Improving Student Comprehension with Logisim-based RISC Processors

Major Bobby Birrer, PhD
Department of Computer and Cyber Sciences
United States Air Force Academy
Colorado Spring, CO USA
bobby.birrer@afacademy.af.edu

Dr. Carlos Salazar
Electrical Engineering and Cyber Systems
United States Coast Guard Academy
New London, CT USA
carlos.l.salazar@uscga. edu

***Abstract*—This Work in Progress paper in Innovative Practice details continued efforts to improve student comprehension of computer architecture across MIPS and ARM. Students majoring in Computer Science at the United States Air Force Academy (USAFA) are typically enrolled in the department's Computer Architecture course during the spring of their sophomore year. These students traditionally struggled to understand exactly what was occurring in the Central Processing Unit (CPU) of a computer and demonstrated poor performance on assignments, tests, and the final exam on these concepts. Other institutions have had success giving their students a series of assignments which culminate in their implementing a complete CPU architecture in a logic simulator. However, this fairly time consuming assignment was not deemed feasible at USAFA where student time is divided between various required duties, academics, and athletics. The authors' previous work showed that providing students with a reduced version of the MIPS (Microprocessor without Interlocked Pipelined Stages) single cycle architecture (SCA) implemented in Logisim increased student comprehension as it allowed them to observe the internal processor operations at a very detailed level. In 2021, the course switched from MIPS to ARM, and students were not provided with this tool and their performance on assessments regressed. In 2022, students were provided with a simulated ARM processor built in Logisim to recreate the improved performance from the MIPS tool. The paper shows five years of data across multiple instructors, virtual/in-person learning, different architectures, and various texts showing that observing simulated processors improves overall student performance with very little additional work from the instructors or the students.**

***Index Terms*—ARM, LEGv8, RISC, Computer Organization, Computer Architecture, Simulation, Logisim**

I. Introduction

USAFA Computer Science Majors complete their computer organization and architecture course, CS351, during the spring of their sophomore year. The students generally performed poorly on assessments of their understanding of the inner workings of the CPU. This was most noticeable on a particular problem that required them to decode an instruction, examine its influence on various computational and decision structures, and report the resulting outputs at various points in the CPU data path. The authors successfully addressed this comprehension gap previously by building a simulated MIPS processor in Logisim, "an education tool for designing and simulating digital circuits" [1]. The students used this simulated processor to complete a number of exercises, which led to increased scores on their homework, an exam, and the comprehensive final exam. However, the course switched to using ARM in 2021, without a simulated processor and students once again experienced issues with exercises related to CPU operation. The authors built a simplified version of the ARM/LEGv8 processor in 2022 and saw student performance rebound.

This paper first describes the USAFA computer organization and architecture course along with the assessments used to evaluate student understanding of the course material. It also explains the difficulties the students experienced in understanding the functionality and implementation of an ARM processor after the course transitioned from MIPS. The paper then describes previous efforts to improve student comprehension of CPU functionality that helped inform this effort. The paper outlines the creation of the Logisim implementation of the LEGv8 processor and how it was used by the students to simulate processor operation. Finally, quantitative data on the students' improved comprehension of the subject matter from the 2017-2022 timeframe is presented.

II. Background

A. Description of the Course and Assessments

When taking CS351, the students have completed their Introduction to Computer Science and Programming Fundamentals courses and are taking a Data Structures and Systems Programming course. CS351 introduces basic computer logic systems, major types of computing system organizations, and machine and assembly language programming and includes digital logic, processor architecture, data representation, memory architecture, performance analysis, computer arithmetic, pipelining and multi-processing. It also covers memory hierarchy, caching, virtual memory, and emerging topics. The course moves quickly, beginning with combinational and sequential digital logic, then to designing arithmetic logic units, and then transitioning to CPU functionality using the MIPS architecture as the primary example.

