COMP 30080: Computer Systems

2021 / 2022

Assignment 5

Pavel Gladyshev

UCD School of Computer Science,
University College Dublin,
Belfield, Dublin 4.

Introduction

There are five assignments in total. Assignments should be submitted using CS Moodle via the 'Assignment submission' links. The assignment deadlines are provided in Moodle. Roughly speaking, one assignment is to be submitted every two weeks of term. Moodle will allow submissions up to 2 weeks late. However, late submissions incur penalties according to UCD policy unless a medical certificate or similar is submitted to the lecturer or the UCD Science Programme Office.

Assignments do not have to be done during lab times. However, Demonstrators are only available during lab times.

Assignment 1 is not assessed. Students should compare their solutions with the model solution provided. All assignment marks and feedback will be available in Moodle.

For all assignments:

- For each assignment, submit an accompanying report (.doc or .pdf) which includes a brief explanation of what you have produced, screenshots showing verification of the submitted program, any workings and the answers for written work questions. Use your student number and assignment number as the file name, e.g. 12345678_ass1.doc
- Make sure that your name and student number are visible in the assembly files, circuits, and report.
- If necessary, scan (or photograph) any handwritten work into a file for submission. Submit the graphics file (.gif or similar) or include the scan in the report.
- Submission is via Moodle. Moodle requires a single file that should be a .zip of all individual files. Use your student number and assignment number as the file name, e.g. 12345678_a1.zip

PLEASE NOTE, THE REPORTS MUST BE IN WORD OR PDF FORMAT, THE ASSEMBLY PROGRAMS MUST BE IN ASM (TEXT) FORMAT AND THE SUBMISSION MUST BE IN ZIP FORMAT. OTHER FORMATS OR CORRUPT FILES WILL NOT BE MARKED

The marking scheme for most questions is: 0=nothing done; 1-3=partial working; 4=working; 5=working and elegant.

Assignment 5: Extending Single-Clock MIPS CPU

In this assignment we will extend a Logisim implementation of MIPS CPU with new MIPS instructions.

1) Required software

For this assignment you will need to use Logisim, a graphical tool for designing and simulating logical circuits. It is already pre-installed in **mips-tools** virtual appliance, but, if you prefer, you can download and install it from its web page:

http://www.cburch.com/logisim/

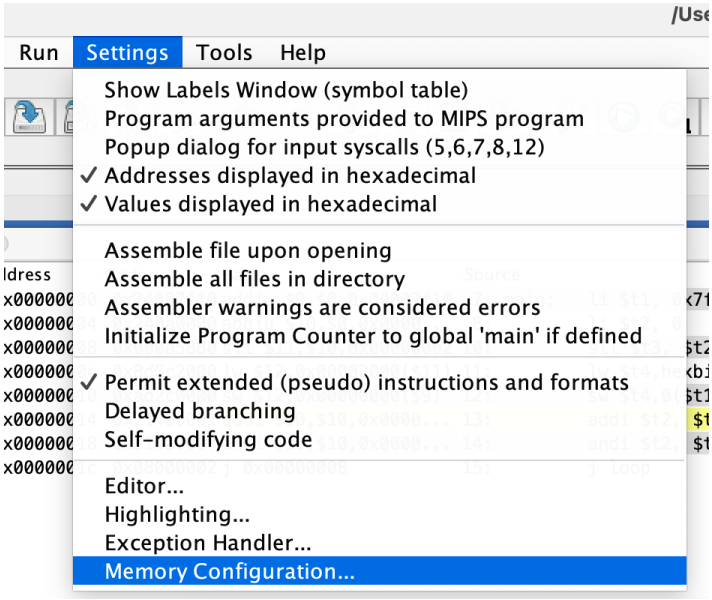
You will also need Mars MIPS simulator, which we used in Assignments 1 and 2.

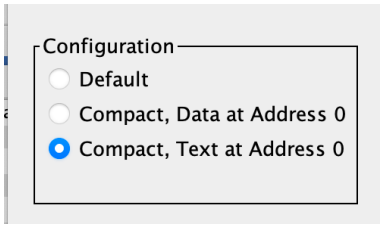
2) Logisim circuit and the accompanying files

Download ZIP file **a5-files.zip** from CS Moodle and extract it somewhere on your computer. The MIPS CPU model you will be expanding is in the file **a5-mips.circ** it can be opened in Logisim (in menu: File → Open…).

