Notes and Tips for CMPEN331 LAB5

LAB 5:
WRITEBACK

SLIDES COMPILED BY
ANDREW JONES

Welcome, Students!

- If you are a student of CMPEN331 going through these slides, welcome!
- If you haven't already, **please go watch the video.** These slides are meant to supplement the video, which contains information that may not be present in these slides.
- Feel free to use these slides as reference as you work on your Lab 5.

Some Starting Notes

- As always, start early!
  - Lab 5 is a lot shorter and somewhat easier than Lab 4, given your Lab 4 is implemented properly
  - Regardless, you should start early in case you run into issues
- This video assumes you have completed Lab 4
  - If you have not already completed Lab 4, You need to do so before starting Lab 5

What exactly are we doing in labs 3 through 5 + final project?

Labs 3 through 5, along with the final project, consists of implementing different stages/features of our five stage MIPS pipeline. The main structure of the five stages will be implemented through the labs, and the final polish and functionality will be implemented in the final project. There is also a bonus for the final project that implements additional instructions.

The bonus on final project is extra credit, I highly recommend that all of you try the bonus if you have time.

- Lab 3: Stages 1 and 2 (Instruction Fetch, Instruction Decode)
- Lab 4: Stages 3 and 4 (Execution, Memory Access)
- Lab 5: Stage 5 (Writeback)
- Final Project: Forwarding Behavior
- Final Project Bonus: Control Transfer Instruction Implementation (branch/jump instructions).

Overview of Lab 5

WRITEBACK (WB)

- Select whether we want to write the data memory output or the ALU output to the register file
- Executes the write into the register file
- That’s it :)

