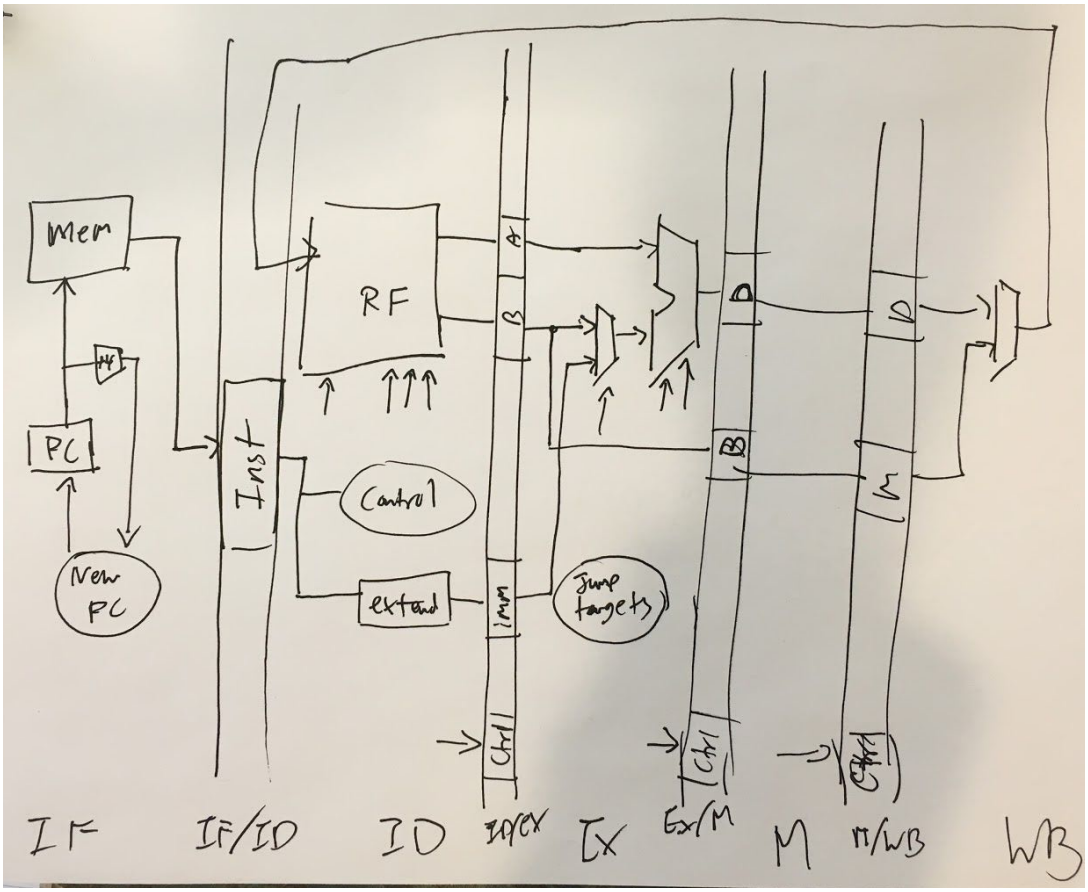
Design Documentation P2

Kevin Klaben (kek228) Jonathan Tong (jt572) February 28, 2019

1 Introduction

- **Purpose:** The purpose of this document is to act as a reference to the processor and explain our design choices
- **Scope:** The scope of this document is everything in a RISC-V pipelined processor that we've covered in class
- **Intended audience:** The intended audience is anyone with at least some basic knowledge of the concepts behind building a processor
- **Summary:** This will provide an overview of each of the components in our processor

