CMSC 22200 1 - Computer Architecture - Instructor(s) - Yanjing Li

Project Title: **College Course Feedback - Spring 2024**

Number Enrolled: **44**
Number of Responses: **21**

Report Comments

Creation Date: **Tuesday, June 18, 2024**

What are the most important things that you learned in this course? Please reflect on the knowledge and skills you gained.

Comments
Microarchitecture and how to simulate a pipeline
superscalar, out of order execution, caches, branch prediction, ISA, page tables, cache and memory coherence
How a processor works––pipeline stages, instruction set architecture (in greater detail than 144), microprocessor, branch prediction, advanced caching, multicore, SIMD, SUPERRRR SCALARRRR, parallel programming optimization, virtual memory (in greater detail, again). The overall emphasis was on the question of how to optimize the processor's various components––where 144 focused on functionality, 222 emphasizes design considerations for these components and how they impact performance. This was the first course I took that truly felt like a Computer SCIENCE course. Overall a wonderful experience.
Registers, Pipelining, Parallelism, Caching, Memory, Program Counter, Branch Prediction
Learned about instruction execution, caches, virtual memory, ISA
This course covers the lower half of the computer systems stack seen in 144 – instruction set architecture (ISA), microarchitecture (uarch), and hardware logic. We started by learning a subset of ARMv8 (called LEGv8), then the basic data path for the processing of ARM instructions at the hardware level. We then covered various uarch techniques to get higher instruction–level throughput – pipelining (and how to support it with branch prediction + operand forwarding), SIMD, superscalar, and out–of–order execution. At the end, we looked at uarch techniques to improve memory access – advanced caching and virtual memory. In general, we saw that the design choices computer architects make is all about tradeoffs in speed, space, complexity, and cost.
How the CPU processes instructions in an efficent manner. Pipelining, Branch Prediction, Caching, Out of Order execution.
CPU pipelining, out–of–order execution, branch prediction, caching, virtual memory
How a CPU is constructed, Instruction Set Architecture (ISA), various microarchitectures, CPU pipelining (this is important), branch prediction, out–of–order and multi–core/Superscalar processors, parallel architectures, cache and (virtual) memory and their synchronization challenges.
Pipeline stages, Branch Prediction, Cache, etc.
Understanding the microarchitecture of computers. The infrastructure needed to optimize the performance of instruction processing. Paralleled computing, pipelined instructions, etc...
Labs were phenomenal but lecturers were a bit
I learned a lot more about computer architecture, ISA protocols, and have a much greater understanding of the interconnection between software and hardware.
How the arm thing works
ISA and uarch
High–level overview of computer microarchitectures and design choices/tradeoffs
how processors work, pipelining, memory allocation
I learned how a processor works and the intricate parts of each component to ensure good communication between hardware and software.
– pipeline – OoO execution – caches

Describe how aspects of this course (lectures, discussions, labs, assignments, etc.) contributed to your learning.

Comments
The labs (more like projects) were probably the most helpful to actually learning and understanding the material though they were difficult
labs were not very useful, but lectures were very informative
Lectures were well–paced and featured interesting questions regarding design choices––active engagement was encouraged, even strange questions were respected. Professor Li is extremely knowledgeable and helpful. Evaluation consisted only of four projects and two finals. Projects were cumulative, so that how you did in earlier stages affected

