SoftwareX 18 (2022) 101105

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

WebRISC-V: A 32/64-bit RISC-V pipeline simulation tool

Gianfranco Mariotti, Roberto Giorgi *

Department of Information Engineering and Mathematics, University of Siena, Via Roma 56, 53100 Siena, Italy

ARTICLE INFO

Article history:
Received 23 February 2021
Received in revised form 25 January 2022
Accepted 5 May 2022

Keywords:
Simulation environments
Interactive learning environments
Computer architectures
Pipeline computing

ABSTRACT

WebRISC-V is a web-based education-oriented tool, which permits the investigation of the pipelined execution of assembly programs according to the RV32IM and RV64IM specifications (32-bit or 64-bit RISC-V processor). The tool permits to evaluate and understand slow-downs in the execution due to pipeline stalls and further investigate the internal state of the pipeline architectural blocks (e.g., registers, memory, multiplexers, ALU).

The pipeline concept is illustrated in the vast majority of university courses in Computer Architecture, since it is the well established standard for implementing high-performance processors. However, the impact of pipelined execution is often underestimated or even unknown, whilst it represents a very important source for the speed-up of programs.

Several similar tools exist and are publicly available, but WebRISC-V is currently the first one that can be executed directly in a web-browser while displaying the cycle-by-cycle detailed pipeline execution for a RISC-V processor.

This paper describes WebRISC-V, compares it against other similar available tools and presents its features. An example of usage for investigating the pipeline with the support of an automatically generated pipeline diagram is provided.

Code metadata

Current code version	1.8.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-21-00043
Code Ocean compute capsule	
Legal Code License	BSD-3-Clause
Code versioning system used	git
Software code languages, tools, and services used	PHP, JavaScript, HTML, CSS
Compilation requirements, operating environments & dependencies	PHP, PHP GMP
If available Link to developer documentation/manual	
Support email for questions	wrv-admin@webriscv.dii.unisi.it

1. Motivation and significance

In the Computer Science [1,2] and Computer Engineering [2] curricula, Computer Architecture (i.e., Architecture and Organization) is recognized as one of the core requirements.

A common problem in teaching Computer Architecture courses is how to help students understand abstract concepts of processor architecture and complex techniques of hardware organization. This is especially notable when teaching the architecture of computers to undergraduate students, as experience shows that they may find difficulties in understanding processor logic and implementation [3].

In order to help understand the internal of the processors, active learning, e.g., taking advantage of advanced software technology like simulators, has been shown an effective teaching technique [4]. Compared to real hardware, simulators are less expensive and more flexible as they are easy to set up, use, modify and even provide access to details sometimes not practically accessible. Considering traditional teaching materials, such as schematic diagrams and text descriptions, simulation can significantly improve students learning and assist instructors in teaching [5–7].

In a pipelined processor, the simultaneous execution of multiple instructions in the different stages produces situations that

* Corresponding author.
E-mail addresses: mariotti@dii.unisi.it (Gianfranco Mariotti), giorgi@unisi.it (Roberto Giorgi).

https://doi.org/10.1016/j.softx.2022.101105

cannot occur in a non pipelined processor, such as hazardous situations that impede a smooth flow through pipeline stages. Software simulation enables practical insights into these situations that cannot be visualized conveniently on real processors. Furthermore, it allows for a high level of individualization, therefore providing significant help in learning efforts. Once the concepts are demonstrated, students can progress at a pace that suits them and practice until they are sure to have mastered the topic.

This paper briefly describes WebRISC-V [8], a web-based RISC-V pipeline simulation environment, compares it with similar tools, and illustrates a detailed example of a simple program that can be accelerated through the pipeline forwarding technique. Other basic features have been introduced in [9].

WebRISC-V is based on the RISC-V Instruction Set Architecture, a high quality modern, free and open ISA standard, in contrast to other commercially popular ISAs that are proprietary [10]. RISC-V is also becoming popular in Computer Architecture courses as a substitute of other ISAs [11]. From a design point of view, the RISC-V avoids the technical pitfalls of other ISAs and is straightforward to implement in many microarchitectural styles [12]. RISC-V has the potential to increase innovation in microprocessor design, reduce computer system cost and, as Moore's law wanes, ease the transition to more specialized computational devices. Initially designed for research in the academic field, it is on its way to become a widely accepted industrial standard for microprocessors [13], and multiple hardware implementations are currently available [14].

Until recently, the most used architecture in the academic field was MIPS, in order to achieve effective and easy learning. With the rise in popularity and adoption of RISC-V, educational tools are starting to appear to explore this ISA [15].

Educational software can allow students to learn assembly programming and study machine language by executing programs and observing run-time changes of the memory and the internal elements of the processor.

In the following, we briefly compare several educational pipeline simulators that are available nowadays.

1.1. A comparison of computer architecture pipeline simulators

PBSE [16] (MARS [17] plug-in), MIPS X-ray [18] (MARS plug-in), DrMIPS [19], Mipster32 [20], UCOMIPSIM [21], Visimips [22], WASP [23] and WebMIPS [24] are tools for the MIPS ISA; Ripes [25] and WebRISC-V are similar tools, but supporting the RISC-V ISA.

