Active Learning Materials for Computer Architecture & Organization: Final Report

BRANDON MYERS, University of Iowa

1 INTRODUCTION

Surveys of computer science and engineering instructors have indicated that two of the common reasons they do not adopt research-based instruction strategies are preparation time and lack of materials [1], [2]. This lack of materials to support research-based instruction was evident in Computer Architecture & Organization (AR). In this SIGCSE Special Project, we proposed to create and disseminate new activities for AR. The proposed outcomes of the project were:

- Eight new, piloted and revised activities for AR
- Expansion of the scope and number of activities to support research-based instruction available to AR instructors by sharing online

We met these outcomes, having written ten activities, piloting six of them during three semesters and the other four during one semester. The activities are publicly available at https://bmyerz.github.io/pogil-for-computer-organization/.

2 THE ACTIVITIES

We based our activities on an instruction strategy called Process-Oriented Guided Inquiry Learning (POGIL). POGIL is built upon cooperative learning and constructivism. A POGIL activity consists of one or more trips through a learning cycle—a procedure that follows a theory of cognition [3]—of *exploration*, *concept invention*, and *application*. We define each phase of the learning cycle in Table 1. And, for each of these phases we list examples from *Procedure Calls* and *Addressable Memories*, two of our activities.

To compose a new activity, we first listed its major learning objectives. From these we worked backward [4], determining *application* questions, identifying the new *concepts* required, identifying a model whose *exploration* would reveal that *concept*, and identifying questions that guide *exploration* of the model. Table 2 lists the ten activities, along with their main learning objectives. The learning objectives are intended to assist instructors in deciding whether the activity might fit into their course.

We piloted each activity in up to three semesters of the *Computer Organization* course at the University of Iowa. Dr. Myers taught the course in student-centered active learning *TILE* (Transform, Interact, Learn, Engage) classrooms [5]. We worked with 33 students in Summer 2017, 68 students in Fall 2017, and 51 students in Spring 2018. Students were primarily third and fourth year computer science majors. Table 2 lists the semesters during which each activity was piloted.

Piloting revealed two main ways to revise our activities. First, we often needed to clarify or decompose the guiding questions in the exploration phase of an activity. These questions guide students to construct new knowledge—a demanding task for students. Therefore, good activity design requires iteration to find the balance of challenge and frustration. An example of this type of revision is shown in Figure 1, which illustrates the addition of a comprehension question to reduce frustration. Second, we recognized the benefit of distinguishing (using asterisks) divergent questions. These open-ended questions, such as “If pipelining reduces the longest delay in a circuit, then why don’t we use an infinite number of pipeline stages?”, ask the student to connect new concepts to a broader context. The instructor can make the divergent questions optional for slower teams to help synchronize the class.

Table 1. Phases of the learning cycle and how each applies to AR.

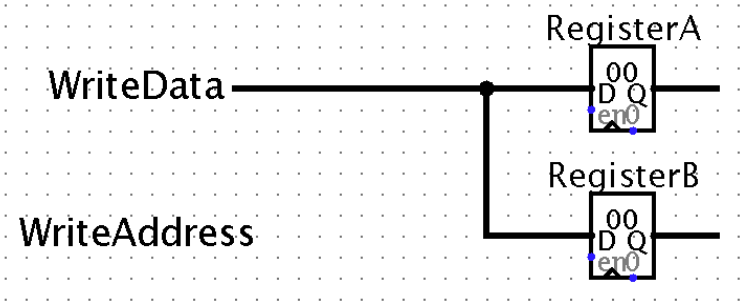
Phase of learning cycle	Definition of phase	Examples in AR activities
Exploration	Students answer comprehension and about a *model*. A model can be any form of information, such as a plot, table, flowchart, game, code, or circuit. The questions are convergent, that is, they lead toward a specific conclusion.	**Procedure calls:** The model is a series of proposed code snippets for implementing a multiply-by-2 procedure call in assembly. Students answer questions about the limitations of each proposal. **Addressable memories:** The model is a digital circuit made from two registers and inputs that say which register to write to. Students answer questions about how the circuit behaves under certain inputs.
Concept invention	Students recognize patterns or synthesize information in the model. The activity assigns the new concept a name.	**Procedure calls:** Jumping to an address stored in a register is called an *indirect jump*. Specific registers are used as *argument registers* and *return value registers*. **Addressable memories:** The circuit that chooses which register to write to is called a *decoder*, and the overall circuit is a *writable memory*. Its parameters are *width*, the size of an entry, and *depth*, the number of entries.
Application	Students apply the new concept to other problems and also answer divergent questions to connect the ideas to the broader context.	**Procedure calls:** Presented with new procedures written in assembly, the students interpret the code. Eventually, students write assembly code for procedures themselves. **Addressable memories:** The students design writable memories with different widths and lengths.

