Instrumentation and Extension of reduced, simulated Single Cycle MIPS architecture to improve Student Comprehension

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

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

***Abstract*—This Innovative Practice Work in Progress Paper presents results of efforts to improve student comprehension of computer architecture. 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 regarding how these components are used in a processor. 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. Instead, a reduced version of the MIPS (Microprocessor without Interlocked Pipelined Stages) single cycle architecture (SCA) as described in Harris and Harris was implemented in Logisim and provided to the students. The students then modified the given circuit to support additional instructions and added logic probes to observe internal values. They then ran assembly language code containing the new instructions through the augmented architecture. Throughout each step, the students could observe the processor's behavior using the previously installed probes, which substantially increased their understanding. The improvements in performance on the CPU questions have been significant and lasting. These results are especially noteworthy due to the minor additional work the students had to perform to increase their understanding.**

Keywords—MIPS, 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 bottom-up nature of the computer architecture course resulted in good performance on cognitive tasks related to many of the building blocks of the CPU (e. g. state machines, ALUs). However, the students generally performed poorly on assessments of their understanding of the inner workings of the CPU. Based on observations and experiences at other institutions facing similar issues, the course director created a model of a MIPS processor using Logisim, an educational tool for designing and simulating digital circuits [1]. In 2018, the course director then added a new assignment that required the students to instrument the provided architecture (by adding logic probes to various points in the circuit) and modify the design to support additional instructions before completing similar assessments that had proven difficult to prior years' students. Improvement on the CPU questions was observed on the homework assignment, the exam given during the semester, and on the comprehensive final exam. Two different instructors teaching in successive years observed the same improvements in performance, increasing from 76.16% on the CPU portion of the final exam to 87.5%.

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 a MIPS processor. The paper then describes related efforts to improve student comprehension of CPU functionality that helped inform this effort. The paper outlines the creation of the Logisim implementation of the MIPS 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 2017-2019 timeframe is presented.

II. Background

A. Description of the Course and Assessments

When taking CS351, the students have completed courses in Intro to Computer Science and Programing Fundamentals and are taking a Data Structures and Systems Programing 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 course uses a variety of assessments to evaluate the students' understanding. During the digital logic block, the students complete a series of assignments requiring them to design logic circuits and implement and test them in Logisim [1]. These assignments are designed to help the students prepare for the midterm exam which tests their comprehension in a controlled environment. The students have historically done well on this portion of the course, performing well on the homework assignments and the exam.

Following the digital logic block, the course shifts into the MIPS architecture. The students are introduced to the MIPS instruction set architecture and how to program in assembly language via the MIPS Assembler and Runtime Simulator (MARS), an integrated development environment and emulator [2]. The lectures follow the Harris and Harris "Digital Design and Computer Architecture" textbook and walk through building the data path and control paths using animated PowerPoint slides and board diagrams [3]. The students follow the examples and practice manually translating between machine code and MIPS assembly mnemonics (ADD, ADDI, XOR, etc.). Prior to 2018, only the MARS simulator was used to show the processor interpreting and executing instructions.

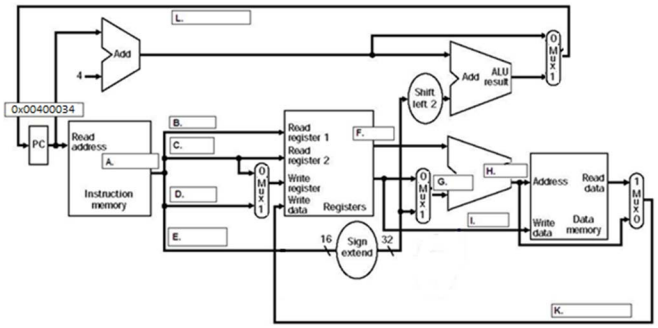

Fig. 1. MIPS Question on Homework and Exams

The students complete a number of assembly language programming exercises using MARS while learning about the architecture. Before the exam that covers the MIPS architecture, the students complete a homework assignment that asks them to examine a MIPS 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 what the students would be required to populate for these assignments. In 2017, the students averaged 46.89% on this portion of the homework, 64.71% on the exam, and 76.16% on the final exam.