Comments
your outcome in later ones. Projects centered around the implementation of a simulator for a processor with the ARMv8 ISA. They sharpened my C programming skills and grew my confidence in my abilities. Programming with a partner is optional––I found at least that implementing on my own allowed me a greater understanding of concepts and even helped to an extent on exams. It felt extremely satisfying as well to gradually implement the different components of the simulated processor. Exams were challenging and had relatively low medians (62 IIRC for the midterm––have not received the final grade yet). They were open–book and open–note, but were written well so as to require an understanding of the material regardless. They were reminiscent of CS 144 exams, but perhaps more challenging. Overall an awesome curriculum.
Lectures exposed us to the material, but the labs and exams forced me to actually learn it
Lectures were where the bulk of the conceptual learning came from. Content is presented in a very well–motivated and sequential manner, I never found myself wondering why we were discussing a particular topic as they all seemed connected and lead into each other. Lectures at times were very information–dense, so it was incredibly helpful of Prof. Li to have the slides used during lecture available to students on canvas to review later and follow along in class. The 4 labs were where our understanding was put to the test, with each lab asking us to provide a C implementation of some microarchitectural element (with the exception of lab1). The labs were really great and definitely my most favorite element of this course. Time consuming and daunting, but infinitely satisfying to succeed and have a complete implementation by the end. What I didn't like as much about the labs was that they were absolutely not balanced in terms of difficulty. lab1 (ISA), when it was assigned, was very intimidating to implement correctly, as we're barely given a week's exposure to ARMv8 before we're asked to give an implementation of the 25–ish instructions apart of LEGv8. Many hours were spent consulting the enormous ARMv8 specification to ensure correct logic, as the lab writeup purposefully leaves it up to the student to go digging for specification details themselves. In retrospect, not that hard now that we've worked with ARM for 9 weeks, but certainly a weed–out lab. lab2 (pipeline) is BY FAR the hardest lab of them all, as lab3 and lab4 completely depend on a correct lab2, and the lab1 code must be completely refactored before even starting on lab2. Not only a test of what we learned in class, but a great test of our ability to design software cleanly. lab3 (branch prediction) and lab4 (cache) were where I felt the labs sort of fell–off in quality. Both took significantly less time to get a semi–working implementation than lab1 or lab2, and lab4 didn't really test our knowledge of any cache concepts we learned in this class (no advanced cache concepts used, could've been a 144 lab if not for the pipeline). What is really nice about these two labs though, is that a lot of the logic is isolated into source files completely separate from lab2. This made it easy to focus ONLY on the branch predictor logic, or ONLY on the cache logic, without worrying about how to fit the pipeline into it all. A good pedagogical design, as if your lab2 was wrong, you could still make headway on labs 3 and 4 before having to go back and correct lab2. TLDR – Labs are the best part of taking CMSC 22000, but they have some logistical issues at times.
Lectures and labs were very helpful.
The labs (which are actually projects) were good, I learned a lot from simulating part of an ARMv8 CPU
Lectures were quite helpful for understanding the concepts. Lecture slides were provided for reference outside of class, but being present for lecture very much helped further the understanding of material (you won't get it all from just reading the slides). Prof. Li is a great lecturer and was always willing to slow down or go back and re–explain, give an example, and/or go through a practice problem to reinforce a concept. The "labs" (programming projects) were interesting, appropriately challenging, and gave a great opportunity to implement some of the concepts discussed in lecture. Understanding the concepts definitely made the projects easier, and going through the projects helped reinforce those concepts. Exams were non–cumulative and were a very fair yet comprehensive assessment of the course content.
Lectures and Labs are super helpful!
Lectures were helpful if a bit dull, labs helped to solidify concepts (though only really covered those from the first half). Lectures included practice problems, which were helpful for studying for exams.
The lectures and practice questions during these lectures. The labs also felt like a practical application of these lectures and were a valuable learning resouce.
Lectures provided a good overview of the topics covered in this class, while labs provided a more practical way to see the applications of what was covered in class.
Lectures are awesome, first one is a bit dry but just push on. Yanjing is awesome!
Posted lecture notes very helpful
Lectures were the main source of learning. Labs were relatively small and light projects involving implementing some of the things we saw in lecture, but weren't that in–depth
labs definitely helped understand course materials. Lectures were ok

Comments
Lectures were not too helpful, tests asked things that were only briefly taught in lecture. Labs were good practice but again, not related to the exams whatsoever and lecture also felt disconnected

Please respond to the following:

	Mean	Median	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
This course challenged me intellectually.	4.52	5.00	4.76%	0.00%	0.00%	28.57%	66.67%
I understood the purpose of this course and what I was expected to gain from it.	4.45	5.00	5.00%	0.00%	0.00%	35.00%	60.00%
I understood the standards for success on assignments.	4.25	5.00	5.00%	5.00%	5.00%	30.00%	55.00%
Class time enhanced my ability to succeed in graded assignments.	3.80	4.00	10.00%	10.00%	10.00%	30.00%	40.00%
I received feedback on my performance that helped me improve my subsequent work.	3.67	4.00	5.56%	5.56%	33.33%	27.78%	27.78%
My work was evaluated fairly.	4.45	5.00	5.00%	0.00%	5.00%	25.00%	65.00%
I felt respected in this class.	4.20	5.00	10.00%	0.00%	10.00%	20.00%	60.00%
Overall, this was an excellent course.	4.05	5.00	9.52%	4.76%	14.29%	14.29%	57.14%

