EAI Endorsed Transactions
on e-Learning

Research Article **EAI.EU**

A Project-Based Learning Approach to Teaching Computer Architecture

D. Cenk Erdil

School of Computer Science & Engineering, Sacred Heart University, Fairfield, CT 06825, USA

Abstract

This article describes details of design and implementation of an upper-level (core-Tier2) computer organization and architecture course with an online hands-on component as a blended learning environment. The revised course content is based on contemporary pedagogical approaches: blended learning, flipped classroom, lead learner, and project-based learning. While revising, ACM/IEEE-CS computing curricula guidelines were aligned with new modules being implemented. Online hands-on component of the course have been focused on using a single-board computer, and associated hardware to provide students relevant skills in designing projects that include hardware and software components. Research data collected after first two cohorts of the revised course show that implementation of the course was received well by the students, which improved the overall course reviews, while maintaining relevant levels of curricular material that includes core knowledge areas.

Received on 02 February 2019; accepted on 27 November 2019; published on 17 December 2019

Keywords: blended learning, computer science education, computer organization and architecture, single-board computers, project-based learning.

doi:10.4108/eai.13-7-2018.162289

1. Introduction

Following the active learning approach in [2, 3, 17], this article describes design of revising an upper-level (i.e., core-Tier2) computer organization and architecture course with an online hands-on component as a blended learning environment [8, 9, 13]. During revising the course, we realigned core modules with the Architecture knowledge area specified in ACM/IEEE joint task force computing curricula [1], and implemented core bodies of knowledge specified, with respect to computer organization and architecture. Most curricula in computer architecture include a two or three-course sequence, and the two common courses are computer organization [1], and computer architecture [2], typically offered in computer science and engineering undergraduate programs.

Collectively, these courses aim to equip students with hardware-related skills, recognition of the hardware-software interface, and a perspective on lower-level problems and common solutions on computers, storage, and related hardware, mainly focusing on implications of von Neumann Architecture into higher level software. In particular, the following knowledge areas constitute computer architecture curriculum [3]:

- Digital logic and digital systems
- Machine level representation of data
- Assembly level machine organization
- Memory system organization and architecture
- Interfacing and communication
- Functional organization (optional, or elective)
- Multiprocessing and alternative architectures (optional, or elective)

*Email: erdild@sacredheart.edu

[1]Common prerequisite to the computer architecture course.

[2]Depending on science or engineering focus, the first course may optionally be Digital Design, making organization and architecture courses second, and third in the sequence, respectively.

[3]For more details of each area, including topics and learning outcomes, please refer to [1].

- Performance enhancements (optional, or elective)

Computer organization [4, 7], and computer architecture [10, 12, 14] courses are ideally offered as two separate courses with a pre-requisite relationship between, and typically in Sophomore, and Junior years, respectively [4]. Due to credit requirements, some computer science departments opt in offering a combined organization and architecture course.

Based on these considerations in our design, the rest of this article provides details of our implementation for a combined computer organization and architecture course, as the sole *hardware* focused course in computer science department of a liberal arts college. The course is offered as a required course, as part of the four-year undergraduate degree requirements, in computer science, software development, information technology, game design and development programs [5], as well as an elective course for a masters degree in computer science. After course redesign, it has been offered first in Fall 2016, and has been continuously offered since then. Data presented in this article were collected during the inaugural year of offering the course for two cohorts: in Fall 2016, and Spring 2017 semesters.

Modules of the hands-on component are based on three particular contemporary curricular guidelines: blended learning, flipped classroom, lead learner, and project-based learning. Using these main approaches, the laboratory component of computer architecture course has been designed to complement the theoretical part and also provide students hands-on skills in working with single-board computers [11, 16], as well as a set of sensors, components, and associated programming interfaces that interact together with the computers.

Another important aspect of this course is to introduce a project component that bodes well with the theoretical part of the course, and hands-on modules, and allows students to design and implement an idea based on what they have learned in class. Students also design and implement a small project with a mandatory hardware component using a single-board computer. As most of the students take this course before their senior year, students reacted well to the idea of designing a small-scale project that would serve as training for their senior capstone projects.