B. Related Work

Other institutions besides USAFA 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" and Harris and Harris's "Digital Design and Computer Architecture", which is used in CS351 both use this approach when describing processors [3-4].

Multiple programs have built tools to allow students to better visualize the operation of processors of various types. Donaldson et al. [5] 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.

Araújo et al. [6] 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. Kabir, Bari and Haque [7] 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.

Schuurman [8] 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.

Wang et al. [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. Wang et al. [9] 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% to over 76%. The techniques of Schuurman [8] and Wang et al. [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 required content.

C. Description of Logisim

Logisim, created by Carl Burch, 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 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 will continue to use to simulate the execution of the MIPS processor.

III. Logisim MIPS Processor and Updated Assignments

For students just learning about computer architecture, constructing a fully operational MIPS processor in a logic simulator is a daunting and time consuming task. Even the reduced SCA of [3] (see Figure 2) is a challenge for them. Rather than have the students get into the minutia of such a large task, the course director implemented a reduced single cycle MIPS processor in Logisim and assigned the students the task of using this simulation (see Figure 3).

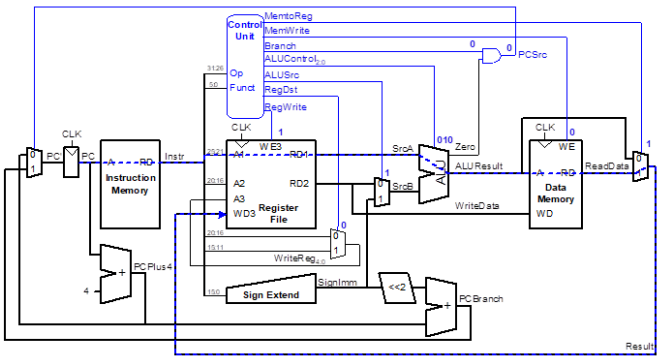

Fig. 2. MIPS SCA from [3] that was implemented in Logisim.

A. Description of Logisim MIPS Processor Design

The top level design (TLD) of the Logisim simulation of the circuit in Figure 2 is shown in Figure 3. The component that the students were asked to modify was the Control Unit at the bottom of the circuit in Figure 3. The Control Unit consists of the Main Decoder and the ALU Decoder which provide the control signals to the ALU, multiplexers, and write enable inputs of the memory component and register file. The Main Decoder was where the circuit needed to be modified to implement the new truth table that provided support for the `addi` instruction (see Figure 4).

All of the Logisim circuit files were provided to the students along with the instruction memory test contents that included the `addi` instruction.

B. Description of Updated Assignment

The additional assignment comprised four tasks. The first task required students to instrument the architecture in the same places as the original homework and exam questions (see Figures 1 and 3).

