WebRISC-V: A RISC-V Educational Simulator featuring RV64IM, Pipeline and Web-Based UI

Gianfranco Mariotti*,[1],
Roberto Giorgi*,[2]

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

ABSTRACT

WebRISC-V is a web-based education-oriented RISC-V pipeline simulation environment, which aims at aiding students learning and easing instructors teaching experience. This simulator permits the execution of RISC-V user-provided source code on a five stage pipeline, making it easy to trace the instruction flow while displaying the data of registers, memory and the internal state of the pipeline elements. Here we introduce WebRISC-V and why it is useful as a didactic tool, present its latest feature updates and briefly compare the simulator against the other available RISC-V educational tools.

KEYWORDS: Simulation Environments; Interactive Learning Environments; Computer Architectures; Pipeline Computing

1 Introduction

One of the main objectives of computer engineering education is to train students to understand the limits and possibilities of computers. Traditionally, this is done with a demonstration method, in which the teacher illustrates concepts on a blackboard. Individualization of teaching is difficult to achieve when using this method, because the teacher has limited time for demonstration and students usually have very heterogeneous knowledge. Software simulation allows for a high level of individualization and can provide significant help in training students, since once the concepts are demonstrated, students can progress at a pace that suits them, can repeat the demonstration several times, and can practice until they are sure to have mastered the argument. In fact, studies [Aln13, Mus10] attest that simulation tools can be used as an effective means of teaching or demonstrating concepts to students, especially if these tools are easily accessible.

Here we present a simulation environment useful to facilitate students in studying and investigating the reasons for the good performance of a pipelined processor, which also makes it easy to examine its basic architectural elements and their internal state.

[1] E-mail: mariotti@dii.unisi.it

[2] E-mail: giorgi@dii.unisi.it

WebRISC-V[3] is a simulation environment accessible via web for the study of the RISC-V processor pipeline in accordance to the book ”Computer Organization and Design: RISC-V Edition” by D. A. Patterson and J. L. Hennessy [PH17]. It takes inspiration from the old WebMIPS [BGM04] simulator, recovering its functionality and extending it, replicating the user experience in the new RISC-V context. WebRISC-V simulates the execution of instructions in the five stages of the pipeline, including the possibility of studying the behavior of the hazard detection and forwarding units and how they solve execution problems.

WebRISC-V was developed and released as open source under the permissive BSD license on GitHub [MG19] and is still being actively developed. It is possible to test the latest version of the simulator online at `http://www.dii.unisi.it/~giorgi/WebRISC-V`

A paper on the initial released version of WebRISC-V was presented in Phoenix, AZ, USA at the WCAE @ ISCA 2019, it is Open Access and available on the ACM digital library [GM19]. Moreover a poster of a previous version of WebRISC-V was shown at the HiPEAC 2020 conference.

2 Features

WebRISC-V [MG19] has had significant improvements over the initial version [GM19]:

- full implementation of RV64IM ("fence" instruction excluded);
- ability of simulating the pipeline with and without forwarding;
- automatic creation of the classic ”pipeline diagram” (see figure 1);
- console for managing some of the ecalls for I/O;
- additional info of the executed instruction directly on the pipeline schematic.

