds new plug-ins to its structure and is therefore constantly improving the range of applications of this environment. In this scenario, the MIPS X-Ray system proposed in this paper is a new plug-in for the MARS application, aiming to provide MARS with the capacity for datapath visualization [16].

C. The Datapath Functional Units

A single-cycle datapath without a pipeline and with a focus on the data and control signals was defined for the development of the MIPS X-Ray, as shown in Figure 2. The use of functional units as black boxes was established as the initial approach, and therefore, only considers the input, output, and control signals as relevant. The operation of each functional unit will be addressed in future work.

Note that the datapath defined for the MIPS X-ray follows the model proposed by Patterson and Hennessy [1] and possesses the following functional units:

• **PC (Program Counter):** this is a simple register with the function of storing the address of the next instruction to be executed. This register sends the value of this address for the instruction memory and receives a new value, which can be derived from the adder, which increments the address value to the next instruction or, in the case of deviations, to the value of the next memory address;

• **ALU (Arithmetic Logic Unit):** this unit is responsible for performing arithmetic and logical calculations. Its input data consist of the values of two registers and a control signal that defines the operation to be applied to the data. This unit generates as the output the resulting value of the operation applied to the data and a binary signal that indicates the result of comparison operations;

• **ALU Control:** this unit defines which operation will be executed by the ALU, receiving its bits directly from opcode and transferring the defined operation to the ALU;

• **Instruction Memory:** this unit works as a memory bank, storing the instructions derived from the source code. This functional unit stores the bits that will be sent to all functional and control units, so that necessary operations can take place. The instruction memory has a data input that receives from the PC the memory location that will be accessed and an output that sends the 32-bit instruction that will be executed;

• **Bank of Registers:** this unit is responsible for storing the 32 registers used by the MIPS architecture. The bank of registers exhibits three register address inputs to be used, one data input to be stored in the registers and one control input, which defines whether the register will read from or write in the registers. This unit also has two outputs for sending the data contained in the registers, which were addressed by the input data;

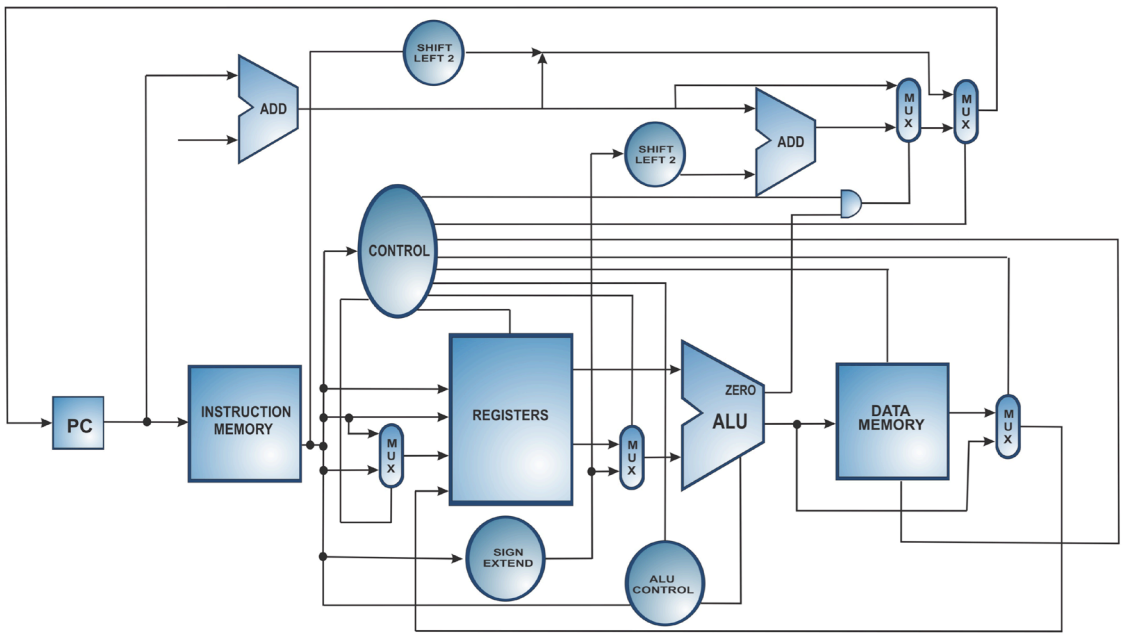

Figure 2. Functional units of the datapath defined for the MIPS X-Ray plug-in.