Additional comments about the course:

Comments
It's definitely a time sink but doable with lots of office hours and probably a lot easier if this stuff is fun for you
projects were fun and pretty straightforward
This course made me confirm my decision to major in computer science. I loved it. Professor Yanjing Li rules. Joshua Ahn is an excellent TA as well.
Your lab grade is determined entirely by a given auto–grader script and test cases. This is fine, except the auto–grader script is embarrassingly slow to execute in labs 3 and 4. If you're a very test–driven programmer, you will HATE the last two labs, because lab3 testing script takes an hour to finish execution and lab4 testing script takes half an hour. It's even worse with the hidden test cases given after the deadline is passed. I ran my solution on the lab3 hidden test cases on both my local machine and the CSIL linux server. Both took roughly *three hours* to finish execution and see what my grade was! The instructors seriously need to use a more optimized script to evaluate student solutions, it is so horribly demotivating to have to wait an hour every time you want to see if you fixed a bug in your code. Perhaps assign the task of rewriting the testing scripts to a TA (especially since one of the students took it upon themselves on ED to write their own more optimized testing script in Python for other students to use)
This course was much easier after having taken more difficult systems courses previously (including Networks, Operating Systems, and Computer Security). Although, this course does not take nearly as much time nor are the projects nearly as difficult. If you haven't taken these courses (or similar) previously, I'd highly recommend working on the projects/labs with a partner. Although partners are optional, I found working with another person to be a huge help. I would definitely not recommend doing the labs alone if you don't have this previous experience working on a large, difficult project (these are basically from scratch and they build on each other!). However, if you are confident in your systems programming & C abilities, the labs are definitely doable alone with some dedication and time management.
Little practice with second–half concepts before the final. Exams had a lot of manually tracing computations by hand, which was error–prone and didn't feel like a good test of understanding. They were also graded with no feedback at all (just the number of points marked off for each problem), and no reference solutions were provided. This made it hard to use them to improve, and also hard to catch grader errors. (Of which I'm sure there were plenty––one of my problems was mis–graded since the grader missed that I put two answers on one line. And it seemed like they didn't even read the required justifications for the T/F questions. It seemed pretty rushed.)
Great class content wise. Presentation of the material could definitely be improved. Classes should not be skipped and content felt too rushed. Lectures were theoretical while exams were more practical. Labs are great learning opportunities but they are time consuming. Start early!!! Labs genuinely made me a much much much better programmer. This class has improved my coding skills more than any other class I've taken, but it is challenging.
Pretty well run and organized.

I would recommend this course to:

	No	Yes
Highly-motivated and well-prepared students	5.00%	95.00%
Anyone interested in the topic	15.00%	85.00%

Thinking about your time in the class, what aspect of the instructor's teaching contributed most to your learning?

Comments
Posting lecture slides was nice
lectures and going through practice problems were helpful
Her openness to answering questions, her depth of knowledge, and her great attitude towards the class all contributed to my learning.
The practice problems
Prof. Li is a rather terse instructor, but it's clear she is extremely knowledgeable on the topics we learned. Anytime a student asked a question, if she chose to answer it herself, she would always be able to give a sensible answer in direct response. Her organization of the course is also excellent, everything is laid out nicely on the course website and super accessible.
Lectures and labs.
Prof. Li's lectures were excellent, engaging, and well–paced. I felt I walked away with a great understanding of the core of each concept; lectures were not bogged down by any fluffy details. However, if I ever had confusions or curiosities about details, Professor Li was always able to answer them clearly and connect back to the larger concept. The slides were a great reference to have during and after the lectures, so much so that I found myself no longer taking notes and instead just paying closer attention in the lecture and relying on the slides to recall concepts later. The provided practice problems (only some of which were discussed in lecture) were especially helpful for preparing for exams–– many of these practice problems were nearly identical to exam questions or covered almost exactly the same content.
Lectures are super helpful!
During lectures, there were practice problems that I felt were valuable moments to 'touch base' on the material.
I liked how the instructor always made sure to pause during lectures and provide students a chance to ask any questions about what she just covered.
Lectures were really good
I like that we covered all the material we set out to cover and didn't fall behind
Lectures
She always posted the slides on the course website. Making it easier to catch this class' fast pace
– Lectures – Labs helped a great deal to understand how a processor works in practice

What could the instructor modify to help you learn more?

