Course Outline

Public

Edit Resource (/COMP1521/19T2/resources/25621/edit)

Contents

- Course Details
- Course Summary
- Assumed Knowledge
- Learning Outcomes
- Teaching Strategies and Rationale
- Student Conduct
- Assessment
- Course Schedule
- Resources for Students
- Course Evaluation and Development

Course Details

Course Code	COMP1521
Course Title	Computer Systems Fundamentals
Units of Credit	6
Convenor	Prof Salil Kanhere (http://www.salilkanhere.net) <salil.kanhere at unsw.edu.au>
Lecturer(s)	Prof Salil Kanhere (http://www.salilkanhere.net) <salil.kanhere at unsw.edu.au> Dr Zainab Abaid (https://webcms3.cse.unsw.edu.au/users/z3491562) <z.abaid at unsw.edu.au>
Admin	Jashank Jeremy (https://webcms3.cse.unsw.edu.au/users/jashankj) <jashank.jeremy at unsw.edu.au>
Classes	**Lectures** : Stream 1 (1UGA): Mon 14-16 Burrows Theatre, Thu 11-13 CLB 7 Stream 2 (1UGB): Wed 16-18 CLB 7, Fri 11-13 Matthews A ... timetable for all classes (https://webcms3.cse.unsw.edu.au/COMP1521/19T2/timetable)
Workload	In class: lectures: 4 hours/week, tutorial/laboratory: 3 hours/week. Outside class: tute/lab: 2 hours/week, assignments: 40 hours, quizzes: 5 hours, ...

Course Website	https://webcms3.cse.unsw.edu.au/COMP1521/19T2/ (https://webcms3.cse.unsw.edu.au/COMP1521/19T2/) https://www.cse.unsw.edu.au/~cs1521/19T2/index.php (https://www.cse.unsw.edu.au/~cs1521/19T2/index.php)
Handbook Entry	https://www.handbook.unsw.edu.au/undergraduate/courses/2019/COMP1521/ (https://www.handbook.unsw.edu.au/undergraduate/courses/2019/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 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

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

Graduate Capability	**Acquired in**

Graduate Capability	Acquired in
Scholars capable of independent and collaborative enquiry, rigorous in their analysis, critique and reflection, and able to innovate by applying their knowledge and skills to the solution of novel as well as routine problems	1-9
Entrepreneurial leaders capable of initiating and embracing innovation and change, as well as engaging and enabling others to contribute to change	-
Professionals capable of ethical, self-directed practice and independent lifelong learning	-
Global citizens who are culturally adept and capable of respecting diversity and acting in a socially just and responsible way	-

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 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.

Tutorials

From week 1, you will also be expected to attend 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 attendance.

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. Labs will be done in pairs, and you and your lab partner should discuss the exercises before going to the lab, to maximise the usefulness of the class. The exercises will need to be submitted and will be assessed by your tutor. During the lab, your tutor will provide feedback on your approach to the problem and on the style of your solution.

Tutors will facilitate you forming pairs in your week 1 lab. The pairs will change several times during the session.

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.

To obtain a mark for a lab exercise, you should (as a pair) demonstrate the completed lab exercise to your tutor during a lab class and both members of the pair should separately 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 Monday 08:59am in order to get any marks for it. You must then (as a pair) also demonstrate your work to your tutor during the first hour of the following week's lab. The code that you submit by Monday is what will be assessed.

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

- *submit your lab work by the following Monday 08:59am*
- *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 two small programming tasks (one MIPS, one C). You must complete the Practice Prac Exam in the lab in the scheduled week.

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

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.

Blogs

We expect all students to maintain a reflective blog during the session, starting week 1. This is useful for us to keep up with what you're doing, but, more importantly, it is helpful for you to think about the improvement of your skills as the course progresses, and to think about how you'll do better in the future.

Up to 1 bonus mark is available for students who keep a regular, insightful blog.

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 (due Week 6; worth 9%)
- Assignment 2 (due Week 10; worth 11%)

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

Final Exam

There will be a three-hour final exam, to be held in the CSE labs during the exam period. 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% (24/60) 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 (https://student.unsw.edu.au/conduct) , Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)) 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.

In particular, 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 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 from UNSW.

If you have any concerns, you may raise them with your lecturer, or approach the School Ethics Officer (mailto:ethics-officer@cse.unsw.edu.au) , Grievance Officer (mailto:grievance-officer@cse.unsw.edu.au) , or one of the student representatives (mailto:stureps.ug@cse.unsw.edu.au) .

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 (https://student.unsw.edu.au/plagiarism) using the words or ideas of others and presenting them as your own. Plagiarism undermines academic integrity and is not tolerated at UNSW.

UNSW and CSE treat plagiarism as academic misconduct, which means that it carries 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. If you haven't done so yet, please take the time to read the full text of UNSW's policy regarding academic honesty and plagiarism (https://student.unsw.edu.au/plagiarism) , and of UNSW's Plagiarism Procedure (https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf) .

