Discussion-like Lab Sections for the UC Berkeley CS61C: Great Ideas in Computer Architecture Course

Connor (Cece) McMahon
Dan Garcia, Ed.
Lisa Yan, Ed.

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-164
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-164.html

May 12, 2023

Discussion-like Lab Sections for the UC Berkeley *CS61C: Great Ideas in Computer Architecture* Course

by Connor McMahon

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Teaching Professor Dan Garcia
Research Advisor

Date

* * * * * * *

Teaching Professor Lisa Yan
Second Reader

Date

Abstract

Discussion-like Lab Sections for the UC Berkeley *CS61C: Great Ideas in Computer Architecture* Course

by

Connor McMahon

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Teaching Professor Dan Garcia, Chair

This Master's report details the restructuring of the laboratory sections in the UC Berkeley *Great Ideas in Computer Architecture* (CS61C) course. CS61C is the third course in the introductory computer science sequence, typically taken by students in the second year of their undergraduate studies. The course has a laboratory component that supplements lecture content and teaches students how to use tools needed to complete the course projects (e.g., *GDB*, *Venus*, and *Logisim*). The goal of this project was to address three areas of concern within the existing lab structure: student achievement, lab completion time, and student experience in the lab environment. These concerns were addressed by creating discussion-like lab sections that promoted active learning and peer instruction. An analysis of the impact of these new sections revealed that they greatly improved student learning outcomes, increased the percentage of students who completed the lab assignments in the allocated time, and generated a collaborative lab environment.

Contents

1 Introduction

At UC Berkeley, *Great Ideas in Computer Architecture* (CS61C) is the third course in the introductory CS sequence typically taken by students in the second year of their undergraduate studies.

The laboratory component of the course was designed to give students hands-on practice in an environment where they can actively discuss course content with their peers – an experience that is unavailable to students during lectures. The material in the lab assignments supplements the lecture content and includes practice with C programming, RISC-V programming, synchronous digital systems, and parallel programming. Additionally, the lab assignments teach students how to use the tools needed to complete the course projects (e.g., *GDB*, *Venus*, and *Logisim*). Students may complete each of the weekly lab assignments during a lab section, where they can receive help from student assistants, or outside of the scheduled lab sections. Once students have completed the lab assignment, they submit it online to be auto-graded and then complete a verbal checkoff with a lab student assistant during a lab section to receive credit.

In the Fall 2020 and Spring 2021 semesters, the students and student assistants consistently reported three problems with the lab assignments: many students failed to meet the lab's learning objectives, exceeded the estimated time to complete lab assignments, and found the lab environment not to be conducive to learning.

To measure the extent of these issues, I conducted a thorough analysis of the labs in Fall 2021. The results of this analysis confirmed that the presence of the issues reported by students and student assistants was pervasive. In most labs, fewer than 30% of students participating in the analysis met the learning objectives. Next, more than half of the students well exceeded the estimated lab completion time. Finally, the lab environment was unfavorable for learning due to a lack of structure and long wait times.

In the same semester, I also rewrote Labs 1 and 2 as I did not want to wait until the lab analysis was completed before beginning to address the reported problems. I developed new exercises for these labs that better support student learning including *worked examples* and *guided problem-solving* activities. The TAs reported that these changes improved student learning outcomes. However, the new assignment specs did not enable students to finish the lab in the allocated time, nor did they produce a positive lab section environment. Therefore, I realized that deeper changes were needed to address these problems.

To address these problems in the following semester, I introduced a new lab structure that transformed the lab environment from an *open-lab* setting to a *closed-lab* setting. In an *open lab*, students work at their own pace, usually individually. There is a lab space staffed by student assistants who are available to help with any questions. By contrast, *closed labs* are scheduled for specific times, encourage student interactivity, and are led by a facilitator who guides the students through the lab. The *closed-lab* structure I implemented follows a *discussion-like*

structure, which as the name implies resembles traditional discussion sections. This report will describe the development of this new structure and an analysis of its efficacy.

Chapter 2 discusses related work. Chapter 3 provides background information on CS61C that is needed to understand the decisions made to improve the labs. Chapter 4 describes the motivation for this project. Chapter 5 details the work completed in Fall 2021 including the preliminary analysis of the labs, the process of rewriting Labs 1 and 2, and the results of rewriting Labs 1 and 2. Chapter 6 describes the implementation of the discussion-like lab sections. Chapter 7 provides an analysis of the results of the new lab structure. Chapter 8 discusses recommended future work. Finally, Chapter 9 concludes the report.

2 Related Work

Lab assignments have been an integral component of CS courses for several decades. In this section, I explore the history of labs in CS courses and analyze the literature on varying lab implementations. Specifically, I compare *open labs* and *closed labs*, and discuss the effects of collaborative learning and worked examples on student achievement.

2.1 History of Laboratory Assignments in Computer Science

In 1989, the Joint Association of Computing Machinery – Institute of Electrical and Electronic Engineers (ACM-IEEE) Curriculum Task Force recommended that introductory computer science courses should be supported by extensive laboratory work (Denning et al., 1989). The report stated that labs should be carefully planned and supervised. Students should attend labs at specified times, and labs should be carried out under the guidance of a lab instructor who ensures that each student follows the correct methodology. Laboratory assignments should allow the average student to complete the work in the allocated time. Organized laboratory sessions should supplement, not replace, the usual programming and other written assignments.

There was initial hesitation to adopt laboratory work due to the the time investment required to develop laboratory content (Knox, 1997). As more laboratory materials were developed and shared among universities, lab sections became commonplace in computer science courses – and continue to play a critical role in computer science education today (McCauley and Manaris, 2002). Numerous studies have compared different laboratory structures and analyzed the effects of collaborative and cooperative learning in laboratories. These studies will be described in greater detail in the next subsections.

An original motivating factor for adding laboratories to CS courses was that labs provided an opportunity for students to actively engage with the material in an environment where they could discuss their ideas and problems with their peers and the instructor – an opportunity that did not exist in lectures (Knox et al., 1996). These learning techniques – now referred to as *active learning* and *peer instruction* – are more beneficial to student learning outcomes than direct instruction from traditional lectures (Prince, 2004; Crouch and Mazur, 2001; Freeman et al., 2014). These findings have sparked the widespread movement to incorporate active learning and peer instruction into lectures. Some departments are taking this movement a step further by replacing their traditional lecture-hall-style rooms with active learning rooms that enable easier collaboration among students (Beichner et al., 2007). These active learning rooms and accompanying lecture structure resemble a laboratory structure – a structure that has proven to be so successful that labs have gone from being supplementary to lectures to replacing them.

2.2 Open vs. Closed Labs

Two modes of administering lab assignments exist: *open labs* and *closed labs*. Open laboratories are like traditional programming assignments, where students are assigned problems to work on at their own pace, usually individually (Prey, 1996). Lab assistants, often senior undergraduate students, staff a lab space where students can receive help as needed (Newby, 2002). By contrast, closed laboratories are scheduled for specific time slots, encourage partner or group work, and have a facilitator who guides the students through the lab.

Thweatt conducted a study analyzing learning outcomes for students in an open laboratory or closed laboratory (Thweatt, 1994). Students in both groups were given the same lab assignment, but students in the closed lab were required to start and complete their assignments in an assigned 2-hour lab section led by a course staff member. Students in the open lab were instructed to complete the lab on their own time. Statistical analysis of students in each lab showed that the means of comprehensive exam scores were higher for students in the closed lab during both semesters during which the study was conducted, but the improvement was not statistically significant. However, the aggregate scores across the two semesters did show a statistically significant improvement in student performance on the comprehensive exam.

Wu, Lin, and Hsu (1997) analyzed the effects of open vs. closed labs on students in a CS2 course. Participants were randomly assigned to either a closed or open lab group. Students in the closed lab group attended three closed lab sessions while students in the open lab group were instructed to complete out-of-class assignments similar to the closed lab assignments. A post-lab achievement test was administered to all participants and the results demonstrated that the closed lab group had higher learning outcomes than the open lab group.

Another study compared the course outcomes regarding the attitude and achievement of students in open vs. closed laboratories (Newby, 2002). The results did not show a significant difference in the means of achievement between both groups. However, using the Attitude to Computers and Computing courses questionnaire (Newby and Fisher, 1997), the study found that students in the closed laboratory experienced lower mean anxiety levels than students in the open laboratory. The author suggests that lowered anxiety in closed labs could improve student achievement based on a study that demonstrated a significant association between computer anxiety and achievement as measured by performance on computing assignments (Marcoulides, 1988).

The evidence that closed labs are better for student achievement than open labs led me to transform CS61C's open-lab environment into a closed-lab environment. When researching types of closed lab designs, I could not find any extensive literature analyzing the effectiveness of different design choices. A literature review on closed-lab design choices found insufficient evidence to determine the impact of design choices on student achievement because the majority of the literature on closed lab designs are case studies that lack the framework needed to accurately measure pedagogical outcomes (Mihail and Roy, 2016).

2.3 Collaborative Learning and Cooperative Learning

Collaborative learning is an instructional method in which students work together in small groups to achieve a common goal (Smith and MacGregor, 1992). By contrast, *cooperative learning* is a structured form of group work where roles and responsibilities are clearly defined to create a strong sense of accountability (Johnson & Johnson, 1987). Some researchers view collaborative learning as encompassing all group-based instructional methods, including cooperative learning, while others argue that collaborative and cooperative learning fundamentally differ (Scager, 2016; Bruffee, 1995). In this review, collaborative and cooperative learning will be grouped together under the term collaborative learning.

The positive effects of collaborative learning on student achievement have been demonstrated in decades of research (Slavin, 1980; Lou, 2001; Johnson et al., 2007). In addition to improving student achievement, collaborative learning can positively affect student self-efficacy, attitudes, motivation, and interest (Johnson & Johnson, 1989; Soh et al., 2005), which can further improve students' learning outcomes (Lishinski et al., 2016).

The use of collaborative learning in CS Education began to spread in the 1990s as CS Education researchers emphasized that traditional CS instruction, which centered on individual learning, left students without the teamwork skills needed to be successful in their careers (Sabin & Sabin, 1994; Yerion & Reinhart, 1995; Prey, 1996). Many universities have integrated collaborative learning in their computer science programs through laboratories. Multiple studies on the effects of collaborative learning in CS laboratories have demonstrated improved student learning outcomes (Wu, 1999; Soh et al., 2005).

One method of incorporating collaborative learning into the laboratory setting is through pair programming. A study on the effectiveness of pair programming in closed CS1 labs divided students into a pair programming section and a solo section (Williams et al., 2002). Students in the pair programming section achieved higher success rates than students in the solo section. "Success" was defined as finishing the course with a C or higher. A chi-square test revealed that the difference in success rate was statistically significant. Additionally, the instructor observed that the questions posed by the pair programming students were more advanced (e.g., hypothetical applications of the concepts in other scenarios) compared to the solo lab, where the instructor spent more time answering basic questions. Students in the pair programming lab more easily resolved problems amongst themselves without the instructor's assistance. In contrast, students in the solo lab had more difficulty interacting with each other because everyone was at different points in their work, and interaction would disturb their progress.

The demonstrated positive effects of collaborative learning led me to incorporate more collaborative learning into CS61C's lab sections to improve student learning outcomes.

2.4 Worked Examples

Research suggests that students struggle to learn introductory programming concepts because the methods used to teach the material lead to cognitive overload (Garner, 2002). Cognitive overload

occurs when a student's cognitive load (i.e., working memory) is exceeded (Sweller, 1988). Cognitive Load Theory states that cognitive overload thwarts learning and should be avoided when designing instructional systems. The theory suggests that the conventional technique of teaching students new skills through solving problems is ineffective because students have not yet developed the schemas to complete the problems. Without these schemas, problem-solving on its own leads to cognitive overload. To avoid cognitive overload and help students develop proper problem-solving schemas, worked examples should be presented alongside problem-solving activities. Extensive research has shown that if students are split into two groups, where one group is given worked examples and the other is given problem-solving activities, the students given worked examples can solve similar subsequent problems more rapidly than students in the later group (Sweller, 1985; Ward and Sweller, 1990).

There is little research comparing the effectiveness of worked examples in computer science, however, some work exists exploring different methods of incorporating worked examples into course material. Muldner et al. (2022) conducted a literature review of worked examples in the context of programming activities and identified five ways examples can be integrated into programming activities. These are text-based "static" examples, modeling examples in which a teacher or student generates a solution in real-time, "dynamic" examples with which students can interact using visualization tools, other "dynamic" examples showing the solution in animated form, and incomplete examples showing partial solutions that students are tasked with completing (e.g., Parsons problems).

The effectiveness of worked examples in many other disciplines motivated me to incorporate worked examples into the CS61C laboratories to improve student achievement levels.

3 Course Background

This section provides general background information about the *Great Ideas in Computer Architecture (CS61C)* course, including the course content, student enrollment, course assignments, and the lab structure. CS61C was fully remote Fall 2020 – Summer 2021 due to the COVID-19 pandemic. The difference between the lab structure during in-person and online semesters will be described.

3.1 Course Overview

3.1.1 Course Content

Great Ideas in Computer Architecture (CS61C) is the third course in the introductory computer science sequence typically taken by students in the second year of their undergraduate studies. While not a prerequisite, students entering the course are strongly encouraged to have already taken *Introduction to Computer Science (CS61A,* taught in Python) and *Data Structures (CS61B,* taught in Java) to provide the fundamental engineering background needed to quickly pick up the large array of topics covered in CS61C. The concepts covered in CS61C include

1. Number Representation
2. Floating Point Representation
3. The C programming language
4. The RISC-V assembly language
5. Compilation vs. Interpretation
6. Boolean Algebra
7. Introduction to Synchronous Digital Systems
8. RISC-V Single Cycle Datapath
9. RISC-V 5-stage Pipeline
10. Memory Hierarchy
11. Cache Coherency
12. Data Level Parallelism
13. Thread Level Parallelism
14. Intro to Operating Systems
15. Virtual Memory

Most of these concepts are covered in the lab assignments as detailed in *Section 3.3 Existing Lab Structure*.

3.1.2 Enrollment and Staffing

CS61C is offered every fall and spring semester, as well as during the summer session. In recent years, the course enrollment has been approximately 1,000 students each fall semester, 500 students each spring semester, and 200 during the summer session. The fall and spring semesters are 16 weeks each, the summer session is 8 weeks long (at twice the speed), and the course is typically co-taught by two instructors.