The students have historically performed well in the digital logic portion of the course, designing, building, and testing circuits to implement solutions to academic problems. Following the digital logic block, the course shifts into RISC architecture. The course used MIPS as the teaching architecture based on the Harris and Harris text [2], but switched in 2021 to

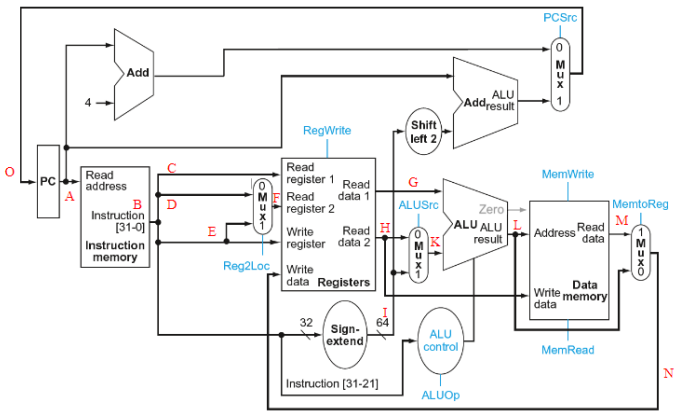

Fig. 1. Example of SCA Assessment Exercise

the LEGv8 subset of ARM as described in Patterson and Hennessy's text [3]. The lectures show the design of the data path and control paths using animated PowerPoint slides and board diagrams. The students follow the examples and practice manually translating between machine code and assembly mnemonics (ADD, ORR, LDUR, STUR, etc.).

Before the exam that covers the CPU architecture, the students complete a homework assignment that asks them to examine a processor state, including register and memory contents. They then fill in the values for various components of the data path based on the register and memory state. They are then asked to complete a similar task on the following exam and again on the final exam. Figure 1 shows an example of the assessment, where the students would be required to determine the appropriate values for A-O for an instruction. Historically, students have struggled to fill in these values in the homework, exam, and final exam, which indicates that they do not fully understand the operation of the processor.

B. Related Work

Other institutions have experienced similar issues with students struggling to fully understand how processors function and have taken a variety of approaches to improve student comprehension. The baseline method of teaching the material generally includes lectures on MIPS, ARM, or other reduced instruction set computing (RISC) architectures. Patterson and Hennessy's seminal "Computer Organization", which is used in CS351, uses this approach when describing processors [3].

Multiple programs have built tools to allow students to better visualize the operation of processors of various types. Reference [4] built a simulated MIPS processor in their simulation tool, DLSim3. The system, which can use Java-based plugins, can simulate a single cycle or pipelined implementation of the MIPS processor. The tool provides an interface to import machine code from MARS and execute it. This allows students to compare the execution across both implementations and see the registers and memory update during program execution.

Reference [5] developed a graphical extension to MARS called MIPS X-Ray that animates which data and control signals and functional units are active during the execution of a MIPS instruction. It displays the instruction mnemonic and the breakdown of the machine code bits into the instruction-specific fields. Reference [6] implemented a similar tool that simulates the execution of a MIPS pipelined processor and displays instructions flowing through the processor over multiple clock cycles. It also allows for probing of the data path during execution.

These tools and others do a great job of showing the changes in registers and memory. However, USAFA instructors wanted a tool that would help the students understand the MIPS data and control paths and allow them to observe the bits of data moving through the system. Also, the students needed to be able to visually examine multiple layers of abstraction to see how gates are built into more complex circuits and eventually into the full processor. By seeing the data at the digital logic/gate-level, the students theoretically would better understand the linkage of digital logic and machine code to assembly language and higher-level languages. Additionally, instructors wanted the option to have students build or modify the processor to increase their comprehension of its construction.

Reference [7] took a similar approach to the USAFA vision, having students build a simple CPU in Logisim. However, the implemented CPU was based on the Mic-1 architecture rather than a MIPS processor. Over the course of multiple assignments, the students built an ALU, datapath, and control unit in Logisim. They also loaded and ran a basic assembly program on their processor to test and make corrections to their architecture as needed. Reference [8] also had students build an ARMv8 processor in Logisim and run a number of programs on it.