We report in Table 1 an analysis on datapath implementation and simulation environment characteristics of these tools. The simulators are grouped and identified by the columns:

- ISA: which ISA the tool supports;
- Simulation Tool: how the educational software is named.

The categories through which the datapath simulators were analyzed are the following:

- Graphical Presentation: how the micro-architecture of the processor is shown to the user by the tool.
  Typically, these simulators fall in one of these three cases:
  1. *Stage Blocks*: a reduced datapath view, exposing only blocks that represent the phases of the pipeline (Fetch, Decode, Execute, Memory and Write-Back);
  2. *Datapath*: a simplified model, without control units and signals;
  3. *Datapath with Control Unit*: a complete datapath model.
- Visualization of Architectural Elements: whether the simulator allows you to display the details of the datapath architectural elements;
- Console I/O Syscalls: whether the tool permits both input and output system calls (I/O), only output ones (O), or none;
- Datapath Implementation: whether the implementation of the processor is single cycle (i.e, without pipelining), pipelined or permits to choose between these two;
- Web Accessibility: specifies if the tool has online accessibility for the convenience of the users. Also specifies if the tool requires plug-ins or is directly usable without any additional software;
- Arch: whether the standard variable format of the simulator's architecture is 32-bit wide, 64-bit wide, or both.

Having described RISC-V growing importance, in our analysis we have considered in more detail the current versions of WebRISC-V and Ripes simulators. As of our knowledge, these two simulators are the only ones that visually expose the datapath of the RISC-V processor for educational purpose. WebRISC-V and Ripes have in-depth focus on different aspects of the processor. Compared to WebRISC-V, Ripes adds facilities to explore cache behavior and how to use memory mapped I/O. Instead WebRISC-V provides more facilities than Ripes for the exploration of the details of the pipelined datapath. For example, it models the so called *branch delay-slot* [26] and permits students to experiment with code-reordering for taking advantage of such-delay slot, whereas this feature is not available in Ripes. WebRISC-V also provides other educational features missing on Ripes, such as explanations for RISC-V Assembly instruction and directives, descriptions and details for high-level components of the datapath and special features on the pipeline table for easier comparison with pencil-and-paper approaches.

2. Software description

Here, we outline the main features of WebRISC-V, a simulation environment useful to facilitate students in investigating the parallel execution of instructions in the different stages of a pipelined processor.

It includes facilities to edit assembly programs and view memory contents, registers, basic architectural elements and their internal state. The user can run interactive simulations through a step-by-step execution of the instructions and observe the changes of the architectural state. Also, the user can replay instructions in order to better detect an interesting event. WebRISC-V includes the simulation of structural, control and data hazard; also, it permits to investigate the forwarding and hazard-detection units, it presents the schematic with or without forwarding and can show the execution in both these two cases. Separately, the simulator can produce a pipeline diagram through a compact view of the active stages at each cycle.

WebRISC-V is a web application. Being a client–server application, it provides advantages and disadvantages compared to native applications. Native applications are developed for running on a particular platform, and are installed directly on such devices for usage. Developers therefore need to create separate versions for each supported platform. Client–server applications are instead developed and executed on a single environment, with no installation required on user machines and upgrades of the software only performed on server side and immediately available to all users. Therefore this makes for easier testing and support compared to the multitude of operating system and hardware configurations that a native application will be installed on, with the drawback of needing network connectivity. In the case of WebRISC-V, we minimize this drawback by releasing the

Table 1
Comparison of datapath simulators.

ISA	Simulation tool	Graphical presentation	Visualization of architectural elements	Console I/O Syscalls	Datapath implementation		Web accessibility		Arch
					single cycle	pipelined	client side[a]	server side[b]	
MIPS	PBSE	Stage blocks		I/O		✓			32-bit
	MIPS X-RAY	Datapath		I/O	✓				32-bit
	DrMIPS	Datapath with control unit	✓		✓	✓			32-bit
	Mipster32	Stage blocks				✓			32-bit
	UCOMIPSIM	Datapath with control unit	✓			✓			32-bit
	Visimips	Datapath with control unit				✓			32-bit
	WASP	Datapath with control unit	✓			✓	✓		32-bit
	WebMIPS	Datapath with control unit	✓			✓		✓	32-bit
RISC-V	RIPES	Datapath with control unit		O	✓	✓			32/64-bit
	WebRISC-V	Datapath with control unit	✓	I/O		✓		✓	32/64-bit

[a]Additional plug-ins required.
[b]Accessible directly from the Web without the use of plug-ins.

open-source code, giving the possibility to spin up your own local server instance, like the installation of a native application.

To validate the correctness of this software tool we have used a pencil-and-paper approach related to several textbook examples [26] that were also assigned as exam exercises for the Computer Architecture course. The outcome from the simulations have been validated regarding several aspects, including the correctness of the data, the correctness of the pipeline diagram with and without forwarding, and the final clock cycle count.

