WebRISC-V: a Web-Based Education-Oriented RISC-V Pipeline Simulation Environment

Roberto Giorgi
Department of Information Engineering and Mathematics
University of Siena
Siena, Italy
giorgi@diism.unisi.it

Gianfranco Mariotti
Department of Information Engineering and Mathematics
University of Siena
Siena, Italy
gianfranco.mariotti94@gmail.com

ABSTRACT

WebRISC-V is a web-based server-side RISC-V assembly language Pipelined Datapath simulation environment, which aims at easing students learning and instructors teaching experience. RISC-V is an open-source Instruction Set Architecture (ISA) that is highly flexible, modular, extensible and royalty free. Because of these reasons, there is an exploding interest both in the industry and academia for the RISC-V. Here, we present the main features of this simulator and how it can be used for a simple exercise in the classroom. This web-based simulator permits the execution of RISC-V user-provided source code on a five-stage pipeline, while displaying the data of registers, memory and the internal state of the pipeline elements. One of the main advantages of WebRISC-V is the immediate availability in the web browser, thanks to its implementation as a server-side script in PHP.

CCS CONCEPTS

• **Computing methodologies → Simulation environments**; • **Applied computing → Interactive learning environments**; • **Computer systems organization → Architectures**; **Pipeline computing**; *Reduced instruction set computing*.

KEYWORDS

Computer Simulation, Computer Architecture, RISC-V, Processor Pipeline

ACM Reference Format:
Roberto Giorgi and Gianfranco Mariotti. 2019. WebRISC-V: a Web-Based Education-Oriented RISC-V Pipeline Simulation Environment. In *Workshop on Computer Architecture Education (WCAE'19), June 22, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3338698.3338894

1 INTRODUCTION

RISC-V is a recent open-source Instruction Set Architecture (ISA) based on reduced instruction set computer (RISC) principles [14]. It has a modular design, consisting of modular ISA parts, with added optional extensions. Two of the base ISAs modules (32-bit integer support or RV32I and 64-bit one or RV64I) are stable ("frozen"), as of the latest specification [20] and, additionally to these base modules, there are some optional modules that are stable too, such as the ones for multiplication (extension M) and floating point instructions (extensions F, D, Q).

Anyone who wants to make a RISC-V core can do so freely, since it has a permissive license for its ISA. Originally, it was designed to support Computer Architecture research and education, but nowadays it is also widely supported by more than 200 members of the RISC-V Foundation (including many big companies). RISC-V enables longer term software investments, since the specification of the ISA is not under control of a single vendor, but it is rather under the control of an independent foundation.

At the same time, the RISC-V is becoming very popular in Computer Architecture courses as a substitute of the MIPS processor. From a teaching point of view, the RISC-V has the same simplicity of the MIPS, but it solves some of its idiosyncrasies (like the usage of register RT instead of register RD in certain instructions). Moreover, it has very good chances to become an industrial standard for future microprocessors.

Experience shows that students may find difficulties in understanding the concepts of conventional instruction pipelining. The availability of a web-based tool can enhance the chances that students investigate the reasons of the good performance of the pipeline and also increases their curiosity in analyzing the internal state of the basic architectural elements. The tool also serves the need of testing simple assembly programs and see immediately the results in the browser. A simulator helps get introduced to the subject.

WebRISC-V emulates the five stages of the complete RISC-V integer pipeline, including the forwarding paths and the possibility to investigate the behavior of the hazard detection and forwarding units. The microarchitecture design is done in accordance with the book "Computer Organization and Design: RISC-V Edition" by D. A. Patterson and J. L. Hennessy [14], in which the pipelined datapath implementation is explored and explained.

The contribution of this work are:

- a refresh of the WebMIPS simulator [2] to cover the RISC-V ISA;
- the online availability of the WebRISC-V simulator as educational tool;
- the availability of the source code WebRISC-V simulator as an open-source tool (see http://www.dii.unisi.it/~giorgi/WebRISC-V/download).

