ECE 469 Microprocessor System Design
Lab 3
ARM Pipelined Processor

Lab Objectives:

1. Develop and understand control and datapath flow in multistage systems.
2. Understand the data and control hazards experienced in pipeline feedback.
3. Implement solutions to issues generated by hazards such as forwarding, branch acceleration and prediction and stalling.

Big Picture:

One of the major projects within this course is the design and simulation of a simple pipelined processor. Lab 1 had you build the two integral components of the processor. Lab 2 had you link these components, as well as some additional logic to create a complete single cycle processor. Lab 3 will have you add pipelining to your processor and resolve the issues that come with the performance enhancement. **This lab is much more involved than its predecessors and its highly recommended that you start early!**

Introduction:

For this lab, you will be building upon the single cycle processor developed in the previous assignment. You will then load a test program that will confirm that the system works. By the end of this lab, you should be comfortable pipelining logical systems and dealing with the issues associated with them.

Pipelined Processor:

Before starting this lab, you should already be very familiar with both the single cycle and pipelined processer. To start with your processor will use the same code as you used in the previous lab along with your own changes and additions. The processor will be a 5-stage pipelining, meaning there will be 4 pipeline registers total. The stages are denoted as Fetch, Decode, Execute, Memory and Writeback. This processor will need to be able to execute each of instructions that were developed for the single cycle processor and deal with the consequences of any hazards that come up. To help guide your design, the primary considerations are listed below:

- There must be 5 pipeline stages.
- Your register file should be able to write in data coming from the fetch pipeline register and read out that same data before it is captured in the decode pipeline register.
- You should assume branches are NOT taken unless proved otherwise in the execute stage.
- Forwarding:
  - You should be able to forward from the memory or writeback stages whenever necessary.

        - For which instructions should you consider forwarding and for which instructions should forwarding be ignored?
        - Are there instructions from which you might be able to forward from one stage but not another?
        - Does stalling play into whether data is forwarded?
    - You should be able to determine whether to and forward branch addresses from the execute stage whenever necessary.
- Stalls:
    - You should be able to stall instructions when forwarding or a branch is taken.
    - When stalling, the pipeline stages mustn't be able to change the processors state (flag reg, memory, reg file etc.)

The pipelined processor schematic is given at the end of this lab assignment for your convenience.

Testing the Pipelined Processor:

As discussed before, attempting to just run a program or simulation until the system works is not productive nor efficient. Working out your expectation prior to running a simulation is a core debugging tactic and will help you understand how the pipelining, data forwarding and branching work. However, once you are confident in your design use the program given below to verify that your system all the issues produced by adding pipelining to a design are correctly resolved. For this lab a new program has been developed, to test not only the instructions but also insures forwarding and stalls work as expected, and is given below:

```
Main      SUB  R0 R15 R15
          ADD  R1 R0 #1
          ORR  R2 R0 R1
          ADD  R2 R0 #2
          SUBS R0 R2 #0
          BEQ  TAG1
          AND  R2 R2 R0
          AND  R1 R2 R0
TAG1      ADD  R9 R1 R0
          STR  R9 [R0, #9]
          LDR  R3 [R0, #9]
          AND  R2 R3 R2
```

However, this test is limited in the number and typed of instructions it uses and you are encouraged to verify your processor by creating and loading your own programs. These can be whatever length you like but should be included in the report if used. The previously used programs should also work with the pipelined processor.

Deliverables:

Please turn in the following:

1. A detailed lab report that includes the procedure and results, as well as an Appendix with the updated arm.sv, your alu.sv, your reg_file.sv as well as any other files you added or used to simulate the processor outside of the given files. The report should also include:
   - The simulation waveforms embedded into the results sections. **These screenshots must indicate**:
     i. An example of forwarding from the memory stage to the execute stage.
     ii. An example of forwarding from the writeback stage to the execute stage.
     iii. An example of stalling for a memory instruction.
     iv. An example of flushing for a branch instruction.

     These should each be accompanied by an explanation of why the situation occurred and how your processor handled it.
   - Any additional programs used to test the processor beyond those provided.
   - Each of the CHANGED design files, most likely arm.sv, as well as any others you may have made and used pasted and formatted in the report Appendix.
   - Follow the report format as outlined in "lab report outline" and "lab report template" documents on canvas.
2. Each of the CHANGED design files, most likely arm.sv, as well as any others you may have made and used, submitted as distinct files into canvas.

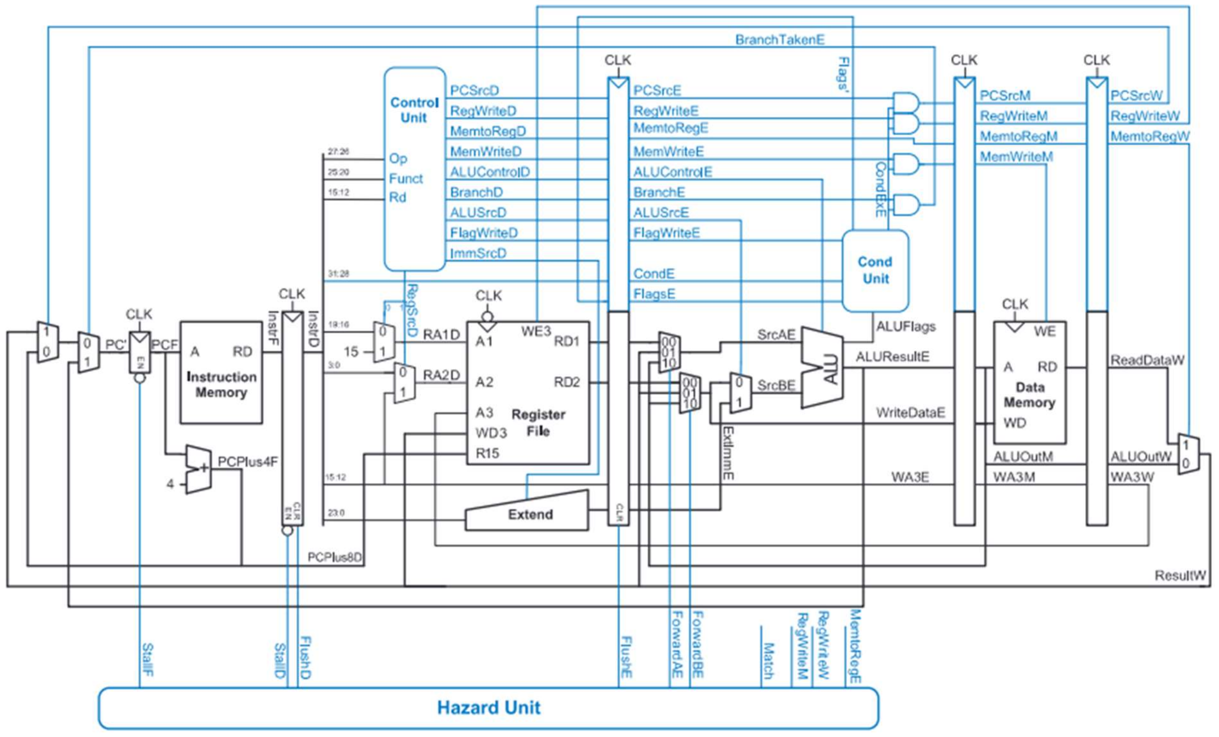

Figure 1: ARM 5 Stage Pipelined Processor