**P2 Design Documentation**

Oscar So (ons4) Michael Rigney (mjr372)

**1 Introduction**

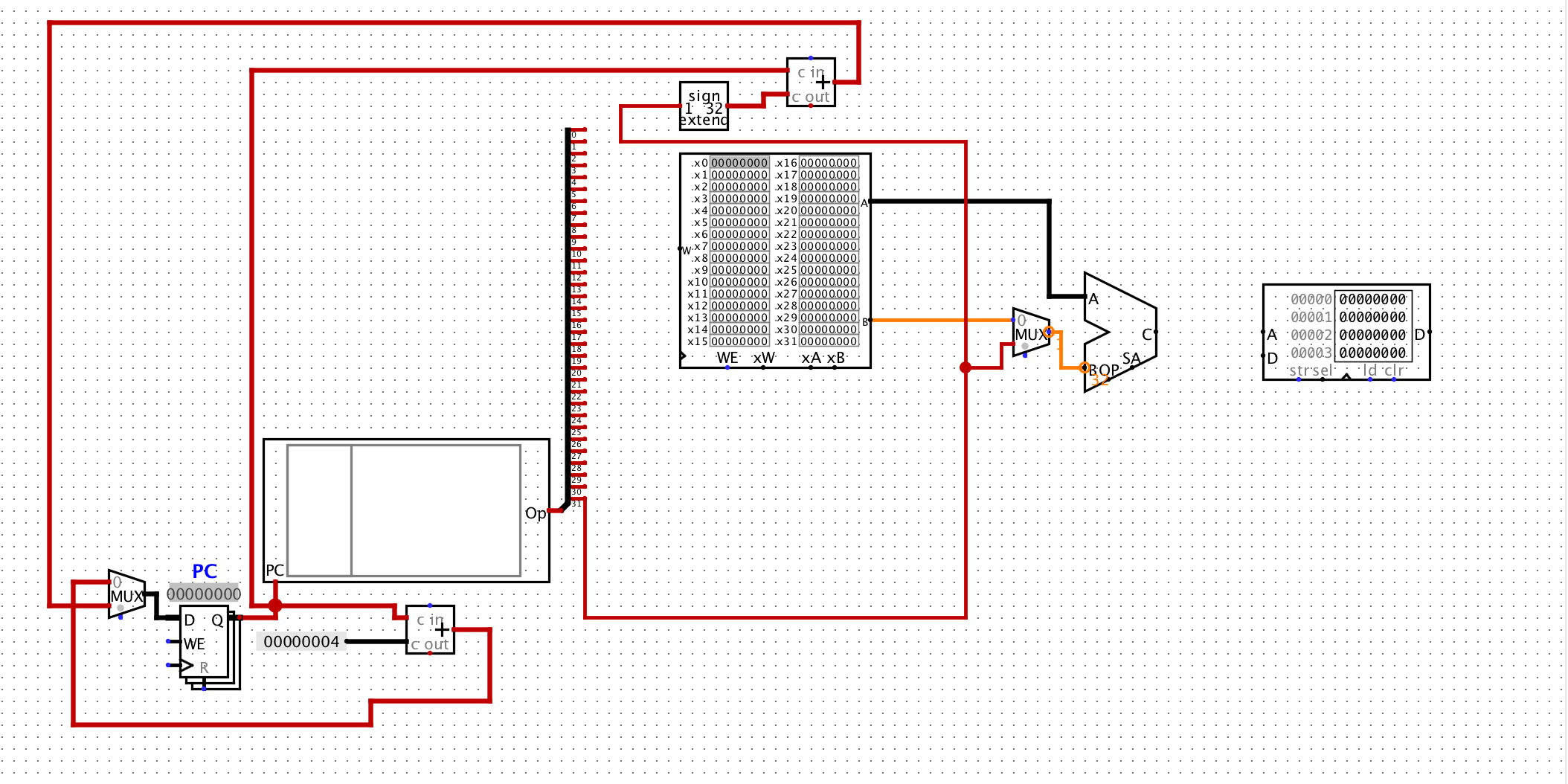
The purpose of this document is to understand our design of pipelining a RISC-V CPU on logism. The document will cover an overview of the document, as well as descriptions for each stage in the CPU and also include sub-circuits and testing. This document should provide people, with knowledge of circuits, who view our design a thorough understanding of each stage component.

• Provide references for any other documents, such as textbook, reference manual, etc.

• Define any important terms, acronyms, or abbreviations.

