ECSE324 : Computer Organization

Introduction

Christophe Dubach

Fall 2021

Revision history: Warren Gross – 2017, Christophe Dubach – W2020/F2020, Brett H. Meyer – W2021, Christophe Dubach – F2021

Timestamp: 2021/08/30 08:59:00

Disclaimer

Lectures are recorded live and posted unedited on *MyCourses* on the same day.

It is possible (and even likely) that I will (sometimes) make mistakes and give incorrect information during the live lectures. If you have any doubts, please check the textbook, or ask on the online forum for clarification.

A Brief (Professional) History of Christophe Dubach

- 2005: MSc

- 2009: PhD, *Using machine-learning to efficiently explore the architecture/compiler co-design space*
- 2012: Lecturer (Assistant Professor)
- 2017: Reader (Associate Professor)

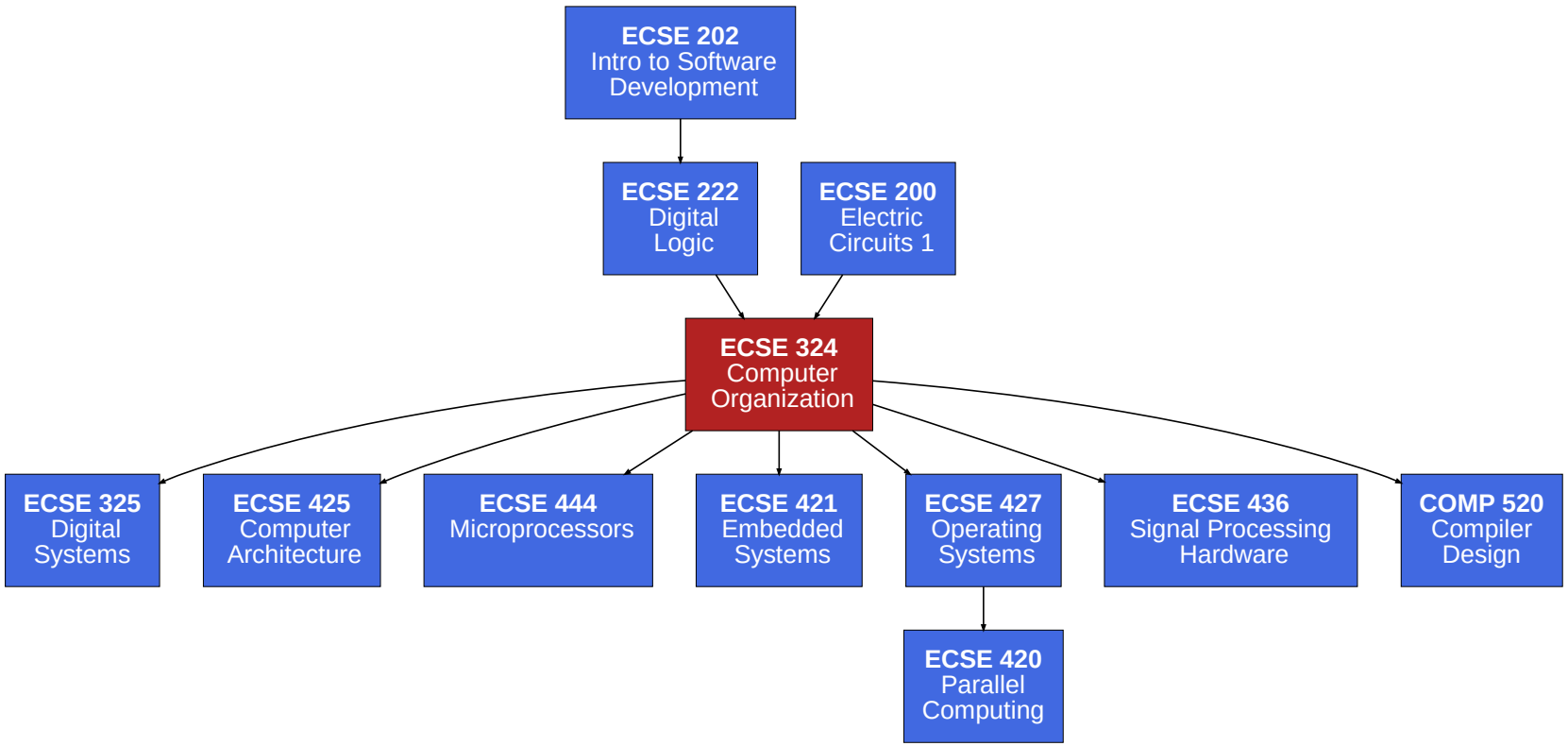
- 2010: Visiting Scientist
  LiquidMetal: a language, compiler, and runtime for high level synthesis of reconfigurable hardware

- 2020: Associate Professor (ECE/CS)
  - `ECSE 324` : Computer Organization
  - `COMP 520` : Compiler Design

What is Computer Organization?