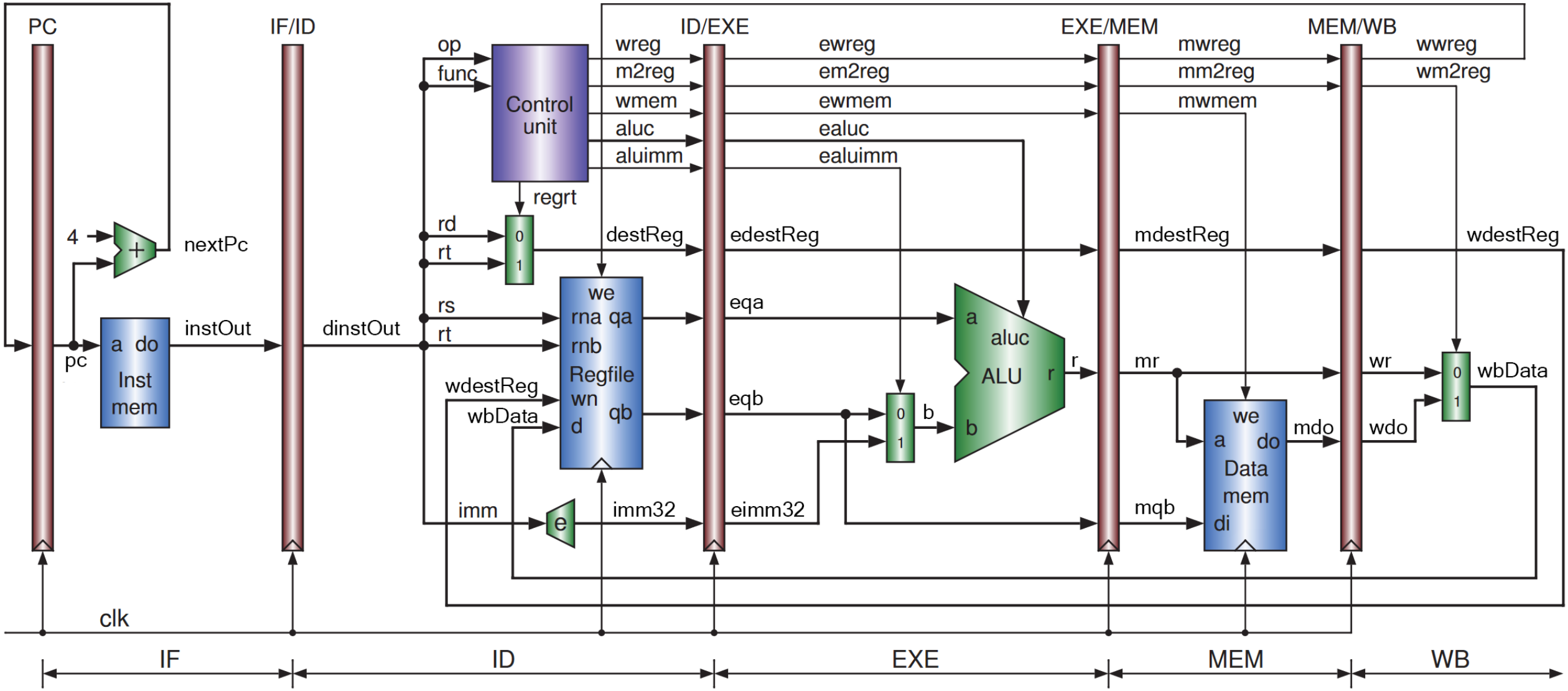
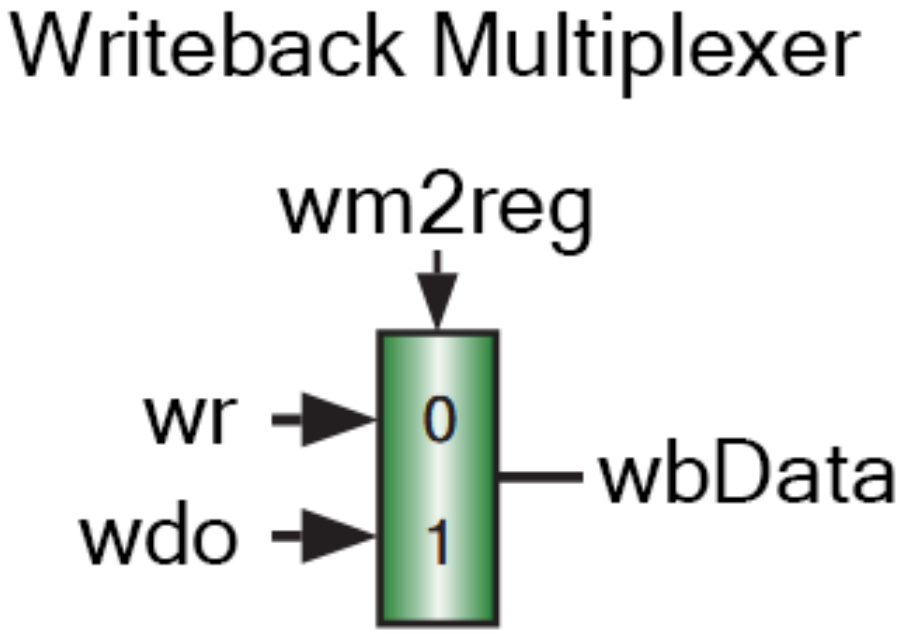


Diagram for Lab 5



WbMux

Inputs:

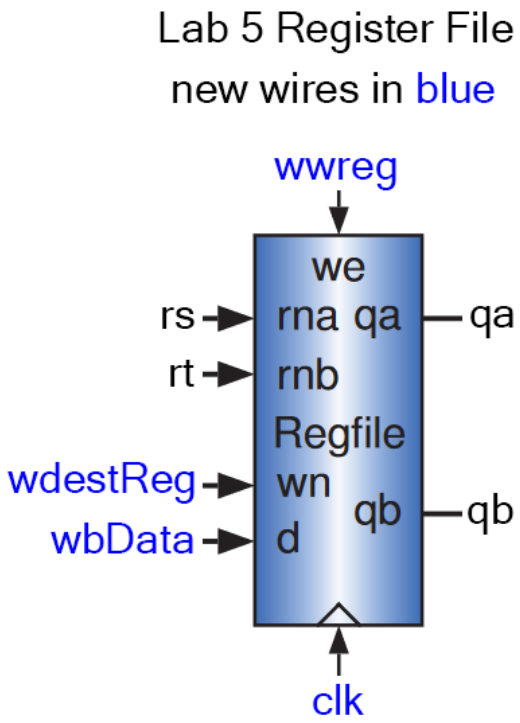
- wr [32 bits]
- wdo [32 bits]
- wm2reg [1 bit]