Reference [9] implemented a similar series of assignments in their CPU design curriculum, having students build a MIPS-based processor using Verilog. The students built the data path and incrementally built the CPU before testing and verifying the operation by running assembly code. The authors tracked the time students spent on the design process along with the changes in success rate of CPU design. The students spent approximately 25 hours executing these assignments and raised their success rate from less than 30 percent to over 76 percent. The techniques of [7] and [8] and [9] are advantageous from a pedagogical approach as they walk the students through the end-to-end process of processor design in a hands-on manner. However, the time investment for those assignments is more than could be supported by the current CS351 course without removing significant amounts of required content.

The authors previously showed in [10] that much smaller, focused assignments using a MIPS processor in Logisim still have significant positive impacts on student performance. Student performance increased between 6 and 14 percent on summative assessments after a homework that required less than three hours to complete.

C. Description of Logisim

Logisim, is "an educational tool for designing and simulating digital logic circuits", including combinational and sequential logic [1]. It offers probe functionality that displays the contents of multi-bit busses, and these probes can be configured to show the values in binary, hexadecimal, and unsigned or signed decimal. It also allows for the construction and saving of sub-circuits, allowing the students to use abstracted building blocks to create their circuits. The students use Logisim early in the CS351 course to implement and test basic logic circuits including adders, comparators, and multiplexers. The students then build a finite state machine and a simple arithmetic logic unit. The assignments help familiarize the students with the Logisim interface which they use to simulate the execution of the ARM processor.

III. Logisim ARM Processor and Assignments

To help students learn a representative SCA, the authors provided students with a simulated processor in Logisim. The students then used this processor in assignments designed to have them interact with the processor and observe and modify its operation. The first iteration of the processor was based on MIPS and described in [10]. This section details the creation of an ARM-version and how the assignments were modified to adapt to the continued evolution of the course.

A. Description of Logisim ARM Processor Design

The authors followed the design from Patterson and Hennessy to create a new variant of the LEGv8 single cycle architecture. Logisim is limited to 32-bit components, so the initial design used two parallel data paths to create the full 64-bit datapath for LEGv8. However, the authors determined this would greatly clutter the appearance of the circuit, making it significantly less useful as a teaching tool. The authors modified the existing LEGv8 architecture to 32-bit registers, data lines, ALU, etc. which they refer to as LEGv8R for LEGv8 Reduced. Given that the LEGv8 architecture is already a reduced approximation of the ARMv8 architecture this seemed a reasonable compromise for pedagogical purposes. Further, the data registers and ALU of the LEGv8 architecture are 64 bits, but both the ARMv8 and LEGv8 architectures use 32-bit instructions meaning that reduction of the data wordsize to 32 bits had a negligible impact on programs and execution.

The entire reduced LEGv8R architecture was implemented from the base components supplied by Logisim and construction proceeded hierarchically with smaller subcomponents incorporated into more complex components until all necessary logic circuits had been created. The completed build supported LDUR, STUR, CBZ and a variety of arithmetic instructions. The Patterson and Hennessy textbook provides detailed instructions on how to add an additional branch instruction which can be demonstrated in class or assigned as homework to the students [3]. Figure 2 shows the top-level datapath for the circuit.