WCAE'19, June 22, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6842-1/19/06.
https://doi.org/10.1145/3338698.3338894

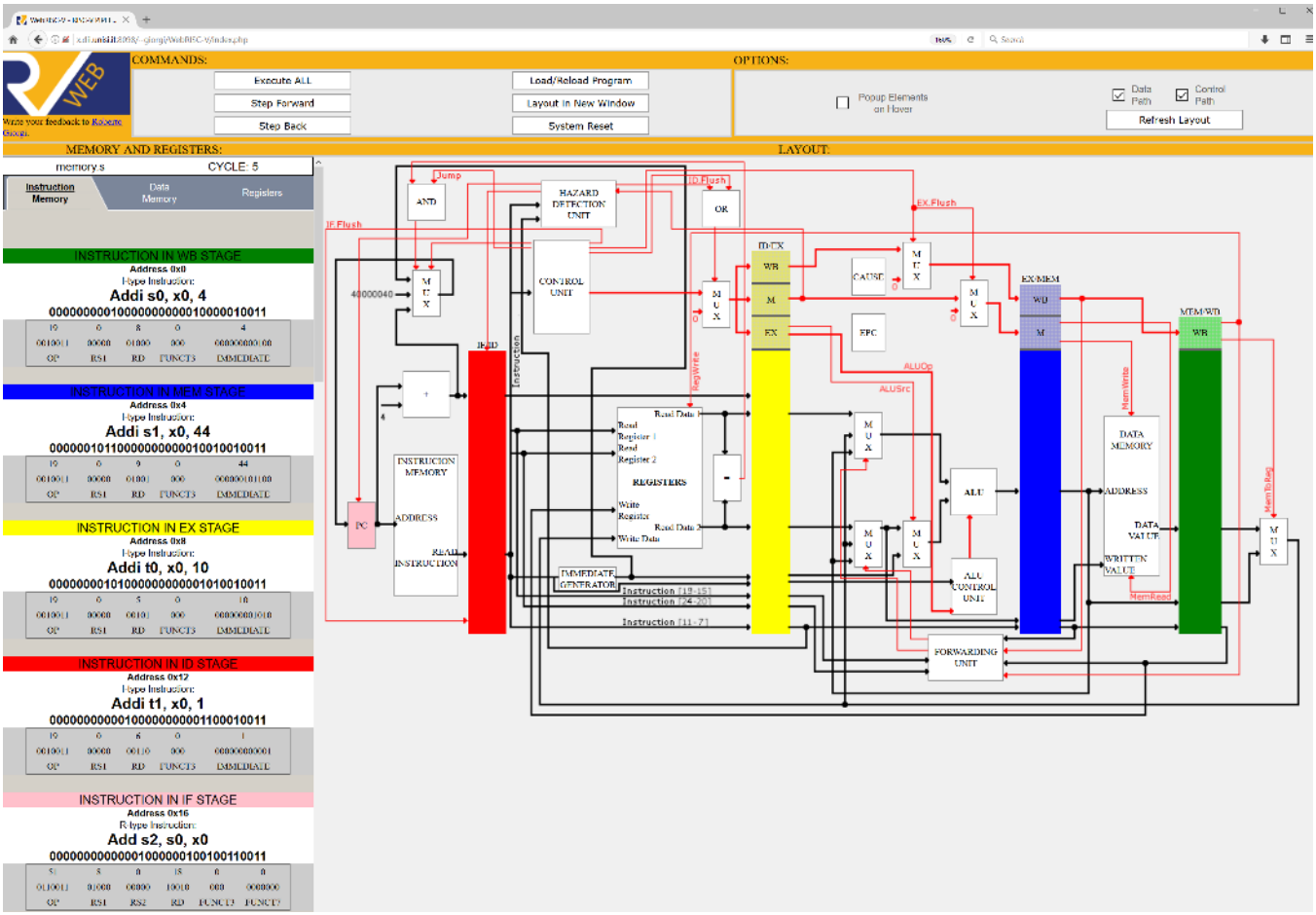

Figure 1: The WebRISC-V main page shows the structure of the pipeline. Each stage buffer has a different color (pink, red, yellow, blue, green). As can be seen on the left, the instructions get colored by the stage in which they are processed. The black wires represent the datapath wires, while the red wires represent the control part ones. In the upper part of the page, there are the commands and options.

