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

Roberto Giorgi and Gianfranco Mariotti

Department of Information Engineering and Mathematics - University of Siena

giorgi@dii.unisi.it, 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 experiences. 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 in the web-browser, thanks to its 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.

Index Terms—Computer Simulation, Computer architecture, RISC-V

I. INTRODUCTION

RISC-V is a recent open-source Instruction Set Architecture (ISA) based on reduced instruction set computer (RISC) principles [1]. 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 [2] 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 widely supported also by more than 200 member 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 using 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 the 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 popular book “Computer Organization and Design: RISC-V Edition” by D. A. Patterson and J. L. Hennessy [1], in which the pipelined datapath implementation is explored and explained.

The contribution of this work are:
- a refresh of the WebMIPS simulator [3] 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).

II. RELATED WORK

Several works are based on MIPS architecture, such as [4], in order to show 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 [5], 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. QtSPIM [6] and MARS [7] 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 and do not show the underlying datapath. However there are simulators that focus on these architectural features, visualizing datapath operation graphically, with some common features. Some show a simple single-cycle datapath, for instance MARS plug-in MIPS X-Ray [8]. DrMIPS [9] 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 checking and forwarding units. Notable examples of this subdivision are WinDLX [10], Mipster32 [11], UCOMIPSIM [12], Visimips [13], W ASP [14] and WebMIPS [3].

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 [3], 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 soft processors are just starting to appear [15]. Therefore, we decided to make a RISC-V educational pipelined datapath simulator, taking inspiration from the old WebMIPS [3] and refreshing it with a more appealing ISA, while keeping its convenient web accessibility.

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 IV 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.

III. FEATURES

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

A. General Structure

WebRISC-V has its back-end written in PHP and its front-end in HTML and JavaScript [16], 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). 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 64-bit RISC-V base ISA module and its multiplication extension, but 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:

- RISC-V ISA itself support;
- 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 IV.

B. 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 instruction, the list of supported instructions is always visible on the left side besides the text box (Figure 2).

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

C. 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.

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

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 state of the architectural elements and read their internal state. 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 check-boxes. The Instruction Memory tab displays for each instruction (Figure 4):

- 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.

![Fig. 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.](image-url)

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).

![Fig. 5: The content of the data memory can be inspected.](image-url)

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).

![Fig. 6: The content of the registers.](image-url)

**D. Pipeline data hazards and forwarding during execution**

In the pipeline implementation, as described in Patterson-Hennessey book [1], 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 at cycle number 6. The left panel lists which pipeline stages 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.

![Fig. 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 indicates the stalled instruction.](image-url)

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.

Fig. 8: The content of the Forwarding Unit.

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) and 4 RV64M multiply extension instructions, as described in the RISC-V specification [2].

IV. 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 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 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.

ACKNOWLEDGMENT

We are grateful to the makers of WebMIPS [3], a MIPS pipeline simulator written in ASP, from which WebRISC-V took great inspiration.

REFERENCES