stated previously we plan to continue using it in future offerings of our courses and have identified features for further expansion. The features that we have identified for future expansion and believe are worthy of mentioning here fall into three categories: increased auto-grading support, increased platform support, and additional emulated hardware.

The first and most likely highest priority feature that we would like to provide is to add a GUI dialog to specify any output commands. Currently, the output commands specified in the input JSON files must be added manually after recording. Being able to quickly select the registers and/or memory sections that the instructor would like to compare would greatly aid in the workflow. Along the lines of auto-grading support we believe that adding a Dockerfile and associated scripts to aid in Gradescope programming assignment integration would be greatly beneficial to the community. The Gradescope integration is likely slated to be developed during the fall term as the current plan is to utilize the RVCE in more courses. The last feature associated with auto-grading is to add support for outputting screen rendering in graphics mode. The actual image rendered to the screen can be accomplished in many ways, for example background 0 or 1 could be used with the other disabled and the resulting image would be the same with the contents of the video memory being vastly different. If an instructor is attempting to assess the contents actually displayed on the screen the current output capabilities would make that difficult. Additionally, we have considered if the screen rendering output should be as hex in the output JSON file, or if the image should be rendered to a PNG. We plan to further investigate this over the coming year.

Currently, Windows, OS-X, and Linux are supported through the use of a Docker container; however, Docker requires superuser access on Linux in order to run, likely preventing students from running it on shared instructional systems. We plan to add support through a Singularity container as well; this would provide the ability for students to launch the container on shared systems in which they have limited access rights. In addition, it could be highly beneficial to add support for a native GUI on Windows and OS-X so that students may directly install the RVCE.

The other features we would like to add are related to additional emulated hardware. The first piece of hardware we believe could be beneficial is a UART type device. By providing a UART interface that was connected to a standard in and out would provide users another debug interface that could be utilized during development. In addition, by having a UART type interface that is connected to the host system, users of the RVCE could in theory network instantiations and actually have multiplayer games. The final piece of emulated hardware that we see would improve the RVCE is to add audio support. Due to difficulties in connecting audio to Docker containers on some platforms, this remains a larger development effort that it may initially appear; however, we hope to eventually support this and the other features.

7 ACKNOWLEDGMENTS

We would like to acknowledge the 45 graduate students in ECS 251, 222 undergraduate students in ECS 50, and 136 undergraduate students in ECS 150 who were the first users of RVCE. Specifically we would like to thank Xiaoyi “Eric” Li who discovered and fixed so many of RVCE’s early bugs.

REFERENCES

[1] Fabrice Bellard. 2005. QEMU, a Fast and Portable Dynamic Translator. In *2005 USENIX Annual Technical Conference (USENIX ATC 05)*. USENIX Association, Anaheim, CA. https://www.usenix.org/conference/2005-usenix-annual-technical-conference/qemu-fast-and-portable-dynamic-translator

[2] TIS Committee. 1995 [Online].. *Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification.* https://refspecs.linuxfoundation.org/elf/elf.pdf

[3] 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 (WCAE’19)*. Association for Computing Machinery, New York, NY, USA, Article 3, 6 pages. https://doi.org/10.1145/3338698.3338894

[4] Mikey Goldweber, Renzo Davoli, and Mattia Biondi. 2021. The Pandos Project and the μMPS3 Emulator. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE ’21)*. Association for Computing Machinery, New York, NY, USA, 122–128. https://doi.org/10.1145/3430665.3456331

[5] Michael Goldweber, Renzo Davoli, and Tomislav Jonjic. 2012. Supporting Operating Systems Projects Using the μMPS2 Hardware Simulator. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE ’12)*. Association for Computing Machinery, New York, NY, USA, 63–68. https://doi.org/10.1145/2325296.2325315

[6] Michael Goldweber, Renzo Davoli, and Mauro Morsiani. 2005. The Kaya OS Project and the μMPS Hardware Emulator. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE ’05)*. Association for Computing Machinery, New York, NY, USA, 49–53. https://doi.org/10.1145/1067445.1067462

[7] Jason Lowe-Power, Abdul Mutaal Ahmad, et al. 2020. The gem5 Simulator: Version 20.0+. arXiv:cs.AR/2007.03152

[8] Jason Lowe-Power and Christopher Nitta. 2019. The Davis In-Order (DINO) CPU: A Teaching-Focused RISC-V CPU Design. In *Proceedings of the Workshop on Computer Architecture Education (WCAE’19)*. Association for Computing Machinery, New York, NY, USA, Article 2, 8 pages. https://doi.org/10.1145/3338698.3338892

[9] Marco Melletti, Michael Goldweber, and Renzo Davoli. 2015. The JaeOS Project and the μARM Emulator. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE ’15)*. Association for Computing Machinery, New York, NY, USA, 3–8. https://doi.org/10.1145/2729094.2742596

[10] SiFive. 2019 [Online].. *Simulating with QEMU.* https://sifive.github.io/freedom-e-sdk-docs/userguide/qemusimulation.html

[11] GTK Development Team. 2018 [Online].. *Gtk – 3.0: The GTK toolkit.* https://docs.gtk.org/gtk3/

[12] Andrew Waterman and Krste Asanović. 2019 [Online].. *The RISC-V Instruction Set Manual Volume I: Unprivileged ISA.* https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMAFDQC/riscv-spec-20191213.pdf

[13] Andrew Waterman and Krste Asanović. 2019 [Online].. *The RISC-V Instruction Set Manual Volume II: Privileged Architecture.* https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMFDQC-and-Priv-v1.11/riscv-privileged-20190608.pdf