2 RELATED WORK

Several works based on the MIPS architecture, such as [1–3, 5–10, 13, 17–19], were designed in order to achieve effective and easy learning through the practice on simulation tools. There exist several types of MIPS educational simulators. Among them there are simulators such as WepSIM [6], which displays hardware models of the processor, including the CPU, the main memory, some I/O devices, and allows the user even to microprogram them to further understand the inner workings. Other tools like JCachesim [1] permits the students to analyze the MIPS code impact on the cache subsystem. QtSPIM [10] and MARS [19] are ISA simulators, which are focused on the assembly code learning. They implement an almost complete MIPS instruction set and show the loaded program execution results through simple visual interfaces. Both of these tools simulate the single-cycle versions of the CPU but do not show the underlying datapath. Other simulators focus on visualizing datapath operation graphically, with some common features. Some of them show a simple single-cycle datapath: for instance MARS plug-in MIPS X-Ray [17]. DrMIPS [13] simulator let the user choose between execution in single-cycle or pipelined mode in the datapath model. Other tools show pipelined datapaths, usually including hazard detection and forwarding units. Notable examples of this group are WinDLX [8], Mipster32 [5], UCOMIPSIM [7], Visimips [9], WASP [18] and WebMIPS [2].

Most of these tools have the disadvantage of being accessible only after installation of additional software on a local computer, with a notable exception being WebMIPS [2], a web accessible pipelined datapath simulator that needs no additional software to work on a client. The use of server-side scripting permits this additional freedom. WebRISC-V also follows this philosophy by using PHP in order to make the tool readily available.

Tools for exploring RISC-V processors are just starting to appear [4, 11, 15, 16]. Therefore, we decided to make a RISC-V educational pipelined datapath simulator, taking inspiration from the old WebMIPS [2] and refreshing it with a more appealing ISA, while keeping its convenient web-accessibility. Some of those tools are valuable as simulation and development tools, but they have not been designed for educational purpose [4, 11, 16]. The Ripes simulator is

specifically designed for education [15] and provides a graphical representation of the flow of the instructions, as well as an assembly editor. WebRISC-V has the advantage of being completely server-side web-based and does not require any installation from the users. Moreover, compared to Ripes, WebRISC-V models and shows the forwarding units and the internal state of each architectural element of the pipeline.

Given the fact that WebMIPS was faithfully presenting the MIPS pipeline, we decided to recover the user experience in the new context of RISC-V. Also, we thought that - for students - it would have been important to minimize the difficulties to understand a totally new environment compared to the MIPS. Moreover, the students can also play with both WebMIPS and WebRISC-V and appreciate the little differences of the two tools. Thus, we recovered almost all features of the WebMIPS, extended some missing ones and fixed some bugs: we will discuss, in Section 4, the differences between the two simulators. Both simulators permit to see the details of the pipelined datapath, the content of the registers, instruction and data memory, input and output values of each architectural element. The students and teachers can dynamically visualize the processing of instructions of source code, using existing examples or by writing code in the browser directly.

3 FEATURES

In this Section, we outline the main features of the WebRISC-V simulator.

3.1 General Structure

WebRISC-V has its back-end written in PHP and its front-end in HTML and JavaScript [12], and as such can be executed from the Web browser, providing the advantage of immediate accessibility to students without any installing (the sister project WebMIPS was previously written in the less supported ASP scripting language). On the other side, if the teacher wants a local installation, he/she has to make a single installation on a Linux or Windows server.

Being a server-side web application, it is installed and executed on a web server and presented to the user on their client interface. This simulator includes most of the instructions of the 64-bit RISC-V base ISA module and its multiplication extension. Here the idea is to support only a subset of the ISA, which is enough to write simple programs like, e.g., a recursive factorial. To avoid slowdowns or crashing of the server, the execution of each uploaded program is limited to 1000 clock cycles, and data memory is limited to 5 KiB. So, in case of eventual programming errors, such as infinite loops, the execution can stop anyway in a short time. As previously outlined, most of the features of the WebMIPS simulator are also available in the WebRISC-V with the addition of:

- support of the RISC-V ISA itself;
- 64-bit support and little-endian addressing;
- microarchitecture modifications of the pipeline design to properly execute the RISC-V instructions.

These additions will be discussed in detail in Section 4.

3.2 Loading Code

WebRISC-V loads the RISC-V assembly via the Load/Reload Program button. The user can choose among one of the built-in examples, modify the existing code, or write it from scratch. When loading the assembly instructions, the parser checks if there are unsupported/miswritten instructions or miswritten labels. If there is an error, the simulation stops and the corresponding line number is displayed. In case of no error, the instructions are ready to execute in the pipeline. To help the student learn the instructions, the list of supported instructions is always visible on the left side, besides the text box (Figure 2).

