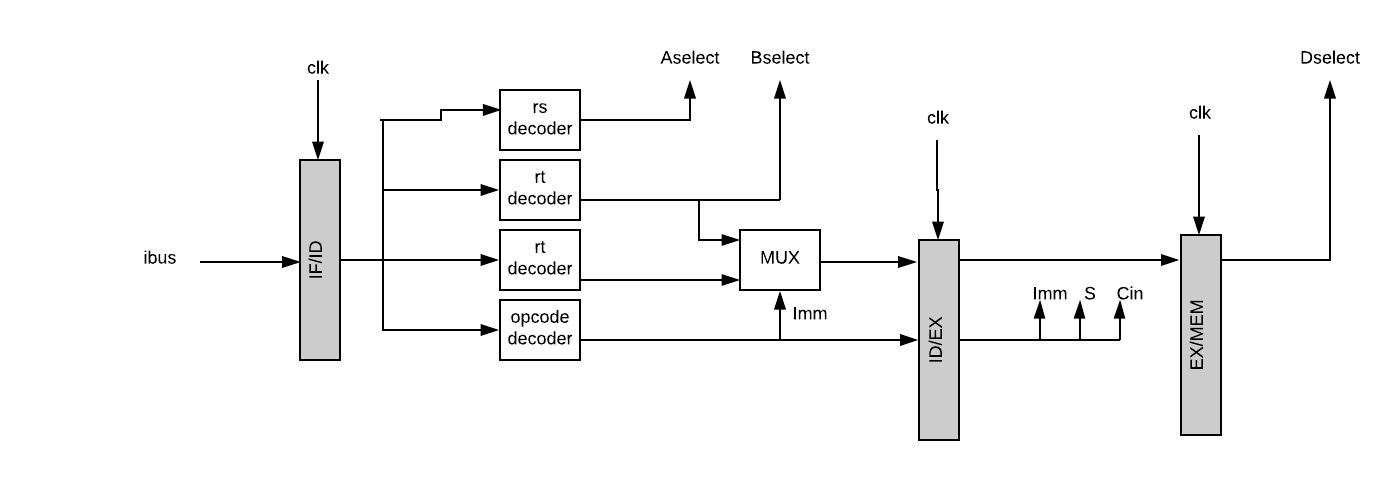
Arjun Gupta

EECE 3324

Lab 4

7/16/2018

1. Description
   1. The main purpose of this lab was to design a controller for a 32-bit pipelined CPU so that both R type and I type instructions were able to be read in from an instruction bus. The controller was designed using three stages, an IF/ID, ID/EX and EX/MEM stages specifically. The main point of the controller is to provide certain control signals to the CPU to use in both the register file and the ALU. This implementation will be completed in a subsequent lab.
2. Block Diagram
   1. Below shows the block diagram of the controller created in lab 4.
3. Steps needed to complete the Lab
   1. The first step to complete this lab was to use a binary decoder to obtain the Aselect, Bselect and Dselect register values in a 32-bit number. The values were obtained from the corresponding 5-bit numbers according to the MIPS reference card. Next, since I type and R type instructions use the rd and rt register differently, a multiplexor had to be created to switch between the rd register when using an R type instruction and the rt register when using an I type instruction. The signal to control the Mux was obtained from the logical opcode decoder. This decoder looked at the opcode and funct subsets of the instruction and determined if it was an I type or an R type instruction. If an I type instruction was discovered, then the control signal “Imm” was set to 1, signifiying that the multiplexor should choose the “rt” register. The Cin and S values were also able to be calculated from the opcode and funct registers as well. Between each stage, a D Flip-Flop allowed the creation of the multi-cycle pipeline to enable each stage to be used once the registers are populated.
4. Lessons Learned
   1. Throughout this lab, I learned how to create this controller, and learned more debugging skills. One issue I ran into was that the Dselect output was an entire clock cycle ahead of what it should have been, which gave me many issues. To solve this, I was able to use non-blocking statements between the D Flip Flops to ensure that all the data was able to be passed through during one rising edge rather than being split up on synthetization.