CPSC 213 Syllabus *(Last Updated: Aug 31, 2023)*

Description

The primary goal of 213 is to help you develop a model of computation that is rooted in what really happens when a program executes.

In the first 8 weeks of the course we will implement a simple instruction set in a hardware simulator and then examine how features of C are implemented in this instruction set. We will refer back to Java when considering memory management and polymorphism and to Dr Racket when considering functions as parameters. You will also develop an ability to read and understand small assembly-language programs.

In the remaining five weeks devices, asynchrony and thus asynchronous programming are introduced. Asynchrony is used to motivate threads and threads to motivate synchronization. You will see both how these abstractions are implemented and how they are used. You will see the connection between thread switch and procedure call. You will be introduced to the notion of atomicity and see why atomic memory-exchange operations are needed to implement synchronization. You will also examine the difference between busy and blocking waiting and solve a set of problems using monitors and condition variables and semaphores in C.

Topics

- Numbers and Memory
- Static Scalars and Arrays
- Instance Variables and Dynamic Allocation
- Static Control Flow
- Procedures and Stack
- Dynamic Control Flow
- I/O Devices and Asynchronous Programming
- Virtual Processors (Threads)
- Synchronization

Contacts

Use Piazza to contact the course staff when your issue is related to course content, including questions about material presented in class, questions about assignments, or logistical questions. Given that your question may be answered by any of the course staff, as well as other students if your question is public, you should expect faster turnout there as compared to email.

All administrative questions should be directed to our course coordinator at cpsc213-admin@cs.ubc.ca.

Instructor contact information and office hours are listed on Piazza.

Textbook and Other Resources

The primary textbook for the course is the *213 Companion.*

The *213 Companion* and the *SM213 Simulator* are available for download from *Lab 0* on Prairie Learn.

Additional material is found in the book: Computer Systems: A programmer's Perspective (3rd edition), by Randal Bryant and David O'Hallaron.

Course Schedule

The course schedule is available in Lab 0 on Prairie Learn. The course is divided into **Units** 1a-1f and 2a-2c. The schedule shows when we estimate we'll be covering each of these units in lecture.   There is a Canvas module for each Unit that lists the unit's reading assignments, learning objectives, and lecture slides.

Additional lecture material including in-class code examples, exercises, and summary videos are available on Prairie Learn. The videos are short summaries organized by topic. They are provided in lieu of lecture streaming or recordings. Additional videos will be added, enthusiastically, by request. Please do not hesitate to request a video for any topic at all if we don't have one and if you think it will help (contact Mike feeley@cs.ubc.ca).

The schedule also shows which topics are covered by the week's lab and assignment. and by the three quizzes. Colour coding connects the lectures to the labs and assignments. Generally, a week's lab and assignment will cover the material completed in the prior week's lecture.

Labs

Lab assignments occur in your 1-hr scheduled lab section each week. They are interactive, lead by a TA. To receive credit for the lab you must attend your registered section and actively participate.

The other, 2-hr, lab section is open for help with assignments and you can attend any of these you like (priority given to students in their registered section).

Assignments

Assignments start at the beginning of the week and end at the end. Due dates for Assignments are in your Canvas calendar. There is an automatic, no-penalty 24-hour grace period after the due date. After that, with very, very limited exceptions, no late assignments will be accepted.

Quizzes

The schedule also shows the weeks for the three, 1-hour **Quizzes**, which you can take at a day and time of your choosing (from a set of options we will provide). These quizzes are in-person, and held in the Computer Based Test Facility. You can only view your quiz while in the facility. Each quiz covers material covered up to the end of the preceding week, generally excluding the material covered by prior quizzes.

Practice Quizzes

Not for marks. You can take these from anywhere, see the answers, and repeat them as often as you like. Obviously a good way to study for a Quiz, **but Quiz questions are a bit different the Practice Quiz questions.**

Assessment

Your grade in the course will be based on the following components:

1%	Class Participation	*full marks for 80% participation on iClicker questions*
1%	Lab Participation	*full marks for 80% active participation in 1-hour group labs*
25%	Assignments (10)	*best 8 of first 9 plus assignments 10*
33%	Quizzes (3)	*can be improved by doing better on the final (see below)*
40%	Final Exam	

To pass the course you must

- achieve a mark of at least 50% on the weighted average of the quizzes (after adjustment) and final exam (note that it is not required to achieve a 50% on them individually); and
- *submit* 80% of the homework assignments (with reasonable attempts to most questions).

