RISC-V Based CPU Design with Logisim [Part 7]

2018-03-29 | RISC-V_CPU | 0 | Page Views:

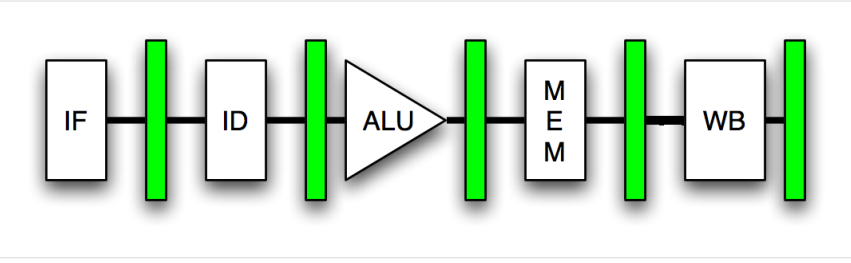
> PipeLining

7. Pipelining

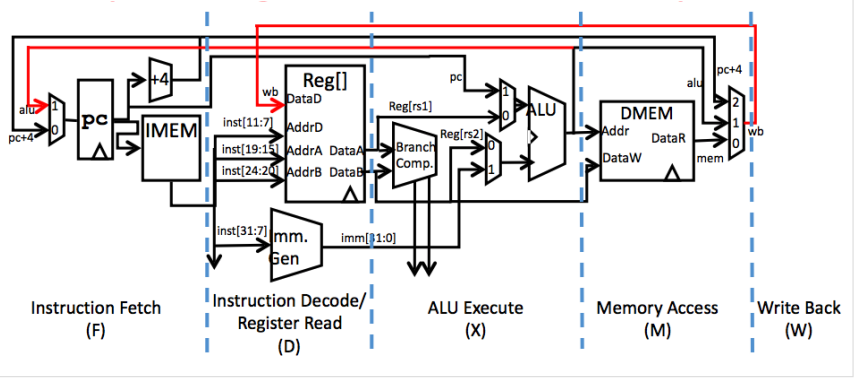
7.1 Introduction

> A pipeline is a set of data processing elements connected in series, where the output of one element is the input of the next one.

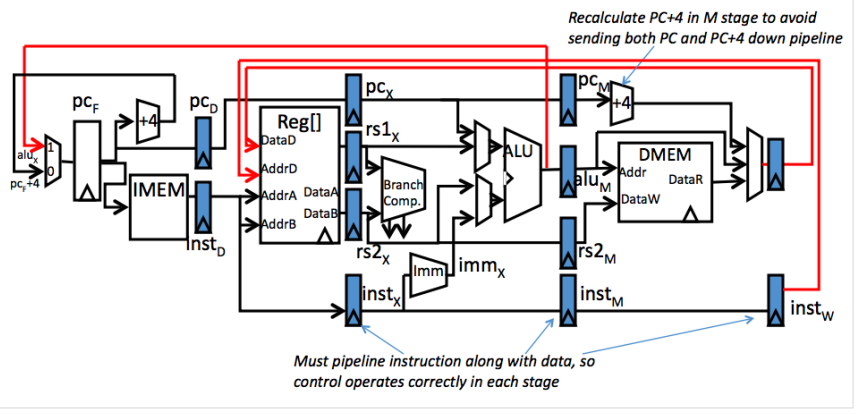
Let's first review our processing structure of our CPU. We can see that we have a structure like this:



Or we can show them in our designing map:

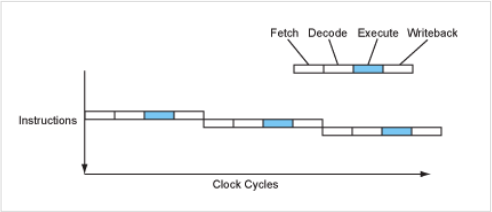


Then let's compare it with our Datapath structure for Pipelined structure:

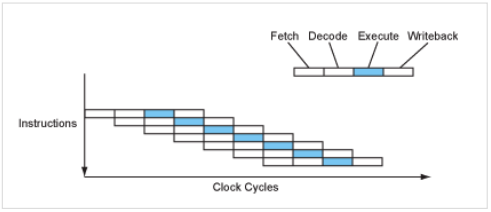


Did you notice some difference with the structure above comparing to our Single-Cycle CPU? Yes! This is an Datapath structure for Pipelined Datapath. Then what is pipelining and what does pipelining do?

You may notice that we are processing the instructions step-by-step. When there're many instructions, we process like this:



However, what if we re-organized the data paths and save the time for. for example, instruction fetching? We can use the data as soon as the last instruction is processed:



This way, we are connecting the data processing elements in series, accelerating our processing speed.