During hands-on modules that started in the first week of the semester, students begin to use Arduino-based single-board computers, and learn how to use small electronic circuits as well as a breadboard and other IC-modules that second-year engineering students typically experiment in a digital design laboratory. This has been a positive learning experience for students as the computer science curricula offered in the department is mostly classical, and did not focus on computer architecture related knowledge areas to a great extent, whereas a large portion of the theoretical part in the course assumed some level of experience. Students then pick a hardware-based project, and complete their projects as an online module, while also completing hands-on exercises as learning activities. Completing basic hands-on exercises as the laboratory component at their own pace provided students the confidence to implement the projects that they have picked.

2. Course within the Curriculum

In the recommended course sequence, combined organization and architecture course is offered in the Fall semester of Junior year (i.e., fifth semester). The course is offered as a required course, as part of the four-year undergraduate degree requirements, in computer science, software development, information technology, game design and development programs [6], as well as an elective course for a masters degree in computer science. Prior to this course, students in the computer science program take the following courses as part of their major:

- Introduction to Programming (freshman, CS0/CS1)
- Software Development (freshman, CS1/CS2)
- Software Systems and Analysis (freshman)
- Discrete Mathematics (freshman)
- Software Development II (sophomore, CS2)
- Data Communications and Networks (sophomore)
- Database Management (sophomore)
- Internetworking (sophomore)
- System Design (sophomore)
- Ethics (junior, at the same semester as Computer Organization and Architecture course)

[4]Although some variations offer these courses in Junior and Senior years; or organization as part of the core, but architecture as elective.

[5]The course is available to students studying information technology and game design and development; however most of them take a *lighter* computer organization course that does not involve hands-on projects.

[6]The course is available to students studying information technology and game design and development; however most of them take a *lighter* computer organization course that does not involve hands-on projects.

Table 1. Demographics of the Fall and Spring cohorts: age distribution, and class standing.

Age Distribution	Fall Cohort	Spring Cohort
Under 18	–	–
18–20	57.1%	37.5%
21–23	39.3%	58.3%
24 or older	3.6%	4.2%
Total	100%	100%

Class Standing	Fall Cohort	Spring Cohort
Sophomore	–	4.2%
Junior	71.4%	58.3%
Senior	25.0%	33.3%
Graduate	3.6%	4.2%
Total	100%	100%

Table 2. Programs of Study by Cohort Size (N). (*) One student in Fall 2016 cohort is studying two programs: Computer Science and Software Development.

Program	Fall 2016	Spring 2017
Computer Science	28	20
Software Development	1	2
Game Design and Development	-	1
Information Technology	-	1
Total	N = 28(*)	N = 24

Table 3. Programs of Study by Year and Number of Students

Program	Sophomores	Juniors	Seniors	Graduates
	Fall 2016 Cohort (N = 28)			
Computer Science	-	20	7	1
Software Development	-	1	-	-
Game Design and Development	-	-	-	-
Information Technology	-	-	-	-
Total	0	20(*)	7	1

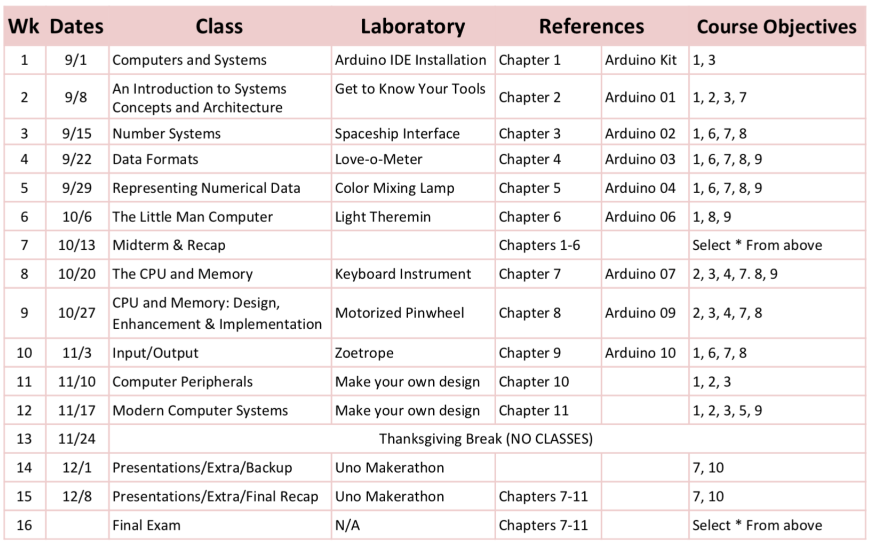
Program	Sophomores	Juniors	Seniors	Graduates
	Spring 2017 Cohort (N = 24)			
Computer Science	1	14	2	1
Software Development	-	2	1	-
Game Design and Development	-	1	1	-
Information Technology	-	1	-	-
Total	1	18	4	1

After this course, students take seven more Computer Science electives, up to 11 credits as a choice of internship or electives, as well as a two-semester long Senior Capstone course.

