# Course Outline

CP1521

Computer Systems Fundamentals

Diploma Program

UNSW Global Education

Term 2 2020

## 1. Staff

| Position | Name | Email |
|---|---|---|
| Course Convenor & Lecturer | Dr Angela Finlayson | A.Finlayson@unswglobal.unsw.edu.au |

## 2. Course information

| | |
|---|---|
| Units of credit (UOC): | 6 |
| Pre-requisite(s): | CP1511 |
| Total course contact hours: | 96 |

### 2.1 Course summary

This course provides a programmer's view on how a computer system executes programs, manipulates data and communicates. It enables students to become effective programmers in dealing with issues of performance, portability, and robustness.

The course assumes that students have completed the first course in programming in the C programming language, CP1511.

### 2.2 Course aims

This course aims to give students an overview of the structure and behaviour of modern computer systems, and to provide a foundation for later courses on networks, operating systems, computer architecture and compilers, where a deeper understanding of systems-level issues is required.

### 2.3 Course learning outcomes (CLO)

At the successful completion of this course you (the student) should be able to:

**1** Describe the layers of architectures in modern computer systems from hardware device levels upwards

**2** Describe the principles of memory management and explain the workings of a system with virtual memory management

**3** Explain how the major components of a CPU work together, including how data (including instructions) is represented in a computer

**4** Design, implement and analyse small programs at the assembly/machine level

**5** Describe the relationship between high-level procedural languages (e.g., C) and assembly/machine language in the conventional machine layer, including how a compiled program is executed in a classical von Neumann machine

**6** Explain how input/output operations are implemented, and describe some basic I/O devices

**7** Describe the components comprising and the services offered by an operating system

**8** Describe the layered structure of standard network architecture

**9** Implement simple programs involving communication and concurrency

## 2.4 Relationship between course and program learning outcomes and assessments

| Course Learning Outcome (CLO) | **Program Learning Outcome (PLO)** | **Related Tasks & Assessment** |
|---|---|---|
| CLO 1 | Understanding of underpinnings (EA1.1) | Exam, Quizzes |
| CLO 2 | Understanding of specialist bodies of engineering knowledge (EA1.3) | Exam, Quizzes |
| CLO 3 | Understanding of specialist bodies of engineering knowledge (EA1.3) | Exam, Quizzes, Labs |
| CLO 4 | Application of established engineering practice (EA2.1) | Labs, Assignments |
| CLO 5 | Conceptual understanding of computer underpinnings (EA1.2) | Exam, Labs |
| CLO 6 | Understanding of specialist bodies of engineering knowledge (EA1.3) | Exam, Quizzes, Labs |
| CLO 7 | Understanding of specialist bodies of engineering knowledge (EA1.3) | Exam, Quizzes |
| CLO 8 | Application of established engineering practice (EA2.1) | Labs |

|  |  |  |
|---|---|---|
| CLO 9 | Application of established engineering practice (EA2.1) | Exam, Labs |

# 3. Strategies and approaches to learning

## 3.1 Learning and teaching activities

This course involves a number of teaching activities:

**Lectures – 4 hours per week**

Lectures present theory and concepts, by way of case studies and practical examples. Lecture notes will be provided in advance of each class. There will be 2 hours of timetabled live streamed lectures each week and 2 hours of lecture videos each week that students must watch in their own time.

**Tutorials – 2 hours per week**

Tutorials allow students to collaboratively work through example problems to illustrate lecture idea, and have concepts from lectures clarified by the tutor. Each student will lead a "code review" (discussion of a piece of software or a system) at least once during the course.

There is up to 1 bonus mark available for students who give an outstanding code review.

**Lab Classes – 2 hours per week**

Lab Classes involve small exercises where students build systems that illustrate the ideas covered in lectures. Students write software, and show their work to the Lab demonstrator for assessment and feedback.

To obtain a mark for a lab exercise you should demonstrate the completed lab exercise to your tutor during a lab class and submit it using give.

You should normally get your lab work assessed during the week for which it is scheduled (i.e. you must complete the week 3 lab exercise during the week 3 lab). If you don't finish it during the lab, you may continue working on it during the week, but you both must submit it (using give) by the following Tuesday 11:59pm in order to get any marks for it. You must then also demonstrate your work to your tutor during the first hour of the following week's lab. The code that you submit by Tuesday 11:59pm is what will be assessed.

Summary: to obtain any lab marks for the Week X lab, you must do 2 things:

**1** submit your lab work by the following Tuesday 11:59pm

**2** demonstrate your work to your tutor in the week X lab class
OR demonstrate your work at the start of the lab in week X+1

You cannot obtain marks by e-mailing lab work to tutors.