Two types of student assistants are involved with the laboratory assignments: teaching assistants (TAs) – also called (under)Graduate Student Instructors or (u)GSIs – and Academic Interns (AIs). CS61C TAs can be undergraduate or graduate students and hold a variety of responsibilities including leading laboratory sections and authoring laboratory assignments. The number of TAs hired per semester depends on the course budget, which is proportional to the number of students in the course.

AIs are undergraduate students who assist TAs in lab sections by answering students' questions and giving lab checkoffs. The AI position trains students to become TAs by providing the experience of assisting students under the guidance of their assigned TA. The number of AIs is not dependent on the number of students enrolled as AIs receive course units for their work instead of pay.

3.2 Course Components

This section briefly describes each component of the course and their relationships to each other.

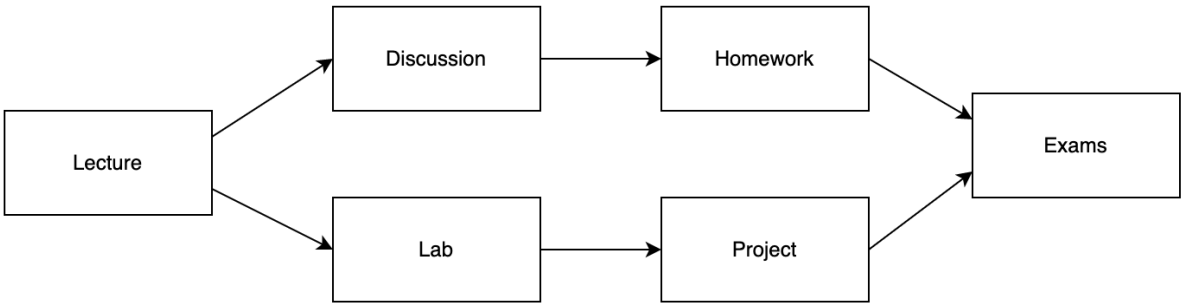

Figure 1. The relationship between each component of the course.

3.2.1 Lecture

Each new topic is first presented to students in lectures. In the fall and spring, students attend 50-minute lectures three times per week or 90-minute lectures two times per week. In the summer, students attend 90-minute lectures four times per week.

3.2.2 Discussion Sections

Due to the large volume of content covered by the course, there is limited time for active learning in lectures. Discussion sections allow students to actively engage with the material covered in the lecture by completing worksheets with their peers under the guidance of a TA. Discussion attendance is optional and does not count toward a student's final grade. In the fall and spring, there is one discussion topic per week with multiple 50-minute time slots for students to choose from. In the summer, there are two discussion topics per week.

3.2.3 Labs

The lab assignments were designed to give students hands-on practice in an environment where they can actively discuss the content with their peers – an experience unavailable to students during lectures. The material in the lab assignments supplements the lecture content and includes practice with C programming, RISC-V programming, synchronous digital systems, and parallel programming. Additionally, the lab assignments teach students how to use tools that are needed to complete the course projects (e.g., *GDB*, *Venus*, and *Logisim*). Students may complete each of the weekly lab assignments during a lab section, where they can receive help from student assistants, or outside of the scheduled lab sections. Once students have completed the lab assignment, they submit it to Gradescope to be auto-graded and then complete a verbal checkoff with a lab staff member (TA or AI) to receive credit. A detailed discussion of the lab material and structure can be found in *Section 3.3 Existing Lab Structure*.

Lab assignments differ from discussion worksheets in that the latter can be solved on paper, whereas the former requires students to write programs or create simulations in software.

3.2.4 Homeworks

Homework assignments help reinforce the material covered in lectures and assist students in preparing for exams. Students are given one week to complete each of the 11 homework assignments. Homework problems are submitted online and auto-graded without individualized feedback.

3.2.5 Projects

The course projects build on top of the lab assignments and give students hands-on practice with the course material. The project topics are programming in C, programming in RISC-V, processor design, and parallel programming in C. Students are given two to three weeks to complete each of the four projects individually or with a partner.

3.2.6 Exams

There are one or two midterms (depending on instructor preference), and one final exam. The midterms and final exam are generally paper-based.

3.3 Existing Lab Structure

Lab assignments were *initially* designed for students to complete within a 2-hour *closed lab* section containing around 30 students. The lab analysis in Section 5 details how the sections shifted from their intended *closed lab* structure to an *open lab* structure. Each lab section had course staff who were available to answer student questions. Students would complete the lab either on their own or with a partner, submit the assignment on Gradescope to be auto-graded, and then complete a verbal checkoff with a lab staff member to receive credit. The online semesters initially required a checkoff, but limited staff resources and immense student stress around checkoffs led to removing lab checkoffs in the middle of Fall 2020. Checkoffs resumed when the course returned in person in Fall 2021. Students could complete the lab assignment inside or outside the lab section, but they were required to go to a lab section to receive a checkoff. Students could attend any lab section to ask questions or receive a checkoff.

3.3.1 Topics

The topic of each lab can be found in Table 1.

Lab Number	Topic
0	Course Intro and Setup
1	Intro to C and GDB
2	Bit Operations, Memory Management, Valgrind
3	Intro to RISC-V Assembly
4	RISC-V Functions and Pointers
5	Intro to Logisim
6	Logisim Pipelining
7	Caches
8	Virtual Memory
9	Data Level Parallelism
10	Thread Level Parallelism

Table 1. Lab topics.

3.3.2 Autograder

Students were given a local copy of the auto-grader for each lab assignment which provided them with immediate feedback on their work. There was no limit to the number of times or how often a student could run their local auto-grader. The local auto-grader was identical to the Gradescope auto-grader, so there were no test cases hidden from the students. To receive credit on the Gradescope assignment, students had to pass all test cases. If students did not pass all test cases, they did not receive credit for the lab assignment.

3.3.3 Checkoff Procedure

When a student was ready to get checked off, they would submit a ticket to the *office hour queue*. The office hour queue is an online tool that was originally developed to keep track of the order in which students requested help during office hours. The tool was also used in CS61C labs to keep track of the students requesting checkoffs and asking lab questions. During the checkoff, the lab staff member would ask each student two questions pertaining to the lab material to ensure that the student had a firm understanding of the content. To ensure fairness and consistency, each lab staff member selected the checkoff questions from a pool of predetermined questions. If the student could answer each question correctly, they received full credit for the lab. If the student did not answer the questions correctly, the lab staff member would provide hints and help the student clarify concepts until they could successfully complete the checkoff. If the student worked with a partner, they would get checked off together, but it was required that each student answer at least two questions without the assistance of their partner to receive credit.

3.3.4 Lab Course Staff

Each lab section was led by one TA and assisted by one to five AIs depending on the average number of students who attended the section. The primary responsibility of each of these individuals was answering questions and delivering checkoffs. The role of the TA and AI differed in that the TA had more expertise in answering conceptual questions and debugging.

4 Motivation

In the Fall 2020 and Spring 2021 semesters, three problems were consistently reported among staff and students.

First, TAs who administered lab checkoffs reported that students were not meeting the learning objectives of the lab assignments as many students were struggling to answer the lab checkoff questions. Additionally, TAs who helped students with projects during office hours frequently stated that students struggled with aspects of the projects that should have been taught in the lab assignments – namely how to use the tools taught by the lab assignments.

Second, students reported through Piazza (the online course forum) and conversations with TAs that lab assignments took longer than the prescribed two hours. Many students reported at least twice this time, with some spending more than 10 hours per lab.

Third, the lab environment in Fall 2020 and Spring 2021 was virtual due to the pandemic, which created a huge challenge for fostering a collaborative environment where students could help each other. Students did not feel comfortable talking to classmates whom they did not know over Zoom, which effectively eliminated all peer-to-peer interaction. Additionally, students reported frustration with long wait times to receive help from course staff on their lab assignments. The fear of neither getting help nor being checked off within the course staff's working hours worried students immensely. Notably, TAs who were present pre-pandemic reported that although student frustration and stress from long queue times were prevalent in earlier, in-person semesters, student sentiment worsened during the pandemic.

To determine the severity of the reported problems, I conducted a thorough analysis of the achieved **learning objectives**, **lab completion time**, and **lab environment** in the Fall of 2021.

In the same semester, I also rewrote Labs 1 and 2 as I did not want to wait until the lab analysis was completed before beginning to address the reported problems. I developed new exercises for these labs that better support student learning including *worked examples* and *guided problem-solving* activities.

5 Fall 2021 Analysis and Rework of Labs 1 and 2

5.1 Lab Analysis

5.1.1 Learning Objectives Evaluation

To determine whether students were meeting lab assignment learning objectives, a study was conducted to measure students' ability to answer pre-existing checkoff questions. The procedure for conducting lab checkoffs is discussed in *Section 3.3.3 Checkoff Procedure*. The pre-existing checkoff questions were chosen because they directly correspond to lab content and require no knowledge outside the lab's scope. In other words, a student who met the learning objectives of the lab should have been able to answer the questions correctly on their first attempt. Students who could not answer a question were provided with hints to help guide them toward the answer. Sample checkoff questions and hints are provided in Table 2.

Lab 4: RISC-V Functions and Calling Convention
Example Question 1: Why must we decrement the stack pointer in the prologue? ● Example Hint 1: What is the prologue? What is the purpose of it? ● Example Hint 2: How do we store data on the stack?
Example Question 2: If a function takes in two arguments via a0 and a1 and has one return value returned through a0, is the value in a1 guaranteed to remain unchanged? ● Example Hint 1: Are the argument registers caller saved or callee saved? ● Example Hint 2: What is the difference between caller and callee saved registers?
Lab 6: Pipelining in Logisim
Example Question 1: Why must we insert additional registers to our datapath to pipeline our design? ● Example Hint 1: Does each stage take the same time to execute? ● Example Hint 2: What would happen if each stage took a different amount of time to execute and we didn't have pipeline registers?
Example Question 2: Which circuit has a higher maximum clock rate: the pipelined circuit or the non-pipelined circuit? Why? ● Example Hint 1: What determines the max clock frequency? ● Example Hint 2: Can you walk me through what happens on each clock cycle in the pipelined and non-pipelined circuit?
Lab 8: Virtual Memory
Example Question 1: If the physical memory size (in bytes) is doubled, how does the number of bits in each entry of the page table change? ● Example Hint 1: What is stored in each entry of the page table? ● Example Hint 2: How does the number of bits in the physical address relate to the size of the physical memory?
Example Question 2: If the page size (in bytes) is doubled, how does the number of entries in the page table change? ● Example Hint 1: What determines how many entries there are in the page table? ● Example Hint 2: What happens to the number of pages if the page size is doubled?

Table 2. Example checkoff questions and hints.

In the Fall 2021 semester, I measured the checkoff performance of 20 to 25 students for each lab assignment. The selected students varied each week. Each student's comprehension level was placed in one of four categories: 1) answered both checkoff questions correctly without

assistance, 2) answered both questions correctly after 1-2 hints total, 3) answered both questions correctly after 3+ hints total, and 4) was not able to answer one or both questions correctly after 3+ hints total. Hints were given in the form of guided questions. If the student did not know the answer to the guided question, the staff member would help the student arrive at the answer to the guided question. The student would then attempt the checkoff question again. If the student needed assistance with answering the guided question, this did not count toward the total number of hints. Student checkoff performance data can be found in Figure 2. Lab 9 is excluded from the table because there were no checkoffs for Lab 9 due to lab sections being canceled for a university holiday.

The data in Figure 2 shows that for Labs 3-10, fewer than 30% of students could answer the questions correctly without any hints (the rightmost bars). In Labs 3, 4, 7, and 8 almost 40% of students could not answer the questions *even with* hints (the left bars). This data indicates that many students were not meeting the learning objectives of the lab assignments.

Students performed better in Labs 1 and 2 relative to Labs 3-10. This is presumably due to the updates made to Labs 1 and 2 during Fall 2021. Although these metrics are better than those in Labs 3-10, they still reveal that a significant portion of students are not meeting the learning objectives of the assignments.

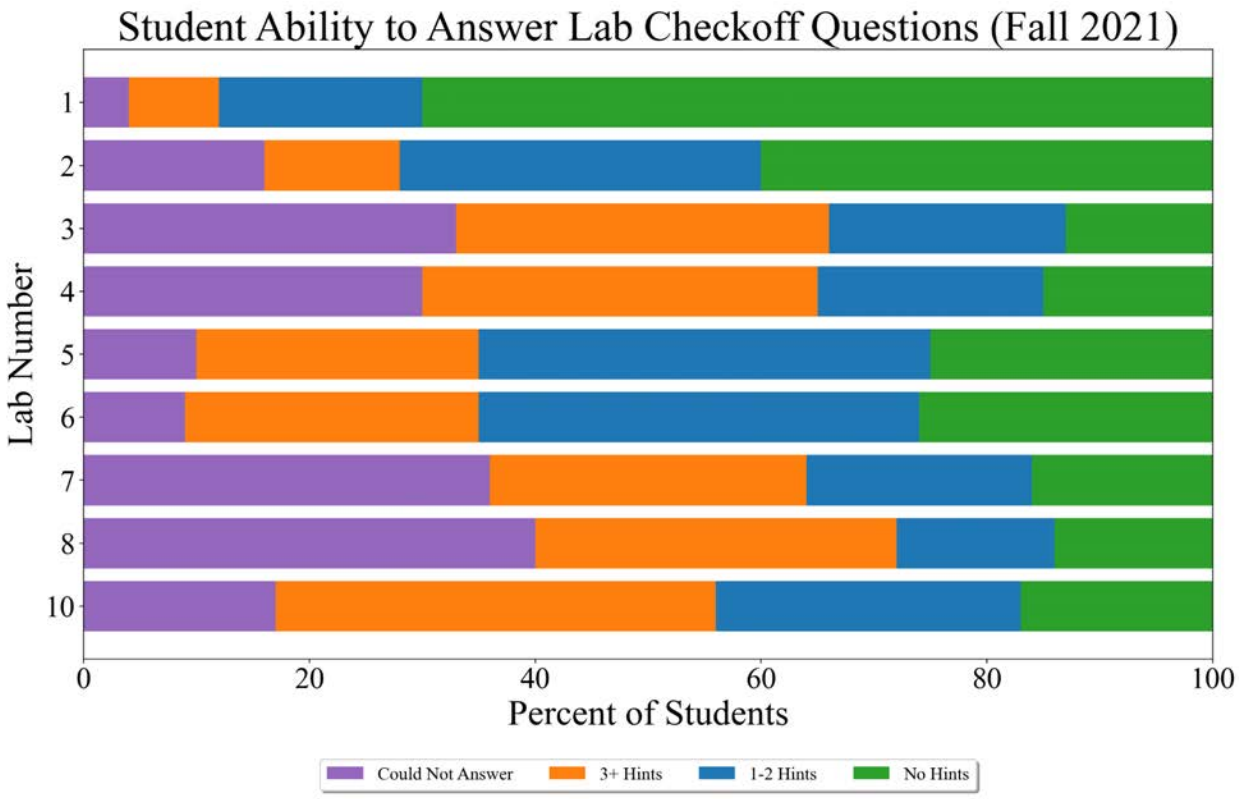

Figure 2. Results from the analysis of student achievement in labs as measured by the number of hints needed to answer checkoff questions.