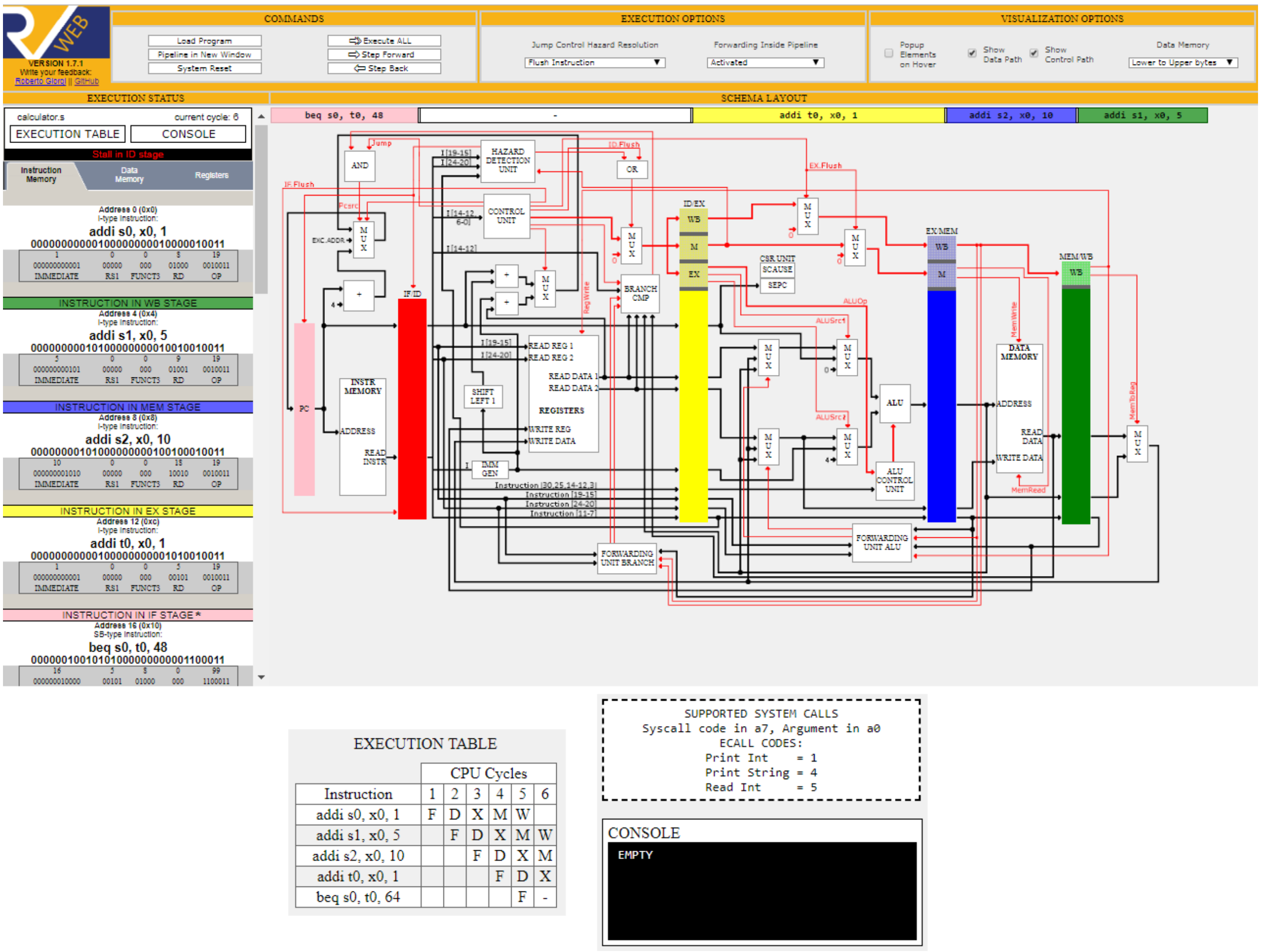
For easier tracking of the program flow, an execution table was implemented and now the instructions currently executing inside the pipeline stages are also shown on the schema. To permit system call ("ecall" instruction) interaction a console was also implemented. The switching of the branch delay slot is now permitted and relative examples of the necessary code changes are present. The forwarding logic can now be excluded from both the schematic and simulation, so the execution is now possible with two different pipeline schemas, with or without forwarding. The simulator now has an updated data memory model, and presents the relative basic .data directives. Before, the simulator focused on a subset of the ISA big enough for the execution of most common algorithms, but now it has been updated and implements the whole RV64I (except the "fence" instruction) and RV64M ISA modules, as they are described in the RISC-V ISA unprivileged specification [WA19]. Furthermore on the editor page is it now possible to find a list of the instructions, their necessary arguments and their meaning.

[3] This work has been partially supported by the European Commission under the AXIOM H2020 project (id. 645496), TERAFLUX (id. 249013), and HiPEAC (id. 779656)

3 RISC-V Educational Tools

Here we briefly compare educational tools, the category that WebRISC-V belongs to. The available didactic RISC-V tools are the ISA simulators Jupiter [Cas18], RARS [Lan17] and Venus [VK18], the ISA simulator and RTL explorer BRISC-V [ABE+19] and the ISA and pipeline simulators Ripes [Pet19] and WebRISC-V.

WebRISC-V is the only simulator belonging to this list that supports the 64-bit RISC-V instruction set. Ripes is the only other pipeline simulator and shares many similarities with WebRISC-V in features, but only supports the 32-bit instruction set and isn't web-based. Of these tools, the only ones other than WebRISC-V that have at least a portion of them on the web and are as such readily accessible are BRISC-V and Venus.