2.1. Software architecture

WebRISC-V has its back-end written in PHP and its front-end in HTML, CSS and JavaScript [27]. Being a server-side web application, it is installed and executed on a web server and presented to the user on their client interface. If the teaching staff wants a local installation, an instance can be hosted with a simple procedure on a Linux or Windows server.

This simulator supports the full implementation of four RISC-V 'modules'[1] as they are described in the RISC-V ISA unprivileged specification [28]: the 32-bit and 64-bit Base Integer ("fence" instruction excluded) modules – identified as 'RV32I' and 'RV64I' – and their Extensions for Integer Multiplication and Division module — identified as 'M', therefore making the tool run according to the 'RV32IM' and 'RV64IM' specifications.

Voluntarily, for easier student's reference, WebRISC-V resembles very closely the schematic used in the Patterson/Hennessy book [26], in which the pipelined datapath implementation is explored and explained.

2.2. Software functionalities

WebRISC-V provides:

- The visualization of the complete architectural schematic of a pipelined RISC-V processor (see Fig. 1);
- The ability of stepping forward and backwards in the execution of code, to better study what is happening inside the pipeline and its elements;
- On a single page view, the monitoring of the information about the current processing state (e.g., cycle count, colored tags to indicate the current stage of an instruction, highlighting of eventual 'bubbles' in the pipelined execution);
- A descriptive explanation of each internal element together with its current state, that can be shown by simply hovering with the mouse (see Fig. 2);
- The ability of simulating multiple hazard resolution modes of the pipeline;
- The possibility of enabling (see Fig. 1a) or disabling (see Fig. 1b) the data forwarding units (with automatic visualization of the corresponding schematic);
- The possibility of visualizing the memory segment contents (Text Segment, Static Data Segment, Dynamic Data Segment) and the registers, as shown in Fig. 4;
- An online editor, with some built-in examples, and a contextually visible full list of the available instructions and directives (see Fig. 3);
- Automatic generation of the classic pipeline diagram (see Fig. 5a); in case of loops, this diagram can be automatically squashed (see Fig. 5b);
- Basic I/O system calls of the simulated RISC-V are made possible by prompting the user via a popup window, which emulates the system console.

In Fig. 2, it is possible to see the simulator in action during cycle-by-cycle execution. The current clock cycle is always visible, together with the state of the stages: empty stages (i.e., stages that are not currently busy executing instructions) and stall 'bubbles' (i.e., waiting cycles due to hazards) are also clearly visible.

In this schematic, the stage buffers are shown, each with a specific color (pink, red, yellow, blue, green respectively for the Fetch, Decode, Execute, Memory Access and Write-Back stages). These colors are used also to highlight the instructions that are being processed, on the left side; those instructions are also visible on top of the architectural schematic, with the same colors, to have a more direct connection with the corresponding stage. By activating the corresponding visualization option, as in the figure, it is possible to see a description and the state of an architectural element that appears on top of the schematic by simply hovering the mouse on such element.

[1] In a few words, a RISC-V 'module' is a subset of instructions.