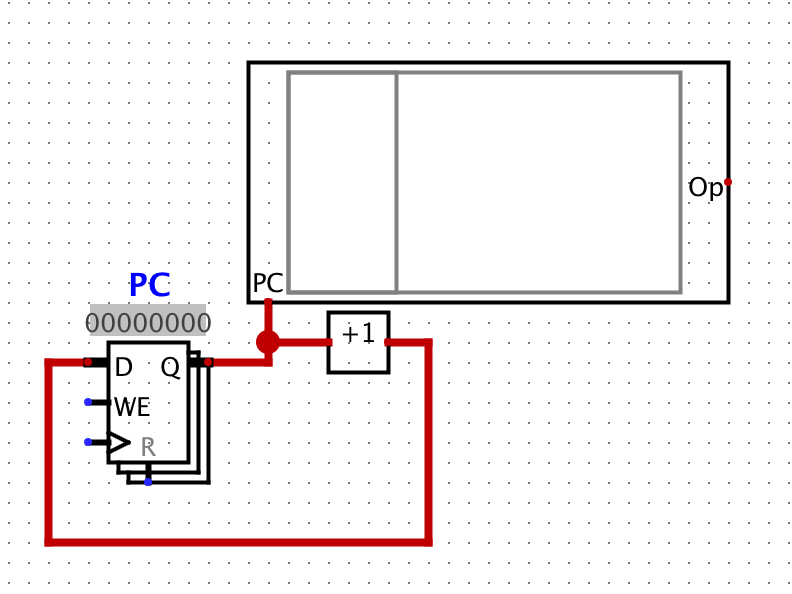
**2 Overview**

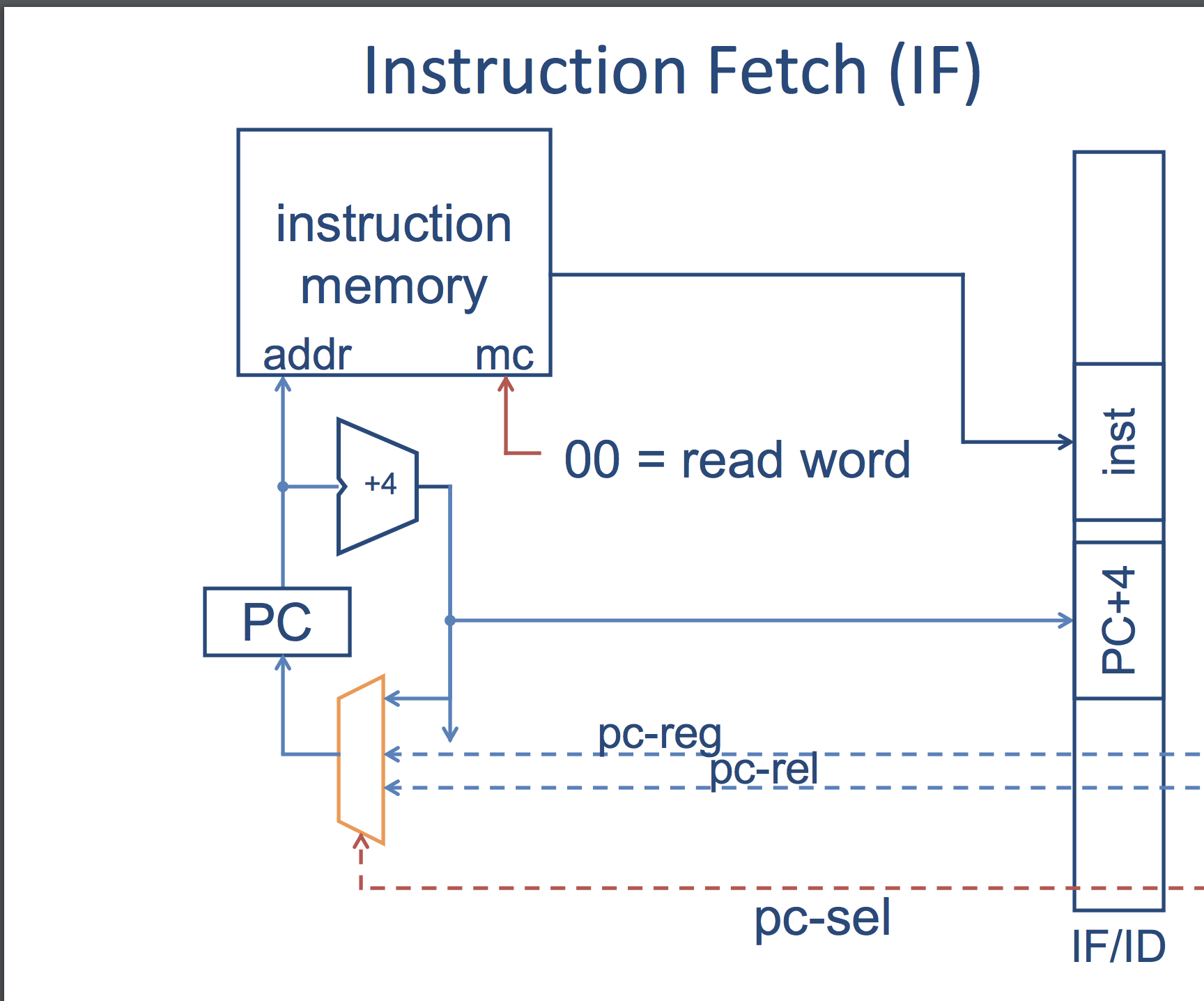
The RISC-V CPU that we will be working on is a 5-staged pipelined CPU that takes in operations and executes it to be written in the memory. In simple words, this is the backbone behind simple codes like logical and arithmetic statements.

