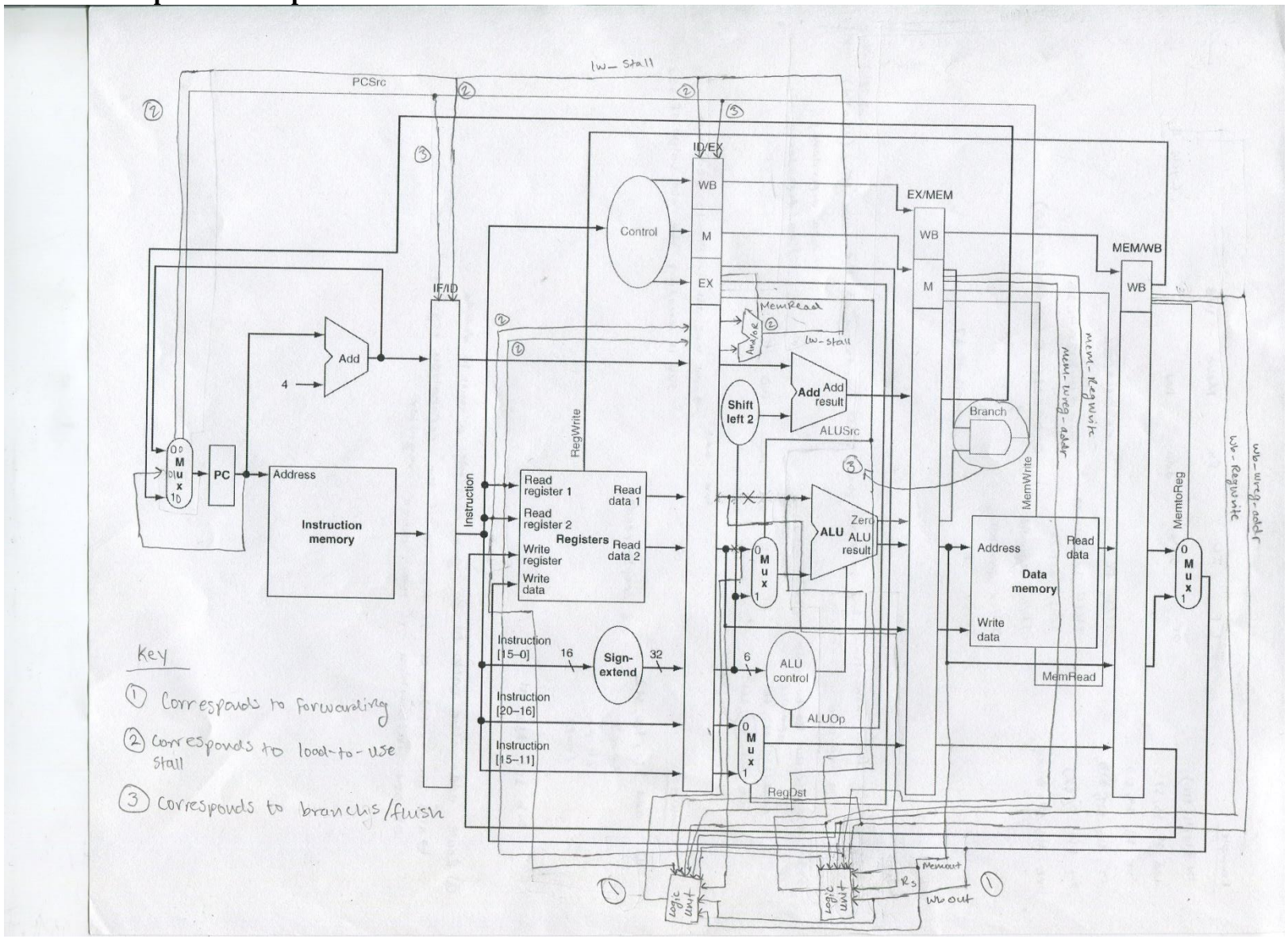
Sreeramamurthy Tripuramallu
903057502

ECE 3056 Assignment 4 Report

Part I: Labeled Pipelined Datapath



Sreeramamurthy Tripuramallu
903057502

Part II: Description of VHDL Modifications

1) ps_fetch:
    a. The lw_stall signal is a new input into fetch, and extra code was added to the Process section. When lw_stall is high, the PC values does not change
2) pipe_reg1:
    a. The lw_stall signal is a new input into pipe_reg1. When this signal is asserted, nothing happens, in other words the outputs do not change to the next instruction

    b. The flush signal is another new signal; when this signal is asserted, the values in pipe_reg1 become "flushed," as in the outputs become 0.
3) ps_decode:
    a. Since there is no write-back stage, the write-data is determined in decode. Initially, this signal was a local file, and was made into an output. In turn, a copy of this signal was made in order to continue using it for calculations in ps_decode stage.

    b. read_register_1_address_copy & read_register_2_address_copy are two other output signals, they are the address values of register Rs and Rt, they are Instruction[25 downto 21] and Instruction[20 downto 1]

4) pipe_reg2:
    a. Both id_read_register_1_address_copy & id_read_register_2_address_copy are two input signals originating from ps_decode. These signals are then set equal to the outputs ex_read_register_1_address_copy & ex_read_register_2_address_copy. These two outputs are then mapped into ps_execute.
    b. The other inputs are lw_stall and flush signals. lw_stall corresponds to the lw-to-use stall; when it is high the associated values of the instruction are 0 (inserting a stall cycle). Similarly, the flush has the same operation, but it is associated with branches.

