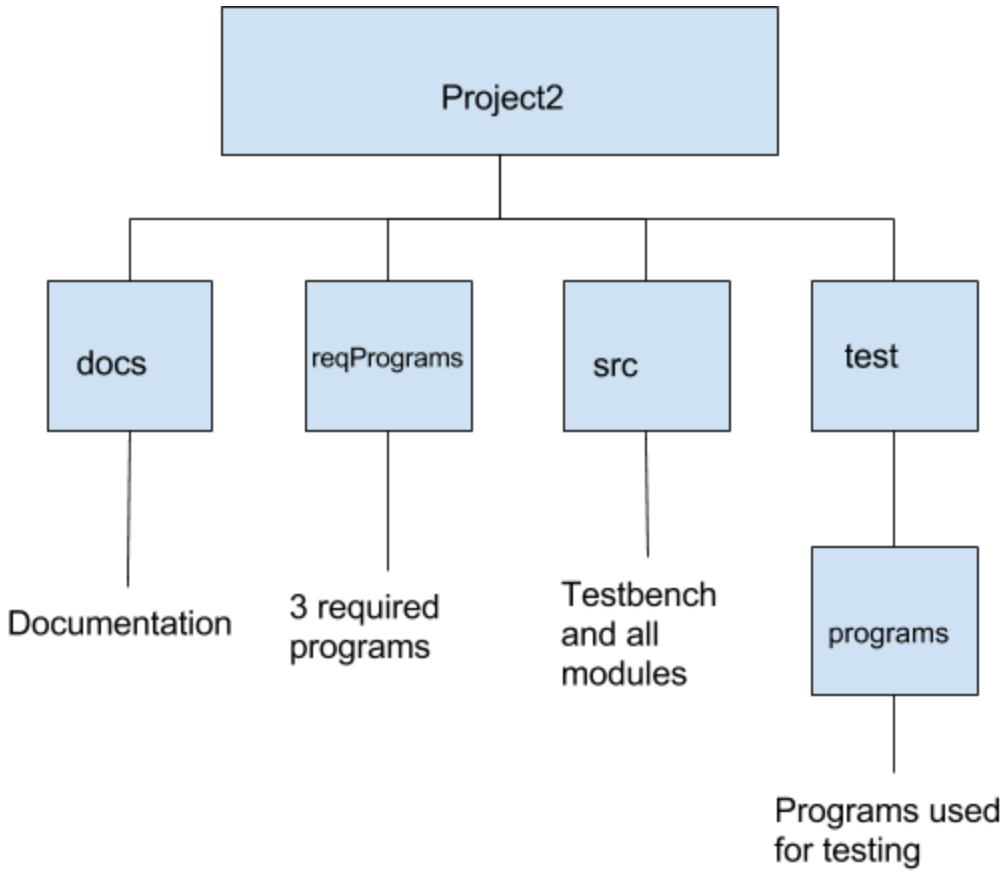
Assignment: Project 2
Course: CSCI 320 - Computer Architecture (Fall '17)
Professor: Alan Marchiori
Team: Verilads - Levi A, Andre A, Stefano C, Tom F

File Structure



Run:
To run the testbench, do the following:

navigate to project2 -> src
$ iverilog testbench.v
$ ./a.out

Compile:
To change the assembly program for testbench to run:

Navigate to project2 -> reqProgram
Open Makefile
Change value of ASMSOURCE to the name of the .s file you want to use
$ make // this should make a .v file for later use, with the same name as the .s file
Navigate to project2 -> src
Open instructionMemory.v
Change input for readmemh to the new .v file created by the makefile

Design:
The processor was designed by creating each pipeline stage and then connecting all the stages together. In the testbench, we have wires organized by which stage they are in, all of the modules are organized by stage as well. We decided to use a modular design when designing the different parts of the processor. Every file, except for the utilities file and the register file, is only responsible for one set of functionality. The utilities file hold the modules for the different types of multiplexers, sign extenders, d latch, and the jump shifter. The register file holds syscall becasue they are closely related. We decided to both hazard detect and the appropriate forwarding or stalling in one hazard detection unit. This was done for ease of organization in the testbench. Also it makes sense for all hazard detection functionality to be in one module so there is not the chance of incorrect wiring that could create problems. Another design choice that was made was to handle the branching/jumping and addressing in one module in the decode stage.

Testing:
Since many of the components that were used in the pipelined processor were also used in the single cycle processor, they were not directly tested in this project. However, as a baseline we used the add test program from the project 1 as a test program to make sure the processor still worked once we implemented the new modules. Along the way, we made tests for the three different types of forwarding supported by our processor (MEM/EX, MEM/MEM, EX/EX) and for stalling. In addition to this we also made tests for different kinds of jumping and branching. Lastly, we also made tests for our data memory module as well.Since we were building our processor to specific requirements we did mostly functional testing, but there was also unit testing.