Familiarize yourself with the design of the circuit and its sub-circuits and try running sample demo.asm provided. To run the demo program in Mars you will need to perform the following steps:

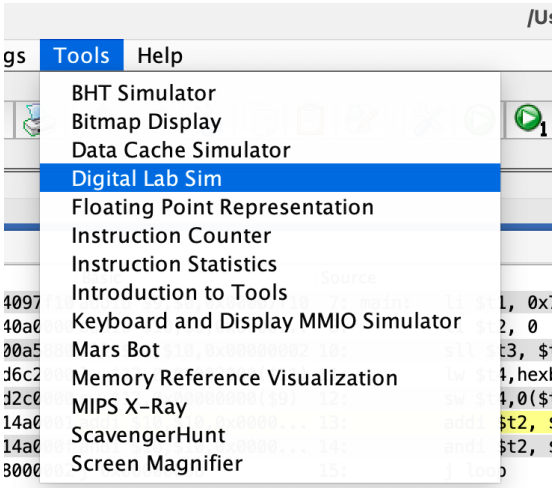
1. Switch Mars simulator into Compact Memory mode with .text starting at 00000000. You can find it in Settings → Memory Configuration...

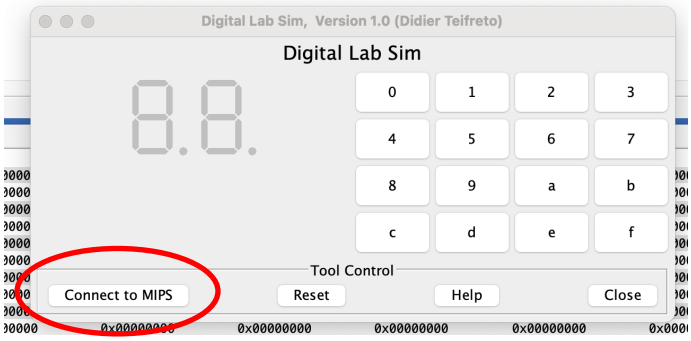




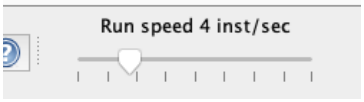
Once you change Mars settings, you will need to **re-start Mars**.

