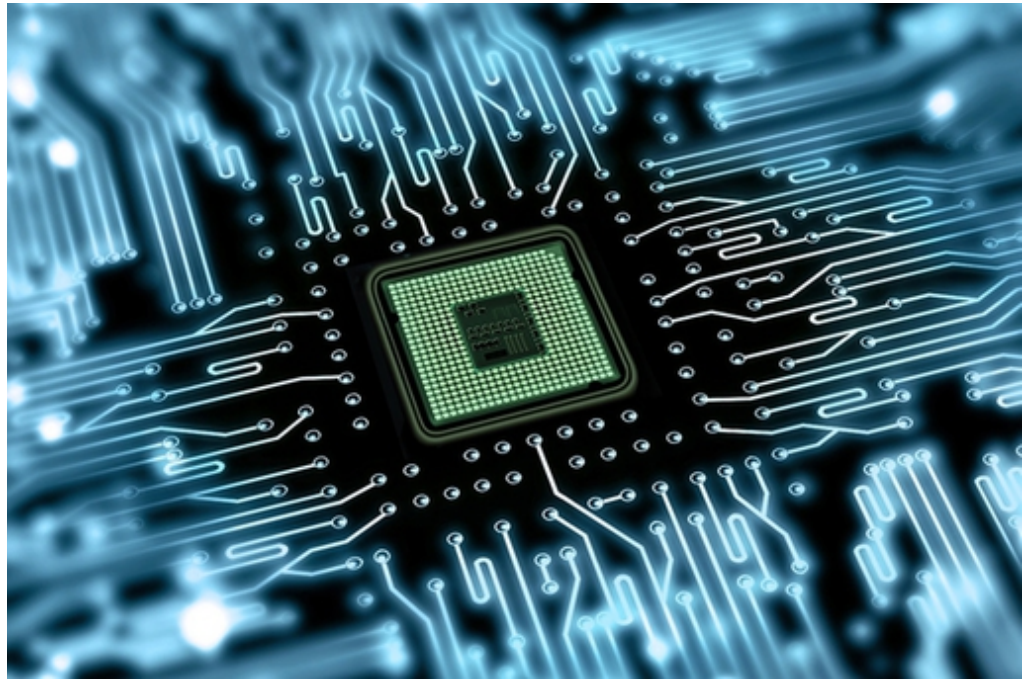
— COMP1521 20T2 —

Computer Systems Fundamentals



Convenor/Lecturer **Andrew Taylor** Course Admin **Jashank Jeremy**

COMP1521 Staff

Convenor, Lecturer	Andrew Taylor
Admin	Jashank Jeremy
Tutors	Grace Liu, Ava Williams, Mariya Shmalko, Andrew Timkov, Thinesh Manisekaran, Dylan Brotherston, Braedon Wooding, Ewan Barnett, Oliver Richards, Richard Jiang, Selina Chua, Oliver Scott, Alexandra Alexus, Shane Kadish, Richard Liu, Emily Chen, Natalie Eleftheriades, Ryan Fallah, Nicholas Sims, Clifford Sesel, Dong Huang, Zac Kologlu, Stephanie Gouw, Ryan King, Matthew Di Meglio, Michael Gribben

COMP1521 Student Background

Students in this course have completed:

- COMP1511 or COMP1911

Everyone has learned *fundamental C programming*
COMP1511 also studied *linked lists*, *ADTs*
Since not everyone has seen these, we won't use them
For this week . . .

- review/strengthen assumed C knowledge
- also research/revise (stacks, queues)

Course Goals

COMP1511/1911 ...

- gets you thinking like a *programmer*
- solving problems by developing programs
- expressing your solution in the C language

COMP1521 ...

- gets you thinking like a *systems programmer*
- with a deep understanding of run-time behaviour
- and better able to reason about your C programs

Note: these are *not* the same goals as COMP2121

COMP1511/1911 vs COMP1521

COMP1511/1911 ...