****

**3 The Fetch Stage**

**3.1 Circuit Diagram**





**3.1.1 Program Counter (PC)**

Register holding the address of the current instruction in instruction memory. Each instruction is 32 bits.

**3.1.2 IF/ID Pipeline Register**

Takes in the 32 bit instruction to be sent to the decoder, as well as the PC + 4 for the address of the next instruction to be fed back into instruction memory.

3.2 Correctness Constraints

State the functional requirement of this module.

• item 1

• item 2

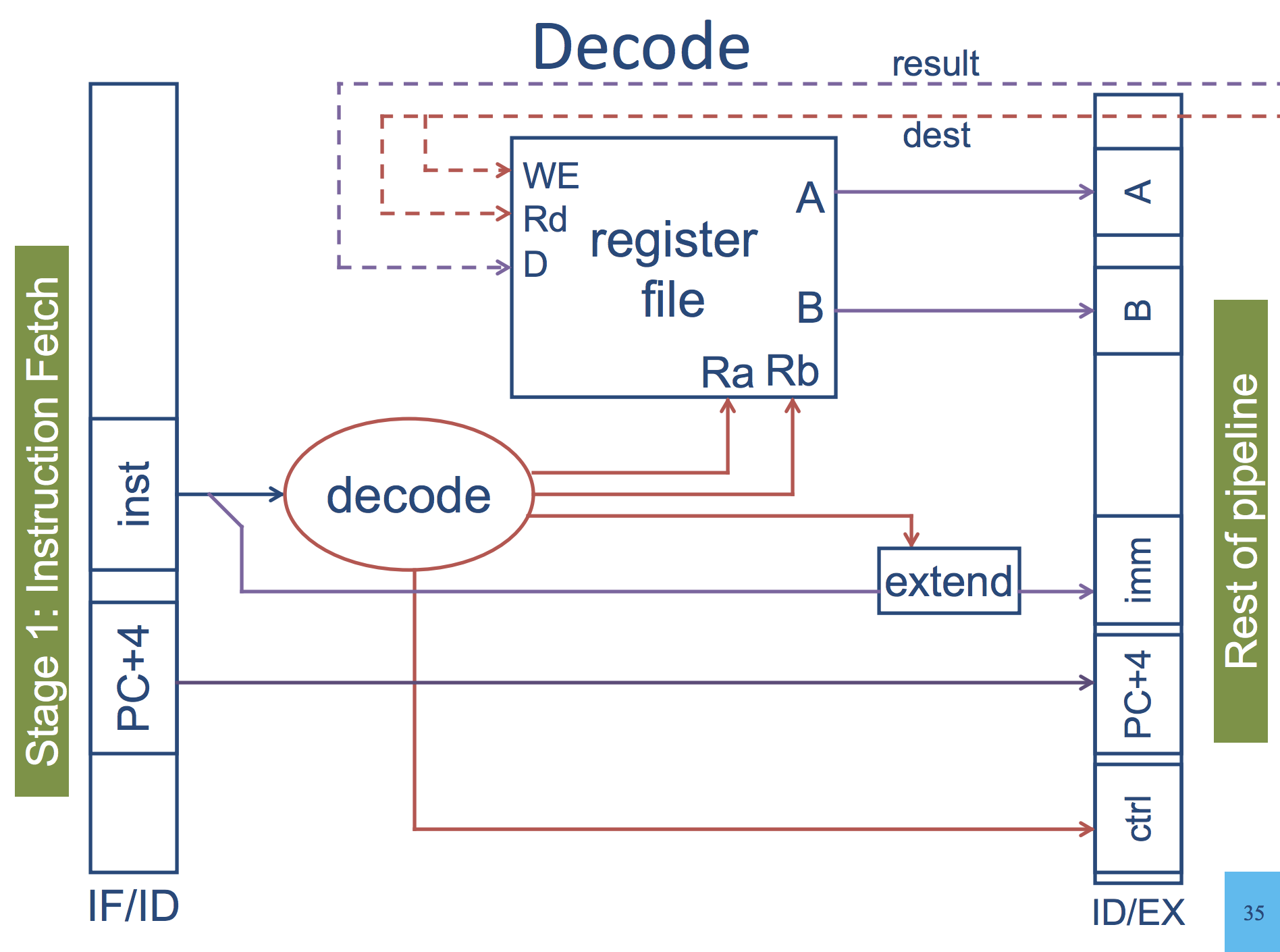
• item 3

3.3 Testing

State the approach to verify the functional correctness of this module.

**4 The Decode Stage**

**4.1 Circuit Diagram**



**4.1 Control Unit**

Take in the instruction bits from IF/ID pipeline register and route the read registers and write register to the Register file. Route the operation to the ALU.

**4.1.1 Register File**

Contains registers with values to be accessed and operated on per the instructions. Stores write back results of operations in registers as well. Outputs A and B 32-bit values to the ID/EX pipeline to be used in the ALU.

**4.1.2 ID/EX Pipeline Register**

Takes in control information for the execution stage (A,B, ctrl) as well as immediates and offsets. Also takes in PC + 4 for computing branch targets later on

4.2 Correctness Constraints

State the functional requirement of this module.

