Northwestern University
CompEng 361 - Fall 2025
Lab 4 - Pipelined CPU

In this project, you will work in groups of two to design a five stage pipelined RISC-V CPU which implements the majority of the RV32IM Instruction Set. Specifically, you must implement the same instructions required by the previous lab. The easiest way to do this is to extend your design from Lab 3 to create the pipelined design.

Your pipelined design should be capable of correctly running any unmodified RV32I program from the previous assignment. Specifically, you should not need to add nops to ensure correct operation. Your pipeline must correctly handle all hazards with a combination of stalls, forwarding, and flushes as appropriate.

One small difference from the previous lab is that you should assume that all multiply/divide instructions are long latency operations and will take many cycles to complete their execution. More precisely, you should build your pipeline control such that when one of these instructions enters the EX stage, it always stalls any following instructions and stays in the execute stage until it has produced its result. Specifically, you should follow this latency table:

Instruction	**Execution Latency (in EX)**
MUL, MULH, MULHSU, MULHU	4
DIV, DIVU, REM, REMU	20
Any other instruction	1

To be clear, you do not need to implement the computation any differently than you did in the previous lab assignment (e.g. you can still use *, /, % operators). We're assuming that someone else will design a proper multicycle multiply/divide unit. You should focus on the control here.

The pipelined processor will be implemented in Verilog (structural/continuous assign as in the last assignment) and must have the following interface and port list:

```
module PipelinedCPU(halt, clk, rst);
      output halt;
      input clk, rst;
```

An included template file has a register file, pipeline registers, and memory components that you should use. Note: There are some small changes between this and the previous assignment – especially in the testbench. You should adapt your design to incorporate the new components. Try to balance the pipeline stages so that your design would have the maximum possible clock rate. A few additional notes:

- Please do NOT change the interface to the module. It must not deviate from what is posted above.

- Your solution should be able to compile and run correctly with unmodified testbench and library files.
- Your solution MUST be entirely in Verilog (no Chisel or System Verilog)
- Your solution should be self-contained in a single Verilog source file without use of any external source files beyond the ones supplied.
- You should use only structural code to implement your design.

You must devise your own testing programs. Make sure the module is thoroughly tested. Try to think of corner cases and make sure they are appropriately handled.

You should turn in a single Verilog file with the following format:

```
<group-name>_lab4.v
```

Do not include testbenches, library files, test programs, or other supporting files.

Note that this is a group assignment. Turn in one submission per group.