FIT1008 – Intro to Computer Science
Workshop Week 2

Semester 1, 2018

Objectives of this practical session

- To be able to write very simple MIPS programs.
- To familiarise yourself with a MIPS simulator (*MARS*).
- Understand how the MIPS architecture relates to assembling and executing simple MIPS programs.

Notes

- Use comments to document your code.
- We will restrict ourselves to instructions in the MIPS reference.
- In this prac you can work in pairs, but your code review should be done with a different partner.
- You can download the MARS simulator here: `http://courses.missouristate.edu/KenVollmar/mars/`

Task 1

Let's warm up with something simple but fundamental [1].

```
1  print("Hello MIPS")
```

[1] Assembly is like maths. You become good at it if you can spend time with it.

- Translate and run the program above to MIPS [2].
- List and discuss all the parts that are implicit in the Python version.
- In your MIPS code identify the following:
  - System calls.
  - MIPS instructions (discuss the purpose of each instruction).
  - Labels.
  - Assembly directives.
  - Global variables.
  - References to general purpose registers.

[2] You **should** use the MIPS reference document – it not necessary to memorise things. If you do not succeed at first, try again. Be patient. If you decide to copy the code from *somewhere* make sure that you understand the meaning of each line

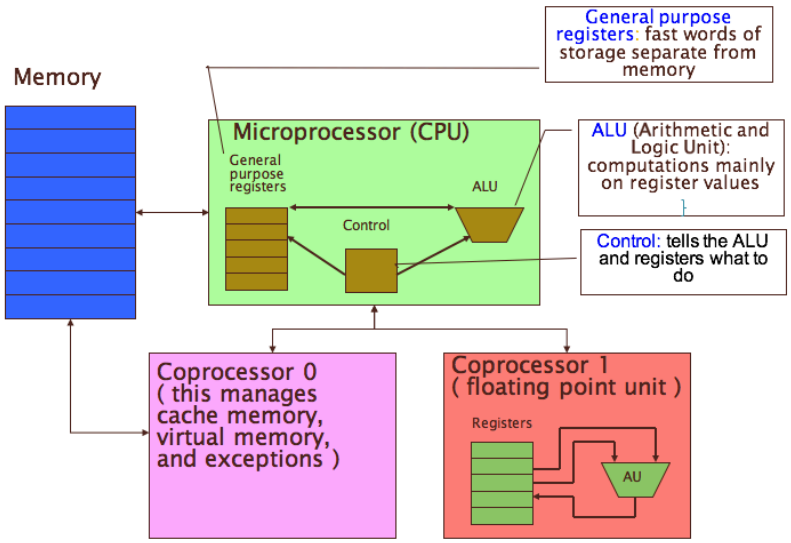

Figure 1: Simplified MIPS architecture

Task 2

When we discussed the MIPS architecture, things were perhaps a little abstract. But now that we can use the MIPS simulator those concepts will become more concrete.

1. Start by assembling your code. If there are no errors in your code, you should see that the interface of MARS switches from *Edit* to *Execute*.
2. Once you are in the *Execute* tab, take a moment to look carefully at all the elements in the graphical interface. You will see that some of them correspond to concepts that we discussed in the architecture. In particular: *Text Segment* and *Data Segment*, in the **memory**; as well as the **Registers**, which are part of the microprocessor. See Figure 1.
3. You can now run your program all at once (play button), just to check that it works.
4. Run the program step by step. What is the initial value of Register *PC*, see how *PC* changes as you step through the program. What is the content of *PC* referring to?
5. Let's look at the text segment. Is there a pattern in the addresses that correspond to each instruction?
6. Note that each part of your **Source** has been assembled into an instruction.

7. What is in the data segment? Maybe it helps to click the check-box *ASCII*.
8. Inspect the content of the register and how it changes when you run the program step by step.

Hopefully things start to make a little more sense now.

Task 2

- Write a Python program that reads in two integers, a and $\mathbf{b}$ and prints out the quotient and the reminder.
- Translate your program to MIPS [3].
- Run your program step by step, pay attention to how the registers change.

[3] **Hint:** Something about *LO* and *HI*

Task 3

Write a Python program to determine if a given integer is a multiple of 3. Translate the program into MIPS [4]. You can then generalise this, reading two integers and determining if the first one is a multiple of the second one.

[4] **This is the code review task.**

Task 4

Generalise the program in the previous exercise, now reading two integers and determining if the first one is a multiple of the second one.