5.1.2 Lab Completion Time

The analysis of lab completion time was split into *working time* and *checkoff time*. Working time is defined as the time spent actively working on the assignment, excluding checkoff time. Checkoff time is defined as the time between when a student requests and completes a checkoff. Every student reported their working time in a completed survey when submitting their assignment to Gradescope. The checkoff time was measured using the office hour queue tool.

The self-reported working times of each lab can be found in Figure 3. These times do not include the time it took for students to receive a lab checkoff because students submitted the assignments on Gradescope before they completed the checkoffs.

The median checkoff time can be found in Table 3. The table divides the checkoff time into the time spent waiting for the checkoff to begin and the time spent performing the checkoff. Lab 9 is excluded from the table because there were no checkoffs for Lab 9 due to lab sections being canceled for a university holiday.

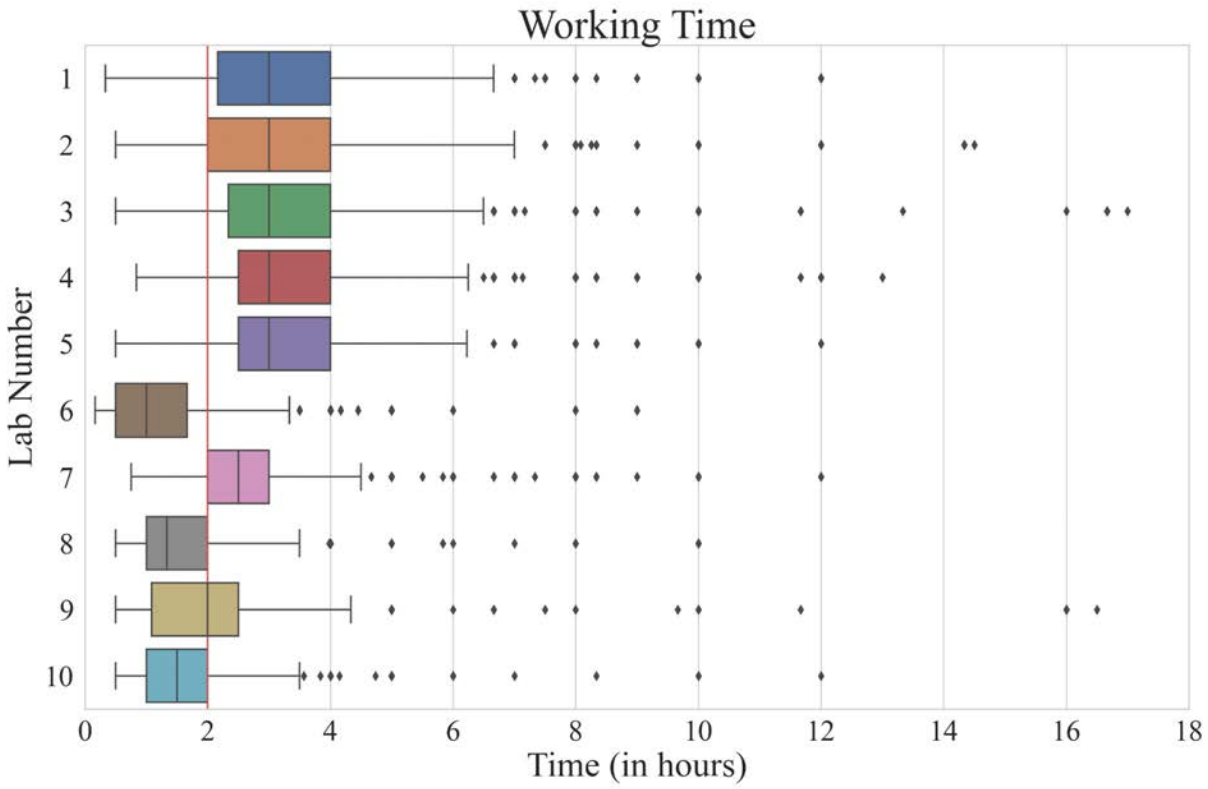

Figure 3. Results of self-reported working time. This figure does not include time spent on checkoffs. The red line indicates the expected time to complete the lab.

Median Time Spent on Checkoffs (in minutes)

Lab Number	Median Time Waiting	Median Time Answering Checkoff Questions	Median Total Checkoff Time
1	25	10	35
2	11	8	19
3	15	9	23
4	13	6	19
5	9	6	15
6	9	5	14
7	15	12	27
8	14	9	23
10	13	8	21

Table 3. The median time students spent on checkoffs split into the time waiting in line to begin the checkoff and the time answering checkoff questions. These times were measured by the office hour queue tool.

Figure 4 displays the *total time* students spent completing the lab (i.e. working time plus checkoff time). The office hour queue did not record identifiable student information, so Figure 4 was generated by adding the median total checkoff time to each student's self-reported working time.

In seven out of ten labs, at least half of the students took longer than the allotted two hours to complete the lab. In five out of ten labs, over one-quarter of students spent more than four hours completing the assignment.

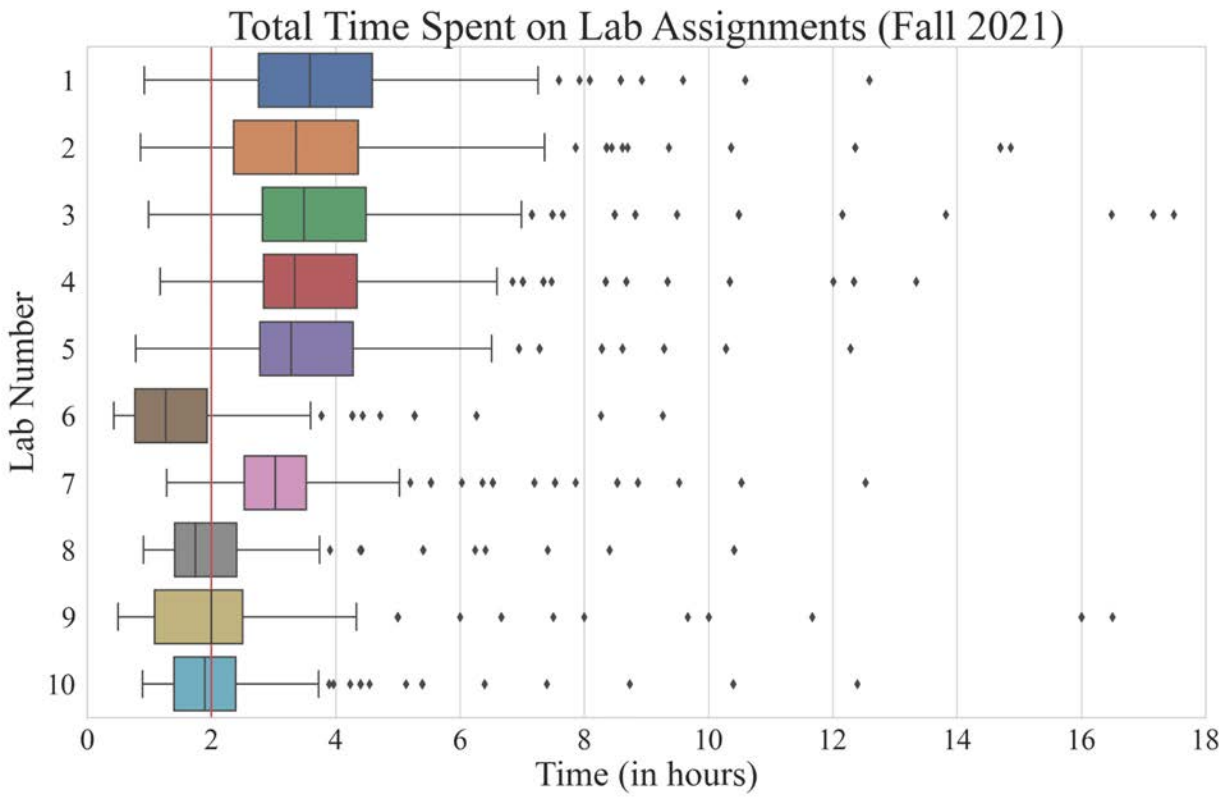

Figure 4. Total time students spent on each lab assignment. The red line indicates the expected time to complete the lab.

5.1.3 Lab Environment

The percentage of students who completed each lab during a lab section can be found in Figure 5. This data was self-reported by students when they submitted their lab assignments to Gradescope. The data show that the majority of students completed the entirety of each lab assignment outside of the lab section. Lab 9 is excluded from the table because lab sections were canceled due to a university holiday.

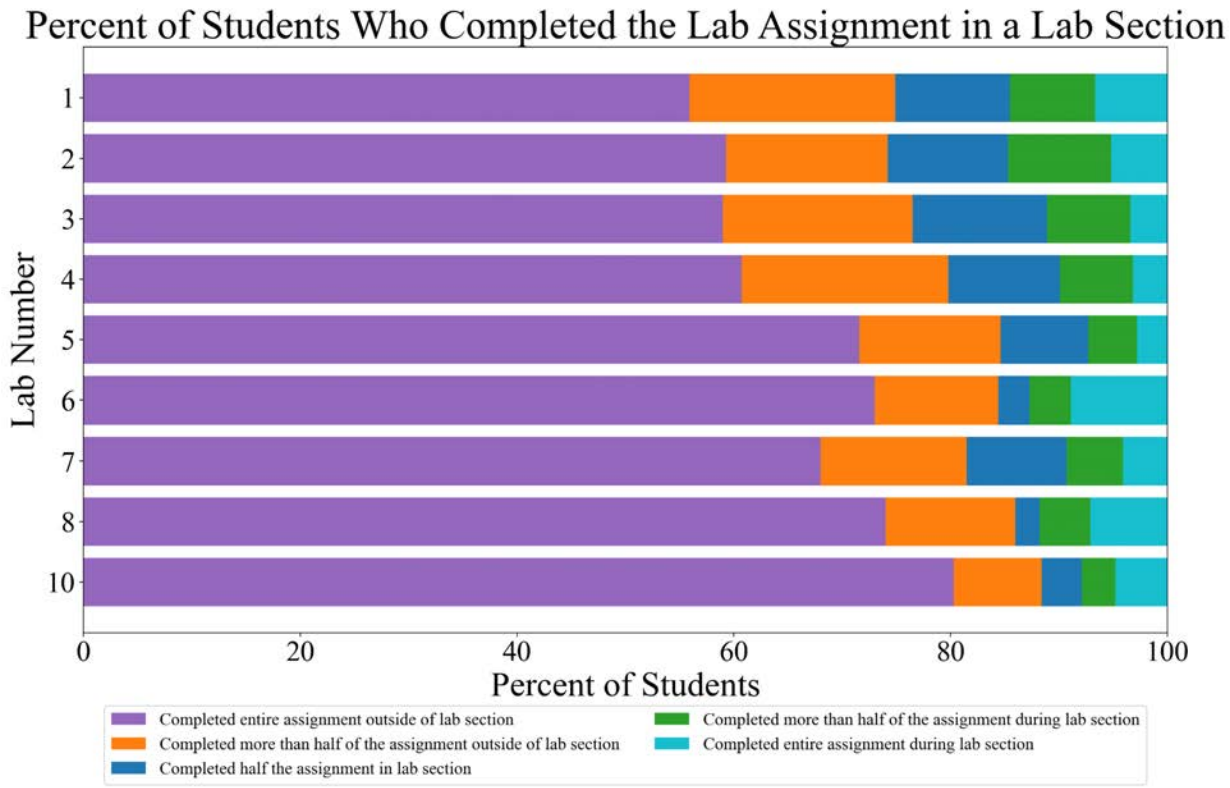

Figure 5. Percentage of students who completed the lab assignments during lab sections.

Students reported that they preferred to complete the lab outside of the section for two main reasons (1) The lab queue was too long, so they preferred to ask for help on Piazza because they could ask a question and then complete other work while waiting for a response, (2) The student preferred to work alone, so they didn't feel the need to attend the lab section.

The wait time to receive help on a lab question as measured by the office hour queue is shown in Figure 6. The data show that for seven out of nine labs, the third quartile of wait times exceeded half an hour. This wait time prevents students from making forward progress and can lead to unproductive struggling and frustration. Note that this wait time would presumably be far worse if more students had come to the lab section to ask questions as opposed to asking them on Piazza.

The number of checkoffs vs. questions that took place during lab sections are reported in Figure 7. In every lab, there were more than twice the number of checkoffs than questions. These statistics demonstrate that lab sections were primarily being used to assess student achievement instead of assisting students with the assignment, which does not align with the original intention of lab sections. Note that the number of checkoff tickets decreased as the semester progressed because students who fell behind with the lab assignments received asynchronous checkoffs that were not recorded by the checkoff system.

