VRV: A Versatile RISC-V Simulator for Education

Noah Krim
nbkrim@ucdavis.edu
Department of Computer Science
University of California, Davis
Davis, California, USA

Joël Porquet-Lupine
jporquet@ucdavis.edu
Department of Computer Science
University of California, Davis
Davis, California, USA

Abstract

This paper introduces VRV, a new and open-source RISC-V simulator designed for educational purposes. VRV aims to fill the gap left by the obsolescence of SPIM, the once-popular MIPS simulator, by providing a user-friendly, versatile tool for teaching computer organization and assembly language programming at undergraduate level. VRV offers both command-line and graphical interfaces, implements a large subset of the RISC-V instruction set, and includes features such as an integrated debugger and support for system-level programming. We discuss the design choices, implementation details, and initial classroom experiences with VRV, highlighting its potential to become a valuable resource in computer architecture education. The source code can be found at https://gitlab.com/luplab/vrv.

ACM Reference Format:
Noah Krim and Joël Porquet-Lupine. 2025. VRV: A Versatile RISC-V Simulator for Education. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE TS 2025), February 26-March 1, 2025, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3641555.3705240

1 Introduction

The field of computer architecture education has long relied on emulators and simulators to provide students with hands-on experience in assembly language programming and computer organization. For many years, in the late 1990s and 2000s, SPIM [5], a MIPS32 simulator, was the gold standard in this domain, featured prominently in Hennessy and Patterson's seminal textbook [6]. However, as the MIPS architecture has now almost completely fallen out of favor in the industry, there is a growing need for educational tools that support more contemporary instruction set architectures (ISAs).

RISC-V, an open-source ISA that has gained significant traction in both academia and industry, presents an ideal replacement for MIPS in educational contexts [3]. While many RISC-V simulators exist, many lack the combination of features that made SPIM so effective as a teaching tool. These features include stability, user-friendliness, both command-line and graphical user interfaces, cross-platform compatibility, and suitability for both interactive use and automated grading.

To address these needs, we have developed VRV, a versatile RISC-V simulator that aims to become the "SPIM of RISC-V" for computer architecture education. VRV combines the best features of SPIM with the modern RISC-V architecture, providing a powerful and flexible platform for teaching assembly language programming and computer organization concepts.

2 Related work

Several educational RISC-V simulators have emerged in recent years, each with its own strengths and limitations.

RARS [1], for instance, offers both command-line and graphical interfaces but is implemented in Java, limiting its potential for future web-based deployment since most web browsers have now abandoned their once native support for Java applets. BRISC-V [2] and WebRISC-V [4] provide rich web-based interfaces, enhancing accessibility for students. However, it makes them unusable in headless autograding scripts.

While these existing tools have made valuable contributions to RISC-V education, we identified an opportunity to create a simulator that more closely mirrors the versatility and user experience of SPIM, while also planning for future online deployment. This led to the development of VRV, which aims to combine the best aspects of existing simulators with additional features and a forward-looking architecture.

3 VRV simulator

VRV is designed to be a comprehensive educational tool for RISC-V assembly language programming and computer organization courses.

At its core, VRV implements the `RV32IMF_Zicsr` instruction set, which includes: the 32-bit base integer instructions (`RV32I`), integer multiplication and division support (`M`-extension), single-precision floating-point support (`F`-extension), and control and status registers support (`Zicsr`-extension). This subset of the RISC-V instruction set provides a solid foundation for teaching fundamental concepts while also allowing exploration of more advanced topics. To facilitate a deeper understanding of computer architecture, VRV supports two privilege levels: machine mode and user mode. This feature enables students to write both user-level programs and system-level code, providing a more comprehensive understanding of computer architecture and operating system concepts.

VRV assembles RISC-V assembly code at runtime. This feature simplifies the workflow for students and instructors alike, as VRV automatically resolves symbol references and expands pseudo-instructions, allowing students to focus on learning assembly language concepts rather than wrestling with potential toolchain complexities.

VRV provides some default system code, mostly to handle unexpected exceptions, therefore allowing students to write user programs without needing to implement low-level exception handling.

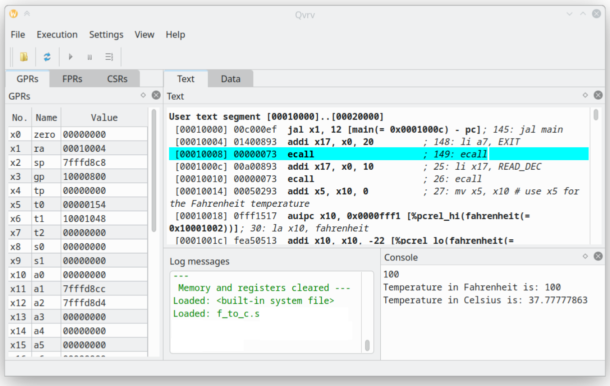
SIGCSE TS 2025, February 26-March 1, 2025, Pittsburgh, PA, USA
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0532-8/25/02
https://doi.org/10.1145/3641555.3705240

However, it also offers the flexibility for students to supply their own system code, enabling their exploration of advanced topics such as interrupt or exception handling and basic operating system development. To facilitate input/output operations, VRV includes a simple, optional memory-mapped TTY device (the LupIO-TTY from the LupIO collection of educational device [7]), allowing students to interact with their programs and learn about memory-mapped I/O concepts in a controlled environment.

One of VRV's most powerful features is its integrated debugger. Students can set breakpoints, watch memory locations, and display register contents, all of which are crucial for understanding program execution and debugging assembly code. This feature significantly enhances the learning experience by providing immediate feedback and visualization of program behavior.