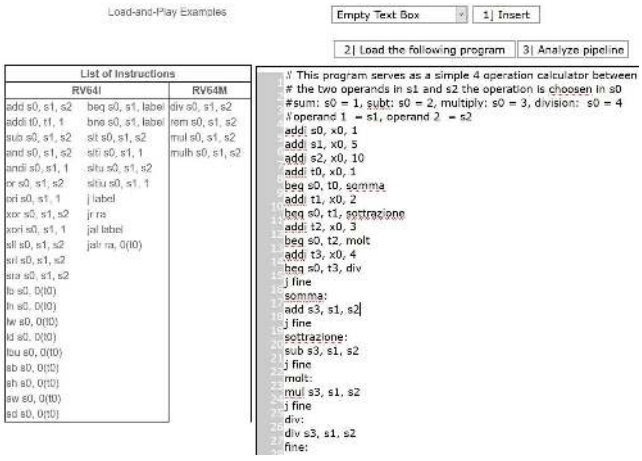

Figure 2: The Load Program page shows one of the built in programs. On the left, there is the complete list of supported instructions, always visible.

3.3 Program Execution

WebRISC-V assembly code execution can happen in two modes, by executing all the code at once, or step-by-step. The stage buffers have a specific color (pink, red, yellow, blue, green respectively for the Fetch, Decode, Execute, Memory and Write-Back stages). The same color is used in the loaded program that is visible on the left. Each instruction get a color based on the stage where it is currently processed (left of Figure 1). The current clock cycle is always visible besides the program name on the left. After the execution has completed, the total number of clock cycles is displayed, as can be seen in Figure 3. Execution of all the code at once is mostly used for verifying the correctness of the assembly code, but could be used if the user is interested in the final state of registers and data memory or the total clock cycles.

By executing in single steps, the user can follow in the left panel the advancing of instructions in each stage of the pipeline and analyze the value of registers and content of data memory. At the same time, in the main page representing the pipeline schematic, the student can observe the internal state of the architectural elements. By clicking on the desired pipeline elements (for example the ALU or Control Unit), or by activating the "pop-up on hover" function and passing the pointer over them, the user can see the input and output values of the unit. The main panel also gives the option to hide the Data or Control wires through the corresponding checkboxes. For each instruction, the Instruction-Memory tab displays (Figure 4):

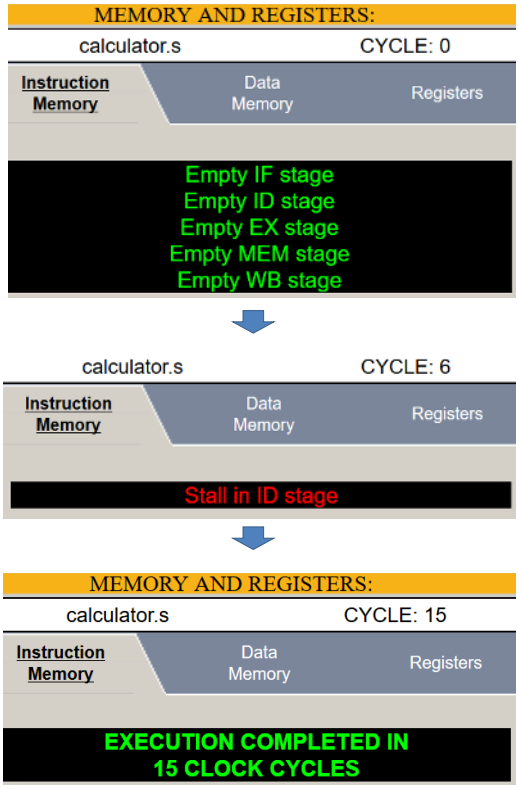

Figure 3: Clock cycle count and final message at the end of the execution.

- the instruction-memory address,
- the instruction type (R, I, S, SB, or UJ),
- the binary representation,
- the binary and decimal value of each field (OPCODE, RS1, RS2, RD, FUNCT, IMMEDIATE)
- the current stage (Fetch, Decode, Execute, Memory, Write-Back) of the instruction in the pipeline.