3. Course Design

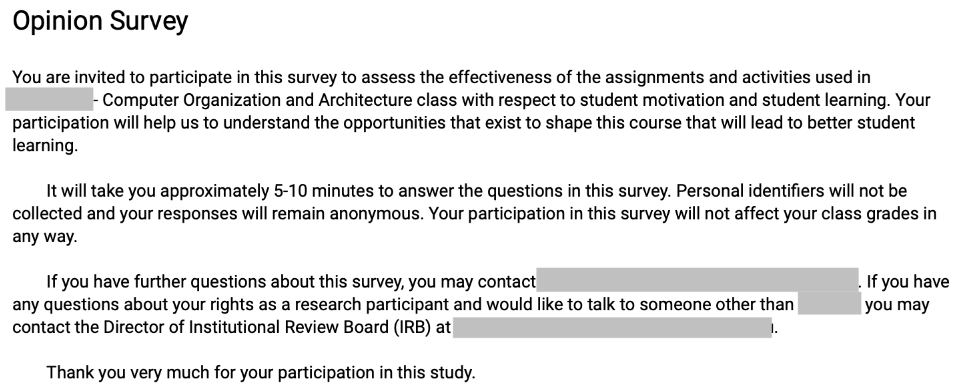
The course was originally designed to introduce computer organization and architecture related concepts, mainly focusing on assembly level machine organization knowledge area, and focused on a particular computer architecture and its assembler language as the hands-on part. Over the years it has been offered by various extended (i.e., adjunct) faculty, and has not been the favorite course neither for faculty nor for students. Within the department-level goals that are tied to self-assessment, the course has the following objectives:

- Understand the basics of computer hardware and how software interacts with computer hardware.
- Know the organization of the central processing unit (CPU) and memory hierarchy.
- Use critical thinking to make informed decisions in the selection of hardware.
- Demonstrate how memory caches and virtual memory work.
- Learn and demonstrate how program performance is affected by processor cache sizes.
- Understand how the architecture affects program performance.



Wk	Dates	Class	Laboratory	References		Course Objectives
1	9/1	Computers and Systems	Arduino IDE Installation	Chapter 1	Arduino Kit	1, 3
2	9/8	An Introduction to Systems Concepts and Architecture	Get to Know Your Tools	Chapter 2	Arduino 01	1, 2, 3, 7
3	9/15	Number Systems	Spaceship Interface	Chapter 3	Arduino 02	1, 6, 7, 8
4	9/22	Data Formats	Love-o-Meter	Chapter 4	Arduino 03	1, 6, 7, 8, 9
5	9/29	Representing Numerical Data	Color Mixing Lamp	Chapter 5	Arduino 04	1, 6, 7, 8, 9
6	10/6	The Little Man Computer	Light Theremin	Chapter 6	Arduino 06	1, 8, 9
7	10/13	Midterm & Recap		Chapters 1-6		Select * From above
8	10/20	The CPU and Memory	Keyboard Instrument	Chapter 7	Arduino 07	2, 3, 4, 7. 8, 9
9	10/27	CPU and Memory: Design, Enhancement & Implementation	Motorized Pinwheel	Chapter 8	Arduino 09	2, 3, 4, 7, 8
10	11/3	Input/Output	Zoetrope	Chapter 9	Arduino 10	1, 6, 7, 8
11	11/10	Computer Peripherals	Make your own design	Chapter 10		1, 2, 3
12	11/17	Modern Computer Systems	Make your own design	Chapter 11		1, 2, 3, 5, 9
13	11/24	Thanksgiving Break (NO CLASSES)				
14	12/1	Presentations/Extra/Backup	Uno Makerathon			7, 10
15	12/8	Presentations/Extra/Final Recap	Uno Makerathon	Chapters 7-11		7, 10
16		Final Exam	N/A	Chapters 7-11		Select * From above

Figure 1. Weekly meeting schedule of the theoretical and hands-on parts of the course.