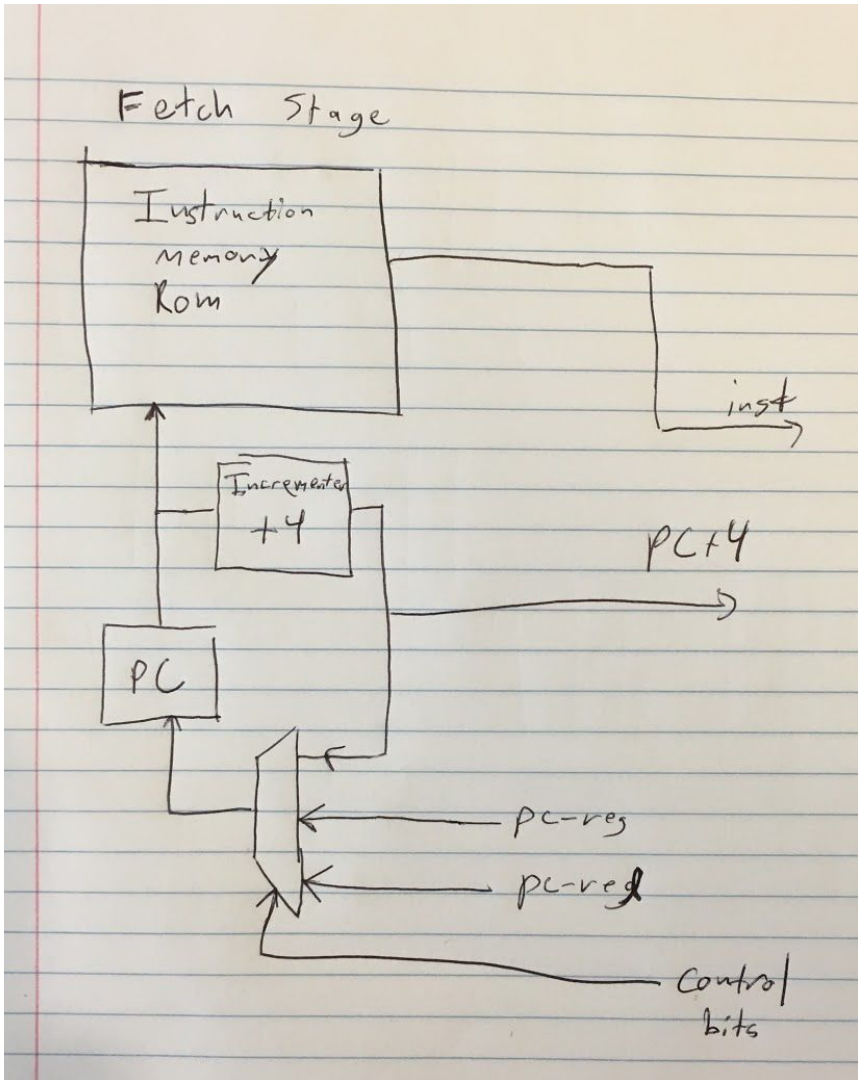
2 Overview



Provides an general overview of the functionality of the system and matters related to its design (including a discussion of the basic design approach and organization). You could split this up to multiple subsections.

3 The Fetch Stage

3.1 Circuit Diagram



Instruction Memory holds the instructions to be completed, the pc outputs the address of the next instruction to be completed. Then the instruction is taken from the rom address and output. The PC is incremented by 4 and then muxed with a pc-reg, and pc-rel based on the control bits of the instruction in the decode stage. The selected pc is the pc which is to be output on the next clock cycle.

3.1.1 Submodule incrementer for the PC

-To increment the PC by 4 each clock cycle

1

3.1.2 mux
Decides between pc+4, pc-rel, and pc-reg based on the control bits passed back after the decode stage.

