Compilers and computer architecture

Preamble, and welcome!

Lecturers:

Hsi-Ming Ho

Email: hsi-ming.ho@sussex.ac.uk

Ian Mackie

Email: i.mackie@sussex.ac.uk

For office hours: please email us for an appointment or join the G5035 Discord server

Thanks to Martin Berger, Niel de Beaudrap, Chad MacKinney, Alex Jeffery, Justin Crow, Jim Fielding, Shaun Ring, and Vilem Liepelt for earlier versions / suggestions / corrections on the slides

September 2022

Administrative matters:
Lecturer & Course materials



- **Name:** Ian Mackie
  Email: `i.mackie@sussex.ac.uk`
  — Lecturer for the first half of the course



- **Name:** Hsi-Ming Ho
  Email: `hsi-ming.ho@sussex.ac.uk`
  — Lecturer for the second half of the course

- **Tutors:**
  Jak Kostrzanowski [`jk446@sussex.ac.uk`]

- **Course materials:** via Canvas
  - Slides / Panopto recordings
  - Discord server link (online chat)

Administrative matters

Lectures and labs / tutorials / ‘classes’

- Lectures: ~ 2h per week
  (Tuesday @ 16:00; Friday @ 11:00)
- Tutorials: 1h per week, various times / TAs.
  (See Sussex Direct/Canvas, for your timetables)
- PAL sessions: sign up on Canvas (Informatics PAL module).

Assessment and Coursework

Assessment: 50% coursework (due week 11)
50% unseen examination

Coursework:

- Involves writing components of a compiler
- Evaluated largely by automated testing
  - If your code doesn't work perfectly, we will look into it — but this will cost you marks
- Three separate sub-tasks:

  Syntactic analysis ; *Semantic analysis* ; *Code generation*.

  - **All** due in week 11
  - Test your code!

Examination:
Online exam, in January.

Prerequisites

- Good Java programming skills are indispensable.
  Don't be afraid to code!
  - Can you can do most questions on your own, on *e.g.*

    `https://leetcode.com`

    classified as "Easy", in less than an hour? If not:
    - better start practicing, and better attend the labs!
    - **ask lots of questions** in the tutorials / labs!
- You should be familiar with the material on "Shell Tools and Scripting" and "Command-line Environment" at:

  `https://missing.csail.mit.edu`
- Helpful to know, but covered in the lectures:
  - Regular expressions, FSMs, *etc.*
  - Registers, stack pointers, assembly

Course content

What this course is about:

- The parts of a compiler —
  broken down into modular phases.
- More superficially: computer architecture.

If you do well in this module: you will

- understand compilers and computers better,
  and be a better programmer;
- learn valuable skills in text processing
  and/or low-level memory management.

If you do *very* well in this module: you will be able to

- implement a compiler yourself — a simple one at least;
- approach the subject of programming language design.

Feedback

In this module, you will receive feedback through:

- The mark and comments on your assessment
- Comments and discussions with tutors in labs
- One to one sessions with the tutors
- Worked examples in lectures and labs
- Model solutions to exercises
- Online discussion forums
- Discussions with your peers on problems
- Feedback on assessment and exams

The more you engage (ask questions and participate in discussions), the more feedback you get.

Literature

Compilers are among the most studied and most well understood parts of informatics. Many good books exist. Here are some recommended texts (though I won't follow any of them closely).

- **Modern Compiler Implementation in Java** (2nd ed.) by Appel and Palsberg. — Probably closest to our course. Moves quite fast.
- **Compilers - Principles, Techniques and Tools** (2nd ed.) by Aho, Lam, Sethi, and Ullman.
  - The first edition is a classic (known as the "Dragon Book") but is a bit obsolete.
  - This edition is much expanded (and a tad long), and goes beyond the scope of this course.

Literature

Some other material:

- **Engineering a Compiler**, by Cooper and Torczon.
- **Stanford University online course on compilers** by Aiken.
  - Covers similar ground as ours, but goes more in-depth: influences the design of this course.
- **Computer Architecture - A Quantitative Approach** (6th ed.) by Hennessey and Patterson.
  - The 'bible' for computer architecture
  - Goes well beyond what is required for our course, but very well written by some of the world's leading experts on computer architecture.

How to enjoy and benefit from this course

- Look at the past papers.
- Use tutorials to get feedback on your solutions.
- Substantial lab exercise should bring it all together.
- Ask questions — in the labs, on Canvas, on the Discord server, or in meetings with the lecturers.
- Design your own mini-languages, and write compilers for them; or have a look at real compilers.

Questions welcome!

Please, don't wait until the end of the course to tell me about any problems you may encounter.

Ask questions:

- On the Discord server
- By email / in meetings with lecturers
  (email for an appointment):
  - **i.mackie@sussex.ac.uk**
  - **hsi-ming.ho@sussex.ac.uk**
- In the labs / tutorials
- On Canvas