• item 1

• item 2

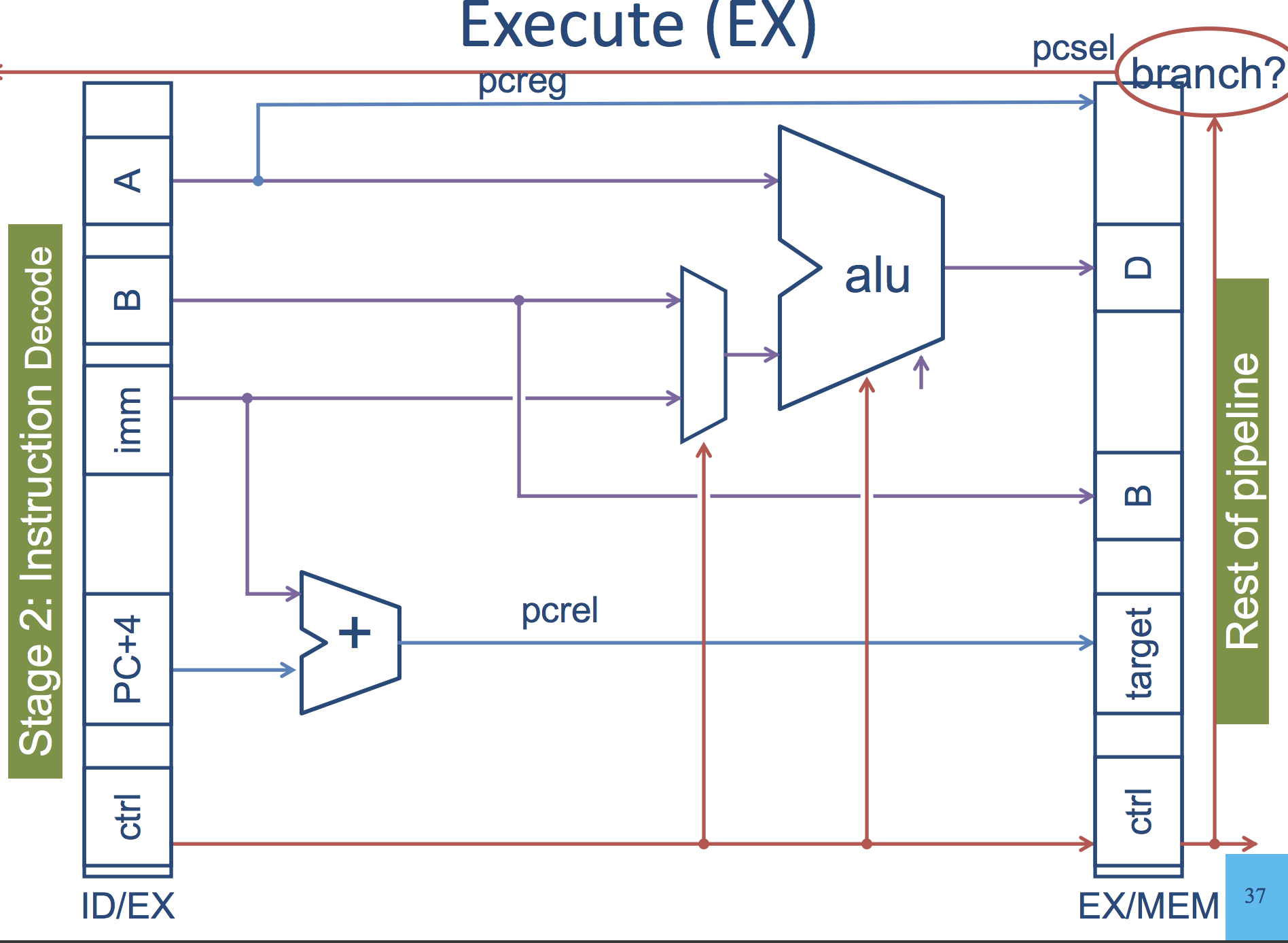
• item 3

4.3 Testing

State the approach to verify the functional correctness of this module.

**5 The Execute Stage**

**5.1 Circuit Diagram**

****

**5.1.1 Arithmetic Logic Unit (ALU)**

Takes A, B (32 bits each), and an operation code and outputs the computed result into the EX/MEM pipeline

**5.1.2 EX/MEM Pipeline**

Takes in the output of the ALU, PC + 4 and the past instruction index, and B for memory storage

**5.2 Correctness Constraints**

State the functional requirement of this module.

