Course Outline

Contents

- Course Details
- Course Summary
  - Assumed Knowledge
  - Learning Outcomes
- Teaching Strategies and Rationale
  - Lectures
  - Tutorials
  - Laboratory Classes
  - Assignments
  - Weekly Tests
- Student Conduct and Academic Integrity
  - Student Conduct
  - Academic Integrity
- Assessment
- Course Schedule
- Resources for Students
- Course Evaluation and Development

Course Details

Course Code	COMP1521
Course Title	Computer Systems Fundamentals
Units of Credit	6
Course Contact	`<cs1521 at cse.unsw.edu.au>`
Convenor/Lecturer	Dr Andrew Taylor `<andrewt at unsw.edu.au>`
Admin	Jashank Jeremy `<jashank.jeremy at unsw.edu.au>`
Classes	**Lectures**: Tue 09-11, on-line Thu 16-18; ... timetable for all classes
Course Website	https://cgi.cse.unsw.edu.au/~cs1521/20T2/
Handbook Entry	https://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP1521/

Course Summary

This course introduces students to how computer systems are structured in terms of basic electronic components, how they are used to implement procedural programs, and how they are structured as a collection of software layers. It introduces students to low-level software layers such as operating systems and network infrastructure, and introduces concurrency concepts. The goal is to give students a solid understanding of what happens when high-level programs are executed, as a basis for further study in important areas of computing such as computer architecture, operating systems, and networks.

Assumed Knowledge

Before commencing this course, students should be able to ...

- write simple programs in the C programming language
- define and invoke functions and return results in C
- define and manipulate structured data in C
- use pointers to access data objects

These are assumed to have been acquired in COMP1511 or COMP1911.

Learning Outcomes

After completing this course, students will be able to ...

1. Describe the architectural layers (fundamental parts) of a modern computer systems from hardware device (chip) 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 and instructions are represented in a computer
4. Design, implement, and analyse small programs at the assembly/machine level
5. Describe the relationship between a high-level procedural language (C) and assembly (machine language) which implements it 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. Implement simple programs involving communication and concurrency

This course contributes to the development of the following graduate capabilities:

Graduate Capability	Acquired in
scholarship: understanding of their discipline in its interdisciplinary context	lectures, assignments
scholarship: capable of independent and collaborative enquiry	lab work, assignments
scholarship: rigorous in their analysis, critique, and reflection	tutorials
scholarship: able to apply their knowledge and skills to solving problems	tutorials, lab work, assignments
scholarship: ethical practitioners	all course-work, by doing it yourself
scholarship: capable of effective communication	blog, tutorials
scholarship: digitally literate	everywhere in CSE
leadership: enterprising, innovative and creative	assignments
leadership: collaborative team workers	lab work, assignments
professionalism: capable of operating within an agreed Code of Practice	all prac work

Teaching Strategies and Rationale

This course uses the standard set of practice-focussed teaching strategies employed by most CSE foundational courses:

- Lectures ... introduce concepts, show examples
- Tutorials ... reinforce concepts and provide additional examples
- Lab Work ... provide examples of using various technologies
- Assignments ... allow you to solve larger problems

Having said that, the second half of the course is more discursive than other CSE foundational courses.

This course is taught the way it is because it aims to give a broad view of many topics in computer systems, to provide a foundation for further study in later systems-related courses. At the same time, it provides further practice in developing software, but at a level closer to the machine than other foundational courses.

Lectures

Lectures will be used to present the theory and practice of the techniques and tools in this course. The lectures will include practical demonstrations of various technologies. Lecture notes will be available on the course web pages before each lecture.

We plan to use broadcast lectures using Teams live events with the opportunity to ask questions via chat. Recording of all lectures will be made available. Lectures may be pre-recorded on some topics.

Tutorials

We plan to use Blackboard Collaborate for tutorials and labs. Your will be emailed links.