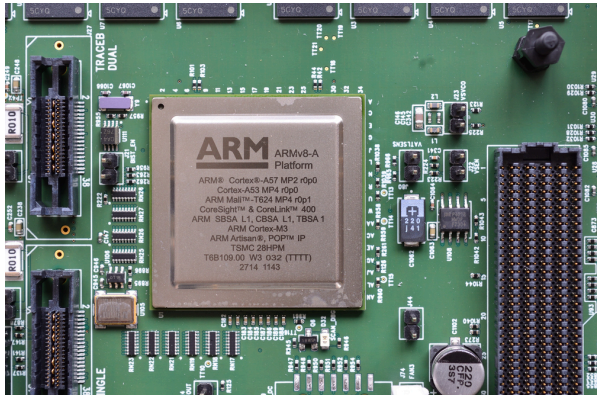
- We will learn about the *design of computers*
  - Organization of computer components
  - How software requirements influence hardware organization
- In previous classes you learned about:
  - How to write programs in Java (ECSE 202)
  - Building blocks of digital hardware (ECSE 222)
- This class will fill in the missing details to tell the story of how a digital machine called a computer runs programs

ECSE 324 in context



What will you take away from this class?

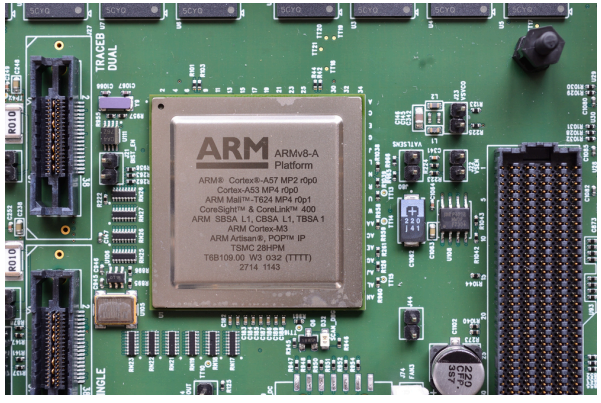
- ECSE 324 focuses on the ARM Cortex-A series of processors
  - ARM processors are used in almost every smartphone (95%)
  - Cortex-A is one of the most popular modern high-performance processor families
- Computer hardware
  - The main internal structures of computers
  - How this hardware executes a software program



source: CC0 1.0 Universal Public Domain Dedication

What will you take away from ECSE 324?

- Computer software
  - How to write programs at the machine level in *assembly* language
- Software compilation
  - How high-level languages (e.g., Java) are translated into assembly and *machine code*



source: CC0 1.0 Universal Public Domain Dedication

Why learn assembly language?

- Learning assembly is the best way to understand hardware: before we can describe hardware, we must first know what it does!
- Assembly programming is important for power and performance: low-level programming is tedious, but can beat compilers, especially for embedded and signal processing systems

Why take ECSE 324?

Is this a hardware or software class?

Both! More precisely, *computer organization* is where software meets hardware.

The computer organization concepts we will cover have not changed for several decades and are unlikely to change in the near future.

