Course Outline

The aims of this course are to introduce students the basic concepts and major components of microprocessors and microcontrollers and to provide students with knowledge and skills for solving problems with microprocessors /microcontrollers.

Contents

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

Course Details

Course Code	COMP2121
Course Title	Microprocessors and Interfacing
Convenor	Hui Wu (/users/z3117461)
Admin	Hui Wu (/users/z3117461)
Classes	**Lectures** : ...fill in times/locations of lectures... Timetable for all classes (/COMP2121/17s2/timetable)
Consultations	... fill in the times/locations of consultations...
Units of Credit	6
Course Website	http://cse.unsw.edu.au/~cs2121/17s2/ (http://cse.unsw.edu.au/~cs2121/17s2/)
Handbook Entry	http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP2121.html (http://www.handbook.unsw.edu.au/undergraduate/courses/current/COMP2121.html)

Course Summary

The aims of this course are to introduce students the basic concepts and major components of microprocessors and microcontrollers and to provide students with knowledge and skills for solving problems with microprocessors /microcontrollers.

Assumed Knowledge

Before commencing this course, students should:

- be able to competently program in C;
- be able to represent integers in binary;
- be able to work independently; and
- be able to work in a group.

These are expected to have been taught in COMP1021 or COMP1091 or COMP1721 or COMP1921 or COMP1927

Student Learning Outcomes

After completing this course, students will:

- know the fundamentals of microprocessors and microcontrollers;
- understand how C programs are converted into AVR machine instructions;
- be proficient in AVR assembly language programming;
- understand typical interface designs between hardware and software; understand how serial communication works;
- understand how analog signals are converted into digital signals and vice versa;
- obtain basic development skills for microprocessor / microcontroller applications.

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
scholarship: capable of independent and collaborative enquiry	labs
scholarship: rigorous in their analysis, critique, and reflection	assignments
scholarship: able to apply their knowledge and skills to solving problems	labs
scholarship: capable of effective communication	assignments
scholarship: information literate	course work
scholarship: digitally literate	course work
leadership: enterprising, innovative and creative	labs
leadership: collaborative team workers	labs
professionalism: capable of independent, self-directed practice	labs
professionalism: capable of lifelong learning	assignments
professionalism: capable of operating within an agreed code of practice	labs

Teaching Strategies

- Lectures: introduce concepts, system components with examples
- Lab exercises: program with AVR assembly programming language
- Assignments: allow students to learn a different microprocessor
- Projects: allow students to solve significant problems

In this course a variety of teaching strategies will be used.

1. Teaching in class, where lectures are delivered and interaction encouraged.
2. Laboratory exercises are performed under supervision and assessed.
3. Two assignments are given which enable you will expand upon the concepts already learned, or will apply concepts learned to create an artifact.
4. Homework will be released each week, which you can do in your own time, and is not assessed.

You are expected to spend at least one hour on the course for every hour you spend in class (i.e., at least 5 hours per week). You should practice programming as much as possible.

Creativity in subjects such as this will only be possible when you are an expert. It is no different from being a pianist, artist, or a mathematician.

You should understand all introduced concepts clearly. If not, you are encouraged to stop and ask.

Teaching Rationale

This course teaches the theory and skills needed to design and implement systems utilising microprocessor systems. Passing this course involves keeping up with the theory and putting in the time to complete the lab exercises. The lab work brings the theory alive and clarifies deeper issues. Without the theory, the lab work is difficult to master. Each therefore complements the other. Both require time. If you allocate sufficient time each week, you won't have any problems passing, and it can be a very rewarding experience. It is our belief that certain activities make a world of difference to your learning experience at University. You should:

- actively participate in lectures - if something is unclear, ask for clarification - if you don't understand something, try to figure it out when you reflect upon the material during private study, and if you still don't get it, raise it during the next class.
- don't imagine you are the only one having trouble with something; chances are somebody else is having trouble as well. However, lecturers aren't necessarily aware of your problems, so please let them know when something is troubling you.
- be prepared to concentrate during classes - if you only pay them half your attention, you'll probably only get half as much out of the experience as you would had you focused on the task completely
- actively participate in tutorials - prepare beforehand - know which problems you have difficulty solving - be prepared to participate in deriving a solution during the tutorial.
- prepare for labs by reading up what is expected well before your lab session - sometimes you will need to complete some preparatory exercise - lab time is too valuable to waste on preparations that could have been completed beforehand.
- make use of the staff; pester your lecturer, your tutor, and your lab demonstrator when you don't understand something.

Academic Honesty and Plagiarism

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 undermines academic integrity and is not tolerated at UNSW. Plagiarism at UNSW is defined as using the words or ideas of others and passing them off as your own.

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)

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

- Student Code Policy (https://www.gs.unsw.edu.au/policy/documents/studentcodepolicy.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)
- Student Misconduct Procedure (https://www.gs.unsw.edu.au/policy/documents/studentmisconductprocedures.pdf)

Assessment

To pass this course, you MUST get at least 50% of the full marks in the labs, return the lab kits, and achieve at least 40/100 in the final exam and 50/100 in the final result. Your final result is calculated based on:

- Lab exercises (20%)
- Assignment (5%). Due at 23:59 Sept 10, 2017
- Project (15%). Due in Week 13
- Final exam (60%).

The final exam is a closed book, written exam, and lasts 2 hours.

Course Schedule

Each week there are two lectures. The following topics will be covered:

- Instruction set architecture.
- Number systems
- AVR assembly programming and assembly process
- Interrupts
- Buses and I/O
- Serial communication
- Analog/digital and digital/analog conversions.

Resources for Students

Text Book:

- Fredrick M. Cady: Microcontrollers and Microcomputers-Principles of Software and Hardware Engineering.

Further References:

1. AVR Data Sheet (on line).
2. AVR Instruction Set (on-line).
3. AVR Assembler Tutorial (on-line).
4. LCD Manual (on-line).
5. Mano, M.M. & Kime, C.R.: Logic and Computer Design Fundamentals, 3rd Edition, Prentice-Hall (2003).
6. Brian Kernighan & Dennis Ritchie: The C Programming Language, 2nd Ed., Prentice Hall, 1988, ISBN: 0-13-110362-8.
7. David Patterson and John Hennessy: Computer Organization & Design: The HW/SW Interface," 2nd Ed Morgan Kaufmann, 1998, ISBN: 1 - 55860 - 491X.

Course Evaluation and Development

This course is evaluated each session using the myExperience system.

Resource created 4 years ago (Wednesday 19 July 2017, 11:58:56 AM), last modified 4 years ago (Friday 25 August 2017, 01:07:30 PM).

Comments (/COMP2121/17s2/forums/search?forum_choice=resource/11151) (/COMP2121/17s2/forums/resource/11151)

There are no comments yet.