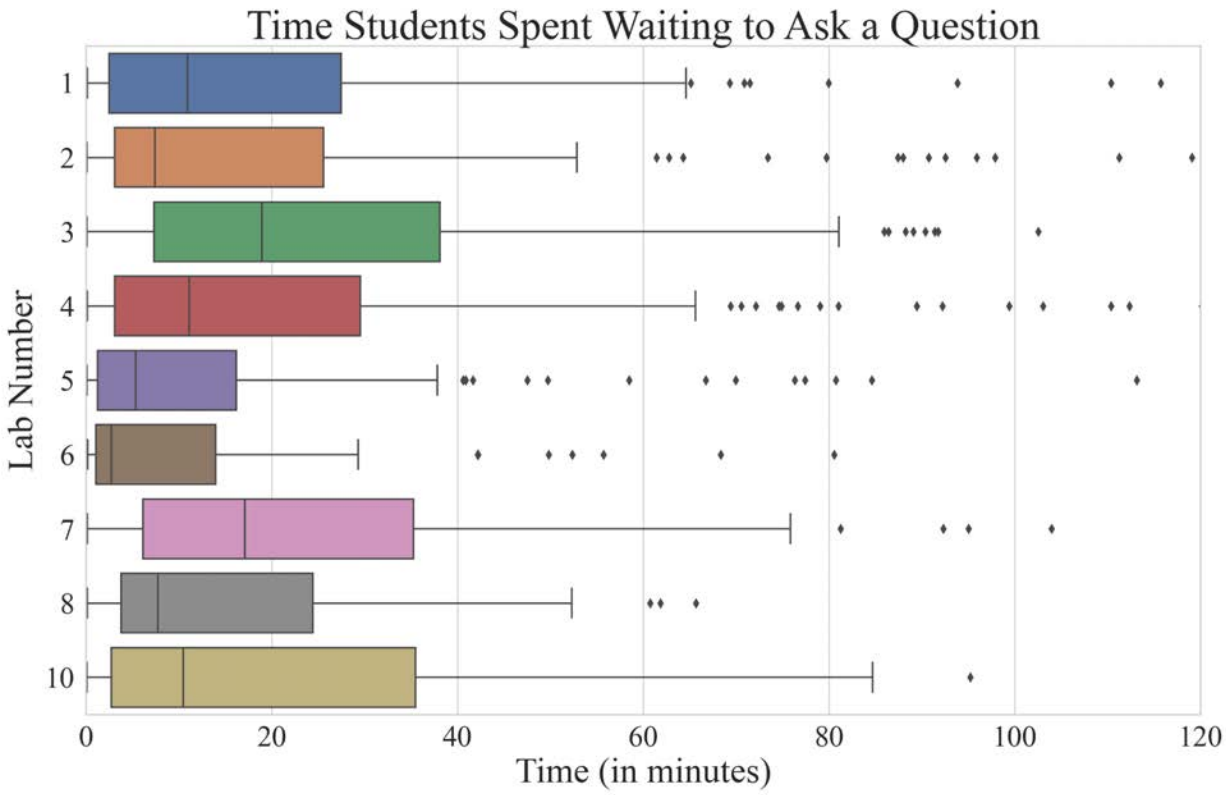

Figure 6. Wait time to receive help on a lab question as measured by the office hour queue system.

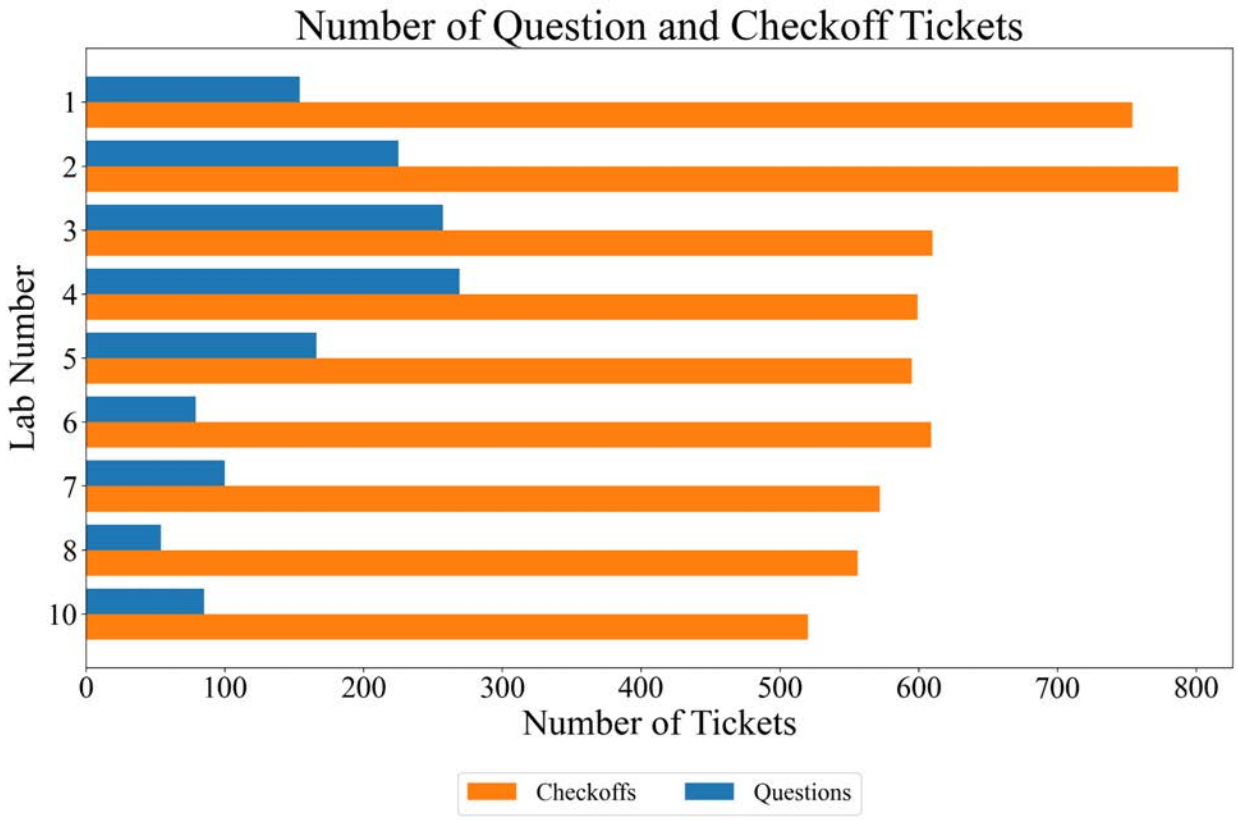

Figure 7. The number of lab question tickets and lab checkoff tickets for each lab as recorded by the office hour queue tool.

At the end of each lab, students filled out a survey asking about improvements that could be made to the lab. Common sentiments included disliking the time it took to complete the labs, the wait time to ask questions and receive a checkoff, and the lack of collaboration in the lab sections. Selected comments are provided below:

- *The lengths of the labs disincentivize collaboration because it's impossible to go to your assigned lab and finish within the 2 hour block. Therefore, I am forced to complete labs on my own in advance because they take up so much time and resort almost immediately to office hours when I get stuck. Makes labs quite lonely and frustrating, especially because they serve as one of the few avenues of collaboration in this class other than the discussion section, which itself isn't frequent enough to foster a collaborative environment.*
- *Please please clarify next semesters that lab time is PURELY for checkoff. There is no way to get practical help when those who are trying to get help have to compete with those who are trying to get checked off. The lab is essentially just homework. It should just be called a Checkoff Section instead of Lab Section.*

- *idk i wish there was more of an expectation of doing the lab in lab rather than just having lab time be when i get checked off, I feel like going to lab is pointless in a way and I have to wait for 30+mins just to get checked off*
- *It is hard to ask a question to a TA or lab assistant when you have a question, because it can be answered in 2 minutes but i have to wait 30 minutes for their attention*
- *Insanely long queue times, including in my own section. Really hard to get help. Creates incentive for students to create meaningless tickets, since by the time they get to them, it's likely they'll have come across an issue regardless.*
- *Labs have been taking forever to do. This is really unfair to my time considering this is a 4 unit class. I am never able to finish lab in-person on time. This in my opinion needs to be changed.*

The lab TAs were also surveyed on their experience. 24 out of 24 lab TAs reported that they felt pressure to spend less time than preferred with each student while answering questions or giving checkoffs due to the long queues. 20 out of 24 lab TAs reported that they were not spending as much time on the checkoffs as they would have liked. Checkoffs were a critical part of the lab because they helped staff members identify if a student had a misconception and spent time with the students ensuring that they understood the material well. However, with the staff being unable to spend as much time as they would have liked on each checkoff, many more students were likely finishing the lab without the staff member having the time to identify and clarify these misconceptions.

5.2 Content Improvement of Labs

Labs 1 and 2 allow students to practice programming and debugging in C. The course lectures focus on fundamental C concepts with which students are unfamiliar: pointers, structs, C strings, and memory management. However, lectures do not discuss the two debugging tools presented in the lab: GNU Debugger (GDB) and Valgrind.

Based on TA accounts in Spring 2021 and Fall 2021, students did not understand how to use GDB and Valgrind to debug programming assignments after these debuggers were taught in the lab. The portions of Labs 1 and 2 that teach students how to use these tools were rewritten to address this issue. The new exercises on GDB and Valgrind and the observed changes in student learning outcomes are discussed below.

GNU Debugger (GDB)

The existing labs did not successfully teach students how to use GDB due to a lack of guidance. The lab assignments presented students with the GDB reference manual and then asked students how to perform various operations (e.g., set a breakpoint in the program). The exercises confused many students about how to use GDB as they had difficulty understanding the manual.

Additionally, students were not given any examples of how GDB can be used to find a bug nor were they given any problems on which they could practice using GDB.

To improve student understanding of GDB, I incorporated worked examples into the assignments. Presenting worked examples with problem-solving activities can improve students' learning outcomes by helping students build the schemas needed to solve problems (Sweller, 1988). A proven method for integrating worked examples into problem-solving activities is to successively interleave the presentation of a worked problem and a similar problem for students to solve on their own. The new lab uses this interleaving technique to teach students how to use GDB.

Students are given a buggy program and are instructed to follow a walkthrough that demonstrates how to use a set of GDB commands to find a bug in the program. Students are then instructed to find an additional bug in the program on their own using the commands they just learned. Next, students are taught an additional set of gdb commands through another walkthrough and finally instructed to find another bug on their own using this new set of commands.

In the last GDB exercise, students follow a walkthrough that provides step-by-step instructions on how to find the cause of a segfault in the program. Students were not given a follow-up problem to practice this skill to keep the lab completion time within the two-hour limit. Although students are not given a practice problem, this worked example on its own can help students develop problem-solving schemas that make them better at solving future similar problems (Sweller, 1985; Ward and Sweller 1990).

5.2.2 Valgrind

The existing labs did not successfully teach students how to use Valgrind due to a lack of guidance. The lab assignments provided students with a program that contained a segfault and asked them to identify where the segfault occurred by reading the code. Students were then prompted to run Valgrind on the program and compare Valgrind's output message to observations they made when reading the code. The lab gave no information about how to read Valgrind's output messages.

The new lab teaches students how to use Valgrind by providing a step-by-step walkthrough on how to run Valgrind and interpret Valgrind's output messages. Explaining how to interpret Valgrind's output messages helps students acquire the schemas they need to interpret the messages on their own in future assignments. As with the segfault GDB example, students were not given a follow-up problem to practice this skill and an attempt to keep the lab completion time within the two-hour limit.

5.2.3 Results of Intervention

I evaluated student understanding of GDB and Valgrind using TA surveys. There were six TAs who were on course staff in Spring 2021 and Fall 2021. The TAs were asked to compare their

perception of the students' ability to use GDB on Project 1 on a one to five scale. The scale is defined below.

5 = significant improvement in student understanding
4 = some improvement in student understanding
3 = no difference in a student understanding
2 = some decrease in student understanding
1 = significant decrease in student understanding

All six TAs reported a significant improvement in the students' ability to use GDB during the first project. This quote is representative of general TA sentiment:

> The new labs have been really helpful with project office hours! Last semester [Spring 2021], when students would come to office hours with a bug in Project 1, I would ask them if they had tried to debug using GDB. Almost every student would tell me that they didn't know how to use GDB, so I would have to spend 20 minutes teaching them how to use it. Now when a student comes to me with a bug on Project 1, almost all of them tell me that they tried to find the bug with GDB and about half of them tell me that they were able to isolate where the bug was using GDB.

TA's perception of student understanding of Valgrind was also positive, but to a lesser extent. Two TAs reported a significant improvement and four TAs reported some improvement. This quote is representative of general TA sentiment:

> More students in Project 1 office hours had attempted to use Valgrind to find their bugs compared to last semester [Spring 2021] before asking for help, but many of them were having trouble understanding how to interpret the output message. People definitely understand it better than last semester, but I think more examples of interpreting the output messages need to be added to the lab.

I presume that the students' difficulty in understanding the Valgrind output messages was rooted in two causes 1) student understanding of memory management was not sufficient 2) the example presented in the lab only covered a few of the possible messages that Valgrind may output. Student misunderstanding of memory management can be attributed to the fact there is only one lecture on memory management and students are not exposed to enough worked examples and practice problems in the lab covering this topic. Additional memory management and Valgrind examples could not be added to the lab in the Fall of 2021 because I was trying to keep the lab completion time within two hours. However, *Section 8* contains recommendations on how more exercises on memory management could be added in the future.

5.3 Conclusion of Fall 2021 Analysis and Pilot

The analysis of the lab assignments in Fall 2021 confirmed that the three problems reported by course staff and students in prior semesters were pervasive. First, most students were falling short of the lab learning objectives, as demonstrated by their inability to answer lab checkoff questions without the assistance of hints. Second, most students exceeded the allotted lab completion time in seven out of ten labs. Third, the lab environment was found to be unfavorable for learning due to a lack of structure and long queue times.

This study also demonstrated that the existing lab model had devolved from its initial *closed lab* setting into an *open lab* setting. Students were no longer completing the lab assignments during a scheduled lab time where they could benefit from working with other students and receiving assistance from a lab course staff member as originally intended. Instead, most students chose to complete the lab assignments outside of a lab section and only met with a lab course staff member to complete their checkoff. The lab sections were primarily used to assess student achievement instead of assisting students with the assignment, which does not align with the original intention of lab sections. Based on student accounts, this shift from a closed lab environment to an open lab environment occurred because students were discouraged from attending lab sections due to long wait times and the desire to work alone.

Rewriting the lab specs for Labs 1 and 2 improved perceived learning outcomes. However, the new specs did not enable students to finish the lab in the allocated time nor did they produce a positive lab section environment. Therefore, altering the remaining lab assignments has the potential to improve students' learning outcomes, but it must be coupled with additional improvements to address the long lab completion times and poor lab environment.