From week 1, you will also be expected to participate a one-hour tutorial session to clarify ideas from lectures and work through exercises based on the lecture material. You should make sure that you use them effectively by examining in advance the material to be covered in each week's tutorial, by asking questions, by offering suggestions, and by generally participating. The tutorial questions will be posted on the Web in the week before each tutorial. There are no marks for tutorial participation.

Laboratory Classes

Following the tutorial class each week, there will be a two-hour laboratory class, during which you will work on a variety of small practical problems involving the tools introduced in lectures. Because this course has a significant practical component, laboratory classes are **important**. If you do not put a good amount of effort into the lab classes, you risk failing the final exam.

Each week, there will be one or more exercises to work on. These exercises will be released in the week preceding the lab class.

During the lab, your tutor will provide feedback on your approach to the problem and on the style of your solution. Some labs may contain exercises which will be assessed during the lab.

Completed exercises need to be submitted. You must submit exercises before the deadline using **give** to obtain a mark for a lab exercise, The usual lab exercise submission deadline will be 17:59 Sunday some lab exercises may have an extended deadline

Starting in week 2, pairs will also be asked to do code reviews in the tutorials, to explain how they tackled a particular problem, and describe interesting features of their solution.

Challenge exercises may be specified for some labs. These will be individual (not pair) exercises. Challenge exercises will never total more than 15% of each week's lab mark. challenge exercises are done individually.

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

The lab exercises for each week are worth in total 2.3 marks. All of your lab marks will be summed to give you a mark out of 18.4; if their sum exceeds 15, your total mark will be capped at 15. As a result, you can obtain full marks for the labs without completing challenge exercises; and you can miss one lab without affecting your mark.

Assignments

There are two assessable programming assignments. Assignments give you the chance to practice what you have learnt on relatively large problems (compared to the small exercises in the labs). Assignments are a *very important* part of this course, therefore it is essential that you attempt them yourself.

- Assignment 1, on Assembly programming; due Week 6; worth 15%
- Assignment 2, on C programming; due Week 10; worth 15%

Late assignments submissions will be penalised. The exact penalty will be specified in the assignment specification: often, it is a 1% reduction in the maximum achievable mark for every hour late.

Weekly Tests

There will be weekly tests from weeks 3–-10 designed to give you timely and realistic feedback of your understanding of the course material. Tests may be programming exercises, multiple choice questions, or both.

These will be conducted in your own time under self-enforced exam-like conditions. Each test will specify the conditions, but typically these will include:

1. no assistance permitted from any person;
2. a time limit;
3. no access to materials (written or online) except specified language documentation or man pages.

Each test is worth 1.7 marks, and will be automarked. Your total mark for the tests component is computed as a sum of your best 6 of 8 test marks. Any violation of the test conditions will result in a mark of zero for the entire test component.

Final Exam

There will be an online exam which students completely remotely (from home). This will be centrally timetabled, and appear in your UNSW exam timetable.

It will contain a mixture of: implementation tasks (which will require you to write C and assembler programs), and “theory” questions (which require analysis and written answers). 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% (18.0/45) on the exam (after scaling), 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.

Student Conduct and Academic Integrity

Student Conduct

The **Student Code of Conduct** (Information, Policy) sets out what the University expects from students as members of the UNSW community. As well as the learning, teaching and research environment, the University aims to provide an environment that enables students to achieve their full potential and to provide an experience consistent with the University's values and guiding principles. A condition of enrolment is that students *inform themselves* of the University's rules and policies affecting them, and conduct themselves accordingly.

Students have the responsibility to observe standards of equity and respect in dealing with every member of the University community. This applies to all activities on UNSW premises and all external activities related to study and research. This includes behaviour in person as well as behaviour on social media: for example, in Facebook groups set up for the purpose of discussing UNSW courses or course work. Behaviour that is considered in breach of the Student Code Policy as discriminatory, sexually inappropriate, bullying, harassing, invading another's privacy, or causing any person to fear for their personal safety is serious misconduct, and can lead to severe penalties, including suspension or exclusion.

If you have any concerns, you may raise them with your lecturer, or approach the School Ethics Officer, Grievance Officer, or one of the student representatives.