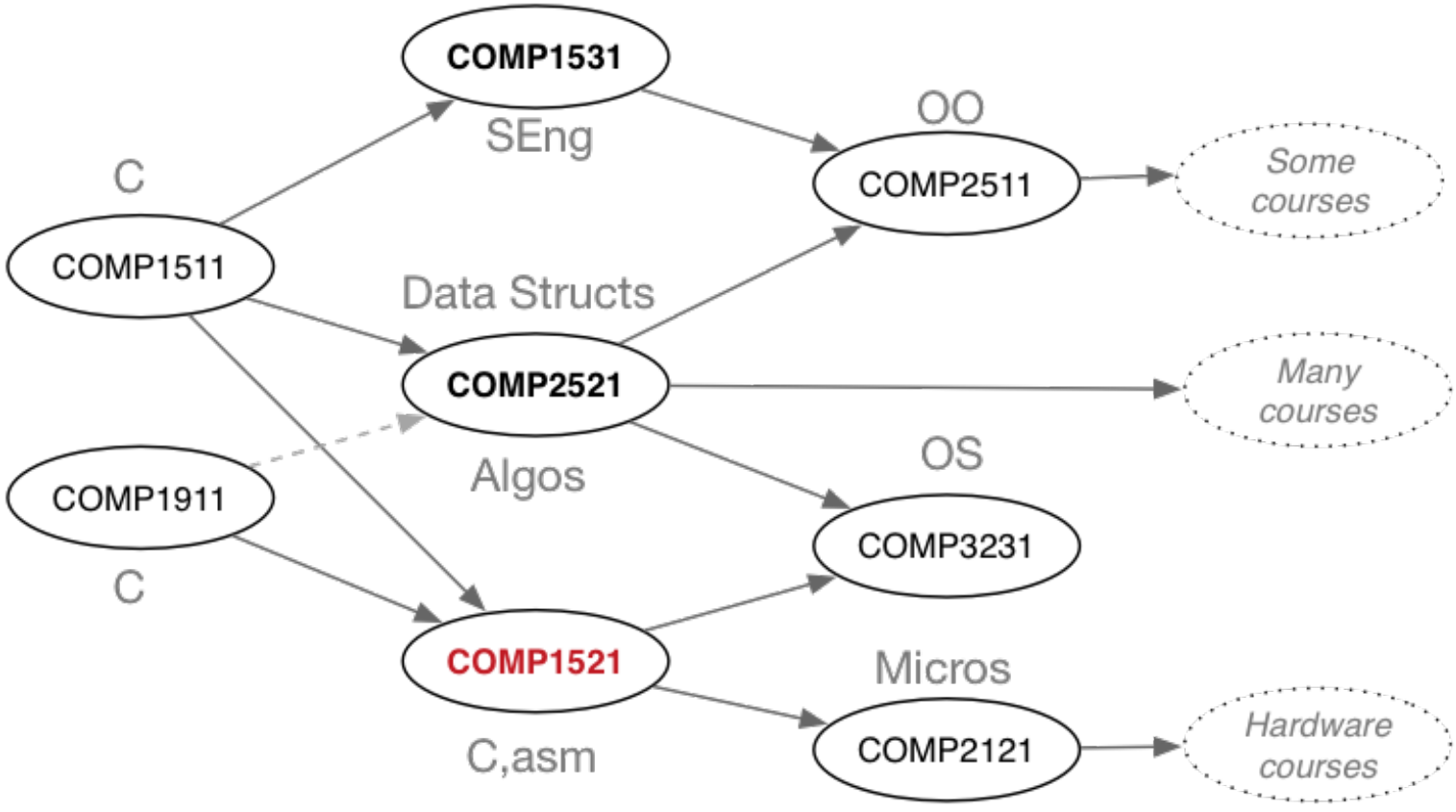
COMP1511/1911 vs COMP1521

COMP1521 . . .

COMP1511/1911 vs COMP1521

or maybe . . .

Course Context



Themes

Major themes . . .

1. software components of modern computer systems
2. how C programs execute (at the machine level)
3. how to write (MIPS) assembly language
4. Unix/Linux system-level programming
5. how operating systems and networks are structured
6. introduction to concurrency, concurrent programming

Goal: you are able to understand execution of software in detail

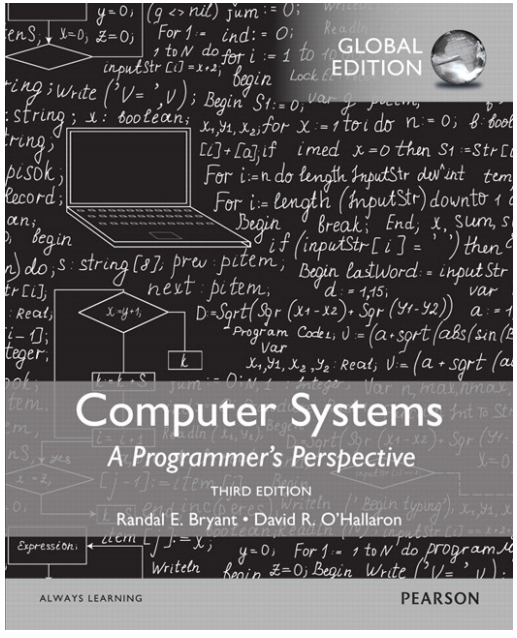
Textbook

There is no prescribed textbook for COMP1521.

Recommended reference . . .

- *Computer Systems:*
  A Programmer's Perspective,
  Bryant and O'Hallaron
- covers most topics, and quite well
- but uses a different machine code

Available in UNSW Bookshop



Textbook

There is no prescribed textbook for COMP1521.
Material has been drawn from.

- "Introduction to Computing Systems:
  from bits and gates to C and beyond",
  Patt and Patel
- "The Elements of Computer Systems:
  Building a modern computer system from first principles",
  Nisan and Schocken
- COMP2121 Course Web Site, Parameswaran and Guo

Note: always give credit to your sources

Systems and Tools

Prac work based on *Linux* tools

- all tools available on the *CSE lab machines*
- can use *VLAB* to connect to CSE from home

Compilers: `dcc` on CSE machines (clang or gcc elsewhere)
Assembly language: MIPS on **QtSpim** (also **Xspim** on CSE)
Use your own favourite text editor
Other tools: `make`, `gdb`, `man`, `bc -l`
Learn to love the *shell* and command-line ... very useful

Lectures

- Tuesday, 09:00—11:00; Thursday, 16:00—18:00; delivered via Microsoft Teams Live Events
  - you will have email about how to access the event
  - feel free to ask questions via chat
  - lectures recorded and linked from course home page.
- present a brief overview of theory
- focus on practical demonstrations of coding
- demonstrate problem-solving (testing, debugging)
- Lecture slides available on the web before lecture.