• item 1

• item 2

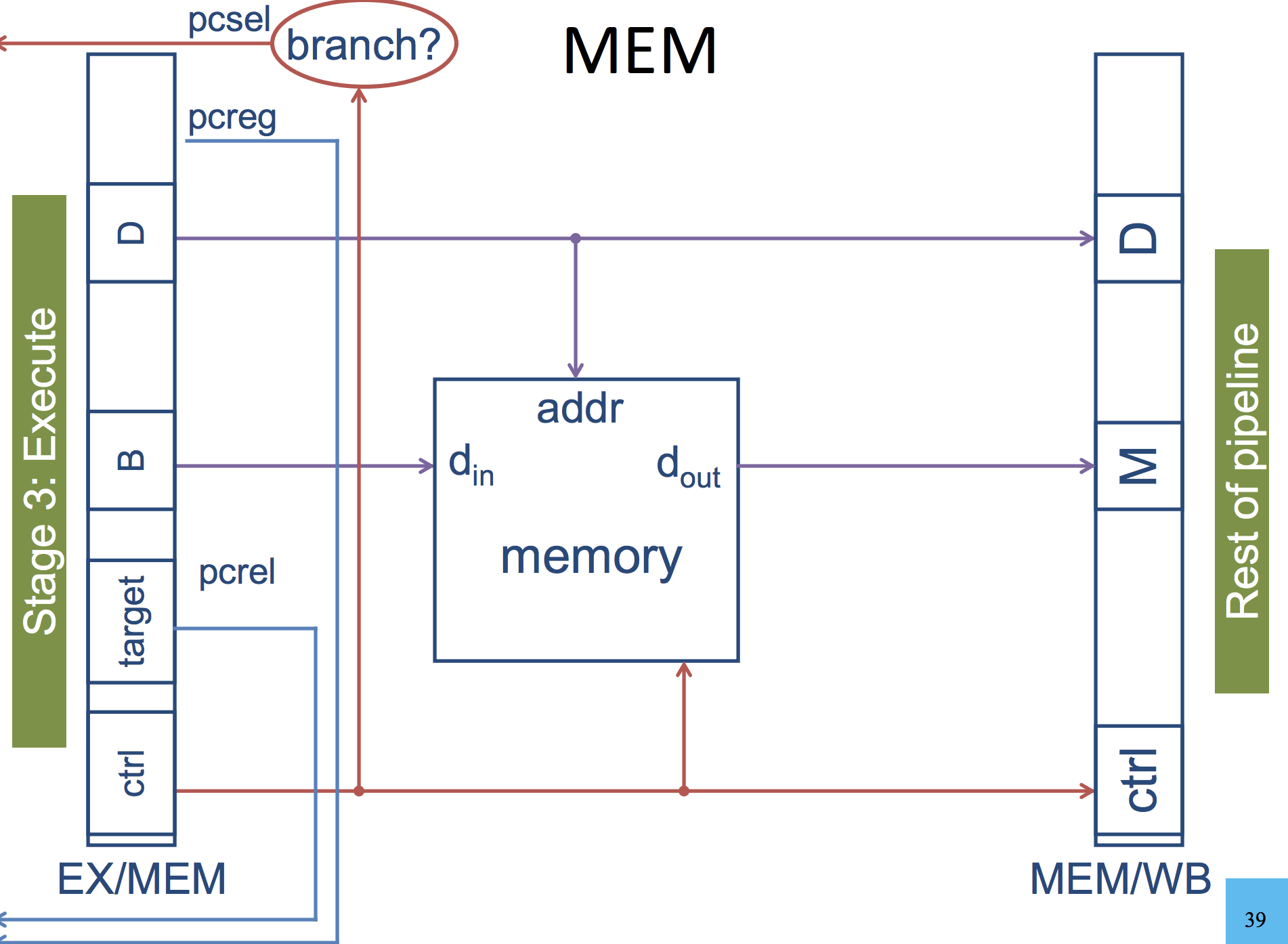
• item 3

5.3 Testing

State the approach to verify the functional correctness of this module.

**6 The Memory Stage**

6.1 Circuit Diagram



6.2 Correctness Constraints

State the functional requirement of this module.

