ECSE324 : Computer Organization

Introduction

Christophe Dubach

Fall 2023

Revision history:
Warren Gross – 2017
Christophe Dubach – W2020, F2020, F2021, F2022, F2023
Brett H. Meyer – W2021, W2022, W2023

Timestamp: 2023/08/28 14:52:00

Disclaimer

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 for clarification online.

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
  - `COMP-764/ECSE-688` : High-level Synthesis of Digital Systems

What is Computer Organization?

We will learn about the *design of computers*

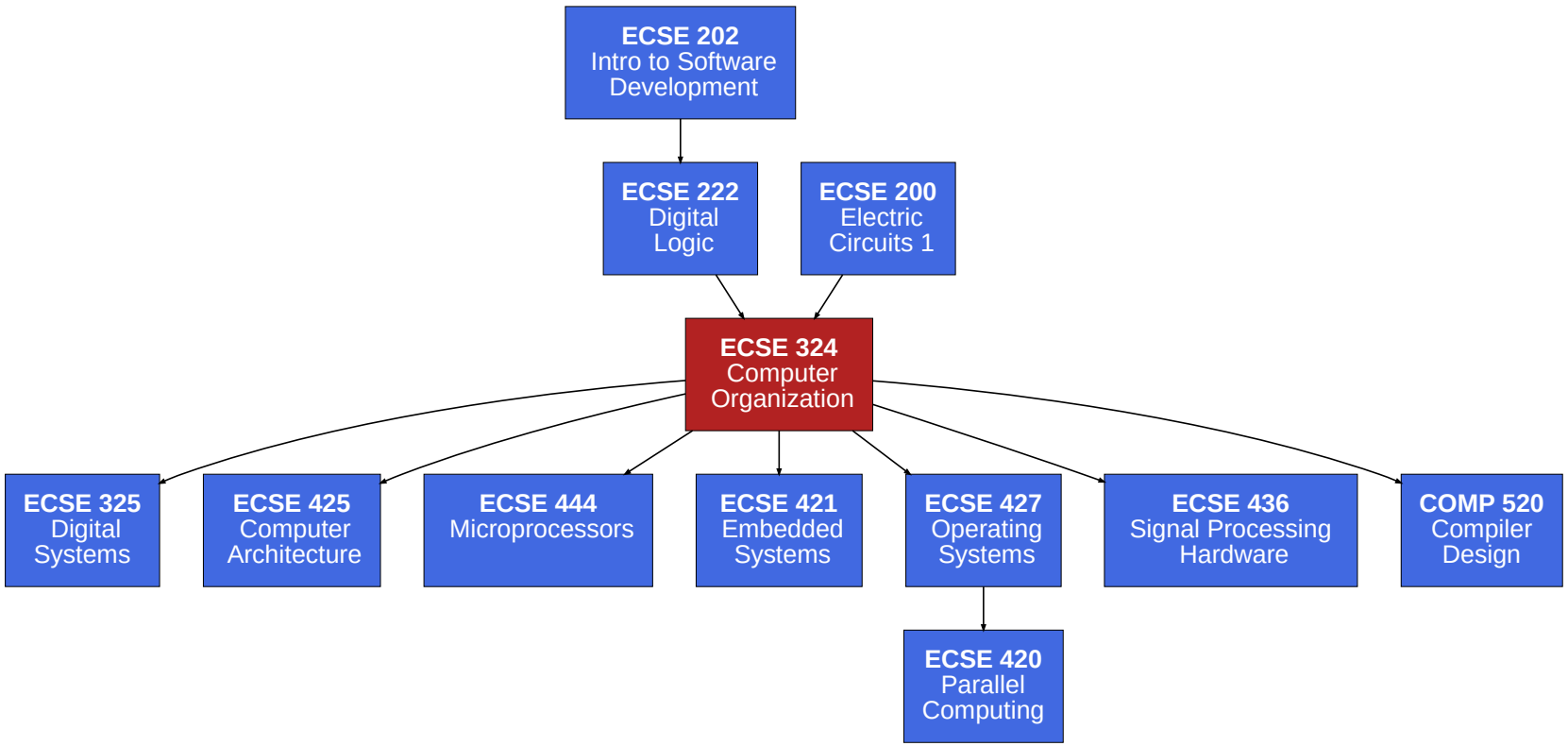
- Organization of computer components
- How software requirements influence hardware organization

In previous classes you learned about:

- How to write programs in Java (ECSE 202)
- Building blocks of digital hardware (ECSE 222)

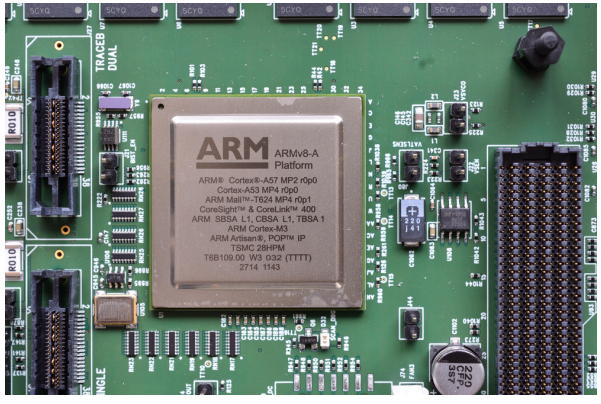
This class will fill in the missing details to tell the story of how a digital machine called a computer runs programs

ECSE 324 in context



What will you take away from ECSE 324?

