RISC Processor Development Laboratories

David Andrews and Perry Alexander
Electrical and Computer Engineering Department
The University of Kansas
{dandrews,palexand}@eecs.ku.edu

Goal

The goal of this project is to walk you through the design and implementation of a 16-bit pipelined RISC microprocessor that follows *Computer Organization & Design*. The overall project is broken into seven laboratories that each can be easily accomplished in a two-week period. The laboratories have been designed to coincide and reinforce the concepts covered during the lectures. The strategy of the laboratories is to minimize the complexity of the overall CPU design by successively forming more complex components from earlier, smaller and simpler components. Using this strategy, you will reach the final pipelined CPU in an orderly and easily achievable fashion. This strategy also introduces a structured design approach in defining components and interfaces and an orderly integration of simpler components into more complex components. In general, the laboratories follow the bottom-up design approach followed in the textbook, starting with primitive components and then "gluing" them together into a more complex set of subsystems and systems. The laboratories were originally based on building the design under schematic capture and simulation. They have been redesigned and updated to allow you to specify the design using VHDL.

Complete RISC Microprocessor Specification

The complete instruction set architecture (ISA) for the RISC microprocessor is given in the table below.

Instruction	Pseudo description	Op_code 4 bits	Rs 4 bits	Rt 4 bits	Rd 4 bits
ADD R_d, R_s, R_t	R_d := R_s + R_t	2	0–15	0–15	0–15
SUB R_d, R_s, R_t	R_d := R_s – R_t	6	0–15	0–15	0–15
AND R_d, R_s, R_t	R_d := R_s "•" R_t	0	0–15	0–15	0–15
OR R_d, R_s, R_t	R_d := R_s "+" R_t	1	0–15	0–15	0–15
SLT R_d, R_s, R_t	if (R_s < R_t) R_d := 1 else R_d := 0	7	0–15	0–15	0–15
LW R_d, off(R_s)	R_d := M[off + R_s]	8	0–15	0–15	offset
SW R_d, off(R_s)	M[off + R_s] := R_r	A	0–15	0–15	offset
BNE R_s, R_t<offset>	if (R_s != R_t) pc := pc + off +4	E	0–15	0–15	offset
JMP <addr_off>	pc := pc:addr:0	F	12 bit offset		

The instruction set has been defined with sufficient number and types of instructions to allow you to write small, effective simulation and test programs. As shown in the table, all instructions are 16 bits in width. Offsets for loads and stores are 4-bit, 2s-complement numbers, and the absolute jump address is 12 bits. The jump address is computed as:

15:13	12 1	0
PC 15:13	addr_offset	0

The opcodes have been defined such that the lower three bits can be fed directly in the ALU as control lines. The ALU control lines and functions are shown below.

ALU Control Lines	Function
0 0 0	And
0 0 1	Or
0 1 0	Add
0 1 1	Subtract (beq)
1 1 1	Set-on-less-than

Hazard Avoidance

You should use the following two hazard avoidance techniques to simplify your design. These techniques should be guaranteed by the compiler or adhered to if hand assembling.

Avoidance Technique	Description
Load-Word Hazard Prevention	Guarantees that the next instruction will not use, as a source, the register being written by an immediately preceding load instruction
Conditional Branch Hazard	Guarantees that the three instructions following a conditional branch should always be executed

Laboratory Descriptions

The seven laboratories are listed below. For each laboratory, you are encouraged to always consider cost versus performance for your design. Understanding the basic cost versus performance trade-off familiar to practicing designers is an important aspect of the process. This may be your first opportunity to engage in a moderately complex design. It is emphasized that there are many solutions to any design, and your job is to balance cost versus performance in each design. To support this basic philosophy, you can calculate a simple figure of merit, *#gates* × *execution time*, for each project. If you are ambitious, this allows you to pursue more efficient or higher performance implementations. A percentage of each laboratory grade is made competitive based on the reported figure of merit. The objective of each laboratory is given in the table below.

Laboratory	Name	Description
1	VHDL Components	Implement familiar components in VHDL to be used throughout the semester
2	Instruction Interpreter	Implement and simulate a behavioral instruction interpreter for the given ISA
3	Register File / ALU	Implement and simulate behavioral and structural models of a register file and ALU
4	Multicycle CPU	Integrate the Register File / ALU with a structural model of a data path and a behavioral model of a multicycle controller to form a complete multicycle CPU
5	Pipelined CPU	Implement a pipelined version of the multicycle CPU
6	Data Hazard and Forwarding Unit	Implement and integrate a data hazard detection and forwarding unit into the pipelined CPU
7	Cache Module	Implement a cache for the CPU