• **Control Unit:** the control unit receives the bits that correspond to the opcode, identifies which instruction will be executed and sends control signals for each functional unit, informing each unit what it should do. For this reason, the control unit is connected to all of the functional units under its control;

• **Data Memory:** the data memory is the unit responsible for storing the data to be persisted after instruction execution. The data from the data memory remain in the memory until either the time at which they are overwritten or when the power is switched off;

• **Sign Extend:** in some situations, all 32 bits of a register are used to store a certain value. When the data considered has a sign bit, it should be extended to the last bit that is reserved for its storage. In this context, the function of the sign extend unit is to fill the remaining bits of the register so that the data signal is not missed;

• **Shift Left:** the function of this unit is to shift the bits of an instruction to the left and fill with zeros the remaining bits that have become empty;

• **Multiplexers:** these functional units are used to define which signal will be transmitted by the system. Multiplexers, similar to the other functional units, are controlled by the control unit;

• **Adders:** adders are arithmetic units with the function of adding the input signals, receiving two values and returning the resulting sum as the output signal.

D. Signal Propagation through the Datapath

The designed functional units work together to process a given input instruction. The modules interact through an exchange of signals or bit sequences. A color sequence to define each type of signal, in which each color represents a set of bits generated by a given functional unit, was used to implement the MIPS X-Ray plug-in.

As shown in Figure 3, a legend is dynamically created based on the current instruction, displayed as (1) the manner in which the bits are divided and sent to each functional unit, (2) the specific mnemonic of the instruction and (3) a description of the control signals arriving at each functional unit.

The MIPS X-Ray plug-in uses the architecture of the MARS simulator to execute one instruction at a time, which is the approach used to visualize the graphical animation produced. Note that such an approach is based on a single-cycle datapath, where each instruction is executed in an atomic manner [16]. Therefore, the user should expect that all of the instruction is presented before jumping to the next step of the plug-in execution.

Several data that can be accessed directly through the MARS simulator were used to implement the plug-in. After compiling the source code, the instructions are loaded into an instruction memory simulator, and the instructions can be accessed through the application programming interface (API) of the simulator. Thus, it is possible to analyze the opcode and define the type of instruction that will be executed.

Additionally, to display information on the current instruction, the MIPS X-Ray plug-in accesses the data from the bank of registers and the data memory, which are stored in MARS in a data structure that simulates those functional units. Access to this data is important for the plug-in because it would not be possible to show all of the information that is being manipulated when using only the instruction bits to represent the data flow, which would make it difficult to understand the real operation of the datapath.

The Graphics library, which belongs to the set of Java standard libraries, was used in this study to implement the MIPS X-Ray plug-in. The graphical animations produced consist of representations of the signals that leave the functional units and go to the destination unit. Each color used is related with the e instruction bits, as shown in Figure 3, through the legend positioned to the left of the datapath.

Finally, to enable the use of different types of datapaths, the MIPS X-Ray plug-in was designed in a modularized way, to separate the operating logic from the parameters used to generate the data flow of the produced graphical animation. Specifically, the animation parameters are stored in an XML file, making it possible to easily change the path of the animation by editing the parameters contained in this file.

E. MIPS X-Ray Benefits to Student Learning

The teaching of hardware technologies and, specially, the topic of computer architecture and organiza¬tion is usually a challenging task in Engineering and Computer Science courses. Several reasons can be pointed out for that, such as [17]: (i) the ever-expanding amount of relevant materials, since new methods have been continuously proposed, (ii) the need for understanding distinct subjects, such as, electronic circuits, digital logic, assembly-language programming and system design, to cite just a few, and (iii) the need to design and execute both hardware and software experiments to demonstrate several important concepts[18].