Comments
Having more practice problems/a practice exam
n/a
Perhaps slightly more evaluations? the exams felt like a one–and–done sort of deal, it felt like there was no opportunity to test your understanding before taking them, and once you see what you've done wrong there is not much to do. I know Prof. Borja Sotomayor's Networks classes features routine exams to confirm understanding—this could be helpful, although I don't know in what time they would be done.
More practice problems prior to exam would be helpful.
Unsure
Many times, when a student would ask a question, Prof. Li would choose not to answer herself but relay the question back onto the student. A question such as "Why is that true?" would often get a response of "why do you think it might be true?" This was novel for the first few weeks, but overtime it just discouraged me from asking any questions and instead saving it for ED later. If I'm asking a question, it's because I don't have an inkling myself on what the answer could be, otherwise I wouldn't be asking the question. Prof. Li has an amazing breadth and depth of knowledge, so I wished she answered the questions asked herself, so we could've picked her brain more.
1. Have exams that more clearly reflect the course content. There aren't many practice problems and the answers provided aren't always really useful ("many possible answers"? How does that help?) 2. The lab writeups should be more clear, maybe add the things that TAs have to clarify on Ed to the writeup 3. Can we please switch to git for projects? SVN hasn't been relevant for about 15 years, it's mental that I can't have branching in version control in 2024
Perhaps check in on the TAs understanding of the projects a little more? I could tell that one TA was far more prepared to answer questions about the labs than the other, who had a tendency to have difficulty understanding questions asked in office hours and on Ed Discussion, and often could only look at the reference solution. I found myself going to office hours less frequently than I usually would because I didn't feel I could rely on one of the TAs for a reliable answer to conceptual or design questions about the labs.
Maybe more practice problems!
Even more practice questions could be nice, they're very helpful.
The lectures often tackled concepts at a high level. It may have been nice to break away from this occasionally so we could be slightly more prepared for labs which focused more on practical application.
Maybe for the first material spend less time waiting on the slides, it really killed the momentum of the class
Maybe improve the answer keys to the practice problems a bit
Meatier labs – this could be better facilitated by a sequential lab structure in which a baseline is provided for each new lab? i.e. failing to complete lab 2 fully doesn't mean you're boned for lab 3 (this would enable prof li to make lab 3 harder/cover more topics)
lectures, going slower and making the slides more descriptive of what was actually being taught. There were a lot of times where the content seemed to be kind of out of nowhere
Teach us what is going to be asked for in the tests on top of the more theoretical knowledge. The bridge between lecture to how to solve a problem was huge and there wasn't enough help to make this connection
Please write a better testing script. The current testing script (for lab2 onwards) works by running both ref and student–sim 1 cycle each (and comparing), then 2 cycles each, and so on— quadratic time complexity. It was painful to see the testing script slow to a crawl near the end of extremely long tests. This also makes iterating on improvements very time–consuming. Ideally, the testing script should be modified to run each ref and student–sim as single, continuous processes, instead of continuously halting and restarting them. This may require adding locking to ref and student–sim src so that they can be synchronized to run only 1 cycle at a time (and until further notice from the testing script parent process). One potential solution is to convert the ref–sim into a library that exposes a 1. initialization function that accepts a lock pointer for synchronization purposes and 2. a generic "run" function. The testing script should also used shared memory or pipes to write outputs instead of highly costly file writes. Please ensure that the lab instructions do not have obvious inconsistencies (relative to the behavior of the reference simulator). I was astounded that the lab3 specification for the flushing conditions for branch prediction was so obviously incorrect, and I was more astounded that no one in the past 8 years since the inception of this project pointed out this error.

The Instructor . . .