Opinion Survey

You are invited to participate in this survey to assess the effectiveness of the assignments and activities used in [redacted]- Computer Organization and Architecture class with respect to student motivation and student learning. Your participation will help us to understand the opportunities that exist to shape this course that will lead to better student learning.

It will take you approximately 5-10 minutes to answer the questions in this survey. Personal identifiers will not be collected and your responses will remain anonymous. Your participation in this survey will not affect your class grades in any way.

If you have further questions about this survey, you may contact [redacted]. If you have any questions about your rights as a research participant and would like to talk to someone other than [redacted] you may contact the Director of Institutional Review Board (IRB) at [redacted].

Thank you very much for your participation in this study.

Figure 2. Preamble text for the opinion survey.

- Demonstrate how instructions can be implemented in a chip.
- Learn how to use Java or C or assembly language to manipulate registers on a computer architecture performing a specific operation with data.
- Learn how computer organization influences high-level languages, and vice versa.
- Develop communication skills in the area of computing technology.

With respect to these facts, evaluation data, departmental expectations, and the need to revise the course to align with ACM/IEEE curricula [1], as well as provide a contemporary pedagogical methodology, following course learning measures were defined and employed:

- Create a program in C or assembly language that works with data that fits and exceeds processor cache sizes, and measure the performance impacts of doing so.
- Write a simple program in Java or C or assembly language to implement a high level program segment.
- Using a single-board computer (e.g., an Arduino board), control an LED display via software.
- Midterm and final exams (written, in-person, theoretical part).
- Design and implement a small project involving a single-board computer.

3.1. Topical Outline

Theoretical part of the course was adequate, and the textbook was well received. The hands-on part of the course was consistently being ranked low in student

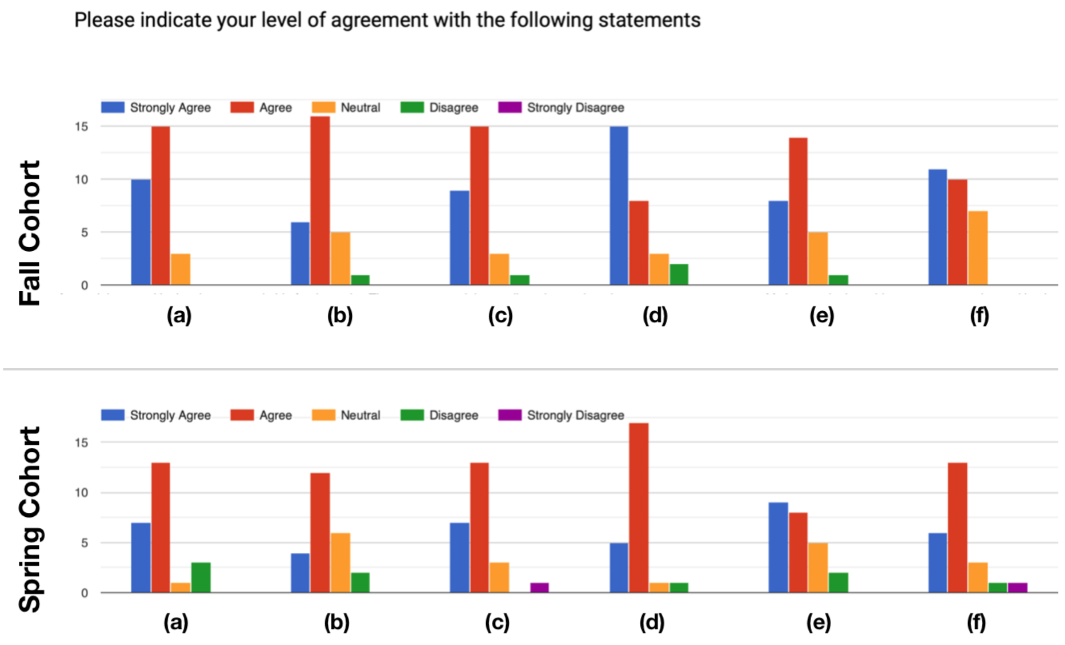


Figure 3. General questions about the course on the opinion survey.

evaluations. Our initial approach was to revamp the course by evaluating several single-board computer options, and related curricular material. We have eventually decided on using Arduino single-board computers [15], with an accompanying laboratory book that provides step-by-step instructions on completing each module in detail [6].