In some of the main reference books on this field [1], [19], [20], the approach commonly used to explain the hardware of a specific processor consists in to divide it in functional units, such as, registers, arithmetic logic unit, data memory, among others, and present the connections and control signals exchanged between them. Additionally, it is frequently addressed the use of an assembly language, which is radically different than most high-level languages, consisting in a well-known hard learning process. For pedagogical purposes, the learning of an assembly language is valuable to understand several low level aspects, as for example, the difficulty of allocating registers for data calculations and address references and what high level control constructs (e.g. if, while, for and switch) look like when translated to assembly. For real world use, on the other hand, sometimes one needs to escape to assembly for performance reasons or to access hardware resources not accessible from a high level language. Due to the complexity of all those concepts, the students usually face great difficulties to understand and use them.

In this scenario, educational simulators have been created to provide support for teaching and learning by mimicking the behaviors of computer hardwares [3], [4],

[10]-[13], [21] and allowing to handle the gap between the assembly language used and the functional units of the corresponding processor under analysis. However, most current simulators abstract an essential part of a processor operation, specifically, the signal propagation in the datapath. Therefore, even though simulators as MARS allow the users to familiarize with functional units and the effects of each instruction executed on them, it is not simple to understand how the processor's datapath was designed.

In fact, the comprehension of the signal propagation in a datapath depends on the presentation of what signals are activated according to the kind of instruction executed. The conditional branch and load instructions, for example, use distinct functional units in the datapth and, certainly, the inclusion of visualization or animation methods in the simulator considered appears as interesting alternative to aid student engagement and deeper understanding. To achieve this goal, we have developed the plug-in MIPS X-Ray for the MARS simulation environment of the MIPS architecture. By using MIPS X-Ray, the students may dynamically visualize the signals inside the MIPS processor, instead of trying to visualize them statically, as presented in the main reference books on this topic.

IV. RESULTS AND TESTS

To evaluate the operation of the MIPS X-Ray plug-in, a validation source code with the main functions of the MIPS instruction set was used. Thus, the identification of the instructions were tested, in addition to the register and memory values. Further, the behavior of the datapath was compared with the diagrams presented from the related basic literature. Figures 3 and 4 illustrate executions of the register and jump instruction types within the proposed plug-in.

To evaluate the use of different datapaths, the initial image of the datapath under evaluation was changed, and the corresponding XML configuration file was edited, defining a datapath configuration in which the location of functional units differed from the original. In this context, the data flow was then modified to coincide with the data input and output locations of each functional unit.

From the tests performed, it was possible to observe that the MIPS X-Ray plug-in successfully managed to display the operation of all instructions of the MIPS architecture set, correctly recognizing each type of instruction with its corresponding opcode, as well as the registers used during code execution.

Finally, the applicability of MIPS X-Ray was evaluated by several students from an undergraduate program in computer engineering of a Brazilian federal university, named CEFET-MG. Specifically, students who have attended the course “Computer Architecture and Organization” have interacted with MIPS X-Ray in practical and theoretical classes. In theoretical classes, MIPS X-Ray worked as an alternative pedagogical tool that contributed to explain the processor’s operation, replacing lots of static images commonly used to achieve this goal. In practical classes, however, after developing and compiling codes in assembly to solve basic math problems, the students used MIPS X-Ray to visualize the code execution in the processor’s datapath.

The MIPS X-Ray usefulness was then measured with the feedback of the students. They were required to answer a form in the end of the course in order to identify the improvements in the learning process due to the usage of MIPS X-Ray plug-in. That survey demonstrated that the students believe that MIPS X-Ray improves significantly the comprehension of several concepts related to the operation of a processor’s datapath, being especially useful for applications involving the understanding of the operation of single-cycle datapaths without pipelines.

V. CONCLUSIONS AND FUTURE STUDIES

This work presents a new tool in the form of a plug-in for the MARS simulator, providing a visual aid in the process of studying the operation of the MIPS processor. The tool developed, called MIPS X-Ray, is presented in the form of a dynamic datapath capable of responding to the source code generated by the user.

The approach employed allows the use of the main features of the MARS tool, such as the assembler, code development tool, and simulator of functional units, as well as the behavior of the processor.

Several possibilities for future studies include the development of new features for the proposed plug- in. In particular, a detailed analysis of the operation of each functional unit present in the datapath of the MIPS X-Ray could be performed, allowing the user to engage the black box approach in addition to collecting detailed information regarding the operation of each unit in the logic diagrams of the operation. Thus, in future versions of the plug-in, it is expected that the user will be able to click on the functional units as the datapath animation is occurring, allowing the user to visualize the inner workings of the functional unit of interest.