6 New Lab Structure

To address the three problems with the labs, I restructured the lab sections from the existing *open lab* structure to a *closed lab* structure. Research has demonstrated that closed labs have better student achievement outcomes, reduce student stress, and promote a stronger community among students than open labs (Thweatt, 1994; Wu et al., 1997; Newby, 2002). Additionally, a closed lab that is well-paced by a TA and has proper staffing to assist struggling students can improve lab completion times. I implemented the closed labs using the *discussion-like lab structure* described below.

6.1 Discussion-like Lab Section

In a *discussion-like lab structure*, the TA begins the lab by providing a brief review of lecture material relevant to the lab assignment. Then the TA gives students time to work together on the first exercise. The TA and other lab course staff are available to assist the students with questions while they work through the exercise. Once most students have completed the exercise (or once a specified period of time has elapsed), the TA reviews the exercise with the students ensuring that everyone understands how to solve the problem. This process is repeated for all exercises. The effects of discussion-like lab formats have not been explicitly studied; however, components of discussion-like lab sections have been individually demonstrated to improve students' learning outcomes as discussed below.

The TA-led review of each problem reduces student cognitive load and improves students' problem-solving skills. When the TA begins the review, it is possible that some students will not have finished the exercise. Such students are likely experiencing cognitive overload because they have not acquired the schemas needed to solve the problem within the given time. Cognitive load theory asserts that these students would benefit more from viewing a worked example (i.e., the TA's revision of the problem) as opposed to continuing to struggle to solve the problem on their own (Sweller, 1985).

Additionally, the TA-led review can also help students who complete the problem in the allotted amount of time. Such students benefit from exposure to an expert's thought process and problem-solving technique by, for example, potentially identifying areas of their code that can be improved and learning better programming practices and strategies for solving future problems (Bower, 2008).

One con of the discussion-like lab structure is that there are no checkoffs. Working through the labs in the discussion-like format consumes the full two-hour time block which leaves no time for checkoffs. Removing checkoffs was a difficult decision to make as checkoffs allowed the course staff to assess each student and correct misconceptions individually. We foresaw that the benefits of the discussion-like structure would outweigh this con because the new structure allotted more time for students to ask questions and the TA leading the section can highlight common misconceptions with the class as a whole.

6.2 Implementation

6.2.1 Lab Section Structure

Each lab section has about 30 students with one TA and three or four AIs. Each lab room has a projector for the TA to display the lab material. A lab guide was created for each lab to guide the TAs and AIs through how to run the lab section. The guides contain suggestions for how much time the class should spend on each problem, explanations of the lab solutions, and common student misconceptions.

Each lab starts with the TA giving a 5-10 minute review of lecture material relevant to the lab assignment. Slides were created for the TAs to use for this review, but the TAs were also allowed to create their own slides. This review time was used to refresh students on the material and provide a chance for students to ask questions about concepts they were unsure about before beginning the lab. After the review, the TA gives students time to work together on the first exercise. During this time, the TA and AIs walk around answering questions that students have.

Once the TA sees that most of the students have finished the problem or the suggested amount of time for working on the problem is up, the TA discusses the problem with the class. When discussing the problem, TAs are instructed to present their own thought processes and to help students develop better problem-solving skills. To make the explanation more interactive, the TA asks students to help solve the problem. For example, the TA might explain the next step to solve the problem and then ask the students how to implement the particular step. Once all students understand the first exercise well, they begin working together on the next exercise. This process repeats until all exercises are completed.

6.2.2 Lab Section Attendance and Grading

There exists a wide range of abilities among the students taking CS61C that stem from the level of exposure students had to programming prior to entering post-secondary education. Research shows that working in mixed-ability groups is beneficial to both high- and low-ability students as high-ability students benefit from providing guidance to low-ability students, and low-ability receiving explanations from their peers (Lou et al. (1996), Webb (1997), and Webb & Palincsar (1996)). Medium-ability students do not benefit from heterogeneous groups when they do not give nor receive explanations, so these students may benefit more from being in homogeneous groups.

Attendance is not required for the discussion-like labs. The new lab structure initially proposed that lab section attendance be required to ensure that lab sections were attended by students with a wide range of abilities to promote better learning outcomes. However, there was strong pushback among instructors and TAs, who argued that “advanced students” who can complete the lab in less than two hours should not be forced to sit in a two-hour lab section. Despite research demonstrating that collaborative learning in mixed-ability settings benefits high-ability students, fears that such students would be disgruntled by having to sit through a

structured lab section eventually won the day. In the end, we decided on an optionhow much policy.

In Spring 2022, fewer lab sections were offered under the new system. It was estimated that about half of the students would choose to attend the lab sections, so eight lab sections were created to support half of the 500 students enrolled in the course. Six of these lab sections were in-person, and two were held online, as some students were still worried about attending in-person activities due to the pandemic. Halving the number of lab sections meant that there was now momore staff timeted to other areas of the class, including office hours (which often had queues that were several hours long).

Finally, students were given the flexibility to attend any section they wanted. Keeping with prior semesters, students still submitted their lab assignments on Gradescope to receive credit. The percentage of their grade that counted for lab remained the same.

6.2.3 Concerns About the New Structure

Teaching staff and instructors had two primary concerns with shifting to the discussion-like structure:

1. "This change makes the lab too easy for students and they won't learn as much. Students need to struggle to learn, and this structure is too hand-holdy. Students will not learn if they don't completely solve the problems on their own."
2. "A student could take advantage of the guided structure by copying the answers and not actively participating in the lab sections."

Addressing Concern 1: Students in CS61C are classified as novice learners in the field of computer architecture. Decades of empirical research on the effects of guided vs. unguided instruction on novice learners, "has provided overwhelming and unambiguous evidence that minimal guidance during instruction is significantly less effective and efficient than guidance specifically designed to support the cognitive processing necessary for learning." (Kirschner et al., 2006). The existing lab structure in which students learn the lab material by completing exercises independently or with a partner is known as problem-based learning (PBL), which is classified as a minimally-guided instructional approach. The proposed solution uses a guided instructional approach.

Addressing Concern 2: It is true that students could take advantage of this lab structure and copy the answers; however, this problem is not unique to a discussion-like lab structure. In the previous lab structure, students could easily copy solutions from their friends without a lab staff member knowing. In this new structure, the staff members can easily see if the student is not participating in the lab and can make attempts to actively engage them. Additionally, the discussion-like structure has the potential to reduce the instances of students absent-mindedly copying someone else's work when they are stressed about completing the lab before the

deadline. Instead, the structure of the lab allows students to focus on learning the material rather than worrying about being able to achieve a certain grade because students know that they will exit the lab section with their assignments completed.

Ultimately, it is important to remember that students who copy answers on lab assignments only hurt themselves because they are not getting the practice they need to complete the projects and exams. Instead of focusing attention on making it harder to "cheat", the focus should be placed on the students who are actually putting in the effort to learn. Focusing on the "cheaters" just makes the class harder for everyone and it makes teaching the class less enjoyable. While there is the potential for students to copy answers, this occurrence would likely be no more than in the previous structure and the benefits of the new structure heavily outweigh this risk.

7 Discussion-like Lab Structure Analysis

The discussion-like lab structure was successfully implemented in the Spring 2022 semester. This chapter provides an analysis of how the structure impacted student achievement of lab learning objectives, lab completion time, and the lab environment.

7.1 Lab Learning Objectives

To measure students' learning outcomes, 20 students were randomly selected to answer the checkoff questions that were asked in Fall 2021. All students chosen to complete the checkoff attended an in-person lab section. Their responses were coded using the same technique as the Fall 2021 study (described in *Section 5.1.1*). The results are shown in Figure 8. For easy comparison, Figure 2 from the Fall 2021 study has been reproduced below. The results show that there was a large improvement in student ability to answer checkoff questions. In the majority of the labs, more than 60% of students participating in the analysis met the learning objectives compared to 30% of students in the prior lab structure.

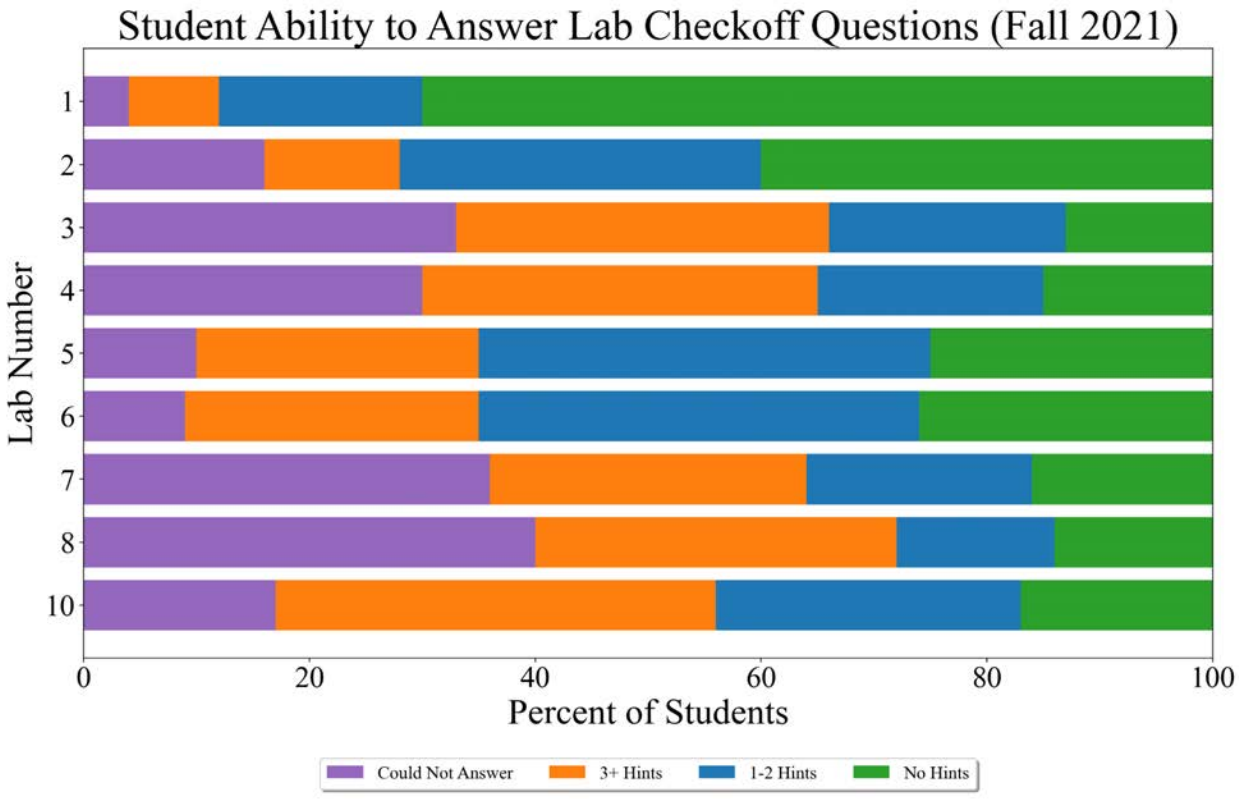

Figure 2. (Duplicate from Section 5.1.1.) Results from the analysis of student achievement in labs as measured by the number of hints needed to answer checkoff questions in Fall 2021.

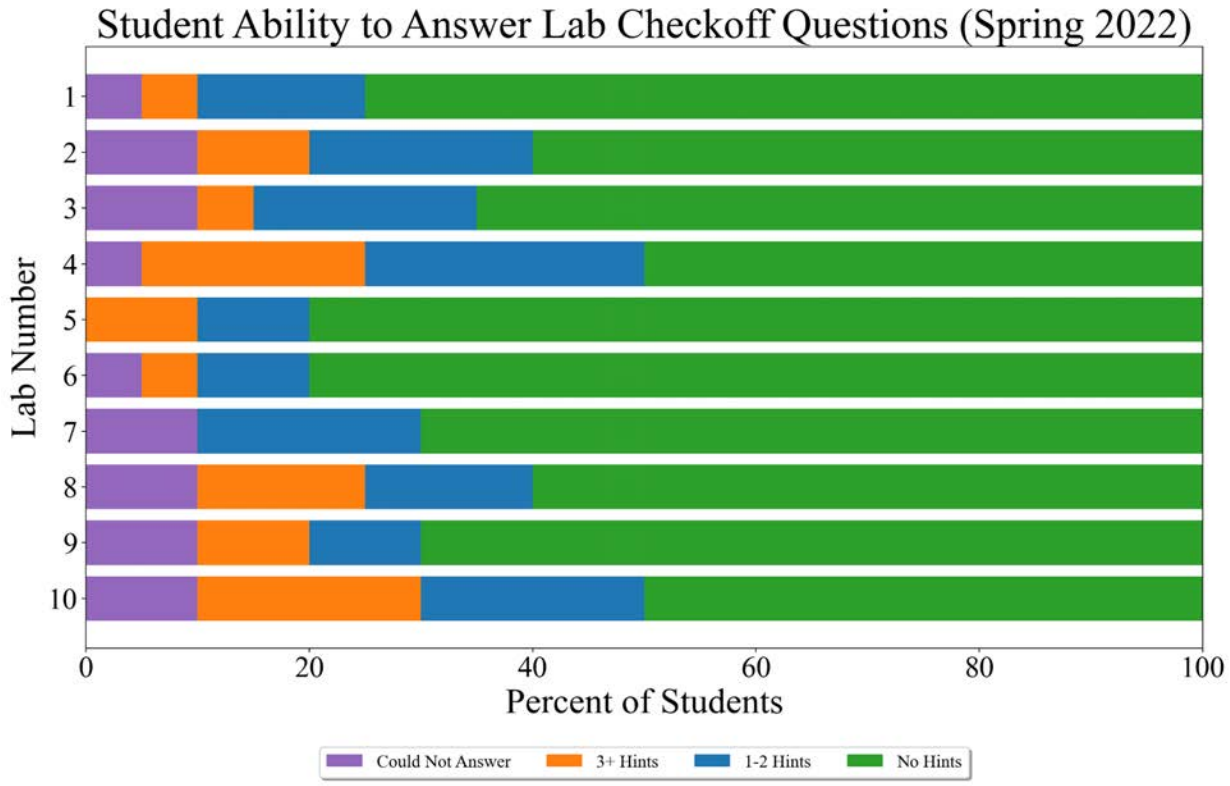

Figure 8. Results from the analysis of student achievement in labs as measured by the number of hints needed to answer checkoff questions in Spring 2022.