After revising the hands-on component, we have decided to update the textbook to better align with weekly hands-on activities, while still preparing users with adequate theoretical material, both for computer organization, and also for computer architecture parts. We have adopted Englander's textbook [5], as it also provides students a reference to how the computer is organized, with a simplified machine reference (i.e., the Little Man Computer), with accompanying machine-level instruction set, in case students choose to use higher-level programming languages to design and implement their projects.

This way, all the students have a baseline understanding of a simplified reference architecture and an instruction set, which greatly covers the general curricular expectations on assembly level machine organization, as well as machine level representation of data. Figure 1 shows the weekly schedule of the course. Overall, the course has two main sections: the first six weeks covers topics such as introduction to computer system concepts and architecture, number systems, and data formats. Post mid-term concepts start expanding into core computer architecture concepts, such as the CPU and memory, input/output, and peripherals.

Laboratory components have been aligned to create a separate online hands-on track that provides an experiential learning hour, and the two tracks (theoretical and hands-on) marge together toward the last third part of the course (starting at around week 11) to encourage students to apply what they have learned in theoretical track to the design of their project. Since the hands-on track is provided mainly as an online activity, students have the flexibility to extend learning activities with hardware components, and derive ideas and new approaches to the design of their individual projects. This online hands-on activity as a blended component was evaluated to be one of the favorite options in this course by students across both cohorts.

4. Opinion Surveys and Results

At the end of each cohort, a set of questions has been administered to students as a non-mandatory, online opinion survey via Google Forms, in addition to the paper-based course evaluations administered by the college. The online survey is administered toward the end of the semester, one week before paper-based course evaluations, and had the following preamble text as Figure 2 shows.

Opinion surveys have started with a set of questions in general about the course, where Figure 3 shows corresponding responses. Following questions were asked about the course in general, where students are asked to answer on a five-point Likert scale (Strongly Agree; Agree; Neutral; Disagree; Strongly Disagree):

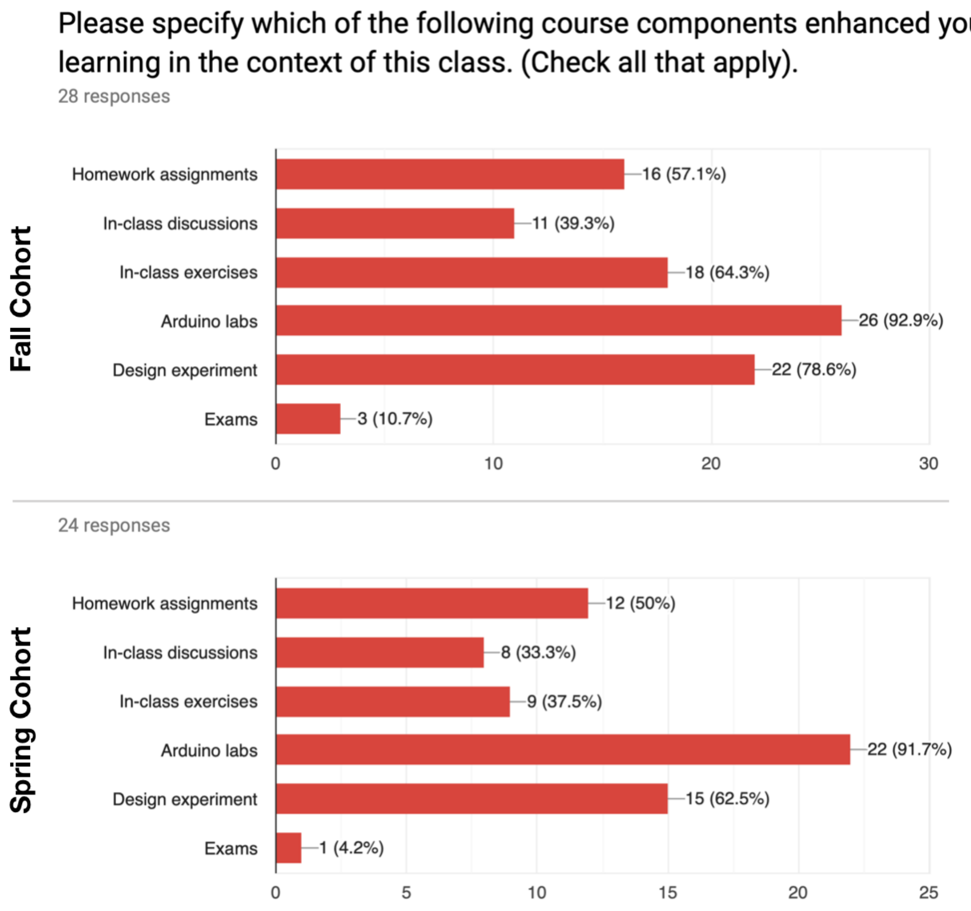


Figure 4. Student opinions on course components on learning enhancement, Fall 2016 and Spring 2017 cohorts. More than 90% of students picked hands-on modules (Arduino labs) as a component that enhanced their learning. The second most picked component was Design experiment (small project that students come up with and designed themselves.