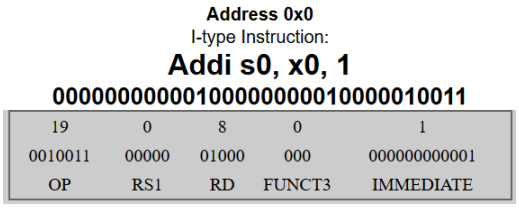

Figure 4: The fields of the binary code are clearly specified for each instruction, along with its mnemonic, machine value (numeric value), instruction type, and the value of each field.

The Data Memory tab displays the dword content (64-bits) at a single address, at a specific address range or the content of the whole data memory (Figure 5).

The Register tab shows the content of the registers. The registers are identified either by their register number or their calling convention name. Their values are shown both in binary and decimal format (Figure 6).

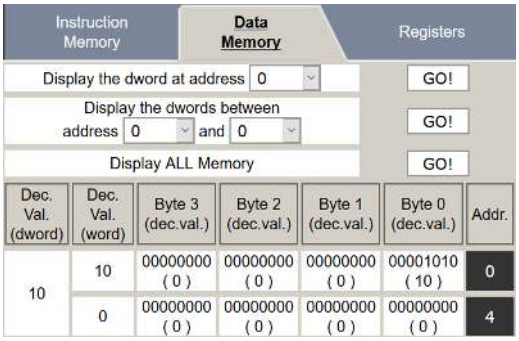

Figure 5: The content of the data memory can be inspected.

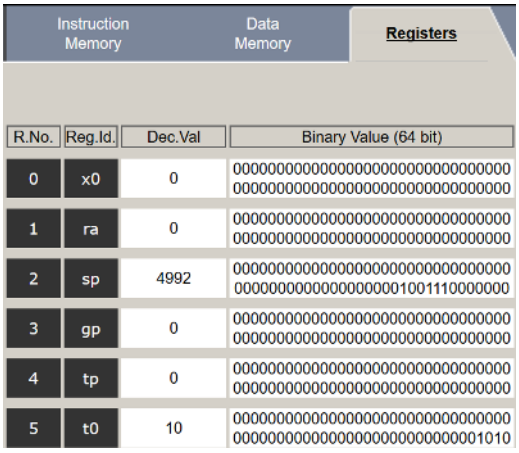

Figure 6: The content of the registers.

3.4 A simple exercise for the student: pipeline data hazards and forwarding during execution

In the pipeline implementation, as described in the Patterson-Hennessy book[14], the branch decision is taken in the Decode stage of the pipeline to save one cycle. To illustrate the functioning of the hazard detection and forwarding units, we use here one of the built-in assembly demonstration examples, i.e., the "simple calculator" program. By using this simple example, we can easily illustrate the functioning of hazard detection and forwarding in the pipeline, as a stall happens happens at cycle number 6. The left panel lists the pipeline stages that are in stall during the execution of the program, as can be seen in the top-left of Figure 7. Continuing execution, the data hazard resolves, the stall passes through the pipeline, and so the three instructions previously fetched continue their execution as is shown on the right of Figure 7.

By selecting the hazard detection and forwarding units in the pipeline, a user can see their corresponding input and output signals (Figure 8 and Figure 9), and so follow the propagation of the stall through the pipeline during execution, as shown in Figure 7.

As of this writing, WebRISC-V implements 31 RV64I integer instructions (two of them are actually pseudo-instructions – the J and the JR, which are mapped on the JAL and JALR, respectively)

