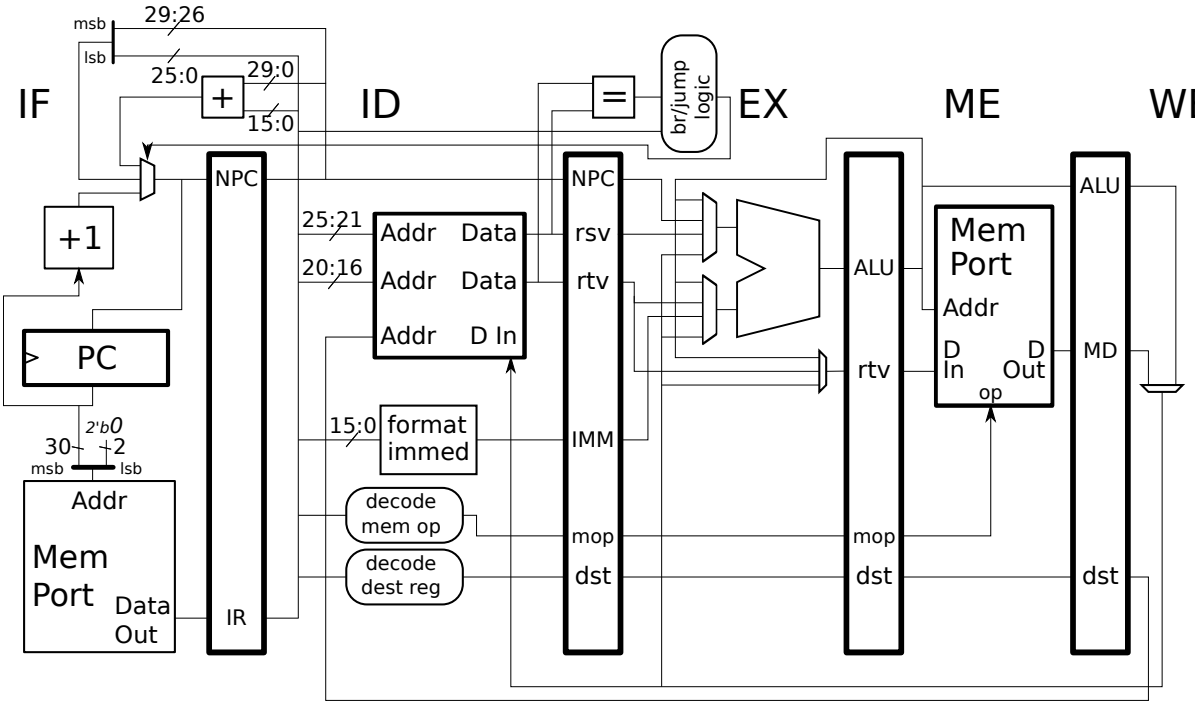
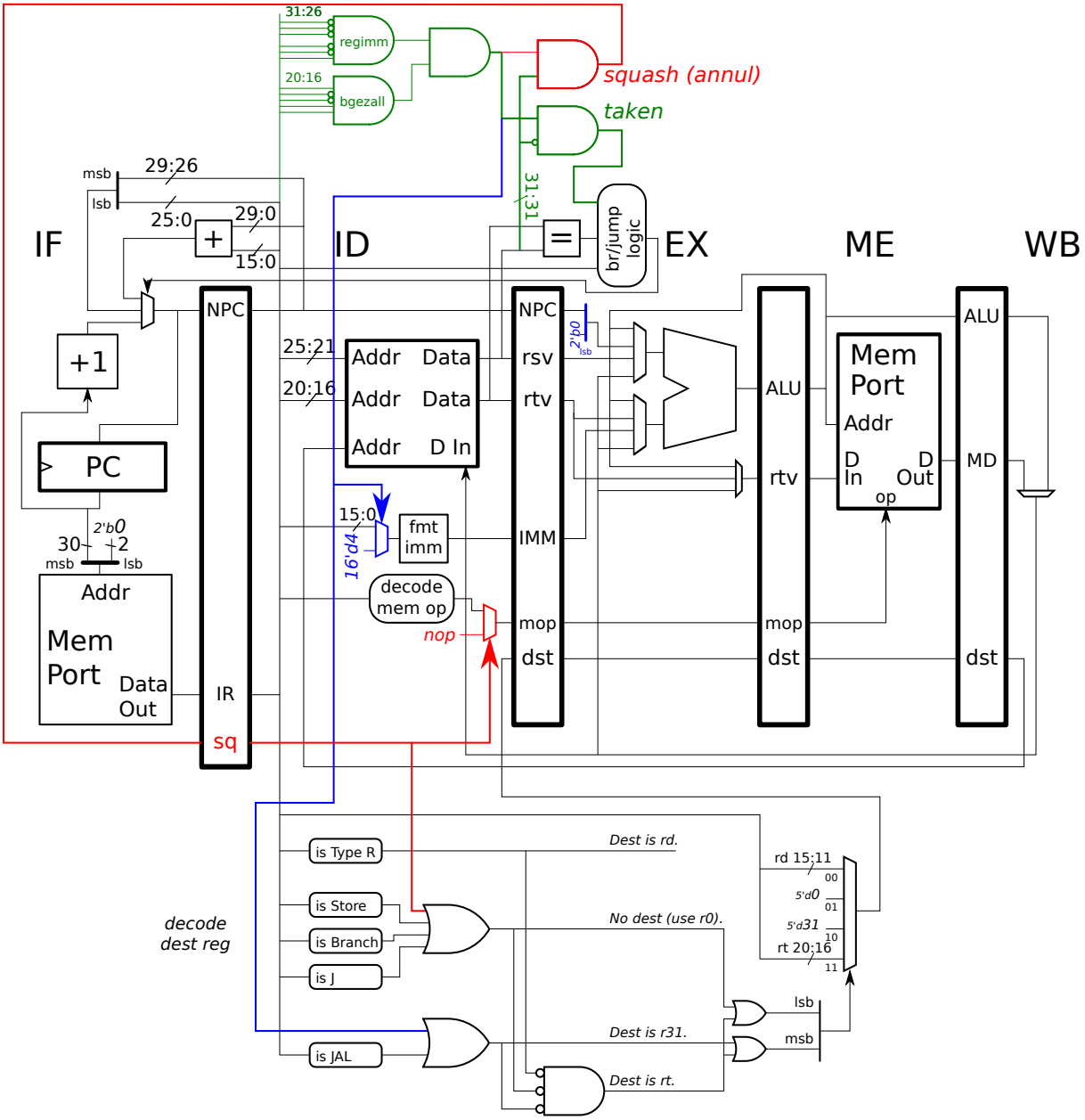
LSU EE 4720 **Homework 3** Solution **Due: 24 February 2017**