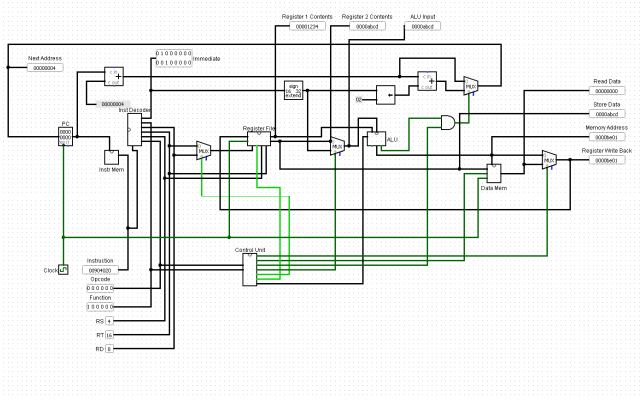

Fig. 3. Instrumented Logisim simulation of the MIPS SCA Processor

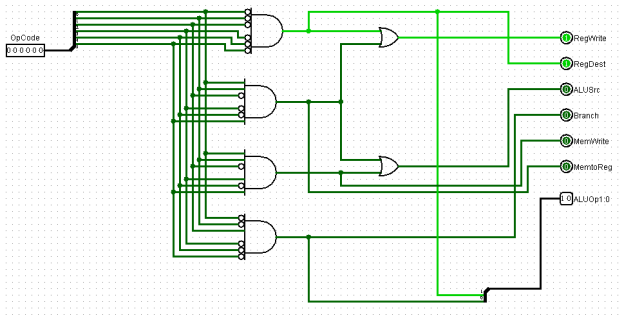

Fig. 4. Main Decoder circuit for the MIPS SCA.

The second task was to load the memory and registers with the contents as described in the homework and use the simulator with their probes to answer the original homework questions (namely the contents of the boxes A through L in Figure 1). This allowed them to see the values across the data path and something which had been so impenetrable for the students became quite simple. They still had to decode the machine language instruction by hand to determine which assembly language instruction it was and then determine which outputs were not used by the instruction.

The third task was to update the architecture to provide support for additional instructions. The specific instruction chosen was the `addi` add immediate instruction. The update required modifications to the control logic truth tables used in the main decoder and required fairly modest changes to the circuitry in Figure 4. Implementing these changes helped solidify their understanding of the decode operation which many of them had previously struggled to comprehend.

The fourth and final task was to test their newly modified circuit. This involved loading the provided image into instruction memory, running through 8 clock cycles, and observing and recording the values at various locations. Three ground truth values along with the contents of the program counter were included to help with debugging.

IV. Results

In spring 2017, the semester before the new assignment was introduced, students averaged 46.89% on the associated

homework problem. Following the introduction of the Logisim MIPS processor and the new assignment, the students' homework scores improved to 80% in spring 2018 and 70.52% in spring 2019. The students' scores on the exam problem improved from 64.71% in the spring of 2017 to 70.4% and 79.17% in the spring of 2018 and 2019, respectively. On the final exam, the scores improved from 76.16% in spring 2017 to 84.42% and 87.5% in the spring of 2018 and 2019, respectively.

The course director and sole instructor for the course changed between spring 2018 and spring 2019, but the improvement in comprehension was consistent across both years and instructors. The students showed a slight regression in performance on the homework assignment between spring 2018 and spring 2019, but this could have been due to the 2019 instructor being new or differences in grading the homework. The resulting average was still over 23% higher than before the simulated processor and new assignment were introduced. The large (33%) improvement on the homework assignment is particularly noteworthy, as it is an improvement on a formative, rather than summative, assessment. This indicates the students achieved a greater understanding of the material earlier than they would have otherwise, potentially increasing their understanding of later subjects. However, further analysis is required to evaluate that possibility.

The average score for each assessment exceeded 70%, which is similar to other efforts that involved students building MIPS processors. However, the process of adding one or two instructions to the processor and monitoring the operation required significantly less time investment from both the instructor and students than building a full processor [9]. The authors did not collect data on the time spent on the assignment, but anecdotal discussions with students suggest approximately three hours or less were required to complete the assignment. The results suggest students were able to gain a similar level of proficiency simply through observing the processor and seeing the data path values change for each instruction.

USAFA requires Computer Science majors to complete 144 credit hours during their four years on top of their other required duties and athletic requirements, which places many constraints on their time. Achieving noticeable improvements in their comprehension and scores with a very small additional time investment is extremely beneficial.

TABLE I. SCORE IMPROVEMENTS

Results by Event and Year	**S17 - 45 Students**		**S18 - 30 Students**		**S19 - 29 Students**	
	Avg	***StdDev***	***Avg***	***StdDev***	***Avg***	***StdDev***
HW	46.89	30.5	79.67	31.7	70.52	27.18
GR	64.71	28.38	70.4	26.52	79.17	18.27
Final	76.16	26.39	84.82	18.1	87.5	20.36

Instructor and student discussions indicate the interactive nature of the simulated processor engaged the students more than the animated PowerPoints and manual translation. The simulated processor also allowed the students to examine different instructions being executed beyond the ones specifically demonstrated in class. Additionally, students were able to see the multiple layers of abstraction in the form of sub-circuits being used to create more complex circuits. This created a stronger instructional linkage between the digital logic portion of the class and the instruction set architecture portion. Students were able to observe the function of the processor at the macro level but were also able to drill down to the individual logic gates that were being used to drive the circuit.