Jump Control Hazard Resolution: Flush Instruction

Forwarding Inside Pipeline: Activated

☐ Popup Elements on Hover ☑ Show Data Path ☑ Show Control Path

Data Memory: Lower to Upper bytes

calculator.s current cycle: 6

Stall in ID stage

beq s0, t0, 48 | - | addi t0, x0, 1 | addi s2, x0, 10 | addi s1, x0, 5

Address 0 (0x0)
I-type Instruction:
addi s0, x0, 1
00000000000100000000010000010011

1	0	0	8	19
000000000001	00000	000	01000	0010011
IMMEDIATE	RS1	FUNCT3	RD	OP

INSTRUCTION IN WB STAGE
Address 4 (0x4)
I-type Instruction:
addi s1, x0, 5
00000000010100000000010010010011

5	0	0	9	19
000000000101	00000	000	01001	0010011
IMMEDIATE	RS1	FUNCT3	RD	OP

INSTRUCTION IN MEM STAGE
Address 8 (0x8)
I-type Instruction:
addi s2, x0, 10
00000000101000000000100100010011

10	0	0	18	19
000000001010	00000	000	10010	0010011
IMMEDIATE	RS1	FUNCT3	RD	OP

INSTRUCTION IN EX STAGE
Address 12 (0xc)
I-type Instruction:
addi t0, x0, 1
00000000000100000000001010010011

1	0	0	5	19
000000000001	00000	000	00101	0010011
IMMEDIATE	RS1	FUNCT3	RD	OP

INSTRUCTION IN IF STAGE *
Address 16 (0x10)
SB-type Instruction:
beq s0, t0, 48
00000010010101000000000001100011

16	5	8	0	99
000000010000	00101	01000	000	1100011

EXECUTION TABLE

	CPU Cycles					
Instruction	1	2	3	4	5	6
addi s0, x0, 1	F	D	X	M	W	
addi s1, x0, 5		F	D	X	M	W
addi s2, x0, 10			F	D	X	M
addi t0, x0, 1				F	D	X
beq s0, t0, 64					F	-

```
        SUPPORTED SYSTEM CALLS
Syscall code in a7, Argument in a0
            ECALL CODES:
          Print Int    = 1
          Print String = 4
          Read Int     = 5
```

CONSOLE

```
EMPTY
```

Figure 1: The WebRISC-V simulator during the execution of the example code "Simple Calculator". We can see here the main, execution table and console pages. In the main page, on the left panel you can display the instruction memory, the data memory or the registers. Each pipeline stage has a different color (pink, red, yellow, blue, green) for easier recognition. As can be seen, the instructions get colored by the stage in which they are processed and also appear on top of the schema. The black and red wires represent respectively the data path and control path wires.

References

[ABE$^+$19] Rashmi Agrawal, Sahan Bandara, Alan Ehret, Mihailo Isakov, Miguel Mark, and Michel Kinsy. The BRISC-V Platform: A Practical Teaching Approach for Computer Architecture. pages 1–8, 06 2019.

[Aln13] Mouhib Alnoukari. Simulation for Computer Sciences Education. *Communications of the ACS*, 6, 01 2013.

[BGM04] I. Branovic, R. Giorgi, and E. Martinelli. WebMIPS: A New Web-Based MIPS Simulation Environment for Computer Architecture Education. In *IEEE Workshop on Computer Architecture Education (WCAE-04)*, pages 93–98, Munich, Germany, jun 2004.

[Cas18] Andres Castellanos. Jupiter, 2018. `https://github.com/andrescv/Jupiter`.

[GM19] Roberto Giorgi and Gianfranco Mariotti. WebRISC-V: A Web-Based Education-Oriented RISC-V Pipeline Simulation Environment. In *Proceedings of the Workshop on Computer Architecture Education*, WCAE'19, pages 3:1–3:6, New York, NY, USA, 2019. ACM.

[Lan17] Benjamin Landers. RARS, 2017. `https://github.com/thethirdone/rars`.

[MG19] Gianfranco Mariotti and Roberto Giorgi. WebRISC-V, 2019. `https://github.com/Mariotti94/WebRISC-V`.

[Mus10] Besim Mustafa. Evaluating A System Simulator For Computer Architecture Teaching And Learning Support. *Innovation in Teaching and Learning in Information and Computer Sciences*, 9:100–104, 02 2010.

[Pet19] Morten Borup Petersen. Ripes, 2019. `https://github.com/mortbopet/Ripes`.

[PH17] David A. Patterson and John L. Hennessy. *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2017.

[VK18] Keyhan Vakil and Stephan K. Venus, 2018. `https://github.com/ThaumicMekanism/venus`.

[WA19] Andrew Waterman and Krste Asanovic. The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA, 2019. `https://riscv.org/specifications/isa-spec-pdf/`.