Table 2. The activities written and piloted. Summer 2017 (su17), Fall 2017 (fa17), Spring 2018 (sp18).

Activity title	Learning objectives	Pilots
Bits and numbers	• Translate integers and fixed-point numbers between bases • Express positive and negative integers in two's complement • Identify the largest and smallest integers representable using N bits	su17, fa17, sp18
Memory organization of programs	• Describe the purpose of data segment, text segment, stack, and heap memory • Draw a diagram of memory contents for an executing program • Translate object-oriented code to assembly language	su17, fa17, sp18
Stored programs	• Discuss the correspondence between assembly language instructions and binary machine code • Read the assembly language/machine code documentation • Translate arithmetic and load/store instructions between assembly language and machine code • Translate labels to addresses for branch and jump instructions	su17, fa17, sp18
Procedure calls	• Explain the importance of indirect jumps, argument registers, and return registers in procedure calls • Use procedure calling convention • Trace a recursive procedure call using memory diagrams • Write assembly code defining and calling a procedure	sp18
Combinational logic	• Convert between a truth table and Boolean equation • Write the truth table for a circuit using a switch model • Explain how a circuit-controlled switch is necessary for composability	sp18
Adders and delay	• Use truth tables to build arithmetic circuits • Explain the need for procedural reasoning in design of larger circuits like adders • Relate delay in RC circuits to a simple model for delay • Apply the simple model for delay to a combinational circuit	sp18
Adders, shifters, multipliers	• Compare the delay of various implementations of arithmetic circuits • Build variable bit shifters using various approaches • Build a multiplier from shifters and adders	su17, fa17, sp18
Sequential logic	• Identify properties of a clock signal • Write the waveform for a sequential circuit • Explain why sequential components are required in a feedback loop • Design a basic sequential circuit from a description of behavior	sp18
Addressable memory and the add instruction	• Build an addressable RAM from registers or smaller memories with fewer ports • Build a simple datapath that can execute a single instruction and program it • Modify the datapath to support a second instruction and program it	su17, fa17, sp18
Engineering digital systems	• Calculate the delay of the critical path in a synchronous circuit, and use it to determine minimum clock period and throughput • Plot and interpret a Pareto optimal curve of delay vs area • Describe the advantages and limitations of pipelining	su17, fa17, sp18

EXCERPT

2. Our writeable memory shown below is missing logic regarding the WriteAddress.



a) Give the Boolean equation for the Enable input of RegisterA.

b) Give the Boolean equation for the Enable input of RegisterB.

c) Now, complete the above circuit diagram by converting those two Boolean equations to gates (ignore right side of registers).

Suppose we built a writeable memory with W=8 ~~4~~, L=4, with registers named R0-R3.

3. How many bits are needed for
   a) the WriteAddress input?
   b) the WriteData input?

4. ~~3.~~ To design the writeable memory, answer the following.
   a) Give the Boolean equation for the Enable input of register R0.

   b) Give the Boolean equation for the Enable input of register R1.

   c) Given the Boolean equation for the Enable input of register R2.

   d) Given the Boolean equation for the Enable input of register R3.

Figure 1. Example of a revision (red strikethrough is removal, green is addition) that was the outcome of piloting with students. This excerpt is from an exploration phase of *Addressable memory and the add instruction*. Here, the students are constructing the implementation of writeable memory. In the first pilot we found students struggled on #3 (the new #4). It turned out it was helpful to precede it with a question (the new #3) re-assessing their comprehension of what the memory parameters W and L meant.