Finally, another possible future study consists of modifying the MIPS X-Ray datapath to allow implementation with the pipeline to be used. This approach will allow the user to analyze other structures used in this process, such as the hazard detection unit, the forwarding unit and the pipeline register bank.

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support of FAPEMIG-Brazil under Proc. APQ-01180-10; CEFET-MG under Proc. PROMEQ - 010/13; CAPES-Brazil and CNPq-Brazil.

REFERENCES

[1] D. A. Patterson and J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface. Morgan Kaufmann, 2008.

[2] M. T. Kabir, M. T. Bari, and A. Haque, "VisiMips: Visual Simulator of MIPS32 Pipelined Processor", in IEEE International Conference on Computer Science & Education, 2011, p. 788-793.

[3] H. Sarjoughian, Y. Chen, and K. Burger, "A Component-based Visual Simulator for MIPS32 Processors", in IEEE Frontiers in Education Conference, 2008, pp. F3B-9.

[4] B. Nikolic, Z. Radivojevic, J. Djordjevic, and V. Milutinovic, "A Survey and Evaluation of Simulators Suitable for Teaching Courses in Computer Architecture and Organization", IEEE Transactions on Education, vol. 52, no. 4, pp. 449-458, 2009. http://dx.doi.org/10.1109/TE.2008.930097

[5] H. Grunbacher, "Teaching Computer Architecture/Organisation Using Simulators", in IEEE Frontiers in Education Conference, 1998, pp. 1107-1112.

[6] M. A. Lusco and Stroud, "PSIM: A Processor SIMulator for Basic Computer Architecture and Operation Education", in IEEE SoutheastCon, 2010, pp. 115-118.

[7] C. Yehezkel, W. Yurcik, M. Pearson, and D. Armstrong, "Three Simulator Tools for Teaching Computer Architecture: Little Man Computer, and RTLSim", ACM Journal on Educational Resources in Computing, vol. 1, no. 4, pp. 60-80, 2001. http://dx.doi.org/10.1145/514144.514732

[8] M. Brorsson, "MipsIt: A Simulation and Development Environment Using Animation for Computer Architecture Education", in ACM workshop on Computer Architecture Education, 2002, p. 12.

[9] K. Vollmar and P. Sanderson, "MARS: An Education-Oriented MIPS Assembly Language Simulator", in ACM SIGCSE Bulletin, vol. 38, no. 1, 2006, pp. 239-243. http://dx.doi.org/10.1145/1124706.1121415

[10] E. Z. Bem and L. Petelczyc, "MiniMIPS: A Simulation Project for the Computer Architecture Laboratory", in ACM SIGCSE Bulletin, vol. 35, no. 1, 2003, pp. 64-68. http://dx.doi.org/10.1145/792548.611934

[11] I. Branovic, R. Giorgi, and E. Martinelli, "WebMIPS: A New Web-based MIPS Simulation Environment for Computer Architecture Education", in ACM workshop on Computer Architecture Education, 2004, p. 19.

[12] J. Garton. (2005, Sep.) ProcessorSim, A Visual MIPS R2000 Processor Simulator. [Online]. Available: http://www.test.org/doe/

[13] L. Ming and C. Qixian, "A Research for the Optimization of MIPS Instruction Set Simulation", in IEEE International Conference on Computer Science & Education, 2009, pp. 1886-1888.

[14] N. Pinckney, T. Barr, M. Dayringer, M. McKnett, N. Jiang, C. Nygaard, D. Money Harris, J. Stanley, and B. Phillips, "A MIPS R2000 Implementation", in IEEE Design Automation Conference, 2008, pp. 102-107.

[15] K. Yi and Y.-H. Ding, "32-Bit RISC CPU Based on MIPS Instruction Fetch Module Design", in IEEE International Joint Conference on Artificial Intelligence, 2009, pp. 754-760.

[16] G. C. Sales, M. R. Araújo, F. L. C. Pádua, and F. L. Corrêa Júnior, "MIPS X-Ray: A Plug-in to MARS Simulator for Datapath Visualization", in IEEE International Conference on Education Technology and Computer, vol. 2, 2010, pp. V2-32.