One of the labs will be used for a Practice Prac Exam, where you will use the exam environment to individually solve a small(ish) programming tasks (one MIPS and one C). You must complete the Practice Prac Exam in the lab in the scheduled week. The Practice Prac Exam will be worth twice the marks of a standard lab.

Lab exercises will be assessed using the following grade system:

| **Grade** | **Criteria** |
|---|---|
| A+ | Outstanding effort; must complete any challenges and go beyond the standard exercises |
| A | Complete, correct, and clear solution to standard lab exercises (worth full marks) |
| B | Most of the standard lab exercises completed, or all completed but with one or more major bugs |
| C | Partial solution only, much of lab not completed or has many glaring errors |
| D | Submitted something , but it's completely hopeless |
| . | Not attempted |

Optional challenge exercises may be specified for some labs.

There will be more lab marks available than necessary to obtain full marks for the 10% lab component. In other words, the total lab mark will be capped, with a small bonus available for consistently outstanding work.

## Assignments – 2 during the course

Assignments are take-home problems that are larger in scope than Lab exercises and require students to use creativity to solve a challenging realistic problem. Each assignment requires students to understand the problem, design a solution, and implement and test their solution.

## Online Quizzes – 5 during the course

Online quizzes encourage students to revise ideas from lectures. There are 5 quizzes and each quiz is worth 2 marks.

### Online Forum

An online forum allows students to ask and answer questions on the tutorial, lab and assignment exercises, and on lecture material.

### Final Exam

There will be a three-hour exam.

It will contain a mixture of: implementation tasks (which will require you to write C and assembler programs); "theory" questions (which require analysis and written answers); multiple choice questions. During this exam you will be able to execute, debug and test your answers. The implementation tasks will be similar to those encountered in lab exercises.

There is a hurdle requirement on the final exam. If you do not score at least 40%  on the exam, you cannot pass this course. If your overall course score exceeds 50%, despite scoring very poorly (<40%) on the exam, the hurdle will be enforced via a grade of UF. Of course, if your overall course score is less than 50%, then your grade will be FL.

## 3.2 Expectations of students

Students are expected to:

- attend all lectures, and ask questions, but otherwise not disturb other students
- attend all tutorials and actively participate in the discussions
- attend all lab classes and work diligently on the exercises
- do all of the assignment work themselves, asking only the forum or tutors for help

On the course forum, students should:

- use relevant/meaningful message titles on all posts
- ask questions clearly and provide sufficient background information that the question can be reasonably answered
- not post significant pieces of code, especially code for assignment

# 4. Course schedule and structure

This course consists of 8 hours of class contact hours per week. You are expected to take an additional 5 hours outside classes to complete assessments, readings, and exam preparation.

| **Week** | **Lectures** | **Tutorial and Labs** | **Assessment** | Related CLO |
|---|---|---|---|---|
| Week 1 | Course Introduction, Debugging and Software Tools | Welcome, C Revision, Input/output, man, bc | | 1 |
| Week 2 | Data Representation | Number Systems, Priority Queues, Big Numbers, gdb | | 3, 4 |
| Week 3 | Data Representation and Instruction set architecture | Bit-manipulation, Memory, Data Representation, Unions, Floats | Quiz 1 | 3, 4 |
| Week 4 | Assembly language Programming | Assembly Language and debugging | | 4 |
| Week 5 | Assembly Language Programming | Assembly Language | Assignment 1 Released | 4 |
| Week 6 | Computer Systems Architecture, Operating Systems, Systems Calles, File Systems | Assembly Language | Quiz 2 | 4 |
| Week 7 | Devices, Virtual Memory | Operating Systems and the Unix File System | | 2,8 |
| Week 8 | Processes and Signals | Devices, Virtual Memory | Quiz 3 | 2,6 |

| | | | | |
|---|---|---|---|---|
| Week 9 | Revision, Practice Practical Exam Preparation | Processes and Signals | Assignment 1 Due<br>Assignment 2 Released | 8 |
| Week 10 | Network Architecture | Practice Practical Exam Revision and Practice Practical Exam | Quiz 4 | 7 |
| Week 11 | Parallelism, concurrency, synchronisation, coordination, communication | Networks and network programming | | 5,7 |
| Week 12 | Exam Preparation | Concurrency and Inter-Process Communication and Exam Preparation | Quiz 5<br>Assignment 2 due | 5,7 |