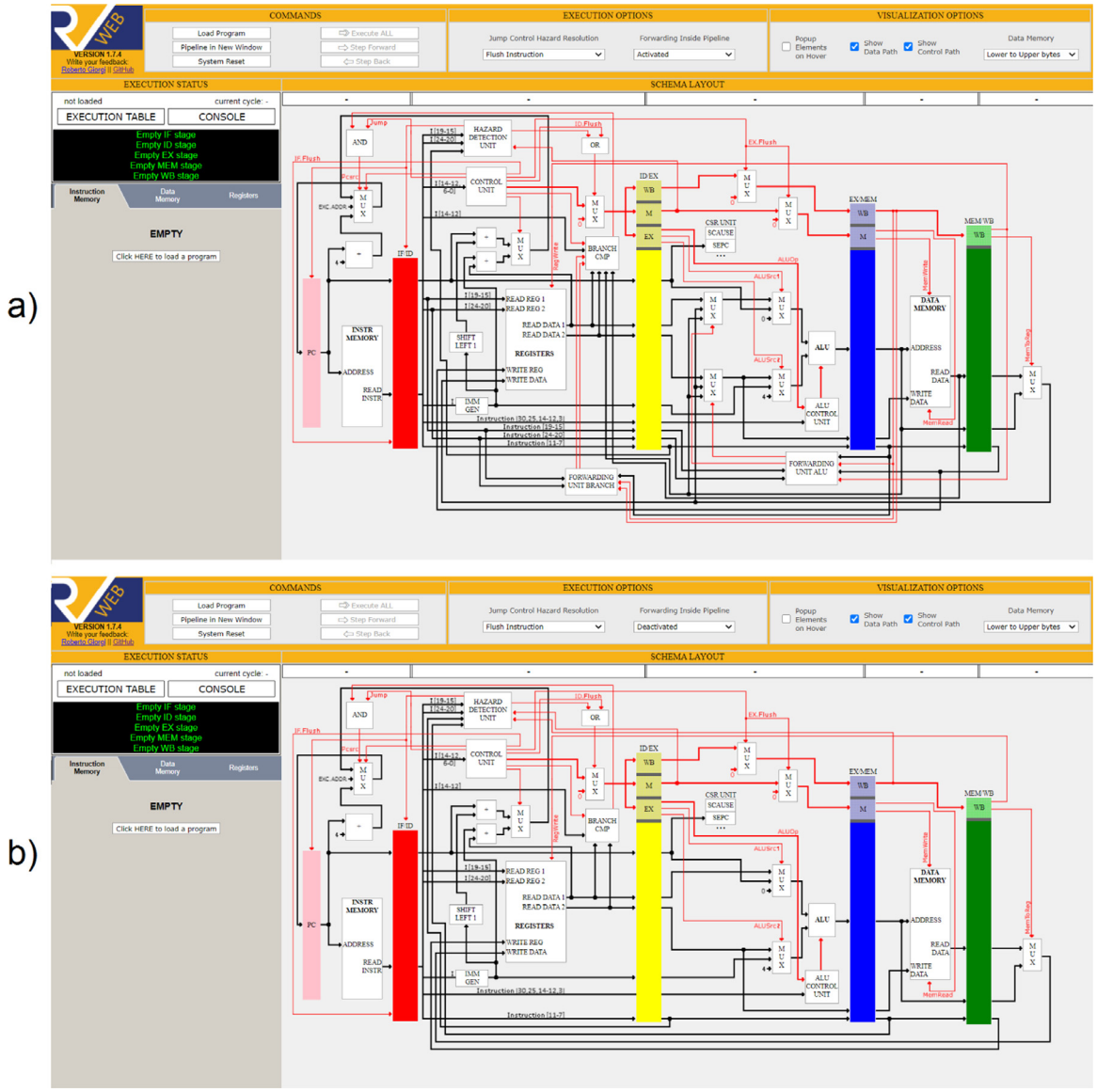

Fig. 1. WebRISC-V main page: Schematic (a) with and (b) without Forwarding.

3. Illustrative examples

Several built-in examples are immediately available to the user for studying specific situations or to become familiar with simple assembly codes.

Here, we present a detailed example of a simple exercise (see Listing 1), that is used for training students at our department by using WebRISC-V [29].

Listing 1: Forwarding Example

```
.text
    addi x12, x0, 2
    addi x10, x3, 8
loop:
    beq x10, x3, fine
    lw x5, 100(x10)
    add x5, x5, x12
    sw x5, 200(x10)
    j loop
    addi x10, x10, -4
fine:
    addi x0, x0, 0
```

This exercise helps illustrate the functioning of hazard detection and forwarding in the pipeline. The user/student can compare the scenario when forwarding is enabled (leading to faster execution) or without it, as a further experimentation option.

It also makes use of the branch *delay-slot* [26] during the execution, for didactic value. The delay-slot is the ability of fetching and executing one instruction after a branch, as if it was before that branch in program order, is an important feature of a pipelined processor, as it may lead to a faster execution. This concept is typically surprising for first-time users as the instructions are executed in a different order than what was originally specified.

The UI is quite straight-forward for usage of the simulator (see Fig. 1). The starting point is loading a program ("Load Program" button): the user can then write an assembly program into the editor textbox or select one of the built-in examples.

Once ready, the user is guided to follow the same process that is done by a computer before running a program, (a step which is typically invisible, but which is important to understand the 'magic' behind the program execution) by pressing the button "Load into memory". If the program is written correctly (otherwise the error is indicated), the visualization comes back to the schematic and the user can observe the cycle-by-cycle details or