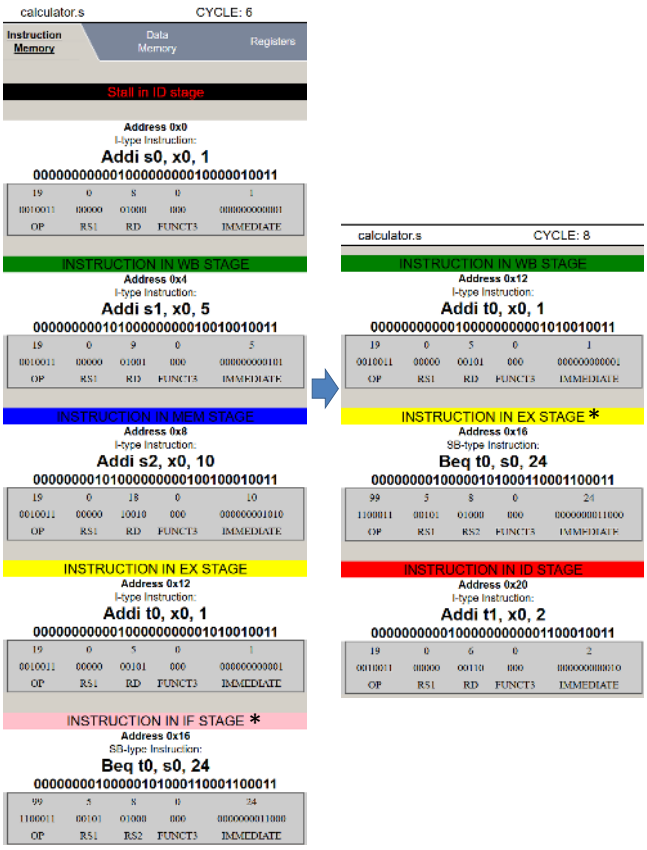

Figure 7: The case of a stall in the Decode stage due to a branch, whose condition is resolved only in the Decode stage. On the right, two cycles later, the three instructions previously fetched in the pipeline continue their execution after the stall that has been caused by the branch instruction. The star, besides the colored stage, indicates the stalled instruction.

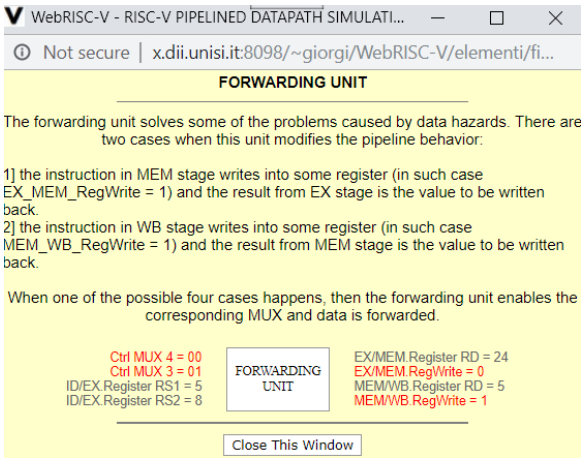

Figure 8: The content of the Forwarding Unit.

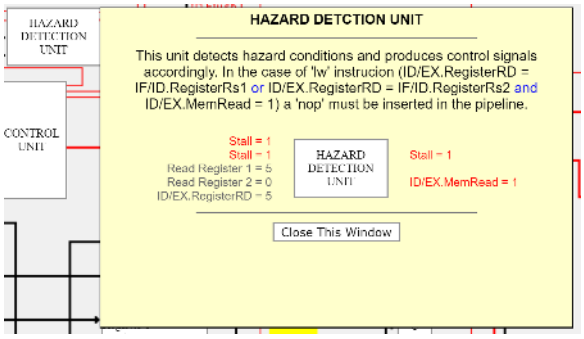

Figure 9: The content of the Hazard Detection Unit appears besides the unit, when the user hovers the mouse on it. The same happens for any other unit of the pipeline.

and 4 RV64M multiply extension instructions, as described in the RISC-V specification [20].

4 WEBRISC-V SPECIFIC FEATURES

There are several features that make WebRISC-V stand out from his sister project WebMIPS. The WebMIPS is big-endian, while the WebRISC-V implements little-endian addressing. Little endianness is more convenient for extending the architecture from words of 32 bits, to double-words of 64 bits or quad-words of 128 bits, so it is an important feature to be noted by the students. There are also some changes in the pipeline from MIPS to RISC-V (Figure 10), such as the removal of the RegDst signal, which selects between the RT and RD registers in the Decode stage, since the RISC-V core instruction