- (a) The level of material covered in the class was suitable for the topic.
- (b) I was able to follow the material without being too overwhelmed.
- (c) The course materials contributed to my learning.
- (d) The objectives of the course were clear.
- (e) The instructor raised questions or problems that encouraged me to think critically.
- (f) My interest in the subject matter was enhanced by the instructor‘s enthusiasm.

Responses to this general question is then reinforced in a set of detailed questions provided about each module of the course, in a subsequent survey page. Immediately after that first question, main components of the course were asked to be evaluated as a multiple answer question, as Figure 4 shows.

Students' overall experience with the course showed the largest amount of change between two cohorts. While in the first cohort, students had a bigger skew toward the higher-end of satisfaction, in the second cohort, the overall satisfaction results exhibited a normal distribution (i.e., the *bell* curve), with the peak of the curve around 8/10. We attribute this to the effect of the first time offering of this course, which incorporates a single-board computer based hands-on module for the first time in the department, and the associated excitement around it. Also, during the Fall cohort, students were able to showcase their projects to the faculty, whereas the Spring cohort was not able to do it due to graduation calendar and related events scheduled toward the end of the Spring semester. Average experiences were 7.68/10 and 7.38/10 for Fall and Spring cohorts, respectively. Prior to redesign of the course, overall experience of the course averaged 3.02/5 (normalized to 6.04/10) for about ten semesters over a seven year period, with average of last three semesters