7.2 Lab Completion Time

Figure 9 contains the self-reported lab completion time in Spring 2022. The median time spent on eight out of ten labs fell within the two-hour mark compared to four out of ten labs in Fall 2021. For easy comparison, Figure 3 from the Fall 2021 study has been reproduced below.

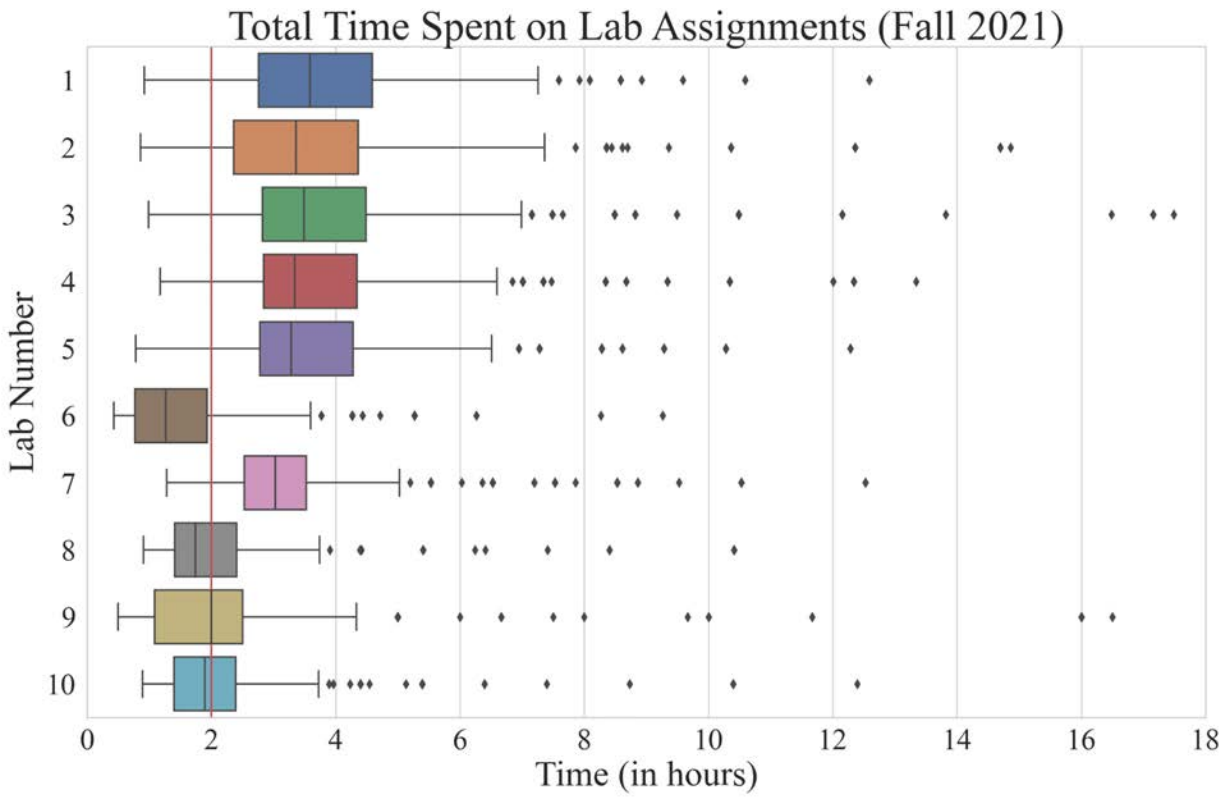

Figure 3. (Duplicate from Section 5.1.2) Total time students spent on each lab assignment in Fall 2021. The red line indicates the expected time to complete the lab.

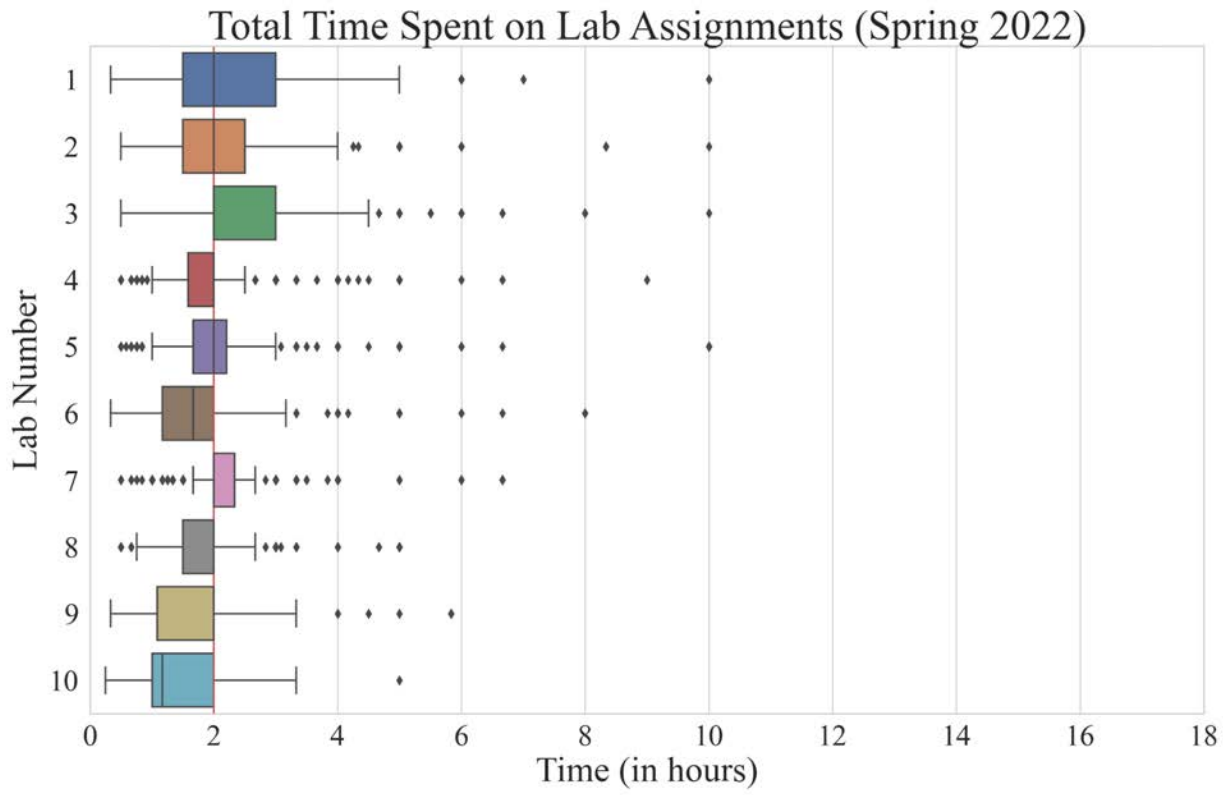

Figure 9. Total time students spent on each lab assignment in Spring 2022. The red line indicates the expected time to complete the lab.

7.3 Lab Environment

7.3.1 Students Completing Lab Assignment in Lab Section

The percentage of students who completed the lab assignments in lab section was much greater in Spring 2022 compared to Fall 2021 as seen in Figures 5 and 10. In Fall 2021 the percentage of students who worked on the assignments during lab section decreased as the semester progressed. In Spring 2022, the trend was the opposite. By the end of Fall 2021, less than 20% of students were working on the assignments in lab section compared to over 50% of students at the end of Spring 2022.

Students who did not attend lab sections were asked why they chose not to attend. The top three responses were 1) I prefer to complete the lab on my own, 2) I prefer to work at my own pace, and 3) The lab times did not fit in my schedule.

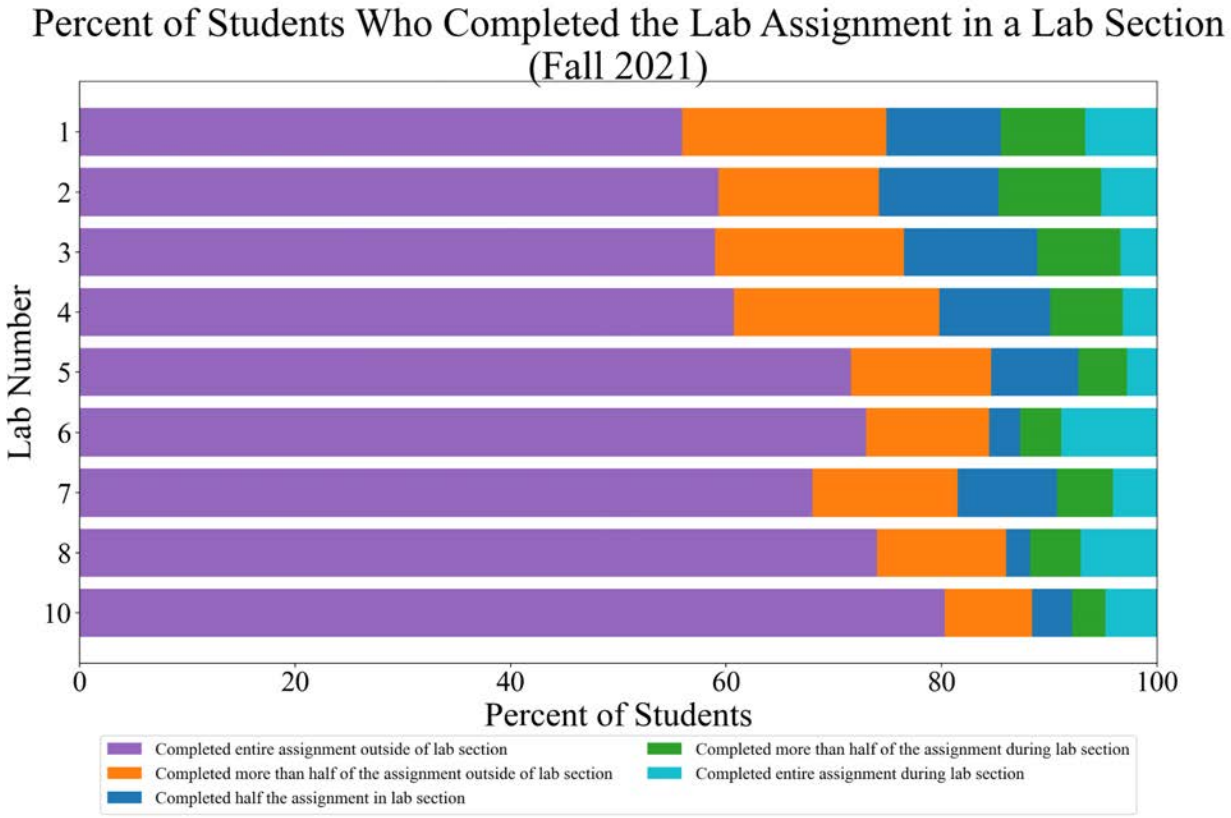

Figure 5. (Duplicate from Section 5.1.3) Percentage of students who completed the lab assignments during lab sections in Fall 2021.

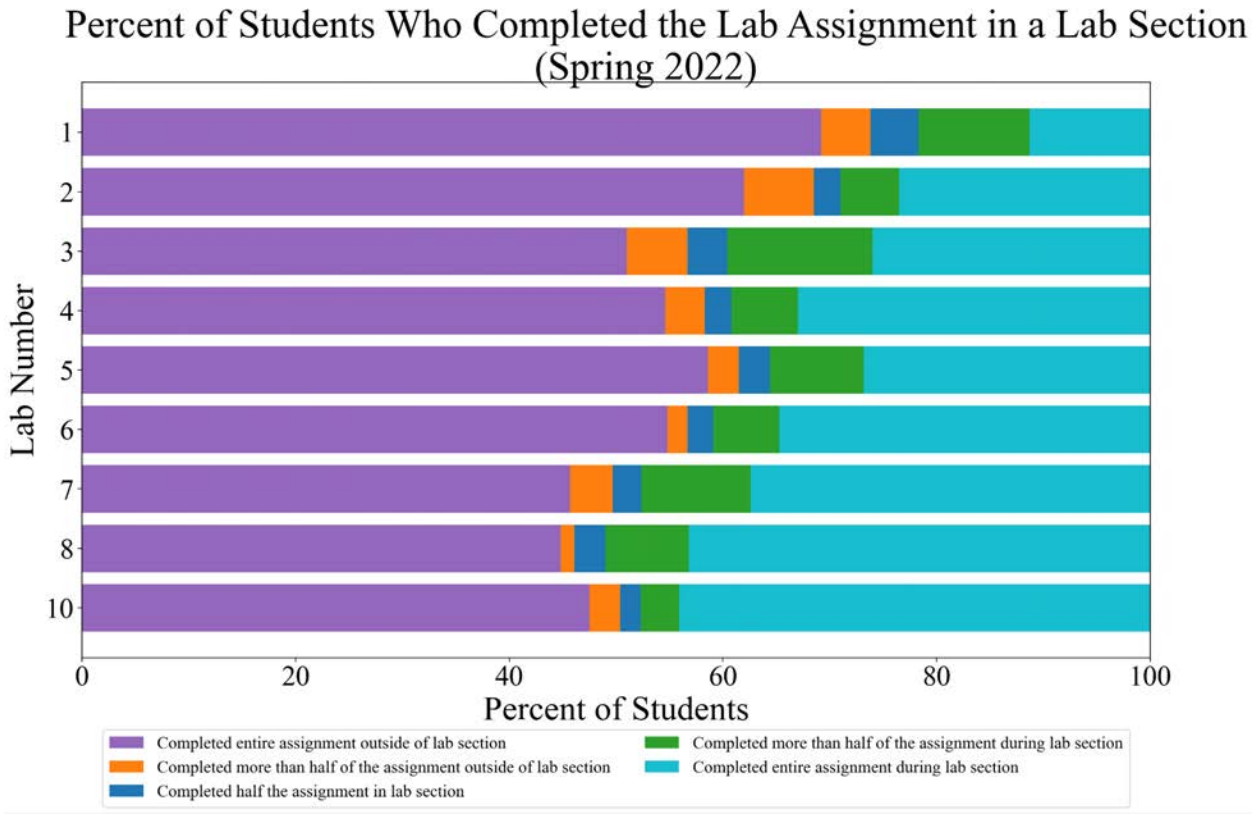

Figure 10. Percentage of students who completed the lab assignments during lab sections in Spring 2022.