• item 1

• item 2

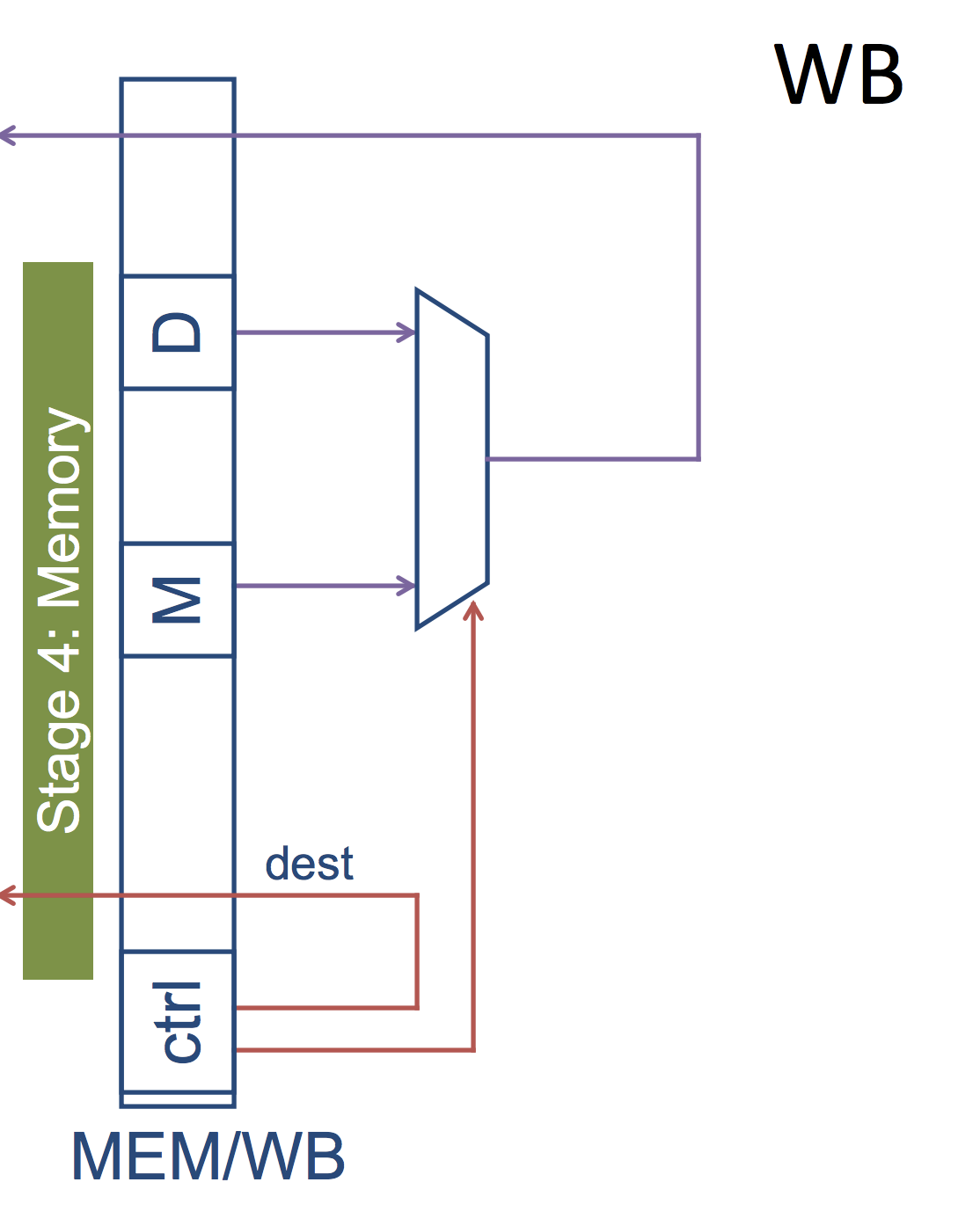
• item 3

6.3 Testing

State the approach to verify the functional correctness of this module.

**7 The Writeback Stage**

7.1 Circuit Diagram



**7.1.1 MEM/WB Pipeline Register**

Takes in the ALU operation result, result of memory operation, and control information for next instruction register index. Sends ALU result to write back to register file and updates PC in the event of a branch or jump.

7.2 Correctness Constraints

State the functional requirement of this module.