V. FUTURE WORK

Due to feedback from 2019 students, the instructor offered the students a partially instrumented version of the processor in advance of the lectures on the MIPS processor. The goal was to have the students use the model to follow the examples during the lecture and evaluate whether that teaching technique would further boost comprehension. Due to COVID-19 the students still received the Logisim processor and were encouraged to use it while engaged in asynchronous distance learning activities. However, the assessment mechanisms changed significantly, making it impossible to draw direct comparisons with the previous measurements. Student feedback was largely positive.

Other options for future research include having the students program a very simple program on the processor by modifying the instruction memory. This would allow them to better link what they are observing on the processor with their more complex program in MARS. This could be extended by requiring the students to implement additional instructions beyond the `addi` and jump instructions. Other enhancements could include modifying the current single cycle implementation to a five-stage pipelined processor. Future work will include an ARM version of the simulated processor.

VI. CONCLUSION

After students modified and observed the operation of a Logisim MIPS processor, their performance on subsequent assignments and exams improved significantly over previous years. The students showed the greatest improvement on the first homework assignment, improving between 23% and 33% in their ability to correctly fill in the values of the MIPS datapath for a given instruction. Performance on the exams improved between 6% and 14%, showing a significant increase in comprehension of the MIPS processor at the end of course. The gains in comprehension have been observed for two years after the introduction of the simulated processor, across two different instructors. Further, the resulting average of student scores on the assessments is similar to other programs that have their students design an entire MIPS processor, with a fraction of time investment. The Logisim MIPS processor offers a valuable capability for instructors and students to monitor the execution of a MIPS SCA all the way from the macro processor level down to the individual logic gates. The interactive nature of the processor provides a distinct advantage over traditional PowerPoint demonstrations and increases overall student engagement and comprehension.

ACKNOWLEDGMENT

Major Birrer thanks his wife and daughter for their support throughout his career.

REFERENCES

[1] Burch, Carl. "Logisim: a graphical system for logic circuit design and simulation. " *Journal on Educational Resources in Computing (JERIC)* 2. 1 (2002): 5-16.

[2] Vollmar, Kenneth, and Pete Sanderson. "MARS: an education-oriented MIPS assembly language simulator. " *Proceedings of the 37th SIGCSE technical symposium on Computer science education*. 2006.

[3] D. Harris, S. Harris Digital Design and Computer Architecture, Second Edition Waltham, MA: Elsevier, 2013

[4] Hennessy, John L. , and David A. Patterson. Computer architecture: a quantitative approach. Elsevier, 2011.

[5] Donaldson, John L. , et al. "Illustrating CPU design concepts with DLSim 3. " 2009 *39th IEEE Frontiers in Education Conference*. IEEE, 2009.

[6] Araujo, Marcio Roberto Dias, et al. "MIPS X-Ray: A MARS Simulator Plug-in for Teaching Computer Architecture. " *International Journal of Recent Contributions from Engineering, Science & IT (iJES)* 2. 2 (2014): 36-42.

[7] Kabir, Md Tahsin, Mohammad Tahmid Bari, and Abul L. Haque. "ViSiMIPS: Visual simulator of MIPS32 pipelined processor. " 2011 *6th International Conference on Computer Science & Education (ICCSE)*. IEEE, 2011.

[8] Schuurman, Derek C. "Step-by-step design and simulation of a simple CPU architecture. " *Proceeding of the 44th ACM technical symposium on Computer science education*. 2013.

[9] Wang, Lisheng, et al. "Research on Multi-Cycle CPU design method of computer organization principle experiment. " 2018 *13th International Conference on Computer Science & Education (ICCSE)*. IEEE, 2018.