Output:

- wbData [32 bits]

Functionality: on any signal change

- If wm2reg is 0, wbData is set to value of wr
- If wm2reg is 1, wbData is set to value of wdo



Updated Regfile

New Inputs:

- wdestReg [5 bits]
- wbData [32 bits]
- wwreg [1 bit]
- clk [1 bit]

No New Outputs

All registers should still be initialized to zero

New Functionality: at negative edge of clk:

- If wwreg is 1, register at position wdestReg is set to wbData

Changes to modules from Lab 4

- Instruction Memory
  - Instruction Memory will have 1 new instruction in it along with the 4 instructions from Lab 4.
    - Note: this instruction is an ADD instruction instead of a LW instruction. If you didn't already implement the functionality for it in your control unit, you will need to do so for this lab.
- Datapath
  - Your Datapath will instantiate the wbMux described earlier, pass in the new inputs for the register file, as well as hook up the necessary wires to them.
  - Your Datapath will have the same inputs and outputs from lab 4, which are all the outputs from:
    - Program Counter
    - IF/ID Pipeline Register
    - ID/EXE Pipeline Register
    - EXE/MEM Pipeline Register
    - MEM/WB Pipeline Register
- You do not need to make any changes to your Testbench.

Final Remarks and Tips

- **I HIGHLY recommend putting different modules into different files.**
  - Yes, the lab doc tips say that you should put all non-simulation modules into a single file, but I've found that students who do use different files for modules spend less time debugging, and have an easier time finding modules when they need to.
  - Accessing the modules may require an extra menu, but from my experience it's much faster than scrolling up and down all of your code looking for the correct module.
- **Format your code.**
  - Proper indentation, formatting, and commenting of your code will make your life much, much easier when you enter the debugging stage of the lab.
  - Students who don't format and comment their code tend to spend a lot more time debugging than students who do format and comment.
  - How you format and comment your code is up to you, do what works best for you.
  - Remember that you will have to look at the code you write while debugging, not to mention the future labs and final project, so make sure it is easy for you to read and understand what's going on in your code.
- **Give everything names that make sense**
  - This fits in with the Format your code section.
  - The names of your modules, wires, and registers should make it fairly clear what they are used for or what they are trying to represent.
  - Naming registers reg1, reg2, etc. is ambiguous and will make debugging very difficult.
  - How you name your modules is completely up to you (with the exception of the testbench), but I recommend using the names within the provided diagrams.
    - Remember that with the testbench, your wires should very clearly represent what each signal is.
  - I also recommend defining module names with their intended stage of the pipeline as a prefix
    - For example, I would define the program counter module as "IF_ProgramCounter" and the regrt multiplexer as "ID_RegrtMux" with IF and ID corresponding to the correct stage of the pipeline.
  - Again, how you name everything is up to you, do what works best for you.

Final Remarks and Tips

- **Start Early!!!**
  - I know I put this tip at the beginning, but I cannot stress enough how important it is to give yourself enough time to complete the labs.
  - I recommend at minimum starting a week in advance of the due date, this will give you plenty of time to write your code, debug it, and access office hours.
- **Utilize Office Hours**
  - The office hours of TA's and LA's are provided on the syllabus.
  - TA's and LA's will gladly help with all aspects of the lab.
  - Note: Before asking TA's and LA's to debug code, please try and debug it yourself (remember, Follow The Data!)
  - If you can't access office hours for whatever reason, you can still email TA's and LA's through Canvas or their PSU email and they will help you through there
    - TA and LA emails are also provided on the syllabus.
    - Please provide lots of detail about what's causing you trouble in the lab, screenshots and Verilog code files help greatly with this!