7.3.2 Wait Time to Ask Questions

In Fall 2021, the office hour queue system was used to keep track of the order in which students submitted requests for questions or checkoffs. If the office hour queue system had not been used, course staff would not have been able to keep track of the order of requests because there were usually dozens of students requesting help at the same time. In Spring 2022, there were usually no more than three students requesting help at the same time. With each lab being staffed with one TA and two or three AIs, the use of the office hour queue system was deemed unnecessary. Instead, students would raise their hands to request help. In the online lab sections, the students would send a message over the online platform to receive help.

The TAs were surveyed on the average amount of time that they perceived students were waiting to receive help. Every TA leading an in-person section reported almost all students waited *less than one minute* to receive help. The TA leading the two online sections stated that almost all students waited *less than five minutes* to receive help. TAs were also asked if they felt pressure to spend less time than they preferred answering questions. All TAs reported that they did not feel pressure to rush through explanations. This experience is a complete reversal from Fall 2021 in which all TAs reported feeling rushed to answer student questions due to long queue times.

7.3.3 Student Participation

To understand student participation levels, lab TAs (both in-person and online) were asked to report on the percentage of students they observed interacting with their peers, the percentage of students who participated in the class-wide discussion on the exercises, and the percentage of students who were attentive during the class-wide discussions. “Attentiveness” was defined as looking at the board while not working on unrelated activities.

All six of the in-person sections reported that over 70% of the students in their lab section were interacting with other students during each lab. The two online lab sections reported that almost none of the students were interacting with each other. They suggested that students may have felt uncomfortable working with people they did not know over Zoom.

Of the six in-person sections, three reported that about one-third of the students were actively asking and answering questions during the class-wide discussions, one reported that half of the students were actively engaged, and two reported that 25% of the students were actively engaged. All in-person TAs reported that at least 90% of students were attentive during the class-wide discussion. The online lab TA reported that about 10% of students were active during the class-wide discussions. This TA was unable to determine how many students were attentive during the class-wide discussions because all students had their cameras off.

TAs were also surveyed about the propensity of students to copy solutions, a concern raised by instructional staff (see *Section 6.2.3*). Two in-person sections reported that they did not see this behavior at all, and four in-person sections reported that there were occasionally one to three students who did this in their section. The lab TA for the two online sections was unable to view this behavior because the students had their cameras turned off. Based on the lack of interaction the TA observed between students, they suspected that there were many students who were not actively attempting to complete the lab. Based on these observations, the presence of students absent-mindedly copying answers during in-person sections is so minimal that the benefits of the new structure heavily outweigh the risk of this occurrence. In the online sections, the unknown behavior of students warrants an additional investigation to determine if the suspicion of students absent-mindedly copying the answers is pervasive.

7.4 Staff Experience

The new lab structure gave more TAs and AIs the ability to work in their desired roles. At the beginning of each semester, the TAs are surveyed on which duties they would like to perform (e.g. lead a discussion section, develop projects, hold office hours, etc). The most highly requested role is leading a discussion section. Unfortunately, there are not enough discussion sections for every TA who wants to lead one because CS61C only offers one discussion section per 100 students. The new lab structure alleviates this problem by providing more slots for TAs to teach in front of students.

AIs commonly report that they want to have the opportunity to lead a portion of a class discussion. At the end of Spring 2022, the AIs were given the opportunity to lead a portion of the

lab discussion under the guidance of a TA. The AIs who participated reported that this was their favorite part of the semester because they had the opportunity to teach in front of a class.

The six TAs who taught in both Fall 2021 and Spring 2022 were surveyed on their personal experience with the new lab. All six TAs preferred the new lab structure over the old one citing that they felt less stressed and more connected with the students in their section(s).

7.5 Student Feedback

Overall student feedback from end-of-semester surveys indicated that the new lab structure was well received. Many students stated that the structure lessened their stress because they knew they would have the support they needed to finish. Selected comments representative of general feedback are listed below.

- *I really enjoy that we get enough time to struggle with trying to solve it on our own, but then ultimately come back together and are able to understand the solution with a nuanced explanation.*
- *I learn everything for this class in lab section. The lectures can be very intimidating because they are so fast, but the labs are much less scary and make the material make more sense. I really appreciate that the TA ensures that everyone understands the content before moving to the next section*
- *I like that lab can be finished within the allocated time! It's way less stressful because I can just focus on learning the material instead of stressing about not being able to figure out how to solve the problem and getting a bad grade*
- *I like the focused labs which try to maximize learning but minimize tedious/frustrating work.*
- *The live labs are so helpful with the projects! The lab walkthroughs were really great because it gave me time to attempt the problem and then verify my solution*

8 Future Work

Many students in CS61C struggle to pick up fundamental course concepts because the fast pace of the course does not provide them with enough time to properly absorb the material. Professors have noted that students' C programming ability has greatly decreased over the last couple of decades. Additionally, many students pass the course without comprehending calling convention, a fundamental concept in RISC-V programming. I believe that the decline in CS61C students' basic programming skills is due to the fact that new material keeps being added to the course without removing anything.

The large amount of material covered in CS61C has led to a course schedule where projects are often released on the same day as the labs that correspond to them. As labs are supposed to prepare students for their projects, students should not be expected to begin the project until they have submitted the corresponding lab(s). Given that students have one week to complete each lab assignment, this means that students cannot start the projects until at least one week after they have been released. However, due to stress about project due dates, many students end up starting the projects without first completing the corresponding lab, which leads them to experience cognitive overload.

This scheduling problem occurs because there are four projects, each of which takes two to four weeks to complete. To ensure students are given enough time to work on each project, they end up being released at the same time as the corresponding labs. This scheduling pattern doesn't actually give students the proper amount of time to complete their projects as they first need to complete the corresponding lab(s). The lab assignment cannot be released earlier as they cannot be released until after the corresponding lecture(s) have occurred.

My proposal for future work will eliminate this scheduling issue and improve student achievement by putting greater emphasis on helping students build their foundational knowledge and removing some emphasis on advanced topics. The concepts that students struggle with the most in the course are C programming and debugging, RISC-V programming and debugging, and Logisim debugging. To this end, I propose adding an additional lab for each of these topics. To make up for the increased workload, some assignments will need to be removed.

I propose removing the virtual memory lab and Project 4. I have chosen these assignments not because I have deemed them to be unimportant. There is just simply too much material in the course and I think that it is more important for students to have a good understanding of fundamentals than it is for them to be exposed to advanced material that they only somewhat understand due to gaps in their foundational knowledge.

The removal of Project 4 is extremely controversial as people believe that it is CS61C's responsibility to teach students how to work within large code bases. With the wide range of material that the course is required to cover, I believe downstream courses have more room to take this on. As data-level parallelism and thread-level parallelism will still be covered in the lab assignments, students will still leave CS61C with knowledge of parallel programming.

Suggested improvements for each existing lab as well as the three additional proposed labs are described in Appendix I. The existing lab assignments should be modified to include worked examples of each concept as this reduces cognitive load. The worked examples should be paired with similar practice problems. Discussion-like lab sections should continue to be offered based on their success in Spring 2022.

9 Conclusion

Based on the results of the Fall 2021 analysis, the existing lab model devolved from its initial closed lab design into an open lab setting. This shift happened because long wait times discouraged many students from attending the lab section. Research has demonstrated that closed labs have better student achievement outcomes, reduce student stress, and promote a stronger community among students than open labs (Thweatt, 1994; Wu et al., 1997; Newby, 2002). Additionally, a closed lab that is well-paced by a TA and has proper staffing to assist students when they are struggling can help students finish the lab in the allotted time. For these reasons, the lab sections were restructured into a closed lab with a discussion-like format.

In a discussion-like lab format, the lab time is segmented into collaborative, student-led work (with TAs available to assist students if needed) followed by a TA-guided review of the problems. It is possible that some students will not have finished the exercise before the TA begins reviewing it. Such students are likely experiencing cognitive overload because they have not acquired the schemas needed to solve the given problem. Cognitive load theory asserts that these students would benefit more from viewing a worked example (i.e., the TA's revision of the problem) as opposed to continuing to struggle to solve the problem on their own as was the case with the previous lab structure (Sweller, 1985).

In addition to helping students who are cognitively overburdened, TA-led review can also help students who were able to complete the problem in the allotted amount of time. Such students benefit from exposure to an expert's thought process and problem-solving technique by, for example, potentially identifying areas of their code that can be improved and learning better programming practices and strategies for solving future problems (Bower, 2008).

An analysis of the new lab structure's implementation demonstrated improved student learning outcomes and decreased median lab completion times. In the majority of the labs, more than 60% of students participating in the analysis met the learning objectives compared to 30% of students in the prior lab structure. Students' self-reported lab completion times indicated that in eight out of ten labs, the majority had completed the lab assignment within the allotted time as compared to only four out of ten labs in the previous semester.

The new structure also improved the lab environment. By the end of Spring 2022, more than 50% of students worked on the assignment during lab section as compared to less than 20% of students at the end of Fall 2021, indicating that students found the discussion-like lab structure more beneficial than the open lab structure. Almost all students attending in-person lab sections waited less than one minute to receive help, with students attending online sections waiting less than five minutes to receive help. This is a major improvement over Fall 2021 where the median wait time to receive help was ten minutes with many students waiting more than 30 minutes. The improvement in wait times reduced pressure on TAs and allowed them to spend as much time answering individual questions as needed. Additionally, students were less stressed because they no longer worried about competing for staff resources and could instead focus on learning the content presented in labs.

Overall, the new lab structure was effective in addressing its inciting problems. More work still needs to be done to continue improving learning outcomes and completion time. In my proposal for future work, I suggest that these areas could be improved by rewriting the lab assignments to incorporate more worked examples to reduce cognitive load and help students acquire problem-solving schemas. Additionally, my proposal for future improvements of lab assignments gives students more time to build foundational knowledge before introducing students to advanced topics. Appendix I discusses the proposed changes.

Appendix I

Recommended Future Lab Exercises

Lab 0: Intro and Setup

Lab 0 did not have a section that students could attend in Spring 2022. In future semesters, there should be a closed lab offered for Lab 0. The topic that students struggled with the most in this lab was git and using the command line. In the future, lab 0 should provide a full review of these two topics. Here is a suggested flow:

1. Accessing Services (same as current version)
2. Installing Software (same as current version)
3. Instructional Accounts and Servers (same as current version)
4. GitHub Setup (same as current version)
5. How to use the command line
    a. Provide students with a guide of the most commonly used commands
6. Install VSCode (and the ssh extension)
    a. Provide students with a guide of step-by-step instructions detailing how to connect to the hive using VSCode's ssh extension
    b. Explanation: One of the most common complaints from CS61C students is that they struggle to use vim on the hive machines. The VSCode ssh extension is a wonderful tool that enables students to program on the hive machines in a familiar environment that is easy to navigate. In prior semesters, there was strong pushback from a couple of TAs about promoting this tool because they had encountered a few students for whom the tool was not working. However, when all of these students were forwarded to me, I easily resolved their issues by either restarting VSCode or reinstalling the extension. Therefore, there is no reason to not support this tool.
7. Git tutorial
    a. Provide students with a guide of the most commonly used commands
    b. Have students clone the *labs* repo on their local machine and on the hive machine.
    c. Provide a guided example that students follow along with where they make a change on their local repo, commit and push this change to origin, and then pull this change to the hive machine
    d. Instruct students to make a change on the hive machine, commit and push the change to origin, and then pull the change to their local machine
    e. Provide a guided example of creating a new branch on the hive machine, pushing this branch to origin, then pulling this branch onto their local machine

f. Instruct students to create a new branch on their local machine, push this branch to origin, then pull this branch onto the hive machine
g. Provide students with a worked example on resolving merge conflicts
h. Instruct students to resolve a merge conflict (the conflict should be provided to them)

Lab 1: Intro to C

The existing C labs do not provide students with enough practice programming in C. The suggested flow of Lab 1 below provides students with better exposure to the basic C programming concepts that are new to them: strings, structs, and pointers. It also covers printing, fixing compiler warnings, and header files as students tend to struggle with these. The lab should isolate each of these concepts (as opposed to introducing them together) to avoid cognitive overload.

1. How to compile and run a program (same as current)
2. Header files
    a. Explain what header files are and how to use them
    b. Worked example: Define a struct in a struct in a header file and demonstrate how to use that struct in a C file
    c. Worked example: Show a program where someone forgot to include a header file. Demonstrate the error that occurs when compiling the code and walk through how to understand and resolve the reported error.
    d. Worked example: Show a program where someone used the wrong path for including the header file. Demonstrate the error that occurs when compiling the code and walk through how to understand and resolve the reported error.
    e. Practice Problem: Give students a program where someone used the wrong path for including the header file and ask them to resolve the error. (Don't tell students what the error is - have them figure this out on their own)
3. Compiler warnings
    a. Worked examples: Show example with spelling issues, incompatible types, too many/too few arguments
    b. Practice problems: Ask students to fix compiler warnings that have been caused by spelling issues, incompatible types, too many/too few arguments
4. Printing
    a. Worked example: Show how to print different types of variables
    b. Practice problem: Ask students to print different type of variables
5. Strings
    a. Worked example: Different ways to create strings and how to print them
    b. Practice problem: Different ways to create strings and how to print them
    c. Worked example: C string functions

        i. strlen
        ii. strcat
        iii. strcpy
        iv. strcmp
    d. Practice Problem: C string functions
6. Structs
    a. Worked example
    b. Practice problem
7. Pointers
    a. Worked example
    b. Practice problem

Lab 2: GDB and Memory Management

GDB should be moved to Lab 2 as there would not be time for it with the new Lab 1 exercises. Additionally, introducing basic C programming and GDB in the same lab is too much information at once for students. The GDB exercises from Spring 2022 significantly improved students' ability to use GDB compared to prior semesters, so those activities should be kept. Lab 2 should also introduce students to memory management and how to find segfaults.