	Mean	Median	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	N/A
Organized the course clearly.	4.47	5.00	5.26%	5.26%	0.00%	15.79%	73.68%	0.00%
Presented lectures that enhanced your understanding.	4.26	5.00	5.26%	5.26%	5.26%	26.32%	57.89%	0.00%
Facilitated discussions that were engaging and useful.	4.11	4.00	5.26%	5.26%	10.53%	31.58%	47.37%	0.00%
Stimulated your interest in the core ideas of the course.	4.26	5.00	5.26%	5.26%	5.26%	26.32%	57.89%	0.00%
Challenged you to learn.	4.58	5.00	5.26%	0.00%	5.26%	10.53%	78.95%	0.00%
Helped you gain significant learning from the course content.	4.53	5.00	5.26%	0.00%	5.26%	15.79%	73.68%	0.00%
Was available and helpful outside of class.	4.11	4.00	5.26%	0.00%	15.79%	31.58%	42.11%	5.26%
Motivated you to think independently.	4.32	5.00	5.26%	0.00%	10.53%	26.32%	57.89%	0.00%
Worked to create an inclusive and welcoming learning environment.	4.32	5.00	5.26%	0.00%	5.26%	36.84%	52.63%	0.00%
Overall, this instructor made a significant contribution to your learning.	4.16	5.00	5.26%	5.26%	15.79%	15.79%	57.89%	0.00%

Please include the name of the TA/CA/Intern you are evaluating. What aspects of the TA's teaching contributed most to your learning? What could the TA modify to help you learn more? Please include any additional feedback for the TA/CA/Intern.

Comments
Joshua Ahn was great and Jie was not so great. Maybe hire TAs that have actually taken the class next time and have both TAs host lab sections. Josh was really great for helping out on the labs, don't think I would've passed without his office hours.
josh was helpful on ed; i did not interact with jie
Joshua Ahn. I never went to office hours but his responses on ed were clear, humorous, and insightful. He also had lightning–fast responses––god bless him.
There were two TAs for this class, Josh and Jie. Josh was an absolutely stellar TA, it seemed that everything he posted on ED in response to students was always infinitely helpful to clarifying how to go about conceptually implementing the labs. 5–star TA. I won't lie, I'm not sure how Jie contributed to this course. 9 times out of 10, a student question would be answered on ED by Josh. If Jie happened to answer, it was rarely satisfying (and Josh would step–in anyway and give an answer that at–times corrected Jie). This is a comment regarding the function of a TA in this course: Since the lab implementations can span thousands of lines of code, it becomes very difficult for a TA to help students directly with code–specific problems. All that can really be done is making suggestions such as "make sure you're doing this", which isn't too helpful if we already conceptually understand what needs to be implemented. I'm not sure how this issue could be solved, but it is a problem I noticed. As a result, I never went to lab past second week, there was rarely a reason to attend since only I knew how the thousands of lines of code could be tweaked to solve my problem..
Joshua
Joshua Ahn. I think they were a solid TA and was very prompt in responding/addressing questions in EdDiscussion.
Jie and Joshua were great!
Josh Ahn was extremely helpful for labs on Ed.
Joshua Ahn – super responsive and seemed to be the one primarily responsible for how well structured and organized the labs and associated grading infrastructure were
Joshua Ahn the goat. He was extremely helpful during labs with being super descriptive of how the labs/projects would be ran, aswell as help understand what we should be taking out from the projects.
Joshua
Joshua Ahn, I didn't attend any of his OH's, but I he was extremely responsive, attentive and precise on edstem. Wonderful TA.

The TA/CA or Intern. . .

	Mean	Median	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	N/A
Facilitated discussions that supported your learning.	4.45	5.00	9.09%	0.00%	9.09%	0.00%	81.82%	0.00%
Gave you useful feedback on your work.	4.55	5.00	9.09%	0.00%	0.00%	9.09%	81.82%	0.00%
Stimulated your interest in the core ideas of the class.	4.20	5.00	9.09%	0.00%	18.18%	0.00%	63.64%	9.09%
Challenged you to learn.	4.27	5.00	9.09%	0.00%	18.18%	0.00%	72.73%	0.00%
Helped you succeed in the class.	4.55	5.00	9.09%	0.00%	0.00%	9.09%	81.82%	0.00%
Was available and helpful outside of class.	4.64	5.00	9.09%	0.00%	0.00%	0.00%	90.91%	0.00%
Overall, this individual made a significant contribution to your learning.	4.45	5.00	9.09%	0.00%	9.09%	0.00%	81.82%	0.00%

How much did the following elements of the course contribute to your learning gains?

