CS450/CS650 Winter 2018 – Coding Project 2

Downloads:

1. p_ref.zip: Reference files for p2 and p3. Contains:
    - mipsQuickRef – Two page summary of instruction set and registers.
    - mipsPRM – Instruction description and encodings (see chapter 3).
    - benchmarks - Seven benchmark programs: BubbleSort, CheckVowel, fact, SimpleAdd, SimpleIf, SumArray, and Swap. There are 5 files for each benchmark: *.c (original source code), *.sh (compile script), *.s (assembly code from generated by MIPS compiler), *.dmp (listing generated by MIPS compiler), and *.x (object code in hexadecimal format). The *.x files are used to populate the memory with the program. The *.dmp files are useful because they show the assembly language instruction and its binary encoded form side by side. The *.c, *.sh and *.s are not needed – just for information.
2. p2.zip: Source files for p2. Contains:
    - Makefile – Has two main targets: SimpleAdd (just console output) and SimpleAdd_wave (gtkwave output).
    - memory.sv – Memory module.
    - mips.sv – MIPS processor stub.
    - p2.out – Expected output for SimpleAdd make target.
    - p2_tb.sv – P2 testbench.
    - params.sv – Memory and processor initialization parameters.
    - regfile.sv – Register file module.
    - tb_SimpleAdd.gtkw – Waveform viewer configuration file.

(10 marks)

The MIPS processor stub module correctly implements the `addiu` instruction. Modify and extend it to correctly execute the other instructions found in the SimpleAdd testbench. You should only modify the `mips` module. *Submit the completed mips.sv file.*

The instructions needed to correctly execute the SimpleAdd testbench are can be found in `SimpleAdd.dmp`. By examining the instruction encoding, you will find that two of those instructions are actually pseudo-instructions that assemble to other instructions (that happen to also be used by their proper names elsewhere in the program).

You will need to copy `SimpleAdd.x` from the benchmark directory into the same directory as the p2 source code so that the memory can load it.

The `SimpleAdd_tb` testbench instantiates the `mips` module (which in turns instantiates the `regfile` module), and two copies of the `memory` module, one for instructions, and one for data. Both memories are initialized with the contents of `SimpleAdd.x`. Instantiating two copies of the memory simulates having an I-cache and a D-cache. Instruction execution starts at address 0x80020000 because that is the target address that the MIPS compiler defaulted to when generating the testbenches. The `mips` module takes 5 cycles to execute each instruction. The testbench shows any register writebacks or memory stores committed by each instruction in the testbench. Running the

testbench with the provided `mips` only generates correct output for the addiu instructions (or any equivalent pseudo-instructions). `p2.out` shows the expected output from the testbench.

Schematic Diagram