[17] W. Yurcik and E. F. Gehringer, "A Survey of Web Resources for Teaching Computer Architecture", in ACM Workshop on Computer Architecture Education, 2002, p. 23.

[18] L. D. Feisel and A. J. Rosa, "The Role of the Laboratory in Undergraduate Engineering Education", Journal of Engineering Education, vol. 94, no. 1, pp. 121-130, 2005. http://dx.doi.org/10.1002/j.2168-9830.2005.tb00833.x

[19] A. S. Tanenbaum, Structured Computer Organization. Prentice Hall PTR, 1984.

[20] J. P. Hayes, Computer Architecture and Organization. McGraw-Hill, 2002.

[21] G. S. Wolffe, W. Yurcik, H. Osborne, and M. A. Holliday, "Teaching Computer Organization/Architecture with Limited Resources Using Simulators," in ACM SIGCSE Bulletin, vol. 34, no. 1, 2002, pp. 176-180. http://dx.doi.org/10.1145/563517.563408

AUTHORS

M. R. D. Araújo is with the Department of Computing, CEFET-MG, Av. Amazonas, 7675, Belo Horizonte, MG, Brazil (e-mail: marcio@decom.cefetmg.br).

F. L. C. Pádua is with the Department of Computing, CEFET-MG, Av. Amazonas, 7675, Belo Horizonte, MG, Brazil (e-mail: cardeal@decom.cefetmg.br).

F. V. Andrade is with the Department of Computing, CEFET-MG, Av. Amazonas, 7675, Belo Horizonte, MG, Brazil (e-mail: vivas@decom.cefetmg.br).

F. L. Corrêa-Júnior is with the Department of Computing, IFMG, Av. Prof. Mário Werneck, 2590, Belo Horizonte, MG, Brazil (e-mail: fabiocorrea@ifmg.edu.br).

Submitted in 23 January 2014. Published as re-submitted by the authors 12 May 2014.