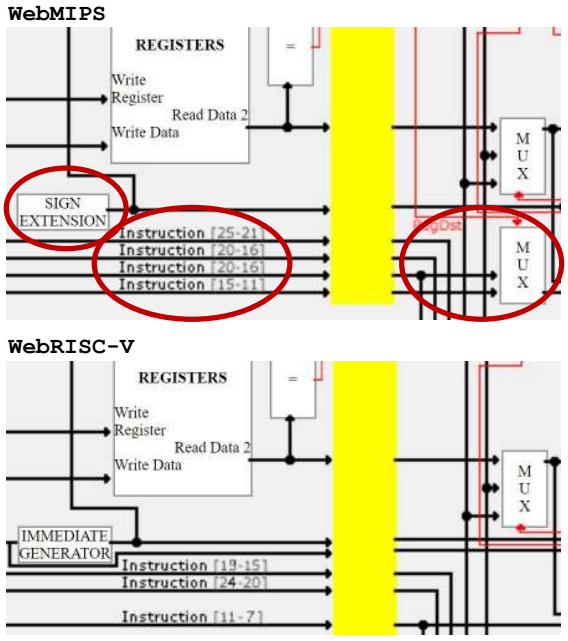

Figure 10: WebMIPS and WebRISC-V pipelines schematic comparison: the WebRISC-V has some simplification for the Instruction fields and there is no need for the selection of the actual register destination.

format was reworked to have only RD as the instruction destination register. Another modification is the substitution of the Sign Extension unit with an Immediate Generation unit, which gets the instruction as input and, while recognizing the instruction format, composes an appropriate immediate for a specific instruction type, as RISC-V has several kinds of configurations for the immediate field.

In addition, the user experience was improved by changing the visualization of some elements in the User Interface. For improved interactivity with the user, various options were added. One of them is a checkbox that makes it possible to dynamically see the content of specific elements by hovering over them with the mouse. Another one is a floating box that always shows the cycle count during the execution. Several other improvements are added to the WebRISC-V, compared to WebMIPS:

- a list of implemented instructions was added into the 'load program' page, to give students a table of easy examples on what arguments a specific instruction needs for its operation;
- for educational purposes, we show the "empty" slots of the pipeline at the beginning of the execution (pipeline fill up) and at the end of the execution (pipeline drain) (see also Figure 3);
- the cycle count and program name is always visible to improve the awareness of the context;
- the student can explore the values inside the pipeline by simply hovering the pointer on a specific architectural element (see Figure 9) (this saves two clicks to open and close a pop-up window - a feature still available for convenience); this feature improves the interactivity of the student with the pipeline exploration and avoids to create too much distraction that could be caused by displaying too much information at once.

Furthermore, a significant change to improve the usability is the Step-Back function, that allows the user to go back one step at anytime during execution, to better compare changes in specific points of the pipeline and facilitate the understanding. The user can can go back and forward to observe the specific changes at each cycle.

CONCLUSIONS

WebRISC-V is a web-based tool, which is based on server-side scripting (the well supported PHP scripting language), which means that it is highly portable on servers and can be used directly from any web browser without requiring any installation procedure on the client side. We plan to improve the graphical interface for an even better user experience and complete the mapping of a few other RISC-V instructions. However, the current set already allows the user to test any algorithm translated to assembly. The WebRISC-V simulator is already available and usable for most of the needs in the Computer Architecture classes. The simulator can be tried at this URL: http://www.dii.unisi.it/~giorgi/WebRISC-V. The source code is also available as indicated in the introduction.

ACKNOWLEDGMENTS

We are grateful to the makers of WebMIPS [2], a MIPS pipeline simulator written in ASP, from which WebRISC-V took great inspiration. We would like to thank the anonymous reviewers for their helpful comments. This work has been partially supported by the European Commission under the AXIOM H2020 project (id. 645496), TERAFLUX (id. 249013), and HiPEAC (id. 779656).

REFERENCES

[1] I. Branovic, R. Giorgi, and C. Prete A. 2002. Web-based training on Computer Architecture: The case for JCachesim. In *IEEE Workshop on Computer Architecture Education (WCAE-02)*. Anchorage, AK, USA, 56–60. http://www.dii.unisi.it/~giorgi/papers/Branovic02a.pdf

[2] I. Branovic, R. Giorgi, and E. Martinelli. 2004. WebMIPS: A New Web-Based MIPS Simulation Environment for Computer Architecture Education.. In *IEEE Workshop on Computer Architecture Education (WCAE-04)*. Munich, Germany, 93–98. https://doi.org/10.1145/1275571.1275596

[3] M De Los Angeles Cifredo-Chacon, Angel Quiros-Olozabal, and Jose Maria Guerrero-Rodriguez. 2015. Computer architecture and FPGAs: A learning-by-doing methodology for digital-native students. *Computer Applications in Engineering Education* 23, 3 (2015), 464–470. https://doi.org/10.1002/cae.21617 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cae.21617

[4] Michael Clark. 2017. rv8: A High Performance RISC-V to x86 Binary Translator.

