Computer Organisation 300096 Laboratories

INTRODUCTION:

Computer Organisation lab consists of **11 workshops** (or laboratories, or practicals). The labs start in week 2. See the detailed delivery schedule on the unit 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. **Only questions answered in writing and completed are assessable.** There are no remedial labs, and no extensions; no late submissions are accepted.

The written work submitted to the tutor should be printed or neatly hand-written. After your lab has been assessed the tutor writes your grade, and signs it on the first page. Please **KEEP marked copy of the lab,** no queries about your lab are accepted if you do not have a copy of corresponding marked lab work sheet.

In the lab students will write simple programs using MIPS assembly language. The software tool used is **PC Spim**, 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. **PC Spim**, 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 PC Spim (http://pages.cs.wisc.edu/~larus/spim.html), 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. Note: In addition to PCSpim, there are other SPIM programming environments: MIPSter, MARS, and EduMIPS64. You can explore MIPSter, MARS, or EduMIPS64 on your own time if you wish.

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.

ASSESSMENT:

You must present the tutor with printed copy of your code, which has your name and student number in the comments. **Please keep the cover sheet marked by the tutor.** The tutor may like to keep (but does not have to keep) other parts of your work: your printed code, answers, etc.

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 Unit Outline and Learning Guide for additional details relating to assessment process.

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. **No marks for the lab can be obtained without demonstrating your work during laboratory**. Students are encouraged to present programs which are fully debugged, and do not have assembly language errors, or run time errors.

No task, or any part of it, will be accepted via e-mail.

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 unit 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.

Lab Sheet 1 Computer Organisation 300096

(Starts session week 2, due in week 3)

Student Name and Number	
Date, Grade and Tutor signature, **max mark 3**	

Keep this cover sheet marked and signed by the tutor.

1. Preparation [Total max. mark: 1]

The main goals of today's lab are: to understand how to write simple programs, to become familiar with PC Spim – the MIPS processor simulator, and to learn how to use its features to debug your programs. Note for 64-bit users: The first time you run PCSpim, it may complain about a missing exception handler (exceptions.s). If you see this message, open Simulator->Settings, look under "Load exception file", and change the path to the following (or something similar):

C:\Program Files (x86)\PCSpim\exceptions.s

1. Before the lab please study "**PCspim.pdf**" and section A.9 (Page 40) in "**HP_AppA.pdf**" (available on the website), as well as lecture notes and relevant sections from the textbook.

General Data	UnitOutline | LearningGuide | Teaching Schedule | Aligning Assessments |
Extra Materials	ascii_chart.pdf | bias_representation.pdf | HP_AppA.pdf | instruction decoding.pdf | masking help.pdf | PCSpim.pdf | PCSpim Portable Version | Library materials

2. The following exercise is to familiarise yourself with **PCSpim**. Note: If you use **QtSpim**, your exercise here will be based on QtSpim.

   a) Make a **hand drawing** of the PCSpim user interface [**0.5 marks**]

   Note: Your hand drawing doesn't need to copy all the information on the user interface; just pick up a couple of lines from each area (pane).

   b) Explain in writing what is the role of four main horizontal areas (panes) in the PCSpim user interface [**0.5 marks**].

2. Workshop Tasks [Total max. mark: 2]

Please answer the lab Questions listed below in writing, print or neatly write your answer:

1. Create a folder for this subject (e.g. CO_300096), and within it create a subfolder for Lab01. Copy two assembly language files *hello*.s and *helloimproved*.s (see Prac. 01 code on the website) to the newly created folder. **NOTE:** PCSpim assembly programs are text files, they can be edited with any text editor, for example NotePad, NotePad++, etc.

2. Start PCSpim. Open file *hello*.s in PCSpim and run it. Meanwhile open the same file in a text editor (e.g. NotePad++, for easy view) and analyse the code. You must read and fully understand all comments! (# ← comments in the code). **NOTE:** The PCSpim simulator is only for running assembly programs and doesn't support code editing; it's not a good code viewer either. For editing assembly programs, you have to use a text editor. So for the lab tasks regarding assembly programs, you'll switch between PCSpim and a text editor.

3. Open *helloimproved.s* in PCSpim and run it. Also open the same file in a text editor (for easy view) and analyse the code; compare it with the *hello.s* code.

   Run *helloimproved*.s in PCSpim again. Pick up a line of code statement from PCSpim and identify the following formats for the code statement: the original text of the instruction, the assembled text, and the machine code.
   Question (0.3): Answer the question in writing. List a couple of lines of code statements from PCSpim. Point out which are the *original text* format of the instructions, the *assembled* version, and the *machine code* version. Briefly explain main differences in between.

4. Run the *helloimproved*.s several times, and observe its results in the console window. Identify the data segment, and the registers. Then single-step through the program, observing the changes in the registers. After the whole program has been executed, experiment with the 'clearing registers' action.
   Question (0.2): What is the effect of clearing registers? Why simulator provides this operation?

5. Still with *helloimproved*.s. Find comment '# line xx' and insert left to the comment mark '#' the following: `sub $s0, $t0` Save and load the program.
   Questions (0.2): Observe what's happening when running the program. What was the effect from this code modification? Explain why.
   [Restore the program to the original form by deleting '`sub $s0, $t0`']

6. Modify the *helloimproved*.s to print on the screen two additional lines of text: one line with your name, and one line with any text you like. Save your program as *myfirstprog.s* Remember to write clear, descriptive comments for your part of the code.
   Demonstration (1): demonstrate to the tutor the program you wrote. Be prepared to explain the code, and answer additional questions about the code and PCSpim.

7. **Questions (0.3):** When running *helloimproved*.s, after the output text is printed to the console, why the cursor is placed in the next (blank) line? What is the role of instruction ‘syscall’? How syscall *services* and *arguments* are selected and specified in the code?

3. Assessment

When you complete all tasks, present to the tutor a **printed copy of your program source code** with your name and student number included in the comments in your code (#…), and typed or neatly written answers to all questions listed in the lab sheet (note that amount of questions vary per lab).

Warning: Any source code duplicated amongst students will result in a zero mark, and possible further action according to the UWS policy on plagiarism (see the unit learning guide).