Laboratory 5

Objective

The objective of this laboratory is to design, build and simulate a pipelined version of your multi-cycle CPU. The instruction set is identical to the single-cycle version. However, now your CPU should have the following stages:

You must also design and develop an instruction decoder that transforms OP values into control signals for the pipeline. In implementing this laboratory, you will construct a third architecture for your KURM CPU. In Laboratory 4, you implemented an architecture that consisted of a data path entity and a controller entity. In Problem 3, this architecture will be replaced by a new data path entity developed in Problem 1 and a decoder unit in Problem 2. Submit a single VHDL model that integrates the results of Problems 1–3. There is no need to submit separate models for each result.

Problem 1

Using your data path from Laboratory 4 as a starting point, implement a five-stage pipelined data path for the KURM processor. Include registers for storing intermediate data between pipeline stages. Your design should include: (i) fetch, (ii) register access, (iii) ALU operation, (iv) data memory access and (v) register file write-back stages. It is anticipated that your data path from Laboratory 4 will require few modifications beyond the addition of the pipeline registers.

Problem 2

Design and implement an instruction decoder to generate control signals for your pipeline. The decoder should take an instruction (or parts thereof) and generate control signals for each pipeline stage.

The decoder should be implemented as a single entity that accepts an instruction and generates the appropriate control signals. Note that the specific controller interface will differ, based on your implementation. Implement the controller with a lookup table in behavioral VHDL. The table will translate the OP instruction field into control signals for the data path. You may need to extract other information (i.e., the register ID for write-back) as well as decoding the operation.

Problem 3

Integrate your decoder with the pipelined data path from Problem 1. Add sufficient register storage in each pipeline register to incorporate control signals generated by the decoder. Do not concern yourself with forwarding or pipeline stalls in this problem.

The result of your work should be a new KURM architecture consisting of a decoder entity/architecture and a pipelined data path entity/architecture. You should be able to replace the architecture from Laboratories 2 and 4 with this architecture and use the same test bench for evaluation. Note that it may be necessary to modify the memory module to handle two ports rather than one.