Exercise sheet 7 – Processor architecture

Goals:

- Pipelining
- Instruction Scheduling

Exercise 7.1: Pipelining

(a) Given is a sequence of instructions and a five-stage-pipeline. State the procedure and be careful, the instructions are not ordered perfectly to fully utilise the pipeline.

Instruction sequence:

Nr.	**Instruction**	**Comment**
(1)	ADD R2, R1, R1	*; R1 = R1 + R2*
(2)	ADD R1, R3, R3	*; R3 = R1 + R3*
(3)	ADD R4, R5, R5	*; R5 = R4 + R5*
(4)	CMP R5, #0, R6	*; R6 = cmp(R5, 0)*
(5)	BNE R6, M	*; Jump to M if result != 0 BNE = branch not equal*
(6)	INST1	*; some random instruction 1*
(7)	INST2	*; some random instruction 2*
(8)	M: INST100	*; some random instruction 100*

Five-stage-pipeline:

Stage	**Operation**
1.	Fetch instruction
2.	Load operands
3.	Execute instruction
4.	Memory access
5.	Save result into register

(b) Suggest ideas for a better pipeline utilisation of *exercise 7.1a*.
Assume there are additional instructions before the given program excerpt.

Exercise 7.2: Pipeline-Simulator

The pipeline simulator is taken from *http://euler.vcsu.edu/curt.hill/mips.html*.

(a) Update the `RA_exercises` repository with `git pull`.

(b) Change into the `RA_exercises/sheet_07/PipelineSimulator` directory.

(c) To compile the pipeline-simulator run `javac GUI.java` from a terminal, or use the `Makefile`.
Hint: Make sure that you have properly installed the JDK and JRE. Within the VM it is already installed.

(d) Start the simulator with: `java GUI`

(e) In *mips.html* you can find some information on how to work with the simulator.

(f) Create a little assembly program, based on *exercise 7.1a* and run the simulator. *Hint: There is no one-to-one mapping possible, because not all required instructions are available.*