2. Enable Digital Lab Sim tool and connect it to MIPS. You can find it in Mars menu under **Tools → Digital Lab Sim**.

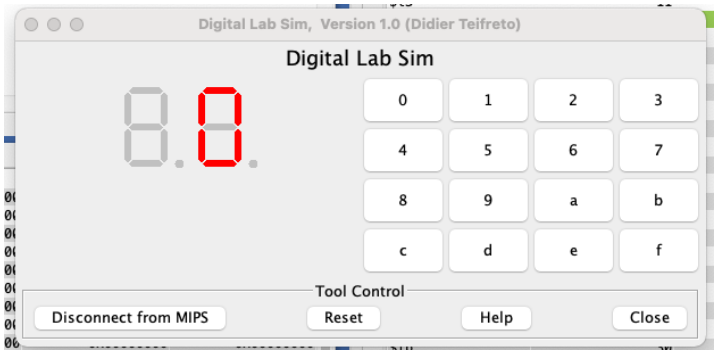




3. Reduce simulation speed to 4 instructions per second (Use Run speed slider in the Mars toolbar).

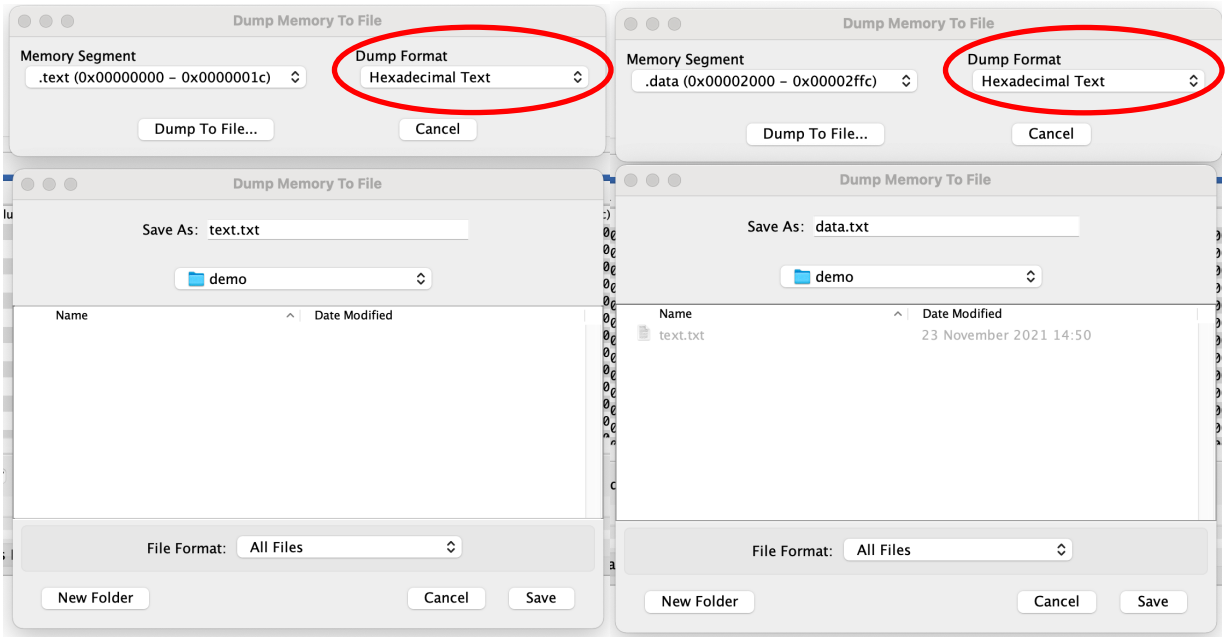


4. Now load the demo.asm file from a5-files.zip, assemble it and run it. If all goes well, you should see hexadecimal digits appearing on the 7-segment display in sequene:

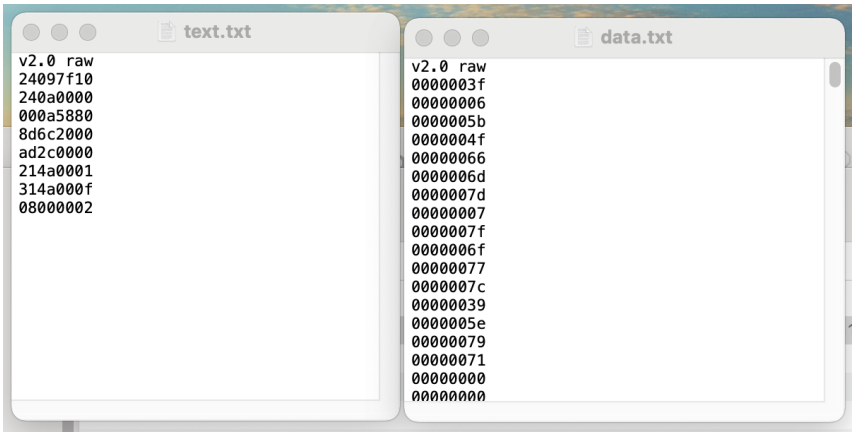


One you get demo.asm running in Mars as explained above, run it in **a5-mips.circ** circuit in Logisim following these additional steps:

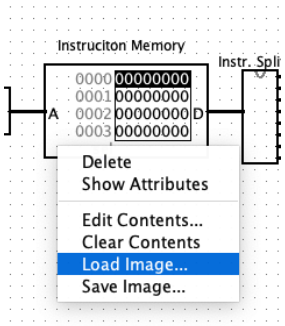
5. Export .text and .data memory segments of the assembled program from Mars into text files using Dump Memory facility. You can find it in Mars under Files → Dump Memory. **Important:** choose "**Hexadecimal Text**" as Dump Format when saving the contents of .text and .data segments!



