ARM Single-Cycle CPU

** Warning – this lab takes significantly longer than the previous two. Start EARLY **

Introduction: For this project you are to design a simple 64-bit ARM Single-Cycle CPU. The CPU instructions to be implemented are listed below. Chapter 4 will give some examples of how architectures are put together, and will be useful as you design your own CPU. For this CPU, you will use your two previous projects (the register file and the ALU) so you will need to have these fully functional before proceeding to work on your CPU. The data memory and instruction memory modules are provided in the files "datamem.sv" and "instructmem.sv" respectively. Depending on the instruction set for this quarter, you may also need some of the math units in "math.sv". We also provide a bunch of test programs – you can change the program loaded by editing the filename specified in "instructmem.sv".

You are responsible for coming up with the top-level testbench for this assignment – use previous labs' testbenches as guidance. Please remember that the rules from lab #1 are still in effect. Also, to demonstrate that your CPU actually works, you will need to set things up so that all the registers are displayed in modelsim, and that we can easily see the contents of the data memory.

Make sure that the clock in your testbench (1) is long enough so that all processing is done within this clock cycle (a VERY long clock is fine) (2) executes enough clock cycles for all of the programs to finish.

The control logic for your CPU can be done in RTL ("always_comb" and "always_ff" blocks).

Note that, although some of the benchmarks are particularly relevant to lab #4, they ALL will run successfully on lab #3, and you will be evaluated on whether your CPU works on any/all of them. Be sure to test ALL the benchmarks on your CPU.

Instruction set:

ADDI Rd, Rn, Imm12: Reg[Rd] = Reg[Rn] + ZeroExtend(Imm12).
ADDS Rd, Rn, Rm: Reg[Rd] = Reg[Rn] + Reg[Rm]. Set flags.
AND Rd, Rn, Rm: Reg[Rd] = Reg[Rn] & Reg[Rm].
B Imm26: PC = PC + SignExtend(Imm26 << 2).
    For lab #4 (only) this instr. has a delay slot.
B.LT Imm19: If (flags.negative != flags.overflow) PC = PC + SignExtend(Imm19<<2).
    For lab #4 (only) this instr. has a delay slot.
CBZ Rd, Imm19: If (Reg[Rd] == 0) PC = PC + SignExtend(Imm19<<2).
    For lab #4 (only) this instr. has a delay slot.
EOR Rd, Rn, Rm: Reg[Rd] = Reg[Rn] ^ Reg[Rm].
LDUR Rd, [Rn, #Imm9]: Reg[Rd] = Mem[Reg[Rn] + SignExtend(Imm9)].
    For lab #4 (only) the value in rd cannot be used in the next cycle.
LSR Rd, Rn, Shamt: Reg[Rd] = Reg[Rn] >> Shamt
STUR Rd, [Rn, #Imm9]: Mem[Reg[Rn] + SignExtend(Imm9)] = Reg[Rd].
SUBS Rd, Rn, Rm: Reg[Rd] = Reg[Rn] - Reg[Rm]. Set flags.

TURN-IN

For this lab you will electronically submit your CPU and demo the functionality of your CPU to the TAs.

Electronic submission requirements:

In ***addition*** to the Lab 1 & Lab 2 requirements, the following also apply:

1. Submit a wave file that illustrates all register contents, program counter, flags, data memory, clock and reset
2. instructmem.sv and datamem.sv should be part of your submission