ECE/CS 552: INTRODUCTION TO COMPUTER ARCHITECTURE

Project Description – Stage 2

Due on Wednesday, April 11, 2018, 23:55

In stage 2 of this project, the task is to design a 5-stage pipelined processor to implement Wisc S18 ISA.

Functional blocks of the single-cycle realization in stage 1 of this project should be reused if all possible.

The major work in stage 2 is the implementation of pipeline control. You will implement control blocks for (a) Hazard (both control and data) detection and mitigation (stall, flush, or forwarding), (b) control and data path modification for data forwarding, including register bypassing. More details are descripted below.

Either Modelsim or Icrarus should be used as the simulator to verify the design. **You are required to follow the Verilog rules as specified by the rules document uploaded on canvas. NOTE: the only exception to this rule is the required use of *inout* (tri-state logic) in the register file. Do not use *inout* anywhere else in your design.**

1. WISC-S18 ISA Summary

WISC-S18 contains a set of 16 instructions specified for a 16-bit data-path with load/store architecture.

The WISC-S18 memory is byte addressable, even though all accesses (instruction fetches, loads, stores) are restricted to halfword (2 byte), naturally-aligned accesses.

WISC-S18 has a register file, and a 3-bit FLAG register. The register file comprises sixteen 16-bit registers and has 2 read ports and 1 write port. Register $0 is hardwired to 0x0000. The FLAG register contains three bits: Zero (Z), Overflow (V), and Sign bit (N).

The list of instructions and their opcodes are summarized in Table 1 below. Please refer Phase 1 specs for specific details.

Table 1: Table of opcodes

Instruction	Opcode
ADD	0000
SUB	0001
RED	0010
XOR	0011
SLL	0100
SRA	0101
ROR	0110
PADDSB	0111
LW	1000
SW	1001
LHB	1010
LLB	1011
B	1100
BR	1101
PCS	1110
HLT	1111

2. Memory System

For this stage of the project, the memory modules to be used in the same as the Phase 1. The processor will have separate single-cycle instruction and data memory which are bye-addressable. The instruction memory has a 16-bit address input and a 16-bit data output. The data memory has a 16-bit address input, a 16-bit data input, a 16-bit data output, and a write enable signal. If the write signal is asserted, the memory will write the data input bits to the address specified by the address.

Verilog modules are provided for both memories.

The instruction memory contains the binary machine code instructions to be executed on your design.

3. Implementation

3.1 Pipelined Design

Your design must use **a five-stage pipeline** (IF, ID, EX, MEM, WB) similar to that in the text. The design will make use of Verilog modules you developed as part of the homework assignments during the semester along with the modules developed for Stage 1 of the project.

You must implement hazard detection so that your pipeline correctly handles all dependences. You are required to implement register bypassing and data forwarding. You need to handle both data hazards and control hazards. You are not required to implement optimizations to reduce branch delays such as branch prediction and branch delay slots.

It is highly recommended to make a table, listing for each instruction, the control signals needed at each pipeline stage. Also, make a table listing the input signals and output signals for data and control hazard detection and mitigation (stall and flush). The data detection unit should assume data forwarding and register bypassing is available.

3.2 Stall/Flush

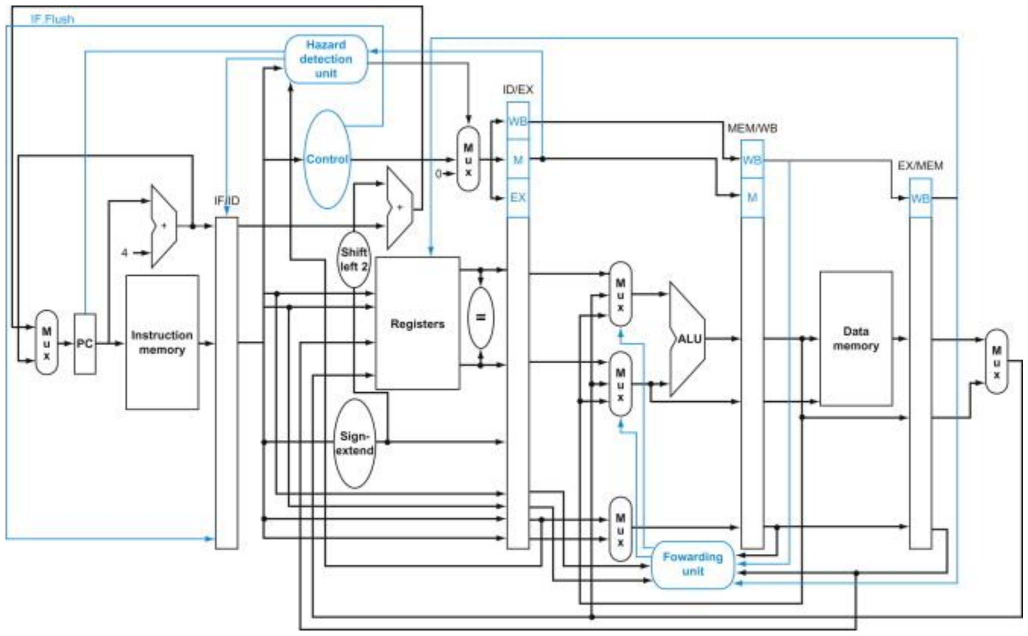
Stall is performed by using a global stall signal - which disables the write-enable signals to the D-FFs of the upstream pipeline registers (of the stall).
Flushes can be performed by converting the flushed operations into NOPs (i.e preventing them from performing register/memory writes and changes to any other processor state such as flags).

3.3 HLT Instruction

The HLT instruction should raise the 'halt' signal only when it reaches the writeback stage. The PC should not change after the HLT instruction unless the HLT is on a wrong branch path. In this scenario when the older branch instruction is resolved, the HLT and any other instructions will be flushed and you branch to the correct path.

3.4 Schematic

A summary diagram of the 5-stage pipeline is shown below:



Note: There are more detailed diagrams of each stage of the pipeline in your textbook (along with details of how hazard detection etc. is developed). You can refer them for building the pipelined stages of the project.

3.5 Reset Sequence

WISC-S18 has an active low reset input (rst_n). Instructions are executed when rst_n is high. If rst_n goes low for one clock cycle, the contents of the state of the machine is reset and starts execution at address 0x0000.

4. Interface

Your top level verilog should be file called ***cpu.v***. It should have a simple 4 signal interface: ***clk***, ***rst_n***, ***hlt*** and ***pc[15:0]***.

Signal Interface of ***cpu.v***		
Signal:	**Direction:**	**Description:**
clk	in	System clock
rst_n	in	Active low reset. A low on this signal resets the processor and causes execution to start at address 0x0000
hlt	out	When your processor encounters the HLT instruction it will assert this signal once it is finished processing the instruction prior to the HLT.
pc[15:0]	Out	PC value over the course of program execution.

5. Submission Requirements

1. You are provided with an assembler to convert your text-level test cases into machine level instructions. You will also be provided with a global test bench and test case. The test case should be run with the testbench and the output should be submitted for Phase 2 evaluation.

2. You are also required to submit a zipped file containing: all the Verilog files of your design, all testbenches used and any other support files.