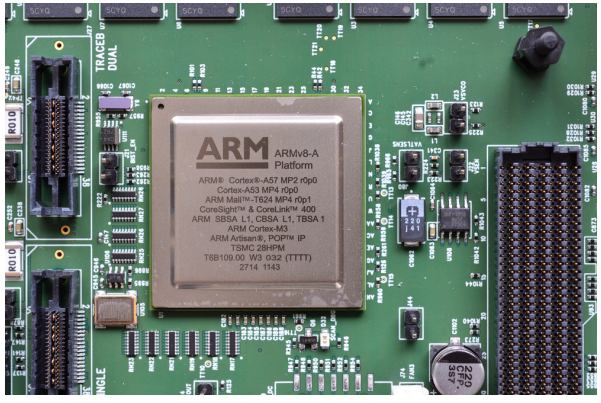
- ECSE 324 focuses on the ARM Cortex-A series of processors
  - ARM processors are used in almost every smartphone (95%)
  - Cortex-A is one of the most popular modern high-performance processor families
- Computer hardware
  - The main internal structures of computers
  - How this hardware executes 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
  - This is especially true for embedded, signal processing, and machine learning 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.

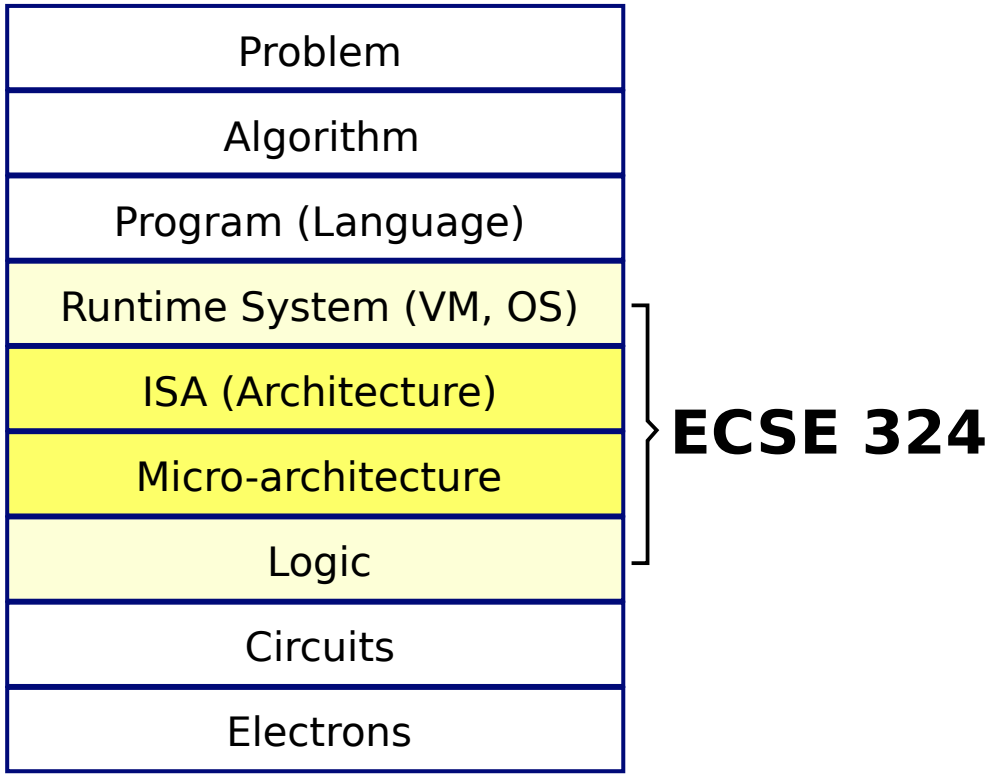
This course is therefore the foundation for efficient design of

- Software, especially for embedded systems
- Hardware, especially microprocessors

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 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
- Office hours, ...