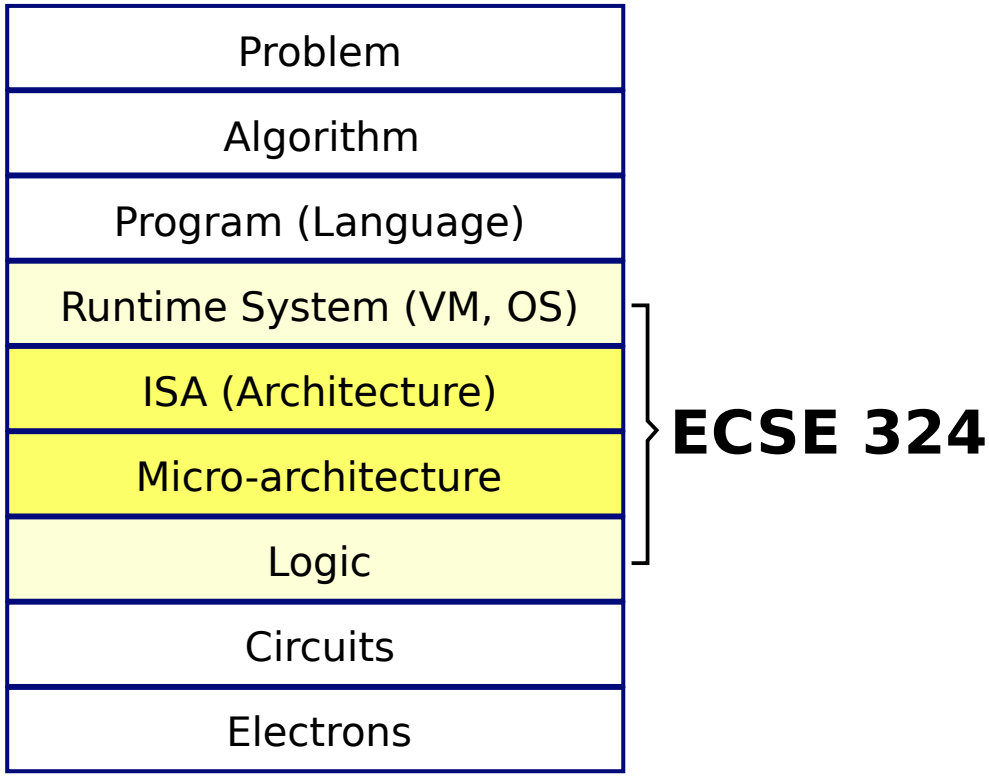
This course is therefore the foundation for efficient design of

- Software, especially for embedded systems
- Hardware, especially microprocessor and multi-processor systems

Why take ECSE 324?

- **Computer engineers**:
  This course sets the stage for your remaining studies; learn the concepts motivating computer system construction.
- **Software engineers**:
  Learn how to write efficient code by understanding the machine it runs on.
- **Electrical engineers**:
  Computers are at the heart of almost all EE systems; learn how they work to better work with them.

Computing system stack: Big picture



What knowledge do you need from ECSE 202 & ECSE 222?

- You should be able to write programs in a “C-like” *high-level imperative language*, such as C or Java
  - This course will use C as the example language
- You should be able to *think algorithmically*
- You should the know basics of *digital logic*
  - Binary number representation
  - Logic gates and flip-flops
  - Binary addition
  - Finite-state machines

  $\Rightarrow$ These are the building blocks of computers!

Course information

Course information is available on
`http://ecse324.ece.mcgill.ca/`.

- Lecture notes
- Lab instructions
- Assignments/Tutorials
- Links to Zoom meetings
- Office hours, ...

All other course communication will occur on *ED*:

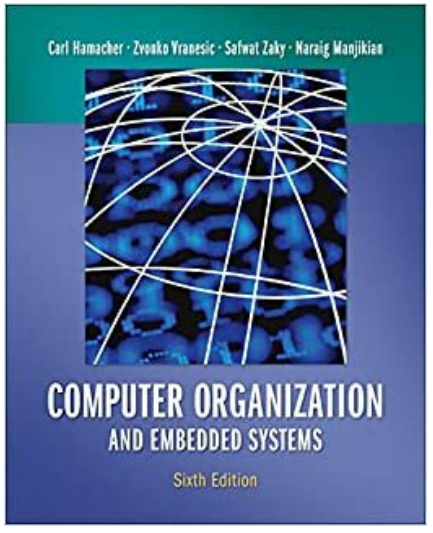
- Important announcements about the course will be posted there
  I assume you read them
- Make sure to register:
  `https://edstem.org/us/join/ng4MNK`

Textbook

The textbook is *required*: lecture will summarize, or elaborate textbook content, but *not* reproduce it.

Computer Organization and Embedded Systems

Hamacher, Vranesic, Zaky, and Manjikian 6th Edition, McGraw-Hill, 2012.



Available at

- VitalSource, an online academic bookstore
- McGill Bookstore Le James

Course syllabus

We will loosely follow the text over the course of the semester.