Finally, VRV offers two equivalent interfaces: a command-line interface (CLI), and a graphical user interface (GUI) implemented using the Qt framework. The CLI is ideal for use in automated autograding scripts and for students who prefer terminal-based workflows, while the GUI provides a more intuitive environment for visual learners and interactive debugging sessions. This dual-interface approach, shown in Figure 1 ensures that VRV can meet the diverse needs of different educational settings and student preferences.

```
$ ./vrv f_to_c.s
Loaded: <built-in system file>
Loaded: f_to_c.s
100
Temperature in Fahrenheit is: 100
Temperature in Celsius is: 37.77777863
```



No.	Name	Value
x0	zero	00000000
x1	ra	00010004
x2	sp	7fffd8c8
x3	gp	10000800
x4	tp	00000000
x5	t0	00000154
x6	t1	10001048
x7	t2	00000000
x8	s0	00000000
x9	s1	00000000
x10	a0	00000000
x11	a1	7fffd8cc
x12	a2	7fffd8d4
x13	a3	00000000
x14	a4	00000000
x15	a5	00000000

```
User text segment [00010000]..[00020000]
 [00010000] 00c000ef  jal x1, 12 [main(= 0x0001000c) - pc]; 145: jal main
 [00010004] 01400893  addi x17, x0, 20          ; 148: li a7, EXIT
 [00010008] 00000073  ecall                     ; 149: ecall
 [0001000c] 00a00893  addi x17, x0, 10          ; 25: li x17, READ_DEC
 [00010010] 00000073  ecall                     ; 26: ecall
 [00010014] 00050293  addi x5, x10, 0           ; 27: mv x5, x10 # use x5 for
the Fahrenheit temperature
 [00010018] 0fff1517  auipc x10, 0x0000fff1 [%pcrel_hi(fahrenheit(=
0x10001002))]; 30: la x10, fahrenheit
 [0001001c] fea50513  addi x10, x10, -22 [%pcrel lo(fahrenheit(=
```

```
---
 Memory and registers cleared ---
Loaded: <built-in system file>
Loaded: f_to_c.s
```

```
100
Temperature in Fahrenheit is: 100
Temperature in Celsius is: 37.77777863
```

Figure 1: VRV CLI and GUI

4 Classroom experience

We introduced VRV at our institution in a lower-division computer organization class during the Fall 2023 term. The course included three programming assignments in assembly code, designed to leverage VRV's capabilities. Students implemented basic programs such as coin change calculation and temperature conversion, then a sorting and binary-searching program using functions, and they finally wrote a trap handler to emulate misaligned memory accesses.

In our end-of-the-term class survey, we included a few 4-point Likert-scale questions about VRV as well as some optional free-response sections. As shown in Figure 2, initial feedback from students was encouraging, with over 75% reporting a positive experience. They often reported appreciating the clean interface and powerful debugging capabilities of VRV.

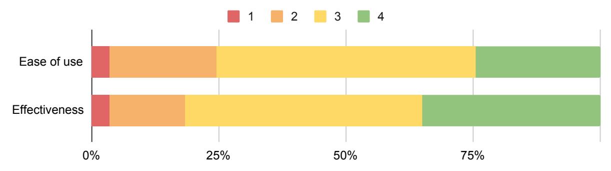


Figure 2: Students' feedback about VRV (114 students total)

However, as with any new software development, we encountered some challenges too. At the time of deployment, the multiplatform support was not ready, preventing us from letting students install the simulator on their own devices. Instead we provided it through our instructional computer facility, so that students could use it over SSH. Some unfortunately reported experiencing difficulty running the Qt-based GUI version over SSH. Additionally, others were sometimes confused with the error messages when assembling code, highlighting the need for improved clarity and specificity in error reporting. The need for comprehensive documentation became apparent during the course too, as we received a few requests for features that actually already existed.

5 Conclusion

VRV aims to fill the gap left by the obsolescence of SPIM, but for the RISC-V computer architecture. Our initial classroom deployment has demonstrated its potential to become a valuable resource for teaching computer organization and assembly language programming. As we continue to refine and expand VRV, we are focusing on improving cross-platform compatibility, enhancing the user experience, and developing a web-based implementation. These efforts will ensure that VRV remains a versatile and powerful tool for computer architecture education in the years to come.

References

[1] 2024. RARS: RISC-V Assembler and Runtime Simulator. https://github.com/TheThirdOne/rars [Accessed 10/14/2024].
[2] Rashmi Agrawal, Sahan Bandara, Alan Ehret, Mihailo Isakov, Miguel Mark, and Michel A Kinsy. 2019. The BRISC-V platform: A practical teaching approach for computer architecture. In *Proceedings of the Workshop on Computer Architecture Education*. 1–8.
[3] Krste Asanović and David A Patterson. 2014. Instruction sets should be free: The case for risc-v. (2014).
[4] Roberto Giorgi and Gianfranco Mariotti. 2019. Webrisc-v: A web-based education-oriented risc-v pipeline simulation environment. In *Proceedings of the workshop on computer architecture education*. 1–6.
[5] James Larus. 2011. SPIM: A MIPS32 Simulator. https://pages.cs.wisc.edu/~larus/spim.html [Accessed 10/14/2024].
[6] David A Patterson and John L Hennessy. 2005. *Computer organization and Design* (third ed.). Morgan Kaufmann,.
[7] Joël Porquet-Lupine. 2021. LupIO: a collection of education-friendly I/O devices. In *2021 ACM/IEEE Workshop on Computer Architecture Education (WCAE)*. IEEE, 1–8.