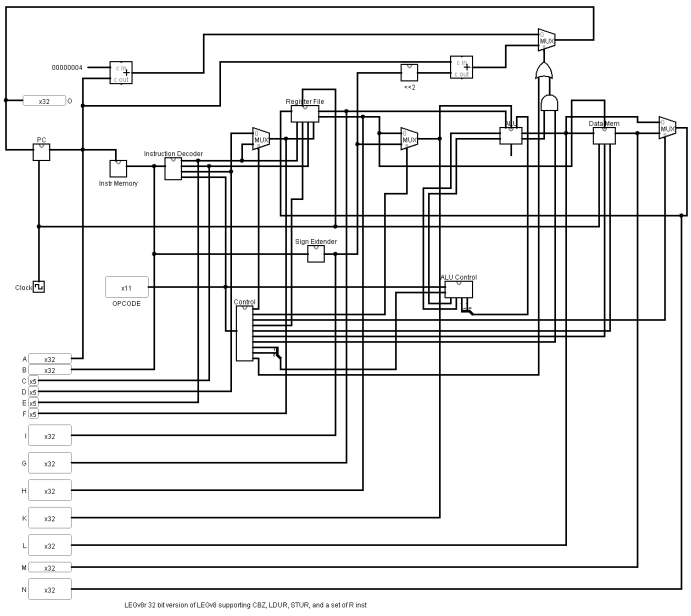

Fig. 2. Top-level Circuit

B. Description of Updated Assignment

When the students used the MIPS simulated processor in Spring 2018 and 2019, they completed four steps as part of dedicated assignment. They instrumented the processor, added a new instruction, loaded instruction memory, data memory, and registers, and then ran a simple program. This series of steps allowed the students to interact with the processor and even modify it without having to build it from scratch.

Due to COVID-19 in Spring 2020, the entire student body was sent home and the second half of the course transitioned to remote learning. As part of the rapid reorganization of the course, the assignment was shifted to an optional exercise. Additionally, the course switched to ARM in Spring of 2021, and the authors did not immediately create a new simulated processor. Therefore, the assignment was not provided in any form that semester.

The authors completed the LEGv8R processor in March of 2022, and distributed it to the students as part of a homework assignment rather than a dedicated assignment. The students were tasked to manually fill in the values of the SCA datapath and control path on the worksheet. They then instrumented the simulated processor and loaded the registers, data memory, and instruction memory. The students then compared the results to their manual calculations. The authors chose this method of instruction to keep additional student workload at a minimum while still utilizing a visual tool to reinforce the concepts.

IV. Results

Table I shows the student averages on the SCA-related questions on the homework, exam, and final exams since 2017. As previously shown in [10], student scores significantly increased when the simulated MIPS processor and associated assignments were introduced.

Assessment	S17: 45 Students	S18: 30 Students	S19: 29 Students	S20: 28 Students	S21: 23 Students	S22: 28 Students
HW	46.89	79.67	70.52	Unavailable	76.60	80.17
GR	64.71	70.40	79.17	Unavailable	42.37	57.02
Final	76.16	84.82	87.50	64.46	66.79	76.85
Architecture	MIPS	MIPS	MIPS	MIPS	ARM	ARM
Format	In-Person	In-Person	In-Person	COVID	Distance Learning	In-Person
Simulated Processor	No	Yes	Yes	Optional	No	Partial

TABLE I
STUDENT SCORES ON ASSESSMENTS

Due to the sudden, COVID-driven shift to distance learning in the Spring 2020 offering of the course, the students were not required to do the assignment using the simulated processor. Instead, they were given the assignment processor as an optional exercise. Anecdotal feedback from the students stated that few completed the exercise or used the processor. The students that did use the processor confirmed that it was helpful, however. Overall, the average scores dropped by an average of 23 percent on the associated final exam questions.

The course director intentionally designed the Spring 2021 offering to support distance learning. However, the course switched from MIPS to ARM and did not have a simulated processor and instead relied on demonstrations, videos, and PowerPoint presentations to explain the SCA. Student performance on SCA assessments regressed to levels not seen since the Spring 2017 offering when a processor was not available. Students averaged 66.79 percent on the SCA final exam questions and anecdotal feedback confirmed that they struggled with those concepts.

The Spring 2022 offering made only minor changes to the course as it returned to in-person learning. The students received the simulated processor ahead of the SCA homework and were required to instrument the data path and record the values on various lines. However, they were not required to modify the data or control paths to add a new instruction as they were in 2018 and 2019. The students performed slightly better on the SCA homework than the previous year, increasing 3 percent. However, when queried on the concepts during class time, they were better able to articulate the SCA functionality than the two previous offerings which did not have a simulated processor assignment. Multiple students stated that the processor was very helpful in increasing their understanding. Student performance on the SCA questions on the first exam increased by almost 15 percent. Additionally, student performance on the SCA questions of the final exam increased by over 10 percent. These scores were comparable to scores from the Spring 2017 semester when no processor was available. This may indicate that significant portions of the improvement were from switching from a distance learning environment to in-person learning. It could also indicate that the instrumentation portion of the exercises has little impact on student understanding. Instead, the main benefit of the simulated processor activities likely comes from adding the additional instructions and seeing a program run.