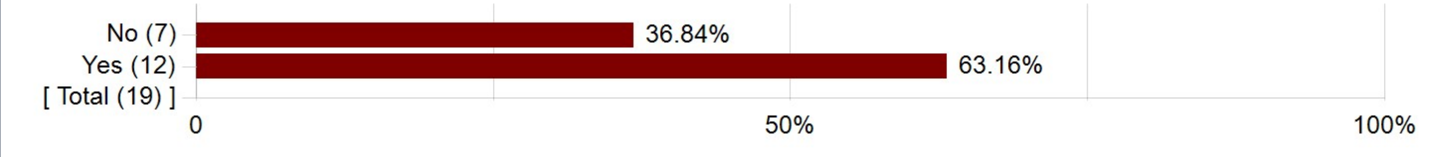
	Mean	Median	No Gain	A Little Gain	Moderate Gain	Good Gain	Great Gain	N/A
Laboratory Experience	3.50	4.00	20.00%	0.00%	0.00%	40.00%	20.00%	20.00%
Field Trips	N/A	N/A	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%
Library Sessions	N/A	N/A	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%
Review Sessions	N/A	N/A	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%
Writing Seminars	N/A	N/A	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%

Other course elements not mentioned above:

Comments
While the labs helped the most in learning the content, they were SUPER unorganized. The lack of tests (specifically for lab3) and the description of the lab itself was fairly unclear at times.

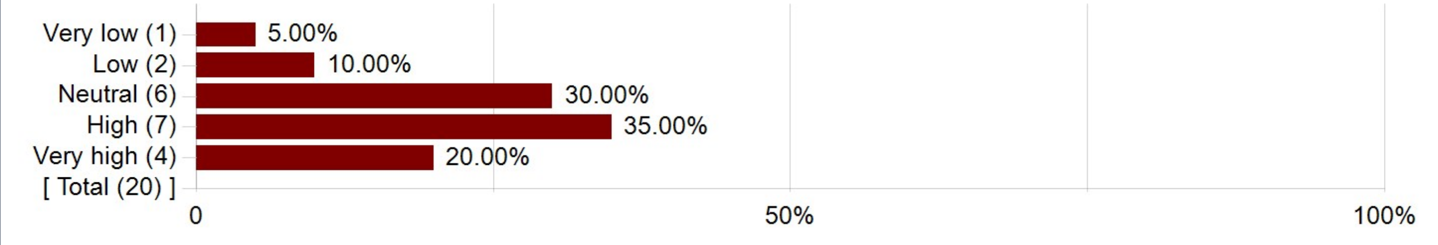
Is this class a requirement of some sort?

Is this class a requirement of some sort?

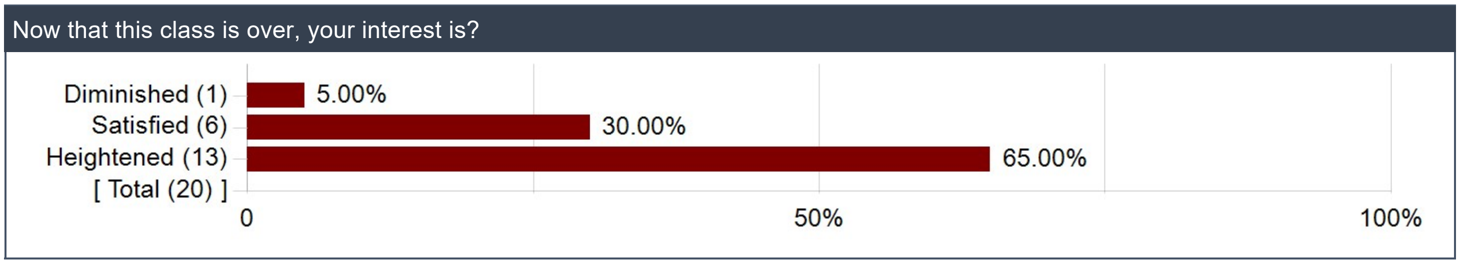


Prior to starting this class, your interest level was?