7.2 Our Goal

Our processor will have a 2-stage pipeline:

> **Instruction Fetch:** An instruction is fetched from the instruction memory.
> **Execute:** The instruction is decoded, executed, and committed (written back). This is a combination of the remaining stages of a normal five-stage RISC-V pipeline.

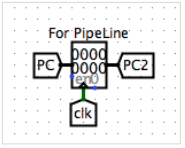
7.3 Realization

7.3.1 Intuitions

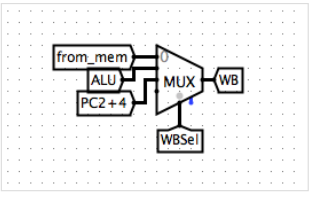
Basically the idea of pipelining is to allow each part of the data processing to be able to run separately, and make connection between the output of the last instruction to the input of the next one. Thus, as what's shown in the image above, we set some "stops" between sections, especially the `Execution` stage, which is the core in our connection of Datapath.

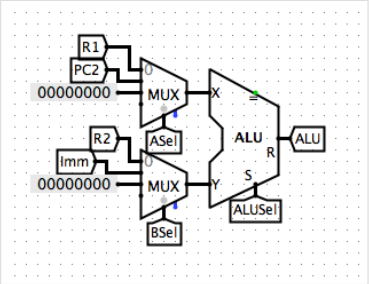
7.3.2 Execution Stage Clarify

Here my way is to introduce a new `PC2` for the connection between the `Decoding` and `Execution` and use the new `PC2` for later stages.

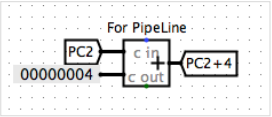


Thus, as a result, the following stages will take actions based on `PC2` such as the `ALU` (option in Input X)and our `Write Back` system.



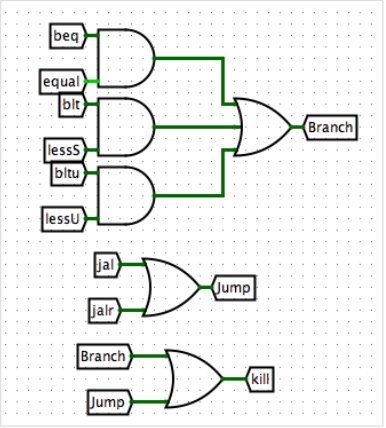


Note that the `PC2+4` , same as our `PC+4` , is define as

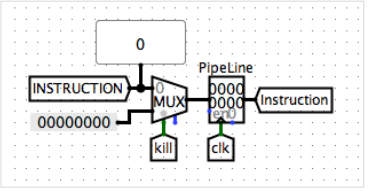


7.3.3 PC Organizing

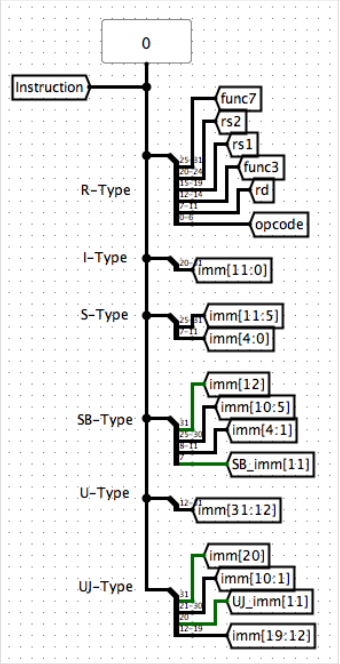
We still need to deal with a condition in our instruction fetching process. Think about if we actually want to jump to another address for next instruction such as `branches` and `jumps` . We should design a method to avoid the messing of `PC` . Thus, we develop a `kill` method in case of `branches` and `jumps` .



And thus, in our instruction fetching stage, we need to add a selection method for our instructions.



Note that we also need to change our instruction input to our instruction after selection for our `Decoding` stage.

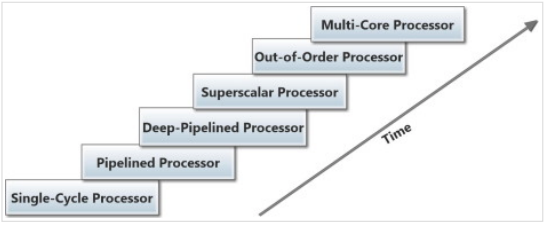


7.3.4 Hazard Avoiding

There are many hazards which we might be careful to. For example, the *Register Access Hazard* (Conflict when the next instruction is trying to fetch and the last instruction is trying to write back into a same register) and *ALU Result Hazard* (Reading from register in the next instruction while the result of the last instruction hasn't been written into the register yet, which results in data inconsistence). However, those hazards can be and should be avoided from the instructions designs. Here's just a brief mention.