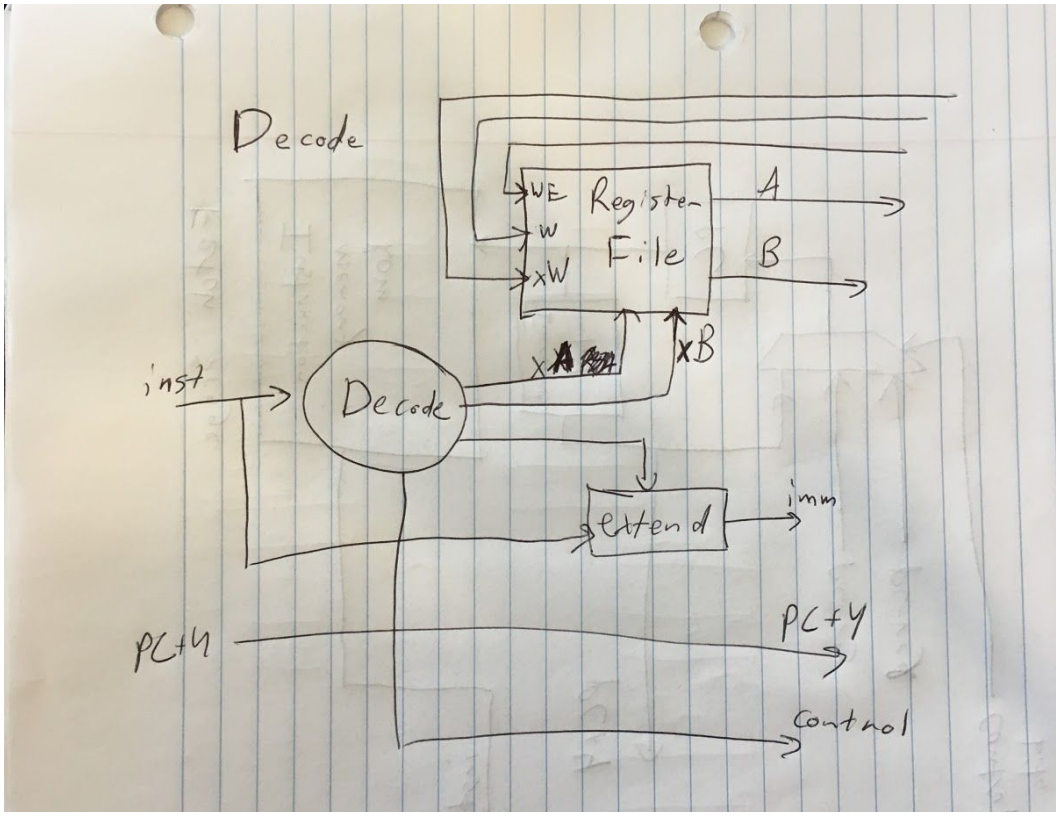
3.2 Correctness Constraints

Use PC to index Program Memory, increment PC

Fetch a new instruction every cycle
• Current PC is index to instruction memory
• Increment the PC at end of cycle (assume no branches for
now)
Write values of interest to pipeline register (IF/ID)• Instruction bits (for later decoding)
• PC+4 (for later computing branch targets)