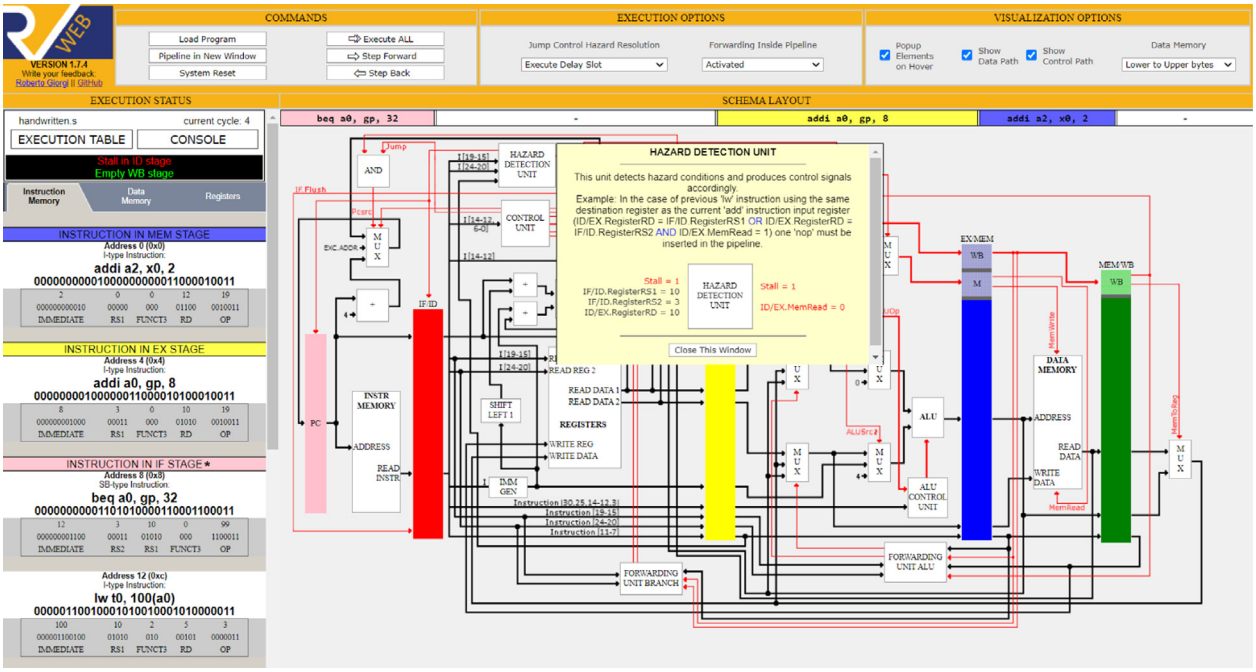

Fig. 2. The case of a stall: A star indicates which instruction has generated the stall. The state of the Hazard Detection Unit is shown.

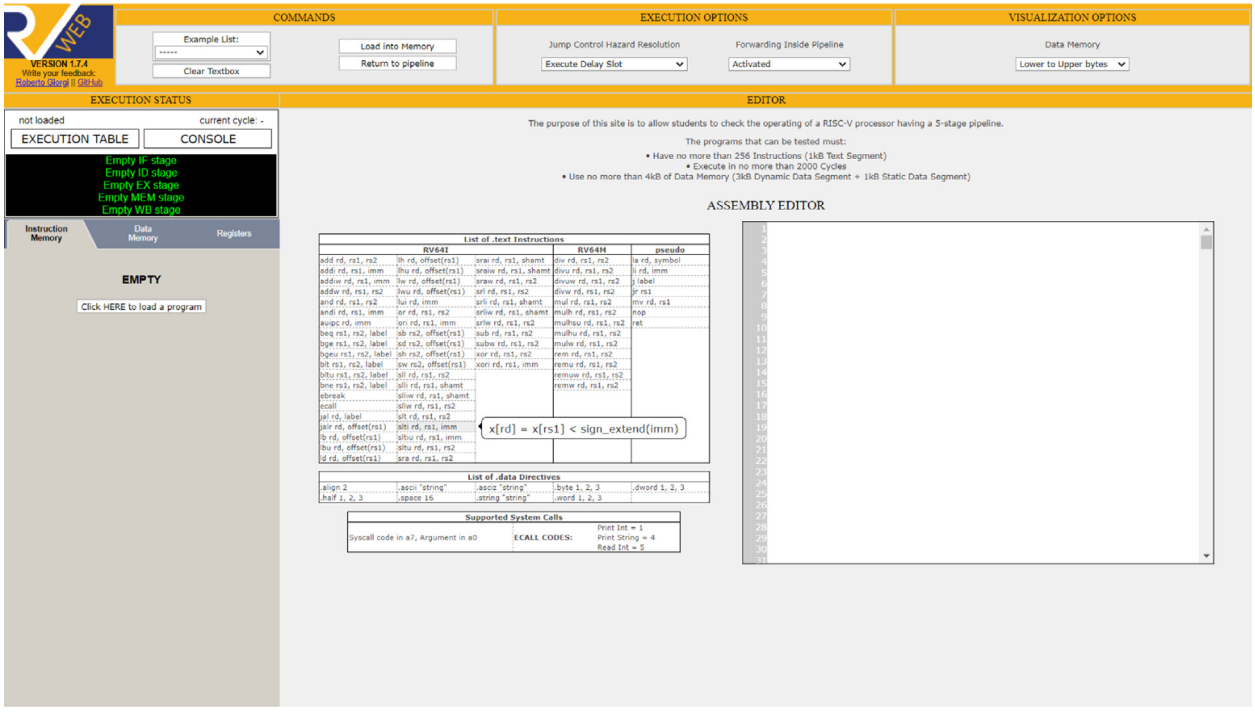

Fig. 3. WebRISC-V assembly editor page: Interactive list of instructions and directives, with explanations of their meaning and usage (and encoding in case of pseudo-instructions).

execute the program at-once until the end. On the left panel, the user can select to observe the instructions flow, the content of the memory, or the content of the registers, as shown in Fig. 4.

During the step-by-step execution of the code, we can detect eventual 'bubbles' generated by stalls inside the pipeline (see Fig. 2).