1. Intro to GDB: start, step, next, finish, print, quit (same as the one in the SP22 Lab 1)
2. Intro to GDB: break, conditional break, run, continue (same as the one in the SP22 Lab 1)
3. Memory management (malloc and free)
    a. Worked example
    b. Practice problem
4. GDB exercise on finding segfaults (same as the one in the SP22 Lab 1)

Lab 3: Valgrind and Memory Management

Valgrind should be moved to Lab 3 as there is no time for it to be covered thoroughly in Lab 2. Memory management should be covered in Lab 2 and Lab 3 as it is a topic that students have great difficulty with. The new Valgrind example only covered a few of the possible messages that Valgrind may output, so additional Valgrind exercises should be added to cover a wider range of possible outputs.

1. Memory management worked example
2. Memory Management practice problem
3. Provide a valgrind overview (same as the one in SP22 Lab 2)
4. Valgrind worked example (same as the one in SP22 Lab 2)

5. Valgrind practice problem applying same concepts as the prior worked example
6. Valgrind worked example demonstrating how to interpret other messages
7. Valgrind practice problem applying same concepts as the prior worked example

Lab 4: Intro to RISC-V Assembly and Venus

The proposed Lab 4 is similar to the Spring 2022 Lab 3, but it covers less information. The Spring 2022 version of the lab was too advanced for students just learning how to program in RISC-V. This first RISC-V lab only had students read and write basic programs and does not mention calling convention.

1. Intro to Venus (same as the one in SP22 Lab 3)
    a. keep in mind that the set up takes longer than you think
2. Review .data, .word, and .text sections
3. Worked example of basic RISC-V program
    a. Demonstrate how to step through the program in Venus and how to read the content of the registers
4. Practice Problem: Ask students how they would modify the code in the worked example to do perform some other operation
5. Worked example of basic RISC-V program
6. Practice problem: Ask students to write their own basic program

Lab 5: RISC-V: Pointers and Calling Convention

Students have great difficulty understanding how to use pointers in RISC-V. This may be caused by a lack of foundational knowledge on how to use pointers in C. Hopefully, the students' understanding of pointers will be better with the additional C lab. Nevertheless, the beginning of Lab 5 should have students focus on how to use pointers in RISC-V. Calling convention is another topic students have great difficulty understanding, so it should be taught across two labs. This gives students multiple opportunities to learn calling convention with breaks in between which should help to better solidify the concept.

1. Worked example on pointers in RISC-V
    a. Demonstrate how to step through the program in Venus and how to read the content of the registers and memory
2. Practice Problem: Ask students how they would modify the code in the worked example to do perform some other operation
3. Explain calling convention
4. Worked example: simple calling convention

5. Practice problem: ask questions about calling convention
6. Worked example: calling convention

Lab 6: RISC-V: Calling Convention and Debugging

As stated above, calling convention should be included in two labs as students have extreme difficulty understanding it. Students should also get practice debugging in RISC-V to help prepare them for Project 2.

1. Worked example on calling convention
2. Practice problem on calling convention
3. Practice problem on calling convention
4. Worked example: debugging
    a. Step through a program that contains a bug. Demonstrate to students the thought process they should have when debugging
5. Practice problem: debugging

Lab 7 : Logisim

This lab is the same as the Spring 2022 version except that the rotate right exercise is removed and a multiplexor exercise is added. The rotate right exercise from Spring 2022 is not included as it caused tremendous cognitive overload. The exercise asked students to combine multiple Logisim features which they have just learned (muxes, splitters, sub-circuits) with a hardware algorithm that is too advanced for students at this stage. Most students in Fall 2021 and Spring 2022 had extreme difficulty with this exercise and had no idea where to start.

Most students completing the rotate right exercise in Fall 2021 and Spring 2022 did not know how to use multiplexores as they were covered in lecture for less than ten minutes and there was no other lab exercise using them. To address this problem, a worked example and practice problem on multiplexores should be added.

1. Intro (same as the intro in SP22)
2. Sub-circuits (same as the one in SP22)
3. Storing state (same as the one in SP22)
4. Advanced Logisim features (same as the one in SP22)
5. Multiplexores
    a. Worked example
    b. Practice Problem

Lab 8: CPU and Pipelining

This lab corresponds to the Spring 2022 Lab 6. In Spring 2022, two exercises were added to Lab 6 to give students more practice building small circuits. The first exercise has students construct the s-type immediate of a 32-bit instruction. The second exercise has students construct the BrUn control signal from a 32-bit instruction. Both of these exercises were well received by students and helped prepare them for Project 3, so they should be kept.

The existing Lab 6 also attempted to teach students how to pipeline circuits. Students were confused by the exercise because the output of the circuit was also an input to the circuit. A majority of students reported that they completed the exercise by following the directions, but they had no idea what was going on. To alleviate this problem, a new exercise is suggested below.

1. Constructing Immediates (same as SP22)
2. Constructing the BrUn Control Signal (same as SP22)
3. Practice problem: ask students to create a circuit that performs the following operation in one clock cycle [(a+b) x (c+d)] + [(a-b) x (c-d)]
4. Worked example: Show students how to break the design up into two stages
5. Practice problem: Ask the students to break the design up into three stages
6. Worked example: Show students how to calculate the critical path of the two stage design
7. Practice problem: Ask students to calculate the critical path of the single-cycle and three stage design

Lab 9: Debugging in Logisim

Every semester, students struggle tremendously with not knowing how to debug Project 3. Various attempts have been made to add guidance to the spec, but students largely continue to struggle with this skill. Debugging in Logisim should be taught to students before they begin Project 3. As the other two Logism labs do not have room for more information, a lab should be developed that is completely devoted to debugging in Logisim.

1. Create an environment similar to Project 3 (similar harnesses, testing infrastructure) with a circuit containing many bugs
2. Worked example: Show students how to step through the design and trace signals
3. Practice problem: Ask students step through the design and trace signals
    a. As an example question, you can ask them what the value of a particular signal is on cycle 5
4. Worked example: Show students how to interpret the test results (Bug #1)
5. Practice problem: Ask students to interpret test results (Bug #2)
    a. As an example question, you can ask students which cycle the error occurred in or which signal is incorrect

6. Worked example: Show students how to find the cause of Bug #1 and how to fix it
7. Practice problem: Ask students to find the cause of Bug #2 and fix it
8. Worked example: Show students how to interpret the test results (Bug #3) and how to find and resolve the bug
    a. Make this bug more advanced than Bug #1
9. Practice problem: Ask students to interpret test results (Bug #4) and how to find and resolve the bug

Lab 10: Caches

The Spring 2022 cache lab had three exercises: cache simulator, loop ordering, and cache blocking. The cache simulator is extremely valuable to students as they get to interact with the program which produces visuals of the cache accesses. In Spring 2022, every TA reported that they wished students had more time on this exercise even though they spent an entire hour on it. TAs reported having to rush through some explanations and felt that they lost some students along the way.

The simulation exercise helps build the foundation of the students' understanding of caches whereas the loop ordering and cache blocking exercises are advanced applications of cache concepts. As understanding the basic principles of caches is more important than understanding advanced applications, I propose that this lab only focus on the cache simulator. The existing exercise will take two hours if the students are not rushed through it.

1. Cache simulation exercise
    a. This should be the same as the Spring 2022 version, but a worked example should be added to the beginning of the lab

Lab 11: SIMD Instructions and Loop Unrolling

This lab covers the same topics as Lab 8 in Spring 2022. Worked examples should be added to this lab to avoid cognitive overload.

1. Loop unrolling
    a. Worked example
    b. Practice problem
2. Provide an overview of the SIMD instruction format
3. SIMD Instructions
    a. Two worked examples
    b. Two practice problems
4. Loop unrolling combined with SIMD

    a. Worked example
    b. Practice problem

Lab 12: Thread-level Parallelism

This lab covers the same topics as Lab 8 in Spring 2022. Worked examples should be added to this lab to avoid cognitive overload. Additionally, to partially make up for removing Project 4, a worked example and practice problem can be added to this lab where students get to combine OpenMP and SIMD instructions.

1. Intro to OpenMP (can be the same as SP22)
2. OpenMP (no critical section)
    a. Worked example
    b. Practice problem
3. OpenMP (critical section)
    a. Worked example
    b. Practice problem
4. Combine OpenMP and SIMD
    a. Worked Example
    b. Practice Problem

Bibliography

Beichner, R. J., Saul, J. M., Abbott, D. S., Morse, J. J., Deardorff, D., Allain, R. J., Bonham, S. W., Dancy, M. H., and Risley, J. S. The student-centered activities for large enrollment undergraduate programs (scale-up) project. *Research-based reform of university physics 1*, 1 (2007), 2--39.

Börstler, J., Nordström, M., and Paterson, J. H. On the quality of examples in introductory java textbooks. *ACM Transactions on Computing Education (TOCE) 11*, 1 (2011), 1--21.

Bower, M. Teaching and learning computing.

Bruffee, K. A. Sharing our toys: Cooperative learning versus collaborative learning. *Change: The Magazine of Higher Learning 27*, 1 (1995), 12--18.

Burch, C. Logisim: A graphical system for logic circuit design and simulation. *J. Educ. Resour. Comput. 2*, 1 (mar 2002), 5--16.

Crouch, C. H., and Mazur, E. Peer instruction: Ten years of experience and results. *American journal of physics 69*, 9 (2001), 970--977.

Denning, P. J., Comer, D. E., Gries, D., Mulder, M. C., Tucker, A., Turner, A. J., and Young, P. R. Computing as a discipline. *Computer 22*, 2 (1989), 63--70.

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., and Wenderoth, M. P. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the national academy of sciences 111*, 23 (2014), 8410--8415.

Garner, S. *Reducing the cognitive load on novice programmers*. Association for the Advancement of Computing in Education (AACE), 2002.

Johnson, D. W., and Johnson, R. T. *Learning together and alone: Cooperative, competitive, and individualistic learning*. Prentice-Hall, Inc, 1987.

Johnson, D. W., and Johnson, R. T. *Cooperation and competition: Theory and research.* Interaction Book Company, 1989.

Johnson, D. W., Johnson, R. T., and Smith, K. The state of cooperative learning in postsecondary and professional settings. *Educational Psychology Review 19* (2007), 15--29.

Kirschner, P. A., Sweller, J., and Clark, R. E. Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational psychologist 41*, 2 (2006), 75--86.

Knox, D., Wolz, U., Joyce, D., Koffman, E., Krone, J., Laribi, A., Myers, J. P., Proulx, V. K., and Reek, K. A. Use of laboratories in computer science education: Guidelines for good practice: Report of the working group on computing laboratories. In *Proceedings of the 1st conference on Integrating technology into computer science education* (1996), pp. 167--181.

Knox, D. L. On-line publication of cs laboratories. In *Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education* (1997), pp. 340--344.

Lin, J. M.-C., Wu, C.-C., and Liu, H.-J. Using simcpu in cooperative learning laboratories. *Journal of Educational Computing Research 20*, 3 (1999), 259--277.

Lou, Y., Abrami, P. C., and d'Apollonia, S. Small group and individual learning with technology: A meta-analysis. *Review of educational research 71*, 3 (2001), 449--521.

Lou, Y., Abrami, P. C., Spence, J. C., Poulsen, C., Chambers, B., and d'Apollonia, S. Within-class grouping: A meta-analysis. *Review of Educational Research 66*, 4 (1996), 423--458.

Marcoulides, G. A. The relationship between computer anxiety and computer achievement. *Journal of Educational Computing Research 4*, 2 (1988), 151--158.

McCauley, R., and Manaris, B. Computer science education at the start of the 21st century-a survey of accredited programs. In *32nd Annual Frontiers in Education* (2002), vol. 2, IEEE, pp. 10--15.

Mihail, R. P., and Roy, K. Closed labs in programming courses: A review. In *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)* (2016), The Steering Committee of The World Congress in Computer Science, Computer Engineering, and Applied Computing, p. 104.

Muldner, K., Jennings, J., and Chiarelli, V. A review of worked examples in programming activities. *ACM Trans. Comput. Educ. 23*, 1 (2022).

Newby, M. An empirical study comparing the learning environments of open and closed computer laboratories. *Journal of Information Systems Education 13* (01 2002).

Newby, M., and Fisher, D. An instrument for assessing the learning environment of a computer laboratory. *Journal of Educational Computing Research 16*, 2 (1997), 179--190.

Prey, J. C. Cooperative learning and closed laboratories in an undergraduate computer science curriculum. *ACM SIGCSE Bulletin 28*, SI (1996), 23--24.

Prince, M. Does active learning work? a review of the research. *Journal of engineering education 93*, 3 (2004), 223--231.

Sabin, R. E., and Sabin, E. P. Collaborative learning in an introductory computer science course. In *Proceedings of the twenty-fifth SIGCSE symposium on Computer science education* (1994), pp. 304--308.

Smith, B. L., and MacGregor, J. T. Collaborative learning: A sourcebook for higher education, 1992.

Soh, L.-K., Samal, A., Person, S., Nugent, G., and Lang, J. Closed laboratories with embedded instructional research design for cs1. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (New York, NY, USA, 2005), SIGCSE '05, Association for Computing Machinery, pp. 297--301.

Sweller, J. Cognitive load during problem solving: Effects on learning. *Cognitive science 12*, 2 (1988), 257--285.

Sweller, J., and Cooper, G. A. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and instruction 2*, 1 (1985), 59--89.

Thweatt, M. Csi closed lab vs. open lab experiment. In *Proceedings of the Twenty-Fifth SIGCSE Symposium on Computer Science Education* (New York, NY, USA, 1994), SIGCSE '94, Association for Computing Machinery, pp. 80--82.

University of California, Berkeley CS61C Course Staff. Venus RISC-V simulator. https://venus.cs61c.org/

Ward, M., and Sweller, J. Structuring effective worked examples. *Cognition and instruction 7*, 1 (1990), 1--39.

Webb, N. M. Assessing students in small collaborative groups. *Theory Into Practice 36*, 4 (1997), 205--213.

Webb, N. M., and Palincsar, A. S. *Group processes in the classroom.* Prentice Hall International, 1996.

Williams, L., Wiebe, E., Yang, K., Ferzli, M., and Miller, C. In support of pair programming in the introductory computer science course. *Computer Science Education 12*, 3 (2002), 197--212.

Wu, C.-C., Lin, J. M.-c., and Hsu, I. Y.-w. Closed laboratories using simlist and simrecur. *Computers & Education 28*, 1 (1997), 55--64.

Yerion, K. A., and Rinehart, J. A. Guidelines for collaborative learning in computer science. *ACM SIGCSE Bulletin 27*, 4 (1995), 29--34.