• item 1

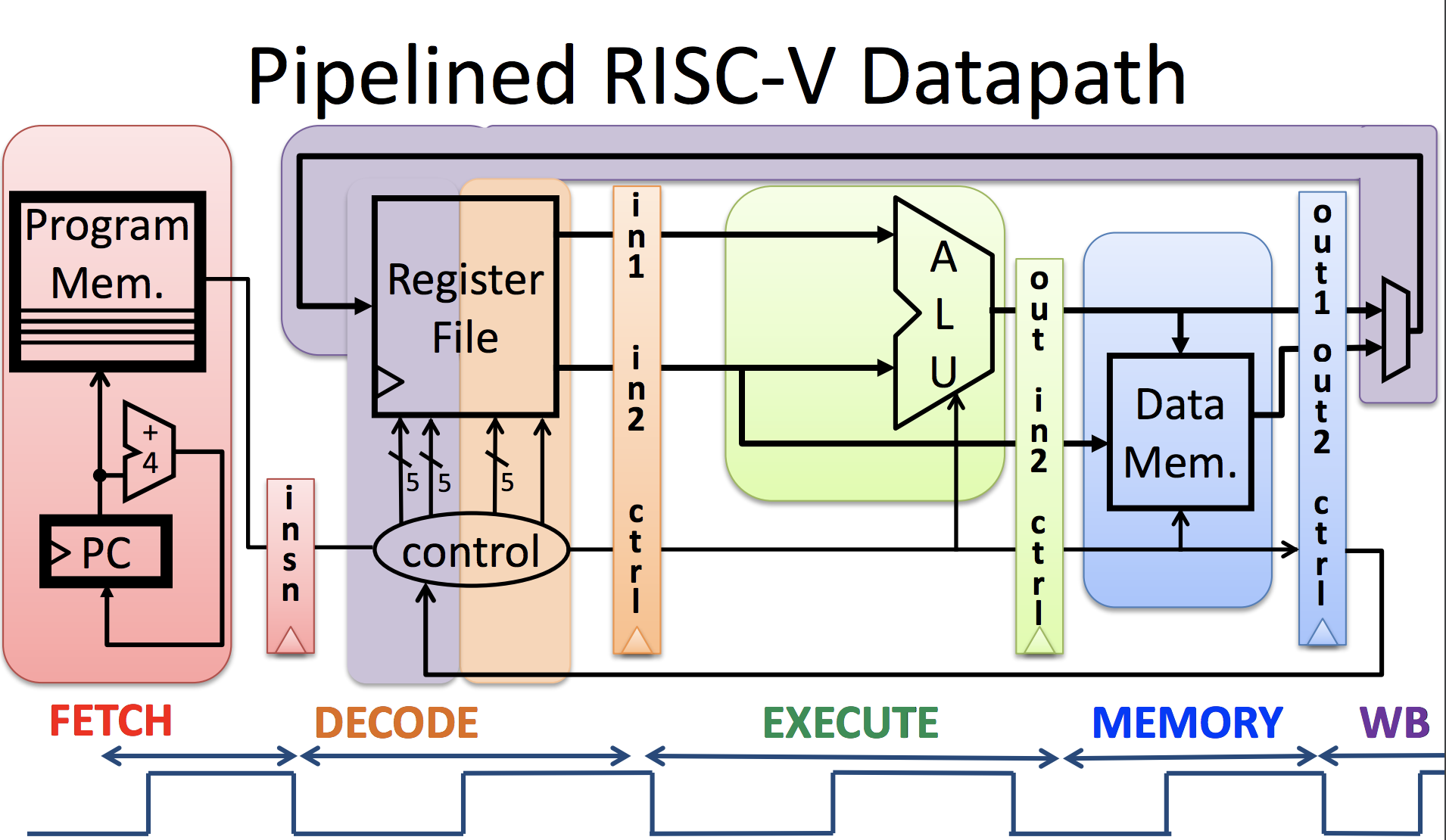
• item 2

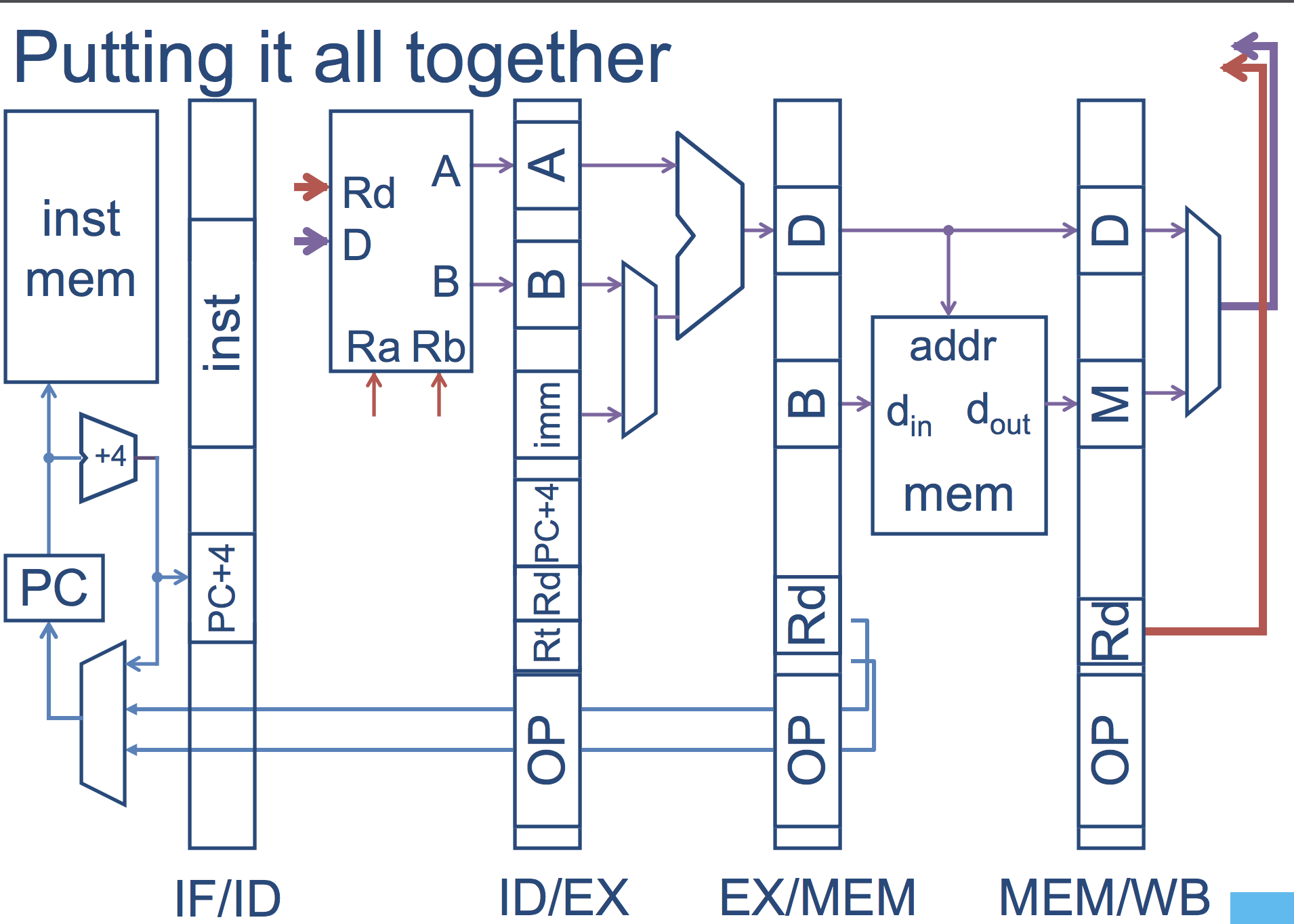
• item 3

7.3 Testing

State the approach to verify the functional correctness of this module.

**8 Overall**

****

****