Another relevant visualization tool is the *pipeline diagram*, which is obtained by pressing the "Execution Table" button. Sometimes, typically due to the presence of loops, the diagram could be too large to fit into the screen and it is anyway difficult to read. Therefore, we introduced a practical way to visualize the 'loop case' by automatically squashing the cycles of the loop in the diagram, but without losing the precise accounting of the executed cycles. In Fig. 5 both versions of the pipeline diagram are shown.

Until now, the user executed the code with forwarding activated: to see the difference in performance, the user can test the program without forwarding. The user can compare the cycles that are necessary for the execution of the code in the these two different situations (with and without forwarding, see Table 2).

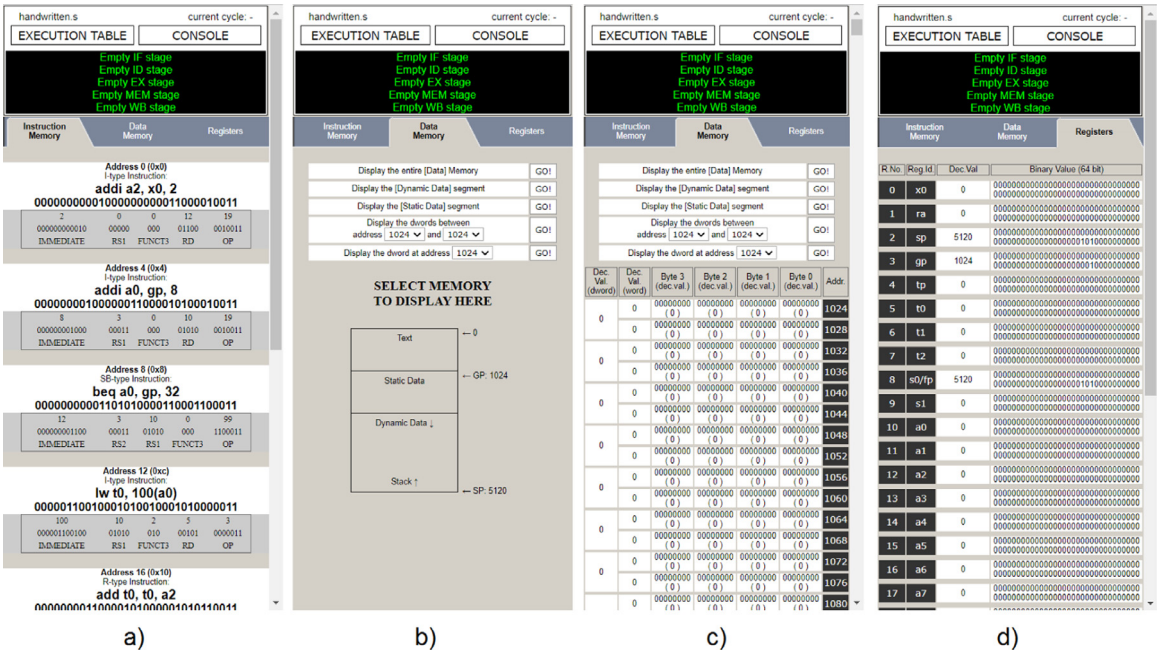

Fig. 4. WebRISC-V views of the left panel: (a) Instruction Memory, Data Memory (b) before and (c) after selecting the Memory Segment to display, (d) Registers.

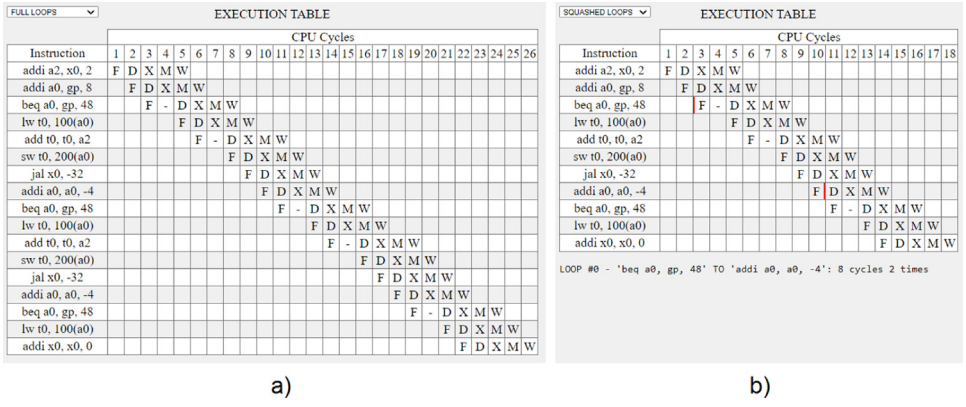

Fig. 5. Pipeline diagram in both visualization methods: (a) 'Full loops' and (b) 'squashed loops'. In (b), we can see a squashed diagram where the 'condensed loop' is marked with two red vertical bars from cycle 3 to cycle 10; also an automatically generated note under the diagram ("LOOP #0 ...") indicates precisely how many cycles are taken from the single loop iteration and how many times this loop is repeated. Through the pipeline diagrams it is easier to analyze the flow of executed instructions (F = Fetch, D = Decode, X=eXecute, M = Memory access, W = Write-back, '-' = pipeline bubble).