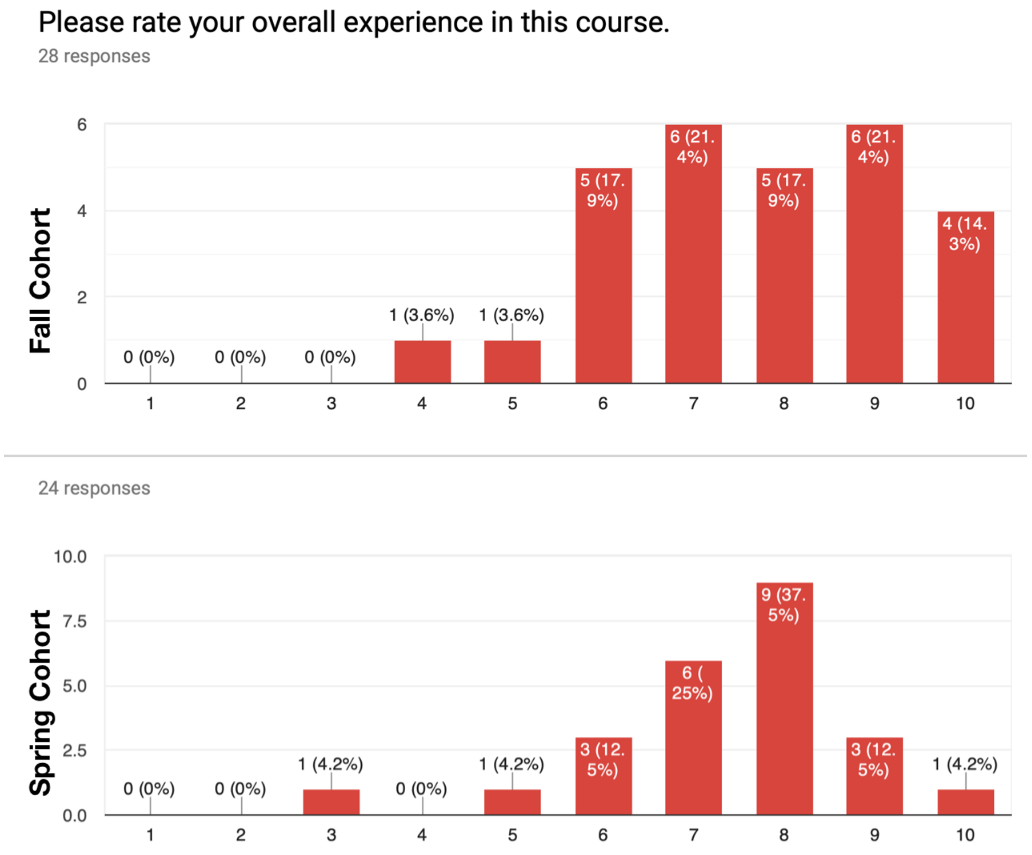


Figure 5. Overall experience of students. Fall cohort satisfaction displays a distinctive high-approval rate, and we attribute this due to the excitement with first time offering of this course, as well as the opportunity to showcase design projects to the faculty in the department. Spring cohort opinions exhibit regular bell curve peaked around 8/10, which is expected.

(pre-revision) being around 2.61/5 (normalized to 5.22/10).

5. Summary

This article provides details of a revised computer organization and architecture course, offered primarily to juniors in the computer science department of a liberal arts college. The ACM/IEEE-CS curricular standards-aligned, online hands-on and project-based learning exercises were well-received by students and have consistently ranked high compared to other learning activities. These hands-on modules provided as the core blended part of the course also boosted overall satisfaction of students, both for the course, as well as for instructor. The tradeoff was to remove the classical, assembly-based machine level programming part as the core laboratory activity. However, students were still able to learn about the assembly level machine organization knowledge area subjects, and were able to complete machine level programming exercises by relying on a simulated computer platform provided in the textbook.

References

[1] ACM/IEEE-CS joint task force on computing curricula, computer science curricula 2013. Technical report. Final Report, December 20, 2013.

[2] Robert B. Barr and John Tagg. From teaching to learning: A new paradigm for undergraduate education. *Change: The Magazine of Higher Learning*, 27(6):12–26, 1995.

[3] Andrea L Beach, Mary Deane Sorcinelli, Ann E Austin, and Jaclyn K Rivard. *Faculty development in the age of evidence: Current practices, future imperatives*. Stylus Publishing, LLC, 2016.

[4] S.P. Dandamudi. *Fundamentals of Computer Organization and Design*. Texts in Computer Science. Springer New York, 2006.

[5] Irv Englander. *The Architecture of Computer Hardware and System Software: An Information Technology Approach*. Wiley, 2009.

[6] Brian W. Evans. *Arduino Programming Notebook*. 2007. https://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf.

[7] O. Garcia. Computer organization and architecture and the laboratory sequence. *Computer*, 10(12):91–96, Dec 1977.

[8] D.Randy Garrison and Heather Kanuka. Blended learning: Uncovering its transformative potential in higher education. *The Internet and Higher Education*, 7(2):95 – 105, 2004.

[9] C.R. Graham. Blended learning systems: Definition, current trends, and future directions. *Bonk, C.J. and Graham, C.R., Eds., Handbook of Blended Learning: Global Perspectives, Local Designs*, 3(21), 2006.

[10] D. C. Hyde. Teaching design in a computer architecture course. *IEEE Micro*, 20(3):23–28, May 2000.

[11] Steven J. Johnston, Philip J. Basford, Colin S. Perkins, Herry Herry, Fung Po Tso, Dimitrios Pezaros, Robert D. Mullins, Eiko Yoneki, Simon J. Cox, and Jeremy Singer. Commodity single board computer clusters and their applications. *Future Generation Computer Systems*, 89:201 – 212, 2018.

[12] C. M. Kellett. A project-based learning approach to programmable logic design and computer architecture. *IEEE Transactions on Education*, 55(3):378–383, Aug 2012.

[13] Russell T. Osguthorpe and Charles R. Graham. Blended learning environments: Definitions and directions. *Quarterly Review of Distance Education*, 4(3):227, 2003.

[14] R. Paharsingh and J. Skobla. A novel approach to teaching microprocessor design using fpga and hierarchical structure. In *2009 IEEE International Conference on Microelectronic Systems Education*, pages 111–114, July 2009.

[15] Arduino Project. Arduino reference manual. https://www.arduino.cc.

[16] Santosh Ray and Ali Al Dhaheri. Using single board computers in university education: A case study. pages 371–377, 03 2017.

[17] Mary C. Wright, Debra Rudder Lohe, and Deandra Little. The role of a center for teaching and learning in a de-centered educational world. *Change: The Magazine of Higher Learning*, 50(6):38–44, 2018.