# 5. Assessment

## 5.1 Assessment tasks

| Assessment task | Length | Weight | Due | CLOs |
|---|---|---|---|---|
| Quizzes | Throughout semester | 10% | Weeks 3,6,8,10,12 | 1-9 |
| Assignment 1 (assembly language) | 4 weeks | 9% | Week 9 | 4 |
| Assignment 2 (C programming) | 4 weeks | 11% | Week 12 | 2,5 |
| Labs | Throughout Semester | 10% | Weekly | 1-7,9 |
| Final Exam | 3 Hours | 60% | Exam period | 1-9 |

Programming assignments are marked primarily on their correctness with respect to the assignment specification. Test cases will be provided for each assignment, and further (unseen) test cases will be used for marking. A small component of the mark will be for code quality.

Lab exercises are similarly marked primarily on their correctness, but Lab demonstrators will also give feedback on code quality.

Online quizzes are multiple choice.

### Final Mark

Your final mark for this course will be computed using the above assessments as follows:

| | | | |
|---|---|---|---|
| CourseWorkMark | = | QuizMark + LabMark + Ass1Mark + Ass2Mark | out of 40 |
| ExamPracMark | = | marks for prac questions on final exam | out of 30 |
| ExamTheoryMark | = | marks for written questions on final exam | out of 30 |
| ExamMark | = | ExamPracMark + ExamTheoryMark | out of 60 |
| ExamOK | = | ExamMark ≥ 24/60 | true/false |
| FinalMark | = | CourseWorkMark + ExamMark | out of 100 |
| FinalGrade | = | UF, if ! ExamOK && FinalMark ≥ 50<br>FL, if FinalMark < 50/100<br>PS, if 50/100 ≤ FinalMark < 65/100<br>CR, if 65/100 ≤ FinalMark < 75/100<br>DN, if 75/100 ≤ FinalMark < 85/100<br>HD, if FinalMark ≥ 85/100 | |

## 5.2 Assessment criteria and standards

In all programming work, the primary assessment criterion is correctness (i.e. does the code produce the expected output/behaviour according to the exercise specification). This will be tested by executing code against a variety of test cases, some of which are available to students, and others of which are used after submission for assessment purposes. Code is also expected to be expressed clearly, with consistent formatting and using relevant variable names.

## 5.3 Submission of assessment tasks

All assignments will be submitted online via CSE’s submission system. Late penalties accrue on an hourly basis. Marks are capped according to how late the submission is, but it would typically be the case that marks are capped at 50% after 36 hours.

If you are unable to submit an assignment by the due date, due to medical reasons or other reasons which significantly affect your ability to carry out your work, you should contact the lecturer as soon as possible, preferably well before the assignment deadline. If the lecturer considers that your ability to complete the assignment on time has been adversely affected, an extension may be granted to make up for the time you were unable to work on the assignment.

Lab exercises must be submitted by the end of the following Tuesday after the lab class. Demonstrators will then look at the exercise, and assess it, possibly asking you to explain what you did in the following lab class. Failure to complete and submit a lab exercise results in a mark of zero for that lab.

Quizzes must be completed by the deadline shown on the quiz. No extensions are possible for quizzes.

## 5.4. Feedback on assessment

Assignments will be marked after the submission deadline and annotated with comments by the tutor. You can discuss the tutor’s comments in a lab class after you have received the feedback.

Lab demonstrators will discuss your lab submission with you during the lab class in the week following the submission.

# 6. Readings and resources

There is no single text book that covers all of the material in this course at the right level of detail and using the same technology base as we are. The lecture notes should provide sufficient detail to introduce topics, and you will then study them in further depth in the tutes, labs and assignments.

There are also many online resources available, and we will provide links to the most useful ones. Some are listed below. If you find others, please post links in the Comments section on the Course Outline page.

The following is a Recommended Reading for this course

*Computer Systems: A Programmer's Perspective* , by Randal E. Bryant and David R. O'Hallaron, Prentice-Hall ( web site )

There are copies in the UNSW Bookstore and in the library. It covers many of the topics in the course, but uses a different machine language (i.e. not MIPS).

Some suggestions for other books that cover at least some of the topics in this course

- *Introduction to Computer Systems: From Bits and Gates to C and Beyond* , by Yale N. Patt and Sanjay J. Patel, McGraw Hill
- *nand2tetris: The Elements of Computing Systems: Building a Modern Computer from First Principles* , by Noam Nisan and Shimon Schocken, MIT Press ( web site , including lecture slides)

Documentation for the various systems used in the course is linked from the course website.