

Design and test a VHDL model of the CPU datapath you designed in Project 4, as described by the block diagram and register transfers designed in that project. Assume that memory is outside of the CPU (one memory if you have a Von Neumann architecture; two memories if you have a Harvard architecture). I will provide a sample memory model for the final project. As would be the case with a microprocessor chip, the datapath must have data and address bus “ports” to connect the CPU to each memory. For this project, the datapath should also have its various control and status signals as external “ports”. These will be connected to a Control Unit in the final project. Test the datapath by forcing control inputs to selected values, to mimic the operation of a control unit.

**Notes:**

1. This is to be a register-transfer-level design - not gate level.
2. The top-level model should contain **only component instantiations**, matching your block diagram (changes may be made to the block diagram as necessary).
3. Design and test VHDL models of each unique component used in your datapath. **Use previously-designed components** from your library whenever possible (adders, multiplexers, etc.).
4. Create a **table** listing all control signals and the values of each control signal required for the instruction fetch cycle, and for the execution cycle for each instruction type.

**Major Datapath Components Likely to be Needed:**

- ❖ **ALU/Adder(s)**. These can be derived from the adder/subtractor of Project 1. For the sake of simplicity, the main ALU should not provide unnecessary functions.
- ❖ **Register file**: Design as a multi-port “memory array”. **DO NOT** instantiate individual registers!
- ❖ **Sign/zero extension logic**, as appropriate, for ALU inputs.
- ❖ **Program counter (PC)** – this should be a “simple register” (do incrementing outside of the PC).
- ❖ **Instruction register (IR)** – this should also be a “simple register”.
- ❖ **Data bus interface** (transceiver). The data bus should be of type *std\_logic\_vector* to support tristate operation and multiple drivers (CPU and Memory). Other CPU signals can be of type *std\_logic\_vector* or *bit\_vector* (or individual bits).
- ❖ Assorted **multiplexers** for data paths and register address inputs.

**Thoroughly simulate each new component individually, before inserting it into the datapath. Annotate and submit each simulation.**

Verify all required register transfers by applying control signals with force commands during simulation, as they would normally be applied by the control unit. **Use your control signal table from Note (4) above to design the datapath test, and show in the simulation where you verified each required register transfer for the CPU.** (If some register transfers are common to multiple instructions, you do not need to show them separately for every instruction – but it might be a good idea to do so anyway.)