V. FUTURE WORK

The authors intend to evaluate [8] as an alternative simulated processor that can be used with the same lightweight assignment. That processor uses the full 64-bit data path and a larger instruction compared to the LEGv8r, making it more robust and suitable for executing actual programs. However, its additional complexity may be counterproductive for the learning objectives in the authors' CS351 course.

Additionally, the authors intend to reintroduce the assignment requiring the students to modify the processor to add instructions following the guidance in Patterson and Hennessy [3] and then run a program. This technique was used in 2018 and 2019, but not in 2020 or 2022. The authors believe this activity provides similar levels of active learning as building the entire processor with a fraction of the workload for the students and instructors. This additional data would also clarify how much of 2022 improvements were from the return to in-person learning rather than the instrumentation exercise.

VI. CONCLUSION

The findings confirmed early research showing that student performance and understanding can be greatly enhanced with a relatively simple, lightweight assignment using a simulated RISC processor. The authors tested this methodology over multiple years, architectures, and instruction methods, and measured quantitatively using multiple assessment mechanisms. The interactive nature of the simulated processor provides a distinct advantage over traditional PowerPoint demonstrations and increases overall student engagement and comprehension. Further, the methodology proved effective without requiring students to actually build the processor, as simply instrumenting and making minor modifications to it significantly increased student performance on the associated assessments.

ACKNOWLEDGMENT

Approved for public release: distribution unlimited. The views expressed in this article, book, or presentation are those of the authors and do not necessarily reflect the official policy or position of the United States Air Force Academy, the Air Force, the Department of Defense, or the U.S. Government.

Major Birrer would like to thank his wife and daughter for their support throughout his career.

REFERENCES

[1] C. Burch, "Logisim: A graphical system for logic circuit design and simulation," *Journal on Educational Resources in Computing (JERIC)*, vol. 2, no. 1, pp. 5–16, 2002.

[2] D. Harris and S. L. Harris, *Digital design and computer architecture*. Morgan Kaufmann, 2010.

[3] D. A. Patterson and J. L. Hennessy, *Computer organization and design ARM edition: the hardware software interface*. Morgan kaufmann, 2016.

[4] J. L. Donaldson, R. M. Salter, J. Kramer-Miller, S. Egorov, and A. Singhal, "Illustrating cpu design concepts with dlsim 3," in *2009 39th IEEE Frontiers in Education Conference*. IEEE, 2009, pp. 1–6.

[5] M. R. D. Araujo, F. L. C. Padua, F. V. Andrade, and F. L. Correa-Junior, "Mips x-ray: A mars simulator plug-in for teaching computer architecture," *International Journal of Recent Contributions from Engineering, Science & IT (iJES)*, vol. 2, no. 2, pp. 36–42, 2014.

[6] M. T. Kabir, M. T. Bari, and A. L. Haque, "Visimips: Visual simulator of mips32 pipelined processor," in *2011 6th International Conference on Computer Science & Education (ICCSE)*. IEEE, 2011, pp. 788–793.

[7] D. C. Schuurman, "Step-by-step design and simulation of a simple cpu architecture," in *Proceeding of the 44th ACM technical symposium on Computer science education*, 2013, pp. 335–340.

[8] M. Kayaalp, "Using logisim-evolution for teaching datapath and control," in *2021 ACM/IEEE Workshop on Computer Architecture Education (WCAE)*. IEEE, 2021, pp. 1–8.

[9] L. Wang, Z. Yu, D. Zhang, and G. Qin, "Research on multi-cycle cpu design method of computer organization principle experiment," in *2018 13th International Conference on Computer Science & Education (ICCSE)*. IEEE, 2018, pp. 1–6.

[10] C. Salazar and M. B. Birrer, "Instrumentation and extension of reduced, simulated single cycle mips architecture to improve student comprehension," in *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2020, pp. 1–5.