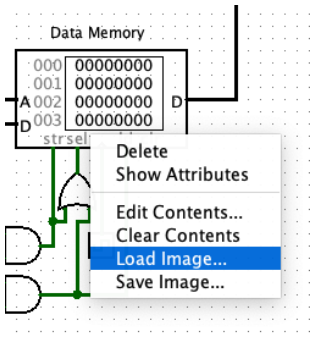
To make the produced files readable by Logisim, open each file in text editor and add **v2.0 raw** as the first line:



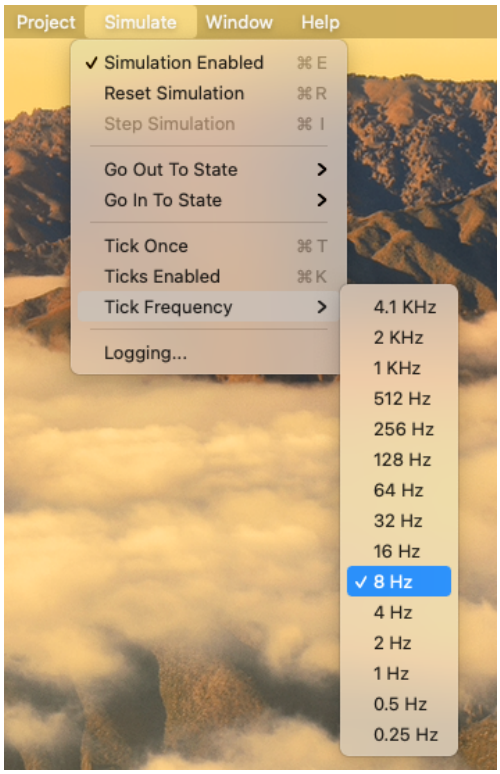
6. Start Logisim and open **a5-mips.circ** file.
7. Reset simulation (In Logisim menu: Simulate → Reset Simulation). This step will wipe out the content of all RAM modules.
8. Load the prepared text.txt file into Instruction Memory by right-clicking on it and choosing Load Image… menu item:



9. Load the prepared data.txt file into Data Memory by right-clicking on it and choosing Load Image… menu item:

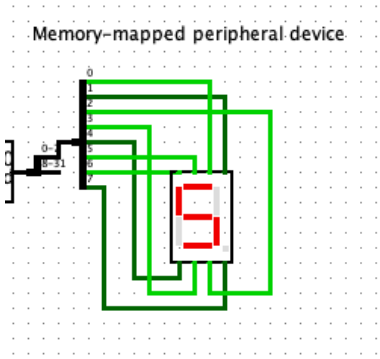


10. To step through the program click Logisim menu item: **Simulate → Tick Once** or just press **Ctrl-T**.
11. To run the program reduce the clock speed to 8 Hz



Then click Logisim menu item **Simulate → Tick Enable**

If all goes well, you should see hexadecimal digits appearing on the 7-segment display as in Mars:



3) Assignment question

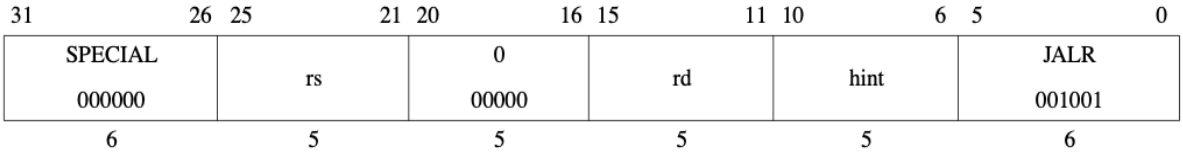
In this assignment you are asked to extend a5-mips.circ to perform MIPS instruction JALR (see Appendix) and run the program test.asm included in a5-files.zip.

Submit the resultant .circ file along with the brief report describing the approach you took (a ZIP file) as the answer to this assignment.

(20 marks in total)

Appendix

Jump and Link Register — JALR



Format: `JALR rs (rd = 31 implied)` MIPS32 (MIPS I)
`JALR rd, rs` MIPS32 (MIPS I)

Purpose:

To execute a procedure call to an instruction address in a register

Description:

`rd ← PC+4, PC ← rs`

Place the return address link in GPR *rd*. The return link is the address of the instruction following the branch, where execution continues after a procedure call.

Restrictions:

The effective target address in GPR *rs* must be naturally-aligned.