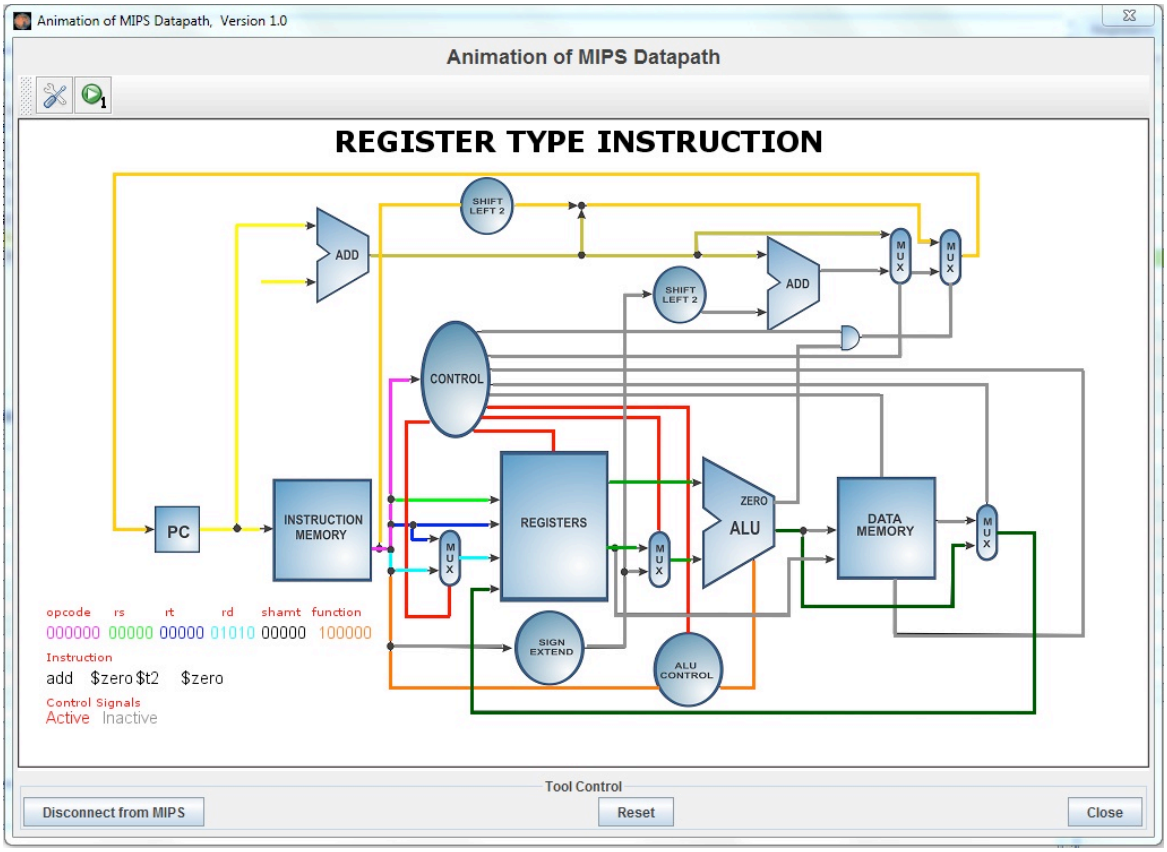

Figure 3. Execution of the register type instruction in the Mips X-ray plug-in.

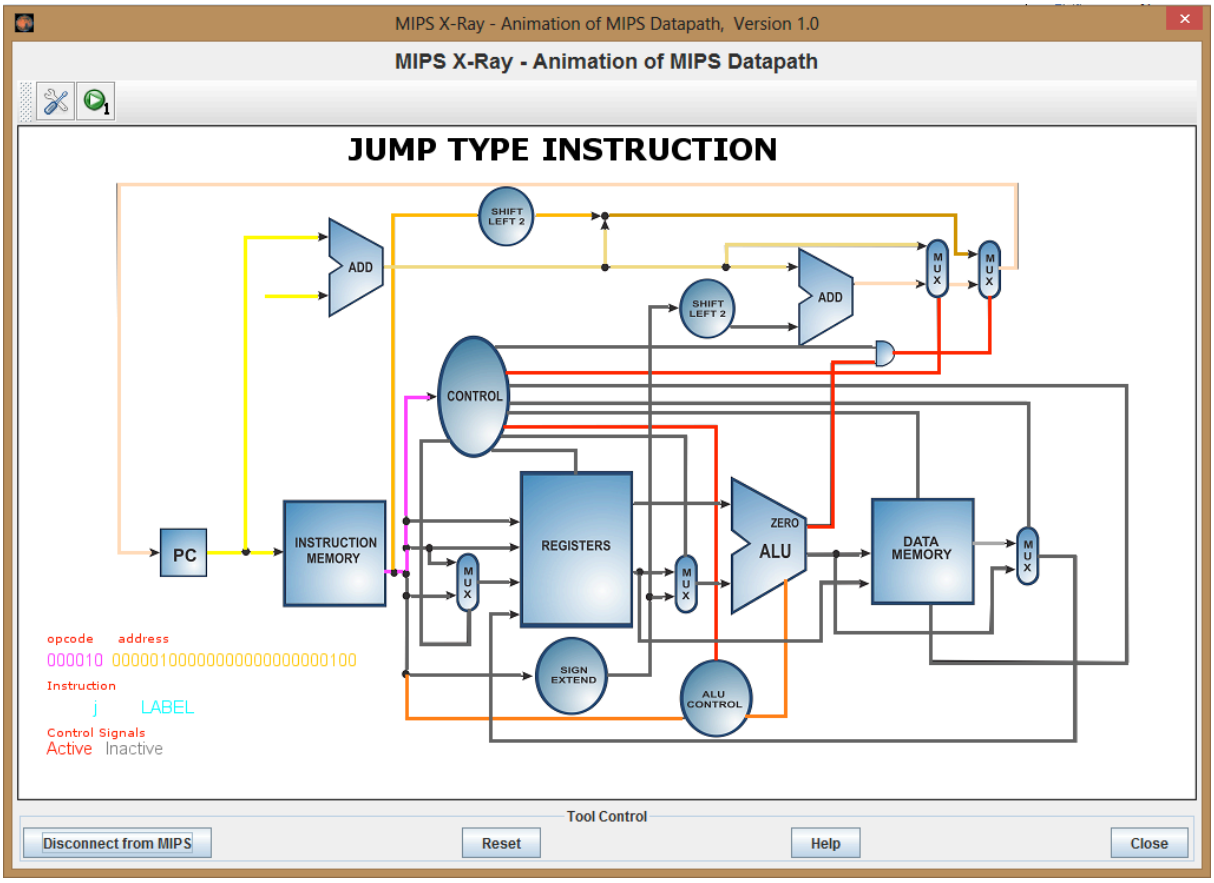

Figure 4. Execution of the jump type instruction in the Mips X-Ray plug-in.

