ECE:3350 Spring 2020 -
Computer Architecture and Organization
Simple Instruction Set Computer (SISC) Project

General Information:

The project consists of building and testing, and modifying a Simple Instruction Set Computer (SISC) processor using Verilog HDL and ModelSim. The project also involves writing machine code programs that correctly execute on this computer. The starting code to be used in each part for this project is provided in zip files in ICON. The project is divided into 4 parts as described below.

Overview of the SISC Processor Architecture

This Simple Instruction Set Computer (SISC) is a multi-cycle RISC computer with separate memory for instructions and data, with the following characteristics:

Word length:	32 bit
Bit ordering:	little endian
General purpose registers:	16 x 32 bits
Program counter:	16 bits
Instruction/data space:	2^16 = 64 kilo-words = 64 KW
Addressing resolution:	32 bit word
Byte ordering:	little endian
Instruction set:	LOAD/STORE-architecture
Register operand length:	32 bit
Immediate operand length:	16 bit
Clock rate:	1 cycle / 10 ns
Cycles per instruction:	5 CPI
Addressing modes:	immediate
	register-direct
	register-indirect
	index
	absolute
	auto pre-increment
	auto post-decrement

NOTE: R0 is a pseudo register; bit bucket as destination, returns zero as source.

Phase 1: You are to implement the datapath for the processor and verify correct operation. The part 1 zip file contains the Verilog files for the various components you will need for the datapath. You are to create the top-level SISC.v file which will instantiate the various components of the datapath, and design the control unit that will provide the various control signals to the other components. Phase 1 is to only implement the register and immediate operand instructions and halt instruction. There is also a part 1 testbench file which will instantiate your top-level SISC.v file and provide it with the reset and clock signals as well as a set of simulated instructions. The test bench file also contains comments about the expected register file contents you should see when you run the simulation.

Phase 2: Starting with your phase 1 design, you are to add the instruction memory, program counter, branch logic and instruction register to the design. The control unit is modified to add instruction fetch to the overall operation and implement the jump and branch instructions. The phase 2 testbench only supplies the clock and reset signals to the top-level SISC.v file. The same instructions as in phase 1 are now contained in the instruction memory. The simulation results with respect to the final register file contents and status should be the same as in phase 1.

Phase 3: Starting with your phase 2 design, you are to add the data memory to the design. You are to modify the control unit to add the basic load (LDX and LDA) and store (SDX and STA) instructions. The phase 3 testbench is unchanged from phase 2 but the instruction memory is altered to add the load and store instructions. The simulation results with respect to the final register file contents and status should be the same as in phase 1 and 2 plus the addition of the load and store instruction results.

Phase 4: You are to implement the rest of the load and store instructions and the swap instruction in a method of your own design. The phase 4 testbench is unchanged from phase 2 but the instruction memory is altered to add the rest of the instructions. The simulation results with respect to the final register file contents and status should be the same as in phases 1-3 plus the addition of the phase 4 instruction results. Additionally, you are to write two simple programs, one to sort data and one to perform a 32-bit by 32-bit multiply. The data memory files for the programs is provided.