3 DISSEMINATION

We made the six mature activities (piloted multiple times) publicly available. The activities can be accessed at https://bmyerz.github.io/pogil-for-computer-organization/.

We presented a poster titled *POGIL Activities for Computer Architecture & Organization* at SIGCSE 2018. The poster attracted constant foot traffic from attendees who teach computer organization, computer architecture, and digital design.

We presented a talk by that title to two different audiences. The first was at the Iowa Undergraduate Computer Science Consortium (IUCSC), an annual meeting of computer science instructors from regional colleges. The second was at the STEM Collaborative Symposium, a meeting of K-20 STEM educators concerned with collaboration across grade levels, across disciplines, and between research and practice. The fact that POGIL has been adopted widely made the talk appropriate for that venue.

4 FINANCES

Spending followed the proposed budget of $5000 to pay for activity writing and revision. The project required one person-month of effort: 2 weeks of Summer pay were covered by the grant and the remaining 2 weeks were covered by the investigator's scholarship time during the academic year.

5 RESOURCES FOR AUTHORSHIP

Members of the SIGCSE community who are interested in writing their own activities have several resources.

- cspogil.org [6] hosts over 200 examples of POGIL activities for CS.
- The POGIL project provides resources for authors [7].
- Kussmaul [8] enumerates patterns of POGIL activities, with examples drawn from CS.
- The POGIL Writers' Retreat is a multi-day workshop for authors to get peer and expert feedback [9].
- POGIL 3-day Regional workshops include an activity writing track [10].

6 FUTURE CONTRIBUTIONS

We plan to continue piloting, revising, writing, and sharing activities. To this end, Dr. Myers will attend the 2018 POGIL Writers' Retreat. There he will write and improve POGIL activities with expert mentors and peers. In the upcoming academic year, we will pilot the existing and new activities with up to another 240 students. We will continue to share resulting improvements and additions in the online repository.

7 CONCLUSION

We thank the SIGCSE Board for their generous support of this project. It has produced principled activities supporting instruction in Computer Architecture & Organization. Just as the public repository cspogil.org inspired this project, we hope our contribution inspires other instructors to write and share materials. A growing number of high-quality materials may encourage more instructors to adopt research-based instruction strategies like POGIL.

REFERENCES

[1] J. E. Froyd, M. Borrego, S. Cutler, C. Henderson, and M. J. Prince, "Estimates of Use of Research-Based Instructional Strategies in Core Electrical or Computer Engineering Courses," *IEEE Trans. Educ.*, vol. 56, no. 4, pp. 393–399, Nov. 2013.

[2] H. H. Hu, B. Knaeble, and C. Mayfield, "Results from a Survey of Faculty Adoption of Process Oriented Guided Inquiry Learning (POGIL) in Computer Science," in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, 2016, pp. 186–191.

[3] M. R. Abraham, "Inquiry and the learning cycle approach," *Chem. Guid. to Eff. Teach.*, vol. 1, pp. 41–52, 2005.

[4] L. D. Fink, *A Self-Directed Guide to Designing Courses for Significant Learning*. Jossey-Bass, 2003.

[5] S. Ingram, B., Jesse, M., Fleagle, S., Florman, J., & Van Horne, "Transform, Interact, Learn, Engage (TILE): Creating Learning Spaces that Transform Undergraduate Education," in *Cases on Higher Education Spaces: Innovation, Collaboration, and Technology*, Hershey, PA: IGI Global, 2013, pp. 165–185.

[6] "CS-POGIL." [Online]. Available: http://cspogil.org. [Accessed: 19-Jun-2018].

[7] "POGIL | Authoring Materials." [Online]. Available: https://pogil.org/authoring-materials. [Accessed: 19-Jun-2018].

[8] C. Kussmaul, "Patterns in Classroom Activities for Process Oriented Guided Inquiry Learning (POGIL)," *Conf. Pattern Lang. Programs*, vol. 23, 2016.

[9] "The POGIL Project - 2018 POGIL Writers' Retreat." [Online]. Available: https://events.pogil.org/event-2783495. [Accessed: 19-Jun-2018].

[10] "The POGIL Project - 2018 North Central Regional Workshop." [Online]. Available: https://events.pogil.org/event-2790302. [Accessed: 19-Jun-2018].