Lab Participation

You receive full credit if you ***actively participate*** in 80% of the weekly lab sessions. You do not have to answer any of the questions correctly, but you must participate.

Assignments

The first 2 assignments are individual work. For the remaining you will be permitted to work with a partner, if you like. Your mark for the first 9 assignments is determined by your best 8 marks.

Quizzes

There are three, 1-hr, in-person quizzes that together comprise 33% of your final course grade (11% each).

You can improve your quiz grades with your final. The way this works is that every learning goal assessed on a quiz is assessed again on the final. If you do better on the final-exam question for a learning goal, then your quiz mark for the same goal will be improved to 80% of your final mark. So, for example, if Quiz 1, Question 3 and Final Question 2 both assess the same learning goal(s), and you get 70% on the quiz question and 95% on final question, your mark for the quiz question will be improved to 76% (i.e., 95% x 80%).

Online Community

Piazza provides our official forum for discussing course material; getting help from each other, the TAs, Geoff, and Mike; and providing help to each other. We have also setup a ***Discord*** server for you to optionally use to socialize with each other online (*TBD*). The Discord server is not monitored but a few TAs have decided to join in order to be part of this community. Use Discord to discuss whatever you like with each other, but please do not ask TAs for help on Discord. Instead, direct this sort of stuff to Piazza, which is monitored, and where everyone can benefit from each other's questions and answers.

Prairie Learn

Most course material and all assessments are on *Prairie Learn*. These assessments are tagged by learning goal.

A supplementary web page (https://www.students.cs.ubc.ca/~cs-213/cur/marks) organizes all of this material by learning goal and summarizes your progress (i.e., grades) by goal.

After this Course You Will ...

- Be a better and *full-stack* programmer.
  - have a deeper understanding of the features of a programming language;
  - understand the execution of your program at various levels of abstraction;
  - be able to more easily learn new programming languages;
  - be able to evaluate design tradeoffs in considering languages most appropriate for solving a given problem.
- Appreciate that system design is a complex set of tradeoffs: a system will not have exactly one optimal answer, there are often many sub-optional answers. Experience with tradeoffs prepares you to deal with tradeoffs in design in real world programming scenarios.
- Develop distinctions between static and dynamic components of programs and systems and be able to describe their implications.
- Utilize synchronization primitives to control interaction in various situations including among processes, threads, and networked communication.
- Understand how computer systems work.

Course Learning Outcomes

1. Using a hardware based model of execution, reason about the limitations, vulnerabilities and idiosyncrasies of the behaviour of a particular program, specifically concerning performance, bugs and security vulnerabilities.
2. Using a hardware based model of data, reason about how programs access data using different types of variables, including the implicit and explicit use of memory references.
3. Translate a statement from a high-level programming language into assembly language; from a large block of assembly language, identify groups of instructions that correspond to high level language features and then write an equivalent high level programming language expression.
4. Identify and correct memory management bugs, particularly in languages with explicit deallocation, and use best practices to write code that is less likely to incur such issues.
5. Compare and contrast how Java and C are translated into a language the CPU understands; identify common features that are implemented in significantly different ways in either language (for instance, memory management, and the duality of subtype polymorphism in Java and function pointers in C). In doing so, explain the tradeoffs associated with each.
6. Reason about the execution of concurrent programs, incl. real time interrupts, and use both asynchronous programming and threads to write concurrent and/or parallel programs. Explain the tradeoffs associated with each.
7. Solve problems using monitors, condition variables and semaphores.

Course Rules and Academic Integrity — *READ THIS CAREFULLY*

Assignments are individual work (or group work, as indicated). You must not share any portion of your assignment with anyone (other than your group parter). You must not consult online or other resources that include any portion of a solution to an assignment.

You must not use a tutor for help with an assignment.

You must not use a generative AI tool (e.g., ChatGPT) for help with an assignment.

All work that you submit is assumed to be your own work (or in the case of group assignments, a collaboration of you and your partner and no one else) unless you explicitly indicate otherwise.

Lecture iClicker answers attributed to you must have have been entered by you while you were attending the lecture in person. You must not ask another student to answer for you.

Course material is provided for registered students only. You must not provide this material anyone else in any form.

Violation of these course rules constitutes academic misconduct. Other examples of misconduct and a more detailed description can be found on the department page on collaboration and on the University Academic Integrity page.