Tutorials

- Tutorials start in week 2.
- Tutorials & labs online, via Blackboard Collaborate
  - you will have email about how to access Collaborate
- tutes clarify lecture material
- work through problems related to lecture topics
- give practice with design *(think before coding)*
- answers available on the web after the week's last tutorial.

To get the best out of tutorials ...

- attempt the problems yourself beforehand
- ask if you don't understand a question or how to solve it
- Do *not* keep quiet in tutorials ... talk, discuss, ...
- Your tutor may ask for your attempt to start a discussion.

Lab Classes

Each tutorial is followed by a two-hour lab class.

- Several exercises, mostly small coding tasks
- Build skills needed for assignments, exam
- Done individually
- Submitted via `give`, before Sunday 18:00
- Automarked (with partial marks) — 15% of final mark
- Labs may include challenge exercises:
  - may be silly, confusing, or impossibly difficult
  - full marks possible without completing any challenge exercises

Weekly Tests

From week 3, weekly tests:

- programming tests
- immediate reality-check on your progress.
- done in your own time under self-enforced exam conditions.
- Time limit of 1 hour
- Automarked (with partial marks) — 10% of final mark
- best 6 of 8 tests used to calculate the 10%
- any violation of test conditions $\Rightarrow$ zero for whole component

Assignments

- Ass1: Assembly Language, weeks 4–7, 15%
- Ass2: C Programming, weeks 7–10, 15%
- Assignments give you experience with larger programming problems than the lab exercises
- Assignments will be carried out individually.
- They *always* take longer than you expect.
- Don't leave them to the last minute.
- There are late penalties applied to maximum assignment marks, typically 2%/hour

Code of Conduct

CSE offers an inclusive learning environment for all students.
In anything connected to UNSW, including social media,
these things are student misconduct and will not be tolerated:

- racist/sexist/offensive language or images
- sexually inappropriate behaviour
- bullying, harassing or aggressive behaviour
- invasion of privacy

Show respect to your fellow students and the course staff

Plagiarism

What is plagiarism?

Presenting the (thoughts or) work of another as your own.

Cheating of any kind constitutes academic misconduct and carries a range of penalties. Please read course intro for details.

Examples of inappropriate conduct:

- groupwork on individual assignments (discussion OK)
- allowing another student to copy your work
- getting your hacker cousin to code for you
- purchasing a solution to the assignment

Plagiarism

What is plagiarism?

Presenting the (thoughts or) work of another as your own.

Cheating of any kind constitutes academic misconduct and carries a range of penalties. Please read course intro for details.

Examples of inappropriate conduct:

- groupwork on individual assignments (discussion OK)
- allowing another student to copy your work
- getting your hacker cousin to code for you
- purchasing a solution to the assignment

Remember

You are only cheating yourself and chances are you will get caught!

Plagiarism

- Labs, tests, assignments must be entirely your own work.
- You can not work on assignment as a pair (or group).
- Plagiarism will be checked for and *penalized*.
- Plagiarism may result in suspension from UNSW.
- Scholarship students may lose scholarship.
- International students may lose visa.
- Supplying your work to any another person may result in loss of all your marks for the lab/assignment.

Final Exam

- online practical exam (you complete from home)
- limited on-line language documentation available
- some multiple-choice/short-answer questions, similar to tut questions.
- some questions will ask you to read C or assembler
- most marks for questions which ask you to write C or assembler
- also may ask you to answer written questions
- you must score 18+/45 on the final exam to pass course

Assessment

- 15% Labs
- 10% Weekly Programming Tests
- 15% Assignment 1 — due week 7
- 15% Assignment 2 — due week 10
- 45% Final Exam

Above marks may be scaled to ensure an appropriate distribution

To pass you must:

- **score 50/100 overall**
- **score 18/45 on final exam**

For example:
55/100 overall, 17/45 on final exam $\Rightarrow$ **55 UF** not 55 PS

How to Pass this Course

- coding is a *skill* that improves with practice
- the more you practise, the easier you will find assignments/exams
- do the lab exercises
- do the assignments *yourself*
- practise programming outside classes
- treat extra tutorial questions like a mini prac exam

Assumed Knowledge

Assumed knowledge:

- design an algorithmic solution
- describe your solution in C code, using ...
  - variables, assignment, tests (==, !, <=, &&, etc)
  - `if`, `while`, `for`, `break`, `scanf()`, `printf()`
  - functions, `return`, prototypes, `*.h`, `*.c`
  - arrays, structs, pointers, `malloc()`, `free()`

Not assumed knowledge:

- linked structures, file operations, ADTs, sorting, *recursion*, *bit operations*