Computer Organisation COMP2008 Laboratories

INTRODUCTION:

Computer Organisation lab consists of **11 workshops** (or laboratories, or practicals). The labs start from week 2 (no lab work in week 1). See the detailed delivery schedule on the subject Web site.

There is a lab sheet for each workshop, which contains the description of the preparation required for the workshop, a list of tasks to be performed, and a number of questions related to workshop topics.

In the labs, students will write programs using MIPS assembly language. The CLASSIC software tool used is **PCSpim/QtSpim**, which is a simulator of the MIPS RISC processor. Using a simulator, instead of using the assembler directly has many advantages in the learning environment. With the simulator it is easy to debug programs, to observe and manipulate the contents of registers and memory etc. The simulator offers much more control over the execution of the programs than bare hardware. Naturally there are also some limitations, especially when it comes to simulating the interactions between the program and its environment, and also a considerably longer execution time.

LAB WORK:

The amount of time needed to complete all the practical tasks may exceed the time allocated to the supervised labs. There are no assignments in this subject, and the total workload is similar to the workload in other subjects at this level. Students are expected to work in their own time, writing, debugging, and testing their programs. **PCSpim/QtSpim**, the main software tool used in the labs, is free. If you have a PC at home, it is a good idea to get a copy of PCSpim/QtSpim (http://pages.cs.wisc.edu/~larus/spim.html, or https://sourceforge.net/projects/spimsimulator/files/, where versions available for Windows, Mac, and Linux boxes), so you can prepare for the labs, and complete the work started at the Uni. You are also allowed to use your own laptops in the lab, if you wish to do so.

In addition to PCSpim/QtSpim, there are other SPIM programming environments: MARS, MIPSter, and EduMIPS64 etc. You can explore MARS, MIPSter, or EduMIPS64 in your own time if you wish to use them for your lab exercises. Each simulator has its advantages; you have free choices. Note that the classic **PCSpim** is not outdated; rather its operation style helps reveal many technical details. The **PCSpim** interface will be normally referenced to as samples in practicals or tests.

It is expected that you read lab instructions, and **prepare yourself well before coming to the lab.** You can do as many lab tasks as you like at home, before coming to your lab. However please **DO NOT** come to a lab completely unprepared, you will be wasting your time and risking getting very low mark.

DUE TIME:

The subject learning guide says practicals are due by "weekly tutorial time for the corresponding laboratory tasks" and the lab sheet gives the due week. They together indicate the accurate due time for an individual student registered in a specific tutorial session. That is, the practical work is due by the beginning of the weekly tutorial time (not after the tutorial session) that the student is registered in. Sorry, it's not feasible to give due on Tuesday at 11:59pm or Wednesday at 09:59 in the weekly lab sheet as there are different tutorial sessions each week.

Lab 1 starts in week 2 and is supposed to be submitted before your corresponding session starts in week 3;
Lab 2 starts in week 3 and is supposed to be submitted before your corresponding session starts in week 4;
Lab 3 starts in week 4 and is supposed to be submitted before your corresponding session starts in week 5;
… …
Lab 6 starts in week 7 and is supposed to be submitted before your corresponding session starts in week 9 (week 8 is intra-session break);
Lab 7 starts in week 9 and is supposed to be submitted before your corresponding session starts in week 10
Lab 8 starts in week 10 and is supposed to be submitted before your corresponding session starts in week 11;
… …

/* A formal expression for starting week and due week of a lab[i] */

With Lab[i], i = {1..11}
Lab[i].start = Week[i+1] (i <= 6)
Lab[i].start = Week[i+2] (i > 6)
Lab[i].due = before your corresponding session in Week[i+2] (i < 6)
Lab[i].due = before your corresponding session in Week[i+3] (i >= 6)

The subject learning guide states that "Unless an extension to a further date is granted to the student by the subject coordinator, no late submission is accepted. An extension of time may be granted only under exceptional circumstances by the subject coordinator. Resubmission of the laboratory work is not permitted in this subject." So please be advised that there are normally no catch-ups and no late submissions. That is, practical tasks submitted after the starting of the due practical session, which the student is registered in, are not accepted normally.

Students should follow the subject administrative rules stated in the subject learning guide. We ensure all students are treated fairly and equally; they are studying in a "level-playing field" that no one suffers from any disadvantage and no one is offered more advantages over others.

ASSESSMENT:

There are 11 lab tasks, maximum mark for each workshop varies as described in each lab sheet. Total max. mark of these 11 lab tasks is 40, which constitutes 40% of the total mark for the subject. Please see also the Subject Learning Guide for additional details relating to assessment process.

The answers to the lab questions should be well documented for grading. The commonly acceptable documentation format is DOC, PDF, and TXT. The work submitted should include:

- All the necessary written responses in text, schematic drawing, or other appropriate format.
- Recorded screen shots or step-by-step description for lab demonstration activity.
- As well as the relevant .s source files (compulsory for programming tasks).

Only questions completed are assessable. There are no remedial labs; no extensions and late submissions are accepted without a genuine reason.

Any program logical errors will detract from the full mark. A logical error is any discrepancy between the program specification and its actual functionality, failure to test for boundary conditions, failure to explore full range of input data etc. Students are encouraged to present programs which are fully debugged, and do not have assembly language errors, or run time errors.

/* For on-campus in-class marking */

You must present your work to the class tutor for grading. The work submitted to the tutor can be printed, neatly hand-written, or eCopy on computer. For questions where written answers are required, only answers in writing are assessable. **No marks for the lab can be obtained without demonstrating your work during laboratory**.

/* For online teaching mode */

You are required to submit your work into the corresponding drop-boxes on vUWS. The grading will be mainly based on the submissions on vUWS. The work submitted should be well-presented and have adequate information ready for after-class marking. You'll likely use a Word .doc file to organise the required written answers to lab questions (preparation questions and lab tasks). For coding tasks, .s program file(s) (source code) must be submitted separately (rather than included in the word document). Faulty submissions (e.g. Prac 3 is submitted as Prac 4 where the latter is supposed to be submitted; the work for a totally different subject is submitted as what is expected for this subject) will result in a zero mark for the submitted practical work.

With online teaching, the in-class demo for the practical tasks is not normally pre-scheduled. However, students may be randomly selected to demonstrate their lab work during tutorial time online (e.g. via Zoom

Breakout room) when possible.

MOSS - A System for Detecting Software Plagiarism

MOSS (for a **M**easure **O**f **S**oftware **S**imilarity) is an automatic system for determining the similarity of programs. To date, the main application of Moss has been in detecting plagiarism in programming classes. Since its development in 1994, Moss has been very effective in this role. The algorithm behind moss is a significant improvement over other cheating detection algorithms. There is a great deal of information regarding MOSS at: http://theory.stanford.edu/~aiken/moss/

MOSS can currently analyse code written in the following languages:

> C, C++, Java, C#, Python, Visual Basic, Javascript, FORTRAN, ML, Haskell, Lisp, Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, **MIPS assembly**, a8086 assembly, a8086 assembly, HCL2.

MOSS system is being used in this subject for any suspected plagiarism. If plagiarism is detected by MOSS and upheld after investigation, you will to face a charge of academic misconduct. So please do not attempt to plagiarise.