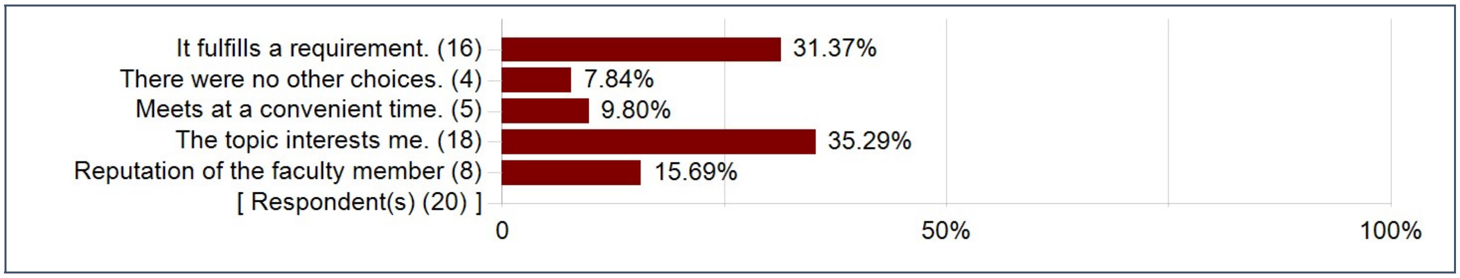
Prior to starting this class, your interest level was?



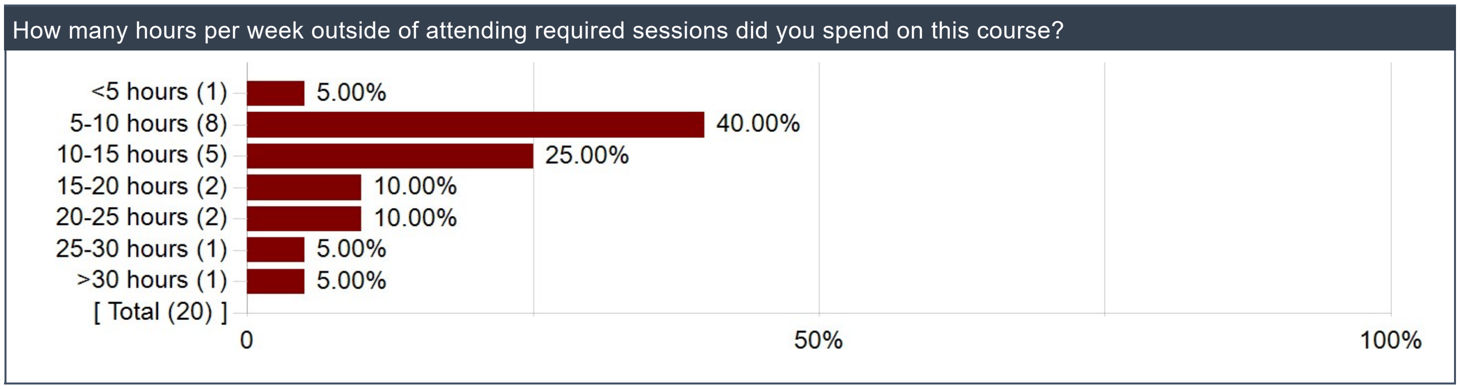
Now that this class is over, your interest is?



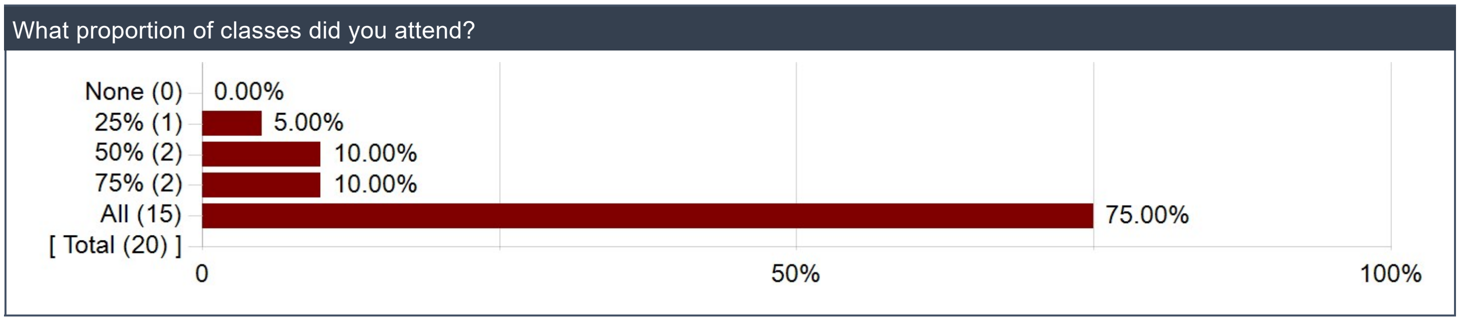
Why did you choose to take this course? (Select all that apply)