Table 2
Execution results.

Pipeline	Cycles
With forwarding	26
Without forwarding	35

As can be seen there is a significant speed-up when executing instructions with the forwarding option.

The user can observe the effect of the forwarding units:

- the result of the sum ("addi") propagates to stage D, where the comparison between the two operands is made for the branch decision ("beq");
- the result of the memory read ("lw") propagates to stage X, to perform the sum ("add");
- the result of the sum ("add") propagates to stage M, in which the data memory is accessed to perform a write operation ("sw");
- the result of the sum ("addi") propagates to stage D; this functional dependence is generated by the delay-slot, active on the jump instruction ("j loop").

It is possible to compare the pipeline diagrams on both kinds of execution to explore the results.

4. Impact

The plausibly most popular undergraduate textbook in computer architecture, having a readership of over one hundred thousand students a year [30] and having been used around the world from the 1990s [31], is "Computer Organization and Design" by D. A. Patterson and J. L. Hennessy [26]. This book introduces to student the basics of modern processors, which are based on a pipelined micro-architecture as offered by WebRISC-V.

Courses in computer architecture have been shifting to include the RISC-V ISA, or to migrate to it from other ISAs such as MIPS, following the growth of RISC-V popularity in both academic research, teaching and industry [10,11,13].

The pipeline concept is a well established standard for implementing high-performance processors, for it represents a significant source of performance speed-up widely used in current processors, therefore being a very significant concept in Computer Architecture [26]. Consequently it is quite important that students of that course learn this concept in an easy and reliable way.

In the classroom pipeline scheduling problems are usually handled by the pencil-and-paper method and are neither small nor easy to solve in most of cases, this being the reason why the amount and the versatility of the exercises that both the professor and students can solve are limited. This might be the reason that explains why students are not confronted with a great set of different examples and problems, which may bring them to not understand all the possible situations that could happen in the scheduling of a pipeline.

Thus WebRISC-V, a simulator that provides an easy and fast way to solve or double-check such exercises, provides a notable contribution as a didactic tool for students.

This software is used as didactic tool in the lessons of Computer Architecture of our department and is offered as one the educational material that is available on the official RISC-V website [15].

The open-source nature of WebRISC-V makes the software readily accessible for contributions seeking to enhance its capabilities. In the foreseeable future, we hope that a community of users can grow around WebRISC-V, so that the tool can improve even more in terms of both features and user-friendliness.

5. Conclusions

WebRISC-V is available online to be used for educational reasons in Computer Architecture classes at the following URL: https://webriscv.dii.unisi.it. The availability of this kind of tools help understanding the hidden details of a pipelined processor and how the performance of the software can be enhanced by appropriate optimizations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially supported by the European Commission under the AXIOM H2020 project (id. 645496), TERAFLUX (id. 249013), and HiPEAC (id. 871174)