To help in solving this problem it might be useful to study the solutions to the following problems which involve hardware implementing branches in the statically scheduled five-stage MIPS implementation we've been working with: Spring 2016 Homework 3 (SPARC-like branch instruction), Spring 2015 Homework 2 Problem 2 (use reg bits for larger displacement) and Problem 3 (logic for IF-stage mux), Spring 2015 Homework 3 Problem 2 (implement bgt, but resolve it in EX), Spring 2011 Final Exam Problem 1 (resolve in ME, with bypass).

Problem 1: Modify the implementation below so that it implements the MIPS II `bgezall` instruction, see the subproblems for details on the hardware to be designed. See the MIPS ISA documentation linked to the course Web page for a description of the `bgezall` instruction. An Inkscape SVG version of the illustration below can be found at `http://www.ece.lsu.edu/ee4720/2017/hw03-p1.svg`. The illustration also appears on the next page.



Solution on next page.



(*a*) Design control logic to detect the instruction and connect its output to the br/jump logic cloud. The control logic should consist of basic gates, **not** a box like `bgezall`.

Solution appears above in green. The green logic also computes a taken signal, and it is that which is connected to the br/jump logic. Note that the ≥ 0 condition is true when the rs value is not negative, which can be inexpensively and quickly checked by looking at bit 31 of `rsv`.

(*b*) Design the control logic to squash the delay slot instruction when `bgezall` is not taken. The control logic should squash the delay slot instruction by changing its destination register and memory operation. Be sure that the control logic squashes the correct instruction, and does so only when `bgezall` is not taken. Do not rely on magic clouds [tm].

Solution appears above in red. The squash signal is asserted when there is a `bgezall` instruction in ID *and* when it is not taken. (A common mistake was to ignore whether or not there was a `bgezall` in ID when generating the

squash signal.) The **squash** signal is put in a new pipeline latch where it joins other signals related to the instruction in IF. In the next cycle the **sq** signal is used to replace the memory op with a **nop** memory operation (the name **nop** is whatever the no memory operation code is called). The **sq** signal also selects register zero as an output register. In the solution the contents of the decode dest reg logic block is shown, which was not necessary in a submitted solution.

(*c*) Add datapath or make other changes needed to compute the return address. Note that NPC is already connected to the ALU. Consider inexpensive ways to compute the second operand. (Adding a 32-bit ID/EX pipeline latch is not considered inexpensive for this problem.)

Solution appears below (and above) in blue. As described in the ISA manual, the return address is PC+8 or NPC+4. In ID the immediate is replaced with the constant 4. This is done before the fmt imm logic so that a 16-bit mux rather than a 32-bit mux can be used. Also, the decode dest logic is modified so that 31 is used as a destination when **bgezall** is in ID. When **bgezall** is in EX the ALU will be used to add NPC and the 4.