5) ps_execute:
   There are many inputs/outputs in this stage as this is the primary stage operation
    a. The first set of signals is associated with forwarding. There are six added inputs: mem_out (the ALU result from mem), wb_out (the ALU result from write-back), read_register_1_address_copy & read_register_2_address_copy (the memory location of Rs and Rt), mem_wreg_addr & wb_wreg_addr (the write-registers associated with mem and write-back), mem_RegWrite & wb_RegWrite (RegWrite signals from the previous two instructions). Using these signals, extensive logic is applied to determine the proper values of Ainput (Rs, mem_out, or wb_out) and Binput (Rt, mem_out, wb_out).

    b. The second set of signals are associated with the load-to-use stall. There are 3 added inputs: id_read_register_1_address_copy &

id_read_register_2_address_copy (the memory location of Rs and Rt of the next instruction), and MemRead (MemRead signal of the current instruction) The signal output is the lw_stall signal which evaluates the three inputs to determine if a stall cycle is necessary.

c. The third set of signals are associated with branch control and flushes. Thus, there is one input, the Branch signal, as well as one output, the PCSource signal. Logic is applied to evaluate the Branch signal as well as subtraction of Ainput and Binput to determine if a branch is necessary. Also, the calculation to the BranchPC calculation is modified (subtracted by 4) in order to calculate the appropriate target address.

6) spim_pipe:
All of the new and necessary signals are connected through the top-level spim_pipe file. All of the new signals described above are created in the various ports and mapped across the necessary port maps.
- ps_fetch: lw_stall is mapped
- ps_decode: write_data, read_register_1_address_copy & read_register_2_address_copy are mapped
- ps_execute: all signals necessary for forwarding, load-to-use stalling, and branch flushing are mapped
- pipe_reg1 & pipe_reg3: lw_stall and flush signals are mapped

Part III: Sequence of Steps Describing Operation/Functionality

1) The first functionality implemented is data forwarding. Instead of creating forwarding multiplexers, the logic for Ainput and Binput (in ps_execute) was modified. Ainput selects between mem_out, wb_out, and register Rs values. Ainput selects mem_out when the mem_RegWrite signals is asserted and the mem write register matches register 1. Similarly, Ainput becomes wb_out when the wb_RegWrite signal is asserted and the write-back write register matches the register 1. Additional logic is added to select wb_out. This logic is present to account for situations when there are two data dependences consecutively. In this cause, mem_out is the selected instead of wb_out. If neither mem_out nor wb_out is necessary, then the contents of Rs is selected.

   Similarly, when ALUSrc is 0, Binput is either mem_out, wb_out, or register Rt. The same logic is applied to select between the three values (however ALUSrc has to 0 in order for one of these values to be selected). Overall, data forwarding uses the various input signals from mem and wb to determine the appropriate values for Ainput and Binput.

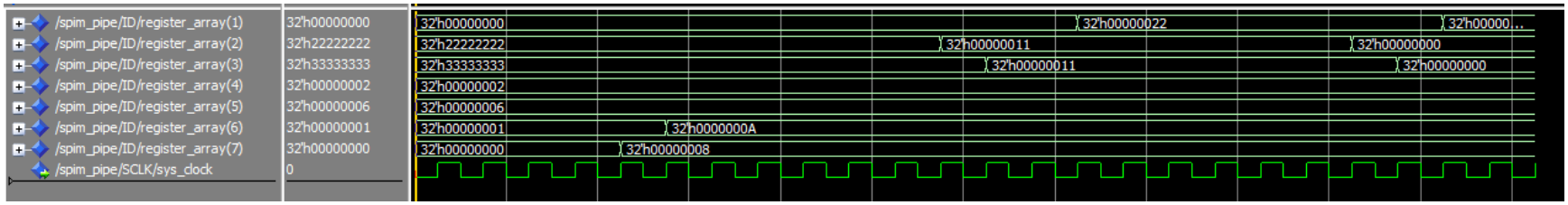
2) The second set of functionality implemented is load-to-use stall. The lw_stall is asserted if a stall cycle is necessary. It is determined in ps_execute by checking if MemRead is asserted and if the write register of the current instruction matches the write register of the next instruction. If so, a stall is necessary and lw_stall becomes 1. This signal is then sent to fetch to ensure that PC remains PC (and not PC+4), pipe_reg1 to make sure that the instruction does not change (thus nothing is supposed to happen) and the instruction

remains in pipe_reg1, and to pipe_reg2 in order to clear the instruction and all of its associated values. In turn, this properly implements load-to-use stalls by inserting proper stall signals when necessary.

3) The final functionality implemented is branching. Inside ps_execute, PCSource becomes an output and is asserted when the Branch signal is 1 (branch is an input) and if the Ainput – Binput is 0 (which means Ainput and Binput are the same). Also, the branch target address is subtracted by 4 in order to result to the right target address. When PCSource is 1, inside of ps_fetch, the new PC value is the branch target address. The PCSource signal is also connected to a flush signal which is sent to pipe_reg1 and pipe_reg2 in order to clear the instructions in the registers. Overall, this logic properly implements branches and flushing. 0

Sreeramamurthy Tripuramallu
903057502

Part IV: Screen Trace of Provided Test Program



The following is a screen trace of the provided test code. Initially, $2 and $3 are 22222222 and 33333333 respectively; these are the original values of the registers. Once lw instructions are executed, the values change to be 11 (hex for 17). The final values for $1 is 22 (hex for 34) at cycle 14 and $6 is 0A (hex for 10) at cycle 5. Overall, the CPI for the unmodified-baseline pipeline was 1.26 (29 cycles/23 instructions), and the CPI for the modified pipeline is 1.04 (24 cycles/23 instructions); thus there is an improvement once modifications were made.