How many hours per week outside of attending required sessions did you spend on this course?



What proportion of classes did you attend?



Please comment on the level of difficulty of the course relative to your background and experience.

Comments
Pretty much 144 on steroids. Labs (projects) were doable without a partner and I felt like the class itself was very fair.
The course was somewhat challenging, but labs being due biweekly made it manageable. Granted, I had already learned my lesson regarding organizing my time, thus I was okay, but even then I had to use the late days and turned one of the assignments in with a one–day penalty. Overall, though, a generous course—start ahead of time as always and READ THE LAB HANDOUTS IN DETAIL—planning ahead will go a long way (vs. debugging your way through).
Quite difficult, I only took 15400
To take this course, you must have taken 144/154, which I definitely feel is sufficient preparation, so everyone will be going in with pretty much the same baseline knowledge. The labs are for–sure the hardest portion of this course. I chose to work on them alone and found the experience pretty comfortable. You're given 2 weeks or more for each, which is plenty of time to chip–away at the code and mull over causes of bugs as you work on other classes. Being super comfortable with C is definitely a must, and I cannot count the number of times I was able to save myself from nasty bugs by using LLDB/GDB and Valgrind. Though, coming right from 144, I imagine pretty much everyone has these skills if you're in this class. For those who are curious, the course text is "Computer Organization and Design ARM Edition". It's a nice book, but a lot of the content is redundant with CS:APP3 from 144, and the content that is new for 22200 is covered at too cursory a level to be helpful for the labs or exams. I found the lecture slides to be a much better resource for this class than the textbook.
This is a quite reasonable Systems course, especially compared to Networks, Operating Systems, or Computer Security. Having taken those three courses previously (among others), I found this course interesting, but not exceptionally challenging given my background. The labs are very reasonable workload wise, especially if you work with a partner, which I recommend. This would be a very doable course for a student fresh out of CS 144, and it will absolutely strengthen your skills and understanding for taking Networks and/or OS in the future.
Labs are the only homework (one due every 2 weeks). With a partner, I didn't find them too hard (though my partner kind of carried me). Most of the grading tests are provided so you can guarantee almost full credit if you want. Exams were also very difficult—averages of 65% and 75% on midterm and final, respectively. Hopefully there's a generous curve.
I was a bit rusty in C programming, but the course was not overbearing in this capacity. The actual course content starts from foundational concepts and, I would argue, does not require a heavy background (especially considering the intro CS sequence is a prereq).
I took this course right after finishing the CS intro sequence. I didn't find the course too challenging and some of what was covered overlapped with what I learned in CMSC 14400.
The C is actually trivial. You don't really use many advanced features or libraries. It is mainly about conceptualizing the CPU, which is the intellectual core of the class
Pretty easy systems elective. Would recommend taking right out of the intro sequence
This class is difficult for all backgrounds. Unless you already dealt with similar topics, this class will challenge you.
Easier than Operating Systems, but has a similar workload structure. I would recommend taking Comp Arch first. I imagine if previous iterations this class included labs on out of order execution and multiple caches, it would've been a lot more difficult.
I took networks, OS, and security prior to this class and I found the labs/projects in this class to be much more reasonable. Working with a partner is not necessary, but helpful. The hardest lab, lab2, is about comparable to pintos2 in OS, but the rest of the labs are MUCH easier. The midterm was ridiculously brutal; time was limited and questions were highly technical with long chains of state manipulations (and so it was easy to make stupid mistakes). The final was much more reasonable.