References

[1] Joint Task Force on Computing Curricula AfCMA, Society IC. Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science. New York, NY, USA: Association for Computing Machinery; 2013.
[2] Force CT. Computing curricula 2020. 2020, https://cc2020.net.
[3] Lakshminarayanan D. Learning difficulties in computer architecture. COMPUSOFT: Int J Adv Comput Technol 2016. URL https://ijact.in/index.php/ijact/article/view/451.
[4] Freeman S, Eddy SL, McDonough M, Smith MK, Okoroafor N, Jordt H, et al. Active learning increases student performance in science, engineering, and mathematics. Proc Natl Acad Sci 2014;111(23):8410–5. http://dx.doi.org/10.1073/pnas.1319030111, arXiv:https://www.pnas.org/content/111/23/8410.full.pdf, URL https://www.pnas.org/content/111/23/8410.
[5] Prasad PWC, Alsadoon A, Beg A, Chan A. Using simulators for teaching computer organization and architecture. Comput Appl Eng Educ 2016;24(2):215–24. http://dx.doi.org/10.1002/cae.21699, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cae.21699, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cae.21699.
[6] Mustafa B. Modern computer architecture teaching and learning support: An experience in evaluation. In: International conference on information society. 2011, p. 411–6. http://dx.doi.org/10.1109/i-Society18435.2011.5978481.
[7] Alnoukari M. Simulation for computer sciences education. Commun ACS 2013;6. URL https://www.researchgate.net/publication/249313265_Simulation_for_Computer_Sciences_Education.
[8] Mariotti G, Giorgi R. WebRISC-V. 2019, https://github.com/Mariotti94/WebRISC-V.
[9] Giorgi R, Mariotti G. WebRISC-V: A web-based education-oriented RISC-V pipeline simulation environment. In: Proceedings of the workshop on computer architecture education. WCAE'19, New York, NY, USA: ACM; 2019, p. 3:1–6. http://dx.doi.org/10.1145/3338698.3338894, URL http://doi.acm.org/10.1145/3338698.3338894.
[10] Greengard S. Will RISC-V revolutionize computing? Commun ACM 2020;63:30–2. http://dx.doi.org/10.1145/3386377.
[11] Zhang K. Evolution and revolution of computer systems courses with the open RISC-V ISA. In: CompEd '19: Proceedings of the ACM conference on global computing education. 2019, p. 171. http://dx.doi.org/10.1145/3300115.3312506.
[12] Waterman A. Design of the RISC-V instruction set architecture (Ph.D. thesis), EECS Department, University of California, Berkeley; 2016, URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-1.html.
[13] Corporation SR. RISC-V market analysis the new kid on the block (cc315-19 ed.). 2019, https://semico.com/content/risc-v-market-analysis-new-kid-block.
[14] Doerflinger A, Albers M, Kleinbeck B, Guan Y, Michalik H, Klink R, et al. A comparative survey of open-source application-class RISC-V processor implementations. In: CF '21: Proceedings of the 18th ACM international conference on computing frontiers. 2021, p. 12–20. http://dx.doi.org/10.1145/3457388.3458657.
[15] RISC-V educational materials. 2020, https://riscv.org/community/learn/educational-materials/.
[16] Lim DX, Smitha KG. Pipelined MIPS simulation: A plug-in to MARS simulator for supporting pipeline simulation and branch prediction. In: 2019 IEEE international conference on engineering, technology and education. 2019, p. 1–7.
[17] Vollmar K, Sanderson P. MARS: An education-oriented MIPS assembly language simulator. In: Proceedings of the 37th SIGCSE technical symposium on computer science education. SIGCSE '06, New York, NY, USA: ACM; 2006, p. 239–43. http://dx.doi.org/10.1145/1121341.1121415, URL http://doi.acm.org/10.1145/1121341.1121415.
[18] Sales GCR, Araújo MRD, Pádua FLC, Corrêa Júnior FL. MIPS X-RAy: A plug-in to MARS simulator for datapath visualization. In: 2010 2nd International conference on education technology and computer, Vol. 2. 2010, p. V2–32–V2–36. http://dx.doi.org/10.1109/ICETC.2010.5529442.
[19] Nova B, Ferreira JC, Araújo A. Tool to support computer architecture teaching and learning. In: 2013 1st International conference of the Portuguese society for engineering education. 2013, p. 1–8. http://dx.doi.org/10.1109/CISPEE.2013.6701965.
[20] de Oliveira Quintas JC. Mipster32: A 32 bit MIPS simulator. LAP LAMBERT Academic Publishing; 2017.
[21] Gersnoviez A, Brox M, Montijano MA, Sújar JA, Moreno CD. UCOMIPSIM 2.0: Pipelined MIPS architecture simulator. In: 2018 XIII Technologies applied to electronics teaching conference. 2018, p. 1–6. http://dx.doi.org/10.1109/TAEE.2018.8476063.
[22] Kabir MT, Bari MT, Haque AL. ViSiMIPS: Visual simulator of MIPS32 pipelined processor. In: 2011 6th International conference on computer science education. 2011, p. 788–93. http://dx.doi.org/10.1109/ICCSE.2011.6028756.
[23] Stojkovic A, Djordjevic J, Nikolic B. WASP: A web-based simulator for an educational pipelined processor. Int J Electr Eng Educ 2007;44(3):197–215. http://dx.doi.org/10.7227/IJEEE.44.3.1.
[24] Branovic I, Giorgi R, Martinelli E. WebMIPS: A new web-based MIPS simulation environment for computer architecture education. In: IEEE workshop on computer architecture education. Munich, Germany; 2004, p. 93–8. http://dx.doi.org/10.1145/1275571.1275596, URL http://www.dii.unisi.it/~giorgi/papers/Branovic04a.pdf.
[25] Petersen MB. Ripes. 2019, https://github.com/mortbopet/Ripes.
[26] Patterson DA, Hennessy JL. Computer organization and design RISC-V edition: The hardware software interface. 1st ed.. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 2017.
[27] Nixon R. Learning PHP, MySQL, JavaScript, and CSS: A step-by-step guide to creating dynamic websites. O'Reilly Media, Inc.; 2012.
[28] Waterman A, Asanovic K. The RISC-V instruction set manual, volume I: Unprivileged ISA. 2019, https://riscv.org/specifications/isa-spec-pdf/.
[29] Mariotti G, Giorgi R. Understanding the advantage of forwarding logic. 2020, https://arcal.dii.unisi.it/lab-webriscv.htm.
[30] Patterson DA, Hennessy JL. Computer organization and design RISC-V edition: The hardware software interface. 2021, https://www.elsevier.com/books/computer-organization-and-design-risc-v-edition/patterson/978-0-12-820331-6.
[31] Clements A. The undergraduate curriculum in computer architecture. IEEE Micro 2000;20(3):13–21. http://dx.doi.org/10.1109/40.846305.