3.3 Testing

4 The Decode Stage



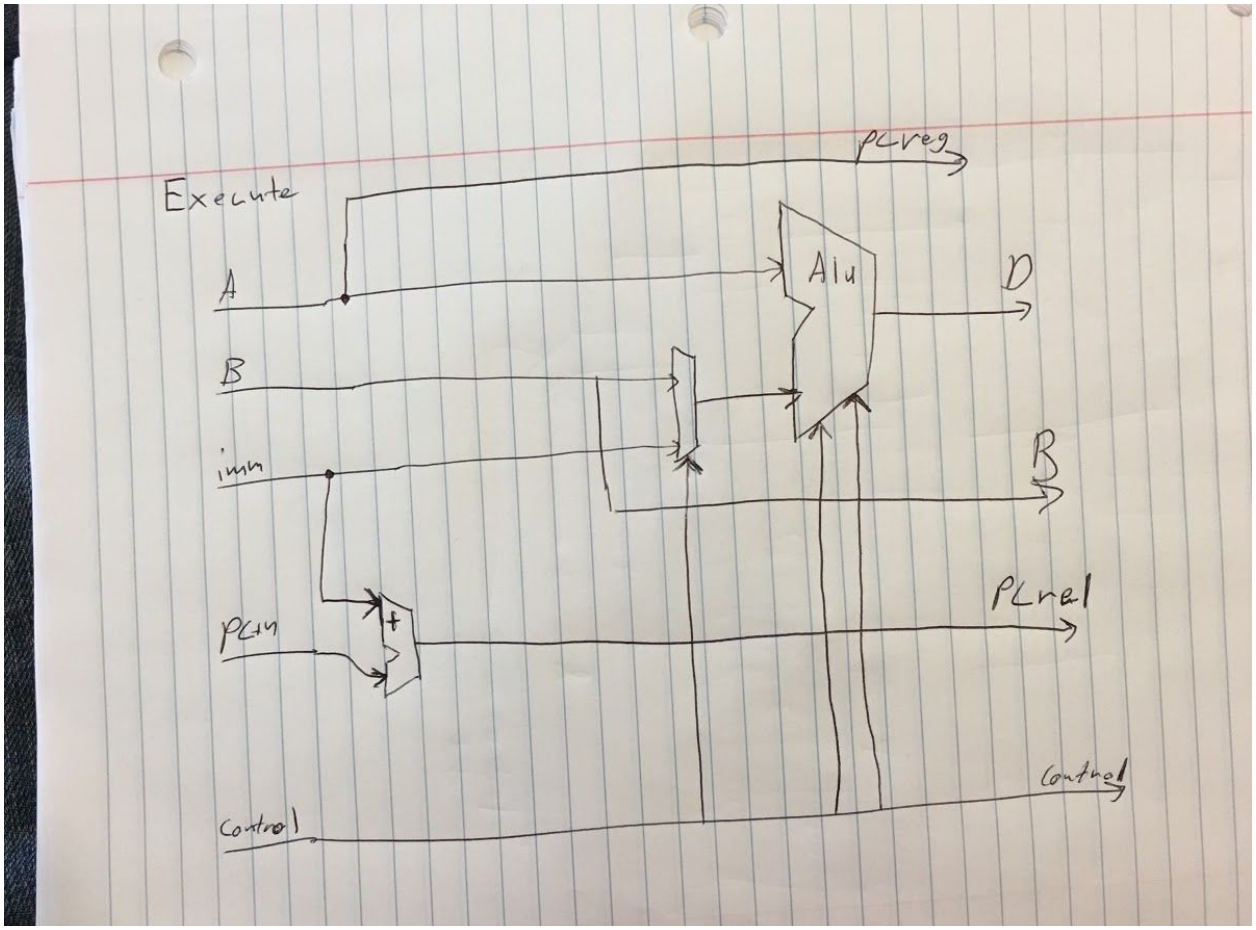
The decode stage takes in an instruction and the pc+4. It uses a splitter to break the 32 bit instruction into several pieces, including a register number for register A, register B, an immediate value, and others. For the immediate value, we sign extend it to 32 bits. For a given instruction, we will parse it into all the possible pieces and send it forward. The control bits will be used in a mux to select between them, i.e. use Register B or Imm when using the ALU.

4.2 Correctness Constraints

Decode instruction, generate control signals, read register file

• On every cycle:
• Read IF/ID pipeline register to get instruction bits • Decode instruction, generate control signals
• Read from register file
• Write values of interest to pipeline register (ID/EX)• Control information, Rd index, immediates, offsets, ...
• Contents of Ra, Rb
• PC+4 (for computing branch targets later)

5 The Execute Stage



The execute stage will be used to handle operations with the ALU. Given an immediate value, register A, register B, and the PC, the control bits will decide which of the inputs will go into the ALU.

5.2 Correctness Constraints

Perform ALU operation Compute targets (PC+4+offset, etc.) in case this is a branch, decide if branch taken

• On every cycle:
• Read ID/EX pipeline register to get values and control bits
• Perform ALU operation
• Compute targets (PC+4+offset, etc.) *in case* this is a branch • Decide if jump/branch should be taken
• Write values of interest to pipeline register (EX/MEM)• Control information, Rd index, ...

• Result of ALU operation
• Value *in case* this is a memory store instruction

6 The Memory Stage

6.2 Correctness Constraints

Perform load/store if needed, address is ALU result

In our case this memory stage does nothing and is simply a pass through.

7 The Writeback Stage

7.2 Correctness Constraints

Select value, write to register file

• On every cycle:

- Read MEM/WB pipeline register to get values and control
- bits
- Select value and write to register file

8 Testing

We plan on testing this design piece by piece with edge cases for each piece of the pipeline to ensure they each are working. As an overall test we plan on loading several instructions into the ROM and inspecting as the processor completes them what is being written to the register file to ensure that together the pipelined processor overall works.