[5] Joao Carlos de Oliveira Quintas. 2017. *Mipster32: A 32 bit MIPS Simulator.* LAP LAMBERT Academic Publishing.

[6] F. Garcia-Carballeira, A. Calderon, S. Alonso-Monsalve, and J. Prieto Cepeda. 2019. WepSIM: an Online Interactive Educational Simulator Integrating Microdesign, Microprogramming, and Assembly Language Programming. *IEEE Transactions on Learning Technologies* (2019), 1–1. https://doi.org/10.1109/TLT.2019.2903714

[7] A. Gersnoviez, M. Brox, M. A. Montijano, J. A. Sújar, and C. D. Moreno. 2018. UCOMIPSIM 2.0: Pipelined MIPS Architecture Simulator. In *2018 XIII Technologies Applied to Electronics Teaching Conference (TAEE)*. 1–6. https://doi.org/10.1109/TAEE.2018.8476063

[8] H. Grunbacher and H. Khosravipour. 1996. WinDLX and MIPSim pipeline simulators for teaching computer architecture. In *Proceedings IEEE Symposium and Workshop on Engineering of Computer-Based Systems*. 412–417. https://doi.org/10.1109/ECBS.1996.494568

[9] M. T. Kabir, M. T. Bari, and A. L. Haque. 2011. ViSiMIPS: Visual simulator of MIPS32 pipelined processor. In *2011 6th International Conference on Computer Science Education (ICCSE)*. 788–793. https://doi.org/10.1109/ICCSE.2011.6028756

[10] James Larus. 2019. *QtSPIM.* http://spimsimulator.sourceforge.net/

[11] Tim Newsome. 2019. *Spike.* https://github.com/riscv/riscv-isa-sim

[12] Robin Nixon. 2012. *Learning PHP, MySQL, JavaScript, and CSS: A Step-by-Step Guide to Creating Dynamic Websites.* O'Reilly Media, Inc.

[13] B. Nova, J. C. Ferreira, and A. Araújo. 2013. Tool to support computer architecture teaching and learning. In *2013 1st International Conference of the Portuguese Society for Engineering Education (CISPEE)*. 1–8. https://doi.org/10.1109/CISPEE.2013.6701965

[14] David A. Patterson and John L. Hennessy. 2017. *Computer Organization and Design RISC-V Edition: The Hardware Software Interface* (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[15] Morten Borup Petersen. 2019. *Ripes.* https://github.com/mortbopet/Ripes

[16] Dustin Richmond, Michael Barrow, and Ryan Kastner. 2018. Everyone's a Critic: A Tool for Exploring RISC-V Projects. *2018 28th International Conference on Field Programmable Logic and Applications (FPL)* (2018), 260–2604.

[17] G. C. R. Sales, M. R. D. Araújo, F. L. C. Pádua, and F. L. Corrêa Júnior. 2010. MIPS X-Ray: A plug-in to MARS simulator for datapath visualization. In *2010 2nd International Conference on Education Technology and Computer*, Vol. 2. V2–32–V2–36. https://doi.org/10.1109/ICETC.2010.5529442

[18] A. Stojkovic, J. Djordjevic, and B. Nikolic. 2007. WASP: A Web-Based Simulator for an Educational Pipelined Processor. *International Journal of Electrical Engineering & Education* 44, 3 (2007), 197–215. https://doi.org/10.7227/IJEEE.44.3.1 arXiv:https://doi.org/10.7227/IJEEE.44.3.1

[19] Kenneth Vollmar and Pete Sanderson. 2006. MARS: An Education-oriented MIPS Assembly Language Simulator. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '06)*. ACM, New York, NY, USA, 239–243. https://doi.org/10.1145/1121341.1121415

[20] Andrew Waterman, Yunsup Lee, David A. Patterson, and Krste Asanović. 2014. *The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.0.* Technical Report UCB/EECS-2014-54. EECS Department, University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-54.html