1. Computer Technology and Abstractions (Chapters 1 and 2)
2. Instruction Set Architecture (Appendix D)
3. Software (Chapter 4)
4. Input/Output (Chapters 3 and 7, Appendix D)
5. Memory (Chapter 8)
6. Processor Implementation (Chapters 5 and 6)
7. Arithmetic (Chapter 9)

Course meetings

You will have the following meetings during the semester:

- 3 hours of *lecture* per week: online
- 2 hours of *tutorial* per week: in-person
- 2 hours of *lab* per week: in-person

Links to the zoom meetings is available on the course website.

All lectures will be recorded, and a link to the recording will be posted on MyCourses on the day of the meeting.

For registration, timetable issues, conflicts, etc, please do not contact course staff. Instead, please contact the McGill ECE Undergraduate office (undergrad.ece@mcgill.ca).

Labs

The laboratory experiments are meant to reinforce the lecture material, and form an integral part of the learning experience in this course.

All labs are to be completed individually.

There will be a total of four labs:

- Lab 0 is not graded
- Labs 1–3 are graded
  - For each lab, you will have to present a demo, and
  - Produce a short report.

Due to the difficult to predict situation, labs will be performed using an online emulator.

Tutorials

Tutorials are here to help you understand how to solve problems.

- We will release assignments a few days before each tutorial session to give you a chance to try to solve the assignments on your own.
- During the tutorial session, the TA will explain how to solve the assignments and answer any questions you may have.

This should prepare you for the midterm and final exam.

Tutorials and labs schedule

There will be no labs/tutorials this week or next week.

- Tutorials and labs start the week of September 13

Sessions:

- To help TAs load balance, only attend the session you registered for.
- Only attend the session you are registred for: any graded activities will not be graded if done in the session you are not registered for.
- To switch lab (or tutorial) sessions, do it through the University (do not contact the instructor or TAs about this, we cannot help).

Evaluation

Final exam	50%
Lab experiments	30%
Midterm exam	20%
Online participation bonus	5%

- The midterm will be held on October 25, 2021.
- The final will be held during the final exam period.

Exam format

The exams are designed so that they can be completed in:

- 1 hour, for the midterm
- 3 hours, for the final exam

assuming you know the material.

The exam will have several multiple-choice questions and a few problems to solve. The assignments and tutorials should prepare you for the exam problems you will encounter. Past exams are available on the website.

Getting help

(Partial) remote learning is challenging for students, and instructors, alike
(they may have their own difficult circumstances).

Nonetheless, we will do our best to help you succeed in this course. There are four main mechanisms for getting help:

- Tutorial sessions where we will explain how to solve exam problems and you can ask questions.
- Lab sessions, where you can ask questions about the labs.
- Online forum, where you can ask questions at any time and get answers from your peers or course instructors.
- In person Prof. office hours:
  - Wed: 9am-9.30am & 3pm-3.30pm
  - Location: MC-758

What you need to do to succeed at this course

- Attend the live lecture. Even if you cannot follow along with everything (or are bored), this will force you to follow the course on a regular schedule.
- This course is different: little math, no equations, but lots of concepts that build on one another and take time to really understand.
- Attend tutorial/lab sessions and ask questions.
- Try to do all the example problems before the tutorial sessions.
- Don't cram: there is a fair amount of technical material that cannot be absorbed quickly.
- Study with your classmates! Teaching is learning; if you can explain something, then you *know* you know it.

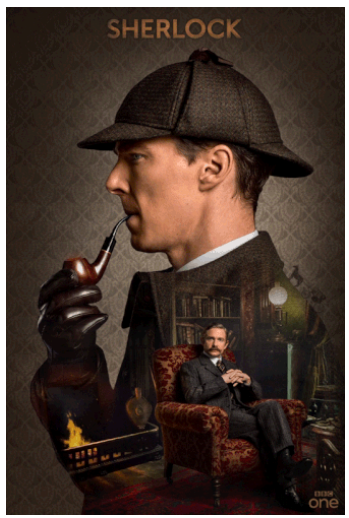
Please don’t cheat!

Cheating is a serious offense and all suspected cases will be reported to the faculty.

- Don’t share your code with others
- Don’t copy code from someone else’s or from the web

Please check carefully the Academic Integrity policies on the course webpage.

We will make use of plagiarism detection software and in addition check your submission manually!



One last word

All questions through online forum, no email!

A typical week for any Professor: hundreds of emails to deal with.

Although not encouraged, if you are unsure, you can post privately on the online discussion forum (only the TAs and myself will see it).

Conclusion

That's it! See you next time for our first lecture on Computer Technology and Abstractions.