Ignorance is not accepted as an excuse for plagiarism. In particular, you are also responsible for ensuring that your assignment files are not accessible by anyone but you by setting the correct permissions in your CSE directory and code repository. 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 (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.pdf)
- Student Misconduct Procedure (https://www.gs.unsw.edu.au/policy/documents/studentmisconductprocedures.pdf)
- Plagiarism Policy Statement (https://www.gs.unsw.edu.au/policy/documents/plagiarismpolicy.pdf)
- Plagiarism Procedure (https://www.gs.unsw.edu.au/policy/documents/plagiarismprocedure.pdf)

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

- Essential Advice for CSE Students (https://www.engineering.unsw.edu.au/computer-science-engineering/about-us/organisational-structure/student-services/policies/essential-advice-for-cse-students)

Special Consideration

If your work in this course is affected by unforeseen adverse circumstances, you should apply for Special Consideration through MyUNSW, including documentation on how your have been affected. If your request is reasonable, and your work has clearly been impacted, then

- for an assignment, you may be granted an extension
- for a Practice Prac Exam, you may be granted an opportunity to take the exam later
- for the Final Exam, you may be offered a Supplementary Exam

Note the use of the word "may": none of the above is guaranteed. It depends on you making a convincing case that the circumstances have clearly impacted your ability to work.

UNSW handles special consideration requests centrally (in the Student Lifecycle division), so all special consideration requests must be submitted via the UNSW Special Consideration (https://iaro.online.unsw.edu.au/special_consideration/home.login) website.

Special consideration requests must be accompanied by documentation, which will be verified by Student Lifecycle. Do *not* email the course convenor directly about special consideration.

If you cannot attend the Final Exam because of illness or misadventure, then you must submit a Special Consideration request, with documentation, through MyUNSW within 24 hours of the exam. If your request is reasonable, then you will be awarded a Supplementary Exam.

Note that UNSW expects you to be available to sit Supplementary Exams (held in the week of Sep 9-13) if required. If you are awarded a Supp and do not attend, then your exam mark will be zero.

For further details on special consideration, see the UNSW Student (https://student.unsw.edu.au/special-consideration) website.

If you are registered with Disability Services, please forward your documentation to cs1521@cse.unsw.edu.au (mailto:cs1521@cse.unsw.edu.au) within the first two weeks of session.

Assessment

Item	Topics	Due	Marks	LOs
Quizzes	All topics	End of weeks 1,2,4,6,8,10	10%	1-9
Assignment 1	Assembly programming	Week 6	9%	4
Assignment 2	C programming	Week 10	11%	2,5
Labs	Most topics	All Weeks	10%	1-5,9
Final Exam	All topics	Exam period	60%	1-9

Each quiz is worth 2 marks. Your total Quiz mark is computed as a sum of your best 5 individual quiz marks.

Lab marks vary. Your total Lab mark is computed by summing all of your lab marks and mapping into a mark out of 10. It is possible for your total Lab mark to be 10/10, even if you miss one lab.

Most labs are done in pairs, but the Practice Prac Exam is done individually.

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 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	Quizzes

Week	Lectures	Tutes	Labs	Assigns	Quizzes
1	Course intro, computer systems, queues, stacks, memory, data representation	Welcome, C revision, Stacks, Queues, Bits	Input/output	-	Quiz 1
2	Data representation (cont.)	Data representation (i)	Bit Manipulation	-	Quiz 2
3	Instruction set architecture, assembly language programming	Data representation (ii)	Data Representation (floats)	-	-
4	Assembly language programming (cont); Compilation, mapping C to assembler	Assembly language	Assembly programming (i)	-	Quiz 3
5	Computer systems architecture, layers, operating systems, system calls	Mapping C to MIPS assembler	Assembly programming (ii)	-	-
6	File systems; memory management	OSs; system calls	System calls	Ass1 due	Quiz 4
7	Virtual memory; processes; interrupts; i/o	File system functions	File system functions	-	-
8	Parallelism, synchronisation, coordination, communication	Virtual memory; processes	Signals	-	Quiz 5
9	Network architecture	Parallelism; concurrency	Concurrent processes	-	-
10	Spare	Networks	Practice Prac Exam	Ass2 due	Quiz 6

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 (http://csapp.cs.cmu.edu/))

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 (http://nand2tetris.org/) , including lecture slides)

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

Course Evaluation and Development

This course ran for the first time in 17s2, and then ran in 18s1 and 18s2. The feedback from those offerings was generally positive. However, the initial material on ADTs and linked structures proved challenging for students from COMP1911. Since linked structures do not feature prominently in systems-level code, we will not discuss them.

The 18s1 and 18s2 offerings ran in a 10-week UNSW3+ mode, and there were no complaints about the pace of the course. This suggests that we won't have any significant issues in 19T2.

The course will be evaluated at the end of the session using the myExperience system. However, you are encouraged to provide feedback during the session so that we can address any problems as soon as possible.

Resource created about a year ago (Monday 01 April 2019, 05:11:54 PM), last modified 11 months ago (Wednesday 29 May 2019, 11:23:14 AM).

Comments

(/COMP1521/19T2/forums/search?forum_choice=resource/25621)

(/COMP1521/19T2/forums/resource/25621)

There are no comments yet.