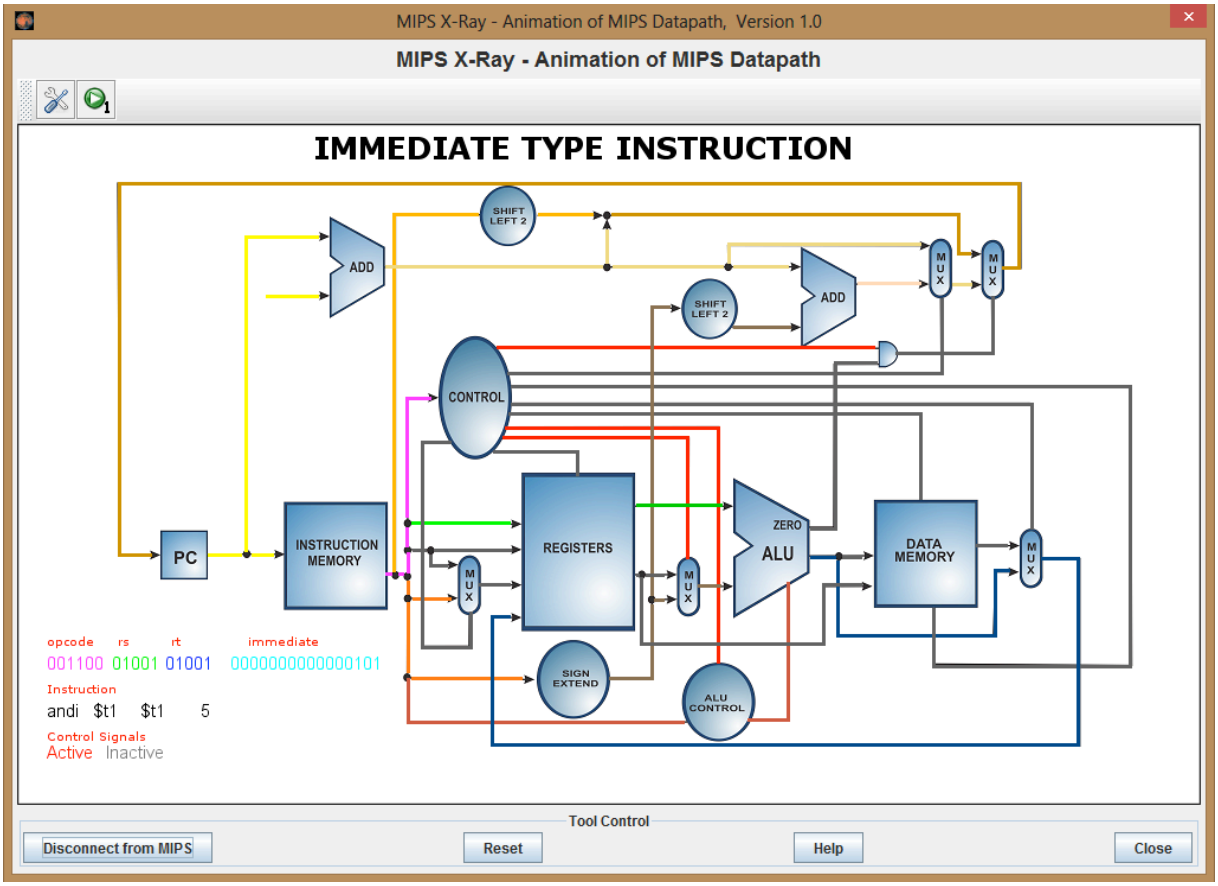

Figure 5. Execution of the immediate type instruction in the Mips X-Ray plug-in.