All other course communication will occur on *ED*:

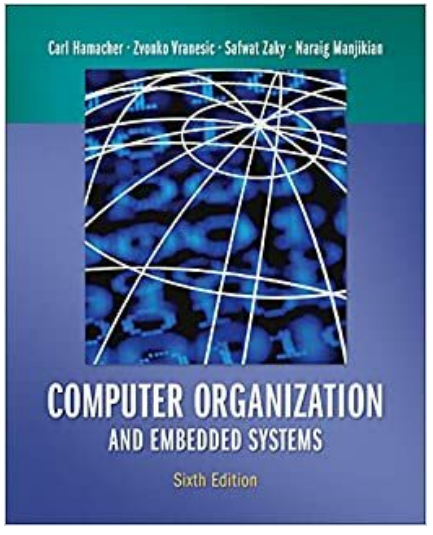
- Important announcements about the course will be posted there
  I assume you read them
- Make sure to register:
  `https://edstem.org/us/join/j2yNaK`

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.



The text is available in hard copy or digital formats:

- Le James
- VitalSource, an online academic bookstore

Course syllabus

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

1. Computer Technology and Abstractions (Chapters 1 and 2)
2. Processor Design (chapter 5)
3. Instruction Set Architecture (Appendix D)
4. Software (Chapter 4)
5. Input/Output (Chapters 3 and 7, Appendix D)
6. Memory (Chapter 8)
7. Processor Performance (Chapters 6 and 8.6)
8. Arithmetic (Chapter 9)

Course meetings

You will have the following meetings during the semester:

- 3 hours of *lecture* per week
- 2 hours of *tutorial* per week
- 2 hours of *lab* per week

All course activities are in-person.

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 submit your working design/code that will be tested.

Labs will be performed using Logisim-evolution (a digital logic simulator) and a CPUlator (a browser-based computer system emulator).

Tutorials/Assignments

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.

Tutorial assignments will not be graded.

Tutorial and lab schedule

- Tutorials and labs start next week!
- A tentative (and subject to change) course schedule is available on the course webpage.

Sessions:

- To help TAs load balance, only attend the sessions you are registered for.
- To switch lab (or tutorial) sessions, do so through the University (do not contact the instructor or TAs about this, we cannot help).

Evaluation

Item	Weight
Midterm exam	20%
Final exam	55%
Lab experiments	25%
Online participation bonus	5%

- The midterm will be held in-person on October 23, 2023.
- The final will be held in-person during the final exam period.

Exam format

The exams are in-person and closed book. They 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 well.

The exams will have:

- a number of multiple-choice, multiple-select, and other such questions;
- and a few longer problems to solve.

The assignments and tutorials should prepare you for the exam problems you will encounter.

Past exams are available on the course webpage.

Getting help with course material

Working during a pandemic 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 discussion, where you can ask questions at any time and get answers from your peers or course instructors.
- Office hours: location and time posted on the course webpage.

What you need to do to succeed at this course

This course is different: little math, no equations, but *lots of concepts* that build on one another and take time to really understand.

- *Attend the lecture*: even if you cannot follow along with everything (or are bored), this will force you to follow the course on a regular schedule.
- *Take notes*: taking notes forces you to pay attention!
- *Attend tutorial/lab sessions* and ask questions.
- *Try to do all the example problems* before the tutorial sessions.
- *Don't cram*: there is an extraordinary amount of technical material in this course that cannot be absorbed quickly.
- *Study with your classmates!* Teaching is learning; if you can explain something, then you *know* you know it.

And ask for help!

A few notes on academic integrity

You are expected to submit your own work for assessment.

- Do ask, and *answer*, questions of peers and instructors about course material.
- Do ask, and *answer*, questions of peers and instructors about lab assignments, but do not give away solutions.

We will use software to check your submissions for plagiarism.

Feel stuck? Ask for help.

Cheating degrades the value of all McGill degrees: yours, your peers', as well as alums', past and future.

Harassment- & Discrimination-free Environment

Harassment can take many forms (online or in person):

- Verbal (*e.g.*, offensive language)
- Physical (*e.g.*, gestures, unwanted physical contact)
- Visual (*e.g.*, wearing clothes with offensive language)

The teaching team strives to provide an harassment-free & discrimination-free environment for you to focus on learning and *will not tolerate any inappropriate behaviour*.

If you notice, or if you are the target, of any inappropriate behaviour, please bring it to the attention of the teaching team (online privately, or in-person, *e.g.*, during office hours) so that we can take appropriate action in response.

You can also contact the Office of Mediation and Reporting (https://www.mcgill.ca/omr/).

One last word

Ask all course-related questions through the discussion forum; please do not send email!

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

- Have a question about course material?
  Ask in the discussion forum.
- Have an administrative question (about grades, missing a lab, etc)?
  Ask in the discussion forum privately.
- Have an *x*?
  Discussion Forum.

Also, consider office hours, an *embarrassingly underutilized* resource.

Conclusion

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