Academic Integrity

UNSW has an ongoing commitment to fostering a culture of learning informed by academic integrity. All UNSW staff and students have a responsibility to adhere to this principle of academic integrity.

Plagiarism is defined as using the words or ideas of others and presenting them as your own. Plagiarism undermines academic integrity, and is not tolerated at UNSW. Instances of plagiarism are treated by UNSW and CSE as acts of academic misconduct, which carry penalties as severe as being excluded from further study at UNSW. There are several on-line resources to help you understand what plagiarism is and how it is dealt with at UNSW.

- Plagiarism and Academic Integrity
- UNSW Plagiarism Procedure

Make sure that you read and understand these. Ignorance is not accepted as an excuse for plagiarism. In particular, at CSE you are responsible for ensuring that your assignment files are not accessible by anyone but you by setting correct permissions in your CSE home directory and for any code repositories you may use. Note also that plagiarism includes paying or asking another person to do a piece of work for you, and then submitting it as your own work.

The pages below describe the policies and procedures in more detail:

- Student Code Policy
- Student Misconduct Procedure
- Plagiarism Policy Statement
- Plagiarism Procedure

You should also read the following page which describes your rights and responsibilities in the CSE context:

- Essential Advice for CSE Students

Assessment

Item	Topics	Due	Marks	LOs
Tests	all topics	Weeks 3-10	10	1-9
Assignment 1	Assembly programming	Week 6	15	4
Assignment 2	C programming	Week 10	15	2,5
Labs	most topics	most weeks	15	1-5,9
Final Exam	all topics	exam period	45	1-9

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

CourseWorkMark	=	TestMark + LabMark + Ass1Mark + Ass2Mark	out of 55
ExamMark			out of 45
ExamOK	=	ExamMark ≥ 18.0/45	true/false
FinalMark	=	CourseWorkMark + ExamMark	out of 100
FinalGrade	=	UF, if ! ExamOK && FinalMark ≥ 50 FL, if FinalMark < 50/100 PS, if 50/100 ≤ FinalMark < 65/100 CR, if 65/100 ≤ FinalMark < 75/100 DN, if 75/100 ≤ FinalMark < 85/100 HD, if FinalMark ≥ 85/100	

Course Schedule

The following is a rough schedule of when topics will be covered. This will most likely change over the session as topics take more or less time to cover.

Week	Lectures	Tutes	Labs	Assigns	Tests
1	Course intro, computer systems, memory, data representation	Welcome, C revision, Bits	Input/output	-	-
2	Data representation (cont.)	Data representation (i)	Bit Manipulation	-	-
3	Instruction set architecture, assembly language programming	Data representation (ii)	Data Representation (floats)	-	Test 1
4	Assembly language programming (cont); Compilation, mapping C to assembler	Assembly language	Assembly programming (i)	-	Test 2
5	Computer systems architecture, layers, operating systems, system calls	Mapping C to MIPS assembler	Assembly programming (ii)	-	Test 3 (due week 7)
6	flexibility week	-	-	Ass1 due	Test 4

Week	Lectures	Tutes	Labs	Assigns	Tests
7	File systems; memory management	OSs; system calls	System calls	-	Test 5
8	Virtual memory; processes; interrupts; i/o	File system functions	File system functions	-	Test 6
9	Parallelism, synchronisation, coordination, communication	Virtual memory; processes	Signals	-	Test 7
10	Revision	Parallelism; concurrency	Practice Prac Exam	Ass2 due	Test 8

Resources for Students

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 architecture (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
- *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.

Course Evaluation and Development

This course has previously run in 17s2, 18s1, 18s2, 19T2 and 19T3.

We have read carefully the myExperience feedback from 19T3 and in response have made changes to the delivery:

The course will be evaluated at the end of the session using the myExperience system.

However, you are encouraged to provide informal feedback during the session, and to let course staff know of any problems as soon as they arise. Suggestions will be listened to openly, positively, constructively, and thankfully, and every reasonable effort will be made to address them.