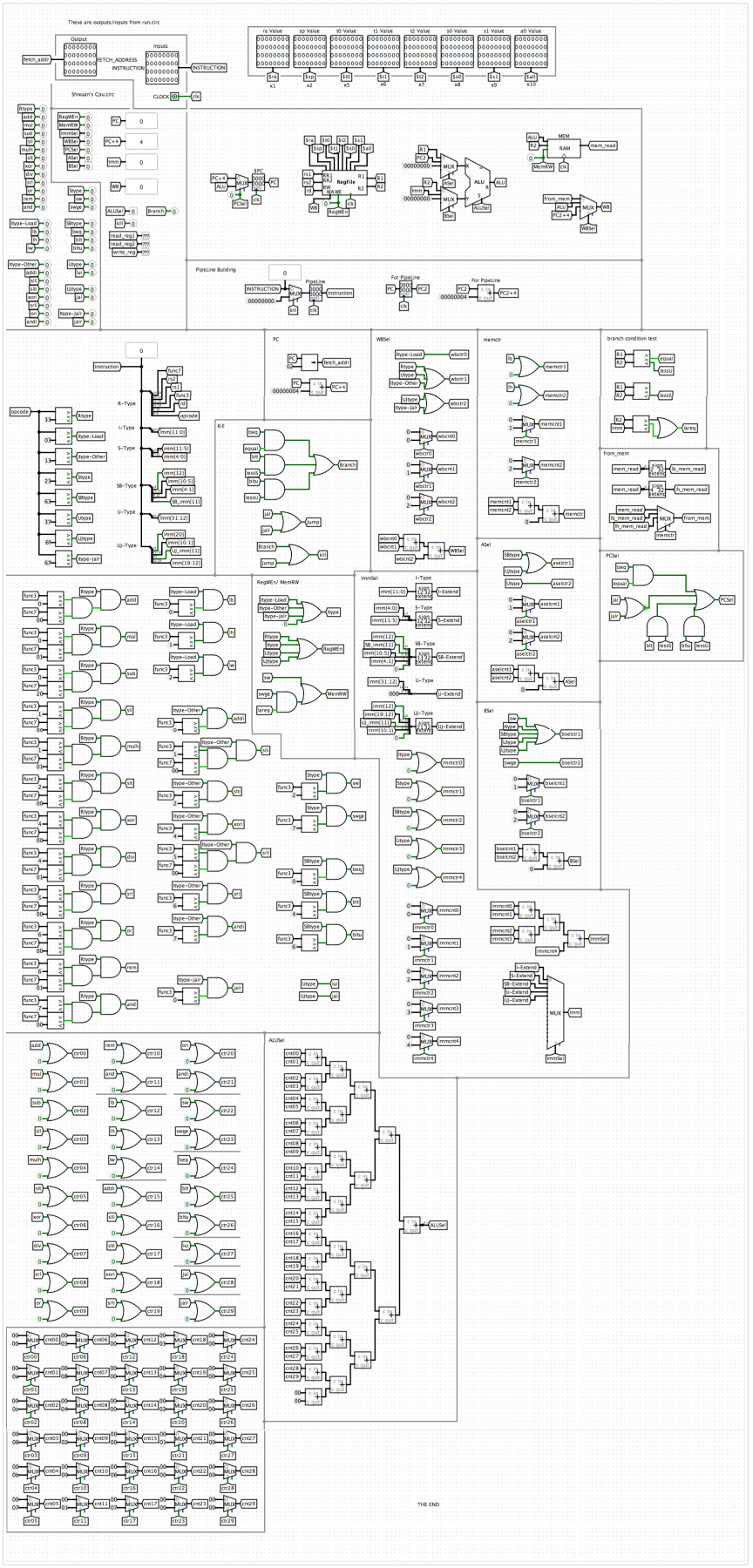
7.4 More on Pipelining

Congrats for reaching the end of the designing of Two-Cycle CPU! Now you might have some intuitions about pipelining and wondering if we have some ways to accelerate the CPU processing speed even more. Well, I'll briefly mention some higher level pipelining designs here. Try search for more information for your own interest~



7.5 Overview

Here is an `Overview` to our structure. I usually try to well-organize the structure of each sections so that it looks clean. Develop your own presentation~



Congratulations! You've finished your journey exploring the world inside a CPU. Feel free to develop our structure and develop more interesting CPU structure for your own interest~ Note that if you have any question, suggestions, or thoughts, feel free to comment below and I will respond as soon as possible~

Donate

UC_Berkeley RISC-V CPU ALU Register Logisim Computer_Architecture Project

RISC-V Based CPU Design with Logisim [Part 6]
CS 61A 学习手册 [Part 0] 简介

Like

Issue Page

No Comment Yet

Write Preview Login with GitHub

Leave a comment

Comment